PHY Interface for the PCI Express*, SATA, USB 3.2, DisplayPort*, and USB4* Architectures

July 2024

Revision 7.0

Reference Number: 643108

Intellectual Property Disclaimer

THIS SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER INCLUDING ANY WARRANTY OF MERCHANTABILITY, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION, OR SAMPLE.

A COPYRIGHT LICENSE IS HEREBY GRANTED TO REPRODUCE AND DISTRIBUTE THIS SPECIFICATION FOR INTERNAL USE ONLY. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY OTHER INTELLECTUAL PROPERTY RIGHTS IS GRANTED OR INTENDED HEREBY.

INTEL CORPORATION AND THE AUTHORS OF THIS SPECIFICATION DISCLAIM ALL LIABILITY, INCLUDING LIABILITY FOR INFRINGEMENT OF PROPRIETARY RIGHTS, RELATING TO IMPLEMENTATION OF INFORMATION IN THIS DOCUMENT AND THE SPECIFICATION. INTEL CORPORATION AND THE AUTHORS OF THIS SPECIFICATION ALSO DO NOT WARRANT OR REPRESENT THAT SUCH IMPLEMENTATION(S) WILL NOT INFRINGE SUCH RIGHTS.

INTEL CORPORATION MAY MAKE CHANGES TO SPECIFICATIONS, PRODUCT DESCRIPTIONS, AND PLANS AT ANY TIME, WITHOUT NOTICE.

Intel Corporation and its subsidiaries (collectively, "Intel") would like to receive input, comments, suggestions and other feedback (collectively, "Feedback") on this specification. To be considered for incorporation into the specification, Feedback must be submitted by e-mail to: pipespecification@intel.com. To the extent that You provide Intel with Feedback, You grant to Intel a worldwide, non-exclusive, perpetual, irrevocable, royalty-free, fully paid, transferable license, with the right to sublicense, under Your Intellectual Property Rights, to make, use, sell, offer for sale, import, disclose, reproduce, make derivative works, distribute, or otherwise exploit Your Feedback without any accounting. As used in this paragraph, "Intellectual Property Rights" means, all worldwide copyrights, patents, trade secrets, and any other intellectual or industrial property rights, but excluding any trademarks or similar rights. By submitting Feedback, you represent that you are authorized to submit Feedback on behalf on your employer, if any, and that the Feedback is not confidential.

Notice: Implementations developed using the information provided in this specification may infringe the patent rights of various parties including the parties involved in the development of this specification. No license, express or implied, by estoppel or otherwise, to any intellectual property rights (including without limitation rights under any party's patents) are granted herein.

Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.

Intel, the Intel logo, and Thunderbolt are trademarks of Intel Corporation or its subsidiaries.

*Other names and brands may be claimed as the property of others

Contents

Figures

Tables

Revision History

Revision Number	Description	Date
7.0	• Typo fixes. Increased the width of the Error Count field in the Rx Margin Status2 and Rx Margin Status3 (new) registers from six to eight bits for USB4. Restored RxStatus 101b encoding definition that was inadvertently dropped in 6.1.1 release. Removed the entry referencing 31.25 Mhz PCLK rate at 2.5 GT/s link rate from Table 3-1 as there is no defined encoding for that PCLK rate. Clarified relationship between RxInPhase01Equalization and RxEqTraining/RxEqEval. Support for PCIe 7 (prelim, version 0.7 of PCIe 7 base spec): 160-bit RxData and TxData width, new Rate definition, LocalPresetIndex update, LocalG7FS, LocalG7LF.	July 2024
6.2.1	• Added optional RxInPhase01Equalization field to the PHY Rx Control1 register. Added RxInPhase01EqRequirement parameter. Filled in attribute column for RxInPam3Mode to be "Level". Added operational sequence (Section 8.35) for switching direction of a differential Rx/Tx pair.	May 2023
6.2	• Add PHY parameter Link Evaluation Feedback Format Supported. Clarified that the PIPE control signals for Rx functionality apply to both Rx and Rx2 unless separate control signals are defined; similar clarification added for PIPE control signals for Tx functionality. Replaced Loopback Slave/Slave Loopback with Loopback Follower/Follower Loopback. Fixed typo in the DisplayPort PCLK Rate[4:0] encoding of 1 (62.52 Mhz corrected to 67.5 Mhz). Added Section 8.3.5 for DisplayPort Power Management details. Updates for USB4 Version 2 (56-bit interface for PAM3, asymmetric mode). Added following signals: TxDetectRx2, TxElecIdle2, RxEIDetecDisable2, RxStandby2, RxStandbyStatus2, RxElecIdle2. Added RxLaneEnable, TxLaneEnable, RxLaneReady, and TxLaneReady register fields. Added controls for an optional MacCLK lane. Added Section 8.13 for USB4 LFPS detection. TxDeemph[17:0] applies to DisplayPort. Correct width of PowerDown[3:0] from 2-bits to 4-bits in Table 8-6. Updated Section 8.21 to remove references to equalization based on legacy interface. Corrected Figure 8-21 PowerDown state to P1. Corrected Table 8-6 to clarify that LPFS in P1 only applies to USB and that receiver detection only applies to USB.	Jan 2023
6.1.1	• Added optional PAM4RestrictedLevels register field and PAM4RestrictedLevelsRequirement PHY datasheet parameter. SRISEnable clarification. L0p clarification in section 8.26 (applicable to Original PIPE architecture only, e.g. when pairing PCIe 6.0 capable MAC with older PHY that does not support SerDes architecture).	March 2022
6.1	• Clarified that PHY Mode, DP_Mode_TX_RX, and SerDesArch are permitted to be changed only during a Reset# assertion and are asynchronous. Updated the SRISEnable description to indicate that there may be a PHY implementation-specific requirement. Clarified that the loopback usage for TxDetectRx/Loopback is not applicable to SerDes architecture. Fixed cross-reference typos. In Section 8.3.2, added P3 to P2 and P2 to P3 as allowable transitions for USB. Updated the USB 3.1 specification references to USB 3.2. Added footnote to Table 3-1 clarifying that RxDataValid is not applicable to a SerDes architecture. • Clarified that RxEIDetectDisable is asynchronous for USB/USB4. Added new Digital Near End Loopback Feature. Added MaxRxClkFrequency PHY parameter. Added constraints on RxCLK when RxValid is deasserted. Add new RxEIDetectDisableSupportedStates PHY parameter. Clarified that Reset# may be asserted at any time and that the PHY must hold itself in its lowest power state during Reset# assertion. Defined power states for USB4 with suggested LTSSM mappings. Defined protocol that allows the MAC to transmit LFPS in USB4 and DP modes; add new parameters MinTimeBeforeLFPS, MaxTimeBeforeLFPS, and MinTimeEIAfterLFPS). Support for PCI Express revision 6.0 (preliminary ver 0.71): Changed LocalTxPresetCoefficients_Cminus2/LocalTxPresetCoefficients_Cminus1/LocalTxPresetCoefficients_Czero/LocalTxPresetCoefficients_Cplus1 from 8-bits to 6-bits; Changed LocalG6FS/LocalG6LF/FS/LF from 8-bits to 6-bits; removed "Voltage Margining Eye" from Rx Margin Control1 register; changed TxDeemph_Cminus2/TxDeemph_Cminus1/TxDeemph_Czero/TxDeemph_Cplus1 from 8-bits to 6-bits. Added gray code/precode labeling into Figure 2-1. Added RxValid2.	August 2021

Revision Number	Description	Date
6.0	• Updated Converged I/O nomenclature to USB4. Removed USB4 from the RxPolarity's applicable protocols as the RxPolarity is not applicable to SerDes. Added clarification to RxValid definition for SerDes mode, indicating the different values. Update to operational rules in Section 8.26.1 to indicate that LocalFS and LocalLF are updated before a PhyStatus deassertion. Clarified valid TxElecIdle[3:0] transition sequences in Section 8.20. Clarifications for configurable pairs definition, especially around Tx+Tx and Rx+Rx combinations. Updated Figure 3-2 and Figure 3-3 to show the second configurable pair for DisplayPort. Added RxData2 and TxData2 signal definitions for use in Rx+Rx and Tx+Tx diff pair combinations. Added Thunderbolt™ 3 signaling rate options of 10.3125 and 20.625 GT/s. Clarified that the actual Max PCLK rate is advertised in the PHY datasheet. Updated the following signals to apply to USB4: RxTermination, RxEIDetectDisable, Width, RxWidth, RxValid. Updated the following signals to apply to DisplayPort Rx: RxTermination, RxWidth. Width updated to apply to DisplayPort Tx. Support for PCI Express 6.0 (preliminary, version 0.5): Updated and added new register fields for four 8-bit coefficients for TxDeemph and LocalTxPresetCoefficients (assume 11 presets for now), 8-bit FS/LF values, ability to specify one of three eyes for timing margining, SerDes required and original PIPE not allowed for generation 6. Added a rate override option to the rate encoding for a DisplayPort. Figure 8-9 was corrupted in previous versions – fixed this. Added support for deep power management with optional asynchronous handshake signals DeepPMReq# and DeepPMAck# and optional Restore# signal for managing PHY context restoration after power rail restoration. Updated wording to be consistent that TxDataValid must be asserted when operating in modes that do not require. Any changes to PowerDown or RxStandby must cause IORecal, RxEqEval, or RxEqTraining operations to abort. RXTermination is required for PCIe. RefClkRequired# can deassert during P1. Added RxCLK2. Corrected typos in register names in Figure 9-4 and Figure 9-5.	June 2020
5.2.1	• Fixed typo in ShortChannelPowerControlSettingsSupported parameter's PowerControlSetting2 suggested description to reference <=10dB channel (rather than a <=5dB channel). Clarification that Table 3-1 is only a sample list of possible width and PCLK rate combinations (and added 4-symbol wide options at 1 GHz for 8 GT/s and 16 GT/s)	February 2020
5.2	• Added support for short reach for MCP applications (new signal ShortChannelPowerControl[1:0] and a new PHY parameter ShortChannelPowerControlSettings). Added 10, 13.5, and 20 GHz frequencies for DisplayPort 2.0. Added PCIe support for RxEqTraining, with new RxEqTrainDone acknowledgment and new PHY parameter PCIeRxEqTrainRequirement. Added support for PHY or controller-initiated Rx recalibration with PhyIORecalRequest, IORecal, and IORecalDone register fields and PHY parameter PhyRecalRequirement. Added clarification to Section 8.20 to specify for TxElecIdle[3:0] bits map into the table. Added clarification that the DisplayPort AUX signals are implemented on a per connector basis. Clarification that RxValid is synchronous to RxCLK in SerDes mode. Added email contact information for submitting feedback (pipespecification@intel.com).	February 2019
5.1.1	• Removed Rx Status0-3; the contents were moved to Tx Status3-6 in previous review of the specification, but the old registers were inadvertently left in the specification.	July 2018

Revision Number	Description	Date
5.1	• PCIe 5.0 formal rate definitions. General typo corrections and clarifications. Added back in external signals table that was inadvertently dropped in the 5.0 revision. ElasticBufferLocationUpdateFrequency moved to the PHY address space with minimum and maximum values to be specified in the PHY datasheet. Clarified that RefClkRequired# is optional for the PHY. updated TxDataValid description to reference usage in original PIPE architecture for USB due to block encoding. Clarified that PHYs must specify their own timing requirements for RxStandby. Added PHY parameters to specify whether PclkChangeOk or PclkChangeAck handshake is required for rate+width changes and for all rate changes. Clarified states for L1 substates in the PowerDown description and RxEIDetectDisable description. Allowed receiver detection in P2 for PCIe. Add USB clarification for timing around LFPS, RxElecIdle and exit from P1 to P0. Added table of USB PowerDown state characteristics. Updated RxStatus description to reflect that the 111b value indicates corrected SKP for USB. PclkChangeOk and PclkChangeAck handshake is required for all rate changes (not just those impacting PCLK). Add clarification on priority of LFPS transmission versus the SuperSpeed data for USB. RxEIDetectDisable can be used to disable LFPS circuit for power savings. Moved GetLocalPresetCoefficients from bit 5 to bit 7 of the PHY Tx Control5 register to allow growth of the LocalPresetIndex field. Deprecated the TxElecIdle+TxCompliance method of turning off a lane. Updated PHY parameters table for USB 3.2 for Tx EQ. Disallowed LFPS transmission in P2 and P3 for USB. Added eDP rates. Moved Tx Control9 register contents to Rx Control4 register. Moved Rx Status0-3 register contents to Tx Status3-6 registers. Updated the LocalPresetIndex valid range for LocalG5LF register field. Updated various entries in the "Lane Margining at Receiver Sequences" table. Added PHY parameter AsynchReceiverDetectSupport to advertise whether the PHY support asynchronous receiver detection in PCIe P2 state. Updated message bus rules, including restrictions on posted-to-posted writes and defined register groups. Added PHY parameter to advertise the time to transition to a valid electrical idle after sending EIOS. Updated Converged I/O interface to 40-bits. Added the rate/width table for Converged I/O. Add PHY parameters for datapath and control path support options. Disallowed LFPS signaling in P2&P3 for USB. RXTermination assertion during Reset for USB changed to be implementation specific. Added sample clocking topologies compatible with PIPE.	March 2018
5.0	• Clarified that the margin NAK is only required for unsupported voltage margin offset requests that are within the PHY-advertised range. Added support for 64-bit data width for PCIe SerDes only. Mapped all the eligible legacy PIPE signals into message bus registers. Added support for a SerDes architecture. Added requirements to support low-pin count versus legacy PIPE interface and SerDes versus original PIPE architecture. Added support for Converged I/O and DisplayPort. Recommendation that USB nominal empty operation should use RxDataValid. Added EB error recovery mechanism controlled via a register bit. Added the RefClkRequired signal to indicate when the reference clock can be safely removed. Reformatted signal tables into separate input and output tables and added a new column indicating relevant protocols. General cleanup and clarifications.	November 2017
4.4.1	• Removed the "PCLK as an input" requirement for message bus. Added wording to allow PHY to choose whether to support L1 substate management via PowerDown[3:0] exclusively or via RxEIDetectDisable and TxCommonModeDisable.	January 2017
4.4	• Added support for PCIe Rx margining and elastic buffer depth control over a message bus interface. Support for PCIe nominal empty elastic buffer mode. Generation 4 updates: LocalLF/FS, LF/FS, Rate, PCLK rates. SRIS support. RXStandby for USB. L1 substate clarifications. General cleanup.	November 2016
4.3	• Added support for PTM (preliminary for review), L1 Substates (preliminary for review), and PCI Express 4.0 (preliminary revision 0.3).	January 2014
4.2	• Added support for USB 3.1 – preliminary review release based on USB 3.1 specification revision 0.9.	July 2013
4.1	• Initial draft with per-lane clocking option	May 2012

Revision Number	Description	Date
4.1	• Updated from Draft 2 feedback based on several reviewers.	December 2011
4.1	• Draft 2. Updates for initial review feedback and addition of several example timing diagrams for PCI Express 3.0-related signals.	December 2011
4.0	• Draft 1 update adding SATA.	September 2011
4.0	• Draft 6 update adding updates based on the PCI Express 3.0 revision 0.9 feedback.	April 2011
4.0	• Draft 3 update adding PCI Express 3.0 revision 0.9.	April 2011
3.0	• Final update	March 2009
2.9	• Added 32-bit data interface support for USB SuperSpeed mode, support for USB SuperSpeed mode receiver equalization training, and support for USB SuperSpeed mode compliance patterns that are not 8b or 10b encoded. • Solid enough for implementation architectures to be finalized.	August 2008
2.75	• Additional updates for SKP handling.	February 2008
2.71	• Updates for SKP handling and USB SuperSpeed PHY power management.	January 2008
2.7	• Initial draft of updates to support the USB specification, revision 3.0.	December 2007
2.0	• Minor updates, stable revision for implementation.	July 2007
1.9	• Minor updates, mostly editorial.	March 2007
1.87	• Removed references to the Compliance Rate determination. Added sections for Tx Margining and Selectable De-emphasis. Fixed up areas (6.4) based on feedback.	September 2006
1.86	• Fixed up more areas based on feedback. Added a section on how to handle CLKREQ#.	February 2006
1.81	• Fixed up areas based on feedback.	December 2005
1.7	• First pass at generation 2 PIPE	November 2005
1.0	• Stable revision for implementation.	June 2003
0.95	• Updates to reflect the 1.0a base Specification. Added multilane suggestions.	April 2003
0.9	• Minor updates. Solid enough for implementations to be finalized.	December 2002
0.8	• More operational detail. Receiver detection sequence changed.	November 2002
0.7	• Includes timing diagrams	November 2002
0.6	• Provides operational detail	October 2002
0.5	• Draft for industry review	August 2002
0.1	• Initial Draft	July 2002

1 Preface

1.1 Scope of this Revision

The PCI Express* (PCIe*), SATA, USB, DisplayPort*, and USB4* PHY Interface Specification has definitions of all functional blocks and signals. This revision includes support for PCIe implementations conforming to the PCIe base specification, revision 7.0, SATA implementations conforming to the SATA specification, revision 3.0, USB implementations conforming to the USB specification, revision 3.2, DisplayPort implementations conforming to the DisplayPort 2.0 specification, and USB4 implementations conforming to the *USB4® Specification v2.0*.

2 Introduction

The **P**HY **I**nterface for the **P**CI **E**xpress (PCIe), SATA, USB[1], DisplayPort, and USB4 Architectures (**PIPE**) is intended to enable the development of functionalities equivalent to the PHYs of PCIe, SATA, USB, DisplayPort, and USB4. Such PHYs can be delivered as discrete Integrated Circuits (ICs) or as macrocells for inclusion in Application-Specific Integrated Circuit (ASIC) designs. The specification defines a set of PHY functions that must be incorporated in a PIPE-compliant PHY; it also defines a standard interface between a PHY and a Media Access Layer (MAC) and a Link Layer ASIC. This specification is not intended to define the internal architecture or design of a compliant PHY chip or macrocell. The PIPE specification is defined to allow several approaches to be used. When possible, the PIPE specification references the PCIe base specification, the SATA 3.0 specification, the USB 3.2 specification, the DisplayPort 1.4 specification, or the USB4 1.0 specification rather than repeating its content. In case of conflicts, the PCIe base specification, the SATA 3.0 specification, the USB 3.2 specification, DisplayPort 1.3 specification, and USB4 1.0 specification must supersede the PIPE specification.

This specification provides some information about how the MAC could use the PIPE interface for several Link Training and Status State Machine (LTSSM) states, link states, and other protocols. This information should be viewed as "guidelines for" or as "one way to implement" base specification requirements. MAC implementations are free to do things in other ways as long as they meet the corresponding specification requirements.

One of the intents of the PIPE specification is to accelerate the development of PCIe, SATA, USB, and USB4 devices. This document defines an interface to which ASIC and endpoint device can be developed by vendors. Peripheral and IP vendors will be able to develop and validate their designs, insulated from the high-speed and analog circuitry issues associated with the PCIe, SATA, USB, DisplayPort, or USB4 PHY interfaces, therefore minimizing the time and risk of their development cycles.

The PIPE specification defines two clocking options for the interface. In the first alternative the PHY provides a clock (PCLK) that clocks the PIPE interface as an output. In the second alternative, the PCLK is provided to each lane of the PHY as an input. The alternative, where the PCLK is provided to each lane of the PHY, was added in revision 4.1 of the PIPE specification. It allows the controller or logic external to the PHY to more easily adjust timing of the PIPE interface to meet timing requirements for silicon implementations. A PHY is only required to support one of the timing alternatives. The two clocking options must be referenced as "PCLK as PHY Output" and "PCLK as PHY Input" respectively. The DisplayPort only supports the "PCLK as PHY Input" clocking option.

Note: **The "PCLK as PHY Output" mode is not supported for PCIe 5.0 and beyond, USB4, or Displayport.**

Figure 2-1 shows the partitioning described in this specification for the PCIe base specification. Figure 2-2 shows the partitioning described in this specification for the USB 3.2 specification. Figure 2-3 shows the partitioning described in this specification for the USB4 1.0 specification.

1. USB refers to USB3. USB4 is referenced explicitly.

Figure 2-1. Partitioning PHY Layer for PCIe

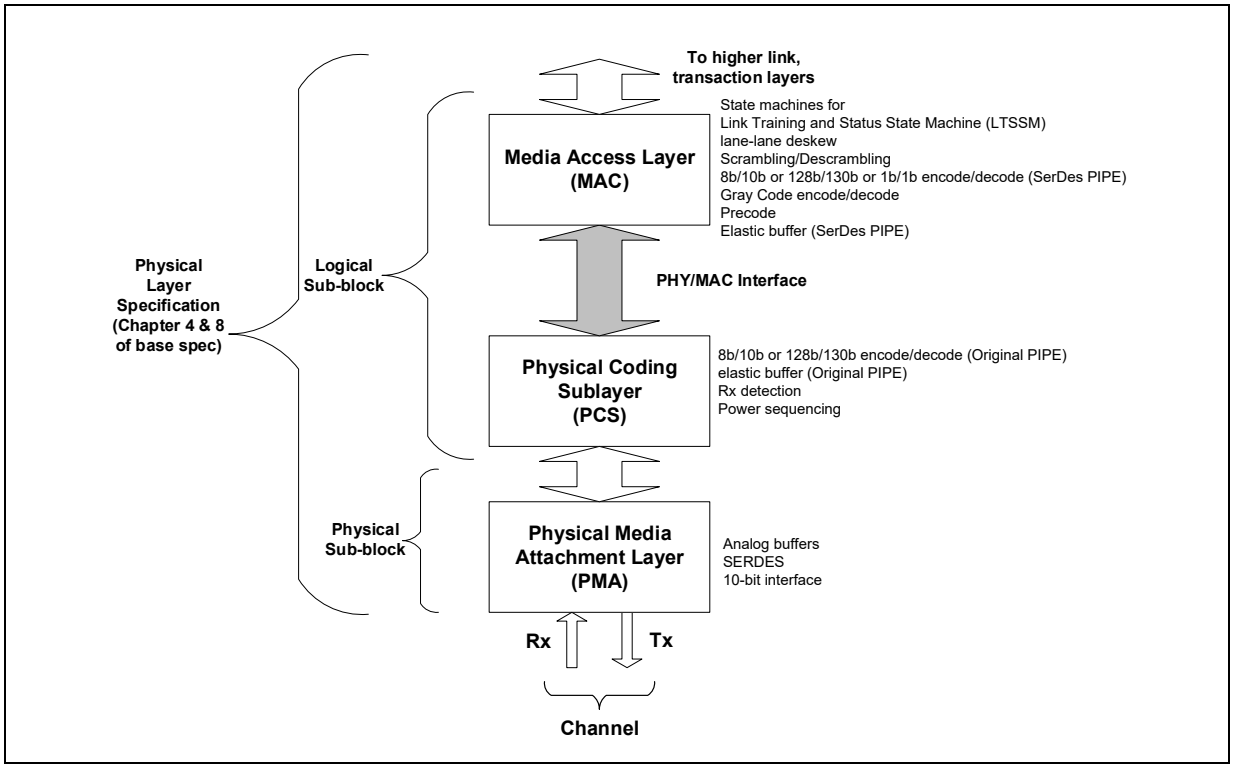


Figure 2-2. Partitioning PHY Layer for USB

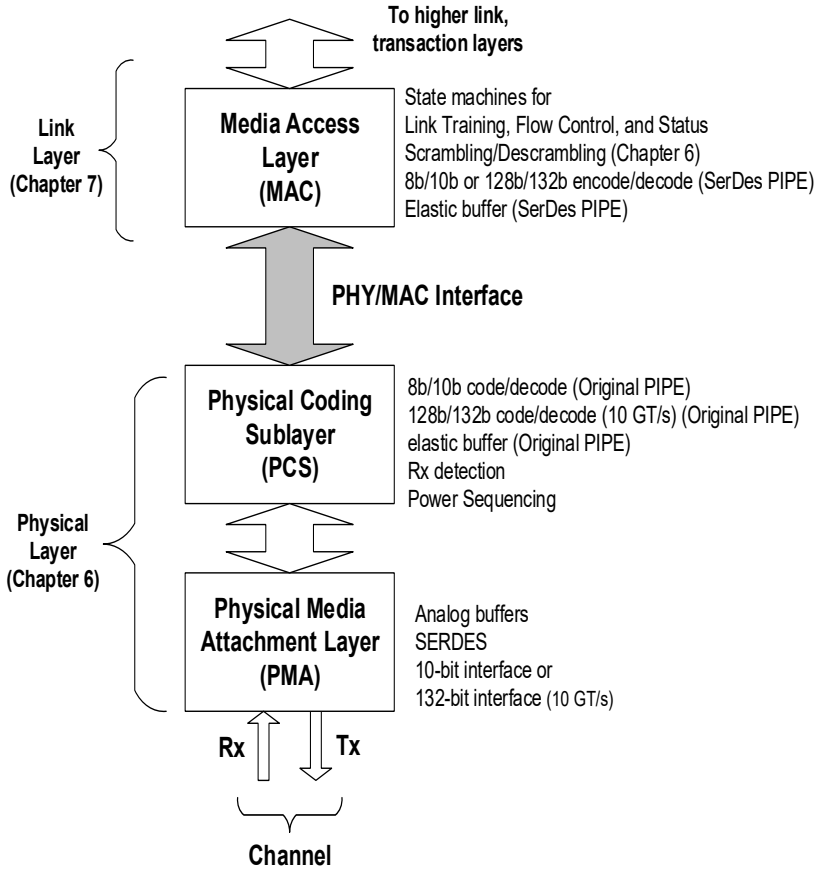
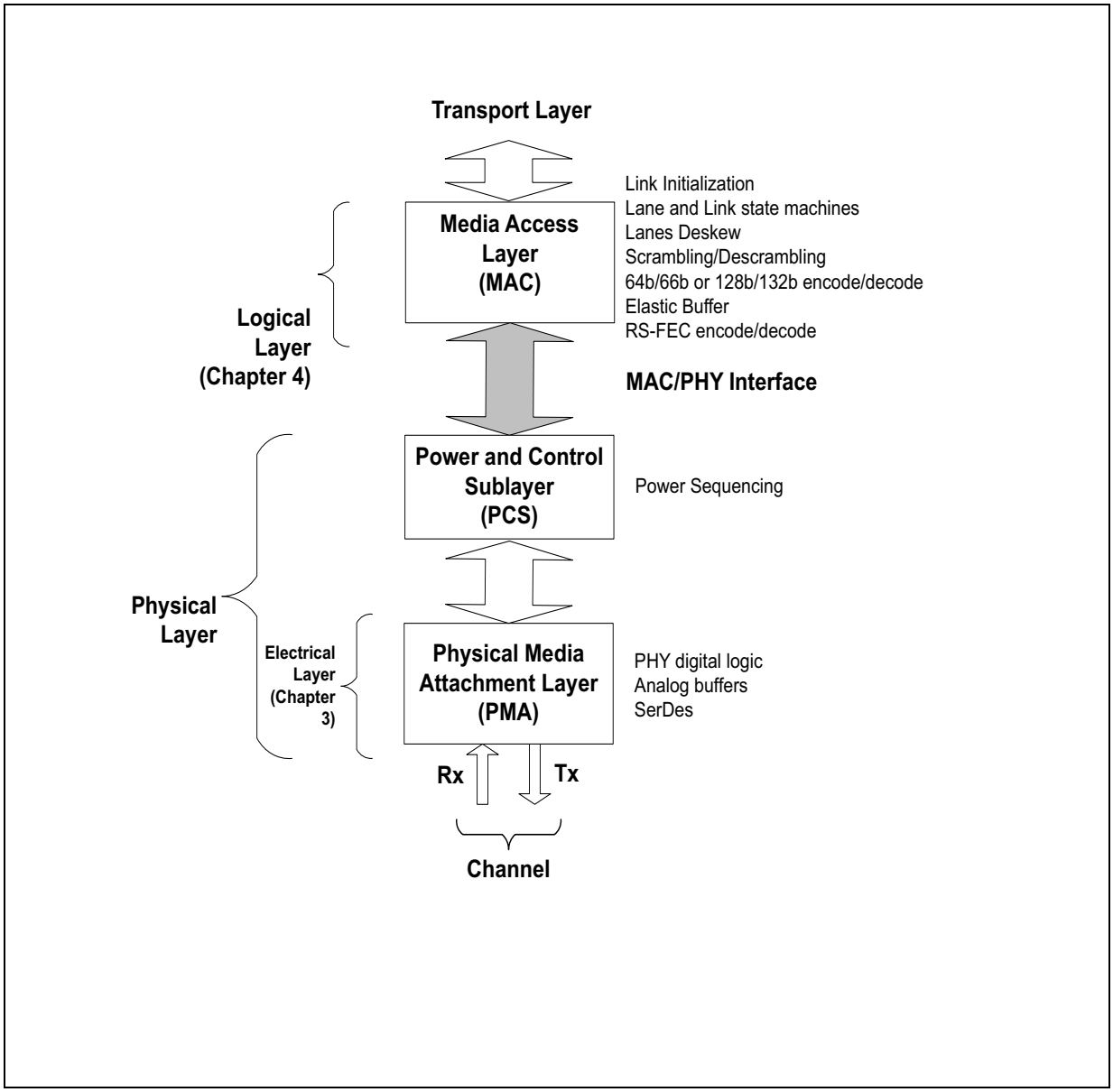


Figure 2-3. Partitioning PHY Layer for USB4



2.1 PCIe PHY Layer

The PCIe PHY layer handles the low level PCIe protocol and signaling. This includes features such as analog buffers, receiver detection, data serialization and de-serialization, 8b/10b encoding and decoding (original PIPE), 128b/130b encoding/decoding (8 GT/s, 16 GT/s, and 32 GT/s) (original PIPE), and elastic buffers (original PIPE). The primary focus of this block is to shift the clock domain of the data from the PCIe rate to one that is compatible with the general logic in the ASIC.

Some key features of the PCIe PHY are:

- The standard PHY interface enables multiple IP sources for the PCIe logical layer and provides a target interface for PCIe PHY vendors.
- Support for 2.5 GT/s only, or 2.5 GT/s and 5.0 GT/s, or 2.5 GT/s, 5.0 GT/s and 8.0 GT/s, or 2.5 GT/s, 5.0 GT/s, 8.0 GT/s, and 16 GT/s, or 2.5 GT/s, 5.0 GT/s, 8.0 GT/s and 16 GT/s and 32 GT/s, or 2.5 GT/s, 5.0 GT/s, 8.0 GT/s, 16 GT/s, 32 GT/s, 64 GT/s,and 128 GT/s serial data transmission rate.
- Utilizes of 8-bit, 16-bit, or 32-bit parallel interfaces to transmit and receive PCIe data. Additionally, it supports a 64-bit interface (in SerDes architecture only).
- Allowed integration of high-speed components into a single functional block as seen by the endpoint device designer.
- Data and clock recovery from serial stream on the PCIe bus.
- Holding registers to stage transmit (Tx) and receive (Rx) data.
- Support of direct disparity control for use in transmitting compliance patterns.
- 8b/10b encoding and decoding, and error indication (original PIPE)
- 128b/130b encoding and decoding, and error indication (original PIPE)
- Receiver detection
- Beacon transmission and reception
- Selectable Tx margining, Tx de-emphasis and signal swing values
- Lane margining at the receiver
- Polarity (original PIPE)
- Electrical Idle entry and exit detection (Squelch)

2.2 USB PHY Layer

The USB PHY layer handles the low-level USB protocol and signaling. This includes features such as analog buffers, receiver detection, data serialization and de-serialization, 8b/10b encoding and decoding, 128b/132b encoding and decoding (10 GT/s), and elastic buffers. The primary focus of this block is to shift the clock domain of the data from the USB rate to one that is compatible with the general logic in the ASIC.

Some key features of the USB PHY are:

- Standard PHY interface that enables multiple IP sources to the USB link layer and provides a target interface for USB PHY vendors.
- Support for 5.0 GT/s and/or 10 GT/s serial data transmission rate.
- Utilizes 8-bit, 16-bit or 32-bit parallel interfaces to transmit and receive USB data.
- Allowing the integration of high-speed components into a single functional block as seen in the device designer.
- Data and clock recovery from serial stream on the USB bus.
- Register holding to stage the Tx and Rx data.
- 8b/10b encoding and decoding, and error indication.
- 128b/132b encoding and decoding and error indication.
- Receiver detection.
- Low Frequency Periodic Signaling (LFPS)

2.3 USB4 PHY Layer

The USB4 PHY Layer handles the low level USB4 protocol and signaling. This includes features such as data serialization and de-serialization, analog buffers, and receiver detection.

Some key features of the USB4 PHY:

- A standard PHY interface enables multiple IP sources for USB4 Link Layer and provides a target interface for USB4 PHY vendors.
- Supports 10 GT/s and/or 20 GT/s serial data transmission rate in combination with a 40-bit parallel interface to transmit and receive USB4 data using PAM2 signaling
- Supports 40 GT/s serial data transmission rate in combination with a 56-bit parallel interface to transmit and receive USB4 data using PAM3 signaling
- Data and clock recovery from serial stream on the USB4 bus
- Holding registers to stage transmit and receive data
- Low Frequency Periodic Signaling (LFPS)

2.4 SATA PHY Layer

The SATA PHY layer handles the low-level SATA protocol and signaling. This includes features such as analog buffers, data serialization and deserialization, 8b/10b encoding and decoding, and elastic buffers. The primary focus of this block is to shift the clock domain of the data from the SATA rate to one that is compatible with the general logic in the ASIC.

Some key features of the SATA PHY are:

- A standard PHY interface that enables multiple IP sources for SATA controllers and provides a target interface for SATA PHY vendors.
- Support of 1.5 GT/s only, or 1.5 GT/s and 3.0 GT/s, or 1.5 GT/s, or 3.0 GT/s, and 6.0 GT/s serial data transmission rate.
- Utilizes 8-bit, 16-bit, or 32-bit parallel interface to transmit and receive SATA data.
- Allows integration of high-speed components into a single functional block as seen in the device designer.
- Data and clock recovery from serial stream on the SATA bus.
- Holding registers to stage transmit and receive data.
- 8b/10b encode/decode and error indication.
- COMINIT and COMRESET transmission and reception.

2.5 DisplayPort PHY Layer

The DisplayPort PHY layer handles the low-level DisplayPort protocol and signaling. This includes features such as data serialization and de-serialization, and analog buffers.

Some key features of the DisplayPort PHY include the following:

- A standard PHY interface that enables multiple IP sources for the DisplayPort link layer and provides a target interface for the DisplayPort PHY vendors.

- Support of 1.62 Gbps, 2.16 Gbps (eDP), 2.43 Gbps (eDP), 2.7 Gbps, 3.24 Gbps (eDP), 4.32 Gbps (eDP), 5.4 Gbps, 8.1 Gbps, 10 Gbps, 13.5 Gbps, and 20 Gbps serial data transmission rates.
- Utilizes 10-bit, 20-bit, or 40-bit parallel interfaces to transmit and receive the DisplayPort data.
- Data and clock recovery from the serial stream on the DisplayPort bus.
- Holding registers to stage Tx and Rx data.

2.6 Low Pin Count Interface and SerDes Architecture

To address the issue of increasing signal count, the message bus interface was introduced in PIPE 4.4 and utilized for PCIe lane margining at the receiver and elastic buffer depth control. In PIPE 5.0, all the legacy PIPE signals without critical timing requirements were mapped into message bus registers so that their associated functionality could be accessed via the message bus interface instead of implementing dedicated signals. Any new features added in PIPE 4.4 and onwards are available only via message bus accesses unless they have critical timing requirements that need dedicated signals.

To facilitate the design of general-purpose PHYs delivered as hard IPs, and to provide the MAC with more freedom to do latency optimizations, a SerDes architecture was defined in PIPE 5.0. This architecture simplifies the PHY and shifts much of the protocol-specific logic into the MAC.

To maximize interoperability between MAC and PHY IPs, PHY designs must adhere to the requirements stated in Table 2-1 to support the Legacy pin interface versus the Low Pin Count interface, and to support original PIPE architecture versus the SerDes architecture. For PCIe, the PCIe 6.0 in Table 2-1 heading refers to a PHY that is configured as PCIe6.0 capable; the selection is done statically based on the capability and does not change with a rate change.

The legacy pin interface refers to a pin interface that utilizes all the applicable dedicated signals, as well as the message bus interface for features not supported through dedicated signals. The low pin count interface refers to a pin interface that utilizes the message bus interface for all features supported through the message bus, using dedicated signals only for features not supported through the message bus. The legacy pin interface dedicated signals are defined in PIPE 4.4.1 and earlier and have been deprecated in PIPE 5.0.

The original PIPE architecture is represented in Figure 4-4, Figure 4-5, Figure 4-6, and Figure 4-7. The SerDes architecture is represented in Figure 4-8 and Figure 4-9.

The legacy pin interface and the low pin count interface are not simultaneously operational, except for the PCIe4.0 lane margining at the receiver being controlled via the low pin count interface, while other operations are managed over the legacy interface. A PHY must be statically configured to utilize either the low pin count interface or the legacy pin interface, for instance, no dynamic switching between the interfaces based on operational rate is permitted. Finally, a SerDes architecture datapath must always utilize the low pin count interface, using the legacy pin interface with SerDes architecture is considered illegal.

Table 2-1. Phy Requirements for Legacy Pin Interface versus the Low Pin Count Interface, and Original PIPE versus SerDes Architecture Support

	USB4 and DisplayPort	USB 3.2 and Lower	PCIe 5.0	PCIe 6.0 and Higher	PCIe 4.0 and Lower	SATA
Legacy Pin Interface	Not allowed	Required (see version 4.4.1)	Not Allowed	Not Allowed	Required (see version 4.4.1)	Required (see version 4.4.1)
Low pin count Interface	Required	Optional	Required	Required	Required for Gen4 Rx margining only, optional for everything else	Optional
Original PIPE Architecture	Not allowed	Required	Recommended[1]	Not Allowed	Required	Required
SerDes Architecture	Required	Optional	Required	Required	Optional	Optional

Note: [1] To provide interoperability with PCIe and USB MACs that choose not to migrate to the SerDes architecture, PHYs are encouraged to provide support for original PIPE via a method where the associated logic can be easily optimized out. With this, designs that do not require a PHY which supports original PIPE are not burdened with any unneeded logic.

2.7 Support for Short Reach (SR) Applications

The PIPE specification supports short channel (also known as Short Reach [SR]) applications, for instance, for multi-chip package solutions. For such applications, the operating power can be reduced significantly by optimizing certain operational and environmental parameters for short channels. For example, the operating power for PCIe can potentially be reduced by up to roughly 50% compared to traditional PCIe applications. While specifying environmental parameters is outside the scope of the PIPE specification, PHY vendors are encouraged to advertise any such environmental knobs that can be changed in short channel applications to reduce power, for instance, reducing the PHY supply voltage. This specification provides hooks for tuning specific operational parameters for reduced power. These operational knobs are PHY vendor-dependent and may include channel loss, receiver equalization activity (including Decision Feedback Equalization [DFE] and Continuous Time Linear Equalization [CTLE]), Tx swing, and clock recovery strategy. PHY vendors that want to support power optimized, short reach applications should identify a useful set of operating points for these knobs that it advertises in its datasheet (via the ShortChannelPowerControlSettingsSupported parameter) that the customer can then select from using the PIPE control interface (via the ShortChannelPowerControl[1:0] signals).

In addition to the just mentioned potential power savings, Multi Chip Package (MCP) applications provide the opportunity for cost savings and additional operational optimization; specifically, it is strongly recommended that DC coupling is used, therefore saving on capacitor insertion cost. As part of the DC coupling support, the controller should bypass explicit receiver detection. For PCIe, the receiver detection operation in the PCIe LTSSM state Detect.Quiet should be bypassed and the LTSSM should automatically proceed to Polling. If the LTSSM transitions back to Detect from Polling due to timeout, it is recommended that a subsequent transition to Polling should occur either upon an electrical idle exit detection or after a 30 to 100 ms timeout. Further optimization based on DC coupling can be implemented to reduce power state (for instance, L1.2) exit latencies.

2.8 Configurable Pairs

Support for configurable Rx and Tx differential pairs was added in version 5.0 of the PIPE specification. This configurability was primarily added to support Type-C alternate mode protocols such as DisplayPort and USB4; however, other applications are possible. Up to two differential pairs are assumed to be operational at any given time. Supported lane combinations are one Rx pair and one Tx pair, two Tx pairs, or two Rx pairs; each combination shares a single set of PIPE per-lane signals. See Section 7 for more information.

3 PHY/MAC Interface

Figure 3-1 shows the data and logical command and status signals between the PHY and the MAC layer for one Rx pair and one Tx pair combination. Figure 3-2 and Figure 3-3 show the data and command and status signals between the PHY and the MAC layer for the DisplayPort DPTX and DPRX, respectively. Full support of PCIe mode, USB mode, SATA mode, DisplayPort mode, and USB4 mode at all rates require different numbers of control and status signals to be implemented. See Section 6.1 for details on which specific signals are required for each operating mode.

Figure 3-1. PHY/MAC Interface

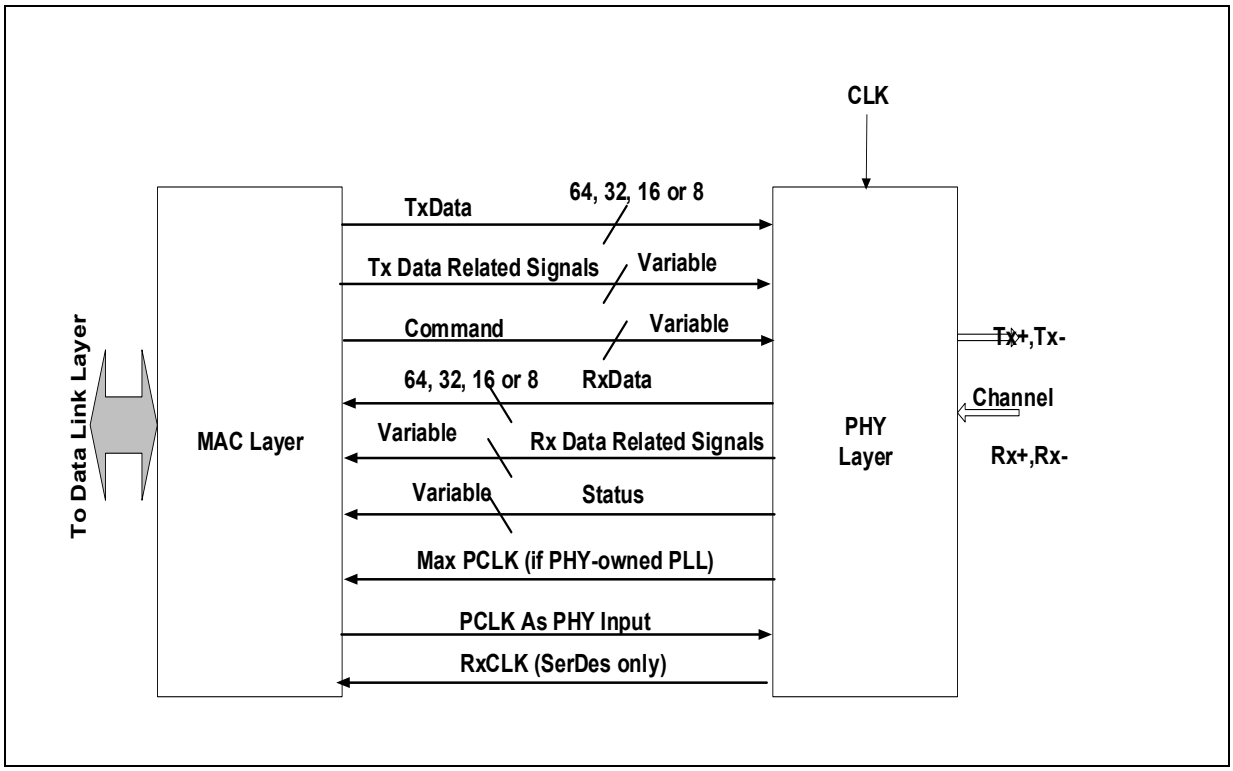


Figure 3-2. DPTX PHY/MAC Interface

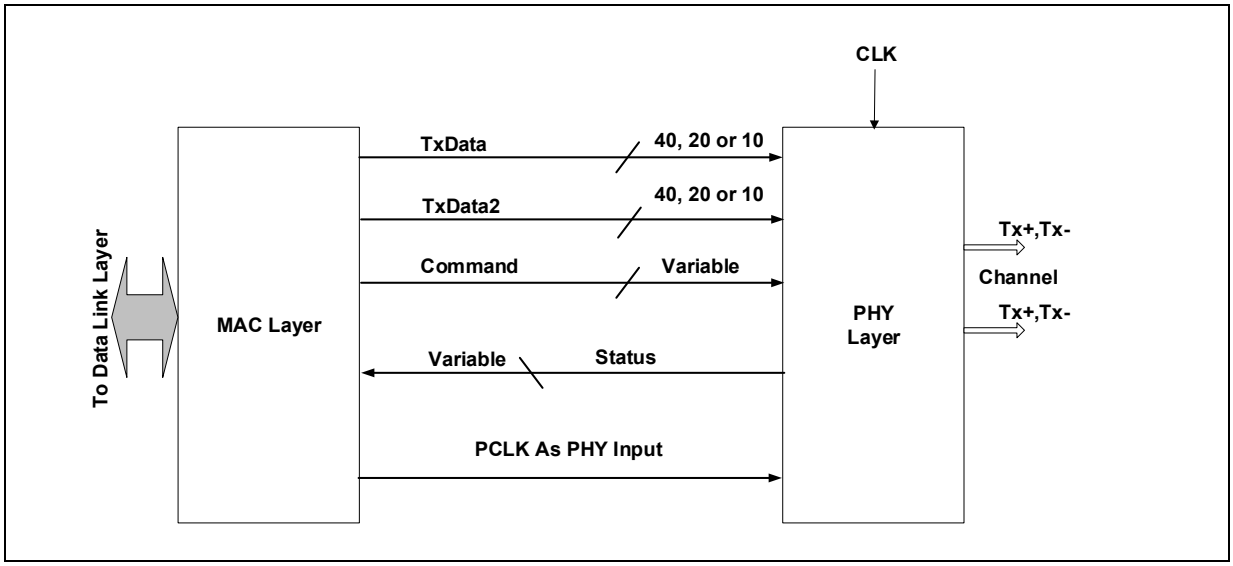
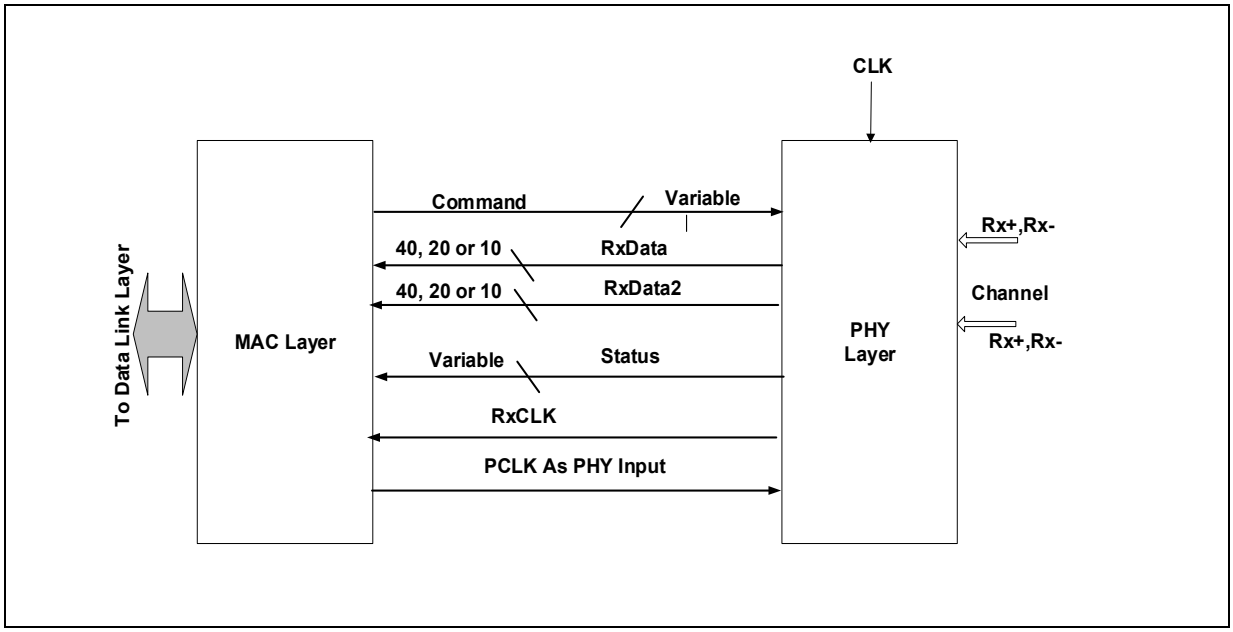


Figure 3-3. DPRX PHY/MAC Interface



This specification allows several different PHY/MAC interface configurations to support several signaling rates.

For PIPE implementations that support only the 2.5 GT/s signaling rate in PCIe mode, implementers can choose to have 16-bit data paths with PCLK running at 125 MHz, or 8-bit data paths with PCLK running at 250 MHz. PIPE implementations that support 5.0 GT/s signaling and 2.5 GT/s signaling in PCIe mode, and therefore can switch between 2.5 GT/s and 5.0 GT/s signaling rates, can be implemented in several ways. An implementation may choose to have PCLK fixed at 250 MHz and use 8-bit data paths

when operating at 2.5 GT/s signaling rate, and 16-bit data paths when operating at 5.0 GT/s signaling rate. Another implementation choice is to use a fixed data path width and change the PCLK frequency to adjust the signaling rate. In this case, an implementation with 8-bit data paths would provide PCLK at 250 MHz for 2.5 GT/s signaling and provide PCLK at 500 MHz for 5.0 GT/s signaling. Similarly, an implementation with 16-bit data paths would provide PCLK at 125 MHz for 2.5 GT/s signaling and 250 MHz for 5.0 GT/s signaling. The sample list of possibilities is shown in Table 3-1.

For PIPE implementations that support 5.0 GT/s USB mode and 10 GT/s, USB mode implementers can choose from the options shown in Table 3-3. A PIPE compliant MAC or PHY is only required to support one option for each USB transfer speed that it supports.

For SATA PIPE implementations that support only the 1.5 GT/s signaling rate implementers can choose to have 16-bit data paths with PCLK running at 75 MHz, or 8-bit data paths with PCLK running at 150, 300 or 600 MHz. The 300 and 600 Mhz options require the use of TXDataValid and RXDataValid signals to toggle the use of data on the data bus.

SATA PIPE implementations that support 1.5 GT/s signaling and 3.0 GT/s signaling in SATA mode, and therefore are able to switch between 1.5 GT/s and 3.0 GT/s signaling rates, can be implemented in several ways. An implementation may choose to have PCLK fixed at 150 MHz and use 8-bit data paths when operating at 1.5 GT/s signaling rate, and 16-bit data paths when operating at 3.0 GT/s signaling rate. Another implementation choice is to use a fixed data path width and change PCLK frequency to adjust the signaling rate. In this case, an implementation with 8-bit data paths could provide PCLK at 150 MHz for 1.5 GT/s signaling and provide PCLK at 300 MHz for 3.0 GT/s signaling. Similarly, an implementation with 16-bit data paths would provide PCLK at 75 MHz for 1.5 GT/s signaling and 150 MHz for 3.0 mode are shown GT/s signaling. A sample list of possible widths and PCLK rates for SATA is shown in Table 3-4. A PIPE compliant MAC or PHY is only required to support one option for each SATA transfer speed that it supports.

A sample list of possible data width and PCLK rate combinations for PCIe mode is shown in Table 3-1; other combinations are possible as long as they conform to the PIPE definitions and the combination of PCLK rate, data width, and TXDataValid/ RXDataValid strobes match the bandwidth across the serial link. A PIPE compliant MAC or PHY is only required to support one option for each PCIe transfer speed that it supports.

Note: PHYs that support greater than x4 link widths must provide an option for 32-bit or less data width.

Table 3-1. PCIe Mode - Possible PCLK Rates and Data Widths (Sheet 1 of 3)

Mode	PCLK	Original PIPE Data Width (SerDes Data Width[1])	TXDataValid and RXDataValid[2] Strobe Rate
2.5 GT/s	4000 Mhz	8 bits (10 bits)	1 in 16 PCLKs
2.5 GT/s	2000 Mhz	8 bits (10 bits)	1 in 8 PCLKs
2.5 GT/s	1000 Mhz	8 bits (10 bits)	1 in 4 PCLKs
2.5 GT/s	500 Mhz	8 bits (10 bits)	1 in 2 PCLKs
2.5 GT/s	250 Mhz	8 bits (10 bits)	N/A
2.5 GT/s	250 Mhz	16 bits (20 bits)	1 in 2 PCLKs

Table 3-1. PCIe Mode - Possible PCLK Rates and Data Widths (Sheet 2 of 3)

Mode	PCLK	Original PIPE Data Width (SerDes Data Width[1])	TXDataValid and RXDataValid[2] Strobe Rate
2.5 GT/s	500 Mhz	16 bits (20 bits)	1 in 4 PCLKs
2.5 GT/s	125 Mhz	16 bits (20 bits)	N/A
2.5 GT/s	250 Mhz	32 bits (40 bits)	1 in 4 PCLKs
2.5 GT/s	62.5 Mhz	32 bits (40 bits)	N/A
2.5 GT/s	62.5 Mhz	N/A (80 bits)	1 in 2 PCLKs
5.0 GT/s	4000 Mhz	8 bits (10 bits)	1 in 8 PCLKs
5.0 GT/s	2000 Mhz	8 bits (10 bits)	1 in 4 PCLKs
5.0 GT/s	1000 Mhz	8 bits (10 bits)	1 in 2 PCLKs
5.0 GT/s	500 Mhz	8 bits (10 bits)	N/A
5.0 GT/s	500 Mhz	16 bits (20 bits)	1 in 2 PCLKs
5.0 GT/s	250 Mhz	16 bits (20 bits)	N/A
5.0 GT/s	250 Mhz	32 bits (40 bits)	1 in 2 PCLKs
5.0 GT/s	125 Mhz	32 bits (40 bits)	N/A
5.0 GT/s	125 Mhz	N/A (80 bits)	1 in 2 PCLKs
5.0 GT/s	62.5 Mhz	N/A (80 bits)	N/A
8.0 GT/s	4000 Mhz	8 bits (10 bits)	1 in 4 PCLKs
8.0 GT/s	2000 Mhz	8 bits (10 bits)	1 in 2 PCLKs
8.0 GT/s	1000 Mhz	8 bits (10 bits)	N/A
8.0 GT/s	1000 Mhz	16 bits (20 bits)	1 in 2 PCLKs
8.0 GT/s	1000 Mhz	32 bits (40 bits)	1 in 4 PCLKs
8.0 GT/s	500 Mhz	16 bits (20 bits)	N/A
8.0 GT/s	500 Mhz	32 bits (40 bits)	1 in 2 PCLKs
8.0 GT/s	250 Mhz	32 bits (40 bits)	N/A
8.0 GT/s	250 Mhz	N/A (80 bits)	1 in 2 PCLKs
8.0 GT/s	125 Mhz	N/A (80 bits)	N/A
16.0 GT/s	4000	8 bits (10 bits)	1 in 2 PCLKs
16.0 GT/s	2000 Mhz	8 bits (10 bits)	N/A
16.0 GT/s	1000 Mhz	16 bits (20 bits)	N/A
16.0 GT/s	1000 Mhz	32 bits (40 bits)	1 in 2 PCLKs
16.0 GT/s	500 Mhz	32 bits (40 bits)	N/A
16.0 GT/s	250 Mhz	N/A (80 bits)	N/A
32 GT/s	4000 Mhz	8 bits (10 bits)	N/A
32 GT/s	2000 Mhz	16 bits (20 bits)	N/A
32 GT/s	1000 Mhz	32 bits (40 bits)	N/A
32 GT/s	500 Mhz	N/A (80 bits)	N/A
64 GT/s	4000 Mhz	N/A (20 bits)	N/A
64 GT/s	2000 Mhz	N/A (40 bits)	N/A
64 GT/s	1000 Mhz	N/A (80 bits)	N/A

Table 3-1. PCIe Mode - Possible PCLK Rates and Data Widths (Sheet 3 of 3)

Mode	PCLK	Original PIPE Data Width (SerDes Data Width[1])	TXDataValid and RXDataValid[2] Strobe Rate
128 GT/s	4000 Mhz	N/A (40 bits)	N/A
128 GT/s	2000 Mhz	N/A (80 bits)	N/A
128 GT/s	1000 Mhz	N/A (160 bits)	N/A

1. For block encoded modes, not all 10, 20, 40, or 80 bits are used. See TXData and RXData signal descriptions for details.
2. RxDataValid is not applicable to SerDes mode

Table 3-2. PCIe Mode (SerDes Only) - Possible RXCLK Rates and Data Widths

Mode	RXCLK	Data Width
2.5 GT/s	250 Mhz	10 bits
2.5 GT/s	125 Mhz	20 bits
2.5 GT/s	62.5 Mhz	40 bits
2.5 GT/s	31.25 Mhz	80 bits
5.0 GT/s	500 Mhz	10 bits
5.0 GT/s	250 Mhz	20 bits
5.0 GT/s	125 Mhz	40 bits
5.0 GT/s	62.5 Mhz	80 bits
8.0 GT/s	1000 Mhz	10 bits
8.0 GT/s	500 Mhz	20 bits
8.0 GT/s	250 Mhz	40 bits
8.0 GT/s	125 Mhz	80 bits
16.0 GT/s	2000 Mhz	10 bits
16.0 GT/s	1000 Mhz	20 bits
16.0 GT/s	500 Mhz	40 bits
16.0 GT/s	250 Mhz	80 bits
32 GT/s	4000 Mhz	10 bits
32 GT/s	2000 Mhz	20 bits
32 GT/s	1000 Mhz	40 bits
32 GT/s	500 Mhz	80 bits
64 GT/s	4000 Mhz	20 bits
64 GT/s	2000 Mhz	40 bits
64 GT/s	1000 Mhz	80 bits
128 GT/s	4000 Mhz	40 bits
128 GT/s	2000 Mhz	80 bits
128 GT/s	1000 Mhz	160 bits

Table 3-3. USB Mode – Possible PCLK or RXClk Rates and Data Widths

Mode	PCLK or RXClk	Original PIPE Data Width (SerDes Data Width)
5.0 GT/s USB	125 Mhz	32 bits (40 bits)
5.0 GT/s USB	250 Mhz	16 bits (20 bits)
5.0 GT/s USB	500 Mhz	8 bits (10 bits)
10.0 GT/s USB	312.5 Mhz	32 bits (40 bits)
10.0 GT/s USB	625 Mhz	16 bits (20 bits)
10.0 GT/s USB	1250 Mhz	8 bits (10 bits)

Table 3-4. SATA Mode – Possible PCLK Rates and Data Widths

Mode	PCLK	Original PIPE Data Width (SerDes Data Width)	TXDataValid/RXDataValid Strobe Rate
1.5 GT/s SATA	600 Mhz	8 bits (10 bits)	1 in 4 PCLKs
1.5 GT/s SATA	300 Mhz	8 bits (10 bits)	1 in 2 PCLKs
1.5 GT/s SATA	150 Mhz	8 bits (10 bits)	N/A
1.5 GT/s SATA	75 Mhz	16 bits (20 bits)	N/A
1.5 GT/s SATA	37.5 Mhz	32 bits (40 bits)	N/A
3.0 GT/s SATA	300 Mhz	8 bits (10 bits)	N/A
3.0 GT/s SATA	150 Mhz	16 bits (20 bits)	N/A
3.0 GT/s SATA	75 Mhz	32 bits (40 bits)	N/A
3.0 GT/s SATA	600 Mhz	8 bits (10 bits)	1 in 2 PCLKs
6.0 GT/s SATA	600 Mhz	8 bits (10 bits)	N/A
6.0 GT/s SATA	300 Mhz	16 bits (20 bits)	N/A
6.0 GT/s SATA	150 Mhz	32 bits (40 bits)	N/A

Note: In SATA mode, if the PHY elasticity buffer is operating in nominal empty mode, RXDataValid may also be used when the EB is empty and no data is available.

Table 3-5. SATA Mode (SerDes Only) – Possible RXCLK Rates and Data Widths

Mode	RXCLK	Data Width
1.5 GT/s SATA	150 Mhz	10 bits
1.5 GT/s SATA	75 Mhz	20 bits
1.5 GT/s SATA	37.5 Mhz	40 bits
3.0 GT/s SATA	300 Mhz	10 bits
3.0 GT/s SATA	150 Mhz	20 bits
3.0 GT/s SATA	75 Mhz	40 bits
6.0 GT/s SATA	600 Mhz	10 bits
6.0 GT/s SATA	300 Mhz	20 bits
6.0 GT/s SATA	150 Mhz	40 bits

Table 3-6 shows possible PCLK and data width options for DisplayPort implementations.

Table 3-6. DPTX and DPRX Mode – Possible PCLK or RXCLK Rates and Data Widths

Mode	PCLK/RXCLK	Data Width
1.62 Gbps DisplayPort	162 Mhz	10 bits
	81 Mhz	20 bits
	40.5 Mhz	40 bits
2.16 Gbps DisplayPort (eDP)	216 Mhz	10 bits
	108 Mhz	20 bits
	54 Mhz	40 bits
2.43 Gbps DisplayPort (eDP)	243 Mhz	10 bits
	121.5 Mhz	20 bits
	60.75 Mhz	40 bits
2.7 Gbps DisplayPort	270 Mhz	10 bits
	135 Mhz	20 bits
	67.5 Mhz	40 bits
3.24 Gbps DisplayPort (eDP)	324 Mhz	10 bits
	162 Mhz	20 bits
	81 Mhz	40 bits
4.32 Gbps DisplayPort (eDP	432 Mhz	10 bits
	216 Mhz	20 bits
	108 Mhz	40 bits
5.4 Gbps DisplayPort	540 Mhz	10 bits
	270 Mhz	20 bits
	135 Mhz	40 bits
8.1 Gbps DisplayPort	810 Mhz	10 bits
	405 Mhz	20 bits
	202.5 Mhz	40 bits
10 Gbps DisplayPort	312.5 Mhz	40 bits[1]
13.5 Gbps DisplayPort	421.875 Mhz	40 bits[2]
20 Gbps DisplayPort	625 Mhz	40 bits[2]

1. 40-bit data width is for consistency with other protocols. For block-encoded DisplayPort modes (that is, 10 Gbps, 13.5 Gbps, and 20 Gbps), the controller utilizes only 8 out of every 10 bits of data. See Section 6.1.1 for more details.

