

## 119. Physical Coding Sublayer (PCS) for 64B/66B, type 200GBASE-R and 400GBASE-R

### 119.1 Overview

#### 119.1.1 Scope

This clause specifies the Physical Coding Sublayer (PCS) that is common to two families of Physical Layer implementation known as 200GBASE-R and 400GBASE-R. The 200GBASE-R PCS is a sublayer of the 200 Gb/s PHYs listed in Table 116–1. The 400GBASE-R PCS is a sublayer of the 400 Gb/s PHYs listed in Table 116–2. The terms 200GBASE-R and 400GBASE-R are used when referring generally to Physical Layers using the PCS defined in this clause. Both 200GBASE-R and 400GBASE-R are 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 200GBASE-R and 400GBASE-R to other standards

Figure 119–1 depicts the relationship of the 200GBASE-R and 400GBASE-R sublayers (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 PCS FEC.

#### 119.1.3 Physical Coding Sublayer (PCS)

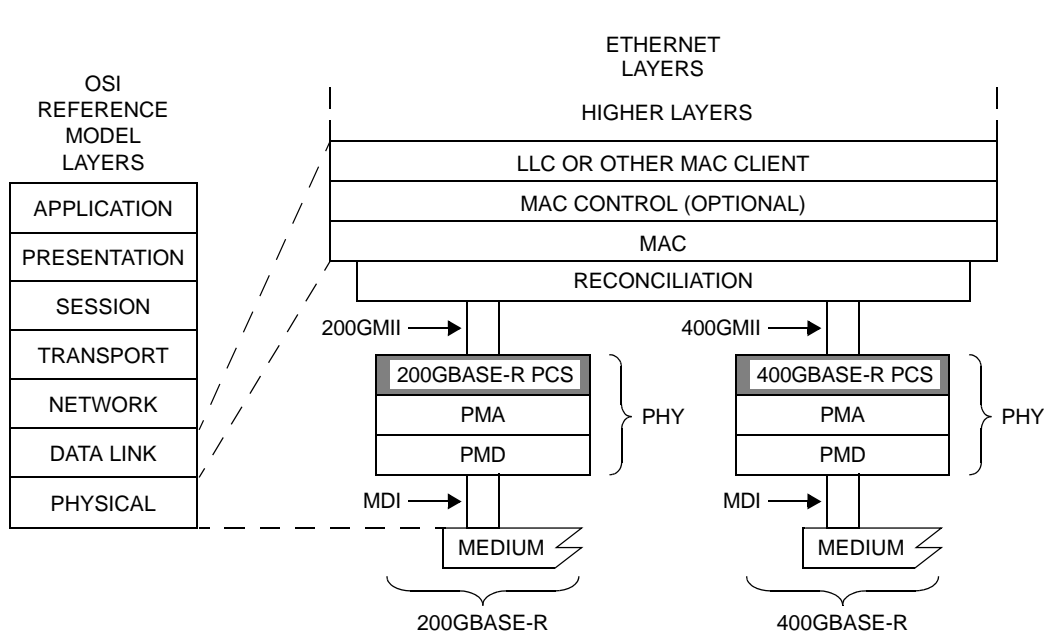
The PCS service interface is the Media Independent Interface (200GMII/400GMII), which is defined in [Clause 117](#). The 200GMII provides a uniform interface to the Reconciliation Sublayer for all 200 Gb/s PHY implementations. The 400GMII provides a uniform interface to the Reconciliation Sublayer for all 400 Gb/s PHY implementations.

The 200GBASE-R and 400GBASE-R PCSs provide all services required by the 200GMII/400GMII, including the following:

- a) Encoding (decoding) of eight 200GMII/400GMII data octets to (from) 66-bit blocks (64B/66B).
- b) Transcoding from 66-bit blocks to (from) 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 200GMII/400GMII 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 200GMII/400GMII. The lower interface of the PCS connects to the PMA sublayer to support a PMD. The 200GBASE-R PCS has a nominal rate at the PMA service interface of 26.5625 Gtransfers/s on each of 8 PCS lanes, which provides capacity for the MAC data rate of 200 Gb/s. The 400GBASE-R PCS has a nominal rate at the PMA



200GMII = 200 Gb/s MEDIA INDEPENDENT INTERFACE  
 400GMII = 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—200GBASE-R and 400GBASE-R PCS relationship to the ISO/IEC Open Systems Interconnection (OSI) reference model and IEEE 802.3 Ethernet model**

service interface of 26.5625 Gtransfers/s on each of 16 PCS lanes, 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.

#### 119.1.4.1 PCS service interface (200GMII/400GMII)

The PCS service interface allows the 200GBASE-R or 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 (200GMII/400GMII) in Clause 117.

#### 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.

### 119.1.5 Functional block diagram

Figure 119–2 provides a functional block diagram of the 200GBASE-R and 400GBASE-R PCSs.

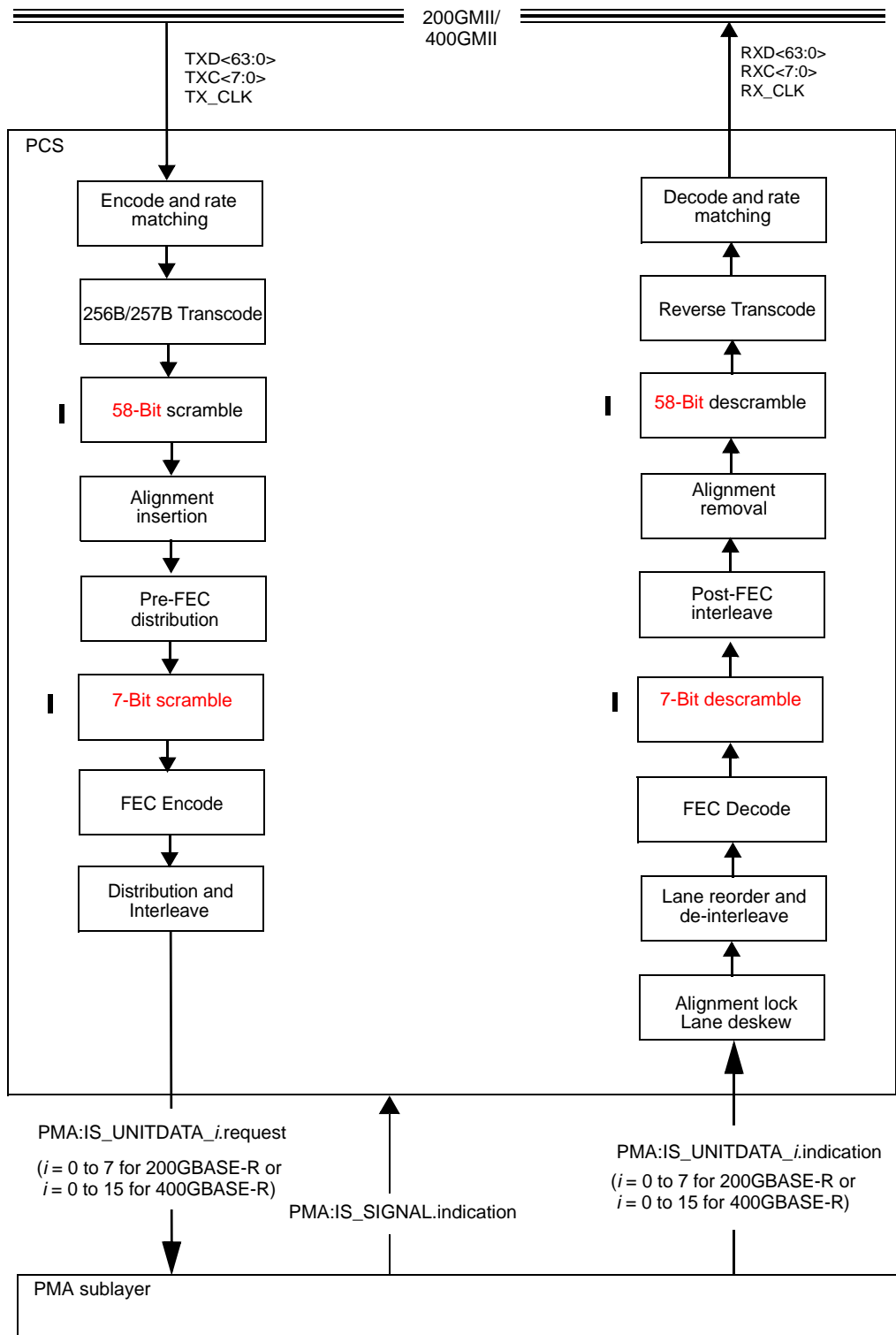


Figure 119–2—Functional block diagram

## 119.2 Physical Coding Sublayer (PCS)

### 119.2.1 Functions within the PCS

The 200GBASE-R and 400GBASE-R PCS are 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 can operate in normal mode or test-pattern mode.

When communicating with the 200GMII/400GMII, the PCS uses an eight octet-wide, synchronous data path, with packet delineation being provided by transmit control signals ( $\text{TXC}_{<n>} = 1$ ) and receive control signals ( $\text{RXC}_{<n>} = 1$ ). When communicating with the PMA, the 200GBASE-R PCS uses 8 encoded bit streams (also known as PCS lanes) and the 400GBASE-R PCS uses 16 encoded bit streams. Per direction (RX or TX), the PCS lanes originate from a common clock but may vary in phase and skew dynamically. The PMA sublayer operates independently of block and packet boundaries. The PCS provides the functions necessary to map packets between the 200GMII/400GMII format and the PMA service interface format.

When the transmit channel is in normal mode, the PCS Transmit process continuously generates 66-bit blocks based upon the  $\text{TXD}_{<63:0>}$  and  $\text{TXC}_{<7:0>}$  signals on the 200GMII/400GMII. The 66-bit blocks are transcoded to 257-bit blocks to reduce the line coding overhead. The transcoded blocks are then scrambled with a 58-bit scrambler to control clock content and baseline wander. Alignment markers are periodically added to the data stream. The data stream is distributed to two 5140-bit blocks and each 5140-bit block is scrambled again with a 7-bit scrambler and then FEC encoded to control errors. The two FEC codewords are then interleaved before data is distributed to individual PCS lanes.

Transmit data-units are sent to the service interface via the  $\text{PMA:IS\_UNITDATA}_i$ .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  $\text{PMA:IS\_UNITDATA}_i$ .request primitive.

The PCS Synchronization process continuously monitors  $\text{PMA:IS\_SIGNAL.indication(SIGNAL\_OK)}$ . When  $\text{SIGNAL\_OK}$  indicates OK, then the PCS synchronization process accepts data-units via the  $\text{PMA:IS\_UNITDATA}_i$ .indication primitive. It attains alignment marker lock based on the common marker (CM) portion that is periodically transmitted on every PCS lane. After alignment markers are found on all PCS lanes, the individual PCS lanes are identified using the unique marker portion (UM) and then re-ordered 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, aligns and reorders the individual PCS lanes, forms a single stream, and sets the  $\text{align\_status}$  flag to indicate whether the PCS has obtained alignment. The PCS then de-interleaves and processes the FEC codewords. Next the PCS re-interleaves the corrected FEC codewords on a 10-bit basis to form a single stream. The PCS then removes alignment markers, descrambles the data, transcodes the data back to 64B/66B and then decodes the 64B/66B encoded data.

The PCS shall provide transmit test-pattern mode for the scrambled idle pattern (see 119.2.4.10).

### 119.2.2 Use of blocks

The PCS maps 200GMII/400GMII 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 64B/66B coding to support transmission of control and data characters. The 64B/66B codestream is then transcoded into a 256B/257B stream and FEC bits are added in this PCS before transmission.

#### 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  $C_0$  to  $C_7$ . The control characters, /Q/ and /Fsig/, for ordered sets are labeled as  $O_0$  since they are only valid on the first octet of the 200GMII/400GMII. The control character for start is labeled as  $S_0$  for the same reason. The control character for terminate is labeled as  $T_0$  to  $T_7$ . The four trailing zero data octets in ordered sets are labeled as  $Z_4$  to  $Z_7$ .

One 200GMII/400GMII 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 200GMII/400GMII 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. 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, 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 82-5](#). 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 200GMII/400GMII, 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 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.

#### 119.2.3.3 Control codes

The same set of control characters are supported by the 200GMII/400GMII and the PCS. The representations of the control characters are the control codes. The 200GMII/400GMII encodes a control

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

The control characters and their mappings to 200GBASE-R/400GBASE-R control codes and 200GMII/400GMII control codes are specified in Table 82-1. All 200GMII/400GMII and 200GBASE-R/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

The valid and invalid blocks are identical to those in 82.2.3.5 with the exception that it is a 200GMII/400GMII data stream instead of XLGMII/CGMII.

