97.3 Physical Coding Sublayer (PCS)

NOTE: Text written in italics are not approved baseline and are included only for placeholder information. The text will be change to match approved baseline when selected.

97.3.1 PCS service interface (GMII)

The PCS service interface allows the 1000BASE-T1 PCS to transfer information to and from a PCS client. The PCS Interface is precisely defined as the Gigabit Media Independent Interface (GMII) in Clause 35.

97.3.2 PCS functions

The PCS comprises one PCS Reset function and two simultaneous and asynchronous operating functions. The PCS operating functions are: PCS Transmit and PCS Receive. All operating functions start immediately after the successful completion of the PCS Reset function.

The PCS reference diagram, Figure 97–5, shows how the two operating functions relate to the messages of the PCS-PMA interface. Connections from the management interface (signals MDC and MDIO) to other layers are pervasive and are not shown in Figure 97–5.

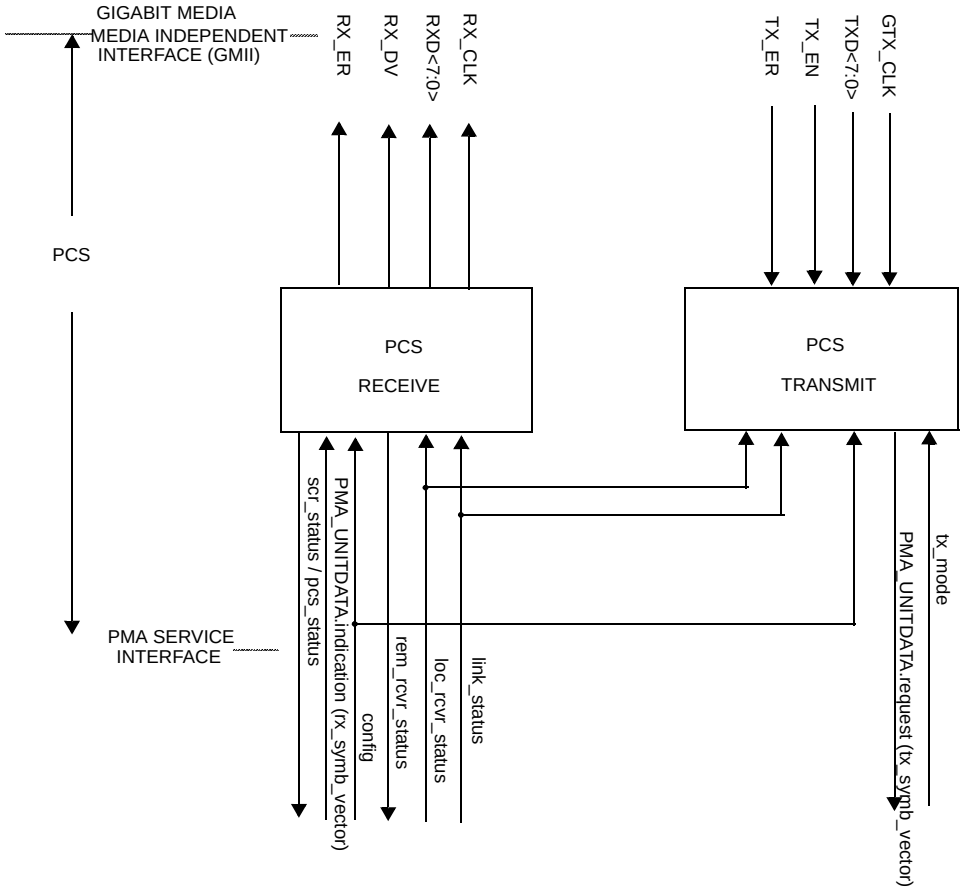


Figure 97–5—PCS reference diagram

97.3.2.1 PCS Reset function

PCS Reset initializes all PCS functions. The PCS Reset function shall be executed whenever one of the following conditions occur:

a) Power on (see 97.3.5.2.2).
b) The receipt of a request for reset from the management entity.

PCS Reset sets pcs_reset=ON while any of the above reset conditions hold true. All state diagrams take the open-ended pcs_reset branch upon execution of PCS Reset. The reference diagrams do not explicitly show the PCS Reset function.

97.3.2.2 PCS Transmit function

The PCS Transmit function shall conform to the PCS 80B/81B Transmit state diagram in Figure 97–13 and the PCS Transmit bit ordering in Figure 97–6 and Figure 97–8.

When communicating with the GMII, the PCS uses an octet-wide, synchronous data path, with packet delimiting being provided by transmit control signals and receive control signals. Alignment to 80B/81B is performed in the PCS. The PMA sublayer operates independently of block and packet boundaries. The PCS provides the functions necessary to map packets between the GMII format and the PMA service interface format.

When the transmit channel is in normal mode, the PCS Transmit process continuously generates 81B blocks based upon the TXD <7:0>, TX_EN and TX_ER signals on the GMII. The subsequent functions of the PCS Transmit process then pack the resulting blocks which are processed by a Reed-Solomon (RS) encoder and then 3B2T mapped into a transmit RS frame of PAM3 symbols. Transmit data-units are sent to the PMA service interface via the PMA_UNITDATA.request primitive. A symbol period, T, is 4/3 ns.

If a PMA_TXMODE.indication message has the value SEND_T, PCS Transmit generates sequences of codes defined in 97.3.4.2 to the PMA via the PMA_UNITDATA.request primitive. These codes are used for training mode and only transmit the values {–TBD, TBD} to keep the transmit power in the training mode the same as the transmit power in normal mode.

During training mode an InfoField is transmitted at regular intervals containing messages for startup operation. By this mechanism, a PHY indicates the status of its own receiver to the link partner and makes requests for remote transmitter settings. (See 97.4.2.5.)

