To: T10 Technical Committee
From: Rob Elliott, HP (elliott@hp.com)
Date: 6 September 2004
Subject: 04-167r1 SAS-1.1 Invalid dword handling

Revision history

Revision 0 (21 June 2004) First revision, split off from 04-115r1.
Revision 1 (6 September 2004) Incorporated comments from July SAS protocol WG.

Related documents

sas1r04 - Serial Attached SCSI 1.1 revision 4
04-172 SAS-1.1 More error counters

Overview

If ALIGNs or NOTIFYs are corrupted, but they are replaced in the data stream by expander(s) with valid ERROR primitives, the elasticity buffer of an expander or the frame recipient could overflow at a later time. If the expander determines that its elasticity buffer is full and it is expecting an ALIGN, it should be allowed to delete the invalid dword (assuming it might have been an ALIGN) rather than replace it with an ERROR. It should only do this when absolutely required, however, since an error in one dword might not show up until a subsequent dword (a characteristic of 8b10b).

If an expander is taking data into a HOLD/HOLDA buffer during an STP connection, it should not be required to remember each invalid dword. It should be allowed to combine any number of invalid dwords into one.

SAS requires that frames received with invalid dwords generate a NAK (inconsistently, but at least in one place), but allows incoming ERRORs to be either ignored and not generate a NAK or to generate a NAK. Invalid dwords and ERRORs should be allowed to be treated the same; there should be no functional difference if the error happened on the immediate physical link (resulting in an invalid dword being received) rather than on a remote physical link (resulting in an ERROR being received from an expander). In all cases, generation of the NAK due to the invalid dword or ERROR should be optional; devices can also hope their CRC checker will catch it.

Discussion of “invalid dwords and unexpected primitives” in the SL_CC state machine is inappropriate, since it only acts on messages. The SL receiver should send an Invalid Dword Received message to SL_CC to indicate that an invalid dword has shown up. Unexpected primitives are already handled by the convention that unexpected messages are simply ignored by of the state machines. Similar changes are proposed for the XL and SSP state machines.

The “default” crutch in the SL state machine overview about sending idle dwords when there is nothing else to send belongs in the SL transmitter section. Similar changes are proposed for the XL and SSP state machines.

XL needs a better description of how ERRORs get forwarded through the ECR; special messages are proposed.

The SMP state machines forgot to mention invalid dwords at all. The same handling as in the other state machines is proposed.

Suggested changes

3.1 Definitions [changes not highlighted in this section. Definitions are sorted by function.]

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

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

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

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

3.1.xx control byte: A byte containing control information defined in table 36 (see 6.2).

3.1.xx data byte: A byte containing data information defined in table 35 (see 6.2).

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

3.1.19 control character (Kxx.y): A character containing control information defined in table 36 (see 6.2).

3.1.22 data character (Dxx.y): A character containing data information defined in table 35 (see 6.2).

3.1.xx invalid character: A character that is not a control character (see 3.1.19) or a data character (see 3.1.22).

3.1.xx valid character: A control character (see 3.1.19) or a data character (see 3.1.22).

3.1.33 dword: A sequence of four contiguous bytes or four contiguous characters considered as a unit. The meaning depends on the context (e.g., when discussing the bits being transmitted over a physical link, dword represents four characters (i.e., 40 bits). When discussing the contents of a frame after 8b10b decoding, dword represents four bytes (i.e., 32 bits)).

3.1.23 data dword: A dword containing a) four data bytes, or b) four data characters with correct disparity.

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

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

3.1.184 valid dword: A data dword (see 3.1.23) or a primitive (see 3.1.93).

3.1.34 dword synchronization: Detection of an incoming stream of dwords from a physical link by a phy. See 6.8.

4.3.2 Transmit data path

...

4.3.n Receive data path

The SP_DWS receiver establishes dword synchronization and sends dwords to the SP_DWS state machine and to the link layer state machine receivers.

Figure 1 shows the receive data path in a SAS phy.

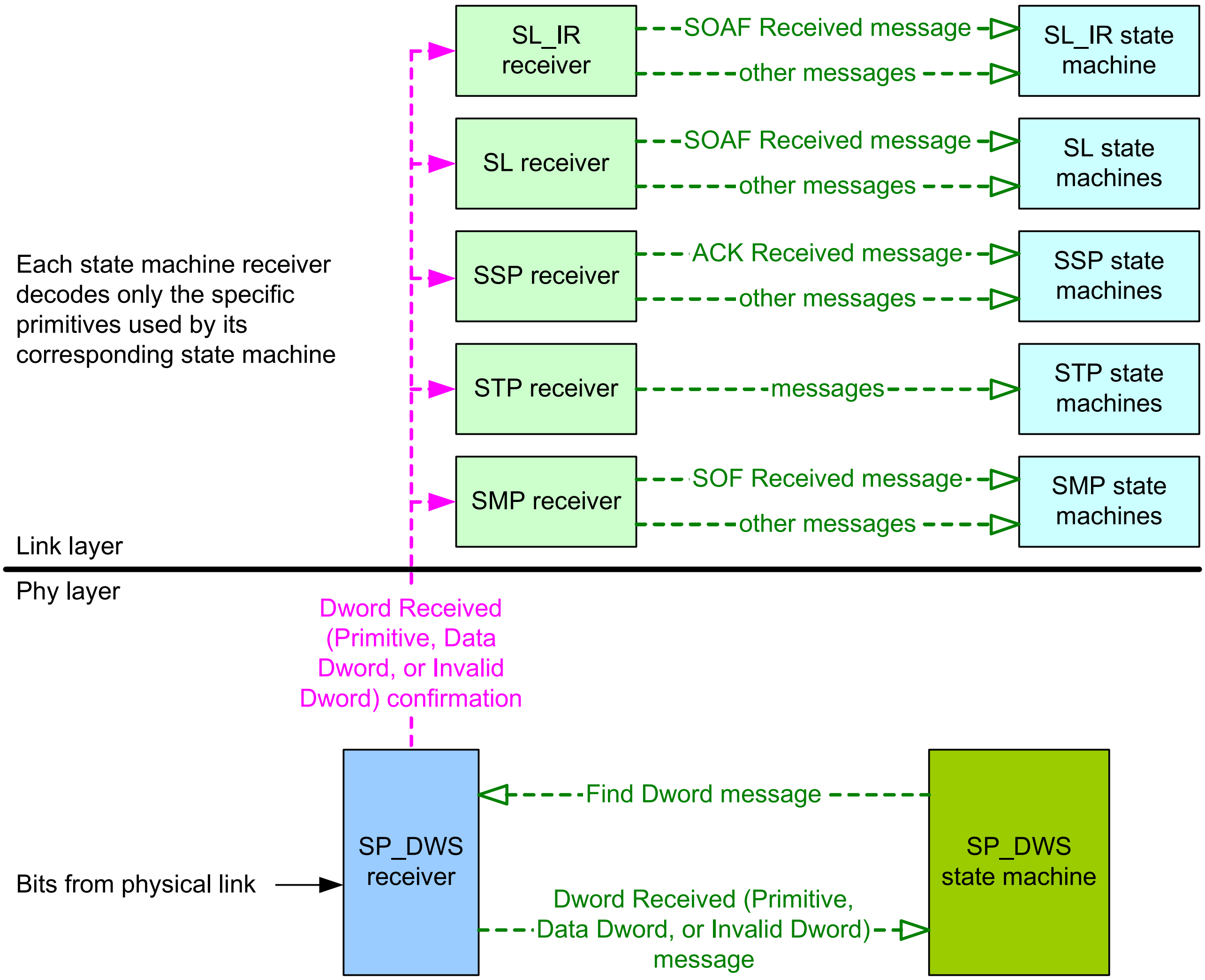

Figure 1 — Receive data path in a SAS phy

Figure 2 shows the receive data path in an expander phy.

