

**PHY Interface
for the
PCI Express* Architecture (PIPE)

For SATA 3.0**

Revision .7

Notice and 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.

ALL SUGGESTIONS OR FEEDBACK RELATED TO THIS SPECIFICATION BECOME THE PROPERTY OF INTEL CORPORATION UPON SUBMISSION.

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

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.

This document is an intermediate draft for comment only and is subject to change without notice. Readers should not design products based on this document.

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

Contributors

Dedicated to the memory of Brad Hosler, the impact of whose accomplishments made the Universal Serial Bus one of the most successful technology innovations of the Personal Computer era.

Table of Contents

1	Preface.....	6
1.1	Scope of this Revision	6
1.2	Revision History	6
2	Introduction	7
2.1	SATA PHY Layer.....	7
3	PHY/MAC Interface	7
4	SATA PHY Functionality	9
4.1	Transmitter Block Diagram (1.5, 3.0, and 6.0 GT/s).....	11
4.2	Receiver Block Diagram (1.5, 3.0 and 6.0 GT/s)	11
4.3	Clocking.....	12
5	PIPE Interface Signal Descriptions	12
5.1	PHY/MAC Interface Signals	12
5.2	External Signals	20
6	PIPE Operational Behavior	21
6.1	Clocking.....	21
6.2	Reset.....	21
6.3	Power Management	21
6.4	Changing Signaling Rate, PCLK Rate, or Data Bus Width.....	22
6.4.1	Fixed data path implementations	22
6.4.2	Fixed PCLK implementations	23
6.5	Clock Tolerance Compensation.....	23
6.6	Error Detection.....	24
6.6.1	8B/10B Decode Errors.....	24
6.6.2	Disparity Errors	25
6.6.3	Elastic Buffer Errors	25
6.7	Loopback	26
6.8	Implementation specific timing and selectable parameter support	27
6.9	Control Signal Decode table – SATA Mode	27
6.10	Required synchronous signal timings.....	27
7	Sample Operational Sequences	27
7.1	Receivers and Electrical Idle	27
7.2	TX OOB Signaling	27

Table of Figures

Figure 3-1: PHY/MAC Interface.	8
Figure 4-1: PHY Functional Block Diagram.....	10
Figure 4-2: Transmitter Block Diagram (1.5, 3.0, and 6.0 GT/s)	11
Figure 4-4: Receiver Block Diagram (1.5, 3.0 and 6.0 GT/s)	12
Figure 4-6: Clocking and Power Block Diagram	12

Table of Tables

Table 3-1. Possible PCLK rates and data widths.....	9
Table 5-1: Transmit Data Interface Signals.....	12
Table 5-2: Receive Data Interface Signals	13
Table 5-3: Command Interface Signals	14
Table 5-4: Status Interface Signals	18
Table 5-5: External Signals	20
Table 6-1 Minimum Elasticity Buffer Size	24

1 Preface

1.1 Scope of this Revision

Revision .7 of the SATA 3.0 PHY Interface Specification defines the intended architecture for updating the PCI Express PHY Interface Specification to support SATA 3.0. This revision includes support for SATA* implementations conforming to the SATA Specification, Revision 3.0.

1.2 Revision History

Revision Number	Date	Description

2 Introduction

The **PHY Interface for the PCI Express Architecture (PIPE)** for SATA 3.0 is intended to enable the development of functionally equivalent SATA PHY's. Such PHY's can be delivered as discrete IC's or as macrocells for inclusion in ASIC designs. The specification defines a set of PHY functions which must be incorporated in a PIPE compliant PHY, and it defines a standard interface between such a PHY and a Media Access Layer (MAC) & Link Layer ASIC. It is not the intent of this specification to define the internal architecture or design of a compliant PHY chip or macrocell. The PIPE specification is defined to allow various approaches to be used. Where possible the PIPE specification references the SATA specification rather than repeating its content. In case of conflicts, the SATA specification shall supersede the PIPE spec.

This spec provides some information about how the MAC could use the PIPE interface for various SATA protocols. This information should be viewed as guidelines for or 'one way to implement' SATA 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 SATA device development. This document defines an interface to which ASIC and device vendors can develop. 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 PCI Express PHY interface, thus minimizing the time and risk of their development cycles.

2.1 SATA PHY Layer

The SATA PHY Layer handles the low level SATA protocol and signaling. This includes features such as; data serialization and deserialization, 8b/10b encoding, analog buffers, 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:

- Standard PHY interface enables multiple IP sources for SATA controllers and provides a target interface for SATA PHY vendors.
- Supports 1.5 GT/s only or 1.5 GT/s and 3.0 GT/s, or 1.5 GT/s, 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 by 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

3 PHY/MAC Interface

Figure 3-1 shows the data and logical command/status signals between the PHY and the MAC layer. These signals are described in Section 5. Full support of SATA at all rates requires 19 control signals and 6 Status signals.

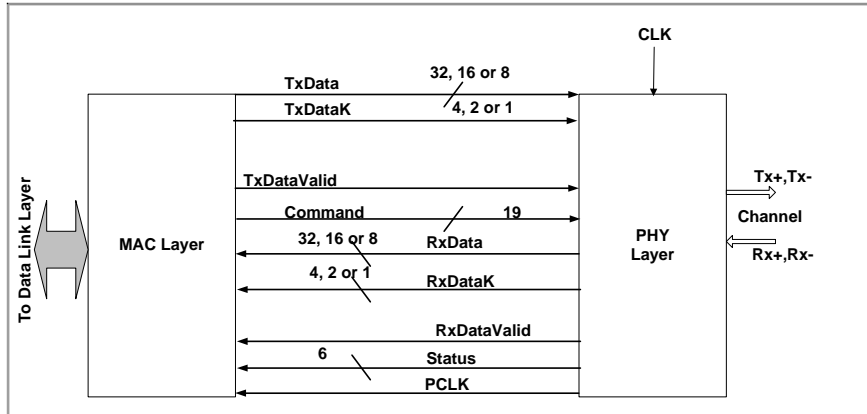


Figure 3-1: PHY/MAC Interface.

This specification allows several different PHY/MAC interface configurations to support various signaling rates. For 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 requires the use of DataValid signals to toggle the use of data on the data bus. 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 GT/s signaling. The full set of possible widths and PCLK rates is shown in Table 3-1. A PIPE compliant MAC or PHY is only required to support one option for each SATA transfer speed that it supports.