In the normal mode of operation, the PMA_TXMODE.indication message has the value SEND_N, and the PCS Transmit function uses a 81B-RS coding technique to generate at each symbol period code-groups that represent data or control. During transmission, the 81B bits are scrambled by the PCS using a PCS scrambler. During data encoding PCS Transmit utilizes a RS frame encoder, then frames are encoded into a sequence of PAM3 symbols and transferred to the PMA.

97.3.2.2.1 Use of blocks

The PCS maps GMII signals into 81-bit blocks inserted into an RS frame, and vice versa, using a 81B-RS coding scheme. The *PAM2* PMA training frame synchronization allow establishment of RS frame and 81B boundaries by the PCS Synchronization process. Blocks and frames are unobservable and have no meaning outside the PCS. The PCS functions ENCODE and DECODE generate, manipulate, and interpret blocks and frames as provided by the rules in 97.3.2.2.2.

97.3.2.2.2 81B-RS 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 encoding defined by the transmission code ensure that sufficient information is present in the PHY bit stream to make clock recovery possible at the receiver. The encoding also preserves the likelihood of detecting any RS frame errors that may occur during transmission and reception of information. In addition, the code enables the receiver to achieve PCS synchronization alignment on the incoming PHY bit stream.

The relationship of block bit positions to GMII, PMA, and other PCS constructs is illustrated in Figure 97–6 for transmit and Figure 97–7 for receive. These figures illustrate the processing of a multiplicity of blocks containing 10 data octets. See 97.3.2.2.5 for information on how blocks containing control characters are mapped.

97.3.2.2.3 Notation conventions

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

80B/81B encodes 10 data octets or control characters into an 81B block.

97.3.2.2.4 Transmission order

The PCS Transmit bit ordering shall conform to Figure 97–6 and Figure 97–8. Note that these figures show the mapping from GMII to 80B/81B block for a block containing ten data characters.

