24.1.4.2 Physical Medium Attachment (PMA) sublayer

The PMA provides a medium-independent means for the PCS and other bit-oriented clients (e.g., repeaters) to support the use of a range of physical media. The 100BASE-X PMA performs the following functions:

a) Mapping of transmit and receive code-bits between the PMA's client and the underlying PMD;
b) Generating a control signal indicating the availability of the PMD to a PCS or other client, also synchronizing with Auto-Negotiation when implemented;
c) Optionally, generating indications of activity (carrier) and carrier errors from the underlying PMD;
d) Optionally, sensing receive channel failures and transmitting the Far-End Fault Indication; and detecting the Far-End Fault Indication; and
e) Recovery of clock from the NRZI data supplied by the PMD.

24.1.4.3 Physical Medium Dependent (PMD) sublayer

100BASE-X uses the FDDI signaling standards ISO 9314-3: 1990 and ANSI X3.263: 199X (TP-PMD). These signaling standards, called PMD sublayers, define 125 Mb/s, full-duplex signaling systems that accommodate multi-mode optical fiber, STP and UTP wiring. 100BASE-X uses the PMDs specified in these standards with the PMD Service Interface specified in 24.4.1.

The MDI, logically subsumed within the PMD, provides the actual medium attachment, including connectors, for the various supported media.

100BASE-X does not specify the PMD and MDI other than including the appropriate standard by reference along with the minor adaptations necessary for 100BASE-X. Figure 24-2 depicts the relationship between 100BASE-X and the PMDs of ISO 9314-3: 1990 (for 100BASE-FX) and ANSI X3.263: 199X (for 100BASE-TX). The PMDs (and MDIs) for 100BASE-TX and 100BASE-FX are specified in subsequent clauses of this standard.

24.1.5 Inter-sublayer interfaces

There are a number of interfaces employed by 100BASE-X. Some (such as the PMA and PMD interfaces) use an abstract service model to define the operation of the interface. The PCS Interface is defined as a set of physical signals, in a medium-independent manner (MII). Figure 24-3 depicts the relationship and mapping of the services provided by all of the interfaces relevant to 100BASE-X.

It is important to note that, while this specification defines interfaces in terms of bits, nibbles, and code-groups, implementations may choose other data path widths for implementation convenience. The only exceptions are: a) the MII, which, when implemented, uses a nibble-wide data path as specified in clause 22, and b) the MDI, which uses a serial, physical interface.
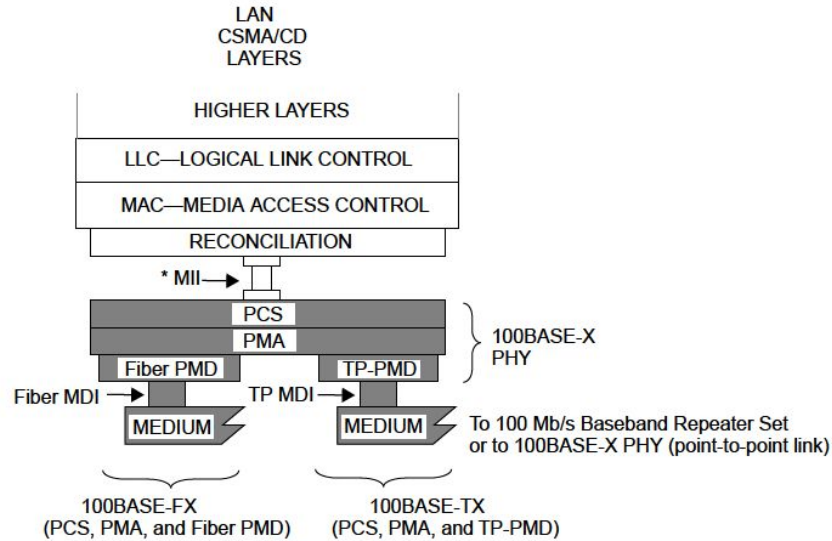
24.1.6 Functional block diagram

Figure 24-4 provides a functional block diagram of the 100BASE-X PHY.

24.1.7 State diagram conventions

The body of this standard is comprised of state diagrams, including the associated definitions of variables, constants, and functions. Should there be a discrepancy between a state diagram and descriptive text, the state diagram prevails.

The notation used in the state diagrams follows the conventions of 21.5; state diagram timers follow the conventions of 14.2.3.2.

Figure 24-2—Relationship of 100BASE-X and the PMDs

24.2 Physical Coding Sublayer (PCS)

24.2.1 Service Interface (MII)

The PCS Service Interface allows the 100BASE-X PCS to transfer information to and from the MAC (via the Reconciliation sublayer) or other PCS client, such as a repeater. The PCS Service Interface is precisely defined as the Media Independent Interface (MII) in clause 22.

In this clause, the setting of MII variables to TRUE or FALSE is equivalent, respectively, to "asserting" or "de-asserting" them as specified in clause 22.

