ANALOG DEVICES

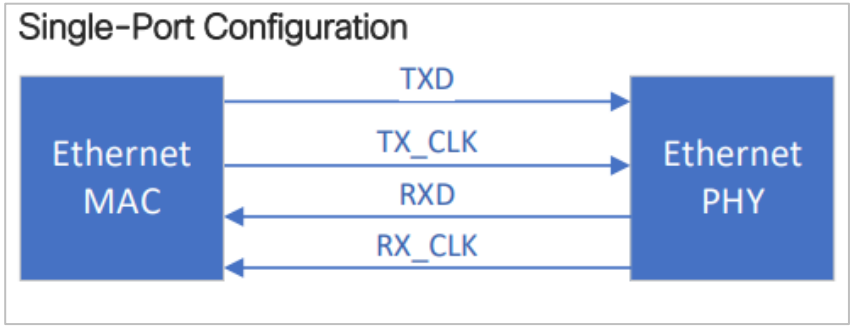
AHEAD OF WHAT'S POSSIBLE™

Proposal for 8B/9B Symbol Encoding for POPI at Rates of 10M to 200M

Brian Murray
Jacobo Riesco

Introduction

- The authors have made a number of presentations on POPI for 10M/100M PHYs
  - Focused on lower speeds using single ended digital IOs
  - Source synchronous with optional clock/data recovery
  - Requirements for low latency, encapsulation and control codes
    - Additional control codes for Local/Remote Fault
    - Options to add control codes for Half-Duplex operation and PLCA
    - Options to add control codes for new features
  - Supporting multi-port PHYs with identification of Port 0 Idle symbols
  - Use scrambling to ensure the spectral integrity of the serial data and for improved EMI
  - Embedded management using a sideband channel and the required data rates
  - Ingress and Egress timestamps/TSSI using the sideband channel including an option to use the IPG for the Ingress timestamp to manage bandwidth requirements
  - Exchange POPI link configuration at power-up/reset for the signalling rate / number of ports
  - Operate in a low power mode when all the PHY links are down



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

Options for Symbol Encoding for POPI

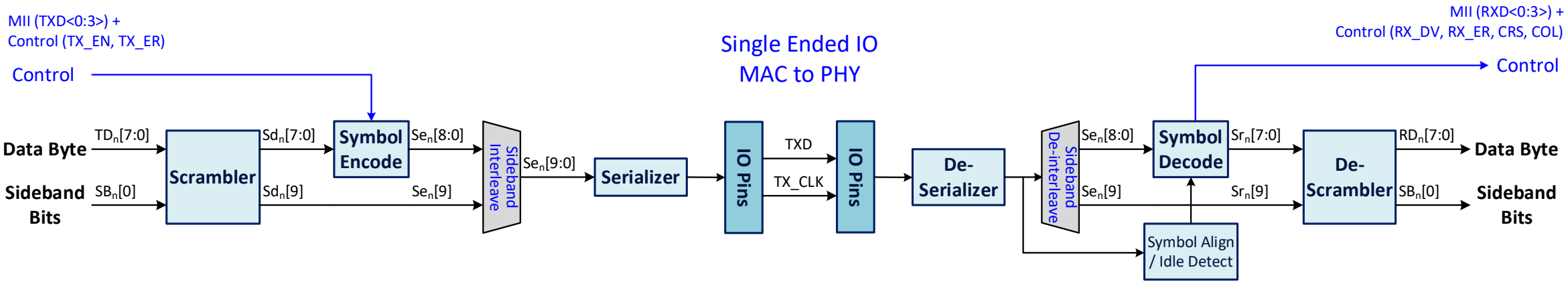
- In July the author gave a presentation considering different options for symbol encoding and the trade-offs of efficiency and latency at lower (< 1G) and higher speeds (≥ 1G)
  - See Symbol Encoding for POPI for 10/100M PHYs
- A proposal was presented for a 100M single PHY using 8B/9B symbol encoding with interleaved sideband bits
  - Operating at a 125 MHz signalling rate supporting a 12.5 Mb/s sideband channel
  - Allowing an implementation of POPI using single ended IO to using the same signalling rate as existing MAC-PHY interfaces
  - Achieving very low latency by mapping each byte at the MII to a byte at the PHY PCS
- This presentation expands on the July presentation to provide a more complete proposal for POPI supporting rates of 10M to 200M for single and multi-port 10M and 100M PHYs

