101.3.2.3.4 Data Detector process within CNU (upstream)

The {EPoC_PMD_Name} CNU PCS transmit path includes the Data Detector process. This process contains a delay line (represented by the *FIFO_FEC_TX* buffer) that stores 66-bit blocks received from the output of the 64B/66B encoder to allow insertion of the FEC parity data into the transmitted data stream. In addition to inserting the FEC parity data into the data stream, the Data Detector process in the {EPoC_PMD_Name} CNU PCS generates the PMA_SIGNAL.request(tx_enable) primitive to turn the transmitter on and off at the correct times.

Upon initialization, the ONU transmitter is turned off. When the first 66-bit block containing data arrives at the *FIFO_FEC_TX* buffer, the Data Detector process in the {EPoC_PMD_Name} CNU PCS sets the PMA_SIGNAL.request(tx_enable) primitive to the value ON, instructing the PMD sublayer to start the process of turning the transmitter on.

When the *FIFO_FEC_TX* buffer becomes empty (i.e., contains only 66-bit blocks with Idle control characters), the Data Detector process in the {EPoC_PMD_Name} CNU PCS sets the PMA_SIGNAL.request(tx_enable) primitive to the value OFF, instructing the PMD sublayer to start the process of turning the transmitter off.

Between individual packets, 66-bit blocks with Idle control character arrive at the *FIFO_FEC_TX* buffer. If the number of these 66-bit blocks with Idle control characters is insufficient to fill the *FIFO_FEC_TX* buffer completely, then the transmitter is not turned off.

The length of the *FIFO_FEC_TX* buffer at the {EPoC_PMD_Name} CNU PCS shall be set such that the delay introduced by the *FIFO_FEC_TX* buffer together with any delay introduced by the PMA sublayer is long enough to turn the transmitter on and to allow transmission of any additional burst elements, such as TBD.

Figure 101–1 illustrates the details of the {EPoC_PMD_Name} 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).

The CNU burst transmission begins with the 65-bit long Start of Burst delimiter (*burstStart* constant, see TBD), which facilitates the detection of the start of a newly incoming data burst. When received at the CLT, the Start of Burst delimiter simplifies allows the FEC codeword alignment for the incoming data stream, even in the presence of bit errors. The Start of Burst delimiter is not part of the first FEC codeword.

The Start of Burst delimiter is followed by the 65-bit long FEC Selector delimiter (*burstFecSelector* constant, see TBD), which identifies the specific FEC code used by the CNU to encode data in the given burst. The FEC Selector delimiter is not part of the first FEC codeword.

The CNU burst ends with the 65-bits long End of Burst delimiter (*burstEnd* constant, see TBD), which facilitates the detection of the end of the current data burst. When received at the CLT, the End of Burst delimiter allows for the rapid reset of the CLT FEC synchronizer, so that it can search for the next burst. The End of Burst delimiter is not part of the last FEC codeword.

