101. Reconciliation Sublayer, Physical Coding Sublayer, and Physical Media Attachment for EPoC

101.1 Overview

This clause describes the Reconciliation Sublayer (RS), Physical Coding Sublayer (PCS) with FEC, and Physical Medium Attachment (PMA) used with 10GPASS-XR point-to-multipoint (P2MP) networks. These are passive or amplified multipoint coaxial cable distribution networks (CCDN) that connect multiple DTEs using a single shared coaxial link. The architecture is asymmetric, based on a tree and branch topology utilizing coaxial taps and splitters. This type of network requires that the Multipoint MAC Control sublayer exists above MAC instances, as described in Clause 103.

101.1.1 Conventions

The notation used in the state diagrams in this clause follows the conventions in 21.5. Should there be a discrepancy between a state diagram and descriptive text, the state diagram prevails. The notation "+ +" after a counter indicates it is to be incremented by 1. The notation "– –" after a counter indicates it is to be decremented by 1. The notation "–=" after a counter indicates that the counter value is to be decremented by the following value. The notation "+=" after a counter indicates that the counter value is to be incremented by the following value. Code examples given in this clause adhere to the style of the "C" programming language.

For equations used in this clause the symbol $\lceil x \rceil$ represents a ceiling function that rounds up ts argument *x* to the next highest integer. The notation $\lfloor x \rfloor$ represents a *floor* function, which returns the value of its argument *x* rounded down to the nearest integer.

101.1.2 Constraints for delay through RS, PCS, and PMA

The operation of EPoC MPCP, as defined in Clause 103, relies on strict timing based on the distribution of timestamps. The actual delay is implementation dependent but an implementation shall maintain a combined delay variation through RS, PCS, and PMA sublayers of no more than 1 time_quantum (see 77.2.2.1) so as not to interfere with the MPCP timing and operation.

101.1.3 Mapping of PCS, and PMA variables

The optional MDIO capability described in Clause 45 defines several variables that may provide control and status information for and about the 10GPASS-XR Phy or are communicated between the CLT and the CNU via the PHY Link. The mapping of MDIO control and status variables to PCS/PMA variables is shown in Table 101–1.

Table 101–1—Variable to MDIO register mapping

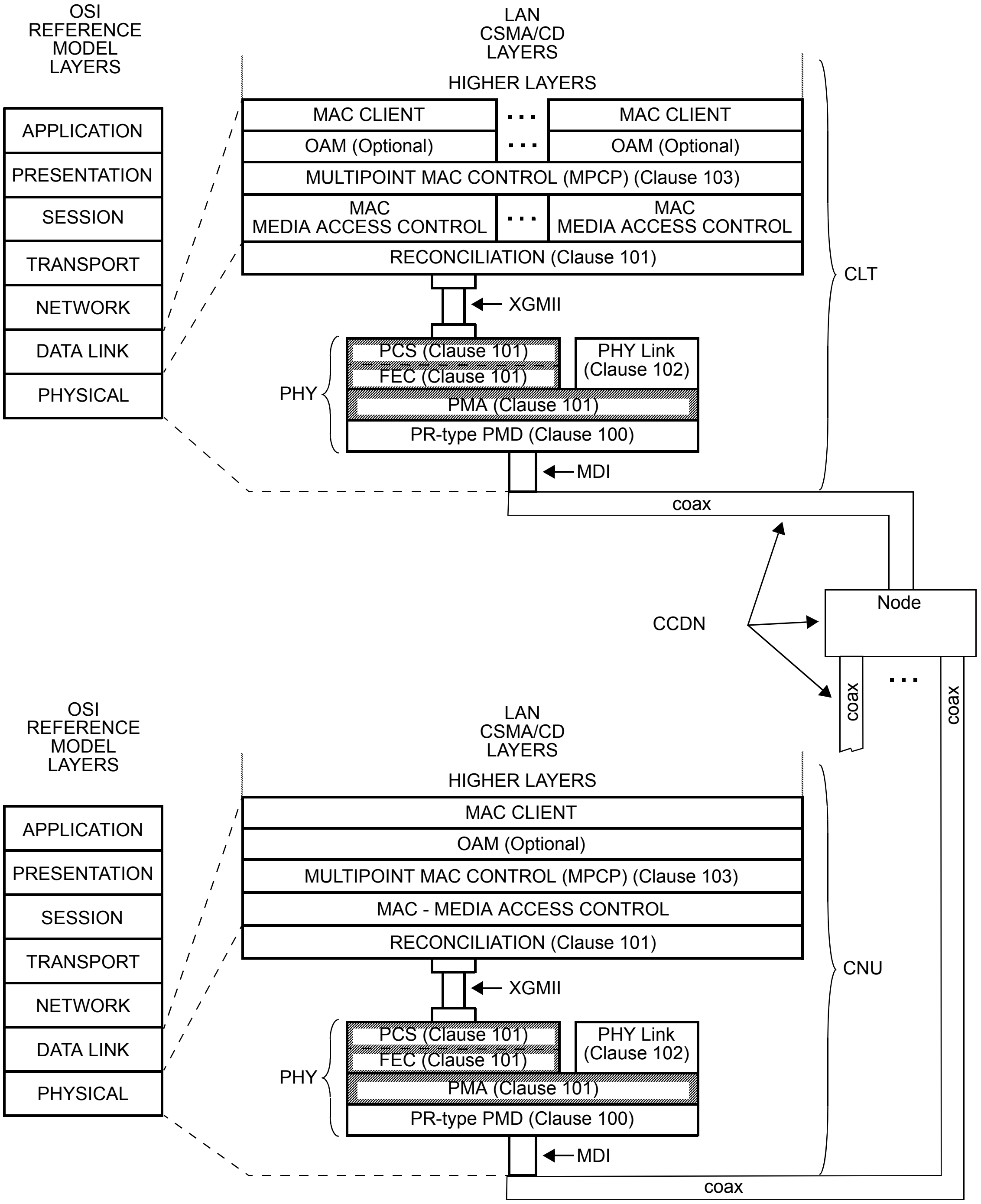
MDIO parameter name	PMA/PMD register name	Register / bit number	PMA/PMD variable		
			Name	Index	Bits(s)
CRC40 Errors	10GPASS-XR control	1900.2	*CRC40ErrCtrl*	0	2
Continuous Pilot Scaling Factor	10GPASS-XR control	1.1900.9:3	*CntPltSF*	0	9:3
DS cyclic prefix	DS OFDM control	1.1901.3:0	*DSNcp*	1	3:0
DS windowing	DS OFDM control	1.1901.6:4	*DSNrp*	1	6:4
DS time interleaving	DS OFDM control	1.1901.11:7	*DS_TmIntrlv*	1	11:7
DS OFDM freq ch 1	DS OFDM channel frequency control 1	1.1902.15:0	*DS_FreqCh1*	2	15:0
DS OFDM freq ch 2	DS OFDM channel frequency control 2	1.1903.15:0	*DS_FreqCh2*	3	15:0
DS OFDM freq ch 3	DS OFDM channel frequency control 3	1.1904.15:0	*DS_FreqCh3*	4	15:0
DS OFDM freq ch 4	DS OFDM channel frequency control 4	1.1905.15:0	*DS_FreqCh4*	5	15:0
DS OFDM freq ch 5	DS OFDM channel frequency control 5	1.1906.15:0	*DS_FreqCh5*	6	15:0
RB size	US OFDM control	1.1907.7	*RBsize*	7	7
US windowing	US OFDM control	1.1907.6:4	*USNrp*	7	6:4
US cyclic prefix	US OFDM control	1.1907.3:0	*USNcp*	7	3:0
Type 2 Repeat	US OFDMA pilot pattern	1.1909.14:12	*Type2_Repeat*	9	14:12
Type 2 Start	US OFDMA pilot pattern	1.1909.11:8	*Type2_Start*	9	11:8
Type 1 Repeat	US OFDMA pilot pattern	1.1909.6:4	*Type1_Repeat*	9	6:4
Type 1 Start	US OFDMA pilot pattern	1.1909.3:0	*Type1_Start*	9	3:0
US copy in process	Profile control	1.1910:11	US_CpyInP	10	11
US profile copy	Profile control	1.1910:10	US_PrflCpy	10	10
US configuration ID	Profile control	1.1910:9:8	US_CID	10	9:8
DS copy in process	Profile control	1.1910:3	DS_CpyInP	10	3
DS profile copy	Profile control	1.1910:2	DS_PrflCpy	10	2
DS configuration ID	Profile control	1.1910:1:0	DS_CID	10	1:0
DS PHY data rate Upper	DS PHY data rate	1.1927.4:0	*DS_DataRate(36:32)*	27	4:0
DS PHY data rate mid	DS PHY data rate	1.1928.15:0	*DS_DataRate(31:16)*	28	15:0
DS PHY data rate lower	DS PHY data rate	1.1929.15:0	*DS_DataRate(15:0)*	29	15:0
US PHY data rate upper	US PHY data rate	1.1930.4:0	*US_DataRate(36:32)*	30	4:0

Table 101–1—Variable to MDIO register mapping *(continued)*

MDIO parameter name	PMA/PMD register name	Register / bit number	PMA/PMD variable		
			Name	Index	Bits(s)
US PHY data rate mid	US PHY data rate	1.1931.5:0	*US_DataRate(31:16)*	30	15:0
US PHY data rate lower	US PHY data rate	1.1932.15:0	*US_DataRate(15:0)*	32	15:0
FEC codeword counter lower	10GPASS-XR FEC codeword counter	1.1933.15:0	*FecCodeWordCount (15:0)*	33	15:0
FEC codeword counter upper	10GPASS-XR FEC codeword counter	1.1934.15:0	*FecCodeWordCount (31:16)*	34	15:0
FEC codeword success counter lower	10GPASS-XR FEC codeword success counter	1.1935.15:0	*FecCodeWordSuccess (15:0)*	35	15:0
FEC codeword success counter upper	10GPASS-XR FEC codeword success counter	1.1936.15:0	*FecCodeWordSuccess (31:16)*	36	15:0
FEC codeword fail counter lower	10GPASS-XR FEC codeword fail counter	1.1937.15:0	*FecCodeWordFail (15:0)*	37	15:0
FEC codeword fail counter upper	10GPASS-XR FEC codeword fail counter	1.1938.15:0	*FecCodeWordFail (31:16)*	38	15:0
DS OFDM channel ID	DS OFDM channel ID register	12.3.2:0	*DS_OFDM_ID*	1000	2:0
DS Modulation Type SC4	10GPASS-XR DS profile descriptor	12.1.3:0	*DS_ModTypeSC(4)*	1001	3:0
DS Modulation Type SC5	10GPASS-XR DS profile descriptor	12.1.7:4	*DS_ModTypeSC(5)*	1001	7:4
DS Modulation Type SC6	10GPASS-XR DS profile descriptor	12.1.11:8	*DS_ModTypeSC(6)*	1001	11:8
DS Modulation Type SC7	10GPASS-XR DS profile descriptor	12.1.15:12	*DS_ModTypeSC(7)*	1001	15:12
DS Modulation Type SC8 through DS Modulation Type 4095	10GPASS-XR DS profile descriptor	12.2 through 12.1023	*DS_ModTypeSC(8)* through *DS_ModTypeSC(4095)*	1002 through 2023	as above
US Modulation Type SC0	10GPASS-XR US profile descriptor	12.1024.3:0	*US_ModTypeSC(0)*	2024	3:0
US Modulation Type SC1	10GPASS-XR US profile descriptor	12.1024.7:4	*US_ModTypeSC(1)*	2024	7:4
US Modulation Type SC2	10GPASS-XR US profile descriptor	12.1024.11:8	*US_ModTypeSC(2)*	2024	11:8
US Modulation Type SC3	10GPASS-XR US profile descriptor	12.1024.15:12	*US_ModTypeSC(3)*	2024	15:12

Table 101–1—Variable to MDIO register mapping *(continued)*

MDIO parameter name	PMA/PMD register name	Register / bit number	PMA/PMD variable		
			Name	**Index**	**Bits(s)**
US Modulation Type SC4 through US Modulation Type 4095	10GPASS-XR US profile descriptor	12.1025 through 12.2047	*US_ModTypeSC(4)* through *US_ModTypeSC(4095)*	2025 through 2047	as above
Real pre-equalizer coefficient SC(0)	10GPASS-XR US pre-equalizer coefficients	12.2048	*EQ_CoefR(0)*	2048	15:0
Imaginary pre-equalizer coefficient SC(0)	10GPASS-XR US pre-equalizer coefficients	12.2049	*EQ_CoefI(0)*	3049	15:0
Real pre-equalizer coefficient SC(1) through Real pre-equalizer coefficient SC(4095)	10GPASS-XR US pre-equalizer coefficients	12.2050 12.2052 ... 12.10236	*EQ_CoefR(1)* through *EQ_CoefR(4095)*	3050 3052 ... 11236	15:0
Imaginary pre-equalizer coefficient SC(1) through Imaginary pre-equalizer coefficient SC(4095)	10GPASS-XR US pre-equalizer coefficients	12.2051 12.2053 ... 12.10237	*EQ_CoefI(0)* through *EQ_CoefI(0)*	3051 3053 ... 11237	15:0



XGMII= 10 GIGABIT MEDIA INDEPENDENT INTERFACE
MDI = MEDIUM DEPENDENT INTERFACE
OAM = OPERATIONS, ADMINISTRATION & MAINTENANCE
CLT = COAXIAL LINE TERMINAL
CCDN= COAX CABLE DISTRIBUTION NETWORK

CNU = COAXIAL NETWORK UNIT
PCS = PHYSICAL CODING SUBLAYER
PHY = PHYSICAL LAYER DEVICE
PMA = PHYSICAL MEDIUM ATTACHMENT
PMD = PHYSICAL MEDIUM DEPENDENT

Figure 101–1—Relationship of EPoC PCS and PMA to the ISO/IEC OSI reference

101.2 Reconciliation Sublayer (RS) for EPoC

101.2.1 Overview of EPoC RS operation

This subclause extends Clause 46 to enable multiple MAC instances to interface with a single Physical Layer, and to enable data links with asymmetric data rates, i.e., with the downstream data rate different than the upstream data rate.

The number of supported MAC instances is limited only by the implementation. It is acceptable for only one MAC instance to be connected to the EPoC Reconciliation Sublayer.

Figure 101–1 shows the relationship between the EPoC RS and the ISO/IEC OSI reference model. The mapping of XGMII signals to PLS service primitives is described in 46.1.7 for XGMII with exceptions noted herein.

101.2.2 Summary of major concepts

A successful registration process, described in 103.3.3, results in the assignment of values to the MODE and LLID variables associated with the given MAC instance. This may be one of many MAC instances in a CLT or a single MAC instance in a CNU. The MODE and LLID variables are used to identify a packet transmitted from that MAC instance and how received packets are directed to that MAC instance. The RS in the CLT shall operate in unidirectional mode as defined in 66.4.

As described in 101.1.2, multiple MAC instances within a CLT are bound to the single XGMII instance. Only one PLS_DATA.request primitive is active at any time. At the CNU, the only MAC instance is bound to the XGMII.

In the transmit direction, the RS maps the active PLS_DATA.request to the XGMII signals (TXD<31:0>, TXC<3:0>, and TX_CLK) according to the MAC instance generating the request. The RS replaces octets of preamble with the values of MODE and LLID variables of the transmitting MAC.

In the receive direction, the MODE and LLID values embedded within the preamble identify the MAC instance to which this packet should be directed. The RS establishes a temporal mapping of the XGMII signals (RXD<31:0>, RXC<3:0> and RX_CLK) to the correct PLS_DATA.indication and PLS_DATA_VALID.indication primitives.

101.2.3 10 Gigabit Media Independent Interface (XGMII)

This subclause describes the interface between the MAC and PHY in a CLT or a CNU. The physical implementation of the interface is primarily intended to be chip-to-chip, but may also be used as a logical interface between ASIC logic modules within an integrated circuit. These interfaces are used to provide media independence, so that an identical media access controller may be used with all 10GPASS-XR PHY types.

101.2.3.1 XGMII structure

The XGMII structure is discussed in 46.1.6, and Figure 46–2 depicts a schematic view of the RS inputs and outputs.

101.2.3.2 XGMII operation

The XGMII operation is discussed in Clause 46.

101.2.4 Functional specifications for multiple MAC instances

This subclause describes the functional specifications for the support of multiple MAC instances connected to a single RS sublayer with EPoC-specific extensions.

101.2.4.1 Variables

The variables of 65.1.3.1 are inherited except the definition of *logical_link_id* is per 76.2.6.1.1.

101.2.4.2 EPoC RS transmit function

The transmit function of the EPoC RS is as described in 65.1.3.2 with exceptions as noted in 76.2.6.1.2. The XGMII transmit function is described in 46.3.1.

101.2.4.3 EPoC RS receive function

The receive function of the EPoC RS is described in 65.1.3.3 with the exceptions as noted in 76.2.6.1.3. The XGMII receive function is described in 46.3.2.

101.3 Physical Coding Sublayer (PCS) for EPoC

101.3.1 Overview

This subclause defines the Physical Coding Sublayer (PCS) for 10GPASS-XR, supporting operation over the point-to-multipoint coaxial medium architecture. The EPoC PCS is specified to support the operation of up to 10 Gb/s in the downstream direction and up to 10 Gb/s in the upstream direction, where the upstream and downstream data rates are configured independently.

This subclause also specifies a forward error correction (FEC) as well as Idle control character insertion and deletion mechanisms. The FEC mechanism increases the available link budget. The FEC codeword additionally includes a CRC40 to ensure that mean time to false frame acceptance of 4.4 x 10^{17} seconds is met. The Idle control character insertion and deletion mechanism accommodates rate adaptation between the MAC and MAC Control Clients operating at 10 Gb/s and the EPoC PCS and PMD sublayers operating at data rates below 10 Gb/s.

Figure 101–1 shows the relationship between the EPoC PCS sublayer and the ISO/IEC OSI reference model. Figure 100–2 illustrates the CLT transmitter functional block diagram, including the PCS, while Figure 100–3 illustrates the CNU transmitter functional block diagram. Figure 100–4 and Figure 100–5 illustrate the functional block diagram of the receive path in the CLT and CNU, respectively in the EPoC PCS.

101.3.2 PCS transmit function

In the CLT, the PCS transmit function operates in a continuous fashion at the data rate of up to 10 Gb/s. In the CNU, the PCS transmit function operates in a burst mode at the data rate of up to 10 Gb/s. Figure 100–2 illustrates the CLT transmitter functional block diagram, while Figure 100–3 illustrates the CNU transmitter functional block diagram..

In the transmit direction, the EPoC PCS includes an Idle Deletion function that performs data rate adaptation and FEC overhead compensation, followed by a 64B/66B Encoder, and a FEC Encoder / Data Detector.

101.3.2.1 Idle deletion process

EDITORS NOTE (to be removed prior to publication) the TF need to do a thorough review of Idle control character deletion process as it is currently written to be applicable to both US & DS and these processes will be very different in EPoC where US/DS rates are different and US has multiple FEC's.

In the transmitting PCS, the Idle deletion process is responsible for deleting excess Idle control characters inserted between individual frames to adjust the data rate enforced by the MAC Control (as defined in Clause 103) to the effective data rate supported by the PCS and PMD. The gaps created within the data stream by the operation of the Idle deletion process are used in one of the following ways:

a) some gaps created by the removal of Idle control characters are filled with FEC parity data (FEC overhead compensation sub-process); and

b) other gaps created by the removal of Idle control characters are discarded in order to decrease the effective data rate between the MAC and PHY (data rate adaptation sub-process).

The Idle deletion process deletes a specific number of 72-bit vectors containing Idle control characters from the data stream composed of a series of 72-bit vectors received from the XGMII. The number of deleted 72-bit vectors containing Idle control characters depends on the EPoC PMD data rate, PMD overhead (including, for example, cyclic prefix), and the size of FEC parity data. The Idle deletion process performs two functions:

a) create gaps by Idle removal to allow for FEC parity / CRC40 and,

b) rate adaptation by Idle removal to adjust from the XGMII rate to the PMD rate.

The operation of the EPoC MPCP defined in Clause 103 ensures that a sufficient number of Idle control characters are present in the data stream, so that the minimum IPG between two adjacent frames is preserved once all excess Idle control characters are removed through the operation of the data rate adaptation and the FEC overhead compensation sub-processes.

101.3.2.1.1 Constants

FEC_OSize
TYPE: real number
The number of 72-bit vectors constituting the parity (overhead) portion of a FEC codeword. To normalize pre-FEC data rate, the Idle deletion process removes FEC_OSize vectors per every PHY_DSize vectors transferred to the 64B/66B Encoder.
Value: 1840/65

EDITORS NOTE (to be removed prior to publication): we should specify a minimum precision for this number.

PHY_DSize
TYPE: 16-bit unsigned integer
The number of 72-bit vectors constituting the payload portion of a FEC codeword. To normalize the effective PCS data rate, the Idle deletion process removes PHY_OSize vectors per every PHY_DSize vectors to the compensation of FEC overhead and PMD derating process.
Value: 220

XGMII_Rate
TYPE: Integer
The data transfer rate of teh XGMII interface.
Value: 10 Gb/s

101.3.2.1.2 Variables

accResidue
TYPE: real number

The variable *accResidue* tracks the accumulation of *PHY_OSizeFrac*, and modifies the *PHY_OSize* with *PHY_RSize*.

EDITORS NOTE (to be removed prior to publication): we should specify a minimum precision for this number.

BEGIN

TYPE: Boolean
This variable is used when initiating operation of the state diagram. It is set to TRUE following initialization and every reset.

countVectorT

TYPE: 16-bit unsigned integer
Counts the number of 72-bit vectors transmitted after the removal of Idle characters as part of data rate adaptation and FEC overhead compensation.

countDelete

TYPE: 16-bit unsigned integer
Counts the number of 72-bit vectors that need to be deleted from the received data stream as part of the data rate adaptation and FEC overhead compensation.

DelayBound

TYPE: 16 bit unsigned
This value represents the number of consecutive 65-bit blocks containing idles used by the Data Detector input and CNU output process to determine end of burst. This value represents the delay sufficient to initiate the transmitter at the CNU and to accomordate timing jitter caused by PMA overhead, such as burst markers, and pilots. This variable is used only by the CNU. By default, the value should be set the same as that for length of the *FIFO_FEC_TX* buffer.

DS_DataRate

See 100.2.6.1.

PCS_Rate:

TYPE: TBD
The transmission rate of PCS data, excluding the control header in the 64/65B line encoder.

$$\mathrm{PCS\_Rate} = DS\_DataRate \times \frac{64}{65} \qquad (101\text{–}1)$$

PHY_OSize

TYPE: 16-bit unsigned integer
To normalize the effective PCS data rate, the Idle deletion process removes PHY_OSize vectors per every PHY_DSize vectors to compensate for FEC overhead and PMD derating process. PHY_OSize is determined by

$$PHY\_OSize = \left\lfloor \frac{XGMII\_Rate - PCS\_Rate}{PCS\_Rate} \times (PHY\_\mathrm{DSize} + FEC\_OSize) + FEC\_OSize \right\rfloor \qquad (101\text{–}2)$$

PHY_OSizeFrac:

TYPE: real number
The fractional part of *PHY_OSize* due to the floor operation in Equation (101–2). The *PHY_OSizeFrac* is given by

$$PHY\_OSizeFrac = \frac{XGMII\_Rate - PCS\_Rate}{PCS\_Rate} \times (PHY\_\mathrm{DSize} + FEC\_OSize) +$$

$$FEC\_OSize - PHY\_OSize$$

EDITORS NOTE (to be removed prior to publication): we should specify a minimum precision for this number.

PHY_RSize

TYPE: 1 bit integer

This variable is used to modify *PHY_OSize* whenever the *accResidue* exceeds 1.

tx_raw<71:0>

This variable is defined in 49.2.13.2.2.

tx_raw_out<71:0>

72-bit vector sent from the output of the Idle deletion process to the 64B/66B Encoder. This vector contains two XGMII transfers mapped as shown for tx_raw<71:0>.

EDITORS NOTE (to be remove prior to publication): Note that the list of variables will be updated per technical decision #45 (http://www.ieee802.org/3/bn/public/decisions/decisions.html) once EPoC-specific FEC and PMD overhead details are settled.

101.3.2.1.3 Counters

countDeleteF

TYPE: 16-bit unsigned integer

Counts the number of 72-bit vectors that need to be deleted from the received data stream as part of the FEC overhead compensation sub-process.

countDeleteP

TYPE: 16-bit unsigned integer

Counts the number of 72-bit vectors that need to be deleted from the received data stream as part of the data rate adaptation sub-process.

countIdleF

TYPE: 16-bit unsigned integer

Counts the number of 72-bit vectors containing Idle control characters or other control vectors as part of the FEC overhead compensation sub-process.

countIdleP

TYPE: 16-bit unsigned integer

Counts the number of 72-bit vectors containing Idle control characters or other control vectors as part of the data rate adaptation sub-process.

countVectorF

TYPE: 16-bit unsigned integer

Counts the number of 72-bit vectors transmitted after the removal of Idle characters as part of the FEC overhead compensation sub-process.

countVectorP

TYPE: 16-bit unsigned integer

Counts the number of 72-bit vectors transmitted after the removal of Idle characters as part of the data rate adaptation sub-process.

EDITORS NOTE (to be remove prior to publication): Note that the list of counters will be updated per technical decision #45 (http://www.ieee802.org/3/bn/public/decisions/decisions.html) once EPoC-specific FEC and PMD overhead details are settled.

101.3.2.1.4 Functions

T_TYPE(tx_raw<71:0>)

This function is defined in 49.2.13.2.3.

EDITORS NOTE (to be remove prior to publication): Note that the list of functions will be updated per technical decision #45 (http://www.ieee802.org/3/bn/public/decisions/decisions.html) once EPoC-specific FEC and PMD overhead details are settled.

101.3.2.1.5 State diagrams

The CLT PCS shall perform the Idle deletion process as shown in Figure 101–2. The CNU PCS shall perform the Idle deletion process as shown in Figure 101–3 (data rate adaptation sub-process) Figure 101–3 and in (FEC overhead compensation sub-process), in the order shown in Figure 101–4. In case of any discrepancy between state diagrams and the descriptive text, the state diagrams prevail.

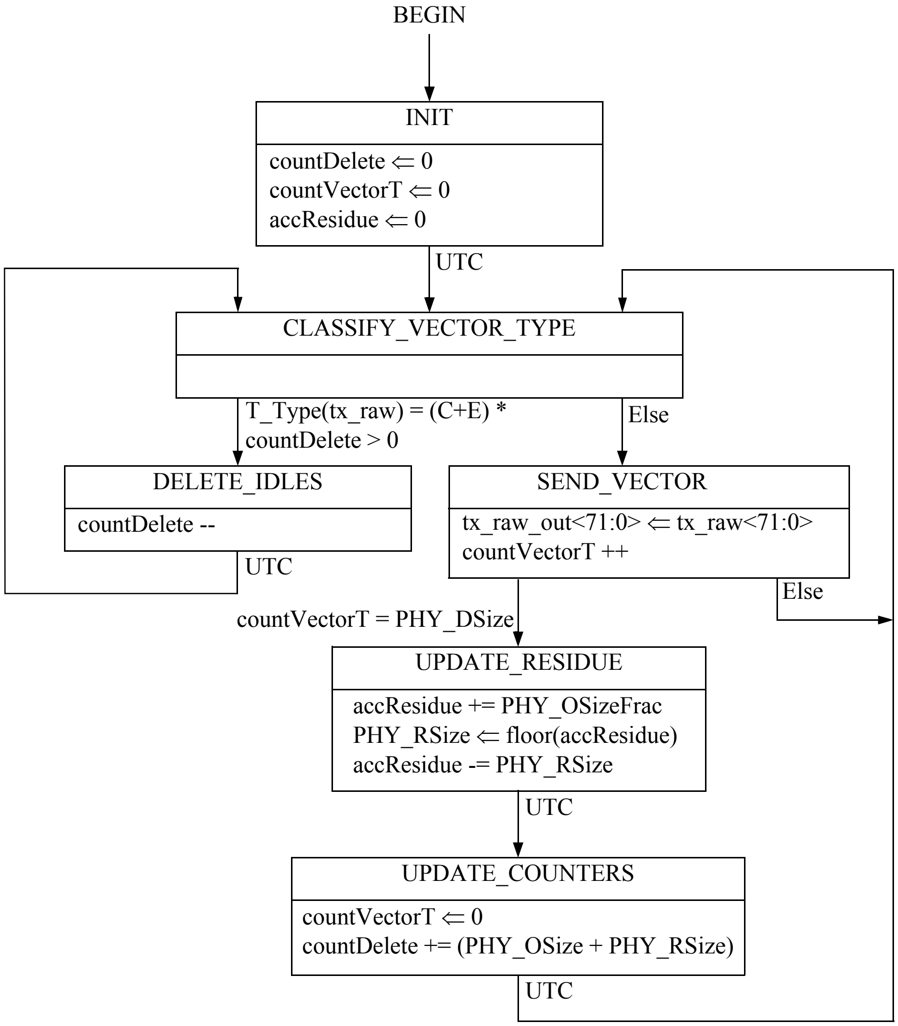


Figure 101–2—CLT Idle deletion process

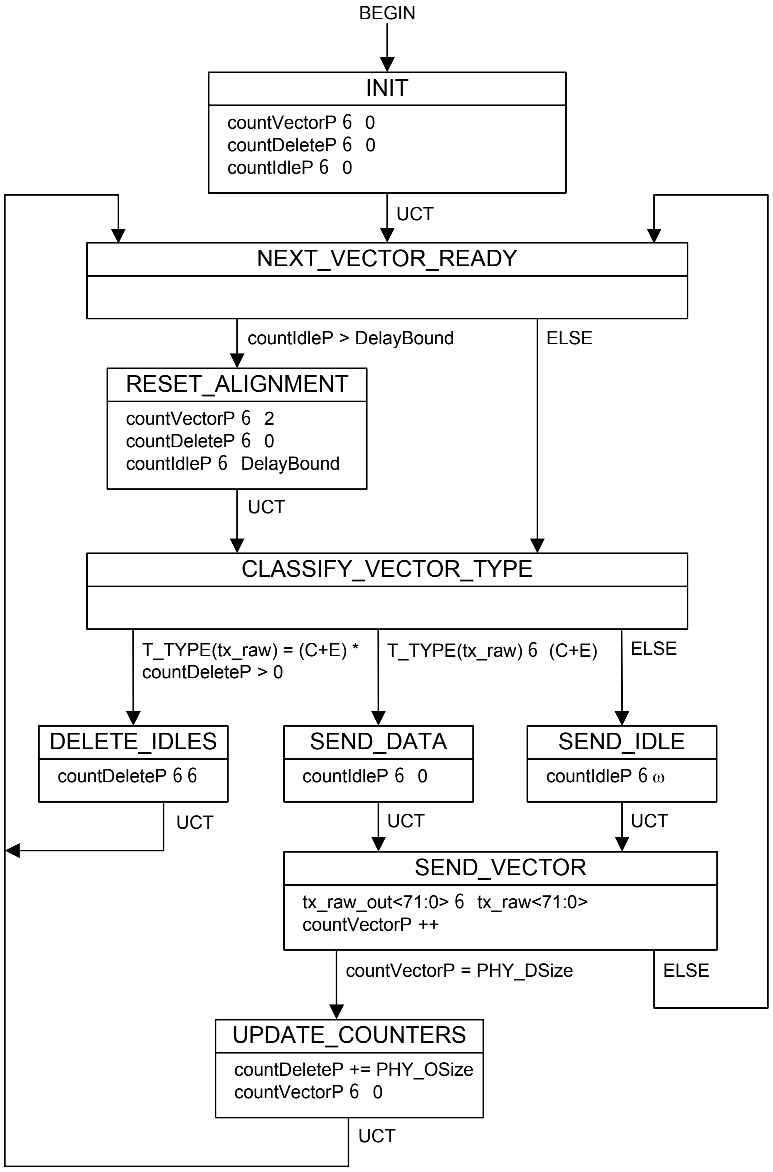


Figure 101–3—CNU Idle deletion process (data rate adaptation sub-process)

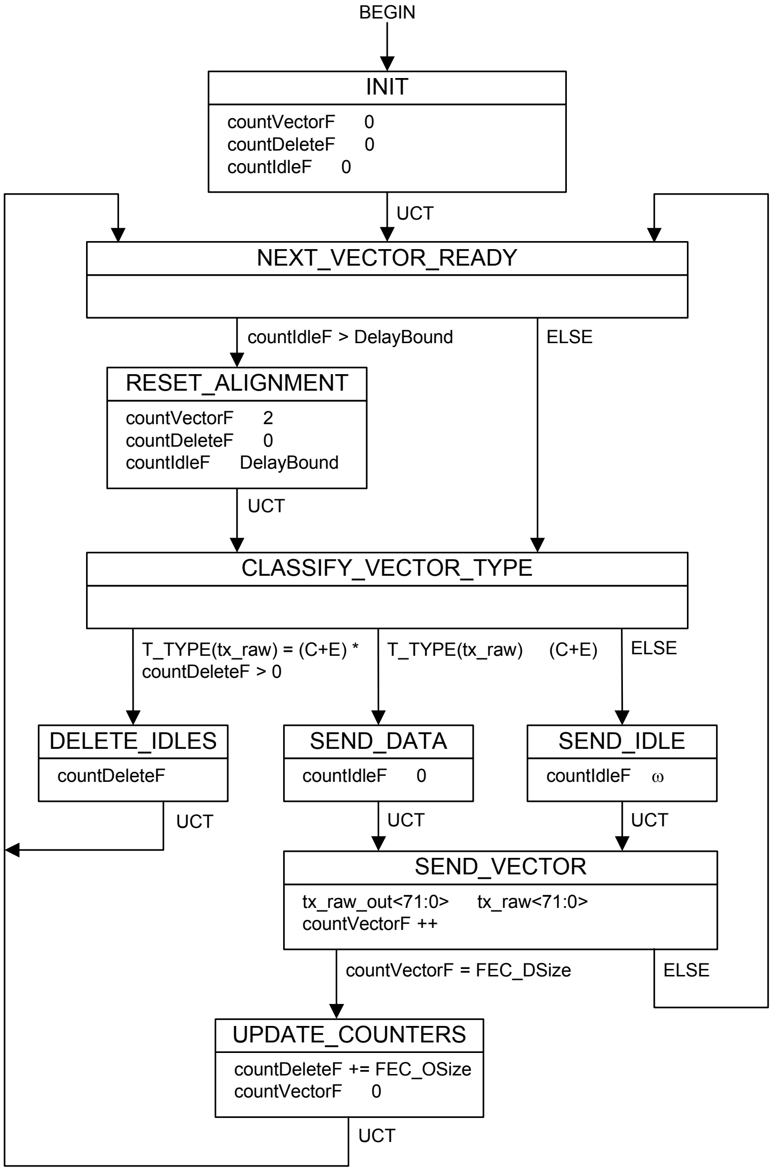


**Figure 101–4—CNU Idle deletion process
(FEC overhead compensation sub-process)**

EDITORS NOTE (to be removed prior to publication): Figure 101–2, Figure 101–3, Figure 101–3, and Figure 101–4 will be updated per technical decision #45 (http://www.ieee802.org/3/bn/public/decisions/decisions.html) once EPoC-specific FEC and PMD overhead details are settled, as well as the burst structure is defined.

101.3.2.2 64B/66B encode

The EPoC PHY utilizes a 64B/66B encoder based on that described in 49.2.5 with several important differences. The EPoC 64B/66B encoder does not include a scrambler function as described in 49.2.6 and the input is a 65B block with a single synch header bit. The state diagram found in Figure 49-12 is followed after which the sync header bit <0> is removed as illustrated in Figure 101–6.

101.3.2.3 CRC40

The CRC40 field contains a 40-bit cyclic redundancy check value. This value is computed as a function of the contents of the B_Q 65-bit blocks (see Table 101–2), forming the payload portion of the FEC codeword.

The encoding is defined by the CRC40 generating polynomial shown in Equation (101–3):

$$x^{40} + x^{26} + x^{23} + x^{17} + x^{3} + 1 \tag{101–3}$$

This CRC40 calculation shall produce the same result as the serial implementation shown in Figure 101–5. At the beginning of each FEC codeword (before the calculation of CRC40 begins), the shift register shall be initialized to the value 0x00. The content of the shift register is transmitted without inversion.

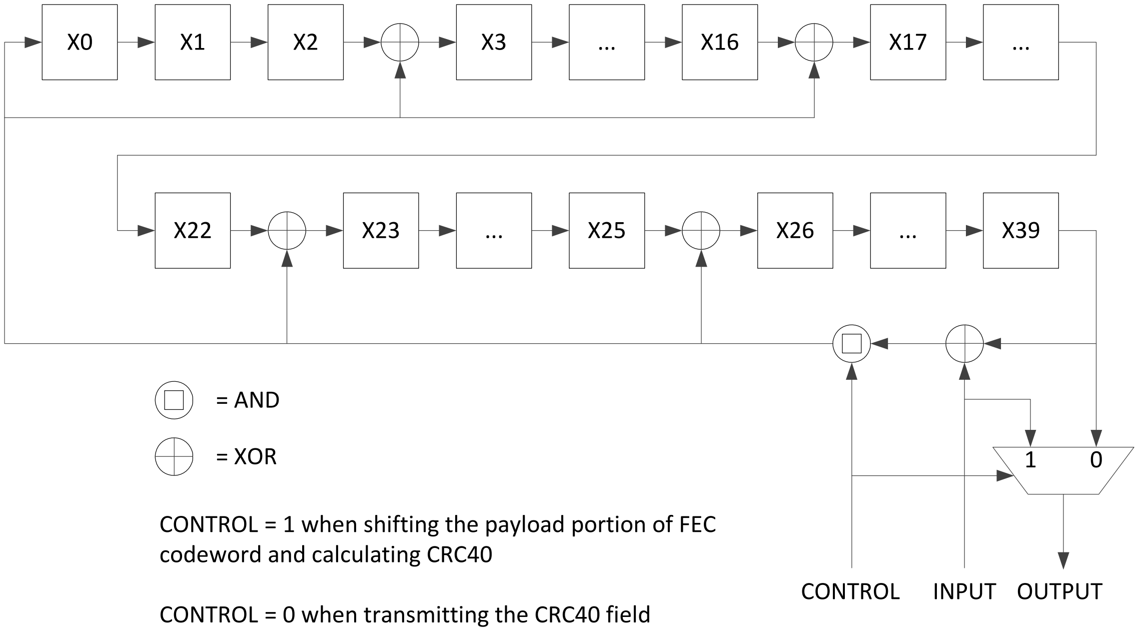

X0
X1
X2
X3
...
X16
X17
...
X22
X23
...
X25
X26
...
X39
= AND
= XOR
1
0
CONTROL = 1 when shifting the payload portion of FEC codeword and calculating CRC40
CONTROL = 0 when transmitting the CRC40 field
CONTROL
INPUT
OUTPUT

Figure 101–5—CRC40 generation

101.3.2.4 Low Density Parity Check (LDPC) Forward Error Correction (FEC) codes

The 10GPASS-XR encodes the transmitted data using a systematic Low-Density Parity-Check (LDPC) (F_C, F_P) code. A LDPC encoder encodes F_P information bits $i_0 \ldots i_{F_P-1}$ into a codeword

$$Fc = (i_0, \ldots, i_{F_P-1}, p_{F_P}, \ldots, p_{F_C-1}) \tag{101–4}$$

by adding F_R parity bits $p_{F_p} \ldots p_{F_C-1}$ obtained so that

$$Hc^T = 0$$

where H is an F_R × F_C binary matrix containing mostly ‘0’ and relatively few ‘1’, called low-density parity-check matrix (see [B32]). The detailed description of such parity check matrices is given in 101.3.2.4.1.

The CLT 10GPASS-XR PCS operating on CCDN shall encode the transmitted data using one of the LDPC (16200, 14400) code per Table 101–2. The CNU 10GPASS-XR PCS operating on CCDN shall encode the transmitted data using one of the LDPC (F_C, F_P) codes per Table 101–2.

Table 101–2—LDCP codes used in EPoC

Upstream (US) Downstream (DS)	Codeword F_C [bits]	Payload F_P [bits]	Parity F_R [bits]	US Filling Threshold F_T		Payload		
				bits	65-bit blocks	65-bit blocks B_Q	CRC bits	Padding bits B_P
US/DS	16200	14400	1800	6601	101	220	40	60
US	5940	5040	900	1601	25	76	40	60
US	1120	840	280	1	1	12	40	20

101.3.2.4.1 LDPC matrix definition

The low-density parity check matrix H for LDPC (F_C, F_P) encoder can be divided into blocks of L^2 sub-matrices. Its compact circulant form is represented by an m × n block matrix:

$$H = \begin{bmatrix} H_{1,1} & H_{1,2} & H_{1,3} & \cdots & H_{1,n} \\ H_{2,1} & H_{2,2} & H_{2,3} & \cdots & H_{2,n} \\ H_{3,1} & H_{3,2} & H_{3,3} & \cdots & H_{3,n} \\ \cdots & \cdots & \cdots & & \cdots \\ H_{m,1} & H_{m,2} & H_{m,3} & \cdots & H_{m,n} \end{bmatrix}$$

where the submatrix $H_{i,j}$ is an L × L all-zero submatrix or a cyclic right-shifted identity submatrix. The last n–m sub-matrix columns represent the parity portion of the matrix. Moreover, nL = F_C, mL = F_P and the code rate is (n–m)/n = (F_C–F_P)/F_C. The sub-matrix size L is called the lifting factor.

The sub-matrix $H_{i,j}$ is represented by a value in {-1, 0,…, L-1}, where a '-1' value represents an all-zero sub-matrix, and the remaining values represent an L × L identity submatrix cyclically right-shifted by the specified value. Such representation of the parity-check matrix is called a base matrix.

Table 101–3 presents a 5 × 45 base matrix of the low-density parity-check matrix H for
LDPC (16200, 14400) code listed in Table 101–2 for downstream and upstream. The lifting factor of the matrix is L=360.

Table 101–4 presents a 5 × 33 base matrix of the low-density parity-check matrix H for LDPC (5940, 5040) code listed in Table 101–2 for upstream. The lifting factor of the matrix is L=180.

Table 101–3—LDPC (16200, 14400) code matrix

Columns	Rows						Columns	Rows				
	1	2	3	4	5			1	2	3	4	5
1	93	274	134	–1	253		**24**	47	1	345	359	–
2	271	115	355	–	273		**25**	76	159	174	342	–1
3	–1	329	175	184	90		**26**	73	56	269	–1	232
4	83	338	24	70	–1		**27**	150	72	329	224	–1
5	26	124	253	247	–1		**28**	349	126	–1	106	21
6	208	–1	242	14	151		**29**	139	277	214	–1	331
7	245	293	–1	22	311		**30**	331	156	–1	273	313
8	200	–1	187	7	320		**31**	118	32	–1	177	349
9	–1	69	94	285	339		**32**	345	111	–1	245	34
10	175	64	26	54	–1		**33**	27	175	–1	98	97
11	331	342	87	–1	295		**34**	294	–1	218	355	187
12	17	–1	302	352	148		**35**	–1	306	104	178	38
13	86	88	–1	26	48		**36**	145	224	40	176	–1
14	–1	139	191	108	91		**37**	279	–1	197	147	235
15	337	–1	323	10	62		**38**	97	206	73	–1	52
16	–1	137	22	298	100		**39**	106	–1	229	280	170
17	238	212	–1	123	232		**40**	160	29	63	–1	58
18	81	–1	245	139	146		**41**	143	106	–1	–1	–1
19	–1	157	294	117	200		**42**	–1	334	270	–1	–1
20	307	195	240	–1	135		**43**	–1	–1	72	221	–1
21	–1	357	84	336	12		**44**	–1	–1	–1	208	257
22	165	81	76	49	–1		**45**	–1	–1	–1	–1	0
23	–1	194	342	202	179							

Table 101–5 presents a 5 × 20 base matrix of the low-density parity-check matrix H for LDPC (1120, 840) code listed in Table 101–2 for upstream. The lifting factor of the matrix is L=56.

Table 101–4—LDCP (5940, 5040) code matrix

Columns	**Rows**						**Columns**	**Rows**				
	1	**2**	**3**	**4**	**5**			**1**	**2**	**3**	**4**	**5**
1	142	54	63	28	52		**18**	84	171	50	–1	168
2	158	172	11	160	159		**19**	–1	65	46	67	158
3	113	145	112	102	75		**20**	64	141	17	82	–1
4	124	28	114	44	74		**21**	66	–1	175	4	1
5	92	55	61	8	46		**22**	97	42	–1	177	49
6	44	19	123	84	71		**23**	1	83	–1	151	89
7	93	159	72	126	42		**24**	115	7	–1	131	63
8	70	22	55	9	11		**25**	8	–1	92	139	179
9	172	96	114	169	108		**26**	108	39	–1	117	10
10	3	12	20	174	153		**27**	–1	121	41	36	75
11	25	85	53	147	–1		**28**	–1	84	138	18	161
12	44	–1	114	24	72		**29**	11	101	–1	–1	–1
13	141	128	42	145	–1		**30**	–1	171	34	–1	–1
14	160	5	33	–1	163		**31**	–1	–1	74	23	–1
15	50	158	4	26	–1		**32**	–1	–1	–1	8	177
16	45	120	66	–1	9		**33**	–1	–1	–1	–1	19
17	118	51	163	–1	2							

Table 101–5—LDCP (1120, 840) code matrix

Columns	**Rows**						**Columns**	**Rows**				
	1	**2**	**3**	**4**	**5**			**1**	**2**	**3**	**4**	**5**
1	5	0	12	0	36		**11**	3	2	19	53	–1
2	14	35	28	51	6		**12**	–1	23	18	13	11
3	12	1	22	16	3		**13**	34	0	52	–1	22
4	1	26	46	31	51		**14**	7	51	–1	52	23
5	2	0	3	13	4		**15**	46	–1	37	33	43
6	37	10	16	39	19		**16**	10	49	–1	–1	–1
7	45	16	51	27	4		**17**	–1	20	34	–1	–1
8	26	16	2	33	45		**18**	–1	–1	39	38	–1
9	24	34	25	8	48		**19**	–1	–1	–1	7	14
10	0	4	29	27	9		**20**	–1	–1	–1	–1	1

101.3.2.5 FEC Encoder and Data Detector processes

101.3.2.5.1 Data Detector process

The 10GPASS-XR PCS transmit path includes the Data Detector process. This process contains a delay line (represented by the *FIFO_FEC_TX* buffer) that stores 65-bit blocks received from the output of the 64B/66B Encoder to allow insertion of the FEC parity data into the transmitted data stream. The length of the *FIFO_FEC_TX* buffer is selected in such a way that it is large enough to compensate for the insertion of the FEC parity data as well as any additional FEC-related overhead, as defined in 101.3.2.5.2.

For the Data Detector process included in the 10GPASS-XR CLT PCS transmit path, the length of the *FIFO_FEC_TX* buffer is set to be the ceiling of the sum of the long codeword parity size + CRC40 size divided by 65.

$$29 = \lceil (1800 + 40) / 65 \rceil$$

101.3.2.5.2 LDPC encode process

The process of padding FEC codewords and appending FEC parity octets in the 10GPASS-XR PCS transmit path is illustrated in Figure 101–6. For the CLT, the FEC Encoder uses a single FEC LDPC codeword size of 16200 indicated by “DS” in Table 101–2 represented by constant FEC_DS_CodeWordSize (see 101.3.2.5.2). For the CNU, the FEC Encoder performs an algorithm as described in 101.3.2.5.4 for selecting the use of one or more “long” codewords (US size 16200), a “medium” codeword (US size 5940) or one or more “short” codewords (US size 1120) as detailed in Table 101–2

The 64B/66B Encoder produces a stream of 66-bit blocks as shown in Figure 101–6 (see 49.2.4.3 for more details); each 66-bit block is composed of 2 bits of sync header and 64 bits of data. These 66-bit blocks are converted to 65-bit blocks by removing the redundant first bit (i.e., sync header bit <0>) in each 66-bit block received from the 64B/66B Encoder and are then delivered to the FEC Encoder and Data Detector input process. In the CLT only, a 65-bit burst time header is placed (accumulated) as the first 65-bit block at the start of a burst. The FEC Encoder accumulates B_Q (see Table 101–2) of these 65-bit blocks to form the payload portion of the FEC codeword.

Next, the FEC Encoder calculates CRC40 (see 101.3.3) over the aggregated B_Q (see Table 101–2) 65-bit blocks, placing the resulting 40 bits of CRC40 code immediately after the B_Q 65-bit blocks, forming the payload portion of the FEC codeword. Finally, based on the codeword size the FEC Encoder appends B_P (see Table 101–2) padding bits (with the binary value of “0”) to the payload of the FEC codeword as shown in Figure 101–6.

The resulting FP bits of data are then passed to the LDPC-encoder specifying a payload length of F_P - B_P bits (14400 - 60 = 14340 bits). The LDPC-encoder generates F_R bits of parity. After encoding, the encoder deletes the B_P bits of padding and constructs the output codeword with a length of (F_P - B_P) + F_R bits; i.e., (14400 - 60) + 1800 = 16140 bits.

101.3.2.5.3 LDPC codeword transmission order

Once the process of calculating FEC parity is complete, the payload portion of the FEC codeword and the parity portion of the FEC codeword are then transferred to the transmit PMA Client.

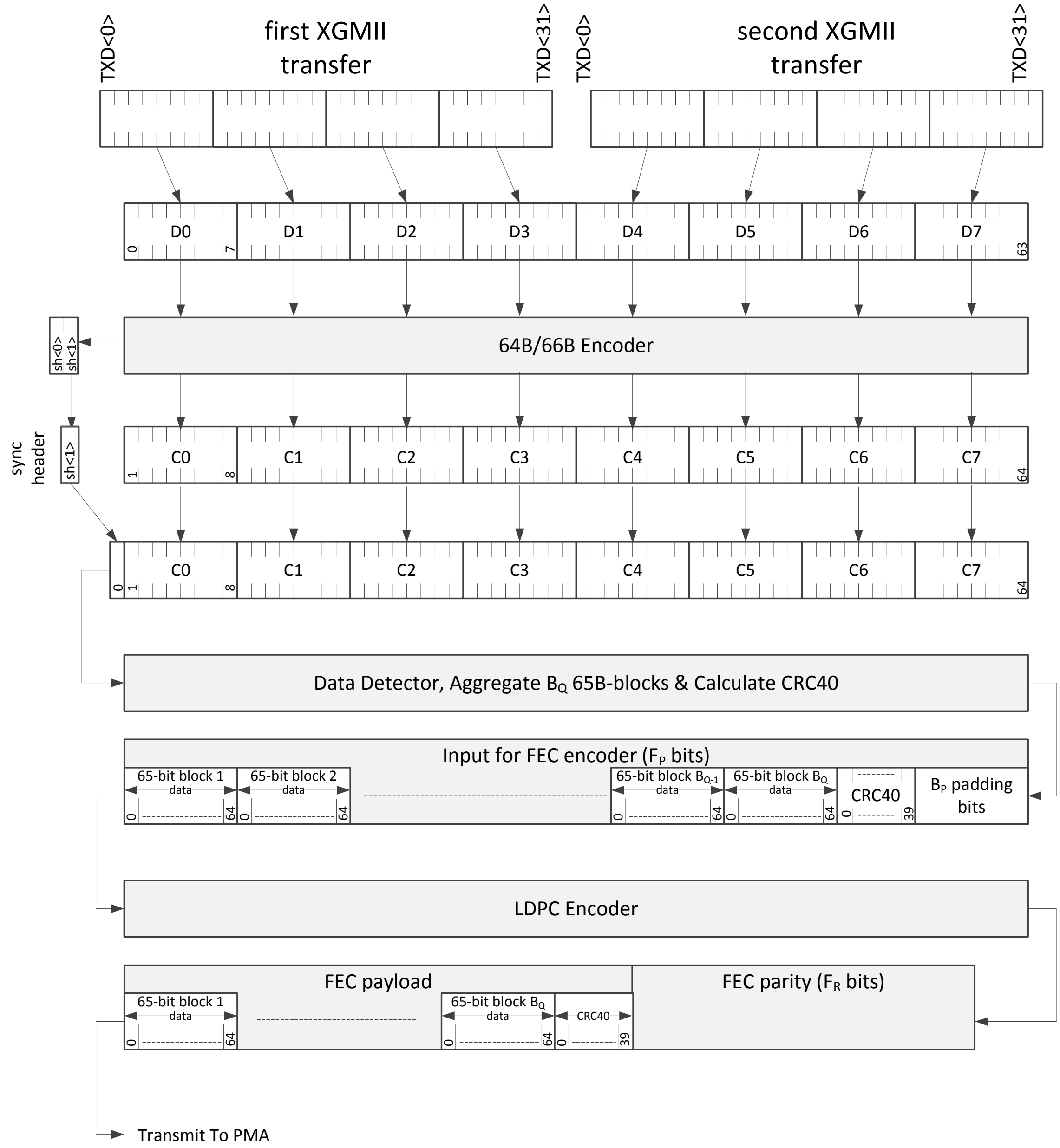
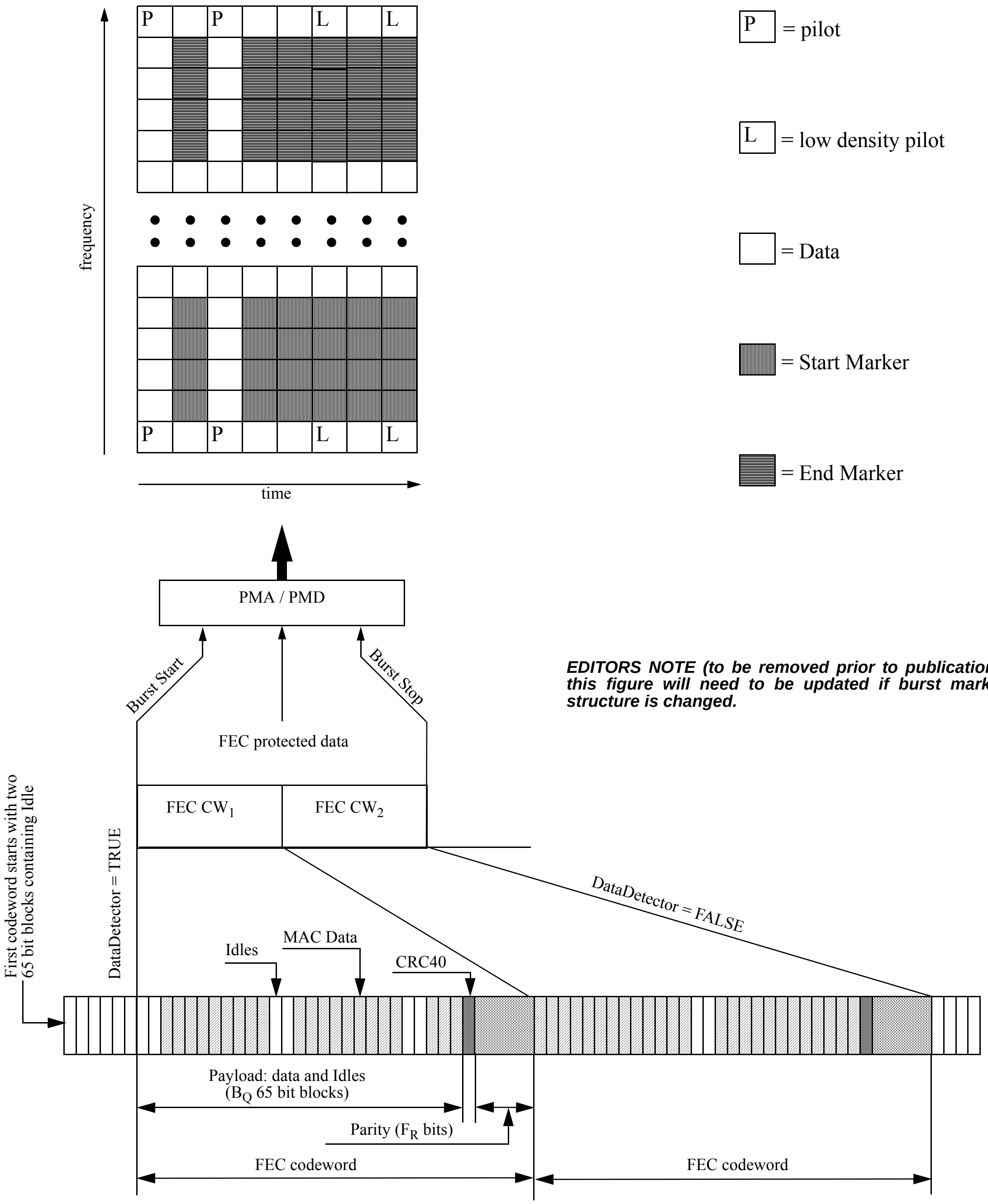


Figure 101–6—10GPASS-XR PCS transmit path processing

Figure 101–7 illustrates the details of the 10GPASS-XR CNU burst structure. In particular, this figure shows the details of the necessary burst elements and the FEC protected portions of the burst transmission, explicitly showing each FEC codeword (FEC CW).



EDITORS NOTE (to be removed prior to publication): this figure will need to be updated if burst marker structure is changed.

Figure 101–7—Details of burst composition

The CNU burst transmission begins with a starting burst marker (see 101.4.3.9), which facilitates the detection of the start of an incoming data burst. When received at the CLT, the burst marker enables FEC codeword alignment to the incoming data stream, even in the presence of bit errors. The burst marker is not part of the first FEC codeword.

The CNU burst ends with the ending burst marker (see 101.4.3.9), which facilitates the detection of the end of the current data burst. When received at the CLT, the ending burst marker allows for the rapid reset of the

CLT FEC synchronizer, so that it can search for the next burst. The ending burst marker is not part of the last FEC codeword.

101.3.2.5.4 Upstream FEC encoding

Upstream bursts in EPoC are variable in length and may contain more than one MAC frame. The upstream makes use of three different LDPC codeword sizes as described Table 101–4. Each codeword size has an associate US Filling Threshold F_T with a specific threshold for each codeword size.

Starting with a 16200 (long) codeword:

1) If there are enough 65-bit blocks B to create and encode a full long codeword (B_Q = 220 for long). Repeat create and encode using long codewords if B ≥ B_Q = 220 blocks are available.
2) If remaining B blocks in burst < B_Q = 220 blocks and ≥ 101 blocks, use a long codeword, shorten to remaining blocks and end the burst with this encoded codeword.
3) If remaining B blocks are < 101 blocks and ≥ 76 blocks, create and encode a medium codeword.
4) If remaining B blocks in burst < B_Q = 76 blocks and ≥ 25 blocks, use a medium codeword, shorten to remaining blocks and end the burst with this encoded codeword.
5) If remaining B blocks are < 25 blocks and ≥ 12 blocks, create and encode a short codeword. Repeat create and encode using short codewords if B ≥ B_Q = 12 blocks are available.
6) If remaining B blocks in burst < B_Q = 12 blocks and ≥ 1 block, use a short codeword, shorten to remaining blocks and end the burst with this encoded codeword.

Every codeword in the burst has a length of determined by the number B of 65-bit blocks encoded:

$$CodewordLength = B \times 65(bits(data)) + 40(bits(CRC)) + F_R(bits(Parity))$$

where:

$1 \le B \le B_Q$ and
B_Q and F_R are set per Table 101–2 based on F_C.

can be from 1 to B_Q blocks maximum, where B_Q is 220, 76, and 12 and F_R is 1800, 900, and 280 for 16200, 5940, 1120 LDPC codewords sizes, respectively (see Table 101–2).

All codeword encoding follows the same procedures as the downstream with the following differences:

- The appropriate values from Table 101–2 are used for the corresponding codeword size being encoded.
- The burstStart indication in the PMA_UNITDATA.request() corresponds to the first bit in the concatenated burst.
- The burstEnd indication in the PMA_UNITDATA.request() corresponds to the last bit of the concatenated burst.
- The nominal data rate of the PMA_UNITDATA_request() is *US_DataRate*, as configured via PHY Link management.

101.3.2.5.5 Constants

FEC_DS_CodeWordSize
Type: 16-bit integer
The fixed size in bits of the downstream FEC LDPC output codeword.
Value: 16140

SH_CTRL

See 76.3.2.5.2

SH_DATA

See 76.3.2.5.2

101.3.2.5.6 Variables

blockCount

TYPE: 16-bit unsigned integer
This variable represents the number of either 65-bit blocks or 66-bit blocks.

B_P

TYPE: 16-bit unsigned integer
VALUE: see Table 101–2
This variable represents the number of padding bits within the payload portion of the downstream FEC codeword.

B_Q

TYPE: 16-bit unsigned integer
VALUE: see Table 101–2
This variable represents the number of 65-bit blocks within the payload portion of the downstream FEC codeword.

burstEnd

TYPE: boolean
When TRUE this flag signals the end of burst.

burstSize

TYPE: integer
This variable is set as input to the transferToPMA process and indicates the number of bits in the ARRAY_IN.

burstStart

TYPE: boolean
When TRUE this flag signals the start of a burst.

CLK

TYPE: boolean
This boolean is TRUE on every negative edge of TX_CLK (see 46.3.1) and represents instances of time at which a 66-bit block is passed from the output of the 64B/66B Encoder into the FEC Encoder. This variable is reset to FALSE upon read.

dataPayload<F_P-1:0>

TYPE: Bit array
This array represents the payload portion of the FEC codeword, accounting for the necessary padding. It is initialized to the size of F_P bits and filled with the binary value of “0”.

dataParity<F_R-1+C_P:0>

TYPE: Bit array
This array represents the parity portion of the FEC codeword, accounting for the necessary padding. It is initialized to the size of F_R bits and filled with the binary value of “0”.

DelayBound

See 101.3.2.1.2

DS_DataRate

See 100.2.6.1.

FIFO_FEC_TX

TYPE: Array of 65-bit blocks

A FIFO array used to store 65-bit blocks, inserted by the input process and retrieved by the output process in the FEC Encoder.

firstcodeword

TYPE: boolean

When TRUE this flag signals the first codeword of a burst.

F_P

TYPE: 16-bit unsigned integer

VALUE: see Table 101–2

This variable represents the number of bits within the payload portion of the FEC codeword.

F_R

TYPE: 16-bit unsigned integer

VALUE: see Table 101–2

This variable represents the number of bits within the parity portion of the FEC codeword.

IdleBlockCount

TYPE: 32 bit unsigned

The number of consecutive non-data 65-bit blocks ending with the most recently received block. The non-data 65-bit blocks are represented by sync header 10 (binary).

lastcodeword

TYPE: boolean

When TRUE this flag signals the last codeword of a burst.

loc

TYPE: 16-bit unsigned integer

This variable represents the position within the given bit array.

PMA_CLK

TYPE: boolean

This Boolean is TRUE on every negative edge of a clock that is synchronized to the PMA_UNITDATA.request (see 101.4.1.2.1) data rate of *DS_DataRate* (see 100.2.6.1).

Short2Payload<FC-1:0>

TYPE: bit array

This bit array is used as a temporary buffer for the second short codeword in the Check_data-payload() function.

Short2blockCount

TYPE: 16-bit unsigned integer

This variable is used as a temporary to hold the *blockCount* of *Short2Payload<>*.

sizeFifo

TYPE: 16-bit unsigned integer

This variable represents the number of 65-bit blocks stored in the FIFO.

Transmitting

TYPE: boolean

When TRUE this boolean variable indicates the device is transmitting. At the CNU, the default value of *Transmitting* is false. At the CLT, this variable is always set to TRUE.

tx_coded<65:0>

TYPE: 66-bit block

This 66-bit block contains 64B/66B encoded data from the output of the 64B/66B Encoder. The format for this data block is shown in Figure 49–7. The left-most bit in the figure is *tx_coded<0>* and the right-most bit is *tx_coded<65>*.

tx_coded_out<F_C-1:0>

TYPE: bit array

This bit array contains the output of the FEC Encoder being passed to the Data Detector. The left-most bit is tx_coded_out<0> and the right-most bit is *tx_coded_out<FC-1>*.

txCount

TYPE: 16-bit unsigned integer

This variable used for counting bits in the Transfer to PMA process.

US_DataRate

See 100.2.6.2.

101.3.2.5.7 Functions

BurstTimeHeader()

This function returns a 65-bit value that is used as the first 65-bit block of an upstream burst. Value: binary 1 followed by the 32-bit PHY Link timestamp value at the time of the call to this function followed by 0x D8 58 E4 AB.

Calculate_CRC40_and_3Parity(paritySize)

This function takes an argument of “LONG”, “MEDIUM”, or “SHORT” and then processes *dataPayload*<> and *tx_coded_out*<> to add CRC40 and appropriate LDPC parity. The *tx_coded_out*<> array is then passed to transferToPMA() with indicated length and *lastcodeword* status.

```
// Calculate_CRC40_and_Parity. Computes CRC40 then encodes
// based on indicated codeword size

Function Calculate_CRC40_and_3Parity( paritySize ) {
    Global: loc, blockCount, dataPayload, firstcodeword,
    lastcodeword;

    IF (paritySize = LONG)
        parityLength = 1800;
    IF (paritySize = MEDIUM)
        parityLength = 900;
    IF (paritySize = SHORT)
        parityLength = 280;

    dataPayload<loc+39:loc> = calculateCrc(dataPayload<loc-1:0>);
    tx_coded_out<loc+39:loc> = dataPayload<loc+39:loc>;
    loc += 40;
    dataParity<parityLength-1:0> = calculateParity(dataPayload<loc-1:0>, loc,
paritySize);
    tx_coded_out<loc+parityLength-1:loc> = dataParity<parityLength-1:0>;
    loc += parityLength;
    transferToPMA(tx_coded_out, loc, lastcodeword);

    // Setup for next codeword in burst if more data
    firstcodeword = FALSE;
    loc = 0;
    block_count = 0;
    resetArray(dataPayload );
    resetArray(dataParity );
    return();
}
```

calculateCrc (ARRAY_IN)

This function calculates a CRC40 value for data included in ARRAY_IN.

calculateParity(ARRAY_IN , Length, paritySize)

This function calculates the LDPC parity (for the code per Table 101–2) for data included in ARRAY_IN up to the specified Length (bits). For any Length < F_P, any remaining bits in ARRAY_IN after Length are considered padding bits (of length B_P bits) and will not be included in the encoder output producing a shortened codeword; i.e., codeword length will be F_P - B_P + F_R. When paritySize is “LONG”, the values for the DS/US codeword size of 16200 are used, for “MEDIUM”, the values for the US codeword size of 5940 are used, and for”SHORT” the values for the US codeword size of 1120 are used.

Check_dataPayload(firstcodeword, lastcodeword, lastcodeword)

This function performs the upstream codeword filling algorithm examining accumulated *blockCount* as per 101.3.2.5.4. This function calls Calculate_CRC40_and_3Parity() indicating the appropriate codeword size to use, as appropriate.

```
// Check_dataPayload() implements the Upstream FEC encoding
Function Check_dataPayload( firstcodeword, lastcodeword )
{
    Global: loc, blockCount, dataPayload, tx_coded_out,
    firstcodeword, lastcodeword;

    IF (lastblock = FALSE) {
        IF (blockCount = 220 )
            Calculate_CRC40_and_3Parity(LONG);
    } ELSE {
        IF (blockCount < 200 AND blockCount >= 101)
            Calculate_CRC40_and_3Parity(LONG);
        IF (blockCount < 101 AND blockCount >= 76)
            Calculate_CRC40_and_3Parity(MEDIUM);
        IF (blockCount < 76 AND blockCount >= 25)
            Calculate_CRC40_and_3Parity(MEDIUM);
        IF (blockCount < 24 AND blockCount > 12) {
            // dataPayload<> contains two short codewords
            // first is 12 blocks, second is remainder
            Short2blockCount = blockCount - 12;
            Short2Payload<Short2loc:0> = dataPayload<loc:12*65>;
            loc = (12*65) - 1;
            blockCount = 12;
            Calculate_CRC40_and_3Parity(SHORT);
            dataPayload<Short2loc:0> = Short2Payload<Short2loc:0>;
            tx_coded_out<Short2loc:0> = Short2Payload<Short2loc:0>;
            loc = (Short2blockCount * 65) - 1;
            blockCount = Short2blockCount;
            Calculate_CRC40_and_3Parity(SHORT)
        } ELSE IF ( blockCount >= 1 ) {
            Calculate_CRC40_and_3Parity(SHORT);
        }
    return();
};
```

resetArray(ARRAY_IN)

This function resets the content of ARRAY_IN, removing all the elements within ARRAY_IN and setting its size to 0.

removeFifoHead(ARRAY_IN)

This function removes the first block in ARRAY_IN and decrements its size by 1.

```
removeFifoHead( ARRAY_IN )
{
   ARRAY_IN[0] = ARRAY_IN[1]
   ARRAY_IN[1] = ARRAY_IN[2]
   ...
   ARRAY_IN[sizeFifo-2] = ARRAY_IN[sizeFifo-1]
   sizeFifo --
}
```

transferToPMA(ARRAY_IN, burstSize, lastcodeword)

This function invokes the Transfer to PMA process to transfer *burstSize* bits contained in ARRAY_IN and stop and start state information to the PMA_UNITDATA.request() service primitive. If *lastcodeword* is TRUE, the *burstEnd* boolean is set TRUE on the last bit transferred to PMA_UNITDATA.request().

101.3.2.5.8 State diagrams

The CLT and the CNU shall implement the Data Detector input process as depicted in Figure 101–8. The CLT shall implement the Data Detector output process as depicted in Figure 101–9. The CNU shall implement the Data Detector output process as depicted in Figure 101–10,

In case of any discrepancy between state diagrams and the descriptive text, the state diagrams prevail.