Example Rates for the Low Speed POPI

- This proposal is for PHY rates of 10M to 200M
  - The upper rate of 200M corresponds to signalling rate of 250 MHz at the pins after scrambling, encoding and interleaving of the sideband channel
  - This is the same signalling rate as a standard 1G RGMII interface on each of the data pins
  - And this the rate that is used for a Dual 100M PHY
  - The lower rate of 10M corresponds to a single 10M PHY and a signalling rate of 12.5 MHz
- The following examples of single and multi-port PHYs supported
  - Single 10M PHY at a signalling rate of 12.5 MHz
  - Quad 10M PHY at a signalling rate of 50 MHz
  - Octal 10M PHY at a signalling rate of 100 MHz
  - Single 100M PHY at a signalling rate of 125 MHz
  - Dual 100M PHY at a signalling rate of 250 MHz

Scrambling and 8B/9B Encoding

- 8B/9B encoding is used for data encapsulation for serialization of the data
- The following is a block diagram of the data flow for the interleave of data and sideband bits, scrambling and 8B/9B encoding
  - One sideband bit $SB_n[0]$ is interleaved with each data octet $TD_n[7:0]$ after 8B/9B encoding
  - The 9-tuple data octet + SB bit are scrambled using a standard 33-bit additive scrambler
  - The scrambled 8-bits corresponding to the data, $Sd_n[7:0]$ are encoded 8B/9B as $Se_n[8:0]$
  - Balanced disparity in the Idle and control codes are used to control running disparity (not in data)
  - The encoded 9-bits and scrambled sideband bit $Sd_n[9]$ are serialized as a 10B symbol $Se_n[9:0]$



Scrambler

- Data and training modes use the same 33 bits side-stream scrambler approach used in 100BASE-T1L and many other previous PHY technologies, Clauses 40, 146, 96, 149, 165 and 190
- Generator polynomials for MAC and PHY are:
  - MAC: $g_M(x) = 1 + x^{13} + x^{33}$
  - PHY: $g_P(x) = 1 + x^{20} + x^{33}$
  - Where we are using the Leader/Follower polynomials from previous standards for the MAC/PHY
- Scrambling is done per 9-tuple made up of the sideband bit $SB_n[0]$ and the data octet $TD_n[7:0]$
  - The bits stored in the scrambler shift register delay line at time n are denoted by $Scr_n[32:0]$
  - At each data byte, the shift register is advanced by one bit, and a new bit represented by $Scr_n[0]$ is generated
  - Encoding rules are based on the generation, at time n, of nine bits: $Sx_n[3:0]$, $Sy_n[3:0]$ and Sg_n
    - The eight $Sx_n[3:0]$ and $Sy_n[3:0]$ bits are used to decorrelate the octet $TD_n[7:0]$ during transmission
    - Sg_n is used to randomize the sideband bit $SB_n[0]$

Scrambler Generation of $Sd_n[7:0]$ and Sg_n

- $Sx_n[3:0]$, $Sy_n[3:0]$ and Sg_n are generated using three uncorrelated bits, X_n, Y_n and $Scr_n[0]$, and an auxiliary generator polynomial, g(x), as in Clause 190:
  - The bits X_n and Y_n derived from elements of the same maximum-length shift register sequence of length $2^{33}-1$ as $Scr_n[0]$, but shifted in time
  - **The associated delays are all large and different so that there is no short-term correlation among the bits X_n, Y_n and $Scr_n[0]$**
  - They are generated as follows:

$$X_n = Scr_n[4] \wedge Scr_n[6]$$
$$Y_n = Scr_n[1] \wedge Scr_n[5]$$

  - The generator polynomial is:

$$g(x) = x^3 \wedge x^8$$

  - $Sx_n[3:0]$, $Sy_n[3:0]$ and Sg_n are generated as follows:

$$Sy_n[0] = Scr_n[0]$$
$$Sy_n[1] = g(Scr_n[0]) = Scr_n[3] \wedge Scr_n[8]$$
$$Sy_n[2] = g^2(Scr_n[0]) = Scr_n[6] \wedge Scr_n[16]$$
$$Sy_n[3] = g^3(Scr_n[0]) = Scr_n[9] \wedge Scr_n[14] \wedge Scr_n[19] \wedge Scr_n[24]$$
$$Sg_n = Y_n = Scr_n[1] \wedge Scr_n[5]$$

