40. Physical Coding Sublayer (PCS), Physical Medium Attachment (PMA) sublayer and baseband medium, type 1000BASE-T

40.1 Overview

The 1000BASE-T PHY is one of the Gigabit Ethernet family of high-speed CSMA/CD network specifications. The 1000BASE-T Physical Coding Sublayer (PCS), Physical Medium Attachment (PMA) and baseband medium specifications are intended for users who want 1000 Mb/s performance over Category 5 balanced twisted-pair cabling systems. 1000BASE-T signaling requires four pairs of balanced cabling, as specified in ISO/IEC 11801:1995 (Class D) and ANSI/EIA/TIA-568-A-1995 (Category 5), and tested for the additional performance parameters specified in ANSI/EIA/TIA-568-B1 Annex D.

NOTE—ISO/IEC 11801:2002 provides a specification (Class D) for media that exceeds the minimum requirements of this standard.

This clause defines the type 1000BASE-T PCS, type 1000BASE-T PMA sublayer, and type 1000BASE-T Medium Dependent Interface (MDI). Together, the PCS and the PMA sublayer comprise a 1000BASE-T Physical Layer (PHY). Provided in this document are fully functional, electrical, and mechanical specifications for the type 1000BASE-T PCS, PMA, and MDI. This clause also specifies the baseband medium used with 1000BASE-T.

40.1.1 Objectives

The following are the objectives of 1000BASE-T:

a) Support the CSMA/CD MAC
b) Comply with the specifications for the GMII (Clause 35)
c) Support the 1000 Mb/s repeater (Clause 41)
d) Provide line transmission that supports full and half duplex operation
e) Meet or exceed FCC Class A/CISPR or better operation
f) Support operation over 100 meters of copper balanced cabling as defined in 40.7
g) Bit Error Ratio of less than or equal to 10^{-10}
h) Support Auto-Negotiation (Clause 28)

40.1.2 Relationship of 1000BASE-T to other standards

Relations between the 1000BASE-T PHY, the ISO Open Systems Interconnection (OSI) Reference Model, and the IEEE 802.3 CSMA/CD LAN Model are shown in Figure 40–1. The PHY sublayers (shown shaded) in Figure 40–1 connect one Clause 4 Media Access Control (MAC) layer to the medium.

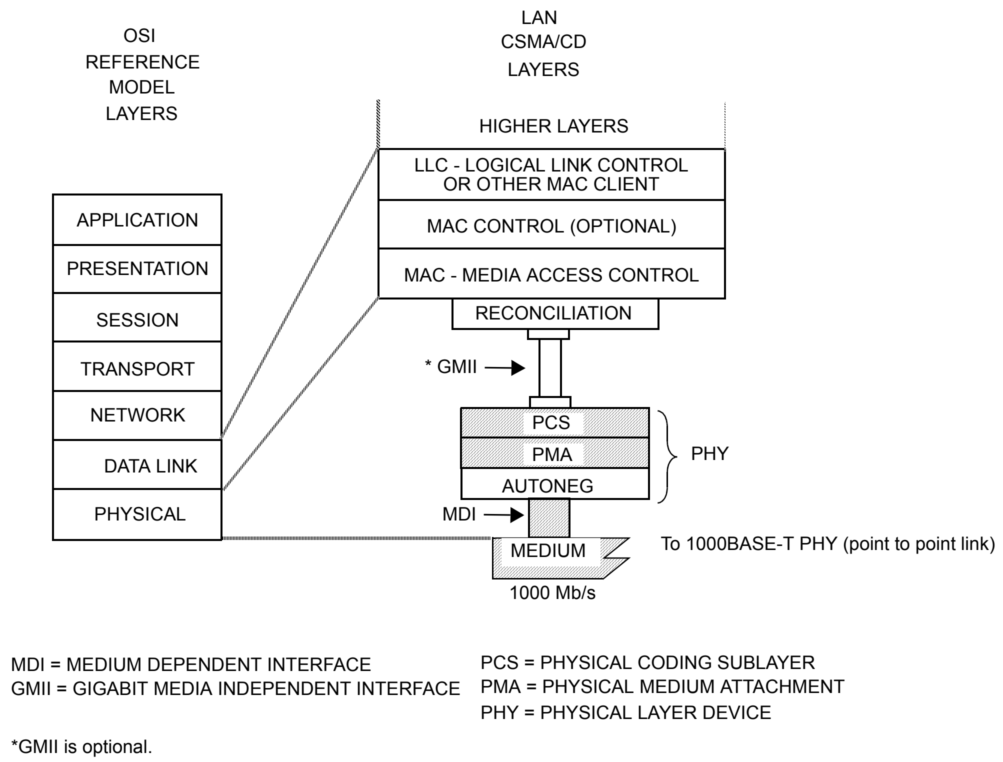

OSI REFERENCE MODEL LAYERS
LAN CSMA/CD LAYERS
APPLICATION
PRESENTATION
SESSION
TRANSPORT
NETWORK
DATA LINK
PHYSICAL
HIGHER LAYERS
LLC - LOGICAL LINK CONTROL OR OTHER MAC CLIENT
MAC CONTROL (OPTIONAL)
MAC - MEDIA ACCESS CONTROL
RECONCILIATION
* GMII
PCS
PMA
AUTONEG
PHY
MDI
MEDIUM
To 1000BASE-T PHY (point to point link)
1000 Mb/s
MDI = MEDIUM DEPENDENT INTERFACE
GMII = GIGABIT MEDIA INDEPENDENT INTERFACE
PCS = PHYSICAL CODING SUBLAYER
PMA = PHYSICAL MEDIUM ATTACHMENT
PHY = PHYSICAL LAYER DEVICE
*GMII is optional.

Figure 40–1—Type 1000BASE-T PHY relationship to the ISO Open Systems Interconnection (OSI) Reference Model and the IEEE 802.3 CSMA/CD LAN Model

40.1.3 Operation of 1000BASE-T

The 1000BASE-T PHY employs full duplex baseband transmission over four pairs of Category 5 balanced cabling. The aggregate data rate of 1000 Mb/s is achieved by transmission at a data rate of 250 Mb/s over each wire pair, as shown in Figure 40–2. The use of hybrids and cancellers enables full duplex transmission by allowing symbols to be transmitted and received on the same wire pairs at the same time. Baseband signaling with a modulation rate of 125 MBd is used on each of the wire pairs. The transmitted symbols are selected from a four-dimensional 5-level symbol constellation. Each four-dimensional symbol can be viewed as a 4-tuple (A_n, B_n, C_n, D_n) of one-dimensional quinary symbols taken from the set {2, 1, 0, –1, –2}. 1000BASE-T uses a continuous signaling system; in the absence of data, Idle symbols are transmitted. Idle mode is a subset of code-groups in that each symbol is restricted to the set {2, 0, –2}to improve synchronization. Five-level Pulse Amplitude Modulation (PAM5) is employed for transmission over each wire pair. The modulation rate of 125 MBd matches the GMII clock rate of 125 MHz and results in a symbol period of 8 ns.

A 1000BASE-T PHY can be configured either as a MASTER PHY or as a SLAVE PHY. The MASTER-SLAVE relationship between two stations sharing a link segment is established during Auto-Negotiation (see Clause 28, 40.5, and Annex 28C). The MASTER PHY uses a local clock to determine the timing of transmitter operations. The SLAVE PHY recovers the clock from the received signal and uses it to determine the timing of transmitter operations, i.e., it performs loop timing, as illustrated in Figure 40–3. In a multiport to single-port connection, the multiport device is typically set to be MASTER and the single-port device is set to be SLAVE.

The PCS and PMA subclauses of this document are summarized in 40.1.3.1 and 40.1.3.2. Figure 40–3 shows the functional block diagram.

A 1000BASE-T PHY with the optional Energy-Efficient Ethernet (EEE) capability may optionally enter the Low Power Idle (LPI) mode to conserve energy during periods of low link utilization. The “Assert LPI” request at the GMII is encoded in the transmitted symbols. Detection of LPI signaling in the received symbols is indicated as “Assert LPI” at the GMII. When LPI signaling is simultaneously transmitted and received, an energy-efficient 1000BASE-T PHY ceases transmission and deactivates transmit and receive functions to conserve energy. The PHY periodically transmits during this quiet period to allow the remote PHY to refresh its receiver state (e.g., timing recovery, adaptive filter coefficients) and thereby track long-term variation in the timing of the link or the underlying channel characteristics. If, during the quiet or refresh periods, normal interframe is asserted at the GMII, the PHY reactivates transmit and receive functions and initiates transmission. This transmission will be detected by the remote PHY, causing it to also exit the LPI mode.

The conditions for supporting the optional EEE capability are defined in 78.3.

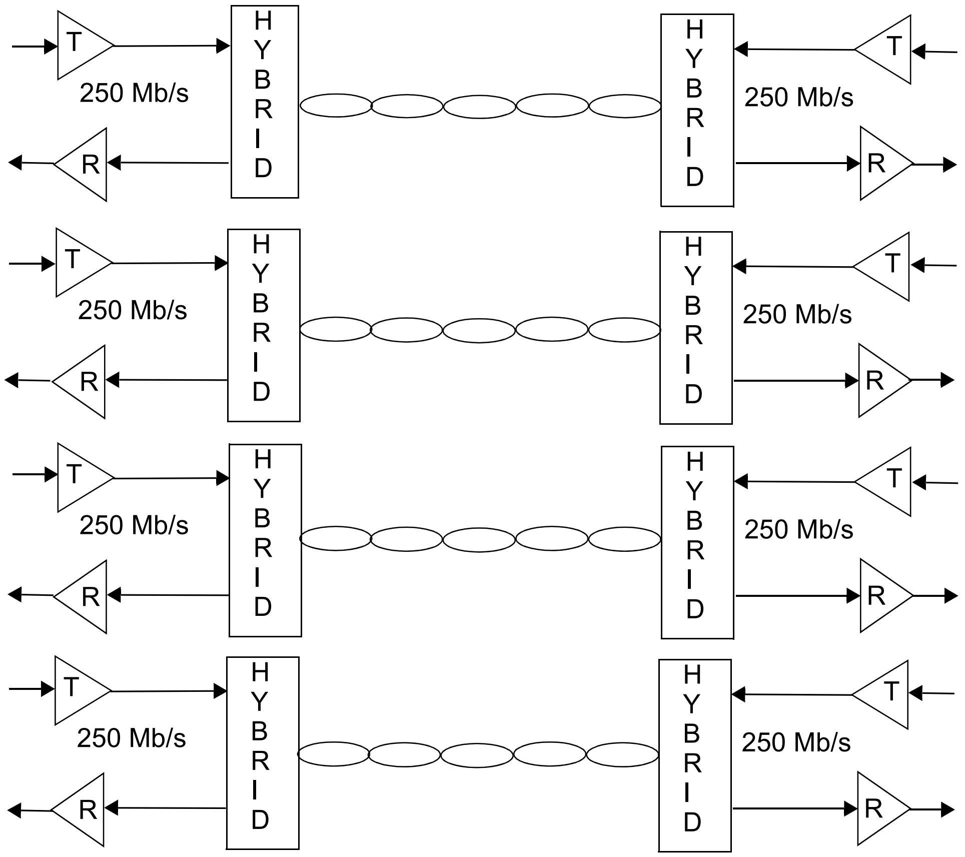
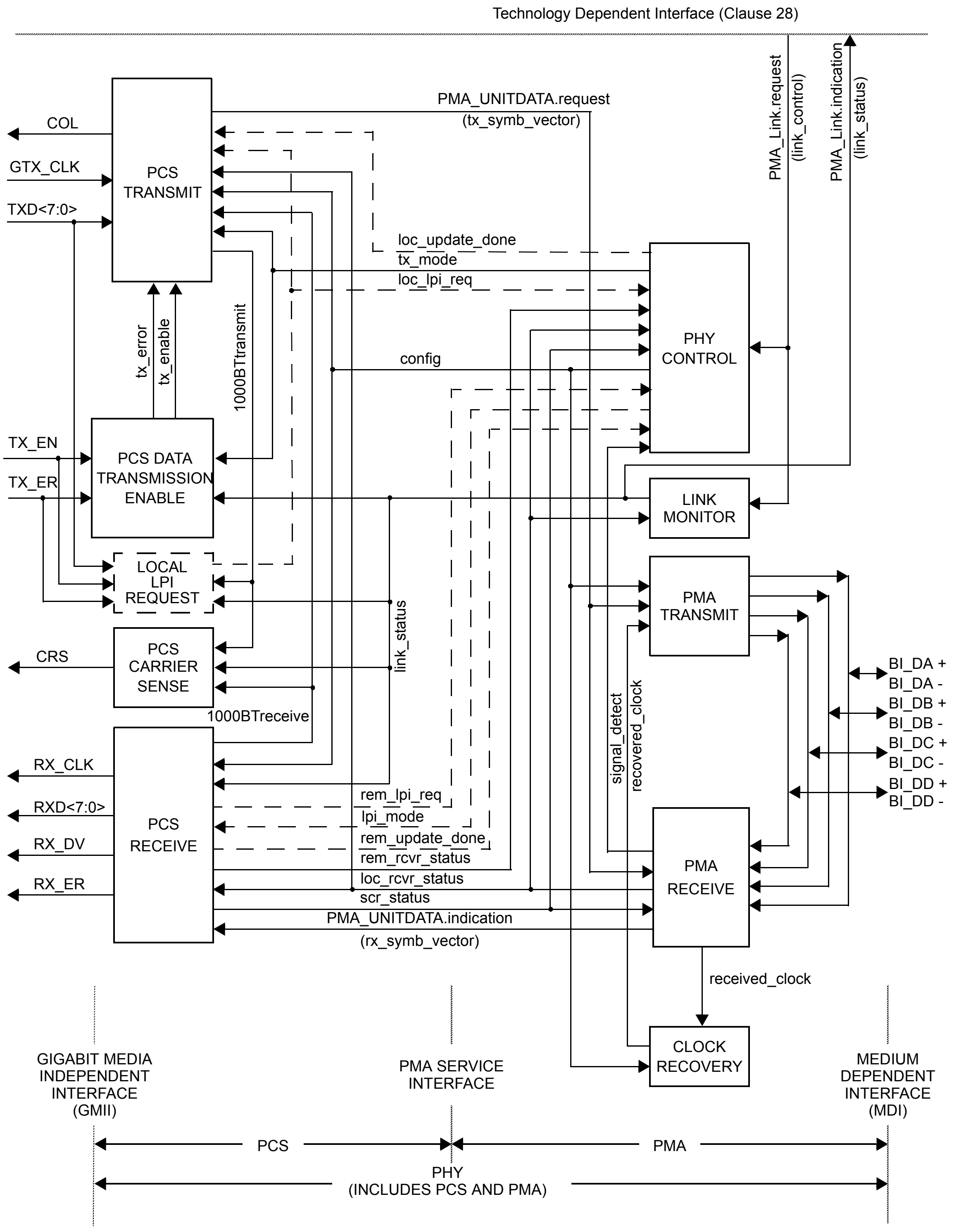


Figure 40–2—1000BASE-T topology



NOTE—The recovered_clock arc is shown to indicate delivery of the received clock signal back the PMA TRANSMIT for loop timing

NOTE—Signals and functions shown with dashed lines are only required for the EEE capability.

Figure 40–3—Functional block diagram

40.1.3.1 Physical Coding Sublayer (PCS)

The 1000BASE-T PCS couples a Gigabit Media Independent Interface (GMII), as described in Clause 35, to a Physical Medium Attachment (PMA) sublayer.

The functions performed by the PCS comprise the generation of continuous code-groups to be transmitted over four channels and the processing of code-groups received from the remote PHY. The process of converting data bits to code-groups is called 4D-PAM5, which refers to the four-dimensional 5-level Pulse Amplitude Modulation coding technique used. Through this coding scheme, eight bits are converted to one transmission of four quinary symbols.

During the beginning of a frame's transmission, when TX_EN is asserted from the GMII, two code-groups representing the Start-of-Stream delimiter are transmitted followed by code-groups representing the octets coming from the GMII. Immediately following the data octets, the GMII sets TX_EN=FALSE, upon which the end of a frame is transmitted. The end of a frame consists of two convolutional state reset symbol periods and two End-of-Stream delimiter symbol periods. This is followed by an optional series of carrier extend symbol periods and, possibly, the start of a new frame during frame bursting. Otherwise, the end of a frame is followed by a series of symbols encoded in the idle mode. The nature of the encoding that follows the end of a frame is determined by the GMII signals TX_ER and TXD<7:0> as specified in Clause 35.

Between frames, a special subset of code-groups using only the symbols {2, 0, –2} is transmitted. This is called idle mode. Idle mode encoding takes into account the information of whether the local PHY is operating reliably or not (see 40.4.2.4) and allows this information to be conveyed to the remote station. During normal operation, idle mode is followed by a data mode that begins with a Start-of-Stream delimiter.

When the PHY supports the optional EEE capability, Idle mode encoding also conveys to the remote PHY information of whether the local PHY is requesting entry into the LPI mode or not. Such requests are a direct translation of "Assert LPI" at the GMII. In addition, Idle mode encoding conveys to the remote PHY whether the local PHY has completed the update of its receiver state or not, as indicated by the PMA PHY Control function.

Further patterns are used for signaling a transmit error and other control functions during transmission of a data stream.

The PCS Receive processes code-groups provided by the PMA. The PCS Receive detects the beginning and the end of frames of data and, during the reception of data, descrambles and decodes the received code-groups into octets RXD<7:0> that are passed to the GMII. The conversion of code-groups to octets uses an 8B1Q4 data decoding technique. PCS Receive also detects errors in the received sequences and signals them to the GMII. Furthermore, the PCS contains a PCS Carrier Sense function, a PCS Collision Presence function, and a management interface.

The PCS functions and state diagrams are specified in 40.3. The signals provided by the PCS at the GMII conform to the interface requirements of Clause 35. The PCS Service Interfaces to the GMII and the PMA are abstract message-passing interfaces specified in 40.2.

40.1.3.2 Physical Medium Attachment (PMA) sublayer

The PMA couples messages from the PMA service interface onto the balanced cabling physical medium and provides the link management and PHY Control functions. The PMA provides full duplex communications at 125 MBd over four pairs of balanced cabling up to 100 m in length.

The PMA Transmit function comprises four independent transmitters to generate five-level, pulse-amplitude modulated signals on each of the four pairs BI_DA, BI_DB, BI_DC, and BI_DD, as described in 40.4.3.1.

The PMA Receive function comprises four independent receivers for five-level pulse-amplitude modulated signals on each of the four pairs BI_DA, BI_DB, BI_DC, and BI_DD, as described in 40.4.3.2. This signal encoding technique is referred to as 4D-PAM5. The receivers are responsible for acquiring clock and providing code-groups to the PCS as defined by the PMA_UNITDATA.indication message. The PMA also contains functions for Link Monitor.

The PMA PHY Control function generates signals that control the PCS and PMA sublayer operations. PHY Control begins following the completion of Auto-Negotiation and provides the start-up functions required for successful 1000BASE-T operation. It determines whether the PHY operates in a normal state, enabling data transmission over the link segment, or whether the PHY sends special code-groups that represent the idle mode. The latter occurs when either one or both of the PHYs that share a link segment are not operating reliably.

When the PHY supports the optional EEE capability, the PMA PHY Control function also coordinates transitions between the LPI mode and the normal operating mode.

PMA functions and state diagrams are specified in 40.4. PMA electrical specifications are given in 40.6.

40.1.4 Signaling

1000BASE-T signaling is performed by the PCS generating continuous code-group sequences that the PMA transmits over each wire pair. The signaling scheme achieves a number of objectives including

a) Forward Error Correction (FEC) coded symbol mapping for data.
b) Algorithmic mapping and inverse mapping from octet data to a quartet of quinary symbols and back.
c) Uncorrelated symbols in the transmitted symbol stream.
d) No correlation between symbol streams traveling both directions on any pair combination.
e) No correlation between symbol streams on pairs BI_DA, BI_DB, BI_DC, and BI_DD.
f) Idle mode uses a subset of code-groups in that each symbol is restricted to the set {2, 0, –2}to ease synchronization, start-up, and retraining.
g) Ability to rapidly or immediately determine if a symbol stream represents data or idle or carrier extension.
h) Robust delimiters for Start-of-Stream delimiter (SSD), End-of-Stream delimiter (ESD), and other control signals.
i) Ability to signal the status of the local receiver to the remote PHY to indicate that the local receiver is not operating reliably and requires retraining.
j) Optionally, ability to signal to the remote PHY a request to enter the LPI mode and to exit the LPI mode and return to normal operation.
k) Optionally, ability to signal to the remote PHY that the update of the local receiver state (e.g., timing recovery, adaptive filter coefficients) has completed.
l) Ability to automatically detect and correct for pair swapping and unexpected crossover connections.
m) Ability to automatically detect and correct for incorrect polarity in the connections.
n) Ability to automatically correct for differential delay variations across the wire-pairs.