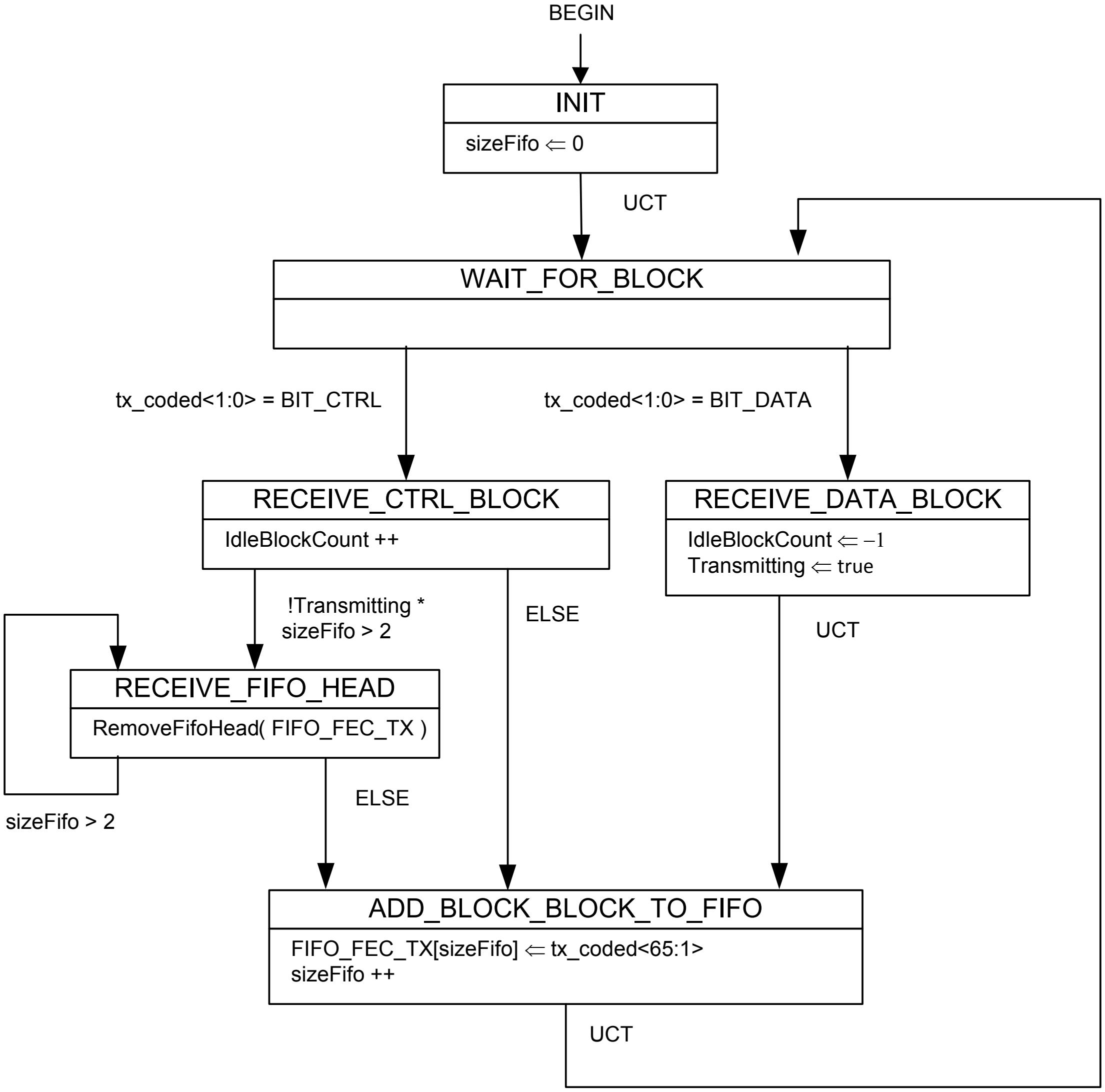


Figure 101–8— FEC Encoder and Data Detector input process state diagram

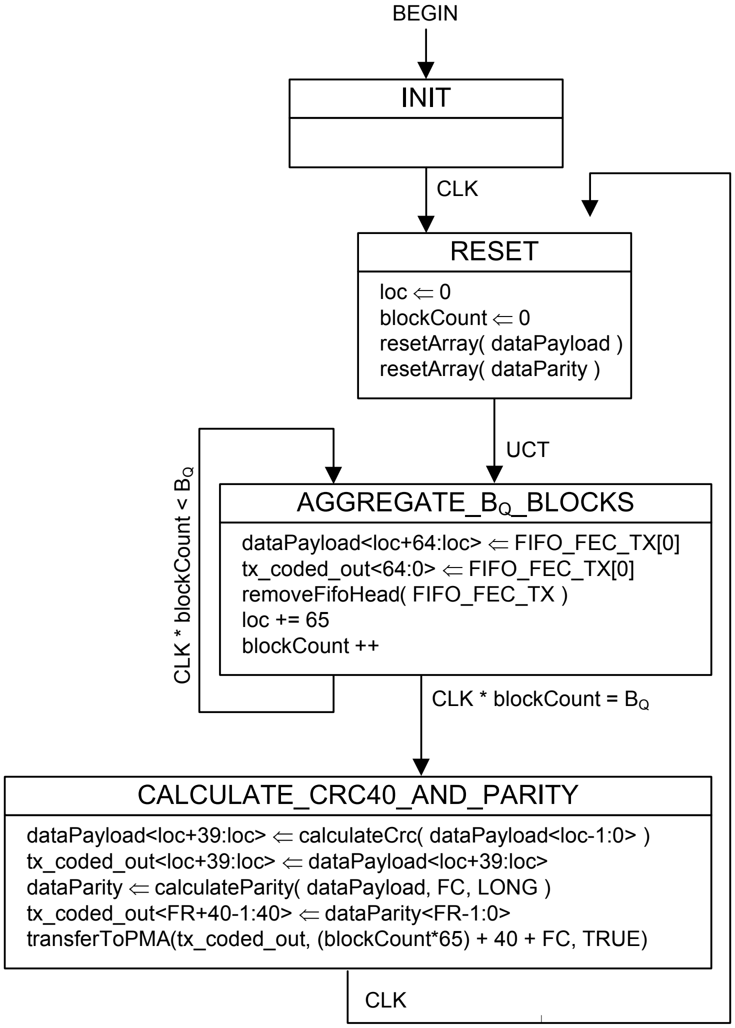

BEGIN
INIT
CLK
RESET
loc ⇐ 0
blockCount ⇐ 0
resetArray(dataPayload)
resetArray(dataParity)
UCT
CLK * blockCount < B_Q
AGGREGATE_B_Q_BLOCKS
dataPayload<loc+64:loc> ⇐ FIFO_FEC_TX[0]
tx_coded_out<64:0> ⇐ FIFO_FEC_TX[0]
removeFifoHead(FIFO_FEC_TX)
loc += 65
blockCount ++
CLK * blockCount = B_Q
CALCULATE_CRC40_AND_PARITY
dataPayload<loc+39:loc> ⇐ calculateCrc(dataPayload<loc-1:0>)
tx_coded_out<loc+39:loc> ⇐ dataPayload<loc+39:loc>
dataParity ⇐ calculateParity(dataPayload, FC, LONG)
tx_coded_out<FR+40-1:40> ⇐ dataParity<FR-1:0>
transferToPMA(tx_coded_out, (blockCount*65) + 40 + FC, TRUE)
CLK

Figure 101–9—CLT FEC Encoder and Data Detector output process

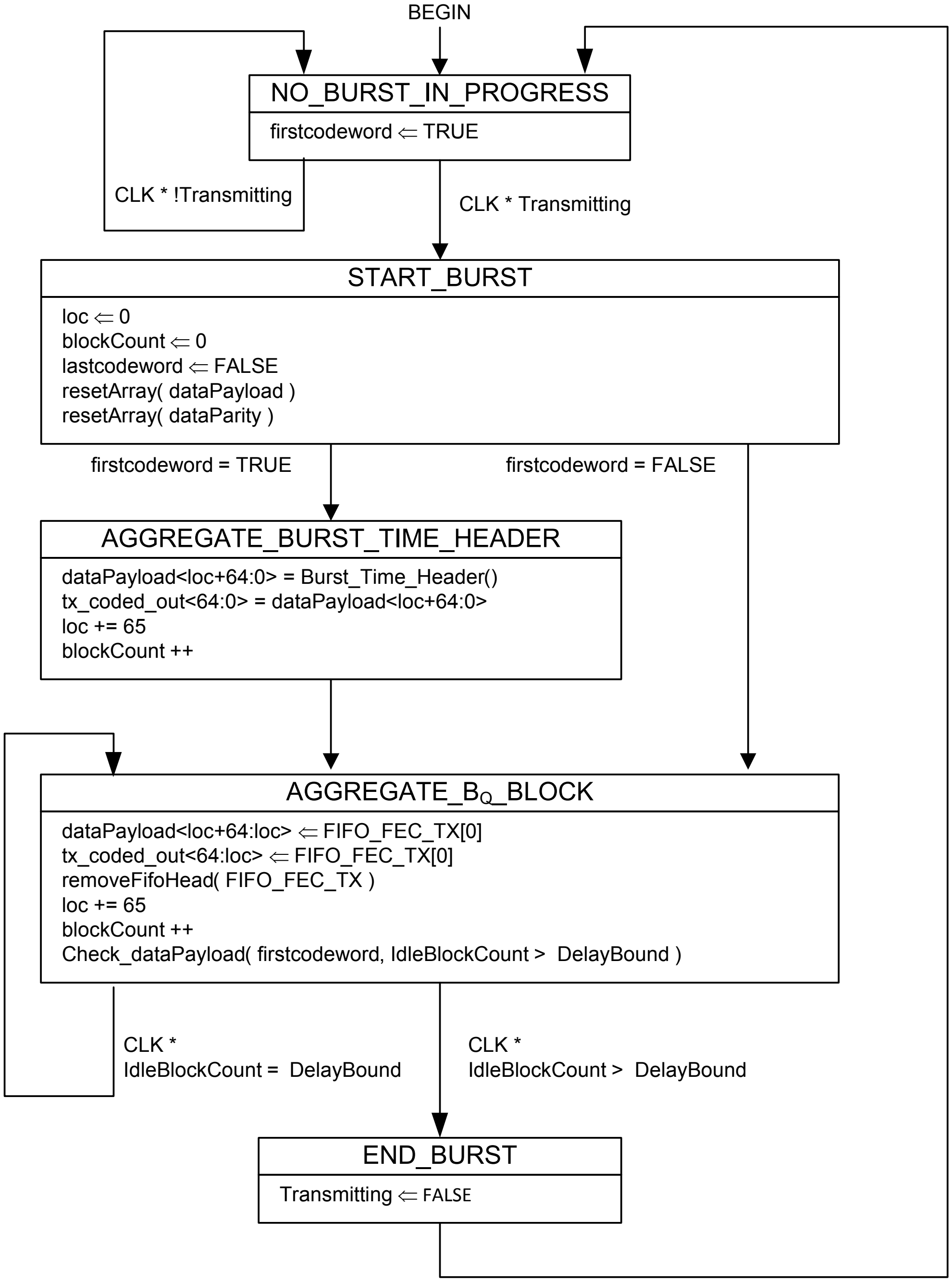

BEGIN
NO_BURST_IN_PROGRESS
firstcodeword ⇐ TRUE
CLK * !Transmitting
CLK * Transmitting
START_BURST
loc ⇐ 0
blockCount ⇐ 0
lastcodeword ⇐ FALSE
resetArray(dataPayload)
resetArray(dataParity)
firstcodeword = TRUE
firstcodeword = FALSE
AGGREGATE_BURST_TIME_HEADER
dataPayload<loc+64:0> = Burst_Time_Header()
tx_coded_out<64:0> = dataPayload<loc+64:0>
loc += 65
blockCount ++
AGGREGATE_B$_Q$_BLOCK
dataPayload<loc+64:loc> ⇐ FIFO_FEC_TX[0]
tx_coded_out<64:loc> ⇐ FIFO_FEC_TX[0]
removeFifoHead(FIFO_FEC_TX)
loc += 65
blockCount ++
Check_dataPayload(firstcodeword, IdleBlockCount > DelayBound)
CLK *
IdleBlockCount = DelayBound
CLK *
IdleBlockCount > DelayBound
END_BURST
Transmitting ⇐ FALSE

Figure 101–10—CNU FEC Encoder and Data Detector output process

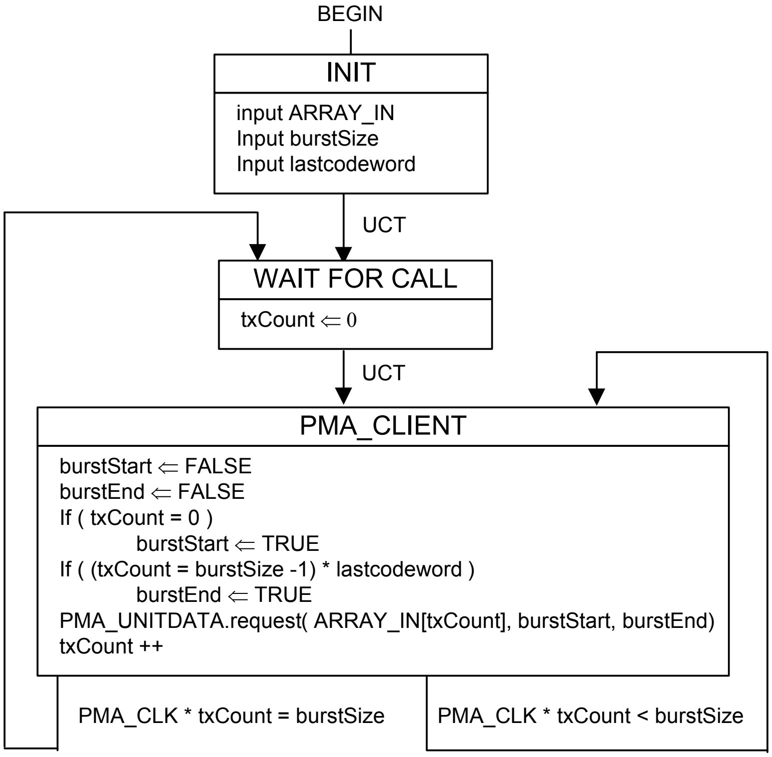


Figure 101–11—Transfer to PMA process

101.3.3 PCS receive function

In the CLT, the PCS receive function operates in a burst fashion at the data rate of up to 10 Gb/s. In the CNU, the PCS receive function operates in a continuous fashion at the data rate of up to 10 Gb/s. Receive direction functional block diagrams for the CLT and CNU are illustrated in Figure 100–4 and Figure 100–5, respectively.

In the receive direction, the EPoC PCS includes a mandatory FEC decoder, followed by a 64B/66B decoder and an Idle control character insertion function performing the function of data rate adaptation and a FEC overhead compensation.

101.3.3.1 FEC decode, respectively

The 10GPASS-XR decodes the received data using LDPC (F_C, F_P) code. The CLT 10GPASS-XR PCS operating on CCDN shall decode the received data using one of the LDPC (F_C, F_P) codes per Table 101–2. The CNU 10GPASS-XR PCS operating on CCDN shall decode the received data using LDPC (16200, 14400) code per Table 101–2.

101.3.3.1.1 Upstream FEC decoding

The CLT receiving PCS process receives an upstream burst from a CNU from the PMA Client of a length of R bits.

Start with $B_Q = 220$, CRC = 40, and $F_R = 1800$ for long codewords:

1) If $R \geq B_Q * 65 + 40 + F_R$ bits, decode a long codeword.
   Repeat and decode using long codewords if remaining bits $\geq B_Q * 65 + 40 + F_R$ bits.
   End processing burst if remaining bits ≤ 0.
2) If remaining bits $< (B_Q = 220) * 65 + 40 + (F_R = 1800)$ and $\geq (B_Q = 101) * 65 + 40 + (F_R = 1800)$, decode a shortened long codeword.
   End processing burst if remaining bits ≤ 0.
3) If remaining bits $\geq (B_Q = 76) * 65 + 40 + (F_R = 900)$, decode a medium codeword.
   End processing burst if remaining bits ≤ 0.
4) If remaining bits $< (B_Q = 76) * 65 + 40 + (F_R = 900)$ and $\geq (B_Q = 25) * 65) + 40 + (F_R = 900)$, decode a shortened medium codeword.
   End processing burst if remaining bits ≤ 0.
5) If remaining bits $\geq (B_Q = 12) * 65 + 40 + (FR = 280)$, decode a short codeword.
   Repeat and decode using short codewords if remaining bits $\geq (B_Q = 12) * 65 + 40 + (F_R = 280)$ bits.
   End processing burst if remaining bits ≤ 0.
6) If remaining bits $\geq 65 + 40 + 280$, decode a shortened short codeword.
7) If remaining bits, declare receiver burst error.

All codeword decoding follows the same procedures as the downstream with the following differences:

1) The appropriate values from Table 101–2 are used for the corresponding codeword size being decoded.
2) The burstStart indication in the PMA_UNITDATA.indication() corresponds to the first bit in the concatenated burst received from a CNU.
3) The burstEnd indication in the PMA_UNITDATA.indication() corresponds to the last bit of the concatenated burst received from a CNU.
4) The nominal data rate of the PMA_UNITDATA_request() is *US_DataRate* calculated by the CLT (see 100.2.6.2).

EDITORS NOTE (to be removed prior to publication): A figure and reference to same is needed showing FEC decoding process in CLT receiver.

101.3.3.1.2 LDPC decoding process within CNU (downstream)

The process of decoding FEC codewords in the 10GPASS-XR CNU receiver is illustrated in Figure 101–12.

EDITORS NOTE (to be removed prior to publication): FEC codeword alignment needs to be tackled somewhere between the PMA and the bottom of the PCS – we had some proposals on how to find FEC codeword lock in the downstream, but I am not sure we baselined anything with sufficient level of detail to actually put it into the draft

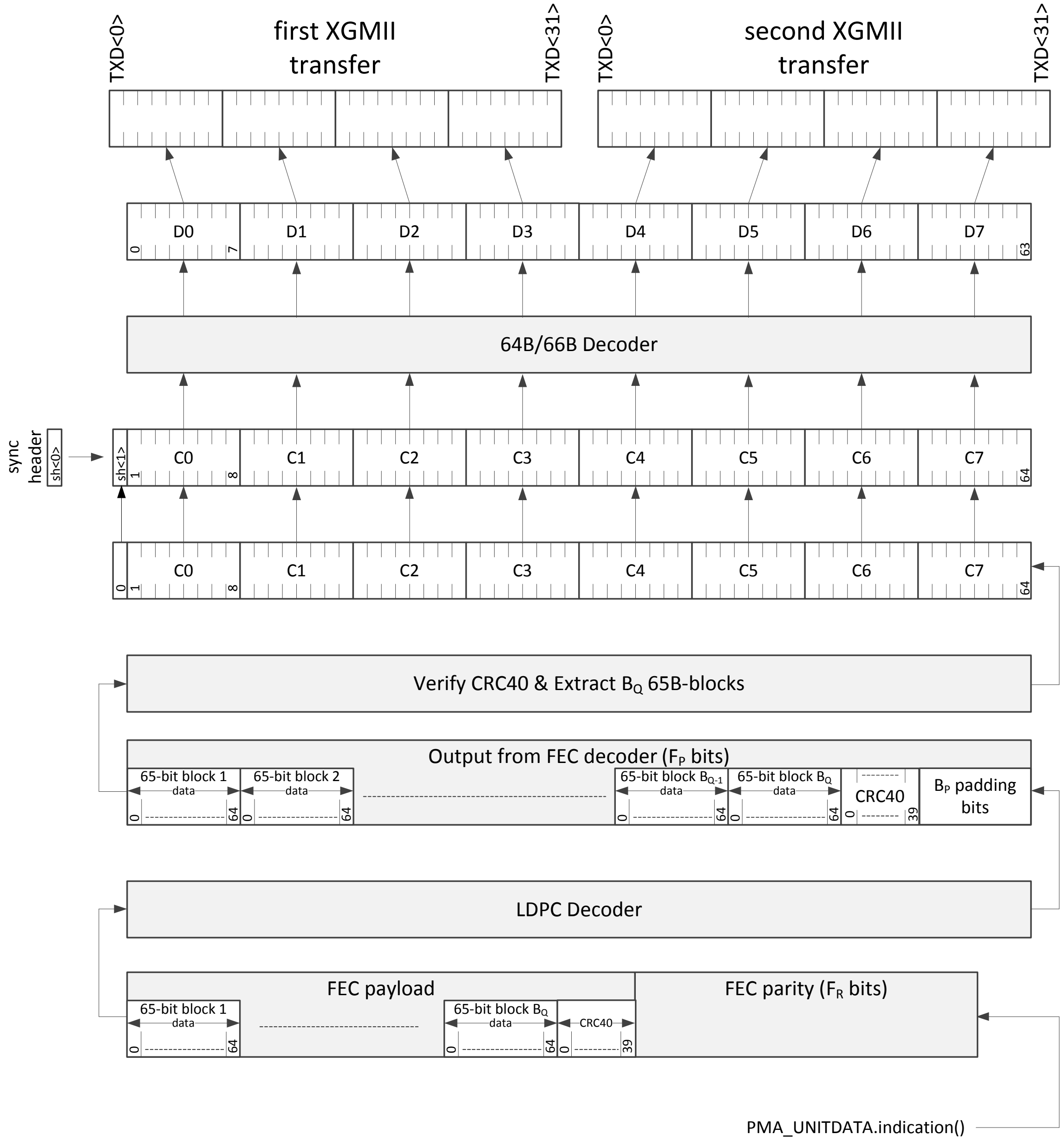


Figure 101–12—PCS Receive bit ordering within CNU (downstream)

Once the alignment to the FEC codeword is found, the 10GPASS-XR CNU receiver aggregates a total of FEC_DS_CodeWordSize bits received from the descrambler as input to the FEC decoder. The output of the FEC decoder contains the full FEC Payload consisting of 65-bit blocks number 1 to B_Q, the CRC40 data, and B_P padding bits with all bits set to the binary value of “0”.

The FEC decoder produces the FEC payload portion of the codeword with the size of F_P (in bits), where bits $<F_P-B_P-1>$ ... $<F_P-1>$ contain padding (with the binary value of “0”). Next, the CRC40 is calculated over the remaining 65-bit blocks 1 through B_Q and then compared with the value of CRC40 retrieved from the received FEC codeword. If both CRC40 codes match, the decoded FEC codeword is treated as error-free. Otherwise, the decoded FEC codeword is treated as errored.

Finally, the FEC decoder prepends each of the B_Q 65-bit blocks with bit <0> of the sync header containing the binary inverse of the value carried in bit <1> of the sync header, producing 66-bit blocks. This also guarantees that properly decoded blocks meet the requirements of 49.2.4.3.

The FEC decoder maintains error monitors to detect FEC codeword successes and failures. See 101.3.3.1.4 for details.

Each resulting 66-bit block is then fed into the 64B/66B decoder, removing the sync header information (bit <0> and bit <1>), which is used to generate control signaling for the XGMII. Finally, the resulting 64-bit block is then separated into two 32-bit portions, which are transmitted across the XGMII on two consecutive transfers, with the proper control signaling retrieved from the sync header information retrieved in the 64B/66B decoder.

101.3.3.1.3 LDPC decoding process within CLT upstream

In the Verify CRC40 and Extract BQ 65B Blocks function of the upstream receiver, the CLT will remove the first 65-bit block of a burst containing the Burst Time Header, the remainder of the 65-bit blocks will be passed to the 64B/66B decoder. The 32-bit PHY Link timestamp value will be extracted. The use of this timestamp in the upstream receiver processing is implementation specific and beyond the scope of this standard. This timestamp is provided as a mechanism to minimize any jitter introduced by OFDMA framing and the 1D to 2D insertion and extraction mechanisms. For example, an implementation may implement a timestamp based burst play out buffer prior to passing any frames contained within a burst to the XGMII.

101.3.3.1.4 Codeword error monitor

The FEC decoder in the CNU shall provide a user-configurable option (variable *CRC40ErrCtrl*) to indicate an uncorrectable FEC codeword to higher layers. If *CRC40ErrCtrl* is TRUE and the calculated value of CRC40 does not match the value of CRC40 retrieved from the received FEC codeword, the FEC decoder replaces bit <0> and <1> in the sync headers in first 64B/66B block and every 8th 64B/66B block, e.g. 1st, 9th, 17th, 25th, etc. as well as the last 64B/66B block from the errored FEC codeword with the binary value of "11". The state table for replacing the sync header bits is shown in Table 101–6. For EPoC, the BER monitor state machine as defined in Clause 49 is disabled.

Table 101–6—Sync header decoding state table

		CRC40ErrCtrl	
		FALSE	**TRUE**
CRC40	matches	Pass SH as received	Pass SH as received
	does not match	Pass SH as received	Replace SH with “11”

101.3.3.1.5 Constants

IDLE

TYPE: 66-bit vector
This constant represents /I/ character with 64B/66B encoding, as defined in 49.2.4.7.

XGMII_Rate

See 101.3.2.1.1.

101.3.3.1.6 Variables

blockCount

see 101.3.2.5.5

B_P

see 101.3.2.5.5

B_Q

see 101.3.2.5.5

CLK

see 101.3.2.5.5

CRC40ErrCtrl

TYPE: boolean

This variable controls the processing of codewords that fail the CRC40 checksum test. When *CRC40ErrCtrl* is TRUE 66B vectors that fail the CRC40 checksum test are flagged as errored. When this variable is set to FALSE 66B vectors that fail the CRC40 checksum test are passed as is.

dataInSize

TYPE:16 bit unsigned integer

VALUE: $(B_Q + 1) \times 65 + 40 + B_P$

This variable represents the size of the *dataIn* array in bits, containing the combination of the payload portion of the FEC codeword ($(B_Q + 1) \times 65$), the CRC40 (40), and the parity portion of the FEC codeword (B_P).

dataCrcA<39:0>

TYPE: Bit array

This array represents the CRC40 recovered from the payload portion of the FEC codeword prior to the FEC decoding process. This array is initialized to the size of 40 bits and filled with the binary value of “0”.

dataCrcB<39:0>

TYPE: Bit array

This array represents the CRC40 calculated over B_Q 65-bit blocks in the payload portion of the FEC codeword after the FEC decoding process. This array is initialized to the size of 40 bits and filled with the binary value of “0”.

dataIn<(dataInSize-1:0>

TYPE: Bit array

This array represents the combination of the payload portion of the FEC codeword, the parity portion of the FEC codeword, CRC40, and all the necessary padding. It is initialized to the size of *dataInSize* bits and filled with the binary value of “0”.

dataOut<F_P-1:0>

TYPE: Bit array

This array represents the combination of the payload portion of the FEC codeword, CRC40, and all the necessary padding. It is initialized to the size of F_P bits and filled with the binary value of “0”.

FecCodeWordCount

TYPE: 32-bit unsigned integer

This variable is incremented for every received FEC codeword. After reaching 0xFF-FF-FF-FF, this variable is set to 0x00-00-00-00.

FecCodeWordFail

TYPE: 32-bit unsigned integer

This variable is incremented for every received FEC codeword for which the decoding process failed. After reaching 0xFF-FF-FF-FF, this variable is set to 0x00-00-00-00.

FecCodeWordSuccess
TYPE: 32-bit unsigned integer
This variable is incremented for every received FEC codeword for which the decoding process completes successfully. After reaching 0xFF-FF-FF-FF, this variable is set to 0x00-00-00-00.

FIFO_FEC_RX
TYPE: Array of 66-bit blocks
A FIFO array used to store tx_coded<65:0> blocks, inserted by the input process in the FEC decoder, while encoded data is then sent to 64B/66B decoder for processing and transmission towards the XGMII.

loc
see 101.3.2.5.5

PMA_CLK
TYPE: boolean
This Boolean is to TRUE on every negative edge of a clock that is synchronized to the PMA_UNITDATA.indication data rate of *DS_DataRate* (see 101.4.1.2.1). This variable is set to FALSE upon read.

rx_coded_in<64:0>
TYPE: 65-bit block
This 65-bit block contains the input into the FEC decoder being passed from PMA. The left-most bit is *rx_coded_in<0>* and the right-most bit is *rx_coded_in<64>*.

rx_coded_in<FC-1:0>
TYPE: bit array
This array contains the input into the FEC decoder being passed from PMA. The left-most bit is *rx_coded_in<0>* and the right-most bit is *rx_coded_out<FC-1>*.

rxCount
TYPE: 16-bit unsigned integer
This variable used for counting bits in the Transfer from PMA process.

sizeFifo
see 101.3.2.5.5

syncFec
TYPE: Boolean
This variable indicates whether the FEC codeword alignment was found (value equal to TRUE) or not (value equal to FALSE).

tx_coded<65:0>
see 101.3.2.5.5

101.3.3.1.7 Functions

calculateCrc (ARRAY_IN)
see 101.3.2.5.5

decodeFec(ARRAY_IN, Length)
This function performs FEC decoding (for the codeword per Table 101–2 or Table 101–2) for data included in ARRAY_IN, comprising the combination of the FEC payload portion, CRC40, and FEC Parity (F_R). Since the FEC Parity and CRC are of constant size, if Length < F_C, the FEC payload will be shortened by F_C - Length bits (considered as padding bits B_P). The output of the decoder will be of length F_P bits comprised of the received FEC payload, the

received CRC40, and the added B_P padding bits. The padding bits will be set to a binary value of 0.

resetArray(ARRAY_IN)
see 101.3.2.5.5

transferFromPMA(ARRAY_OUT)
This function invokes the Transfer From PMA process to transfer a downstream burst (codeword) from the PMA to the FEC decoder.

101.3.3.1.8 State diagrams

EDITORIAL NOTE (to be removed prior to publication): if descriptions of CNU PMA and CLT FEC decoding processes are not clear state diagrams for them will be needed.

The CLT PCS shall implement the transfer from PMA process as shown in Figure 101–13.

The CNU PCS shall implement the LDPC decoding process input process, and LDPC decoding output process as shown in Figure 101–14 and Figure 101–15, respectively.

In case of any discrepancy between state diagrams and the descriptive text, the state diagrams prevail.

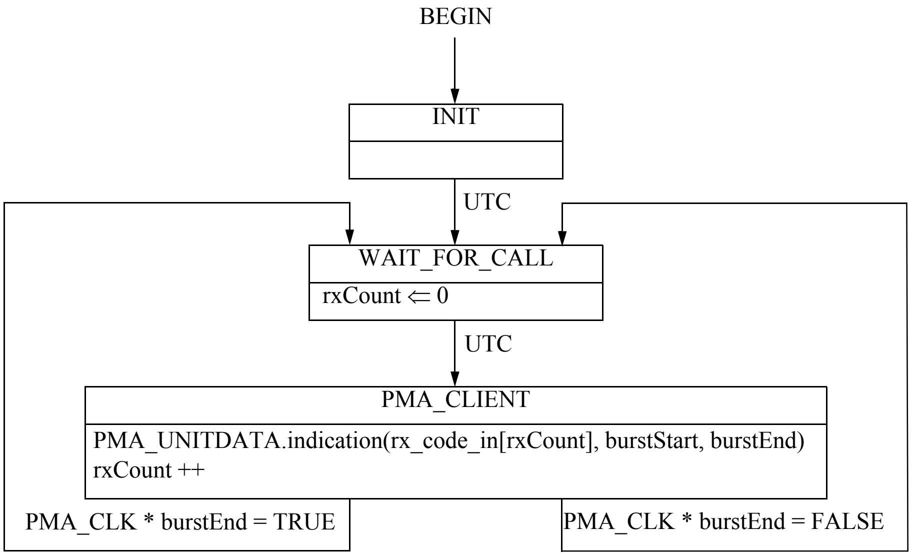

BEGIN
INIT
UTC
WAIT_FOR_CALL
rxCount ⇐ 0
UTC
PMA_CLIENT
PMA_UNITDATA.indication(rx_code_in[rxCount], burstStart, burstEnd)
rxCount ++
PMA_CLK * burstEnd = TRUE
PMA_CLK * burstEnd = FALSE

Figure 101–13—Upstream CLT transfer from PMA process

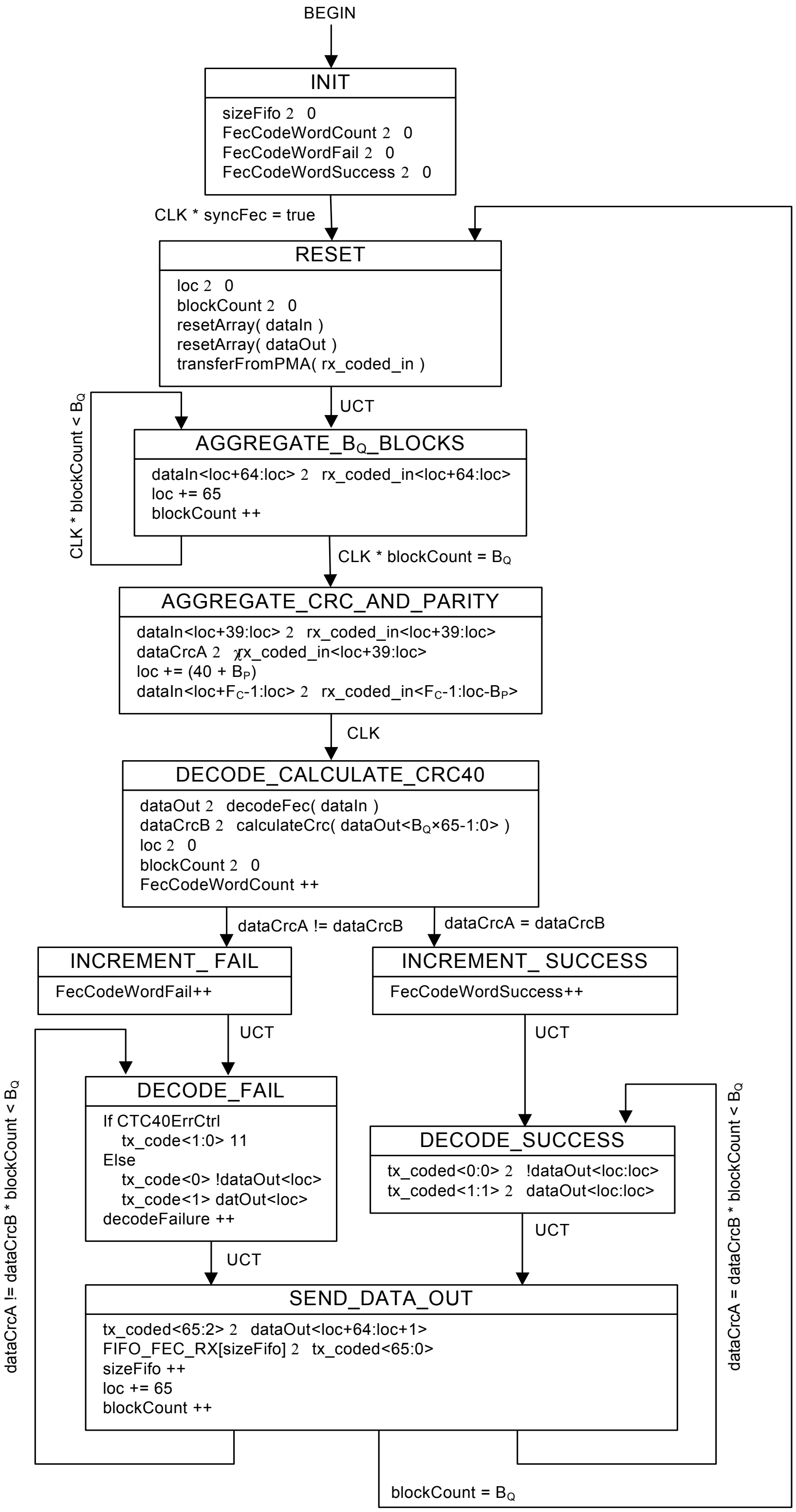


Figure 101–14—CNU FEC decoding input process

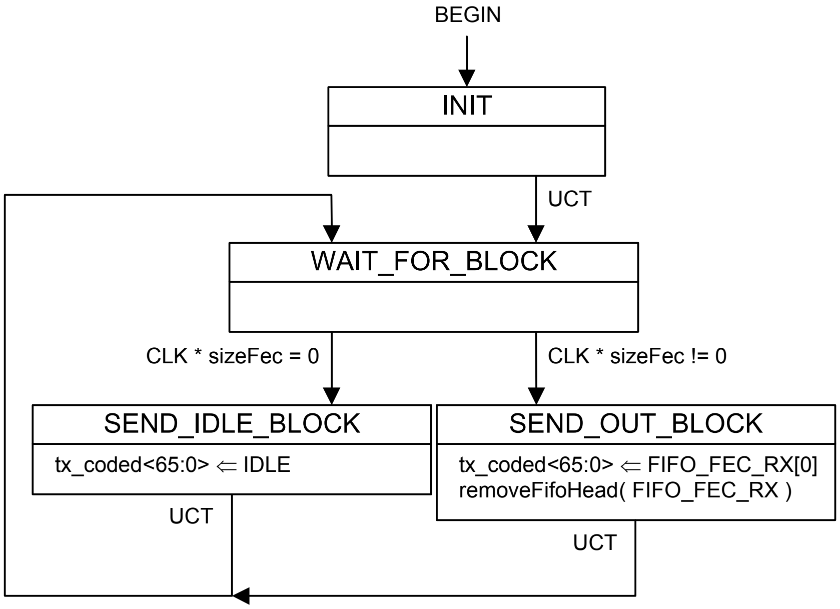


Figure 101–15—FEC decode, output process state diagram (CNU)

101.3.3.2 64B/66B decode

The EPoC PHY utilizes a 64B/66B decoder based on that described in 49.2.11 with several important differences. The EPoC 64B/66B decoder does not include a descrambler function as described in 49.2.10 and the input is a 65B block with a single synch header bit. The state diagram found in Figure 49-17 is followed after the addition of sync header bit <0> as illustrated in Figure 101–12.

101.3.3.3 Idle control character insertion process

In the receiving PCS, the Idle control character insertion process inserts Idle control characters into the data stream with gaps as received from the FEC decoder and 64B/66B decoder, adjusting the effective PCS and PMD data rate to the data rate expected by the MAC Control (as defined in Clause 103). Effectively, the Idle control character insertion process fills in the gaps created after the removal of FEC parity data, as well as compensates for the derating of the EPoC PMD relative to the EPoC MAC.

The Idle control character insertion process (see Figure 101–16) is composed of:

a) a receive process, receiving 72-bit vectors from the 64B/66B decoder and writing them into the Idle Insertion FIFO (called FIFO_II); and
b) a transmit process, reading 72-bit vectors from FIFO_II and transferring them to the XGMII.

The receive process receives 72-bit vectors from the 64B/66B decoder at a slower data rate than the nominal XGMII data rate for two reasons:

a) the FEC parity data is removed within the FEC decoder, leaving behind gaps in the data stream; and
b) the data rate supported by EPoC PCS and PMD is lower than the data rate expected by MAC Control Client, requiring data rate adaptation between the PCS and MAC.

The transmit process outputs 72-bit vectors at the nominal XGMII data rate.

To match the difference in data rates between the receive process and the transmit process, the Idle control character insertion process inserts additional 72-bit vectors containing Idle control characters. The additional blocks are inserted between frames and not necessarily at the same locations where FEC parity data was removed within the FEC decoder.

EDITORS NOTE (to be removed prior to publication): the text in this subclause needs to be updated to account for FEC parity removal and CRC40.

101.3.3.3.1 Constants

FIFO_II_SIZE

TYPE: 16-bit unsigned integer

This constant represents the size of Idle Insertion FIFO buffer. The size of this buffer is selected in such a way that it is able to accommodate the number of 66-bit vectors sufficient to fill the gap introduced by removing the FEC parity data for a maximum size MAC frame, and compensate for the maximum supported difference between the MAC rate and PMD rate.

Value: TBD

EDITORS NOTE (to be removed prior to publication): It seems that the FIFO_II_SIZE depends on the two following items: (a) the type of FEC and the size of FEC parity that is removed from data stream at regular intervals; and (b) the data rate differential between the PMD and the MAC. Every time the data rate changes, the size of FIFO_II may need to be adapted as well, to make sure that no additional delay / jitter is introduced. Whether such a change is needed, needs to be studied in more detail when more PMD/PCS details are available.

IDLE_VECTOR

TYPE: 72-bit binary array

This constant represents a 72-bit vector containing Idle control characters.

LBLOCK_R

This constant is defined in 49.2.13.2.1.

EDITORS NOTE (to be removed prior to publication): Note that the value of FIFO_II_SIZE, as well as the list of constants will be updated per technical decision #43 and #45 (http://www.ieee802.org/3/bn/public/decisions/decisions.html) once EPoC-specific FEC and PMD overhead details are settled.

101.3.3.3.2 Variables

BEGIN

TYPE: Boolean

This variable is used when initiating operation of the state diagram. It is set to TRUE following initialization and every reset.

FIFO_II

TYPE: Array of 72-bit vectors

The FIFO_II buffer is used to perform data rate adaptation between XGMII data rate and the EPoC PMD data rate. Upon initialization, all elements of this array are filled with instances of IDLE_VECTOR. The FIFO_II buffer has the size of FIFO_II_SIZE (see 101.3.3.3.1).

RX_CLK

TYPE: Boolean

This variable represents the RX_CLK signal defined in 46.3.2.1.

rx_raw_in<71:0>

TYPE: 72-bit binary array

This variable represents a 72-bit vector received from the output of the 64B/66B decoder. RXD<0> through RXD<31> for the second transfer are placed in rx_raw<40> through rx_raw<71>, respectively.

rx_raw_out<71:0>

TYPE: 72-bit binary array

This variable represents a 72-bit vector passed from the Idle control character insertion process to XGMII. The vector is mapped to two consecutive XGMII transfers as follows:

Bits rx_raw<3:0> are mapped to RXC<3:0> for the first transfer;
Bits rx_raw<7:4> are mapped to RXC<3:0> for the second transfer;
Bits rx_raw<39:8> are mapped to RXD<31:0> for the first transfer;
Bits rx_raw<71:40> are mapped to RXD<31:0> for the second transfer.

countVector

TYPE: 16-bit unsigned integer

This variable represents the number of 72-bit vectors stored in the FIFO_II at the given moment of time.

101.3.3.3.3 Functions

T_TYPE(rx_raw<71:0>)

This function is defined in 49.2.13.2.3.

101.3.3.3.4 Messages

DECODER_UNITDATA.indicate(rx_raw_in<71:0>)

A signal sent by the EPoC PCS Receive process, conveying the next received 72-bit vector.

DUDI

Alias for DECODER_UNITDATA.indicate(rx_raw_in<71:0>).

101.3.3.3.5 State diagrams

The CLT and CNU PCS shall perform the Idle control character insertion process as shown in Figure 101–16. In case of any discrepancy between state diagrams and the descriptive text, the state diagrams prevail.

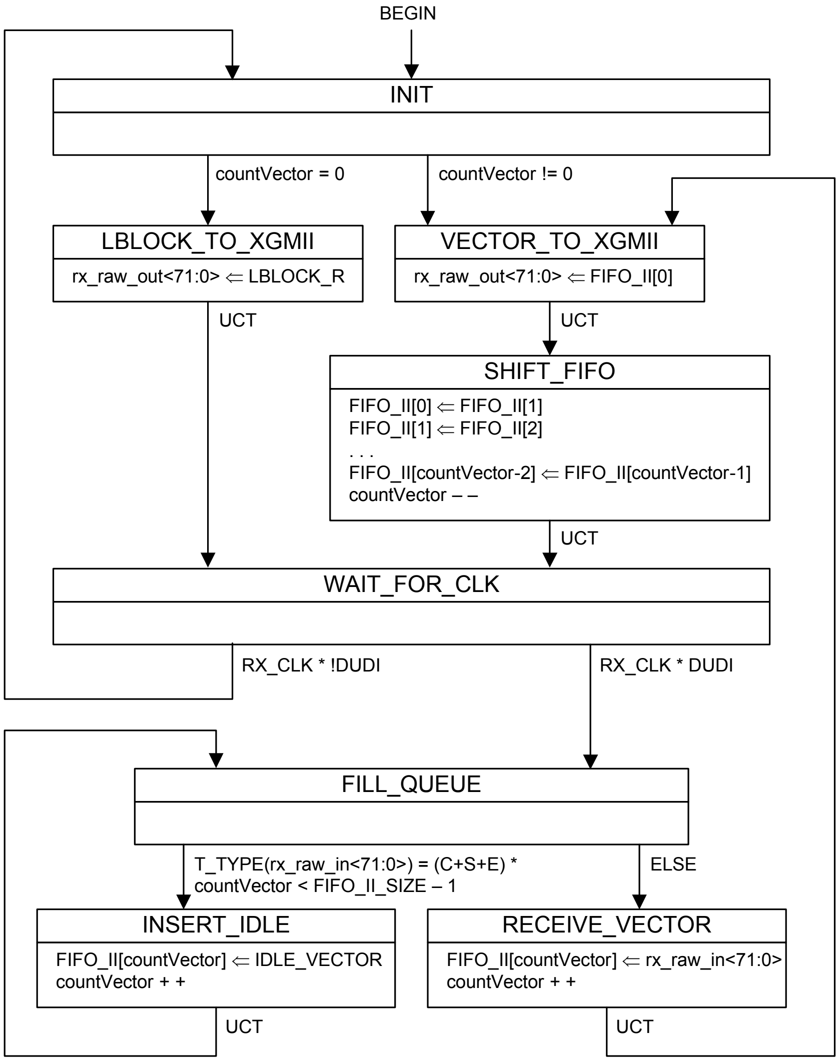


Figure 101–16—Idle control character insertion process state diagram

101.4 10GPASS-XR PMA

101.4.1 Overview

101.4.1.1 OFDM Profile descriptors

In an OFDM link a large number of parameters must be set in order to ensure both ends of a link operate as intended. In EPoC this set of parameters is referred to as a profile. Table 101–9 lists the variables that are included in the upstream and downstream OFDM profiles.

Two copies of every profile variable exist; an active copy and an inactive copy. During a profile switch (see 102.2.3.1) the network switches operation from the currently active profile to the inactive profile.

The currently active profile can be copied to the inactive profile via a profile copy. This is controlled via the *DS_PrflCpy* and *US_PrflCpy* variables. While the copy is in process, as indicated by the *DS_CpyInP* and *US_CpyInP* variables profile switches are prohibited and profile writes are ignored.

Table 101–7—Profile variables

Downstream Profile	Upstream Profile
DS_ModTypeSC(n)	*US_ModTypeSC(n)*

EDITORS NOTE (to be removed prior to publication): If the Modulation Type SC(n) are the only variables in a profile then the following text would be preferred to the above text and table for this subclause.

In the EPoC OFDM link the modulation or each subcarrier is set explicitly, the entire set of subcarrier modulation settings is referred to as the channel profile. Two copies of both the upstream and downstream profiles exist; an active copy and an inactive copy. During a profile switch (see 102.2.3.1) the network switches operation from the currently active profile to the inactive profile.

The currently active profile can be copied to the inactive profile via a profile copy. This is controlled via the *DS_PrflCpy* and *US_PrflCpy* variables. While the copy is in process, as indicated by the *DS_CpyInP* and *US_CpyInP* variables profile switches are prohibited and profile writes are be ignored.

101.4.1.1.1 Variables

DS_CpyInP

TYPE: boolean
This variable indicates that a copy of the currently active downstream profile to the inactive profile is in process. Note that while this variable has a value of one writes to all downstream profile variables shall be ignored and switching between profiles is prohibited.

DS_PrflCpy

TYPE: boolean
When this variable is set to one a copy of the currently active downstream profile to the inactive profile is initiated. Once initiated this action continues to completion (i.e., it cannot be interrupted or aborted once initiated).

US_CpyInP

TYPE: boolean
This variable indicates that a copy of the currently active upstream profile to the inactive profile is in process. Note that while this variable has a value of one writes to all upstream profile variables shall be ignored and switching between profiles is prohibited.

US_PrflCpy

TYPE: boolean
When bit this variable is set to one a copy of the currently active upstream profile to the inactive profile is initiated. Once initiated this action continues to completion (i.e., it cannot be interrupted or aborted once initiated).

EDITORS NOTE (to be removed prior to publication): the above definitions were copied from those in Cl 45. We should probably keep these are reference them from Cl 45 rather than keep both.

101.4.1.2 PMA Service Interface

The EPoC PMA provides a Service Interface to the 10GPASS-XR PCS sublayer, i.e., the PMA client. These services are described in an abstract manner and do not imply any particular implementation. The PMA Service Interface shall support the exchange of data between the PMA and the PMA client.

The PMA inputs serial data from the PCS and, after processing, passes serial data to the PMD and vice versa. It also generates an additional status indication for use by its client.

The following primitives are defined:
PMA_UNITDATA.request(tx_data_bit<bit>, burstStart, burstEnd)
PMA_UNITDATA.indication(rx_data_bit<bit>, burstStart, burstEnd)

101.4.1.2.1 PMA_UNITDATA.request

This primitive defines the transfer of data (in the form of data bits) from the PMA client to the PMA and notifies the PMA of the start and the end of the data burst.

PMA_UNITDATA.request is generated by the PMA client's transmit process.

101.4.1.2.2 Semantics of the service primitive

PMA_UNITDATA.request(tx_data_bit<bit>, burstStart, burstEnd)

The data conveyed by PMA_UNITDATA.request is a single data bit which has been prepared for transmission by the PMA client. The boolean variable burstStart is set to TRUE when the data bit is the first bit at the start of transmission burst, and is set to FALSE otherwise. The Boolean variable burstEnd is set to TRUE when the data bit is the last bit of a transmission burst, and is set to FALSE otherwise. In the downstream direction, the CLT transmission burst is composed of a single FEC codeword whereas in the CNU upstream, the burst may comprise of one or more concatenated FEC codewords (see 101.3.2.5.3).

101.4.1.2.3 When generated

The PMA client continuously sends data bits to the PMA at a nominal rate of *DS_DataRate* in the downstream direction. In the upstream direction, the nominal rate is *US_DataRate* during a burst. Refer to 100.2.6.2.

NOTE: *DS_DataRate* is calculated by the PMA after the downstream PHY has been configured. It is based on the sum of the available data bits per subcarrier over the timespan of a downstream OFDM frame consisting of 128 modulation symbols. *DS_DataRate* is a constant during the span of the PHY configuration. If re-configured, *DS_DataRate* must be re-calculated.

NOTE: *US_DataRate* is calculated by the PMA after the upstream PHY has been configured. It is based on the sum of the available data bits per Resource Element over the timespan of the upstream superframe consisting of 256 symbols plus 6 Probe Period symbols. *US_DataRate* is a constant during the span of the PHY configuration. If re-configured, *US_DataRate* must be re-calculated.

101.4.1.2.4 Effect of receipt

Upon receipt of this primitive, the PMA Symbol Mapper transfers the data bit into the OFDM frame.

In the CLT and upon the start of a downstream frame and when burstStart is TRUE, the PMA Symbol Mapper updates the FEC Codeword Pointer (FCP) in the downstream PHY Link. See 101.4.2.8.

In the CNU, both burstStart and burstEnd parameters are used by the upstream Symbol Mapper for placing start and end burst markers, respectively, into the appropriate resource elements. See 101.4.3.9.

101.4.1.3 PMA_UNITDATA.indication

This primitive defines the transfer of data in the form of bits from the PMA to its client. PMA_UNITDATA.indication is used by the client's synchronization process.

EDITORS NOTE (to be removed prior to publication): a precise description of what is meant by "PMA_UNITDATA.indication is used by the client's synchronization process" is needed.

101.4.1.3.1 Semantics of the service primitive

PMA_UNITDATA.indication(rx_data_bit<bit>, burstStart, burstEnd)

The data conveyed by PMA_UNITDATA.indication is single data bit that has been prepared for by the PMA receive process to the PMA client. The boolean variable burstStart is set to TRUE when the data bit is the first bit at the start of received burst, and is set to FALSE otherwise. The boolean variable burstEnd is set to TRUE when the data bit is the last bit of a received burst, and is set to FALSE otherwise.

101.4.1.3.2 When generated

The PMA sends one rx_data_bit <bit> to the PMA client corresponding to the receipt of each bit of received from the receiver Symbol De-Mapper.

In the CNU, the PMA continuously sends data bits to the PMA client at a nominal rate of *DS_DataRate* in the downstream direction. In the upstream direction, the nominal rate is *US_DataRate* during a burst. Refer to 100.2.6.1 and 100.2.6.2.

101.4.1.3.3 Effect of receipt

The effect of receipt of this primitive by the client is unspecified by the PMA sublayer.

101.4.2 Downstream PMA transmit function

101.4.2.1 Overview

The downstream PMA transmit functional diagram is shown in Figure 100–3. The PMA supports five 190 MHz wide OFDM channels; each containing 3800 subcarriers. Each OFDM channel is associated with the following processing functions: Time and Frequency Interleaver (see 101.4.2.9), Pilot Insertion (see 101.4.2.10), Inverse Discrete Fourier Transform (IDFT) (see 101.4.2.11), and cyclic prefix and Windowing (see 101.4.2.12). The outputs of each OFDM channel are digitally combined. All OFDM channels use the same sampling rate clock as per Table 101–8 and follow the same frame timing.

OFDM channel 1 shall always be enabled but is muted during RxMER testing (see 100.3.2). Optional OFDM channels 2, 3, 4, and 5 are enabled when configured for operation. When enabled, each OFDM channel is configured for placement in the downstream RF frequency band of the coax cable distribution network (see Figure 100–1). The encompassed spectrum of any OFDM channel does not overlap with that of any other OFDM channel. The tight time skew requirements (see Table 101–8) permit the active edge subcarrier of one OFDM channel to be placed immediately adjacent to that of the adjacent OFDM channel without any guard band.

The Symbol Mapper distributes PCS data over all active subcarriers in all OFDM channels that are enabled that are configured to carry data. See 101.4.2.8.

The PHY Link is processed by the OFDM channel 1 IDFT and cyclic prefix and Windowing functions.

101.4.2.2 Time and frequency synchronization

This subclause specifies the timing and frequency synchronization requirements for CLT transmitters and CNU receivers.

The purpose of this section is to ensure that the CLT transmitter can provide proper timing and frequency references for EPoC downstream OFDM operation and that the CNU receiver can acquire the system timing and subcarrier from the downstream for proper EPoC operation.

The CLT downstream OFDM symbol and subcarrier frequency and timing relationship is defined in 101.4.2.3.

Tolerances for the downstream subcarrier clock frequency are given in this subclause Table 100–3. Functional requirements involving the downstream subcarrier clock frequency and downstream signal generation are contained in 101.4.2.3, which couple the subcarrier clock frequency tolerance performance to the phase noise requirements of Table 100–3 and the downstream OFDM symbol clock requirements of this subclause. Each cycle of the downstream subcarrier clock is 4096 cycles (50 kHz subcarrier spacing) of the downstream OFDM symbol clock (which is nominally 204.8 MHz), since the subcarrier clock period is defined as the FFT duration for each OFDM symbol. Functional requirements on locking the downstream waveform to the 10.24 MHz Master Clock are then equivalently functional requirements locking the downstream subcarrier clock to the Master Clock. Downstream OFDM symbol clock jitter requirements (which are in the time domain) of Table 101–8 are equivalently requirements on the downstream subcarrier clock (and its harmonics). The requirements on the OFDM symbol clock are effectively measured on observables in the downstream waveform, which include the downstream subcarrier clock frequency (manifested in the subcarrier spacing) and downstream subcarrier frequencies.

CLT transmitters and CNU receivers shall conform to the requirements given in Table 101–8.

Table 101–8—Downstream time and frequency synchronization

Item	Requirement
OFDM Symbol Clock Jitter	- < [-21 + 20*log (f_{DS} /204.8)] dBc (i.e., < 0.07 ns RMS) 10 Hz to 100 Hz - < [-21 + 20*log (f_{DS} /204.8)] dBc (i.e., < 0.07 ns RMS) 100 Hz to 1 kHz - < [-21 + 20*log (f_{DS} /204.8)] dBc (i.e., < 0.07 ns RMS) 1 kHz to 10 kHz - < [-4 + 20*log (f_{DS} /204.8)] dBc (i.e., < 0.5 ns RMS) 10 kHz to 100 kHz - < [2 + 20*log (f_{DS} /204.8)] dBc (i.e., < 1 ns RMS) 100 kHz to (f_{DS} /3), where f_{DS} is the frequency of the measured downstream OFDM clock in MHz. [a]
Inter-channel time skew	156.25 ns
CNU Timing Acquisition Accuracy	better than 1 sample (1/204.8 MHz).
Acquisition Time	< 60 seconds

[a] The CLT uses a value of f_{DS} that is an integral multiple or divisor of the downstream symbol clock. For example, an f_{DS} = 409.6 MHz clock may be measured if there is no explicit 204.8 MHz clock available.

The CLT shall lock the 204.8 MHz downstream OFDM Clock and downstream OFDM RF transmissions to the 10.24 MHz CLT Master Clock (Table 101–8).

Inter-channel time skew is defined as the maximum transmission time skew between any two OFDM channels.

The CNU timing acquisition accuracy for the downstream clock timing is defined with respect to downstream PHY Link frame. The CNU shall adjust its clock to synchronize its own clock timing with PHY Link frame for proper operation. The CNU acquires downstream clock timing from the downstream signal (pilots, preambles, or mixed pilots, preambles, and data).

Acquisition Time for the CNU is defined as the time required for a CNU with no previous network frequency plan knowledge to achieve downstream signal acquisition (frequency and time lock).

EDITORS NOTE (to be removed prior to publication): The text of subclause 7.5.3 in DOCSIS 3.1 IO3 should be reviewed for applicability to this subclause.

In addition to meeting the clock jitter requirements given above, the CLT is required to meet the phase noise specifications defined in Table 100–4. In the event of a conflict between the clock jitter and the phase noise requirement, the CLT shall meet the more stringent requirement.

101.4.2.3 Subcarrier Clocking

The "locking" of subcarrier "clock and carrier" are defined and characterized by the following rules:

- Each OFDM symbol is defined with an FFT duration (equal to subcarrier clock period) of 20 μs. For each OFDM symbol, the subcarrier clock period (μs) may vary from nominal with limits defined in 101.4.2.2.
- The number of cycles of each subcarrier generated by the CLT during one period of the subcarrier clock (for each OFDM symbol) MUST be an integer number. The CLT subcarrier clock shall be synchronous with the 10.24 MHz Master Clock defined by:

  subcarrier clock frequency = (40 / 8192) * Master Clock frequency
- The limitation on the variation from nominal of the subcarrier clock frequency at the output connector is defined in 101.4.2.2.
- Each OFDM symbol has a cyclic prefix which is an integer multiple of 1 / 64th, of the subcarrier clock period.
- Each OFDM symbol duration is the sum of one subcarrier clock period and the cyclic prefix duration.
- The number of cycles of each subcarrier generated by the CLT during the OFDM symbol duration (of each symbol) shall be $K + K * L / 64$, where K is an integer related to the subcarrier index and frequency up-conversion of the OFDM channel, and L is an integer related to the cyclic prefix. (K is an integer related to the subcarrier index and increases by 1 for each subcarrier).
- The phase of each subcarrier within one OFDM symbol is the same, when each is assigned the same constellation point (I + jQ), relative to the Reference Time of the OFDM symbol. There is nominally no change in phase on each subcarrier for every cycle of 64 OFDM symbols, when both are assigned the same I + jQ, and referenced to the Reference Time of their respective OFDM symbol.

101.4.2.4 Subcarrier configuration and bit loading

Each subcarrier in an OFDM channel is configured using the *DS_ModTypeSC(n)* variables (where $0 \leq n \leq 4095$). These variables allow the Phy to configure each subcarrier to be nulled, to be a continuous pilot, to have a specific bit loading (such as 512-QAM or 1024-QAM), or to be excluded. Subcarriers that are not configured as excluded are active subcarriers. Subcarrier configuration in an EPoC OFDM channel of 192 MHz shall conform to the rules outlined in Table 101–9.

All CNUs and the CLT in an EPoC network share the same downstream subcarrier configuration and bit loading including nulled subcarriers, continuous pilots, bit loaded subcarriers and excluded subcarriers.

Table 101–9—Downstream subcarrier configuration rules

Parameter	Limit	Unit
Minimum number of active subcarriers in a contiguous group	40	subcarriers
Minimum OFDM channel guard band	1	MHz
Maximum excluded spectrum in the encompassed spectrum	20	%

101.4.2.4.1 Nulled subcarriers

Nulled subcarriers do not carry MAC or PHY Link data but may be used as pilots. Nulled subcarriers are not modulated except when being used as a scattered pilot in the downstream direction (see 101.4.2.6.1).

101.4.2.4.2 Continuous pilots

In the downstream direction continuous pilots are used to help delineate the downstream PHY Link (see 101.4.2.6.2).

101.4.2.4.3 Bit loaded subcarriers

When a subcarrier is used to carry MAC data it uses the modulation type of QPSK or 2^n-QAM, where the integer n is $4 \le n \le 16$, assigned via the *DS_ModTypeSC(n)* variable except when used as a downstream scattered Pilot (see 101.4.2.5.1).

There is at least one contiguous 22 MHz or greater band of subcarriers with an assigned bit loading in any single 192 MHz OFDM channel. This 22 MHz band may include subcarriers intended as Pilots and PHY Link subcarriers. A 1 MHz guardband of excluded subcarriers above and below this 22 MHz creates a minimum width OFDM channel of 24 MHz encompassed spectrum.

101.4.2.4.4 Excluded subcarriers

An EPoC PHY shall not transmit energy into a subcarrier that has been excluded from the OFDM channel (i.e, excluded subcarriers have zero amplitude). Typically there is a band edge Exclusion Band at both the top and bottom of the OFDM channel and there may be up to 14 exclusion bands internal to a single 192 MHz OFDM channel.

Exclusion bands are limited to 20% or less of encompassed spectrum (see Table 101–9).

101.4.2.4.5 Variables

DS_ModTypeSC(n)

TYPE: 4-bit binary

This set of variables determines the modulation parameters for each of the 4096 downstream OFDM subcarriers ($0 < n < 4095$). Each variable controls one of the 4096 subcarriers that comprise an OFDM channel, with *DS_ModTypeSC(0)* controlling subcarrier zero, *DS_ModTypeSC(1)* controlling subcarrier 1, etc. The assignment of bits to each modulation type is show below.

bit 3 2 1 0
1 1 1 1 = Excluded subcarrier
1 1 1 0 = 16384-QAM
1 1 0 1 = 8192-QAM
1 1 0 0 = 4096-QAM

1 0 1 1 = 2048-QAM
1 0 1 0 = 1024-QAM
1 0 0 1 = 512-QAM
1 0 0 0 = 256-QAM
0 1 1 1 = 128-QAM
0 1 1 0 = 64-QAM
0 1 0 1 = 32-QAM
0 1 0 0 = 16-QAM
0 0 1 1 = 8-QAM
0 0 1 0 = QPSK
0 0 0 1 = BPSK (Used for continuous pilots only)
0 0 0 0 = null (carries no data but used for Wideband Probing)

101.4.2.5 Framing

The downstream OFDM frame is synchronized to the downstream PHY Link frame and the downstream Timestamp (see 102.2). Each downstream frame is composed of 128 OFDM symbols as illustrated in Figure 101–17 and Figure 102–12. The first 8 symbols of the downstream frame coincide with the 8 downstream symbols of the downstream PHY Link preamble. The Timestamp marks the first subcarrier of the first symbol after the Preamble.

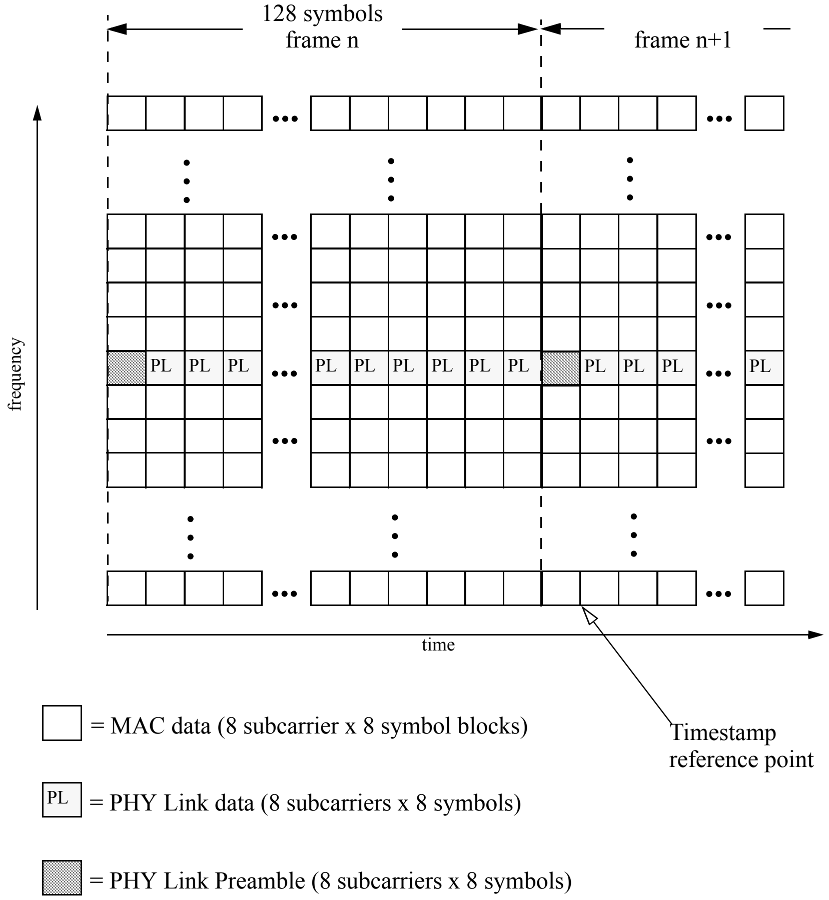

128 symbols
frame n
frame n+1
frequency
PL
time
Timestamp
reference point
= MAC data (8 subcarrier x 8 symbol blocks)
PL = PHY Link data (8 subcarriers x 8 symbols)
= PHY Link Preamble (8 subcarriers x 8 symbols)

Figure 101–17—DS OFDM frame structure

101.4.2.6 Pilot map

Downstream pilots are comprised of subcarriers modulated with a predefined data sequence known to all CNUs. The pilot data sequence is conveyed via the Pilot Insertion function (see 101.4.2.10 and Figure 100–2). Pilot Insertion follows time and frequency interleaving, before IDFT processing.

There are two types of downstream pilots: continuous and scattered. Scattered pilots occur at different frequency locations in different symbols in a repeating cyclic pattern. Continuous pilots occur at fixed frequencies in every symbol.

EDITORS NOTE (to be removed prior to publication): On the to-do list: the text and sections for Downstream Pilot Patterns may need to be rationalized with the Pilot Map and Pilot Insertion functions shown in Figure 101-2 or vice versa.

101.4.2.6.1 Scattered pilots

The scattered pilot pattern shall be synchronized to the PHY Link as illustrated in Figure 101–18. The first OFDM symbol after the PHY Link preamble has a scattered pilot in the subcarrier just after the highest frequency subcarrier of the PHY Link.

The remainder of the scattered pilot pattern is placed so that in each symbol scattered pilots occur every 128 subcarriers. From symbol to symbol, scattered pilots are shifted by one subcarrier position in the direction of the increasing frequency. This will result in scattered pilots placed in the exclusion band and in the 400 kHz PHY Link band, such scattered pilots are not transmitted.

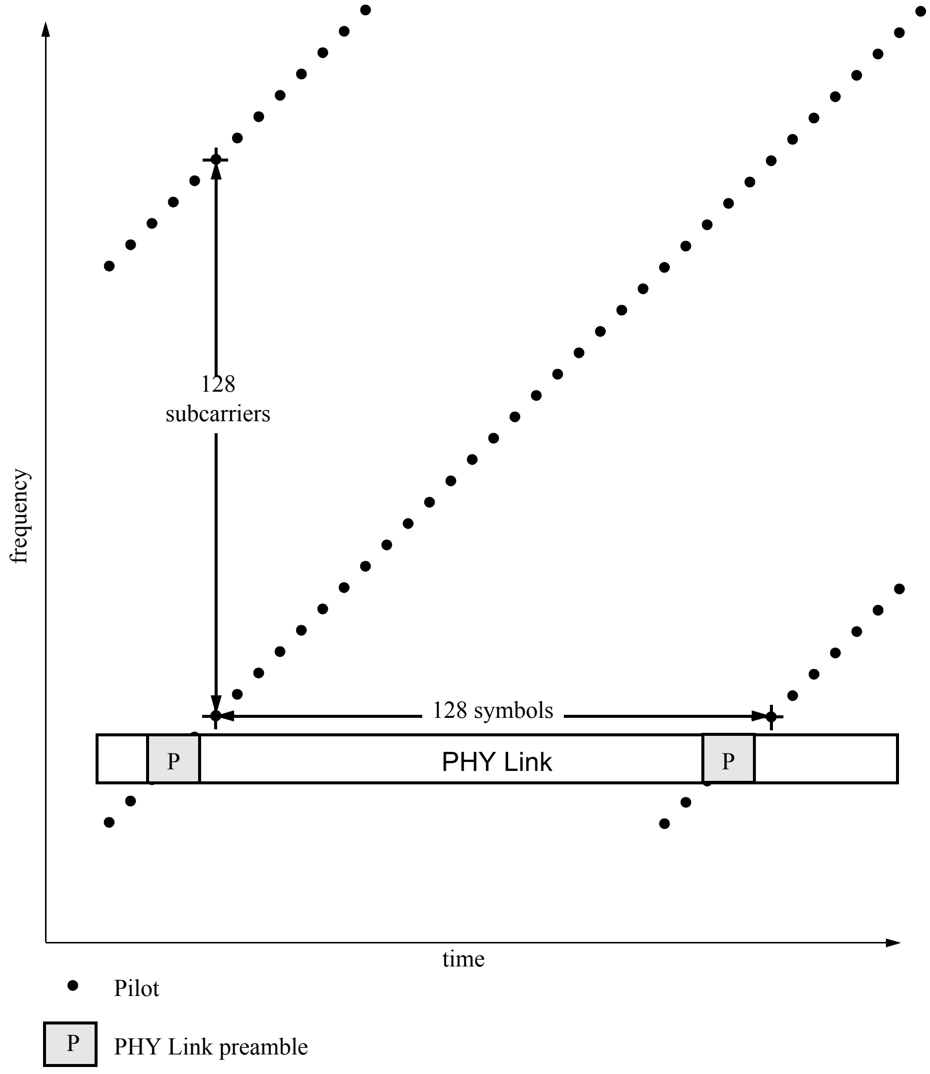


Figure 101–18—Downstream scattered pilot pattern

Mathematically, the scattered pilot pattern shall be defined as follows.

Let a subcarrier just after the PHY Link preamble be referred to as *x(m,n)*,
where:
m is the frequency index
n is the time index (i.e., the OFDM symbol number)

The scattered pilots in the 128 symbols following (and including symbol *n*) are given by:

Symbol *n*:*x(n, m±128i)*, for all non-negative integers i
Symbol (*n+1*):*x(n+1, m±128i + 1)* , for all non-negative integers i
Symbol (*n+2*):*x(n+2, m±128i + 2)* , for all non-negative integers i
...
Symbol (*n+127*):*x(n+127, m±128i + 127)* , for all non-negative integers i

Each of the above locations is a scattered pilot, provided that it does not fall on a continuous pilot, on the PHY Link, on an exclusion zone or on a excluded subcarrier. If the scattered pilot coincides with a continuous pilot it is treated as a continuous pilot and not as a scattered pilot.

This pattern repeats every 128 symbols. That is, symbol (*128+n*) has the same scattered pilot pattern as symbol *n*.

101.4.2.6.2 Continuous pilots

Continuous pilots occur at the same frequency location in all symbols and are used for receiver synchronization. Placement of continuous pilots is determined in two ways:

1) predefined continuous pilot placement around the PHY Link (see Figure 102–3), or
2) continuous pilot placement defined via PHY Link instructions.

Note that continuous and scattered pilots can overlap; the amount of overlap, in terms of number of carriers, changes from symbol to symbol. Overlapping pilots are treated as continuous pilots.

101.4.2.6.3 Predefined continuous pilots around the PHY Link

As discussed in 102.2, the PHY Link is placed at the center of a 6 MHz spectral region. Four pairs of predefined continuous pilots shall be placed symmetrically around the PHY Link as shown in Figure 102–8 at the distances indicated in Table 101–10. The spacing between each pilot pair and the PHY Link are different to prevent all pilots from being impacted at the same time by echo or interference.

The locations of the continuous pilots are defined with reference to the edges of the PHY Link band. Hence, once the PHY Link has been detected, these continuous pilots also become known to the receiver.

Table 101–10 provides the values of *d1, d2, d3,* and *d4*, measured in number of subcarriers from the PHY Link edge. That is, *dx* is the absolute value of the difference between the index of the continuous pilot and the index of the PHY Link subcarrier at the PHY Link edge nearest to the continuous pilot. The index of a subcarrier is the integer *k* of the IDFT definition given in 101.4.2.11. For example, let the lowest frequency subcarrier of the PHY Link have the IDFT index *k* equal to 972. Then according to Table 101–10 the continuous pilot nearest to this lowest frequency PHY Link subcarrier will have the IDFT index *k* of (972-15)=957. The index *k* of the highest frequency PHY Link subcarrier of this OFDM channel is 979. Hence continuous pilot that is nearest upper frequency edge of the PHY Link has an index *k* of 994.

For each distance (*dx*) defined in Table 101–10, the CLT places two pilots: one *dx* subcarriers above and one *dx* subcarriers below the edge of the PHY Link band.

Table 101–10—Subcarrier distances for placement of predefined pilots

	d_1	d_2	d_3	d_4
PHY Link 8 subcarriers	15	24	35	47

101.4.2.6.4 Continuous pilot placement defined by PHY Link message

The CLT defines a set of continuous pilots distributed as uniformly as possible (see below) over the entire OFDM spectrum in addition to the predefined continuous pilots described in 101.4.2.6.3.

The CLT ensures that there are no isolated active OFDM spectral regions that are not covered by continuous pilots.

The CLT provides the continuous pilot placement definition via the 10GPASS-XR DS profile descriptor variables *DS_ModTypeSC(n)* using the PHY Link EPoC message block format contained in 102.2.3.3.

The CLT shall place continuous pilots (excluding the eight continuous pilots around the PHY Link) per the 8 Steps below after calculating a value for N_{PC} using Equation (101–5).

The CLT obtains the value of N_{PC} using the following formula:

$$N_{PC} = min\left(max\left(8, \left\lceil CntPltSF \times \left(\frac{F_{max} - F_{min}}{190e6}\right)\right\rceil\right), 120\right) \quad (101–5)$$

where:

F_{max} refers to frequency in Hz of the highest frequency active subcarrier of the OFDM channel
F_{min} refers to frequency in Hz of the lowest frequency active subcarrier of the OFDM channel
$CntPltSF$ is the continuous pilot scaling factor

The number of continuous pilots is between 16 and 128. This range includes the eight continuous pilots around the PHY Link channel.

The value of *CntPltSF* in Equation (101–5) is kept as a parameter that can be adjusted by the CLT. The typical value proposed for *CntPltSF* is 48.

The CLT shall follow Step 1 through Step 8 as specified below for defining the frequencies for the location of these continuous pilots.

Step 1:

Merge all the subcarriers between F_{max} and F_{min} eliminating the following:

1) Exclusion bands,
2) 6 MHz band containing the PHY Link,
3) Known regions of interference.

Let the merged frequency band be defined as the frequency range $[0, F_{mergedmax}]$.

Step 2:

Define a set of N_{PC} frequencies using the following equation:

$$F_i = \frac{F_{mergedmax}}{2N_{PC}} + \frac{i \times F_{mergedmax}}{N_{PC}}, \text{ for } i = 0, 1, \ldots, N_{PC} - 1 \quad (101–6)$$

This yields a set of uniformly spaced N_{PC} frequencies:

$$\left\{\frac{F_{mergedmax}}{2N_{PC}}, \frac{3F_{mergedmax}}{2N_{PC}}, \ldots, F_{mergedmax} - \frac{F_{mergedmax}}{2N_{PC}}\right\} \qquad (101\text{–}7)$$

Step 3:

Map the set of frequencies given above to the nearest subcarrier locations in the merged spectrum. This will give a set of N_{PC} approximately uniformly spaced subcarriers in the merged domain.

Step 4:

De-merge the merged spectrum through the inverse of the operations through which the merged spectrum was obtained in step 1.

Step 5:

If any continuous pilot is within 1 MHz of a band edge, move this inwards (but avoiding subcarrier locations impacted by interferences like CSO/CTB) so that every continuous pilot is at least 1 MHz away from a band edge. This is to prevent continuous pilots from being impacted by external interferences. If the width of the spectral region does not allow the continuous pilot to be moved 1 MHz from the edge then the continuous pilot has to be placed at the center of the spectral band.

Step 6:

Identify any spectral regions containing active subcarriers (separated from other parts of the spectrum by exclusion bands on each side) that do not have any continuous pilots. Introduce an additional continuous pilot at the center of every such isolated active spectral region.

In the unlikely event that the inclusion of these extra pilots results in the total number of continuous pilots defined by PHY Link exceeding 120, return to step 1 and re-do the calculations after decrementing the value of N_{PC} by one.

Step 7:

Test for periodicity in the continuous pilot pattern and disturb periodicity, if any, through the perturbation of continuous pilot locations using a suitable algorithm. A simple procedure would be to introduce a random perturbation of up to ±5 subcarrier locations around each continuous pilot location, but avoiding subcarrier locations impacted by interferences like CSO/CTB.

Step 8:

The CLT shall transmit this continuous pilot pattern to the CNUs in the system and communicate the placement using the PHY Link.

101.4.2.6.5 Variables

CntPltSF

TYPE:7-bit unsigned integer
This variable is used to determine the number of continuous pilots in the downstream OFDM channels.
$120 \geq \mathit{CntPltSF} \geq 48$.

101.4.2.7 Scrambler

The Phy shall scramble the output of the LDPC FEC encoding process using a linear feedback shift register mechanism as illustrated in Figure 101–19.

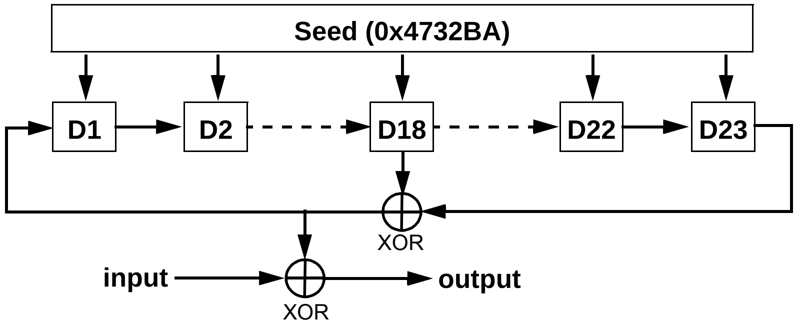


Figure 101–19—Scrambler

The scrambler is defined by the following polynomial.

$$x^{23} + x^{18} + 1$$

The scrambler is initialized to the hexadecimal value of 0x4732BA. The CLT initializes the scrambler at the first codeword of the downstream frame. The CNU initializes the scrambler with the hexadecimal value at the beginning of each grant.

101.4.2.8 Symbol mapper

101.4.2.8.1 Introduction

The Symbol Mapping function performs the following:

- Begins by initializing (resetting) the scrambler function (see 101.4.3.6.4) and setting an bit counter to 1 (see 101.4.3.6.x) as well as initializes to begin mapping with the lowest numbered active subcarrier.
- Continually accepts a tx_unit (bit) from the Scrambler via the PMA_UNITDATA.request as well as monitors the boolean state of burstStart and burstEnd (see 101.4.2.1) and the start of OFDM frame indication from the Frame Timing function. Also, the FCP bit counter is incremented for each bit processed.
- Per OFDM symbol, allocates scrambled tx_unit bits to all active OFDM data-carrying subcarriers (over all active channels) in ascending order based on a mapping configuration that anticipates frequency interleaving, staggered pilot placement and PHY Link signals per channel. While processing tx_unit bits, upon a start of frame indication, the bit counter is reset to one and the scrambler is reset before mapping the current bit to a subcarrier. Upon the next transition of burstStart = TRUE, the FCP update function calculates the next FCP value and updates the current PHY Link message (see 101.4.2.8.4).
- Converts Tx_unit bits to an array of QAM constellation points using a two-dimensional array with an I and Q “bin” value for each subcarrier and passes these values to the Interleaver.
- When the last active subcarrier of the current symbol is completed, counter k is reset to 1 and processing of the next OFDM symbol begins.

