119. Physical Coding Sublayer (PCS) for 64B/66B, type 400GBASE-R

119.1 Overview

119.1.1 Scope

This clause specifies the Physical Coding Sublayer (PCS) that is used for the Physical Layer implementation known as 400GBASE-R. The 400GBASE-R PCS is a sublayer of the 400 Gb/s PHYs listed in Table 116–1. The term 400GBASE-R is used when referring generally to Physical Layers using the PCS defined in this clause.

400GBASE-R is based on a 64B/66B code. The 64B/66B code supports transmission of data and control characters. The 64B/66B code is then transcoded to 256B/257B encoding to reduce the overhead and make room for Forward Error Correction (FEC). The 256B/257B encoded data is then FEC encoded before being transmitted. Data distribution is introduced to support multiple lanes in the Physical Layer. Part of the distribution includes the periodic insertion of an alignment marker, which allows the receive PCS to align data from multiple lanes.

119.1.2 Relationship of 400GBASE-R to other standards

Figure 119–1 depicts the relationship of the 400GBASE-R sublayer (shown shaded), the Ethernet MAC and Reconciliation Sublayers, and the higher layers.

This clause borrows heavily from Clause 82 and Clause 91. 64B/66B encoding is reused with the addition of transcoding, and RS-FEC.

119.1.3 Physical Coding Sublayer (PCS)

The PCS service interface is the Media Independent Interface (CDMII), which is defined in Clause 116. The CDMII provides a uniform interface to the Reconciliation Sublayer for all 400 Gb/s PHY implementations.

The 400GBASE-R PCS provides all services required by the CDMII, including the following:

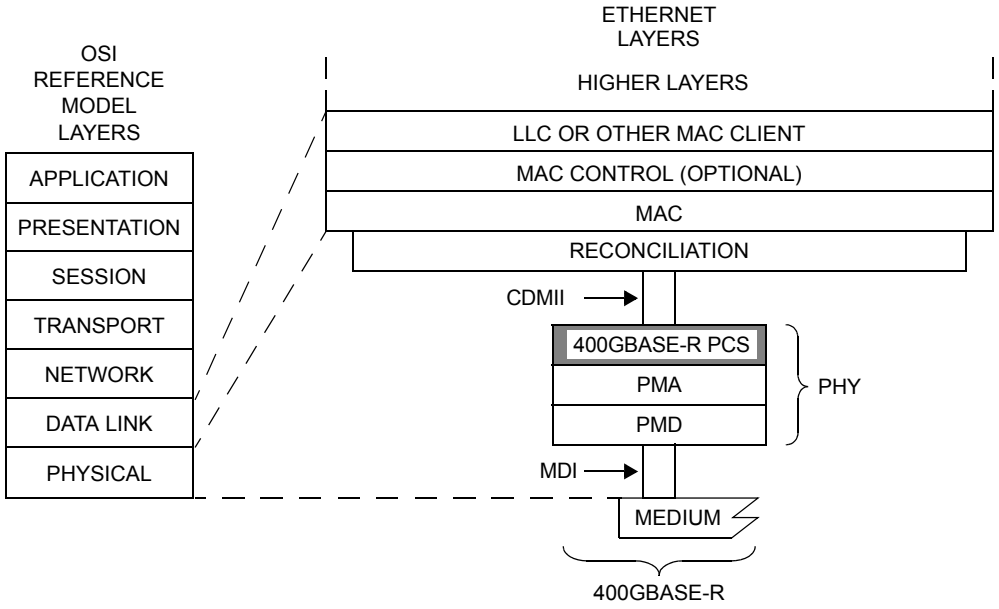
a) Encoding (decoding) of eight CDMII data octets to (from) 66-bit blocks (64B/66B).
b) Transcoding from 66-bit blocks to 257-bit blocks
c) Reed-Solomon encoding (decoding) the 257-bit blocks
d) Transferring encoded data to (from) the PMA.
e) Compensation for any rate differences caused by the insertion or deletion of alignment markers or due to any rate difference between the CDMII and PMA through the insertion or deletion of idle control characters.
f) Determining when a functional link has been established and informing the management entity via the MDIO when the PHY is ready for use.

119.1.4 Inter-sublayer interfaces

The upper interface of the PCS may connect to the Reconciliation Sublayer through the CDMII. The lower interface of the PCS connects to the PMA sublayer to support a PMD. The 400GBASE-R PCS has a nominal rate at the PMA service interface of 26.5625 Gtransfers/s on each of 16 PCS lane, which provides capacity for the MAC data rate of 400 Gb/s.

It is important to note that, while this specification defines interfaces in terms of bits, octets, and frames, implementations may choose other data-path widths for implementation convenience.

Copyright © 2015 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.



CDMII = 400 Gb/s MEDIA INDEPENDENT INTERFACE
LLC = LOGICAL LINK CONTROL
MAC = MEDIA ACCESS CONTROL
MDI = MEDIUM DEPENDENT INTERFACE
PCS = PHYSICAL CODING SUBLAYER
PHY = PHYSICAL LAYER DEVICE
PMA = PHYSICAL MEDIUM ATTACHMENT
PMD = PHYSICAL MEDIUM DEPENDENT

Figure 119–1—400GBASE-R PCS relationship to the ISO/IEC Open Systems Interconnection (OSI) reference model and IEEE 802.3 Ethernet model

119.1.4.1 PCS service interface (CDMII)

The PCS service interface allows the 400GBASE-R PCS to transfer information to and from a PCS client. The PCS client is the Reconciliation Sublayer. The PCS Service Interface is precisely defined as the Media Independent Interface (CDMII) in Clause 116.

119.1.4.2 Physical Medium Attachment (PMA) service interface

The PMA service interface for the PCS is described in an abstract manner and does not imply any particular implementation. The PMA Service Interface supports the exchange of encoded data between the PCS and PMA sublayer. The PMA service interface is defined in 120.3 and is an instance of the inter-sublayer service interface definition in 116.3.

Copyright © 2015 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

119.1.5 Functional block diagram

Figure 119–2 provides a functional block diagram of the 400GBASE-R PCS.

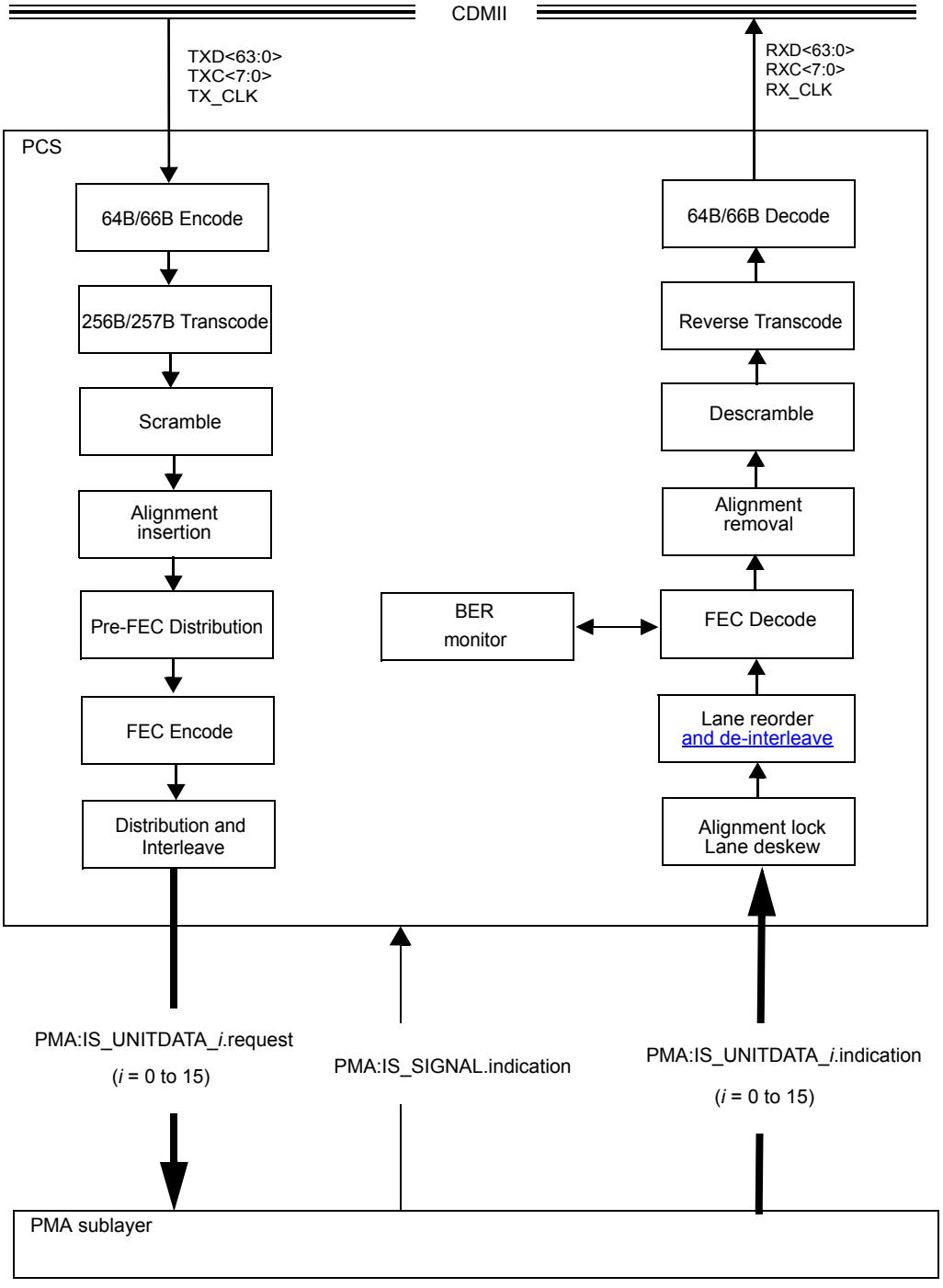


Figure 119–2—Functional block diagram

Copyright © 2015 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

119.2 Physical Coding Sublayer (PCS)

119.2.1 Functions within the PCS

The 400GBASE-R PCS is composed of the PCS Transmit and PCS Receive processes. The PCS shields the Reconciliation Sublayer (and MAC) from the specific nature of the underlying channel. The PCS transmit channel and receive channel can each operate in normal mode or test-pattern mode.

When communicating with the CDMII, the PCS uses an eight octet-wide, synchronous data path, with packet delineation being provided by transmit control signals (TXC<n> = 1) and receive control signals (RXC<n> = 1). When communicating with the PMA, the 400GBASE-R PCS uses 16 encoded bit streams. The PMA sublayer operates independently of block and packet boundaries. The PCS provides the functions necessary to map packets between the CDMII format and the PMA service interface format.

Note that these serial streams originate from a common clock in each direction, but may vary in phase and skew dynamically.

When the transmit channel is in normal mode, the PCS Transmit process continuously generates 66-bit blocks based upon the TXD <63:0> and TXC <7:0> signals on the CDMII. The 66-bit blocks are transcoded to 257-bit blocks to reduce the line coding overhead. The transcoded blocks are then scrambled to control clock content and baseline wander. Alignment markers are periodically added to the data stream. The data stream is distributed to two FEC codewords and ~~the stream is~~ then FEC encoded to control errors. ~~The~~Two FEC ~~encoded~~codewords are then interleaved before data is~~data is then~~ distributed to individual PCS lanes.

~~[Editor's note: How FEC codewords are distributed is TBD, and if multiple FEC codewords are interleaved before distribution is TBD.]~~

Transmit data-units are sent to the service interface via the PMA:IS_UNITDATA_*i*.request primitive.

When the transmit channel is in test-pattern mode, a test pattern is packed into the transmit data-units that are sent to the PMA service interface via the PMA:IS_UNITDATA_*i*.request primitive.