The PHY may operate in three basic modes, normal mode, training mode, or an optional LPI mode. In normal mode, PCS generates code-groups that represent data, control, or idles for transmission by the PMA. In training mode, the PCS is directed to generate only idle code-groups for transmission by the PMA, which enable the receiver at the other end to train until it is ready to operate in normal mode. In LPI mode, the PCS is directed to generate only idle code-groups encoded with LPI request and update status indications, or zeros as dictated by the PMA PHY Control function. (See the PCS reference diagram in 40.2.)

40.1.5 Inter-sublayer interfaces

All implementations of the balanced cabling link are compatible at the MDI. Designers are free to implement circuitry within the PCS and PMA in an application-dependent manner provided that the MDI and GMII (if the GMII is implemented) specifications are met. When the PHY is incorporated within the physical bounds of a single-port device or a multiport device, implementation of the GMII is optional. System operation from the perspective of signals at the MDI and management objects are identical whether the GMII is implemented or not.

40.1.6 Conventions in this clause

The body of this clause contains state diagrams, including 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.

The values of all components in test circuits shall be accurate to within ±1% unless otherwise stated.

Default initializations, unless specifically specified, are left to the implementer.

40.2 1000BASE-T Service Primitives and Interfaces

1000BASE-T transfers data and control information across the following four service interfaces:

a) Gigabit Media Independent Interface (GMII)
b) PMA Service Interface
c) Medium Dependent Interface (MDI)
d) Technology-Dependent Interface

The GMII is specified in Clause 35; the Technology-Dependent Interface is specified in Clause 28. The PMA Service Interface is defined in 40.2.2 and the MDI is defined in 40.8.

40.2.1 Technology-Dependent Interface

1000BASE-T uses the following service primitives to exchange status indications and control signals across the Technology-Dependent Interface as specified in Clause 28:

PMA_LINK.request (link_control)

PMA_LINK.indication (link_status)

40.2.1.1 PMA_LINK.request

This primitive allows the Auto-Negotiation algorithm to enable and disable operation of the PMA as specified in 28.2.6.2.

40.2.1.1.1 Semantics of the primitive

PMA_LINK.request (link_control)

The link_control parameter can take on one of three values: SCAN_FOR_CARRIER, DISABLE, or ENABLE.

SCAN_FOR_CARRIER	Used by the Auto-Negotiation algorithm prior to receiving any fast link pulses. During this mode the PMA reports link_status=FAIL.PHY processes are disabled.
DISABLE	Set by the Auto-Negotiation algorithm in the event fast link pulses are detected. PHY processes are disabled. This allows the Auto-Negotiation algorithm to determine how to configure the link.
ENABLE	Used by Auto-Negotiation to turn control over to the PHY for data processing functions.

40.2.1.1.2 When generated

Auto-Negotiation generates this primitive to indicate a change in link_control as described in Clause 28.

40.2.1.1.3 Effect of receipt

This primitive affects operation of the PMA Link Monitor function as defined in 40.4.2.5.

40.2.1.2 PMA_LINK.indication

This primitive is generated by the PMA to indicate the status of the underlying medium as specified in 28.2.6.1. This primitive informs the PCS, PMA PHY Control function, and the Auto-Negotiation algorithm about the status of the underlying link.

40.2.1.2.1 Semantics of the primitive

PMA_LINK.indication (link_status)

The link_status parameter can take on one of three values: FAIL, READY, or OK.

FAIL	No valid link established.
READY	The Link Monitor function indicates that a 1000BASE-T link is intact and ready to be established.
OK	The Link Monitor function indicates that a valid 1000BASE-T link is established. Reliable reception of signals transmitted from the remote PHY is possible.

40.2.1.2.2 When generated

The PMA generates this primitive continuously to indicate the value of link_status in compliance with the state diagram given in Figure 40–17.

40.2.1.2.3 Effect of receipt

The effect of receipt of this primitive is specified in 40.3.3.1.

40.2.2 PMA Service Interface

1000BASE-T uses the following service primitives to exchange symbol vectors, status indications, and control signals across the service interfaces:

PMA_TXMODE.indication (tx_mode)

PMA_CONFIG.indication (config)

PMA_UNITDATA.request (tx_symb_vector)

PMA_UNITDATA.indication (rx_symb_vector)

PMA_SCRSTATUS.request (scr_status)

PMA_RXSTATUS.indication (loc_rcvr_status)

PMA_REMRXSTATUS.request (rem_rcvr_status)

PMA_LPIMODE.indication(lpi_mode)

PMA_LPIREQ.request(loc_lpi_req)

PMA_REMLPIREQ.request(rem_lpi_req)

PMA_UPDATE.indication(loc_update_done)

PMA_REMUPDATE.request(rem_update_done)

The use of these primitives is illustrated in Figure 40–4.

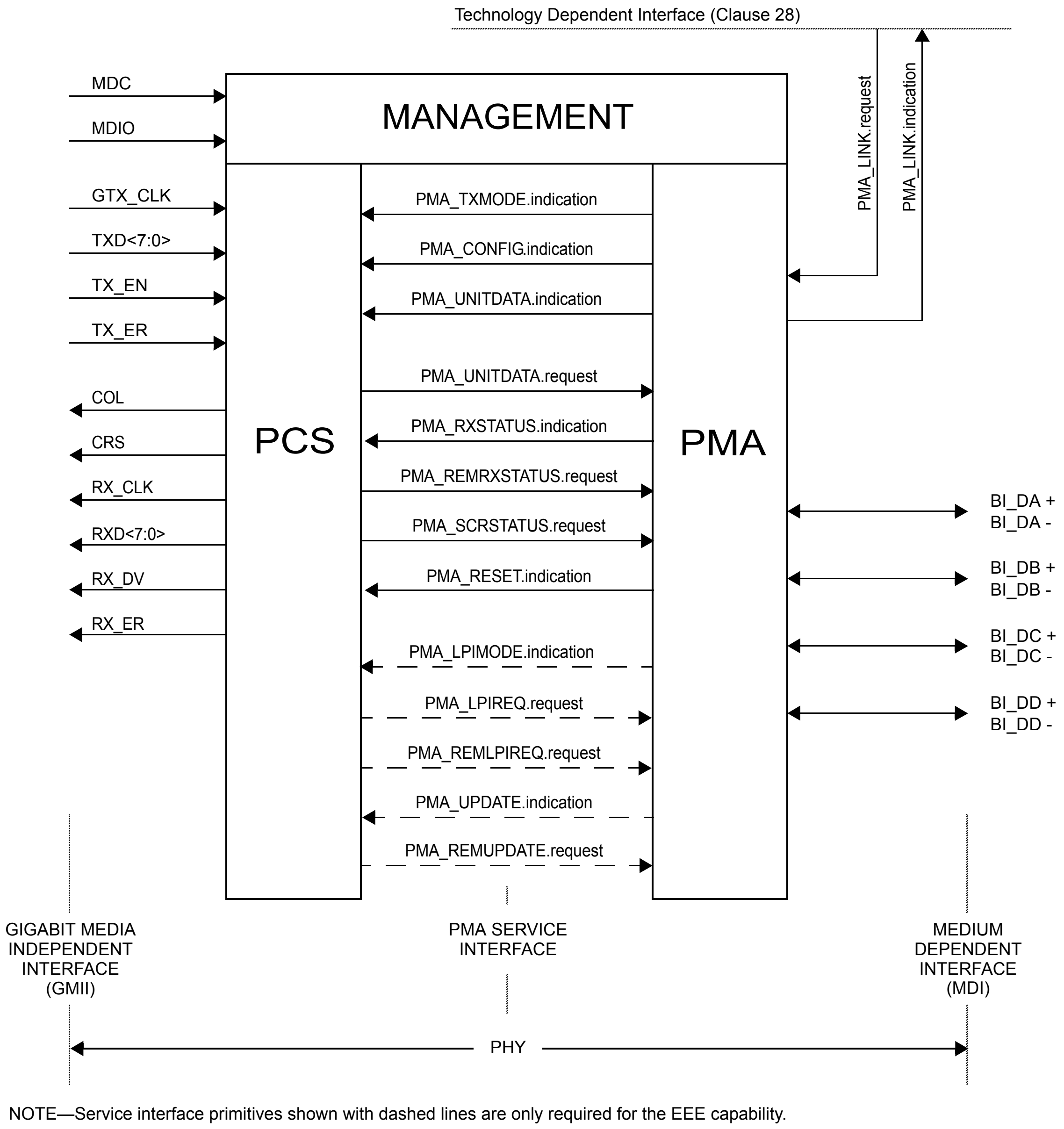


Figure 40–4—1000BASE-T service interfaces

40.2.3 PMA_TXMODE.indication

The transmitter in a 1000BASE-T link normally sends over the four pairs, code-groups that can represent a GMII data stream, control information, or idles.

40.2.3.1 Semantics of the primitive

PMA_TXMODE.indication (tx_mode)

PMA_TXMODE.indication specifies to PCS Transmit via the parameter tx_mode what sequence of code-groups the PCS should be transmitting. The parameter tx_mode can take on one of the following three values of the form:

SEND_N This value is continuously asserted when transmission of sequences of code-groups representing a GMII data stream (data mode), control mode or idle mode is to take place.

SEND_I This value is continuously asserted in case transmission of sequences of code-groups representing the idle mode is to take place.

SEND_Z This value is continuously asserted in case transmission of zeros is required.

40.2.3.2 When generated

The PMA PHY Control function generates PMA_TXMODE.indication messages continuously.

40.2.3.3 Effect of receipt

Upon receipt of this primitive, the PCS performs its Transmit function as described in 40.3.1.3.

40.2.4 PMA_CONFIG.indication

Each PHY in a 1000BASE-T link is capable of operating as a MASTER PHY and as a SLAVE PHY. MASTER-SLAVE configuration is determined during Auto-Negotiation (40.5). The result of this negotiation is provided to the PMA.

40.2.4.1 Semantics of the primitive

PMA_CONFIG.indication (config)

PMA_CONFIG.indication specifies to PCS and PMA Transmit via the parameter config whether the PHY must operate as a MASTER PHY or as a SLAVE PHY. The parameter config can take on one of the following two values of the form:

MASTER This value is continuously asserted when the PHY must operate as a MASTER PHY.

SLAVE This value is continuously asserted when the PHY must operate as a SLAVE PHY.

40.2.4.2 When generated

PMA generates PMA_CONFIG.indication messages continuously.

40.2.4.3 Effect of receipt

PCS and PMA Clock Recovery perform their functions in MASTER or SLAVE configuration according to the value assumed by the parameter config.

40.2.5 PMA_UNITDATA.request

This primitive defines the transfer of code-groups in the form of the tx_symb_vector parameter from the PCS to the PMA. The code-groups are obtained in the PCS Transmit function using the encoding rules defined in 40.3.1.3 to represent GMII data streams, an idle mode, or other sequences.

40.2.5.1 Semantics of the primitive

PMA_UNITDATA.request (tx_symb_vector)

During transmission, the PMA_UNITDATA.request simultaneously conveys to the PMA via the parameter tx_symb_vector the value of the symbols to be sent over each of the four transmit pairs BI_DA, BI_DB, BI_DC, and BI_DD. The tx_symb_vector parameter takes on the form:

SYMB_4D A vector of four quinary symbols, one for each of the four transmit pairs BI_DA, BI_DB, BI_DC, and BI_DD. Each quinary symbol may take on one of the values –2, –1, 0, +1, or +2.

The quinary symbols that are elements of tx_symb_vector are called, according to the pair on which each will be transmitted, tx_symb_vector[BI_DA], tx_symb_vector[BI_DB], tx_symb_vector[BI_DC], and tx_symb_vector[BI_DD].

40.2.5.2 When generated

The PCS generates PMA_UNITDATA.request (SYMB_4D) synchronously with every transmit clock cycle.

40.2.5.3 Effect of receipt

Upon receipt of this primitive the PMA transmits on the MDI the signals corresponding to the indicated quinary symbols. The parameter tx_symb_vector is also used by the PMA Receive function to process the signals received on pairs BI_DA, BI_DB, BI_DC, and BI_DD.

40.2.6 PMA_UNITDATA.indication

This primitive defines the transfer of code-groups in the form of the rx_symb_vector parameter from the PMA to the PCS.

40.2.6.1 Semantics of the primitive

PMA_UNITDATA.indication (rx_symb_vector)

During reception the PMA_UNITDATA.indication simultaneously conveys to the PCS via the parameter rx_symb_vector the values of the symbols detected on each of the four receive pairs BI_DA, BI_DB, BI_DC, and BI_DD. The rx_symbol_vector parameter takes on the form:

SYMB_4DA vector of four quinary symbols, one for each of the four receive pairs BI_DA, BI_DB, BI_DC, and BI_DD. Each quinary symbol may take on one of the values –2, –1, 0, +1, or +2.

The quinary symbols that are elements of rx_symb_vector are called, according to the pair upon which each symbol was received, rx_symbol_vector[BI_DA], rx_symbol_vector[BI_DB], rx_symbol_vector[BI_DC], and rx_symb_vector[BI_DD].

40.2.6.2 When generated

The PMA generates PMA_UNITDATA.indication (SYMB_4D) messages synchronously with signals received at the MDI. The nominal rate of the PMA_UNITDATA.indication primitive is 125 MHz, as governed by the recovered clock.

40.2.6.3 Effect of receipt

The effect of receipt of this primitive is unspecified.

40.2.7 PMA_SCRSTATUS.request

This primitive is generated by PCS Receive to communicate the status of the descrambler for the local PHY. The parameter scr_status conveys to the PMA Receive function the information that the descrambler has achieved synchronization.

40.2.7.1 Semantics of the primitive

PMA_SCRSTATUS.request (scr_status)

The scr_status parameter can take on one of two values of the form:

OK	The descrambler has achieved synchronization.
NOT_OK	The descrambler is not synchronized.

40.2.7.2 When generated

PCS Receive generates PMA_SCRSTATUS.request messages continuously.

40.2.7.3 Effect of receipt

The effect of receipt of this primitive is specified in 40.4.2.3, 40.4.2.4, and 40.4.6.1.

40.2.8 PMA_RXSTATUS.indication

This primitive is generated by PMA Receive to indicate the status of the receive link at the local PHY. The parameter loc_rcvr_status conveys to the PCS Transmit, PCS Receive, PMA PHY Control function, and Link Monitor the information on whether the status of the overall receive link is satisfactory or not. Note that loc_rcvr_status is used by the PCS Receive decoding functions. The criterion for setting the parameter loc_rcvr_status is left to the implementer. It can be based, for example, on observing the mean-square error at the decision point of the receiver and detecting errors during reception of symbol streams that represent the idle mode.

40.2.8.1 Semantics of the primitive

PMA_RXSTATUS.indication (loc_rcvr_status)

The loc_rcvr_status parameter can take on one of two values of the form:

OK	This value is asserted and remains true during reliable operation of the receive link for the local PHY.
NOT_OK	This value is asserted whenever operation of the link for the local PHY is unreliable.

40.2.8.2 When generated

PMA Receive generates PMA_RXSTATUS.indication messages continuously on the basis of signals received at the MDI.

40.2.8.3 Effect of receipt

The effect of receipt of this primitive is specified in Figure 40–16a and in subclauses 40.2 and 40.4.6.2.

40.2.9 PMA_REMRXSTATUS.request

This primitive is generated by PCS Receive to indicate the status of the receive link at the remote PHY as communicated by the remote PHY via its encoding of its loc_rcvr_status parameter. The parameter rem_rcvr_status conveys to the PMA PHY Control function the information on whether reliable operation of the remote PHY is detected or not. The criterion for setting the parameter rem_rcvr_status is left to the implementer. It can be based, for example, on asserting rem_rcvr_status is NOT_OK until loc_rcvr_status is OK and then asserting the detected value of rem_rcvr_status after proper PCS receive decoding is achieved.

40.2.9.1 Semantics of the primitive

PMA_REMRXSTATUS.request (rem_rcvr_status)

The rem_rcvr_status parameter can take on one of two values of the form:

OK The receive link for the remote PHY is operating reliably.
NOT_OKReliable operation of the receive link for the remote PHY is not detected.

40.2.9.2 When generated

The PCS generates PMA_REMRXSTATUS.request messages continuously on the basis on signals received at the MDI.

40.2.9.3 Effect of receipt

The effect of receipt of this primitive is specified in Figure 40–16a.

40.2.10 PMA_RESET.indication

This primitive is used to pass the PMA Reset function to the PCS (pcs_reset=ON) when reset is enabled.

The PMA_RESET.indication primitive can take on one of two values:

TRUE Reset is enabled.
FALSE Reset is not enabled.

40.2.10.1 When generated

The PMA Reset function is executed as described in 40.4.2.1.

40.2.10.2 Effect of receipt

The effect of receipt of this primitive is specified in 40.4.2.1.

40.2.11 PMA_LPIMODE.indication

This primitive is generated by the PMA to indicate that the PHY has entered the LPI mode of operation.

40.2.11.1 Semantics of the primitive

PMA_LPIMODE.indication(lpi_mode)

PMA_LPIMODE.indication specifies to the PCS Receive function, via the parameter lpi_mode, whether or not the PHY has entered LPI mode. The parameter lpi_mode can take on one of the following values of the form:

ON	This value is asserted with the PHY is operating in LPI mode.
OFF	This value is asserted during normal operation.

40.2.11.2 When generated

The PMA PHY Control function generates PMA_LPIMODE.indication messages continuously.

40.2.11.3 Effect of receipt

Upon receipt of this primitive, the PCS performs its Receive function as described in 40.3.1.4.

40.2.12 PMA_LPIREQ.request

This primitive is generated by the PCS to indicate a request to enter the LPI mode.

40.2.12.1 Semantics of the primitive

PMA_LPIREQ.request (loc_lpi_req)

PMA_LPIREQ.request specifies to the PMA PHY Control, via the parameter loc_lpi_req, whether or not the PHY is requested to enter the LPI mode. The parameter loc_lpi_req can take on one of the following values of the form:

TRUE	This value is continuously asserted when “Assert LPI” is present on the GMII. Note that “Assert LPI” at the GMII implies that no frame transmission is in progress hence 1000BTtransmit (see 40.3.3.1) will be set to FALSE by the PCS Transmit state diagram.
FALSE	This value is continuously asserted when “Assert LPI” is not present at the GMII.

40.2.12.2 When generated

The PCS Local LPI Request function generates PMA_LPIREQ.request messages continuously.

40.2.12.3 Effect of receipt

Upon receipt of this primitive, the PMA performs its PHY Control function as described in 40.4.2.4.

40.2.13 PMA_REMLPIREQ.request

This primitive is generated by the PCS to indicate a request to enter LPI mode as communicated by the remote PHY via its encoding of its loc_lpi_req parameter.

40.2.13.1 Semantics of the primitive

PMA_REMLPIREQ.request (rem_lpi_req)

PMA_REMLPIREQ.request specifies to the PMA PHY Control, via the parameter rem_lpi_req, whether or not the remote PHY is requesting entry into the LPI mode. The parameter rem_lpi_req can take on one of the following values of the form:

TRUE	This value is continuously asserted when LPI is encoded in the received symbols.
FALSE	This value is continuously asserted when LPI is not encoded in the received symbols.

40.2.13.2 When generated

The PCS Receive function generates PMA_REMLPIREQ.request messages continuously on the basis of the signals received at the MDI.

40.2.13.3 Effect of receipt

Upon receipt of this primitive, the PMA performs its PHY Control function as described in 40.4.2.4.

40.2.14 PMA_UPDATE.indication

This primitive is generated by the PMA to indicate that the PHY has completed the update of its receiver state (e.g., timing recovery, adaptive filter coefficients).

40.2.14.1 Semantics of the primitive

PMA_UPDATE.indication(loc_update_done)

PMA_UPDATE.indication specifies to the PCS Transmit functions, via the parameter loc_update_done, whether or not the PHY has completed the update of its receiver state. The parameter loc_update_done can take on one of the following values of the form:

TRUE	This value is asserted when the PHY has completed the current update.
FALSE	This value is asserted when the PHY is ready for the next update or when the current update is still in progress.

40.2.14.2 When generated

The PMA PHY Control function generates PMA_UPDATE.indication messages continuously.

40.2.14.3 Effect of receipt

Upon receipt of this primitive, the PCS performs its Transmit function as described in 40.3.1.3 and 40.3.1.4.

40.2.15 PMA_REMUPDATE.request

This primitive is generated by the PCS to indicate that the remote PHY has completed the update of its receiver state (e.g., timing recovery, adaptive filter coefficients).

40.2.15.1 Semantics of the primitive

PMA_REMUPDATE.request(rem_update_done)

PMA_REMUPDATE.indication specifies to the PMA PHY Control function, via the parameter rem_update_done, whether or not the remote PHY has completed the update of its receiver state. The parameter rem_update_done can take on one of the following values of the form:

TRUE	This value is asserted when the remote PHY has completed the current update.
FALSE	This value is asserted to when the remote PHY is ready for the next update or when the current update is still in progress.

40.2.15.2 When generated

The PCS Receive function generates PMA_REMUDPATE.request messages continuously.