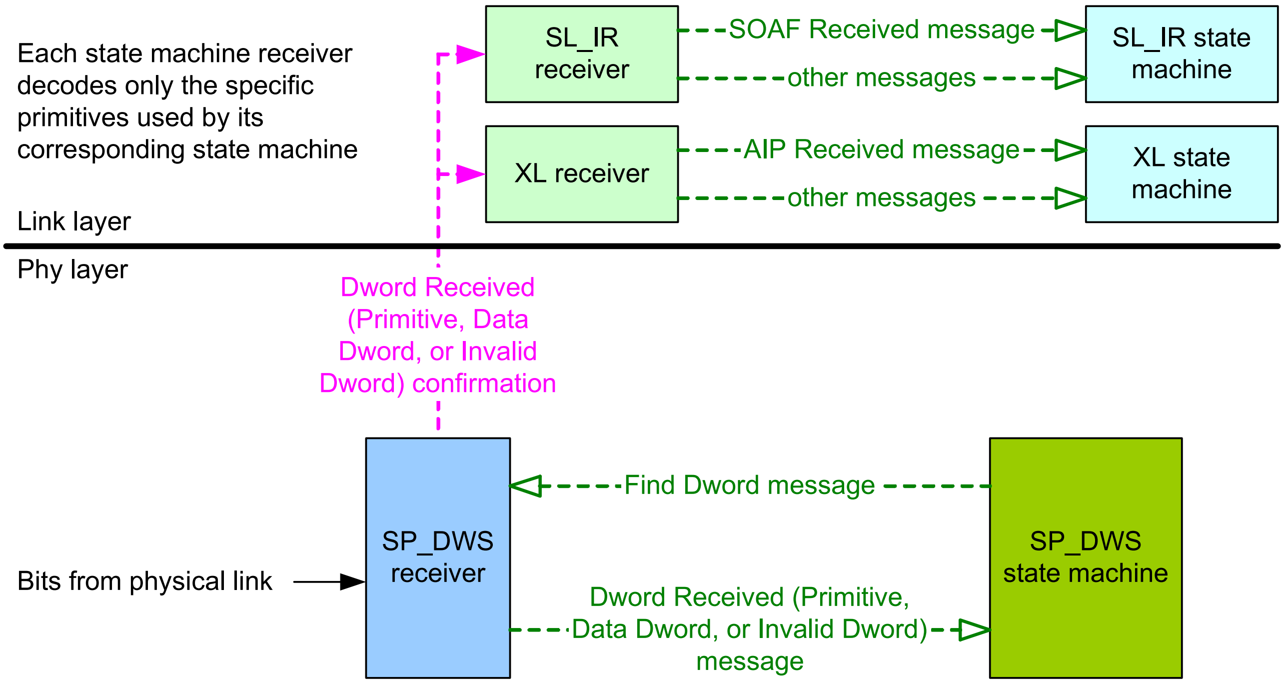

Figure 2 — Receive data path in an expander phy

6.2 ~~Encoding (8b10b)~~8b10b coding

6.2.1 ~~Encoding~~ 8b10b coding overview

All ~~data bytes~~information transferred in SAS ~~are~~ is encoded into 10-bit ~~data~~ characters using 8b10b encoding. Information includes data bytes (e.g., representing data in a frame) and control bytes (e.g., used for frame delimiters). ~~Additional characters not related to data bytes are called control characters.~~

Running disparity shall be maintained separately on each physical link. During a connection, 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~~ Phys within a device may or may not begin operation with the same disparity after the reset sequence.

6.2.2 8b10b coding introduction

Information to be transmitted across a physical link shall be encoded eight bits at a time into a 10-bit ~~transmission~~ character and then transmitted serially bit-by-bit across the physical link. Information received over the physical link shall be collected ten bits at a time, and those ~~transmission~~ characters that are used for data, called data characters, shall be decoded into the correct 8-bit ~~codes~~data bytes. The 10-bit characters support all 256 8-bit combinations. Some of the remaining 10-bit ~~transmission~~ characters, referred to as control characters, are used for functions that are to be distinguishable from the contents of a frame. The rest of the 10-bit characters are invalid characters.

~~The~~ 8b10b ~~en~~coding~~s defined by the transmission code~~ ensures that sufficient transitions are present in the serial bit stream to make clock recovery possible at the receiver. Such encoding also greatly increases the likelihood of detecting any single or multiple bit errors that may occur during transmission and reception of information. In addition, some of the control characters contain a distinct and easily recognizable bit pattern ~~(~~called a comma pattern~~)~~ which assists a receiver in achieving ~~word~~ character and dword alignment on the incoming bit stream.

6.2.3 8b10b coding notation conventions

This subclause uses letter notation for describing information bits and control variables. Such notation differs from the bit notation specified by the remainder of this standard. The following text describes the translation process between these notations and provides a translation example. It also describes the conventions used to name valid ~~transmission~~ characters. This text is provided for the purposes of terminology clarification only.

An unencoded information byte is composed of eight information bits A, B, C, D, E, F, G, H and the control variable Z. This information is encoded into the bits a, b, c, d, e, i, f, g, h, j of a 10-bit ~~transmission~~ character.

An information bit contains either a binary zero or a binary one. A control variable has either the value D or the value K. When the control variable associated with an unencoded information byte contains the value D, that byte is referred to as a ~~valid~~ data byte. When the control variable associated with an unencoded information byte contains the value K, that byte is referred to as a ~~valid~~ control byte.

The information bit labeled A corresponds to bit 0 in the numbering scheme of this standard, B corresponds to bit 1, and so on, as shown in table 1. Bit H is the most significant bit of the byte; bit A is the least significant bit of the byte.

Table 1 — Bit designations

Bit notation:	7	6	5	4	3	2	1	0	Control variable
Unencoded bit notation:	H	G	F	E	D	C	B	A	Z

Each valid ~~transmission~~ character has been given a name using the following convention:

Zxx.y

where:

a) Z is the control variable of the unencoded information byte. The value of Z is used to indicate whether the ~~transmission~~ character is a data character (Z = D) or a control character (Z = K);
b) xx is the decimal value of the binary number composed of the bits E, D, C, B, and A of the unencoded information byte in that order; and
c) y is the decimal value of the binary number composed of the bits H, G, and F of the unencoded information byte in that order.

Table 2 shows the conversion from byte notation to the ~~transmission~~ character naming convention~~ described above~~.

Table 2 — Conversion example

Byte notation	BCh
Bit notation	7 6 5 4 3 2 1 0 Control 1 0 1 1 1 1 0 0 K
Unencoded bit notation	H G F E D C B A Z 1 0 1 1 1 1 0 0 K
Unencoded bit notation reordered to conform with Zxx.y naming convention	Z E D C B A H G F K 1 1 1 0 0 1 0 1
~~Transmission~~ ~~c~~Character name	K 28 . 5

Most Kxx.y combinations do not result in valid ~~transmission~~ characters within the 8b10b coding scheme. Only those combinations that result in control characters as specified by table 4 are considered valid.

6.3 Character encoding and decoding

6.3.1 Introduction

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

6.3.2 Transmission order

Within the definition of the 8b10b ~~transmission~~ code, the bit positions of the ~~transmission~~ characters are labeled a, b, c, d, e, i, f, g, h, and j. Bit a shall be transmitted first, followed by bits b, c, d, e, i, f, g, h, and j, in that order. ~~Bit i shall be transmitted between bit e and bit f, rather than in the order that would be indicated by the letters of the alphabet.~~

NOTE 1 Bit i is transmitted between bit e and bit f, rather than in the order that would be indicated by the letters of the alphabet.

Characters within primitives shall be transmitted sequentially beginning with the control character used to distinguish the primitive (e.g., K28.3 or K28.5) and proceeding character by character from left to right within the definition of the primitive until all characters of the primitive are transmitted.

The contents of a frame shall be transmitted sequentially beginning with the primitive used to denote the start of frame and proceeding character-by-character from left to right within the definition of the frame until the primitive used to denote the end of frame is transmitted.

6.3.3 ~~Valid and invalid transmission~~ Data and control characters

~~6.3.3.1 Definitions~~

Table 35 and table 36 define the ~~valid~~ data characters (i.e.,Dxx.y characters) and ~~valid~~ control characters (i.e.,Kxx.y characters), respectively, and shall be used for both generating ~~valid~~ ~~transmission~~ characters (i.e., encoding) and checking the validity of received ~~transmission~~ characters (i.e., decoding). Each data character and control character entry has two columns that represent two (not necessarily different) ~~transmission~~ characters, corresponding to the current value of the running disparity. Running disparity is a binary parameter with ~~either the value~~a negative (-) or ~~the value~~ positive (+) value. ~~The running disparity at the beginning of a primitive is the beginning running disparity (beginning RD).~~

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

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

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

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

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

a) Running disparity at the end of any sub-block is positive if the sub-block contains more ones than zeros. It is also positive at the end of the six-bit sub-block if the six-bit sub-block is 000111b, and it is positive at the end of the four-bit sub-block if the four-bit sub-block is 0011b.
b) Running disparity at the end of any sub-block is negative if the sub-block contains more zeros than ones. It is also negative at the end of the six-bit sub-block if the six-bit sub-block is 111000b, and it is negative at the end of the four-bit sub-block if the four-bit sub-block is 1100b.

c) Otherwise, running disparity at the end of the sub-block is the same as at the beginning of the sub-block.

All sub-blocks with equal numbers of zeros and ones are ~~disparity~~ have neutral disparity (i.e., the ending disparity is the same as the beginning disparity). In order to limit the run length of zeros or ones between sub-blocks, the 8b10b ~~transmission~~ code rules specify that sub-blocks encoded as 000111b or 0011b are generated only when the running disparity at the beginning of the sub-block is positive; thus, running disparity at the end of these sub-blocks ~~shall also be~~is also positive. Likewise, sub-blocks containing 111000b or 1100b are generated only when the running disparity at the beginning of the sub-block is negative; thus, running disparity at the end of these sub-blocks ~~shall also be~~is also negative.

Table 35 defines the ~~valid~~ data characters ~~(Dxx.y characters)~~.