The transmitter uses the number of bits per subcarrier as defined in Table 100–3 when bit mapping subcarrier (MAC) data to QAM constellations. Permissible modulation Types are listed in Table 100–2. QAM constellation mappings are described in 101.4.4.

101.4.2.8.2 Transmitter bit loading for symbol mapping

The excluded versus non-excluded (active) status, PHY Link use, continuous pilot placement, and bit loading pattern (profile) information for each subcarrier is provided by the Subcarrier Configuration and Bit Loading Function. This information is configured by management.

The notation $S^{(E)}$ is used here to define the non-empty set of excluded subcarriers.

The notation $S^{(C)}$ is used here to define the set of continuous pilots (see 101.4.2.6.2).

The notation $S^{(P)}$ is used here to define the set of PHY Link subcarriers (see 101.4.2.6.3).

For bit loading, continuous pilots and the PHY Link are treated in the same manner as excluded subcarriers; hence, the set of subcarriers that includes the PHY Link, continuous pilots and excluded subcarriers is defined as:

$$S^{(PCE)} = S^{(P)} \cup S^{(C)} \cup S^{(E)} \tag{101–8}$$

The subcarriers in the set $S^{(PCE)}$ do not carry MAC data (PHY Link carries signaling information).

Bit loading information includes the option for zero bit-loading. Such subcarriers are referred to as zero-bit-loaded subcarriers and are BPSK modulated, as described in 101.4.4.2.

EDITORS NOTE (to be removed prior to publication): May need to adjust “zero-bit-loaded” via more socialization on its use.

All active subcarriers, with the exception of pilots, are transmitted with the same average power.

EDITORS NOTE (to be removed prior to publication): the above statement seems out of place and would be more appropriate in 101.4.2.7 Pilot Insertion.

Scattered pilots do not occur at the same frequency in every symbol; in some cases scattered pilots will overlap with continuous pilots. If a scattered pilot overlaps with a continuous pilot, then that pilot is no longer considered to be a scattered pilot. It is treated as a continuous pilot.

Because the locations of scattered pilots change from one OFDM symbol to another, the number of overlapping continuous and scattered pilots changes from symbol to symbol. Since overlapping pilots are treated as continuous pilots, the number of scattered pilots changes from symbol to symbol.

The following notation is used here:

N:The total number of subcarriers in the OFDM symbol (4096 per OFDM channel)
N_C:The number of continuous pilots in an OFDM symbol
N_S:The number of scattered pilots in an OFDM symbol
N_E:The number of excluded subcarriers in an OFDM symbol
N_P:The number of PHY Link subcarriers in an OFDM symbol
N_D:The number of data subcarriers in an OFDM symbol
$\mathrm{N_I}$:The number of scattered pilots and data subcarriers in the OFDM symbol

The values of N, N_C, N_E and N_P do not change from symbol to symbol for a given OFDM template; the values of N_S and N_D change from symbol to symbol.

The following equations holds for all symbols:

$$N = N_C + N_S + N_E + N_P + N_D \tag{101–9}$$

$$N_I = N_S + N_D \tag{101–10}$$

Interleaving and de-interleaving are applied to the set of data subcarriers and scattered pilots of size N_I.

101.4.2.8.3 Bit loading

EDITORS NOTE (to be removed prior to publication): a state diagram is needed for this subclause.

The bit loading pattern defines the QAM constellations assigned to each of the 4096 (N) subcarriers per OFDM channel of the OFDM transmission. Let the bit loading pattern per OFDM channel for configuration *i* be defined as $A_i(k)$, where:

k is the subcarrier index that goes from 0 to 4095

$A_i(k) \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14\}$. A value of 0 indicates that the subcarrier k is zero-bit-loaded. Other values indicate that the modulation of subcarrier k is QAM with order $2^{A_i(k)}$. See variable *DS_ModTypeSC(n)* defined in 101.4.2.4.5.

The function $A_i(k)$ and subsequent bit loading functions are defined per OFDM channel as the interleaver function is replicated with identical operation per OFDM channel. As five OFDM channels are accommodated (refer to Table 101–13) with a labeling of 1 to 5 (L), the bit loading processing ordering is $L = \{1, 2, 3, 4, 5\}$. OFDM channel 1 is always present and contains active data subcarriers.

Note that downstream RF spectrum availability as well as device implementation will determine OFDM channel presence and actual subcarrier use. The symbol mapping function therefore shall process all active subcarriers per symbol across all OFDM channels.

Let the sequence

$\{A_i(k), k = 0, 1, \ldots, (N-1), k \notin S^{(PCE)}\}$ be arranged as N_I consecutive values of another sequence:

$B_i(k), k = 0, 1, \ldots, (N_I - 1)$.

Given the locations of the excluded subcarriers, continuous pilots and the PHY Link in the OFDM configuration, it is possible to obtain the bit-loading pattern $B_i(k)$ that is applicable only to spectral locations without excluded subcarriers, continuous pilots, and PHY Link subcarriers. However, note that $B_i(k)$ does contain the spectral locations occupied by scattered pilots; these locations change from symbol to symbol.

The bit loading pattern is defined in the domain in which subcarriers are transmitted on the media, however bit loading is processed prior to interleaving. As such there is a permutation mapping of subcarriers, defined by the interleaving function, between the domain in which bit loading is applied to subcarriers and the domain in which subcarriers are transmitted.

The total number of subcarriers that pass through the interleaver and de-interleaver is given in Equation (101–10) and does not change from symbol to symbol. The frequency interleaver introduces a one-to-one permutation mapping P on the N_I subcarriers.

Note: the corresponding permutation mapping applied at the receiver de-interleaver is P^{-1}.

The bit loading pattern at the input to the interleaver. This is given by:

$$C_i(k) = P^{-1}(B_i(k)) \tag{101–11}$$

The sequence $C_i(k)$ is obtained by sending $\{B_i(k), k = 0, 1, \ldots, N-1\}$ through the frequency de-interleaver.

Note that $C_i(k)$ gives the bit-loading pattern for N_I subcarriers per OFDM channel. Scattered pilots are avoided in the bit-loading process and identified by a two-dimensional binary pattern $D(k, j)$ per OFDM channel. The scattered pilot pattern has a periodicity of 128 in the time dimension, this binary pattern also has periodicity 128 in the column dimension *j*.

$D(k, j)$ is defined for $k = 0, 1, ..., (N_I -1)$ and for $j = 0, 1, ..., 127$.

The binary pattern $D(k, j)$ begins with the transmitted scattered pilot pattern defined in 101.4.2.6.1.

The CLT executes the following steps to obtain the pattern $D(k, j)$:

1) Define a two-dimensional binary array $P(k, j)$ per OFDM channel in the subcarrier transmitted domain that contains a one for each scattered pilot location and zero otherwise:
   $P(k, j)$, for $k = 0, 1, ..., 4095$ and for $j = 0, 1, ...,127$
   The first column of this binary sequence corresponds to the first OFDM symbol following the preamble of the PHY Link.
2) Exclude the rows corresponding to excluded subcarriers, continuous pilots, and PHY Link from the two-dimensional array $P(k, j)$ to give an array $Q(k, j)$. The number of rows of the resulting array is N_I and the number of columns is 128.
3) Pass this two-dimensional binary array $Q(k, j)$ through the frequency de-interleaver and then the time de-interleaver, with each column treated as an OFDM symbol. After the 128 columns of the pattern have been input into the interleaver, re-insert the first *DS_TmIntrlv* columns, where *DS_TmIntrlv* is the depth of the time interleaver. This is equivalent to periodically extending $Q(k, j)$ along the dimension *j* and passing (128 + *DS_TmIntrlv*) columns of this extended sequence through the frequency de-interleaver and the time de-interleaver.
4) Discard the first *DS_TmIntrlv* symbols coming out of the time de-interleaver and collect the remaining 128 columns into an array to give the binary two-dimensional array $D(k, j)$ of size (N_I × 128).

For bit loading the CLT accesses the appropriate column *j* of the binary pattern bit $D(k, j)$ together with the appropriate bit loading profile $C_i(k, j)$. If the value of the bit $D(k, j)$ is 1, the CLT skips this subcarrier *k* and moves to the next subcarrier. This subcarrier is included as a placeholder for a scattered pilot that will be inserted in this subcarrier location after interleaving. After each symbol the column index *j* has to be incremented modulo 128.

The CLT uses this binary two-dimensional array $D(k, j)$ of size (N_I × 128) in order to do bit-loading of OFDM subcarriers, as described earlier in this subclause.

The corresponding operation in the CNU is de-mapping the QAM subcarriers to get Log-Likelihood-Ratios (LLRs) corresponding to the transmitted bits. This operation, described below, is much simpler than the mapping operation in the transmitter.

The scattered pilots and data subcarriers of every received symbol are subjected to frequency and time de-interleaving. The scattered pilots have to be tagged so that these can be discarded at the output of the time and frequency de-interleavers. This gives N_I subcarriers for every OFDM symbol. The CNU accesses theses N_I de-interleaved subcarriers together with the bit-loading pattern $C_i(k)$ to implement the de-mapping of the QAM subcarriers into LLRs. If the subcarrier *k* happens to be a scattered pilot, then this subcarrier, as well as the corresponding value $C_i(k)$, is skipped and the CNU moves to the next subcarrier (k +1).

101.4.2.8.4 FCP calculation

The FCP calculation is supplied by the Symbol Mapper via a function call UpdateFCP. The Symbol Mapper resets the bit counter, FCPbitCnt, at the start of each downstream frame and increments it for every bit processed in the frame. On the first transition of burstStart to TRUE from the PMA_UNITDATA.request after the start of a new frame the Symbol Mapper calls the UpdateFCP function with the counter value. The UpdateFCP function calculates the next (new) FCP value based on the supplied value, the *DS_Frame_Data_Load* (see 100.2.6.1) and *FEC_DS_CodeWordSize* (see 101.3.2.5.2) per Equation (101–12). The UpdateFCP function shall complete and pass the new FCP value to the PHY Link with sufficient time for insertion in the PHY Link frame currently being processed.

$$\text{FCP} = (\text{FCPbitCnt} + \text{DS\_Frame\_Data\_Load}\ ) \bmod \text{FEC\_DS\_CodeWordSize} \qquad (101\text{–}12)$$

101.4.2.8.5 Variables

DS_Frame_Data_Load
see 100.2.6.1

101.4.2.8.6 Constants

FEC_DS_CodeWordSize
see 101.3.2.5.5

101.4.2.8.7 Counters

FCPbitCnt
This counter is reset to zero at the beginning of each downstream frame and is incremented for each bit that is transfered from the Symbol Mapper to the Time Interleaver function.

101.4.2.8.8 Functions

UpdateFCP
This function Calculates the next FCP value per Equation (101–12)

101.4.2.9 Time and frequency interleaver

101.4.2.9.1 Overview

The CLT first applies a time interleaver to all N_I subcarrier)see Equation (101–10)) in a group of *DS_TmIntrlv* OFDM symbols. The CLT then subjects these N_I × *DS_TmIntrlv* subcarriers to frequency interleaving. There is a single Time and Frequency interleaving function per OFDM channel for the MAC data path.

101.4.2.9.2 Time interleaving

The CLT shall time interleave after OFDM symbols have been mapped to QAM constellations and before they are frequency interleaved.

The time interleaver is a convolutional interleaver that operates at the OFDM subcarrier level. If the depth of the interleaver is *M*, then there are *M* branches, as shown in Figure 101–20.

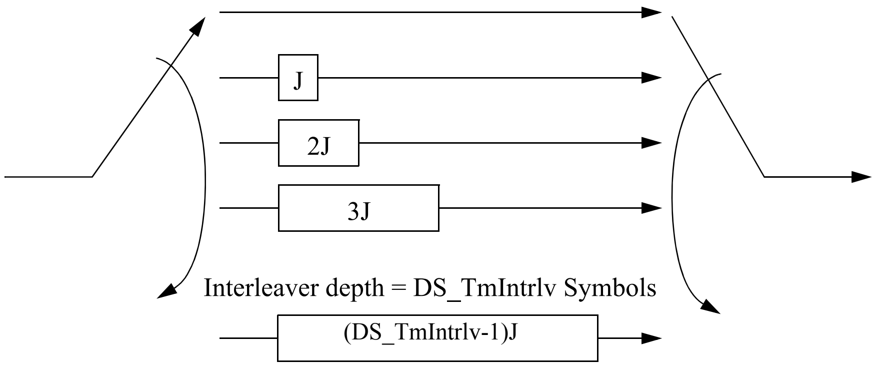


Figure 101–20—Time interleaver structure

The CLT shall support values of *DS_TmIntrlv* of from 1 to 32 (see 101.4.2.9.5).

Each branch is a delay line; the input and output will always be connected to the same delay line. This delay line will be clocked to insert a new subcarrier's data into the delay line and to extract a subcarrier's data from the delay line. Next, the commutator switches the input and the output to the next delay line in the direction shown by the arrows in Figure 101–20. After the delay line with the largest delay, the commutator will move to the delay line with zero delay.

The lowest frequency subcarrier of an OFDM symbol always goes through the branch with zero delay. Then the commutator switch at input and the corresponding commutator switch at output are rotated by one position for every new subcarrier.

The value of *J* is given by the following equation:

$$J = \left\lceil \frac{N_I}{DS\_\mathrm{TmIntrlv}} \right\rceil \tag{101–13}$$

Where, N_I (see Equation (101–10)) is the number of data subcarriers and scattered pilots in an OFDM symbol.

If N_I were not divisible by *DS_TmIntrlv*, all of the branches would not be filled. Therefore, "dummy subcarriers" are added to the symbol to make the number of subcarriers equal to a multiple of *DS_TmIntrlv*. The number of dummy subcarriers is given by:

$$J \times DS\_TmIntrlv - N_I \tag{101–14}$$

The dummy subcarriers are added for definition purposes only; at the output of the interleaver these dummy subcarriers are discarded.

101.4.2.9.3 Frequency interleaving

The CLT shall perform frequency interleaving after time interleaving; subcarriers containing continuous pilots, excluded subcarriers, or PHY Link data are not frequency interleaved.

The frequency interleaver works on individual OFDM symbols. Each symbol to be interleaved consists of N_I subcarriers indexed from *0* to N_I*-1* in ascending frequency order. These N_I subcarriers are made up of N_D data subcarriers and N_S scattered pilot placeholders. Although N_D and N_S are not the same for every symbol, the value of N_I is a constant for all OFDM symbols in the downstream frame for a given system configuration.

Conceptually, frequency interleaving of each individual OFDM symbol is performed using memory arranged in a 2-D store comprising 2^L rows and *K* columns where *L* and *K* are chosen depending on the size of the FFT used for creating the OFDM symbols. If the number of data subcarriers and scattered pilots in the OFDM symbol is N_I, then the number of columns, *K*, is given by the following equation:

$$K = \left\lceil \frac{N_I}{127} \right\rceil \qquad (101–15)$$

If N_I is not an integer multiple of 2^L, then the last column will only be partially filled during the frequency interleaving process. The number of data subcarriers in the last column, *C*, is given by:

$$C = N_I - 2^L(K-1) \qquad (101–16)$$

The frequency interleaver follows the following process; note that rows are numbered 1 to 2^L-1, and columns are numbered from 0 to *K-1*:

1) Write successive consecutive subcarriers into the 2-D store in the row given by the *L* bit CRC (cyclic redundancy check) value of each *L* bit row address (Figure 4).
2) Rotate the subcarriers in each row written by the same L bit CRC value of the row address modulo the number of columns in that row (either modulo *K* for a row below *C* or modulo *K*-1 for row *C* and higher) using a right circular shift (Figure 101–21a).
3) Rotate the subcarriers in each column by the *L* bit CRC value of [*K*-1 minus the column address] using a downward circular shift (Figure 101–21b). Note that the last column *K*-1 with a CRC value of 0 is not rotated.
4) Read the subcarriers out of the 2-D store column-wise from row 0, column 0 to row *C*-1, column *K*-1.

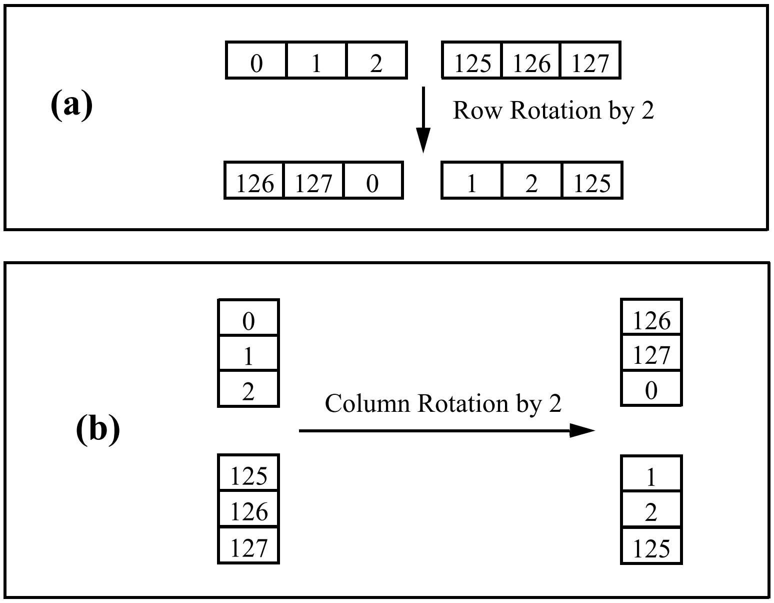


Figure 101–21—Rotation of subcarriers: a) Horizontal rows b) Vertical columns

Assume that the input subcarriers of the OFDM symbol are initially arranged into the 2-D store in sequential order column-wise from row 0, column 0 to row *C*-1, column *K*-1. The above procedure relocates each sequential input subcarrier number in row *r*, column *c* into a permuted output subcarrier number in the 2-D store in that position in row *r*, column *c* as *sc(r,c)* given by:

$$sc(r,c) = sc_0[(r - CRC(K-1-c))mod2^L] + (c - (r - CRC(K-1-c)mod2^L))modM \qquad (101\text{–}17)$$

where $M = K$, for $(r - CRC(K - 1 - c))mod2^L < C$

otherwise $M = K\text{-}1$

$sc(r,c) \in [0,1,\ldots,N_{I-1}]$ and $sc_0[n]$ is defined as an array of 2^L elements where each element contains the cumulative number of subcarriers previously written into the 2-D store prior to writing into the permuted output row *n*. Note that if the last column contains fewer subcarriers than 2^L, the cumulative value in $sc_0[n]$ takes into account those previously written permuted output rows that were shorter by one subcarrier (i.e. those prior row addresses that were greater than or equal to *C*, the number of subcarriers in the last column).

The structure of the two-dimensional store is shown in Figure 101–22.

.

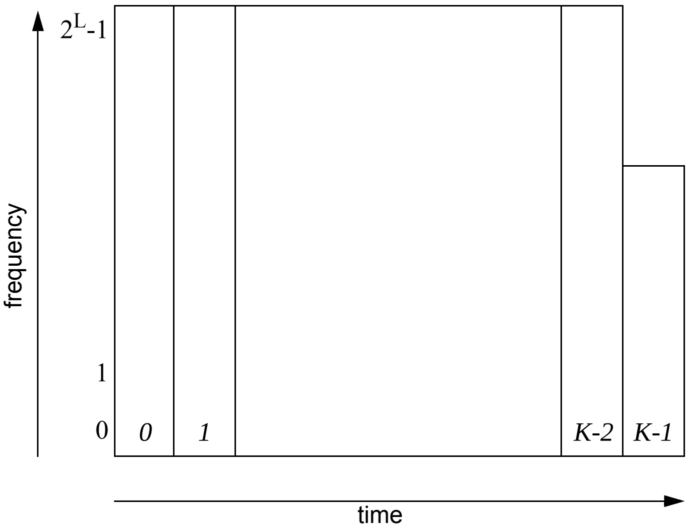


Figure 101–22—Two-dimensional store block structure

The m-stage linear feedback shift register for calculating the CRC of each row address is defined using a generator polynomial of degree $m = L$ in the finite (Galois) field GF[2]:

$$G(X) = g_mX^m + g_{m-1}X^{m-1} + g_{m-2}X^{m-2} + \ldots + g_2X^2 + g_1X^1 + g_0$$

where the coefficients g_m corresponding to the feedback taps of the linear feedback shift register are chosen such that the resulting generator polynomial is primitive (i.e. if the polynomial is prime and cannot be factored, and if it is a factor that evenly divides X^N+1, where $N = 2^m-1$). This guarantees that each L bit address for the 2^L rows is unique and the CRC values span the entire set of 2^L, L-bit addresses.

The linear feedback shift register implementing the generator polynomial is shown in Figure 101–23.

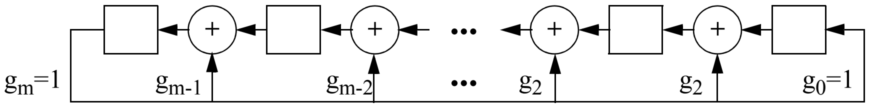


Figure 101–23—Row address linear feedback shift register

Calculation of the CRC c_{m-1}, c_{m-2}, …, c_1, c_0 for each row address b_{m-1}, b_{m-2}, …, b_1, b_0 using this linear feedback shift register structure is shown in Figure 4.

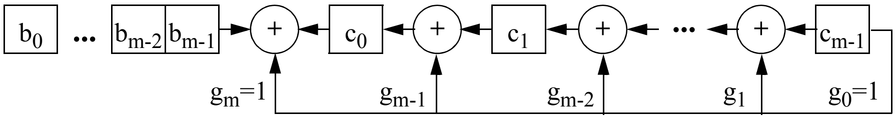


Figure 101–24—CRC calculation using a linear feedback shift register

De-interleaving is accomplished by reversing the interleaving process described above. Each symbol to be de-interleaved consists of N_I subcarriers indexed from 0 to N_{I-1} in ascending frequency order. Assume that the input subcarriers of the interleaved OFDM symbol are arranged into the 2-D store in sequential order column-wise from row 0, column 0 to row 2^L, column C.

The frequency de-interleaver performs the following process to reverse the interleaving process; note that rows are numbered from 0 to 2^L-1, and columns are numbered from 0 to K-1:

1) Write the subcarriers into the 2-D store column-wise from column 0, row 0 to column K-1, row C.
2) Rotate the subcarriers in each column by the L bit CRC value of [K-1 minus the column address] using an upward circular shift (reverse of Figure 101–23b). Note that the last column K-1 with a CRC value of 0 in not rotated.
3) Rotate the subcarriers in each row written by the same L bit CRC value of the row address modulo the number of columns in that row (either modulo K for a row below C or modulo K-1 for row C and higher) using a left circular shift (reverse of Figure 101–23a).
4) Read the subcarriers out of the 2-D store row-wise in the row order given by the L bit CRC value of each L bit row address skipping the last column at or beyond row C.

101.4.2.9.4 Interleaving impact on continuous pilots, scattered pilots, PHY Link and excluded spectral region

The CLT interleaves the subcarriers that are tagged to act as placeholders for scattered pilots. The actual BPSK modulation to these placeholder subcarriers is applied after interleaving as described in 101.4.2.9.2 and 101.4.2.9.3.

The CLT retains a reference pattern for inserting scattered pilot placeholders prior to interleaving. Since the scattered pilot pattern repeats every 128 symbols, this pattern is a (N_I *x 128*) two-dimensional bit pattern. A value of one in this bit-pattern indicates the location of a scattered pilot. The CLT inserts data subcarriers where this reference pattern has a zero and scattered pilot placeholders where this pattern has a one.

This reference pattern may be derived from the following procedure:

1) In the time-frequency plane, create a two-dimensional bit-pattern of zeros and ones from the transmitted "diagonal" scattered pilot patterns described in 101.4.2.6.1. This pattern has a periodicity of 128 symbols and has a value of one for a scattered pilot location and zero otherwise. Let the time axis be horizontal and the frequency axis vertical.
2) Delete all horizontal lines containing continuous pilots, excluded subcarriers, and PHY Link from the above mentioned two-dimensional bit pattern; note that some scattered pilots could coincide with continuous pilots. These locations are treated as continuous pilot locations.
3) Send the resulting bit-pattern through the frequency de-interleaver and the time de-interleaver in succession. This will give another two-dimensional bit pattern that has a periodicity of 128 symbols. The appropriate 128-symbol segment of this bit-pattern is chosen as the reference bit pattern referred to above.

The synchronization of the scattered pilot pattern to the PHY Link preamble, as described in Figure 101–18 uniquely defines the 128-symbol segment that is used as the reference pattern.

Note that the number of OFDM subcarriers that are interleaved does not change from symbol to symbol. The insertion of continuous pilots, PHY Link, and excluded regions happens after both time and frequency interleaving. Interleaving is independent of individual subcarrier modulation and the modulation pattern of these data subcarriers may change from symbol to symbol.

101.4.2.9.5 Variables

DS_TmIntrlv|

TYPE: 5-bit integer

This variable indicates the number of time interleaved OFDM symbols in the downstream direction. The value is between 1 and 32.

101.4.2.10 Pilot insertion

Continuous and scattered pilots shall be BPSK modulated using the pseudo-random sequence generated by the 13-bit linear feedback shift register, illustrated in Figure 101–25 with polynomial $(x^{13}+x^{12}+x^{11}+x^{8}+1)$ and described below.

This linear feedback shift register is initialized to all ones at the $k=0$ index of the discrete Fourier transform defining the OFDM signal (refer to 101.4.2.11). It is then clocked after every subcarrier of the IDFT. If the subcarrier is a pilot (scattered or continuous), then the BPSK modulation for that subcarrier is taken from the linear feedback shift register output.

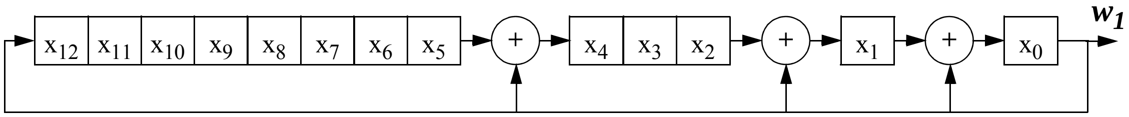


Figure 101–25—13-Bit Linear Feedback Shift Register for the pilot modulation Pseudo-Random Sequence

Let the output of the linear feedback shift register be w_k then BPSK modulation used for the pilot is:

$w_k = 0$: BPSK constellation point = $1 + j0$

$w_k = 1$: BPSK constellation point = $-1 + j0$

101.4.2.10.1 Pilot boosting

The CLT shall multiply the real and imaginary components of continuous and scattered pilots by a real-valued number such that the amplitude of the continuous and scattered pilots is twice the root-mean-square value of the amplitude of other subcarriers of the OFDM symbol. That is, continuous and scattered pilots are boosted by approximately 6 dB with reference to other subcarriers.

101.4.2.11 Inverse Discrete Fourier Transform (IDFT)

The CLT OFDM and CNU OFDMA signals are assembled in the frequency domain using 4096 subcarriers per OFDM/OFDMA channel. The signal is composed of: MAC data subcarriers, scattered pilots, continuous pilots, PHY Link subcarriers, zero-bit-value subcarriers, and excluded subcarriers (zero valued into the IDFT).These OFDM/OFDMA signals are described in IDFT Equation (101–18).

$$x(i) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} X(k) \exp\left(j \frac{2\pi i\left(k - \frac{N}{2}\right)}{N} \right), \text{ for } i = 0, 1, \ldots, (N-1) \qquad (101\text{–}18)$$

Where: *N* equals 4096, *X(0)* is the lowest frequency component and *X(N-1)* is the highest frequency component.

The resulting time domain discrete signal, *x(i)*, is a baseband complex-valued signal, sampled at 204.8 Msamples per second resulting in a subcarrier spacing of 50 kHz.

Once the CNU detects the downstream PHY Link and receives the *DS_FreqCh1* variable (see Table 101–1), the CNU knows the location of *k* = 0.

101.4.2.11.1 Variables

DS_FreqCh(n)
See . 100.2.7.3

101.4.2.12 Cyclic prefix and windowing

CLT and CNU cyclic prefix (CP) and windowing processing begins with the *N*-point output of the IDFT:

$\{x(0), x(1), \ldots, x(N-1)\}$

The variable *DSNcp* represents the provisioned duration, in OFDM clocks, of the downstream cyclic prefix parameter (see Table 101–11) for the CLT. The *DSNcp* samples at the end of this *N*-point IDFT are copied and prepended to the beginning of the IDFT output to give a sequence of length (*N*+*DSNcp*):

$\{x(N-DSNcp), x(N-DSNcp+1), \ldots, x(N-1), x(0), x(1), \ldots, x(N-1)\}$

The variable *DSNrp* represents the provisioned duration, in OFDM clocks, of the downstream windowing parameter (see Table 101–12 for the CLT. The *DSNrp* samples at the start of the *N*-point IDFT are copied and appended to the end of the IDFT output to give a sequence of length (*N*+*DSNcp*+*DSNrp*):

$\{x(N-DSNcp), x(N-DSNcp+1), \ldots, x(N-1), x(0), x(1), \ldots, x(N-1), x(0), x(1), ..., x(DSNrp-1)\}$

Let this extended sequence of length (*N* + *DSNcp* + *DSNrp*) be defined as:

$$\{y(i),\ i=0, 1, \ldots, (N + DSNcp + DSNrp-1)\} \qquad (101\text{–}19)$$

DSNrp samples at both ends of this extended sequence are subject to tapering. This tapering is achieved using a raised-cosine window function; a window is defined to be applied to this entire extended sequence. This window has a flat top and raised-cosine tapering at the edges, as shown in Figure 101–26.

The window function *w(i)* is symmetric at the center; therefore, only the right half of the window is defined in the following equation:

$$w\left(i + \frac{N + DSNcp + DSNrp}{2}\right) = 1.0 \qquad (101\text{–}20)$$

$$for\ i = 0, 1, \ldots, \frac{(N + DSNcp + DSNrp)}{2} - 1$$

$$w\left(i + \frac{N + DSNcp + DSNrp}{2}\right) = \frac{1}{2}\left(1 - \sin\left(\frac{\pi}{\alpha(N + DSNcp)}\left(i - \frac{N + DSNcp}{2} + \frac{1}{2}\right)\right)\right) \qquad (101\text{–}21)$$

$$\text{for } i = \left(\left(\frac{N + DSNcp - DSNrp}{2}\right), \ldots, \left(\frac{N + DSNcp + DSNrp}{2} - 1\right)\right)$$

Here,

$$\alpha = \frac{DSNrp}{N + DSNcp} \qquad (101\text{–}22)$$

defines the window function for (*N*+*DSNcp*+*DSNrp*)/2 samples. The complete window function of length (*N* + *DSNcp* + *DSNrp*) is defined using the symmetry property as:

$$w\left(\frac{N + DSNcp + DSNrp}{2} - i - 1\right) = w\left(\frac{N + DSNcp + DSNrp}{2} + 1\right), \qquad (101\text{–}23)$$

$$\text{for } i = 0, 1, \ldots, \frac{(N + DSNcp + DSNrp)}{2} - 1$$

This yields a window function (or sequence): {*w(i), i=0, 1, …, (N + DSNcp + DSNrp – 1)*}. The length of this sequence is an even-valued integer.

The above window function is applied to the sequence {*y(i)*}:

$$z(i) = y(i)\, w(i), \quad for\ i = 0, 1, \ldots, (N + DSNcp + DSNrp - 1) \qquad (101\text{–}24)$$

Each successive set of *N* samples at the output of the IDFT yields a sequence *z(i)* of length *(N + DSNcp + DSNrp).* Each of these sequences is overlapped at each edge by *DSNrp* samples with the preceding and following sequences, as shown in the last stage of Figure 101–26. Overlapping regions are added together.

To define this "overlap and add" function mathematically, consider two successive sequences $z_1(i)$ and $z_2(i)$. The overlap and addition operations of these sequences are defined using the following equation:

$$z_1(N + DSNcp + i) + z_2(i), \quad for\ i=0, 1, \ldots, DSNrp - 1 \qquad (101\text{–}25)$$

That is, the last *DSNrp* samples of sequence $z_1(i)$ are overlapped and added to the first *DSNrp* samples of sequence $z_2(i)$.

The length of the extended OFDM symbol is *(N + DSNcp + DSNrp)* samples. Of this, (*DSNrp* / 2) samples are within the preceding symbol, and (*DSNrp* / 2) samples are within the following symbol. This yields a symbol period of (*N* + *DSNcp*) samples.

The process of cyclic prefix and windowing is illustrated in Figure 101–26. The CLT shall use one of the permissible values for *DSNcp* and for *DSNrp* in the downstream direction given in Table 101–11 and Table 101–12, respectively. CP and Window sizes shall be selected such that the *DSNrp* value is less than the CP value.

Table 101–11—Size of cyclic prefix (*DSNcp*), downstream direction

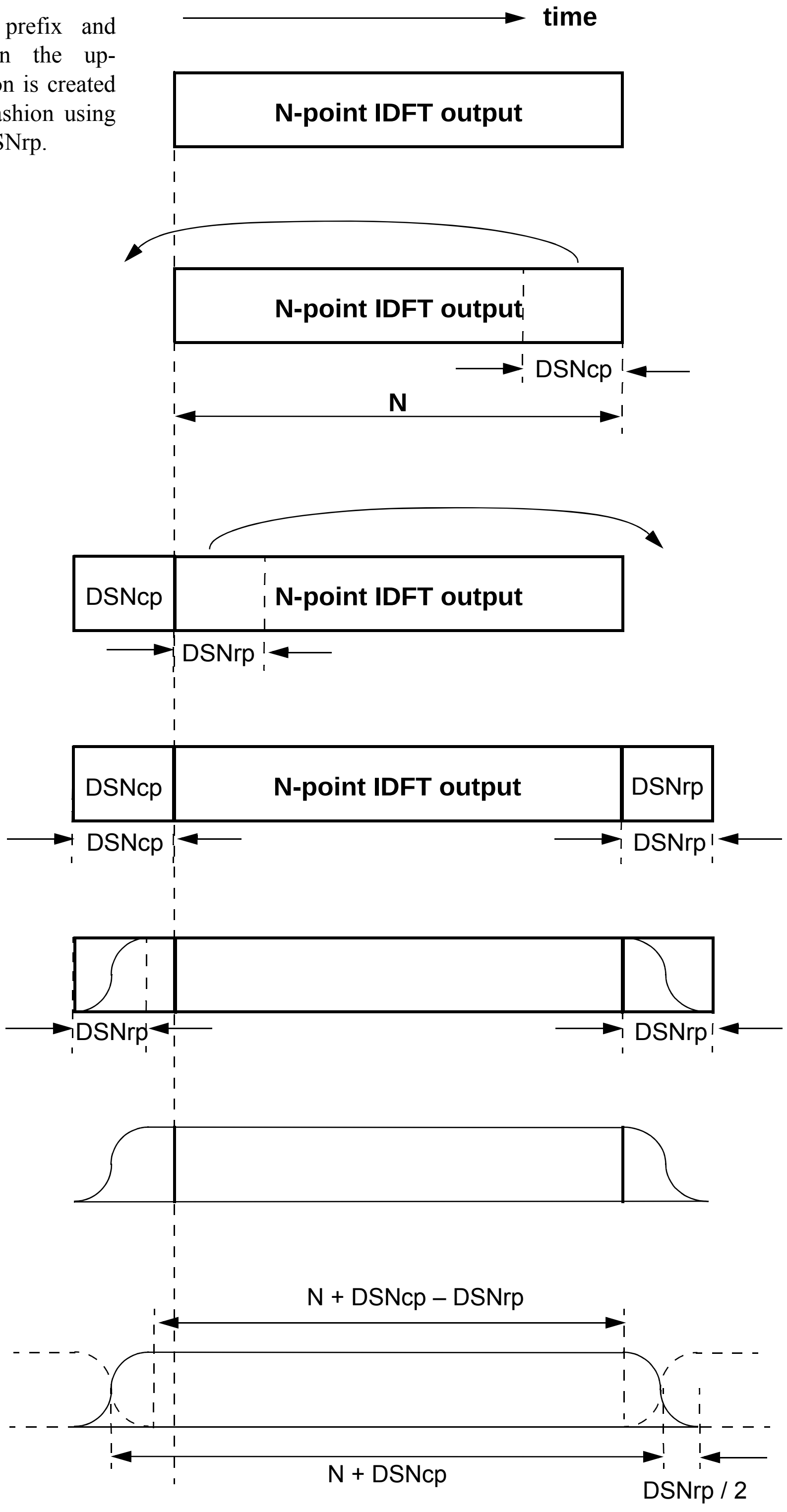
DSNcp[a]	**[μs]**
256	1.25
512	2.5
768	3.75

[a] *DSNcp* represents sample clock period, equal to 1/ 204.8 MHz

Table 101–12—Size of OFDM Window (*DSNrp*), downstream direction

DSNrp[a]	**[us]**
0	0.0000
64	0.3125
128	0.6250
192	0.9375
256	1.2500

[a] *DSNrp* represents sample clock period, equal to 1/ 204.8 MHz



Note: cyclic prefix and Windowing in the upstream direction is created in a similar fashion using USNcp and USNrp.

Figure 101–26—Cyclic prefix and windowing algorithm

101.4.2.12.1 Variables

DSNcp

TYPE: 4-bit binary

This variable controls the size of the cyclic prefix in the downstream direction per the enumeration below.

bit 3 2 1 0
0 1 0 0 = reserved
0 0 1 1 = 768 samples
0 0 1 0 = 512 samples
0 0 0 1 = 256 samples
0 0 0 0 = reserved
samples refer to OFDM clock (204.8 MHz)

DSNrp

TYPE: 3-bit binary

This variable controls the size of the windowing function in the downstream direction per the enumeration below.

bit 2 1 0
1 1 x = reserved
1 0 1 = 256 samples
1 0 0 = 192 samples
0 1 1 = 128 samples
0 1 1 = 64 samples
0 0 1 = reserved
0 0 0 = 0 samples (windowing disabled)
samples refer to OFDM clock (204.8 MHz)

101.4.2.13 OFDM channel Requirements

Table 101–13 enumerates multiple OFDM channel operational requirements.

Table 101–13—Multiple OFDM channel requirements

Item	Requirement
OFDM channel 1 configuration	OFDM channel 1 is always be enabled. OFDM channel 1 processes subcarriers for data as well as the PHY Link.
OFDM channel 2, 3, 4, 5 configuration	OFDM channels 2, 3, 4, or 5 may be enabled or disabled for operation. OFDM channels are enabled in ascending order: e.g. enable OFDM channel 2 before enabling OFDM channel 3, enable OFDM channel 3 before enabling OFDM channel 4, enable OFDM channel 4 before enabling OFDM channel 5.
OFDM channel frequency placement	An OFDM channel may be configured for operation in any portion of the downstream Frequency Band as per Table 100-1

Table 101–13—Multiple OFDM channel requirements *(continued)*

Item	Requirement
OFDM channel subcarrier indexing relation to RF frequency	OFDM channel 1: 0 to 4095 OFDM channel 2: 4096 to 8191 OFDM channel 3: 8192 to 12287 OFDM channel 4: 12288 to 16383 OFDM channel 5: 16384 to 20479
OFDM channel subcarrier frequency ordering	RF frequency correlates to subcarrier index; i.e., the lower the subcarrier index, the lower the RF frequency.
Minimum encompassed Spectrum	The minimum encompassed spectrum of any enabled OFDM channel is 22 MHz as per Table 100-1.
Encompassed spectrum overlap	The encompassed spectrum of any enabled OFDM channel does not overlap with that of any other enabled OFDM channel.
Adjacent OFDM channel placement	The CLT transmitter permits placement of the edge subcarrier of an OFDM channel's encompassed spectrum immediately adjacent to the edge subcarrier of another OFDM channel's encompassed spectrum without any frequency guard band.

101.4.3 Upstream PMA transmit function

101.4.3.1 Overview

101.4.3.2 Time and frequency synchronization

101.4.3.3 Frame Timing

The Frame Timing function is reset by the PHY Link during link auto-negotiation. The state machine of Framing Timing implemented the RB Superframe structure timing as per 101.4.3.3.1.

101.4.3.3.1 RB Superframe configuration and burst transmission

The upstream Superframe shall be composed of the Probe Period followed by 256 OFDMA symbols. Each Probe Period is six OFDMA symbols in duration. An RB Frame is one Resource Block column (i.e., one column of Resource Blocks over the entire upstream spectrum). Each Resource Block is composed of one subcarrier and has a duration of either 8 or 16 symbols and is set using the *RBsize* variable. Changing the Resource Block duration results in a network restart. The superframe structure is illustrated in Figure 101–27.

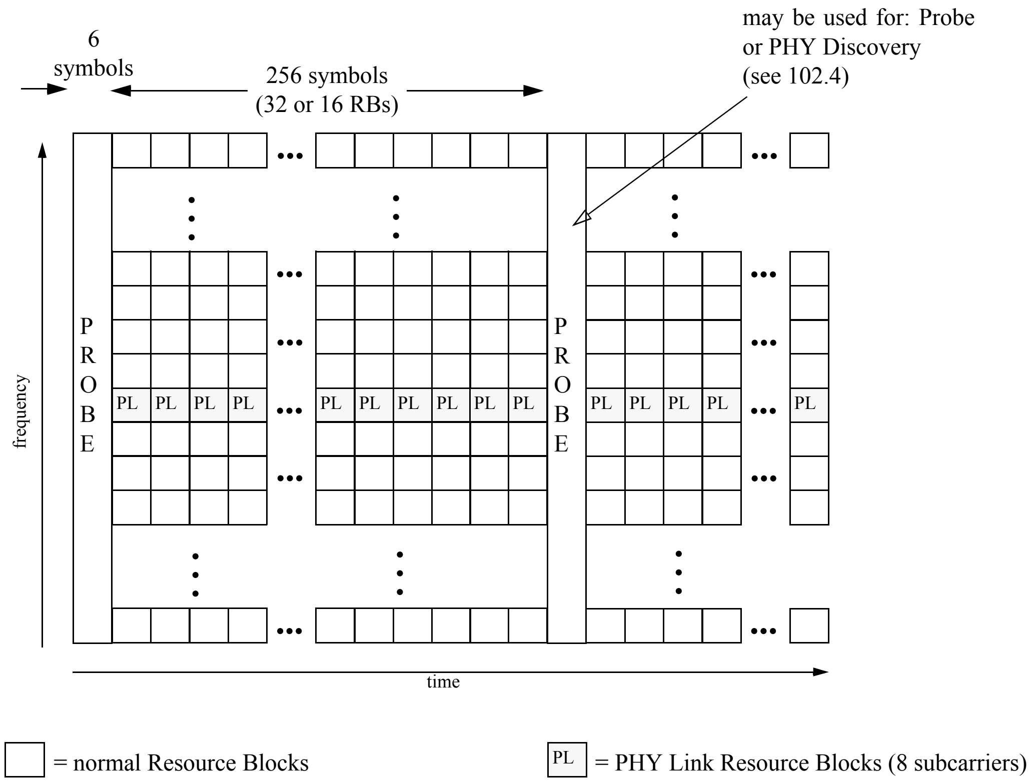


Figure 101–27—US superframe structure

101.4.3.3.2 OFDMA transmission burst start

An OFDMA transmission shall start with a Type 2 resource block followed by four contiguous subcarriers which include the start burst marker (see 101.4.3.9).

101.4.3.3.3 OFDMA transmission internal to a burst

In general resource blocks internal to a burst may be of Type 0, Type 1 or Type 2 as determined by the provisioned Pilot pattern (see 101.4.3.7). An OFDMA transmission may straddle excluded subcarriers, unused subcarriers and the Probe Period. For example if the transmission burst start and stop markers straddle an exclusion band or if the burst start marker occurs in one RB Frame and the stop marker occurs in a subsequent RB Frame. In such cases, where the OFDMA transmission crosses a band edge (anywhere an excluded subcarrier is adjacent to an active subcarrier), the active subcarrier immediately adjacent to the band edge shall be of Type 2.

101.4.3.3.4 OFDMA transmission burst end

An OFDMA transmission shall end with a Type 2 resource block preceded by four contiguous subcarriers which include the stop burst marker (see 101.4.3.9).

101.4.3.3.5 Variables

RBsize

TYPE: boolean

This variable determines the size of the upstream Resource Blocks. When *RBsize* is TRUE then Resource Block size is 16 symbols, When *RBsize* is FALSE then Resource Block size is 8 symbols.

RBlen(*RBsize*)

TYPE: integer

This integer represents the number of symbols for the configured resource block size. See Table 101–1

Value: 8 when *RBsize* is 0, 16 when *RBsize* is 1.

SCLK

TYPE: boolean

This Boolean is TRUE on every negative edge of a clock that is synchronized to the period of the Extended OFDM symbol time comprised of the 20 μs useful symbol time plus the cyclic prefix time *USNcp*.

RBSF_reset

TYPE: boolean

This boolean variable is used by the PHY Link to reset the Frame Timing state. A positive transition from value FALSE to value TRUE will cause the state machine to reset to the beginning of the RB Superframe on SCLK.

Probe_start

TYPE: boolean

This boolean transitions from the value of FALSE to TRUE for the first symbol of the Probe region. At the end of the first symbol of the probe region, the value transitions to FALSE until the first symbol of the next RB Superframe.

RB_Frame_start

TYPE: boolean

This boolean transitions from the value of FALSE to TRUE for the first symbol of each RB Frame in the RB Superframe as related to *RBsize*. The value is FALSE for all probe region symbols and for symbol 2 through *RBsize* for each RB Frame.

SYMcount

TYPE: integer

This is variable is used for counting the symbols of the RB Superframe.

RBmod

TYPE: integer

This variable is used to set the symbols per RB Frame upon reset.

101.4.3.3.6 State Diagram

The CLT PMA shall implement the frame timing process as shown in Figure 101–28.

In case of any discrepancy between state diagrams and the descriptive text, the state diagrams prevail.

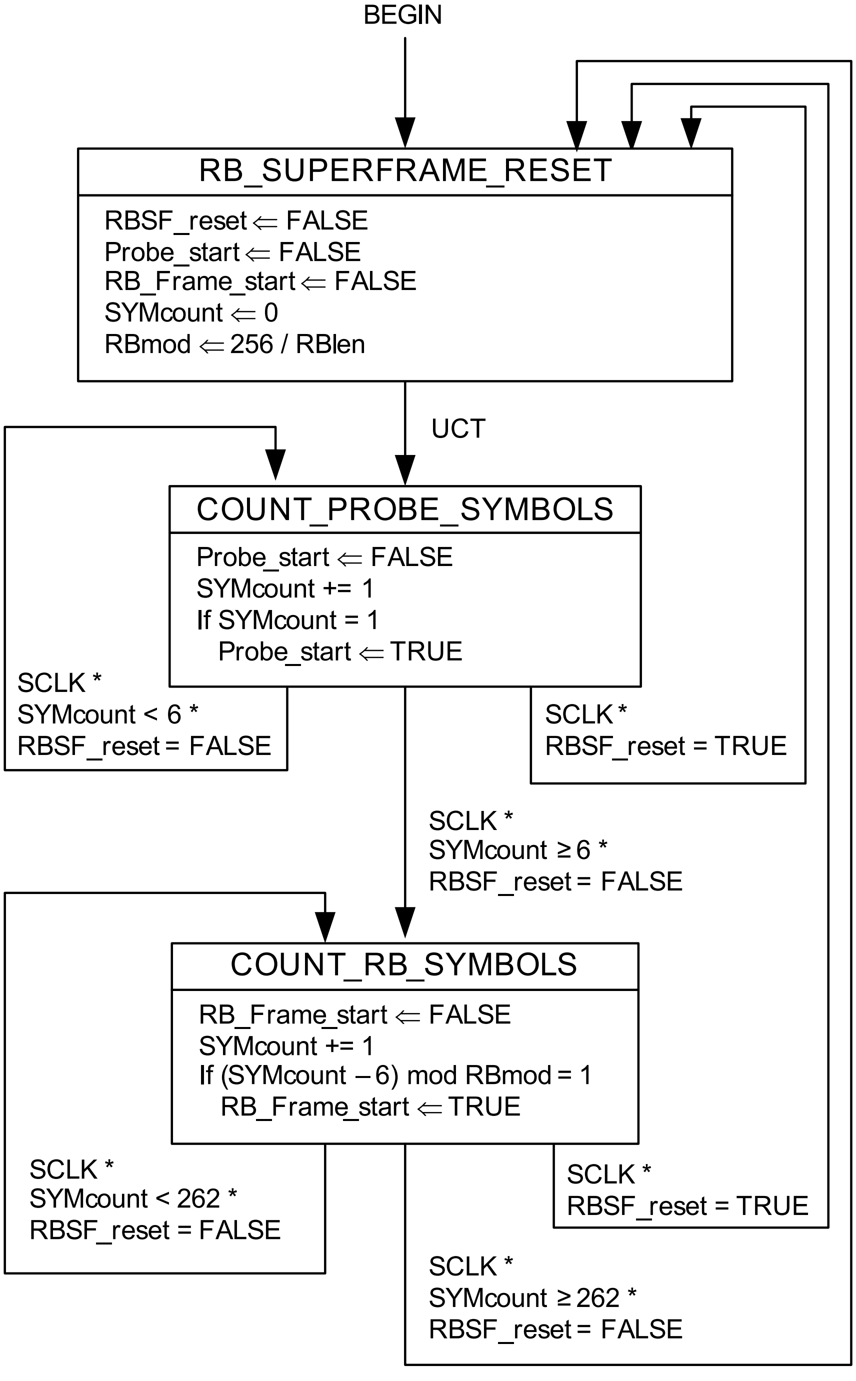


Figure 101–28—Framing Timing

101.4.3.4 Subcarrier configuration and bit loading

Each subcarrier in the OFDMA channel is configured using the *US_ModTypeSC(n)* (where $0 \leq n \leq 4095$) or TypeN_Repeat/TypeN_Start (N = 1 or 2) variables. These variables allow the Phy to configure each subcarrier to be nulled, to have a specific bit loading (such as 512-QAM or 1024-QAM), or to be Excluded and to define the pilot pattern to be used in upstream transmissions. Subcarriers that are not configured as excluded are active subcarriers. Subcarrier configuration in an EPoC OFDM channel of 192 MHz shall conform to the rules outlined in Table 101–14.

All devices in an EPoC network share the same upstream subcarrier configuration and bit loading including nulled subcarriers, pilot pattern, bit loaded subcarriers and excluded subcarriers.

Table 101–14—Upstream subcarrier configuration rules

Parameter	Limit	Unit
Minimum number of combined active and nulled subcarriers for Probe	180	subcarriers
Minimum OFDMA channel guard band	1	MHz
Max OFDMA channel encompassed spectrum	190	MHz

101.4.3.4.1 Nulled subcarriers

Nulled subcarriers do not carry MAC or PHY Link data but may be used as probes. Nulled subcarriers are not modulated except when being used as a Probe Symbol in the upstream direction (see 102.4.2).

101.4.3.4.2 Bit loaded subcarriers

When a subcarrier is used to carry MAC data it uses the modulation type of QPSK or 2^n-QAM, where $4 \leq n \leq 16$, assigned via the *US_ModTypeSC(n)* variables except when used as an upstream pilot (see 101.4.3.7), Probe, PHY Discovery Response (see 102.4.1), or burst marker (see 101.4.3.9).

There is at least one contiguous 10 MHz or greater band of active subcarriers in any single 192 MHz OFDM channel (see Table 100–11). This may include subcarriers intended as Pilots and PHY Link subcarriers. A 1 MHz guardband of excluded subcarriers above and below this 10 MHz creates a minimum width OFDM channel of 10 MHz encompassed spectrum.

101.4.3.4.3 Excluded subcarriers

EPoC devices shall not transmit energy into a subcarrier that has been excluded from the OFDM channel (i.e, excluded subcarriers have zero amplitude). Typically there is a band edge Exclusion Band at both the top and bottom of the OFDM channel and there may be up to 14 exclusion bands internal to a single 192 MHz OFDM channel.

101.4.3.4.4 Variables

US_ModTypeSC(n)

TYPE: 4-bit binary

This set of variables determine the modulation parameters for each of the 4096 upstream OFDM subcarriers. $0 < n < 4095$. Each variable controls one of the 4096 subcarriers that comprise the OFDMA channel, with *US_ModTypeSC(0)* controlling subcarrier zero, *US_ModTypeSC(1)* controlling subcarrier 1, etc. The assignment of bits to each modulation type is show below.

bit 3 2 1 0
1 1 1 1 = Excluded subcarrier
1 1 1 0 = 16384-QAM
1 1 0 1 = 8192-QAM
1 1 0 0 = 4096-QAM
1 0 1 1 = 2048-QAM
1 0 1 0 = 1024-QAM
1 0 0 1 = 512-QAM
1 0 0 0 = 256-QAM
0 1 1 1 = 128-QAM
0 1 1 0 = 64-QAM
0 1 0 1 = 32-QAM
0 1 0 0 = 16-QAM
0 0 1 1 = 8-QAM
0 0 1 0 = QPSK
0 0 0 1 = BPSK (Used for continuous pilots only)
0 0 0 0 = null (carries no data but used for Wideband Probing)

101.4.3.5 Probe generator

101.4.3.5.1 Variables

PrbEQ

TYPE: boolean

When this variable is TRUE the CNU transmits the probe symbol with equalization applied. When this variable is FALSE the CNU transmits the probe symbol without equalization applied.

PrbSkp

TYPE: integer

The value of this variable determines the number of subcarriers to be skipped in the probe symbol. The range of PrbSkp is from zero to seven.

PrbStrtSC

TYPE: integer

The value of this variable determines the starting subcarrier to be used in the probe symbol. The range of PrbStrtSC is from zero to seven.

RcvPrbID

TYPE: set of 15-bit Integers

This set of variables contains the received set of PrbID variables from the most recent downstream PHY Link frame. PrbID. When US_FrmCnt = RF_ID the values in this set replaces the values in the ActPrbID set. (see Figure 102–2).

101.4.3.6 Upstream Symbol mapper

The US symbol mapper consists of an idle loop process and a fill process. The idle loop process is initialized when the upstream profile is configured and set to align with the start of the RB Superframe. An RB Superframe is comprised of a Probe Period (first six OFDMA symbols) followed by 32 (*RBsize* value “0”) or 16 (*RBsize* value “1”) RB Frames (see 101.4.2.5). The main task of the idle loop is to walk through all data bits in all data carrying resource elements (in each RB Frame) at the clock rate established by *US_DataRate* (see 100.2.6.2). The fill process, upon an assertion of the start of a burst from the PMA Client, will start filling at the resource block, resource element, and fill bit that is at the current walk point. This includes placing the start burst marker elements, filling bits in data resource elements and low density pilots as specified, creating the QAM map for the fill word bits, sensing the endBurst notification, padding to the end of the current resource block, encoding and placing the end burst marker elements and then terminating. Both processes are continuous loops. During the idle walk and fill processes, two-dimensional arrays (resource block size by total subcarriers) termed RB Frames are allocated and passed to the staging process when complete. RB Frames initially consist of null values that correspond to excluded subcarriers. Non-null values are inserted by the fill process when mapping burst markers and filling data resource elements, and when later processed by the Pilot Insert process fill "P" type pilots patterns (see 101.4.3.7). Null resource blocks are equivalent to excluded subcarriers and produce no energy on the corresponding OFDMA subcarrier after the IDFT process.

101.4.3.6.1 Variables

BITPOS

Type: integer

This is an integer used by the fill process to indicate the current bit position being filled in *FILLWORD*<>.

END

Type: enumerated type

This variable that ends the end state processing of the fill process, where:

"FALSE" indicates a burstStart = TRUE has not been received from received PMA_UNITDATA.request() primitive,

"PAD" indicates burstStart = TRUE has been received and the fill process is padding bits to the end of the current RB,

"TRUE" indicates that the fill process has reached the end of the last RB while padding.

FILLWORD<16:1>

Type: bit array

This an array that stages the bits to fill a resource element prior to mapping to QAM symbols.

FIRST

Type: boolean

This is used within the fill process to indicate status of placing the first bit passed in the function call. If TRUE, the passed first bit is used, otherwise the bit receive from the processed PMA_UNITDATA.request() is used.

FRB

Type: integer

This variable is used to determine the current resource block being processed in the fill process. The range of values if from 0 to 4095.

FRE

Type: integer

This variable is used to determine the current resource element being processed in the fill process. The range of values is from –1 to 16.

ICLK

Type: clock

This variable represents a clock running at *US_DataRate* (see 100.2.6.2).

IDLEBITS

Type: integer

This variable is used by the idle loop to increment through the bit loading of the current data carrying resource element.

IRB

Type: integer

This variable is used to determine the current resource block being processed in the idle process. The range of values if from –1 to 4095.

IRE

Type: integer
This variable is used to determine the current resource element being processed in the idle process. The range of values if from 0 to *RBlen*.

LBIT

Type: integer
This variable records the last bit filled in the current data resource element (QAM symbol) before mapping. The value can be from 1 to 15, where 1 represents the LSB. The value is set to the bit loading for the data RE and then decremented during the symbol mapper fill process for mapping data burst bits to the current data resource element. This value is reset for each new data resource element being filled. The value is incremented while filling bits and stops incrementing upon receiving an end of burst indication. This value is not incremented or reset when placing padding bits.

LRE

Type: integer
This variable records the last filled data resource element in the current resource block. The value can be from 1 to 16, where 1 represents the first resource element in the resource block (in time) and up to 8 or 16 (*RBlen*) representing the last resource element in the resource block (later in time). The value is reset for each new resource block being filled. The value is incremented to the current data resource element being filled and stops incrementing upon receiving an end of burst indication. This value is not incremented or reset when placing padding bits.

PILOT_MAP<4096>

Type: array of enumerated values
This array defines the pilot pattern use for Type 1 and Type 2 pilot patterns as defined in the upstream profile descriptor (see 101.4.3.7). The pilot pattern is fixed for the RB Superframe configuration and remains constant.   The enumerated type values are:
"EX": no Type 0, Type 1, or Type 2 pattern is configured for this subcarrier
"T0": this resource block, if used for data, will contains a Type 0 pattern (see 101.4.3.7)
"T1": this resource block, if used for data, will contains a Type 1 pattern (see 101.4.3.7)
"T2": this resource block, if used for data, will contains a Type 2 pattern (see 101.4.3.7)
"PHYLINK": this resource block is reserved for use by the PHY Link
All other enumerated type values are reserved. For any used resource element containing data and not containing either a start or end burst marker, this value will be used to set the *RB_Type* array element value for further processing by the interleaver and pilot insertion functions.

EDITORS NOTE (to be removed prior to publication): need to rationalize subcarrier use with EX and null subcarriers, Clause 45, and other descriptions. Should “interleaver” in the above be replaced with Symbol Mapper?

RB_Frame<4096, *RBlen*>

Type: Array of I_value and Q_value bin value pair.
This two-dimensional array holds the I and Q QAM symbol bin values that are passed from the symbol mapper to the staging function, then to the IDFT for transmission by the CNU. When first allocated, RB_Frame contains null values ("0") in all I and Q values. I and Q bin values are signed 16-bit integers. Index 4095:0 represents the total number of possible subcarriers in the upstream corresponding OFDMA channel, *RBlen* represents the number of resource elements in resource block. Subcarrier 0, RB number 1 (RB_Frame<0,1>) is reserved for special use in coordinating symbol mapper idle and fill processing: I_value of "0" indicates the *RB_Frame* has not transferred to the staging process, I_value of "1" indicates the *RB_Frame* has been passed to the staging process, Q_value is always "0".

RB_Type<4096>

Type: array of enumerated values

This array defines the use of each resource block, the values and descriptions are:

"EX": the resource block is excluded (no energy output from IDFT), all resource elements will contain null I and Q value pairs.

"T0": resource block in use, Type 0 pilot pattern (see 101.4.3.7)

"T1": resource block in use, Type 1 pilot pattern (see 101.4.3.7)

"T2": resource block in use, Type 2 pilot pattern (see 101.4.3.7)

"SM": resource block in use, contains a start marker pattern

"EM": resource block in use, contains an end marker pattern

All other enumerated type values are reserved. The use of this array indicates the availability and type of each used resource block to pilot insertion. A null (unused) resource block array element produces no energy output from the IDFT for the corresponding OFDMA subcarrier. (This is also true for excluded subcarriers).

RBlen(*RBsize*)

See 101.4.3.3.5.

101.4.3.6.2 Functions

Allocate_RB_Frame(*RB_Frame*, *RB_Type*)

This function allocates a new *RB_Frame* array with all I_value and Q_value bin pairs set to null "0" and a new *RB_Type* array with all enumerations set to "EX".

BITLOAD(*resource_element*, *resource block*)

This function returns the current bit loading capacity of the current data carrying resource element or low density pilot. Bit loading is based on the setting of US_ModTypeSC() (see Table 101-1) for this resource block (subcarrier) as well as resource elements containing low density pilots in Type 2 resource blocks (see 101.4.3.7).

FILL_PROCESS(*FIRSTRB*, *FIRSTBIT*)

This function calls the fill process to begin execution, at the resource block indicated by *FIRSTRB* index, with the *FIRSTBIT* indicating the first bit to be filled in MSB of the first available data resource element. See Figure 101.x.x.x.

Initialize_Pilot_Map()

This function initializes the *PILOT_MAP*<> array based on the profile descriptor information for indicating Type 0, Type 1, and Type 2 patterns. See Table 101-TBD and 101.4.3.7.

Map_End_Marker (*RB_number*, *Last_Bit*, *Last_RE*)

This function starts at the current *RB_number*. This function then constructs the end burst marker by encoding the *Last_Bit* and *Last_RE* information. See 101.4.3.9.3 for end marker encoding. The value of *Last_Bit* directly indicates fill position in the current resource element. A values of *Last_RE* ranges from “0” which indicates the first resource element of the resource block up to *RBlen* – 1.This function then places end burst marker elements in the resource block according to the *RBlen* of the Superframe. This function continues to increment resource blocks (subcarriers) and places the remaining end burst marker elements in the next usable resource block as indicated in *US_ModTypeSC(n)* setting in the US profile descriptor. See Table 101-TBD. For each resource block used for an end burst marker element, the corresponding entry in the *RB_Type* array is set to "EM". Excluded subcarriers and null subcarriers are skipped. The next usable resource block is defined as an *US_ModTypeSC(n)* value from binary 0001 (BPSK) to binary 1110 (16384-QAM) and not being used by the PHY Link.

If at any time this function increments beyond the last usable resource block in the current *RB_Frame*<> (highest usable subcarrier configured in the profile descriptor), It examines the I_value of *RB_Frame*<0,1>. If "0", this function sets the value to "1" and passes the *RB_Frame*<> array and RB_Type array to the staging function and allocates a new

RB_Frame<> array and *RB_Type* array use the Allocate_RB_Frame() function. If I_value of *RB_Frame*<0,1> is "1", the function skips passing to staging and allocations, assumes a new *RB_Frame*<> has been allocated, and increments to the first data carrying resource block and resource element in the new *RB_Frame*<>.

Map_Start_Marker (*RB_number*)

This function begins by placing the first resource block of a start burst marker in the current resource block *RB_number*, according to the *RBlen* of the Superframe (see 101.4.3.9). This function continues to increment resource blocks (subcarriers) and placing the remaining start burst marker elements in the next usable resource block(s) as indicated in *US_ModTypeSC(n)* setting in the US profile descriptor (defined as a *US_ModTypeSC(n)* value from binary 0001 (BPSK) to binary 1110 (16384-QAM) and not being used by the PHY Link). See Table 101-TBD. Excluded subcarriers and null subcarriers are skipped. After placing the last start burst marker resource block, this function returns the value in *RB_number*.

For each resource block used for a start burst marker element, the corresponding entry in the *RB_Type<RB_number>* array is set to "SM". The next usable resource block is defined as a *US_ModTypeSC(n)* value from binary 0001 (BPSK) to binary 1110 (16384-QAM) and not being used by the PHY Link.

If at any time this function increments beyond the last usable resource block in the current *RB_Frame* (highest usable subcarrier configured in the profile descriptor), It examines the I_value of *RB_Frame*<0,1>. If "0", this function sets the value to "1" and passes the *RB_Frame* array and *RB_Type* array to the staging function and allocates a new *RB_Frame* array and *RB_Type* array use the Allocate_RB_Frame() function. If I_value of *RB_Frame*<0,1> is "1", skips passing to staging and allocations, assume a new *RB_Frame* has been allocated, and increments to the first data carrying resource block and resource element in the new *RB_Frame*.

Map_to_QAM(*resource_block*, *resource_element* , *FILLWORD<>*, *FILLBITS*)

This function maps the bits in *FILLWORD<>* into the I and Q bin value pairs for the current resource element in the current *resource_block* (subcarrier) in *RB_Frame<>* (see 101.4.3.3). *FILLBITS* represents the bitloading of the current data resource element or low density pilot, where *FILLWORD<FILLBITS>* represents the MSB and *FILLWORD*<1> represents the LSB for the mapping.

If *RB_Type<resource_block>* is EX, this function~~s~~ overrides the value and sets *RB_Type<resource_block*) to *PILOT_MAP<resource_block>* to indicate this resource block is in use and contains non-null values.

Next_RE(r*esource_element*, *resource_block*)

This function increments to the next usable data carry resource element starting from the passed resource_element number in the passed resource_block (subcarrier). This includes data carry resource elements and low density pilots, pilot resource elements are skipped. See 101.4.3.7. The resource element is returned and may have a value from 1 to *RBlen*. If *resource_element* has a value of -1, this function increments to the next resource block and begins with the first useable resource element

This function will first increment to the next usable resource element in the current resource block. If necessary, this function will increment to the next data carrying resource block (e.g. Type 0, 1, or 2). The resource block number (subcarrier) from 0 to 4095

When advancing to the next resource block and *FILL_STATE* = FILL, the value from *Pilot_Map<resource block>* sets the value of *RB_Type<resource block>*.

If at any time this function increments beyond the last usable resource block in the current *RB_Frame* (highest usable subcarrier configured in the profile descriptor), It examines the I_value of *RB_Frame*<0,1>. If "0", this function sets the value to "1" and passes the *RB_Frame* array and *RB_Type* array to the staging function and allocates a new *RB_Frame* array and *RB_Type* array using the allocate_RB_Frame() function. If I_value of *RB_Frame*<0,1> is "1", skips passing to staging and allocations, and assumes a new

RB_Frame has already been allocated, and increments to the first data carrying resource block and resource element in the new RB Frame.
Upon return, the updated resource element index is returned via *resource_element*, and the current resource block (subcarrier) index returned via *resource_block*.

Reset_Scrambler()
This function initializes the bit scrambler with the seed value. See TBD.

EDITORS NOTE (to be removed prior to publication): there is currently no mention of a scrambler in the US PMA. We should probably point to the DS scrambler (101.4.2.6) noting the difference in when the seed is loaded.

Scramble(*bit*)
This function provides a bit scrambler, local to the upstream data symbol mapper function. The passed *bit* is used as input to the scrambler, the output is used as the return value. See TBD.

Stage_RB_Frame(*RB_Frame*, *RB_Type*)
This function examines the I_value of *RB_Frame*<0,1>. If "0", this function sets the value to "1" and passes the *RB_Frame*<> array and *RB_Type*<> array to the staging function. If I_value of *RB_Frame*<0,1> is "1", this function simply returns.

101.4.3.6.3 State diagrams

The CNU PMA shall perform the symbol mapper idle process as shown in Figure 101–29.

The CNU PMA shall perform the symbol mapper fill process as shown in Figure 101–30

.

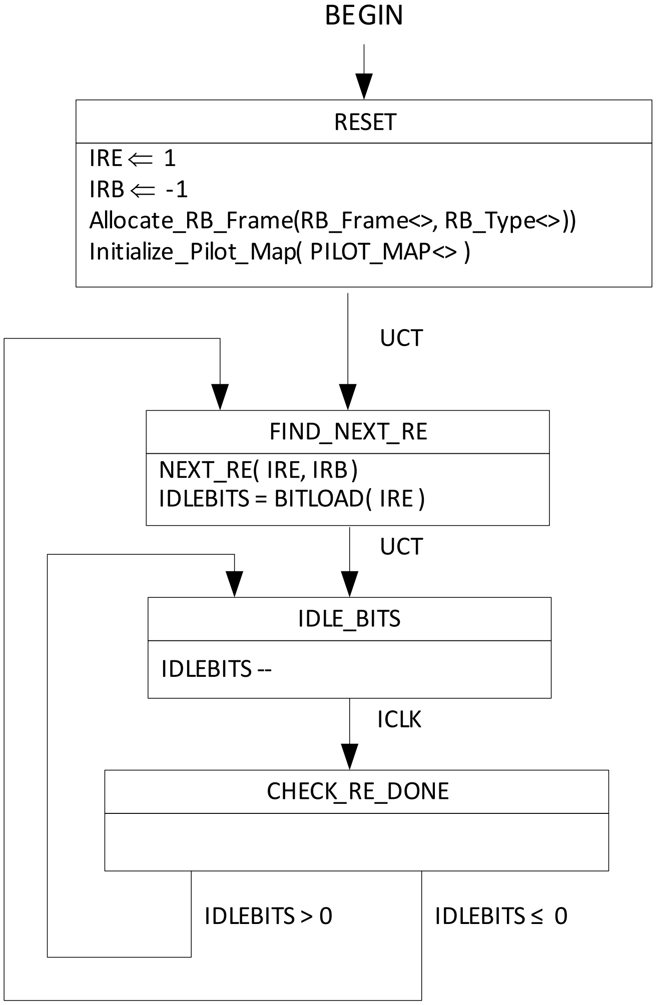


Figure 101–29—Upstream symbol mapper idle loop state diagram

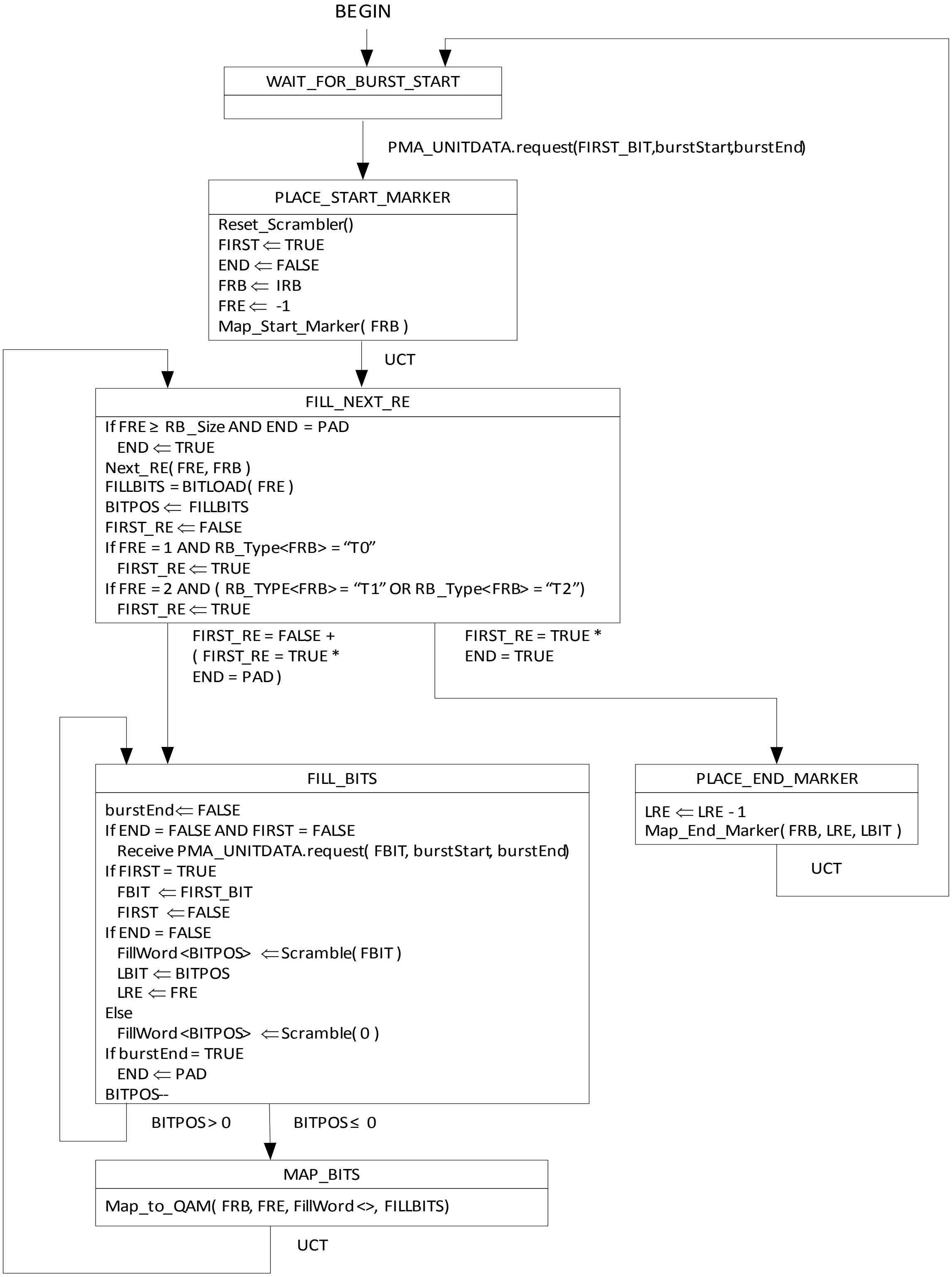


Figure 101–30—Upstream symbol mapper fill process state diagram

101.4.3.6.4 Minimum gap time and burst marker overhead

The CLT shall ensure a minimum gap time between bursts from any CNU equal to the transmission time of one (1) resource block expressed in units of *time_quant~~a~~um* (see 77.2.2.2) equal to 16 ns.

Let *Highestbitload* be the highest bit loading among all active subcarriers in any data carrying symbol (non-probe symbol) in an upstream RB frame, the number of *time_quant~~a~~um* per resource block is calculated as per Equation (101–26):

$$RB\_time\_quanta = RBlen \times (Highestbitload)/(US\_DataRate)/(time\_quantum) \qquad (101–26)$$

Start and end burst marker overhead is a multiple of *RB_time_quanta*. For *RBlen* of 8, total burst marker overhead is eight (8) times *RB_time_quanta*. For *RBlen* of 16, total burst marker overhead is four (4) times *RB_time_quanta*. See 101.4.3.9 for burst marker resource block use.