24.2.2 Functional requirements

The PCS comprises the Transmit, Receive, and Carrier Sense functions for 100BASE-T. In addition, the collisionDetect signal required by the MAC (COL on the MII) is derived from the PMA code-bit stream. The PCS shields the Reconciliation sublayer (and MAC) from the specific nature of the underlying channel. Specifically for receiving, the 100BASE-X PCS passes to the MII a sequence of data nibbles derived from incoming code-groups, each comprised of five code-bits, received from the medium. Code-group alignment and MAC packet delimiting is performed by embedding special non-data code-groups. The MII uses a nibble-wide, synchronous data path, with packet delimiting being provided by separate TX_EN and RX_DV
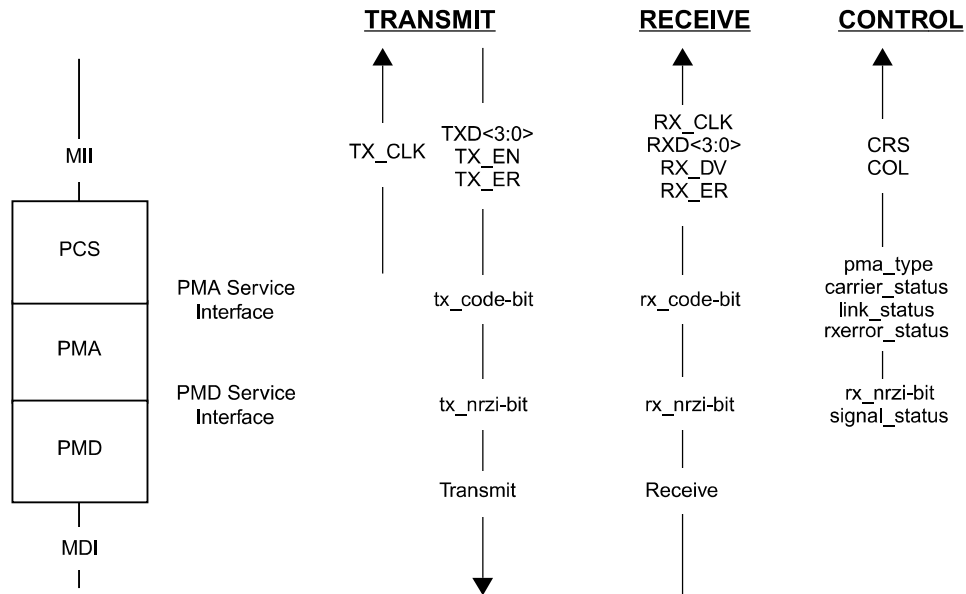
160

TRANSMIT   **RECEIVE**   **CONTROL**

MII

PCS

PMA Service
Interface

PMA

PMD Service
Interface

PMD

MDI

TX_CLK   TXD<3:0>
TX_EN
TX_ER

RX_CLK
RXD<3:0>
RX_DV
RX_ER

CRS
COL

tx_code-bit   rx_code-bit

pma_type
carrier_status
link_status
rxerror_status

tx_nrzi-bit   rx_nrzi-bit

rx_nrzi-bit
signal_status

Transmit   Receive

Figure 24-3—Interface mapping

signals. The PCS provides the functions necessary to map these two views of the exchanged data. The process is reversed for transmit.

The following provides a detailed specification of the functions performed by the PCS, which comprise five parallel processes (Transmit, Transmit Bits, Receive, Receive Bits, and Carrier Sense). Figure 24-4 includes a functional block diagram of the PCS.

The Receive Bits process accepts continuous code-bits via the PMA_UNITDATA.indicate primitive. Receive monitors these bits and generates RXD <3:0>, RX_DV and RX_ER on the MII, and the internal flag, receiving, used by the Carrier Sense and Transmit processes.

The Transmit process generates continuous code-groups based upon the TXD <3:0>, TX_EN, and TX_ER signals on the MII. These code-groups are transmitted by Transmit Bits via the PMA_UNITDATA request primitive. The Transmit process generates the MII signal COL based on whether a reception is occurring simultaneously with transmission. Additionally, it generates the internal flag, transmitting, for use by the Carrier Sense process.

The Carrier Sense process asserts the MII signal CRS when either transmitting or receiving is TRUE. Both the Transmit and Receive processes monitor link_status via the PMA_LINK.indicate primitive, to account for potential link failure conditions.

24.2.2.1 Code-groups

The PCS maps four-bit nibbles from the MII into five-bit code-groups, and vice versa, using a 4B/5B block coding scheme. A code-group is a consecutive sequence of five code-bits interpreted and mapped by the PCS. Implicit in the definition of a code-group is an establishment of code-group boundaries by an alignment function within the PCS Receive process. It is important to note that, with the sole exception of the SSD, which is used to achieve alignment, code-groups are undetectable and have no meaning outside the 100BASE-X physical protocol data unit, called a "stream."

The coding method used, derived from ISO 9314-1: 1989, provides

M I I

TXD <3:0>
TX_EN
TX_ER

TX_CLK
COL

CRS

RXD <3:0>
RX_DV
RX_ER
RX_CLK

PCS

CARRIER
SENSE

transmitting

receiving

TRANSMIT

RECEIVE

tx_bits [4:0]

rx_bits [9:0]

TRANSMIT BITS

RECEIVE BITS

tx_code-bit

link_status

rx_code-bit

carrier_status
rxerror_status

PMA

CARRIER
DETECT

FAR-END FAULT
GENERATE

faulting

LINK MONITOR

FAR-END FAULT
DETECT

TX

RX

tx_nrzi-bit

signal_status

link_control

rx_nrzi-bit

PMD x

Transmit

Receive

M D I

Figure 24-4—Functional block diagram

a) Adequate codes (32) to provide for all Data code-groups (16) plus necessary control code-groups;
b) Appropriate coding efficiency (4 data bits per 5 code-bits; 80%) to effect a 100 Mb/s Physical Layer interface on a 125 Mb/s physical channel as provided by FDDI PMDs; and
c) Sufficient transition density to facilitate clock recovery (when not scrambled).

Table 24-1 specifies the interpretation assigned to each five bit code-group, including the mapping to the nibble-wide (TXD or RXD) Data signals on the MII. The 32 code-groups are divided into four categories, as shown.

For clarity in the remainder of this clause, code-group names are shown between /slashes/. Code-group sequences are shown in succession, e.g.: /1/2/....

The indicated code-group mapping is identical to ISO 9314-1: 1989, with four exceptions:

a) The FDDI term *symbol* is avoided in order to prevent confusion with other 100BASE-T terminology. In general, the term *code-group* is used in its place.
b) The /S/ and /Q/ code-groups are not used by 100BASE-X and are interpreted as INVALID.
c) The /R/ code-group is used in 100BASE-X as the second code-group of the End-of-Stream delimiter rather than to indicate a Reset condition.
d) The /H/ code-group is used to propagate receive errors rather than to indicate the Halt Line State.

24.2.2.1.1 Data code-groups

A Data code-group conveys one nibble of arbitrary data between the MII and the PCS. The sequence of Data code-groups is arbitrary, where any Data code-group can be followed by any other Data code-group. Data code-groups are coded and decoded but not interpreted by the PCS. Successful decoding of Data code-groups depends on proper receipt of the Start-of-Stream delimiter sequence, as defined in table 24-1.