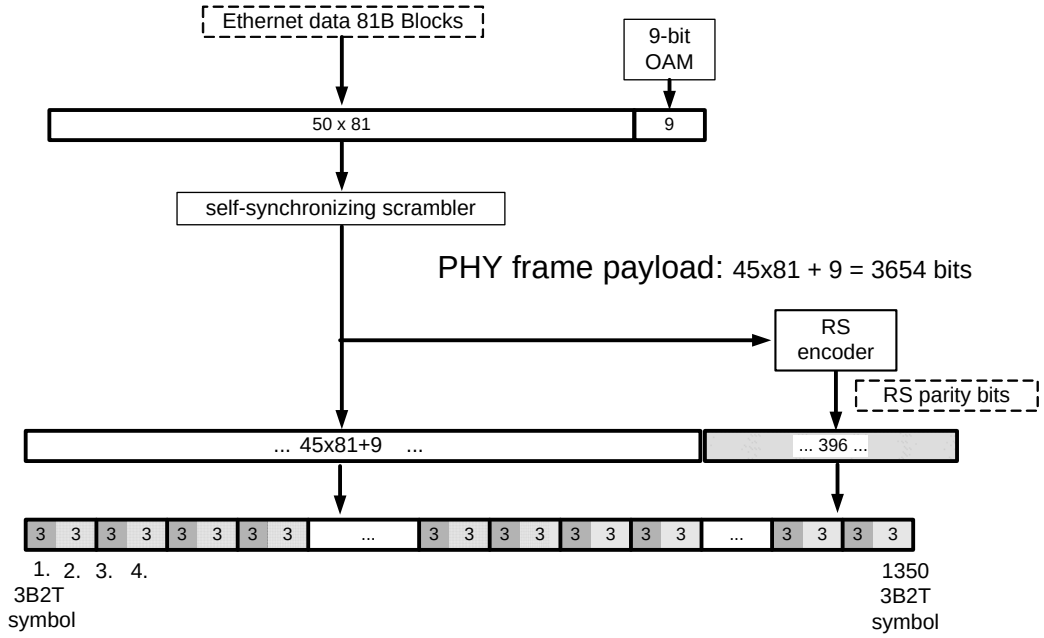


Figure 97–8—PCS detailed transmit bit ordering

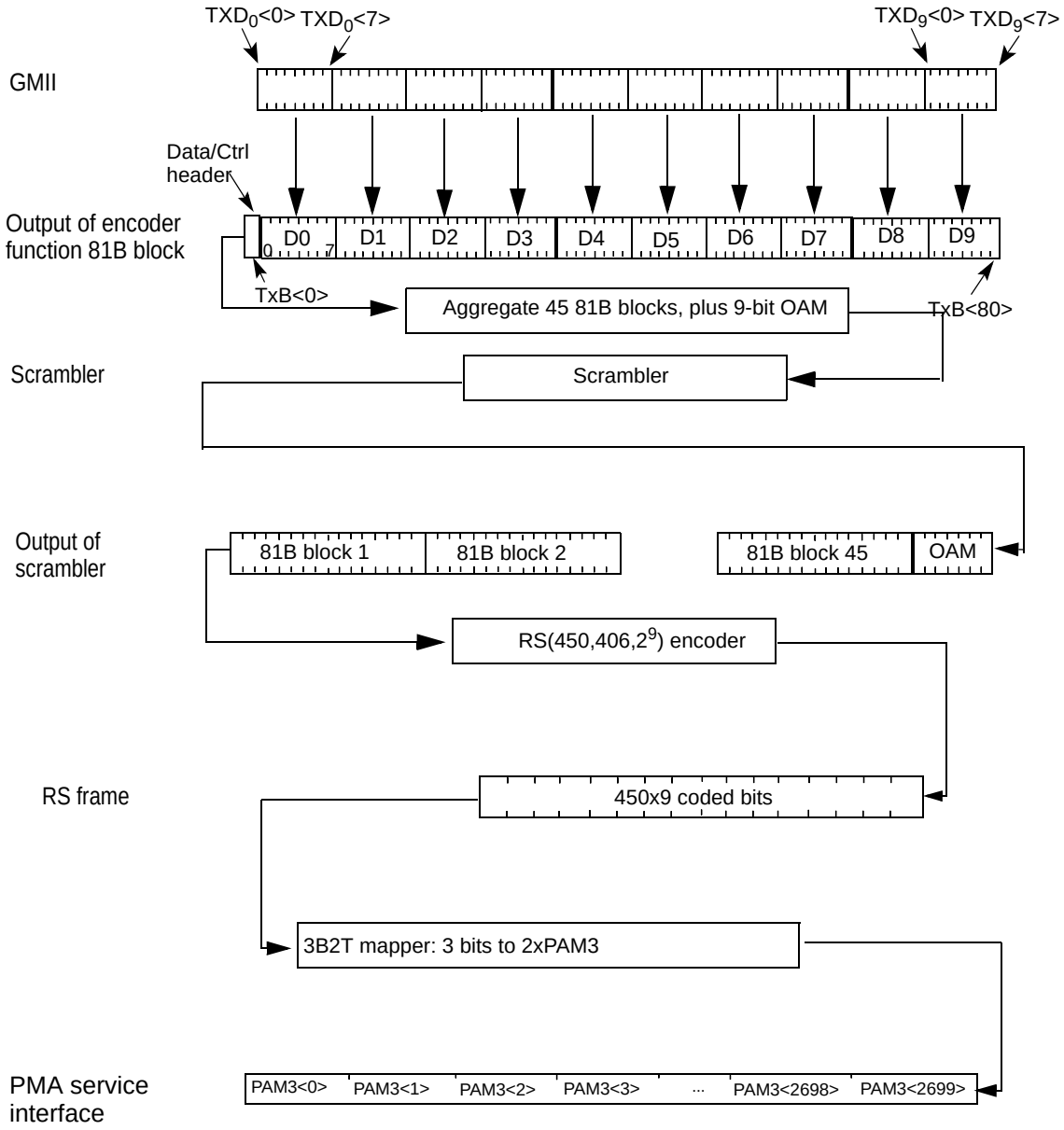


Figure 97–6—PCS Transmit bit ordering

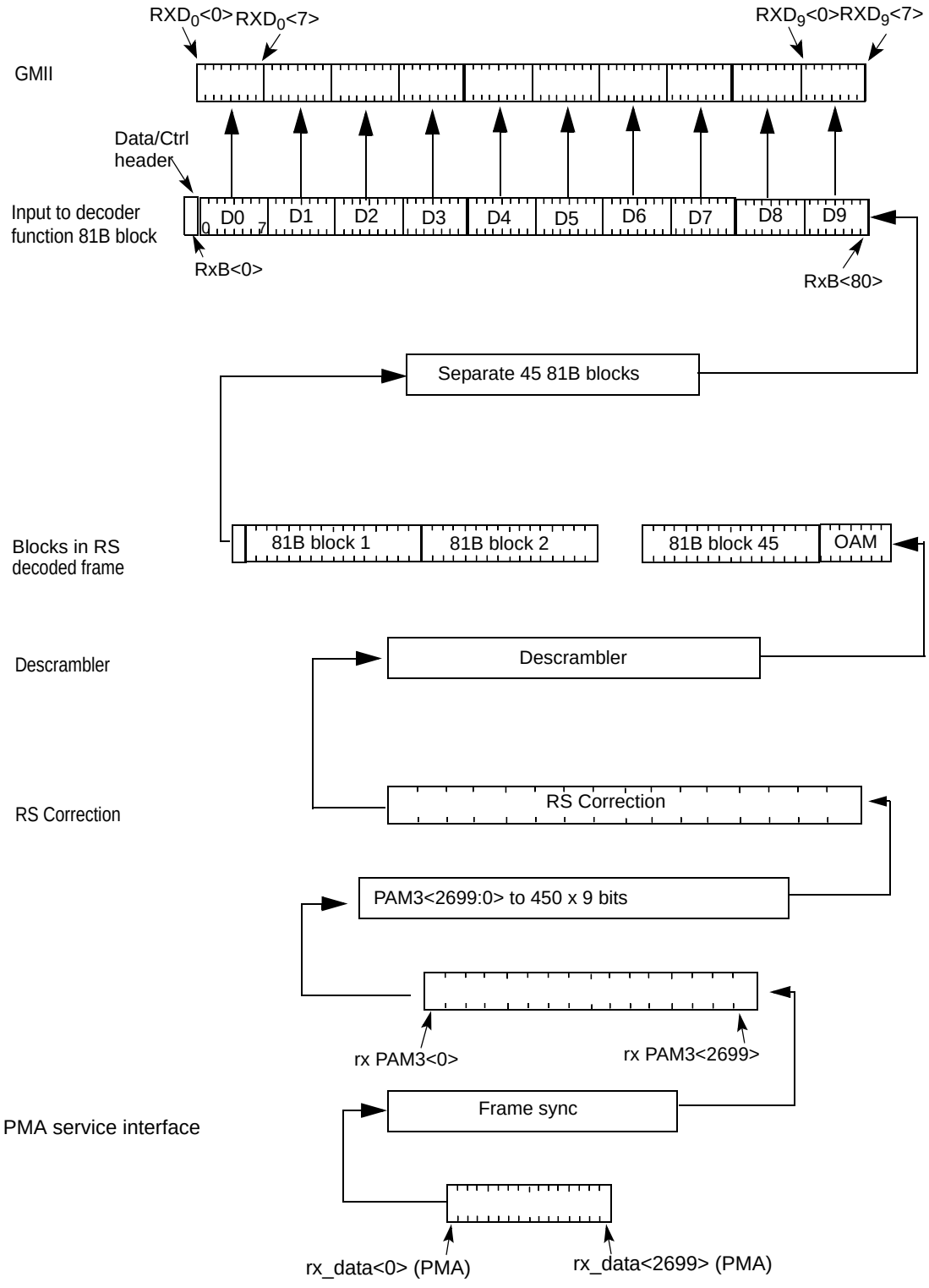


Figure 97–7—PCS Receive bit ordering

97.3.2.2.5 Block structure

Blocks consist of 81 bits. The first bit of a block is the data/ctrl header. Blocks are either data blocks or control blocks. The data/ctrl header is 0 for data blocks and 1 for control blocks. The remainder of the block contains the payload.