$$Sx_n[0] = X_n = Scr_n[4] \wedge Scr_n[6]$$
$$Sx_n[1] = g(X_n) = Scr_n[7] \wedge Scr_n[9] \wedge Scr_n[12] \wedge Scr_n[14]$$
$$Sx_n[2] = g^2(X_n) = Scr_n[10] \wedge Scr_n[12] \wedge Scr_n[20] \wedge Scr_n[22]$$
$$Sx_n[3] = g^3(X_n) = Scr_n[13] \wedge Scr_n[15] \wedge Scr_n[18] \wedge Scr_n[20] \wedge Scr_n[23] \wedge Scr_n[25] \wedge Scr_n[28] \wedge Scr_n[30]$$

- The scrambled 9-tuple, $Sd_n[8:0]$ is generated as follows:

$$Sd_n[8] = Sg_n \wedge SB_n[0]$$
$$Sd_n[7:4] = Sx_n[3:0] \wedge TD_n[7:4]$$
$$Sd_n[3:0] = Sy_n[3:0] \wedge TD_n[3:0]$$

8B/9B Symbol Encoding

- 8B/9B encoding provides idle encoding, data encapsulation and any additional control codes we need for PLCA, local/remote fault, etc.
  - 9B encoding has a total of 512 codes available
  - 256 of these are used to encode the scrambler data octet $Sd_n[7:0]$ as 9-bits $Se_n[8:0]$
    - We rely on scrambler to provide control of disparity on a statistical basis
  - Use two sets of 32 codes for Idle to ensure spectral integrity similar to what was done in 802.3dg, to control running disparity and to allow easy detection of correct symbol alignment
    - Complimentary sets of 32 codes with positive and negative disparity, select codebook to balance disparity
    - Use scrambler bits $Sx_n[3:0]$ and $Sy_n[0]$ to select the entry in codebook
    - If a large number of symbols not in the Idle codebook are received, then the symbol alignment is incorrect
  - Use sets of 4 codes for every control code to ensure randomization to avoid spectral spurs
    - Use scrambler bit $Sy_n[1]$ and disparity to select one of 4 possible codes
  - Provide 4 codes for each of SSD, ESD, Error Propagation, Carrier Extend, LPI and 11 other control codes
  - Provide two sets of 16 codes for Sequence – Sequence ordered sets control code
    - Encode Sequence code by using scrambler bits $Sx_n[3:0]$ and disparity to select the entry in the Sequence codebook
    - Encode Sequence ordered sets with a Sequence code followed by 3 scrambled data bytes
    - Use Sequence ordered sets for Local Fault, Remote Fault and Link Interruption

Allocation of 9B Codes

- An allocation of 9B codes to Data, Idle and Control codes is shown here
  - Choose mostly codes with disparity ±1 for data (240 out of 256)
  - Two sets of 32 entries for Idle codebooks for symbol alignment, randomization and running disparity
  - SSD, ESD, and 14 primary control codes and up to 23 spare control codes
  - Two sets of 16 entries for Sequence ordered sets codebooks for randomization and running disparity
  - Randomize Idle, Sequence and control codes based on scrambler bits to ensure spectral integrity

Type	Number	Randomization	Disparity	Transitions	No. of Codes
Data	256	x 1	±1, ±3	2 → 8	256
Idle + / -	2	x 32	±3	3 → 4	64
SSD / ESD	2	x 4	±1	3 → 4	8
Control	14	x 4	±3	3 → 4	56
Sequence + / -	2	x 16	±3	1 → 2	32
Spare Control	23	x 4	±5, ±7	1 → 3	92
Unused	2	x 1	±7	1	2
COMMA / MARK	2	x 1	±9	0	2
					512

Sequence Ordered Sets - Clause 46.3.4

- The Sequence ordered sets to send link fault signaling is defined in clause 46.3.4
  - Status is signaled in a four byte Sequence ordered set as shown in Table 46–5.
  - Local Fault, Remote Fault & Link Interruption are continuous not transitory signals

46.3.4 Link fault signaling

Link fault signaling operates between the remote RS and the local RS. Faults detected between the remote RS and the local RS are received by the local RS as Local Fault. Only an RS originates Remote Fault signals.