#### 119.2.3.5 Idle (/I/)

Idle and LPI 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 Encode and rate matching

The PCS generates 66-bit blocks based upon the TXD<63:0> and TXC<7:0> signals received from the 200GMII/400GMII. One 200GMII/400GMII data transfer is encoded into one 66-bit block. If the transmit PCS 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 (or LPI) control characters.

Idle (or LPI) control characters or sequence ordered sets are removed, if necessary, to accommodate the insertion of the alignment markers. See 119.2.3.5 and 119.2.3.8 for the deletion and insertion rules, and 119.2.4.4 for more details on alignment markers.

The transmit PCS generates blocks as specified in the transmit state diagram as shown in Figure 119–14. 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.

Note—The stream of 66-bit blocks generated by this process, together with the FEC\_degraded\_SER and rx\_local\_degraded bits are used as the reference signal for mapping to OTN. See ITU-T G.709 [B50].

#### 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.

Note—This transcoder differs from that described in 91.5.2.5, there is no scrambling of the first 5 bits since this clause scrambles the complete 257-bit blocks after transcoding.

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:

- a) tx\_xcoded<0> = 1
- b) tx\_xcoded<(64j+64):(64j+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:

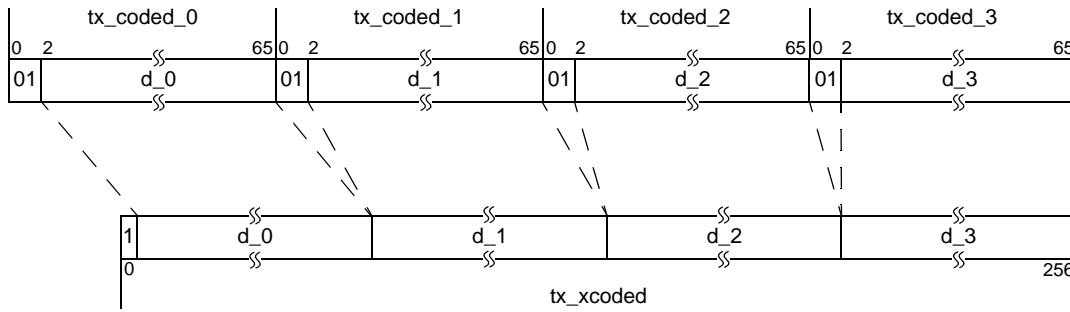
- a1) tx\_xcoded<0> = 0
- b1) tx\_xcoded<j+1> = tx\_coded\_j<1> for  $j=0$  to 3
- c1) Let  $c$  be the smallest value of  $j$  such that tx\_coded\_c<0>=1. In other words, tx\_coded\_c is the first 66-bit control block that was received in the current group of four blocks.
- d1) Let tx\_payloads<(64j+63):64j> = 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<(64c+8):5> = tx\_payloads<(64c+3):0>  
tx\_xcoded<256:(64c+9)> = tx\_payloads<255:(64c+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:

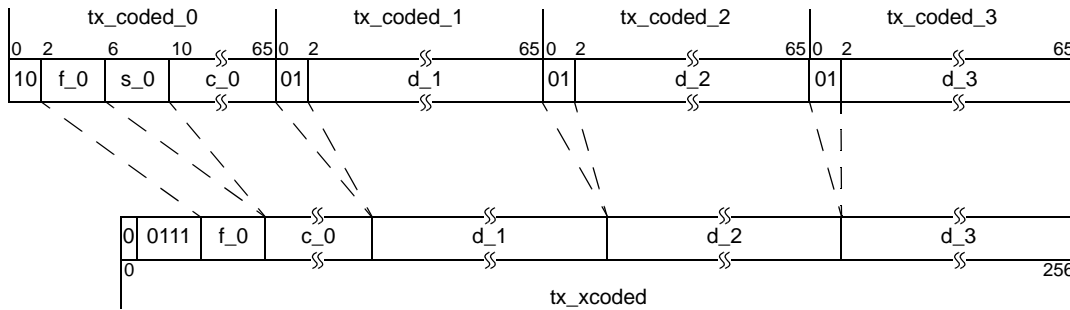
- a2) tx\_xcoded<0> = 0
- b2) tx\_xcoded<j+1> = 1 for  $j=0$  to 3
- c2) Let tx\_payloads<(64j+63):64j> = tx\_coded\_j<65:2> for  $j=0$  to 3
- d2) Omit the second nibble (based on transmission order) of tx\_coded\_0 per the following expressions.  
tx\_xcoded<8:5> = tx\_payloads<3:0>  
tx\_xcoded<256:9> = tx\_payloads<255:8>

Several examples of the construction of tx\_xcoded<256:0> are shown in Figure 119–3. In Figure 119–3, d\_j indicates the jth 66-bit block contains only data octets, c\_j indicates the jth 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.

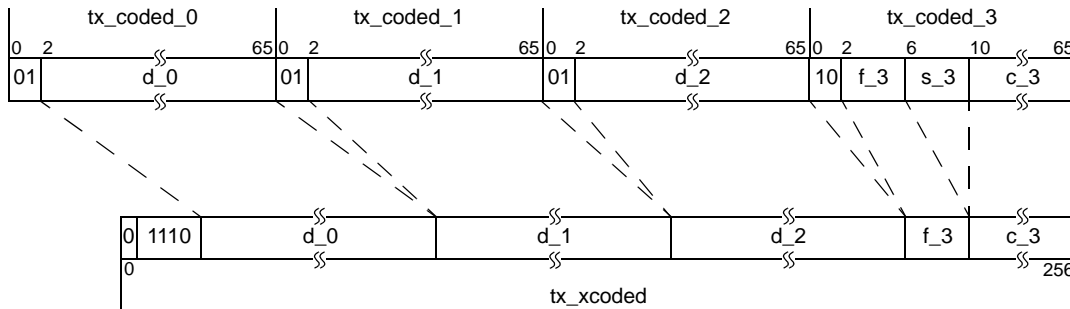
For each 257-bit block, bit 0 shall be the first bit transmitted.



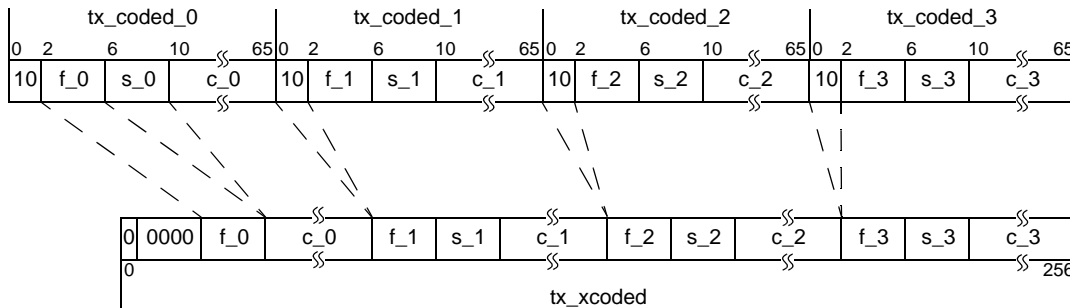
**Example 1: All data blocks**



**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 119–3—Examples of the construction of tx\_xcoded**

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54



### 119.2.4.3 58-bit Scrambler

The transcoded 257-bit block,  $tx\_xcoded<256:0>$ , shall be scrambled with a 58-bit self-synchronizing scrambler to generate  $tx\_scrambled<256:0>$ . The scrambler polynomial is identical to that in Clause 49, see Equation (49-1) for the definition of the polynomial.

### 119.2.4.4 Alignment marker mapping and insertion

In order to support deskew and reordering of the individual PCS lanes at the receive PCS, alignment markers corresponding to PCS lanes are periodically inserted after being processed by the alignment marker mapping function. The alignment marker mapping function compensates for the operation of the symbol distribution function and rearranges the alignment marker bits so that they appear on the PCS lanes intact and in the desired sequence. This preserves the properties of the alignment markers (e.g. DC balance, transition density) and provides a deterministic pattern for the purpose of synchronization.

For the 200GBASE-R PCS, an alignment marker group is composed of the alignment markers for all 8 PCS lanes plus an additional 65-bit pad and a 3-bit status field to yield the equivalent of four 257-bit blocks. For the 400GBASE-R PCS, an alignment marker group is composed of the alignment markers for all 16 PCS lanes plus an additional 133-bit pad and a 3-bit status field to yield the equivalent of eight 257-bit blocks. The alignment marker group is aligned to the beginning of two FEC messages, and interrupts any data transfer that is already in progress. The pad bits at the end of the alignment marker group shall be set to a free running PRBS9 pattern, defined by the polynomial  $x^9 + x^5 + 1$ . The initial value of the PRBS9 pattern generator may be any pattern other than all zeros. The fixed pad within the alignment markers and the PRBS9 pad at the end of the alignment maker group are ignored on receive.

Room for the alignment marker group is created by the transmit PCS (see 119.2.4.1). Special properties of the alignment marker group are that it is not scrambled, does not conform to the encoding rules as outlined in Figure 82-5 and is not transcoded. This is possible because the alignment marker group is added after encoding, transcoding, and scrambling, and removed before descrambling, transcoding, and 64B/66B decoding. The alignment marker group is not scrambled, which allows the receiver to directly search for and find the individual alignment markers, deskew the PCS lanes, and reassemble the aggregate stream before descrambling is performed. The alignment markers are formed from a known pattern that is defined to be balanced and with many transitions and therefore scrambling is not necessary.

The format of each PCS lane's alignment marker is shown in Figure 119-4. There is a portion that is common across all alignment markers (designated as  $CM_0$  to  $CM_5$ ), a unique portion per PCS lane (designated as  $UM_0$  to  $UM_5$ ), and finally a unique pad per PCS lane (designated as  $UP_0$  to  $UP_2$ ). Common synchronization logic independent of the received PCS lane number can be used with the common portion of the alignment marker.

The content of the alignment markers shall be as shown in Table 119-1 for the 200GBASE-R PCS and as shown in Table 119-2 for the 400GBASE-R PCS. The contents depend on the PCS lane number and the octet number, with  $CM_0$  through  $CM_5$  being identical across all alignment markers to allow for common synchronization across lanes. The format shown in Table 119-1 defines how the alignment markers appear on a given PCS lane. 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.

The transmit alignment marker status field allows the local PCS to communicate the status of the optional FEC degraded feature to the remote PCS. It is set as follows:

$tx\_am\_sf<2:0> = \{FEC\_degraded\_SER,0,0\}$

See 119.2.5.3 for more information on the optional FEC degrade feature.

#### 119.2.4.4.1 AM creation for the 200GBASE-R PCS

For the 200GBASE-R PCS, the alignment marker mapping function creates a set of 8 alignment markers, and in combination with an additional 65-bit PRBS9 pad and a 3-bit status field, the PCS generates an alignment marker group. Let  $am_x<119:0>$  be the alignment marker for PCS lane  $x$ ,  $x=0$  to 7, where bit 0 is the first bit transmitted. The alignment markers shall be mapped to  $am\_mapped<959:0>$  in a manner that yields the same result as the following process.

For  $x=0$  to 7,  $am_x<119:0>$  is constructed as follows:

$am_x<119:0>$  is set to  $CM_0, CM_1, CM_2, UP_0, CM_3, CM_4, CM_5, UP_1, UM_0, UM_1, UM_2, UP_2, UM_3, UM_4$  and  $UM_5$ , as shown in Figure 119–4 (bits 119:0) using the values in Table 119–1 for PCS lane number  $x$ .

As an example, the variable  $am_0$  is sent as (left most bit sent first, showing the first 32 bits transmitted of  $am_0$ ):

01011001 01010010 01100100 10100000

The variable  $am\_mapped$  is then derived from 10-bit interleaving the group of 8 alignment markers  $am_x$  per the following procedure:

For all  $k=0$  to 11

For all  $j=0$  to 3

if even( $k$ )

$am\_mapped<80k+20j+9:80k+20j> = am_{\{2j\}}<10k+9:10k>$

$am\_mapped<80k+20j+19:80k+20j+10> = am_{\{2j+1\}}<10k+9:10k>$

else

$am\_mapped<80k+20j+9:80k+20j> = am_{\{2j+1\}}<10k+9:10k>$

$am\_mapped<80k+20j+19:80k+20j+10> = am_{\{2j\}}<10k+9:10k>$

The additional 65-bit pad is appended to variable  $am\_mapped$  as follows:

$am\_mapped<1024:960> = PRBS9<64:0>$

In this expression,  $PRBS9<0>$  is the first PRBS9 bit output of the 65-bit pad.

