ANALOG DEVICES

AHEAD OF WHAT'S POSSIBLE™

Single 10M or 100M PHY POPI Requirements

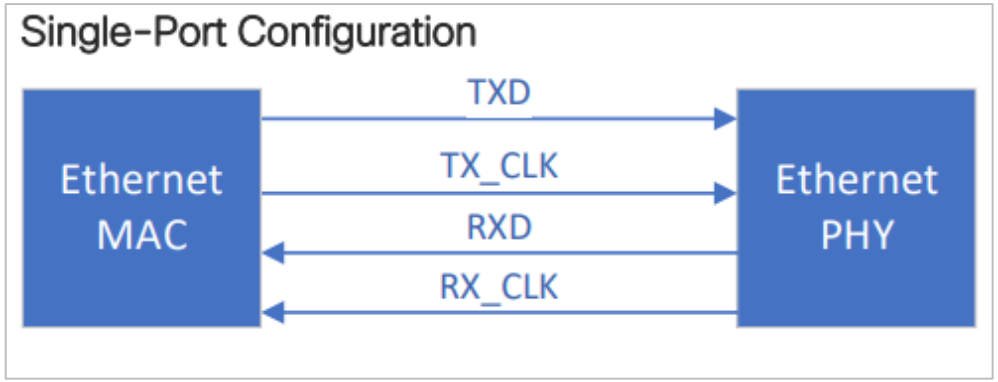
Brian Murray
Jacobo Riesco

Introduction

- This presentation is a continuation of our January presentation to develop the POPI requirements for a 10M or 100M single PHY
  - With particular focus on the 802.3dg 100BASE-T1L PHY standard under development
  - Include the requirements for a full duplex 10BASE-T1L PHY
  - And be compatible with legacy full duplex 100M PHY standards, e.g. 100BASE-TX and 100BASE-T1
- The requirements include the following:
  - The pin interface requirements
  - The embedded interfaces for MDIO and PTP
  - The features that should be supported
- Ensure we follow the same approach being developed for POPI for higher speed PHYS and multi-port PHYs
  - In particular using the same frame formats and control for the MDIO and PTP embedded interfaces

Review of POPI Pin Interface for a 10 or 100M Single PHY

- For a single 10M or 100M PHY the desire is to have a simple low pin count interface using standard IOs and digital logic
  - Support 100M data plus overhead for control, PTP time stamping, MDIO and clause 36 ordered sets
  - Operate at a raw data rate on the pins in the range of 110 – 160 Mb/s
  - Use single ended IOs to avoid the complexity of a SerDes while still keeping to a very low pin count
- Use a simple 4-pin interface with single ended IOs
  - TXD, TX_CLK, RXD, RX_CLK
  - Same pin count as a SerDes (4 pins)
  - Signalling rates of ~125 MHz are similar to existing GE PHY MAC interfaces
  - Clock/data recovery is optional
  - Source synchronous
  - Proven technology, low power/area, available in every technology node
    - An area efficient 1G SerDes might get close to the area of an RGMII MAC interface with 12 pins, but not 4 pins



Pin Optimized PHY Interface, Call For Interest Consensus Building (page 23)

Review of POPI –MDIO Management Interface

- MDIO is a medium bandwidth interfaces that should be embedded with POPI so additional pins between the MAC and PHY are not required
  - The MAC interface typically uses a 2-pin shared bus for the MDIO management interface
    - This is a very low overhead for a multi-port switch or for a legacy RMII/RGMII MAC interface
    - However, for a single PHY using POPI this would represent 50% more pins
- Clause 22 specifies an interface speed of 2.5 MHz for MDIO, many PHYs support speeds up to 6.25 MHz
  - Sometimes even higher MDIO speeds are used for multi-port PHYs or for PTP requirements
  - However, with POPI each port would have its own dedicated MDIO and PTP embedded interface
- Hence, an embedded MDIO interface supporting 2.5 to 6.25 Mb/s should be sufficient for a single 10M or 100M PHY