Sublayers within the PHY are capable of detecting faults that render a link unreliable for communication. Upon recognition of a fault condition a PHY sublayer indicates Local Fault status on the data path. When this Local Fault status reaches an RS, the RS stops sending MAC data or LPI, and continuously generates a Remote Fault status on the transmit data path (possibly truncating a MAC frame being transmitted). When Remote Fault or Link Interruption status is received by an RS, the RS stops sending MAC data or LPI, and continuously generates Idle control characters. When the RS no longer receives fault status messages, it returns to normal operation, sending MAC data or LPI.

Status is signaled in a four byte Sequence ordered set as shown in Table 46–5. The PHY indicates Local Fault with a Sequence control character in lane 0 and data characters of 0x00 in lanes 1 and 2 plus a data character of 0x01 in lane 3. The RS indicates a Remote Fault with a Sequence control character in lane 0 and data characters of 0x00 in lanes 1 and 2 plus a data character of 0x02 in lane 3. Though most fault detection is on the receive data path of a PHY, in some specific sublayers, faults can be detected on the transmit side of the PHY. This is also indicated by the PHY with a Local Fault status.

For operation with links that may be temporarily interrupted, optional detection of a third fault condition, Link Interruption, is provided. Link Interruption is indicated by the PHY receive function by continuously sending the Link Interruption ordered set as defined in Table 46–5.

The RS reports the fault status of the link. Local Fault indicates a fault detected on the receive data path between the remote RS and the local RS. Remote Fault indicates a fault on the transmit path between the local RS and the remote RS. The RS shall implement the link fault signaling state diagram (see Figure 46–11).

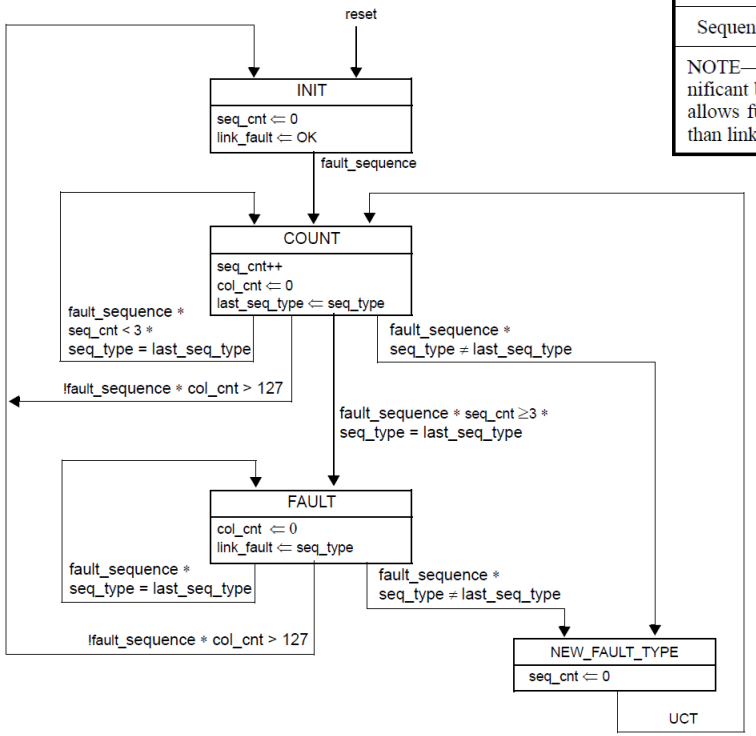


Figure 46–11—Link fault signaling state diagram

Table 46–5—Sequence ordered sets

Lane 0	Lane 1	Lane 2	Lane 3	Description
Sequence	0x00	0x00	0x00	Reserved
Sequence	0x00	0x00	0x01	Local Fault
Sequence	0x00	0x00	0x02	Remote Fault
Sequence	0x00	0x00	0x03	Link Interruption
Sequence	≥ 0x00	≥ 0x00	≥ 0x04	Reserved

NOTE—Values in Lane 1, Lane 2, and Lane 3 columns are in hexadecimal, most significant bit to least significant bit (i.e., <7:0>). The link fault signaling state diagram allows future standardization of reserved Sequence ordered sets for functions other than link fault indications

The RS output onto TXC<3:0> and TXD<31:0> is controlled by the variable link_fault.

a) **link_fault = OK**
The RS shall send MAC frames as requested through the PLS service interface.
In the absence of MAC frames, the RS shall generate Idle control characters.

b) **link_fault = Local Fault**
The RS shall continuously generate Remote Fault Sequence ordered sets.

c) **link_fault = Remote Fault or link_fault = Link Interruption**
The RS shall continuously generate Idle control characters.