Mode	PCLK	Data Width
1.5 GT/s SATA	600 Mhz	8 bits* DataValid is asserted every fourth PCLK to indicate valid data.
1.5 GT/s SATA	300 Mhz	8 bits* DataValid is asserted every PCLK to indicate valid data.
1.5 GT/s SATA	150 Mhz	8 bits
1.5 GT/s SATA	75 Mhz	16 bits
1.5 GT/s SATA	37.5 Mhz	32 bits
3.0 GT/s SATA	300 Mhz	8 bits
3.0 GT/s SATA	150 Mhz	16 bits

3.0 GT/s SATA	75 Mhz	32 bits
3.0 GT/s SATA	600 Mhz	8 bits* DataValid is toggled every PCLK to indicate valid data.
6.0 GT/s SATA	600 Mhz	8 bits
6.0 GT/s SATA	300 Mhz	16 bits
6.0 GT/s SATA	150 Mhz	32 bits

Table 3-1. Possible PCLK rates and data widths

Note: If the PHY elasticity buffer is operating in nominal empty mode – then DataValid may also be used when the EB is empty and no data is available.

There may be PIPE implementations that support multiples of the above configurations. PHY implementations that support multiple configurations at the same rate must support the width and PCLK rate control signals.

4 SATA PHY Functionality

Figure 4-1 shows the functional block diagram of the 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.

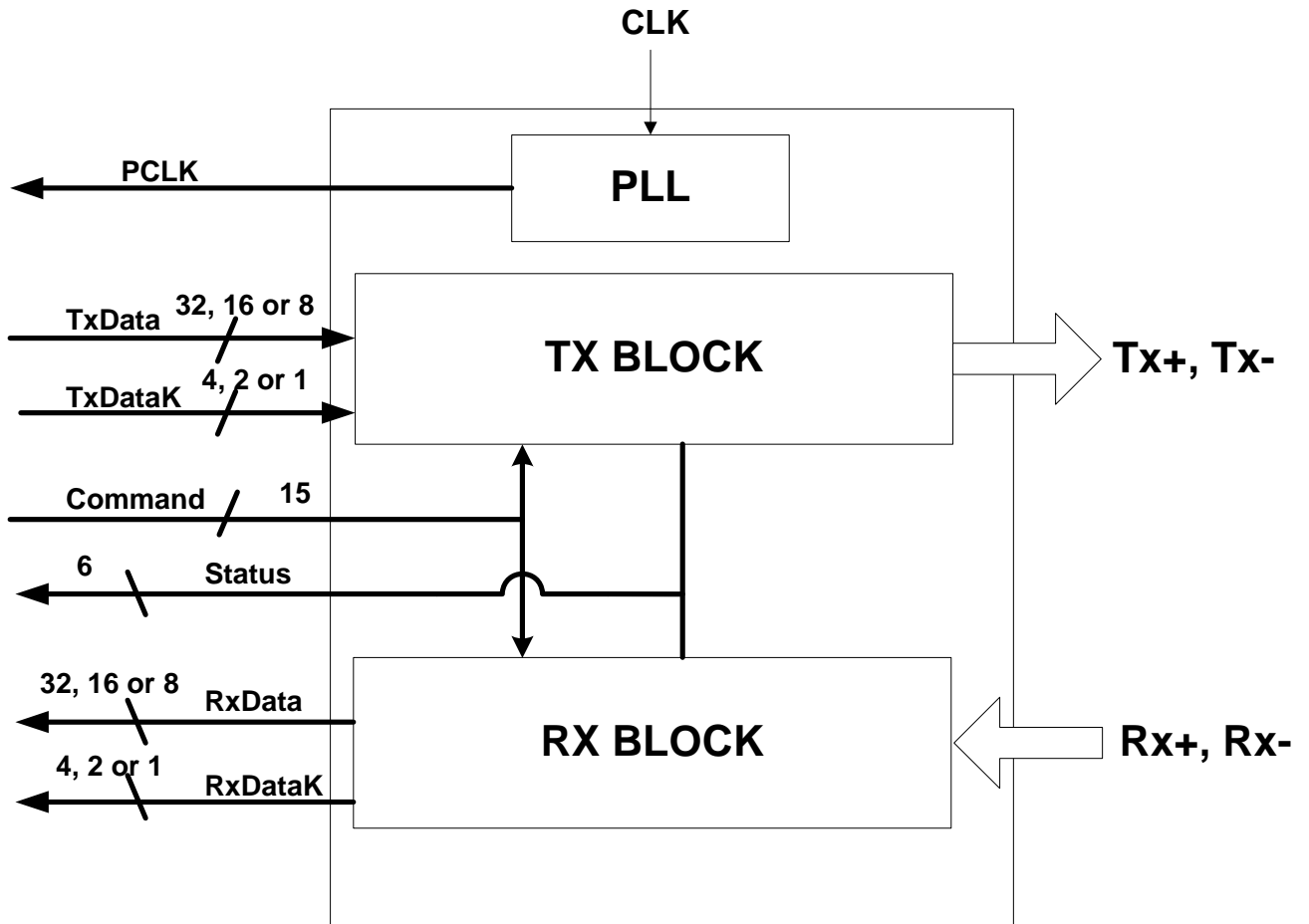


Figure 4-1: PHY Functional Block Diagram

Sections below provide descriptions of each of the blocks shown in Figure 4-1: PHY Functional Block Diagram. 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 Transmitter Block Diagram (1.5, 3.0, and 6.0 GT/s)

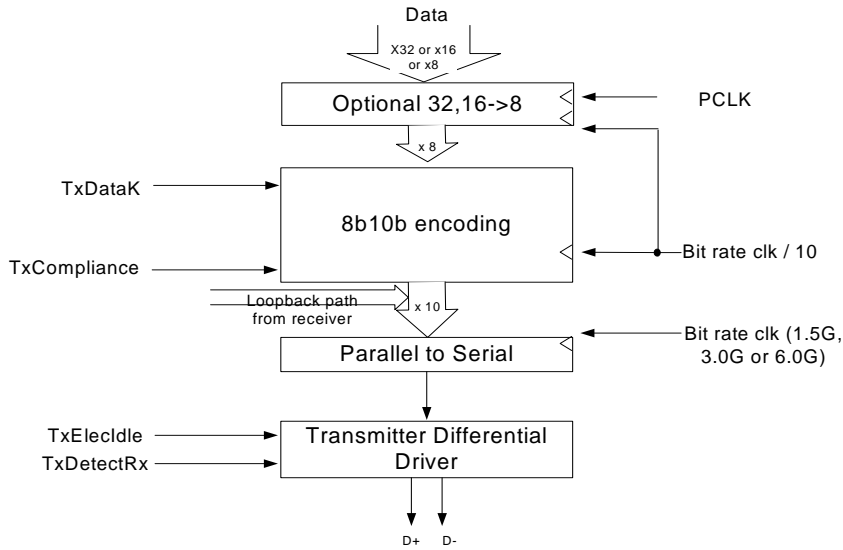


Figure 4-2: Transmitter Block Diagram (1.5, 3.0, and 6.0 GT/s)