Table 3 — ~~Valid data~~ Data characters (part 1 of 5)

Name	Data byte		Data character	
	Binary representation (HGF EDCBA)	Hexadecimal representation	Current RD - abcdei fghj (binary)	Current RD + abcdei fghj (binary)
D00.0	000 00000	00h	100111 0100	011000 1011
D01.0	000 00001	01h	011101 0100	100010 1011
D02.0	000 00010	02h	101101 0100	010010 1011
D03.0	000 00011	03h	110001 1011	110001 0100
D04.0	000 00100	04h	110101 0100	001010 1011
D05.0	000 00101	05h	101001 1011	101001 0100
D06.0	000 00110	06h	011001 1011	011001 0100
D07.0	000 00111	07h	111000 1011	000111 0100
D08.0	000 01000	08h	111001 0100	000110 1011
D09.0	000 01001	09h	100101 1011	100101 0100
D10.0	000 01010	0Ah	010101 1011	010101 0100
D11.0	000 01011	0Bh	110100 1011	110100 0100
D12.0	000 01100	0Ch	001101 1011	001101 0100
D13.0	000 01101	0Dh	101100 1011	101100 0100
D14.0	000 01110	0Eh	011100 1011	011100 0100
D15.0	000 01111	0Fh	010111 0100	101000 1011
D16.0	000 10000	10h	011011 0100	100100 1011
D17.0	000 10001	11h	100011 1011	100011 0100
D18.0	000 10010	12h	010011 1011	010011 0100
D19.0	000 10011	13h	110010 1011	110010 0100
D20.0	000 10100	14h	001011 1011	001011 0100
D21.0	000 10101	15h	101010 1011	101010 0100
D22.0	000 10110	16h	011010 1011	011010 0100
D23.0	000 10111	17h	111010 0100	000101 1011
D24.0	000 11000	18h	110011 0100	001100 1011
D25.0	000 11001	19h	100110 1011	100110 0100
D26.0	000 11010	1Ah	010110 1011	010110 0100
D27.0	000 11011	1Bh	110110 0100	001001 1011
D28.0	000 11100	1Ch	001110 1011	001110 0100
D29.0	000 11101	1Dh	101110 0100	010001 1011
D30.0	000 11110	1Eh	011110 0100	100001 1011
D31.0	000 11111	1Fh	101011 0100	010100 1011
D00.1	001 00000	20h	100111 1001	011000 1001
D01.1	001 00001	21h	011101 1001	100010 1001
D02.1	001 00010	22h	101101 1001	010010 1001
D03.1	001 00011	23h	110001 1001	110001 1001
D04.1	001 00100	24h	110101 1001	001010 1001
D05.1	001 00101	25h	101001 1001	101001 1001
D06.1	001 00110	26h	011001 1001	011001 1001
D07.1	001 00111	27h	111000 1001	000111 1001
D08.1	001 01000	28h	111001 1001	000110 1001
D09.1	001 01001	29h	100101 1001	100101 1001

Table 3 — ~~Valid data~~ Data characters (part 2 of 5)

Name	Data byte		Data character	
	Binary representation (HGF EDCBA)	Hexadecimal representation	Current RD - abcdei fghj (binary)	Current RD + abcdei fghj (binary)
D10.1	001 01010	2Ah	010101 1001	010101 1001
D11.1	001 01011	2Bh	110100 1001	110100 1001
D12.1	001 01100	2Ch	001101 1001	001101 1001
D13.1	001 01101	2Dh	101100 1001	101100 1001
D14.1	001 01110	2Eh	011100 1001	011100 1001
D15.1	001 01111	2Fh	010111 1001	101000 1001
D16.1	001 10000	30h	011011 1001	100100 1001
D17.1	001 10001	31h	100011 1001	100011 1001
D18.1	001 10010	32h	010011 1001	010011 1001
D19.1	001 10011	33h	110010 1001	110010 1001
D20.1	001 10100	34h	001011 1001	001011 1001
D21.1	001 10101	35h	101010 1001	101010 1001
D22.1	001 10110	36h	011010 1001	011010 1001
D23.1	001 10111	37h	111010 1001	000101 1001
D24.1	001 11000	38h	110011 1001	001100 1001
D25.1	001 11001	39h	100110 1001	100110 1001
D26.1	001 11010	3Ah	010110 1001	010110 1001
D27.1	001 11011	3Bh	110110 1001	001001 1001
D28.1	001 11100	3Ch	001110 1001	001110 1001
D29.1	001 11101	3Dh	101110 1001	010001 1001
D30.1	001 11110	3Eh	011110 1001	100001 1001
D31.1	001 11111	3Fh	101011 1001	010100 1001
D00.2	010 00000	40h	100111 0101	011000 0101
D01.2	010 00001	41h	011101 0101	100010 0101
D02.2	010 00010	42h	101101 0101	010010 0101
D03.2	010 00011	43h	110001 0101	110001 0101
D04.2	010 00100	44h	110101 0101	001010 0101
D05.2	010 00101	45h	101001 0101	101001 0101
D06.2	010 00110	46h	011001 0101	011001 0101
D07.2	010 00111	47h	111000 0101	000111 0101
D08.2	010 01000	48h	111001 0101	000110 0101
D09.2	010 01001	49h	100101 0101	100101 0101
D10.2	010 01010	4Ah	010101 0101	010101 0101
D11.2	010 01011	4Bh	110100 0101	110100 0101
D12.2	010 01100	4Ch	001101 0101	001101 0101
D13.2	010 01101	4Dh	101100 0101	101100 0101
D14.2	010 01110	4Eh	011100 0101	011100 0101
D15.2	010 01111	4Fh	010111 0101	101000 0101
D16.2	010 10000	50h	011011 0101	100100 0101
D17.2	010 10001	51h	100011 0101	100011 0101
D18.2	010 10010	52h	010011 0101	010011 0101
D19.2	010 10011	53h	110010 0101	110010 0101
D20.2	010 10100	54h	001011 0101	001011 0101
D21.2	010 10101	55h	101010 0101	101010 0101
D22.2	010 10110	56h	011010 0101	011010 0101
D23.2	010 10111	57h	111010 0101	000101 0101
D24.2	010 11000	58h	110011 0101	001100 0101
D25.2	010 11001	59h	100110 0101	100110 0101
D26.2	010 11010	5Ah	010110 0101	010110 0101
D27.2	010 11011	5Bh	110110 0101	001001 0101
D28.2	010 11100	5Ch	001110 0101	001110 0101
D29.2	010 11101	5Dh	101110 0101	010001 0101
D30.2	010 11110	5Eh	011110 0101	100001 0101
D31.2	010 11111	5Fh	101011 0101	010100 0101
D00.3	011 00000	60h	100111 0011	011000 1100

Table 3 — ~~Valid data~~ Data characters (part 3 of 5)

Name	Data byte		Data character	
	Binary representation (HGF EDCBA)	Hexadecimal representation	Current RD - abcdei fghj (binary)	Current RD + abcdei fghj (binary)
D01.3	011 00001	61h	011101 0011	100010 1100
D02.3	011 00010	62h	101101 0011	010010 1100
D03.3	011 00011	63h	110001 1100	110001 0011
D04.3	011 00100	64h	110101 0011	001010 1100
D05.3	011 00101	65h	101001 1100	101001 0011
D06.3	011 00110	66h	011001 1100	011001 0011
D07.3	011 00111	67h	111000 1100	000111 0011
D08.3	011 01000	68h	111001 0011	000110 1100
D09.3	011 01001	69h	100101 1100	100101 0011
D10.3	011 01010	6Ah	010101 1100	010101 0011
D11.3	011 01011	6Bh	110100 1100	110100 0011
D12.3	011 01100	6Ch	001101 1100	001101 0011
D13.3	011 01101	6Dh	101100 1100	101100 0011
D14.3	011 01110	6Eh	011100 1100	011100 0011
D15.3	011 01111	6Fh	010111 0011	101000 1100
D16.3	011 10000	70h	011011 0011	100100 1100
D17.3	011 10001	71h	100011 1100	100011 0011
D18.3	011 10010	72h	010011 1100	010011 0011
D19.3	011 10011	73h	110010 1100	110010 0011
D20.3	011 10100	74h	001011 1100	001011 0011
D21.3	011 10101	75h	101010 1100	101010 0011
D22.3	011 10110	76h	011010 1100	011010 0011
D23.3	011 10111	77h	111010 0011	000101 1100
D24.3	011 11000	78h	110011 0011	001100 1100
D25.3	011 11001	79h	100110 1100	100110 0011
D26.3	011 11010	7Ah	010110 1100	010110 0011
D27.3	011 11011	7Bh	110110 0011	001001 1100
D28.3	011 11100	7Ch	001110 1100	001110 0011
D29.3	011 11101	7Dh	101110 0011	010001 1100
D30.3	011 11110	7Eh	011110 0011	100001 1100
D31.3	011 11111	7Fh	101011 0011	010100 1100
D00.4	100 00000	80h	100111 0010	011000 1101
D01.4	100 00001	81h	011101 0010	100010 1101
D02.4	100 00010	82h	101101 0010	010010 1101
D03.4	100 00011	83h	110001 1101	110001 0010
D04.4	100 00100	84h	110101 0010	001010 1101
D05.4	100 00101	85h	101001 1101	101001 0010
D06.4	100 00110	86h	011001 1101	011001 0010
D07.4	100 00111	87h	111000 1101	000111 0010
D08.4	100 01000	88h	111001 0010	000110 1101
D09.4	100 01001	89h	100101 1101	100101 0010
D10.4	100 01010	8Ah	010101 1101	010101 0010
D11.4	100 01011	8Bh	110100 1101	110100 0010
D12.4	100 01100	8Ch	001101 1101	001101 0010
D13.4	100 01101	8Dh	101100 1101	101100 0010
D14.4	100 01110	8Eh	011100 1101	011100 0010
D15.4	100 01111	8Fh	010111 0010	101000 1101
D16.4	100 10000	90h	011011 0010	100100 1101
D17.4	100 10001	91h	100011 1101	100011 0010
D18.4	100 10010	92h	010011 1101	010011 0010
D19.4	100 10011	93h	110010 1101	110010 0010
D20.4	100 10100	94h	001011 1101	001011 0010
D21.4	100 10101	95h	101010 1101	101010 0010
D22.4	100 10110	96h	011010 1101	011010 0010
D23.4	100 10111	97h	111010 0010	000101 1101

Table 3 — ~~Valid data~~ Data characters (part 4 of 5)

