Instructions to the Editor :

This document uses clause 49 numbering. This should make it easier to create the new clause based on existing clause 49 text. Once the clause number for 10GBASE-KR PCS is known the numbering should be corrected accordingly.

49. Physical Coding Sublayer (PCS) for 64B/66B, type 10GBASE-KR

49.1 Overview

49.1.1 Scope

This clause specifies the Physical Coding Sublayer (PCS) for Ethernet operation over electrical backplanes ~~that is common to a family of 10 Gb/s Physical Layer implementations~~, known as 10GBASE-KR. This PCS ~~can~~ connects directly to ~~one of~~ the 10GBASE-KR Physical Layers~~.:~~ ~~10GBASE-SR, 10GBASE-LR, and 10GBASE-ER. Alternatively, this PCS can connect to a Wan Interface Sublayer (WIS), which will produce the 10GBASE-W encoding (10GBASE-R encoded data stream encapsulated into frames compatible with SONET and SDH networks) for transport by the 10GBASE-W Physical Layers: 10GBASE-SW, 10GBASE-LW, and 10GBASE-EW. The term 10GBASE-KR is used when referring generally to physical layers using the PCS defined here.~~

The 10GBASE-KR is based on a 64B/66B code. The 64B/66B code supports data and control characters, while maintaining robust error detection. The 10GBASE-KR adds additional error protection to frames to support use over electrical backplanes.

10GBASE-KR PCS maps the interface characteristics of ~~the WIS when present, and~~ the PMA sublayer to the services expected by the Reconciliation and XGXS sublayers. 10GBASE-KR can be extended to support any full duplex medium requiring only that the medium be compliant at the PMA level. 10GBASE-KR PCS may be attached through the PMA sublayer to a ~~LAN~~ 10GBASE-KR PMD sublayer supporting a data rate of 10 Gb/s ~~or it may be attached to a WAN PMD through the WIS and PMA sublayers~~. ~~When attached to a WAN sublayer, this PCS adapts the data stream to the WAN data rate.~~

49.1.2 Objectives

The following are the objectives of 10GBASE- KR:
a) Support the full duplex Ethernet MAC.
b) Provide 10 Gb/s data rate at the XGMII.
c) Support ~~LAN~~ the 10GBASE-KR PMD~~s~~ operating at 10 Gb/s ~~and WAN PMDs operating at SONET STS-192c/SDH VC-4-64c rate~~.
d) ~~Support cable plants using optical .ber compliant with ISO/IEC 11801: 1995 as specified in Clause 52.~~ Provide additional error protection for frames transferred across electrical backplanes.

e) ~~Allow for a nominal network extent of up to 40 km.~~
~~f)~~ Support a BER objective of 10^{-12}.

49.1.3 Relationship of 10GBASE-KR to other standards

Figure 49–1 depicts the relationships among the 10GBASE-KR sublayers (shown shaded), the Ethernet MAC and reconciliation layers, and the higher layers.

49.1.4 Summary of 10GBASE-KR ~~and 10GBASE-W~~ sublayer~~s~~

The following subclauses provide an overview of the 10GBASE-KR ~~and 10GBASE-W~~ sublayer~~s~~. Figure 49–1 depicts the relationship between the 10GBASE-KR PCS and its associated sublayers.

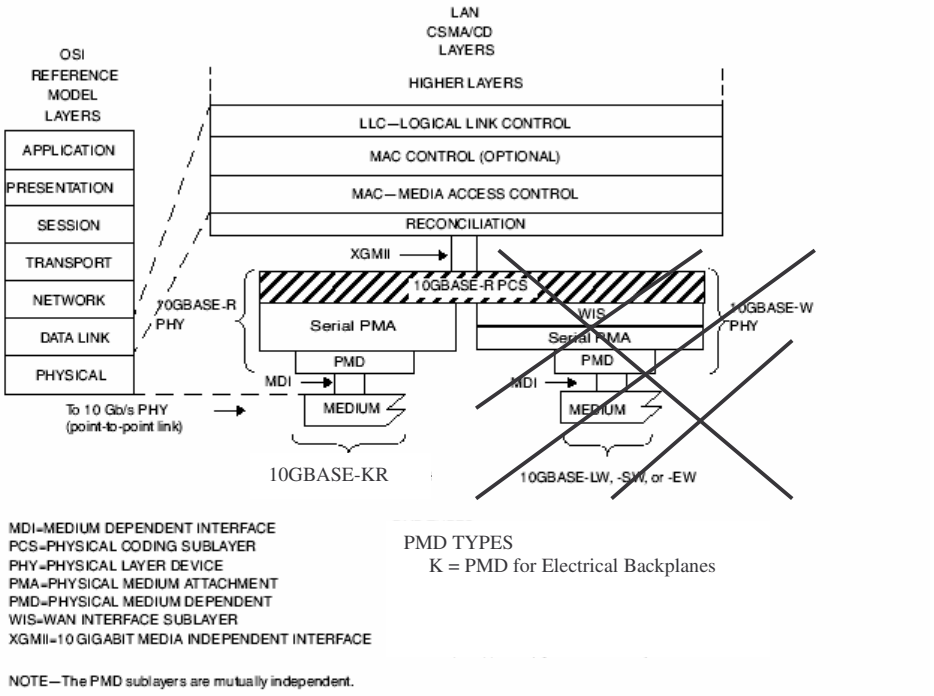

Figure 49–1—10GBASE-R PCS relationship to the ISO/IEC Open Systems Interconnection (OSI) reference model and IEEE 802.3 CSMA/CD LAN model

49.1.4.1 Physical Coding Sublayer (PCS)

The PCS service interface is the 10 Gigabit Media Independent Interface (XGMII), which is defined in Clause 46. The XGMII provides a uniform interface to the Reconciliation Sublayer for all 10 Gb/s PHY implementations (e.g., not only 10GBASE-KR but also other types of 10 Gigabit PHY entities).

The 10GBASE-KR PCS provides all services required by the XGMII, including the following:
a) Encoding (decoding) of eight XGMII data octets to (from) 66-bit blocks (64B/66B).
b) Transferring encoded data to (from) the PMA in 16 bit transfers.
c) ~~When connected to a WAN PMD, deleting (inserting) idles to compensate for the rate differencebetween the MAC and PMD.~~Adding, checking, and removing an additional CRC8 error protection field to all frames
d) Determining when a functional link has been established and informing the management entity via the MDIO when the PHY is ready for use.

~~49.1.4.2 WAN Interface Sublayer (WIS)~~

~~The WIS provides a medium independent means for the PCS to operate over WAN links. It creates a10GBASE-W encoding by encapsulating the encoded data steam from the 10GBASE-R PCS in frames compatible with SONET and SDH transmission formats. The WIS is specified in Clause 50.~~

49.1.4.3 Physical Medium Attachment (PMA) sublayer

The PMA provides a medium-independent means for the PCS to support the use of a range of physical media. The 10GBASE-KR PMA performs the following functions:
a) Mapping of transmit and receive data streams between the PCS ~~or WIS~~ and PMA via the PMA service interface.
b) Serialization (deserialization) of bits for transmission (reception) on the underlying serial PMD.
c) Recovery of clock from the received data stream.
d) Mapping of transmit and receive bits between the PMA and PMD via the PMD service interface.
e) Optionally provides data loopback at the PMA service interface.

The PMA is specified in Clause 51.

49.1.4.4 Physical Medium Dependent (PMD) sublayer

The PMD and its media are specified in Clause 72~~52~~.
The MDI, logically subsumed within each PMD subclause, is the actual medium attachment for the various supported media.

49.1.4.5 Bit ordering across 10GBASE-KR ~~and 10GBASE-W~~ sublayers.

The ordering of bits and octets ~~for the case when no WIS is present~~ between the PCS and PMA is shown in Figure 49–2. The diagram depicts the bit mappings for the data path at the service interfaces.

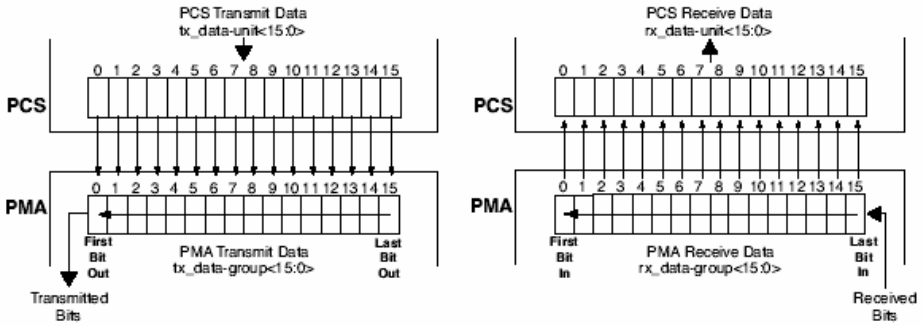

Figure 49–2—10GBASE-R PHY Transmission order

~~The bit/octet ordering when a WIS is interposed between the PCS and PMA is shown in Figure 49–3. The details of WIS frame generation, as well as internal interfaces, have been omitted. Note that since SONET convention is to send the most signi.cant bit (MSB) of each octet .rst and Ethernet convention is to send the least signi.cant bit (LSB) of each octet .rst, the LSBs from the PCS octets are mapped into the MSBs of the WIS octets. It should be clear from the diagram that the Ethernet order of transmission (i.e., least signi.cant bit .rst), and hence the error detecting properties of the Ethernet CRC, are preserved in spite of the different bit ordering conventions followed by the WIS. Also, the SONET/SDH bit labeling conventions are different from the usual IEEE 802.3 bit labeling. The bits of a SONET/SDH octet are labeled from 1 to 8 with bit 1 being the MSB. Ethernet conventions label bits of an n bit .eld from 0 to n–1 with bit 0 being the LSB. Figure 49–3 shows the results of these conventions. For example, tx_data unit<0> through tx_data unit<7> map to bits 1 through 8 respectively of a WIS octet.~~

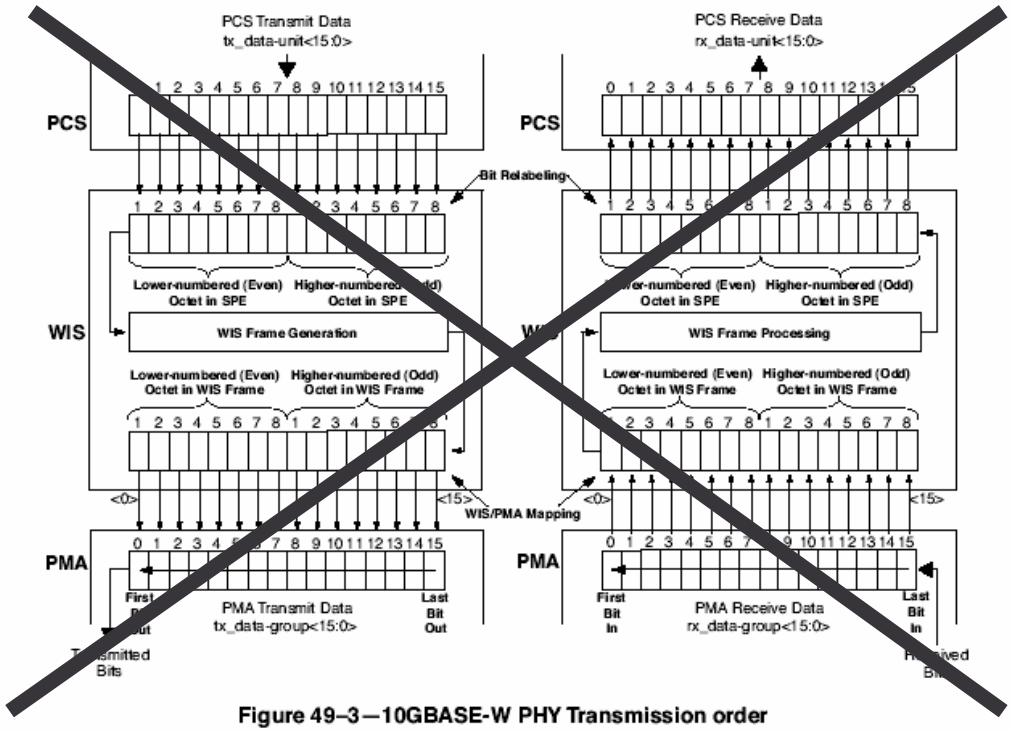

Figure 49–3—10GBASE-W PHY Transmission order

49.1.5 Inter-sublayer interfaces

There are a number of interfaces employed by 10GBASE-KR. Some (such as the PMA service interface) use an abstract service model to define the operation of the interface. The PCS service interface is the XGMII that is defined in Clause 46. The XGMII has an optional physical instantiation. An optional physical instantiation of the PMA service interface has also been defined (see Clause 51). It is called XSBI (10 Gigabit Sixteen Bit Interface). Figure 49–4 depicts the relationship and mapping of the services provided by all of the interfaces relevant to 10GBASE-KR.

