P802.3dt control ordered sets and link metadata (comments 52, 78-80, 82, 83)

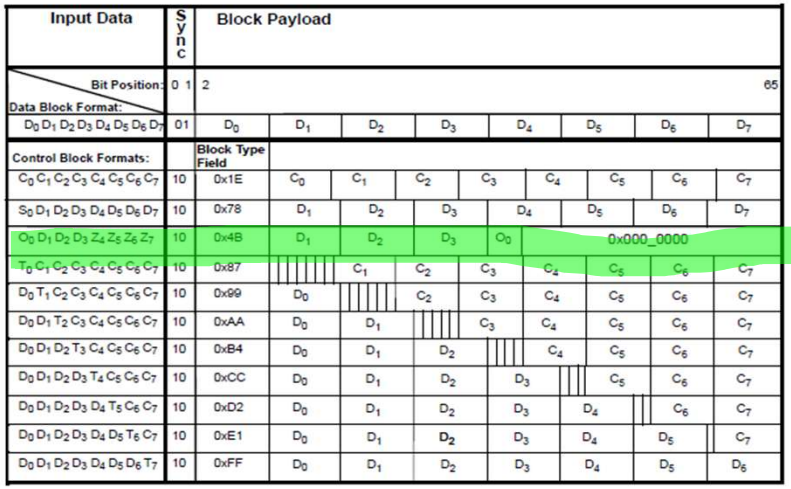
Tom Huber, Nokia

Control ordered sets

Comment 52

Existing text of clause 82 in 802.3-2022

- All ordered sets use the same format in the 66b coding and the same xMII control code
  - 802.3 defines only the Sequence ordered set
  - Other SDOs have defined other types of ordered sets, and 802.3 has chosen to acknowledge these (and to not define ordered sets using those O codes) in the interest of avoiding interop issues – those ordered sets use the format and control code as shown below



Input Data	Sync	Block Payload								
Bit Position: 0 1 2 … 65 / Data Block Format:										
$D_0 D_1 D_2 D_3 D_4 D_5 D_6 D_7$	01	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	
Control Block Formats:		Block Type Field								
$C_0 C_1 C_2 C_3 C_4 C_5 C_6 C_7$	10	0x1E	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7
$S_0 D_1 D_2 D_3 D_4 D_5 D_6 D_7$	10	0x78	D_1	D_2	D_3	D_4	D_5	D_6	D_7	
$O_0 D_1 D_2 D_3 Z_4 Z_5 Z_6 Z_7$	10	0x4B	D_1	D_2	D_3	O_0	0x000_0000			
$T_0 C_1 C_2 C_3 C_4 C_5 C_6 C_7$	10	0x87		C_1	C_2	C_3	C_4	C_5	C_6	C_7
$D_0 T_1 C_2 C_3 C_4 C_5 C_6 C_7$	10	0x99	D_0		C_2	C_3	C_4	C_5	C_6	C_7
$D_0 D_1 T_2 C_3 C_4 C_5 C_6 C_7$	10	0xAA	D_0	D_1		C_3	C_4	C_5	C_6	C_7
$D_0 D_1 D_2 T_3 C_4 C_5 C_6 C_7$	10	0xB4	D_0	D_1	D_2		C_4	C_5	C_6	C_7
$D_0 D_1 D_2 D_3 T_4 C_5 C_6 C_7$	10	0xCC	D_0	D_1	D_2	D_3		C_5	C_6	C_7
$D_0 D_1 D_2 D_3 D_4 T_5 C_6 C_7$	10	0xD2	D_0	D_1	D_2	D_3	D_4		C_6	C_7
$D_0 D_1 D_2 D_3 D_4 D_5 T_6 C_7$	10	0xE1	D_0	D_1	D_2	D_3	D_4	D_5		C_7
$D_0 D_1 D_2 D_3 D_4 D_5 D_6 T_7$	10	0xFF	D_0	D_1	D_2	D_3	D_4	D_5	D_6	

