Item	Feature	Subclause	Status	Support	Value/Comment
MF12	Value of speed selection bit for single speed PHY	22.2.4.1.3	M		Set to match the correct PHY speed
MF13	Single speed PHY ignores writes to speed selection bit	22.2.4.1.3	M		
MF14	Auto-Negotiation enable	22.2.4.1.4	M		By setting 0.12 = 1
MF15	Duplex mode, speed selection have no effect when Auto-Negotiation is enabled	22.2.4.1.4	M		If 0.12=1, bits 0.13 and 0.8 have no effect on link configuration
MF16	PHY without Auto-Negotiation returns value of zero	22.2.4.1.4	M		Yes (if 1.3=0, then 0.12=0)
MF17	PHY without Auto-Negotiation ignores writes to enable bit	22.2.4.1.4	M		Yes (if 1.3=0, 0.12 always = 0 and cannot be changed)
MF18	Response to management transactions in power down	22.2.4.1.5	M		Remains active
MF19	Spurious signals in power down	22.2.4.1.5	M		None (not allowed)
MF20	TX_CLK and RX_CLK stabilize within 0.5 s	22.2.4.1.5	M		Yes (after both bits 0.11 and 0.10 are cleared to zero)
MF21	PHY Response to input signals while isolated	22.2.4.1.6	M		NONE
MF22	High impedance on PHY output signals while isolated	22.2.4.1.6	M		Yes (TX_CLK, RX_CLK, RX_DV, RX_ER, RXD<3:0>, COL, and CRS)
MF23	Response to management transactions while isolated	22.2.4.1.6	M		Remains active
MF24	Default value of isolate	22.2.4.1.6	M		0.10 =1
MF25	PHY without Auto-Negotiation returns value of zero	22.2.4.1.7	M		0.9 = 0 if 1.3 = 0 or 0.12 = 0
MF26	PHY without Auto-Negotiation ignores writes to restart bit	22.2.4.1.7	M		0.9 = 0, cannot be changed if 1.3 = 0 or 0.12 = 0
MF27	Restart Auto-Negotiation	22.2.4.1.7	M		When 0.9 = 1 if 0.12 = 1 and 1.3 = 1
MF28	Return 1 until Auto-Negotiation initiated	22.2.4.1.7	M		0.9 is self clearing to 0
MF29	Auto-Negotiation not effected by clearing bit	22.2.4.1.7	M		
MF30	Value of duplex mode bit for PHYs with one duplex mode	22.2.4.1.8	M		Set 0.8 to match the correct PHY duplex mode
MF31	PHY with one duplex mode ignores writes to duplex bit	22.2.4.1.8	M		Yes (0.8 remains unchanged)
MF32	Loopback not affected by duplex mode	22.2.4.1.8	M		Yes (0.8 has no effect on PHY when 0.14 = 1)
MF33	Assertion of COL in collision test mode	22.2.4.1.9	M		Within 512 BT after TX_EN is asserted

Item	Feature	Subclause	Status	Support	Value/Comment
MF34	De-assertion of COL in collision test mode	22.2.4.1.9	M		Within 4 BT after TX_EN is de-asserted
MF35	Reserved bits written as zero	22.2.4.1.10	M		
MF36	Reserved bits ignored when read	22.2.4.1.10	M		
MF37	PHY returns 0 in reserved bits	22.2.4.1.10	M		
MF38	Effect of write on status register	22.2.4.2	M		No effect
MF39	Reserved bits ignored when read	22.2.4.2.6	M		
MF40	PHY returns 0 in reserved bits	22.2.4.2.6	M		
MF41	PHY returns 0 if Auto-Negotiation disabled	22.2.4.2.8	M		Yes (1.5 = 0 when 0.12 = 0)
MF42	PHY returns 0 if it lacks ability to perform Auto-Negotiation	22.2.4.2.8	M		Yes (1.5 = 0 when 1.3 = 0)
MF43	Remote fault has latching function	22.2.4.2.9	M		Yes (once set will remain set until cleared)
MF44	Remote fault cleared on read	22.2.4.2.9	M		Yes
MF45	Remote fault cleared on reset	22.2.4.2.9	M		Yes (when 0.15 = 1)
MF46	PHY without remote fault returns value of zero	22.2.4.2.9	M		Yes (1.4 always 0)
MF47	Link status has latching function	22.2.4.2.11	M		Yes (once cleared by link failure will remain cleared until read by MII)
MF48	Jabber detect has latching function	22.2.4.2.12	M		Yes (once set will remain set until cleared)
MF49	Jabber detect cleared on read	22.2.4.2.12	M		
MF50	Jabber detect cleared on reset	22.2.4.2.12	M		
MF51	100BASE-T4 and 100BASE-X PHYs return 0 for jabber detect	22.2.4.2.12	M		Yes (1.1 always = 0 for 100BASE-T4 and 100BASE-TX)
MF52	MDIO not driven if register read is unimplemented	22.2.4.3	M		Yes (MDIO remain high impedance)
MF53	Write has no effect if register written is unimplemented	22.2.4.3	M		
MF54	Registers 2 and 3 constitute unique identifier for PHY type	22.2.4.3.1	M		
MF55	MSB of PHY identifier is 2.15	22.2.4.3.1	M		
MF56	LSB of PHY identifier is 3.0	22.2.4.3.1	M		
MF57	Composition of PHY identifier	22.2.4.3.1	M		22-bit OUI, 6-bit model, 4-bit version per figure 22-12
MF58	Format of management frames	22.2.4.4	M		Per table 22-9

Item	Feature	Subclause	Status	Support	Value/Comment
MF59	Idle condition on MDIO	22.2.4.4.1	M		High impedance state
MF60	MDIO preamble sent by STA	22.2.4.4.2	M		32 contiguous logic one bits
MF61	MDIO preamble observed by PHY	22.2.4.4.2	M		32 contiguous logic one bits
MF62	Assignment of PHYAD 0	22.2.4.4.5	M		Address of PHY attached via Mechanical Interface
MF63	Assignment of REGAD 0	22.2.4.4.6	M		MII control register address
MF64	Assignment of REGAD 1	22.2.4.4.6	M		MII status register address
MF65	High impedance during first bit time of turnaround in read transaction	22.2.4.4.7	M		
MF66	PHY drives zero during second bit time of turnaround in read transaction	22.2.4.4.7	M		
MF67	STA drives MDIO during turn-around in write transaction	22.2.4.4.7	M		
MF68	First data bit transmitted	22.2.4.4.8	M		Bit 15 of the register being ad-dressed

22.7.3.5 Signal timing characteristics

Item	Feature	Subclause	Status	Support	Value/Comment
ST1	Timing characteristics measured in accordance with annex 22C	22.3	M		
ST2	Transmit signal clock to output delay	22.3.1	M		Min = 0 ns; Max = 25 ns per figure 22-14
ST3	Receive signal setup time	22.3.2	M		Min = 10 ns per figure 22-15
ST4	Receive signal hold time	22.3.2	M		Min = 10 ns per figure 22-15
ST5	MDIO setup and hold time	22.3.4	M		Setup min = 10 ns; Hold min = 10 ns per figure 22-16
ST6	MDIO clock to output delay	22.3.4	M		Min = 0 ns; Max = 300 ns per figure 22-17