The 3-bit transmit alignment marker status field is then appended to the variable  $am\_mapped$  as follows:

$am\_mapped<1027:1025> = tx\_am\_sf<2:0>$

Alignment marker mapping is shown in Figure 119–5.

The alignment marker group  $am\_mapped<1027:0>$  shall be inserted so it appears in the output stream every  $81\ 920 \times 257$ -bit blocks. The variable  $tx\_scrambled\_am<10279:0>$  is constructed in one of two ways. Let the set of vectors  $tx\_scrambled\_i<256:0>$  represent consecutive values of  $tx\_scrambled<256:0>$ .

For a 10280-bit block with an alignment marker group inserted:

$tx\_scrambled\_am<1027:0> = am\_mapped<1027:0>$

For all  $i=0$  to 35

$tx\_scrambled\_am<257i+1284:257i+1028> = tx\_scrambled\_i<256:0>$

For a 10280-bit block without an alignment marker group:

For all  $i=0$  to 39

$tx\_scrambled\_am<257i+256:257i> = tx\_scrambled\_i<256:0>$

Alignment marker repetition rate is shown in Figure 119–7.

#### 119.2.4.4.2 AM creation for the 400GBASE-R PCS

For the 400GBASE-R PCS, the alignment marker mapping function creates a set of 16 alignment markers, and in combination with an additional 133-bit PRBS9 pad and a 3-bit status field, the PCS generates an alignment marker group.

Let  $am_x<119:0>$  be the alignment marker for PCS lane  $x$ ,  $x=0$  to 15, where bit 0 is the first bit transmitted. The alignment markers shall be mapped to  $am\_mapped<1919:0>$  in a manner that yields the same result as the following process.

For  $x=0$  to 15,  $am_x<119:0>$  is constructed as follows:

$am_x<119:0>$  is set to  $CM_0, CM_1, CM_2, UP_0, CM_3, CM_4, CM_5, UP_1, UM_0, UM_1, UM_2, UP_2, UM_3, UM_4$  and  $UM_5$ , as shown in Figure 119–4 (bits 119:0) using the values in Table 119–2 for PCS lane number  $x$ .

As an example, the variable  $am_0$  is sent as (left most bit sent first, showing the first 32 bits transmitted of  $am_0$ ):

01011001 01010010 01100100 01101101

The variable  $am\_mapped$  is then derived from 10-bit interleaving the group of 16 alignment markers  $am_x$  per the following procedure:

For all  $k=0$  to 11

For all  $j=0$  to 7

if even( $k$ )

$am\_mapped<160k+20j+9:160k+20j> = am_{\{2j\}}<10k+9:10k>$

$am\_mapped<160k+20j+19:160k+20j+10> = am_{\{2j+1\}}<10k+9:10k>$

else

$am\_mapped<160k+20j+9:160k+20j> = am_{\{2j+1\}}<10k+9:10k>$

$am\_mapped<160k+20j+19:160k+20j+10> = am_{\{2j\}}<10k+9:10k>$

The additional 133-bit pad is appended to variable  $am\_mapped$  as follows:

$am\_mapped<2052:1920> = PRBS9<132:0>$

In this expression,  $PRBS9<0>$  is the first PRBS9 bit output of the 133-bit pad.

The 3-bit transmit alignment marker status field is then appended to the variable  $am\_mapped$  as follows:

$am\_mapped<2055:2053> = tx\_am\_sf<2:0>$

Alignment marker mapping is shown in Figure 119–6.

The alignment marker group  $am\_mapped<2055:0>$  shall be inserted so it appears in the output stream every  $163\ 840 \times 257$ -bit blocks. The variable  $tx\_scrambled\_am<10279:0>$  is constructed in one of two ways. Let the set of vectors  $tx\_scrambled\_i<256:0>$  represent consecutive values of  $tx\_scrambled<256:0>$ .

For a 10280-bit block with an alignment marker group inserted:

$tx\_scrambled\_am<2055:0> = am\_mapped<2055:0>$

For all  $i=0$  to 31

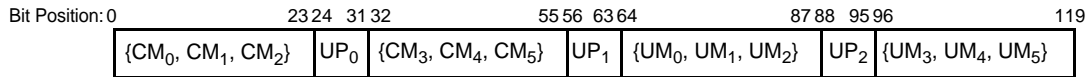
$tx\_scrambled\_am<257i+2312:257i+2056> = tx\_scrambled\_i<256:0>$

For a 10280-bit block without an alignment marker group:

For all  $i=0$  to 39

$tx\_scrambled\_am<257i+256:257i> = tx\_scrambled\_i<256:0>$

Alignment marker repetition rate is shown in Figure 119–8.



**Figure 119-4—Alignment marker format**

**Table 119-1—200GBASE-R alignment marker encodings**

PCS lane number	Encoding <sup>a</sup> {CM <sub>0</sub> , CM <sub>1</sub> , CM <sub>2</sub> , UP <sub>0</sub> , CM <sub>3</sub> , CM <sub>4</sub> , CM <sub>5</sub> , UP <sub>1</sub> , UM <sub>0</sub> , UM <sub>1</sub> , UM <sub>2</sub> , UP <sub>2</sub> , UM <sub>3</sub> , UM <sub>4</sub> , UM <sub>5</sub> }
0	0x9A, 0x4A, 0x26, 0x05, 0x65, 0xB5, 0xD9, 0xD6, 0xB3, 0xC0, 0x8C, 0x29, 0x4C, 0x3F, 0x73
1	0x9A, 0x4A, 0x26, 0x04, 0x65, 0xB5, 0xD9, 0x67, 0x5A, 0xDE, 0x7E, 0x98, 0xA5, 0x21, 0x81
2	0x9A, 0x4A, 0x26, 0x46, 0x65, 0xB5, 0xD9, 0xFE, 0x3E, 0xF3, 0x56, 0x01, 0xC1, 0x0C, 0xA9
3	0x9A, 0x4A, 0x26, 0x5A, 0x65, 0xB5, 0xD9, 0x84, 0x86, 0x80, 0xD0, 0x7B, 0x79, 0x7F, 0x2F
4	0x9A, 0x4A, 0x26, 0xE1, 0x65, 0xB5, 0xD9, 0x19, 0x2A, 0x51, 0xF2, 0xE6, 0xD5, 0xAE, 0xD
5	0x9A, 0x4A, 0x26, 0xF2, 0x65, 0xB5, 0xD9, 0x4E, 0x12, 0x4F, 0xD1, 0xB1, 0xED, 0xB0, 0x2E
6	0x9A, 0x4A, 0x26, 0x3D, 0x65, 0xB5, 0xD9, 0xEE, 0x42, 0x9C, 0xA1, 0x11, 0xBD, 0x63, 0x5E
7	0x9A, 0x4A, 0x26, 0x22, 0x65, 0xB5, 0xD9, 0x32, 0xD6, 0x76, 0x5B, 0xCD, 0x29, 0x89, 0xA4

<sup>a</sup>Each octet is transmitted LSB to MSB.




1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54

**Table 119–2—400GBASE-R alignment marker encodings**

PCS lane number	Encoding <sup>a</sup> {CM <sub>0</sub> , CM <sub>1</sub> , CM <sub>2</sub> , UP <sub>0</sub> , CM <sub>3</sub> , CM <sub>4</sub> , CM <sub>5</sub> , UP <sub>1</sub> , UM <sub>0</sub> , UM <sub>1</sub> , UM <sub>2</sub> , UP <sub>2</sub> , UM <sub>3</sub> , UM <sub>4</sub> , UM <sub>5</sub> }
0	0x9A, 0x4A, 0x26, 0xB6, 0x65, 0xB5, 0xD9, 0xD9, 0x01, 0x71, 0xF3, 0x26, 0xFE, 0x8E, 0x0C
1	0x9A, 0x4A, 0x26, 0x04, 0x65, 0xB5, 0xD9, 0x67, 0x5A, 0xDE, 0x7E, 0x98, 0xA5, 0x21, 0x81
2	0x9A, 0x4A, 0x26, 0x46, 0x65, 0xB5, 0xD9, 0xFE, 0x3E, 0xF3, 0x56, 0x01, 0xC1, 0x0C, 0xA9
3	0x9A, 0x4A, 0x26, 0x5A, 0x65, 0xB5, 0xD9, 0x84, 0x86, 0x80, 0xD0, 0x7B, 0x79, 0x7F, 0x2F
4	0x9A, 0x4A, 0x26, 0xE1, 0x65, 0xB5, 0xD9, 0x19, 0x2A, 0x51, 0xF2, 0xE6, 0xD5, 0xAE, 0x0D
5	0x9A, 0x4A, 0x26, 0xF2, 0x65, 0xB5, 0xD9, 0x4E, 0x12, 0x4F, 0xD1, 0xB1, 0xED, 0xB0, 0x2E
6	0x9A, 0x4A, 0x26, 0x3D, 0x65, 0xB5, 0xD9, 0xEE, 0x42, 0x9C, 0xA1, 0x11, 0xBD, 0x63, 0x5E
7	0x9A, 0x4A, 0x26, 0x22, 0x65, 0xB5, 0xD9, 0x32, 0xD6, 0x76, 0x5B, 0xCD, 0x29, 0x89, 0xA4
8	0x9A, 0x4A, 0x26, 0x60, 0x65, 0xB5, 0xD9, 0x9F, 0xE1, 0x73, 0x75, 0x60, 0x1E, 0x8C, 0x8A
9	0x9A, 0x4A, 0x26, 0x6B, 0x65, 0xB5, 0xD9, 0xA2, 0x71, 0xC4, 0x3C, 0x5D, 0x8E, 0x3B, 0xC3
10	0x9A, 0x4A, 0x26, 0xFA, 0x65, 0xB5, 0xD9, 0x04, 0x95, 0xEB, 0xD8, 0xFB, 0x6A, 0x14, 0x27
11	0x9A, 0x4A, 0x26, 0x6C, 0x65, 0xB5, 0xD9, 0x71, 0x22, 0x66, 0x38, 0x8E, 0xDD, 0x99, 0xC7
12	0x9A, 0x4A, 0x26, 0x18, 0x65, 0xB5, 0xD9, 0x5B, 0xA2, 0xF6, 0x95, 0xA4, 0x5D, 0x09, 0x6A
13	0x9A, 0x4A, 0x26, 0x14, 0x65, 0xB5, 0xD9, 0xCC, 0x31, 0x97, 0xC3, 0x33, 0xCE, 0x68, 0x3C
14	0x9A, 0x4A, 0x26, 0xD0, 0x65, 0xB5, 0xD9, 0xB1, 0xCA, 0xFB, 0xA6, 0x4E, 0x35, 0x04, 0x59
15	0x9A, 0x4A, 0x26, 0xB4, 0x65, 0xB5, 0xD9, 0x56, 0xA6, 0xBA, 0x79, 0xA9, 0x59, 0x45, 0x86

<sup>a</sup>Each octet is transmitted LSB to MSB.

