ANALOG DEVICES

AHEAD OF WHAT'S POSSIBLE™

100BASE-T1L Block Encoding and Remote Fault

Philip Curran
Brian Murray
Jacobo Riesco

Introduction

- This presentation presents an update to the text for the draft for clause 199.3.4 PCS Transmit function for the block encoding
- At the IEEE plenary in March 2025 we presented a proposal on link fault signaling and the changes required to clause 22 to support this
  - See Murray_3dg_01_03122025
    Clause 22 MII Changes for 802.3dg Required by Sequence Ordered Sets
    - Minimum change to clause 22 MII and use the Q code to signal Remote Fault across the link
    - Supports the spirit of fault signaling without adding a lot of complexity
- Adopted motions
  - March 2025 Motion #2:
    Move that the IEEE P802.3dg Task Force adopt slides 10 to 13 of Murray_3dg_02_03122025
    - Proposed text for the PCS block structure (block encoding)
  - March 2025 Motion #3:
    Move that the IEEE P802.3dg Task Force adopt slide 10 of Murray_3dg_01_03122025
    - Add **Assert local fault** / **Assert remote fault** to Clause 22 MII

Proposed Improvements

- At the IEEE plenary in March 2025 we also had a presentation on proposed improvements to the adopted proposals
  - See Lo_3dg_01_0325 Remote Fault Handling
- Possible improvements
  - The linkup sequence proposed in March toggled between Idle and Fault, which is different from XGMII

    Local Fault ➔ Idle ➔ Remote Fault ➔ Idle
  - It would be better to have a cleaner linkup sequence like the XGMII fault signaling, that just went from fault to idle

    Local Fault ➔ Remote Fault ➔ Idle
  - Is it possible to keep the /Q/ code reserved for possible future use
- There was a lot of positive discussion at the March meeting and a commitment to explore if it is possible to support these improvements
- This presentation proposes changes to achieve the desired improvements
  - And presents an update to the associated text and tables for the draft

Mapping MII Transfers to 8N/(8N + 1) Blocks

- Figure 190-5 is a diagram showing the mapping of the MII transfers to the 8N/(8N+1) blocks and 6T symbols
  - This is the exact same as draft 1.0