The upper interface of the PCS may connect to the Reconciliation Sublayer through the XGMII or the PCS may connect to an XGXS sublayer. The XGXS and the Reconciliation Sublayer provide the same service interface to the PCS. The lower interface of the PCS ~~may connect to the WIS to support a WAN PMD or~~ connects to the PMA sublayer to support a 10GBASE-KR ~~LAN~~ PMD. ~~The WIS and PMA interfaces are functionally equivalent except for data rate. When the PCS is connected directly to a LAN PMA, the nominal rate of the PMA service interface is 644.53 Mtransfers/s which provides capacity for the MAC data rate of 10 Gb/s. When the PCS is connected to a WAN PMA, the nominal rate of the WIS service interface is 599.04 Mtransfers/s and the MAC uses IFS stretch mode to ensure that there will be enough idle time that the PCS can delete idles to adjust to the lower rate. Since the data rates are different, WIS and PMA interface connections pose somewhat different constraints. The PCS shall support connection to either a WIS or to a PMA and may optionally support both.~~

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.

49.1.6 Functional block diagram

Figure 49–4 provides a functional block diagram of the 10GBASE-KR PHY.

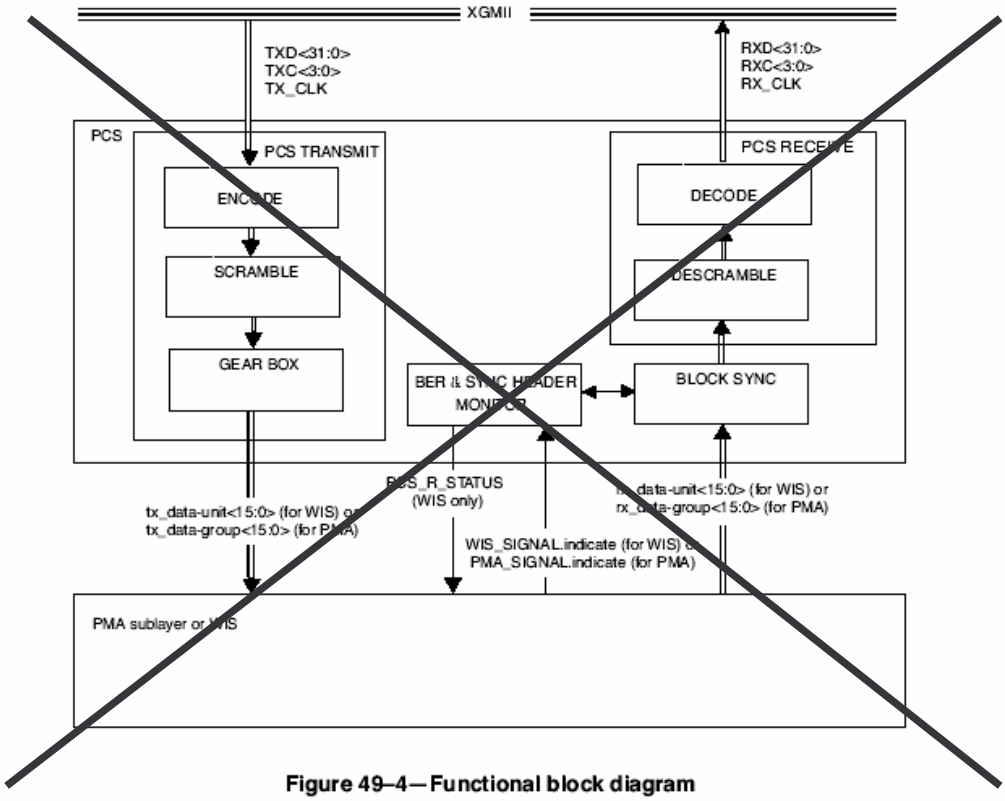

Figure 49–4—Functional block diagram

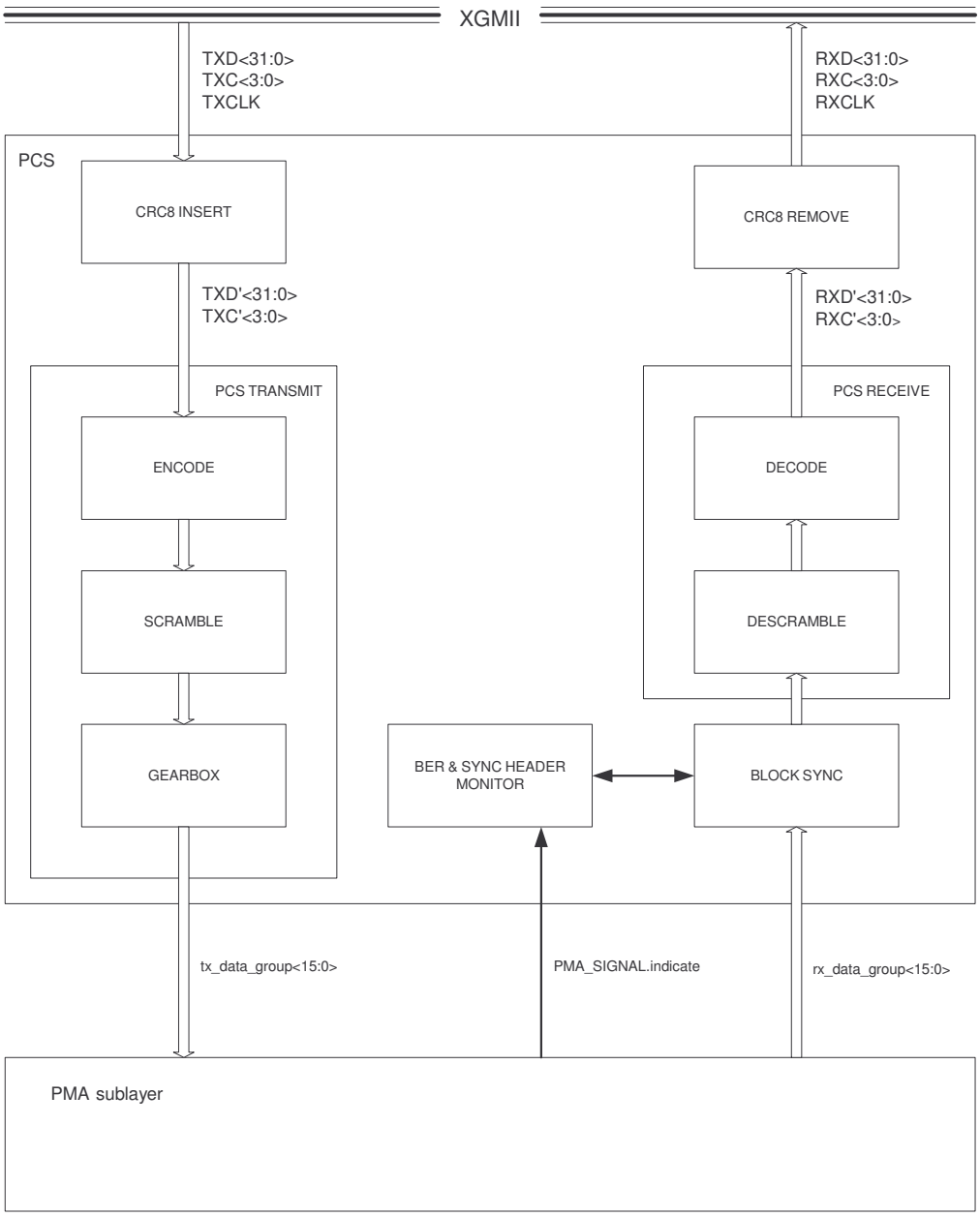

Figure 49-4 Functional block diagram

49.2 Physical Coding Sublayer (PCS)

49.2.1 PCS service interface (XGMII)

The PCS service interface allows the 10GBASE-KR PCS to transfer information to and from a PCS client. A PCS client is generally the Reconciliation Sublayer or an XGXS sublayer. The PCS Interface is precisely defined as the 10 Gigabit Media Independent Interface (XGMII) in Clause 46.

49.2.2 Functions within the PCS

The PCS comprises the CRC8 Insert, PCS Transmit, Block Synchronization, CRC8 Remove, PCS Receive, and BER monitor processes for 10GBASE-KR. 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~~, when not attached to a WIS,~~ test-pattern mode. ~~When the PCS is attached to a WIS, the WIS provides the test-pattern functionality~~.

When communicating with the XGMII, the PCS uses a four octet-wide, synchronous data path, with packet delimiting being provided by transmit control signals (TXCn = 1) and receive control signals (RXCn = 1). When communicating with the PMA ~~or WIS~~, the PCS uses a 16-bit wide, synchronous data path that conveys 16 encoded bits. Alignment to 64B/66B block is performed in the PCS. The ~~WIS and~~ PMA sublayer~~s~~ operate~~s~~ independent of block and packet boundaries. The PCS provides the functions necessary to map packets between the XGMII format and the PMA service interface format.

When the transmit channel is in normal mode, the CRC8 Insert process continuously processes XGMII signals TXD<31:0> and TXC<3:0>, adding an additional CRC8 field to all frames. The PCS Transmit process continuously generates blocks based upon the modified XGMII signals TXD'<31:0> and TXC'<3:0> ~~signals on the XGMII~~from the CRC8 Insert process. The Gearbox function of the PCS Transmit process then packs the resulting bits into 16-bit transmit data-units. Transmit data-units are sent to the PMA ~~or WIS~~ service interface via the PMA_UNITDATA.request ~~or WIS_UNITDATA~~.request primitive~~, respectively~~. ~~When the WIS is present, the PCS Transmit process also adapts between the XGMII and WIS data rates by deleting idle characters.~~

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_UNITDATA.request primitive.

When the receive channel is in normal mode, the PCS Synchronization process continuously monitors PMA_SIGNAL.indicate(SIGNAL_OK) ~~or WIS_SIGNAL.indicate(SIGNAL_OK).~~ When SIGNAL_OK indicates OK, then the PCS Synchronization process accepts data-units via the PMA_UNITDATA.indicate primitive ~~or the WIS_UNITDATA.indicate primitive~~. It attains block sychronization based on the 2-bit synchronization headers and conveys received blocks to the PCS Receive process. The PCS Synchronization process sets the sync_status flag to indicate whether the PCS has obtained synchronization.