24.2.2.1.2 Idle code-groups

The Idle code-group (/I/) is transferred between streams. It provides a continuous fill pattern to establish and maintain clock synchronization. Idle code-groups are emitted from, and interpreted by, the PCS.

24.2.2.1.3 Control code-groups

The Control code-groups are used in pairs (/J/K/, /T/R/) to delimit MAC packets. Control code-groups are emitted from, and interpreted by, the PCS.

24.2.2.1.4 Start-of-Stream Delimiter (/J/K/)

A Start-of-Stream Delimiter (SSD) is used to delineate the boundary of a data transmission sequence and to authenticate carrier events. The SSD is unique in that it may be recognized independently of previously established code-group boundaries. The Receive function within the PCS uses the SSD to establish code-group boundaries. A SSD consists of the sequence /J/K/.

On transmission, the first 8 bits of the MAC preamble are replaced by the SSD, a replacement that is reversed on reception.

24.2.2.1.5 End-of-Stream delimiter (/T/R/)

An End-of-Stream delimiter (ESD) terminates all normal data transmissions. Unlike the SSD, an ESD cannot be recognized independent of previously established code-group boundaries. An ESD consists of the sequence /T/R/.

Table 24-1—4B/5B code-groups

	PCS code-group [4:0] 4 3 2 1 0	Name	MII (TXD/RXD) <3:0> 3 2 1 0	Interpretation
D A T A	1 1 1 1 0	0	0 0 0 0	Data 0
	0 1 0 0 1	1	0 0 0 1	Data 1
	1 0 1 0 0	2	0 0 1 0	Data 2
	1 0 1 0 1	3	0 0 1 1	Data 3
	0 1 0 1 0	4	0 1 0 0	Data 4
	0 1 0 1 1	5	0 1 0 1	Data 5
	0 1 1 1 0	6	0 1 1 0	Data 6
	0 1 1 1 1	7	0 1 1 1	Data 7
	1 0 0 1 0	8	1 0 0 0	Data 8
	1 0 0 1 1	9	1 0 0 1	Data 9
	1 0 1 1 0	A	1 0 1 0	Data A
	1 0 1 1 1	B	1 0 1 1	Data B
	1 1 0 1 0	C	1 1 0 0	Data C
	1 1 0 1 1	D	1 1 0 1	Data D
	1 1 1 0 0	E	1 1 1 0	Data E
	1 1 1 0 1	F	1 1 1 1	Data F
	1 1 1 1 1	I	undefined	IDLE; used as inter-stream fill code
C O N T R O L	1 1 0 0 0	J	0 1 0 1	Start-of-Stream Delimiter, Part 1 of 2; always used in pairs with K
	1 0 0 0 1	K	0 1 0 1	Start-of-Stream Delimiter, Part 2 of 2; always used in pairs with J
	0 1 1 0 1	T	undefined	End-of-Stream Delimiter, Part 1 of 2; always used in pairs with R
	0 0 1 1 1	R	undefined	End-of-Stream Delimiter, Part 2 of 2; always used in pairs with T
I N V A L I D	0 0 1 0 0	H	Undefined	Transmit Error; used to force signaling errors
	0 0 0 0 0	V	Undefined	Invalid code
	0 0 0 0 1	V	Undefined	Invalid code
	0 0 0 1 0	V	Undefined	Invalid code
	0 0 0 1 1	V	Undefined	Invalid code
	0 0 1 0 1	V	Undefined	Invalid code
	0 0 1 1 0	V	Undefined	Invalid code
	0 1 0 0 0	V	Undefined	Invalid code
	0 1 1 0 0	V	Undefined	Invalid code
	1 0 0 0 0	V	Undefined	Invalid code
	1 1 0 0 1	V	Undefined	Invalid code

24.2.2.1.6 Invalid code-groups

The /H/ code-group indicates that the PCS's client wishes to indicate a Transmit Error to its peer entity. The normal use of this indicator is for repeaters to propagate received errors. Transmit Error code-groups are emitted from the PCS, at the request of the PCS's client through the use of the TX_ER signal, as described in 24.2.4.2.
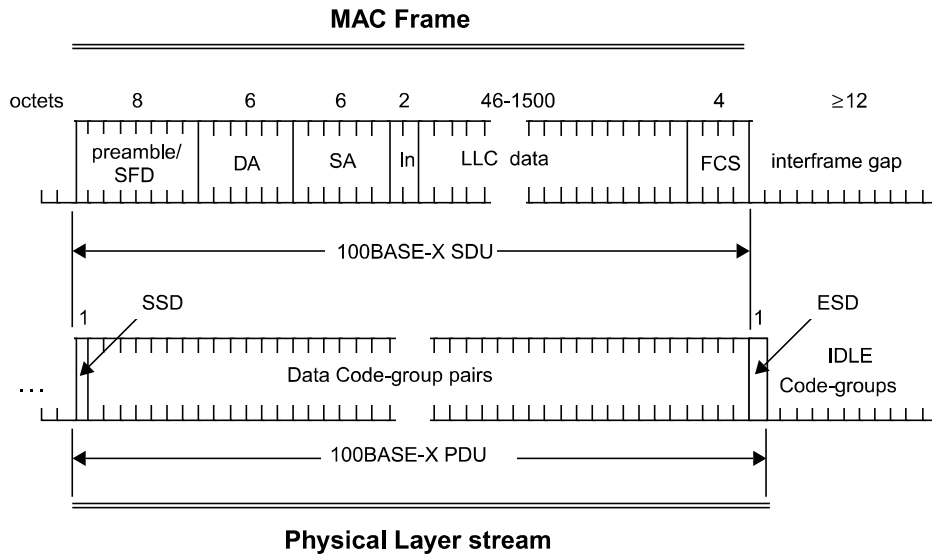
The presence of any invalid code-group on the medium, including /H/, denotes a collision artifact or an error condition. Invalid code-groups are not intentionally transmitted onto the medium by DTE's. The PCS indicates the reception of an Invalid code-group on the MII through the use of the RX_ER signal, as described in 24.2.4.4.

