25 April 2002 revision 0

12 May 2002 revision 1

The Serial Attached SCSI working group (http://www.serialattachedscsi.com) is pleased to provide the INCITS T10 Technical Committee this proposed first working draft of the Serial Attached SCSI standard. T10/02-098r0 contains the project proposal.

The Serial Attached SCSI working group was formed in late 2001 to lead the industry in defining a new serial point-to-point enterprise device attach interface, leveraging the Serial ATA physical layer and the SCSI protocol set to support a wide range of price/performance storage products. Serial Attached SCSI extends the capabilities of the Serial ATA interface with proven SCSI robustness and versatility. Serial Attached SCSI also extends the legacy of parallel SCSI by preserving 20 years of enterprise-proven software, providing an unprecedented range of customer choice, and expanding the capability of this device level interface well into the future.

The SCSI Trade Association (http://www.scsita.org) is supporting the development and promotion of this technology.

The members of the Serial Attached SCSI working group are:

Adaptec	I-TECH
Amphenol	KnowledgeTek
Compaq	LSI Logic
Crossroads Systems	Marvell
Cypress Semiconductor	Maxtor
Data Transit	Molex
Dell	NEC Electronics
Eurologic Systems	QLogic
FCI USA	Seagate
Fujitsu Limited	ServerWorks (Broadcom)
Hewlett-Packard	Sierra Logic
Hitachi America	Silicon Image
IBM	Western Digital

This page is (almost) blank.

Proposed Working Draft

T10
Project xxxx-D

Revision pre1
12 May 2002

Information Technology - Serial Attached SCSI (SAS)

T10 Technical Editor:

Robert C Elliott
MC 150801
Compaq Computer Corporation
P.O. Box 692000
Houston, TX 77269-2000
USA

Telephone: (281) 518-5037
Email: Robert.Elliott@Compaq.com

Reference number
ISO/IEC xxxxx-xxx:200x
ANSI INCITS.xxx:200x

Points of Contact:

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

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

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

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

INCITS T10 Committee
Web Site: http://www.t10.org

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

International Committee for Information Technology Standards (INCITS) Secretariat
Suite 200
1250 Eye Street, NW
Washington, DC 20005
USA

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

Information Technology Industry Council
Web site: http://www.itic.org

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

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

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

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

Revision history

R.1 Revision 02-157r0 (25 April 2002):

First release as T10 proposal 02-157r0.

R.2 Revision 02-157r1 (12 May 2002):

Released as T10 proposal 02-157r1. Incorporates:

- Editorial corrections from 02-183r0 Minutes from the SAS PHY study group 1-2 May 2002
- Made the Port layer a separate top-level section
- 02-168r1 SAS ALIGN primitives
- 02-165r1 SAS SMP Report Manufacturer Info
- 02-167r1 SAS ERROR primitive
- Miscellaneous editorial and technical corrections from the SAS protocol study group 29-30 April 2002

American National Standard
for Information Technology

Serial Attached SCSI (SAS)

Secretariat

International Committee for Information Technology Standards

Approved mm dd yyyy

American National Standards Institute, Inc.

Abstract

This standard specifies the functional requirements for the Serial Attached SCSI (SAS) protocol, which defines transmission of SCSI protocol over a Serial ATA compatible physical layer and defines addressing of multiple target devices for the Serial ATA protocol.

American National Standard

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

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

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

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

CAUTION: The developers of this standard have requested that holders of patents that may be required for the implementation of the standard, disclose such patents to the publisher. However, neither the developers nor the publisher have undertaken a patent search in order to identify which, if any, patents may apply to this standard. As of the date of publication of this standard, following calls for the identification of patents that may be required for the implementation of the standard, no such claims have been made. No further patent search is conducted by the developer or the publisher in respect to any standard it processes. No representation is made or implied that licenses are not required to avoid infringement in the use of this standard.

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

Contents

Tables

Figures

1 Scope

The SCSI family of standards provides for many different transport protocols that define the rules for exchanging information between different SCSI devices. This standard defines the rules for exchanging information between SCSI devices using a Serial ATA-compatible physical layer. Other SCSI transport protocol standards define the rules for exchanging information between SCSI devices using other interconnects.

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

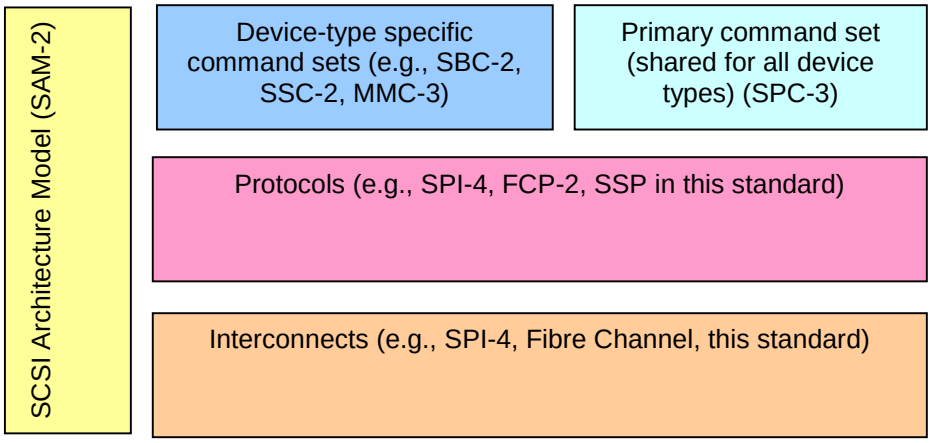

Figure 1. SCSI document relationships

This standard also defines the rules for exchanging information between Serial ATA-compatible initiators and Serial ATA targets. Figure 2 shows the relationship of this standard to other standards and related projects in the ATA family of standards.

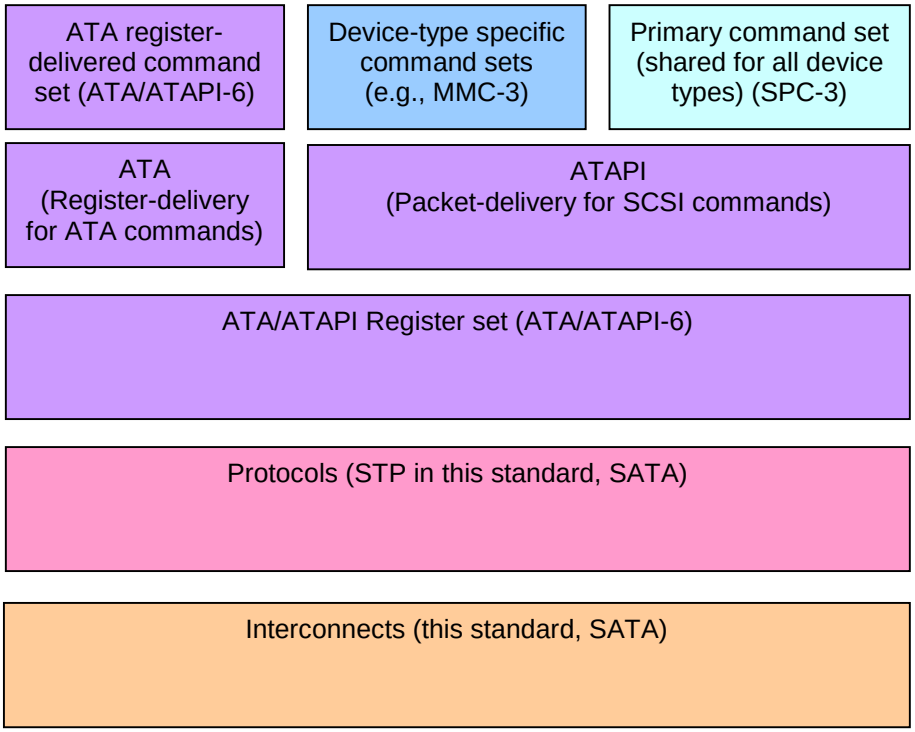

Figure 2. ATA document relationships

Figure 1 and Figure 2 are intended to show the general relationship of the documents to one another, and are not intended to 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 References

2.1 References overview

The following standards contain provisions that, by reference in the text, constitute provisions of this standard.

At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards listed below.

Copies of the following documents may be obtained from ANSI:

a) approved ANSI standards;
b) approved and draft international and regional standards (ISO, IEC, CEN/CENELEC, ITUT); and
c) approved and draft foreign standards (including BSI, JIS, and DIN).

For further information, contact ANSI Customer Service Department at +1.212.642.4900 (telephone) or via the World Wide Web at http://www.ansi.org.

Additional availability contact information is provided below as needed.

2.2 Normative references

AT Attachment with Packet Interface-6 (ATA/ATAPI-6). Latest revision. Available from http://www.t13.org.

2.3 References under development

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

AT Attachment with Packet Interface-7 (ATA/ATAPI-7). Latest revision. Available from http://www.t13.org.

SCSI Architecture Model-2 (SAM-2). Latest revision. Available from http://www.t10.org.

SCSI Primary Commands-3 (SPC-3). Latest revision. Available from http://www.t10.org.

For more information on the current status of these documents, contact the INCITS Secretariat at +1.202.737.8888 (telephone) or via Email at incits@itic.org. To obtain copies of these documents, contact Global Engineering at 15 Inverness Way, East Englewood, CO 80112-5704 at +1.303.792.2181 (telephone) or +1.800.854.7179 (telephone).

2.4 Other references

Serial ATA: High Speed Serialized AT Attachment (SATA). Revision 1.0, 29-August-2001. APT Technologies, Inc., Dell Computer Corporation, IBM Corporation, Intel Corporation, Maxtor Corporation, and Seagate Technology. Available from http://www.serialata.org.

[Editor's note: include Serial ATA errata and design guides once they are publicly available.]

SFF-8410 Specification for HSS Copper Testing and Performance Requirements. Rev 16.1, 20 March 2000. Available from http://www.sffcommittee.org.

~~*SFF-TBD*~~*SFF-8482 Specification for SAS Plug Connector*. Available from http://www.sffcommittee.org.

~~*SFF-TBDE*~~*SFF-8483 Specification for SAS Multi Lane Copper Connector*. Available from http://www.sffcommittee.org.

3 Definitions, symbols, abbreviations, keywords, and conventions

3.1 Definitions

3.1.1. **attach**: Physically attach.

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

3.1.3. **ATA domain**: An I/O system consisting of a set of ATA devices that communicate with one another by means of a service delivery subsystem.

3.1.4. **ATA device**: A device that contains one or more ATA ports that are connected to a service delivery subsystem and supports an ATA application protocol.

3.1.5. **ATA initiator device**: An ATA device containing application clients and ATA initiator ports that originate device service and task management requests to be processed by an ATA target device. Called a “host system” in ATA.

3.1.6. **ATA initiator port**: An ATA initiator device object that acts as the connection between application clients and the service delivery subsystem through which requests and responses are routed. Called a “host adapter” in ATA.

3.1.7. **ATA port**: An ATA initiator port or an ATA target port.

3.1.8. **ATA target device**: An ATA device containing logical units and ATA target ports that receives device service and task management requests for processing. Called a “device” in ATA.

3.1.9. **ATA target port**: An ATA target device object that contains a task router and acts as the connection between device servers and task managers and the service delivery subsystem through which requests and responses are routed. Called a “device” in ATA.

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

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

3.1.12. **big-endian**: A format for transmission or storage of binary data in which the most significant byte appears first. In a multi-byte value, the byte containing the MSB is stored in the lowest address and the byte containing the LSB is stored in the highest address.

3.1.13. **bit**: A character used to represent one of the two digits in the numeration system with a base of two, and only two, possible states of a physical entity or system.

3.1.14. **bit synchronization**: Detection of an incoming stream of bits from a physical link by a phy.

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

3.1.16. **character**: A sequence of 10 contiguous bits considered as a unit. A byte is encoded as a character using 8b10b coding.

3.1.17. **character synchronization**: Detection of an incoming stream of characters from a physical link by a phy.

3.1.18. **connection**: An association between an initiator port and a target port that begins with an open and normally ends with a close. During a connection all transmitted dwords are associated with the I_T nexus formed by that initiator port and target port.

3.1.19. **checksum**: The value computed on data to detect error or manipulation during transmission.

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

3.1.21. **DC**: The non-AC component of a signal. In this standard, means all frequency components below 100 kHz.

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

3.1.23. **device**: A physical entity.

3.1.24. **device name**: A worldwide name assigned to an initiator device, target device, or expander device.

3.1.25. **dword**: A sequence of 4 contiguous bytes or 4 contiguous characters considered as a unit.

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

3.1.27. **Dxx.y**: Data character. A character in 8b10b coding representing a byte of data.

3.1.28. **edge expander device**: An expander device with simple routing capabilities.

3.1.29. **expander**: See expander device.

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

3.1.31. **expander port**: A SAS expander device object that routes SSP, SMP, and STP frames to and from logical links. Contains one or more phys.

3.1.32. **fanout expander device**: An expander device with complex routing capabilities.

3.1.33. **field**: A group of one or more contiguous bits.

3.1.34. **frame:** A sequence of dwords between a start of frame primitive and an end of frame primitive.

3.1.35. **frame information structure (FIS)**: The SATA frame format.

3.1.36. **hard reset**: A SAS device action in response to a reset event in which a SAS device performs the operations described in 4.6.

3.1.37. **hash function**: A mathematical function that maps values from a large domain into a smaller range, and that reduces a long value into a hash value.

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

3.1.39. **I_T_L nexus**: A nexus that exists between a SCSI initiator port, a SCSI target port, and a logical unit. This relationship replaces the prior I_T nexus.

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

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

3.1.42. **initiator**: See initiator port.

3.1.43. **initiator device:** See SAS initiator device.

3.1.44. **initiator port:** See SAS initiator port.

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

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

3.1.47. **Kxx.y**: control character. A character in 8b10b coding that does not represent a byte of data.

3.1.48. **least significant bit (LSB):** In a binary code, the bit or bit position that is assigned to, or represents, the smallest quantity or increment that can be represented by the code.

3.1.49. **link**: A logical link, physical link or a wide physical link.

3.1.50. **link reset**: Performing the link reset sequence.

3.1.51. **link reset sequence**: Performing OOB signal exchange and speed negotiation.

3.1.52. **little-endian**: A format for transmission or storage of binary data in which the least significant byte appears first. In a multi-byte value, the byte containing the LSB is stored in the lowest address and the byte containing the MSB is stored in the highest address.

3.1.53. **logical link**: A time-division multiplexed portion of a physical link. The multiplexing divisor may be 1.

3.1.54. **loss of signal**: The input signal to the receiver is below the minimum amplitude.

3.1.55. **most significant bit (MSB):** In a binary code, the bit or bit position that is assigned to, or represents, the largest quantity or increment that can be represented by the code.

3.1.56. **nexus**: A relationship between a SCSI initiator port and a SCSI target port that may extend to a logical unit and a queue tag.

3.1.57. **object**: An architectural abstraction or container that encapsulates data types, services, or other objects that are related in some way. See SAM-2.

3.1.58. **out-of-band (OOB) signal**: Pattern of ALIGN primitives and idle time used during the link reset sequence.

3.1.59. **partial pathway**: The set of physical links participating in a connection request which has not reached a SAS endpoint (the connection request has been sent by the source device and confirmed as received by at least one expander device with AIP).

3.1.60. **pathway**: A set of physical links between a SAS initiator port and a SAS target port.

3.1.61. **phy**: An object that is part of a device and is the part of a port which interfaces to a service delivery subsystem.

3.1.62. **physical link**: Two differential signal pairs, one pair in each direction, that connect two phys.

3.1.63. **port:** An initiator port, target port, or an expander port, each of which contains one or more phys.

3.1.64. **power on**: Power being applied.

3.1.65. **primitive:** Dword starting with K28.5 or K28.3 followed by 3 characters.

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

3.1.67. **rate:** Data transfer rate of a logical link or physical link: 1,5 Gbps or 3,0 Gbps.

3.1.68. **reset event:** An event that triggers a hard reset from a SAS device. The only reset event defined in this standard is receiving a RESET primitive sequence.

3.1.69. **rho**: The reflection coefficient of a the transmission media, that is, the ratio of the voltage applied to the transmission media divided by the reflected voltage.

3.1.70. **SAS device**: A SCSI device, an ATA device, or an expander device in a SAS domain.

3.1.71. **SAS domain:** The I/O system defined by this standard that may serve as an ATA domain and/or a SCSI domain.

3.1.72. **SAS initiator device**: A SCSI initiator device or an ATA initiator device in a SAS domain.

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

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

3.1.75. **SAS target device**: A SCSI target device or an ATA target device in a SAS domain.

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

3.1.77. **SAS target/initiator device**: A SCSI target/initiator device or an ATA target/initiator device in a SAS domain.

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

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

3.1.80. **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-2.

3.1.81. **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-2.

3.1.82. **SCSI initiator device:** A SCSI device containing application clients and SCSI initiator ports that originate device service and task management requests to be processed by a SCSI target device. See SAM-2.

3.1.83. **SCSI initiator port:** A SCSI initiator device object that acts as the connection between application clients and the service delivery subsystem through which requests and responses are routed. See SAM-2.

3.1.84. **SCSI port**: A SCSI initiator port or a SCSI target port. See SAM-2.

3.1.85. **SCSI target device:** A SCSI device containing logical units and SCSI target ports that receives device service and task management requests for processing. See SAM-2.

3.1.86. **SCSI target port:** A SCSI target device object that contains a task router and acts as the connection between device servers and task managers and the service delivery subsystem through which requests and responses are routed. See SAM-2.

3.1.87. **SCSI target/initiator device**: A device that has all the characteristics of a SCSI target device and a SCSI initiator device. See SAM-2.

3.1.88. **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-2.

3.1.89. **scrambling**: Modifying data by XORing each bit with a pattern generated by a linear feedback shift register to cause constant 0 and constant 1 patterns to be changed into more varied patterns.

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

3.1.91. **Serial ATA (SATA):** The protocol defined by SATA.

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

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

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

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

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

3.1.97. **SATA target port:** An ATA target port in a SAS domain or a SATA domain. Contains one phy.

3.1.98. **SSP initiator port**: A SCSI initiator port in a SAS domain. Contains one or more phys.

3.1.99. **SSP target port**: A SCSI target port in a SAS domain. Contains one or more phys.

3.1.100. **STP initiator port**: An ATA initiator port in a SAS domain. Contains one or more phys.

3.1.101. **target:** See target port.

3.1.102. **target device:** See SAS target device.

3.1.103. **target port:** See SAS target port.

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

3.1.105. **task manager:** An agent within the device server that executes task management functions (see SAM-2).

3.1.106. **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-2).

3.1.107. **total jitter**: Measured jitter including deterministic jitter and random jitter.

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

3.1.109. **valid dword**: A primitive or a dword with four legal characters and valid running disparity.

3.1.110. **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.

3.1.111. **wide link:** A group of physical links that attaches a wide port to another wide port.

3.1.112. **wide port:** A port that contains more than one phy.

[Editor's note: ensure there are no "expander", "target", or "initiator" only uses so those terms can be deleted. Note that "expander" aliases to a device while "initiator" and "target" alias to ports.]

3.2 Symbols and abbreviations

Symbols and abbreviations used in this standard include:

ACK	acknowledge
AIP	arbitration in progress
ANSI	American National Standards Institute (http://www.ansi.org)
ATA	AT attachment
ATAPI	AT attachment packet interface
AWG	American wire gauge
BER	bit error rate
BSI	British Standards Institution (http://www.bsi-global.com)
CDB	command descriptor block
CEN	European Committee for Standardization (http://www.cenorm.be)
CENELE	European Committee for Electrotechnical Standardization (http://www.cenelec.org)
CRC	cyclic redundancy check
dB	decibel
DIN	German Institute for Standardization (http://www.din.de)
DJ	deterministic jitter
FIS	frame information structure
Gbps	gigabits per second (10^9 bits per second)
IEC	International Engineering Consortium (http://www.iec.ch)
IEEE	Institute of Electrical and Electronics Engineers (http://www.ieee.org)
ISO	International Standards Organization (http://www.iso.ch)
ITI	Information Technology Industry Council (http://www.itic.org)
ITUT	International Telecommunications Union Telecommunications Standardization Sector (http://www.itu.int)
IU	information unit
JIS	Japanese Industrial Standards Committee (http://www.jisc.org)
kHz	kilohertz
LED	light-emitting diode
LLP	lower level protocol
LSB	least significant bit
LUN	logical unit number
Mbaud	megabaud (10^6 transitions per second)
MBps	megabytes per second (10^6 bytes per second)
MHz	megahertz
MSB	most significant bit
ms	millisecond (10^{-6} seconds)
mV	millivolt
N/A	not applicable
NAA	name address authority
NAK	negative acknowledgement
INCITS	International Committee for Information Technology Standards
nF	nanoFarad
ns	nanosecond (10^{-9} seconds)
OOB	out-of-band
PLL	phase lock loop
P-P	peak-to-peak
ppm	parts per million
ps	picosecond (10^{-12} seconds)
RJ	random jitter
Rsvd	reserved
SAM-2	SCSI Architecture Model - 2 standard
SAS	Serial Attached SCSI
SATA	Serial ATA

SCSI	Small Computer System Interface - 3 family of standards
SJ	sinusoidal jitter
SMP	Serial Management Protocol
SPC-3	SCSI Primary Commands - 3 standard
SSC	spread spectrum clocking
SSP	Serial SCSI Protocol
STP	Serial ATA Tunneled Protocol
TJ	total jitter
UI	unit interval
ULP	upper level protocol
V	volt
XOR	exclusive logical OR

3.3 Keywords

3.3.1. **expected:** A keyword used to describe the behavior of the hardware or software in the design models assumed by this standard. Other hardware and software design models may also be implemented.

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

3.3.3. **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.

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

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

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

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

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

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

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

3.3.11. **shall:** A keyword indicating a mandatory requirement. Designers are required to implement all such mandatory requirements to ensure interoperability with other products that conform to this standard.

3.3.12. **should:** A keyword indicating flexibility of choice with a strongly preferred alternative; equivalent to the phrase “it is strongly recommended.”

3.4 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 this clause or in the text where they first appear.

Names of commands, status codes, sense keys, and additional sense codes are in all uppercase (e.g., REQUEST SENSE, BUSY, NO SENSE, INVALID FIELD IN CDB).

Names of fields and state variables are in small uppercase (e.g. ALLOCATION LENGTH). Normal case is used when the contents of a field or state variable are being discussed. Fields or state variables containing only one bit are usually referred to as the NAME bit instead of the NAME field.

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

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

Table 1. ISO and American numbering conventions

ISO	American
0,6	0.6
1 000	1,000
1 323 462,9	1,323,462.9

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

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

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

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

If a conflict arises between text, tables or figures, the order of precedence to resolve the conflicts is text, then tables, and finally figures. Not all tables or figures are fully described in the text. Tables show data formats and values.

Notes do not constitute any requirements for implementers.

3.5 State machine state diagram conventions

Figure 3 shows how state diagrams are formatted.

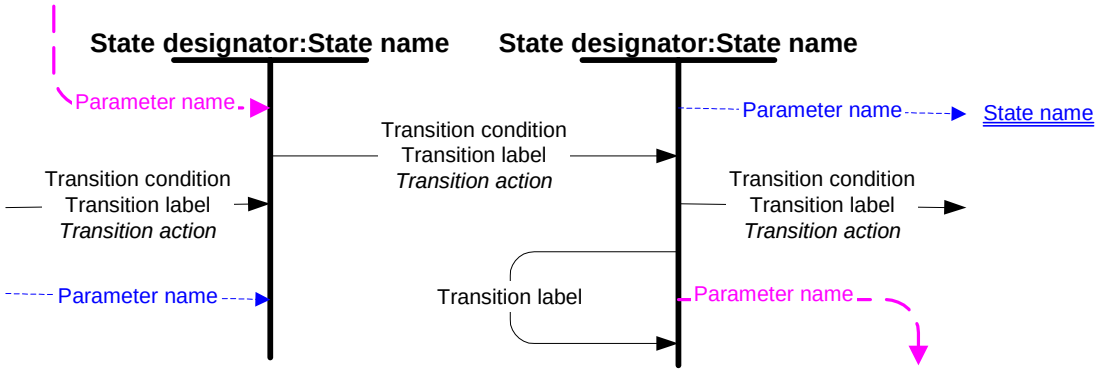

Figure 3. State diagram conventions

Each state is identified by a state designator and a state name. The state designator (e.g., SL1) is unique among all states in all state diagrams 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 appear in other state diagrams. If the same primary function occurs in other states in the same state diagram, they are designated with a unique letter at the end of the state name. Additional actions may be taken while in a state and these actions are described in the state description text.

Each transition is identified by an optional transition label, an optional transition condition, and an optional transition action. The transition label consists of the state designator of the state from which the transition is being made and the state designator of the state to which the transition is being made (e.g., SL1:SL2). In some cases, the transition to enter or exit a state diagram may come from or go to a number of state diagrams, depending on the operation. In this case, the state designator is labeled xx. The transition condition is a brief description of the event or condition that causes the transition to occur. The transition action describes an action that is taken when the transition occurs. The conditions and actions are described fully in the transition description text.

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

In some cases parameters are passed into or out of a state machine from another state machine within the same layer of the protocol. This type of parameter passing is identified by a dashed line originating from or going to the left or right of the figure, labeled with a parameter name and optionally a state machine name or a state name. The description of the use of the parameter and the conditions for passing it are described in the state description text. If the parameter is passed to or from a state machine that is not in the same figure the name of the destination state machine is identified by double-underlining the destination state name.

In some cases parameters are passed into or out of a state machine from another layer of the protocol. This type of parameter passing is identified by a curved dashed line originating from or going to the top or bottom of the figure and a parameter label. The description of the use of the parameter and the conditions for passing it are described in the state description text.

3.6 Notation for procedures and functions

In this standard, the model for functional interfaces between SCSI objects is a remote procedure call. 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.

Input-1, Input-2, …: A comma-separated list of names identifying caller-supplied input data objects.

Output-1, Output-2, …: A comma-separated list of names identifying output data objects to be returned by the procedure.

“[…]”: Brackets enclosing optional or conditional parameters and arguments.

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

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

Where:

Found = Flag

Flag, which, if set, indicates that a matching item was located.

Input Arguments:

Pattern = … /* Definition of Pattern object */

Object containing the search pattern.

Item List = Item<NN> /* Definition of Item List as an array of NN Item objects*/

Contains the items to be searched for a match.

Output Arguments:

Item Found = Item … /* Item located by the search procedure */

This object is only returned if the search succeeds.

4 General

4.1 Standard overview

This standard defines the Serial Attached SCSI (SAS) interconnect and three 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 (which is a mapping of ATA) expanded to support multiple initiators and targets; and
c) Serial Management Protocol (SMP): a management protocol.

Figure 4 shows the layers used by this standard.

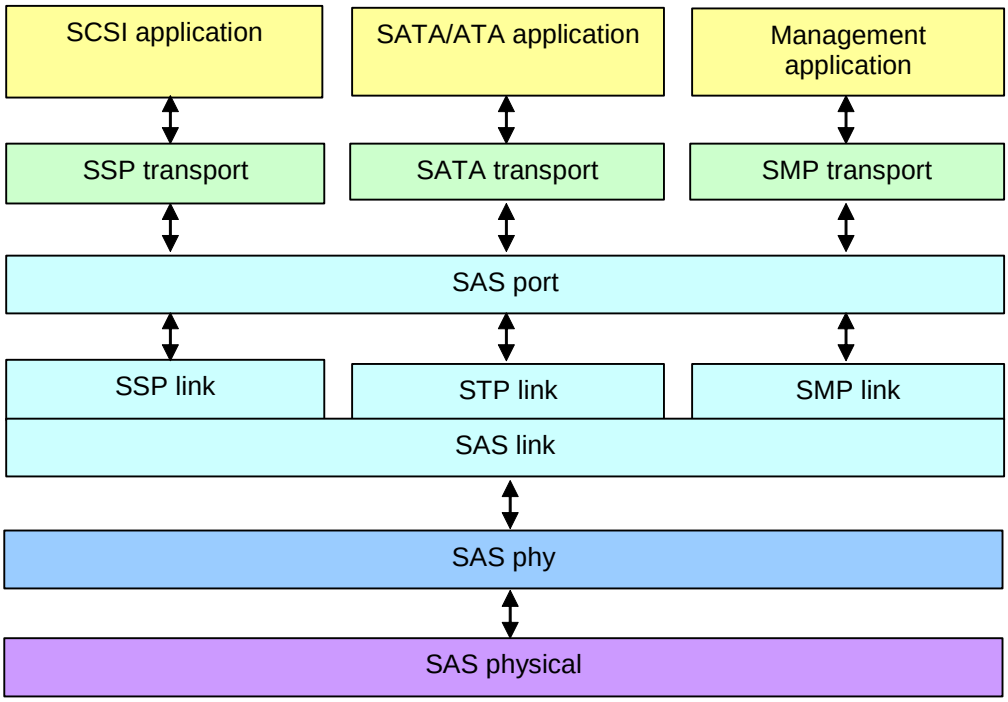

Figure 4. Document organization

The layers are partitioned as follows:

a) Physical: Chapter 5 describes the interconnect layer. This includes connectors, cables, backplanes, and electrical characteristics;
b) Phy: Chapter 6 describes the phy layer. It includes encoding, bit order, OOB signals, reset sequences, dword synchronization, and spin-up.
b) Link: Chapter 7 describes the link layer. It includes primitives, clock skew management, address frames, wide links, logical links, rate matching, power management, test modes, connections, and frame transmission for SSP, STP, and SMP;
c) Port: Chapter 8 describes the port layer. It sits between potentially multiple link layers (for wide links) and multiple transport layers (for multiple protocols);
~~c~~d) Transport: Chapter 9 describes the upper part of the protocol layer. It includes the SSP, STP, and SMP frame definitions; and
~~d~~e) Application: Chapter 10 describes SCSI-specific features (e.g., mode pages ~~and log pages~~) and ATA-specific features.

Chapter 11 describes the mapping of SAS SSP to the SCSI architecture model. The annexes provide additional information on the CRC algorithm, the hash algorithm, the scrambling algorithm, and the SAS logo.

4.2 Architecture

4.2.1 Architecture overview

An ATA domain is an I/O system consisting of a set of ATA devices that communicate with one another by means of a service delivery subsystem. A SCSI domain is an I/O system consisting of a set of SCSI devices that communicate with one another by means of a service delivery subsystem. A SAS domain is the I/O system defined by this standard that may serve as an ATA domain and/or a SCSI domain.

ATA devices are either ATA initiator devices or ATA target devices. SCSI devices are either SCSI initiator devices, SCSI target devices, or SCSI target/initiator devices. A SAS device is an ATA device or SCSI device with ports in a SAS domain.

Table 2 shows the comparable SCSI, ATA, and SAS objects.

Table 2. SCSI, ATA, and SAS comparable objects

SCSI object	ATA object	SAS object
SCSI initiator device	ATA initiator device	SAS initiator device
SCSI initiator port	ATA initiator port	SAS initiator port
SCSI target device	ATA target device	SAS target device
SCSI target port	ATA target port	SAS target port
SCSI domain	ATA domain	SAS domain

The service delivery subsystem in a SAS domain may include expander devices.

Each port contains one or more phys.

4.2.2 Initiator devices

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

Initiator ports may support SSP and/or STP. SATA-only initiator ports are not supported in SAS domains; they are described by the SATA standard.

Figure 5 shows an initiator device.

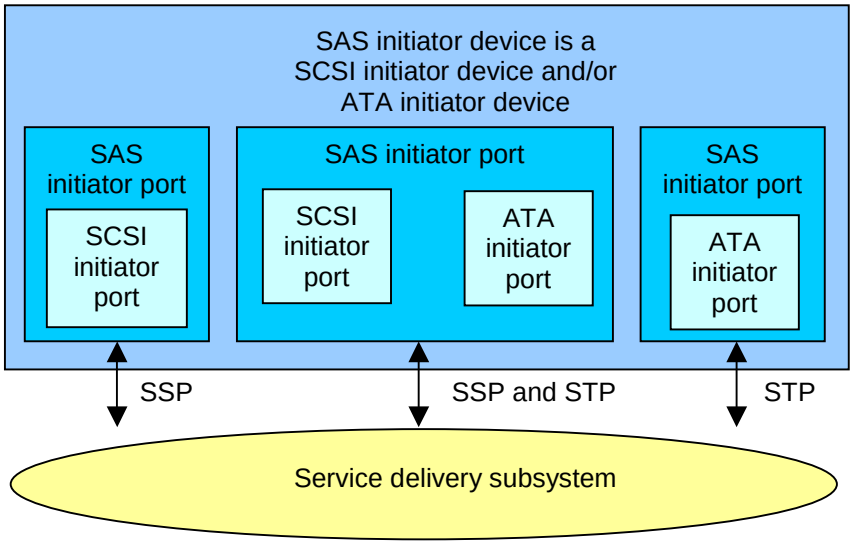

Figure 5. Initiator device

Table 3 shows initiator port protocol support possibilities.

Table 3. Initiator port protocol support

SSP	STP	SATA	Description
no	no	no	Not applicable.
no	no	yes	SATA-only initiator ports are not allowed in SAS domains; they are only supported in SATA domains.
no	yes	no	Initiator port uses STP.
no	yes	yes	Initiator port uses STP in SAS domains. Initiator port uses SATA protocol in SATA domains.
yes	no	no	Initiator port uses SSP.
yes	no	yes	Initiator port uses SSP in SAS domains. Initiator port uses SATA protocol in SATA domains.
yes	yes	no	Initiator port uses either SSP or STP.
yes	yes	yes	Initiator port uses either SSP or STP in SAS domains. Initiator port uses SATA protocol in SATA domains.

4.2.3 Target devices

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

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

Figure 6 shows a target device.

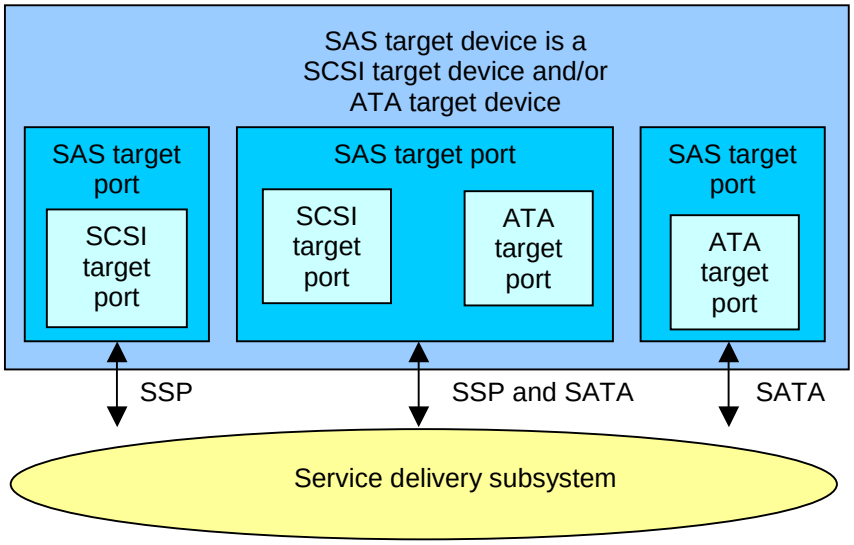

Figure 6. Target device

Table 4 shows target port protocol support possibilities.

Table 4. Target port protocol support

SSP	SATA	Description
no	no	Not applicable.
no	yes	Target port uses SATA. Target port is allowed in a SAS domain if the expander device to which it is connected supports STP.
yes	no	Target port uses SSP.
yes	yes	Target port uses SSP in SAS domains. Target port uses SATA in SATA domains.

4.2.4 Target/initiator devices

Target/initiator devices are devices which support both initiator device and target device roles. Their ports are called target/initiator ports, which support both initiator port and target port roles.

4.2.5 Expander devices

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

Expander ports may support being attached to SSP and/or STP initiator ports, SSP and/or SATA target ports, and to other expander ports.

Figure 7 shows an expander device.

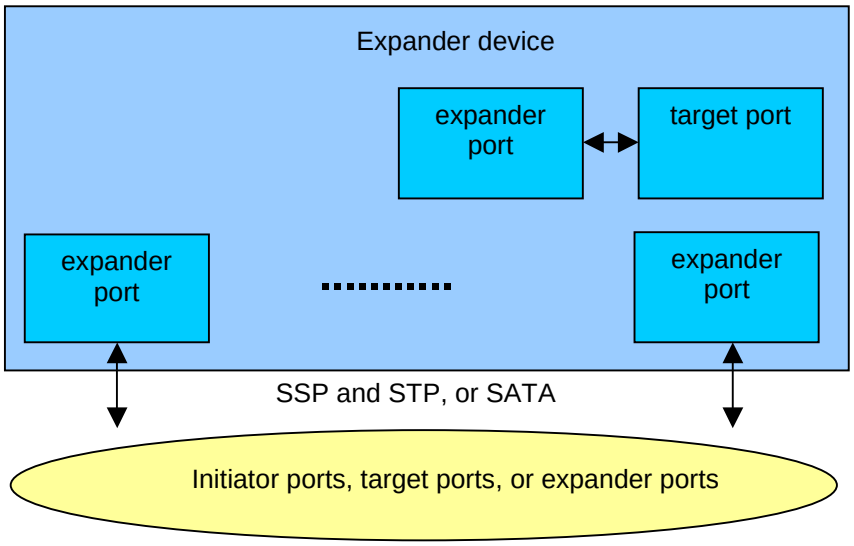

Figure 7. Expander device

There are two classes of expander devices:

a) fanout expander devices: expanders that contain complex routing capability; and

b) edge expander devices: expanders that only contain simple routing capability.

This standard does not define the protocol used to build an expander device (edge or fanout) with multiple chips. Expander devices composed of multiple chips may require some chips implement routing table communication between the chips similar to that managed by the fanout expander device for the edge expander devices.

4.2.6 Physical links and phys

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

A phy is a transceiver in a device that electrically interfaces to a physical link.

Figure 8 shows two phys attached with a physical link.

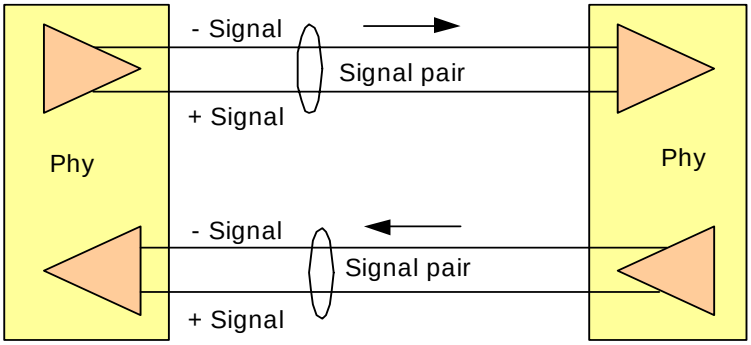

Figure 8. Phy

A device may contain one or more phys. A port may contain one or more phys. Ports in a device may be dynamically allocated phys based on physical link initialization.

4.2.7 Wide physical links and relationship to ports

When a device has one or more of its phys attached to another device, a port is created. If there are n phys attached to the same other device, an n-wide port is created and the set of physical links is called an n-wide physical link.

Figure 9 shows examples of wide physical links.

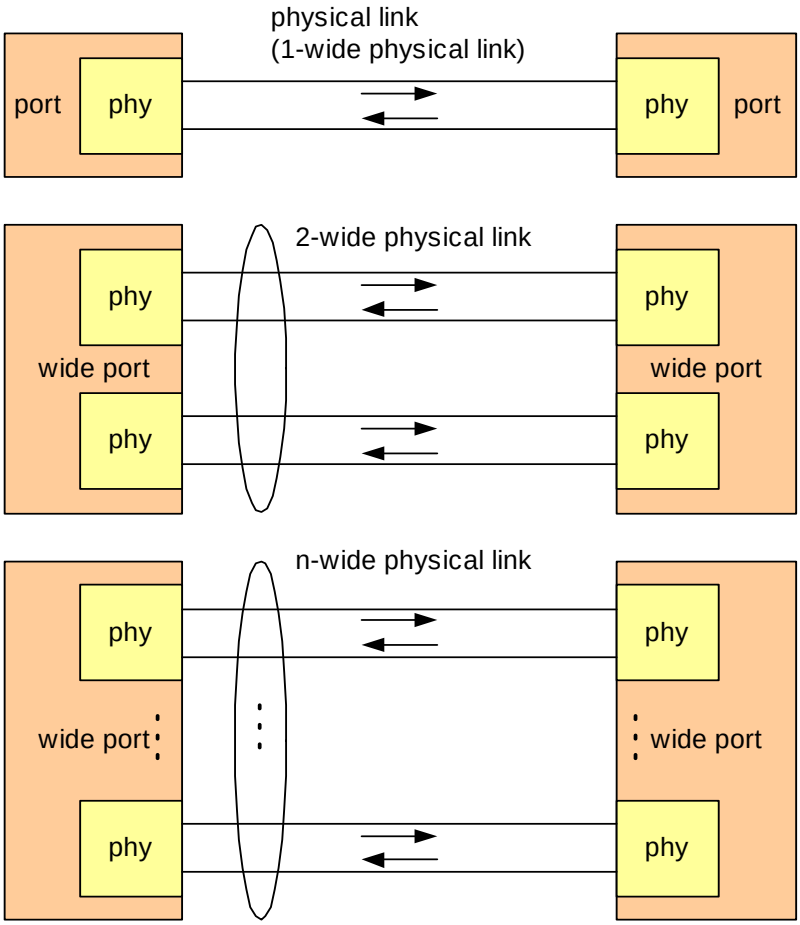

Figure 9. Wide physical links

4.2.8 Domains and connections

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

This standard defines SAS domains supporting the following connections:

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

SATA defines the SATA domain which supports a SATA initiator port to SATA target port connection.

Figure 10 shows the possible domains and connections.

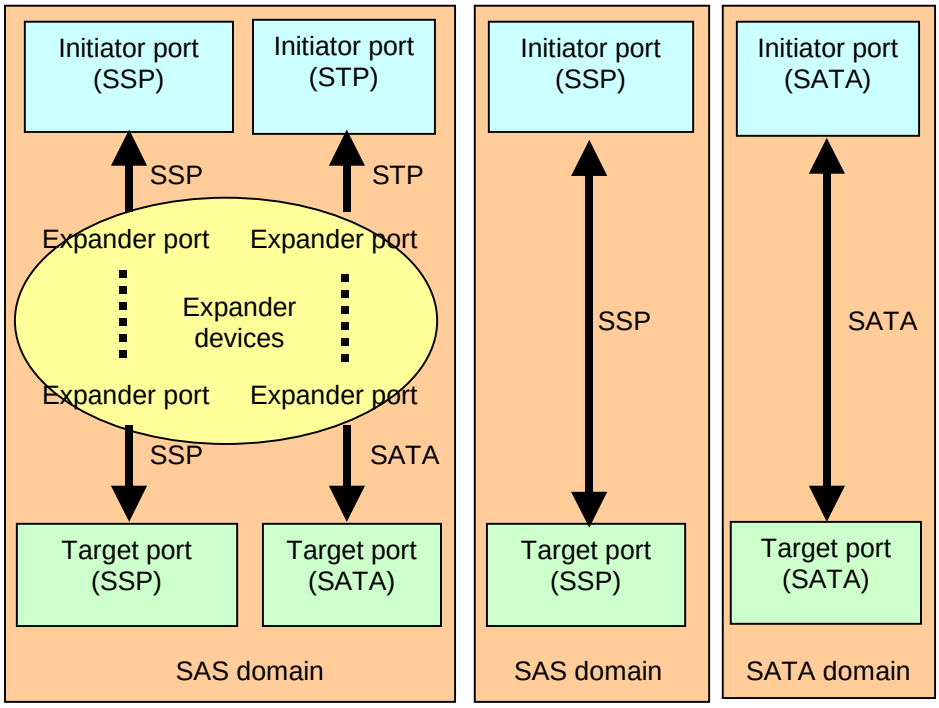

Figure 10. Domains and connections

The expander port attached to a SATA target port translates STP to SATA; other expander ports pass through STP traffic. All expander ports pass through SSP traffic. Expander ports never translate SSP to SATA.

The SAS initiator ports of an initiator device shall be in separate SAS domains. Figure 11 shows an initiator device whose initiator ports are in separate SAS domains.

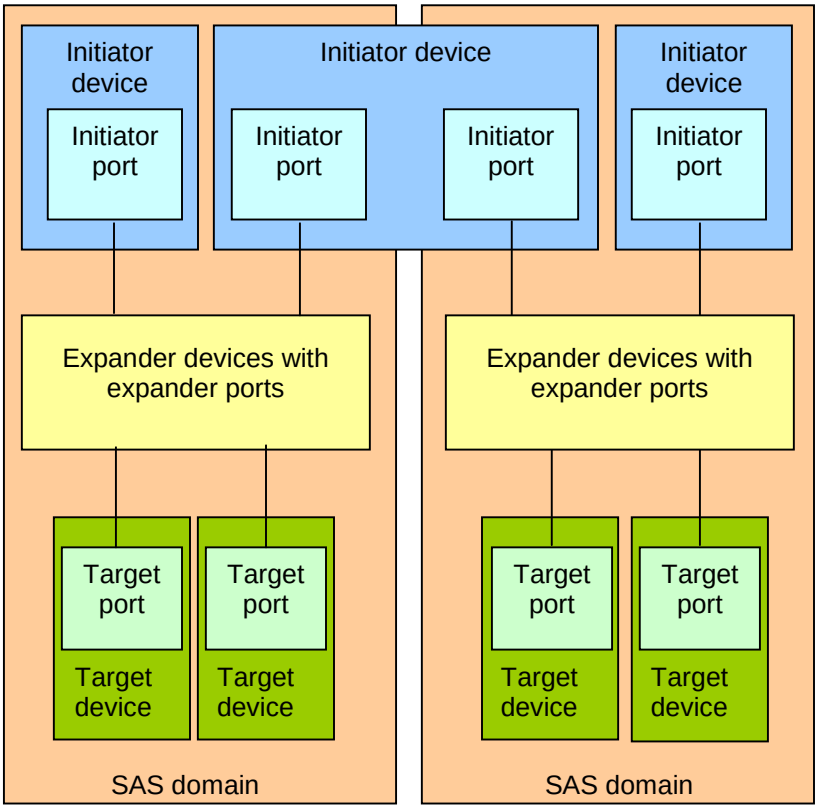

Figure 11. Initiator device spanning two SAS domains

The SAS target ports of a target device shall be in separate SAS domains. If initiator ports meet only at a target device, then multiple SAS domains are present. Figure 12 shows an example of two SAS domains that meet at a target device.

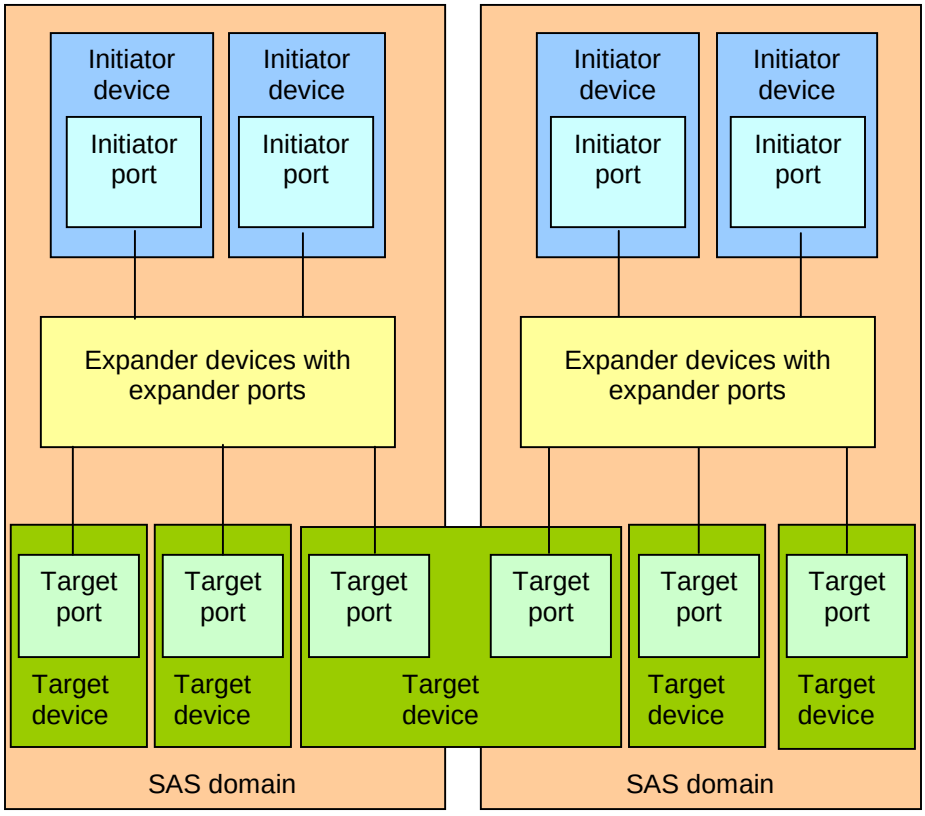

Figure 12. Target device spanning two SAS domains

4.2.9 Expander topologies

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

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

Each edge expander device may be attached to no more than one fanout expander device. Each edge expander device may be attached to up to 64 initiator ports or target ports.

An edge expander device may be attached to another edge expander device only if there are no other expanders in the SAS domain.

Three levels of expander devices are allowed between initiator ports and target ports, as shown in Figure 13.

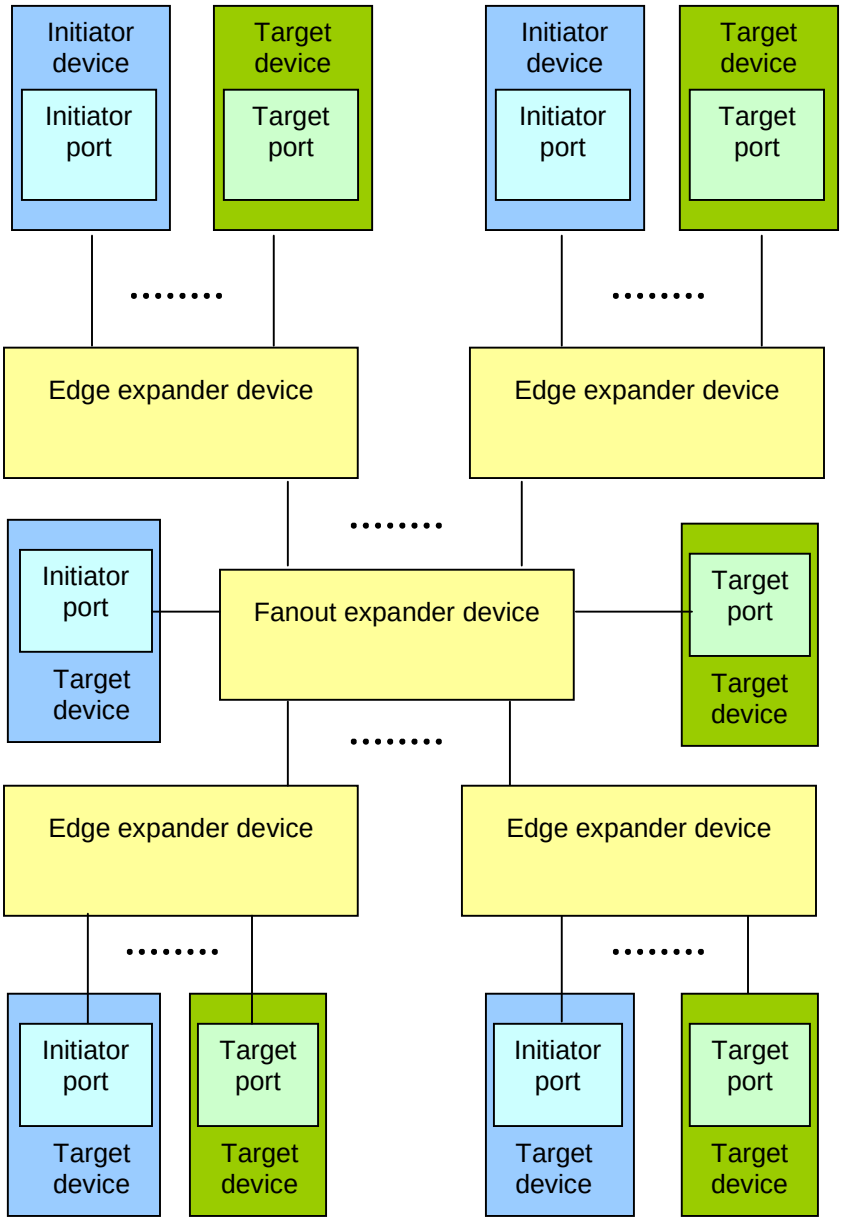

Figure 13. Expander topologies

4.2.10 Connections and pathways

One connection may be active on a physical link at a time. If the connection is an SSP or SMP connection and there are no dwords to transmit associated with that connection, idle dwords are transmitted. If the connection is an STP connection and there are no dwords to transmit associated with that connection, SATA_SYNCs, SATA_CONTs, or scrambled random data are transmitted. Any physical link that is not carrying a connection carries idle dwords.

A pathway is the physical route of a connection. In the case where an initiator port is directly attached to a target port the pathway and the physical link are identical. In the case where there are expanders between an initiator port and a target port, the pathway consists of all the links required to route dwords between the initiator port and the target port.

Figure 14 shows examples of pathways.

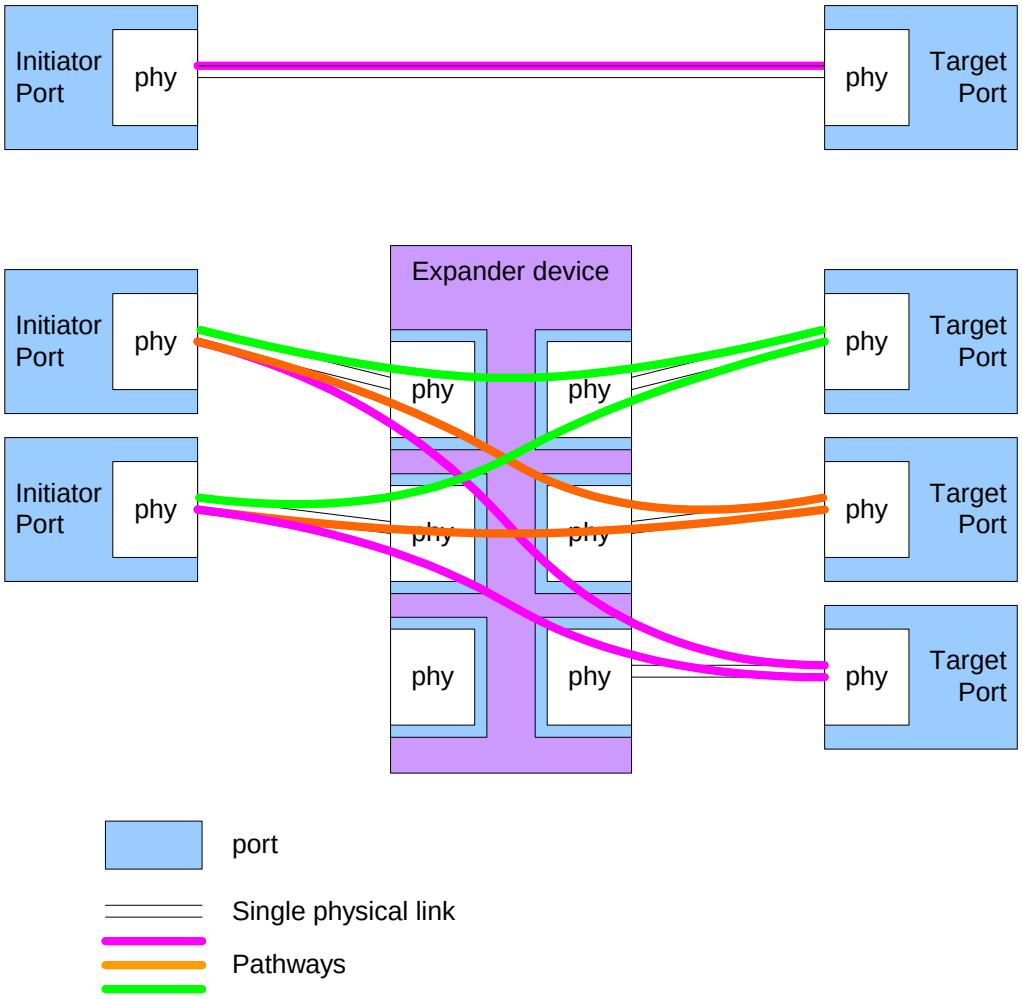

Figure 14. Pathways

A partial pathway is the set of physical links participating in a connection request which has not reached a SAS endpoint (e.g., the connection request (OPEN address frame) has been sent by the connection originator and confirmed received by at least one expander via the AIP primitive).

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

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

4.3 Names and identifiers

4.3.1 Names and identifiers overview

Device names are worldwide unique names for devices within a protocol. Port names are worldwide unique names for ports within a protocol. Port identifiers are the values by which ports are identified within a domain, and are used as addresses. Phy identifiers are unique within a device.

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

Table 5. Names and identifiers

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

4.3.2 Device names

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

Table 6 defines the device name format. Device names shall be compatible with the NAA (Name Address Authority) IEEE Registered format identification descriptor defined in SPC-3.

Table 6. Device name format

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

The NAA field contains 5h.

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

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

The device name shall be worldwide unique.

4.3.3 Hashed device name

SSP frames include a hashed version of the device name for extra 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 7 lists the parameters for the code.

Table 7. Hashed device name code parameters

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

The generator polynomial for this code is:

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

After multiplication of the factors, the generator polynomial is:

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

4.3.4 Port names

Port names, used by SCSI features such as persistent reservations, are not defined in SAS, because the device name used by ports in different SAS domains may be the same. Also, there is no login process in SSP to exchange port names.

[Editor's note: If desired, the port name could be considered the device name plus the attached expander device name. This set of names is worldwide unique within the SAS protocol (wider than just within the domain, which is all that is required of port identifiers). Or, we could require endnodes treat ports with the same name as the same port in all domains, potentially allowing traffic to span the domains.]

4.3.5 Port identifiers

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

4.3.6 Phy identifier

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

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

4.4 SCSI and ATA architectural notes

4.4.1 STP differences from SATA

Some of the differences of STP compared with SATA are:

a) STP adds addressing multiple SATA targets;
b) STP allows multiple initiators to share access to a SATA target;
c) interface power management is not supported;
d) far-end analog loopback testing is not supported;
e) target ports are not sent to sleep during near-end analog loopback testing; and
f) use of the CONT primitive is mandatory.

4.4.2 SCSI architecture notes

Some of the SCSI architecture limitations of SSP are:

a) a limited number of ports are supported (64 ports per expander);
b) third-party reservations with RESERVE (10) are not supported. No initiator identifier format is provided for the RESERVE (10) command;
c) command reference numbers are not supported;
d) port names are not supported. No login/logout IUs are defined to communicate them;
e) the TARGET RESET task management function is not supported;
f) the WAKEUP task management function is not supported; and
g) untagged commands are not supported.

However, SAS does support these SCSI features:

a) variable-length CDBs;
b) bidirectional commands;
c) 8 byte LUNs;
d) native asynchronous event notification; and
e) autosense.

4.5 Transmit data path

Figure 15 shows the transmit data path inside an initiator port or target port, indicating where each state machine fits.

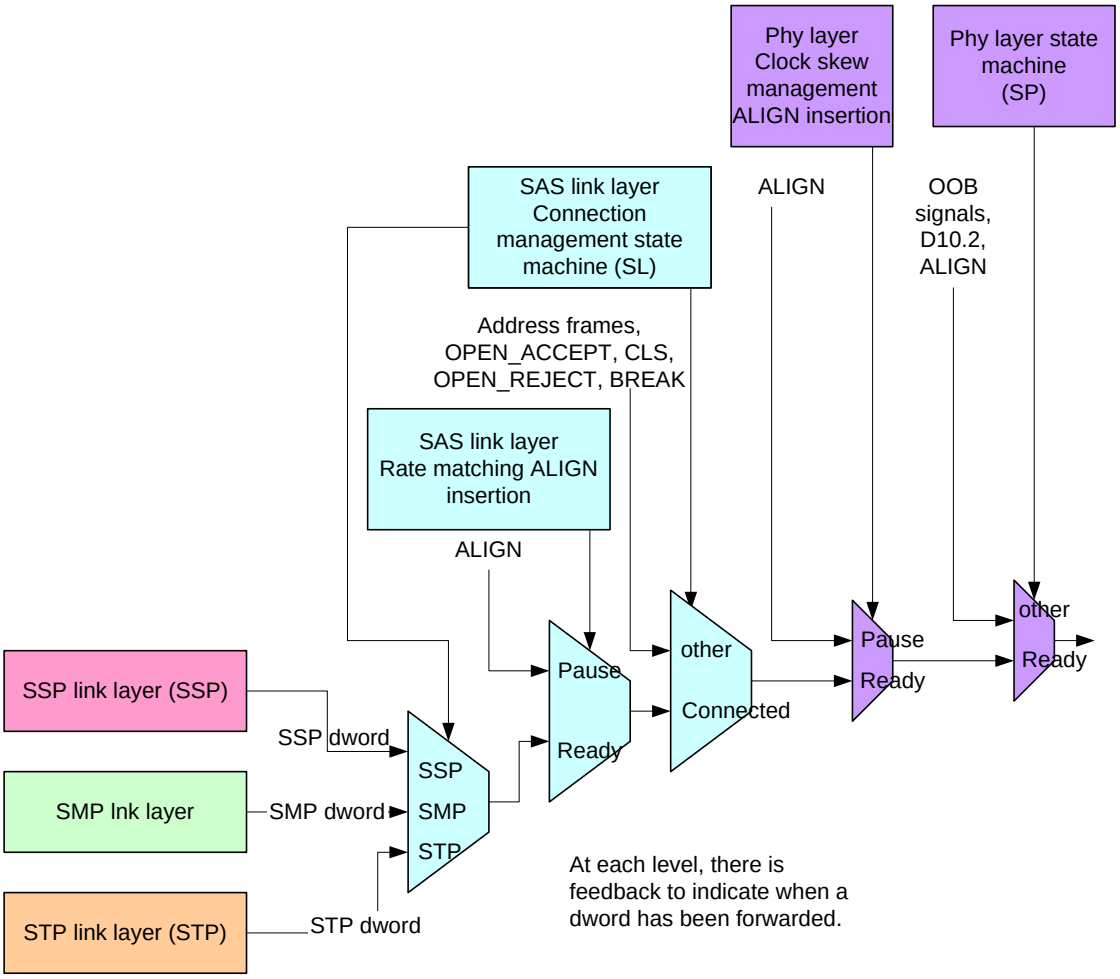

Figure 15. Transmit data path and state machines

Figure 16 shows the transmit data path for the SSP link, transport, and application layers.

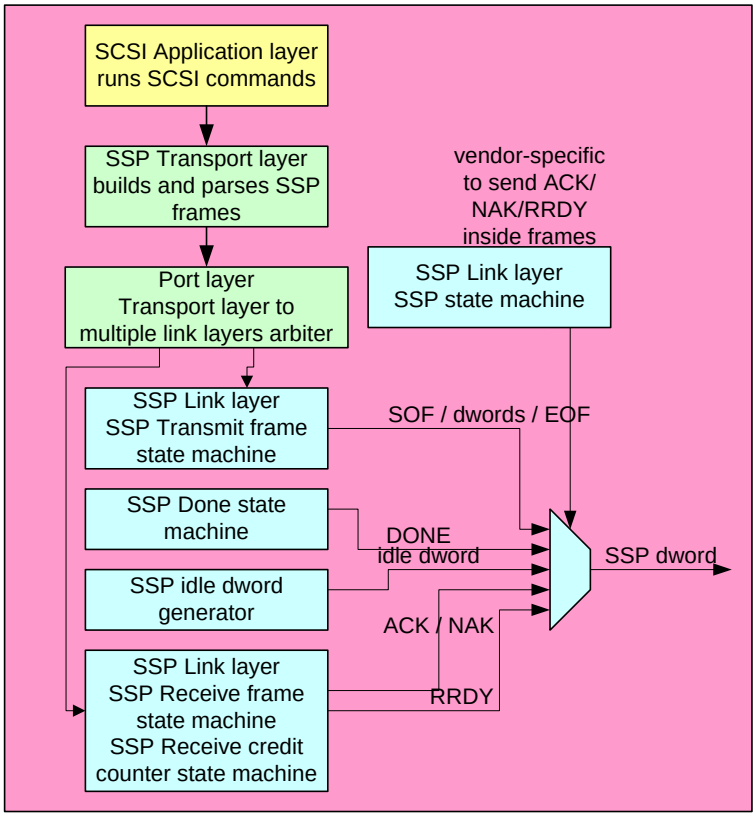

Figure 16. SSP link, transport, and application layer state machines

Figure 17 shows the transmit data path for the STP link, transport, and application layers.

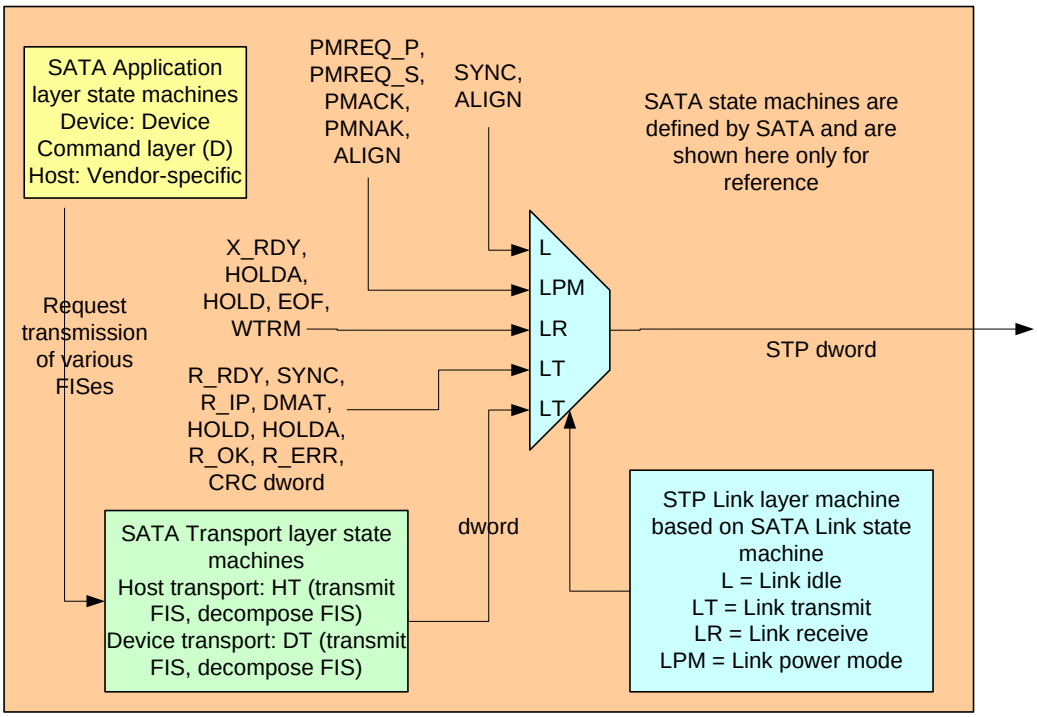

Figure 17. STP link, transport, and application layer state machines

The transport layers communicate~~s~~ with more than one link layer when wide links are present. The port layer controls this.

4.6 Resets

4.6.1 Reset overview

Figure 18 describes the reset terminology.

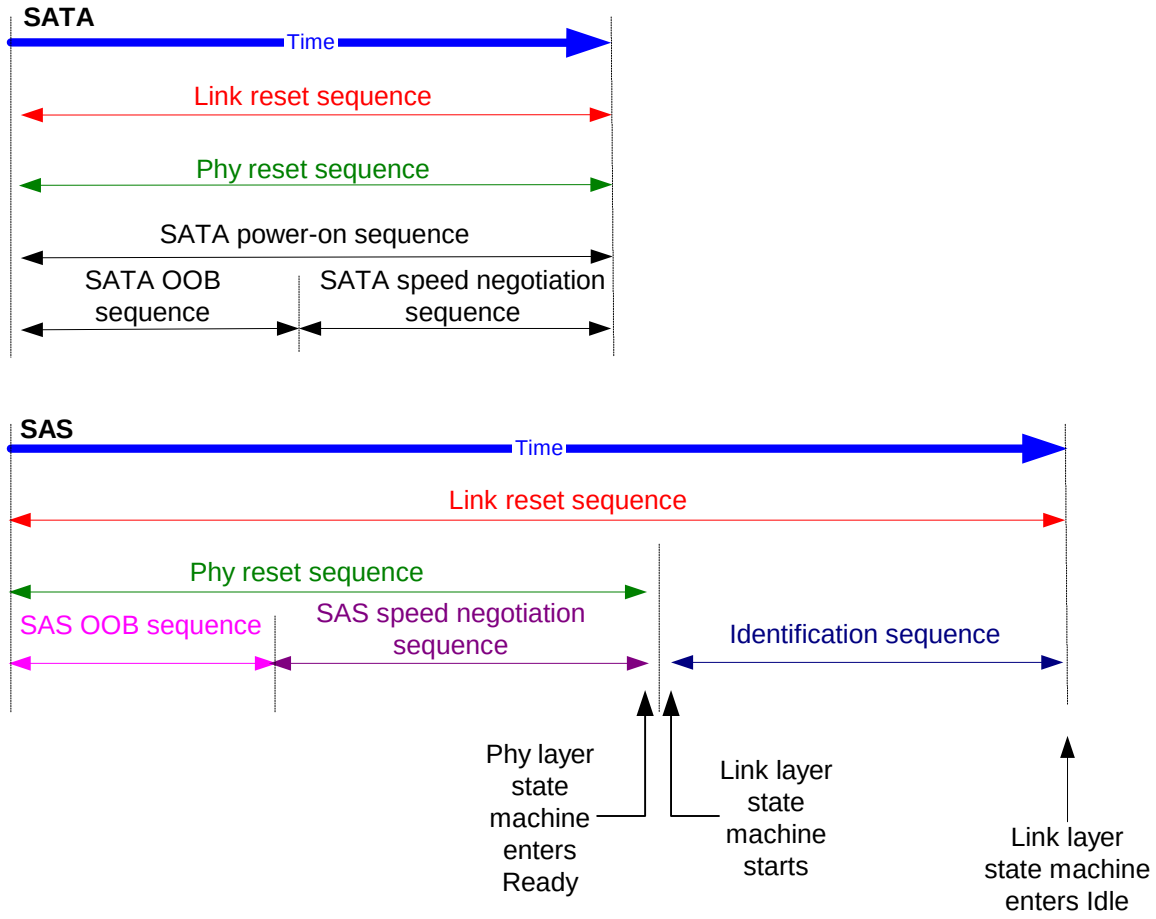

Figure 18. Reset terminology

The phy reset sequences, including the OOB sequence and speed negotiation sequences, are described in 6.4. The identification sequence is described in 7.5.