101.4.3.7 Pilot patterns

A Resource Block may be any one of three types as illustrated in Figure 101–31. Type 0 Resource Blocks contain only data resource elements modulated per the *ModTypeSC(n)* variable where *n* is the subcarrier index of the Resource Block. Type 1 Resource Blocks contain two pilots in the first and third resource element transmitted. Type 2 Resource Blocks contain a Low Density Pilot, in the last and third from last resource elements transmitted, in addition to the two pilots of the Type 1 Resource Block. The Low Density Pilot resource element is modulated using the higher modulation order of either BPSK or 4 bits lower than the bit loading specified in the *ModTypeSC(n)* variable for that subcarrier. Each RB type is configured via the variables *Type1_Start*, *Type1_Repeat*, *Type2_Start*, and *Type2_Repeat* as described below. The configuration of these variables determines the upstream transmission pilot pattern that all CNUs in the network use. However the pattern is defined over the entire 4095 subcarrier range with subcarrier 0 being the first subcarrier and subcarrier 4095 being the last subcarrier in the range. Excluded subcarrier settings override the pilot pattern definition, and Type 2 pilot definitions override Type 1 definitions. See 101.4.3.3.1 for additional rules on Pilot Type usage in burst transmissions.

The TypeN_Start variable determines on which subcarrier the repeating pattern for Type N pilot starts and the TypeN_Repeat variable determines the number of subcarriers between Type N pilots in the repeating pattern is encoded as shown in Table 101–15. *US_ModTypeSC(n)* excluded subcarriers override the repetitive pilot pattern and the Type 2 Pilot pattern overrides the Type 1 Pilot pattern.

Table 101–15—TypeN_Repeat[a] encoding

TypeN_Repeat value	Number of SC between TypeN Pilots
000	TypeN Pilot pattern disabled
001	0

Table 101–15—TypeN_Repeat[a] encoding *(continued)*

TypeN_Repeat value	Number of SC between TypeN Pilots
010	1
011	3
100	7
101	15
11x	reserved

[a] N = 1 or 2

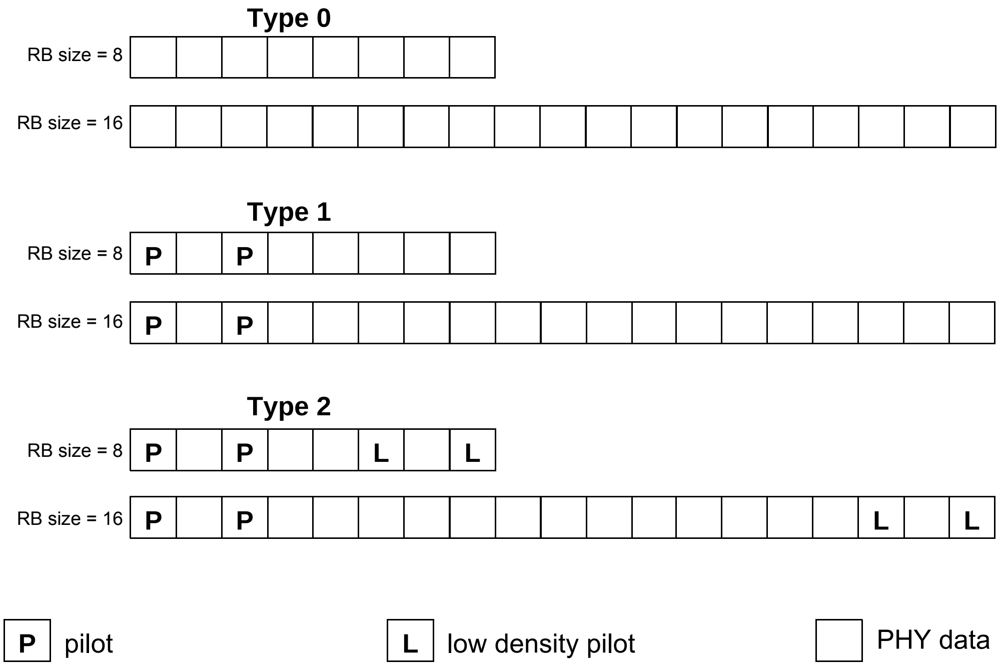


Figure 101–31—Resource Block types

101.4.3.7.1 variables

Type1_Repeat

TYPE: 3-bit unsigned integer

This variable indicates the number of subcarriers, from 0 to 7, between repeating Type 1 Pilots.

Type1_Start

TYPE:4-bit unsigned integer

This variable indicates the number, between 0 and 15, of the first subcarrier designated as a Type 1 Pilot.

Type2_Repeat

TYPE:3-bit unsigned integer

This variable indicates the number of subcarriers, from 0 to 7, between repeating Type 2 Pilots.

Type 1Start

TYPE:4-bit unsigned integer

This variable indicates the number, between 0 and 15, of the first subcarrier designated as a Type 2 Pilot.

EDITORS NOTE (to be removed prior to publication): the above definition are essentially copies from Cl 45.2.1.112. Recommend keeping this and referencing this from Cl 45.

101.4.3.8 Staging and Pilot Insertion

101.4.3.8.1 Staging

The Staging function accepts an RB_Frame and RB_Type array from the Symbol Mapper when transferred from the Symbol Mapper. Staging then accumulates (copies) a PHY Link RB Frame (8 subcarriers by resource block size 8 or 16) if available for this RB_Frame. If a PHY Link RB Frame was copied, the RB_Type entries for the corresponding subcarriers are set to “T2” (see RB_Type array in 101.x.x.x).

Pilot insertion then proceeds as per the procedure in 101.4.3.8.1. Upon a positive transition of RB_Frame_start (value FALSE to value TRUE) as set by Frame Timing, the RB_Frame and RB_Type arrays are transferred to the IDFT process.

101.4.3.8.2 Pilot Insertion

Upstream pilot insertion is performed using a BPSK mapped bit sequence generated by a pseudo-random sequence (PRBS) generator defined by the polynomial $X^{12} + X^9 + X^8 + X^5 + 1$ (illustrated in Figure 102–28). Pilots are inserted after the RB Frame is processed by the symbol mapper (see 101.4.2.8) and the interleaving functions (see 101.4.3.10) and before the RB Frame is passed to the IDFT function.

The method for pilot insertion shall be as follows:

- The PRBS generator is initialized with the seed value 0xBFF at the beginning of each RB Frame for the subcarrier with index k=0 of the IDFT Equation (101–18) (see 101.4.2.11).
- the PRBS generator is clocked once for every subcarrier of the IDFT.
- Pilots are inserted (mapped) into the “P” positions of PHY Link subcarriers in each subcarrier where a CNU is transmitting a PHY Link message (see 102.3.4) and in each resource block to be transmitted containing a data burst designated as Type 1 or Type 2 resource block (see 101.4.3.7).
- “P” pilots are BPSK modulated with the output of the feedback shift register, with a value of 0 mapping to (1 + j0) and a value of 1 mapping to (-1 + j0).
- The same BPSK value is used for each “P” location in a subcarrier.

When an RB Frame has been processed by this function, the RB Frame is passed to the IDFT function for further processing.

The Pilot Insertion function provides the PMD_SIGNAL.request primitive generation (see 100.2.1.4):

- If any “P” pilots are inserted in this RB Frame, PMD_SIGNAL.request(ENABLE) is asserted when the RB Frame is passed to the IDFT function.
- If no “P” pilots are inserted in this RB Frame, PMD_SIGNAL.request(DISABLE) is asserted when the RB Frame is passed to the IDFT function.

EDITORS NOTE (to be removed prior to publication): PMD_SIGNAL.request needs to be coordinated with the Probe Generator function and needs to be glitchless if the same CNU transmitted anything in the probe region followed by any transmission in the RB Frame following the probe region. Might be provided by an OR *function on the same signal generation from each function before going into PMD Functions box in Figure 100-3, or something that just surveys the input to the IDFT after both functions (in this case, modify and move this text into that functional description based on any non-null I/Q bin input to the IDFT.*

101.4.3.9 Burst markers

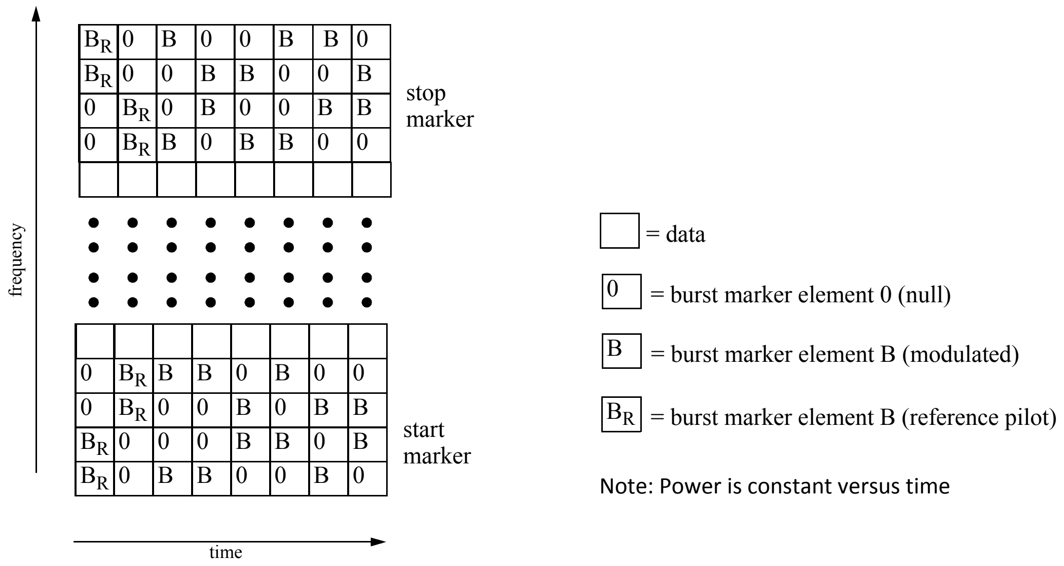
101.4.3.9.1 Introduction

Burst markers are used to indicate the start or end of a burst received via the PMA service interface. A burst marker is a predefined sequence of two types burst marker elements: B’s and 0’s, where B’s represent differential QPSK (D-QPSK) modulated symbols (see 101.4.3.9.3), and 0’s represent nulls (i.e., no energy being transmitted). Each burst marker element is transmitted in one resource element. B burst marker elements are boosted by 3dB. The first modulated B marker element on a subcarrier is encoded as a reference pilot. There are separate burst marker patterns for 8 and 16 symbol Resource Blocks.

Burst markers are placed by the upstream Symbol Mapper function (see 101.4.3.6).

101.4.3.9.2 Burst marker start and stop sequences

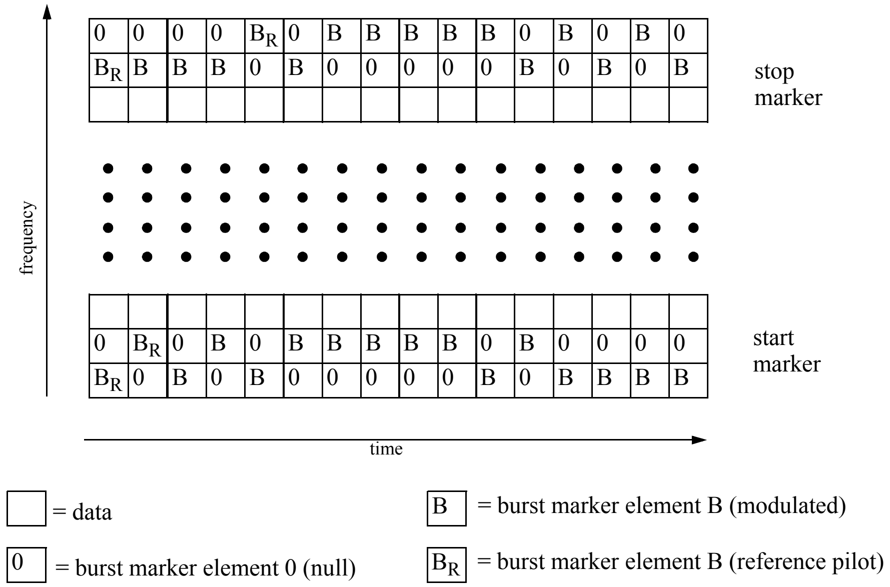
For the 8 symbol Resource Block, the start and stop burst marker sequences are defined in Figure 101–32. For the 16 symbol Resource Block, the start and stop marker sequences are defined in Figure 101–33.


stop
marker
start
marker
frequency
time
= data
0 = burst marker element 0 (null)
B = burst marker element B (modulated)
B_R = burst marker element B (reference pilot)
Note: Power is constant versus time

B_R	0	B	0	0	B	B	0
B_R	0	0	B	B	0	0	B
0	B_R	0	B	0	0	B	B
0	B_R	B	0	B	B	0	0

0	B_R	B	B	0	B	0	0
0	B_R	0	0	B	0	B	B
B_R	0	0	0	B	B	0	B
B_R	0	B	B	0	0	B	0

Figure 101–32—Burst marker in 8 symbol resource block


0 0 0 0 B_R 0 B B B B B 0 B 0 B 0
B_R B B B 0 B 0 0 0 0 0 B 0 B 0 B
stop marker
frequency
0 B_R 0 B 0 B B B B B 0 B 0 0 0 0
B_R 0 B 0 B 0 0 0 0 0 B 0 B B B B
start marker
time
= data
B = burst marker element B (modulated)
0 = burst marker element 0 (null)
B_R = burst marker element B (reference pilot)

Note: Power is constant versus time

Figure 101–33—Burst marker example in 16 symbol resource block

101.4.3.9.3 Burst marker B element encoding

The first modulated D-QPSK symbol for each subcarrier of a start and end burst marker is a reference pilot and is modulated with value (11). This is indicated by the position of the B_R burst marker element in Figure 101–32, Figure 101–33, Figure 101–34, and Figure 101–35.

The remaining B burst marker elements are D-QPSK modulated with information value pairs and parity encoding value pairs. The parity encoding is a Reed-Solomon coding over GF(2^4) with t = 2. The Reed-Solomon code of RS(15,11) is shortened to length 6 of 7 depending on the resource block length. The RS generator is shown in equation Equation (101–27) where the primitive element alpha (*a*) is 0x2.

$$g(x) = (x + a^0)(x + a^1)(x + a^2)(x + a^3) \quad (101\text{–}27)$$

The RS primitive polynomial is shown in equation Equation (101–28).

$$p(x) = x^4 + x + 1 \quad (101\text{–}28)$$

For Resource Block size of 8, two information code symbols designated I2 and I1 contain 8 information bits and are encoded and shortened to a length of 6:

(0 0 0 0 0 0 0 0 0 I2 I1 P4 P3 P2 P1)

For Resource Block size of 16, three information code symbols designated I3, I2, and I1 contain 12 information bits and are encoded and shortened to a length of 7:

(0 0 0 0 0 0 0 0 I3 I2 I1 P4 P3 P2 P1)

Each information code symbol is represented by a pair of D-QPSK symbols. The high order pair is designated as $I_{\#H}$ and the low order pair by $I_{\#L}$, for # = 1, 2, or 3. For each parity code symbol high order pair is designated as $P_{\#H}$ and the low order pair by $P_{\#L}$, for # = 1, 2, 3, or 4.

For start burst markers and Resource Block size 8, the two information code symbols I_2 and I_1 are each set to 0xF (i.e. 0xFF, all one's in all information symbols) with the D-QPSK modulated symbol pair placement as per Figure 101–34.

For start burst markers and Resource Block size 16, the three information code symbols $I_{3,}$ $I_{2,}$ and $I_1$are each set to 0xF (i.e., 0xFFF, all ones in all information symbols) with the D-QPSK modulated symbol pair placement as per Figure 101–35.

The start burst marker setting of 0xFF and 0xFFF in RB Frames of size 8 and 16, respectively designates that the first bit of data for the burst starts in the MSB bit of the first usable data resource element in the resource block immediately following the start burst marker. All other values and designations are reserved.

For stop burst markers and Resource Block size 8, the two information code symbols are set as follows: L_2 encodes the last resource block value as designated by the symbol mapper (see 101.4.3.6) as per Table 101–16 and L_1 encodes the last fill bit position value as designated by the symbol mapper (see 101.4.3.6) as per Table 101–17 with the D-QPSK modulated symbol pair placement as per Figure 101–34.

For stop burst markers and Resource Block size 16, the three information code symbols are set as follows: L_3 encodes a pad value of 0x00 (i.e., not null), L_2 encodes the last resource block value as designated by the symbol mapper (see 101.4.3.6) as per Table 101–16 and L_1 encodes the last fill bit position value as designated by the symbol mapper (see 101.4.3.6) as per Table 101–17 with the D-QPSK modulated symbol pair placement as per Figure 101–35.

Table 101–16—Last resource element position encoding

Last RE Position in Last Resource Block	MSB Pointer Bits (I_{2H})I($_{2L}$) (0_{MSB} 0)$_H$ (0_{MSB} 0)$_L$
0	(00)$_H$(00)$_L$
1	(00)$_H$(01)$_L$
2	(00)$_H$(10)$_L$
3	(01)$_H$(11)$_L$
4	(01)$_H$(00)$_L$
5	(01)$_H$(01)$_L$
6	(01)$_H$(10)$_L$
7	(01)$_H$(11)$_L$
8	(10)$_H$(00)$_L$
9	(10)$_H$(01)$_L$
10	(10)$_H$(10)$_L$

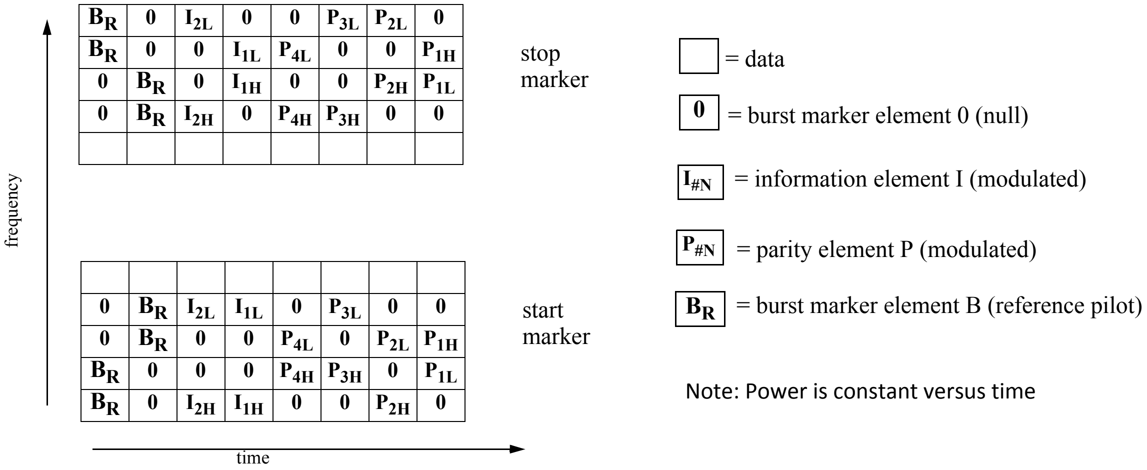
Table 101–16—Last resource element position encoding *(continued)*

Last RE Position in Last Resource Block	**MSB Pointer Bits** $(I_{2H})I(_{2L})$ $(0_{MSB}\ 0)_H\ (0_{MSB}\ 0)_L$
11	$(10)_H(11)_L$
12	$(11)_H(00)_L$
13	$(11)_H(01)_L$
14	$(11)_H(10)_L$
15	$(11)_H(11)_L$

.

Table 101–17—Last bit position encoding

Last Fill Bit Position in Last RE	**MSB Pointer Bits** $(I_{1H})I(_{1L})$ $(0_{MSB}\ 0)_H\ (0_{MSB}\ 0)_L$
0	$(00)_H(00)_L$
1	$(00)_H(01)_L$
2	$(00)_H(10)_L$
3	$(01)_H(11)_L$
4	$(01)_H(00)_L$
5	$(01)_H(01)_L$
6	$(01)_H(10)_L$
7	$(01)_H(11)_L$
8	$(10)_H(00)_L$
9	$(10)_H(01)_L$
10	$(10)_H(10)_L$
11	$(10)_H(11)_L$
12	$(11)_H(00)_L$
13	$(11)_H(01)_L$
14	$(11)_H(10)_L$
15	$(11)_H(11)_L$


frequency
time
stop marker
start marker
= data
0 = burst marker element 0 (null)
$I_{\#N}$ = information element I (modulated)
$P_{\#N}$ = parity element P (modulated)
B_R = burst marker element B (reference pilot)

stop marker

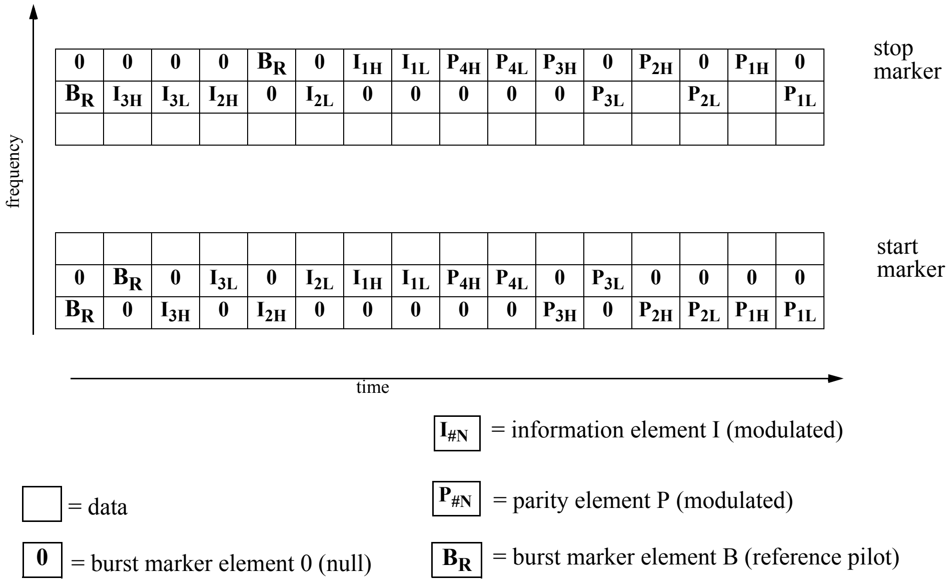
B_R	0	I_{2L}	0	0	P_{3L}	P_{2L}	0
B_R	0	0	I_{1L}	P_{4L}	0	0	P_{1H}
0	B_R	0	I_{1H}	0	0	P_{2H}	P_{1L}
0	B_R	I_{2H}	0	P_{4H}	P_{3H}	0	0

start marker

0	B_R	I_{2L}	I_{1L}	0	P_{3L}	0	0
0	B_R	0	0	P_{4L}	0	P_{2L}	P_{1H}
B_R	0	0	0	P_{4H}	P_{3H}	0	P_{1L}
B_R	0	I_{2H}	I_{1H}	0	0	P_{2H}	0

Note: Power is constant versus time

Figure 101–34—Burst marker encoding in 8 symbol resource block


frequency
time
stop marker
start marker
$I_{\#N}$ = information element I (modulated)
= data
$P_{\#N}$ = parity element P (modulated)
0 = burst marker element 0 (null)
B_R = burst marker element B (reference pilot)

stop marker

0	0	0	0	B_R	0	I_{1H}	I_{1L}	P_{4H}	P_{4L}	P_{3H}	0	P_{2H}	0	P_{1H}	0
B_R	I_{3H}	I_{3L}	I_{2H}	0	I_{2L}	0	0	0	0	0	P_{3L}		P_{2L}		P_{1L}

start marker

0	B_R	0	I_{3L}	0	I_{2L}	I_{1H}	I_{1L}	P_{4H}	P_{4L}	0	P_{3L}	0	0	0	0
B_R	0	I_{3H}	0	I_{2H}	0	0	0	0	0	P_{3H}	0	P_{2H}	P_{2L}	P_{1H}	P_{1L}

Note: Power is constant versus time

Figure 101–35—Burst marker encoding in 16 symbol resource block

101.4.3.10 Interleaver and OFDM framer

101.4.3.11 Pre-equalization and Inverse Discrete Fourier Transform (IDFT)

The CNU upstream IDFT uses the same definition as in the downstream. See 101.4.2.11.

101.4.3.11.1 Pre-equalization coefficients

Linear pre-equalization is performed in the digital domain before upstream IDFT processing. The CNU shall support a single complex coefficient (equalizer tap) per subcarrier that may be updated via the PHY Link. A CNU shall use a default value of *1+j0* for all pre-equalizer coefficients.

On update, the PHY Link may either indicate a complex coefficient (initialization or reset) or may indicate a multiplication factor. Coefficients are updated only as part of a response to upstream probes received by the CLT. See 102.4.2.

EDITORS NOTE (to be removed prior to publication): data rate/load impact of updating 256 CNUs over the PHY Link on a periodic cycle (less than 30 second update, typical, maybe slower for EPoC and "small plant") will need to be studied. If the downstream PHY Link is overwhelmed, it may be necessary to 1) use OAM messaging for adjustments to 'linked' CNUs or 2) to reduce resolution if performed in PHY Link only. The TF needs to agree on how Pre-Equalization Coefficients (i.e., mdio registers) are adjusted via the PHY Link and how Probes are scheduled by the CLT.

Upon reception of a coefficient multiplication factor, the CNU processes the update per subcarrier as follows:

$$C(k) \Leftarrow C(k) \times A(k) \qquad (101\text{–}29)$$

where *C(k)* is the pre-equalizer coefficient of the k-th subcarrier as used in the last probe transmission, and *A(k)* is the coefficient updates in variables *EQ_CoefR(k)* and *EQ_CoefI(k)* (see 101.4.3.11.2), received via the PHY Link. The symbol "×" indicates a complex multiplication.

The CNU normalizes the new calculated coefficients as follows:

1) Upon applying any updates, the CNU shall normalize the new calculated coefficients as follows: *mean (abs (Ck)^2) = 1* (summation is over all *k* subcarriers, which are active subcarriers).
2) The CNU shall apply the newly calculated coefficients for transmitting within 10 ms after receiving an update via a PHY Link message.
3) The newly calculated coefficients for transmitting shall take affect at the beginning of a transmission.

On transmissions, the CNU shall:

1) Always pre-equalize all transmissions other than probe and PHY Discovery signals.
2) Transmit a probe signal with or without pre-equalization (all coefficients are reset to $1+j\times 0$) as instructed by the CLT using the PHY Link probe instruction described in 102.4.2.

EDITORS NOTE (to be removed prior to publication): The following topics need to be covered in some fashion for CLT operation during probe request and response. This text will likely appear in PHY Link clause:
1) The CLT shall specify the subcarriers (i.e., frequency range) over which coefficient updates are to be performed
2) Need to make sure that when switching from current upstream profile to the next where there is a change between excluded and active subcarriers use, the CNU upstream PHY is reset. This will force a re-evaluation of pre-equalizer coefficients.]

101.4.3.11.2 Variables

EQ_CoefR(n)
TYPE: Q2.14 format signed fractional number

This set of variables determines the real and imaginary part of the pre-equalizer settings for the upstream transmitter. Each variable in the set controls one subcarrier of the 4096 subcarriers that comprise the OFDMA channel, with *EQ_CoefR(0)* controlling the real number setting for subcarrier 0 and *EQ_CoefR(1)* controlling the real number setting for subcarrier 1 and so on. Thus *EQ_CoefR(4096)* controls the real settings for subcarrier 4095.

EQ_CoefI(n)

TYPE:Q2.14 format signed fractional number

This set of variables determines the real and imaginary part of the pre-equalizer settings for the upstream transmitter. Each variable in the set controls one subcarrier of the 4096 subcarriers that comprise the OFDMA channel, with *EQ_CoefI(0)* controlling the imaginary number setting for subcarrier 0 and *EQ_CoefI(1)* controlling the imaginary number setting for subcarrier 1and so on. Thus *EQ_CoefI(4095)* controls the imaginary settings for subcarrier 4095.

EDITORS NOTE (to be removed prior to publication): the above definition are essentially copies from Cl 45.2.7a.3. Recommend keeping this and referencing this from Cl 45.

101.4.3.12 Cyclic prefix and windowing

EDITORS NOTE (to be removed prior to publication): Cyclic prefix and windowing function needs to reflect symbol duplication for PHY Link Discovery Response message.

The CNU upstream cyclic prefix and windowing function uses the same definition as in the downstream. See 101.4.2.12. The CNU shall use one of the permissible values for *USNcp* and *USNrp* in the upstream direction given in Table 101–18 and Table 101–19, respectively.

Table 101–18—Size of cyclic prefix (USNcp), upstream direction

USNcp[a]	[µs]
256	1.25
384	1.875
512	2.5
640	3.125
768	3.75

[a] USNcp represents sample clock period, equal to 1/ 204.8 MHz

Table 101–19—Size of OFDM window (USNrp), upstream direction

USNrp[a]	[us]
0	0.0000
64	0.3125
128	0.6250
192	0.9375
256	1.2500

[a] USNrp represents sample clock period, equal to 1/ 204.8 MHz

101.4.3.12.1 Variables

USNcp

TYPE: 4-bit binary

This variable controls the size of the cyclic prefix in the upstream direction per the enumeration below.

bit 3 2 1 0
1 x x x = reserved
0 1 1 1 = 768 samples
0 1 1 0 = 640 samples
0 1 0 1 = reserved
0 1 0 0 = 512 samples
0 0 1 1 = reserved
0 0 1 0 = 384 samples
0 0 0 1 = reserved
0 0 0 0 = 256 samples
samples refer to OFDM clock (204.8 MHz)

USNrp

TYPE: 3-bit binary

This variable controls the size of the windowing function in the upstream direction per the enumeration below.

2 1 0
1 1 1 = 256 samples
1 1 0 = 192 samples
1 0 1 = reserved
1 0 0 = 128 samples
0 1 1 = reserved
0 1 0 = 64 samples
0 0 1 = reserved
0 0 0 = 0 samples (windowing disabled)
samples refer to OFDM clock (204.8 MHz)

101.4.4 Constellation structure and mapping

After LDPC encoding and scrambling for downstream and upstream transmissions, the output bit stream of the scrambler must be mapped to QAM symbols such that first bit is the least-significant bit of the first QAM subcarrier constellation m-tuple, see Figure 101–36.

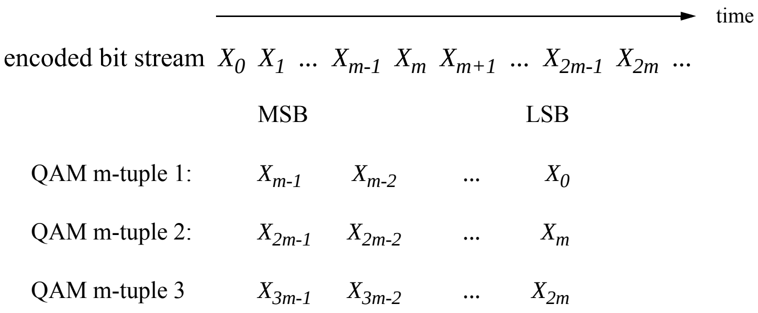

time
encoded bit stream X_0 X_1 ... X_{m-1} X_m X_{m+1} ... X_{2m-1} X_{2m} ...
MSB
LSB
QAM m-tuple 1: X_{m-1} X_{m-2} ... X_0
QAM m-tuple 2: X_{2m-1} X_{2m-2} ... X_m
QAM m-tuple 3 X_{3m-1} X_{3m-2} ... X_{2m}

Figure 101–36—Bitstream to QAM m-tuple mapping

The m-tuples must be modulated onto subcarriers using QAM constellation. As described in the following sub-sections, the QAM constellation structure and mappings are defined inductively and use Gray mapping as their base.

101.4.4.1 One dimensional Gray mapping for m-tuple binary bits

1) When m=1, the Gray mapping is define to be $\mathrm{Gray}_1(0) = 1$ and $\mathrm{Gray}_1(1) = -1$
2) When m>1, the Gray mapping is defined inductively, i.e.,
$\mathrm{Gray}_m(x_{m-1} \cdot x_{m-2} \cdot \ldots \cdot x_0) = (1 - 2 \cdot x_0) \cdot (2^{m-1} + \mathrm{Gray}_{m-1} \cdot (x_{m-1} \cdot x_{m-2} \cdot \ldots \cdot x_0))$

101.4.4.2 Constellation structure and mapping of BPSK

Let $m = 1$ and a binary bit is x. The BPSK mapping is: $(I_1(x), Q_1(x)) = (\mathrm{Gray}_1(x), 0) = \begin{cases} (1, 0) & (x = 0) \\ (-1, 0) & (x = 1) \end{cases}$. Also see Figure 101–37.

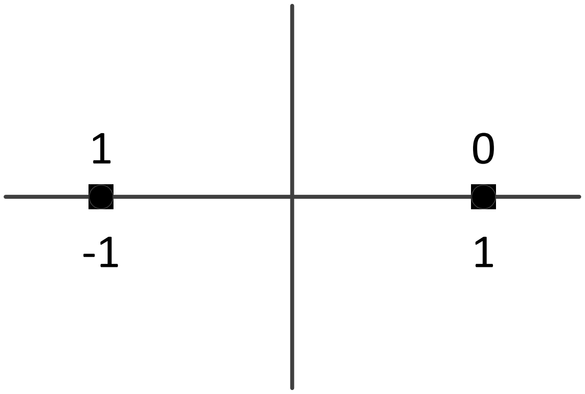


Figure 101–37—BPSK

101.4.4.3 Constellation structure and mapping of 2^{2n}–QAM

Let $m = 2 \cdot n$ and the m-tuple binary bits are $x_0, ..., x_{n-1}, x_n, ..., x_{2 \cdot n - 1}$. The mapping from that m-tuple to a 2^m –QAM is defined by

$$(I_{2n}(x_{2n-1}, ..., x_n, x_{n-1}, ..., x_0), Q_{2n}(x_{2n-1}, ..., x_n, x_{n-1}, ..., x_0)) = (\mathrm{Gray}_n(x_{n-1}, ..., x_0), \mathrm{Gray}_n(x_{n-1}, ..., x_0)) \qquad (101–30)$$

where the Gray mapping is defined in 101.4.4.1. Some of the examples are given the following figures.

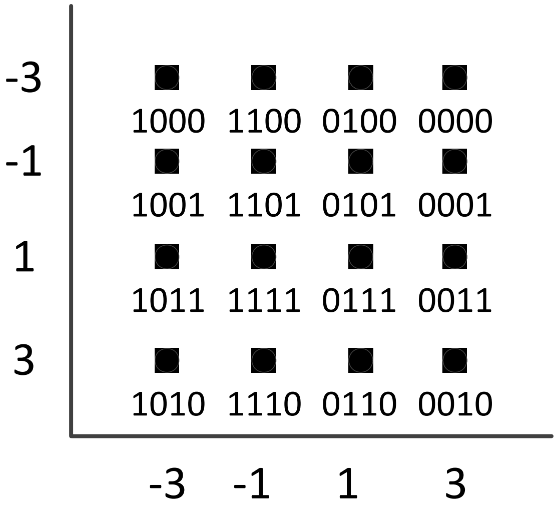


Figure 101–38—16-QAM

101.4.4.4 Constellation structure and mapping of 2^{2n+1}–QAM (n>0)

Let $m = 2 \cdot n + 1$ and the m-tuple binary bits are $x_0, ..., x_{n-1}, x_n, ..., x_{2 \cdot n}$. Firstly, map this m-tuple to a rectangular constellation defined by

$$(I_{rct}(x_{2n}, ..., x_n, x_{n-1}, ..., x_0), Q_{rct}(x_{2n}, ..., x_n, x_{n-1}, ..., x_0)) = (\mathrm{Gray}_{n+1}(x_{2n}x_{2n-1}...x_n), \mathrm{Gray}_n(x_{n-1}, x_{n-2}, ..., x_0)) \qquad (101–31)$$

where the Gray mapping is defined in 101.4.4.1. Then the structures and mappings of cross-constellations are generated in the following sub-sections.

101.4.4.4.1 Constellation structure and mapping of 8–QAM

Let the constellation signal and its mapping be denoted by $(I_3(x_2x_1x_0), Q_3(x_2x_1x_0))$ then

$$\begin{cases} \begin{cases} I_3(x_2x_1x_0) = I_{rct}(x_2x_1x_0) + 1 \\ Q_3(x_2x_1x_0) = Q_{rct}(x_2x_1x_0) \end{cases} & I_{rct}(x_2x_1x_0) < 3 \\ \begin{cases} I_3(x_2x_1x_0) = 3 - I_{rct}(x_2x_1x_0) \\ Q_3(x_2x_1x_0) = sign(Q_{rct}(x_2x_1x_0)) \cdot (|Q_{rct}(x_2x_1x_0)| + 2) \end{cases} & \text{otherwise} \end{cases}$$

where the sign function is defined by $sign(a) = \begin{cases} 1 & a \geq 0 \\ -1 & a < 0 \end{cases}$

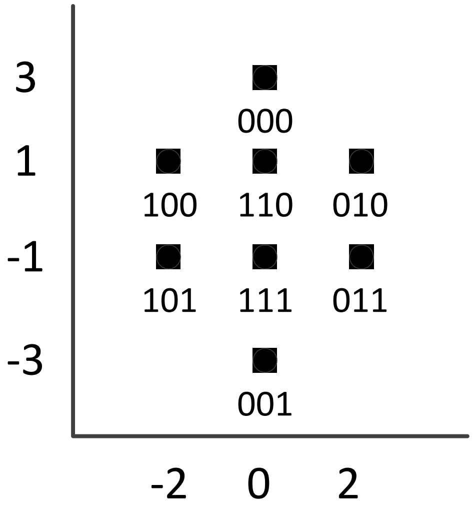


Figure 101–39—8-QAM

101.4.4.4.2 Constellation structure and mapping of 2^{2n+1}–QAM with n>1

Let the mapping be denoted by $(I_{2n+1}(x_{2n}x_{2n-1}...x_0), Q_{2n+1}(x_{2n}x_{2n-1}...x_0))$ and let $s = 2^{n-1}$. Then, when $|I_{rct}(x_{2n}x_{2n-1}...x_0)| < 3s$

$$\begin{cases} I_{2n+1}(x_{2n}x_{2n-1}...x_0) = I_{rct}(x_{2n}x_{2n-1}...x_0) \\ Q_{2n+1}(x_{2n}x_{2n-1}...x_0) = Q_{rct}(x_{2n}x_{2n-1}...x_0) \end{cases}$$

and when $|I_{rct}(x_{2n}x_{2n-1}...x_0)| \geq 3s$

$$\begin{cases} \begin{cases} I_{2n+1}(x_{2n}x_{2n-1}...x_0) = sign(I_{rec}(x_{2n}x_{2n-1}...x_0)) \cdot (|I_{rct}(x_{2n}x_{2n-1}...x_0)| - 2s) \\ Q_{2n+1}(x_{2n}x_{2n-1}...x_0) = sign(Q_{rct}(x_{2n}x_{2n-1}...x_0)) \cdot (4s - |Q_{rct}(x_{2n}x_{2n-1}...x_0)|) \end{cases} & \text{if } |Q_{rct}(x_{2n}x_{2n-1}...x_0)| > s \\ \begin{cases} I_{2n+1}(x_{2n}x_{2n-1}...x_0) = sign(I_{rec}(x_{2n}x_{2n-1}...x_0)) \cdot (4s - |I_{rct}(x_{2n}x_{2n-1}...x_0)|) \\ Q_{2n+1}(x_{2n}x_{2n-1}...x_0) = sign(Q_{rct}(x_{2n}x_{2n-1}...x_0)) \cdot (|Q_{rct}(x_{2n}x_{2n-1}...x_0)| + 2s) \end{cases} & \text{if } |Q_{rct}(x_{2n}x_{2n-1}...x_0)| \leq s \end{cases}$$

Figure 101–40 presents the 32-QAM structure and mapping.

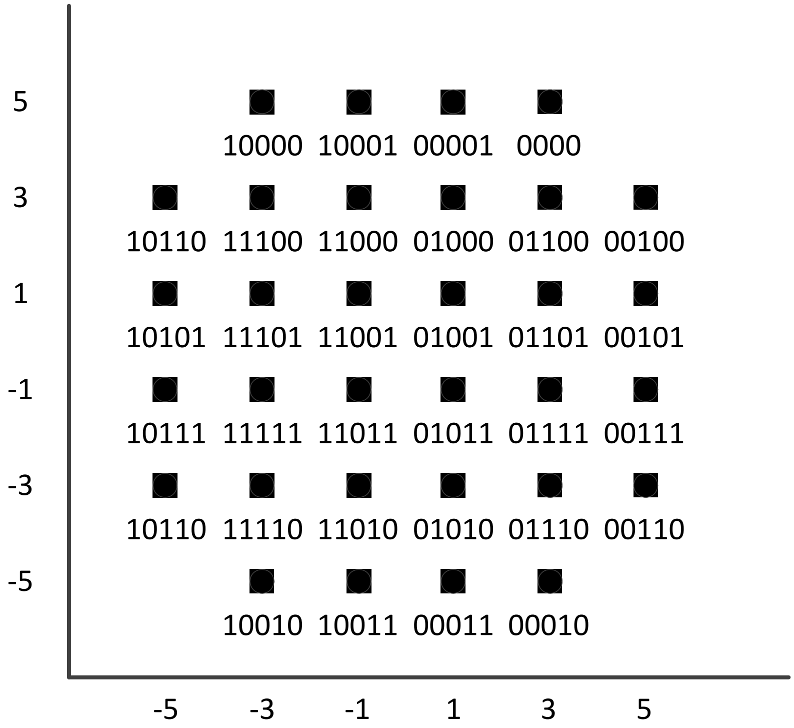


Figure 101–40—32-QAM

101.4.4.5 QAM constellation scaling

Both real and imaginary axes of a QAM constellation shall be scaled. The scaling factors given in column 3 of Table 101–20 ensure that the mean square value of all QAM constellations are equal to 1.0.

Table 101–20—QAM constellation scaling factors

QAM constellation	m number of bits	scaling factor
BPSK	1	1
QPSK	2	$1/(\sqrt{2})$
8-QAM	3	$1/(\sqrt{5})$
16-QAM	4	$1/(\sqrt{10})$
32-QAM	5	$1/(\sqrt{20})$
64-QAM	6	$1/(\sqrt{42})$
128-QAM	7	$1/(\sqrt{82})$
256-QAM	8	$1/(\sqrt{170})$
512-QAM	9	$1/(\sqrt{330})$
1024-QAM	10	$1/(\sqrt{682})$

Table 101–20—QAM constellation scaling factors *(continued)*

2048-QAM	11	$1/(\sqrt{1322})$
4096-QAM	12	$1/(\sqrt{2730})$
8192-QAM	13	$1/(\sqrt{5290})$
16384-QAM	14	$1/(\sqrt{10922})$

101.5 Power-saving capabilities

EDITORS NOTE (to be removed prior to publication): This subclause is reserved for the summary of the power-saving capabilities for this PMD type. This material would be all new in the amendment added by IEEE P802.3bn EPoC Task Force

101.6 TimeSync capability

EDITORS NOTE (to be removed prior to publication): This subclause is reserved for the summary of the TimeSync capabilities for this PMD type. Given that it is a new PMD design, we can embed TimeSync capability from day one. This involves primarily guaranteeing repeatable and stable delay as well as support for specific capability registers. See IEEE Std 802.3-2012, Clause 90 for more details. This material would be all new in the amendment added by IEEE P802.3bn EPoC Task Force.

101.7 Protocol implementation conformance statement (PICS) proforma for Clause 101, Reconciliation Sublayer, Physical Coding Sublayer, and Physical Media Attachment for EPoC[2]

101.7.1 Introduction

The supplier of a protocol implementation that is claimed to conform to Clause 101, Reconciliation Sublayer, Physical Coding Sublayer, and Physical Media Attachment for EPoC, shall complete the following protocol implementation conformance statement (PICS) proforma.

A detailed description of the symbols used in the PICS proforma, along with instructions for completing the PICS proforma, can be found in Clause 21.

[2] *Copyright release for PICS proformas:* Users of this standard may freely reproduce the PICS proforma in this subclause so that it can be used for its intended purpose and may further publish the completed PICS.

101.7.2 Identification

101.7.2.1 Implementation identification

Supplier[1]	
Contact point for enquiries about the PICS[1]	
Implementation Name(s) and Version(s)[1,3]	
Other information necessary for full identification—e.g., name(s) and version(s) for machines and/or operating systems; System Name(s)[2]	
NOTE 1—Required for all implementations. NOTE 2—May be completed as appropriate in meeting the requirements for the identification. NOTE 3—The terms Name and Version should be interpreted appropriately to correspond with a supplier's terminology (e.g., Type, Series, Model).	

101.7.2.2 Protocol summary

Identification of protocol standard	IEEE Std 802.3bn-2012, Clause 101, Reconciliation Sublayer, Physical Coding Sublayer, and Physical Media Attachment for EPoC
Identification of amendments and corrigenda to this PICS proforma that have been completed as part of this PICS	
Have any Exception items been required? No [ ] Yes [ ] (See Clause 21; the answer Yes means that the implementation does not conform to IEEE Std 802.3bn-2012.)	

Date of Statement	

101.7.3 Major capabilities/options

Item	Feature	Subclause	Value/Comment	Status	Support
CLT	CLT Functionality	101.2.1	Device supports the functionality required for CLT	O	Yes [ ] No [ ] N/A [ ]
CNU	CNU Functionality	101.2.1	Device supports the functionality required for NCU	O	Yes [ ] No [ ] N/A [ ]

101.7.4 PICS proforma tables for Reconciliation Sublayer, Physical Coding Sublayer, and Physical Media Attachment for EPoC

101.7.4.1 General specifications

Item	Feature	Subclause	Value/Comment	Status	Support
G1	Unidirectional mode	101.2.2	Device operates in unidirectional transmission mode	CLT: M	Yes [ ] No [ ] N/A [ ]
G2	Delay variation	101.1.2	Combined delay variation through RS, PCS, and PMA sublayers is limited to 1 time_quantum	M	Yes [ ] No [ ]
G3	PMA to PCS transfer function	101.3.3.1.8	Meets the requirements of Figure 101-13	CLT: M	Yes [ ] No [ ] N/A [ ]
G4	PMA service interface	101.4.1.2	Support for PMA_UNITDATA.request() and PMA_UNITDATA.indication()	M	Yes [ ] No [ ]

101.7.4.2 Management functions

Item	Feature	Subclause	Value/Comment	Status	Support
TX1	DS Excluded Subcarriers	01.4.2.4.4	No transmissions in excluded subcarriers.	M	Yes [ ] No [ ]
TX2	US Excluded Subcarriers	101.4.3.4.3	No transmissions in excluded subcarriers.	CNU: M	Yes [ ] No [ ] N/A [ ]
TX3	US Band Edge	101.4.3.3.3	Band edges bounded by Type 2 resource block	CNU: M	Yes [ ] No [ ] N/A [ ]
TX4	Burst Start	101.4.3.3.2	Burst begins with start burst marker	CNU: M	Yes [ ] No [ ] N/A [ ]
TX5	Burst End	101.4.3.3.4	Burst ends with start end marker	CNU: M	Yes [ ] No [ ] N/A [ ]

101.7.4.3 OFDM Configuration functions

Item	Feature	Subclause	Value/Comment	Status	Support
OC1	FDM channel 1	101.4.2.1	OFDM channel 1 always enabled	M	Yes [ ] No [ ]
OC2	DS Frame Timing	101.4.3.3.6	Meets the requirements of Figure 101-28	CNU: M	Yes [ ] No [ ] N/A [ ]
OC3	DS Subcarrier configuration	101.4.2.4	Meets the requirements of Table 101-9	M	Yes [ ] No [ ]
OC4	DS CP values	101.4.2.12	As shown in Table 101-11	CLT: M	Yes [ ] No [ ] N/A [ ]
OC5	DS Windowing values	101.4.2.12	As shown in Table 101-12	CLT: M	Yes [ ] No [ ] N/A [ ]
OC6	DS Windowing relative to CP	101.4.2.12	Windowing value less than CP value	CLT: M	Yes [ ] No [ ] N/A [ ]
OC7	DS Profile changes ignored	101.4.1.1.1	When DS_CpyInP is a one writes to all downstream profile variables shall be ignored and switching between profiles is prohibited	CNU: M	Yes [ ] No [ ] N/A [ ]
OC8	US Superframe	101.4.3.3.1	Six symbol Probe Period followed by 256 symbols	CNU: M	Yes [ ] No [ ] N/A [ ]
OC9	US subcarrier configuration	101.4.3.4	Meets the requirements of Table 101-14	CNU: M	Yes [ ] No [ ] N/A [ ]
OC10	US CP size	101.4.3.12	Per Table 101-18	CNU: M	Yes [ ] No [ ] N/A [ ]
OC11	US Windowing size	101.4.3.12	Per Table 101-19	CNU: M	Yes [ ] No [ ] N/A [ ]
OC12	US Profile changes ignored	101.4.1.1.1	When US_CpyInP is a one writes to all upstream profile variables shall be ignored and switching between profiles is prohibited	CNU: M	Yes [ ] No [ ] N/A [ ]
OC13	QAM constellation scaling	101.4.4.5	Per table 101-20	M	Yes [ ] No [ ]

101.7.4.4 OFDM Timing

101.7.4.5 Data Detector functions

Item	Feature	Subclause	Value/Comment	Status	Support
OT1	Downstream Synchronization	101.4.2.2	CLT transmitters and CNU receivers meet the requirements of Table 101-8	M	Yes [ ] No [ ]
OT2	CLT synchronization	101.4.2.2	CLT OFDM Clock and RF transmissions locked to 10.24 MHz Master Clock	CLT: M	Yes [ ] No [ ] N/A [ ]
OT3	CLT Subcarrier clock source	01.4.2.3	Synchronous with the 10.24 MHz Master Clock	CLT: M	Yes [ ] No [ ] N/A [ ]
OT4	CLT Subcarrier clock frequency	101.4.2.3	Meets the requirements of Eq (101-X)	CLT: M	Yes [ ] No [ ] N/A [ ]
OT5	CLT phase noise	101.4.2.2	CLT meets the more stringent requirements from Tables 100-4 (Phase noise) and 101-8 (clock jitter)	CLT: M	Yes [ ] No [ ] N/A [ ]
OT6	CNU synchronization	101.4.2.2	CNU Syncronizes it's clock to PHY Link frame	CNU: M	Yes [ ] No [ ] N/A [ ]
OT7	FCP Update	101.4.2.8.4	FPC value passed with sufficient time for insertion in PHY Link frame	CLT: M	Yes [ ] No [ ] N/A [ ]

Item	Feature	Subclause	Value/Comment	Status	Support
DD1	Data Detector input process	101.3.2.5.8	Meets the requirements of Figure 101-8	M	Yes [ ] No [ ]
DD2	CLT Data Detector output process	101.3.2.5.8	Meets the requirements of Figure 101-9	CLT: M	Yes [ ] No [ ] N/A [ ]
DD3	CNU Data Detector output process	101.3.2.5.8	Meets the requirements of Figure 101-8	CNU: M	Yes [ ] No [ ] N/A [ ]

101.7.4.6 IDLE insertion and deletion functions

Item	Feature	Subclause	Value/Comment	Status	Support
IDI1	CLT Idle Deletion function implementation	101.3.2.1.5	Meets the requirements of Figure 101–2	CLT: M	Yes [ ] No [ ] N/A [ ]
IDI2	CNU Idle Deletion function implementation	101.3.2.1.5	Meets the requirements of Figure 101-3 and Figure 101-4	CNU: M	Yes [ ] No [ ] N/A [ ]
IDI3	Idle Insertion	101.3.3.3.5	Meets the requirements of Figure 101-16	M	Yes [ ] No [ ]

101.7.4.7 FEC functions

Item	Feature	Subclause	Value/Comment	Status	Support
FE1	CLT FEC Encoder	101.3.2.4	LDPC (16200, 14400)	CLT: M	Yes [ ] No [ ] N/A [ ]
FE2	CNU FEC Decoder	101.3.3	LDPC (16200, 14400)	CNU: M	Yes [ ] No [ ] N/A [ ]
FE3	CNU FEC input/output	101.3.3.1.8	CNU FEC input process meets the requirements of Figure 101-14 and CNU FEC output process meets the requirements of Figure 101-15	CNU: M	Yes [ ] No [ ] N/A [ ]
FE4	CNU FEC Encoder	101.3.2.4	LDPC (16200, 14400), LDPC (5940,5040) and LDPC (1120, 840)	CNU: M	Yes [ ] No [ ] N/A [ ]
FE5	CLT FEC Decoder	101.3.3	LDPC (16200, 14400), LDPC (5940,5040) and LDPC (1120, 840)	CLT: M	Yes [ ] No [ ] N/A [ ]
FE6	CRC40 Calculation	101.3.2.3	Meets the requirements of Figure 101–5	M	Yes [ ] No [ ]
FE7	CRC40 Initialization	101.3.2.3	CRC40 calculation initialized to 0x00 at the beginning of each FEC codeword	M	Yes [ ] No [ ]
FE8	Uncorrectable FEC codeword indication	101.3.3.1.4	Uncorrectable FEC codewords marker under user configuration per Table 101-6	M	Yes [ ] No [ ]

101.7.4.8 Encoding functions

Item	Feature	Subclause	Value/Comment	Status	Support
EN1	CLT Scrambler	101.4.2.7	Downstream data scrambler meets the requirement of Figure 101-19	CLT: M	Yes [ ] No [ ] N/A [ ]
EN2	CLT Symbol Mapping	101.4.2.8.3	Symbol mapper processes all active subcarriers	CLT: M	Yes [ ] No [ ] N/A [ ]
EN3	CLT Time Interleaving	101.4.2.9.2	Time interleaving meets the requirement of 101.4.2.9.2	CLT: M	Yes [ ] No [ ] N/A [ ]
EN4	CLT Time Interleaving depth	101.4.2.9.2	Between 1 and 32 inclusive	CLT: M	Yes [ ] No [ ] N/A [ ]
EN5	CLT Frequency Interleaving	101.4.2.9.3	Frequency interleaving meets the requirements of 101.4.2.9.3	CLT: M	Yes [ ] No [ ] N/A [ ]
EN6	CNU Symbol mapper Idle function	101.4.3.6.3	Meets the requirements of Figure 101-29	CNU: M	Yes [ ] No [ ] N/A [ ]
EN7	CNU Symbol mapper fill function	101.4.3.6.3	Meets the requirements of Figure 101-30	CNU: M	Yes [ ] No [ ] N/A [ ]

101.7.4.9 Pilots

Item	Feature	Subclause	Value/Comment	Status	Support
PI1	Scattered pilot definition	101.4.2.6.1	Defined per 101.4.2.6	CLT: M	Yes [ ] No [ ] N/A [ ]
PI2	Scattered pilot synchronization	101.4.2.6.1	Meets the requirements of Figure 101-18	CLT: M	Yes [ ] No [ ] N/A [ ]
PI3	Continuous Pilot placement	101.4.2.6.4	Meets the Eq 101-5 and the eight steps given in 101.4.2.6.4	CLT: M	Yes [ ] No [ ] N/A [ ]
PI4	PHY Link Continous Pilots	101.4.2.6.3	Four pairs placed symmetrically about the PHY Link as in Figure 102-8 and Table 101-10	CLT: M	Yes [ ] No [ ] N/A [ ]
PI5	Pilot modulation	101.4.2.10	BPSK using pseudo-random generator shown in Figure 101-25	CLT: M	Yes [ ] No [ ] N/A [ ]
PI6	Pilot boosting	101.4.2.10.1	Amplitude of pilots is 2 times the RMS value of the amplitude of other subcarriers	CLT: M	Yes [ ] No [ ] N/A [ ]
PI7	US Pilot Insertion	101.4.3.8.2	Meets the requirements of 101.4.3.8.2	CNU: M	Yes [ ] No [ ] N/A [ ]

101.7.4.10 Equalization

Item	Feature	Subclause	Value/Comment	Status	Support
EQ1	Equalized subcarriers	101.4.3.11.1	All data transmissions except Probe and PHY Discovery Response. Probe equalization under control of PHY Link	CNU: M	Yes [ ] No [ ] N/A [ ]
EQ2	US Subcarrier equalization	101.4.3.11.1	Single complex coefficient per subcarrier updated by the PHY Link	CNU: M	Yes [ ] No [ ] N/A [ ]
EQ3	US Subcarrier equalization default	101.4.3.11.1	Default value of 1+j0	CNU: M	Yes [ ] No [ ] N/A [ ]
EQ4	Equalizer coefficient normalization	101.4.3.11.1	Adjusting the mean of (abs (Ck)^^{2}) to be 1	CNU: M	Yes [ ] No [ ] N/A [ ]
EQ5	Equalizer coefficient activation time	101.4.3.11.1	Within 10 ms of receipt at the beginning of a transmission	CNU: M	Yes [ ] No [ ] N/A [ ]

102. EPoC PHY Link

EDITORS NOTE (to be removed prior to publication): Probe processing needs to be pulled out of the PHY Link.

102.1 PHY Link overview and architecture

The PHY Link is a low level communications channel used between the CLT PHY and its' subtended CNU PHYs. It is used to communicate PHY OFDM channel parameters and to negotiate initialization of CNUs that wish to join or rejoin the EPoC network. A small amount of RF spectrum is dedicated to the PHY Link at network setup time for both the upstream (US) and the downstream (DS) directions (see 102.2.1 and 102.3.1). In a multi OFDM channel PHY only OFDM channel one has a PHY Link. The PHY Link uses a straightforward query response protocol with broadcast capability to transfer information in variables between the CLT and its subtended CNUs and vice versa. Both the upstream and the downstream PHY Link include a frame structure. Each frame is composed of message blocks containing timing, control information, status information, PHY Instructions or PHY Responses. The frame is padded to achieve a fixed bit length and encoded in multiple FEC codewords.

The upstream superframe (see 101.4.3.3.1) begins with the Probe Period. CNU PHY Discovery Responses and probing are performed during the Probing Period. The PHY Discovery Response is used for initial CNU bring up and is fully described in 102.4.1.4. Probing is used to perform fine ranging and periodic link maintenance tasks and is described in 102.4.2.

102.1.1 PHY Link frame structure and protocol

The PHY Link frame is illustrated in Figure 102–1 and Figure 102–2.

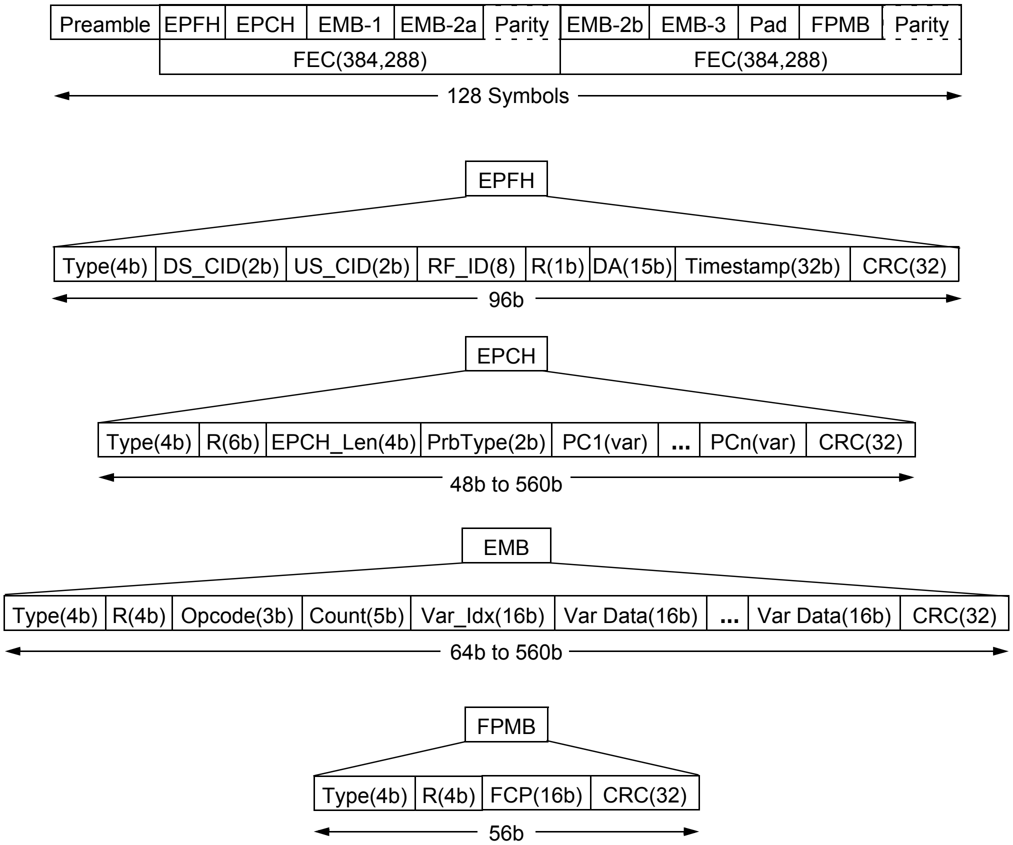


Figure 102–1—Downstream PHY Link frame

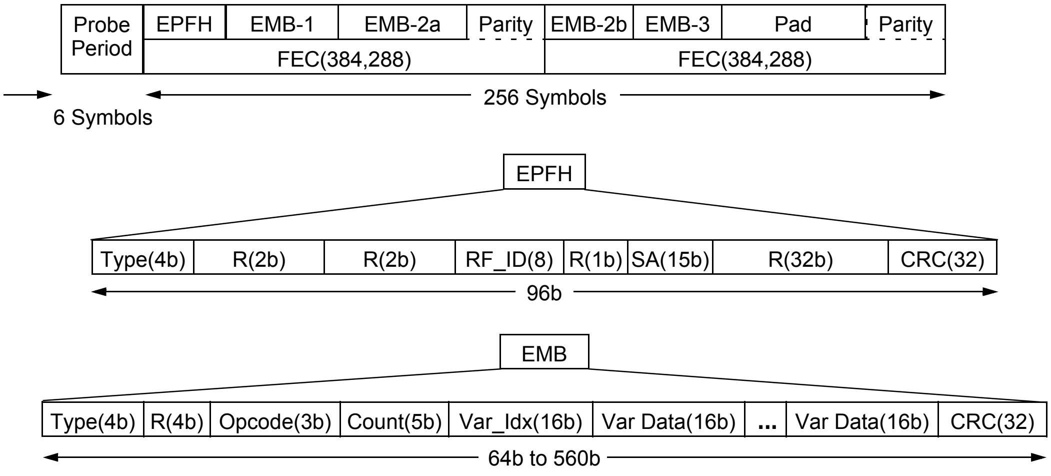


Figure 102–2—Upstream PHY Link frame

The PHY Link protocol is a query and response protocol where the CLT transmits one or more instructions enclosed in EpoC Message Blocks to a CNU or group of CNUs. Each instruction can perform a read, write or write/verify operation. Read and write/verify instructions cannot be addressed to a group of CNUs. The PHY Link frame shall be fixed; the downstream length is 128 OFDM symbols long and the upstream length is 262 OFDM symbols long.

The CLT and the CNU shall support both an upstream and a downstream PHY Link channel.

102.1.2 PHY Link block diagram

The architecture of the PHY Link data path is illustrated in Figure 102–3 and Figure 102–4. The relationship between the PHY Link functional blocks and the rest of the 10GPASS-XR PHY is illustrated in Figure 100–2 and Figure 100–3.

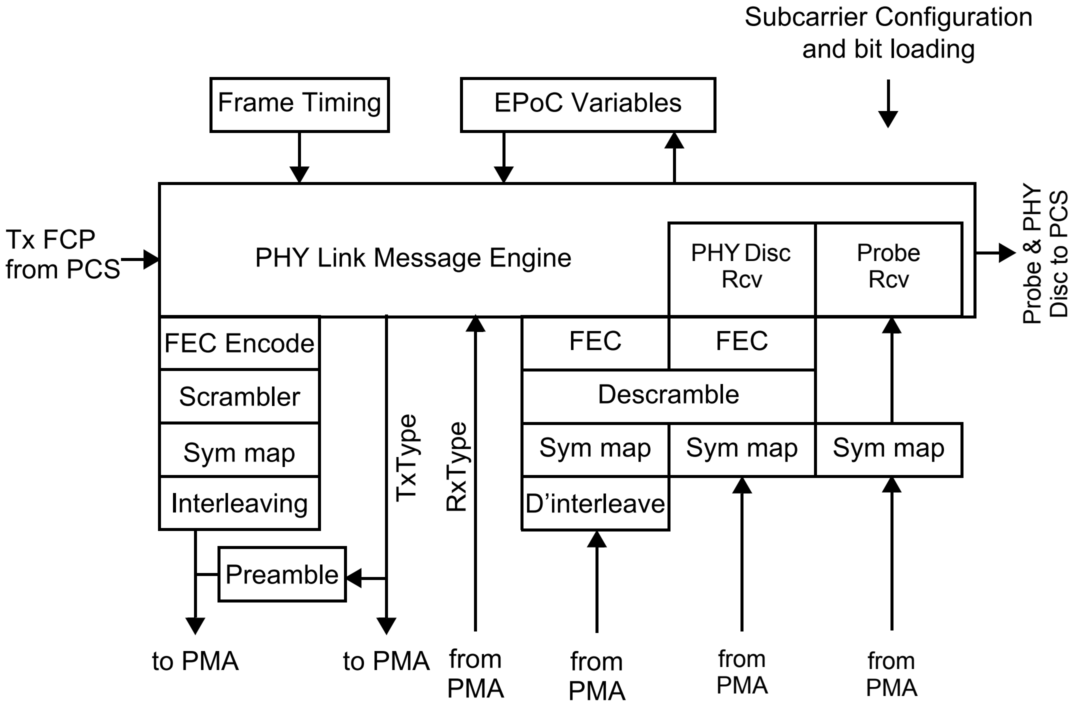


Figure 102–3—PHY Link CLT architecture

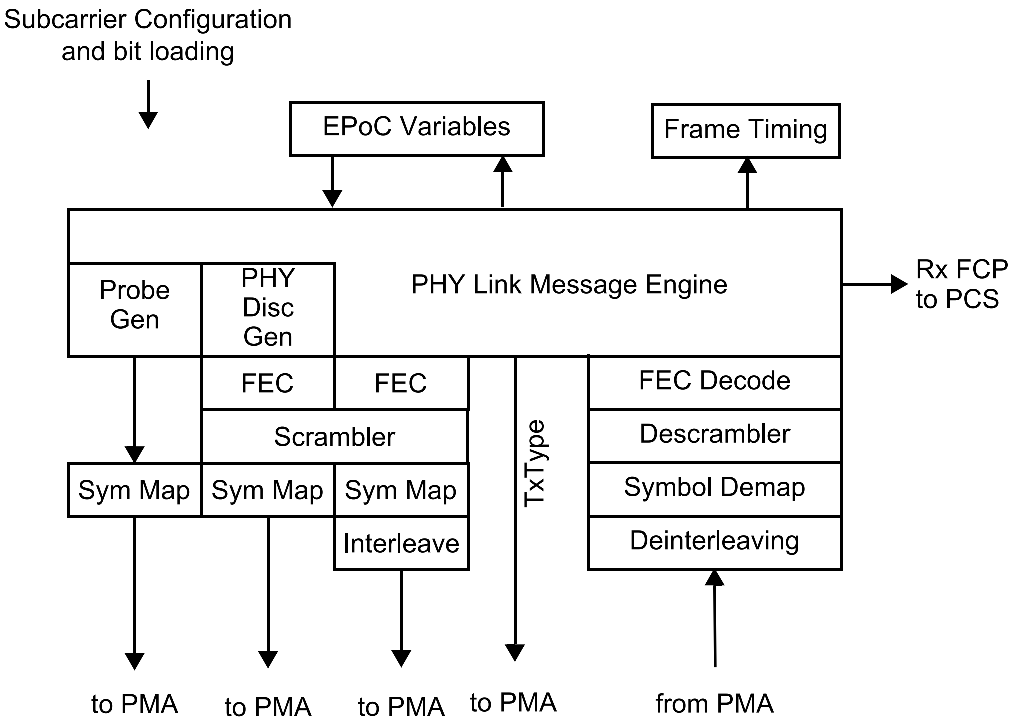


Figure 102–4—PHY Link CNU architecture

102.1.3 PHY Link Message Engine

The PHY Link Message Engine block is responsible for the origination and termination of all messages passed over the PHY Link and all PHY to PHY signaling. In the downstream direction there are four message blocks; the EPoC PHY Frame Header (EPFH), the EPoC Probe Control Header (EPCH), the EPoC message block, and the FEC Parity message block. The upstream PHY Link Message Engine also has the two additional PHY to PHY signaling types; PHY Discovery Response and Probing.

The content of each message block is detailed below as is the characteristic of the two additional PHY signaling types. The details of the PHY Message Engine behavior is described in 102.2.3 and 102.3.2.

Once a PHY Link message block has been created the stream of bytes is converted into a stream of bits, MSB first, as illustrated in Figure 102–5.

EDITORS NOTE (to be removed prior to publication): Bit mapping figure needs to be examine for correctness *and consistency with 802.3 style, especially little endian vs big endian.*

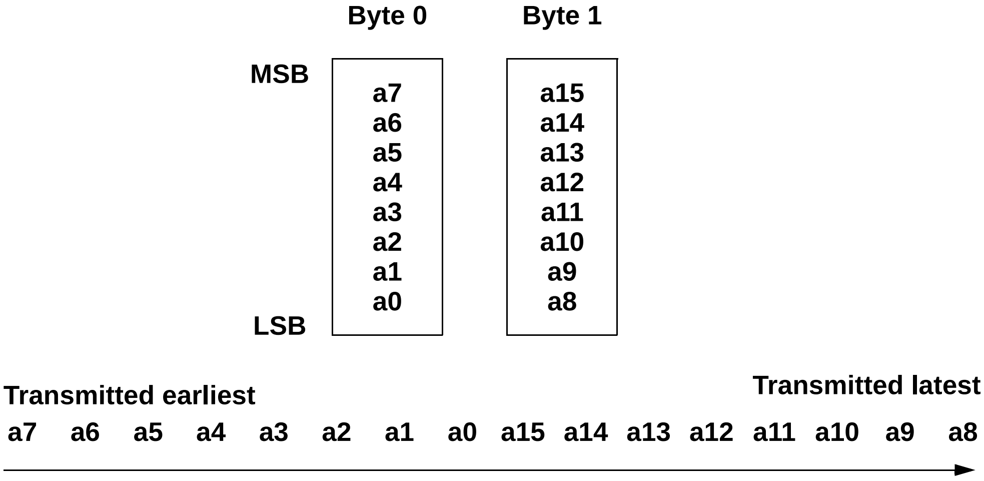


Figure 102–5—Mapping bytes into a bit stream for FEC encoding

102.1.4 PHY Link FEC encoder

The PHY Link uses several LDPC FEC encoders. These encoders are derived from mother encoders using either puncturing only or both shortening and puncturing. A general description of LDPC codes is given in 101.3.2.4.

102.1.4.1 Parity-Check matrices of mother codes

102.1.4.1.1 LDPC (480,288) mother code

The base matrix of a parity check matrix for the (480,288) mother code is listed in Table 102–1, where the sub-matrix size (lifting factor) is L = 48 (see 101.3.2.4 for the general definition of a base matrix).

Table 102–1—Base matrix of (480,288) LDPC code parity check matrix

16	1	28	9	40	38	16	-	-	-
28	42	36	11	39	9	8	38	-	-
5	2	18	16	25	47	-	2	19	-
18	18	40	18	0	34	-	-	7	32

102.1.4.1.2 LDPC (160,80) mother code

The base matrix of a parity check matrix for the (160,80) mother code is listed in Table 102–2, where the sub-matrix size (lifting factor) is L = 16 (see 101.3.2.4 for the general definition of a base matrix).

Table 102–2—Base matrix for (160, 80) DPC code parity check matrix

1	11	10	12	7	9	-	-	-	-
2	1	14	15	14	14	12	-	-	-
0	9	3	2	-	-	11	7	-	-
6	8	-	10	3	-	-	10	4	-
12	13	11	-	0	-	-	-	5	2

102.1.4.2 Shortening and puncturing encoder

Shortening encoder consists of 3 steps:

Step 1: Pad zero bits to the payload bits to fit for the codeword size of the mother code, the entire bits are called mother code information bits and the coordinates corresponded to the padded zeros are called shortening coordinates
Step 2: Encode the information bits obtained in Step 1 using mother code encoder
Step 3: Delete the shortening coordinates: i.e. all the padded zero bits in Step 1.

Puncturing encoder consists of 2 steps:

Step 1: Encode the payload (with or without padding) bits using mother encoder.
Step 2: Delete the puncturing coordinates of encoded codeword of Step 1, where the puncturing coordinates are given coordinates of the mother code codeword.

102.1.4.2.1 LDPC (384,288) puncturing encoder

The LDPC (384,288) encoder is operated on the (480, 288) LDPC mother code encoder with the puncturing.

The mother code is defined in 102.1.4.1.1. The puncturing operation is as follows (also see Figure 102–6):

Denote the information bits sent to the mother code encoder by (a_0, ... , a_{287})

Let the encoding output be (a_0, ... , a_{287}, b_{288}, ... , b_{479}), where (b_{288}, ... , b_{479}) are parity-check bits.

The coordinates to be deleted by the puncturing step are:

Period 1: 48 consecutive coordinates a_{48}, ... , a_{95}
Period 2: 48 consecutive coordinates b_{384}, ... , b_{431}

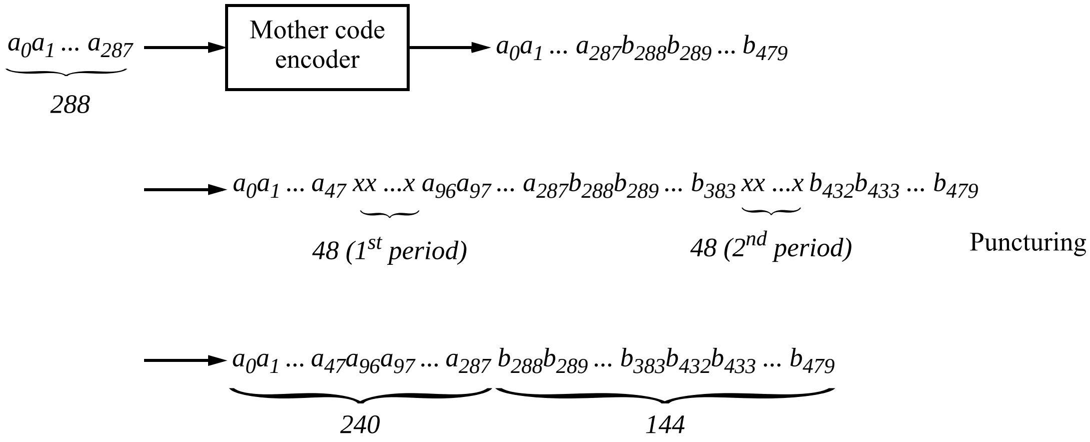

$\underbrace{a_0a_1 \dots a_{287}}_{288}$
Mother code encoder
$a_0a_1 \dots a_{287}b_{288}b_{289} \dots b_{479}$
$a_0a_1 \dots a_{47}\,\underbrace{xx \dots x}_{48\,(1^{st}\,period)}\,a_{96}a_{97} \dots a_{287}b_{288}b_{289} \dots b_{383}\,\underbrace{xx \dots x}_{48\,(2^{nd}\,period)}\,b_{432}b_{433} \dots b_{479}$
Puncturing
$\underbrace{a_0a_1 \dots a_{47}a_{96}a_{97} \dots a_{287}}_{240}\,\underbrace{b_{288}b_{289} \dots b_{383}b_{432}b_{433} \dots b_{479}}_{144}$

Figure 102–6—Puncturing encoder for (384,288) LDPC FEC

EDITORS NOTE (to be removed prior to publication): In draft 1.0 the figure above was redrawn in native FrameMaker format, original authors are advised to review.