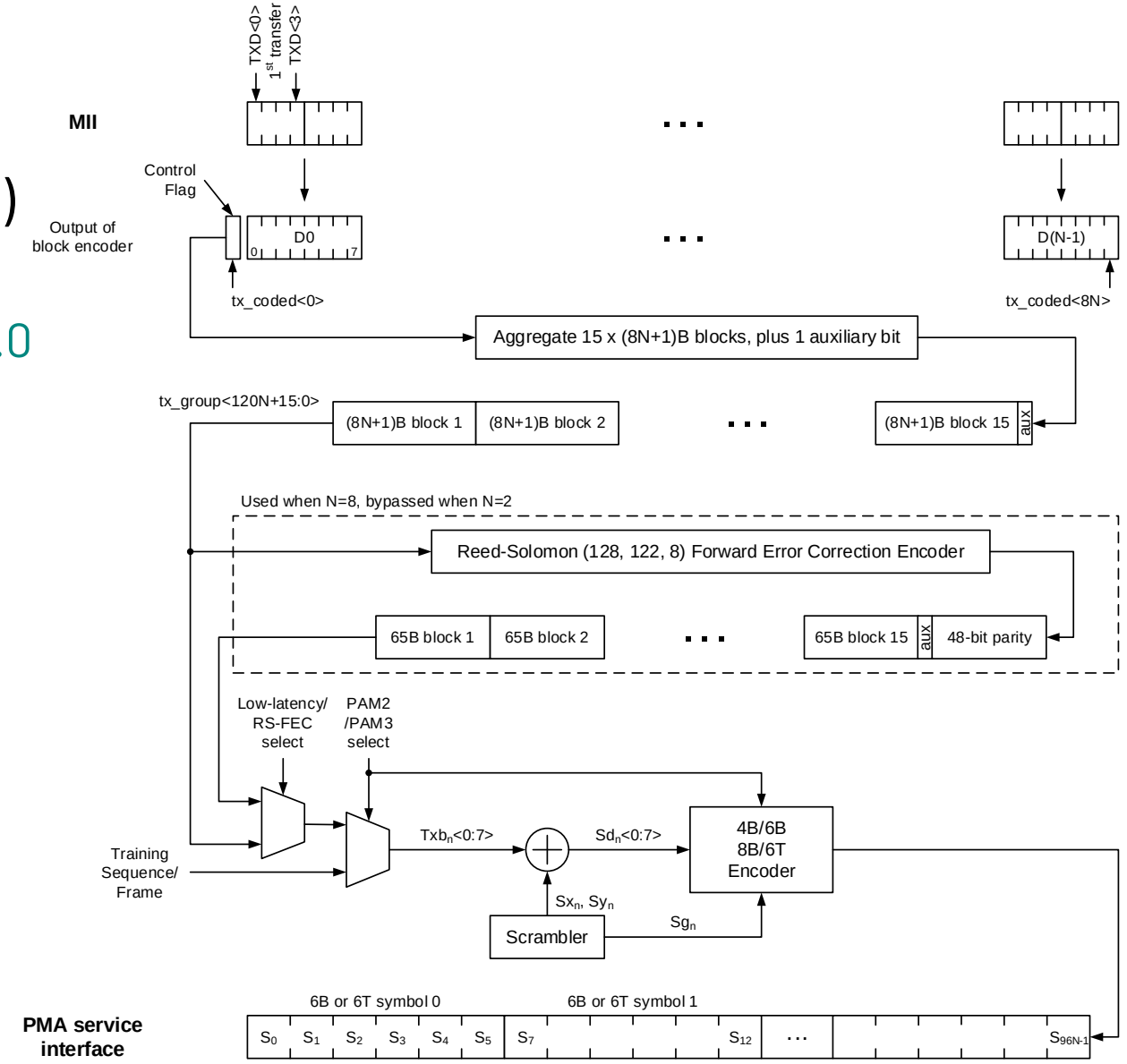

Figure 190–5—
PCS Transmit bit ordering

Encoding of data on the MII into an 8N/(8N+1) Block

- The encoding of data on the MII into an 8N/(8N+1) block is done as follows and is the exact same as draft 1.0
- Block encoding
  - Blocks consist of 8N + 1 bits
  - The first bit of a block is the control flag
  - Blocks are either data blocks or control blocks
    - The control flag is 0 for data blocks and 1 for control blocks
  - The remainder of the block contains the payload
- The encoding is a two-step process
  - The first step converts two MII transfers at a time into a control symbol indication TS, and an octet, TOCT
  - The second step packs a set of N values of TS and of TOCT into an 8N+1 bit block
- Transfers over the MII are delineated as alternating even and odd transfers
  - The conversion of the even and odd transfers to the values TS and TOCT follows Table 190–2 using the categories defined in Table 190–1

MII Transfer Categories

- The MII transfer categories are defined in an update to Table 190-1
  - The category ARF is Assert remote fault
  - For each category there is a complementary category which includes all MII transfers that do not belong to that category
  - An MII transfer may belong to more than one category
  - For example, ARF belongs to the categories ARF and IDL

Table 190–1— MII transfer categories

Category	TX_EN	TX_ER	TXD<3:0>	Description
DAT	1	0	–	Normal data transmission
ERR	1	1	–	Transmit error propagation
ARF	0	1	0100	Assert remote fault
IDL	0	–	–	Idle signaling

IEEE P802.3dg™/D1.0, 29th April 2025

Table 190–1—MII transfer categories

Category	tx_enable	tx_error	TXD<3:0>	Description
	loc_phy_ready = FALSE			
~~NOT_RDY~~	~~—~~	~~—~~	~~—~~	~~PHY not ready for MII transfer~~
	loc_phy_ready = TRUE			
DAT	1	0	—	Normal data transmission
ERR	1	1	—	Transmit error propagation
~~NIF~~	~~0~~	~~0~~	~~—~~	~~Normal inter-frame~~
~~ALPI~~	~~0~~	~~1~~	~~0001~~	~~Assert LPI~~
ARF	0	1	0100	Assert remote fault
IDL	0	—	—	

Changes from draft 1.0

- There is a new Figure 190–12—PCS (8N)B/(8N + 1)B Transmit state diagram
- The new state diagram now handles NOT_RDY by encoding N /Ix/ control symbols into every (8N+1)B block while loc_phy_ready is FALSE
- The NIF category is now redundant because it is handled as part of the IDL category
- The new state diagram now handles Assert LPI by sending the sleep signal which is composed entirely of /LI/ control symbols

MII Transfer to TS and TOCT Mapping

- This is the exact same as draft 1.0, except for changes to the Table
- Evaluation from top to bottom
  - Evaluation of the even and odd transfers against the conditions listed in the table proceeds from top to bottom
  - The first condition that is satisfied has priority, and TS and TOCT are set in accordance with the corresponding row in the table
- TOCT values
  - The table shows the TOCT values for control symbols using symbolic representations for clarity
  - The associated numerical values is shown in Table 190–3
- Delayed encoding
  - **dly_enc** records a requirement to implement a delayed encoding
  - This condition arises when a pair of transfers require two control symbols to be encoded
  - The encoding of one of these control symbols is delayed
  - For example, the encoding of transmit error propagation is delayed if it is signaled during the first MII transfer of a frame
  - When transmit error propagation is signaled during the last transfer of the frame and this transfer is even, the frame is extended by one byte
    - This allows the decoder to observe both the transmit error propagation event and the mismatch in byte alignment
    - The Transmit Error Propagation symbol (/E/) is followed by the Terminate symbol /Tu0/

Table 190–2— MII transfer to TS and TOCT mapping

Previous transfer	Even transfer	Odd transfer	Current dly_enc	TS	TOCT	Next dly_enc
IDL	DAT	!ERR	–	1	/Sp/	FALSE
IDL	DAT	ERR	–	1	/Sp/	TRUE
IDL	ERR	–	–	1	/Sp/	TRUE
–	IDL	DAT	–	1	/Su/	FALSE
–	IDL	ERR	–	1	/Su/	TRUE
!IDL	DAT	DAT	TRUE	1	/E/	FALSE
!IDL	IDL	–	–	1	/Tp/	FALSE
!IDL	DAT, TXD = 0x0	IDL	–	1	/Tu0/	FALSE
!IDL	DAT, TXD = 0x1	IDL	–	1	/Tu1/	FALSE
!IDL	DAT, TXD = 0x2	IDL	–	1	/Tu2/	FALSE
!IDL	DAT, TXD = 0x3	IDL	–	1	/Tu3/	FALSE
!IDL	DAT, TXD = 0x4	IDL	–	1	/Tu4/	FALSE
!IDL	DAT, TXD = 0x5	IDL	–	1	/Tu5/	FALSE
!IDL	DAT, TXD = 0x6	IDL	–	1	/Tu6/	FALSE
!IDL	DAT, TXD = 0x7	IDL	–	1	/Tu7/	FALSE
!IDL	DAT, TXD = 0x8	IDL	–	1	/Tu8/	FALSE
!IDL	DAT, TXD = 0x9	IDL	–	1	/Tu9/	FALSE
!IDL	DAT, TXD = 0xA	IDL	–	1	/TuA/	FALSE
!IDL	DAT, TXD = 0xB	IDL	–	1	/TuB/	FALSE
!IDL	DAT, TXD = 0xC	IDL	–	1	/TuC/	FALSE
!IDL	DAT, TXD = 0xD	IDL	–	1	/TuD/	FALSE
!IDL	DAT, TXD = 0xE	IDL	–	1	/TuE/	FALSE
!IDL	DAT, TXD = 0xF	IDL	–	1	/TuF/	FALSE
!IDL	ERR	IDL	–	1	/E/	TRUE
IDL	IDL	IDL	TRUE	1	/Tu0/	FALSE
!IDL	ERR	!IDL	–	1	/E/	FALSE
!IDL	DAT	ERR	–	1	/E/	FALSE
IDL	ARF	ARF	FALSE	1	/R/	FALSE
IDL	IDL	IDL	FALSE	1	/I/	FALSE
!IDL	DAT TOCT<3:0> = TXD	DAT TOCT<7:4> = TXD	FALSE	0	Data Value	FALSE
Otherwise				1	/I/	FALSE