When the PCS Synchronization process has obtained synchronization, the BER monitor process monitors the signal quality asserting hi_ber if excessive errors are detected. When sync_status is asserted and hi_ber is de-asserted, the PCS Receive process continuously accepts blocks. The PCS Receive process monitors these blocks and generates RXD'_<31:0> and RXC'<3:0> ~~on the XGMII~~ signals as input to the CRC8 Remove process. The CRC8 Remove process continuously processes signals RXD' <31:0> and RXC'<3:0>, recognizing frames, checking and removing the additional CRC8 field in the frames, and generating the XGMII signals RXD<31:0> and RXC<3:0>. ~~When the WIS is present, the PCS Receive process also adapts between the WIS and XGMII data rates by inserting idle characters. The PCS Receive process also sends the WIS_SIGNAL.request(PCS_R_STATUS) primitive to the WIS to indicate its status. The primitive is sent when the value of PCS_R_STATUS changes. The value of PCS_R_STATUS shall be FAIL when the Receive state machine is in the RX_INIT state and OK otherwise.~~

When the receive channel is in test-pattern mode, the BER monitor process is 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.

A PCS that supports direct connection to a PMA shall provide transmit test-pattern mode for the square wave and pseudo-random patterns, and shall provide receive test-pattern mode for the pseudo-random pattern. It may provide support for the PRBS31 test pattern. Support of the optional PRBS31 test pattern shall include both the transmit and the receive capability for that pattern. Test-pattern mode is activated separately for transmit and receive. A PCS that supports direct connection to a PMA shall support transmit test-pattern mode and receive test-pattern mode operating simultaneously so as to support loopback testing. ~~Test-pattern mode is provided by the WIS when a WIS is present.~~

49.2.3 Use of blocks

The PCS maps XGMII signals into 66-bit blocks, and vice versa, using a 64B/66B coding scheme. The synchronization headers of the blocks allow establishment of block boundaries by the PCS Synchronization process. Blocks are unobservable and have no meaning outside the PCS. The PCS functions ENCODE and DECODE generate, manipulate, and interpret blocks as provided by the rules in 49.2.4.

49.2.4 64B/66B transmission code

The PCS uses a transmission code to improve the transmission characteristics of information to be transferred across the link and to support transmission of control and data characters. The encodings defined by the transmission code ensure that sufficient transitions are present in the PHY bit stream to make clock recovery possible at the receiver. The encoding also preserves the likelihood of detecting any single or multiple bit errors that may occur during transmission and reception of information. In addition, the synchronization headers of the code enable the receiver to achieve block alignment on the incoming PHY bit stream. The 64B/66B transmission code specified for use in this standard has a high transition density and is a runlength-limited code.

The relationship of block bit positions to XGMII, PMA, and other PCS constructs is illustrated in Figure 49–5 for transmit and Figure 49–6 for receive. These figures illustrate the processing of a block containing 8 data octets. See 49.2.4.3 for information on how blocks containing control characters are mapped. Note that the sync header is generated by the encoder and bypasses the scrambler.

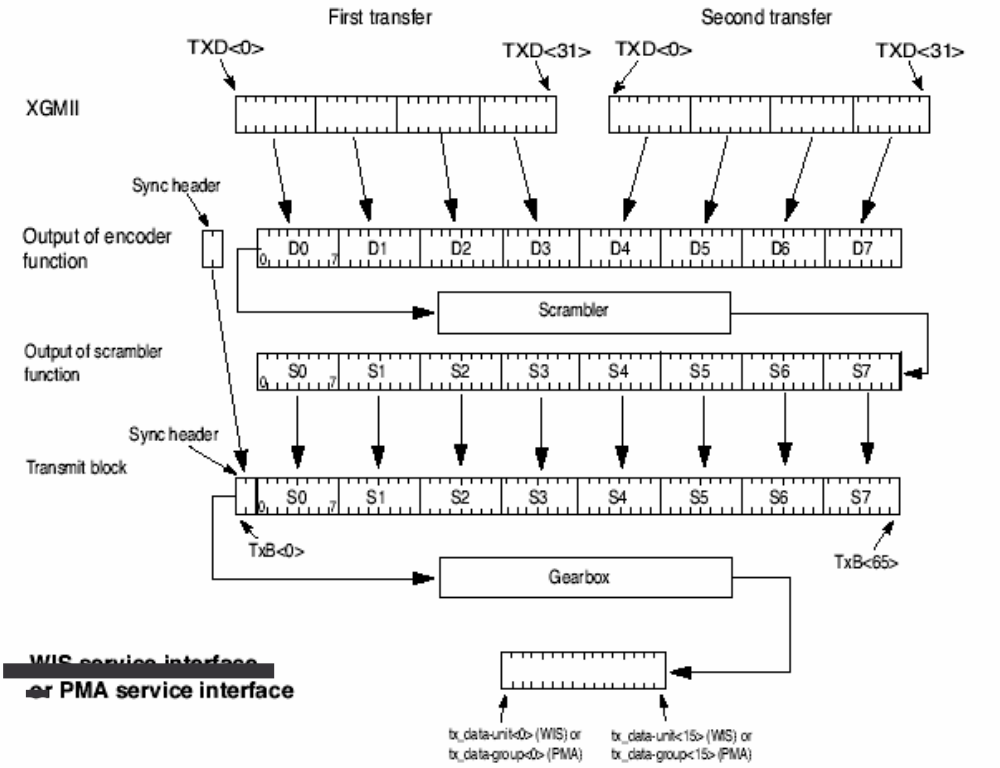

Figure 49–5—PCS Transmit Bit Ordering

49.2.4.1 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 D0 to D_7. Control characters other than /O/, /S/

and /T/ are labeled C_0 to C_7. The control character for ordered_set is labeled as O_0 or O_4 since it is only valid on the first octet of the XGMII. The control character for start is labeled as S_0 or S_4 for the same reason. The control character for terminate is labeled as T_0 to T_7.

Two consecutive XGMII transfers provide 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 XGMII transfers. 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 .rst transmitted bit (the LSB) on the left.

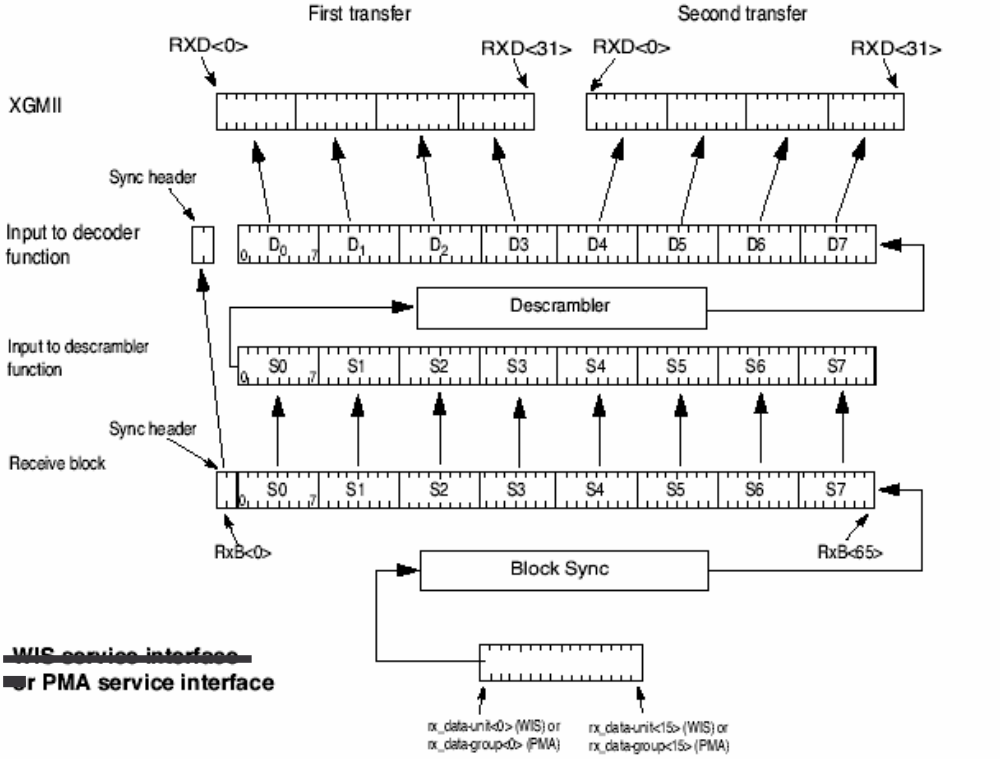

Figure 49–6—PCS Receive Bit Ordering

49.2.4.2 Transmission order

Block bit transmission order is illustrated in Figure 49–5 and Figure 49–6. Note that these figures show the mapping from XGMII to 64B/66B block for a block containing eight data characters.

49.2.4.3 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. The payload is scrambled and the sync header bypasses the scrambler. Therefore, the sync header is the only position in the block that always contains a transition. This feature of the code is used to obtain block synchronization.

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 or Terminate character, that character is implied by the block type field. Other control characters are encoded in a 7-bit control code or a 4-bit O Code. Each control block contains eight characters.

The format of the blocks is as shown in Figure 49–7. 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 XGMII interface, 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 octets 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.

Bits and field positions are shown with the least significant bit on the left. Hexadecimal numbers are shown in normal hexadecimal. For example the block type field 0x1e is sent as 01111000 representing bits 2 through 9 of the 66 bit block. The least significant bit for each field is placed in the lowest numbered position of the field. All unused values of block type field are reserved.

49.2.4.4 Control codes

The same set of control characters are supported by the XGMII and the 10GBASE-KR PCS. The representations of the control characters are the control codes. XGMII encodes a control character into an octet (an eight bit value). The 10GBASE-KR PCS encodes the start and terminate control characters implicitly by the block type field. The 10GBASE-KR PCS encodes the ordered_set control codes using a combination of the block type field and a 4-bit O code for each ordered_set. The 10GBASE-KR PCS encodes each of the other control characters into a 7-bit C code).

The control characters and their mappings to 10GBASE-KR control codes and XGMII control codes are specified in Table 49–1. All XGMII and 10GBASE-KR control code values that do not appear in the table shall not be transmitted and shall be treated as 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
$C_0 C_1 C_2 C_3/O_4 D_5 D_6 D_7$	10	0x2d	C_0	C_1	C_2	C_3	O_4	D_5	D_6	D_7
$C_0 C_1 C_2 C_3/S_4 D_5 D_6 D_7$	10	0x33	C_0	C_1	C_2	C_3		D_5	D_6	D_7
$O_0 D_1 D_2 D_3/S_4 D_5 D_6 D_7$	10	0x66	D_1	D_2	D_3	O_0		D_5	D_6	D_7
$O_0 D_1 D_2 D_3/O_4 D_5 D_6 D_7$	10	0x55	D_1	D_2	D_3	O_0	O_4	D_5	D_6	D_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
$O_0 D_1 D_2 D_3/C_4 C_5 C_6 C_7$	10	0x4b	D_1	D_2	D_3	O_0	C_4	C_5	C_6	C_7
$T_0 C_1 C_2 C_3/C_4 C_5 C_6 C_7$	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 49–7—64B/66B Block Formats

49.2.4.5 Ordered sets

Ordered sets are used to extend the ability to send control and status information over the link such as remote fault and local fault status. Ordered sets consist of a control character followed by three data characters. Ordered sets always begin on the first octet of the XGMII. 10 Gigabit Ethernet uses one kind of ordered_set: the sequence ordered_set (see 46.3.4). The sequence ordered_set control character is denoted /Q/. An additional ordered_set, the signal ordered_set, has been reserved and it begins with another control code. The 4-bit O field encodes the control code. See Table 49–1 for the mappings.

49.2.4.6 Valid and invalid blocks

A block is invalid if any of the following conditions exists:
a) The sync field has a value of 00 or 11.
b) The block type field contains a reserved value.
c) Any control character contains a value not in Table 49–1.
d) Any O code contains a value not in Table 49–1.
e) The set of eight XGMII characters does not have a corresponding block format in Figure 49–7.