102.1.4.2.2 LDPC (128,80) puncturing encoder

The LDPC (128,80) encoder is operated on the (160, 80) LDPC mother code encoder with the puncturing.

The mother code is defined in 102.1.4.1.2

The puncturing operation is as follows (also see Figure 102–7):

Denote the information bits sent to the mother code encoder by ($a_0, \dots , a_{79}$) and let the encoding output be ($a_0, \dots , a_{79}, b_{80}, \dots , b_{159}$), where ($b_{80}, \dots , b_{159}$) are parity-check bits. Then do puncturing on the puncturing coordinates given by

Period 1: 16 consecutive coordinates $a_0, \dots , a_{15}$
Period 2: 16 consecutive coordinates $b_{144}, \dots , b_{159}$

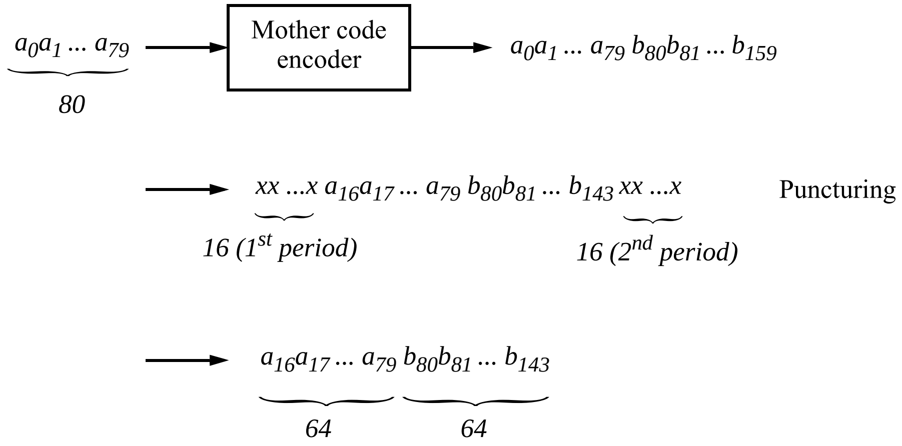

$\underbrace{a_0a_1 \ldots a_{79}}_{80}$
Mother code encoder
$a_0a_1 \ldots a_{79}\, b_{80}b_{81} \ldots b_{159}$
$\underbrace{xx \ldots x}_{16\ (1^{st}\ period)}\, a_{16}a_{17} \ldots a_{79}\, b_{80}b_{81} \ldots b_{143}\, \underbrace{xx \ldots x}_{16\ (2^{nd}\ period)}$
Puncturing
$\underbrace{a_{16}a_{17} \ldots a_{79}}_{64}\, \underbrace{b_{80}b_{81} \ldots b_{143}}_{64}$

Figure 102–7—Puncturing encoder for the (128, 80) LDPC FEC

102.1.5 PHY Link scrambler

The PHY shall scramble the output of the PHY Link FEC encoding process using a linear feedback shift register mechanism as illustrated in Figure 101–19.

The scrambler is defined by the following polynomial.

$$x^{23} + x^{18} + 1$$

The scrambler is initialized to the hexadecimal value of 0x4732BA. The PHY initializes the scrambler with the hexadecimal value at the beginning of the first OFDM symbol following the PHY Link preamble in the downstream direction. In the upstream direction the PHY initializes the scrambler at the beginning of an upstream PHY Link transmission.

The PHY does not scramble the PHY Link preamble.

102.1.6 PHY Link symbol map and constellation mapping

The PHY maps the scrambled bit stream of normal PHY Link data into a complex number using the assigned modulation order. In the downstream direction the assigned modulation order is always 16-QAM and uses the mapping shown in 101.4.4. The upstream PHY Link may use 16-QAM or a higher order modulation (see 101.4.4 for mapping structure). The PHY multiplies the real and imaginary parts by the appropriate factor in Table 101–20 to ensure that mean-square value of the QAM constellation is unity.

102.1.7 Interleaving

Data in the PHY Link channel is time interleaved. For the downstream direction this time interleaving is described in 102.2.1.3. while it is described in 102.3.1.3 for the upstream direction.

Transmissions in the upstream direction for PHY Discovery Response are not time interleaved.

Control for the interleaving process is conveyed using the TxType in the CNU (see Figure 102–4) and RxType in the CLT (see Figure 102–3).

102.1.8 Mapping of PHY Link variables

The optional MDIO capability described in Clause 45 defines several variables that may provide control and status information for and about the PHY Link or are communicated between CLT and CNU via the PHY Link. Mapping of MDIO control and status variables to PHY Link variables is shown in Table 102–3.

EDITORS NOTE (to be removed prior to publication): not all variables need to be included in Cl 45. We need to determine how to index variables that need to be communicated over the PHY Link that are not included in Cl 45. Current "rule" is:

*If 1.1900 <= RegAdd <=1.1999 Then Index = RegAdd - 1.1900)*1000) (i.e., 0-99)*
46 indexes in this range were in use as of Draft 1.4.
*If 12.0000 <= RegAdd Then Index = (RegAdd - 12.0000)*1000 + 1000 (i.e., 1000 +)*
12287 indexes in this range are in use as of Draft 1.4
If variable is not in Cl 45 use indexes 500-999

Table 102–3—10GPASS-XR MDIO/PHY Link variable mapping

MDIO Variable	PMA/PMD register name	Register/ bit Number	PMA/PMD variable		
			Name	Index	Bit(s)
PHY Discovery complete	10GPASS-XR control	1.1900.1	*PhyDiscCmplt*	0	1
Transmit enable	10GPASS-XR control	1.1900.0	*TxEnable*	0	0
DS windowing	DS OFDM control	1.1901.6:4	*DSNrp*	1	6:4
DS cyclic prefix	DS OFDM control	1.1901.3:0	*DSNcp*	1	3:0
DS OFDM freq	DS OFDM channel frequency control 1	1.1902.15:0	*DS_FreqCh1*	2	15:0
DS OFDM freq	DS OFDM channel frequency control 1	1.1903.15:0	*DS_FreqCh2*	3	15:0
DS OFDM freq	DS OFDM channel frequency control 3	1.1904.15:0	*DS_FreqCh3*	4	15:0
DS OFDM freq	DS OFDM channel frequency control 4	1.1905.15:0	*DS_FreqCh4*	5	15:0
DS OFDM freq	DS OFDM channel frequency control 5	1.1906.15:0	*DS_FreqCh5*	6	15:0
Rnd	US OFDM control	1.1907.15:8	*Rnd*	7	15:8
RB Size	US OFDM control	1.1907.7	*RBsize*	7	7
US windowing	US OFDM control	1.1907.6:4	*USNrp*	7	6:4
US cyclic prefix	US OFDM control	1.1907.3:0	*USNcp*	7	3:0
US OFDM freq	US OFDM channel frequency control	1.1908.15:0	*US_FreqCh1*	8	15:0
DS PHY Link Start	DS PHY Link control	1.1911.11:0	*DS_PhyLinkStrt*	11	11:0

Table 102–3—10GPASS-XR MDIO/PHY Link variable mapping *(continued)*

MDIO Variable	PMA/PMD register name	Register/ bit Number	PMA/PMD variable		
			Name	Index	Bit(s)
US PHY Link start	US PHY Link control	1.1912.11:0	*US_PhyLinkStrt*	12	11:0
PHY Discovery start lower	PHY Discovery control	1.1913.15:0	*DiscStrt* (31:16)	13	15:0
PHY Discovery start upper	PHY Discovery control	1.1914.15:0	*DiscStrt* (31:16)	14	15:0
Assigned CNU_ID flag	New CNU control	1.1915.15	*AssgndCNU_ID*	15	15
Allowed CNU_ID	New CNU control	1.1915.14:0	*AllwdCNU_ID*	15	14:0
New CNU Range	New CNU info	1.1916.15:0	*NewCNU_Rng*	16	15:0
New CNU MAC 0	New CNU info	1.1917.15:0	*New_MAC* (15:0)	17	15:0
New CNU MAC 1	New CNU info	1.1918.15:0	*New_MAC* (31:16)	18	15:0
New CNU MAC 2	New CNU info	1.1919.15:0	*New_MAC* (47:32)	19	15:0
PHY timing offset lower	PHY timing offset	1.1922.15:0	*PhyTimingOffset* (15:0)	22	15:0
PHY timing offset upper	PHY timing offset	1.1923.15:0	*PhyTimingOffset* (31:16)	23	15:0
PHY power offset	PHY power offset	1.1924.7:0	*PHYPowerOffset*	24	7:0
PHY Ranging offset lower	PHY ranging offset	1.1925.15:0	*PhyRngOffset* (15:0)	25	15:0
PHY Ranging offset Upper	PHY ranging offset	1.1926.15:0	*PhyRngOffset* (31:16)	26	15:0
DS PHY Data Rate lower	DS PHY Data Rate	1.1927.15:0	*DS_DataRate (15:0)*	27	15:0
DS PHY Data Rate mid	DS PHY Data Rate	1.1928.15:0	*DS_DataRate (31:16)*	28	15:0
DS PHY Data Rate upper	DS PHY Data Rate	1.1929.4:0	*DS_DataRate (36:32)*	29	4:0
US PHY Data Rate lower	US PHY Data Rate	1.1930	*US_DataRate (15:0)*	30	15:0
US PHY Data Rate mid	US PHY Data Rate	1.1930.15:0	*US_DataRate (31:16)*	31	15:0
US PHY Data Rate upper	US PHY Data Rate	1.1931.4:0	*US_DataRate (36:32)*	32	4:0
PHY Link EPFH counter	PHY Link EPFH counter	1.1932.15:0	*EPFHcnt*	39	15:0
PHY Link EPFH error counter	PHY Link EPFH error counter	1.1940.15:0	*EPCHcnt*	40	15:0
PHY Link EPCH counter	PHY Link EPCH counter	1.194115:0	*EMBcnt*	41	15:0

Table 102–3—10GPASS-XR MDIO/PHY Link variable mapping *(continued)*

MDIO Variable	PMA/PMD register name	Register/ bit Number	PMA/PMD variable		
			Name	**Index**	**Bit(s)**
PHY Link EPCH error counter	PHY Link EPCH error counter	1.1942.15:0	*FPMBcnt*	42	15:0
PHY Link EMB counter	PHY Link EMB counter	1.1943.15:0	*EPFHerr*	43	15:0
PHY Link EMB error counter	PHY Link EMB error counter	1.1944.15:0	*EPCHerr*	44	15:0
PHY Link FPMB counter	PHY Link FPMB counter	1.1945.15:0	*EMBerr*	45	15:0
PHY Link FPMB error counter	PHY Link FPMB error counter	1.1946.15:0	*FPMBerr*	46	15:0
DS OFDM channel ID	DS OFDM channel ID	*12.0.2:0*	*DS_OFDM_ID*	100	2:0
DS modulation type SC4	10GPASS-XR DS profile descriptor control 1	12.1.3:0	*DS_ModTypeSC(4)*	101	3:0
DS modulation type SC5	10GPASS-XR DS profile descriptor control 1	12.1.7:4	*DS_ModTypeSC(5)*	1001	7:4
DS modulation type SC6	10GPASS-XR DS profile descriptor control 11	12.1.11:8	*DS_ModTypeSC(6)*	1001	11:8
DS modulation type SC7	10GPASS-XR DS profile descriptor control 1	12.1.15:12	*DS_ModTypeSC(7)*	1001	15:12
DS modulation type SC8 through DS modulation type SC 4095	10GPASS-XR DS profile descriptor control 2 through 0GPASS-XR DS profile descriptor control 1023	12.2 through 12.1023	*DS_ModTypeSC(8)* through *DS_ModTypeSC(4095)*	1002 to 2023	
US modulation type SC0	10GPASS-XR US profile descriptor control 0	12.1024.3:0	*US_ModTypeSC(0)*	2024	3:0
US modulation type SC1	10GPASS-XR US profile descriptor control 0	12.1024.7:4	*US_ModTypeSC(1)*	2024	7:4
US modulation type SC2	10GPASS-XR US profile descriptor control 0	12.1024.11:8	*US_ModTypeSC(2)*	2024	11:8
US modulation type SC3	10GPASS-XR US profile descriptor control 0	12.1024.15:12	*US_ModTypeSC(3)*	2024	15:12

Table 102–3—10GPASS-XR MDIO/PHY Link variable mapping *(continued)*

MDIO Variable	PMA/PMD register name	Register/ bit Number	PMA/PMD variable		
			Name	**Index**	**Bit(s)**
US modulation type SC4 through US modulation type SC 4095	10GPASS-XR US profile descriptor control 1 through 10GPASS-XR US profile descriptor control 1023	12.1025 through 12.2047	*US_ModTypeSC(4)* through *US_ModTypeSC(4095)*	2025 to 3047	
Real pre-equalizer coefficient SC(0)	10GPASS-XR US pre-equalizer coefficients	12.2048.15:0	*EQ_CoefR(0)*	3048	15:0
Imaginary pre-equalizer coefficient SC(0)	10GPASS-XR US pre-equalizer coefficients	12.2049.15:0	*EQ_CoefI(0)*	3049	15:0
Real pre-equalizer coefficient SC(1) through Real pre-equalizer coefficient SC(4095)	10GPASS-XR US pre-equalizer coefficients	Even registers 12.2050 through 12.10238	*EQ_CoefR(1)* through *EQ_CoefR(4095)*	Even indexes 3050 to 11238	15:0
Imaginary pre-equalizer coefficient SC(1) through Imaginary pre-equalizer coefficient SC(4095)	10GPASS-XR US pre-equalizer coefficients	Odd registers 12.4051 through 12.10239	*EQ_CoefI(1)* throuigh *EQ_CoefI(4095)*	Odd indexes 3051 to 11239	15:0

102.2 Downstream PHY Link

102.2.1 DS PHY Link physical layer

102.2.1.1 Resource allocation

During network setup the downstream PHY Link shall be allocated 400 kHz of spectrum. The allocated spectrum for the downstream PHY Link shall reside anywhere within a 24 MHz contiguous OFDM channel spectrum (i.e., 24 MHz with no internal exclusion bands) and have at least 3 MHz of contiguous spectrum above and below it for a total band of 6 MHz. This PHY Link band also includes eight pilot tone subcarriers placed symmetrically above and below the information subcarriers as illustrated in Figure 102–8; see 101.4.2.6.3 for exact placement of pilots. No additional continuous pilots are allowed within this 6 MHz band (see 101.4.2.6). However, scattered pilots are allowed in this spectrum in subcarriers that normally carry MAC data. The downstream PHY Link is located per the *DS_PhyLinkStrt* variable (see DS PHY Link Start parameter, 45.2.1.138) that determines the lowest frequency subcarrier of the PHY Link information channel. The downstream PHY Link shall use the same OFDM Symbol size and cyclic prefix duration as the downstream MAC data channel.

.

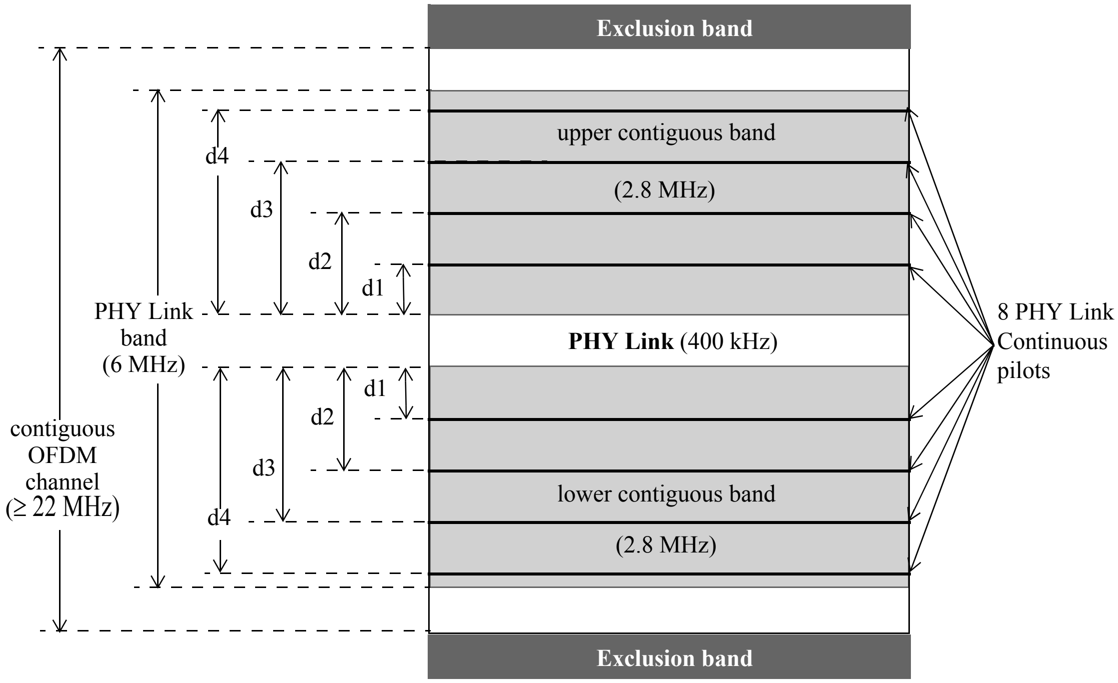


Figure 102–8—DS PHY Link spectrum placement

102.2.1.2 DS PHY Link modulation

The downstream PHY Link uses a 16-QAM constellation for all information subcarriers as specified under PHY Link CLT Tx / CNU Rx in Table 100–2.

102.2.1.3 DS PHY Link subcarrier block interleaving

Figure 102–6 shows 288 data bits entering the LDPC encoder and 384 encoded bits exiting it. This sequence is in effect time-reverse ordered. The time-ordered sequence takes the form shown in Figure 102–9. The PHY shall map the 384 FEC encoded data bits, as processed by the scrambler, to 96 4-bit nibbles {u_i, i=0, 1, … , 95} as shown in Figure 102–9.

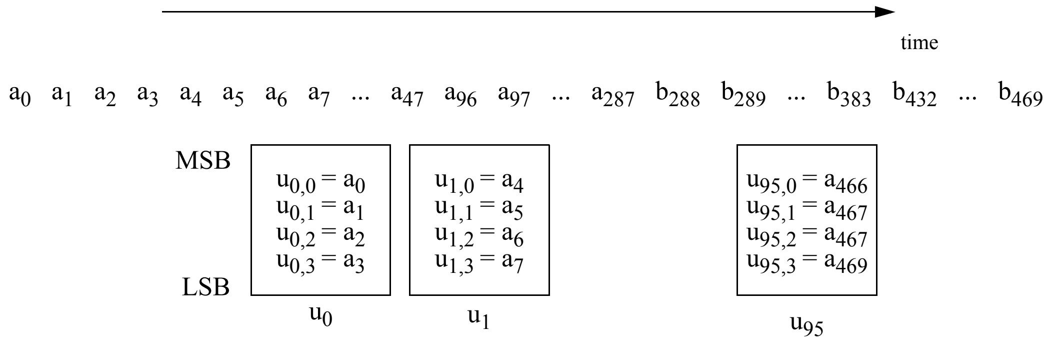


Figure 102–9—DS PHY Link mapping of FEC output to stream nibbles

The resulting 4-bit nibbles from this mapping operation are then time interleaved. The PHY shall interleave the 96-nibble sequence $\{u_0\ u_1\ u_2\ldots\ u_{95}\}$ as illustrated in Figure 102–10.

Conceptually, the Phy uses an 8 x 2 array to perform interleaving. The PHY writes the values u_i along the rows of this two-dimensional array, as shown in Figure 102–10. The PHY reads this two-dimensional array along vertical columns to form the two-dimensional sequence $\{v_{t,f},\ t=0,1,\ldots,11 \text{ and } f=0,1,\ldots,7\}$. This operation is mathematically represented as:

$$v_{t,f} = u_{t+12f} \qquad (102\text{–}1)$$

The PHY maps each of the 8-point sequences given below to the 8 successive PHY Link subcarriers of an OFDM symbol after scrambling as described in the 102.1.5.

$$V_t = \{v_{t,f},\ f=0,1,\ldots,7\} \text{ for 12 successive OFDM symbols } t = 0, 1, \ldots, 11 \qquad (102\text{–}2)$$

Therefore each FEC codeword will occupy the PHY Link segment of twelve successive OFDM symbols. There will be ten such codewords in a 128-symbol PHY Link frame, including the 8-symbol preamble.

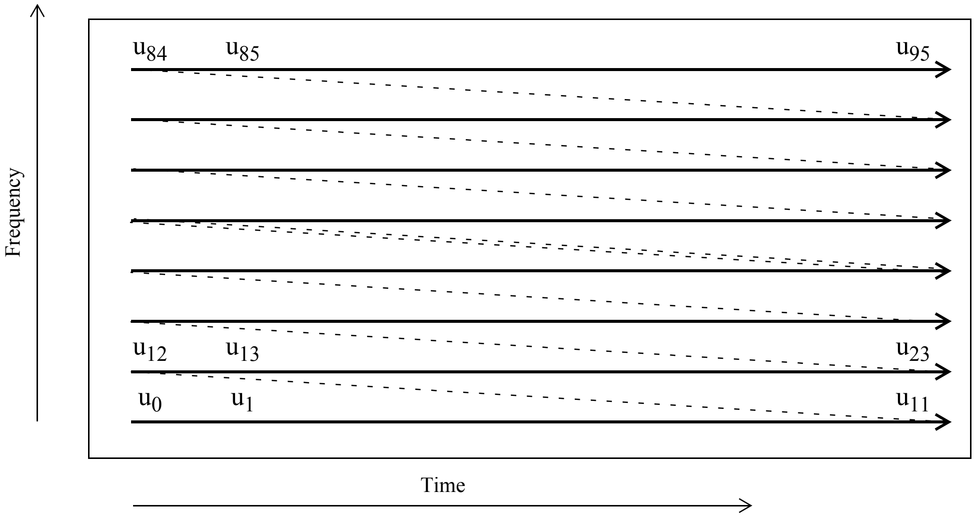


Figure 102–10—DS PHY Link subcarrier block interleaving

The PHY Link is time interleaved separately from the MAC data channel. The PHY Link preamble is not time interleaved.

102.2.2 DS preamble

The downstream Preamble shall be a fixed pattern of 64 bits as illustrated in Table 102–4, modulated using binary phase-shift keying (BPSK), that fill the first eight symbols of the PHY Link frame. The pattern is selected to enable the CNU to easily ascertain the PHY Link subcarrier s without prior knowledge of the PHY Link's precise location in the RF spectrum range of an EPoC network. Detection of the PHY Link is the first action a CNU must take to join an EPoC network. Detection of the PHY Link is the first action a CNU must take to join an EPoC network.

The CLT maps each of the binary bits shown in Table 102–4 to a BPSK constellation point in the complex plane using the following transformation:

0 → (1 + j0)
1 → (-1 + j0).

Table 102–5 is provided for information purposes and illustrates the receiver processing of the PHY Link preamble.

Table 102–4—DS PHY Link 2-D preamble

subcarrier	Symbol							
	1	2	3	4	5	6	7	8
8	1	0	1	0	0	0	1	1
7	1	0	1	0	0	0	1	1

Table 102–4—DS PHY Link 2-D preamble *(continued)*

subcarrier	Symbol							
	1	2	3	4	5	6	7	8
6	1	0	1	0	0	0	1	1
5	0	0	0	0	1	0	0	1
4	1	0	1	0	0	0	1	1
3	0	0	0	0	1	0	0	1
2	0	0	0	0	1	0	0	1
1	1	0	1	0	0	0	1	1

Table 102–5—DS PHY Link differential demodulation sequence

subcarrier	Symbol							
	1	2	3	4	5	6	7	8
8	X	1	1	1	0	0	1	0
7	X	1	1	1	0	0	1	0
6	X	1	1	1	0	0	1	0
5	X	0	0	0	1	1	0	1
4	X	1	1	1	0	0	1	0
3	X	0	0	0	1	1	0	1
2	X	0	0	0	1	1	0	1
1	X	1	1	1	0	0	1	0

102.2.3 DS frame

The downstream PHY Link uses a frame format, illustrated in Figure 102–1, to which the 128 symbol staggered pilot pattern is aligned with as described in 101.4.2.6. The 128 symbol downstream PHY Link frame is composed of a Preamble (see 102.2.2), one 96-bit EPoC PHY Frame Header (EPFH), one variable length EPoC Probe Control Header (EPCH), some number of variable length EPoC message blocks (EMB), one 56-bit FEC Parity message block (FPMB), and padding. The message blocks within the frame are protected with a FEC mechanism. Each message block and it's included fields is described below. The number of optional EPoC message blocks contained within the frame is limited only by the frame size.

Each message block contains a Type field used to identify the contents of the block. CLTs shall use the appropriate message Type fields listed in Table 102–6 in each message block. The contents of the each message block is protected by a CRC(32). See 3.2.9 for a description of how this field is calculated. The

CNU shall calculate a CRC(32) on the data fields within each message block received and, if the calculated CRC(32) does not match the received CRC(32) discard the message and take no action based on it.

Table 102–6—PHY Link message block Type values

Type value	message block
0x00- 0x08	reserved
0x09	EPoC PHY Frame Header
0x0A	EPoC Probe Control Header
0x0B	EPoC message block
0x0C	FEC Parity message block
0x0D to 0x0F	reserved

EDITORS NOTE (to be removed prior to publication): if we need to include message block Types defined in D3.1 the above table will need modification. D3.1 include the following:
Type 1 = Time Stamp MB with 24b CRC
Type 2 = Energy Management MB
Type 3 = MC MB
Type 4 = Trigger MBDS

102.2.3.1 DS EPoC PHY Frame Header

The downstream EPoC PHY Frame Header includes a Type field, the Configuration ID fields (*DS_CID* and *US_CID*), the Return Frame ID field (RF_ID), the PHY Link DA field, the PHY Timestamp field, and a CRC(32) as illustrated in Figure 102–1.

102.2.3.1.1 Configuration ID and profile activation.

The Configuration ID fields are two bit fields used to inform a CNU, to switch from one modulation profile to another. There is one field for control of the downstream profile (*DS_CID*) and one field for the upstream profile (*US_CID*). Each CNU contains two profiles in each direction, copy “A” and copy “B”; only one of which is active at any given time. The CLT shall ensure that the inactive profile in all CNUs is identical prior to making it the active profile. The CLT updates the unused profile then, using the PHY Configuration ID field, switches the CNU to the updated profile. Once the CLT begins the switchover, as indicated by Configuration ID field values 0b01 or 0b10 it shall complete the switchover. During a switchover the value of the Configuration ID field is either incremented or decremented by one in each successive frame; thus a switchover takes three PHY Link frame times. Table 102–7 summarizes the use and meaning of the PHY Configuration ID bits and their operation is illustrated in Figure 102–11.

102.2.3.1.2 Response Frame ID

In the downstream direction the new profile is activated at the first symbol (i.e., the symbol containing the PHY Link Preamble) in the next PHY Link frame. In the Upstream direction the new profile is activated in the first symbol of the Probe Period following the frame identified by the Return Frame ID field.

EDITORS NOTE (to be removed prior to publication): Definitions and normative text needed to explain and define the modulation profile and its' requirements.

Table 102–7—Configuration ID bits

PHY Configuration ID		
bit 1	**bit 0**	**Meaning**
0	0	Copy “A” in use
0	1	prepare for switchover
1	0	
1	1	Copy “B” in use

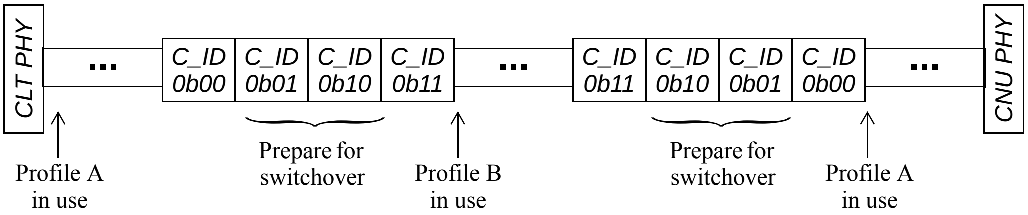


Figure 102–11—Configuration ID bit usage

The Response Frame ID field is an 8-bit field which indicates to the receiving CNU which RB Frame to use for the response message to this frame. The CLT shall ensure that all CNUs have sufficient time (as determined by the variable PhyLnkRspTm) to respond to the downstream PHY Link frame.

102.2.3.1.3 PHY Link DA

The PHY Link DA is an address field that identifies the CNU that any EPoC Message Blocks in that PHY frame are targeted for. This field is 15 bits and may be a unicast address or a broadcast address (see Table 102–8). In the CNU if the DA does not match the assigned address (CNU_ID) or the broadcast address then the EMBs in the frame are discarded and no response is made. The CLT shall only transmit the valid values of the PHY DA as given in Table 102–8.

Table 102–8—PHY DA field values

Field Values	Use
0x7F00 .. 0x7FF0	Broadcast addresses
0x7EFF ..0x0001	Unicast CNU addresses
0x0000	CLT PHY DA address

102.2.3.1.4 PHY Timestamp

The PHY Timestamp is a 32 bit field which the EPoC PHY uses to synchronize the US PHY Link frame and ODFMA symbols. The LocalTS_ctr, from which this field is set, is clocked from the 204.8 MHz OFDM clock. When a CNU PHY that has *TxEnable* equal to FALSE receives a PHY Frame addressed to it or to the broadcast address it shall reset it's LocalTS_ctr to the value in the Timestamp. The reference point for the Timestamp shall be the first sample of the PHY Link symbol immediately following the Preamble (Figure 102–12). For additional information on the use of the Timestamp see 101.3.3.1.3.

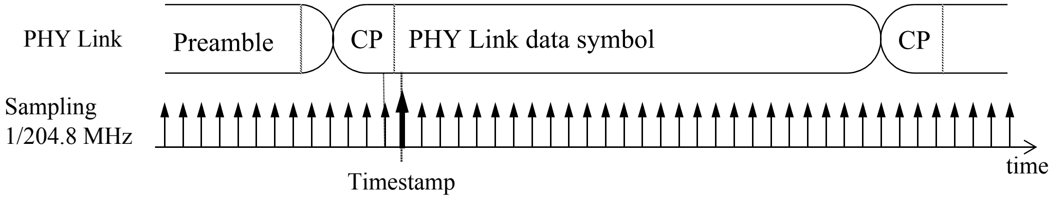


Figure 102–12—PHY Link Timestamp reference point.

The CNU PHY Link receiver maintains counts of EPoC PHY Frame Headers received and those received that have CRC-32 errors using the variables *EPFHcnt* and *EPFHerr*, respectively.

102.2.3.2 EPoC Probe Control Header message block

The EPoC Probe Control Header includes a Type field, the EPCH_Len field, the PrbType field, some number of Probe Control (PCn) fields, and a CRC(32) as illustrated in Figure 102–1.

The EPCH_Len field is a 4-bit number that conveys the number of Probe Control fields contained in the message block. Each EPoC Probe Control Header may contain 0 to 15 Probe Control fields.

The PrbType field is a 2-bit field that conveys the type of Probe Control fields contained in the message block. There are three types of Probe Control fields; a Probe Scheduling type, a broadcast PHY Discovery type and a Unicast PHY Discovery type as shown in Table 102–9.

Table 102–9—PrbType values

PrbType value	Probe Control field	Probe Control length
0x00b	Probe Scheduling	32-bits
0x01b	Broadcast PHY Discovery	16-bits
0x10b	Unicast PHY Discovery	64-bits
0x11b	Reserved	n/a

The CNU PHY Link receiver maintains counts of EPoC Probe Control Headers received and those received that have CRC-32 errors using the variables *EPCHcnt* and *EPCHerr,* respectively.

102.2.3.2.1 Probe Scheduling type Probe Control fields

The Probe Scheduling type Probe Control fields each enable one CNU to participate in the first Probe Period of the upstream frame following the Response Frame ID. Each Probe Scheduling field contains a PrbID, PrbStrtSC, PrbSkip, PrbEQ, StrtSym, SymNum, and several reserved (R) subfields as illustrated in Figure 102–13. To enable a CNU to participate in a Probe Period the CNU_ID of the participating CNU is placed in PrbID subfield of the Probe Control. The remaining subfields set the corresponding variables in the designated CNU. A Probe Control field set to all zeros or any broadcast CNU_ID is ignored and does not enable any CNU. For additional information on use of the Probe Control fields see 102.4.2.3.

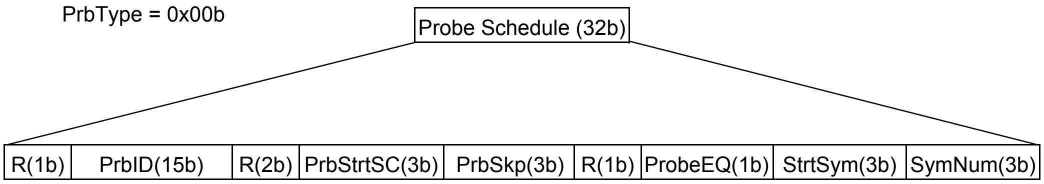


Figure 102–13—Probe Scheduling type Probe Control field

102.2.3.2.2 Broadcast PHY Discovery type Probe Control fields

The Broadcast PHY Discovery type Probe Control fields each enable one PHY Discovery window within the Probe Period of the upstream frame following the Response Frame ID. Each Broadcast PHY Discovery field contains a PHY Discovery window descriptor (PHY Disc_N) which designates the starting subcarrier of the PHY Discovery window as illustrated in Figure 102–14. For additional information on use of the Broadcast PHY Discovery type Probe Control fields see 102.4.1.3.

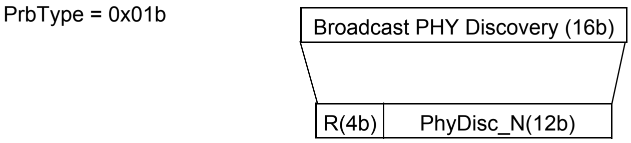


Figure 102–14—Broadcast PHY Discovery type Probe Control field

102.2.3.2.3 Unicast PHY Discovery type Probe Control fields

The Unicast PHY Discovery type Probe Control fields each enable a single CNU, as designated by a MAC address, to use a unique PHY Discovery window within the Probe Period of the upstream frame following the Response Frame ID. Each Unicast PHY Discovery field contains a MAC address and a PHY Discovery window descriptor (PHY Disc_N) which designates the starting subcarrier of the PHY Discovery window as illustrated in Figure 102–14. For additional information on use of the Broadcast PHY Discovery type Probe Control fields see 102.4.1.3.

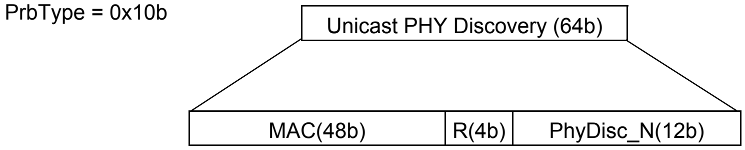


Figure 102–15—Unicast PHY Discovery type Probe Control field

102.2.3.3 DS EPoC message block

The downstream EPoC message block contains a Type field (see Table 102–6) a PHY Instruction and a CRC(32).

The CLT can perform read and write operations on the EPoC Variables of subtended CNUs via PHY Instructions. Each instruction contains an OPCODE, a Variable Group Count, a Variable Index and up to 31 Variable Group Data fields.

The PHY Link OPCODE is a 3 bit field that conveys the operation of the PHY Instruction in which it resides. The CLT shall only transmit the valid values of the PHY DA and OPCODE fields as given in Table 102–8 and Table 102–10, respectively.

Table 102–10—Valid OPCODE values

Value	Operation	Description
b000	NOP	no operation, the CNU acknowledges this instruction only
b001	Read	The CNU responds with the contents of variables starting at the index specified by the Variable Group Index field and subsequent variables as specified by the Variable Group Count field
b010	Write	the CNU stores the values given in the data field starting at the variable given in the Variable Group Index field and subsequent variables as specified by the Variable Group Count field
b011	Write/Verify	the CNU first writes the variables as specified by Variable Group Index and Variable Group Count field and then responds with the values contained in those same variables
b1xx	reserved	no operation, the CNU ignores these instructions

The Variable Group Count field specifies the number of 16-bit EPoC Variable groups contained in a write or write/verify PHY Instruction or the number of 16-bit EPoC Variable groups that are to be returned in a read or write/verify PHY Instruction. The Variable Group Count field has a value of 0 to 31. The Variable Group Count field in a NOP instruction always has a value of 0. Whereas, for a read, write or write/verify instruction the Variable Group Count always has a value between 1 and 31.

The 16-bit EPoC Variable Index field specifies the variable group at which the CNU is to begin the read, write or write/verify operation. The NOP instruction does not include an Index field.

The 16-bit Variable Group Data fields contain the data values to be written in consecutive variable groups starting with the group indicated by the Variable Index field and continuing for the number of groups indicated by Variable Group Count of the target CNU. The Variable Group Data fields are valid only for a write or write/verify PHY Instructions; or in the response to a read or write/verify PHY Instruction. In the event there is a discrepancy between the Variable Group Count field and the number of 16-bit Variable Group Data fields the CNU shall write no data to it's variables and returns a Nack indication (see 102.3.2.2).

The CNU PHY Link receiver maintains counts of EPoC message blocks received and those received that have CRC-32 errors using the variables *EMBcnt* and *EMBerr,* respectively.

102.2.3.4 DS Padding

The Pad field is used to fill the PHY frame in the event there are unused bits after the message blocks and FEC fields have been populated. The Pad field consists of all zeros and is ignored upon receipt.

102.2.3.5 DS FEC Parity message block

The FEC Parity message block includes a Type field, the FEC Codeword pointer and a CRC(32). The FEC Codeword Pointer (FCP) is a 16-bit field used to identify the start of the first FEC codeword in the next PHY Link frame.

The CNU PHY Link receiver maintains counts of FEC Parity message blocks received and those received that have CRC-32 errors using the variables *FPMEcnt* and *FPMEerr,* respectively.

102.2.4 DS PHY Link FEC

The downstream PHY Link shall use a (384,288) binary punctured LDPC code described in 102.1.4.1.1 and 102.1.4.2.1.

102.2.5 Downstream PHY Link response time.

The CNU shall decode and be capable of acting on instructions included in a downstream PHY Link frame within 4.8 ms.

EDITORS NOTE (to be removed prior to publication): We might want to consider creating a variable that the CNU can pass to the CLT to indicate what it's min response time is if it can be shorter than this. For example:
US_PlnkRspTm
TYPE: 16-bit integer
This read only variable indicates the PHYs minimum response time to a downstream PHY Link instruction in units of 16/204.8 MHz. The maximum value for this variable is 61440 (4.8 ms).
A complementary register may be defined in Cl 45.

102.2.6 DS State Diagrams

102.2.6.1 Constants

EMB

TYPE: 4-bit integer
This constant represents the PHY Link message type for the EPoC Message Blocks
VALUE: 0x0B

EPCHtp

TYPE: 4-bit integer

This constant represents the PHY Link message type for the EPoC Probe Control Header message block
VALUE: 0x0A

EPFHtp
TYPE: 4-bit integer
This constant represents the PHY Link message type for the EPoC PHY Frame Header message block
VALUE: 0x09

FPMBtp
TYPE: 4-bit integer
This value represents the PHY Link message type for the FEC Parity message block
VALUE: 0x0C

MaxMBlen
TYPE: integer
This constant represents the maximum number of bits in the downstream PHY Link frame minus the length of the one CRC 32 and the FEC Pointer message block and excluding FEC Parity.
VALUE: 2824

102.2.6.2 Counters

EMBcnt
TYPE: 16-bit integer
This variable counts the number of EPoC message blocks received. The variable is cleared when read and does not roll over at maximum count.

EMBerr
TYPE: 16-bit integer
This variable counts the number of EPoC message blocks received with CRC32 errors. The variable is cleared when read and does not roll over at maximum count.

EPCHcnt
TYPE: 16-bit integer
This variable counts the number of EPoC Probe Control Headers received. The variable is cleared when read and does not roll over at maximum count.

EPCHerr
TYPE: 16-bit integer
This variable counts the number of EPoC Probe Control Headers received with CRC32 errors. The variable is cleared when read and does not roll over at maximum count.

EPFHcnt
TYPE: 16-bit integer
This variable counts the number of EPoC PHY Frame Headers received. The variable is cleared when read and does not roll over at maximum count.

EPFHerr
TYPE: 16-bit integer
This variable counts the number of EPoC PHY Frame Headers received with CRC32 errors. The variable is cleared when read and does not roll over at maximum count.

FPMEcnt
TYPE: 16-bit integer
This variable counts the number of FEC Parity message blocks received. The variable is cleared when read and does not roll over at maximum count.

FPMEerr

TYPE: 16-bit integer

This variable counts the number of FEC Parity message blocks received with CRC32 errors. The variable is cleared when read and does not roll over at maximum count.

LocalTS

TYPE: 32 bit unsigned

This counter holds the value of the local Timestamp. The counter is advanced by the OFDM clock (1/204.8) and rolls over to zero from 0xFFFFFFFF. At the CLT the counter shall track the transmit clock, while at the CNU the counter shall track the receive clock. For accuracy of receive clock, see {ref}. Changing the value of this variable while running using Layer Management is highly undesirable and is unspecified.

102.2.6.3 Variables

BEGIN

TYPE: boolean

This variable is used when initiating operation of the functional block state diagram. It is set to TRUE following initialization and every reset.

DS_CID

TYPE: 2 bit unsigned integer

This variable represents the downstream Configuration ID value as described in 102.2.3.1.

FCP

TYPE: integer

This variable represents the beginning of the first MAC data FEC codeword in the current downstream PHY Link frame as described in 102.2.3.5.

FmLen

TYPE: integer

This variable represents the total number of bits transmitted in the current PHY Link frame.

PC_Fifo

TYPE: bit array

This variable holds the Probe Scheduling, Broadcast PHY Discovery or Unicast PHY Discovery variables to be transmitted in the next EPCH message block.

PhyDA

TYPE: 15 bit unsigned integer

This variable represents the CNU_ID of the intended recipient of the EPoC frame.

PhyDA_Fifo

TYPE: bit array

This variable holds the CNU_IDs to which the PHY Link frame and each PHY Link Instruction is to be sent. For any single PHY Link Frame there is one entry for the frame and one entry for each instruction.

PhyTD

TYPE:bit array

This variable represents a bit array corresponding to data to be sent over the PHY Link.This variable is used to accumulate payload of outgoing PHY Link message blocks, for example to set the Timetamp Message Block.

PhyTxFifo

TYPE: bit array

This variable holds a series of PHY Instructions to be transmitted in the next PHY frame. Each entry in the fifo includes Opcode, Count, Variable Group Index and Data fields for each instruction.

RF_ID

TYPE: 8 bit integer
This variable represents the Response Frame ID as described in 102.2.3.1.

RT

TYPE: boolean
This variable represents the Response Type as described in 102.2.3.1.

PhyLnkRspTm

TYPE: 16-bit Integer
The value of this variable defines the minimum time, in OFDM clocks, after receiving the last bit of the FEC, needed by the CNU to decode and prepare the response to a PHY Link Instruction.

PrbCtrl

TYPE: 32 bit binary
These variables represents the eight Probe control fields as described in 102.2.3.2.

StrtOfFm

TYPE: boolean
When this variable transitions from FALSE to TRUE it indicates the beginning of an OFDM frame.

tmpDA

TYPE: 15 bit unsigned integer
This variable represents the CNU_ID of the intended recipient of the EPoC message blocks included in the PHY Link frame.

TxEnable

TYPE: boolean
This variable enables the device to transmit onto the media when TRUE. It is set to FALSE following initialization and every reset.

TxPre

TYPE: boolean
When TRUE this variable indicates the PHY Link should be sending the preamble pattern as defined in 102.2.2.

US_CID

TYPE: 2 bit integer
This variable represents the upstream Configuration ID value as described in 102.3.2.1.

102.2.6.4 Functions

CRC32(x)

This function returns a 32 bit CRC of the bit array x (See 3.2.9).

LEN(x)

This function returns the length of variable x.

PCnxt(x)

This function removes and returns the next x bits from the PC_Fifo bit array.

POP()

This function removes one record from the PhyTxFifo.

PUSH()

This function returns one record from the PhyTxFifo

Send(x)

This function transfers the contents of variable x to the PHY Link FEC Encoder block. When the transfer is complete the variable length is zero.

102.2.6.5 State diagrams

The CLT PHY Link transmit process shall conform to the state diagram shown in Figure 102–16.

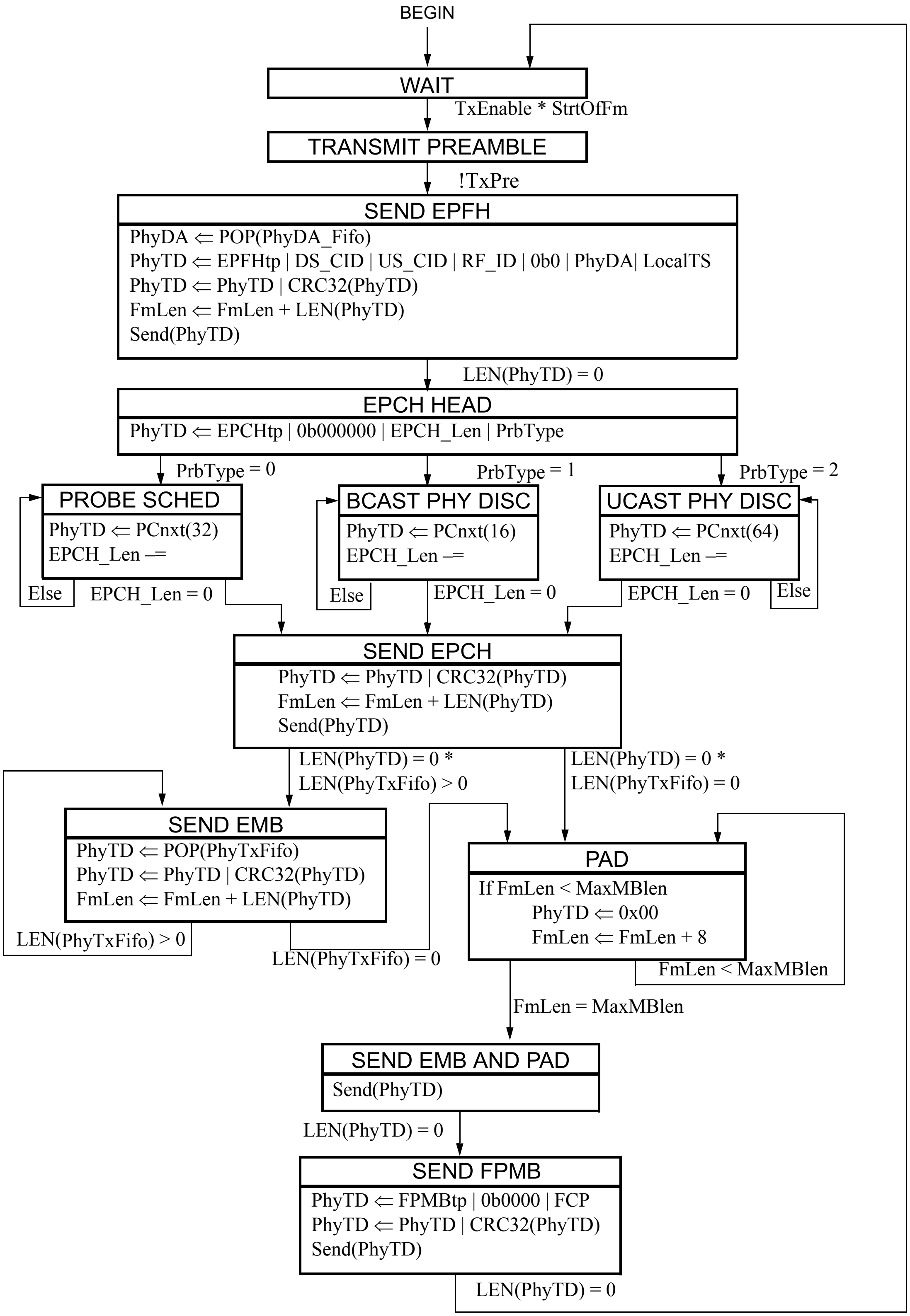


Figure 102–16—CLT PHY Link transmit process state diagram

102.3 Upstream PHY Link

102.3.1 US PHY Link physical layer

102.3.1.1 US Resource Allocation

During network setup the upstream PHY Link shall be allocated 400 kHz of spectrum. The upstream PHY Link is located per the *US_PHyLinkStrt* variable (see US PHY Link Start, 45.2.1.139) that determines the lowest frequency subcarrier of the PHY Link information channel. The upstream PHY Link shall use the same cyclic prefix duration and window size as the upstream MAC data channel.

102.3.1.2 US PHY Link modulation

The upstream PHY Link shall use any of the modulation formats listed under PHY Link CNU Tx/CLT Rx in Table 100–2.

102.3.1.3 US PHY Link Subcarrier Block Interleaving

102.3.2 US Frame

The upstream PHY Link frame is composed of the EPoC PHY Frame Header, optional EPoC message blocks and a FEC. These messages are described below.

Each message block contains a Type field used to identify the contents of the block. CNUs shall use the appropriate message Type fields listed in Table 102–6 in each message block.

The contents of the each message block is protected by a CRC(32). See 3.2.9 for a description of how this field is calculated. The CLT shall calculate a CRC(32) on the data fields within each message block received and, if the calculated CRC(32) does not match the received CRC(32) discard the message and take no action based on it. The CLT PHY Link receiver maintains count of EPoC PHY Frame Headers received and those received that have CRC-32 error using the variables *EPFHcnt* and *EPFHerr*, respectively.

102.3.2.1 US EPoC PHY Frame Header

The upstream PHY Frame Header includes a Type field, the Return Frame ID field, the PHY SA and a CRC(32) as illustrated in Figure 102–2. The Type field and Return Frame ID field are echoes of the same fields in the downstream message to which the CNU is responding. The PHY SA is an address field that identifies the CNU the PHY frame is transmitted from. This field is 15 bits and is always the unicast address (CNU_ID) associated with the CNU transmitting in the PHY Link (see Table 102–8). The CNU SA is assigned during the PHY Discovery process (see 102.4.1).

102.3.2.2 US EPoC message block

The EPoC message block contains a Type field (see Table 102–6), the PHY Response and a CRC(32).

If the downstream PHY Link EPoC PHY Frame Header contains the unicast CNU_ID for the CNU, the addressed CNU shall respond to PHY Link instructions using the PHY Response. Each Response contains an OPCODE, a Variable Group Count, an Variable Group Index and up to 31 Variable Group Data fields.There is a one to one correspondence between PHY Instructions in the downstream PHY Link and PHY Responses in the upstream PHY Link.

The PHY Response OPCODE is a 3 bit value that conveys the acknowledge type for the PHY Instruction to which the CNU is responding and the success or failure of the PHY Instruction command. CNUs shall use the valid values of the acknowledgment type given in Table 102–11.

Table 102–11—OPCODE Acknowledgement values

Value	Operation	Description
b000	NOP ACK	NOP Instruction acknowledge returned in response to a successfully received NOP Instruction
b001	Read ACK	Read Instruction acknowledge returned in response to a successfully received and executed read Instruction along with the requested variables as specified in the Variable Group Index and Variable Group Count fields of the PHY Instruction
b010	Write ACK	Write Instruction acknowledge returned in response to a successfully received and executed write Instruction
b011	Write/Verify ACK	Write/Verify Instruction acknowledge returned in response to a successfully received and executed write/verify Instruction along with the requested variables as specified in the Variable Group Index and Variable Group Count fields of the PHY Instruction
b100	NOP NACK	NOP Instruction negative acknowledge returned in response to a unsuccessfully received NOP Instruction {might just want to keep this as a reserved value}
b101	Read NACK	Read Instruction negative acknowledge returned in response to an unsuccessfully received or executed read Instruction
b110	Write NACK	Write Instruction negative acknowledge returned in response to an unsuccessfully received or executed write Instruction
b111	Write/Verify NACK	Write/Verify Instruction negative acknowledge returned in response to an un successfully received or executed write/verify Instruction

The Variable Group Count field specifies the number of 16-bit variable groups that are being returned in response to a read or write/verify PHY Instruction. In the event the CNU is returning a Nack response (Acknowledgment values 0b100 through 0b0111) the Variable Group Count field shall be set to zero and is ignored at the CLT.

The 16-bit Variable Group Index field specifies the first index for which the CNU is returning data due to a read or write/verify operation or the first index of the corresponding write Instruction.

The 16-bit Variable Group Data fields contain the data values read from the variables due to a read or write/verify PHY Instruction. In the event the CNU is responding to a write instruction this field is omitted. The Variable Group Data field should be omitted in Nack Responses (Acknowledgment values 0b100 to 0b111) and, if included, shall be ignored at the CLT.

The CLT PHY Link receiver maintains count of EPoC message blocks received and those received that have CRC-32 error using the variables *EMBcnt* and *EMBerr, respectively.*

102.3.2.2.1 Padding

The Pad field is used to fill the PHY frame in the event there are unused bits after the message blocks and FEC fields have been accounted for.

102.3.3 US PHY Link FEC

The upstream PHY Link shall use a (384,288) binary punctured LDPC code described in 102.1.4.2.1.

102.3.4 US PHY Link pilot pattern

The upstream PHY Link utilizes a pilot pattern to assist the CLT receiver in capturing the bursting PHY Link transmissions. The PHY Link pilot pattern is illustrated in Figure 102–17. PHY Link pilots are BPSK encoded.

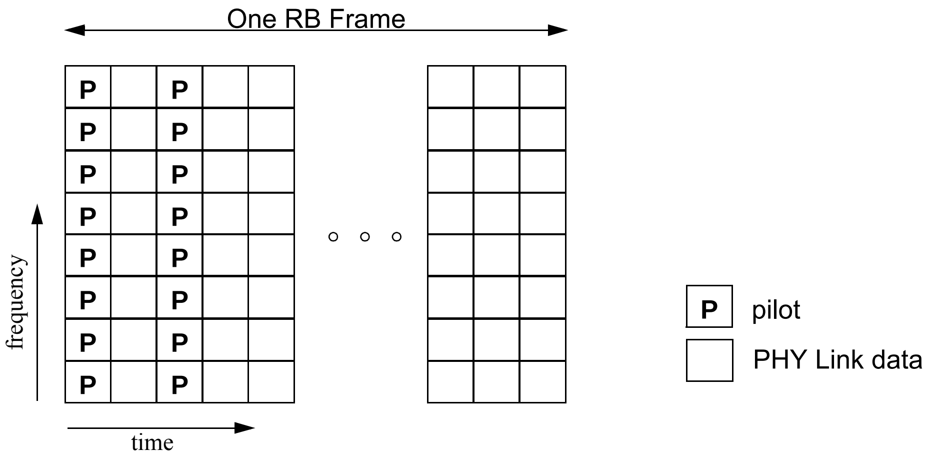


Figure 102–17—Upstream PHY Link pilot pattern

102.3.5 US state diagrams

102.3.5.1 Constants

EPFHtp

See102.2.6.1.

USPLfec

TYPE: Integer
This constant represents the US PHY Link FEC data length.
Value: 288

102.3.5.2 Counters

EMBcnt

See 102.2.6.2.

EMBerr

See 102.2.6.2.

EPFHcnt

See 102.2.6.2.

EPFHerr

See 102.2.6.2.

US_FmCnt

TYPE: 9-bit unsigned

This modulo 262 counter tracks the OFDMA symbols within the upstream superframe. Symbol zero is the first symbol in the Probe Period.

102.3.5.3 Variables

CNU_ID

TYPE: 15-bit integer that carries the value of the CNU_ID assigned by the CLT to the CNU during PHY Discovery process (see 102.4.1.6).

PhyDA

See102.2.6.3.

PhyTD

See102.2.6.3.

PhyTxFifo

See102.2.6.3.

RF_ID

See 102.2.6.3.

RT

See 102.2.6.3.

TxEnable

See 102.2.6.3.

PdCmplt

TYPE: boolean

When TRUE this variable indicates the CNU has completed the PHY Discovery process and is allowed to transmit in the OFDMA MAC and PHY Link data paths.

102.3.5.4 Functions

CRC32()

See 102.2.6.4.

LEN()

See102.2.6.4.

Mod(x,y)

This function returns the remainder of x/y.

POP()

See102.2.6.4.

Send()

See102.2.6.4.

102.3.5.5 Timers

102.3.5.6 Messages

102.3.5.7 State diagrams

The CNU PHY Link transmit process shall conform to the state diagram shown inFigure 102–18.

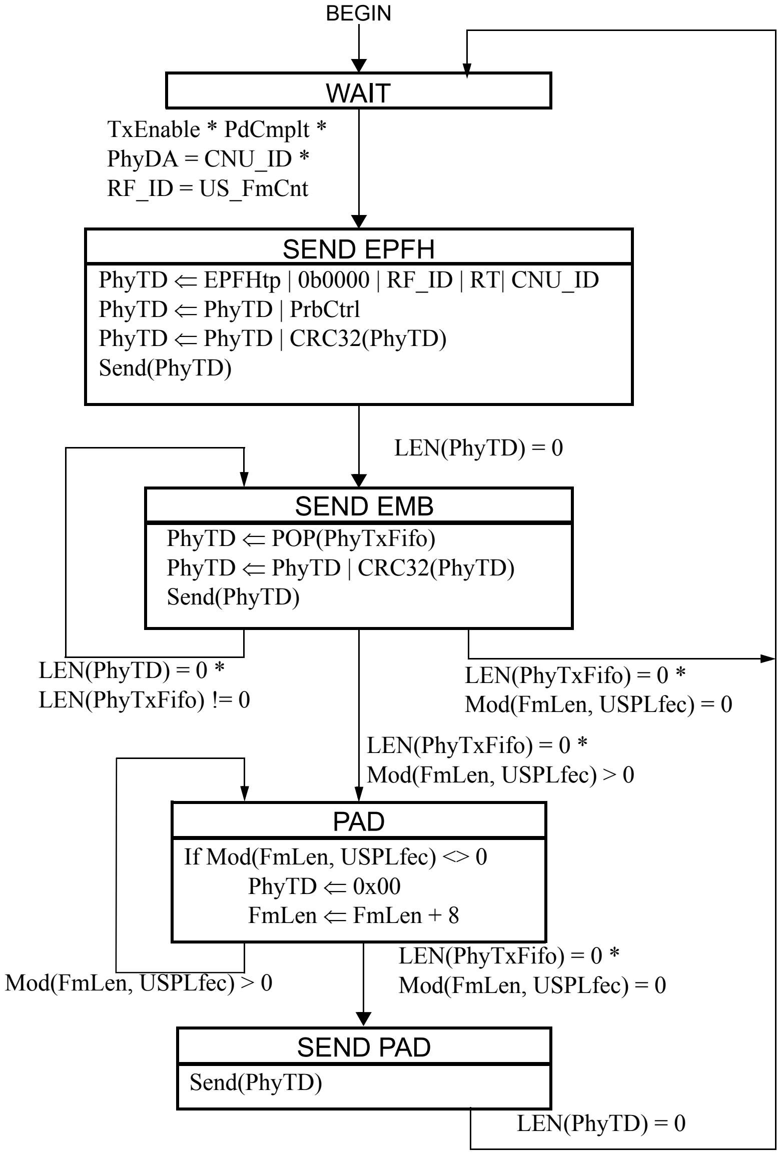


Figure 102–18—CNU PHY Link transmission control state diagram

102.4 PHY Discovery and Wideband Probing

When a CNU first joins the EPoC network it has no prior knowledge of the upstream RB Superframe configuration and timing necessary to produce transmissions that are orthogonal to the rest of the EPoC network when viewed from the perspective of the CLT receiver. PHY Discovery is the process whereby newly connected or off-line CNUs are provided timing and operating parameters necessary to function properly in the EPoC network.

While an EPoC network is in operation, periodic verification of the CNUs OFDMA timing is needed to ensure orthogonality. This is accomplished using Wideband Probing. Wideband Probing is also used during the PHY Discovery process to fine tune the timing of CNUs joining the network.

102.4.1 PHY Discovery

102.4.1.1 Overview of PHY Discovery

The PHY Discovery process is composed of; PHY Link acquisition, PHY Discovery window opening, PHY Discovery Response, CNU_ID Allocation, and Wideband Probing. Each of these steps is described in detail in the following sections. The PHY Discovery message exchange is illustrated in Figure 102–19.

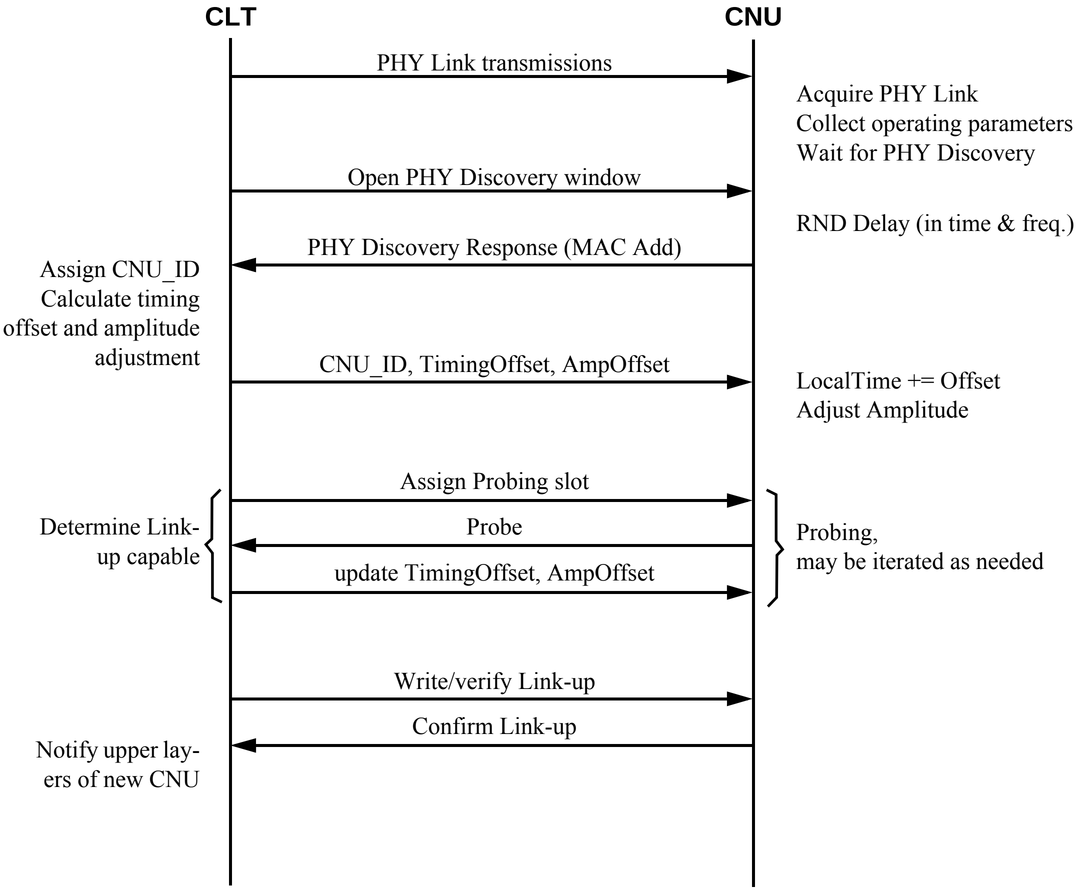

CLT
CNU
PHY Link transmissions
Acquire PHY Link
Collect operating parameters
Wait for PHY Discovery
Open PHY Discovery window
RND Delay (in time & freq.)
PHY Discovery Response (MAC Add)
Assign CNU_ID
Calculate timing
offset and amplitude
adjustment
CNU_ID, TimingOffset, AmpOffset
LocalTime += Offset
Adjust Amplitude
Assign Probing slot
Determine Link-up capable
Probe
update TimingOffset, AmpOffset
Probing,
may be iterated as needed
Write/verify Link-up
Confirm Link-up
Notify upper layers of new CNU

Figure 102–19—PHY Discovery message exchange

102.4.1.2 PHY Link acquisition

When a CNU joins an EPoC network it must first locate the downstream PHY Link. This is typically done via a vendor specific correlation algorithm. Prior to any transmission in an EPoC network the CNU shall locate the downstream PHY Link, gather the required PHY Discovery operating parameters listed in Table 102–13 and synchronize it's local clock to the downstream OFDM clock. Once the CNU has completed the prerequisites for transmission it waits for the CLT to issue a PHY Discovery window.

EDITORS NOTE (to be removed prior to publication): the TF needs to determine precisely how accurately the CNU must be synchronized to the OFDM clock before US transmission is allowed.

102.4.1.3 PHY Discovery window opening

The CLT periodically makes available PHY Discovery windows during which off-line CNUs are given the opportunity to make themselves known to the CLT. The periodicity of these windows is unspecified. The CLT signifies that a PHY Discovery period is occurring by transmitting either a Broadcast Phy Discovery type or Unicast Phy Discovery type Probe Control message block.

The PHY Discovery window is coincident with the Probe symbols (see 101.4.2.10) but is only four symbols in duration to allow for timing ambiguity in the PHY Discovery Response.

102.4.1.4 PHY Link Discovery Response

Off-line CNUs, upon being notified of a PHY Discovery window, wait for the beginning of the PHY Discovery window and then transmits a PHY Discovery Response to the CLT. PHY Discovery windows are unique in that they are the only times when multiple CNUs can access the coax cable distribution network using the same RF spectrum simultaneously, and transmission overlap can occur in both time and frequency. In order to reduce transmission overlaps, a contention algorithm is used by all off-line CNUs. Measures are taken to reduce the probability for overlaps by artificially introducing a random distribution in the PHY Discovery window used by each CNU. Each CNU selects a random number of PHY Discovery windows it waits before transmitting the PHY Discovery Response. Multiple valid PHY Discovery Responses that overlap in time may be received by the CLT during a single PHY Discovery window depending on the modulated spectrum of OFDM channel 0.

Included in the PHY Discovery Response is a preamble (see 102.4.1.5), the CNU's MAC address and a 32-bit CRC. See 3.2.9 for a description of how the 32-bit CRC is calculated. The data in the PHY Discovery Response shall be encoded using a (128,80) binary punctured LDPC code described in 102.1.4.2.2. Bit mapping in the PHY Discovery Response is as shown for the PHY Link in Figure 102–5.

The PHY Discovery Response is comprised of 128 subcarriers with a duration of four symbols and may include exclusions between these 128 subcarriers (see Figure 102–20). have . In a single Probe Period there may be up to 16 PHY Discovery windows. For the purposes of PHY Discovery Response contention mitigation the broadcast PHY Discovery windows are numbered consecutively from lower starting frequency to higher starting frequency and increasing as time progresses as illustrated in Figure 102–21. Unicast PHY Discovery windows are not numbered for contention mitigation purposes.

In the event there is an analog fiber segment between the CLT and CNU the CLT can delay the PHY Discovery Response by the amount of time specified in PhyRngOffset.

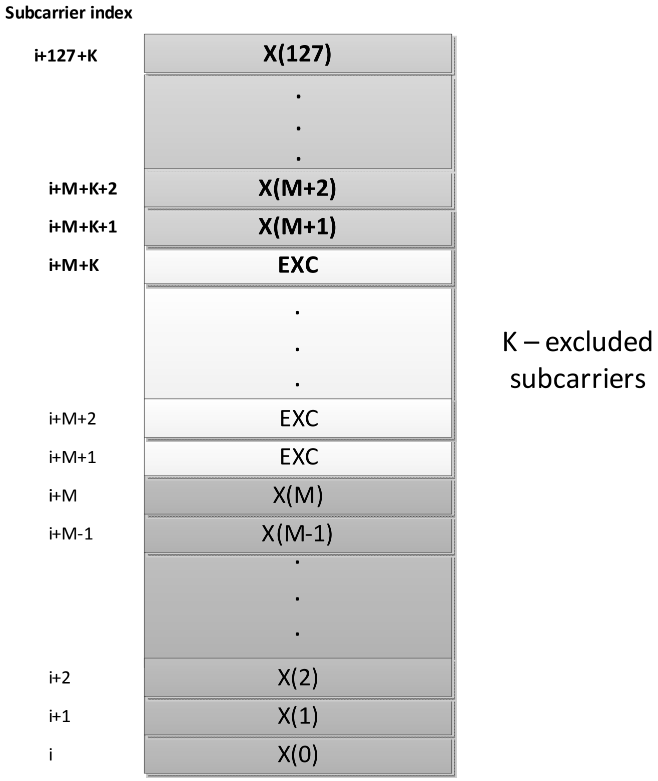


Figure 102–20—Figure 101-PHY Discovery Response sequence with exclusions

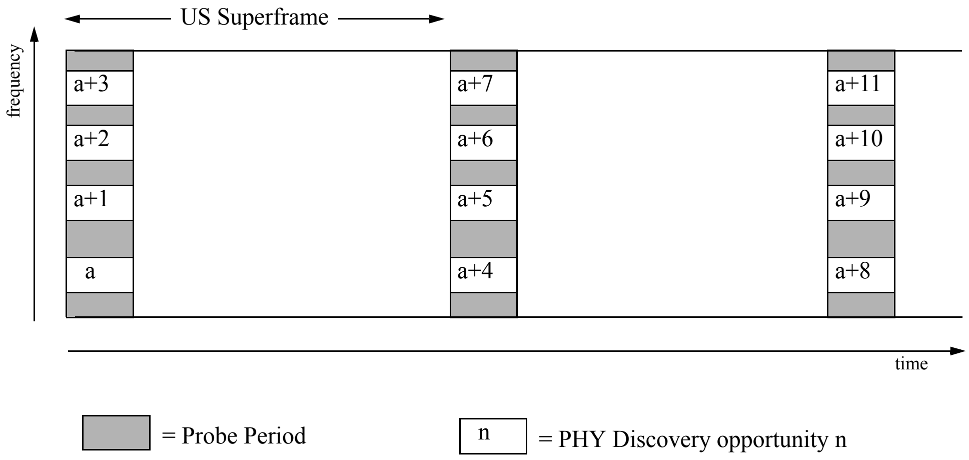


Figure 102–21—PHY Discovery opportunity numbering

The CNU PHY Discovery Response is only allowed after a CNU has completed the PHY Discovery prerequisites (102–13). In the PHY Discovery Response message the preamble used is the special PHY Discovery Preamble (see 102.4.1.5) and the only data included is the CNU MAC address protected by a CRC(32). The PHY duplicates symbols of the upstream PHY Discovery Response transmission. This duplication is accomplished by duplicating the time domain samples at the output of the iFFT in the upstream data path for these signals, and adding cyclic prefix and windowing (per variables *USNcp* and *USNrp*, respectively) as illustrated in Figure 102–22. Control for the duplication process is conveyed using the TxType in the CNU (see Figure 102–4).

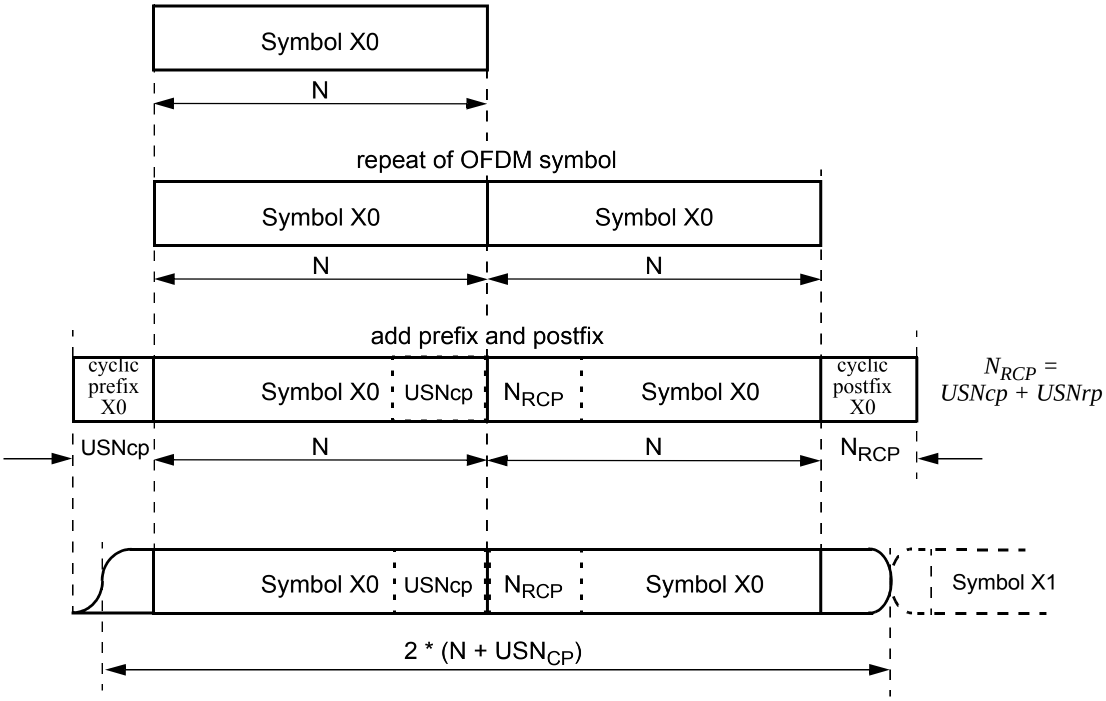


Figure 102–22—Symbol duplication, cyclic prefix, and windowing algorithm for PHY Discovery Response

102.4.1.5 PHY Discovery preamble

The PHY Discovery preamble is transmitted in the first two symbols of the PHY Discovery Response. The first symbol of the preamble shall be populated with a BPSK mapped 128 bit sequence generated by a pseudo-random sequence generator defined by the polynomial $X^7 + X^1 + 1$ seeded with a fixed bit pattern of 0x55 (see Figure 102–23) at the beginning of the PHY Discovery Response (see Figure 102–24). The output of the sequence generator is mapped using BPSK modulation (see 101.4.4.2) where a bit value of 0 is mapped to a BPSK value of plus 1 and a bit value of 1 is mapped to a BPSK value of minus 1. The second symbol of the PHY Discovery preamble shall be a duplicate copy of the first symbol as illustrated in Figure 102–22.