MII Transfer to TS and TOCT Mapping

- **Changes from draft 1.0**
  - We have now split the **Even transfer** column into two
    - E.g. instead of **DAT after IDL**
    - ➢ Have **IDL** in **Previous transfer**
    - ➢ and **DAT** in **Even transfer**
  - Remove **NOT_RDY**
    - As this is not a category of an MII transfer
  - Remove **ALPI** and remove **NIF**
    - **ALPI** and **NIF** are handled as **IDL**

Table 190–2— MII transfer to TS and TOCT mapping

Previous transfer	Even transfer	Odd transfer	Current dly_enc	TS	TOCT	Next dly_enc
IDL	DAT	!ERR	–	1	/Sp/	FALSE
IDL	DAT	ERR	–	1	/Sp/	TRUE
IDL	ERR	–	–	1	/Sp/	TRUE
–	IDL	DAT	–	1	/Su/	FALSE
–	IDL	ERR	–	1	/Su/	TRUE
!IDL	DAT	DAT	TRUE	1	/E/	FALSE
!IDL	IDL	–	–	1	/Tp/	FALSE
!IDL	DAT, TXD = 0x0	IDL	–	1	/Tu0/	FALSE
!IDL	DAT, TXD = 0x1	IDL	–	1	/Tu1/	FALSE
!IDL	DAT, TXD = 0x2	IDL	–	1	/Tu2/	FALSE
!IDL	DAT, TXD = 0x3	IDL	–	1	/Tu3/	FALSE
!IDL	DAT, TXD = 0x4	IDL	–	1	/Tu4/	FALSE
!IDL	DAT, TXD = 0x5	IDL	–	1	/Tu5/	FALSE
!IDL	DAT, TXD = 0x6	IDL	–	1	/Tu6/	FALSE
!IDL	DAT, TXD = 0x7	IDL	–	1	/Tu7/	FALSE
!IDL	DAT, TXD = 0x8	IDL	–	1	/Tu8/	FALSE
!IDL	DAT, TXD = 0x9	IDL	–	1	/Tu9/	FALSE
!IDL	DAT, TXD = 0xA	IDL	–	1	/TuA/	FALSE
!IDL	DAT, TXD = 0xB	IDL	–	1	/TuB/	FALSE
!IDL	DAT, TXD = 0xC	IDL	–	1	/TuC/	FALSE
!IDL	DAT, TXD = 0xD	IDL	–	1	/TuD/	FALSE
!IDL	DAT, TXD = 0xE	IDL	–	1	/TuE/	FALSE
!IDL	DAT, TXD = 0xF	IDL	–	1	/TuF/	FALSE
!IDL	ERR	IDL	–	1	/E/	TRUE
IDL	IDL	IDL	TRUE	1	/Tu0/	FALSE
!IDL	ERR	!IDL	–	1	/E/	FALSE
!IDL	DAT	ERR	–	1	/E/	FALSE
IDL	ARF	ARF	FALSE	1	/R/	FALSE
IDL	IDL	IDL	FALSE	1	/I/	FALSE
!IDL	DAT TOCT<3:0> = TXD	DAT TOCT<7:4> = TXD	FALSE	0	Data Value	FALSE
Otherwise				1	/I/	FALSE

Table 199–2— MII transfer to TS and TOCT mapping

Even transfer	Odd transfer	Current dly_enc	TS	TOCT	Next dly_enc
~~NOT_RDY~~	~~–~~	~~–~~	~~1~~	~~/Ix/~~	~~FALSE~~
	~~NOT_RDY~~		~~1~~	~~/Ix/~~	~~FALSE~~
DAT after IDL	!ERR	–	1	/Sp/	FALSE
DAT after IDL	ERR	–	1	/Sp/	TRUE
ERR after IDL	–	–	1	/Sp/	TRUE
IDL	DAT	–	1	/Su/	FALSE
IDL	ERR	–	1	/Su/	TRUE
DAT after !IDL	DAT	TRUE	1	/E/	FALSE
IDL after !IDL	–	–	1	/Tp/	FALSE
DAT and TXD = 0x0	IDL	–	1	/Tu0/	FALSE
DAT and TXD = 0x1	IDL	–	1	/Tu1/	FALSE
DAT and TXD = 0x2	IDL	–	1	/Tu2/	FALSE
DAT and TXD = 0x3	IDL	–	1	/Tu3/	FALSE
DAT and TXD = 0x4	IDL	–	1	/Tu4/	FALSE
DAT and TXD = 0x5	IDL	–	1	/Tu5/	FALSE
DAT and TXD = 0x6	IDL	–	1	/Tu6/	FALSE
DAT and TXD = 0x7	IDL	–	1	/Tu7/	FALSE
DAT and TXD = 0x8	IDL	–	1	/Tu8/	FALSE
DAT and TXD = 0x9	IDL	–	1	/Tu9/	FALSE
DAT and TXD = 0xA	IDL	–	1	/TuA/	FALSE
DAT and TXD = 0xB	IDL	–	1	/TuB/	FALSE
DAT and TXD = 0xC	IDL	–	1	/TuC/	FALSE
DAT and TXD = 0xD	IDL	–	1	/TuD/	FALSE
DAT and TXD = 0xE	IDL	–	1	/TuE/	FALSE
DAT and TXD = 0xF	IDL	–	1	/TuF/	FALSE
ERR after !IDL	IDL	–	1	/E/	TRUE
IDL after IDL	IDL	TRUE	1	/Tu0/	FALSE
ERR after !IDL	!IDL	–	1	/E/	FALSE
DAT after !IDL	ERR	–	1	/E/	FALSE
~~ALPI after IDL~~	~~ALPI~~	~~FALSE~~	~~1~~	~~/L/~~	~~FALSE~~
ARF after IDL	ARF	FALSE	1	/Q/	FALSE
~~NIF after IDL~~	~~IDL~~	~~FALSE~~	~~1~~	~~/I/~~	~~FALSE~~
IDL after IDL	NIF → IDL	FALSE	1	/I/	FALSE
DAT after !IDL TOCT<3:0> = TXD	DAT TOCT<7:4> = TXD	FALSE	0	Data Value	FALSE
Otherwise		–	1	/I/	FALSE