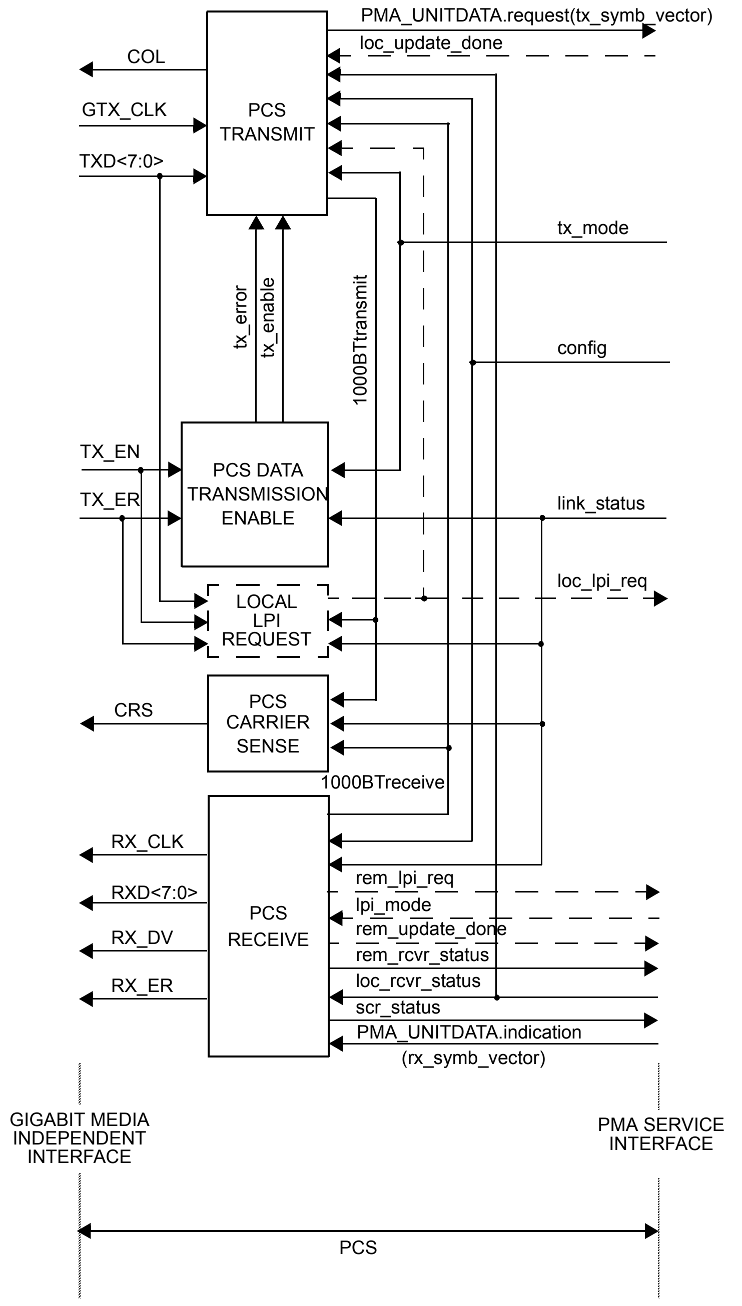
40.2.15.3 Effect of receipt

Upon receipt of this primitive, the PMA performs its PHY Control function as described in 40.4.2.4.

40.3 Physical Coding Sublayer (PCS)

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

The PCS reference diagram, Figure 40–5, shows how the four 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 40–5. Management is specified in Clause 30. See also Figure 40–7, which defines the structure of frames passed from PCS to PMA.



NOTE—Signals and functions shown with dashed lines are only required for the EEE capability.

Figure 40–5—PCS reference diagram

40.3.1 PCS functions

40.3.1.1 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 36.2.5.1.3).
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.

40.3.1.2 PCS Data Transmission Enable

The PCS Data Transmission Enabling process generates the signals tx_enable and tx_error, which PCS Transmit uses for data and carrier extension encoding. The process uses logical operations on tx_mode, TX_ER, TX_EN, and TXD<7:0>. The PCS shall implement the Data Transmission Enabling process as depicted in Figure 40–8 including compliance with the associated state variables as specified in 40.3.3.

40.3.1.3 PCS Transmit function

The PCS Transmit function shall conform to the PCS Transmit state diagram in Figure 40–10.

The PCS Transmit function generates the GMII signal COL based on whether a reception is occurring simultaneously with transmission. The PCS Transmit function is not required to generate the GMII signal COL in a 1000BASE-T PHY that does not support half duplex operation.

In each symbol period, PCS Transmit generates a code-group (A_n, B_n, C_n, D_n) that is transferred to the PMA via the PMA_UNITDATA.request primitive. The PMA transmits symbols A_n, B_n, C_n, D_n over wire-pairs BI_DA, BI_DB, BI_DC, and BI_DD respectively. The integer, n, is a time index that is introduced to establish a temporal relationship between different symbol periods. A symbol period, T, is nominally equal to 8 ns. In normal mode of operation, between streams of data indicated by the parameter tx_enable, PCS Transmit generates sequences of vectors using the encoding rules defined for the idle mode. Upon assertion of tx_enable, PCS Transmit passes a SSD of two consecutive vectors of four quinary symbols to the PMA, replacing the first two preamble octets. Following the SSD, each TXD<7:0> octet is encoded using an 4D-PAM5 technique into a vector of four quinary symbols until tx_enable is de-asserted. If TX_ER is asserted while tx_enable is also asserted, then PCS Transmit passes to the PMA vectors indicating a transmit error. Note that if the signal TX_ER is asserted while SSD is being sent, the transmission of the error condition is delayed until transmission of SSD has been completed. Following the de-assertion of tx_enable, a Convolutional State Reset (CSReset) of two consecutive code-groups, followed by an ESD of two consecutive code-groups, is generated, after which the transmission of idle or control mode is resumed.

If a PMA_TXMODE.indication message has the value SEND_Z, PCS Transmit passes a vector of zeros at each symbol period to the PMA via the PMA_UNITDATA.request primitive.

If a PMA_TXMODE.indication message has the value SEND_I, PCS Transmit generates sequences of code-groups according to the encoding rule in training mode. Special code-groups that use only the values {+2, 0, –2} are transmitted in this case. Training mode encoding also takes into account the value of the parameter loc_rcvr_status. By this mechanism, a PHY indicates the status of its own receiver to the link partner during idle transmission.

When the PHY supports the optional EEE capability, the LPI mode encoding also takes into account the value of the parameter loc_lpi_req. By this mechanism, the PHY indicates whether it requests to operate in LPI mode or return to the normal mode of operation. In addition, LPI mode encoding takes into account the value of loc_update_done. By this mechanism, the PHY indicates whether it has completed the update of its receiver state (e.g., timing recovery, adaptive filter coefficients) or not, as indicated by the PMA PHY Control function.

In the normal mode of operation, the PMA_TXMODE.indication message has the value SEND_N, and the PCS Transmit function uses an 8B1Q4 coding technique to generate at each symbol period code-groups that represent data, control or idle based on the code-groups defined in Table 40–1 and Table 40–2. During transmission of data, the TXD<7:0> bits are scrambled by the PCS using a side-stream scrambler, then encoded into a code-group of quinary symbols and transferred to the PMA. During data encoding, PCS Transmit utilizes a three-state convolutional encoder.

The transition from idle or carrier extension to data is signalled by inserting a SSD, and the end of transmission of data is signalled by an ESD. Further code-groups are reserved for signaling the assertion of TX_ER within a stream of data, carrier extension, CSReset, and other control functions. During idle and carrier extension encoding, special code-groups with symbol values restricted to the set {2, 0, –2} are used. These code-groups are also generated using the transmit side-stream scrambler. However, the encoding rules for the idle, SSD, and carrier extend code-groups are different from the encoding rules for data, CSReset, CSExtend, and ESD code-groups. During idle, SSD, and carrier extension, the PCS Transmit function reverses the sign of the transmitted symbols. This allows, at the receiver, sequences of code-groups that represent data, CSReset, CSExtend, and ESD to be easily distinguished from sequences of code-groups that represent SSD, carrier extension, and idle.

PCS encoding involves the generation of the four-bit words $Sx_n[3:0]$, $Sy_n[3:0]$, and $Sg_n[3:0]$ from which the quinary symbols (A_n, B_n, C_n, D_n) are obtained. The four-bit words $Sx_n[3:0]$, $Sy_n[3:0]$, and $Sg_n[3:0]$ are determined (as explained in 40.3.1.3.2) from sequences of pseudorandom binary symbols derived from the transmit side-stream scrambler.

40.3.1.3.1 Side-stream scrambler polynomials

The PCS Transmit function employs side-stream scrambling. If the parameter config provided to the PCS by the PMA PHY Control function via the PMA_CONFIG.indication message assumes the value MASTER, PCS Transmit shall employ

$$g_{\mathrm{M}}(x) = 1 + x^{13} + x^{33}$$

as transmitter side-stream scrambler generator polynomial. If the PMA_CONFIG.indication message assumes the value of SLAVE, PCS Transmit shall employ

$$g_{\mathrm{S}}(x) = 1 + x^{20} + x^{33}$$

as transmitter side-stream scrambler generator polynomial. An implementation of master and slave PHY side-stream scramblers by linear-feedback shift registers is shown in Figure 40–6. 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 implementer. In no case shall the scrambler state be initialized to all zeros.

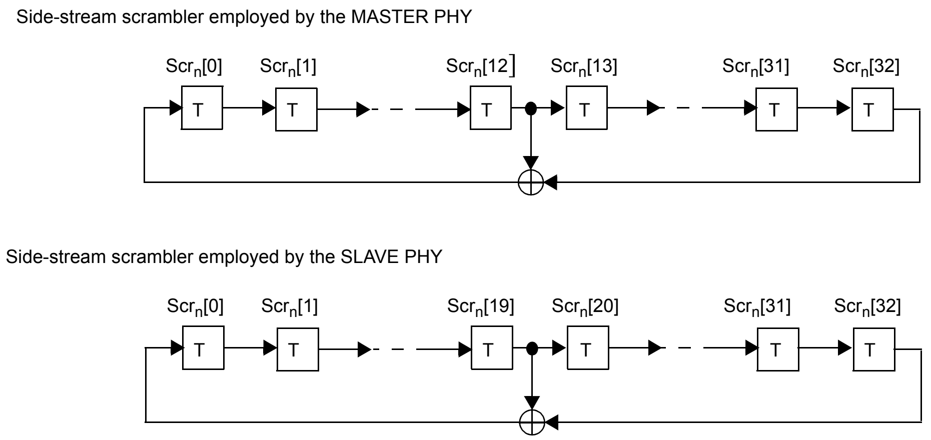

Side-stream scrambler employed by the MASTER PHY
Scrn[0]
Scrn[1]
Scrn[12]
Scrn[13]
Scrn[31]
Scrn[32]
T
Side-stream scrambler employed by the SLAVE PHY
Scrn[0]
Scrn[1]
Scrn[19]
Scrn[20]
Scrn[31]
Scrn[32]
T

Figure 40–6—A realization of side-stream scramblers by linear feedback shift registers

40.3.1.3.2 Generation of bits $Sx_n[3:0]$, $Sy_n[3:0]$, and $Sg_n[3:0]$

PCS Transmit encoding rules are based on the generation, at time n, of the twelve bits $Sx_n[3:0]$, $Sy_n[3:0]$, and $Sg_n[3:0]$. The eight bits, $Sx_n[3:0]$ and $Sy_n[3:0]$, are used to generate the scrambler octet $Sc_n[7:0]$ for decorrelating the GMII data word TXD<7:0> during data transmission and for generating the idle and training symbols. The four bits, $Sg_n[3:0]$, are used to randomize the signs of the quinary symbols (A_n, B_n, C_n, D_n) so that each symbol stream has no dc bias. These twelve bits are generated in a systematic fashion using three bits, X_n, Y_n, and $Scr_n[0]$, and an auxiliary generating polynomial, g(x). The two bits, X_n and Y_n, are mutually uncorrelated and also uncorrelated with the bit $Scr_n[0]$. 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. These two bits are derived from elements of the same maximum-length shift register sequence of length $2^{33} – 1$ as $Scr_n[0]$, but shifted in time. The associated delays are all large and different so that there is no short-term correlation among the bits $Scr_n[0]$, X_n, and Y_n. The bits X_n and Y_n are generated as follows:

$X_n = Scr_n[4] \wedge Scr_n[6]$

$Y_n = Scr_n[1] \wedge Scr_n[5]$

where ^ denotes the XOR logic operator. From the three bits X_n, Y_n, and $Scr_n[0]$, further mutually uncorrelated bit streams are obtained systematically using the generating polynomial

$g(x) = x^3 \wedge x^8$

The four bits $Sy_n[3:0]$ are generated using the bit $Scr_n[0]$ and g(x) as in the following equations:

$Sy_n[0] = Scr_n[0]$

$Sy_n[1] = g(Scr_n[0]) = Scr_n[3] \wedge Scr_n[8]$

$Sy_n[2] = g^2(Scr_n[0]) = Scr_n[6] \wedge Scr_n[16]$

$Sy_n[3] = g^3(Scr_n[0]) = Scr_n[9] \wedge Scr_n[14] \wedge Scr_n[19] \wedge Scr_n[24]$

The four bits $Sx_n[3:0]$ are generated using the bit X_n and g(x) as in the following equations:

$Sx_n[0] = X_n = Scr_n[4] \wedge Scr_n[6]$

$Sx_n[1] = g(X_n) = Scr_n[7] \wedge Scr_n[9] \wedge Scr_n[12] \wedge Scr_n[14]$

$Sx_n[2] = g^2(X_n) = Scr_n[10] \wedge Scr_n[12] \wedge Scr_n[20] \wedge Scr_n[22]$

$$\begin{aligned} Sx_n[3] = g^3(X_n) = {} & Scr_n[13] \wedge Scr_n[15] \wedge Scr_n[18] \wedge Scr_n[20] \wedge \\ & Scr_n[23] \wedge Scr_n[25] \wedge Scr_n[28] \wedge Scr_n[30] \end{aligned}$$

The four bits $Sg_n[3:0]$ are generated using the bit Y_n and g(x) as in the following equations:

$Sg_n[0] = Y_n = Scr_n[1] \wedge Scr_n[5]$

$Sg_n[1] = g(Y_n) = Scr_n[4] \wedge Scr_n[8] \wedge Scr_n[9] \wedge Scr_n[13]$

$Sg_n[2] = g^2(Y_n) = Scr_n[7] \wedge Scr_n[11] \wedge Scr_n[17] \wedge Scr_n[21]$

$$\begin{aligned} Sg_n[3] = g^3(Y_n) = {} & Scr_n[10] \wedge Scr_n[14] \wedge Scr_n[15] \wedge Scr_n[19] \wedge \\ & Scr_n[20] \wedge Scr_n[24] \wedge Scr_n[25] \wedge Scr_n[29] \end{aligned}$$

By construction, the twelve bits $Sx_n[3:0]$, $Sy_n[3:0]$, and $Sg_n[3:0]$ are derived from elements of the same maximum-length shift register sequence of length $2^{33}–1$ as $Scr_n[0]$, but shifted in time by varying delays. The associated delays are all large and different so that there is no apparent correlation among the bits.

40.3.1.3.3 Generation of bits $Sc_n[7:0]$

The bits $Sc_n[7:0]$ are used to scramble the GMII data octet TXD[7:0] and for control, idle, and training mode quartet generation. The definition of these bits is dependent upon the bits $Sx_n[3:0]$ and $Sy_n[3:0]$ that are specified in 40.3.1.3.2, the variable tx_mode that is obtained through the PMA Service Interface, the variable tx_enable_n that is defined in Figure 40–8, and the time index n.

The four bits $Sc_n[7:4]$ are defined as

$$Sc_n[7:4] = \begin{cases} Sx_n[3:0] & \text{if } (\text{tx\_enable}_{n-2} = 1) \\ [0\ 0\ 0\ 0] & \text{else} \end{cases}$$

The bits $Sc_n[3:1]$ are defined as

$$Sc_n[3:1] = \begin{cases} [0\ 0\ 0] & \text{if } (\text{tx\_mode} = \text{SEND\_Z}) \\ Sy_n[3:1] & \text{else if } (n\text{-}n_0) = 0 \pmod 2 \\ (Sy_{n-1}[3:1] \wedge [1\ 1\ 1]) & \text{else} \end{cases}$$

where n_0 denotes the time index of the last transmitter side-stream scrambler reset.

The bit $Sc_n[0]$ is defined as

$$Sc_n[0] = \begin{cases} 0 & \text{if } (\textit{tx\_mode} = \text{SEND\_Z}) \\ Sy_n[0] & \text{else} \end{cases}$$

40.3.1.3.4 Generation of bits $Sd_n[8:0]$

The PCS Transmit function generates a nine-bit word $Sd_n[8:0]$ from Sc_n that represents either a convolutionally encoded stream of data, control, or idle mode code-groups. The convolutional encoder uses a three-bit word $cs_n[2:0]$, which is defined as

$$cs_n[1] = \begin{cases} Sd_n[6] \wedge cs_{n-1}[0] & \text{if } (tx\_enable_{n-2} = 1) \\ 0 & \text{else} \end{cases}$$

$$cs_n[2] = \begin{cases} Sd_n[7] \wedge cs_{n-1}[1] & \text{if } (tx\_enable_{n-2} = 1) \\ 0 & \text{else} \end{cases}$$

$$cs_n[0] = cs_{n-1}[2]$$

from which $Sd_n[8]$ is obtained as

$$Sd_n[8] = cs_n[0]$$

The convolutional encoder bits are non-zero only during the transmission of data. Upon the completion of a frame, the convolutional encoder bits are reset using the bit $csreset_n$. The bit $csreset_n$ is defined as

$$csreset_n = (tx\_enable_{n-2}) \text{ and } (\text{not } tx\_enable_n)$$

The bits $Sd_n[7:6]$ are derived from the bits $Sc_n[7:6]$, the GMII data bits $TXD_n[7:6]$, and from the convolutional encoder bits as

$$Sd_n[7] = \begin{cases} Sc_n[7] \wedge TXD_n[7] & \text{if } (csreset_n = 0 \text{ and } tx\_enable_{n-2} = 1) \\ cs_{n-1}[1] & \text{else if } (csreset_n=1) \\ Sc_n[7] & \text{else} \end{cases}$$

$$Sd_n[6] = \begin{cases} Sc_n[6] \wedge TXD_n[6] & \text{if } (csreset_n = 0 \text{ and } tx\_enable_{n-2} = 1) \\ cs_{n-1}[0] & \text{else if } (csreset_n=1) \\ Sc_n[6] & \text{else} \end{cases}$$

The bits $Sd_n[5:4]$ are derived from the bits $Sc_n[5:4]$ and the GMII data bits $TXD_n[5:4]$ as

$$Sd_n[5:4] = \begin{cases} Sc_n[5:4] \wedge TXD_n[5:4] & \text{if } (tx\_enable_{n-2} = 1) \\ Sc_n[5:4] & \text{else} \end{cases}$$

The bit $Sd_n[3]$ is used to scramble the GMII data bit $TXD_n[3]$ during data mode and to encode loc_lpi_req otherwise. It is defined as

$$Sd_n[3] = \begin{cases} Sc_n[3] \wedge TXD_n[3] \text{ if } (tx\_enable_{n-2} = 1) \\ Sc_n[3] \wedge 1 \text{ else if } ((loc\_lpi\_req = \text{TRUE}) \text{ and } (tx\_mode \neq \text{SEND\_Z})) \\ Sc_n[3] \text{ else} \end{cases}$$

The bit $Sd_n[2]$ is used to scramble the GMII data bit $TXD_n[2]$ during data mode and to encode loc_rcvr_status otherwise. It is defined as

$$Sd_n[2] = \begin{cases} Sc_n[2] \wedge TXD_n[2] \text{ if } (tx\_enable_{n-2} = 1) \\ Sc_n[2] \wedge 1 \text{ else if } ((loc\_rcvr\_status = \text{OK}) \text{ and } (tx\_mode \neq \text{SEND\_Z})) \\ Sc_n[2] \text{ else} \end{cases}$$

The bits $Sd_n[1:0]$ are used to transmit carrier extension information during tx_mode=SEND_N and are thus dependent upon the bits $cext_n$ and $cext\_err_n$. In addition, bit $Sd_n[1]$ is used to encode loc_update_done. These bits are dependent on the variable tx_error_n, which is defined in Figure 40–8. These bits are defined as

$$cext_n = \begin{cases} tx\_error_n \text{ if } ((tx\_enable_n = 0) \text{ and } (TXD_n[7:0] = \text{0x0F})) \\ 0 \text{ else} \end{cases}$$

$$cext\_err_n = \begin{cases} tx\_error_n \text{ if } ((tx\_enable_n = 0) \text{ and } (TXD_n[7:0] \neq \text{0x0F}) \text{ and } (loc\_lpi\_req = FALSE)) \\ 0 \text{ else} \end{cases}$$

$$Sd_n[1] = \begin{cases} Sc_n[1] \wedge TXD_n[1] \text{ if } (tx\_enable_{n-2} = 1) \\ Sc_n[1] \wedge 1 \text{ else if } ((loc\_update\_done = \text{TRUE}) \text{ and } (tx\_mode \neq \text{SEND\_Z})) \\ Sc_n[1] \wedge cext\_err_n \text{ else} \end{cases}$$

$$Sd_n[0] = \begin{cases} Sc_n[0] \wedge TXD_n[0] \text{ if } (tx\_enable_{n-2} = 1) \\ Sc_n[0] \wedge cext_n \text{ else} \end{cases}$$

40.3.1.3.5 Generation of quinary symbols TA_n, TB_n, TC_n, TD_n

The nine-bit word $Sd_n[8:0]$ is mapped to a quartet of quinary symbols (TA_n, TB_n, TC_n, TD_n) according to Table 40–1 and Table 40–2 shown as $Sd_n[6:8] + Sd_n[5:0]$.

