Editing Instructions: replace the empty constant definitions in Cl 143 with the following:

Add in section 143.4.3.2

INTER_ENV_IDLE

TYPE: 72-bit vector

Value: 0xFF 08 08 08 08 08 08 08 08

The value of an EQ which represents idle space between transmissions

PARITY_PLACEHLDR

TYPE: 72-bit vector

Value: 0xFF 09 09 09 09 09 09 09 09

The value of an EQ which represents FEC Parity bits within a transmission.

Editing Instructions: Replace Subclause 142.2 with the following. Note this document shows changes from Cl 142.2 via MS Word mark-up.

142.2 Physical Coding Sublayer (PCS) for 100G-EPON

142.2.1 Overview

This subclause defines the physical coding sublayers {NG-EPON type} supporting burst mode operation over the point-to-multipoint physical medium. The {NG-EPON type, symmetric} PCS is specified to support {NG-EPON types}, where both the receive and transmit paths operate at multiples of 25.78125 Gb/s rate. The {NG-EPON type, asymmetric} PCS supports {NG-EPON types}, in which OLT transmit path and ONU receive path operate at 25.78125 Gb/s, while the ONU transmit path and the OLT receive path operate at 10.3125 Gb/s rate. Figure XXX and Figure XXX show the relationship between the PCS sublayer and the ISO/IEC OSI reference model.

The PCS functional block diagram is shown in 0.

This subclause also specifies a forward error correction (FEC) mechanism to increase the optical link budget or the fiber distance.

tbd

Figure 142- 1 PCS Functional Block Diagram

142.2.1.1 {NG-EPON, asymmetric} PCS

{TBD}

142.2.1.2 {NG-EPON, symmetric} PCS

{TBD}

142.2.2 PCS transmit function

This subclause defines the transmit direction of the physical coding sublayers for {NG-EPON type}. In the OLT, the PCS transmit function operates at a 25.78125 Gb/s rate in a continuous mode. In the ONU, the PCS transmit function may operate at a 25.78125 Gb/s rate, as specified herein ({NG-EPON type, symmetric}), or at a {TBD} Gb/s rate, as specified in {TBD} ({NG-EPON type, asymmetric}). For all {NG-EPON type}, the ONU PCS operates in a burst mode in the transmit direction. The PCS includes a mandatory}LDPC FEC encoder. The functional block diagram for the PCS transmit function is shown in 0. The transmit function consists of the following functional blocks.

- Transmit/Encode block (see 142.2.2.1),
- Data Detect block (ONU only, see 142.2.2.2),
- 64B/66B to 256B/257B Transcoder (see 142.2.2.3),
- Scrambler (see 142.2.2.4),
- FEC Encoder (see 142.2.2.5), and
- Gear Box (see142.2.2.6).

142.2.2.1 Transmit/Encode

The Transmit/Encode functional block accepts data from the one 25GMII interface and converts two consecutive 36-bit transfers into a single 72-bit tx_raw vector which is then encoded into a single 64B/66B block. The 64B/66B block structure is as defined in 49.2.4 with exceptions as noted in this subclause. The state diagram of the Transmit/Encode block is shown in Figure 142- 2.

142.2.2.1.1 Block Structure

The 25BGASE-PR PCS supports all the block type fields in Figure 49-7 except block type field values of: 0x2d, 0x33, 0x66, 0x55, and 0x4b.

142.2.2.1.2 Control codes

The {NG EPON type} PCS supports the control codes shown in Table 142- 1. The representations of the control characters are the control codes. Control characters are transferred over the 25GMII as an 8-bit value. The 25GBASE-PR PCS encodes the start and terminate control characters implicitly using the block type field. The 25GBASE-PR PCS does not encode the ordered set control codes. The 10GBASE-R PCS encodes each of the other control characters into a 7-bit C code.

The control characters and their mappings to 25GBASE-PR control codes are specified in Table 142- 1. All 25GMII and 25GBASE-PR control code values that do not appear in the table shall not be transmitted and are treated as an error if received.