Figure 82–5—64B/66B block formats

Table 82–1—Control codes

Control character	Notation	XLGMII/ CGMII control code	40/100GBASE-R O code[a]	40GBASE-R and 100GBASE-R control code
idle	/I/	0x07		0x00
LPI	/LI/	0x06		0x06
start	/S/	0xFB		Encoded by block type field
terminate	/T/	0xFD		Encoded by block type field
error	/E/	0xFE		0x1E
Sequence ordered set	/Q/	0x9C	0x0	Encoded by block type 0x4B plus O code, control codes are set to 0x00

[a] INCITS T11 Fibre Channel uses O code 0xF for the signal ordered set. OIF uses O code 0x5 for the FlexE [B59] ordered set. ITU-T uses O code 0xC for the G.8312 [B47] MTN path ordered set.

OIF FlexE and ITU-T MTN ordered set formats

OIF-FLEXE-03.0

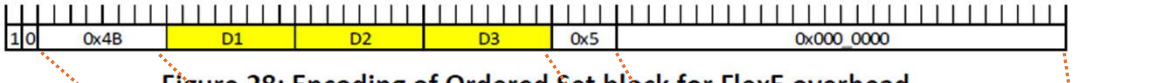


Figure 28: Encoding of Ordered Set block for FlexE overhead

Ordered set format per 802.3 Figure 82-5

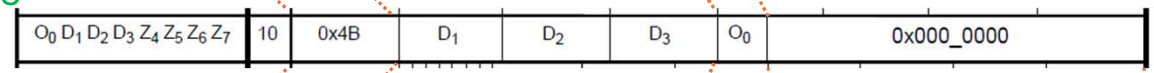




Figure 8-7 – MTNP OAM 66B block structure

Recommendation ITU-T G.8312

- Both of these ordered sets use the existing format
  - This is not inherently a problem, as 0x0000000 is a valid value to insert in D4'-D7
- These OS are never present at the MII; they are inserted and extracted in functions that extend the PCS
- These OS are not link metadata and may not have the behavior of control OS that 802.3 defines
- 802.3dt should not define anything about the O codes that other SDOs use; they are not part of 802.3

Changes introduced in P802.3dt D1.0

- P802.3dt proposes to redefine block type 4B and define a new “control ordered set” that is used for all ordered sets with O codes other than 0x0

Input Data	Sync	Block Payload							
Bit Position: / Data Block Format:	0 1	2							65
$D_0\,D_1\,D_2\,D_3\,D_4\,D_5\,D_6\,D_7$	01	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
Control Block Formats:		Block Type Field							
$C_0\,C_1\,C_2\,C_3\,C_4\,C_5\,C_6\,C_7$	10	0x1E	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7
$S_0\,D_1\,D_2\,D_3\,D_4\,D_5\,D_6\,D_7$	10	0x78	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$O_0\,D_1\,D_2\,D_3\,D_4\,D_5\,D_6\,D_7$	10	0x4B	D_1	D_2	D_3	O	D'4	D_5	D_6	D_7
$T_0\,C_1\,C_2\,C_3\,C_4\,C_5\,C_6\,C_7$	10	0x87		C_1	C_2	C_3	C_4	C_5	C_6	C_7
$D_0\,T_1\,C_2\,C_3\,C_4\,C_5\,C_6\,C_7$	10	0x99	D_0		C_2	C_3	C_4	C_5	C_6	C_7
$D_0\,D_1\,T_2\,C_3\,C_4\,C_5\,C_6\,C_7$	10	0xAA	D_0	D_1		C_3	C_4	C_5	C_6	C_7
$D_0\,D_1\,D_2\,T_3\,C_4\,C_5\,C_6\,C_7$	10	0xB4	D_0	D_1	D_2		C_4	C_5	C_6	C_7
$D_0\,D_1\,D_2\,D_3\,T_4\,C_5\,C_6\,C_7$	10	0xCC	D_0	D_1	D_2	D_3		C_5	C_6	C_7
$D_0\,D_1\,D_2\,D_3\,D_4\,T_5\,C_6\,C_7$	10	0xD2	D_0	D_1	D_2	D_3	D_4		C_6	C_7
$D_0\,D_1\,D_2\,D_3\,D_4\,D_5\,T_6\,C_7$	10	0xE1	D_0	D_1	D_2	D_3	D_4	D_5		C_7
$D_0\,D_1\,D_2\,D_3\,D_4\,D_5\,D_6\,T_7$	10	0xFF	D_0	D_1	D_2	D_3	D_4	D_5	D_6	

