Interpretation Number:	1-11/01 - Item 1
Topic:	Definition of CEXT and CEXT_Err symbols
Relevant Clauses:	Figure 40-9
Classification:	Unambiguous

Interpretation Request

Referring to Fig.40-9, state - 'CARRIER EXTENSION' transmits either CEXT symbols if TXD<7:0> = 0x0F or CEXT_Err symbols if TXD<7:0> != 0x0F.

However if we look at Table 40-1 and Table 40-2 Bit-to-symbol mapping (even and odd subsets) there is no mapping for CEXT_Err.

Further in Clause 40.3.3.1, variable CEXT_Err is defined as code-group generated in Idle mode to denote carrier extension with error indication, as specified in Clause 40.3.1.3.

So the question is: what symbols does one transmit on the 4-twisted pairs to denote CEXT_Err ?

Are they from Idles/CEXT portion of table 40-1 dependent on Sd(n)[1:0] as per Clause 40.3.1.3.4 ?

Interpretation for IEEE std 802.3-2000

In subclause 40.3.1.3.5 the standard states that 'the nine-bit word $Sd_n[8:0]$ is mapped to a quartet of quinary symbols according to Table 40-1 and Table 40-2'. Hence to determine the symbols to be sent the value of the nine-bit word $Sd_n[8:0]$ has to be first determined.

Subclause 40.3.1.3.4 'Generation of bits $Sd_n[8:0]$' specifies how the nine-bit word $Sd_n[8:0]$ is calculated. Using this information, the value of the nine-bit word $Sd_n[8:0]$ in the 'CARRIER EXTENSION' state of Figure 40-9 'PCS Transmit state diagram' can be derived as follows (Note equations are numbered in the order they appear in the subclause).

$Sd_n[8]$ is derived using the first 4 equations that appear in subclause 40.3.1.3.4 as follows:

[eqn 1] $cs_n[1] = 0$
[eqn 2] $cs_n[2] = 0$
[eqn 3] $cs_n[0] = cs_{n-1}[2] = 0$

Hence:

[eqn 4] $Sd_n[8] = cs_n[0] = 0$

$Sd_n[7:0]$ is derived using the remaining equations in subclause 40.3.1.3.4 as follows:

[eqn 5] $csreset_n = (tx\_enable_{n-2})$ and (not tx_enable_n)
$= 0$ and 1
$= 0$

Since $Sd_n[7:4]$ is derived from $Sc_n[7:4]$, the value of $Sc_n[7:4]$ must be derived first from subclause 40.3.1.3.3 as follows:

[eqn 1] $Sd_n[7:4] = 0000 \quad (tx\_enable_{n2} = 0)$

Returning to the equations in subclause 40.3.1.3.4, based on $csreset_n = 0$, $tx\_enable_{n-2} = 0$ and the value $Sc_n[7:4]$ derived above:

[eqn 6] $Sd_n[7] = Sc_n[7] = 0$
[eqn 7] $Sd_n[6] = Sc_n[7] = 0$
[eqn 8] $Sd_n[5:4] = Sc_n[5:4] = 00$
$Sd_n[3] = Sc_n[3]$
[eqn 9] $Sd_n[2] = Sc_n[2] \wedge (loc\_rcvr\_status = \text{OK})$
[eqn 12] $Sd_n[1] = Sc_n[1] \wedge cext\_err_n$
[eqn 13] $Sd_n[0] = Sc_n[0] \wedge cext_n$

Where:

[eqn 10] $cext_n = (tx\_error_n$ and $TXD_n[7:0] = \text{0x0F})$
[eqn 11] $cext\_err_n = (tx\_error_n$ and $TXD_n[7:0] \neq \text{0x0F})$

Hence it can be seen that in the 'CARRIER EXTENSION' state of Figure 40-9:

During Carrier Extension (tx_error_n and $TXD_n[7:0] = \text{0x0F}$)

$Sd_n[8:4] = 00000$
$Sd_n[3] = Sc_n[3]$
$Sd_n[2] = Sc_n[2] \wedge (loc\_rcvr\_status = \text{OK})$
$Sd_n[1] = Sc_n[1]$
$Sd_n[0] = Sc_n[0] \wedge 1$

During Carrier Extension Error (tx_error_n and $TXD_n[7:0] \neq \text{0x0F}$)

$Sd_n[8:4] = 00000$
$Sd_n[3] = Sc_n[3]$
$Sd_n[2] = Sc_n[2] \wedge (loc\_rcvr\_status = \text{OK})$
$Sd_n[1] = Sc_n[1] \wedge 1$
$Sd_n[0] = Sc_n[0]$

Further it can be seen that during the Idle state:

