101.3 Physical Coding Sublayer (PCS) for EPoC

This subclause will be modelled after 76.3 for 10G-EPON, with all the necessary changes for EPoC, e.g., changing FEC definition structure, presence of line coding and its type, scrambling / interleaving. The current structure is just first order approximation and will be modified as more contributions for PCS structure and functions arrive.

101.3.1 Overview

This subclause defines the Physical Coding Sublayer (PCS) for {EPoC_PMD_NAME}, supporting TDD and FDD mode 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, in the function of the assigned RF spectrum.

This subclause also specifies a forward error correction (FEC) mechanism to increase the available link budget and the Idle control character insertion and Idle control character deletion mechanisms - part of the data rate adaptation function combining the MAC and MAC Control Clients operating at 10 Gb/s with EPoC PCS and PMD layers operating at the data rates below 10 Gb/s.

{Figure 101-X} shows the relationship between the EPoC PCS sublayer and the ISO/IEC OSI reference model.

101.3.1.1 EPoc_PMD_Name PCS

The EPoC PCS extends the 10GBASE-PR PCS described in {Clause 76} to support TDD and FDD mode of operation over the point-to-multipoint coaxial medium architecture. Figure 101–1 illustrates the functional block diagram of the downstream path in the EPoC PCS operating in FDD mode, Figure 101–2 illustrates the functional block diagram of the downstream path in the EPoC PCS operating in FDD mode, and Figure 101–3 shows the functional block diagram of the upstream path in the EPoC PCS for both the TDD and FDD modes.

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

101.3.2.1 LDPC codes

The {EPoC_PMD_Name} encodes the transmitted data using a systematic LDPC (F_C, F_P) code. A LDPC encoder encodes F_P information bits $i_0 \dots i_{F_P-1}$ into a codeword

$$c = (i_0, \dots, i_{F_P-1}, p_{F_P}, \dots, p_{F_C-1})$$

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

$$Hc^T = 0$$

where H is an $F_R \times F_C$ binary matrix containing mostly ‘0’ and relatively few ‘1’, called low-density parity-check matrix. (see [1] and [2]). The detailed description of such parity check matrices is given in 101.3.2.2.

{to be included in informative references: [1] R. G. Gallager, “Low density parity check codes,” IRE Trans. Inform. Theory, vol. IT-8, pp. 21–28, Jan. 1962.; [2] T. Richardson and R. Urbanke, “Modern Coding Theory," Cambridge University Press, 2008}

The CLT {EPoC_PMD_Name} PCS operating on amplified CCDN shall encode the transmitted data using the LDPC (F_C, F_P) code per Table 101–1. The CNU {EPoC_PMD_Name} PCS operating on amplified CCDN shall encode the transmitted data using LDPC (F_C, F_P) codes per Table 101–2.

The CLT {EPoC_PMD_Name} PCS operating on amplified CCDN shall decode the received data using one of the LDPC (F_C, F_P) codes per Table 101–2. The CNU {EPoC_PMD_Name} PCS operating on amplified CCDN shall decode the received data using the LDPC (F_C, F_P) code per Table 101–1.

Table 101–1—LDCP codes used by the CLT {EPoC_PMD_Name} PCS for amplified CCDN

Codeword F_C [bits]	Payload F_P [bits]	Parity F_R [bits]	Payload			Parity		
			65-bit blocks B_Q	CRC bits	Padding bits B_P	65 -bit blocks C_Q	Parity bits in last block C_{PL}	Padding bits C_P
16200	14400	1800	220	40	60	28	20	45

Table 101–2—LDCP codes used by the CLT {EPoC_PMD_Name} PCS for amplified CCDN

Codeword F_C [bits]	Payload F_P [bits]	Parity F_R [bits]	Payload			Parity		
			65-bit blocks B_Q	CRC bits	Padding bits B_P	65-bit blocks C_Q	Parity bits in last block C_{PL}	Padding bits C_P
16200	14400	1800	220	40	60	28	20	45
5940	5040	900	76	40	60	14	30	35
1120	840	280	12	40	20	4	60	5

Annex 101A gives an example of LDPC (F_C, F_P) FEC encoding. {we will need to select one of the codes from the family of codes we use in either downstream or upstream and then generate examples}

Annex 101B gives an example of LDPC (F_C, F_P) FEC decoding. {we will need to select one of the codes from the family of codes we use in either downstream or upstream and then generate examples}

101.3.2.2 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:

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. In this specification, the sub-matrix size L is called the lifting factor.

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

In this specification, the sub-matrix $H_{i,j}$ is represented by a value in {-1, 0,…, L-1}, where a '-1' value represents an all-zero submatrix, 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 101–1 for downstream and Table 101–2 for upstream, respectively. The lifting factor of the matrix is L=360.

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

Columns	Rows				
	1	2	3	4	5
1	93	274	134	-1	253
2	271	115	355	-1	273
3	-1	329	175	184	90
4	83	338	24	70	-1
5	26	124	253	247	-1
6	208	-1	242	14	151
7	245	293	-1	22	311
8	200	-1	187	7	320
9	-1	69	94	285	339
10	175	64	26	54	-1
11	331	342	87	-1	295
12	17	-1	302	352	148
13	86	88	-1	26	48
14	-1	139	191	108	91
15	337	-1	323	10	62
16	-1	137	22	298	100
17	238	212	-1	123	232
18	81	-1	245	139	146
19	-1	157	294	117	200
20	307	195	240	-1	135

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

Columns	Rows				
	1	2	3	4	5
21	-1	357	84	336	12
22	165	81	76	49	-1
23	-1	194	342	202	179
24	47	1	345	359	-1
25	76	159	174	342	-1
26	73	56	269	-1	232
27	150	72	329	224	-1
28	349	126	-1	106	21
29	139	277	214	-1	331
30	331	156	-1	273	313
31	118	32	-1	177	349
32	345	111	-1	245	34
33	27	175	-1	98	97
34	294	-1	218	355	187
35	-1	306	104	178	38
36	145	224	40	176	-1
37	279	-1	197	147	235
38	97	206	73	-1	52
39	106	-1	229	280	170
40	160	29	63	-1	58
41	143	106	-1	-1	-1
42	-1	334	270	-1	-1
43	-1	-1	72	221	-1
44	-1	-1	-1	208	257
45	-1	-1	-1	-1	0

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

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

Columns	Rows				
	1	2	3	4	5
29	11	101	-1	-1	-1
30	-1	171	34	-1	-1
31	-1	-1	74	23	-1
32	-1	-1	-1	8	177
33	-1	-1	-1	-1	19

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

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

101.3.3 PCS transmit function

In the CLT, the PCS transmit function operates in a continuous (FDD mode) or burst (TDD mode) fashion at the data rate of up to 10 Gb/s, depending on the allocated RF spectrum and the configured operation mode. In the CNU, the PCS transmit function operates in a burst fashion (TDD and FDD modes) at the data rate of up to 10 Gb/s, depending on the allocated RF spectrum and the configured operation mode. Figure 101–1 illustrates the transmit direction of CLT PCS operating in FDD mode, Figure 101–2 illustrates the transmit direction of CLT PCS operating in TDD mode, and Figure 101–3 illustrates the transmit direction of the CNU PCS.

The EPoC PCS includes a mandatory FEC in the transmit direction, along with 64B/66B encoder as well as an Idle control character deletion function performing data rate adaptation and FEC overhead compensation functions.