24.2.2.2 Encapsulation

The 100BASE-X PCS accepts frames from the MAC through the Reconciliation sublayer and MII. Due to the continuously signaled nature of the underlying PMA, and the encoding performed by the PCS, the 100BASE-X PCS encapsulates the MAC frame (100BASE-X Service Data Unit, SDU) into a Physical Layer stream (100BASE-X Protocol Data Unit, PDU).

Except for the two code-group SSD, data nibbles within the SDU (including the non-SSD portions of the MAC preamble and SFD) are not interpreted by the 100BASE-X PHY. The conversion from a MAC frame to a Physical Layer stream and back to a MAC frame is transparent to the MAC.

Figure 24-5 depicts the mapping between MAC frames and Physical Layer streams.

MAC Frame

| octets | 8 | 6 | 6 | 2 | 46-1500 | 4 | ≥12 |

preamble/SFD | DA | SA | ln | LLC data | FCS | interframe gap

100BASE-X SDU

SSD | ESD

1 | 1

Data Code-group pairs | IDLE Code-groups

100BASE-X PDU

Physical Layer stream

Figure 24-5—PCS encapsulation

A properly formed stream can be viewed as comprising three elements:

a) *Start-of-Stream Delimiter.* The start of a Physical Layer stream is indicated by a SSD, as defined in 24.2.2.1. The SSD replaces the first octet of the preamble from the MAC frame and vice versa.

b) *Data Code-groups.* Between delimiters (SSD and ESD), the PCS conveys Data code-groups corresponding to the data nibbles of the MII. These Data code-groups comprise the 100BASE-X Service Data Unit (SDU). Data nibbles within the SDU (including those corresponding to the MAC preamble and SFD) are not interpreted by the 100BASE-X PCS.

c) *End-of-Stream Delimiter.* The end of a properly formed stream is indicated by an ESD, as defined in 24.2.2.1. The ESD is transmitted by the PCS following the de-assertion of TX_EN on the MII, which corresponds to the last data nibble composing the FCS from the MAC. It is transmitted during the period considered by the MAC to be the interframe gap (IFG). On reception, ESD is interpreted by the PCS as terminating the SDU.

Between streams, IDLE code-groups are conveyed between the PCS and PMA.

24.2.2.3 Data delay

The PCS maps a non-aligned code-bit data path from the PMA to an aligned, nibble-wide data path on the MII, both for Transmit and Receive. Logically, received bits must be buffered to facilitate SSD detection and alignment, coding translation, and ESD detection. These functions necessitate an internal PCS delay of at least two code-groups. In practice, alignment may necessitate even longer delays of the incoming code-bit stream.

When the MII is present as an exposed interface, the MII signals TX_CLK and RX_CLK, not depicted in the following state diagrams, shall be generated by the PCS in accordance with clause 22.

24.2.2.4 Mapping between MII and PMA

Figure 24-6 depicts the mapping of the nibble-wide data path of the MII to the five-bit-wide code-groups (internal to the PCS) and the code-bit path of the PMA interface.
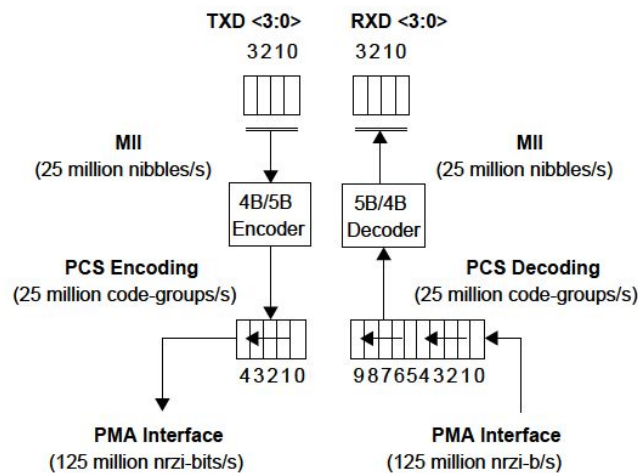
Figure 24-6—PCS reference diagram

Upon receipt of a nibble from the MII, the PCS encodes it into a five-bit code-group, according to 24.2.2.1. Code-groups are serialized into code-bits and passed to the PMA for transmission on the underlying medium, according to figure 24-6. The first transmitted code-bit of a code-group is bit 4, and the last code-bit transmitted is bit 0. There is no numerical significance ascribed to the bits within a code-group; that is, the code-group is simply a five-bit pattern that has some predefined interpretation.

Similarly, the PCS deserializes code-bits received from the PMA, according to figure 24-6. After alignment is achieved, based on SSD detection, the PCS converts code-groups into MII data nibbles, according to 24.2.2.1.

24.2.3 State variables

24.2.3.1 Constants

DATA
> The set of 16 code-groups corresponding to valid DATA, as specified in 24.2.2.1. (In the Receive state diagram, the set operators $\in$ and $\notin$ are used to represent set membership and non-membership, respectively.)

ESD
> The code-group pair corresponding to the End-of-Stream delimiter, as specified in 24.2.2.1.

ESD1
> The code-group pair corresponding to the End-of-Stream delimiter, Part 1 (/T/), as specified in 24.2.2.1.

ESD2
> The code-group pair corresponding to the End-of-Stream delimiter, Part 2 (/R/), as specified in 24.2.2.1.

HALT
> The Transmit Error code-group (/H/), as specified in 24.2.2.1.

IDLE
> The IDLE code-group, as specified in 24.2.2.1.

IDLES
> A code-group pair comprised of /I/I/; /I/ as specified in 24.2.2.1.

SSD
> The code-group pair corresponding to the Start-of-Stream delimiter, as specified in 24.2.2.1.

SSD1
> The code-group corresponding to the Start-of-Stream delimiter, Part 1 (/J/), as specified in 24.2.2.1.

SSD2
> The code-group corresponding to the Start-of-Stream delimiter, Part 2 (/K/), as specified in 24.2.2.1.

24.2.3.2 Variables

In the following, values for the MII parameters are definitively specified in clause 22.

COL
> The COL signal of the MII as specified in clause 22.