4.2 Receiver Block Diagram (1.5, 3.0 and 6.0 GT/s)

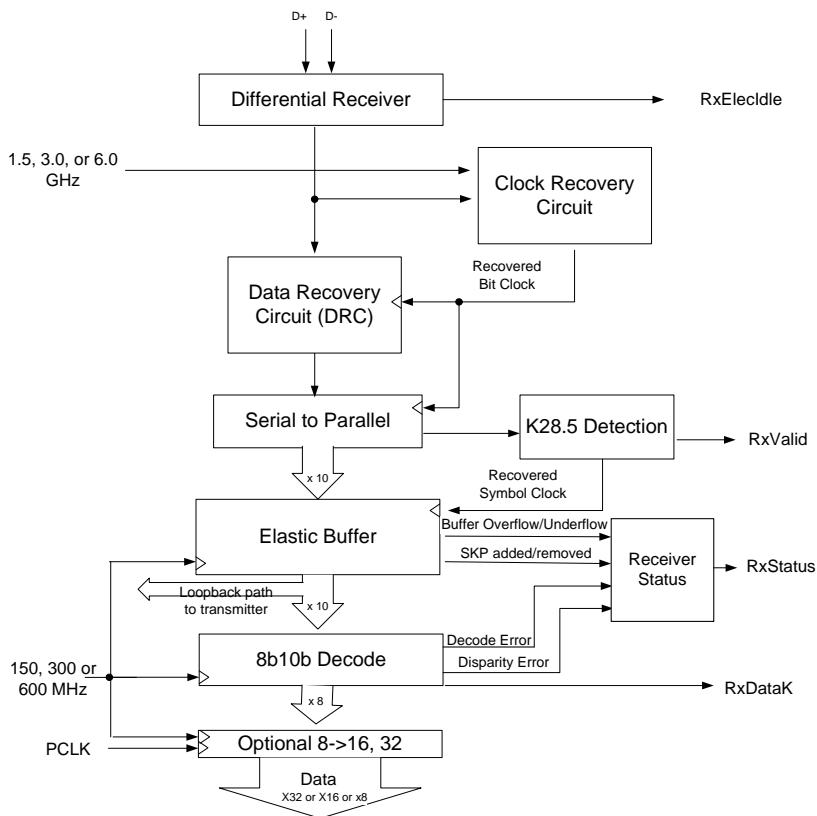


Figure 4-3: Receiver Block Diagram (1.5, 3.0 and 6.0 GT/s)

4.3 Clocking

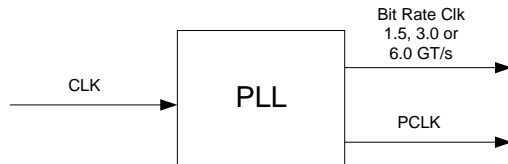


Figure 4-4: Clocking and Power Block Diagram

5 PIPE Interface Signal Descriptions

5.1 PHY/MAC Interface Signals

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. Thus 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 tables below.

Table 5-1: Transmit Data Interface Signals

Name	Direction	Active Level	Description
Tx+, Tx-	Output	N/A	The SATA differential outputs from the PHY.
TxData[31:0] for 32-bit interface TxData[15:0] for 16-bit interface TxData[7:0] for 8-bit interface	Input	N/A	Parallel SATA Express data input bus. For the 16-bit interface, 16 bits represent 2 symbols of transmit 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 transmit data. Bits [23:16] are the third symbol to be transmitted, and bits [31:24] are the fourth symbol.

TxDataK[3:0] for 32-bit interface TxDataK[1:0] for 16-bit interface TxDataK for 8-bit interface	Input	N/A	Data/Control for the symbols of transmit data. For 32-bit interfaces, Bit 0 corresponds to the low-byte of TxData, Bit3 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. Data bytes are scrambled and control bytes are not.
TxDataValid	Input	N/A	This signal allow the MAC to instruct the PHY to ignore the data interface for one clock cycle. A value of one indicates the phy will use the data, a value of zero indicates the phy will not use the data.

Table 5-2: Receive Data Interface Signals

Name	Direction	Active Level	Description
Rx+, Rx-	Input	N/A	The SATA differential inputs to the PHY.
RxData[31:0] for 32-bit interface RxData[15:0] for 16-bit interface or RxData[7:0] for 8-bit interface	Output	N/A	Parallel SATA data output bus. For 16-bit interface, 16 bits represents 2 symbols of receive 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 4 symbols of receive data. Bits [23:16] are the third symbol received, and bits [31:24] are the fourth symbol received.
RxDataK[3:0] for 32-bit interface RxDataK[1:0] for 16-bit interface RxDataK for 8-bit interface	Output	N/A	Data/Control bit for the symbols of receive data. For 32-bit interfaces, Bit 0 corresponds to the low-byte of RxData, Bit3 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. When the PHY is in a SATA mode, the first valid data following an ALIGN primitive must appear as byte 0 in the receive data.
RxDataValid	Output	N/A	This signal allows the PHY to instruct the MAC to ignore the data interface for one clock cycle. A value of one indicates the MAC will use the data, a value of zero indicates the MAC will not use the data.

Table 5-3: Command Interface Signals

Name	Direction	Active Level	Description
TxDetectRx/Loopback	Input	High	<p>Used to tell the PHY to begin loopback. Refer to Section Error! Reference source not found. for details on the required values for all control signals to perform loopback.</p> <p>Loopback support is optional for SATA PHYs. Loopback is only valid 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.</p>
TxElectIdle	Input	High	<p>Forces Tx output to electrical idle when asserted in all power states.</p> <ul style="list-style-type: none"> When deasserted while in P0 (as indicated by the <i>PowerDown</i> signals), indicates that there is valid data present on the <i>TxData[.]</i> and <i>TxDataK[.]</i> pins and that the data must be transmitted. <p>• See section 6.3 for the definitions of PHY power states.</p>
TX Pattern[1:0]	Input	N/A	<p>Controls which pattern the PHY sends at the Gen 1 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.</p> <p>0 ALIGN 1 D24.3 2 D10.2 3 Reserved</p>
EBBufferMode	Input	N/A	<p>Controls the operating mode of the elasticity buffer</p> <p>0 Elasticity buffer operates in nominal half-full mode. 1 Elasticity buffer operates in nominal empty mode.</p> <p>This signal is only supported by a PHY that supports more than one elasticity buffer mode.</p>