Table 49–1—Control Codes

Control Character	Notation	XGMII Control Code	10GBASE-R Control Code	10GBASE-R O Code	8B/10B Code[a]
idle	/I/	0x07	0x00		K28.0 or K28.3 or K28.5
start	/S/	0xfb	Encoded by block type field		K27.7
terminate	/T/	0xfd	Encoded by block type field		K29.7
error	/E/	0xfe	0x1e		K30.7
Sequence ordered_set	/Q/	0x9c	Encoded by block type field plus O code	0x0	K28.4
reserved0	/R/[b]	0x1c	0x2d		K28.0
reserved1		0x3c	0x33		K28.1
reserved2	/A/	0x7c	0x4b		K28.3
reserved3	/K/	0xbc	0x55		K28.5
reserved4		0xdc	0x66		K28.6
reserved5		0xf7	0x78		K23.7
Signal ordered_set[c]	/Fsig/	0x5c	Encoded by block type field plus O code	0xF	K28.2

[a]For information only. The 8B/10B code is specified in Clause 36. Usage of the 8B/10B code for 10 Gb/s operation is specified in Clause 48.
[b]The codes for /A/, /K/, and /R/ are used on the XAUI interface to signal idle. They are not present on the XGMII when no errors have occurred, but certain bit errors cause the XGXS to send them on the XGMII.
[c]Reserved for INCITS T11 Fibre Channel use.

49.2.4.7 Idle (/I/)

Idle control characters (/I/) are transmitted when idle control characters are received from the XGMII. Idle characters may be added or deleted by the PCS to adapt between clock rates. /I/ insertion and deletion shall occur in groups of 4. /I/s may be added following idle or ordered sets. They shall not be added while data is being received. When deleting /I/s, the first four characters after a /T/ shall not be deleted.

49.2.4.8 Start (/S/)

The start control character (/S/) indicates the start of a packet. This delimiter is only valid on the first octet of the XGMII (TXD<0:7> and RXD<0:7>). Receipt of an /S/ on any other octet of TxD indicates an error. Block type field values implicitly encode an /S/ as the fifth or first character of the block. These are the only characters of a block on which a start can occur.

49.2.4.9 Terminate (/T/)

The terminate control character (/T/) indicates the end of a packet. Since packets may be any length, the /T/ can occur on any octet of the XGMII interface and within any character of the block. The location of the /T/ in the block is implicitly encoded in the block type field. A valid end of packet occurs when a block containing a /T/ is followed by a control block that does not contain a /T/.

49.2.4.10 ordered_set (/O/)

The ordered_set control characters (/O/) indicate the start of an ordered_set. There are two kinds of ordered sets: the sequence ordered_set and the signal ordered_set (which is reserved). When it is necessary to designate the control character for the sequence ordered_set specifically, /Q/ will be used. /O/ is only valid on the first octet of the XGMII. Receipt of an /O/ on any other octet of TXD indicates an error. Block type field values implicitly encode an /O/ as the first or fifth character of the block. The 4-bit O code encodes the specific /O/ character for the ordered_set.

Sequence ordered_sets may be deleted by the PCS to adapt between clock rates. Such deletion shall only occur when two consecutive sequence ordered sets have been received and shall delete only one of the two. Only Idles may be inserted for clock compensation. Signal ordered_sets are not deleted for clock compensation.

49.2.4.11 Error (/E/)

The /E/ is sent whenever an /E/ is received. It is also sent when invalid blocks are received. The /E/ allows physical sublayers such as the XGXS and PCS to propagate received errors. See R_BLOCK_TYPE and T_BLOCK_TYPE function definitions in 49.2.13.2.3 for further information.

49.2.X1 CRC8 Insert process

The CRC8 Insert process provides additional error protection to frames passed from the XGMII by adding an additional 8 CRC parity bits to each frame. This CRC covers all data characters of the frame including the CRC32 field. The CRC8 data character replaces the original /T/ character of the frame, and a new /T/ character overwrites the first IPG character.

The CRC8 Insert state machine parses XGMII signals in order to recognize frame boundaries, control CRC8 generation, and modify XGMII words. The CRC8 Insert process only modifies XGMII signals at the end of frames in order to insert the CRC8 field.

Table 49-X1 indicates how XGMII signals are modified.. In this table subscripts indicate the character position in an XGMII word. Subscript 0 indicates the first character - equivalent to TXD<7:0> and TXC<0>, and subscript 3 the last character - equivalent to TXD<31:24> and TXC<3>.

- R_0 through R_3 indicate the four characters of a (raw) input word.
- M_0 through M_3 indicate the four characters of the (modified) output word

The CRC8 Insert process generates modified XGMII signals TXD'<31:0> and TXC'<3:0> based on the TXD<31:0> and TXC<3:0> signals received from the XGMII. These modified signals are then block encoded by the PCS transmit process.

The CRC8 Insert process shall modify XGMII signals as specified in the CRC8 Insert state machine shown in Figure 49–X2.

ins_raw				ins_mod			
R_0	R_1	R_2	R_3	M_0	M_1	M_2	M_3
/T/	$\overline{/T/}$	$\overline{/T/}$	$\overline{/T/}$	CRC8	/T/	R_2	R_3
$\overline{/T/}$	/T/	$\overline{/T/}$	$\overline{/T/}$	R_0	CRC8	/T/	R_3
$\overline{/T/}$	$\overline{/T/}$	/T/	$\overline{/T/}$	R_0	R_1	CRC8	/T/
$\overline{/T/}$	$\overline{/T/}$	$\overline{/T/}$	/T/	R_0	R_1	R_2	CRC8
Else				/T/	R_1	R_2	R_3

Table 49-X1 Translation table for CRC8 insertion

49.2.X1.1 CRC8 generator

The CRC8 parity bits are generated by the following CRC8 cyclic generator polynomial:

$C(x) = 1 + x + x^5 + x^6 + x^8$

The CRC8 generated for a frame shall produce the same result as the implementation shown in Figure 49– X1. In Figure 49– X1 the 8 delay elements S0, ..., S7, shall be initialized to zero at the start of each received frame. After this all bytes of frame Data are used serially (in transmission order) to compute the CRC8 with the switch connected, which is setting CRCgen in Figure 49– X1. After all the frame data has been processed, the switch is disconnected (setting CRCout) and the 8 values stored in the delay elements are then placed in the CRC8 character for transmission in the order illustrated, first S7, followed by S6, and so on until the final value S0.

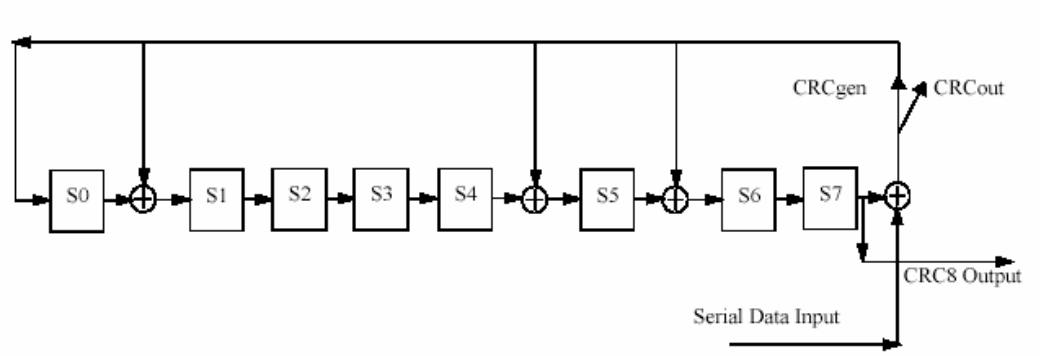

Figure 49-X1 CRC8 generator

49.2.5 Transmit process

The transmit process generates blocks based upon the TXD'<31:0> and TXC'<3:0> signals received from the CRC8 Insert process ~~XGMII~~. Two XGMII data transfers are encoded into each block. It takes 4.125 PMA_UNITDATA ~~or WIS_UNITDATA~~ transfers to send a block of data. Therefore, if the PCS is connected to an XGMII and PMA sublayer where the ratio of their transfer rates is exactly 16:33, then the transmit process does not need to perform rate adaptation. Where the XGMII and PMA sublayer data rates are not synchronized to that ratio, the transmit process will need to insert idles, delete idles, or delete sequence ordered sets to adapt between the rates. ~~The WIS data rate is always slower than the XGMII data rate and a PCS connected to a WIS must delete idles or sequence ordered sets to adapt between rates.~~

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

49.2.6 Scrambler.

The payload of the block is scrambled with a self-synchronizing scrambler. The scrambler shall produce the same result as the implementation shown in Figure 49–8. This implements the scrambler polynomial:

$G(x) = 1 + x^{39} + x^{58}$ (49–1)

There is no requirement on the initial value for the scrambler. The scrambler is run continuously on all payload bits. The sync header bits bypass the scrambler.

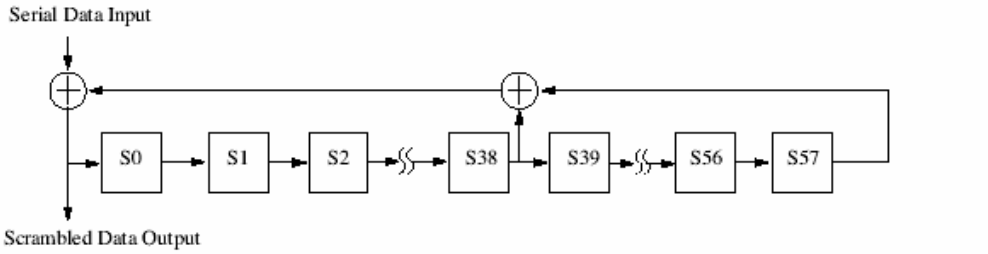

Figure 49–8—Scrambler

49.2.7 Gearbox

The gearbox adapts between the 66-bit width of the blocks and the 16-bit width of the PMA ~~or WIS~~ interface. It receives the 66-bit blocks. When the transmit channel is operating in normal mode, the gearbox sends 16 bits of transmit data at a time via ~~WIS_UNITDATA.request or~~ PMA_UNITDATA.request primitives. The UNITDATA.request primitives are fully packed with bits. For example, if one block happened to start with the sync header on bits 0 and 1 of a PMA_UNITDATA.request, then the last two bits of that block would be on bits 0 and 1 of a PMA_UNITDATA.request and the next block would begin with a sync header on bits 2 and 3 of that PMA_UNITDATA.request. When a PMA_UNITDATA.request ~~or WIS_UNITDATA.request~~ contains bits from two blocks, then the bits from the first block shall be placed in the lowest numbered bits of tx_data-group<15:0>. The bits shall be packed into the tx_data-group in sequence with the lowest numbered bit of the block going into the lowest numbered bit of the part of tx_data-group<15:0> bits containing bits from that block (see Figure 49-5).

The gearbox functionality is necessary when the optional PMA compatibility interface, XSBI, is implemented since that interface passes data over a 16-bit wide path~~. It is also necessary when connecting to a WIS since the WIS processes the data stream with 8-bit granularity.~~ When ~~neither the WIS nor~~ the XSBI is <u>not</u> implemented, the internal data-path width between the PCS and PMA is an implementation choice. Depending on the path width, the gearbox functionality may not be necessary.