SAS devices treat the link reset sequence as having little impact. The HARD RESET primitive may be used during the identification sequence to trigger a hard reset.

SATA devices treat the link reset sequence as a hard reset.

4.6.2 Hard reset

Between the link reset sequence and the IDENTIFY address frame, if a SAS port receives a HARD RESET primitive sequence, it shall be considered a reset event and cause a hard reset of the port.

When a port detects a hard reset, it shall stop transmitting on all its phys for 1 ms. Care should be taken when sending HARD RESET to an expander device.

If the phy is part of a SCSI device, the SCSI device shall perform the actions defined for hard reset in SAM-2.

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

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

4.6.3 Loss of signal

When a phy detects loss of incoming signal, it shall stop transmitting for 1 ms. It shall not cause a reset event of the port.

4.7 I_T nexus loss

When a port receives OPEN_REJECT(NO DESTINATION) in response to a connection request, it shall retry the connection request repeatedly for the time indicated by the I_T NEXUS LOSS field in the Protocol-Specific Port Control mode page (see 10.1.1.2). If the port continues to receive rejections until that time expires, it shall consider the rejections as an I_T nexus loss (see SAM-2).

4.8 Expander device model

4.8.1 Expander device model overview

Figure 19 shows the model of an expander device.

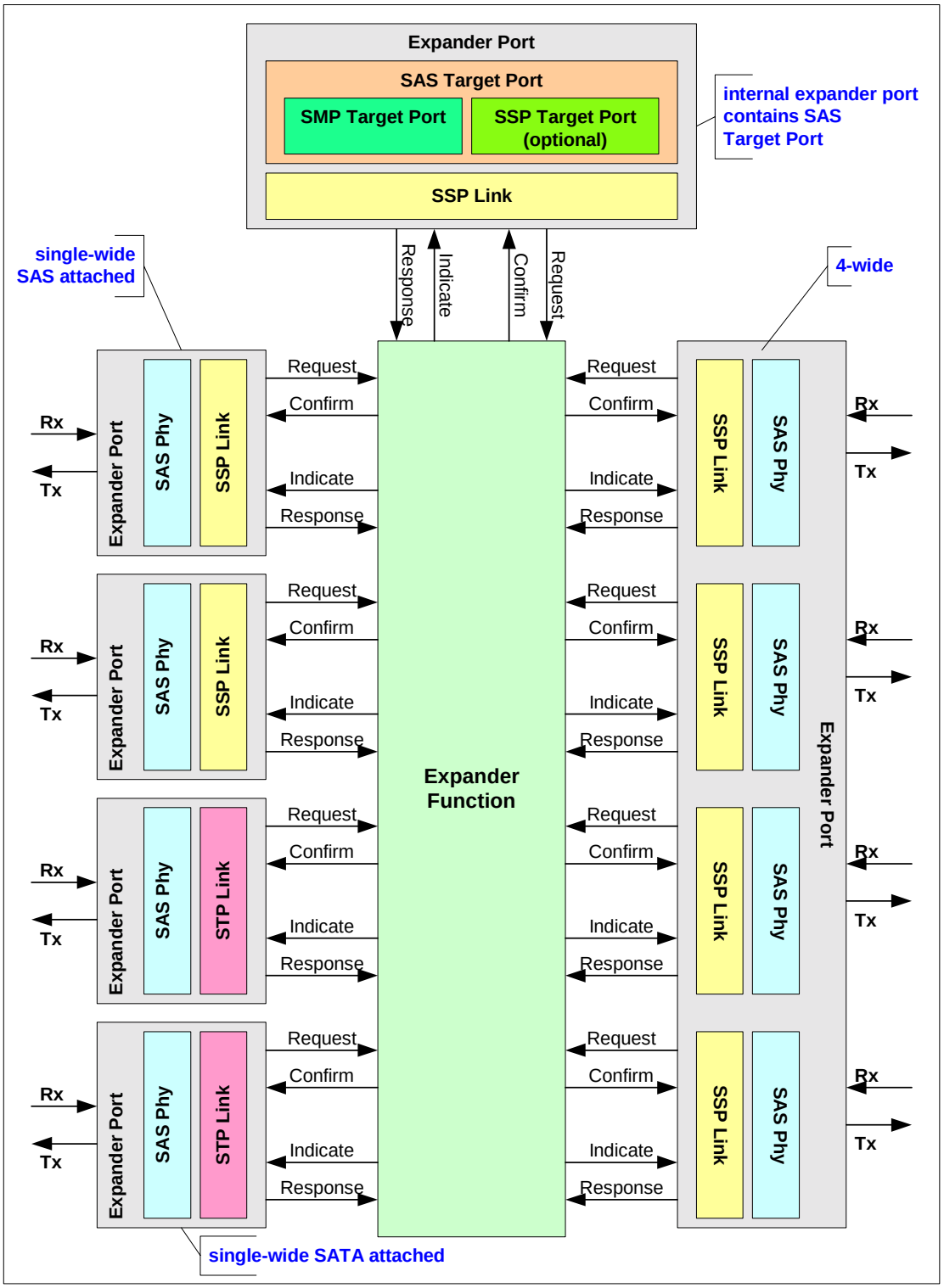

Figure 19. Expander device model

An expander device may present a virtual target port to an initiator port that acts like an SSP target port but represents an embedded target device, like a SCSI enclosure services device.

4.8.2 Expander service interface

The expander device provides arbitration and routing service between links present within expander ports. All routing occurs between links (which abstract the width of the physical interconnect). To assist in the specification of the link state transition diagrams, the interaction between Links are defined in terms of a service interface to the Expander Function.

Figure 20 describes the expander service interface.

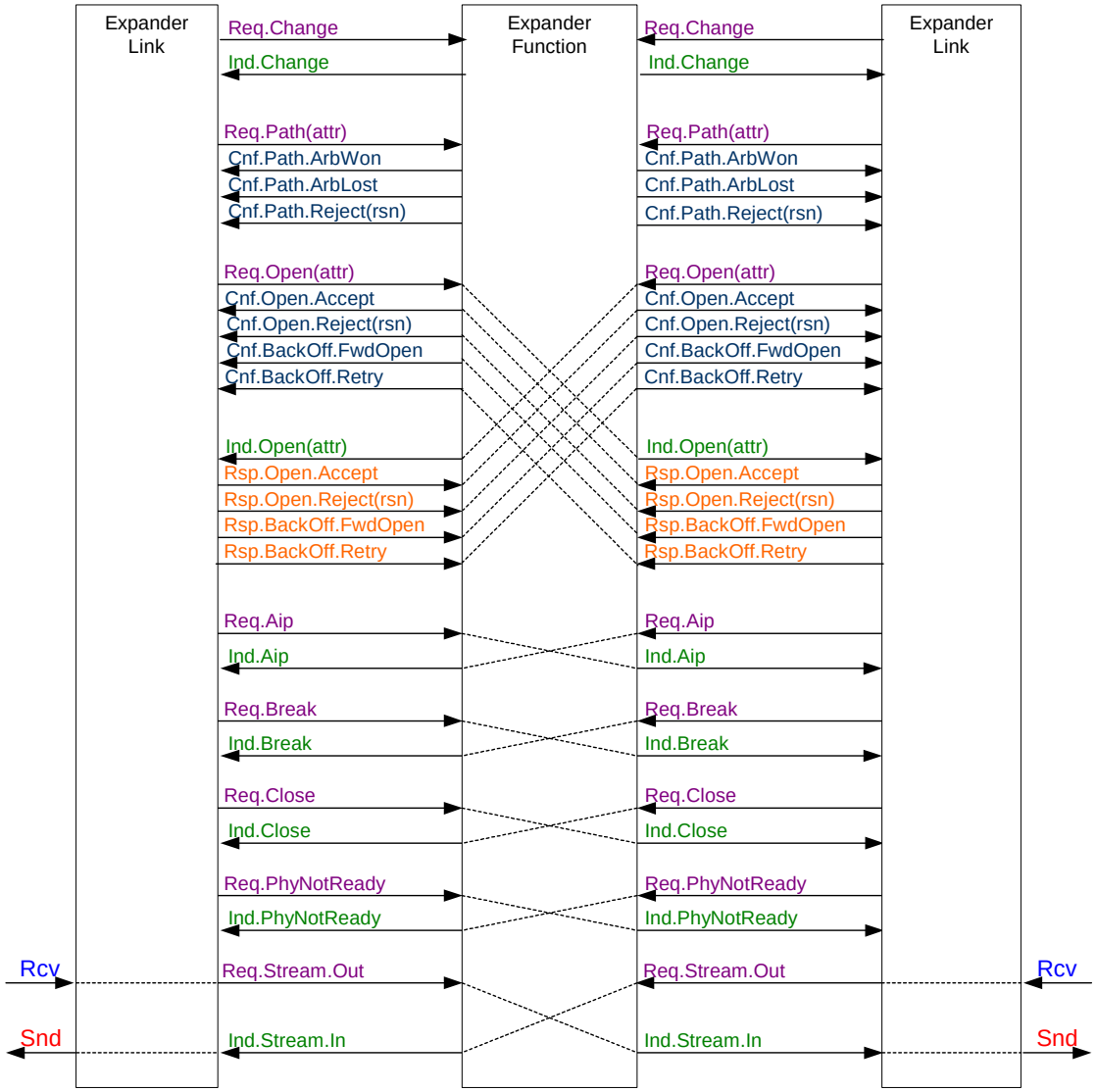

Figure 20. Expander service interface

4.8.3 Expander service interface primitives

4.8.3.1 Request primitive

The Request primitive is passed from a Link to the Expander Function to request that a service be initiated, typically based on reception of a primitive or frame by the Link. The arguments to Request are:

a) Request.Path(attributes);
b) Request.Open(attributes);
c) Request.Aip;
d) Request.Break;

e) Request.Change;
f) Request.Close;
g) Request.PhyNotReady; and
h) Request.StreamOut

4.8.3.2 Confirm primitive

The Confirm primitive is passed from the Expander Function to a Link to convey the results of one or more associated previous service requests. The arguments to Confirm are:

a) Confirm.Path.ArbWon;
b) Confirm.Path.ArbLost;
c) Confirm.Path.ArbReject;
d) Confirm.Open.Accept;
e) Confirm.Open.Reject(reason);
f) Confirm.Backoff.FwdOpen; and
g) Confirm.Backoff.Retry.

4.8.3.3 Indicate primitive

The Indicate primitive is passed from the Expander Function to a Link to indicate an internal event. This indication may be related to a remote service request (a result of a service request from a different Link) or may be caused by an event internal to the Expander Function. The arguments to Indicate are:

a) Indicate.Open(attributes);
b) Indicate.Aip;
c) Indicate.Break;
d) Indicate.Change;
e) Indicate.Close;
f) Indicate.PhyNotReady; and
g) Indicate.StreamIn

4.8.3.4 Response primitive

The Response primitive is passed from the Link to the Expander Function to complete a previously invoked indicate primitive. The arguments to Response are:

a) Response.Open.Accept;
b) Response.Open.Reject(reason);
c) Response.Backoff.FwdOpen; and
d) Response.Backoff.Retry.

4.8.4 Expander function requirements

4.8.4.1 Device name mapping

The Expander Function shall provide device name to phy number mapping using the device names and type which are exchanged during phy initialization.

Fanout expander devices must perform additional discovery for each attached edge expander device in order to establish a routing table.

4.8.4.2 Routing

The Expander Function shall provide routing resources between pairs of links during connections. The Expander Function shall provide the routing resources to support at least one active connection.

4.8.4.3 Broadcast/CHANGE primitive processing

The Expander Function shall process broadcast primitive requests specified by each link via the Request.Change expander primitive. The Expander Function is responsible to schedule the transmission of a CHANGE primitive sequence on each port (using the Indicate.Change expander primitive) other than the one on which it received the CHANGE primitive sequence. The Expander Function shall ensure that the CHANGE primitive sequence is only scheduled for transmission on a single link of a wide port.

4.8.4.4 Arbitration and resource management

4.8.4.4.1 Arbitration overview

The Expander Function shall arbitrate and assign or deny path resources for connection attempts requested by each link in response to receiving valid OPEN address frames.

Arbitration includes adherence to the SAS fairness algorithm and path recovery. Path recovery is used to avoid potential deadlock scenarios within the SAS topology by deterministically choosing which partial pathway(s) to tear down to allow at least one connection to complete.

An expander link requests path resources using the Request.Path(attributes) expander primitive.

The Expander Function uses the Confirm.Path.ArbWon, Confirm.Path.ArbLost, and Confirm.Path.ArbReject expander primitives to provide arbitration confirmation back to the requesting link.

Each path request contains the Arbitration Wait Timer value and the source device name of the received OPEN address frame. These two values are used according to SAS fairness rules to determine connection request priority.

The Expander Function shall generate the Confirm.Path.ArbReject expander primitive when any of the following conditions are met:

a) the Request.Path(attributes) does not map to a valid phy;
b) the Request.Path(attributes) specifies an unsupported link rate; or
c) the Request.Path(attributes) specifies a destination port which contains at least one partial pathway -AND- pathway recovery rules require this connection request to release path resources.

The Expander Function shall generate the Confirm.Path.ArbLost expander primitive when all of the following conditions are met:

a) the Request.Path(attributes) maps to an available phy at a supported link rate; and
b) the destination phy of this connection request has received a higher priority OPEN address frame with this phy as its destination (this case occurs when two phys both receive an OPEN address frame destined for each other, the Expander Function shall provide arbitration lost confirmation to the phy that received the lowest priority OPEN address frame).

The Expander Function shall generate the Confirm.Path.ArbWon expander primitive when all of the following conditions are met:

a) the Request.Path(attributes) maps to an available phy at a supported link rate; and
b) no higher priority connection requests are present with this phy as the destination.

4.8.4.4.1.1 Arbitration status

Arbitration status shall be conveyed between expander devices and by expander devices to SAS endpoints using four types of AIP primitives. This status is used to monitor the progress of connection attempts and to facilitate pathway recovery as part of deadlock avoidance.

AIP(NORMAL) is sent by an expander link:

a) after receipt of a valid OPEN address frame and prior to arbitrating for path resources (i.e. before it is aware of the status of the destination port); and
b) during path arbitration unless the Expander Function indicates WAITING_ON_PARTIAL, WAITING_ON_CONNECTION, or WAITING_ON_DEVICE via the Indicate.Aip expander primitive.

AIP(WAITING_ON_CONNECTION) is sent by an expander link when:

a) The Expander Function provides indication via the Indicate.Aip(WAITING_ON_CONNECTION) expander primitive that the connection request is not able to be completed because all links within the destination port are busy with pending or active connections.

AIP(WAITING_ON_PARTIAL) is sent by an expander link when the Expander Function provides indication via the Indicate.Aip(WAITING_ON_PARTIAL) expander primitive that the connection request is not able to be completed because:

a) there are no unallocated links available within the destination port to complete the connection request; and
b) at least one link within the destination port contains a blocked partial pathway.

AIP(WAITING_ON_DEVICE) is sent by an expander device when:

a) The Expander Function provides indication via the Indicate.Aip(WAITING_ON_DEVICE) expander primitive that the OPEN address frame has been forwarded to a SAS endpoint.

The arbitration status of an expander link is set to the last value of AIP received.

4.8.4.4.2 Partial Pathway Timer

The Expander Function maintains a Partial Pathway Timeout timer (PPT timer) for each link within the Expander. This timer is used for pathway recovery when potential deadlock conditions exist. The default value for the Partial Pathway Timeout value (PPTOV) shall be 7 microseconds with a minimum configurable value of 1 microsecond and a maximum configurable value of 15 microseconds.

The Expander Function shall decrement the PPT timer once per microsecond, until reaching zero, when the following conditions are met:

b) there are no unallocated links within a requested destination port available to complete the connection; and
c) at least one link within the requested destination port contains a blocked partial pathway.

When either of the conditions above are not met, the Expander Function shall hold the PPT timer at an initial value of PPTOV.

4.8.4.4.3 Pathway Recovery

Pathway recovery provides a means to abort connection requests in order to prevent deadlock.

When the PPT timer for an arbitrating link expires, i.e. reaches a value of zero, the Expander Function shall determine whether to continue the connection request or to abort the connection request as follows:

The Expander Function shall instruct the arbitrating link to reject the connection request by transmitting OPEN_REJECT(PATHWAY_BLOCKED) when the PPT timer expires and one of the following conditions are met:

a) the AWT value of the arbitrating link (link requesting the connection) is less than the AWT value of all links within the destination port with an arbitration status of WAITING_ON_PARTIAL; and
b) the AWT value of the arbitrating link is equal to the AWT value of all links within the destination port with an arbitration status of WAITING_ON_PARTIAL and the source device name of the arbitrating link is less than the source device name of all links within the destination port with an arbitration status of WAITING_ON_PARTIAL.

If none of the conditions above are met the Expander Function shall reinitialize the PPT timer with the PPTOV value. If the PPT timer expires a second time, the Expander Function shall instruct the arbitrating link to reject the connection request by transmitting OPEN_REJECT(PATHWAY_BLOCKED) when the following conditions are met:

a) the source device name of the arbitrating link is less than the source device name of any links within the destination port with an arbitration status of WAITING_ON_PARTIAL.

5 Physical layer

5.1 SATA cables and connectors

Figure 21 shows the cables and connectors defined by SATA (for reference). A “SATA host” is equivalent to a SAS initiator device; a “SATA device” is equivalent to a SAS target device.

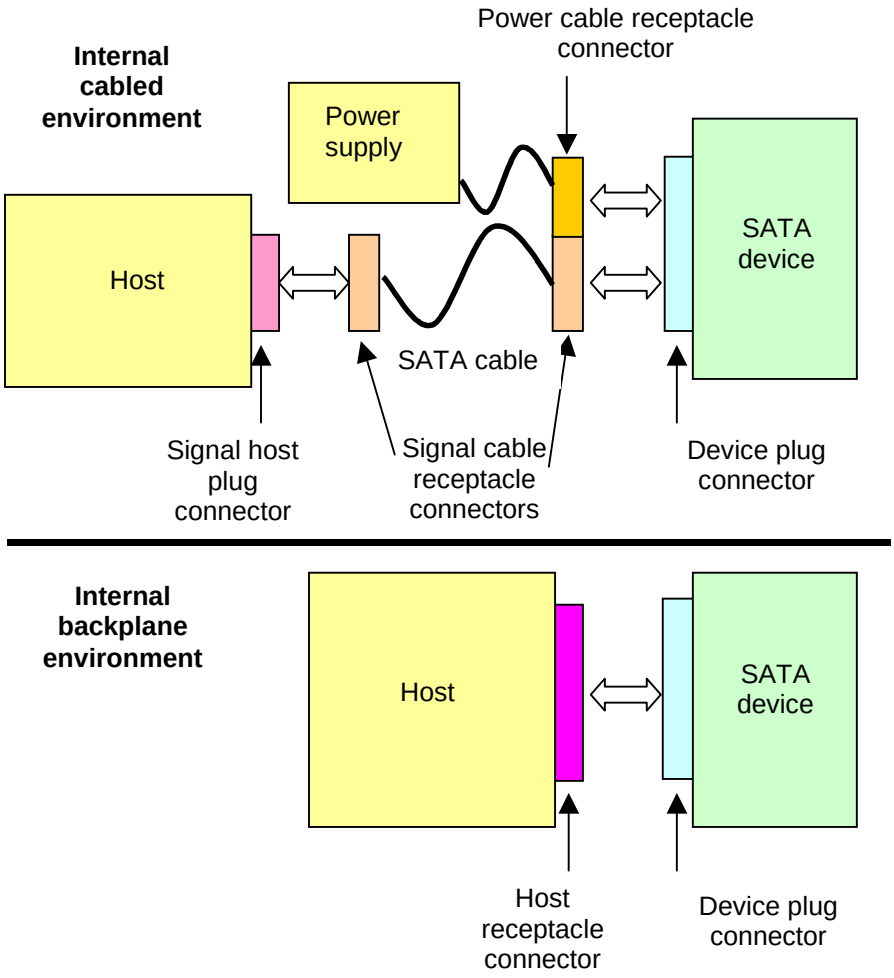

Figure 21. SATA cables and connectors (reference)

5.2 SAS cables and connectors

This standard supports internal cable and internal backplane environments.

Figure 22 shows the cables and connectors defined in this standard to support an external environment.

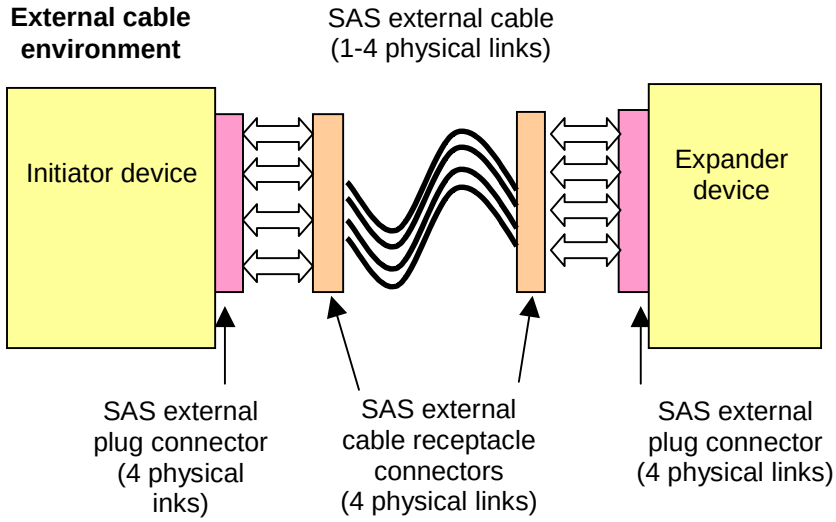

Figure 22. SAS cables and connectors - external environment

Figure 23 shows the connectors defined by this standard for internal environments.

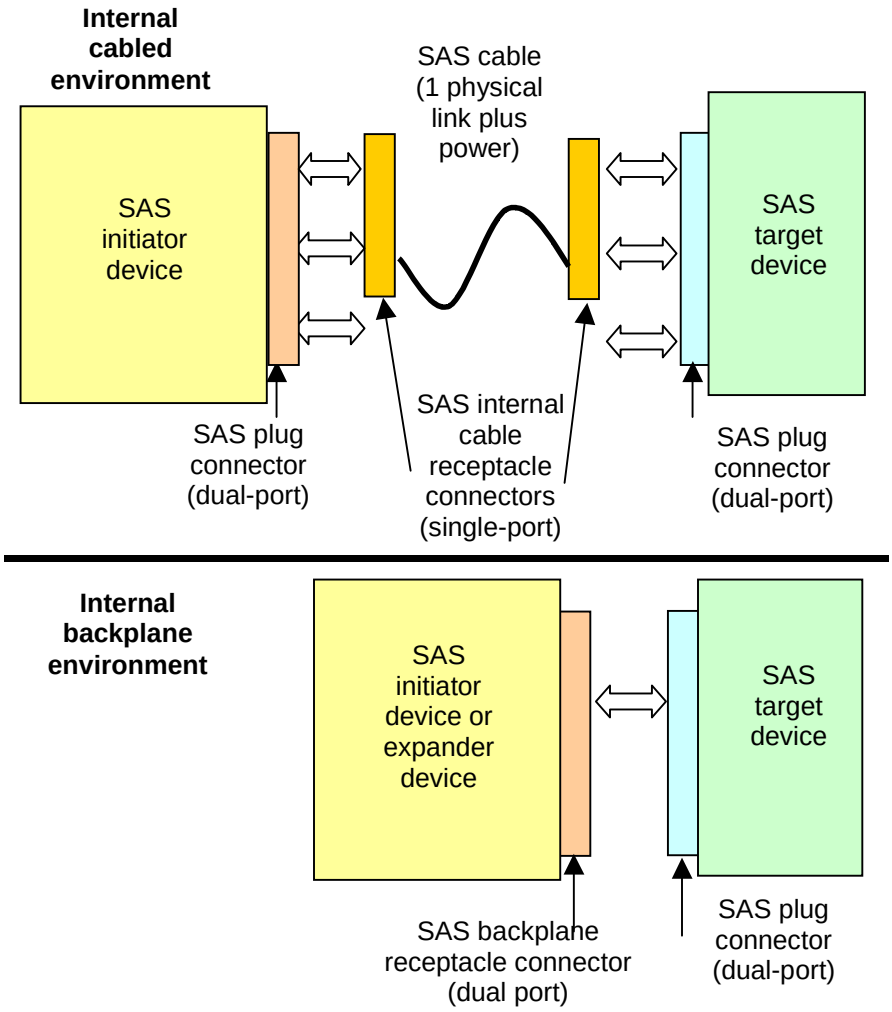

Figure 23. SAS connectors - internal environment

Table 8 summarizes the connectors defined by this standard.

Table 8. Connectors

Type of connector	Reference	Attaches to	Reference
SAS plug	5.3.2	SAS internal cable receptacle	5.3.3
		SAS backplane receptacle	5.3.4
SAS internal cable receptacle	5.3.3	SAS plug	5.3.2
		SATA device plug (single-port)	SATA
SAS backplane receptacle	5.3.4	SAS plug	5.3.2
		SATA device plug (single-port)	SATA
SAS external cable receptacle	5.3.5	SAS external plug	5.3.6
SAS external plug	5.3.6	SAS external cable receptacle	5.3.5

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

5.3 Connectors

5.3.1 Connectors overview

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

5.3.2 SAS plug connector

5.3.2.1 SAS plug connector overview

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

Table 9 shows the pins in the SAS plug connector.

Table 9. SAS plug connector pins

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

Notes:
Precharge and voltage pins shall be connected together.
S8 through S14 are no-connects on single port implementations.

Unlike SATA, the initiator device and target device share the same connector pin assignments. Therefore, the cable or backplane must swap the transmit and receive signal pairs.

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

SAS plug connectors shall support at least 500 insertions and 500 removals.

5.3.3 SAS internal cable receptacle connector

SAS internal cables shall use the SAS internal cable receptacle connector. The SAS internal cable receptacle connector is defined in ~~SFF-TBD~~SFF-8482. It attaches to a SAS plug connector, providing contact for the power pins and the primary physical link.

Table 9 defines the signal assignment. The secondary signals, pins S8 through S14, are not used.

5.3.4 SAS backplane receptacle connector

SAS backplanes shall use the SAS backplane receptacle connector. The SAS backplane receptacle connector is defined in ~~SFF-TBD~~SFF-8482. It attaches to a SATA device plug connector, providing contact for only the power pins and the primary signal pins, or a SAS plug connector providing contact for the power pins and both primary and secondary signal pins.

Table 9 defines the signal assignments.

5.3.5 SAS external cable receptacle connector

SAS external cables shall use the SAS external cable receptacle connector. The SAS external cable receptacle connector is defined in ~~SFF-TBDE~~SFF-8483. It includes special SAS keying. It attaches to a SAS external plug connector, providing contact for four physical links.

Table 10 shows how the connector signal pairs are used for applications using one, two, three, or four of the physical links.

Table 10. Physical link usage in wide connector

Physical link	Signal	Signal pin to use based on number of physical links supported by the cable			
		One	Two	Three	Four
1	T1+	S1	S1	S1	S1
	T1-	S2	S2	S2	S2
	R1+	S3	S3	S3	S3
	R1-	S4	S4	S4	S4
3	T3+	N/C	N/C	S5	S5
	T3-	N/C	N/C	S6	S6
	R3+	N/C	N/C	S7	S7
	R3-	N/C	N/C	S8	S8
4	T4+	N/C	N/C	N/C	S9
	T4-	N/C	N/C	N/C	S10
	R4+	N/C	N/C	N/C	S11
	R4-	N/C	N/C	N/C	S12
2	T2+	N/C	S13	S13	S13
	T2-	N/C	S14	S14	S14
	R2+	N/C	S15	S15	S15
	R2-	N/C	S16	S16	S16
Note: N/C: Not connected					

5.3.6 SAS external plug connector

SAS devices with external ports shall use the SAS external plug connector. The SAS external plug connector is defined in ~~SFF-TBDE~~SFF-8483. It includes special SAS keying. It attaches to a SAS external cable receptacle connector, providing contact for the four physical links.

5.4 Cables

5.4.1 SAS internal cables

SAS internal cables shall use SAS internal cable receptacle connectors. The internal cable carries power signals (including READY LED).

SATA cables shall not be used for internal connections between SAS devices.

5.4.2 SAS external cables

The SAS external cable is defined in ~~SFF-TBDE~~SFF-8483. The external cable does not carry power signals (including READY LED).

Although the connector always supports 4 physical links, the cable may support 1, 2, 3, or 4 physical links.

5.5 Backplanes

Backplanes designed with an expander device that supports SATA-only target ports shall support the SATA electrical specification for signals at the device connector. Since the expander device supports SAS electrical specifications, signals from the expander phy to the target phy may have better characteristics; signals driven by the target phy, however, only need to follow SATA rules.

SFF-8460 has the backplane recommendations

5.6 READY LED pin

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

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

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

Table 11 describes the output characteristics of the READY LED signal.

Table 11. Output characteristics of the READY LED signal

State	Output current	Output voltage
Drive LED off	$-100\ \mu A < I_{OH} < 100\ \mu A$	$0 < V_{OH} < 5.5\ V$
Drive LED on	$I_{OL} > 30\ mA$	$0 < V_{OL} < 0.5V$

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

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

a) If rotating media is spinning and the target device is processing a command, the target device shall not assert READY LED. The LED is off.
b) If the rotating media is not spinning, the target device shall assert READY LED with a 2 second cycle of 20% on, 80% off. The target device shall also assert READY LED while processing a command. The intent is that the light appears to be flashing with the 20%/80% cycle when the media is not spinning but that an indication is presented each time activity is in process. The target device may be removed with no danger of mechanical damage in this state.
c) If the target device is in the process of performing a spin-up or spin-down, it shall assert READY LED in such a manner that the light flashes on and off with a 1 second cycle using a 50% duty cycle. The light is on for approximately 0,5 seconds and off for approximately 0,5 seconds.
d) If the target device is ready, it shall assert READY LED continuously except when the target device is processing a command. When processing a command, the target device shall negate READY LED for a period long enough to be detected by an observer. The light is usually on, but flashes off when commands are processed.
e) If the target device is formatting, it shall toggle READY LED on and off with each cylinder change.

During the spin-down process, the target device transitions immediately from pattern d) to pattern c). When the target has reached a state stable enough for it to be removed without mechanical damage, it shall change from pattern c) to pattern b).

5.7 Driver and receiver electrical characteristics

5.7.1 Compliance and reference points

Compliance points are defined as the interoperability points at which the interoperability specifications shall be met. SAS compliance points are always at separable connectors, where SAS devices are interchangeable. Table 12 lists the compliance points.

Table 12. Compliance points

Compliance point	Type	Description
Dt	intra-enclosure	Drive connector; transmit serial port
Dr	intra-enclosure	Drive connector; receive serial port
Ct	inter-enclosure	External cabinet connector; transmit serial port
Cr	inter-enclosure	External cabinet connector; receive serial port

5.7.2 Optional interoperability points

Optional interoperability points are not points where any specifications are required to be met. In the normal case, these points are not accessible; however, devices at these points may be interchangeable, depending on the system design. These values are informative, listed to provide useful guidance values for aspects such as the allowable jitter output, jitter tolerance, and voltage levels at the inputs or outputs of an expander device or an initiator device that is consistent with the budgeting of the same attributes at compliance points. Table 13 lists the optional interoperability points.

Table 13. Optional interoperability points

Optional interoperability points	Description
Xt	Expander phy; transmit serial port
Xr	Expander phy; receive serial port
It	Initiator phy; transmit serial port
Ir	Initiator phy; receive serial port

5.7.3 General interface specification

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

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

This section defines the interfaces of the serial electrical signal at the interoperability points Dt, Dr, Ct, and Cr in a TxRx connection. The existence of a Dt, Dr, Ct, and Cr point is determined by the existence of a connector at that point in a TxRx Connection.

Each conforming SAS phy shall be compatible with this serial electrical interface to allow interoperability within a SAS environment. All TxRx connections described in this clause shall operate within the BER objective (10^{-12}). The parameters specified in this section support meeting that requirement under all conditions including the minimum input and output amplitude levels. The system level BER requirements (e.g., BER < 10^{-12}) often require that the signal quality (eye amplitude and jitter) be better than 10^{-12} (usually a BER < 10^{-14}) in order to achieve the desired system BER with margin.

These specifications are based on ensuring interoperability across multiple vendors supplying the technologies (transceivers, expanders, initiators and cable plants) under the tolerance limits specified in the document. TxRx connections operating at these maximum distances may require some form of equalization (e.g., transmitter pre-emphasis, receiver adaptive equalization, or passive cable equalization) to enable the signal requirements to be met. Longer distances may be obtained by specifically engineering a TxRx connection based on knowledge of the technology characteristics and the conditions under which the TxRx connection is installed and operated (e.g., a closed engineering environment); however, such distance extensions are outside the scope of this standard.

Table 14 defines the general interface characteristics.

Table 14. General interface characteristics

Characteristic	Units	1,5 Gbps	3,0 Gbps
Data rate	MBps	150	300
Nominal bit rate	Mbaud	1 500	3 000
Unit interval (UI)	ps	666,667	333,333
Data rate tolerance at Xr[2b]	ppm	+350/-5 150	+350/-5 150
~~Xr[3]~~	~~ppm~~	~~+350/-150~~	~~+350/-150~~
Data rate tolerance at Dr, Cr, Ir	ppm	+/-100	+/-100
Data rate tolerance at Dt, Ct, It, Xt	ppm	+/-100	+/-100
Media Impedance[1a]	ohm	100	100

Notes:
~~1~~a. The media impedances are the differential, or odd mode, impedances.
~~2~~b. Supports a SATA 1.0 device with spread spectrum clocking.
~~3. Supports a SATA 1.0 device without spread spectrum clocking.~~

5.7.4 Eye masks

5.7.4.1 Eye masks overview

The eye masks shown in this clause shall be interpreted as graphical representations of the voltage and time limits. The time values between X1 and 1-X1 cover all but 10^{-12} of the jitter population. The random content of the total jitter population has a range of +/-7 sigma. Current oscilloscope technology only supports approximately +/-3 sigma, therefore the traditional method of using an oscilloscope to compare the signals against these masks to ascertain jitter compliance is invalid. The oscilloscope remains valid for determining rise/fall times, amplitude, and under and overshoots.

5.7.4.2 Transmitted eye masks at Dt, Ct, It, and Xt

Figure 24 describes the transmitted eye masks at the Dt, Ct, It, and Xt compliance points. These eye masks are directly applicable to compliance points, and are indirectly applicable to reference points.

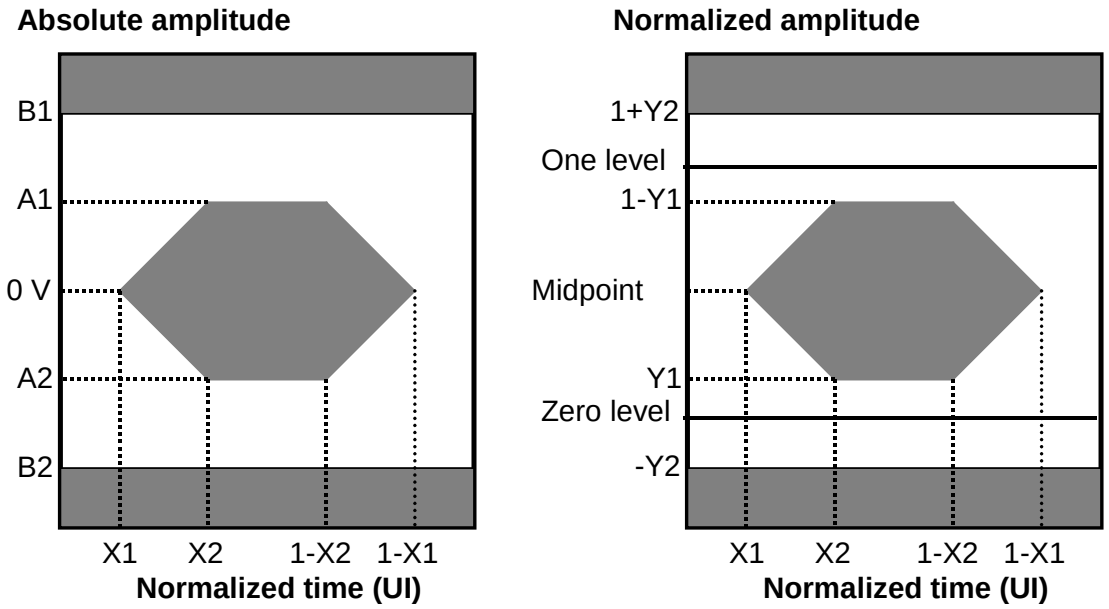

Figure 24. Absolute and normalized amplitude eye diagrams at Dt, Ct, It, and Xt

For unbalanced drivers the absolute amplitude values assume AC coupling between the test load and the driver. Drivers must meet the normalized and the absolute amplitude requirements. The Y1 and Y2 amplitudes allow

signal undershoot ~~of 10% and overshoot of 20%, respectively,~~ relative to the levels determined to be one and zero.

To accurately determine the one and zero levels for use with the normalized mask, use an oscilloscope having an internal histogram capability. Use the voltage histogram capability and set the time limits of the histogram to extend from 0,4 UI to 0,6 UI. Set the voltage limits of the histogram to include only the data associated with the one level. The one level to be used with the normalized mask shall be the mean of the histogram. Repeat this procedure for the zero level.

The eye diagram mask applies to jitter after application of a single pole high-pass frequency-weighting function, which progressively attenuates jitter at 20 dB/decade below a frequency of bit rate/1 667.

5.7.4.3 Delivered (receive) eye mask at Dr, Cr, Ir, and Xr

Figure 25 describes the delivered (received) eye mask. This eye mask applies to jitter after the application of a single pole high-pass frequency-weighting function, which progressively attenuates jitter at 20 dB/decade below a frequency of the bit rate/1 667.

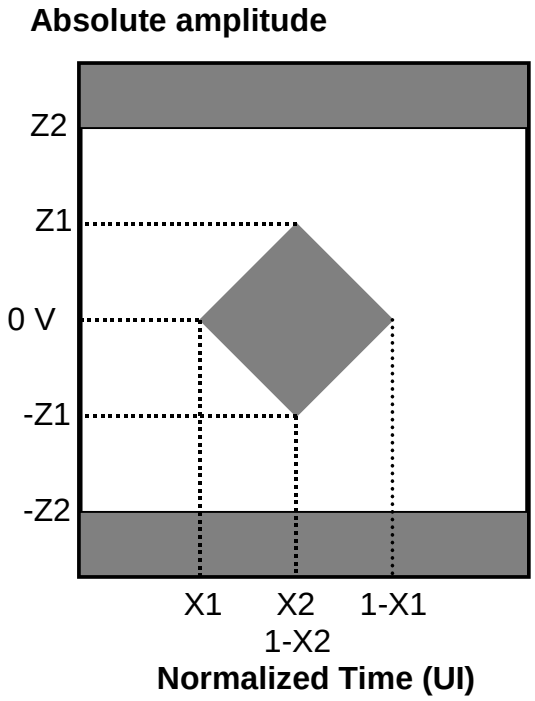

Figure 25. Eye mask at Dr, Cr, Ir, and Xr

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

5.7.4.4 Jitter tolerance masks

Tolerance eye masks at Dt, Ct, It, and Xt shall be based on Figure 26 and shall be constructed using the X2, Z1 and Z2 values given in Table 18. X1 values shall be half the value for total jitter given in the tables for jitter value frequencies above bit rate/1 667.

Note that the tolerance masks are identical to the output masks except that the X1 and X2 values are each increased by half the amount of the sinusoidal jitter values given in the jitter tolerance tables.

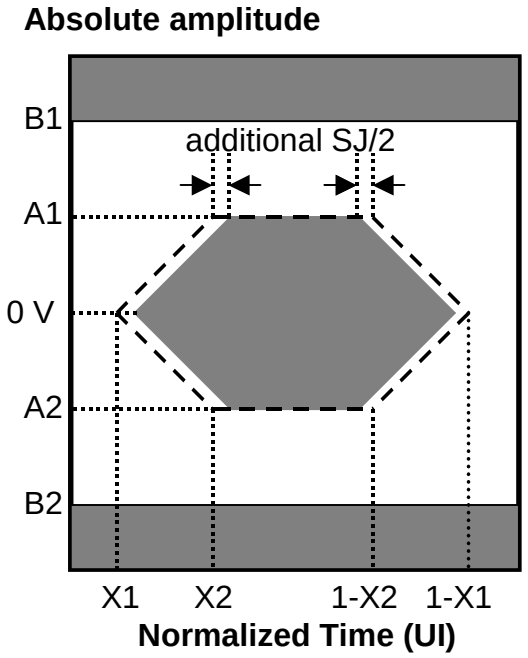

Figure 26. Deriving a tolerance mask at Dt, Ct, It, or Xt

Receiver Tolerance eye masks at Dr, Cr, Ir, and Xr shall be based on Figure 27 and shall be constructed using the X2 and Z2 values given in Table 18. X1 shall be half the value for total jitter given in these tables for jitter frequencies above bit rate/1 667.

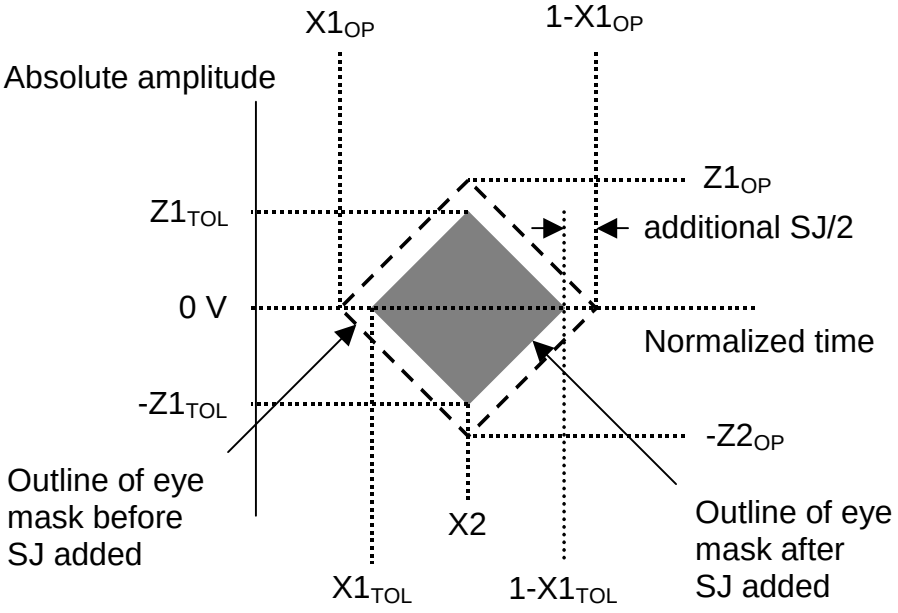

Figure 27. Deriving a tolerance mask at Dr, Cr, Ir, or Xr

However, the leading and trailing edge slopes of Figure 25 shall be preserved. As a result the amplitude value of Z1 is less than that given in Table 16 and must therefore be calculated from those slopes as follows:

$Z1_{Tol} = Z1_{OP} \times (X2_{OP} – (0,5 \times (\text{additional SJ UI})) - X1_{OP})/(X2_{OP} - X1_{OP})$

$Z1_{Tol}$ = value for Z1 to be used for the tolerance masks

$Z1_{OP}$, $X1_{OP}$, and $X2_{OP}$ are the values in Table 18 for Z1, X1, and X2

Note that the X1 points in the receive tolerance masks are greater than the X1 points in the output masks, again due to the addition of sinusoidal jitter.

Figure 28 defines the sinusoidal jitter mask.

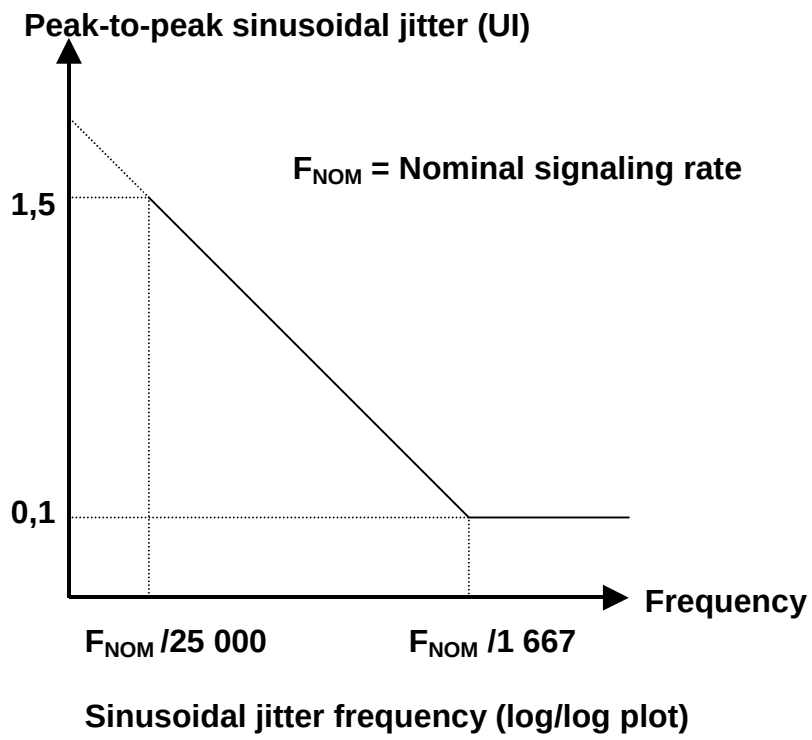

Figure 28. Sinusoidal jitter mask

5.7.5 Transmitted signal characteristics

This section defines the inter-operability requirements of the transmitted signal at the driver end of a TxRx connection as measured into the test load specified in Figure 29. All specifications are based on differential measurements.

Table 15 specifies the transmitted signal characteristics at Tx compliance points and optional Tx compliance points.

Table 15. Transmitted signal characteristics at Tx compliance points

Compliance point	Signal characteristic	Units	SATA 1.0	1,5 Gbps	3,0 Gbps
Dt	Jitter Output Eye Mask (see Figure 24)[a]	N/A	N/A	See Table 17	See Table 17
	(B1-B2)[b]	mV(P-P)	N/A	1 200	1 600
	(A1-A2)[c]	mV(P-P)	N/A	600	800
	X1[d]	UI	N/A	0,125	0,175
	X2	UI	N/A	X1 + 0,21	X1 + 0,21
	Skew[hg]	ps	N/A	20	10
	Tx Off Voltage[e]	mV(P-P)	N/A	< 50	< 50
	Y1	N/A	N/A	0,1	**[TBD]**
	Y2	N/A	N/A	0,2	**[TBD]**
	Maximum rise/fall time[f]	ps	N/A	273	137
	Minimum rise/fall time[f]	ps	N/A	133	67
Ct	Jitter Output Eye Mask (see Figure 24)[a]	N/A	N/A	See Table 17	See Table 17
	(B1-B2)[b]	mV(P-P)	N/A	1 600	1 600
	(A1-A2)[c]	mV(P-P)	N/A	800	800
	X1[d]	UI	N/A	0,125	0,175
	X2	UI	N/A	X1 + 0,21	X1 + 0,21
	Skew[g]	ps	N/A	20	10
	Tx off voltage[e]	mV(P-P)	N/A	< 50	< 50
	Y1	N/A	N/A	0,1	**[TBD]**
	Y2	N/A	N/A	0,2	**[TBD]**
	Maximum rise/fall time[f]	ps	N/A	273	137
	Minimum rise/fall time[f]	ps	N/A	133	67
~~Xr~~Xt, It optional compliance points	Jitter output eye mask (see Figure 24)[a]	N/A	See Table 17	See Table 17	See Table 17
	(B1-B2)[2]	mV(P-P)	900[h]	1 200	1 600
	(A1-A2)[3]	mV(P-P)	600[h]	600	800
	X1[d]	UI	0,125	0,125	0,175
	X2	UI	X1 + 0,21	X1 + 0,21	X1 + 0,21
	Skew[g]	ps	20	20	20
	Tx off voltage[e]	mV(P-P)	< 50	< 50	< 50
	Y1	N/A	0,1	0,1	**[TBD]**
	Y2	N/A	0,2	0,2	**[TBD]**
	Maximum rise/fall time[f]	ps	273	273	137
	Minimum rise/fall time[f]	ps	133	133	67

Notes:
a) Drivers shall meet both the absolute and normalized amplitude requirements.
b) The B amplitude specification identifies the maximum signal peak (including overshoots) that can be delivered into a resistive load matching those shown in Figure 29.
c) The minimum allowed peak-to-peak eye amplitude opening that shall be delivered into a resistive load matching those shown in Figure 29, is equal to the A1-A2 amplitude shown above.
d) The value of X1 shall be half the value for total jitter given in Table 17. The test or analysis shall include for the effects of a single pole high-pass frequency-weighting function, which progressively attenuates jitter at 20 dB/decade below a frequency of bit rate/1 667. The value for X1 applies at a total jitter probability of 10^{-12}. At this level of probability direct visual comparison between the mask and actual signals is not a valid method for determining compliance with the jitter output requirements.
e) The transmitter off voltage is the maximum voltage measured at point compliance points Ct and Dt and reference points It and Xt when the transmitter is logically turned off or is unpowered.
f) Rise/fall time measurements to be made using an oscilloscope with a bandwidth including probes of at least 1,8 times the baud rate.
g) Skew measurements shall be made using an oscilloscope with a bandwidth including probes of at least 1,8 times the baud rate.
h) Minimum and maximum recommended launch levels. Implementation should consider possible loss to achieve SATA 1.0 compliance at the drive connector. At the drive connector, SATA 1.0 allows 600 mV peak-to-peak maximum, 325 mV peak-to-peak minimum.

[Editor's note: The preemphasis specs for Y1 and Y2 appropriate for 3,0 Gbps are still under investigation. More than 20% seems likely.]

5.7.6 Received signal characteristics

Table 16 defines the inter-operability requirements of the delivered signal at the receiver end of a TxRx connection as measured into the test load specified in Figure 29. It also defines optional Rx compliance points. All specifications are based on differential measurements.

Table 16. Delivered signal characteristic at Rx compliance points

Compliance point	Signal characteristic	Units	SATA 1.0	1,5 Gbps	3,0 Gbps
Dr	Jitter output eye mask[b] (see Figure 25)	N/A	N/A	See Table 17	See Table 17
	2 x Z2	mV(P-P)	N/A	1 200	1 600
	2 x Z1	mV(P-P)	N/A	325	275
	X1[a]	UI	N/A	0,25	0,25
	X2	UI	N/A	0,50	0,50
	Skew[c]	ps	N/A	50	25
	Max voltage (non-op)	mV(P-P)	N/A	2 000	2 000
	OOB detect threshold min	mV(P-P)	N/A	150	150
	Max near-end crosstalk	mV(P-P)	N/A	100	100
Cr	Jitter output eye mask[b] (see Figure 25)		N/A	See Table 17	See Table 17
	2 x Z2	mV(P-P)	N/A	1 600	1 600
	2 x Z1	mV(P-P)	N/A	275	275
	X1[a]	UI	N/A	0,275	0,275
	X2	UI	N/A	0,50	0,50
	Skew[c]	ps	N/A	50	25
	Max voltage (non-op)	mV(P-P)	N/A	2 000	2 000
	OOB detect threshold min	mV(P-P)	N/A	150	150
	Max near-end crosstalk	mV(P-P)	N/A	100	100
Xr, Ir optional compliance points	Jitter output eye mask[b] (see Figure 25)		See Table 17	See Table 17	See Table 17
	2 x Z2	mV(P-P)	600	1 200	1 600
	2 x Z1	mV(P-P)	225	325	275
	X1[a]	UI	0,275	0,275	0,275
	X2	UI	0,50	0.50	0,50
	Skew[c]	ps	50	50	25
	Max voltage (non-op)	mV(P-P)	2 000	2 000	2 000
	OOB detect threshold min	mV(P-P)	100	150	150
	Max near-end crosstalk	mV(P-P)	< 50	100	100

Notes:

a) The value for X1 shall be half the value given for total jitter in Table 17. The test or analysis shall include the effects of a single pole high-pass frequency-weighting function, which progressively attenuates jitter at 20 dB/decade below a frequency of bit rate/1 667.

b) The value for X1 applies at a total jitter probability of 10^{-12}. At this level of probability direct visual comparison between the mask and actual signals is not a valid method for determining compliance with the jitter output requirements.

c) Skew measurements shall be made using an oscilloscope with a bandwidth including probes of at least 1,8 times the baud rate. Maximum transmitter and maximum interconnect skew is assumed.

5.7.7 Jitter output

Table 17 defines the allowable output jitter at the compliance points. Entries for inter-enclosure TxRx connections and for intra-enclosure TxRx connections are covered.

Table 17. Jitter output compliance points

Compliance point	1,5 Gbps		3 Gbps	
	Deterministic jitter	Total jitter	Deterministic jitter	Total jitter
Dt	0,15	0,25	0,20	0,30
Dr	0,30	0,50	0,35	0,50
Ct	0,15	0,25	0,20	0,35
Cr	0,35	0,55	0,35	0,55
It	0,15	0,25	0,20	0,35
Ir	0,35	0,55	0,35	0,55
Xt	0,15	0,25	0,20	0,35
Xr	0,35	0,55	0,35	0,55

Notes:

It, Ir, Xt, and Xr are optional compliance points.

Units are in UI.

The values for jitter in this section are measured at the average amplitude point.

Total jitter (TJ) is the sum of deterministic jitter (DJ) and random jitter (RJ). If the actual deterministic jitter is less than the maximum specified, then the random jitter may increase as long as the total jitter does not exceed the specified maximum total jitter.

Total jitter is specified at a probability of 10^{-12}.

The deterministic and total values in this table apply to jitter after application of a single pole high-pass frequency-weighting function, which progressively attenuates jitter at 20 dB/decade below a frequency of bit rate/1 667.

Values at the various points are determined by the application.

If total jitter delivered at any point is less than the maximum allowed, then the jitter distribution of the signals is allowed to be asymmetric. The total jitter plus the magnitude of the asymmetry must not exceed the allowed maximum total jitter. The numerical difference between the average of the peaks (at 10^{-12}) and the average of the individual events is the measure of the asymmetry. Jitter peak-to-peak measured < (TJ max - |Asymmetry|).

SERDES voltage sensitivity to switching power supply noise: Jitter values specified above shall be achieved even in an environment of switching noise pulses with peak-to-peak amplitude of 50 mV, rise/fall times from 10 ns to 500 ps, and multiple pulse duration of up to 2,0 µs maximum. A real-time sampling scope is required to make this measurement.

5.7.8 Jitter tolerance

Table 18 defines the minimum allowable jitter tolerance at the required and optional compliance points. Entries for inter-enclosure TxRx connections and for intra-enclosure TxRx connections are covered. SERDES voltage sensitivity to switching power supply noise: Jitter values specified above shall be achieved even in an environment of switching noise pulses with peak-to-peak amplitude of 50 mV, rise/fall times from 10 ns to 500 ps, and multiple pulse duration of up to 2,0 µs maximum. A real-time sampling scope is required to make this measurement.

Table 18. Jitter tolerance compliance points

Compliance point	1,5 Gbps			3,0 Gbps		
	Sinusoidal jitter[b]	Deterministic jitter[c,f]	Total jitter[f]	Sinusoidal jitter[d]	Deterministic jitter[e,f]	Total jitter[f]
Ct	0,10	0,15	0,35	0,10	0,20	0,45
Cr	0,10	0,10	0,65	0,10	0,35	0,65
Dt	0,10	0,15	0,35	0,10	0,20	0,45
Dr	0,10	0,30	0,65	0,10	0,30	0,65
It, Xt[g]	0,10	0,15	0,35	0,10	0,20	0,45
Ir, Xr[g]	0,10	0,35	0,65	0,10	0,35	0,65

Notes:

a) Units are in UI.

b) Sinusoidal swept frequency: 900 kHz to > 5 MHz. The jitter values given are normative for a combination of DJ, RJ, and SJ (sinusoidal jitter) which receivers shall be able to tolerate without exceeding a BER of 10^{-12}.

c) Deterministic jitter: 900 kHz to 750 MHz. No value is given for random jitter. For compliance with this standard, the actual random jitter amplitude shall be the value that brings total jitter to the stated value at a probability of 10^{-12}.

d) Sinusoidal swept frequency: 1 800 kHz to > 5 MHz. Receivers shall tolerate sinusoidal jitter of progressively greater amplitude at lower frequencies, according to the mask in Figure 28 with the same DJ and RJ levels as were used in the high frequency sweep.

e) Deterministic jitter: 1 800 kHz to 1 500 MHz. The additional 0,1 UI of sinusoidal jitter is added to ensure the receiver has sufficient operating margin in the presence of external interference.

f) The deterministic and total values in this table apply to jitter after application of a single pole high-pass frequency-weighting function, which progressively attenuates jitter at 20 dB/decade below a frequency of bit rate/1 667.

g) Optional compliance point.

5.7.9 Impedance specifications

Table 19 defines impedance requirements.

Table 19. Impedance requirements

Requirement	Units	1,5 Gbps	3,0 Gbps
Time domain reflectometer rise time[a,b]	ps	85	85
Common mode impedance[b,c,d]	ohm	32,5 +/-7,5	32,5 +/-7,5
Cable pair matching[b,c,d]	ohm	+/- 5	+/- 5
Media (PCB or cable)[b,c,d]	ohm	100 +/-10	100 +/- 10
Through connection[b]	ohm	100 +/-15	100 +/-15
Receiver termination[b,e,f]	ohm	100 +/-15	100 +/-15
Transmitter source termination[b,e,f]	ohm	100 +/-15	100 +/-15

Notes:

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

b) All measurements are made through mated connector pairs.

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

d) Where the media has an electrical length of > 4 ns the procedure detailed in SFF-8410, or an equivalent procedure, shall be used to determine the impedance.

e) The receiver termination impedance specification applies to each and every receiver in a TxRx connection and covers all time points between the connector nearest the receiver, the receiver, and the transmission line terminator. This measurement shall be made from that connector.

f) At the time point corresponding to the connection of the receiver to the transmission line the input capacitance of the receiver and its connection to the transmission line may cause the measured impedance to fall below the minimum impedances specified in this table. The area of the dip caused by this capacitance is directly proportional to the capacitance. An approximate value for the area is given by the product of the amplitude of the dip (in units of rho, the reflection coefficient) and its width~~-1~~[a] (in picoseconds) measured at the half amplitude point. The product calculated by this method shall not be greater than 150 ps. The amplitude is defined as being the difference in rho between the rho at the nominal impedance and the rho at the minimum impedance point.

5.7.10 Electrical TxRx connections

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

Each electrical TxRx connection segment shall comply with the impedance requirements of Table 19 for the media from which they are formed. An optional equalizer network, when present in a TxRx connection, shall exist and operate as part of the cable plant.

TxRx connections that are composed entirely of electrically conducting media shall be applied only to homogenous ground applications, such as between devices within an enclosure or rack, or between enclosures interconnected by a common ground return or ground plane. This restriction minimizes safety and interference concerns caused by any voltage differences that could otherwise exist between equipment grounds.

5.7.11 Driver characteristics

For all inter-enclosure TxRx connections, the output shall be AC coupled to the cable through a transmission network.

For intra-enclosure TxRx connections the expander shall be AC coupled to the media. Other drivers may be AC or DC coupled.

The driver shall have the output voltages and timing listed in Table 15 and Table 17 measured at the designated interoperability points. The default point is Ct for inter-cabinet TxRx connections and Dt for intra-cabinet TxRx connections. The measurements shall be made across a load equivalent to that shown in Figure 29.

The relevant eye diagrams are given in 5.7.4. The normalized amplitudes, Y1 and Y2, allow signal undershoots ~~of 10% and overshoots of 20%, respectively~~. The driver shall meet both the normalized and absolute values.

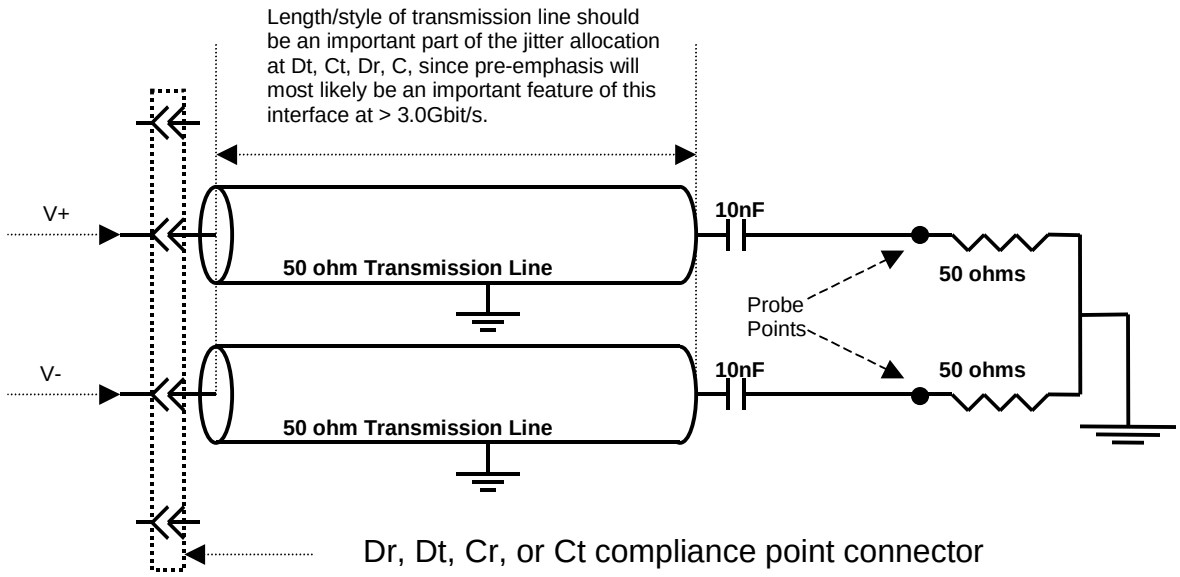

Figure 29. Test loads

5.7.12 Receiver characteristics

The receiver shall be AC-coupled to the media through a receive network. The receive network shall terminate the TxRx connection by a 100 ohm equivalent impedance as specified in Table 19.

The receiver shall operate within the BER objective (10^{-12}) when a SAS signal with valid voltage and timing characteristics is delivered to the interoperability point from a 100 ohm source. The delivered SAS signal shall be considered valid if it meets the voltage and timing limits specified in Table 17 when measured across a load equivalent to those of Figure 29.

Additionally the receiver shall also operate within the BER objective when the signal at a receiving phy has the additional sinusoidal jitter present that is specified in the Table 18. Jitter tolerance figures are given in 5.7.4.4 for all interoperability points in a TxRx connection. The figures given assume that any external interference occurs prior to the point at which the test is applied. When testing the jitter tolerance capability of a receiver the additional 0,1 UI of sinusoidal jitter may be reduced by an amount proportional to the actual externally induced interference between the application point of the test and the input to the receiving phy. The addition of additional jitter reduces the eye opening in both voltage and time; see the Jitter tolerance masks in 5.7.8.

5.7.13 Spread spectrum clocking

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

5.8 Non-tracking clock architecture

Phys shall be designed with a non-tracking clock architecture, where the receive clock has no relationship to the transmit clock. Expander phys that support being attached to SATA target phys shall tolerate clock tracking by the SATA target phy.

6 Phy layer

The link layer transmits and receives primitives and frames. It is divided into two portions:

a) SAS link layer - functions shared by SSP, STP, and SMP protocols; and

b) SSP, STP, and SMP link layers - functions specific to each transport protocol.

6.1 Encoding (8b/10b)

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

All characters transferred in SAS are grouped into 4 byte sequences called dwords. A primitive is a dword whose first byte is a control character and remaining bytes are data characters.

Like SATA, primitives are defined with both negative and positive starting disparity. Unlike SATA, SAS defines more than one primitive starting with the K28.5 control character. Table 20 shows special character usage in SAS and SATA.

Table 20. Special character usage

First character	Usage in SATA	Usage in SAS
K28.3	All primitives except ALIGN	STP primitives
K28.5	ALIGN	ALIGN and all SAS, SSP and SMP primitives
Dxx.y	Data	Data

Disparity shall be maintained separately on each physical link. Expander devices shall convert incoming 10 bit characters to 8 bit bytes and generate the 10 bit character with correct disparity for the output physical link. Physical links do not necessarily begin operation with the same disparity after the reset sequence.

6.2 Bit order

Dwords transmitted in a STP connection shall be transmitted in the bit order specified by SATA.

[Editor's note: where the scrambler fits in, before or after bit transposition, is TBD.]

Dwords for other protocols and outside of connections shall be transmitted in the bit order in Figure 30.

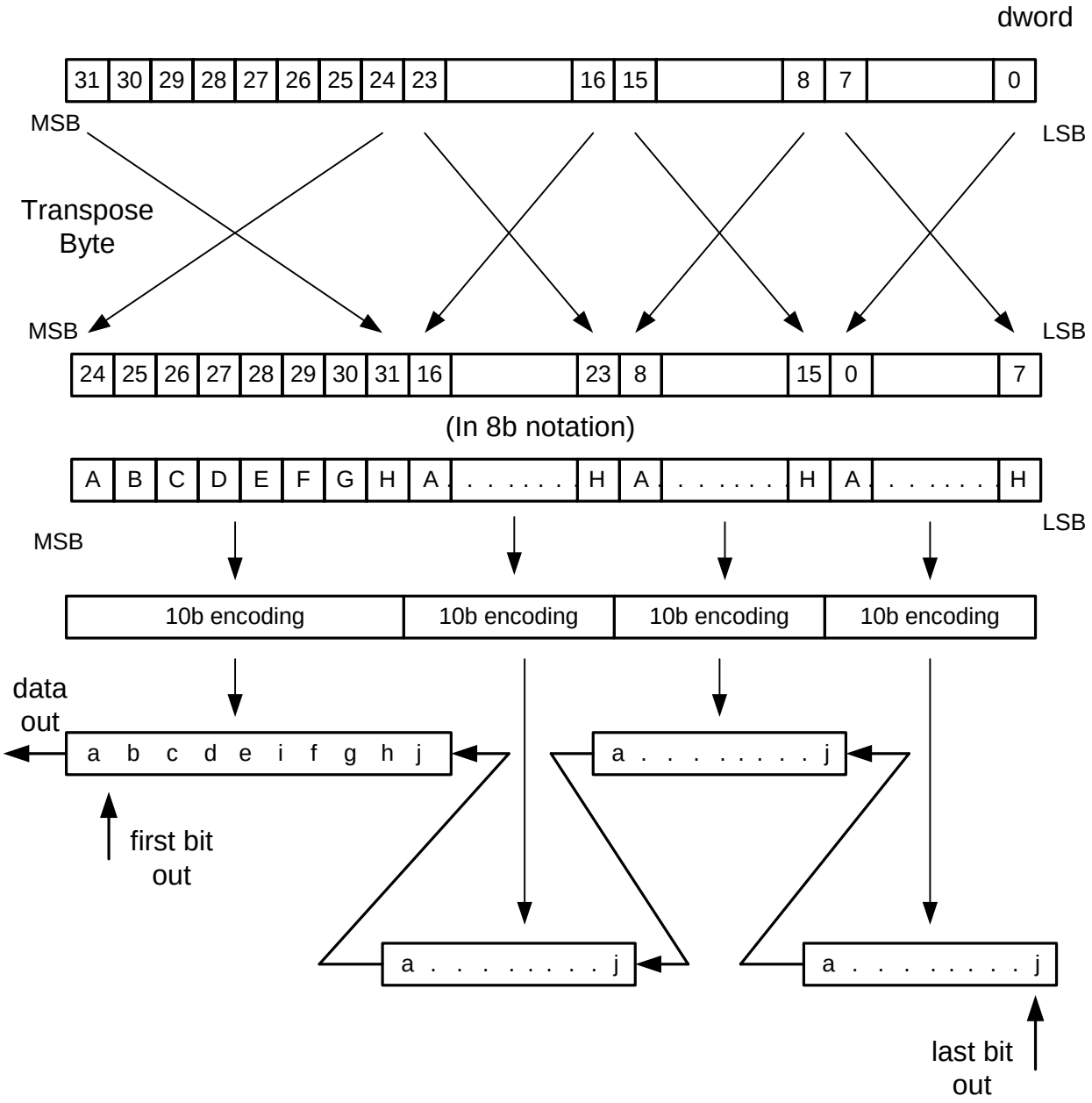

Figure 30. SAS bit transmission order

Figure 31 shows SAS bit reception order.