CRS
> The CRS signal of the MII as specified in clause 22.

link_status
> The link_status parameter as communicated by the PMA_LINK.indicate primitive.

> Values: FAIL; the receive channel is not intact
> READY; the receive channel is intact and ready to be enabled by Auto-Negotiation
> OK; the receive channel is intact and enabled for reception

receiving
> A boolean set by the Receive process to indicate non-IDLE activity (after squelch). Used by the Carrier Sense process, and also interpreted by the Transmit process for indicating a collision.

> Values: TRUE; unsquelched carrier being received
> FALSE; carrier not being received

rx_bits [9:0]
> A vector of the 10 most recently received code-bits from the PMA as assembled by Receive Bits and processed by Receive. rx_bits [0] is the most recently received (newest) code-bit; rx_bits [9] is the least recently received code-bit (oldest). When alignment has been achieved, it contains the last two code-groups.

rx_code-bit
> The rx_code-bit parameter as communicated by the most recent PMA_UNITDATA.indicate primitive (that is, the value of the most recently received code-bit from the PMA).

RX_DV
> The RX_DV signal of the MII as specified in clause 22. Set by the Receive process, RX_DV is also interpreted by the Receive Bits process as an indication that rx_bits is code-group aligned.

RX_ER
> The RX_ER signal of the MII as specified in clause 22.

RXD <3:0>
> The RXD <3:0> signal of the MII as specified in clause 22.

transmitting
> A boolean set by the Transmit Process to indicate a transmission in progress. Used by the Carrier Sense process.

> Values: TRUE; the PCS's client is transmitting
> FALSE; the PCS's client is not transmitting

tx_bits [4:0]
> A vector of code-bits representing a code-group prepared for transmission by the Transmit Process and transmitted to the PMA by the Transmit Bits process.

TX_EN
> The TX_EN signal of the MII as specified in clause 22.

TX_ER
> The TX_ER signal of the MII as specified in clause 22.

TXD <3:0>
> The TXD <3:0> signal of the MII as specified in clause 22.

24.2.3.3 Functions

nibble DECODE (code-group)
> In Receive, this function takes as its argument a five-bit code-group and returns the corresponding MII RXD <3:0> nibble, per table 24-1.

code-group ENCODE (nibble)
> In the Transmit process, this function takes as its argument an MII TXD <3:0> nibble, and returns the corresponding five-bit code-group per table 24-1.

SHIFTLEFT (rx_bits)
> In Receive Bits, this function shifts rx_bits left one bit placing rx_bits [8] in rx_bits [9], rx_bits [7] in rx_bits [8] and so on until rx_bits [1] gets rx_bits [0].

24.2.3.4 Timers

code-bit_timer

> In the Transmit Bits process, the timer governing the output of code-bits from the PCS to the PMA and thereby to the medium with a nominal 8 ns period. This timer shall be derived from a fixed frequency oscillator with a base frequency of 125 MHz $\pm$ 0.005% and phase jitter above 20 kHz less than $\pm$ 8°.

24.2.3.5 Messages

gotCodeGroup.indicate

> A signal sent to the Receive process by the Receive Bits process after alignment has been achieved signifying completion of reception of the next code-group in rx_bits(4:0), with the preceding code-group moved to rx_bits [9:5]. rx_bits [9:5] may be considered as the "current" code-group.

PMA_UNITDATA.indicate (rx_code-bit)

> A signal sent by the PMA signifying that the next code-bit from the medium is available in rx_code-bit.

sentCodeGroup.indicate

> A signal sent to the Transmit process from the Transmit Bits process signifying the completion of transmission of the code-group in tx_bits [4:0].

24.2.4 State diagrams

24.2.4.1 Transmit Bits

Transmit Bits is responsible for taking code-groups prepared by the Transmit process and transmitting them to the PMA using PMA_UNITDATA request, the frequency of which determines the transmit clock. Transmit deposits these code-groups in tx_bits with Transmit Bits signaling completion of a code-group transmission with sentCodeGroup.indicate.

The PCS shall implement the Transmit Bits process as depicted in figure 24-7 including compliance with the associated state variables as specified in 24.2.3.

24.2.4.2 Transmit

The Transmit process sends code-groups to the PMA via tx_bits and the Transmit Bits process. When initially invoked, and between streams (delimited by TX_EN on the MII), the Transmit process sources continuous Idle code-groups (/I/) to the PMA. Upon the assertion of TX_EN by the MII, the Transmit process passes an SSD (/J/K/) to the PMA, ignoring the TXD <3:0> nibbles during these two code-group times. Following the SSD, each TXD <3:0> nibble is encoded into a five-bit code-group until TX_EN is deasserted. If, while TX_EN is asserted, the TX_ER signal is asserted, the Transmit process passes Transmit Error code-groups (/H/) to the PMA. Following the de-assertion of TX_EN, an ESD (/T/R/) is generated, after which the transmission of Idle code-groups is resumed by the IDLE state.

Collision detection is implemented by noting the occurrence of carrier receptions during transmissions, following the model of 10BASE-T. The indication of link_status $\neq$ OK by the PMA at any time causes an immediate transition to the IDLE state and supersedes any other Transmit process operations.