Reset#	Input	Low	<p>Resets the transmitter and receiver. This signal is asynchronous.</p> <p>The PHY reports its default power state after reset as defined in section 6.8.</p>																																																			
PowerDown[2:0]	Input	N/A	<p>Power up or down the transceiver. Power states</p> <table border="1"> <thead> <tr> <th>[2]</th> <th>[1]</th> <th>[0]</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> <td>POWER_STATE_0 Operational state.</td> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> <td>POWER_STATE_1 Phy specific.</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> <td>POWER_STATE_2 Phy specific.</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> <td>POWER_STATE_3 Phy specific.</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> <td>POWER_STATE_4 Phy specific.</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> <td>POWER_STATE_5 Phy specific.</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> <td>POWER_STATE_6 Phy specific.</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> <td>POWER_STATE_7 Phy specific.</td> </tr> </tbody> </table> <p>A PIPE compliant SATA PHY is recommended to support at least 4 states other than POWER_STATE_0. There must be at least one additional power state meeting the requirements shown in the following table</p> <table border="1"> <thead> <tr> <th>PCLK State</th> <th>TX Common Mode State</th> <th>Exit Latency to POWER_STATE_0</th> </tr> </thead> <tbody> <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> </tbody> </table> <p>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.</p> <p>Note: PLL shutdown is only possible if PowerDown is set to a state with PCLK off.</p> <p>An informative table with value of all signals after reset will be added in the next revision.</p>	[2]	[1]	[0]	Description	0	0	0	POWER_STATE_0 Operational state.	0	0	1	POWER_STATE_1 Phy specific.	0	1	0	POWER_STATE_2 Phy specific.	0	1	1	POWER_STATE_3 Phy specific.	1	0	0	POWER_STATE_4 Phy specific.	1	0	1	POWER_STATE_5 Phy specific.	1	1	0	POWER_STATE_6 Phy specific.	1	1	1	POWER_STATE_7 Phy specific.	PCLK State	TX Common Mode State	Exit Latency to POWER_STATE_0	Off	Off	< 10 ms	Off	On	< 10 us	On	On	< 10 us	On	Off	< 300 us
[2]	[1]	[0]	Description																																																			
0	0	0	POWER_STATE_0 Operational state.																																																			
0	0	1	POWER_STATE_1 Phy specific.																																																			
0	1	0	POWER_STATE_2 Phy specific.																																																			
0	1	1	POWER_STATE_3 Phy specific.																																																			
1	0	0	POWER_STATE_4 Phy specific.																																																			
1	0	1	POWER_STATE_5 Phy specific.																																																			
1	1	0	POWER_STATE_6 Phy specific.																																																			
1	1	1	POWER_STATE_7 Phy specific.																																																			
PCLK State	TX Common Mode State	Exit Latency to POWER_STATE_0																																																				
Off	Off	< 10 ms																																																				
Off	On	< 10 us																																																				
On	On	< 10 us																																																				
On	Off	< 300 us																																																				

PHY Mode[2:0]	Input	N/A	Selects PHY operating mode.	
			Value	Description
			0	PCI Express
			1	USB SuperSpeed
			2	SATA
			3	Reserved
			4	Reserved
			5	Reserved
			6	Reserved
7	Reserved			
			Implementation of this signal is not required for PHYs that only support a single mode.	
Rate[1:0]	Input	N/A	Control the link signaling rate. 0 Use 1.5 GT/s signaling rate 1 Use 3.0GT/s signaling rate 2 Use 6.0 GT/s signaling rate 3 Reserved PIPE implementations that only support one signaling rate do not implement this signal.	
Width[1:0]	Input	N/A	Control the PIPE data path width 0 8 bits 1 16 bits 2 32 bits 3 Reserved PIPE implementations that only support one option at each signaling rate do not implement this signal.	
PCLK Rate[2:0]	Input	N/A	Control the PIPE PCLK rate 0 37.5 Mhz 1 75 Mhz 2 150 Mhz 3 300 Mhz 4 600 Mhz 5 Reserved 6 Reserved 7 Reserved PIPE implementations that only support one option at each signaling rate do not implement this signal.	
RxStandby	Input	N/A	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.	

RxStandbyStatus	Output	N/A	<p>The PHY uses this signal to indicate its RxStandby state. 0 – Active 1 – Standby</p> <p>RxStandbyStatus reflects the state of the high speed receiver. The high speed receiver is always off in PHY states that do not provide PCLK. The PHY indicates in section 6.8 any other power states in which the high speed receiver is always off..</p>
EncodeDecodeBy pass	Input	N/A	<p>Controls whether the PHY performs 8b/10b encode and decode. 0 – 8b/10b encode/decode performed normally by the PHY. 1 – 8b/10b encode/decode bypassed.</p> <p>The MAC can only change this signal during reset or in a power state other than POWER_STATE_0.</p> <p>When EncodeDecodeBypass is one the TxDataK and RxDataK interfaces are not used and the data bus width is 10, 20, or 40 bits.</p>

Table 5-4: Status Interface Signals

Name	Direction	Active Level	Description
RxValid	Output	High	Indicates symbol lock on <i>RxData</i> and <i>RxDataK</i> .
PhyStatus	Output	High	Used to communicate completion of several PHY functions including power management state transitions and rate change. 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.
AlignDetect	Output	High	Indicates receiver detection of an Align. A PHY is only required to assert this signal when the Elasticity Buffer is running in nominal empty mode.
RxEleIdle	Output	High	Indicates receiver detection of an electricalidle. This is an asynchronous signal. The time the signal is asserted must match the actual idle time on the analog bus within -16/+0 ns.

RxStatus[2:0]	Output	N/A	Encodes receiver status and error codes for the received data stream when receiving data.			
			[2]	[1]	[0]	Description
			0	0	0	Received data OK
			0	0	1	1 ALIGN added Asserted with 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	1 ALIGN 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	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 decode error Note: 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 Note: This error is never reported if EncodeDecodeBypass is asserted.			