PCS lane, <i>i</i>	am_mapped 10-bit symbol index, <i>k</i>													
	0	1	2	3	4	5	6	7	8	9	10	11	12	
0	A	B	A	B	A	B	A	B	A	B	A	B	A	B
1	B	A	B	A	B	A	B	A	B	A	B	A	B	A
2	A	B	A	B	A	B	A	B	A	B	A	B	A	B
3	B	A	B	A	B	A	B	A	B	A	B	A	B	A
4	A	B	A	B	A	B	A	B	A	B	A	B	A	B
5	B	A	B	A	B	A	B	A	B	A	B	A	B	A
6	A	B	A	B	A	B	A	B	A	B	A	B	A	B
7	B	A	B	A	B	A	B	A	B	A	B	A	B	A

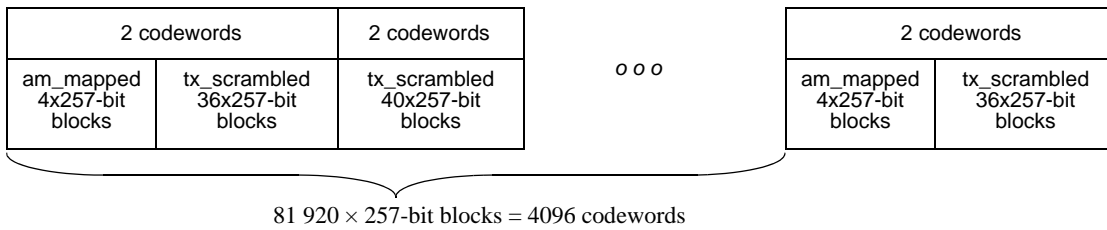
 = 65-bit pad    
  = 3-bit status field    
  = Resumption of 257-bit blocks  
 A = from FEC codeword A     B = from FEC codeword B

**Figure 119–5—200GBASE-R alignment marker mapping to PCS lanes**

PCS lane, <i>i</i>	am_mapped 10-bit symbol index, <i>k</i>												12
	0	1	2	3	4	5	6	7	8	9	10	11	
0	A	B	A	B	A	B	A	B	A	B	A	B	A
1	B	A	B	A	B	A	B	A	B	A	B	A	B
2	A	B	A	B	A	B	A	B	A	B	A	B	A
3	B	A	B	A	B	A	B	A	B	A	B	A	B
4	A	B	A	B	A	B	A	B	A	B	A	B	A
5	B	A	B	A	B	A	B	A	B	A	B	A	B
6	A	B	A	B	A	B	A	B	A	B	A	B	A
7	B	A	B	A	B	A	B	A	B	A	B	A	B
8	A	B	A	B	A	B	A	B	A	B	A	B	A
9	B	A	B	A	B	A	B	A	B	A	B	A	B
10	A	B	A	B	A	B	A	B	A	B	A	B	A
11	B	A	B	A	B	A	B	A	B	A	B	A	B
12	A	B	A	B	A	B	A	B	A	B	A	B	A
13	B	A	B	A	B	A	B	A	B	A	B	A	B
14	A	B	A	B	A	B	A	B	A	B	A	B	A
15	B	A	B	A	B	A	B	A	B	A	B	A	B

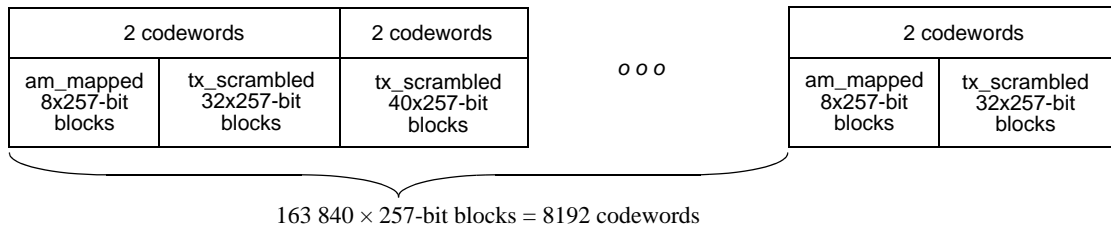
= 133-bit pad   
  = 3-bit status field   
  = Resumption of 257-bit blocks  
 A = from FEC codeword A                      B = from FEC codeword B

**Figure 119-6—400GBASE-R alignment marker mapping to PCS lanes**



**Figure 119-7—200GBASE-R alignment marker insertion period**

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54



**Figure 119–8—400GBASE-R alignment marker insertion period**

#### 119.2.4.5 Pre-FEC distribution

In order to improve error correction capability, each set of two consecutive Reed-Solomon FEC codewords is interleaved before being distributed to form the PCS lanes. To enable this interleaving, the Pre-FEC distribution function receives a 10280-bit block `tx_scrambled_am`, and shall perform a 10-bit symbol round robin distribution to form two 514-symbol FEC messages, `pmA` and `pmB`, which are subsequently scrambled by the 7-bit scrambler. The following describes the 10-bit round robin distribution process.

For all  $i=0$  to 513

$$\begin{aligned} pm_A\langle(513-i)\rangle &= tx\_scrambled\_am\langle(20i+9):(20i)\rangle \\ pm_B\langle(513-i)\rangle &= tx\_scrambled\_am\langle(20i+19):(20i+10)\rangle \end{aligned}$$

#### 119.2.4.6 Pre-FEC scrambling

In order to improve clock content once data is distributed and multiplexed, each message shall be scrambled with the scrambler as defined in Figure 50-9. The scrambler state is reset to 111111 prior to the first bit of each message. For the 200GBASE-R PCS, the first 480 bits of message `pmA` or message `pmB` are not scrambled if they contain an alignment marker group, though the scrambler continues to advance. For the 400GBASE-R PCS, the first 960 bits of message `pmA` or message `pmB` are not scrambled if they contain an alignment marker group, though the scrambler continues to advance.

Message `mA` and message `mB` are created by scrambling message `pmA` and message `pmB` respectively. These messages are subsequently each encoded by the RS FEC.

#### 119.2.4.7 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 shall implement an  $RS(544,514)$  based FEC encoder. The PCS distributes a group of  $40 \times 257$ -bit blocks from `tx_scrambled_am` on a 10-bit round robin basis into two 5140-bit message blocks, `mA` and `mB`, as described in 119.2.4.5. These are then encoded using  $RS(544,514)$  encoder into codeword A and codeword B, respectively. The  $RS(544,514)$  code is based on the generating polynomial given by Equation (119–1).

$$g(x) = \prod_{j=0}^{2t-1} (x - \alpha^j) = g_{2t}x^{2t} + g_{2t-1}x^{2t-1} + \dots + g_1x + g_0 \quad (119-1)$$

In Equation (119-1),  $\alpha$  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  $m_0$ .

$$m(x) = m_{k-1}x^{n-1} + m_{k-2}x^{n-2} + \dots + m_1x^{2t+1} + m_0x^{2t} \quad (119-2)$$

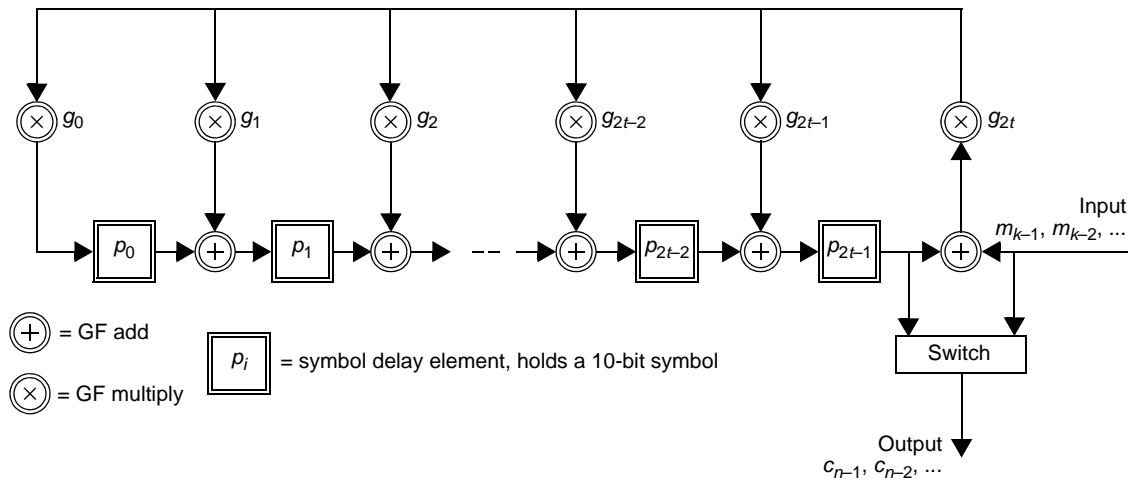
Each message symbol  $m_i$  is the bit vector  $(m_{i,9}, m_{i,8}, \dots, m_{i,1}, m_{i,0})$ , which is identified with the element  $m_{i,9}\alpha^9 + m_{i,8}\alpha^8 + \dots + m_{i,1}\alpha + m_{i,0}$  of the finite field. The message symbols are composed of the variable  $m_A$  for codeword A and  $m_B$  for codeword B (including a group of alignment markers when appropriate). 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  $p_0$ .

$$p(x) = p_{2t-1}x^{2t-1} + p_{2t-2}x^{2t-2} + \dots + p_1x + p_0 \quad (119-3)$$

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 119-9. 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 first and the coefficient of the lowest power of  $x$ ,  $c_0 = p_0$  is last.



**Figure 119-9—Reed-Solomon encoder functional model**

The coefficients of the generator polynomial for the RS(544,514) code are presented in Table 119-3. Example codewords for the RS(544,514) code are provided in Annex 119A.



**Table 119–3—Coefficients of the generator polynomial  $g_j$  (decimal)**

$i$	$g_i$	$i$	$g_i$	$i$	$g_i$
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		

#### 119.2.4.8 Symbol distribution

Once the data has been FEC encoded, two FEC codewords ( $c_A<543:0>$  and  $c_B<543:0>$ ) are interleaved on a 10-bit basis before the data is distributed to each PCS lane.

The interleaving of two codewords for the 200GBASE-R PCS shall follow this procedure:

For all  $k=0$  to 135

For all  $j=0$  to 3

if even( $k$ )

$$tx\_out<8k+2j> = c_A<543-4k-j>$$

$$tx\_out<8k+2j+1> = c_B<543-4k-j>$$

else

$$tx\_out<8k+2j> = c_B<543-4k-j>$$

$$tx\_out<8k+2j+1> = c_A<543-4k-j>$$

The interleaving of two codewords for the 400GBASE-R PCS shall follow this procedure:

For all  $k=0$  to 67

For all  $j=0$  to 7

if even( $k$ )

$$tx\_out<16k+2j> = c_A<543-8k-j>$$

$$tx\_out<16k+2j+1> = c_B<543-8k-j>$$

else

$$tx\_out<16k+2j> = c_B<543-8k-j>$$

$$tx\_out<16k+2j+1> = c_A<543-8k-j>$$

Once the data has been Reed-Solomon encoded and interleaved, it shall be distributed to 8 PCS lanes for a 200GBASE-R PCS and to 16 PCS lanes for a 400GBASE-R PCS, one 10-bit symbol at a time, from the lowest to the highest PCS lane. The first bit transmitted from each 10-bit symbol is bit 0.

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54

### 119.2.4.9 Transmit bit ordering and distribution

The transmit bit ordering and distribution are illustrated in Figure 119–10 for a 200GBASE-R PCS.

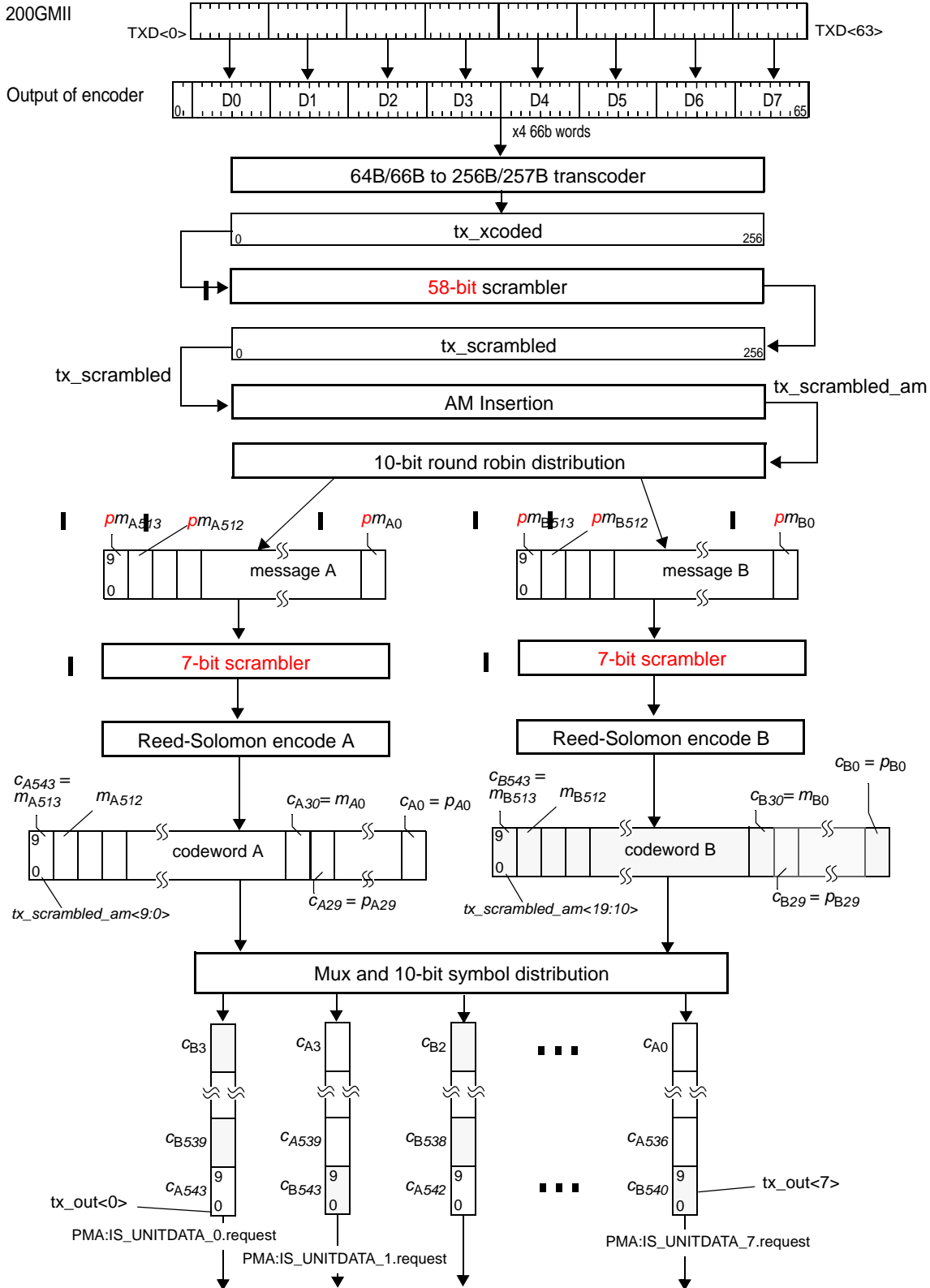


Figure 119–10—200GBASE-R Transmit bit ordering and distribution

The transmit bit ordering and distribution is illustrated in Figure 119–11 for a 400GBASE-R PCS.