The PCS shall implement the Transmit process as depicted in figure 24-8 including compliance with the associated state variables as specified in 24.2.3.
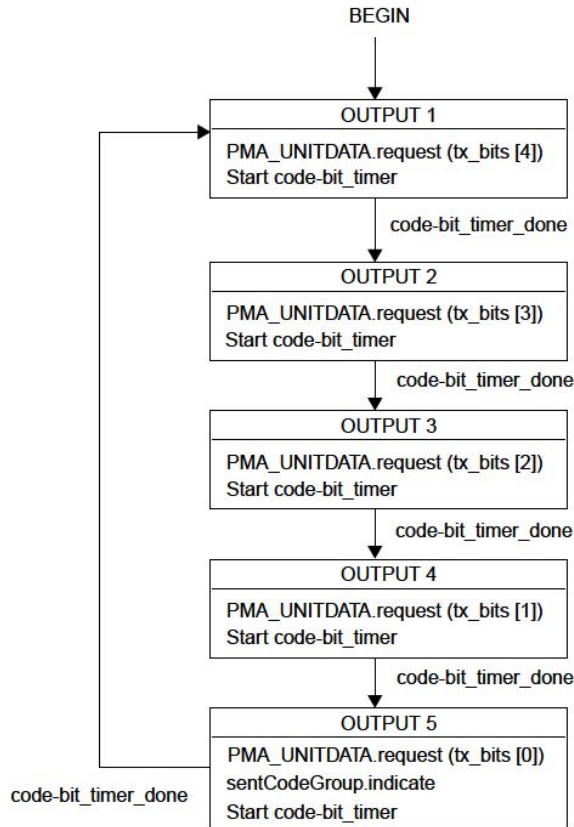
BEGIN

```
                              ┌─────────────────────────────────────┐
                              │            OUTPUT 1                  │
                              ├─────────────────────────────────────┤
                              │ PMA_UNITDATA.request (tx_bits [4])   │
                              │ Start code-bit_timer                 │
                              └─────────────────────────────────────┘
                                          │ code-bit_timer_done
                              ┌─────────────────────────────────────┐
                              │            OUTPUT 2                  │
                              ├─────────────────────────────────────┤
                              │ PMA_UNITDATA.request (tx_bits [3])   │
                              │ Start code-bit_timer                 │
                              └─────────────────────────────────────┘
                                          │ code-bit_timer_done
                              ┌─────────────────────────────────────┐
                              │            OUTPUT 3                  │
                              ├─────────────────────────────────────┤
                              │ PMA_UNITDATA.request (tx_bits [2])   │
                              │ Start code-bit_timer                 │
                              └─────────────────────────────────────┘
                                          │ code-bit_timer_done
                              ┌─────────────────────────────────────┐
                              │            OUTPUT 4                  │
                              ├─────────────────────────────────────┤
                              │ PMA_UNITDATA.request (tx_bits [1])   │
                              │ Start code-bit_timer                 │
                              └─────────────────────────────────────┘
                                          │ code-bit_timer_done
                              ┌─────────────────────────────────────┐
                              │            OUTPUT 5                  │
                              ├─────────────────────────────────────┤
                              │ PMA_UNITDATA.request (tx_bits [0])   │
  code-bit_timer_done         │ sentCodeGroup.indicate               │
                              │ Start code-bit_timer                 │
                              └─────────────────────────────────────┘
```

Figure 24-7—Transmit Bits state diagram

24.2.4.3 Receive Bits

The Receive Bits process collects code-bits from the PMA interface passing them to the Receive process via rx_bits. rx_bits [9:0] represents a sliding, 10-bit window on the PMA code-bits, with newly received code-bits from the PMA (rx_code-bit) being shifted into rx_bits [0]. This is depicted in figure 24-9. Bits are collected serially until Receive indicates alignment by asserting RX_DV, after which Receive Bits signals Receive for every five code-bits accumulated. Serial processing resumes with the de-assertion of RX_DV.

The PCS shall implement the Receive Bits process as depicted in figure 24-10 including compliance with the associated state variables as specified in 24.2.3.

24.2.4.4 Receive

The Receive process state machine can be viewed as comprising two sections: prealigned and aligned. In the prealigned states, IDLE, CARRIER DETECT, and CONFIRM K, the Receive process is waiting for an indication of channel activity followed by a SSD. After successful alignment, the incoming code-groups are decoded while waiting for stream termination.
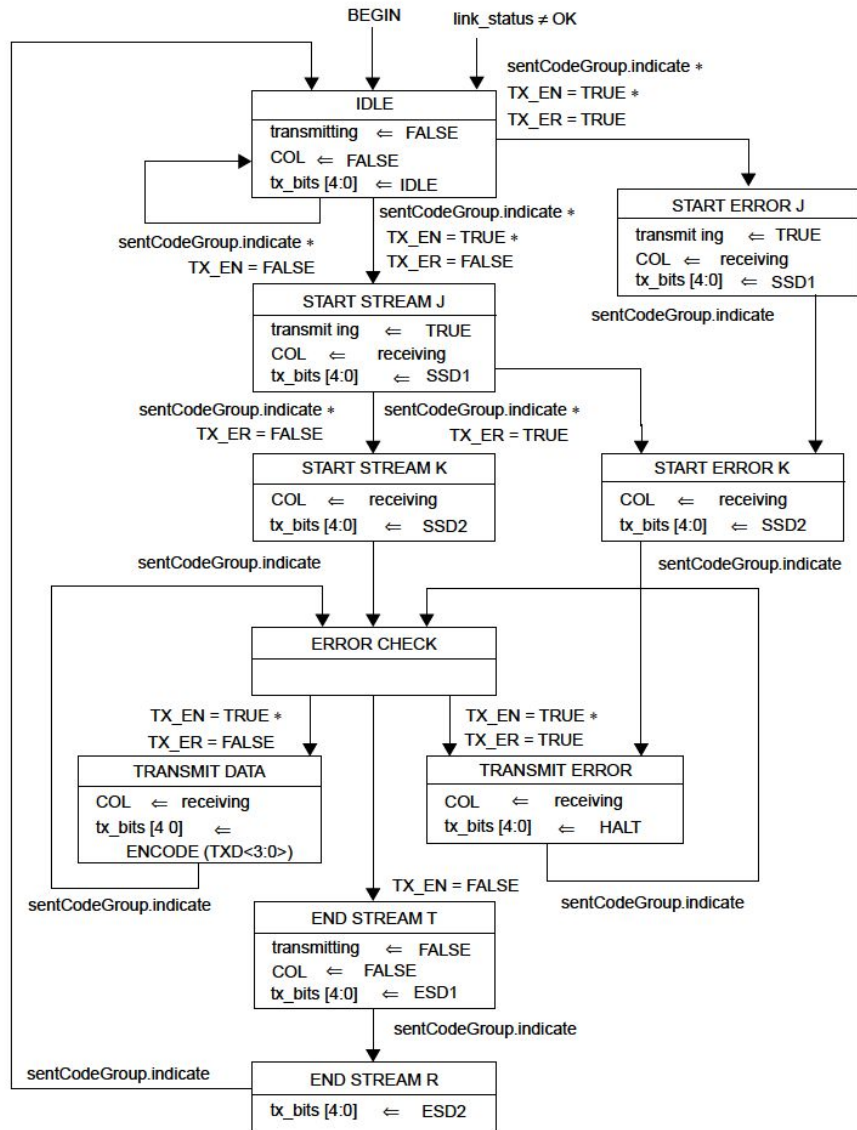
Figure 24-8—Transmit state diagram