Name	Data byte		Data character	
	Binary representation (HGF EDCBA)	Hexadecimal representation	Current RD - abcdei fghj (binary)	Current RD + abcdei fghj (binary)
D24.4	100 11000	98h	110011 0010	001100 1101
D25.4	100 11001	99h	100110 1101	100110 0010
D26.4	100 11010	9Ah	010110 1101	010110 0010
D27.4	100 11011	9Bh	110110 0010	001001 1101
D28.4	100 11100	9Ch	001110 1101	001110 0010
D29.4	100 11101	9Dh	101110 0010	010001 1101
D30.4	100 11110	9Eh	011110 0010	100001 1101
D31.4	100 11111	9Fh	101011 0010	010100 1101
D00.5	101 00000	A0h	100111 1010	011000 1010
D01.5	101 00001	A1h	011101 1010	100010 1010
D02.5	101 00010	A2h	101101 1010	010010 1010
D03.5	101 00011	A3h	110001 1010	110001 1010
D04.5	101 00100	A4h	110101 1010	001010 1010
D05.5	101 00101	A5h	101001 1010	101001 1010
D06.5	101 00110	A6h	011001 1010	011001 1010
D07.5	101 00111	A7h	111000 1010	000111 1010
D08.5	101 01000	A8h	111001 1010	000110 1010
D09.5	101 01001	A9h	100101 1010	100101 1010
D10.5	101 01010	AAh	010101 1010	010101 1010
D11.5	101 01011	ABh	110100 1010	110100 1010
D12.5	101 01100	ACh	001101 1010	001101 1010
D13.5	101 01101	ADh	101100 1010	101100 1010
D14.5	101 01110	AEh	011100 1010	011100 1010
D15.5	101 01111	AFh	010111 1010	101000 1010
D16.5	101 10000	B0h	011011 1010	100100 1010
D17.5	101 10001	B1h	100011 1010	100011 1010
D18.5	101 10010	B2h	010011 1010	010011 1010
D19.5	101 10011	B3h	110010 1010	110010 1010
D20.5	101 10100	B4h	001011 1010	001011 1010
D21.5	101 10101	B5h	101010 1010	101010 1010
D22.5	101 10110	B6h	011010 1010	011010 1010
D23.5	101 10111	B7h	111010 1010	000101 1010
D24.5	101 11000	B8h	110011 1010	001100 1010
D25.5	101 11001	B9h	100110 1010	100110 1010
D26.5	101 11010	BAh	010110 1010	010110 1010
D27.5	101 11011	BBh	110110 1010	001001 1010
D28.5	101 11100	BCh	001110 1010	001110 1010
D29.5	101 11101	BDh	101110 1010	010001 1010
D30.5	101 11110	BEh	011110 1010	100001 1010
D31.5	101 11111	BFh	101011 1010	010100 1010
D00.6	110 00000	C0h	100111 0110	011000 0110
D01.6	110 00001	C1h	011101 0110	100010 0110
D02.6	110 00010	C2h	101101 0110	010010 0110
D03.6	110 00011	C3h	110001 0110	110001 0110
D04.6	110 00100	C4h	110101 0110	001010 0110
D05.6	110 00101	C5h	101001 0110	101001 0110
D06.6	110 00110	C6h	011001 0110	011001 0110
D07.6	110 00111	C7h	111000 0110	000111 0110
D08.6	110 01000	C8h	111001 0110	000110 0110
D09.6	110 01001	C9h	100101 0110	100101 0110
D10.6	110 01010	CAh	010101 0110	010101 0110
D11.6	110 01011	CBh	110100 0110	110100 0110
D12.6	110 01100	CCh	001101 0110	001101 0110
D13.6	110 01101	CDh	101100 0110	101100 0110
D14.6	110 01110	CEh	011100 0110	011100 0110

Table 3 — ~~Valid data~~ Data characters (part 5 of 5)

Name	Data byte		Data character	
	Binary representation (HGF EDCBA)	Hexadecimal representation	Current RD - abcdei fghj (binary)	Current RD + abcdei fghj (binary)
D15.6	110 01111	CFh	010111 0110	101000 0110
D16.6	110 10000	D0h	011011 0110	100100 0110
D17.6	110 10001	D1h	100011 0110	100011 0110
D18.6	110 10010	D2h	010011 0110	010011 0110
D19.6	110 10011	D3h	110010 0110	110010 0110
D20.6	110 10100	D4h	001011 0110	001011 0110
D21.6	110 10101	D5h	101010 0110	101010 0110
D22.6	110 10110	D6h	011010 0110	011010 0110
D23.6	110 10111	D7h	111010 0110	000101 0110
D24.6	110 11000	D8h	110011 0110	001100 0110
D25.6	110 11001	D9h	100110 0110	100110 0110
D26.6	110 11010	DAh	010110 0110	010110 0110
D27.6	110 11011	DBh	110110 0110	001001 0110
D28.6	110 11100	DCh	001110 0110	001110 0110
D29.6	110 11101	DDh	101110 0110	010001 0110
D30.6	110 11110	DEh	011110 0110	100001 0110
D31.6	110 11111	DFh	101011 0110	010100 0110
D00.7	111 00000	E0h	100111 0001	011000 1110
D01.7	111 00001	E1h	011101 0001	100010 1110
D02.7	111 00010	E2h	101101 0001	010010 1110
D03.7	111 00011	E3h	110001 1110	110001 0001
D04.7	111 00100	E4h	110101 0001	001010 1110
D05.7	111 00101	E5h	101001 1110	101001 0001
D06.7	111 00110	E6h	011001 1110	011001 0001
D07.7	111 00111	E7h	111000 1110	000111 0001
D08.7	111 01000	E8h	111001 0001	000110 1110
D09.7	111 01001	E9h	100101 1110	100101 0001
D10.7	111 01010	EAh	010101 1110	010101 0001
D11.7	111 01011	EBh	110100 1110	110100 1000
D12.7	111 01100	ECh	001101 1110	001101 0001
D13.7	111 01101	EDh	101100 1110	101100 1000
D14.7	111 01110	EEh	011100 1110	011100 1000
D15.7	111 01111	EFh	010111 0001	101000 1110
D16.7	111 10000	F0h	011011 0001	100100 1110
D17.7	111 10001	F1h	100011 0111	100011 0001
D18.7	111 10010	F2h	010011 0111	010011 0001
D19.7	111 10011	F3h	110010 1110	110010 0001
D20.7	111 10100	F4h	001011 0111	001011 0001
D21.7	111 10101	F5h	101010 1110	101010 0001
D22.7	111 10110	F6h	011010 1110	011010 0001
D23.7	111 10111	F7h	111010 0001	000101 1110
D24.7	111 11000	F8h	110011 0001	001100 1110
D25.7	111 11001	F9h	100110 1110	100110 0001
D26.7	111 11010	FAh	010110 1110	010110 0001
D27.7	111 11011	FBh	110110 0001	001001 1110
D28.7	111 11100	FCh	001110 1110	001110 0001
D29.7	111 11101	FDh	101110 0001	010001 1110
D30.7	111 11110	FEh	011110 0001	100001 1110
D31.7	111 11111	FFh	101011 0001	010100 1110

Table 36 defines the ~~valid~~ control characters ~~(Kxx.y characters)~~. Comma patterns, two bits of one polarity followed by five bits of the opposite polarity, are underlined.

Table 4 — ~~Valid control~~ Control characters

Name	Control byte		Control character	
	Binary representation (HGF EDCBA)	Hexadecimal representation	Current RD - abcdei fghj	Current RD + abcdei fghj
K28.0	000 11100	1Ch	001111 0100	110000 1011
K28.1	001 11100	3Ch	001111 1001	110000 0110
K28.2	010 11100	5Ch	001111 0101	110000 1010
K28.3	011 11100	7Ch	001111 0011	110000 1100
K28.4	100 11100	9Ch	001111 0010	110000 1101
K28.5	101 11100	BCh	001111 1010	110000 0101
K28.6	110 11100	DCh	001111 0110	110000 1001
K28.7	111 11100	FCh	001111 1000	110000 0111
K23.7	111 10111	F7h	111010 1000	000101 0111
K27.7	111 11011	FBh	110110 1000	001001 0111
K29.7	111 11101	FDh	101110 1000	010001 0111
K30.7	111 11110	FEh	011110 1000	100001 0111

NOTE 2 - .K28.1, K28.5, and K28.7 are the only valid characters which contain comma patterns. The K28.7 control character is not used because it introduces a false comma pattern when followed by any of the following control or data characters: K28.y, D3.y, D11.y, D12.y, D19.y, D20.y, or D28.y, where y is a value in the range 0 to 7, inclusive.

Only K28.3, K28.5 and K28.6 are used in this standard ~~(see 7.2)~~ as defined in table 32. See 7.2 for details on primitives.

Table 5 — Control character usage

First character of a dword	Usage in SAS physical links	Usage in SATA physical links
K28.3	Primitives used only inside STP connections	All primitives except ALIGN
K28.5	ALIGN and most primitives defined in this standard	ALIGN
K28.6	~~SATA_ERROR, used on SATA physical links~~Not used	~~Not used~~SATA_ERROR
~~Dxx.y~~	~~Data~~	~~Data~~

6.3.~~3.2~~x ~~Generating transmission~~ Encoding characters in the transmitter

~~The appropriate entry in table 35 or table 36 shall be found for the data byte or control byte for which a transmission character is to be generated (encoded). The current value of the transmitter's running disparity shall be used to select the transmission character from its corresponding column. For each transmission character transmitted, a new value of the running disparity shall be calculated. This new value shall be used~~

~~as the transmitter's current running disparity for the next valid data byte or control byte to be encoded and transmitted.~~

To transmit a data byte, the transmitter shall select the appropriate character from table 35 based on the current value of the transmitter's running disparity. To transmit a control byte, the transmitter shall select the appropriate character from table 36 based on the current value of the transmitter's running disparity. After the transmitting the character, the transmitter shall calculate a new value for its running disparity based on that character. This new value shall be used as the transmitter's current running disparity for the next character transmitted. This process is called 8b10b encoding.