49.2.8 Test-pattern generators

When the transmit channel is operating in test-pattern mode, it sends 16 bits of test pattern at a time via PMA_UNITDATA.request primitives. When the PCS allows direct connection to the PMA, the test-pattern generator shall be implemented. ~~The test-pattern generator does not apply to a PCS, which only supports connection to the WIS. A PCS which supports both WIS and direct PMA attachment may reject or allow an attempt to activate a transmit test-pattern mode when a WIS is attached.~~

There are two types of required transmit test patterns: square wave and pseudo-random. The square wave pattern is intended to aid in conducting certain transmitter tests. It is not intended for receiver tests and the receiver is not expected to receive this test pattern. The pseudo-random test-pattern mode is suitable for receiver tests and for certain transmitter tests. There is also an optional PRBS31 test pattern, which may be used for some transmit and receiver tests. When this option is supported, both the PRBS31 test-pattern generator and the PRBS31 test-pattern checker shall be provided. See Clause 52.9 for recommendations on the appropriate pattern for tests.

When square wave pattern is selected, the PCS will send a repeating pattern of n ones followed by n zeros where n may be any number between 4 and 11 inclusive. The value of n is an implementation choice.

When pseudo-random pattern is selected, the test pattern is generated by the scrambler using the seeds loaded through the MDIO registers and the selected data pattern. The scrambler is loaded with a seed or its inverse at the start of a block every 128 blocks. The seeds are loaded in the following pattern:

Seed A
Seed A Invert

Seed B
Seed B Invert

Invert indicates that the seed is inverted for that load. Either 64 zeros or the 64-bit encoding for two Local Fault ordered_sets can be selected as the data pattern. After loading Seed A or Seed B, the scrambler input shall be driven with the data pattern. After loading Seed A Invert or Seed B Invert, the scrambler input shall be driven with the inverse of the data pattern. While in pseudo-random test-pattern mode, the sync headers will be the control sync header, 10. Thus the pseudo-random test pattern is a series of blocks with the control sync header and a pseudo-random payload. The characteristics of the pseudo-random test pattern can be varied by varying the seed values and data input.

When the optional PRBS31 mode is selected, the PRBS31 pattern generator sends 16 bits of PRBS31 test pattern at a time via PMA_UNITDATA.request primitives. The PRBS31 test pattern is the output of a Pseudo-Random Bit Sequence of order 31 (PRBS31) generator. The PRBS31 pattern generator shall produce the same result as the implementation shown in Figure 49–9. This implements the inverted version of the bit stream produced by the polynomial:

$$G(x) = 1 + x^{28} + x^{31} \qquad (49–2)$$

The initial value of the PRBS31 pattern generator shall not be all zeros. It may be any other value.

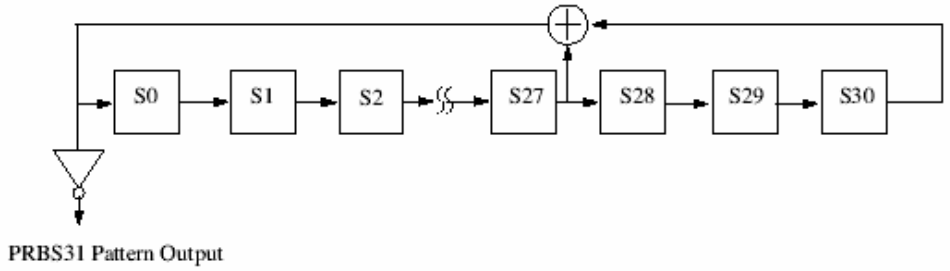

Figure 49–9—PRBS31 pattern generator

49.2.9 Block synchronization

When the receive channel is operating in normal mode, the block synchronization function receives data via 16-bit PMA_UNITDATA.request ~~or WIS_UNITDATA.request~~ primitives. It shall form a bit stream from the primitives by concatenating requests with the bits of each primitive in order from rx_data-group<0> to rx_data-group<15> (see Figure 49–6). It obtains lock to the 66-bit blocks in the bit stream using the sync headers and outputs 66-bit blocks. Lock is obtained as specified in the block lock state machine shown in Figure 49–12.

49.2.10 Descrambler

The descrambler processes the payload to reverse the effect of the scrambler using the same polynomial. It shall produce the same result as the implementation shown in Figure 49–10.

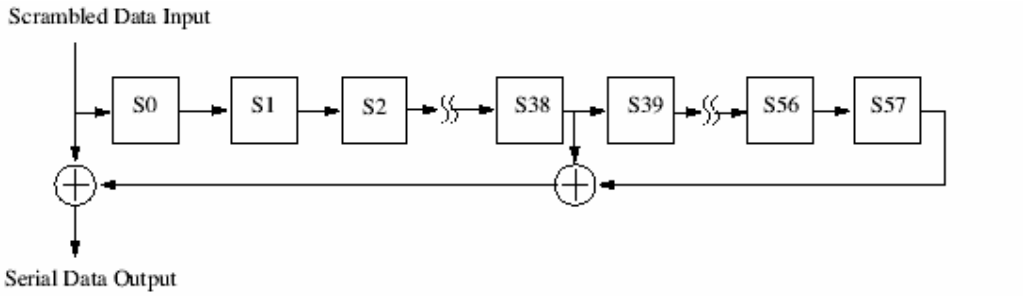

Figure 49–10—Descrambler

49.2.11 Receive process

The receive process decodes blocks to produce RXD'<31:0> and RXC'<3:0> for processing by ~~transmission to~~ the CRC8 Remove process ~~XGMII~~. Two XGMII data ~~transfers~~ words are decoded from each block. Where the XGMII and PMA sublayer data rates are not synchronized to a 16:33 ratio, the receive process will insert idles, delete idles, or delete sequence ordered sets to adapt between rates. ~~The WIS data rate is always slower than the XGMII data rate and a PCS connected to a WIS will insert idles to adapt between the rates.~~

The receive process decodes blocks as specified in the receive state machine shown in Figure 49–15.

49.2.X2 CRC8 Remove process

The CRC8 Remove process removes and checks the additional CRC8 error protection field from frames before they are passed to the XGMII. The Remove process replaces the CRC8 data character and the/T/ character following it with a /T/ and an /I/ character to restore the frame to its original length. If the CRC8 checksum carried in the frame does not match the value generated by the CRC8 generator defined in 49.2.X.1 the frame is in error. The remove process shall take two actions on frames with CRC8 errors[1].

1. The last character of the CRC32 field is replaced with a /E/ character, ensuring that the MAC layer will recognize this frame as in error.
2. The errored_block_count counter is incremented by one.

The CRC8 Remove state machine parses XGMII signals in order to recognize frame boundaries, check the CRC8 field, and modify XGMII words. The CRC8 Insert process only modifies XGMII signals at the end of frames in order to remove the CRC8 field, or mark an errored frame.

Table 49-X2 indicates how XGMII signals are modified. In this table subscripts indicate the character position in an XGMII word. Subscript 0 indicates the first character - equivalent to DATA<7:0> and CONTROL<0>, and subscript 3 the last character - equivalent to DATA<31:24> and CONTROL<3>.

- R_0 through R_3 indicate the four characters of a (raw) input word.
- N_0 through N_4 indicate the characters of the next (raw) input word. That is the word following R_0 through R_3. Only N_0 through N_1 are used in the table.
- M_0 through M_3 indicate the four characters of the (modified) output word

The CRC8 Remove process generates XGMII signals RXD<31:0> and RXC<3:0> based on the RXD'<31:0> and RXC'<3:0> words from the PCS Receive process.

The CRC8 Remove process translates words as specified in the CRC8 Remove state machine shown in Figure 49–X3.

---

[1] If a frame contains /E/ characters, which will most likely cause CRC8 to fail, should the error actions take place ?. It is important that the CRC8 be removed and the original /T/ restored, but adding the /E/, and possibly re-counting the error seems counter-productive.

rem_raw				next_rem_raw		rem_mod			
R_0	R_1	R_2	R_3	N_0	N_1	M_0	M_1	M_2	M_3
___ /T/	Good CRC8	/T/	___ /T/	X	X	R_0	/T/	/I/	R_3
___ /T/	___ /T/	Good CRC8	/T/	X	X	R_0	R_1	/T/	/I/
___ /T/	___ /T/	___ /T/	Good CRC8	/T/	X	R_0	R_1	R_2	/T/
___ /T/	___ /T/	___ /T/	___ /T/	Good CRC8	/T/	R_0	R_1	R_2	R_3
___ /T/	Bad CRC8	/T/	___ /T/	X	X	/E/	/T/	/I/	R_3
___ /T/	___ /T/	Bad CRC8	/T/	X	X	R_0	/E/	/T/	/I/
___ /T/	___ /T/	___ /T/	Bad CRC8	/T/	X	R_0	R_1	/E/	/T/
___ /T/	___ /T/	___ /T/	___ /T/	Bad CRC8	/T/	R_0	R_1	R_2	/E/
/T/	___ /T/	___ /T/	___ /T/	X	X	/I/	R_1	R_2	R_3
___ /T/	/T/	___ /T/	___ /T/	X	X	/T/	/I/	R_2	R_3
Else						R_0	R_1	R_2	R_3

Table 49-X2 Translation table for CRC8 removal

49.2.12 Test-pattern checker

When the PCS allows direct connection to the PMA, the pseudo-random test-pattern checker shall be implemented. ~~The pseudo-random test-pattern checker does not apply to a PCS, which only supports connection to the WIS.~~ When the PRBS31 test-pattern option is supported, the PRBS31 receive test-pattern checker shall be implemented. ~~A PCS that supports both WIS and direct PMA attachment may reject or allow an attempt to activate a receive test-pattern mode when a WIS is attached.~~

When the receive channel is operating in pseudo-random test-pattern mode, the pseudo-random test-pattern checker checks the bits received via PMA_UNITDATA.indicate primitives.

The pseudo-random test-pattern checker utilizes the lock state machine and the descrambler operating as they do during normal data reception. The hi_ber state machine is disabled during receive test-pattern mode. When block_lock is true and the pseudo-random receive test-pattern mode is active, the pseudo-random testpattern checker observes the output from the descrambler. When the output of the descrambler is the data pattern or its inverse, a match is detected. Since the transmitter's scrambler is loaded with a seed value every 128 blocks and the receiver's descrambler is running normally, a mismatch will be detected once every 128 blocks in the absence of errors. The pseudo-random test-pattern checker will count 128-block windows. When operating in pseudo-random test pattern, the test-pattern error counter counts blocks with a mismatch corrected to remove the effect of loading a new seed. The first block with a mismatch in a window shall not increment the test-pattern error counter. Any subsequent block with a mismatch in a window indicates an error and shall increment the test-pattern error counter.

When the receive channel is operating in PRBS31 test mode, the PRBS31 pattern checker checks the bits received in PMA_UNITDATA.indicate primitives relative to the PRBS31 test pattern.

The PRBS31 pattern error checker is self-synchronizing. It compares each bit received to the result of the PRBS31 generator based on the prior 31 bits received. It shall produce the same result as the implementation shown in Figure 49–11. When no errors occur, the PRBS31 pattern error signal will be zero. When an isolated bit error occurs, it will cause the PRBS31 pattern error signal to go high three times; once when it is received and once when it is at each tap. The test-pattern error counter shall increment once for each bit time that the PRBS31 pattern error signal is high.

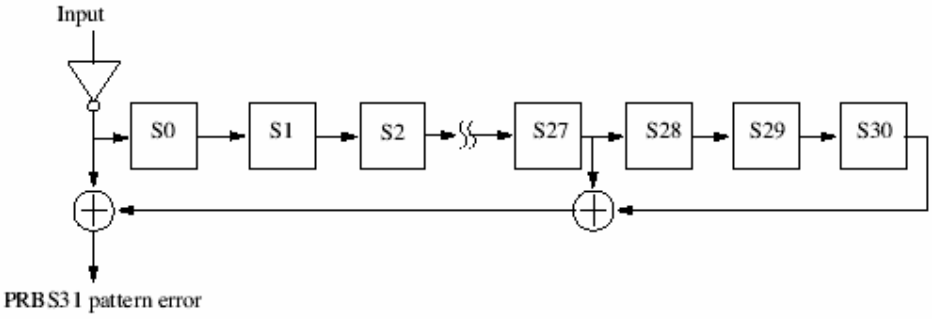

Figure 49–11—PRBS31 pattern checker

Note that the test-pattern error counter behavior is dependent on the test-pattern mode. In pseudo-random test mode, it is counting block errors. In PRBS31 test mode, it is counting bit errors at the PRBS31 pattern checker output.