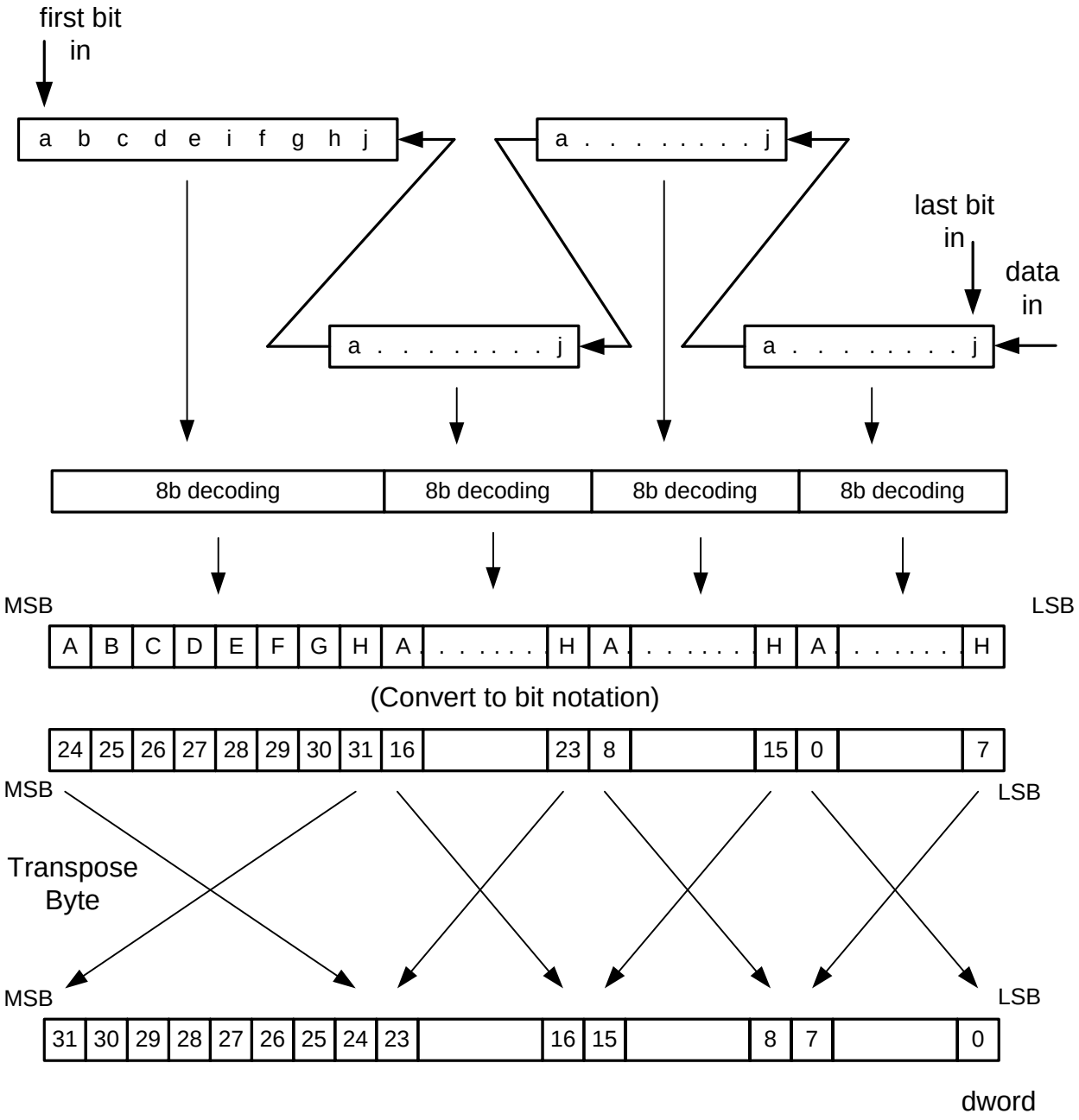

Figure 31. SAS bit reception order

6.3 Out of band (OOB) signals

SATA defines “out of band” (OOB) signals used for phy resets. OOB signals are low-speed signal patterns detected by the phy that do not appear in normal data streams. They consist of 160 UI (106.7 ns) bursts of ALIGNs followed by defined amounts of idle time. The signals are differentiated by the length of idle time between the bursts of ALIGNs.

SATA defines two OOB signals: COMINIT/COMRESET and COMWAKE. SAS devices identify themselves with a SAS-specific OOB signal called COMSAS.

[Editor's note: COMSAS timing is under debate again.]

Table 21 describes the SATA OOB signals. Each signal shall be sent six times. Each signal shall be received four times to be detected.

Table 21. OOB signal transmitter requirements

Signal	Idle time minimum	Idle time nominal	Idle time maximum	Notes
COMWAKE	55 ns	106.7 ns (160 UIs)	175 ns	
COMINIT/COMRESET	175 ns	320 ns (480 UIs)	525 ns	Named COMRESET when sent by an initiator, and named COMINIT when sent by a target.

Expanders shall not pass through OOB signals. An expander device shall run the link reset sequence independently on each physical link (initiator phy to expander phy, expander phy to expander phy, or expander phy to target phy).

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.

Figure 32~~Figure 32~~ illustrates the timing required for the generation and detection of all the SAS OOB signals. The signals labeled ComXxxDet and ComXxxSent (where Xxx is Wake, Init, or Sas), are inputs to the SAS phy Initialization state machine. ComXxxDet is an output from the OOB detection circuitry that indicates the detection of an incoming COMINIT, COMWAKE, or COMSAS sequence. ComXxxSent is an output from the OOB generation circuitry that indicates that a burst of six COMWAKE, COMINIT, or COMSAS sequences has been output onto the link.

COMINIT as defined in the SATA specification is the same as COMRESET, and they are used in this standard interchangeably.

A SAS transmitter should send OOB sequences with ALIGNs at the G1 rate, but may send them at its slowest supported rate and shall not send them at a rate faster than its slowest supported rates.

A SAS receiver shall detect OOB sequences comprised of ALIGNs transmitted at any rate up to its highest supported rate. This includes rates below its lowest supported rates.

The ALIGNs used in OOB sequences are not required to be at generation 1 (G1) rates (1,5 Gbps), as this rate might not be supported in future generations of SAS devices. The ALIGNs are only required to generate an envelope for the detection circuitry, as required for any signaling that may be AC coupled. If G2 ALIGNs are used, the number of ALIGNs doubles compared with G1 ALIGNs. The time for these bursts and spaces is important, not the absolute number of ALIGNs in each burst.

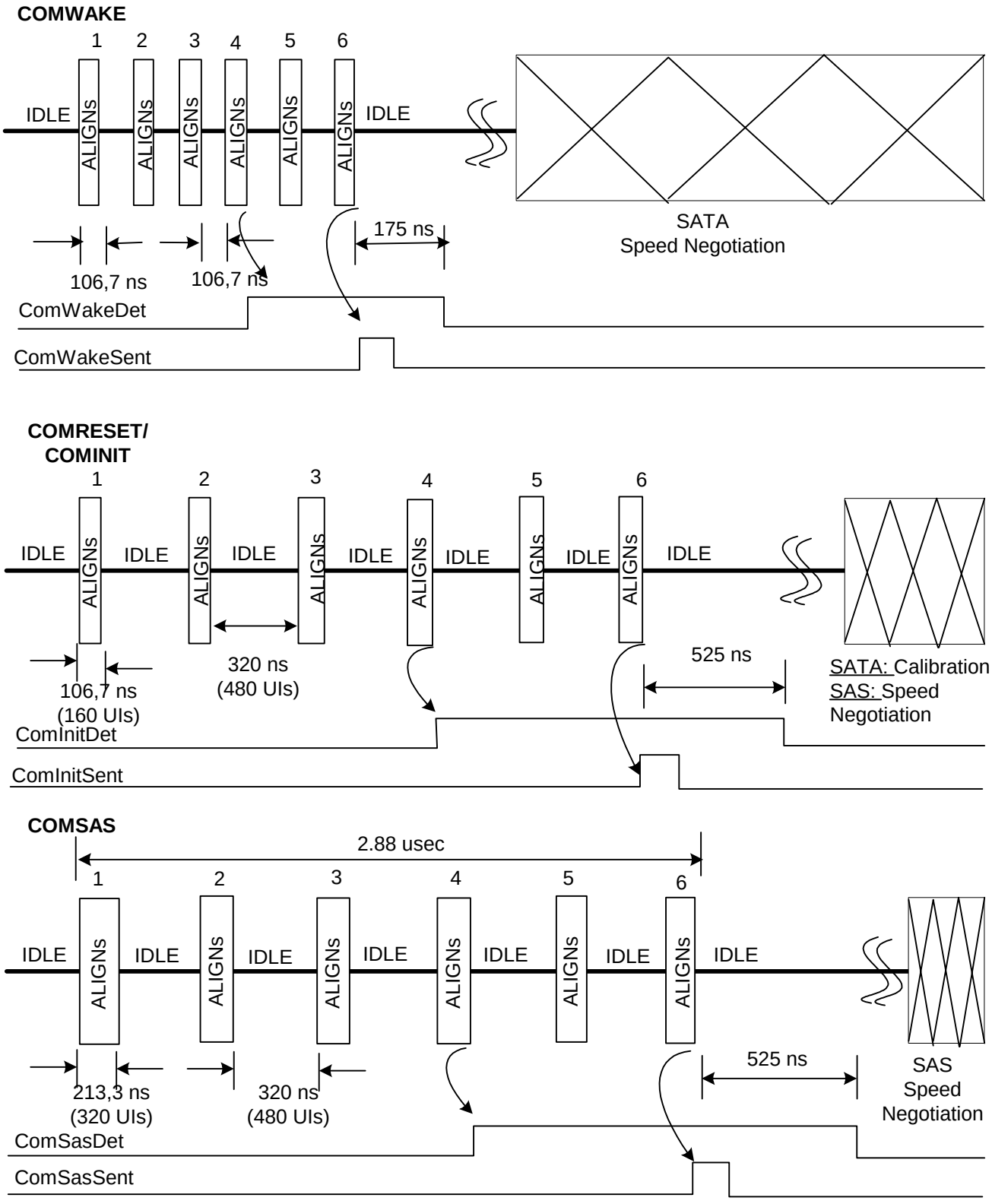

Figure 32. Phy reset sequence generation and detection

Table 22 defines the timing specifications for OOB signals.

Table 22. OOB signal timing specifications

Parameter	Nominal	Minimum	Maximum	Comments
COMSAS detector off threshold		175 ns	525 ns	Detector shall reject all bursts with spacings outside of this range.
COMSAS detector on threshold	320 ns	304 ns	336 ns	Detector shall detect all burst with spacings meeting this period.
COMSAS transmit spacing	320 ns	310,4 ns	329,6 ns	As measured from receiver thresholds of last and first differential crosspoints of the burst.
Hot-Plug Timeout	100 ms	10 ms	500 ms	The hot plug retry timer.
Rate change detect		0 ns	320 ns	Transmitter shall change speeds within this time.
Unit interval at 1,5 Gbps	666,667 ps	666,600 ps	666,734 ps	Based on clock tolerance.
Speed Negotiation Window	109,22 µs	109,20 µs	109,24 µs	The speed negotiation window is UI * 4096 * 40.
COMSAS detect timeout period	3,5 µs			Timer.
ALIGN Detect Timeout			880 µs	Allows over 32 768 G1 dwords.

6.4 Phy reset sequences

6.4.1 SATA phy reset sequence

Figure 33 depicts the link reset sequence between a SATA initiator device and target device. See SATA for detailed requirements.

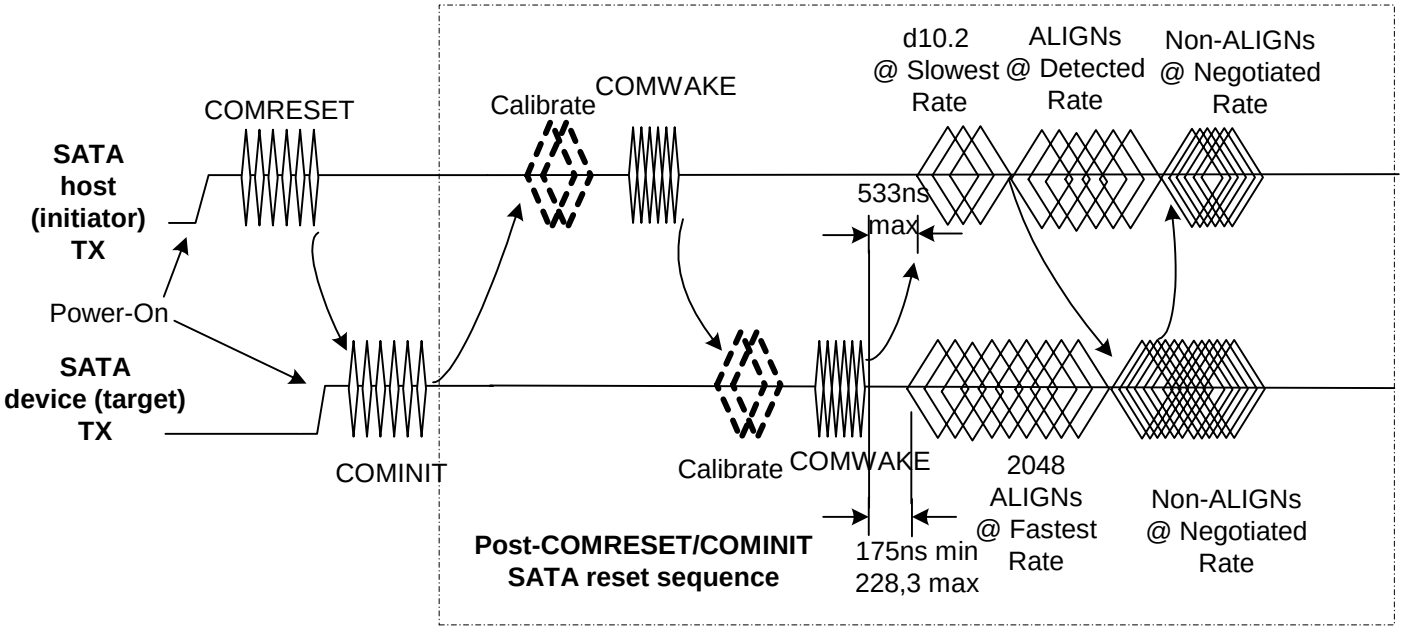

Figure 33. SATA phy reset sequence

6.4.2 SAS to SATA reset sequence

SAS phys shall send COMSAS during the SATA calibration phase to identify themselves as SAS phys rather than SATA phys.

Figure 34~~Figure 34~~ shows a reset sequence between a SAS phy and a SATA phy (i.e., between an expander device and a SATA target device, or between a SAS initiator device and a SATA target device). The two possible cases are presented. The first case is that a legacy SATA phy misinterprets the COMSAS to be a COMRESET and responds with a COMINIT. The second case is that the SATA phy ignores the COMSAS and provides no response within the COMSAS detect timeout of 3,5 µs. The SAS phy state machine treats these two cases the same, and determines that a SATA phy is attached. The normal SATA reset sequence shall be used thereafter.

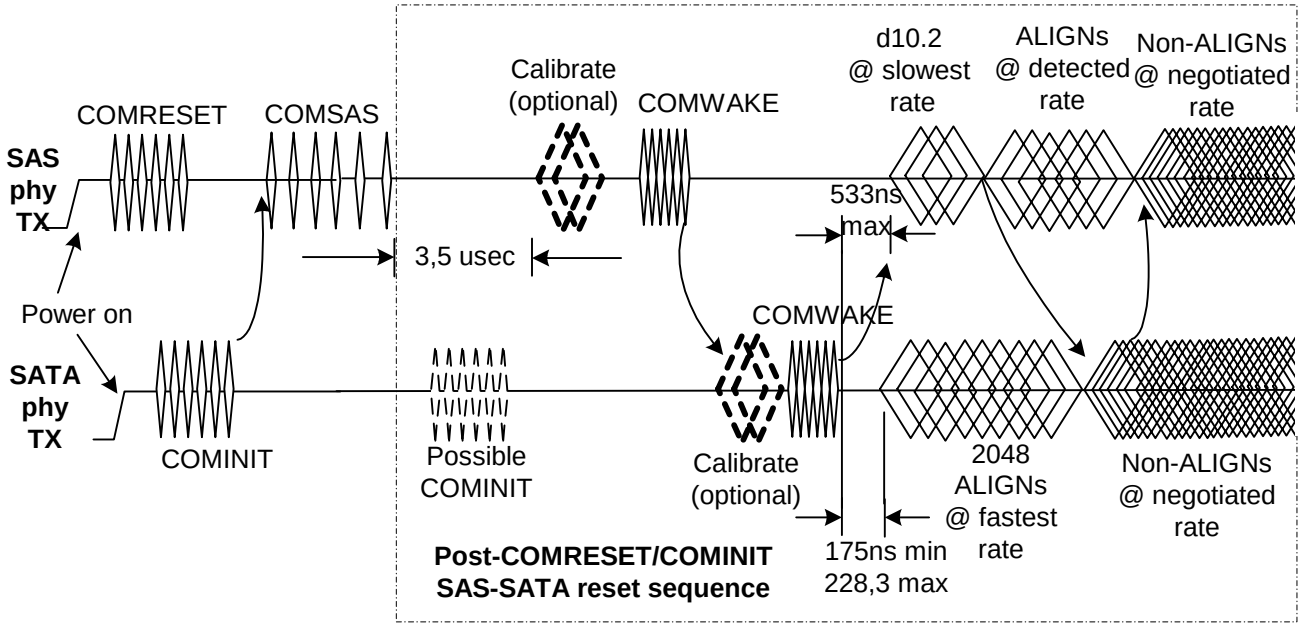

Figure 34. SAS device to SATA device phy reset sequence

6.4.3 SAS to SAS reset sequence

A SAS device shall distinguish between COMINIT and COMSAS and continue with a SAS speed negotiation sequence rather than the SAS reset sequence.

Figure 35~~Figure 35~~ illustrates several different power-up sequences based on whether one SAS phy powers up before, after, or at the same time as the SAS phy to which it is attached.

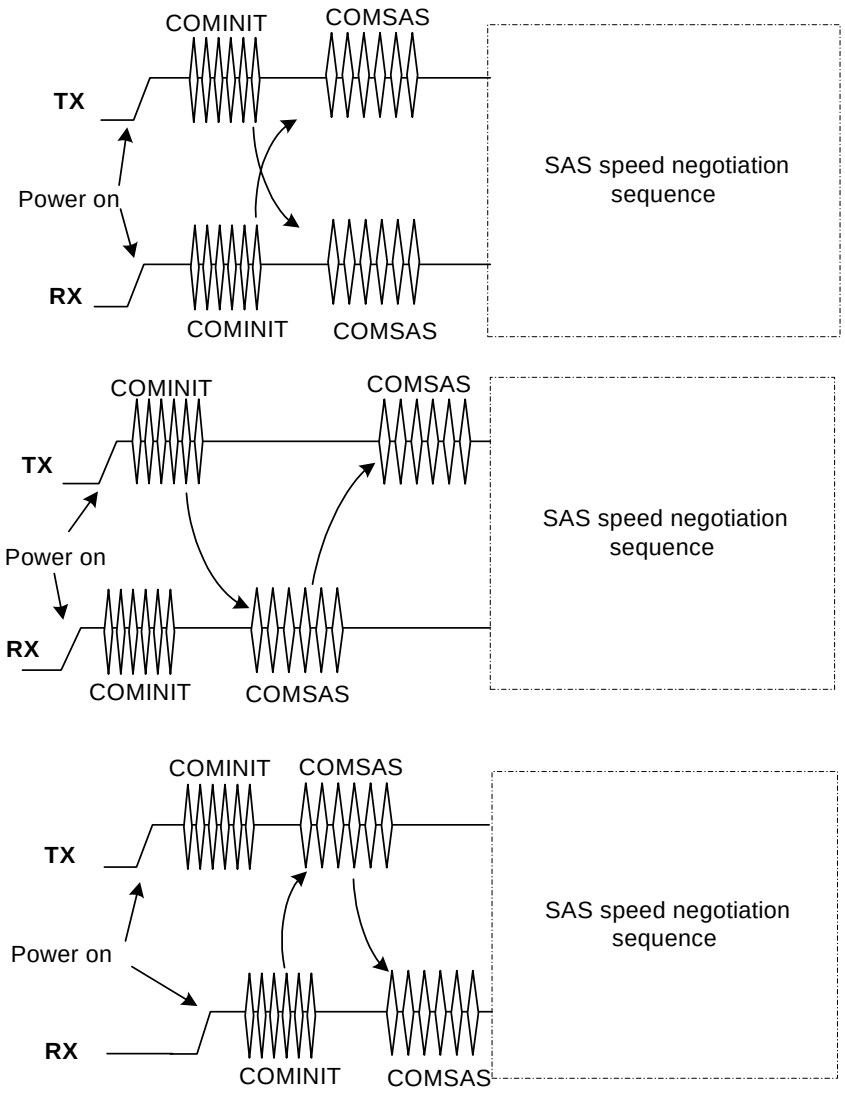

Figure 35. SAS-to-SAS phy reset sequences

The SAS- speed negotiaton sequence is a peer-to-peer negotiation technique that doesn't assume initiator device and target device roles like the SATA speed negotiation sequence.

Figure 36~~Figure 36~~ shows a speed negotiation between a SAS phy that supports speeds G1 thru G3 attached to a SAS phy that only supports speeds G1 and G2.

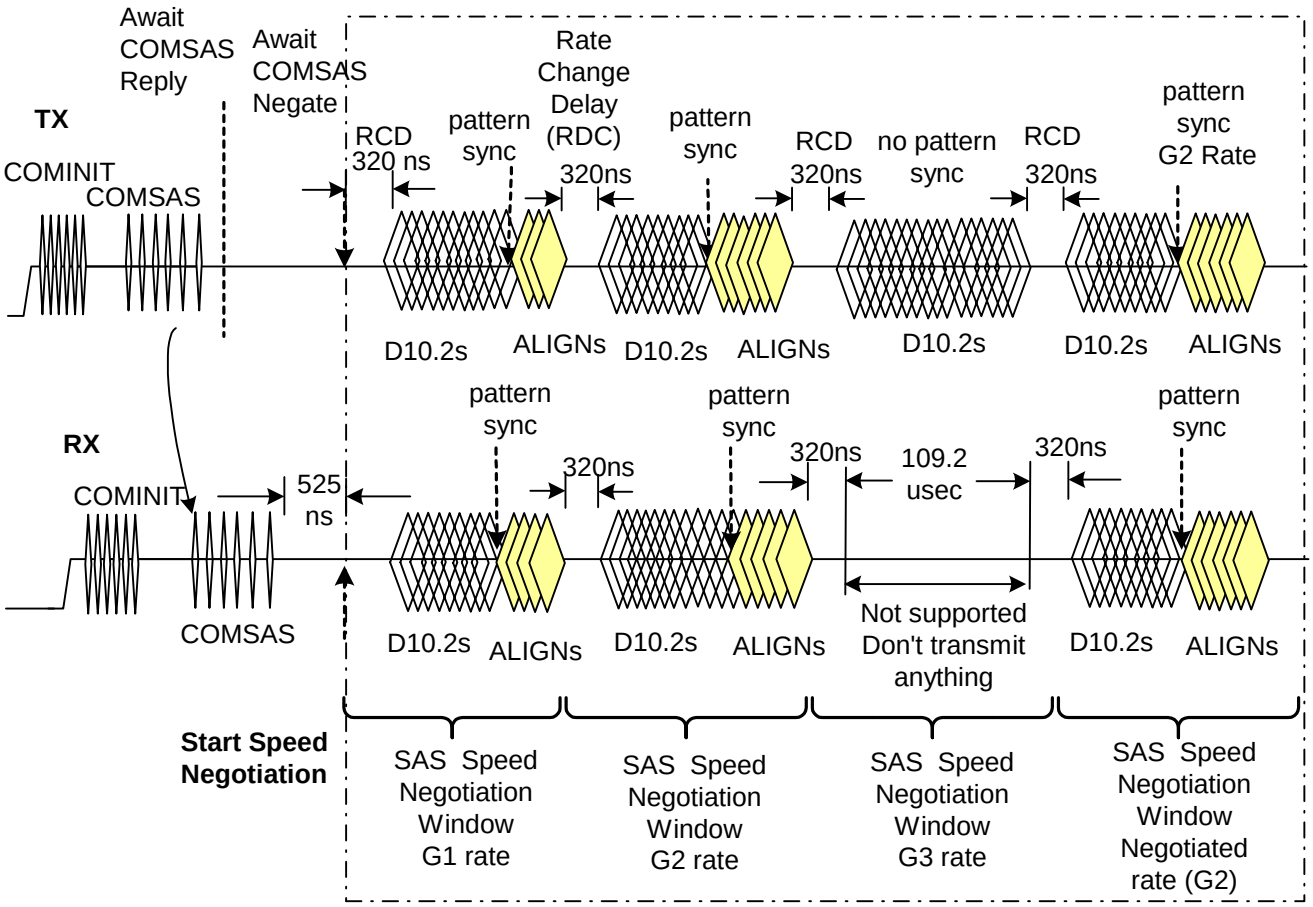

Figure 36. SAS speed negotiation sequence (example 1)

Figure 37~~Figure 37~~ shows speed negotiation between a SAS device that supports G1 thru G3 rates and a SAS device that only supports the G2 rate.

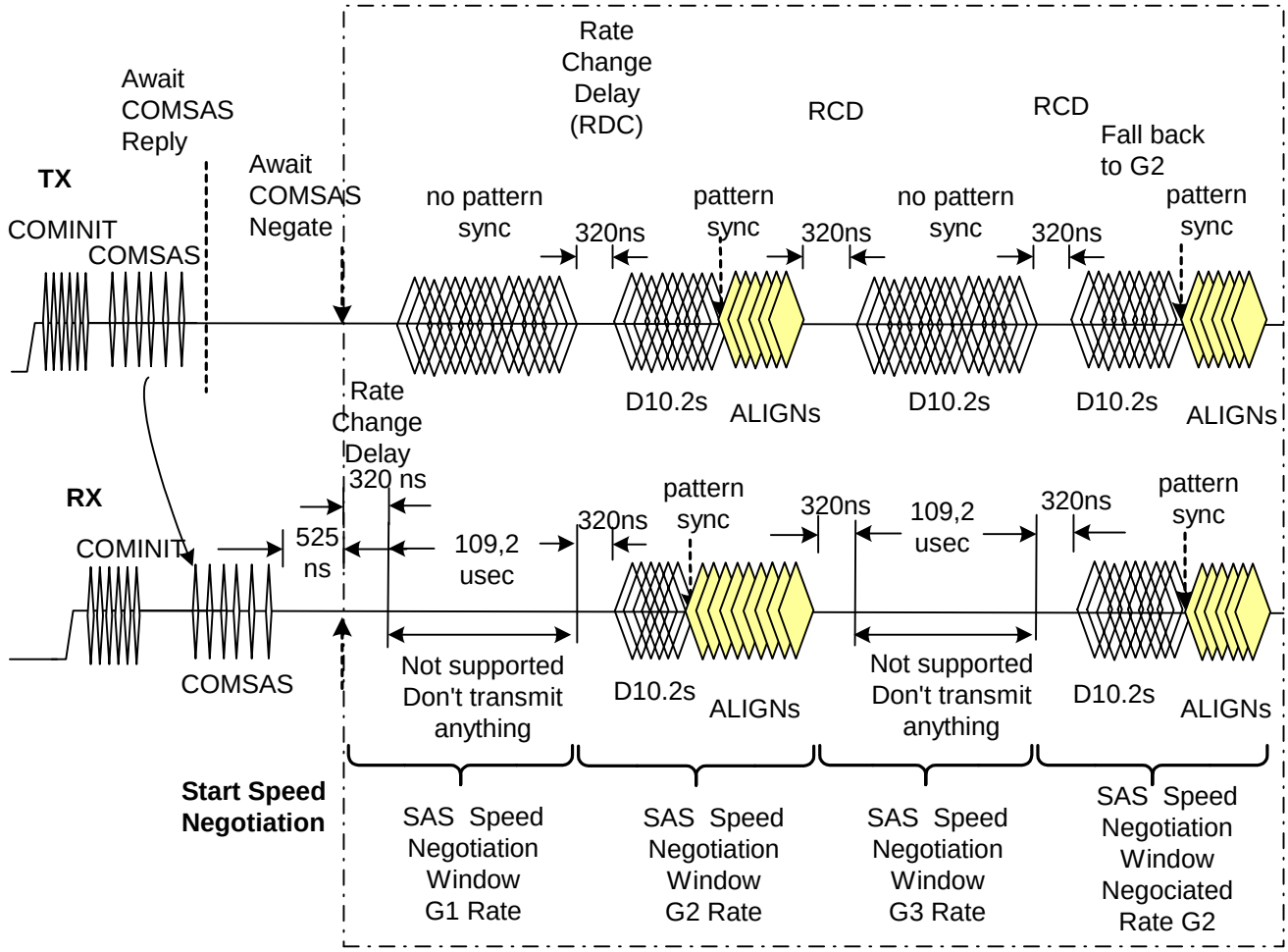

Figure 37. SAS speed negotiation sequence (example 2)

Figure 38~~Figure 38~~ shows the same speed negotiation sequence as in Figure 36~~Figure 36~~, except for some reason, one of the SAS device's PLLs does not obtain pattern synchronization and detect ALIGNs during the final speed negotiation window. If this occurs, the handshake is not complete and the OOB sequence shall be retried starting with COMINIT, forcing the link to retry the whole reset sequence.

Any time a SAS phy fails speed negotiation, it shall wait the Hot Plug Time-out Delay before attempting a retry.

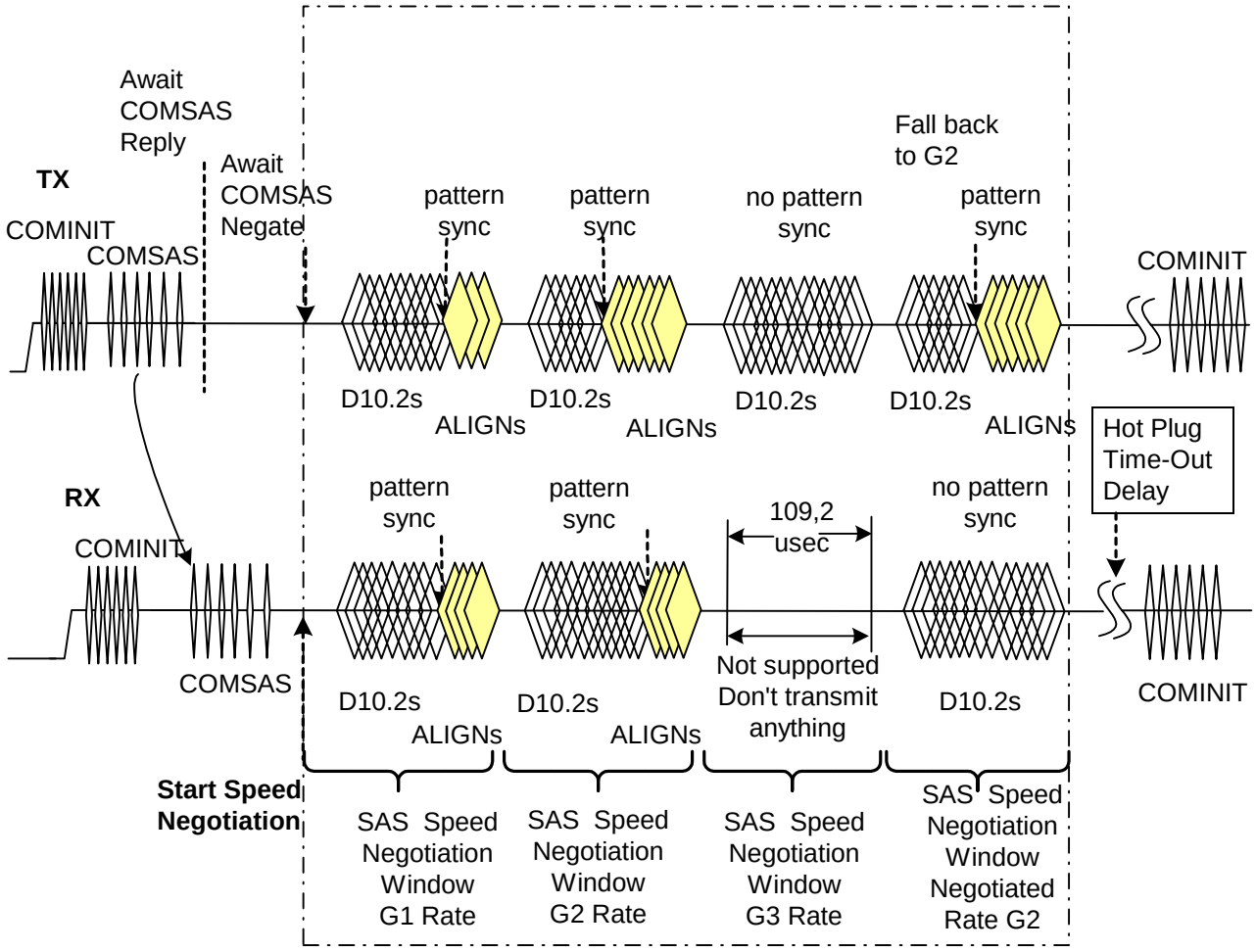

Figure 38. SAS speed negotiation sequence (example 3)

6.5 Phy reset sequence after signal cable insertion

Since SATA and SAS signal cable connectors do not include power lines, it is not possible to detect the physical insertion of the signal cable connector into a receptacle. In order to get around this limitation, SAS ports should periodically transmit a COMINIT sequence if they have not detected a COMINIT sequence within a Hot-Plug Timeout period.

Figure 39 illustrates how two SAS phys complete the phy reset sequence if the phys are not attached at power on. In this scenario, SAS phy 2 is connected to SAS phy 1 via a cable some time before SAS phy 2's second Hot-Plug timeout occurs. SAS phy 2's OOB detection circuitry detects a COMINIT after the connection is made, and therefore SAS phy 2 transmits the COMSAS sequence, since it has both sent and detected a COMINIT sequence. Upon detection of COMSAS, SAS phy 1 transmits its own COMSAS sequence bypassing the sent/received COMINIT requirement. The normal SAS speed negotiation process follows.

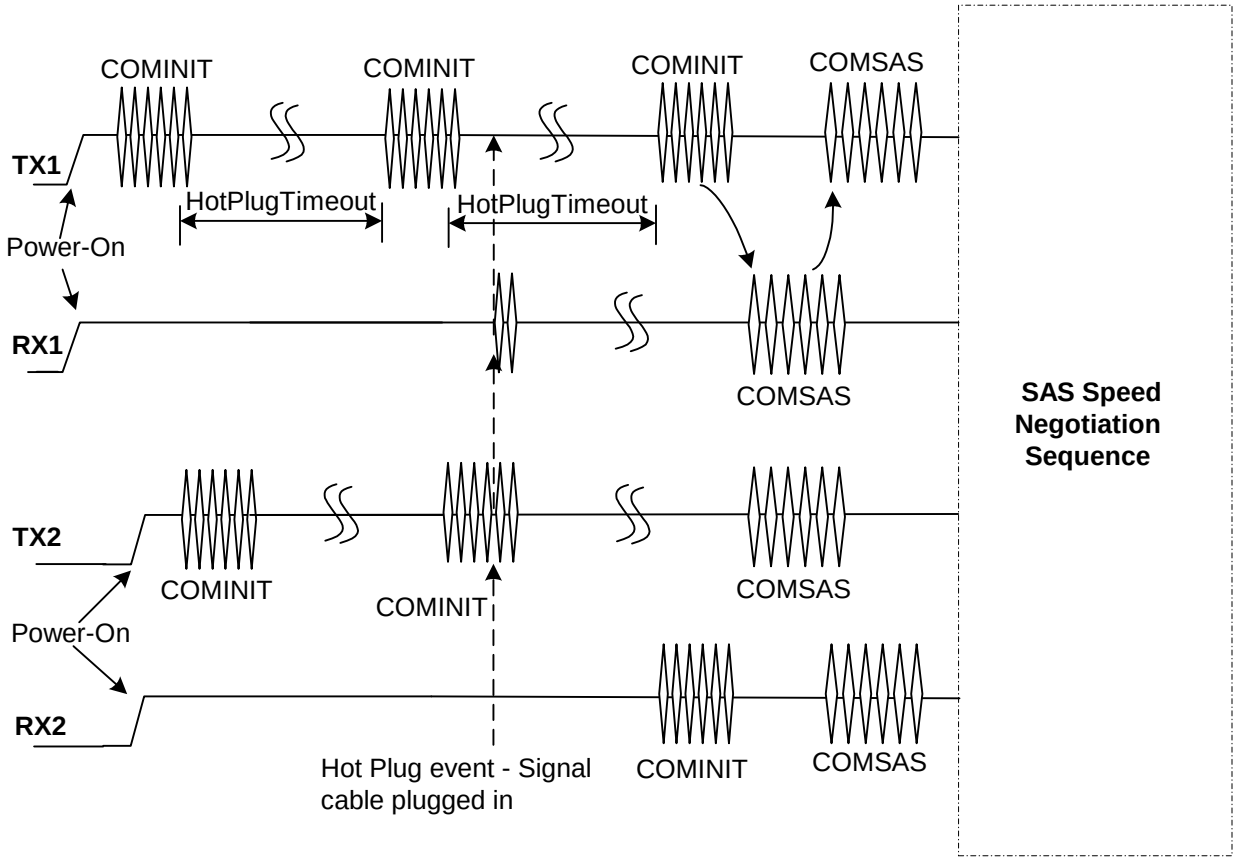

Figure 39. Hot-plug and the phy reset sequence

6.6 SAS phy state machine

6.6.1 SAS phy state machine overview

The SAS phy (SP) state machine can be broken down into three sections:

a) COMINIT/COMSAS handshake states;
b) SAS speed negotiation states; and
c) SATA host emulation states.

SAS phys shall maintain the following timers:

a) Hot Plug Time-out timer;
b) Speed Negotiation Window Time-Out (SNWTO) timer; and
c) COMSAS detect timeout timer.

The phy state machine functions to bring SAS links to a working state. There are multiple entry mechanisms to get the state machine to its initial SP0:SAS_Reset state.

Power on forces the SAS phy state machine into the SP0:SAS_Reset state..

The ULP may force a transition to the SP0:SAS_Reset state. In devices supporting SMP, the PHY CONTROL function may be used to trigger this transition.

The Dword Synchronization state machine (DWS), which runs as a link error policeman, sets policy on the operational state of the link by monitoring received errors on the link. Entry into the DWS:0-ACQ Sync Start state indicates that the link has lost D-word synchronization and triggers a counter. If D-word synchronization is not re-established in the timeout period, the phy initialization state machine shall change to the SP0:SAS_Reset state.

The Dword Synchronization state machine is active only after the phy reset sequence has completed.

6.6.2 COMINIT/COMSAS handshaking states

Figure 40~~Figure 40~~ shows the COMINIT/COMSAS handshake states, in which the SAS device transmits an initial COMINIT sequence, and awaits to receive a COMINIT sequence back. These states include a prefix of SAS_ in the state name. Once a COMINIT has been detected, the SAS device transmits a COMSAS sequence instead or in place of the normal SATA calibration pattern, and waits to see if a SAS device responds with a COMSAS. If the SAS device does not detect a COMSAS sequence within a COMSAS Detect Timeout period (3,5 µs), it assumes that a SATA Device is attached, and jumps to the SATA Host emulation states. Otherwise, a SAS-capable port is attached and the SAS speed negotiation states are entered.

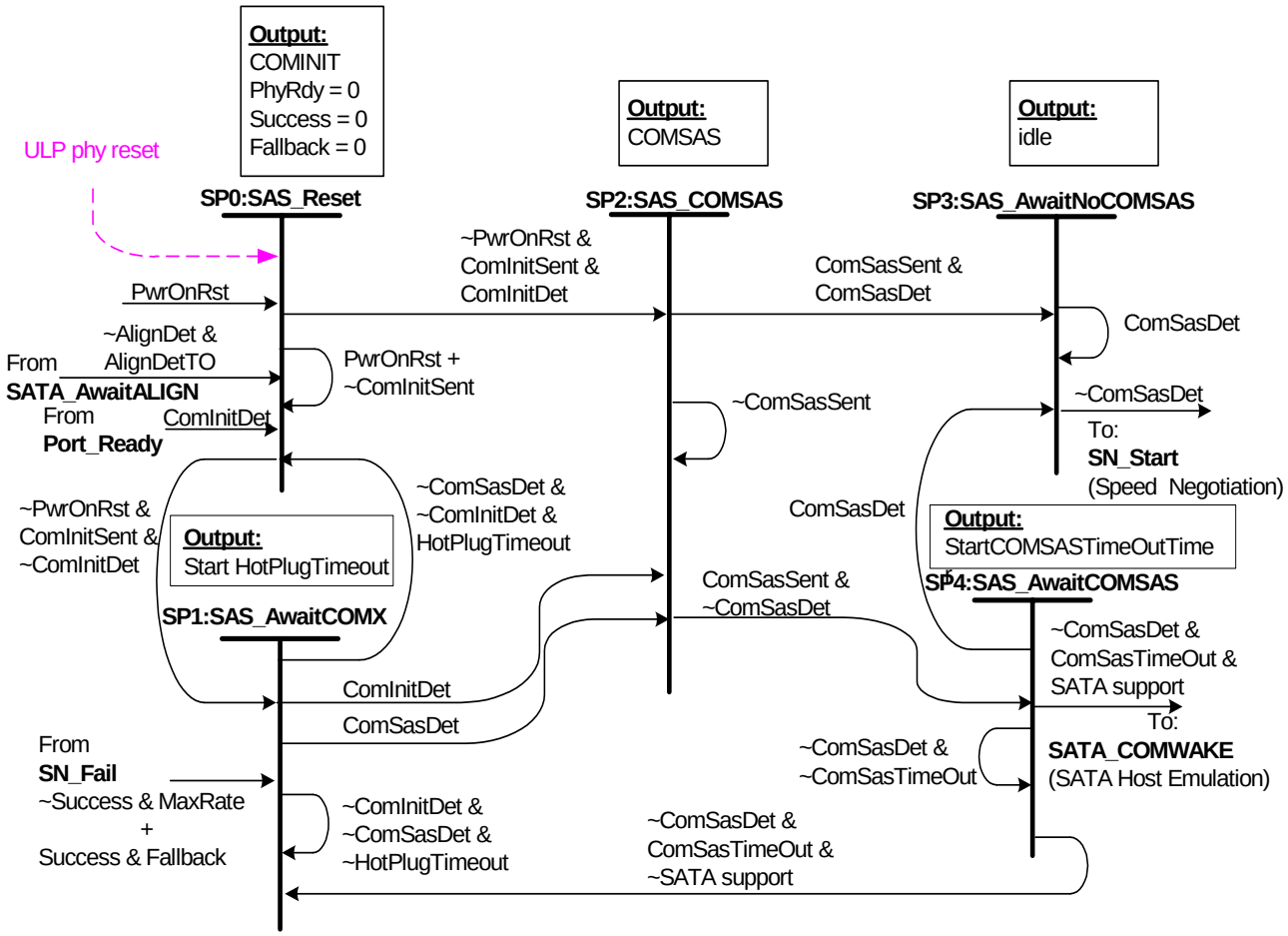

Figure 40. SAS phy state machine - COMINIT/COMSAS handshaking states

6.6.2.1 SP0:SAS_Reset state

6.6.2.1.1 State description

This state is entered when power-on reset (PwrOnRst) is asserted or general reset is requested from the ULP. It is also entered if COMINIT is detected in states SP13:PHY_Ready or SP22:SATA_PHY_Ready or if ALIGN primitives are not detected within an ALIGN Detect Timeout period after entering state SP18:SATA_AwaitALIGN.

While in this state, the COMINIT sequence is transmitted in multiples of six, as long as PwrOnRst is asserted. The state machine exits this state when it has been determined that a COMINIT burst has been sent, and PwrOnRst is de-asserted. PhyRdy is cleared to 0 upon entering this state. The Fallback and Success flags are also cleared upon entering this state.

6.6.2.1.2 Transition to SP2:SAS_COMSAS

Occurs when PwrOnRst is de-asserted, a COMINIT burst has been sent, a COMINIT was detected.

6.6.2.1.3 Transition to SP1:SAS_AwaitCOMX

Occurs when PwrOnRst is de-asserted, a COMINIT burst has been sent, a COMINIT was not detected.

6.6.2.2 SP1:SAS_AwaitCOMX state

6.6.2.2.1 State description

This state is entered when a COMINIT sequence has been sent during state SP0:SAS_RESET and a COMINIT sequence was not detected. Upon entering this state, a Hot-Plug Timeout counter shall be initialized and enabled to count down. The state machine exits this state when it detects a COMINIT, and a COMSAS sequence, or if a Hot-Plug timeout occurs before any of the OOB signals are detected.

This state is also entered from state SP10:SN_Fail if the there was not a successful lock and the max frequency has been attempted or there was a successful lock but the fallback didn't work. The state machine shall wait the Hot Plug time out delay before attempting the connection again.

6.6.2.2.2 Transition to SP0:SAS_Reset

Occurs when a Hot-Plug timeout occurs before any OOB signals are detected.

6.6.2.2.3 Transition to SP2:SAS_COMSAS

Occurs when a COMINIT or COMSAS sequence is detected. This indicates that a connection has been detected, so the SAS device transmits a COMSAS sequence to inform the other end of the link that it supports SAS protocol.

6.6.2.3 SP2:SAS_COMSAS state

6.6.2.3.1 State description

This state is entered when a COMINIT has been sent during state SP0:SAS_Reset, an incoming COMINIT was detected. The state machine remains in this state until the COMSAS sequence is transmitted.

6.6.2.3.2 Transition to SP3:SAS_AwaitNoCOMSAS

Occurs if a COMSAS has been sent and received. This acknowledges that two SAS ports are connected.

6.6.2.3.3 Transition to SP4:SAS_AwaitCOMSAS

Occurs if a COMSAS sequence is not detected while COMSAS is being transmitted. It cannot be determined yet if a SAS or SATA port is connected.

6.6.2.4 SP3:SAS_AwaitNoCOMSAS state

6.6.2.4.1 State description

This state is entered when a COMSAS sequence is detected in state SP2:SAS_COMSAS or state SP4:SAS_AwaitCOMSAS.

The state machine remains in this state until the end of the COMSAS sequence is detected, at which point the SAS speed negotiation begins.

6.6.2.4.2 Transition to SP5:SN_Start

Occurs when the end of the COMSAS sequence is detected.

6.6.2.5 SP4:SAS_AwaitCOMSAS state

6.6.2.5.1 State description

This state is entered when a COMSAS sequence has been transmitted, but a COMSAS has not yet been detected in state SP2:SAS_COMSAS.

Upon entering this state the COMSAS Timeout counter shall be initialized and enabled to count down. The state machine remains in this state until either a COMSAS sequence is detected or a COMSAS detect timeout period (3,5 µs) occurs.

6.6.2.5.2 Transition to SP3:SAS_AwaitNoCOMSAS

Occurs when a COMSAS sequence is detected.

6.6.2.5.3 Transition to SP3:SAS_AwaitNoCOMSAS

All SAS devices that support SATA target attach shall transition to this state when a COMSAS sequence is not detected within the COMSAS detect timeout period (3,5 µs).

6.6.2.5.4 Transition to SAS_AwaitNoCOMX

SAS devices that don't support SATA target attach shall transition to this state when a COMSAS sequence is not detected within the COMSAS detect timeout period (3,5 µs).

6.6.3 SAS speed negotiation states

[Editor's note: in all state diagrams, make sure that multiple arrows don't meet at the same location]

Figure 41~~Figure 41~~ shows the SAS Speed Negotiation states, in which the SAS phy has detected that it is attached to a SAS phy, and engages in the SAS speed negotiation procedure. These states are indicated by state names with a prefix of SN or PHY.

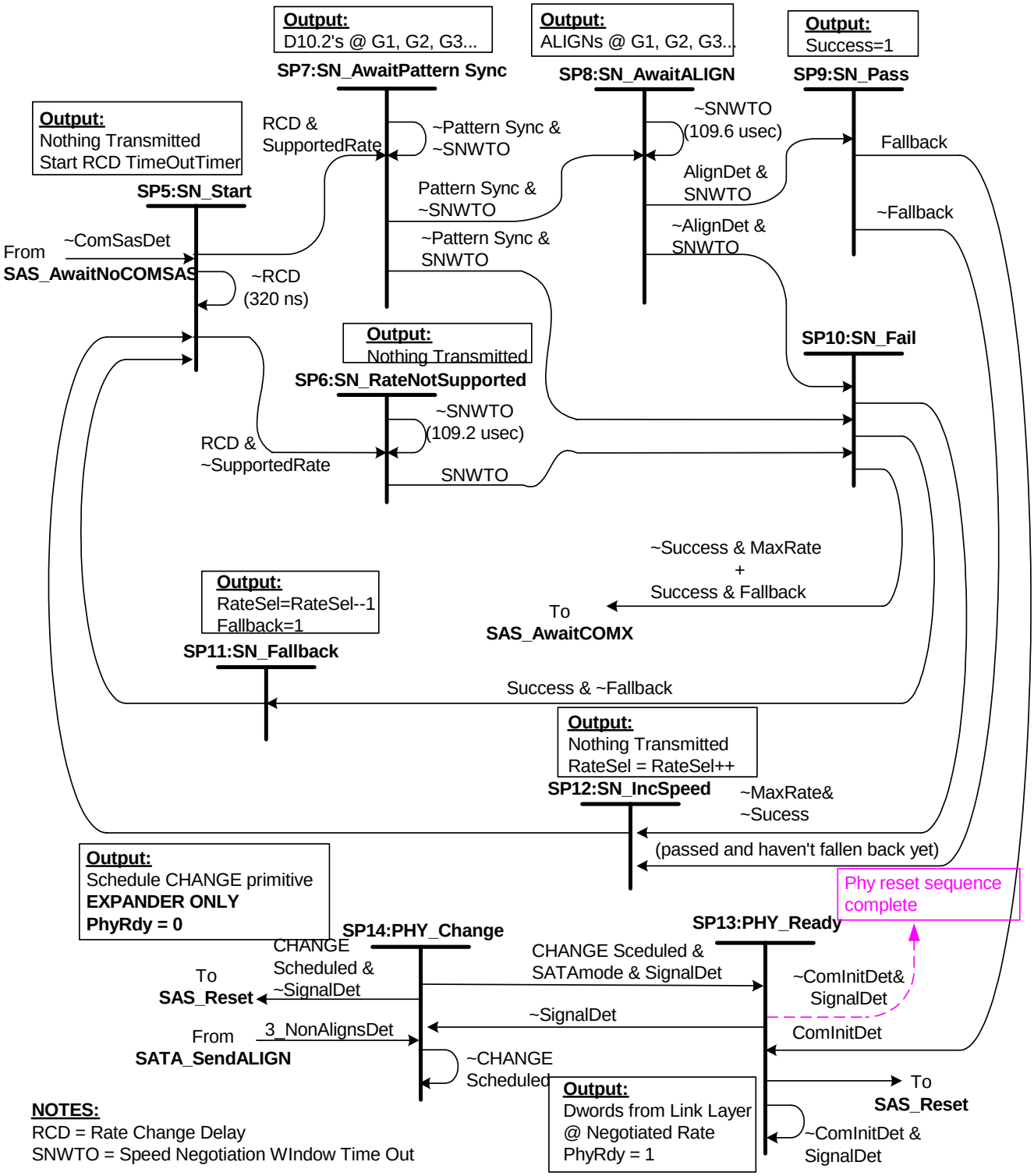

Figure 41. SAS phy state machine - SAS speed negotiation states

6.6.3.1 SP5:SN_Start state

6.6.3.1.1 State description

This state marks the beginning a SAS speed negotiation process. It is used to get the transmit lines to an idle state in between SAS speed negotiation windows. A SAS speed negotiation window is defined to be 109,2 µs

(4096 Gen1 UIs). This state is entered from state SP3:SAS_AwaitNoCOMSAS after the end of a COMSAS sequence. It can also be entered from states SN_Inc Speed or SP11:SN_Fallback for another pass, after a speed negotiation window occurs. Upon entering this state, the Rate Change Delay timer is loaded with an initial value and enabled to count down. This allows time required for a transmitter to switch to either the next higher or next lower supported speed. The state machine remains in this state until a Rate Change Delay (320ns) has elapsed. The transmitter is turned off and nothing is transmitted during this state.

6.6.3.1.2 Transition to SP6:SN_RateNotSupported
Occurs when the Rate Change Delay timer expires and the current rate is not supported.

6.6.3.1.3 Transition to SP7:SN_AwaitPatternSync
Occurs when the Rate Change Delay timer expires and the current rate is supported.

6.6.3.2 SP6:SN_RateNotSupported state

6.6.3.2.1 State description
This state is entered after a Rate Change Delay (RCD) from state SP5:SN_Start when the currently selected rate is not supported. During this state nothing is transmitted for an entire speed negotiation window. Upon entering this state the SNWTO counter is loaded and enabled to count down. The state machine exits from this state after the SNWTO counter expires.

6.6.3.2.2 Transition to SP10:SN_Fail
Occurs after the SNWTO counter expires.

6.6.3.3 SP7:SN_AwaitPatternSync state

6.6.3.3.1 State description
The state machine starts transmitting D10.2 characters at the current supported rate (Gen1, Gen2, Gen3…) in this state. Upon entering this state, the Speed Negotiation Window Timeout (SNWTO) counter is loaded and enabled to count down. The state machine exits this state when a SNWTO occurs without the pattern synchronization or immediately when the pattern synchronization occurs.

6.6.3.3.2 Transition to SP8:SN_AwaitALIGN
Occurs when pattern synchronization is obtained before an SNWTO period.

6.6.3.3.3 Transition to SP10:SN_FAIL
Occurs when the pattern synchronization has not obtained within an SNWTO.

6.6.3.4 SP8:SN_AwaitALIGN state

6.6.3.4.1 State description
This state is entered from state SP7:SN_AwaitPatternSync if the PLL has locked and obtained pattern synchronization within an SNWTO. Pattern sync means detecting D10.2s or the 01010101 or 101010…pattern. During this state the ALIGN primitive is transmitted. This state is exited when the SNWTO expires. During this window the state machine latches the fact that Aligns were received. If so the hand shake is complete and this speed window is successful.

6.6.3.4.2 Transition to SP9:SN_Pass
Occurs if ALIGNs are detected within an SNWTO period. must be reduced and a final speed negotiation window must be attempted.

6.6.3.4.3 Transition to SP10:SN_Fail
Occurs if no ALIGNs were detected within an SNWTO period. This indicates that the other device must not have been able to lock at the current rate.

6.6.3.5 SP9:SN_Pass state

6.6.3.6 State description
This state is entered from state SP8:SN_AwaitALIGN after an SNWTO occurs. This is a single clock state that set the Success flag, indicating that there was at least one successful speed negotiation.

6.6.3.6.1 Transition to SP13:PHY_Ready

The transition to the ready state is taken if the speed negotiation has completed. This is indicated by the Fallback flag. If this flag is set it indicates that the speed negotiation has progressed to where it failed and then had fallen back to the last negotiated speed.

6.6.3.6.2 Transition to SP12:SN_IncSpeed

If the state machine has not fallen back, this means that it has not progressed past the breaking point so the speed is increased for the next attempt.

6.6.3.7 SP10:SN_Fail state

6.6.3.7.1 State description

This state is entered from state SP7:SN_AwaitPatternSync or SP8:SN_AwaitALIGN if the other end does not signal that it supports the rate or has locked at this rate. It is also entered when the rate is not supported and locking is not possible.

6.6.3.7.2 Transition to SP0:SAS_Reset

Occurs when the state machine fails to lock and the max rate has been attempted or when the speed negotiation fails after dropping back to the last working frequency.

6.6.3.7.3 Transition to SP11:SN_Fallback

Transition to fallback if the speed negotiation has failed and there was a previous success.

6.6.3.8 SP11:SN_Fallback state

6.6.3.8.1 State description

This state is entered from state SP10:SN_Fail after an SNWTO occurs indicating that the current tested speed failed but there was a former success. This is a one clock state that sets the Fallback flag to indicate that the sequence has progressed through a successful negotiation to one that fails and now needs to fall back to the highest mutually supported speed.

6.6.3.8.2 Transition to SP5:SN_Start

Occurs on the next clock.

6.6.3.9 SP12:SN_IncSpeed state

This is a one clock state that increments the speed for the next attempt. This state is entered when the transmitter has not reached its maximum supported rate during the SAS speed negotiation process. This can either be when the previous window has failed or passed but the highest supported rate hasn't been tested.

6.6.3.9.1 Transition to SP5:SN_Start

Occurs on the next clock with the transmitter set for the next highest rate.

6.6.3.10 SP13:PHY_Ready state

6.6.3.10.1 State description

This state is entered from state SP9:SN_Pass if the Fallback flag is set, which is an indication that the SAS speed negotiation procedure has completed.

The state machine remains in this state until a COMINIT sequence is detected or if SignalDet is de-asserted. While in this state, the PhyRdy signal to the link layer shall be set to 1 and dwords from the link layer are transmitted at the negotiated rate.

Note that the OOB state machine stays in this state but passes control to next layer until the state of the Phy changes.

6.6.3.10.2 Transition to SP14:PHY_Change

Occurs if SignalDet is de-asserted. This is an indication that a device has been unplugged from the port, and requires the expander to send CHANGE primitives to connected initiators.

6.6.3.10.3 Transition to SP0:SAS_Reset

Occurs if COMINIT is asserted by the connected device**.**

6.6.3.11 SP14:PHY_Change state

6.6.3.11.1 State description

This state is entered from state SP13:PHY_Ready if SignalDet is de-asserted. SignalDet is a real time indication that valid signals are being detected by the receiver. When de-asserted, this signal indicates that an unplug event occurred, and should trigger an Expander to send a CHANGE primitive to all of its initiator ports, notifying them of a failed or removed device. This state can also be entered from state SP20:SATA_SendALIGN after three back-to-back non-ALIGN primitives are detected during the SATA speed negotiation sequence. The completion of a SATA speed negotiation requires a CHANGE primitive since SATA targets do not transmit the IDENTIFY address frame after link initialization.

6.6.3.11.2 Transition to SP13:PHY_Ready

Occurs after the CHANGE primitive is scheduled, but only if SataMode=1 and SignalDet=1.

6.6.3.11.3 Transition to SP0:SAS_Reset

Occurs when the CHANGE primitive is scheduled, but SignalDet = 0.

6.6.4 SATA host emulation states

Figure 42~~Figure 42~~ shows the SATA Host emulation states, in which the SAS device has detected that it is connected a SATA target and behaves as if it were a SATA Host as far as OOB is concerned, initiating the speed negotiation. These states are indicated by state names with a prefix of SATA_. During SATA Host emulation, the SAS device transmits a COMWAKE sequence and then waits to receive a COMWAKE. Once the COMWAKE sequence is detected, the SAS device follows the speed negotiation sequence defined in the SATA 1.0 specification.

SAS PHY for SAS Hosts, Targets and Expanders should use similar state machines to the one defined in this specification, with the exception that SAS Initiators would implement the SATA Host emulation Power Management states (shown as shaded).

The power management states that are defined in this specification are for SAS initiators that intend to attach directly to SATA targets. For SAS-only initiators, these states are not needed.

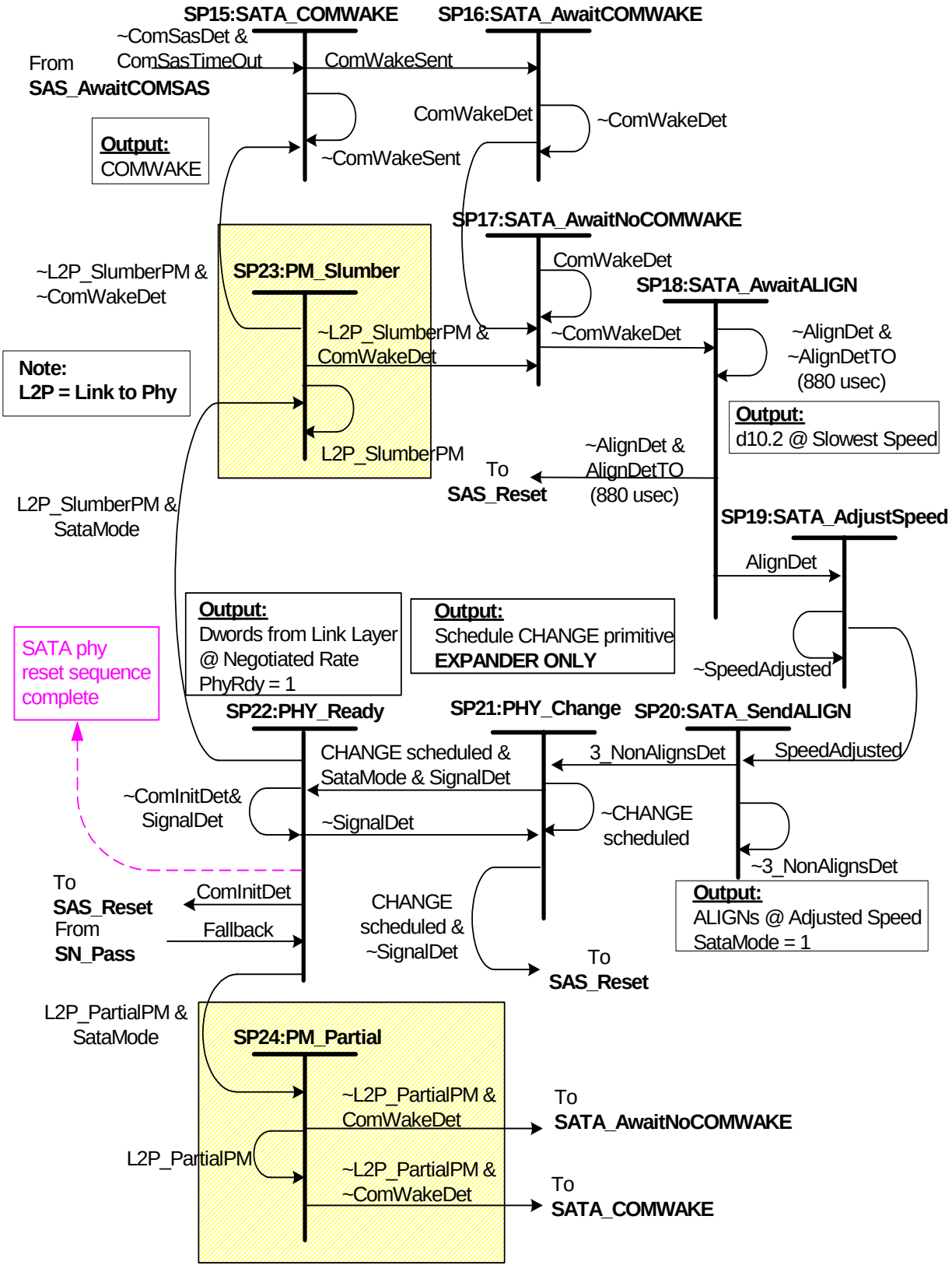

Figure 42. SAS phy state machine - SATA host emulation states

6.6.4.1 SP15:SATA_COMWAKE state

6.6.4.1.1 State description

This is the first state of the SATA Host Emulation states. This state is entered from state SP4:SAS_AwaitCOMSAS when a COMSAS is not detected within a COMSAS detect timeout period of 3,5 µs. The state machine remains in this state until a COMWAKE sequence has been transmitted.

6.6.4.1.2 Transition to SP16:SATA_AwaitCOMWAKE

Occurs when the COMWAKE sequence has been transmitted.

6.6.4.2 SP16:SATA_AwaitCOMWAKE state

6.6.4.2.1 State description

This state is entered when a COMWAKE sequence has been sent during state SP15:SATA_COMWAKE. The state machine remains in this state until a COMWAKE sequence has been detected.

6.6.4.2.2 Transition to SATA_AawitNoCOMWAKE

Occurs when a COMWAKE sequence is detected.

6.6.4.3 SP17:SATA_AwaitNoCOMWAKE state

6.6.4.3.1 State description

This state is entered when a COMWAKE sequence is detected in state SATA_AwaitCOMX. The state machine remains in this state until the end of the COMWAKE sequence is detected.

6.6.4.3.2 Transition to SP18:SATA_AwaitALIGN

Occurs when the end of the COMWAKE sequence has been detected.

6.6.4.4 SP18:SATA_AwaitALIGN state

6.6.4.4.1 State description

This state is entered from state SP17:SATA_AwaitNoCOMWAKE after the end of a COMWAKE sequence has been detected. While in this state, the SAS device is transmitting D10.2 characters at its slowest supported transmit speed. The state machine remains in this state until either an ALIGN primitive is detected at any of its supported rates, or 880 µs (32K Gen1 dwords) have elapsed without detecting any ALIGN primitives.

The SAS device shall start transmitting D10.2 characters no later than 20 Gen 1 dwords (533ns) after COMWAKE was de-asserted.

6.6.4.4.2 Transition to SP19:SATA_AdjustSpeed

Occurs when the SAS device detects an ALIGN primitive at any of its supported rates.

6.6.4.4.3 Transitition to SP0:SAS_Reset

Occurs when the SAS device does not detect any ALIGN primitives within 880 µs (32K Gen1 dwords)

6.6.4.5 SP19:SATA_AdjustSpeed state

6.6.4.5.1 State description

This state is entered from state SP18:SATA_AwaitALIGN when the SAS device detects an ALIGN primitive at any of its supported rates. During this state, the SAS device's PHY transmitter circuitry adjusts to the same rate of the ALIGN primitives that were detected by the receiver circuitry. The state machine remains in this state until the transmitter rate has been adjusted appropriately.

6.6.4.5.2 Transition to SP20:SATA_SendALIGN

Occurs when the transmitter rate has been adjusted to the rate of the received ALIGN primitives.

6.6.4.6 SP20:SATA_SendALIGN state

6.6.4.6.1 State description

This state is entered from state SP19:SATA_AdjustSpeed after the transmitter rate has been adjusted to the rate of the received ALIGN primitives. While in this state the SAS device transmits ALIGN primitives at the adjusted rate until the receiver circuitry detects three back-to-back non-ALIGN primitives, at which point the state machine transitions to state SP22:SATA_PHY_Ready, where PhyRdy is set to 1 to indicate that the link has been brought up successfully.

6.6.4.6.2 Transition to SP21:SATA_PHY_Change

Occurs when the SAS device's receiver circuitry detects three back-to-back non-ALIGN primitives.

6.6.4.7 SP21:SATA_PHY_Change state

6.6.4.7.1 State description

This state is entered from state SP20:SATA_SendALIGN. It In the case of a SATA target, it is the expanders responsibility to request the CHANGE primitive be sent.

6.6.4.7.2 Transition to SP22:SATA_PHY_Ready

Occurs after the CHANGE primitive is scheduled.

6.6.4.8 SP22:SATA_PHY_Ready state

6.6.4.8.1 State description

This state is entered from state SP21:SATA_PHY_Change state after the CHANGE primitive has been scheduled. It can also be entered from state SP20:SATA_SendALIGN after three back-to-back non-ALIGN primitives are detected during the SATA speed negotiation sequence. The state machine remains in this state until a COMINIT sequence is detected. While in this state, the PhyRdy signal to the link layer shall be set to 1 and dwords from the link layer are transmitted at the negotiated rate.

6.6.4.8.2 Transition to SP0:SAS_Reset

Occurs when a COMINIT sequence is detected.

6.6.4.9 SP23:PM_Slumber state

6.6.4.9.1 State description

This state is entered from SP22:SATA_PHY_Ready if the device is operating in SATA mode and L2P_SlumberPM is received from the link layer. Exit from this state is driven from receipt of COMWAKE or de-assertion of the slumber signal.

The phy shall remember if COMWAKE was detected during SP23:PM_Slumber to determine if the wakeup request was originated from the host or the phy.

6.6.4.9.2 Transition to SP17:SATA_AwaitNoCOMWAKE

Occurs when the receiver circuitry detects COMWAKE from the device at the other end of the link. This transition should not occur until the phy interface is recovered from Slumber mode and is ready to initiate communications.

6.6.4.9.3 Transition to SP15:SATA_COMWAKE

The device shall transition to SP15:SATA_COMWAKE if the link layer de-asserts the slumber signal to the PHY interface.

6.6.4.10 SP24:PM_Partial state

6.6.4.10.1 State description

This state is entered from SP22:SATA_PHY_Ready if L2P_PartialPM is received from the link layer. Exit from this state is driven from receipt of COMWAKE or de-assertion of the partial signal.

The phy shall remember if COMWAKE was detected during SP24:PM_Partial to determine if the wakeup request was originated from the host or the phy.