Data blocks contain ten data bytes. Control blocks begin with a 5-bit pointer field that indicates the location of the first control code within the block. Bit 0 to 3 of pointer points to next byte that is a control symbol. Bit 4 of pointer indicates whether the next control symbol is the final control symbol of the block: 0 = final, 1 = more control symbols. If the first octet in the block is a control character, the pointer field is followed by a 3-bit control code. Otherwise the pointer field is followed by one or more data bytes until the position of the first control code. Then the 3-bit control code indicates type of control character. The control code is followed by a 5-bit pointer field to the next control character if the prior pointer field value was greater than 15 (i.e. bit 4 =1). The pointer field may be followed by a data byte or control code depending on the value of the pointer field. In this way any combination of ten data bytes and control characters may be encapsulated within an 81B block.

The 81B block encoding is defined by the following equations where N = 10.

N = number of GMII bytes encoded into block
Bytes numbered n = 0, 1, 2, …, N-1. Byte 0 is the first one presented on GMII.
TC[n] = 0 if byte n is data byte on GMII, 1 if byte n is control byte on GMII
TC[-1] = 1 by definition
TD[n][0:7] = GMII byte n TXD[0:7] if TC[n] = 0
TD[n][5:7] = 010 – IPG, 101 – LPI, 001 – TX Error if TC[n] = 1. TD[n][0:4] is undefined.
B[0:8N] is the 8N+1 block. Bit 0 transmitted first.
OR(p) = Bitwise OR of TC[p:N-1]
NEXT(p)[0:3] = bit position of lowest bit in TC[p:N-1] that is a 1. Bit 3 is MSB.
NEXT(p)[4] = 0 if Bitwise SUM of TC[p:N-1] = 1, else 1

B[0] = OR(0)

B[8n+1:8n+4] = TD[n][0:3] – if OR(n) = 0
NEXT(n)[0:3] – if OR(n) = 1 AND TC[n-1] = 1
TD[n-1][3:6] – if OR(n) = 1 AND TC[n-1] = 0

B[8n+5] = TD[n][4] – if OR(n) = 0
NEXT(n)[4] – if OR(n) = 1 AND TC[n-1] = 1
TD[n-1][7] – if OR(n) = 1 AND TC[n-1] = 0

B[8n+6:8n+8] = TD[n][5:7] – if OR(n) = 0
TD[n][5:7] – if OR(n) = 1 AND TC[n] = 1
TD[n][0:2] – if OR(n) = 1 AND TC[n] = 0

97.3.2.2.6 Control codes

A subset control characters defined at the GMII are supported by the 1000BASE-T1 PCS. When TX_ER is asserted, the 1000BASE-T1 PCS will convey an Error symbol in the 80B81B block code. When TX_EN is not asserted and no other supported control code is present at the GMII, the 1000BASE-T1 PCS will convey and Idle symbol in the 80B81B block code.

The control characters and their mappings to 1000BASE-T1 control codes and GMII control codes are specified in Table 97–1. All GMII and 1000BASE-T1 control code values that do not appear in the table shall not be transmitted and shall be treated as an error if received.

Table 97–1—GMII Control Code Mapping

Control Code[0:2]	GMII Transmit	GMII Receive
001	Transmit Error Propagation	Data Reception Error
010	Normal Inter-Frame	Normal Inter-Frame
101	*Assert Low Power Idle*	*Assert Low Power Idle*
other	Reserved	Reserved

97.3.2.2.7 Valid and invalid blocks

A block is invalid if any of the following conditions exists:

a) The block contains an invalid pointer.
b) Any control character contains a value not in Table 97–1.
c) The RS block contains the payload of an uncorrectable received RS frame or the first 80B/81B block following an invalid received RS frame to account for self-synchronizing scrambler error propagation.

97.3.2.2.8 Idle

Idle (Normal Interframe) control characters are transmitted when TX_EN is not asserted and no other supported control code is present at the GMII. Idle characters may be added or deleted by the PCS to adapt between clock rates. They shall not be added while data is being received. When deleting , the first four Idles after a TX_EN is deasserted shall not be deleted.

97.3.2.2.9 Error

The Error is sent when TX_ER is asserted. It is also sent when invalid blocks are received. Error allows physical sublayers such as the PCS to propagate received errors. See R_BLOCK_TYPE and T_BLOCK_TYPE function definitions in 97.3.5.2.4 for further information.

97.3.2.2.10 Transmit process

The transmit process generates blocks based upon the TXD<7:0>, TX_EN and TX_ER signals received from the GMII. Ten GMII data transfers are encoded into each block. It takes 2700 PMA_UNITDATA transfers to send an RS frame of data. Where the GMII and PMA sublayer data rates are not synchronized to that ratio, the transmit process needs to insert idles, or delete idles to adapt between the rates.

The transmit process generates blocks as specified in the transmit process state diagram. The contents of each block are contained in a vector tx_coded<80:0>, which is passed to the scrambler. tx_coded<0> contains the data/ctrl header and the remainder of the bits contain the block payload.