Encoding of error indication:

If tx_error_n=1 when the condition ($tx\_enable_n * tx\_enable_{n-2}$) = 1, error indication is signaled by means of symbol substitution. In this condition, the values of $Sd_n[5:0]$ are ignored during mapping and the symbols corresponding to the row denoted as “xmt_err” in Table 40–1 and Table 40–2 shall be used.

Encoding of Convolutional Encoder Reset:

If tx_error_n=0 when the variable $csreset_n$ = 1, the convolutional encoder reset condition is normal. This condition is indicated by means of symbol substitution, where the values of $Sd_n[5:0]$ are ignored during mapping and the symbols corresponding to the row denoted as “CSReset” in Table 40–1 and Table 40–2 shall be used.

Encoding of Carrier Extension during Convolutional Encoder Reset:

If tx_error_n=1 when the variable $csreset_n$ = 1, the convolutional encoder reset condition indicates carrier extension. In this condition, the values of $Sd_n[5:0]$ are ignored during mapping and the symbols corresponding to the row denoted as “CSExtend” in Table 40–1 and Table 40–2 shall be used when TXD_n = 0x’0F, and the row denoted as “CSExtend_Err” in Table 40–1 and Table 40–2 shall be used when TXD_n ¦ 0x’0F. The latter condition denotes carrier extension with error. In case carrier extension with error is indicated during the first octet of CSReset, the error condition shall be encoded during the second octet of CSReset, and during the subsequent two octets of the End-of-Stream delimiter as well. Thus, the error condition is assumed to persist during the symbol substitutions at the End-of-Stream.

Encoding of Start-of-Stream delimiter:

The Start-of-Stream delimiter (SSD) is related to the condition $SSD_{n,}$ which is defined as (tx_enable_n) * (!tx_enable_{n-2}) = 1, where “*” and “!” denote the logic AND and NOT operators, respectively. For the generation of SSD, the first two octets of the preamble in a data stream are mapped to the symbols corresponding to the rows denoted as SSD1 and SSD2 respectively in Table 40–1. The symbols corresponding to the SSD1 row shall be used when the condition (tx_enable_n) * (!tx_enable_{n-1}) = 1. The symbols corresponding to the SSD2 row shall be used when the condition (tx_enable_{n-1}) * (!tx_enable_{n-2}) = 1.

Encoding of End-of-Stream delimiter:

The definition of an End-of-Stream delimiter (ESD) is related to the condition ESD_n, which is defined as (!tx_enable_{n-2}) * (tn_enable_{n-4}) = 1. This occurs during the third and fourth symbol periods after transmission of the last octet of a data stream.

If carrier extend error is indicated during ESD, the symbols corresponding to the ESD_Ext_Err row shall be used. The two conditions upon which this may occur are

(tx_error_n) * (tx_error_{n-1}) * (tx_error_{n-2}) * ($TXD_n \neq$ 0x0F) = 1, and
(tx_error_n) * (tx_error_{n-1}) * (tx_error_{n-2}) * (tx_error_{n-3}) * ($TXD_n \neq$ 0x0F) = 1.

The symbols corresponding to the ESD1 row in Table 40–1 shall be used when the condition (!tx_enable_{n-2}) * (tx_enable_{n-3}) = 1, in the absence of carrier extend error indication at time n.

The symbols corresponding to the ESD2_Ext_0 row in Table 40–1 shall be used when the condition (!tx_enable_{n-3}) * (tx_enable_{n-4}) * (!tx_error_n) * (!tx_error_{n-1}) = 1.

The symbols corresponding to the ESD2_Ext_1 row in Table 40–1 shall be used when the condition (!tx_enable_{n-3}) * (tx_enable_{n-4}) * (!tx_error_n) * (tx_error_{n-1}) * (tx_error_{n-2}) * (tx_error_{n-3}) = 1.

The symbols corresponding to the ESD2_Ext_2 row in Table 40–1 shall be used when the condition (!tx_enable_{n-3}) * (tx_enable_{n-4}) * (tx_error_n) * (tx_error_{n-1}) * (tx_error_{n-2}) * (tx_error_{n-3}) * (TXD_n= 0x0F) = 1, in the absence of carrier extend error indication.

NOTE—The ASCII for Table 40–1 and Table 40–2 is available at http://standards.ieee.org/downloads/802.3/.[8]

[8]*Copyright release for symbol codes:* Users of this standard may freely reproduce the symbol codes in this subclause so it can be used for its intended purpose.

Table 40–1—Bit-to-symbol mapping (even subsets)

		$Sd_n[6:8]$ = [000]	$Sd_n[6:8]$ = [010]	$Sd_n[6:8]$ = [100]	$Sd_n[6:8]$ = [110]
Condition	**$Sd_n[5:0]$**	**TA_n,TB_n,TC_n,TD_n**	**TA_n,TB_n,TC_n,TD_n**	**TA_n,TB_n,TC_n,TD_n**	**TA_n,TB_n,TC_n,TD_n**
Normal	000000	0, 0, 0, 0	0, 0,+1,+1	0,+1,+1, 0	0,+1, 0,+1
Normal	000001	–2, 0, 0, 0	–2, 0,+1,+1	–2,+1,+1, 0	–2,+1, 0,+1
Normal	000010	0,–2, 0, 0	0,–2,+1,+1	0,–1,+1, 0	0,–1, 0,+1
Normal	000011	–2,–2, 0, 0	–2,–2,+1,+1	–2,–1,+1, 0	–2,–1, 0,+1
Normal	000100	0, 0,–2, 0	0, 0,–1,+1	0,+1,–1, 0	0,+1,–2,+1
Normal	000101	–2, 0,–2, 0	–2, 0,–1,+1	–2,+1,–1, 0	–2,+1,–2,+1
Normal	000110	0,–2,–2, 0	0,–2,–1,+1	0,–1,–1, 0	0,–1,–2,+1
Normal	000111	–2,–2,–2, 0	–2,–2,–1,+1	–2,–1,–1, 0	–2,–1,–2,+1
Normal	001000	0, 0, 0,–2	0, 0,+1,–1	0,+1,+1,–2	0,+1, 0,–1
Normal	001001	–2, 0, 0,–2	–2, 0,+1,–1	–2,+1,+1,–2	–2,+1, 0,–1
Normal	001010	0,–2, 0,–2	0,–2,+1,–1	0,–1,+1,–2	0,–1, 0,–1
Normal	001011	–2,–2, 0,–2	–2,–2,+1,–1	–2,–1,+1,–2	–2,–1, 0,–1
Normal	001100	0, 0,–2,–2	0, 0,–1,–1	0,+1,–1,–2	0,+1,–2,–1
Normal	001101	–2, 0,–2,–2	–2, 0,–1,–1	–2,+1,–1,–2	–2,+1,–2,–1
Normal	001110	0,–2,–2,–2	0,–2,–1,–1	0,–1,–1,–2	0,–1,–2,–1
Normal	001111	–2,–2,–2,–2	–2,–2,–1,–1	–2,–1,–1,–2	–2,–1,–2,–1
Normal	010000	+1,+1,+1,+1	+1,+1, 0, 0	+1, 0, 0,+1	+1, 0,+1, 0
Normal	010001	–1,+1,+1,+1	–1,+1, 0, 0	–1, 0, 0,+1	–1, 0,+1, 0
Normal	010010	+1,–1,+1,+1	+1,–1, 0, 0	+1,–2, 0,+1	+1,–2,+1, 0
Normal	010011	–1,–1,+1,+1	–1,–1, 0, 0	–1,–2, 0,+1	–1,–2,+1, 0
Normal	010100	+1,+1,–1,+1	+1,+1,–2, 0	+1, 0,–2,+1	+1, 0,–1, 0
Normal	010101	–1,+1,–1,+1	–1,+1,–2, 0	–1, 0,–2,+1	–1, 0,–1, 0
Normal	010110	+1,–1,–1,+1	+1,–1,–2, 0	+1,–2,–2,+1	+1,–2,–1, 0
Normal	010111	–1,–1,–1,+1	–1,–1,–2, 0	–1,–2,–2,+1	–1,–2,–1, 0
Normal	011000	+1,+1,+1,–1	+1,+1, 0,–2	+1, 0, 0,–1	+1, 0,+1,–2
Normal	011001	–1,+1,+1,–1	–1,+1, 0,–2	–1, 0, 0,–1	–1, 0,+1,–2
Normal	011010	+1,–1,+1,–1	+1,–1, 0,–2	+1,–2, 0,–1	+1,–2,+1,–2
Normal	011011	–1,–1,+1,–1	–1,–1, 0,–2	–1,–2, 0,–1	–1,–2,+1,–2
Normal	011100	+1,+1,–1,–1	+1,+1,–2,–2	+1, 0,–2,–1	+1, 0,–1,–2
Normal	011101	–1,+1,–1,–1	–1,+1,–2,–2	–1, 0,–2,–1	–1, 0,–1,–2
Normal	011110	+1,–1,–1,–1	+1,–1,–2,–2	+1,–2,–2,–1	+1,–2,–1,–2

Table 40–1—Bit-to-symbol mapping (even subsets) ***(continued)***

		Sd_n[6:8] = [000]	**Sd_n[6:8] = [010]**	**Sd_n[6:8] = [100]**	**Sd_n[6:8] = [110]**
Condition	**Sd_n[5:0]**	**TA_n,TB_n,TC_n,TD_n**	**TA_n,TB_n,TC_n,TD_n**	**TA_n,TB_n,TC_n,TD_n**	**TA_n,TB_n,TC_n,TD_n**
Normal	011111	–1,–1,–1,–1	–1,–1,–2,–2	–1,–2,–2,–1	–1,–2,–1,–2
Normal	100000	+2, 0, 0, 0	+2, 0,+1,+1	+2,+1,+1, 0	+2,+1, 0,+1
Normal	100001	+2,–2, 0, 0	+2,–2,+1,+1	+2,–1,+1, 0	+2,–1, 0,+1
Normal	100010	+2, 0,–2, 0	+2, 0,–1,+1	+2,+1,–1, 0	+2,+1,–2,+1
Normal	100011	+2,–2,–2, 0	+2,–2,–1,+1	+2,–1,–1, 0	+2,–1,–2,+1
Normal	100100	+2, 0, 0,–2	+2, 0,+1,–1	+2,+1,+1,–2	+2,+1, 0,–1
Normal	100101	+2,–2, 0,–2	+2,–2,+1,–1	+2,–1,+1,–2	+2,–1, 0,–1
Normal	100110	+2, 0,–2,–2	+2, 0,–1,–1	+2,+1,–1,–2	+2,+1,–2,–1
Normal	100111	+2,–2,–2,–2	+2,–2,–1,–1	+2,–1,–1,–2	+2,–1,–2,–1
Normal	101000	0, 0,+2, 0	+1,+1,+2, 0	+1, 0,+2,+1	0,+1,+2,+1
Normal	101001	–2, 0,+2, 0	–1,+1,+2, 0	–1, 0,+2,+1	–2,+1,+2,+1
Normal	101010	0,–2,+2, 0	+1,–1,+2, 0	+1,–2,+2,+1	0,–1,+2,+1
Normal	101011	–2,–2,+2, 0	–1,–1,+2, 0	–1,–2,+2,+1	–2,–1,+2,+1
Normal	101100	0, 0,+2,–2	+1,+1,+2,–2	+1, 0,+2,–1	0,+1,+2,–1
Normal	101101	–2, 0,+2,–2	–1,+1,+2,–2	–1, 0,+2,–1	–2,+1,+2,–1
Normal	101110	0,–2,+2,–2	+1,–1,+2,–2	+1,–2,+2,–1	0,–1,+2,–1
Normal	101111	–2,–2,+2,–2	–1,–1,+2,–2	–1,–2,+2,–1	–2,–1,+2,–1
Normal	110000	0,+2, 0, 0	0,+2,+1,+1	+1,+2, 0,+1	+1,+2,+1, 0
Normal	110001	–2,+2, 0, 0	–2,+2,+1,+1	–1,+2, 0,+1	–1,+2,+1, 0
Normal	110010	0,+2,–2, 0	0,+2,–1,+1	+1,+2,–2,+1	+1,+2,–1, 0
Normal	110011	–2,+2,–2, 0	–2,+2,–1,+1	–1,+2,–2,+1	–1,+2,–1, 0
Normal	110100	0,+2, 0,–2	0,+2,+1,–1	+1,+2, 0,–1	+1,+2,+1,–2
Normal	110101	–2,+2, 0,–2	–2,+2,+1,–1	–1,+2, 0,–1	–1,+2,+1,–2
Normal	110110	0,+2,–2,–2	0,+2,–1,–1	+1,+2,–2,–1	+1,+2,–1,–2
Normal	110111	–2,+2,–2,–2	–2,+2,–1,–1	–1,+2,–2,–1	–1,+2,–1,–2
Normal	111000	0, 0, 0,+2	+1,+1, 0,+2	0,+1,+1,+2	+1, 0,+1,+2
Normal	111001	–2, 0, 0,+2	–1,+1, 0,+2	–2,+1,+1,+2	–1, 0,+1,+2
Normal	111010	0,–2, 0,+2	+1,–1, 0,+2	0,–1,+1,+2	+1,–2,+1,+2
Normal	111011	–2,–2, 0,+2	–1,–1, 0,+2	–2,–1,+1,+2	–1,–2,+1,+2
Normal	111100	0, 0,–2,+2	+1,+1,–2,+2	0,+1,–1,+2	+1, 0,–1,+2
Normal	111101	–2, 0,–2,+2	–1,+1,–2,+2	–2,+1,–1,+2	–1, 0,–1,+2
Normal	111110	0,–2,–2,+2	+1,–1,–2,+2	0,–1,–1,+2	+1,–2,–1,+2

Table 40–1—Bit-to-symbol mapping (even subsets) *(continued)*

		$Sd_n[6:8]$ = [000]	$Sd_n[6:8]$ = [010]	$Sd_n[6:8]$ = [100]	$Sd_n[6:8]$ = [110]
Condition	**$Sd_n[5:0]$**	**TA_n,TB_n,TC_n,TD_n**	**TA_n,TB_n,TC_n,TD_n**	**TA_n,TB_n,TC_n,TD_n**	**TA_n,TB_n,TC_n,TD_n**
Normal	111111	–2,–2,–2,+2	–1,–1,–2,+2	–2,–1,–1,+2	–1,–2,–1,+2
xmt_err	XXXXXX	0,+2,+2,0	+1,+1,+2,+2	+2,+1,+1,+2	+2,+1,+2,+1
CSExtend_Err	XXXXXX	–2,+2,+2,–2	–1,–1,+2,+2	+2,–1,–1,+2	+2,–1,+2,–1
CSExtend	XXXXXX	+2, 0, 0,+2	+2,+2,+1,+1	+1,+2,+2,+1	+1,+2,+1,+2
CSReset	XXXXXX	+2,–2,–2,+2	+2,+2,–1,–1	–1,+2,+2,–1	–1,+2,–1,+2
SSD1	XXXXXX	+2,+2,+2,+2	—	—	—
SSD2	XXXXXX	+2,+2,+2,–2	—	—	—
ESD1	XXXXXX	+2,+2,+2,+2	—	—	—
ESD2_Ext_0	XXXXXX	+2,+2,+2,–2	—	—	—
ESD2_Ext_1	XXXXXX	+2,+2, –2,+2	—	—	—
ESD2_Ext_2	XXXXXX	+2,–2,+2,+2	—	—	—
ESD_Ext_Err	XXXXXX	–2,+2,+2,+2	—	—	—
Idle/Carrier Extension	000000	0, 0, 0, 0	—	—	—
Idle/Carrier Extension	000001	–2, 0, 0, 0	—	—	—
Idle/Carrier Extension	000010	0,–2, 0, 0	—	—	—
Idle/Carrier Extension	000011	–2,–2, 0, 0	—	—	—
Idle/Carrier Extension	000100	0, 0,–2, 0	—	—	—
Idle/Carrier Extension	000101	–2, 0,–2, 0	—	—	—
Idle/Carrier Extension	000110	0,–2,–2, 0	—	—	—
Idle/Carrier Extension	000111	–2,–2,–2, 0	—	—	—
Idle/Carrier Extension	001000	0, 0, 0,–2	—	—	—
Idle/Carrier Extension	001001	–2, 0, 0,–2	—	—	—
Idle/Carrier Extension	001010	0,–2, 0,–2	—	—	—
Idle/Carrier Extension	001011	–2,–2, 0,–2	—	—	—
Idle/Carrier Extension	001100	0, 0,–2,–2	—	—	—

Table 40–1—Bit-to-symbol mapping (even subsets) ***(continued)***

		$Sd_n[6:8]$ = [000]	$Sd_n[6:8]$ = [010]	$Sd_n[6:8]$ = [100]	$Sd_n[6:8]$ = [110]
Condition	$Sd_n[5:0]$	TA_n,TB_n,TC_n,TD_n	TA_n,TB_n,TC_n,TD_n	TA_n,TB_n,TC_n,TD_n	TA_n,TB_n,TC_n,TD_n
Idle/Carrier Extension	001101	–2, 0,–2,–2	—	—	—
Idle/Carrier Extension	001110	0,–2,–2,–2	—	—	—
Idle/Carrier Extension	001111	–2,–2,–2,–2	—	—	—

Table 40–2—Bit-to-symbol mapping (odd subsets)

		$Sd_n[6:8]$ = [001]	$Sd_n[6:8]$ = [011]	$Sd_n[6:8]$ = [101]	$Sd_n[6:8]$ = [111]
Condition	$Sd_n[5:0]$	TA_n,TB_n,TC_n,TD_n	TA_n,TB_n,TC_n,TD_n	TA_n,TB_n,TC_n,TD_n	TA_n,TB_n,TC_n,TD_n
Normal	000000	0, 0, 0,+1	0, 0,+1, 0	0,+1,+1,+1	0,+1, 0, 0
Normal	000001	–2, 0, 0,+1	–2, 0,+1, 0	–2,+1,+1,+1	–2,+1, 0, 0
Normal	000010	0,–2, 0,+1	0,–2,+1, 0	0,–1,+1,+1	0,–1, 0, 0
Normal	000011	–2,–2, 0,+1	–2,–2,+1, 0	–2,–1,+1,+1	–2,–1, 0, 0
Normal	000100	0, 0,–2,+1	0, 0,–1, 0	0,+1,–1,+1	0,+1,–2, 0
Normal	000101	–2, 0,–2,+1	–2, 0,–1, 0	–2,+1,–1,+1	–2,+1,–2, 0
Normal	000110	0,–2,–2,+1	0,–2,–1, 0	0,–1,–1,+1	0,–1,–2, 0
Normal	000111	–2,–2,–2,+1	–2,–2,–1, 0	–2,–1,–1,+1	–2,–1,–2, 0
Normal	001000	0, 0, 0,–1	0, 0,+1,–2	0,+1,+1,–1	0,+1, 0,–2
Normal	001001	–2, 0, 0,–1	–2, 0,+1,–2	–2,+1,+1,–1	–2,+1, 0,–2
Normal	001010	0,–2, 0,–1	0,–2,+1,–2	0,–1,+1,–1	0,–1, 0,–2
Normal	001011	–2,–2, 0,–1	–2,–2,+1,–2	–2,–1,+1,–1	–2,–1, 0,–2
Normal	001100	0, 0,–2,–1	0, 0,–1,–2	0,+1,–1,–1	0,+1,–2,–2
Normal	001101	–2, 0,–2,–1	–2, 0,–1,–2	–2,+1,–1,–1	–2,+1,–2,–2
Normal	001110	0,–2,–2,–1	0,–2,–1,–2	0,–1,–1,–1	0,–1,–2,–2
Normal	001111	–2,–2,–2,–1	–2,–2,–1,–2	–2,–1,–1,–1	–2,–1,–2,–2
Normal	010000	+1,+1,+1, 0	+1,+1, 0,+1	+1, 0, 0, 0	+1, 0,+1,+1
Normal	010001	–1,+1,+1, 0	–1,+1, 0,+1	–1, 0, 0, 0	–1, 0,+1,+1
Normal	010010	+1,–1,+1, 0	+1,–1, 0,+1	+1,–2, 0, 0	+1,–2,+1,+1
Normal	010011	–1,–1,+1, 0	–1,–1, 0,+1	–1,–2, 0, 0	–1,–2,+1,+1
Normal	010100	+1,+1,–1, 0	+1,+1,–2,+1	+1, 0,–2, 0	+1, 0,–1,+1
Normal	010101	–1,+1,–1, 0	–1,+1,–2,+1	–1, 0,–2, 0	–1, 0,–1,+1