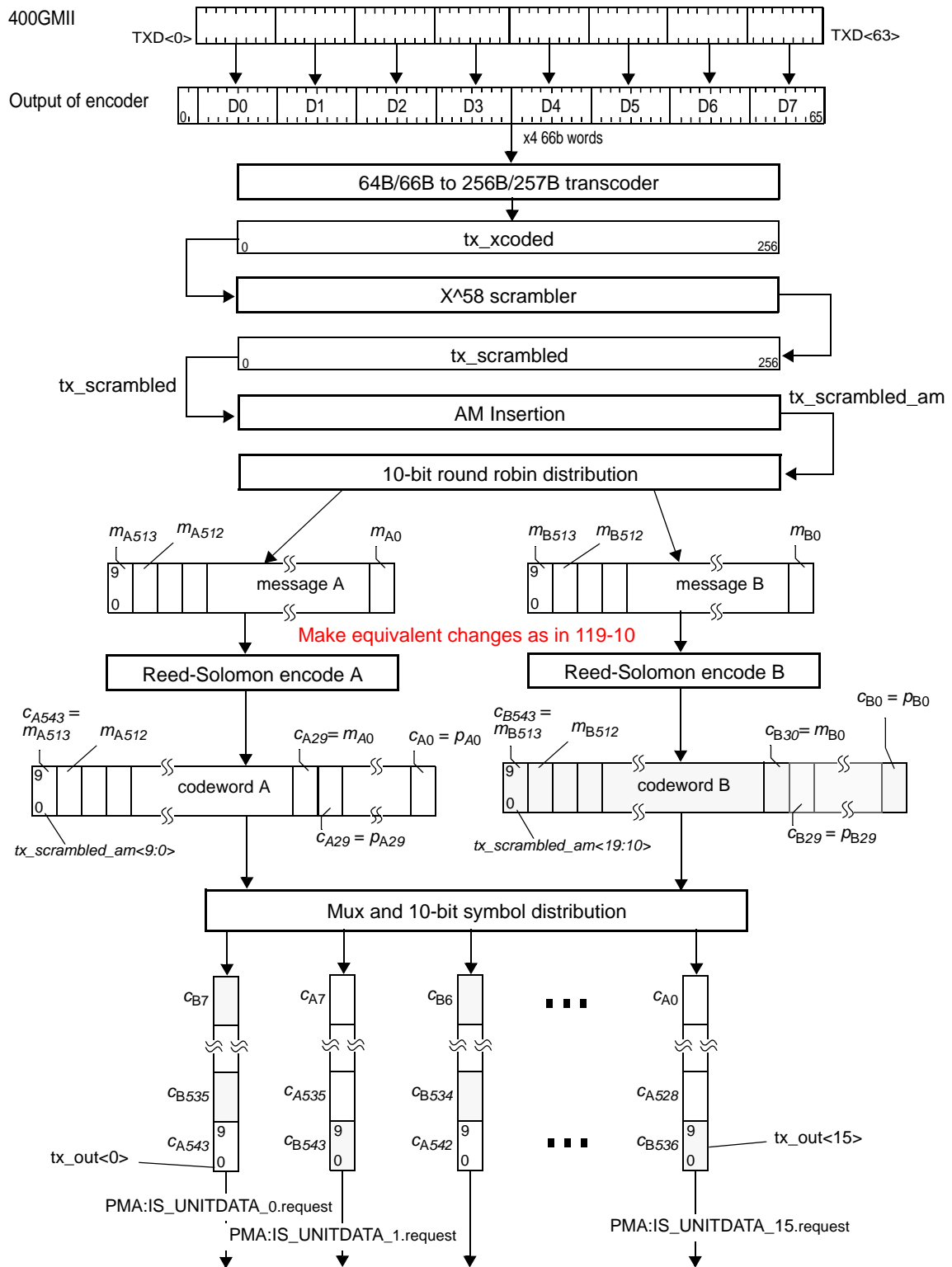


Figure 119–11—400GBASE-R Transmit bit ordering and distribution

#### 119.2.4.10 Test-pattern generators

The PCS shall have the ability to generate a scrambled idle test pattern which 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 82-5. The test pattern is sent continuously and is transcoded, 58-bit scrambled, alignment markers are inserted, 7-bit scrambled, and finally 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 119.2.4.8).

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 receive PCS forms  $n$  separate bit streams by concatenating the bits from each of the  $n$  PMA:IS\_UNIT-DATA\_ $i$ .indication primitives in the order they are received (where  $n = 8$  for a 200GBASE-R PCS and  $n = 16$  for a 400GBASE-R PCS). It obtains lock to the alignment markers as specified by the alignment marker lock state diagram shown in Figure 119–12. Note that alignment marker lock is achieved before FEC codewords are processed and therefore the alignment markers are processed in a high error probability environment.

After alignment marker lock is achieved on each of the  $n$  lanes (bit streams), all inter-lane Skew is removed as specified by the PCS synchronization state diagram shown in Figure 119–13. The PCS receive function shall support a maximum Skew of 180 ns, and maximum Skew Variation of 4 ns, between PCS lanes.

##### 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 unique portion (UM<sub>0</sub> to UM<sub>5</sub>) of 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 two FEC codewords are de-interleaved to reconstruct the original stream of two FEC codewords.

##### 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 Reed-Solomon decoder shall be capable of correcting any combination of up to  $t=15$  symbol errors in a codeword. The Reed-Solomon decoder 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  $10^{-16}$ . This limit is also expected to apply for  $t+2$  errors,  $t+3$  errors, and so on.

If bypass error indication is not supported or not enabled, when the Reed-Solomon decoder determines that a codeword contains errors that were not corrected, it shall cause the PCS receive function to set every 66-bit block within the two associated codewords to an error block (set to EBLOCK\_R). This may be achieved by setting the synchronization header to 11 for all 66-bit blocks created from these codewords by the 256B/257B to 64B/66B transcoder.

The Reed-Solomon decoder may optionally provide the ability to bypass the error indication feature to reduce the delay contributed by the FEC function. The presence of this option is indicated by the assertion of the `FEC_bypass_indication_ability` variable (see 119.3). When the option is provided it is enabled by the assertion of the `FEC_bypass_indication_enable` variable (see 119.3).

When `FEC_bypass_indication_enable` is asserted, additional error monitoring is performed by the Reed-Solomon decoder to reduce the likelihood that errors in a packet are not detected. The Reed-Solomon decoder counts the number of symbol errors detected on all PCS lanes in consecutive non-overlapping blocks of 8192 codewords. When the number of symbol errors in a block of 8192 codewords exceeds 5560, the Reed-Solomon decoder shall cause the PCS receive function to set every 66-bit block to an error block (set to `EBLOCK_R`) for a period of 60 ms to 75 ms. This may be achieved by setting the synchronization header to 11 for all 66-bit blocks created by the 256B/257B to 64B/66B transcoder for this time period.

The Reed-Solomon decoder may optionally provide the ability to signal a degradation of the received signal. The presence of this option is indicated by the assertion of the `FEC_degraded_SER_ability` variable (see 119.3). When the option is provided it is enabled by the assertion of the `FEC_degraded_SER_enable` variable (see 119.3).

When `FEC_degraded_SER_enable` is asserted, additional error monitoring is performed by the PCS. The Reed-Solomon decoder counts the number of symbol errors detected on all PCS lanes in consecutive non-overlapping blocks of `FEC_degraded_SER_interval` (see 119.3.1) codewords, where the least significant bit of `FEC_degraded_SER_interval` is ignored (evaluated as 0) to make the number of codewords even. When the number of symbol errors exceeds the threshold set in `FEC_degraded_SER_activate_threshold` (see 119.3.1), the `FEC_degraded_SER` bit (see 119.3.1) is set. At the end of each interval, if the number of symbol errors is less than `FEC_degraded_SER_deactivate_threshold`, the `FEC_degraded_SER` bit is cleared. If either `FEC_degraded_SER_ability` or `FEC_degraded_SER_enable` is de-asserted then the `FEC_degraded_SER` bit is cleared.

#### 119.2.5.4 Post FEC descramble

After the Reed-Solomon decoder processes the data, messages are descrambled by reversing the 7-bit scrambling shown in Figure 50-9.

#### 119.2.5.5 Post FEC interleave

After the messages are descrambled, data is interleaved on a 10-bit basis into `rx_scrambled_am` from two messages corresponding to 40 transcoded blocks in order to recreate the transmitted data stream.

#### 119.2.5.6 Alignment marker removal

For the 200GBASE-R PCS, every 81 920 x 257-bit blocks (corresponds to 4096 codewords) the first 1028 bits of `rx_scrambled_am` blocks is the vector `am_rx<1027:0>` 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 3-bit receive alignment marker status field is assigned from the variable `am_rx` as follows:

`rx_am_sf<2:0> = am_rx<1027:1025>`

For the 400GBASE-R PCS, every 163 840 x 257-bit blocks (corresponds to 8192 codewords) the first 2056 bits of `rx_scrambled_am` blocks 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 3-bit receive alignment marker status field is assigned from the variable `am_rx` as follows:

`rx_am_sf<2:0> = am_rx<2055:2053>`

The vector `am_rx` shall be removed from `rx_scrambled_am` to create `rx_scrambled` prior to descrambling.

### 119.2.5.7 58-bit Descrambler

The descrambler shall process  $rx\_scrambled_{<256:0>}$  to reverse the effect of the 58-bit scrambler using the polynomial given in Equation (49-1).

### 119.2.5.8 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)  $rx\_coded\_j_{<65:2>} = rx\_xcoded_{<(64j+64):(64j+1>}$  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:  
 $rx\_payloads_{<(64c+3):0>} = rx\_xcoded_{<(64c+8):5>}$   
 $rx\_payloads_{<(64c+7):(64c+4>}$  is set to a value derived by cross-referencing  $rx\_payloads_{<(64c+3):64c>}$  using Figure 82-5. For example, if  $rx\_payloads_{<(64c+3):64c>}$  is 0xE then  $rx\_payloads_{<(64c+7):(64c+4>}$  is 0x1. If no match to  $rx\_payloads_{<(64c+3):64c>}$  is found,  $rx\_payloads_{<(64c+7):(64c+4>}$  is set to 0000  
 $rx\_payloads_{<255:(64c+8)>} = rx\_xcoded_{<256:(64c+9)>}$
- c2)  $rx\_coded\_j_{<65:2>} = rx\_payloads_{<(64j+63):64j>}$  for  $j=0$  to 3
- d2) If  $rx\_xcoded_{<j+1>}=0$ ,  $rx\_coded\_j_{<0>}=1$  and  $rx\_coded\_j_{<1>}=0$  for  $j=0$  to 3
- e2) If  $rx\_xcoded_{<j+1>}=1$ ,  $rx\_coded\_j_{<0>}=0$  and  $rx\_coded\_j_{<1>}=1$  for  $j=0$  to 3
- f2) If  $rx\_payloads_{<(64c+7):(64c+4>} = 0000$ ,  $rx\_coded\_c_{<1>}=1$  (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  $c = 0$
- b3) Let  $rx\_payloads$  be a vector representing the payloads of the four 66-bit blocks. It is derived using the following expressions.  
 $rx\_payloads_{<(64c+3):0>} = rx\_xcoded_{<(64c+8):5>}$   
 $rx\_payloads_{<(64c+7):(64c+4>} = 0000$   
 $rx\_payloads_{<255:(64c+8)>} = rx\_xcoded_{<256:(64c+9)>}$
- c3)  $rx\_coded\_j_{<65:2>} = rx\_payloads_{<(64j+63):(64j>}$  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.