5.2 External Signals

Table 5-5: External Signals

Name	Direction	Active Level	Description								
CLK	Input	Edge	This differential Input is used to generate the bit-rate clock for the PHY transmitter and receiver. Specs for this clock signal (frequency, jitter, ...) are implementation dependent and must be specified for each implementation. This clock may have a spread spectrum modulation.								
PCLK	Output	Rising Edge	Parallel interface data clock. All data movement across the parallel interface is synchronized to this clock. This clock operates at 37.5, 75, 150MHz, 300MHz, or 600 MHz depending on the <i>Rate</i> control input and the data interface width. The rising edge of the clock is the reference for all signals. Spread spectrum modulation on this clock is allowed.								
Max PCLK	Output	Rising Edge	Parallel interface data clock. This fixed rate clock operates at the the following rate: <table style="margin-left: 20px; border: none;"> <tr> <td>Max rate supported</td> <td>Max PCLK</td> </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. Spread spectrum modulation on this clock is allowed.	Max rate supported	Max PCLK	1.5 GT/s	150 MHz.	3.0 GT/s	300 MHz.	6.0 GT/s	600 MHz.
Max rate supported	Max PCLK										
1.5 GT/s	150 MHz.										
3.0 GT/s	300 MHz.										
6.0 GT/s	600 MHz.										

6 PIPE Operational Behavior

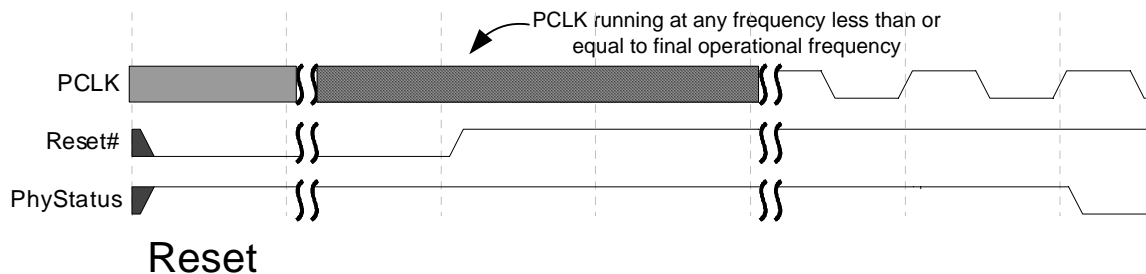
6.1 Clocking

There are three clock signals used by the PHY Interface component. The first (*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 operating modes of the PHY. This clock may have spread spectrum modulation.

The second clock (*PCLK*) is an output from the PHY 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. The rising edge of this clock is the reference point. This clock may also have spread spectrum modulation. The third clock (*MAX PCLK*) is a constant frequency clock with a frequency determined by the maximum SATA signaling rate supported by the PHY.

6.2 Reset

When the MAC wants to reset the PHY (e.g.; initial power on), the MAC must hold the PHY in reset until power and *CLK* to the PHY are stable. The PHY signals that *PCLK* is valid (ie. *PCLK* has been running at its operational frequency for at least one clock) and the PHY is in the specified power state by the de-assertion of *PhyStatus* after the MAC has stopped holding the PHY in reset. While *Reset#* is asserted the MAC should have *TxDetectRx/Loopback* deasserted, *TxElecIdle* asserted, *PowerDown* set to the default value reported by the PHY, , and *Rate*, *Width*, and *PCLK Rate* set to any option supported by the PHY.



6.3 Power Management

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 Table 5-3. . *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 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 assertion of *PhyStatus*. Transitions into and out of power states that do not provide `PCLK` are described below. 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 link states in the SATA specification. When transitioning into a power state that does not provide `PCLK`, the PHY must assert *PhyStatus* before `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 `PCLK` on for at least one cycle after asserting *PhyStatus*. When transitioning out of a state that does not provide `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 are allowed by PIPE. However, a MAC must ensure that SATA specification timing requirements are met.

6.4 Changing Signaling Rate, `PCLK` Rate, or Data Bus Width

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` and *TxElecIdle* and *RxStandby* are asserted, or in a low power state where `PCLK` is provided. The rate can be changed, or the width can be changed, or the `PCLK` rate can be changed, or any combination of the rate and width and `PCLK` rate, can be changed simultaneously. When the MAC changes the *Rate* signal, and/or the *Width* signal, and/or the *PCLK rate* signal, 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 and/or width changes until the PHY has indicated that the change has completed.

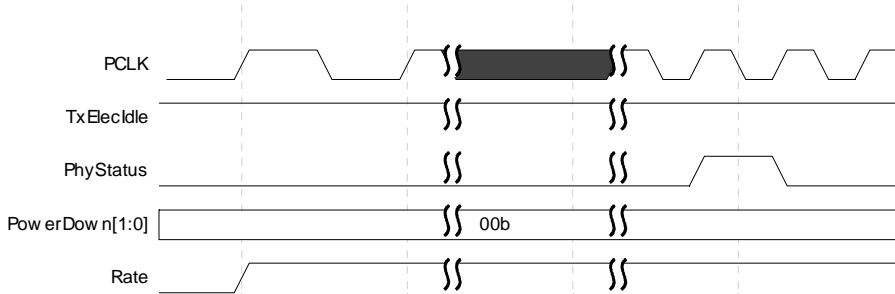
There are instances where conditions indicate both a speed change and/or width and/or `PCLK` rate change and a power state change for the PHY. In such cases the MAC must change the signaling rate and/or width and/or `PCLK` rate, before changing the power state.

Some PHY architectures may allow a speed change and a power state change to occur at the same time as a rate and/or width and/or `PCLK` rate change. If a PHY supports this, the MAC must change the rate and/or width and/or `PCLK` 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 and/or width and/or `PCLK` rate change is complete.

6.4.1 Fixed data path implementations

The figure below shows logical timings for implementations that change `PCLK` frequency when the MAC changes the signaling rate. 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.

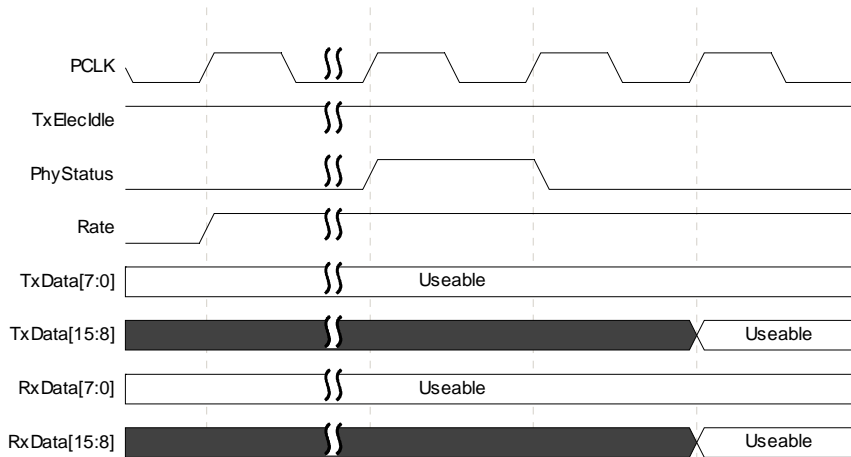
Also during the clock transition period, the frequency of *PCLK* must not exceed the PHY's defined maximum PCLK frequency. The amount of time between when *Rate* is changed and the PHY completes the rate change is a PHY specific value. Implementations that change the PCLK rate and data path width using the Fixed data path implementations timings.