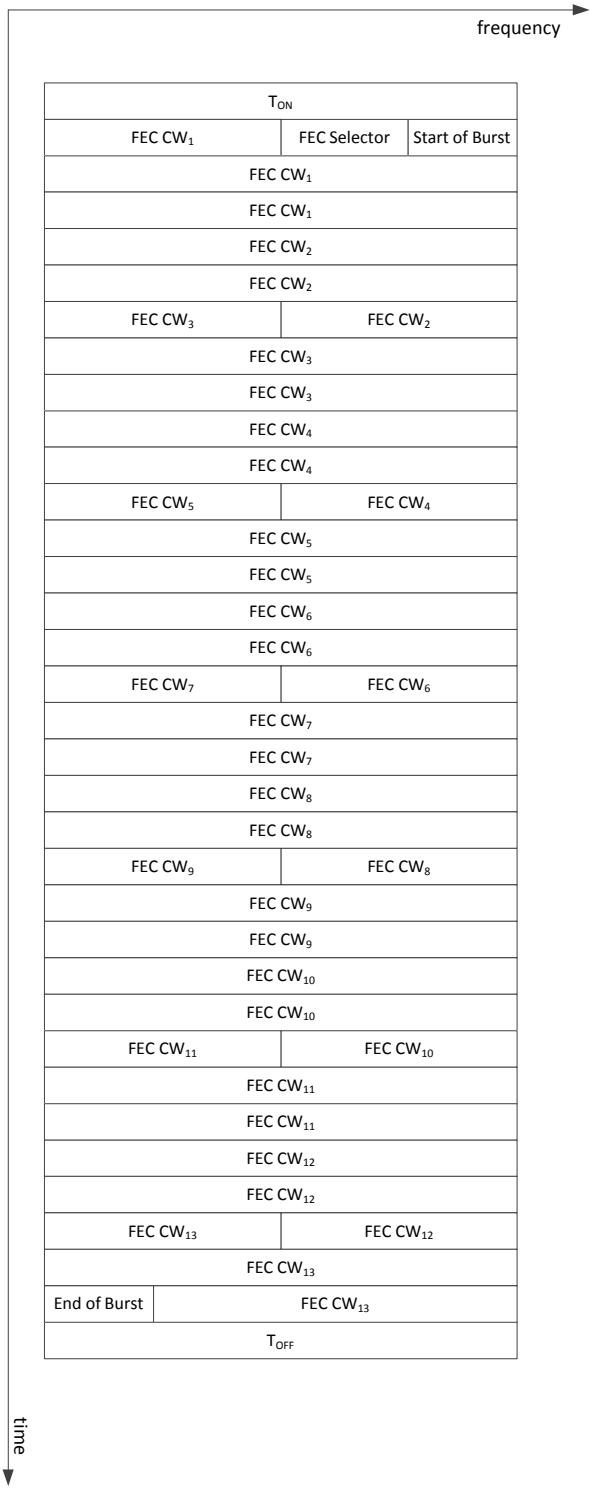

Figure 101–1—Details of CNU burst structure

101.3.2.3.5 LDPC Encode process within CNU (upstream)

The process of padding FEC codewords and appending FEC parity octets in the {EPoC_PMD_Name} CNU PCS transmit path is illustrated in Figure 101–8.

The 64B/66B encoder produces a stream of 66-bit blocks, which are delivered to the FEC Encode and Data Detector input process, as shown in Figure 101–8. The FEC Encode and Data Detector input process accumulates B_Q (see Table 101-6) of these 66-bit blocks to form the payload of a FEC codeword, removing the redundant first bit (i.e., sync header bit <0>) in each 66-bit block received from the 64B/66B encoder. The first bit <0> of the sync header in the 66-bit block in the transmit direction is guaranteed to be the complement of the second bit <1> of the sync header – see 49.2.4.3 for more details. Only one of the FEC codes defined in Table 101-6 is active at any time, as selected by register TBD.

Next, the FEC encoder calculates CRC40 (see 101.3.2.3.6) over the aggregated B_Q 65-bit blocks, placing the resulting 40 bits of CRC40 code immediately after the B_Q 65-bit blocks, forming the payload of the FEC codeword. Finally, the FEC encoder prepends B_P (see Table 101-6) padding bits (with the binary value of "0") to the payload of the FEC codeword as shown in Figure 101–8.

This resulting data is then LDPC-encoded, resulting in the F_R (see Table 101-6) bit of parity data. The first 25 bits of parity data are inserted into the 65-bit block carrying CRC40 code, complementing it. The remaining F_R-25 bits of parity data is then divided into C_Q (see Table 101-6) 65-bit blocks. Note that 65-bit blocks carrying CRC40 data and parity data do not include sync header. The last 65-bit block of the parity data contains C_{PL} (see Table 101-6) bits of parity data, and the remaining C_P (see Table 101-6) bits are filled with padding (with the binary value of "0").

101.3.2.3.6 LDPC codeword transmission order within CNU (upstream)

{the content of this subclause ought to be quite similar with the content of 101.3.2.3.5}

101.3.2.3.7 CRC40

{the content of this subclause will provide details about CRC40 used in EPoC to guarantee MTTFPA}

101.3.2.3.8 State diagrams

101.3.2.3.8.1 Constants

B_P

VALUE: see Table 101–5 for downstream FEC, Table 101–6 for upstream FEC
This constant represents the number of padding bits within the payload portion of the FEC codeword.

B_Q

VALUE: see Table 101–5 for downstream FEC, Table 101–6 for upstream FEC
This constant represents the number of 65-bit blocks within the payload portion of the FEC codeword.

C_P

VALUE: see Table 101–5 for downstream FEC, Table 101–6 for upstream FEC
This constant represents the number of padding bits within the last 65-bit block of the parity portion of the FEC codeword.

C_Q

VALUE: see Table 101–5 for downstream FEC, Table 101–6 for upstream FEC

This constant represents the number of 65-bit blocks within the parity portion of the FEC codeword.

F_P

VALUE: see Table 101–5 for downstream FEC, Table 101–6 for upstream FEC
This constant represents the number of bits within the payload portion of the FEC codeword.

F_R

VALUE: see Table 101–5 for downstream FEC, Table 101–6 for upstream FEC
This constant represents the number of bits within the parity portion of the FEC codeword.

101.3.2.3.8.2 Variables

blockCount

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

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 $F_R + C_P$ bits and filled with the binary value of "0".

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.

loc

TYPE: 16-bit unsigned integer
This variable represents the position within the given bit array.

SH_CTRL

See 76.3.2.5.2

SH_DATA

See 76.3.2.5.2

sizeFifo

TYPE: 16-bit unsigned integer
This variable represents the number of 65-bit blocks stored in the FIFO.

tx_coded<65:0>

TYPE: 66-bit block
This 66-bit block contains 64B/66B encoded data from the output of 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<64:0>

TYPE: 65-bit block

This 65-bit block contains the output of the FEC encoder being passed towards the Data Detector. The left-most bit is tx_coded_out<0> and the right-most bit is tx_coded_out<64>.