Note—The stream of 66-bit blocks generated by this process is used as the reference signal for de-mapping from OTN. See ITU-T G.709 [B50].

### 119.2.5.9 Decode and rate matching

The receive PCS decodes blocks to produce RXD<63:0> and RXC<7:0> for transmission to the 200GMII/400GMII. One 200GMII/400GMII data transfer is decoded from each block. The receive PCS may insert idle control characters to compensate for the removal of alignment markers. If the receive PCS 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 (see 82.2.3.6 and 82.2.3.9 for insertion and deletion rules).

The PCS receive decodes blocks as specified in the receive state diagram shown in Figure 119–15.

## 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 200GMII/400GMII containing /E/ in all the eight character locations.

EBLOCK\_T<65:0>

66 bit vector to be sent to the transcoder containing /E/ in all the eight character locations.

LBLOCK\_R<71:0>

72 bit vector to be sent to the 200GMII/400GMII 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 transcoder 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.

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 a PCS lane according to the process described in 119.2.4.4, consists of 96 known bits. The 48 bits of the common marker portion are compared on a nibble-wise basis (12 comparisons). If 9 or more nibbles in the candidate block match the corresponding known nibbles in the common portion of the alignment marker payload, 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 $x = 0:7$ for 200GBASE-R and $x = 0:15$ for 400GBASE-R	1
current_pcs_l	2
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_pcs_l to confirm that the location of the alignment marker payload sequence has been detected.	3
cw <sub>A</sub> _bad	4
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 A and is set to false otherwise.	5
cw <sub>B</sub> _bad	6
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 B and is set to false otherwise.	7
deskew_done	8
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.	9
align_status	10
A variable set by the PCS synchronization 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 PCS synchronization process is not complete.	11
hi_ser	12
This variable is defined when the FEC_bypass_indication_ability variable is set to one. When FEC_bypass_indication_enable is set to one, this bit is set to one if the number of RS-FEC symbol errors in a window of 8192 codewords exceeds the threshold (see 119.2.5.3) and is set to zero otherwise. This variable is mapped to the bit defined in 45.2.3.47k (3.801.2).	13
pcs_alignment_valid	14
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.	15
pcs_enable_deskew	16
A Boolean variable that enables and disables the PCS synchronization 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.	17
pcs_lane	18
A variable that holds the PCS lane number 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 48 bits that are in the positions of the unique marker bits in the received alignment marker payload are compared to the expected values for a given payload position and PCS lane on a nibble-wise basis (12 comparisons). If 9 or more nibbles in the candidate block match the corresponding known nibbles for any payload position on a given PCS lane, then the PCS lane number is assigned accordingly.	19
pcs_lane_mapping< $x$ >	20
A variable that holds the value of the pcs_lane received on physical lane $x$ .	21
first_pcs_l	22
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.	23
reset	24
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.	25
restart_lock	26
Boolean variable that is set by the PCS synchronization process to restart the alignment marker	27
	28
	29
	30
	31
	32
	33
	34
	35
	36
	37
	38
	39
	40
	41
	42
	43
	44
	45
	46
	47
	48
	49
	50
	51
	52
	53
	54

lock process on all PCS lanes. It is set to true after 3 consecutive uncorrected codewords are received (3_BAD state) or when 5 Alignment Markers in a row fail to match (5_BAD state) and set to false upon entry into the LOSS_OF_ALIGNMENT state.	1
rx_coded<65:0>	2
Vector containing the input to the 64B/66B decoder. The format for this vector is shown in Figure 82-5. The leftmost bit in the figure is rx_coded<0> and the rightmost bit is rx_coded<65>.	3
rx_raw<71:0>	4
Vector containing one 200GMII/400GMII 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.	5
rx_rm_degraded	6
Boolean variable that is asserted true when the receiver detects rx_am_sf<2> asserted true for two consecutive alignment marker periods. It is deasserted when rx_am_sf<2> is deasserted for two consecutive alignment marker periods. If a Clause 45 MDIO is implemented, the status of this variable is reflected in bit 3.801.5.	7
signal_ok	8
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.	9
slip_done	10
Boolean variable that is set to true when the SLIP requested by the alignment marker lock state diagram has been completed indicating that the next candidate 120-bit block position can be tested.	11
test_amp	12
Boolean variable that is set to true when a candidate block position is available for testing and false when the FIND_1ST state is entered.	13
test_cw	14
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.	15
tx_coded<65:0>	16
Vector containing the output from the 64B/66B encoder. The format for this vector is shown in Figure 82-5. The leftmost bit in the figure is tx_coded<0> and the rightmost bit is tx_coded<65>.	17
tx_raw<71:0>	18
Vector containing one 200GMII/400GMII 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.	19

### 119.2.6.2.3 Functions

AMP_COMPARE	20
This function compares the values of first_pcs1 and current_pcs1 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_pcs1 and first_pcs1 are 0, amp_match is set to true.	21
DECODE(rx_coded<65:0>)	22
Decodes the 66-bit vector returning rx_raw<71:0>, which is sent to the 200GMII/400GMII. The DECODE function shall decode the block as specified in 119.2.3.	23
ENCODE(tx_raw<71:0>)	24
Encodes the 72-bit vector returning tx_coded<65:0>. The ENCODE function shall encode the block as specified in 119.2.3.	25
R_TYPE(rx_coded<65:0>)	26
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.	27
Values: C; The vector contains a sync header of 10 and one of the following:	28
a) A block type field of 0x1E and eight valid control characters other than /E/ or /L/;	29

- 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.
- 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 82-1](#).

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 200GMII/400GMII control code specified in [Table 82-1](#). 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 82-1](#).

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\_bad\_count

Counts the number of consecutive alignment markers that don't match the expected values for a given PCS lane.

#### amp\_counter

This counter counts the interval of *i* FEC codewords containing normal alignment marker payload sequences (where *i* is 4096 for the 200GBASE-R PCS, and 8192 for the 400GBASE-R PCS).

`cwA_bad_count` 1  
Counts the number of consecutive uncorrected FEC codewords for codeword A. This counter is set 2  
to zero when an FEC codeword A is received and `cwA_bad` is false. 3  
`cwB_bad_count` 4  
Counts the number of consecutive uncorrected FEC codewords for codeword B. This counter is set 5  
to zero when an FEC codeword B is received and `cwB_bad` is false. 6  
7

### 119.2.6.3 State diagrams 8

The 200GBASE-R PCS shall implement eight alignment marker lock processes and the 400GBASE-R PCS 9  
shall implement sixteen alignment marker lock processes as depicted in Figure 119–12. An alignment 10  
marker lock process operates independently on each lane. The alignment marker lock state diagram shown 11  
in Figure 119–12 determines when the PCS has obtained alignment marker lock to the received bit stream 12  
for a given lane of the service interface. Each alignment marker lock process looks for two valid alignment 13  
markers  $i \times 10$ -bit Reed-Solomon symbols apart (on a per PCS lane basis, where  $i = 139\,264$  for a 14  
200GBASE-R PCS and  $i = 278\,528$  for a 400GBASE-R PCS) to gain alignment marker lock. When the 15  
alignment marker lock process achieves lock for a lane, and if a Clause 45 MDIO is implemented, the PCS 16  
shall record the PCS lane number received on that lane of the service interface in the appropriate lane 17  
mapping register (3.400 to 3.415). Once in lock, a lane goes out of alignment marker lock only when 18  
`restart_lock` is signaled. This occurs when the PCS synchronization process determines that three 19  
uncorrectable codewords in a row are seen, or when the alignment marker lock process sees five alignment 20  
markers in a row that fail to match the expected pattern on a given lane. 21  
22

The PCS shall run the synchronization process as depicted in Figure 119–13. The PCS synchronization 23  
process is responsible for determining if the PCS is capable of presenting coherent data to the 24  
200GMII/400GMII. The synchronization process ensures that all PCS lanes have alignment marker lock, are 25  
locked to different alignment markers, and that the Skew is within the boundaries of what the PCS can 26  
deskew. The synchronization process, along with the alignment marker lock process, are restarted if three 27  
consecutive FEC codewords from the same codeword (A or B) are uncorrectable. 28  
29

The Transmit state diagram shown in Figure 119–14 controls the encoding of 66-bit transmitted blocks. It 30  
makes exactly one transition for each 66-bit transmit block processed. Though the Transmit state diagram 31  
sends Local Fault ordered sets when reset is asserted, the scrambler is not guaranteed to be operational 32  
during reset. Thus, the Local Fault ordered sets may not appear on the PMA service interface. 33  
34

The Receive state diagram shown in Figure 119–15 controls the decoding of received blocks. It makes 35  
exactly one transition for each receive 66-bit block processed. 36  
37

The PCS shall perform the functions of alignment marker lock, PCS synchronization, Transmit, and Receive 38  
as specified in the respective state diagrams. 39  
40