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

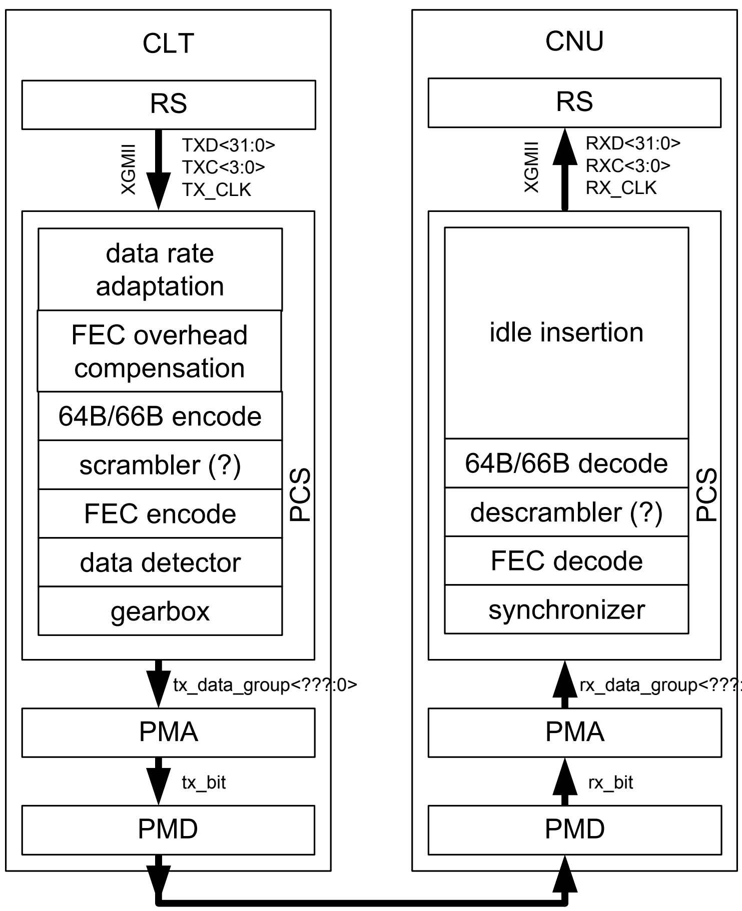

Figure 101–1—EPoC PCS functional block diagram, downstream path for FDD mode

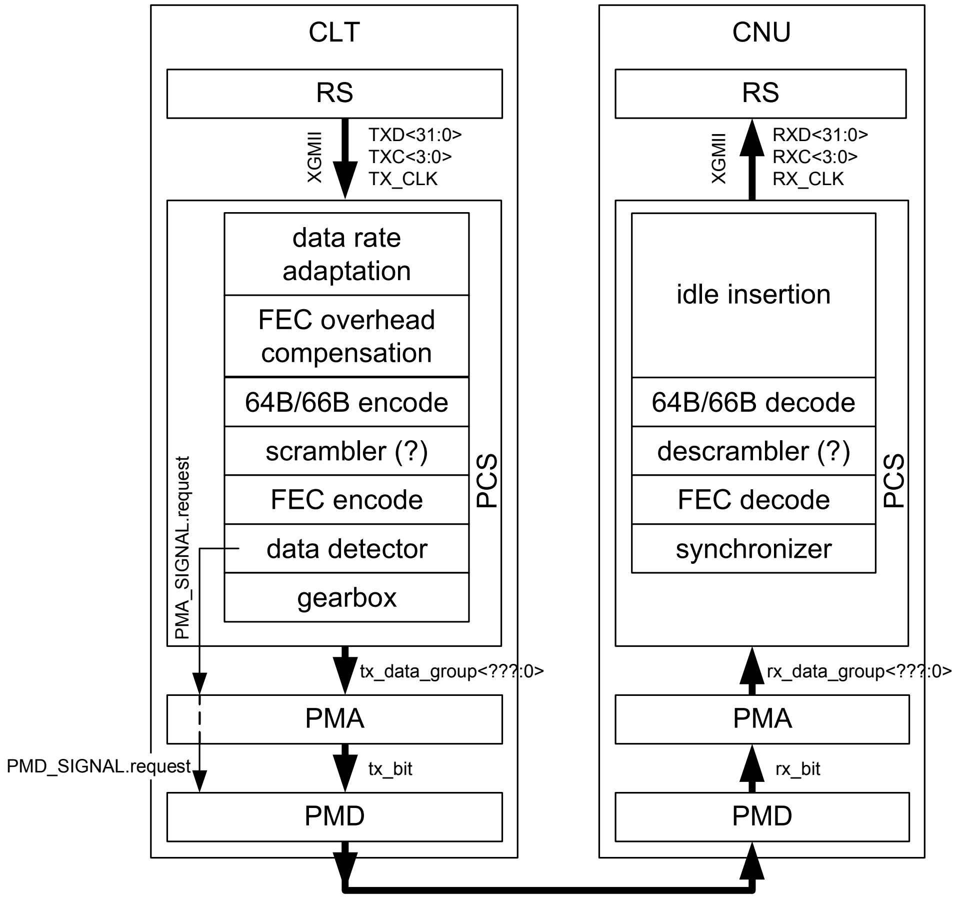

Figure 101–2—EPoC PCS functional block diagram, downstream path for TDD mode

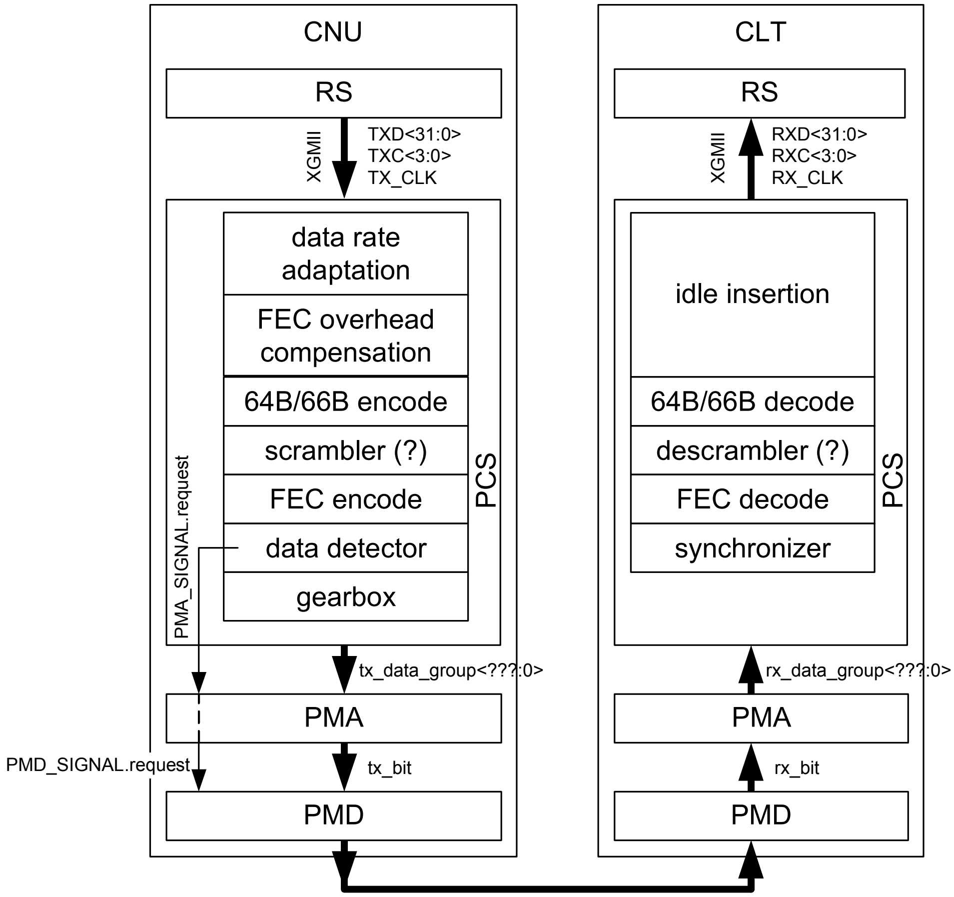

Figure 101–3—EPoC PCS functional block diagram, upstream path

101.3.3.1 Idle control character deletion process

In the transmitting PCS, the Idle control character deletion process is responsible for deleting excess Idle control characters inserted in between individual frames to adjust the data rate enforced by the MAC Control (as defined in {Clause 102}) to the effective data rate supported by the PCS and PMD. The gaps created within the data stream by the operation of the Idle control character 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 data rate between the MAC and PHY, while maintaining the effective data rate unchanged (data rate adaptation sub-process).

The Idle control character 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 control character deletion process is composed of two sub-processes executed in the following order:

a) data rate adaptation sub-process, where the PCS discards a specific number of excess Idle control characters to decrease the data rate to match the effective data rate supported by the EPoC PMD; at the output of the data rate adaptation sub-process, the data stream still contains excess Idle control characters; and
b) FEC overhead compensation sub-process, where the PCS discards the remaining excess Idle control characters to prepare space in the de-rated data stream for PHY parity data; at the output of the FEC overhead compensation sub-process, the data stream does not contain any excess Idle control characters.