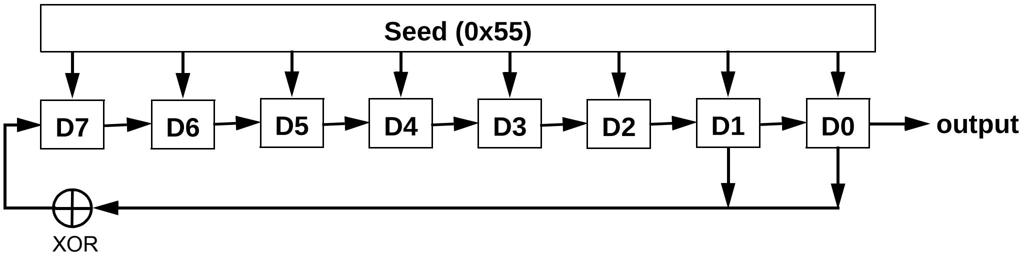


Figure 102–23—PHY Discovery Preamble generator

102.4.1.6 CNU_ID allocation

Upon receipt of a valid PHY Discovery Response, the CLT updates the CNU by allocating and assigning a new port identity (CNU_ID) per the AllwdCNU_ID variable. To allocate the CNU_ID the CLT shall use the CNU_ID Allocation message as defined in Table 102–12. The CLT calculates an OFDMA timing offset and adjusts the CNU's transmit power using the *PHYTimingOffset* and *PHYPowerOffset* variables, respectively. These parameters are transmitted to the CNU via the CNU_ID Allocation instruction. This instruction uses a broadcast DA with the first EMB being a read of the CNU MAC Address variable. The value written is the CNU MAC address received in the PHY Discovery Response followed by the AllwdCNU_ID variable. Subsequent EMBs are used to write the CNU *PHYTimingOffset*, and *PHYPowerOffset* variables.

Table 102–12—Special Instruction sequences

Special Instruction	EMB order	Read/Write	Variable
CNU_ID Allocation	1	Write	*New_MAC0* through *New_MAC2*
	2	Write	AllwdCNU_ID
	3	Write	*PHYTimingOffset* *PHYPowerOffset*

When the CNU receives the *PhyTimingOffset* variable it shall add the new value of *PhyTimingOffset* to the RangingOffset.

102.4.1.7 PHY Discovery State Diagrams

102.4.1.7.1 Constants

Pad96

TYPE: 96 bit binary
This constant holds 96 bits of padding.
Value: 0

102.4.1.7.2 Variables

AllwdCNU_ID

TYPE: 15-bit integer

This variable is used to indicate to the 10GPASS-XR PHY a valid CNU_ID value. The value may be assigned to a new CNU when the associated CNU_ID assigned flag is set to zero, when the flag is set to one it is an indication that this value has already been assigned to a CNU and it should not be use for another CNU.

DS_OFDM_ID

TYPE: 3-bit integer

This variable is a pointer to one of the five possible OFDM channels in the downstream EPoC network. Thus when *DS_OFDM_ID* is set to a value of one variables DS_ModTypeSC(n) reflect the OFDM descriptor for OFDM channel one. When *DS_OFDM_ID* is set to a value of two variables DS_ModTypeSC(n) reflect the OFDM descriptor for OFDM channel two, etc.

New_MAC

TYPE: 48-bit hex

This vairable holds the MAC address of the CNU.

NxtWin

TYPE: integer

This variable holds the subcarrier index for the next PHY Discovery window.

PdRndDly

TYPE: integer

This variable indicates the random delay, in PHY Discovery windows, selected by the PHY to avoid contention during PHY Discovery.

PdData

TYPE: bit array

This variable holds the data to be transmitted in the PHY Discovery window.

PdWinFifo

TYPE: bit array

This fifo holds the list of 12-bit subcarrier indexes for each PHY Discover window from the most recent EPoC Probe Control Header message block.

PdWinTp

TYPE: boolean

This variable indicates if the list PHY Discover windows in PdWinFifo are unicast or broadcast windows. PdWinTp has the value of Ucast when the PHY Discovery window is unicast and Bcast when the window is broadcast.

PdCmplt

See102.3.5.3.

PhyRngOffset

TYPE: 32-bit integer

This variable may be used to provision a delay in the ranging response, in units of 1/204.8 MHz, in the event there is an analog optical segment between the CLT and the CNUs as described in 102.4.1.4. This variable defaults to a value of 0 on reset.

Rnd

TYPE: 8-bit integer

This variable is used as a seed in the back-off algorithm for the PHY Discovery Response.

SoSF

TYPE: boolean

This variable indicate the beginning of the upstream superframe and is set to TRUE for one OFDM clock period at the beginning of the superframe.

TxEnable

See 102.2.6.3.

102.4.1.7.3 Counters

SC_Cnt

TYPE: 12-bit

This counter tracks the subcarriers within each symbol and increments once for every subcarrier. Subcarrier zero is the first subcarrier while subcarrier 4095 is the last. The counter is reset at the beginning of every symbol.

102.4.1.7.4 Functions

CRC32()

See 102.2.6.4.

Len(x)

See 102.2.6.4

PD_Pre

This function returns the PHY Discovery preamble as described in 102.4.1.5.

POP(x)

See 102.2.6.4

rnd(r)

This function returns a random integer number uniformly distributed between 1 and r.

102.4.1.7.7 State Diagrams

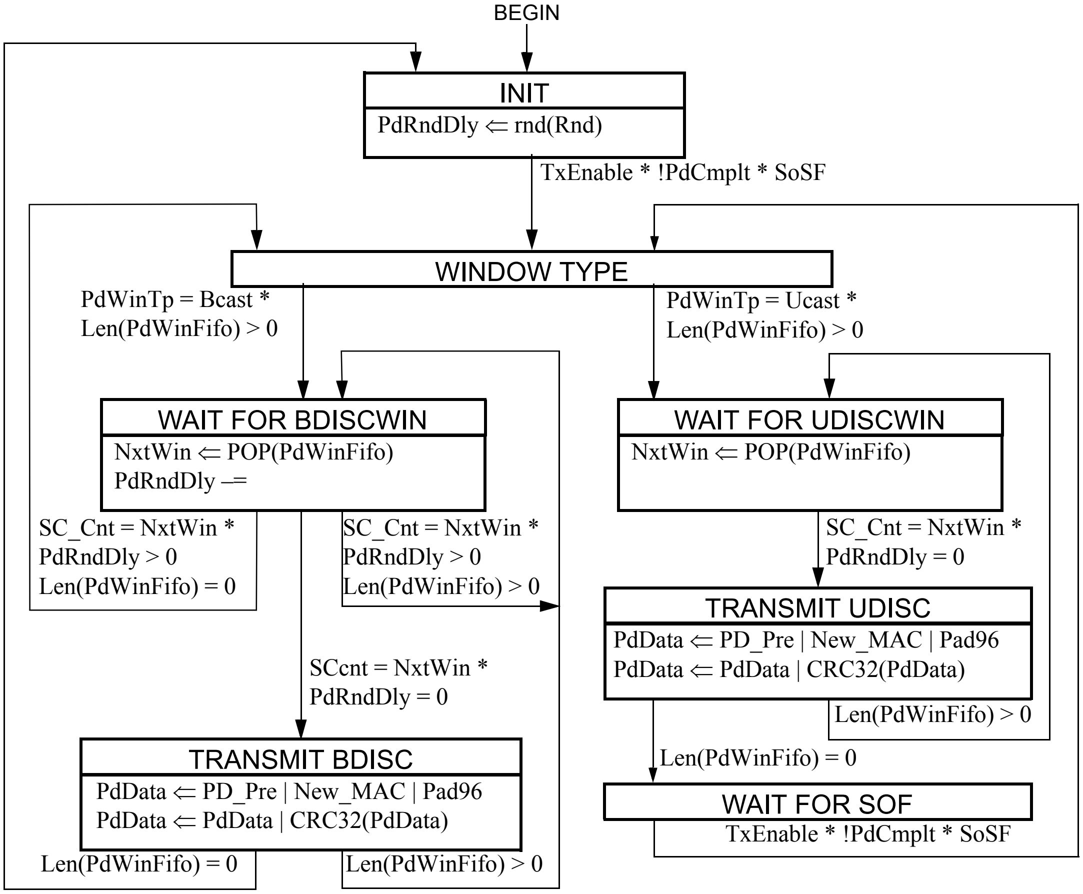


Figure 102–24—CNU PHY Discovery response transmission control state diagram

102.4.2 Upstream Wideband Probing

102.4.2.1 Introduction

In upstream Wideband Probing a CNU transmits pilots spanning all active subcarriers. The CNU transmits one pilot per subcarrier . Each pilot is a predefined BPSK symbol. Each upstream superframe begins with six symbols, called the Probe Period, designated for probing and PHY Discovery. Each symbol within the Probe Period is refered to as a probing symbol. The CLT may use the received probing symbol for:

1) Upstream OFDM channel estimation. The CLT computes the coefficients of the upstream pre-equalizer for each CNU and sends them back to that CNU.
2) Upstream SNR measurement. The CLT measures the SNR per subcarrier and computes the upstream bit loading tables.
3) Upstream timing adjustment. During CNU bring up the CLT may use Wideband Probing to adjust the timing of the new CNU to the upstream RB Frame and superframe.

102.4.2.2 Probing symbol pilots

Probing symbol pilots are BPSK symbols.

Probing symbol pilot i is always associated with the i-th subcarrier of the symbol, where:

i = 0, 1, ... , 4095

(Subcarriers are numbered in ascending order of frequency starting from 0.)

102.4.2.3 Probing symbol scheduling

The CLT may allocate a one or more probing symbols to a CNU within the Probe Period and instruct the CNU to transmit the probing sequence in that symbol. CLT specifies the probing symbol within the Probe Period via the EPoC Probe Control Header message block in the downstream PHY Link frame. The CLT assigns a CNU either all the pilots of the assigned probing symbol, or a subset of (scattered) pilots of the assigned probing symbol.

The CLT allocates subcarriers within a probing symbol by sending five variables to the CNU: *PrbStrtSC*, *PrbSkp*, *PrbEQ*, *StrtSym* and *SymNum* (see 101.4.3.5.1). Figure 102–25 and Figure 102–26 illustrate the use of *PrbStrtSC* and *PrbSkp*. The CNU uses the *PrbStrtSC* and *PrbSkp* variables to determine which subcarriers are to be used for probing transmission, as follows:

- The *PrbStrtSC* variable is the starting subcarrier number.
- The *PrbSkp* variable is the number of subcarriers to be skipped between successive pilots. *PrbSkp* = 0 implies no skipping of subcarriers (i.e., all subcarriers are used for probing).

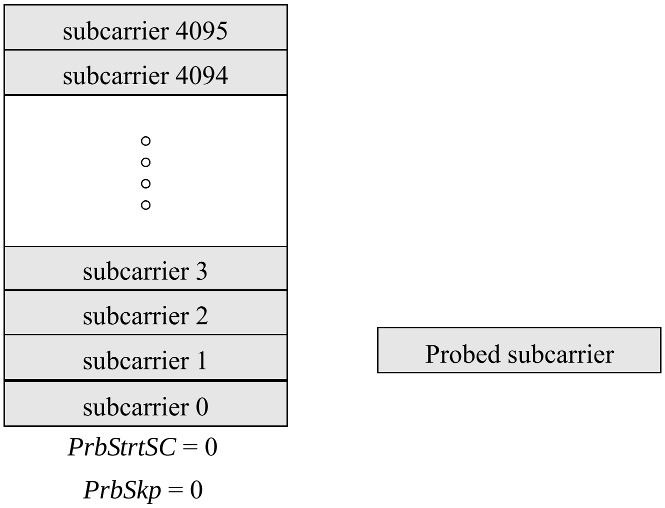


Figure 102–25—Example, all subcarriers used for probing, no skipping

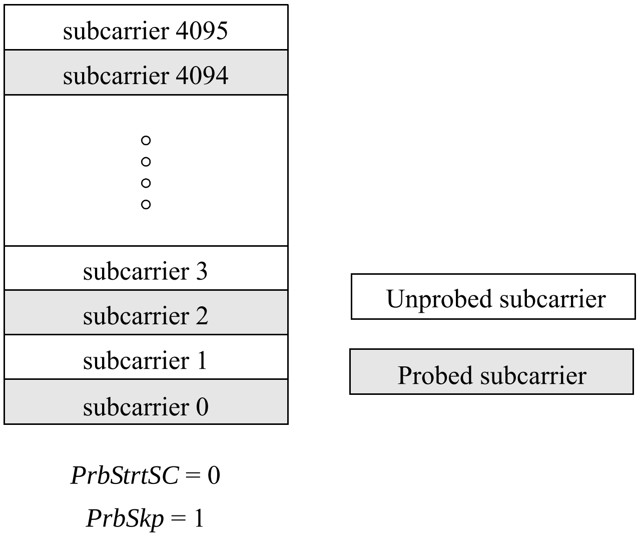


Figure 102–26—Example, alternate subcarriers used for probing

To schedule a single CNU in a probing symbol without skipping subcarriers, the CLT does the following:

1) Allocate a specific probing symbol to a single CNU using *StrtSym* and *SymNum*.
2) Set *PrbSkp* to zero.
3) Set *PrbStrtSC* to the number of the first subcarrier.

To schedule a single CNU in a probing symbol with skipping subcarriers to create nulls (as illustrated in Figure 102–26), the CLT does the following:

1) Allocate a specific probing symbol to a single CNU using *StrtSym* and *SymNum*.
2) Set *PrbSkp* to a non-zero positive integer value.
3) Set *PrbStrtSC* to the number of the first subcarrier.

To schedule multiple CNUs in a probing symbol (as illustrated in Figure 102–27) the CLT does the following:

1) Allocate the same probing symbol at any given time to more than one CNU using *StrtSym* and *SymNum*.
2) Assign a different *PrbStrtSC* number to each CNU.
3) Assign the same *PrbSkp* value to every CNU within the probing symbol.

This method can be used with or without skipping subcarriers to create nulls. To create nulls, specify a *PrbSkp* value equal to, or greater than, the number of CNUs in the pattern.

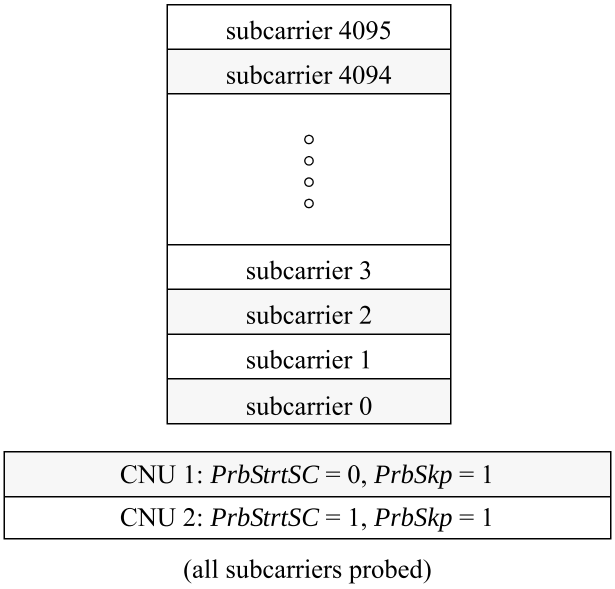


Figure 102–27—Scheduling two CNUs in the same probing symbol

The *PrbEQ* variable determines if the pilots transmitted are to be equalized (*PrbEQ* TRUE) or un-equalized (*PrbEQ* FALSE).

StrtSym and *SymNum* variables determine which symbols within the Probe Period are to be used by the CNU. The value of *StrtSym* determines the first symbol within the Probe Period to begin probe transmissions and *SymNum* determines the number of adjacent symbols in which to send probe pilots. The CLT is responsible for properly setting the Probe Control fields so that the probe pilots are defined within the Probe Period. The CNU shall not transmit probe pilots in response to settings which define transmission outside the configured Probe Period. For example if *StrtSym* equal 4 and *SymNum* equals 3 the CNU does not transmit any probe pilots. The CNU is not responsible for crosschecking Probe Control settings to ensure that no two CNUs are assigned the same subcarriers for probing. Should the CLT make such an assignment it will receive a garbled response.

102.4.2.4 Probing sequence

The Probe symbol shall be populated with a BPSK mapped bit sequence generated by a pseudo-random sequence generator defined by the polynomial $X^{12} + X^9 + X^8 + X^5 + 1$ seeded with a fixed bit pattern of 0xBFF at the beginning of each upstream superframe (illustrated in Figure 102–28). The output of the sequence generator is mapped using BPSK modulation (see 101.4.4.2) where a bit value of 0 is mapped to a BPSK value of plus 1 and a bit value of 1 is mapped to a BPSK value of minus 1.

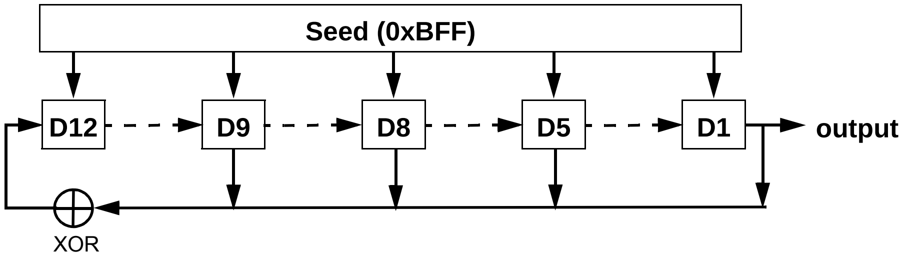


Figure 102–28—Probe sequence generator

102.4.2.5 Probe symbol repetition

Probes use all active subcarriers of a symbol. The first symbol in a Probe is generated as described in 102.4.2.4. The second symbol is a direct copy of the first symbol.

102.4.2.6 Wide Band Probing State Diagrams

102.4.2.6.1 Constants

Exclude

This constant informs the PMA that the subcarrier it is associated with is not to be used for transmission.

102.4.2.6.2 Variables

ActPrbID

TYPE: set of 15-bit Integers
When the CNU_ID of the CNU is contained in this set of variables the CNU is allowed to transmit a Probe Signal per the Probe Control variable *PrbEQ*, *PrbSkp*, *PrbStrtSC*, *StrtSym* and *SymNum*. This set of variables is updated with each downstream PHY Link frame when *RF_ID* = *US_FmCnt* (See Figure 102–30). Each element in this set is one of the *PrbIDs* received in the most recent EPoC Probe Control Header.

CNU_ID

See 102.3.5.3

PhyDisc_N (N = 1, 2, ... 16)

TYPE: 12-bit Integers
Each of these variables designates the starting subcarrier of a PHY Discovery window within the Probe Period.

PrbData

TYPE: bit array
This variable holds the probe sequence bits to be transmitted during the Probe Period.

PrbEQ

See 101.4.3.5.1.

PrbSkp

See 101.4.3.5.1.

PrbStrtSC
See 101.4.3.5.1.

RcvPrbID
See 101.4.3.5.1.

SC_clk
TYPE: boolean
This clear on read variable goes TRUE at the beginning of each subcarrier.

StrtSym
TYPE: integer
The value of this variable determines the starting symbol within the Probe Period in which to begin transmitting probe pilots. The range of this variable is from one to six. The sum of *StrtSym* and *SymNum* is less than or equal to six.

SymNum
TYPE: integer
The value of this variable determines the number of consecutive symbols in which to transmit probe pilots. The range of this variable is from one to six. The sum of *StrtSym* and *SymNum* is less than or equal to six.

US_ModTypeSC(n)
See 101.4.3.4.4 for the definition of these variables.

102.4.2.7 Counters

SC_Cnt
See 102.4.1.7.3.

US_FmCnt
See 102.3.5.2.

102.4.2.8 Functions

Mod(x,y)
This function returns the remainder of x divided by y.

PrbSeq
This function returns one bit worth of probe data using the sequence generator described in 102.4.2.4. The function is reset at the beginning of the Probe Period.

SndPrbData(x)
This function transfers one symbols worth of Probe data (PrbData) from the Probe Symbol Mapper function to the PMA. If x = TRUE then the probe data is to be sent with equalization, if x = FALSE then probe data is sent without equalization.

102.4.2.9 State Diagrams

The CNU probe transmit process shall conform to the state diagram shown in Figure 102–29.

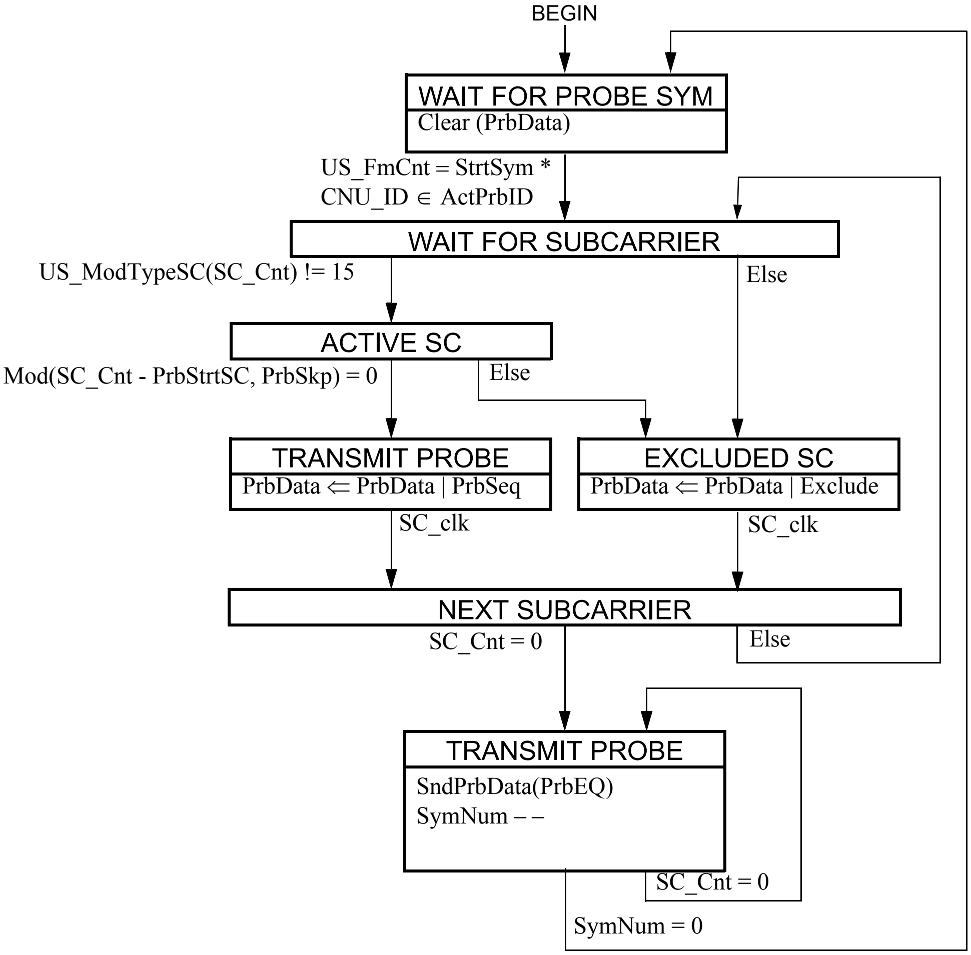


Figure 102–29—CNU Probe transmission control state diagram

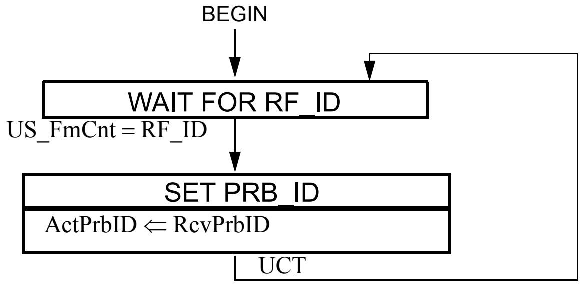


Figure 102–30—PrbID update state diagram

EDITORS NOTE (to be removed prior to publication); we need to define a minimum time of 2.5 ms between the EPCH message and the beginning of the Probe Period.

102.4.3 Link-up declaration

Before declaring a CNU is in the link-up state the CLT shall ensure that a CNU joining the EPoC network is properly aligned to the upstream OFDMA timing and is cognizant of all necessary provisioning parameters needed to properly operate in the OFDMA network without adverse impact to the EPoC network or other services operating in RF spectrum unused by the EPoC network. A list of required parameters is given in Table 102–13.

Once the CLT has determined that a new CNU can participate in the EPoC network and verified the variables listed in Table 102–13 it may declare the new CNU as link-up status by setting the PhyDiscComplete variable in the CNU to TRUE. Once the CLT has verified the CNU is in the link-up status by reading the *TxEnable* variable as TRUE it may set the *AssgndCNU_ID* to TRUE to notify the upper layers that the associated CNU_ID has been assigned and a device has been added to the network (see 45.2.1.141).

EDITOR NOTE (to be removed prior to publication): Is a better definition of what is required for Link-up needed?

There may exist situations when the CLT requires that a CNU go through the PHY Discovery sequence again. Similarly, there may be situations where a CNU needs to inform the CLT of its desire to leave the EPoC network. The CNU can then go through the PHY Discovery sequence again.

Table 102–13—Required variables for PHY Discovery Response and Link-Up

Variable/Variable Group	**Index**	**bit(s)**	**Required for [a]**	
			PHY Discovery	**Link-Up**
TxEnable	0	0	T	T
PhyDiscCmplt	0	1		T
DS_ChCnt	1	14:12		Y
DS_TmIntrlv	1	11:7		Y
DSNrp	1	6:4		Y
DSNcp	1	3:0		Y
DS_FreqCh1 through *DS_FreqCh5* [a]	2 - 5	15:0		Y
Rnd	7	15:8	Y	Y
RBsize	7	7	Y	Y
USNrp	7	6:4	Y	Y
USNcp	7	3:0	Y	Y
US_FreqCh1	8	15:0	Y	Y
Type2_Repeat	9	14:12		

Table 102–13—Required variables for PHY Discovery Response and Link-Up *(continued)*

Variable/Variable Group	Index	bit(s)	Required for [a]	
			PHY Discovery	Link-Up
Type2_Start	9	11:8		
Type1_Repeat	9	6:4		
Type1_Start	9	3:0		
DS_PhyLinkStrt	11	11:0	Y	
US_PhyLinkStrt	12	11:0	Y	
AllwdCNU_ID [b]	15	14:0		Y
PhyTimingOffset	22, 23	15:0		Y
PhyPowerOffset	24	15:0		Y
PhyRngOffset	25, 26	15:0	Y	Y
DS_DataRate	27, 28, 29	15:0	Y	Y
US_DataRate	30, 31, 32	15:0	Y	Y
DS_OFDM_ID [c]	100	2:0		Y
DS_ModTypeSC(4) through *DS_ModTypeSC(4095)* [c]	101 - 1123	15:12, 11:8, 7:4, & 3:0		Y
US_ModTypeSC(0) through *US_ModTypeSC(4095)*	1124 - 2147	15:12, 11:8, 7:4, & 3:0	Y	Y
EQ_CoefR(0) through EQ_CoefR(4095)	2148 - 103338 (even)	15:0		Y
EQ_CoefI(0) through EQ_CoefI(4095)	2149 - 10339 (odd)	15:0		Y
OFDMA_ClkSync	TBD	TBD	T	T
DS_PHY_LinkSync	TBD	TBD	T	T

[a] T = variable must be TRUE, Y = variable must have been written in the CNU via the PHY Link.
[b] *AllwdCNU_ID* is set using the CNU_ID Allocation special instruction (see 102.4.1.6).
[c] The downstream OFDM descriptor must be written for each OFDM channel that contains active subcarriers.

EDITORS NOTE (to be removed prior to publication): OFDMA_ClkSync and DS_PHY_LinkSync are currently undefined.

In some instances the CNU may fail to achieve link-up status. This may happen for a number of reasons; for example the CNU may be unable to support the downstream or upstream Profile due to network conditions. In these circumstances the CLT may take mitigating action outside the scope of this standard and attempt to bring up the CNU at a later time.

102.4.4 Link-down declaration

There are three ways to declare a CNU in the Link-down state; CNU PHY self declared, CLT declared, CNU Upper layer declared. These are described below.

102.4.4.1 CNU PHY self declared link-down

If any of the conditions listed in Table 102–14 become TRUE the CNU shall declare itself to be link-down and set both PhyDiscComplete and *TxEnable* to FALSE. If subsequent analysis indicates the device is ready to re-join the EPoC network, as described in 102.4.1 it may do so by going through the normal discover process.

Table 102–14—Link-down causal conditions

Condition	Description
OFDMA clock misaligned	|Received Timestamp - LocalTS| > TBD(5?)
DS PHY Link loss of frame	unable to decode the DS PHY Link frame for 3 or more consecutive frames
Change in DS OFDM channel count	Any provisioned change to *DS_ChCnt*
Change in DS time interleaver	Any provisioned change to *DS_TmrIntrlv*
Change of cyclic prefix length	Any provisioned change to *DSNcp* or *USNcp*
Change of OFDM channel frequency	Any provisioned change to *DS_FreqCh1, DS_FreqCh1, DS_FreqCh2, DS_FreqCh3, DS_FreqCh4, DS_FreqCh5 or US_FreqCh1*
Change of Resource Block size	Any provisioned change to RBsize
Change of Pilot structure	Any provisioned change to *Type2_Repeat, Type2_Start, Type1_Repeat* or *Type1_Start*
Change of PHY Link	Any provisioned change to *DS_PhyLinkStrt* or *US_PhyLnkStrt*
Change of PHY ranging offset	Any provisioned change to *PhyRngOffset*

EDITOR NOTE (to be removed prior to publication): are other conditions pertinent?

102.4.4.2 CLT declared link-down

The CLT may declare a CNU to be in a link-down state by setting PhyDiscComplete to FALSE. It may further force the CNU to reassess its' readiness for participation on the network by setting *TxEnable* to FALSE.

102.4.4.3 Upper layer declared link-down

The upper layers may declare a CNU to be in a link-down state by setting PhyDiscComplete to FALSE. It may further force the CNU to reassess its' readiness for participation on the network by setting *TxEnable* to FALSE.

102.5 Protocol implementation conformance statement (PICS) proforma for Clause 102, EPoC PHY Link[3]

102.5.1 Introduction

The supplier of a protocol implementation that is claimed to conform to Clause 102, clause title, shall complete the following protocol implementation conformance statement (PICS) proforma.

A detailed description of the symbols used in the PICS proforma, along with instructions for completing the PICS proforma, can be found in Clause 21.

102.5.2 Identification

102.5.2.1 Implementation identification

Supplier[1]	
Contact point for enquiries about the PICS[1]	
Implementation Name(s) and Version(s)[1,3]	
Other information necessary for full identification—e.g., name(s) and version(s) for machines and/or operating systems; System Name(s)[2]	
NOTE 1—Required for all implementations. NOTE 2—May be completed as appropriate in meeting the requirements for the identification. NOTE 3—The terms Name and Version should be interpreted appropriately to correspond with a supplier’s terminology (e.g., Type, Series, Model).	

102.5.2.2 Protocol summary

Identification of protocol standard	IEEE Std 802.3xx-2012, Clause 102, clause title
Identification of amendments and corrigenda to this PICS proforma that have been completed as part of this PICS	
Have any Exception items been required? No [ ] Yes [ ] (See Clause 21; the answer Yes means that the implementation does not conform to IEEE Std 802.3xx-2012.)	

Date of Statement	

[3] *Copyright release for PICS proformas:* Users of this standard may freely reproduce the PICS proforma in this subclause so that it can be used for its intended purpose and may further publish the completed PICS.

102.5.3 Major capabilities/options

Item	Feature	Subclause	Value/Comment	Status	Support
PL	PHY Link supported	102.1.1	for both US and DS directions	M	Yes [ ] N/A [ ]

102.5.4 PICS proforma tables for EPoC PHY Link

102.5.4.1 PHY Link general specifications

Item	Feature	Subclause	Value/Comment	Status	Support
PG1	Profile activation	102.2.3.1.1	CLT ensures all CNUs have matching profile before switching into.	CLT:M	Yes [ ] No [ ] N/A [ ]
PG2	Profile switching	102.2.3.1.1	Once started completes	CLT:M	Yes [ ] No [ ] N/A [ ]
PG3	Transmission prerequisites	102.4.1.2	Meets the requirements of Table 102-13	CNU: M	Yes [ ] No [ ] N/A [ ]
PG4	Link-up declaration	102.4.3	CLT verifies requirements of Table 102-3 are met.	CLT:M	Yes [ ] No [ ] N/A [ ]
PG5	Link-down declaration	102.4.4.1	Link-down declared by setting PhyDiscComplete and TxEnable to FALSE if any conditions in Table 102-14 are TRUE	CNU: M	Yes [ ] No [ ] N/A [ ]
PG6	PHY Link scrambler	102.1.5	Meets the requirements of Figure 101-19	M	Yes [ ] N/A [ ]

102.5.4.2 PHY Link timing

Item	Feature	Subclause	Value/Comment	Status	Support
PT1	CLT LocalTS sync	102.2.6.2	Tracks transmit clock	CLT:M	Yes [ ] No [ ] N/A [ ]
PT2	CNU LocalTS sync	102.2.6.2	Tracks receive clock	CNU: M	Yes [ ] No [ ] N/A [ ]
PT3	LocalTS updates	102.2.3.1.4	LocalTS set to received timestamp when [TxEnable] is FALSE	CNU: M	Yes [ ] No [ ] N/A [ ]
PT4	US OFDMA timing adjustment	102.4.1.6	Upon receipt PhyTimingOffset added to LocalTS	CNU: M	Yes [ ] No [ ] N/A [ ]
PT5	PHY Link response time	102.2.3.1.2	CLT ensures CNU has sufficient time to responsd to DS PHY Link EMB	CNU: M	Yes [ ] No [ ] N/A [ ]
PT6	DS PHY Link response time	102.2.5	4.8 ms maximum	M	Yes [ ] No [ ]

102.5.4.3 DS Framing

Item	Feature	Subclause	Value/Comment	Status	Support
DF1	DS PHY Link frame length	102.1.1	128 OFDM symbols	M	Yes [ ] N/A [ ]
DF2	DS Preamble	102.2.2	Meets the fixed patern in Table 102-4 modulated with BPSK in the first 8 symbols of the DS Frame.	M	Yes [ ] N/A [ ]
DF3	DS PHY Link Type field	102.2.3	As listed in Table 102-6	M	Yes [ ] N/A [ ]
DF4	DS PHY Link CRC	102.2.3	CNU rejects data if calculated and received CRC don't match	CNU: M	Yes [ ] No [ ] N/A [ ]
DF5	DS PHY Link DA	102.2.3.1.3	As listed in Table 102-8	CLT:M	Yes [ ] No [ ] N/A [ ]
DF7	DS PHY Link OPCODE	102.2.3.3	As listed in Table 102-10	CLT:M	Yes [ ] No [ ] N/A [ ]
DF66	DS Timestamp ref	102.2.3.1.4	first sample of the PHY Link symbol immediately following the Preamble per Figure 102–12.	CLT:M	Yes [ ] No [ ] N/A [ ]

102.5.4.4 DS Encoding and transmission

Item	Feature	Subclause	Value/Comment	Status	Support
DET1	DS PHY Link FEC	102.2.4	(384,288) binary punctured LDPC code described in 102.1.4.1.1 and 102.1.4.2.1	M	Yes [ ] No [ ]
DET2	DS PHY Link bit mapping	102.2.1.3	Meets the requirements of Figure 102-9	M	Yes [ ] No [ ]
DET3	DS PHY Link interleaving	102.2.1.3	Meets the requirements of Figure 102-10	M	Yes [ ] No [ ]
DET4	DS PHY Link transmit process	102.2.6.5	Meets the requirements of Figure 102-16	CLT:M	Yes [ ] No [ ] N/A [ ]

102.5.4.5 DS OFDM

Item	Feature	Subclause	Value/Comment	Status	Support
DO1	DS PHY Link spectrum allocation	102.2.1.1	400 kHz	M	Yes [ ] No [ ]
DO2	DS PHY Link surrounding spectrum	102.2.1.1	24 MHz contiguous, with at lease 3 MHz above and below the PHY Link allocations	M	Yes [ ] No [ ]
DO3	DS PHY Link symbol	102.2.1.1	Same CP duration and windowing size as DS data	M	Yes [ ] No [ ]

102.5.4.6 US Encoding and transmission

Item	Feature	Subclause	Value/Comment	Status	Support
UET1	US PHY Link FEC	102.3.3	(384,288) binary punctured LDPC code described in 102.1.4.1.1 and 102.1.4.2.1	M	Yes [ ] No [ ]
UET2	US PHY Link transmit process	102.3.5.7	Meets the requirements of Figure 102-18	CNU: M	Yes [ ] No [ ] N/A [ ]

102.5.4.7 US Framing

Item	Feature	Subclause	Value/Comment	Status	Support
UF1	US PHY Link frame length	102.1.1	262 OFDM symbols	M	Yes [ ] No [ ]
UF2	US PHY Link Type field	102.3.2	As listed in Table 102-6	CNU: M	Yes [ ] No [ ] N/A [ ]
UF3	US PHY Link CRC	102.3.2	CCL rejects data if calculated and received CRC don't match	CLT:M	Yes [ ] No [ ] N/A [ ]
UF4	US PHY Link addressing	102.3.2.2	CNU responds to unicast downstream PHY Link messages if DA equals CNU_ID	CNU: M	Yes [ ] No [ ] N/A [ ]

102.5.4.8 US OFDM

Item	Feature	Subclause	Value/Comment	Status	Support
UO1	US PHY Link spectrum allocation	102.3.1.1	400 kHz	CLT:M	Yes [ ] No [ ] N/A [ ]
UO2	US PHY Link symbol	102.3.1.1	Same CP duration and windowing size as US data	CNU: M	Yes [ ] No [ ] N/A [ ]
UO3	US PHY Link modulation	102.3.1.2	Meets the requirements of Table 100-2 per column "PHY Link CNU Tx/CLT Rx"	M	Yes [ ] No [ ]

102.5.4.9 Communication Protocol

Item	Feature	Subclause	Value/Comment	Status	Support
CP1	US PHY Link ACK	102.3.2.2	As listed in Table 102-11	M	Yes [ ] No [ ]
CP2	US PHY Link NACK	102.3.2.2	Variable Group Count field set to zero and is ignored at receiver	M	Yes [ ] No [ ]

Item	Feature	Subclause	Value/Comment	Status	Support
CP3	US PHY Link NACK data	102.3.2.2	Variable Group Data fields ignored if not omitted at CLT	CLT:M	Yes [ ] No [ ] N/A [ ]
CP4	DS EMB discrepancy	102.2.3.3	No action if Variable Group Count field disagrees with the number of 16-bit Variable Group Data fields, NACK returned.	CNU: M	Yes [ ] No [ ] N/A [ ]

102.5.4.10 PHY Discovery

Item	Feature	Subclause	Value/Comment	Status	Support
PD1	PHY Discovery Response FEC	102.4.1.4	(128,80) binary punctured LDPC code described in 102.1.4.2.2	M	Yes [ ] No [ ]
PD2	PHY Discovery preamble	102.4.1.5	BPSK modulated 128 bit sequence generated per Figure Figure 102-23	M	Yes [ ] No [ ]
PD3	PHY Discovery preamble symbol	102.4.1.5	second symbol duplicate of first per Figure 102-22	M	Yes [ ] No [ ]
PD4	CNU_ID allocation	102.4.1.6	Per Table 102-12	M	Yes [ ] No [ ]

102.5.4.11 Probes

Item	Feature	Subclause	Value/Comment	Status	Support
PR1	Probe data	102.4.2.4	BPSK modulated sequence generated per Figure Figure 102-28	CNU: M	Yes [ ] No [ ] N/A [ ]
PR2	Probe transmissions	102.4.2.3	No probe transmission outside confitured Probe Period	CNU: M	Yes [ ] No [ ] N/A [ ]
PR3	Probe transmit process	102.4.2.9	Meets the requirements of Figure 102-29	CNU: M	Yes [ ] No [ ] N/A [ ]

103. Multipoint MAC Control for EPoC

103.1 Overview

This clause deals with the mechanism and control protocols required in order to reconcile the EPoC (EPON protocol over coax) into the Ethernet framework. The P2MP medium is a coax cable distribution network (CCDN) in which passive and usually active elements are present in the signal's path from source to destination. When combined with the EPON protocol, such a network is referred to as EPON protocol over coax network (EPoC).

P2MP is an asymmetric medium based on a tree (or tree-and-branch) topology. The DTE connected to the trunk of the tree is called cable line terminal (CLT) and the DTEs connected at the branches of the tree are called cable network units (CNU). The CLT typically resides at the service provider's facility, while the CNUs are located at the subscriber premises.

The EPoC topology is similar to the P2MP topology of EPON with the optical line terminal being replaced by a CLT, the optical network units replaced by CNU and operating over a coaxial network rather than an optical network.

A simplified P2MP topology example is depicted in Figure 103–1. Clause 67 provides additional examples of P2MP topologies.

Topics dealt with in this clause include allocation of upstream transmission resources to different CNUs, discovery and registration of CNUs into the network, and reporting of congestion to higher layers to allow for dynamic bandwidth allocation schemes and statistical multiplexing across the CCDN.

This clause does not deal with topics including bandwidth allocation strategies, authentication of end-devices, quality-of-service definition, provisioning, or management.

This clause specifies the multipoint control protocol (MPCP) to operate a coax cable distribution network by defining a Multipoint MAC Control sublayer as an extension of the MAC Control sublayer defined in Clause 31, and supporting current and future operations as defined in Clause 31and annexes.

Each CCDN consists of a node located at the root of the tree assuming the role of CLT, and multiple nodes located at the tree leaves assuming roles of CNUs. The network operates by allowing CNUs multiplexed in frequency and in time to share the upstream medium. The MPCP located at the CLT is responsible for timing the different transmissions. Reporting of congestion by the different CNUs may assist in optimally allocating the bandwidth across the CCDN.

Automatic discovery of end stations is performed, culminating in registration through binding of a CNU to a CLT port by allocation of a Logical Link ID (see LLID 76.2.6.1.3.2), and dynamic binding to a MAC connected to the CLT.

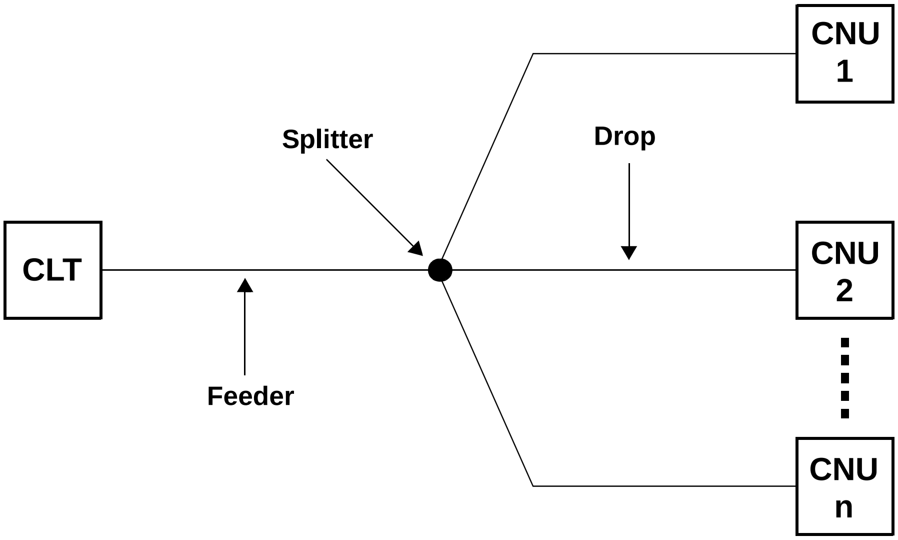


Figure 103–1—Coax cable distribution network topology example

The Multipoint MAC Control functionality shall be implemented for subscriber access devices containing point-to-multipoint Physical Layer devices defined in Clause 100, Clause 101 and Clause 102.

103.1.1 Goals and objectives

The goals and objectives of this clause are the definition of a point-to-multipoint Ethernet network utilizing a coax cable medium.

Specific objectives met include the following:

a) Support of point-to-point Emulation (P2PE) as specified
b) Support multiple LLIDs and MAC Clients at the CLT
c) Support a single LLID per CNU MAC
d) Support a mechanism for single copy broadcast
e) Flexible architecture allowing dynamic allocation of bandwidth
f) Use of 32 bit timestamp for MAC Control timing distribution
g) MAC Control based architecture
h) Ranging of discovered devices for improved network performance
i) Continuous ranging for compensating round trip time variation

103.1.2 Position of Multipoint MAC Control within the IEEE 802.3 hierarchy

Multipoint MAC Control defines the MAC control operation for optical point-to-multipoint networks. Figure 103–2 depicts the architectural positioning of the Multipoint MAC Control sublayer with respect to the MAC and the MAC Control client. The Multipoint MAC Control sublayer takes the place of the MAC Control sublayer to extend it to support multiple clients and additional MAC control functionality.

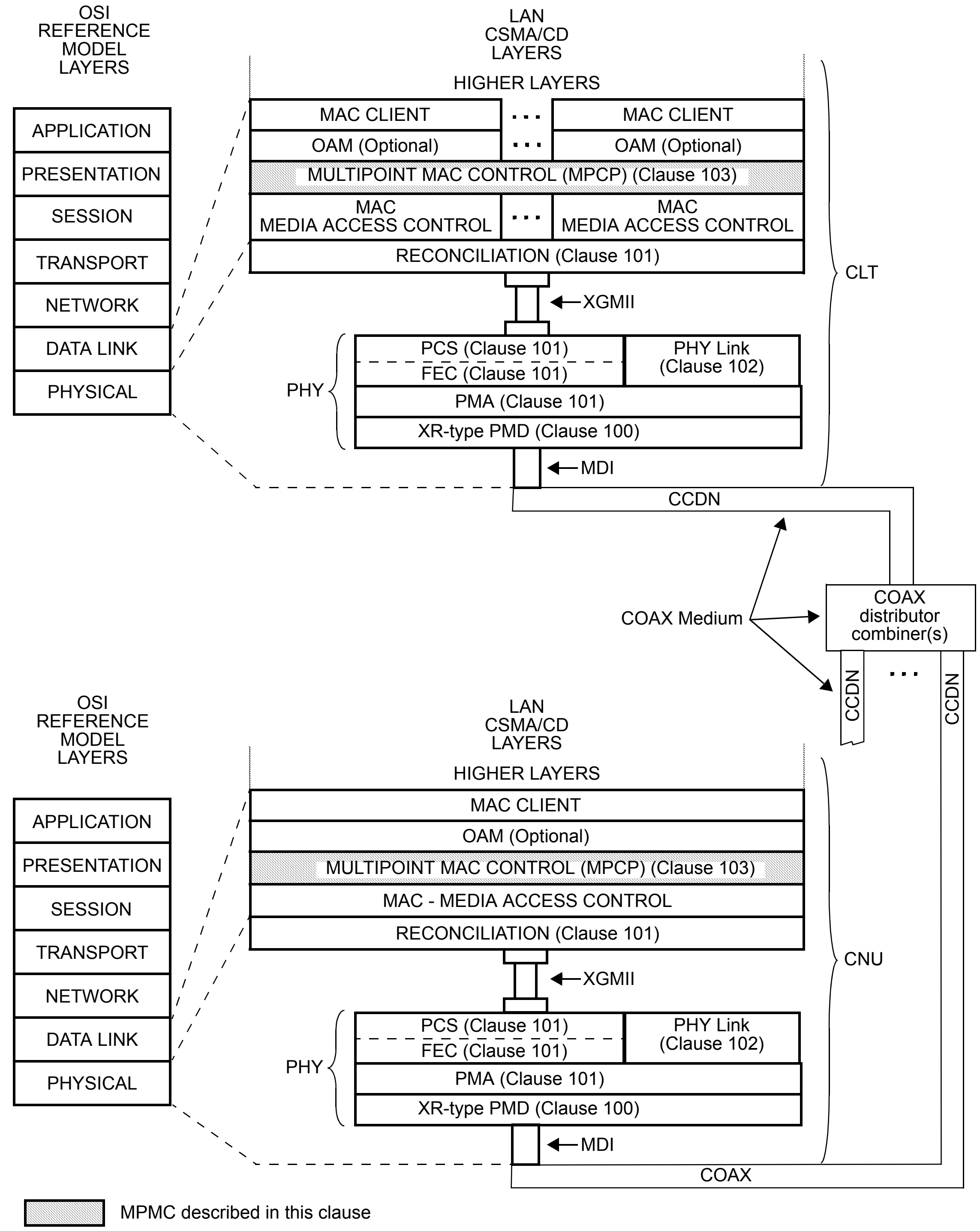
Multipoint MAC Control is defined using the mechanisms and precedents of the MAC Control sublayer. The MAC Control sublayer has extensive functionality designed to manage the real-time control and manipulation of MAC sublayer operation. This clause specifies the extension of the MAC Control mechanism to manipulate multiple underlying MACs simultaneously. This clause also specifies a specific protocol implementation for MAC Control.

The Multipoint MAC Control sublayer is specified such that it can support new functions to be implemented and added to this standard in the future. MultiPoint Control Protocol (MPCP), the management protocol for P2MP is one of these protocols. Non-real-time, or quasi-static control (e.g., configuration of MAC operational parameters) is provided by Layer Management. Operation of the Multipoint MAC Control sublayer is transparent to the MAC.

As depicted in Figure 103–2, the layered system instantiates multiple MAC entities, using a single Physical Layer. The individual MAC instances offer a point-to-point emulation service between the CLT and the CNU. An additional MAC is instantiated to communicate to all EPoC CNUs at once. This instance takes maximum advantage of the broadcast nature of the downstream channel by sending a single copy of a frame that is received by all EPoC CNUs. This MAC instance is referred to as Single Copy Broadcast (SCB).

The CNU only requires one MAC instance since frame filtering operations are done at the RS layer before reaching the MAC. Therefore, MAC and layers above are emulation-agnostic at the CNU (see 101.2.4.3).

Although Figure 103–2 and supporting text describe multiple MACs within the CLT, a single unicast MAC address may be used by the CLT. Within the EPoC Network, MACs are uniquely identified by their LLIDs, which are dynamically assigned by the registration process.



XGMII	= 10 GIGABIT MEDIA INDEPENDENT INTERFACE
MDI	= MEDIUM DEPENDENT INTERFACE
OAM	= OPERATIONS, ADMINISTRATION & MAINTENANCE
CLT	= COAX LINE TERMINAL
CNU	= COAX NETWORK UNIT
PCS	= PHYSICAL CODING SUBLAYER
PHY	= PHYSICAL LAYER DEVICE
PMA	= PHYSICAL MEDIUM ATTACHMENT
PMD	= PHYSICAL MEDIUM DEPENDENT

Figure 103–2—Relationship of Multipoint MAC Control and the OSI protocol stack for EPoC

103.1.3 Functional block diagram

Figure 103–3 provides a functional block diagram of the Multipoint MAC Control architecture.

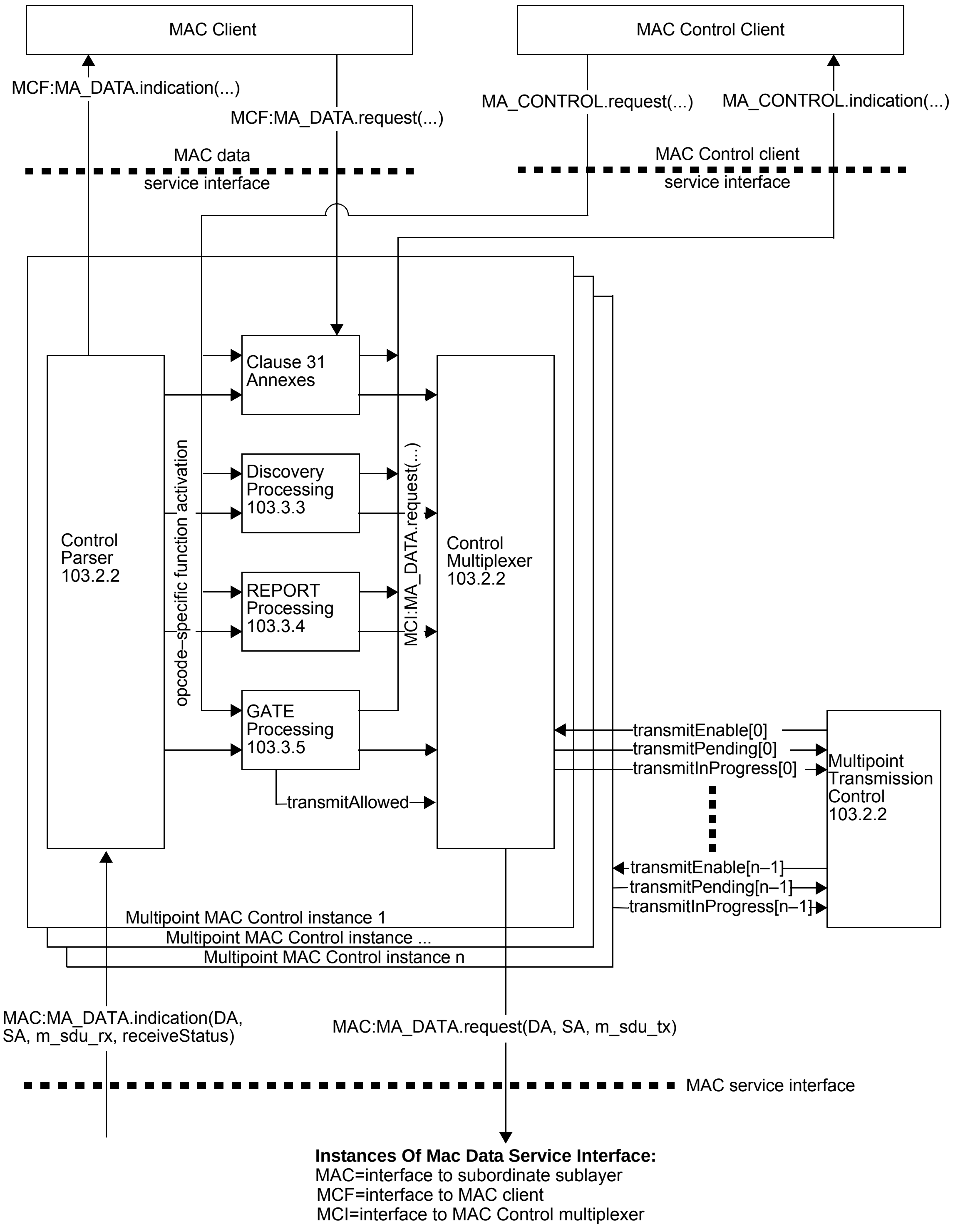


Figure 103–3—Multipoint MAC Control functional block diagram

103.1.4 Service interfaces

The MAC Client communicates with the Control Multiplexer using the standard service interface specified in 2.3. Multipoint MAC Control communicates with the underlying MAC sublayer using the standard service interface specified in Annex 4A.3.2. Similarly, Multipoint MAC Control communicates internally using primitives and interfaces consistent with definitions in Clause 31.

103.1.5 State diagram conventions

The body of this standard comprises 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 augmented as follows:

a) [start x_timer, y] sets expiration of y to timer x_timer.

b) [stop x_timer] aborts the timer operation for x_timer asserting x_timer_not_done indefinitely.

The state diagrams use an abbreviation MACR as a shorthand form for MA_CONTROL.request and MACI as a shorthand form for MA_CONTROL.indication.

The vector notations used in the state diagrams for bit vector use 0 to mark the first received bit and so on (for example data[0:15]), following the conventions of 3.1 for bit ordering. When referring to an octet vector, 0 is used to mark the first received octet and so on (for example m_sdu[0..1]).

a < b: A function that is used to compare two (cyclic) time values. Returned value is TRUE when b is larger than a allowing for wrap around of a and b. The comparison is made by subtracting b from a and testing the MSB. When MSB(a - b) = 1 the value TRUE is returned, else FALSE is returned. In addition, the following functions are defined in terms of a < b:
a > b is equivalent to !(a < b or a = b)
a ≥ b is equivalent to !(a < b)
a ≤ b is equivalent to !(a > b)

103.1.6 Other conventions

For equations used in this clause the symbol $\lceil x \rceil$ represents a ceiling function that rounds up ts argument *x* to the next highest integer. The symbol $\lfloor x \rfloor$ represents a floor function that rounds down its argument x to the next lowest integer.

103.2 Multipoint MAC Control operation

As depicted in Figure 103–3, the Multipoint MAC Control functional block comprises the following functions:

a) Multipoint Transmission Control. This block is responsible for synchronizing Multipoint MAC Control instances associated with the Multipoint MAC Control. This block maintains the Multipoint MAC Control state and controls the multiplexing functions of the instantiated MACs.

b) Multipoint MAC Control Instance n. This block is instantiated for each MAC and respective MAC and MAC Control clients associated with the Multipoint MAC Control. It holds all the variables and state associated with operating all MAC Control protocols for the instance.

c) Control Parser. This block is responsible for parsing MAC Control frames, and interfacing with Clause 31 entities, the opcode specific blocks, and the MAC Client.

d) Control Multiplexer. This block is responsible for selecting the source of the forwarded frames.

e) Clause 31 annexes. This block holds MAC Control actions as defined in Clause 31 annexes for support of legacy and future services.

f) Discovery, Report, and Gate Processing. These blocks are responsible for handling the MPCP in the context of the MAC.

103.2.1 Principles of Multipoint MAC Control

As depicted in Figure 103–3, Multipoint MAC Control sublayer may instantiate multiple Multipoint MAC Control instances in order to interface multiple MAC and MAC Control clients above with multiple MACs below. A unique unicast MAC instance is used at the CLT to communicate with each CNU. The individual MAC instances utilize the point-to-point emulation service between the CLT and the CNU as defined in 101.2.

At the CNU, a single MAC instance is used to communicate with a MAC instance at the CLT. In that case, the Multipoint MAC Control contains only a single instance of the Control Parser/Multiplexer function.

Multipoint MAC Control protocol supports several MAC and client interfaces. Only a single MAC interface and Client interface is enabled for transmission at a time. There is a tight mapping between a MAC service interface and a Client service interface. In particular, the assertion of the MAC:MA_DATA.indication primitive in MAC j leads to the assertion of the MCF:MA_DATA.indication primitive to Client j. Conversely, the assertion of the request service interface in Client i leads to the assertion of the MAC:MA_DATA.request primitive of MAC i. Note that the Multipoint MAC sublayer need not receive and transmit packets associated with the same interface at the same time. Thus the Multipoint MAC Control acts like multiple MAC Controls bound together with common elements.

The scheduling algorithm is implementation dependent, and is not specified for the case where multiple transmit requests happen at the same time.

The reception operation is as follows. The Multipoint MAC Control instances generate MAC:MA_DATA.indication service primitives continuously to the underlying MAC instances. Since these MACs are receiving frames from a single PHY only one frame is passed from the MAC instances to Multipoint MAC Control. The MAC instance responding to the MAC:MA_DATA.indication is referred to as the enabled MAC, and its service interface is referred to as the enabled MAC interface. The MAC passes to the Multipoint MAC Control sublayer all valid frames. Invalid frames, as specified in 3.4, are not passed to the Multipoint MAC Control sublayer in response to a MAC:MA_DATA.indication service primitive.

The enabling of a transmit service interface is performed by the Multipoint MAC Control instance in collaboration with the Multipoint Transmission Control. Frames generated in the MAC Control are given priority over MAC Client frames, in effect, prioritizing the MA_CONTROL primitive over the MCF:MA_DATA primitive, and for this purpose MCF:MA_DATA.request primitives may be delayed, discarded or modified in order to perform the requested MAC Control function. For the transmission of this frame, the Multipoint MAC Control instance enables forwarding by the MAC Control functions, but the MAC Client interface is not enabled. The reception of a frame in a MAC results in generation of the MAC:MA_DATA.indication primitive on that MAC’s interface. Only one receive MAC interface is enabled at any given time since there is only one PHY interface.

The information of the enabled interfaces is stored in the controller state variables, and accessed by the Multiplexing Control block.

The Multipoint MAC Control sublayer uses the services of the underlying MAC sublayer to exchange both data and control frames.

Receive operation (MAC:MA_DATA.indication) at each instance:

a) A frame is received from the underlying MAC

b) The frame is parsed according to Length/Type field

c) MAC Control frames are demultiplexed according to opcode and forwarded to the relevant processing functions

d) Data frames (see 31.5.1) are forwarded to the MAC Client by asserting MCF:MA_DATA.indication primitives

Transmit operation (MAC:MA_DATA.request) at each instance:

e) The MAC Client signals a frame transmission by asserting MCF:MA_DATA.request, or

f) A protocol processing block attempts to issue a frame, as a result of a previous MA_CONTROL.request or as a result of an MPCP event that generates a frame.

g) When allowed to transmit by the Multipoint Transmission Control block, the frame is forwarded.

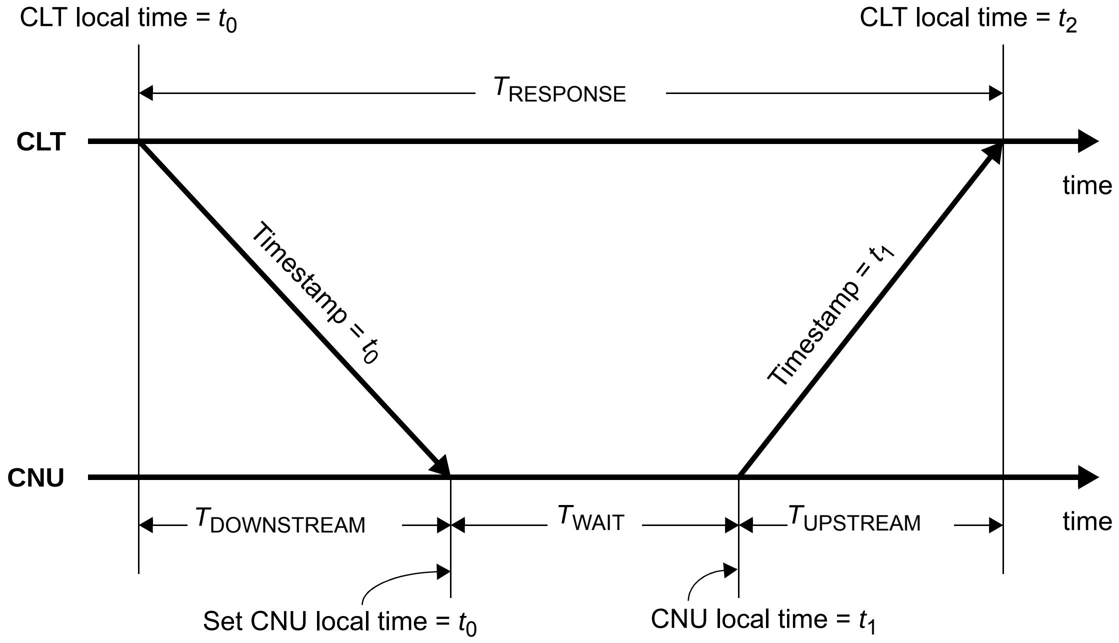
103.2.1.1 Ranging and timing process

Both the CLT and the CNU have 32 bit counters that increment every 16 ns. These counters provide a local time stamp. When either device transmits an MPCPDU, it maps its counter value into the *timestamp* field. The time of transmission of the first octet of the MPCPDU frame from the MAC Control to the MAC is taken as the reference time used for setting the *timestamp* value.

When the CNU receives MPCPDUs, it sets its counter according to the value in the *timestamp* field in the received MPCPDU.

When the CLT receives MPCPDUs, it uses the received *timestamp* value to calculate or verify a round trip time between the CLT and the CNU. The Round Trip Time (*RTT*) is equal to the difference between the timer value and the value in the *timestamp* field. The calculated *RTT* is notified to the client via the MA_CONTROL.indication primitive. The client can use this *RTT* for the ranging process.

A condition of timestamp drift error occurs when the difference between CLT's and CNU's clocks exceeds some predefined threshold. This condition can be independently detected by the CLT or a CNU. The CLT detects this condition when an absolute difference between new and old *RTT* values measured for a given CNU exceeds the value of guardThresholdCLT (see 103.2.2.1), as shown in Figure 103–11. a CNU detects the timestamp drift error condition when absolute difference between a *timestamp* received in an MPCPDU and the localTime counter exceeds guardThresholdCNU (see 103.2.2.1), as is shown in Figure 103–12.



$T_{\mathrm{DOWNSTREAM}}$ = downstream propagation delay

T_{UPSTREAM} = upstream propagation delay

T_{WAIT} = wait time at CNU = $t_1 - t_0$

T_{RESPONSE} = response time at CLT = $t_2 - t_0$

$$RTT = T_{\mathrm{DOWNSTREAM}} + T_{\mathrm{UPSTREAM}} = T_{\mathrm{RESPONSE}} - T_{\mathrm{WAIT}} = (t_2 - t_0) - (t_1 - t_0) = t_2 - t_1$$

Figure 103–4—Round trip time calculation

103.2.2 Multipoint transmission control, Control Parser, and Control Multiplexer

EDITORS NOTE (to be removed prior to publication): Figs 103-12 & 103-13 and associated variables and functions need updating.

The purpose of the multipoint transmission control is to allow only one of the multiple MAC clients to transmit to its associated MAC and subsequently to the RS layer at one time by only asserting one transmitEnable signal at a time.

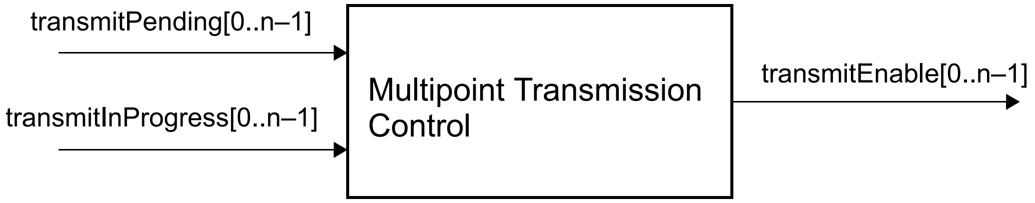


Figure 103–5—Multipoint Transmission Control service interfaces

Multipoint MAC Control Instance n function block communicates with the Multipoint Transmission Control using *transmitEnable[n]*, *transmitPending[n]*, and *transmitInProgress[n]* state variables (see Figure 103–3).

The Control Parser is responsible for opcode independent parsing of MAC frames in the reception path. By identifying MAC Control frames, demultiplexing into multiple entities for event handling is possible. Interfaces are provided to existing Clause 31 entities, functional blocks associated with MPCP, and the MAC Client.

The Control Multiplexer is responsible for forwarding frames from the MAC Control opcode-specific functions and the MAC Client to the MAC. Multiplexing is performed in the transmission direction. Given multiple MCF:MA_DATA.request primitives from the MAC Client, and MA_CONTROL.request primitives from the MAC Control Clients, a single MAC:MA_DATA.request service primitive is generated for transmission. At the CLT, multiple MAC instances share the same Multipoint MAC Control, as a result, the transmit block is enabled based on an external control signal housed in Multipoint Transmission Control for transmission overlap avoidance. At the CNU, the Gate Processing functional block interfaces for upstream transmission administration.

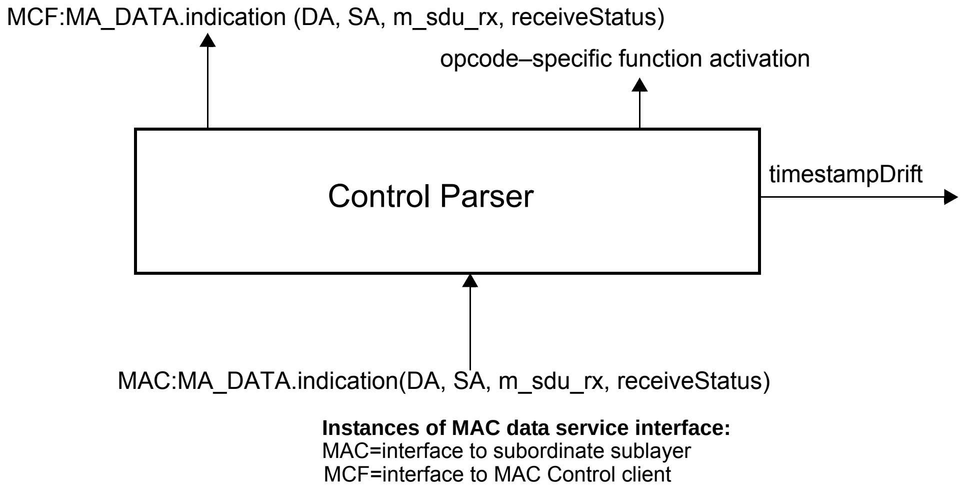
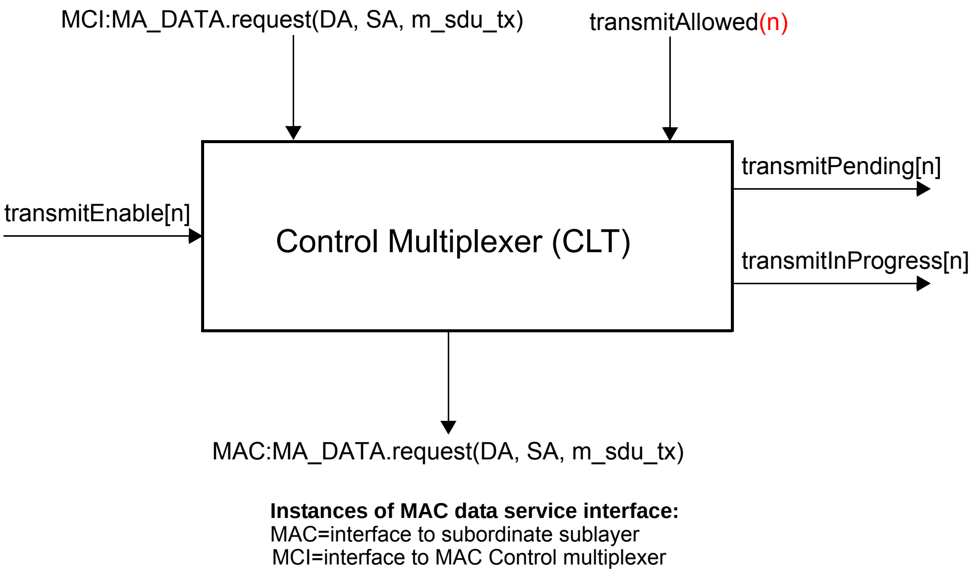
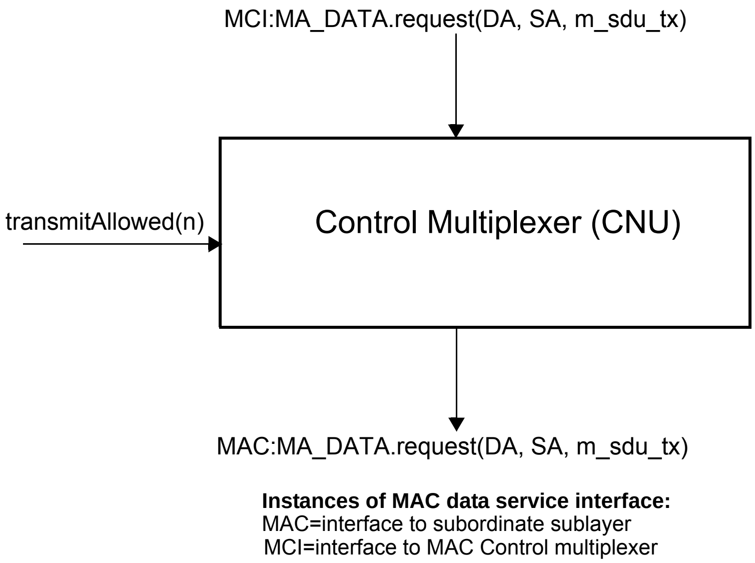

MCF:MA_DATA.indication (DA, SA, m_sdu_rx, receiveStatus)
opcode–specific function activation
Control Parser
timestampDrift
MAC:MA_DATA.indication(DA, SA, m_sdu_rx, receiveStatus)
Instances of MAC data service interface:
MAC=interface to subordinate sublayer
MCF=interface to MAC Control client

Figure 103–6—Control Parser service interfaces


MCI:MA_DATA.request(DA, SA, m_sdu_tx)
transmitAllowed(n)
transmitEnable[n]
Control Multiplexer (CLT)
transmitPending[n]
transmitInProgress[n]
MAC:MA_DATA.request(DA, SA, m_sdu_tx)
Instances of MAC data service interface:
MAC=interface to subordinate sublayer
MCI=interface to MAC Control multiplexer

NOTE—MAC:MA_DATA.request primitive may be issued from multiple MAC Control processing blocks.

Figure 103–7—CLT Control Multiplexer service interfaces


MCI:MA_DATA.request(DA, SA, m_sdu_tx)
transmitAllowed(n)
Control Multiplexer (CNU)
MAC:MA_DATA.request(DA, SA, m_sdu_tx)
Instances of MAC data service interface:
MAC=interface to subordinate sublayer
MCI=interface to MAC Control multiplexer

NOTE—MAC:MA_DATA.request primitive may be issued from multiple MAC Control processing blocks.

Figure 103–8—CNU Control Multiplexer service interfaces

103.2.2.1 Constants

DS_FEC_CW_Sz

TYPE: integer

This constant represents the approximate size of FEC codeword in whole octets (DS_FEC_Pld_Sz + DS_FEC_Prty_Sz).

Value: 1987

DS_FEC_CW_Sz_FRAC

TYPE: real number

This constant represents the exact size of the FEC codeword in whole and fractional octets.

Value: 1760+2944/13 (1760 +(1840*64/65/8)

DS_FEC_Prty_Sz

TYPE: integer

This constant represents the size of FEC codeword parity field in octets.

Value: 227

DS_FEC_Pld_Sz

TYPE: integer

This constant represents the size of FEC codeword payload in octets.

VALUE: 1760 1760 (220 block of 64-bits as seen from the MAC Table 101-2)

guardThresholdCLT

TYPE: integer

This constant holds the maximum amount of drift allowed for a *timestamp* received at the CLT. This value is measured in units of time_quantum.

VALUE: 12

guardThresholdCNU

TYPE: integer

This constant holds the maximum amount of drift allowed for a *timestamp* received at the CNU. This value is measured in units of time_quantum.

VALUE: 8

MAC_Control_type

TYPE: integer

The value of the Length/Type field as defined in 64.2.2.1.

VALUE: 0x8808

tailGuard

TYPE: integer

This constant holds the value used to reserve space at the end of the upstream transmission at the CNU in addition to the size of last MAC service data unit (m_sdu) in units of octets. Space is reserved for the MAC overheads including: preamble, SFD, DA, SA, Length/Type, FCS, and minimum interpacket gap. The sizes of the above listed MAC overhead items are described in 3.1.1. The size of the minimum IPG is described in 4A.4.2.

VALUE: 38

time_quantum

This constant is defined in 64.2.2.1.

tqSizeC

TYPE: integer

This constant represents octet transmission times in 128 time_quantum.

VALUE: TBD

EDITORS NOTE (to be removed prior to publication): the list of constants will be updated per technical decision #44 (http://www.ieee802.org/3/bn/public/ decisions/decisions.html) once EPoC-specific FEC and PMD *overhead details are settled.*

103.2.2.2 Counters

localTime

TYPE: 32 bit unsigned

This variable holds the value of the local timer used to control MPCP operation and is defined in 64.2.2.2.