6.3.~~3.3~~x ~~Validity of received~~ Decoding characters in the receiver

~~The columns in table 35 and table 36 corresponding to the current value of the receiver's running disparity shall be searched for each received transmission character. If the received transmission character is found in the proper column, then the transmission character shall be considered valid and the associated data byte or control byte determined (decoded).~~ After receiving a character, the receiver shall search the character columns in table 35 and table 36 corresponding to its current running disparity to determine the data byte or control byte to which the character corresponds. This process is called 10b8b decoding. If the received ~~transmission~~ character is not found in the proper column, then the ~~transmission~~ character shall be considered invalid ~~and the dword containing the character shall be considered an invalid dword~~. [new para]

~~Independent~~Regardless of the received ~~transmission~~ character's validity, the received ~~transmission~~ character shall be used to calculate a new value of running disparity in the receiver. [join para] This new value shall be used as the receiver's current running disparity for the next received ~~transmission~~ character.

Detection of a code violation does not necessarily indicate that the ~~transmission~~ character in which the code violation was detected is in error. Code violations may result from a prior error that altered the running disparity of the bit stream but did not result in a detectable error at the ~~transmission~~ character in which the error occurred. The example shown in table 37 exhibits this behavior. These errors may span dword boundaries. Expanders forwarding such a dword forward it as an ERROR (see 7.2.5.7).

Table 6 — Delayed code violation example

	RD	First character	RD	Second character	RD	Third character	RD
Transmitted character stream	-	D21.1	-	D10.2	-	D23.5	+
Transmitted bit stream	-	101010 1001	-	010101 0101	-	111010 1010	+
Bit stream after error	-	101010 1011 (error in second to last bit)	+	010101 0101	+	111010 1010	+
Decoded character stream	-	D21.0 (rather than D21.1; not detected as an error)	+	D10.2 (no error)	+	Code violation (although D23.5 was properly received)	+

6.4 Dwords [moved from 6.2.1]

All characters transferred in SAS are grouped into four-character sequences called dwords. [new para]

A primitive is a dword whose first character is a K28.3 or K28.5 control character and whose remaining three characters are data characters with correct disparity.

Primitives are defined with both negative and positive starting running disparity (see 6.3.3.1). SAS defines primitives starting with the K28.5 ~~and K28.6~~ control character~~s~~ (see 7.2). SATA defines primitives starting with the K28.3 and K28.5 control characters, which are used in SAS during STP connections (see 7.2).

~~Primitives are defined in 7.2.~~

A data dword is a dword ~~starting with a data character~~that contains four valid data characters with correct disparity.

A dword containing an invalid character shall be considered an invalid dword.

6.8 SP_DWS (phy layer dword synchronization) state machine

6.8.1 SP_DWS state machine overview

Each phy includes an SP_DWS state machine and SP_DWS receiver.

~~This~~ The SP_DWS state machine establishes the same dword boundaries at the receiver as at the attached transmitter by searching for control characters. ~~A receiver in the phy~~ The SP_DWS receiver monitors and decodes the incoming data stream and forces K28.5 characters into the first character position to effectively perform dword alignment when requested by the SP_DWS state machine. K28.5 characters with either disparity shall be accepted. The SP_DWS receiver continues to reestablish dword alignment by forcing received K28.5 characters into the first character position until a ~~valid~~ K28.5-based primitive (i.e., K28.5, Dxx.y, Dxx.y, Dxx.y) with correct disparity on each data character is detected. The resultant primitives, dwords and valid dword indicators (e.g., encoding error indicators) are sent to this state machine to enable it to determine the dword synchronization policy.

After dword synchronization has been achieved, this state machine monitors invalid dwords that are received. When an invalid dword is detected, it requires two valid dwords to nullify its effect. When four invalid dwords are detected without nullification, dword synchronization is considered lost.

While dword synchronization is lost, the data stream received is invalid and dwords shall not be passed to the link layer.

6.8.2 SP_DWS receiver

The SP_DWS receiver receives the following messages from the SP_DWS state machine:

a) Find Dword.

The SP_DWS receiver sends the following messages to the SP_DWS state machine:

a) Dword Received (~~Valid~~ Primitive);
b) Dword Received (~~Valid~~ Data Dword); and
c) Dword Received (Invalid).

The SP_DWS receiver also sends Dword Received confirmations to the link layer state machine receivers (e.g., SL_IR, SL, SSP, SMP, and XL).

[No change needed to figure 73]

Upon receiving a Find Dword message, the SP_DWS receiver shall monitor the input data stream and force each K28.5 character detected into the first character position ~~as~~ of a possible dword. If the next three characters are data characters with correct disparity, it shall send the dword as a Dword Received (~~Valid~~ Primitive) message to the SP_DWS state machine. Until it receives another Find Dword message, for every four characters it receives it shall:

a) send a Dword Received (Invalid) message to the SP_DWS state machine if the dword is an invalid dword (see 3.1.66);
b) send a Dword Received (~~Valid~~ Primitive) message to the SP_DWS state machine if the dword is a primitive (i.e., the dword contains a K28.5 character in the first character position followed by three data characters); or
c) send a Dword Received (~~Valid~~ Data Dword) message to the SP_DWS state machine if the dword is a data dword (i.e., is not an invalid dword or a primitive).

6.8.3 SP_DWS0:AcquireSync state

6.8.3.1 State description

This is the initial state of this state machine.

After receiving a Start DWS message, this state shall:

a) send a Find Dword message to the SP_DWS receiver; and
b) initialize and start the DWS Reset Timeout timer;

If this state is entered from SP_DWS1:Valid1 or SP_DWS2:Valid2, this state shall send a Find Dword message to the SP_DWS receiver. and the DWS Reset Timeout timer shall continue running.

If this state is entered from SP_DWS1:Valid1 or SP_DWS2:Valid2 and the DWS Reset Timeout timer has expired, this state may send a DWS Reset message to the SP state machine (e.g., if the phy chooses to initiate a new link reset sequence because dword synchronization has been lost for too long).

This state shall not send a DWS Reset message to the SP until the DWS Reset Timeout timer expires. If the DWS Reset Timeout timer expires, this state may send a DWS Reset message to the SP state machine.

6.8.3.2 Transition SP_DWS0:AcquireSync to SP_DWS1:Valid1

This transition shall occur after sending a Find Dword message and receiving a Dword Received (~~Valid~~ Primitive) message.

6.8.4 SP_DWS1:Valid1 state

6.8.4.1 State description

This state is reached after one valid primitive has been received. This state waits for a second valid primitive or an invalid dword.

The DWS Reset Timeout timer shall continue running.

6.8.4.2 Transition SP_DWS1:Valid1 to SP_DWS0:AcquireSync

This transition shall occur after receiving a Dword Received (Invalid) message or after the DWS Reset Timeout timer expires.

6.8.4.3 Transition SP_DWS1:Valid1 to SP_DWS2:Valid2

This transition shall occur after receiving a Dword Received (~~Valid~~ Primitive) message.

6.8.5 SP_DWS2:Valid2 state

6.8.5.1 State description

This state is reached after two valid primitives have been received without adjusting the dword synchronization. This state waits for a third valid primitive or an invalid dword.

The DWS Reset Timeout timer shall continue running.

6.8.5.2 Transition SP_DWS2:Valid2 to SP_DWS0:AcquireSync

This transition shall occur after receiving a Dword Received (Invalid) message or after the DWS Reset Timeout timer expires.

6.8.5.3 Transition SP_DWS2:Valid2 to SP_DWS3:SyncAcquired

This transition shall occur after receiving a Dword Received (~~Valid~~ Primitive) message.

6.8.6 SP_DWS3:SyncAcquired state

6.8.6.1 State description

This state is reached after three valid primitives have been received without adjusting the dword synchronization.

The most recently received primitive and all subsequent dwords shall be forwarded for processing by the link layer.

This state waits for a Dword Received (Invalid) message, which indicates that dword synchronization might be lost.

6.8.6.2 Transition SP_DWS3:SyncAcquired to SP_DWS4:Lost1

This transition shall occur after receiving a Dword Received (Invalid) message.

6.8.7 SP_DWS4:Lost1 state

6.8.7.1 State description

This state is reached when one invalid dword has been received and not nullified. This state waits for a Dword Received message.

6.8.7.2 Transition SP_DWS4:Lost1 to SP_DWS5:Lost1Recovered

This transition shall occur after receiving a Dword Received (~~Valid~~ Data Dword) message or a Dword Received (~~Valid~~ Primitive) message.

6.8.7.3 Transition SP_DWS4:Lost1 to SP_DWS6:Lost2

This transition shall occur after receiving a Dword Received (Invalid) message.

6.8.8 SP_DWS5:Lost1Recovered state

6.8.8.1 State description

This state is reached when a valid dword has been received after one invalid dword had been received. This state waits for a Dword Received message.

6.8.8.2 Transition SP_DWS5:Lost1Recovered to SP_DWS3:SyncAcquired

This transition shall occur after receiving a Dword Received (~~Valid~~ Data Dword) message or a Dword Received (~~Valid~~ Primitive) message.

6.8.8.3 Transition SP_DWS5:Lost1Recovered to SP_DWS6:Lost2

This transition shall occur after receiving a Dword Received (Invalid) message.

6.8.9 SP_DWS6:Lost2 state

6.8.9.1 State description

This state is reached when two invalid dwords have been received and not nullified. This state waits for a Dword Received message.

6.8.9.2 Transition SP_DWS6:Lost2 to SP_DWS7:Lost2Recovered

This transition shall occur after receiving a Dword Received (~~Valid~~ Data Dword) message or a Dword Received (Valid Primitive) message.