22.7.3.6 Electrical characteristics

Item	Feature	Subclause	Status	Support	Value/Comment
EC1	Signal paths are either point to point, or a sequence of point-to-point transmission paths	22.4.2	M		
EC2	MII uses unbalanced signal transmission paths	22.4.2	M		
EC3	Characteristic impedance of electrically long paths	22.4.2	M		68 Ω ± 15%
EC4	Output impedance of driver used to control reflections	22.4.2	M		On all electrically long point to point signal paths
EC5	V_{oh}	22.4.3.1	M		≥ 2.4 V ($I_{oh} = -4$ mA)
EC6	V_{ol}	22.4.3.1	M		≤ 0.4 V ($I_{ol} = 4$ mA)
EC7	Performance requirements to be guaranteed by ac specifications	22.4.3.2	M		Min switching potential change (including its reflection) ≥ 1.8 V
EC8	$V_{ih(min)}$	22.4.4.1	M		2 V
EC9	$V_{il(max)}$	22.4.4.1	M		0.8 V
EC10	Input current measurement point	22.4.4.2	M		At MII connector
EC11	Input current reference potentials	22.4.4.2	M		Reference to MII connector +5 V and COMMON pins
EC12	Input current reference potential range	22.4.4.2	M		0 V to 5.25 V
EC13	Input current limits	22.4.4.2	M		Per table 22-10

Item	Feature	Subclause	Status	Support	Value/Comment
EC14	Input capacitance for signals other than MDIO	22.4.4.3	M		≤ 8 pF
EC15	Input capacitance for MDIO	22.4.4.3	M		≤ 10 pF
EC16	Twisted-pair composition	22.4.5	M		Conductor for each signal with dedicated return path
EC17	Single-ended characteristic impedance	22.4.5.2	M		68 Ω ± 10%
EC18	Characteristic impedance measurement method	22.4.5.2	M		With return conductor connected to cable shield
EC19	Twisted-pair propagation delay	22.4.5.3	M		≤ 2.5 ns
EC20	Twisted-pair propagation delay measurement method	22.4.5.3	M		With return conductor connected to cable shield
EC21	Twisted-pair propagation delay measurement frequency	22.4.5.3	M		25 MHz
EC22	Twisted-pair propagation delay variation	22.4.5.4	M		≤ 0.1 ns
EC23	Twisted-pair propagation delay variation measurement method	22.4.5.4	M		With return conductor connected to cable shield
EC24	Cable shield termination	22.4.5.5	M		To the connector shell
EC25	Cable conductor DC resistance	22.4.5.6	M		≤ 150 mΩ
EC26	Effect of hot insertion/removal	22.4.6	M		Causes no damage
EC27	State of PHY output buffers during hot insertion	22.4.6	M		High impedance
EC28	State of PHY output buffers after hot insertion	22.4.6	M		High impedance until enabled via Isolate bit

22.7.3.7 Power supply

Item	Feature	Subclause	Status	Support	Value/Comment
PS1	Regulated power supply provided	22.5	M		To PHY by Reconciliation sublayer
PS2	Power supply lines	22.5	M		+5 V and COMMON (return of +5 V)
PS3	Regulated supply voltage limits	22.5.1	M		5 Vdc ± 5%
PS4	Over/under voltage limits	22.5.1	M		Over limit = 5.25 Vdc Under limit = 0 V
PS5	Load current limit	22.5.2	M		750 mA
PS6	Surge current limit	22.5.2	M		≤ 5 A peak for 10 ms
PS7	PHY can power up from current limited source	22.5.2	M		From 750 mA current limited source
PS8	Short-circuit protection	22.5.3	M		When +5 V and COMMON are shorted

22.7.3.8 Mechanical characteristics

Item	Feature	Subclause	Status	Support	Value/Comment
*MC1	Use of Mechanical Interface	22.6	O		Optional
MC2	Connector reference standard	22.6.1	MC1:M		IEC 1076-3-101: 1995
MC3	Use of female connector	22.6.1	MC1:M		At MAC/RS side
MC4	Use of male connector	22.6.1	MC1:M		At PHY mating cable side
MC5	Connector shell plating	22.6.2	MC1:M		Use conductive material
MC6	Shield transfer impedance	22.6.2	MC1:M		After 500 cycles of mating/ unmating, per table 22-11
MC7	Additions to provide for female shell to male shell conductivity	22.6.2	MC1:M		On shell of conductor with male contacts
MC8	Clearance dimensions	22.6.4	MC1:M		15 mm × 50 mm, per figure 22-19

23. Physical Coding Sublayer (PCS), Physical Medium Attachment (PMA) sublayer and baseband medium, type 100BASE-T4

23.1 Overview

The 100BASE-T4 PCS, PMA, and baseband medium specifications are aimed at users who want 100 Mb/s performance, but would like to retain the benefits of using voice-grade twisted-pair cable. 100BASE-T4 signaling requires four pairs of Category 3 or better cable, installed according to ISO/IEC 11801: 1995, as specified in 23.6. This type of cable, and the connectors used with it, are simple to install and reconfigure. 100BASE-T4 does not transmit a continuous signal between packets, which makes it useful in battery powered applications. The 100BASE-T4 PHY is one of the 100BASE-T family of high-speed CSMA/CD network specifications.

23.1.1 Scope

This clause defines the type 100BASE-T4 Physical Coding Sublayer (PCS), type 100BASE-T4 Physical Medium Attachment (PMA) sublayer, and type 100BASE-T4 Medium Dependent Interface (MDI). Together, the PCS and PMA layers comprise a 100BASE-T4 Physical Layer (PHY). Provided in this document are full functional, electrical, and mechanical specifications for the type 100BASE-T4 PCS, PMA, and MDI. This clause also specifies the baseband medium used with 100BASE-T4.

23.1.2 Objectives

The following are the objectives of 100BASE-T4:

a) To support the CSMA/CD MAC.
b) To support the 100BASE-T MII, Repeater, and optional Auto-Negotiation.
c) To provide 100 Mb/s data rate at the MII.
d) To provide for operating over unshielded twisted pairs of Category 3, 4, or 5 cable, installed as horizontal runs in accordance with ISO/IEC 11801: 1995, as specified in 23.6, at distances up to 100 m (328 ft).
e) To allow for a nominal network extent of 200 m, including:
    1) Unshielded twisted-pair links of 100 m.
    2) Two-repeater networks of approximately a 200 m span.
f) To provide a communication channel with a mean ternary symbol error rate, at the PMA service interface, of less than one part in 10^8.

23.1.3 Relation of 100BASE-T4 to other standards

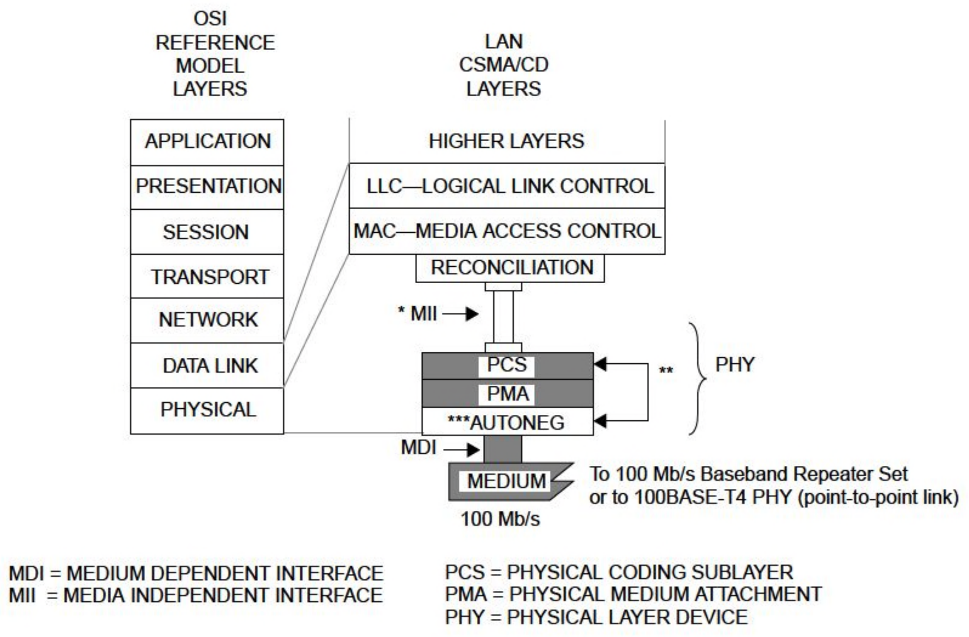
Relations between the 100BASE-T4 PHY and the ISO Open Systems Interconnection (OSI) reference model and the IEEE 802.3 CSMA/CD LAN model are shown in figure 23-1. The PHY Layers shown in figure 23-1 connect one clause 4 Media Access Control (MAC) layer to a clause 27 repeater. This clause also discusses other variations of the basic configuration shown in figure 23-1. This whole clause builds on clauses 1 through 4 of this standard.