$Sd_n[8:4] = 00000$
$Sd_n[3] = Sc_n[3]$
$Sd_n[2] = Sc_n[2] \wedge (loc\_rcvr\_status = \text{OK})$
$Sd_n[1] = Sc_n[1]$
$Sd_n[0] = Sc_n[0]$

In summary, in both the 'SEND IDLE' and 'CARRIER EXTENSION' states of Figure 40-9, the value of the bits $Sd_n[8:4]$ is 00000 and the value of the bits $Sd_n[3:0]$ are derived from bits $Sc_n[3:0]$. Bit $Sd_n[3]$ is equal to the bit $Sc_n[3]$, and bits $Sd_n[2:0]$ are equal to either the true or inverse of the bits $Sc_n[2:0]$ on a bit by bit basis. The selection between the true and inverse of the $Sc_n[2:0]$ bits is determined by the value of loc_rcvr_status for $Sc_n[2]$, $cext\_err_n$ for $Sc_n[1]$ and $cext_n$ for $Sc_n[0]$. The $Sc_n[3:0]$ bits are derived from the Side Stream Scrambler described in subclause 40.3.1.3.1.

Since Sdn[8:4] is 00000, the quinary symbols in Table 40-1 labelled with the condition 'Idle/Carrier Extension' will be sent in both the 'SEND IDLE' and 'CARRIER EXTENSION' states. The sequence in which these symbols are sent will be determined by the $Sd_n[3:0]$ bits which are derived from the $Sc_n[3:0]$ bits. A different sequence of these 'Idle/Carrier Extension' symbols will be sent dependent on $cext_n$, $cext\_err_n$ and loc_rcvr_status since these control the inversion of the $Sc_n[2:0]$ bits used in the generation of the $Sd_n[2:0]$ bits.

Therefore, specifically for CEXT_Err, as can be seen above, the standard describes that the quinary symbols sent are the symbols in Table 40-1 marked with the condition 'Idle/Carrier Extension'. The sequence in which these symbols are sent will be controlled by the $Sc_n[3:0]$ bits. The sequence will be unique for CEXT_Err since this is the only time when $Sc_n[1]$ is inverted and $Sc_n[0]$ is true in the generation of $Sd_n[3:0]$.

A maintenance change has been raised to make this more specific. This change request is available at the
URL: http://www.ieee802.org/3/maint/requests/maint_1083.pdf.

Interpretation Number:	1-11/01 - Item 2
Topic:	Definition of CEXT symbols and IDLE symbols
Relevant Clauses:	Figure 40-10a
Classification:	Unambiguous

Interpretation Request

Referring to Fig.40-10a (part a), state - 'EXTENDING' goes to either state 'CARRIER EXTENSION' if Rx(n-1) is CEXT or state 'CARRIER EXTENSION with ERROR' if Rx(n-1) is IDLE

However if we look at Table 40-1 Bit-to-symbol mapping (even subsets) the mapping for IDLE and CEXT is the same.

Further, as per Clause 40.3.1.3.4, for tx-path :

Sd(n)[1] = Sc(n)[1] ^ cext_err(n) (if tx_enable(n-2) = 0)

Sd(n)[0] = Sc(n)[0] ^ cext(n) (if tx_enable(n-2) = 0)

and so for Rx-path, the answer seems to be :

cext(n) = Sd(n)[0] ^ Sc(n)[0] (if RX_DV = 0)

and

cext_err(n) = Sd(n)[1] ^ Sc(n)[1] (if RX_DV = 0)

and

Idle = others (while RX_DV = 0)

Is this assumption correct ?

So the question is: In the Rx-path how does one differentiate between Idles/CEXT/CEXT_Err in table 40-1?

Seems to be dependent on Sd(n)[1:0] ?

Interpretation for IEEE std 802.3-2000

As the standard states in 40.3.1.4 PCS Receive, ‘To achieve correct operation, PCS Receive uses the knowledge of the encoding rules that are employed in the idle mode.’ Further, as seen above in the response to Item 1, the sequence of symbols sent during IDLE, Carrier Extension and Carrier Extension Error is controlled by the current Sc_n value.

Hence to extract IDLE/CEXT/CEXT_Err from Rx_n (which maps to Sd_n), knowledge of the current scrambler state (via Sc_n) is required.

Interpretation Number:	1-11/01 - Item 3
Topic:	Scrambler generator polynomial
Relevant Clauses:	40.6.1.1.2
Classification:	Defect

Interpretation Request

Lastly, there seems to be a typo in Clause 40.6.1.1.2 - Test Modes of Std 802.3, 2000 Edition

The scrambler generator polynomial should be :

gs1 = 1 + x^9 + x^11 instead of : s1 = 1 + x^9 + x^1

Interpretation for IEEE std 802.3-2000

This represents a conflict within the standard. A change request has been generated to correct this which is available at the URL: http://www.ieee802.org/3/maint/requests/maint_1084.pdf