The operation of the EPoC MPCP defined in {Clause 102} ensures that a sufficient number of excess 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.

101.3.3.1.1 Constants

FEC_DSize

TYPE: 16-bit unsigned integer
The number of 72-bit vectors constituting the payload portion of a FEC codeword. To normalize pre-FEC data rate, the Idle control character deletion process removes FEC_OSize vectors per every FEC_DSize vectors transferred to the 64B/66B encoder.
Value: {TBD}

FEC_OSize

TYPE: 16-bit unsigned integer
The number of 72-bit vectors constituting the parity (overhead) portion of a FEC codeword. To normalize pre-FEC data rate, the Idle control character deletion process removes FEC_OSize vectors per every FEC_DSize vectors transferred to the 64B/66B encoder.
Value: {TBD}

Note that the list of constants 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.

101.3.3.1.2 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.

delayBound

TYPE: 16-bit unsigned integer

This value represents the delay sufficient to initiate the transmitter at the CNU and to stabilize the receiver at the CLT (i.e., the maximum FIFO size expressed in units of 66-bit blocks). The value of delayBound includes {to be added when the burst structure is known}. This variable is used only by the CNU.

PHY_DSize

TYPE: 16-bit unsigned integer

The number of 72-bit vectors constituting (together with PHY_OSize) the denominator in the EPoC PCS de-rating Equation (101–1). To normalize the effective PCS data rate, the Idle control character deletion process removes PHY_OSize vectors per every PHY_DSize vectors transferred to the FEC overhead compensation sub-process.

Value: {TBD, reference how it is calculated ?}

$$\text{PCS\_Rate} = \text{XGMII\_Rate} \times \frac{\text{PHY\_DSize}}{\text{PHY\_DSize} + \text{PHY\_OSize}} \tag{101–1}$$

PHY_OSize

TYPE: 16-bit unsigned integer

The number of 72-bit vectors constituting the numerator in the EPoC PCS de-rating Equation (101–1). To normalize the effective PCS data rate, the Idle control character deletion process removes PHY_OSize vectors per every PHY_DSize vectors transferred to the FEC overhead compensation sub-process.

Value: {TBD, reference how it is calculated ?}

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 control character deletion process to the 64B/66B encoder. This vector contains two XGMII transfers mapped as shown for tx_raw<71:0>.

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.

101.3.3.1.3 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.

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.

101.3.3.1.4 Functions

T_TYPE(tx_raw<71:0>)

This function is defined in {49.2.13.2.3}.

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.

101.3.3.1.5 State diagrams

The CLT PCS shall perform the Idle control character deletion process as shown in Figure 101–4 (data rate adaptation sub-process) and in Figure 101–5 (FEC overhead compensation sub-process), in the order shown in {Figure 101-X1}. The CNU PCS shall perform the Idle control character deletion process as shown in Figure 101–6 (data rate adaptation sub-process) and in Figure 101–7 (FEC overhead compensation sub-process), in the order shown in {Figure 101-X1}. In case of any discrepancy between state diagrams and the descriptive text, the state diagrams prevail.

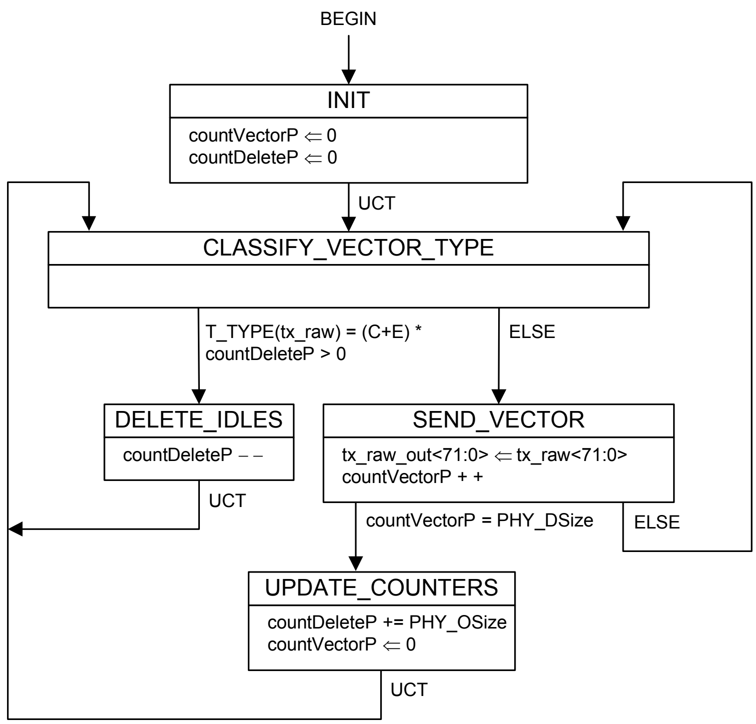

Figure 101–4—CLT Idle control character deletion process (data rate adaptation sub-process)

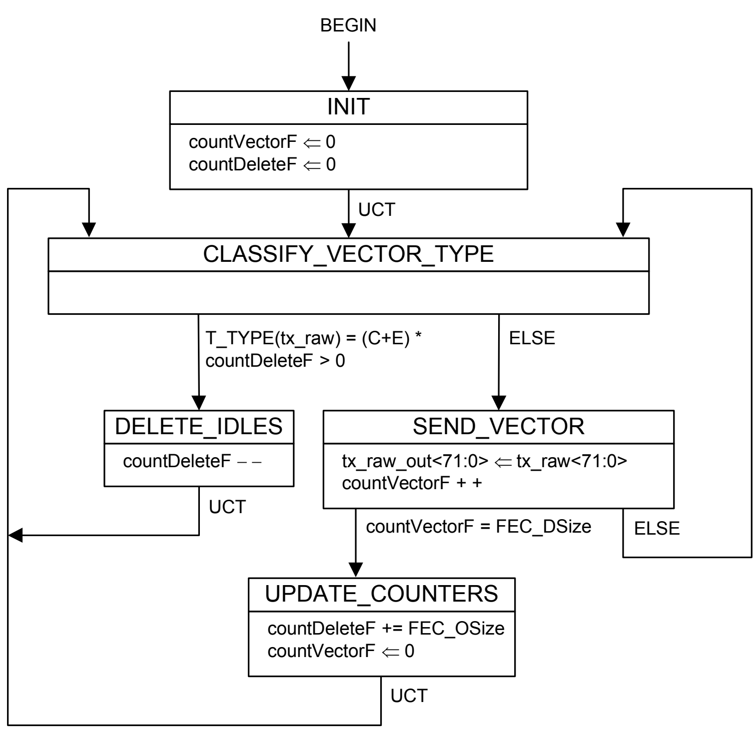

Figure 101–5—CLT Idle control character deletion process (FEC overhead compensation sub-process)