23.1.4 Summary

The following paragraphs summarize the PCS and PMA clauses of this document.

23.1.4.1 Summary of Physical Coding Sublayer (PCS) specification

The 100BASE-T4 PCS couples a Media Independent Interface (MII), as described in clause 22, to a Physical Medium Attachment sublayer (PMA).


OSI REFERENCE MODEL LAYERS
APPLICATION
PRESENTATION
SESSION
TRANSPORT
NETWORK
DATA LINK
PHYSICAL
LAN CSMA/CD LAYERS
HIGHER LAYERS
LLC—LOGICAL LINK CONTROL
MAC—MEDIA ACCESS CONTROL
RECONCILIATION
* MII
PCS
PMA
***AUTONEG
**
PHY
MDI
MEDIUM
To 100 Mb/s Baseband Repeater Set or to 100BASE-T4 PHY (point-to-point link)
100 Mb/s
MDI = MEDIUM DEPENDENT INTERFACE
MII = MEDIA INDEPENDENT INTERFACE
PCS = PHYSICAL CODING SUBLAYER
PMA = PHYSICAL MEDIUM ATTACHMENT
PHY = PHYSICAL LAYER DEVICE

* MII is optional.
** AUTONEG communicates with the PMA sublayer through the PMA service interface messages PMA_LINK.request and PMA_LINK.indicate.
*** AUTONEG is optional.

Figure 23-1—Type 100BASE-T4 PHY relationship to the ISO Open Systems Interconnection (OSI) reference model and the IEEE 802.3 CSMA/CD LAN model

The PCS Transmit function accepts data nibbles from the MII. The PCS Transmit function encodes these nibbles using an 8B6T coding scheme (to be described) and passes the resulting ternary symbols to the PMA. In the reverse direction, the PMA conveys received ternary symbols to the PCS Receive function. The PCS Receive function decodes them into octets, and then passes the octets one nibble at a time up to the MII. The PCS also contains a PCS Carrier Sense function, a PCS Error Sense function, a PCS Collision Presence function, and a management interface.

Figure 23-2 shows the division of responsibilities between the PCS, PMA, and MDI layers.

Physical level communication between PHY entities takes place over four twisted pairs. This specification permits the use of Category 3, 4, or 5 unshielded twisted pairs, installed according to ISO/IEC 11801: 1995, as specified in 23.6. Figure 23-3 shows how the PHY manages the four twisted pairs at its disposal.

The 100BASE-T4 transmission algorithm always leaves one pair open for detecting carrier from the far end (see figure 23-3). Leaving one pair open for carrier detection in each direction greatly simplifies media access control. All collision detection functions are accomplished using only the unidirectional pairs TX_D1 and RX_D2, in a manner similar to 10BASE-T. This collision detection strategy leaves three pairs in each direction free for data transmission, which uses an 8B6T block code, schematically represented in figure 23-4.

8B6T coding, as used with 100BASE-T4 signaling, maps data octets into ternary symbols. Each octet is mapped to a pattern of 6 ternary symbols, called a 6T code group. The 6T code groups are fanned out to three independent serial channels. The effective data rate carried on each pair is one third of 100 Mb/s,

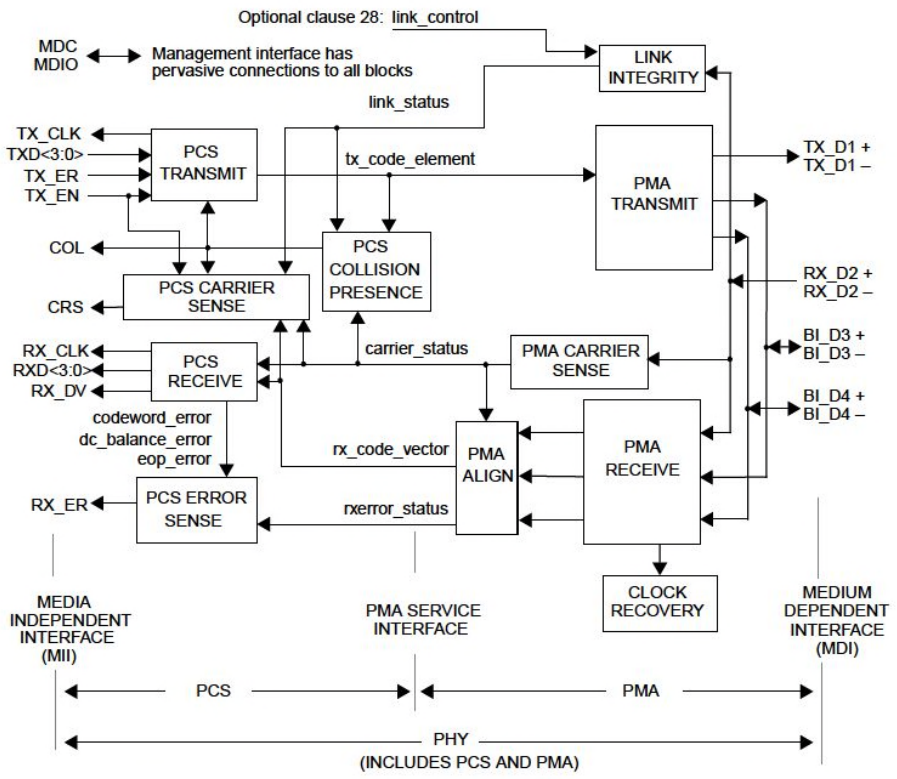


Figure 23-2—Division of responsibilities between 100BASE-T4 PCS and PMA

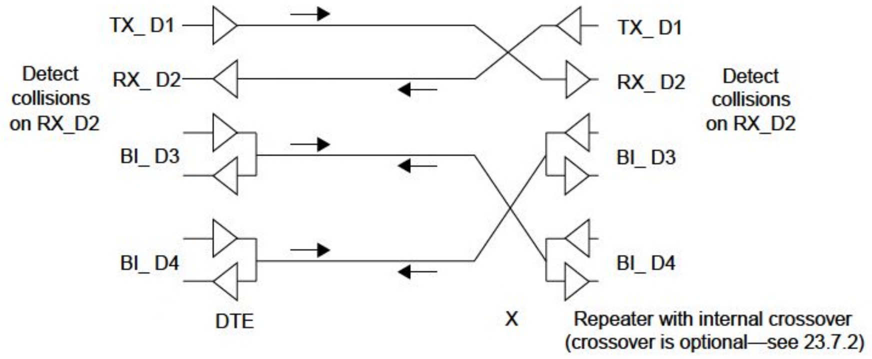

TX_ D1
RX_ D2
Detect collisions on RX_D2
BI_ D3
BI_ D4
DTE
X
TX_ D1
RX_ D2
Detect collisions on RX_D2
BI_ D3
BI_ D4
Repeater with internal crossover
(crossover is optional—see 23.7.2)

Figure 23-3—Use of wire pairs

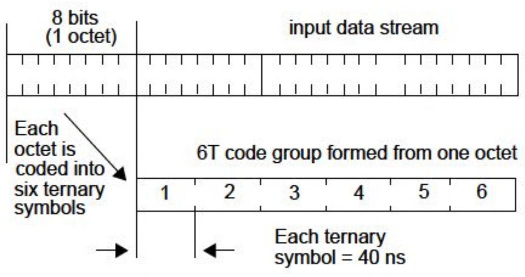

8 bits
(1 octet)
input data stream
Each octet is coded into six ternary symbols
6T code group formed from one octet
1
2
3
4
5
6
Each ternary symbol = 40 ns

Figure 23-4—8B6T coding

which is 33.333... Mb/s. The ternary symbol transmission rate on each pair is 6/8 times 33.33 Mb/s, or precisely 25.000 MHz.