When the receive channel is in normal or test-pattern mode, the PCS Synchronization process continuously monitors PMA:IS_SIGNAL.indication(SIGNAL_OK). When SIGNAL_OK indicates OK, then the PCS synchronization process accepts data-units via the PMA:IS_UNITDATA_*i*.indication primitive. It attains alignment marker lock based on the repeated AM0 value on each one of the PCS lanes. After alignment markers are found on all PCS lanes, the individual PCS lanes are identified using TBD. The PCS lanes can then be reordered and deskewed. Note that a particular transmit PCS lane can be received on any receive lane of the service interface due to the Skew and multiplexing that occurs in the path.

~~The PCS deskew process deskews and aligns the individual PCS lanes, removes the alignment markers, forms a single stream, and sets the align_status flag to indicate whether the PCS has obtained alignment. The PCS then processes the FEC blocks, transcodes the data back to 64B/66B, descrambles the data and then decodes the 64B/66B encoded data.~~ The PCS deskew process deskews, aligns and reorders the individual PCS lanes, forms a single stream, and sets the align_status flag to indicate whether the PCS has obtained alignment. The PCS then de-interleaves and processes the FEC codewords. Next the PCS removes alignment markers, descrambles the data, transcodes the data back to 64B/66B and then decodes the 64B/66B encoded data.

~~[Editor's note: Multiple FEC codewords might be interleaved on transmit, so the receive processing in this area is TBD. BER monitoring is TBD.]~~

When the receive channel is in test-pattern mode, the BER monitor process may be disabled. The Receive process will be held in the RX_INIT state. The received bits will be compared to the test pattern and errors counted.

Copyright © 2015 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

The PCS shall provide transmit test-pattern mode for the scrambled idle pattern (see 119.2.4.9), and shall provide receive test-pattern mode for the scrambled idle pattern (see 119.2.5.8). Test-pattern mode is activated separately for transmit and receive. The PCS shall support transmit test-pattern mode and receive test-pattern mode operating simultaneously so as to support loopback testing.

119.2.2 Use of blocks

The PCS maps CDMII signals into 66-bit blocks, and vice versa, using a 64B/66B coding scheme. The PCS functions ENCODE and DECODE generate, manipulate, and interpret blocks as defined in 119.2.3.

119.2.3 64B/66B code

The PCS uses 64B66B coding to support transmission of control and data characters. For the 400GBASE-R implementation, this code is further modified by the transcoding and FEC that occurs in this PCS.

119.2.3.1 Notation conventions

For values shown as binary, the leftmost bit is the first transmitted bit.

64B/66B encodes 8 data octets or control characters into a block. Blocks containing control characters also contain a block type field. Data octets are labeled D_0 to D_7. The control characters /I/ and /E/ are labeled C0 to C7. The control characters, /Q/ and /Fsig/, for ordered sets are labeled as O0 since they are only valid on the first octet of the CDMII. The control character for start is labeled as S0 for the same reason. The control character for terminate is labeled as T0 to T7. The four trailing zero data octets in ordered sets are labeled as Z4 to Z7.

One CDMII transfer provides eight characters that are encoded into one 66-bit transmission block. The subscript in the above labels indicates the position of the character in the eight characters from the CDMII transfer.

Contents of block type fields, data octets, and control characters are shown as hexadecimal values. The LSB of the hexadecimal value represents the first transmitted bit. For instance, the block type field 0x1E is sent from left to right as 01111000. The bits of a transmitted or received block are labeled TxB<65:0> and RxB<65:0>, respectively, where TxB<0> and RxB<0> represent the first transmitted bit. The value of the sync header is shown as a binary value. Binary values are shown with the first transmitted bit (the LSB) on the left.

119.2.3.2 64B/66B Block structure

Blocks consist of 66 bits. The first two bits of a block are the synchronization header (sync header). Blocks are either data blocks or control blocks. The sync header is 01 for data blocks and 10 for control blocks. Thus, there is always a transition between the first two bits of a block. The remainder of the block contains the payload.

Data blocks contain eight data characters. Control blocks begin with an 8-bit block type field that indicates the format of the remainder of the block. For control blocks containing a Start, Terminate character, or ordered set, that character is implied by the block type field. Other control characters are encoded in a 7-bit control code. Each control block encodes eight characters.

The format of the blocks is as shown in Figure 119–3. In the figure, the column labeled Input Data shows, in abbreviated form, the eight characters used to create the 66-bit block. These characters are either data characters or control characters and, when transferred across the CDMII, the corresponding TXC or RXC bit is set accordingly. Within the Input Data column, D_0 through D_7 are data octets and are transferred with the corresponding TXC or RXC bit set to zero. All other characters are control characters and are transferred

Copyright © 2015 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

with the corresponding TXC or RXC bit set to one. The single bit fields (thin rectangles with no label in the figure) are sent as zero and ignored upon receipt.

All unused values of block type field are invalid; they shall not be transmitted and shall be considered an error if received.

Input Data	Sync	Block Payload							
Bit Position: / Data Block Format:	0 1	2							65
$D_0\,D_1\,D_2\,D_3\,D_4\,D_5\,D_6\,D_7$	01	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
Control Block Formats:		**Block Type Field**							
$C_0\,C_1\,C_2\,C_3\,C_4\,C_5\,C_6\,C_7$	10	0x1E	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7
$S_0\,D_1\,D_2\,D_3\,D_4\,D_5\,D_6\,D_7$	10	0x78	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$O_0\,D_1\,D_2\,D_3\,Z_4\,Z_5\,Z_6\,Z_7$	10	0x4B	D_1	D_2	D_3	O_0	0x000_0000		
$T_0\,C_1\,C_2\,C_3\,C_4\,C_5\,C_6\,C_7$	10	0x87		C_1	C_2	C_3	C_4	C_5	C_6	C_7
$D_0\,T_1\,C_2\,C_3\,C_4\,C_5\,C_6\,C_7$	10	0x99	D_0		C_2	C_3	C_4	C_5	C_6	C_7
$D_0\,D_1\,T_2\,C_3\,C_4\,C_5\,C_6\,C_7$	10	0xAA	D_0	D_1		C_3	C_4	C_5	C_6	C_7
$D_0\,D_1\,D_2\,T_3\,C_4\,C_5\,C_6\,C_7$	10	0xB4	D_0	D_1	D_2		C_4	C_5	C_6	C_7
$D_0\,D_1\,D_2\,D_3\,T_4\,C_5\,C_6\,C_7$	10	0xCC	D_0	D_1	D_2	D_3		C_5	C_6	C_7
$D_0\,D_1\,D_2\,D_3\,D_4\,T_5\,C_6\,C_7$	10	0xD2	D_0	D_1	D_2	D_3	D_4		C_6	C_7
$D_0\,D_1\,D_2\,D_3\,D_4\,D_5\,T_6\,C_7$	10	0xE1	D_0	D_1	D_2	D_3	D_4	D_5		C_7
$D_0\,D_1\,D_2\,D_3\,D_4\,D_5\,D_6\,T_7$	10	0xFF	D_0	D_1	D_2	D_3	D_4	D_5	D_6	

Figure 119–3—64B/66B block formats

WARNING

Any changes to the coding that affects the transcoding may impact ITU payloads.

119.2.3.3 Control codes

The same set of control characters are supported by the CDMII and the PCS. The representations of the control characters are the control codes. The CDMII encodes a control character into an octet (an eight bit value). The 400GBASE-R PCS encodes the start and terminate control characters implicitly by the block type field. The 400GBASE-R PCS encodes the ordered set control codes using the block type field. The 400GBASE-R PCS encodes each of the other control characters into a 7-bit control code.

The control characters and their mappings to 400GBASE-R control codes and CDMII control codes are specified in Table 82-1. All CDMII and 400GBASE-R control code values that do not appear in the table shall not be transmitted and shall be considered an error if received. The ability to transmit or receive Low Power Idle (LPI) is required for PHYs that support EEE (see Clause 78). If EEE has not been negotiated, LPI shall not be transmitted and shall be treated as an error if received.

119.2.3.4 Valid and invalid blocks

Copyright © 2015 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

The valid and invalid blocks are identical to those in 82.2.3.5 with the exception that it is a CDMII data stream instead of XLGMII/CGMII.

119.2.3.5 Idle (/I/)

Idle control characters are identical to those in 82.2.3.6.

119.2.3.6 Start (/S/)

Start control characters are identical to those in 82.2.3.7.

119.2.3.7 Terminate (/T/)

The terminate control characteristics are identical to those in 82.2.3.8.

119.2.3.8 Ordered set (/O/)

Ordered sets are specified identically as in 82.2.3.9.

119.2.3.9 Error (/E/)

In both the 64B/66B encoder and decoder, the /E/ is generated whenever an /E/ is detected. The /E/ is also generated when invalid blocks are detected. The /E/ allows the PCS to propagate detected errors. See R_TYPE and T_TYPE function definitions in 119.2.6.2.3 for further information.

119.2.4 Transmit

119.2.4.1 Transmit process

The transmit process generates 66-bit blocks based upon the TXD<63:0> and TXC<7:0> signals received from the CDMII. One CDMII data transfer is encoded into one 66-bit block. The transmit process must delete idle control characters or sequence ordered sets to accommodate the transmission of alignment markers. If the PCS transmit process spans multiple clock domains, it may also perform clock rate compensation via the deletion of idle control characters or sequence ordered sets or the insertion of idle control characters.

There are sufficient idle control characters to delete in order to make room for alignment markers, in addition to handling clock compensation. Idle control characters or sequence ordered sets are removed, if necessary, to accommodate the insertion of the 120-bit alignment markers. See 119.2.4.4 for more details.

The transmit process generates blocks as specified in the transmit process state diagram. The contents of each block are contained in a vector tx_coded<65:0>, which is passed to the transcoder. tx_coded<1:0> contains the sync header and the remainder of the bits contain the block payload.

119.2.4.2 64B/66B to 256B/257B transcoder

The transcoder constructs a 257-bit block, tx_xcoded<256:0>, from a group of four 66-bit blocks, tx_coded_*j*<65:0> where *j*=0 to 3. For each group of four 66-bit blocks, *j*=3 corresponds to the most recently received block. Bit 0 in each 66-bit block is the first bit received and corresponds to the first bit of the synchronization header.

If for all *j*=0 to 3, tx_coded_*j*<0>=0 and tx_coded_*j*<1>=1, tx_xcoded<256:0> shall be constructed as follows:

Copyright © 2015 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

a) tx_xcoded<0> = 1
b) tx_xcoded<(64*j*+64):(64*j*+1)> = tx_coded_*j*<65:2> for *j*=0 to 3

If for all *j*=0 to 3, tx_coded_*j*<0> ≠ tx_coded_*j*<1> (valid synchronization header) and for any *j*=0 to 3, tx_coded_*j*<0>=1 and tx_coded_*j*<1>=0, tx_xcoded<256:0> shall be constructed as follows:

a1) tx_xcoded<0> = 0
b1) tx_xcoded<*j*+1> = tx_coded_*j*<1> for *j*=0 to 3
c1) Let *c* be the smallest value of *j* such that tx_coded_*c*<0>=1. In other words, tx_coded_*c* is the first 66-bit control block that was received in the current group of four blocks.
d1) Let tx_payloads<(64*j*+63):64*j*> = tx_coded_*j*<65:2> for *j*=0 to 3
e1) Omit tx_coded_*c*<9:6>, which is the second nibble (based on transmission order) of the block type field for tx_coded_*c,* from tx_xcoded per the following expressions.
tx_xcoded<(64*c*+8):5> = tx_payloads<(64*c*+3):0>
tx_xcoded<256:(64*c*+9)> = tx_payloads<255:(64*c*+8)>

If for any *j*=0 to 3, tx_coded_*j*<0> = tx_coded_*j*<1> (invalid synchronization header), tx_xcoded<256:0> shall be constructed as follows:

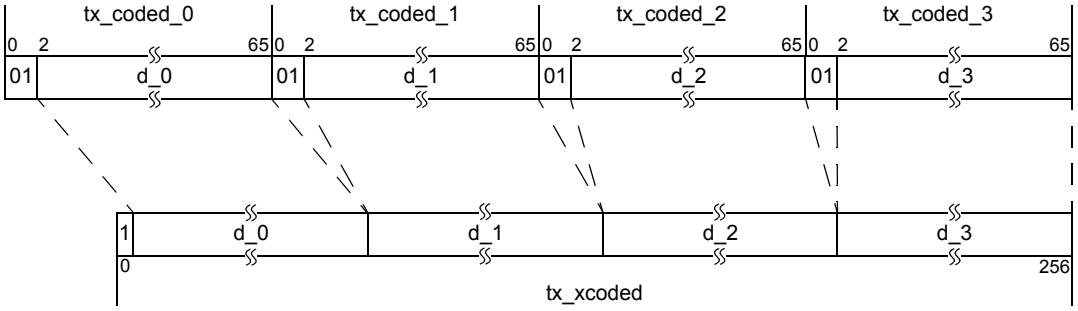
a2) tx_xcoded<0> = 0
b2) tx_xcoded<*j*+1> = 1 for *j*=0 to 3
c2) Let tx_payloads<(64*j*+63):64*j*> = tx_coded_*j*<65:2> for *j*=0 to 3
d2) Omit the second nibble (based on transmission order) of tx_coded_0 per the following expressions.
tx_xcoded<8:5> = tx_payloads<3:0>
tx_xcoded<256:9> = tx_payloads<255:8>

Several examples of the construction of tx_xcoded<256:0> are shown in Figure 119–4. In Figure 119–4, d_*j* indicates the *j*th 66-bit block contains only data octets, c_*j* indicates the *j*th 66-bit block contains one or more control characters, f_*j* denotes the first nibble of the block type field for 66-bit block *j*, and s_*j* denotes the second nibble of the block type field for 66-bit block *j*.