Rate change with fixed data path

6.4.2 Fixed PCLK implementations

The figure below shows logical timings for implementations that change the width of the data path for different signaling rates. PCLK may be stopped during a rate change. Implementations that do not change the width or the PCLK rate for a rate change (using options that use TxDataValid and RxDataValid) use the fixed PCLK timings.



Rate change with fixed PCLK frequency

6.5 Clock Tolerance Compensation

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 ALIGNs as shown in Table 6-1

Phy Mode	Worst Case Frequency Offset	Symbol Depth – Nominal Half Full
----------	-----------------------------	----------------------------------

		Buffer
SATA	5600 ppm	15 symbols

Table 6-1 Minimum Elasticity Buffer Size

Two models are defined for the elastic buffer operation in the PHY. The PHY may support one or both of these models.

For the Nominal Half Full buffer model, the PHY is responsible for inserting or removing ALIGNs in the received data stream to avoid elastic buffer overflow or underflow. The PHY monitors the receive data stream, and when an ALIGN is received, the PHY can add or remove one ALIGN symbol as appropriate to manage its elastic buffer to keep the buffer as close to half full as possible. Whenever a 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* shall be asserted during the clock cycle when the first symb of the ALIGN is moved across the parallel interface.

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 requires the MAC to support the RxDataValid signal.

6.6 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 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 interface), the errors should be signaled with the priority shown below.

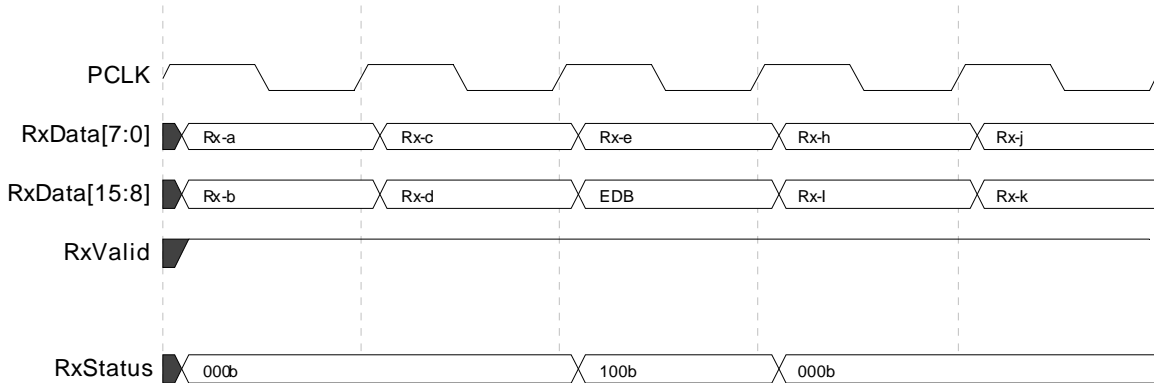
1. 8B/10B decode error
2. Elastic buffer overflow
3. Elastic buffer underflow (Can not occur in Nominal Empty buffer model)
4. Disparity error
5. Misalign

If an error occurs during a SKP ordered-set, such that the error signaling and SKP added/removed signaling on *RxStatus* would occur on the same CLK, then the error signaling has precedence.

6.6.1 8B/10B Decode Errors

For a detected 8B/10B decode error, the PHY should place a symbol in the data stream in place of the bad byte, and encode *RxStatus* with a decode error during the clock cycle when the effected byte is transferred across the parallel interface. In the example below, the receiver is receiving a stream of bytes Rx-a through Rx-z, and byte Rx-f has an 8B/10B decode error. In

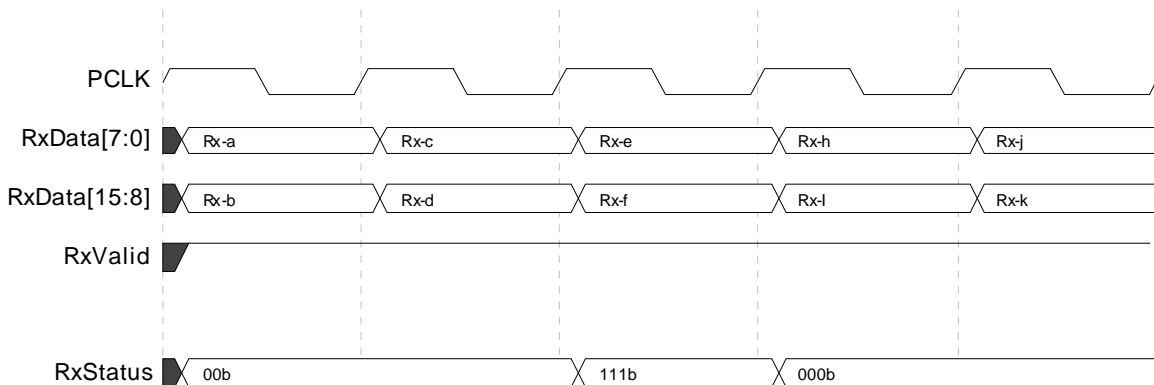
place of that byte, the PHY places an EDB on the parallel interface, and sets *RxStatus* to the 8B/10B decode error code. Note that a byte that can't be decoded may also have bad disparity, but the 8B/10B error has precedence. Also note that for a 16-bit interface, if the bad byte is on the lower byte lane, the byte on the higher byte lane may have bad disparity, but again, the 8B/10B error has precedence.



8B/10B Decode Error

6.6.2 Disparity Errors

For a detected disparity error, the PHY should assert *RxStatus* with the disparity error code during the clock cycle when the effected byte is transferred across the parallel interface. For 16-bit interfaces, it is not possible to discern which byte (or possibly both) had the disparity error. In the example below, the receiver detected a disparity error on either (or both) Rx-e or Rx-f data bytes, and indicates this with the assertion of *RxStatus*. (MACs often treat 8B/10B errors and disparity errors identically.)