6.8.9.3 Transition SP_DWS6:Lost2 to SP_DWS8:Lost3

This transition shall occur after receiving a Dword Received (Invalid) message.

6.8.10 SP_DWS7:Lost2Recovered state

6.8.10.1 State description

This state is reached when a valid dword has been received after two invalid dwords had been received. This state waits for a Dword Received message.

6.8.10.2 Transition SP_DWS7:Lost2Recovered to SP_DWS4:Lost1

This transition shall occur after receiving a Dword Received (~~Valid~~ Data Dword) message or a Dword Received (~~Valid~~ Primitive) message.

6.8.10.3 Transition SP_DWS7:Lost2Recovered to SP_DWS8:Lost3

This transition shall occur after receiving a Dword Received (Invalid) message.

6.8.11 SP_DWS8:Lost3 state

6.8.11.1 State description

This state is reached when three invalid dwords have been received and not nullified. This state waits for aDword Received message.

If a Dword Received (Invalid) message is received (i.e., the fourth non-nullified invalid dword is received), this state shall send a DWS Lost message to the SP state machine.

6.8.11.2 Transition SP_DWS8:Lost3 to SP_DWS9:Lost3Recovered

This transition shall occur after receiving a Dword Received (~~Valid~~ Data Dword) message or a Dword Received (~~Valid~~ Primitive) message.

6.8.11.3 Transition SP_DWS8:Lost3 to SP_DWS0:AcquireSync

This transition shall occur after sending a DWS Lost message.

6.8.12 SP_DWS9:Lost3Recovered state

6.8.12.1 State description

This state is reached when a valid dword has been received after three invalid dwords had been received.

This state waits for a Dword Received message.

If a Dword Received (Invalid) message is received (i.e., the fourth non-nullified invalid dword is received), this state shall send a DWS Lost message to the SP state machine.

6.8.12.2 Transition SP_DWS9:Lost3Recovered to SP_DWS6:Lost2

This transition shall occur after receiving a Dword Received (~~Valid~~ Data Dword) message or a Dword Received (~~Valid~~ Primitive) message.

6.8.12.3 Transition SP_DWS9:Lost3Recovered to SP_DWS0:AcquireSync

This transition shall occur after sending a DWS Lost message.

7.2 Primitives

7.2.1 Primitives overview

Primitives are dwords whose first character is a K28.3~~,~~or K28.5~~, or K28.6 control character~~. Primitives are not considered big-endian or little-endian; they are just interpreted as first, second, third, and last characters. Table 7 defines the primitive format.

Table 7 — Primitive format

Character	Description
First	K28.5 ~~or K28.6~~ control character (for primitives defined in this standard) or K28.3 control character (for primitives defined by SATA).
Second	Constant data character.
Third	Constant data character.
Last	Constant data character.

...

7.2.2 Primitives summary

...

Table 8 lists the primitives used only inside STP connections and on SATA physical links.

Table 8 — Primitives used only inside STP connections and on SATA physical links

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

[a] The Use column indicates when the primitive is used:
a) STP: SAS physical links, inside STP connections; or
b) SATA: SATA physical links.

[b] The From and To columns indicate the type of ports that originate each primitive or are the intended destinations of each primitive:
a) I for STP initiator ports and SATA host ports;
b) E for expander ports; and
c) T for STP target ports and SATA device ports.
Expander ports are not considered originators of primitives that are passing through from expander port to expander port.

[c] The Primitive sequence type columns indicate whether the primitive is sent as a single primitive sequence, a repeated primitive sequence, a continued primitive sequence, a triple primitive sequence, or a redundant primitive sequence (see 7.2.4).

[d] Although listed with primitives, SATA_ERROR is not a primitive; it is an invalid dword used on SATA physical links (see 7.2.3).

7.2.3 Primitive encodings

...

Table 9 lists the primitive encodings for primitives used only inside STP connections and on SATA physical links.

Table 9 — Primitive encoding for primitives used only inside STP connections and on SATA physical links

Primitive	Character			
	1st	2nd	3rd	4th (last)
SATA_CONT	K28.3	D10.5	D25.4	D25.4
SATA_DMAT	K28.3	D21.5	D22.1	D22.1
SATA_EOF	K28.3	D21.5	D21.6	D21.6
SATA_ERROR [a] [b]	K28.6	D02.0	D01.4	D29.7
SATA_HOLD	K28.3	D10.5	D21.6	D21.6
SATA_HOLDA	K28.3	D10.5	D21.4	D21.4
SATA_PMACK	K28.3	D21.4	D21.4	D21.4
SATA_PMNAK	K28.3	D21.4	D21.7	D21.7
SATA_PMREQ_P	K28.3	D21.5	D23.0	D23.0
SATA_PMREQ_S	K28.3	D21.4	D21.3	D21.3
SATA_R_ERR	K28.3	D21.5	D22.2	D22.2
SATA_R_IP	K28.3	D21.5	D21.2	D21.2
SATA_R_OK	K28.3	D21.5	D21.1	D21.1
SATA_R_RDY	K28.3	D21.4	D10.2	D10.2
SATA_SOF	K28.3	D21.5	D23.1	D23.1
SATA_SYNC	K28.3	D21.4	D21.5	D21.5
SATA_WTRM	K28.3	D21.5	D24.2	D24.2
SATA_X_RDY	K28.3	D21.5	D23.2	D23.2

[a] Except for SATA_ERROR, all values are defined by SATA (see ATA/ATAPI-7 V3).
[b] SATA_ERROR is not technically a primitive since it starts with K28.6. It does not appear inside STP connections. It is an invalid dword, used by expander devices forwarding an error onto a SATA physical link (see 7.2.7.1).

7.2.5.7 ERROR

ERROR ~~is~~ should be sent by an expander device when it is forwarding dwords from a SAS physical link or SATA physical link to a SAS physical link and it receives an invalid dword or an ERROR.

NOTE 3 Since an 8b10b coding error in one dword might not be detected until the next dword, expander devices should avoid deleting invalid dwords or ERRORs unless necessary (e.g., if the elasticity buffer is full). Otherwise, they might hide evidence that an error has occurred.

See 7.15 for details on error handling by expander devices.

~~SAS phys may ignore ERROR or treat it as an invalid dword.~~

7.2.5.8 HARD_RESET

HARD_RESET is used to force a phy to generate a hard reset to its port. This primitive is only valid after the phy reset sequence without an intervening identification sequence (see 4.4) and shall be ignored at other times.

7.2.6.5 NAK (Negative acknowledgement)

NAK indicates the negative acknowledgement of an SSP frame and the reason for doing so.

The versions of NAK representing different reasons are defined in table 65.

Table 10 — NAK primitives

Primitive	Description
NAK (CRC ERROR)	The frame had a bad CRC or an invalid dword or ERROR was received during frame reception.
NAK (RESERVED 0)	Reserved. Processed the same as NAK (CRC ERROR).
NAK (RESERVED 1)	Reserved. Processed the same as NAK (CRC ERROR).
NAK (RESERVED 2)	Reserved. Processed the same as NAK (CRC ERROR).

7.2.7.1 SATA_ERROR

SATA_ERROR ~~is~~ may be sent by an expander device when it is forwarding dwords from a SAS physical link to a SATA physical link and it receives an invalid dword or an ERROR. SATA_ERROR is an invalid dword, not a primitive.

See 6.8 for details on error handling by expander devices.

7.9.5.4 SL_IR_RIF (receive IDENTIFY address frame) state machine

7.9.5.2 SL_IR transmitter and receiver

The SL_IR transmitter receives the following messages from the SL_IR state machines indicating primitive sequences, frames, and dwords to transmit:

a) Transmit IDENTIFY Address Frame;
b) Transmit HARD_RESET; and
c) Transmit Idle Dword.

The SL_IR transmitter sends the following messages to the SL_IR state machines:

a) HARD_RESET Transmitted; and
b) IDENTIFY Address Frame Transmitted.

The SL_IR receiver sends the following messages to the SL_IR state machines indicating primitive sequences and dwords received from the SP_DWS receiver (see 6.8.2):

a) SOAF Received;
b) EOAF Received;
c) Data Dword Received;
d) ERROR Received;
e) Invalid Dword Received; and
f) HARD_RESET Received.

The SL_IR receiver shall ignore all other dwords.

[Update figure 87 to match]

7.9.5.4 SL_IR_RIF (receive IDENTIFY address frame) state machine

7.9.5.4.1 SL_IR_RIF state machine overview

The SL_IR_RIF state machine receives an IDENTIFY address frame and checks the IDENTIFY address frame to determine if the frame should be accepted or discarded by the link layer.

This state machine consists of the following states:

a) SL_IR_RIF1:Idle (see 7.9.5.4.2)(initial state);
b) SL_IR_RIF2:Receive_Identify_Frame (see 7.9.5.4.3); and
c) SL_IR_RIF3:Completed (see 7.9.5.4.4).

This state machine shall start in the SL_IR_RIF1:Idle state. This state machine shall transition to the SL_IR_RIF1:Idle state from any other state after receiving a Phy Layer Not Ready confirmation.

7.9.5.4.2 SL_IR_RIF1:Idle state

7.9.5.4.2.1 State description

This state waits for an SOAF to be received from the physical link, indicating an address frame is arriving.

7.9.5.4.2.2 Transition SL_IR_RIF1:Idle to SL_IR_RIF2:Receive_Identify_Frame

This transition shall occur after both:

a) a Start SL_IR Receiver confirmation is received; and
b) an SOAF Received message is received.