Refer to annex 23A for a complete listing of 8B6T code words.

The PCS functions and state diagrams are specified in 23.2. The PCS electrical interface to the MII conforms to the interface requirements of clause 21. The PCS interface to the PMA is an abstract message-passing interface specified in 23.3.

23.1.4.2 Summary of physical medium attachment (PMA) specification

The PMA couples messages from the PMA service interface onto the twisted-pair physical medium. The PMA provides communications, at 100 Mb/s, over four pairs of twisted-pair wiring up to 100 m in length.

The PMA Transmit function, shown in figure 23-2, comprises three independent ternary data transmitters. Upon receipt of a PMA_UNITDATA request message, the PMA synthesizes one ternary symbol on each of the three output channels (TX_D1, BI_D3, and BI_D4). Each output driver has a *ternary* output, meaning that the output waveform can assume any of three values, corresponding to the transmission of ternary symbols CS0, CS1 or CS-1 (see 23.4.3.1) on each of the twisted pairs.

The PMA Receive function comprises three independent ternary data receivers. The receivers are responsible for acquiring clock, decoding the Start of Stream Delimiter (SSD) on each channel, and providing data to the PCS in the synchronous fashion defined by the PMA_UNITDATA.indicate message. The PMA also contains functions for PMA Carrier Sense and Link Integrity.

PMA functions and state diagrams appear in 23.4. PMA electrical specifications appear in 23.5.

23.1.5 Application of 100BASE-T4

23.1.5.1 Compatibility considerations

All implementations of the twisted-pair link shall be compatible at the MDI. The PCS, PMA, and the medium are defined to provide compatibility among devices designed by different manufacturers. Designers are free to implement circuitry within the PCS and PMA (in an application-dependent manner) provided the MDI (and MII, when implemented) specifications are met.

23.1.5.2 Incorporating the 100BASE-T4 PHY into a DTE

The PCS is required when used with a DTE. The PCS provides functions necessary to the overall system operation (such as 8B6T coding) and cannot be omitted. Refer to figure 23-1.

When the PHY is incorporated within the physical bounds of a DTE, conformance to the MII interface is optional, provided that the observable behavior of the resulting system is identical to a system with a full MII implementation. For example, an integrated PHY may incorporate an interface between PCS and MAC that is logically equivalent to the MII, but does not have the full output current drive capability called for in the MII specification.

23.1.5.3 Use of 100BASE-T4 PHY for point-to-point communication

The 100BASE-T4 PHY, in conjunction with the MAC specified in clauses 1-4 (including parameterized values in 4.4.2.3 to support 100 Mb/s operation), may be used at both ends of a link for point-to-point applications between two DTEs. Such a configuration does not require a repeater. In this case each PHY may connect through an MII to its respective DTE. Optionally, either PHY (or both PHYs) may be incorporated into the DTEs without an exposed MII.

23.1.5.4 Support for Auto-Negotiation

The PMA service interface contains primitives used by the Auto-Negotiation algorithm (clause 28) to automatically select operating modes when connected to a like device.

23.2 PCS functional specifications

The 100BASE-T4 PCS couples a Media Independent Interface (MII), as described in clause 22, to a 100BASE-T4 Physical Medium Attachment sublayer (PMA).

At its interface with the MII, the PCS communicates via the electrical signals defined in clause 22.

The interface between PCS and the next lower level (PMA) is an abstract message-passing interface described in 23.3. The physical realization of this interface is left to the implementor, provided the requirements of this standard, where applicable, are met.

23.2.1 PCS functions

The PCS comprises one PCS Reset function and five simultaneous and asynchronous operating functions. The PCS operating functions are PCS Transmit, PCS Receive, PCS Error Sense, PCS Carrier Sense, and PCS Collision Presence. All operating functions start immediately after the successful completion of the PCS Reset function.

The PCS reference diagram, figure 23-5, shows how the five 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 23-5. The management functions are specified in clause 30. See also figure 23-6, which defines the structure of frames passed from PCS to PMA. See also figure 23-7, which presents a reference model helpful for understanding the definitions of PCS Transmit function state variables ohr1-4 and tsr.

23.2.1.1 PCS Reset function

The PCS Reset function shall be executed any time either of two conditions occur. These two conditions are “power on” and the receipt of a reset request from the management entity. The PCS Reset function initializes all PCS functions. The PCS Reset function sets pcs_reset ≤ ON for the duration of its reset function. All state diagrams take the open-ended pcs_reset branch upon execution of the PCS Reset function. The reference diagrams do not explicitly show the PCS Reset function.

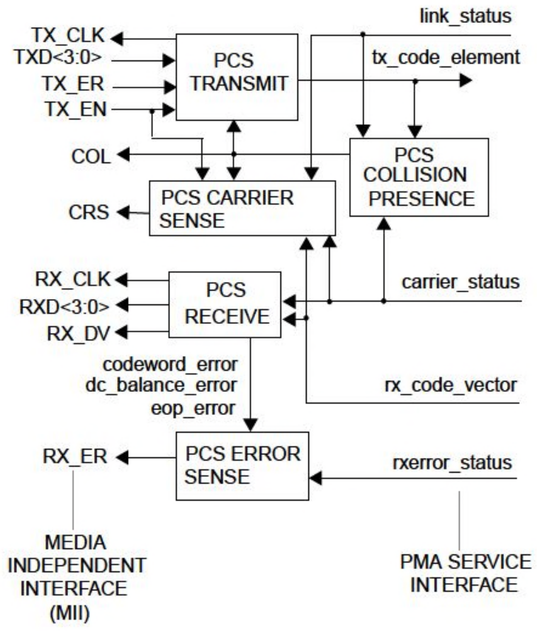


Figure 23-5—PCS reference diagram

23.2.1.2 PCS Transmit function

The PCS Transmit function shall conform to the PCS Transmit state diagram in figure 23-8.

The PCS Transmit function receives nibbles from the TXD signals of the MII, assembles pairs of nibbles to form octets, converts the octets into 6T code groups according to the 8B6T code table, and passes the resulting ternary data to the PMA using the PMA_UNITDATA request message. The state diagram of figure 23-8 depicts the PCS Transmit function operation. Definitions of state variables tsr, ohr, sosa, sosb, eop1-5, and tx_extend used in that diagram, as well as in the following text, appear in 23.2.4.1. The physical structure represented in figure 23-7 is not required; it merely serves to explain the meaning of the state diagram variables ohr and tsr in figure 23-8. Implementors are free to construct any logical devices having functionality identical to that described by this functional description and the PCS Transmit state diagram, figure 23-8.

PCS Transmit makes use of the tsr and ohr shift registers to manage nibble assembly and ternary symbol transmission. Nibbles from the MII go into tsr, which PCS Transmit reads as octets. PCS Transmit then encodes those octets and writes 6T code groups to the ohr registers. The PMA_UNITDATA.request message passes ternary symbols from the ohr registers to the PMA. In each state diagram block, the ohr loading operations are conducted first, then tx_code_vector is loaded and the state diagram waits 40 ns.