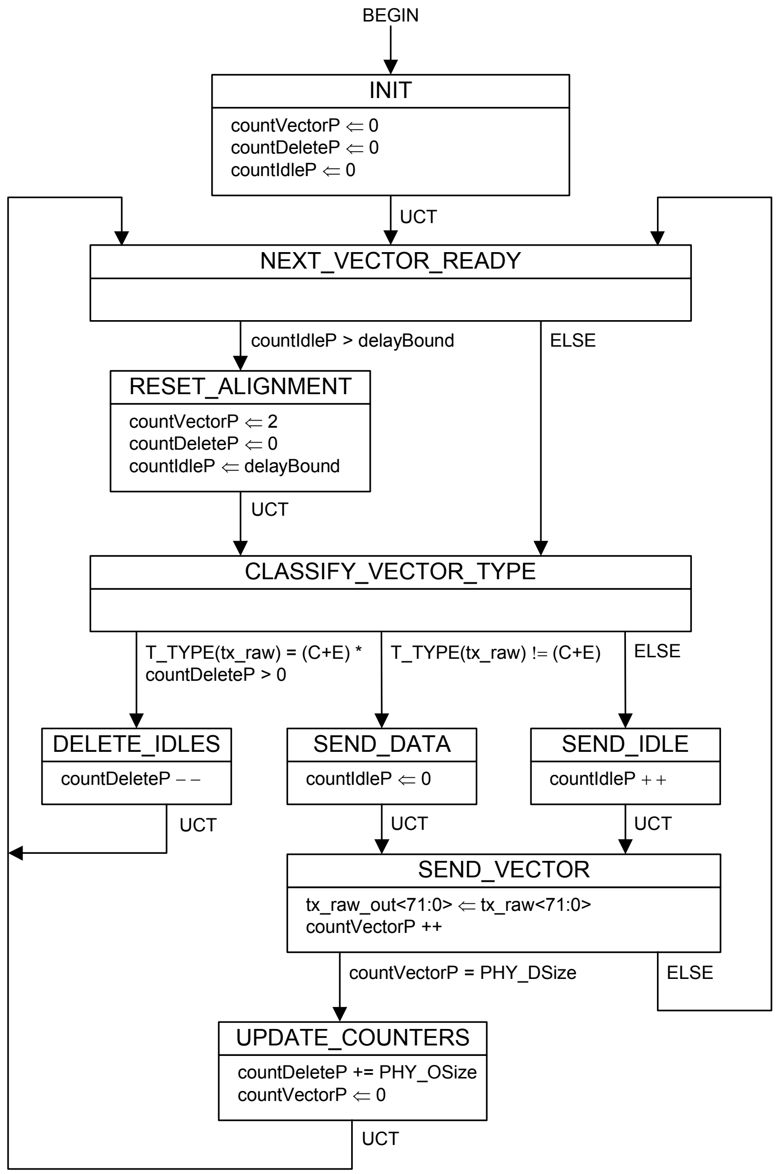

Figure 101–6—CNU Idle control character deletion process (data rate adaptation sub-process)

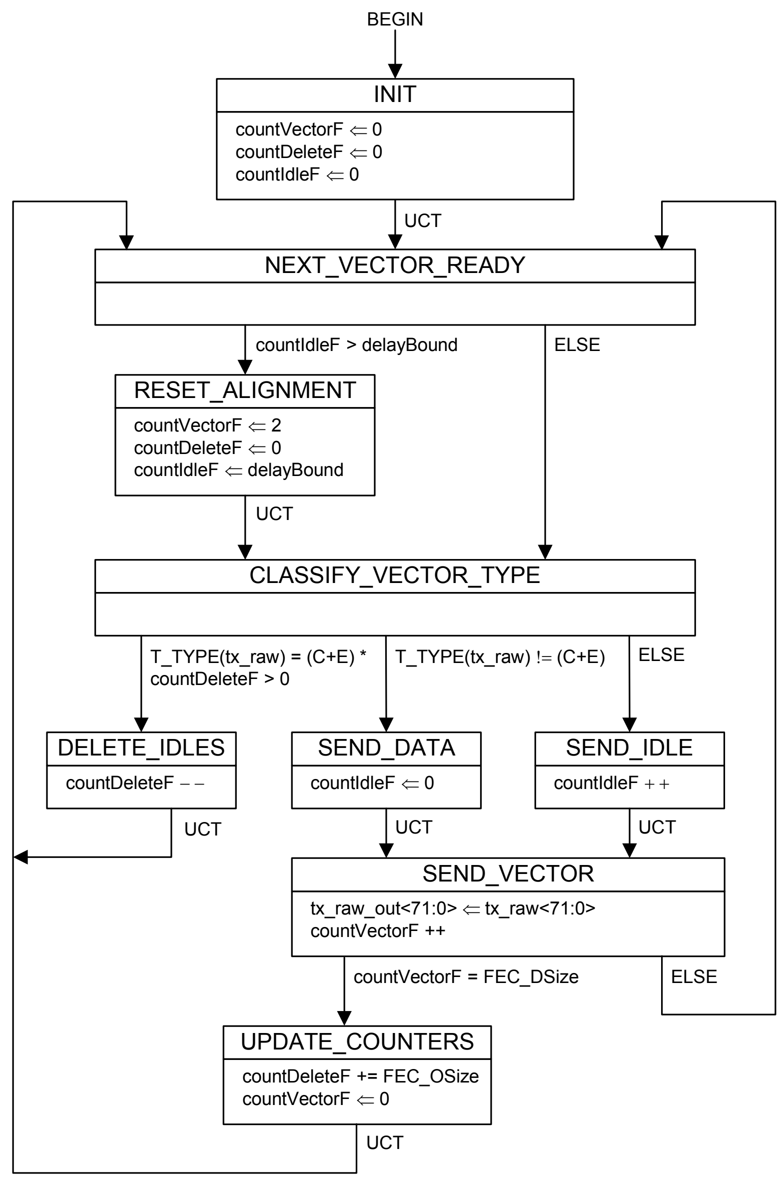

Figure 101–7—CNU Idle control character deletion process (FEC overhead compensation sub-process)

Note that Figure 101–4, Figure 101–6, Figure 101–6, and Figure 101–7 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.

101.3.3.2 64B/66B Encode

The 64B/66B encoder shall perform the functions specified in {Figure 49–16}. The 64B/66B encoding process is as described in {49.2.4}, with the following exceptions:

a) the 64B/66B encode process in the EPoC PCS operates on 72-bit vectors obtained from the output of the Idle control character deletion process (see 101.3.3.1), rather than directly from the XGMII; and

b) the 64B/66B encode process in the EPoC PCS operates on bursty data stream produced by the Idle control character deletion process, unlike in 10GBASE-R PCS, where data stream to the input of the 64B/66B encoder is taken directly from the XGMII and hence continuous.

101.3.3.3 FEC encoding process (FDD)

The {EPoC_PMD_Name} 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 $i_0 \cdots i_{F_P-1}$ into a codeword

$$c = (i_0, \ldots, i_{F_P-1}, p_{F_P}, \ldots, p_{F_C-1})$$

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

$$Hc^T = 0$$

where H is an $F_R \times F_C$ binary matrix containing mostly '0' and relatively few '1', called low-density parity-check matrix. (see [1] and [2]). The detailed description of such parity check matrices is given in 101.3.2.2.

{to be included in informative references: [1] R. G. Gallager, "Low density parity check codes," IRE Trans. Inform. Theory, vol. IT-8, pp. 21–28, Jan. 1962.; [2] T. Richardson and R. Urbanke, "Modern Coding Theory," Cambridge University Press, 2008}

The CLT {EPoC_PMD_Name} PCS operating on amplified CCDN shall encode the transmitted data using one of the LDPC (F_C, F_P) codes per Table 101–6, as selected using register TBD. The CNU {EPoC_PMD_Name} PCS operating on amplified CCDN shall encode the transmitted data using one of the LDPC (F_C, F_P) codes per Table 101–7, as selected using register TBD.

Table 101–6—LDCP codes used by the CLT {EPoC_PMD_Name} PCS for amplified CCDN

Codeword F_C [bits]	Payload F_P [bits]	Parity F_R [bits]	Payload			Parity		
			65-bit blocks B_Q	CRC bits	Padding bits B_P	65-bit blocks C_Q	Parity bits in last block C_{PL}	Padding bits C_P
16200	14400	1800	220	40	60	28	20	45

Annex 101A gives an example of LDPC (F_C, F_P) FEC encoding. {we will need to select one of the codes from the family of codes we use in either downstream or upstream and then generate examples}

Table 101–7—LDCP codes used by the CLT {EPoC_PMD_Name} PCS for amplified CCDN

Codeword F_C [bits]	Payload F_P [bits]	Parity F_R [bits]	Payload			Parity		
			65-bit blocks B_Q	CRC bits	Padding bits B_P	65-bit blocks C_Q	Parity bits in last block C_{PL}	Padding bits C_P
16200	14400	1800	220	40	60	28	20	45
5940	5040	900	76	40	60	14	30	35
1120	840	280	12	40	20	4	60	5

101.3.3.3.1 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:

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

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. In this specification, the sub-matrix size L is called the lifting factor.

In this specification, the sub-matrix $H_{i,j}$ is represented by a value in {-1, 0,…, L-1}, where a '-1' value represents an all-zero submatrix, 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 101–8 presents a 5 × 45 base matrix of the low-density parity-check matrix H for LDPC (16200, 14400) code listed in Table 101–6 for downstream and Table 101–7 for upstream, respectively. The lifting factor of the matrix is L=360.