7.9.5.4.3 SL_IR_RIF2:Receive_Identify_Frame state

7.9.5.4.3.1 State description

This state receives the dwords of an address frame and the EOAF.

If this state receives an SOAF Received message, then this state shall discard the address frame (i.e., the subsequent Data Dword Received and EOAF Received messages) and send an Address Frame Failed confirmation to the management application layer to indicate that an invalid IDENTIFY address frame was received.

If this state receives more than eight Data Dword Received messages after an SOAF Received message and before an EOAF Received message, then this state shall discard the address frame and send an Address Frame Failed confirmation to the management application layer to indicate that an invalid IDENTIFY address frame was received.

If this state receives an Invalid Dword Received message or an ERROR Received message after an SOAF Received message and before an EOAF Received message, then this state shall:

a) ignore the invalid dword or ERROR; or
b) discard the address frame and send an Address Frame Failed confirmation to the management application layer to indicate that an invalid IDENTIFY address frame was received.

After receiving an EOAF Received message, this state shall check if it the IDENTIFY address frame is valid.

This state shall accept an IDENTIFY address frame and send an Identify Received message to the SL_IR_IRC state machine if:

a) the ADDRESS FRAME TYPE field is set to Identify;
b) the number of bytes between the SOAF and EOAF is 32; and
c) the CRC field contains a valid CRC.

Otherwise, this state shall discard the IDENTIFY address frame and send an Address Frame Failed confirmation to the management application layer to indicate that an invalid IDENTIFY address frame was received.

7.9.5.4.3.2 Transition SL_IR_RIF2:Receive_Identify_Frame to SL_IR_RIF3:Completed

This transition shall occur after sending an Identify Received message or Address Frame Failed confirmation.

7.9.5.4.4 SL_IR_RIF3:Completed state

This state waits for a Phy Layer Not Ready confirmation.

7.14 SL (link layer for SAS phys) state machines

7.14.2 SL transmitter and receiver

The SL transmitter receives the following messages from the SL state machines specifying primitive sequences, frames, and dwords to transmit:

a) Transmit Idle Dword;
b) Transmit SOAF/Data Dwords/EOAF;
c) Transmit OPEN_ACCEPT;
d) Transmit OPEN_REJECT with an argument indicating the specific type (e.g., Transmit OPEN_REJECT (Retry));
e) Transmit BREAK;
f) Transmit BROADCAST; and
g) Transmit CLOSE with an argument indicating the specific type (e.g., Transmit CLOSE (Normal)).

When there is no outstanding message specifying a dword to transmit, the SL transmitter shall transmit idle dwords.

The SL transmitter sends the following messages ~~from~~ to the SL state machines based on dwords that have been transmitted:

a) SOAF/Data Dwords/EOAF Transmitted.

The SL receiver sends the following messages to the SL state machines indicating primitive sequences and dwords received from the SP_DWS receiver (see 6.8.2):

a) SOAF Received;
b) EOAF Received;
c) BROADCAST Received with an argument indicating the specific type (e.g., BROADCAST Received (Change));
d) BREAK Received;
e) OPEN_ACCEPT Received;
f) OPEN_REJECT Received with an argument indicating the specific type (e.g., OPEN_REJECT Received (No Destination));
g) AIP Received;
h) CLOSE Received with an argument indicating the specific type (e.g., CLOSE Received (Normal));
i) ERROR Received;
j) Data Dword Received; and
k) Invalid Dword Received;

The SL receiver shall ignore all other dwords.

...

[Update figure 92 and 93 to match]

7.14.3 SL_RA (receive OPEN address frame) state machine

The SL_RA state machine's function is to receive address frames and determine if the received address frame is an OPEN address frame and whether or not it was received successfully. This state machine consists of one state.

This state machine receives SOAFs, dwords of an OPEN address frames, and EOAFs.

This state machine shall ignore all messages except SOAF Received, Data Dword Received, and EOAF Received.

If this state machine receives a subsequent SOAF Received message after receiving an SOAF Received message but before receiving an EOAF Received message, then this state machine shall discard the Data Dword Received messages received before the subsequent SOAF Received message.

If this state machine receives more than eight Data Dword Received messages after an SOAF Received message and before an EOAF Received message, then this state machine shall discard the address frame. If this state machine receives an Invalid Dword Received message or an ERROR Received message after an SOAF Received message and before an EOAF Received message, then this state machine shall:

a) ignore the invalid dword or ERROR; or
b) discard the address frame.

After receiving an EOAF Received message, this state machine shall check if the address frame is a valid OPEN address frame.

This state machine shall accept an address frame ~~a valid OPEN address frame~~ if:

a) the ADDRESS FRAME TYPE field is set to Open;
b) the number of data dwords between the SOAF and EOAF is 8; and
c) the CRC field contains a valid CRC.

Otherwise, this state machine shall discard the address frame. If the frame is not discarded then this state machine shall send a OPEN Address Frame Received message to the SL_CC0:Idle state and the SL_CC1:ArbSel state with an argument that contains all the data dwords received in the OPEN address frame.

7.14.4 SL_CC (connection control) state machine

7.14.4.1 SL_CC state machine overview

...

~~Unless otherwise stated within the state description, all invalid dwords and unexpected primitives (i.e., any primitive not described in the description of the SL_CC state) received within any SL state shall be ignored and idle dwords shall be transmitted.~~

Any message received by a state that does not describe handling such a message shall be ignored.

7.15 XL (link layer for expander phys) state machine

7.15.1 XL state machine overview

~~...~~

~~Unless otherwise stated within a state description, all invalid dwords and unexpected primitives received within any XL state shall be ignored.~~

Any message received by a state that does not describe handling such a message shall be ignored.

7.15.2 XL transmitter and receiver

The XL transmitter receives the following messages from the XL state machine ~~indicating~~ specifying primitive sequences, frames, and dwords to transmit:

a) Transmit Idle Dword;
b) Transmit AIP with an argument indicating the specific type (e.g., Transmit AIP (Normal));
c) Transmit BREAK;
d) Transmit BROADCAST with an argument indicating the specific type (e.g., Transmit BROADCAST (Change));
e) Transmit CLOSE with an argument indicating the specific type (e.g., Transmit CLOSE (Normal));
f) Transmit OPEN_ACCEPT;
g) Transmit OPEN_REJECT, with an argument indicating the specific type (e.g., Transmit OPEN_REJECT (No Destination));
h) Transmit OPEN Address Frame; and
i) Transmit Dword.

The XL transmitter sends the following messages to the XL state machine based on dwords that have been transmitted:

a) a) OPEN Address Frame Transmitted.

...

When there is no outstanding message specifying a dword to transmit, the XL transmitter shall transmit idle dwords.

The XL receiver sends the following messages to the XL state machine indicating primitive sequences, frames, and dwords received from the SP_DWS receiver (see 6.8.2):

a) AIP Received with an argument indicating the specific type (e.g., AIP Received (Normal));
b) BREAK Received;
c) BROADCAST Received;
d) CLOSE Received;
e) OPEN_ACCEPT Received;
f) OPEN_REJECT Received;
g) OPEN Address Frame Received;
h) Dword Received with an argument indicating the valid data dword or valid primitive received; and
i) Invalid Dword Received.

The XL receiver shall ignore all other dwords.

While receiving an address frame, if the XL receiver receives an invalid dword or ERROR, then the XL receiver shall:

a) ignore the invalid dword or ERROR; or
b) discard the address frame.

[Update figure 94, 95, and 96 to match]

7.15.10 XL7:Connected state

7.15.10.1 State description

...

If:

a) an ~~invalid dword is received with the~~ Invalid Dword Received message is received; and
b) the expander phy is forwarding to an expander phy attached to a SAS physical link,

the expander phy shall:

a) send an ERROR primitive with the Transmit Dword request instead of the invalid dword; or
b) delete the invalid dword.

If:

a) an ~~invalid dword or an~~ ERROR primitive is received with the Dword Received message or an Invalid Dword Received message is received; and
b) the expander phy is forwarding to an expander phy attached to a SATA physical link,

the expander phy shall:

a) send a SATA_ERROR primitive with the Transmit Dword request instead of the invalid dword or ERROR primitive; or
b) delete the ERROR primitive or invalid dword.

7.15.11 XL8:Close_Wait state

7.15.11.1 State description

If:

a) an ~~invalid dword is received with the~~ Invalid Dword Received message is received;
b) and the expander phy is forwarding to an expander phy attached to a SAS physical link,

the expander phy shall:

a) send an ERROR primitive with the Transmit Dword request instead of the invalid dword; or
b) delete the invalid dword.

If:

a) an ~~invalid dword or an~~ ERROR primitive is received with the Dword Received message or an Invalid Dword Received message is received; and
b) the expander phy is forwarding to an expander phy attached to a SATA physical link,

the expander phy shall:

a) send a SATA_ERROR primitive with the Transmit Dword request instead of the invalid dword or ERROR primitive; or
b) delete the invalid dword or ERROR primitive.

...

7.16.3 SSP frame transmission and reception

...

Receiving SSP phys shall acknowledge SSP frames within 1 ms if not discarded as described in 7.16.7.7 with either a positive acknowledgement (ACK) or a negative acknowledgement (NAK). ACK means the SSP frame was received into a frame buffer without errors. NAK (CRC ERROR) means the SSP frame was received with a CRC error, an invalid dword, or an ERROR primitive.

> NOTE 4 It is not required that frame recipients generate NAK (CRC ERROR) from invalid dwords and ERRORs (see 7.16.7.2).

...

7.16.7 SSP (link layer for SSP phys) state machines

7.16.7.1 SSP state machines overview

[In Figure 99, add Invalid Dword message into SSP_RF.]