Table 3-7. USB4 Mode – Possible PCLK or RXCLK Rates and Data Widths (Sheet 1 of 2)

Mode	PCLK/RXCLK	Data Width[1]
10 GT/s USB4	1.25 Ghz	10 bits
	625 Mhz	20 bits
	312.5 Mhz	40 bits
20 GT/s USB4	2.5 Ghz	10 bits
	1.25 Ghz	20 bits
	625 Mhz	40 bits
40 GT/s (25.6 GT/s PAM3)	914 Mhz	56 bits

Table 3-7. USB4 Mode – Possible PCLK or RXCLK Rates and Data Widths (Sheet 2 of 2)

Mode	PCLK/RXCLK	Data Width[1]
10.3125 GT/s USB4 (Thunderbolt™ 2)	1.2890625 Ghz	10 bits
	644.53125 Mhz	20 bits
	322.265625 Mhz	40 bits
20.625 GT/s USB4 (Thunderbolt™ 2)	2.578125 Ghz	10 bits
	1.2890625 Ghz	20 bits
	644.53125 Mhz	40 bits

1. While the data widths are 10, 20, or 40 bits for consistency with other protocols, USB4 only utilizes only 8 out of every 10 bits of data since it uses block encoding. See Section 6.1.1 for more details.

Note: When a MAC that implements the TXDataValid signal is using a mode that does not use TXDataValid the MAC shall keep TXDataValid asserted. When a PHY that implements RXDataValid is in a mode that does not use RXDataValid the PHY mustkeep RXDataValid asserted.

There may be PIPE implementations that support multiples of these configurations. PHY implementations that support multiple configurations at the same rate must support the width and PCLK rate control signals. A PHY that supports multiple rates in PCIe mode or SATA mode or USB mode must support configurations across all supported rates that are fixed at the PCLK rate. A PHY that supports multiple rates in PCIe mode or SATA mode must support configurations across all supported rates that are fixed data path width.

4 PCIe, USB, USB4, and DisplayPort PHY Functionality

Figure 4-1 shows the functional block diagram of the PHY for a Tx differential pair and Rx differential pair combination. The functional blocks shown are not intended to define the internal architecture or design of a compliant PHY but to serve as an aid for signal grouping. Functional PHY diagrams illustrating Tx+Tx and Rx+Rx combinations are provided in Figure 4-2 and Figure 4-3, respectively. Note that while these diagrams illustrate the scenario where the Phase Lock Loop (PLL) is in the PHY, other topologies are possible where the PLL is external to the PHY as described in Section 8.1.1.

Figure 4-1. PHY Functional Block Diagram for Tx+Rx Usage

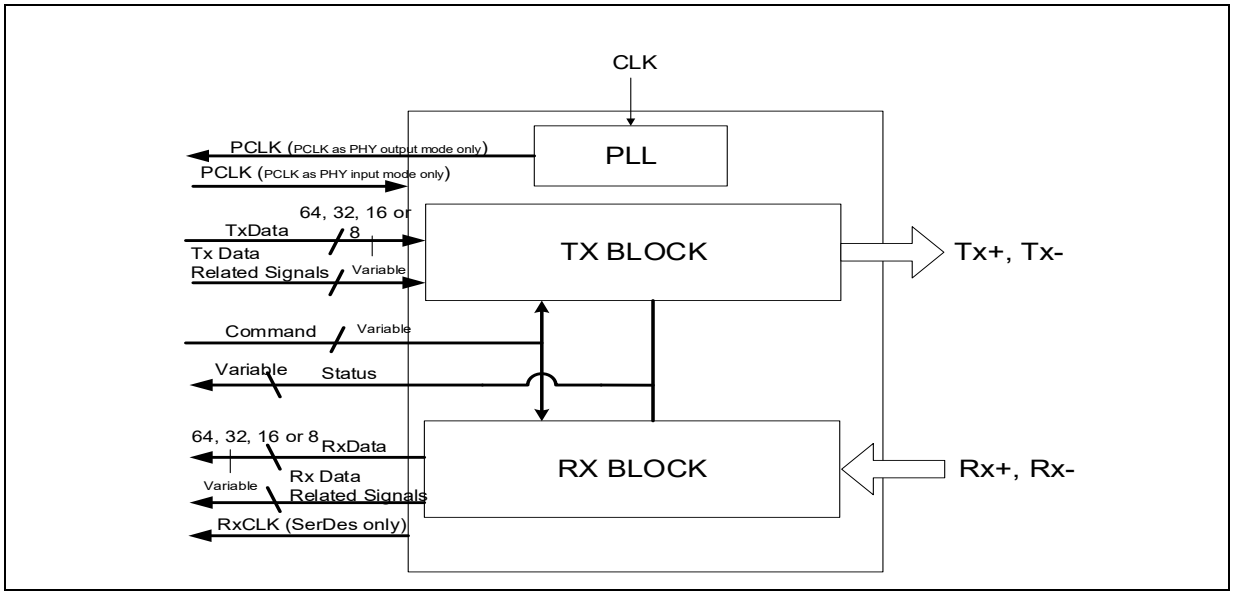


Figure 4-2. PHY Functional Diagram for Tx+Tx Usage Case

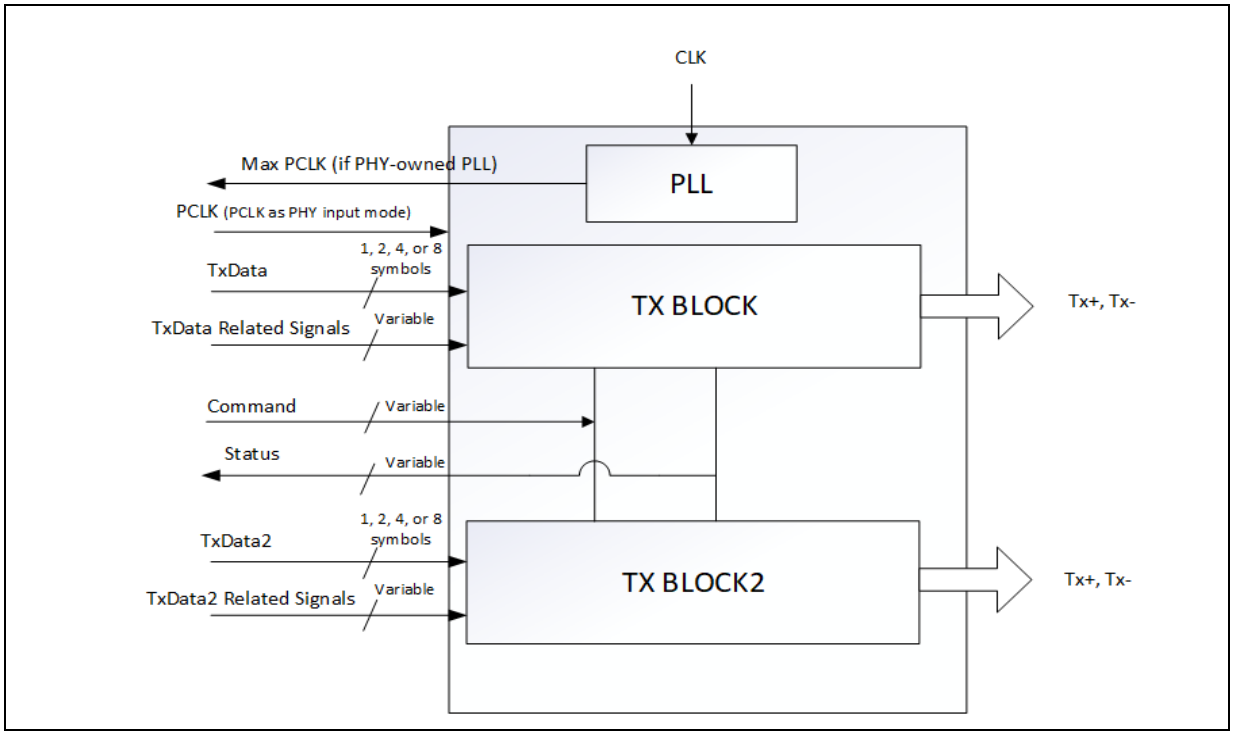
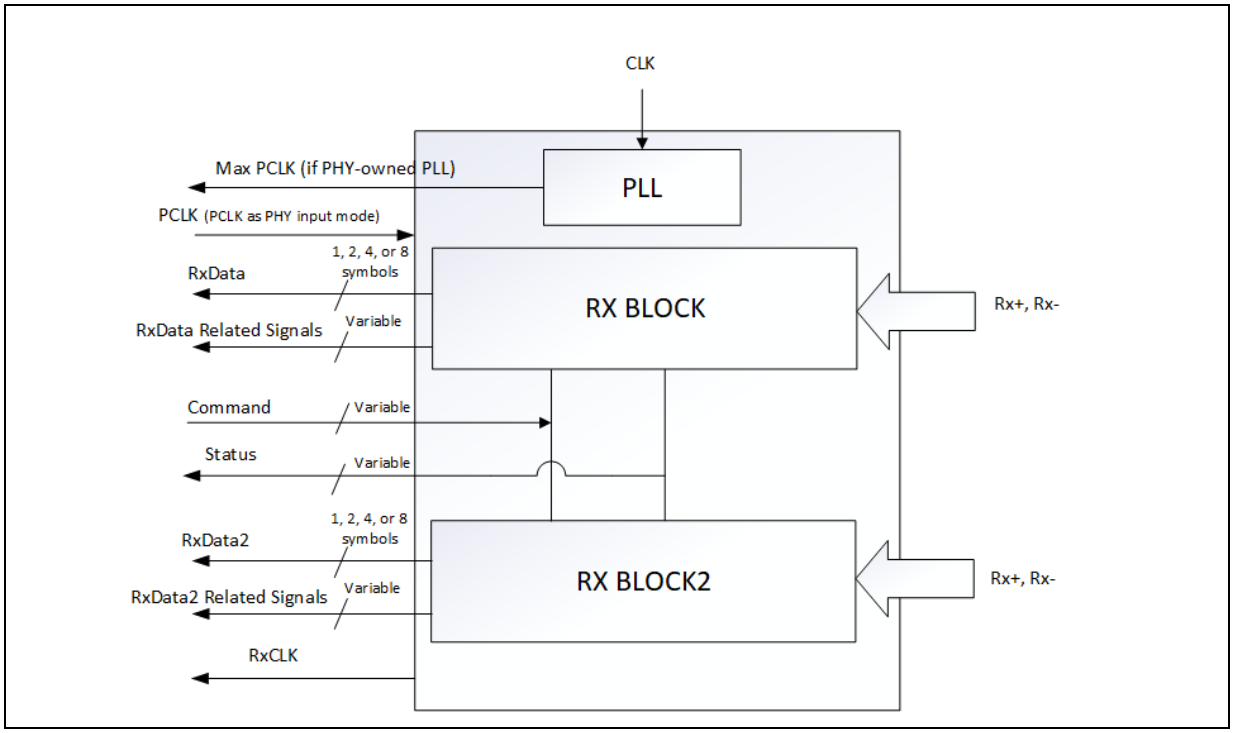


Figure 4-3. PHY Functional Diagram for Rx+Rx Usage Case



Section 4.1 and Section 4.2 provide descriptions of each of the blocks shown in Figure 4-1, Figure 4-2, Figure 4-3. These blocks represent high-level functionality that is required to exist in the PHY implementation. These descriptions and diagrams describe general architecture and behavioral characteristics. Different implementations are possible and acceptable.

4.1 Original PIPE Architecture

Figure 4-4. Transmitter Block Diagram (2.5 and 5.0 GT/s)

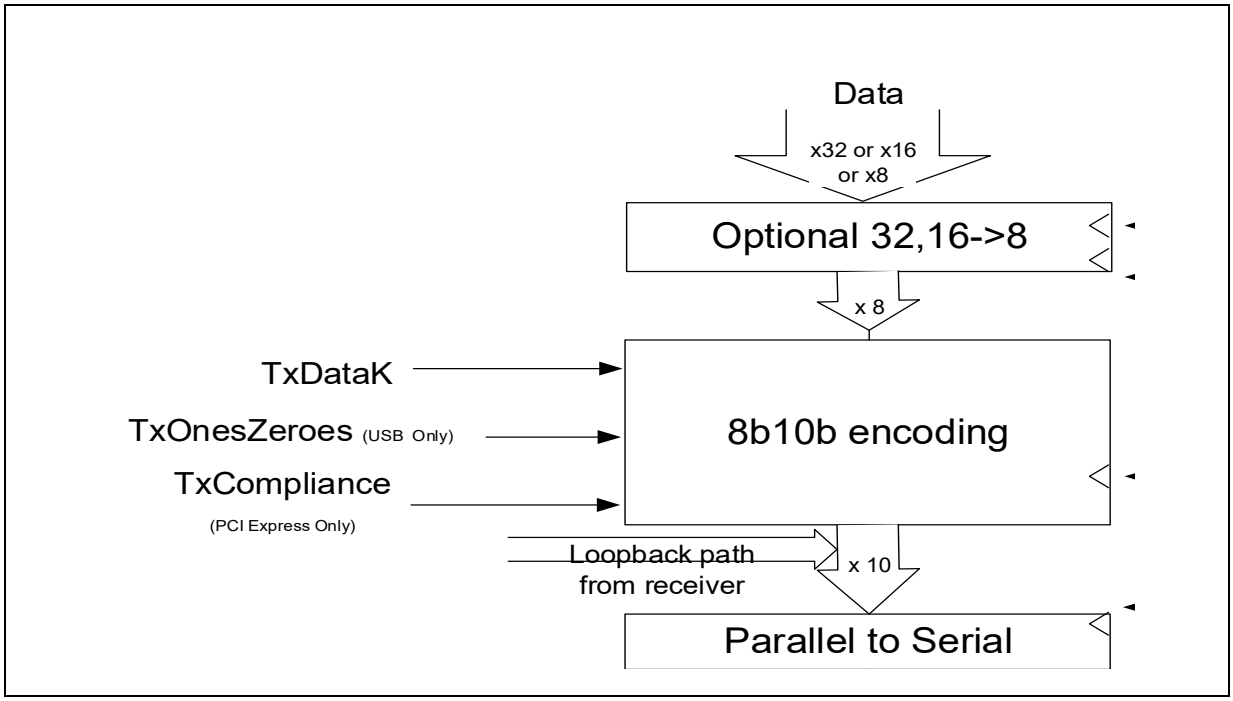


Figure 4-5. Transmitter Block Diagram (8.0/10/16/32 GT/s)

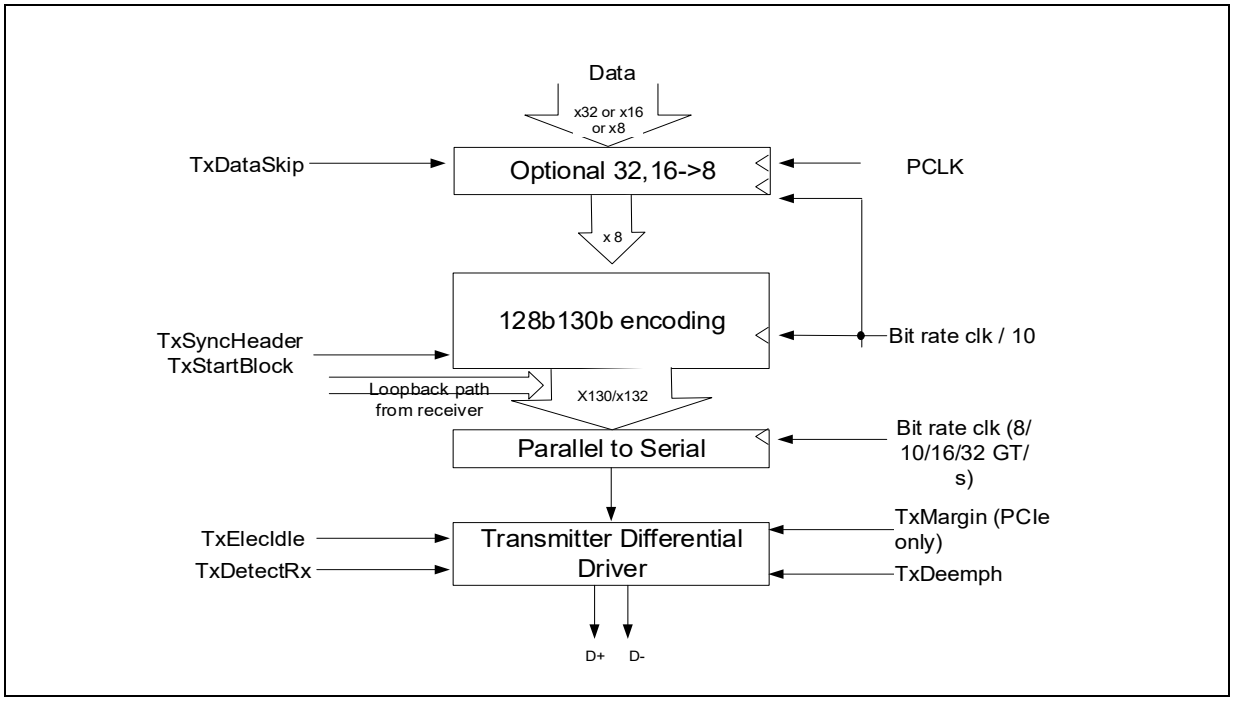


Figure 4-6. Receiver Block Diagram (2.5 and 5.0 GT/s)

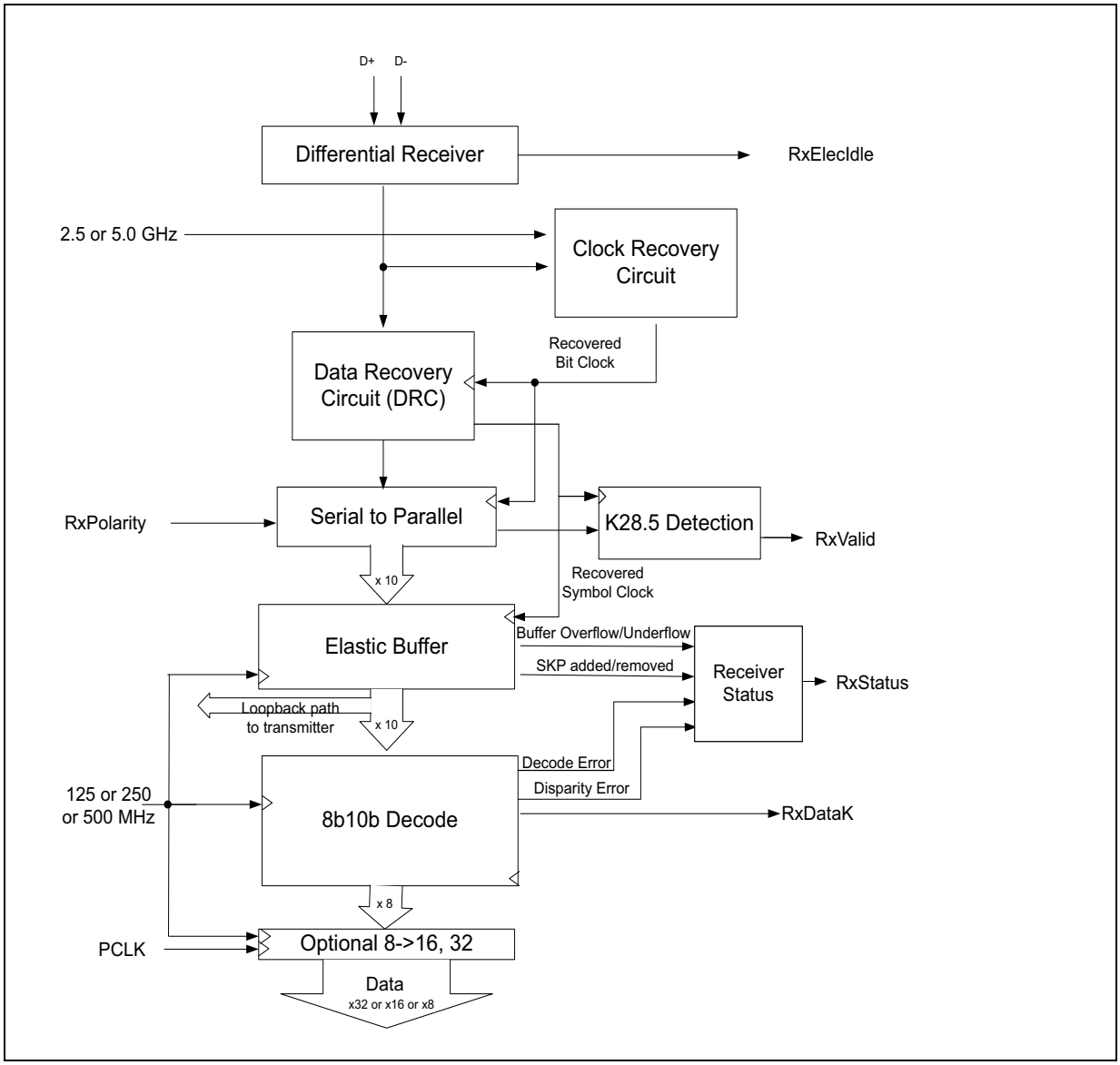
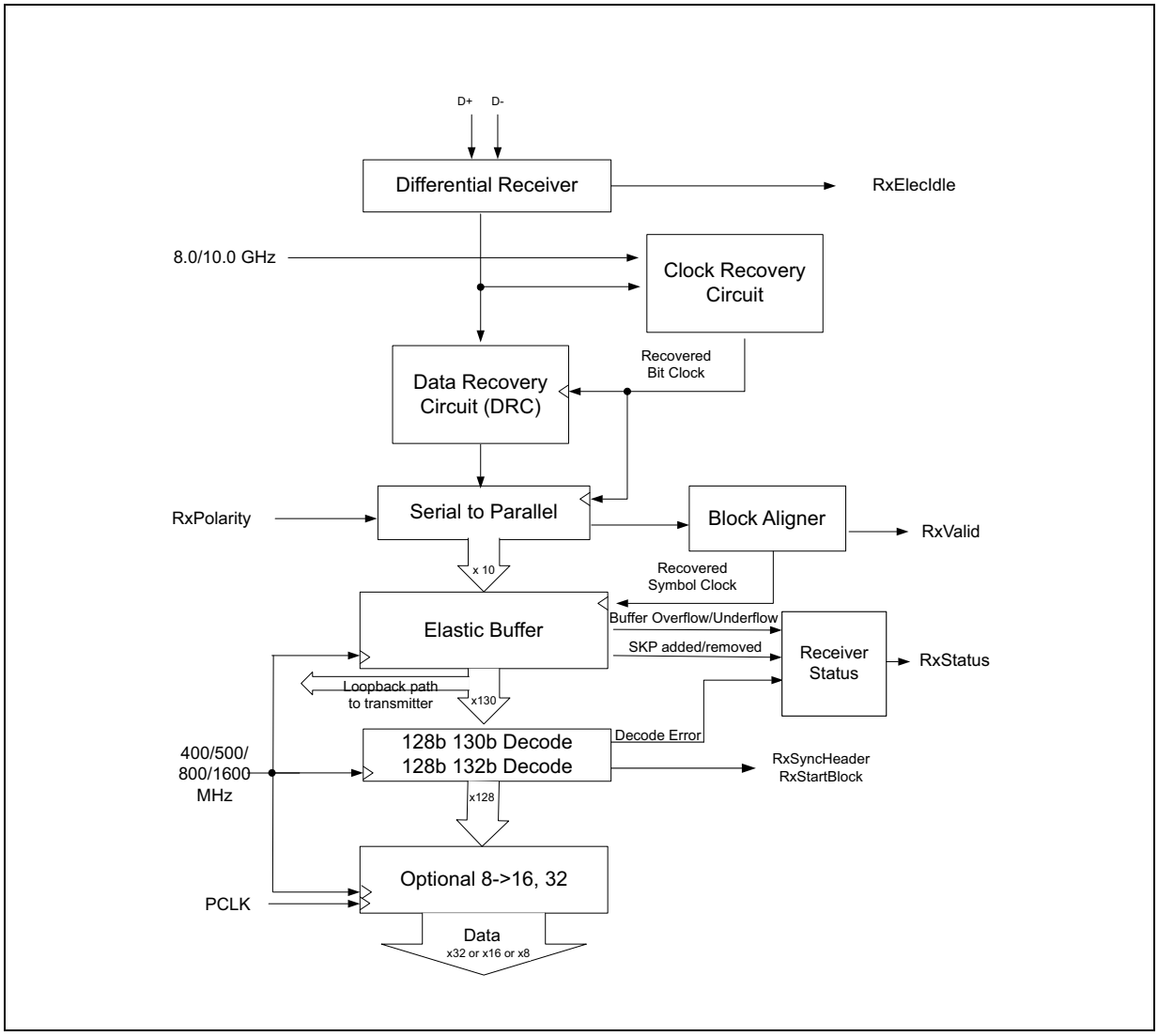


Figure 4-7. Receiver Block Diagram (8.0/10/16 GT/s)



4.2 SerDes Architecture

With the SerDes architecture, the PHY implements minimal digital logic compared to the original PIPE architecture. Figure 4-8 shows the transmitter functionality implemented in the PHY. The data received from the MAC goes through a parallel to serial converter before being driven out on differential wires. Note that in the SerDes architecture, all loopback logic resides in the MAC. Figure 4-9 shows the receiver functionality implemented in the PHY. The data received on the input differential wires goes through a serial to parallel converter before being forwarded to the MAC along with a recovered clock, RXCLK.

Figure 4-8. SerDes Architecture: PHY Transmitter Block Diagram

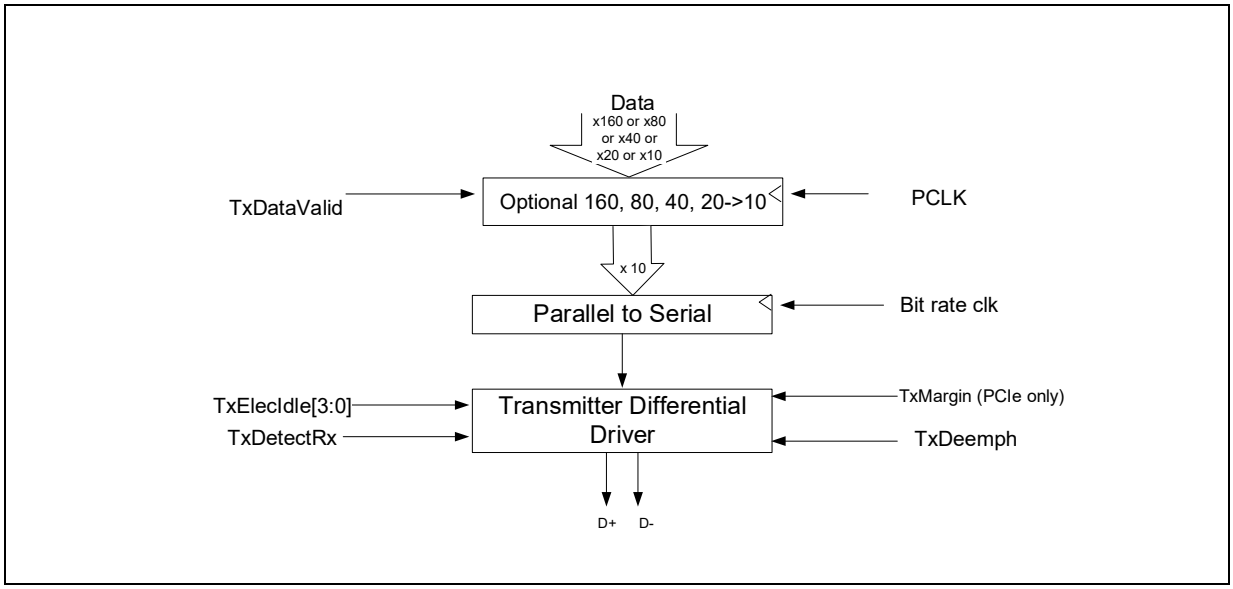
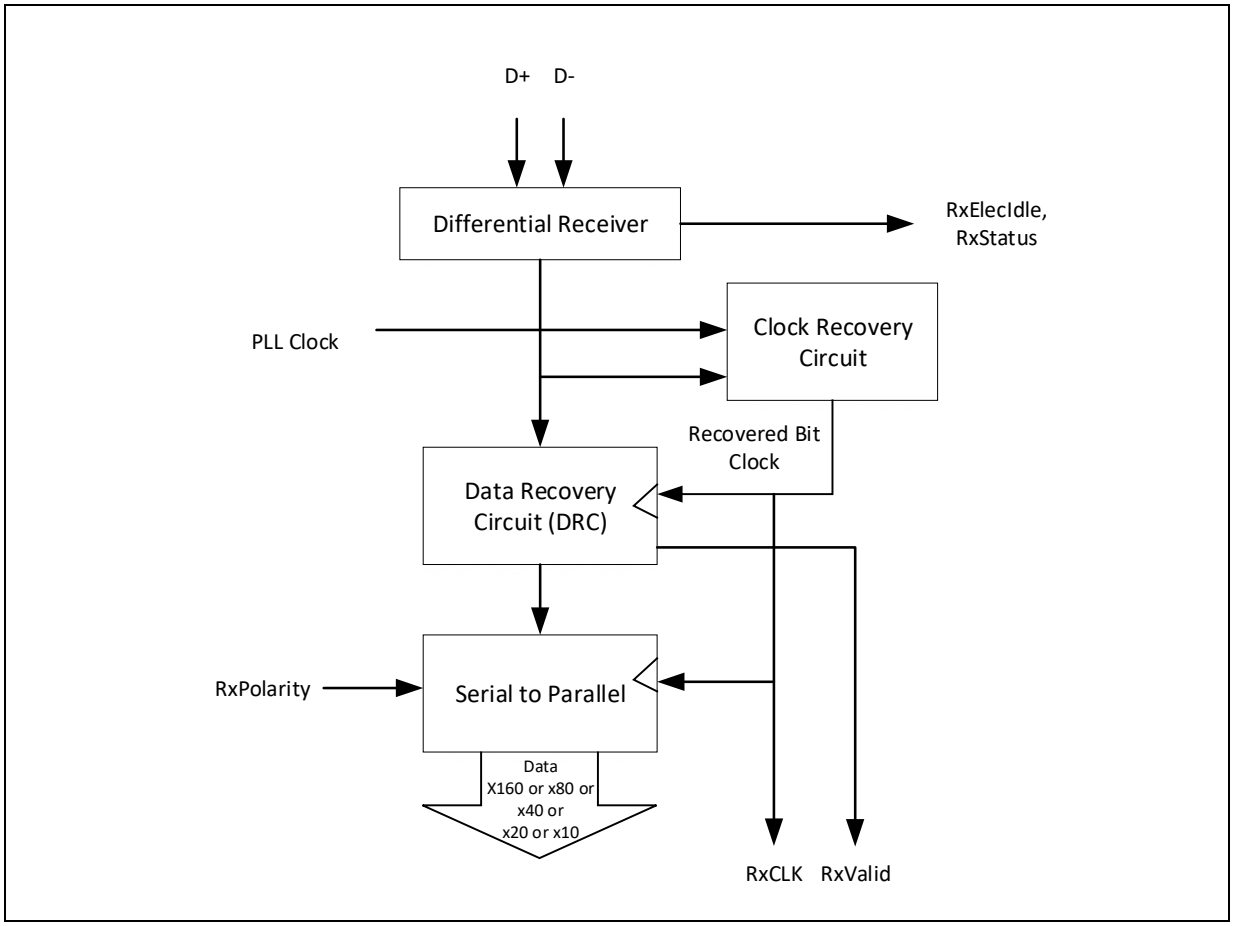


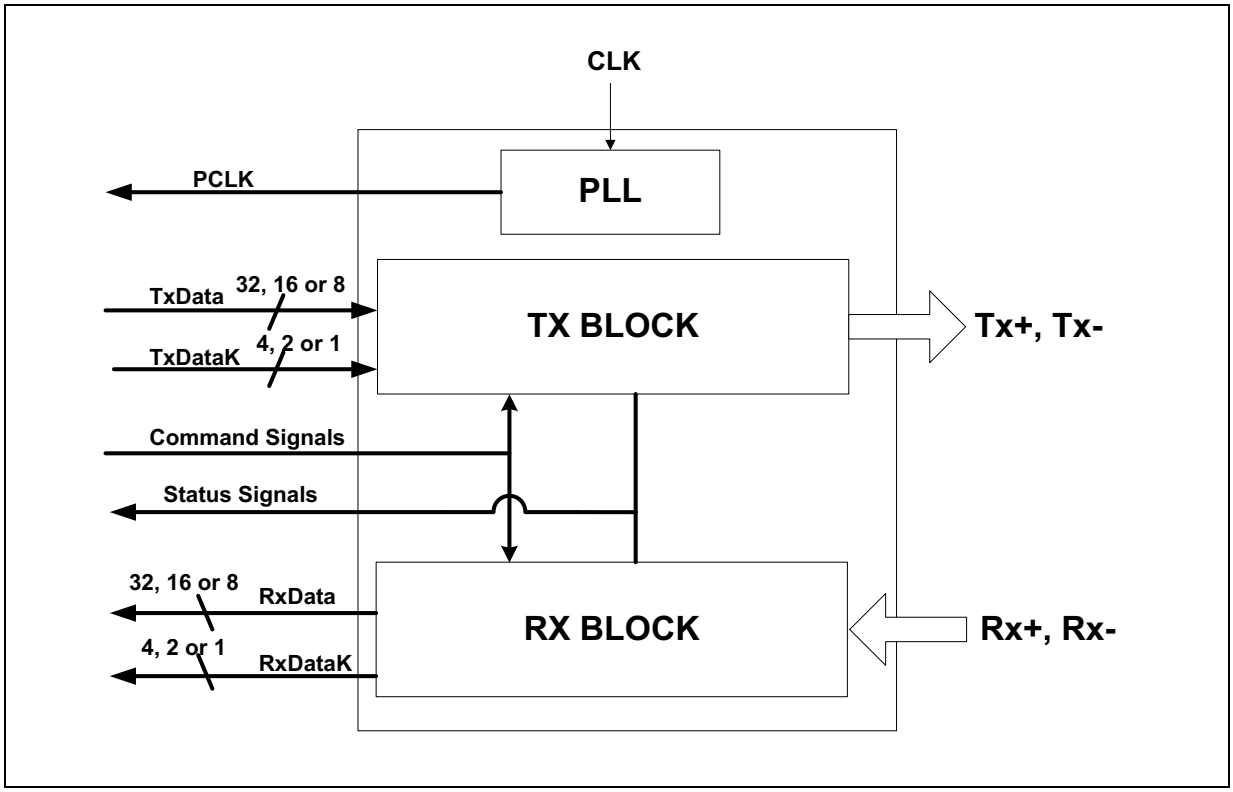
Figure 4-9. SerDes Architecture: PHY Receiver Block Diagram



5 SATA PHY Functionality

Figure 4-1 shows the functional block diagram of a SATA PHY. The functional blocks shown are not intended to define the internal architecture or design of a compliant PHY but to serve as an aid for signal grouping.

Figure 5-1. PHY Functional Block Diagram



The following sections provide descriptions of each of the blocks shown in Figure 5-1. These blocks represent high-level functionality that is required to exist in the PHY implementation. These descriptions and diagrams describe general architecture and behavioral characteristics. Different implementations are possible and acceptable.

Figure 5-2. Transmitter Block Diagram (1.5, 3.0, and 6.0 GT/s)

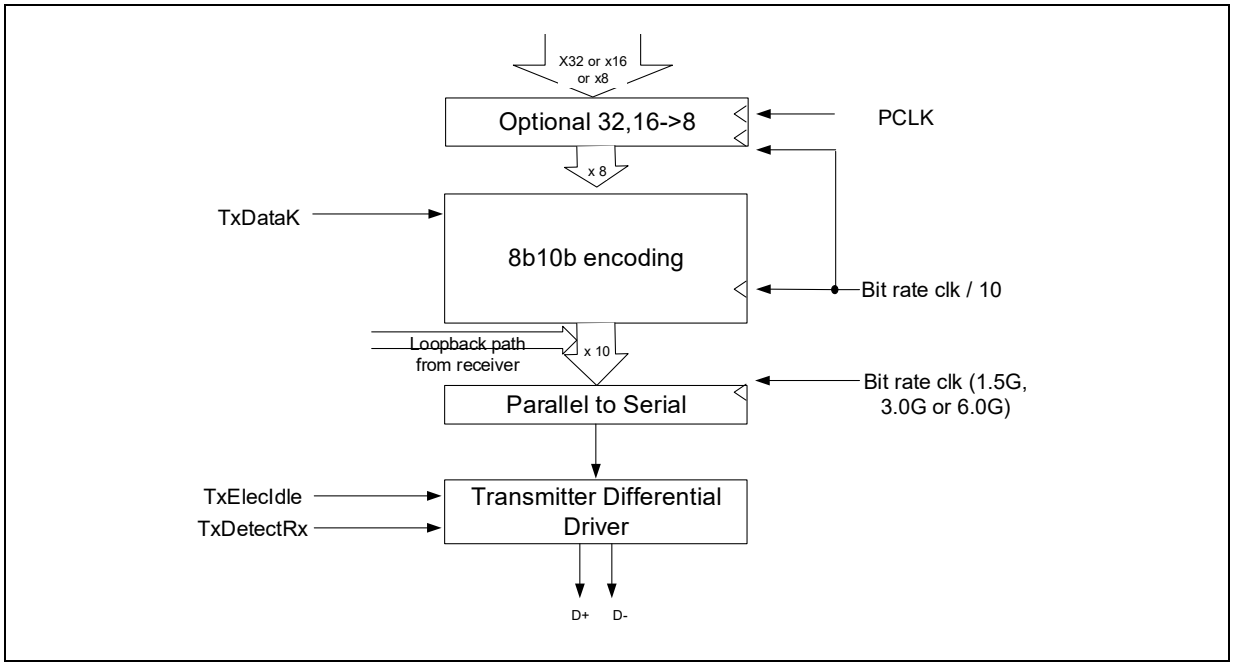
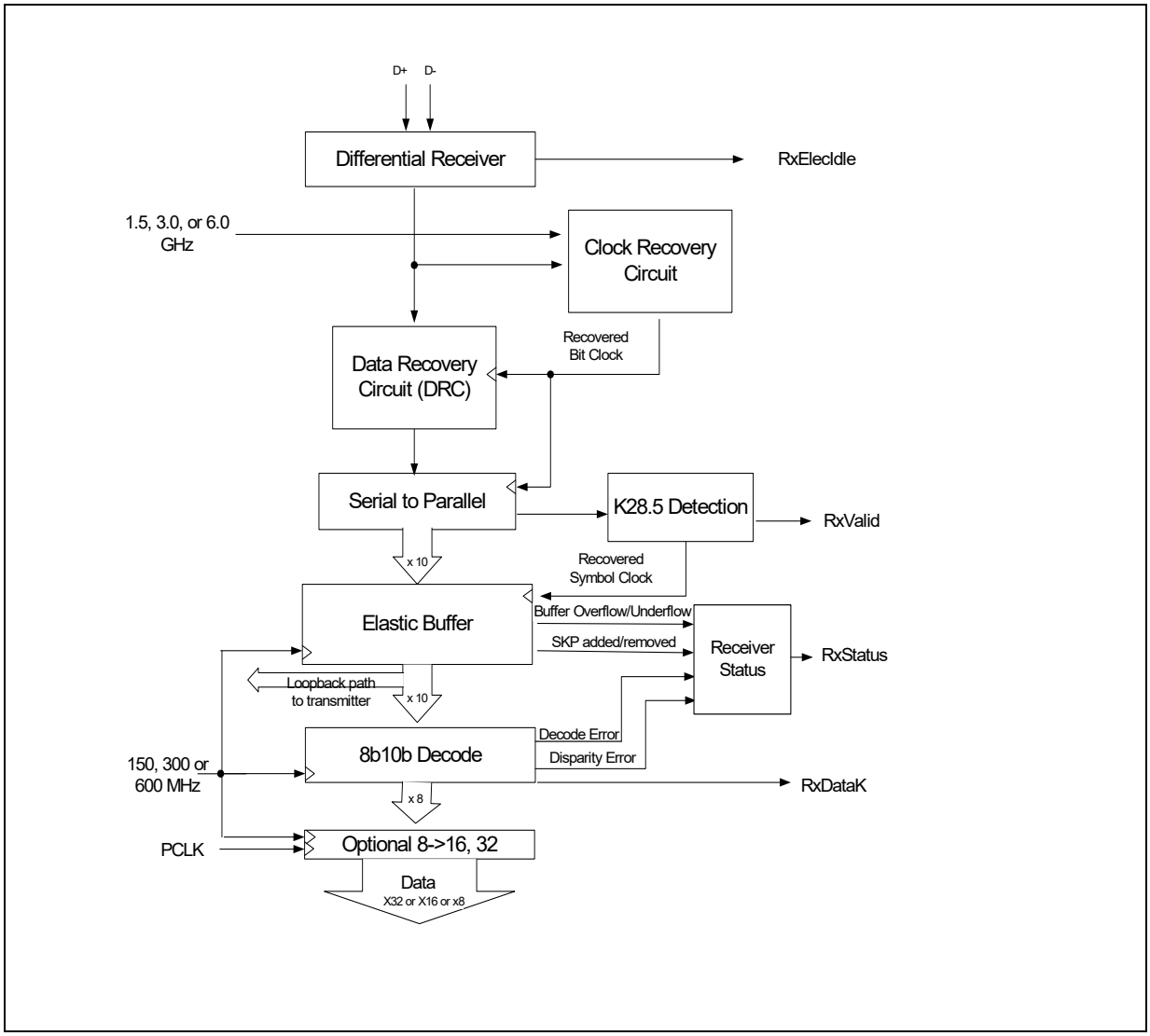


Figure 5-3. Receiver Block Diagram (1.5, 3.0, and 6.0 GT/s)



6 PIPE Interface Signal Descriptions

The PHY input and output signals are described in the following tables. Note that Input/Output is defined from the perspective of a PIPE compliant PHY component. Therefore, a signal described as an "Output" is driven by the PHY and a signal described as an "Input" is received by the PHY. A basic description of each signal is provided. More details on their operation and timing can be found in following sections. All signals on the "parallel" side of a PIPE implementation are synchronous with PCLK, with exceptions noted in the following tables. In the SerDes architecture, RxData is synchronous with RxCLK. PHYs that only support SerDes architecture do not require the signals marked as "not used in the SerDes architecture", however, PHYs that support both original PIPE and SerDes architecture must implement all the signals. Each signal has a column that indicates the relevant protocols; USB refers to USB 3.2 and lower, while USB4 is indicated separately.

As described in Section 2.8, up to two differential pairs are operational at any given time. For PIPE control signals that refer to Rx functionality, a control signal applies to both Rx and Rx2 unless separate control signals are defined for each of the differential pairs. For PIPE control signals that refer to Tx functionality, a control signal applies to both Tx and Tx2 unless separate control signals are defined for each of the two differential pairs.

Note: For USB4 and DisplayPort, the low speed side channel is not part of the PIPE definition, however, the appendix lists the DisplayPort AUX signals.

6.1 PHY/MAC Interface Signals – Common for SerDes and Original PIPE

This section describes signals that are applicable to both SerDes architecture and original PIPE. Any deltas in usage between the two architectures are noted in the description.

6.1.1 Data Interface

Table 6-1. Tx Data Interface Input Signals

Name	Active Level	Description	Relevant Protocols
Original PIPE: TxData[31:0] for 32-bit interface TxData[15:0] for 16-bit interface TxData[7:0] for 8-bit interface SerDes arch: TxData[159:0] for 160-bit interface TxData[79:0] for 80-bit interface TxData[39:0] for 40-bit interface TxData[19:0] for 20-bit interface TxData[9:0] for 10-bit interface Serdes arch: (USB4 only) TxData[55:0] for 56-bit interface	N/A	Parallel data input bus for Tx differential pair. For Original PIPE architecture, the TxData signal width options are 32, 16, and 8 bits. For the 16-bit interface, 16 bits represent two symbols of Tx data. Bits [7:0] are the first symbol to be transmitted, and bits [15:8] are the second symbol. For the 32-bit interface, 32 bits represent the 4 symbols of Tx data. Bits [23:16] are the third symbol to be transmitted, and bits [31:24] are the fourth symbol. Bit zero is the first to be transmitted. For the SerDes architecture, the TxData signal width options are 160, 80, 40, 20, and 10 bits. For the 80-bit interface, 80 bits represent eight symbols of Tx data. Bits [49:40], bits [59:50], bits [69:60], and bits [79:70] are the fifth, sixth, seventh, and eighth symbols, respectively. For the 160-bit interface, 160 bits represent 16 symbols of Tx data. Bits [89:80], bits [99:90], bits [109:100], bits [119:110], bits [129:120], bits [139: 130], bits [149:140], bits [159:150] are the ninth, tenth, eleventh, twelfth, thirteenth, fourteenth, fifteenth, and sixteenth symbols, respectively. For block encoded data[1], only 8 bits out of each 10-bit slice are used, for example, [7:0] represent byte0, [9:8] are reserved, [17:10] represent byte1, and [19:18] are reserved, and so forth. Bit zero is the first to be transmitted. Additionally for SerDes architecture, USB4 uses a 56-bit interface instead of an 80-bit interface. Please refer to Section 8.34.3 for details of the 56-bit interface.	PCIe, SATA, USB, DisplayPort Tx, and USB4
SerDes arch: TxData2[55:0] for 56-bit interface TxData2[39:0] for 40-bit interface TxData2[19:0] for 20-bit interface TxData2[9:0] for 10-bit interface	N/A	Parallel data input bus for the Tx2 differential pair. For the SerDes architecture, the TxData signal width options are 56, 40, 20, and 10 bits. For the 40-bit interface, 40 bits represent four symbols of Tx data. Bits [9:0], bits [19:10], bits [29:20], and bits [39:30] are the represent the first, second, third, and fourth symbols, respectively. Please refer to Section 8.34.3 for details of the 56-bit interface.	DisplayPort Tx, USB4
TxDataValid	N/A	PCI Express mode and SATA mode and USB mode (original PIPE-only): This signal allows the MAC to instruct the PHY to ignore the data interface for one clock cycle. A value of one indicates that the PHY will use the data, and a value of zero indicates that the PHY will not use the data. It is required that the MAC assert TxDataValid at all times when the PHY is in a mode that does not require the signal. All PCI Express modes at 8 GT/s, 16 GT/s, 32 GT/s, 64 GT/s,and 128 GT/s and all USB modes at 10 GT/s use TxDataValid. See Table 3-1, Table 3-2, and Table 3-3 for a list of other modes that use TxDataValid. See Section 8.26 for details on USB usage; this signal is not applicable to USB SerDes architecture designs.	PCIe, SATA, USB (original PIPE-only)

1. For PCIe operating at 8 GT/s or higher link speed, USB4, and USB 10 GT/s link speed, the data bits are utilized as per the block encoded data description detailed in the previous tables. For all other modes, all the data bits are utilized.

Table 6-2. Tx Data Interface Output Signals (Sheet 1 of 2)

Name	Active Level	Description	Relevant Protocols

Table 6-2. Tx Data Interface Output Signals (Sheet 2 of 2)

Tx+, Tx-	N/A	The Tx differential outputs from the PHY. All transmitters must be AC-coupled to the media. See Section 4.3.1.2 of the PCI Express base specification or Section 6.2.2 of the USB 3.2 specification.	PCIe, SATA, USB, DisplayPort Tx, and USB4
Tx2+, Tx2-	N/A	The Tx2 differential outputs from the PHY. All transmitters must be AC-coupled to the media.	DisplayPort Tx, USB4

Table 6-3. Rx Data Interface Input Signals

Name	Active Level	Description	Relevant Protocols
Rx+, Rx-	N/A	The Rx differential inputs to the PHY	PCIe, SATA, USB, DisplayPort Rx, and USB4
Rx2+, Rx2-	N/A	The Rx2 differential inputs to the PHY	DisplayPort Rx, USB4

Table 6-4. Rx Data Interface Output Signals

Name	Active Level	Description	Relevant Protocols
Original PIPE: RxData[31:0] for 32-bit interface RxData[15:0] for 16-bit interface or RxData[7:0] for 8-bit interface SerDes arch: RxData[159:0] RxData[79:0] for 80-bit interface RxData[39:0] for 40-bit interface RxData[19:0] for 20-bit interface or RxData[9:0] for 10-bit interface Serdes arch (USB4 only): RxData[55:0] for 56-bit interface	N/A	Parallel data output bus for Rx diff pair. For the 16-bit interface, 16 bits represent two symbols of Rx data. Bits [7:0] are the first symbol received, and bits [15:8] are the second symbol. For the 32-bit interface, 32 bits represent the four symbols of Rx data. Bits [23:16] are the third symbol received, and bits [31:24] are the fourth symbol received. Bit zero is the first bit received. When the PHY is in SATA mode, the first valid data following an ALIGN primitive must appear as byte zero in the Rx data. For the SerDes architecture, the RxData signal width options are 160, 80, 40, 20, and 10 bits. For the 80-bit interface, 80 bits represent eight symbols of Rx data. Bits [49:40], bits [59:50], bits [69:60], and bits [79:70] are the fifth, sixth, seventh, and eighth symbols, respectively. For the 160-bit interface, 160 bits represent 16 symbols of Rx data. Bits [89:80], bits [99:90], bits [109:100], bits [119:110], bits [129:120], bits [139: 130], bits [149:140], bits [159:150] are the ninth, tenth, eleventh, twelfth, thirteenth, fourteenth, fifteenth, and sixteenth symbols, respectively. For block-encoded data[1], only 8 bits out of each 10-bit slice are used, for example, [7:0] represent byte0, [9:8] are reserved, [17:10] represent byte1, and [19:18] are reserved, and so forth. Bit zero is the first bit received. Additionally for SerDes architecture, USB4 uses a 56-bit interface instead of an 80-bit interface. Please refer to Section 8.34.3 for details of the 56-bit interface. RxData is synchronous to RxCLK in the SerDes mode.	PCIe, SATA, USB, DisplayPort Rx, USB4
SerDes arch: RxData2[55:0] for 56-bit interface, RxData2[39:0] for 40-bit interface, RxData2[19:0] for 20-bit interface, or RxData2[9:0] for 10-bit interface	N/A	Parallel data output bus for Rx2 diff pair. For SerDes architecture, the RxData signal width options are 56, 40, 20, and 10 bits. For the 40-bit interface, 40 bits represent 4 symbols of receive data. Bits [9:0], bits [19:10], bits[29:20], and bits[39:30] represent the first, second, third, and fourth symbols, respectively. Please refer to Section 8.34.3 for details of the 56-bit interface. RxData2 is synchronous to RxCLK2 in SerDes mode.	DisplayPort RX, USB4

1. For PCIe operating at 8 GT/s or higher link speed, USB4, and USB 10 GT/s link speed, the data bits are utilized as per the block encoded data description detailed in the previous tables. For all other modes, all the data bits are utilized.

6.1.2 Command Interface

Table 6-5. Command Interface Input Signals (Sheet 1 of 13)

<table>
<tr><th>Name</th><th>Active Level</th><th colspan="2">Description</th><th>Relevant Protocols</th></tr>
<tr><td rowspan="12">PHY mode[3:0]</td><td rowspan="12">N/A</td><td colspan="2">Selects the PHY operating mode.</td><td rowspan="12">PCIe, SATA, USB, DisplayPort, and USB4</td></tr>
<tr><td>Value</td><td>Description</td></tr>
<tr><td>0</td><td>PCIe</td></tr>
<tr><td>1</td><td>USB</td></tr>
<tr><td>2</td><td>SATA</td></tr>
<tr><td>3</td><td>DisplayPort</td></tr>
<tr><td>4</td><td>Reserved</td></tr>
<tr><td>5</td><td>Reserved</td></tr>
<tr><td>6</td><td>Reserved</td></tr>
<tr><td>7</td><td>USB4</td></tr>
<tr><td>All Others Reserved</td><td></td></tr>
<tr><td colspan="2">Implementation of this signal is not required for PHYs that only support a single mode.
The MAC is permitted to change this signal only during Reset# assertion. This signal is asynchronous.</td></tr>
<tr><td>DP_mode_Tx_Rx</td><td>N/A</td><td colspan="2">This signal is used to distinguish between DPTx and DPRx when PHY mode=0x3. A value of “0” specifies DPTx; a value of “1” specifies DPRx.

The MAC is permitted to change this signal only during the Reset# assertion. This signal is asynchronous.</td><td>DisplayPort</td></tr>
<tr><td>SerDesArch</td><td>High</td><td colspan="2">This signal indicates whether the SerDes architecture is enabled. Displayport and USB4 must always set this to “1”.

The MAC is permitted to change this signal only during the Reset# assertion. This signal is asynchronous.</td><td>PCIe, SATA, USB, DisplayPort, and USB4</td></tr>
<tr><td>SRISEnable</td><td>High</td><td colspan="2">Used to tell the PHY to configure itself to support Separate Reference Clock (Refclk) with Independent Spread Spectrum Clocking (SRIS) for PCIe.

SRISEnable must be set by the MAC before the first receiver detection. The PHY internally does sequencing and gates the exit to P0 with having setup for SRIS if SRISEnable is asserted.

PHYs may have implementation-specific requirements for how early this signal must be stable, for example, before configuring the PLL. For PCIe Rev 5.0 and earlier MACs, this signal must not change after that point.

For PCIe Rev 6.0 capable and newer MACs, this signal is permitted to change during link training in the Configuration LTSSM state. Specifically, this signal may be asserted initially and then deassert during link training</td><td>PCIe</td></tr>
</table>

Table 6-5. Command Interface Input Signals (Sheet 2 of 13)

Name	Active Level	Description	Relevant Protocols
TxDetectRx/ Loopback	High	Used to tell the PHY to begin a receiver detection operation or to begin loopback or to signal LFPS during P0 for USB Polling state. See Section 8.20 and Section 8.21 for details on the required values for all control signals to perform loopback and receiver detection operations and to signal Polling.LFPS. **Note:** loopback usage is not applicable to SerDes architecture since loopback is implemented by the MAC in SerDes. For receive detect in PHY power states where the PCLK can be gated, this signal is asynchronous; in all other states, it is synchronous to PCLK. SATA mode: Loopback support is optional for SATA PHYs. Loopback is only valid in SATA mode when EncodeDecodeBypass is asserted. The Rx elasticity buffer must be active during loopback. If the PHY runs out of data to transmit during loopback – it must transmit ALIGNs. TxDetectRx is not used in SATA mode. USB4 mode, Displayport TX mode: This signal is used when the MAC is configured to generate LFPS via the MacTransmitLFPS bit in the PHY Common Control0 register. This signal indicates to the PHY that the LFPS pattern is being transmitted. This corresponds to the Tx differential pair.	PCIe, SATA, USB, DisplayPort TX, USB4
TxDetectRx2	High	Refer to TxDetectRx for description. This signal corresponds to the Tx2 differential pair.	USB4, DisplayPort Tx
TxElecIdle[3:0]	High	Forces Tx output to electrical idle when asserted except in loopback. See Section 8.21, Section 8.22, or Section 8.23 for the full description and usage of this pin. **Note:** The MAC must always have TxDataValid asserted when TxElecIdle transitions to either asserted or deasserted; TxDataValid is a qualifier for TxElecIdle sampling. See Section 8.3 for the definitions of the PHY power states. For original PIPE architecture and for non-PCIe mode SerDes architecture, only bit zero of this signal is used and all other bits are reserved. For SerDes architecture in PCIe mode at 32 GT/s or lower rate, one bit is required per two symbols of interface data. For example, for an eight-symbol wide interface, bit zero would apply to symbols 0 and 1, bit 1 would apply to symbols 2 and 3, bit 2 would apply to symbols 4 and 5, bit 3 would apply to symbols 6 and 7. For narrower interfaces, unused bits of this signal are reserved. This is due to EIOS truncation rules in Section 4.2.4.2 of the PCIe 4.0 Base specification and due to the maximum time to transition to a valid electrical idle after sending an EIOS. For the SerDes architecture in PCIe mode at 64 GT/s rate and 128 GT/s, one bit is required per four symbols of interface data. For example, for an eight-symbol wide interface, bit zero would apply to symbols 0 through 3, and bit 1 would apply to symbols 4 through 7, etc. Unused bits of this signal are reserved.	PCIe, SATA USB, USB4, DisplayPort
TxElecIdle2	High	This signal is used in asymmetric mode and corresponds to the Tx2 differential pair. Please refer to TxElecIdle for more details.	USB4, DisplayPort Tx
Reset#	Low	Resets the transmitter and receiver. This signal is asynchronous and is permitted to assert at any time regardless of PHY state. The PHY must hold itself in its lowest power state on the lane during the Reset# assertion, irrespective of values on other PIPE signals. The PHY reports its default power state after reset as defined in Section 8.1.2.	PCIe, SATA, USB, DisplayPort, and USB4

Table 6-5. Command Interface Input Signals (Sheet 3 of 13)

<table>
<tr><th>Name</th><th>Active Level</th><th>Description</th><th>Relevant Protocols</th></tr>
<tr><td>PowerDown[3:0]</td><td>N/A</td><td>
Powers up or down the transceiver. Power states of the PCIe mode:
[3] [2] [1] [0] Description
0 0 0 0 P0, normal operation
0 0 0 1 P0s, low recovery time latency, power saving state
0 0 1 0 P1, longer recovery time latency, lower power state
0 0 1 1 P2, lowest power state
0 1 0 0 POWER_STATE_4 PHY-specific
0 1 0 1 POWER_STATE_5 PHY-specific
0 1 1 0 POWER_STATE_6 PHY-specific
0 1 1 1 POWER_STATE_7 PHY-specific
1 0 0 0 POWER_STATE_8 PHY-specific
1 0 0 1 POWER_STATE_9 PHY-specific
1 0 1 0 POWER_STATE_10 PHY-specific
1 0 1 1 POWER_STATE_11 PHY-specific
1 1 0 0 POWER_STATE_12 PHY-specific
1 1 0 1 POWER_STATE_13 PHY-specific
1 1 1 0 POWER_STATE_14 PHY-specific
1 1 1 1 POWER_STATE_15 PHY-specific

In PCLK as the PHY output mode, when transitioning from P2 to P1, the signaling is asynchronous (since PCLK is not running).

A PIPE PHY that supports PCIe L1 PM substates managed exclusively via this PowerDown signal must support at least one PHY-specific power state, meeting each of the requirements shown in the following table, in addition to the legacy power states. If the PHY supports multiple suitable states with different exit latencies it is the responsibility of the MAC to decide which states to use.
<table>
<tr><th>PCLK State</th><th>Tx Common mode State</th><th>RxElecIdle Supported</th><th>When to return PhyStatus when exiting?</th><th>Exit Latency to P0</th></tr>
<tr><td>Off</td><td>Off</td><td>No</td><td>Before transmit common mode established</td><td>Implementation Specific</td></tr>
<tr><td>Off</td><td>On</td><td>No</td><td>N/A</td><td>Implementation Specific</td></tr>
<tr><td colspan="5">When managing L1 substates via sideband signals, the PHY must define at least one PowerDown encoding where the PCLK can be turned off and the TxCommonmodeState and RxElecIdle are controlled through TxCommonmodeDisable and RxEIDetectDisable; in this case, the MAC must hold the PowerDown value constant when in L1 substates.</td></tr>
</table>
</td><td>PCIe, USB, and USB4</td></tr>
</table>

Table 6-5. Command Interface Input Signals (Sheet 4 of 13)

<table>
<tr><th>Name</th><th>Active Level</th><th>Description</th><th>Relevant Protocols</th></tr>
<tr><td></td><td></td><td>
If PowerDown changes during the IORecal, RxEqEval, or RxEqTraining operations, the PHY must abort the request and return the handshake acknowledgment.
USB mode and USB4 modes:
[3] [2] [1] [0] Description
0 0 0 0 P0, normal operation
0 0 0 1 P1, low recovery time latency, power saving state
0 0 1 0 P2, longer recovery time latency, lower power state
0 0 1 1 P3, lowest power state
0 1 0 0 POWER_STATE_4 PHY-specific
0 1 0 1 POWER_STATE_5 PHY-specific
0 1 1 0 POWER_STATE_6 PHY-specific
0 1 1 1 POWER_STATE_7 PHY-specific
1 0 0 0 POWER_STATE_8 PHY-specific
1 0 0 1 POWER_STATE_9 PHY-specific
1 0 1 0 POWER_STATE_10 PHY-specific
1 0 1 1 POWER_STATE_11 PHY-specific
1 1 0 0 POWER_STATE_12 PHY-specific
1 1 0 1 POWER_STATE_13 PHY-specific
1 1 1 0 POWER_STATE_14 PHY-specific
1 1 1 1 POWER_STATE_15 PHY-specific
When transitioning from P3 to P0, the signaling is asynchronous (since PCLK is not running).

For USB, the following are the characteristics of the power states that must be minimally implemented:
<table>
<tr><th>PowerDown</th><th>PCLK State</th><th>Tx common mode state</th><th>Operations</th></tr>
<tr><td>P0</td><td>On</td><td>On</td><td>Transmit/Receive high speed data
Transmit/Receive LFPS
Termination control</td></tr>
<tr><td>P1</td><td>On</td><td>On</td><td>Transmit/Receive LFPS
Termination control</td></tr>
<tr><td>P2</td><td>On</td><td>Off</td><td>Receive LFPS
Termination control
Remote receiver detection</td></tr>
<tr><td>P3</td><td>Off</td><td>Off</td><td>Off
Receive LFPS
Termination control
Remote receiver detection</td></tr>
</table>
If PowerDown changes during the IORecal, RxEqEval, or RxEqTraining operations, the PHY must abort the request and return the handshake acknowledgment.

For USB4, refer to Section 8.3.3 for characteristics of power states that must be minimally implemented.
</td><td>PCIe, USB, and USB4</td></tr>
</table>

Table 6-5. Command Interface Input Signals (Sheet 5 of 13)

<table>
<tr><th>Name</th><th>Active Level</th><th>Description</th><th>Relevant Protocols</th></tr>
<tr><td>PowerDown[3:0]
SATA Mode</td><td>N/A</td><td>SATA mode:
Power up or down the transceiver. Power states
[3] [2] [1] [0] Description
0 0 0 0 POWER_STATE_0 Operational state
0 0 0 1 POWER_STATE_1 Phy specific
0 0 1 0 POWER_STATE_2 Phy specific
0 0 1 1 POWER_STATE_3 Phy specific
0 1 0 0 POWER_STATE_4 Phy specific
0 1 0 1 POWER_STATE_5 Phy specific
0 1 1 0 POWER_STATE_6 Phy specific
0 1 1 1 POWER_STATE_7 Phy specific
1 0 0 0 POWER_STATE_8 Phy specific
1 0 0 1 POWER_STATE_9 Phy specific
1 0 1 0 POWER_STATE_10 Phy specific
1 0 1 1 POWER_STATE_11 Phy specific
1 1 0 0 POWER_STATE_12 Phy specific
1 1 0 1 POWER_STATE_13 Phy specific
1 1 1 0 POWER_STATE_14 Phy specific
1 1 1 1 POWER_STATE_15 Phy specific
A PIPE compliant SATA PHY is recommended to support at least 4 states other than POWER_STATE_0. There must be at least one power state meeting each of the requirements shown in the following table
<table>
<tr><th>PCLK State</th><th>TX Common Mode State</th><th>Exit Latency to POWER_STATE_0</th></tr>
<tr><td>Off</td><td>Off</td><td>< 10 ms</td></tr>
<tr><td>Off</td><td>On</td><td>< 10 us</td></tr>
<tr><td>On</td><td>On</td><td>< 10 us</td></tr>
<tr><td>On</td><td>Off</td><td>< 300 us</td></tr>
</table>
Exit latency to POWER_STATE_0 is measured from when the MAC changes the Power down value to when the PHY deasserts PHY status. The actual PHY latency must provide enough margin from the indicated limits to enable compliant device behavior per the SATA specification. A MAC must map the available PHY states to SATA states.