Table 101–9 presents a 5 × 33 base matrix of the low-density parity-check matrix H for LDPC (5940, 5040) code listed in Table 101–7 for upstream. The lifting factor of the matrix is L=180.

Table 101–10 presents a 5 × 20 base matrix of the low-density parity-check matrix H for LDPC (1120, 840) code listed in Table 101–7 for upstream. The lifting factor of the matrix is L=56.

101.3.3.3.2 LDPC encoding process within CLT (downstream)

The process of padding FEC codewords and appending FEC parity octets in the {EPoC_PMD_Name} CLT transmitter is illustrated in Figure 101–8.

Table 101–8—LDPC (16200, 14400) code matrix

Columns	Rows				
	1	2	3	4	5
1	93	274	134	-1	253
2	271	115	355	-1	273
3	-1	329	175	184	90
4	83	338	24	70	-1
5	26	124	253	247	-1
6	208	-1	242	14	151
7	245	293	-1	22	311
8	200	-1	187	7	320
9	-1	69	94	285	339
10	175	64	26	54	-1
11	331	342	87	-1	295
12	17	-1	302	352	148
13	86	88	-1	26	48
14	-1	139	191	108	91
15	337	-1	323	10	62
16	-1	137	22	298	100
17	238	212	-1	123	232
18	81	-1	245	139	146
19	-1	157	294	117	200
20	307	195	240	-1	135
21	-1	357	84	336	12
22	165	81	76	49	-1
23	-1	194	342	202	179
24	47	1	345	359	-1
25	76	159	174	342	-1
26	73	56	269	-1	232
27	150	72	329	224	-1
28	349	126	-1	106	21
29	139	277	214	-1	331
30	331	156	-1	273	313
31	118	32	-1	177	349
32	345	111	-1	245	34

Table 101–8—LDPC (16200, 14400) code matrix *(continued)*

Columns	Rows				
	1	2	3	4	5
33	27	175	-1	98	97
34	294	-1	218	355	187
35	-1	306	104	178	38
36	145	224	40	176	-1
37	279	-1	197	147	235
38	97	206	73	-1	52
39	106	-1	229	280	170
40	160	29	63	-1	58
41	143	106	-1	-1	-1
42	-1	334	270	-1	-1
43	-1	-1	72	221	-1
44	-1	-1	-1	208	257
45	-1	-1	-1	-1	0

Table 101–9—LDCP (5940, 5040) code matrix

Columns	Rows				
	1	2	3	4	5
1	142	54	63	28	52
2	158	172	11	160	159
3	113	145	112	102	75
4	124	28	114	44	74
5	92	55	61	8	46
6	44	19	123	84	71
7	93	159	72	126	42
8	70	22	55	9	11
9	172	96	114	169	108
10	3	12	20	174	153
11	25	85	53	147	-1
12	44	-1	114	24	72
13	141	128	42	145	-1
14	160	5	33	-1	163

Table 101–9—LDCP (5940, 5040) code matrix *(continued)*

Columns	Rows				
	1	2	3	4	5
15	50	158	4	26	-1
16	45	120	66	-1	9
17	118	51	163	-1	2
18	84	171	50	-1	168
19	-1	65	46	67	158
20	64	141	17	82	-1
21	66	-1	175	4	1
22	97	42	-1	177	49
23	1	83	-1	151	89
24	115	7	-1	131	63
25	8	-1	92	139	179
26	108	39	-1	117	10
27	-1	121	41	36	75
28	-1	84	138	18	161
29	11	101	-1	-1	-1
30	-1	171	34	-1	-1
31	-1	-1	74	23	-1
32	-1	-1	-1	8	177
33	-1	-1	-1	-1	19

Table 101–10—LDCP (1120, 840) code matrix

Columns	Rows				
	1	2	3	4	5
1	5	0	12	0	36
2	14	35	28	51	6
3	12	1	22	16	3
4	1	26	46	31	51
5	2	0	3	13	4
6	37	10	16	39	19
7	45	16	51	27	4
8	26	16	2	33	45

Table 101–10—LDCP (1120, 840) code matrix *(continued)*

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

The 64B/66B encoder produces a stream of 66-bit blocks, which are then delivered to the FEC encoder. The FEC encoder 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.

Next, the FEC encoder calculates CRC40 (see) 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 bits 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 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} bits of parity data, and the remaining C_P bits are filled with padding (binary “0”).

101.3.3.3.3 LDPC codeword transmission order within CLT (downstream)

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 towards the Data Detector, one 65-bit block at a time. Note that the B_P padding bits used to generate the FEC codeword are not transmitted towards the Data Detector. The C_P padding bits in the last parity codeword (block number C_Q) are transmitted towards the Data Detector.

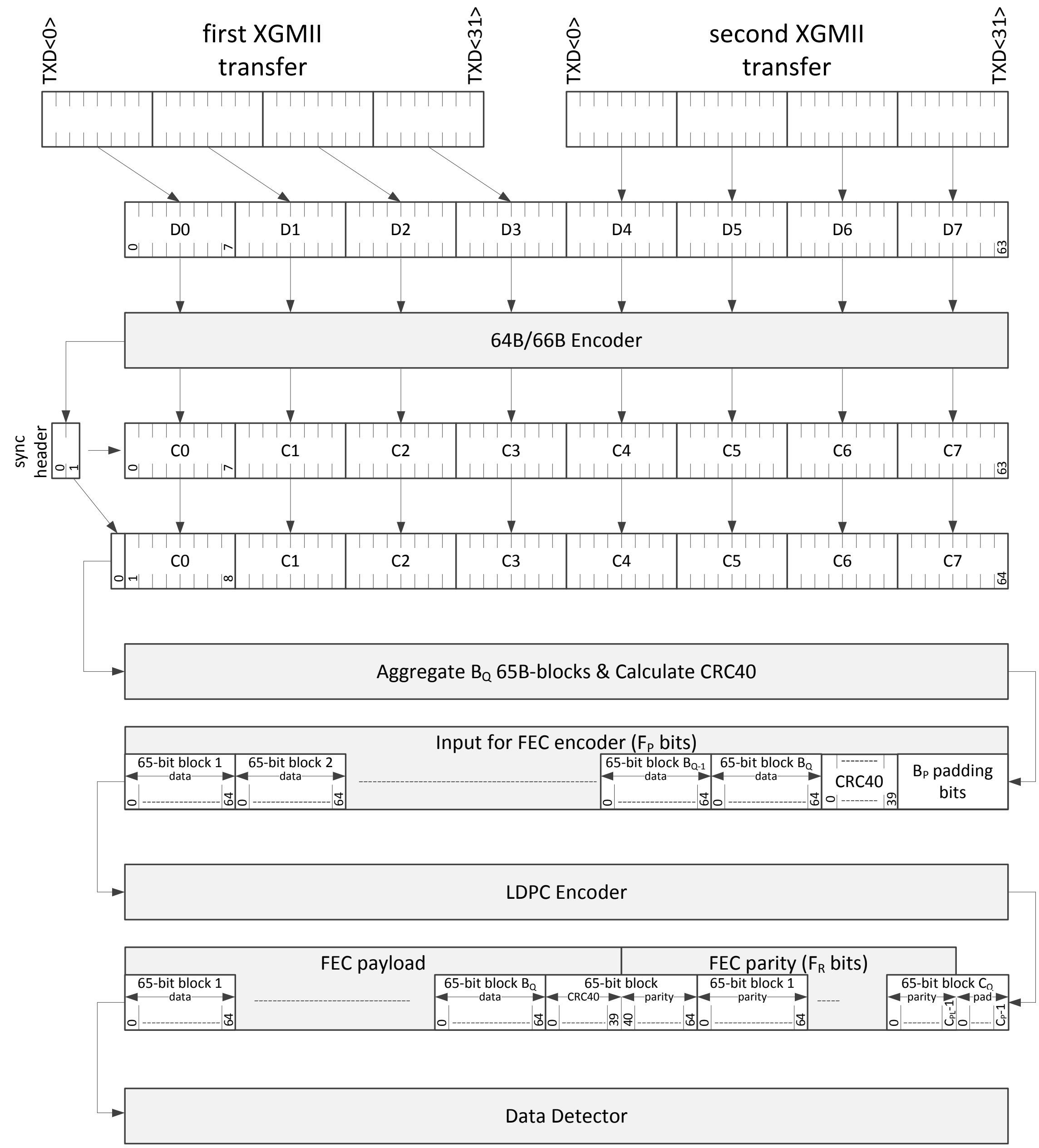

Figure 101–8—PCS Transmit bit ordering within CLT (downstream)

101.3.3.3.4 LDPC encoding process within CNU (upstream)

{the upstream FEC encoding for CNU will be described when we have a consistent proposal on how to mix three different FEC codes into a single transmission slot}

101.3.3.3.5 LDPC codeword transmission order within CNU (upstream)

{the content of this subclause ought to be quite similar with the content of 101.3.3.3.4}

101.3.3.3.6 CRC40

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

101.3.3.3.7 State diagrams

101.3.3.3.7.1 Constants

B_P
: VALUE: see Table 101–6 for downstream FEC, Table 101–7 for upstream FEC
: This constant represents the number of padding bits within the payload portion of the FEC codeword.