7.16.7.2 SSP transmitter and receiver

The SSP transmitter receives the following messages from the SSP state machines ~~indicating~~ specifying primitive sequences and frames to transmit:

a) Transmit RRDY with an argument indicating the specific type (e.g., Transmit RRDY (Normal));
b) Transmit CREDIT_BLOCKED;
c) Transmit ACK;
d) Transmit NAK with an argument indicating the specific type (e.g., Transmit NAK (CRC Error));
e) Transmit Frame (i.e., SOF/data dwords/EOF); and
f) Transmit DONE with an argument indicating the specific type (e.g., Transmit DONE (Normal)).

The SSP transmitter sends the following messages to the SSP state machines based on dwords that have been transmitted:

a) DONE Transmitted;
b) RRDY Transmitted;
c) CREDIT_BLOCKED Transmitted;
d) ACK Transmitted;
e) NAK Transmitted; and
f) Frame Transmitted.

~~When the SSP transmitter is not processing a message to transmit, it shall transmit idle dwords.~~

When there is no outstanding message specifying a dword to transmit, the SSP transmitter shall transmit idle dwords.

The SSP receiver sends the following messages to the SSP state machines indicating primitive sequences and dwords received from the SP_DWS receiver (see 6.8.2):

a) ACK Received;
b) NAK Received;

c) RRDY Received;
d) CREDIT_BLOCKED Received;
e) DONE Received with an argument indicating the specific type (e.g., DONE Received (Normal));
f) SOF Received;
g) EOF Received;
h) ERROR Received;
i) Data Dword Received; and
j) Invalid Dword Received.
k) EOF Received.

The SSP receiver shall ignore all other dwords.

[Update figure 102 and 103 to match]

7.16.7.7 SSP_RF (receive frame control) state machine

...

The SSP_RF state machine's function is to receive frames and determine whether or not those frames were received successfully. This state machine consists of one state.

This state machine:

a) checks the frame to determine if the frame should be accepted or discarded;
b) checks the frame to determine if an ACK or NAK should be transmitted; and
c) sends a Frame Received confirmation to the port layer.

If consecutive SOF Received messages are received without an intervening EOF Received message (i.e., SOF, data dwords, SOF, data dwords, and EOF instead of SOF, data dwords, EOF, SOF, data dwords, and EOF) then this state machine shall discard all dwords between those SOFs.

If this state receives a subsequent SOF Received message after receiving an SOF Received message but before receiving an EOF Received message, then this state shall discard the Data Dword Received messages received before the subsequent SOF Received message.

The frame (i.e., all the dwords between an SOF and EOF) shall be discarded if any of the following conditions are true:

a) the number of data dwords between the SOF and EOF is less than 7;
b) the number of data dwords after the SOF is greater than 263 data dwords;
c) the Rx Credit Status (Credit Exhausted) message is received; or
d) the DONE Received message is received.

This state shall discard the frame if

a) this state receives more than 263 Data Dword Received messages after an SOF Received message and before an EOF Received message;
b) this state receives less than 7 Data Dword Received messages after an SOF Received message and before an EOF Received message,
c) this state receives an Rx Credit Status (Credit Exhausted) message; or
d) this state receives a DONE Received message.

If this state receives an Invalid Dword Received message or an ERROR Received message after receiving an SOF Received message and before receiving an EOF Received message, then this state machine shall:

a) ignore the invalid dword or ERROR; or
b) discard the frame, send a Frame Received message to the SSP_RCM state machine, send a Frame Received message to the SSP_RIM state machine, and send a Frame Received (Unsuccessful) message to the SSP_TAN1:Idle state.

If the frame is discarded then no further action is taken by this state machine. If the frame is not discarded then If the frame is not discarded and the frame CRC is bad, this state machine shall:

a) send a Frame Received message to the SSP_RCM state machine; and
b) send a Frame Received message to the SSP_RIM state machine; and

c) send a Frame Received (Unsuccessful) message to the SSP_TAN1:Idle state.

If the frame is not discarded and the frame CRC is good ~~and the frame contained no invalid data dwords~~, this state machine shall:

a) send a Frame Received message to the SSP_RCM state machine;
b) send a Frame Received message to the SSP_RIM state machine;
c) send a Frame Received (Successful) message to the SSP_TAN1:Idle state; and:
    A) if the last Rx Balance Status message received had an argument of Balanced, send a Frame Received (ACK/NAK Balanced) confirmation to the port layer; or
    B) if the last Rx Balance Status message received had an argument of Not Balanced, send a Frame Received (ACK/NAK Not Balanced) confirmation to the port layer.

~~If the frame CRC is bad or the frame contained invalid data dwords, this state machine shall send a Frame Received (Unsuccessful) message to the SSP_TAN1:Idle state.~~

7.16.7.11 SSP_TAN (transmit ACK/NAK control) state machine

...

Any time this state machine receives a Frame Received (Unsuccessful) message it shall send a Transmit NAK (CRC Error) message to the SSP transmitter.

7.18 SMP link layer

7.18.1 SMP frame transmission and reception

Inside an SMP connection, the source device transmits a single SMP_REQUEST frame and the destination device responds with a single SMP_RESPONSE frame (see 9.4).

Frames are surrounded by SOF and EOF as shown in figure 104. ~~There is no acknowledgement of SMP frames with ACK and NAK. There is no credit exchange with RRDY.~~ See 7.18.4 for error handling details.

> NOTE 5 Unlike SSP, there is no acknowledgement of SMP frames with ACK and NAK and there is no credit exchange with RRDY.

...

The last data dword after the SOF prior to the EOF always contains a CRC (see 7.5). The SMP link layer state machine checks that the frame is not too short and that the CRC is valid (see 7.18.4).

7.18.4.2 SMP transmitter and receiver

The SMP transmitter receives the following messages from the SMP state machines ~~indicating~~ specifying dwords and frames to transmit:

a) Transmit Idle Dword; and
b) Transmit Frame.

The SMP transmitter sends the following messages to the SMP state machines:

a) Frame Transmitted.

When there is no outstanding message specifying a dword to transmit, the SMP transmitter shall transmit idle dwords.

The SMP receiver sends the following messages to the SMP state machines indicating primitive sequences and dwords received from the SP_DWS receiver (see 6.8.2):

a) SOF Received;
b) EOF Received;
c) Data Dword Received; and
d) Invalid Dword Received.

The SMP receiver shall ignore all other dwords.

[Update figure 109 and 110 to match]

7.18.4.3.4 SMP_IP3:Receive_Frame state

This state checks the SMP response frame and determines if the SMP response frame was successfully received (e.g., no CRC error).

If this state receives a subsequent SOF Received message after receiving an SOF Received message but before receiving an EOF Received message, then this state shall discard the Data Dword Received messages received before the subsequent SOF Received message.

This state shall discard the frame and send a Frame Received (SMP Failure) confirmation to the port layer if:

a) this state receives more than 258 Data Dword Received messages after an SOF Received message and before an EOF Received message; or
b) this state receives less than 2 Data Dword Received messages after an SOF Received message and before an EOF Received message.

If this state receives an Invalid Dword Received message or an ERROR Received message after an SOF Received message and before an EOF Received message, then this state machine shall:

a) ignore the invalid dword or ERROR; or
b) discard the frame and send a Frame Received (SMP Failure) confirmation to the port layer.

If the SMP response frame is received with a CRC error, this state shall discard the frame and send a Frame Received (SMP Failure) confirmation to the port layer.

~~If the number of dwords between the SOF and EOF of the SMP response frame is less than 2, or the number of dwords after an SOF is greater than 258, this state shall send a Frame Received (SMP Failure) confirmation to the port layer.~~

If the SMP response frame is received with no CRC error and the SMP response frame is valid, this state shall:

a) send a Frame Received confirmation to the port layer; and
b) send a Request Close message to the SL state machines (see 7.14).

If an SMP Transmit Break request is received, this state shall send a Request Break message to the SL state machines and terminate.

This state shall request idle dwords be transmitted by repeatedly sending Transmit Idle Dword messages to the SMP transmitter.

7.18.4.4.2 SMP_TP1:Receive_Frame state

7.18.4.4.2.1 State description

This state waits for an SMP frame and determines if the SMP frame was successfully received (e.g., no CRC error).

If this state receives an Invalid Dword Received message or an ERROR Received message after receiving an SOF Received message and before receiving an EOF Received message, then this state shall discard the Data Dword Received messages received before the subsequent SOF Received message.

This state shall discard the frame, send a Request Break message to the SL state machines (see 7.14) and shall terminate the state machine if:

a) this state receives more than 258 Data Dword Received messages after an SOF Received message and before an EOF Received message; or
b) this state receives less than 2 Data Dword Received messages after an SOF Received message and before an EOF Received message.

If this state receives an Invalid Dword Received message or an ERROR Received message after an SOF Received message and before an EOF Received message, then this state machine shall:

a) ignore the invalid dword or ERROR; or
b) discard the frame, send a Request Break message to the SL state machines (see 7.14) and shall terminate the state machine.

If the SMP request frame is received with a CRC error, this state shall discard the frame, send a Request Break message to the SL state machines (see 7.14) and shall terminate the state machine.

~~If an SMP frame is received, this state shall send a Request Break message to the SL state machines (see 7.14) and terminate if:~~

~~a) the SMP frame has a CRC error;~~
~~b) the number of data dwords between the SOF and EOF is less than 2; or~~
~~c) the number of data dwords after the SOF is greater than 258.~~

Otherwise, this state shall send a Frame Received confirmation to the port layer.

This state shall request idle dwords be transmitted by repeatedly sending Transmit Idle Dword messages to the SMP transmitter.