24.2.4.4.1 Detecting channel activity

The detection of activity on the underlying channel is used both by the MAC (via the MII CRS signal and the Reconciliation sublayer) for deferral purposes, and by the Transmit process for collision detection. Activity, signaled by the assertion of receiving, is indicated by the receipt of two non-contiguous ZEROS within any 10 code-bits of the incoming code-bit stream.
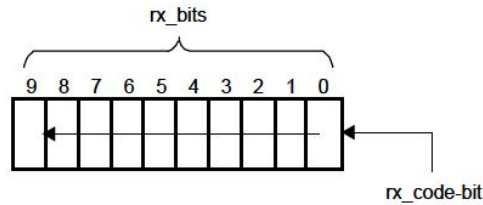
rx_bits

rx_code-bit

Figure 24-9—Receive Bits reference diagram

24.2.4.4.2 Code-group alignment

After channel activity is detected, the Receive process first aligns the incoming code-bits on code-group boundaries for subsequent data decoding. This is achieved by scanning the rx_bits vector for a SSD (/J/K/). The MII RX_DV signal remains deasserted during this time, which ensures that the Reconciliation sublayer will ignore any signals on RXD <3:0>. Detection of the SSD causes the Receive process to enter the START OF STREAM J state.

Well-formed streams contain SSD (/J/K/) in place of the first eight preamble bits. In the event that something else is sensed immediately following detection of carrier, a False Carrier Indication is signaled to the MII by asserting RX_ER and setting RXD to 1110 while RX_DV remains de-asserted. The associated carrier event, as terminated by 10 ONEs, is otherwise ignored.

24.2.4.4.3 Stream decoding

The Receive process substitutes a sequence of alternating ONE and ZERO data-bits for the SSD, which is consistent with the preamble pattern expected by the MAC.

The Receive process then performs the DECODE function on the incoming code-groups, passing decoded data to the MII, including those corresponding to the remainder of the MAC preamble and SFD. The MII signal RX_ER is asserted upon decoding any code-group following the SSD that is neither a valid Data code-group nor a valid stream termination sequence.

24.2.4.4.4 Stream termination

There are two means of effecting stream termination in the Receive process (figure 24-11).

A normal stream termination is caused by detection of an ESD (/T/R/) in the rx_bits vector. In order to preserve the ability of the MAC to properly delimit the FCS at the end of the frame (that is, to avoid incorrect AlignmentErrors in the MAC) the internal signal receiving (and through it, the MII CRS signal, per clause 22) is de-asserted immediately following the last code-bit in the stream that maps to the FCS. Note that the condition link_status ≠ OK during stream reception (that is, when receiving = TRUE) causes an immediate transition to the LINK FAILED state and supersedes any other Receive process operations.

A premature stream termination is caused by the detection of two Idle code-groups (/I/I) in the rx_bits vector prior to an ESD. Note that RX_DV remains asserted during the nibble corresponding to the first five contiguous ONEs while RX_ER is signaled on the MII. RX_ER is also asserted in the LINK FAILED state, which ensures that RX_ER remains asserted for sufficient time to be detected.
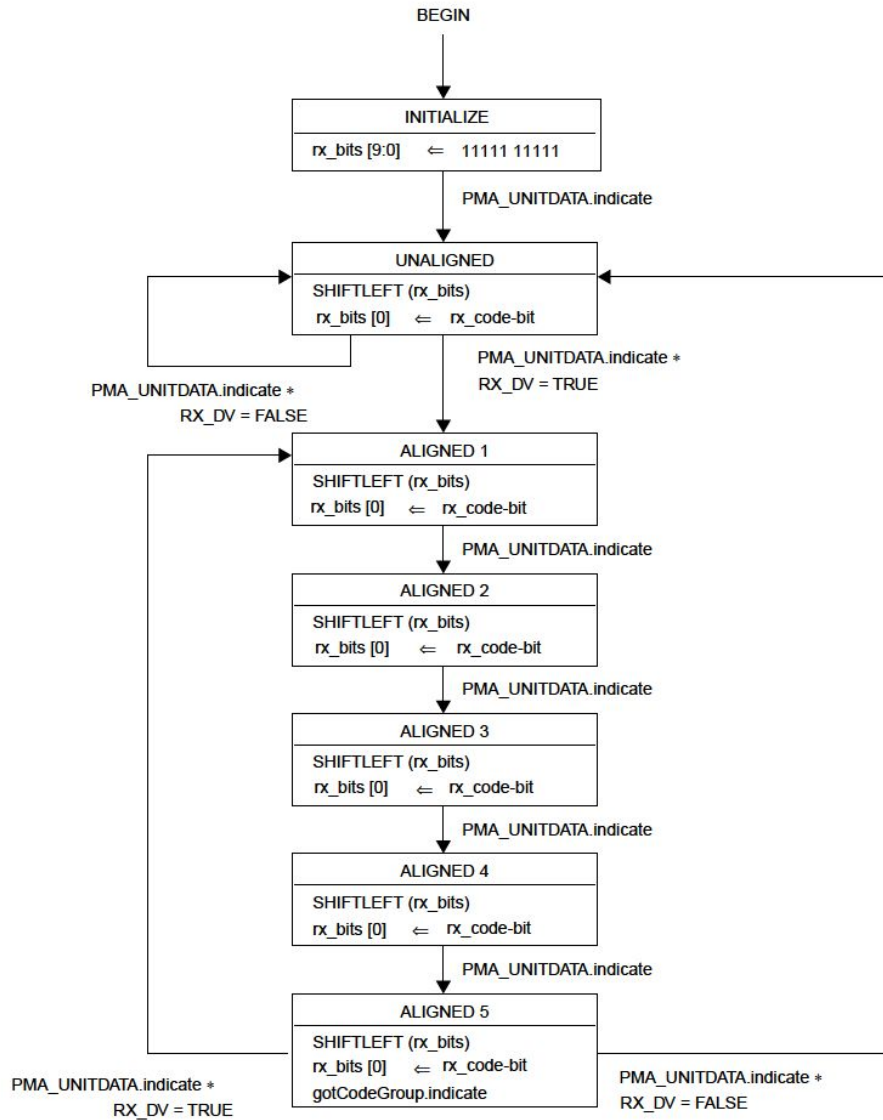
Stream termination causes a transition to the IDLE state.