Single PHY POPI – PTP Timestamping

- PTP timestamping is also a medium bandwidth interface that should be embedded with POPI so additional pins are not required
  - PTP timestamping is sometimes supported using two additional pins (Tx and Rx Start of packet pins) or is supported with register operations over the MDIO
- PTP timestamping could require 4 bytes for a 64 byte packet
  - On the receive side a timestamp is usually sent from the PHY to the MAC for every packet
    - This covers MACsec where the payload is encrypted and the PTP packets cannot be identified
  - On the transmit side only some PTP packets require timestamp information to be sent from the PHY back to the MAC
    - But some application use cases could require a significant percentage of packets to be timestamped
- This suggests an upper bound of ~ 8 Mb/s for 100M PTP timestamping
  - 6.25 Mb/s for the receive packets and 25% of 6.25 Mb/s for the transmit packets
- Hence, a shared embedded MDIO/PTP interface supporting 10 Mb/s should be sufficient for a single 100M PHY and 2.5 Mb/s for a single 10M PHY
  - For a 100M PHY this would support 10 Mb/s write operations and 2 Mb/s read operations
  - For a 10M PHY this would support 2.5 Mb/s write operations and 1.5 Mb/s read operations

Single PHY POPI – Embedded Interfaces

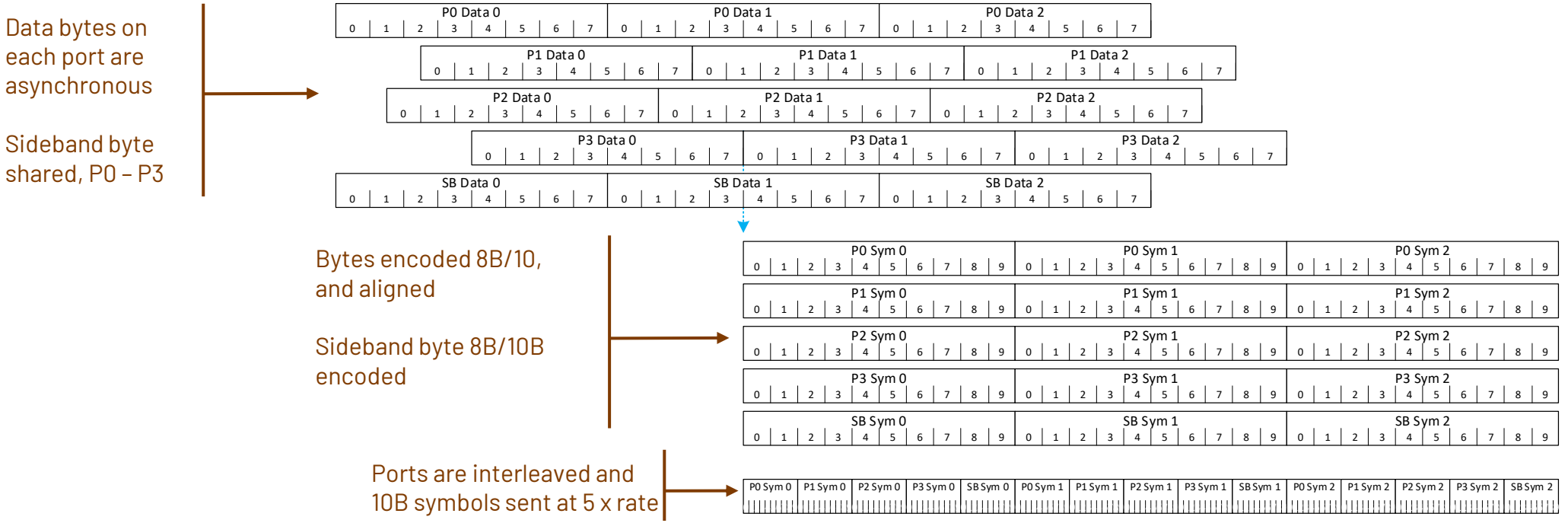
- There is a strong preference to embed MDIO, PTP, link information and clause 36 like ordered sets with POPI to achieve the lowest pin count interface
- Some of these embedded interfaces can operates asynchronous to the 100M data interface as they do not have any critical latency requirements
  - They operate today at the speed of packets not bytes or bits
  - These interfaces as implemented today have no error protection or CRC, which should be addressed in POPI
- The PTP interface does not have critical latency requirements, like the payload data; but it is much more sensitive to delay than MDIO
  - Ideally, the timestamp information is available to the MAC close to the start of the receive packet and without too large a delay after transmitting a packet to be timestamped
  - The overhead margin available on the PTP interface is very dependent on the worst case conditions, e.g. all 64-byte packets and a higher frequency of Tx packet timestamping
  - With 64-byte packets, the timestamp information has to be sent more frequently
- POPI should support 10 Mb/s or more for a sideband channel to support all 3 of these interfaces; MDIO, PTP and link information
- The sideband channel will required clause 36 like framing, for start and end of packet, packet type, CRC (or other error protection), etc.
  - This will most likely push the required raw sideband bit rate beyond 12.5 Mb/s, e.g. > payload x 1/8