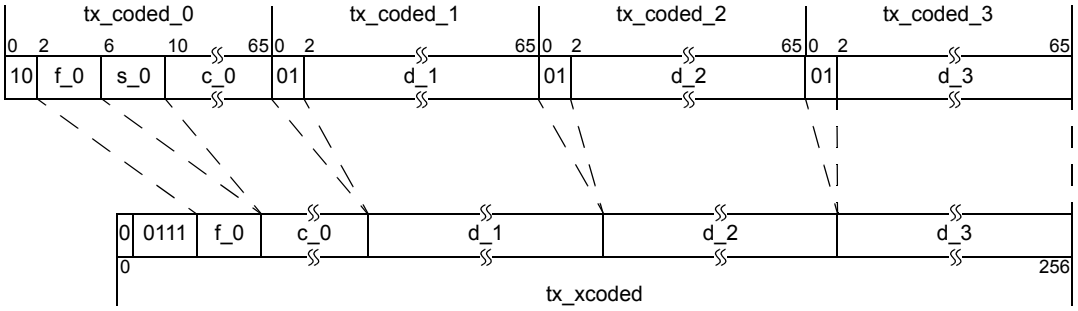
For each 257-bit block, bit 0 shall be the first bit transmitted.

Copyright © 2015 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

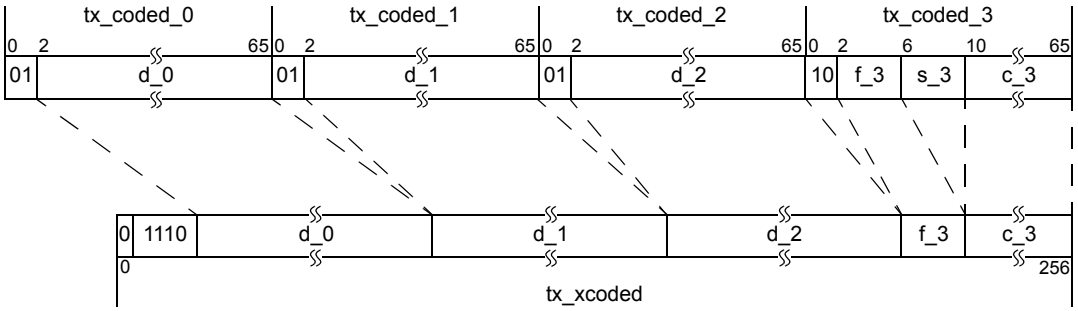
.



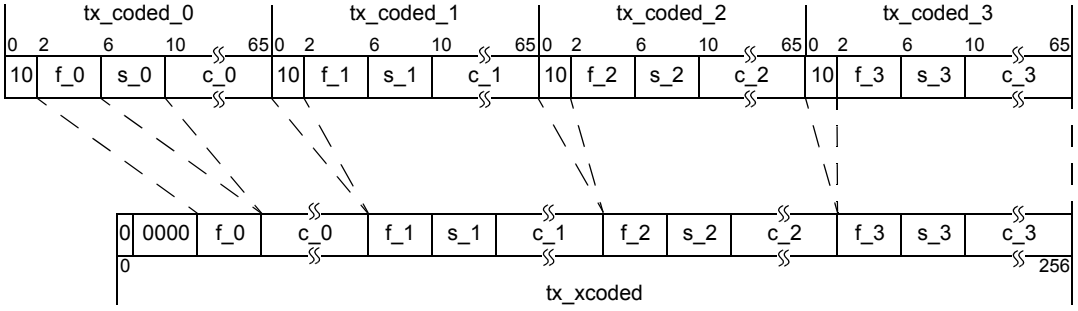
Example 1: All data blocks



Example 2: Control block followed by three data blocks



Example 3: Three data blocks followed by a control block



Example 4: All control blocks

Figure 119–4—Examples of the construction of tx_xcoded

Copyright © 2015 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

119.2.4.3 Scrambler

The payload, tx_xcoded<256:0>, is scrambled with a self-synchronizing scrambler to generate tx_scrambled<256:0>. The scrambler is identical to the scrambler used in Clause 49, see 49.2.6 for the definition of the scrambler.

119.2.4.4 Alignment marker insertion

In order to support deskew and reordering of individual PCS lanes at the receive PCS, alignment markers are added periodically to each PCS lane. Each alignment marker is defined as a unique 120-bit block. The alignment markers are inserted as a group, aligned to the beginning of a FEC block, and interrupt any data transfer that is already in progress. A 136-bit pad is appended to the alignment markers to yield the equivalent of eight 257-bit blocks. The pad bits shall be set to a free running PRBS9 pattern, defined by the polynomial $x^9 + x^5 + 1$. The pad shall not be checked on receive.

Room for the alignment markers is created by periodically deleting idle control characters from the CDMII data stream. Special properties of the alignment markers are that they are not scrambled, do not conform to the encoding rules as outlined in Figure 119–3 and are not transcoded. This is possible because the alignment markers are added after encoding, transcoding, and scrambling, and removed before descrambling, transcoding, and 64B/66B decoding. The alignment markers are not scrambled in order to allow the receiver to find the alignment markers, deskew the PCS lanes, and reassemble the aggregate stream before descrambling is performed. The alignment markers themselves are formed from a known pattern that is defined to be balanced and with many transitions and therefore scrambling is not necessary. The group of alignment markers shall be inserted once every 161920 257-bit blocks, one alignment marker per PCS lane. Alignment marker mapping and repetition rate are shown in Figure 119–6 and Figure 119–7.

The format of the alignment markers is shown in Figure 119–5. There is a portion that is common across all alignment markers, and then a unique portion per PCS lane.

The content of the alignment markers shall be as shown in Table 119–1. The contents depend on the PCS lane number and the octet number, with the first 64 bits being identical across all alignment markers to allow for common synchronization across lanes. What is shown in Table 119–1 is how the alignment markers appear on the PCS lanes. In the FEC codewords, they appear in a permuted format due to the codeword interleaving that occurs before FEC codewords are distributed to PCS lanes.

As an example, the lane marker for 400GBASE-R lane number 0 is sent as (left most bit sent first):

10000011 00010110 10000100 00101111 01111100 01111001 01111011 11010000 TBD

~~*[Editor's note: Show bit pattern example here after AMs are defined.]*~~

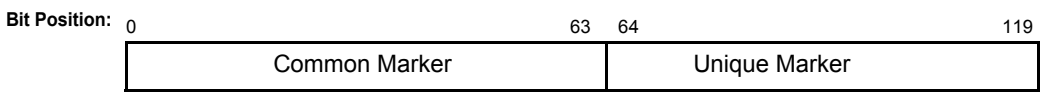


Figure 119–5—Alignment marker format

Copyright © 2015 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

Table 119–1—400GBASE-R Alignment marker encodings

PCS lane number	Encoding[a] $\{M_0, M_1, M_2, BIP_3, M_4, M_5, M_6, BIP_7\}$
0	0xC1, 0x68, 0x21, 0xF4, 0x3E, 0x97, 0xDE, 0x0B, next 56b are TBD
1	0xC1, 0x68, 0x21, 0xF4, 0x3E, 0x97, 0xDE, 0x0B, next 56b are TBD
2	0xC1, 0x68, 0x21, 0xF4, 0x3E, 0x97, 0xDE, 0x0B, next 56b are TBD
3	0xC1, 0x68, 0x21, 0xF4, 0x3E, 0x97, 0xDE, 0x0B, next 56b are TBD
4	0xC1, 0x68, 0x21, 0xF4, 0x3E, 0x97, 0xDE, 0x0B, next 56b are TBD
5	0xC1, 0x68, 0x21, 0xF4, 0x3E, 0x97, 0xDE, 0x0B, next 56b are TBD
6	0xC1, 0x68, 0x21, 0xF4, 0x3E, 0x97, 0xDE, 0x0B, next 56b are TBD
7	0xC1, 0x68, 0x21, 0xF4, 0x3E, 0x97, 0xDE, 0x0B, next 56b are TBD
8	0xC1, 0x68, 0x21, 0xF4, 0x3E, 0x97, 0xDE, 0x0B, next 56b are TBD
9	0xC1, 0x68, 0x21, 0xF4, 0x3E, 0x97, 0xDE, 0x0B, next 56b are TBD
10	0xC1, 0x68, 0x21, 0xF4, 0x3E, 0x97, 0xDE, 0x0B, next 56b are TBD
11	0xC1, 0x68, 0x21, 0xF4, 0x3E, 0x97, 0xDE, 0x0B, next 56b are TBD
12	0xC1, 0x68, 0x21, 0xF4, 0x3E, 0x97, 0xDE, 0x0B, next 56b are TBD
13	0xC1, 0x68, 0x21, 0xF4, 0x3E, 0x97, 0xDE, 0x0B, next 56b are TBD
14	0xC1, 0x68, 0x21, 0xF4, 0x3E, 0x97, 0xDE, 0x0B, next 56b are TBD
15	0xC1, 0x68, 0x21, 0xF4, 0x3E, 0x97, 0xDE, 0x0B, next 56b are TBD

[a]Each octet is transmitted LSB to MSB.