BEGIN

INITIALIZE

rx_bits [9:0] ⇐ 11111 11111

PMA_UNITDATA.indicate

UNALIGNED

SHIFTLEFT (rx_bits)

rx_bits [0] ⇐ rx_code-bit

PMA_UNITDATA.indicate *
RX_DV = TRUE

PMA_UNITDATA.indicate *
RX_DV = FALSE

ALIGNED 1

SHIFTLEFT (rx_bits)

rx_bits [0] ⇐ rx_code-bit

PMA_UNITDATA.indicate

ALIGNED 2

SHIFTLEFT (rx_bits)

rx_bits [0] ⇐ rx_code-bit

PMA_UNITDATA.indicate

ALIGNED 3

SHIFTLEFT (rx_bits)

rx_bits [0] ⇐ rx_code-bit

PMA_UNITDATA.indicate

ALIGNED 4

SHIFTLEFT (rx_bits)

rx_bits [0] ⇐ rx_code-bit

PMA_UNITDATA.indicate

ALIGNED 5

SHIFTLEFT (rx_bits)

rx_bits [0] ⇐ rx_code-bit

gotCodeGroup.indicate

PMA_UNITDATA.indicate *
RX_DV = TRUE

PMA_UNITDATA.indicate *
RX_DV = FALSE

Figure 24-10—Receive Bits state diagram

The PCS shall implement the Receive process as depicted in figure 24-11 including compliance with the associated state variables as specified in 24.2.3.

24.2.4.5 Carrier Sense

The Carrier Sense process generates the signal CRS on the MII, which (via the Reconciliation sublayer) the MAC uses for frame receptions and for deferral. The process operates by performing a logical OR operation on the internal messages receiving and transmitting, generated by the Receive and Transmit processes, respectively.
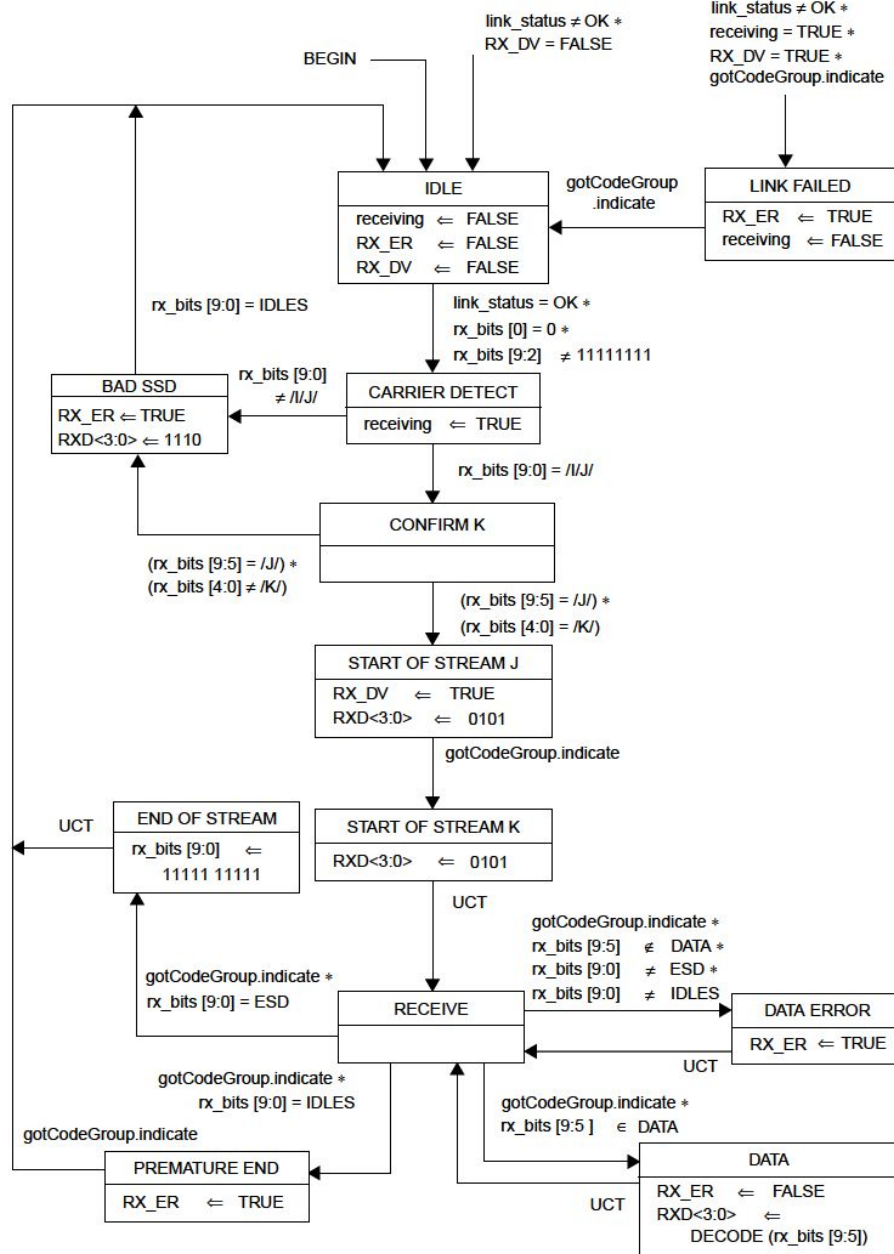
Figure 24-11—Receive state diagram