TOCT Symbol to TOCT Value Mapping

- Table 190-3 is the mapping of the symbolic representation to the numerical value of TOCT

- **Changes from draft 1.0**
  - /Ix/ and /E/ share code 0x10
    - /Ix/ can never be transmitted before tx_mode is SEND_N
    - /E/ can never be transmitted after tx_mode is SEND_N
  - /R/ is a new code for Assert remote fault and uses 0x0C
    - Previously used by /Ix/
  - /Ix/ is now defined as Assert PHY not ready
  - Assert LPI is uses the symbol /LI/ to be consistent with previous clauses, which use /L/ for local fault
  - /Q/ is now longer used, keeping code 0x00 reserved

Table 190–3— TOCT symbol to TOCT value mapping

TOCT Symbol	Definition	TOCT Value
/Ix/	Assert PHY not ready	0x10
/E/	Transmit Error Propagation	
/I/	Normal Inter-Frame	0x08
/Su/	Start of packet on odd nibble	0x18
/Tp/	End of packet after odd nibble	0x04
/LI/	Assert LPI	0x14
/R/	Assert Remote Fault	0x0C
/Sp/	Start of packet on even nibble	0x1C
/Tu0/	End of packet after even nibble, last data nibble = 0x0	0x01
/Tu8/	End of packet after even nibble, last data nibble = 0x8	0x11
/Tu4/	End of packet after even nibble, last data nibble = 0x4	0x09
/TuC/	End of packet after even nibble, last data nibble = 0xC	0x19
/Tu2/	End of packet after even nibble, last data nibble = 0x2	0x05
/TuA/	End of packet after even nibble, last data nibble = 0xA	0x15
/Tu6/	End of packet after even nibble, last data nibble = 0x6	0x0D
/TuE/	End of packet after even nibble, last data nibble = 0xE	0x1D
/Tu1/	End of packet after even nibble, last data nibble = 0x1	0x03
/Tu9/	End of packet after even nibble, last data nibble = 0x9	0x13
/Tu5/	End of packet after even nibble, last data nibble = 0x5	0x0B
/TuD/	End of packet after even nibble, last data nibble = 0xD	0x1B
/Tu3/	End of packet after even nibble, last data nibble = 0x3	0x07
/TuB/	End of packet after even nibble, last data nibble = 0xB	0x17
/Tu7/	End of packet after even nibble, last data nibble = 0x7	0x0F
/TuF/	End of packet after even nibble, last data nibble = 0xF	0x1F

Table 199–3— TOCT symbol to TOCT value mapping

TOCT Symbol	Definition	TOCT Value
~~/Q/~~	~~Assert remote fault~~	~~0x00~~
/E/	Transmit Error Propagation	0x10
/I/	Normal Inter-Frame, ~~loc_phy_ready = OK~~	0x08
/Su/	Start of packet on odd nibble	0x18
/Tp/	End of packet after odd nibble	0x04
/L/	Assert LPI	0x14
/Ix/	~~Normal Inter-Frame, loc_phy_ready = NOT_OK~~	~~0x0C~~
/Sp/	Start of packet on even nibble	0x1C
/Tu0/	End of packet after even nibble, last data nibble = 0x0	0x01
/Tu8/	End of packet after even nibble, last data nibble = 0x8	0x11
/Tu4/	End of packet after even nibble, last data nibble = 0x4	0x09
/TuC/	End of packet after even nibble, last data nibble = 0xC	0x19
/Tu2/	End of packet after even nibble, last data nibble = 0x2	0x05
/TuA/	End of packet after even nibble, last data nibble = 0xA	0x15
/Tu6/	End of packet after even nibble, last data nibble = 0x6	0x0D
/TuE/	End of packet after even nibble, last data nibble = 0xE	0x1D
/Tu1/	End of packet after even nibble, last data nibble = 0x1	0x03
/Tu9/	End of packet after even nibble, last data nibble = 0x9	0x13
/Tu5/	End of packet after even nibble, last data nibble = 0x5	0x0B
/TuD/	End of packet after even nibble, last data nibble = 0xD	0x1B
/Tu3/	End of packet after even nibble, last data nibble = 0x3	0x07
/TuB/	End of packet after even nibble, last data nibble = 0xB	0x17
/Tu7/	End of packet after even nibble, last data nibble = 0x7	0x0F
/TuF/	End of packet after even nibble, last data nibble = 0xF	0x1F

Python Code for 8N/(8N+1) Block Encoding

- Python code
  - The 8N/(8N+1) encoder should produce the same result as the following code
  - This is the exact same as draft 1.0

```
# Definition of variables:
#
# Inputs:
# N        Integer set to 2 when RS-FEC is disabled, or 8, when
#          RS-FEC is enabled.
# TS       List-like object of length N. TS[i] is 1 if TOCT[i] is
#          a control symbol and is 0 otherwise.
# TOCT     List-like object containing N integers, each in the
#          range 0 - 255.
#
# Output:
# tx_coded List containing 8N+1 integers, each having value 0 or 1.

for i in range(N):
  if i > 0:
    TS_prev = TS[i-1]
  else:
    TS_prev = 1

  TOR = int(1 in TS[i:])

  if i < N-1:
    TOR_next = int(1 in TS[i+1:])
  else:
    TOR_next = 0

  if TS_prev and TOR:
    TPTR = i + TS[i:].index(1)

  if TS[i]:
    if (TOCT[i] & 0x01):
      TCTL = (TOCT[i] & 0x1F)
    else:
      TCTL = (TOCT[i] & 0x1C) | (TOR_next << 1)

  if TOR:
    tx_octet = 0

    if TS_prev:
      tx_octet |= TPTR
    else:
      tx_octet |= ((TOCT[i-1] >> 5) & 0x07)

    if TS[i]:
      tx_octet |= (TCTL << 3)
    else:
      tx_octet |= ((TOCT[i] & 0x1F) << 3)
  else:
    tx_octet = TOCT[i]

  if i == 0:
    tx_coded = [TOR]

  for bit_indx in range(8):
    tx_coded.append((tx_octet >> bit_indx) & 0b1)
```