The first 5 octets assembled by the PCS Transmit function are encoded into the sosa code word and the next 3 octets assembled are encoded into the sosb code word. This guarantees that every packet begins with a valid preamble pattern. This is accomplished by the definition of tsr. In addition, the PCS Transmit state diagram also specifies that at the start of a packet all three output holding registers ohr1, ohr3 and ohr4 will be loaded with the same value (sosa). This produces the ternary symbols labeled P3 and P4 in figure 23-6.

At the conclusion of the MAC frame, the PCS Transmit function appends eop1-5. This is accomplished by defining a variable tx_extend to stretch the TX_EN signal, and defining tsr during this time to be a sequence of constants that decodes to the proper eop code groups.

The encoding operation shall use the 8B6T code table listed in annex 23A, and the dc balance encoding rules listed below. Encoding is performed separately for each transmit pair.

23.2.1.2.1 DC balance encoding rules

The encoding operation maintains dc balance on each transmit pair by keeping track of the cumulative weight of all 6T code groups (see *weight of 6T code group*, annex 21A) transmitted on that pair. For each pair, it initiates the cumulative weight to 0 when the PCS Transmit function is in the AWAITING DATA TO TRANSMIT state. All 6T code groups in the code table have weight 0 or 1. The dc balance algorithm conditionally negates transmitted 6T code groups, so that the code weights transmitted on the line include 0, +1, and –1. This dc balance algorithm ensures that the cumulative weight on each pair at the conclusion of each 6T code group is always either 0 or 1, so only one bit per pair is needed to store the cumulative weight. As used below, the phrase “invert the cumulative weight bit” means “if the cumulative weight bit is zero then set it to one, otherwise set it to zero.”

After encoding any octet, except the constants sosa, sosb, eop1-5 or bad_code, update the cumulative weight bit for the affected pair according to rules a) through c):

a) If the 6T code group weight is 0, do not change the cumulative weight.
b) If the 6T code group weight is 1, and the cumulative weight bit is 0, set the cumulative weight bit to 1.
c) If the 6T code group weight is 1, and the cumulative weight bit is also 1, set the cumulative weight bit to 0, and then algebraically negate all the ternary symbol values in the 6T code group.

After encoding any of the constants sosa, sosb, or bad_code, update the cumulative weight bit for the affected pair according to rule d):

d) Do not change the cumulative weight. Never negate sosa, sosb or bad_code.

After encoding any of the constants eop1-5, update the cumulative weight bit for the affected pair according to rules e) and f):

e) If the cumulative weight is 0, do not change the cumulative weight; algebraically negate all the ternary symbol values in eop1-5.
f) If the cumulative weight is 1, do not change the cumulative weight.

NOTE—The inversion rules for eop1-5 are opposite rule b). That makes eop1-5 look very unlike normal data, increasing the number of errors required to synthesize a false end-of-packet marker.

23.2.1.3 PCS Receive function

The PCS Receive function shall conform to the PCS Receive state diagram in figure 23-9.

The PCS Receive function accepts ternary symbols from the PMA, communicated via the PMA_UNITDATA.indicate message, converts them using 8B6T coding into a nibble-wide format and passes them up to the MII. This function also generates RX_DV. The state diagram of figure 23-9 depicts the PCS Receive function. Definitions of state variables ih2, ih3, and ih4 used in that diagram, as well as in the following text, appear in 23.2.4.1.

The last 6 values of the rx_code_vector are available to the decoder. PCS Receive makes use of these stored rx_code_vector values as well as the ih2-4 registers to manage the assembly of ternary symbols into 6T code groups, and the conversion of decoded data octets into nibbles. The last 6 ternary symbols for pair BI_D3 (as extracted from the last 6 values of rx_code_vector) are referred to in the state diagram as BI_D3[0:5]. Other pairs are referenced accordingly.

The PCS Receive state diagram starts the first time the PCS receives a PMA_UNITDATA.indicate message with rx_code_vector=DATA (as opposed to IDLE or PREAMBLE). The contents of this first PMA_UNITDATA.indicate (DATA) message are specified in 23.4.1.6.

After the sixth PMA_UNITDATA.indicate (DATA) message (state DECODE CHANNEL 3), there is enough information to decode the first data octet. The decoded data is transmitted across the MII in two parts, a least significant nibble followed by a most significant nibble (see clause 22).

During state COLLECT 4TH TERNARY SYMBOL the PCS Receive function raises RX_DV and begins shifting out the nibbles of the 802.3 MAC SFD, least significant nibble first (SFD:LO). The most significant nibble of the 802.3 MAC SFD, called SFD:HI, is sent across the MII during the next state, COLLECT 5TH TERNARY SYMBOL.

Once eop is signaled by the decode operation, the state diagram de-asserts RX_DV, preventing the end-of-packet bits from reaching the MII. At any time that RX_DV is de-asserted, RXD<3:0> shall be all zeroes.

The decode operation shall use the 8B6T code table listed in annex 23A, and the error-detecting rules listed in 23.2.1.3.1. Decoding and maintenance of the cumulative weight bit is performed separately for each receive pair.

23.2.1.3.1 Error-detecting rules

The decoding operation checks the dc balance on each receive pair by keeping track of the cumulative weight of all 6T code group received on that pair. For each pair, initialize the cumulative weight to 0 when the PCS Receive function is in the AWAITING INPUT state. As in the encoding operation, only one bit per pair is needed to store the cumulative weight.

Before decoding each octet, check the weight of the incoming code group and then apply rules a) through h) in sequence:

a) If the received code group is eop1 (or its negation), set eop=ON. Then check the other pairs for conformance to the end-of-packet rules as follows: Check the last four ternary symbols of the next pair, and the last two ternary symbols from the third pair for exact conformance with the end-of-packet pattern specified by PCS Transmit, including the cumulative weight negation rules. If the received data does not conform, set the internal variable eop_error=ON. Skip the other rules.
b) If the received code group weight is greater than 1 or less than –1, set the internal variable dc_balance_error=ON. Decode to all zeros. Do not change the cumulative weight.
c) If the received code group weight is zero, use the code table to decode. Do not change the cumulative weight.
d) If the received code group weight is +1, and the cumulative weight bit is 0, use the code table to decode. Invert the cumulative weight bit.
e) If the received code group weight is –1, and the cumulative weight bit is 1, algebraically negate each ternary symbol in the code group and then use the code table to decode. Invert the cumulative weight bit.
f) If the received code group weight is +1 and the cumulative weight bit is 1, set the internal variable dc_balance_error=ON. Decode to all zeros. Do not change the cumulative weight.
g) If the received code group weight is –1 and the cumulative weight bit is 0, set the internal variable dc_balance_error=ON. Decode to all zeros. Do not change the cumulative weight.
h) If the (possibly negated) code group is not found in the code table, set codeword_error =ON. Decode to all zeros. Do not change the cumulative weight.

The variables dc_balance_error, eop_error and codeword_error shall remain OFF at all times other than those specified in the above error-detecting rules.

The codeword_error=ON indication for a (possibly negated) code group not found in the code table shall set RX_ER during the transfer of both affected data nibbles across the MII.

The dc_balance_error=ON indication for a code group shall set RX_ER during the transfer of both affected data nibbles across the MII.

The eop_error=ON indication shall set RX_ER during the transfer of the last decoded data nibble of the previous octet across the MII. That is at least one RX_CLK period earlier than the requirement for codeword_error and dc_balance_error.

These timing requirements imply consideration of implementation delays not specified in the PCS Receive state diagram.

RX_DV is asserted coincident with the transmission across the MII of valid packet data, including the clause 4 MAC SFD, but not including the 100BASE-T4 end-of-packet delimiters eop1-5. When a packet is truncated due to early de-assertion of carrier_status, an RX_ER indication shall be generated and RX_DV shall be de-asserted, halting receive processing. The PCS Receive Function may use any of the existing signals codeword_error, dc_balance_error, or eop_error to accomplish this function.