Single PHY POPI – Low Latency Data Requirement

- The 10/100/1000M legacy low pin count interfaces make a new data nibble or byte available to the PCS each cycle for encoding and transfer to the line
  - At 100M using MII or RMII a new data nibble is available to the PCS every 25 MHz cycle
  - At 1G using GMII or SGMII a new data byte is available to the PCS every 125 MHz cycle
  - This one to one mapping between a byte on the MII interface and a byte at the PCS minimizes the latency compared to cases where higher order block codes like 64B/66B are used
  - At low speeds the latency of each bit is much more significant than at Multi-Gig speeds
- For a single 10 or 100M PHY POPI should use a block encoding that ensures a new data byte is available each cycle for encoding and transfer to the line
  - For example 8B/10B or similar low order block encoding could be used by POPI to achieve this
    - Each payload data byte is encoded as a 10B symbol
  - At 100M a new data byte can be made available to the PHY PCS every 12.5 MHz cycle
- POPI must run at a higher rate than the payload rate to support additional bits or bytes for the PTP and MDIO sideband channel
  - Must do this with minimum additional latency to support the interleaving of the sideband bits/bytes

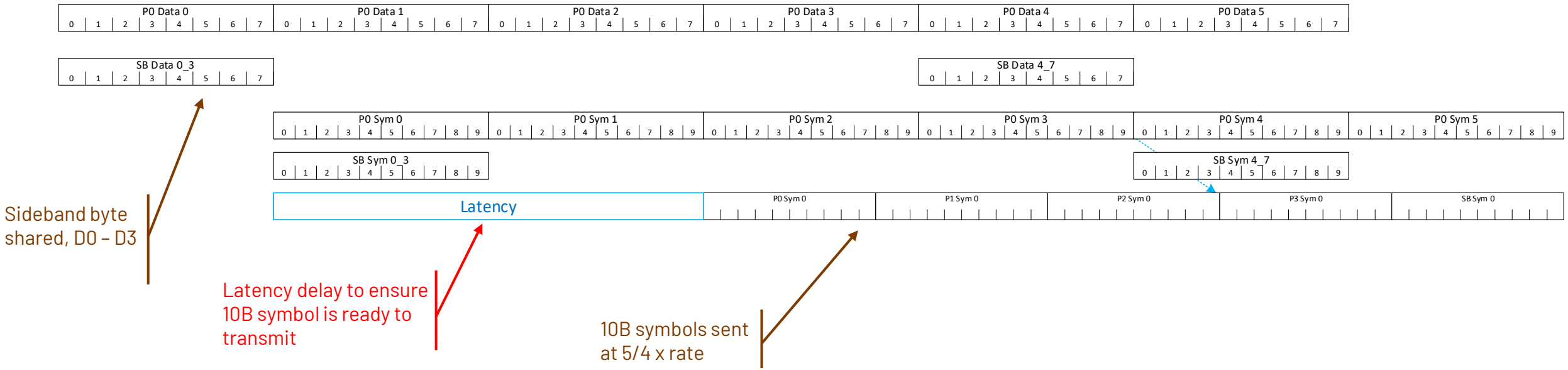
Single PHY POPI – Options for Low Latency

- For a multi-port PHY, POPI can share the sideband channel over a number of ports
  - One obvious choice is sharing a sideband byte over 4 ports
  - A quad is a very common sub-multiple, and this gives us 25% overhead (156.25 Mb/s per port)
  - So we would run the POPI interface 25% times faster than the 8B/10B encoded payload rate
    - The 4 data bytes are each encoded as 8B/10B and sent first, followed by the sideband byte as a 10B symbol
    - With no latency cost, compared to the case of no sideband channel