49.2.13 Detailed functions and state diagrams

49.2.13.1 State diagram conventions

The body of this subclause is comprised of state diagrams, including the associated definitions of variables, constants, and functions. 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. State diagram timers follow the conventions of 14.2.3.2. The notation ++ after a counter or integer variable indicates that its value is to be incremented.

49.2.13.2 State variables

49.2.13.2.1 Constants

EBLOCK_R<71:0>
: 72 bit vector to be sent to the CRC Remove process ~~XGMII interface~~ 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 CRC Remove process ~~XGMII interface~~ containing two Local Fault ordered_sets. The Local Fault ordered_set is defined in 46.3.4.

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

49.2.13.2.2 Variables

ber_test_sh
: Boolean variable that is set true when a new sync header is available for testing and false when BER_TEST_SH state is entered. A new sync header is available for testing when the Block Sync process has accumulated enough bits from the PMA ~~or the WIS~~ to evaluate the header of the next block

block_lock

Boolean variable that is set true when receiver acquires block delineation

hi_ber

Boolean variable which is asserted true when the ber_cnt exceeds 16 indicating a bit error ratio $>10^{-4}$

reset

Boolean variable that controls the resetting of the PCS. 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 into low-power mode.

r_test_mode

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

rx_coded<65:0>

Vector containing the input to the 64B/66B decoder. The format for this vector is shown in Figure 49–7. The leftmost bit in the figure is rx_coded<0> and the rightmost bit is rx_coded<65>.

rx_raw<71:0>

Vector containing two successive XGMII transfers from the PCS Receive process. RXC'<0> through RXC'<3> for the first transfer are placed in rx_raw<0> through rx_raw<3>, respectively. RXC'<0> through RXC'<3> for the second transfer are placed in rx_raw<4> through rx_raw<7>, respectively. RXD'<0> through RXD'<31> for the first transfer are placed in rx_raw<8> through rx_raw<39>, respectively. RXD'<0> through RXD'<31> for the second transfer are placed in rx_raw<40> through rx_raw<71>, respectively.

sh_valid

Boolean indication that is set true if received block rx_coded has valid sync header bits. That is, sh_valid is asserted if rx_coded<0> ≠ rx_coded<1> and de-asserted otherwise.

signal_ok

Boolean variable that is set based on the most recently received value of PMA_UNITDATA.indicate(SIGNAL_OK) ~~or WIS_UNITDATA.indicate(SIGNAL_OK)~~. It is true if the value was OK and false if the value was FAIL.

slip_done

Boolean variable that is asserted true when the SLIP requested by the Block Lock State Machine has been completed indicating that the next candidate block sync position can be tested.

test_sh

Boolean variable that is set true when a new sync header is available for testing and false when TEST_SH state is entered. A new sync header is available for testing when the Block Sync process has accumulated enough bits from the PMA ~~or the WIS~~ to evaluate the header of the next block

tx_coded<65:0>

Vector containing the output from the 64B/66B encoder. The format for this vector is shown in Figure 49–7. The leftmost bit in the figure is tx_coded<0> and the rightmost bit is tx_coded<65>.

tx_raw<71:0>

Vector containing two successive modified XGMII transfers from the CRC8 Insertion process. TXC'<0> through TXC'<3> for the first transfer are placed in tx_raw<0> through tx_raw<3>, respectively. TXC'<0> through TXC'<3> for the second transfer are placed in tx_raw<4> through tx_raw<7>, respectively. TXD'<0> through TXD'<31> for the first transfer are placed in tx_raw<8> through tx_raw<39>, respectively. TXD'<0> through TXD'<31> for the second transfer are placed in tx_raw<40> through tx_raw<71>, respectively.

rem_raw<35:0>

Vector containing an XGMII word input to the CRC8 Removal process from the PCS Receive process. RXC'<0> through RXC'<3> are placed in rem_raw<0> through rem_raw<3>, respectively. RXD'<0> through RXD'<31> for the transfer are placed in rem_raw<4> through rem_raw<35>, respectively.

next_rem_raw<35:0>

Prescient Vector containing the value that of the rem_raw vector immediately following the current rem_raw vector.

rem_mod<35:0>

Vector containing one XGMII transfer output from the CRC8 Removal process. RXC<0> through RXC<3> are in rem_mod<0> through rem_mod<3>, respectively. RXD<0> through RXD<31> for the transfer are in rem_mod<4> through rem_mod<35>, respectively.

ins_raw<35:0>

Vector containing one XGMII transfer. RXC<0> through RXC<3> are placed in ins_raw<0> through ins_raw<3>, respectively. RXD<0> through RXD<31> for the transfer are placed in ins_raw<4> through ins_raw<35>, respectively.

ins_t0_raw<35:0>

Vector set to the value of ins_raw with the first character (ins_raw<0>, and ins_raw<4> through ins_raw<11>) replaced with a /T/ character.

ins_mod<35:0>

Vector containing the output from the CRC8 Insertion process. The format for this vector is that of a single XGMII transfer. RXC'<0> through RXC'<3> are placed in ins_mod<0> through ins_mod<3>, respectively. RXD'<0> through RXD'<31> for the transfer are placed in ins_mod<4> through ins_mod<35>, respectively.

49.2.13.2.3 Functions

DECODE(rx_coded<65:0>)

Decodes the 66-bit vector returning rx_raw<71:0> which is sent to the XGMII. The DECODE function shall decode the block as specified in 49.2.4.

ENCODE(tx_raw<71:0>)

Encodes the 72-bit vector returning tx_coded<65:0> of which tx_coded<63:0> is sent to the scrambler. The two high order sync bits bypass the scrambler. The ENCODE function shall encode the block as specified in 49.2.4.

R_BLOCK_TYPE = {C, S, T, D, E}

This function classifies each 66-bit rx_coded vector as belonging to one of the five types depending on its contents.

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/;

b) A block type field of 0x2d or 0x4b, a valid O code, and four valid control characters;

c) A block type field of 0x55 and two valid O codes.

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

a) A block type field of 0x33 and four valid control characters;

b) A block type field of 0x66 and a valid O code;

c) 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.

A valid control character is one containing a 10GBASE-R control code specified in Table 49–1. A valid O code is one containing an O code specified in Table 49–1.

R_TYPE(rx_coded<65:0>)

Returns the R_BLOCK_TYPE of the rx_coded<65:0> bit vector.

R_TYPE_NEXT

Prescient end of packet check function. It returns the R_BLOCK_TYPE of the rx_coded vector immediately following the current rx_coded vector.

SLIP

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

T_BLOCK_TYPE = {C, S, T, D, E}

This function classifies each 72-bit tx_raw vector as belonging to one of the five types depending on its contents.

Values:

C; The vector contains one of the following:

a) eight valid control characters other than /O/, /S/, /T/ and /E/;

b) one valid ordered_set and four valid control characters other than /O/, /S/ and /T/;

c) two valid ordered sets.

S; The vector contains an /S/ in its first or fifth character, any characters before the S character are valid control characters other than /O/, /S/ and /T/ or form a valid ordered_set, 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 XGMII control code specified in Table 49–1. A valid ordered_set consists of a valid /O/ character in the first or fifth characters and data characters in the three characters following the /O/. A valid /O/ is any character with a value for O code in Table 49–1.

T_TYPE(tx_raw<71:0>)

Returns the T_BLOCK_TYPE of the tx_raw<71:0> bit vector.

CRC8_INS(ins_raw<35:0>)

Translates the 36-bit vector returning a modified 36-bit vector vector <35:0>. Depending on the SM state this modified value may replace the raw XGMII data sent on to the PCS Receive process. The function shall translate the vector as specified in Table 49.X1

CRC8_INSERT_BLOCK_TYPE = {S, D, T, T4, I, C, E}

This function classifies each 36bit raw XGMII vector as to belonging to one of seven types depending on its contents.

Values:

S: The vector contains an /S/ in its first character, and all other characters are data characters

D: The vector contains 4 data characters

T: The vector contains a /T/ in one of its first 3 characters, all characters before the /T/ are data characters, and all characters after the /T/ are valid control characters other than /O/, /S/, or /T/.

T4: The vector contains a /T/ in its last character, and all other characters are data characters.

I: The vector contains a valid control character other than /O/, /S/, or /T/ in its first character and and all other characters are valid control characters.

C: The vector contains an /O/, /S/, or /T/ in its first character and and all other characters are valid control characters.

E: The vector does not meet the criteria for any other value.

INS_TYP(ins_raw<35:0>)

Returns the CRC8_INSERT_BLOCK_TYPE of the ins_raw<35:0> bit vector.

CRC8_REM(rem_raw<35:0>)

Translates the 36-bit XGMII vector returning a modified XGMII vector <35:0>. Depending on the SM state this modified value may replace the raw XGMII data output from the layer. The function shall translate the vector as specified in the Table 49-X2.

CRC8_REMOVE_BLOCK_TYPE = {S, D, T12, T34, C, E}

This function classifies each 36bit raw XGMII vector as to belonging to one of six types depending on its contents.

Values:

S: The vector contains a /S/ in its first character, and all other characters are data characters

D: The vector contains 4 data characters.

T12: The vector contains a /T/ in one of its first 2 characters, and all characters before the /T/ are data characters.

T34: The vector contains a /T/ in one of its last 2 characters, and all characters before the /T/ are data characters..

C: The first character is a valid control character other than /T/, all other characters are valid control characters.

E: The vector does not meet the criteria for any other value.

REM_TYP(vector<35:0>)

Returns the CRC8_REMOVE_BLOCK_TYPE of a <35:0> bit XGMII vector.

49.2.13.2.4 Counters

ber_cnt

Count up to a maximum of 16 of the number of invalid sync headers within the current 25 μs period.

sh_cnt

Count of the number of sync headers checked within the current 64 block window.

sh_invalid_cnt

Count of the number of invalid sync headers within the current 64 block window.

49.2.13.2.5 Timers

State diagram timers follow the conventions of 14.2.3.2.

125us_timer

Timer that is triggered every 125 μs +1%, –25%.

49.2.13.3 State diagrams

The Lock state machine shown in Figure 49–12 determines when the PCS has obtained lock to the received data stream. The BER Monitor state machine shown in Figure 49–13 monitors the received signal for high bit error ratio.

The Transmit state machine shown in Figure 49–14 controls the encoding of transmitted blocks. It makes exactly one transition for each transmit block processed. Though the Transmit state machine sends Local Fault ordered sets when reset is asserted, the scrambler and gearbox may not be operational during reset. Thus, the Local Fault ordered sets may not appear on the ~~WIS or~~ PMA service interface.

The Receive state machine shown in Figure 49–15 controls the decoding of received blocks. It makes exactly one transition for each receive block processed.

The CRC8 Insert state machine shown in Figure 49–X2 controls the insertion of CRC8 in transmitted XGMII words. It makes exactly one transition for each XGMII word processed.

The CRC8 Insert state machine shown in Figure 49–X3 controls the removal, and checking of CRC8 from received XGMII words. It makes exactly one transition for each XGMII word processed.

The PCS shall perform the functions of Lock, BER CRC8 Insert, CRC8 Remove, Monitor, Transmit and Receive as specified in these state machines.

49.2.14 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 be provided.