Note: PLL shutdown is only possible if PowerDown is set to a state with PCLK off.</td><td>SATA</td></tr>
</table>

Table 6-5. Command Interface Input Signals (Sheet 6 of 13)

<table>
<tr><th>Name</th><th>Active Level</th><th>Description</th><th>Relevant Protocols</th></tr>
<tr><td>PowerDown[3:0]
DisplayPort mode</td><td>N/A</td><td>
DisplayPort mode power states:

[3] [2] [1] [0] Description
0 0 0 0 POWER_STATE_0 operational state
0 0 0 1 POWER_STATE_1 PHY-specific
0 0 1 0 POWER_STATE_2 PHY-specific
0 0 1 1 POWER_STATE_3 PHY-specific
0 1 0 0 POWER_STATE_4 PHY-specific
0 1 0 1 POWER_STATE_5 PHY-specific
0 1 1 0 POWER_STATE_6 PHY-specific
0 1 1 1 POWER_STATE_7 PHY-specific
1 0 0 0 POWER_STATE_8 PHY-specific
1 0 0 1 POWER_STATE_9 PHY-specific
1 0 1 0 POWER_STATE_10 PHY-specific
1 0 1 1 POWER_STATE_11 PHY-specific
1 1 0 0 POWER_STATE_12 PHY-specific
1 1 0 1 POWER_STATE_13 PHY-specific
1 1 1 0 POWER_STATE_14 PHY-specific
1 1 1 1 POWER_STATE_15 PHY-specific
A PIPE-compliant DPRx PHY is recommended to support the following power states, although the mapping to the above power state encodings is PHY implementation specific:
<table>
<tr><th>Main Link Rx</th><th>Aux Link</th><th>Exit Latency</th><th>Required</th></tr>
<tr><td>Enabled</td><td>Enabled</td><td>N/A</td><td>Yes</td></tr>
<tr><td>Disabled</td><td>Enabled for differential signal monitoring</td><td><1 ms</td><td>Yes</td></tr>
<tr><td>Disabled</td><td>Enabled for differential signal monitoring</td><td><80 ms</td><td>No</td></tr>
<tr><td>Disabled</td><td>Enabled</td><td><0.5</td><td>Yes for eDP only</td></tr>
<tr><td>Disabled</td><td>Enabled</td><td><20 us</td><td>Yes for eDP only</td></tr>
</table>
A PIPE compliant DPTX PHY is recommended to support the following power states, although the mapping to specific power state encodings is PHY implementation specific:
<table>
<tr><th>Main Link TX</th><th>Aux Link</th><th>DP_PWR</th></tr>
<tr><td>Enabled</td><td>Enabled</td><td>Enabled</td></tr>
<tr><td>Disabled</td><td>Enabled</td><td>Enabled</td></tr>
</table>
If PowerDown changes during IORecal, RxEqEval, or RxEqTraining operations, the PHY must abort the request and return the handshake acknowledgment.
</td><td>DisplayPort</td></tr>
</table>

Table 6-5. Command Interface Input Signals (Sheet 7 of 13)

<table>
<tr><th>Name</th><th>Active Level</th><th>Description</th><th>Relevant Protocols</th></tr>
<tr><td>ShortChannelPowerControl[1:0]</td><td>N/A</td><td>This signal is optionally supported by the PHYs to enable multi-chip package solutions that can be optimized for power due to a shorter channel reach. This signal must be stable before Reset# is deasserted and cannot change value until the next Reset# assertion. Possible settings (see the PHY parameter “ShortChannelPowerControlSettingsSupported” for details associated with each setting):
PowerControlSetting0 (00b): Normal operation
PowerControlSetting1 (01b): The most aggressive power-optimized setting for MCP (required if the ShortChannelPowerControl[1:0] signal is implemented)
PowerControlSetting2 (10b): Optionally supported
PowerControlSetting3 (11b): Optionally supported
Note: For all ShortChannelPowerControl[1:0] settings that correspond to an MCP solution, it is strongly recommended to bypass receiver detection as described earlier in Section 2.7.</td><td>PCIe</td></tr>
<tr><td>RxEIDetectDisable</td><td>High</td><td>PCIe:

Optionally implemented to facilitate PCIe L1 substate management. When asserted, this signal asynchronously disables the receiver electrical idle detect logic, forcing the RxElecIdle PHY output to a value of “1”. If this signal transitions to deasserted after being asserted, the RxElecIdle output must be forced to a high value until the electrical idle detect logic is functional.

The PHY may choose to support managing L1 substates via this signal and the TxCommonmodeDisable signal instead of the PowerDown[3:0] signal. In this case, the PowerDown[3:0] signal must be held at a constant value through the L1 substate transitions.

In addition to the legacy states, the following are the minimum combinations required to be implemented by designs that support L1 substates:
<table>
<tr><th>PCLK State</th><th>TX Common Mode State</th><th>RxElecIdle Supported</th><th>When to return PhyStatus when exiting?</th><th>Exit Latency to P0</th></tr>
<tr><td>Off</td><td>TxCommonModeDisable = ‘1’</td><td>RxEIDetectDisable=’1’</td><td>Before transmit common mode established</td><td>Implementation Specific</td></tr>
<tr><td>Off</td><td>TxCommonModeDisable=’0’</td><td>RxEIDetectDisable=’1’</td><td>N/A</td><td>Implementation Specific</td></tr>
</table>
RxEIDetectDisable may be optionally implemented by the PHY to allow the MAC to disable receiver Electrical Idle detect logic to provide power savings in states in addition to L1. A PHY must advertise in its datasheet which PowerDown states it supports use of RxEIDetectDisable, via the parameter RxEIDetectDisableSupportedStates.
USB/USB4:
This signal may be optionally implemented by the PHY to allow the MAC to disable the LFPS circuit to provide power savings. The PHY datasheet specifies whether this usage is available. This signals is asynchronous for USB/USB4.</td><td>PCIe, USB, USB4</td></tr>
<tr><td>RxEIDetectDisable2</td><td>High</td><td>USB4: This signal is used in asymmetric mode and corresponds to the Rx2 differential pair. Refer to RxEIDetectDisable for more details.</td><td>USB4</td></tr>
</table>

Table 6-5. Command Interface Input Signals (Sheet 8 of 13)

Name	Active Level	Description	Relevant Protocols
TxCommonmode Disable	High	Optionally implemented by the PHY to facilitate the L1 substate management. When asserted, this signal asynchronously disables the transmitter DC common mode logic. **Note:** The PHY may choose to support managing L1 substates via this signal and the RxEIDetectDisable signal instead of the PowerDown[3:0] signal. This signal is only valid when PowerDown is at a value that supports L1 substate management via TxCommonmodeDisable and RxEIDetectDisable. This signal is only used by PHYs that support PCIe L1 substates.	PCIe
DeepPMReq#	Low	Deep Power Management Request This is an asynchronous signal that is optionally implemented by the PHY to enable deep power management for maximum power savings while in non-P0 PowerDown states. The MAC is not permitted to assert this signal while in the P0 PowerDown state. A value of "0" indicates that the PHY is permitted to enter a deep power management state. A value of "1" indicates that the PHY must exit any deep power management state to enable the shortest possible exit latency associated with the current PowerDown state. The default value of this signal out of reset must be "0". DeepPMReq# is an asynchronous signal and it requires a full handshake with DeepPMAck#. See Section 8.3.4 for more details.	PCIe, SATA, USB, USB4, DisplayPort
Restore#	Low	Restore Window This is an asynchronous signal optionally implemented by the PHY to receive indication that it is in a restore window, during which context previously saved off outside of the PHY is being restored. A value of "0" indicates that the PHY is in a restore window. A value of "1" indicates that PHY is not in a restore window. **Note:** Both "0" and "1" are valid values when coming out of reset. See Section 8.3.4 for more details.	PCIe, SATA, USB, USB4, and DisplayPort

Table 6-5. Command Interface Input Signals (Sheet 9 of 13)

<table>
<tr><th>Name</th><th>Active Level</th><th colspan="2">Description</th><th>Relevant Protocols</th></tr>
<tr><td rowspan="30">Rate[3:0]</td><td rowspan="30">N/A</td><td colspan="2">Control the link signaling rate.
PCIe mode:</td><td rowspan="30">PCIe, SATA, USB, DisplayPort, and USB4</td></tr>
<tr><td>Value</td><td>Description</td></tr>
<tr><td>0</td><td>Use 2.5 GT/s signaling rate</td></tr>
<tr><td>1</td><td>Use 5.0 GT/s signaling rate</td></tr>
<tr><td>2</td><td>Use 8.0 GT/s signaling rate</td></tr>
<tr><td>3</td><td>Use 16.0 GT/s signaling rate.</td></tr>
<tr><td>4</td><td>Use 32.0 GT/s signaling rate</td></tr>
<tr><td>5</td><td>Use 64 GT/s signaling rate</td></tr>
<tr><td>6</td><td>Use 128 GT/s signaling rate</td></tr>
<tr><td>7 thru 15</td><td>Reserved</td></tr>
<tr><td colspan="2">SATA mode:</td></tr>
<tr><td>Value</td><td>Description</td></tr>
<tr><td>0</td><td>Use 1.5 GT/s signaling rate</td></tr>
<tr><td>1</td><td>Use 3.0 GT/s signaling rate</td></tr>
<tr><td>2</td><td>Use 6.0 GT/s signaling rate</td></tr>
<tr><td>3 thru 15</td><td>Reserved</td></tr>
<tr><td colspan="2">USB mode:</td></tr>
<tr><td>Value</td><td>Description</td></tr>
<tr><td>0</td><td>Use 5.0 GT/s signaling rate</td></tr>
<tr><td>1</td><td>Use 10.0 GT/s signaling rate</td></tr>
<tr><td>2 thru 15</td><td>Reserved</td></tr>
<tr><td colspan="2">DisplayPort mode:</td></tr>
<tr><td>Value</td><td>Description</td></tr>
<tr><td>0</td><td>Use 1.62 Gbps signaling rate</td></tr>
<tr><td>1</td><td>Use 2.7 Gbps signaling rate</td></tr>
<tr><td>2</td><td>Use 5.4 Gbps signaling rate</td></tr>
<tr><td>3</td><td>Use 8.1 Gbps signaling rate</td></tr>
<tr><td>4</td><td>Use 2.16 Gbps signaling rate</td></tr>
<tr><td>5</td><td>Use 2.43 Gbps signaling rate</td></tr>
</table>

Table 6-5. Command Interface Input Signals (Sheet 10 of 13)

<table>
<tr><th>Name</th><th>Active Level</th><th colspan="2">Description</th><th>Relevant Protocols</th></tr>
<tr><td rowspan="14"></td><td rowspan="14"></td><td>Value</td><td>Description</td><td rowspan="14">PCIe, SATA, USB, DisplayPort, and USB4</td></tr>
<tr><td>6</td><td>Use 3.24 Gbps signaling rate</td></tr>
<tr><td>7</td><td>Use 4.32 Gbps signaling rate</td></tr>
<tr><td>8</td><td>Use 10 Gbps signaling rate</td></tr>
<tr><td>9</td><td>Use 13.5 Gbps signaling rate</td></tr>
<tr><td>10</td><td>Use 20 Gbps signaling rate</td></tr>
<tr><td>11 thru 14</td><td>Reserved</td></tr>
<tr><td>15</td><td>Rate Override mode: The rate is specified via a vendor-specific PHY Rate register, which is outside the scope of this specification.</td></tr>
<tr><td colspan="2">USB4 mode:</td></tr>
<tr><td>Value</td><td>Description</td></tr>
<tr><td>0</td><td>Use 10.0 GT/s signaling rate</td></tr>
<tr><td>1</td><td>Use 20.0 GT/s signaling rate</td></tr>
<tr><td>2</td><td>Use 40.0 GT/s signaling rate</td></tr>
<tr><td>3 thru 13</td><td>Reserved</td></tr>
<tr><td>14</td><td>Use 10.3125 GT/s signaling</td></tr>
<tr><td>15</td><td>Use 20.625 GT/s signaling</td></tr>
<tr><td colspan="2">PIPE implementations that only support one signaling rate do not implement this signal.</td></tr>
<tr><td rowspan="16">Width[2:0]</td><td rowspan="16">N/A</td><td colspan="2">Controls the PIPE data path width. For SerDes architecture, this applies only to the transmit side and RxWidth[1:0] controls the receive side.
If EncodeDecodeBypass is “0”</td><td rowspan="16">PCIe, SATA, USB,
USB4
DisplayPort TX</td></tr>
<tr><td>Value</td><td>Datapath Width</td></tr>
<tr><td>0</td><td>8 bits</td></tr>
<tr><td>1</td><td>16 bits</td></tr>
<tr><td>2</td><td>32 bits</td></tr>
<tr><td>3-7</td><td>Reserved</td></tr>
<tr><td colspan="2">If EncodeDecodeBypass is “1” or in SerDes architecture</td></tr>
<tr><td>Value</td><td>Datapath Width</td></tr>
<tr><td>0</td><td>10 bits</td></tr>
<tr><td>1</td><td>20 bits</td></tr>
<tr><td>2</td><td>40 bits</td></tr>
<tr><td>3</td><td>80 bits (PCIe SerDes only)/56 bits (USB4 only)</td></tr>
<tr><td>4</td><td>160bits (PCIe SerDes only)</td></tr>
<tr><td>5-7</td><td>Reserved</td></tr>
<tr><td colspan="2">Note: PHYs that support greater than x4 link width must provide option of 32-bit data width or smaller.

PIPE implementations that only support one option at each signaling rate do not implement this signal</td></tr>
</table>

Table 6-5. Command Interface Input Signals (Sheet 11 of 13)

Name	Active Level	Description	Relevant Protocols
PCLK Rate[4:0]	N/A	Control the PIPE PCLK rate SATA Mode: 0 37.5 Mhz 1 75 Mhz 2 150 Mhz 3 300 Mhz 4 600 Mhz All others Reserved PCIe Mode: 0 62.5 Mhz 1 125 Mhz 2 250 Mhz 3 500 Mhz 4 1000 Mhz 5 2000 Mhz 6 4000 Mhz All others Reserved USB Mode: 0 125 Mhz 1 250 Mhz 2 312.5 Mhz (10 GT/s) 3 500 Mhz 4 625 Mhz (10 GT/s) 5 1250 Mhz (10 GT/s) All others Reserved	PCIe, SATA, USB, and DisplayPort

Table 6-5. Command Interface Input Signals (Sheet 12 of 13)

<table>
<tr><th>Name</th><th>Active Level</th><th>Description</th><th>Relevant Protocols</th></tr>
<tr><td></td><td></td><td>DisplayPort mode:
<table><tr><th>Value</th><th>Rate</th></tr><tr><td>0</td><td>40.5 Mhz</td></tr><tr><td>1</td><td>67.5 Mhz</td></tr><tr><td>2</td><td>81 Mhz</td></tr><tr><td>3</td><td>135 Mhz</td></tr><tr><td>4</td><td>162 Mhz</td></tr><tr><td>5</td><td>202.5 Mhz</td></tr><tr><td>6</td><td>270 Mhz</td></tr><tr><td>7</td><td>405 Mhz</td></tr><tr><td>8</td><td>540 Mhz</td></tr><tr><td>9</td><td>810 Mhz</td></tr><tr><td>10</td><td>54 Mhz</td></tr><tr><td>11</td><td>60.75 Mhz</td></tr><tr><td>12</td><td>108 Mhz</td></tr><tr><td>13</td><td>121.5 Mhz</td></tr><tr><td>14</td><td>160 Mhz</td></tr><tr><td>15</td><td>216 Mhz</td></tr><tr><td>16</td><td>243 Mhz</td></tr><tr><td>17</td><td>324 Mhz</td></tr><tr><td>18</td><td>432 Mhz</td></tr><tr><td>19</td><td>312.5 Mhz</td></tr><tr><td>20</td><td>421.875 Mhz</td></tr><tr><td>21</td><td>625 Mhz</td></tr><tr><td>All others</td><td>Reserved</td></tr></table>USB4 mode: This signal is not used.
PIPE implementations that do not support more than one PCLK rate for any analog signaling rate do not implement this signal.</td><td>PCIe, SATA, USB, and DisplayPort</td></tr>
<tr><td>RXTermination</td><td>High</td><td>Controls presence of receiver terminations:
<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>Terminations removed</td></tr><tr><td>1</td><td>Terminations present</td></tr></table></td><td>USB, PCIe, DisplayPort RX, USB4</td></tr>
</table>

Table 6-5. Command Interface Input Signals (Sheet 13 of 13)

Name	Active Level	Description	Relevant Protocols
RxStandby	High	SATA Mode: Controls whether the PHY RX is active when the PHY is in any power state with PCLK on. 0 – Active 1 – Standby RxStandby is ignored when the PHY is in any power state where the high speed receiver is always off. PCIe Mode: Controls whether the PHY RX is active when the PHY is in P0 or P0s. 0 – Active 1 – Standby RxStandby is ignored when the PHY is in states other than P0 or P0s. If RxStandby changes during IORecal, RxEqEval, or RxEqTraining operations, the PHY must abort the request and return the handshake acknowledgment. USB Mode and USB4 Mode: Controls whether the PHY RX is active when the PHY is in any power state with PCLK on. 0 – Active 1 – Standby RxStandby is ignored when the PHY is in any power state where the high speed receiver is always off. If RxStandby changes during IORecal, RxEqEval, or RxEqTraining operations, the PHY must abort the request and return the handshake acknowledgment.	PCIe, SATA, USB, USB4
RxStandby2	High	This signal corresponds to the Rx2 differential pair	USB4

Table 6-6. Command Interface Output Signals (Sheet 1 of 2)

Name	Active Level	Description	Relevant Protocols
RefClkRequired #	Low	This signal is deasserted by the PHY when the reference clock can be safely removed in low power states. This signal must remain asserted low in all states except P2, P1 and PowerDown states assigned to L1 substate support. While in P2, P1, or L1 substate PowerDown states, the PHY deasserts this signal when it is ready for reference clock removal. While in P2 or P1 or L1 substate PowerDown states, the PHY asserts this signal when it detects a P2 or P1 or L1 substate exit request. This signal is optionally implemented by the PHY. The MAC is required to prevent CLKREQ# from being deasserted if this signal is asserted.	PCIe

Table 6-6. Command Interface Output Signals (Sheet 2 of 2)

Name	Active Level	Description	Relevant Protocols
RxStandbyStatus	High	SATA mode and PCIe mode and USB4 mode: The PHY uses this signal to indicate its RxStandby state. 0: Active 1: Standby RxStandbyStatus reflects the state of the high-speed receiver. The high-speed receiver is always off in the PHY states that do not provide PCLK. PCIe mode: RxStandbyStatus is undefined when the power state is P1 or P2. This signal is not applicable to USB mode.	PCIe, SATA, and USB4
RxStandbyStatus2	High	This corresponds to the Rx2 differential pair. Refer to RxStandbyStatus for more details.	USB4

6.1.3 Status Interface

Table 6-7. Status Interface Input Signals

Name	Active Level	Description	Relevant Protocols
PclkChangeAck	High	Only used when PCLK is a PHY input. Asserted by the MAC when a PCLK rate change or rate change or, if required, width change is complete and stable. After the MAC asserts PclkChangeAck the PHY responds by asserting PhyStatus for one cycle and de-asserts PclkChangeOk at the same time as PhyStatus. The controller shall deassert PclkChangeAck when PclkChangeOk is sampled low. This signal is not used by any Phymode that does not perform dynamic rate changes.	PCIe, SATA, and USB
AsyncPowerChangeAck	High	Only used when transitioning between two power states without PCLK. After the PHY asserts PhyStatus to acknowledge the power state change the MAC responds by asserting AsyncPowerChangeAck until it samples the PhyStatus deasserted. Implementation of this signal is only required for PHYs that support PCIe L1 PM substates managed via the PowerDown signal.	PCIe

Table 6-8. Status Interface Output Signals (Sheet 1 of 3)

Name	Active Level	Description	Relevant Protocols
RxValid	High	Indicates symbol lock and valid data on *RxData* and *RxDataK* and further qualifies RxDataValid when used. In the original PIPE architecture: PCIe mode at 8 GT/s, 16 GT/s, and 32 GT/s and USB mode at 10 GT/s only: When BlockAlignControl=1: • RxValid indicates that the block aligner is conceptually in the "Aligned" state (see the PCIe or USB 3.2 spefications) • If the block aligner transitions from "Aligned" to the "Unaligned" state RxValid can deassert anywhere within a block • If the block aligner transitions from "Unaligned" to the "Aligned" state RxValid is asserted at the start of a block **Note:** A PHY is not required to force its block aligner to the unaligned state when BlockAlignControl transitions to one. When BlockAlignControl=0: • RxValid is constantly high indicating that the block aligner is conceptually in the "Locked" state (see the PCIe or USB 3.2 specifications). RxValid can be dropped on detecting and elastic buffer underflow or overflow. If deasserted, it must not reassert while the **BlockAlignControl** is deasserted. In the SerDes architecture, RxValid is used to indicate that the recovered clock is stable, that is, a value of "1" indicates that RxCLK is stable and a value of "0" indicates that RxCLK is not stable. The MAC can start symbol or block lock after RxValid is asserted. This signal is synchronous to RxCLK in SerDes mode.	PCIe, USB, USB4, SATA, and DisplayPort Rx
RxValid2	High	RxValid2 is used to indicate that the recovered clock is stable, that is, a value of "1" indicates that RxCLK2 is stable and a value of "0" indicates that RxCLK2 is not stable. The MAC can start symbol or block lock after RxValid2 is asserted. This signal is synchronous to RxCLK2.	DisplayPort Rx, USB4
PhyStatus	High	Used to communicate the completion of several PHY functions including stable PCLK and Max PCLK (depending on the clocking mode) after Reset# deassertion, power management state transitions, rate change, and receiver detection. When this signal transitions during entry and exit from any PHY state where PCLK is not provided, then the signaling is asynchronous. In error situations (where the PHY fails to assert PhyStatus) the MAC can take MAC-specific error recovery actions.	PCIe, SATA, USB, DisplayPort, and USB4
RxElecIdle	High	Indicates receiver detection of an electrical idle. While deasserted with the PHY in P2 (PCIe mode) or the PHY in P0, P1, P2, or P3 (USB mode and USB4 mode), indicates detection of either: PCIe mode: A beacon USB and USB4 modes: LFPS This is an asynchronous signal. See RxEIDetectDisable for additional information. PCIe mode: It is required at the 5.0 GT/s, 8.0 GT/s, 16 GT/s, 32 GT/s, 64 GT/s,and 128 GT/s rates that a MAC uses logic to detect electrical idle entry instead of relying on the RxElecIdle signal. SATA mode: The time the signal is asserted must match the actual idle time on the analog bus within -16/+0 ns.	PCIe, SATA, USB, and USB4

Table 6-8. Status Interface Output Signals (Sheet 2 of 3)

Name	Active Level	Description	Relevant Protocols
RxElecIdle2	High	This corresponds to the Rx2 differential pair. See RxElecIdle2 for details.	USB4
RxStatus[2:0]	N/A	Encodes receiver status and error codes for the received data stream when receiving data.	PCIe, SATA, and USB

2	1	0	Description
0	0	0	Received data OK
0	0	1	PCIe mode: 1 SKP added USB mode: 1 SKP ordered set added SATA mode: 1 ALIGN added Asserted with the first byte of ALIGN that was added. An align may only be added in conjunction with receiving one or more aligns in the data stream and only when the elasticity buffer is operating in half full mode.
0	1	0	PCIe mode: 1 SKP removed USB mode: 1 SKP ordered set removed SATA mode: 1 or more ALIGNs removed This status is asserted with first non-ALIGN byte following an ALIGN. This status message is applicable to both EB buffer modes.
0	1	1	PCIe and USB modes: Receiver detected SATA mode: Misalign Signaled on the first symbol of an ALIGN that was received misaligned in elasticity buffer nominal half full mode. Signaled on the first data following an align in elasticity buffer nominal empty mode.
1	0	0	Both 8B/10B (128B/130B[1]) decode error and (optionally) the receive disparity error. This error is never reported if EncodeDecodeBypass is asserted.
1	0	1	Elastic Buffer Overflow
1	1	0	Elastic buffer underflow: This error code is not used if the elasticity buffer is operating in the nominal buffer empty mode.
1	1	1	Receive disparity error (Reserved if Receive Disparity error is reported with code 0b100) Not used if EncodeDecodeBypass is asserted. For USB3 Gen2, it indicates "SKP Corrected".

Note: The only status applicable to the SerDes architecture is "Receiver detected" (0x3).

PowerPresent	High	USB mode: It indicates the presence of VBUS. Implementation of this signal is only required for PHYs that support the USB mode.	USB
PclkChangeOk	High	Only used when PCLK is a PHY input. Asserted by the PHY when it is ready for the MAC to change the PCLK rate or Rate or, if required, width. The PHY shall only assert this signal after the MAC has requested a PCLK rate change by changing PCLK_Rate or rate change by changing Rate or, if required, a width change by changing Width. This signal is not used for DisplayPort or USB4 mode.	PCIe, SATA, and USB

Table 6-8. Status Interface Output Signals (Sheet 3 of 3)

Name	Active Level	Description	Relevant Protocols
DeepPMAck#	Low	Deep Power Management Ack. This is an asynchronous signal used to acknowledge transitions on DeepPMReq#. This signal is only required on a PHY that implements DeepPMReq#. A value of "0" indicates that PHY has acknowledged a request on DeepPMReq# to enter a deep power management state; however, it does not indicate that the PHY has actually entered a deep power management state. A value of "1" indicates that the PHY acknowledged a request on DeepPMReq# to exit a deep power management state. The PHY must first exit any deep power management state before returning this handshake. See Section 8.3.4 for more details.	PCIe, SATA, USB, USB4, and DisplayPort

1. Disparity errors are not reported when the rate is 8.0 GT/s, 16 GT/s, or 32 GT/s.

6.1.4 Message Bus Interface

The message bus interface provides a way to initiate and participate in non-latency sensitive PIPE operations using a small number of wires, it also enables future PIPE operations to be added without adding additional wires. The use of this interface requires the device to be in a power state with PCLK running. Control and status bits used for PIPE operations are mapped into 8-bit registers that are hosted in 12-bit address spaces in the PHY and the MAC. The registers are accessed via read and write commands driven over the signals listed in Table 6-9. These signals are synchronous with the PCLK and are reset with Reset#. The specific commands and framing of the transactions sent over the message bus interface are described in the following subsections.

Table 6-9. Message Bus Interface Signals

Name	Direction	Description
M2P_MessageBus[7:0]	Input	The MAC multiplexes command, any required address, and any required data for sending read and write requests to access the PHY PIPE registers and for sending read completion responses and write ack responses to PHY initiated requests.
P2M_MessageBus[7:0]	Output	The PHY multiplexes command, any required address, and any required data for sending read and write requests to access MAC PIPE registers and for sending read completion responses and write ack responses to MAC initiated requests.

Errors in SKP ordered sets must be reported by the PHY as 128/130 decode errors. An error in an SKP ordered set must be reported if there is an error in the first 4N+1 symbols of the skip ordered set.

6.1.4.1 Message Bus Interface Commands

The 4-bit commands used for accessing the PIPE registers across the message bus are defined in Table 6-10. A transaction consists of a command and any associated address and data, as specified in the table. The table also specifies the number of PCLK cycles

that it takes to transfer the transaction across the message bus interface. The order in which the bits are transferred across the interface are illustrated in Table 6-11, Table 6-12, Table 6-13, and Table 6-14.

To address the case where multiple PIPE interface signals can change on the same PCLK, the concept of write_uncommitted and write_committed is introduced. A series of write_uncommitted transactions followed by one write_committed transaction provides a mechanism by which all the uncommitted writes and the final committed write are executed in an atomic manner, taking effect during the same PCLK cycle.

To enable the write_uncommitted command, designs must implement a write buffer in the PHY and the MAC, where each write buffer entry can accommodate the three bytes worth of information associated with each write transaction. The minimum write buffer depth required is five, however, this number may increase in the future when new PIPE operations are mapped into the message bus interface.

Table 6-10. Message Bus Commands

Encoding	Command	Description	Required Fields	Cycles to Transmit
4'b0000	NOP	Idle. See Table 6-11.	Command[3:0]	1
4'b0001	write_uncommitted	The current write should be saved off into a write buffer and its associated data values are updated into the relevant PIPE register at a future time when a write_committed is received. This is useful for signals that must change in the same cycle but that are distributed among multiple registers. See Table 6-14.	Command[3:0], Address[11:0], Data[7:0]	3
4'b0010	write_committed	The current write, as well as any previously uncommitted writes saved into the write buffer, should be committed, that is, their values should be updated into the PIPE registers. Once a write_committed is sent, no new writes, whether committed or uncommitted, may be sent until a write_ack is received. See Table 6-14.	Command[3:0], Address[11:0], Data[7:0]	3
4'b0011	read	Used to read contents of a PIPE register. Only one read can be outstanding at a time in each direction. See Table 6-12.	Command[3:0], Address[11:0]	2
4 "b0100	read completion	Data response to a read. See Table 6-13.	Command[3:0], Data[7:0]	2
4'b0101	write_ack	Used to acknowledge receipt of a write_committed and readiness to accept another write. The Ack is sent when the write buffer is flushed and the resulting PIPE operation is guaranteed to start in a deterministic amount of time. **Note:** This does not provide confirmation that the PIPE operation triggered by the write has completed. See Table 6-11.	Command[3:0]	1
All others	Reserved	N/A	N/A	N/A

Table 6-11. Command Only Message Bus Transaction Timing (NOP, write_ack)

		M2P/P2M_MessageBus							
Time		7	6	5	4	3	2	1	0
	t	Cmd[3:0]				0000b			

Table 6-12. Command+Address Message Bus Transaction Timing (Read)

		M2P/P2M_MessageBus							
		7	6	5	4	3	2	1	0
Time	t	Cmd[3:0]				Addr[11:8]			
	t+1	Addr[7:0]							

Table 6-13. Command+Data Message Bus Transaction Timing (Read Completion)

		M2P/P2M_MessageBus							
		7	6	5	4	3	2	1	0
Time	t	Cmd[3:0]				0000b			
	t+1	Data[7:0]							

Table 6-14. Command+Address+Data Message Bus Transaction Timing (Write_uncommitted, Write_committed)

		M2P/P2M_MessageBus							
		7	6	5	4	3	2	1	0
Time	t	Cmd[3:0]				Addr[11:8]			
	t+1	Addr[7:0]							
	t+2	Data[7:0]							

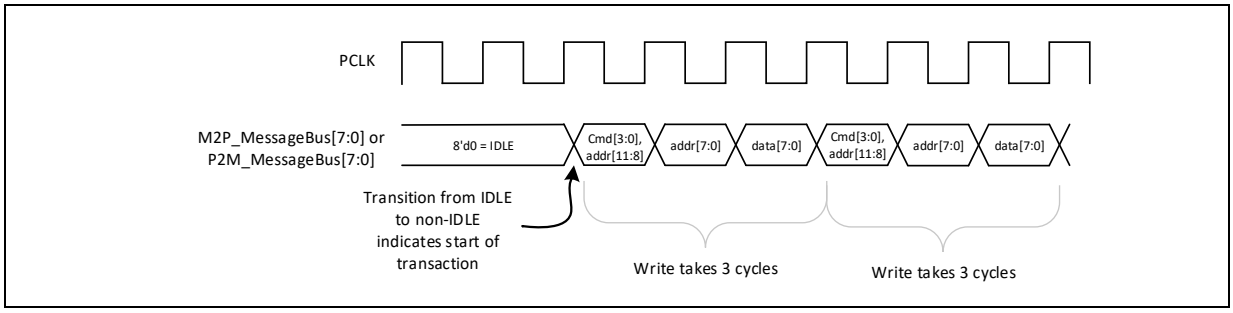
6.1.4.2 Message Bus Interface Framing

The framing of transactions is implicitly derived by adhering to the following rules:

1. All zeroes must be driven on the message bus when idle.
2. An idle to a non-idle transition indicates the start of a transaction; a new transaction can immediately start the cycle after the end of the previous transaction without an intervening idle.
3. The number of cycles to transmit a transaction depends on the command and is specified in Table 6-10.
4. The cycles associated with one transaction must be transferred in contiguous cycles.

Figure 6-1 illustrates the framing of a couple of transactions on the message bus. The start of the first transaction is inferred by the idle to a non-idle transition. The command is decoded as a write, which takes three cycles to transmit. Since the cycle following the end of the write is non-idle, it is inferred to be the start of the next transaction, which is decoded to be another write that takes three cycles to transmit.

Figure 6-1. Message Bus Transaction Framing



6.2 PHY/MAC Interface Signals – SerDes Architecture Only

This section describes any signals for SerDes architecture that are required in addition to those defined in Section 6.1.

6.2.1 Data Interface

Table 6-15. SerDes Only: Rx data Interface Output Signals

Name	Active Level	Description	Relevant Protocols
RxCLK	Rising Edge	**This clock signal is only used in the SerDes architecture.** Recovered clock used for RxData in the SerDes architecture. When RxValid deasserts, the PHY must keep the RxCLK running for at least eight clocks to enable the MAC to latch the deassertion of RxValid. The PHY must advertise a maximum RxCLK value in its datasheet (MaxRxClkFrequency) that it guarantees not to exceed even when RxValid is deasserted.	PCIe, USB, DisplayPort Rx, and USB4
RxCLK2	Rising Edge	**This clock signal is only used in the SerDes architecture.** Recovered clock used for RxData2 in the SerDes architecture.	DisplayPort Rx

6.2.2 Command Interface

Table 6-16. SerDes Only: Command Interface Input Signals

<table>
<tr><th>Name</th><th>Active Level</th><th colspan="2">Description</th><th>Relevant Protocols</th></tr>
<tr><td rowspan="9">RxWidth[2:0]</td><td rowspan="9">N/A</td><td colspan="2">This signal is only used in the SerDes architecture.

It controls the PIPE receive data path width.</td><td rowspan="9">PCIe, SATA, USB, USB4, and DisplayPort Rx</td></tr>
<tr><td>Value</td><td>Datapath Width</td></tr>
<tr><td>0</td><td>10 bits</td></tr>
<tr><td>1</td><td>20 bits</td></tr>
<tr><td>2</td><td>40 bits</td></tr>
<tr><td>3</td><td>80 bits (PCIe SerDes only)/56-bits USB4</td></tr>
<tr><td>4</td><td>160 bits</td></tr>
<tr><td>5-7</td><td>Reserved</td></tr>
<tr><td colspan="2">Note: PHYs that support greater than x4 link width must provide option of 32-bit data width or smaller.

PIPE implementations that only support one option at each signaling rate do not implement this signal.</td></tr>
</table>

6.2.3 MacCLK Lane Signals

A MacCLK lane is an optional feature that is implemented only by PHYs that support MacCLK. The signals defined here are per MacCLK lane. These signals are independent of the other PIPE signals. Please refer to Section 8.1.2 for more details. The MacCLK lane signals are in the MacCLKReset# domain. The MAC and the PHY must not rely on the signals being held at a valid value when MacCLKReset# is asserted. If default values are specified, the MAC and the PHY must guarantee that the signals they drive are stable and at their reset values when MacCLKReset# deasserts. The PHY specifies the minimum MacCLKReset# assertion pulse duration it requires via the paramater MinimumMacCLKReset#AssertionPulse.

Table 6-17. MacCLK Lane Input Signals

Name	Active Level	Description	Relevant Protocols
MacCLKPHYMode [3:0]	N/A	Selects the protocol associated with this MacCLK lane: 0 - PCIe 1 - USB 2 - SATA 3 - DisplayPort 4 - Reserved 5 - Reserved 6 - Reserved 7 - USB4 All other reserved. This signal is asynchronous and is permitted to change only during MacCLKReset# assertion. The MAC must ensure this signal is valid at MacCLKReset# deassertion.	PCIe, SATA, USB, DisplayPort, USB4
MacCLKRate[4:0]	N/A	Selects the rate of the MacCLK. This uses the following encoding definitions: 0-30: Uses the same encoding definitions a PCLK Rate. 31: Vendor specific override This signal is asynchronous and is permitted to change during MacCLKReset# assertion. The MAC must ensure this signal is at a valid value at MacCLKReset# deassertion. When MacCLKReset# is deasserted, this signal is permitted to change only when MacCLKReq and MacCLKAck are deasserted. The MAC must ensure MacCLKRate is stable before MacCLKReq asserts.	PCIe, SATA, USB, DisplayPort, USB4
MacCLKReq	High	The MAC sets this signal to "1" to request the MacCLK to run and sets this signal to "0" to request the MacCLK to stop. When this signal is asserted, the PHY is required to start the MacCLK and then to subsequently assert MacCLKAck to indicate the MacCLK is valid and running at the desired rate. When this signal is deasserted, the PHY is required to stop the MacCLK and deassert MacCLKAck before or after MacCLK is stopped. This signal is asynchronous. The MAC must ensure this signal is valid at MacCLKReset# deassertion. This signal defaults to "0" when MacCLKReset# deasserts.	PCIe, SATA, USB, DisplayPort, USB4
MacCLKSSCEnable	High	The MAC sets this to signal to "1" to request SSC to be enabled on the MacCLK. This signal is set to "0" to request SSC to be disabled on the MacCLK. This signal is asynchronous and is permitted to change during MacCLKReset# assertion. The MAC must ensure this signal is at a valid value at MacCLKReset# deassertion. MacCLKSSCEnable is permitted to be dynamically asserted or deasserted while MacCLK is running.	PCIe, SATA, USB, DisplayPort, USB4
MacCLKReset#	Low	Resets the MacCLK lane state. This signal is asynchronous and is permitted to assert at any time regardless of PHY state.	PCIe, SATA, USB, DisplayPort, USB4

Table 6-18. MacCLK Lane Output Signals

Name	Active Level	Description	Relevant Protocols
MacCLK	Rising Edge	Clock from the PHY. The frequency of MacCLK is determined by MacCLKPHYMode and MacCLKRate or via a vendor specific mechanism if selected via MacCLKRate.	PCIe, SATA, USB, DisplayPort, USB4
MacCLKAck	High	4-way full handshake with MacCLKReq. The PHY sets this signal to "1" to indicate the MacCLK is valid and running at the desired frequency. The PHY sets this signal to "0" to indicate that it has or it is in the process of stopping the MacCLK. This signal is asynchronous. This signal defaults to "0" when MacCLKReset# deasserts.	PCIe, SATA, USB, DisplayPort, USB4

6.3 PHY/MAC Interface Signals – Original PIPE Only

This section describes the signals for original PIPE that are required in addition to those define in Section 6.1.

6.3.1 Data Interface

Table 6-19. Original PIPE Only: Tx Data Interface Input Signals

Name	Active Level	Description	Relevant Protocols
TxDataK[7:0] for 64-bit interface TxDataK[3:0] for 32-bit interface TxDataK[1:0]for 16-bit interface TxDataK for 8-bit interface	N/A	**This signal is not used in the SerDes architecture.** Data/Control for the symbols of Tx data. For 64-bit interfaces, Bit 0 corresponds to the low-byte of TxData and Bit 7 corresponds to the upper byte. For 32-bit interfaces, Bit 0 corresponds to the low-byte of TxData, Bit 3 corresponds to the upper byte. For 16-bit interfaces, Bit 0 corresponds to the low-byte of TxData, Bit 1 to the upper byte. A value of zero indicates a data byte, a value of 1 indicates a control byte. Not used in PCIe mode at 8 GT/s or higher rates. Not used in USB mode at 10 GT/s. Not used in USB4 mode.	PCIe, SATA, and USB
TxStartBlock	N/A	**This signal is not used in the SerDes architecture.** PCIe mode and USB mode: Only used at the 8.0 GT/s, 16 GT/s, and 32 GT/s PCIe signaling rates and the 10 GT/s USB signaling rate. These signals allow the MAC to tell the PHY that the starting byte for a 128b block. The starting byte for a 128b block must always start with byte zero of the data interface.	PCIe and USB

Table 6-20. Original PIPE Only: Rx data Interface Output Signals

Name	Active Level	Description	Relevant Protocols
RxDataK[3:0] for 32-bit interface RxDataK[1:0] for 16-bit interface RxDataK for 8-bit interface	N/A	**This signal is not used in the SerDes architecture.** Data/Control bit for the symbols of Rx data. For 32-bit interfaces, Bit 0 corresponds to the low-byte of RxData, Bit 3 corresponds to the upper byte. For 16-bit interface, Bit 0 corresponds to the low-byte of RxData[15:0], Bit 1 to the upper byte. A value of zero indicates a data byte; a value of 1 indicates a control byte. Not used in PCIe mode at 8 GT/s or higher rates or USB mode at 10 GT/s or USB4 mode. When the PHY is in a SATA mode, the first valid data following an ALIGN primitive must appear as byte zero in the Rx data.	PCIe, SATA, and USB
RxDataValid	N/A	**This signal is not used in the SerDes architecture.** PCIe mode and SATA mode and USB mode: This signal allows the PHY to instruct the MAC to ignore the data interface for one clock cycle. A value of one indicates that the MAC will use the data, a value of zero indicates that the MAC will not use the data. RxDataValid shall not assert when RxValid is de-asserted in PHY modes that require the use of RxDataValid. If a PHY supports the RxDataValid signal, it must keep RxDataValid asserted when the PHY is in a mode that does not require the signal. The MAC may ignore RxDataValid when it is in a mode that does not require the signal.	PCIe, SATA, USB
RxStartBlock	N/A	**This signal is not used in the SerDes architecture.** PCIe mode and USB mode: Only used at the 8.0 GT/s, 16 GT/s, and 32 GT/s PCIe signaling rates and the 10 GT/s USB signaling rate. This signal allows the PHY to tell the MAC the starting byte for a 128b block. The starting byte for a 128b block must always start with byte zero of the data interface. If there is an invalid sync header decoded on RxSyncHeader[3:0] and block alignment is still present (RxValid == 1), then the PHY will assert RxStartBlock with the invalid sync header on RxSyncHeader[3:0] RxStartBlock shall not assert when RxValid is de-asserted	PCIe, SATA, USB

6.3.2 Command Interface

Table 6-21. Command Interface Input Signals (Sheet 1 of 2)

Name	Active Level	Description	Relevant Protocols
TxCompliance	High	**This signal is not used in the SerDes architecture.** PCIe mode: Sets the running disparity to negative. Used when transmitting the PCIe compliance pattern. Implementation of this signal is only required for PHYs that support the PCIe mode. This signal is sampled by TxDataValid.	PCIe

Table 6-21. Command Interface Input Signals (Sheet 2 of 2)

Name	Active Level	Description	Relevant Protocols
TxSyncHeader[3:0]	N/A	**This signal is not used in the SerDes architecture.** PCIe mode: Only the lower two bits ([1:0]) are utilized. Provides the sync header for the PHY to use in the next 130b block. The PHY reads this value when the TxStartBlock signal is asserted. This signal is only used at the 8.0 GT/s, 16 GT/s, and 32 GT/s signaling rates. USB mode: Provides the sync header for the PHY to use in the next 132b block. The PHY reads this value when the TxStartBlock signal is asserted. This signal is only used at the 10 GT/s signaling rate.	PCIe and USB

Table 6-22. Original PIPE Only: Command Interface Output Signals

Name	Active Level	Description	Relevant Protocols
RxSyncHeader[3:0]	N/A	**This signal is not used in the SerDes architecture.** PCIe mode: Only the lower two bits ([1:0]) are utilized. It provides the sync header for the MAC to use with the next 128b block. The MAC reads this value when the RxStartBlock signal is asserted. This signal is only used at the 8.0 GT/s, 16 GT/s, and 32 GT/s signaling rates. USB mode: Provides the sync header for the MAC to use with the next 128b block. The MAC reads this value when the RxStartBlock signal is asserted. This signal is only used at the 10.0 GT/s signaling rate. **Note:** The PHY must pass blocks and headers normally across the PIPE interface even if the decoded SyncHeader is invalid.	PCIe, USB

Table 6-23. Original PIPE Only: Status Interface Output Signals

Name	Active Level	Description	Relevant Protocols
AlignDetect	High	**This signal is not used in the SerDes architecture.** Indicates the receiver detection of an ALIGN. A PHY is only required to assert this signal when the elasticity buffer is running in nominal empty mode. The PHY must only toggle this signal after obtaining the bit and symbol lock. Each ALIGN received must map to AlignDetect being asserted for one PCLK. The spacing between PCLK pulses for ALIGNs should map the analog spacing of the received ALIGNs as closely as possible. However, there is no guarantee to have a PCLK domain spacing between back to back AlignDetect pulses that match the analog spacing exactly due to differences in the receive clock domain and the PCLK domain. For example: 1.5 GT/s with 8-bit data path PCLK=150 MHz, the nominal spacing is 4 PCLKs. 3.0 GT/s with 8-bit data path PCLK=300 MHz, the nominal spacing is 4 PCLKs. 6.0 GT/s with 16-bit data path PCLK=300 MHz, the nominal spacing is every other PCLK. Due to differences in the PCLK and receive clocks, the nominal spacing can be off by one PCLK in either direction. In the example with the PCLK rate being equals to Gen3 received clock rate, clock domain crossing could lead to AlignDetect being asserted for consecutive PCLK cycles without gap.	SATA

6.4 External Signals – Common for SerDes and Original PIPE

Table 6-24. External Input Signals

Name	Active Level	Description	Relevant Protocols
CLK	Edge	This differential input is used to generate the bit-rate clock for the PHY transmitter and receiver. Specifications for this clock signal (frequency, jitter, and so forth) are implementation-dependent and must be specified for each implementation. This clock may have a spread spectrum modulation.	PCIe, SATA, USB, DisplayPort, and USB4
PCLK	Rising Edge	**This signal is relevant for the "PCLK as PHY Input" mode only.** All data movement across the parallel interface is synchronized to this clock. This clock operates at a frequency set by the **PCLK rate**. The rising edge of the clock is the reference for all signals. Spread spectrum modulation on this clock is allowed.	PCIe, SATA, USB, DisplayPort, USB4

Table 6-25. External Output Signals (Sheet 1 of 2)

<table>
<tr><th>Name</th><th>Active Level</th><th>Description</th><th>Relevant Protocols</th></tr>
<tr><td>PCLK</td><td>Rising Edge</td><td>This signal is relevant for “PCLK as PHY Output” mode only.

All data movement across the parallel interface is synchronized to this clock. This clock operates at a frequency set by PCLK Rate. The rising edge of the clock is the reference for all signals. The spread spectrum modulation on this clock is allowed.</td><td>PCIe, SATA, and USB</td></tr>
<tr><td>Max PCLK</td><td>Rising Edge</td><td>Parallel interface data clock. This fixed rate clock operates at the rate advertised in the PHY datasheet subject to the following limitations:

PCIe mode:
<table>
<tr><th>Max rate supported</th><th>Maximum Max PCLK</th></tr>
<tr><td>2.5 GT/s</td><td>250 MHz</td></tr>
<tr><td>5.0 GT/s</td><td>500 MHz</td></tr>
<tr><td>8.0 GT/s</td><td>1000 MHz</td></tr>
<tr><td>16.0 GT/s</td><td>2000 MHz</td></tr>
<tr><td>32.0 GT/s</td><td>4000 Mhz</td></tr>
<tr><td>64.0 GT/s</td><td>4000 Mhz</td></tr>
<tr><td>128 GT/s</td><td>4000 Mhz</td></tr>
</table>
This clock is provided whenever PCLK is active.

SATA mode:
<table>
<tr><th>Max rate supported</th><th>Maximum Max PCLK</th></tr>
<tr><td>1.5 GT/s</td><td>150 MHz</td></tr>
<tr><td>3.0 GT/s</td><td>300 MHz</td></tr>
<tr><td>6.0 GT/s</td><td>600 MHz</td></tr>
</table>
This clock is provided whenever PCLK is active.

USB mode:
<table>
<tr><th>Max rate supported</th><th>Maximum Max PCLK</th></tr>
<tr><td>5.0 GT/s</td><td>500 MHz</td></tr>
<tr><td>10.0 GT/s</td><td>1250 MHz</td></tr>
</table>
This clock is provided whenever the PCLK is active.

USB4 mode:
<table>
<tr><th>Max rate supported</th><th>Maximum Max PCLK</th></tr>
<tr><td>10 GT/s</td><td>1250 Mhz</td></tr>
<tr><td>20 GT/s</td><td>2500 Mhz</td></tr>
</table>
Spread spectrum modulation on this clock is allowed.

This signal is optional for most cases in “PCLK as PHY Output” mode and required for “PCLK as PHY Input” mode.</td><td>PCIe, SATA, USB, DisplayPort, and USB4</td></tr>
</table>

Table 6-25. External Output Signals (Sheet 2 of 2)

<table>
<tr><th>Name</th><th>Active Level</th><th>Description</th><th>Relevant Protocols</th></tr>
<tr><td>DataBusWidth[1:0]</td><td>N/A</td><td>This field reports the width of the data bus that the PHY is configured for.

This field is optional.

For Original PIPE architecture:
<table>
<tr><th>[1]</th><th>[0]</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>32-bit mode</td></tr>
<tr><td>0</td><td>1</td><td>16-bit mode</td></tr>
<tr><td>1</td><td>0</td><td>8-bit mode</td></tr>
<tr><td>1</td><td>1</td><td>Reserved</td></tr>
<tr><td colspan="3">For SerDes architecture:</td></tr>
<tr><th>[1]</th><th>[0]</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>10-bit mode</td></tr>
<tr><td>0</td><td>1</td><td>20-bit mode</td></tr>
<tr><td>1</td><td>0</td><td>40-bit mode</td></tr>
<tr><td>1</td><td>1</td><td>80-bit mode</td></tr>
</table></td><td>PCIe, SATA, USB, DisplayPort, and USB4</td></tr>
</table>

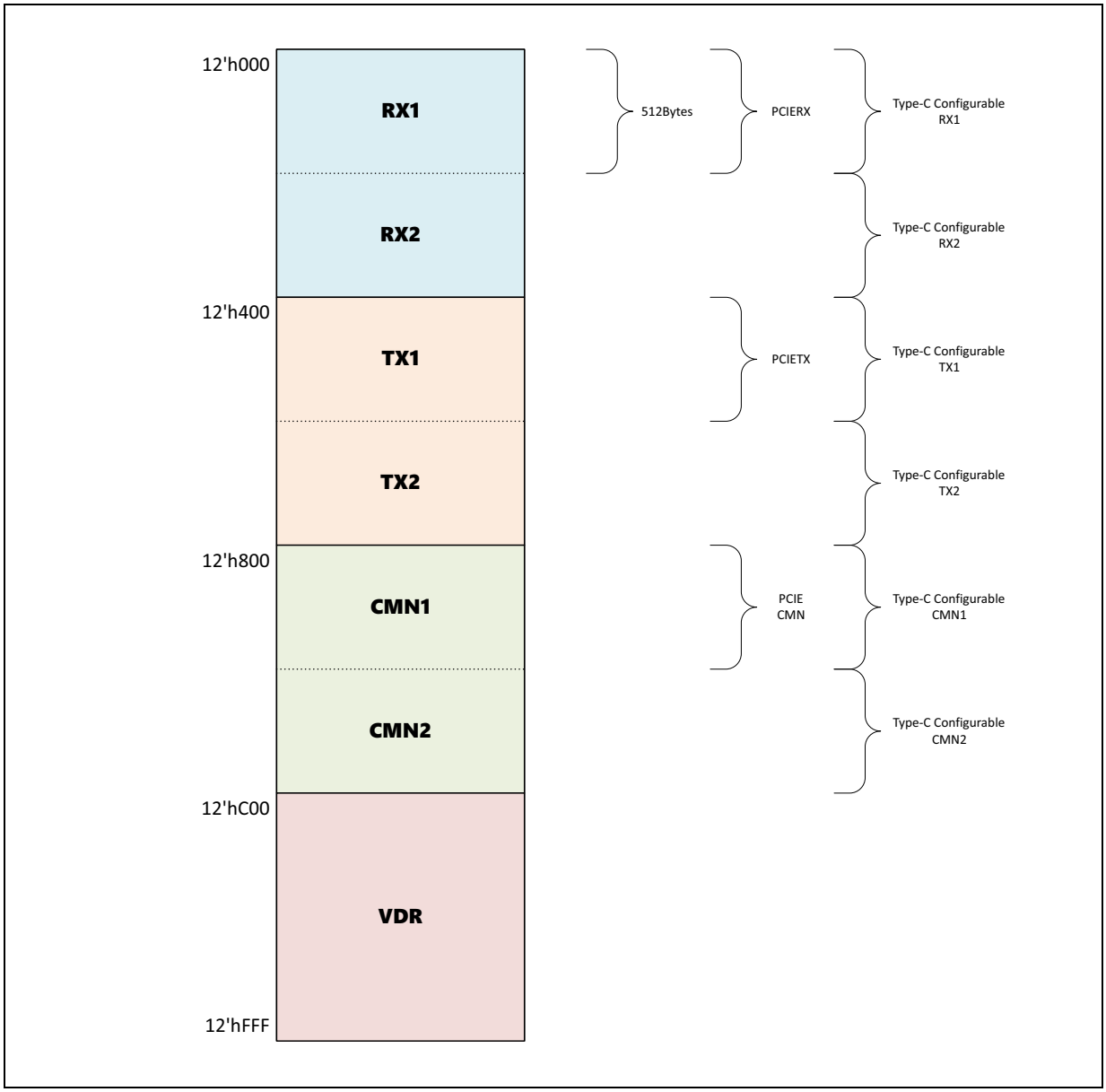
7 PIPE Message Bus Address Spaces

The PIPE specification defines 12-bit address spaces to enable the message bus interface; the MAC and the PHY each implement a unique 12-bit address space as shown in Figure 7-1. These address spaces are used to host registers associated with certain PIPE operations. The MAC and PHY access specific bits in the registers to initiate operations, to participate in handshakes, or to indicate status. The MAC initiates requests on the message bus interface to access registers hosted in the PHY address space. The PHY initiates requests on the message bus interface to access registers hosted in the MAC address space.

Each 12-bit address space is divided into four main regions: the receiver address region, the transmitter address region, the common address region, and the vendor-specific address region. The receiver address region is used to configure and report the status related to receiver operation; it spans the 1024 KB region from 12'h000 to 12'h3FF and supports up to two receivers with 512 KB allocated to each. The transmitter address region is used to configure and report status related to transmitter operation; it spans the 1024 KB region from 12'h400 to 12'h7FF and supports up to two transmitters: TX1 and TX2, with a 512KB region associated with each. The common address region hosts the registers relevant to both receiver and transmitter operation; it spans the 1024 KB region from 12'h800 to 12'hBFF and supports up two sets of Rx/Tx pairs with 512 KB allocated towards the common registers for each pair. The vendor-specific address region is the 1024K region from 12'hC00 to 12'hFFF, which enables individual vendors to define registers as needed outside of those defined in this PIPE specification.

As noted in the previous paragraphs, the address space is defined to support configurable Rx/Tx pairs. Up to two differential pairs are assumed to be operational at any one time. Supported combinations are one Rx and one Tx pair, two Tx pairs, or two Rx pairs.

Figure 7-1. Message Bus Address Space



The PCIe Rx margining operations and elastic buffer depth are controlled via registers hosted in these address spaces. Additionally, several legacy pipe control and status signals have been mapped into the registers hosted in these address spaces.

The following subsections define the PHY registers and the MAC registers. Individual register fields are specified as required or optional. In addition, each field has an attribute description of either level or 1-cycle assertion. When a level field is written, the value written is maintained by the hardware until the next write to that field or until a reset occurs. When a 1-cycle field is written to assert the value high, the hardware maintains the assertion for only a single cycle and then automatically resets the value to zero on the next cycle.

7.1 PHY Registers

Table 7-1 lists the PHY registers and their associated address. The details of each register are provided in the following subsections.

To support configurable pairs, the same registers defined for RX1 are also defined for RX2, the same registers defined for TX1 are defined for TX2, and the same registers defined for CMN1 are defined for CMN2. Only two differential pairs are active at a time based on configuration; valid combinations correspond to registers defined in RX1+TX1+CMN1, RX1+RX2+CMN1+CMN2, or TX1+TX2+CMN1+CMN2.

A PHY that does not support configurable pairs only implements registers defined for RX1, TX1, and CMN1.

Table 7-1. PHY Registers (Sheet 1 of 2)

Byte Address	Register Name	Notes
12'h0	RX1: Rx Margin Control0	
12'h1	RX1: Rx Margin Control1	
12'h2	RX1: Elastic buffer control	N/A for SerDes architecture
12'h3	RX1: PHY Rx Control0	N/A for SerDes architecture
12'h4	RX1: PHY Rx Control1	
12'h5	RX1: PHY Rx Control2	
12'h6	RX1: PHY Rx Control3	
12'h7	RX1: Elastic buffer location update frequency	N/A for SerDes architecture
12'h8	RX1: PHY Rx Control4	Some fields N/A for SerDes architecture
12'h9	RX1: PHY Rx Control 5	
12'h10-12'h1FF	RX1: Reserved	
12'h200 to 12'h3FF	RX2: Same registers are defined in this region for RX2 as for RX1 above.	
12'h400	TX1: PHY Tx Control0	N/A for SerDes architecture
12'h401	TX1: PHY Tx Control1	N/A for SerDes architecture
12'h402	TX1: PHY Tx Control2	
12'h403	TX1: PHY Tx Control3	
12'h404	TX1: PHY Tx Control4	
12'h405	TX1: PHY Tx Control5	
12'h406	TX1: PHY Tx Control6	
12'h407	TX1: PHY Tx Control7	
12'h408	TX1: PHY Tx Control8	
12'h409	TX1: PHY Tx Control9	
12'h40A	TX1: PHY Tx Control10	
12'h40B-12'h5FF	TX1: Reserved	
12'h600-12'h7FF	TX2: Same registers are defined in this region for TX2 as for TX1 in previous rows.	
12'h800	CMN1: PHY Common Control0	N/A for SerDes architecture
12'h801	PHY Near-end loopback control	
12'h802-12'h9FF	CMN1: Reserved	

Table 7-1. PHY Registers (Sheet 2 of 2)

Byte Address	Register Name	Notes
12'hA00 – 12'BFF	CMN2: Same registers are defined in this region for CMN2 as for CMN1 above	
12'hC00-12'hFFF	VDR: Reserved	

7.1.1 Address 0h: Rx Margin Control0

This register is used along with the Rx Margin Control1 to control the PCIe Lane Margining at the receiver.

Table 7-2. Address 0h: Rx Margin Control0

Bit	Default	Attribute	Required	Description
[7:5]	0h	N/A	N/A	Reserved
[4]	0h	Level	USB4	**Voltage Margining Eye**: This field is used only for PAM3 signaling rates. This field indicates which of the eyes is being margined. 0 - Bottom Eye 1 - Top Eye
[3]	0h	1-cycle	PCIe (optional)	**Sample Count Reset**: This field is used to reset the "Sample Count[6:0]" field of the Rx Margin Status1 register.
[2]	0h	1-cycle	PCIe (optional)	**Error Count Reset**: This field is used to reset the "Error Count[5:0]" field of the Rx Margin Status2 register.
[1]	0h	Level	PCIe	**Margin Voltage or Timing:** This field is used to select between margining voltage (1'b0) or margining timing (1'b1). The value can be changed only when margining is stopped.
[0]	0h	Level	PCIe	**Start Margin:** This field is used to start and stop margining. A transition from 1'b0 to 1'b1 starts the margining process. A transition from 1'b1 to 1'b0 stops the margining process.

7.1.2 Address 1h: Rx Margin Control1

This register is used along with the Rx Margin Control0 to control the PCIe Rx margining.

Table 7-3. Address 1h: Rx Margin Control1

Bit	Default	Attribute	Required	Description
[7]	0h	Level	PCIe	**Margin Direction:** This field is used to control the time or voltage direction for margining. For timing margining, this field steps the time left (1'b0) or right (1'b1).[1] For voltage margining, this field steps the voltage up (1'b0) or down (1'b1). This value can be changed only when the margining is stopped. This field should be ignored by PHYs that do not support individual time or voltage margining as advertised in the PHY datasheet.
[6:0]	0h	Level	PCIe	**Margin Offset:** This field is used to change the margin offset a number of steps from the default position. This value can be changed even during the margining process.

1. This is reversed from the timing margining direction convention used in the PCIe Base Specification.

7.1.3 Address 2h: Elastic Buffer Control

This register is used to control the elastic buffer depth, enabling the controller to optimize latency in nominal half full mode. The ability to control the elastic buffer depth is an optional feature that may be especially beneficial for retimers operating in the PCIe SRIS mode.

Table 7-4. Address 2h: Elastic Buffer Control

Bit	Default	Attribute	Required	Description
[7:0]	0h	Level	PCIe (optional)	**Elastic Buffer Depth Control:** This field is used to set the elastic buffer depth. The MAC must choose from the supported values advertised in the PHY datasheet. This value can only be changed during the transmission of TS1 ordered sets. The PHY performs the adjustment as quickly as possible without waiting for SKPs. The PHY signals completion of elastic buffer depth adjustment by setting the Elastic Buffer Status register. **Note:** This field is not used in the SerDes architecture.

7.1.4 Address 3h: PHY Rx Control0

This register is used to control receiver functionality.

Table 7-5. Address 3h: PHY Rx Control0 (Sheet 1 of 2)

Bit	Default	Attribute	Required	Description	
[7:2]	0h	N/A	N/A	Reserved	
[1]	0h	Level	PCIe, USB, and SATA	**RxPolarity:** This field is used to control the polarity inversion on the received data.	
				Value	**Description**
				0	PHY does no polarity inversion
				1	PHY does polarity inversion
				Note: This field is not used in the SerDes architecture.	

Table 7-5. Address 3h: PHY Rx Control0 (Sheet 2 of 2)

<table>
<tr><th>Bit</th><th>Default</th><th>Attribute</th><th>Required</th><th colspan="2">Description</th></tr>
<tr><td rowspan="5">[0]</td><td rowspan="5">0h</td><td rowspan="5">Level</td><td rowspan="5">PCIe (optional), SATA (optional), and USB (optional)</td><td colspan="2">Elasticity Buffer Mode: This field is used to select the elasticity buffer operating mode.</td></tr>
<tr><td>Value</td><td>Description</td></tr>
<tr><td>0</td><td>Nominal half full buffer mode</td></tr>
<tr><td>1</td><td>Nominal empty buffer Mode</td></tr>
<tr><td colspan="2">This field can only be changed when the receiver is OFF and the Pclk is running, for example, the P0 with RXStandby is asserted or P1.
This field is not valid when TxDetectRx/ Loopback is asserted. The PHY is responsible for switching to Nominal Half Full Buffer mode when loopback follower is requested. The Process Control System (PCS) is responsible for making the stream switch and abiding by the PCIe base specification rules for follower loopback stream switching, for instance, switch on the 10b boundary in 8b/10b modes.
Note: This field is not used in the SerDes architecture.</td></tr>
</table>

7.1.5 Address 4h: PHY Rx Control1

This register is used to control the receiver functionality.