Table 142- 1 Control Codes

Control character	Notation	25GMII control code	25GBASE-PR control code
idle	/I/	0x07	0x00
Inter-envelope idle	/IEI/	0x08	0x08
Parity placeholder	/P/	0x09	0x09
start	/S/	0xFB	Encoded by block type field
terminate	/T/	0xFD	Encoded by block type field
error	/E/	0xFE	0x1E

142.2.2.1.3 Constants

EBLOCK_T - see 49.2.13.2.1.

LBLOCK_T - see 49.2.13.2.1.

142.2.2.1.4 Variables

tx_coded – see 49.2.13.2.2.

tx_raw – see 49.2.13.2.2.

142.2.2.1.5 Functions

ENCODE(tx_raw) - see 49.2.13.2.3.

NextTxValid(prev_tx_coded, next_tx_raw)
This function returns a Boolean indicating whether the next_tx_raw vector is valid given the classification of the

current (next_tx_raw) and previously transmitted (prev_tx_coded) vectors. The function returns the values according to Table 142- 2. Vector classifications used in Table 142- 2 are shown in Table 142- 3.

Table 142- 2 NextTxValid and NextRxValid function output

		next_tx_raw/next_rx_coded vector classification						
		IEI	**S**	**D**	**T**	**I**	**P**	**Other**
prev_tx_coded/prev_rx_raw vector classification	L	true	false	false	false	false	false	false
	IEI	true	true	false	false	false	true	false
	S	true	true	true	true	true	true	false
	D	true	true	true	true	true	true	false
	T	true	true	true	false	true	true	false
	I	true	true	true	false	true	true	false
	P	true	true	true	true	true	true	false
	other	true	true	true	true	true	true	false

Table 142- 3 Vector classifications

		Criteria for tx_raw/rx_raw vector	**Criteria for tx_coded/rx_coded vector**
Classification	**L**	rx_raw<71:0> = LBLOCK_R (see 49.2.13.2.1). This classification does not apply to tx_raw<71:0>.	tx_coded<65:0> = LBLOCK_T (see 49.2.13.2.1). This classification does not apply to rx_coded<65:0>.
	IEI	Vector composed of INTER_ENV_IDLE (see 143.4.3.2)	Vector composed of Inter-envelope idle (vector<1:0> = 10, vector<9:2> = 0x1E, and all control codes = 0x08).
	S	Vector beginning with a Start control code symbol (vector<7:0> = 0x80, vector<15:8> = 0xFB)	Vector comprised of a Start control code symbol (vector<1:0> = 10 and vector<9:2> = 0x78)
	D	Vector of all data bytes (vector<7:0> = 0x00)	Vector of all data bytes (vector<1:0> = 01)
	T	Vector which includes a Terminate control code symbol (vector<7:0> ∈ {0xFF, 0x7F, 0x3F, 0x1F, 0x0F, 0x07, 0x03, 0x01}, 1st control code octet = 0xFD, and all other control characters are valid)	Vector which includes a Terminate control code symbol (vector<1:0> = 10 and vector<9:2> ∈ {0x87, 0x99, 0xAA, 0xB4, 0xCC, 0xD2, 0xE1, 0xFF}, and all control characters are valid)
	I	Vector composed of all Idle control code symbols (vector<7:0> = 0xFF and all other octets= 0x07)	Vector composed of all Idle control code symbols vector<1:0> = 10 and vector<65:2> = 0x00..00)
	P	Vector composed of PARITY_PLACEHLDR (see 143.4.3.2)	Vector composed of all Parity placeholder (vector<1:0> = 10, vector<9:2> = 0x1E, and all control codes = 0x09)

	E	rx_raw<71:0> = EBLOCK_R (see 49.2.13.2.1). This classification does not apply to tx_raw<71:0>.	tx_coded<65:0> = EBLOCK_T (see 49.2.13.2.1). This classification does not apply to rx_coded<65:0>.