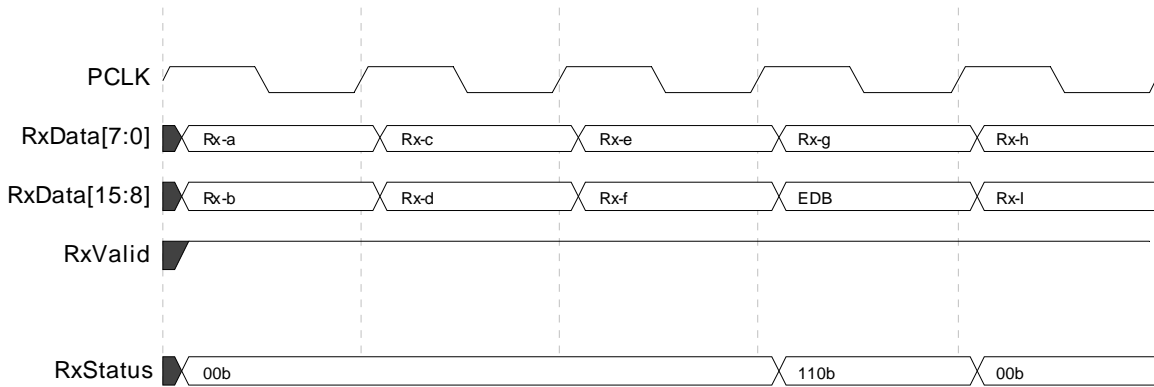


Disparity Error

6.6.3 Elastic Buffer Errors

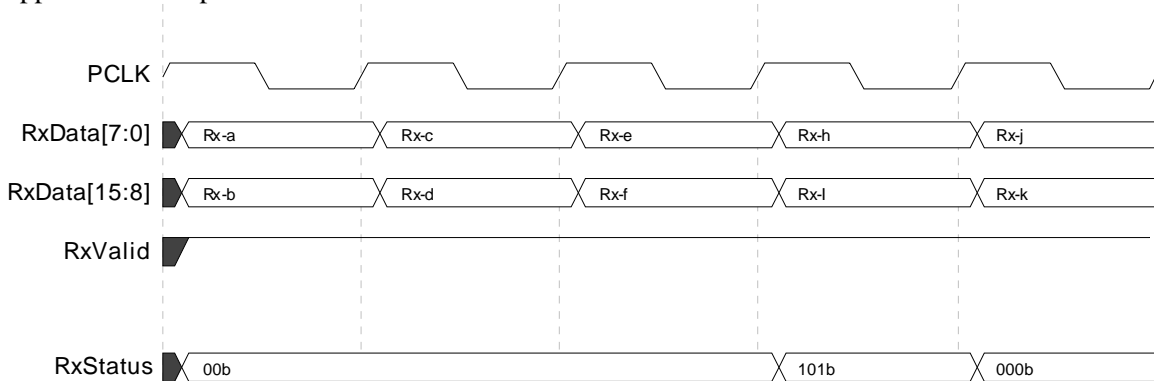
For elastic buffer errors, an underflow should be signaled during the clock cycle or clock cycles when a spurious symbol is moved across the parallel interface. The symbol moved across the interface should be the EDB symbol. In the timing diagram below, the PHY is receiving a repeating set of symbols Rx-a thru Rx-z. The elastic buffer underflows causing the EDB symbol 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 is presented on the parallel interface.

Note that underflow is not signaled when the PHY is operating in Nominal Empty buffer mode.



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 timing diagram below, 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.



Elastic Buffer Overflow

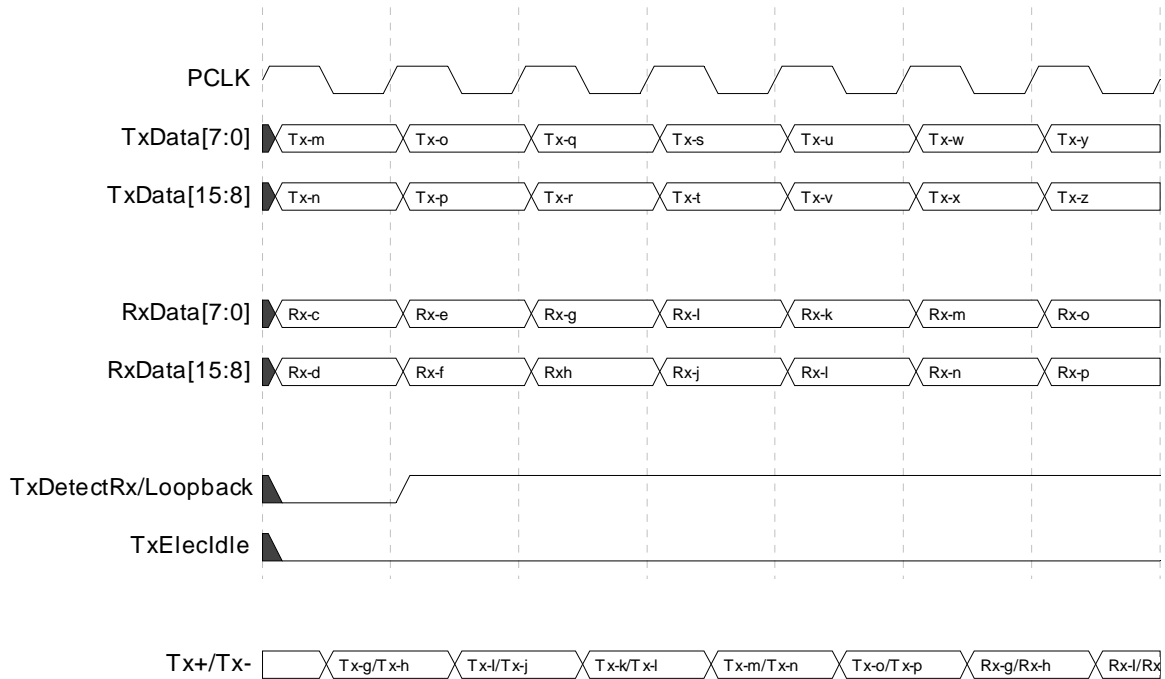
6.7 Loopback

- The PHY must support an internal loopback mode when *EncodeDecodeBypass* is asserted.

The PHY begins to loopback data when the MAC asserts *TxDetectRx/Loopback* while doing normal data transmission (ie. 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 timing diagram below shows example timing for 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.



Loopback start

6.8 Implementation specific timing and selectable parameter support

PHY vendors (macrocell or discrete) must specify typical and worst case timings for the cases listed in the table below.