Copyright © 2015 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

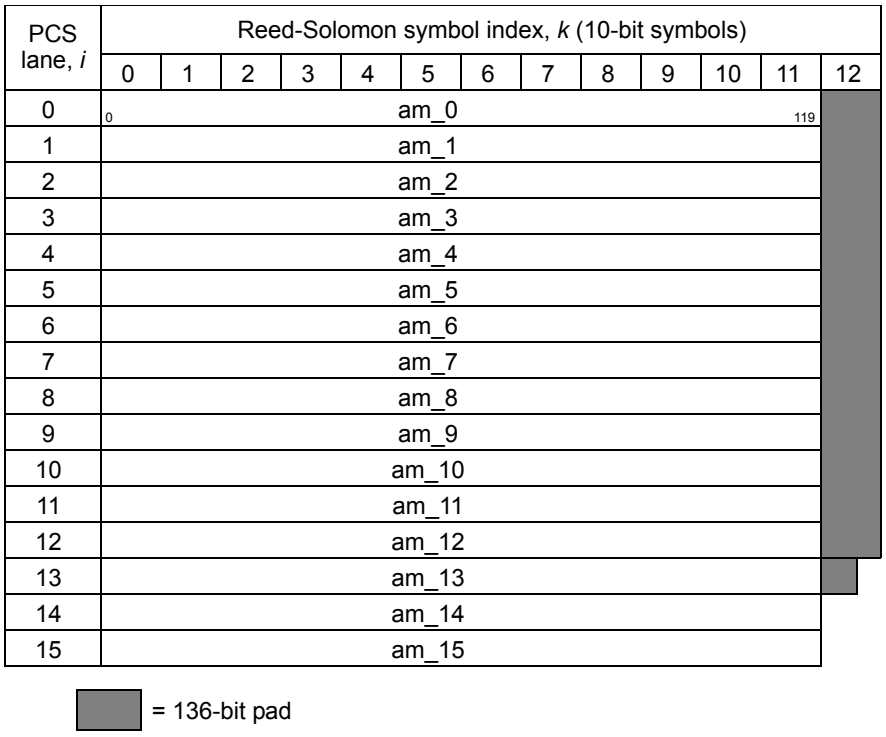


Figure 119–6—Alignment marker Mapping to PCS lanes

Copyright © 2015 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

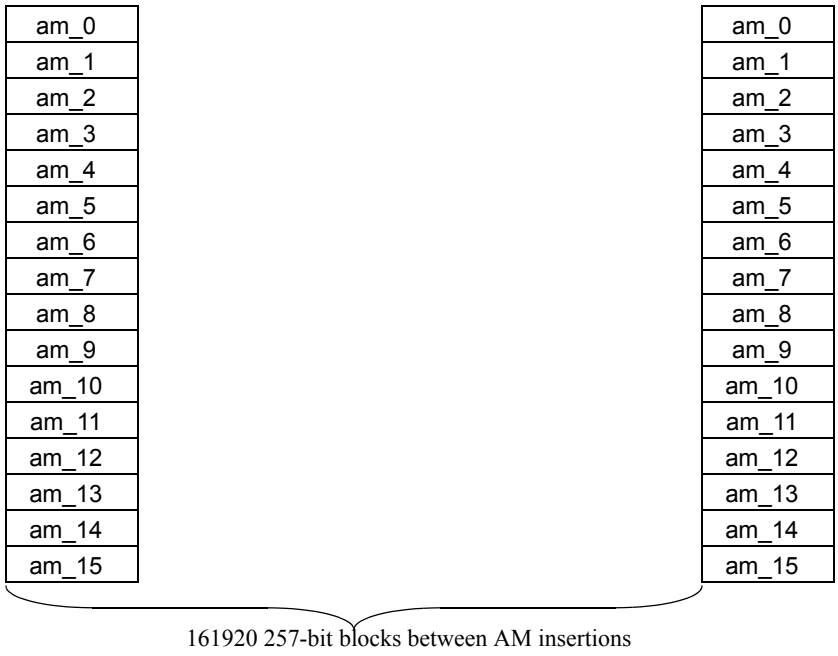


Figure 119–7—Alignment marker insertion period

119.2.4.5 Pre-FEC Distribution

Two Reed-Solomon FEC codewords are interleaved before data is distributed to the PCS lanes to improve error correction capability. Data is distributed to those two FEC codewords by performing a 10 b round robing distribution of the tx_scrambled<256:0> data as follows.

For all j=0 to 39, tx_temp<10279:0> shall be constructed as follows:

tx_temp<(256+257j):(0+257j)> = tx_scrambled_j<256:0>

For all i=0 to 513, tx_prefec0<5139:0> and tx_prefec1<5139:0> shall be constructed as follows:

tx_prefec0<(9+10i):(0+10i)> = tx_temp<(9+20i):(0+20i)>

tx_prefec1<(9+10i):(0+10i)> = tx_temp<(19+20i):(10+20i)>

119.2.4.6 Reed-Solomon encoder

The PCS sublayer employs a Reed-Solomon code operating over the Galois Field GF(2^{10}) where the symbol size is 10 bits. The encoder processes k message symbols to generate $2t$ parity symbols, which are then appended to the message to produce a codeword of $n=k+2t$ symbols. For the purposes of this clause, a particular Reed-Solomon code is denoted RS(n,k).

The PCS sublayer shall implement RS(544,514). ~~Each *k* symbol message corresponds to 20~~The PCS interleaves two FEC codewords, therefore each k-symbol message corresponds to one half of a group of 40 interleaved 257-bit blocks produced by the transcoder (with the exception of the alignment marker blocks). Each code is based on the generating polynomial given by Equation (119–1).

Copyright © 2015 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

$$g(x) = \prod_{j=0}^{2t-1}(x-\alpha^j) = g_{2t}x^{2t} + g_{2t-1}x^{2t-1} + \ldots + g_1 x + g_0 \quad (119\text{–}1)$$

In Equation (119–1), α is a primitive element of the finite field defined by the polynomial $x^{10}+x^3+1$.

Equation (119–2) defines the message polynomial $m(x)$ whose coefficients are the message symbols m_{k-1} to m_0.

$$m(x) = m_{k-1}x^{n-1} + m_{k-2}x^{n-2} + \ldots + m_1 x^{2t+1} + m_0 x^{2t} \quad (119\text{–}2)$$

Each message symbol m_i is the bit vector $(m_{i,9}, m_{i,8}, \ldots, m_{i,1}, m_{i,0})$, which is identified with the element $m_{i,9}\alpha^9 + m_{i,8}\alpha^8 + \ldots + m_{i,1}\alpha + m_{i,0}$ of the finite field. The message symbols are composed of the bits ~~of the transcoded blocks tx_scrambled (including a group of alignment markers when appropriate) such that bit 0 of the first transcoded block in the message is bit 0 of m_{k-1} and bit 256 of the last transcoded block in the message~~of the variable tx_prefec0 for codeword0 and tx_prefec1 for codeword1 (including a group of alignment markers when appropriate) such that bit 0 of tx_prefec0 or tx_prefec1 as appropriate, is bit 0 of m_{k-1} and bit 5139 of tx_prefec0 or tx_prefec1 as appropriate, is bit 9 of m_0. The first symbol input to the encoder is m_{k-1}.

Equation (119–3) defines the parity polynomial $p(x)$ whose coefficients are the parity symbols p_{2t-1} to p_0.

$$p(x) = p_{2t-1}x^{2t-1} + p_{2t-2}x^{2t-2} + \ldots + p_1 x + p_0 \quad (119\text{–}3)$$

The parity polynomial is the remainder from the division of $m(x)$ by $g(x)$. This may be computed using the shift register implementation illustrated in Figure 119–8. The outputs of the delay elements are initialized to zero prior to the computation of the parity for a given message. After the last message symbol, m_0, is processed by the encoder, the outputs of the delay elements are the parity symbols for that message.

The codeword polynomial $c(x)$ is then the sum of $m(x)$ and $p(x)$ where the coefficient of the highest power of x, $c_{n-1} = m_{k-1}$ is ~~transmitted~~ first and the coefficient of the lowest power of x, $c_0 = p_0$ is ~~transmitted~~ last. ~~The first bit transmitted from each symbol is bit 0~~

For all j=0 to 5439, variable tx_postfec0<5439:0> and tx_postfec1<5439:0> shall be constructed as follows:

tx_postfec0<j> = c<5139-j>

tx_postfec1<j> = c<5139-j>

Copyright © 2015 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

.

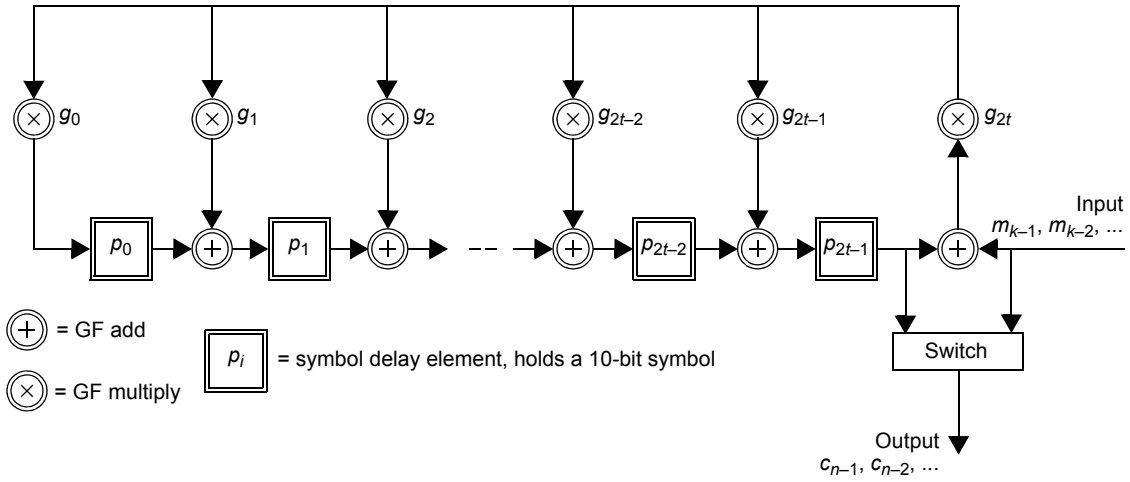


Figure 119–8—Reed-Solomon encoder functional model

The coefficients of the generator polynomial for each code are presented in Table 119–2. Example codewords for each code are provided in Annex 91A.

Table 119–2—Coefficients of the generator polynomial g_i (decimal)

i	RS(544,514)	*i*	RS(544,514)	*i*	RS(544,514)
0	523	11	883	22	565
1	834	12	503	23	108
2	128	13	942	24	1
3	158	14	385	25	552
4	185	15	495	26	230
5	127	16	720	27	187
6	392	17	94	28	552
7	193	18	132	29	575
8	610	19	593	30	1
9	788	20	249		
10	361	21	282		

119.2.4.7 Symbol distribution

~~*[Editor's note: How data is distributed to PCS lanes is TBD. It is possible that multiple FEC frames will*~~

Copyright © 2015 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

~~***be interleaved before distribution. The distribution process is shown in Figure 119–9.]***~~

Once the data has been FEC encoded, two FEC codewords are interleaved before the data is distributed to each PCS lane. The interleaving of two codewords shall follow this procedure:

For all j=0 to 271

For all i=0 to 9

tx_out<20j+i> = tx_postfec0<10j+i>
tx_out<20j+i+10> = tx_postfec1<10j+i>

Once the data has been Reed-Solomon encoded and interleaved, it shall be distributed to 16 PCS lanes, one 10b symbol at a time, from the lowest to the highest PCS lane. The first bit transmitted from each 10b symbol is bit 0.

TBD

Figure 119–9—PCS Block distribution

Copyright © 2015 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

119.2.4.8 Transmit bit ordering

[Editor's note: The transmit bit ordering is TBD and will be illustrated in Figure 119–10 once defined.]

TBD

Figure 119–10—Transmit bit ordering

119.2.4.9 Test-pattern generators

The PCS shall have the ability to generate and detect a scrambled idle test pattern. This test-pattern mode is suitable for receiver tests and for certain transmitter tests.

When a scrambled idle pattern is enabled, the test pattern is generated by the PCS. The test pattern is an idle control block (block type=0x1E) with all idles as defined in Figure 119–3. This is sent continuously and is transcoded, scrambled and encapsulated by the FEC.

When the transmit channel is operating in test-pattern mode, the encoded bit stream is distributed to the PCS Lanes as in normal operation (see 119.2.4.7).

If a Clause 45 MDIO is implemented, then control of the test-pattern generation is from the BASE-R PCS test-pattern control register (bit 3.42.3).

119.2.5 Receive function

119.2.5.1 Alignment lock and deskew

The RS-FEC receive function forms 16 bit streams by concatenating the bits from each of the 16 PMA:IS_UNITDATA_*i*.indication primitives in the order they are received. It obtains lock to the alignment markers as specified by the alignment marker lock state diagram shown in Figure 119–11.

After alignment marker lock is achieved on all 16 lanes, all inter-lane Skew is removed as specified by the PCS synchronization state diagram shown in Figure 119–12. The FEC receive function shall support a maximum Skew of 180 ns between FEC lanes and a maximum Skew Variation of 4 ns.