Table 40–2—Bit-to-symbol mapping (odd subsets) ***(continued)***

		$Sd_n[6:8]$ = [001]	$Sd_n[6:8]$ = [011]	$Sd_n[6:8]$ = [101]	$Sd_n[6:8]$ = [111]
Condition	$Sd_n[5:0]$	TA_n,TB_n,TC_n,TD_n	TA_n,TB_n,TC_n,TD_n	TA_n,TB_n,TC_n,TD_n	TA_n,TB_n,TC_n,TD_n
Normal	010110	+1,–1,–1, 0	+1,–1,–2,+1	+1,–2,–2, 0	+1,–2,–1,+1
Normal	010111	–1,–1,–1, 0	–1,–1,–2,+1	–1,–2,–2, 0	–1,–2,–1,+1
Normal	011000	+1,+1,+1,–2	+1,+1, 0,–1	+1, 0, 0,–2	+1, 0,+1,–1
Normal	011001	–1,+1,+1,–2	–1,+1, 0,–1	–1, 0, 0,–2	–1, 0,+1,–1
Normal	011010	+1,–1,+1,–2	+1,–1, 0,–1	+1,–2, 0,–2	+1,–2,+1,–1
Normal	011011	–1,–1,+1,–2	–1,–1, 0,–1	–1,–2, 0,–2	–1,–2,+1,–1
Normal	011100	+1,+1,–1,–2	+1,+1,–2,–1	+1, 0,–2,–2	+1, 0,–1,–1
Normal	011101	–1,+1,–1,–2	–1,+1,–2,–1	–1, 0,–2,–2	–1, 0,–1,–1
Normal	011110	+1,–1,–1,–2	+1,–1,–2,–1	+1,–2,–2,–2	+1,–2,–1,–1
Normal	011111	–1,–1,–1,–2	–1,–1,–2,–1	–1,–2,–2,–2	–1,–2,–1,–1
Normal	100000	+2, 0, 0,+1	+2, 0,+1, 0	+2,+1,+1,+1	+2,+1, 0, 0
Normal	100001	+2,–2, 0,+1	+2,–2,+1, 0	+2,–1,+1,+1	+2,–1, 0, 0
Normal	100010	+2, 0,–2,+1	+2, 0,–1, 0	+2,+1,–1,+1	+2,+1,–2, 0
Normal	100011	+2,–2,–2,+1	+2,–2,–1, 0	+2,–1,–1,+1	+2,–1,–2, 0
Normal	100100	+2, 0, 0,–1	+2, 0,+1,–2	+2,+1,+1,–1	+2,+1, 0,–2
Normal	100101	+2,–2, 0,–1	+2,–2,+1,–2	+2,–1,+1,–1	+2,–1, 0,–2
Normal	100110	+2, 0,–2,–1	+2, 0,–1,–2	+2,+1,–1,–1	+2,+1,–2,–2
Normal	100111	+2,–2,–2,–1	+2,–2,–1,–2	+2,–1,–1,–1	+2,–1,–2,–2
Normal	101000	0, 0,+2,+1	+1,+1,+2,+1	+1, 0,+2, 0	0,+1,+2, 0
Normal	101001	–2, 0,+2,+1	–1,+1,+2,+1	–1, 0,+2, 0	–2,+1,+2, 0
Normal	101010	0,–2,+2,+1	+1,–1,+2,+1	+1,–2,+2, 0	0,–1,+2, 0
Normal	101011	–2,–2,+2,+1	–1,–1,+2,+1	–1,–2,+2, 0	–2,–1,+2, 0
Normal	101100	0, 0,+2,–1	+1,+1,+2,–1	+1, 0,+2,–2	0,+1,+2,–2
Normal	101101	–2, 0,+2,–1	–1,+1,+2,–1	–1, 0,+2,–2	–2,+1,+2,–2
Normal	101110	0,–2,+2,–1	+1,–1,+2,–1	+1,–2,+2,–2	0,–1,+2,–2
Normal	101111	–2,–2,+2,–1	–1,–1,+2,–1	–1,–2,+2,–2	–2,–1,+2,–2
Normal	110000	0,+2, 0,+1	0,+2,+1, 0	+1,+2, 0, 0	+1,+2,+1,+1
Normal	110001	–2,+2, 0,+1	–2,+2,+1, 0	–1,+2, 0, 0	–1,+2,+1,+1
Normal	110010	0,+2,–2,+1	0,+2,–1, 0	+1,+2,–2, 0	+1,+2,–1,+1
Normal	110011	–2,+2,–2,+1	–2,+2,–1, 0	–1,+2,–2, 0	–1,+2,–1,+1
Normal	110100	0,+2, 0,–1	0,+2,+1,–2	+1,+2, 0,–2	+1,+2,+1,–1

Table 40–2—Bit-to-symbol mapping (odd subsets) *(continued)*

		$Sd_n[6:8] = [001]$	$Sd_n[6:8] = [011]$	$Sd_n[6:8] = [101]$	$Sd_n[6:8] = [111]$
Condition	$Sd_n[5:0]$	TA_n,TB_n,TC_n,TD_n	TA_n,TB_n,TC_n,TD_n	TA_n,TB_n,TC_n,TD_n	TA_n,TB_n,TC_n,TD_n
Normal	110101	–2,+2, 0,–1	–2,+2,+1,–2	–1,+2, 0,–2	–1,+2,+1,–1
Normal	110110	0,+2,–2,–1	0,+2,–1,–2	+1,+2,–2,–2	+1,+2,–1,–1
Normal	110111	–2,+2,–2,–1	–2,+2,–1,–2	–1,+2,–2,–2	–1,+2,–1,–1
Normal	111000	+1,+1,+1,+2	0, 0,+1,+2	+1, 0, 0,+2	0,+1, 0,+2
Normal	111001	–1,+1,+1,+2	–2, 0,+1,+2	–1, 0, 0,+2	–2,+1, 0,+2
Normal	111010	+1,–1,+1,+2	0,–2,+1,+2	+1,–2, 0,+2	0,–1, 0,+2
Normal	111011	–1,–1,+1,+2	–2,–2,+1,+2	–1,–2, 0,+2	–2,–1, 0,+2
Normal	111100	+1,+1,–1,+2	0, 0,–1,+2	+1, 0,–2,+2	0,+1,–2,+2
Normal	111101	–1,+1,–1,+2	–2, 0,–1,+2	–1, 0,–2,+2	–2,+1,–2,+2
Normal	111110	+1,–1,–1,+2	0,–2,–1,+2	+1,–2,–2,+2	0,–1,–2,+2
Normal	111111	–1,–1,–1,+2	–2,–2,–1,+2	–1,–2,–2,+2	–2,–1,–2,+2
xmt_err	XXXXXX	+2,+2, 0,+1	0,+2,+1,+2	+1,+2,+2, 0	+2,+1,+2, 0
CSExtend_Err	XXXXXX	+2,+2, –2,–1	–2,+2,–1,+2	–1,+2,+2,–2	+2,–1,+2,–2
CSExtend	XXXXXX	+2, 0,+2,+1	+2, 0,+1,+2	+1, 0,+2,+2	+2,+1, 0,+2
CSReset	XXXXXX	+2,–2,+2,–1	+2,–2,–1,+2	–1,–2,+2,+2	+2,–1,–2,+2

40.3.1.3.6 Generation of A_n, B_n, C_n, D_n

The four bits $Sg_n[3:0]$ are used to randomize the signs of the quinary symbols (A_n, B_n, C_n, D_n) so that each symbol stream has no dc bias. The bits are used to generate binary symbols (SnA_n, SnB_n, SnC_n, SnD_n) that, when multiplied by the quinary symbols (TA_n, TB_n, TC_n, TD_n), result in (A_n, B_n, C_n, D_n).

PCS Transmit ensures a distinction between code-groups transmitted during idle mode plus SSD and those transmitted during other symbol periods. This distinction is accomplished by reversing the mapping of the sign bits when the condition (tx_enable_{n-2} + tx_enable_{n-4}) = 1. This sign reversal is controlled by the variable $Srev_n$ defined as

$$Srev_n = tx\_enable_{n-2} + tx\_enable_{n-4}$$

The binary symbols SnA_n, SnB_n, SnC_n, and SnD_n are defined using $Sg_n[3:0]$ as

$$SnA_n = \begin{cases} +1 \text{ if } [(Sg_n[0] \wedge Srev_n) = 0] \\ -1 \text{ else} \end{cases}$$

$$SnB_n = \begin{cases} +1 \text{ if } [(Sg_n[1] \wedge Srev_n) = 0] \\ -1 \text{ else} \end{cases}$$

$$SnC_n = \begin{cases} +1 \text{ if } [(Sg_n[2] \wedge Srev_n) = 0] \\ -1 \text{ else} \end{cases}$$

$$SnD_n = \begin{cases} +1 \text{ if } [(Sg_n[3] \wedge Srev_n) = 0] \\ -1 \text{ else} \end{cases}$$

The quinary symbols (A_n, B_n, C_n, D_n) are generated as the product of (SnA_n, SnB_n, SnC_n, SnD_n) and (TA_n, TB_n, TC_n, TD_n) respectively.

$$A_n = TA_n \times SnA_n$$

$$B_n = TB_n \times SnB_n$$

$$C_n = TC_n \times SnC_n$$

$$D_n = TD_n \times SnD_n$$

40.3.1.4 PCS Receive function

The PCS Receive function shall conform to the PCS Receive state diagram in Figure 40–11a including compliance with the associated state variables as specified in 40.3.3.

The PCS Receive function accepts received code-groups provided by the PMA Receive function via the parameter rx_symb_vector. To achieve correct operation, PCS Receive uses the knowledge of the encoding rules that are employed in the idle mode. PCS Receive generates the sequence of vectors of four quinary symbols (RA_n, RB_n, RC_n, RD_n) and indicates the reliable acquisition of the descrambler state by setting the parameter scr_status to OK. The sequence (RA_n, RB_n, RC_n, RD_n) is processed to generate the signals RXD<7:0>, RX_DV, and RX_ER, which are presented to the GMII. PCS Receive detects the transmission of a stream of data from the remote station and conveys this information to the PCS Carrier Sense and PCS Transmit functions via the parameter 1000BTreceive.

When the PHY supports the optional EEE capability, the PCS Receive uses the knowledge of the encoding rules that are employed in the idle mode to derive the values of the variables rem_lpi_req and rem_update_done.

40.3.1.4.1 Decoding of code-groups

When the PMA indicates that correct receiver operation has been achieved by setting the loc_rcvr_status parameter to the value OK, the PCS Receive continuously checks that the received sequence satisfies the encoding rule used in idle mode. When a violation is detected, PCS Receive assigns the value TRUE to the parameter 1000BTreceive and, by examining the last two received vectors (RA_{n-1}, RB_{n-1}, RC_{n-1}, RD_{n-1}) and (RA_n, RB_n, RC_n, RD_n), determines whether the violation is due to reception of SSD or to a receiver error.

Upon detection of SSD, PCS Receive also assigns the value TRUE to the parameter 1000BTreceive that is provided to the PCS Carrier Sense and Collision Presence functions. During the two symbol periods corresponding to SSD, PCS Receive replaces SSD by preamble bits. Upon the detection of SSD, the signal RX_DV is asserted and each received vector is decoded into a data octet RXD<7:0> until ESD is detected.

Upon detection of a receiver error, the signal RX_ER is asserted and the parameter rxerror_status assumes the value ERROR. De-assertion of RX_ER and transition to the IDLE state (rxerror_status=NO_ERROR) takes place upon detection of four consecutive vectors satisfying the encoding rule used in idle mode.

During reception of a stream of data, PCS Receive checks that the symbols RA_n, RB_n, RC_n, RD_n follow the encoding rule defined in 40.3.1.3.5 for ESD whenever they assume values ± 2. PCS Receive processes two consecutive vectors at each time n to detect ESD. Upon detection of ESD, PCS Receive de-asserts the signal RX_DV on the GMII. If the last symbol period of ESD indicates that a carrier extension is present, PCS Receive will assert the RX_ER signal on the GMII. If no extension is indicated in the ESD2 quartet, PCS Receive assigns the value FALSE to the parameter receiving. If an extension is present, the transition to the IDLE state occurs after detection of a valid idle symbol period and the parameter receiving remains TRUE until check_idle is TRUE. If a violation of the encoding rules is detected, PCS Receive asserts the signal RX_ER for at least one symbol period.

A premature stream termination is caused by the detection of invalid symbols during the reception of a data stream. Then, PCS Receive waits for the reception of four consecutive vectors satisfying the encoding rule used in idle mode prior to de-asserting the error indication. Note that RX_DV remains asserted during the symbol periods corresponding to the first three idle vectors, while RX_ER=TRUE is signaled on the GMII. The signal RX_ER is also asserted in the LINK FAILED state, which ensures that RX_ER remains asserted for at least one symbol period.

40.3.1.4.2 Receiver descrambler polynomials

The PHY shall descramble the data stream and return the proper sequence of code-groups to the decoding process for generation of RXD<7:0> to the GMII. 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 $g'_S(x) = 1 + x^{13} + x^{33}$.

40.3.1.5 PCS Carrier Sense function

The PCS Carrier Sense function generates the GMII signal CRS, which the MAC uses for deferral in half duplex mode. The PCS shall conform to the Carrier Sense state diagram as depicted in Figure 40–12 including compliance with the associated state variables as specified in 40.3.3. The PCS Carrier Sense function is not required in a 1000BASE-T PHY that does not support half duplex operation.

40.3.1.6 PCS Local LPI Request function

The PCS Local LPI Request function generates the signal loc_lpi_req, which indicates to the PMA PHY Control function whether or not the local PHY is requested to enter the LPI mode. When the PHY supports the optional EEE capability, the PCS shall conform to the Local LPI Request state diagram as depicted in Figure 40–9 including compliance with the associated state variables as specified in 40.3.3.

40.3.2 Stream structure

The tx_symb_vector and rx-symb_vector structure is shown in Figure 40–7.

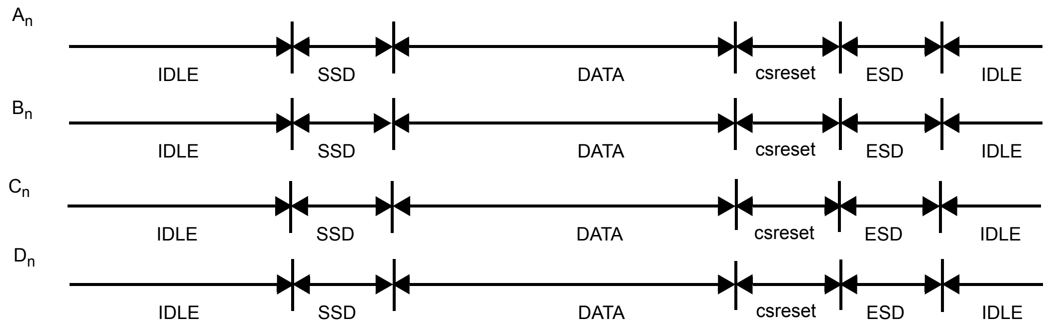


Figure 40–7—The tx_symb_vector and rx-symb_vector structure

40.3.3 State variables

40.3.3.1 Variables

CEXT
A sequence of vectors of four quinary symbols corresponding to the code-group generated in idle mode to denote carrier extension, as specified in 40.3.1.3.

CEXT_Err
A sequence of vectors of four quinary symbols corresponding to the code-group generated in idle mode to denote carrier extension with error indication, as specified in 40.3.1.3.

COL
The COL signal of the GMII as specified in 35.2.2.12.

config
The config parameter set by PMA and passed to the PCS via the PMA_CONFIG.indication primitive.
Values: MASTER, SLAVE.

CRS
The CRS signal of the GMII as specified in 35.2.2.11.

CSExtend
A vector of four quinary symbols corresponding to the code-group indicating convolutional encoder reset condition during carrier extension, as specified in 40.3.1.3.

CSExtend_Err
A vector of four quinary symbols corresponding to the code-group indicating convolutional encoder reset condition during carrier extension with error indication, as specified in 40.3.1.3.

CSReset
A vector of four quinary symbols corresponding to the code-group indicating convolutional encoder reset condition in the absence of carrier extension, as specified in 40.3.1.3.

DATA

A vector of four quinary symbols corresponding to the code-group indicating valid data, as specified in 40.3.1.3.

ESD1

A vector of four quinary symbols corresponding to the first code-group of End-of-Stream delimiter, as specified in 40.3.1.3.

ESD2_Ext_0

A vector of four quinary symbols corresponding to the second code-group of End-of-Stream delimiter in the absence of carrier extension over the two ESD symbol periods, as specified in 40.3.1.3.

ESD2_Ext_1

A vector of four quinary symbols corresponding to the second code-group of End-of-Stream delimiter when carrier extension is indicated during the first symbol period of the End-of-Stream delimiter, but not during the second symbol period, as specified in 40.3.1.3.

ESD2_Ext_2

A vector of four quinary symbols corresponding to the second code-group of End-of-Stream delimiter when carrier extension is indicated during the two symbol periods of the End-of-Stream delimiter, as specified in 40.3.1.3.

ESD_Ext_Err

A vector of four quinary symbols corresponding to either the first or second code-group of End-of-Stream delimiter when carrier extension with error is indicated during the End-of-Stream delimiter, as specified in 40.3.1.3.

IDLE

A sequence of vectors of four quinary symbols representing the special code-group generated in idle mode in the absence of carrier extension or carrier extension with error indication, as specified in 40.3.1.3.

link_status

The link_status parameter set by PMA Link Monitor and passed to the PCS via the PMA_LINK.indication primitive.
Values: OK or FAIL

loc_rcvr_status

The loc_rcvr_status parameter set by the PMA Receive function and passed to the PCS via the PMA_RXSTATUS.indication primitive.
Values: OK or NOT_OK

pcs_reset

The pcs_reset parameter set by the PCS Reset function.
Values: ON or OFF

(RA_n, RB_n, RC_n, RD_n)

The vector of the four correctly aligned most recently received quinary symbols generated by PCS Receive at time n.

1000BTreceive

The receiving parameter generated by the PCS Receive function.
Values: TRUE or FALSE

rem_rcvr_status
The rem_rcvr_status parameter generated by PCS Receive.
Values: OK or NOT_OK

repeater_mode
See 36.2.5.1.3

Rx_n
Alias for rx_symb_vector (a vector RA_n, RB_n, RC_n, RD_n) at time n.

rxerror_status
The rxerror_status parameter set by the PCS Receive function.
Values: ERROR or NO_ERROR

RX_DV
The RX_DV signal of the GMII as specified in 35.2.2.7.

RX_ER
The RX_ER signal of the GMII as specified in 35.2.2.9.

rx_symb_vector
A vector of four quinary symbols received by the PMA and passed to the PCS via the PMA_UNITDATA.indication primitive.
Value: SYMB_4D

RXD[7:0]
The RXD<7:0> signal of the GMII as specified in 35.2.2.8.

SSD1
A vector of four quinary symbols corresponding to the first code-group of the Start-of-Stream delimiter, as specified in 40.3.1.3.5.

SSD2
A vector of four quinary symbols corresponding to the second code-group of the Start-of-Stream delimiter, as specified in 40.3.1.3.5.

1000BTtransmit
A Boolean used by the PCS Transmit Process to indicate whether a frame transmission is in progress. Also used by the Carrier Sense and Local LPI Request processes.
Values: TRUE: The PCS is transmitting a stream
FALSE: The PCS is not transmitting a stream

TXD[7:0]
The TXD<7:0> signal of the GMII as specified in 35.2.2.4.

tx_enable
The tx_enable parameter generated by PCS Transmit as specified in Figure 40–8.
Values: TRUE or FALSE

tx_error
The tx_error parameter generated by PCS Transmit as specified in Figure 40–8.
Values: TRUE or FALSE

TX_EN
: The TX_EN signal of the GMII as specified in 35.2.2.3.

TX_ER
: The TX_ER signal of the GMII as specified in 35.2.2.5.

tx_mode
: The tx_mode parameter set by the PMA PHY Control function and passed to the PCS via the PMA_TXMODE.indication primitive.
Values: SEND_Z, SEND_N, or SEND_I

Tx_n
: Alias for tx_symb_vector at time n.

tx_symb_vector
: A vector of four quinary symbols generated by the PCS Transmit function and passed to the PMA via the PMA_UNITDATA.request primitive.
Value: SYMB_4D

xmt_err
: A vector of four quinary symbols corresponding to a transmit error indication during normal data transmission or reception, as specified in 40.3.1.3.