Table 7-6. Address 4h: PHY Rx Control1 (Sheet 1 of 2)

Bit	Default	Attribute	Required	Description
[7:3]	0h	N/A	N/A	Reserved

Table 7-6. Address 4h: PHY Rx Control1 (Sheet 2 of 2)

Bit	Default	Attribute	Required	Description
[4]	0h	Level	PCIe (optional)	**RxInPhase01Equalization:** This field is set to "1" by the MAC to communicate to the PHY that the far end transmitter has applied the requested preset during Phase0 (for Upstream Port) or Phase1 of transmitter equalization. This field must be cleared when exiting Phase 1. If RxEqTraining is used, this field must be set prior to or at the same time as RxEqTraining. If RxEqEval is used, this field must be set prior to or at the same time as RxEqEval. See **Figure 9-19** for an example of a valid sequence. The PHY advertises in its datasheet via the RxInPhase01EqRequirement parameter whether this signaling functionality is required.
[3]	0h	Level	USB4	**RxPAM3Mode:** This field is set to "1" by the MAC to inform the PHY that it is receiving data in PAM3 mode.
[2]	0h	Level	PCIe (optional)	**PAM4RestrictedLevels:** This field is set to "1" by the MAC to inform the PHY that the link partner is transmitting or that it will soon be transmitting TS0's at 64 GT/s or higher rates. This field is only applicable to PCIe at 64 GT/s and higher rates. The MAC must set this field to "1" after a rate change to 64 GT/s or higher rate if it is expecting to perform a Tx Equalization procedure; this field must be set after PhyStatus indication of rate change completion but prior to the MAC deasserting RxStandby. The MAC must NOT set this field after a rate change to 64 GT/s if no Tx Equalization procedure is expected (i.e. due to PCIe NoEq). During Tx Equalization at 64 GT/s or higher, the MAC must clear this field when it requests the link partner to transition from transmitting TS0's to TS1's; this occurs during Phase 2 for Upstream Lanes and during Phase 3 for Downstream Lanes. If Phase 2 and Phase 3 of Equalization is bypassed, then the MAC must clear this field during Recovery.RcvrLock; it must subsequently initiate receiver training via either RxEqTraining or RxEqEval. The PHY must autonomously clear this field when performing a rate change to 64 GT/s or higher before it returns PhyStatus for the rate change. The PHY advertises in its datasheet via the PAM4RestrictedLevelsRequirement parameter whether this signaling functionality is required.
[1]	0h	1-cycle	PCIe (optional), USB (optional), and USB4 (optional)	**IORecal:** This field is set to "1" to request the PHY to do an Rx recalibration. The controller asserts this signal either in response to the PhyIORecalRequest or autonomously if it determines a recalibration is needed. The PHY indicates completion of recalibration via the IORecalDone bit. The PHY advertises in its datasheet via the PhyRecalRequirement parameter whether this functionality is required.
[0]	0h	Level (USB), 1-cycle (PCIe)	USB, USB4, and PCIe (optional)	**RxEqTraining:** This field is set to 1'b1 to instruct the receiver to bypass normal operation to perform equalization training. While performing training the state of the RxData interface is undefined. Implementation of this bit is optional for PCIe. The PHY advertises in its datasheet via the PCIeRxEqTrainRequirement parameter whether this functionality is required. If implemented for the PCIe, a full handshake with RxEqTrainDone is required.

7.1.6 Address 5h: PHY Rx Control2

This register is used to control receiver functionality.

Table 7-7. Address 5h: PHY Rx Control2

Bit	Default	Attribute	Required	Description
[7:3]	0h	N/A	N/A	Reserved
[2:0]	0h	N/A	N/A	Reserved

7.1.7 Address 6h: PHY Rx Control3

This register is used to control receiver functionality.

Table 7-8. Address 6h: PHY Rx Control3

Bit	Default	Attribute	Required	Description
[7:3]	0h	N/A	N/A	Reserved
[2]	0h	Level	PCIe	**InvalidRequest:** This field is used to indicate that the Link Evaluation feedback requested a link partner Tx EQ setting that was out of range. The MAC sets this bit to "1" when it detects an out-of-range error locally based on a calculated link partner transmitter coefficient based on the last valid link equalization feedback, or if it receives a NACK response from the link partner. The MAC resets this bit to "0" the next time it asserts RxEQEval. When a MAC sets this bit, it must subsequently ask the PHY to perform an RxEQ evaluation using the last valid setting a second time. This field is only applicable at the 8.0 GT/s, 16 GT/s, 32 GT/s, 64 GT/s,and 128 GT/s signaling rates.
[1]	0h	Level	PCIe, USB4, and DisplayPort Rx (optional)	**RxEqInProgress:** This field is used by the MAC to indicate when a link equalization evaluation is in progress. The PHY may optionally use this field to enable and disable functionality that is only needed during link equalization evaluations. For PCIe: The MAC sets this bit to "1" at the same time as it sets RxEqEval to "1" in phase 2 or phase 3 of the link equalization process. The MAC resets this bit to "0" at the end of phase two or phase three.
[0]	0h	Level	PCIe, USB4, and DisplayPort Rx (optional)	**RxEqEval:** This field is set to "1" by the MAC to instruct the PHY to start the evaluation of the far-end transmitter Tx EQ settings. For PCIe, this field is only used at the 8.0 GT/s, 16 GT/s, 32 GT/s, 64 GT/s,and 128 GT/s signaling rates.

7.1.8 Address 7h: Elastic Buffer Location Update Frequency

Table 7-9. Address 7h: Elastic Buffer Location Update Frequency

Bit	Default	Attribute	Required	Description
[7:0]	5h	Level	No	**ElasticBufferLocationUpdateFrequency:** This field specifies the maximum update frequency to the ElasticBufferLocation field. The update frequency should not exceed 16×N symbol times, where N is the value programmed in this register. **Note:** This field is not used in the SerDes architecture.

7.1.9 Address 8h: PHY Rx Control4

This register is used to control the receiver functionality.

Table 7-10. Address 8h: PHY Rx Control4

Bit	Default	Attribute	Required	Description
[7:2]	0h	N/A	N/A	Reserved
[1]	0h	1-cycle	PCIe and USB	**ElasticBufferResetControl:** When asserted, this signal causes the PHY to initiate an EB reset sequence. See Section 8.15.3.1 for details. **Note:** This field is not used in the SerDes architecture.
[0]	0h	Level	PCIe and USB	**BlockAlignControl:** This field controls whether the PHY performs a block alignment. When BlockAlignControl=0 the PHY disables searching for the EIEOS (PCIe)/SYNC OS (USB) on a bit boundary. When BlockAlignControl = 1 the PHY enables searching for the EIEOS(PCIe)/SYNC OS (USB) on a bit boundary. A MAC must set BlockAlignControl to the same value for all active lanes in a link. A MAC must set BlockAlignControl to "0" when in a datastream and must set it to "1" otherwise. This field is only used at the PCIe 8.0 GT/s, 16 GT/s, and 32 GT/s signaling rates and at the USB 10.0 GT/s signaling rate. When the PHY is in Loopback Follower mode it ignores the BlockAlignControl and is responsible for maintaining alignment. **Note:** This field is not used in the SerDes architecture.

7.1.10 Address 9h: PHY Rx Control 5

This register controls receiver functionality.

Table 7-11. Address 9h: PHY Rx Control 5

Bit	Default	Attribute	Required	Description
[7:1]	0h	N/A	N/A	Reserved
[0]	1h[1]	Level	USB4 and DisplayPort	**RxLaneEnable:** This field is set to "1" by the MAC to instruct the PHY to enable the receiver. The PHY indicates completion of this operation via the RxLaneReady register bit.

1. The default value of RxLaneEnable for the RX2 region is 0h.

7.1.11 Address 400h: PHY Tx Control0

This register is used to control the transmitter functionality.

Table 7-12. Address 400h: PHY Tx Control0 (Sheet 1 of 2)

Bit	Default	Attribute	Required	Description
[7:2]	0h	N/A	N/A	Reserved

Table 7-12. Address 400h: PHY Tx Control0 (Sheet 2 of 2)

Bit	Default	Attribute	Required	Description
[1:0]	0h	Level	SATA	**Tx Pattern[1:0]:** This field controls which pattern the PHY sends at the Gen1 rate when sending OOB or initialization signaling. The PHY transmits this pattern at the Gen 1 rate regardless of what rate the PHY is configured at. 0 ALIGN 1 D24.3 2 D10.2 3 Reserved See Section 8.23 for a more detailed description of the usage of these pins. **Note:** This field is not used in the SerDes architecture.

7.1.12 Address 401h: PHY Tx Control1

This register is used to control the transmitter functionality.

Table 7-13. Address 401h: PHY Tx Control1

Bit	Default	Attribute	Required	Description
[7:1]	0h	N/A	N/A	Reserved
[0]	0h	Level	USB	**TxOnesZeros:** This field is used when transmitting USB-compliance patterns CP7 or CP8. When this field is set, the transmitter is to transmit an alternating sequence of 50-250 ones and 50–250 zeros (regardless of the state of the TxData interface). This field is only applicable to 8b/10b modes. **Note:** This field is not used in the SerDes architecture.

7.1.13 Address 402h: PHY Tx Control2

This register is used to control the transmitter functionality.

Table 7-14. Address 402h: PHY Tx Control2 (Sheet 1 of 2)

Bit	Default	Attribute	Required	Description
[7:6]	0h	N/A	N/A	Reserved

Table 7-14. Address 402h: PHY Tx Control2 (Sheet 2 of 2)

<table>
<tr><th>Bit</th><th>Default</th><th>Attribute</th><th>Required</th><th>Description</th></tr>
<tr><td>[5:0]</td><td>1h</td><td>Level</td><td>PCIe, USB, USB4, and DisplayPort</td><td>
TxDeemph[5:0]/TxDeemph_Cminus1[5:0]

For applicable protocols with the exception of PCIe at 64 GT/s and higher rates and with the exception of USB4 at 40 GT/s, this field is TxDeemph[5:0] and defined as follows:

This field is part of TxDeemph[17:0], which selects the transmitter de-emphasis.

PCIe mode when the rate is 2.5 or 5.0 GT/s:

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>-6 dB de-emphasis</td></tr>
<tr><td>1</td><td>-3.5 dB de-emphasis</td></tr>
<tr><td>2</td><td>No de-emphasis</td></tr>
<tr><td>3</td><td>Reserved</td></tr>
</table>
PIPE implementations that only support 2.5 GT/s do not implement this field. PIPE PHY implementations that do not support low swing are not required to support the no-de-emphasis mode.

PCIe mode when the rate is 8.0 GT/s, 16 GT/s, or 32 GT/s:
[5:0] C_{-1}
[11:6] C_{0}
[17:12] C_{+1}
USB mode when the rate is 10.0 GT/s:
[5:0] C_{-1}
[11:6] C_{0}
[17:12] C_{+1}

The field is not defined for USB Mode when the rate is 5.0 GT/s

USB4 mode when the rate is 10 GT/s or 20 GT/s:
[5:0] C_{-1}
[11:6] C_{0}
[17:12] C_{+1}

DisplayPort mode when the rate is 10 GT/s or 13.5 GT/s or 20 GT/s
[5:0] C_{-1}
[11:6] C_{0}
[17:12] C_{+1}

Note: The MAC must ensure that only supported values are used for TxDeemph. In cases where the implementation is required to keep track of Tx coefficients from previous states, this shall be done by the MAC.

For PCIe at 64 GT/s and higher rates and USB4 at 40 GT/s, this field is TxDeemph_Cminus1[5:0] and defined as follows:

TxDeemph_Cminus1[5:0] is part of a set of fields that selects the transmitter de-emphasis and corresponds to precursor C_{-1}. See TxDeemph_Cminus2[5:0], TxDeemph_Czero[5:0], and TxDeemph_Cplus1[5:0] for the other coefficients.

The MAC must ensure that only supported values are used for TxDeemph_Cminus1[5:0]. In cases where the implementation is required to keep track of Tx coefficients from previous states, this must be done by the MAC.
</td></tr>
</table>

7.1.14 Address 403h: PHY Tx Control3

This register is used to control the transmitter functionality.

Table 7-15. Address 403h: PHY Tx Control3

Bit	Default	Attribute	Required	Description
[7:6]	0h	N/A	N/A	Reserved
[5:0]	0h	Level	PCIe, USB, USB4, and DisplayPort	**TxDeemph[11:6]/TxDeemph_Czero[5:0]** For applicable protocols with the exception of PCIe at 64 GT/s and higher rates and with the exception of USB4 at 40 GT/s,, this field is TxDeemph[11:6] and defined as follows: This field is part of TxDeemph[17:0], which selects the transmitter de-emphasis. See TxDeemph[5:0] for a detailed description. For PCIe at 64 GT/s and higher rates and USB4 at 40 GT/s, this field is TxDeemph_Czero[5:0] and defined as follows: TxDeemph_Czero[5:0] field is part of a set of fields that select the transmitter de-emphasis. This field corresponds to cursor C_0. See TxDeemph_Cminus2[5:0], TxDeemph_Cminus1[5:0], and TxDeemph_Cplus1[5:0] for the other coefficients. **Note:** The MAC must ensure that only supported values are used for TxDeemph_Czero[5:0]. In cases where the implementation is required to keep track of Tx coefficients from previous states, this shall be done by the MAC.

7.1.15 Address 404h: PHY Tx Control4

This register is used to control the transmitter functionality.

Table 7-16. Address 404h: PHY Tx Control4

Bit	Default	Attribute	Required	Description
[7:6]	0h	N/A	N/A	Reserved
[5:0]	0h	Level	PCIe, USB, USB4, and DisplayPort	**TxDeemph[17:12]/TxDeemph_Cplus1[5:0]** For applicable protocols with the exception of PCIe at 64 GT/s and higher rates and with the exception of USB4 at 40 GT/s,, this field is TxDeemph[17:12] and defined as follows: This field is part of TxDeemph[17:0], which selects the transmitter de-emphasis. See TxDeemph[5:0] for detailed description. For PCIe at 64 GT/s and higher rates and USB4 at 40 GT/s, this field is TxDeemph_Cplus1[5:0] and defined as follows: TxDeemph_Cplus1[5:0] is part of a set of fields that select the transmitter de-emphasis. This field corresponds to post cursor C_{+1}. See TxDeemph_Cminus2[5:0], TxDeemph_Cminus1[5:0], and TxDeemph_Czero[5:0] for the other coefficients. **Note:** The MAC must ensure that only supported values are used for TxDeemph_Cplus1[5:0]. In cases where the implementation is required to keep track of Tx coefficients from previous states, this must be done by the MAC.

7.1.16 Address 405h: PHY Tx Control5

This register is used to control the transmitter functionality.

Table 7-17. Address 405h: PHY Tx Control5

Bit	Default	Attribute	Required	Description
[7]	0h	1-cycle	PCIe	**GetLocalPresetCoefficients:** This field is used to request a preset to co-efficient mapping for the preset on LocalPresetIndex[5:0] to coefficients on LocalTxPresetCoefficient[17:0] Maximum Response time of PHY is 128 nSec. **Note:** A MAC can make this request anytime after reset. **Note:** This field is only used with a PHY that requires dynamic preset coefficient updates.
[6]	0h	N/A	N/A	Reserved
[5:0]	0h	Level	PCIe	**LocalPresetIndex[5:0]:** This field is used to indicate the index for the local PHY preset coefficients requested by the MAC and is encoded as follows: 000000b – 8 GT/s Preset P0, 000001b – 8 GT/s Preset P1, 000010b – 8 GT/s Preset P2, 000011b – 8 GT/s Preset P3, 000100b – 8 GT/s Preset P4, 000101b – 8 GT/s Preset P5, 000110b – 8 GT/s Preset P6, 000111b – 8 GT/s Preset P7, 001000b – 8 GT/s Preset P8, 001001b – 8 GT/s Preset P9, 001010b – 8 GT/s Preset P10. 001011b – 16 GT/s Preset P0, 001100b – 16 GT/s Preset P1, 001101b – 16 GT/s Preset P2, 001110b – 16 GT/s Preset P3, 001111b – 16 GT/s Preset P4, 010000b – 16 GT/s Preset P5, 010001b – 16 GT/s Preset P6, 010010b – 16 GT/s Preset P7, 010011b – 16 GT/s Preset P8, 010100b – 16 GT/s Preset P9, 010101b – 16 GT/s Preset P10 010110b – 32 GT/s Preset P0, 010111b – 32 GT/s Preset P1, 011000b – 32 GT/s Preset P2, 011001b – 32 GT/s Preset P3, 011010b – 32 GT/s Preset P4, 011011b – 32 GT/s Preset P5, 011100b – 32 GT/s Preset P6, 011101b – 32 GT/s Preset P7, 011110b – 32 GT/s Preset P8, 011111b – 32 GT/s Preset P9, 100000b – 32 GT/s Preset P10 100001b – 64 GT/s Preset P0, 100011b – 64 GT/s Preset P1, 100100b – 64 GT/s Preset P2, 100101b – 64 GT/s Preset P3, 100110b – 64 GT/s Preset P4, 100111b – 64 GT/s Preset P5, 101000b – 64 GT/s Preset P6, 101001b – 64 GT/s Preset P7, 101010b – 64 GT/s Preset P8, 101011b – 64 GT/s Preset P9, 101100b – 64 GT/s Preset P10 101101b - 128 GT/s Preset P0, 101110b - 128 GT/s Preset P1, 101111b - 128 GT/s Preset P2, 110000b - 128 GT/s Preset P3, 110001b - 128 GT/s Preset P4, 110010b - 128 GT/s Preset P5, 110011b - 128 GT/s Preset P6, 110100b - 128 GT/s Preset P7, 110101b - 128 GT/s Preset P8, 110110b - 128 GT/s Preset P9, 110111b - 128 GT/s Preset P10 All others – Reserved This field is only used with a PHY that requires dynamic preset coefficient updates.

7.1.17 Address 406h: PHY Tx Control6

This register is used to control the transmitter functionality.

Table 7-18. Address 406h: PHY Tx Control6

Bit	Default	Attribute	Required	Description
[7:6]	0h	N/A	N/A	**Reserved**
[5:0]	0h	Level	PCIe	**FS[5:0]:** This field reflects the FS value advertised by the link partner. The MAC must only change this value when a new FS value is captured during link training. A PHY may optionally consider this value when deciding how long to evaluate Tx equalization settings of the link partner. The MAC must only change this field when a new FS value is captured during link training or if there is a rate change. The MAC must drive the relevant 8 GT/s values when the operational rate is 8 GT/s, it must drive the relevant 16 GT/s values when the operational rate is 16 GT/s, it must drive the relevant 32 GT/s values when the operational rate is 32 GT/s, it must drive the relevant 64 GT/s values when the operational rate is 64 GT/s, and it must drive the relevant 128 GT/s values when the operational rate is 128 GT/s

7.1.18 Address 407h: PHY Tx Control7

This register is used to control the transmitter functionality.

Table 7-19. Address 407h: PHY Tx Control7

Bit	Default	Attribute	Required	Description
[7:6]	0h	N/A	N/A	Reserved
[5:0]	0h	Level	PCIe	**LF[5:0]:** This field reflects the LF value advertised by the link partner. The MAC must only change this value when a new LF value is captured during link training or when there is a rate change. A PHY may optionally consider this value when deciding how long to evaluate Tx equalization settings of the link partner. The MAC must drive the relevant 8 GT/s values when the operational rate is 8 GT/s, it must drive the relevant 16 GT/s values when the operational rate is 16 GT/s, it must drive the relevant 32 GT/s values, when the operational rate is 32 GT/s, it must drive the relevant 64 GT/s values when the operational rate is 64 GT/s, and it must drive the relevant 128 GT/s values when the operational rate is 128 GT/s.

7.1.19 Address 408h: PHY Tx Control8

This register is used to control the transmitter functionality.

Table 7-20. Address 408h: PHY Tx Control8 (Sheet 1 of 2)

Bit	Default	Attribute	Required	Description
[7:4]	0h	N/A	N/A	Reserved

Table 7-20. Address 408h: PHY Tx Control8 (Sheet 2 of 2)

Bit	Default	Attribute	Required	Description			
[3]	0h	Level	PCIe	**TxSwing:** This field controls transmitter voltage swing level.			
				Value	**Description**		
				0	Full swing		
				1	Low swing (optional)		
				Implementation of this signal is optional if only full swing is supported. This field is not used at the 8.0 GT/s or higher signaling rates.			
[2:0]	0h	Level	PCIe	**TxMargin[2:0]:** This field selects transmitter voltage levels.			
				[2]	**[1]**	**[0]**	**Description**
				0	0	0	TxMargin value 0 = Normal operating range
				0	0	1	TxMargin value 1 = 800–1200 mV for Full swing OR 400-700mV for Half swing
				0	1	0	TxMargin value 2 = required and vendor-defined
				0	1	1	TxMargin value 3 = required and vendor-defined
				1	0	0	TxMargin value 4 = required and 200–400 mV for Full swing OR 100–200 mV for Half swing if the last value or vendor-defined
				1	0	1	TxMargin value 5 = optional and 200-400mV for Full swing OR 100-200mV for Half swing if the last value or vendor-defined or reserved if no other values supported
				1	1	0	TxMargin value 6 = optional and 200–400mV for Full swing OR 100–200mV for Half swing if the last value or vendor-defined or reserved if no other values supported
				1	1	1	TxMargin value 7 = optional and 200-400 mV for Full swing or 100-200 mV for Half swing if the last value or reserved if no other values supported
				PIPE implementations that only support PCIe mode and the 2.5 GT/s signaling rate do not implement this field.			

7.1.20 Address 409h: PHY Tx Control9

This register is used to control the transmitter functionality.

Table 7-21. Address 409h: PHY Tx Control9 (Sheet 1 of 2)

Bit	Default	Attribute	Required	Description
[7:6]	0h	N/A	N/A	Reserved

Table 7-21. Address 409h: PHY Tx Control9 (Sheet 2 of 2)

Bit	Default	Attribute	Required	Description
[5:0]	0h	Level	PCIe and USB4	**TxDeemph_Cminus2[5:0]:** This field is part of a set of fields that select transmitter de-emphasis. This field corresponds to precursor C_{-2}. See TxDeemph_Cminus1[5:0], TxDeemph_Czero[5:0], and TxDeemph_Cplus1[5:0] for the other coefficients. This is only applicable to PCIe Mode when the rate is at 64 GT/s or 128 GT/s and USB4 at 40 GT/s. For lower PCIe operational rates and lower USB4 rates, refer to TxDeemph[17:0]. **Note:** The MAC must ensure that only supported values are used for TxDeemph_Cminus2[5:0]. In cases where the implementation is required to keep track of Tx coefficients from previous states, this shall be done by the MAC.

7.1.21 Address 40Ah: PHY TX Control 10

This register is used to controller transmitter functionality.

Table 7-22. Address 40Ah: PHY TX Control 10

Bit	Default	Attribute	Required	Description
[7:1]	0h	N/A	N/A	Reserved
[0]	1h[1]	Level	USB4 and DisplayPort	**TxLaneEnable**: This field is set to "1" by the MAC to instruct the PHY to enable the transmitter. The MAC is permitted to enable or disable a transmitter in any power state while TxElecIdle is asserted. The PHY indicates completion of this operation via the TxLaneReady register bit.

1. The default value of TxLaneEnable for the RX2 region is 0h.

7.1.22 Address 800h: PHY Common Control0

This register is used to control functionalities relevant to both the receiver and the transmitter functionality.

Table 7-23. Address 800h: PHY Common Control0 (Sheet 1 of 2)

Bit	Default	Attribute	Required	Description
[7:2]	0h	N/A	N/A	Reserved
[1]	0h	N/A	DisplayPort and USB4	**MacTransmitLFPS:** This field controls whether the PHY or the MAC transmit the LFPS. 0 – PHY transmits LFPS 1 – MAC transmits LFPS The MAC must only change this when LFPS is not transmitting, and the MAC must not transmit LFPS at least 10 us after changing this bit.

Table 7-23. Address 800h: PHY Common Control0 (Sheet 2 of 2)

Bit	Default	Attribute	Required	Description
[0]	0h	Level	PCIe (optional), USB (optional), SATA	**EncodeDecodeBypass:** This field controls whether the PHY performs 8b/10b (or 128b/130b) encode and decode. 0 – Encode/decode performed normally by the PHY. 1 – Encode/decode bypassed. The MAC can only change this signal during reset or in a power state other than POWER_STATE_0 (SATA Mode) or P0 (PCI Express Mode). SATA Mode: When EncodeDecodeBypass is one the TxDataK and RxDataK interfaces are not used and the data bus width is 10, 20, or 40 bits. PCIe Mode and USB Mode: When EncodeDecodeBypass is one the TxDataK and RxDataK interfaces are not used. The data bus width is 10, 20, or 40 bits if rate is 2.5 or 5.0 GT/s. The data bus width is 8, 16, or 32 bits if the rate is 8.0 GT/s, 16 GT/s, or 32 GT/s (PCIe) or 10 GT/s (USB). The TxStartBlock and RxStartBlock signals are not used. **Note:** This field is not used in the SerDes architecture.

7.1.23 Address 801h: PHY Near End Loopback Control

This register is required for PHYs that support the optional NELB feature.

Table 7-24. Address 801h: PHY Near End Loopback Control (Sheet 1 of 2)

Bit	Default	Attribute	Required	Description
[7]	0h	Level	PCIe, USB, USB4, and SATA	**NELB Enable** 0 – Normal Operation 1 – Loopback Operation
[6:3]	0h	N/A	N/A	Reserved

Table 7-24. Address 801h: PHY Near End Loopback Control (Sheet 2 of 2)

Bit	Default	Attribute	Required	Description
[2:0]	0h	Level	PCIe, USB, USB4, and SATA	**NELB Loopback Position** Encodings for the original PIPE: Please refer to Figure 8-45. 0 - at PIPE interface boundary 1 - at Elastic Buffer Inlet 2 - at PIPE interface side of deserializer/serializer 3 - at die pad side of deserializer/serializer 4 - High Speed Digital / Analog Path 1 5 - PHY Specific Path 1 6 - PHY Specific Path 2 7 - PHY Specific Path 3 Encodings for SerDes architecture: Please refer to Figure 8-46. 0 - at PIPE interface boundary 1 - reserved 2 - reserved 3 - at die edge side of de-serializer/serializer 4 - High Speed Digital / Analog Path 1 5 - PHY Specific Path 1 6 - PHY Specific Path 2 7 - PHY Specific Path 3 At least one of the positions 0, 1, or 2 must be support by PHYs that support the NELB feature. It is strongly recommended to support position 1 or 2 in Original PIPE architecture mode.

7.2 MAC Registers

Table 7-25 lists the MAC registers and their associated address. The details of each register are provided in the following subsections.

Table 7-25. MAC Registers (Sheet 1 of 2)

Byte Address	Register Name	Notes
12'h0	RX1: Rx Margin Status0	
12'h1	RX1: Rx Margin Status1	
12'h2	RX1: Rx Margin Status2	
12'h3	RX1: Elastic Buffer Status	N/A for SerDes Architecture
12'h4	RX1: Elastic Buffer Location	N/A for SerDes Architecture
12'h5	RX1: Rx Status0	
12'h6	RX1: Rx Control0	
12'h7	RX1: Rx Margin Status3	
12'h8-12'h9	RX1: Reserved	
12'hA	RX1: Rx Link Evaluation Status0	
12'hB	RX1: Rx Link Evaluation Status1	
12'hC	RX1: Rx Status 4	
12'hD	RX1: Rx Status 5	
12'hE	RX1: Rx Link Evaluation Status2	
12'hF	RX1: Rx Link Evaluation Status3	

Table 7-25. MAC Registers (Sheet 2 of 2)

Byte Address	Register Name	Notes
12'h10	RX1: Rx Status 6	
12'h11-12'h1FF	RX1: Reserved	
12'h200 to 12'h3FF	RX2: Same registers are defined in this region for RX2 as for RX1 above.	
12'h400	TX1: Tx Status0	
12'h401	TX1: Tx Status1	
12'h402	TX1: Tx Status2	
12'h403	TX1: Tx Status3	
12'h404	TX1: Tx Status4	
12'h405	TX1: Tx Status5	
12'h406	TX1: Tx Status6	
12'h407	TX1: Tx Status7	
12'h408	TX1: Tx Status8	
12'h409	TX1: Tx Status9	
12'h40A	TX1: Tx Status10	
12'h40B	TX1: Tx Status11	
12'h40C	TX1: TX Status12	
12'h40D-12'h5FF	TX1: Reserved	
12'h600-12'h7FF	TX2: Same registers are defined in this region for TX2 as for TX1 above	
12'h800	CMN1: Near End Loopback Status	
12'h801-12'h9FF	CMN1: Reserved	
12'hA00-12'hBFF	CMN2: Reserved Same registers are defined in this region for CMN2 as for CMN1 above	
12'hC00-12'hFFF	VDR: Reserved	

7.2.1 Address 0h: Rx Margin Status0

Table 7-26. Address 0h: Rx Margin Status0

Bit	Default	Attribute	Required	Description
[7:2]	0h	N/A	N/A	Reserved
[1]	0h	1-cycle	PCIe (optional)	**Margin Nak:** This field is used by the PHY to indicate that a voltage margin request corresponds to an unsupported offset that falls within the advertised range. This field may be asserted in response to a change to the "Start Margin" field or "Margin Offset[6:0]" field or "Margin Direction" field during voltage margining only. This field is only written once per committed write affecting either of the above three fields. When this field is set, the "Margin Status" should not be set. The design must support the minimum voltage offset requirement stated in the PCIe base specification. **Note:** If the voltage margin offset requested falls outside of the PHY advertised range, the PHY is not required to communicate a NAK by setting this field; this is assumed to be a MAC error and PHY behavior is undefined.
[0]	0h	1-cycle	PCIe	**Margin Status:** This field is used by the PHY to acknowledge a valid change to the "Start Margin" field or the "Margin Offset[6:0]" field. This field is only written once per committed write affecting either of the above two fields. For example, if both "Start Margin" and "Margin Offset[6:0]" are changed, but one is changed with an uncommitted write and one is changed with a committed write, this "Margin Status" field is only written once to acknowledge both changes.

7.2.2 Address 1h: Rx Margin Status1

Table 7-27. Address 1h: Rx Margin Status1

Bit	Default	Attribute	Required	Description
[7]	0h	N/A	N/A	Reserved
[6:0]	0h	Level	PCIe (optional)	**Sample Count:** This field indicates the number of bits that have been margined and can increment only when "Start Margin" is asserted. The value of this field is 3×log2(number of bits margined). This field stops incrementing when the "Error Count" saturates. This field only resets on a PIPE reset or when the MAC writes to the "Sample Count Reset" bit in the Rx Margin Control1 register. This field is only required if the sampling rate is not reported in the PHY datasheet. If used, this field must be updated by the PHY every time the associated value changes; implementations may collapse multiple updates into a single write only to avoid creating a backlog of writes.

7.2.3 Address 2h: Rx Margin Status2

Table 7-28. Address 2h: Rx Margin Status2 (Sheet 1 of 2)

Bit	Default	Attribute	Required	Description
[7:6]	0h	N/A	N/A	Reserved

Table 7-28. Address 2h: Rx Margin Status2 (Sheet 2 of 2)

Bit	Default	Attribute	Required	Description
[7:0]	0h	Level	PCIe (optional) and USB4	**Error Count[7:0]:** This field is only required if errors do not happen in the data stream and thus an independent error sampler is implemented in the PHY. This field is used by the PHY to report actual bit errors to the MAC. This field can increment only when "Start Margin" is asserted. This field only resets on a PIPE reset or when the MAC writes to the "Error Count Reset" bit in the Rx Margin Control1 register. If used, this field must be updated by the PHY every time the associated value changes; implementations may collapse multiple updates into a single write to avoid creating a backlog of writes. For PCIe, the upper two bits are reserved and only the lower 6 bits are utilized. For USB4, the Error Count is 16 bits, refer to RX Margin Status3 in Section 7.2.8 for upper 8 bits.

7.2.4 Address 3h: Elastic Buffer Status

Table 7-29. Address 3h: Elastic Buffer Status

Bit	Default	Attribute	Required	Description
[7:1]	0h	N/A	N/A	Reserved
[0]	0h	1-cycle	PCIe (optional)	**Elastic Buffer Status:** The PHY sets this status bit to 1'b1 when it has completed its elastic buffer depth adjustment to the value specified in the Elastic Buffer Control register. **Note:** This field is not used in the SerDes architecture.

7.2.5 Address 4h: Elastic Buffer Location

Table 7-30. Address 4h: Elastic Buffer Location

Bit	Default	Attribute	Required	Description
[7:0]	0h	Level	PCIe (optional), and USB (optional)	**ElasticBufferLocation:** This field reflects the number of entries that are currently in the elastic buffer. Whenever the number of entries in the elastic buffer changes the PHY schedules an update to this register, with the frequency of update not to exceed that programmed in the ElasticBufferLocationUpdateFrequency field. **Note:** This field is not used in the SerDes architecture.

7.2.6 Address 5h: Rx Status0

Table 7-31. Address 5h: Rx Status0 (Sheet 1 of 2)

Bit	Default	Attribute	Required	Description
[7:1]	0h	N/A	N/A	Reserved
[1]	0h	1-cycle	PCIe (optional), USB (optional), and USB4 (optional)	**IORecalDone:** This field is set to "1" to indicate that an IORecal operation has successfully completed.

Table 7-31. Address 5h: Rx Status0 (Sheet 2 of 2)

Bit	Default	Attribute	Required	Description
[0]	0h	1-cycle	PCIe (optional)	**RxEqTrainDone:** This field is set to 1'b1 to indicate that receiver training is complete in response to an RxEqTraining request. This handshake is not applicable to RxEqTraining requests for USB as that is timer-based.

7.2.7 Address 6h: Rx Control0

Table 7-32. Address 6h: Rx Control0

Bit	Default	Attribute	Required	Description
[7:1]	0h	N/A	N/A	Reserved
[0]	0h	1-cycle	PCIe (optional), USB (optional), and USB4 (optional)	**PhyIORecalRequest:** The PHY sets this to "1" to indicate that the controller should enter Recovery and request of an Rx recalibration via the IORecal bit. The PHY advertises in its datasheet via the PhyRecalRequirement parameter whether this functionality is required.

7.2.8 Address 7h: Rx Margin Status3

Table 7-33. Address 7h: Rx Margin Status3

Bit	Default	Attribute	Required	Description
[7:0]	0h	Level	USB4	Error Count[15:8]: Refer to Rx Margin Status2 in Section 7.2.3 for the definition of the Error Count field and the lower 8 bits.

Table 7-34. Address 8h: Reserved

Bit	Default	Attribute	Required	Description
[7:0]	0h	N/A	N/A	Reserved

Table 7-35. Address 9h: Reserved

Bit	Default	Attribute	Required	Description
[7:0]	0h	N/A	N/A	Reserved

7.2.9 Address Ah: Rx Link Evaluation Status0

Table 7-36. Address Ah: Rx Link Evaluation Status0

Bit	Default	Attribute	Required	Description
[7:0]	0h	Level	PCIe, USB4, and DisplayPort Rx (optional)	**LinkEvaluationFeedbackFigureMerit[7:0]:** This field provides the PHY link equalization evaluation Figure of Merit Value. The value is encoded as an unsigned integer from 0 to 255. An encoding of 0 is the worst, and an encoding of 255 is the best. A PHY does not update this field if it does not provide link equalization evaluation feedback using the Figure of Merit format. For PCIe, this field is only used at the 8.0 GT/s, 16 GT/s, 32 GT/s, 64 GT/s,and 128 GT/s signaling rates. **Note:** The write_committed associated with an update to this field indicates that the RxEqEval has completed.

7.2.10 Address Bh: Rx Link Evaluation Status1

Table 7-37. Address Bh: Rx Link Evaluation Status1

Bit	Default	Attribute	Required	Description
[7:6]	0h	N/A	N/A	Reserved
[5:0]	0h	Level	PCIe (optional if FOM supported)	**LinkEvaluationFeedbackDirectionChange[5:0]:** This field provides the link equalization evaluation feedback in the direction change format. Feedback is provided for each coefficient: [1:0] C_{-1} [3:2] C_0 [5:4] C_1 The feedback value for each coefficient is encoded as follows: 00 - No change 01 – Increment 10 – Decrement 11 - Reserved A PHY does not update this field if it is does not provide link equalization evaluation feedback using the direction change format. **Note:** In 8.0 GT/s mode the MAC shall ignore the C_0 value and use the correct value per the PCIe specification. These signals are only used at the 8.0 GT/s, 16 GT/s, and 32 GT/s signaling rates. See Section 7.2.13 and Section 7.2.14 for 64 GT/s and 128 GT/s signaling rate. Note that C_{-1} and C_1 are encoded as the absolute value of the actual FIR coefficient and thus incrementing or decrementing either value refers to the magnitude of the actual FIR coefficient. For example, if C_{-1} is 000001b the FIR coefficient is negative one and a request to increment C_{-1} will increase it in the direction of 000002b that decreases the FIR coefficient. **Note:** The write_committed associated with an update to this field indicates that the RxEqEval has completed.

7.2.11 Address Ch: Rx Status4

Table 7-38. Address Ch: Rx Status4

Bit	Default	Attribute	Required	Description
[7:6]	0h	N/A	N/A	Reserved
[5:0]	0h	Level	PCIe	**LocalG5FS[5:0]** This field reflects the FS value for the PHY. These signals are only used by a PHY that requires dynamic preset coefficient updates. The FS value is valid for 32 GT/s. This field must be updated by the PHY before the first PhyStatus pulse after a rate change to 32 GT/s or in response to GetLocalPresetCoefficients when LocalPresetIndex[5:0] > 21 and <=32.

7.2.12 Address Dh: Rx Status5

Table 7-39. Address Dh: Rx Status5

Bit	Default	Attribute	Required	Description
[7:6]	0h	N/A	N/A	Reserved
[5:0]	0h	Level	PCIe	**LocalG5LF[5:0]** This field reflects the LF value for the PHY. This signal is only used by a PHY that requires dynamic preset coefficient updates. The LF value is valid for 32 GT/s. LocalG5LF[5:0] must be sampled whenever LocalG5FS[5:0] is sampled.

7.2.13 Address Eh: Rx Link Evaluation Status2

Table 7-40. Address Eh: Rx Link Evaluation Status2

Bit	Default	Attribute	Required	Description
[7]	0h	N/A	N/A	Reserved
[6:4]	0h	Level	PCIe	**LinkEvalFeedbackDirectionChange_Cminus1[2:0]:** This field provides link equalization evaluation feedback in the direction change format for coefficient C_{-1} when operating at a rate of 64 GT/s and 128 GT/s. See LinkEvalFeedbackDirectionChange_Cminus2[2:0] for more details.
[3]	0h	N/A	N/A	Reserved
[2:0]	0h	Level	PCIe	**LinkEvalFeedbackDirectionChange_Cminus2[2:0]:** This field provides link equalization evaluation feedback in the direction change format for coefficient C_{-2} when operating at a rate of 64 GT/s and 128 GT/s. For lower signaling rates, see LinkEvaluationFeedbackDirectionChange[5:0]. The feedback value is encoded as follows: 000 -- No change 001 -- Increment by 1 010 -- Decrement by 1 011 -- Increment by 2 100 -- Decrement by 2 101 -- Increment by 4 110 -- Decrement by 4 111 -- Reserved A PHY does not update this field if it does not provide link equalization evaluation feedback using the Direction Change format. Note that C_{-2}, C_{-1}, and C_1 are encoded as the absolute value of the actual FIR coefficient and thus incrementing or decrementing either value refers to the magnitude of the actual FIR coefficient. For example, if C_{-1} is 000001b the FIR coefficient is negative one and a request to increment C_{-1} will increase it in the direction of 000002b which decreases the FIR coefficient. **Note:** The write_committed associated with an update to this field indicates that the RxEqEval has completed.

7.2.14 Address Fh: Rx Link Evaluation Status3

Table 7-41. Address Fh: Rx Link Evaluation Status3

Bit	Default	Attribute	Required	Description
[7]	0h	N/A	N/A	Reserved
[6:4]	0h	Level	PCIe	**LinkEvalFeedbackDirectionChange_Cplus1[2:0]:** This field provides link equalization evaluation feedback in the direction change format for coefficient C_{+1} when operating at a rate of 64 GT/s and 128 GT/s. See LinkEvalFeedbackDirectionChange_Cminus2[2:0] for more details.
[3]	0h	N/A	N/A	Reserved
[2:0]	0h	Level	PCIe	**LinkEvalFeedbackDirectionChange_Czero[2:0]:** This field provides link equalization evaluation feedback in the direction change format for coefficient C_0 when operating at a rate of 64 GT/s and 128 GT/s. See LinkEvalFeedbackDirectionChange_Cminus2[2:0] for more details.

7.2.15 Address 10h: Rx Status6

Table 7-42. Address 10h: Rx Status 6

This register controls receiver functionality.

Bit	Default	Attribute	Required	Description
[7:1]	0h	N/A	N/A	Reserved
[0]	0h	Level	USB4 and DisplayPort	RxLaneReady: This field is set to "1" by the PHY to indicated that the receiver is enabled.

7.2.16 Address 400h: Tx Status0

Table 7-43. Address 400h: Tx Status0

Bit	Default	Attribute	Required	Description
[7:6]	0h	N/A	N/A	Reserved
[5:0]	0h	Level	PCIe	**LocalTxPresetCoefficients[5:0]/ LocalTxPresetCoefficients_Cminus1[5:0]:** For applicable protocols with the exception of PCIe at 64 GT/s and higher rates, this field is LocalTxPresetCoefficients[5:0] and defined as follows: This field forms part of LocalTxPresetCoefficients[17:0], which are the coefficients for the preset on the LocalPresetIndex[5:0] after a GetLocalPresetCoefficients request for 8 GT/s, 16 GT/s, or 32 GT/s. . LocalTxPresetCoefficients[17:0] is defined as follows: [5:0] C_{-1} [11:6] C_0 [17:12] C_{+1} The MAC will reflect these coefficient values on the TxDeemph bus when MAC wants to apply this preset. This field is only updated by a PHY that requires dynamic preset coefficient updates. For PCIe at 64 GT/s and higher rates, this field is LocalTxPresetCoefficients_Cminus1[5:0] and defined as follows: LocalTxPresetCoefficients_Cminus1[5:0] is part of a set of four coefficients for the preset on the LocalPresetIndex[5:0] after a GetLocalPresetCoefficients request for 64 GT/s or 128 GT/s. This field corresponds to precursor C_{-1}. See LocalTxPresetCoefficients_Cminus2[5:0], LocalTxPresetCoefficients_Czero[5:0], and LocalTxPresetCoefficients_Cplus1[5:0] for the other coefficients. The MAC will reflect these coefficient values on TxDeemph_Cminus1 when the MAC wants to apply this preset. This field is only updated by a PHY that requires dynamic preset coefficient updates.

7.2.17 Address 401h: Tx Status1

Table 7-44. Address 401h: Tx Status1

Bit	Default	Attribute	Required	Description
[7:6]	0h	N/A	N/A	Reserved
[5:0]	0h	Level	PCIe	**LocalTxPresetCoefficients[11:6]/ LocalTxPresetCoefficients_Czero[5:0]:** For applicable protocols (except for PCIe at 64 GT/s and higher rates), this field is LocalTxPresetCoefficients[11:6] and defined as follows: This field forms part of LocalTxPresetCoefficients[17:0]. See LocalTxPresetCoefficients[5:0] description for details. For PCIe at 64 GT/s and higher rates, this field is LocalTxPresetCoefficients_Czero[5:0] and defined as follows: LocalTxPresetCoefficients_Czero[5:0] is part of a set of four coefficients for the preset on the LocalPresetIndex[5:0] after a GetLocalPresetCoefficients request for 64 GT/s or 128 GT/s. This field corresponds to precursor C_0. See LocalTxPresetCoefficients_Cminus2[5:0], LocalTxPresetCoefficients_Cminus1[5:0], and LocalTxPresetCoefficients_Cplus1[5:0] for the other coefficients. The MAC will reflect these coefficient values on TxDeemph_Czero when the MAC wants to apply this preset. This field is only updated by a PHY that requires dynamic preset coefficient updates.

7.2.18 Address 402h: Tx Status2

Table 7-45. Address 402h: Tx Status2 (Sheet 1 of 2)

Bit	Default	Attribute	Required	Description
[7:6]	0h	N/A	N/A	Reserved

Table 7-45. Address 402h: Tx Status2 (Sheet 2 of 2)

Bit	Default	Attribute	Required	Description
[5:0]	0h	level	PCIe	**LocalTxPresetCoefficients[17:12]/ LocalTxPresetCoefficients_Cplus1[5:0]:** For applicable protocols (except for PCIe at 64 GT/s and higher rates), this field is LocalTxPresetCoefficients[17:12] and defined as follows: This field forms part of LocalTxPresetCoefficients[17:0]. See LocalTxPresetCoefficients[5:0] description for details. For PCIe at 64 GT/s and higher rates, this field is LocalTxPresetCoefficients_Cplus1[5:0] and defined as follows: LocalTxPresetCoefficients_Cplus1[5:0] is part of a set of four coefficients for the preset on the LocalPresetIndex[5:0] after a GetLocalPresetCoefficients request for 64 GT/s or 128 GT/s. This field corresponds to precursor C_{+1}. See LocalTxPresetCoefficients_Cminus2[5:0], LocalTxPresetCoefficients_Cminus1[5:0], and LocalTxPresetCoefficients_Czero[5:0] for the other coefficients. The MAC will reflect these coefficient values on TxDeemph_Cplus1 when the MAC wants to apply this preset. This field is only updated by a PHY that requires dynamic preset coefficient updates.

7.2.19 Address 403h: Tx Status3

Table 7-46. Address 403h: Tx Status3

Bit	Default	Attribute	Required	Description
[7:6]	0h	N/A	N/A	Reserved
[5:0]	0h	Level	PCIe	**LocalFS[5:0]:** This field reflects the FS value for the PHY. These signals are only used by a PHY that requires dynamic preset coefficient updates. The FS value is valid for 8 GT/s. This field shall be updated by the PHY before PhyStatus deasserts after RESET# and before the first PhyStatus pulse after a rate change to 8 GT/s or in response to GetLocalPresetCoefficients when LocalPresetIndex[5:0] < 11.

7.2.20 Address 404h: Tx Status4

Table 7-47. Address 404h: Tx Status4

Bit	Default	Attribute	Required	Description
[7:6]	0h	N/A	N/A	Reserved
[5:0]	0h	Level	PCIe	**LocalLF[5:0]:** This field reflects the LF value for the PHY. This signal is only used by a PHY that requires dynamic preset coefficient updates. The LF value is valid for 8 GT/s. LocalLF[5:0] must updated whenever LocalFS[5:0] is updated

7.2.21 Address 405h: Tx Status5

Table 7-48. Address 405h: Tx Status5

Bit	Default	Attribute	Required	Description
[7:6]	0h	N/A	N/A	Reserved
[5:0]	0h	Level	PCIe	**LocalG4FS[5:0]:** This field reflects the FS value for the PHY. These signals are only used by a PHY that requires dynamic preset coefficient updates. The FS value is valid for 16 GT/s. This field must be updated by the PHY before the first PhyStatus pulse after a rate change to 16 GT/s or in response to GetLocalPresetCoefficients when LocalPresetIndex[5:0] > 10 and <=21.

7.2.22 Address 406h: Tx Status6

Table 7-49. Address 406h: Tx Status6

Bit	Default	Attribute	Required	Description
[7:6]	0h	N/A	N/A	Reserved
[5:0]	0h	Level	PCIe	**LocalG4LF[5:0]:** This field reflects the LF value for the PHY. This signal is only used by a PHY that requires dynamic preset coefficient updates. The LF value is valid for 16 GT/s. LocalG4LF[5:0] must be sampled whenever LocalG4FS[5:0] is sampled.

7.2.23 Address 407h: Tx Status7

Table 7-50. Address 407h: Tx Status7

Bit	Default	Attribute	Required	Description
[7:6]	0h	N/A	N/A	Reserved
[5:0]	0h	Level	PCIe	**LocalTxPresetCoefficients_Cminus2[5:0]:** This field is part of a set of four coefficients for the preset on the LocalPresetIndex[5:0] after a GetLocalPresetCoefficients request for 64 GT/s or 128 GT/s. This field corresponds to precursor C_{-2}. See LocalTxPresetCoefficients_Cminus1[5:0], LocalTxPresetCoefficients_Czero[5:0], and LocalTxPresetCoefficients_Cplus1[5:0] for the other coefficients. For lower operational rates, see LocalTxPresetCoefficients[17:0]. The MAC will reflect these coefficient values on TxDeemph_Cminus2 when the MAC wants to apply this preset. This field is only updated by a PHY that requires dynamic preset coefficient updates.

7.2.24 Address 408h: Tx Status8

Table 7-51. Address 408h: Tx Status8

Bit	Default	Attribute	Required	Description
[7:6]	0h	N/A	N/A	Reserved
[5:0]	0h	Level	PCIe	**LocalG6FS[5:0]:** This field reflects the FS value for the PHY. These signals are only used by a PHY that requires dynamic preset coefficient updates. The FS value is valid for 64 GT/s. This field must be updated by the PHY before the first PhyStatus pulse after a rate change to 64 GT/s, or in response to GetLocalPresetCoefficients when LocalPresetIndex[5:0] > 32 and <=44

7.2.25 Address 409h: Tx Status9

Table 7-52. Address 409h: Tx Status9

Bit	Default	Attribute	Required	Description
[7:6]	0h	N/A	N/A	Reserved
[5:0]	0h	Level	PCIe	**LocalG6LF[5:0]:** This field reflects the LF value for the PHY. This signal is only used by a PHY that requires dynamic preset coefficient updates. The LF value is valid for 64 GT/s. LocalG6LF[5:0] must be sampled whenever LocalG6FS[5:0] is sampled.

7.2.26 Address 40Ah: Tx Status10

Table 7-53. Address 40Ah: Tx Status10

Bit	Default	Attribute	Required	Description
[7:1]	0h	N/A	N/A	Reserved
[0]	0h	Level	USB4 and DisplayPort	**TxLaneReady:** This field is set to "1" by the PHY to indicate that the transmitter is enabled.

7.2.27 Address 40Bh: Tx Status11

Table 7-54. Address 40Bh: TxStatus11

Bit	Default	Attribute	Required	Description
[7:6]	0	N/A	N/A	Reserved
[5:0]	0	Level	PCIe	**LocalG7FS[5:0]:** This field reflects the FS value for the PHY. These signals are only used by a PHY that requires dynamic preset coefficient updates. The FS value is valid for 128 GT/s. This field must be updated by the PHY before the first PhyStatus pulse after a rate change to 128 GT/s, or in response to GetLocalPresetCoefficients when LocalPresetIndex[5:0] > 44

7.2.28 Address 40Ch: Tx Status12

Table 7-55. Address 40Ch: TxStatus12

Bit	Default	Attribute	Required	Description
[7:6]	0	N/A	N/A	Reserved
[5:0]	0	Level	PCIe	**LocalG7LF[5:0]:** This field reflects the LF value for the PHY. These signals are only used by a PHY that requires dynamic preset coefficient updates. The LF value is valid for 128 GT/s. Local G7LF[5:0] must be sampled whenever LocalG7FS[5:0] is sampled.

7.2.29 Address 800h: Near End Loopback Status

This register is required only for PHYs that support the optional NELB feature.

Table 7-56. Address 800h: Near End Loopback Status (Sheet 1 of 2)

Bit	Default	Attribute	Required	Description
[7:2]	0h	N/A	N/A	Reserved
[1]	0h	Level	PCIe, USB, USB4, and SATA	**NELB Error:** This field is set to "1" to indicate an error if the requested operation through the NELB Control register (See Section 7.1.21) is not support. Specifically, the following two scenarios may result in an error status: 1) TheNELB position is not supported or 2) The N ELB enable bit is cleared when the PHY does not support exit from NELB via message bus and requires a PIPE reset instead.

Table 7-56. Address 800h: Near End Loopback Status (Sheet 2 of 2)

Bit	Default	Attribute	Required	Description
[0]	0h	Level	PCIe, USB, USB4, and SATA	**NELB State:** This field indicates whether the PHY is in NELB mode: 0 – Normal Operation 1 – NELB Mode

8 PIPE Operational Behavior

8.1 Clocking

There are three clock signals used by the PHY interface component. The first clock (**CLK**) is a reference clock that the PHY uses to generate internal bit rate clocks for transmitting and receiving data. The specifications for this signal are implementation-dependent and must be fully specified by vendors. The specifications may vary for different PHY operating modes. This clock may have a spread spectrum modulation that matches a system Reference Clock (REFCLK) (for example, the spread spectrum modulation could come from a REFCLK from the Card Electro-Mechanical Specification [CEMS]).

The second clock (**PCLK**) is an output from the PHY in "PCLK as PHY Output" mode and an input to each PHY lane in the "PCLK as PHY Input" mode and is the parallel interface clock used to synchronize data transfers across the parallel interface. This clock runs at a rate dependent on the **Rate**, **PCLK Rate**, and **PHY Mode** control inputs and data interface width. The rising edge of this clock is the reference point. This clock may also have a spread spectrum modulation. The CLK and PCLK must be sourced from the same reference clock and must contain the same clocking characteristics, that is, they can be mesochronous with each other.

The third clock (**MAX PCLK**) is a constant frequency clock with a frequency determined by the maximum signaling rate supported by the PHY and is only required in "PCLK as PHY Input" mode or in all modes for a PHY that supports PCIe at 8 GT/s or higher maximum speed. The Max PCLK value should be set to the maximum PCLK supported by the PHY.

The fourth clock (MacCLK) is an optional clock with support advertised by the PHY vendor parameter "MacCLK Support". This clock is independent of the data lanes and is specified using MacCLK lane signals.

8.1.1 Clocking Topologies

This section describes some clocking topologies that are compatible with PIPE. Figure 8-1 shows PCLK as a PHY output. This topology is only applicable for legacy PIPE implementations and is not supported for PCIe Gen5 designs, USB4 or DisplayPort. Figure 8-2 shows PCLK as a PHY input with the PLL residing in the PHY; the PHY provides a source for PCLK, in this case, MAX PCLK, that is mesochronous to the PHY's bit rate clock. Figure 8-3 shows the PCLK as a PHY input with the PLL that provides the PCLK source residing outside of the PHY; the reference clock for PLL that sources the bit rate clock and the PLL that provides the PCLK source must be the same. Figure 8-4 shows CLK as a PHY input with a single PLL that provides the source for PCLK as well as for the bit rate clock.

Figure 8-1. PCLK as PHY Output

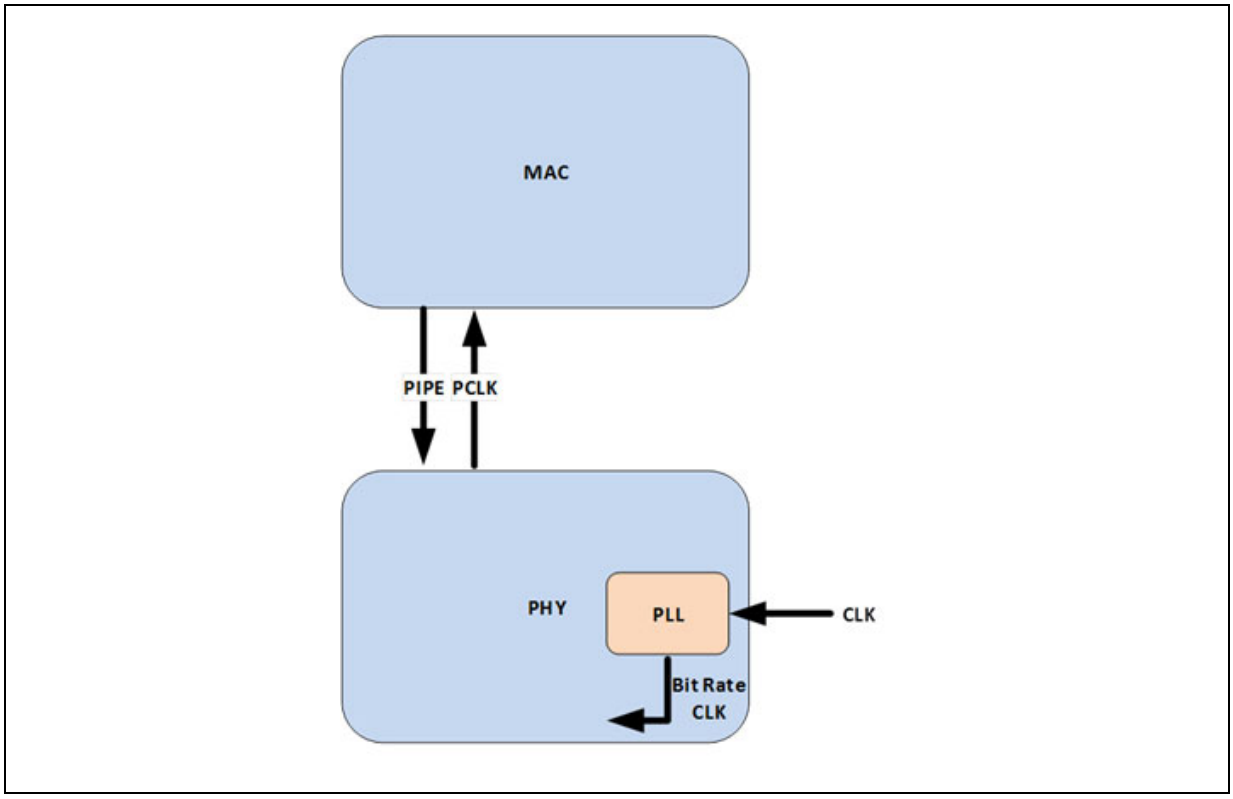


Figure 8-2. PCLK as PHY Input with a PHY-owned PLL

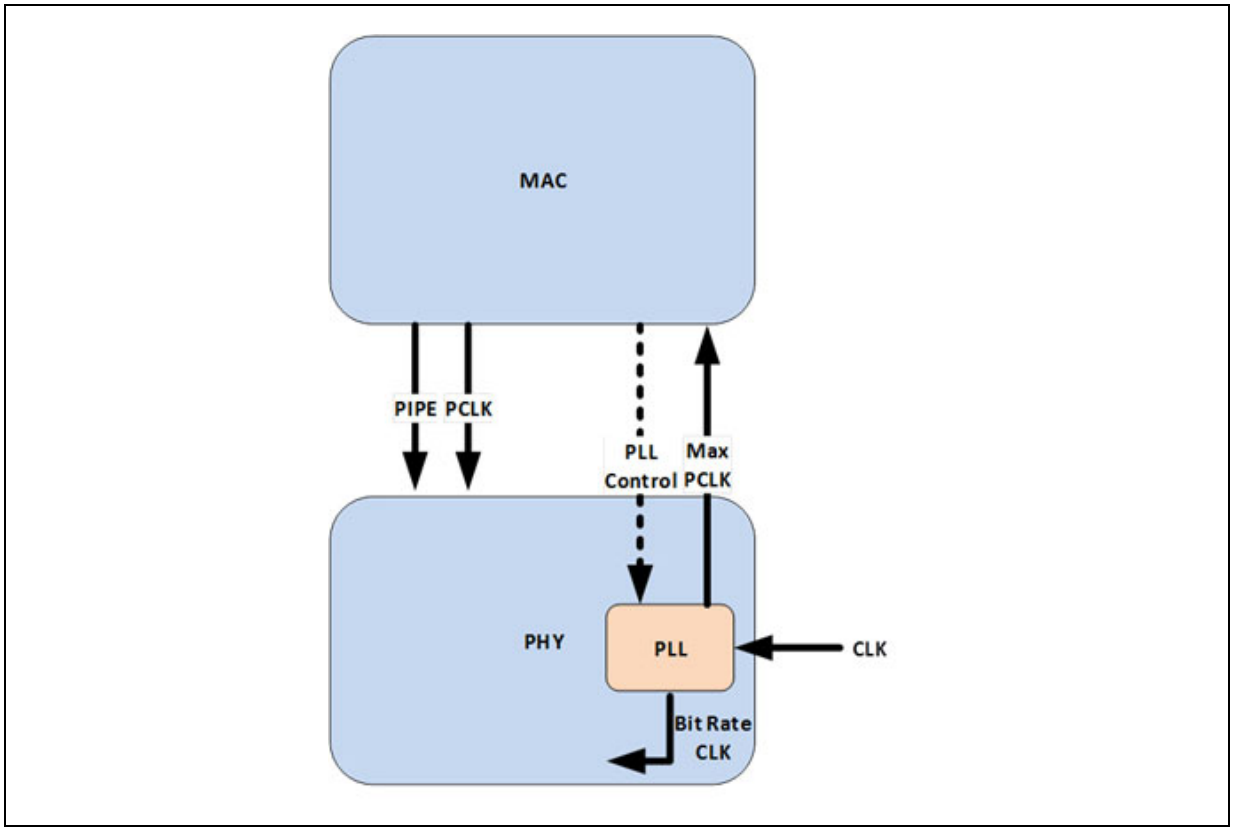


Figure 8-3. PCLK as PHY Input with an External PLL and PHY PLL

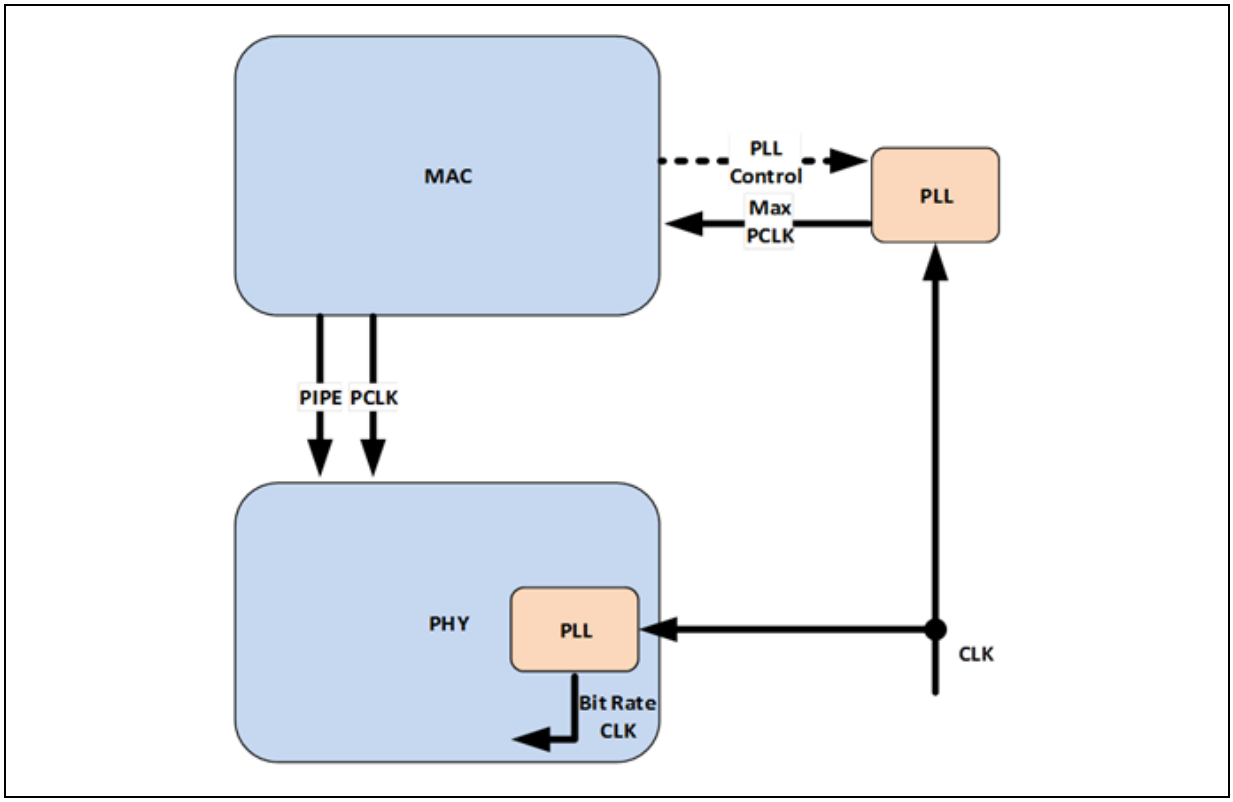
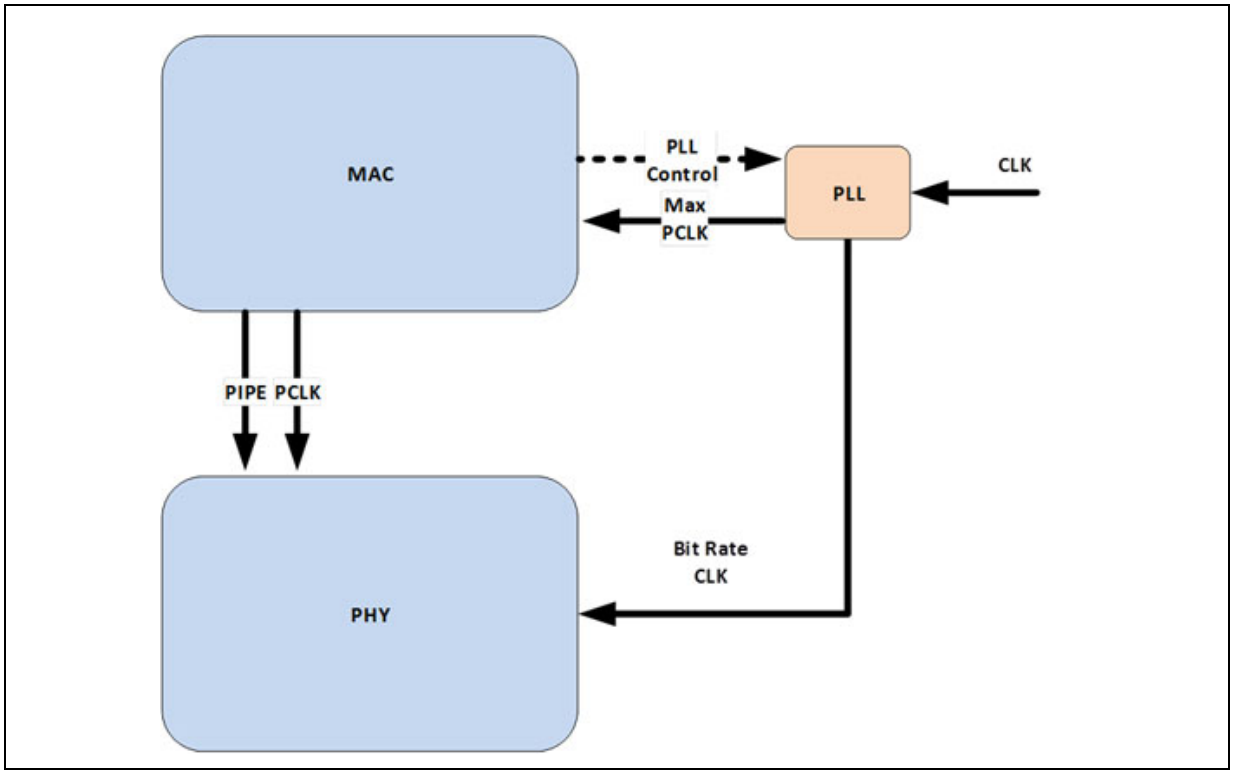


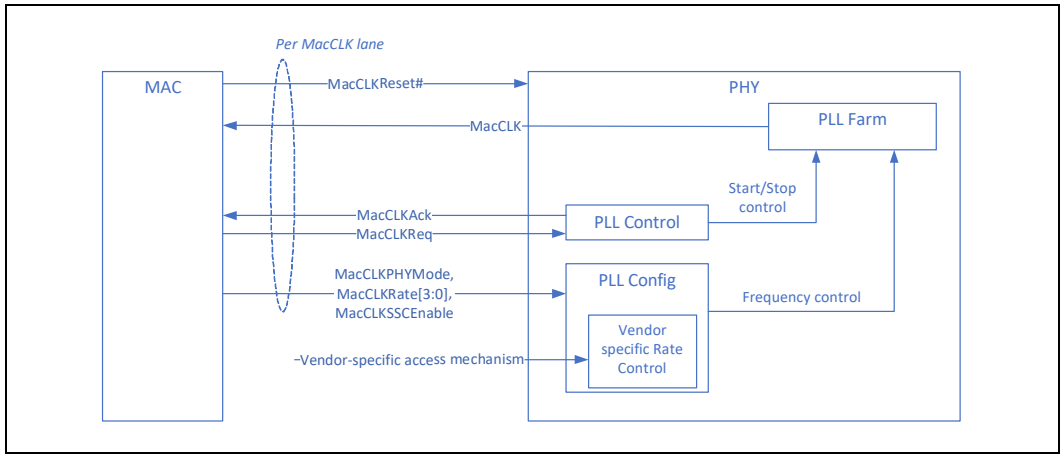
Figure 8-4. PCLK as PHY Input with External PLL



8.1.2 MacCLK Clocking Scheme

The MacCLK is an optional clock that is independent of data lanes. MacCLK frequency is controlled via a MacCLK lane. Figure 8-5 shows the signals associated with a single MacCLK lane. Multiple MacCLK lanes may be implemented; for example, DisplayPort and USB4 implementations are likely to implement two lanes. Some implementations may choose to map a MacCLK lane to a specific data lane with a direct correlation to PCLK. Each MacCLK lane consists of the following signals: MacCLKReset#, MacCLK, MacCLKReq, MacCLKSSCEnable, MacCLKAck, MacCLKPHYMode, and MacCLKRate.