Copyright © 2015 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

119.2.5.2 Lane reorder and de-interleave

PCS lanes can be received on different lanes of the service interface from which they were originally transmitted. The PCS receive function shall order the PCS lanes according to the PCS lane number. The PCS lane number is defined by the alignment marker that is mapped to each PCS lane (see 119.2.4.4).

After all PCS lanes are aligned, deskewed, and reordered, the ~~FEC lanes are multiplexed together in the proper order to reconstruct the original stream of FEC codewords.~~ two FEC codewords can be de-interleaved in the proper order to reconstruct the original stream of two FEC codewords.

~~***[Editors note: how data is distributed to PCS lanes is from FEC frames is TBD, and there is the possibility that multiple FEC frames will be interleaved. So how data is reformed before FEC decoding is TBD.]***~~

119.2.5.3 Reed-Solomon decoder

The Reed-Solomon decoder extracts the message symbols from the codeword, corrects them as necessary, and discards the parity symbols. ~~The message symbols~~ Data is reverse distributed into rx_scrambled.from two associated codewords corresponding to ~~2~~40 transcoded blocks ~~rx_scrambled~~.

The RS-FEC sublayer shall be capable of correcting any combination of up to t=15 symbol errors in a codeword. The RS-FEC sublayer shall also be capable of indicating when an errored codeword was not corrected. The probability that the decoder fails to indicate a codeword with t+1 errors as uncorrected is not expected to exceed 10^{-6}. This limit is also expected to apply for t+2 errors, t+3 errors, and so on.

The Reed-Solomon decoder may provide the option to perform error detection without error correction to reduce the delay contributed by the RS-FEC sublayer. The presence of this option is indicated by the assertion of the FEC_bypass_correction_ability variable (see 119.3). When the option is provided, it is enabled by the assertion of the FEC_bypass_correction_enable variable (see 119.3).

NOTE—The PHY may rely on the error correction capability of the RS-FEC to achieve its performance objectives. It is recommended that acceptable performance of the underlying link is verified before error correction is bypassed.

The Reed-Solomon decoder indicates errors to the 64B/66B decoder by intentionally corrupting 66-bit block synchronization headers. When the Reed-Solomon decoder determines that a codeword contains errors (when the bypass correction feature is enabled) or contains errors that were not corrected (when the bypass correction feature is not supported or not enabled), it shall ensure that, for every 257-bit block within the two associated ~~-~~codewords, the synchronization header for all 66-bit blocks at the output of the 256B/257B to 64B/66B transcoder, rx_coded_0<1:0>, is set to 11. This causes the PCS to ~~discard~~mark all frames that are fully or partially within the two associated codewords.

119.2.5.4 Alignment marker removal

The first 2056 message bits in every 8096th codeword is the vector am_rx<2055:0> where bit 0 is the first bit received. The specific codewords that include this vector are indicated by the alignment lock and deskew function.

The vector am_rx shall be removed prior to transcoding.

119.2.5.5 Descrambler

The payload, rx_scrambled<256:0>, is descrambled with a self-synchronizing scrambler to generate rx_xcoded<256:0>.

The descrambler is identical to that used in Clause 49, see 49.2.10 for the definition.

Copyright © 2015 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

119.2.5.6 256B/257B to 64B/66B transcoder

The transcoder extracts a group of four 66-bit blocks, rx_coded_*j*<65:0> where *j*=0 to 3, from each 257-bit block rx_xcoded<256:0>. Bit 0 of the 257-bit block is the first bit received.

If rx_xcoded<0> is 1, rx_coded_*j*<65:0> for *j*=0 to 3 shall be derived as follows.

a1) rx_coded_*j*<65:2> = rx_xcoded<(64*j*+64):(64*j*+1)> for *j*=0 to 3
b1) rx_coded_*j*<0>=0 and rx_coded_*j*<1>=1 for all *j*=0 to 3

If rx_xcoded<0> is 0 and any rx_xcoded<*j*+1>=0 for *j*=0 to 3, rx_coded_*j*<65:0> for *j*=0 to 3 shall be derived as follows.

a2) Let *c* be the smallest value of *j* such that rx_xcoded<*j*+1>=0. In other words, rx_coded_*c* is the first 66-bit control block in the resulting group of four blocks.
b2) Let rx_payloads be a vector representing the payloads of the four 66-bit blocks. It is derived using the following expressions:
rx_payloads<(64*c*+3):0> = rx_xcoded<(64*c*+8):5>
rx_payloads<(64*c*+7):(64*c*+4)> = 0000 (an arbitrary value that is later replaced by s_*c*)
rx_payloads<255:(64*c*+8)> = rx_xcoded<256:(64*c*+9)>
c2) rx_coded_*j*<65:2> = rx_payloads<(64*j*+63):64*j*> for *j*=0 to 3
d2) Let f_*c*<3:0> = rx_coded_*c*<5:2> be the scrambled first nibble (based on transmission order) of the block type field for rx_coded_*c*.
e2) The block type field may be uniquely identified by either its most or least significant nibble. Since g<3:0> is the least significant nibble of the block type field (per the transmission order), derive h<3:0> by cross-referencing to g<3:0> using Figure 119–3. For example, if g<3:0> is 0xE then h<3:0> is 0x1. If no match to g<3:0> is found, h<3:0> is set to 0000.
f2) If rx_xcoded<*j*+1>=0, rx_coded_*j*<0>=1 and rx_coded_*j*<1>=0 for *j*=0 to 3
g2) If rx_xcoded<*j*+1>=1, rx_coded_*j*<0>=0 and rx_coded_*j*<1>=1 for *j*=0 to 3
h2) If h<3:0> = 0000, rx_coded_*c*<1>=1 (invalidate synchronization header)

If rx_xcoded<0> is 0 and all rx_xcoded<*j*+1>=1 for *j*=0 to 3, rx_coded_*j*<65:0> for *j*=0 to 3 shall be derived as follows.

a3) Set *c* = 0 and h<3:0> = 0000.
b3) Let rx_payloads be a vector representing the payloads of the four 66-bit blocks. It is derived using the following expressions.
rx_payloads<(64*c*+3):0> = rx_xcoded<(64*c*+8):5>
rx_payloads<(64*c*+7):(64*c*+4)> = 0000 (an arbitrary value that is later replaced by s_*c*)
rx_payloads<255:(64*c*+8)> = rx_xcoded<256:(64*c*+9)>
c3) rx_coded_*j*<65:2> = rx_payloads<(64*j*+63):(64*j*)> for *j*=0 to 3
d3) rx_coded_*j*<0>=0 and rx_coded_*j*<1>=0 for *j*=0 and 2
e3) rx_coded_*j*<0>=1 and rx_coded_*j*<1>=1 for *j*=1 and 3

The 66-bit blocks are transmitted in order from *j*=0 to 3. Bit 0 of each block is the first bit transmitted.

119.2.5.7 Receive process

The receive process decodes blocks to produce RXD<63:0> and RXC<7:0> for transmission to the CDMII. One CDMII data transfer is decoded from each block. The receive process must insert idle control characters to compensate for the removal of alignment markers. If the PCS receive process spans multiple clock domains, it may also perform clock rate compensation via the deletion of idle control characters or sequence ordered sets or the insertion of idle control characters.

Copyright © 2015 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

The receive process decodes blocks as specified in the receive state diagram shown in Figure 119–16.

119.2.5.8 Test-pattern checker

When the receive channel is operating in scrambled idle test-pattern mode, the scrambled idle test-pattern checker checks the bits received via PMA:IS_UNITDATA_*i*.indication primitives.

The scrambled idle test-pattern checker utilizes the alignment marker lock state diagram, the PCS deskew state diagram, and the descrambler operating as they do during normal data reception. The BER monitor state diagram is disabled during receive test-pattern mode. When align_status is true and the scrambled idle receive test-pattern mode is active, the scrambled idle test-pattern checker observes the sync header and the output from the descrambler. When the sync header and the output of the descrambler is the all idle pattern, a match is detected. When operating in scrambled idle test pattern, the test-pattern error counter counts blocks with a mismatch. Any mismatch indicates an error and shall increment the test-pattern error counter.

If a Clause 45 MDIO is implemented, then control of the test-pattern reception is from the BASE-R PCS test-pattern control register (bit 3.42.2). In addition errors are counted in the BASE-R PCS test-pattern error counter register (3.43.15:0).

119.2.6 Detailed functions and state diagrams

119.2.6.1 State diagram conventions

The body of this subclause is composed of state diagrams, including the associated definitions of variables, functions, and counters. Should there be a discrepancy between a state diagram and descriptive text, the state diagram prevails.

The notation used in the state diagrams follows the conventions of 21.5. The notation ++ after a counter or integer variable indicates that its value is to be incremented.

119.2.6.2 State variables

119.2.6.2.1 Constants

EBLOCK_R<71:0>
: 72 bit vector to be sent to the CDMII containing /E/ in all the eight character locations.

EBLOCK_T<65:0>
: 66 bit vector to be sent to the PMA containing /E/ in all the eight character locations.

LBLOCK_R<71:0>
: 72 bit vector to be sent to the CDMII containing one Local Fault ordered set. The Local Fault ordered set is defined in 119.3.

LBLOCK_T<65:0>
: 66 bit vector to be sent to the PMA containing one Local Fault ordered set.

119.2.6.2.2 Variables

all_locked
: A Boolean variable that is set to true when amps_lock<*x*> is true for all *x* and is set to false when amps_lock<*x*> is false for any *x*.

amp_counter_done
: Boolean variable that indicates that amp_counter has reached its terminal count.

amp_match
: Boolean variable that holds the output of the function AMP_COMPARE.

Copyright © 2015 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

amp_valid

Boolean variable that is set to true if the received 120-bit block is a valid alignment marker payload. The alignment marker payload, mapped to an PCS lane according to the process described in 119.2.4.4, consists of 120 known bits. The bits are compared on a TBD basis. If TBD, the candidate block is considered a valid alignment marker payload.

amps_lock<*x*>

Boolean variable that is set to true when the receiver has detected the location of the alignment marker payload sequence for a given lane on the PMA service interface, where *x* = 0:15.

current_pcsl

A variable that holds the PCS lane number corresponding to the current alignment marker payload that is recognized on a given lane of the PMA service interface. It is compared to the variable first_pcsl to confirm that the location of the alignment marker payload sequence has been detected.

cw_bad

A Boolean variable that is set to true if the Reed-Solomon decoder (see 119.2.5.3) fails to correct the current FEC codeword and is set to false otherwise.

deskew_done

A Boolean variable that is set to true when pcs_enable_deskew is set to true and the deskew process is completed. Otherwise, this variable is set to false.

align_status

A variable set by the PCS alignment process to reflect the status of PCS lane-to-lane alignment. Set to true when all lanes are synchronized and aligned and set to false when the deskew process is not complete.

pcs_alignment_valid

Boolean variable that is set to true if all PCS lanes are aligned. PCS lanes are considered to be aligned when amps_lock<*x*> is true for all *x*, each PCS lane is locked to a unique alignment marker payload sequence (see 119.2.4.4), and the PCS lanes are deskewed. Otherwise, this variable is set to false.

pcs_enable_deskew

A Boolean variable that enables and disables the deskew process. Received bits may be discarded whenever deskew is enabled. It is set to true when deskew is enabled and set to false when deskew is disabled.

pcs_lane

A variable that holds the PCS lane number (0 to 15) received on lane *x* of the PMA service interface when amps_lock<*x*>=true. The PCS lane number is determined by the alignment marker payloads based on the mapping defined in 119.2.4.4. The 56 bits that are in the positions of the unique bits in the received alignment marker payload are compared to the expected values for a given payload position and FEC lane on a TBD basis. If no more than TBD nibbles in the candidate block fail to match the corresponding known nibbles for any payload position on a given PCS lane, then the PCS lane number is assigned accordingly.

first_pcsl

A variable that holds the PCS lane number that corresponds to the first alignment marker payload that is recognized on a given lane of the PMA service interface. It is compared to the PCS lane number corresponding to the second alignment marker payload that is tested.