B_Q
: VALUE: see Table 101–6 for downstream FEC, Table 101–7 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 101–6 for downstream FEC, Table 101–7 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 101–6 for downstream FEC, Table 101–7 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 101–6 for downstream FEC, Table 101–7 for upstream FEC
: This constant represents the number of bits within the payload portion of the FEC codeword.

F_R
: VALUE: see Table 101–6 for downstream FEC, Table 101–7 for upstream FEC
: This constant represents the number of bits within the parity portion of the FEC codeword.

101.3.3.3.7.2 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>.

101.3.3.3.7.3 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 101–6 or Table 101–7) 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 --
}
```

101.3.3.3.7.4 Messages

TBD

101.3.3.3.7.5 State diagrams

The CLT PCS shall implement the LDPC encoding process, comprising the input process as shown in Figure 101–9 and the output process as shown in Figure 101–10. The CNU PCS shall implement the LDPC encoding process, comprising the input process as shown in Figure 101–9 and the output process as shown in Figure 101–10.

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

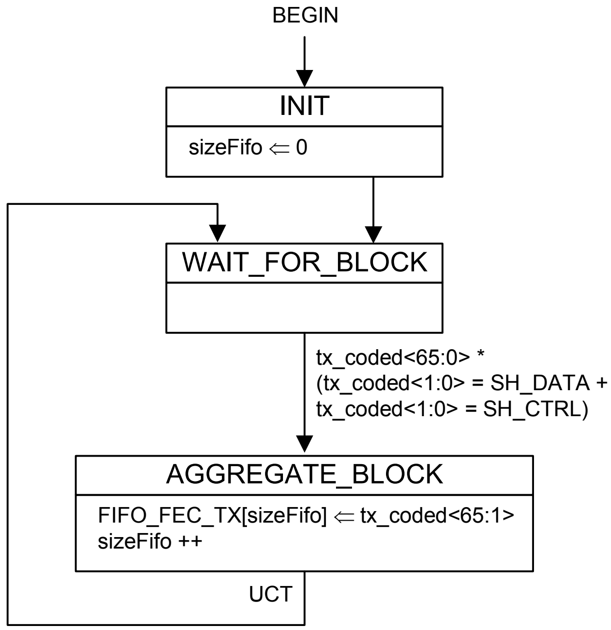

Figure 101–9—FEC encoder, input process state diagram

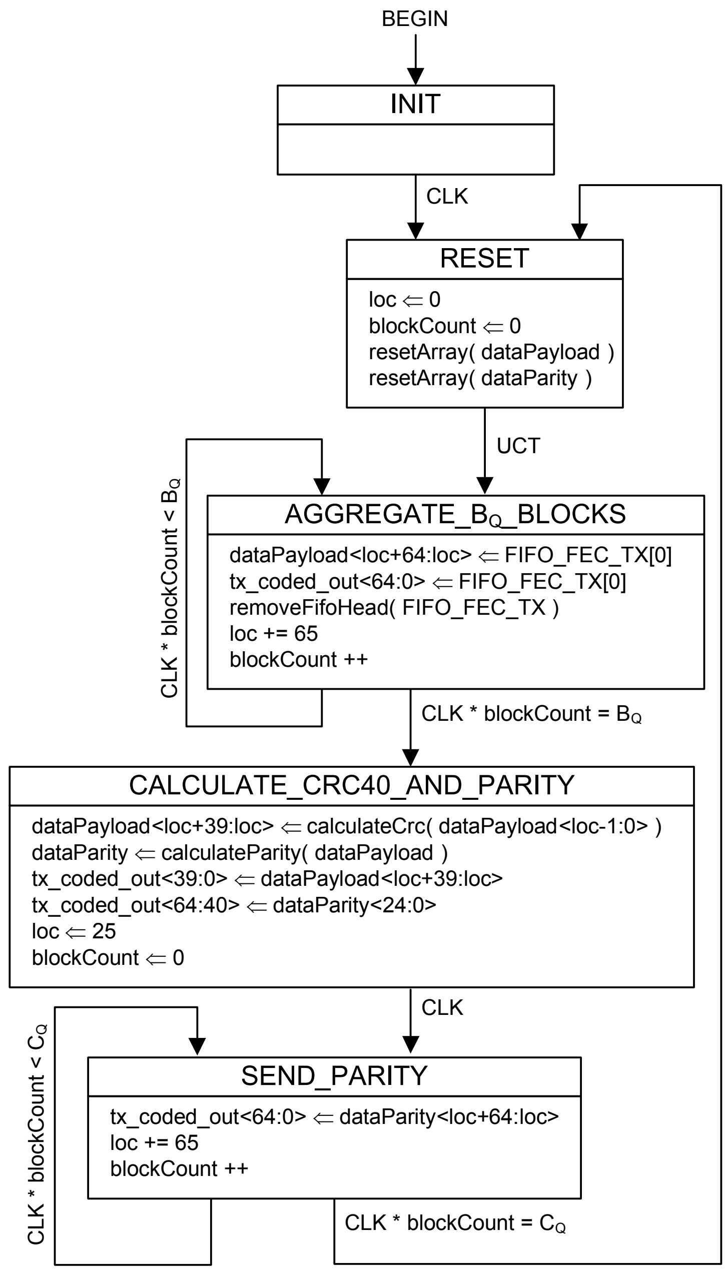

Figure 101–10—FEC encoder, output process state diagram (CLT)

101.3.4 PCS receive function

In the CLT, the PCS receive function operates in a burst fashion (for both FDD and TDD modes) at the data rate of up to 10 Gb/s, depending on the allocated RF spectrum and the configured operation mode. In the CNU, the PCS transmit function operates in a continuous (FDD mode) or burst (TDD mode) fashion at the data rate of up to 10 Gb/s, depending on the allocated RF spectrum and the configured operation mode. Figure 101–1 illustrates the receive direction of CNU PCS and Figure 101–3 illustrates the receive direction of the CLT PCS.

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.

101.3.4.1 FEC decoding process

~~The {EPoC_PMD_Name} decodes the received data using LDPC (F_C, F_P) code. The CLT {EPoC_PMD_Name} PCS operating on amplified CCDN shall decode the received data using one of the LDPC (F_C, F_P) codes per Table 101–2, as selected using register TBD. The CNU {EPoC_PMD_Name} PCS operating on amplified CCDN shall decode the received data using one of the LDPC (F_C, F_P) codes per Table 101–1, as selected using register TBD.~~

~~Annex 101B gives an example of LDPC (F_C, F_P) FEC decoding. {we will need to select one of the codes from the family of codes we use in either downstream or upstream and then generate examples}~~

101.3.4.1.1 LDPC decoding process within CLT (upstream)