The following state variables are only required for the optional EEE capability:

loc_lpi_req
: The loc_lpi_req variable is set by the PCS Local LPI Request function and indicates whether or not the local PHY is requested to enter the LPI mode. It is passed to the PMA PHY Control function via the PMA_LPIREQ.request primitive. In the absence of the optional EEE capability, the PHY shall operate as if the value of this variable is FALSE.
Values: TRUE or FALSE

lpi_mode
: The lpi_mode variable is generated by the PMA PHY Control function and indicates whether or not the local PHY has entered LPI mode. It is passed to the PCS Receive function via the PMA_LPIMODE.indication primitive. In the absence of the optional EEE capability, the PHY operates as if the value of this variable is OFF.
Values: ON or OFF

rem_lpi_req
: The rem_lpi_req variable is generated by the PCS Receive function and indicates whether or not the remote PHY is requesting entry into LPI mode. It is passed to the PMA PHY Control function via the PMA_REMLPIREQ.request primitive. In the absence of the optional EEE capability, the PHY shall operate as if the value of this variable is FALSE.
Values: TRUE or FALSE

40.3.3.2 Functions

check_end
: A function used by the PCS Receive process to detect the reception of valid ESD symbols. The check_end function operates on the next two rx_symb_vectors, (Rx_{n+1}) and (Rx_{n+2}), available via PMA_UNITDATA.indication, and returns a Boolean value indicating whether these two consecutive vecctors contain symbols corresponding to a legal ESD encoding or not, as specified in 40.3.1.3.

check_idle
A function used by the PCS Receive process to detect the reception of valid idle code-groups after an error condition during the process. The check_idle function operates on the current rx_symb_vector and the next three rx_symb_vectors available via PMA_UNITDATA.indication and
returns a Boolean value indicating whether the four consecutive vectors contain symbols corresponding to the idle mode encoding or not, as specified in 40.3.1.3.

DECODE
In the PCS Receive process, this function takes as its argument the value of rx_symb_vector and returns the corresponding GMII RXD<7:0> octet. DECODE follows the rules outlined in 40.2.6.1.

ENCODE
In the PCS Transmit process, this function takes as its argument GMII TXD <7:0> and returns the corresponding tx_symb_vector. ENCODE follows the rules outlined in 40.2.5.1.

40.3.3.3 Timer

symb_timer
Continuous timer:The condition symb_timer_done becomes true upon timer expiration.

Restart time: Immediately after expiration; timer restart resets the condition symb_timer_done.

Duration: 8 ns nominal. (See clock tolerance in 40.6.1.2.6.)

Symb-timer shall be generated synchronously with TX_TCLK. In the PCS Transmit state diagram, the message PMA_UNITDATA.request is issued concurrently with symb_timer_done.

40.3.3.4 Messages

PMA_UNITDATA.indication (rx_symb_vector)
A signal sent by PMA Receive indicating that a vector of four quinary symbols is available in rx_symb_vector. (See 40.2.6.)

PMA_UNITDATA.request (tx_symb_vector)
A signal sent to PMA Transmit indicating that a vector of four quinary symbols is available in tx_symb_vector. (See 40.2.5.)

PUDI
Alias for PMA_UNITDATA.indication (rx_symb_vector).

PUDR
Alias for PMA_UNITDATA.request (tx_symb_vector).

STD
Alias for symb_timer_done.

40.3.4 State diagrams

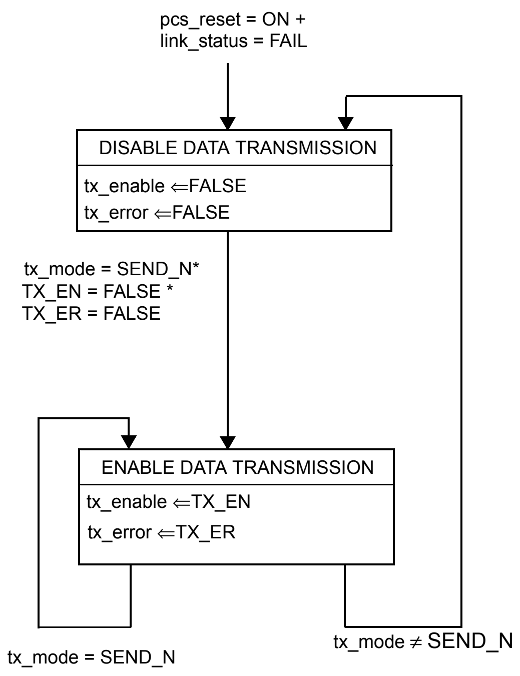


Figure 40–8—PCS Data Transmission Enabling state diagram

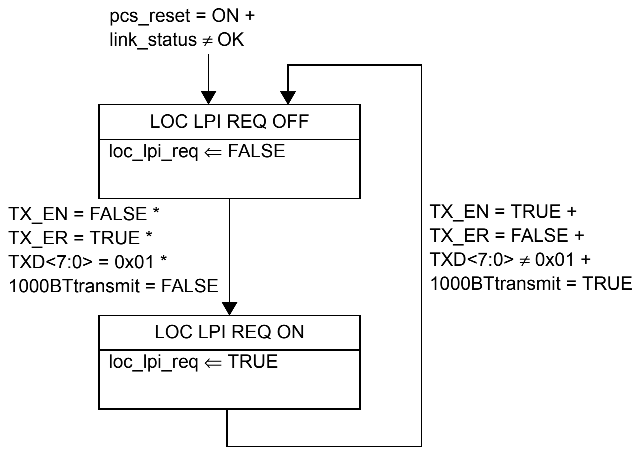


Figure 40–9—PCS Local LPI Request state diagram (optional)

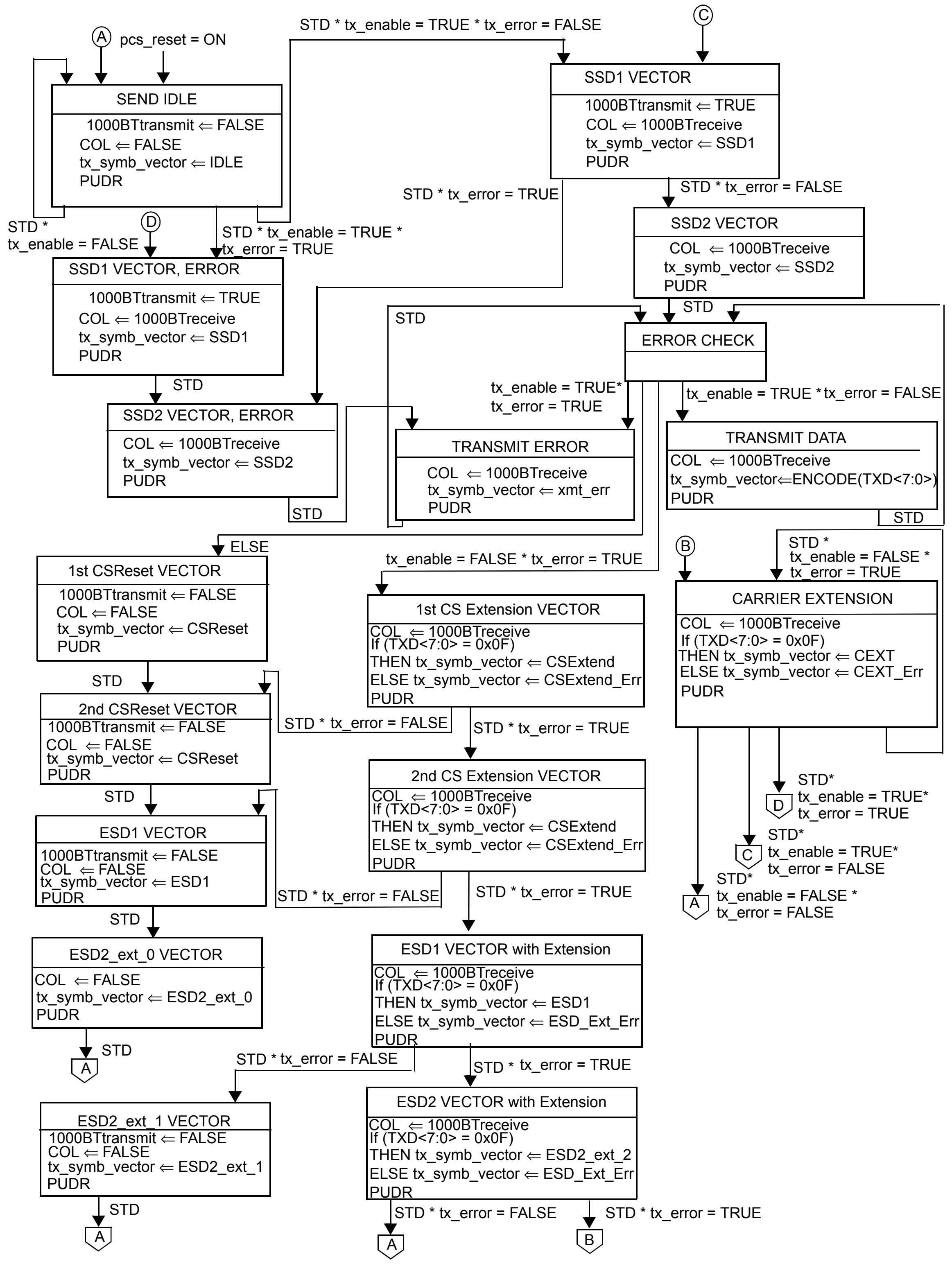


Figure 40–10—PCS Transmit state diagram

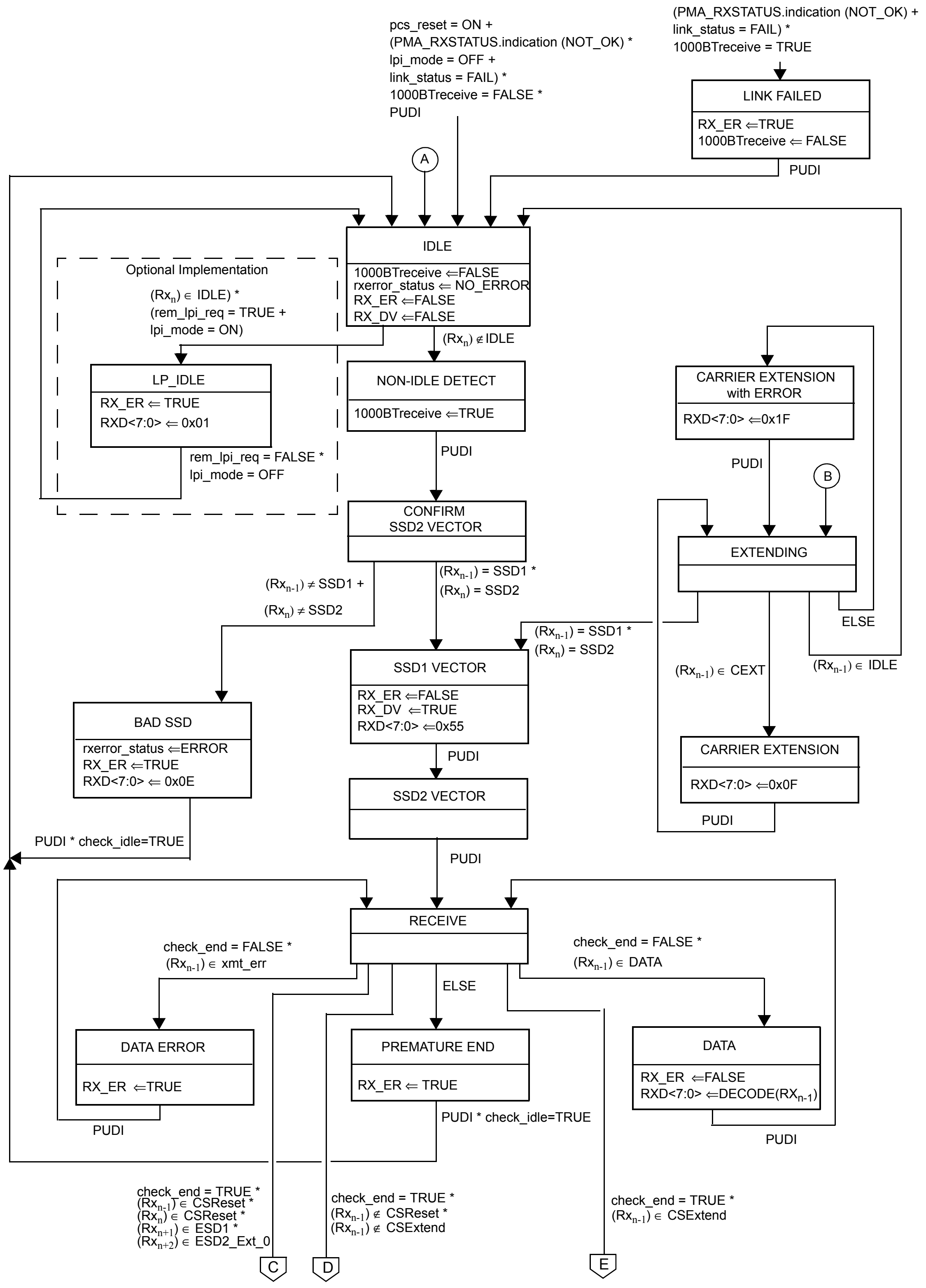


Figure 40–11a—PCS Receive state diagram, part a

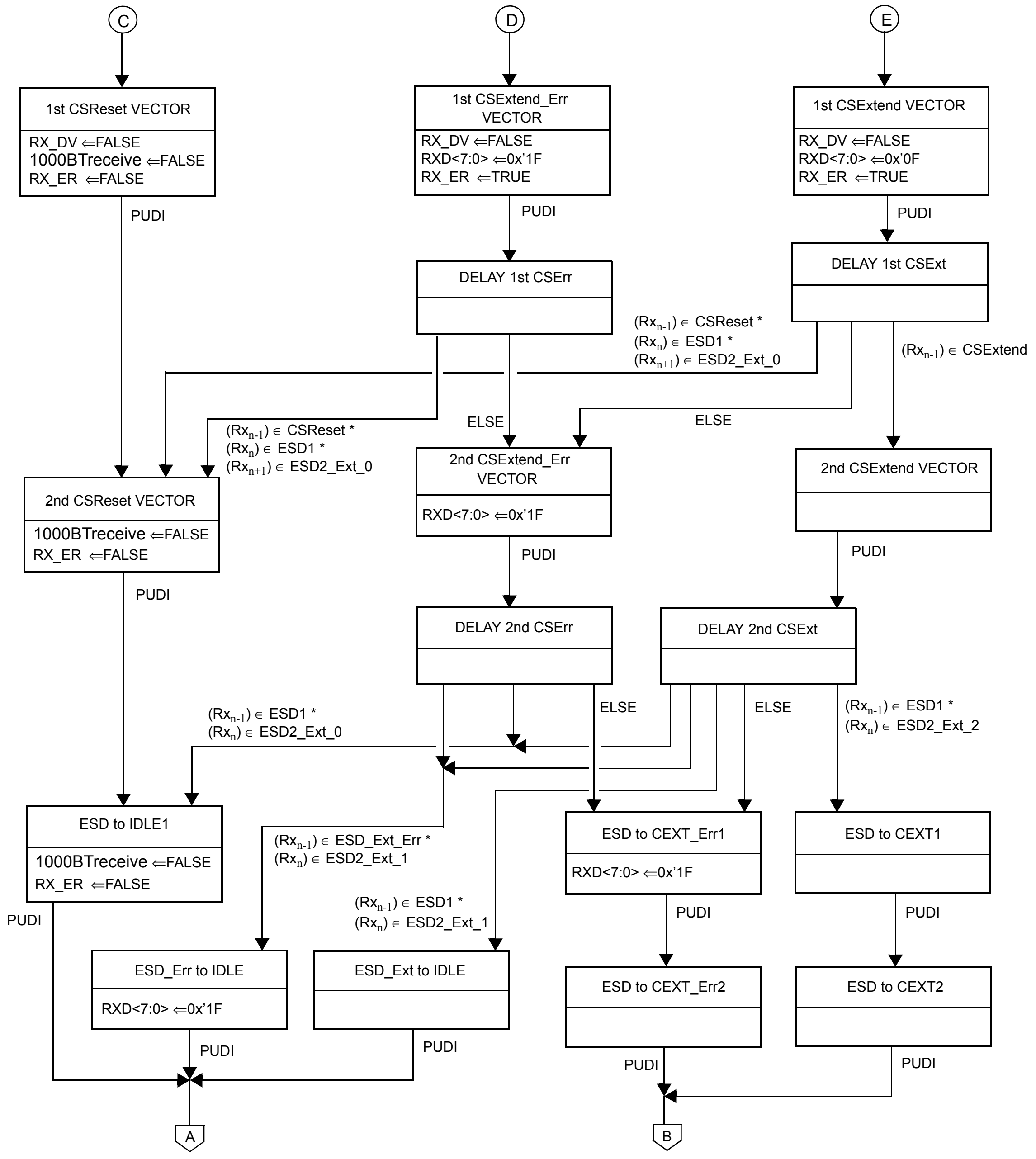


Figure 40–11b—PCS Receive state diagram, part b

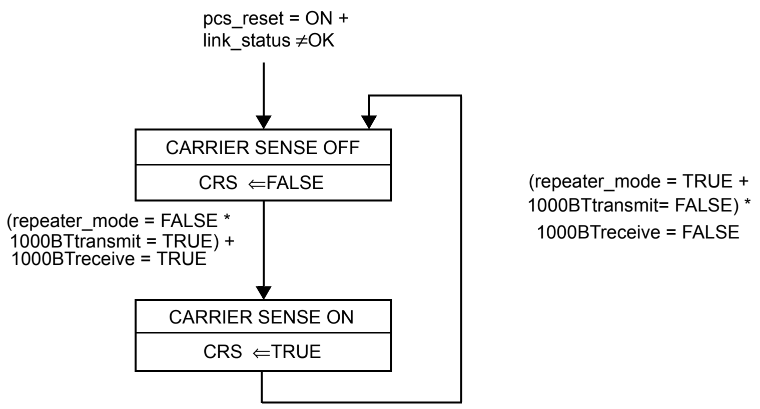


Figure 40–12—PCS Carrier Sense state diagram

40.3.4.1 Supplement to state diagram

Figure 40–13 reiterates the information shown in Figure 40–10 in timing diagram format. It is informative only. Time proceeds from left to right in the figure.