Note on Synchronization - Summary of the Changes

- We now have synchronization whether the RS supports remote fault signaling or not
  - And will seamlessly go from PHY not ready to remote fault to idle

    Local Fault ➔ Remote Fault ➔ Idle

Proposed Text for the Draft

- The following slides include the proposed changes to the text of draft 1.0 required for the draft
  - A number of sections, tables and state diagrams are completely new and are shown inside a blue outline text box with a 'New Text' or 'New Figure' or 'New Table' pointer
  - Some sections have existing text that has been completely rewritten, in these cases the new text (inside) is shown side by side with the original text inside a red outline text box
  - Original text boxes have a header *IEEE P802.3dg™/D1.0, 29th April 2025*
    - Original text included for reference that is being kept, is shown inside a brown outline text box
- Note that there are additional Tables and Figures
  - The same numbering is used for existing tables and figures as used in draft 1.0 so that reference to existing draft 1.0 tables and figures is easier
  - New proposed figures have been given numbers 5p5, 13A, 13B and 13C , so that figures 14 and above have the same figure numbers as draft 1.0
  - The tables and figures will be renumbered in the update to the draft
  - Reference to table or figures whose number will change in the draft are show in red

Proposed Text for the Draft – PMA service interface

- Text changes for section 190.2.2.11 PMA_REMPHYIDLE.request
  - Add or /R/ at the end of the last paragraph in 190.2.2.11.1 Semantics of the primitive

New Text

190.2.2.11 PMA_REMPHYIDLE.request

This primitive is generated by the PCS Receive function to indicate whether a sequence of symbols representing idle mode is detected.

190.2.2.11.1 Semantics of the primitive

PMA_REMPHYIDLE.request(rem_phy_idle)

The rem_phy_idle parameter can take on one of two values: TRUE or FALSE.

TRUE	A sequence of symbols representing idle mode is detected.
FALSE	A sequence of symbols representing idle mode is not detected.

The rem_phy_idle parameter is set to FALSE when the pcs_rx_mode parameter is TRAIN. A sequence of symbols representing idle mode is detected if 256 consecutive control symbols are received, each of which is either /I/, /Ix/ or /R/.

IEEE P802.3dg™/D0.3, 5th March 2025

199.2.2.11 PMA_REMPHYIDLE.request

This primitive is generated by the PCS Receive function to indicate whether a sequence of symbols representing idle mode is detected.

199.2.2.11.1 Semantics of the primitive

PMA_REMPHYIDLE.request(rem_phy_idle)

The rem_phy_idle parameter can take on one of two values: TRUE or FALSE.

TRUE	A sequence of symbols representing idle mode is detected.
FALSE	A sequence of symbols representing idle mode is not detected.

The rem_phy_idle parameter is set to FALSE when the pcs_rx_mode parameter is TRAIN. A sequence of symbols representing idle mode is detected if 256 consecutive control octets are received, each of which is either /I/ or /Ix/ as specified in 199.3.3.4.

Note, the above change to add /R/ is already in draft 1.0

Proposed Text for the Draft – PMA service interface

- Text changes for section 190.2.2.13 PMA_REMPHYREADY.request
  - Add or /R/ at the end of the last paragraph in 190.2.2.13.1 Semantics of the primitive

New Text

190.2.2.13 PMA_REMPHYREADY.request

This primitive is generated by the PCS Receive function to indicate whether a sequence of symbols representing idle mode with PHY ready encoding is detected.

190.2.2.13.1 Semantics of the primitive

PMA_REMPHYREADY.request(rem_phy_ready)

The rem_phy_ready parameter can take on one of two values: TRUE or FALSE.

TRUE	A sequence of symbols representing idle mode with PHY ready encoding is detected.
FALSE	A sequence of symbols representing idle mode with PHY ready encoding is not detected.

The rem_phy_ready parameter is set to FALSE when the pcs_rx_mode parameter is TRAIN. A sequence of symbols representing idle mode with PHY ready encoding is detected if 4 consecutive control characters are received, each of which is either /I/ or /R/.

IEEE P802.3dg™/D0.3, 5th March 2025

199.2.2.13 PMA_REMPHYREADY.request

This primitive is generated by the PCS Receive function to indicate whether a sequence of symbols representing idle mode with PHY ready encoding is detected.

199.2.2.13.1 Semantics of the primitive

PMA_REMPHYREADY.request(rem_phy_ready)

The rem_phy_ready parameter can take on one of two values: TRUE or FALSE.

TRUE	A sequence of symbols representing idle mode with PHY ready encoding is detected.
FALSE	A sequence of symbols representing idle mode with PHY ready encoding is not detected.

The rem_phy_ready parameter is set to FALSE when the pcs_rx_mode parameter is TRAIN. A sequence of symbols representing idle mode with PHY ready encoding is detected if 4 consecutive control characters are received, each of which is /I/.

Note, the above change is already in draft 1.0

Proposed Text for the Draft – PCS Transmit function

- Delete section 190.3.2 PCS Data Transmission Enable
  - Delete section 190.3.2 on page 57, lines 1 to 11

Delete section

IEEE P802.3dg™/D1.0, 29th April 2025

190.3.2 PCS Data Transmission Enable

The PCS Data Transmission Enable function shall conform to the PCS data transmission enabling state diagram in Figure 190–4. When tx_mode is equal to SEND_N, the signals tx_enable and tx_error are equal to the values of the MII signals TX_EN and TX_ER respectively; otherwise, tx_enable and tx_error are set to the value FALSE.