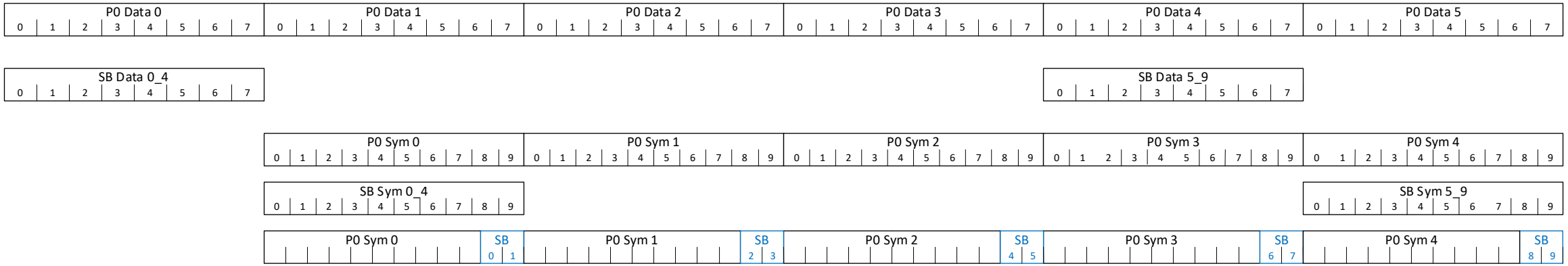
Single PHY POPI – Options for Low Latency

- For a single PHY, supporting an overhead of 25%
  - POPI would operate 25% times faster than the 8B/10B encoded payload rate (156.25 Mb/s)
    - Send a sideband byte as a 10B symbol, once every 4 data bytes encoded as 8B/10B symbols
  - However interleaving the sideband symbol every 4 data symbols adds latency
    - The payload data must be buffered in advance of the 4 to 5 data rate adaptation



Single PHY POPI – Options for Low Latency

- An alternative is to interleave the sideband 10B symbol, 2 bits at a time over 5 data symbols
  - This would avoid a latency hit and would operate at a 20% higher data rate (150 Mb/s)
  - This is slightly more complicated, but should not be an issue at lower data rates like 100M



  - A new 10 data symbol is available to transmit every cycle
- Other approaches can be explored to manage the trade-off of sideband overhead and POPI signalling rate while minimizing the latency

Summary

- For a single 10M or 100M PHY, POPI should be implemented as a simple 4-pin interface with single ended IOs and digital logic
- Should operate at signalling rates similar to existing GE PHY MAC interfaces, e.g. 125 to 156.25 MHz
- Should provide a new data byte from the MAC interface to the PHY PCS every 12.5 MHz cycle
- Should minimize latency on the data channel required to interleave the data and sideband channels
- Support MDIO, PTP and clause 36 like ordered sets as embedded interfaces with POPI to achieve the lowest pin count interface
- Follow the same logical format for frames used for the embedded interfaces like MDIO and PTP as used by higher speed /multi-port POPI

Questions ?

Backup

Single PHY POPI – Clause 36 Ordered Sets

- Clause 36 ordered sets are used for encapsulation and are also used as a low bandwidth interface
- In SGMII and USGMII clause 36 ordered sets are used for control and configuration
  - E.g. Auto-Negotiation information, error propagation, faults, LPI, etc.
- Clause 36 ordered sets
  - Configuration ordered sets convey information about the link
    - Link-up, link failure, Auto-Negotiation error, FD/HD, Pause
  - Encapsulation ordered sets are used for the start and end of packet, carrier extend and error propagation
  - LPI ordered sets are used for Energy Efficient Ethernet
- Clause 36 ordered sets may have to be amended/supplemented to support PLCA
  - And we may also want to also include clause 46 sequence ordered set for fault signalling

Table 36–3—Defined ordered sets

Code	Ordered Set	Number of Code-Groups	Encoding
/C/	**Configuration**		Alternating /C1/ and /C2/
/C1/	Configuration 1	4	/K28.5/D21.5/Config_Reg[a]
/C2/	Configuration 2	4	/K28.5/D2.2/Config_Reg[a]
/I/	**IDLE**		Correcting /I1/, Preserving /I2/
/I1/	IDLE 1	2	/K28.5/D5.6/
/I2/	IDLE 2	2	/K28.5/D16.2/
	Encapsulation		
/R/	Carrier_Extend	1	/K23.7/
/S/	Start_of_Packet	1	/K27.7/
/T/	End_of_Packet	1	/K29.7/
/V/	Error_Propagation	1	/K30.7/
/LI/	**LPI**		Correcting /LI1/, Preserving /LI2/
/LI1/	LPI 1	2	/K28.5/D6.5/
/LI2/	LPI 2	2	/K28.5/D26.4/

[a]Two data code-groups representing the Config_Reg value.