Figure 8-5. MacCLK Lane

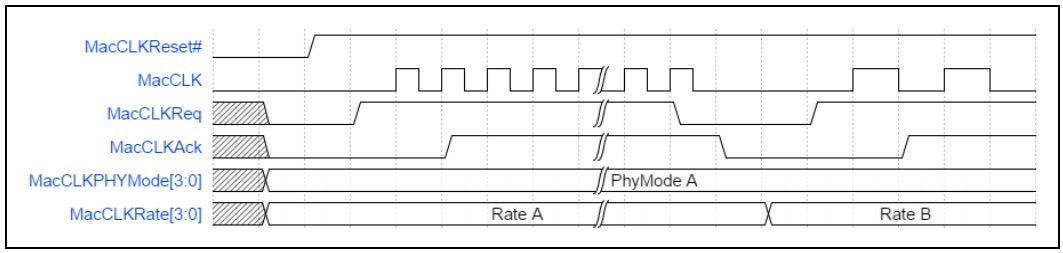


The following rules apply to each MacCLK lane:

- MacCLKPHYMode, MacCLKRate, MacCLKSSCEnable, MacCLKReq, MacCLKAck must be at a valid value at MacCLKReset# deassertion.
- MacCLKPHYMode must not change after MacCLKReset# deassertion.
- Once MacCLK is running, it must remain stable until MacCLKReq is deasserted.
- Signals that affect MacCLK frequency are sampled only on the rising edge of MacCLKReq.
- MacCLKRate is permitted to change only when MacCLKReq and MacCLKAck are deasserted.
- MACCLKSSCEnable is permitted to be dynamically asserted or deasserted while MacCLK is running. The PHY specifies the maximum time it takes to complete a SSC enable or disable via the MaxSSCEnableDisableTime parameter.
- MacCLKRate defines an override encoding that indicates that a vendor specific mechanism of specifying MacCLK frequency is used.

Figure 8-6 illustrates basic MacCLK operation that follows the above described rules. In this example, MacCLk is requested at an initial rate; subsequently, MacCLKReq and MacCLKAck deassert before a a new rate is requested.

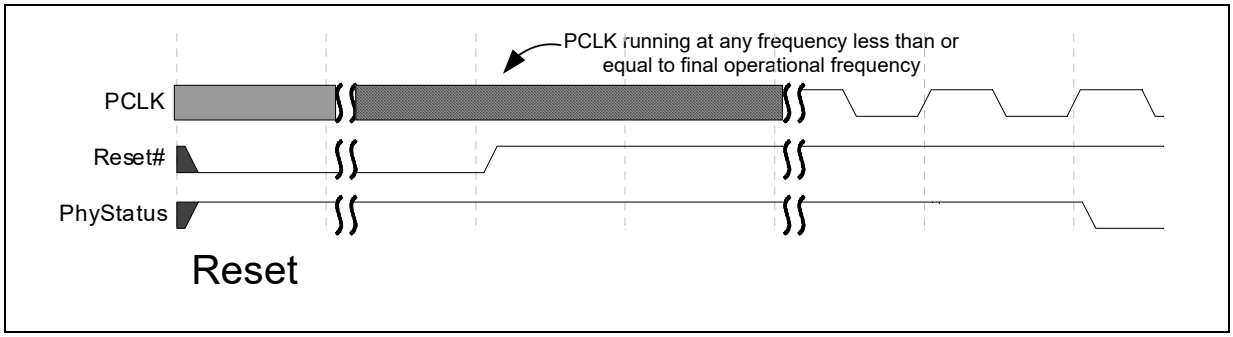
Figure 8-6. Basic MacCLK Lane Operation



8.2 Reset

When the MAC wants to reset the PHY (for instance, during the initial power-on), the MAC must hold the PHY in reset until the power and **CLK** to the PHY are stable. For PCLK as a PHY output, the PHY signals that **PCLK** and the MAX PCLK are valid (that is, the PCLK or the MAX PCLK has been running at its operational frequency for at least one clock) and the PHY is in the specified power state by the deassertion of **PhyStatus** after the MAC has stopped holding the PHY in reset. The MAC must not perform any operational sequences until the PhyStatus is returned for the Reset# deassertion. While **Reset#** is asserted, the MAC should have **TxDetectRx/Loopback** deasserted, **TxElecIdle** asserted, **TxCompliance** deasserted, **PowerDown** = P1 (PCIe mode) or **PowerDown** = P2 (USB Mode), or **PowerDown** set to the default value reported by the PHY (SATA Mode), **PHY mode** set to the desired PHY operating mode, **SerDesArch** configured for PIPE or SerDes architecture, **DP_Mode_Tx_Rx** set to desired mode, and **Rate** set to 2.5 GT/s signaling rate for a PHY in PCIe mode or 5.0 GT/s or 10 GT/s (highest supported) for a PHY in USB mode or any rate supported by the PHY in SATA mode. The state of **TxSwing** during the **Reset#** assertion is implementation specific. **RxTermination** assertion in USB mode is implementation specific.

Figure 8-7. Reset# Deassertion and PhyStatus for PCLK as PHY Output



8.3 Power Management

8.3.1 Power Management – PCIe Mode

The power management signals allow the PHY to minimize power consumption. The PHY must meet all timing constraints provided in the PCIe base specification regarding clock recovery and link training for the various power states. The PHY must also meet all terminations requirements for transmitters and receivers.

Four standard power states are defined: P0, P0s, P1, and P2. The P0 state is the normal operational state for the PHY. When directed from P0 to a lower power state, the PHY can immediately take whatever power saving measures are appropriate. A PHY is allowed to implement additional PHY-specific power states; the L1 substate support requires implementation of additional PHY-specific power states. A MAC may use any of the PHY specific states as long as the PCIe base specification requirements are still met.

In states P0, P0s, and P1, the PCLK is required to be kept operational. For all state transitions between these three states and any PHY-specific states where **PCLK** is operational, the PHY indicates successful transition into the designated power state by a single cycle assertion of **PhyStatus**. Transitions into and out of a P2 or a PHY-specific

state where **PCLK** is not operational are described in following sections. For all power state transitions, the MAC must not begin any operational sequences or further power state transitions until the PHY has indicated that the initial state transition is completed.

Mapping of PHY power states to states in the LTSSM found in the base specification are included as follows. A MAC may alternately use PHY-specific states as long as the base specification requirements are still met:

- P0 state: All internal clocks in the PHY are operational. P0 is the only state where the PHY transmits and receives PCIe signaling.

  P0 is the appropriate PHY power management state for most states in the LTSSM. Exceptions are listed in the following subsections for each lower power PHY state.

- P0s state: **PCLK** must stay operational. The MAC may move the PHY to this state only when the transmit channel is idle.

  P0s state can be used when the transmitter is in the **Tx_L0s.Idle** state.

  While the PHY is in either P0 or P0s power states, if the receiver is detecting an electrical idle, the receiver portion of the PHY can take appropriate power saving measures. The PHY must be capable of obtaining the bit and symbol lock within the PHY-specified time (N_FTS with or without common clock) upon resumption of signaling on the receive channel. This requirement only applies if the receiver had previously been bit and symbol-locked while in P0 or P0s states.

- P1 state: Selected internal clocks in the PHY can be turned off. The **PCLK** must stay operational. The MAC will move the PHY to this state only when both transmit and receive channels are idle. The PHY must not indicate a successful entry into the P1 (by asserting the **PhyStatus**) until PCLK is stable and the operating DC common mode voltage is stable and within specification (as per the base specification).

  P1 can be used for the **Disabled** state, all the **Detect** states, and the **L1.Idle** state (only if the L1 substates are not supported) of the LTSSM.

- P2 state: Selected internal clocks in the PHY can be turned off. The parallel interface is in an asynchronous mode and **PCLK** is turned off. P2 can be used for the **L1.Idle**, **L2.Idle**, and **L2.TransmitWake** states of the LTSSM.

PCLK as PHY Output: When transitioning into a P2, the PHY must assert **PhyStatus** before the **PCLK** is turned off and then deassert **PhyStatus** when the PCLK is fully off and when the PHY is in the P2 state. When transitioning out of the P2, the PHY asserts **PhyStatus** as soon as possible and leaves it asserted until after **PCLK** is stable.

PCLK as PHY Input: When transitioning into P2, the PHY must assert **PhyStatus** for one input PCLK cycle when it is ready for PCLK to be removed. When transitioning out of P2, the PHY must assert **PhyStatus** for one input PCLK cycle as soon as possible once it has transitioned to P0 and is ready for operation.

When transitioning out of a state that does not provide **PCLK** to another state that does not provide **PCLK**, the PHY asserts **PhyStatus** as soon as the PHY state transition is complete and leaves it asserted until the MAC asserts **AsyncPowerChangeAck**. Once the MAC asserts **AsyncPowerChangeAck** the PHY deasserts **PhyStatus**.

PHYs should be implemented to minimize power consumption during P2 as this is when the device will have to operate within the vaux power limits (as described in the PCIe base specification).

Figure 8-8. PCIe P2 Entry and Exit with PCLK as PHY Output

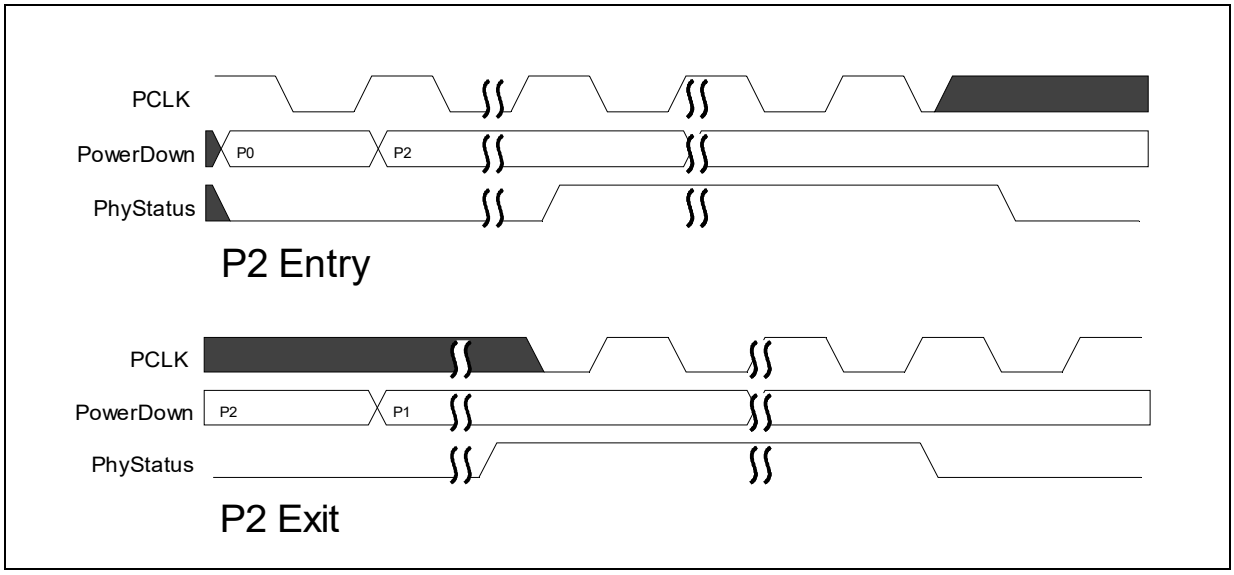
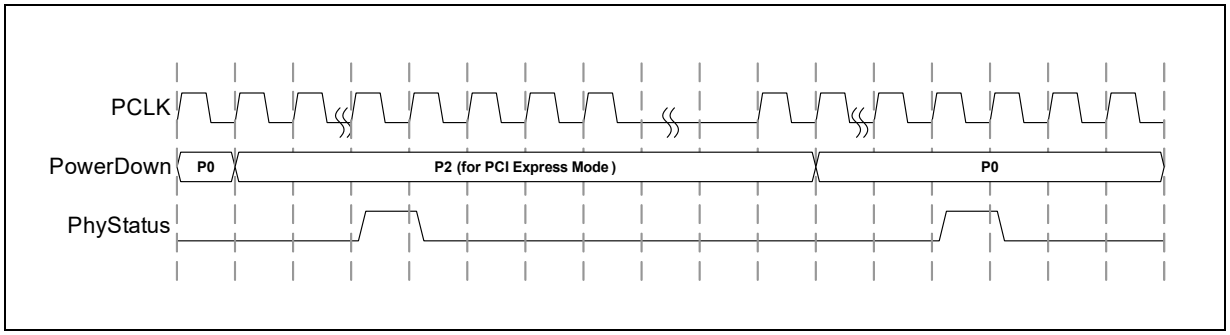


Figure 8-9. PCIe P2 Entry and Exit with PCLK as PHY Input



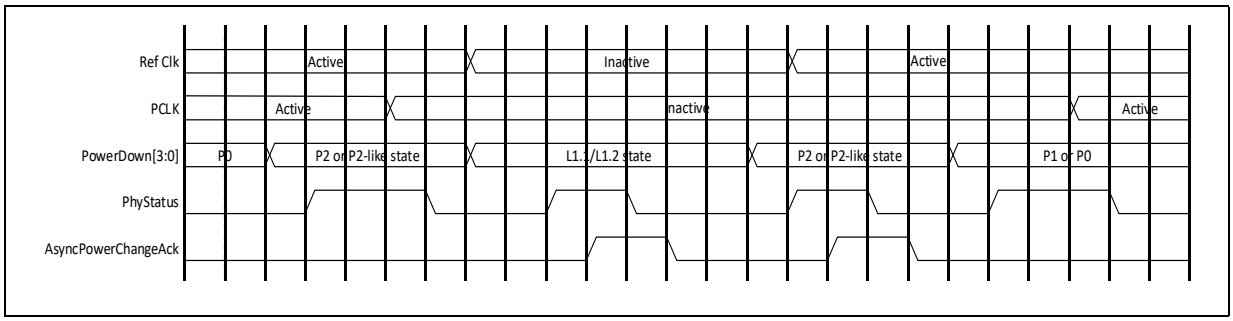
There is a limited set of legal power state transitions that a MAC can ask the PHY to make. Those legal transitions are: P0 to P0s, P0 to P1, P0 to P2, P0s to P0, P1 to P0, and P2 to P0. The base specification also describes what causes those state transitions.

Transitions to and from any pair of PHY power states including at least one PHY specific power state are also allowed by PIPE (unless otherwise prohibited). However, a MAC must ensure that PCIe specification timing requirements are met.

For L1 substate entry, the PHY must support a state where PCLK is disabled, the REFCLK can be removed, and the Rx electrical idle and Tx common mode are on; this can be P2 or a P2-like state. Figure 8-8 illustrates how a transition into and out of an L1 substate could occur. P2 or a P2-like state maps to L1.Idle; and the PhyStatus and AsyncPowerChangeAck signals are used as described earlier in this section. Alternatively, the PHY may implement a L1 substate management using a single PowerDown[3:0] encoding augmented with the RxEIDetectDisable and TxCommonModeDisable signals; the PowerDown state must remain constant across L1 substate transitions when this alternative mechanism is used. Using distinct PowerDown[3:0] encodings to define the L1 substates allows flexibility to specify different exit latencies; while using RxEIDetectDisable and TxCommonModeDisable, it may eliminate the need to do a handshake with AsyncPowerChangeAck. The PHY may

support either mechanism or both; this capability must be advertised in the PHY datasheet. The sideband mechanism of L1 substate management via RxEIDetectDisable and TxCommonModeDisable requires PCLK as PHY input mode.

Figure 8-10. L1 SubState Entry and Exit with PCLK as PHY Output



8.3.2 Power Management – USB Mode

The power management signals allow the PHY to minimize power consumption. The PHY must meet all timing constraints provided in the USB 3.2 specification regarding clock recovery and link training for the various power states. The PHY must also meet all termination requirements for transmitters and receivers.

Four power states are defined: P0, P1, P2, and P3. The P0 state is the normal operational state for the PHY. When directed from P0 to a lower power state, the PHY can immediately take whatever power saving measures are appropriate.

In the states P0, P1 and P2, the PCLK must be kept operational. For all state transitions between these three states, the PHY indicates successful transition into the designated power state by a single cycle assertion of **PhyStatus**. Transitions into and out of P3 are described in following subsections. For all power state transitions, the MAC must not begin any operational sequences or further power state transitions until the PHY has indicated that the initial state transition is completed.

Mapping of PHY power states to states in the LTSSM found in the USB specification are included in following subsections. A MAC may alternately use PHY-specific states as long as the base specification requirements are still met:

- P0 state: All internal clocks in the PHY are operational. P0 is the only state where the PHY transmits and receives USB signaling.
- P0 is the appropriate PHY power management state for all cases where the link is in U0 and all other link state except those listed in following entries for P1, P2, and P3.
- P1 state: **PCLK** must stay operational. The MAC will move the PHY to this state only when the PHY is transmitting idles and receiving idles. The P1 state can be used for the **U1** link state.
- P2 state: Selected internal clocks in the PHY can be turned off. **PCLK** must stay operational. The MAC will move the PHY to this state only when both transmit and receive channels are idle. The PHY must not indicate successful entry into P2 (by asserting **PhyStatus**) until PCLK is stable and the operating DC common mode voltage is stable and within specification (as per the base specification).
- P2 can be used for the **U2**, **Rx.Detect**, and **SS.Inactive**.

- P3 state: Selected internal clocks in the PHY can be turned off. The parallel interface is in an asynchronous mode and **PCLK** output is turned off.

PCLK as PHY output: When transitioning into P3, the PHY must assert the **PhyStatus** before **PCLK** is turned off and then deassert **PhyStatus** when PCLK is fully off and when the PHY is in the P3 state. When transitioning out of P3, the PHY asserts **PhyStatus** as soon as possible and leaves it asserted until after **PCLK** is stable.

PCLK as PHY input: When transitioning into P3, the PHY must assert **PhyStatus** for one input **PCLK** cycle when it is ready for PCLK to be removed. When transitioning out of P3, the PHY must assert **PhyStatus** for one input PCLK cycle as soon as possible once it has transitioned to P0 and is ready for operation.

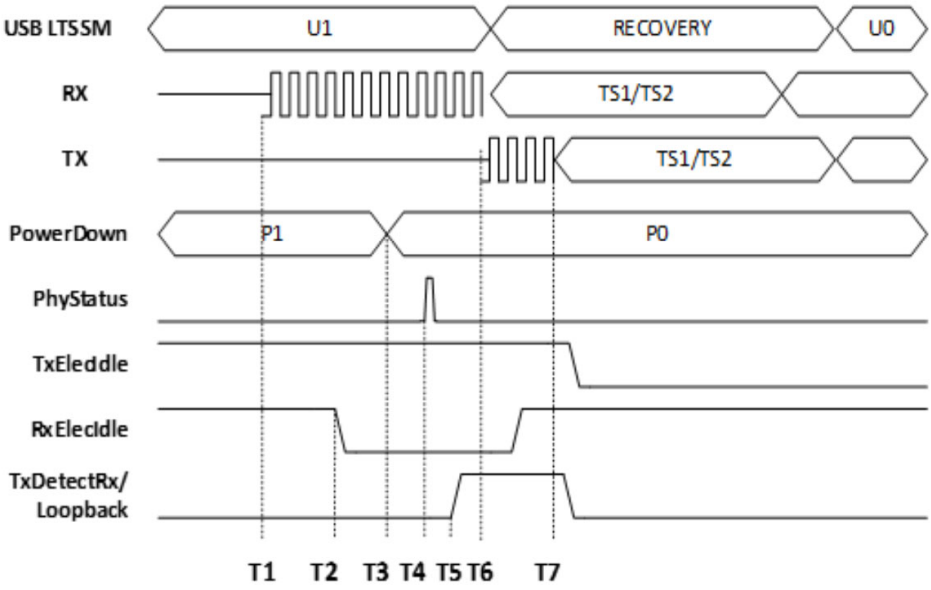
PHYs should be implemented to minimize power consumption during P3 as this is when the device will have to operate within power limits described in the USB 3.0 specification:

- The P3 state must be used in states **SS.disabled** and **U3**.
- There is a limited set of legal power state transitions that a MAC can ask the PHY to make. Referencing the main state diagram in the USB specification and the mapping of link states to PHY power states described in the preceding paragraphs, those legal transitions are: P0 to P1, P0 to P2, P0 to P3, P1 to P0, P2 to P0, P2 to P3, P3 to P0, P3 to P2, and P1 to P2. The base specification also describes what causes those state transitions.

U1 has strict exit latency requirements as described in the USB base specification. Figure 8-11 illustrates the timing requirements for PIPE signals associated with U1 exit with the following explanation:

- T2-T1: PHY decodes LFPS and reflects it through RxElecIdle (120 ns maximum)
- T4-T3: P1 to P0 transition latency (300 ns maximum)
- T6-T5: LFPS transmit latency (100 ns maximum)
- T7-T1: 0.6–0.9 us from USB Specification

Figure 8-11. USB U1 Exit



8.3.3 Power Management – USB4 Mode

The power management signals allow the PHY to minimize power consumption. The PHY must meet all timing and electrical constraints provided in the USB4 Specification regarding link states for the various power states.

Seven power states (P0, P0rx, P1, P2, P3, P4, and P0tx) are defined to fully optimize the power consumption of USB4 link states. The PHY must support all the power states that correspond to PowerDown encodings of 0 thru 3 specified in Table 8-1. The power states in the table corresponding to PowerDown encodings of 4 and 5 are optional. The PHY is permitted to implement additional PHY-specific power states. Table 8-1 provides suggested mappings of power states to USB4 link states, however, the MAC is permitted to choose alternate mappings using PHY implementation-specific states as long as the USB4 base specification requirements are met.

Table 8-1. USB4 PHY Power States (Sheet 1 of 2)

State	Description	PowerDown Encoding	Suggested USB4 Link State Mapping	Common Mode	LFPS Detector[1]	PLL	RxTerm	Total Latency[2]
P0	Fully active	0x0	CL0	ON	NA	ON	ON	NA
P0rx	Rx off Tx on	0x0 (with RxStandBy asserted)	CL0s.Rx	ON	ON	ON	ON	10 us
P1	Rx+Tx off Fast exit	0x1	CL1	ON	ON	ON	ON	10 us

Table 8-1. USB4 PHY Power States (Sheet 2 of 2)

State	Description	PowerDown Encoding	Suggested USB4 Link State Mapping	Common Mode	LFPS Detector[1]	PLL	RxTerm	Total Latency[2]
P2	Rx+Tx off med exit	0x2	CL1/2	ON	ON	ON/ OFF (see notes for P2)	ON	25 us
P3	Shutdown	0x3	CLd	ON/OFF (PHY implementation choice)	OFF	OFF	OFF	1ms
P4	Rx+Tx off slow exit	0x4	CL2	ON	ON	OFF	ON	100 us
P0tx	Tx off Rx on	0x5	CL0s.Tx	ON	OFF	ON	ON	10 us

1. LFPS detector enablement is controlled by the MAC in the case that the PHY implements the "RxEIDetectDisable" control wire.
2. The total latency is defined as the time interval to enter a state and exit back to P0. The intention is to ensure the return to a fully active state in the case of wake event race scenario. Budget includes turn OFF and ON of PLL when the power state allows. In the case that the PHY has completed the entry phase, this delay can be considered as the exit latency budget in which the PHY can take extra power optimizations.

Additional descriptions of each power state are:

- P0: Fully active the power state. The transmitter and receiver are ON.
- P0rx: Transmitter only power state. The receiver is OFF. Suitable for unidirectional traffic (for instance, transmitting the tunneled DisplayPort traffic).
- P1: Transmitter and receiver are OFF. This power state provides power savings while meeting low latency exit requirements.
- P2: Similar to P1 but with an increased total latency budget. The MAC is permitted to turn off the PLL in this state.
- P3: Shutdown mode, this is the lowest power consumption state. The link is virtually disconnected (CLd). Wake events are propagated via a sideband channel so common mode and LFPS detector are turned off. This state is suitable for deep sleep.
- P4: Transmitter and receiver are OFF with a total latency of up to 100 us delay. This state is suitable where higher exit latencies are tolerated (for example, storage devices).
- P0tx: Receiver only mode, the transmitter is OFF. Suitable for unidirectional traffic (for example, receiving tunneled DP traffic).

8.3.4 Power Management – SATA Mode

The power management signals allow the PHY to minimize power consumption. The PHY must meet all timing constraints provided in the SATA specification regarding clock recovery and link training for the various power states. The PHY must also meet all termination requirements for transmitters and receivers.

A minimum of five power states are defined, POWER_STATE_0 and a minimum of four additional states that meet minimum requirements defined in Section 6.1. The POWER_STATE_0 state is the normal operational state for the PHY. When directed from POWER_STATE_0 to a lower power state, the PHY can immediately take whatever power saving measures are appropriate.

For all state transitions between POWER_STATE_0 and lower power states that provide PCLK, the PHY indicates the successful transition into the designated power state by a single cycle assertion of **PhyStatus**. The PHY must complete transmitting all data transferred across the PIPE interface before the change in the PowerDown signals before an assertion of the **PhyStatus**. Transitions into and out-of-power state that does not provide PCLK are described as follows. For all power state transitions, the MAC must not begin any operational sequences or further power state transitions until the PHY has indicated that the initial state transition is completed. Power state transitions between two power states that do not provide PCLK are not allowed.

Mapping of PHY power states to link states in the SATA specification is MAC specific:

- POWER_STATE_0: All internal clocks in the PHY are operational. POWER_STATE_0 is the only state where the PHY transmits and receives SATA signaling.

  POWER_STATE_0 is the appropriate PHY power management state for most of the link states in the SATA specification. When transitioning into a power state that does not provide **PCLK**, the PHY must assert the **PhyStatus** before the **PCLK** is turned off and then deassert **PhyStatus** when PCLK is fully off and when the PHY is in the low power state. The PHY must leave the PCLK on for at least one cycle after asserting **PhyStatus**. For PCLK as PHY output, when transitioning out of a state that does not provide a PCLK, the PHY asserts **PhyStatus** as soon as possible and leaves it asserted until after **PCLK** is stable.

Transitions between any pair of PHY power states (except two states that do not provide PCLK) are allowed by PIPE. However, a MAC must ensure that SATA specification timing requirements are met.

8.3.5 Power Management - DisplayPort Mode

The power management signals allow the PHY to minimize power consumption. The PHY must meet all timing and electrical constraints provided in the DisplayPort specification regarding link states for the various power states.

Table 8-2 specifies required power states, which are defined to fully optimize the power consumption of eDP link states. A PHY is permitted to implement additional PHY specific power states. The MAC is permitted to use any of the PHY specific states as long as the Displayport specification requirements are still met. Transitions between any pair of PHY DisplayPort power states are allowed unless specifically prohibited.

The MAC controls the PLLs. The PLL is permitted to be turned off by the MAC in all the power states that allow the PLL to be off. Transition to a power state must not automaticaly result in the PLL being turned off.

Additional details of each power state are as follows:

- P0: Fully active power state. Transmitter is on.
- P1: Transmitter is off. This mode is suitable for low latency exit requirements.
- P2: Similar to P1 with a slightly larger total latency budget. The MAC is permitted to turn off the PLL in this mode as long as the latency requirements are met. The typical usage is expected to be during PSR and Panel Replay shallow sleep.

- P3: Shutdown mode. This is the lowest PHY power consumption state.
- P4: Transmitter is off. This mode is suitable for usages with higher exit latency budgets. A typical usage is during PSR and Panel Replay deep sleep.

Table 8-2. DisplayPort PHY Power States

State	Description	PowerDown Encoding	Common Mode	PLL[1]	Total Latency[2]
P0	Fully active	0x0	ON	ON	NA
P1	Tx off Fast exit	0x1	ON	ON	10 us
P2	Tx off Medium exit	0x2	ON	ON/OFF	25 us
P3	Shutdown	0x3	ON/OFF (Phy implementation specific)	ON/OFF	1 ms
P4	Tx off Slow exit	0x4	ON/OFF (Phy implementation specific	ON/OFF	150 us

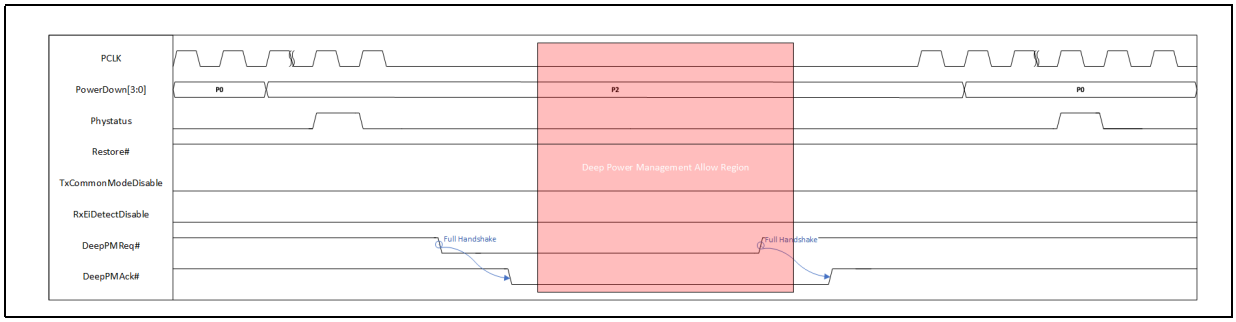
1. Where PLL is specified as ON/OFF, the MAC determines whether it should be on or off.
2. The total latency is defined as the time interval to enter a state and exit back to P0. The intention is to ensure the return to a fully active state in the case of wake event race scenario. Budget includes turn OFF and ON of PLL when the power state allows. In the case that the PHY has completed the entry phase, this delay can be considered as the exit latency budget in which the PHY can take extra power optimizations.

8.3.6 Asynchronous Deep Power Management

8.3.6.1 Deep Power Management Control Handshake Sequencing

The PIPE specification enables deep power management states during certain PowerDown states by defining a set of asynchronous handshake signals DeepPMReq# and DeepPMAck#. During deep power management states, the PHY is permitted to take appropriate actions to reduce power such as clock gating, power gating, or power rail removal. By implementing Active State Deep Power Management (ASDPM) mechanisms, for instance, through latency tolerance reporting or workload monitoring, the MAC determines when it can tolerate higher exit latencies and subsequently notifies the PHY that it is permitted to enter a deep power management state by asserting DeepPMReq#. The PHY acknowledges this request immediately by asserting DeepPMAck#; the actual PHY entry to a deep power management state occurs after the DeepPMAck# is asserted, and it is possible that PHY internal conditions may prevent entry from ever happening. Since DeepPMReq# and DeepPMAck# are asynchronous signals, the PCLK is permitted to remain gated during transitions into and out of the deep power management states. The MAC directs the PHY to exit its deep power management state by deasserting DeepPMReq#. Upon detecting deassertion of DeepPMReq#, the PHY must exit its deep power management state and then signal that the exit has occurred by deasserting DeepPMAck#. The MAC confirms that the PHY is not in a deep power management state before transitioning PowerDown states; this enables PowerDown to always be used in a synchronous manner when the PCLK is a PHY input. Figure 8-12 illustrates the handshake sequencing for entering and exiting deep power management.

Figure 8-12. DeepPMReq#/DeepPMAck# Handshake Sequencing

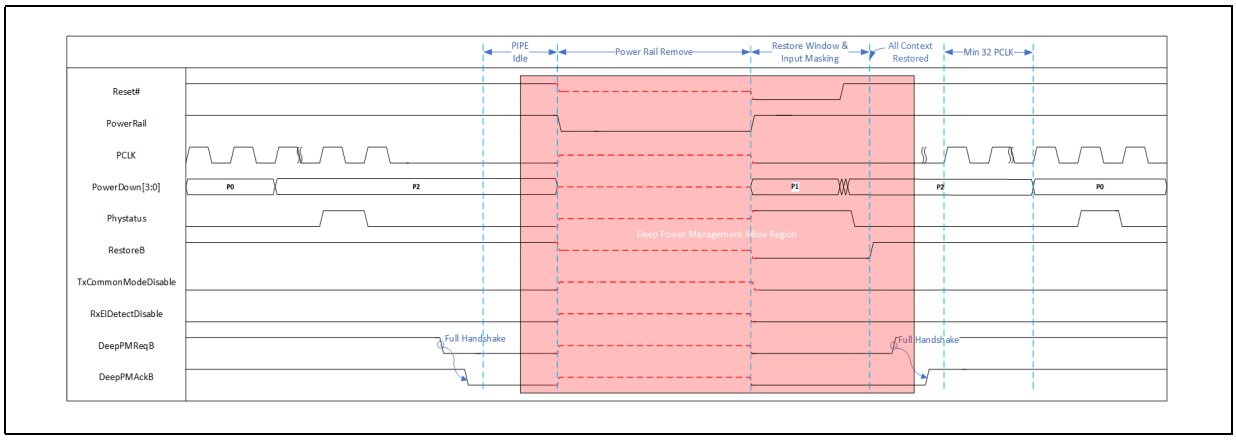


8.3.6.2 Power Removal and PHY Context Restoration after Power is Restored

Before power gating or a power rail removal in a deep power management state, a PHY may choose to save internal context to a location outside of the PHY as a power savings optimization. The MAC must guarantee that the PIPE interface is idle with no outstanding operations before the power removal. The mechanism for saving context is not part of the PIPE specification, however, the PIPE specification does define a restore window during which context is restored and specifies required MAC and PHY behavior during and immediately after the restore window.

The MAC notifies the PHY of entry to the restore window through the assertion of Restore#. The exit from the restore window is signaled via the deassertion of the Restore# signal. The Restore# must be asserted before a power rail ramp. During the restore window, all context that was saved off before a power gating or the power rail removal is restored; during this time, the MAC and the PHY must ignore any toggling of any input PIPE interface signals, except for PCLK and the Restore# signal. Upon an exit from the restore window, the MAC and the PHY must immediately resume monitoring of the input PIPE interface signals. After an exit from the restore window, the MAC and the PHY must wait for PCLK to become active and to toggle for a minimum of 32 cycles before toggling any PIPE signals that are synchronous to PCLK. Figure 8-13 illustrates the key requirements before a power removal and around the restore window.

Figure 8-13. PHY Context Restoration after the Power is Restored



8.4 Changing Signaling Rate, PCLK Rate, or Data Bus Width

8.4.1 PCIe Mode

The signaling rate of the link, the PCLK rate, or the data bus width can be changed only when the PHY is in the P0 or P1 power state and **TxElecIdle** and **RxStandby** (P0 only) are asserted. When the MAC changes the **Rate** signal, and the **Width** signal, and/or the **PCLK rate** signal in the PCLK as the PHY Output mode, the PHY performs the rate change and the width change and the PCLK rate change and signals its completion with a single cycle assertion of **PhyStatus**. The MAC must not perform any operational sequences, power state transitions, deassert **TxElecIdle** or **RxStandby**, or further signaling rate changes until the PHY has indicated that the signaling rate change has completed. The sequence is the same in PCLK as PHY input mode except that the MAC needs to know when the input PCLK rate or rate, or potentially width, can be safely changed. After the MAC changes rate and either PCLK_Rate, data width, or both, any change to the PCLK can happen only after the PclkChangeOk output has been driven high by the PHY. The MAC changes the input PCLK, if necessary, and then handshakes by asserting PclkChangeAck. The PHY responds by asserting PhyStatus for one input PCLK cycle and deasserts PclkChangeOk on the trailing edge of PhyStatus.

Note: PclkChangeOk is used by the PHY if the MAC changes PCLK_Rate and rate. The PHY datasheet indicates whether the same handshake is also required for every rate change.

Table 8-1 summarizes the handshake requirements. The MAC deasserts PclkChangeAck when PclkChangeOk is sampled low and may deassert **TxElecIdle** and/or **RxStandby** after PhyStatus is sampled high. There are instances where LTSSM state machine transitions indicate both a speed change or width or PCLK rate change and a power state change for the PHY. In these instances, the MAC must change (if necessary) the signaling rate, width and/or the PCLK rate before changing the power state.

Table 8-3. PclkChangeOK/PclkChangeAck Requirements

Rate	Width	PCLK Rate	PclkChangeOK /PclkChangeAck Handshake Required?
Stable	Don’t care	Don’t care	Not applicable
Change	Stable	Stable	Optional (parameter)
Change	Stable	Change	Required
Change	Change	Stable	Optional (parameter)
Change	Change	Change	Required

Some PHY architectures may allow a speed change and a power state change to occur at the same time as a rate, width, or PCLK rate change. If a PHY supports this, the MAC must change the rate, width, PCLK rate at the same PCLK edge that it changes the **PowerDown** signals. This can happen when transitioning the PHY from P0 to either P1 or P2 states. The completion mechanisms are the same as previously defined for the power state changes and indicate not only that the power state change is complete, but also that the rate, width, or rate change is complete.

8.4.2 USB Mode

The signaling rate of the link, PCLK rate, or the Data Bus Width can be changed only when the PHY is in the P0 or P2 power state and **TxElecIdle** and **RxStandby** are asserted. Any combination of at least two of the rate and width and PCLK rate, can be changed simultaneously. The MAC is not allowed to change only one of the three. When the MAC changes the **Rate** signal, and/or the **Width** signal, and/or the **PCLK rate** signal in PCLK as PHY Output mode, the PHY performs the rate change and/or the width change and/or the PCLK rate change and signals its completion with a single cycle assertion of **PhyStatus**. The MAC must not perform any operational sequences, power state transitions, deassert **TxElecIdle** or **RxStandby**, or further signaling rate changes until the PHY has indicated that the signaling rate change has completed. The sequence is the same in PCLK as PHY Input mode except the MAC needs to know when the input PCLK rate or Rate can be safely changed. After the MAC changes PCLK_Rate the change to the PCLK can happen only after the PclkChangeOk output has been driven high by the PHY. The MAC changes the input PCLK, and then handshakes by asserting PclkChangeAck. The PHY responds by asserting PhyStatus for one input PCLK cycle and deasserts PclkChangeOk on the trailing edge of PhyStatus. Note that PclkChangeOk is only used by the PHY if the MAC changes the PCLK_Rate or Rate.The MAC deasserts PclkChangeAck when PclkChangeOk is sampled low and may deassert an **TxElecIdle** and/or **RxStandby** after the PhyStatus is sampled high.

Some PHY architectures may allow a speed change and a power state change to occur at the same time as a rate, width, or rate change. If a PHY supports this, the MAC must change the rate, width, or rate at the same PCLK edge that it changes the **PowerDown** signals. This can happen when transitioning the PHY from P0 to either P2 or P3 states. The completion mechanisms are the same as previously defined for the power state changes and indicate not only that the power state change is complete, but also that the rate, width, or rate change is complete.

8.4.3 SATA Mode

The signaling rate of the link, PCLK rate, or the data bus width can be changed only when the PHY is in POWER_STATE_0 a prnd **TxElecIdle** and **RxStandby** are asserted, or in a low-power state where the PCLK is provided. When the MAC changes the **Rate** signal, and/or the **Width** signal, and/or the **PCLK rate** signal in PCLK as PHY Output mode, the PHY performs the rate change, the width change, or the PCLK rate change and signals its completion with a single cycle assertion of **PhyStatus**. The MAC must not perform any operational sequences, power state transitions, deassert **TxElecIdle**, or **RxStandby**, or further signaling rate or width changes until the PHY has indicated that the change has completed.

The sequence is the same in the PCLK as the PHY Input mode except the MAC needs to know when the input PCLK rate can be safely changed. After the MAC changes PCLK_Rate, the change to the PCLK can happen only after the PclkChangeOk output has been driven high by the PHY. The MAC changes the input PCLK, and then handshakes by asserting PclkChangeAck. The PHY responds by asserting PhyStatus for one input PCLK cycle and deasserts PclkChangeOk on the trailing edge of PhyStatus. Note that PclkChangeOk is only used by the PHY if the MAC changes PCLK_Rate. The MAC deasserts PclkChangeAck when PclkChangeOk is sampled low and may deassert **TxElecIdle** and/or **RxStandby** after PhyStatus is sampled high.

There are instances where conditions indicate both a speed change, width, and PCLK rate change and a power state change for the PHY. In such cases, the MAC must change the signaling rate, width, or rate, before changing the power state.

Some PHY architectures may allow a speed change and a power state change to occur at the same time as a rate, width, or rate change. If a PHY supports this, the MAC must change the rate, width, or rate at the same PCLK edge that it changes the **PowerDown** signals. The completion mechanisms are the same as previously defined for the power state changes and indicate not only that the power state change is complete, but also that the rate, width, or rate change is complete.

8.4.4 Fixed Data Path Implementations

The following figure shows the logical timings for implementations that change PCLK frequency when the MAC changes the signaling rate and PCLK is a PHY Output. Implementations that change the **PCLK** frequency when changing signaling rates must change the clock such that the time the clock is stopped (if it is stopped) is minimized to prevent any timers using **PCLK** from exceeding their specifications. In addition, during the clock transition period, the frequency of **PCLK** must not exceed the PHY's defined maximum clock frequency. The amount of time between when **Rate** is changed and the PHY completes the rate change is a PHY-specific value. These timings also apply to implementations that keep the data path fixed by using options that make use of the TxDataValid and RxDataValid signals.

Figure 8-14. Rate Change with Fixed Data Path

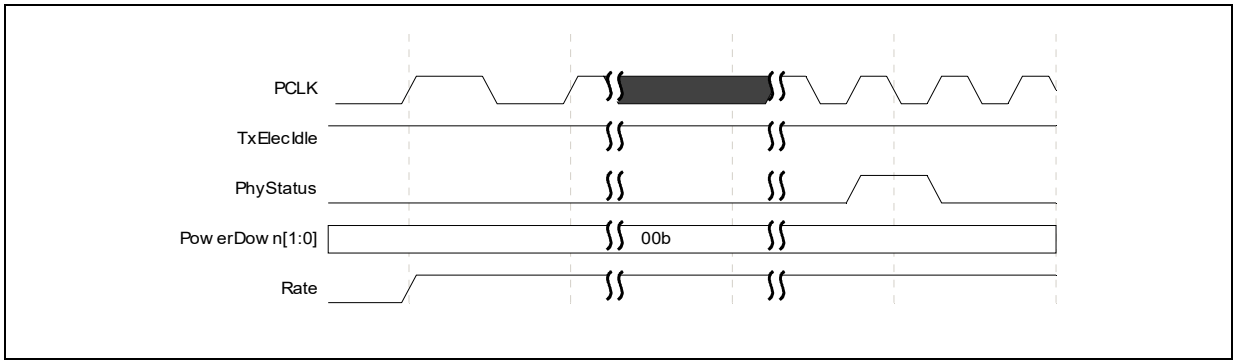
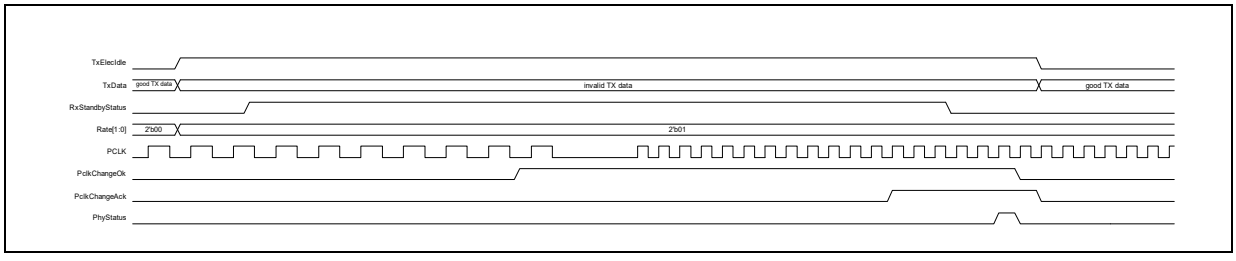


Figure 8-15 shows the logical timings for an implementation that changes the PCLK frequency when the MAC changes the signaling rate and PCLK is a PHY Input.

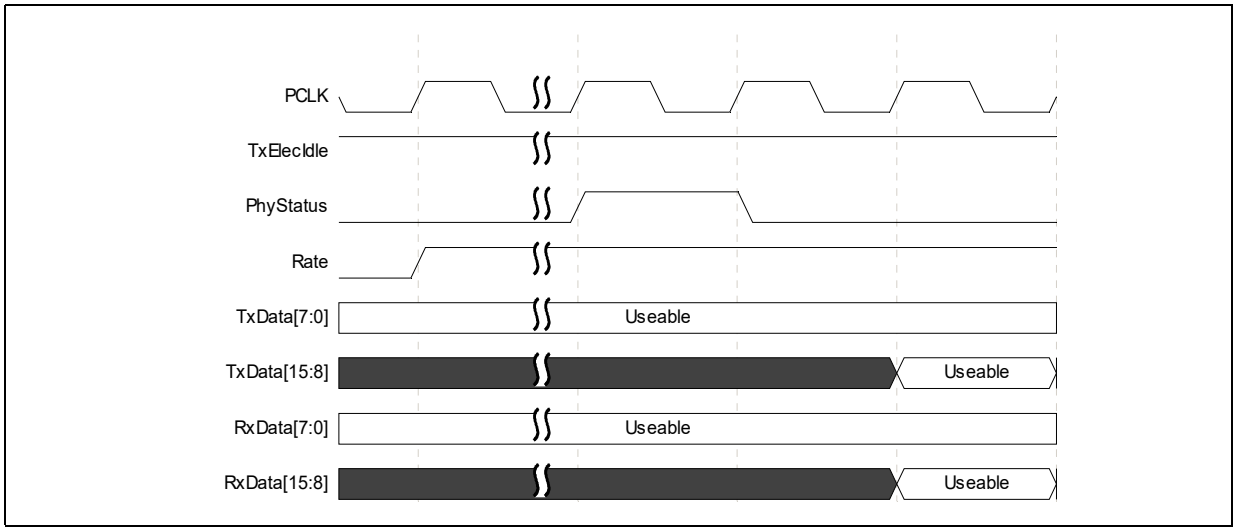
Figure 8-15. Change from PCIe 2.5 Gt/s to 5.0 Gt/s with PCLK as PHY Input.



8.4.5 Fixed PCLK Implementations

Figure 8-16 shows the logical timings for implementations that change the width of the data path for different signaling rates. PCLK may be stopped during a rate change. These timings also apply to fixed PCLK implementations that make use of the TxDataValid and RxDataValid signals.

Figure 8-16. Rate change with Fixed PCLK Frequency

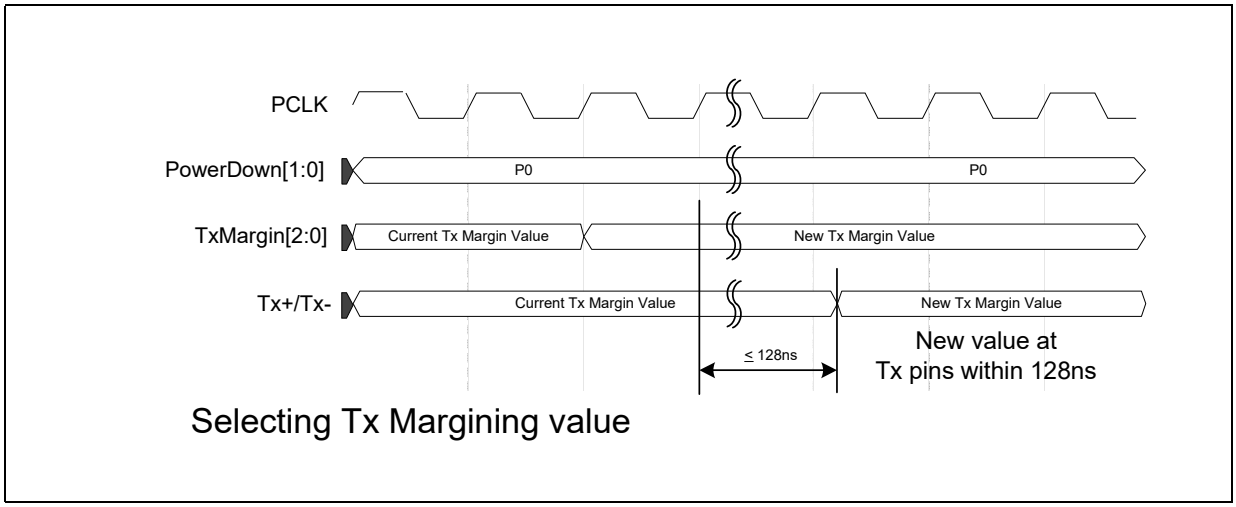


8.5 Transmitter Margining – PCIe Mode and USB Mode

While in the P0 power state, the PHY can be instructed to change the value of the voltage at the transmitter pins. When the MAC changes **TxMargin[2:0]**, the PHY must be capable of transmitting with the new setting within 128 ns.

There is a limited set of legal **TxMargin[2:0]** and **Rate** combinations that a MAC can select. See the PCIe base specification for a complete description of legal settings when the PHY is in PCIe mode. The USB specification for a complete description of the legal settings when the PHY is in USB mode.

Figure 8-17. Selecting Tx Margining Value



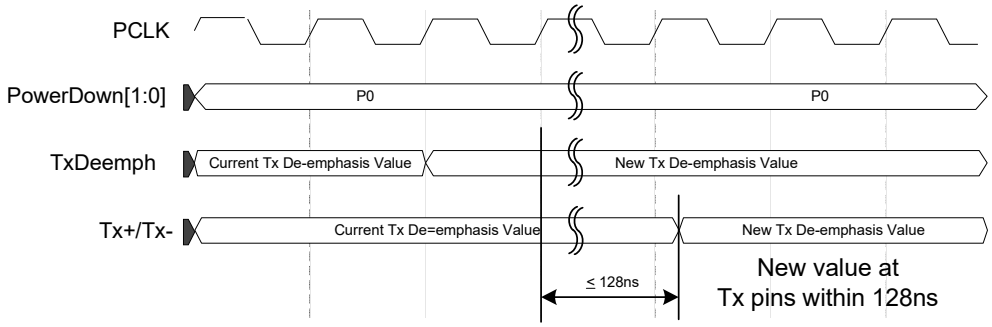
8.6 Selectable De-Emphasis – PCIe Mode

While in the P0 power state and transmitting at 5.0GT/s, 8.0 GT/s, 16 GT/s, 32 GT/s, 64 GT/s, or 128 GT/s, the PHY can be instructed to change the value of the transmitter equalization. When the signaling rate is 5.0 GT/s and the MAC changes **TxDeemph**, the PHY must be capable of transmitting with the new setting within 128 ns. When the signaling rate is 8.0 GT/s, 16 GT/s, 32 GT/s, 64 GT/s,or 128 GT/s and the MAC changes **TxDeemph**, the PHY must be capable of transmitting with the new setting within 256 ns.

There is a limited set of legal **TxDeemph** and **Rate** combinations that a MAC can select. See the PCIe base specification for a complete description.

The MAC must ensure that **TxDeemph** is selecting -3.5db whenever **Rate** is selecting 2.5 GT/s.

Figure 8-18. Selecting Tx De-Emphasis Value



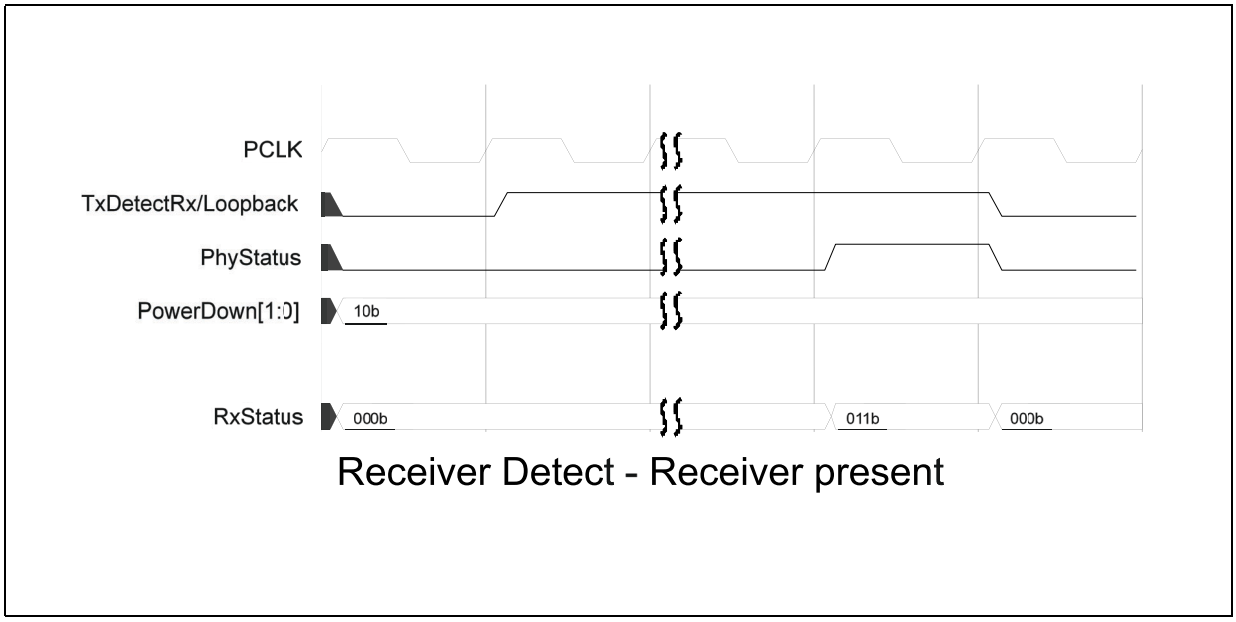
Selecting Tx De-emphasis value

8.7 Receiver Detection – PCIe Mode and USB Mode

While in the P1 or optionally P2 power state and PCIe mode or in the P2 or P3 power state and USB mode, the PHY can be instructed to perform a receiver detection operation to determine if there is a receiver at the other end of the link. Basic operation of receiver detection is that the MAC requests the PHY to do a receiver detect sequence by asserting **TxDetectRx/Loopback**. When the PHY has completed the receiver detect sequence, it asserts **PhyStatus** for one clock and drives the **RxStatus** signals to the appropriate code. After the receiver detection has completed (as signaled by the assertion of **PhyStatus**), the MAC must deassert **TxDetectRx/Loopback** before initiating another receiver detection, a power state transition, or signaling a rate change.

Once the MAC has requested a receiver detect sequence (by asserting **TxDetectRx/Loopback**), the MAC must leave **TxDetectRx/Loopback** asserted until after the PHY has signaled completion by the assertion of **PhyStatus**. When receiver detection is performed in USB mode with the PHY in P3 or PCIe in P2, the PHY asserts PhyStatus and signals the appropriate receiver detect value until the MAC deasserts TxDetectRx/Loopback.

Figure 8-19. Receiver Detect – Receiver Present

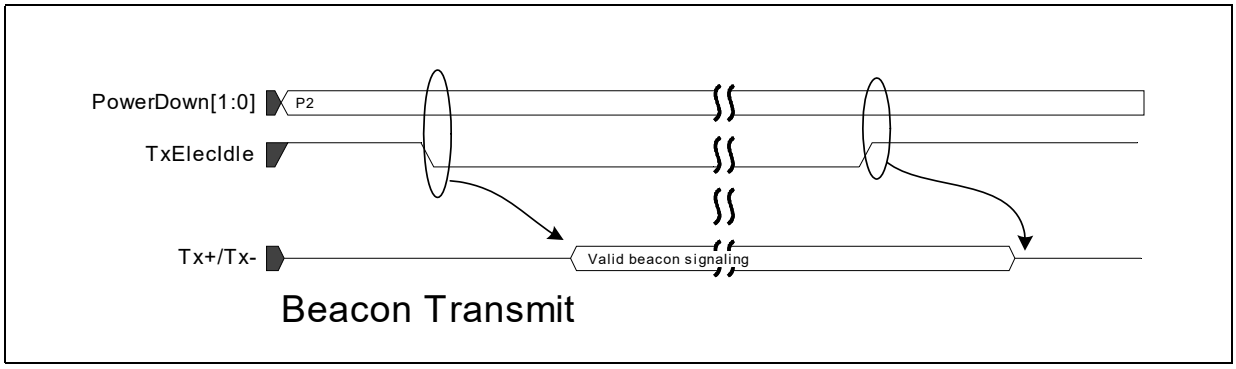


Detected Condition	RxStatus Code
Receiver not present	000b
Receiver present	011b

8.8 Transmitting a Beacon – PCIe Mode

When the PHY has been put in the P2 power state, and the MAC wants to transmit a beacon, the MAC deasserts **TxElecIdle** and the PHY should generate a valid beacon until **TxElecIdle** is asserted. The MAC must assert **TxElecIdle** before transitioning the PHY to P0.