Editor's Note (to be removed prior to Working Group Ballot):

Figure 199-3 needs to be updated.

Proposed Text for the Draft – PCS Transmit function

- Text changes for section 190.3.3.1 Use of blocks
  - Rename section to 190.3.3.1 Use of (8N + 1)B blocks and PCS frames
  - Replace text on page 58, lines 14 to 21

190.3.3.1 Use of (8N + 1)B blocks and PCS frames

The PCS Transmit function maps MII signals into (8N + 1)B blocks inserted into a PCS frame, optionally using an RS-FEC coding scheme. The PCS Transmit function generates (8N + 1)B blocks and PCS frames as provided by the rules in 190.3.3.4, 190.3.3.6 and optionally 190.3.3.7. When RS-FEC is disabled, N is 2 and the PCS frame consists of 256 bits. When RS-FEC is enabled, N is 8 and the PCS frame consists of 1,024 bits. Each octet in the PCS frame is encoded as a code-group consisting of 6 PAM3 symbols.

The formatted training frame described in 190.3.5.2 allows establishment of PCS frame and (8N + 1)B boundaries by the PCS Synchronization process. Each nibble in the PMA training frame is encoded as a code-group consisting of 6 PAM2 symbols.

A PCS partial frame is comprised of 32 code-groups where each code-group consists of 6 transmit symbols. The PCS partial frame period corresponds to 192 transmit symbol periods independently of whether RS-FEC is enabled and independently of whether PAM2 or PAM3 symbols are transmitted. The relationship between the PCS partial frame, the PCS frame when RS-FEC is disabled, the PCS frame when RS-FEC is enabled, and the formatted training frame is shown in Figure 190–8. In LPI transmit mode, PCS partial frame boundaries delimit sleep, quiet, refresh, alert, and wake cycles.

(8N + 1)B blocks and PCS frames are unobservable and have no meaning outside the PCS.

IEEE P802.3dg™/D1.0, 29th April 2025

190.3.3.1 Use of blocks

The PCS maps MII signals into (8N)B/(8N + 1)B blocks inserted into a PCS frame, optionally using an RS-FEC coding scheme. The PAM2 PMA training frame synchronization allows establishment of PCS frame and (8N + 1)B boundaries by the PCS Synchronization process. Blocks and frames are unobservable and have no meaning outside the PCS. During the LPI mode, PCS partial frame boundaries delimit sleep, wake, refresh, and quiet cycles. The PCS Transmit function generates blocks and frames as provided by the rules in 190.3.3.4, 190.3.3.6, and optionally 190.3.3.7.

Proposed Text for the Draft – PCS Transmit function

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

- Text changes for section 190.3.3.4 Block structure
  - Rename section to 190.3.3.4 Block encoding

Rename section

190.3.3.4 Block encoding

Blocks consist of 8N + 1 bits. The first bit of a block is the control flag. Blocks are either data blocks or

  - Delete text on page 60, lines 33 to 34

Delete text

IEEE P802.3dg™/D1.0, 29th April 2025

The category ALPI is only used when EEE is enabled for the link. Otherwise Assert LPI signaling is treated as normal inter-frame signaling.

33
34
35

  - Update Table 190–1—MII transfer categories

New Table

Table 190–1— MII transfer categories

Category	TX_EN	TX_ER	TXD<3:0>	Description
DAT	1	0	–	Normal data transmission
ERR	1	1	–	Transmit error propagation
ARF	0	1	0100	Assert remote fault
IDL	0	–	–	Idle signaling

IEEE P802.3dg™/D1.0, 29th April 2025

Table 190–1—MII transfer categories

Category	tx_enable	tx_error	TXD<3:0>	Description
	loc_phy_ready = FALSE			
NOT_RDY	—	—	—	PHY not ready for MII transfer
	loc_phy_ready = TRUE			
DAT	1	0	—	Normal data transmission
ERR	1	1	—	Transmit error propagation
NIF	0	0	—	Normal inter-frame
ALPI	0	1	0001	Assert LPI
ARF	0	1	0100	Assert remote fault
IDL	0	—	—	

Proposed Text for the Draft – PCS Transmit function

- Text changes for section 190.3.3.4 Block encoding
  - Replace text on page 61, lines 1 to 2

IEEE P802.3dg™/D1.0, 29th April 2025

An MII transfer may belong to more than one of the categories in Table 190–1. For example, normal inter-frame signaling belongs to the categories NIF and IDL.

New Text

In Table 190–1 any MII transfer for which TX_EN is 0, including Assert LPI and Assert remote fault, is categorized as IDL. An MII transfer may belong to more than one of the categories in Table 190–1. For example, Assert remote fault belongs to the categories ARF and IDL.