6.6.4.10.2 Transition to SP17:SATA_AwaitNoCOMWAKE

Occurs when the receiver circuitry detects COMWAKE.

6.6.4.10.3 Transition to SP15:SATA_COMWAKE

The device shall transition to SP15:SATA_COMWAKE if the link layer de-asserts the slumber signal to the PHY interface.

6.7 Dword synchronization state machine

6.7.1 Dword synchronization state machine overview

A dword synchronization (DWS) state machine is an entity the resides in each SAS phy. The DWS state machine establishes the same dword boundaries at the receiver as at the attached transmitter (see figure 1). The DWS state machine requires an error free string of dwords to obtain synchronization. The DWS state machine tolerates invalid dwords as defined in the state machine before losing dword synchronization. While not in dword synchronization, information received is invalid and not passed to the SAS phy end point connection state machine. The DWS state machine contains the following states:

a) DWS0: ACQ Sync Start;
b) DWS1: 1 valid;
c) DWS1: 2 valid;
d) DWS3: Sync acquired;
e) DWS4: Lose Sync 1 invalid;
f) DWS4a: Lose Sync 1 - 1 valid;
g) DWS5: Lose Sync 2 invalid;
h) DWS4a: Lose Sync 2 - 1 valid;
i) DWS6: Lose Sync 3 invalid; and
j) DWS6a: Lose Sync 3 - 1 valid.

Figure 43 shows the dword synchronization state machine.

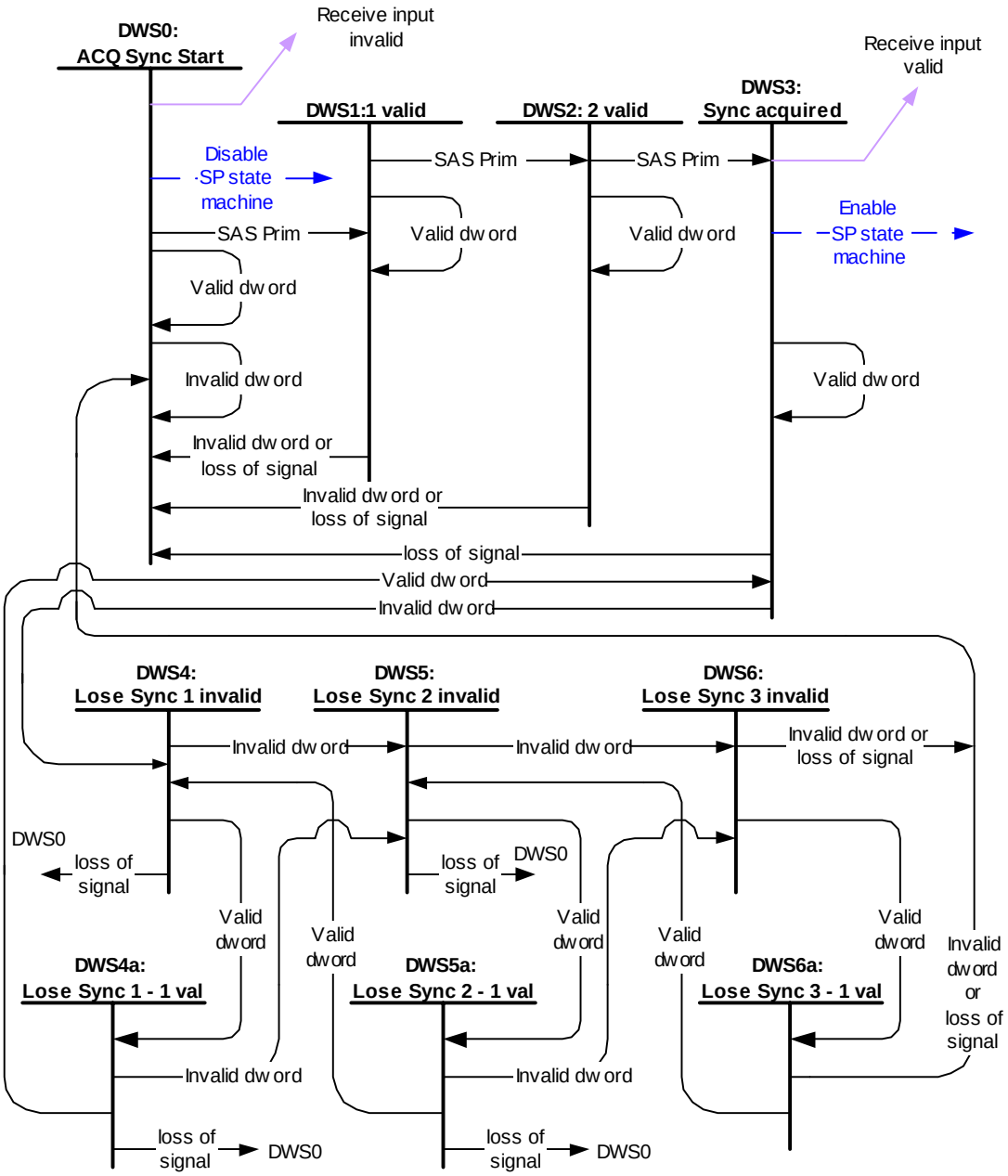

Figure 43. Dword synchronization state machine

6.7.2 DWS0: ACQ Sync Start state

6.7.2.1 DWS0: ACQ Sync Start state description

After OOB or loss of signal when dword synchronization was previously valid, the DWS ACQ SYC Start (acquire synchronization start) state is entered. On entry into this state, SP state machine is disabled. The receiver monitors information received and forces the first K28.5 character into the first byte position for word alignment. The receiver continues to reestablish word alignment by forcing received K28.5 characters into the first byte position until a valid SAS primitive is detected using the currently dword alignment.

6.7.2.2 Transition DWS0:DWS1 (ACQ SYNC Start:1 valid)

The DWS0:DWS1 transition shall occur when a valid SAS primitive is detected. This indicates the present dword alignment is valid for this word.

6.7.2.3 Transition DWS0a:DWS0a (ACQ SYNC Start: ACQ SYNC Start)

The DWS0a:DWS0a transition shall occur when a valid dword is detected that is not a SAS primitive.

6.7.2.4 Transition DWS0b:DWS0b (ACQ SYNC Start: ACQ SYNC Start)

The DWS0b:DWS0b transition shall occur when an invalid dword or a loss of signal is detected. The receiver continues to monitor information received and forces the first K28.5 character into the first byte position for word alignment.

6.7.3 DWS1: 1 valid state

6.7.3.1 DWS1: 1 valid state description

The receiver monitors information received until a valid SAS primitive, invalid dword or a loss of signal is detected.

6.7.3.2 Transition DWS1:DWS2 (1 valid:2 valid)

The DWS1:DWS2 transition shall occur when a valid SAS primitive is detected. This indicates the present dword alignment is valid for this word.

6.7.3.3 Transition DWS1:DWS1 (1 valid: 1valid)

The DWS1:DWS1 transition shall occur when a valid dword is detected that is not a SAS primitive.

6.7.3.4 Transition DWS1:DWS0 (1 valid: ACQ SYNC Start)

The DWS1:DWS0 transition shall occur when an invalid dword or a loss of signal is detected. A new attempt at word alignment is started.

6.7.4 DWS2: 2 valid state

6.7.4.1 DWS2: 2 valid state description

The receiver has detected 2 valid SAS primitives using the current word alignment. The receiver monitors information received until a valid SAS primitive, invalid dword or a loss of signal is detected.

6.7.4.2 Transition DWS2:DWS3 (2 valid:3 valid)

The DWS1:DWS2 transition shall occur when a valid SAS primitive is detected. This indicates the present dword alignment is valid for this word.

6.7.4.3 Transition DWS2:DWS2 (2 valid:2 valid)

The DWS2:DWS2 transition shall occur when a valid dword is detected that is not a SAS primitive.

6.7.4.4 Transition DWS2:DWS0 (1 valid: ACQ SYNC Start)

The DWS1:DWS0 transition shall occur when an invalid dword or a loss of signal is detected. A new attempt at word alignment is started.

6.7.5 DWS3: Sync acquired state

6.7.5.1 DWS3: Sync acquired state description

The receiver has detected 3 valid SAS primitives using the current word alignment. The SAS phy end point connection state machine is enabled. The current SAS primitive is valid for processing. The receiver monitors information received until an invalid dword is received or a loss of signal occurs.

6.7.5.2 Transition DWS3:DWS4 (Sync acquired:Lose Sync 1 invalid)

The DWS3:DWS4 transition shall occur when an invalid dword is detected. An expander forwarding the dword to another link shall replace the invalid dword with ERROR or SATA_ERROR (see 7.11.8.3).

6.7.5.3 Transition DWS3:DWS3 (Sync acquired: Sync acquired)

The DWS3:DWS3 transition shall occur when a valid dword is detected.

6.7.5.4 Transition DWS3:DWS0 (Sync acquired: ACQ SYNC Start)

The DWS3:DWS0 transition shall occur when a loss of signal is detected. A new attempt at word alignment is started.

6.7.6 DWS4: Lose Sync 1 invalid state

6.7.6.1 DWS4: Lose Sync 1 invalid state description

The receiver has detected an invalid dword. This is the first state in the process that may result in losing dword synchronization. The receiver monitors information received.

6.7.6.2 Transition DWS4:DWS4a (Lose Sync 1 invalid:Lose Sync 1 - 1 val)

The DWS4:DWS4a transition shall occur when a valid dword is detected. This indicates the present dword alignment may still be valid for this word.

6.7.6.3 Transition DWS4:DWS5 (Lose Sync 1 invalid:Lose Sync 2 invalid)

The DWS4:DWS5 transition shall occur when an invalid dword is detected. An expander forwarding the dword to another link shall replace the invalid dword with ERROR or SATA_ERROR (see 7.11.8.3).

6.7.6.4 Transition DWS4:DWS0 (Lose Sync 1 invalid: ACQ SYNC Start)

The DWS4:DWS0 transition shall occur when a loss of signal is detected. A new attempt at word alignment is started.

6.7.7 DWS4a: Lose Sync 1 - 1 val state

6.7.7.1 DWS4a: Lose Sync 1 - 1 val state description

The receiver has detected a valid dword after an invalid dword. Dword synchronization may still be valid. The receiver monitors information received for valid dwords, invalid dwords and loss of signal.

6.7.7.2 Transition DWS4a:DWS3 (Lose Sync 1 - 1 val:Sync acquired)

The DWSa4:DWS3 transition shall occur when a valid dword is detected. Two valid dwords have been received since the last invalid dword.

6.7.7.3 Transition DWS4a:DWS5 (Lose Sync 1 - 1 val:Lose Sync 2 invalid)

The DWS4a:DWS5 transition shall occur when an invalid dword is detected. An expander forwarding the dword to another link shall replace the invalid dword with ERROR or SATA_ERROR (see 7.11.8.3).

6.7.7.4 Transition DWS4a:DWS0 (Lose Sync 1 - 1 val: ACQ SYNC Start)

The DWS4a:DWS0 transition shall occur when a loss of signal is detected. A new attempt at word alignment is started.

6.7.8 DWS5: Lose Sync 2 invalid state

6.7.8.1 DWS5: Lose Sync 2 invalid state description

The receiver has detected an invalid dword without sufficient valid dwords to nullify a previous invalid dwords. The receiver monitors information received for valid dwords, invalid dwords or loss of signal.

6.7.8.2 Transition DWS5:DWS5a (Lose Sync 2 invalid:Lose Sync 2 - 1 val)

The DWS5:DWS5a transition shall occur when a valid dword is detected. This indicates the present dword alignment is still valid for this word.

6.7.8.3 Transition DWS5:DWS6 (Lose Sync 2 invalid:Lose Sync 2 invalid)

The DWS5:DWS6 transition shall occur when an invalid dword is detected. An expander forwarding the dword to another link shall replace the invalid dword with ERROR or SATA_ERROR (see 7.11.8.3).

6.7.8.4 Transition DWS5:DWS0 (Lose Sync 2 invalid: ACQ SYNC Start)

The DWS5:DWS0 transition shall occur when a loss of signal is detected. A new attempt at word alignment is started.

6.7.9 DWS5a: Lose Sync 2 - 1 val state

6.7.9.1 DWS5a: Lose Sync 2 - 1 val state description

The receiver has detected a valid dword after an invalid dword. Dword synchronization may still be valid. The receiver monitors information received for valid dwords, invalid dwords and a loss of signal is detected.

6.7.9.2 Transition DWS5a:DWS4 (Lose Sync 2 - 1 val:Lose Sync 1 invalid)

The DWS5a:DWS4 transition shall occur when a valid dword is detected. Two valid dwords have been received since the last invalid dword.

6.7.9.3 Transition DWS5a:DWS6 (Lose Sync 2 - 1 val:Lose Sync 3 invalid)

The DWS5a:DWS6 transition shall occur when an invalid dword is detected. An expander forwarding the dword to another link shall replace the invalid dword with ERROR or SATA_ERROR (see 7.11.8.3).

6.7.9.4 Transition DWS5a:DWS0 ((Lose Sync 2 - 1 val: ACQ SYNC Start)

The DWS5a:DWS0 transition shall occur when a loss of signal is detected. A new attempt at word alignment is started.

6.7.10 DWS6: Lose Sync 3 invalid state

6.7.10.1 DWS6: Lose Sync 3 invalid state description

The receiver has detected an invalid dword without sufficient valid dwords to nullify a previous invalid dwords. The receiver monitors information received for valid dwords, invalid dwords and a loss of signal.

6.7.10.2 Transition DWS6:DWS6a (Lose Sync 3 invalid:Lose Sync 3 - 1 val)

The DWS6:DWS6a transition shall occur when a valid dword is detected. This indicates the present dword alignment may still valid for this word.

6.7.10.3 Transition DWS6:DWS0 (Lose Sync 3 invalid:ACQ Sync Start)

The DWS6:DWS0 transition shall occur when an invalid dword or loss of signal is detected. If an invalid dword is detected, at least 4 invalid dwords have been received without sufficient intervening valid dwords to nullify there affect. Dword synchronization is lost and must be reacquired.

6.7.11 DWS6a: Lose Sync 3 - 1 val state

6.7.11.1 DWS6a: Lose Sync 3 - 1 val state description

The receiver has detected a valid dword after an invalid dword. Dword synchronization may still be valid. The receiver monitors information received for valid dwords, invalid dwords and loss of signal.

6.7.11.2 Transition DWS6a:DWS5 (Lose Sync 3 - 1 val:Lose Sync 2 invalid)

The DWS6a:DWS5 transition shall occur when a valid dword is detected. Two valid dwords have been received since the last invalid dword.

6.7.11.3 Transition DWS6a:DWS0 (Lose Sync 3 - 1 val:ACQ Sync Start)

The DWS6a:DWS0 transition shall occur when an invalid dword or loss of signal is detected. If an invalid dword is detected, at least 4 invalid dwords have been received without sufficient intervening valid dwords to nullify there affect. Dword synchronization is lost and must be reacquired.

6.8 Spin-up

A SATA target device with rotating media spins up:

a) automatically after power on (allowed by SATA);
b) after its phy is enabled (allowed by SATA);
c) after the reset sequence has completed (recommended by SATA); or
d) after the Power Up in Standby flag is cleared by an application (if the ATA Power Up in Standby feature is implemented).

NOTE: The ATA Power Up in Standby feature is not widely implemented, since it requires the target device to include a nonvolatile memory to remember the state of the Power Up in Standby flag. Desktop-class disk drives do not typically have nonvolatile memory storage.

SAS target devices with rotating media shall not spin-up until the reset sequence has completed.

If it does not receive COMSAS during the reset sequence, a SAS target device supporting SATA should spin-up immediately after detecting COMWAKE unless it supports the ATA Power Up in Standby feature and that feature has been enabled. This ensures that the target device spins up when connected to a SATA initiator port, where the initiator port assumes the target device has spun up automatically.

If it does receive COMSAS during the reset sequence, a SAS target device shall not spin-up until a SCSI START STOP UNIT command is received with the START bit set to one by the appropriate logical init.

NOTE: Enclosures supporting both SATA-only target devices and SAS target devices may need to sequence power to each target device to avoid excessive power consumption during power on, since the SATA-only target devices are allowed to spin-up automatically.

7 Link layer

7.1 Primitives

7.1.1 Primitives overview

Primitives are dwords whose first byte is a K28.5 or K28.3 control character. Primitives are not considered big-endian or little-endian; they are just interpreted as first, second, third, and last bytes. Table 23 shows the primitive format.

Table 23. Primitive format

Byte	Description
First	K28.5 control character for SAS primitives; K28.3 control character for SATA primitives.
Second	Constant data character.
Third	Constant data character.
Last	Constant data character.

7.1.2 Primitive summary

Table 24 lists the SAS primitives.

Table 24. SAS primitives

Primitive	Use	From			To			Primitive sequence type
		I	E	T	I	E	T	
SAS primitives:								
AIP(NORMAL)	SAS		E		I	E	T	Single
AIP(WAITING ON PARTIAL)	SAS		E		I	E	T	Single
AIP(WAITING ON DEVICE)	SAS		E		I	E	T	Single
AIP(WAITING ON CONNECTION)	SAS		E		I	E	T	Single
AIP(4)	SAS		E		I	E	T	Single
ALIGN	All	I		T	I		T	Single
ALIGN(~~2~~1)	All	I		T	I		T	Single
ALIGN(~~3~~2)	All	I		T	I		T	Single
ALIGN(~~4~~3)	All	I		T	I		T	Single
BREAK	All	I		T	I	E	T	Redundant
CHANGE	SAS		E		I			Redundant
CHANGE(1)	SAS		E		I			Redundant
CHANGE(2)	SAS		E		I			Redundant
CHANGE(3)	SAS		E		I			Redundant
CLOSE	SAS	I		T	I		T	Triple
EOAF	SAS	I	E	T	I	E	T	Single
ERROR	SAS		E		I	E	T	Single
HARD RESET	SAS	I	E		I	E		Redundant
OPEN_ACCEPT	SAS	I		T	I		T	Single
OPEN_REJECT(NO DESTINATION)	SAS		E		I	E	T	Single
OPEN_REJECT(BAD DESTINATION)	SAS		E		I	E	T	Single
OPEN_REJECT(WRONG DESTINATION)	SAS	I	E	T	I	E	T	Single
OPEN_REJECT(LINK RATE NOT SUPPORTED)	SAS	I	E	T	I	E	T	Single
OPEN_REJECT(RETRY)	SAS	I	E	T	I	E	T	Single
OPEN_REJECT(PROTOCOL NOT SUPPORTED)	SAS	I	E	T	I	E	T	Single
OPEN_REJECT(NO MORE STP TASK FILE SETS)	SAS		E		I			Single
OPEN_REJECT(PATHWAY BUSY)	SAS		E		I		T	Single
OPEN_REJECT(8)	SAS							Single
SOAF	SAS	I	E	T	I	E	T	Single
SSP/SMP primitives:								
ACK	SSP	I		T	I		T	Single
DONE(CLOSE CONNECTION)	SSP	I		T	I		T	Single
DONE(CREDIT TIMEOUT)	SSP	I		T	I		T	Single
DONE(ACK/NAK TIMEOUT)	SSP	I		T	I		T	Single
DONE(3)	SSP	I		T	I		T	Single
EOF	SSP, SMP	I		T	I		T	Single
NAK(GENERAL ERROR)	SSP	I		T	I		T	Single
NAK(CRC ERROR)	SSP	I		T	I		T	Single
NAK(2)	SSP	I		T	I		T	Single
NAK(3)	SSP	I		T	I		T	Single
RRDY	SSP	I		T	I		T	Single
RRDY(1)	SSP	I		T	I		T	Single
SOF	SSP, SMP	I		T	I		T	Single

Table 25 lists the STP and SATA primitives and their usage.

Table 25. STP and SATA primitives

Primitive	Use	From			To			Primitive sequence type
		I	E	T	I	E	T	
SATA_CONT	STP, SATA	I		T	I		T	Single
SATA_DMAT	STP, SATA	I		T	I		T	Single
SATA_EOF	STP, SATA	I		T	I		T	Single
SATA_ERROR	SATA		E				T	Single
SATA_HOLD	STP, SATA	I		T	I		T	Repeated
SATA_HOLDA	STP, SATA	I		T	I		T	Repeated
SATA_PMACK	STP, SATA							Single
SATA_PMNAK	STP, SATA	I	E				T	Single
SATA_PMREQ_P	STP, SATA							Repeated
SATA_PMREQ_S	STP, SATA							Repeated
SATA_R_ERR	STP, SATA	I		T	I		T	Repeated
SATA_R_IP	STP, SATA	I		T	I		T	Repeated
SATA_R_OK	STP, SATA	I		T	I		T	Repeated
SATA_R_RDY	STP, SATA	I		T	I		T	Repeated
SATA_SOF	STP, SATA	I		T	I		T	Once
SATA_SYNC	STP, SATA	I		T	I		T	Repeated
SATA_WTRM	STP, SATA	I		T	I		T	Repeated
SATA_X_RDY	STP, SATA	I		T	I		T	Repeated

The Use columns indicate when primitives are used and is defined in Table 26.

Table 26. SAS primitive uses

Use	Description
SSP	SAS physical links, i~~I~~nside SSP connections
SMP	SAS physical links, i~~I~~nside SMP connections
STP	SAS physical links, i~~I~~nside STP connections
SAS	SAS physical links, o~~O~~utside connections
All	SAS physical links, b~~B~~oth outside connections or inside any type of connection
SATA	SATA physical links

The From and To columns indicate the type of ports that may originate each primitive: I for initiator ports, E for expander ports, and T for target ports. Expander ports are not considered originators of primitives that are passing through from initiator ports or target ports.

The Primitive sequence type columns indicate whether the primitive is sent as a repeated primitive sequence, a simple primitive sequence, or a redundant primitive sequence (see 7.1.3).

Table 27 defines the SAS primitive encoding.

Table 27. SAS primitive encoding

Primitive	1st	2nd	3rd	4th (last)
SAS primitives:				
AIP(NORMAL)	K28.5	D27.4	D27.4	D27.4
AIP(WAITING ON PARTIAL)	K28.5	D27.4	D24.0	D04.7
AIP(WAITING ON DEVICE)	K28.5	D27.4	D30.0	D29.7
AIP(WAITING ON CONNECTION)	K28.5	D27.4	D07.3	D24.0
AIP(4)	K28.5	D27.4	D01.4	D07.3
ALIGN	K28.5	D10.2	D10.2	D27.3
ALIGN(~~2~~1)	K28.5	D07.0	D07.0	D07.0
ALIGN(~~3~~2)	K28.5	D01.3	D01.3	D01.3
ALIGN(~~4~~3)	K28.5	D27.3	D27.3	D27.3
BREAK	K28.5	D02.0	D24.0	D07.3
CHANGE	K28.5	D02.0	D31.4	D30.0
CHANGE(1)	K28.5	D02.0	D04.7	D01.4
CHANGE(2)	K28.5	D02.0	D16.7	D31.4
CHANGE(3)	K28.5	D02.0	D29.7	D16.7
CLOSE	K28.5	D02.0	D30.0	D27.4
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
OPEN_ACCEPT	K28.5	D16.7	D16.7	D16.7
OPEN_REJECT(NO DESTINATION)	K28.5	D29.7	D29.7	D29.7
OPEN_REJECT(BAD DESTINATION)	K28.5	D29.7	D24.0	D01.4
OPEN_REJECT(WRONG DESTINATION)	K28.5	D29.7	D30.0	D31.4
OPEN_REJECT(LINK RATE NOT SUPPORTED)	K28.5	D29.7	D07.3	D16.7
OPEN_REJECT(RETRY)	K28.5	D29.7	D27.4	D24.0
OPEN_REJECT(PROTOCOL NOT SUPPORTED)	K28.5	D29.7	D31.4	D07.3
OPEN_REJECT(NO MORE STP TASK FILE SETS)	K28.5	D29.7	D04.7	D27.4
OPEN_REJECT(PATHWAY BUSY)	K28.5	D29.7	D16.7	D04.7
OPEN_REJECT(8)	K28.5	D29.7	D02.0	D30.0
SOAF	K28.5	D24.0	D30.0	D01.4
SSP/SMP primitives:				
ACK	K28.5	D01.4	D01.4	D01.4
DONE(CLOSE CONNECTION)	K28.5	D30.0	D30.0	D30.0
DONE(CREDIT TIMEOUT)	K28.5	D30.0	D07.3	D27.4
DONE(ACK/NAK TIMEOUT)	K28.5	D30.0	D01.4	D04.7
DONE(3)	K28.5	D30.0	D27.4	D29.7
EOF	K28.5	D24.0	D16.7	D27.4
NAK(GENERAL ERROR)	K28.5	D01.4	D27.4	D04.7
NAK(CRC ERROR)	K28.5	D01.4	D31.4	D29.7
NAK(2)	K28.5	D01.4	D04.7	D24.0
NAK(3)	K28.5	D01.4	D16.7	D07.3
RRDY	K28.5	D01.4	D24.0	D16.7
RRDY(1)	K28.5	D01.4	D07.3	D30.0
SOF	K28.5	D24.0	D04.7	D07.3

Table 28 lists the STP and SATA primitive encodings. Except for SATA_ERROR, ~~These~~ these values are defined by SATA.

Table 28. STP/SATA primitive encoding

Primitive	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	K28.3	K28.3	K28.3	K28.3
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

7.1.3 Primitive sequences

7.1.3.1 Primitive sequence overview

Table 29 summarizes the types of primitive sequences.

Table 29. Primitive sequences

Primitive sequence type	Number of times sent	Number of times received to detect
Single	1	1
Repeated	2	1
Triple	3	3
Redundant	6	3

ALIGNs may be sent inside primitive sequences without affecting the count or breaking the consecutiveness requirements. Rate matching ALIGNs shall be sent inside primitive sequences inside of connections if rate matching is enabled. Any number of ALIGNs may be inserted in primitive sequences.

7.1.3.2 Single primitive sequence

Primitives labeled as single primitive sequences are sent one time.

7.1.3.3 Repeated primitive sequence

STP primitives that form repeated primitive sequences (e.g., SATA_HOLD) shall be sent two times, then be followed by SATA_CONT, then be followed by random scrambled data until a response is received.

Figure 44 shows an example of a repeated primitive sequence.

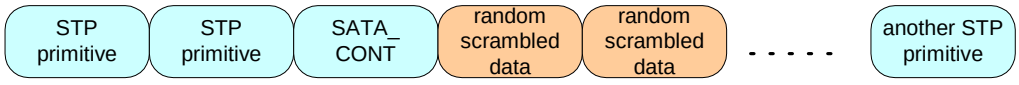

Figure 44. Repeated primitive sequence

7.1.3.4 Triple primitive sequence

SAS and SSP primitives that form triple primitive sequences (e.g., BREAK) shall be sent three times consecutively and followed by idle dwords until a response is received. ALIGNs may be sent inside primitive sequences without affecting the count or breaking the consecutiveness requirements.

Receivers shall detect a triple primitive sequence by receiving the identical primitive in three consecutive dwords. Whenever absence of a triple primitive sequence is meaningful, receivers shall decode three idle dwords before acting (e.g., at the end of closing a connection). A receiver shall not detect primitive sequences a second time until it has received three consecutive dwords that are not any of the following:

a) the original primitive; or
b) ALIGN primitives.

Figure 45 shows an example of a simple primitive sequence.

Figure 45. Simple primitive sequence

7.1.3.5 Redundant primitive sequence

SAS primitives that form redundant primitive sequences (e.g., CHANGE) shall be sent six times consecutively. ALIGNs may be sent inside primitive sequences without affecting the count or breaking the consecutiveness requirements.

A receiver shall detect redundant sequences by receiving an identical primitive for three consecutive dwords. Redundant sequences shall only be detected outside of a connection. A receiver shall not detect redundant sequences a second time until it has received six consecutive dwords that are not any of the following:

a) the original primitive; or
b) the ALIGN primitive.

Figure 46 shows an example of a broadcast primitive sequence.

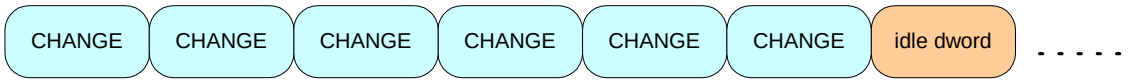

Figure 46. Redundant primitive sequence

7.1.4 SAS primitives

7.1.4.1 AIP

AIP (arbitration in progress) is sent by an expander device after a connection request to indicate that the connection request is being processed.

There are several versions of AIP, each indicating a different state of the connection request. The versions are defined in Table 30.

Table 30. AIP primitives

Primitive	Description
AIP(NORMAL)	Expander device has just accepted the connection request.
AIP(WAITING ON PARTIAL)	Expander device has determined the routing for the connection request, but the destination phys are all busy with other connection requests that have also received AIP(WAITING ON PARTIAL).
AIP(WAITING ON DEVICE)	Expander device has determined the routing for the connection request and forwarded it to the output physical link.
AIP(WAITING ON CONNECTION)	Expander device has determined the routing for the connection request, but the destination phys are all busy with active connections.
AIP(4)	Reserved.

See 7.11 for details on connections.

7.1.4.2 ALIGN, ALIGN(1), ALIGN(2), and ALIGN(3)

ALIGN primitives are used for character and dword alignment during the reset sequence, and ~~is used to maintain dword alignment thereafter~~are used for rate machine and clock skew management thereafter.

~~ALIGN is the only K28.5 primitive defined by SATA, and is the only SATA primitive including the comma character used for character alignment. It has neutral disparity - disparity is the same after the ALIGN as it was before the ALIGN.~~

~~ALIGN(2), ALIGN(3), and ALIGN(4) may be used interchangeably with ALIGN after the reset sequence has completed. Rate matching requires the sender rotate through them to reduce EMI~~ (see 7.11.8.3).

ALIGN shall be used during the reset sequence. ALIGN, ALIGN(1), ALIGN(2), and ALIGN(3) may be used interchangeably after the reset sequence. SAS ports shall accept any of the ALIGN primitives after the reset sequence has completed.

SAS ports that are capable of sending ALIGN primitives with less than seven non-ALIGN dwords between them shall rotate through all of the ALIGN primitives. SAS ports that are capable of rate matching (see 7.11.8.3).shall rotate through all of the ALIGN primitives. Any other SAS port may transmit ALIGN alone or may rotate through all the ALIGN primitives.

SAS ports that rotate through the ALIGN primitives shall transmit ALIGN, then ALIGN(1), then ALIGN(2), then ALIGN(3), then ALIGN, etc. SAS devices that do not transmit all of the ALIGN primitives shall transmit ALIGN only.

See 7.2 for requirements for ALIGN insertion and other details on clock skew management.

7.1.4.3 BREAK

BREAK is used to abandon an open connection request or break a connection.

See 7.11.5 and 7.11.6 for details on breaking connections.

7.1.4.4 CHANGE

CHANGE is sent by an expander device to notify initiator ports and other expander devices that a configuration change has occurred. CHANGE primitive sequences shall only be sent outside of connections.

CHANGE(1), CHANGE(2), and CHANGE(3) are reserved.

See 7.8 for details on domain changes.

7.1.4.5 CLOSE

CLOSE is used to close an open connection. This primitive may be originated by an SSP or STP initiator port, an SSP target port, or by an expander device on behalf of an SATA target port.

See 7.11.7 for details on closing connections.

7.1.4.6 EOAF

EOAF indicates the end of an address frame.

See 7.4 for details on address frames.

7.1.4.7 ERROR

ERROR is sent by an expander when it is forwarding dwords from one link to another and it receives an invalid dword.

See 7.11.8.3 for details on error handling by expanders.

~~7.1.4.7~~7.1.4.8 HARD RESET

HARD RESET is used to force a phy to generate a hard reset to its port. This primitive is only valid after the link reset sequence before the IDENTIFY address frame.

See 4.6 for details on hard resets.

~~7.1.4.8~~7.1.4.9 OPEN_ACCEPT

OPEN_ACCEPT indicates the acceptance of a connection request.

See 7.11 for details on connection requests.

~~7.1.4.9~~7.1.4.10 OPEN_REJECT

OPEN_REJECT indicates that a connection request has been rejected.

There are several versions of OPEN_REJECT, each indicating a different reason for rejecting the connection request. The versions are defined in Table 31.

Table 31. OPEN_REJECT primitives

Primitive	Sender	Response	Description
OPEN_REJECT(NO DESTINATION)	Expander device	Abandon connection request.	No such destination device.
OPEN_REJECT(BAD DESTINATION)	Expander device	Abandon connection request.	The destination device name equals the source device name or the expander devices detected a routing problem.
OPEN_REJECT(WRONG DESTINATION)	Destination device	Retry until I_T nexus loss timer expires.	The destination device name does not match the end device.
OPEN_REJECT(LINK RATE NOT SUPPORTED)	Expander device	Retry until I_T nexus loss timer expires.	Requested physical link rate is not supported on some link between the source device and destination device.
OPEN_REJECT(RETRY)	Destination device	Retry connection request.	Destination device exists but is not able to accept connections; try later.
OPEN_REJECT(PROTOCOL NOT SUPPORTED)	Destination device	Abandon connection request.	Destination device exists but does not support the requested initiator/target role and/or protocol.
OPEN_REJECT(TOO MANY STP INITIATORS)	Expander device	Abandon connection request.	Destination device exists but there are too many STP initiators are trying to access the edge expander device.
OPEN_REJECT(PATHWAY BLOCKED)	Expander device	Retry connection request.	An expander determined the pathway was blocked by higher priority connection requests.
OPEN_REJECT(8)			Reserved.

See 7.11 for details on connection requests.

~~7.1.4.10~~7.1.4.11 SOAF

SOAF indicates the start of an address frame.

See 7.4 for details on address frames.

7.1.5 SSP primitives

7.1.5.1 ACK

ACK indicates the positive acknowledgement of an SSP frame.

See 7.13.2 for details on SSP frame transmission.

7.1.5.2 DONE

DONE is used to start closing an SSP connection. 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.

There are several versions of DONE, each indicating a different reason for closing the connection. The versions are defined in Table 32. The SSP state machine section describes when these are used (see 7.13.6).

Table 32. DONE primitives

Primitive	Description
DONE(CLOSE CONNECTION)	Finished sending all frames.
DONE(CREDIT TIMEOUT)	Timed out waiting for an RRDY.
DONE(ACK/NAK TIMEOUT)	Timed out waiting for an ACK or NAK. The ACK/NAK count does not match the frame count. Sender is going to send BREAK in 1 ms unless DONE is received prior to that.
DONE(3)	Reserved

See 7.13.5 for details on closing SSP connections.

7.1.5.3 EOF

EOF indicates the end of an SSP or SMP frame.

See 7.13.2 for details on SSP frame transmission and 7.15.1 for details on SMP frame transmission.

7.1.5.4 NAK

NAK indicates the negative acknowledgement of an SSP frame.

There are several versions of NAK, each indicating a different reason for the negative acknowledgement. The versions are defined in Table 33.

Table 33. NAK primitives

Primitive	Description
NAK(GENERAL ERROR)	The frame was rejected; no further reason is available.
NAK(CRC ERROR)	There is room to store the frame, but the frame had a bad CRC.
NAK(2)	Reserved
NAK(3)	Reserved

See 7.13.2 for details on SSP frame transmission.

7.1.5.5 RRDY

RRDY is used to increase SSP frame credit.

RRDY(1) is reserved.

See 7.13.3 for details on SSP flow control.

7.1.5.6 SOF

SOF indicates the start of an SSP or SMP frame.

See 7.13.2 for details on SSP frame transmission and 7.15.1 for details on SMP frame transmission.

7.1.6 STP primitives

7.1.6.1 SATA_PMACK, SATA_PMNAK, SATA_PMREQ_P, and SATA_PMREQ_S

SATA_PMREQ_P and SATA_PMREQ_S request entry into the interface power management partial and slumber states. SATA_PMACK is used to accept a power management request. SATA_PMNAK is used to reject a power management request.

See 7.4 for rules on handling the power management primitives.

7.1.6.2 SATA_HOLD and SATA_HOLDA

SATA assumes that once a SATA_HOLD primitive is sent, SATA_HOLDA will arrive within 20 dwords. This limits the amount of buffering required in the SATA device sending SATA_HOLD.

Since the physical link from an expander device to a target port running SATA protocol must adhere to this limit, expander devices should introduce as little latency as possible during STP connections. The expander device shall generate its own SATA_HOLDs to throttle data for any latency it adds.

7.1.6.3 SATA_R_RDY and SATA_X_RDY

When a SATA port has a frame to send, it asserts SATA_X_RDY and waits for SATA_R_RDY to send the frame. Expander devices shall defer sending SATA_R_RDY until the STP connection is established.

7.1.6.4 Other STP primitives

Other primitives are used in STP connections as defined in SATA.

7.2 Clock skew management

The internal clock for a device is typically based on a PLL with its own clock generator. This is used when transmitting data onto the physical link. When receiving, however, data must be latched based on a clock derived from the input data itself. Although the input clock is nominally a fixed rate, it can differ slightly from the internal clock due to accepted manufacturing tolerance and, for SATA links, due to spread spectrum clocking. Over time, if the input clock is faster than the internal clock, the device may receive data and not be able to forward it to an internal buffer; this is called an overrun. If the input clock is slower than the internal clock, the device may not have data when needed in an internal buffer; this is called an underrun.

To solve this, devices shall insert ALIGN primitives in the data stream. Receivers may pass the ALIGN through to their internal buffers, or may strip it out when an overrun occurs. Receivers may add ALIGN primitives when an underrun occurs. The internal logic shall ignore all ALIGN primitives.

Elasticity buffer circuitry, as shown in Figure 47, is required to absorb the slight differences in frequencies between the initiator port, target port, and expander device. Each port shall have a frequency tolerance of +/- 100 ppm. The maximum frequency difference shall be 200 ppm.

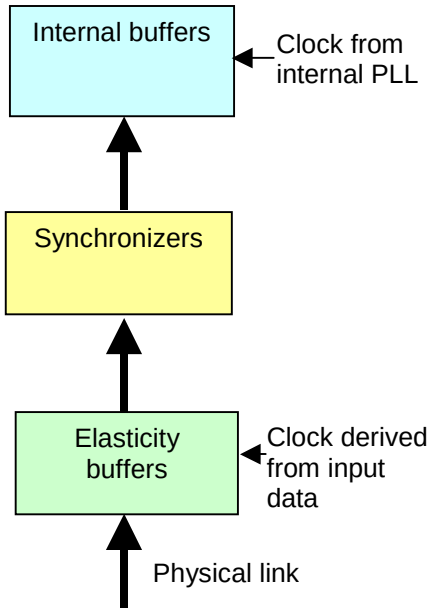

Figure 47. Elasticity buffers

In SATA, devices may assume an ALIGN is being sent solely by detecting the K28.5 control character. Since SAS defines numerous K28.5 primitives, SAS devices shall decode all four characters in a primitive before assuming it is an ALIGN.

In SSP and SMP, ports shall not send ALIGNs during a frame because a dword is not available.

A SAS port that is the original source for the data shall periodically insert ALIGNs into the data stream. No more than 2 047 dwords shall be sent without inserting an ALIGN. ALIGNs may be inserted more often.

A STP initiator port shall insert extra ALIGNs. No more than 254 dwords shall be sent without inserting two consecutive ALIGNs. ALIGNs may be inserted more often but shall be inserted as two consecutive ALIGNs. These extra ALIGNs are in addition to the ALIGN required for all SAS ports between every 2 047 dwords.

An expander device that is repeating data is allowed to insert or delete as many ALIGNs as required to match the transmit and receive rates. One possible implementation is for the expander device to delete all ALIGNs at the receive port and to insert ALIGNs at the transmit port whenever its elasticity buffer is empty.

An expander device that is attached to a SATA target device is allowed to insert or delete as many ALIGNs as required on the STP side to match the transmit and receive rates. One possible implementation is for the expander to delete all ALIGNs at the receive port and to insert two consecutive ALIGNs at the transmit port when its elasticity buffer is empty or when 245 non-ALIGN dwords have been transmitted. This expander device is not required to insert ALIGNs in pairs when transmitting data to the initiator on the STP side.

7.3 Idle links

While no connection is open and a logical link is idle, or while an SSP or SMP connection is open and the logical link is idle, SAS devices send random scrambled data called idle dwords.

While an STP connection is open, between frames, SAS STP devices send SATA_SYNC primitives. After sending two SATA_SYNC primitives, SAS STP devices shall send SATA_CONT and start sending random scrambled data. SATA devices are allowed but not required to send SATA_CONT.

See 7.9 for details on scrambling.

7.4 Address frames

7.4.1 Address frames overview

Address frames are used for physical link initialization, connection requests, and routing table exchange. The address frame follows an SOAF primitive and ends with an EOAF primitive. Primitives may be inserted in the address frame. Address frames shall be sent only outside connections. Address frames shall not be terminated early. All data in an address frame is scrambled.

Table 34 describes the address frame format.

Table 34. Address frame format

Byte	7	6	5	4	3	2	1	0
0					ADDRESS FRAME TYPE			
1		Frame type dependent bytes						
27								
28	(MSB)	CRC						
31								(LSB)

The ADDRESS FRAME TYPE field indicates the type of address frame and is defined in Table 35. This field determines the definition of the frame type dependent bytes.

Table 35. Address frame types

ADDRESS FRAME TYPE	Frame type	Description
0h	Identify	Physical link initialization
1h	Open	Connection request
All others		Reserved

The CRC field is a CRC value (see 7.10) that is computed over the entire address frame.

Address frames with unknown address frame types or with CRC errors shall be ignored by the recipient.

7.4.2 IDENTIFY address frame

Table 36 defines the IDENTIFY address frame format used for physical link initialization. The IDENTIFY address frame is sent after the physical link reset sequence completes indicating a SAS physical link.

Table 36. IDENTIFY address frame format

Byte	7	6	5	4	3	2	1	0
0	Reserved				ADDRESS FRAME TYPE (0h)			
1	PHY IDENTIFIER							
2	Reserved				MAXIMUM PHYSICAL LINK RATE			
3	DEVICE TYPE		STP INITIATOR	STP TARGET	SSP INITIATOR	SSP TARGET	SMP INITIATOR	SMP TARGET
4	Reserved							
19								
20	(MSB)	DEVICE NAME						
27								(LSB)
28	(MSB)	CRC						
31								(LSB)

The PHY IDENTIFIER field indicates the phy identifier of the phy that is sending the IDENTIFY address frame.

The MAXIMUM PHYSICAL LINK RATE field indicates the maximum physical link rate supported by the phy and is defined in Table 37.

Table 37. Maximum physical link rate

MAXIMUM PHYSICAL LINK RATE	Maximum physical link rate
0h	1,5 Gbps
1h	3,0 Gbps
2h - Fh	Reserved

The DEVICE TYPE field indicates the type of device containing the phy, and is defined in Table 38. A device which is capable of being both an end device and an expander device shall only report its expander device type in this field.

Table 38. Device types

DEVICE TYPE	Device type
00b	End device only
01b	Edge expander device
10b	Fanout expander device
11b	Reserved

The STP INITIATOR bit indicates the device is an STP initiator device or STP target/initiator device.

The STP TARGET bit indicates the device is an STP target device or STP target/initiator device.

The SSP INITIATOR bit indicates the device is an SSP initiator device or SSP target/initiator device.

The SSP TARGET bit indicates the device is an SSP target device or SSP target/initiator device.

The SMP INITIATOR bit indicates the device is an SMP initiator device or SMP target/initiator device.

The SMP TARGET bit indicates the device is an SMP target device or SMP target/initiator device.

The DEVICE NAME field indicates the device name of the device sending the IDENTIFY address frame.

7.4.3 OPEN address frame

Table 39 defines the OPEN address frame format used for connection requests.

Table 39. OPEN address frame format

Byte	7	6	5	4	3	2	1	0
0	INITIATOR	Rsvd	PROTOCOL		ADDRESS FRAME TYPE (1h)			
1	Reserved				LINK RATE			
2	(MSB)							
3	INITIATOR CONNECTION TAG							(LSB)
4								
9	Reserved							
10	SCALE	(MSB)						
11	ARBITRATION WAIT TIME							(LSB)
12	(MSB)							
19	DESTINATION DEVICE NAME							(LSB)
20	(MSB)							
27	SOURCE DEVICE NAME							(LSB)
28	(MSB)							
31	CRC							(LSB)

The PROTOCOL field indicates the protocol for the connection being requested and is defined in Table 40.

Table 40. Protocol

PROTOCOL	Protocol
00b	SMP
01b	SSP
10b	STP
11b	Reserved

An INITIATOR bit set to one indicates the source device is acting as an initiator device and, if the protocol is SSP, will grant credit (i.e., will send RRDY) when the connection is established. An INITIATOR bit set to zero indicates the source device is acting as a target device and, if the protocol is SSP, may not grant credit when the connection is established.

If a target/initiator device sets the INITIATOR bit to one, it shall operate only in its initiator role during the connection. If a target/initiator device sets the INITIATOR bit to zero, it shall operate only in its target role during the connection.

An initiator-only device shall reject connection requests from an initiator device with OPEN_REJECT(PROTOCOL NOT SUPPORTED). A target-only device shall reject connection requests from a target device with OPEN_REJECT(PROTOCOL NOT SUPPORTED).

The LINK RATE field indicates the rate of the link on which the OPEN address frame originated, and the rate at which all links on the pathway must support, and is defined in Table 41.

Table 41. Link rate

LINK RATE	Link rate
0h	1,5 Gbps
1h	3,0 Gbps
2h - Fh	Reserved

The INITIATOR CONNECTION TAG field is used for SSP and STP connection requests to provide an initiator port an easier context lookup when the target port originates a connection request. SSP or STP initiator ports shall set the INITIATOR CONNECTION TAG field to FFFFh if they do not require the field be provided by the target port. Otherwise, an SSP or STP initiator port shall set the INITIATOR CONNECTION TAG field to a unique value per target port. When requesting a connection to an initiator port, a target port shall set the INITIATOR CONNECTION TAG field to the value received in connection requests from the initiator port. An initiator port shall use the same tag for all connection requests to the same target port, and may only change the tag when it has no commands outstanding to that

target port. Targets are not required to check consistency of the tags in different connection requests from the same initiator port. SMP initiator ports shall set the INITIATOR CONNECTION TAG field to FFFFh.

The SCALE bit and the ARBITRATION WAIT TIME field indicate how long the port sending the OPEN address frame has been waiting for a connection request to be accepted, and are defined in Table 42. See 7.11.3 for details on arbitration fairness.

Table 42. Arbitration wait time

SCALE	ARBITRATION WAIT TIME		
	Description	**Minimum value (0000h) meaning**	**Maximum value (7FFFh) meaning**
0b	Microseconds since the connection request was first made	0 µs	32 767 µs
1b	Milliseconds since the connection request was first made minus 32 768 microseconds	32 768 µs	32 768 ms + 32 768 µs

The DESTINATION DEVICE NAME field indicates the device name of the port with which a connection is being requested.

The SOURCE DEVICE NAME field indicates the device name of the port sending the OPEN address frame.

7.5 Identification sequence

7.5.1 Overview

After the phy reset sequence has been completed indicating the physical link is using SAS rather than SATA, each phy shall send an IDENTIFY address frame (see 7.4.2) carrying its device name and other information. Each phy shall also expect to receive an IDENTIFY address frame from the phy to which it is attached. The combination of a phy reset sequence and the identification sequence is called a link reset sequence.

If a device supports more than one phy, it shall send the same device name on all phys for which it is capable of sharing within a port.

If a device detects the same device name incoming on different physical links, it shall treat those physical links as a wide link and consider those phys part of the same port.

If a device detects different device names incoming on different physical links, it shall treat those physical links as independent physical links and consider those phys part of different ports attached to different SAS domains.

If a device does not receive a valid IDENTIFY address frame within 1 ms of phy reset sequence completion, it should send its IDENTIFY address frame again. If IDENTIFY continues to not show up, it may restart the phy reset sequence.

If a device receives an additional IDENTIFY address frame after receiving the first one, it shall resend its own IDENTIFY address frame.

7.5.2 Initiator device specific rules

After identifying that it is attached to an expander device after a link reset sequence, or after receiving a CHANGE primitive sequence, an initiator port should perform a level-order traversal of the domain by opening an SMP connection to each expander device and use the DISCOVER function (see 9.4.4.2) to retrieve a list of attached device names . The order of traversal should be:

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

When this is done after a link reset sequence, this lets the initiator discover information about all the devices in the domain. When this is done after a CHANGE, this lets the initiator port determine what changed in the domain.

This information may be used to select link rates for connection requests.

7.5.3 Fanout expander device specific rules

After learning that it is attached to an edge expander device, a fanout expander device shall use the SMP DISCOVER function (see 9.4.4.2) to retrieve the list of device names to which the edge expander device is attached.

After receiving a CHANGE primitive sequence from an edge expander device, the fanout expander device shall use the SMP DISCOVER function to obtain an updated list of device names from that edge expander device.

7.6 Power management

SATA interface power management is not supported in SAS.

STP initiator ports shall not generate SATA_PMREQ_P, SATA_PMREQ_S, or SATA_PMACK. If an STP initiator port receives SATA_PMREQ_P or SATA_PMREQ_S, it shall reply with SATA_PMNAK.

If an expander device receives SATA_PMREQ_P or SATA_PMREQ_S from a SATA target port while an STP connection is not open, it shall not forward it to any initiator port and shall reply with a SATA_PMNAK. If the primitives arrives while an STP connection is open, it may forward it to the STP initiator port.

An expander device may intercept SATA register FIS transfers and hide the existence of support for interface power management in the SCONTROL and SSTATUS registers.

SCSI idle and standby power conditions, implemented with the START STOP UNIT command (see SBC-2) and the Power Condition mode page (see SPC-3), may be supported by SSP initiator ports and target ports. The SCSI sleep power condition shall not be used.

ATA idle and standby power modes, implemented with the IDLE, IDLE IMMEDIATE, STANDBY, STANDBY IMMEDIATE, and CHECK POWER MODE commands (see ATA), may be supported by STP initiator ports. The ATA sleep power mode, implemented with the SLEEP command, shall not be used.

7.7 Tests

7.7.1 Tests overview

Table 43 lists the test modes defined for SAS devices.

Table 43. Test modes

Test	Started by
Far-end retimed loopback	SMP
Near-end analog loopback	Initiator phys: vendor-specific Expander phys: SMP

Figure 48 shows the test modes.

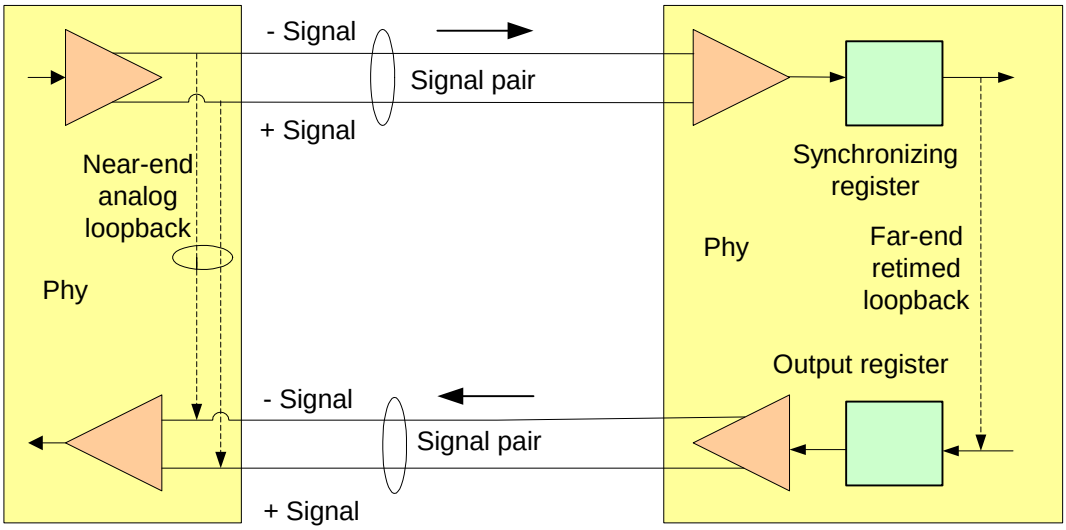

Figure 48. Test modes

7.7.1.1 Near-end analog loopback test

A phy performing the near end analog loopback test shall internally route its transmitter to its receiver. The phy shall not transmit anything on the link and shall ignore all incoming signals from the link, including OOB signals.

This test mode may be invoked in an initiator device using vendor-specific means.

An application client may request an expander device set one of its phys into near-end analog loopback test mode using the SMP PHY_CONTROL function. After enabled, all data sent to that expander device shall be routed to the expander phy to be looped back. Expanders in the pathway treat this as an open SMP connection.

Once the test is completed, the application client shall send a BREAK or CLOSE primitive sequence or have the initiator phy start a phy reset sequence. The expander device shall disable the loopback mode and start a phy reset sequence on that phy. The phy device shall loopback each of the BREAK or CLOSE primitives before the loopback mode is disabled.

7.7.1.2 Far-end retimed loopback test

A phy performing the far-end retimed loopback test shall internally route its receiver to its transmitter, after synchronizing incoming data to its internal clock. The device in which the phy is located shall not forward the data internally (e.g. if it is an expander device, it shall not forward it to any other expander ports).

This test mode may be invoked on an expander phy or a target phy using the SMP PHY_CONTROL function. Expanders in the pathway treat this as an open SMP connection.

The phy shall remain in this mode until a BREAK or CLOSE primitive sequence is detected or until a phy reset sequence is detected. The phy shall loopback each of the BREAK or CLOSE primitives before disabling the loopback mode.

7.8 Domain changes

SAS initiator ports scan the domain with SMP (see 7.15) to search for expander devices and target ports after power on or receiving a CHANGE primitive sequence.

The CHANGE primitive sequence shall only be sent outside of a connection. The expander device shall send CHANGE to one physical link attached to each expander port. The expander device should not send CHANGE to more than one physical link per expander port.

CHANGE shall not be sent to any phy that is part of the expander port that is the cause for sending CHANGE.

Expander devices shall send CHANGE for the following reasons:

a) after an expander phy has lost bit synchronization;

b) after the link reset sequence completes; and
d) after the expander device receives CHANGE.

CHANGE shall only be sent outside of connections. CHANGE may be sent by initiator ports to force expander devices to exchange device names, but should not be sent by target ports.

An expander device is not required to queue multiple CHANGE indications for the same expander port. If a second CHANGE indication is requested before the first indication has been transmitted, the second indication may be dropped.

A fanout expander device that detects CHANGE on a link to an edge expander device shall use the SMP DISCOVER function to detect any change in the list of device names attached the edge expander device (see 7.5.3).

An Initiator port that detects CHANGE should perform a level order traversal through the domain until it has discovered all expander devices. If the initiator port detects a port with a WWN it has already encountered, it has found a loop. It should disable the expander port used to reach this previously-encountered port to break the loop.

7.9 Scrambling

Scrambling is used to reduce the probability that long strings of zeros or ones do not appear on the physical link. These patterns can cause issues in the physical layers.

There are several types of scrambling:

a) Repeated SATA primitive scrambling. After a SATA_CONT primitive, random scrambled data is sent until a primitive other than ALIGN is sent;
b) SAS idle dwords. When a connection is not open and the physical link is idle, random scrambled data is sent;
c) SSP idle dwords. When an SSP connection is open and the physical link is idle (i.e., between frames), random scrambled data is sent;
d) Address frame scrambling. After an SOAF primitive, all data is scrambled until the EOAF primitive;
e) SSP frame data scrambling. During an SSP connection after an SOF primitive, all data is scrambled until the EOF primitive;
f) STP frame data scrambling. During an STP connection after a SATA_SOF primitive, all data is scrambled until the SATA_EOF primitive; and
g) SMP frame data scrambling. During an SMP connection after an SOF primitive, all data is scrambled until the EOF primitive.

The scrambling polynomial is $x^{16} + x^{15} + x^{13} + x^4 + 1$. The data scrambling value shall be initialized to FFFFh at each SOF and SATA_SOF by both the transmitter and receiver. The data being transmitted shall be XORed with the data scrambling value by the transmitter, and the data being received shall be XORed with the data scrambling value by the receiver.

Table 44 shows whether the scrambling logic shall treat data as big-endian or little-endian.

Table 44. Scrambling endianness

Data	Scrambling endianness
Address frame	Big-endian
SSP frame	Big-endian
SMP frame	Big-endian
STP frame	Little-endian
Repeated SATA primitive	Little-endian
SAS or SSP idle dwords	Vendor-specific

[Editor's note: whether scrambling happens before or after bits within each byte are reversed for 8b10b coding and CRC generation is TBD]

Annex C contains information on scrambling implementations.

7.10 CRC

7.10.1 CRC overview

All frames include cyclic redundancy check (CRC) values to help detect transmission errors.

Frames transmitted in a STP connection shall include a CRC as defined by SATA. Address frames, SSP frames, and SMP frames shall include a CRC as defined by this standard.

Annex A contains information on CRC generation/checker implementation.

Table 45 shows whether the CRC logic shall treat data as big-endian or little-endian.

Table 45. CRC endianness

Data	CRC endianness
Address frame	Big-endian
SSP frame	Big-endian
SMP frame	Big-endian
STP frame	Little-endian

Table 46 defines the CRC polynomials.

Table 46. CRC polynomials

Function	Definition
F(x)	A polynomial of degree k-1 that is used to represent the k bits of the frame covered by the CRC. For the purposes of the CRC, the coefficient of the highest order term shall be the first bit transmitted.
L(x)	A degree 31 polynomial with all of the coefficients equal to one, i.e., $L(x) = X^{31} + X^{30} + X^{29} + \ldots + X^2 + X^1 + 1$
G(x)	The standard generator polynomial: $G(x) = X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
R(x)	The remainder polynomial which is of degree less than 32.
P(x)	The remainder polynomial on the receive checking side which is of degree less than 32.
Q(x)	The greatest multiple of G(x) in $(X^{32} F(x)) + (X^k L(x))$
Q*(x)	$X^{32} Q(x)$
M(x)	The sequence which is transmitted.
M*(x)	The sequence which is received.
C(x)	A unique polynomial remainder produced by the receiver upon reception of an error free sequence. This polynomial has the value: $C(X) = X^{32} L(X) / G(X)$ $C(x) = X^{31} + X^{30} + X^{26} + X^{25} + X^{24} + X^{18} + X^{15} + X^{14} + X^{12} + X^{11} + X^{10} + X^8 + X^6 + X^5 + X^4 + X^3 + X + 1$

7.10.2 CRC generation

The equations that are used to generate the CRC from F(x) are as follows:

In the following equation, note that adding L(x) (all ones) to R(x) simply produces the one''s complement of R(x); this equation is specifying that the R(x) is inverted before it is sent out:

CRC value in frames = L(X)+R(X) = R$(X)

where R$(X) is the one''s complement of R(X)

The CRC is calculated by:

$(X^{32} F(x) + X^k L(X)) / G(X) = Q(X) + R(X) / G(X)$

The following equation specifies that the CRC is appended to the end of F(x):

$M(x) = x^{32} F(x) + CRC$

NOTE - All arithmetic is modulo 2.

The bit order of F(x) presented to the CRC function is the same order as the bit transmission order. This order is shown in Figure 49.

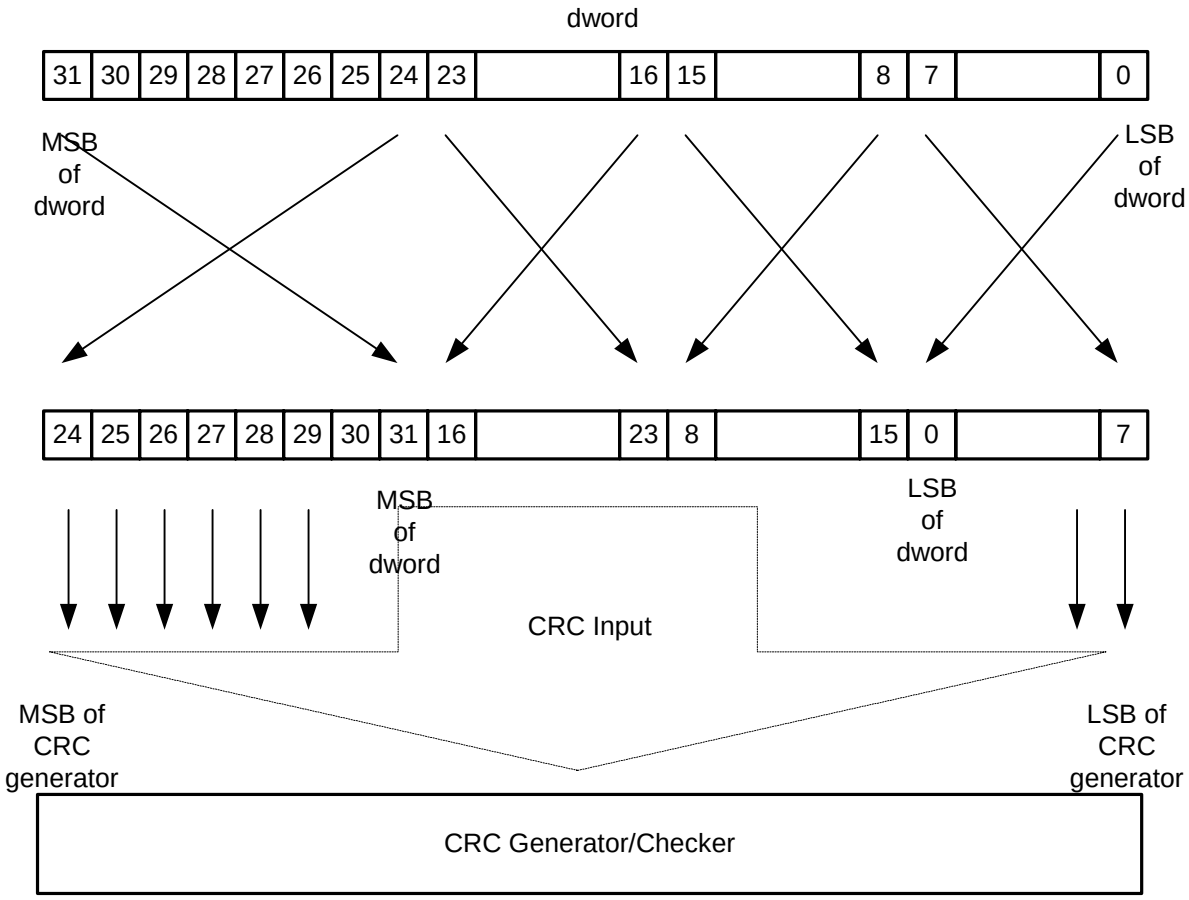

Figure 49. CRC generator bit order

7.10.3 CRC checking

The received sequence M*(x) may differ from the transmitted sequence M(x) if there are transmission errors. The process of checking the sequence for validity involves dividing the received sequence by G(x) and testing the remainder. Direct division, however, does not yield a unique remainder because of the possibility of leading zeros. Thus a term L(x) is prepended to M*(x) before it is divided. Mathematically, the received checking is shown in the following equation:

$$X^{32} (M^*(X)+X^k L(X)) / G(X)=Q^*(X) + P(X) / G(X)$$

In the absence of errors, the unique remainder is the remainder of the division:

$$P(X) / G(X) = X^{32} L(X) / G(X) = C(X)$$

The bit order of F(x) presented to the CRC checkering function is the same order as the bit transmission order. This order is shown in Figure 49. Annex A contains examples of CRC generation/checker implementations.

7.11 Connections

7.11.1 Connection overview

A connection is opened between an initiator port and a target port before communication can begin.

SSP initiator ports open SSP connections to send SCSI commands, task management functions, or transfer data. SSP target ports open SSP connections to transfer data or send status.

STP initiator ports open STP connections to send SATA frames. Expander devices open STP connections on behalf of SATA target ports to send SATA frames.

The OPEN address frame is used to request that a connection be opened. AIP, OPEN_ACCEPT and OPEN_REJECT are the responses to an OPEN address frame. BREAK is used to abandon connection requests and to unilaterally break a connection. CLOSE is used for orderly closing a connection.

For an SSP connection, connections stretch from the initiator port to the target port. For an STP connection, connections exist between the initiator port and the expander device attached to a SATA target port. The SATA target port is not aware of SAS connection management.

7.11.2 Opening a connection

7.11.2.1 Connection request

The OPEN address frame (see 7.4.3) is used to open a connection from a source port to a destination port.

To make a connection request, the source port shall send an OPEN address frame. This frame contains the source device name, destination device name, protocol (SSP, STP, or SMP), and link rate. The source port shall send idle dwords after the OPEN address frame until it receives a response or decides to abandon the connection request with BREAK.

After sending an OPEN address frame, the source port shall initialize an open timeout timer to 1 millisecond and start the timer. Whenever an AIP primitive is received, the source port shall reinitialize and restart the timer. Source ports are not required to enforce a limit on the number of AIPs received before abandoning the connection request, but they may do so. When any connection response is received, the source port shall reinitialize the timer. If the timer expires before a connection response is received, the source port may assume the destination port does not exist and shall send BREAK to abandon the connection request.

The OPEN address frame flows through expander devices onto intermediate physical links. If one of the intermediate physical links does not support the requested link rate, the expander shall return OPEN_REJECT (LINK RATE NOT SUPPORTED). If the OPEN address frame reaches the destination, it shall return either OPEN_ACCEPT or OPEN_REJECT. Rate matching is used on any physical links in the pathway that are faster than the requested link rate.

7.11.2.2 Connection request responses

Table 47 shows the possible responses to an OPEN address frame.

Table 47. Connection request responses

Response	Description
AIP	Arbitration in progress. While the expander devices are trying to open a connection to the selected destination port, it returns an AIP primitive to the source port. The source port shall set its open timeout timer back to its initial value when it receives an AIP primitive. AIP is sent by an expander device while it is internally arbitrating for access to an expander port.
OPEN_ACCEPT	Connection request accepted. This is sent by the destination port.
OPEN_REJECT	Connection request rejected. This is sent in response by the destination port or by an expander device. The different versions are described in 7.1.4.10.
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. See 7.11.3 for details.
BREAK	The destination port or expander port may reply with BREAK indicating the connection is not being established.
No response; timer expires	The source port shall abandon the connection request by sending BREAK.

An initiator port or target port shall accept an incoming connection request if the DESTINATION DEVICE NAME field matches its device name and the PROTOCOL field matches one of its supported protocols.

7.11.3 Arbitration fairness

SAS supports least-recently used arbitration fairness.

Each initiator port, target port, and expander port may include an arbitration wait timer counting time from when the port makes a connection request until its request is granted. The arbitration wait timer shall count in microseconds from 0 ms to 32 767 ms. The timer shall stop incrementing when its value reaches 32 767 ms.

Initiator ports and target ports should implement arbitration wait timers. If implemented, they shall set the timer to zero and start the timer when they send the first OPEN address frame for the connection request. When the port resends the OPEN address frame (e.g., after losing arbitration and handling an inbound OPEN address frame), it shall set the ARBITRATION WAIT TIME field to the current value of the arbitration wait timer.

An initiator port or target port may be unfair, setting the ARBITRATION WAIT TIME field to a higher value than its arbitration wait timer indicates. However, unfair ports shall not set the SCALE bit to one; this limits the amount of unfairness and helps prevent livelocks.

Expander ports shall maintain arbitration wait timers. The expander port that receives an OPEN address frame shall set the timer to the value of the incoming ARBITRATION WAIT TIME field and start the timer as it arbitrates for internal access to the outgoing expander port. When the expander device sends the OPEN address frame out another expander port, it shall set the outgoing ARBITRATION WAIT TIME field to the current value of the timer maintained by the incoming expander port.

A port shall stop the timer and set it to zero when it wins arbitration, receiving either OPEN_ACCEPT or OPEN_REJECT. A port shall stop the timer and set it to zero when it loses arbitration to a connection request that satisfies its arbitration request, receiving an OPEN address frame from the destination port with matching PROTOCOL and LINK RATE fields.

When arbitrating for access to an outgoing expander port, the expander device shall select the connection request from the expander port with the largest arbitration wait time value. If the largest arbitration wait times are identical, then the connection request with the largest SOURCE DEVICE NAME shall win arbitration.

If two connection requests pass on a physical link, the winner shall be determined by comparing field values in this order:

1) largest ARBITRATION WAIT TIME field value;

2) largest SOURCE DEVICE NAME field value; or

3) largest LINK RATE field value.

See 7.11.3 for details on the ARBITRATION WAIT TIME field.

7.11.4 Expander devices and connection requests

7.11.4.1 All expander devices

Before an expander device sends AIP, it may have sent an OPEN address frame on the same physical link. Fairness dictates which OPEN address frame will win (see 7.11.3).

After an expander device sends an AIP, it shall not send an OPEN address frame unless it has higher priority than the incoming connection request.

Expander devices shall send no more than 3 AIP primitives (one AIP primitive sequence) consecutively without sending an idle dword. Expander devices shall send AIP primitive sequence every 128 dwords.

Expander devices shall AIP immediately after receiving an OPEN address frame.

7.11.4.2 Edge expander devices

When an edge expander device receives a connection request, it shall compare the destination device name to the device names of the devices to which each of its phys is attached.

If they match, then the expander device shall arbitrate for access to one of the matching physical links and pass along the connection request.

If they do not match, but an expander device is attached and the request did not come from that expander device, the connection request shall be forwarded to the expander device.

If they do not match, and no expander device is attached or the request came from an expander device, the edge expander device shall reply with OPEN_REJECT(NO DESTINATION).