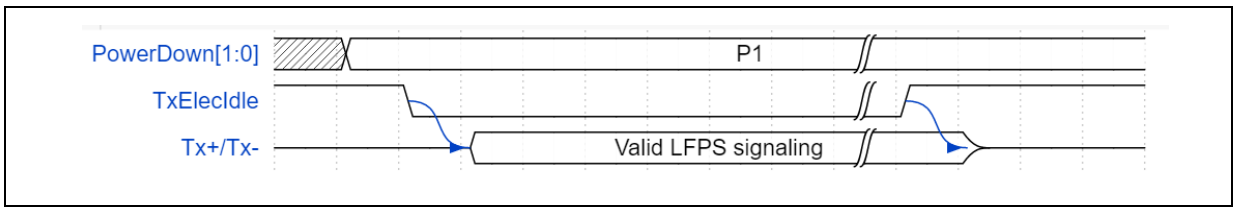
Figure 8-20. Beacon Transmit



8.9 Transmitting LFPS – USB Mode

When the PHY is in P1 and the MAC wants to transmit LFPS, the MAC deasserts **TxElecIdle** and the PHY should generate valid LFPS until **TxElecIdle** is asserted. The MAC must assert **TxElecIdle** before transitioning the PHY to P0. The length of time **TxElecIdle** is deasserted is varied for different events. When the PHY is in P0 and the MAC wants to transmit LFPS, the MAC must assert both **TxElecIdle** and **TxDetectRx/Loopback** for the desired duration of an LFPS burst. The PHY is required to complete a full LFPS period before transitioning to SuperSpeed data, and, as a consequence, it may drop SuperSpeed data if these requests overlap. This requirement does not apply to TxOnesZeros requests. See Chapter 6 in the USB 3.0 specification for more details.

Figure 8-21. LFPS Transmit

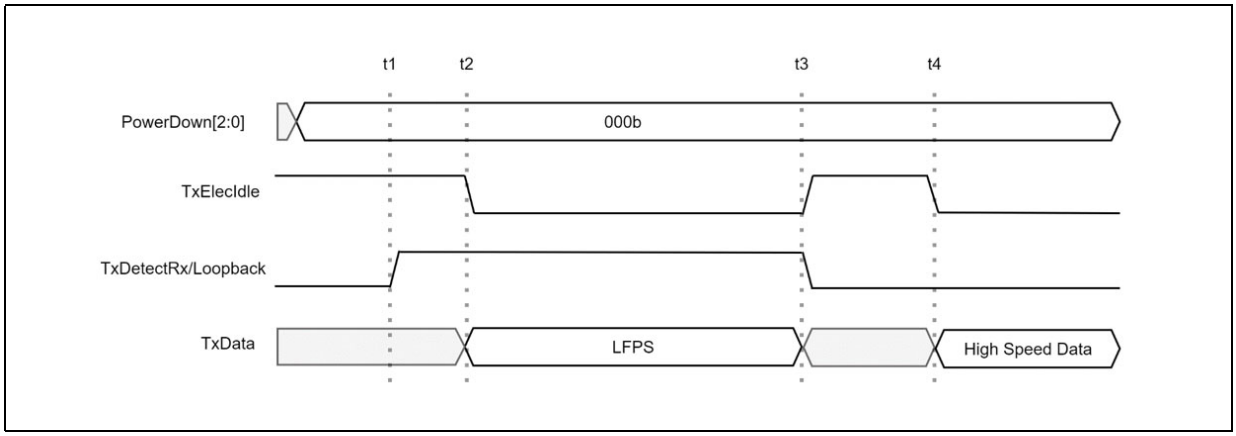


8.10 Transmitting LFPS – USB4 and DisplayPort Modes

By default, the PHY is responsible for transmitting LFPS. See Section 8.24 for relevant PIPE control signals. The MAC can configure the PHY to allow the MAC to transmit LFPS on the parallel data interface TxData by setting the MacTransmitLFPS field in the PHY Common Control0 register prior to transitioning the link to P0 PowerDown state. The advantage of enabling the MAC to generate LFPS is that it provides easier timing control for switchover to high speed data at a clean LPFS cycle boundary. This section describes the sequence required for the MAC to use the parallel data interface to transmit LFPS.

Figure 8-22 illustrates the requirements that must be adhered to for the MAC to transmit LFPS. When the MAC wants to transmit LFPS, it must inform the PHY in advance (t1) by asserting TxDetectRxLoopback while TxElecIdle is asserted; this allows the PHY to make any internal transmitter adjustments necessary to meeting LFPS electrical requirements. The MAC must deassert TxElecIdle (t2) when it starts transmitting LFPS. When the MAC wants to resume transmitting regular high-speed data, it must inform the PHY in advance (t3) by asserting TxElecIdle and deasserting TxDetectRx/Loopback; this allows the PHY to make any internal transmitter adjustments necessary for high-speed data transmission. High-speed data transmission resumes when TxElecIdle deasserts (t4).

Figure 8-22. LFPS Transmit for USB4 and DisplayPort Modes



t1 – MAC informs the PHY that the MAC plans to transmit LFPS

t2 – MAC starts transmitting LFPS over the parallel data interface

t3 – MAC stops transmitting LFPS over the parallel data interface

t4 – MAC starts transmitting High Speed data over parallel data interface

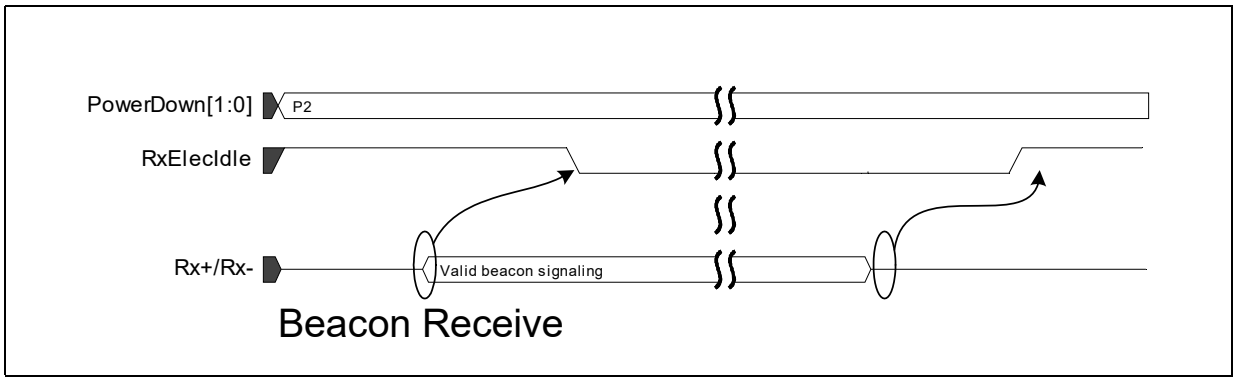
The transitional time window (t2-t1) allows the PHY time to adjust any parameters necessary for transmitting LFPS. The minimum time and maximum time must be specified in the PHY datasheet (MinTimeBeforeLFPS and MaxTimeBeforeLFPS) for each protocol.

The transitional time window (t4-t3) returns the link to Electrical Idle to allow the PHY time to restore parameters necessary for high-speed transmission. The minimum time must be specified in the PHY datasheet (MinTimeEIAfterLFPS) for each protocol.

8.11 Detecting a Beacon – PCIe Mode

The PHY receiver must monitor at all times (except during reset or when RxEIDetectDisable is set) for electrical idle. When the PHY is in the P2 power state, and **RxElecIdle** is deasserted, then a beacon is being detected.

Figure 8-23. Beacon Receive

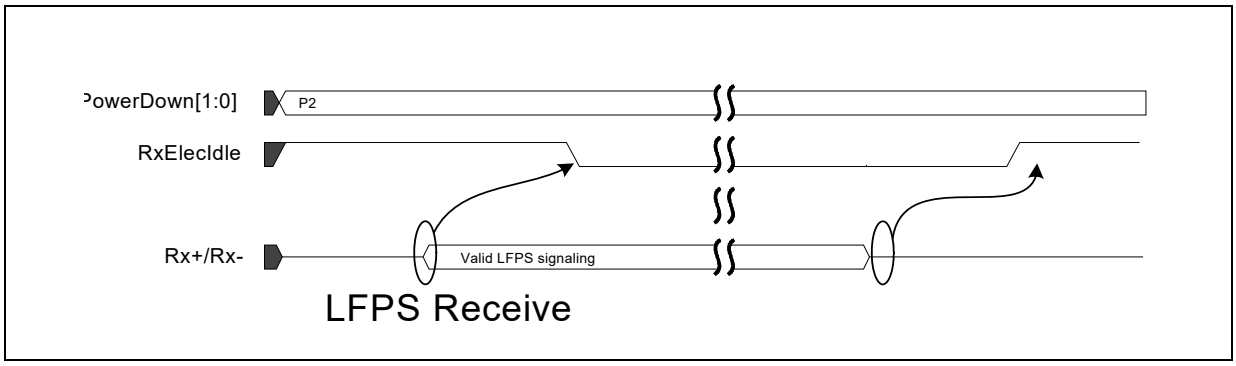


8.12 Detecting Low Frequency Periodic Signaling – USB Mode

The PHY receiver must monitor at all times (except during reset, when Rx terminations are removed, or when RxEIDetectDisable is set) for LFPS. When the PHY is in the P0, P1, P2, or P3 power state, and **RxElecIdle** is deasserted, then LFPS is being detected. The length of time **RxElecIdle** is deasserted indicates the length of time Low Frequency Periodic Signaling is detected. See to Chapter 6 in the USB 3.0 specification for more details on the length of LFPS for various events.

The PHY needs to differentiate LPFS received for Ping from Exit LFPS. When the PHY receives LFPS for up to two cycles only, it should deassert RxElecIdle for a maximum of 200 ns. For U1, there is a strict latency requirement for a USB controller to detect and respond back as defined in the USB specification Chapter 6 LPFS section. The PHY should not take more than 120 ns to deassert RxElecIdle after detecting LFPS in P0 and P1, and P2. For P3, the PHY is allowed to take us to 10us to deassert RxElecIdle.

Figure 8-24. LFPS Receive



8.13 Detecting Low Frequency Periodic Signaling in USB4 Mode

The PHY receiver must monitor for LFPS at all times when RxEIDetectDisable is clear. When the PHY is in P0, P1, P2 or P4 power state, it must deassert RxElecIdle when LFPS is detected. The length of time RxElecIdle is deasserted indicates the length of time LPFS is detected.

8.14 Clock Tolerance Compensation

Note: **This section is not applicable to SerDes architecture.**

The PHY receiver contains an elastic buffer used to compensate for differences in frequencies between bit rates at the two ends of a Link. The elastic buffer must be capable of holding enough symbols to handle worst case differences in frequency and worst-case intervals between symbols that can be used for rate compensation for the selected PHY mode.

Two models are defined for the elastic buffer operation in the PHY. The PHY may support one or both models. The Nominal Empty buffer model is only supported in PCIe, USB, or SATA Mode.

For the Nominal Empty buffer model, the PHY attempts to keep the elasticity buffer as close to empty as possible. In the Nominal Empty mode, the PHY uses the RxDataValid interface to tell the MAC when no data is available. The Nominal Empty buffer model provides a smaller worst case and average latency than the Nominal Half Full buffer model, but it requires the MAC to support the RxDataValid signal. The PHY removes all SKP symbols in Nominal Empty buffer mode.

For the Nominal Half Full buffer model, the PHY is responsible for inserting or removing SKP symbols, ordered sets, or ALIGNs in the received data stream to avoid elastic buffer overflow or underflow. The PHY monitors the receive data stream, and when a Skip ordered set or ALIGN is received, the PHY can add or remove one SKP symbol (PCIe Mode at 2.5 or 5 GT/s), four SKP symbols (PCIe Mode at 8 GT/s, 16 GT/s, or 32 GT/s), one SKP ordered set (USB Mode at 5 GT/s), or one ALIGN from each SKP or ALIGN as appropriate to manage its elastic buffer to keep the buffer as close to half full as possible. In USB mode at 5 GT/s, the PHY must only add or remove SKP ordered sets. In USB mode at 10 GT/s, the PHY must only add or remove multiples of four SKP symbols. Whenever the SKP symbols or an ordered set is added to or removed, the PHY will signal this to the MAC using the **RxStatus[2:0]** signals. These signals have a non-zero value for one clock cycle and indicate whether an SKP symbol or ordered set was added to or removed from the received SKP ordered sets. For PCIe, the timing of **RxStatus[2:0]** assertion depends on the operational rate since SKP ordered sets are encoded differently in 8b/10b mode versus 128/130b mode. In PCIe mode at 2.5 or 5 GT/s, **RxStatus[2:0]** must be asserted during the clock cycle when the COM symbol of the SKP ordered set is moved across the parallel interface. In PCIe mode at 8, 16 GT/s or 32 GT/s, **RxStatus[2:0]** must assert anytime between and including the start of the SKP ordered set and the SKP_END symbol. In SATA mode, whenever an ALIGN symbol is added or removed, the PHY will signal this to the MAC using the **RxStatus[2:0]** signals. These signals have a non-zero value for one clock cycle and indicate whether an ALIGN was added or removed. **RxStatus** must be asserted during the clock cycle when the first symbol of the added ALIGN is moved across the parallel interface.

In PCIe mode, the rules for operating in Nominal Empty buffer mode are as follows:

- Use of the RxDataValid is required
- All SKP symbols of SOS are removed (8b/10b SKP or 128/130 AA)
- When an empty condition happens (caused by clock drift or SOS removal):
  - RxValid must remain high:
    - RxValid should only be dropped for symbol alignment loss or block alignment loss
  - RxDataValid must be deasserted
  - RxStatus must be 0
- EB full can still occur and is considered an error
- Notification of an SOS coming through the EB must be reported in the following manner:
  - 8b/10b: COM of SOS must be passed with RxStatus = SKP removed (010), SKP symbols dropped
  - 128/130: Start of SOS block, with first byte SKP_END or SKP_END_CTRL, must be passed with RxStatus = SKP Removed (010), all AA SKP symbols dropped
- The EB is permitted to start RxDataValid as soon as data is available, but should never assert faster than the usual RxDataValid rate:
  - That is: rate=1, width=2, pclk_rate=2, RxDataValid should never assert for two consecutive pclk cycles

— That is: rate=1, width=2, pclk_rate=3, RxDataValid assertions must always have at least 3 pclk cycles of de-assertion between them

- Example of valid optimization by EB:
  — Rate=1, width=2, pclk_rate=3
  — RxDataValid (t=0, t=1, and so forth, E=EB Empty): 1000100010001000EE100010001
  — Vs. non-optimized: 10001000100010000EE00100010001
  — Non-optimized design builds EB depth in-order to maintain RxDataValid fixed cycle rate

In USB mode for the Nominal Empty buffer model the PHY attempts to keep the elasticity buffer as close to empty as possible. This means that the PHY will be required to insert SKP ordered sets into the received data stream when no SKP ordered sets have been received, unless the RxDataValid signal is used. The Nominal Empty buffer model provides a smaller worst case and average latency than the Nominal Half Full buffer model, but it requires the MAC to support receiving SKP ordered sets any point in the data stream.

In SATA mode for the Nominal Empty buffer model the PHY attempts to keep the elasticity buffer as close to empty as possible. In Nominal Empty mode the PHY uses the RxDataValid interface to tell the MAC when no data is available. The Nominal Empty buffer model provides a smaller worst case and average latency than the Nominal Half Full buffer model, but it requires the MAC to support the RxDataValid signal.

It is recommended that a PHY and MAC support the Nominal Empty buffer model in USB mode using the RxDataValid signal. The alternative of inserting SKPs in the data stream when no SKPs have been received is not recommended. The following figure shows a sequence where a PHY operating in PCIe Mode added an SKP symbol in the data stream.

Figure 8-25. Clock Correction – Add an SKP

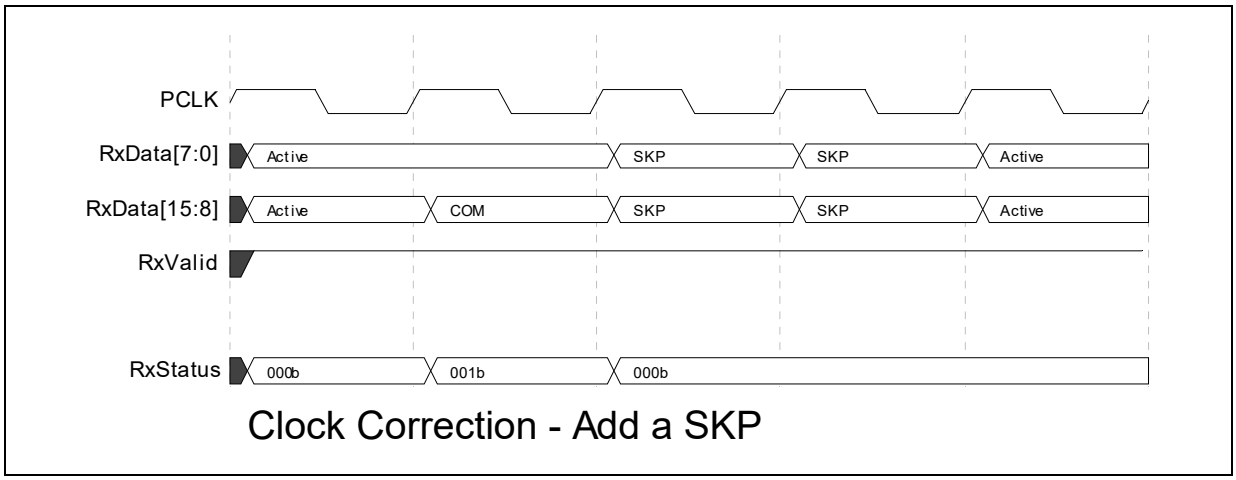
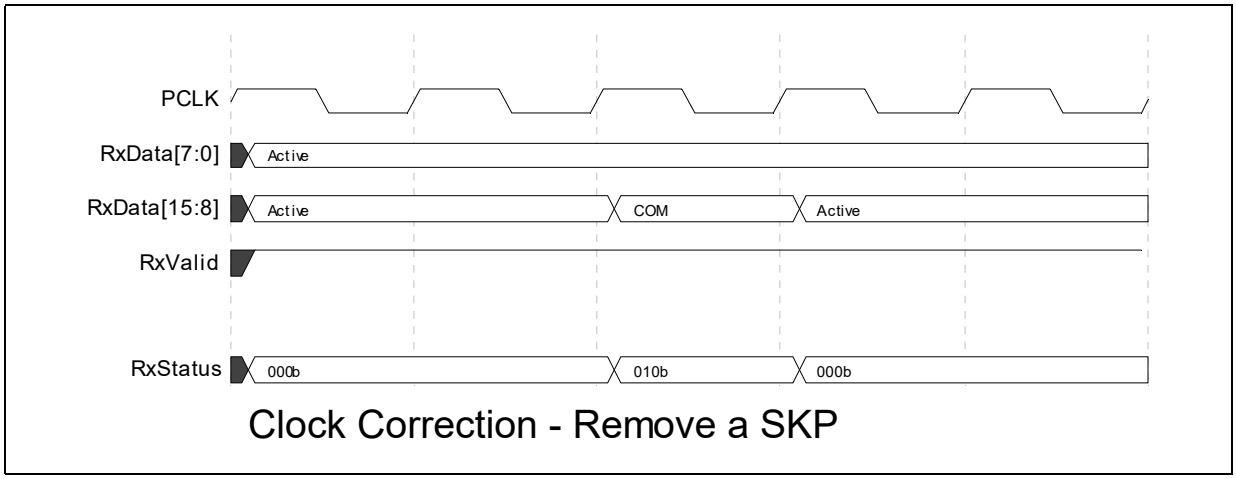


Figure 8-26 shows a sequence where a PHY operating in PCIe mode removed an SKP symbol from an SKP ordered set that only had one SKP symbol, resulting in a “bare” COM transferring across the parallel interface.

Figure 8-26. Clock Correction – Remove an SKP



8.15 Error Detection

The PHY is responsible for detecting receive errors of several types. These errors are signaled to the MAC layer using the receiver status signals (**RxStatus[2:0]**). Because of higher level error detection mechanisms (like CRC) built into the Data Link layer, there is no need to specifically identify symbols with errors, but reasonable timing information about when the error occurred in the data stream is important. When a receive error occurs, the appropriate error code is asserted for one clock cycle at the point in the data stream across the parallel interface closest to where the error actually occurred. There are four error conditions (five for SATA mode) that can be encoded on the **RxStatus** signals. If more than one error should happen to occur on a received byte (or set of bytes transferred across a 16-bit, 32-bit, or 64-bit interface), the errors should be signaled with the priority shown as follows:

1. 8B/10B decode error or block decode error
2. Elastic buffer overflow
3. Elastic buffer underflow (Cannot occur in Nominal Empty buffer model)
4. Disparity errors
5. Misalign (SATA mode only)

If an error occurs during an SKP ordered set or ALIGN, such that the error signaling and SKP or ALIGN added and removed signaling on **RxStatus** would occur on the same PCLK, then the error signaling has precedence.

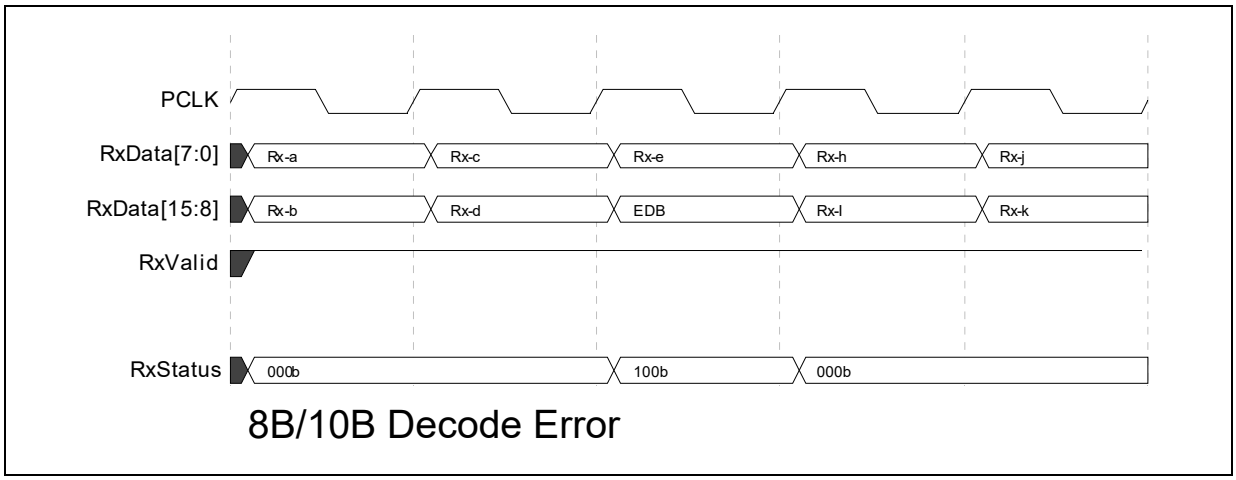
Note that the PHY does not signal 128/130B (PCIe) or 128/132B (USB) header errors. The raw received header bits are passed across the interface and the controller is responsible for any block header error detection and handling.

8.15.1 8B/10B Decode Errors

For a detected 8B/10B decode error, the PHY should place an End Bad (EDB) symbol (for PCIe or SATA) or SUB symbol (for USB) in the data stream in place of the bad byte, and encode **RxStatus** with a decode error during the clock cycle when the affected byte is transferred across the parallel interface. In the following example, the receiver

is receiving a stream of bytes Rx-a through Rx-z, and byte Rx-f has an 8B/10B decode error. In place of that byte, the PHY places an EDB (for PCIe or SATA) or SUB (for USB) on the parallel interface, and it sets **RxStatus** to the 8B/10B decode error code. Note that a byte that cannot be decoded may also have bad disparity, but the 8B/10B error has precedence. Also note that for greater than 8-bit interface, if the bad byte is on the lower byte lane, one of the other bytes may have bad disparity, but again, the 8B/10B error has precedence.

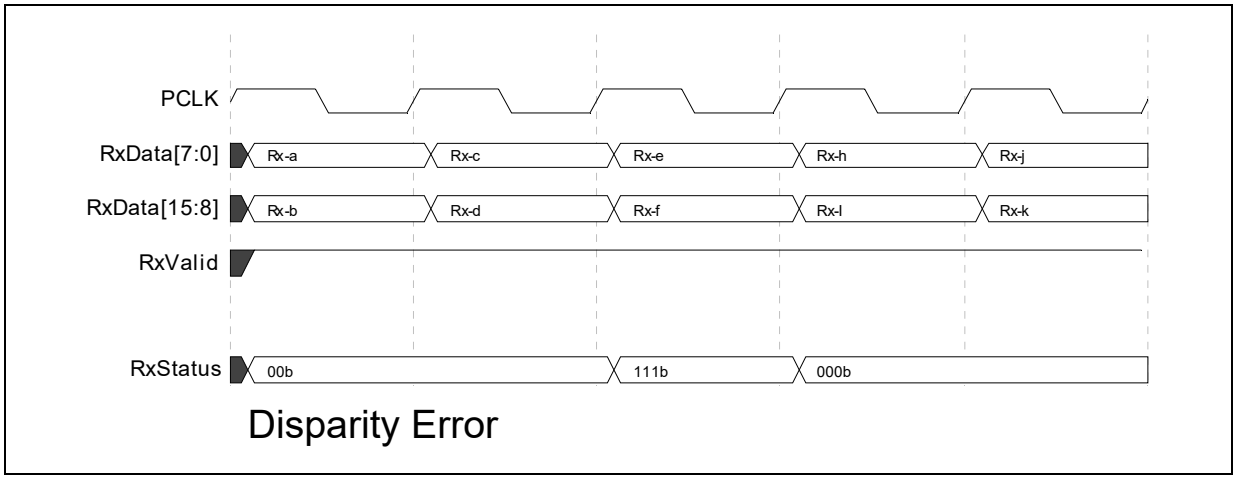
Figure 8-27. 8B/10B Decode Error



8.15.2 Disparity Errors

For a detected disparity error, the PHY should assert **RxStatus** with the disparity error code during the clock cycle when the affected byte is transferred across the parallel interface. For greater than 8-bit interfaces, it is not possible to discern which byte (or possibly both) had the disparity error. In the following example, the receiver detected a disparity error on either (or both) Rx-e or Rx-f data bytes, and it indicates this with the assertion of **RxStatus**. Optionally, the PHY can signal disparity errors as 8B/10B decode error (using code 0b100). (MACs often treat 8B/10B errors and disparity errors identically). When operating in the USB mode, signaling disparity errors is optional.

Figure 8-28. Disparity Error

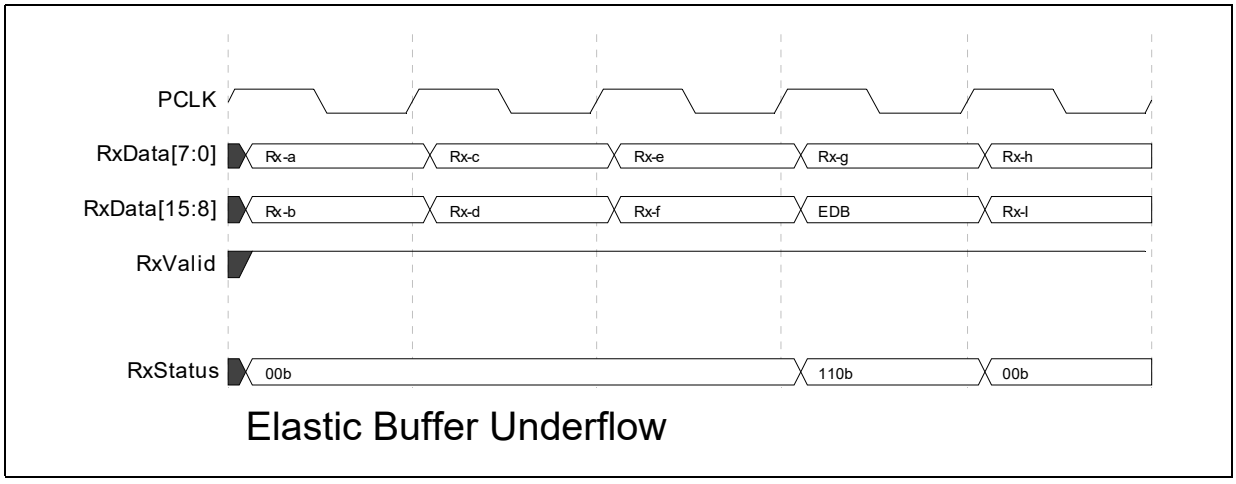


8.15.3 Elastic Buffer Errors

For elastic buffer errors, an underflow should be signaled during the clock cycle or clock cycles when a spurious symbol is moved across the parallel interface. The symbol moved across the interface should be the EDB symbol (for PCIe or SATA) or SUB symbol (for USB). In the following timing diagram, the PHY is receiving a repeating set of symbols Rx-a thru Rx-z. The elastic buffer underflows causing the EDB symbol (for PCIe) or SUB symbol (for USB) to be inserted between the Rx-g and Rx-h Symbols. The PHY drives **RxStatus** to indicate buffer underflow during the clock cycle when the EDB (for PCIe) or SUB (for USB) is presented on the parallel interface.

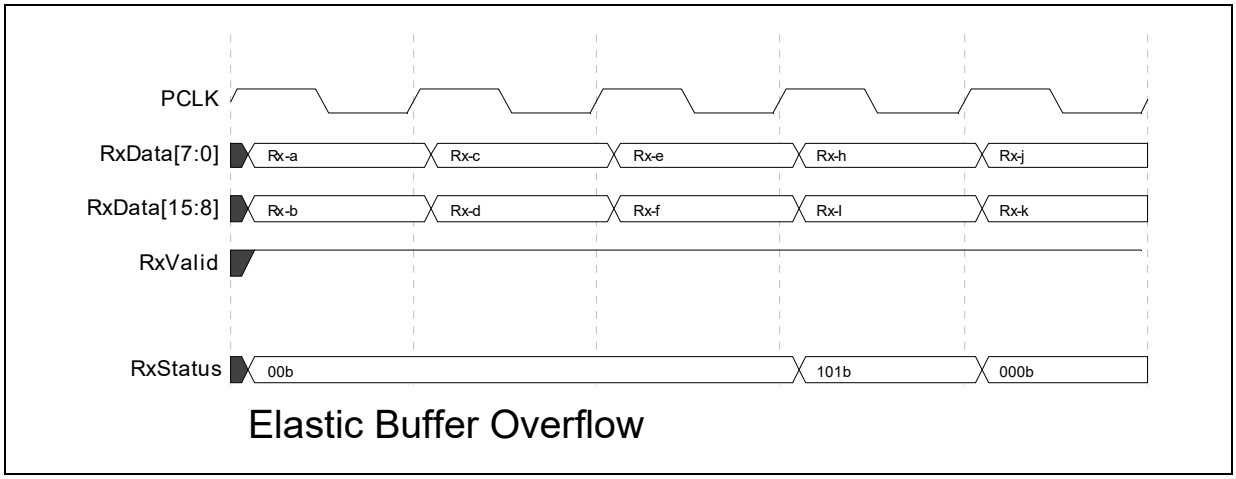
Note: The underflow is not signaled when the PHY is operating in Nominal Empty buffer mode. In this mode SKP ordered sets are moved across the interface whenever data needs to be inserted or the RxDataValid signal is used. The RxDataValid method is preferred.

Figure 8-29. Elastic Buffer Underflow



For an elastic buffer overflow, the overflow should be signaled during the clock cycle where the dropped symbol or symbols would have appeared in the data stream. For the 16-bit interface, it is not possible, or necessary, for the MAC to determine exactly where in the data stream the symbol was dropped. In the following timing diagram, the PHY is receiving a repeating set of symbols Rx-a thru Rx-z. The elastic buffer overflows causing the symbol Rx-g to be discarded. The PHY drives **RxStatus** to indicate buffer overflow during the clock cycle when Rx-g would have appeared on the parallel interface.

Figure 8-30. Elastic Buffer Overflow



8.15.3.1 Elastic Buffer Reset

The MAC can set the ElasticBufferResetControl bit (see Section 7.2.9 and Section 7.1.9) to initiate an EB reset sequence in the PHY. The PHY must complete the EB reset sequence within 16 PCLK cycles as follows:

1. Assert RxStatus to value of 1xx with RxValid
2. Hold RxStatus to 1xx while maintaining RxValid and RxDataValid
3. Move pointers back to their initial state
4. Release RxStatus to indicate clean data is being forwarded again

8.16 Loopback

- For USB and PCIe modes, the PHY must support an internal loopback as described in the corresponding base specification.
- For SATA the PHY may optionally support an internal loopback mode when EncodeDecodeBypass is asserted.
- In the SerDes architecture, loopback is handled in the MAC instead of the PHY.

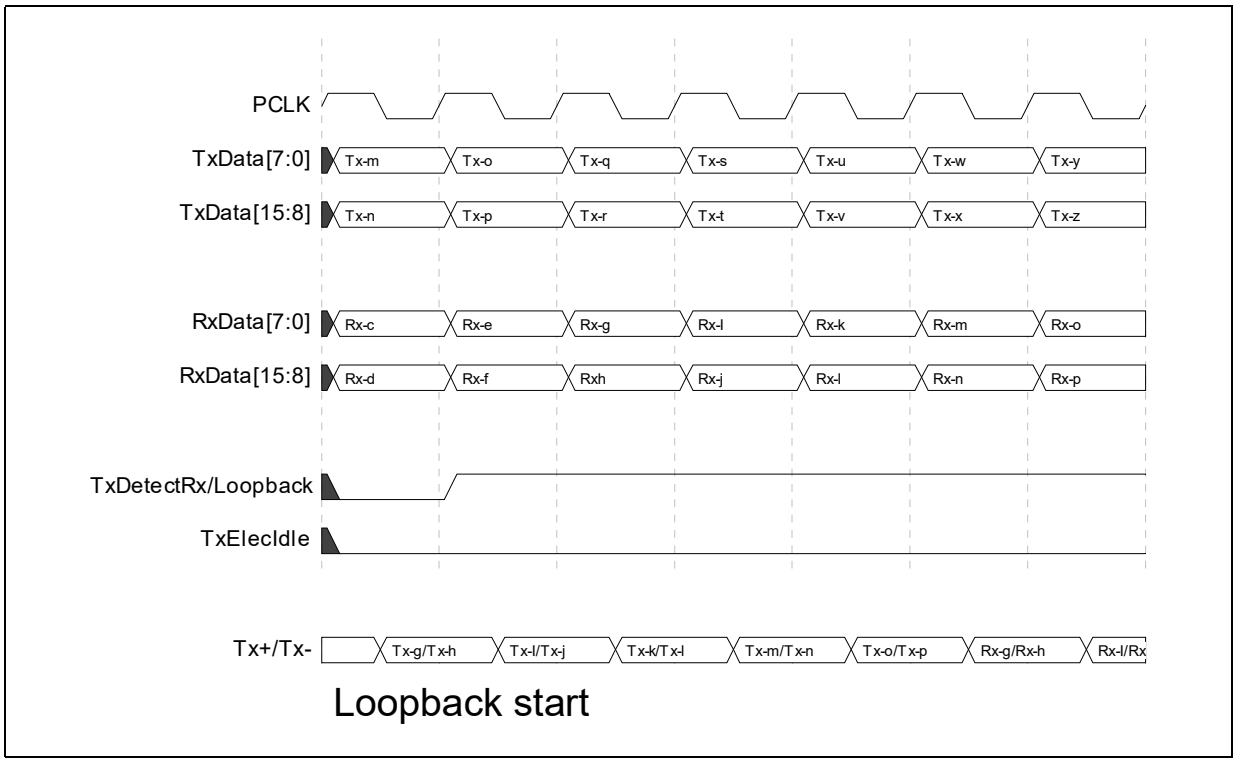
The PHY begins to loopback data when the MAC asserts **TxDetectRx/Loopback** while doing normal data transmission (that is, when **TxElecIdle** is deasserted). The PHY must, within the specified receive and transmit latencies, stop transmitting data from the parallel interface, and begin to loopback received symbols. While doing loopback, the PHY continues to present received data on the parallel interface.

The PHY stops looping back received data when the MAC deasserts **TxDetectRx/ Loopback**. Transmission of data on the parallel interface must begin within the specified transmit latency.

The following timing diagram shows the example timing for the beginning loopback. In this example, the receiver is receiving a repeating stream of bytes, Rx-a thru Rx-z. Similarly, the MAC is causing the PHY to transmit a repeating stream of bytes Tx-a thru Tx-z. When the MAC asserts **TxDetectRx/Loopback** to the PHY, the PHY begins to

loopback the received data to the differential **Tx+/Tx-** lines. Timing between assertion of **TxDetectRx/Loopback** and when Rx data is transmitted on the Tx pins is implementation dependent.

Figure 8-31. Loopback Start



The next timing diagram shows an example of switching from loopback mode to normal mode when the PHY is operating in PCIe Mode.

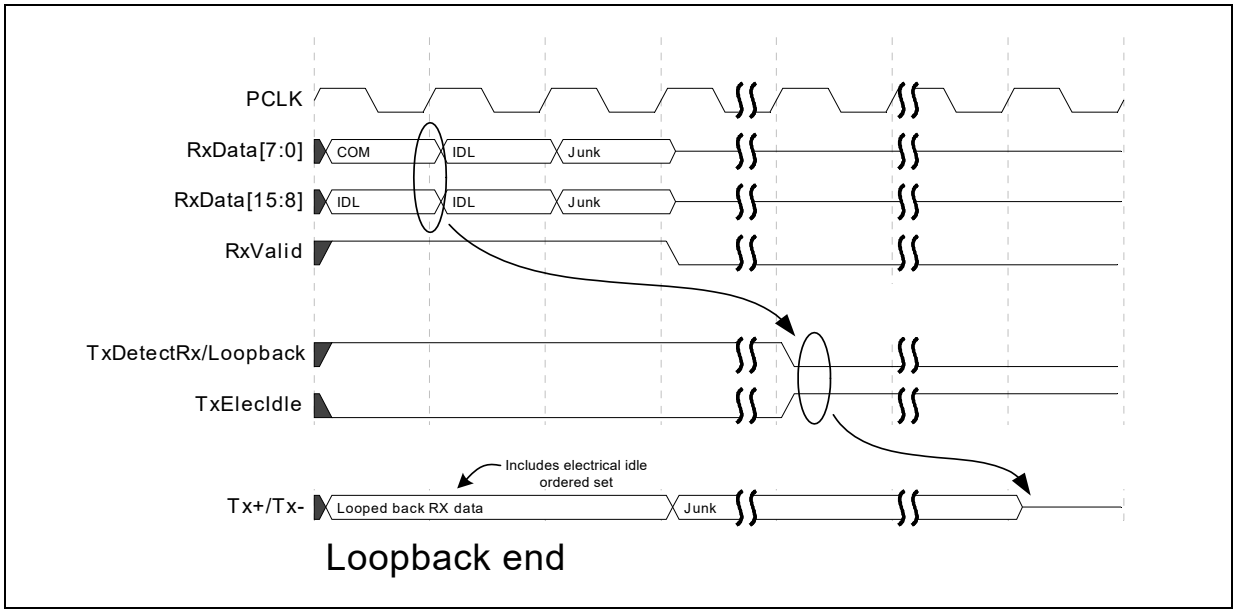
In PCIe Mode, when the MAC detects an electrical idle ordered set, the MAC deasserts the **TxDetectRx/Loopback** and asserts **TxElecIdle**. The PHY must transmit at least three bytes of the electrical idle ordered set before going to electrical idle.

Note: Transmission of the electrical idle ordered set should be part of the normal pipeline through the PHY and should not require the PHY to detect the electrical idle ordered set.

The base specification requires that a Loopback follower be able to detect and react to an electrical idle ordered set within 1 ms. The PHY's contribution to this time consists of the PHY's Receive Latency plus the PHY's Transmit Latency (see Section 8.20).

When the PHY is operating in USB mode, the device must only transition out of loopback on detection of LFPS signaling (reset) or when the VBUS is removed. When valid LFPS signaling is detected, the MAC transitions the PHY to the P2 power state to begin the LFPS handshake.

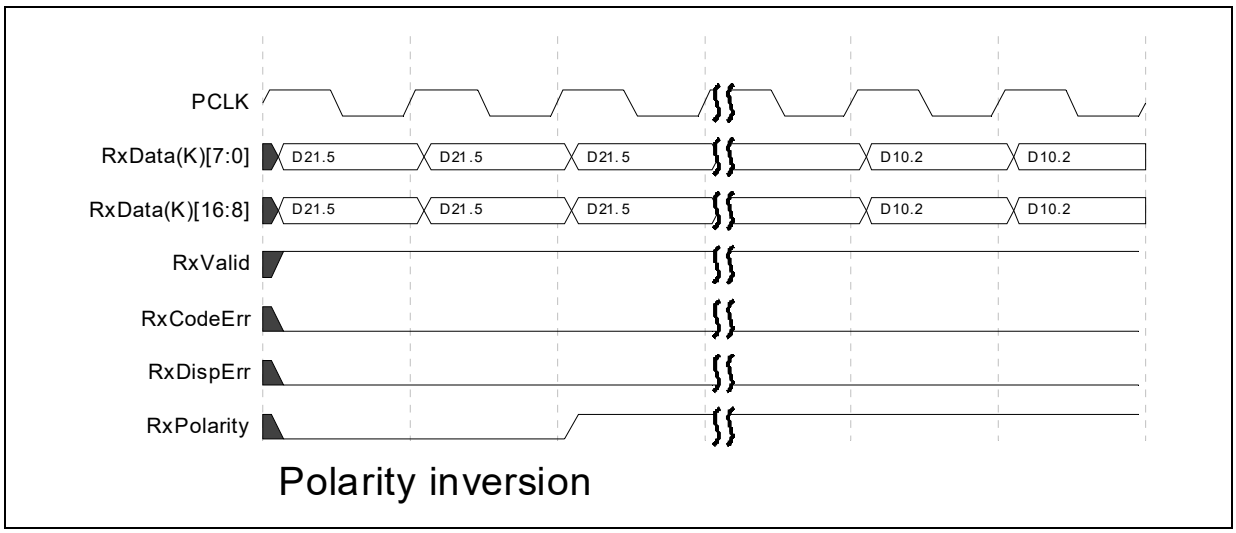
Figure 8-32. Loopback End



8.17 Polarity Inversion – PCIe and USB Modes

To support lane polarity inversion, the PHY must invert received data when **RxPolarity** is asserted. Inverted data must begin showing up on **RxData[]** within 20 PCLKs of when **RxPolarity** is asserted.

Figure 8-33. Polarity Inversion

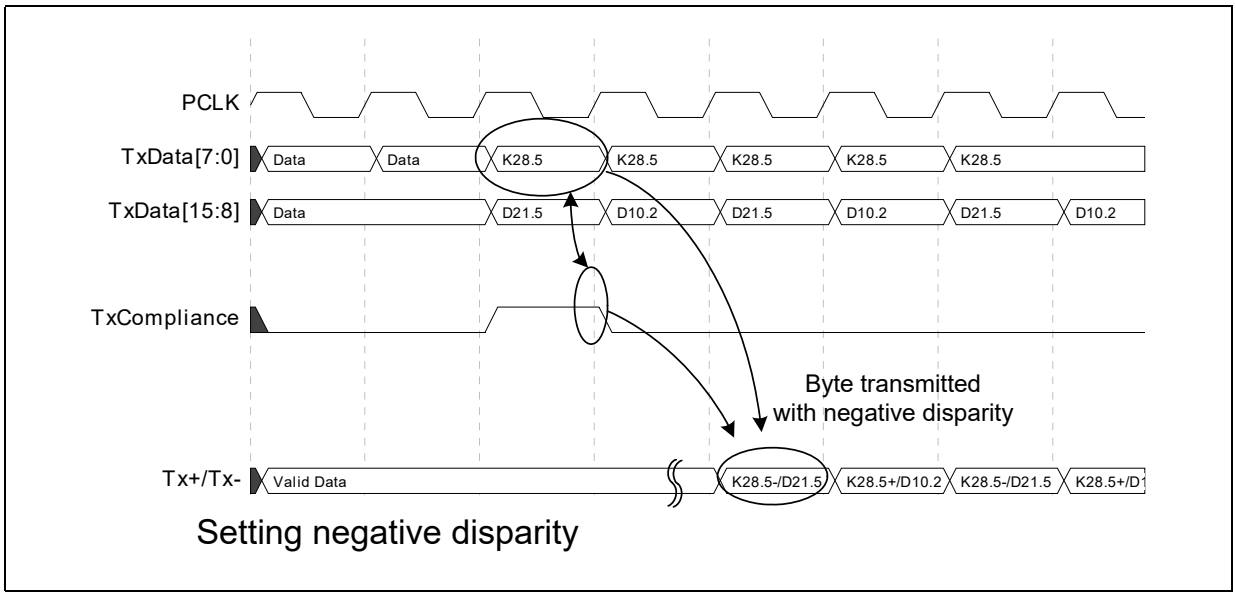


8.18 Setting Negative Disparity (PCIe Mode)

To set the running disparity to negative, the MAC asserts **TxCompliance** for one clock cycle that matches with the data that is to be transmitted with negative disparity. For a 16-bit interface, the low order byte will be the byte transmitted where running disparity

is negative. The example shows how **TxCompliance** is used to transmit the PCIe compliance pattern in PCIe mode. **TxCompliance** is only used in PCIe mode and is qualified by **TxDataValid** when **TxDataValid** is being used.

Figure 8-34. Setting Negative Disparity

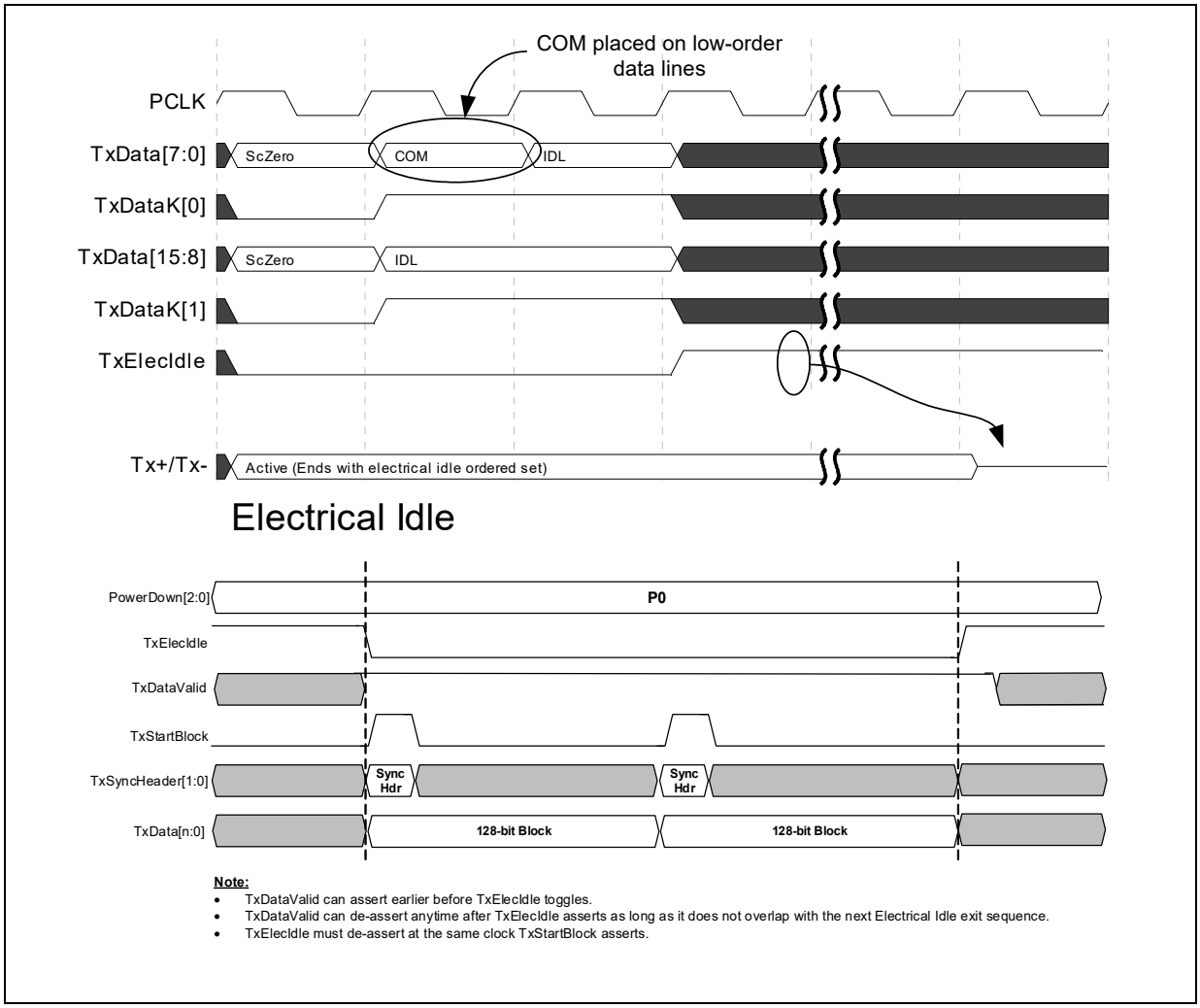


8.19 Electrical Idle – PCIe Mode

The base specification requires that devices send an Electrical Idle ordered set before Tx+/Tx- goes to the electrical idle state. For a 16-bit interface or 32-bit interface, the MAC must always align the electrical idle ordered set on the parallel interface so that the COM symbol is on the low-order data lines (**TxDataK[7:0]**). Figure 8-35 shows an example of electrical idle exit and entry for a PCIe 8 GT/s or 16 GT/s interface. TxDataValid must be asserted whenever TxElecIdle toggles as it is used as a qualifier for sampling TxElecIdle.

Note: For SerDes architecture, 1 bit of TxElecIdle is required per 16 bits of data.

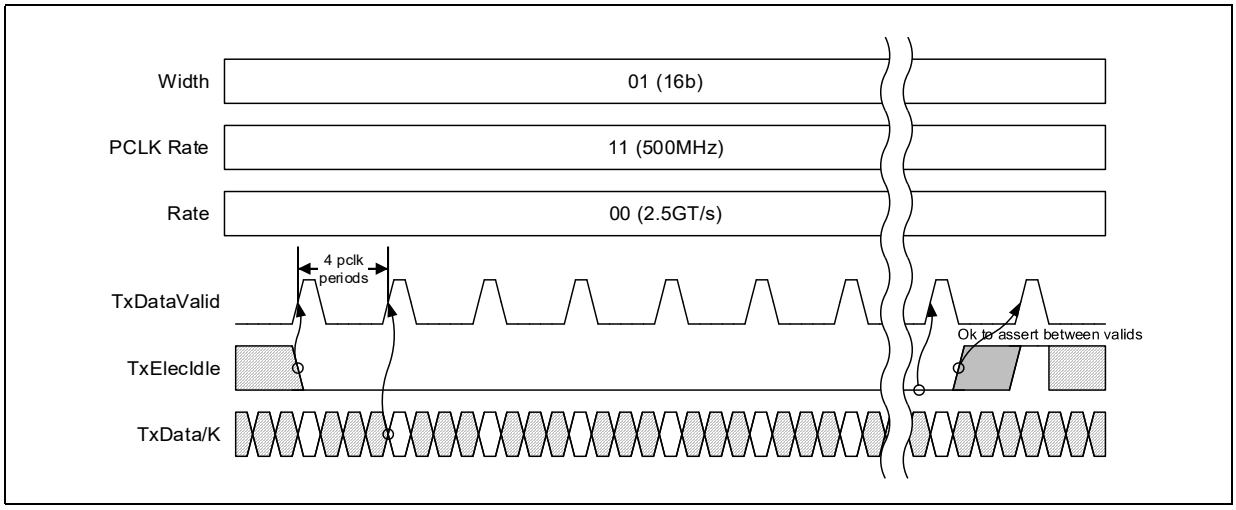
Figure 8-35. PCIe 3.0 TxDataValid Timings for Electrical Idle Exit and Entry



Note: Figure 8-35 only shows two blocks of TxData and thus TxDataValid does not deassert during the data. Other examples in the specification show longer sequences where TxDataValid deasserts.

When data throttling is happening, TxElecIdle must be set long enough to be sampled by TxDataValid as shown in Figure 8-36.

Figure 8-36. Data Throttling and TxElecIdle



8.20 Electrical Idle – All

The PIPE specification does not require RxStandby to be asserted within any amount of time after Electrical Idle or that it be asserted at all. Individual PHYs that rely on specific timing relationships for proper operation must specify their own timing requirements for RxStandby assertion, which may vary depending on whether they have staggering requirements.

8.21 Link Equalization Evaluation

While in the P0 power state, the PHY can be instructed to perform evaluation of the current Tx equalization settings of the link partner. Basic operation of the equalization evaluation is that the MAC requests the PHY to evaluate the current equalization settings by setting the **RxEqEval register field**. When the PHY has completed evaluating the current equalization settings, it writes to the **LinkEvaluationFeedbackDirectionChange** or the **LinkEvaluationFeedbackFigureMerit** register fields or both. After link equalization evaluation has completed, the MAC must clear the **RxEqEval** register field before initiating another evaluation.

Once the MAC has requested link equalization evaluation (by setting the **RxEqEval register bit**), the MAC must leave **RxEqEval** set until after the PHY has signaled completion by writing to the **LinkEvaluationFeedbackDirectionChange** or **LinkEvaluationFeedbackFigureMerit** register fields unless the MAC needs to abort the evaluation due to high level timeouts or error conditions. To abort an evaluation the MAC clears the **RxEqEval** register bit before the PHY has signaled completion. If the MAC aborts the evaluation the PHY must signal completion as quickly as possible. The MAC ignores returned evaluation values in an abort scenario.

Please refer to Section 9.10 for example waveforms illustrating successful equalization, invalid coefficient request, aborted equalization, and aborted equalization with race condition scenarios.

8.22 Implementation-Specific Timing and Selectable Parameter Support

PHY vendors (macrocell or discrete) must specify typical and worst-case timings for the cases listed in Table 8-4. Other implementation specific parameters listed in Table 8-4 must also be specified advertised by the PHY in its datasheet.

Table 8-4. Parameters Advertised in PHY Datasheet (Sheet 1 of 7)

Parameter	Description
Transmit Latency	Time for data moving between the parallel interface and the PCIe, SATA or USB serial lines. Timing is measured from when the data is transferred across the parallel interface (that is, the rising edge of **PCLK**) and when the first bit of the equivalent 10-bit symbol is transmitted on the **Tx+/Tx-** serial lines. The PHY reports the latency for each operational mode the PHY supports. **Note:** If the transmit latency is different when EncodeDecodeBypass is asserted – the PHY must report this latency separately.
Receive Latency	Time for data moving between the parallel interface and the PCIe, SATA or USB serial lines. Timing is measured from when the first bit of a 10-bit symbol is available on the **Rx+/Rx-** serial lines to when the corresponding 8-bit data is transferred across the parallel interface (that is, the rising edge of **PCLK**). The PHY reports the latency for each operational mode the PHY supports. The reported latency is the nominal latency assuming the elasticity buffer is full to its nominal operating level. **Note:** If the receive latency is different when EncodeDecodeBypass is asserted – the PHY must report this latency separately. Additionally, the expected latency must be reported separately for both elasticity buffer operating modes.
Power State After Reset	The PHY power state immediately following reset. The state after reset needs to provide PCLK and have common mode off. Reporting this parameter is required if the PHY supports either SATA mode or PCIe mode at 8 GT/s.
Loopback enable latency	Amount of time it takes the PHY to begin looping back receive data. Timed from when **TxDetectRx/Loopback** is asserted until the receive data is being transmitted on the serial pins. The PHY reports the latency for each operational mode the PHY supports.
Transmit Beacon – PCIe Mode.	Timed from when the MAC directs the PHY to send a beacon (power state is P2 and **TxElecIdle** is deasserted) until the beacon signaling begins at the serial pins.
Receive Beacon – PCIe Mode	Timed from when valid beacon signaling is present at the receiver pins until **RxElecIdle** is deasserted.
Transmit LFPS – USB Mode	Timed from when the MAC directs the PHY to send LFPS signaling until the LFPS signaling begins at the serial pins. Times are reported for each possible P state if the times are different for different power states.
Receive LFPS – USB Mode	Timed from when valid LFPS signaling is present at the receiver pins until **RxElecIdle** is deasserted.
MinTimBeforeLFPS -- USB4 Mode, DP Mode	The minimum required time from when the MAC signals to the PHY that it plans to transmit LFPS to when the MAC starts to transmit LFPS. Time is specified independently for each protocol.
MaxTimeBeforeLFPS – USB4 Mode, DP Mode	The maximum allowable time from when the MAC signals to the PHY that it plans to transmit LFPS to when the MAC must start transmitting LFPS. Time is specified independently for each protocol.
MinTimeEIAfterLFPS – USB4 Mode, DP Mode	The minimum required time in Electrical Idle after the MAC stops transmitting LFPS before the MAC is permitted to start transmitting high speed data. Time is specified independently for each protocol.
N_FTS with common clock (PCIe Mode)	Number of FTS ordered sets required by the receiver to obtain reliable bit and symbol lock when operating with a common clock. Note: This value may be required to be reported separately per rate.
N_FTS without common clock (PCIe Mode)	Number of FTS ordered sets required by the receiver to obtain reliable bit and symbol lock when operating without a common clock. Note: This value may be required to be reported separately per rate.

Table 8-4. Parameters Advertised in PHY Datasheet (Sheet 2 of 7)

Parameter	Description
PHY lock time	Amount of time for the PHY receiver to obtain reliable bit and symbol lock after valid symbols are present at the receiver. The PHY reports the time for each operational mode the PHY supports.
P0s to P0 transition time PCIe Mode.	Amount of time for the PHY to return to P0 state, after having been in the P0s state. Time is measured from when the MAC sets the **PowerDown** signals to P0 until the PHY asserts **PhyStatus**. PHY asserts **PhyStatus** when it is ready to begin data transmission and reception.
P1 to P0 transition time. PCIe Mode.	Amount of time for the PHY to return to P0 state, after having been in the P1 state. Time is measured from when the MAC sets the **PowerDown** signals to P0 until the PHY asserts **PhyStatus**. PHY asserts **PhyStatus** when it is ready to begin data transmission and reception.
P2 to P0 transition time PCIe Mode.	Amount of time for the PHY to go to P0 state, after having been in the P2 state. Time is measured from when the MAC sets the **PowerDown** signals to P1 until the PHY deasserts **PhyStatus**.
P1 to P0 transition time. USB Mode.	Amount of time for the PHY to return to P0 state, after having been in the P1 state. Time is measured from when the MAC sets the **PowerDown** signals to P0 until the PHY asserts **PhyStatus**. PHY asserts **PhyStatus** when it is ready to begin data transmission and reception.
P2 to P0 transition time. USB Mode.	Amount of time for the PHY to return to P0 state, after having been in the P2 state. Time is measured from when the MAC sets the **PowerDown** signals to P0 until the PHY asserts **PhyStatus**. PHY asserts **PhyStatus** when it is ready to begin data transmission and reception.
P3 to P0 transition time USB Mode.	Amount of time for the PHY to go to P0 state, after having been in the P3 state. Time is measured from when the MAC sets the **PowerDown** signals to P0 until the PHY deasserts **PhyStatus**. PHY asserts **PhyStatus** when it is ready to begin data transmission and reception.
Power state transition times between two power states that provide PCLK.	Amount of time for the PHY to transition to a new power state. Time is measured from when the MAC sets the **PowerDown** signals to POWER_STATE_X until the PHY asserts **PhyStatus**. PHY asserts **PhyStatus** when it is ready to begin data transmission and reception. The PHY reports this transition between each pair of power states it supports in each PHY mode it supports.
Power state transition times between a power state without PCLK and a power state with PCLK.	Amount of time for the PHY to go to a power state providing PCLK, after having been in a power state that does not provide PCLK. Time is measured from when the MAC sets the **PowerDown** signals to the new power state until the PHY deasserts **PhyStatus**. The PHY reports this time for each possible transition between a power state that does not provide PCLK and a power state that does provide PCLK. The PHY reports this transition time between each pair of power states it supports in each PHY mode it supports.
Power state transition times between a power state without PCLK and a power state without PCLK.	Amount of time for the PHY to go to a power state without PCLK, after having been in a power state that does not provide PCLK. Time is measured from when the MAC sets the **PowerDown** signals to the new power state until the PHY deasserts **PhyStatus**. The PHY reports this time for each possible transition between a power state that does not provide PCLK and a power state that does not provide PCLK. The PHY reports this transition time between each pair of power states it supports in each PHY mode it supports.
Supported power states.	The PHY lists each power state it supports for each PHY mode it supports. For each power state supported it reports whether PCLK is provided, the exit latency to the active power state, whether RxElecIdle is supported in the state, and the common mode state. **Note:** This is done for all states not already listed separately.
L1 Substate Management Mechanism	The PHY reports that of the following mechanisms it supports for L1 substate management: • Exclusively managed via PowerDown[3:0] • Managed via RxEIDetectDisable and TxCommonModeDisable • Both of the previous mechanisms are supported
RxEIDetectDisableSupported States for PCIe Mode	For PCIe mode, the PHY specifies which PowerDown states the MAC can use RxEIDetectDisable to disable the receiver Electrical Idle detect logic for power savings.