hi_ber

TBD

r_test_mode

Boolean variable that is asserted true when the receiver is in test-pattern mode.

reset

Boolean variable that controls the resetting of the PCS sublayer. It is true whenever a reset is necessary including when reset is initiated from the MDIO, during power on, and when the MDIO has put the PCS sublayer into low-power mode.

restart_lock

Boolean variable that is set by the PCS alignment process to reset the synchronization process on

Copyright © 2015 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

all PCS lanes. It is set to true after 3 consecutive uncorrected codewords are received (3_BAD state) and set to false upon entry into the LOSS_OF_ALIGNMENT state.

rx_coded<65:0>

Vector containing the input to the 64B/66B decoder. The format for this vector is shown in Figure 119–3. The leftmost bit in the figure is rx_coded<0> and the rightmost bit is rx_coded<65>.

rx_raw<71:0>

Vector containing one XLGMII/CGMII transfer. RXC<0> through RXC<7> are from rx_raw<0> through rx_raw<7>, respectively. RXD<0> through RXD<63> are from rx_raw<8> through rx_raw<71>, respectively.

signal_ok

Boolean variable that is set based on the most recently received value of PMA:IS_SIGNAL.indication(SIGNAL_OK). It is true if the value was OK and false if the value was FAIL.

slip_done

Boolean variable that is set to true when the SLIP requested by the synchronization state diagram has been completed indicating that the next candidate 120-bit block position can be tested.

test_amp

Boolean variable this is set to true when a candidate block position is available for testing and false when the FIND_1ST state is entered.

test_cw

Boolean variable that is set to true when a new FEC codeword is available for decoding and is set to false when the TEST_CW state is entered.

tx_coded<65:0>

Vector containing the output from the 64B/66B encoder. The format for this vector is shown in Figure 119–3. The leftmost bit in the figure is tx_coded<0> and the rightmost bit is tx_coded<65>.

tx_raw<71:0>

Vector containing one XLGMII/CGMII transfer. TXC<0> through TXC<7> are placed in tx_raw<0> through tx_raw<7>, respectively. TXD<0> through TXD<63> are placed in tx_raw<8> through tx_raw<71>, respectively.

119.2.6.2.3 Functions

AMP_COMPARE

This function compares the values of first_pcsl and current_pcsl to determine if a valid alignment marker payload sequence has been detected and returns the result of the comparison using the variable amp_match. If current_pcsl and first_pcsl are 0, amp_match is set to true.

DECODE(rx_coded<65:0>)

Decodes the 66-bit vector returning rx_raw<71:0>, which is sent to the CDMII. The DECODE function shall decode the block as specified in 119.2.3.

ENCODE(tx_raw<71:0>)

Encodes the 72-bit vector returning tx_coded<65:0> of which tx_coded<65:2> is sent to the scrambler. The two bits of the sync header bypass the scrambler. The ENCODE function shall encode the block as specified in 119.2.3.

R_TYPE(rx_coded<65:0>)

This function classifies the current rx_coded<65:0> vector as belonging to one of the following types, depending on its contents. The classification results are returned via the r_block_type variable.

Values: C; The vector contains a sync header of 10 and one of the following:

a) A block type field of 0x1E and eight valid control characters other than /E/ or /LI/;

b) A block type field of 0x4B.

LI; For EEE capability, the LI type is supported where the vector contains a sync header of 10, a block type field of 0x1E and eight control characters of 0x06 (/LI/).

S; The vector contains a sync header of 10 and the following:

a) A block type field of 0x78.

Copyright © 2015 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

T; The vector contains a sync header of 10, a block type field of 0x87, 0x99, 0xAA, 0xB4, 0xCC, 0xD2, 0xE1 or 0xFF and all control characters are valid.
D; The vector contains a sync header of 01.
E; The vector does not meet the criteria for any other value.

Valid control characters are specified in Table 119–1.

NOTE—A PCS that does not support EEE classifies vectors containing one or more /LI/ control characters as type E.

R_TYPE_NEXT

This function classifies the 66-bit rx_coded vector that immediately follows the current rx_coded<65:0> vector as belonging to one of the five types defined in R_TYPE, depending on its contents. It is intended to perform a prescient end of packet check. The classification results are returned via the r_block_type_next variable.

SLIP

Causes the next candidate block position to be tested. The precise method for determining the next candidate block position is not specified and is implementation dependent. However, an implementation shall ensure that all possible block positions are evaluated.

T_TYPE = (tx_raw<71:0>)

This function classifies each 72-bit tx_raw vector as belonging to one of the following types depending on its contents. The classification results are returned via the t_block_type variable.

Values: C; The vector contains one of the following:
a) Eight valid control characters other than /O/, /S/, /T/, /LI/, and /E/;
b) One valid ordered set.
LI; For EEE capability, this vector contains eight /LI/ characters.
S; The vector contains an /S/ in its first character, and all characters following the /S/ are data characters.
T; The vector contains a /T/ in one of its characters, all characters before the /T/ are data characters, and all characters following the /T/ are valid control characters other than /O/, /S/ and /T/.
D; The vector contains eight data characters.
E; The vector does not meet the criteria for any other value.

A tx_raw character is a control character if its associated TXC bit is asserted. A valid control character is one containing an XLGMII/CGMII control code specified in Table 119–1. A valid ordered set consists of a valid /O/ character in the first character and data characters in the seven characters following the /O/. A valid /O/ is any character with a value for O code in Table 119–1.

NOTE—A PCS that does not support EEE classifies vectors containing one or more /LI/ control characters as type E.

119.2.6.2.4 Counters

amp_counter

This counter counts the 8192 FEC codewords that separate the ends of two consecutive normal alignment marker payload sequences. An FEC codeword is 340 bits per PCS lane.

cw_bad_count

Counts the number of consecutive uncorrected FEC codewords. This counter is set to zero when an FEC codeword is received and cw_bad is false for that codeword.

119.2.6.3 State diagrams

The 400GBASE-R PCS shall implement sixteen alignment marker lock processes as depicted in Figure 119–11. An alignment marker lock process operates independently on each lane. The alignment

Copyright © 2015 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

marker lock state diagram shown in Figure 119–11 determines when the PCS has obtained alignment marker lock to the received bit stream for a given lane of the service interface. Each alignment marker lock process looks for two valid alignment markers 8192 FEC codewords apart to gain alignment marker lock. Once in lock, a lane will go out of alignment marker lock when three FEC blocks in a row are not correctable. When the alignment marker lock process achieves lock for a lane, and if a Clause 45 MDIO is implemented, the PCS shall record the number of the PCS lane received on that lane of the service interface in the appropriate lane mapping register (3.400 to 3.415).

The PCS shall run the synchronization process as depicted in Figure 119–13. The PCS synchronization process is responsible for determining if the PCS is capable of presenting coherent data to the CDMII. The synchronization process ensures that all PCS lanes have alignment marker lock, are locked to different alignment markers, and that the Skew is within the boundaries of what the PCS can deskew. Synchronization lock, along with alignment marker lock, are restarted if three FEC codewords in a row are not correctable.

[Editor's note: The BER Monitor state diagram is TBD.]

The Transmit state diagram shown in Figure 119–15 controls the encoding of transmitted blocks. It makes exactly one transition for each transmit block processed. Though the Transmit state diagram sends Local Fault ordered sets when reset is asserted, the scrambler may not be operational during reset. Thus, the Local Fault ordered sets may not appear on the PMA service interface.

The Receive state diagram shown in Figure 119–16 controls the decoding of received blocks. It makes exactly one transition for each receive block processed.

The PCS shall perform the functions of alignment marker lock, PCS synchronization, BER Monitor, Transmit, and Receive as specified in the respective state diagrams.