97.3.2.2.11 PCS scrambler.

The payload of the PCS PHY frame is scrambled with a self-synchronizing scrambler. The scrambler for the MASTER shall produce the same result as the implementation shown in Figure 97–9. This implements the scrambler polynomial:[20]

$$G(x) = 1 + x^{39} + x^{58} \quad (55–1)$$

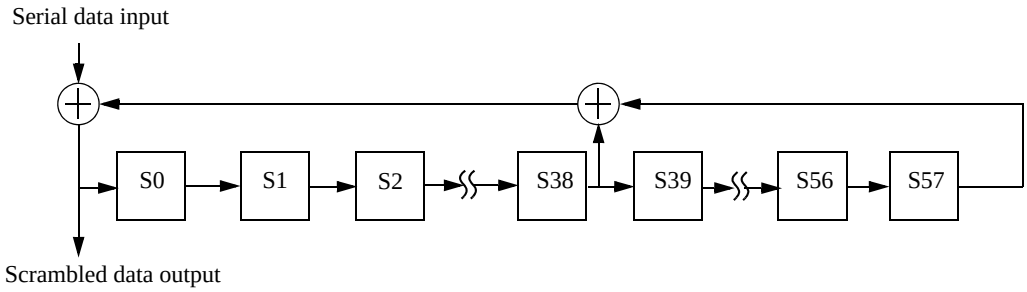
The scrambler for the SLAVE shall produce the same result as the implementation shown in Figure 97–9. This implements the scrambler polynomial:

$$G(x) = 1 + x^{19} + x^{58} \quad (55–2)$$

[20]The convention here, which considers the most recent bit into the scrambler to be the lowest order term, is consistent with most references and with other scramblers shown in this standard. Some references consider the most recent bit into the scrambler to be the highest order term and would therefore identify this as the inverse of the polynomial in Equation (55–1). In case of doubt, note that the conformance requirement is based on the representation of the scrambler in the figure rather than the polynomial equation.

The initial seed values for the MASTER and SLAVE are left to the implementor. The scrambler is run continuously on all payload bits.

PCS scrambler employed by the MASTER



PCS scrambler employed by the SLAVE

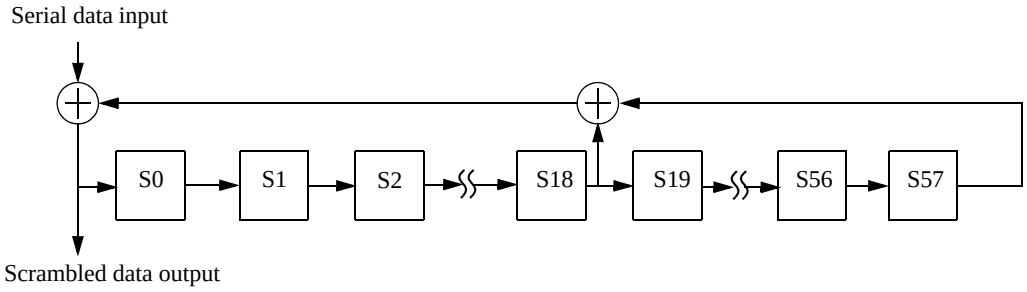


Figure 97–9—MASTER and SLAVE PCS scramblers

97.3.2.2.12 RS encoder

The resulting payload of scrambled 45 81B blocks, followed by the corresponding 9 reserved bits (*Tentatively 111 101 010* listed MSB to LSB) results in a total payload of $45 \times 81 + 9 = 3646$ bits.

The encoder shall follow the notation described in 97.3.2.2.3 where the LSB is the first bit into the RS encoder and the first transmitted bit.

The group of 406 9-bit symbols are encoded using a Reed-Solomon encoder operating over the Galois Field GF(2^9) where the symbol size is 9 bits. The encoder process 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, the particular Reed-Solomon code is denoted RS(n,k) where n = 450 and k = 406.

The resulting RS block size is 450 9-bit symbols, a total of 4050 bits.

97.3.2.2.13 3B2T to PAM3

Every 9-bit symbol is divided into three 3-bit groups with the LSB bits as the first group. Each 3-bit group

Table 97–2—3B2T Mapping to PAM3

B[2], B[1], B[0]	T[1], T[0]
000	-1,-1
001	0,-1
010	-1,0
011	-1,+1
100	+1,0
101	+1,-1
110	+1,+1
111	0,+1

then mapped by the 3B2T into 2 PAM3 symbols.The mapping of 3B2T to PAM3 is illustrated in Table 97–2. B[0] is the LSB and T[0] is the first PAM3 symbol transmitted.

The 3B2T mapper generates 2700 PAM3 symbols per RS frame that are sent to the PMA via PMA_UNITDATA.request. The mapping of 3B2T to PAM3 is illustrated in Table 97–2.

97.3.2.2.14 81B-RS framer

The 81B-RS framer adapts between the 81-bit width of the 81B blocks and the PAM3 input to the PMA. When the transmit channel is operating in normal mode, the 81B-RS sends one PAM3 symbol of transmit data at a time via PMA_UNITDATA.request primitives. The PMA_UNITDATA.request primitives are fully packed with bits.

97.3.2.3 PCS Receive function

The PCS Receive function shall conform to the PCS 80B/81B receive state diagram in Figure 97–14 and the PCS Receive bit ordering in Figure 97–7 including compliance with the associated state variables as specified in 97.3.5.

The PCS Receive function accepts received code-groups provided by the PMA Receive function via the parameter rx_symb_vector. The PCS receiver uses knowledge of the encoding rules to correctly align the 81B-RS frames. The received 81B-RS frames are decoded with error correction; the framing is checked; and the 80B/81B ordered sets are converted to 10 data blocks to obtain the signals RXD<7:0>, RX_DV and RX_ER for transmission to the GMII.

During PMA training mode, PCS Receive checks the received PAM2 framing and signals the reliable acquisition of the descrambler state by setting the parameter scr_status to OK.

When the PCS Synchronization process has obtained synchronization, the RS frame error rate (RFER) monitor process monitors the signal quality asserting hi_rfer if excessive RS frame errors are detected (RS parity error). If 40 consecutive RS frame errors are detected, the block_lock flag is de-asserted. When block_lock is asserted and hi_rfer is de-asserted, the PCS Receive process continuously accepts blocks. The PCS Receive process monitors these blocks and generates RXD<7:0>, RX_DV and RX_ER for transmission to the GMII.