8B/9B Encoding for Sequence Ordered Sets

- A Sequence ordered set is a 4-byte sequence of a Sequence control code followed by 3 data bytes
  - In XGMII this is a Sequence control character in lane 0 and 3 data characters in lanes 1, 2 and 3
  - Local Fault, Remote Fault & Link Interruption are continuous not transitory signals
- POPI using 8B/9B encoding will signal a Sequence ordered set as follows:
  - Send Sequence code by using scrambler bits $Sx_n[3:0]$ and disparity to select the entry in the Sequence codebook followed by 3 data bytes scrambled
  - This ensures that continuous Sequence ordered sets are randomized on the POPI link
- Use Sequence ordered sets for future control code and signalling needs
  - Sequence ordered sets can be used to include PLCA Beacon and PLCA Commit primitives
  - In particular, can be used for future control codes that require continuous signalling
  - Almost an infinite number of future codes are available, 2^24
- Use symbol control codes for future latency sensitive control codes
  - Only 3 of the 14 allocated control codes allocated (CE, EP, LPI) so 11 unused plus 23 spare control codes
- Provides a very good alignment between 8B/9B and 64B/66B encoding of control signals used in lower speed and higher speed POPI

Disparity Control / Mitigation

- During data running disparity is not explicitly controlled
  - In 8B/9B encoding there are not sufficient codes to absolutely bound disparity
  - The use of a scrambler provides control of disparity, transitions and run length on a statistical basis
  - The disparity range is minimized by choosing mostly codes with disparity ±1; 240 out of 256 and 16 codes with ±3
- In Idle we have sufficient codes available to provide complimentary positive and negative disparity for each code
  - For Idle we propose 64 codes to ensure spectral integrity during Idle arranged as two complimentary codebooks one with positive and the other with negative disparity
  - For every control code we propose 4 codes so we have randomization to avoid spectral spurs and arrange these as a complimentary positive and negative disparity sets of 2
- A running disparity of the POPI scrambled stream, including the interleaved sideband bits is maintained
  - During Idle when an Idle or a Sequence or a control code is selected, the positive or negative disparity version is chosen to minimize the current running disparity - the receiver accepts either code
- A simulation will be run to quantify the disparity bound using the above scheme

Low Latency Data Requirement

- The 10/100/1000M legacy MAC/PHY 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
- The proposal for higher speed POPI is to interleave the data symbols and the sideband symbols
  - Use a 4:1 ratio of data to sideband for a single or multi-port PHYs
  - For a Quad or Octal PHY there is no impact on latency – however for single or dual there is a penalty
  - See Single 10M or 100M PHY POPI Requirements – March 2026 slides 7, 8, 9 & 10
- This proposal interleaves the sideband 1-bit at a time with the data symbols to ensure that the sideband channel has no impact on latency on the data
  - Using one sideband bit per symbol for a 12.5Mb/s sideband channel with POPI signalling at 125 MHz
  - Can use 2 sideband bits for 25 Mb/s at 137.5MHz signalling or 3 bits for 37.5 Mb/s at 150 MHz

Symbol Alignment and Scrambler Lock at Power-up

- At power-up and reset the MAC and PHY start sending scrambled Idle codes
  - Use scrambler bits $Sx_n[3:0]$ with all bits inverted and $Sy_n[0]$ to select entry in codebook
    - Select codebook entry with bits $\overline{Sx_n[3]}, \overline{Sx_n[2]}, \overline{Sx_n[1]}, \overline{Sx_n[0]}, Sy_n[0]$
    - Transmitting with with all bits inverted indicates to the other end that we are in power-up and in the process of locking the symbol alignment and scrambler
  - Receiver chooses a candidate symbol alignment and selects the received symbol $Se_n[9:0]$
    - Checks if the 9B symbol $Se_n[8:0]$ is an element of the Idle codebook and if a significant number of received symbols are not entries in the Idle codebook we know the symbol alignment is not correct
    - Adjust the symbol alignment by 1 bit and continue until symbol alignment is achieved
  - The scrambler can now be loaded, a bit at a time from the LSB of the received symbol and scrambler lock can be verified after a number 9B symbols are correctly decoded
  - When symbol alignment and scrambler lock have been verified, start transmitting idle codes selected using $Sx_n[3:0]$ and $Sy_n[0]$ without inverting bits
    - Select codebook entry with bits $Sx_n[3], Sx_n[2], Sx_n[1], Sx_n[0], Sy_n[0]$
    - The received 9B Idle symbols can be decoded and checked against scrambler bits $Sx_n[3:0]$ to see if the other side is indicating symbol alignment and scrambler lock
  - When both sides are transmitting Idle codes indicating symbol alignment and scrambler lock – the power-up is complete