NextTxVector()
This function returns the next 72-bit vector from the 25GMII.

142.2.2.1.6 State Diagrams

The OLT and the ONU shall implement the Transmit/Encode process as depicted in Figure 142- 2.

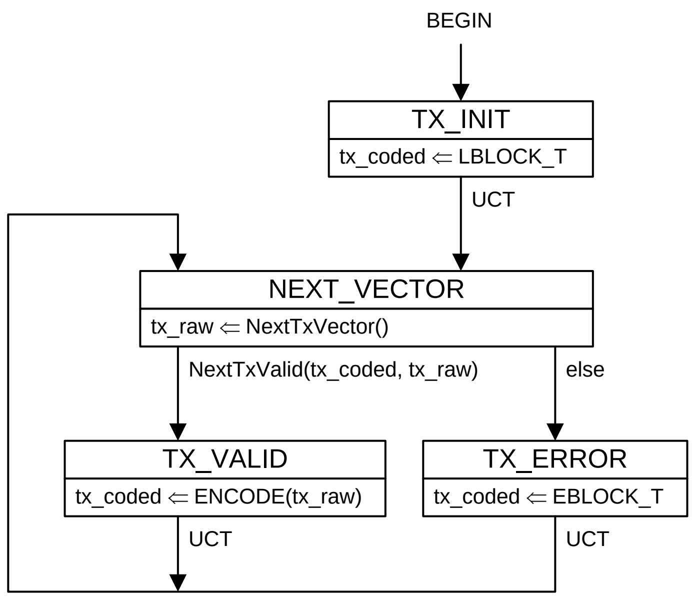

Figure 142- 2 Transmit/Encode State Diagram

142.2.2.2 Data detector

{TBD}

142.2.2.2.1 Burst Mode operation (ONU only)

{TBD}

142.2.2.3 64B/66B to 256B/257B transcoder

The 64B/66B to 256B/257B transcoder converts four consecutive 64B/66B blocks the into one 256B/257B block as described in 91.5.2.5 and passes the resulting 257-bit-wide block to the Scrambler functional block. In the OLT the 64B/66B blocks are received from Transmitter/Encoder functional block whereas in the ONU the 64B/66B blocks are received from the Data Detector.

142.2.2.4 Scrambler

See 49.2.6.

142.2.2.5 FEC encoder

Editing Instruction: Retain what is in D0.7 for this sub-clause

142.2.2.6 Gearbox

{TBD}

142.2.3 PCS receive Function

This subclause defines the receive direction of physical coding sublayers for {NG-EPON type}. In the ONU, the PCS receive function operates at a 25.78125 Gb/s rate in a continuous mode. In the OLT, the PCS receive function may operate at a 25.78125 Gb/s rate, as specified herein ({NG-EPON type, symmetric}), or at a 10.3125 Gb/s rate, compliant with Clause {TBD} ({NG-EPON type, asymmetric}). For all {NG-EPON types}, the OLT PCS receive function operates in burst mode. The PCS includes a mandatory FEC decoder. The functional block diagram for the PCS receive function is shown in 0. The receive function consists of the following functional blocks:

- Synchronizer block (see 142.2.3.1 and 142.2.3.2),
- FEC Decoder (see 142.2.3.4),
- Descrambler (see 142.2.3.5),
- 256B/257B to 64B/66B Transcoder (see 142.2.3.6), and
- Receiver/Decode block (see 142.2.3.7).

142.2.3.1 OLT synchronizer

{TBD}

142.2.3.2 ONU Synchronizer

The ONU synchronization receives data via the {TBD}-bit PMA_UNITDATA.indication primitive. The synchronizer forms a bit stream from the primitives by concatenating requests with the bits of each primitive in order from rx_data-group<xx> to rx_data-group<xx> (see Figure was 76–19). It obtains lock to the FEC codewords within the bit stream using the mechanism shown in Figure 142- 3 and outputs {TBD} codewords to the FEC decoder function.