23.2.1.4 PCS Error Sense function

The PCS Error Sense function performs the task of sending RX_ER to the MII whenever rxerror_status=ERROR is received from the PMA sublayer or when any of the PCS decoding error conditions occur. The PCS Error Sense function shall conform to the PCS Error Sense state diagram in figure 23-10.

Upon detection of any error, the error sense process shall report RX_ER to the MII before the last nibble of the clause 4 MAC frame has been passed across the MII. Errors attributable to a particular octet are reported to the MII coincident with the octet in which they occurred.

The timing of rxerror_status shall cause RX_ER to appear on the MII no later than the last nibble of the first data octet in the frame.

23.2.1.5 PCS Carrier Sense function

The PCS Carrier Sense function shall perform the function of controlling the MII signal CRS according to the rules presented in this clause.

While link_status = OK, CRS is asserted whenever rx_crs=ON or TX_EN=1, with timing as specified in 23.11.2, and table 23-6.

23.2.1.6 PCS Collision Presence function

A PCS collision is defined as the simultaneous occurrence of tx_code_vector≠IDLE and the assertion of carrier_status=ON while link_status=OK. While a PCS collision is detected, the MII signal COL shall be asserted, with timing as specified in 23.11.2 and table 23-6.

At other times COL shall remain de-asserted.

23.2.2 PCS interfaces

23.2.2.1 PCS–MII interface signals

The following signals are formally defined in 22.2.2. Jabber detection as specified in 22.2.4.2.12 is not required by this standard.

Table 23-1—MII interface signals

Signal name	Meaning
TX_CLK	Transmit Clock
TXD<3:0>	Transmit Data
TX_ER	Forces transmission of illegal code
TX_EN	Frames Transmit Data
COL	Collision Indication
CRS	Non-Idle Medium Indication
RX_CLK	Receive Clock
RXD<3:0>	Receive Data
RX_DV	Frames Receive SFD and DATA
RX_ER	Receive Error Indication
MDC	Management Data Clock
MDIO	Management Data

23.2.2.2 PCS–Management entity signals

The management interface has pervasive connections to all functions. Operation of the management control lines MDC and MDIO, and requirements for managed objects inside the PCS and PMA, are specified in clauses 22 and 30, respectively.

The loopback mode of operation shall be implemented in accordance with 22.2.4.1.2. The loopback mode of operation loops back transmit data to receive data, thus providing a way to check for the presence of a PHY.

No spurious signals shall be emitted onto the MDI when the PHY is held in power-down mode as defined in 22.2.4.1.5 (even if TX_EN is ON) or when released from power-down mode, or when external power is first applied to the PHY.

23.2.3 Frame structure

Frames passed from the PCS sublayer to the PMA sublayer shall have the structure shown in figure 23-6. This figure shows how ternary symbols on the various pairs are synchronized as they are passed by the PMA_UNITDATA.indicate and PMA_UNITDATA request messages. Time proceeds from left to right in the figure.

In the frame structure example, the last 6T code group, DATA N, happens to appear on transmit pair BI_D3. It could have appeared on any of the three transmit pairs, with the five words eop1 through eop5 appended afterward as the next five octets in sequence. The end of packet as recognized by the PCS is defined as the end of the last ternary symbol of eop1. At this point a receiver has gathered enough information to locate the last word in the packet and check the dc balance on each pair.

If the PMA service interface is exposed, data carried between PCS and PMA by the PMA_UNITDATA.indicate and PMA_UNITDATA request messages shall have a clock in each direction. Details of the clock

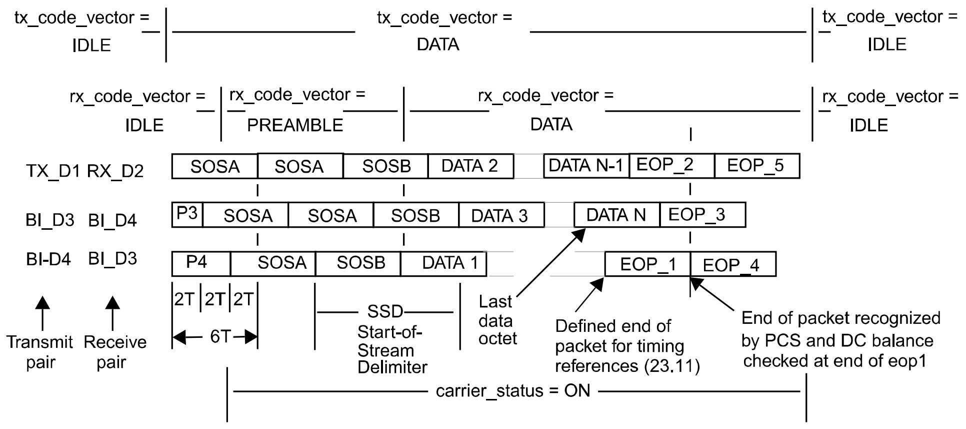


Figure 23-6—PCS sublayer to PMA sublayer frame structure

implementation are left to the implementor. The choice of binary encoding for each ternary symbol is left to the implementor.

The following frame elements appear in figure 23-6 (ternary symbols are transmitted leftmost first):

SOSA	The succession of six ternary symbols: `[ 1 -1 1 -1 1 -1]`, which is the result of encoding the constant sosa.
SOSB	The succession of six ternary symbols: `[ 1 -1 1 -1 -1 1]`, which is the result of encoding the constant sosb.
P3	The succession of two ternary symbols: `[ 1 -1]`.
P4	The succession of four ternary symbols: `[ 1 -1 1 -1]`.
DATA	A 6T code group that is the result of encoding a data octet in a packet that is not part of the clause 4 MAC preamble or SFD.
EOP1-5	A 6T code group that is the result of encoding one of the end-of-packet patterns eop1-5.

23.2.4 PCS state diagrams

The notation used in the state diagrams follows the conventions of 21.5. Transitions shown without source states are evaluated continuously and take immediate precedence over all other conditions.

23.2.4.1 PCS state diagram constants

Register tsr may take on any of the nine constant values listed below (sosa through eop5, bad_code, and zero_code). These values are used to describe the functional operation of the coding process.

NOTE—Implementors are under no obligation to implement these constants in any particular way. For example, some implementors may choose to implement these codes as special flag bits attached to MII TXD nibble registers. Other implementors may choose to implement insertion of these codes on the downstream side of the coder function, using precoded 6T sequences.

All 6T code words are sent leftmost ternary symbol first.

sosa	A constant that encodes to:	`[ 1 -1 1 -1 1 -1].`
sosb	A constant that encodes to:	`[ 1 -1 1 -1 -1 1].`
eop1	A constant that encodes to:	`[ 1 1 1 1 1 1].`
eop2	A constant that encodes to:	`[ 1 1 1 1 -1 -1].`
eop3	A constant that encodes to:	`[ 1 1 -1 -1 0 0].`
eop4	A constant that encodes to:	`[ -1 -1 -1 -1 -1 -1].`
eop5	A constant that encodes to:	`[ -1 -1 0 0 0 0].`
bad_code	A constant that encodes to:	`[ -1 -1 -1 1 1 1].`
zero_code	A constant that encodes to:	`[ 0 0 0 0 0 0].`

23.2.4.2 PCS state diagram variables

codeword_error

Indicates reception of invalid 6T code group.

Values: ON and OFF

Set by: PCS Receive; error-detecting rules

dc_balance_error

Indicates reception of dc coding violation.

Values: ON and OFF

Set by: PCS Receive; error-detecting rules

eop

Indicates reception of eop1.

A state variable set by the decoding operation. Reset to OFF when in PCS Receive state AWAITING INPUT. When the decoder detects eop1 on any pair, it sets this flag ON. The timing of eop shall be adjusted such that the last nibble of the last decoded data octet in a packet is the last nibble sent across the MII by the PMA Receive state diagram with RX_DV set ON.