If the output port indicated by the routing table matches the input port, it shall reply with OPEN_REJECT(BAD DESTINATION). When two edge expander devices are attached, this mean requests to non-existent devices

return OPEN_REJECT(BAD DESTINATION) rather than OPEN_REJECT(NO DESTINATION) - when a fanout expander device is involved, an OPEN_REJECT(NO DESTINATION) is sent.

Table 48 shows the routing table maintained by an edge expander device.

Table 48. Edge expander device routing table

Phy	Device name	Expander device attached?
Internal	Edge expander device name	Yes or no
Phy 0	Attached device name	Yes or no
Phy 1	Attached device name	Yes or no
…	…	…
Phy n	Attached device name	Yes or no

7.11.4.3 Fanout expander devices

When a fanout expander device receives a connection request, it compares the destination device name to the device names of the devices to which each of its phys is attached. For each port which is attached to an edge expander, it also compares the device names to which that edge expander device reported being attached.

If it finds a match, it shall arbitrate for access to one of the matching physical links and pass along the connection request.

If it does not find a match, it shall reply with OPEN_REJECT(NO DESTINATION). If the output port indicated by the routing table matches the input port, it shall reply with OPEN_REJECT(BAD DESTINATION).

Table 49 shows the routing table maintained by a fanout expander device.

Table 49. Fanout expander device routing table

Phy	Phy in attached expander device	Device name
Internal		Fanout expander device name
Phy 0	Itself	Fanout expander device attached device name (may be an edge expander device name)
	Phy 0	Edge expander device attached device name
	Phy 1	Edge expander device attached device name
	…	Edge expander device attached device name
	Phy 63	Edge expander device attached device name
Phy 1	Itself	Fanout expander device attached device name (may be an edge expander device name)
	Phy 0	Edge expander device attached device name
	Phy 1	Edge expander device attached device name
	…	Edge expander device attached device name
	Phy 63	Edge expander device attached device name
…		
Phy n	Itself	Fanout expander device attached device name (may be an edge expander device name)
	Phy 0	Edge expander device attached device name
	Phy 1	Edge expander device attached device name
	…	Edge expander device attached device name
	Phy 63	Edge expander device attached device name

7.11.5 Abandoning a connection request

The BREAK primitive sequence may be used to abandon a connection request. The source port shall send a BREAK primitive sequence after the open timer expires or if it chooses to abandon its request for any other reason.

After sending BREAK, the source port shall initialize a break timeout timer to 1 millisecond and start the timer. If the timer expires before a break response is received, the source port may assume the physical link is unusable.

Table 50 lists the possible responses to a BREAK sent before a connection response has been received.

Table 50. Abandon connection request responses

Response	Description
BREAK	This confirms that the open connection request has been abandoned.
Open response listed in 7.11.2	The BREAK was too late and an open response arrived late. The originator shall honor this as a response to the open request it was attempting to abandon.
No response; timer expires	The originating port shall assume the connection request has been abandoned.

The BREAK response is generated by the expander, not the target. If the expander device has sent a connection request to the destination, it shall also send BREAK to the destination. If the expander device has not sent a connection request to the destination, it shall not send BREAK to the destination. The expander device shall send BREAK back to the originating port after it has ensured that an open response will not occur. Figure 50 shows BREAK usage.

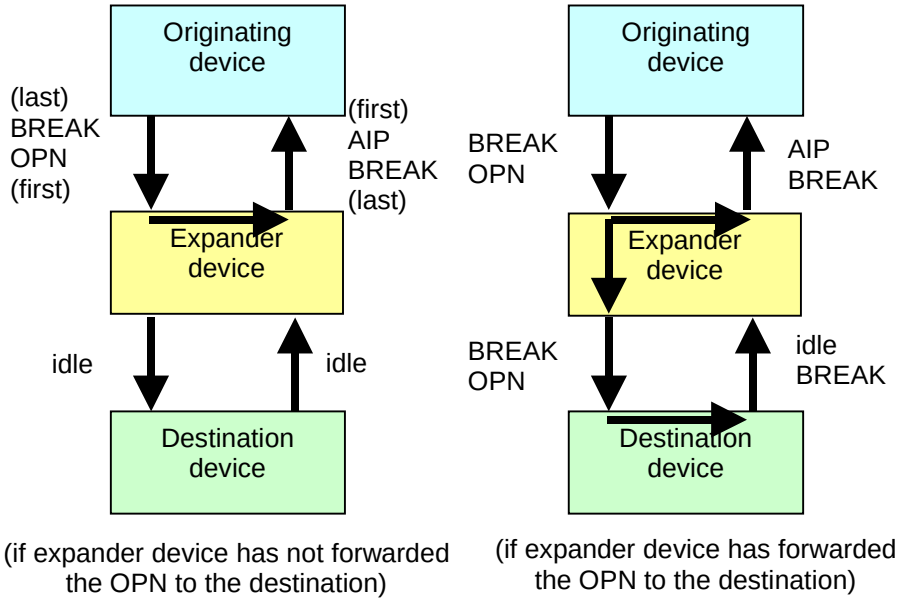

Figure 50. BREAK usage

Figure 51 illustrates the sequence for a connection request that times out.

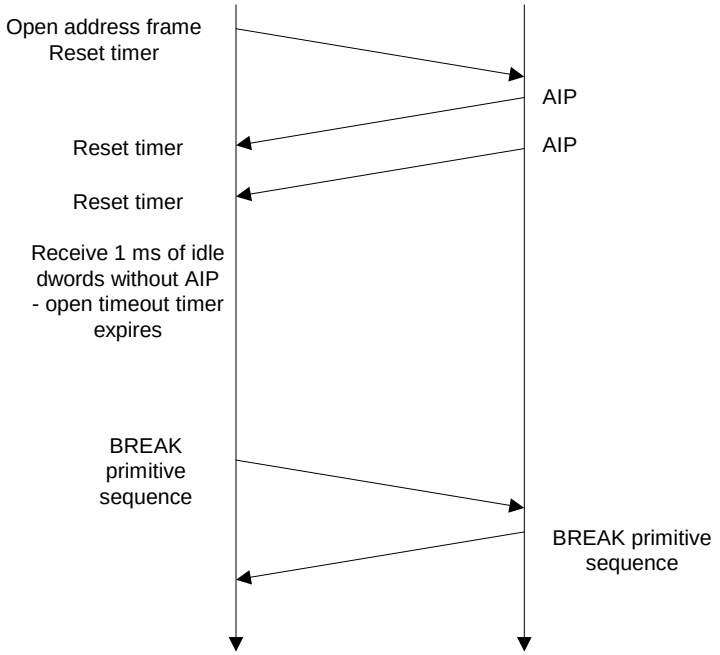

Figure 51. Connection request timeout example

If a source port opens a connection but gets back the wrong protocol response (i.e., source port sends a connection request with one protocol but receives an OPEN_ACCEPT indicating another protocol), the source port shall break the connection with BREAK rather than close it with CLOSE.

If a port detects a timeout while closing a connection (e.g., while exchanging DONE primitives in SSP, or while exchanging CLOSE primitives in any protocol) it may send BREAK to break the connection.

7.11.6 Breaking an open connection

A BREAK primitive sequence may also be used to break an open connection, in cases where CLOSE is not available. After sending BREAK, the originating port shall ignore all incoming data except for BREAK primitives.

Table 51 lists the possible responses to a BREAK sent after a connection has been established.

Table 51. Break connection responses

Response	Description
BREAK	This confirms that the open connection has been broken.
No response; timer expires	The originating port shall assume the connection has been broken.

7.11.7 Closing a connection

The CLOSE primitive is used to close a connection of any protocol. The originating port for the close may have been either the source or destination port when the connection was opened.

When an SSP device has both sent and received DONE, it shall send a CLOSE primitive sequence. When an STP initiator port or an expander device decides to close an STP connection, it shall send a CLOSE primitive sequence.

No additional primitives for the connection shall follow the CLOSE. Expander devices shall close the full-duplex connection upon seeing a CLOSE primitive sequence in each direction.

When a port has both sent and received CLOSE, it shall consider the connection closed.

Figure 52 illustrates the sequence for a closing an SSP connection.

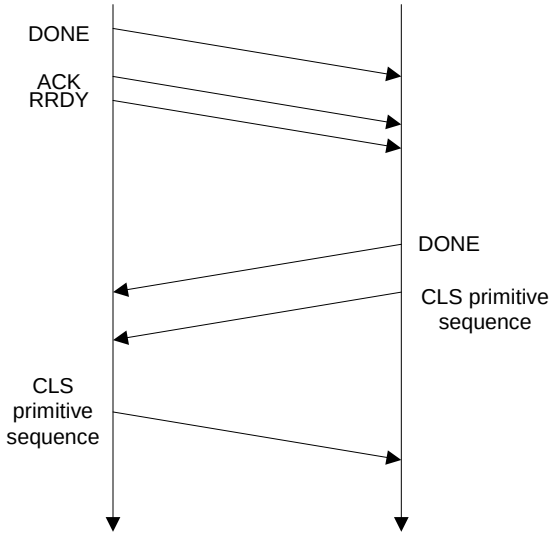

Figure 52. Closing an SSP connection example

7.11.8 Connection management state diagrams

7.11.8.1 Endpoint connection management

7.11.8.1.1 SAS link layer states overview

A SAS link layer state connection management machine (SL state machine) is an entity the resides on each SAS phy that is contained within a SAS port that controls the flow of dwords on a link. The SL state machine contains the following states:

a) SL0:Idle state;
b) SL1:Arb_Sel state;
c) SL2:Selected state;
d) SL4:Connected state;
e) SL5:Disconnect_wait state;
f) SL6:Break_wait state; and
g) SL7:Break state.

Each SAS phy shall transmit ALIGNs, as necessary, regardless of the state.

The SL state machine shall be activated after completion of the phy reset sequence. The idle state of the SL state machine shall be entered after completion of the identification sequence.

[Editor's note: need to run some identification sequence states before entering SL0:Idle]

Inputs to the SL state machine include:

a) port layer requests (e.g., OPEN address frame);
b) primitives and dwords received from the SAS link receiver state machine; and
c) parameters from other state machines.

Outputs from the SL state machine include:

a) confirmations to the port layer:
b) primitives and dwords transmitted from the SAS link transmitter state machine; and
c) parameters to other state machines.

Unless otherwise stated within the state description, all disparity errors, illegal characters, and unexpected primitives (i.e., any primitive not described in the description of the SL state) received within any SL state shall be ignored.

Any detection of an internal link error shall cause the SL state machine to transition to the break_wait state (i.e., SLx:SL6 transition).

Figure 53 describes the SAS link endpoint connection management state machine.

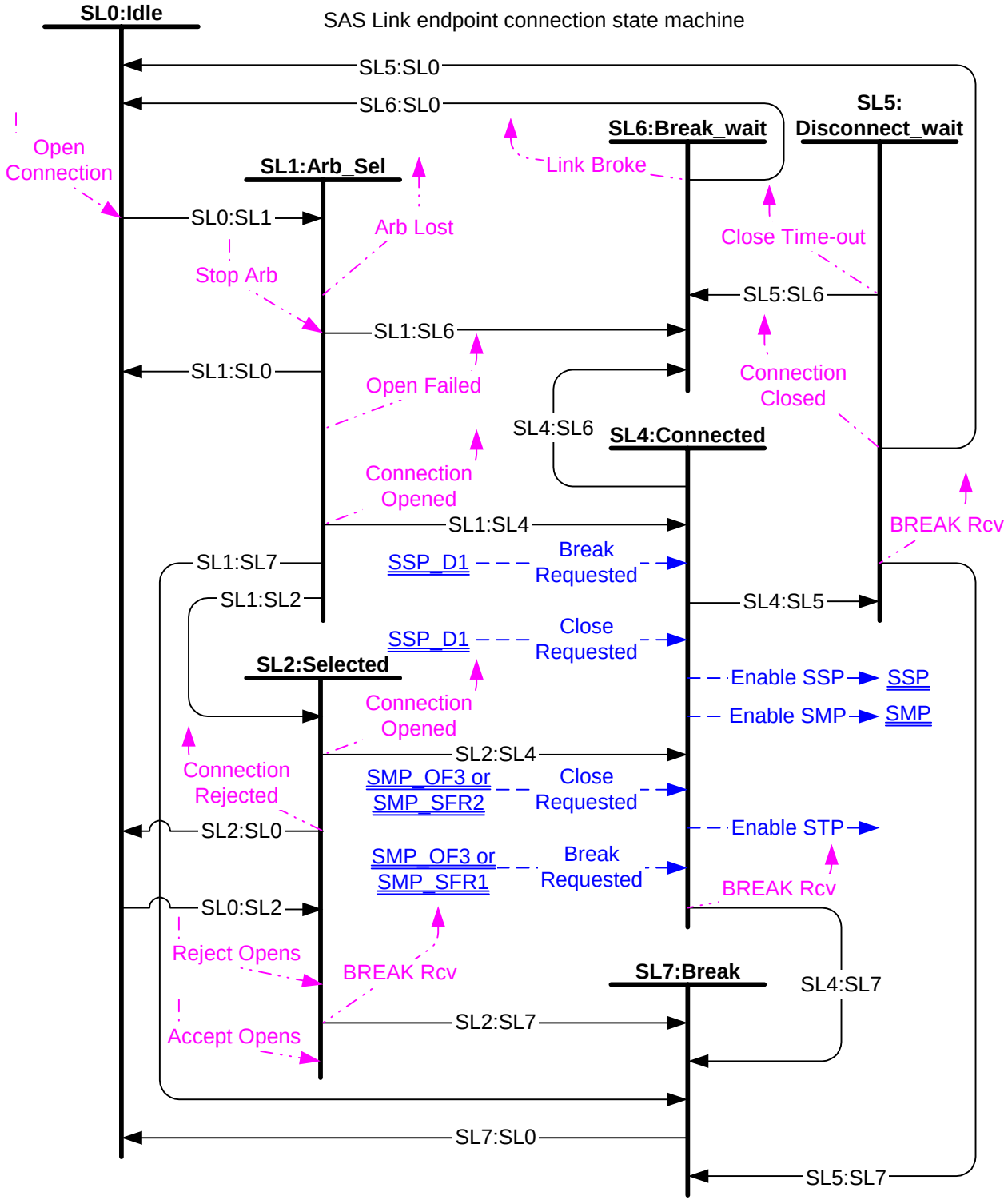

Figure 53. SAS link endpoint connection management state diagram

7.11.8.1.2 SL0:Idle state

7.11.8.1.2.1 SL0:Idle state description

The SL0:Idle state is the initial state and is the state that is used when there is no active connection.

The SL0:Idle state shall be exited as the result of:

a) an open connection request from the port layer; or
b) upon receiving an OPEN address frame.

While in the SL0:Idle state the SAS phy shall transmit idle dwords.

7.11.8.1.2.2 Transition SL0:SL1 (Idle:Arb_sel)

The SL0:SL1 transition shall occur after the port layer requests an open connection.

This transition passes to the SL1:Arb_sel state:

a) the destination port identifier of the SAS port to be opened;
b) the protocol to be used;
c) the AWT to be used; and
d) the link rate to be used.

7.11.8.1.2.3 Transition SL0:SL2 (Idle:Selected)

The SL0:SL2 transition shall occur after a valid OPEN address frame (i.e., no CRC error, the PROTOCOL field in the OPEN address frame indicates a supported protocol, or the DESTINATION PORT IDENTIFIER field in the OPEN address frame indicates the identifier of SAS port that received the OPEN address frame) is received. This transition passes the protocol, destination port identifier, and link rate of the SAS port requesting the open to the selected state.

7.11.8.1.3 SL1:Arb_sel state

7.11.8.1.3.1 SL1:Arb_sel state description

The SL1:Arb_sel state is used to arbitrate for a connection and to specify the destination of the connection.

The SL1:Arb_sel state shall transmit an OPEN address frame:

a) with the DESTINATION PORT IDENTIFIER field set to the identifier of the SAS port to be opened;
b) the protocol to be used;
c) the link rate to be used; and
d) with ARBITRATION WAIT TIME field set to the value of the AWT.

After transmitting the OPEN address frame, the state machine shall initialize an open time-out timer to 1 millisecond and start the open time-out timer.

If an AIP is received the open time-out timer shall be set to its initial value. The SL1_Arb_sel state shall not enforce a limit on the number of AIPs received.

The SL1_Arb_sel state shall implement fairness rules (see 7.11.3) by using the value of the AWT passed by the SL0_Idle:SL1_Arb_sel transition.

The SL1_Arb_sel state shall be exited upon receiving:

a) an OPEN_REJECT;
b) a BREAK primitive sequence;
c) an OPEN_ACCEPT;
d) OPEN address frame if the fairness rules indicate the received OPEN address frame overrides the transmitted OPEN address frame; or
e) after the open time-out timer is exceeded.

The SL1_Arb_sel state shall not be exited if:

a) no AIPs have been received;
b) an OPEN address frame is received; and
c) fairness rules indicate the received OPEN address frame shall be ignored.

7.11.8.1.3.2 Transition SL1:SL0 (Arb_sel:Idle)

The SL1:SL0 transition shall occur after an OPEN_REJECT is received.

If an OPEN_REJECT(WRONG DESTINATION) is received the port layer shall be notified of the current value of the AWT and that arbitration was lost with a reason code of OPEN FAILED - WRONG DESTINATION before making the SL1:SL0 transition.

If an OPEN_REJECT(INVALID LINK RATE) is received the port layer shall be sent a confirmation using the open failed parameter that arbitration was lost with a reason code of OPEN FAILED - INVALID LINK RATE before making the SL1:SL0 transition.

If an OPEN_REJECT(INVALID PROTOCOL TYPE) is received the port layer shall be sent a confirmation using the open failed parameter that arbitration was lost with a reason code of OPEN FAILED - INVALID PROTOCOL TYPE before making the SL1:SL0 transition.

If an OPEN_REJECT(RETRY) is received the port layer shall be sent a confirmation using the open failed parameter that arbitration was lost with a reason code of OPEN FAILED - RETRY before making the SL1:SL0 transition.

If an OPEN_REJECT(PATHWAY BLOCKED) is received the port layer shall be sent a confirmation using the open failed parameter that arbitration was lost with a reason code of OPEN FAILED - PATHWAY BLOCKED before making the SL1:SL0 transition.

7.11.8.1.3.3 Transition SL1:SL2 (Arb_sel:Selected)

If one or more AIPs have been received, the SL1:SL2 transition shall occur after a valid OPEN address frame is received.

The OPEN address frame shall override the request that resulted from the SL0:SL1 transition. The SL1:SL2 transition passes destination port identifier of the SAS port requesting to be opened to the selected state.

If no AIPs have been received, the SL1:SL2 transition shall occur after:

a) a valid OPEN address frame is received; and
b) fairness rules indicate the received OPEN address frame shall be used.

This transition passes the protocol, destination port identifier, and link rate of the SAS port requesting to be opened to the selected state.

7.11.8.1.3.4 Transition SL1:SL4 (Arb_sel:Connected)

The SL1:SL4 transition shall occur if an OPEN_ACCEPT is received.

If the PROTOCOL field in the transmitted OPEN address frame indicated an STP link is to be opened then send a confirmation to the port layer using the connection opened indication that an STP link has been opened before the SL1:SL4 transition.

The SL1:SL4 transition shall pass a parameter indicating an STP link has been opened to the connected state. At this point an STP link has been opened between the source SATA port and the destination SATA port.

If the PROTOCOL field in the transmitted OPEN address frame indicated an SSP link is to be opened then send a confirmation to the port layer using the connection opened indication that an SSP link has been opened before the SL1:SL4 transition.

The SL1:SL4 transition shall pass a parameter indicating an SSP link has been opened to the connected state and the hashed value of the source port identifier and the destination port identifier. At this point an SSP link has been opened between the source SCSI port and the destination SCSI port.

If the PROTOCOL field in the transmitted OPEN address frame indicated an SMP link is to be opened then send a confirmation to the port layer using the connection opened indication that an SMP link has been opened before the SL1:SL4 transition.

The SL1:SL4 transition shall pass a parameter indicating an SMP link has been opened to the connected state and the hashed value of the source port identifier and the destination port identifier. At this point an SMP link has been opened between the source SAS port and the destination SAS port.

7.11.8.1.3.5 Transition SL1:SL6 (Arb_sel:Break_wait)

The SL1:SL6 transition shall occur after:

a) the port layer requests using the stop arb parameter the arbitration be stopped and a confirmation sent to the port layer using the arb lost parameter that the arbitration failed with a reason code of ARBITRATION LOST - PORT LAYER REQUEST; or
b) there is no response to the open connection request within the open time-out and a confirmation sent to the port layer using the arb lost parameter that the arbitration failed with a reason code of ARBITRATION LOST - OPEN TIME-OUT OCCCURED.

7.11.8.1.3.6 Transition SL1:SL7 (Arb_sel:Break)

The SL1:SL7 transition shall occur after:

a) a BREAK primitive sequence is received; and

b) the arb_sel state has confirmed to the port layer using the arb lost parameter that the arbitration failed with a reason code of ARBITRATION LOST - BREAK RECEIVED.

7.11.8.1.4 SL2:Selected state

7.11.8.1.4.1 SL2:Selected state description

The selected state completes the establishment of an SSP, SMP, or STP link.

The selected state shall be exited after:

a) transmitting an OPEN_ACCEPT;
b) transmitting an OPEN_REJECT;
c) receiving a BREAK primitive sequence;
d) detecting an unsupported protocol;
e) detecting an unsupported link rate; or
f) detecting an incorrect destination port identifier.

While in the selected state, the SAS port accepts opening a connection between it and the destination SAS port by transmitting an OPEN_ACCEPT.

The selected state may reject a connection request by transmitting an OPEN_REJECT.

7.11.8.1.4.2 Transition SL2:SL0 (Selected:Idle)

The SL2:SL0 transition shall occur if:

a) the last port layer request was a reject open request, after transmitting an OPEN_REJECT(RETRY), and after a confirmation is sent to the port layer using the connection rejected parameter with a reason code CONNECTION REJECTED - RETRY;
b) the requested protocol is not supported, after transmitting an OPEN_REJECT(INVALID PROTOCOL TYPE), and after a confirmation is sent to the port layer using the connection rejected parameter with a reason code CONNECTION REJECTED - INVALID PROTOCOL TYPE;
c) the requested destination port identifier does not match the identifier of this SAS port, after transmitting an OPEN_REJECT(WRONG DESTINATION), and after a confirmation is sent to the port layer using the connection rejected parameter with a reason code CONNECTION REJECTED - WRONG DESTINATION; or
d) the requested link rate is not supported, after transmitting an OPEN_REJECT(INVALID LINK RATE), and after a confirmation is sent to the port layer using the connection rejected parameter with a reason code CONNECTION REJECTED - INVALID LINK RATE.

7.11.8.1.4.3 Transition SL2:SL4 (Selected:Connected)

If the requested protocol is SSP and the current port layer request is an accept opens then the SL2:SL4 transition shall occur after the SCSI port being opened:

a) transmits an OPEN_ACCEPT; and
b) sends a confirmation to the port layer using the connection opened parameter that an SSP link has been opened.

The SL2:SL4 transition shall pass a parameter indicating an SSP link has been opened to the connected state and the hashed value of the source port identifier and the destination port identifier. At this point an SSP link has been opened between source SCSI port and the destination SCSI port.

If the requested protocol is SMP and the current port layer request is an accept opens then the SL2:SL4 transition shall occur after the SAS port being opened:

a) transmits an OPEN_ACCEPT; and
b) sends a confirmation to the port layer using the connection opened parameter that an SMP link has been opened.

The SL2:SL4 transition shall pass a parameter indicating an SMP link has been opened to the connected state and the hashed value of the source port identifier and the destination port identifier. At this point an SMP link has been opened between source SAS port and the destination SAS port.

If the requested protocol is STP and the current port layer request is an accept opens request then the SL2:SL4 transition shall occur after the SATA port being opened:

a) transmits an OPEN_ACCEPT; and
b) sends a confirmation to the port layer using the connection opened parameter that an STP link has been opened.

The SL2:SL4 transition shall pass a parameter indicating an STP link has been opened to the connected state. At this point an STP link has been opened between source SATA port and the destination SATA port.

7.11.8.1.4.4 Transition SL2:SL7 (Selected:Break)

The SL2:SL7 transition shall occur after a BREAK primitive sequence is received and after the arb_sel state has sent a confirmation to the port layer using the BREAK rcv parameter that the connection was terminated with a reason code of CONNECTION CLOSED - BREAK RECEIVED.

7.11.8.1.5 SL4:Connected state

7.11.8.1.5.1 SL4:Connected state description

The connected state enables a set of connection state machines than control the transfer of data between SAS devices.

If there was a parameter in the transition from the selected state or arb_sel state that indicates an SMP link has been opened then the connected state shall send an enable SMP parameter and the hashed value of the source port identifier and the destination port identifier to the SMP state machines (see 9.4.5).

If there was a parameter in the transition from the selected state or arb_sel state that indicates an SSP link has been opened then the connected state shall send an enable SSP parameter and the hashed value of the source port identifier and the destination port identifier to the SSP state machines (see 9.2.5).

If there was a parameter in the transition from the selected state or arb_sel state that indicates an STP link has been opened then the connected state shall send an enable STP parameter to the STP state machines.

The connected state shall be exited after receiving a break requested parameter, close requested parameter, or the receipt a BREAK primitive sequence.

7.11.8.1.5.2 Transition SL4:SL5 (Connected:Disconnect_wait)

The SL4:SL5 transition shall occur after receipt of a close requested parameter.

For an SSP link the close requested parameter is received from the DONE_wait state of the SSP DONE control state machine.

For an SMP link the close requested parameter is received from the wait_transmit_frame state of the SMP frame response state machine or the rcv_response_frame state of the originate SMP frame state machine.

For an STP link the close requested parameter is received from the STP link layer state machine.

7.11.8.1.5.3 Transition SL4:SL6 (Connected:Break_wait)

The SL4:SL6 transition shall occur after receipt of a break requested parameter.

For an SSP link the break requested parameter is received from the DONE_wait state of the DONE control state machine.

For an SMP link the break requested parameter is received from the wait_originate_frame state of the SMP frame response state machine or the rcv_response_frame state of the originate SMP frame state machine.

For an STP link the break requested parameter is received from the STP link layer state machine.

7.11.8.1.5.4 Transition SL4:SL7 (Connected:Break)

The SL4:SL7 transition shall occur after a BREAK primitive sequence is received and after the connected state has the port layer using the BREAK rcv parameter that the connection was terminated with a reason code of CONNECTION CLOSED - BREAK RECEIVED.

7.11.8.1.6 SL5:Disconnect_wait state

7.11.8.1.6.1 SL5:Disconnect_wait state description

The disconnect_wait state closes the connection and releases all resources associated with this connection.

The disconnect_wait state shall be exited after a CLOSE primitive sequence is received or a BREAK primitive sequence is received or after the close time-out timer is exceeded. The CLOSE primitive sequence may be received at any time while in the disconnect_wait state.

The disconnect_wait state shall transmit a CLOSE primitive sequence. After transmitting the CLOSE primitive sequence the disconnect_wait state machine shall transmit at least three idle dwords and shall initialize a close time-out timer to 1 millisecond and start the timer.

7.11.8.1.6.2 Transition SL5:SL0 (Disconnect_wait:Idle)

The SL5:SL0 transition shall occur after:

a) transmitting a CLOSE primitive sequence;
b) receiving a CLOSE primitive sequence; and
c) the disconnect_wait state has sent a confirmation to the port layer using the close time-out parameter that the close succeeded with a reason code of CONNECTION CLOSED - NO ERRORS OCCURED.

7.11.8.1.6.3 Transition SL5:SL6 (Disconnect_wait:Break_wait)

The SL5:SL6 transition shall occur when there is no response to a CLOSE primitive sequence transmitted within a close time-out and after the disconnect_wait state has sent a confirmation to the port layer using the close time-out parameter that a close failed with a reason code of CONNECTION CLOSED - CLOSE TIME-OUT OCCURED.

7.11.8.1.6.4 Transition SL5:SL7 (Disconnect_wait:Break)

The SL5:SL7 transition shall occur after a BREAK primitive sequence are received and after the disconnect_wait state has sent a confirmation to the port layer using the BREAK rcv parameter that the connection was terminated with a reason code of CONNECTION CLOSED - BREAK RECEIVED.

7.11.8.1.7 SL6:Break_wait state

7.11.8.1.7.1 SL6:Break_wait state description

The break_wait state closes any connection and releases all resources associated with this connection.

The break_wait state shall be exited after a BREAK primitive sequence is received or a break time-out occurs.The BREAK primitive sequence may be received at any time while in the break_wait state.

The break_wait state shall transmit a BREAK primitive sequence. After transmitting a BREAK primitive sequence the break_wait state machine shall transmit at least three idle dwords and shall initialize a break time-out timer to 1 millisecond and start the timer.

While in the break_wait state all primitives received except BREAK and ALIGN shall be ignored.

7.11.8.1.7.2 Transition SL6:SL0 (Break_wait:Idle)

The SL6:SL0 transition shall occur after receiving a BREAK primitive sequence or if the break time-out is exceeded. If a BREAK primitive sequence is not received before the timer is exceeded, the break state shall send a confirmation to the port layer using the link broke parameter that the link is unusable with a reason code of CONNECTION CLOSED - LINK BROKE and then transition to the idle state.

7.11.8.1.8 SL7:Break state

7.11.8.1.8.1 SL7:Break state description

The SL7:Break state closes any connection and releases all resources associated with this connection.

The SL7:Break state shall be exited after transmitting a BREAK primitive sequence.

While in the SL7:Break state all primitives received except ALIGN shall be ignored.

7.11.8.1.8.2 Transition SL7:SL0 (Break:Idle)

The SL7:SL0 transition shall occur after transmitting a BREAK primitive sequence.

7.11.8.2 Expander port

7.11.8.2.1 SAS Expander Link state overview

Each link within an Expander port contains a Expander Link (XL) state machine to control the flow of dwords on the link and to establish and maintain connections with another Expander Link state machine as facilitated by the Expander Function.

The Idle state shall be enabled as result of the completion of phy initialization and discovery.

Unless otherwise stated within a state description, all disparity errors, illegal characters, and unexpected primitives received within any XL state shall be ignored.

The following Request expander primitives shall always be passed to the Expander Function, independent of the Expander Link (XL) state:

a) Request.Aip - AIP received from SAS phy;
b) Request.Break - BREAK primitive sequence received from SAS phy;
c) Request.Change - CHANGE primitive sequence received from SAS phy;
d) Request.Close - CLS primitive sequence received from SAS phy; and
e) Request.PhyNotReady - SAS phy not ready.

The following Request expander primitives shall be passed to the Expander Function only designated by specific Expander Link (XL) states:

a) Request.Path(attributes);
b) Request.Open(attributes); and
c) Request.StreamOut .

The following Response expander primitives shall be passed to the Expander Function only as designated by specific Expander Link (XL) states:

a) Response.Open.Accept;
b) Response.Open.Reject(reason);
c) Response.Backoff.FwdOpen; and
d) Response.Backoff.Retry.

7.11.8.2.2 SAS Expander Link state diagram

Figure 54 defines the main portion of the expander link layer connection state machine.

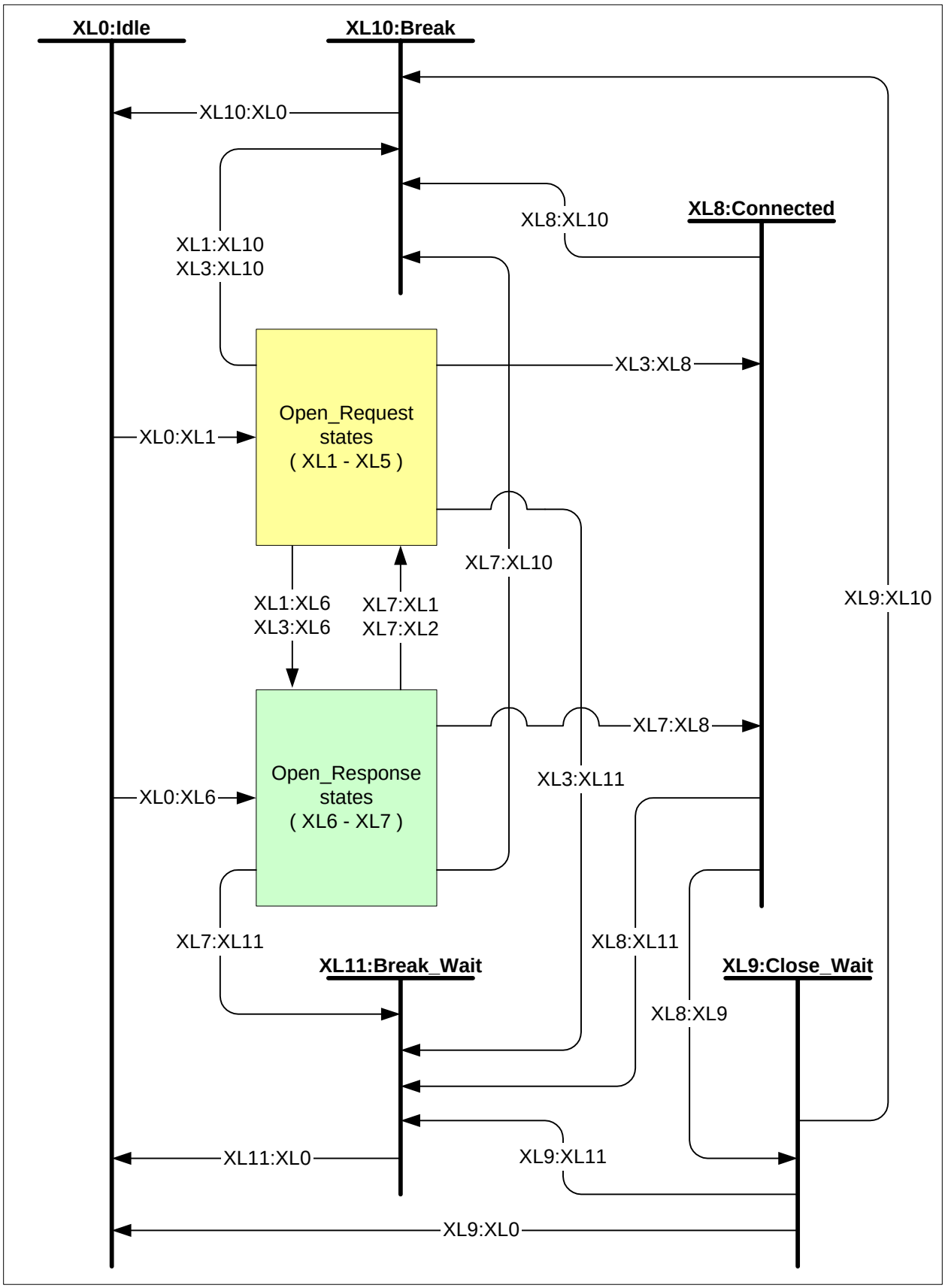

Figure 54. SAS Expander Connection state diagram

Figure 55 defines the Open_Request states of the expander link layer connection state machine.

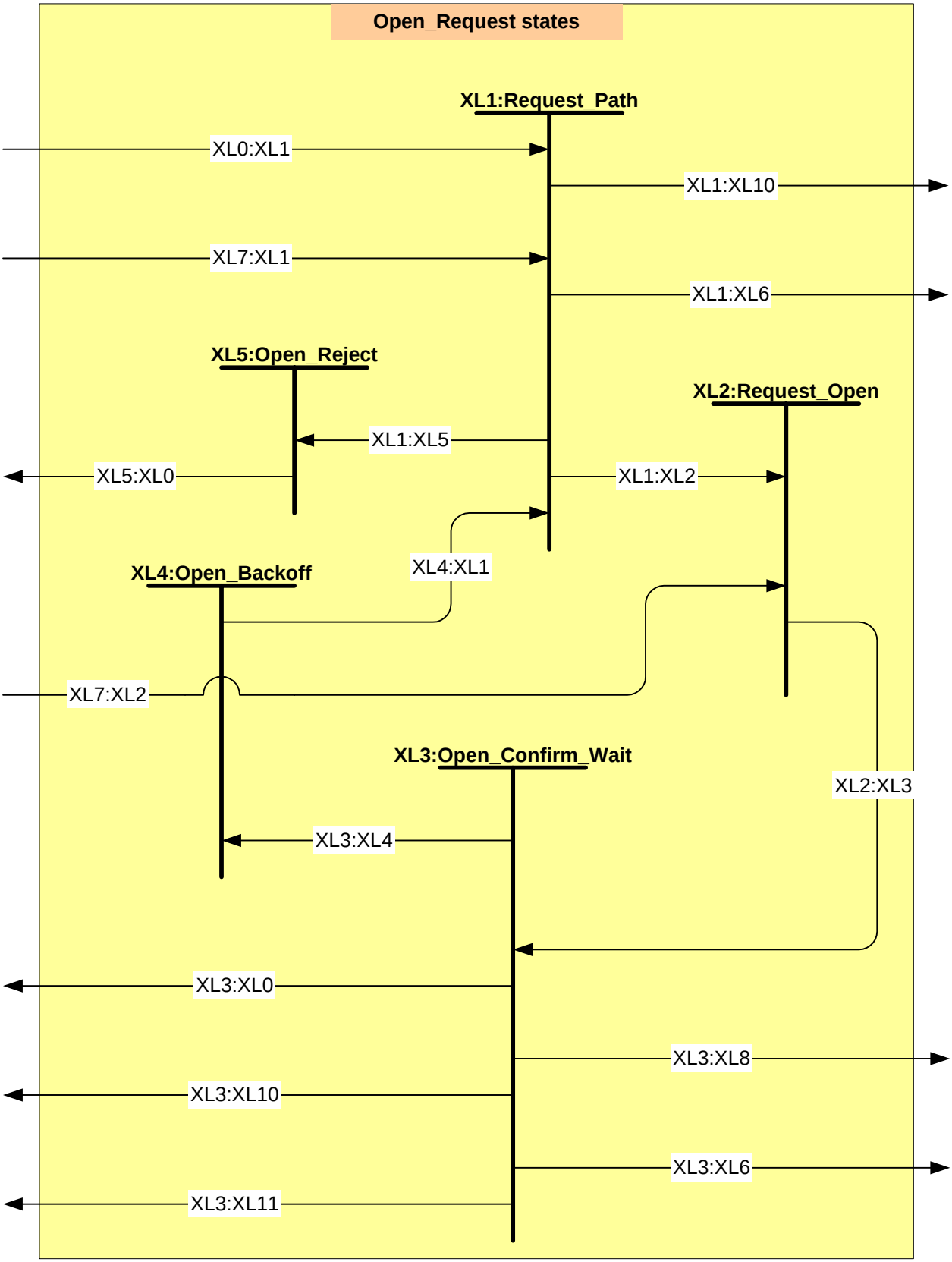

Figure 55. SAS Expander Connection Open_Request states

Figure 56 defines the Open_Response states of the expander link layer connection state machine.

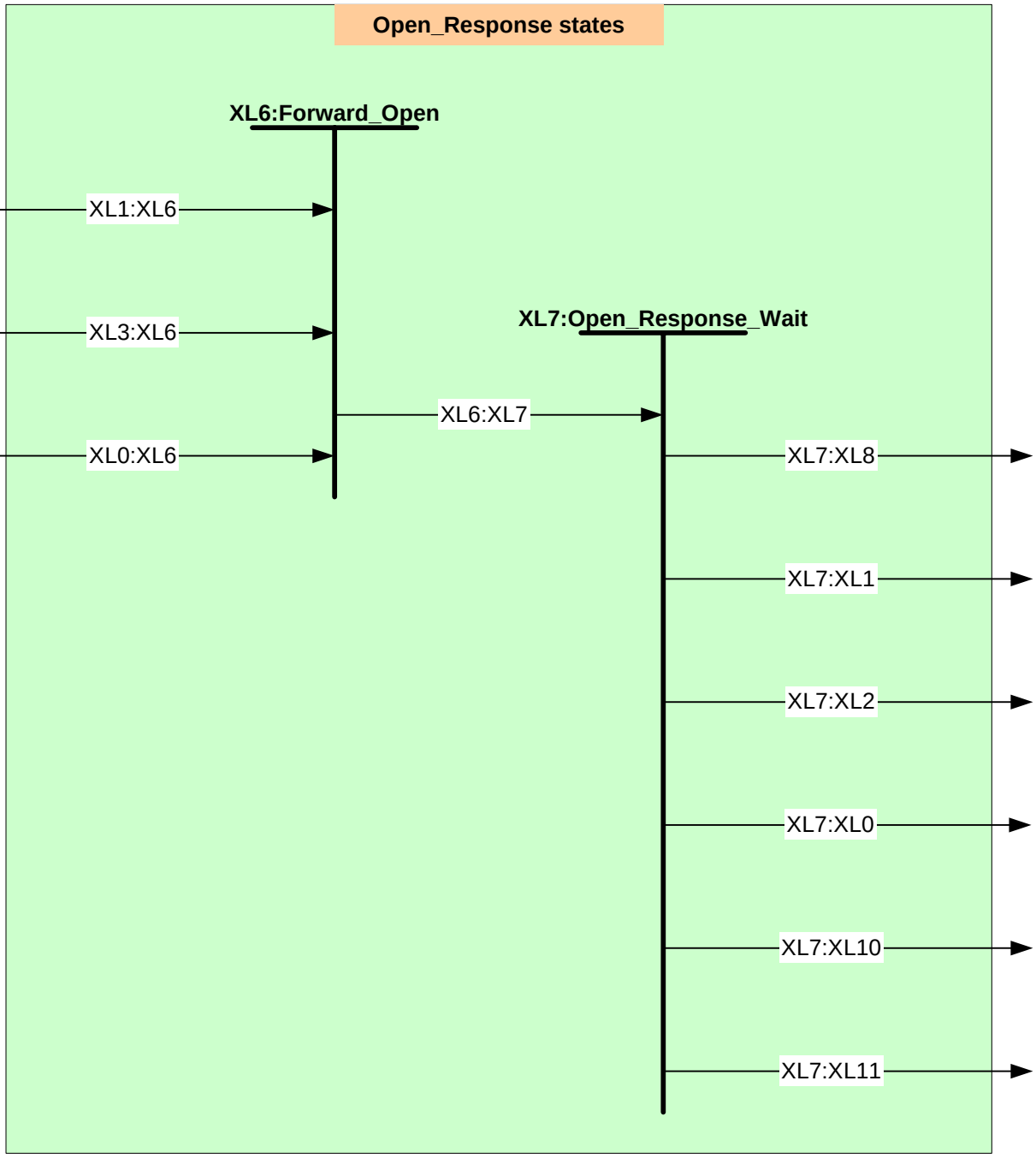

Figure 56. SAS Expander Connection Open_Response states

7.11.8.2.3 XL0:Idle state

7.11.8.2.3.1 XL0:Idle state description

The Idle state is the initial state and the state that occurs when there is no pending or active connection.

The Idle state shall transmit idle dwords.

7.11.8.2.3.2 Transition XL0:XL1 (Idle:Request_Path)

The XL0:XL1 transition shall occur when a valid OPEN address frame is received from the SAS phy.

7.11.8.2.3.3 Transition XL0:XL6 (Idle:Forward_Open)

The XL0:XL6 transition shall occur when a valid OPEN address frame is indicated by the Expander Function via the Indicate.Open expander primitive.

7.11.8.2.3.4 Transition XL0:XL10 (Idle:Break)

The XL0:XL10 transition shall occur when a BREAK primitive sequence is received from the SAS phy.

7.11.8.2.4 XL1:Request_Path state

7.11.8.2.4.1 XL1:Request_Path state description

The Request_Path state is used to specify and arbitrate for a link within a destination port via the Request.Path expander primitive to the Expander Function.

The Request_Path state shall transmit AIP to the SAS phy.

The Request_Path state shall pass the Request.Path(attributes) expander primitive to the Expander Function. The Request.Path attributes include destination device name, source device name, Arbitration Wait Timer value, link rate, and protocol.

7.11.8.2.4.2 Transition XL1:XL2 (Request_Path:Request_Open)

The XL1:XL2 transition shall occur when path arbitration won has been confirmed by the Expander Function via the Confirm.Path.ArbWon expander primitive.

7.11.8.2.4.3 Transition XL1:XL5 (Request_Path:Open_Reject)

The XL1:XL5 transition shall occur when path arbitration has been rejected by the Expander Function via the Confirm.Path.ArbReject expander primitive.

The Expander Function shall reject path arbitration when the specified destination port is invalid or non-operational or when necessary to avoid deadlock.

7.11.8.2.4.4 Transition XL1: XL6 (Request_Path:Forward_Open)

The XL1:XL6 transition shall occur when path arbitration lost has been confirmed by the Expander Function via the Confirm.Path.ArbLost expander primitive.

7.11.8.2.4.5 Transition XL1:XL10 (Request_Path:Break)

The XL1:XL10 transition shall occur when a BREAK primitive sequence is received from the SAS phy.

7.11.8.2.5 XL2:Request_Open state

7.11.8.2.5.1 XL2:Request_Open state description

The Request_Open state is used to request that an OPEN address frame be sent on the destination port as specified by the Request.Open(attributes) expander primitive.

The Request_Open state shall transmit AIP to the SAS phy.

The Request_Open state shall pass the Request.Open(attributes) expander primitive to the Expander Function. The Request.Open attributes include destination device name, source device name, Arbitration Wait Timer value, link rate, and protocol.

7.11.8.2.5.2 Transition XL2:XL3 (Request_Open:Open_Confirm_Wait)

The XL2:XL3 transition shall occur unconditionally.

7.11.8.2.6 XL3:Open_Confirm_Wait state

7.11.8.2.6.1 XL3:Open_Confirm_Wait state description

The Open_Confirm_Wait state waits for confirmation to an OPEN address frame sent on a destination port.

During the Open_Confirm_Wait state, dword transmission to the SAS phy shall occur as follows:

AIP when the Expander Function indicates reception of AIP via the Ind.Aip expander primitive

OPEN_ACCEPT when the Expander Function confirms reception of OPEN_ACCEPT via the Cnf.Open.Accept expander primitive.

OPEN_REJECT when the Expander Function confirms reception of OPEN_REJECT via the Cnf.Open.Reject expander primitive.

idle dwords when none of the previous conditions are present.

7.11.8.2.6.2 Transition XL3:XL0 (Open_Confirm_Wait:Idle)

The XL3:XL0 transition shall occur when all of the following conditions are met:

OPEN_REJECT has been confirmed via the Expander Function Confirm.Open.Reject expander primitive –and–

path resources have been released for this connection request

7.11.8.2.6.3 Transition XL3:XL4 (Open_Confirm_Wait:Open_Backoff)

The XL3:XL4 transition shall occur when the Expander Function confirms via the Confirm.Backoff.Retry expander primitive that this link must release path resources and retry the connection in order to avoid deadlock.

7.11.8.2.6.4 Transition XL3:XL6 (Open_Confirm_Wait:Forward_Open)

The XL3:XL6 transition shall occur when the Expander Function confirms via the Confirm.Backoff.FwdOpen expander primitive that this link must forward the other link's received OPEN address frame and switch roles to connection responder in order to avoid deadlock.

7.11.8.2.6.5 Transition XL3:XL8 (Open_Confirm_Wait:Connected)

The XL3:XL8 transition shall occur when OPEN_ACCEPT has been confirmed via the Expander Function Confirm.Open.Accept expander primitive.

7.11.8.2.6.6 Transition XL3:XL10 (Open_Confirm_Wait:Break)

The XL3:XL10 transition shall occur when a BREAK primitive sequence is received from the SAS phy.

7.11.8.2.6.7 Transition XL3:XL11 (Open_Confirm_Wait:Break_Wait)

The XL3:XL11 transition shall occur when a BREAK primitive sequence is indicated by the Expander Function via the Indicate.Break expander primitive.

7.11.8.2.7 XL4:Open_Backoff state

7.11.8.2.7.1 XL4:Open_Backoff state description

The Open_Backoff state is used to release path resources in order to avoid deadlock.

The Open_Backoff state shall transmit AIP to the SAS phy.

7.11.8.2.7.2 Transition XL4:XL1 (Open_Backoff:Request_Path)

The XL4:XL1 transition shall occur after path resources have been released for this connection request.

7.11.8.2.8 XL5:Open_Reject state

7.11.8.2.8.1 XL5:Open_Reject state description

The Open_Reject state is used to reject a connection request.

The Open_Reject state shall transmit OPEN_REJECT(reason) to the SAS phy using the reason code supplied via the Cnf.Open.Reject(reason) expander primitive.

7.11.8.2.8.2 Transition XL5:XL0 (Open_Reject:Idle)

The XL5:XL0 transition shall occur when all of the following conditions are met:

OPEN_REJECT has been transmitted –and–

path resources have been released for this connection request

7.11.8.2.9 XL6:Forward_Open state

7.11.8.2.9.1 XL6:Forward_Open state description

The Forward_Open state is used to forward an OPEN address frame as specified by the Expander Function Indicate.Open expander primitive.

The Forward_Open state shall transmit an OPEN address frame.

7.11.8.2.9.2 Transition XL6:XL7 (Forward_Open:Open_Response_Wait)

The XL6:XL7 transition shall occur after the OPEN address frame has been transmitted.

7.11.8.2.10 XL7:Open_Response_Wait state

7.11.8.2.10.1 XL7:Open_Response_Wait state description

The Open_Response_Wait state waits for a response to an OPEN address frame sent by this SAS phy and determines the appropriate action to take based on the response.

The Open_Response_Wait state shall transmit idle dwords.

The Open_Response_Wait state shall pass the Response.Open.Accept expander primitive to the Expander Function when an OPEN_ACCEPT is received by the SAS phy.

The Open_Response_Wait state shall pass the Response.Open.Reject(reason) expander primitive to the Expander Function when an OPEN_REJECT(reason) is received by the SAS phy.

The Open_Response_Wait state shall pass the Response.Backoff.FwdOpen expander primitive to the Expander Function when any of the following conditions are met:

AIP has not been received and a higher priority OPEN address frame is received which requires path resources already allocated to this pending connection

AIP has been received and any OPEN address frame is received which requires path resources already allocated to this pending connection

The Open_Response_Wait state shall pass the Response.Backoff.Retry expander primitive to the Expander Function when any of the following conditions are met:

AIP has not been received and a higher priority OPEN address frame is received which requires path resources not currently allocated to this pending connection

AIP has been received and any OPEN address frame is received which requires path resources not currently allocated to this pending connection

7.11.8.2.10.2 Transition XL7:XL0 (Open_Response_Wait:Idle)

The XL7:XL0 transition shall occur when OPEN_REJECT is received by the SAS phy.

7.11.8.2.10.3 Transition XL7:XL1 (Open_Response_Wait:Request_Path)

The XL7:XL1 transition causes a role change from connection responder to connection requestor and shall occur when any of the following conditions are met:

AIP has not been received and a higher priority OPEN address frame is received which requires path resources not currently allocated to this pending connection

AIP has been received and any OPEN address frame is received which requires path resources not currently allocated to this pending connection

7.11.8.2.10.4 Transition XL7:XL2 (Open_Response_Wait:Request_Open)

The XL7:XL2 transition causes a role change from connection responder to connection requestor and shall occur when any of the following conditions are met:

AIP has not been received and a higher priority OPEN address frame is received which requires path resources already allocated to this pending connection

AIP has been received and any OPEN address frame is received which requires path resources already allocated to this pending connection

7.11.8.2.10.5 Transition XL7:XL8 (Open_Response_Wait:Connected)

The XL7:XL8 transition shall occur when OPEN_ACCEPT is received by the SAS phy.

7.11.8.2.10.6 Transition XL7:XL10 (Open_Response_Wait:Break)

The XL7:XL10 transition shall occur when a BREAK primitive sequence is received from the SAS phy.

7.11.8.2.10.7 Transition XL7:XL11 (Open_Response_Wait:Break_Wait)

The XL7:XL11 transition shall occur when a BREAK primitive sequence is indicated by the Expander Function via the Indicate.Break expander primitive.

7.11.8.2.11 XL8:Connected state

7.11.8.2.11.1 XL8:Connected state description

The Connected state provides a completed circuit between links of two expander ports.

The Connected state shall transmit all dwords indicated by the Expander Function via the Indicate.StreamIn expander primitive.

7.11.8.2.11.2 Transition XL8:XL9 (Connected:Close_Wait)

The XL8:XL9 transition shall occur when a CLS primitive sequence is indicated by the Expander Function via the Indicate.Close expander primitive.

7.11.8.2.11.3 Transition XL8:XL10 (Connected:Break)

The XL8:XL10 transition shall occur when a BREAK primitive sequence is received from the SAS phy.

7.11.8.2.11.4 Transition XL8:XL11 (Connected:Break_Wait)

The XL8:XL11 transition shall occur when a BREAK primitive sequence is indicated by the Expander Function via the Indicate.Break expander primitive.

7.11.8.2.12 XL9:Close_Wait state

7.11.8.2.12.1 XL9:Close_Wait state description

The Close_Wait state closes a connection and releases any utilized path resources.

The Close_Wait state shall transmit a CLS primitive sequence to the SAS phy.

7.11.8.2.12.2 Transition XL9:XL0 (Close_Wait:Idle)

The XL9:XL0 transition shall occur when all of the following conditions are met:

A CLS primitive sequence has been both transmitted and received by this SAS phy

path resources have been released for this connection

7.11.8.2.12.3 Transition XL9:XL10 (Close_Wait:Break)

The XL9:XL10 transition shall occur when a BREAK primitive sequence is received from the SAS phy.

7.11.8.2.12.4 Transition XL9:XL11 (Close_Wait:Break_Wait)

The XL9:XL10 transition shall occur when a BREAK primitive sequence is indicated by the Expander Function via the Indicate.Break expander primitive.

7.11.8.2.13 XL10:Break state

7.11.8.2.13.1 XL10:Break state description

The Break state closes any connection and releases any utilized path resources.

The Break state shall transmit a BREAK primitive sequence.

7.11.8.2.13.2 Transition XL10:XL0 (Break:Idle)

The XL10:XL0 transition shall occur after transmitting a BREAK primitive sequence.

7.11.8.2.14 XL11:Break_Wait state

7.11.8.2.14.1 XL11:Break_Wait state description

The Break_Wait state closes any connection and releases any utilized path resources.

The Break state shall transmit a BREAK primitive sequence. After transmitting the BREAK primitive sequence the Break_Wait state shall initialize a break time-out timer to 1 millisecond and start the timer.

7.11.8.2.14.2 Transition XL11:XL0 (Break_Wait:Idle)

The XL10:XL0 transition shall occur after a BREAK primitive sequence is received or after the break time-out timer expires, whichever occurs first.

7.11.8.3 Error handling

Expanders that are forwarding dwords from a SAS or SATA physical link to a SAS physical link shall replace each invalid dword with the ERROR primitive.

Expanders that are forwarding dwords to a SATA physical link shall replace an invalid dword or an ERROR primitive with four K28.3 characters. This sequence generates a code violation in SATA.

Expander devices need not check CRCs on frames.

7.12 Rate matching

Each connection request contains the link rate of the source port.

Initiator ports shall use SMP to determine the rate of the target port and the rates of every intermediate expander device-to-expander device physical link and shall not request a rate that is greater than the slowest physical link rate found.

If the rates do not match, the faster phy shall insert ALIGNs between dwords to match the slower rate.

The faster phy shall rotate between ALIGN, ALIGN(~~2~~1), ALIGN(~~3~~2), and ALIGN(~~4~~3) to reduce EMI.

Figure 57 shows an example of rate matching between a 1,5 Gbps source link and a 3,0 Gbps destination link, with one ALIGN inserted between dwords on the faster link.

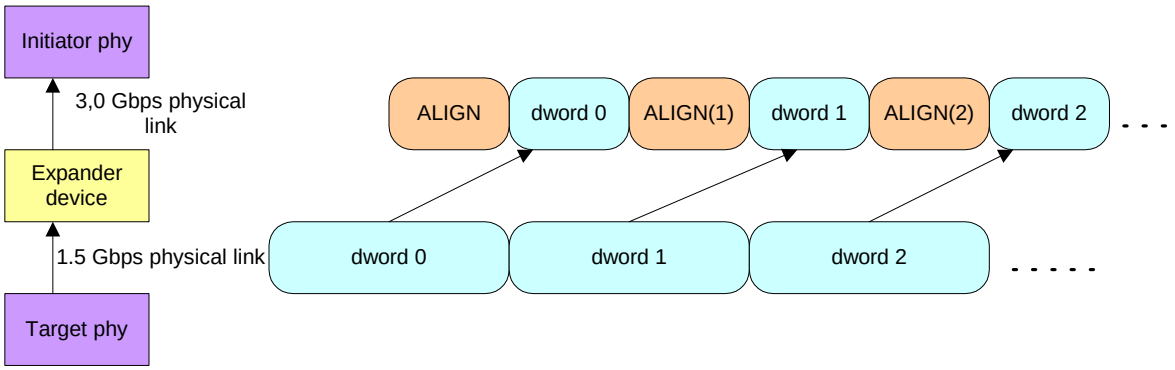

Figure 57. Rate matching example

7.13 SSP link layer

7.13.1 Full duplex

SSP is a full duplex protocol. A port may receive a frame or primitive on a physical link the same time it is sending a frame or primitive on the same physical link.

When a connection is open and a port has no more frames to send, it shall send DONE to start closing the connection. The other direction may still be active, so the DONE may be followed by RRDYs, ACKs, and NAKs.

7.13.2 SSP frame transmission

During an SSP connection, frames are preceded by SOF and followed by EOF as shown in Figure 58.

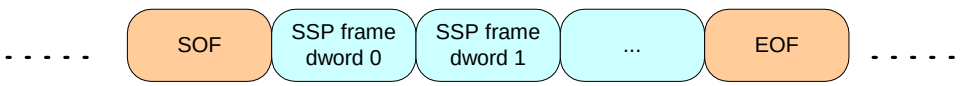

Figure 58. SSP frame transmission

Every frame shall be acknowledged within 1 ms with either a positive acknowledgement (ACK) or a negative acknowledgement (NAK). ACK means the frame was received into a frame buffer without errors. NAK means the frame was received with an error (e.g. a CRC error). There are several versions of NAK indicating the specific cause for the rejection. NAK(CRC ERROR) indicates the frame was received, but the CRC check failed. NAK(GENERAL ERROR) indicates that some other type of error occurred.

Some interlocked frames which are NAKed may be retried; see TBD. Non-interlocked frames which are NAKed shall not be retried. The SCSI command associated with a NAKed frame that is not successfully retried shall be terminated with a CHECK CONDITION status with a sense key of ABORTED COMMAND and an additional sense code of DATA PHASE CRC ERROR DETECTED.

7.13.3 SSP flow control

SSP devices grant credit for permission to send frames with the RRDY primitive. Each RRDY primitive increments credit by one frame. Frame transmission decrements credit by one frame. Credit of zero frames is established at the beginning of each connection.

SSP devices shall not increment credit past 255 frames.

An SSP initiator port or an SSP target/initiator port acting in its initiator role shall never refuse to provide credit by sending RRDY because it needs to send a frame itself. An SSP target port or an SSP target/initiator port acting in its target role may withhold credit because it needs to send a frame itself. This prevents deadlocks where both ports are waiting on the other to provide credit.

7.13.4 Interlocked frames

Table 52 shows which frames shall be interlocked.

Table 52. Frame interlock requirements

Frame type	Interlock requirement
COMMAND	Interlocked
TASK	Interlocked
XFER_RDY	Interlocked
DATA	Non-interlocked
RESPONSE	Interlocked
AEN	Interlocked
AEN_RESPONSE	Interlocked

Before sending an interlocked frame, an SSP port shall wait for all frames to be acknowledged with ACK or NAK, even if credit is available. After sending an interlocked frame, an SSP port shall not send another frame until it has been acknowledged with ACK or NAK, even if credit is available.

Before sending a non-interlocked frame, an SSP port shall wait for:

a) all non-interlocked frames with different tags; and
b) all interlocked frames;

to be acknowledged with ACK or NAK, even if credit is available.

After sending a non-interlocked frame, an SSP port may send another non-interlocked frame with the same tag if credit is available. The port shall not send:

a) a non-interlocked frame with a different tag; or
b) an interlocked frame;

until all frames have been acknowledged with ACK or NAK, even if credit is available.

Interlocking does not prevent sending and receiving interlocked frames simultaneously (e.g., an initiator port could be sending COMMAND while receiving RESPONSE).

A port may send primitives responding to traffic on the back channel (e.g. an ACK or NAK to acknowledge a frame, or an RRDY to grant more backchannel credit) while waiting for an interlocked frame it sent to be acknowledged. These primitives may also be interspersed within the frame.

Figure 59 shows an example of interlocked frame transmission.

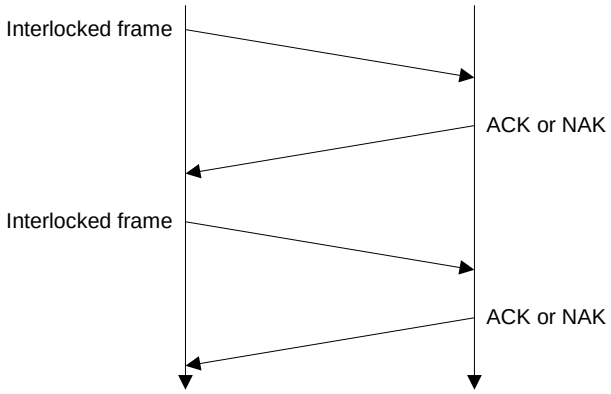

Figure 59. Interlocked frames

Figure 60 shows an example of non-interlocked frame transmission with the same tags.

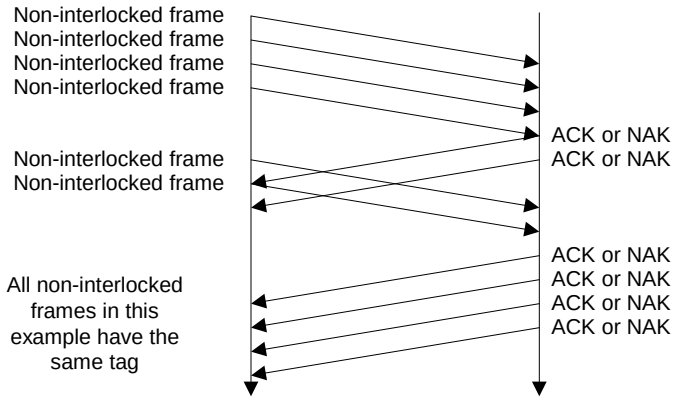

Figure 60. Non-interlocked frames with the same tag

Figure 61 shows an example of non-interlocked frame transmission with different tags.

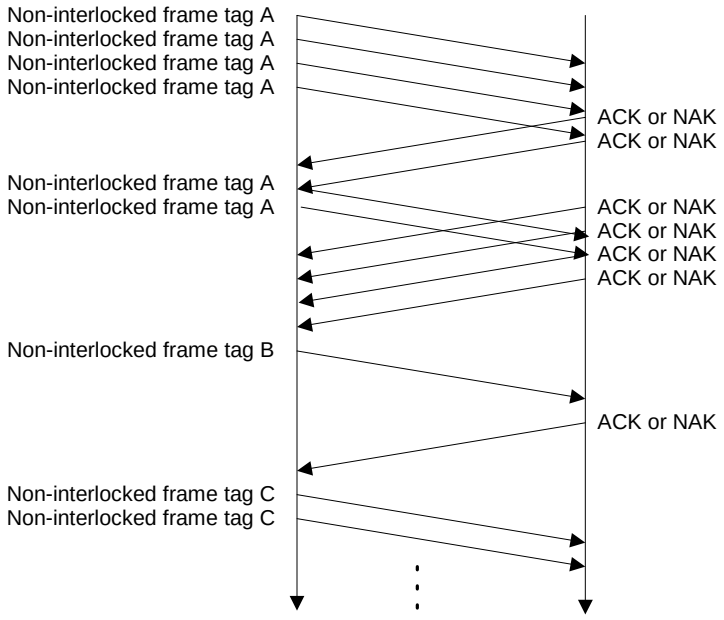

Figure 61. Non-interlocked frames with different tags

7.13.5 Preparing to close an SSP connection

DONE is exchanged prior to closing an SSP connection.

When it has no frames to send on its own, an SSP device should send DONE. This signals that it will originate no more frames during this connection. However, it may send ACK, NAK, and RRDY after sending DONE if the other device is still sending frames on the back channel.

Once a port has both sent and received DONE, it may close the connection by sending the CLOSE primitive sequence (see 7.11.7).

There are several versions of the DONE primitive indicating additional information about why the SSP connection is being closed. DONE(CLOSE CONNECTION) indicates normal completion; the sender has no more frames to send. DONE(CREDIT TIMEOUT) indicates the sender still has frames to send, but did not receive an RRDY granting frame credit within 1 ms. DONE(ACK/NAK TIMEOUT) indicates the sender sent a frame but did not receive the corresponding ACK or NAK within 1 ms; the ACK/NAK count is imbalanced and the sender will send a BREAK primitive sequence in 1 ms unless the recipient replies with DONE and the connection is closed with CLOSE.

7.13.6 SSP link layer state machines

7.13.6.1 Overview

The SSP link layer contains several state machines that run in parallel to control the flow of dwords on the link during an SSP connection. The SSP link state machines are as follows:

a) SAS phy receiver (R state machine);
~~c)~~ b) Receive frame control (RF state machine);
~~d)~~ c) Receive frame credit monitor (RCM state machine);
~~e)~~ d) Receive interlocked frame monitor (RIM state machine);
~~f)~~ e) SAS phy transmitter (T state machine);
~~g)~~ f) Transmit frame control (TF state machine);
~~h)~~ g) Transmit RRDY control (TR state machine);
~~i)~~ h) Transmit ACK/NAK control (TAN state machine);
~~j)~~ i) Transmit interlocked frame monitor (TIM state machine);
~~k)~~ j) Transmit frame credit monitor (TCM state machine); and
~~l)~~ k) DONE control (D state machine).

All the state machines within SSP shall begin on an indication of an enable SSP parameter from the SL state machine (see 7.11.8.1).

All the state machines within SSP shall stop after:

a) indicating using the request close parameter from the SSP_D1:DONE_wait state of the DONE control state machine to the SL state machine that the connection shall be closed;
~~c)~~ b) indicating using the request break parameter from the SSP_D1:DONE_wait state of the DONE control state machine to the SL state machine that a BREAK primitive sequence shall be transmitted; or
~~d)~~ c) receiving a BREAK primitive sequence.

If a state machine consists of multiple states the initial state is as indicated in state machine description in this subclause.

The R state machine's function is to receive primitives and frames and indicate to other SSP state machines the receipt of those dwords. The R state machine contains the SSP_R1:Receive state (see 7.13.6.2).

The RF state machine's function is to receive frames and indicate the successful or unsuccessful receipt of those frames. The RF state machine contains the following states:

a) Initial state: SSP_RF1:Frame_rcv (see 7.13.6.3); and
~~c)~~ b) SSP_RF2:Accepted_frame (see 7.13.6.4).

The RCM state machine's function is to ensure that there was credit given to the originator for every frame that is received. The RCM state machine contains the SSP_RCM1:Rcv_credit_monitor state (see 7.13.6.5).

The RIM state machine's function is to indicate to the accepted_frame state when the number of ACKs and NAKs transmitted equal the number of the EOFs received. The RIM state machine contains the SSP_RIM1:Rcv_interlock_monitor state (see 7.13.6.6).

The T state machine's function is to transmit primitives and frames to the link and indicate to other state machines the transmission of those dwords. The T state machine contains the SSP_T1:Transmit state (see 7.13.6.7).

The TF state machine's function it to control when the T state machine transmits an SOF, frame dwords, EOF, and a DONE to the link. The TF state machine contains the following states:

a) Initial state:SSP_TF1:Connected_idle state (see 7.13.6.8)
~~c)~~ b) SSP_TF2:ACK/NAK_wait state (see 7.13.6.9)
~~d)~~ c) SSP_TF3:Credit_wait state (see 7.13.6.10);
~~e)~~ d) SSP_TF4:Indicate_frame_tx state (see 7.13.6.11); and
~~f)~~ e) SSP_TF5:Indicate_DONE_tx state (see 7.13.6.12).

The TR state machine's function is to indicate to the transmit state that an RRDY is to be transmitted and indicate to the rcv_credit_monitor state any time an RRDY has been transmitted. The TR state machine contains the following states:

a) Initial state: SSP_TR1:Tx_RRDY_idle state (see 7.13.6.16); and
~~c)~~ b) SSP_TR2:Indicate_RRDY_tx state (see 7.13.6.17).

The TAN state machine's function is to indicate to the transmit state that an ACK or an NAK is to be transmitted and indicate to the rcv_interlock_monitor state any time an ACK or NAK has been transmitted. The TAN state machine contains the following states:

a) Initial state: SSP_TAN1:Tx_ACK/NAK_idle state (see 7.13.6.18); and
~~c)~~ b) SSP_TAN2:Indicate_ACK/NAK_tx state (see 7.13.6.19).

The TIM state machine's function is to indicate to the ACK/NAK_wait state when the number of ACKs and NAKs received is equal to the number of EOFs transmitted. The TIM state machine contains the SSP_TIM1:Tx_interlock_monitor state (see 7.13.6.13).

The TCM state machine's function is to ensure that credit was received from the originator before a frame is transmitted. The TCM state machine contains the SSP_TCM1:Tx_credit_monitor state (see 7.13.6.14).

The D state machine's function is to ensure a DONE has been received and transmitted before transitioning out of the connected state. The D state machine contains the SSP_D1:Done_wait state (see 7.13.6.15).

Figure 62 shows the states related to frame transmission.