41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54

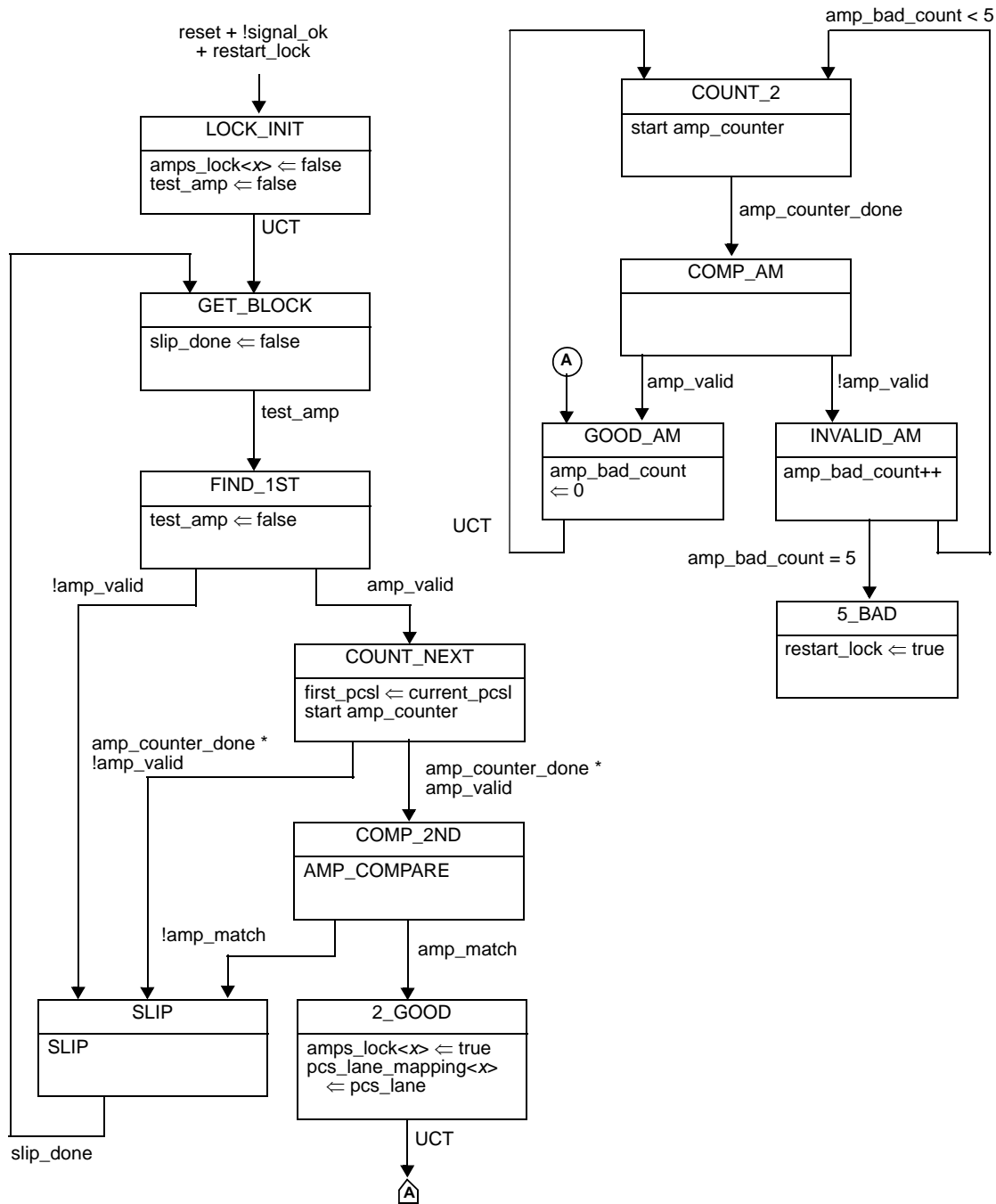


Figure 119–12—Alignment marker lock state diagram

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54

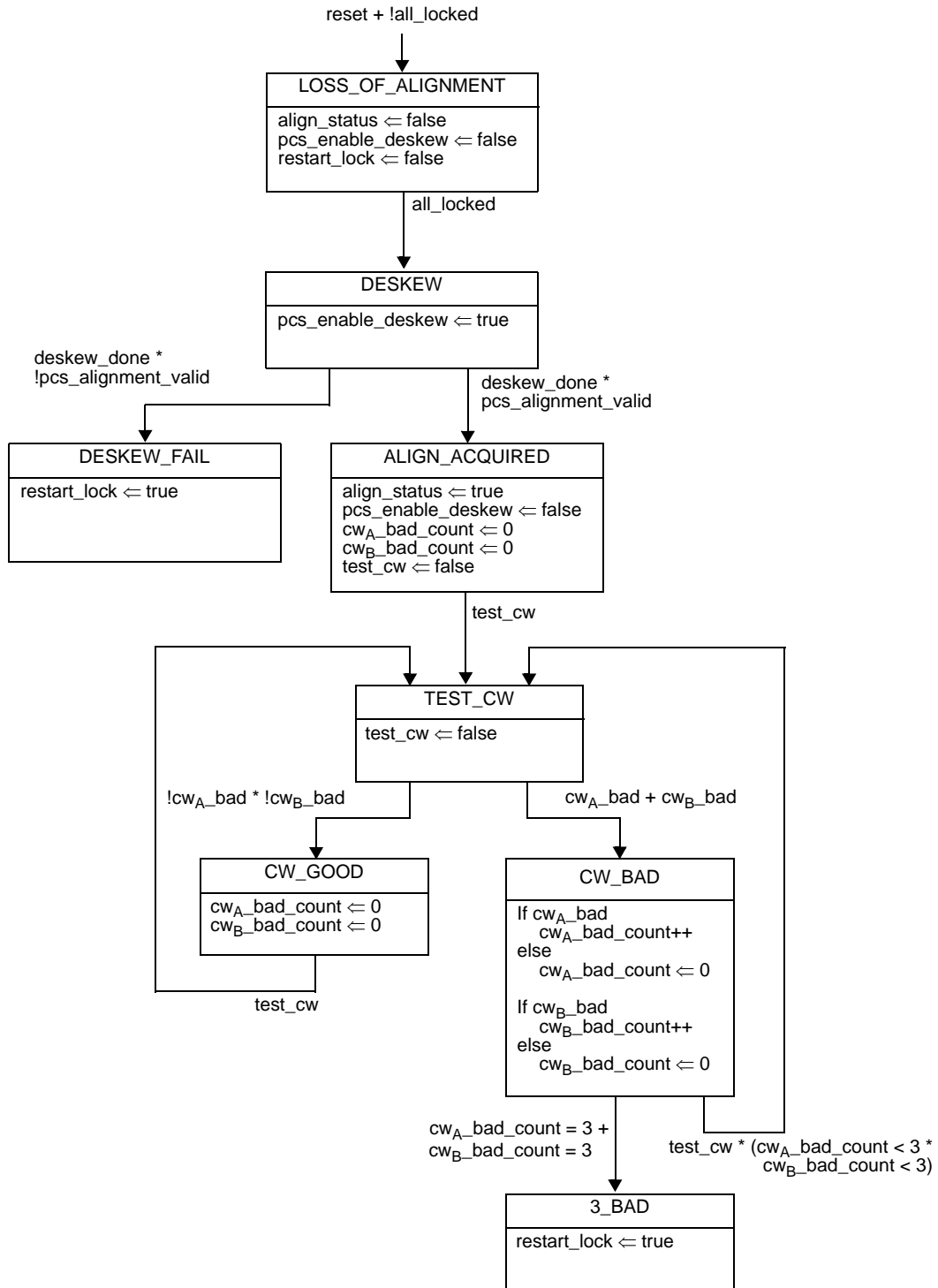
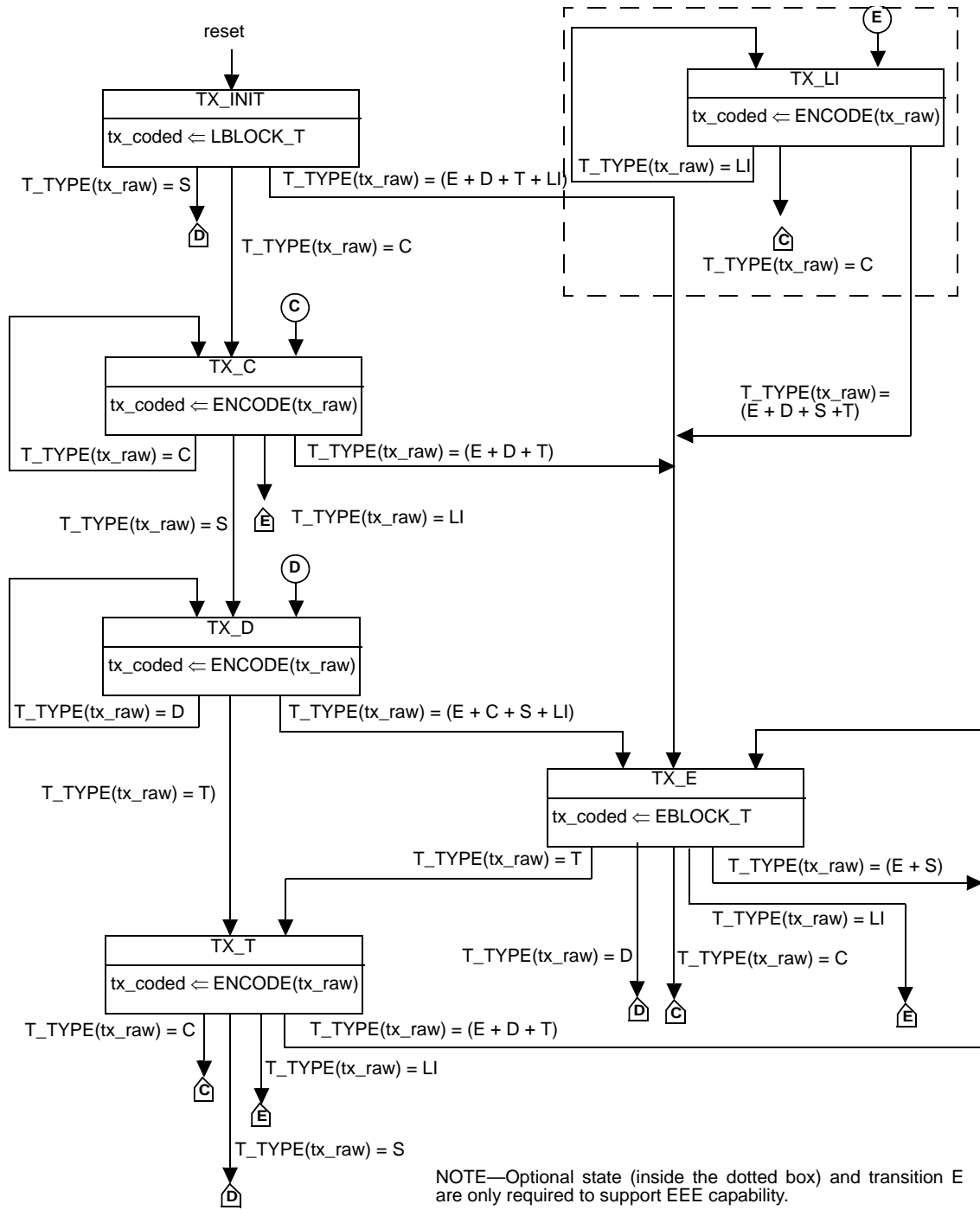


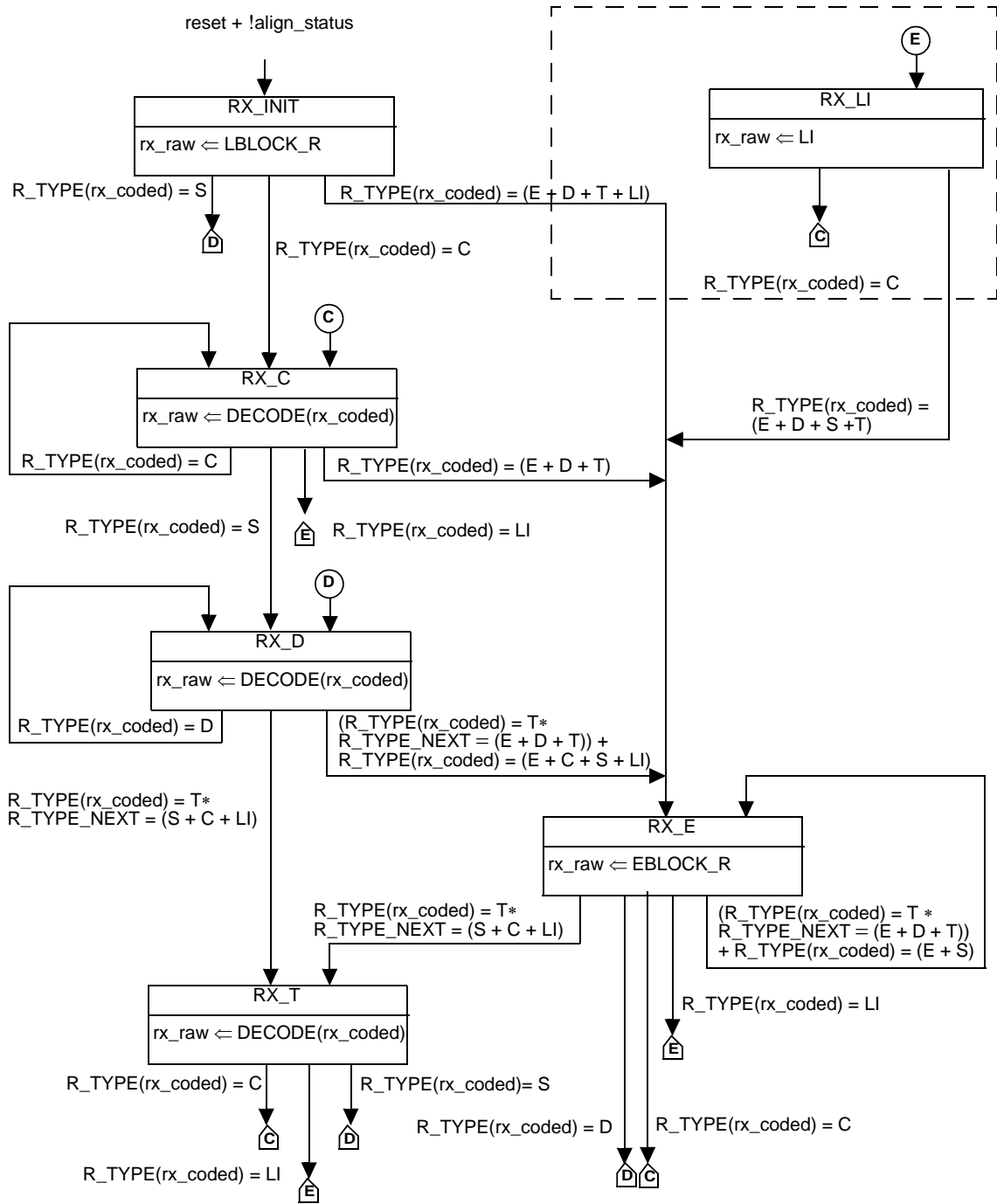
Figure 119-13—PCS synchronization state diagram

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54



**Figure 119–14—Transmit state diagram**

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54



NOTE—Optional state (inside the dotted box) and transition E are only required to support EEE capability.