8B/9B Encoding for Multi-port PHYs

- For a multi-port PHY the symbols for each of the ports are interleaved
  - So for N ports we interleave the 10B symbol $Se_n[9:0]$ for each port, where each 10B symbol corresponds to a data byte at the MII
  - With 8B/9B encoding the sideband channel is interleaved one bit at a time with each 10B symbol so we have no latency impact on the data path
  - Hence we have N sideband bits for N ports available for the single sideband channel
- The Idle symbols for Port 0 are encoding in a slightly different way to allow easy identification of Port 0 symbols during Idle
  - The Idle symbol use scrambler bits $Sx_n[3:0]$ with bit 2 inverted and $Sy_n[0]$ to select entry in codebook
  - For Port 0 select codebook entry with bits $Sx_n[3], \overline{Sx_n[2]}, Sx_n[1], Sx_n[0], Sy_n[0]$
  - For other ports select codebook entry with bits $Sx_n[3], Sx_n[2], Sx_n[1], Sx_n[0], Sy_n[0]$

Encoding for Link Fault Conditions – Link Status Pin

- Many industry standard MAC/PHY interfaces embed information about the link; including speed, duplex and link status
  - Typically exchanged after Auto-Negotiation or when the link status changes
  - In POPI it is proposed to use a sideband Macro Frame to send this information from the PHY to the MAC using a Link Status Macro Frame
  - At a 100M speed using a sideband Macro Frame to send loc_rx_status information from the PHY to the MAC reconciliation sublayer (RS) takes approximately 10 μs
    - A Macro Frame is 64 bits, and assuming a change in loc_rx_status information is a high priority a change in loc_rx_status can be conveyed in 128 sideband bits times, equivalent to 1024 data bits times (128 bytes)
  - The PHY also requires time to determine a change in loc_rx_status, e.g. a change in signal quality
- In Industrial Ethernet applications it can be important to have a fast indication of a change in link status via a Link Status pin; especially for fail over to a redundant link
  - In many PHY technologies, link fault signalling is defined; typically PHY technologies post XGMII /clause 46
  - When the PHY determines that loc_rx_status is NOT_OK, Local Fault is sent to the RS using Sequence ordered sets
    - A Sequence ordered set is 32 data bits times, so 320 ns to send link status information from the PHY to the RS
  - So link fault signalling can signal a fault condition very quickly and the Macro Frame can follow with more detailed information on the fault condition

Encoding for the Sideband Channel

- The sideband channel is a sequence of Macro Frames and Idle bytes
  - Macro Frames are composed of a sequence of 8 bytes with the first byte being the MacroFrame Type
    - See the most recent presentation from Jason Potterf; POPI Sideband MicroFrame Format Proposal
  - Hence a sideband channel consistes of Idle bytes and 8-byte Macro Frames
  - For low speed POPI the sideband channel is transmitted 1-bit at a time interleaved with the encoded data for each port
  - In higher speed POPI using a SerDes and 64B/66B encoding an 8-byte Macro Frame naturally fits into a 64B/66B data frame and idle bytes are sent as a 64B/66B Idle blocks
- For low speed POPI a means is required to indicate sideband Idle bytes and to indicate the alignment of an 8-byte Macro Frame
- During power-up the sideband channel consists of Idle bytes
  - We propose using alternating Comma and Mark codes of bytes of all zeros and all ones to represent the Idle bytes
    - `0b00000000, 0b11111111, 0b00000000, 0b11111111`
  - In a multi-port PHY these should be byte aligned to start on Port 0 symbols
- After power-up, when Macro Frames can be transmitted, they should start byte aligned with the alternating Comma and Mark codes
  - Hence, we must exclude 0x00 and 0xFF from the possible MacroFrame Type codes
    - Note, 0xFF is currently used as a Macro frame code
  - In practice, will likely have Comma and Mark codes in between Macro Frames; but in theory can have a continuous stream of Macro Frames
  - Synchronized by valid MacroFrame Type codes every 8 bytes, or Comma and Mark codes and also have symbol alignment and scrambler lock for the overall 8B/9B POPI bit stream