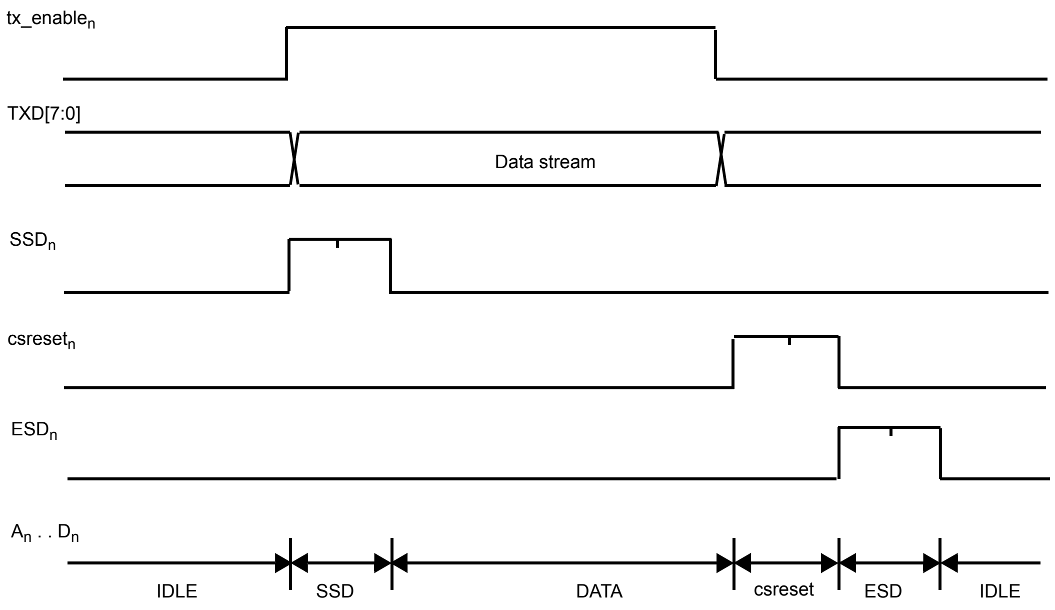


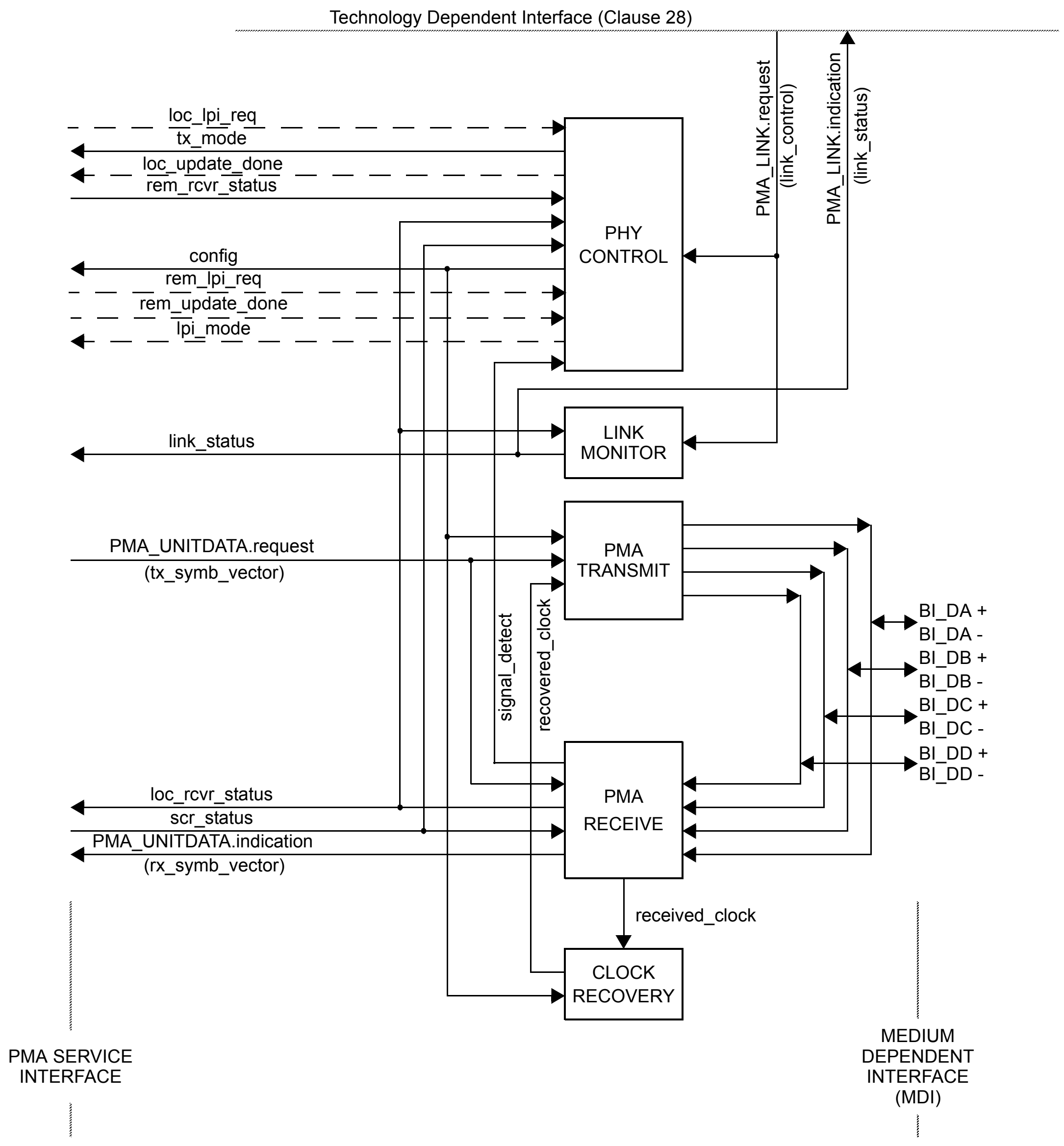
Figure 40–13—PCS sublayer to PMA timing

40.4 Physical Medium Attachment (PMA) sublayer

40.4.1 PMA functional specifications

The PMA couples messages from a PMA service interface specified in 40.2.2 to the 1000BASE-T baseband medium, specified in 40.7.

The interface between PMA and the baseband medium is the Medium Dependent Interface (MDI), which is specified in 40.8.



NOTE 1—The recovered_clock arc is shown to indicate delivery of the received clock signal back the PMA TRANSMIT for loop timing.

NOTE 2—Signals and functions shown with dashed lines are only required for the EEE capability.

Figure 40–14—PMA reference diagram

40.4.2 PMA functions

The PMA sublayer comprises one PMA Reset function and five simultaneous and asynchronous operating functions. The PMA operating functions are PHY Control, PMA Transmit, PMA Receive, Link Monitor, and Clock Recovery. All operating functions are started immediately after the successful completion of the PMA Reset function.

The PMA reference diagram, Figure 40–14, shows how the operating functions relate to the messages of the PMA Service interface and the signals of the MDI. Connections from the management interface, comprising the signals MDC and MDIO, to other layers are pervasive and are not shown in Figure 40–14. The management interface and its functions are specified in Clause 22.

40.4.2.1 PMA Reset function

The PMA Reset function shall be executed whenever one of the two following conditions occur:

a) Power on (see 36.2.5.1.3)
b) The receipt of a request for reset from the management entity

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

40.4.2.2 PMA Transmit function

The PMA Transmit function comprises four synchronous transmitters to generate four 5-level pulse-amplitude modulated signals on each of the four pairs BI_DA, BI_DB, BI_DC, and BI_DD. PMA Transmit shall continuously transmit onto the MDI pulses modulated by the quinary symbols given by tx_symb_vector[BI_DA], tx_symb_vector[BI_DB], tx_symb_vector[BI_DC] and tx_symb_vector[BI_DD], respectively. The four transmitters shall be driven by the same transmit clock, TX_TCLK. The signals generated by PMA Transmit shall follow the mathematical description given in 40.4.3.1, and shall comply with the electrical specifications given in 40.6.

When the PMA_CONFIG.indication parameter config is MASTER, the PMA Transmit function shall source TX_TCLK from a local clock source while meeting the transmit jitter requirements of 40.6.1.2.5. When the PMA_CONFIG.indication parameter config is SLAVE, the PMA Transmit function shall source TX_TCLK from the recovered clock of 40.4.2.6 while meeting the jitter requirements of 40.6.1.2.5.

40.4.2.3 PMA Receive function

The PMA Receive function comprises four independent receivers for quinary pulse-amplitude modulated signals on each of the four pairs BI_DA, BI_DB, BI_DC, and BI_DD. PMA Receive contains the circuits necessary to both detect quinary symbol sequences from the signals received at the MDI over receive pairs BI_DA, BI_DB, BI_DC, and BI_DD and to present these sequences to the PCS Receive function. The signals received at the MDI are described mathematically in 40.4.3.2. The PMA shall translate the signals received on pairs BI_DA, BI_DB, BI_DC, and BI_DB into the PMA_UNITDATA.indication parameter rx_symb_vector with a symbol error ratio of less than 10^{-10} over a channel meeting the requirements of 40.7.

To achieve the indicated performance, it is highly recommended that PMA Receive include the functions of signal equalization, echo and crosstalk cancellation, and sequence estimation. The sequence of code-groups assigned to tx_symb_vector is needed to perform echo and self near-end crosstalk cancellation.

The PMA Receive function uses the scr_status parameter and the state of the equalization, cancellation, and estimation functions to determine the quality of the receiver performance, and generates the loc_rcvr_status variable accordingly. The precise algorithm for generation of loc_rcvr_status is implementation dependent.

40.4.2.4 PHY Control function

PHY Control generates the control actions that are needed to bring the PHY into a mode of operation during which frames can be exchanged with the link partner. PHY Control shall comply with the state diagram description given in Figure 40–16a and Figure 40–16b.

During Auto-Negotiation PHY Control is in the DISABLE 1000BASE-T TRANSMITTER state and the transmitters are disabled. When the Auto-Negotiation process asserts link_control=ENABLE, PHY Control enters the SLAVE SILENT state. Upon entering this state, the maxwait timer is started and PHY Control forces transmission of zeros by setting tx_mode=SEND_Z. The transition out of the SLAVE SILENT state depends on whether the PHY is operating in MASTER or SLAVE mode. In MASTER mode, PHY Control transitions immediately to the TRAINING state. In SLAVE mode, PHY Control transitions to the TRAINING state only after the SLAVE PHY converges its decision feedback equalizer (DFE), acquires timing, and acquires its descrambler state, and sets scr_status=OK**.**

For the SLAVE PHY, the final convergence of the adaptive filter parameters is completed in the TRAINING state. The MASTER PHY performs all its receiver convergence functions in the TRAINING state. Upon entering the TRAINING state, the minwait_timer is started and PHY Control forces transmission into the idle mode by asserting tx_mode=SEND_I. After the PHY completes successful training and establishes proper receiver operations, PCS Transmit conveys this information to the link partner via transmission of the parameter loc_rcvr_status. (See $Sd_n[2]$ in 40.3.1.3.4.) The link partner's value for loc_rcvr_status is stored in the local device parameter rem_rcvr status. When the minwait_timer expires and the condition loc_rcvr_status=OK is satisfied, PHY Control transitions into either the SEND IDLE OR DATA state if rem_rcvr_status=OK or the SEND IDLE state if rem_rcvr_status=NOT_OK. On entry into either the SEND IDLE or SEND IDLE OR DATA states, the maxwait_timer is stopped and the minwait_timer is started.

The normal mode of operation corresponds to the SEND IDLE OR DATA state, where PHY Control asserts tx_mode=SEND_N and transmission of data over the link can take place. In this state, when no frames have to be sent, idle transmission takes place.

If unsatisfactory receiver operation is detected in the SEND IDLE OR DATA or SEND IDLE states (loc_rcvr_status=NOT_OK) and the minwait_timer has expired, transmission of the current frame is completed and PHY Control enters the SLAVE SILENT state. In the SEND IDLE OR DATA state, whenever a PHY that operates reliably detects unsatisfactory operation of the remote PHY (rem_rcvr_status=NOT_OK) and the minwait_timer has expired, it enters the SEND IDLE state where tx_mode=SEND_I is asserted and idle transmission takes place. In this state, encoding is performed with the parameter loc_rcvr_status=OK. As soon as the remote PHY signals satisfactory receiver operation (rem_rcvr_status=OK) and the minwait_timer has expired, the SEND IDLE OR DATA state is entered.

When the PHY supports the optional EEE capability, PHY Control will transition to the LPI mode in response to concurrent requests for LPI mode from the local PHY (loc_lpi_req = TRUE) and remote PHY (rem_lpi_req = TRUE).

Upon activation of the LPI mode, the PHY Control asserts tx_mode = SEND_I for a period of time defined by lpi_update_timer, which allows the remote PHY to prepare for cessation of transmission. When lpi_update_timer expires, PHY Control transitions to the POST_UPDATE state, signals to the remote PHY that is has completed the update by setting loc_update_done = TRUE, and starts the lpi_postupdate_timer. When lpi_postupdate_timer expires, PHY Control transitions to the WAIT_QUIET state. If there is a request to wake (loc_lpi_req = FALSE or rem_lpi_req = FALSE) while in the POST_UPDATE state, PHY Control will wait for confirmation that the remote PHY has completed the update (rem_update_done = TRUE) and is prepared for cessation of transmission before proceeding to the WAIT_QUIET state.

Upon entry into the WAIT_QUIET state, PHY Control asserts tx_mode = SEND_Z and transmission ceases. During the WAIT_QUIET and QUIET states, the PHY may deactivate transmit and receive functions in order to conserve energy. However, in the WAIT_QUIET state, the PHY shall be capable of correctly decoding rem_lpi_req. The PHY will remain in the QUIET state no longer than the time implied by lpi_quiet_timer.

When lpi_quiet_timer expires, the PHY initiates a wake sequence. The wake sequence begins with a transition to the WAKE state where the PHY will transmit (tx_mode = SEND_I) for the period lpi_waketx_timer and simultaneously start a parallel timer, lpi_wakemz_timer. Since it is likely that transmit circuits were deactivated while in the QUIET state, this transmission is not expected to be compliant 1000BASE-T signaling, but rather of sufficient quality and duration to be detected by the remote PHY receiver and initiate the wake sequence in the remote PHY.

Upon expiration of lpi_waketx_timer, the PHY will enter the WAKE_SILENT state and cease transmission (tx_mode = SEND_Z). The PHY will remain in the WAKE_SILENT state until lpi_wakemz_timer has expired, at which point it is assumed that the transmitter circuits have stabilized and compliant 1000BASE-T signaling can be transmitted. At this point the MASTER transitions to the WAKE_TRAINING state and transmits to the SLAVE PHY.

The remaining wake sequence is essentially an accelerated training mode sequence leading to entry into the UPDATE state.

Once scrambler synchronization is achieved, the incoming value of rem_lpi_req can be determined. If the LPI mode is no longer requested by either the local or remote PHY, then both PHYs return to the SEND IDLE OR DATA state and the normal mode of operation (tx_mode = SEND_N). If both PHYs continue to request the LPI mode, then both PHYs remain in the UPDATE state and continue to transmit for a time defined by lpi_update_timer. This time is intended to allow the remote PHY to refresh its receiver state (e.g., timing recovery, adaptive filter coefficients) and thereby track long-term variation in the timing of the link or the underlying channel characteristics. If lpi_update_timer expires and both PHYs continue to request the LPI mode, then the PHY transitions to the POST_UPDATE state.

PHY Control may force the transmit scrambler state to be initialized to an arbitrary value by requesting the execution of the PCS Reset function defined in 40.3.1.1.

40.4.2.5 Link Monitor function

Link Monitor determines the status of the underlying receive channel and communicates it via the variable link_status. Failure of the underlying receive channel typically causes the PMA's clients to suspend normal operation.

The Link Monitor function shall comply with the state diagram of Figure 40–17.

Upon power on, reset, or release from power down, the Auto-Negotiation algorithm sets link_control=SCAN_FOR_CARRIER and, during this period, sends fast link pulses to signal its presence to a remote station. If the presence of a remote station is sensed through reception of fast link pulses, the Auto-Negotiation algorithm sets link_control=DISABLE and exchanges Auto-Negotiation information with the remote station. During this period, link_status=FAIL is asserted. If the presence of a remote 1000BASE-T station is established, the Auto-Negotiation algorithm permits full operation by setting link_control=ENABLE. As soon as reliable transmission is achieved, the variable link_status=OK is asserted, upon which further PHY operations can take place.

40.4.2.6 Clock Recovery function

The Clock Recovery function couples to all four receive pairs. It may provide independent clock phases for sampling the signals on each of the four pairs.

The Clock Recovery function shall provide clocks suitable for signal sampling on each line so that the symbol error ratio indicated in 40.4.2.3 is achieved. The received clock signal must be stable and ready for use when training has been completed (loc_rcvr_status=OK). The received clock signal is supplied to the PMA Transmit function by received_clock.

40.4.3 MDI

Communication through the MDI is summarized in 40.4.3.1 and 40.4.3.2.

40.4.3.1 MDI signals transmitted by the PHY

The quinary symbols to be transmitted by the PMA on the four pairs BI_DA, BI_DB, BI_DC, and BI_DD are denoted by tx_symb_vector[BI_DA], tx_symb_vector[BI_DB], tx_symb_vector[BI_DC], and tx_symb_vector[BI_DD], respectively. The modulation scheme used over each pair is 5-level Pulse Amplitude Modulation. PMA Transmit generates a pulse-amplitude modulated signal on each pair in the following form:

$$s(t) = \sum_{k} a_k h_1(t - kT)$$

In the above equation, a_k represents the quinary symbol from the set {2, 1, 0, –1, –2} to be transmitted at time kT, and $h_1(t)$ denotes the system symbol response at the MDI. This symbol response shall comply with the electrical specifications given in 40.6.

40.4.3.2 Signals received at the MDI

Signals received at the MDI can be expressed for each pair as pulse-amplitude modulated signals that are corrupted by noise as follows:

$$r(t) = \sum_{k} a_k h_2(t - kT) + w(t)$$

In this equation, $h_2(t)$ denotes the impulse response of the overall channel between the transmit symbol source and the receive MDI and $w(t)$ is a term that represents the contribution of various noise sources. The four signals received on pairs BI_DA, BI_DB, BI_DC, and BI_DD shall be processed within the PMA Receive function to yield the quinary received symbols rx_symb_vector[BI_DA], rx_symb_vector[BI_DB], rx_symb_vector[BI_DC], and rx_symb_vector[BI_DD].

40.4.4 Automatic MDI/MDI-X Configuration

Automatic MDI/MDI-X Configuration is intended to eliminate the need for crossover cables between similar devices. Implementation of an automatic MDI/MDI-X configuration is optional for 1000BASE-T devices. If an automatic configuration method is used, it shall comply with the following specifications. The assignment of pin-outs for a 1000BASE-T crossover function cable is shown in Table 40–12 in 40.8.

40.4.4.1 Description of Automatic MDI/MDI-X state diagram

The Automatic MDI/MDI-X state diagram facilitates switching the BI_DA(C)+ and BI_DA(C)– with the BI_DB(D)+ and BI_DB(D)– signals respectively prior to the auto-negotiation mode of operation so that FLPs can be transmitted and received in compliance with Clause 28 Auto-Negotiation specifications. The correct polarization of the crossover circuit is determined by an algorithm that controls the switching function. This algorithm uses an 11-bit linear feedback shift register (LFSR) to create a pseudo-random sequence that each end of the link uses to determine its proposed configuration. Upon making the selection to either MDI or MDI-X, the node waits for a specified amount of time while evaluating its receive channel to determine whether the other end of the link is sending link pulses or PHY-dependent data. If link pulses or PHY-dependent data are detected, it remains in that configuration. If link pulses or PHY-dependent data are not detected, it increments its LFSR and makes a decision to switch based on the value of the next bit. The state diagram does not move from one state to another while link pulses are being transmitted.

40.4.4.2 Pseudo-random sequence generator

One possible implementation of the pseudo-random sequence generator using a linear-feedback shift register is shown in Figure 40–15. The bits stored in the shift register delay line at time n are denoted by S[10:0]. At each sample period, the shift register is advanced by one bit and one new bit represented by S[0] is generated. Switch control is determined by S[10].

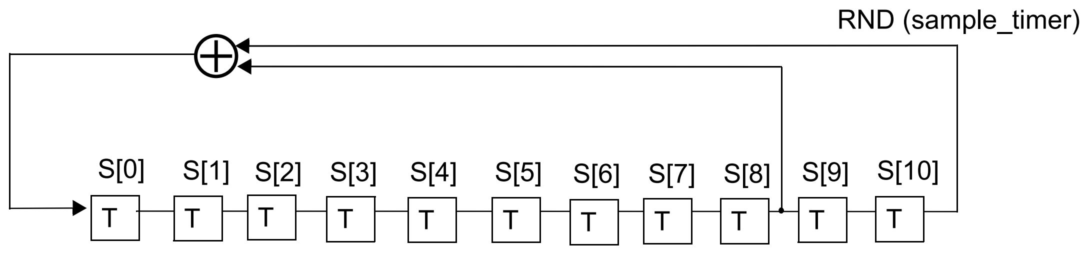


Figure 40–15—Automatic MDI/MDI-X linear-feedback shift register

40.4.5 State variables

40.4.5.1 State diagram variables

config
: The PMA shall generate this variable continuously and pass it to the PCS via the PMA_CONFIG.indication primitive.

Values: MASTER or SLAVE

link_control
: This variable is defined in 28.2.6.2.

Link_Det
: This variable indicates linkpulse = true or link_status = READY or OK has occurred at the receiver since the last time sample_timer has been started.

Values: TRUE: linkpulse = true or link_status = READY or OK has occurred since the last time sample_timer has been started.
FALSE: otherwise

linkpulse
: This variable is defined in 28.2.6.3.

link_status

This variable is defined in 28.2.6.1.

loc_rcvr_status

Variable set by the PMA Receive function to indicate correct or incorrect operation of the receive link for the local PHY.

Values: OK: The receive link for the local PHY is operating reliably.
NOT_OK: Operation of the receive link for the local PHY is unreliable.

MDI_Status

This variable defines the condition of the Automatic MDI/MDI-X physical connection.

Values: MDI: The BI_DA, BI_DB, BI_DC, and BI_DD pairs follow the connections as described in the MDI column of Table 40–12.

MDI-X: The BI_DA, BI_DB, BI_DC, and BI_DD pairs follow the connections as described in the MDI-X column of Table 40–12.

pma_reset

Allows reset of all PMA functions.

Values: ON or OFF

Set by: PMA Reset

rem_rcvr_status

Variable set by the PCS Receive function to indicate whether correct operation of the receive link for the remote PHY is detected or not.

Values: OK: The receive link for the remote PHY is operating reliably.

NOT_OK: Reliable operation of the receive link for the remote PHY is not detected.

RND (sample_timer)

This variable is defined as bit S[10] of the LSFR described in 40.4.4.2

scr_status

The scr_status parameter as communicated by the PMA_SCRSTATUS.request primitive.

Values: OK: The descrambler has achieved synchronization.

NOT_OK: The descrambler is not synchronized. Note that when the PHY supports the optional EEE capability and signal_detect is FALSE, scr_status is set to NOT_OK.

T_Pulse

This variable indicates that a linkpulse is being transmitted to the MDI.

Values: TRUE: Pulse being transmitted to the MDI

FALSE: Otherwise

tx_enable

The tx_enable parameter generated by PCS Transmit as specified in Figure 40–8.