{TBD description of block handling}

While in codeword lock, the synchronizer copies the FEC-protected bits from each data block and the parity bits of the codeword into an input buffer. When the codeword is complete, the FEC decoder is triggered, and the input buffer is freed for the next codeword.

When in codeword lock, the state diagram continues to check for sync header validity. If 16 or more sync headers in a codeword pair (62 blocks) are invalid, then the state diagram deasserts codeword lock. In addition, if the persist_dec_fail signal becomes set, then codeword lock is deasserted (this check ensures that certain false-lock cases are not persistent.)

142.2.3.2.1 Constants

FEC_CW_SZ

TYPE: Integer
The size of the FEC Codeword in bits.
VALUE: {TBD}

Note to Editor: Note FEC_CW_SZ will likely be defined before this section and could just be cross referenced.

FecFailLimit

TYPE: Integer
The number of FEC decoding failures allowed while in codeword lock before declaring out of lock
VALUE: {TBD}

MatchTarget

TYPE: Integer
The number of parity delimiters required to transition from a codeword out of lock start to a codeword lock state.
VALUE: {TBD}

PD

TYPE: binary array of {TBD}-bits

The burst delimiter bit pattern found at the beginning of each FEC Parity block.

VALUE: {TBD}

142.2.3.2.1 Variables and counters

FecDecodeFail

TYPE: Boolean

This clear on read variable indicates the most recent completed FEC codeword decoding failed.

FecDecodeSucceed

TYPE: Boolean

This clear on read variable indicate the most recently completed FEC codeword decoding succeeded.

FecFailCount

TYPE: Integer

This counter track the number of consecutive FEC decoding failures.

Match

TYPE: Boolean

This variable holds the most recent result of the Compare() function.

MatchCount

TYPE: Integer

This counter tracks the number of consecutive successful parity delimiter matched.

rx_buffer

TYPE: binary array

This array hold the sequence of concatenated bits received from the PMA_UNITDATA.indication primitive.

142.2.3.2.1 Functions

Compare(v, p)

This function compares bit by bit its two arguments and returns a Boolean ‘true’ if the number of bits that are different is less or equal to the Hamming threshold of {TBD} otherwise the function returns false.

Slip(v, bc)

This function removes “bc” bits from the passed array “v”.

142.2.3.2.1 State Diagrams

The ONU Synchronizer shall implement the state diagram as depicted in Figure 76–20.