Table 8-4. Parameters Advertised in PHY Datasheet (Sheet 3 of 7)

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>ShortChannelPowerControlSettingsSupported</td><td>The PHY lists each power control setting that it supports via the ShortChannelPowerControl[1:0] signals. For each power control setting it supports, the PHY provides details of the optimizations made, which may include (but are not limited to) the following: Channel loss (for instance, 5 dB, 10 dB, 15 dB, 25 dB), DFE receiver equalization activity factor (for instance, completely turned off or reduced DFE taps), whether CTLE is turned on or off, and whether clock recovery is shared or per lane. The following table should be filled in and published:
<table>
<tr><th>PowerControlSetting</th><th>Description (Suggested)</th><th>pJ/bit</th><th>Optimization (or Specify Setting not Supported)</th></tr>
<tr><td>00b (PowerControlSetting0)</td><td>Normal operation</td><td>Vendor-specified</td><td>N/A</td></tr>
<tr><td>01b (PowerControlSetting1)</td><td>Most power optimized setting for <=5dB channel and half swing transmitter</td><td>Vendor-specified</td><td>Vendor-defined</td></tr>
<tr><td>10b (PowerControlSetting2)</td><td>Most power optimized setting for <=10dB channel and half swing transmitter</td><td>Vendor-specified</td><td>Vendor-defined</td></tr>
<tr><td>11b (PowerControlSetting3)</td><td>Vendor-defined</td><td>Vendor-specified</td><td>Vendor-defined</td></tr>
</table></td></tr>
<tr><td>LFPS Circuit Disable for USB Mode</td><td>The PHY reports whether the MAC can use RxEIDetectDisable to disable the LFPS circuit for power savings.</td></tr>
<tr><td>LFPS Circuit Disable for USB4 Mode</td><td>The PHY reports whether the MAC can use RxEIDetectDisable to disable the LFPS circuit for power savings.</td></tr>
<tr><td>Simultaneous Rate and Power State Change</td><td>The PHY reports if it supports simultaneous rate and power state changes for each PHY mode it supports.</td></tr>
<tr><td>Data Rate change time. PCIe Mode and SATA Mode.</td><td>Amount of time the PHY takes to perform a data rate change. Time is measured from when the MAC changes Rate to when the PHY signals rate change is complete with the single clock assertion of PhyStatus. There may be separate values for each possible change between different supported rates for each supported PHY mode.</td></tr>
<tr><td>Transmit Margin values supported. PCIe Mode and USB Mode.</td><td>Transmitter voltage levels.
[2] [1] [0] Description
0 0 0 TxMargin value 0 =
0 0 1 TxMargin value 1 =
0 1 0 TxMargin value 2 =
0 1 1 TxMargin value 3 =
1 0 0 TxMargin value 4 =
1 0 1 TxMargin value 5 =
1 1 0 TxMargin value 6 =
1 1 1 TxMargin value 7 =</td></tr>
<tr><td>Max Equalization Settings for C_{-1}</td><td>Reports the maximum number of settings supported by the PHY for the 8.0 GT/s, 16 GT/s, and 32 GT/s equalization. The maximum number of settings must be less than 64.</td></tr>
<tr><td>Max Equalization Settings for C_0</td><td>Reports the maximum number of settings supported by the PHY for the 8.0 GT/s, 16 GT/s, and 32 GT/s equalization. The maximum number of settings must be less than 64.</td></tr>
<tr><td>Max Equalization Settings for C_1</td><td>Reports the maximum number of settings supported by the PHY for the 8.0 GT/s, 16 GT/s, and 32 GT/s equalization. The maximum number of settings must be less than 64.</td></tr>
<tr><td>Default Equalization settings for full swing preset Pn.</td><td>Reports the recommended setting values for C_{-1}, C_0, C_1 for each full swing preset. Note: This should be reported separately per rate.</td></tr>
<tr><td>Default Equalization settings for half swing preset Pn.</td><td>Reports the recommended setting values for C_{-1}, C_0, C_1 for each half swing preset. Note: This should be reported separately per rate.</td></tr>
<tr><td>Default Equalization settings for recommended Tx EQ value of 0 dB preshoot and -2.5 dB de-emphasis.</td><td>Reports the recommended setting values for C_{-1}, C_0, C_1 for the USB 3.1 0 dB preshoot and -2.5 dB de-emphasis recommended Tx EQ setting.</td></tr>
</table>

Table 8-4. Parameters Advertised in PHY Datasheet (Sheet 4 of 7)

Parameter	Description
Default Equalization settings for recommended Tx EQ value of 2.7 dB preshoot and -3.3 dB de-emphasis.	Reports the recommended setting values for C_{-1}, C_0, C_1 for the USB 3.1 0 dB preshoot and -2.5 dB de-emphasis recommended Tx EQ setting.
Default Equalization settings for recommended Tx EQ value of 2.2 dB preshoot and -3.1 dB de-emphasis	Reports the recommended setting values for C_{-1}, C_0, C_1 for the USB 3.2 2.2 dB preshoot and -3.1 dB de-emphasis.
Default Equalization settings for recommended Tx EQ value of 0 dB preshoot and 0 dB de-emphasis	Reports the recommended setting values for C_{-1}, C_0, C_1 for the USB 3.2 0 dB preshoot and 0 dB de-emphasis.
Default Equalization settings for recommended Tx EQ value of 0 dB preshoot and -3.1 dB de-emphasis	Reports the recommended setting values for C_{-1}, C_0, C_1 for the USB 3.2 0 dB preshoot and -3.1 dB de-emphasis.
Default Equalization settings for recommended Tx EQ value of 2.2 dB preshoot and 0 dB de-emphasis	Reports the recommended setting values for C_{-1}, C_0, C_1 for the USB 3.2 2.2 dB preshoot and 0 dB de-emphasis.
Dynamic Preset Coefficient Update Support	A PHY indicates if it dynamically updates coefficients.
Link Evaluation Feedback Format Supported	The PHY reports whether it supports link evaluation feedback in the Figure of Merit format, in the direction change format, or whether it supports both formats. The PHY must support at least one format.
Figure of Merit range	If the PHY reports link equalization feedback in the Figure of Merit format it reports the maximum value it will report. The maximum value must be less than 256.
Figure of Merit for BER target	If the PHY reports link equalization feedback in the Figure of Merit format it reports the minimum value that the PHY estimates corresponds to a link BER of E-12.
Default Link Partner Preset[3:0]	If the PHY prefers the link parter to start with a specific preset during link evaluation it reports the preferred starting preset. The default link partner preset value is encoded as follows: 0000b – Preset P0. 0001b – Preset P1. 0010b – Preset P2. 0011b – Preset P3. 0100b – Preset P4. 0101b – Preset P5. 0110b – Preset P6. 0111b – Preset P7. 1000b – Preset P8. 1001b – Preset P9. 1010b – Preset P10. 1011b – Reserved 1100b – Reserved 1101b – Reserved 1110b – Reserved 1111b – No Preference. **Note:** This should be reported separately per rate.
Beacon Support	The PHY indicates whether it supports beacon transmission. Beacon transmission is optional. 1: Beacon transmission is supported. 0: Beacon transmission is not supported.

Table 8-4. Parameters Advertised in PHY Datasheet (Sheet 5 of 7)

Parameter	Description
EncodeDecodeBypassSupport[3:0]	The PHY indicates whether it supports optional EncodeDecodeBypass mode at each signaling rate. [0] Rate[1:0] = 0 [1] Rate[1:0] = 1 [2] Rate[1:0] = 2 [3] Rate[1:0] = 3 The support value for each rate is encoded as follows: 0 - No support for EncodeDecodeBypass 1 – Support for EncodeDecodeBypass
NoDeemphasisSupport[1:0]	The PHY indicates whether it supports an optional No De-emphasis signaling mode at 2.5 and 5.0 GT/s signaling rates. [0] Support at 2.5 GT/s [1] Support at 5.0 GT/s The support value for each rate is encoded as follows: 0 – No support for a no de-emphasis signaling mode. 1 – Support for a no de-emphasis signaling mode.
SupportedLFPresets	List of presets the PHY supports at 8 GT/s, 16 GT/s, and 32 GT/s for half swing in addition to the 5 required by the base specification.
PCLK Mode[1:0]	The PHY indicates whether it supports PCLK as a PHY output or PCLK as a PHY input. [0] Supports PCLK as an output [1] Supports PCLK as an input The support value for each rate is encoded as follows: 0 – No support. 1 – Support. Configuration for a PHY that supports both PCLK modes is PHY specific.
PHYClockInsertionDelay	A PHY that supports "PCLK as an input" mode must report the maximum delay and the minimum delay (insertion delay) for any sequential logic at the MAC/PHY interface that will use PCLK in the PHY in picoseconds.
SupportedPhyModes	List of all modes the PHY supports for the PHY Mode[1:0] input.
MaximumPCIExpressRate	Value for DataRate input corresponding to the maximum rate the PHY supports while in PCIe mode. This field is undefined if the PHY does not support PCIe mode.
MaximumSataRate	Value for the DataRate input corresponding to the maximum rate the PHY supports while in SATA Mode. This field is undefined if the PHY does not support SATA mode.
ListofSupportedSataModes	List of all supported signaling rate, width, PCLK rate combinations supported in Table 3-2.
ListofSupportedPCIExpressModes	List of all supported signaling rate, width, PCLK rate combinations supported in Table 3-1.
MaximumEntriesInElasticityBuffer	Maximum number of entries that can be stored in the elasticity buffer. The PHY reports the maximum number of entries for each operational mode the PHY supports.
ElasticityBufferEntrySize	Size of a data entry in the elasticity buffer in bits. The PHY reports this size for each operation mode the PHY supports.
MaximumElasticBufferLocationUpdateFrequency	Maximum update frequency the PHY supports for updating the ElasticBufferLocation register. This field is only relevant for original PIPE architecture.
MinimumElasticBufferLocationUpdateFrequency	Minimum update frequency the PHY supports for updating the ElasticBufferLocation register. This field is only relevant for original PIPE architecture.

Table 8-4. Parameters Advertised in PHY Datasheet (Sheet 6 of 7)

Parameter	Description
EnhancedPTMTimingSupport	The PHY indicates whether it supports optional elasticity buffer location information through the ElasticBufferLocation control signals to allow more accurate timing of received packets within the MAC. The support value is encoded as follows: 0 – No support. 1 – Support.
L1PMSubStatesSupport	The PHY indicates whether it supports optional L1 PM Substates. A PHY that supports L1 PM Substates must support asynchronous power state transitions. The support value is encoded as follows: 0 – No support. 1 – Support.
RXMarginingVoltageSupported[1]	The PHY indicates whether it supports voltage margining, encoded as follows: 0 – No Support 1 – Support. The PHY needs to specify this value for PCIe at 16 GT/s, 32 GT/s, 64 GT/s, and 128 GT/s.
RXMarginingSamplingRateVoltage[5:0][1]	Percentage of bits margined during voltage margining mode is calculated as 1/ 64×(Sampling_Rate[5:0]+1). Allowable values: 0-63. The PHY needs to specify this value for PCIe at 16 GT/s, 32 GT/s, 64 GT/s, and 128 GT/s.
RXMarginingSamplingRateTiming[5:0][1]	Percentage of bits margined during timing margining mode is calculated as 1/ 64×(Sampling_Rate[5:0]+1). Allowable values: 0-63. The PHY needs to specify this value for PCIe at 16 GT/s, 32 GT/s, 64 GT/s, and 128 GT/s.
RXMarginingIndependentLeftRight[1]	The PHY indicates whether it supports independent left and right time margining. The support value is encoded as follows: 0 – No Support 1 – Support. The PHY needs to specify this value for PCIe at 16 GT/s, 32 GT/s, 64 GT/s, and 128 GT/s.
RXMarginingIndependentUpDown[1]	The PHY indicates whether it supports independent up and down voltage margining. The support value is encoded as follows: 0 – No Support 1 – Support. The PHY needs to specify this value for PCIe at 16 GT/s, 32 GT/s, 64 GT/s, and 128 GT/s.
RXMarginingIndependentErrorSampler[1]	The PHY indicates whether it supports an error sampler independent from the main sampler to allow higher BER's to be measured. The support value is encoded as follows: 0 – No Support 1 – Support. The PHY needs to specify this value for PCIe at 16 GT/s, 32 GT/s, 64 GT/s, and 128 GT/s.
RXMarginingVoltageSteps[6:0][1]	Total number of voltage steps, minimum range +/- 50mV. A value of zero indicates that voltage margining is not supported. Allowable non-zero values: 32–127. The PHY needs to specify this value for PCIe at 16 GT/s, 32 GT/s, 64 GT/s, and 128 GT/s.
RXMarginingTimingSteps[5:0][1]	Total number of timing steps, minimum range +/-0.2UI. Allowable values: 8–63. The PHY needs to specify this value for PCIe at 16 GT/s, 32 GT/s, 64 GT/s, and 128 GT/s.
RXMarginingMaxVoltageOffset[6:0][1]	Offset at maximum step value as percentage of one volt. Allowable values: 5–50. The PHY needs to specify this value for PCIe at 16 GT/s, 32 GT/s, 64 GT/s, and 128 GT/s.

Table 8-4. Parameters Advertised in PHY Datasheet (Sheet 7 of 7)

Parameter	Description
RXMarginingMaxTimingOffset[6:0][1]	Offset at maximum step value as percentage of nominal UI. Allowable values: 20–50. The PHY needs to specify this value for PCIe at 16 GT/s, 32 GT/s, 64 GT/s, and 128 GT/s.
RXMarginingMaxLanes[5:0][1]	Maximum number of lanes that can be margined simultaneously. Allowable values: 1–32. Recommended value=number of lanes the PHY supports. The PHY needs to specify this value for PCIe at 16 GT/s, 32 GT/s, 64 GT/s, and 128 GT/s.
RXMarginingSampleReportingMethod[1]	Indicates whether a sample frequency or a sample count is reported. This value is encoded as follows: 0 – Sample Count Reported 1 – Sample Frequency Reported The PHY needs to specify this value for PCIe at 16 GT/s, 32 GT/s, 64 GT/s, and 128 GT/s.
RXMarginingMaxTimingOffsetChange[6:0]	Maximum number of steps margin offset can be changed with one command during timing margining. Allowable values: 1–127. The PHY needs to specify this value for PCIe at 16 GT/s, 32 GT/s, 64 GT/s, and 128 GT/s.
RXMarginingMaxVoltageOffsetChange[6:0]	Maximum number of steps margin offset can be changed with one command during voltage margining. Allowable values: 1–127. The PHY needs to specify this value for PCIe at 16 GT/s, 32 GT/s, 64 GT/s, and 128 GT/s.
RXMessageBusWriteBufferDepth[3:0]	The PHY indicates the number of write buffer entries that it has implemented to receive writes from the MAC, where one entry can hold the three bytes of information associated with each write transaction.
TXMessageBusMinWriteBufferDepth[3:0]	The PHY indicates the minimum number of write buffer entries it expects the MAC to implement to receive writes from the PHY. Allowable values: 0–8. The MAC may choose to implement more than the minimum required by the PHY; however, there may not be any benefit in doing so.
WidthChangeHandshakeRequirement	The PHY indicates whether it needs the MAC to use the PclkChangeOk/PclkChangeAck handshake for rate plus width changes.
RateChangeHandshakeRequirement	The PHY indicates whether it needs the MAC to use the PclkChangeOK/PclkChangeAck handshake for all rate changes.
AsynchReceiverDetectSupport	The PHY indicates whether is supports asynchronous receiver detection in PCIe P2 power state.
EIOS to Valid Electrical Idle Transition Time (PCIe mode)	The PHY indicates the value of $T_{TX\text{-}IDLE\text{-}SET\text{-}TO\text{-}IDLE}$.
Datapath Options Supported	The PHY indicates whether it supports SerDes architecture and original PIPE. The PHY specifies how it should be configured to use one or the other option.
Control Path Options Supported	The PHY indicates whether it supports the Low Pin Count signal interface and the legacy signal interface. The PHY specifies how it should be configured to use one or the other option.
PCIeRxEqTrainRequirement	The PHY indicates whether it needs the controller to implement support for RxEqTraining/ RxEqTrainDone handshake for PCIe.
PhyRecalRequirement	The PHY indicates whether it needs the controller to implement support for PhyIORecalRequest, IORecal, and IORecalDone.
PAM4RestrictedLevelsRequirement	The PHY indicates whether it requires the controller to implement the PAM4RestrictedLevels field in the PHY RX Control1 message bus register.
RxInPhase01EqRequirement	The PHY indicates whether it requires the controller to implement the RxInPhase01Equalization field in the PHY RX Control1 message bus register.
MaxRxClkFrequency	The PHY advertises the maximum RxCLK frequency that it guarantees not to exceed even if RxValid is deasserted.
MinimumMacCLKReset#AssertionPulse	PHYs that support the MacCLK feature specify the minimum MacCLKReset# assertion pulse duration required.
MaxSSCEnableDisableTime	The PHY indicates the maximum time it takes to complete SSC enable or disable.

1. See the PCIe base specification. In case of discrepancy, the PCIe base specification supersedes the PIPE specification.

8.23 Control Signal Decode Table – PCIe Mode

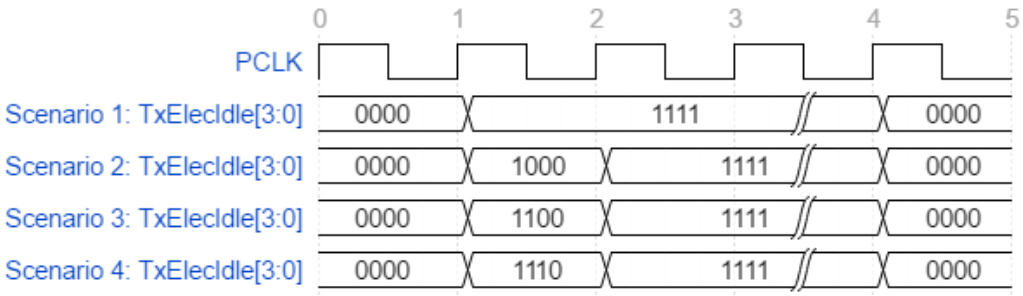
Table 8-5 summarizes the encodings of four of the seven control signals that cause different behaviors depending on power state. For the other three signals, **Reset#** always overrides any other PHY activity. **TxCompliance** and **RxPolarity** are only valid when the PHY is in P0 and is actively transmitting. Note that these rules only apply to lanes that have not been "turned off" as described in Section 10 (multi-lane PIPE).

For SerDes mode, the rules summarized in Table 8-5 apply to each of the TxElecIdle[3:0] bits independently for the P0 and P0s states. There is an expectation that entering Electrical Idle must occur from MSB to LSB, that is, valid values of TxElecIdle[3:0] are 1000b, 1100b, 1110b, 1111b, and 0000b. Figure 8-7 shows the various scenarios of valid TxElecIdle transitions. Transitions into Electrical Idle can include a single cycle of only a subset of data bytes driven to Electrical Idle followed by all data bytes in Electrical Idle; transitions out of Electrical Idle must be done simultaneously for all data bytes. For the P1 and P2 power states, all the bits of TxElecIdle[3:0] are expected to be driven to the same value so only TxElecIdle[0] needs to be decoded.

Table 8-5. Control Signal Decode Table – PCIe Mode

PowerDown[1:0]	TxDetectRx/ Loopback	TxElecIdle	Description
P0: 00b	0	0	PHY is transmitting data. MAC is providing data bytes to be sent every clock cycle.
	0	1	PHY is not transmitting and is in electrical idle.
	1	0	PHY goes into loopback mode.
	1	1	Illegal. MAC should never do this.
P0s: 01b	Don't care	0	Illegal. MAC should always have PHY doing electrical idle while in P0s. PHY behavior is undefined if **TxElecIdle** is deasserted while in P0s or P1.
		1	PHY is not transmitting and is in electrical idle. **Note:** Any data transferred across the PIPE interface before **TxElecIdle** is asserted, but not yet signaled on the analog interface is signaled before the analog interface becomes idle.
P1: 10b	Don't care	0	Illegal. MAC should always have PHY doing electrical idle while in P1. PHY behavior is undefined if **TxElecIdle** is deasserted while in P0s or P1.
	0	1	PHY is idle.
	1	1	PHY does a receiver detection operation.
P2: 11b	Don't care	0	PHY transmits Beacon signaling
	0	1	PHY is idle
	1	1	PHY does a receiver detection operation

Figure 8-37. Possible TxElecIdle[3:0] Transition Scenarios



8.24 Control Signal Decode Table – USB Mode, USB4 Mode, and DisplayPort Mode

Table 8-6 summarizes the encodings of four of the seven control signals that cause different behaviors depending on power state. For the other three signals, **Reset#** always overrides any other PHY activity. **RxPolarity** is only valid, and therefore should only be asserted, when the PHY is in P0 and is actively transmitting.

Note: The same table is applicable to TxDetectRx2 and TxElecIdle2.

Table 8-6. Control Signal Decode Table – USB Mode, USB4 Mode, and DisplayPort Mode (Sheet 1 of 2)

PowerDown[3:0]	TxDetectRx/ Loopback	TxElecIdle	Description
P0: 0000b	0	0	PHY is transmitting data. MAC is providing data bytes to be sent every clock cycle.
	0	1	PHY is not transmitting and is in electrical idle. Note that any data transferred across the PIPE interface before **TxElecIdle** is asserted, but not yet signaled on the analog interface is signaled before the analog interface becomes idle.
	1	0	USB mode: PHY goes into loopback mode. USB4 and DisplayPort mode: If PHY configured to transmit LFPS, PHY goes into loopback mode. If MAC configured to transmit LFPS, indicates that MAC is transmitting LFPS on TxData parallel data interface.
	1	1	USB mode: PHY transmits LFPS signaling USB4 and DisplayPort mode: If PHY configured to transmit LFPS, PHY transmits LFPS signaling. If MAC configured to transmit LFPS, indicates that MAC plans to transmit LFPS on TxData soon and PHY should prepare.
P1: 0001b	Don't care	0	USB: PHY transmits LFPS signaling USB4 and DisplayPort: Not allowed.
		1	PHY is not transmitting and is in electrical idle.

Table 8-6. Control Signal Decode Table – USB Mode, USB4 Mode, and DisplayPort Mode (Sheet 2 of 2)

PowerDown[3:0]	TxDetectRx/ Loopback	TxElecIdle	Description
P2: 0010b or P3: 0011b	Don't care	0	Not allowed
	0	1	PHY is idle.
	1	1	USB: PHY does a receiver detection operation. USB4 and DisplayPort: Not allowed.

8.25 Control Signal Decode Table – SATA Mode

The following table summarizes the encodings of the control signals that cause different behaviors in POWER_STATE_0. For other control signals, **Reset#** always overrides any other PHY activity.

Note: The PHY transmit latency reported in Section 8.20 must be consistent for all the different behaviors in POWER_STATE_0. This means that the amount of time OOB signaling is present on the analog Tx pair must be the same as the time OOB signaling was indicated on the PIPE interface.

Table 8-7. Control Signal Decode Table – SATA Mode

PowerDown[2:0]	TxDetectRx/ Loopback	TxElecIdle	Description
POWER_STATE_0: 00b	0	0	PHY is transmitting data. MAC is providing data bytes to be sent every clock cycle.
	0	1	PHY is not transmitting and is in electrical idle. **Note:** Any data transferred across the PIPE interface before **TxElecIdle** is asserted, but not yet signaled on the analog interface is signaled before the analog interface becomes idle.
	1	0	PHY goes into loopback mode.
	1	1	PHY transmits OOB signaling with pattern determined by Tx Pattern. Note that a PHY must ensure the transition between OOB signaling and data signaling is performed smoothly on a symbol boundary on the analog interface.
Power Stater other than POWER_STATE_0	Don't care	Don't care	PHY is not transmitting and is in electrical idle.

8.26 Required Synchronous Signal Timings

To improve interoperability between MACs and PHYs from different vendors the following timings for synchronous signals are required:

- Setup time for input signals: No greater than 25% of cycle time
- Hold time for input signals: 0 ns
- PCLK to data valid for outputs: No greater than 25% of cycle time

8.27 128b/130b Encoding and Block Synchronization (PCIe 8 GT/s, 16 GT/s, and 32 GT/s)

For every block (usually 128 bits – shorter/longer SKP blocks are sometimes transmitted by Retimers) that is moved across the PIPE TxData interface at the 8.0 GT/s rate, 16 GT/s rate, or 32 GT/s rate, the PHY must transmit two extra bits. The MAC must use the TxDataValid signal periodically to allow the PHY to transmit the built-up backlog of data. For example, if the TxData bus is 16-bits wide and PCLK is 500 MHz then every eight blocks the MAC must deassert TxDataValid for one PCLK to allow the PHY to transmit the 16-bit backlog of built up data. The buffers used by the PHY to store Tx data related to the 128/130b encoding rate mismatch must be empty when the PHY comes out of reset and must be empty whenever the PHY exits electrical idle (since Tx buffers are flushed before entry to idle). The PHY must use RxDataValid in a similar fashion. TxDataValid and RxDataValid must be deasserted for one clock exactly every N blocks when the PIPE interface is operating at 8 GT/s or 16 GT/s, where N is 4 for an 8 bit-wide interface, 8 for a 16 bit wide interface, and 16 for a 32 bit wide interface. The MAC must first deassert TxDataValid immediately after the end of the Nth transmitted block following reset or exit from electrical idle. Examples of the timing for TxDataValid are shown in Figure 8-38 for an 8 bit interface and in Figure 8-39 for a 16 bit interface. The PHY must first deassert RxDataValid immediately after the end of the Nth received block transmitted across the PIPE interface following reset or exit from electrical idle. Examples of timings for RxDataValid and other Rx related signals for a 16-bit wide interface are shown in Figure 8-40.

Figure 8-38. PCIe 8 GT/s or Higher TxDataValid Timing for 8 Bit-Wide TxData Interface

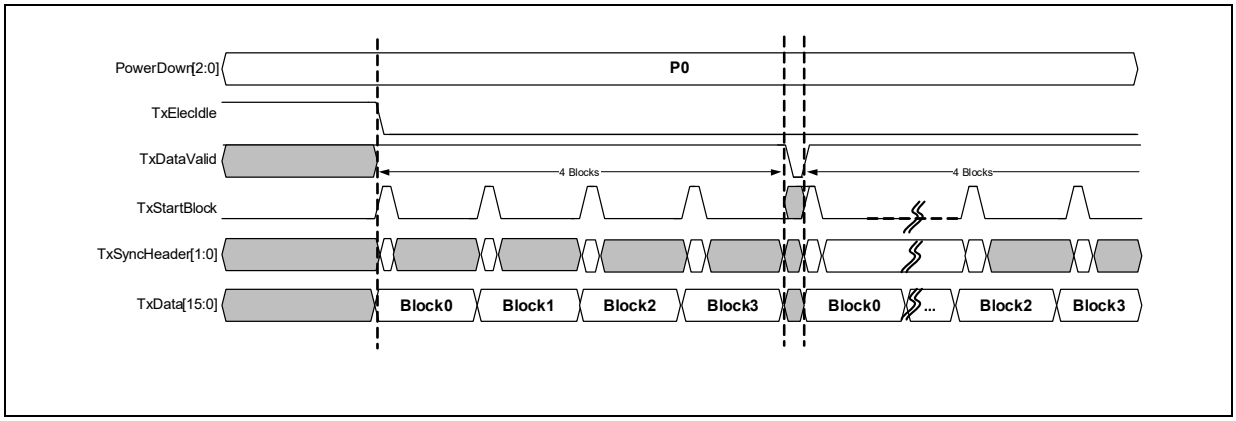


Figure 8-39. PCIe 8 GT/s or Higher TxDataValid Timing for 16 Bit-Wide TxData Interface

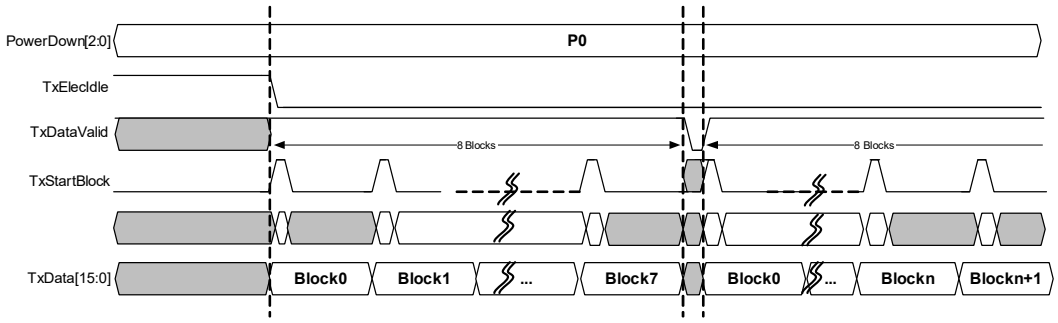
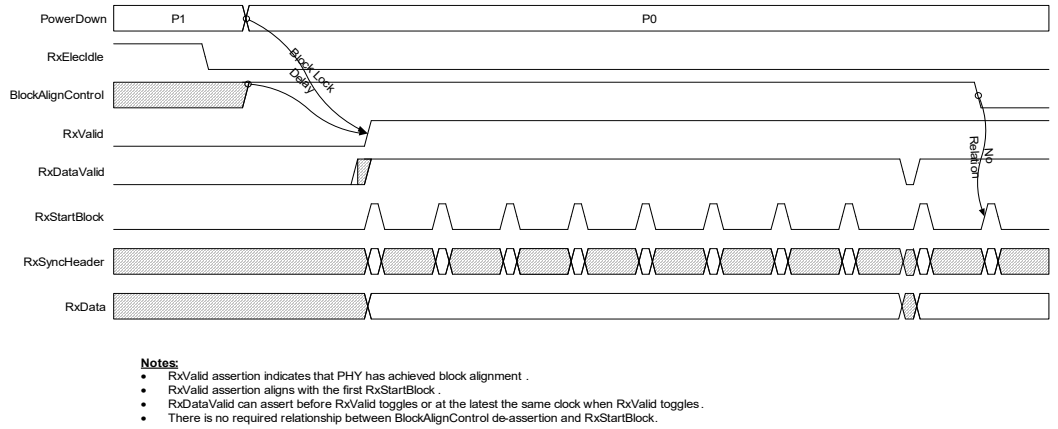


Figure: TxDataValid Timing for16-bit TxData Interface

Figure 8-40. PCIe 8 GT/s or Higher RxDataValid Timing for 16 Bit-Wide RxData Interface



Notes:
- RxValid assertion indicates that PHY has achieved block alignment .
- RxValid assertion aligns with the first RxStartBlock .
- RxDataValid can assert before RxValid toggles or at the latest the same clock when RxValid toggles.
- There is no required relationship between BlockAlignControl de-assertion and RxStartBlock.

There are situations, such as upconfigure or L0p, when a MAC must start transmissions on idle lanes while some other lanes are already active. In any such situation, the MAC must wait until the cycle after TxDataValid is deasserted to allow the PHY to transmit the backlog of data due to 128b/130b to start transmissions on previously idle lanes.

8.28 128b/132b Encoding and Block Synchronization (USB 10 GT/s)

For every 128 bits that are moved across the PIPE TxData interface at the 10.0 GT/s rate the PHY must transmit 132 bits. The MAC must use the TxDataValid signal periodically to allow the PHY to transmit the built-up backlog of data. For example – if the TxData bus is 16 bits wide and PCLK is 625 MHz then every four blocks the MAC must deassert TxDataValid for one PCLK to allow the PHY to transmit the 16 bit backlog of built up data. The buffers used by the PHY to store Tx data related to the 128/132b encoding rate mismatch must be empty when the PHY comes out of reset and must be empty whenever the PHY exits electrical idle (since Tx buffers are flushed before entry to idle). The PHY must use RxDataValid in a similar fashion. TxDataValid and

RxDataValid must be deasserted for one clock exactly every N blocks when the PIPE interface is operating at 10 GT/s, where N is 2 for an 8 bit wide interface, 4 for a 16 bit wide interface, and 8 for a 32 bit wide interface. The MAC must first deassert TxDataValid immediately after the end of the Nth transmitted block following reset or exit from electrical idle.

8.29 Message Bus Interface

8.29.1 General Operational Rules

The message bus interface can be used after Reset# is deasserted and PCLK is stable. The message bus interface must return to its idle state immediately upon assertion of Reset# and must remain idle until Reset# is deasserted and PhyStatus is deasserted, with the exception of LocalLF and LocalFS updates by the PHY as described in Section 9.8. Since the MAC is aware of when PCLK is stable, the requirement that PCLK must be an input to use the message bus allows the MAC to only issue transactions on the message bus after PCLK becomes stable.

For each write_committed issued, the initiator must wait for a write_ack response before issuing any new write_uncommitted or write_committed transactions. A sequence of write_uncommitted transactions must always be followed by a write_committed transaction; only a single write_ack response is expected. The initiator must ensure that the total number of outstanding writes, that is, writes issued since the last write_ack was received, must not exceed the write buffer storage implemented by the receiver.

Transmission of a write_ack must not depend on receiving a write_ack.

Only one read can be outstanding at a time in each direction. The initiator must wait for a read completion before issuing a new read since there are no transaction IDs associated with outstanding reads.

To facilitate design simplicity, reads and writes cannot be mixed. There must not be any reads outstanding when a write is issued; conversely, there must not be any writes outstanding when a read is issued. An outstanding write is any write_committed that has not received a write_ack or any write_uncommitted without a subsequent write_committed that has received a write_ack.

Posted-to-posted MAC to PHY writes are those that result in a PHY to MAC write to be generated in response. For simplification of the verification space, the MAC must only have one outstanding post-to-posted write that is waiting for a write in response. Table 8-8 lists the posted-to-posted writes generated by the MAC. Additionally, any vendor defined writes with posted-to-posted properties most conform to the same restriction of only one outstanding.

Table 8-8. Posted-to-Posted Writes

Post-to-Posted Register Write	PHY Write Generated in Response
Rx Margin Control0 register to stop/start margining	Rx Margin Status0
PHY Tx Control5 register to assert GetLocalPresetCoefficients	Tx Status0, Tx Status1, Tx Status2
PHY Rx Control3 register to assert RxEqEval	Rx Link Evaluation Status0 and Rx Link Evaluation Status1
Elastic Buffer Control	Elastic Buffer Status

Certain registers are defined as part of a register group. To simplify validation space, whenever one register in a register group needs to be updated, all the registers in the register group must be updated using a sequence of uncommitted writes and a single committed write. The defined register groups are listed in Table 8-9, where each row corresponds to a register group.

Table 8-9. Defined Register Groups

Register Groups (One per Row)
MAC Tx Status 0/1/2
PHY Tx Control 2/3/4
MAC Rx Status 0/1
MAC Rx Status 2/3
MAC Rx Status 4/5

8.29.2 Message Bus Operations vs. Dedicated Signals

For simplicity, dependencies between message bus operations and dedicated signals are kept to a minimum. The dependencies that do exist are there only because no acceptable workarounds for eliminating them have been identified; these dependencies are documented in this section:

- The PHY must wait for the write_ack to come back for any write to LocalLF, LocalFS, LocalG4LF, or LocalG4FS, if any, before it asserts PhyStatus for a rate change.

8.30 PCIe Lane Margining at the Receiver

Table 8-10 provides the sequence of PIPE message bus commands associated with various receiver margining operations; different sequences are shown for independent and dependent samplers.

Table 8-10. Lane Margining at the Receiver Sequences (Sheet 1 of 4)

Operation	Type of Sampler	Sequence		
		Direction	Message Bus Command	Description
Start Margining Success	Independent	M-->P	UWr	RxMarginControl1={1'b?,7'b?} (direction, offset)
		M-->P	CWr	RxMarginC6ontrol0=8'b000011?1 (clear error/sample and set start)
				Mac clears its error count snapshot
		P-->M	Ack	
				PHY clears its error and sample counters due to MAC setting Sample Count Reset and Error Count Reset bits in RxMarginControl0
		P-->M	UWr	RxMarginStatus1.SampleCount=0
		P-->M	UWr	RxMarginStatus2.ErrorCount=0
		P-->M	CWr	RxMarginStatus0.MarginStatus=1
		M-->P	Ack	
	Dependent	M-->P	UWr	RxMarginControl1={1'b?,7'b?} (direction, offset)
		M-->P	CWr	RxMarginControl0=8'b000011?1 (set start) (error/sample clears are a don't care)
				Mac clears its error count snapshot
		P-->M	Ack	
		P-->M	CWr	RxMarginStatus0.MarginStatus=1
		M-->P	Ack	
Offset Change Success	Independent	M-->P	UWr	RxMarginControl0=8'b000011?1 (clear error/sample counts)
		M-->P	CWr	RxMarginControl1={1'b?,7'b?} (direction, offset)
				Mac clears its error count snapshot
		P-->M	Ack	
				PHY clears its error and sample counters due to MAC setting Sample Count Reset and Error Count Reset bits in RxMarginControl0
		P-->M	UWr	RxMarginStatus1.SampleCount=0
		P-->M	UWr	RxMarginStatus2.ErrorCount=0
		P-->M	CWr	RxMarginStatus0.MarginStatus=1
		M-->P	Ack	
	Dependent	M-->P	CWr	RxMarginControl1={1'b?,7'b?} (direction, offset)
				Mac clears its error count snapshot
		P-->M	Ack	
		P-->M	CWr	RxMarginStatus0.MarginStatus=1
		M-->P	Ack	
Clear Error	Independent	M-->P	CWr	RxMarginControl0=8'b000001?1 (clear error, hold t vs v, maintain start)
		P-->M	Ack	
		P-->M	UWr	RxMarginStatus1.SampleCount=current
		P-->M	CWr	RxMarginStatus2.ErrorCount=0
		M-->P	Ack	
	Dependent			Mac clears its error count snapshot

Table 8-10. Lane Margining at the Receiver Sequences (Sheet 2 of 4)

Operation	Type of Sampler	Sequence		
		Direction	Message Bus Command	Description
Stop Margining	Independent	M-->P	UWr	RxMarginControl1={1'b?,7'b?} (direction, offset)
		M-->P	CWr	RxMarginControl0=8'b00000000 (stop, clear t vs v)
		P-->M	Ack	
		P-->M	UWr	RxMarginStatus1.SampleCount=Final
		P-->M	UWr	RxMarginStatus2.ErrorCount=Final
		P-->M	CWr	RxMarginStatus0.MarginStatus=1
		M-->P	Ack	
	Dependent	M-->P	UWr	RxMarginControl1={1'b?,7'b?} (direction, offset)
		M-->P	CWr	RxMarginControl0=8'b00000000 (stop, clear t vs v)
		P-->M	Ack	
		P-->M	CWr	RxMarginStatus0.MarginStatus=1
		M-->P	Ack	
Start Margining NAK	Independent	M-->P	UWr	RxMarginControl1={1'b?,7'b?} (direction, offset)
		M-->P	CWr	RxMarginControl0=8'b000011?1 (clear error/sample and start)
				Mac clears its error count snapshot
		P-->M	Ack	
				PHY clears its error and sample counters due to MAC setting Sample Count Reset and Error Count Reset bits in RxMarginControl0
		P-->M	UWr	RxMarginStatus1.SampleCount=0
		P-->M	UWr	RxMarginStatus2.ErrorCount=0
		P-->M	CWr	RxMarginStatus0.MarginNak=1
		M-->P	Ack	
				MAC changes execution status to 11 (NAK)
				The "Stop Margining" sequence should be followed.
	Dependent	M-->P	UWr	RxMarginControl1={1'b?,7'b?} (direction, offset)
		M-->P	CWr	RxMarginControl0=8'b000011?1 (set start) (error/sample clears are a don't care)
				Mac clears its error count snapshot
		P-->M	Ack	
				PHY detects bad margin request, places/keeps margin logic in normal functional operation mode
		P-->M	CWr	RxMarginStatus0.MarginNak=1
		M-->P	Ack	
				MAC changes execution status to 11 (NAK)
				The "Stop Margining" sequence should be followed.

Table 8-10. Lane Margining at the Receiver Sequences (Sheet 3 of 4)

Operation	Type of Sampler	Sequence		
		Direction	Message Bus Command	Description
Offset Change NAK	Independent	M-->P	UWr	RxMarginControl0=8'b000011?1 (clear error/sample counts)
		M-->P	CWr	RxMarginControl1={1'b?,7'b?} (direction, offset)
				Mac clears its error count snapshot
		P-->M	Ack	
				"PHY clears its error and sample counters due to MAC setting Sample Count Reset and Error Count Reset bits in RxMarginControl0. PHY detects bad offset, places/keeps margin logic in normal functional operation mode (margin off)"
		P-->M	UWr	RxMarginStatus1.SampleCount=0
		P-->M	UWr	RxMarginStatus2.ErrorCount=0
		P-->M	CWr	RxMarginStatus0.MarginNak=1
		M-->P	Ack	
				MAC changes execution status to 11 (NAK)
				The "Stop Margining" sequence should be followed.
	Dependent	M-->P	CWr	RxMarginControl1={1'b?,7'b?} (direction, offset)
				Mac clears its error count snapshot
		P-->M	Ack	
				PHY detects bad offset, places/keeps margin logic in normal functional operation mode (margin off)
		P-->M	CWr	RxMarginStatus0.MarginNak=1
		M-->P	Ack	
				MAC changes execution status to 11 (NAK)
				The "Stop Margining" sequence should be followed.
Error and Sample Counts Update (under limit)	Independent			PHY detects a change in error or sample count **Note:** Multiple updates may be combined into single write to avoid backlog.
		P-->M	UWr	RxMarginStatus1.SampleCount= new current
		P-->M	CWr	RxMarginStatus2.ErrorCount= new current
		M-->P	Ack	
				MAC changes execution status to new error/sample count
	Dependent			MAC detects a change in error count
				MAC changes execution status to new error count

Table 8-10. Lane Margining at the Receiver Sequences (Sheet 4 of 4)

Operation	Type of Sampler	Sequence		
		Direction	Message Bus Command	Description
Error Limit Exceeded	Independent			PHY detects a change in error or sample count
		P-->M	UWr	RxMarginStatus1.SampleCount= current
		P-->M	CWr	RxMarginStatus2.ErrorCount= new current
		M-->P	Ack	
				MAC compares error update to limit, detects limit exceeded
		M-->P	UWr	RxMarginControl1={1'b?,7'b?} (direction, offset)
		M-->P	CWr	RxMarginControl0=8'b00000000 (stop, clear t vs v)
		P-->M	Ack	
		P-->M	UWr	RxMarginStatus1.SampleCount=Final
		P-->M	UWr	RxMarginStatus2.ErrorCount=Final
		P-->M	CWr	RxMarginStatus0.MarginStatus=1
		M-->P	Ack	
				MAC changes execution status to 00 (error limit exceeded)
	Dependent			MAC observes error count has exceeded limit
		M-->P	UWr	RxMarginControl1={1'b?,7'b?} (direction, offset)
		M-->P	CWr	RxMarginControl0=8'b00000000 (stop, clear t vs v)
		P-->M	Ack	
		P-->M	CWr	RxMarginStatus0.MarginStatus=1
		M-->P	Ack	
				MAC changes execution status to 00 (error limit exceeded)
Sample Count Saturated	Independent			PHY detects a change in error or sample count
		P-->M	UWr	RxMarginStatus1.SampleCount= ==7'h7F
		P-->M	CWr	RxMarginStatus2.ErrorCount= new current
		M-->P	Ack	
	Dependent			N/A

8.31 Short Channel Power Control

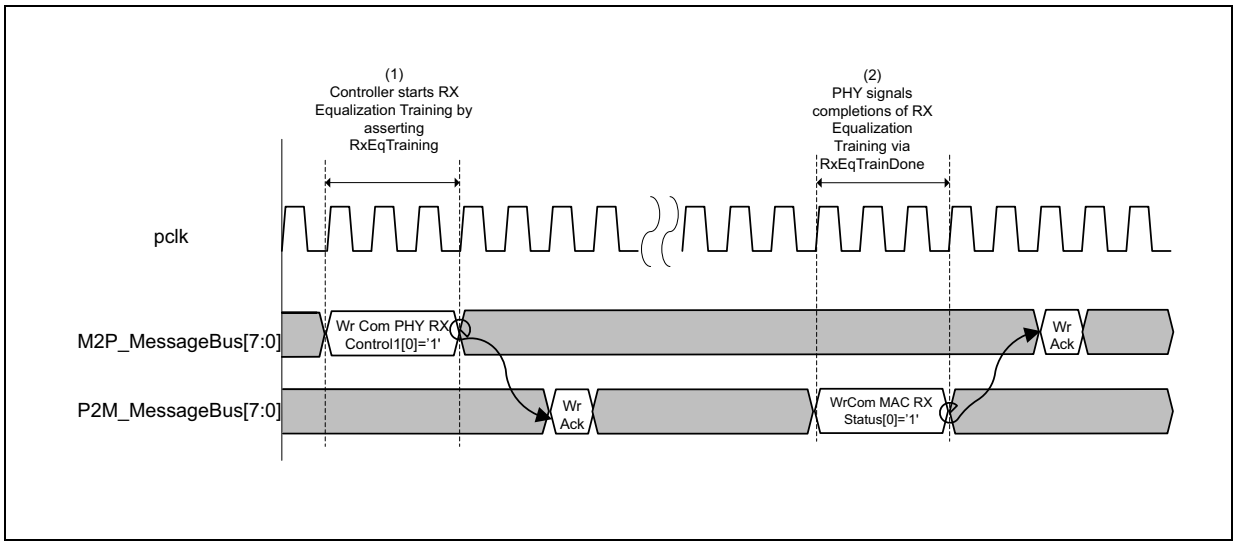
For short reach (for instance, MCP applications), there should be a provision to revert the ShortChannelPowerControl[1:0] signal to normal operation mode for situations where an optimized mode setting prevents link up. For example, if a particular setting is not compatible with a 2.5GT/s link speed and works only at higher link speeds, the expectation is that the ShortChannelPowerControl[1:0] signal would be set to normal operation mode to bring the link up initially before changing the value to an optimized power control setting while transitioning to higher link speeds.

8.32 RxEqTraining

For PCIe, there are several scenarios where the controller may request that the PHY perform receiver equalization. These may be in response to far end transmitter coefficient changes, loopback entry, rate changes, and support of no equalization on

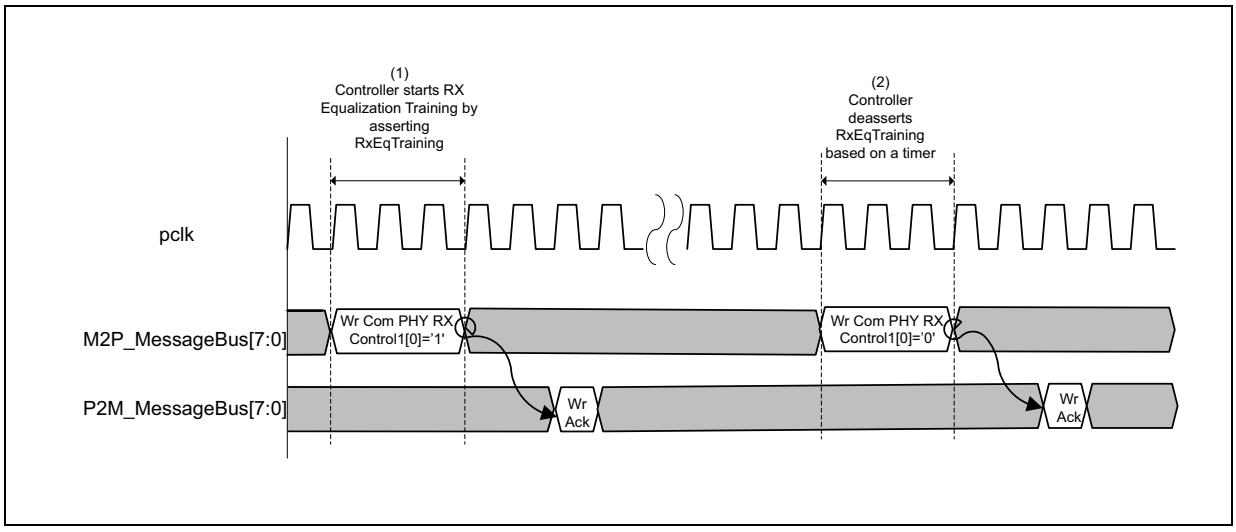
the transmitter side. For PCIe, the PHY sets the RxEqTrainDone bit in the Rx Status0 register to indicate completion of receiver equalization. Figure 8-41 shows the message bus sequence for managing PCIe receiver equalization.

Figure 8-41. PCIe Receiver Equalization



For USB, the controller instructs the PHY to perform receiver equalization during Polling.ExEQ. For USB, receiver equalization is timer based and the RxEqTrainDone bit is not used. Figure 8-42 shows the message bus sequence for USB receiver equalization.

Figure 8-42. USB Receiver Equalization



8.33 PHY Recalibration

In certain situations, the PHY may need to be recalibrated. These situations may include changes in operating conditions, for instance, Vref changes, or detection of certain error conditions. The PIPE specification provides mechanisms for either the controller or the PHY to initiate recalibration. Recalibration must occur during Recovery, so if the PHY determines that a recalibration is necessary, it notifies the controller that it should enter Recovery and request a recalibration. Figure 8-43 shows the sequence of message bus commands for a controller initiated PHY recalibration. Figure 8-44 shows the sequence of message bus commands for a PHY initiated PHY recalibration; this sequence essentially consists of the PHY notifying the controller that it should request a recalibration, then the controller follows the same steps as it would for a controller initiated PHY recalibration. After the PHY notifies the controller that the recalibration operation is complete by setting the IORecalDone bit, the controller is permitted to exit Recovery and resume normal operation on the link.

Figure 8-43. PHY Recalibration Initiated by Controller

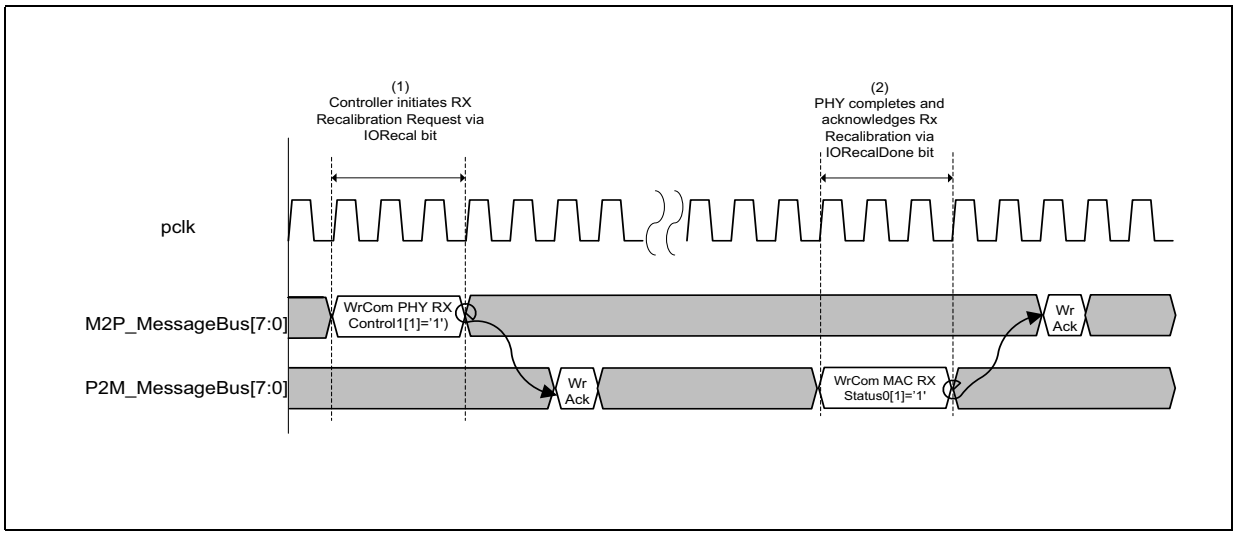
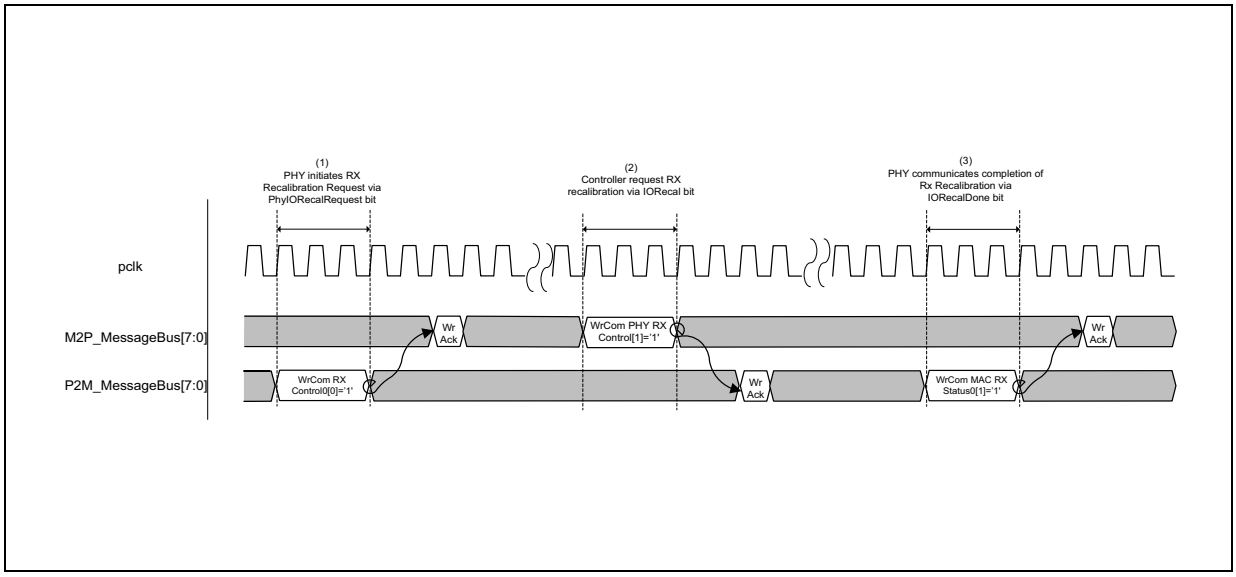


Figure 8-44. PHY Recalibration Initiated by PHY



8.34 Digital Near End Loopback

The PIPE specification defines an optional Digital Near-End Loopback (DNELB) operational mode to facilitate HVM testing; toggling signals via functional testing in loopback mode enables fault testing. This feature is applicable to PCIe, USB, USB4, and SATA.

Several possible loopback points from the transmit to the receive datapath are recommended as shown in Figure 8-45 and Figure 8-46. Figure 8-25 shows loopback paths in a PHY with original PIPE architecture; LB0 through LB4 are recommended paths and correspond to encodings defined in the PHY Near End Loopback Control register (See Section 7.1.23), while LB5 is a potential PHY implementation specific path. The loopback paths may require logic, represented by *f* in the diagrams, to convert between the Tx and Rx paths clock frequencies and data width. Figure 8-46 shows loopback paths in a PHY with SerDes architecture; L0, L3, and LB4 are recommended paths and correspond to encodings defined in the PHY Near End Loopback Control register (See Section 7.1.23), while LB5 is a potential PHY implementation specific path.

Figure 8-45. Original PIPE Architecture: DNELB Path Examples

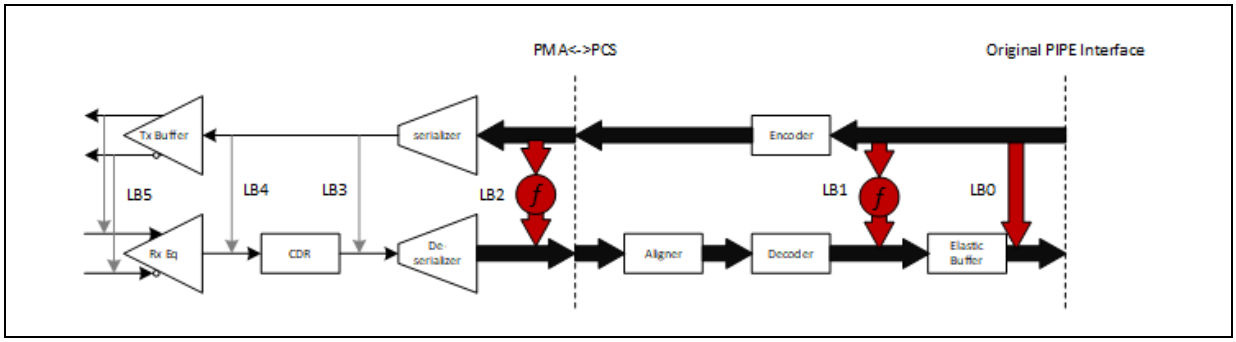
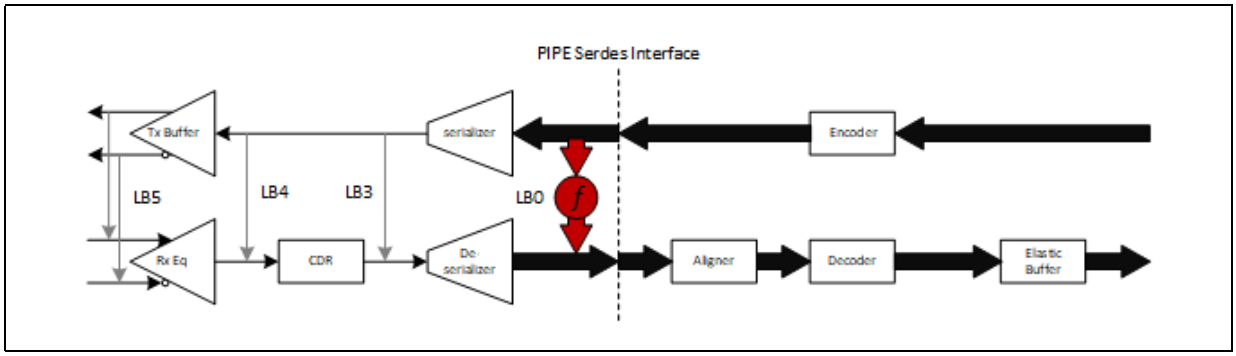


Figure 8-46. SerDes Architecture: DNELB Path Examples



The following subsections specify the sequences for entering and exiting DNELB mode; also specified are any differences in behavior of PIPE signals when operating in DNELB mode.

8.34.1 PIPE Operations and Signals in DNELB Mode

The general philosophy is to enable the LTSSM to train as closely as possible to normal operational mode. Unless differences are called out specifically in this section, PIPE signals operate the same as they do in normal operation.

- Data Path:
  - Original PIPE Architecture
    - **Rate**, **Width**, and **PCLK Rate** operate the same in DNELB mode as they do in normal operation
    - Tx interface operates same as in normal operation
      - No additional limitations on **TxDataValid** or **TxElecIdle**
- Rx interface operates same as in normal operation
  - **TxDataValid** and **TxElecIdle** have impact on **RxDataValid** and **RxElecIdle**; the loopback location determines how the Tx side translates to the Rx side
- SerDes Architecture:
  - **Rate**, **Width**, **PCLK Rate**, and **RxWidth** operate the same in NELB mode as in normal operation
  - Tx interface operates the same as in normal operation
    - No additional limitations on **TxDataValid** or **TxElecIdle**
  - Rx interface operates the same as in normal operation
    - **RxValid**, **RxData**, and **RxCLK** operate the same as in normal operation
  - PHY must convert the Tx data path into the expected Rx data path format:
    - Convert from **PCLK** / **TxDataValid** / **Width** to **RxCLK** / **RxWidth** format
    - If **RxCLK** is slower than **PCLK**, the PHY is permitted to skew the duty cycle; however, the shorter of the low or high portion of the clock period must not be shorter than normal operation. For example, if **RxCLK** is 250MHz and **PCLK** is 1GHz, acceptable duty cycles include 1 ns/3 ns or 500 ps/3.5 ns. An asymmetric duty cycle enables simplification of generation of **RxCLK** via a simple digital divide of incoming **PCLK**.

- **RxValid**:
  - The assertion of **RxValid** should continue to be affected by the Rx logic downstream of the loop back point. For instance, if alignment (8b/10b or 128/130b) must occur first before **RxValid** is asserted, then it must be required in NELB when the alignment logic is in the active Rx data path.
  - In SerDes mode, **RxValid** must assert when the **RxCLK** is stable and the **TxElecIdle** is no longer asserted.
- **RxElecIdle**:
  - The **RxElecIdle** signal must reflect the **TxElecIdle** signal with minimal delay, especially at the end of the data stream. If **RxElecIdle** is not asserted soon enough after **TxElecIdle**, this may result in false wakeups from L1. The **RxElecIdle** signal should not lag the **RxData** by more than normal operation would. **RxElecIdle** is permitted to precede the **RxData** if that is how normal operation would happen.
- **RxStatus**:
  - For **TxDetectRx** functionality, see following entries.
  - For error and skip adjustment status
    - If bad data will be transferred on initial EI exit, mark as required by protocol with **RxStatus**
      - If the loopback point is in the analog or high speed digital domains that may be susceptible to bit errors, **RxStatus** should reflect these as required by the protocol mode
      - Note: The PHY may need to be tuned to avoid errors if a loopback path in the analog or high-speed digital domain is selected.
    - If a skip adjustment is done, **RxStatus** must reflect such action.
- Tx Detect Receiver:
  - The results of doing Tx receiver detect will be equal to the state of **RXTermination** (**RxStatus** of 3)
  - If asynchronous **TxDetectRx** is supported in normal operation, it must also be supported in NELB mode.
- RxEqEval / RxEqTrain:
  - Depending on loopback position, these operations may return fabricated dummy results. The PHY must indicate in its datasheet which loop back positions result in fabricated dummy results.
  - To enable the MAC to consume the results in the same manner as it does in normal operation, the PHY must adhere to the following rules when returning fabricated dummy results:
    - Directional feedback must return no update required (0)
    - Figure of Merit Feedback must return a non-zero value
      - It is recommended that the MAC have the ability to shorten its search algorithms in NELB mode.
    - RxEqEval / RxEqTrain response time in NELB mode must not exceed 10 us.
- Powerdown / Phystatus:
  - This operates the same as in normal operation.
- RxStandby / RxStandbyStatus:
  - This operates the same as in normal operation:

    - If RxStandbyStatus returns in response to RxStandby, it should continue to do so in NELB mode.
    - If RxStandbyStatus is not supported (for instance, USB) or guaranteed, the same holds true in NELB mode.
- Rate Changes:
  — This operates the same as in normal operation.
  — The same combinations of **Rate**, **Width**, **PCLK Rate**, **RxWIDTH**, and so forth. It must be supported in NELB mode as in normal operation.
  — Clock changes related to rate changes should remain the same as in normal operation.
    - For instance, **PclkChangeOk** / **PclkChangeAck** should remain the same
- **TxDetectRx/Loopback** must not be used to indicate follower loopback operation while NELB is enabled
- Start of data transfer on Rx data path:
  — If the PHY cannot guarantee all bits will be looped back when the Rx data path is exiting electrical idle, it must clearly specify in its datasheet when the first data will appear.
    - for instance, The block aligner requires two EIEOS's to begin forwarding data and the data from the Tx data path up to the 2nd EIEOS would not be seen (similar thing for 8b/10b aligner)
- NELB must function when **SRISEnable** is high or low:
  — No false or additional ppm is required to be applied if **SRISEnable** is high
- **RefClkRequired#** must function as it does in normal operation
- **DataBusWidth** must function as it does in normal operation
- **TxDeemph** and **TxSwing** must continue to be consumed by the PHY. The PHY may choose to ignore them if the loopback point is not affected by the Tx EQ settings.
- The controller must not use the following functions while in NELB as the PHY may not return a response:
  — Receiver Lane Margining
  — TxMargin
  — IORecal
- Elastic Buffer controls as defined in the message bus section (depth, run mode, and so forth) must continue to function the same as normal operation
- BlockAlignControl functionality is expected to function the same as in normal operation
- EncodeDecodeBypass must operate as it does in normal operation
- **RxPolarity** functionality must continue to operate the same as in normal operation:
  — PHY or MAC can implement implementation specific means to invert the data for increased test coverage
- Local Preset Fetch bus must function as it does in normal operation:
  — GetLocalPresetCoefficients, LocalPresetIndex, LocalFS, LocalLF, LocalG{5,4}FS, LocalG{5,4}LF, LocalTxPresetCoefficient
- FS and LF of link partner must continue to be reported as in normal operation:

- In NELB mode, the controller must not perform any operations that require the use of the following:
  - **RxEIDetectDisable**
  - **TxCommonModeDisable**
  - **AsyncPowerChangeAck**
- **AlignDetect** should continue to follow normal operation
- **Tx Pattern** must continue to follow normal operation
- **PowerPresent** must be set high in NELB mode
- **TxOneZeros** must be low in NELB mode

8.34.2 Entry and Exit from NELB Mode

The handshake sequence for Entry into NELB Mode is as follows:

1. MAC releases PIPE lane reset
2. MAC moves to proper PowerDown state
3. MAC writes to the PHY NELB Control register with position desired and enable set (PHY sends write Ack)
4. PHY does its internal setup for NELB
5. PHY writes to the NELB Status register in the MAC, setting **NELB State** to 1 (MAC sends write Ack)
6. MAC now free to train in NELB

The handshake sequence for exit from NELB is as follows:

- Option 1: MAC asserts PIPE lane reset OR
- Option 2: (The PHY must specify in its datasheet whether it supports this exit method)
  1. Mac moves to proper PowerDown state
  2. MAC sends **NELB Control** message to PHY with enable cleared (PHY sends write Ack back)
  3. PHY does set up to move back to normal operation
  4. PHY sends **NELB Acknowledgment** message to MAC, setting **NELB State** to 0 (MAC sends write Ack back)
  5. MAC now free to train in normal operation

Handshake rules:

- Handshake must occur in a PowerDown state that has both Tx and Rx off but pclk running
- Handshake must complete prior to doing transmitter receiver detect if TxdetectRx is to be used
- Handshake may be done at any data rate if supported by PHY. The PHY must support the handshake at initial protocol defined rate (for instance, 2.5GT/s for PCIe)
- If PHY is to report an error in the NELB Status register on entry, the state must indicate **out of NELB**, and the PHY must remain functionally in normal mode

- The MAC is encouraged to use the exit error indication to initiate a PIPE lane reset or otherwise block functional operation.
- The **LB Position** cannot be changed while in NELB. NELB must be exited and re-entered with the new **LB Position**.