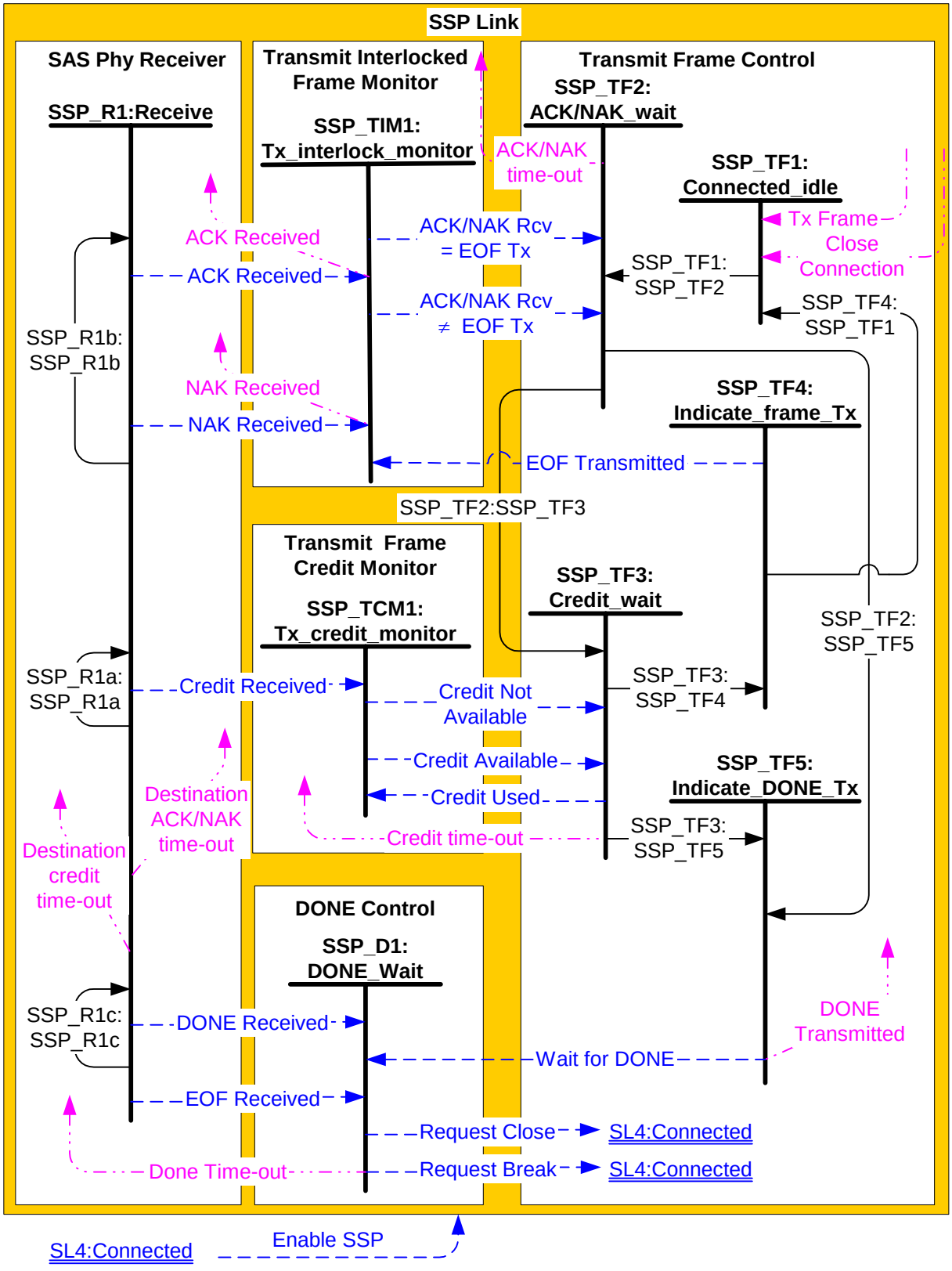

Figure 62. SSP link layer (part 1)

Figure 63 shows the states related to frame reception.

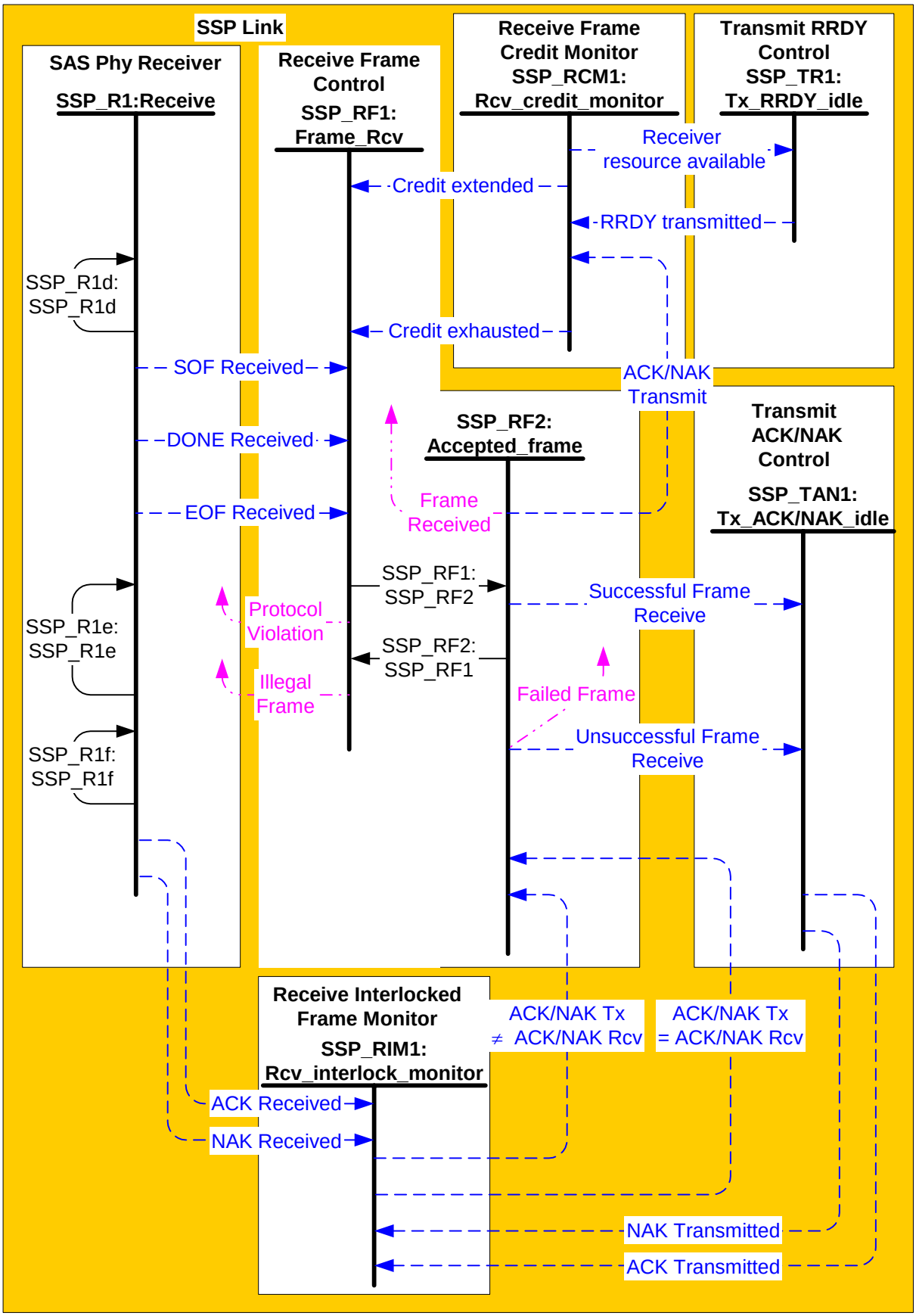

Figure 63. SSP link layer (part 2)

Figure 64 shows the states related to primitive reception.

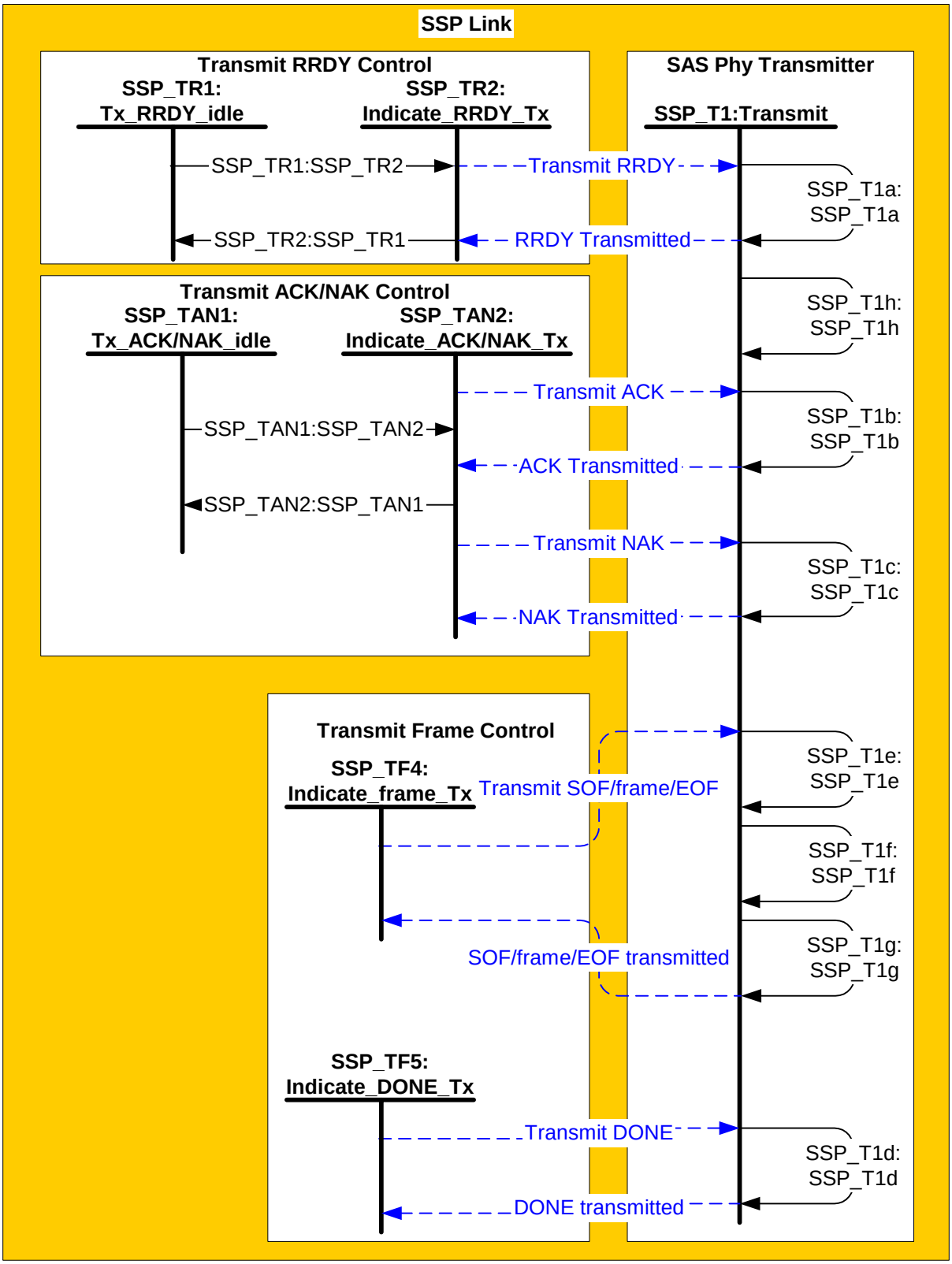

Figure 64. SSP link layer (part 3)

7.13.6.2 SSP_R1:Receive state

7.13.6.2.1 SSP_R1:Receive state description

The receive state receives frames and primitives.

As a result of receiving an RRDY the receive state shall send a credit received parameter to the tx_credit_monitor state.

As a result of receiving an ACK or a NAK the receive state shall send an ACK received parameter or a NAK received parameter to the following states:

a) tx_interlock_monitor state; and
~~c)~~ b) rcv_interlock_monitor state.

As a result of receiving a DONE the receive state shall:

a) send a DONE received parameter with its reason code to the DONE_wait state; and
~~c)~~ b) send a DONE received parameter to the frame_rcv state.

The receive state sends a confirmation to the port layer using the destination ACK/NAK time-out parameter if a DONE(ACK/NAK time-out) is received as a warning to the port layer that the transmitter is going to close the connection within one millisecond.

The receive state sends a confirmation to the port layer using the destination credit time-out parameter if a DONE(credit time-out) is received as a warning to the port layer that frame transmission was stopped by the transmitter because credit was unavailable for at least one millisecond.

As a result of receiving the SOF the receive state shall send an SOF received parameter to the frame_rcv state.

As a result of receiving the EOF the receive state shall send an EOF received parameter to the following states:

a) DONE_wait state; and
~~c)~~ b) rcv_interlock_monitor state.

7.13.6.2.2 Transition SSP_R1a:SSP_R1a (Receive:Receive)

The SSP_R1a:SSP_R1a transition shall occur every time an RRDY is received.

7.13.6.2.3 Transition SSP_R1b:SSP_R1b (Receive:Receive))

The SSP_R1b:SSP_R1b transition shall occur every time an ACK or a NAK is received.

7.13.6.2.4 Transition SSP_R1c:SSP_R1c (Receive:Receive))

The SSP_R1c:SSP_R1c transition shall occur every time a DONE is received.

7.13.6.2.5 Transition SSP_R1d:SSP_R1d (Receive:Receive)

The SSP_R1d:SSP_R1d transition shall occur every time an SOF is received.

7.13.6.2.6 Transition SSP_R1e:SSP_R1e (Receive:Receive)

The SSP_R1e:SSP_R1e transition shall occur every time an dword associated with a frame is received.

7.13.6.2.7 Transition SSP_R1f:SSP_R1f (Receive:Receive)

The SSP_R1f:SSP_R1f transition shall occur every time an EOF is received.

7.13.6.3 SSP_RF1:Frame_rcv state

7.13.6.3.1 SSP_RF1:Frame_rcv state description

The frame_rcv state checks the frame to determine if the frame should be accepted or discarded by the link.

The frame (i.e., all the dwords between an SOF and EOF) shall be discarded and a confirmation sent to the port layer using the illegal frame parameter that an illegal frame was received on the current connection if any one or more of the following conditions is true:

a) the last parameter received from the rcv_credit_monitor state was the credit exhausted transmitted parameter;
~~c)~~ b) the number of bytes between the SOF and EOF is greater than 1 024 bytes;
~~d)~~ c) the CRC is valid and the source or destination of the frame does not match the source and destination of the current connection; or

~~e)~~ ~~d)~~ a DONE received parameter has been received from the receive state.

If consecutive SOF received parameters are received from the receive state without an intervening EOF (i.e., SOF dwords SOF dwords EOF instead of SOF dwords EOF SOF dwords EOF) received parameter then the frame_rcv state shall:

a) not send any parameter to any other state;
~~c)~~ ~~b)~~ discard all dwords between the SOFs; and
~~d)~~ ~~c)~~ send a confirmation to the port layer using the protocol violation parameter that a protocol violation occurred on the current connection.

7.13.6.3.2 Transition SSP_FR1:SSP_RF2 (Frame_rcv:Accepted_frame)

The SSP_FR1:SSP_FR2 transition shall occur after the frame_rcv state determines the received frame shall be accepted.

A frame shall be accepted if:

a) the last parameter received from the rcv_credit_monitor state was the credit extended parameter;
~~c)~~ ~~b)~~ the number of bytes between the SOF and EOF is less than or equal 1 024 bytes;
~~d)~~ ~~c)~~ the CRC is valid; and
~~e)~~ ~~d)~~ the source and destination of the frame matches the source and destination of the current connection.

When a CRC error is detected the SSP__FR1:SSP_FR2 transition shall contain an indication that a CRC error occurred within the frame.

7.13.6.4 SSP_RF2:Accepted_frame state

7.13.6.4.1 Accepted_frame state description

The accepted_frame state determines if the last frame received was successfully received (e.g., no CRC indication).

If there was no indication of a CRC error the accepted_frame state shall:

a) send a successful frame receive parameter to the tx_ACK/NAK_idle state;
~~c)~~ ~~b)~~ send an ACK/NAK transmit parameter to the rcv_credit_monitor state; and
~~d)~~ ~~c)~~ send a confirmation to the port layer using the frame received parameter that a frame was received on the current connection. The frame received parameter shall contain an indication that either:
d) the number of ACK/NAKs transmitted is equal to EOFs received if last the last parameter received from the rcv_interlock_monitor was the ACK/NAK tx = EOF rcv parameter; or
~~f)~~ ~~e)~~ the number of ACK/NAKs transmitted is not equal to EOFs received if last the last parameter received from the rcv_interlock_monitor was the ACK/NAK tx _ EOF rcv parameter.

If there was an indication of an CRC error the accepted_frame state shall:

a) send an unsuccessful frame receive parameter to the tx_ACK/NAK_idle state;
~~c)~~ ~~b)~~ send an ACK/NAK transmit parameter to the rcv_credit_monitor state; and
~~d)~~ ~~c)~~ send a confirmation to the port layer using the failed frame parameter that a frame error (i.e, CRC error) occurred on the current connection.

7.13.6.4.2 Transition SSP_FR2:SSP_RF1 (Accepted_frame:Frame_rcv)

The SSP_FR2:SSP_FR1 transition shall occur after the parameters have been passed to the tx_ACK/ NAK_idle state, the rcv_credit_monitor state, and the port layer.

7.13.6.5 SSP_RCM1:Rcv_credit_monitor state

The rcv_credit_monitor state monitors the receivers resources and keeps track of the number of RRDYs transmitted verses the number of ACKs and NAKs transmitted.

Anytime resources are released or become available the rcv_credit_monitor state shall send a receiver resources available parameter to the tx_RRDY_idle state.

When the SSP state machines are entered into as the result of receiving an enable SSP parameter from the connected state there is no frame receiver resource credit for the current connection. The rcv_credit_monitor state shall only send a receiver resources available parameter to the tx_RRDY_idle state after frame receive resources become available. The specifications for when or how resources become available is outside the scope of this standard.

The rcv_credit_monitor state shall only indicate through the receiver resource available parameter the amount of resources available to handle received frames (e.g., if the rcv_credit_monitor state has resources for 5 frames the maximum number of receiver resources available requests outstanding is 5).

The rcv_credit_monitor state shall use the RRDY transmitted parameter from the tx_RRDY_Idle state to keep track of the number of RRDYs transmitted. The rcv_credit_monitor state shall use the ACK/NAK transmit parameter from the accepted_frame state to keep a track of the number of frames received.

Anytime the number of RRDYs exceeds the number of ACKs and NAKs the rcv_credit_monitor state shall indicate to the frame_rcv state using the credit extended parameter that credit has been given to the transmitter.

Anytime the number of RRDYs is less than or equal the number of ACKs and NAKs the rcv_credit_monitor state shall indicate to the frame_rcv state using the credit exhausted parameter that on credit has been given to the transmitter.

7.13.6.6 SSP_RIM1:Rcv_interlock_monitor state

The rcv_interlock_monitor state monitors the number of ACKs and NAKs transmitted verses the number of ACKs and NAKs received.

The rcv_interlock_monitor state shall use the ACK transmitted parameter and the NAK transmitted parameter from the tx_ACK/NAK_idle state to keep track of the number of ACKs and NAKs transmitted. The rcv_interlock_monitor state shall use the ACK received parameter and the NAK received parameter from the receive state to keep a track of the number of frames received.

Anytime the sum of the ACKs and NAKs transmitted does not equal the number of ACKs and NAKs received the rcv_interlock_monitor state shall indicate to the accepted_frame state using the ACK/NAK tx _ ACK/NAK rcv parameter that the number of frames transmitted is not equal to the number of frames received.

Anytime the sum of the ACKs and NAKs transmitted equals the number of ACKs and NAKs received the rcv_interlock_monitor state shall indicate to the accepted_frame state using the ACK/NAK tx = ACKs and NAKs rcv parameter that the number of frames transmitted is equal to the number of frames received.

When the connected state is entered into the sum of the ACKs and NAKs transmitted shall be set equal to the number of ACKs and NAKs received.

7.13.6.7 SSP_T1:Transmit state

7.13.6.7.1 SSP_T1:Transmit state description

The transmit state requests the link to:

a) transmit an RRDY when a transmit RRDY parameter is received from the indicate_RRDY_tx state;
~~c)~~ b) transmit an ACK when a transmit ACK parameter is received from the indicate_ACK/NAK_tx state;
~~d)~~ c) transmit an NAK when the transmit NAK parameter is received from the indicate_ACK/NAK_tx state;
~~e)~~ d) transmit a DONE with a reason code when the transmit DONE parameter is received from the indicate_DONE_tx state; and
~~f)~~ e) transmit an SOF/frame/EOF when the transmit SOF/frame/EOF parameter is received from the indicate_frame_tx state.

In the absence of any transmit requests the transmit state shall transmit idle dwords and ALIGNs as necessary.

On an indication to transmit the transmit state shall transmit the indicated primitive or dword in the next available dword. If there are multiple indications to transmit primitives the following priority should be followed when transmitting the primitives:

1) ALIGN;
~~3)~~ 2) RRDY;
~~4)~~ 3) ACK or NAK;
~~5)~~ 4) DONE; then
~~6)~~ 5) SOF or EOF.

On an indication that a SOF/frame/EOF is to be transmitted the transmit state shall transmit an SOF in the dword before the first dword of the frame and an EOF in first dword after the last dword of the frame. If during the transmission of a frame an indication that a primitive is to be transmitted occurs the transmit state may transmit the indicated primitive by inserting the primitive between the frames' dwords.

The transmit state shall indicate using the SOF/frame/EOF transmitted parameter to the indicate_frame_tx state each time an EOF is transmitted. The transmit state may indicate the EOF has been transmitted before the EOF has been transmitted if the transmit state supports multiple SOF/frame/EOF parameters.

The transmit state shall indicate using the RDDY transmitted parameter to the indicate_RRDY_tx state when an RRDY is transmitted.

The transmit state shall indicate using the ACK transmitted parameter to the indicate_ACK/NAK_tx state when an ACK is transmitted.

The transmit state shall indicate using the NAK transmitted parameter to the indicate_ACK/NAK_tx state when an NAK is transmitted.

The transmit state shall indicate using the DONE transmitted parameter to the indicate_DONE_tx state when a DONE is transmitted.

7.13.6.7.2 Transition SSP_T1a:SSP_T1a (Transmit:Transmit)

The SSP_T1a:SSP_T1a transition shall occur every time an RRDY is transmitted.

7.13.6.7.3 Transition SSP_T1b:SSP_T1b (Transmit:Transmit)

The SSP_T1b:SSP_T1b transition shall occur every time an ACK is transmitted.

7.13.6.7.4 Transition SSP_T1c:SSP_T1c (Transmit:Transmit)

The SSP_T1c:SSP_T1c transition shall occur every time a NAK is transmitted.

7.13.6.7.5 Transition SSP_T1d:SSP_T1d (Transmit:Transmit)

The SSP_T1d:SSP_T1d transition shall occur every time a DONE is transmitted.

7.13.6.7.6 Transition SSP_T1e:SSP_T1e (Transmit:Transmit)

The SSP_T1e:SSP_T1e transition shall occur every time an SOF is transmitted.

7.13.6.7.7 Transition SSP_T1f:SSP_T1f (Transmit:Transmit)

The SSP_T1f:SSP_T1f transition shall occur every time a dword associated with a frame is transmitted.

7.13.6.7.8 Transition SSP_T1g:SSP_T1g (Transmit:Transmit)

The SSP_T1g:SSP_T1g transition shall occur every time an EOF is transmitted.

7.13.6.7.9 Transition SSP_T1h:SSP_T1h (Transmit:Transmit)

The SSP_T1h:SSP_T1h transition shall occur every time an idle is transmitted.

7.13.6.8 SSP_TF1:Connected_idle state

7.13.6.8.1 SSP_TF1:Connected_idle state overview

The connected_idle state waits for a request from the port layer to transmit a frame or to close the connection.

7.13.6.8.2 Transition SSP_TF1:SSP_TF2 (Connected_idle:ACK/NAK_wait)

The SSP_TF1:SSP_TF2 transition shall occur when the port layer requests a frame be transmitted or the connection be closed.

The port layer indicates a frame is to be transmitted and whether the transmitted frame is an interlocked frame or a non-interlocked frame using the tx frame parameter. When the port layer indicates an interlocked frame is to be transmitted the SSP_TF1:SSP_TF2 transition shall contain an indication that the frame to be transmitted is an interlocked frame. When the port layer indicates a non-interlocked frame is to be transmitted the SSP_TF1:SSP_TF2 transition shall contain an indication that the frame to be transmitted is a non-interlocked frame.

The port layer indicates a connection is to be closed using the close connection parameter. When the port layer indicates the connection is to be closed the SSP_TF1:SSP_TF2 transition shall contain an indication that the connection is to be closed.

7.13.6.9 SSP_TF2:ACK/NAK_wait state

7.13.6.9.1 SSP_TF2:ACK/NAK_wait state overview

The ACK/NAK_wait state monitors the ACK/NAK rcv = EOF tx parameter and the ACK/NAK rcv _ EOF tx parameter from the tx_interlock_monitor state.

If the last parameter received from the tx_interlock_monitor state was a ACK/NAK rcv _ EOF tx parameter and an interlocked frame is to be transmitted then the number of ACKs and NAKs received and the number of frames transmitted is not in balance. The ACK/NAK_wait state shall then wait, up to a ACK/NAK time-out, for an ACK/NAK rcv = EOF tx parameter to be indicated from the tx_interlock_monitor state before exiting the ACK/NAK_wait state.

The ACK/NAK_wait state shall initialize an ACK/NAK time-out timer to one millisecond and start the timer each time it is transitioned into.

7.13.6.9.2 Transition SSP_TF2:SSP_TF3 (ACK/NAK_wait:Credit_wait)

The SSP_TF2:SSP_TF3 transition shall occur after an ACK/NAK rcv = EOF tx parameter from the tx_interlock_monitor state is indicated for interlocked frames or right away (i.e., zero time in the ACK/ NAK_wait state) for non-interlocked frames.

7.13.6.9.3 Transition SSP_TF2:SSP_TF5 (ACK/NAK_wait:Indicate_DONE_tx)

The SSP_TF2:SSP_TF5 transition shall occur if:

a) an ACK/NAK rcv = EOF tx parameter from the tx_interlock_monitor state is indicated; and

~~c)~~ b) if there was an indication by the transition from the connected_idle state that the connection is to be closed.

When the connection is to be closed the SSP_TF2:SSP_TF5 transition shall contain an indication that the port layer has requested the connection be closed.

If an ACK/NAK time-out occurs an SSP_TF2:SSP_TF5 transition shall occur after the ACK/NAK_wait state sends a confirmation to the port layer using the ACK/NAK time-out parameter that an ACK/NAK time-out occurred.

When a time-out has occurred the SSP_TF2:SSP_TF5 transition shall contain an indication that an ACK/ NAK time-out occurred.

7.13.6.10 SSP_TF3:Credit_wait state

7.13.6.10.1 SSP_TF3:Credit_wait state overview

The credit_wait state monitors the credit not available parameter and the credit available parameter from the tx_credit_monitor state.

If the last parameter received from the tx_credit_monitor state was a credit not available parameter then there is no transmit frame credit available and the credit_wait state shall wait, up to a credit time-out, until a credit available parameter is indicated from the tx_credit_monitor state before exiting the credit_wait state.

The credit_wait state shall initialize a credit time-out timer to one millisecond and start the timer on the transition into the credit_wait state.

7.13.6.10.2 Transition SSP_TF3:SSP_TF4 (Credit_wait:Indicate_frame_tx)

The SSP_TF3:SSP_TF3 transition shall occur after a credit available parameter from the tx_credit_monitor state indicates there is transmit frame credit is available to transmit a frame and after indicating to the tx_credit_monitor state using the credit used parameter that a transmit credit was used.

7.13.6.10.3 Transition SSP_TF3:SSP_TF5 (Credit_wait:Indicate_DONE_tx)

The SSP_TF3:SSP_TF5 transition shall occur if the credit time-out timer is exceeded before transmit frame credit is available and after the credit_wait state sends a confirmation to the port layer using the credit time-out parameter that a credit time-out occurred.

When a time-out has occurred the SSP_TF3:SSP_TF5 transition shall contain an indication that a credit time-out occurred.

7.13.6.11 SSP_TF4:Indicate_frame_tx state

7.13.6.11.1 SSP_TF4:Indicate_frame_tx state description

The indicate_frame_tx state requests a frame be transmitted by sending a transmit SOF/frame/ EOF parameter to the transmit state. Each time transmit SOF/frame/EOF parameter is sent from the indicate_frame_tx state one SOF/frame/EOF is transmitted by the transmit state.

7.13.6.11.2 Transition SSP_TF4:SSP_TF1 (Indicate_frame_tx:Connected_idle)

The SSP_TF4:SSP_TF1 transition shall occur after:

a) the transmit state has indicated to the indicate_frame_tx state using the SOF/frame/EOF transmitted parameter that the frame has been transmitted in the link; and

~~c)~~ b) the indicate_frame_tx state indicates to the tx_interlock_monitor state using the EOF transmitted parameter that a frame has been transmitted.

7.13.6.12 SSP_TF5:Indicate_DONE_tx state

The indicate_DONE_tx state requests on of the following DONEs be transmitted by sending a transmit DONE parameter to the transmit state:

a) DONE(close connection) if the transition into the indicate_DONE_tx indicated a port layer has requested the connection be closed;

~~c)~~ b) DONE(ACK/NAK time-out) if the transition into the indicate_DONE_tx indicated an ACK/NAK time-out occurred; or

~~d)~~ c) DONE(credit time-out) if the transition into the indicate_DONE_tx indicated a credit time-out time-out occurred.

After a DONE transmitted parameter has been indicated from the transmit state the indicate_DONE_tx state shall:

a) indicate to the DONE_wait state using the wait for DONE parameter that no more frames are going to be transmitted during the current connection with a reason code of CLOSE CONNECTION, ACK/NAK TIME-OUT, or CREDIT TIME-OUT; and

~~c)~~ b) send a confirmation to the port layer using the DONE transmitted parameter that a DONE was transmitted.

7.13.6.13 SSP_TIM1:Tx_interlock_monitor state

The tx_interlock_monitor state monitors the number of ACKs and NAKs received and the number of frames transmitted. The tx_interlock_monitor indicates to the ACK/NAK_wait state using the ACK/NAK rcv = EOF tx parameter when the number of ACKs and NAKs received is equal to the number of frames transmitted. The tx_interlock_monitor indicates to the ACK/ NAK_wait state using the ACK/NAK rcv _ EOF tx parameter when the number of ACKs and NAKs received is not equal to the number of frames transmitted.

The ACK/NAK/frame_cnt state keeps track of the number of frames transmitted using the EOF transmitted parameter received from the indicate_frame_tx state.

The ACK/NAK/frame_cnt state keeps track the number of frames confirmed received using the ACK received parameter and NAK received parameter from the receive state (i.e., for every frame transmitted it is required a ACK or NAK be received).

Every time an ACK received parameter or a NAK received parameter is received from the receive state the ACK/NAK_wait state shall send a confirmation to the port layer using the ACK received parameter or the NAK received parameter that an ACK or a NAK was received.

On entry into the connected state the number of ACKs and NAKs received shall be set equal to the number of EOFs transmitted.

7.13.6.14 SSP_TCM1:Tx_credit_monitor state

The tx_credit_monitor state shall keep track of the number of transmit frame credits received verses the number of transmit frame credits used. The tx_credit_monitor state adds transmit frame credit for each credit received parameter indication from the receive state and subtracts transmit frame credit for each credit used parameter indication from the credit_wait state.

The tx_credit_monitor indicates to the credit_wait state using the credit not available parameter any time there is no transmit frame credit available.

The tx_credit_monitor indicates to the credit_wait state using the credit available parameter any time there is transmit frame credit available.

When the connected state is entered into the initial value of the transmit frame credit in the tx_credit_monitor state shall be set to zero.

7.13.6.15 SSP_D1:DONE_wait state

7.13.6.15.1 DONE_wait state description

The DONE_wait state ensures that a DONE has been received and transmitted before transitioning to the close state. The DONE may be transmitted and received in any order.

After a DONE received parameter with a reason code of CLOSE CONNECTION or CREDIT TIME-OUT was received from the receive state and a wait for DONE parameter with a reason code of CLOSE CONNECTION or CREDIT TIME-OUT was received from the indicate_DONE_tx state the DONE_wait state shall indicate to the SL state machine (see 7.11.8.1) using the request close parameter that a CLOSE shall be transmitted.

After the DONE_wait state receives a wait for DONE parameter with a reason code of CLOSE CONNECTION or CREDIT TIME-OUT from the indicate_DONE_tx state it shall initialize a receive DONE time-out timer to one millisecond. If the timer is exceeded before an EOF received parameter or a DONE received parameter from the receive state is indicated the DONE_wait state shall;

a) send a confirmation to the port layer using the DONE time-out parameter that a done time-out occurred; and

~~c)~~ b) shall indicate to the SL state machine using the request break parameter that a BREAK primitive sequence shall be transmitted.

After the DONE_wait state receives a wait for DONE parameter with a reason code of ACK/NAK TIME-OUT from the indicate_DONE_tx state it shall initialize a receive DONE time-out timer to one millisecond. If the DONE_wait state receives a DONE received parameter from the receive state before the DONE time-out timer is exceeded the DONE_wait state shall indicate to the SL state machine using the request close parameter that a CLOSE shall be transmitted. If the timer is exceeded the DONE_wait state shall indicate to the SL state machine using the request break parameter that a BREAK primitive sequence shall be transmitted.

7.13.6.16 SSP_TR1:Tx_RRDY_idle state

7.13.6.16.1 Tx_RRDY_idle state description

The tx_RRDY_idle state waits for a receiver resource available parameter from the rcv_credit_monitor state.

When the tx_RRDY_idle state is transitioned into from the indicate_RRDY_tx state it shall indicate to the rcv_credit_monitor state using the RRDY transmitted parameter that an RRDY has been transmitted.

7.13.6.16.2 Transition SSP_TR1:SSP_TR2 (Tx_RRDY_idle:Indicate_RRDY_tx)

The SSP_TR1:SSP_TR2 transition shall after an indication from the rcv_credit_monitor state using the receiver resource available parameter that credit is available.

7.13.6.17 SSP_TR2:Indicate_RRDY_tx state

7.13.6.17.1 Indicate_RRDY_tx state description

The indicate_RRDY_tx state requests a single RRDY be transmitted by sending a transmit RRDY parameter to the transmit state each time it is transitioned into from the tx_RRDY_idle state.

7.13.6.17.2 Transition SSP_TR2:SSP_TR1 (Indicate_RRDY_tx:Tx_RRDY_idle)

The SSP_TR2:SSP_TR1 transition shall occur after an indication from the transmit state using the RRDY transmitted parameter that an RRDY was transmitted.

7.13.6.18 SSP_TAN1:Tx_ACK/NAK_idle state

7.13.6.18.1 Tx_ACK/NAK_idle state description

The tx_ACK/NAK_idle state waits for an indication from the accepted_frame state.

When the tx_ACK/NAK_idle state is transitioned into from the indicate_ACK/NAK_tx state with an indication that an ACK was transmitted it shall indicate to the rcv_interlock_monitor state using the ACK transmitted parameter that an ACK has been transmitted.

When the tx_ACK/NAK_idle state is transitioned into from the indicate_ACK/NAK_tx state with an indication that a NAK was transmitted it shall indicate to the rcv_interlock_monitor state using the NAK transmitted parameter that an NAK has been transmitted.

7.13.6.18.2 Transition SSP_TAN1:SSP_TAN2 (Tx_ACK/NAK_idle:Indicate_ACK/NAK_tx)

The SSP_TAN1:SSP_TAN2 transition shall occur on an indication from the accepted_frame state using the successful frame receive parameter that an ACK is to be transmitted or the unsuccessful frame receive parameter that a NAK is to be transmitted.

In the case of an successful frame receive a transmit ACK indication is passed to the indicate_ACK/ NAK_tx state.

In the case of a unsuccessful frame receive a transmit NAK indication is passed to the indicate_ACK/ NAK_tx state.

If multiple requests for transmitting ACKs and or NAKs occur then the order the ACK and NAK transmissions are requested shall be in the same order as the indications for transmitting the ACK sand NAKs were received.

7.13.6.19 SSP_TAN2:Indicate_ACK/NAK_tx state

7.13.6.19.1 Indicate_ACK/NAK_tx state description

The indicate_ACK/NAK_tx state indicates to the transmit state using the transmit ACK parameter or the transmit NAK parameter that a single ACK or NAK be transmitted each time it is transitioned into from the tx_ACK/NAK_idle state.

A transmit ACK parameter is indicated to the transmit state if the transition from the tx_ACK/NAK_idle indicated an ACK is to be transmitted.

A transmit NAK parameter is indicated to the transmit state if the transition from the tx_ACK/NAK_idle indicated a NAK is to be transmitted.

7.13.6.19.2 Transition SSP_TAN2:SSP_TAN1 (Indicate_ACK/NAK_tx:Tx_ACK/NAK_idle)

The SSP_TAN2:SSP_TAN1 transition shall occur after an indication from the transmit state using the ACK transmitted parameter or the NAK transmitted parameter that an ACK or NAK has been transmitted.

7.14 STP link layer

7.14.1 STP frame transmission

STP frame transmission is defined by SATA. During an STP connection, frames are preceded by SATA_SOF and followed by SATA_EOF as shown in Figure 65.

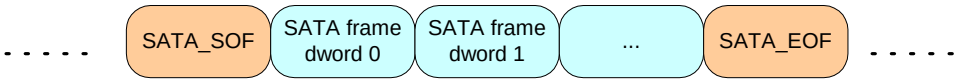

Figure 65. STP frame transmission

7.14.2 STP flow control

STP flow control is defined by SATA. The HOLD and HOLDA primitives are used as defined in SATA.

7.14.3 Preparing to close an STP connection

An STP initiator port shall not originate closing an STP connection.

An expander device may close an STP connection while the ~~SATA physical link is idle~~ the SATA device is waiting for SATA_RRDY. An expander device shall break an STP connection if the SATA physical link loses dword synchronization.

See 7.11.7 for details on closing connections.

7.15 SMP link layer

7.15.1 SMP frame transmission

Inside an SMP connection, the source device sends 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 66. There is no acknowledgement of SMP frames with ACK and NAK. There is no credit exchange with RRDY.

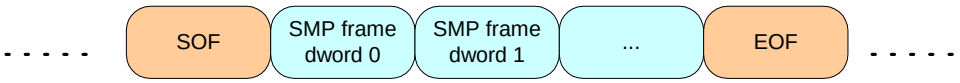

Figure 66. SMP frame transmission

7.15.2 SMP flow control

By accepting an SMP connection, the destination device indicates it is ready to receive one SMP_REQUEST frame.

When the source device sends one SMP_REQUEST frame, it shall be ready to receive one SMP_RESPONSE frame.

7.15.3 Preparing to close an SMP connection

After receiving the SMP_RESPONSE frame, the source device shall send a CLOSE primitive sequence to close the connection. The source device may leave the connection open to run loopback tests (see 7.7).

After receiving a CLOSE, the destination device shall reply with a CLOSE primitive sequence.

See 7.11.7 for details on closing connections.

7.15.4 SMP link layer state machines

7.15.4.1 Overview

The SMP link contains several state machines that run in parallel to control the flow of dwords on a link that are associated with the transfer of SMP frames. The SMP link state machines are as follows:

a) SAS phy receiver (R state machine);
~~c)~~ b) Originate SMP frame (OF state machine);
~~d)~~ c) SAS phy transmitter (T state machine); and
~~e)~~ d) SMP frame response (SFR state machine).

All the state machines within SMP shall begin on an indication of an enable SMP parameter from the SL state machine (see 7.11.8.1).

All the state machines within SMP shall stop after:

a) an indication of a request close parameter from the wait_transmit_frame state of the SMP frame response state machine to the SL state machine that the connection shall be closed;
~~c)~~ b) an indication of a request close parameter from the rcv_response_frame state of the originate SMP frame state machine to the SL state machine that the connection shall be closed;
~~d)~~ c) an indication of a request break parameter from the wait_originate_frame state of the SMP frame response state machine to the SL state machine that BREAK primitive sequence shall be shall be transmitted;
~~e)~~ d) an indication of a request break parameter from the rcv_response_frame state of the originate SMP frame state machine to the SL state machine that BREAK primitive sequence shall be shall be transmitted; or
~~f)~~ e) receiving a BREAK primitive sequence.

If a state machine consists of multiple states the initial state is as indicated in state machine description in this subclause.

The R state machine's function is to receive primitives and frames and indicate to other SMP state machines the receipt of those dwords. The R state machine contains the SMP_R1:Receive state (see 7.15.4.2).

The OF state machine's function is to transmit an SMP frame and then receive the corresponding response frame indicating the successful or unsuccessful receipt of the response frame. The OF state machine contains the following states:

a) Initial state: SMP_OF1:Command_idle (see 7.15.4.3);

~~c)~~ b) SMP_OF2:Indicate_frame_tx (see 7.15.4.4);
~~d)~~ c) SMP_OF3:Rcv_response_frame (see 7.15.4.5).

The T state machine's function is to transmit primitives and frames to the link and indicate to other state machines the transmission of those dwords. The T state machine contains the SMP_T1:Transmit state (see 7.15.4.6).

The SFR state machine's function is to receive an SMP frame indicating the successful or unsuccessful receipt of the SMP frame and then transmit the corresponding response frame to the link. The SFR state machine contains the following states:

a) Initial state:SMP_SFR1:Wait_orginate_frame state (see 7.15.4.7); and
~~c)~~ b) SMP_SFR2:wait_transmit_frame state (see 7.15.4.8); and
~~d)~~ c) SMP_SFR3:Loopback (see 7.15.4.9).

Figure 67 shows the states related to frame transmission.

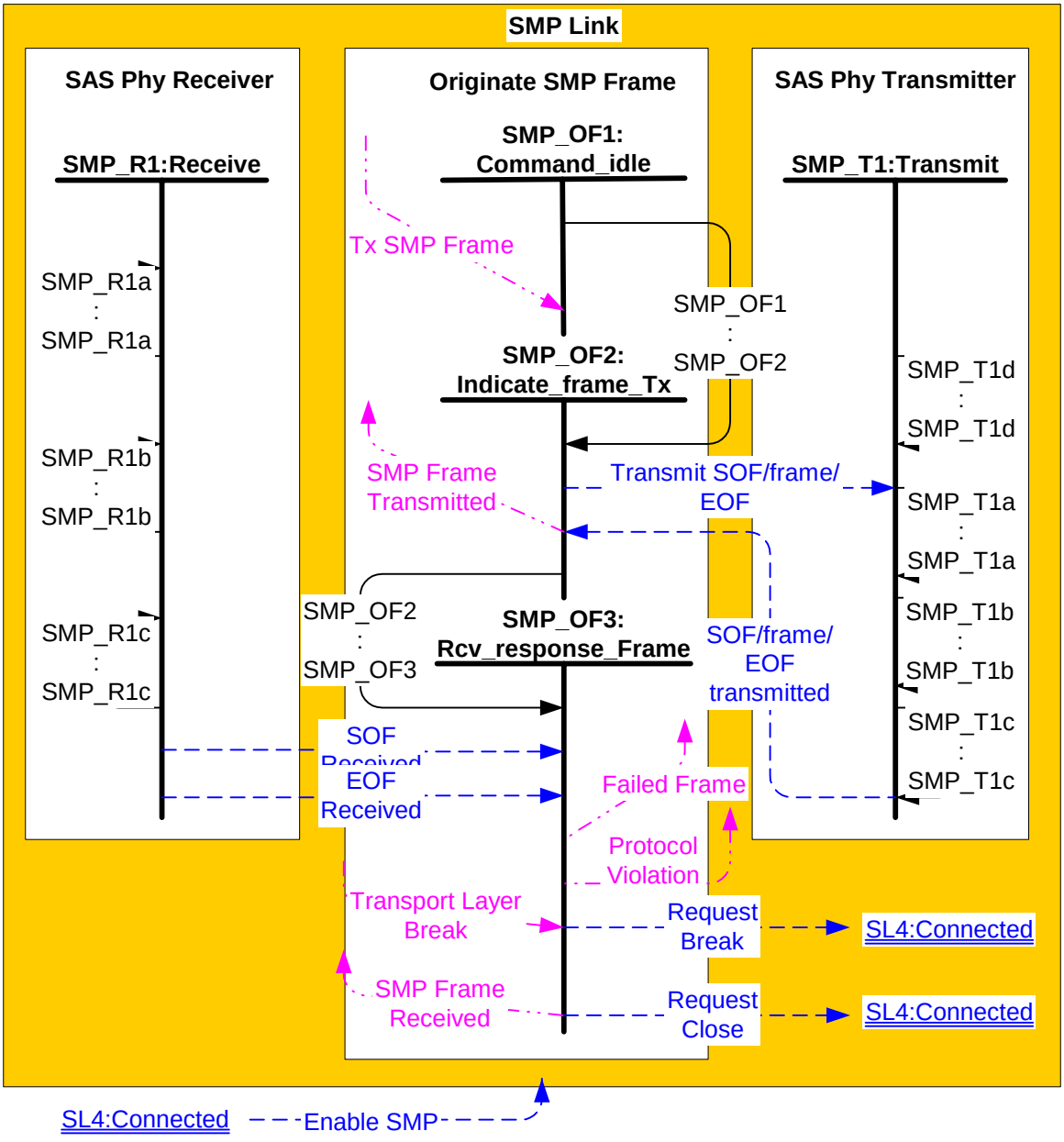

Figure 67. SMP link layer (part 1)

Figure 68 shows the states related to frame reception.

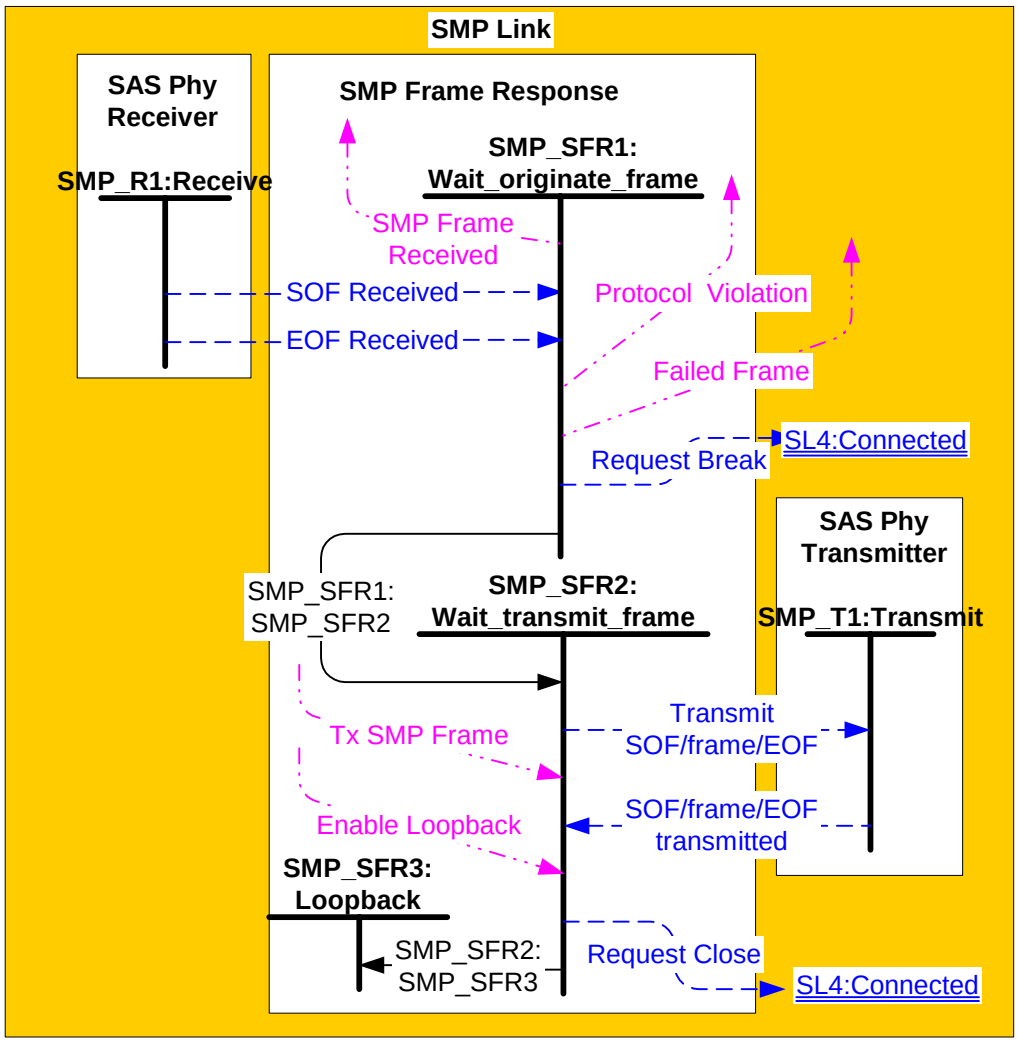

Figure 68. SMP link layer (part 2)

7.15.4.2 SMP_R1:Receive state

7.15.4.2.1 SMP_R1:Receive state description

The receive state receives frames and primitives.

As a result of receiving an SOF the receive state shall indicate using the SOF received parameter that an SOF was received to the following states:

a) wait_originate_frame state; and
~~c)~~ b) rcv_response_frame state.

As a result of receiving an EOF the receive state shall indicate using the EOF received parameter that an EOF was received to the following states:

a) wait_originate_frame state; and
~~c)~~ b) rcv_response_frame state.

7.15.4.2.2 Transition SMP_R1a:SMP_R1a (Receive:Receive)

The SMP_R1a:SMP_R1a transition shall occur every time an SOF is received.

7.15.4.2.3 Transition SMP_R1b:SMP_R1b (Receive:Receive)

The SMP_R1b:SMP_R1b transition shall occur every time an dword associated with a frame is received.

7.15.4.2.4 Transition SMP_R1c:SMP_R1c (Receive:Receive)

The SMP_R1c:SMP_R1c transition shall occur every time an EOF is received.

7.15.4.3 SMP_OF1:Command_idle state

7.15.4.3.1 SMP_OF1:Command_idle state description

The command_idle state shall only transition to the indicate_frame_tx states on a request from the port layer to transmit an SMP frame.

7.15.4.3.2 Transition SMP_OF1:SMP_OF2 (Command_idle:Indicate_frame_tx)

The SMP_OF1:SMP_OF2 transition shall occur when the port layer requests using the tx SMP frame parameter that an SMP frame is to be transmitted

7.15.4.4 SMP_OF2:Indicate_frame_tx state

7.15.4.4.1 SMP_OF2:Indicate_frame_tx state description

The indicate_frame_tx state indicates to the transmit state using the transmit SOF/frame/EOF parameter that an SMP frame be transmitted.

After the SOF/frame/EOF transmitted parameter is received from the transmit state the indicate_frame_tx state shall send a confirmation to the port layer using the SMP frame transmitted parameter that the SMP frame has been transmitted.

7.15.4.4.2 Transition SMP_OF2:SMP_OF3 (Indicate_frame_tx:Rcv_response_frame)

The SMP_OF2:SMP_OF3 transition shall occur after the indicate_frame_tx state has confirmed to the port layer using the SMP frame transmitted parameter that an SOF/frame/EOF has been transmitted.

7.15.4.5 SMP_OF3:Rcv_response_frame state

7.15.4.5.1 SMP_OF3:Rcv_response_frame state description

The rcv_response_frame state checks the response SMP frame and determines if the SMP frame was successfully received (e.g., no CRC error).

If the SMP response frame is received with a CRC error the rcv_response_frame state shall send a confirmation to the port layer using the failed frame parameter that the SMP response frame receive error occurred.

If the number of bytes between the SOF and EOF of the SMP response frame is greater than 1 024 bytes or the source and destination of the SMP response frame does not match the source and destination of the current connection the rcv_response_frame state shall send a confirmation to the port layer using the protocol violation parameter that the SMP response frame receive error occurred.

If the SMP response frame is received with no CRC error and the SMP response frame is valid the rcv_response_frame state shall:

a) send a confirmation to the port layer using the SMP frame received parameter that the SMP response frame was received; and

~~c)~~ b) indicate to the SL state machine (see 7.11.8.1) using the request close parameter that a CLOSE shall be transmitted.

If a transport layer break parameter is requested from the port layer the rcv_response_frame state shall indicate to the SL state machine using the request break parameter that a BREAK primitive sequence shall be transmitted.

7.15.4.6 SMP_T1:Transmit state

7.15.4.6.1 SMP_T1:Transmit state description

The transmit state requests the link to transmit an SOF/frame/EOF when the indicate_frame_tx state using the transmit SOF/frame/EOF parameter indicates an SMP frame be transmitted.

In the absence of any transmit requests the transmit state shall transmit idle dwords and ALIGNs as necessary.

On an indication that a SOF/frame/EOF is to be transmitted the transmit state shall transmit an SOF in the dword before the first dword of the frame and an EOF in first dword after the last dword of the frame. If during the transmission of a frame an indication that a primitive is to be transmitted occurs the transmit state may transmit the indicated primitive by inserting the primitive between the frames' dwords.

The transmit state shall indicate using the SOF/frame/EOF transmitted parameter to the indicate_frame_tx state each time an EOF is transmitted.

7.15.4.6.2 Transition SMP_T1a:SMP_T1a (Transmit:Transmit)

The SMP_T1a:SMP_T1a transition shall occur every time an SOF is transmitted.

7.15.4.6.3 Transition SMP_T1b:SMP_T1b (Transmit:Transmit)

The SMP_T1b:SMP_T1b transition shall occur every time a dword associated with a frame is transmitted.

7.15.4.6.4 Transition SMP_T1c:SMP_T1c (Transmit:Transmit)

The SMP_T1c:SMP_T1c transition shall occur every time an EOF is transmitted.

7.15.4.6.5 Transition SMP_T1d:SMP_T1d (Transmit:Transmit)

The SMP_T1d:SMP_T1d transition shall occur every time an idle is transmitted.

7.15.4.7 SMP_SFR1:Wait_originate_frame state

7.15.4.7.1 SMP_SRF1:Wait_originate_frame state description

The wait_originate_frame state checks the originate SMP frame and determines if the SMP frame was successfully received (e.g., no CRC error).

If the SMP originate frame is received with a CRC error the wait_originate_frame state shall:

a) send a confirmation to the port layer using the failed frame parameter that the SMP response frame receive error occurred; and

~~c)~~ b) indicate to the SL state machine (see 7.11.8.1) using the request break parameter that a BREAK primitive sequence shall be transmitted.

If the number of bytes between the SOF and EOF of the SMP originate frame is greater than 1 024 bytes or the source and destination of the SMP originate frame does not match the source and destination of the current connection the wait_originate_frame state shall:

a) send a confirmation to the port layer using the protocol violation parameter that the SMP response frame receive error occurred; and

~~c)~~ b) indicate to the SL state machine using the request break parameter that a BREAK primitive sequence shall be transmitted.

7.15.4.7.2 Transition SMP_SFR1:SMP_SFR2 (Wait_orginate_frame:Wait_transmit_frame)

The SMP_SFR1:SMP_SFR2 transition shall occur after the wait_originate_frame state determines the received SMP originate frame is accepted and after the rcv_response_frame state sends a confirmation to the port layer using the SMP frame received parameter that the SMP originate frame was received.

An SMP originate frame shall be accepted if:

a) there was no CRC error;
b) the number of bytes between the SOF and EOF is less than or equal to 1 024 bytes; and

~~d)~~ c) the source and destination of the SMP originate frame matches the source and destination of the current connection.

7.15.4.8 SMP_SFR2:Wait_transmit_frame state

7.15.4.8.1 SMP_SFR2:Wait_transmit_frame state description

The wait_transmit_frame state, after receiving a request from the port layer using the tx SMP frame parameter or an enable loopback parameter, indicates to the transmit state using the transmit SOF/frame/EOF parameter that an SMP response frame is to be transmitted.

If the enable loopback was not requested by the port layer then after the SOF/frame/EFO transmitted parameter is indicated from the transmit state the indicate_frame_tx state shall indicate to the SL state machine (see 7.11.8.1) using the request close parameter that a CLOSE shall be transmitted.

7.15.4.8.2 Transition SMP_SFR2:SMP_SFR3 (Wait_transmit_frame:Loopback)

The SMP_SFR2:SMP_SFR3 transition shall occur after the SOF/frame/EFO transmitted parameter is indicated from the transmit state if the port layer requested loopback be enabled.

7.15.4.9 SMP_SFR3:Loopback state

The loopback state shall cause all dwords received on the link to be transmitted back onto the link without any modifications to those dwords.

8 Port layer

8.1 Overview

The port connection (PC) state machines interface with the link layer connection state machine(s) (the SAS link endpoint connection state machine) and the SSP, SMP, and STP transport layer state machines to establish port connections and disconnections. Three state machines that run in parallel to accomplish this.

The PC connection state machines are as follows:

a) Receive and queue connection requests (R&Q state machine);
b) Counters and timers (TMR state machine); and
c) Control (Port control state machine).

The Target R&Q machine's purpose is to:

a) Queue all of the Interlocked or Non Interlocked frame transmission requests to the specified destination port. For SSP initiator ports this consists of COMMAND, DATA, TASK, and AEN_RESPONSE frames, as well as SMP Request frames. For SSP target ports this consists of DATA, RESPONSE, XFER_RDY, and AEN frames, as well as SMP Response frames to all Initiators.
b) Determine the priority of frame transmission;
c) If not connected, initiate a connection sequence;
d) If /(When) connected, indicate to the transport layer when a singe frame transmission can occur;
e) Notify the transport layer if a connection cannot be opened;
f) For Targets Only, initialize and monitor the counters and timers supported in the Disconnect-Reconnect mode page;
g) If a Disconnect-Reconnect counter or timer reaches the specification limit then initiate a Disconnect sequence and notify the transport layer. The transport layer will later request a new data transmission request to continue with the data transmission.
h) If a Disconnect sequence is requested from the Transport layer, then initiate a disconnect sequence;
i) Notify the transport layer if a connection has been exited (this also allows transport layer recovery for premature disconnections);
j) If a DONE has been transmitted, do not allow another IU transmission until after the connection has been completely closed down (SL0:Idle entered);
k) If DONEs have been transmitted and received, or a CLOSE or BREAK has been either transmitted or received, do not allow any transmissions and ignore any receptions until after the connection has been completely closed down (SL0:Idle entered); and
l) Initiate a "Close Connection" function to get DONE transmitted if the device is opened by a receive function and has nothing to transmit to this Initiator (or Target).

The Target TMR machine's purpose is to:

a) Accept initialization by the R&Q machine;
b) Time the connected milliseconds and notify the R&Q machine when this value exceeds the Maximum Connect Time limit specified in the Disconnect-Reconnect mode page for SSP devices;
c) Count the number of bytes transmitted by the target in a connection and notify the R&Q machine when this value exceeds the Maximum Burst Size limit specified in the Disconnect-Reconnect mode page for SSP devices; and
d) Time the Arbitration Wait Time (AWT).

The C machine's purpose is to:

a) Indicate if the device is not connected for receive or transmit;
b) Indicate if the device is connected for receive or transmit;
c) Request or retry a connection sequence via the link layer;
d) Initialize the Arbitration Wait Timer;
e) Create the "Open Connection" request to the link layer which also includes:
    A) Destination port identifier of the SAS port to be opened;
    B) The protocol to be used;
    C) The AWT value to be placed into the Open Address Frame by the link layer; and
    D) The link rate to be used to the arb_sel state.

f) Indicate if a connection could or could not be established;
g) Indicate if the device is connected but cannot transmit an IU; and
h) Indicate if the device is in process of closing a connection and cannot receive or transmit until after the connection is closed and a new connection is established.

Figure 69 shows the PC state machines.

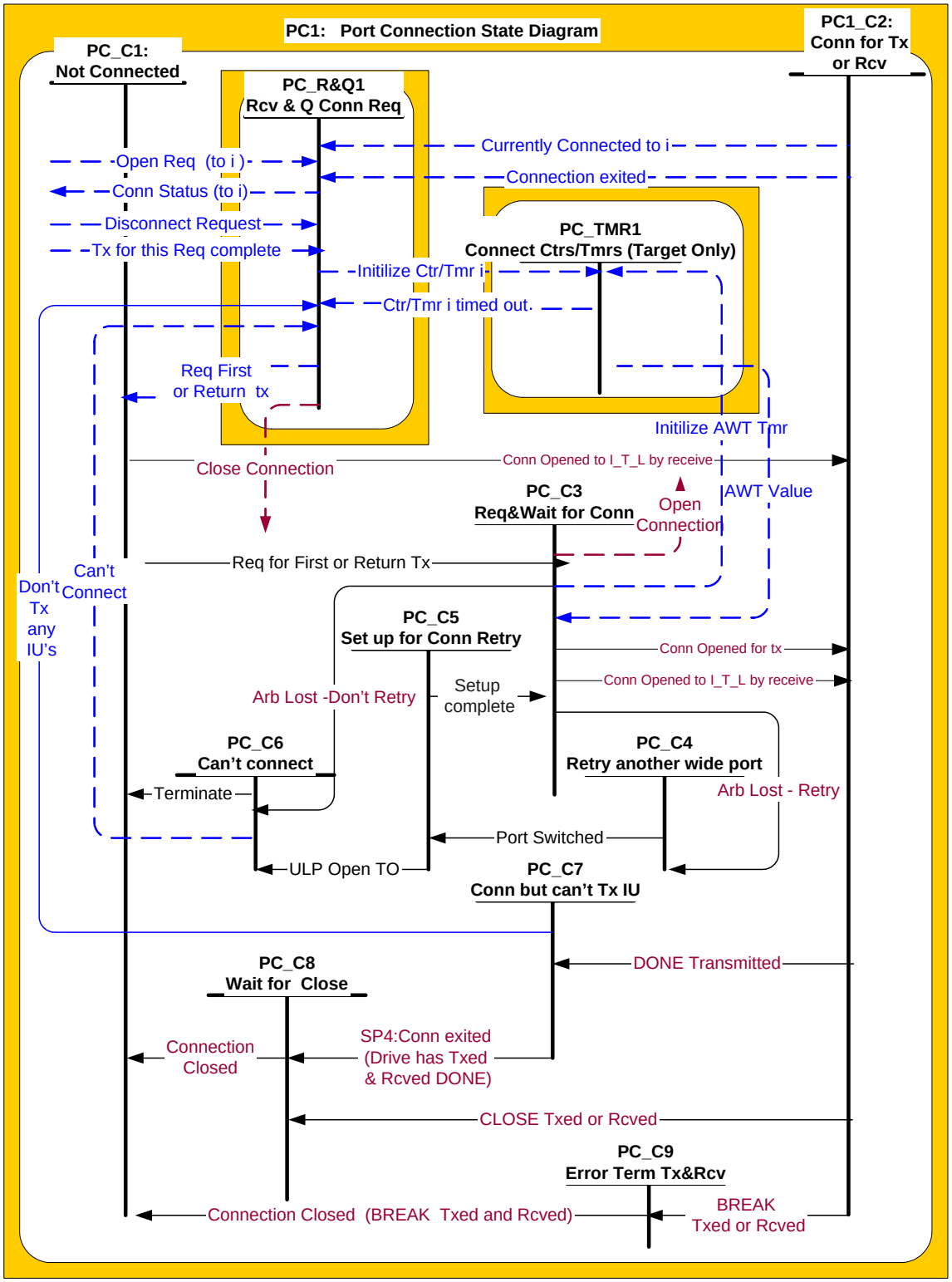

Figure 69. Port connection and supporting state machines

8.2 R&Q1:Receive and queue connection request state machine

A transport layer state machine requests an open (Open Req) for a specific destination port to transmit at least one interlocked or non interlocked frame. Multiple requests (normally with different tag values) may exist at the same time to the same or different destination ports. The R&Q1 state machine queues each of these requests and determines the priority order of servicing. The priorities are as follows, from highest to lowest priority:

1) A destination port currently connected;
2) the continuation of transmission of non-interlocked frames;
3) returned to transmission request; and
4) next highest priority queued request.

The R&Q1 state machine responds to the transport layer with a connection status (Conn Status) for the requested destination port. The status is be one of:

a) Connected to this destination – frame transmission may occur;
b) Wait for connection to this destination;
c) Arbitration Lost (Arb Lost) – No destination;
d) Arbitration Lost – ULP Open time out exceeded; or
e) Connection Exited (either normally, or prematurely).
f) Disconnect in process – Disconnect-Reconnect timer/counter exceeded

If a) status is sent, the transport layer may transmit this frame. After the frame has been sent (i.e., “Tx for this Req complete” signal received), the R&Q1 state machine shall clear its request queue of this request and check if this was an Interlocked or Non Interlocked frame.

a) If an Interlocked frame was sent, the R&Q machine shall check its request queue for the next frame to be transmitted.
b) If a Non Interlocked frame was sent, The R&Q machine shall wait for the next subsequent Open request (or Disconnect request) from this requester.

If not connected and in the PC_C1 state, the R&Q machine shall send to the transport layer a “Wait for connection to this destination port” Conn Status and initiate a connection sequence by issuing a “request a first or return tx” request.

a) The “first” refers to the first time this transmission request has been serviced and shall cause the AWT timer to be initialized when the PC_C3 state is entered.
b) The “return” refers to establishing a connection for a frame transmission that had previously requested an open but had been preempted by a connection opened to this port from another source port as a result of a selected (SL2_Selected) path. This shall cause the AWT timer to not be initialized when the PC_C3 state is entered.

The PC_C3 state shall perform a “Open Connection” request to the link layer. One of the following situations shall result:

a) The connection is opened for this destination port as a result of the request for this transmit connection (SL1:Arb_Sel path) and the requested frame transmission can occur.
b) The connection is opened for this destination port as a result of a selected (SL2_Selected) path and the requested frame transmission can occur.
c) The connection is opened with this port as the destination port as a result of a selected (SL2_Selected) path and the requested frame transmission can not occur. In this case the R&Q machine shall reexamine its queue to determine if any transmit requests exists for this different Target (or Initiator). The original frame shall be requeued for transmission with a “return” status for priority and AWT control.
d) The connection is denied and is not established. The transport layer shall be notified and the R&Q machine shall check its queue for the next transmit request

For Target ports only, when a connection is first established (detected by the leading edge of the “Currently Connected to i” signal from the PC_C2 state), the R&Q machine shall initialize the counters and timers in the TMR machine.

For target ports only, if connected, the R&Q machine shall check to see if any of the CTR/Timers have exceeded the specified limits after each frame transmission and prior to sending Connection Status back to the transport layer. (i.e., check CTR/Timers after each “Tx for this Req complete” signal is received and check CTR/Timers after each “Open Req (to i)” is received.

a) If the CTR/Timer’s have not exceeded the specified limits, frame transmission may continue.

b) If the CTR/Timer's have exceeded the specified limits, The R&Q machine shall request a "Close Connection" function from the Link layer and notify the transport layer with a "Disconnect in process – Disconnect-Reconnect timer/counter exceeded" status. To continue transmission of this data, the transport layer shall generate a new "Open Req" transmission request which is placed in the request queue.

If a Disconnect request is received, The R&Q machine shall request a "Close Connection" function from the Link layer. Any queue entry transmission waiting for a response from the transport layer shall also be cleared.

When a connection is exited, the R&Q machine shall notify the Transport layer. (This also allows the transport layer to recover for a premature disconnection if a frame transmission has been approved but before the transport layer has notified the R&Q machine that the transmission is completed)

If a DONE has been transmitted, the R&Q machine shall not allow another IU transmission until after the connection has been completely closed down (SL0:Idle entered)

If DONEs have been transmitted and received, or if a CLOSE or BREAK has been transmitted or received, the R&Q machine shall not allow any transmissions and shall ignore any receptions until after the connection has been completely closed down (SL0:Idle entered)

The R&Q machine shall detect if the connection was opened as a result of a device selection (detect "Currently Connected to i" signal from PC_C2 via the SL2_Selected path), determine that there is no pending transmit requests in the queue to this Initiator (or Target) and request a 'Close Connection" function from the link layer so that a DONE is transmitted to the destination port so that the connection can be closed by a DONE being received from the destination port.

8.3 TMR1:Counters and timers state machine

This state machine contains the timers and counters to measure and determine when limits have been exceeded for the optional SSP target port Disconnect-Reconnect mode page.

These timers and counters are initialized by the R&Q machine when a connection is first established.

The Maximum Connect Time timer times the connected milliseconds during SSP connections and notifies the R&Q machine when this value exceeds the limit specified in the MAXIMUM CONNECT TIME field of the Disconnect-Reconnect mode page.

The Maximum Burst Size limit counter counts the number of bytes transmitted by the target port in an SSP connection and notifies the R&Q1 machine when this value exceeds the limit specified in the MAXIMUM BURST SIZE field of the Disconnect-Reconnect mode page.

This state also contains the Arbitration Wait Timer. This timer shall time the amount of time since arbitration was first started for a frame transmission. This timer value shall be passed to the link layer when an "Open Connection" function is requested.

This timer shall be initialized when a request for a "first" time frame transmission has occurred.

This timer shall be reinitialized when the following reason codes are received from the link layer:

a) OPEN FAILED-RETRY.