Figure 119-15—Receive state diagram

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54



## 119.3 PCS management

The following objects apply to PCS management. If an MDIO Interface is provided (see Clause 45), they are accessed via that interface. If not, it is recommended that an equivalent access is provided.

### 119.3.1 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 119-4, and MDIO status bits to PCS status variables as shown in Table 119-5.

**Table 119-4—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
LPI_FW	EEE control and capability	3.20.0	LPI_FW
PCS FEC bypass indication enable	PCS FEC control register	3.800.1	FEC_bypass_indication_enable
PCS FEC degraded SER enable	PCS FEC control register	3.800.2	FEC_degraded_SER_enable
PCS FEC degraded SER activate threshold	PCS FEC degraded SER activate threshold register	3.806, 3.807	FEC_degraded_SER_activate_threshold
PCS FEC degraded SER deactivate threshold	PCS FEC degraded SER deactivate threshold register	3.808, 3.809	FEC_degraded_SER_deactivate_threshold
PCS FEC degraded SER interval	PCS FEC degraded SER interval	3.810, 3.811	FEC_degraded_SER_interval

**Table 119-5—MDIO/PCS status variable mapping**

MDIO status variable	PCS register name	Register/ bit number	PCS status variable
BASE-R and MultiGBASE-T receive link status	BASE-R and MultiGBASE-T PCS status 1 register	3.32.12	PCS_status
Lane <i>x</i> aligned	Multi-lane BASE-R PCS alignment status 3 and 4 registers	3.52.7:0 3.53.7:0	am_lock< <i>x</i> >
PCS lane alignment status	Multi-lane BASE-R PCS alignment status 1 register	3.50.12	align_status
Lane <i>x</i> mapping	Lane <i>x</i> mapping register	3.400 through 3.415	pcs_lane_mapping< <i>x</i> >
PCS FEC bypass indication ability	PCS FEC status register	3.801.1	FEC_bypass_indication_ability
PCS FEC corrected codewords	PCS FEC corrected codewords counter register	3.802, 3.803	FEC_corrected_cw_counter
PCS FEC uncorrected codewords	PCS FEC uncorrected codewords counter register	3.804, 3.805	FEC_uncorrected_cw_counter
PCS FEC symbol errors, PCS lanes 0 to <i>x</i>	PCS FEC symbol error counter register, lanes 0 to <i>x</i>	3.600 to 3.631	FEC_symbol_error_counter_ <i>i</i>
Tx LPI indication	PCS status 1	3.1.9	Tx LPI indication
Tx LPI received	PCS status 1	3.1.11	Tx LPI received
Rx LPI indication	PCS status 1	3.1.8	Rx LPI indication
Rx LPI received	PCS status 1	3.1.10	Rx LPI received
EEE wake error counter	EEE wake error counter	3.22	Wake_error_counter
PCS FEC degraded SER ability	PCS FEC status register	3.801.3	FEC_degraded_SER_ability
PCS FEC degraded SER	PCS FEC status register	3.801.4	FEC_degraded_SER
Remote degraded SER received	PCS FEC status register	3.801.5	rx_rm_degraded
PCS FEC high SER	PCS FEC status register	3.801.2	hi_ser

## 119.4 Loopback

When the PCS is in loopback, the PCS shall accept data on the transmit path from the 200GMII/400GMII and return it on the receive path to the 200GMII/400GMII. In addition, the PCS shall transmit what it receives from the 200GMII/400GMII to the PMA sublayer, and shall ignore all data presented to it by the PMA sublayer. If a Clause 45 MDIO is implemented, then the PCS is placed in the loopback when the loopback bit from the PCS control 1 register (bit 3.0.14) is set to a one.

## 119.5 Delay constraints

The maximum delay contributed by the 200GBASE-R PCS (sum of transmit and receive delays at one end of the link) shall be no more than 160 256 BT (313 pause\_quanta or 801.28 ns). 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 320 000 BT (625 pause\_quanta or 800 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.

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54

## 119.6 Protocol implementation conformance statement (PICS) proforma for Clause 119, Physical Coding Sublayer (PCS) for 64B/66B, type 200GBASE-R and 400GBASE-R<sup>4</sup>

### 119.6.1 Introduction

The supplier of a protocol implementation that is claimed to conform to Clause 119, Physical Coding Sublayer (PCS) for 64B/66B, type 200GBASE-R and 400GBASE-R, 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 <sup>1</sup>	
Contact point for inquiries about the PICS <sup>1</sup>	
Implementation Name(s) and Version(s) <sup>1,3</sup>	
Other information necessary for full identification—e.g., name(s) and version(s) for machines and/or operating systems; System Name(s) <sup>2</sup>	
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) 64B/66B, type 200GBASE-R and 400GBASE-R
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 <a href="#">Clause 21</a> ; the answer Yes means that the implementation does not conform to IEEE Std 802.3bs-201x.)	

Date of Statement	
-------------------	--

<sup>4</sup>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.

### 119.6.3 Major capabilities/options

Item	Feature	Subclause	Value/Comment	Status	Support
MII	200GMII or 400GMII logical interface	117, 119.1.4.1	Logical interface is supported	O	Yes [ ] No [ ]
*PCS200	PCS for 200GBASE-R	119.1.1		O.1	Yes [ ] No [ ]
*PCS400	PCS for 400GBASE-R	119.1.1		O.1	Yes [ ] No [ ]
*MD	MDIO	45, 119.3	Registers and interface supported	O	Yes [ ] No [ ]
*BI	Bypass indication	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	119.2.1		M	Yes [ ]
FDD	Support for optional FEC degraded detection	119.2.5.3	In the FEC decoder can optionally detect FEC SER degraded at a programmable threshold	O	Yes [ ] No [ ]

### 119.6.4 PICS proforma tables for Physical Coding Sublayer (PCS) 64B/66B, type 200GBASE-R and 400GBASE-R

#### 119.6.4.1 Transmit function

Item	Feature	Subclause	Value/Comment	Status	Support
TF1	64B/66B to 256B/257B transcoder	119.2.4.2	tx_xcoded<256:0> constructed per 119.2.4.2	M	Yes [ ]
TF2	Transmission bit ordering	119.2.4.9	First bit transmitted is bit 0	M	Yes [ ]
TF3	Pad value	119.2.4.4	PRBS9	M	Yes [ ]
TF4	Alignment marker insertion	119.2.4.4		M	Yes [ ]
TF5	Pre-FEC distribution	119.2.4.5	Distribute the data to two FEC codewords	M	Yes [ ]
TF6	Reed-Solomon encoder	119.2.4.7	RS(544,514)	M	Yes [ ]
TF7	Symbol distribution	119.2.4.8	Distribution is based on 10b symbols	M	Yes [ ]

### 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	M	Yes [ ]
RF4	Reed-Solomon decoder	119.2.5.3	Capable of indicating when a codeword was not corrected.	M	Yes [ ]
RF5	Error monitoring while error indication is bypassed	119.2.5.3	When the number of symbol errors in a block of 8192 codewords exceeds 5560, corrupt 66-bit block synchronization headers	BI:M	Yes [ ] N/A [ ]
RF6	Bypass indication error marking	119.2.5.3	Synchronization headers are marked for 60 ms to 75 ms when the error threshold is reached.	BI:M	Yes [ ] N/A [ ]
RF7	256B/257B to 64B/66B transcoder	119.2.5.8	rx_coded_j<65:0>, j=0 to 3 constructed per 119.2.5.8	M	Yes [ ]

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54

### 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, 119.2.6.2.3		M	Yes [ ]
C2	Decoder (and DECODE function) implements the code as specified	119.2.3, 119.2.6.2.3		M	Yes [ ]
C3	Only valid block types are transmitted	119.2.3.2		M	Yes [ ]
C4	Invalid block types are treated as an error	119.2.3.2		M	Yes [ ]
C5	Only valid control characters are transmitted	119.2.3.3		M	Yes [ ]
C6	Invalid control characters are treated as an error	119.2.3.3		M	Yes [ ]
C7	If EEE has not been negotiated, LPI is not transmitted	119.2.3.3		EEE:M	Yes [ ] N/A [ ]
C8	If EEE has not been negotiated, LPI is treated as an error if received	119.2.3.3		EEE:M	Yes [ ] N/A [ ]
C9	Idles do not interrupt data	119.2.3.5		M	Yes [ ]
C10	IDLE control code insertion and deletion	119.2.3.5	Insertion or Deletion in groups of 8 /I/s	M	Yes [ ]
C11	Sequence ordered set deletion	119.2.3.8	Only one whole ordered set of two consecutive sequence ordered sets may be deleted	M	Yes [ ]

### 119.6.4.4 Scrambler and descrambler

Item	Feature	Subclause	Value/Comment	Status	Support
S1	58-bit scrambler	119.2.4.3	Performs as shown in Figure 49–8	M	Yes [ ]
S2	58-bit descrambler	119.2.5.7	Performs as shown in Figure 49–10	M	Yes [ ]
S3	7-bit scrambler	119.2.4.6	Performs as shown in Figure 50–9	M	Yes [ ]
S4	7-bit descrambler	119.2.5.4	Performs as shown in Figure 50–9	M	Yes [ ]

### 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 in 119.2.4.4	M	Yes [ ]
AM2	Alignment marker form	119.2.4.4	Alignment markers are formed as described in 119.2.4.4	M	Yes [ ]
AM3	Lane mapping	119.2.6.3	PCS lane number is captured	MD:M	Yes [ ] N/A [ ]
AM4	Alignment marker removal	119.2.5.6	Alignment markers are removed prior to descrambling as described in 119.2.5.6	M	Yes [ ]

### 119.6.5 Test-pattern modes

Item	Feature	Subclause	Value/Comment	Status	Support
JT1	Scrambled idle transmit test-pattern generator is implemented	119.2.4.10		M	Yes [ ]

### 119.6.6 Bit order

Item	Feature	Subclause	Value/Comment	Status	Support
B1	Transmit bit order	119.2.4.9	Placement of bits into the PCS lanes as shown in Figure 119–10 or Figure 119–11	M	Yes [ ]

### 119.6.7 Management

Item	Feature	Subclause	Value/Comment	Status	Support
M1	Alternate access to PCS management object is provided	119.3		O	Yes [ ] No [ ]
M2	Mapping of MDIO control bits and MDIO status bits	119.3.1	Table 119–4 and Table 119–5	MD:M	Yes [ ] N/A [ ]

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54



### 119.6.7.1 State diagrams

Item	Feature	Subclause	Value/Comment	Status	Support
SM1	Alignment marker lock	119.2.6	Implements 8 alignment marker lock processes as depicted in Figure 119–12	PCS200:M	Yes [ <input type="checkbox"/> N/A [ <input type="checkbox"/>
SM2	Alignment marker lock	119.2.6	Implements 16 alignment marker lock processes as depicted in Figure 119–12	PCS400:M	Yes [ <input type="checkbox"/> N/A [ <input type="checkbox"/>
SM3	The SLIP function evaluates all possible block positions	119.2.6.2.3		M	Yes [ <input type="checkbox"/>
SM4	PCS synchronization state diagram	119.2.6	Meets the requirements of Figure 119–13	M	Yes [ <input type="checkbox"/>
SM5	Transmit process	119.2.6	Meets the requirements of Figure 119–14	M	Yes [ <input type="checkbox"/>
SM6	Receive process	119.2.6	Meets the requirements of Figure 119–15	M	Yes [ <input type="checkbox"/>

### 119.6.7.2 Loopback

Item	Feature	Subclause	Value/Comment	Status	Support
L1	Supports loopback	119.4		M	Yes [ <input type="checkbox"/>
L2	When in loopback, transmits what it receives from the 200GMII/400GMII	119.4		M	Yes [ <input type="checkbox"/>
L3	When in loopback, ignore all data presented by the PMA sublayer.	119.4		M	Yes [ <input type="checkbox"/>

### 119.6.7.3 Delay constraints

Item	Feature	Subclause	Value/Comment	Status	Support
TIM1	PCS delay constraint	119.5	No more than 160 256 BT for sum of transmit and receive path delays for 200GBASE-R.	PCS200: M	Yes [ <input type="checkbox"/> N/A [ <input type="checkbox"/>
TIM2	PCS delay constraint	119.5	No more than 320 000 BT for sum of transmit and receive path delays for 400GBASE-R.	PCS400: M	Yes [ <input type="checkbox"/> N/A [ <input type="checkbox"/>

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54