Config_Reg Base Page bits	Management register bit
Full Duplex (FD)	4.5 Full Duplex
Half Duplex (HD)	4.6 Half Duplex
PAUSE (PS1)	4.7 PAUSE
ASM_DIR (PS2)	4.8 ASM_DIR
Remote Fault (RF2, RF1)	4.13:12 Remote Fault

Table 37–3—Remote Fault encoding

RF1	RF2	Description
0	0	No error, link OK (default)
0	1	Offline
1	0	Link_Failure
1	1	Auto-Negotiation_Error

Single PHY POPI – Features Supported

- In the Study group and the consensus call for interest meeting a number of different features that POPI might support were discussed
  - We have already covered the most important ones of these; MDIO, PTP, LPI
- Half-duplex operation (COL/CRS)
  - Desire here is to support half-duplex PHYs like 10BASE-T1S, 100BASE-TX & 10BASE-T in HD mode
  - All the legacy non-standard MAC interface infer COL/CRS at the MAC side from RX_DV and TX_EN
  - Why do anything different for POPI ?
  - We suggest that we should **not** try to support half-duplex operation within POPI as it adds complexity for no benefit
- PLCA support
  - PLCA defines some extra MII codes, e.g. Commit and Beacon
  - These could be added using clause 36 like ordered sets in POPI
- Two-pair 10/100 PHY compatibility
  - If POPI only supports full-duplex there is nothing else to be done to support two-pair 10/100 PHYs

Table 22–1—Permissible encodings of TXD<3:0>, TX_EN, and TX_ER

TX_EN	TX_ER	TXD<3:0>	Indication
0	0	0000 through 1111	Normal inter-frame
0	1	0000	Reserved
0	1	0001	Assert LPI
0	1	0010	PLCA BEACON request
0	1	0011	PLCA COMMIT request
0	1	0100 through 1111	Reserved
1	0	0000 through 1111	Normal data transmission
1	1	0000 through 1111	Transmit error propagation

Single PHY POPI – Local / Remote Fault

- 802.3dg added Assert Local Fault / Assert Remote Fault to clause 22 to support fault signalling between the PHY and RS
  - And added encoding of Assert Remote Fault in the PCS so that it can be sent from the local RS to the remote RS over the link
  - This was done t add the fault signalling developed in clause 46 for 10G (and Multi-Gig) PHYs to 100BASE-T1L
- Assert Local Fault and Assert Remote Fault should be added using clause 36 like ordered sets in POPI

22.2.2.4 TXD (transmit data)

Change Table 22–1 as shown (unchanged rows not shown):

Table 22–1—Permissible encodings of TXD<3:0>, TX_EN, and TX_ER

TX_EN	TX_ER	TXD<3:0>	Indication
…			
0	1	0100	Assert remote fault
0	1	0100~~0~~1 through 1111	Reserved
…			

22.2.2.8 RXD (receive data)

Change Table 22–2 as shown (unchanged rows not shown):

Table 22–2—Permissible encoding of RXD<3:0>, RX_ER, and RX_DV

RX_DV	RX_ER	RXD<3:0>	Indication
…			
0	1	0100	Assert remote fault
0	1	0101	Assert local fault
0	1	0110~~0~~ through 1101	Reserved
…			

Single PHY POPI – 10BASE-T1S

- It would seem desirable in order to be complete that we should also support 10BASE-T1S, especially as this is a recent PHY and new products could adopt POPI
  - However, it does seem unlikely that a 10BASE-T1S single PHY would be implemented as a stand alone PHY with a MAC interface
  - There is already a 3-pin interface defined to allow the analog (maybe high voltage) portion of the PHY to be implemented stand alone
  - It is more likely that the low voltage digital portion of 10BASE-T1S would be integrated with a MAC
- This suggests, that the 10BASE-T1S requirements should not be a high priority
  - The end result may be usable as a 10BASE-T1S MAC interface anyway