Values: ON and OFF

Set by: PCS Receive; error-detecting rules

eop_error

Indicates reception of data with improper end-of-packet coding.

Values: ON and OFF

Set by: PCS Receive; error-detecting rules ih2, ih4, and ih3 (input holding registers)

A set of holding registers used for the purpose of holding decoded data octets in preparation for sending across the MII one nibble at a time. One register is provided for each of the three receive pairs RX_D2, BI_D4, and BI_D3, respectively.

Value:	octet
Set by:	PCS Receive

Each time the PCS Receive function decodes a 6T code group, it loads the result (an octet) into one of the ih2-4 registers. These three registers are loaded in round-robin fashion, one register being loaded every two ternary symbol times.

The PCS Receive state diagram reads nibbles as needed from the ih2-4 registers and stuffs them into RXD.

ohr1, ohr3, and ohr4 (output holding registers)

(See figure 23-7.) A set of shift registers used for the purpose of transferring coded 6T ternary symbol groups one ternary symbol at a time into the PMA. One register is provided for each of the three transmit pairs TX_D1, BI_D3, and BI_D4, respectively.

Value:	6T code group. Each of the six cells holds one ternary symbol (i.e., –1, 0, or 1).
Set by:	PCS Transmit

Each time the PCS Transmit function encodes a data octet, it loads the result (a 6T code group) into one of the ohr registers. Three registers are loaded in round-robin fashion, one register being loaded every two ternary symbol times. The PCS shall transmit octets on the three transmit pairs in round-robin fashion, in the order TX_D1, BI_D3, and BI_D4, starting with TX_D1.

The PMA_UNITDATA request (DATA) message picks the least significant (rightmost) ternary symbol from each ohr register and sends it to the PMA, as shown below. (Note that 6T code words in annex 23A are listed with lsb on the left, not the right.)

tx_code_vector[TX_D1] = the LSB of ohr1, also called ohr1[0]

tx_code_vector[BI_D3] = the LSB of ohr3, also called ohr3[0]

tx_code_vector[BI_D4] = the LSB of ohr4, also called ohr4[0]

After each PMA_UNITDATA request message, all three ohr registers shift right by one ternary symbol, shifting in zero from the left. The PCS Transmit function loads a new 6T code group into each ohr immediately after the last ternary symbol of the previous group is shifted out.

At the beginning of a preamble, the PCS Transmit function loads the same value (sosa) into all three output holding registers, which causes alternating transitions to immediately appear on all three output pairs. The result on pairs BI_D3 and BI_D4 is depicted by code words P3 and P4 in figure 23-6.

pcs_reset

Causes reset of all PCS functions when ON.

Values:	ON and OFF
Set by:	PCS Reset

rx_crs

A latched asynchronous variable. Timing for the MII signal CRS is derived from rx_crs.

Values:	ON and OFF
Set ON when:	carrier_status changes to ON
Set OFF when	either of two events occurs: carrier_status changes to OFF, or detection of eop1, properly framed, on any of the lines RX_D2, BI_D4, or BI_D3

Additionally, if, 20 ternary symbol times after rx_crs falls, carrier_status remains set to ON then set rx_crs=ON.

NOTE—A special circuit for the detection of eop1 and subsequent de-assertion of rx_crs, faster than the full 8B6T decoding circuits, is generally required to meet the timing requirements for CRS listed in clause 23.11.

tsr (transmit shift register)

(See figure 23-7.) A shift register defined for the purpose of assembling nibbles from the MII TXD into octets.

Values:	The variable tsr always contains both the current nibble of TXD and the previous nibble of TXD. Valid values for tsr therefore include all octets. Register tsr may also take on any of the nine constant values listed in 23.2.4.1.
Nibble order:	When encoding the tsr octet, the previous TXD nibble is considered the least significant nibble.
Set by:	PCS Transmit

During the first 16 TX_CLK cycles after TX_EN is asserted, tsr shall assume the following values in sequence regardless of TXD: sosa, sosa, sosa, sosa, sosa, sosa, sosa, sosa, sosa, sosa, sosb, sosb, sosb, sosb, sosb, sosb. This action substitutes the 100BASE-T4 preamble for the clause 4 MAC preamble. The PCS Transmit state diagram samples the tsr only every other clock, which reduces the number of sosa and sosb constants actually coded to 5 and 3, respectively.

During the first 10 TX_CLK cycles after TX_EN is de-asserted, tsr shall assume the following values in sequence, regardless of TXD: eop1, eop1, eop2, eop2, eop3, eop3, eop4, eop4, eop5, eop5. This action appends the 100BASE-T4 end-of-packet delimiter to each pair. The PCS Transmit state diagram samples the tsr only every other clock, which reduces the number of eop1-5 constants actually coded to 1 each.

Except for the first 16 TX_CLK cycles after TX_EN is asserted, any time TX_ER and TX_EN are asserted, tsr shall assume the value bad_code with such timing as to cause both nibbles of the affected octet to be encoded as bad_code. If TX_ER is asserted at any time during the first 16 TX_CLK cycles after TX_EN is asserted, tsr shall during the 17th and 18th clock cycles assume the value bad_code.

If TX_EN is de-asserted on an odd nibble boundary, the PCS shall extend TX_EN by one TX_CLK cycle, and behave as if TX_ER were asserted during that additional cycle.

Except for the first 10 TX_CLK cycles after TX_EN is de-asserted, any time TX_EN is not asserted, tsr shall assume the value zero_code.

tx_extend

A latched, asynchronous state variable used to extend the TX_EN signal long enough to ensure complete transmission of all nonzero ternary symbols in eop1-5.

Values:	ON and OFF
Set ON upon:	rising edge of TX_EN
Set OFF upon	either of two conditions: a) In the event of a collision (COL is asserted at any time during transmission) set tx_extend=OFF when TX_EN de-asserts. b) In the event of no collision (COL remains de-asserted throughout transmission) set tx_extend=OFF upon completion of transmission of last ternary symbol in eop4.

NOTES

1—The 6T code group eop5 has four zeroes at the end. The 6T code group eop4 contains the last nonzero ternary symbol to be transmitted.

2—The effect of a collision, if present, is to truncate the frame at the original boundary determined by TX_EN. Noncolliding frames are extended, while colliding frames are not.

23.2.4.3 PCS state diagram timer

tw1_timer

A continuous free-running timer.

Values:	The condition tw1_timer_done goes true when the timer expires.
Restart when:	Immediately after expiration (restarting the timer resets condition tw1_timer_done).
Duration:	40 ns nominal.

TX_CLK shall be generated synchronous to tw1_timer (see tolerance required for TX_CLK in 23.5.1.2.10).

On every occurrence of tw1_timer_done, the state diagram advances by one block. The message PMA_UNITDATA request is issued concurrent with tw1_timer_done.