This timer shall not be reinitialized when the following reason codes are received from the link layer:

a) OPEN FAILED-WRONG DESTINATION;
b) OPEN FAILED-INVALID LINK RATE;
c) OPEN FAILED-INVALID PROTOCOL TYPE;
d) OPEN FAILED-PATHWAY BLOCKED; or
e) ARBITRATION LOST-OPEN TIME-OUT OCCURRED.

This timer shall not be initialized when a request for a "return" frame transmission is requested.

8.4 Port connection state machine

8.4.1 PC_C1:Not Connected state

8.4.1.1 State description

The Not Connected state is the idle state for the Port connection state machine.

This state is entered when no connections exist on the port. This state is entered when all connection is closed.

This state is exited when a receive connection is established by a link layer. This state is exited when the R&Q state machine requests a first or return transmit.

8.4.1.2 Transition PC_C1:PC_C2 (Not Connected:Connected for Transmit or Receive)

The PC_C1:PC_C2 transition shall occur when a connection request is accepted by established by a link layer (the link layer SL4:Connected state is entered via the SL2:Selected state).

8.4.1.3 Transition PC_C1:PC_C3 (Not Connected:Request and Wait for connection)

The PC_C1:PC_C3 transition shall occur when the R&Q state machine requests a first or continued transmit function be performed.

8.4.2 PC_C2:Connected for Transmit or Receive (Conn for Tx or Rcv) state

8.4.2.1 State description

This state indicates that a connection exists between two ports and that primitives and frames may be transmitted and received between them.

This state shall notify the R&Q state machine when a connection is first established, that a connection exists, and shall indicate when the connection is closed.

8.4.2.2 Transition PC_C2:PC_C7 (Conn for Tx or Rcv:Conn but cannot Tx IU)

The PC_C2:PC_C7 transition shall occur when a DONE has been transmitted by this port.

8.4.2.3 Transition PC_C2:PC_C8 (Conn for Tx or Rcv:Wait for Close)

The PC_C2:PC_C8 transition shall occur when a CLOSE has been transmitted or received by this port.

8.4.2.4 Transition PC_C2:PC_C9 (Conn for Tx or Rcv:Error Term Tx and Rcv)

The PC_C2:PC_C9 transition shall occur when a Break has been transmitted or received by this port.

8.4.3 PC_C3:Request and Wait for a connection (Req&Wait for conn) state

8.4.3.1 State description

This state requests an “Open Connection” function to a specific destination port to be performed by a link layer and interprets the resultant responses.

When this state is entered for the first time from the PC_C1:Not Connected state, this state shall:

a) initialize a ULP open time out timer to 1 ms which limits the total time than an open function is attempted for this open request to the specified destination port;
b) Select the link through which the connection request is to be made;
c) Initialize the AWT for a “req for first tx” frame transmission request; and
d) Not initialize the AWT for a “req for return” tx frame transmission request.

This state shall build the “Open Connection” parameter list for the link layer which consists of:

a) Destination port device name;
b) The protocol to be used;
c) The link rate to be used; and
d) The current AWT timer value

This state shall pass the “Open Connection” request plus parameters to the Link layer.

This state shall then monitor the link layer response to determine the result of this “Open Connection” request or to determine if an receive connection occurred, overriding the “Open Connection” request.

8.4.3.2 Transition PC_C3:PC_C2 (Req&Wait for conn:Conn for Tx or Rcv)

The PC_C3:PC_C2 transition shall occur as a result of link layer state machine establishing a connection.

8.4.3.3 Transition PC_C3:PC_C4 (Req&Wait for conn:Retry another wide port)

The PC_C3:PC_C4 transition shall occur as a result of receiving from the link layer:

a) OPEN FAILED-RETRY;
b) OPEN FAILED-PATHWAY BLOCKED; or
c) ARBITRATION LOST-OPEN TIME-OUT OCCURRED.

8.4.3.4 Transition PC_C3:PC_C6 (Req&Wait for conn:Cannot connect)

The PC_C3:PC_C6 transition shall occur as a result of receiving from the link layer:

a) OPEN FAILED-WRONG DESTINATION;
b) OPEN FAILED-INVALID LINK RATE; or
c) OPEN FAILED-INVALID PROTOCOL TYPE.

8.4.4 PC_C4:Retry another wide port state

8.4.4.1 State description

This state shall set up the next link (if the port is a wide port) to attempt another Open Connection function.

If only one port exists the same link shall be re-specified.

If more than one port exists, then this state may select a different link to be used for the Open Connection function.

8.4.4.2 Transition PC_C4:PC_C5 (Retry another wide port:Set up Conn Retry)

The PC_C4:PC_C5 transition shall occur when the port for the Open Connection function has been set up or switched.

8.4.5 PC_C5:Set up for Connection Retry (Set up for Conn Retry) state

8.4.5.1 State definition

This state shall first check to see if the UPL Open timeout timer has exceeded the specified limit and take the appropriate action if it has.

If the ULP Open timeout timer has not reached its limit, this state shall ensure that the link layer is in the SL0:Idle state prior to transitioning to the PC_C3 state.

This state shall reinitialize the AWT timer if the link layer status was OPEN FAILED-RETRY.

This state shall not reinitialize the AWT timer if the link layer status was OPEN FAILED-PATHWAY BLOCKED OR ARBITRATION LOST-OPEN TIME-OUT OCCURRED.

This state shall delay 15 microseconds before it can be exited.

8.4.5.2 Transition PC_C5:PC_C3 (Set up for Conn Retry:Req&Wait for Conn)

The PC_C5:PC_C3 transition shall occur if the ULP Open timeout timer has not reached the specified limit and if the link layer is in the SL0:Idle state.

8.4.5.3 Transition PC_C5:PC_C6 (Set up for Conn Retry:Cannot connect)

The PC_C5:PC_C6 transition shall occur if the ULP Open timeout timer has reached the specified limit.

8.4.6 PC_C6:Cannot connect state

8.4.6.1 State definition

This state shall request the link layer perform a “Stop Arb” function, set up the Connection Status to either Arb Lost – No Retry or Arb Lost -ULP Open time out exceeded to return to the transport layer ,and then wait for the link layer to enter the SL0:Idle state.

8.4.6.2 Transition PC_C6:PC_C1 (Cannot connect:Not connected)

The PC_C6:PC_C1 transition shall occur when the link layer enters the SL0:Idle state.

8.4.7 PC_C7:Connected but cannot transmit frames (Conn but cannot Tx) state

8.4.7.1 State definition

This state is entered only during SSP connections. It indicates that a DONE has been received and that this port cannot transmit any additional frames until the connection has been closed and reestablished. While in this state, the device may transmit primitives (i.e, ACK’s, NAK’s, RRDY’s, etc) and receive IU’s and primitives.

8.4.7.2 Transition PC_C7:PC_C8 (Conn but cannot Tx:Wait for close)

The PC_C7:PC_C8 transition shall occur when the link layer SL4:Connected state has been exited.

8.4.8 PC_C8:Wait for Close state

8.4.8.1 State definition

This state is entered only during SSP connections. This state indicates that a DONE has been transmitted and received by the Initiator(or Target) or that a CLOSE has been transmitted or received and that this device cannot transmit or receive anything until the connection has been closed and reestablished.

8.4.8.2 Transition PC_C8:PC_C1 (Wait for Close; Not connected)

The PC_C8:PC_C1 transition shall occur when the link layer indicates that the connection has been closed.

8.4.9 PC_C9:Error terminate transmit and receive (Error Term Tx&Rcv) state

8.4.9.1 State definition

This state indicates that a BREAK has been either transmitted and received by the link layer and that this device cannot transmit anything or receive anything until the connection has been closed and reestablished.

8.4.9.2 Transition PC_C9:PC_C1 (Error Term Tx&Rcv:Not connected)

The PC_C8:PC_C1 transition shall occur when the link layer indicates that the connection has been closed.

9 Transport layer

9.1 Transport layer overview

The transport layer constructs and parses frame contents.

The transport layer only receives frames that were ACKed by the link layer.

9.2 SSP transport layer

9.2.1 SSP frame format

Table 53 defines the SSP frame format.

Table 53. General frame format

Byte	7	6	5	4	3	2	1	0
0	INFORMATION UNIT TYPE							
1	(MSB)							
2				HASHED DESTINATION DEVICE NAME				
3								(LSB)
4	Reserved							
5	(MSB)							
6				HASHED SOURCE DEVICE NAME				
7								(LSB)
8	Reserved							
9	Reserved							
10	Reserved						TIMEOUT	Rsvd
11	Reserved						NUMBER OF FILL BYTES	
12	COMMAND ID							
13				Reserved				
15								
16	(MSB)							
17				TAG				(LSB)
18	(MSB)							
19				TARGET PORT TRANSFER TAG				(LSB)
20				Reserved				
23								
24				INFORMATION UNIT				
m								
	Fill bytes, if needed							
n - 3	(MSB)							
n				CRC				(LSB)

Table 54 defines the INFORMATION UNIT TYPE field.

Table 54. INFORMATION UNIT TYPE field

INFORMATION UNIT TYPE	Name	Information unit	Originator	Size of INFORMATION UNIT field
01h	DATA	Data	Initiator port or Target port	0 to 1024 bytes
03h	XFER_RDY	Transfer ready	Target port	12 bytes
06h	COMMAND	Command	Initiator port	28 to 284 bytes
07h	RESPONSE	Response	Target port	24 to 1024 bytes
16h	TASK	Task management function	Initiator port	32 bytes
20h	AEN	Asynchronous event notification	Target port	8 bytes
21h	AEN_RESPONSE	Asynchronous event notification response	Initiator port	4 bytes

A frame containing a COMMAND information unit is called a COMMAND frame; a frame containing a TASK information unit is called a TASK frame; etc.

The HASHED DESTINATION DEVICE NAME field contains the hashed value (see 4.3.3) of the destination device name. The recipient may compare this to its own device name to verify that the frame was delivered to the correct destination. If the HASHED DESTINATION DEVICE NAME does not match the device name used to open the connection, the frame may be discarded.

The HASHED SOURCE DEVICE NAME fields contains the hashed value of the source device name. The recipient may compare this to the device name which opened the connection to verify the frame was from the correct source. If the HASHED SOURCE DEVICE NAME does not match the device name used to open the connection, the frame may be discarded.

The TIMEOUT bit may be set to one for RESPONSE information unit types and shall be set to zero for all other types. This field indicates the frame is a retransmission after the target port timed out waiting for the ACK or NAK for its previous attempt to send the frame.

The NUMBER OF FILL BYTES field indicates the number of fill bytes between the INFORMATION UNIT field and the CRC field. The initiator port or target port sending a frame shall provide fill bytes to ensure that the CRC field is 4-byte aligned.

The initiator port shall set the COMMAND ID field for COMMAND and TASK information unit types to a value that is unique across any two sequential commands using the same I_T_L_Q nexus (i.e., re-using the same tag). The initiator port shall set the COMMAND ID field to zero for DATA and AEN_RESPONSE frames. ~~The target port shall include this value in frames for that I_T_L_Q containing XFER_RDY, DATA, and RESPONSE information unit types.~~ The target port shall include the value from the COMMAND or TASK frame in XFER_RDY, DATA, and RESPONSE frames for the same I_T_L_Q. The target port shall set the COMMAND ID field to zero for AEN frames. Although the field is 8 bits wide, initiators may implement it by toggling a single bit; more than one command or task management request~~s~~ should not be outstanding except in an error situation.

The TAG field allows the initiator port to establish a context for commands and task management functions.

For ~~the~~ COMMAND and TASK ~~information unit types~~frames, the initiator port shall set the TAG field to a value that is ~~assigned by the initiator port shall be~~ unique for the target port. The same tag shall not be reused when sending COMMAND or TASK frames to different LUNs in the same target port. The same tag may be reused when sending frames to different target ports. The TAG field in a COMMAND IU contains the tag defined in SAM-2; the TAG field in a TASK IU does not correspond to a SAM-2 tag.

For ~~the~~ DATA, XFER_RDY, and RESPONSE~~ information unit types~~ frames, the target port shall set the TAG field ~~indicates~~ to that of the command or task management function to which the ~~information unit~~ frame pertains.

For the AEN and AEN_RESPONSE frames, the TAG field shall be set to zero.

The TARGET PORT TRANSFER TAG field allows the target port to quickly establish a write data context when it has multiple outstanding XFER_RDY IUs. This field shall be set by the target port in the XFER_RDY IU. Target ports that need this field shall set a value that is unique for the I_T nexus. Target ports that do not need this field shall

set it to FFFFh. For each DATA IU that is sent in response to a XFER_RDY IU, the initiator port shall set the TARGET PORT TRANSFER TAG field to the value that was in the corresponding XFER_RDY IU. For each DATA IU that is not sent in response to a XFER_RDY IU (due to a non-zero FIRST BURST SIZE field in the Disconnect-Reconnect mode page), the initiator port shall set the TARGET PORT TRANSFER TAG field to FFFFh. The initiator port shall set this field to FFFFh for all IUs other than DATA_IUs. The target port shall set this field to FFFFh for all IUs other than XFER_RDY IUs.

The INFORMATION UNIT field contains ~~command, task management function, response, data, or transfer ready~~the information unit. The maximum size of the INFORMATION UNIT field is 1 024 bytes, making the maximum size of the frame 1 052 bytes (1 024 byte data + 24 byte header + 4 byte CRC).

The CRC field is a CRC value (see 7.10) that is computed over the entire frame including the fill bytes.

Target ports shall check the CRC before processing a frame containing a COMMAND or TASK IU. Initiator ports shall check the CRC before processing a frame containing a RESPONSE IU. CRC checking is done by the link layer, not the transport layer.

[Editor's note: since CRC checking and possibly hashed device name checking is done at the link layer, the basic frame format may fit better in the SSP link layer section.]

9.2.2 Information unit sequences

Initiator ports may send COMMAND and TASK IUs unsolicited.

Target ports shall only send XFER_RDY IUs while processing a write or bidirectional command. Initiator ports shall only send DATA IUs in response to an XFER_RDY IU. Target ports ~~may~~ shall not send an XFER_RDY IU before receiving all write data for the previous XFER_RDY IU.

~~[Editor's note: on wide links (with a wide initiator and a wide target), require the TTT be different on each outstanding XFER_RDY? Wide link hopping rules need more consideration in general. If a command is sent down one physical link, can the XFER_RDY, DATA, and RESPONSE IUs be returned on another physical link? The current proposal is: an I_T_L_Q nexus is only allowed on one physical link at a time. It may hop to another physical link whenever the outstanding ACK/NAK count goes to zero.]~~

Target ports shall only send DATA IUs while processing a read or bidirectional command.

Target ports shall only send RESPONSE IUs in response to a COMMAND or TASK IU after all required data transfers have been performed.

Target ports may send an AEN IU when an asynchronous event occurs (see 9.2.4). Initiator ports shall send an AEN_RESPONSE IU in response to each AEN IU.

Figure 70 shows a task management sequence.

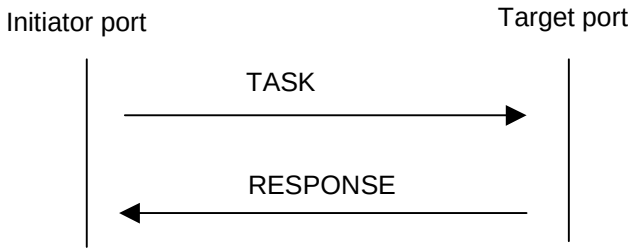

Figure 70. Task management sequence

Figure 71 shows a write command sequence.

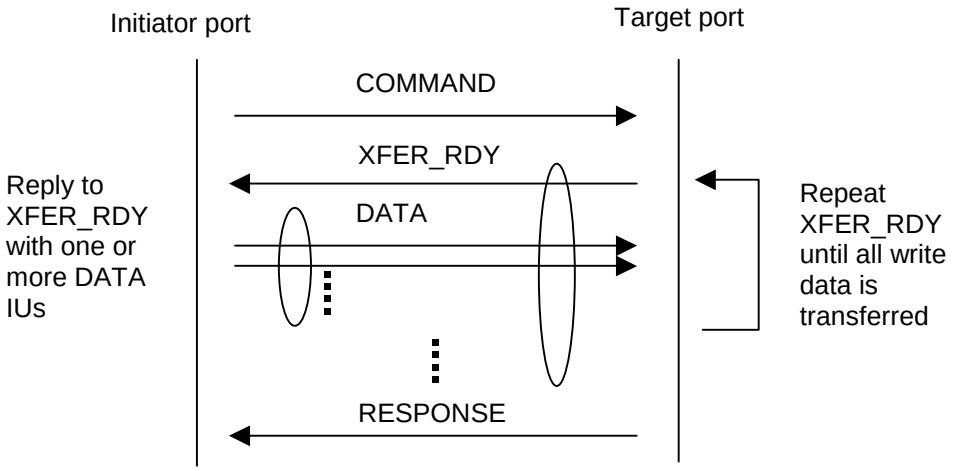

Figure 71. Write sequence

Figure 72 shows a read command sequence.

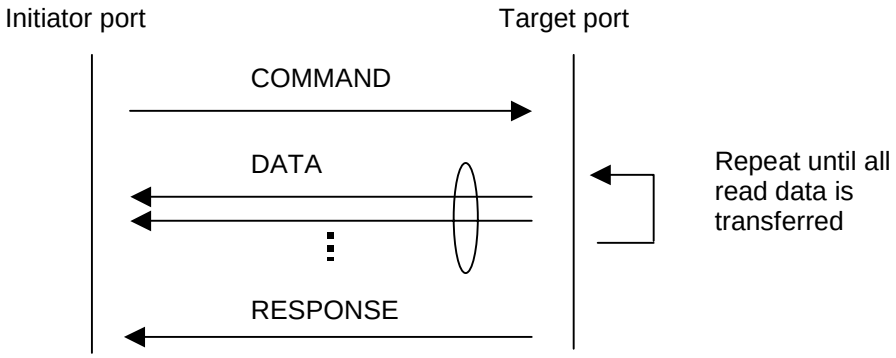

Figure 72. Read sequence

Figure 73 shows a bidirectional command sequence.

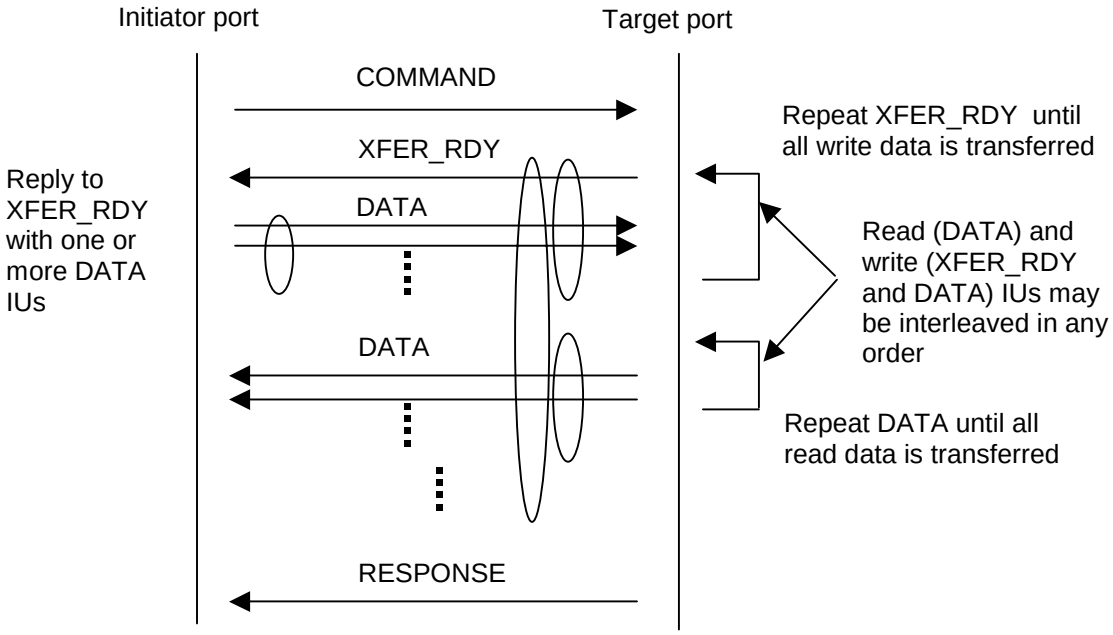

Figure 73. Bidirectional sequence

Figure 74 shows an asynchronous event notification sequence.

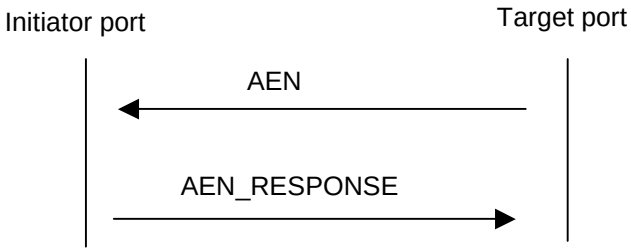

Figure 74. Asynchronous event notification sequence

9.2.3 Information units

9.2.3.1 COMMAND information unit

Table 55 defines the command IU. The COMMAND IU is sent by an initiator port to request a command be performed by a device server in a logical unit.

Table 55. COMMAND information unit

Byte	7	6	5	4	3	2	1	0
0	(MSB)							
7				LOGICAL UNIT NUMBER				(LSB)
8				Reserved				
9				Reserved			TASK ATTRIBUTE	
10				Reserved				
11				ADDITIONAL CDB LENGTH (n dwords)			Reserved	
12								
27				CDB				
28								
27+n/4				ADDITIONAL CDB BYTES				

The LOGICAL UNIT NUMBER field specifies the address of the logical unit. The structure of the logical unit number field shall be as defined in SAM-2. If the addressed logical unit does not exist, the task manager shall follow the SCSI rules for selection of invalid logical units defined in SPC-2.

The TASK ATTRIBUTE field is defined in Table 56.

Table 56. TASK ATTRIBUTE field

TASK ATTRIBUTE	Task attribute	Description
000b	SIMPLE	Requests that the task be managed according to the rules for a simple task attribute (see SAM-2).
001b	HEAD OF QUEUE	Requests that the task be managed according to the rules for a head of queue task attribute (see SAM-2).
010b	ORDERED	Requests that the task be managed according to the rules for an ordered attribute (see SAM-2).
011b	Reserved	
100b	ACA	Requests that the task be managed according to the rules for an automatic contingent allegiance task attribute (see SAM-2).
101b-111b	Reserved	

The ADDITIONAL CDB LENGTH field contains the length in 4-byte dwords 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 reserved.

The contents of the CDB are defined in the SCSI command standards (e.g., SPC-3).

9.2.3.2 TASK information unit

Table 57 defines the task management function IU. The TASK IU is sent by an initiator port to request a task management function be performed by a task manager in a logical unit.

Table 57. TASK information unit

Byte	7	6	5	4	3	2	1	0
0	(MSB)							
7				LOGICAL UNIT NUMBER				(LSB)
8				Reserved				
9				Reserved				
10				TASK MANAGEMENT FUNCTION				
11				Reserved				
12	(MSB)							
13				TAG OF TASK TO BE MANAGED				(LSB)
14								
31				Reserved				

The LOGICAL UNIT NUMBER field specifies the address of the logical unit. The structure of the logical unit number field shall be as defined in SAM-2. If the addressed logical unit does not exist, the task manager shall follow the SCSI rules for selection of invalid logical units defined in SPC-2.

Table 58 defines the TASK MANAGEMENT FUNCTION field.

Table 58. Task management functions

TASK MANAGEMENT FUNCTION	Task management function	Description
01h	ABORT TASK	The task manager shall perform the ABORT TASK task management function with L set to LOGICAL UNIT NUMBER and Q set to TAG OF TASK TO BE MANAGED (see SAM-2).
02h	ABORT TASK SET	The task manager shall perform the ABORT TASK SET task management function with L set to LOGICAL UNIT NUMBER (see SAM-2).
04h	CLEAR TASK SET	The task manager shall perform the CLEAR TASK SET task management function with L set to LOGICAL UNIT NUMBER (see SAM-2).
08h	LOGICAL UNIT RESET	The task manager shall perform the LOGICAL UNIT RESET task management function with L set to LOGICAL UNIT NUMBER (see SAM-2).
20h	Reserved	
40h	CLEAR ACA	The task manager shall perform the CLEAR ACA task management function with L set to LOGICAL UNIT NUMBER (see SAM-2).
80h	QUERY TASK	The task manager shall perform the QUERY TASK task management function with L set to LOGICAL UNIT NUMBER and Q set to TAG OF TASK TO BE MANAGED (see SAM-2).
All others	Reserved	

If TASK MANAGEMENT FUNCTION contains a reserved value, the task manager shall return a RESPONSE IU with its STATUS field set to GOOD and its RSP_CODE field set to TASK MANAGEMENT FUNCTION NOT SUPPORTED. The TARGET RESET task management function defined in SAM-2 is not supported.

If TASK MANAGEMENT FUNCTION is set to ABORT TASK or QUERY TASK, the TAG OF TASK TO BE MANAGED field specifies the TAG value from the command IU that contained the task to be aborted or checked. For all other task management functions, the TAG OF TASK TO BE MANAGED field shall be ignored.

For ABORT TASK, if the TAG OF TASK TO BE MANAGED field does not contain the tag of a task currently in the task set, the task manager shall return a RESPONSE IU with its RSP_CODE field set to TASK MANAGEMENT FUNCTION COMPLETE.

For QUERY TASK, if the TAG OF TASK TO BE MANAGED field does not contain the tag of a task currently in the task set, the task manager shall return a RESPONSE IU with its RSP_CODE field set to TASK MANAGEMENT FUNCTION FAILED.

9.2.3.3 XFER_RDY information unit

Table 59 defines the transfer ready IU. The XFER_RDY IU is sent by a target port to request write data from the initiator port.

Table 59. XFER_RDY information unit

Byte	7	6	5	4	3	2	1	0
0	Reserved							
3								
4	(MSB)	WRITE DATA LENGTH						
7								(LSB)
8	Reserved							
11								

The WRITE DATA LENGTH field indicates how many bytes of write data the initiator port may send to the logical unit (using DATA IUs). If the value in the MAXIMUM BURST SIZE field in the Disconnect-Reconnect mode page is not zero, the value in the WRITE DATA LENGTH field is constrained by the value in the MAXIMUM BURST SIZE field (see 10.1.1.1.4).

9.2.3.4 DATA information unit

Table 60 defines the data IU. The DATA IU is sent by an initiator port to deliver write data and is sent by a target port to deliver read data.

Table 60. DATA information unit

Byte	7	6	5	4	3	2	1	0
0	DATA							
n-1								

The DATA field contains the read or write data. The maximum size of the data IU is the maximum size of any IU in an SSP frame (see 9.2.1).

An initiator port shall only send a DATA IU:

a) in response to an XFER_RDY IU; or
b) after sending a COMMAND IU if the FIRST BURST SIZE field in the Disconnect-Reconnect mode page is not zero (see 10.1.1.1.5).

If the value in the MAXIMUM BURST SIZE field on the Disconnect-Reconnect mode page is not zero, the maximum amount of data that is transferred at one time by a target port per I_T_L_Q nexus is constrained by the value in the MAXIMUM BURST SIZE field (see 10.1.1.1.4).

The DATA IU shall only contain write data for a single XFER_RDY IU.

9.2.3.5 RESPONSE information unit

Table 61 defines the response IU. The RESPONSE IU is sent by a target port to deliver SCSI status (GOOD, CHECK CONDITION, etc.) and sense data, or to deliver SSP-specific status (illegal IU format, etc.).

Table 61. RESPONSE information unit

Byte	7	6	5	4	3	2	1	0
0	Reserved							
9								
10	Reserved						SNSVALID	RSPVALID
11	STATUS							
12	Reserved							
15								
16	(MSB) SENSE DATA LIST LENGTH (n bytes)							
19								(LSB)
20	(MSB) RESPONSE DATA LIST LENGTH (m bytes)							
23								(LSB)
24	RESPONSE DATA							
23+m								
24+m	SENSE DATA							
24+m+n								

A SNSVALID bit of one indicates the contents of the SENSE DATA LIST LENGTH field is valid and the SENSE DATA field is present. A SNSVALID bit of zero indicates the SENSE DATA LIST LENGTH field is not valid and the SENSE DATA field is not present.

If sense data is provided, SNSVALID shall be set to one and the SENSE DATA LIST LENGTH field shall indicate the number of bytes in the SENSE DATA field. The SENSE DATA LIST LENGTH field shall only contain lengths that are multiples of four.

If no sense data is provided, SNSVALID shall be set to zero. The application client shall ignore the SENSE DATA LIST LENGTH field and shall assume the SENSE DATA field has a length of zero.

If the SNSVALID bit is set to one, the SENSE DATA field contains sense data (see SAM-2). If the RSPVALID bit is set to one, the contents of the SENSE DATA field are not reliable and shall be ignored by the application client.

A RSPVALID bit of one indicates the contents of the RESPONSE DATA LIST LENGTH field is valid and the RESPONSE DATA field is present. A RSPVALID bit of zero indicates the RESPONSE DATA LIST LENGTH field is not valid and the RESPONSE DATA field is not present.

If response data is provided, RSPVALID shall be set to one and the RESPONSE DATA LIST LENGTH field shall be set to four. Other lengths are reserved for future standardization.

If no response data is provided, RSPVALID shall be set to zero. The application client shall ignore the RESPONSE DATA LIST LENGTH field and shall assume the RESPONSE DATA field has a length of zero.

No combination of valid SENSE DATA LIST LENGTH fields and RESPONSE DATA LIST LENGTH fields shall cause the IU to exceed the maximum IU size for SSP frames (see 9.2.1).

Response data shall be provided in each response IU that is sent in response to an task management function IU. The information in the RSP_CODE field shall indicate the completion status of the task management function. Response data may be provided in other response IUs. When response data is provided, the STATUS field and SENSE DATA fields are invalid.

The STATUS field contains the status of a task that completes. See SAM-2 for a list of status codes. If the RSPVALID bit is set to one, the contents of the STATUS field are not reliable and shall be ignored by the application client.

Table 62 defines the RESPONSE DATA field, which contains information describing certain protocol failures detected during processing of a request received by the target port. The RESPONSE DATA field shall be present if the target port detects any of the conditions described by a non-zero RSP_CODE value.

Table 62. RESPONSE DATA field

Byte	7	6	5	4	3	2	1	0
0	Reserved							
1	Reserved							
2	Reserved							
3	RSP_CODE							

Table 63 defines the RSP_CODE field.

Table 63. RSP_CODE field

RSP_CODE	Description
00h	Either: a) NO FAILURE, when responding to a COMMAND IU; or b) TASK MANAGEMENT FUNCTION COMPLETE, when responding to a TASK IU.
02h	COMMAND OR TASK FIELDS INVALID
04h	TASK MANAGEMENT FUNCTION NOT SUPPORTED
05h	TASK MANAGEMENT FUNCTION FAILED
All others	Reserved

9.2.3.6 AEN information unit

Table 64 defines the asynchronous event notification IU. The AEN IU is sent by a target port to notify an initiator port of an asynchronous event. The target port shall not send another AEN IU to the initiator port until the previous one is responded to with an AEN_RESPONSE IU.

Table 64. AEN information unit

Byte	7	6	5	4	3	2	1	0
0	(MSB)	LOGICAL UNIT NUMBER						
7								(LSB)

The LOGICAL UNIT NUMBER field identifies which logical unit is reporting an asynchronous event.

9.2.3.7 AEN_RESPONSE information unit

Table 65 defines the asynchronous event notification response IU. The AEN_RESPONSE IU is sent by an initiator port in response to an AEN IU from a target port. AEN_RESPONSE grants the target port permission to send another AEN IU to the initiator port.

Table 65. AEN_RESPONSE information unit

Byte	7	6	5	4	3	2	1	0
0	Reserved							
3								

The LOGICAL UNIT NUMBER field identifies which logical unit is reporting an asynchronous event.

9.2.4 Asynchronous event reporting

SAS SSP target ports shall send an AEN IU when an enabled asynchronous event has occurred in one of their logical units. Asynchronous events are defined in SAM-2 and are enabled by the ready AER (RAERP), unit attention AER (UAAERP), and deferred error AER (EAERP) bits in the Control mode page of SPC-3. SAS SSP target ports shall not support the RAERP bit but may support the UAAERP and EAERP bits.

The AEN IU includes the logical unit number. If the initiator port saves this value, it should send a REQUEST SENSE command to the logical unit to retrieve the sense data associated with the event.

If the initiator port does not save this value, it should send a REPORT LUNS command to the REPORT LUNS well known logical unit in the target port to retrieve a list of logical units which have asynchronous events pending. It should then send a TEST UNIT READY or REQUEST SENSE command to each logical unit to retrieve the sense data associated with the asynchronous events.

If the initiator port already has commands pending at the logical unit, it may not need to send an extra command; the sense data will be reported on the next command completing with GOOD status.

Logical units behind SSP target ports shall not support the RAERP, UAAERP, and EAERP bits in the control mode page (see SPC-3). SSP target ports shall send a single AEN primitive whenever any of the conditions described by those bits occurs in any of their logical units.

9.2.5 SSP transport layer handling of link layer errors

9.2.5.1 COMMAND frame

If an initiator port sends a COMMAND frame and times out waiting for ACK or NAK it shall close the connection with DONE(ACK/NAK TIMEOUT) and request a new connection to that target port. It shall send a QUERY TASK task management function to determine whether the command was received or not. If QUERY TASK returns a TASK MANAGEMENT FUNCTION COMPLETE response, the initiator port shall assume the command was ACKed. If QUERY TASK returns a TASK MANAGEMENT FUNCTION FAILED response, and a RESPONSE frame has not yet been received for that I_T_L_Q, the initiator port shall assume the command was NAKed or lost and may reuse the tag.

9.2.5.2 DATA frame

If a target port sends a DATA frame and does not receive an ACK or NAK, it shall close the connection with DONE(ACK/NAK TIMEOUT) and return a CHECK CONDITION status for that command with a sense key of ABORTED COMMAND and an additional sense code of ACK/NAK TIMEOUT.

If a target port sends a DATA frame and receives a NAK, it shall return a CHECK CONDITION status for that command with a sense key of ABORTED COMMAND and an additional sense code of NAK RECEIVED.

9.2.5.3 TASK frame

If an initiator port sends a TASK frame and times out waiting for ACK or NAK it shall close the connection with DONE(ACK/NAK TIMEOUT) and request a new connection to that target port. It shall resend the TASK frame.

9.2.5.4 RESPONSE frame

If a target port sends a RESPONSE frame and does not receive an ACK or NAK, it shall close the connection with DONE(ACK/NAK TIMEOUT), open a new connection, and try sending the RESPONSE frame again. It shall do this at least one time. The TIMEOUT bit shall be set to one on each of the retries. The target port shall send no other frames for that I_T nexus until the RESPONSE frame is successfully sent or abandoned.

If an initiator port receives a RESPONSE frame with a TIMEOUT bit of one, and it has previously ACKed the RESPONSE frame (based on the COMMAND_ID field and the I_T_L_Q), it shall ACK the retry and discard the extra RESPONSE frame. If it has not previously ACKed the RESPONSE frame, it shall ACK or NAK it and use it appropriately.

9.2.5.5 AEN frame

If an target port sends an AEN frame and times out waiting for ACK or NAK it shall close the connection with DONE(ACK/NAK TIMEOUT) and request a new connection to that target port. It shall resend the AEN frame.

9.2.5.6 AEN_RESPONSE frame

If an initiator port sends an AEN_RESPONSE frame and does not receive an ACK or NAK it shall close the connection with DONE(ACK/NAK TIMEOUT), open a new connection, and try sending the AEN_RESPONSE frame again. It shall do this at least one time.

9.2.6 SSP transport layer error handling

9.2.6.1 General error handling

If a port receives a frame shorter than the minimum frame size (28 bytes), it shall discard the frame.

If a port receives a frame that is too small for the indicated IU type, it shall discard the frame.

9.2.6.2 Target port error handling

[Editor's note: there are several new additional sense codes here. See the SPC-3 requested changes annex.]

If a target port receives a COMMAND frame that is too short to contain a LUN field, it shall discard the frame.

If a target port receives a COMMAND frame that contains a LUN field but is too small to contain a CDB, or the ADDITIONAL CDB LENGTH field indicates the frame should be longer than it is, the target port shall return a CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code of INFORMATION UNIT TOO SHORT.

If a target port receives a COMMAND frame that the ADDITIONAL CDB LENGTH field indicates the frame should be shorter than it is, the target port shall return a CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code of INFORMATION UNIT TOO LONG.

If a target port receives a TASK frame that is too short, it shall return TASK MANAGEMENT FUNCTION FAILED.

If a target port receives a COMMAND frame or TASK frame with a TAG already in use, it may respond as defined in SAM-2.

If a target port receives a DATA frame with an unknown TARGET PORT TRANSFER TAG, it shall terminate the command with a CHECK CONDITION status with a sense key of ABORTED COMMAND and an additional sense code of ILLEGAL TARGET PORT TRANSFER TAG RECEIVED.

If a target port receives a DATA frame with an unknown TAG, it shall ~~ignore~~ discard the frame.

If a target port receives a DATA frame with more write data than expected, it shall ~~ignore~~ discard the frame and terminate the command with a CHECK CONDITION status with a sense key of ABORTED COMMAND and an additional sense code of TOO MUCH WRITE DATA.

9.2.6.3 Initiator port error handling

If an initiator port receives a AEN_RESPONSE, COMMAND, or TASK frame , or a target port receives an AEN or XFER_RDY frame, it shall discard the frame.

If an initiator port receives a DATA, XFER_RDY, or RESPONSE frame with an unknown TAG, it shall discard the frame. It may send an ABORT TASK or ABORT TASK SET.

If an initiator port receives an XFER_RDY frame in response to a command with no write data, it shall discard the frame. It may then send an ABORT TASK request to abort the command.

If an initiator port receives an XFER_RDY frame requesting more write data than expected, it shall send an ABORT TASK to abort the command.

If an initiator port receives a DATA frame with more read data than expected, it shall discard the frame and send an ABORT TASK to abort the command. It may receive a RESPONSE for the command before being able to abort it.

If an initiator port receives a RESPONSE frame larger than indicated by the frame definition, it shall discard the frame.

If an initiator port receives an XFER_RDY frame that is not 8 bytes long, it shall discard the frame.

9.2.7 SSP transport layer state diagrams

[Editor's note: Data-out (write) commands, bidirectional commands, task management functions, and AENs all need to be added.]

9.2.7.1 Initiator device state machine

9.2.7.1.1 State machine overview

Figure 75 describes the initiator device state machine.

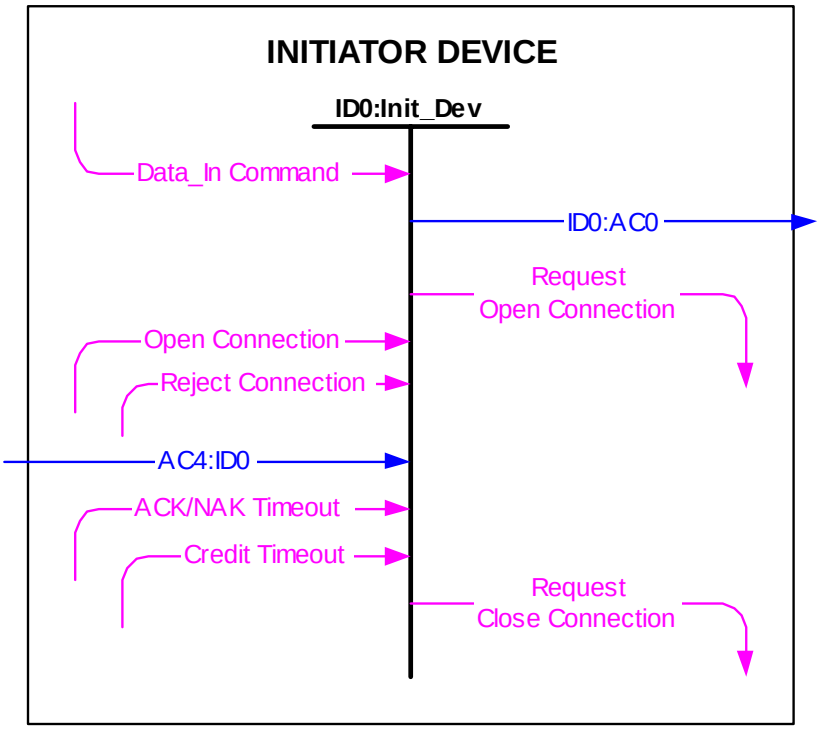

Figure 75. SSP initiator device state machine

If the initiator device receives a task from an application to fetch data from a target, an application client shall be created to process the task. The request for the task shall consist of the target identifier (i.e., WWN), LUN, CDB, and task attribute.

While in this state the initiator device may request that the port layer open a connection with a target device in order to transmit a command frame. Included in the request shall be the protocol (i.e., SSP), the link rate, and the WWN of the target.

The initiator device may be informed by the port layer that a connection has been opened, rejected, or closed because of a timeout. The initiator may request that the connection be re-opened at a later time.

The initiator device may request that the port layer close an open connection for a vendor-specific reason (e.g., there are no more outstanding tasks or an application client has reported that a COMMAND frame transmission has failed).
This state machine goes idle after the application client has been created.

9.2.7.1.1.1 Parameter ID0:AC0 (Init_Dev to App_Client Process_Cmd)

The I0:AC0 parameter shall be sent when an application client has been created to process the data_in task.

9.2.7.2 Application client state machine

9.2.7.2.1 State machine overview

Figure 76 describes the application client state machine.

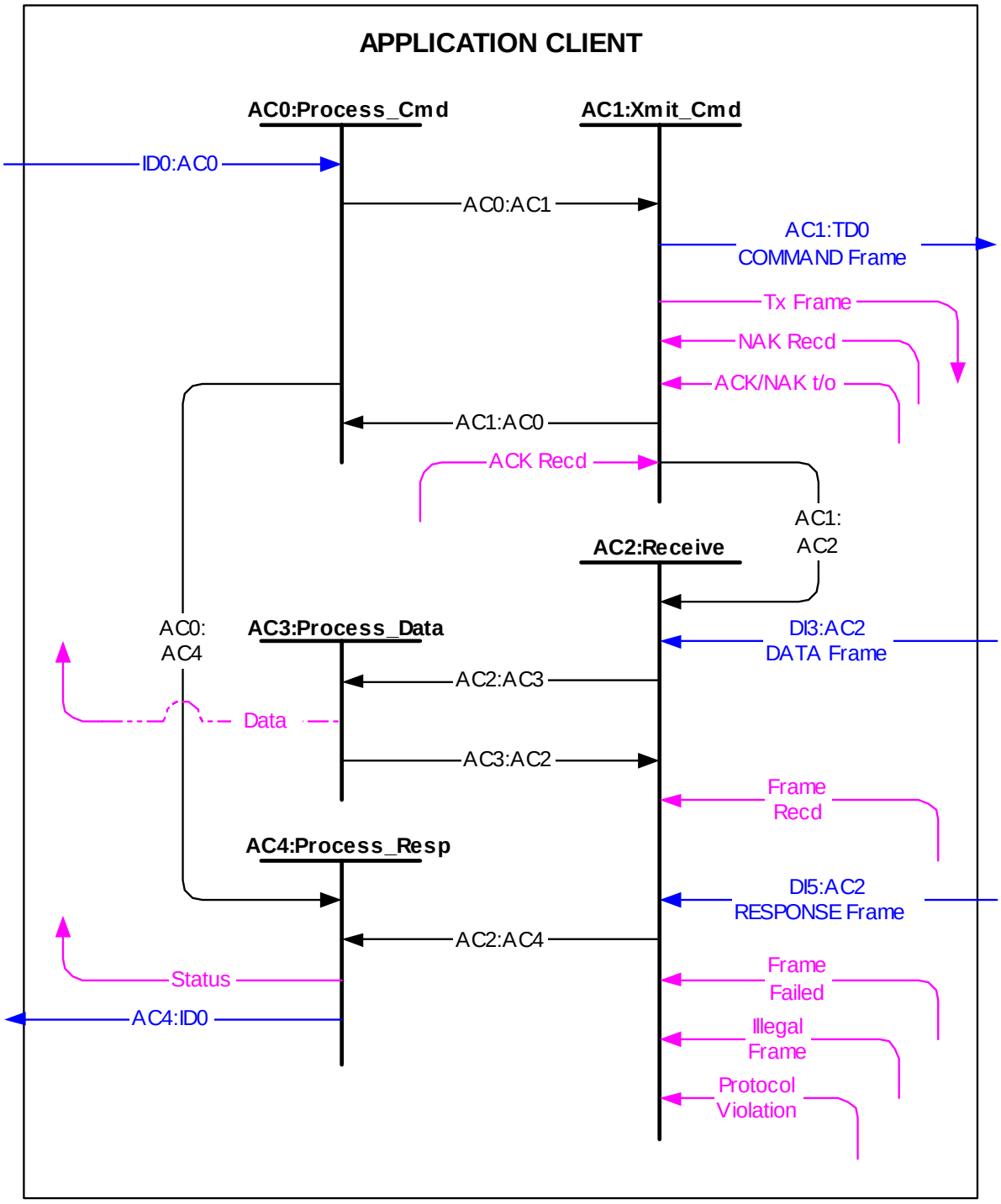

Figure 76. SSP application client state machine

9.2.7.2.2 AC0:Process_Cmd state

9.2.7.2.2.1 AC0:Process_Cmd state description

If this state was entered as a result of an application client being created to process a task, the application client builds a COMMAND frame and passes this frame to the initiator port. The COMMAND frame shall contain the DESTINATION CHECK data, SOURCE CHECK data, CMD_ID, tag, LUN, CDB, and task attribute.

If this state was entered as the result of the application client being informed by the initiator port that an error occurred during transmission of a previous COMMAND frame, the application client may pass the COMMAND frame to the initiator port again, or it may transition to prepare to send status to the application.

This state is exited after the COMMAND frame has been passed to the initiator port or as the result of an error to be reported to the application.

9.2.7.2.2.2 Transition AC0:AC1 (App_Client Process_Cmd to App_Client Xmit_Cmd)

The AC0:AC1 transition shall occur after the application client has passed the COMMAND frame to the initiator port.

9.2.7.2.2.3 Transition AC0:AC4 (App_Client Process_Cmd to App_Client Process_Resp)

The AC0:AC4 transition shall occur after the application client has determine that it has command transmission status to send to the application.

9.2.7.2.3 AC1:Xmit_Cmd state

9.2.7.2.3.1 AC1:Xmit_Cmd state description

In this state the initiator port sends the COMMAND frame to the target port or reports to the application client that an error occurred during transmission of the COMMAND frame.

The initiator port shall notify the link layer that the frame is to be transmitted and that the frame is interlocked.

The link layer informs the initiator port if the frame was transmitted with or without error (i.e., whether a NAK or ACK was received for the frame). The initiator port informs the application client of the transmission status.

While in this state the link layer may inform the application client that the connection has been terminated as the result of an ACK/NAK timeout or a Credit timeout. The application client may attempt to send the COMMAND frame for the task during a subsequent connection.

This state is exited after the initiator port sends the COMMAND frame to the target port and/or has informed the application client of the transmission status.

9.2.7.2.3.2 Parameter AC1:TD0 (App_Client Xmit_Cmd to Tgt_Dev)

The initiator port shall send the COMMAND frame to the target port.

9.2.7.2.3.3 Transition AC1:AC0 (App_Client Xmit_Cmd to App_Client Process_Cmd)

The AC1:AC0 transition shall occur after the initiator port has informed the application client that an error occurred during transmission of the COMMAND frame.

9.2.7.2.3.4 Transition AC1:AC2 (App_Client Xmit_Cmd to App_Client Receive)

The AC1:AC2 transition shall occur after the initiator port has informed the application client that the COMMAND frame was transmitted without error.

9.2.7.2.4 AC2:Receive state

9.2.7.2.4.1 AC2:Receive state description

In this state the initiator port receives a DATA or RESPONSE frame from the target port. The link layer informs the initiator port whether the frame was received with or without error. If the frame was received without error, the initiator port shall pass the frame to the application client. If the frame was received with error, the initiator port shall pass the error information to the application client.

This state is exited when the initiator port has passed the frame to the application client, or the initiator port has notified the application client of the error.

9.2.7.2.4.2 Transition AC2:AC3 (App_Client Receive to App_Client Process_Data)

The AC2:AC3 transition shall occur after the initiator port has passed a DATA frame to the application client.

9.2.7.2.4.3 Transition AC2:AC4 (App_Client Receive to App_Client Process_Resp)

The AC2:AC4 transition shall occur after the initiator port has passed a RESPONSE frame to the application client.

9.2.7.2.5 AC3:Process_Data state

9.2.7.2.5.1 AC3:Process_Data state description

In this state the application client processes the DATA frame and informs the application.
This state is exited when the application client has processed the DATA frame.

9.2.7.2.5.2 Transition AC3:AC2 (App_Client Process_Data to App_Client Receive)

The AC3:AC2 transition shall occur after the application client has processed a DATA frame.

9.2.7.2.6 AC4:Process_Resp state

9.2.7.2.6.1 AC4:Process_Resp state description

In this state the application client passes status to the application.
This state is exited when the application client has passed status to the application.

9.2.7.2.6.2 Parameters AC4:ID0 (App_Client Process_Resp to Initiator Device Idle)

The AC4:ID0 signal shall occur after the application client has passed status to the application.

9.2.7.3 Target device state machine

9.2.7.3.1 State machine overview

Figure 77 defines the target device state machine.

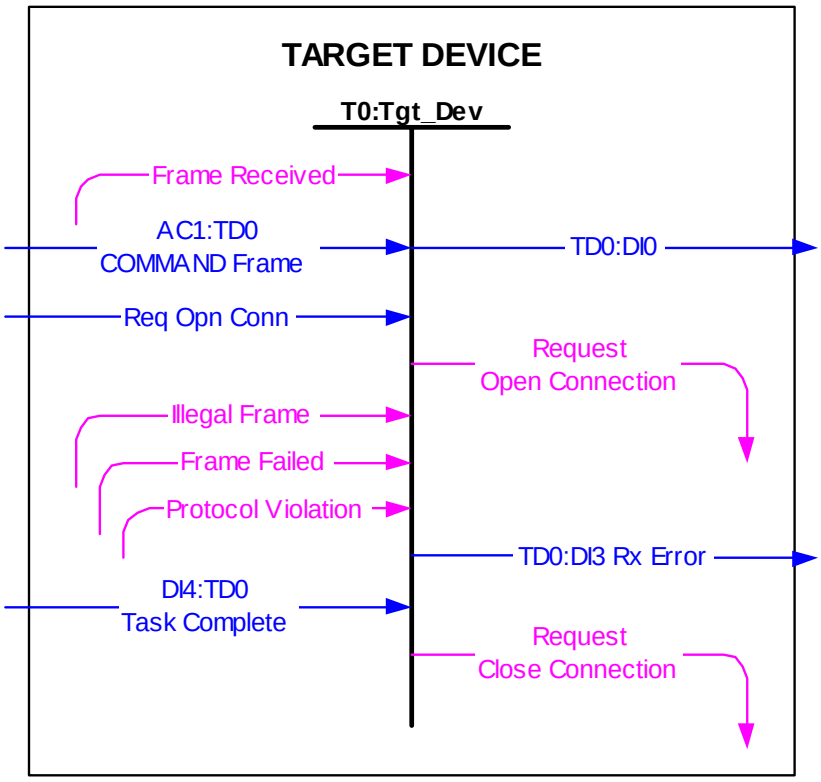

Figure 77. SSP target device state machine

9.2.7.3.2 TD0:Tgt_Dev state

9.2.7.3.2.1 TD0:Tgt_Dev state description

In this state the target port receives a COMMAND frame from the initiator port. The link layer informs the target port whether the frame was received with or without error. The target port may also request that a connection be open or closed (e.g., as the result of having no frames to send or a time described in the disconnect-reconnect page expiring).

If the frame was received without error, the target port shall pass the COMMAND frame to the task router. The task router shall check the DESTINATION CHECK data and SOURCE CHECK data. If this data is correct the task router shall pass the tag, task attribute, and CDB to the task manager of the logical unit specified in the COMMAND information unit.

If an error occurred while receiving the frame, and there is sufficient data to report the error, the task router shall inform the device server of the error.

This state is exited when the task router has passed the tag, task attribute, and CDB to the task manager of the logical unit specified in the COMMAND information unit, or the device server has been informed of an error.

9.2.7.3.2.2 Parameters TD0:DI0 (Tgt_Dev to DI_Task Process_Cmd)

The TD0:DI0 transition shall occur after the task router has passed the tag, task attribute, and CDB to the task manager of the logical unit specified in the COMMAND information unit.

9.2.7.3.2.3 Parameters TD0:DI3 (Tgt_Dev to DI_Task Prep_Resp)

The TD:DI3 transition shall occur when the task router detects that an error has occurred receiving a COMMAND frame, there is sufficient information for a RESPONSE frame to be generated, and this information has been passed to the device server.

9.2.7.4 Data in task state machine

9.2.7.4.1 State machine overview

Figure 78 defines the data in task state machine.

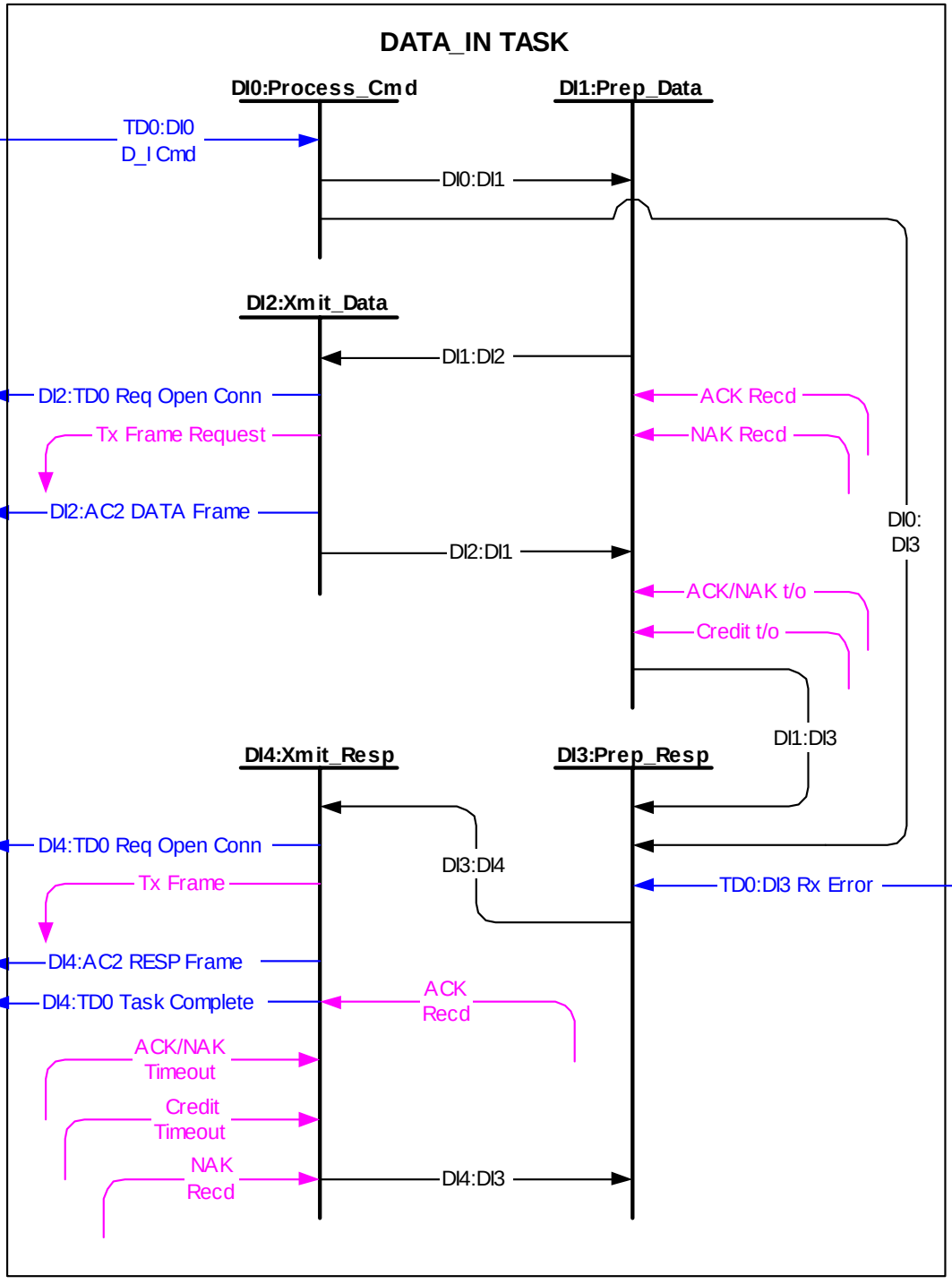

Figure 78. SSP Data in task state machine

9.2.7.4.2 DI0:Process_Cmd state

9.2.7.4.2.1 DI0:Process_Cmd state description

In this state the task manager shall process the tag and the task attribute, place the task in the task set, and pass the tag and the CDB to the device server.

This state is exited when the task manager has passed the tag and the CDB to the device server or has determined that there was an error.

9.2.7.4.2.2 Transition DI0:DI1 (DI_Task Process_Cmd to DI_Task Prep_Data)

The DI0:DI transition shall occur after the task manager has passed the tag and the CDB to the device server.

9.2.7.4.2.3 Transition DI0:DI3 (DI_Task Process_Cmd to DI_Task Prep_Resp)

The DI0:D3 transition shall occur after the task manager has determined that there was an error.

9.2.7.4.3 DI1:Prep_Data state

9.2.7.4.3.1 DI1:Prep_Data state description

In this state the device server shall construct a DATA frame and pass the frame to the target port. This frame shall contain the tag and data for the task. If more than one frame is required to transfer all of the data for the task, this data shall be sequential.

While in this state the link layer may inform the device server that the connection has been terminated as the result of an ACK/NAK timeout or a Credit timeout. The device server shall attempt to send the DATA frame for the task during a subsequent connection.

The link layer informs the target port if the frame was transmitted with or without error (i.e., whether a NAK or ACK was received for the frame). The target port informs the device server of the transmission status.

This state is exited when the device server has passed a DATA frame to the target port or has status to send to the initiator.

9.2.7.4.3.2 Transition DI1:DI2 (DI_Task Prep_Data to DI_Task Xmit_Data)

The DI1:DI2 transition shall occur after the device server has passed a DATA frame to the target port to be sent to the initiator.

9.2.7.4.3.3 Transition DI1:DI3 (DI_Task Prep_Data to DI_Task Prep_Resp)

The DI1:DI3 transition shall when the device server has status to send to the initiator.

9.2.7.4.4 DI2:Xmit_Data state

9.2.7.4.4.1 DI2:Xmit_Data state description

In this state the target port shall send the frame to the initiator port. The transport layer shall notify the link layer that a frame is to be transmitted and whether the frame is interlocked or not.

If there is no connection, the target port may request that a connection be opened.

This state is exited when the target port has sent a frame to the initiator port.

9.2.7.4.4.2 Parameters DI2:AC2 (DI_Task Xmit_Data to App_Client Receive)

The target port shall send a DATA frame to the initiator port.

9.2.7.4.4.3 Transition DI2:DI1 (DI_Task Xmit_Data to DI_Task Prep_Data)

The DI2:DI1 transition shall occur after the target port has sent a DATA frame to the initiator port.

9.2.7.4.5 DI3:Prep_Resp state

9.2.7.4.5.1 DI3:Prep_Resp state description

If this state was entered as a result of the device server having status to send to the initiator, the device server shall build a RESPONSE frame and pass this frame to the target port. The RESPONSE frame shall contain the tag and status for the task.

If this state was entered as the result of the device server being informed by the target port that an error occurred during transmission of a previous RESPONSE frame, the device server shall again pass the RESPONSE frame to the target port, or stop after some vendor specific number of retries.

If an error occurs during transmission of the RESPONSE frame, the device server shall attempt to send the frame during a subsequent connection.

This state is exited after the device server has passed the RESPONSE frame to the target port.

9.2.7.4.5.2 Transition DI3:DI4 (DI_Task Prep_Resp to DI_Task Xmit_Resp)

The DI3:DI4 transition shall occur after the device server has passed a RESPONSE frame to the target port.

9.2.7.4.6 DI4:Xmit_Resp state

9.2.7.4.6.1 DI4:Xmit_Resp state description

In this state the target port sends a RESPONSE frame to the initiator port. The target port shall notify the link layer that the frame is to be transmitted and that the frame is interlocked. The target port informs the device server of the transmission status.

If there is no connection, the target port may request that a connection be opened.

The link layer informs the target port if the frame was transmitted with or without error (i.e., whether a NAK or ACK was received for the frame). While in this state the link layer may inform the device server that the connection has been terminated as the result of an ACK/NAK timeout or a Credit timeout.

If an error occurred during the transmission, the target port notifies the device server. If no error occurred during transmission of the frame the task is complete.

This state is exited after the target port sends the receives acknowledgement from the initiator port that the frame was received or an error occurs.

9.2.7.4.6.2 Parameters DI4:AC2 (DI_Task Xmit_Resp to App_Client Receive)

The target port shall send the RESPONSE frame to the initiator port.

9.2.7.4.6.3 Parameters DI4:TD0 (DI_Task Xmit_Resp to Tgt_Dev Recv_Cmd)

The DI4:TD0 transition shall occur after the target port has informed the device server that the RESPONSE frame was transmitted without error.

9.2.7.4.6.4 Transition DI3:DI3 (DI_Task Xmit_Resp to DI_Task Prep_Resp)

The DI5:DI4 transition shall occur after the target port has informed the device server that an error occurred during transmission of the RESPONSE frame.

9.3 STP transport layer

9.3.1 SATA tunneling

STP encapsulates the SATA protocol with connection management.

Table 66 shows a target port sending a SATA frame to an expander port. The expander device opens a connection to an STP initiator port or to an expander port on the path to the STP initiator port solely for the frame.

Table 66. SATA target port sending a frame

SATA target port to expander port	Expander port to expander port or STP initiator port
SATA_SYNC	idle dword
SATA_X_RDY	OPEN address frame
<repeats>	SATA_X_RDY
<repeats>	<repeats>
<wait for SATA_R_RDY>	<wait for SATA_R_RDY>
SATA_SOF	SATA_SOF
FIS	FIS
CRC	CRC
SATA_EOF	SATA_EOF
SATA_WTRM	SATA_WTRM
SATA_SYNC	CLOSE

Table 67 shows an STP initiator port sending a frame, with the expander device attached to the SATA target port opening a connection solely for the frame.

Table 67. STP initiator port sending a frame

STP initiator port to expander port	Expander port to SATA target port
idle dword	SATA_SYNC
OPEN address frame	SATA_SYNC
SATA_X_RDY	SATA_X_RDY
<repeats>	<repeats>
<wait for SATA_R_RDY>	<wait for SATA_R_RDY>
SATA_SOF	SATA_SOF
FIS	FIS
CRC	CRC
SATA_EOF	SATA_EOF
SATA_WTRM	SATA_WTRM
CLOSE	SATA_SYNC

Other primitives may be interspersed during the connection as defined by SATA: SATA_DMAT, SATA_HOLD, SATA_HOLDA, SATA_R_ERR, SATA_R_IP, SATA_R_OK.

The expander device adds the OPEN address frame and CLOSE on the expander port attached to the STP initiator. The expander device removes the OPEN address frame and CLOSE on the expander port attached to the SATA target port. While the connection is open, the expander device is not involved. Both initiator port and target port use SATA protocol, using with SATA_HOLD/SATA_HOLDA flow control, while the connection is open.

If the expander device adds latency, the SATA_HOLD/SATA_HOLDA buffers in the SATA target port may overflow. SATA target ports assume that a SATA_HOLDA acknowledging a SATA_HOLD arrives within 20 dwords. If the expander device latency increases the latency, data may be lost. Figure 79 shows an example of the problem.

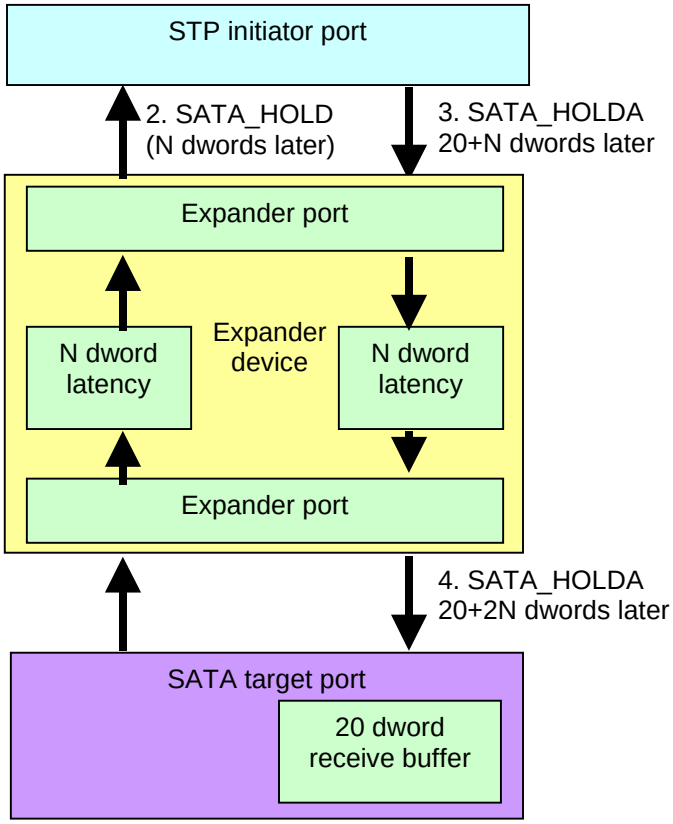

Figure 79. HOLD/HOLDA latency through an expander device

To solve this, expander devices shall include buffering equal to the round-trip latency they add for data flowing to a SATA target port. Figure 80 shows where the buffer sits in the expander device for a SATA target port. The expander device replies to SATA_HOLD with its own SATA_HOLDA within 20 dwords, buffering any additional data that arrives from the STP initiator port.

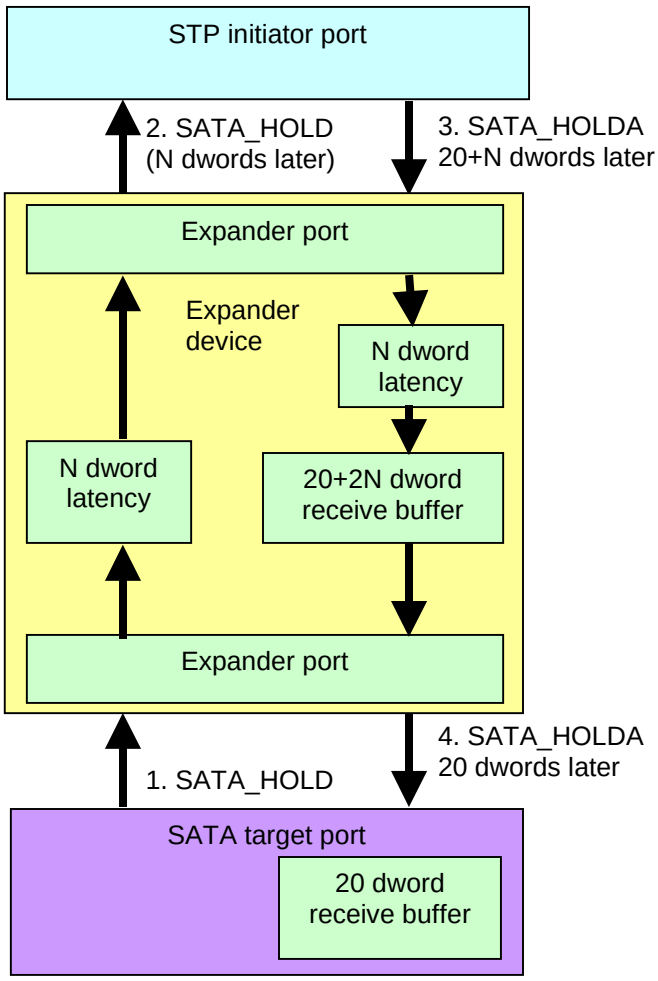

Figure 80. Receive buffer in an expander device for SATA target port

STP initiator ports shall provide a 64 dword buffer to receive data after SATA_HOLD before expecting SATA_HOLDA. This relaxes the amount of buffering needed in the expander device for the data path to the initiator port. Figure 81 shows the expander device buffering needed for an STP initiator port.