Byte Stuffing for Multi-port PHYs with 8B/9B Encoding

- The working assumption and implementation in practice for Industry Standard MAC-PHY interfaces is to exactly match the interface data rate to the Max Speed x Number of Ports
  - Bit stuffing is used for sub-rates
  - E.g. for a multi-port 1G interface with ports at a lower speeds like 10M or 100M the bits are replicated 10 x or 100 x
- For a POPI interface between a MAC and a Multi-port PHY with less than N ports enabled, the interface can run at less than the max speed
  - We can operate at a lower speed if we know that a number of port will be disabled for a long time
  - Sometimes, a port may be disabled but we want to reserved bandwidth to cover the case of the port being enabled later
- Propose using a simple byte / symbol stuffing scheme for excess bandwidth not required by the current *Speed x Number of Active Ports*
  - Both the MAC and single or multi-port PHY know the number of enabled ports, the speed of each port and the number of disabled ports
  - The 10B symbols corresponding to data, Idle or control for each port are interleaved with the symbols of the other port
  - For each disabled port we interleave a 10B symbol of scrambled zeros
    - Note, for an enabled port with the link down Idles are sent – as at any point in time the link could come up
  - For each port operating at a lower speed than the max speed supported by the POPI link we use byte stuffing to match the max speed
    - Hence, for a 100M POPI link a port operating at a 10M speed will receive a new byte for every 10 bytes of a 100M port
    - We use byte stuffing with by 9 x 10B symbols of scrambled zeros to expand the 1 x 10B symbol for the data byte to 10 x 10B symbols
- These 10B symbols stuffed scrambled zeros are available to be used for additional sideband channel traffic
  - Any lower speed port is a stream of encoded bytes of all zero, e.g. 0x00 and are thus sideband Idle bytes
  - A Macro Frame can be transmitted by replacing 8 Idle bytes of the lower speed port with an 8-byte Macro Frame
  - Likewise, a Macro Frame can also be transmitted by replacing 8 Idle bytes of a disabled port with an 8-byte Macro Frame

Example Rates for the Low Speed POPI

- This proposal is for PHY rates of 10M to 200M
  - The upper rate of 200M corresponds to signalling rate of 250 MHz at the pins after scrambling, encoding and interleaving of the sideband channel
    - This is the same signalling rate as a standard 1G RGMII interface on each of the data pins
  - The lower rate of 10M corresponds to a single 10M PHY and a signalling rate of 12.5 MHz
- The following examples of single and multi-port PHYs supported

PHY Rate (Mb/s)	Num Ports	8B/9B	Sideband Bits	POPI Line Rate (MHz)
10	1	8B/9B	1 SB_bit	**12.5**
10	4	8B/9B	1 SB_bit	**50**
10	8	8B/9B	1 SB_bit	**100**
100	1	8B/9B	1 SB_bit	**125**
100	2	8B/9B	1 SB_bit	**250**

Summary

- A proposal has been presented for 8B/9B symbol encoding with interleaved sideband bits for 10M and 100M single and multi-port PHYs
  - This operates at a 12.5 to 250 MHz signalling rate supporting a 12.5 Mb/s sideband channel/ port with very low latency by mapping each byte at the MII to a byte at the PHY PCS
    - Achieves the goal of **ensuring a new data byte is available each cycle for encoding and transfer to the PCS** which is very important at lower speeds
  - Is much more efficient than using 8B/10B encoding and allows an implementation of POPI using single ended IO to use the same signalling rate as existing MAC-PHY interfaces
  - Supports clause 36 ordered sets and clause 49 Sequence ordered sets maintaining alignment with the encoding of control signals for higher speed POPI using 64B/66B encoding
  - Uses scrambling to ensure the spectral integrity of the serial data and for improved EMI
  - Provides a scheme for symbol alignment and scrambler lock at power-up/reset
  - Uses balanced disparity in the Idle and control codes to control running disparity
  - Provides a scheme for alignment of the sideband channel Macro Frames and byte stuffing for ports operating at a lower speed or for disabled ports

Questions ?