Copyright © 2015 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

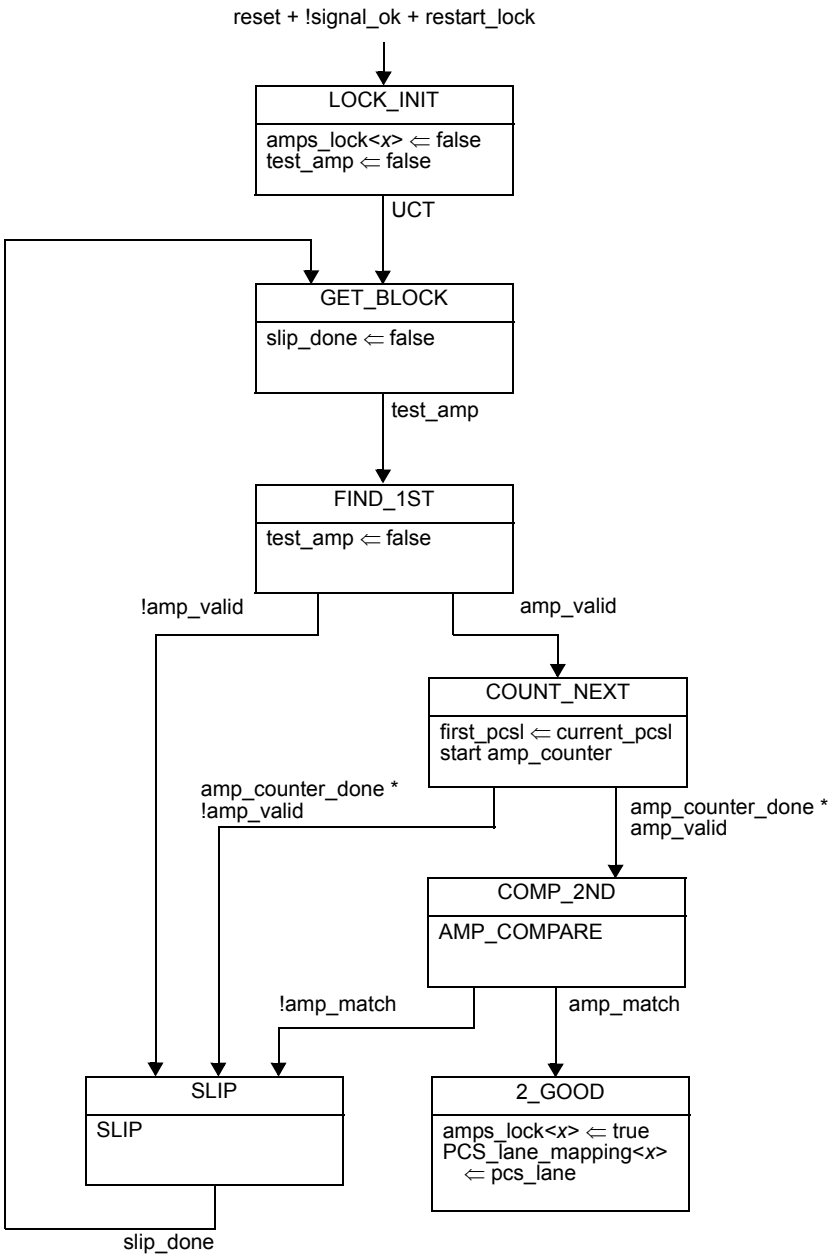


Figure 119–11—Alignment marker lock state diagram

Copyright © 2015 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

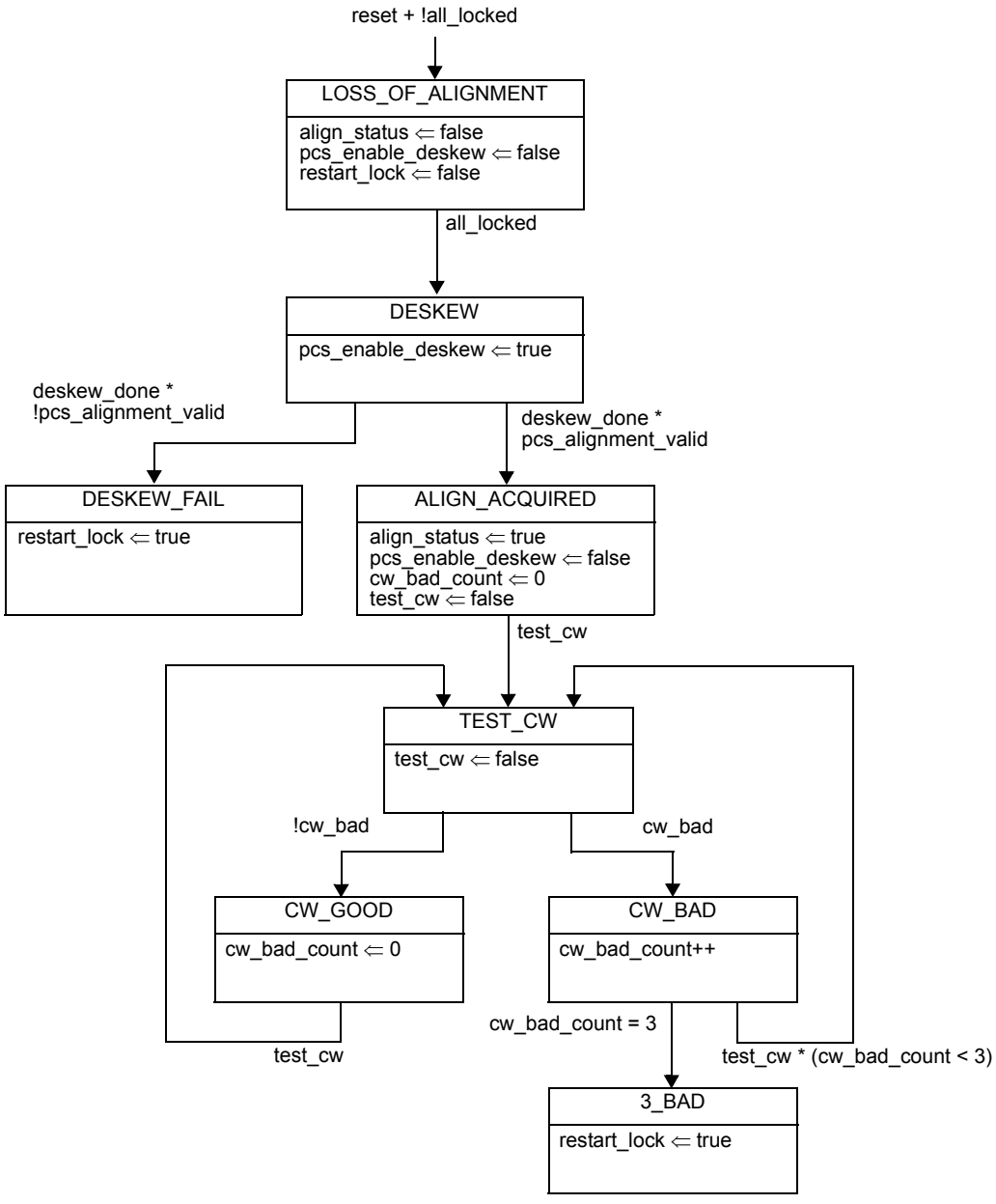


Figure 119–13—PCS synchronization state diagram

Copyright © 2015 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

TBD

Figure 119–14—BER monitor state diagram

Copyright © 2015 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

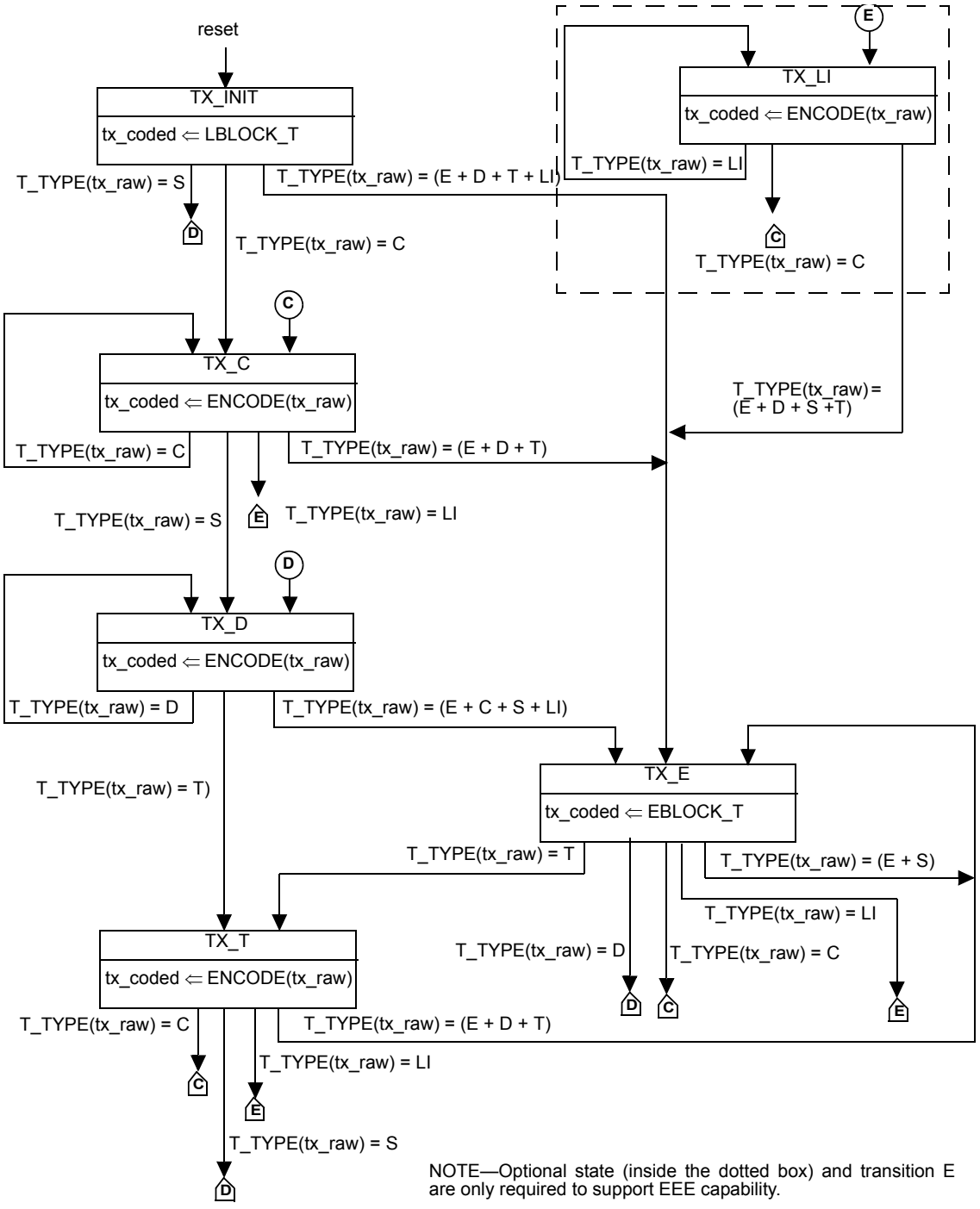


Figure 119–15—Transmit state diagram

Copyright © 2015 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

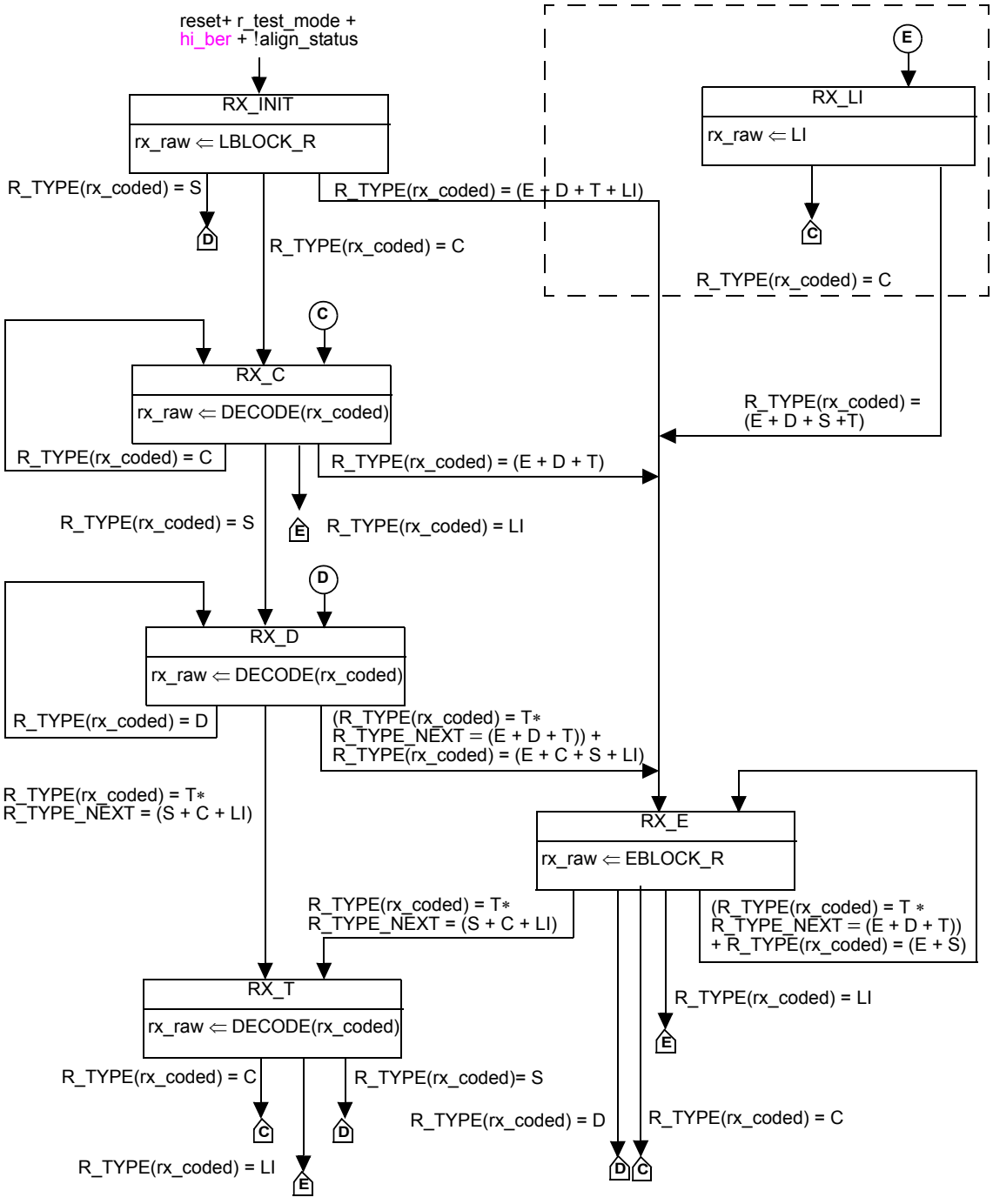


Figure 119–16—Receive state diagram

Copyright © 2015 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

119.3 PCS MDIO function mapping

The optional MDIO capability described in Clause 45 defines several registers that provide control and status information for and about the PCS. If MDIO is implemented, it shall map MDIO control bits to PCS control variables as shown in Table 119–3, and MDIO status bits to PCS status variables as shown in Table 119–4.