When the receive channel is in training mode, the PCS Synchronization process continuously monitors PMA_RXSTATUS.indication (loc_rcvr_status). When loc_rcvr_status indicates OK, then the PCS Synchronization process accepts data-units via the PMA_UNITDATA.request primitive. It attains frame and block synchronization based on the PMA training frames and conveys received blocks to the PCS Receive process. The PCS Synchronization process sets the block_lock flag to indicate whether the PCS has obtained synchronization. *The PMA training sequence includes 1 bit pattern on pair A every TBD PAM2 symbols, which is aligned with the PCS Partial RS frame boundary.* When the PCS Synchronization process is synchronized to this pattern, block_lock is asserted.

97.3.2.3.1 Frame and block synchronization

When the receive channel is operating in normal mode, the frame and block synchronization function receives data via PAM3 PMA_UNITDATA.request primitive. It shall form a PAM3 stream from the primitive by concatenating requests in order from rx_data<0> to rx_data<2699> (see Figure 97–7). It obtains block_lock to the RS frames during the PAM2 training pattern using synchronization bits provided in the training sequence. The 81-bit blocks are extracted based on their location in the RS frame.

97.3.2.3.2 PCS 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 implementations shown in Figure 97–10 for the MASTER and the SLAVE.

97.3.3 Test-pattern generators

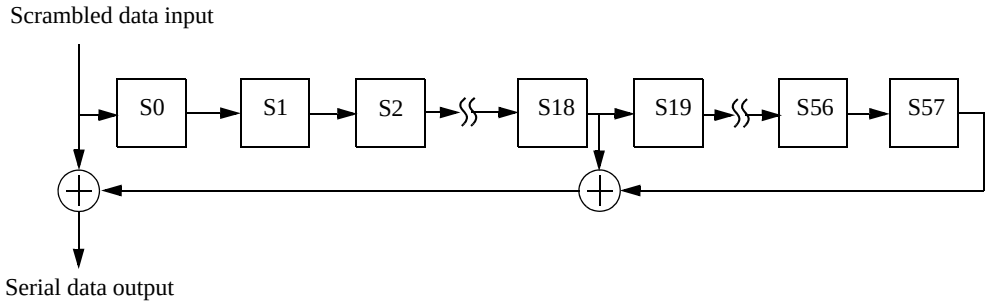
The test-pattern generator mode is provided for enabling joint testing of the local transmitter, the channel and remote receiver. When the transmit PCS is operating in test-pattern mode it shall transmit continuously as illustrated in Figure 97–6, with the input to the scrambler set to zero and the initial condition of the scrambler set to any non-zero value. When the receiver PCS is operating in test-pattern mode it shall receive continuously as illustrated in Figure 97–7. After acquiring the self-synchronizing scrambler state, the output of the received scrambled values should ideally be zero. Any nonzero values correspond to receiver bit errors. This mode is further described as test mode 7 in 97.5.2

97.3.4 PMA training side-stream scrambler polynomials

NOTE: The training scrambler has not been selected.

The PCS Transmit function employs side-stream scrambling for generating 2-level PAM PMA training sequences as shown in Figure 97–11. An implementation of MASTER and SLAVE PHY side-stream scramblers is shown in the "Main PN sequence" box. The bits stored in the shift register delay line at time n are denoted by $Scr_n[32:0]$. At each symbol period, the shift register is advanced by one bit, and one new bit represented by $Scr_n[0]$ is generated. The transmitter side-stream scrambler is reset upon execution of the PCS Reset function. If PCS Reset is executed, all bits of the 33-bit vector representing the side-stream scrambler state are arbitrarily set. The initialization of the scrambler state is left to the implementor. In no case shall the scrambler state be initialized to all zeros.

PCS descrambler employed by the MASTER



PCS descrambler employed by the SLAVE

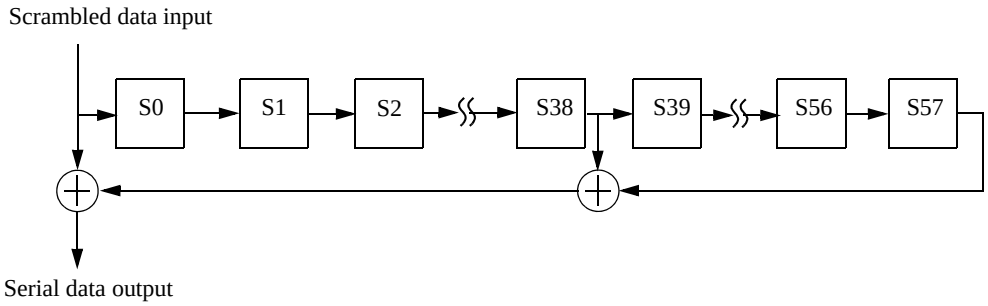


Figure 97–10—MASTER and SLAVE PCS descramblers

Figure 97–11—A realization of PMA training PAM2 sequences

97.3.4.1 Generation of Sa_n

PMA training signal encoding rules are based on the generation, at time n, of the bit Sa_n. The bit is generated in a systematic fashion using the bits in $Scr_n[32:0]$. For both MASTER and SLAVE PHYs, they are obtained by the same linear combinations of bits stored in the transmit scrambler shift register delay line. The bit is generated using the bit $Scr_n[0]$ and the equations in Figure 97–11 in the "Derived sequences" box.

97.3.4.2 Generation of symbol TA_n

The bit Sa_n is mapped to symbol TA_n as shown in Figure 97–11.

NOTE: Training patterns, training scrambler, sync bit, and info field have not been accepted as baseline.

The inversion the pair at 256 intervals ($n = k \times 256, k = 0, 1, 2, \ldots$) defines the RS boundary during data mode. If requested by the link partner, the PCS shall reset the training mode scrambler every 16384 periods aligned with the 256 symbol period inversion on pair A, which corresponds to the time instants $n = m \times 16384, m = 0, 1, 2, \ldots$

Notice that over the repeating time intervals of 16384 and of length 128, $m \times 16384 - 128 \le n < m \times 16384, m = 1, 2, 3, \ldots$, the PMA training pattern in pair A is XOR'ed with the InfoField. Thus, pair A transmits the InfoField, which communicates to the remote transceiver settings of THP and power backoff and other control information.