101.3.2.3.8.3 Functions

calculateCrc (ARRAY_IN)

This function calculates CRC40 for data included in ARRAY_IN.

calculateParity(ARRAY_IN)

This function calculates LDPC parity (for the code per Table 101–5 or Table 101–6) for data included in ARRAY_IN.

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 --
}
```

101.3.2.3.8.4 Messages

TBD

101.3.2.3.8.5 State diagrams

The CLT PCS shall implement the LDPC encoding process, comprising the input process as shown in Figure 101–2 and the output process as shown in Figure 101–3. The CNU PCS shall implement the LDPC encoding process, comprising the input process as shown in Figure 101–2 and the output process as shown in Figure 101–3.

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

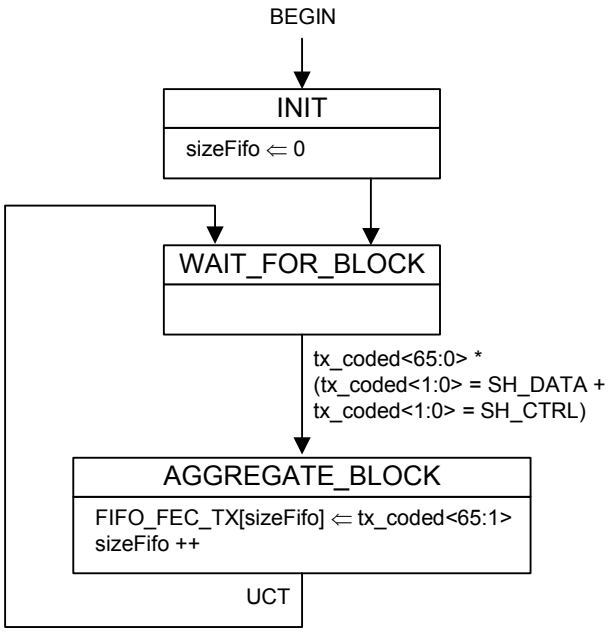

Figure 101–2—FEC encoder, input process state diagram

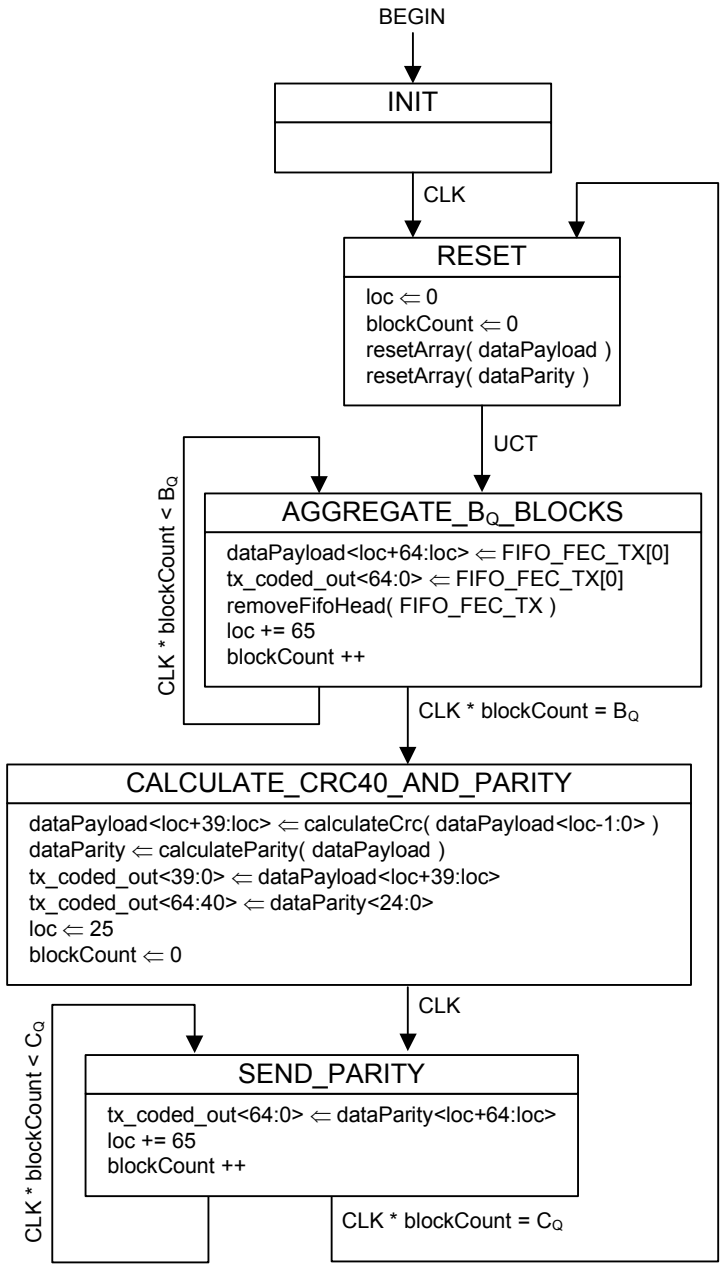

Figure 101–3—FEC encoder, output process state diagram (CLT)