103.2.2.3 Variables

BEGIN

TYPE: Boolean

This variable is used when initiating operation of the functional block state diagram. It is set to TRUE following initialization and every reset.

data_rx

TYPE: bit array

This variable represents a 0-based bit array corresponding to the payload of a received MPCPDU and is defined in 64.2.2.3.

data_tx

TYPE: bit array

This variable represents a 0-based bit array corresponding to the payload of an MPCPDU being transmitted and is defined in 64.2.2.3.

fecOffset

TYPE: 32 bit unsigned

A variable that advances by one after every octet time. In the CLT, after reaching the value of DS_FEC_CW_Sz, this variable is held for a period of time for PMD derating and then reset to zero as illustrated in Figure 103–9. In the CNU, this variable is assigned in Figure 103–14.

grantStart

TYPE: Boolean

This variable indicates beginning of a grant transmission and is defined in 77.2.2.3.

IdleGapCount

TYPE: 32-bit unsigned

This variable represents length of gap between subsequent frames, expressed in the unit of octet time. This variable advances by 1 after every 8-bit times.

initial_derating_delay

TYPE: 24 bit unsigned

This variable is used to set the time-out interval for derating_timer defined in 103.2.2.5. The *initial_derating_delay* value is represented in units of octets.

newRTT

TYPE: 16 bit unsigned

This variable temporarily holds a newly-measured Round Trip Time and is defined in 64.2.2.3.

m_sdu_rx

TYPE: bit array

Equal to the concatenation of the Length/Type and *data_rx* variables and is defined in 77.2.2.3.

m_sdu_tx

TYPE: bit array

Equal to the concatenation of the Length/Type and *data_tx* variables and is defined in 77.2.2.3.

m_sdu_ctl

TYPE: bit array

Equal to the concatenation of the MAC_Control_type and *data_tx* variables and is defined in 77.2.2.3.

Octet_CLK

TYPE: Boolean

This boolean value is TRUE for every octet time period, i.e. the amount of time used to transmit one octet in 10Gb/s MAC data rate.

OctetsRemaining

TYPE: 32 bit unsigned

This variable is an alias for the expression (((*stopTime* – localTime) × tqSizeC/128) – *tqOffset*). It denotes the number of octets that can be transmitted between the current time and the end of the grant.

OctetsRequired

TYPE: 16 bit unsigned

This variable represents a total transmission time of next packet and is used to check whether the next packet fits in the remainder of the transmission window. The value of *OctetsRequired* includes packet transmission time, tailGuard defined in 103.2.2.1, and FEC parity data overhead. This variable is measured in units of octets.

opcode_rx

TYPE: 16 bit unsigned

This variable holds an opcode of the last received MPCPDU and is defined in 64.2.2.3.

opcode_tx

TYPE: 16 bit unsigned

This variable holds an opcode of an outgoing MPCPDU and is defined in 64.2.2.3.

packet_initiate_delay

TYPE: 18 bit unsigned

This variable is used to set the time-out interval for packet_initiate_timer defined in 103.2.2.5. The *packet_initiate_delay* value is represented in units of octets.

PCS_Rate

See 101.3.2.1.2 and Equation (101–1).

ResetBound

TYPE: 32-bit unsigned

|This variable represents the value of DelayBound (see 101.3.2.1.2) expressed in units of octet time (i.e., ResetBound = 8 * DelayBound).

RTT

TYPE: 16 bit unsigned

This variable holds the measured Round Trip Time to the CNU. The *RTT* value is represented in units of time_quanta.

stopTime

TYPE: 32 bit unsigned

This variable holds the value of the localTime counter corresponding to the end of the nearest grant and is defined in 64.2.2.3.

timestamp

TYPE: 32 bit unsigned

This variable holds the value of *timestamp* of the last received MPCPDU frame and is defined in 64.2.2.3.

timestampDrift

TYPE: Boolean
This variable is used to indicate whether an error is signaled as a result of uncorrectable timestamp drift and is defined in 64.2.2.3.

tqOffset

TYPE: 8 bit unsigned
This variable denotes the offset (in octet times) of the current actual time from the localTime variable and is defined in 77.2.2.3.

transmitAllowed

TYPE: Boolean
This variable is used to control PDU transmission at the CNU and at the CLT and is defined in 64.2.2.3.

transmitEnable

TYPE: Boolean array
This array is defined in 64.2.2.3.

transmitInProgress

TYPE: Boolean array
This array is defined in 64.2.2.3.

transmitPending

TYPE: Boolean array
This is defined in 64.2.2.3.

EDITORS NOTE (to be removed prior to publication): The list of variables will be updated per technical decision #44 (http://www.ieee802.org/3/bn/public/ decisions/decisions.html) once EPoC-specific FEC and PMD overhead details are settled.

103.2.2.4 Functions

abs(n)

This function returns the absolute value of the parameter n.

Opcode–specific function(opcode)

Functions exported from opcode specific blocks that are invoked on the arrival of a MAC Control message of the appropriate opcode.

CheckGrantSize(length)

This function calculates the future time at which the transmission of the current frame (including the FEC parity overhead) is completed.

```
GntSize = 0
length = length + fecOffset
Do While length > 0
    i = 0               ' Calculate size of  full CWs except smallest fec
    For i = 0 To fecNum – 2' for each FEC
        Do While length > fecPldSz(i)
        length = length - fecPldSz(i)
        GntSize = GntSize + fecCwSz(i)
    Loop
    i = 0‘              ' Calculate size for remaining data
    For i = 0 To fecNum – 1' for each FEC
        tmpCw = Fix(length / fecPldSz(i))' Fix is a floor function (rounds towards 0)
        If length - tmpCw * fecPldSz(i) > 0 Then‘ shortened last CW
            cwFit(i) = length + (tmpCw + 1) * fecPrtySz(i)
        Else        ' no shortened last CW
```

```
                cwFit(i) = length + tmpCw * fecPrtySz(i)
            End If
        Next i
        i = 0
        For i = 0 To fecNum – 2‘ for each FEC' select optimal Size for remaining length
            If cwFit(i) < cwFit(i + 1) Then
                GntSize = GntSize + cwFit(i)
                length = length - cwFit(i)
            End If
        Next i
        If length > 0 Then ' pick up smallest shortened cw if len >  0
            GntSize = GntSize + cwFit(2)
            length = length - cwFit(2)
        End If
Loop
CheckGrantSize = GntSize - fecOffset' return calculated length
```

EDITORS NOTE (to be removed prior to publication): this function definition will need to be converted from visual basic into “quasi C”

PHY_Overhead(length, β)

This function calculates the additional amount of time (in octet times) that the MPCP control multiplexer waits following transmission of a frame of size ‘length’ by the MAC. The additional time is added to allow the insertion of parity data into the frame by the PHY layer and to adjust the data rate to the effective data rate supported by the PCS and PMD. PHY_Overhead().-returns the number of octets that the PHY inserts during transmission of a particular packet. Parameter ‘length’ represents the size of an entire frame including preamble, SFD, DA, SA, Length/Type, FCS, and IPG which are accounted for in tailGuard. The following formula is used to calculate the overhead:

$$PMD\_Overhead(length,\beta) \ = \ FEC\_Overhead(length) + Derating\_Overhead(length,\beta)$$

$$FEC\_Overhead(length) \ = \ 12 + FEC\_PARITY\_SIZE \times \left\lfloor \frac{fecOffset + length}{FEC\_PAYLOAD\_SIZE} \right\rfloor$$

$$Derating\_Overhead(length,\beta) \ =$$

$$\left\lceil (\beta - 1) \times FEC\_CODEWORD\_SIZE\_FRAC \times \left\lfloor \frac{fecOffset + length}{FEC\_PAYLOAD\_SIZE} \right\rfloor \right\rceil$$

$$\beta \ = \ \frac{XGMII\_Rate}{PCS\_Rate}$$

select()

This function selects the next Multipoint MAC Control instance allowed to initiate transmission of a frame. The function returns an index to the *transmitPending* array for which the value is not FALSE. The selection criteria in the presence of multiple active elements in the list is implementation dependent.

SelectFrame()

This function enables the interface, which has a pending frame. If multiple interfaces have frames waiting at the same time, only one interface is enabled. The selection criteria is not specified, except for the case when some of the pending frames have Length/Type = MAC_Control. In this case, one of the interfaces with a pending MAC Control frame shall be enabled.

sizeof(sdu)

This function returns the size of the sdu in octets.

transmissionPending()

This function returns TRUE if any of the Multipoint MAC Control instances has a frame waiting to be transmitted and is defined in 64.2.2.4.

EDITORS NOTE (to be removed prior to publication): the list of function will be updated per technical decision #44 (http://www.ieee802.org/3/bn/public/ decisions/decisions.html) once EPoC-specific FEC and PMD overhead details are settled. In particular, further checks are needed for the function CheckGrantSize(), in relation to data rata adaption changes.

103.2.2.5 Timers

derating_timer:

This timer is used to suspend the advancing of *fecOffset* in order to derate the MAC frame transmission to be able to match the PMD rate.

packet_initiate_timer

This timer is used to delay frame transmission from MAC Control to avoid variable MAC delay while MAC enforces IPG after a previous frame and is defined in 64.2.2.5.

103.2.2.6 Messages

MAC:MA_DATA.indication(DA, SA, m_sdu, receiveStatus)

The service primitive is defined in 31.3.

MCF:MA_DATA.indication(DA, SA, m_sdu, receiveStatus)

The service primitive is defined in 31.3.

MAC:MA_DATA.request (DA, SA, m_sdu)

The service primitive is defined in 31.3. The action invoked by this service primitive is not considered to end until the transmission of the frame by the MAC has concluded. The ability of the MAC control layer to determine this is implementation dependent.

MCF:MA_DATA.request (DA, SA, m_sdu)

The service primitive is defined in 31.3.

103.2.2.7 State diagrams

The Multipoint transmission control function in the CLT shall implement state diagram shown in Figure 103–10. Control parser function in the CLT shall implement state diagram shown in Figure 103–11. Control parser function in the CNU shall implement state diagram shown in Figure 103–12. Control multiplexer function in the CLT shall implement state diagram shown in Figure 103–13. Control multiplexer function in the CNU shall implement state diagram shown in Figure 103–14.

EDITORS NOTE (to be removed prior to publication): Figures 102–13 and 102–14 will be updated per technical decision #44 (http://www.ieee802.org/3/bn/public/decisions/decisions.html) once EPoC-specific FEC and PMD over-head details are settled.

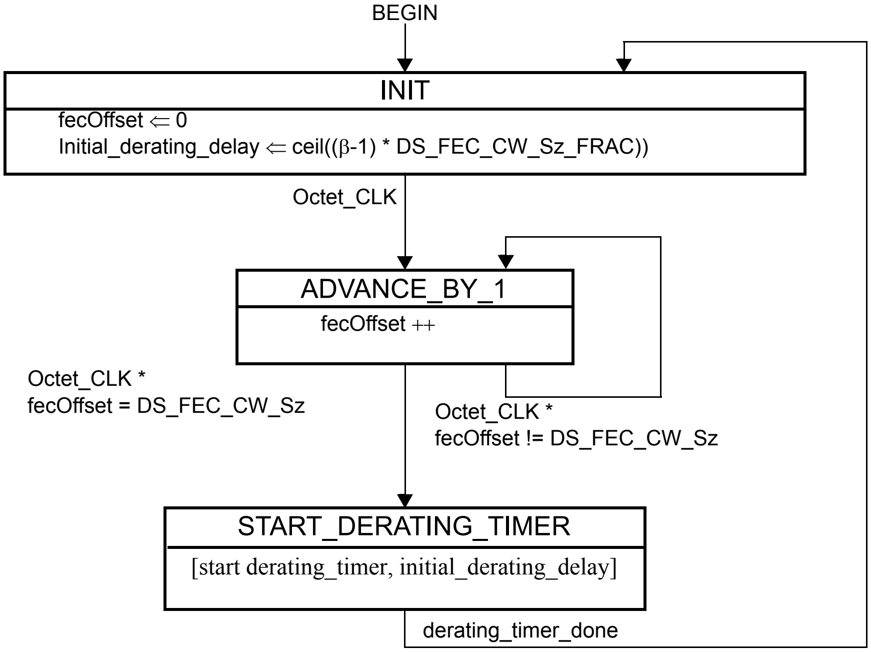


Figure 103–9—CLT fecOffset state diagram

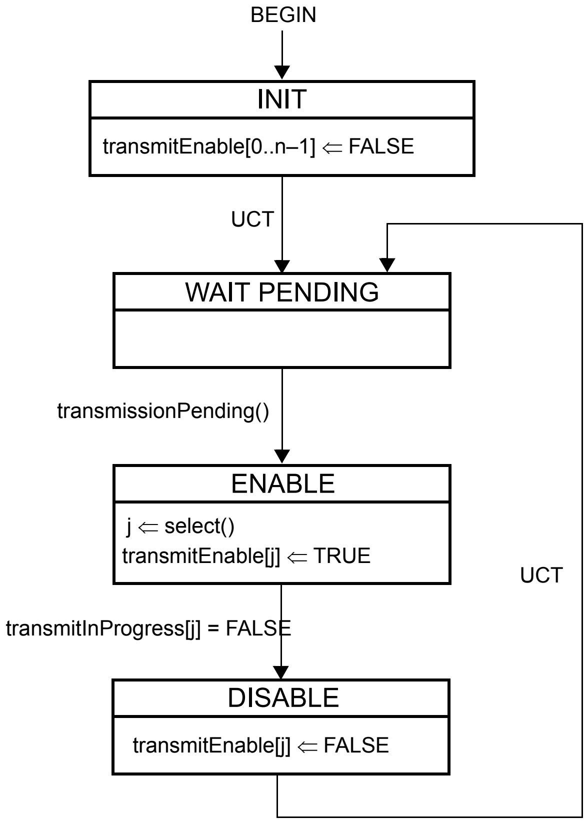
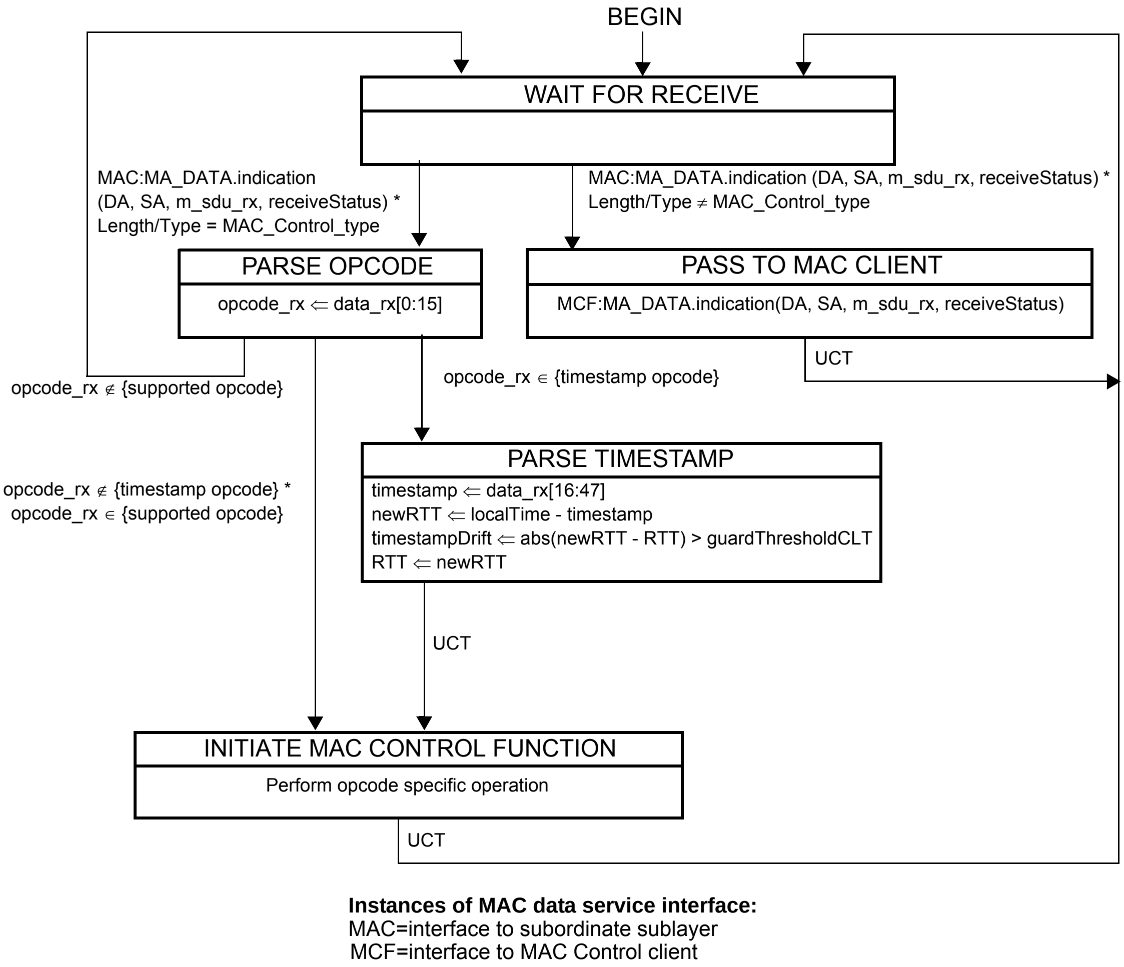


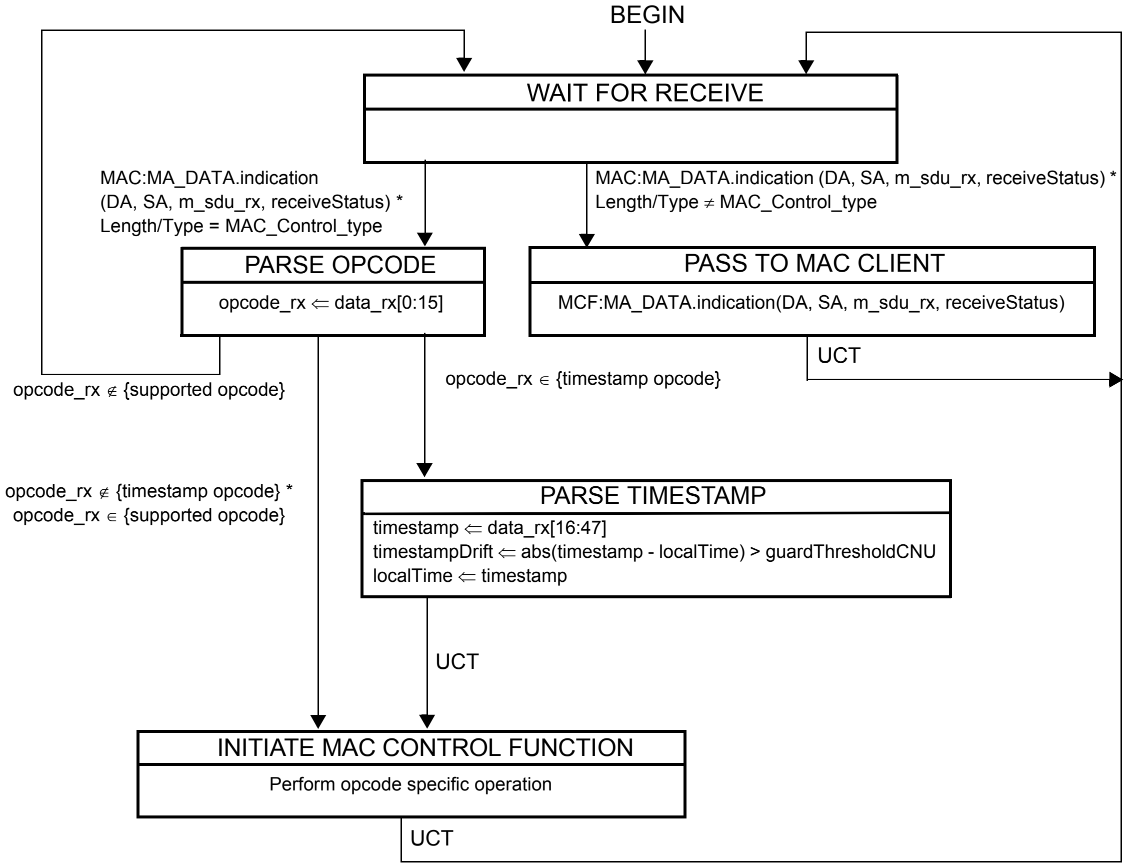
Figure 103–10—CLT Multipoint Transmission Control state diagram


BEGIN
WAIT FOR RECEIVE
MAC:MA_DATA.indication
(DA, SA, m_sdu_rx, receiveStatus) *
Length/Type = MAC_Control_type
MAC:MA_DATA.indication (DA, SA, m_sdu_rx, receiveStatus) *
Length/Type ≠ MAC_Control_type
PARSE OPCODE
opcode_rx ⇐ data_rx[0:15]
PASS TO MAC CLIENT
MCF:MA_DATA.indication(DA, SA, m_sdu_rx, receiveStatus)
UCT
opcode_rx ∉ {supported opcode}
opcode_rx ∈ {timestamp opcode}
PARSE TIMESTAMP
opcode_rx ∉ {timestamp opcode} *
opcode_rx ∈ {supported opcode}
timestamp ⇐ data_rx[16:47]
newRTT ⇐ localTime - timestamp
timestampDrift ⇐ abs(newRTT - RTT) > guardThresholdCLT
RTT ⇐ newRTT
UCT
INITIATE MAC CONTROL FUNCTION
Perform opcode specific operation
UCT
Instances of MAC data service interface:
MAC=interface to subordinate sublayer
MCF=interface to MAC Control client

NOTE—The opcode–specific operation is launched as a parallel process by the MAC Control sublayer, and not as a synchronous function. Progress of the generic MAC Control Receive state diagram (as shown in this figure) is not implicitly impeded by the launching of the opcode specific function.

Refer to Annex 31A for list of supported opcodes and timestamp opcodes.

Figure 103–11—CLT Control Parser state diagram

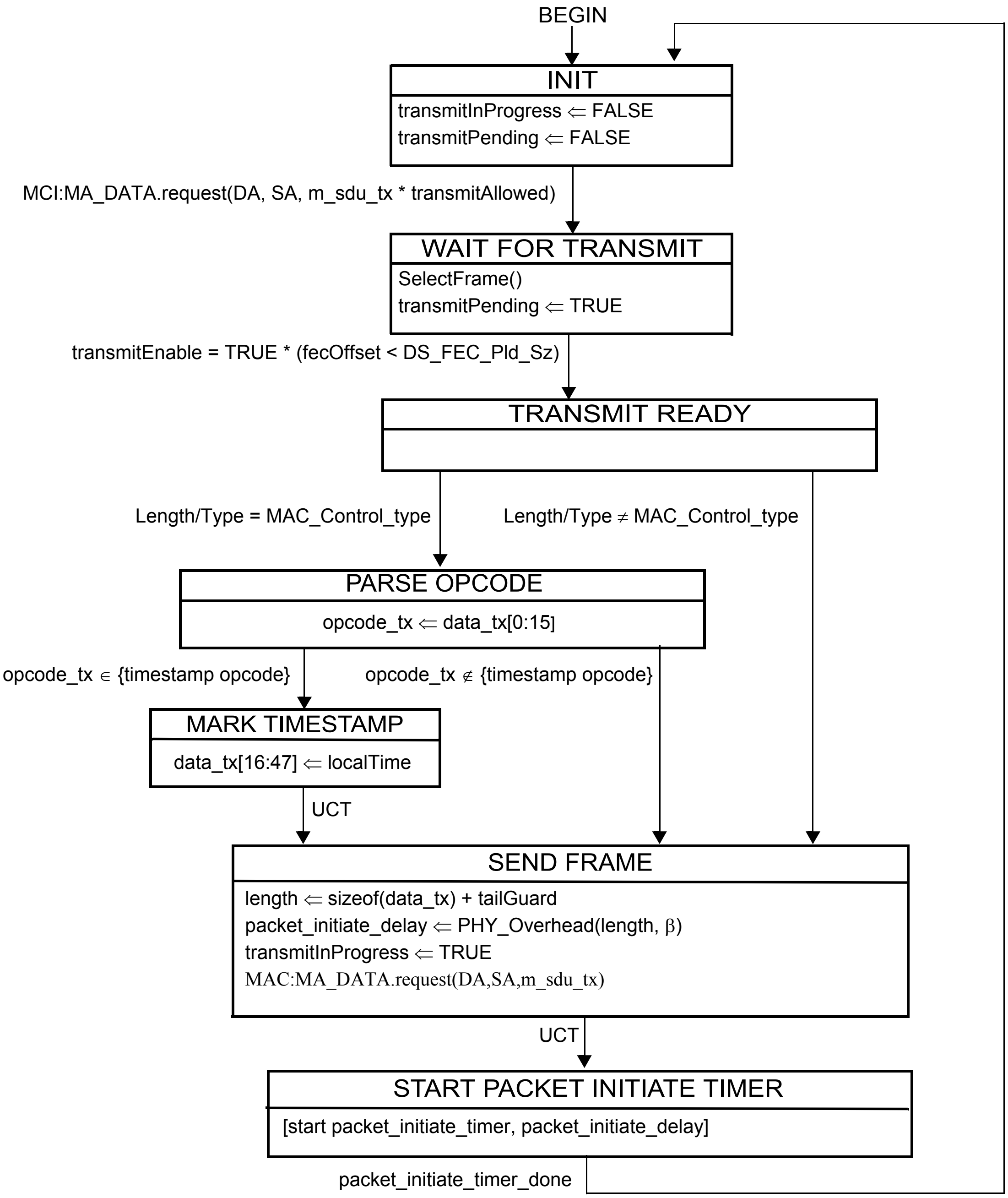

BEGIN
WAIT FOR RECEIVE
MAC:MA_DATA.indication
(DA, SA, m_sdu_rx, receiveStatus) *
Length/Type = MAC_Control_type
MAC:MA_DATA.indication (DA, SA, m_sdu_rx, receiveStatus) *
Length/Type ≠ MAC_Control_type
PARSE OPCODE
opcode_rx ⇐ data_rx[0:15]
PASS TO MAC CLIENT
MCF:MA_DATA.indication(DA, SA, m_sdu_rx, receiveStatus)
UCT
opcode_rx ∉ {supported opcode}
opcode_rx ∈ {timestamp opcode}
opcode_rx ∉ {timestamp opcode} *
opcode_rx ∈ {supported opcode}
PARSE TIMESTAMP
timestamp ⇐ data_rx[16:47]
timestampDrift ⇐ abs(timestamp - localTime) > guardThresholdCNU
localTime ⇐ timestamp
UCT
INITIATE MAC CONTROL FUNCTION
Perform opcode specific operation
UCT

Instances of MAC data service interface:
MAC=interface to subordinate sublayer
MCF=interface to MAC Control client

NOTE—The opcode–specific operation is launched as a parallel process by the MAC Control sublayer, and not as a synchronous function. Progress of the generic MAC Control Receive state diagram (as shown in this figure) is not implicitly impeded by the launching of the opcode specific function.

Refer to Annex 31A for list of supported opcodes and timestamp opcodes.

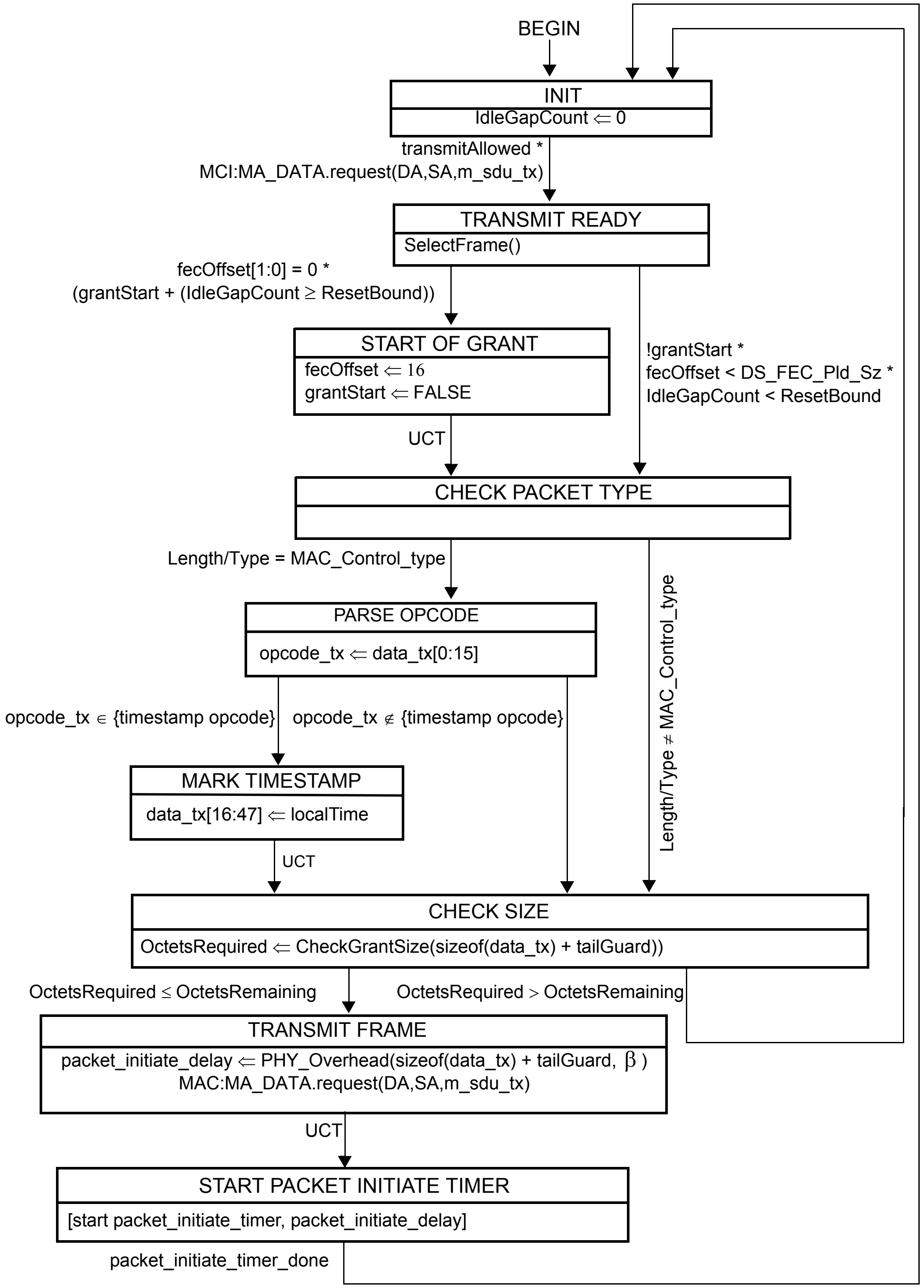
Figure 103–12—CNU Control Parser state diagram


BEGIN
INIT
transmitInProgress ⇐ FALSE
transmitPending ⇐ FALSE
MCI:MA_DATA.request(DA, SA, m_sdu_tx * transmitAllowed)
WAIT FOR TRANSMIT
SelectFrame()
transmitPending ⇐ TRUE
transmitEnable = TRUE * (fecOffset < DS_FEC_Pld_Sz)
TRANSMIT READY
Length/Type = MAC_Control_type
Length/Type ≠ MAC_Control_type
PARSE OPCODE
opcode_tx ⇐ data_tx[0:15]
opcode_tx ∈ {timestamp opcode}
opcode_tx ∉ {timestamp opcode}
MARK TIMESTAMP
data_tx[16:47] ⇐ localTime
UCT
SEND FRAME
length ⇐ sizeof(data_tx) + tailGuard
packet_initiate_delay ⇐ PHY_Overhead(length, β)
transmitInProgress ⇐ TRUE
MAC:MA_DATA.request(DA,SA,m_sdu_tx)
UCT
START PACKET INITIATE TIMER
[start packet_initiate_timer, packet_initiate_delay]
packet_initiate_timer_done

Instances of MAC data service interface:
MAC=interface to subordinate sublayer
MCI=interface to MAC Control multiplexer

Figure 103–13—CLT Control Multiplexer state diagram

EDITORS NOTE (to be removed prior to publication): Figs 103-12 & 103-13 and associated variables and functions need updating.


BEGIN
INIT
IdleGapCount ⇐ 0
transmitAllowed *
MCI:MA_DATA.request(DA,SA,m_sdu_tx)
TRANSMIT READY
SelectFrame()
fecOffset[1:0] = 0 *
(grantStart + (IdleGapCount ≥ ResetBound))
!grantStart *
fecOffset < DS_FEC_Pld_Sz *
IdleGapCount < ResetBound
START OF GRANT
fecOffset ⇐ 16
grantStart ⇐ FALSE
UCT
CHECK PACKET TYPE
Length/Type = MAC_Control_type
Length/Type ≠ MAC_Control_type
PARSE OPCODE
opcode_tx ⇐ data_tx[0:15]
opcode_tx ∈ {timestamp opcode}
opcode_tx ∉ {timestamp opcode}
MARK TIMESTAMP
data_tx[16:47] ⇐ localTime
UCT
CHECK SIZE
OctetsRequired ⇐ CheckGrantSize(sizeof(data_tx) + tailGuard))
OctetsRequired ≤ OctetsRemaining
OctetsRequired > OctetsRemaining
TRANSMIT FRAME
packet_initiate_delay ⇐ PHY_Overhead(sizeof(data_tx) + tailGuard, β)
MAC:MA_DATA.request(DA,SA,m_sdu_tx)
UCT
START PACKET INITIATE TIMER
[start packet_initiate_timer, packet_initiate_delay]
packet_initiate_timer_done

Instances of MAC data service interface:
MAC=interface to subordinate sublayer
MCI=interface to MAC Control multiplexer

Figure 103–14—CNU Control Multiplexer state diagram

EDITORS NOTE (to be removed prior to publication): Figs 103-12 & 103-13 and associated variables and functions need updating.

103.3 Multipoint Control Protocol (MPCP)

As depicted in Figure 103–3, the Multipoint MAC Control functional block comprises the following functions:

a) Discovery Processing. This block manages the discovery process, through which a CNU is discovered and registered with the network while compensating for *RTT*.

b) Report Processing. This block manages the generation and collection of report messages, through which bandwidth requirements are sent upstream from the CNU to the CLT.

c) Gate Processing. This block manages the generation and collection of gate messages, through which multiplexing of multiple transmitters is achieved.

As depicted in Figure 103–3, the layered system may instantiate multiple MAC entities, using a single Physical Layer. Each instantiated MAC communicates with an instance of the opcode specific functional blocks through the Multipoint MAC Control. In addition some global variables are shared across the multiple instances. Common state control is used to synchronize the multiple MACs using MPCP procedures. Operation of the common state control is generally considered outside the scope of this document.

103.3.1 Principles of Multipoint Control Protocol

Multipoint MAC Control enables a MAC Client to participate in a point-to-multipoint CCDN by allowing it to transmit and receive frames as if it was connected to a dedicated link. In doing so, it employs the following principles and concepts:

a) A MAC client transmits and receives frames through the Multipoint MAC Control sublayer.

b) The Multipoint MAC Control decides when to allow a frame to be transmitted using the client interface Control Multiplexer.

c) Given a transmission opportunity, the MAC Control may generate control frames that would be transmitted in advance of the MAC Client’s frames, utilizing the inherent ability to provide higher priority transmission of MAC Control frames over MAC Client frames.

d) Multiple MACs operate on a shared medium by allowing only a single MAC to transmit upstream across the network at any given time and frequency a using an Orthogonal Frequency Division Multiple Access (OFDMA) method.

e) Such gating of upstream transmission is orchestrated through the Gate Processing function.

f) New devices are discovered in the network and allowed transmission through the Discovery Processing function.

g) Fine control of the network bandwidth distribution can be achieved using feedback mechanisms supported in the Report Processing function.

h) The operation of P2MP network is asymmetric, with the CLT assuming the role of master, and the CNU assuming the role of slave.

103.3.2 Compatibility considerations

103.3.2.1 PAUSE operation

Even though MPCP is compatible with flow control, optional use of flow control may not be efficient in the case of large propagation delay. If flow control is implemented, then the timing constraints in Annex 31B supplement the constraints found at 103.3.2.4.

NOTE—AC at a CNU can receive frames from unicast channel and SCB channel. If the SCB channel is used to broadcast data frames to multiple CNUs, the CNU's MAC may continue receiving data frames from SCB channel even after the CNU has issued a PAUSE request to its unicast remote-end.

103.3.2.2 Optional Shared LAN emulation

By combining P2PE, suitable filtering rules at the CNU, and suitable filtering and forwarding rules at the CLT, it is possible to emulate an efficient shared LAN. Support for shared LAN emulation is optional, and requires an additional layer above the MAC, which is out of scope for this document. Thus, shared LAN emulation is introduced here for informational purposes only.

Specific behaviour of the filtering layer at the RS is specified in 101.2.4.3.

103.3.2.3 Multicast and single copy broadcast support

In the downstream direction, the CCDN is a broadcast medium. In order to make use of this capability for forwarding broadcast frames from the CLT to multiple recipients without multiple duplication for each CNU, the SCB and multicast LLID support is introduced.

The CLT has at least one MAC associated with every CNU. In addition one more MAC at the CLT is marked as the SCB MAC. Moreover, the CLT has a multicast MAC associated with each defined multicast LLID. The SCB MAC handles all downstream broadcast traffic, but is never used in the upstream direction for client traffic, except for client registration. Similarly, the multicast MACs handle downstream multicast traffic, but are never used in the upstream direction for client traffic. Optional higher layers may be implemented to perform selective broadcast and multicast of frames. Such layers may require additional MACs (multicast MACs) to be instantiated in the CLT for some or all CNUs increasing the total number of MACs beyond the number of CNUs + 1.

When connecting the SCB MAC or a multicast MAC to an IEEE 802.1D bridge port it is possible that loops may be formed due to the broadcast or multicast nature of the associated LLIDs. Thus it is recommended that this MAC not be connected to an IEEE 802.1D bridge port.

Configuration of SCB channels as well as filtering and marking of frames for support of SCB is defined in 101.2.4.3 for EPoC compliant Reconciliation Sublayers.

103.3.2.4 Delay requirements

The MPCP protocol relies on strict timing based on distribution of timestamps. A compliant implementation needs to guarantee a constant delay through the MAC and PHY in order to maintain the correctness of the timestamping mechanism. The actual delay is implementation dependent; however, a complying implementation shall maintain a delay variation of no more than 1 time_quantum through the MAC.

The CLT shall not grant less than TBD time_quanta into the future, in order to allow the CNU processing time when it receives a gate message. The CNU shall process all messages in less than this period. The CLT shall not issue more than one message every TBD time_quanta to a single CNU. The unit of time_quantum is defined in 64.2.2.1.

103.3.3 Discovery processing

EDITORS NOTE (to be removed prior to publication): Highlighted material below on Discovery processing and State diagrams needs to be rationalized with CL 101 and 102. If no comments are received on this material in the next comment round it will be assumed that no rationalization is needed and highlighting and this note will be removed.

Discovery is the process whereby newly connected or off-line CNUs are provided access to the CCDN. The process is driven by the CLT, which periodically makes available Discovery Windows during which off-line CNUs are given the opportunity to make themselves known to the CLT. The periodicity of these windows is unspecified and left up to the implementor. The CLT signifies that a discovery period is occurring by broadcasting a discovery GATE MPCPDU, which includes the starting time and length of the discovery window. With the appropriate settings of individual flags contained in this 16 bit wide field, the CLT notifies all the CNUs about its upstream and downstream channel transmission capabilities. Note that the CLT may simultaneously support more than one data rate in the given transmission direction.

Off-line CNUs, upon receiving a Discovery GATE MPCPDU, wait for the period to begin and then transmit a REGISTER_REQ MPCPDU to the CLT. Discovery windows are unique in that they are the only times when multiple CNUs can access the CCDN simultaneously, and transmission overlap can occur. In order to reduce transmission overlaps, a contention algorithm is used by all CNUs. Measures are taken to reduce the probability for overlaps by artificially simulating a random distribution of distances from the CLT. Each CNU shall wait a random amount of time before transmitting the REGISTER_REQ MPCPDU that is shorter than the length of the discovery window. It should be noted that multiple valid REGISTER_REQ MPCPDUs can be received by the CLT during a single discovery window. Included in the REGISTER_REQ MPCPDU is the CNU's MAC address and number of maximum pending grants.

Note: CNUs that have not completed PHY Discovery process (see 102.4.1) will not respond to Discovery GATE MPCPDUs.

In order to assure maximum utilization of the upstream channel and to decrease the required size of the guard band between individual data bursts, the registering CNU notifies the CLT of the RF on/off times, by setting appropriate values in the RF On Time and RF Off Time fields, where both values are expressed in the units of time_quanta.

Upon receipt of a valid REGISTER_REQ MPCPDU, the CLT registers the CNU, allocating and assigning a new port identity (LLID), and bonding a corresponding MAC to the LLID.

The next step in the process is for the CLT to transmit a REGISTER MPCPDU to the newly discovered CNU, which contains the CNU's LLID, and the CLT's required synchronization time. Moreover, the CLT echoes the maximum number of pending grants. The CLT also sends the target value of RF on time and RF off time, which may be different than RF on time and RF off time delivered by the CNU in the REGISTER_REQ MPCPDU.

The CLT now has enough information to schedule the CNU for access to the CCDN and transmits a standard GATE message allowing the CNU to transmit a REGISTER_ACK. Upon receipt of the REGISTER_ACK, the discovery process for that CNU is complete, the CNU is registered and normal message traffic can begin. It is the responsibility of Layer Management to perform the MAC bonding, and start transmission from/to the newly registered CNU. The discovery message exchange is illustrated in Figure 103–15.

There may exist situations when the CLT requires that a CNU go through the discovery sequence again and reregister. Similarly, there may be situations where a CNU needs to inform the CLT of its desire to deregister. The CNU can then reregister by going through the discovery sequence. For the CLT, the REGISTER message may indicate a value, Reregister or Deregister, that if either is specified forces the receiving CNU into reregistering. For the CNU, the REGISTER_REQ message contains the Deregister bit that signifies to the CLT that this CNU should be deregistered.

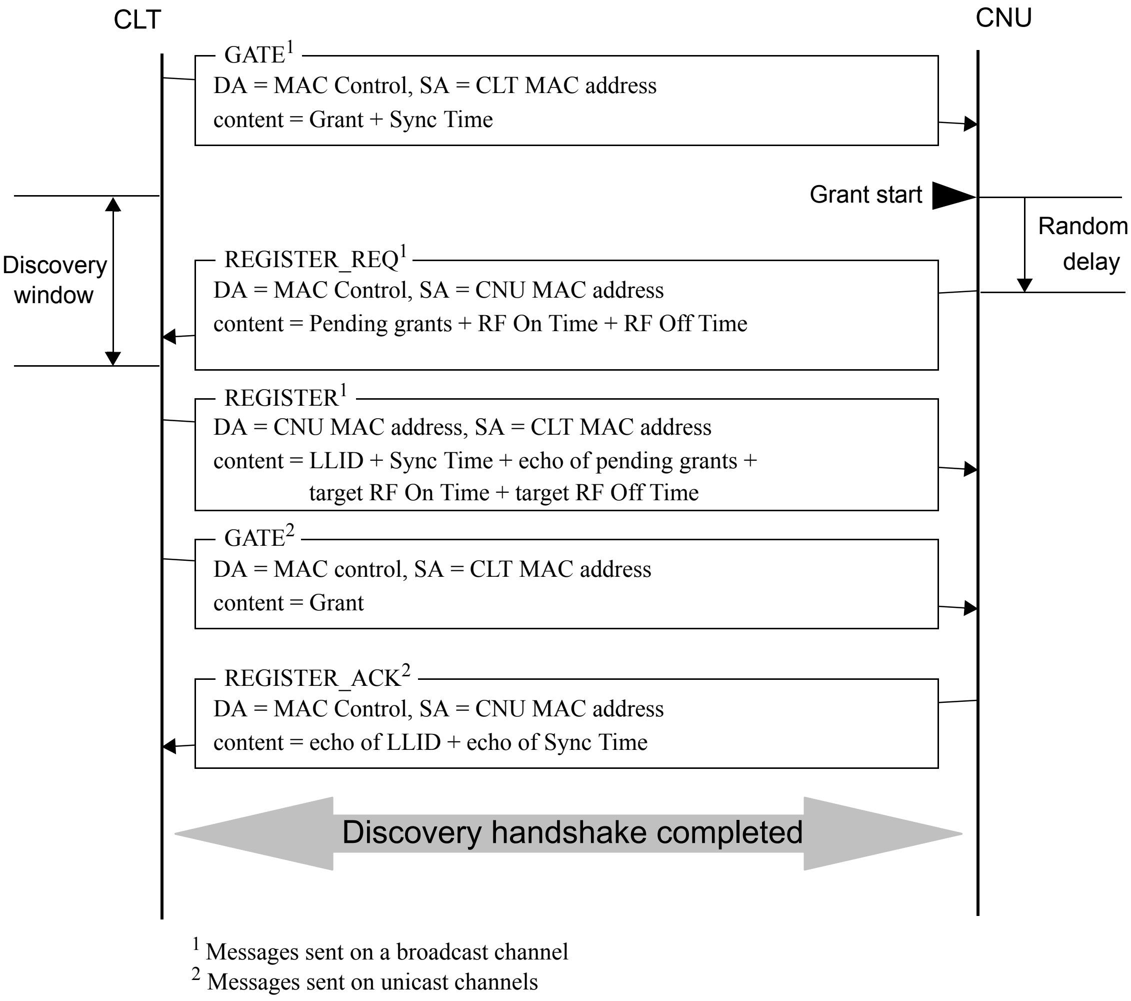

CLT
CNU
GATE[1]
DA = MAC Control, SA = CLT MAC address
content = Grant + Sync Time
Grant start
Random delay
Discovery window
REGISTER_REQ[1]
DA = MAC Control, SA = CNU MAC address
content = Pending grants + RF On Time + RF Off Time
REGISTER[1]
DA = CNU MAC address, SA = CLT MAC address
content = LLID + Sync Time + echo of pending grants + target RF On Time + target RF Off Time
GATE[2]
DA = MAC control, SA = CLT MAC address
content = Grant
REGISTER_ACK[2]
DA = MAC Control, SA = CNU MAC address
content = echo of LLID + echo of Sync Time
Discovery handshake completed
[1] Messages sent on a broadcast channel
[2] Messages sent on unicast channels

Figure 103–15—Discovery handshake message exchange

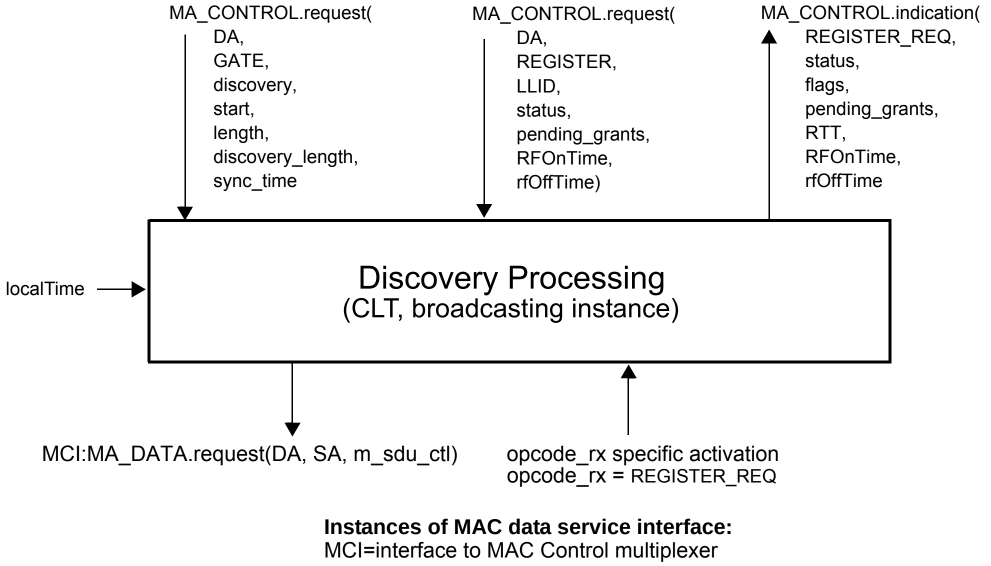


Figure 103–16—Discovery Processing service interfaces (CLT, broadcast instance)

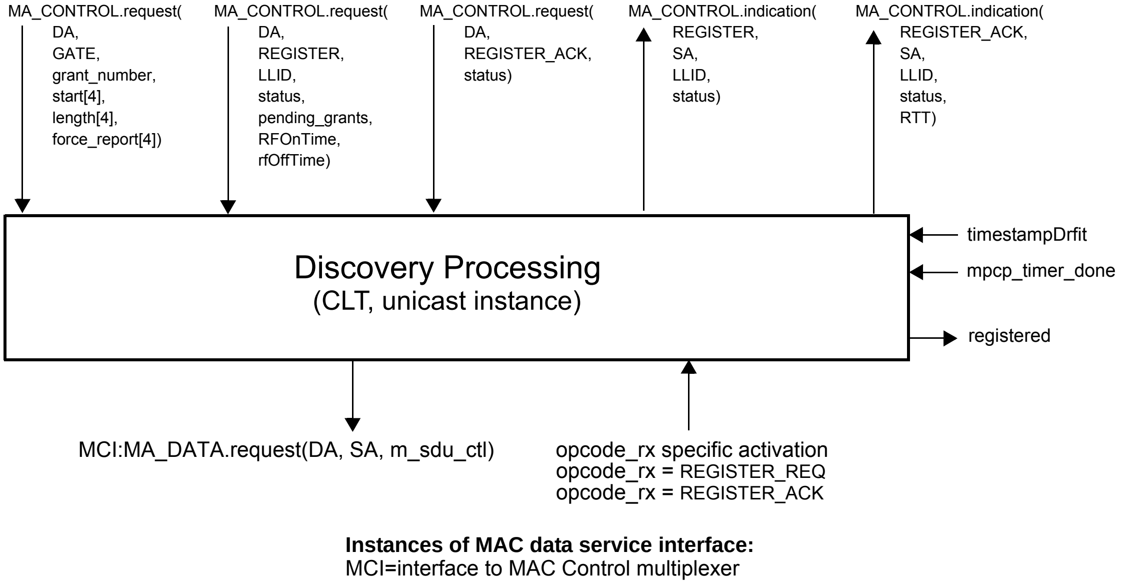


Figure 103–17—Discovery Processing service interfaces (CLT, unicast instance)

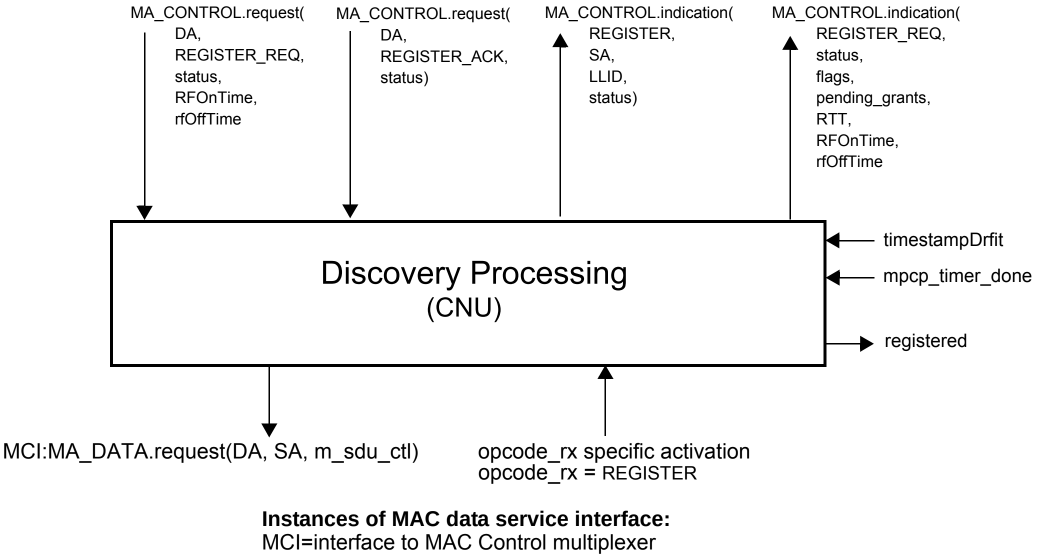


Figure 103–18—Discovery Processing service interfaces (CNU)

103.3.3.1 Constants

rfOffTimeCapability

TYPE: 8 bit unsigned

This constant represents the time required to terminate the RF, in units of time_quantum. While the default value corresponds to a maximum allowed Toff (as specified in Table TBD), implementations may set it to the actual value time period required for turning off the PMD, as specified in 100.x.y.

VALUE: 0x20 (512 ns, default value)

rfOnTimeCapability

TYPE: 8 bit unsigned

This constant represents the time required to initialize the RF, in units of time_quantum. While the default value corresponds to a maximum allowed Ton (as specified in Table TBD), implementations may set it to the actual value time period required for turning on the PMD, as specified in 100.x.y.

VALUE: 0x20 (512 ns, default value)

103.3.3.2 Variables

BEGIN

This variable is defined in 103.2.2.3.

data_rx

This variable is defined in 64.2.2.3.

data_tx

This variable is defined in 64.2.2.3.

grantEndTime

TYPE: 32 bit unsigned

This variable holds the time at which the CLT expects the CNU grant to complete. Failure of a

REGISTER_ACK message from a CNU to arrive at the CLT before *grantEndTime* is a fatal error in the discovery process, and causes registration to fail for the specified CNU, who may then retry to register. The value of *grantEndTime* is measured in units of time_quantum.

insideDiscoveryWindow

TYPE: Boolean

This variable holds the current status of the discovery window and is defined in 64.3.3.2.

rfOffTime

TYPE: 8 bit unsigned

This variable holds the time required to terminate the RF. It counts in time_quanta units the time period required for turning off the PMD, as specified by the value of Toff in 100.x.y.

VALUE: *rfOffTime*Capability (default value)

rfOnTime

TYPE: 8 bit unsigned

This variable holds the time required to initiate the PMD. It counts in time_quanta units the time period required for turning on the PMD, as specified by the value of Ton in 100.x.y.

VALUE: *rfOnTime*Capability (default value)

localTime

This variable is defined in 64.2.2.2.

m_sdu_ctl

This variable is defined in 77.2.2.3.

opcode_rx

This variable is defined in 64.2.2.3.

pendingGrants

TYPE: 16 bit unsigned

This variable holds the maximum number of pending grants that a CNU is able to queue.

registered

TYPE: Boolean

This variable holds the current result of the Discovery Process and is defined in 64.3.3.2.

syncTime

TYPE: 16 bit unsigned

This variable holds the time required to stabilize the receiver at the CLT. It counts time_quanta units from the point where transmission output is stable to the point where synchronization has been achieved. The value of *syncTime* includes gain adjustment interval (Treceiver_settling), clock synchronization interval (Tcdr), and code-group alignment interval (Tcode_group_align), as specified in 100.x.y. The CLT conveys the value of *syncTime* to CNUs in Discovery GATE and REGISTER messages. During the synchronization time a CNU sends synchronization pattern (SP, see Y.3.2.5.2) followed by burst delimiter pattern (BURST_DELIMITER, see Y.3.2.5.2).

EDITORS NOTE (to be removed prior to publication): the Task Force needs to agree that this definition of syncTime is acceptable or modify this. The last sentence will need modification such as "During the synchronization time a CNU sends preamble (SP, see Y.3.2.5.2). Immediately after the preamble the CNU transmits Start of Burst delimiter pattern (BURST_DELIMITER, see Y.3.2.5.2)."

timestampDrift

This variable is defined in 64.2.2.3.

103.3.3.3 Functions

None.

103.3.3.4 Timers

discovery_window_size_timer

This timer is used to wait for the event signaling the end of the discovery window and is defined in 64.2.2.4.

mpcp_timer

This timer is used to measure the arrival rate of MPCP frames in the link and is defined in 64.2.2.4.

103.3.3.5 Messages

MA_DATA.indication(DA, SA, m_sdu, receiveStatus)

This service primitive is defined in 2.3.2.

MA_DATA.request (DA, SA, m_sdu)

This service primitive is defined in 2.3.2.

MA_CONTROL.request(DA, GATE, discovery, start, length, discovery_length, sync_time)

This service primitive is used by the MAC Control client at the CLT to initiate the Discovery Process. This primitive takes the following parameters:

DA: Multicast or unicast MAC address.
GATE: Opcode for GATE MPCPDU as defined in Table 31A-1.
discovery: Flag specifying that the given GATE message is to be used for discovery only.
start: Start time of the discovery window.
length: Length of the grant given for discovery.
discovery_length: Length of the discovery window process.
sync_time: The time interval required to stabilize the receiver at the CLT.

MA_CONTROL.request(DA, GATE, grant_number, start[4], length[4], force_report[4])

This service primitive is used by the MAC Control client at the CLT to issue the GATE message to a CNU. This primitive takes the following parameters:

DA: Multicast MAC Control address as defined in Annex 31B.
GATE: Opcode for GATE MPCPDU as defined in Table 31A-1.
grant_number: Number of grants issued with this GATE message. The number of grants ranges from 0 to 4.
start[4]: Start times of the individual grants. Only the first grant_number elements of the array are used.
length[4]: Lengths of the individual grants. Only the first grant_number elements of the array are used.
force_report[4]: Flags indicating whether a REPORT message should be generated in the corresponding grant. Only the first grant_number elements of the array are used.

MA_CONTROL.request(DA,REGISTER_REQ,status,*rfOnTime*,*rfOffTime*)

This service primitive is used by a client at the CNU to request the Discovery Process to perform a registration. This primitive takes the following parameters:

DA: Multicast MAC Control address as defined in Annex 31B.
REGISTER_REQ: opcode for REGISTER_REQ MPCPDU as defined in Table 31A-1.
status: This parameter takes on the indication supplied by the flags field in the REGISTER_REQ MPCPDU as defined in Table 103–3.
rfOnTime: This parameter holds the *rfOnTime* value, expressed in units of time_quanta, as reported by MAC client and specified in 103.3.6.3.
rfOffTime: This parameter holds the *rfOffTime* value, expressed in units of time_quanta, as reported by MAC client and specified in 103.3.6.3.

MA_CONTROL.indication(REGISTER_REQ, status, flags, pending_grants, *RTT*, *rfOnTime*, *rfOffTime*)

The service primitive is issued by the Discovery Process to notify the client and Layer Management that the registration process is in progress. This primitive takes the following parameters:

REGISTER_REQ: Opcode for REGISTER_REQ MPCPDU as defined in Table 31A-1.

status: This parameter holds the values incoming or retry. Value incoming is used at the CLT to signal that a REGISTER_REQ message was received successfully. The value retry is used at the CNU to signal to the client that a registration attempt failed and needs to be repeated.

flags: This parameter holds the contents of the flags field in the REGISTER_REQ message. This parameter holds a valid value only when the primitive is generated by the Discovery Process in the CLT.

pending_grants: This parameter holds the contents of the pending_grants field in the REGISTER_REQ message. This parameter holds a valid value only when the primitive is generated by the Discovery Process in the CLT.

RTT: The measured round trip time to/from the CNU is returned in this parameter. *RTT* is stated in time_quanta units. This parameter holds a valid value only when the primitive is generated by the Discovery Process in the CLT.

rfOnTime: This parameter holds the contents of the *rfOnTime* field in the REGISTER_REQ message. This parameter holds a valid value only when the primitive is generated by the Discovery Process in the CLT.

rfOffTime: This parameter holds the contents of the *rfOffTime* field in the REGISTER_REQ message. This parameter holds a valid value only when the primitive is generated by the Discovery Process in the CLT.

MA_CONTROL.request(DA, REGISTER, LLID, status, pending_grants, *rfOnTime*, *rfOffTime*)

The service primitive is used by the MAC Control client at the CLT to initiate acceptance of a CNU. This primitive takes the following parameters:

DA: Unicast MAC address or multicast MAC Control address as defined in Annex 31B.

REGISTER: Opcode for REGISTER MPCPDU as defined in Table 31A-1.

LLID: This parameter holds the logical link identification number assigned by the MAC Control client.

status: This parameter takes on the indication supplied by the flags field in the REGISTER MPCPDU as defined in Table 103–4.

pending_grants: This parameters echoes back the pending_grants field that was previously received in the REGISTER_REQ message.

rfOnTime: This parameter carries the target value of RF On Time for the given CNU transmitter. This value may be different than the *rfOnTime* value carried in the REGISTER_REQ MPCPDU received from the corresponding CNU MAC during Discovery stage.

rfOffTime: This parameter carries the target value of RF Off Time for the given CNU transmitter. This value may be different than the *rfOffTime* value carried in the REGISTER_REQ MPCPDU received from the corresponding CNU MAC during Discovery stage.

MA_CONTROL.indication(REGISTER, SA, LLID, status)

This service primitive is issued by the Discovery Process at the CLT or a CNU to notify the MAC Control client and Layer Management of the result of the change in registration status. This primitive takes the following parameters:

REGISTER: Opcode for REGISTER MPCPDU as defined in Table 31A-1.

SA: This parameter represents the MAC address of the CLT.

LLID: This parameter holds the logical link identification number assigned by the MAC Control client.

status: This parameter holds the value of accepted / denied / deregistered / reregistered.

MA_CONTROL.request(DA, REGISTER_ACK, status)

This service primitive is issued by the MAC Control clients at the CNU and the CLT to acknowledge the registration. This primitive takes the following parameters:

DA: Multicast MAC Control address as defined in Annex 31B.

REGISTER_ACK: Opcode for REGISTER_ACK MPCPDU as defined in Table 31A-1.

status: This parameter takes on the indication supplied by the flags field in the REGISTER MPCPDU as defined in Table 103–5.

MA_CONTROL.indication(REGISTER_ACK, SA, LLID, status, *RTT*)

This service primitive is issued by the Discovery Process at the CLT to notify the client and Layer Management that the registration process has completed. This primitive takes the following parameters:

REGISTER_ACK: Opcode for REGISTER_ACK MPCPDU as defined in Table 31A-1.

SA: This parameter represents the MAC address of the reciprocating device (CNU address at the CLT, and CLT address at the CNU).

LLID: This parameter holds the logical link identification number assigned by the MAC Control client.

status: This parameter holds the value of accepted/denied/reset/deregistered.

RTT: The measured round trip time to/from the CNU is returned in this parameter. *RTT* is stated in time_quanta units. This parameter holds a valid value only when the invoking Discovery Process in the CLT.

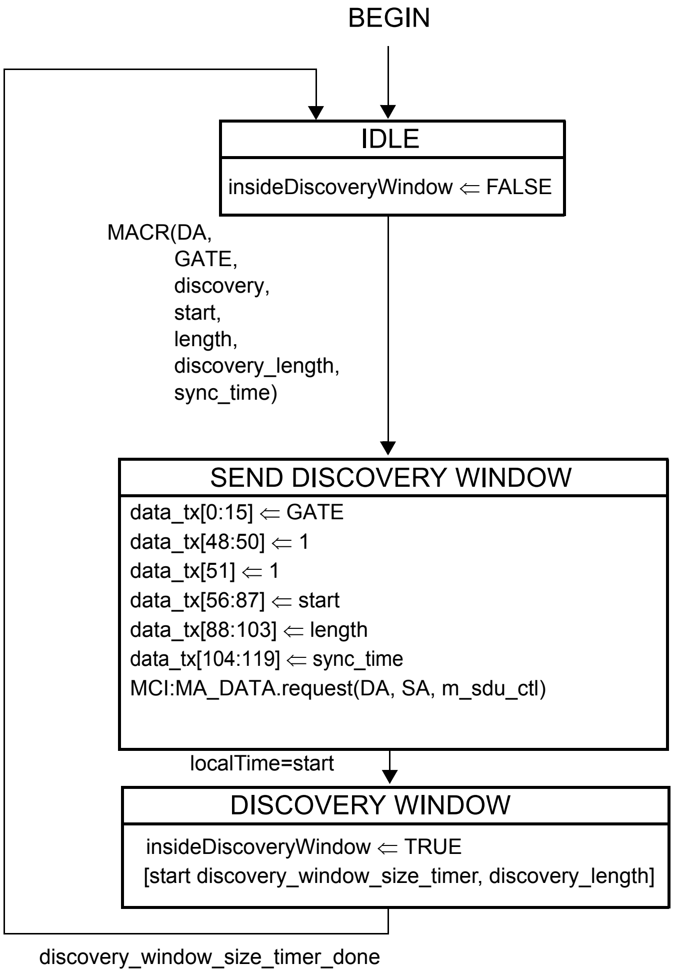
Opcode-specific function(opcode)

Functions exported from opcode specific blocks that are invoked on the arrival of a MAC Control message of the appropriate opcode.

103.3.3.6 State Diagrams

The Discovery Process in the CLT shall implement the discovery window setup state diagram shown in Table 103–19, request processing state diagram as shown in Table 103–20, register processing state diagram as shown in Table 103–21, and final registration state diagram as shown in Table 103–22. The discovery process in the CNU shall implement the registration state diagram as shown in Table 103–23.

Instantiation of state diagrams as described in Table 103–19, Table 103–20, and Table 103–21 is performed only at the Multipoint MAC Control instances attached to the broadcast LLID (0x7FFE). Instantiation of state diagrams as described in Table 103–22 and Table 103–23 is performed for every Multipoint MAC Control instance, except the instance attached to the broadcast channel.



Instances of MAC data service interface:
MCI=interface to MAC Control multiplexer

Figure 103–19—Discovery Processing CLT Window Setup state diagram

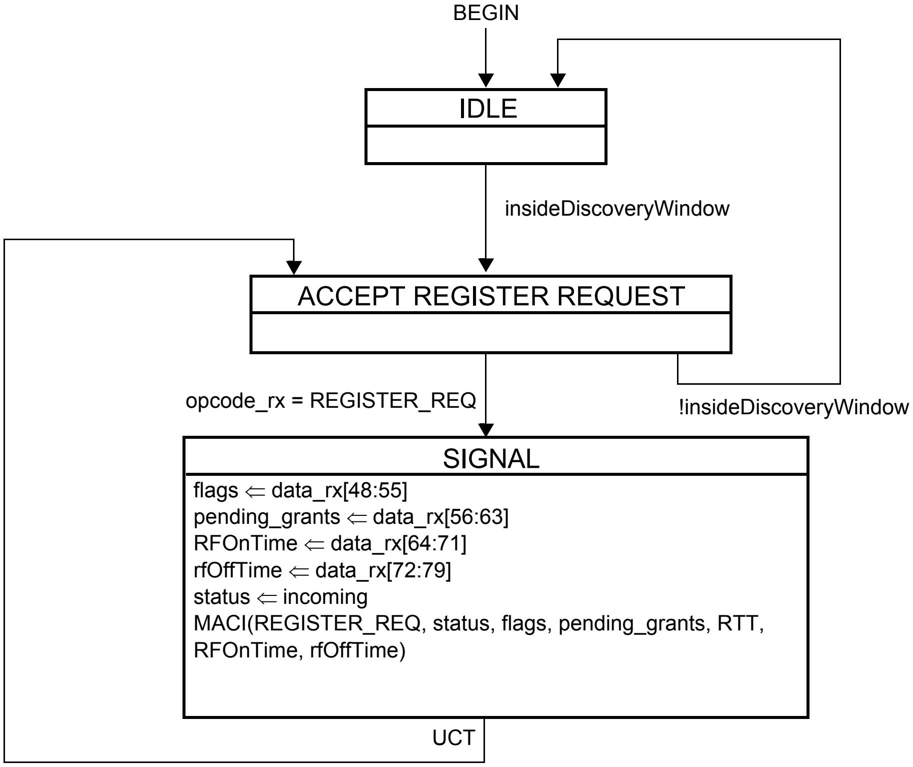


Figure 103–20—Discovery Processing CLT Process Requests state diagram

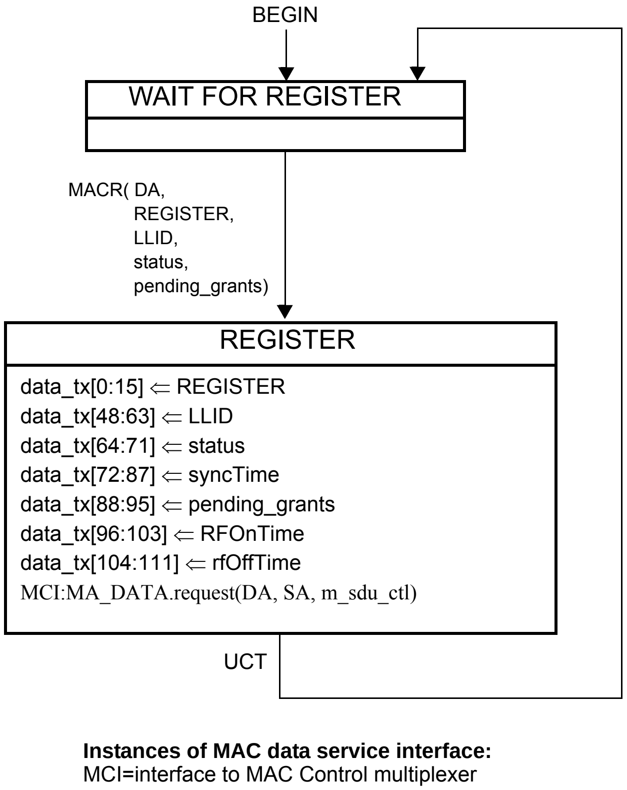
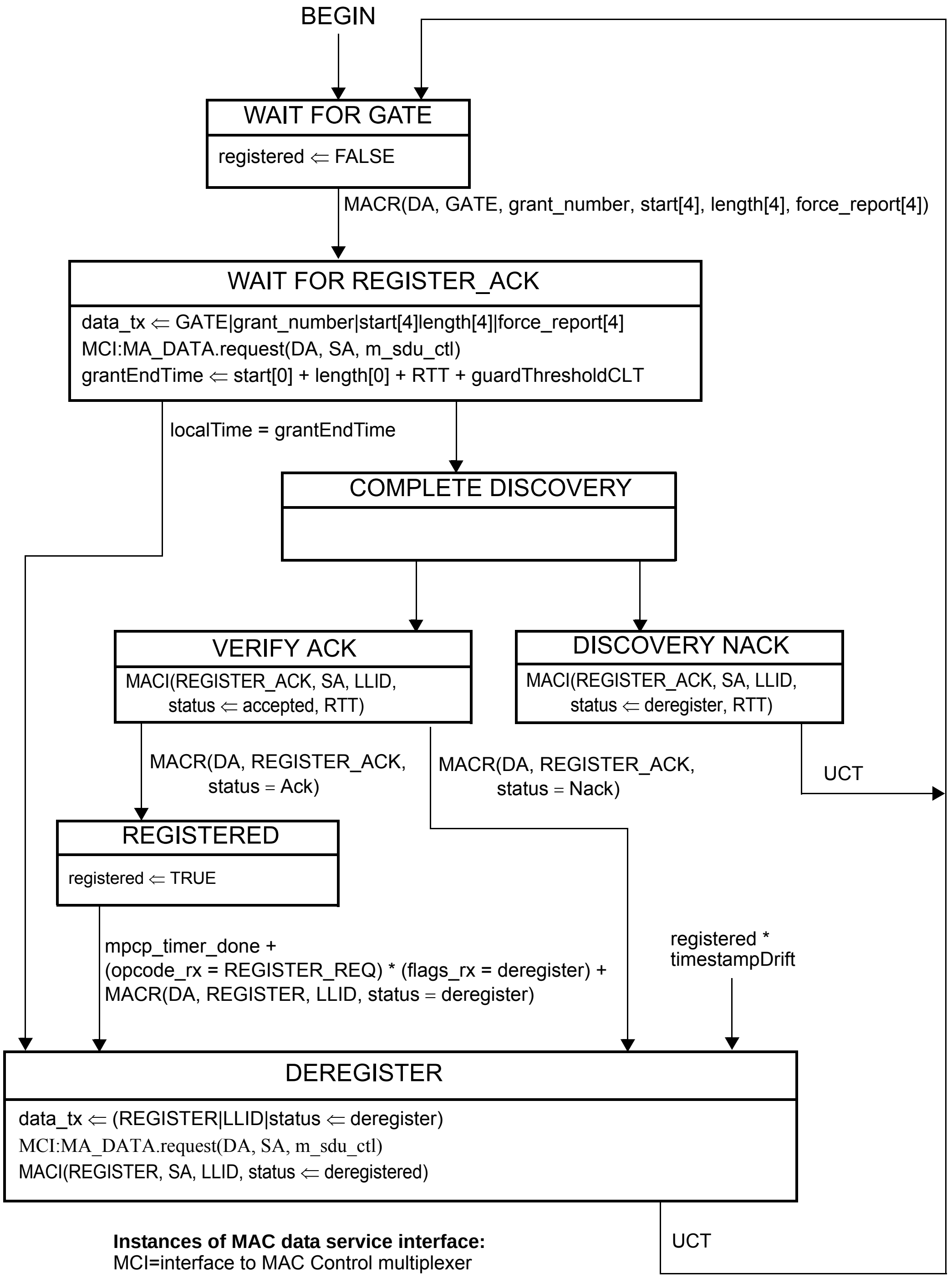

BEGIN
WAIT FOR REGISTER
MACR(DA,
REGISTER,
LLID,
status,
pending_grants)
REGISTER
data_tx[0:15] ⇐ REGISTER
data_tx[48:63] ⇐ LLID
data_tx[64:71] ⇐ status
data_tx[72:87] ⇐ syncTime
data_tx[88:95] ⇐ pending_grants
data_tx[96:103] ⇐ RFOnTime
data_tx[104:111] ⇐ rfOffTime
MCI:MA_DATA.request(DA, SA, m_sdu_ctl)
UCT
Instances of MAC data service interface:
MCI=interface to MAC Control multiplexer

Figure 103–21—Discovery Processing CLT Register state diagram


BEGIN
WAIT FOR GATE
registered ⇐ FALSE
MACR(DA, GATE, grant_number, start[4], length[4], force_report[4])
WAIT FOR REGISTER_ACK
data_tx ⇐ GATE|grant_number|start[4]length[4]|force_report[4]
MCI:MA_DATA.request(DA, SA, m_sdu_ctl)
grantEndTime ⇐ start[0] + length[0] + RTT + guardThresholdCLT
localTime = grantEndTime
COMPLETE DISCOVERY
VERIFY ACK
MACI(REGISTER_ACK, SA, LLID, status ⇐ accepted, RTT)
DISCOVERY NACK
MACI(REGISTER_ACK, SA, LLID, status ⇐ deregister, RTT)
MACR(DA, REGISTER_ACK, status = Ack)
MACR(DA, REGISTER_ACK, status = Nack)
UCT
REGISTERED
registered ⇐ TRUE
mpcp_timer_done +
(opcode_rx = REGISTER_REQ) * (flags_rx = deregister) +
MACR(DA, REGISTER, LLID, status = deregister)
registered *
timestampDrift
DEREGISTER
data_tx ⇐ (REGISTER|LLID|status ⇐ deregister)
MCI:MA_DATA.request(DA, SA, m_sdu_ctl)
MACI(REGISTER, SA, LLID, status ⇐ deregistered)
UCT
Instances of MAC data service interface:
MCI=interface to MAC Control multiplexer

NOTE—The MAC Control Client issues the grant following the REGISTER message, taking the CNU processing delay of REGISTER message into consideration.