Figure 82–5—64B/66B block formats

Table 82–1—Control codes

Control character	Notation	XLGMII/ CGMII control code	40/100GBASE-R O code[a]	40GBASE-R and 100GBASE-R control code
idle	/I/	0x07		0x00
LPI	/LI/	0x06		0x06
start	/S/	0xFB		Encoded by block type field
terminate	/T/	0xFD		Encoded by block type field
error	/E/	0xFE		0x1E
Sequence ordered set	/Q/	0x9C	0x0	Encoded by block type 0x4B plus O code, control codes are set to 0x00
Control ordered set	/C/	0x5C	0x1 - 0xF	Encoded by block type 0x4B and a nonzero O code.

[a] INCITS T11 Fibre Channel uses O code 0xF for the signal ordered set. OIF uses O code 0x3 for the FlexE [B59] ordered set. ITU-T uses O code 0xC for the G.8312 [B47] MTN path ordered set. UEC uses O code 0x6 for the control ordered set [B68a].

The problem

- Ordered sets defined by other SDOs do not fully comply with the new definition of control ordered set
  - They use the 802.3-2022 format (unchanged since 802.3ba-2010) for block type 4B, where the “control codes” – meaning Z4-Z7 – are fixed at zero
    - Changing Z4-Z7 to D4’-D7 in format 0x4B doesn’t break that; 0 is a valid value to encode in D4’-D7. But it is not correct to say that these ordered sets can use other values than 0 in those bytes.
  - The revised Table 82-1 states that these ordered sets defined by other SDOs are now ‘control ordered sets’ in the context of 802.3
    - The FlexE and MTN ordered sets never appear at the MII, so the xMII control code is not relevant to them
- Defining control ordered sets as all OS with non-zero O codes implies that 802.3 can use the codes that are being used by INCITS, OIF, and ITU-T, and thus the same O code could mean two different ordered sets

Proposed solution

- Consider not changing the format of control block type 4B
  - Instead, introduce a different control block format for the new control ordered sets
- Whether or not a new format is introduced, change table 82-1 to not define O codes 0x5, 0xC, and 0xF as “control ordered sets”
  - They should be left undefined, as they are not part of the 802.3 standard

Table 82–1—Control codes

Control character	Notation	XLGMII/ CGMII control code	40/100GBASE-R O code[3]	40GBASE-R and 100GBASE-R control code
idle	/I/	0x07		0x00
LPI	/LI/	0x06		0x06
start	/S/	0xFB		Encoded by block type field
terminate	/T/	0xFD		Encoded by block type field
error	/E/	0xFE		0x1E
Sequence ordered set	/Q/	0x9C	0x0	Encoded by block type 0x4B plus O code, control codes are set to 0x00
Control ordered set	/C/	0x5C	0x1 - 0xF	Encoded by block type 0x4B and a nonzero O code.

[3] DICITS T11 Fibre Channel uses O code 0xF for the signal ordered set. OIF uses O code 0x5 for the FlexE [B39] ordered set. ITU-T uses O code 0xC for the G.8312 [B47] MTN path ordered set. UEC uses O code 0x6 for the control ordered set [B68a].