  - New table, Table 190–2— MII transfer to TS and TOCT mapping

New Table

Table 190–2— MII transfer to TS and TOCT mapping

Previous transfer	Even transfer	Odd transfer	Current dly_enc	TS	TOCT	Next dly_enc
IDL	DAT	!ERR	–	1	/Sp/	FALSE
IDL	DAT	ERR	–	1	/Sp/	TRUE
IDL	ERR	–	–	1	/Sp/	TRUE
–	IDL	DAT	–	1	/Su/	FALSE
–	IDL	ERR	–	1	/Su/	TRUE
!IDL	DAT	DAT	TRUE	1	/E/	FALSE
!IDL	IDL	–	–	1	/Tp/	FALSE
!IDL	DAT, TXD = 0x0	IDL	–	1	/Tu0/	FALSE
!IDL	DAT, TXD = 0x1	IDL	–	1	/Tu1/	FALSE
!IDL	DAT, TXD = 0x2	IDL	–	1	/Tu2/	FALSE
!IDL	DAT, TXD = 0x3	IDL	–	1	/Tu3/	FALSE
!IDL	DAT, TXD = 0x4	IDL	–	1	/Tu4/	FALSE
!IDL	DAT, TXD = 0x5	IDL	–	1	/Tu5/	FALSE
!IDL	DAT, TXD = 0x6	IDL	–	1	/Tu6/	FALSE
!IDL	DAT, TXD = 0x7	IDL	–	1	/Tu7/	FALSE
!IDL	DAT, TXD = 0x8	IDL	–	1	/Tu8/	FALSE
!IDL	DAT, TXD = 0x9	IDL	–	1	/Tu9/	FALSE
!IDL	DAT, TXD = 0xA	IDL	–	1	/TuA/	FALSE
!IDL	DAT, TXD = 0xB	IDL	–	1	/TuB/	FALSE
!IDL	DAT, TXD = 0xC	IDL	–	1	/TuC/	FALSE
!IDL	DAT, TXD = 0xD	IDL	–	1	/TuD/	FALSE
!IDL	DAT, TXD = 0xE	IDL	–	1	/TuE/	FALSE
!IDL	DAT, TXD = 0xF	IDL	–	1	/TuF/	FALSE
!IDL	ERR	IDL	–	1	/E/	TRUE
IDL	IDL	IDL	TRUE	1	/Tu0/	FALSE
!IDL	ERR	!IDL	–	1	/E/	FALSE
!IDL	DAT	ERR	–	1	/E/	FALSE
IDL	ARF	ARF	FALSE	1	/R/	FALSE
IDL	IDL	IDL	FALSE	1	/I/	FALSE
!IDL	DAT TOCT<3:0> = TXD	DAT TOCT<7:4> = TXD	FALSE	0	Data Value	FALSE
Otherwise				1	/I/	FALSE

Proposed Text for the Draft – PCS Transmit function

- Text changes for section 190.3.3.4 Block encoding
  - New table, Table 190–3— TOCT symbol to TOCT value mapping

Table 190–3— TOCT symbol to TOCT value mapping

TOCT Symbol	Definition	TOCT Value
/Ix/	Assert PHY not ready	0x10
/E/	Transmit Error Propagation	
/I/	Normal Inter-Frame	0x08
/Su/	Start of packet on odd nibble	0x18
/Tp/	End of packet after odd nibble	0x04
/LI/	Assert LPI	0x14
/R/	Assert Remote Fault	0x0C
/Sp/	Start of packet on even nibble	0x1C
/Tu0/	End of packet after even nibble, last data nibble = 0x0	0x01
/Tu8/	End of packet after even nibble, last data nibble = 0x8	0x11
/Tu4/	End of packet after even nibble, last data nibble = 0x4	0x09
/TuC/	End of packet after even nibble, last data nibble = 0xC	0x19
/Tu2/	End of packet after even nibble, last data nibble = 0x2	0x05
/TuA/	End of packet after even nibble, last data nibble = 0xA	0x15
/Tu6/	End of packet after even nibble, last data nibble = 0x6	0x0D
/TuE/	End of packet after even nibble, last data nibble = 0xE	0x1D
/Tu1/	End of packet after even nibble, last data nibble = 0x1	0x03
/Tu9/	End of packet after even nibble, last data nibble = 0x9	0x13
/Tu5/	End of packet after even nibble, last data nibble = 0x5	0x0B
/TuD/	End of packet after even nibble, last data nibble = 0xD	0x1B
/Tu3/	End of packet after even nibble, last data nibble = 0x3	0x07
/TuB/	End of packet after even nibble, last data nibble = 0xB	0x17
/Tu7/	End of packet after even nibble, last data nibble = 0x7	0x0F
/TuF/	End of packet after even nibble, last data nibble = 0xF	0x1F

New Table

Proposed Text for the Draft – PCS Transmit function

- Text changes for section 190.3.3.5 Control codes
  - Delete paragraph on page 65, lines 14 to 18

IEEE P802.3dg™/D1.0, 29th April 2025

190.3.3.5 Control codes

A subset of control characters defined at the MII is supported by the 100BASE-T1L PCS. Supported control characters at the MII are conveyed as control codes in the (8N)B/(8N+1)B block. For some control characters at the MII one of two different control codes is conveyed depending on whether more control codes follow in the (8N)B/(8N+1)B block. For other control characters only one control code is used since the presence of more control codes in the (8N)B/(8N+1)B block may be inferred.

Delete paragraph

When TX_EN is not asserted and no supported control code is present at the MII, the 100BASE-T1L PCS conveys a Normal Inter-Frame symbol code in the (8N)B/(8N+1)B block. When the loc_phy_ready parameter is TRUE, the symbol conveyed is /I/ and, when the loc_phy_ready parameter is FALSE, the symbol conveyed is /Ix/.

Proposed Text for the Draft - PCS Transmit function

- Text changes for section 190.3.3.4 Block encoding
  - Add new sections 190.3.3.5.1 Assert PHY not ready and 190.3.3.5.2 Normal Inter-Frame after section 190.3.3.5 Control codes

IEEE P802.3dg™/D1.0, 29th April 2025

190.3.3.5 Control codes

A subset of control characters defined at the MII is supported by the 100BASE-T1L PCS. Supported control characters at the MII are conveyed as control codes in the (8N)B/(8N+1)B block. For some control characters at the MII one of two different control codes is conveyed depending on whether more control codes follow in the (8N)B/(8N+1)B block. For other control characters only one control code is used since

New Sections

190.3.3.5.1 Assert PHY not ready

During link establishment the local PHY signals to the remote PHY to communicate that it is not ready to enter the normal operational mode. The 100BASE-T1L PCS repeatedly conveys the Assert PHY not ready symbol code (/Ix/) in the (8N)B/(8N+1)B block.

190.3.3.5.2 Normal Inter-Frame

When TX_EN is not asserted and no supported control code is present at the MII, the 100BASE-T1L PCS conveys a Normal Inter-Frame symbol code (/I/) in the (8N)B/(8N+1)B block.