Figure 103–22—Discovery Processing CLT Final Registration state diagram

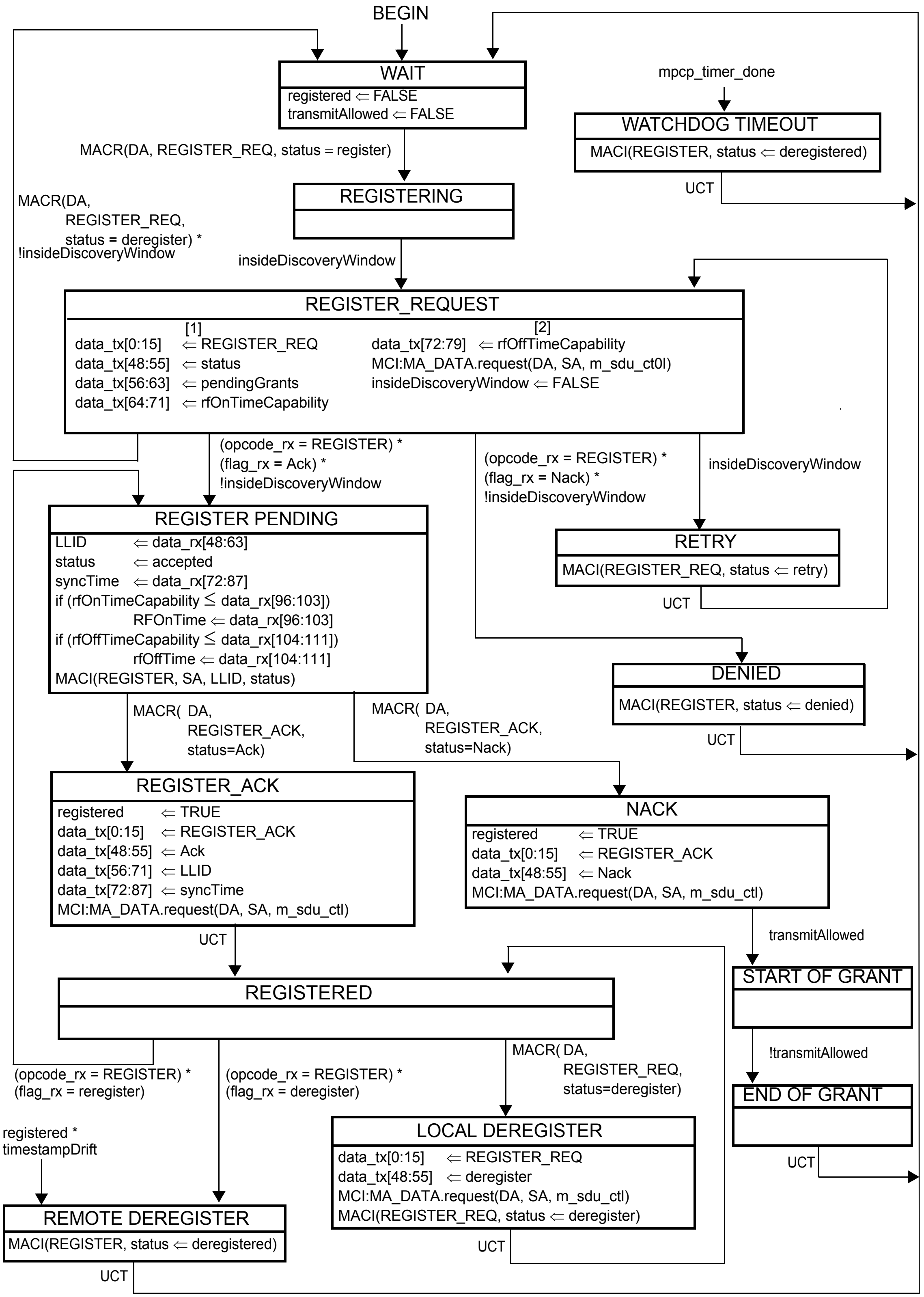


Figure 103–23—Discovery Processing CNU Registration state diagram

103.3.4 Report Processing

The Report Processing functional block has the responsibility of dealing with queue report generation and termination in the network. Reports are generated by higher layers and passed to the MAC Control sublayer by the MAC Control clients. Status reports are used to signal bandwidth needs as well as for arming the CLT watchdog timer.

Reports shall be generated periodically, even when no request for bandwidth is being made. This keeps a watchdog timer in the CLT from expiring and deregistering the CNU. For proper operation of this mechanism the CLT shall grant the CNU periodically.

The Report Processing functional block, and its MPCP protocol elements are designed for use in conjunction with an IEEE 802.1P capable bridge.

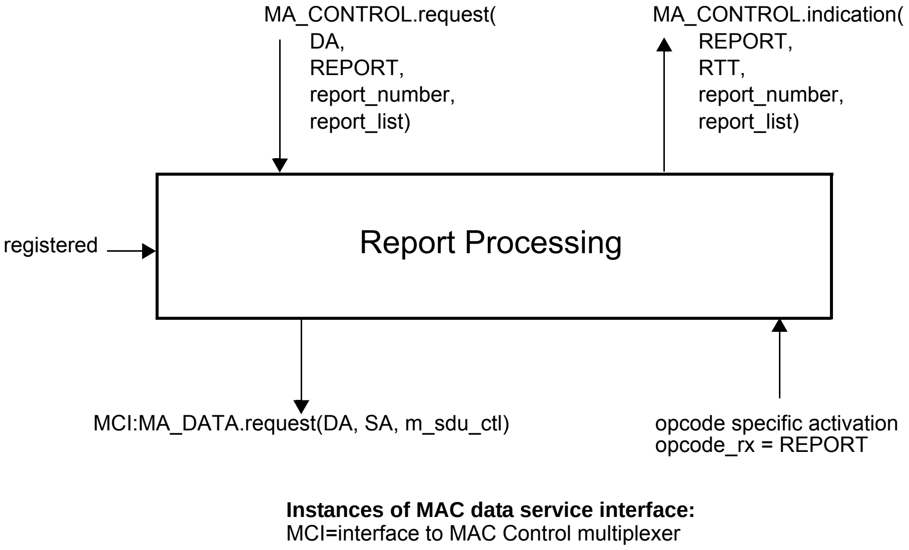


Figure 103–24—Report Processing service interfaces

103.3.4.1 Constants

None.

103.3.4.2 Variables

BEGIN

TYPE: Boolean
This variable is used when initiating operation of the functional block state diagram. It is set to TRUE following initialization and every reset.

data_rx

This variable is defined in 103.2.2.3.

data_tx

This variable is defined in 103.2.2.3.

m_sdu_ctl

This variable is defined in 103.2.2.3.

mpcp_timeout

TYPE: 32 bit unsigned

This variable represents the maximum allowed interval of time between two MPCPDU messages. Failure to receive at least one frame within this interval is considered a fatal fault and leads to deregistration. This variable is expressed in units of time_quanta.

VALUE: 0x03B9ACA0 (1 s, default value)

opcode_rx

This variable is defined in 103.2.2.3.

registered

This variable is defined in 103.3.3.2.

report_timeout

TYPE: 32 bit unsigned

This variable represents the maximum allowed interval of time between two REPORT messages generated by the CNU, expressed in units of time_quanta.

VALUE: 0x002FAF08 (50 ms, default value)

103.3.4.3 Functions

None.

103.3.4.4 Timers

report_periodic_timer

CNUs are required to generate REPORT MPCPDUs with a periodicity of less than *report_timeout* value. This timer counts down time remaining before a forced generation of a REPORT message in a CNU.

mpcp_timer

This timer is defined in 103.3.3.4.

103.3.4.5 Messages

MA_DATA.request (DA, SA, m_sdu)

The service primitive is defined in 2.3.2.

MA_CONTROL.request(DA, REPORT, report_number, report_list)

This service primitive is used by a MAC Control client to request the Report Process at the CNU to transmit a queue status report. This primitive may be called at variable intervals, independently of the granting process, in order to reflect the time varying aspect of the network. This primitive uses the following parameters:

DA:	Multicast MAC Control address as defined in Annex 31B.
REPORT:	Opcode for REPORT MPCPDU as defined in Table 31A-1.
report_number:	The number of queue status report sets located in report list. The report_number value ranges from 0 to a maximum of 13.
report_list:	The list of queue status reports. A queue status report consists of two fields: valid and status. The parameter valid is a Boolean array of length of 8. The index of an element of this array reflects the numbered priority queue in the IEEE 802.1P nomenclature. An element with the value of '0' or FALSE indicates that the corresponding status field is not present (the length of status field is 0),while '1' or TRUE indicates that the corresponding status field is present (the length of status field is 2 octets).

The parameter status is an array of 16 bit unsigned integer values. This array consists only of entries whose corresponding bit in field valid is set to TRUE.

MA_CONTROL.indication(REPORT, *RTT*, report_number, report_list)

The service primitive is issued by the Report Process at the CLT to notify the MAC Control client and higher layers the queue status of the MPCP link partner. This primitive may be called multiple times, in order to reflect the time-varying aspect of the network. This primitive uses the following parameters:

REPORT: Opcode for REPORT MPCPDU as defined in Table 31A-1.

RTT: This parameter holds an updated round trip time value that is recalculated following each REPORT message reception.

report_number: The number of queue status report sets located in report list. The report_number value ranges from 0 to a maximum of 13.

report_list: The list of queue status reports. A queue status report consists of two fields: valid and status. The parameter valid is a Boolean array of length of 8. The index of an element of this array reflects the numbered priority queue in the IEEE 802.1P nomenclature. An element with the value of '0' or FALSE indicates that the corresponding status field is not present (the length of status field is 0),while '1' or TRUE indicates that the corresponding status field is present (the length of status field is 2 octets). The parameter status is an array of 16 bit unsigned integer values. This array consists only of entries whose corresponding bit in field valid is set to TRUE.

Opcode-specific function(opcode)

Functions exported from opcode specific blocks that are invoked on the arrival of a MAC Control message of the appropriate opcode.

103.3.4.6 State diagrams

The report process in the CLT shall implement the report processing state diagram as shown in Figure 103–25. The report process in the CNU shall implement the report processing state diagram as shown in Figure 103–26. Instantiation of state diagrams as described is performed for Multipoint MAC Control instances attached to unicast LLIDs only.

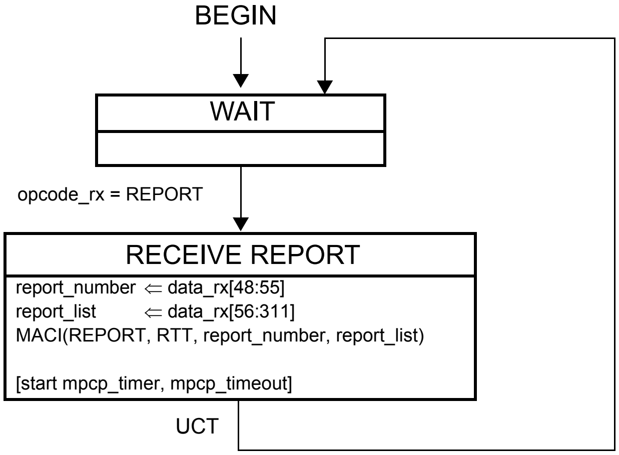

BEGIN
WAIT
opcode_rx = REPORT
RECEIVE REPORT
report_number ⇐ data_rx[48:55]
report_list ⇐ data_rx[56:311]
MACI(REPORT, RTT, report_number, report_list)
[start mpcp_timer, mpcp_timeout]
UCT

Figure 103–25—Report Processing state diagram at CLT

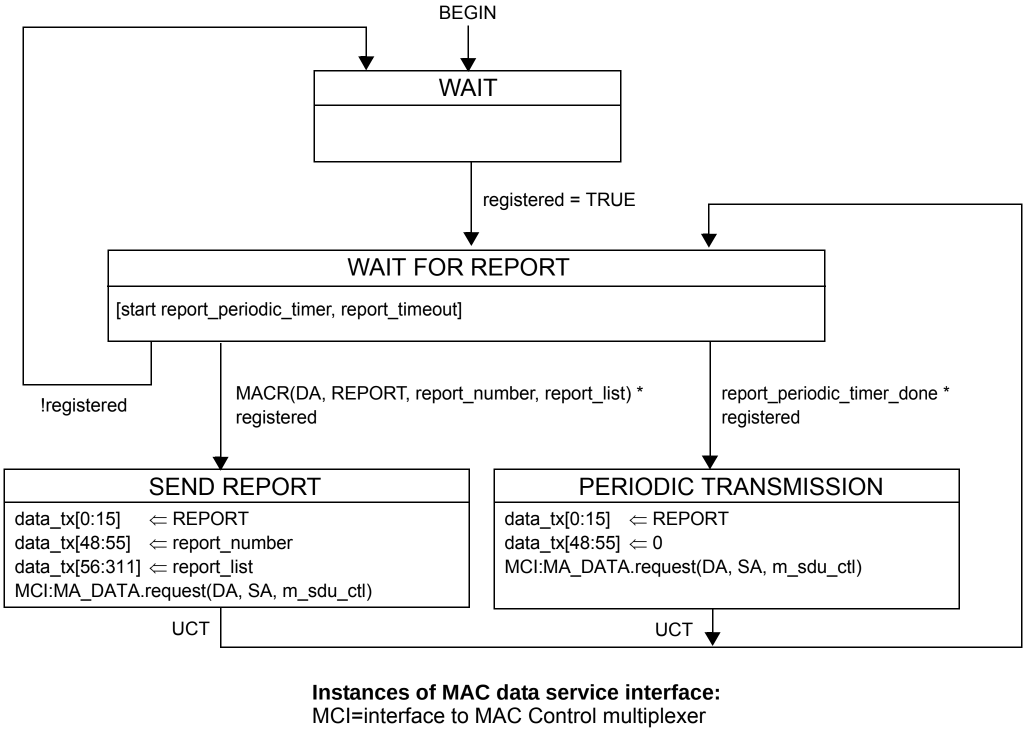

BEGIN
WAIT
registered = TRUE
WAIT FOR REPORT
[start report_periodic_timer, report_timeout]
!registered
MACR(DA, REPORT, report_number, report_list) * registered
report_periodic_timer_done * registered
SEND REPORT
data_tx[0:15] ⇐ REPORT
data_tx[48:55] ⇐ report_number
data_tx[56:311] ⇐ report_list
MCI:MA_DATA.request(DA, SA, m_sdu_ctl)
PERIODIC TRANSMISSION
data_tx[0:15] ⇐ REPORT
data_tx[48:55] ⇐ 0
MCI:MA_DATA.request(DA, SA, m_sdu_ctl)
UCT
UCT
Instances of MAC data service interface:
MCI=interface to MAC Control multiplexer

Figure 103–26—Report Processing state diagram at CNU

103.3.5 Gate Processing

A key concept pervasive in Multipoint MAC Control is the ability to arbitrate a single transmitter out of a plurality of CNUs. The CLT controls a CNU’s transmission by the assigning of grants.

The transmitting window of a CNU is indicated in the GATE message where start time and length are specified. A CNU begins transmission when its localTime counter matches the start_time value indicated in the GATE message. A CNU concludes its transmission with sufficient margin to ensure that the RF transmitter is turned off before the grant length interval has elapsed.

Multiple outstanding grants may be issued to each CNU. The CLT shall not issue more than the maximum supported maximum outstanding grants as advertised by the CNU during registration (see pending grants in 103.3.6.3).

In order to maintain the watchdog timer at the CNU, grants are periodically generated. For this purpose empty GATE messages may be issued periodically.

When registered, the CNU ignores all gate messages where the Discovery flag is set.

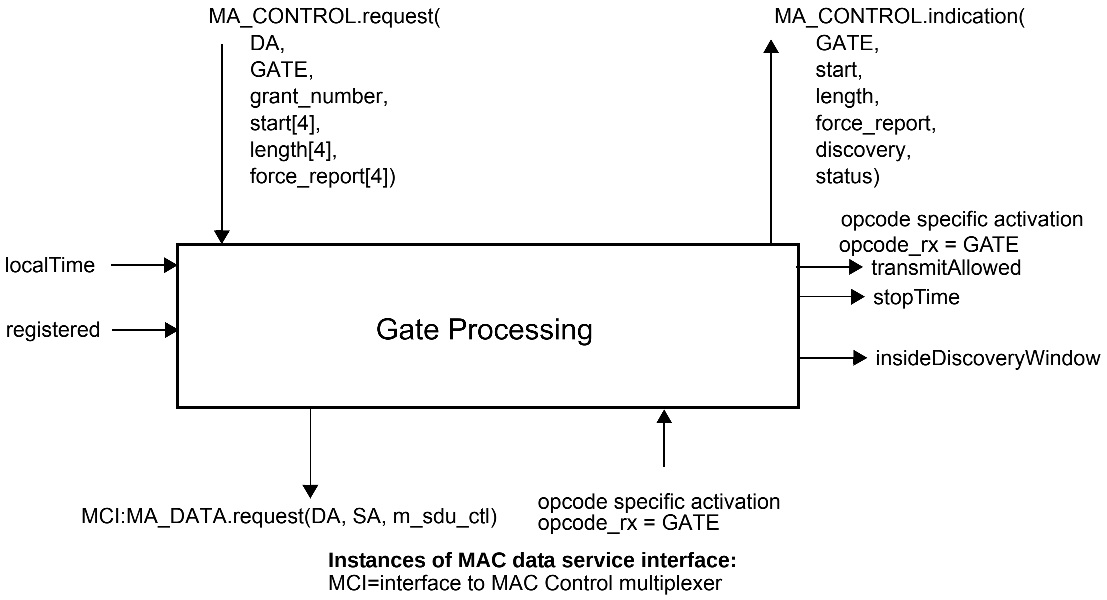


Figure 103–27—Gate Processing service interface

103.3.5.1 Constants

max_future_grant_time

TYPE: 32 bit unsigned

This constant holds the time limiting the future time horizon for a valid incoming grant.

VALUE: 0x03B9ACA0 (1 s)

min_processing_time

TYPE: 32 bit unsigned

This constant is the time required for the CNU processing time.

VALUE: TBD

minGrantLength

TYPE: 32 bit unsigned

This constant represents the minimum data portion of a grant. The minGrantLength is equal to one FEC codeword (see DS_FEC_CW_Sz in 103.2.2.1), less the initial 16 idle octets, expressed in units of time_quanta. The minimum grant length accepted by a CNU is equal to minGrantLength + *BurstOverhead* (see 103.3.5.2).

VALUE: TBD

tqSizeC

This constant is defined in 103.2.2.1.

103.3.5.2 Variables

BEGIN

TYPE: Boolean

This variable is used when initiating operation of the functional block state diagram. It is set to TRUE following initialization and every reset.

BurstOverhead

TYPE: integer

This variable represents the burst overhead and equals the sum of *rfOnTime*, *rfOffTime*, *syncTime* and an additional {TBD} time_quanta to account for END_BURST_DELIMITER and two leading IDLE vectors of the payload. This variable is expressed in units of time_quanta.

counter

TYPE: integer

This variable is used as a loop iterator counting the number of incoming grants in a GATE message.

currentGrant

TYPE:

```
structure
{
    DA:             48 bit unsigned, a.k.a MAC address type
    start           32 bit unsigned
    length          16 bit unsigned
    force_report    Boolean
    discovery       Boolean
}
```

This variable is used for local storage of a pending grant state during processing. It is dynamically set by the Gate Processing functional block and is not exposed.
The state is a structure field composed of multiple subfields.

data_rx

This variable is defined in 103.2.2.3.

data_tx

This variable is defined in 103.2.2.3.

effectiveLength

TYPE: 32 bit unsigned

This variable is used for temporary storage of a normalized net time value. It holds the net effective length of a grant normalized for elapsed time, and compensated for the periods required to turn the RF on and off, and waiting for receiver lock.

gate_timeout

TYPE: 32 bit unsigned

This variable represents the maximum allowed interval of time between two GATE messages generated by the CLT to the same CNU, expressed in units of time_quanta.
VALUE: 0x002FAF08 (50 ms, default value)

grantList

TYPE: list of elements having the structure define in *currentGrant*

This variable is used for storage of the list of pending grants. It is dynamically set by the Gate Processing functional block and is not exposed. Each time a grant is received it is added to the list. The list elements are structure fields composed of multiple subfields. The list is indexed by the start subfield in each element for quick searches.

grantStart

This variable is defined in 103.2.2.3.

insideDiscoveryWindow

This variable is defined in 103.3.3.2.

maxDelay

TYPE: 16 bit unsigned

This variable holds the maximum delay that can be applied by a CNU before sending the REGISTER_REQ MPCPDU. This delay is calculated such that the CNU would have sufficient time to transmit the REGISTER_REQ message and its associated overhead (FEC parity data, end-of-frame sequence, etc.) and terminate the RF before the end of the discovery grant.

m_sdu_ctl
This variable is defined in 103.2.2.3.

nextGrant
TYPE: element having same structure as defined in *currentGrant*
This variable is used for local storage of a pending grant state during processing. It is dynamically set by the Gate Processing functional block and is not exposed. The content of the variable is the next grant to become active.

nextStopTime
TYPE: 32 bit unsigned
This variable holds the value of the localTime counter corresponding to the end of the next grant.

opcode_rx
This variable is defined in 103.2.2.3.

registered
This variable is defined in 103.3.3.2.

stopTime
This variable is defined in 103.2.2.3.

syncTime
This variable is defined in 103.3.3.2.

transmitAllowed
This variable is defined in 103.2.2.3.

103.3.5.3 Functions

empty(list)
This function is use to check whether the list is empty. When there are no elements queued in the list, the function returns TRUE. Otherwise, a value of FALSE is returned.

InsertInOrder(sorted_list, inserted_element)
This function is used to queue an element inside a sorted list. The queuing order is sorted. In the condition that the list is full the element may be discarded. The length of the list is dynamic and it's maximum size equals the value advertised during registration as maximum number of pending grants.

IsBroadcast(grant)
This function is used to check whether its argument represents a broadcast grant, i.e., grant given to multiple CNUs. This is determined by the destination MAC address of the corresponding GATE message. The function returns the value TRUE when MAC address is a global assigned MAC Control address as defined in Annex 31B, and FALSE otherwise.

PeekHead(sorted_list)
This function is used to check the content of a sorted list. It returns the element at the head of the list without dequeuing the element.

Random(r)
This function is used to compute a random integer number uniformly distributed between 0 and r. The randomly generated number is then returned by the function.

RemoveHead(sorted_list)

This function is used to dequeue an element from the head of a sorted list. The return value of the function is the dequeued element.

103.3.5.4 Timers

gntWinTmr

This timer is used to wait for the event signaling the end of a grant window.
VALUE: The timer value is dynamically set according to the signaled grant length.

gate_periodic_timer

The CLT is required to generate GATE MPCPDUs with a periodicity of less than *gate_timeout* value. This timer counts down time remaining before a forced generation of a GATE message in the CLT.

mpcp_timer

This timer is defined in 103.3.3.4.

rndDlyTmr

This timer is used to measure a random delay inside the discovery window. The purpose of the delay is to a priori reduce the probability of transmission overlap during the registration process, and thus lowering the expectancy of registration time in the CCDN.
VALUE: A random value less than the net discovery window size less the REGISTER_REQ MPCPDU frame size less the idle period and RF turn on and off delays less the preamble size less the IFG size. The timer value is set dynamically based on the parameters passed from the client.

103.3.5.5 Messages

MA_DATA.request (DA, SA, m_sdu)

The service primitive is defined in 2.3.2.

MA_CONTROL.request(DA, GATE, grant_number, start[4], length[4], force_report[4])

This service primitive is defined in 103.3.3.5.

MA_CONTROL.indication(GATE, start, length, force_report, discovery, status)

This service primitive issued by the Gate Process at the CNU to notify the MAC Control client and higher layers that a grant is pending. This primitive is invoked multiple times when a single GATE message arrives with multiple grants. It is also generated at the start and end of each grant as it becomes active. This primitive uses the following parameters:

GATE: Opcode for GATE MPCPDU as defined in Table 31A-1.

start: start time of the grant. This parameter is not present when the parameter status value is equal to deactive.

length: Length of the grant. This parameter is not present when the parameter status value is equal to deactive.

force_report: Flags indicating whether a REPORT message should be transmitted in this grant. This parameter is not present when the parameter status value is equal to deactive.

discovery: This parameter holds the value TRUE when the grant is to be used for the discovery process, and FALSE otherwise. This parameter is not present when the parameter status value is equal to deactive.

status: This parameter takes the value arrive on grant reception, active when a grant becomes active, and deactive at the end of a grant.

Opcode-specific function(opcode)

Functions exported from opcode specific blocks that are invoked on the arrival of a MAC Control message of the appropriate opcode.

103.3.5.6 State diagrams

The gating process in the CLT shall implement the Gate processing state diagram as shown in Figure 103–28. The gating process in the CNU shall implement the Gate processing state diagram as shown in Figure 103–29 and Figure 103–30. Instantiation of state diagrams as described is performed for all Multi-point MAC Control instances.

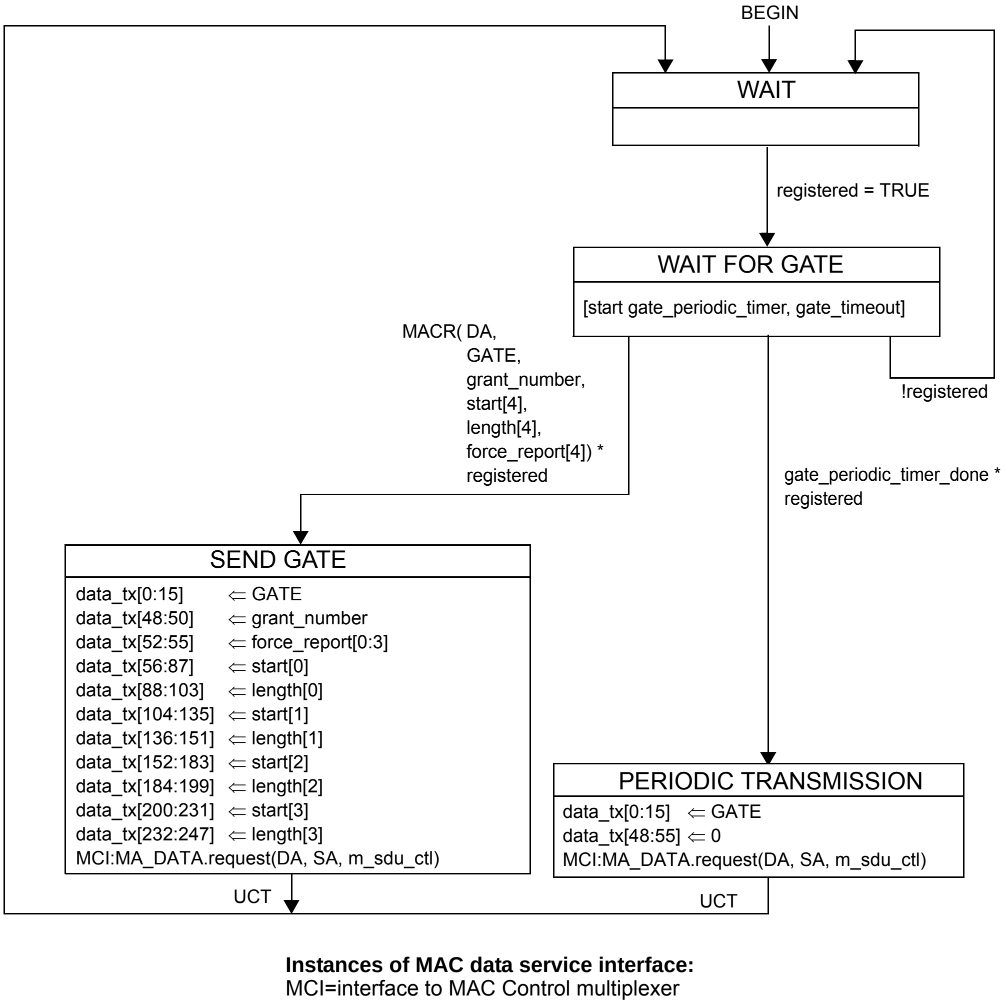

BEGIN
WAIT
registered = TRUE
WAIT FOR GATE
[start gate_periodic_timer, gate_timeout]
MACR(DA,
GATE,
grant_number,
start[4],
length[4],
force_report[4]) *
registered
!registered
gate_periodic_timer_done *
registered
SEND GATE
data_tx[0:15] ⇐ GATE
data_tx[48:50] ⇐ grant_number
data_tx[52:55] ⇐ force_report[0:3]
data_tx[56:87] ⇐ start[0]
data_tx[88:103] ⇐ length[0]
data_tx[104:135] ⇐ start[1]
data_tx[136:151] ⇐ length[1]
data_tx[152:183] ⇐ start[2]
data_tx[184:199] ⇐ length[2]
data_tx[200:231] ⇐ start[3]
data_tx[232:247] ⇐ length[3]
MCI:MA_DATA.request(DA, SA, m_sdu_ctl)
PERIODIC TRANSMISSION
data_tx[0:15] ⇐ GATE
data_tx[48:55] ⇐ 0
MCI:MA_DATA.request(DA, SA, m_sdu_ctl)
UCT
UCT
Instances of MAC data service interface:
MCI=interface to MAC Control multiplexer

Figure 103–28—Gate Processing state diagram at CLT

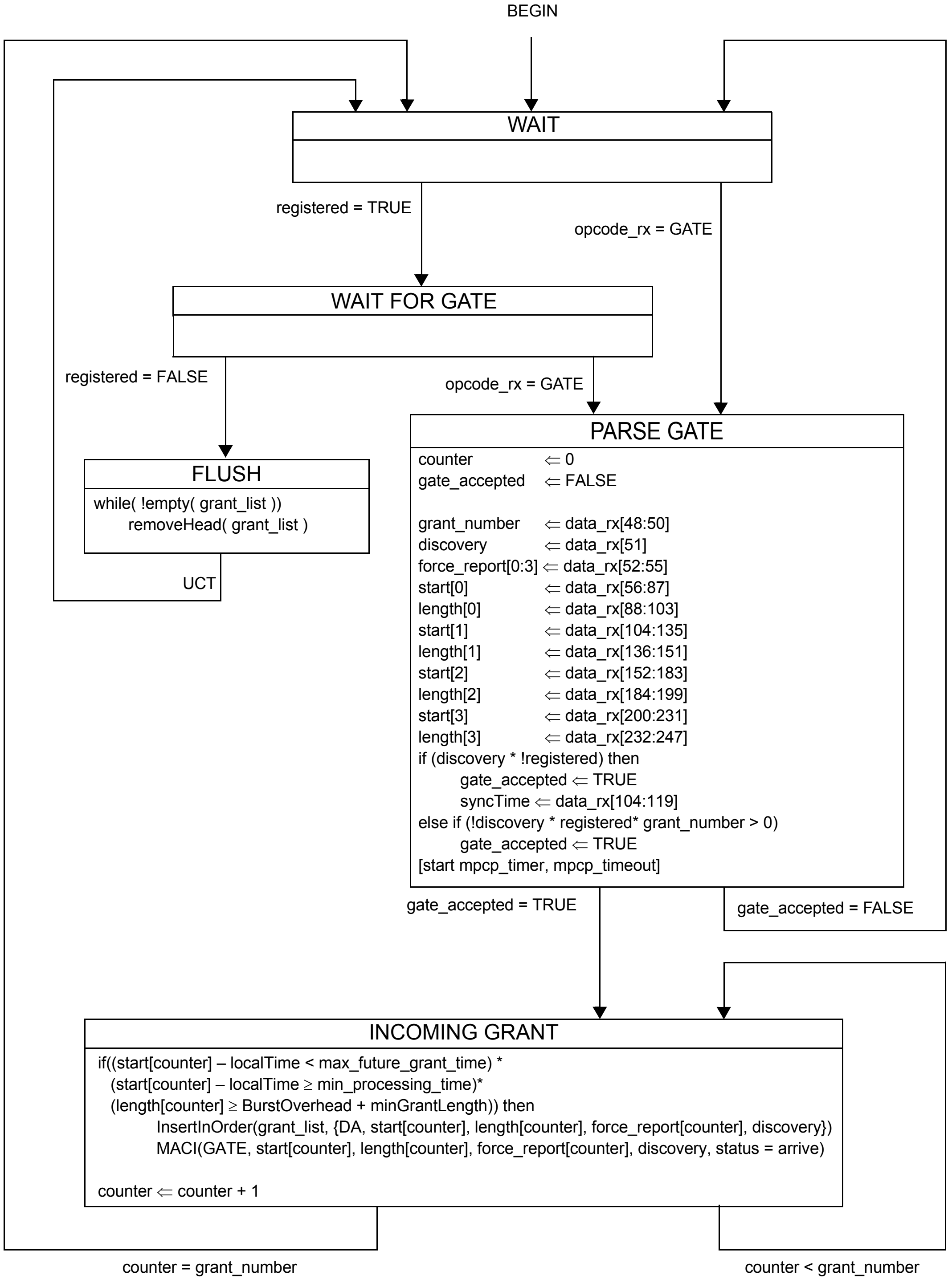

BEGIN
WAIT
registered = TRUE
opcode_rx = GATE
WAIT FOR GATE
registered = FALSE
opcode_rx = GATE
PARSE GATE
counter ⇐ 0
gate_accepted ⇐ FALSE
grant_number ⇐ data_rx[48:50]
discovery ⇐ data_rx[51]
force_report[0:3] ⇐ data_rx[52:55]
start[0] ⇐ data_rx[56:87]
length[0] ⇐ data_rx[88:103]
start[1] ⇐ data_rx[104:135]
length[1] ⇐ data_rx[136:151]
start[2] ⇐ data_rx[152:183]
length[2] ⇐ data_rx[184:199]
start[3] ⇐ data_rx[200:231]
length[3] ⇐ data_rx[232:247]
if (discovery * !registered) then
gate_accepted ⇐ TRUE
syncTime ⇐ data_rx[104:119]
else if (!discovery * registered* grant_number > 0)
gate_accepted ⇐ TRUE
[start mpcp_timer, mpcp_timeout]
FLUSH
while(!empty(grant_list))
removeHead(grant_list)
UCT
gate_accepted = TRUE
gate_accepted = FALSE
INCOMING GRANT
if((start[counter] – localTime < max_future_grant_time) *
(start[counter] – localTime ≥ min_processing_time)*
(length[counter] ≥ BurstOverhead + minGrantLength)) then
InsertInOrder(grant_list, {DA, start[counter], length[counter], force_report[counter], discovery})
MACI(GATE, start[counter], length[counter], force_report[counter], discovery, status = arrive)
counter ⇐ counter + 1
counter = grant_number
counter < grant_number

Figure 103–29—Gate Processing CNU Programing state diagram

EDITORS NOTE (to be removed prior to publication): the figure above “Gate Processing CNU Programing state diagram” will require modification if sub-clause 10x.4 “Discovery Process in dual-rate systems” is removed.

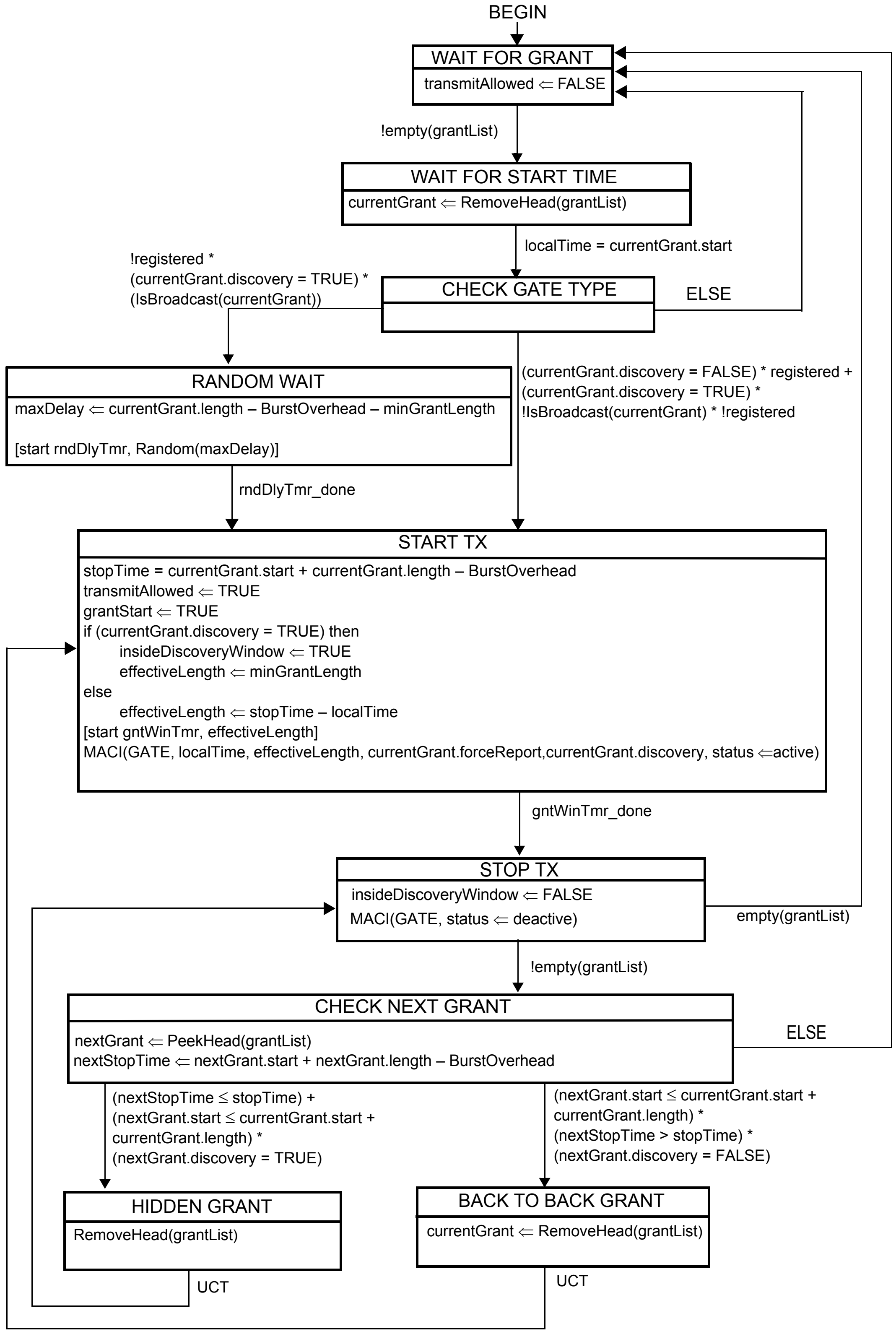

BEGIN
WAIT FOR GRANT
transmitAllowed ⇐ FALSE
!empty(grantList)
WAIT FOR START TIME
currentGrant ⇐ RemoveHead(grantList)
localTime = currentGrant.start
!registered *
(currentGrant.discovery = TRUE) *
(IsBroadcast(currentGrant))
CHECK GATE TYPE
ELSE
RANDOM WAIT
maxDelay ⇐ currentGrant.length – BurstOverhead – minGrantLength
[start rndDlyTmr, Random(maxDelay)]
(currentGrant.discovery = FALSE) * registered +
(currentGrant.discovery = TRUE) *
!IsBroadcast(currentGrant) * !registered
rndDlyTmr_done
START TX
stopTime = currentGrant.start + currentGrant.length – BurstOverhead
transmitAllowed ⇐ TRUE
grantStart ⇐ TRUE
if (currentGrant.discovery = TRUE) then
insideDiscoveryWindow ⇐ TRUE
effectiveLength ⇐ minGrantLength
else
effectiveLength ⇐ stopTime – localTime
[start gntWinTmr, effectiveLength]
MACI(GATE, localTime, effectiveLength, currentGrant.forceReport,currentGrant.discovery, status ⇐active)
gntWinTmr_done
STOP TX
insideDiscoveryWindow ⇐ FALSE
MACI(GATE, status ⇐ deactive)
empty(grantList)
!empty(grantList)
CHECK NEXT GRANT
nextGrant ⇐ PeekHead(grantList)
nextStopTime ⇐ nextGrant.start + nextGrant.length – BurstOverhead
ELSE
(nextStopTime ≤ stopTime) +
(nextGrant.start ≤ currentGrant.start +
currentGrant.length) *
(nextGrant.discovery = TRUE)
(nextGrant.start ≤ currentGrant.start +
currentGrant.length) *
(nextStopTime > stopTime) *
(nextGrant.discovery = FALSE)
HIDDEN GRANT
RemoveHead(grantList)
BACK TO BACK GRANT
currentGrant ⇐ RemoveHead(grantList)
UCT
UCT

Figure 103–30—Gate Processing CNU Activation state diagram

103.3.6 MPCPDU structure and encoding

The MPCPDU structure shall be as shown in Figure 103–31, and is further defined in the following definitions:

a) Destination Address (DA). The DA in MPCPDU is the MAC Control Multicast address as specified in the annexes to Clause 31, or the individual MAC address associated with the port to which the MPCPDU is destined.

b) Source Address (SA). The SA in MPCPDU is the individual MAC address associated with the port through which the MPCPDU is transmitted. For MPCPDUs originating at the CLT end, this can be the address any of the individual MACs. These MACs may all share a single unicast address, as explained in 103.1.2.

c) Length/Type. The Length/Type in MPCPDUs carries the MAC_Control_Type field value as specified in 31.4.1.3.

d) Opcode. The opcode identifies the specific MPCPDU being encapsulated. Values are defined in Table 31A-1.

e) *Timestamp*. The *timestamp* field conveys the content of the localTime register at the time of transmission of the MPCPDUs. This field is 32 bits long and counts time in units of time_quanta.

f) Data/Reserved/PAD. These 40 octets are used for the payload of the MPCPDUs. When not used they would be filled with zeros on transmission, and be ignored on reception.

g) FCS. This field is the Frame Check Sequence, typically generated by the underlying MAC. Based on the MAC instance used to generate the specific MPCPDU, the appropriate LLID shall be generated by the RS.

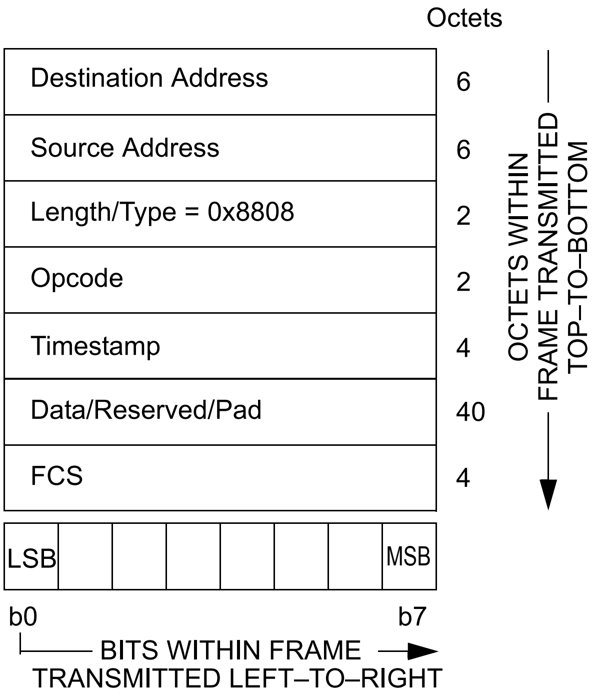


Figure 103–31—Generic MPCPDU

103.3.6.1 GATE description

The purpose of GATE message is to grant transmission windows to CNUs for both discovery messages and normal transmission. Up to four grants can be included in a single GATE message. The number of grants can also be set to zero for using the GATE message as an MPCP keep alive from CLT to the CNU.

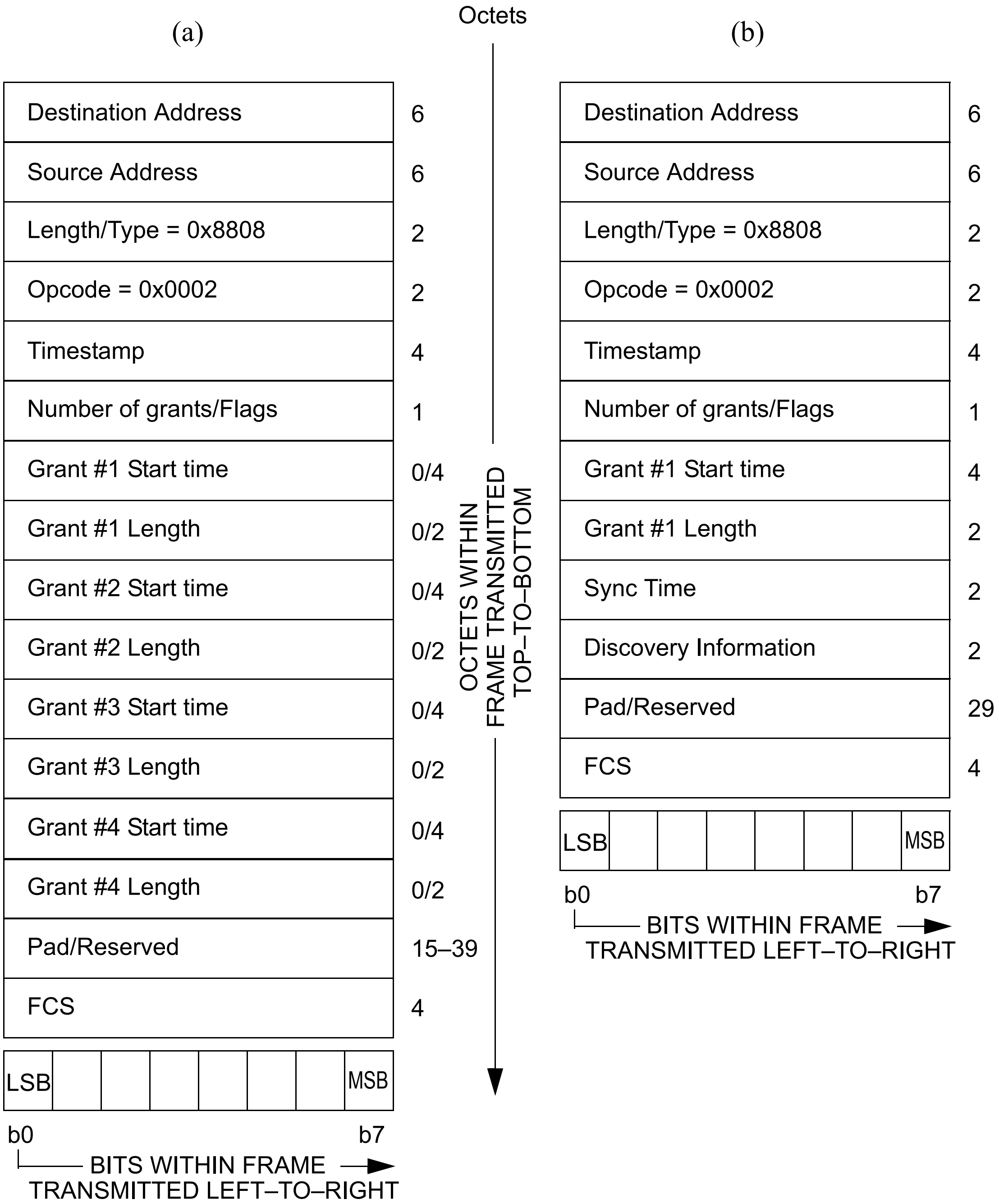


Figure 103–32—GATE MPCPDU: (a) normal GATE MPCPDU, (b) discovery GATE MPCPD

The GATE MPCPDU is an instantiation of the Generic MPCPDU, and is further defined using the following definitions:

a) Opcode. The opcode for the GATE MPCPDU is 0x0002.

b) Flags. This is an 8 bit flag register that holds the following flags: As presented in Table 103–1, the Number of grants field contains the number of grants, composed of valid Length, Start Time pairs in this MPCPDU. This is a number between 0 and 4.

NOTE When Number of grants is set to 0, sole purpose of message is conveying of *timestamp* to CNU.

The Discovery flag field indicates that the signaled grants would be used for the discovery process, in which case a single grant shall be issued in the GATE message.
The Force Report flag fields ask the CNU to issue a REPORT message related to the corresponding grant number at the corresponding transmission opportunity indicated in this GATE.

Table 103–1—GATE MPCPDU Number of grants/flags fields

Bit	Flag field	Values
0 – 2	Number of grants	0 – 4
3	Discovery	0 – Normal GATE 1 – Discovery GATE
4	Force Report Grant 1	0 – No action required 1 – A REPORT frame should be issued at the corresponding transmission opportunity indicated in Grant 1
5	Force Report Grant 2	0 – No action required 1 – A REPORT frame should be issued at the corresponding transmission opportunity indicated in Grant 2
6	Force Report Grant 3	0 – No action required 1 – A REPORT frame should be issued at the corresponding transmission opportunity indicated in Grant 3
7	Force Report Grant 4	0 – No action required 1 – A REPORT frame should be issued at the corresponding transmission opportunity indicated in Grant 4

c) Grant #n Start Time. This 32 bit unsigned field represents the start time of the grant. The start time is compared to the local clock, to correlate the start of the grant. Transmitted values shall satisfy the condition Grant #n Start Time < Grant #n+1 Start Time for consecutive grants within the same GATE MPCPDU.

d) Grant #n Length. This 16 bit unsigned field represents the length of the grant. The length is counted in 1 time_quantum increments. There are 4 Grants that are possibly packed into the GATE MPCPDU. The *rfOnTime*, *syncTime*, *rfOffTime*, two initial Idle blocks, FEC parity overhead, and burst terminator sequence (composed of three END_BURST_DELIMITER blocks) are included in and thus consume part of the Grant #n length.

e) Sync Time. This is an unsigned 16 bit value signifying the required synchronization time of the CLT receiver. The CNU calculates the effective grant length by subtracting the *syncTime*, *rfOnTime*, *rfOffTime* and END_BURST_DELIMITER from the grant length it received from the CLT. The value is counted in 1 time_quantum increments. The advertised value includes synchronization requirement on all receiver elements including PMD, PMA and PCS. This field is present only when the GATE is a discovery GATE, as signaled by the Discovery flag and is not present otherwise.

f) Discovery Information. This 16 bit flag register is reserved; all bits are ignored on reception.

g) Pad/Reserved. This is an empty field that is transmitted as zeros, and ignored on reception. The size of this field depends on the used Grant #n Length/Start Time entry-pairs as well as the presence of the Sync Time and Discovery Information fields, and varies in length from 15–39 accordingly.

The GATE MPCPDU shall be generated by a MAC Control instance mapped to an active CNU, and as such shall be marked with a unicast type of LLID, except when the MPCPDU is a discovery GATE, as indicated by the Discovery flag being set to TRUE. For the discovery procedure, a MAC Control instance is mapped to all CNUs, and therefore, the discovery GATE MPCPDU is marked with the appropriate broadcast LLID (see 103.3.2.3).

103.3.6.2 REPORT description

REPORT messages have several functionalities. Time stamp in each REPORT message is used for round trip (*RTT*) calculation. In the REPORT messages CNUs indicate the upstream bandwidth needs they request per IEEE 802.1Q priority queue. REPORT messages are also used as keep–alives from CNU to CLT. CNUs shall issue REPORT messages periodically in order to maintain link health at the CLT as defined in 103.3.4. In addition, the CLT may specifically request a REPORT message.

The REPORT MPCPDU is an instantiation of the Generic MPCPDU, and is further defined using the following definitions:

a) Opcode. The opcode for the REPORT MPCPDU is 0x0003.

b) Number of Queue Sets. This field specifies the number of requests in the REPORT message. A REPORT frame may hold multiple sets of Report bitmap and Queue #n as specified in the Number of Queue Sets field.

c) Report bitmap. This is an 8 bit flag register that indicates which queues are represented in this REPORT MPCPDU, see Table 103–2.

Table 103–2—REPORT MPCPDU Report bitmap fields

Bit	Flag field	Values
0	Queue 0	0 – queue 0 report is not present; 1 – queue 0 report is present
1	Queue 1	0 – queue 1 report is not present; 1 – queue 1 report is present
2	Queue 2	0 – queue 2 report is not present; 1 – queue 2 report is present
3	Queue 3	0 – queue 3 report is not present; 1 – queue 3 report is present
4	Queue 4	0 – queue 4 report is not present; 1 – queue 4 report is present
5	Queue 5	0 – queue 5 report is not present; 1 – queue 5 report is present
6	Queue 6	0 – queue 6 report is not present; 1 – queue 6 report is present
7	Queue 7	0 – queue 7 report is not present; 1 – queue 7 report is present

d) Queue #n Report. This value represents the length of queue #n at time of REPORT message generation. The reported length shall be adjusted and rounded up to the nearest time_quantum to account for the necessary inter–frame spacing and preamble. FEC parity overhead is not included in the reported length. The Queue #n Report field is an unsigned 16 bit integer representing the transmission request in units of time_quanta. This field is present only when the corresponding flag in the Report bitmap is set.

e) Pad/Reserved. This is an empty field that is transmitted as zeros, and ignored on reception. The size of this field depends on the used Queue Report entries, and accordingly varies in length from 0 to 39.

The REPORT MPCPDU shall be generated by a MAC Control instance mapped to an active CNU, and as such shall be marked with a unicast type of LLID.

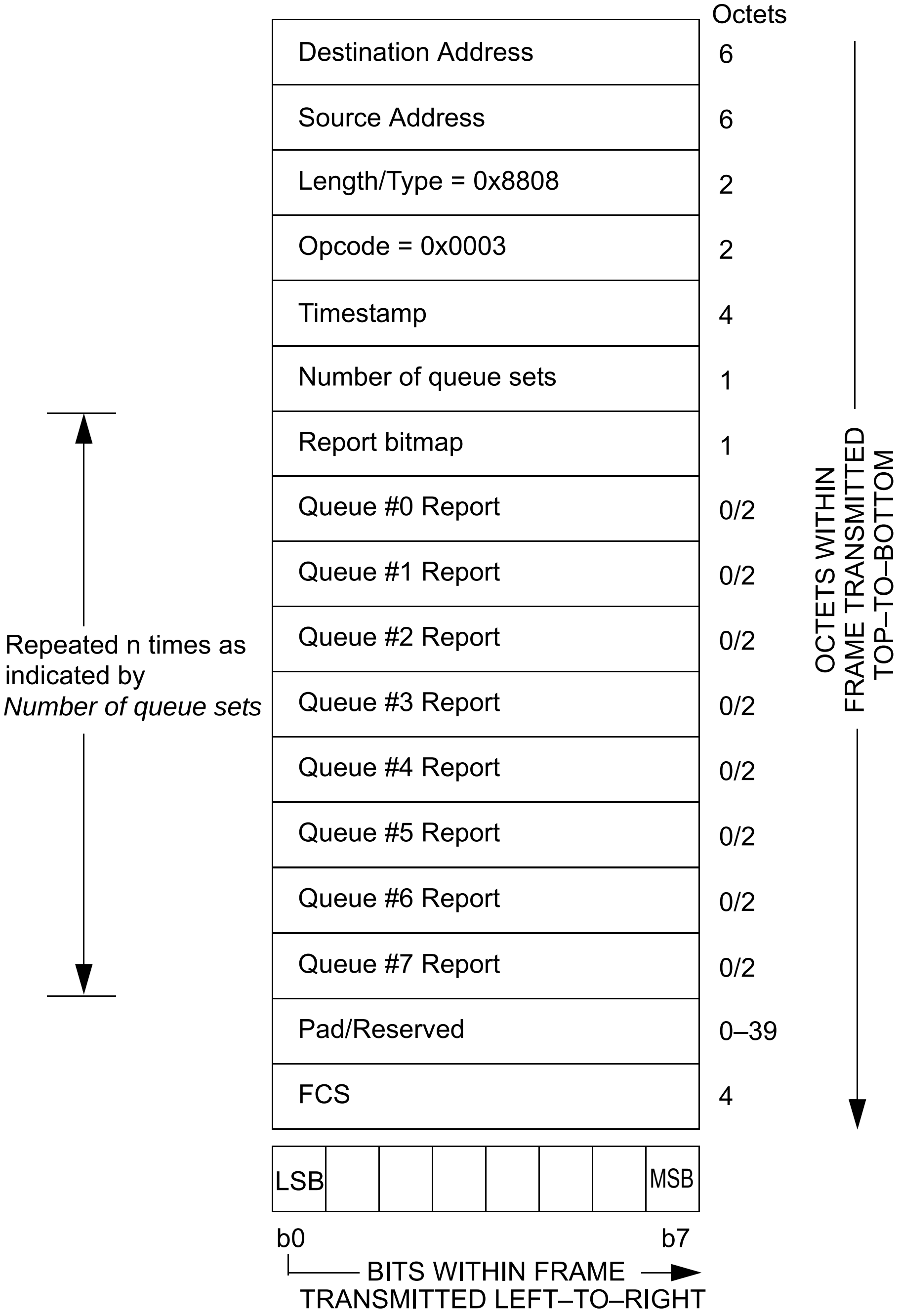


Figure 103–33—REPORT MPCPDU

103.3.6.3 REGISTER_REQ description

The REGISTER_REQ MPCPDU is an instantiation of the Generic MPCPDU, and is further defined using the following definitions:

a) Opcode. The opcode for the REGISTER_REQ MPCPDU is 0x0004.

b) Flags. This is an 8 bit flag register that indicates special requirements for the registration, as presented in Table 103–3.

Table 103–3—REGISTER_REQ MPCPDU Flags fields

Value	Indication	Comment
0	Reserved	Ignored on reception.
1	Register	Registration attempt for CNU.
2	Reserved	Ignored on reception.
3	Deregister	This is a request to deregister the CNU. Subsequently, the MAC is deallocated and the LLID may be reused.
4 – 255	Reserved	Ignored on reception.

c) Pending grants. This is an unsigned 8 bit value signifying the maximum number of future grants the CNU is configured to buffer. The CLT should not grant the CNU more than this maximum number of Pending grants vectors comprised of {start, length, force_report, discovery} into the future.

d) Discovery Information. This 16 bit flag register is reserved; all bits are ignored on reception.

e) RF On Time. This field is 1 octet long and carries the RF On Time characteristic for the given CNU transmitter. The value is expressed in the units of time_quanta.

f) RF Off Time. This field is 1 octet long and carries the RF Off Time characteristic for the given CNU transmitter. The value is expressed in the units of time_quanta.

g) Pad/Reserved. This is an empty field that is transmitted as zeros, and ignored on reception.

The REGISTER_REQ MPCPDU shall be generated by a MAC Control instance mapped to an undiscovered CNU, and as such shall be marked with a broadcast type of LLID (103.3.2.3).

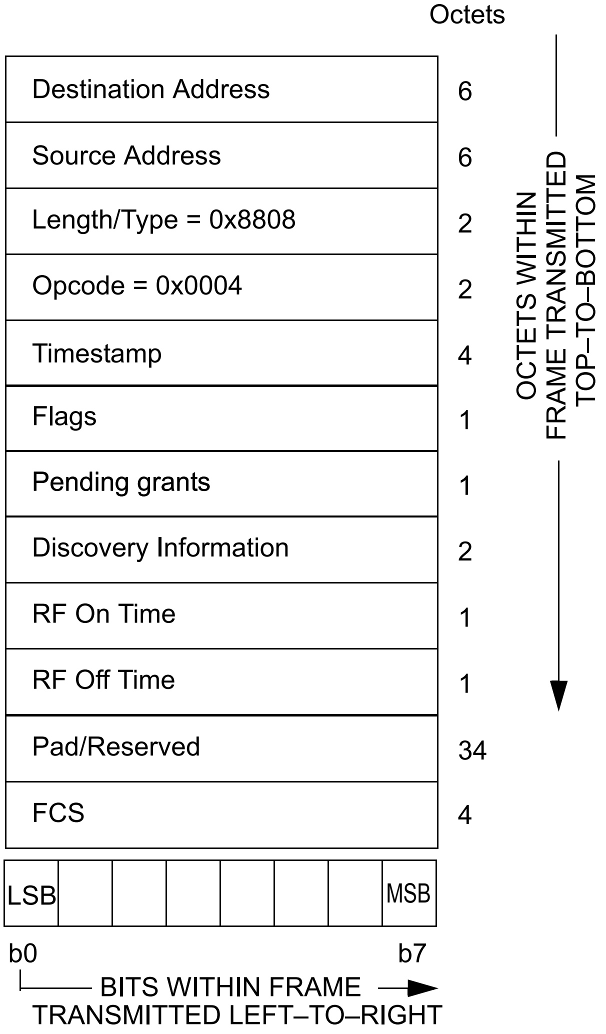


Figure 103–34—REGISTER_REQ MPCPDU

103.3.6.4 REGISTER description

The REGISTER MPCPDU is an instantiation of the Generic MPCPDU, and is further defined using the following definitions:

a) DA. The destination address used shall be an individual MAC address.

b) Opcode. The opcode for the REGISTER MPCPDU is 0x0005.

c) Assigned Port. This field holds a 16 bit unsigned value reflecting the LLID of the port assigned following registration.

d) Flags. this is an 8 bit flag register that indicates special requirements for the registration, as presented in Table 103–4.

e) Sync Time. This is an unsigned 16 bit value signifying the required synchronization time of the CLT receiver. The CNU calculates the effective grant length by subtracting the *syncTime*, *rfOnTime*, *rfOffTime*, and END_BURST_DELIMITER from the grant length it received from the CLT. The value is counted in 1 time_quantum increments. The advertised value includes synchronization requirement on all receiver elements including PMD, PMA, and PCS.

Table 103–4—REGISTER MPCPDU Flags field

Value	Indication	Comment
0	Reserved	Ignored on reception.
1	Reregister	The CNU is explicitly asked to re-register.
2	Deregister	This is a request to deallocate the port and free the LLID. Subsequently, the MAC is deallocated.
3	Ack	The requested registration is successful.
4	Nack	he requested registration attempt is denied by the MAC Control Client.
5 – 255	Reserved	Ignored on reception.

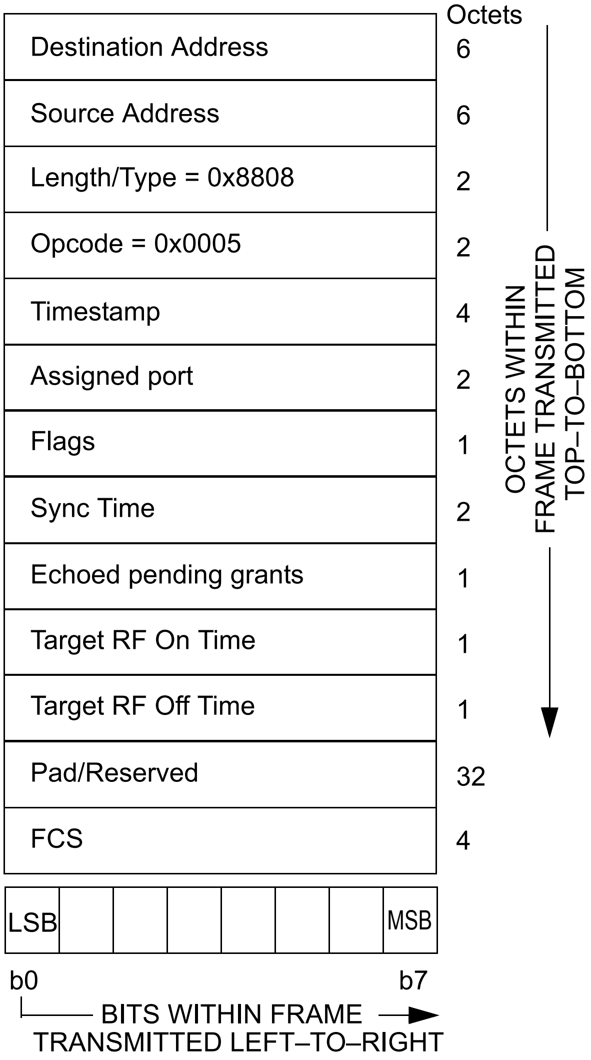


Figure 103–35—REGISTER MPCPDU

f) Echoed pending grants. This is an unsigned 8 bit value signifying the number of future grants the CNU may buffer before activating. The CLT should not grant the CNU more than this number of grants into the future.

g) Target RF On Time. This is an unsigned 8 bit value, expressed in the units of time_quanta, signifying the RF On Time for the given CNU transmitter. This value may be different from RF On Time delivered by the CNU in the REGISTER_REQ MPCPDU during the Discovery process. The CNU updates the local *rfOnTime* variable per state diagram in Figure 103–23.

h) Target RF Off Time. This is an unsigned 8 bit value, expressed in the units of time_quanta, signifying the RF Off Time for the given CNU transmitter. This value may be different from RF Off Time delivered by the CNU in the REGISTER_REQ MPCPDU during the Discovery process. The CNU updates the local *rfOffTime* variable per state diagram in Figure 103–23.

i) Pad/Reserved. This is an empty field that is transmitted as zeros, and ignored on reception.

The REGISTER MPCPDU shall be generated by a MAC Control instance mapped to all CNUs and such frame is marked by the broadcast LLID (103.3.2.3).

103.3.6.5 REGISTER_ACK description

The REGISTER_ACK MPCPDU is an instantiation of the Generic MPCPDU, and is further defined using the following definitions:

a) Opcode. The opcode for the REGISTER_ACK MPCPDU is 0x0006.

b) Flags. This is an 8 bit flag register that indicates special requirements for the registration, as presented in Table 103–5.

c) Echoed assigned port. This field holds a 16 bit unsigned value reflecting the LLID for the port assigned following registration.

d) Echoed Sync Time. This is an unsigned 16 bit value echoing the required synchronization time of the CLT receiver as previously advertised (103.3.6.4).

e) Pad/Reserved. This is an empty field that is transmitted as zeros, and ignored at reception.

Table 103–5—REGISTER_ACK MPCPDU Flags fields

Value	Indication	Comment
0	Nack	The requested registration attempt is denied by the MAC Control Client.
1	Ack	The registration process is successfully acknowledged.
2 – 255	Reserved	Ignored on reception.

The REGISTER_ACK MPCPDU shall be generated by a MAC Control instance mapped to an active CNU, and as such shall be marked with a unicast type of LLID.

103.4 Protocol implementation conformance statement (PICS) proforma for Clause 103, Multipoint MAC Control for EPoC[4]

103.4.1 Introduction

The supplier of a protocol implementation that is claimed to conform to Clause 103, clause title, shall complete the following protocol implementation conformance statement (PICS) proforma.

A detailed description of the symbols used in the PICS proforma, along with instructions for completing the PICS proforma, can be found in Clause 21.

103.4.2 Identification

103.4.2.1 Implementation identification

Supplier[1]	
Contact point for enquiries about the PICS[1]	
Implementation Name(s) and Version(s)[1,3]	
Other information necessary for full identification—e.g., name(s) and version(s) for machines and/or operating systems; System Name(s)[2]	
NOTE 1—Required for all implementations. NOTE 2—May be completed as appropriate in meeting the requirements for the identification. NOTE 3—The terms Name and Version should be interpreted appropriately to correspond with a supplier’s terminology (e.g., Type, Series, Model).	

EDITORS NOTE (to be removed prior to publication): Implementation ID table is different in 2012 version.

[4] *Copyright release for PICS proformas:* Users of this standard may freely reproduce the PICS proforma in this subclause so that it can be used for its intended purpose and may further publish the completed PICS.

103.4.2.2 Protocol summary

Identification of protocol standard	IEEE Std 802.3xx-2012, Clause 103, clause title
Identification of amendments and corrigenda to this PICS proforma that have been completed as part of this PICS	
Have any Exception items been required? No [ ] Yes [ ] (See Clause 21; the answer Yes means that the implementation does not conform to IEEE Std 802.3xx-2012.)	

Date of Statement	

103.4.3 Major capabilities/options

Item	Feature	Subclause	Value/Comment	Status	Support
*CLT	CLT functionality	103.1	Device supports functionality required for CLT	O/1	Yes [ ] No [ ]
*CNU	CNU functionality	103.1	Device supports functionality required for CNU	O/1	Yes [ ] No [ ]

103.4.4 PICS proforma tables for Multipoint MAC Control

103.4.4.1 Compatibility considerations

Item	Feature	Subclause	Value/Comment	Status	Support
CC1	Delay through MAC	103.3.2.4	Maximum delay variation of 1 time_quantum	M	Yes [ ] No [ ]
CC2	CLT grant time delays	103.3.2.4	Not grant nearer than 1 time_quanta into the future	CLT:M	Yes [ ] No [ ] N/A [ ]
CC3	CNU processing delays	103.3.2.4	Process all messages in less than TBD time_quanta	CNU:M	Yes [ ] No [ ] N/A [ ]
CC4	CLT grant issuance	103.3.2.4	Not grant more than one message every TBD time_quanta to a single CNU	CLT:M	Yes [ ] No [ ] N/A [ ]

103.4.4.2 Multipoint MAC Control

Item	Feature	Subclause	Value/Comment	Status	Support
OM1	CLT localTime	64.2.2.2	Track transmit clock	CLT:M	Yes [ ] No [ ] N/A [ ]
OM2	CNU localTime	64.2.2.2	Track receive clock	CNU:M	Yes [ ] No [ ] N/A [ ]
OM3	Random wait for transmitting REGISTER_REQ messages	77.3.3	Shorter than length of discovery window	CNU:M	Yes [ ] No [ ] N/A [ ]
OM4	Periodic report generation	77.3.4	Reports are generated periodically	CNU:M	Yes [ ] No [ ] N/A [ ]
OM5	Periodic granting	77.3.4	Grants are issued periodically	CLT:M	Yes [ ] No [ ] N/A [ ]
OM6	Issuing of grants	77.3.5	Not issue more than maximum supported grants	CLT:M	Yes [ ] No [ ] N/A [ ]

103.4.4.3 State diagrams

Item	Feature	Subclause	Value/Comment	Status	Support
SM1	Multipoint Transmission Control	103.2.2.7	Meets the requirements of Figure 103–10	M	Yes [ ] No [ ]
SM2	CLT Control Parser	103.2.2.7	Meets the requirements of Figure 103–11	M	Yes [ ] No [ ]
SM3	CNU Control Parser	103.2.2.7	Meets the requirements of Figure 103–12	M	Yes [ ] No [ ]
SM4	CLT Control Multiplexer	103.2.2.7	Meets the requirements of Figure 103–13	CLT:M	Yes [ ] No [ ] N/A [ ]
SM5	CNU Control Multiplexer	103.2.2.7	Meets the requirements of Figure 103–14	CLT:M	Yes [ ] No [ ] N/A [ ]
SM6	Discovery Processing CLT Window Setup	103.3.3.6	Meets the requirements of Figure 103–19	CLT:M	Yes [ ] No [ ] N/A [ ]
SM7	Discovery Processing CLT Process Requests	103.3.3.6	Meets the requirements of Figure 103–20	CLT:M	Yes [ ] No [ ] N/A [ ]
SM8	Discovery Processing CLT Register	103.3.3.6	Meets the requirements of Figure 103–21	CNU:M	Yes [ ] No [ ] N/A [ ]
SM9	Discovery Processing CLT Final Registration	103.3.3.6	Meets the requirements of Figure 103–22	CLT:M	Yes [ ] No [ ] N/A [ ]
SM10	Discovery Processing CNU Registration	103.3.3.6	Meets the requirements of Figure 103–23	CNU:M	Yes [ ] No [ ] N/A [ ]
SM11	Report Processing at CLT	103.3.4.6	Meets the requirements of Figure 103–25	CLT:M	Yes [ ] No [ ] N/A [ ]
SM12	Report Processing at CNU	103.3.4.6	Meets the requirements of Figure 103–26	CNU:M	Yes [ ] No [ ] N/A [ ]
SM13	Gate Processing at CLT	103.3.5.6	Meets the requirements of Figure 103–28	CLT:M	Yes [ ] No [ ] N/A [ ]
SM14	Gate Processing at CNU	103.3.5.6	Meets the requirements of Figure 103–29	CNU:M	Yes [ ] No [ ] N/A [ ]
SM15	Gate Processing CNU Activation	103.3.5.6	Meets the requirements of Figure 103–30	CNU:M	Yes [ ] No [ ] N/A [ ]

103.4.4.4 MPCP

Item	Feature	Subclause	Value/Comment	Status	Support
MP1	MPCPDU structure	77.3.6	As in Figure 103–31	M	Yes [ ] No [ ]
MP2	LLID for MPCPDU	77.3.6	RS generates LLID for MPCPDU	M	Yes [ ] No [ ]
MP3	Grants during discovery	77.3.6.1	Single grant in GATE message during discovery	CLT:M	Yes [ ] No [ ] N/A [ ]
MP4	Grant start time	77.3.6.1	Grants within one GATE MPCPDU are sorted by their Start time values	CLT:M	Yes [ ] No [ ] N/A [ ]
MP5	GATE generation	77.3.6.1	GATE generated for active CNU except during discovery	CLT:M	Yes [ ] No [ ] N/A [ ]
MP6	GATE LLID	77.3.6.1	Unicast LLID except for discovery	CLT:M	Yes [ ] No [ ] N/A [ ]
MP7	REPORT issuing	77.3.6.2	Issues REPORT periodically	CNU:M	Yes [ ] No [ ] N/A [ ]

Item	Feature	Subclause	Value/Comment	Status	Support
MP8	REPORT generation	77.3.6.2	Generated by active CNU	CNU:M	Yes [ ] No [ ] N/A [ ]
MP9	REPORT queue #n	77.3.6.2	REPORT Queue #n length rounding	CNU:M	Yes [ ] No [ ] N/A [ ]
MP10	REPORT LLID	77.3.6.2	REPORT has unicast LLID	CNU:M	Yes [ ] No [ ] N/A [ ]
MP11	REGISTER_REQ generation	77.3.6.3	Generated by undiscovered CNU	CNU:M	Yes [ ] No [ ] N/A [ ]
MP12	REGISTER_REQ LLID	77.3.6.3	Use broadcast LLID	CNU:M	Yes [ ] No [ ] N/A [ ]
MP13	REGISTER DA address	77.3.6.4	Use individual MAC address	CLT:M	Yes [ ] No [ ] N/A [ ]
MP14	REGISTER generation	77.3.6.4	Generated for all CNUs	CLT:M	Yes [ ] No [ ] N/A [ ]
MP15	REGISTER_ACK generation	77.3.6.5	Generated by active CNU	CNU:M	Yes [ ] No [ ] N/A [ ]
MP16	REGISTER_ACK LLID	77.3.6.5	Use unicast LLID	CNU:M	Yes [ ] No [ ] N/A [ ]
MP16	MAC enable	103.2.2.4	MAC Control interface has prioroty over other clients	CLT:M	Yes [ ] No [ ] N/A [ ]