Table 119–3—MDIO/PCS control variable mapping

MDIO control variable	PCS register name	Register/ bit number	PCS control variable
Reset	PCS control 1 register	3.0.15	reset
Loopback	PCS control 1 register	3.0.14	Loopback
Transmit test-pattern enable	BASE-R PCS test-pattern control register	3.42.3	tx_test_mode
Receive test-pattern enable	BASE-R PCS test-pattern control register	3.42.2	rx_test_mode
LPI_FW	LPI fast wake enable	3.20.0	LPI_FW
FEC bypass correction enable	RS-FEC control register	1.200.0	FEC_bypass_correction_enable

Table 119–4—MDIO/PCS status variable mapping

MDIO status variable	PCS register name	Register/ bit number	PCS status variable
BASE-R and 10GBASE-T receive link status	BASE-R and 10GBASE-T PCS status 1 register	3.32.12	PCS_status
BASE-R and 10GBASE-T PCS high BER	BASE-R and 10GBASE-T PCS status 1 register	3.32.1	hi_ber - TBD
Lane *x* aligned	Multi-lane BASE-R PCS alignment status 3 and 4 registers	3.52.7:0 3.53.7:0	am_lock<*x*>
PCS lane alignment status	Multi-lane BASE-R PCS alignment status 1 register	3.50.12	align_status
BER	BASE-R and 10GBASE-T PCS status 2 register BER high order counter register	3.33.13:8 3.44.15:0	ber_count - TBD
Test-pattern error counter	BASE-R PCS test-pattern error counter register	3.43.15:0	test_pattern_error_count
Lane *x* mapping	Lane *x* mapping register	3.400 through 3.415	lane_mapping
FEC bypass correction ability	RS-FEC status register	1.201.0	FEC_bypass_correction_ability
FEC corrected codewords	RS-FEC corrected codewords counter register	1.202, 1.203	FEC_corrected_cw_counter

Copyright © 2015 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

Table 119–4—MDIO/PCS status variable mapping *(continued)*

MDIO status variable	PCS register name	Register/ bit number	PCS status variable
FEC uncorrected codewords	RS-FEC uncorrected codewords counter register	1.204, 1.205	FEC_uncorrect-ed_cw_counter
FEC symbol errors, PCS lanes 0 to 15	RS-FEC symbol error counter register, PCS lanes 0 to 15	1.210 to 1.229, 1.600 ro 1.611	FEC_symbol_er-ror_counter_*i*
Tx LPI indication	Tx LPI indication	3.1.9	Tx LPI indication
Tx LPI received	Tx LPI received	3.1.11	Tx LPI received
Rx LPI indication	Rx LPI indication	3.1.8	Rx LPI indication
Rx LPI received	Rx LPI received	3.1.10	Rx LPI received
Wake_error_counter	Wake_error_counter	3.22	Wake_error_counter

119.4 Loopback

If a Clause 45 MDIO is implemented, then the PCS shall be placed in the loopback mode when the loopback bit from the PCS control 1 register (bit 3.0.14) is set to a one. In this mode, the PCS shall accept data on the transmit path from the CDMII and return it on the receive path to the CDMII. In addition, the PCS shall transmit what it receives from the CDMII to the PMA sublayer, and shall ignore all data presented to it by the PMA sublayer.

119.5 Delay constraints

The maximum delay contributed by the 400GBASE-R PCS (sum of transmit and receive delays at one end of the link) shall be no more than TBD BT (TBD pause_quanta or TBD ns). A description of overall system delay constraints and the definitions for bit times and pause_quanta can be found in 116.4 and its references.

Copyright © 2015 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

119.6 Protocol implementation conformance statement (PICS) proforma for Clause 119, Physical Coding Sublayer (PCS) for 400 Gb/s[3]

119.6.1 Introduction

The supplier of a protocol implementation that is claimed to conform to Clause 119, Physical Coding Sublayer (PCS) for 400 Gb/s, shall complete the following protocol implementation conformance statement (PICS) proforma.

A detailed description of the symbols used in the PICS proforma, along with instructions for completing the PICS proforma, can be found in Clause 21.

119.6.2 Identification

119.6.2.1 Implementation identification

Supplier[1]	
Contact point for inquiries about the PICS[1]	
Implementation Name(s) and Version(s)[1,3]	
Other information necessary for full identification—e.g., name(s) and version(s) for machines and/or operating systems; System Name(s)[2]	
NOTE 1—Required for all implementations. NOTE 2—May be completed as appropriate in meeting the requirements for the identification. NOTE 3—The terms Name and Version should be interpreted appropriately to correspond with a supplier's terminology (e.g., Type, Series, Model).	

119.6.2.2 Protocol summary

Identification of protocol standard	IEEE Std 802.3bs-201x, Clause 119, Physical Coding Sublayer (PCS) for 400 Gb/s
Identification of amendments and corrigenda to this PICS proforma that have been completed as part of this PICS	
Have any Exception items been required? No [ ] Yes [ ] (See Clause 21; the answer Yes means that the implementation does not conform to IEEE Std 802.3bs-201x.)	

Date of Statement	

[3]*Copyright release for PICS proformas:* Users of this standard may freely reproduce the PICS proforma in this subclause so that it can be used for its intended purpose and may further publish the completed PICS.

Copyright © 2015 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

119.6.3 Major capabilities/options

Item	Feature	Subclause	Value/Comment	Status	Support
CDE400	CDMII logical interface	117, 119.1.4.1	Logical interface is supported	O	Yes [ ] No [ ]
MD	MDIO	45, 119.3	Registers and interface supported	O	Yes [ ] No [ ]
BEC	Bypass error correction	119.2.5.3	Capability is supported	O	Yes [ ] No [ ]
DC	Delay constraints	119.5	Conforms to delay constraints specified in 119.5	M	Yes [ ]
EEE	EEE capability	119.2.3.3	Capability is supported	O	Yes [ ] No [ ]
*JTM	Supports test-pattern mode	119.6.5		O	Yes [ ] No [ ]

119.6.4 PICS proforma tables for Physical Coding Sublayer (PCS) for 400 Gb/s

119.6.4.1 Transmit function

Item	Feature	Subclause	Value/Comment	Status	Support
TF1	Skew tolerance	119.2.5.1	Maximum Skew of 180 ns between PCS lanes and a maximum Skew Variation of 4 ps	M	Yes [ ]
TF2	64B/66B to 256B/257B transcoder	119.2.4.2	tx_xcoded<256:0> constructed per 119.2.4.2	M	Yes [ ]
TF3	Transmission bit ordering	119.2.4.8	First bit transmitted is bit 0	M	Yes [ ]
TF4	Pad value	119.2.4.4	PRBS9	M	Yes [ ]
TF5	Alignment marker insertion	119.2.4.4	First 2056 message bits to be transmitted from every 8096th codeword	M	Yes [ ]
TF6	Pre-FEC distribution	119.2.4.5	Distribute the data to two FEC codewords	M	Yes [ ]
TF~~6~~7	Reed-Solomon encoder for 400GBASE-R PCS	119.2.4.5	RS(544,514)	M	Yes [ ]
TF~~7~~8	Symbol distribution	119.2.4.7	Distribution is ~~TBD~~based on 10b symbols	M	Yes [ ]

Copyright © 2015 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

119.6.4.2 Receive function

Item	Feature	Subclause	Value/Comment	Status	Support
RF1	Skew tolerance	119.2.5.1	Maximum Skew of 180 ns between PCS lanes and a maximum Skew Variation of 4 ns	M	Yes [ ]
RF2	Lane reorder and de-interleave	119.2.5.2	Order the PCS lanes according to the PCS lane number, and de-interleave the FEC codewords	M	Yes [ ]
RF3	Reed-Solomon decoder	119.2.5.3	Corrects any combination of up to t=15 symbol errors in a codeword unless error correction bypassed	M	Yes [ ]
RF4	Reed-Solomon decoder	119.2.5.3	Capable of indicating when a codeword was not corrected.	M	Yes [ ]
RF5	Error indication function	119.2.5.3	Corrupts 66-bit block synchronization headers for uncorrected errored codewords (or errored codewords when correction is bypassed)	M	Yes [ ]
RF6	Error indication when error correction is bypassed	119.2.5.3	Error indication is not bypassed	M	Yes [ ] N/A [ ]
RF9	256B/257B to 64B/66B transcoder	119.2.5.6	rx_coded_j<65:0>, j=0 to 3 constructed per 119.2.5.6	M	Yes [ ]

Copyright © 2015 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

119.6.4.3 64B/66B Coding rules

Item	Feature	Subclause	Value/Comment	Status	Support
C1	Encoder (and ENCODE function) implements the code as specified	119.2.3 and 119.2.6.2.3		M	Yes [ ] No [ ]
C2	Decoder (and DECODE function) implements the code as specified	119.2.3 and 119.2.6.2.3		M	Yes [ ] No [ ]
C3	Only valid block types are transmitted	119.2.3.2		M	Yes [ ] No [ ]
C4	Invalid block types are treated as an error	119.2.3.2		M	Yes [ ] No [ ]
C5	Only valid control characters are transmitted	119.2.3.3		M	Yes [ ] No [ ]
C6	Invalid control characters are treated as an error	119.2.3.3		M	Yes [ ] No [ ]
C7	Idles do not interrupt data	119.2.3.5		M	Yes [ ] No [ ]
C8	IDLE control code insertion and deletion	119.2.3.5	Insertion or Deletion in groups of 8 /I/s	M	Yes [ ] No [ ]
C9	Sequence ordered set deletion	119.2.3.8	Only one whole ordered set of two consecutive sequence ordered sets may be deleted	M	Yes [ ] No [ ]

119.6.4.4 Scrambler and Descrambler

Item	Feature	Subclause	Value/Comment	Status	Support
S1	Scrambler	119.2.4.3	Performs as shown in Figure 49–8	M	Yes [ ] No [ ]
S2	Descrambler	119.2.5.5	Performs as shown in Figure 49–10	M	Yes [ ] No [ ]

Copyright © 2015 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

119.6.4.5 Alignment Markers

Item	Feature	Subclause	Value/Comment	Status	Support
AM1	Alignment marker insertion	119.2.4.4	Alignment markers are inserted periodically as described in section 119.2.4.4	M	Yes [ ] No [ ]
AM2	Alignment marker form	119.2.4.4	Alignment markers are formed as described in section 119.2.4.4	M	Yes [ ] No [ ]
AM3	Lane mapping	119.2.6.3	PCS lane number is captured	MD:M	Yes [ ] No [ ]

119.6.5 Test-pattern modes

Item	Feature	Subclause	Value/Comment	Status	Support
JT1	Scrambled idle transmit test-pattern generator is implemented	119.2.4.9		JTM:M	Yes [ ] No [ ] N/A[ ]
JT2	Scrambled idle receive test-pattern checker is implemented	119.2.5.8		JTM:M	Yes [ ] No [ ] N/A[ ]
JT3	Transmit and receive test-pattern modes can operate simultaneously	119.2.1		JTM:M	Yes [ ] No [ ] N/A[ ]

119.6.5.1 Bit order

Item	Feature	Subclause	Value/Comment	Status	Support
B1	Transmit bit order	119.2.4.8	Placement of bits into the PCS lanes as shown in Figure 119–10	M	Yes [ ] No [ ]

119.6.6 Management

Item	Feature	Subclause	Value/Comment	Status	Support
M1	Alternate access to PCS Management objects is provided	119.3		O	Yes [ ] No [ ]

Copyright © 2015 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

119.6.6.1 State diagrams

Item	Feature	Subclause	Value/Comment	Status	Support
SM1	Alignment Marker Lock	119.2.6	Implements 16 alignment marker lock processes as depicted in Figure 119–11	M	Yes [ ] No [ ]
SM2	The SLIP functions evaluates all possible blocks	119.2.6.2.3		M	Yes [ ] No [ ]
SM3	PCS synchronization state diagram	119.2.6	Meets the requirements of Figure 119–13	M	Yes [ ] No [ ]
SM4	BER Monitor TBD			M	Yes [ ] No [ ]
SM5	Transmit process	119.2.6	Meets the requirements of Figure 119–15	M	Yes [ ] No [ ]
SM6	Receive process	119.2.6	Meets the requirements of Figure 119–16	M	Yes [ ] No [ ]

119.6.6.2 Loopback

Item	Feature	Subclause	Value/Comment	Status	Support
L1	Supports loopback	119.4		M	Yes [ ] No [ ] N/A[ ]
L2	When in loopback, transmits what it receives from the CDMII	119.4		M	Yes [ ] No [ ]

119.6.6.3 Delay constraints

Item	Feature	Subclause	Value/Comment	Status	Support
TIM1	PCS Delay Constraint	119.5	No more than TBD BT for sum of transmit and receive path delays for 400GBASE-R.	M	Yes [ ] No [ ]

Copyright © 2015 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.