97.3.4.3 PMA training mode descrambler polynomials

The PHY shall acquire descrambler state synchronization to the PAM2 training sequence and report success through scr_status. For side-stream descrambling, the MASTER PHY shall employ the receiver descrambler generator polynomial $g'_M(x) = 1 + x^{20} + x^{33}$ and the SLAVE PHY shall employ the receiver descrambler generator polynomial $g'_S(x) = 1 + x^{13} + x^{33}$.

97.3.5 Detailed functions and state diagrams

97.3.5.1 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.

97.3.5.2 State diagram parameters

97.3.5.2.1 Constants

EBLOCK_R<99:0>
100 bit vector to be sent to the GMII containing symbol errors in all 10 character locations.

IBLOCK_R<99:0>
100 bit vector to be sent to the GMII containing idles in all 10 character locations.

IBLOCK_T<99:0>
100 bit vector to be sent to the encoder containing idles in all 10 character locations.

RFER_CNT_LIMIT
TBD. Number of Reed Solomon frames with uncorrectable errors.

RFRX_CNT_LIMIT
TBD. Number of Reed Solomon frames received over bit error rate interval.

97.3.5.2.2 Variables

RFER_test_lf
Boolean variable that is set true when a new RS frame is available for testing and false when

RFER_TEST_LF state is entered. A new RS frame is available for testing when the Block Sync process has accumulated enough symbols from the PMA to evaluate the next RS frame.

block_lock
: Boolean variable that is set true when receiver acquires block delineation.

hi_rfer
: Boolean variable which is asserted true when the rfer_cnt exceeds RFER_CNT_LIMIT indicating a bit error ratio > 4×10^{-4}.

pcs_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.

rx_coded<81:0>
: Vector containing the input to the 80B/81B decoder including a block valid flag. The format for rx_coded<80:0> is shown in Figure 97–7. The leftmost bit in the figure is rx_coded<0> and the rightmost bit is rx_coded<80>. rx_coded<81> (not shown in the figure) is set to 1 if all parity checks of the Reed Solomon frame are satisfied, otherwise it is set to 0.

rx_raw<99:0>
: Vector containing 10 successive GMII output transfers. Each transfer is numbered from 0 to 9 with the first transfer numbered as the 0th transfer. The nth GMII transfer is labeled as RX_DV[n], RX_ER[n], RXD[n][7:0]. For n = 0 to 9, rx_raw<8n> = RX_DV[n], rx_raw<8n+1> = RX_ER[n], rx_raw<8n+9:8n+2> = RXD[n][7:0]

rf_valid
: Boolean indication that is set true if received Reed Solomon frame is valid. The frame is valid if all parity checks of the coded bits are satisfied.

tx_coded<80:0>
: Vector containing the output from the 80B/81B encoder. The format for this vector is shown in Figure 97–9. The leftmost bit in the figure is tx_coded<0> and the rightmost bit is tx_coded<80>.

tx_raw<99:0>
: Vector containing 10 successive GMII transfers. Each transfer is numbered from 0 to 9 with the first transfer numbered as the 0th transfer. The nth GMII transfer is labeled as TX_EN[n], TX_ER[n], TXD[n][7:0]. For n = 0 to 9, tx_raw<8n> = TX_EN[n], tx_raw<8n+1> = TX_ER[n], tx_raw<8n+9:8n+2> = TXD[n][7:0]

tx_data_mode
: Set true when tx_mode = SEND_N, otherwise false.

97.3.5.2.3 Timers

State diagram timers follow the conventions described in 14.2.3.2.

97.3.5.2.4 Functions

DECODE(rx_symb_vector<81:0>)
: In the PCS Receive process, this function takes as its argument 82-bit rx_coded<81:0> from the Reed Solomon decoder and descrambler. If rx_coded<81> = 1 then the decoder decodes the 81B-Reed Solomon bit vector rx_coded<80:0> returning a vector rx_raw<99:0>, which is sent to the GMII. The DECODE function shall decode the block based on code specified in 97.3.2.2.2. If rx_coded<81> = 0 then the decoder returns EBLOCK_R.

ENCODE(tx_raw<99:0>)
: Encodes the 100-bit vector received from the GMII, returning 81-bit vector tx_coded. The ENCODE function shall encode the block as specified in 97.3.2.2.2.

97.3.5.2.5 Counters

rfer_cnt
: Count up to a maximum of RFER_CNT_LIMIT of the number of invalid Reed Solomon frames within the current RFRX_CNT_LIMIT Reed Solomon frame period.

rfrx_cnt
: Count number Reed Solomon frames received during current period.

97.3.5.3 Messages

PMA_UNITDATA.indication (rx_symb_vector)
: A signal sent by PMA Receive indicating that a PAM3 symbol is available in rx_symb_vector.

PMA_UNITDATA.request (tx_symb_vector)
: A signal sent to PMA Transmit indicating that a PAM3 symbol is available in tx_symb_vector.

PCS_status
: Indicates whether the PCS is in a fully operational state. (See 97.3.6.1.)

RX_AGGREGATE
: A signal sent to PCS Receive indicating that 9 aligned 9-bit Reed Solomon symbols are aggregated in rx_coded<80:0>.

TX_AGGREGATE
: A signal sent to PCS Transmit indicating that 10 GMII transfers are aggregated in tx_raw<99:0>.

RX_FRAME
: A signal sent to PCS Receive indicating that a full Reed Solomon frame has been decoded and the variable rf_valid is updated.

TX_FRAME
: A signal sent to PCS Transmit indicating that a full Reed Solomon frame has been transmitted.

97.3.5.4 State diagrams

The RFER Monitor state diagram shown in Figure 97–12 monitors the received signal for high RS frame error ratio.

The 80B/81B Transmit state diagram shown in Figure 97–13 controls the encoding of 81B transmitted blocks. It makes exactly one transition for each 81B transmit block processed.