49.2.14.1 Status

PCS_status:
: Indicates whether the PCS is in a fully operational state. It is only true if block_lock is true and hi_ber is false. This status is reflected in MDIO register 3.32.12. A latch low view of this status is reflected in MDIO register 3.1.2 and a latch high of the inverse of this status, Receive fault, is reflected in MDIO register 3.8.10.

block_lock:
: Indicates the state of the block_lock variable. This status is reflected in MDIO register 3.32.0. A latch low view of this status is reflected in MDIO register 3.33.15.

hi_ber:
: Indicates the state of the hi_ber variable. This status is reflected in MDIO register .32.1. A latch high view of this status is re.ected in MDIO register 3.33.14.

49.2.14.2 Counters

The following counters are reset to zero upon read and upon reset of the PCS. When they reach all ones, they stop counting. Their purpose is to help monitor the quality of the link.

ber_count:
: 6-bit counter that counts each time BER_BAD_SH state is entered. This counter is reflected in MDIO register bits 3.33.13:8. Note that this counter counts a maximum of 16 counts per 125 µs since the BER_BAD_SH can be entered a maximum of 16 times per 125 µs window.

errored_block_count:
: 8-bit counter. When the receiver is in normal mode, errored_block_count counts once for each time RX_E state is entered, and once for every frame received with a CRC8 error. This counter is reflected in MDIO register bits 3.33.7:0.

test_pattern_error_count:
: 16-bit counter. When the receiver is in test-pattern mode, the test_pattern_error_count counts errors as described in 49.2.12. This counter is reflected in MDIO register bits 3.43.14:0.

49.2.14.3 Test mode control

RX_test_mode:
: Boolean variable controlling transmit channel operating mode. When false, the transmit channel operates in normal mode. When true, the transmit channel operates in test-pattern mode.

rx_test_mode:
: Boolean variable controlling receive channel operating mode. When false, the receive channel operates in normal mode. When true, the receive channel operates in test-pattern mode.

49.2.14.4 Loopback

The PCS shall be placed in Loopback mode when the Loopback bit in MDIO register 3.0.14 is set to a logic one. In this mode, the PCS shall accept data on the transmit path from the XGMII and return it on the receive path to the XGMII. In addition, the PCS shall transmit a continuous stream of 0x00FF data words to the PMA ~~or WIS~~ sublayer, and shall ignore all data presented to it by the PMA ~~or WIS~~ sublayer.

49.2.15 Delay constraints

Predictable operation of the MAC Control PAUSE operation (Clause 31, Annex 31B) demands that there be an upper bound on the propagation delays through the network. This implies that MAC, MAC Control sublayer, and PHY implementers must conform to certain delay maxima, and that network planners and administrators conform to constraints regarding the cable topology and concatenation of devices. The sum of transmit and receive delay contributed by the 10GBASE-KR PCS shall be no more than 3584 BT.