{the upstream FEC decoding for CLT will be described when we have a consistent proposal on how to mix three different FEC codes into a single transmission slot}

101.3.4.1.2 LDPC decoding process within CNU (downstream)

The process of decoding FEC codewords in the {EPoC_PMD_Name} CNU receiver is illustrated in Figure 101–11.

{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}

Once the alignment to FEC codeword is found, the {EPoC_PMD_Name} CNU receiver aggregates the total of $B_Q + 1 + C_Q$ 65-bit blocks received from the PMA, forming the FEC payload (blocks number 1 to B_Q, and bits <0> through <39> from the following 65-bit block) and the FEC parity (bits <40> through <64> from the 65-bit block following payload portion of the FEC codeword and followed by blocks number 1 to C_Q) portions of the codeword. Note that the C_P padding bits in the last parity codeword (block number C_Q) are locally generated within the PMA and transmitted to the PCS.

Next, B_P padding bits are inserted immediately after the end of the CRC40 data, and then the last 65-bit block (number C_Q) of the parity portion of FEC codeword is truncated, removing the last C_{PL} bits, forming the input into the FEC decoder.

The FEC decoder produces the FEC payload portion of the codeword with the size of F_P (in bits), where bits $<F_P-B_P-1> \ldots <F_P-1>$ 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. The behavior of the FEC decoder in the presence of CRC40 code failure depends on status of the user-configurable option to indicate an uncorrectable FEC codeword.

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 in the CNU shall provide a user-configurable option to indicate an uncorrectable FEC codeword (due to an excess of symbols containing errors) to higher layers. If this user-configurable option is enabled 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 all B_Q blocks with the binary value of "11". If this user-configurable option is disabled, the FEC decoder does not make any further changes to the sync headers in all B_Q blocks.

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.

101.3.4.1.3 State diagrams

101.3.4.1.3.1 Constants

B_P
: see 101.3.3.3.7

B_Q
: see 101.3.3.3.7

C_Q
: see 101.3.3.3.7

dataInSize
: VALUE: $(B_Q + 1 + C_Q) \times 65 + B_P$
: This constant represents the size of the *dataIn* array, containing the combination of the payload portion of the FEC codeword, the parity portion of the FEC codeword, CRC40, and all the necessary padding.

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

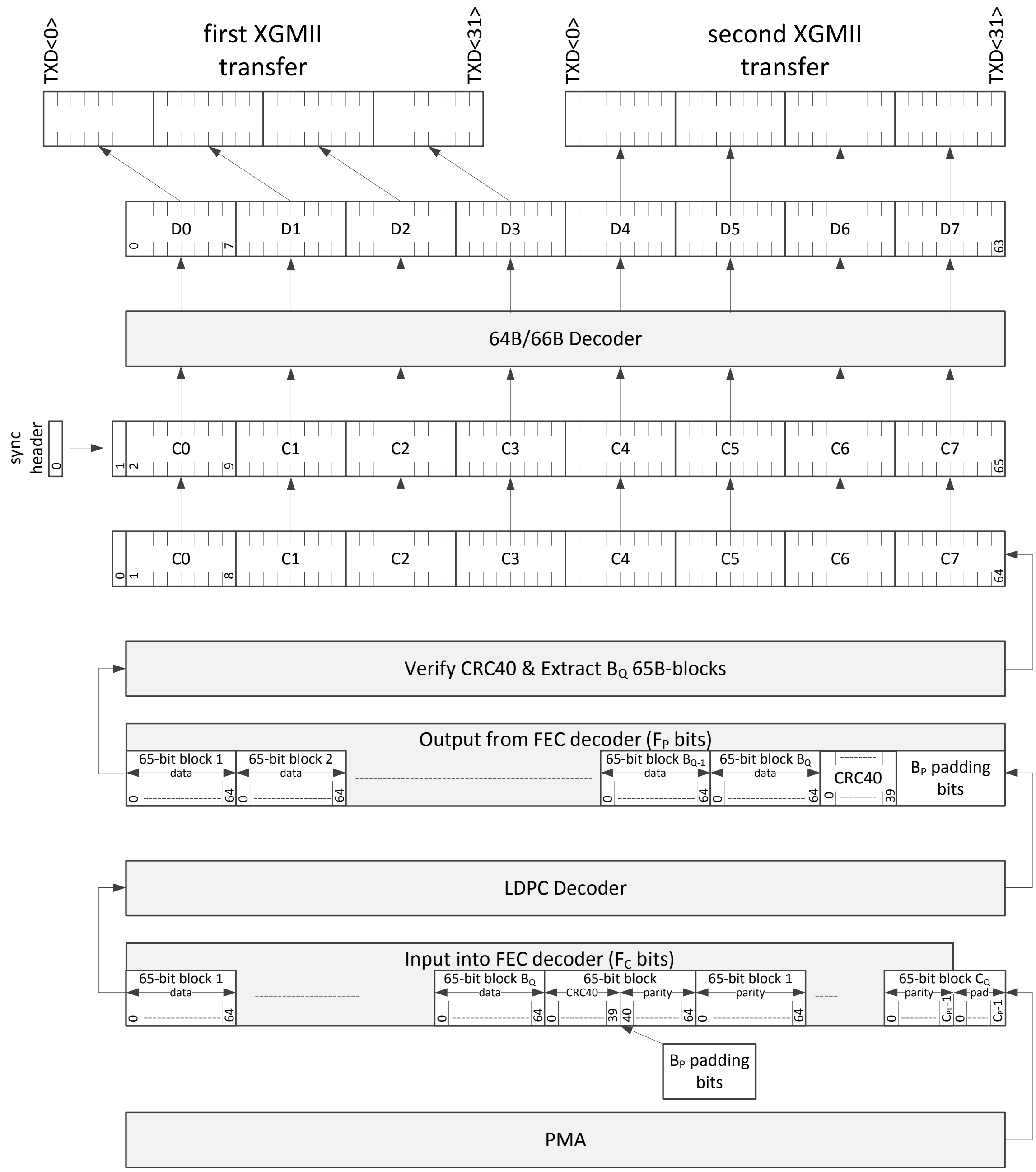

Figure 101–11—PCS Receive bit ordering within CNU (downstream)

101.3.4.1.3.2 Variables

blockCount
see 101.3.3.3.7

CLK
see 101.3.3.3.7

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”.

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 101.3.3.3.7