Change 0x1 – 0xF to 0x1 – 0x4, 0x6-0xB, 0xD-0xE

Change “a nonzero O code” to “one of the listed O codes”

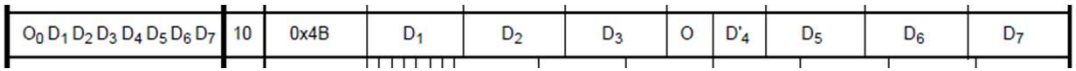
Add text: The format of these ordered sets is outside the scope of this specification

Link metadata service interface

Comments 78-80, 82, 83

Definition in subclause 195.3.5

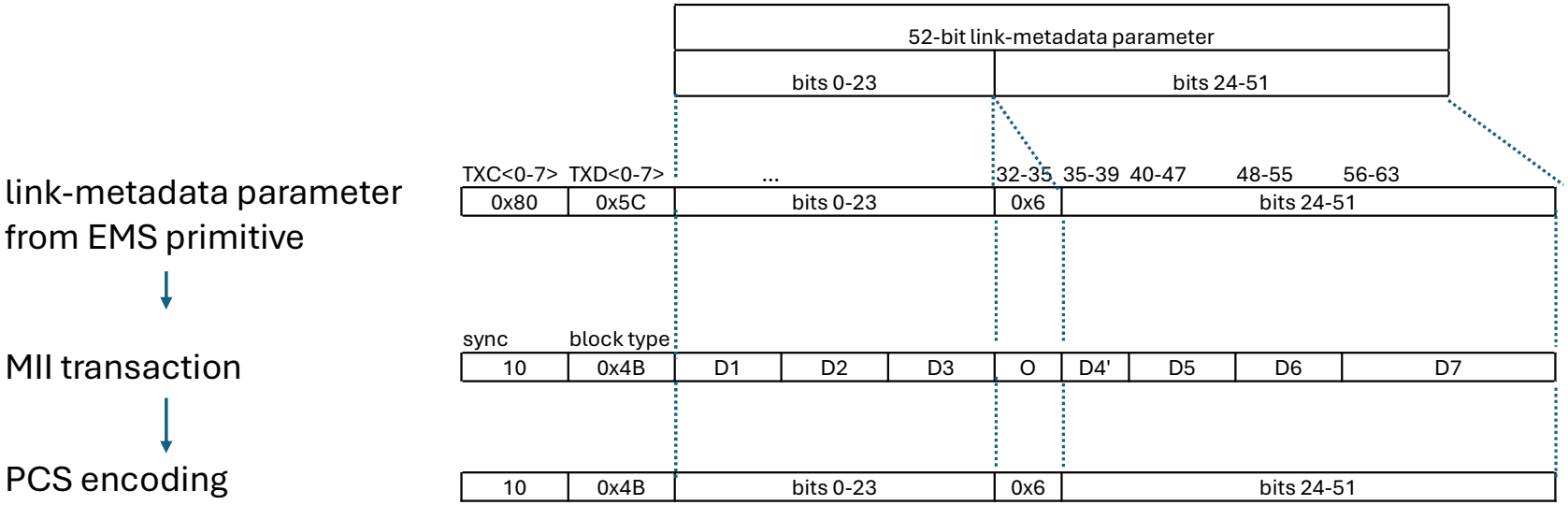
- The service interface defines the link_metadata parameter as 7 octets
- However, the parameter is encoded into block type 4B, which is (re-)defined as 6.5 octets and a 4-bit O code that is set to 0x6



- There is no need for the EMS Client to be aware of this encoding detail (in the same way it does not provide the control block type 0x4B, it shouldn't need to provide the O code)
- It is architecturally cleaner to define link_metadata as 52 bits and specify how they are mapped to the MII transaction in the gRS and/or to the control ordered set block in the PCS

Proposed solution

- Define link_metadata as 52 bits in subclause 195.3.5
- In 195.4.1.2, describe the mapping of the 52 bits to a control ordered set over the xMII
  - The figure below also shows the mapping from the MII to the PCS encoding, which could optionally also be included in 195.4.1.2.



Thank you!