Proposed Text for the Draft - PCS Transmit function

- Text changes for section 190.3.3.4 Block encoding
  - Add new sentence to end of 190.3.3.5.1 Assert LPI and increment paragraph numbering

IEEE P802.3dg™/D1.0, 29th April 2025

190.3.3.5.1 Assert LPI

When EEE is enabled for the link, the LPI client may request operation in the LPI transmit mode by setting TX_EN to 0, and TX_ER to 1, and TXD<3:0> to 0001. The 100BASE-T1L PCS conveys an Assert LPI symbol (/L/) in the (8N)B/(8N+1)B block.

Inc to 190.3.3.5.3

190.3.3.5.3 Assert LPI

When EEE is enabled for the link, the LPI client may request operation in the LPI transmit mode by setting TX_EN to 0, and TX_ER to 1, and TXD<3:0> to 0001. The 100BASE-T1L PCS repeatedly conveys the Assert LPI symbol (/LI/) in the (8N)B/(8N+1)B block to indicate to the link partner that it is transitioning to the LPI transmit mode.

New Text

Proposed Text for the Draft - PCS Transmit function

- Text changes for section 190.3.3.4 Block encoding
  - Add new section 190.3.3.5.4 Assert remote fault after section 190.3.3.5.3 Assert LPI

IEEE P802.3dg™/D1.0, 29th April 2025

190.3.3.5.3 Assert LPI

When EEE is enabled for the link, the LPI client may request operation in the LPI transmit mode by setting TX_EN to 0, and TX_ER to 1, and TXD<3:0> to 0001. The 100BASE-T1L PCS repeatedly conveys the Assert LPI symbol (/LI/) in the (8N)B/(8N+1)B block to indicate to the link partner that it is transitioning to the LPI transmit mode.

New Section

190.3.3.5.4 Assert remote fault

During link establishment the local RS may wish to send a remote fault indication to the remote RS to communicate that it is not ready to receive data. In this case the RS may signal Assert remote fault by setting TX_EN to 0, TX_ER to 1, and TXD<3:0> to 0100. The 100BASE-T1L PCS conveys a remote fault symbol code (/R/) in the (8N)B/(8N+1)B block.

Proposed Text for the Draft – PCS Transmit function

- Reorder section 190.3.3.5.2 Transmit Error Propagation to be after section 190.3.3.5.1 Assert LPI and before section 190.3.3.5.3 Start

New Order

190.3.3.5.4 Assert remote fault

During link establishment the local RS may wish to send a remote fault indication to the remote RS to communicate that it is not ready to receive data. In this case the RS may signal Assert remote fault by setting TX_EN to 0, TX_ER to 1, and TXD<3:0> to 0100. The 100BASE-T1L PCS conveys a remote fault symbol code (/R/) in the (8N)B/(8N+1)B block.

190.3.3.5.5 Start

When TX_EN switches from 0 to 1, the 100BASE-T1L PCS conveys a Start symbol code in the (8N)B/(8N+1)B block. When the first nibble after TX_EN switches is an even nibble, the symbol conveyed is /Sp/ and, when it is an odd nibble, the symbol conveyed is /Su/.

190.3.3.5.6 Terminate

When TX_EN switches from 1 to 0, the 100BASE-T1L PCS conveys a Terminate symbol code in the (8N)B/(8N+1)B block. When the first nibble after TX_EN switches is an even nibble, the symbol conveyed is /Tp/ and, when it is an odd nibble, the symbol conveyed is one of the symbols /Tu0/ to /TuF/, depending on the value presented at TXD<3:0> during the corresponding even nibble.

190.3.3.5.7 Transmit Error Propagation

During the process of transmitting a frame, the RS may require that the PHY deliberately corrupt the contents of the frame in such a manner that a receiver will detect the corruption with the highest degree of probability. In this case the RS may request Transmit Error Propagation by setting TX_EN to 1 and TX_ER to 1. The 100BASE-T1L PCS conveys a Transmit Error Propagation symbol (/E/) in the (8N)B/(8N+1)B block.

190.3.3.6 Transmit process

The transmit process generates blocks based upon the TXD<3:0>, TX_EN, and TX_ER signals received

IEEE P802.3dg™/D1.0, 29th April 2025

190.3.3.5.1 Assert LPI

When EEE is enabled for the link, the LPI client may request operation in the LPI transmit mode by setting TX_EN to 0, and TX_ER to 1, and TXD<3:0> to 0001. The 100BASE-T1L PCS conveys an Assert LPI symbol (/L/) in the (8N)B/(8N+1)B block.

190.3.3.5.2 Transmit Error Propagation

During the process of transmitting a frame, the RS may require that the PHY deliberately corrupt the contents of the frame in such a manner that a receiver will detect the corruption with the highest degree of probability. In this case the RS may request Transmit Error Propagation by setting TX_EN to 1 and TX_ER to 1. The 100BASE-T1L PCS conveys a Transmit Error Propagation symbol (/E/) in the (8N)B/(8N+1)B block.

190.3.3.5.3 Start

When TX_EN switches from 0 to 1, the 100BASE-T1L PCS conveys a Start symbol code in the (8N)B/(8N+1)B block. When the first nibble after TX_EN switches is an even nibble, the symbol conveyed is /Sp/ and, when it is an odd nibble, the symbol conveyed is /Su/.

190.3.3.5.4 Terminate

When TX_EN switches from 1 to 0, the 100BASE-T1L PCS conveys a Terminate symbol code in the (8N)B/(8N+1)B block. When the first nibble after TX_EN switches is an even nibble, the symbol conveyed is /Tp/ and, when it is an odd nibble, the symbol conveyed is one of the symbols /Tu0/ to /TuF/, depending on the value presented at TXD<3:0> during the corresponding even nibble.

190.3.3.6 Transmit process

The transmit process generates blocks based upon the TXD<3:0>, TX_EN, and TX_ER signals received

Conclusions

- This presentation presents an update to the text and tables for clause 190.3.3.4 Block structure for the draft
  - Adds desired improvements discussed at the previous meeting
  - Ensures we have a clean fault handling sequence of PHY not ready to remote fault to idle
  - Keep the /Q/ code (0x00) reserved for possible future use

Questions ?