Transmit Latency	Time for data moving between the parallel interface and the SATA serial lines. Timing is measured from when the data is transferred across the parallel interface (ie. the rising edge of <i>PCLK</i>) and when the first bit of the equivalent 10-bit symbol is transmitted on the <i>Tx+/Tx-</i> serial lines. Note: If the transmit latency is different when <i>EncodeDecodeBypass</i> is asserted – the PHY must report this latency separately.
Receive Latency	Time for data moving between the parallel interface and the SATA serial lines. Timing is measured from when the first bit of a 10-bit symbol is available on the <i>Rx+/Rx-</i> serial lines to when the corresponding 8-bit data is transferred across the parallel interface (i.e. the rising edge of <i>PCLK</i>). Note: If the receive latency is different when <i>EncodeDecodeBypass</i> is

	asserted – the PHY must report this latency separately.
Power State After Reset	The PHY power state immediately following reset. The state after reset needs to provide PCLK and have common mode off.
Loopback enable latency	Amount of time it takes the PHY to begin looping back receive data. Timed from when <i>TxDetectRx/Loopback</i> is asserted until the receive data is being transmitted on the serial pins.
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.
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 <i>PowerDown</i> signals to <i>POWER_STATE_X</i> until the PHY asserts <i>PhyStatus</i> . PHY asserts <i>PhyStatus</i> when it is ready to begin data transmission and reception. The PHY reports this transition between each pair of power states 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 <i>PowerDown</i> signals to the new power state until the PHY deasserts <i>PhyStatus</i> . 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.
Reset to ready time	Timed from when <i>Reset#</i> is deasserted until the PHY deasserts <i>PhyStatus</i> .
Supported power states.	The PHY supports each power state it supports. For each power state supported it reports whether PCLK is provided, the exit latency, and the common mode state.
Simultaneous Rate and Power State Change	The PHY reports if it supports simultaneous rate and power state changes.
Data Rate change time.	Amount of time the PHY takes to perform a data rate change. Time is measured from when the MAC changes <i>Rate</i> to when the PHY signals rate change complete with the single clock assertion of <i>PhyStatus</i> . There may be separate values for each possible change between 1.5, 3.0, and 6.0 GT/s..

6.9 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 6.8 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.

<i>PowerDown</i> [2:0]	<i>TxDetectRx/Loopback</i>	<i>TxElecIdle</i>	Description
<i>POWER_STATE_0_00b</i>	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 <i>TxElecIdle</i> 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.
			PHY is not transmitting and is in electrical idle.

6.10 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	0ns
PCLK to data valid for outputs	No greater than 25% of cycle time

7 Sample Operational Sequences

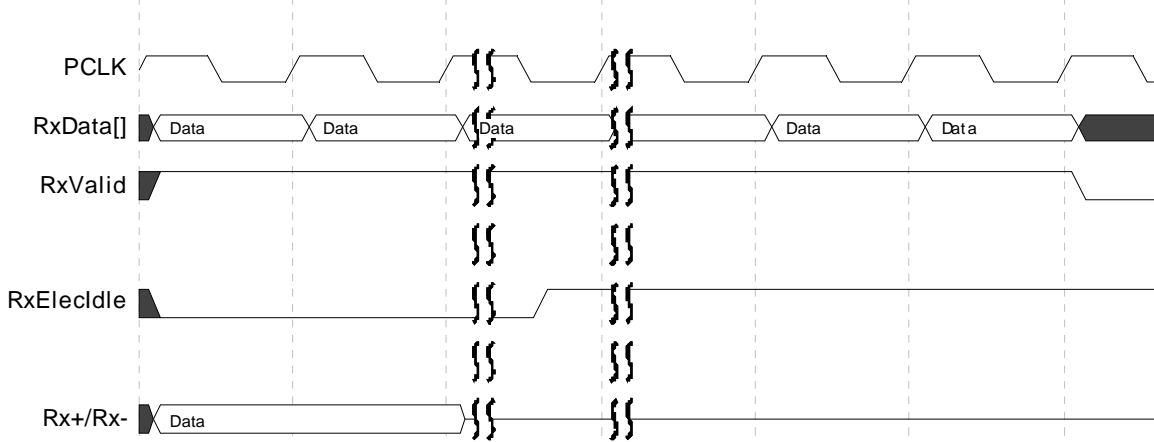
These sections show sample timing sequences for some of the more common SATA operations. These are *sample* sequences and timings and are not required operation.

7.1 Receivers and Electrical Idle

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.

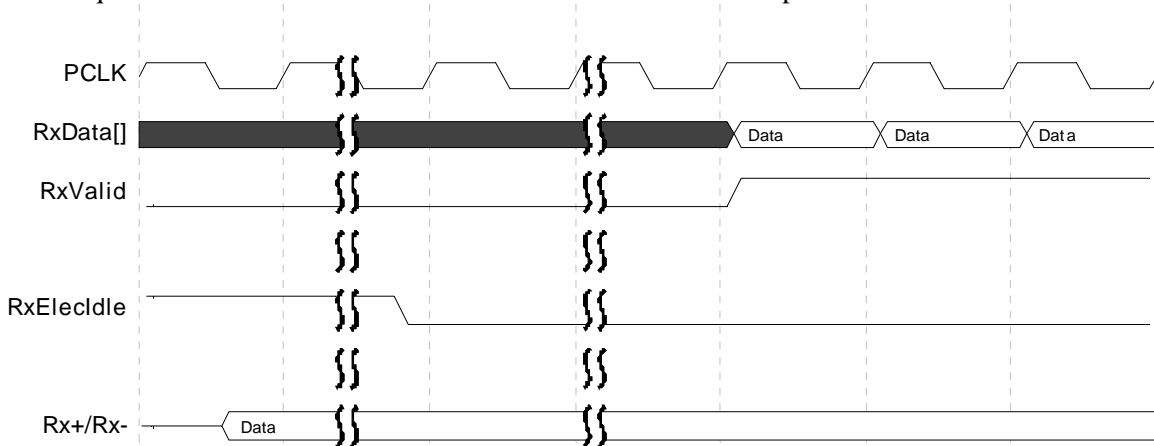
The first diagram shows how the interface responds when the receive channel has been active and then goes to electrical idle. In this case, the delay between *RxElecIdle* being asserted and *RxValid* being deasserted is directly related to the depth of the implementations elastic buffer and symbol

synchronization logic.



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.



Receiver Idle to Active

7.2 TX OOB Signaling

This example will be added in the next revision.