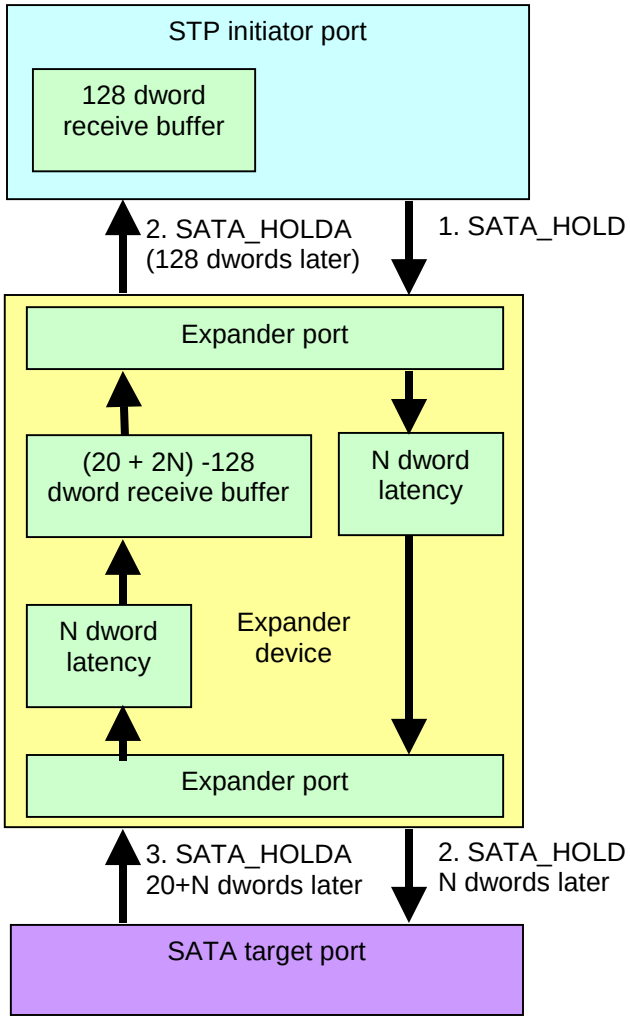

Figure 81. Receive buffer in an expander device for STP initiator port

In a SCSI domain with a single initiator port, when a SATA target port sends an SATA_X_RDY, the expander device may use the time between SATA_X_RDY and SATA_R_RDY to insert an OPEN address frame to open a connection to the initiator port. In a SAS domain with multiple initiator ports with command-tag queuing supported by the target ports, the expander device should accept the start of the frame including the tag, use SATA_HOLD to stall the rest of the frame while the expander device opens a connection to the proper initiator port, and release SATA_HOLD when the connection is ready and the start of the frame has been sent. Only data FISes are subject to flow control, so the expander device shall be capable of accepting a whole register FIS frame.

An expander device may issue CLOSE at the end of each frame, after a timeout waiting for another frame, after every n frames, after a certain time period, after a SATA_CONT is detected, after a SATA_HOLD is detected.

A SATA target port sends a Register - Device to Host FIS after completing the link reset sequence. The expander device shall update a set of shadow registers with these contents and shall not deliver them to any STP initiator port. The STP initiator ports may read the shadow register contents using the REPORT SATA PORT expander function.

9.3.2 SATA tunneling for multiple initiator ports

An expander device that only supports one initiator port can fairly easily tunnel SATA. The connection request creates a virtual circuit, after which the expander device just passes through dwords in both directions unchanged. When a CLOSE is detected in each direction it closes the virtual circuit.

Multiple initiator ports sharing a SATA target presents special problems for expander devices, especially if command overlap, tagged command queuing, and PACKET queuing are supported. SATA drives don't understand multiple host contexts; they have a single task file register set. Accesses from different initiator ports must not overlap. To support multiple initiator ports; an expander device must shadow the task file registers, providing one set per initiator port.

If tagged command queuing is not supported, the expander device may restrict access to the target port for the initiator port that sent the command until the command is complete. If tagged command queuing is supported, the expander device must generate tags for the SATA target port independent of the tags used by each initiator port, which may overlap. When the SATA target port wants to deliver data, the expander device must accept the beginning of the frame, determine which tag is being used and open a connection to the correct initiator port. In single initiator port or non-tagged command queuing environments, the expander device may open a connection after receiving SATA_X_RDY before sending SATA_R_RDY and need not inspect the beginning of a frame.

Expander devices should provide a task file register set for each initiator port. If an expander device is connected to more initiator ports than it has task file register sets, the expander device should respond to any unserviceable requests with OPEN_REJECT(TOO MANY STP INITIATORS).

9.3.3 STP state diagrams

[Editor's note: insert notes about STP modifications to the SATA transport layer state diagram(s) here.]

9.4 SMP transport layer

9.4.1 SMP overview

Inside an SMP connection, the source device sends an SMP_REQUEST frame and the destination device replies with an SMP_RESPONSE frame.

Table 68 shows the types of frames.

Table 68. SMP frame types

Name	Information unit	Originator
SMP_REQUEST	Management request	Initiator port
SMP_RESPONSE	Management response	Target port

Expander devices shall support the SMP_REQUEST and SMP_RESPONSE frames. Other target ports may support these frames.

Figure 82 shows an SMP frame sequence.

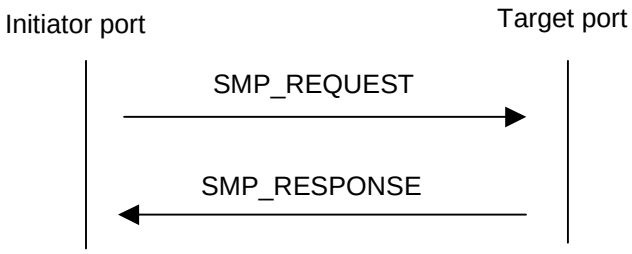

Figure 82. SMP frame sequence

9.4.2 SMP_REQUEST frame

Table 69 shows the SMP request frame format.

Table 69. SMP_REQUEST frame format

Byte	7	6	5	4	3	2	1	0
0	INFORMATION UNIT TYPE (40h)							
1	Reserved							
23								
24	FUNCTION							
25	ADDITIONAL REQUEST BYTES							
m								
	Fill bytes, if needed							
n - 3	(MSB)	CRC						
n								(LSB)

The INFORMATION UNIT TYPE field is set to 40h indicating this is an SMP_REQUEST frame.

The FUNCTION field indicates which function is being requested.

The ADDITIONAL REQUEST BYTES definition and length is based on the function (see 9.4.4.1). The maximum size of the ADDITIONAL REQUEST BYTES field is 1 023 bytes, making the maximum size of the frame 1 052 bytes (1 024 byte data + 24 byte header + 4 byte CRC).

Fill bytes shall be included so the CRC field is aligned on a four byte boundary.

The CRC field is a CRC value (see 7.10) that is computed over the entire frame.

9.4.3 SMP_RESPONSE frame

The SMP_RESPONSE frame is sent by the target port in response to an SMP_REQUEST frame.

Table 70 defines the SMP_RESPONSE frame.

Table 70. SMP_RESPONSE frame

Byte	7	6	5	4	3	2	1	0
0	INFORMATION UNIT TYPE (41h)							
1	Reserved							
23								
24	FUNCTION RESULT							
25	ADDITIONAL RESPONSE BYTES							
m								
	Fill bytes, if needed							
n - 3	(MSB)	CRC						
n								(LSB)

The INFORMATION UNIT TYPE field is set to 41h indicating this is an SMP_RESPONSE frame.

The FUNCTION RESULT field is defined in Table 71.

Table 71. Function results

FUNCTION RESULT	Meaning	Description
00h	REQUEST ACCEPTED	The additional response bytes contain the requested information.
01h	UNKNOWN FUNCTION	The target port does not support the requested SMP function.
All others		Reserved

The ADDITIONAL RESPONSE BYTES field definition depends on the function requested, and are described in the model section. The maximum size of the ADDITIONAL RESPONSE BYTES field is 1 023 bytes, making the maximum size of the frame 1 052 bytes (1 024 byte data + 24 byte header + 4 byte CRC).

Fill bytes shall be included so the CRC field is aligned on a four byte boundary.

The CRC field is a CRC value (see 7.10) that is computed over the entire frame.

9.4.4 Functions

9.4.4.1 Function overview

The FUNCTION field in the SMP REQUEST frame is defined in Table 72.

Table 72. Management functions

FUNCTION	Function	Description
00h	DISCOVER	Return the device names attached to a device.
01h	REPORT GENERAL	Return general information about the device.
02h	REPORT SATA CAPABILITIES	Return information about which SATA features an expander device supports.
03h	INQUIRY	Return vendor and product identification.
04h - 0Fh		Reserved for general input functions.
10h	REPORT PHY	Return information about the specified phy.
11h	REPORT PHY ERROR LOG	Return error logging information about the specified phy.
12h	REPORT PHY SATA	Return information about a phy currently attached to a SATA target port.
13h	REPORT PHY DEVICE NAMES	Return all the device names routed to the specified phy.
13h - 1Fh		Reserved for phy input functions.
20h - 3Fh		Reserved for input functions.
40h - 7Fh		Vendor-specific.
80h - 8Fh		Reserved for general output functions.
90h	PHY CONTROL	Request actions by the specified phy.
91h - 9Fh		Reserved for phy output functions.
A0h - BFh		Reserved for output functions.
C0h - FFh		Vendor-specific.

9.4.4.2 DISCOVER function

The DISCOVER function returns the device names attached to a device. This function shall be implemented by all expander devices and may be implemented by other types of devices.

Table 73 defines the request format.

Table 73. DISCOVER request

Byte	7	6	5	4	3	2	1	0
0	FUNCTION (00h)							
1	Reserved							
3								

Table 74 defines the response format.

Table 74. DISCOVER response

Byte	7	6	5	4	3	2	1	0
0	FUNCTION RESULT							
1	Reserved							
3								
4	(MSB)	DEVICE NAME VALID BITMASK						
11								(LSB)
12	(MSB)	ATTACHED FANOUT EXPANDER BITMASK						
19								(LSB)
20	Reserved							
31								
32	(MSB)	DEVICE NAME 0						
39								(LSB)
…	…							
536	(MSB)	DEVICE NAME 63						
543								(LSB)
544	(MSB)	CRC						
547								(LSB)

The FUNCTION RESULT field is defined in 9.4.3.

The DEVICE NAME BITMASK field contains one bit for each of the 64 possible devices attached to the device. If a bit is set to one, the corresponding DEVICE NAME field is valid. If a bit is set to zero, the corresponding DEVICE NAME field is invalid.

The ATTACHED FANOUT EXPANDER BITMASK field contains one bit for each of the device names. If a bit is set to one, the device name is that of a fanout expander device. If the bit is set to zero, the device name is not that of a fanout expander device. If the expander device is a fanout expander device, none of these bits are set to one.

Each DEVICE NAME NN field contains the device name of an attached device. If the device is an expander device and the attached device is a SATA target device, the device name is created by the expander device.

9.4.4.3 REPORT GENERAL function

The REPORT GENERAL function returns general information about the device. This function may implemented by any type of device and should be implemented by expander devices.

Table 75 defines the request format.

Table 75. REPORT GENERAL request

Byte	7	6	5	4	3	2	1	0
0	FUNCTION (01h)							
1	Reserved							
3								

Table 76 defines the response format.

Table 76. REPORT GENERAL response

Byte	7	6	5	4	3	2	1	0
0	FUNCTION RESULT							
1	Revision							
2	NUMBER OF PHYS							
3	INPUT PHY IDENTIFIER							
4	Reserved							
7								
8	(MSB)	DEVICE NAME						
15								(LSB)
16	(MSB)	ACTIVE PHY BITMASK						
23								(LSB)
24	(MSB)	ATTACHED FANOUT EXPANDER BITMASK						
31								(LSB)
32	(MSB)	ATTACHED EDGE EXPANDER BITMASK						
39								(LSB)
40	(MSB)	ATTACHED SAS INITIATOR BITMASK						
47								(LSB)
48	(MSB)	ATTACHED SAS TARGET BITMASK						
53								(LSB)
54	(MSB)	ATTACHED SATA BITMASK						
75								(LSB)
76	(MSB)	PHY RATE MULTIBITMASK						
83								(LSB)
84	(MSB)	FUNCTIONS SUPPORTED BITMASK						
115								(LSB)

The FUNCTION RESULT field is defined in 9.4.3.

The NUMBER OF PHYS field contains the number of phys in the device.

The INPUT PHY IDENTIFIER field contains the identifier of the phy through which the REPORT GENERAL request was received.

The DEVICE NAME field contains the device name (see 4.3.2).

For each of the PHY BITMASK fields, the LSB contains the value for phy 0 and the MSB contains the value for phy 63.

The ACTIVE PHY BITMASK field contains a bitmask indicating which phys are currently attached to devices and have completed the link reset sequence.

The ATTACHED FANOUT EXPANDER BITMASK field contains a bitmask indicating which phys are connected to fanout expander devices.

The ATTACHED FANOUT EXPANDER BITMASK field contains a bitmask indicating which phys are connected to edge expander devices.

The ATTACHED SAS INITIATOR BITMASK field contains a bitmask indicating which phys are connected to SAS initiator devices.

The ATTACHED SAS TARGET BITMASK field contains a bitmask indicating which phys are connected to SAS target devices.

The ATTACHED SATA BITMASK field contains a bitmask indicating which phys are connected to SATA target devices.

The PHY RATE MULTIBITMASK field indicates the physical link rate of each phy. Each phy is described with four bits, as defined in Table 77.

Table 77. Phy rates

PHY RATE	Rate
0h	Phy does not exist
1h	Phy exists, rate unknown
2h	Phy exists but is disabled
3h	1,5 Gbps
4h	3,0 Gbps
5h - Fh	Reserved

The FUNCTIONS SUPPORTED BITMASK field indicates which SMP functions are supported by the device. The MSB corresponds to function 00h and the LSB corresponds to function FFh.

9.4.4.4 REPORT SATA CAPABILITIES function

The REPORT SATA CAPABILITIES function returns information about which SATA features an expander device supports. This function shall implemented by expander devices supporting attachment to SATA target devices. It shall not be implemented by any other type of device.

Table 78 defines the request format.

Table 78. REPORT SATA CAPABILITIES request

Byte	7	6	5	4	3	2	1	0
0	FUNCTION (02h)							
1	Reserved							
3								

Table 79 defines the response format.

Table 79. REPORT SATA CAPABILITIES response

Byte	7	6	5	4	3	2	1	0
0	FUNCTION RESULT							
1	Reserved						ATA QUEUING CAPABLE	SATA CAPABLE
2	SATA VERSION							
3	NUMBER OF INITIATOR PORTS							

The FUNCTION RESULT field is defined in 9.4.3.

A SATA CAPABLE bit set to one indicates the expander device supports attachment to SATA target ports. A SATA CAPABLE bit set to zero indicates the expander device does not support SATA target ports.

An ATA QUEUING CAPABLE bit set to one indicates the expander device supports the ATA queued feature set (e.g., PACKET, READ DMA QUEUED, SERVICE, and WRITE DMA QUEUED commands). This does not indicate that the SATA target port supports queuing, just that the expander device is capable of support it.

The SATA VERSION field indicates the version of the SATA specification to which the expander device was designed. SATA 1.0 is reported as 00h.

The NUMBER OF INITIATOR PORTS field indicates how many initiator ports the expander device is capable of allowing to share access to a SATA target port. This may be restricted in expander devices which support the ATA queued feature set. Connection requests exceeding this limit shall result in OPEN_REJECT(TOO MANY STP INITIATORS).

9.4.4.5 REPORT MANUFACTURER INFORMATION

The REPORT MANUFACTURER INFORMATION function returns vendor and product identification. This function may be implemented by any type of device.

Table 78 defines the request format.

Table 80. REPORT MANUFACTURER INFORMATION request

Byte	7	6	5	4	3	2	1	0
0	FUNCTION (03h)							
1	Reserved							
3								

Table 79 defines the response format.

Table 81. REPORT MANUFACTURER INFORMATION response

Byte	7	6	5	4	3	2	1	0
0	FUNCTION RESULT							
1	Reserved							
3								
4	ADDITIONAL LENGTH (33h)							
5	Reserved							
7								
8	(MSB)	VENDOR IDENTIFICATION						
15								(LSB)
16	(MSB)	PRODUCT IDENTIFICATION						
31								(LSB)
32	(MSB)	PRODUCT REVISION LEVEL						
35								(LSB)
36	Vendor-specific							
55								

The FUNCTION RESULT field is defined in 9.4.3.

ASCII data fields shall contain only graphic codes (i.e., code values 20h through 7Eh). Left-aligned fields shall place any unused bytes at the end of the field (highest offset) and the unused bytes shall be filled with space characters (20h).

The ADDITIONAL LENGTH field indicates the length in bytes of the parameters, including the ADDITIONAL LENGTH field. If the ADDITIONAL REQUEST BYTES of the SMP_REQUEST is too small to transfer all of the parameters, the ADDITIONAL LENGTH shall not be adjusted to reflect the truncation.

The VENDOR IDENTIFICATION field contains eight bytes of ASCII data identifying the vendor of the product. The data shall be left aligned within the field. The vendor identification string should be one defined in SPC-3 for the Standard INQUIRY data VENDOR IDENTIFICATION field.

The PRODUCT IDENTIFICATION field contains sixteen bytes of ASCII data as defined by the vendor. The data shall be left aligned within the field.

The PRODUCT REVISION LEVEL field contains four bytes of ASCII data as defined by the vendor. The data shall be left-aligned within the field.

9.4.4.6 REPORT PHY function

The REPORT PHY function returns information about the specified phy. This function may implemented by any type of device.

Table 82 defines the request format.

Table 82. REPORT PHY request

Byte	7	6	5	4	3	2	1	0
0	FUNCTION (10h)							
1	Reserved							
2								
3	PHY IDENTIFIER							

The PHY IDENTIFIER field indicates the phy for which information shall be reported.

Table 83 defines the response format.

Table 83. REPORT PHY response

Byte	7	6	5	4	3	2	1	0
0	FUNCTION RESULT							
1	REPORT PHY RESULT							
2	Reserved					SAS PROTOCOL	SATA PENDING	PHYSICAL LINK ACTIVE
3	MAXIMUM PHYSICAL LINK RATE				HARDWARE MAXIMUM PHYSICAL LINK RATE			
4	MINIMUM PHYSICAL LINK RATE				HARDWARE MINIMUM PHYSICAL LINK RATE			
5	Reserved				CURRENT PHYSICAL LINK RATE			
6	ATTACHED PHY IDENTIFIER							
7	ATTACHED DEVICE TYPE		ATTACHED STP INITIATOR	ATTACHED STP TARGET	ATTACHED SSP INITIATOR	ATTACHED SSP TARGET	ATTACHED SMP INITIATOR	ATTACHED SMP TARGET
8	(MSB) ATTACHED DEVICE NAME							
15								(LSB)
16								
31								

The FUNCTION RESULT field is defined in 9.4.3.

The REPORT PHY RESULT field is defined in Table 84.

Table 84. Report PHY result

REPORT PHY RESULT	Description
00h	Phy exists; rest of data is valid.
01h	Phy does not exist; rest of data is invalid.
All others	Reserved.

The PHYSICAL LINK ACTIVE bit is set to one if the phy has completed the link reset sequence and is active.

The SATA PENDING bit is set to one if the physical link is inactive because the beginning of the link reset sequence indicated SATA capability, and the expander device did not complete the link reset sequence.

The SAS PROTOCOL bit is set to one if the link reset sequence indicated SAS capability and is set to zero if the link reset sequence indicated SATA capability. If the SAS PROTOCOL bit is one, the ATTACHED fields indicate the information received during the link initialization sequence. If the SAS PROTOCOL is zero, the attached fields are all set to zero.

The HARDWARE MAXIMUM PHYSICAL LINK RATE field indicates the maximum physical link rate supported by the phy.

The MAXIMUM PHYSICAL LINK RATE field indicates the maximum physical link rate set by the PHY CONTROL function.

The HARDWARE MINIMUM PHYSICAL LINK RATE field indicates the minimum physical link rate supported by the phy.

The MINIMUM PHYSICAL LINK RATE field indicates the minimum physical link rate set by the PHY CONTROL function.

The CURRENT PHYSICAL LINK RATE field indicates the physical link rate negotiated during the link reset sequence. The current physical link rate may be outside the minimum physical link rate and maximum physical link rate if they have been changed since the link reset sequence.

The ATTACHED PHY IDENTIFIER field indicates the PHY IDENTIFIER value received during the link reset sequence.

The ATTACHED DEVICE TYPE field indicates the DEVICE TYPE value received during the link reset sequence.

The ATTACHED STP INITIATOR bit indicates the SSP INITIATOR value received during the link reset sequence.

The ATTACHED STP TARGET bit indicates the SSP TARGET value received during the link reset sequence.

The ATTACHED SSP INITIATOR bit indicates the SSP INITIATOR value received during the link reset sequence.

The ATTACHED SSP TARGET bit indicates the SSP TARGET value received during the link reset sequence.

The ATTACHED SMP INITIATOR bit indicates the SSP INITIATOR value received during the link reset sequence.

The ATTACHED SMP TARGET bit indicates the SSP TARGET value received during the link reset sequence.

If SAS PROTOCOL is set to one, the ATTACHED DEVICE NAME field indicates the device name received during the link reset sequence. If SAS PROTOCOL is set to zero, this field contains the unique device name assigned by the expander device.

9.4.4.7 REPORT PHY ERROR LOG function

The REPORT PHY ERROR LOG returns error logging information about the specified phy. This function may implemented by any type of device.

Table 85 defines the request format.

Table 85. REPORT PHY ERROR LOG request

Byte	7	6	5	4	3	2	1	0
0	FUNCTION (11h)							
1	Reserved							
2								
3	PHY IDENTIFIER							

The PHY IDENTIFIER field indicates the phy for which information shall be reported.

Table 86 defines the response format.

Table 86. REPORT PHY ERROR LOG response

Byte	7	6	5	4	3	2	1	0
0	FUNCTION RESULT							
1	REPORT PHY ERROR LOG RESULT							
2	Reserved							
3								
4	(MSB)	INVALID CHARACTER COUNT						
7								(LSB)
8	(MSB)	DISPARITY ERROR COUNT						
11								(LSB)
12	(MSB)	LOSS OF BIT SYNCHRONIZATION COUNT						
15								(LSB)

The FUNCTION RESULT field is defined in 9.4.3.

The REPORT PHY ERROR LOG RESULT field is defined in Table 87.

Table 87. Report phy error log result

REPORT PHY ERROR LOG RESULT	Description
00h	Phy exists; rest of data is valid.
01h	Phy does not exist; rest of data is invalid.
All others	Reserved.

The INVALID CHARACTER COUNT field indicates the number of invalid 8b10b characters that have been received.

The DISPARITY ERROR COUNT field indicates the number of disparity errors that have been detected.

The LOSS OF BIT SYNCHRONIZATION COUNT field indicates the number of times bit synchronization has been lost.

9.4.4.8 REPORT PHY SATA function

The REPORT PHY SATA function returns information about the SATA state for a specified phy. This function shall be implemented by expander devices supporting attachment to SATA target devices. It shall not be implemented by any other type of device.

Table 88 defines the request format.

Table 88. REPORT PHY SATA request

Byte	7	6	5	4	3	2	1	0
0	FUNCTION (12h)							
1	Reserved							
2								
3	PHY IDENTIFIER							

The PHY IDENTIFIER field indicates the phy for which information shall be reported.

Table 89 defines the response format.

Table 89. REPORT PHY SATA response

Byte	7	6	5	4	3	2	1	0
0	FUNCTION RESULT							
1	REPORT PHY SATA RESULT							
2	Reserved							
3	Reserved							
4	REGISTER DEVICE TO HOST FIS							
23								

The FUNCTION RESULT field is defined in 9.4.3.

The REPORT PHY SATA RESULT field is defined in Table 90.

Table 90. Report phy SATA result

REPORT PHY SATA RESULT	Description
00h	Phy exists; rest of data is valid.
01h	Phy does not exist; rest of data is invalid.
02h	Phy reported SATA protocol in the reset sequence, but the reset sequence has not been completed; rest of data is invalid.
03h	Phy is not using SATA protocol; rest of data is invalid.
All others	Reserved.

The REGISTER DEVICE TO HOST FIS field contains the contents of the last Register - Device to Host FIS delivered by the attached target device. The first byte is always set to 34h, the FIS type (see SATA).

For reference, the FIS is described in Table 91.

Table 91. SATA Register - Device to Host FIS (reference)

Byte	7	6	5	4	3	2	1	0
0	FIS TYPE (34h)							
1	Rsvd	INTR	Reserved					
2	STATUS							
3	ERROR							
4	SECTOR NUMBER							
5	CYLINDER LOW							
6	CYLINDER HIGH							
7	DEVICE HEAD							
8	SECTOR NUMBER EXPANDED							
9	CYLINDER LOW EXPANDED							
10	CYLINDER HIGH EXPANDED							
11	Reserved							
12	SECTOR COUNT							
13	SECTOR COUNT EXPANDED							
14	Reserved							
15	Reserved							
16	Reserved							
17	Reserved							
18	Reserved							
19	Reserved							

9.4.4.9 REPORT PHY DEVICE NAMES function

The REPORT PHY DEVICE NAMES function returns all the device names routed to the specified phy. This function shall be implemented by fanout expander devices and shall not be implemented by any other type of device.

Table 92 defines the request format.

Table 92. REPORT PHY DEVICE NAMES request

Byte	7	6	5	4	3	2	1	0
0	FUNCTION (13h)							
1	Reserved							
2								
3	PHY IDENTIFIER							

The PHY IDENTIFIER field indicates the phy for which information shall be reported.

Table 93 defines the response format.

Table 93. REPORT PHY DEVICE NAMES response

Byte	7	6	5	4	3	2	1	0
0	FUNCTION RESULT							
1	REPORT PHY DEVICE NAMES RESULT							
2	Reserved							
3	Reserved							
4	(MSB)	DEVICE NAME VALID BITMASK						
11								(LSB)
12		Reserved						
19								
20		Reserved						
31								
32	(MSB)	DEVICE NAME 0						
39								(LSB)
…	…							
536	(MSB)	DEVICE NAME 63						
543								(LSB)
544	(MSB)	CRC						
547								(LSB)

The FUNCTION RESULT field is defined in 9.4.3.

The REPORT PHY DEVICE NAMES RESULT field is defined in Table 94.

Table 94. REPORT PHY DEVICE NAMES result

REPORT PHY DEVICE NAMES RESULT	Description
00h	Phy exists; rest of data is valid.
01h	Phy does not exist; rest of data is invalid.
All others	Reserved.

The DEVICE NAME VALID BITMASK field contains one bit for each of the 64 device names. If a bit is set to one, the corresponding DEVICE NAME field is valid. If a bit is set to zero, the corresponding DEVICE NAME field is invalid.

Each DEVICE NAME field contains a device name for which connection requests are which is routed through this phy.

9.4.4.10 PHY CONTROL function

The PHY CONTROL function requests actions by the specified phy. This function may implemented by any type of device.

Table 95 defines the request format.

Table 95. PHY CONTROL request

Byte	7	6	5	4	3	2	1	0
0	FUNCTION (90h)							
1	Reserved							
2								
3	PHY IDENTIFIER							
4	PHY OPERATION							
5	MINIMUM PHYSICAL LINK RATE				MAXIMUM PHYSICAL LINK RATE			
6	Reserved							
7								

The PHY IDENTIFIER field indicates the phy to which the PHY CONTROL request applies.

Table 96 defines the PHY OPERATION field.

Table 96. Phy operation

PHY OPERATION	Operation	Description
00h	NOP	No operation.
01h	LINK RESET	Perform a link reset sequence on the specified phy.
02h	HARD RESET	Perform a link reset sequence on the specified phy. Before the IDENTIFY address frame, send HARD RESET repeatedly. After the next link reset sequence recognized, do not send HARD RESET; send an IDENTIFY address frame again.
01h	ENABLE	Enable the specified phy. Perform a link reset sequence.
02h	DISABLE	Disable the specified phy. Stop transmitting and receiving.
03h	NEA LOOPBACK	Set the specified phy to near-end analog loopback mode (see 7.7.1.1).
04h	FER LOOPBACK	Set the specified phy to far-end retimed loopback test mode (see 7.7.1.2).
05h	CLEAR ERROR LOG	Clear the error log counters for the specified phy.
All others		Reserved.

The MINIMUM PHYSICAL LINK RATE field and MAXIMUM PHYSICAL LINK RATE field indicate which physical link rates the phy shall support during a reset sequence. These fields may be set in the same request in which the LINK RESET operation is requested, or may be set beforehand.

Table 97 defines the response format.

Table 97. PHY CONTROL response

Byte	7	6	5	4	3	2	1	0
0	FUNCTION RESULT							
1	PHY CONTROL RESULT							
2	NUMBER OF LOGICAL LINKS							
3	Reserved							

The FUNCTION RESULT field is defined in 9.4.3.

The PHY CONTROL RESULT field is defined in Table 98.

Table 98. Phy control result

PHY CONTROL RESULT	Description
00h	Operation succeeded.
01h	Phy does not exist.
02h	Operation specified by PHY OPERATION failed (e.g., reset attempted but failed to achieve bit synchronization).
03h	Unknown PHY OPERATION value.
04h	Other problems.
All others	Reserved.

9.4.5 SMP transport layer state diagrams

[Editor's note: insert SMP transport layer state diagram(s) here.]

10 Application layer

10.1 SCSI application layer

10.1.1 SCSI mode parameters

10.1.1.1 Disconnect-Reconnect mode page

10.1.1.1.1 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 99 defines the parameters which are applicable to SSP.

The application client sends the values in the fields to be used by the device server to control the SSP connections by means of a MODE SELECT command. The device server shall then communicate the field values to the SSP target port. The field values are communicated from the device server to the SSP target port in a vendor specific manner.

SSP devices shall only use the disconnect-reconnect page parameter fields defined below in this subclause. If any other fields within the disconnect-reconnect page of the MODE SELECT command contain a non-zero value, the device server shall return CHECK CONDITION status for that MODE SELECT command. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to ILLEGAL FIELD IN PARAMETER LIST.

Table 99. Disconnect-Reconnect mode page for SSP

Byte	7	6	5	4	3	2	1	0
0	PS	Rsvd	PAGE CODE (02h)					
1	PAGE LENGTH (0Eh)							
2	Reserved							
3	Reserved							
4	(MSB)	BUS INACTIVITY LIMIT						
5								(LSB)
6	(MSB)	Reserved						
7								(LSB)
8	(MSB)	MAXIMUM CONNECT TIME LIMIT						
9								(LSB)
10	(MSB)	MAXIMUM BURST SIZE						
11								(LSB)
12	Reserved							
13	Reserved							
14	(MSB)	FIRST BURST SIZE						
15								(LSB)

The PARAMETERS SAVEABLE (PS) bit is defined in SPC-3.

The PAGE CODE (PS) field is 02h and the PAGE LENGTH field is 0Eh.

10.1.1.1.2 BUS INACTIVITY TIME LIMIT field

The value in the BUS INACTIVITY TIME LIMIT field specifies the maximum period that a target port is permitted to maintain a connection without transferring a frame to the initiator port. This value shall be the number of 100 ~~m~~µs increments between frames that the target port transmits during a connection. When this number is exceeded, the target port shall prepare to close the connection (i.e., by requesting to have the link layer transmit DONE). This value may be rounded as defined in SPC-3. A value of zero in this field shall specify that there is no bus inactivity time limit.

10.1.1.1.3 MAXIMUM CONNECT TIME LIMIT field

The value in the MAXIMUM CONNECT TIME LIMIT field specifies the maximum duration of a connection. This value shall be the number of 100 ~~m~~µs increments that a target port transmits during a connection after which the target port shall prepare to close the connection (i.e., a value of one in this field specifies that the time shall be less than

or equal to 100 ~~m~~µs, a value of two in this field specifies that the time shall be less than or equal to 200 ~~m~~µs, etc.). If a target port is transferring a frame when the maximum connection time limit is exceeded, the target port shall complete transfer of the frame before preparing to close the connection. A value of zero in this field shall specify that there is no maximum connection time limit.

10.1.1.1.4 MAXIMUM BURST SIZE field

For read data, the value in the MAXIMUM BURST SIZE field shall specify the maximum amount of data that is transferred during a connection by a target port per I_T_L_Q nexus without transferring at least one frame for a different I_T_L_Q nexus. If there is read data for only one I_T_L_Q nexus, the target port shall prepare to close the connection after the amount of data specified by the MAXIMUM BURST SIZE field is transferred to the initiator port.

For write data, the value shall specify the maximum amount of data that a target port requests via a single XFER_RDY information unit.

This value shall be specified in 512-byte increments (i.e., a value of one in this field specifies that the number of bytes transferred to the initiator port for the nexus shall be less than or equal to 512, a value of two in this field specifies that the number of bytes transferred to the initiator port for the nexus shall be less than or equal to 1 024, etc.). The device server may round this value down as defined in SPC-3. A value of zero in this field shall specify that there is no maximum burst size.

10.1.1.1.5 FIRST BURST SIZE field

The value in the FIRST BURST SIZE field specifies the maximum amount of write data that may be sent by the initiator port to the target port without having to receive an XFER_RDY information unit from the target port. Specifying a non-zero value in the FIRST BURST SIZE field is equivalent to an implicit XFER_RDY information unit for each command requiring write data where the transfer length is specified in the WRITE DATA LENGTH field.

The rules for data transferred using the value in the FIRST BURST SIZE field are the same as those used for data transferred for an XFER_RDY information unit (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 information unit).

This value shall specify 512-byte increments of data (i.e., a value of one in this field specifies that the number of bytes transferred by the initiator port shall be less than or equal to 512, a value of two in this field specifies that the number of bytes transferred by the initiator port shall be less than or equal to 1 024, etc.).

If the amount of data to be transferred for the command is less than the amount of data specified by the FIRST BURST SIZE field, the target port shall not transmit an XFER_RDY information unit for the command. If the amount of data to be transferred for the command is greater than the amount of data specified by the FIRST BURST SIZE field, the target port shall request additional data for the command by sending XFER_RDY information units to the initiator port. All data for the command need not be transferred during this connection

A value of zero in this field shall specify that there is no first burst size, i.e., an initiator port shall transmit no data frames to the target port before receiving an XFER_RDY information unit.

10.1.1.2 Protocol-Specific Port mode page

The Protocol-Specific Port mode page (see SPC-3) contains parameters that affect SAS SSP target port operation. This page shall be implemented by all SAS SSP target devices. Table 100 shows the format of the page for SAS SSP.

Table 100. Protocol-Specific Port Control mode page for SAS SSP

Byte	7	6	5	4	3	2	1	0
0	PS	LONG (0)	PAGE CODE (19h)					
1	PAGE LENGTH (06h)							
2	Reserved				PROTOCOL IDENTIFIER (6h)			
3	Reserved							
4	(MSB)	I_T NEXUS LOSS TIME						
5								(LSB)
6	(MSB)	Reserved						
7								(LSB)

The LONG field shall be set to zero, indicating only the short format is used. The PROTOCOL IDENTIFIER field shall be set to 6h indicating this is a SAS SSP specific mode page.

The I_T NEXUS LOSS TIME field indicates how long in milliseconds the target port shall retry connection requests that are rejected with OPEN_REJECT (NO DESTINATION) before treating it as an I_T nexus loss (see 4.7). The default setting shall be 2 000 ms. An I_T NEXUS LOSS TIME of zero indicates the target port shall never consider rejections an I_T nexus loss.

10.1.1.3 Protocol-Specific Logical Unit mode page

SSP logical units shall not include the Protocol-Specific Logical Unit mode page (see SPC-3).

10.1.2 SCSI log parameters

No SAS-specific log pages are defined.

10.1.3 SCSI commands

No SAS-specific SCSI commands are defined.

10.2 ATA application layer

No SAS-specific ATA commands are defined.

11 SCSI architecture mapping

11.1 Names and identifiers

Table 101 describes how various SAM-2 objects are implemented in SSP.

Table 101. SAM-2 object mapping

SAM-2 object	SSP implementation
Initiator port identifier	Initiator device name
Initiator port name	Not defined
Target port identifier	Target device name
Target port name	Not defined
Device name	Device name

11.2 Protocol services

11.2.1 Protocol services overview

SSP provides the protocol services required by SAM-2 in support of the Execute Command remote procedure call and several task management remote procedure calls.

Table 102 shows the mapping of the remote procedure calls to protocol services and how each protocol service is implemented in SSP.

Table 102. SCSI architecture mapping

Remote procedure call	Type of protocol service	Protocol service interaction	Protocol service	I/T[1]	SSP implementation
Execute Command	Request/ Confirmation	Request	Send SCSI Command	I	COMMAND IU
		Indication	SCSI Command Received	T	Receipt of the COMMAND IU
		Response	Send Command Complete	T	RESPONSE IU
		Confirmation	Command Complete Received	I	Receipt of the RESPONSE IU
	Data Transfer[2]	Request	Send Data-In	T	DATA IUs
		Confirmation	Data-In Delivered	T	ACK of DATA IUs
		Request	Receive Data-Out	T	XFER_RDY IU
		Confirmation	Data-Out Received	T	Receipt of DATA IUs
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 IU
		Indication	Task Management Request Received	T	Receipt of the TASK IU
		Response	Task Management Function Executed	T	RESPONSE IU
		Confirmation	Received Task Management Function-Executed	I	Receipt of the RESPONSE IU
Report Asynchronous Event[3]	Request/ Confirmation	Request	Send Asynchronous Event Notification	T	AEN IU
		Indication	Asynchronous Event Notification Received	I	Receipt of AEN IU
		Response	Retrieve Asynchronous Event Report	I	REQUEST SENSE command
		Confirmation	Asynchronous Event Report Delivered	T	Receipt of REQUEST SENSE command

Note 1: I/T indicates whether the initiator device (I) or the target device (T) implements the protocol service.
Note 2: Initiator device Data Transfer protocol services are not specified by SAM-2.
Note 3: Asynchronous event reporting remote procedure call and protocol services are not yet specified by SAM-2.

[Editor's note: added "Task Management" before "Function Executed" to be clearer. This diverges from SAM-2 and should be undone unless SAM-2 is changed.]

11.2.2 Send SCSI Command protocol service

The Send SCSI Command protocol service request is implemented by an initiator port sending a COMMAND IU. Table 103 shows the arguments to the Send SCSI Command protocol service.

Table 103. Send SCSI Command protocol service arguments

Argument	SAS SSP implementation
I_T_L_x nexus	I_T_L_Q nexus: a) I_T indicated by the open connection; b) L indicated by the LOGICAL UNIT NUMBER field in the frame header; and c) Q indicated by the TAG field in the frame header.
CDB	Carried by the CDB field in the COMMAND IU.
[Task Attribute]	Carried by the TASK ATTRIBUTE field in the COMMAND IU.
[Data-In Buffer Size]	Maximum of 2^32
[Data-Out Buffer]	Internal to initiator port
[Data-Out Buffer Size]	Maximum of 2^32
[Autosense Request]	True
[Command Reference Number]	Ignored

11.2.3 SCSI Command Received protocol service

The SCSI Command Received protocol service indication is implemented by a device server receiving a COMMAND IU. Table 104 shows the arguments to the SCSI Command Received protocol service.

Table 104. SCSI Command Received protocol service arguments

Argument	SAS SSP implementation
I_T_L_x nexus	I_T_L_Q nexus: a) I_T indicated by the open connection; b) L indicated by the LOGICAL UNIT NUMBER field in the frame header; and c) Q indicated by the TAG field in the frame header.
CDB	Carried by the CDB field in the COMMAND IU.
[Task Attribute]	Carried by the TASK ATTRIBUTE field in the COMMAND IU.
[Autosense Request]	True
[Command Reference Number]	Ignored

11.2.4 Send Command Complete protocol service

The Send Command Complete protocol service response is implemented by a device server sending a RESPONSE IU. Table 105 shows the arguments to the Send Command Complete protocol service.

Table 105. Send Command Complete protocol service arguments

Argument	SAS SSP implementation
I_T_L_x nexus	I_T_L_Q nexus: a) I_T indicated by the open connection; b) L indicated by the LOGICAL UNIT NUMBER field in the frame header; and c) Q indicated by the TAG field in the frame header.
[Sense Data]	Carried by the RESPONSE IU SENSE DATA field
Status	Carried by the RESPONSE IU STATUS field
Service Response	TASK COMPLETE: The RESPONSE IU contains an RSPVALID bit set to zero and a STATUS field set to a value other than INTERMEDIATE or INTERMEDIATE-CONDITION MET. LINKED COMMAND COMPLETE: The RESPONSE IU contains an RSPVALID bit set to zero and a STATUS field set to INTERMEDIATE or INTERMEDIATE-CONDITION MET. SERVICE DELIVERY OR TARGET FAILURE: The RESPONSE IU contains a RSPVALID bit set to one.

11.2.5 Command Complete Received protocol service

The Command Complete Received protocol service confirmation is implemented by an initiator port receiving a RESPONSE IU or a NAK for the COMMAND IU. Table 106 shows the arguments to the Command Complete Received protocol service.

Table 106. Command Complete Received protocol service arguments

Argument	SAS SSP implementation
I_T_L_x nexus	I_T_L_Q nexus: a) I_T indicated by the open connection; b) L indicated by the LOGICAL UNIT NUMBER field in the frame header; and c) Q indicated by the TAG field in the frame header.
[Data-In Buffer]	Internal to initiator port
[Sense Data]	The RESPONSE IU SENSE DATA field
Status	The RESPONSE IU STATUS field
Service Response	TASK COMPLETE: The RESPONSE IU contains an RSPVALID bit set to zero and a STATUS field set to a value other than INTERMEDIATE or INTERMEDIATE-CONDITION MET. LINKED COMMAND COMPLETE: The RESPONSE IU contains an RSPVALID bit set to zero and a STATUS field set to INTERMEDIATE or INTERMEDIATE-CONDITION MET. SERVICE DELIVERY OR TARGET FAILURE: The RESPONSE IU contains a RSPVALID bit set to one, or the COMMAND IU was NAKed.

11.2.6 Send Data-In protocol service

The Send Data-In protocol service request is implemented by a device server sending a DATA IU. Table 107 shows the arguments to the Send Data-In protocol service.

Table 107. Send Data-In protocol service arguments

Argument	SAS SSP implementation
I_T_L_x nexus	I_T_L_Q nexus: a) I_T indicated by the open connection; b) L indicated by the LOGICAL UNIT NUMBER field in the frame header; and c) Q indicated by the TAG field in the frame header.
Device Server Buffer	Internal to device server
Application Client Buffer Offset	Based on number of DATA IUs previously sent
Request Byte Count	Indicated by the size of the DATA IU

11.2.7 Data-In Delivered protocol service

The Data-In Delivered protocol service indication is implemented by a device server receiving an ACK for a DATA IU. Table 108 shows the arguments to the Data-In Delivered protocol service.

Table 108. Data-In Delivered protocol service arguments

Argument	SAS SSP implementation
I_T_L_x nexus	I_T_L_Q nexus: a) I_T indicated by the open connection; b) L indicated by the LOGICAL UNIT NUMBER field in the frame header; and c) Q indicated by the TAG field in the frame header.

11.2.8 Receive Data-Out protocol service

The Receive Data-Out protocol service request is implemented by a device server sending an XFER_RDY IU. Table 109 shows the arguments to the Receive Data-Out protocol service.

Table 109. Receive Data-Out protocol service arguments

Argument	SAS SSP implementation
I_T_L_x nexus	I_T_L_Q nexus: a) I_T indicated by the open connection; b) L indicated by the LOGICAL UNIT NUMBER field in the frame header; and c) Q indicated by the TAG field in the frame header.
Application Client Buffer Offset	Based on number of DATA IUs received
Request Byte Count	Carried by the XFER_RDY IU WRITE DATA LENGTH field.
Device Server Buffer	Internal to device server.

11.2.9 Data-Out Received protocol service

The Data-Out Received protocol service indication is implemented by a device server receiving DATA IUs in response to an XFER_RDY IU. Table 110 shows the arguments to the Data-Out Received protocol service.

Table 110. Data-Out Received protocol service arguments

Argument	SAS SSP implementation
I_T_L_x nexus	I_T_L_Q nexus: a) I_T indicated by the open connection; b) L indicated by the LOGICAL UNIT NUMBER field in the frame header; and c) Q indicated by the TAG field in the frame header.

11.2.10 Send Task Management Request protocol service

The Send Task Management Request protocol service request is implemented by an initiator port sending a TASK IU. Table 111 shows the arguments to the Send Task Management Request protocol service.

Table 111. Send Task Management Request protocol service arguments

Argument	SAS SSP implementation
I_T_L_x nexus	I_T_L_Q nexus: a) I_T indicated by the open connection; b) L indicated by the LOGICAL UNIT NUMBER field in the frame header; and c) Q indicated by the TAG field in the frame header.
Function Identifier	Carried by the TASK MANAGEMENT FUNCTION field in the TASK IU. Only Abort Task, Abort Task Set, Clear ACA, Clear Task Set, Logical Unit Reset, and Query Task are supported.

11.2.11 Task Management Request Received protocol service

The Task Management Request Received protocol service indication is implemented by a device server receiving a TASK IU. Table 112 shows the arguments to the Task Management Request Received protocol service.

Table 112. Task Management Request Received protocol service arguments

Argument	SAS SSP implementation
I_T_L_x nexus	I_T_L_Q nexus: a) I_T indicated by the open connection; b) L indicated by the LOGICAL UNIT NUMBER field in the frame header; and c) Q indicated by the TAG field in the frame header.
Function Identifier	Carried by the TASK MANAGEMENT FUNCTION field in the TASK IU. Only these task management functions are supported: a) Abort Task; b) Abort Task Set; c) Clear ACA; d) Clear Task Set; e) Logical Unit Reset; and f) Query Task.

11.2.12 Task Management Function Executed protocol service

The Task Management Function Executed protocol service response is implemented by a device server sending a RESPONSE IU. Table 113 shows the arguments to the Task Management Function Executed protocol service.

Table 113. Task Management Function Executed protocol service arguments

Argument	SAS SSP implementation
I_T_L_x nexus	I_T_L_Q nexus: a) I_T indicated by the open connection; b) L indicated by the LOGICAL UNIT NUMBER field in the frame header; and c) Q indicated by the TAG field in the frame header.
Service Response	FUNCTION REJECTED: The RESPONSE IU contains a SNSVALID bit set to zero, an RSPVALID bit set to one, and an RSP_CODE field set to: a) COMMAND OR TASK FIELDS INVALID; b) TASK MANAGEMENT FUNCTION NOT SUPPORTED; or c) TASK MANAGEMENT FUNCTION FAILED. FUNCTION COMPLETE: The RESPONSE IU contains a SNSVALID bit set to zero, an RSPVALID bit set to one, and an RSP_CODE field set to TASK MANAGEMENT FUNCTION COMPLETE. SERVICE DELIVERY OR SUBSYSTEM FAILURE: The RESPONSE IU contains a SNSVALID bit set to one.

11.2.13 Received Task Management Function-Executed protocol service

The Received Task Management Function-Executed protocol service confirmation is implemented by an initiator port receiving a RESPONSE IU or receiving a NAK for the TASK IU. Table 114 shows the arguments to the Received Task Management Function-Executed protocol service.

Table 114. Received Task Management Function-Executed protocol service arguments

Argument	SAS SSP implementation
I_T_L_x nexus	I_T_L_Q nexus: a) I_T indicated by the open connection; b) L indicated by the LOGICAL UNIT NUMBER field in the frame header; and c) Q indicated by the TAG field in the frame header.
Service Response	FUNCTION REJECTED: The RESPONSE IU contains a SNSVALID bit set to zero, an RSPVALID bit set to one, and an RSP_CODE field set to: a) COMMAND OR TASK FIELDS INVALID; b) TASK MANAGEMENT FUNCTION NOT SUPPORTED; or c) TASK MANAGEMENT FUNCTION FAILED. FUNCTION COMPLETE: The RESPONSE IU contains a SNSVALID bit set to zero, an RSPVALID bit set to one, and an RSP_CODE field set to TASK MANAGEMENT FUNCTION COMPLETE. SERVICE DELIVERY OR TARGET FAILURE: The RESPONSE IU contains a SNSVALID bit set to one, or the TASK IU was NAKed.

11.2.14 Asynchronous event notification protocol services

SAM-2 does not currently model asynchronous event reporting; it will probably be added to SAM-3. The steps for SAS SSP are:

1) Target port sends AEN IU;
2) Initiator port receives AEN IU, sends AEN_RESPONSE;
3) Application client sends REQUEST SENSE command to the logical unit; and
4) Logical unit receives REQUEST SENSE command, delivers sense data.

A CRC implementation (informative)

A.1 CRC generator and checker implementation examples

Figure 83 shows an example of a CRC generator.

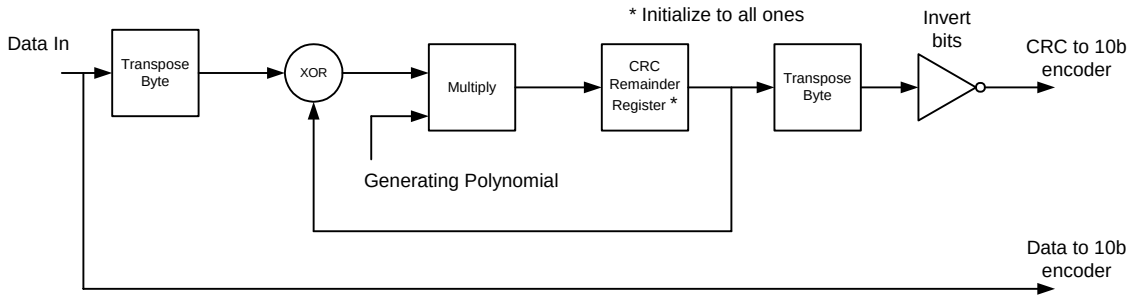

Figure 83. CRC generator example

Figure 84 shows an example of a CRC checker.

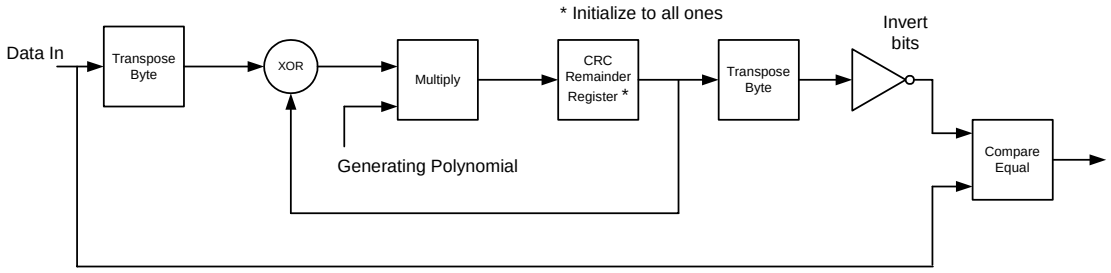

Figure 84. CRC checker example

The resulting CRC for test patterns 1, 2 and 3 below are included to provide a validation of CRC generating and check implementations.

A.2 CRC examples

Table 115 shows several CRC examples. Data is shown in dwords.

Table 115. CRC examples

Data	CRC
00010203h 04050607h 08090A0Bh 0C0D0E0Fh 10111213h 14151617h 18191A1Bh 1C1D1E1Fh	8A7E2691h
00000001h 00000000h 00000000h 00000000h 00000000h 00000000h 00000000h 00000000h	898C0D7Ah
00000000h 00000000h 00000000h 00000000h 00000000h 00000000h 00000000h 00000001h	3B650D6Eh

B Device name hashing (informative)

B.1 Hashing overview

4.3.3 describes hashed device names and the algorithm used to create them.

B.2 Hash collision probability

The following are Monte-Carlo simulations evaluating the probability of collision in a system containing 128 devices. Four models were used for the models for the simulations.

The random model uses a system with 128 randomly chosen 64-bit integers as WWNs.

The sequential model uses a system with 128 sequentially-assigned WWNs starting from a random 64-bit base.

The lots model uses:

a) 2 sequentially assigned initiator WWNs with unique company IDs and random vendor specific identifiers;
b) 125 randomly drawn WWNs from a 10,000-unit production lot. The vendor specific identifiers within the lot were assigned by 10 WWN-writers, randomly drawn from a pool of 4096 possible WWN-writers. Each WWN-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) 1 randomly chosen WWN (representing a replacement unit) with another unique company ID.

The threelots model uses:

a) 2 sequentially assigned initiator WWNs with unique company IDs and random vendor specific identifiers;
b) 125 randomly drawn WWNs from three 10,000-unit lots. The vendor specific identifiers within each lot were assigned by 10 WWN-writers, randomly drawn from a pool of 4096 possible WWN-writers for that vendor. Each WWN-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) 1 randomly chosen WWN (representing a replacement unit) with another unique company ID.

Table 116 shows the results of Monte-Carlo simulation.

Table 116. Monte-Carlo simulation results

WWN model	Trials	Collisions	Average Collisions per System
lots	2,000,000,000	45,063	.0000225315
threelots	2,000,000,000	662,503	0.0003312515
random	10,000,000	4,882	0.0004882
sequential	10,000,000	0	0

B.3 Hash generation

One way to implement the hashing encoder in hardware is to use serial shift registers as show in Figure 85. For error correction purposes, the number of data bits is limited to 39. For hashing purposes, the circuit shown serves as a divider. Because the period of this generator polynomial is 63, any binary sequence of length exceeding 63 is treated as a 63-bit sequence with bit 63 * L + k added to bit k modula 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 modula 2 to bit 0. With this wrapping, a binary sequence of any length can be treated as an equivalent binary sequence of 63 bits, which, in turn, may be treated as a degree-62 polynomial. After feeding this equivalent degree-62 polynomial into the circuit shown, the shift register contains the remainder from dividing the degree-62 input polynomial by the generator polynomial. This remainder is the hashed result.

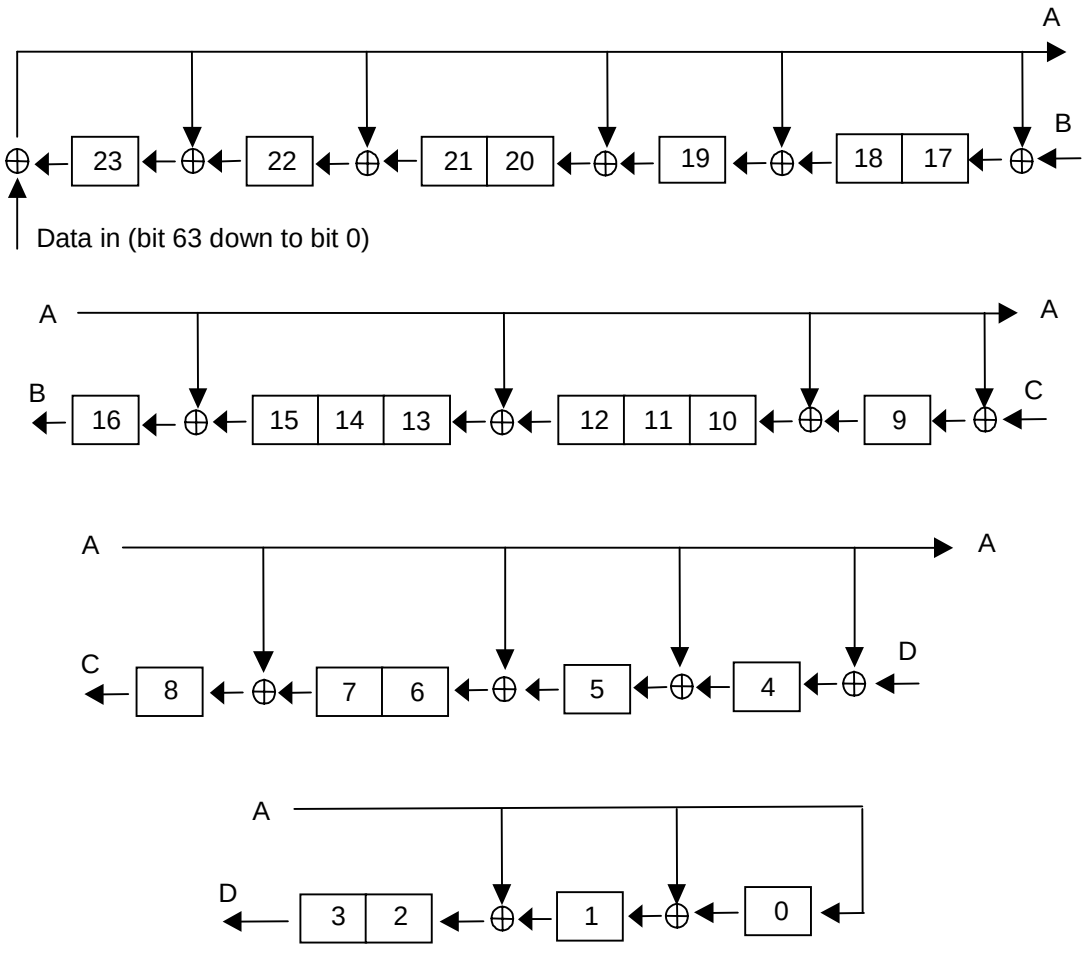

Figure 85. BCH(69, 39, 9) code generator

B.4 Hash implementation in C

The following is an example C program that generates a 24-bit hashed value from a 64-bit value.

```
typedef unsigned int uint32_t;
uint32_t hash(uint32_t upperbits, uint32_t lowerbits)
{
    const unsigned distance_9_poly = 0x1DB2777;
    uint32_t msb = 0x1000000;
    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;
}
```

B.5 Hash implementation with XORs

These equations generate the 24-bit DEVICE NAME CHECK field for the SSP frame header from a 64-bit device name. 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;
```

B.6 Hash examples

Table 117 shows examples using realistic device names as input values.

Table 117. Hash results for realistic device names

64-bit input value	24-bit hashed value
500107534F0CFC88h	D0B992h
50010B2B3CBF639h	B5DF59h
5002037E157FEC63h	B064F7h
50004CF6FBCE3889h	88FF12h
50020374C4657EC7h	F36570h
50010D92A016E450h	9F9571h
50002A58850ACC66h	64B6B9h
50008C7BEE7910DEh	8D6135h
500508BDC22CAC94h	86ECF1h
500805F3334B0AD3h	752AB2h
500A0B8AFAA6A820h	5543A7h
500805E6BCC55C68h	463DEDh

Table 118 shows examples using a “walking ones” pattern to generate the input values.

Table 118. 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 119 shows examples using a “walking zeros” pattern to generate the input values.

Table 119. Hash results for a walking zeros pattern

64-bit input value	24-bit hashed value	64-bit input value	24-bit hashed value
FFFFFFFFFFFFFFFEh	000000h	FFFFFFFEFFFFFFFFh	5915B5h
FFFFFFFFFFFFFFFDh	B64EEEh	FFFFFFFDFFFFFFFFh	046584h
FFFFFFFFFFFFFFFBh	01F445h	FFFFFFFBFFFFFFFFh	BE85E6h
FFFFFFFFFFFFFFF7h	B5A664h	FFFFFFF7FFFFFFFFh	106255h
FFFFFFFFFFFFFFEFh	062551h	FFFFFFEFFFFFFFFFh	968A44h
FFFFFFFFFFFFFFDFh	BA044Ch	FFFFFFDFFFFFFFFFh	407D11h
FFFFFFFFFFFFFFBFh	196101h	FFFFFFBFFFFFFFFFh	36B4CCh
FFFFFFFFFFFFFF7Fh	848CECh	FFFFFF7FFFFFFFFFh	DB2776h
FFFFFFFFFFFFFEFFh	647041h	FFFFFEFFFFFFFFFFh	DB2775h
FFFFFFFFFFFFFDFFh	7EAE6Ch	FFFFFDFFFFFFFFFFh	DB2773h
FFFFFFFFFFFFFBFFh	4B1236h	FFFFFBFFFFFFFFFFh	DB277Fh
FFFFFFFFFFFFF7FFh	206A82h	FFFFF7FFFFFFFFFFh	DB2767h
FFFFFFFFFFFEFFFFh	F69BEAh	FFFFEFFFFFFFFFFFh	DB2757h
FFFFFFFFFFFDFFFFh	805E4Dh	FFFFDFFFFFFFFFFFh	DB2737h
FFFFFFFFFFFBFFFFh	6DD503h	FFFFBFFFFFFFFFFFh	DB27F7h
FFFFFFFFFFF7FFFFh	6DE4E8h	FFFF7FFFFFFFFFFFh	DB2677h
FFFFFFFFFFEFFFFFh	6D873Eh	FFFEFFFFFFFFFFFFh	DB2577h
FFFFFFFFFFDFFFFFh	6D4092h	FFFDFFFFFFFFFFFFh	DB2377h
FFFFFFFFFFBFFFFFh	6CCFCAh	FFFBFFFFFFFFFFFFh	DB2F77h
FFFFFFFFFF7FFFFFh	6FD17Ah	FFF7FFFFFFFFFFFFh	DB3777h
FFFFFFFFFEFFFFFFh	69EC1Ah	FFEFFFFFFFFFFFFFh	DB0777h
FFFFFFFFFDFFFFFFh	6596DAh	FFDFFFFFFFFFFFFFh	DB6777h
FFFFFFFFFBFFFFFFh	7D635Ah	FFBFFFFFFFFFFFFFh	DBA777h
FFFFFFFFF7FFFFFFh	4C885Ah	FF7FFFFFFFFFFFFFh	DA2777h
FFFFFFFFEFFFFFFFh	2F5E5Ah	FEFFFFFFFFFFFFFFh	D92777h
FFFFFFFFDFFFFFFFh	E8F25Ah	FDFFFFFFFFFFFFFFh	DF2777h
FFFFFFFFBFFFFFFFh	BC8D2Dh	FBFFFFFFFFFFFFFFh	D32777h
FFFFFFFF7FFFFFFFh	1473C3h	F7FFFFFFFFFFFFFFh	CB2777h
FFFFFFFEFFFFFFFFh	9EA968h	EFFFFFFFFFFFFFFFh	FB2777h
FFFFFFFDFFFFFFFFh	503B49h	DFFFFFFFFFFFFFFFh	9B2777h
FFFFFFFBFFFFFFFFh	16387Ch	BFFFFFFFFFFFFFFFh	5B2777h
FFFFFFF7FFFFFFFFh	9A3E16h	7FFFFFFFFFFFFFFFh	000000h

C Scrambing (informative)

[Editor's note: Sample code will be provided here for big-endian scrambling.]

D SAS logo (informative)

The SAS logo is shown in Figure 86. This logo should be included on all connectors to SAS ports.

Figure 86. SAS logo

E REPORT LUNs enhancement for asynchronous event notification [SPC-3 proposal]

E.1 Overview

When an asynchronous event occurs in a logical unit, the target port sends an AEN IU to notify an initiator port of the event. Although this IU include the logical unit number, the initiator port is not required to store that information and may not be able to remember which logical unit reported the asynchronous event.

SSP target ports shall include the REPORT LUNS well-known logical unit and support the report code value requesting logical units that have asynchronous events pending (defined below).

[Editor's note: 02-175r0 proposes dropping the two-step REPORT LUNS/REQUEST SENSE process and adding a new sense data descriptor (for the new format) that includes a logical unit number. This would be retrieved through the well-known LUN in one step.]

E.2 REPORT LUNs command [proposed changes to SPC-3]

[Editor's note: quoting existing text…]

The REPORT LUNS command (see table 99) requests that the peripheral device logical unit inventory accessible to the initiator port via the addressed SCSI target port be sent to the application client. The logical unit inventory is a list that shall include the logical unit numbers of all logical units having a PERIPHERAL QUALIFIER value of 000b (see 7.3.2). Logical unit numbers for logical units with PERIPHERAL QUALIFIER values of 100b, 101b, 110b, or 111b may optionally be included in the logical unit inventory.

A SCSI device that is capable of supporting a LUN address other than zero shall support a REPORT LUNS command that is addressed to logical unit zero. Support of the REPORT LUNS command by logical units other than logical unit zero is optional. Support of the REPORT LUNS command on devices having only a single logical unit with the logical unit number of zero is optional.

SPC-3 Table 99 - REPORT LUNS command

	7	6	5	4	3	2	1	0
0	OPERATION CODE (A0h)							
1	Reserved							
2	SELECT REPORT							
3	REPORT SUBSET [new field]							
4	Reserved							
5	Reserved							
6	(MSB)							
7				ALLOCATION LENGTH				
8								
9								(LSB)
10	Reserved							
11	CONTROL							

The SELECT REPORT field (see table 100) specifies the types of logical unit addresses that shall be reported.

SPC-3 Table 100 - Select report code values

Code	Description
00h	The logical unit addresses reported shall be limited to the following addressing methods (see SAM-2): a) Logical unit addressing method, b) Peripheral device addressing method; and c) Flat space addressing method.
01h	The list of logical units shall only contain well known logical units, if any. If there are no well known logical units the LUN LIST LENGTH field shall be zero.
02h	All logical units accessible to the initiator port via the addressed SCSI target port shall be reported.
03h - FFh	Reserved.

[This is the proposed new field description:]

The REPORT SUBSET field (see table xx) specifies the types of logical unit addresses that shall be reported.

SPC-3 Table xx - Report subset code values

Code	Description
00h	All logical units chosen with the SELECT REPORT field shall be reported.
01h	Only logical units chosen with the SELECT REPORT field with pending asynchronous events shall be reported.
02h - FFh	Reserved.

F SPC-3 protocol-specific changes [SPC-3 proposal]

F.1 Protocol identifier

In the protocol-specific parameters section, assign a protocol identifier for SAS SSP.

Section 8.5.1 Protocol specific parameters introduction

SPC-3 Table 231 - PROTOCOL IDENTIFIER values

Protocol identifier	Description	Protocol standard
6h	Serial Attached SCSI (SAS) Serial SCSI Protocol (SSP)	SAS

F.2 Transport ID

[Add this to section 8.5.4 Access controls TransportID access identifiers]

Section 8.5.4.x SAS SSP TransportIDs for initiators using SAS SSP

A Serial Attached SCSI (SAS) Serial SCSI Protocol (SSP) TransportID (see Table xxx) identifies a SAS SSP initiator device.

Table xxx. SAS SSP TransportID format

Byte	7	6	5	4	3	2	1	0
0	Reserved				PROTOCOL CODE (6h)			
1	Reserved							
2	Reserved							
3	Reserved							
4	DEVICE NAME							
7								
8	Reserved							
23								

The DEVICE NAME field specifies the device name of the initiator device.

F.3 Target descriptor

In the EXTENDED COPY section, add a descriptor type code for SAS SSP.

SPC-3 Table 24 - EXTENDED COPY descriptor type values

Descriptor type code	Reference	Description	Shorthand
E9h	8.5.3.x	SAS SSP target descriptor	

Add this section to the protocol-specific parameters chapter:

Section 8.5.3.xx SAS SSP target descriptor

The target descriptor format shown in Table xxx is used to identify an EXTENDED COPY target using its SAS SSP device name (see SAS).

[Editor's note: "target" is not "target port" here, it's the misnamed EXTENDED COPY data source or destination. Maybe SPC-3 will rename target descriptors to "logical unit descriptors."]

Table xxx. SAS SSP target descriptor format

Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E9h)							
1	Reserved		NUL	PERIPHERAL DEVICE TYPE				
2	Reserved							
3								
4	LOGICAL UNIT NUMBER							
11								
12	DEVICE NAME							
19								
20	Reserved							
27								
28	Device type specific parameters							
31								

The DESCRIPTOR TYPE CODE, PERIPHERAL DEVICE TYPE and NUL fields and the device type specific parameters are described in 7.3.6.1.

The LOGICAL UNIT NUMBER field specifies the logical unit within the SCSI device addressed by the data in the DEVICE NAME field that shall be the target (source or destination) for EXTENDED COPY operations.

The DEVICE NAME field specifies the device identifier of the SCSI target device to be used when this target descriptor identifies the source or destination of an EXTENDED COPY operation.

F.4 Additional sense codes

Assign new additional sense codes for a sense key of ABORTED COMMAND under DATA PHASE ERROR (4Bh/00h) for:

a) 4Bh/01h ILLEGAL TARGET PORT TRANSFER TAG RECEIVED; ~~and~~
b) 4Bh/02h TOO MUCH WRITE DATA;~~.~~
c) 4Bh/03h ACK/NAK TIMEOUT; and
d) 4Bh/04h NAK RECEIVE.

Assign new additional sense codes for a sense key of ILLEGAL REQUEST:

a) TBD/00h INVALID INFORMATION UNIT;
b) TBD/01h INFORMATION UNIT TOO SHORT; and
c) TBD/02h INFORMATION UNIT TOO LONG.

G QUERY TASK task management function [SAM-3 proposal]

Create a new task management function called QUERY TASK with an I_T_L_Q argument.

Add QUERY TASK to table 30.

SAM-2 Section 6.1 Introduction

SAM-2 Table 30 — Task Management Functions

Task management function	Nexus	Reference
ABORT TASK	I_T_L_Q	6.2
ABORT TASK SET	I_T_L	6.3
CLEAR ACA	I_T_L	6.4
CLEAR TASK SET	I_T_L	6.5
LOGICAL UNIT RESET	I_T_L	6.6
TARGET RESET	I_T_L	6.7
WAKEUP	I_T	6.8
QUERY TASK [new row]	I_T_L_Q	6.x

The task manager response to task management requests is subject to the presence of access restrictions, as managed by ACCESS CONTROL OUT and ACCESS CONTROL IN commands (see SPC-3), as follows:

a) A task management request of ABORT TASK, ABORT TASK SET ~~or~~ CLEAR ACA*, or QUERY TASK* shall not be affected by the presence of access restrictions;

b) A task management request of CLEAR TASK SET or LOGICAL UNIT RESET received from an initiator that is denied access to the logical unit (either because it has no access rights or because it is in the pending-enrolled state) shall cause no change to the logical unit;

c) A TARGET RESET task management request shall initiate a logical unit reset as described in 5.8.7 for all logical units to which the initiator has access, and shall cause no change to any logical units to which the initiator is denied access; and

d) The task management function Service Response shall not be affected by the presence of access restrictions.

Add this section:

Section 6.x QUERY TASK [new section]

Function call:

Service Response = QUERY TASK (IN (I_T_L_Q Nexus))

Description:

Protocols may or may not support QUERY TASK and may or may not require logical units accessible through target ports using such protocols to support QUERY TASK.

The task manager shall return a response of FUNCTION COMPLETE if the specified task exists, or FUNCTION REJECTED if the specified task does not exist.