8.34.3 USB4 PAM3 Encoding on PIPE Interface

Each pair of binary bits on the TxData[55:0], TxData2[55:0], RxData[55:0], or RxData2[55:0] interface represents a PAM3 analog level. The mapping follows the USB4 specification:

- 0 maps to the lower voltage level, V_{-1}
- 1 maps to the middle voltage level, V_0
- 2 maps to the upper voltage level, V_1

The 56-bit data interface represents four symbols. Bits [13:0] comprise the first symbol, bits [27:14] comprise the second symbol, bits [41:28] comprise the third symbol, and bits [55:42] comprise the third symbol.

Within each symbol, bits[1:0] are trit 0 of the symbol, bits [3:2] are trit 1 of the symbol, bits [5:4] are trit 2 of the symbol, and so forth. Please refer to the USB4 specification for further details.

8.35 Switching Between Rx and Tx Pairs

USB4 supports asymmetric mode. To transition between symmetric and asymmetric mode requires switching the direction of a differential Rx/Tx pair. The message bus sequence for switching from Rx to Tx operation is shown in Figure 8-47; and the sequence for switching from Tx to Rx direction is specified in Figure 8-48. The traffic on all other differential pairs must not be impacted by this transition. This transition is permitted in any PowerDown state in which the message bus is operational; however, at a minimum, it must be supported in PS0.

Figure 8-47. Transitioning from Rx to Tx Operation

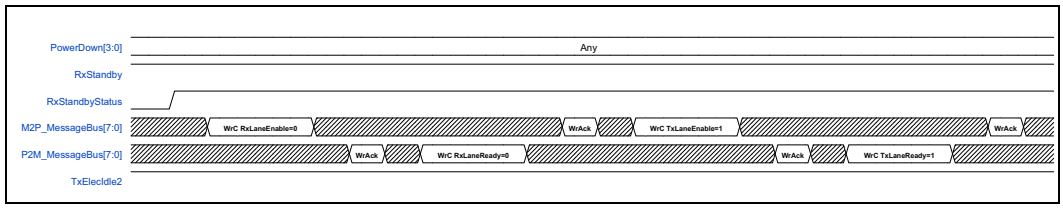
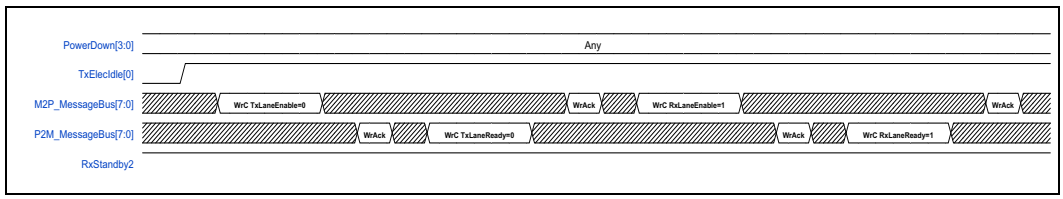


Figure 8-48. Transitioning from Tx to Rx Operation



9 Sample Operational Sequences

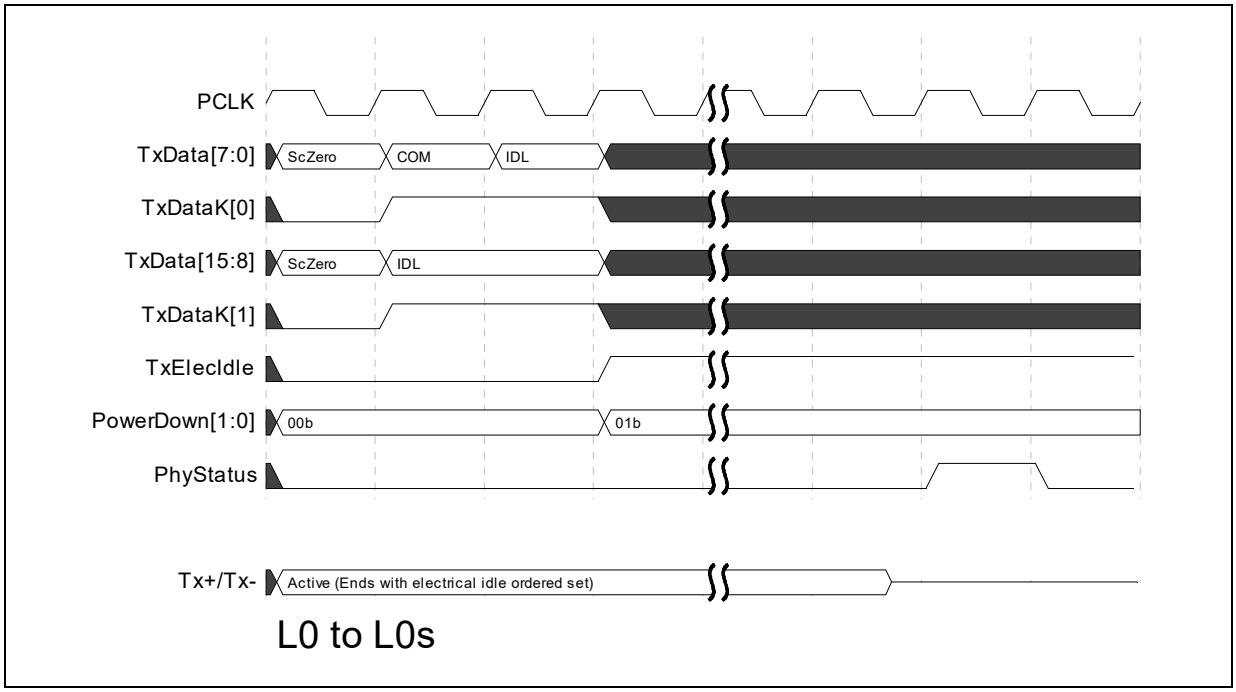
These sections show sample timing sequences for some of the more common PCIe, SATA, and USB operations. These are **sample** sequences and timings and are not required operation.

9.1 Active PM L0 to L0s and Back to L0 – PCIe Mode

This example shows one way a PIPE PHY can be controlled to perform Active State Power Management on a link for the sequence of the link being in L0 state, transitioning to L0s state, and then transitioning back to L0 state.

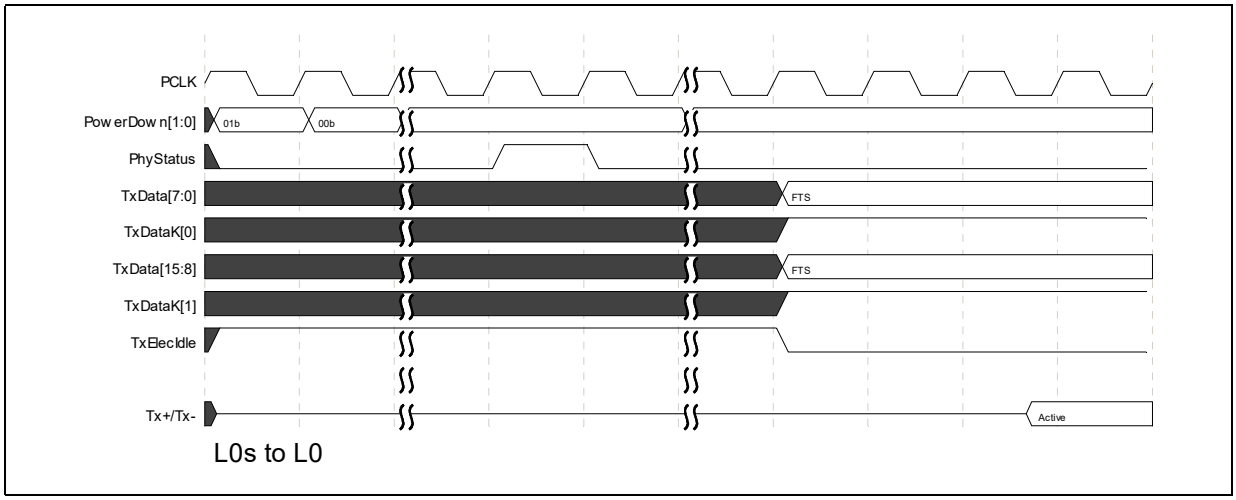
When the MAC and higher levels have determined that the link should transition to L0s, the MAC transmits an electrical idle ordered set and then has the PHY transmitter go idle and enter P0s. Note that for a 16-bit or 32-bit interface, the MAC should always align the electrical idle on the parallel interface so that the COM symbol is in the low-order position (**TxDataK[7:0]**).

Figure 9-1. L0 to L0s



To cause the link to exit the L0s state, the MAC transitions the PHY from the P0s state to the P0 state, waits for the PHY to indicate that it is ready to transmit (by the assertion of **PhyStatus**), and then begins transmitting Fast Training Sequences (FTS). Note: This is an example of L0s to L0 transition when the PHY is running at 2.5 GT/s.

Figure 9-2. L0s to L0

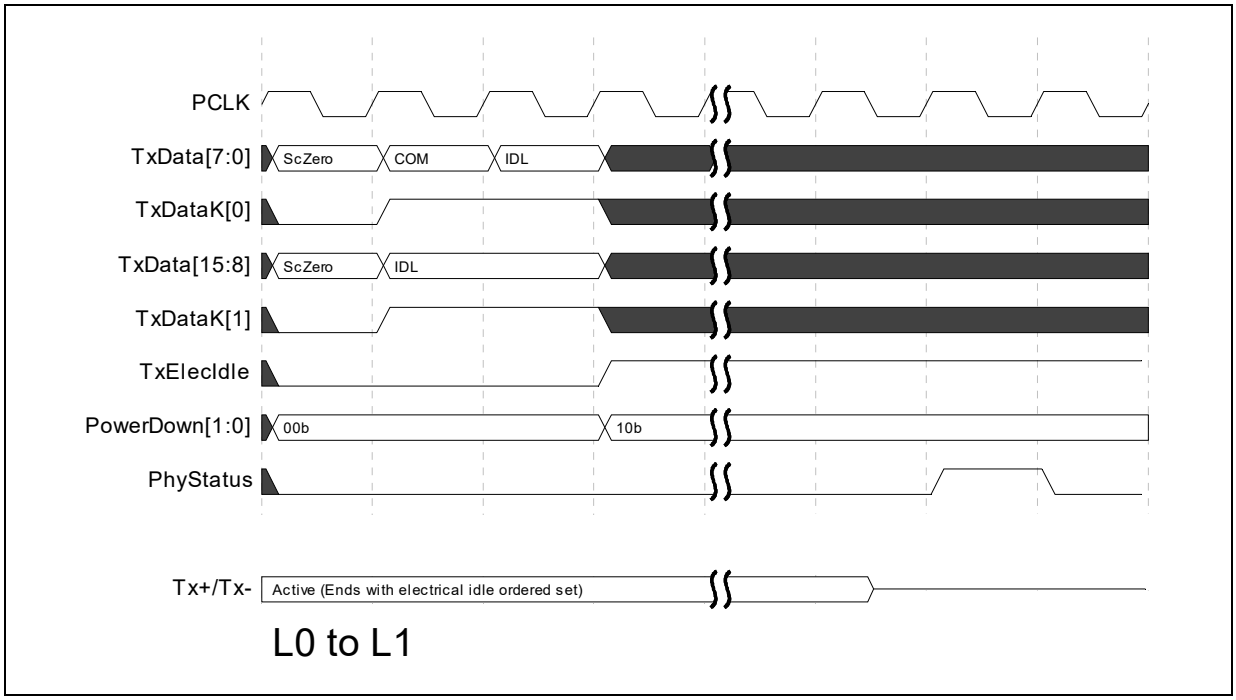


9.2 Active PM to L1 and Back to L0 - – PCIe Mode

This example shows one way a PIPE PHY can be controlled to perform Active State Power Management on a link for the sequence of the link being in L0 state, transitioning to L1 state, and then transitioning back to L0 state. This example assumes that the PHY is on an endpoint (that is, it is facing upstream) and that the endpoint has met all the requirements (as specified in the base specification) for entering L1.

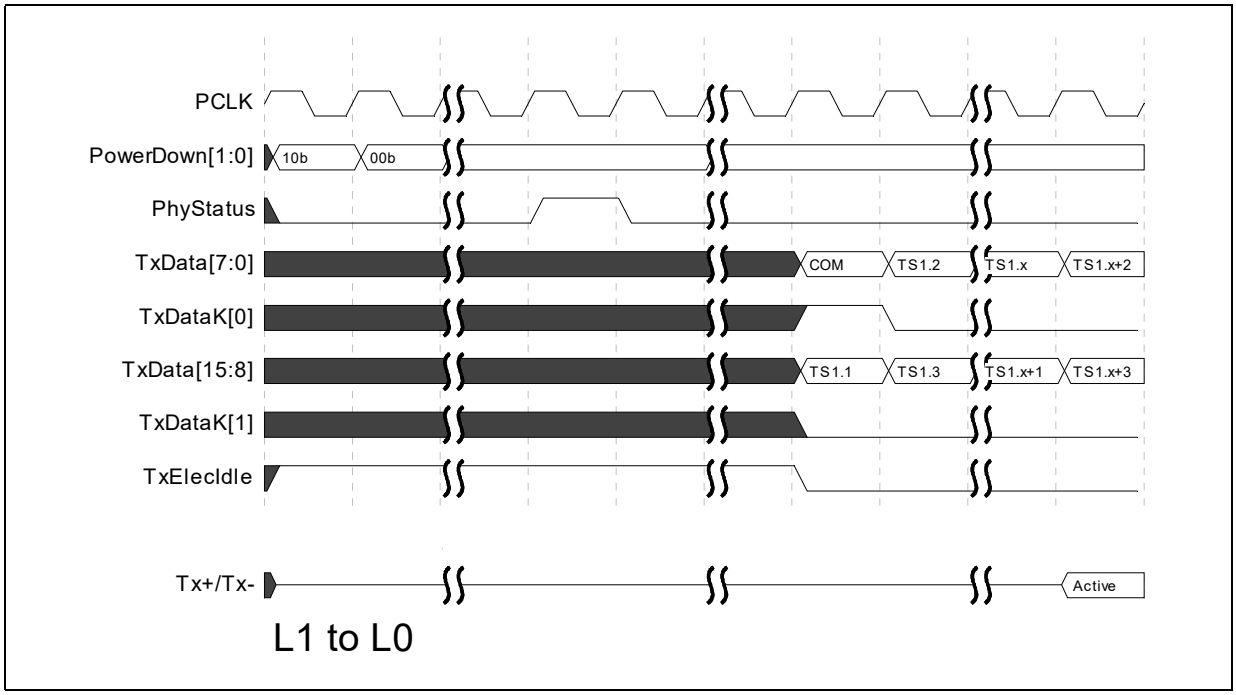
After the MAC has had the PHY send PM_Active_State_Request_L1 messages, and has received the PM_Request_ACK message from the upstream port, it then transmits an electrical idle ordered set, and has the PHY transmitter go idle and enter P1.

Figure 9-3. L0 to L1



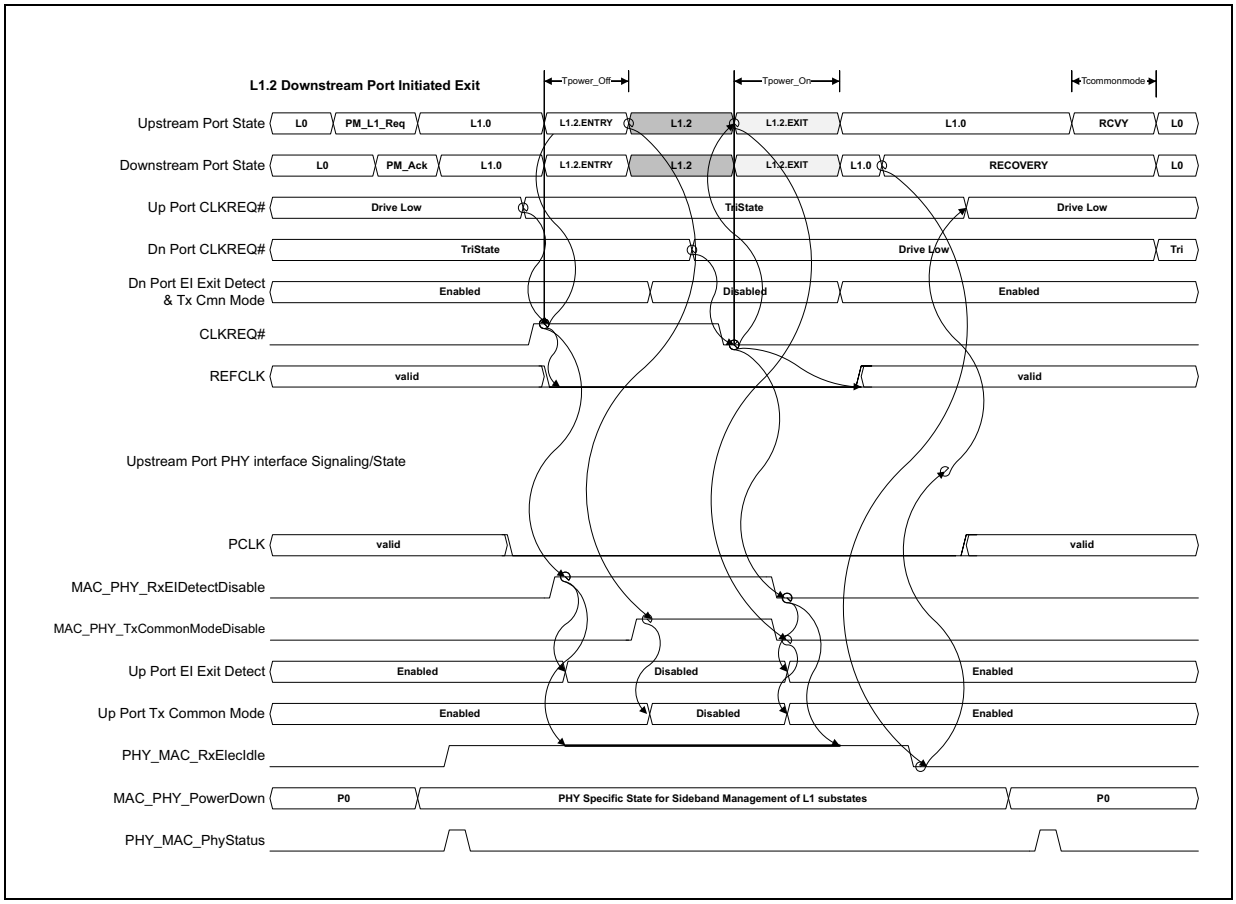
To cause the link to exit the 1 state, the MAC transitions the PHY from the P1 state to the P0 state, waits for the PHY to indicate that it is ready to transmit (by the assertion of **PhyStatus**), and then begins transmitting training sequence ordered sets (TS1s). Note, this is an example when the PHY is running at 2.5 GT/s.

Figure 9-4. L1 to L0



9.3 Downstream Initiated L1 Substate Entry Using Sideband Mechanism

Figure 9-5. L1 Substate Management Using RxEIDetectDisable and TxCommonModeDisable



9.4 Receivers and Electrical Idle – PCIe Mode Example

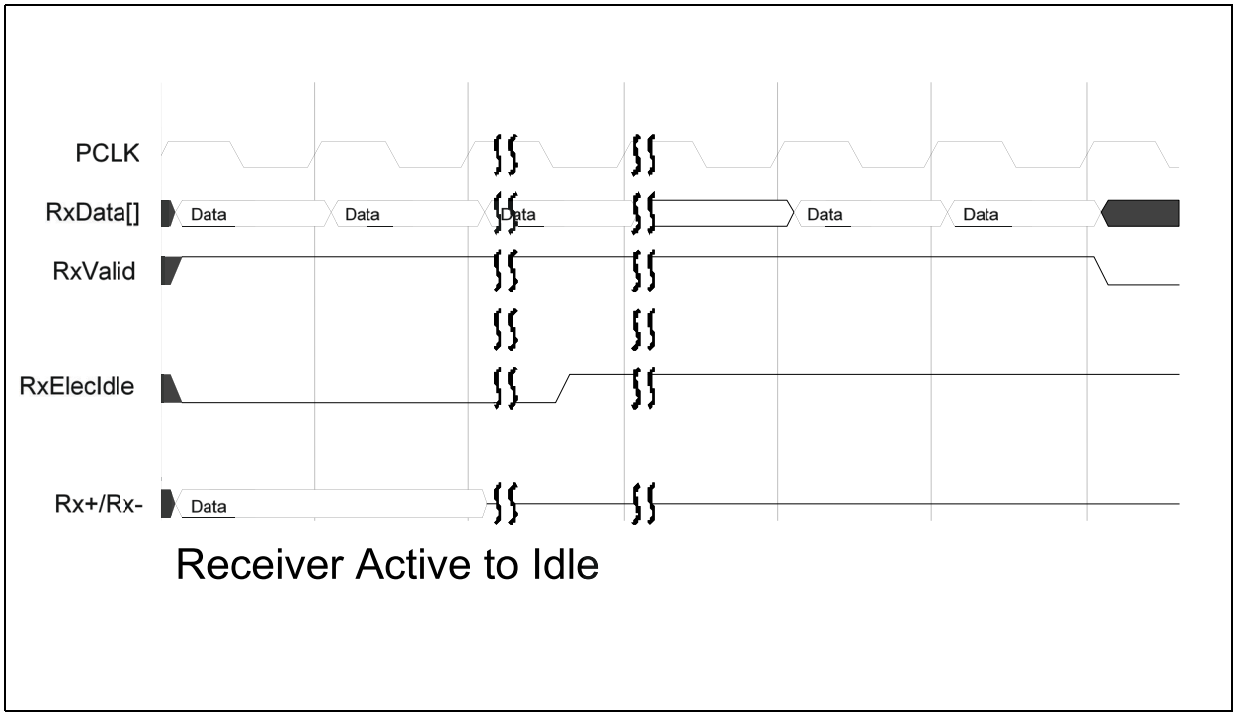
This section only applies to a PHY operating to 2.5 GT/s. Note that when operating at 5.0 GT/s or 8 GT/s signaling rates, **RxElecIdle** may not be reliable. MACs should see the PCIe Revision 3.0 base specification or USB 3.0 specification for methods of detecting entry into the electrical idle condition.

See Section 6.1.3 for the definition of **RxElecIdle** when operating at 5.0 GT/s. This section shows some examples of how PIPE interface signaling may happen as a receiver transitions from active to electrical idle and back again. In these transitions, there may be a significant time difference between when **RxElecIdle** transitions and when **RxValid** transitions.

The first diagram shows how the interface responds when the receive channel has been active and then goes to electrical idle. In this case, the delay between **RxElecIdle** being asserted and **RxValid** being deasserted is directly related to the depth of the

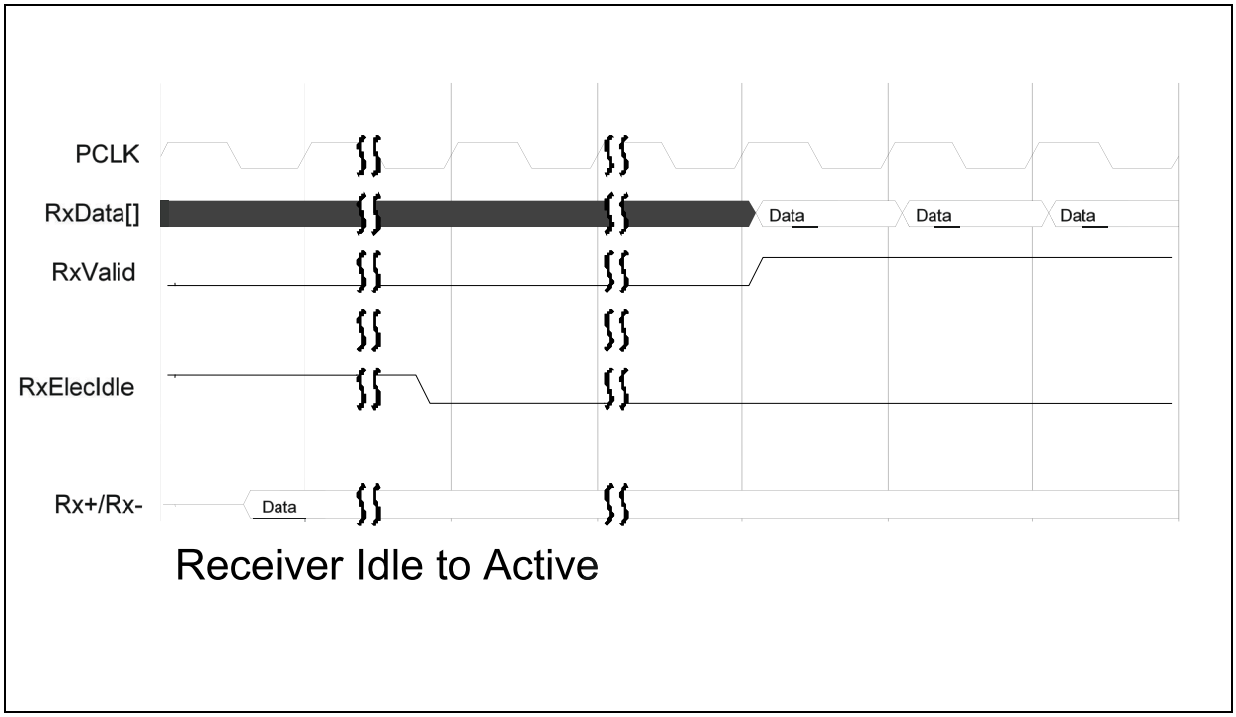
implementations elastic buffer and symbol synchronization logic. Note that the transmitter that is going to electrical idle may transmit garbage data and this data will show up on the **RxData[]** lines. The MAC should discard any symbols received after the electrical idle ordered set until RxValid is deasserted.

Figure 9-6. Receiver Active to Idle



The second diagram shows how the interface responds when the receive channel has been idle and then begins signaling again. In this case, there can be significant delay between the deassertion of **RxElecIdle** (indicating that there is activity on the **Rx+/ Rx-** lines) and RxValid being asserted (indicating valid data on the **RxData[]** signals). This delay is composed of the time required for the receiver to retrain as well as elastic buffer depth.

Figure 9-7. Receiver Idle to Active



9.5 Using CLKREQ# with PIPE – PCIe Mode

CLKREQ# is used in some implementations by the downstream device to cause the upstream device to stop signaling on REFCLK. When REFCLK is stopped, this will typically cause the CLK input to the PIPE PHY to stop as well. The PCIe CEM specification allows the downstream device to stop REFCLK when the link is in either L1 or L2 states. For implementations that use CLKREQ# to further manage power consumption, PIPE compliant PHYs can be used as follows:

The general usage model is that to stop REFCLK, the MAC puts the PHY into the P2 power state, then deasserts CLKREQ#. To get the REFCLK going again, the MAC asserts CLKREQ#, and then after some PHY and implementation specific time, the PHY is ready to use again.

9.5.1 CLKREQ# in L1

If the MAC is moving the link to the L1 state and intends to deassert CLKREQ# to stop REFCLK, then the MAC follows the proper sequence to get the link to L1, but instead of finishing by transitioning the PHY to P1, the MAC transition the PHY to P2. Then the MAC deasserts CLKREQ#.

When the MAC wants to get the link alive again, it can:

- Assert CLKREQ#
- Wait for REFCLK to be stable (implementation specific)
- Wait for the PHY to be ready (PHY specific)
- Transition the PHY to P0 state and begin training.

9.5.2 CLKREQ# in L2

If the MAC is moving the link to the L1 state and intends to deassert CLKREQ# to stop REFCLK, then the MAC follows the proper sequence to get the link to L2. Then the MAC deasserts CLKREQ#.

When the MAC wants to get the link alive again, it can:

- Assert CLKREQ#
- Wait for REFCLK to be stable (implementation specific)
- Wait for the PHY to be ready (PHY specific)
- Transition the PHY to P0 state and begin training.

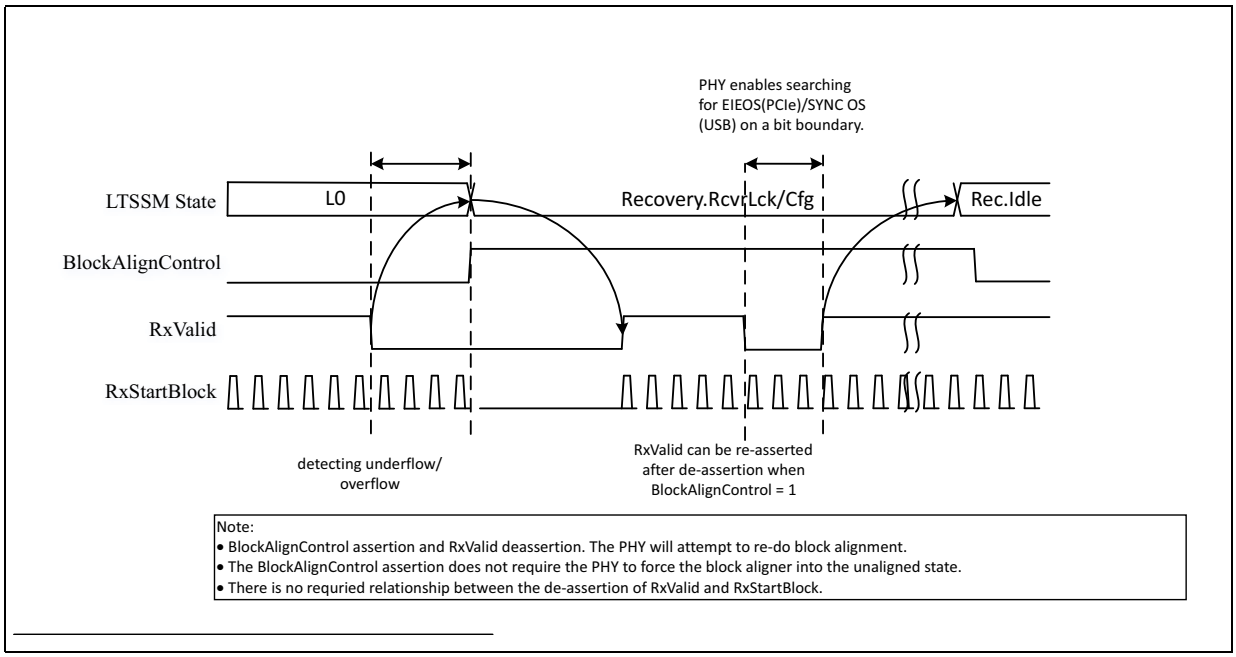
9.5.3 Delayed CLKREQ# in L1

The MAC may want to stop REFCLK after the link has been in L1 and idle for a while. In this case, the PHY is in the P1 state and the MAC must transition the PHY into the P0 state, and then the P2 state before deasserting CLKREQ#. Getting the link operational again is the same as the preceding cases.

9.6 Block Alignment

Figure 9-8 provides an example of a block alignment sequence using the BlockAlignControl pin. The PHY attempts to do alignment when BlockAlignControl is asserted and the PHY receiver is active.

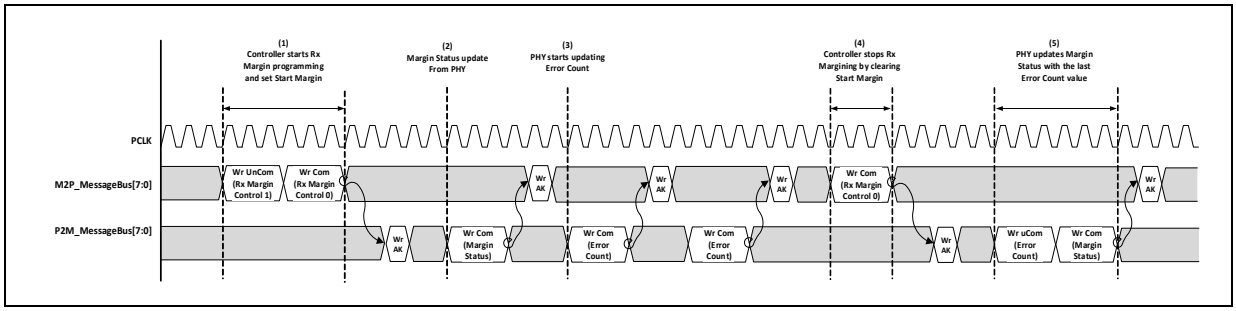
Figure 9-8. BlockAlignControl Example Timing



9.7 Message Bus: Rx Margining Sequence

Figure 9-9 shows an example of an Rx margining sequence. The MAC issues a write_uncommitted to address 0x1 followed by a write_committed to address 0x0 to set up the margining parameters and to start margining in the Rx Margin Control1 and Rx Margin Control0 registers. The PHY issues a write_ack to acknowledge that it has flushed the write buffer. Subsequently, upon processing a change in the "Start Margin" bit of the Rx Margin Control0 register, the PHY issues a write_committed to address 0x0 to assert the "Margin Status" bit. During the margining process, the PHY periodically issues write_committed transactions to address 0x2 to update the "Error Count[3:0]" value. The MAC acknowledges receipt of these writes by issuing corresponding write_ack transactions. Finally, the MAC stops the margining process by issuing a write_committed to address 0x0 to deassert the "Start Margin" bit. The PHY issues a write_ack to acknowledge that it has flushed the write buffer. In response to the "Start Margin" deassertion, the PHY pushes its final "Error Count[3:0]" value to the MAC via a write_uncommitted transaction to the "Rx Margin Status2" register, and then issues a write_committed to assert "Rx Margin Status0.Margin Status".

Figure 9-9. Sample Rx Margining Sequence



9.8 Message Bus: Updating LocalFS/LocalLF and LocalG4FS/LocalG4LF

Figure 9-10 shows a sequence where LocalFS and LocalLF are updated out of reset and, subsequently, LocalG4FS and LocalG4LF are updated after a rate change. Note that PhyStatus deasserts only after the write_ack returns for the LocalFS and LocalLF update out of reset. Similarly, the one cycle PhyStatus assertion occurs after the write_ack returns for the LocalG4FS and LocalG4LF update after a rate change. This is one of the rare cases where a dependency between a message bus operation and a dedicated signal exists. While this example shows LocalG4FS and LocalG4LF being updated after a rate change, it is not a requirement to wait until after the rate change to update these values; for instance, they can be updated out of reset if their values are already known by then. Note, this flexibility in timing of when updates can occur was intentionally introduced with the low pin count interface by allocating separate LocalFS and LocalLF registers per data rate.

Figure 9-10. LocalFS/LocalLF/LocalG4FS/LocalG4LF Updates Out of Reset and After Rate Change

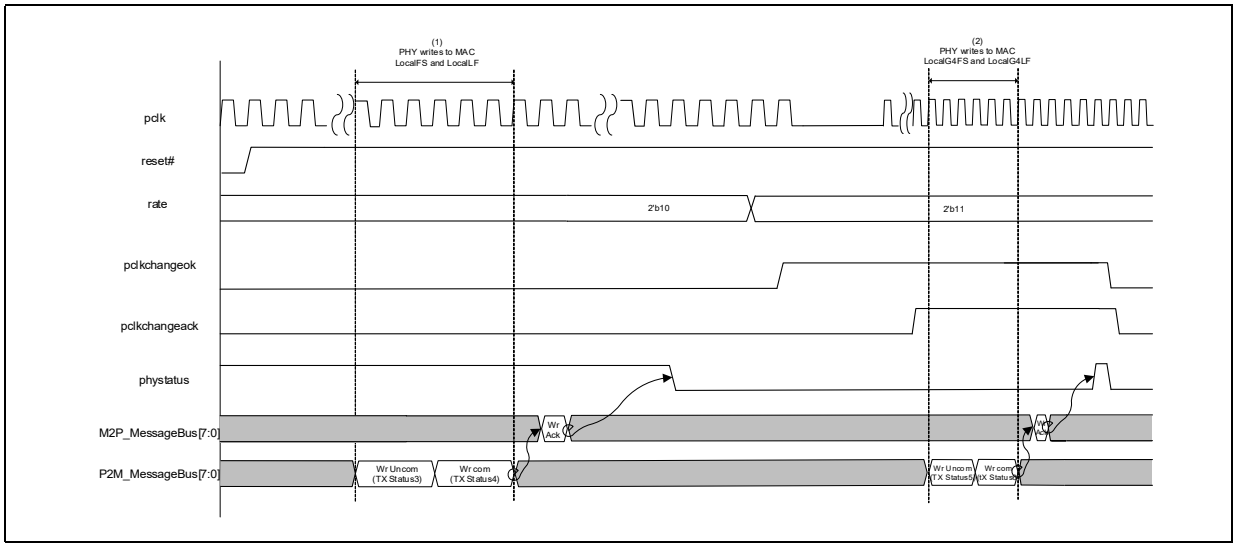
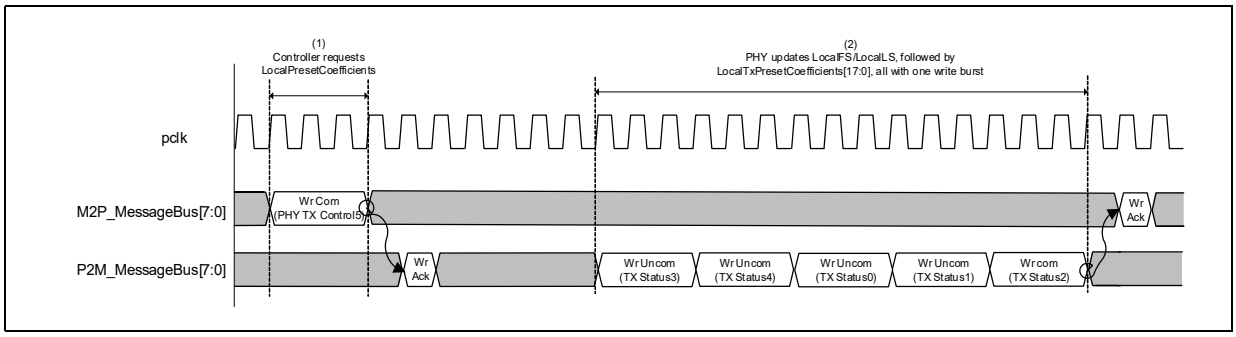


Figure 9-11 shows a sequence where LocalFS and LocalLF are updated in response to a GetLocalPresetCoefficients request where the LocalPresetIndex corresponds to an 8 GT/s rate. Note that the LocalFS and LocalLF values must be updated before or at the same cycle as the LocalTxPresetCoefficents are returned.

Figure 9-11. LocalFS/LocalLF Update Due to GetLocalPresetCoefficients

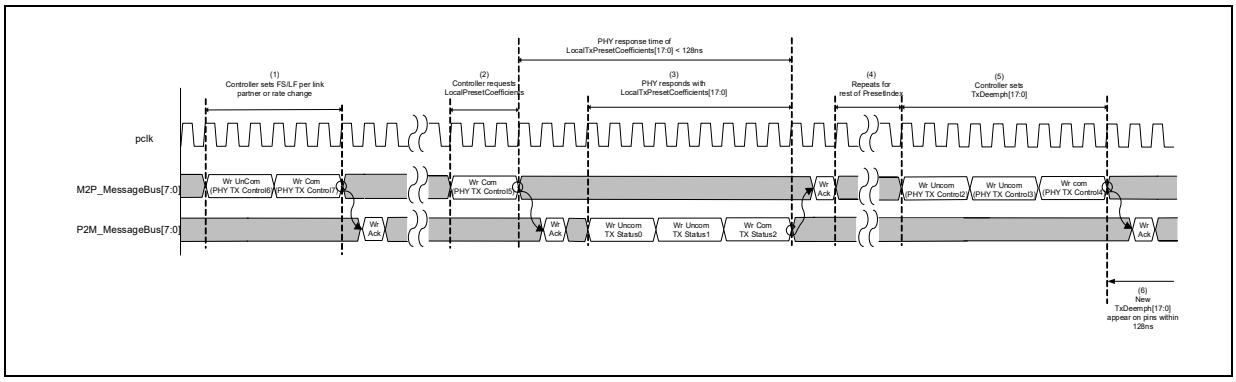


9.9 Message Bus: Updating TxDeemph

Figure 9-12 shows a sequence where the MAC makes a GetLocalPresetCoefficients request for one or more values of LocalPresetIndex and the, subsequently, update the TxDeemph value. Note that for every GetLocalPresetCoefficients request, there is a 128 ns maximum response time for the PHY to return the LocalTxPresetCoefficients value; this time is shown in the diagram from the end of the second write_committed to the end of the third write_committed. This maximum response time requirement only exists for designs that use just-in-time fetching of GetLocalPresetCoefficients in response to Tx coefficients request from the link partner; designs that fetch ahead of time can circumvent this requirement. Additionally, after the write_committed for TxDeemph, the new TxDeemph value must be reflected on the pins within 128 ns. Note

that while Figure 9-12does not show LocalLF and LocalFS getting returned in response to a GetLocalPresetCoefficients request, they can be returned along with LocalTxPresetCoefficients similar to what is done in Figure 9-11.

Figure 9-12. Updating TxDeemph after GetLocalPresetCoefficients Request



9.10 Message Bus: Equalization

Figure 9-13 shows a successful equalization sequence. RxEqInProgress is asserted for the entire duration of equalization. Multiple RxEqEval requests are made during the equalization process corresponding to different coefficient requests to the far end transmitter. When all the RxEqEval requests are complete, RxEqInProcess is deasserted. Note, the PHY does not necessarily have to write to both the LinkEvaluationFeedbackFigureMerit and LinkEvaluationFeedbackDirectionChange register fields; it could write to only to one.

Figure 9-13. Successful Equalization

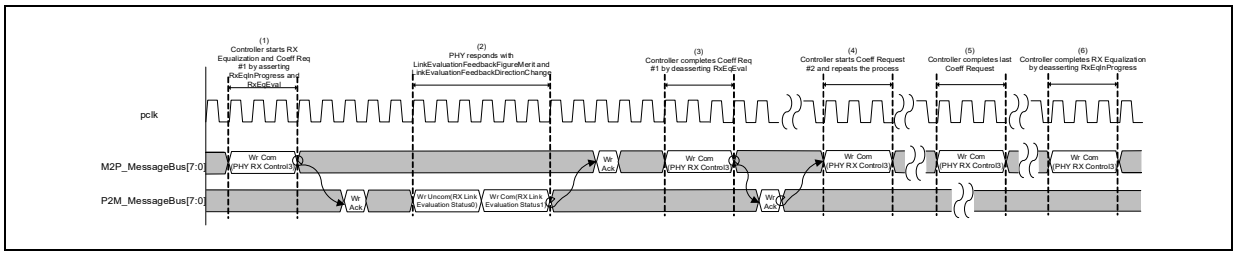


Figure 9-14 shows an equalization sequence where the feedback received indicates an invalid coefficient request for the link partner. Note that the write to assert InvalidRequest must happen before a new request is initiated; the write to deassert InvalidRequest can happen in the same cycle as an RxEqEval request for a new coefficient.

Figure 9-14. Equalization with Invalid Request

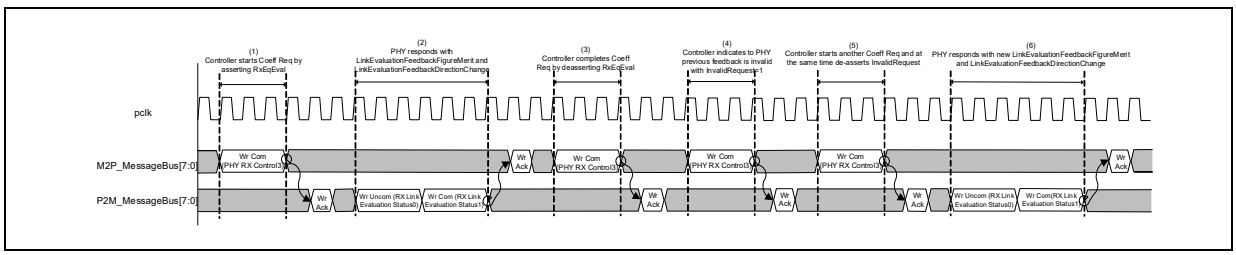


Figure 9-15 shows a sequence where the MAC aborts the RxEqEval request before the link evaluation feedback is returned by the PHY. Figure 9-16 shows a sequence where the MAC aborts the RxEqEval request while the link evaluation feedback is being returned by the PHY, that is, there is an overlap. In both abort case, the MAC must ignore the feedback value returned by the PHY.

Figure 9-15. Aborted Equalization, Scenario #1

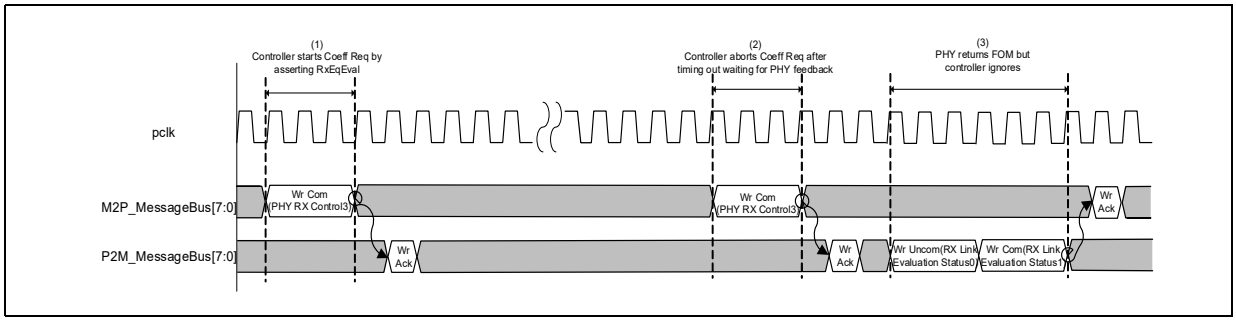
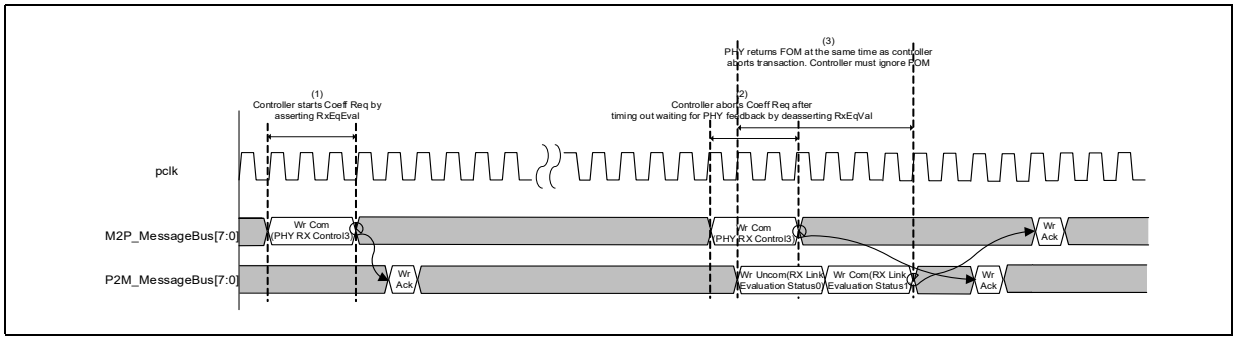


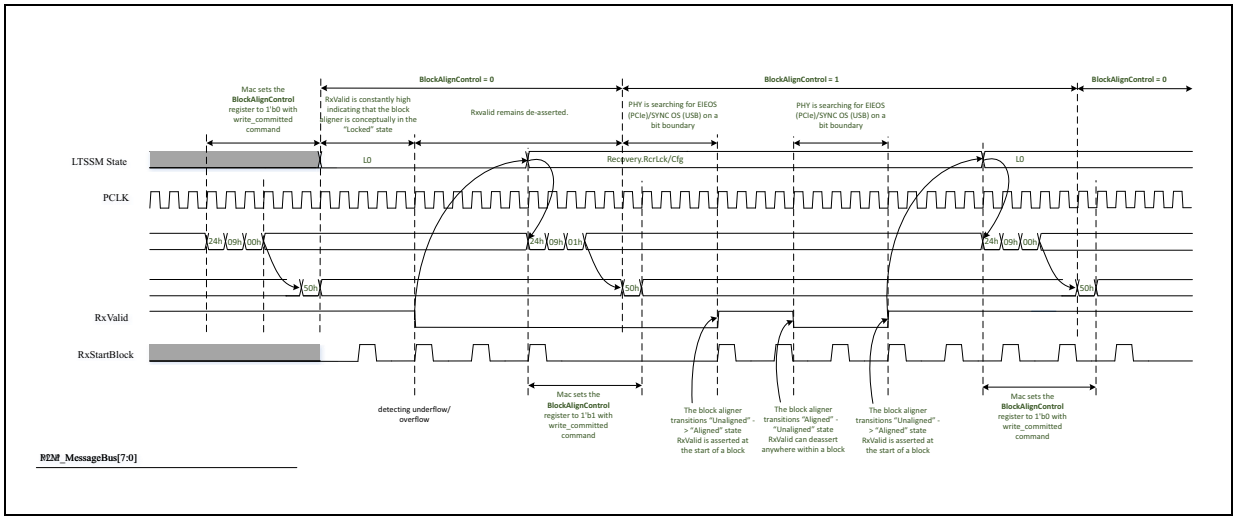
Figure 9-16. Aborted Equalization, Scenario #2



9.11 Message Bus: BlockAlignControl

Figure 9-17 shows a sequence where BlockAlignControl is used to reestablish block alignment after a loss of alignment is detected. This sequence also shows how RxValid transitions during this process.

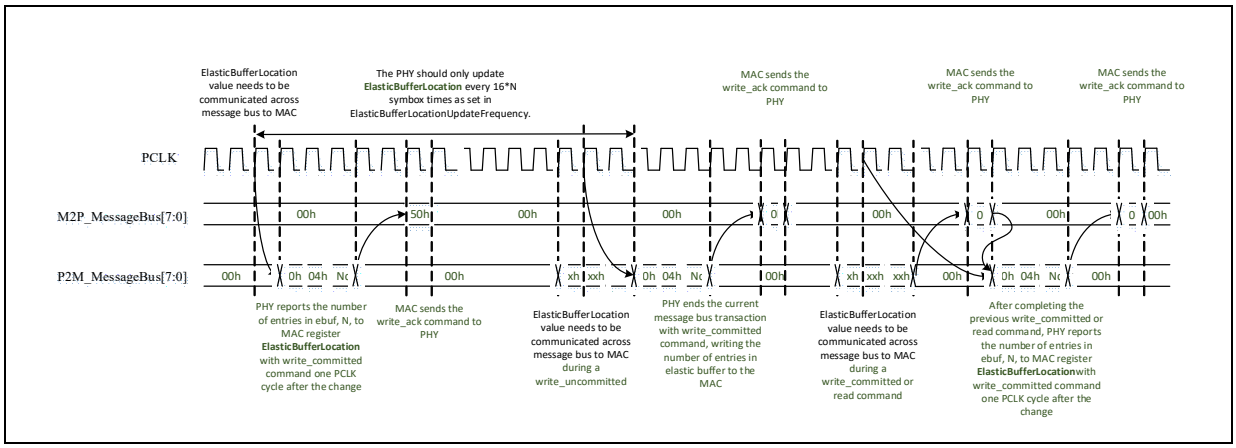
Figure 9-17. Message Bus: BlockAlignControl Example



9.12 Message Bus: ElasticBufferLocation Update

Figure 9-18 shows the update of ElasticBufferLocation across the message bus. The frequency of update across the message bus is controlled by the MAC by setting the value in the ElasticBufferLocationUpdateFrequency register.

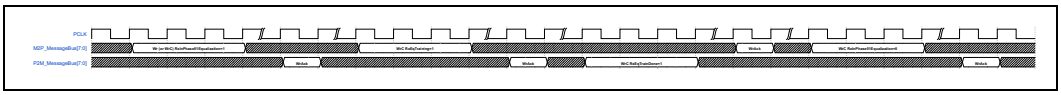
Figure 9-18. Message Bus: Updating ElasticBufferLocation



9.13 Message Bus: RxInPhase01Equalization Update

Figure 9-19 shows an example sequence where RxInPhase01Equalization is set prior to RxEqTraining being set. Both an uncommitted or a committed write are acceptable for first write in this sequence; the key requirement is that RxInPhase01Equalization is set before or at the same time as RxEqTraining.

Figure 9-19. Example Sequence: RxInPhase01Equalization and RxEqTraining Relationship

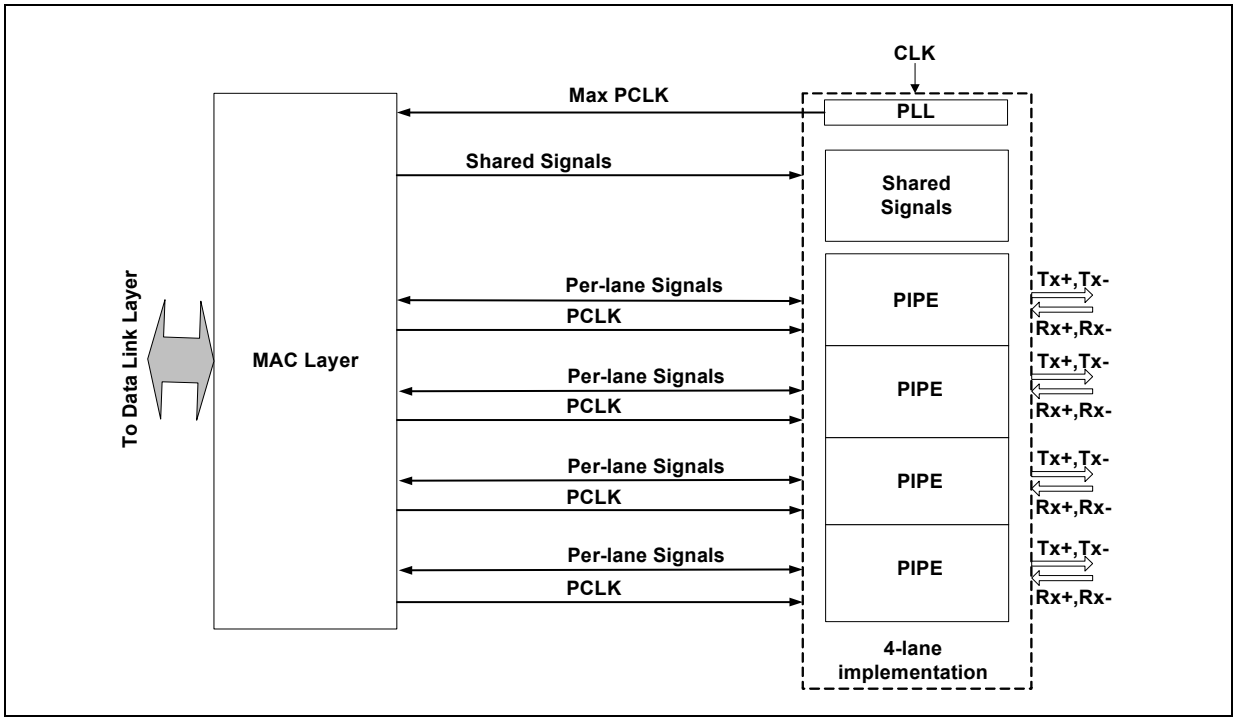


10 Multi-Lane PIPE – PCIe Mode

This section describes a suggested method for combining multiple PIPEs together to form a multi-lane implementation. It describes which PIPE signals can be shared between each PIPE of a multi-lane implementation, and which signals should be unique for each PIPE. There are two types of PHY. "Variable" PHYs that are designed to support multiple links of variable maximum widths and "Fixed" PHYs that are designed to support a fixed number of links with fixed maximum widths.

The figure shows an example four-lane implementation of a multilane PIPE solution with PCLK as a PHY input. The signals that can be shared are shown in the figure as "Shared Signals" while signals that must be replicated for each lane are shown as "Per-lane signals".

Figure 10-1. Four-Lane PIPE Implementation



The MAC layer is responsible for handling lane-to-lane deskew and it may be necessary to use the per-lane signaling of SKP insertion and removal to help perform this function.

Table 10-1. The MAC Layer

Shared Signals	Per-Lane Signals or Shared Signals	Per-lane Signals
CLK Max PCLK	EncodeDecodeBypass BlockAlignControl TxSwing TxMargin[2:0] TxDetectRx/Loopback Rate Width[1:0] PCLK Rate[2:0] Reset# TxDataValid PCLK Restore#	TxData[], TxDataK[] RxData[], RxDataK[] TxStartBlock TxElecIdle TxCompliance RxPolarity RxValid RxElecIdle RxStatus[2:0] RxDataValid RxStartBlock TxDeemph[17:0] PowerDown[1:0] PhyStatus RxPresetHint[2:0] RxEqEval LinkEvaluationFeedbackFigureMerit[7:0] LinkEvaluationFeedbackDirectionChange[7:0] InvalidRequest TxSyncHeader[1:0] RxSyncHeader[1:0] RxStandby RxStandbyStatus FS[5:0] LF[5:0] PHYMode[1:0] SRISEnable Elasticity Buffer Mode TxPattern[1:0] TxOnesZeros RxEqTraining LocalTxPresetCoefficients[17:0] LocalFS[5:0] LocalLF[5:0] LocalPresetIndex[5:0] GetLocalPresetCoefficients LocalTxCoefficientsValid RxEqInProgress RXTermination AlignDetect PowerPreset PclkChangeOk PclkChangeAck ElasticBufferLocation[N:0] AsyncPowerChangeAck M2P_MessageBus[7:0] P2M_MessageBus[7:0] RxCLK DeepPMReq# DeepPMAck#

A MAC must use all "Per-Lane Signals or Shared Signals" that are inputs to the PHY consistently on all lanes in the link. A PHY in "PCLK as PHY Output" mode must ensure that PCLK and Max PCLK are synchronized across all lanes in the link. A MAC must provide a synchronized PCLK as an input for each lane when controlling a PHY in "PCLK as PHY Input" mode with no more than 300 ps of skew on PCLK across all lanes.

It is recommended that a MAC be designed to support both PHYs that implement all signals per lane and those that implement the "Per-Lane or Shared Signals" per link. A "Variable" PHY must implement the signals in "Per-Lane Signals or Shared Signals" per lane. A "Fixed" PHY may implement the signals in "Per-Lane Signals or Shared Signals" as either Shared or Per-Lane. A "Fixed" PHY should implement all the signals in "Per-Lane Signals or Shared Signals" consistently as either Shared or Per-Lane.

Note: The following method to turn off a lane using TxElecIdle and TxCompliance is deprecated; PowerDown is used instead. Alternatively, the PHY must hold itself in its lowest power state when Reset# is asserted; the MAC is permitted to choose this mechanism instead of PowerDown to turn off lanes not in use.

In cases where a multi-lane has been "trained" to a state where not all lanes are in use (like a x4 implementation operating in x1 mode), a special signaling combination is defined to "turn off" the unused lanes allowing them to conserve as much power as the implementation allows. This special "turn off" signaling is done using the **TxElecIdle** and **TxCompliance** signals. When both are asserted, that PHY can immediately be considered "turned off" and can take whatever power saving measures are appropriate. The PHY ignores any other signaling from the MAC (except for **Reset#** assertion) while it is "turned off". Similarly, the MAC should ignore any signaling from the PHY when the PHY is "turned off ". There is no "handshake" back to the MAC to indicate that the PHY has reached a "turned off" state.

There are two normal cases when a lane can get turned off:

1. During LTSSM Detect state, the MAC discovers that there is no receiver present and will "turn off" the lane.
2. During LTSSM Configuration state (specifically Configuration.Complete), the MAC will "turn off" any lanes that did not become part of the configured link.

As an example, both cases could occur when a x4 device is plugged into a x8 slot. The upstream device (the one with the x8 port) will not discover receiver terminations on four of its lanes so it will turn them off. Training will occur on the remaining four lanes, and let's suppose that the x8 device cannot operate in x4 mode, so the link configuration process will end up settling on x1 operation for the link. Then both the upstream and downstream devices will "turn off" all but the one lane configured in the link.

When the MAC wants to get "turned off" lanes back into an operational state, there are two cases that need to be considered:

1. If the MAC wants to reset the multi-lane PIPE, it asserts **Reset#** and drives other interface signals to their proper states for reset (see Section 6.2). Note that this stops signaling "turned off" to all lanes because **TxCompliance** is deasserted during reset. The multi-lane PHY asserts **PhyStatus** in response to **Reset#** being asserted and will deassert **PhyStatus** when **PCLK** is stable.
2. When normal operation on the active lanes causes those lanes to transition to the LTSSM Detect state, then the MAC sets the **PowerDown[1:0]** signals to the P1 PHY power state at the same time that it deasserts "turned off" signaling to the inactive lanes. Then as with normal transitions to the P1 state, the multi-lane PHY will assert **PhyStatus** for one clock when all internal PHYs are in the P1 state and **PCLK** is stable.

A Appendix

A.1 DisplayPort AUX Signals

The set of DisplayPort AUX signals listed in Table A-1 should be implemented per connector.

Table A-1. DisplayPort AUX Signals

Name	Direction	Active Level	Description
TxAuxData	Input	N/A	DisplayPort asynchronous transmit data for AUX CH
TxAuxOE	Input	High	DisplayPort asynchronous data output enable for AUX CH. Assertion of this signal must be mutually exclusive with assertion of RxAuxIE.
RxAuxIE	Input	High	DisplayPort asynchronous data input enable for AUX CH. Assertion of this signal must be mutually exclusive with assertion of TxAuxOE.
RxAuxData	Output	N/A	DisplayPort asynchronous data output for AUX CH
DCAux+	Output	Low	Optional: DPRX asynchronous AUX+ pulldown status signal indicates a source is connected (DC voltage)
DCAux-	Output	High	Optional: DPRX asynchronous AUX-pullup status signal indicates a connected source is powered up (DC voltage)
AuxRxElecIdle	Output	High	Indicates whether differential signaling is detected