Values: TRUE or FALSE as per 40.3.3.1.

tx_mode

PCS Transmit sends code-groups according to the value assumed by this variable.

Values: SEND_N: This value is continuously asserted when transmission of sequences of code-groups representing a GMII data stream, control information, or idle mode is to take place.

SEND_I: This value is continuously asserted when transmission of sequences of code-groups representing the idle mode is to take place.

SEND_Z: This value is asserted when transmission of zero code-groups is to take place.

The following state variables are only required for the optional EEE capability:

loc_lpi_req
The loc_lpi_req variable is set by the PCS Local LPI Request function and indicates whether or not the local PHY is requested to enter the LPI mode. It is passed to the PMA PHY Control function via the PMA_LPIREQ.request primitive. In the absence of the optional EEE capability, the PHY operates as if the value of this variable is FALSE.
Values: TRUE: "Assert LPI" is present at the GMII.
FALSE: "Assert LPI" is not present at the GMII.

loc_udpate_done
The loc_update_done variable is generated by the PMA PHY Control function and indicates whether or not the local PHY has completed the update of its receiver state. It is passed to the PCS Transmit function via the PMA_UPDATE.indication primitive. In the absence of the optional EEE capability, the PHY shall operate as if the value of this variable is FALSE.
Values: TRUE: The PHY has completed the current update.
FALSE: The PHY is ready for the next update or the current update is still in progress.

lpi_mode
The lpi_mode variable is generated by the PMA PHY Control function and indicates whether or not the local PHY has entered the LPI mode. It is passed to the PCS Receive function via the PMA_LPIMODE.indication primitive. In the absence of the optional EEE capability, the PHY shall operate as if the value of this variable is OFF.
Values: ON: The PHY is operating in LPI mode.
OFF: The PHY is in normal operation.

rem_lpi_req
The rem_lpi_req variable is generated by the PCS Receive function and indicates whether or not the remote PHY is requesting entry into LPI mode. It is passed to the PMA PHY Control function via the PMA_REMLPIREQ.request primitive. In the absence of the optional EEE capability, the PHY operates as if the value of this variable is FALSE.
Values: TRUE: LPI is encoded in the received symbols.
FALSE: LPI is not encoded in the received symbols.

rem_update_done
The rem_update_done variable is generated by the PCS Receive function and indicates whether or not the remote PHY has completed the update of its receiver state. It is passed to the PMA PHY Control function via the PMA_REMUPDATE.request primitive. In the absence of the optional EEE capability, the PHY shall operate as if the value of this variable is FALSE.
Values: TRUE: The remote PHY has completed the current update.
FALSE: The remote PHY is ready for the next update or the current update is still in progress.

signal_detect
The signal_detect variable is set by the PMA Receive function and indicates the presence of a signal at the MDI, as defined in 40.6.1.3.5.
Values: TRUE: There is a signal present at the MDI.
FALSE: There is no signal present at the MDI.

40.4.5.2 Timers

All timers operate in the manner described in 14.2.3.2 with the following addition. A timer is reset and stops counting upon entering a state where "stop timer" is asserted.

A_timer
: An asynchronous (to the Auto-Crossover state diagram) free-running timer that provides for a relatively arbitrary reset of the state diagram to its initial state. This timer is used to reduce the probability of a lock-up condition where both nodes have the same identical seed initialization at the same point in time.

  Values: The condition A_timer_done becomes true upon timer expiration.

  Duration: This timer shall have a period of 1.3 s ± 25%.

  Initialization of A_timer is implementation specific.

maxwait_timer
: A timer used to limit the amount of time during which a receiver dwells in the SLAVE SILENT and TRAINING states. The timer shall expire 750 ms ± 10 ms if config = MASTER or 350 ms ± 5 ms if config = SLAVE. This timer is used jointly in the PHY Control and Link Monitor state diagrams. The maxwait_timer is tested by the Link Monitor to force link_status to be set to FAIL if the timer expires and loc_rcvr_status is NOT_OK. See Figure 40–16a.

minwait_timer
: A timer used to determine the minimum amount of time the PHY Control stays in the TRAINING, SEND IDLE, or DATA states. The timer shall expire 1 µs ± 0.1µs after being started.

sample_timer
: This timer provides a long enough sampling window to ensure detection of Link Pulses or link_status, if they exist at the receiver.

  Values: The condition sample_timer_done becomes true upon timer expiration.

  Duration: This timer shall have a period of 62 ms ± 2 ms.

stabilize_timer
: A timer used to control the minimum time that loc_rcvr_status must be OK before a transition to Link Up can occur. The timer shall expire 1 µs ± 0.1 µs after being started.

The following timers are only required for the optional EEE capability:

lpi_link_fail_timer
: This timer defines the maximum time the PHY allows between entry into the WAKE state and subsequent entry into the UPDATE or SEND IDLE OR DATA states before forcing the link to restart.

  Values: The condition lpi_link_fail_timer_done becomes true upon timer expiration.
  Duration: This timer shall have a period between 90 µs and 110 µs.

lpi_postupdate_timer
: This timer defines the maximum time the PHY dwells in the POST_UPDATE state before proceeding to the WAIT_QUIET state.

  Values: The condition lpi_postupdate_timer_done becomes true upon timer expiration.
  Duration: This timer shall have a period between 2.0 µs and 3.2 µs.

lpi_quiet_timer

This timer defines the maximum time the PHY remains quiet before initiating transmission to refresh the remote PHY.

Values: The condition lpi_quiet_timer_done becomes true upon timer expiration.
Duration: This timer shall have a period between 20 ms and 24 ms.

lpi_waitwq_timer

This timer defines the maximum time the PHY dwells in the WAIT_QUIET state before forcing the link to restart.

Values: The condition lpi_waitwq_timer_done becomes true upon timer expiration.
Duration: This timer shall have a period between 10 µs and 12 µs.

lpi_wake_timer

This timer defines the expected time for the PHY to transition from the LPI mode to normal operation.

Values: The condition lpi_wake_timer_done becomes true upon timer expiration. For each transition of lpi_wake_timer_done from false to true, the wake error counter (see 40.5.1.1) shall be incremented.
Duration: This timer shall have a period that does not exceed 16.5 µs.

lpi_waketx_timer

This timer defines the time the PHY transmits to ensure detection by the remote PHY receiver and trigger an exit from the low power state.

Values: The condition lpi_waketx_timer_done becomes true upon timer expiration.
Duration: This timer shall have a period between 1.2 µs and 1.4 µs.

lpi_wakemz_timer

This timer defines the time allowed for the PHY transmitter to achieve compliant operation following activation.

Values: The condition lpi_wakemz_timer_done becomes true upon timer expiration.
Duration: This timer shall have a period between 4.25 µs and 5.00 µs.

lpi_update_timer

This timer defines the time the PHY transmits to facilitate a refresh of the remote PHY receiver.

Values: The condition lpi_update_timer_done becomes true upon timer expiration.
Duration: For a PHY configured as the MASTER, this timer shall have a period between 0.23 ms and 0.25 ms. For a PHY configured as the SLAVE, this timer shall have a period between 0.18 ms and 0.20 ms.

40.4.6 State Diagrams

40.4.6.1 PHY Control state diagram

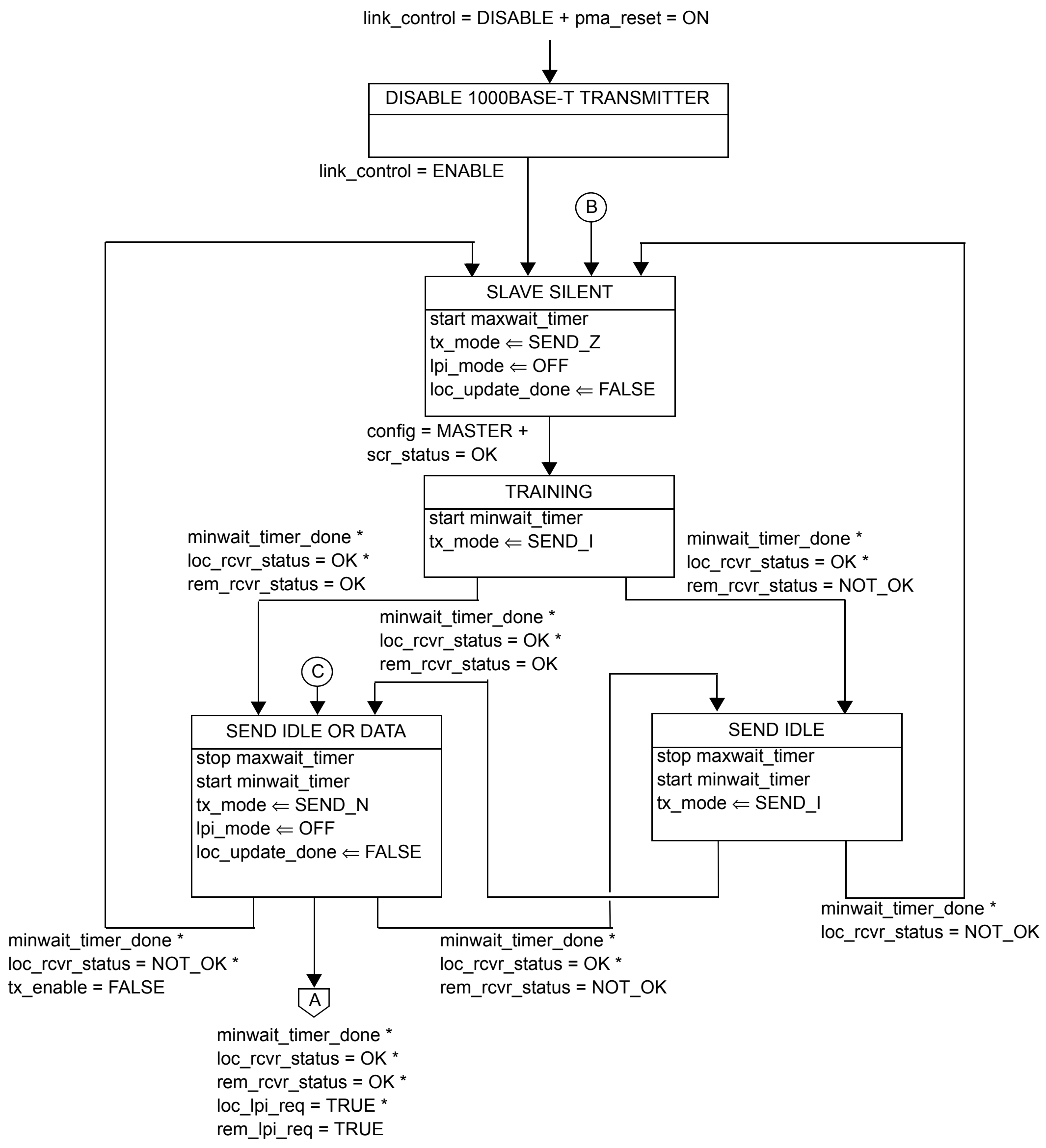


Figure 40–16a—PHY Control state diagram, part a

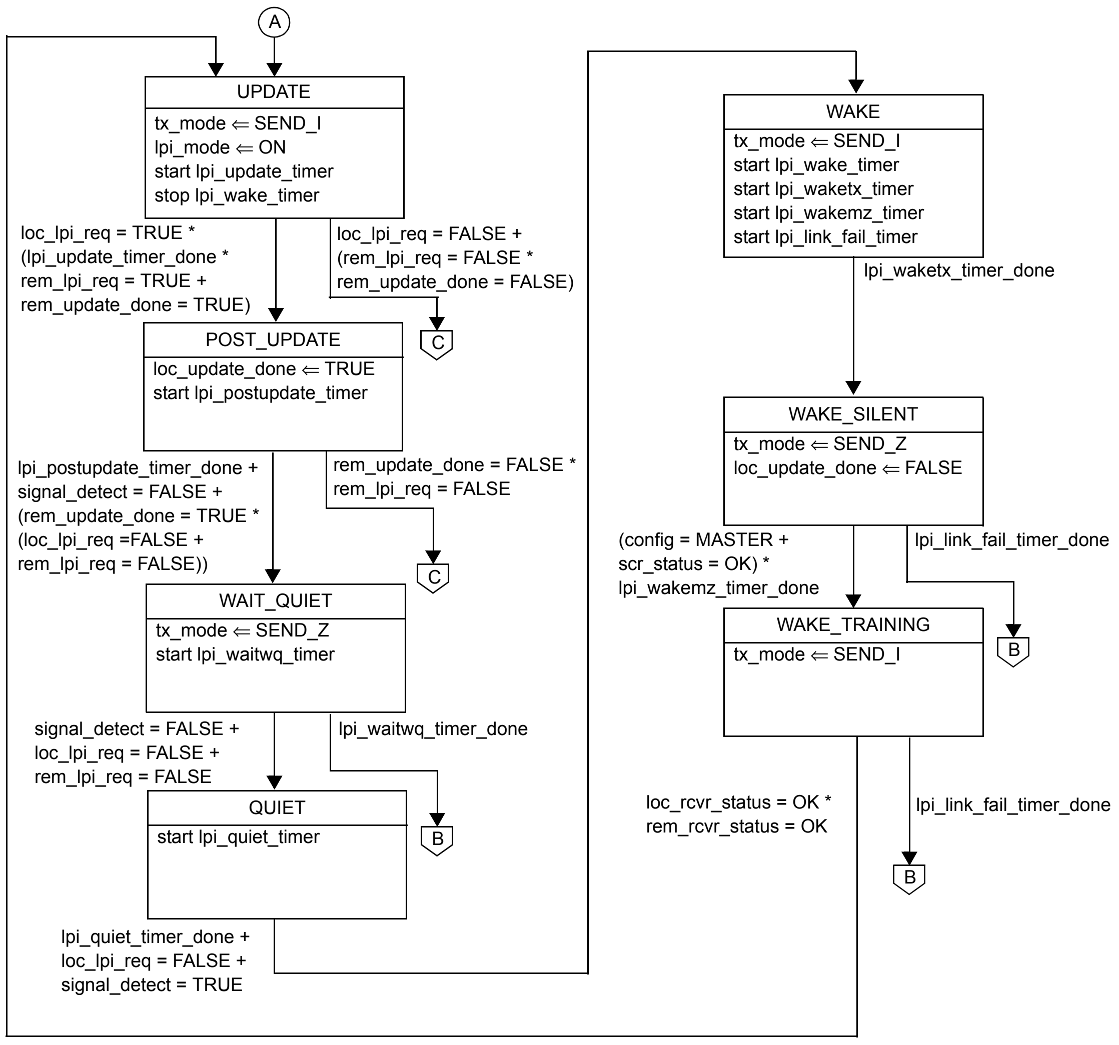
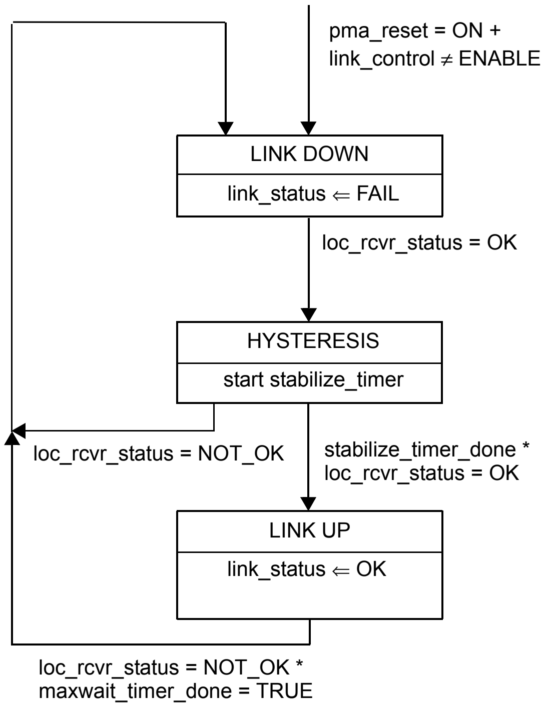

A
UPDATE
tx_mode ⇐ SEND_I
lpi_mode ⇐ ON
start lpi_update_timer
stop lpi_wake_timer
loc_lpi_req = TRUE * (lpi_update_timer_done * rem_lpi_req = TRUE + rem_update_done = TRUE)
loc_lpi_req = FALSE + (rem_lpi_req = FALSE * rem_update_done = FALSE)
C
POST_UPDATE
loc_update_done ⇐ TRUE
start lpi_postupdate_timer
lpi_postupdate_timer_done + signal_detect = FALSE + (rem_update_done = TRUE * (loc_lpi_req =FALSE + rem_lpi_req = FALSE))
rem_update_done = FALSE * rem_lpi_req = FALSE
C
WAIT_QUIET
tx_mode ⇐ SEND_Z
start lpi_waitwq_timer
signal_detect = FALSE + loc_lpi_req = FALSE + rem_lpi_req = FALSE
lpi_waitwq_timer_done
B
QUIET
start lpi_quiet_timer
lpi_quiet_timer_done + loc_lpi_req = FALSE + signal_detect = TRUE
WAKE
tx_mode ⇐ SEND_I
start lpi_wake_timer
start lpi_waketx_timer
start lpi_wakemz_timer
start lpi_link_fail_timer
lpi_waketx_timer_done
WAKE_SILENT
tx_mode ⇐ SEND_Z
loc_update_done ⇐ FALSE
(config = MASTER + scr_status = OK) * lpi_wakemz_timer_done
lpi_link_fail_timer_done
B
WAKE_TRAINING
tx_mode ⇐ SEND_I
loc_rcvr_status = OK * rem_rcvr_status = OK
lpi_link_fail_timer_done
B

Figure 40–16b—PHY Control state diagram, part b (optional)

40.4.6.2 Link Monitor state diagram



NOTE 1—maxwait_timer is started in PHY Control state diagram (see Figure 40–16a).
NOTE 2—The variables link_control and link_status are designated as link_control_(1GigT) and link_status_(1GigT), respectively, by the Auto-Negotiation Arbitration state diagram (Figure 28–18).

Figure 40–17—Link Monitor state diagram

40.4.6.2.1 Auto Crossover state diagram

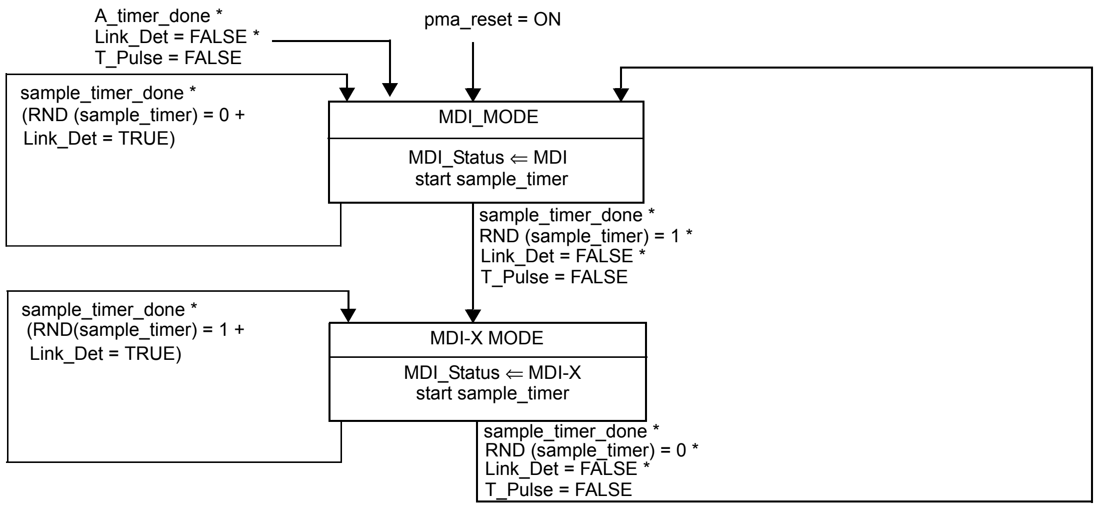


Figure 40–18—Auto Crossover state diagram

40.5 Management interface

1000BASE-T makes extensive use of the management functions provided by the MII Management Interface (see 22.2.4), and the communication and self-configuration functions provided by Auto-Negotiation (Clause 28).

40.5.1 Support for Auto-Negotiation

All 1000BASE-T PHYs shall provide support for Auto-Negotiation (Clause 28) and shall be capable of operating as MASTER or SLAVE.

Auto-Negotiation is performed as part of the initial set-up of the link, and allows the PHYs at each end to advertise their capabilities (speed, PHY type, half or full duplex) and to automatically select the operating mode for communication on the link. Auto-negotiation signaling is used for the following two primary purposes for 1000BASE-T:

a) To negotiate that the PHY is capable of supporting 1000BASE-T half duplex or full duplex transmission.
b) To determine the MASTER-SLAVE relationship between the PHYs at each end of the link.
c) To negotiate EEE capabilities as specified in 28C.12.

This relationship is necessary for establishing the timing control of each PHY. The 1000BASE-T MASTER PHY is clocked from a local source. The SLAVE PHY uses loop timing where the clock is recovered from the received data stream.