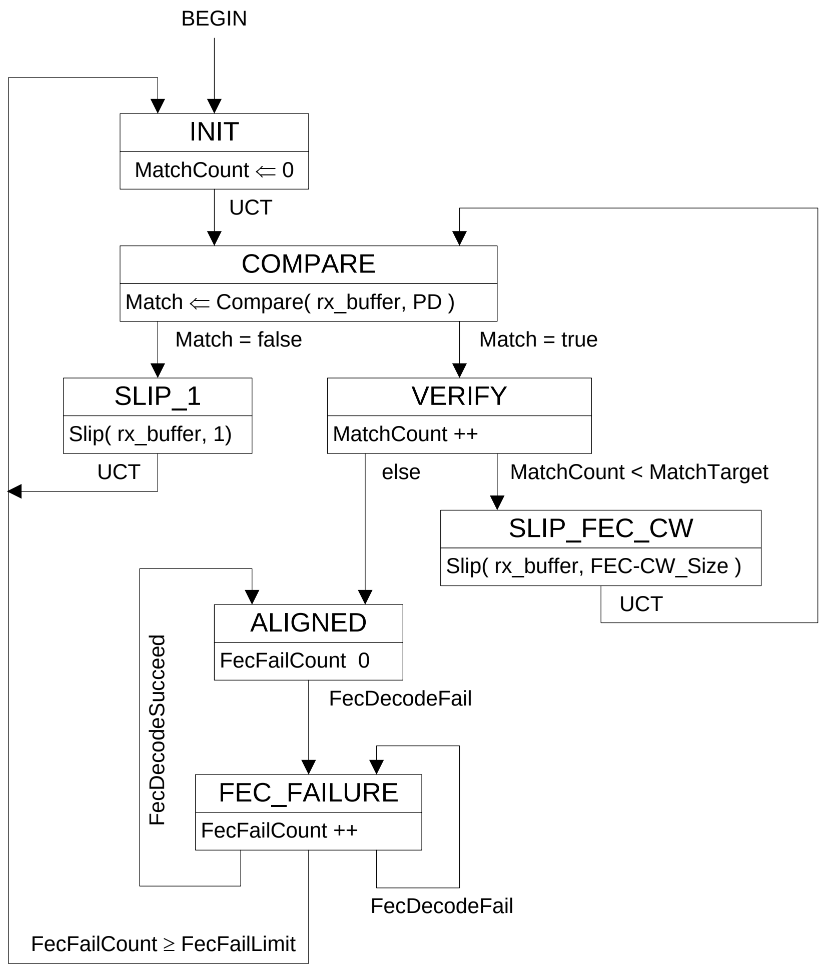

Figure 142- 3 ONU Synchronizer state diagram

142.2.3.3 BER monitor

{TBD}

142.2.3.4 FEC decoder

{TBD}

142.2.3.5 Descrambler

See 49.2.10.

142.2.3.6 256B/257B to 64B/66B transcoder

The 256B/257B to 64B/66B transcoder converts one 256B/257B block received from the Descrambler functional block into four consecutive 64B/66B blocks as described in 91.5.3.5 and passes these to the Receiver/Decoder functional block.

142.2.3.7 Receive/Decode

See 49.2.11. The decoder shall perform functions specified in the state diagram shown in Figure 49–17.

142.2.3.7.1 Constants

EBLOCK_R - see 49.2.13.2.1.

LBLOCK_R - see 49.2.13.2.1.

142.2.3.7.2 Variables

rx_coded – see 49.2.13.2.2.

rx_raw – see 49.2.13.2.2.

142.2.3.7.3 Functions

DECODE(rx_coded) - see 49.2.13.2.3.

NextRxValid(prev_rx_raw, next_rx_coded)

> This function returns a Boolean indicating whether the next_rx_coded vector is valid given the classification of the previously received prev_rx_raw vector. The function returns the values according to Table 142- 2. Vector classifications used in Table 142- 2 are shown in Table 142- 3.

NextRxVector()

> function which returns the next 66-bit vector from the Descrambler.

142.2.3.7.4 State Diagrams

The OLT and the ONU shall implement the Receive/Decode process as depicted in Figure 142- 4.

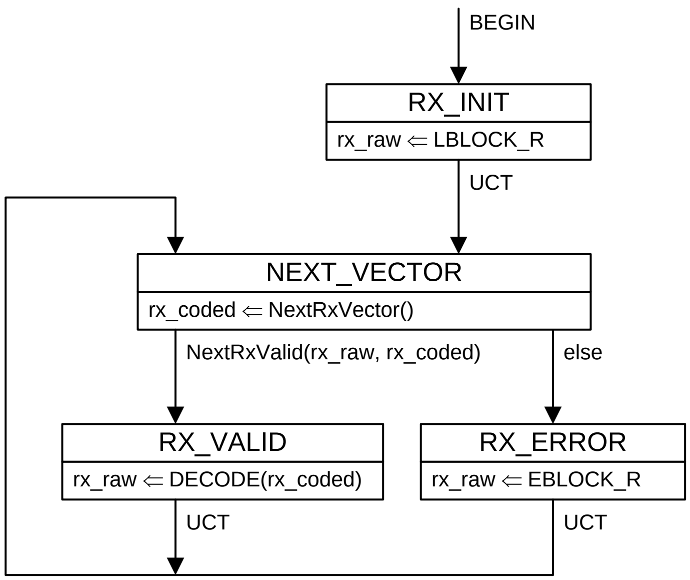

Figure 142- 4 Receive/Decode State Diagram.