23.2.4.4 PCS state diagram functions

encode()

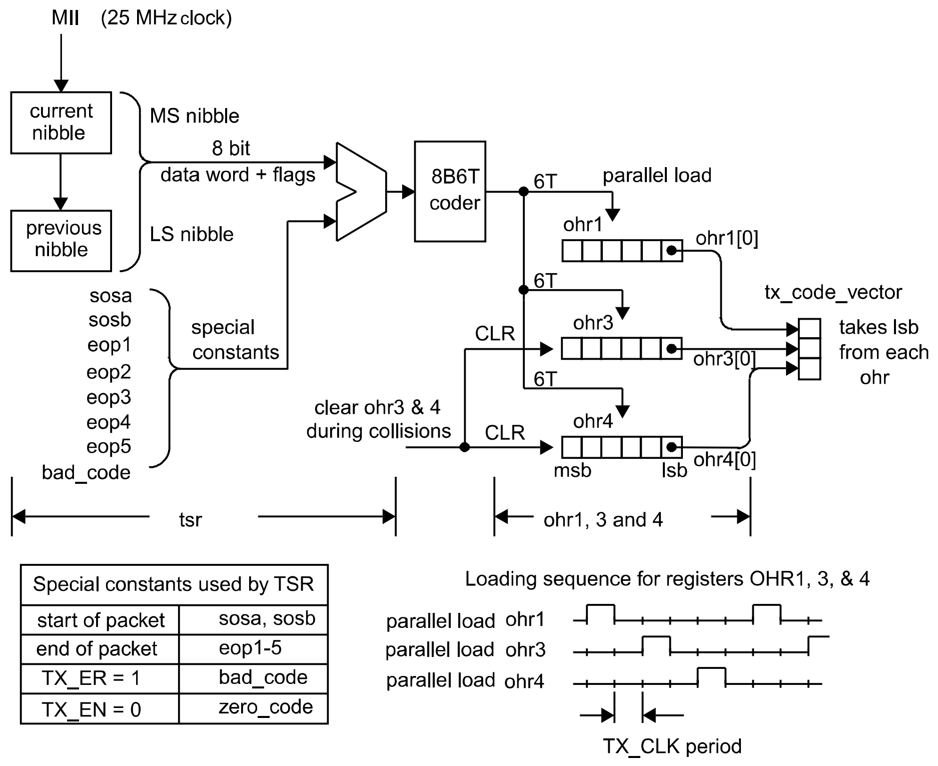
The encode operation of 23.2.1.2.

Argument:	octet
Returns:	6T code group

decode()

The decode operation of 23.2.1.3.

Argument:	6T code group
Returns:	octet



Special constants used by TSR	
start of packet	sosa, sosb
end of packet	eop1-5
TX_ER = 1	bad_code
TX_EN = 0	zero_code

Figure 23-7—PCS Transmit reference diagram

23.2.4.5 PCS state diagrams

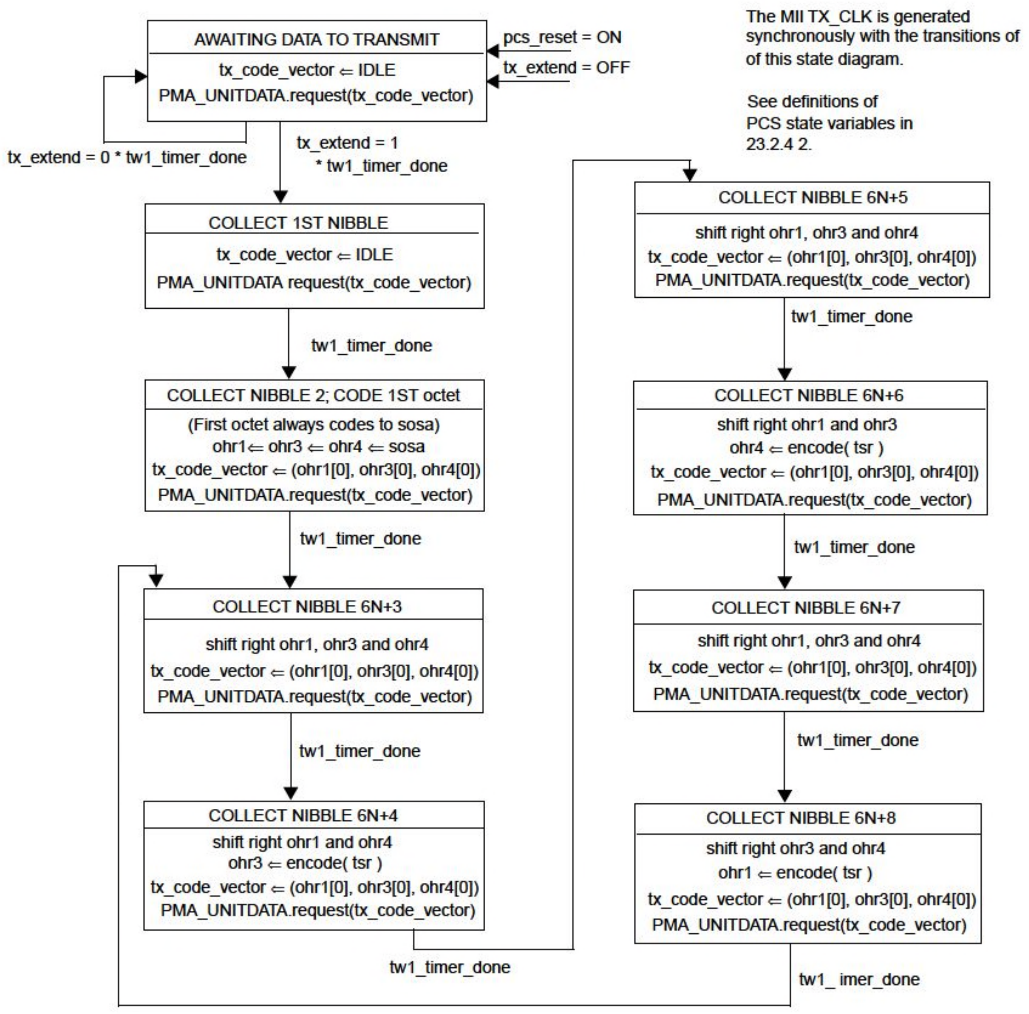

AWAITING DATA TO TRANSMIT
tx_code_vector ⇐ IDLE
PMA_UNITDATA.request(tx_code_vector)
pcs_reset = ON
tx_extend = OFF
tx_extend = 0 * tw1_timer_done
tx_extend = 1 * tw1_timer_done
The MII TX_CLK is generated synchronously with the transitions of of this state diagram.
See definitions of PCS state variables in 23.2.4 2.
COLLECT 1ST NIBBLE
tx_code_vector ⇐ IDLE
PMA_UNITDATA request(tx_code_vector)
tw1_timer_done
COLLECT NIBBLE 2; CODE 1ST octet
(First octet always codes to sosa)
ohr1⇐ ohr3 ⇐ ohr4 ⇐ sosa
tx_code_vector ⇐ (ohr1[0], ohr3[0], ohr4[0])
PMA_UNITDATA.request(tx_code_vector)
tw1_timer_done
COLLECT NIBBLE 6N+3
shift right ohr1, ohr3 and ohr4
tx_code_vector ⇐ (ohr1[0], ohr3[0], ohr4[0])
PMA_UNITDATA.request(tx_code_vector)
tw1_timer_done
COLLECT NIBBLE 6N+4
shift right ohr1 and ohr4
ohr3 ⇐ encode(tsr)
tx_code_vector ⇐ (ohr1[0], ohr3[0], ohr4[0])
PMA_UNITDATA.request(tx_code_vector)
tw1_timer_done
COLLECT NIBBLE 6N+5
shift right ohr1, ohr3 and ohr4
tx_code_vector ⇐ (ohr1[0], ohr3[0], ohr4[0])
PMA_UNITDATA.request(tx_code_vector)
tw1_timer_done
COLLECT NIBBLE 6N+6
shift right ohr1 and ohr3
ohr4 ⇐ encode(tsr)
tx_code_vector ⇐ (ohr1[0], ohr3[0], ohr4[0])
PMA_UNITDATA.request(tx_code_vector)
tw1_timer_done
COLLECT NIBBLE 6N+7
shift right ohr1, ohr3 and ohr4
tx_code_vector ⇐ (ohr1[0], ohr3[0], ohr4[0])
PMA_UNITDATA.request(tx_code_vector)
tw1_timer_done
COLLECT NIBBLE 6N+8
shift right ohr3 and ohr4
ohr1 ⇐ encode(tsr)
tx_code_vector ⇐ (ohr1[0], ohr3[0], ohr4[0])
PMA_UNITDATA.request(tx_code_vector)
tw1_ imer_done

Figure 23-8—PCS Transmit state diagram