The 80B/81B Receive state diagram shown in Figure 97–14 controls the decoding of 81B received blocks. It makes exactly one transition for each receive block processed.

The PCS shall perform the functions of RFER Monitor, Transmit, and Receive as specified in these state diagrams.

97.3.6 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.

97.3.6.1 Status

PCS_status:
: Indicates whether the PCS is in a fully operational state. It is only true if block_lock is true and hi_rfer 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_rfer:

Indicates the state of the hi_rfer variable. This status is reflected in MDIO register 3.32.1. A latch high view of this status is reflected in MDIO register 3.33.14.

97.3.6.2 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.

RFER_count:

6-bit counter that counts each time RFER_BAD_RF state is entered. This counter is reflected in MDIO register bits 3.33.13:8. The counter is reset when register 3.33 is read by management. Note that this counter counts a maximum of RFER_CNT_LIMIT counts per RFRX_CNT_LIMIT

period since the RFER_BAD_RF can be entered a maximum of RFER_CNT_LIMIT times per RFRX_CNT_LIMIT window.

.

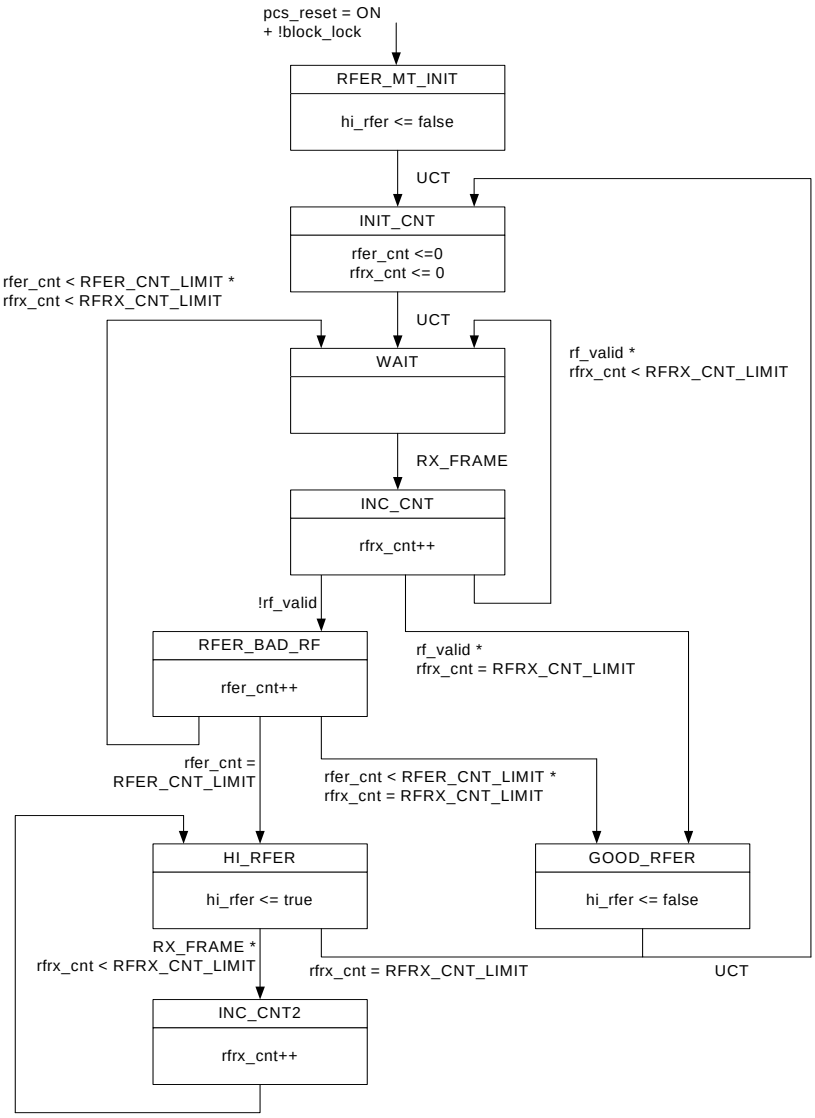


Figure 97–12—RFER monitor state diagram

97.3.6.3 Loopback

The PCS shall be placed in loopback mode when the loopback bit in MDIO register 3.0.14 is set to a one. In this mode, the PCS shall accept data on the transmit path from the GMII and return it on the receive path to the GMII. In addition, the PCS shall transmit a continuous stream of GMII to 81B-RS encoded PAM3 symbols to the PMA sublayer, and shall ignore all data presented to it by the PMA sublayer.

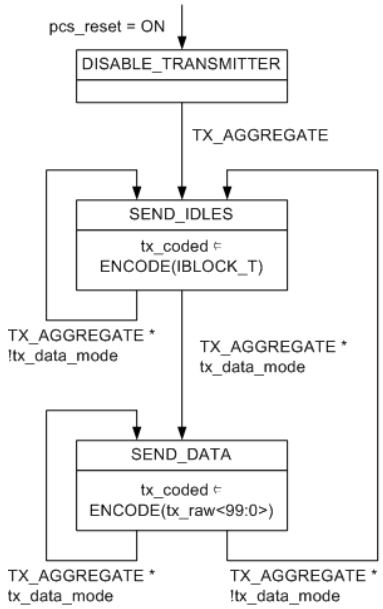


Figure 97–13—PCS Transmit state diagram

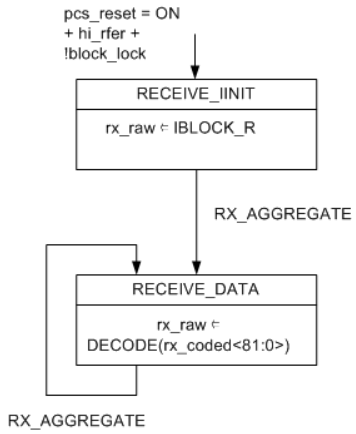


Figure 97–14—PCS Receive state diagram