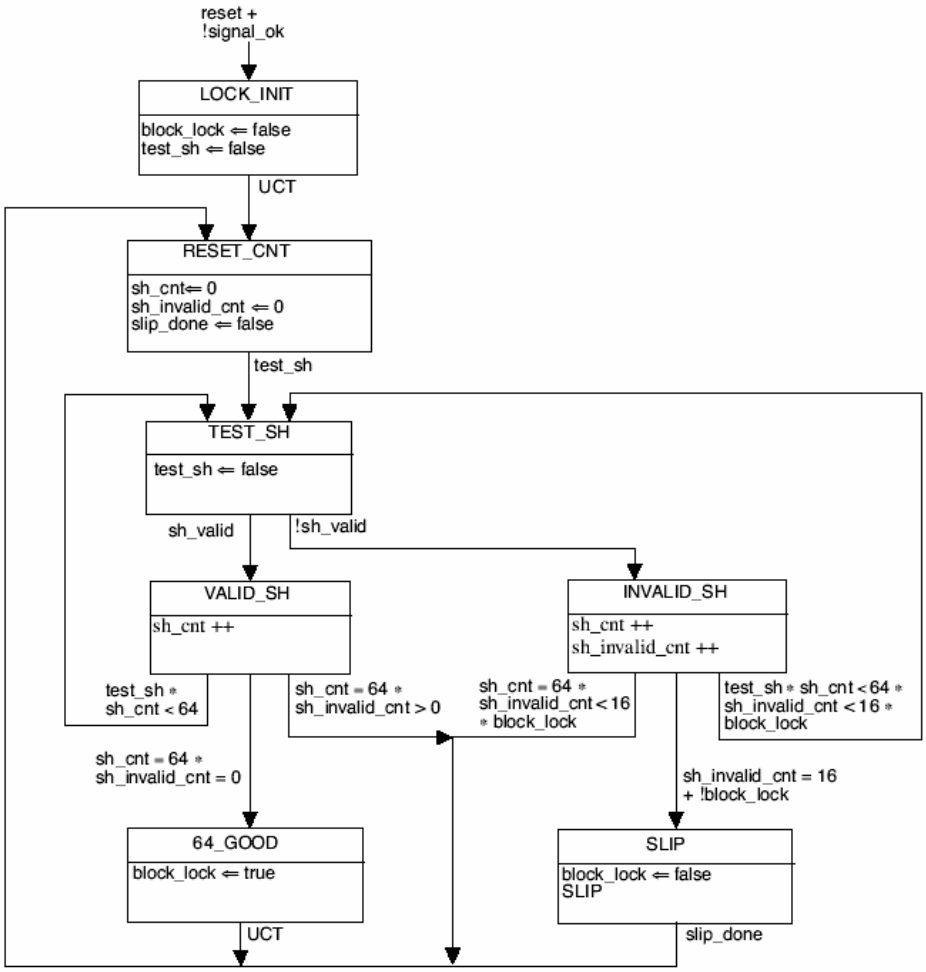

Figure 49–12—Lock state machine

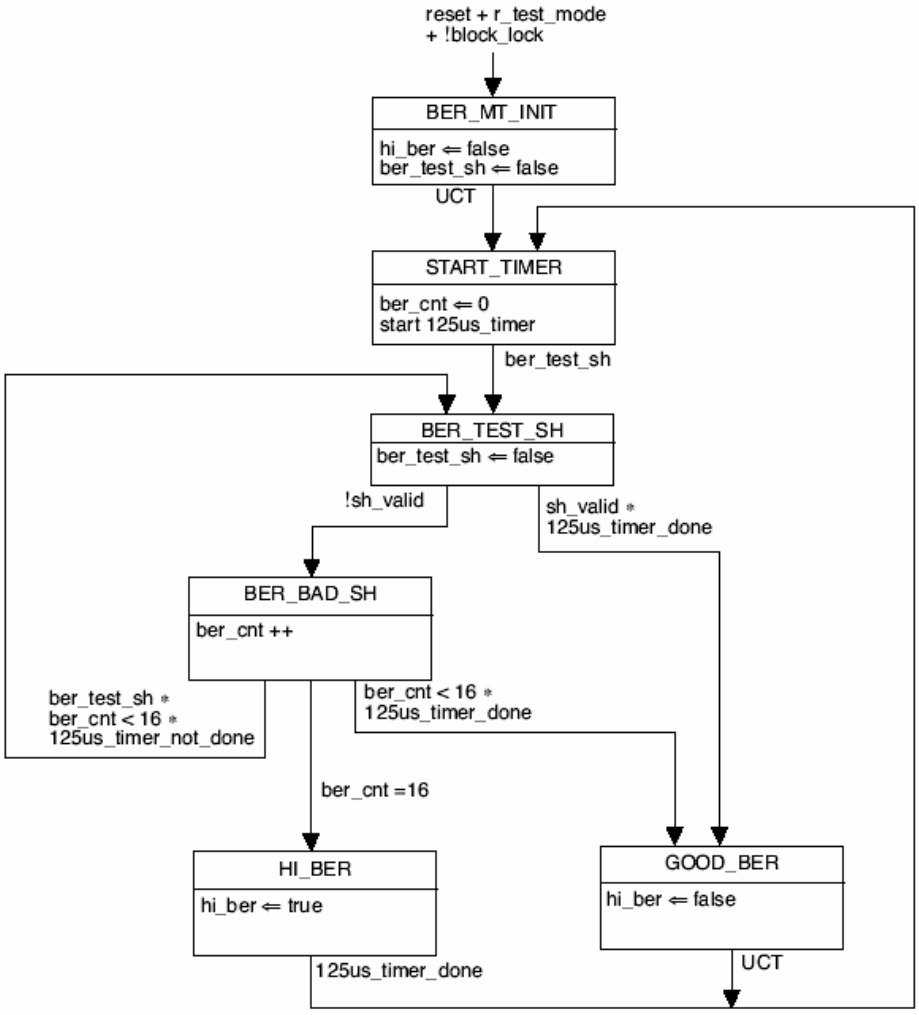

Figure 49–13—BER monitor state machine

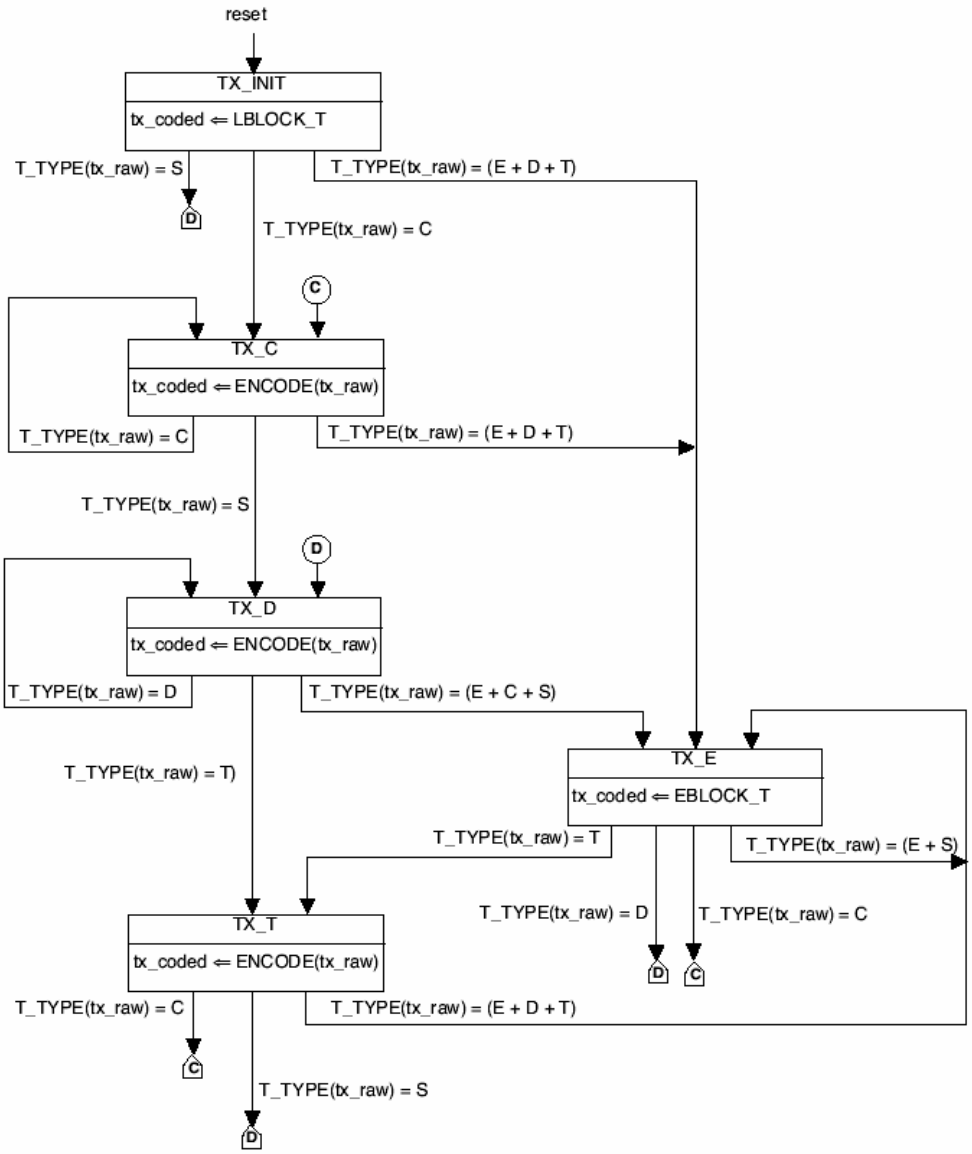

Figure 49–14—Transmit state machine

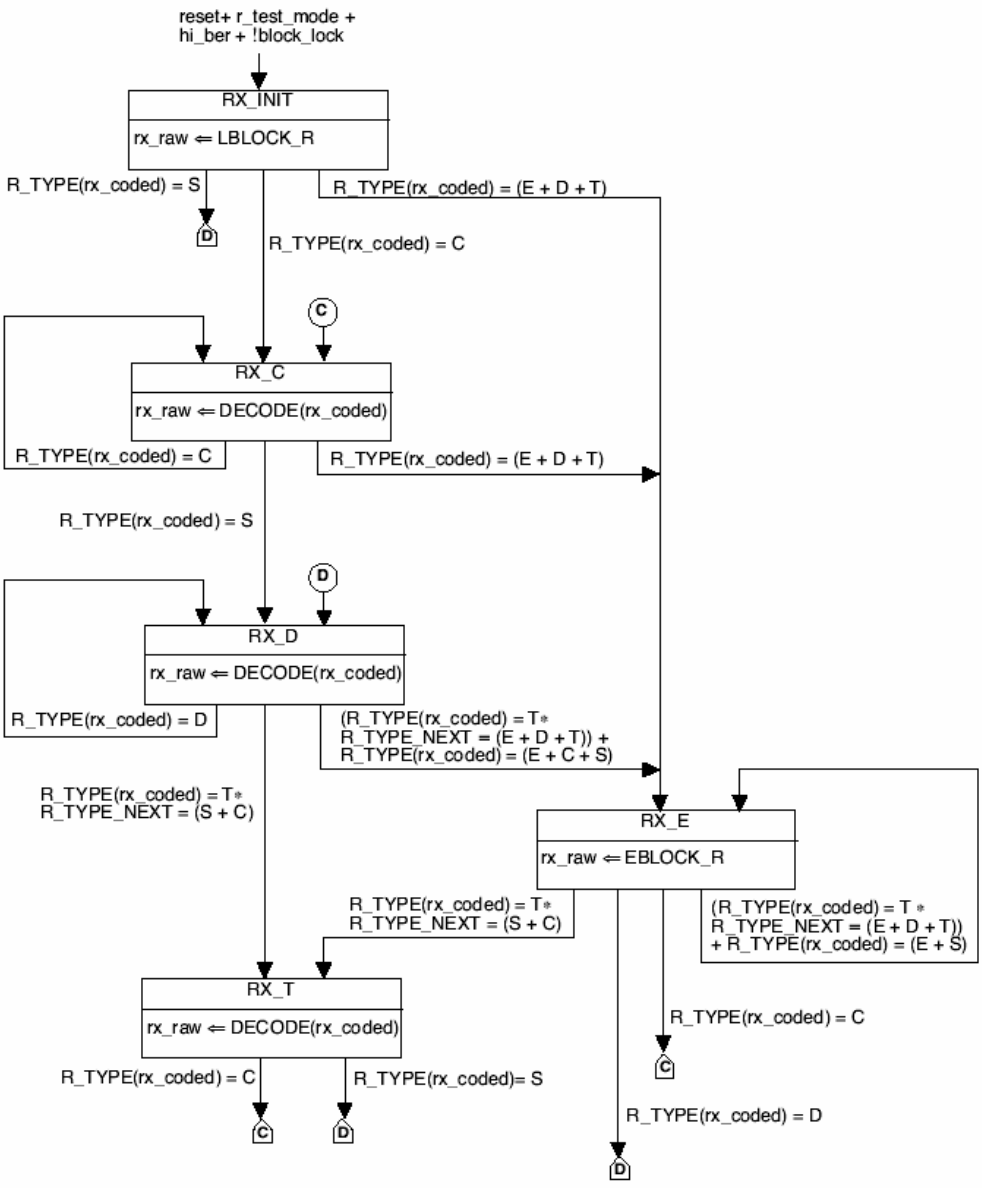

Figure 49–15—Receive state machine

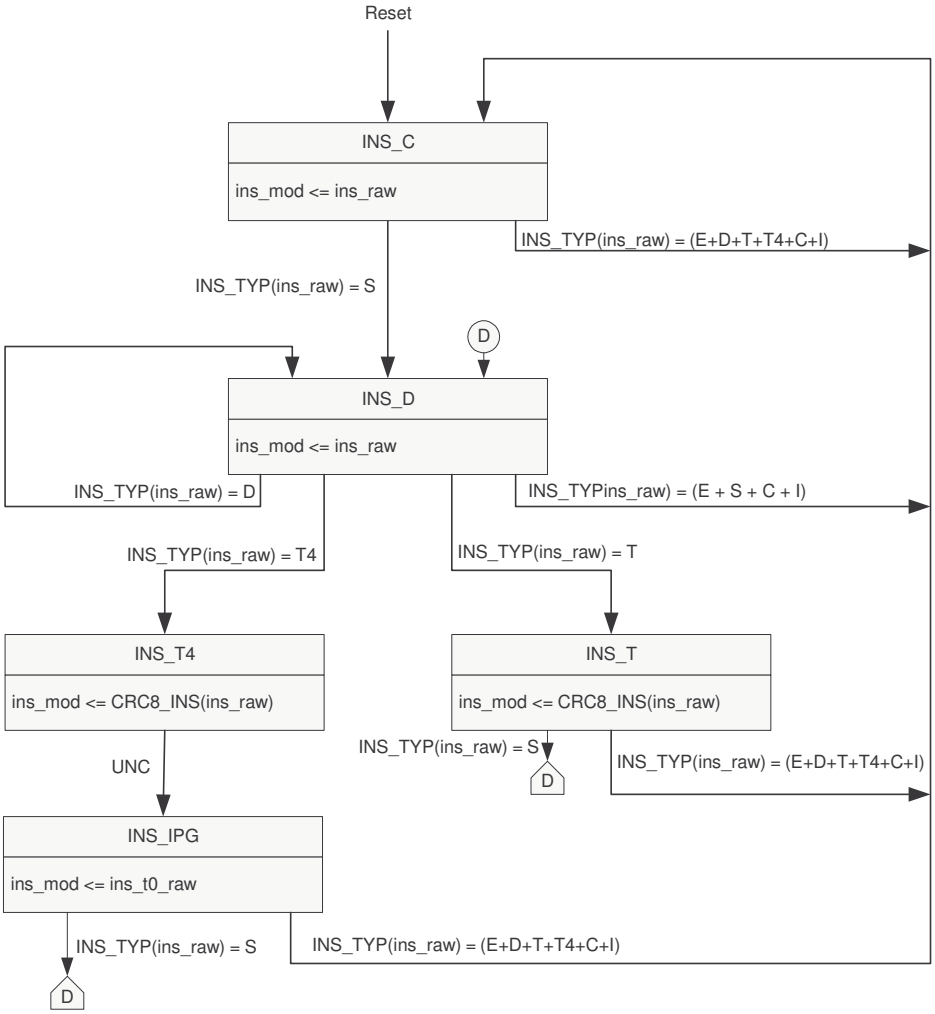

Figure 49-X2 CRC8 insert state machine

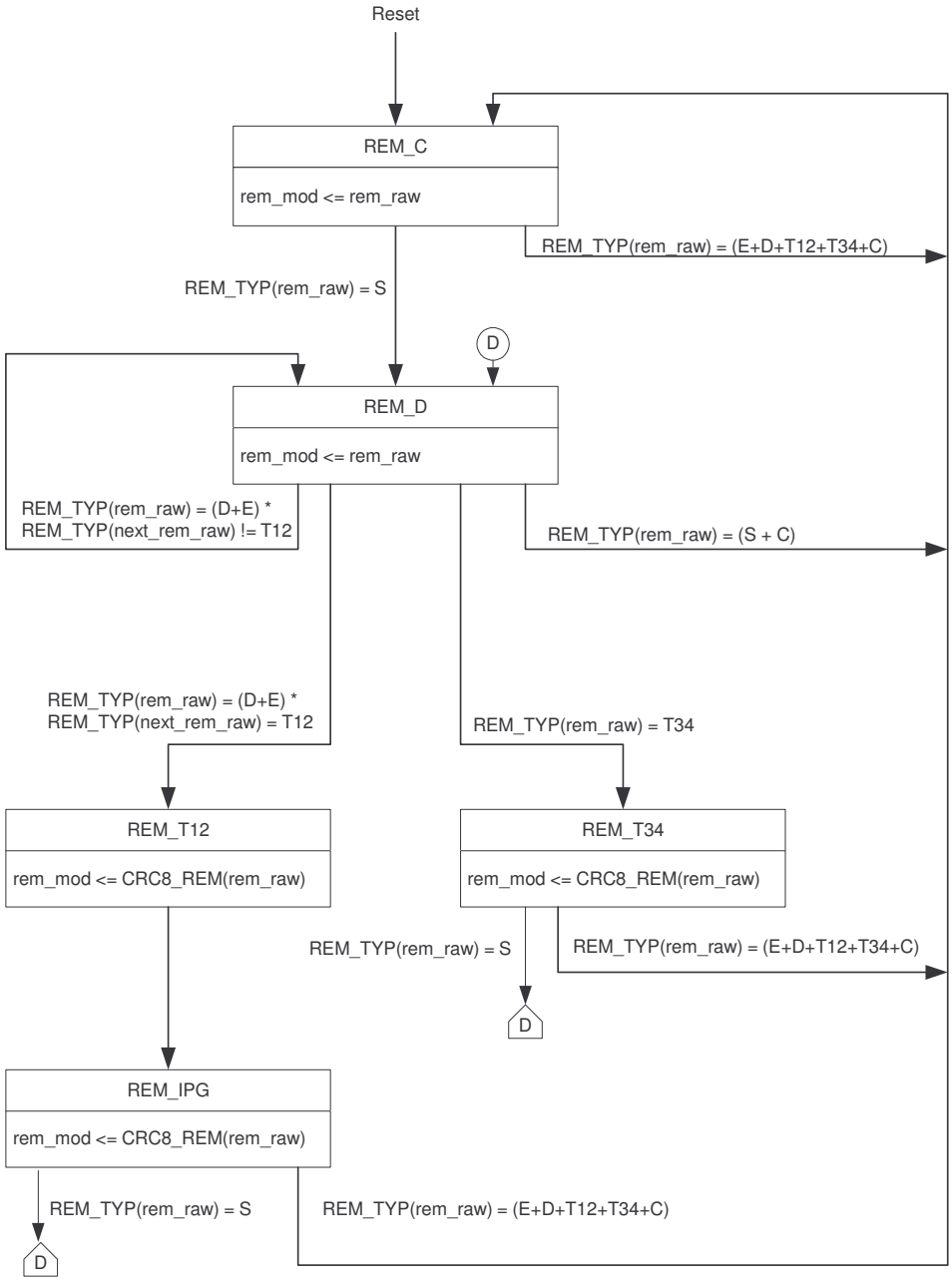

Figure 49-X3 CRC8 Remove state machine