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>.

sizeFifo

see 101.3.3.3.7

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 101.3.3.3.7

101.3.4.1.3.3 Functions

calculateCrc (ARRAY_IN)

see 101.3.3.3.7

decodeFec(ARRAY_IN)

This function performs FEC decoding (for the code per Figure 101–6 or Figure 101–7) for data included in ARRAY_IN, comprising the combination of the payload portion of the FEC code-word, the parity portion of the FEC codeword, CRC40, and all the necessary padding.

resetArray(ARRAY_IN)

see 101.3.3.3.7

101.3.4.1.3.4 Messages

TBD

101.3.4.1.3.5 State diagrams

The CNU PCS shall implement the LDPC decoding process, comprising the input process as shown in Figure 101–12 and the output process as shown in Figure 101–10.

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

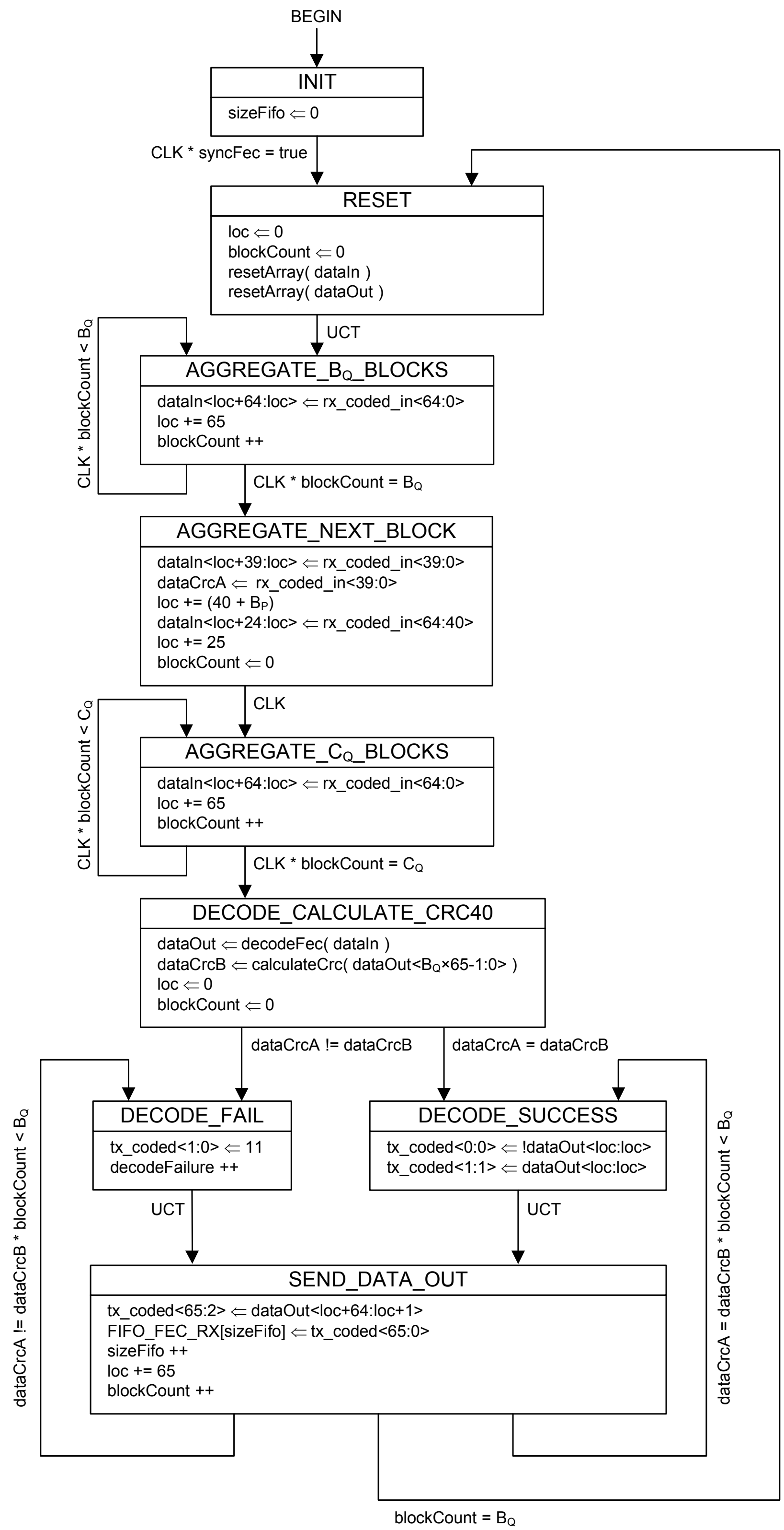

Figure 101–12—FEC decoder, input process state diagram (CNU)

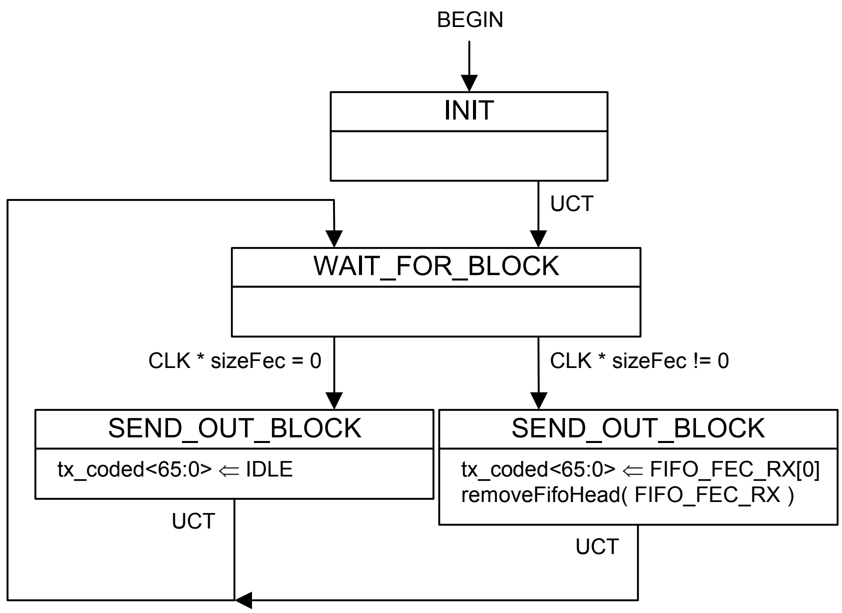

Figure 101–13—FEC decoder, output process state diagram (CNU)

101.3.4.2 Codeword Error Monitor

101.3.4.3 Descrambler / Interleaver

101.3.4.4 64B/66B Decode

The 64B/66B decoder shall perform the functions specified in {Figure 49–17}. The 64B/66B decoding process is as described in {49.2.11}, with the following exceptions:

a) the 64B/66B decode process in the EPoC PCS produces 72-bit vectors fed into the Idle control character insertion process (see 101.3.4.5), rather than directly into the XGMII; and

b) the 64B/66B decode process in the EPoC PCS operates on bursty data stream produced by the FEC decoder, unlike in 10GBASE-R PCS, where data stream to the input of the 64B/66B decoder is taken directly from the descrambler and hence continuous.

101.3.4.5 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 enforced by the MAC Control (as defined in {Clause 102}). 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 101-3}) 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 supported 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.

101.3.4.5.1 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}

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}.

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.

101.3.4.5.2 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 101.3.4.5.1).

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.

101.3.4.5.3 Functions

T_TYPE(rx_raw<71:0>)
: This function is defined in {49.2.13.2.3}.

101.3.4.5.4 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>).

101.3.4.5.5 State diagrams

The CLT and CNU PCS shall perform the Idle control character insertion process as shown in Figure 101–14. In case of any discrepancy between state diagrams and the descriptive text, the state diagrams prevail.

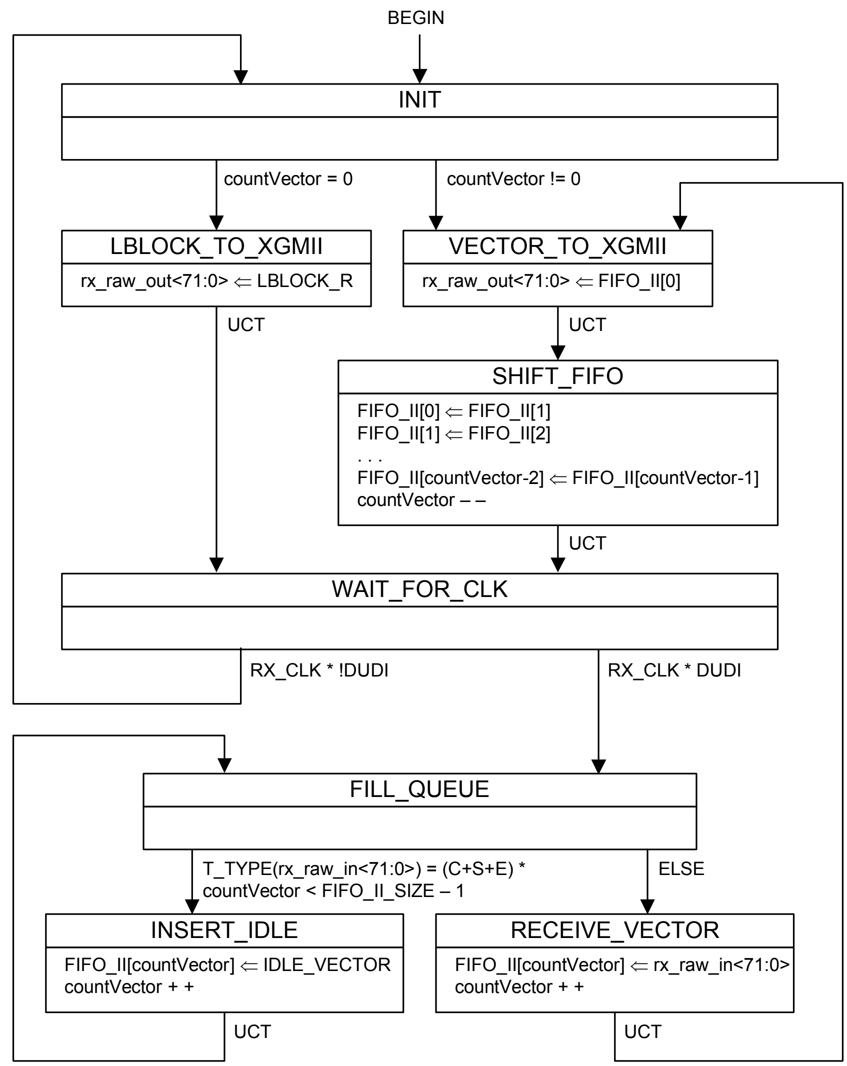

Figure 101–14—Idle control character insertion process state diagram