USB 2.0
Transceiver Macrocell Interface (UTMI) Specification

Version 1.05
3/29/2001

Please send comments via electronic mail to:
steve.mcgowan@intel.com

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.

INTEL DISCLAIMS ALL LIABILITY, INCLUDING LIABILITY FOR INFRINGEMENT OF ANY PROPRIETARY RIGHTS, RELATING TO IMPLEMENTATION OF INFORMATION IN THIS SPECIFICATION. INTEL DOES NOT WARRANT OR REPRESENT THAT SUCH IMPLEMENTATION(S) WILL NOT INFRINGE SUCH RIGHTS.

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.

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

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

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

All product names are trademarks, registered trademarks, or service marks of their respective owners.

Contributors

Jon Lueker
Steve McGowan (Editor)
Ken Oliver
Dean Warren

Table of Contents

Table of Figures

Table of Tables

1 Preface

1.1 Scope of this Revision

Version 1.05 of the USB 2.0 Transceiver Macrocell Interface (UTMI).

1.2 Revision History

Revision Number	Date	Description
1.05	3/30/01	1) Relaxed TX Start Delay Timing. See TXValid in Table 4 and footnote 3 and 6.4.4. 2) Clarify that TXValid can be asserted in the TX Wait and Send SYNC states. 3) Section 5.13, remove Bus Idle term from the Transmit State machine and add bullet to identify SIEs responsibilities to check for Bus Idle before asserting TXValid. 4) Section 5.13, added bullet to clarify the state of TXReady in the TX Wait and Send SYNC states. 5) Modified Figures 15 to 18 to show CRC byte ordering correctly. Also added explanatory notes. 6) Corrected TXValid signal description for FS assertion delay of SYNC pattern. 7) Cleaned up conflicting descriptions of the negation of RXActive after the assertion of RXError, so that they matched the RX State Machine. 8) Corrected conflicting CLK “Usable” duty cycle tolerance. 9) Corrected several timing/delay values in section 6. 10) Section 6.4, added timing information for 30MHz CLK. 11) Many clarifications, cross reference additions, and text corrections.
1.04	10/19/00	1) Deleted first 2 paragraphs in section 6.4. They were out of date. 2) In section 5.22.5, added discussion of how Vbus effects SuspendM and Reset. 3) Added section 4.1.5. A discussion of OpMode/TXValid timing. 4) Added section 4.1.1.1 to further clarify the CKL options that are available. 5) Relaxed FS timing on TXValid in Table 4 fron 2 to 5 CLKs. 6) Added footnote to Table 5 describing FS Idle State of RXActive. 7) Added clarifying text to Figure 8 (CRC5 calculation). 8) Expanded discussion of Bit Stuff error detection and reporting in section 5.8.1.1. 9) Corrected CRC hi/lo labels in Figure 16.
1.03	8/4/00	1) Corrected Section 5.2.2.2, 2nd paragraph, 2nd line, FS to HS. 2) Added section 4.1.4.3. 3) Corrected Figures 13, 15, and 16, and 6th bullet in section 5.13 to show proper DataIn timing for TXValid. 4) Changed Figure 4 to drop Don't Cares on DataIn. 5) Clarified the TXReady signal description. 6) Corrected DataIn timing in Figure 31. 7) Dropped mention of back-to-back packet transmissions in section 5.13. 8) In section 5.8 the Abort 2 and Terminate states were added to allow RXActive to be held after an error. Also added section 5.8.1.1. 9) Added section 6.4.6 Start and End Delay Summary. 10) Added section 4.4.1, a discussion of RXActive negation. 11) Modified the description of RXActive signal to reference "start of Idle state" vs. "end of EOP". 12) Deleted Note in section 4.1.4 and added sections 4.1.4.3 and 4.1.4.4. 13) Rewrote section 4.1.4.2 Signaling Levels. 14) Added Figures 36 and 37, rewrote sections 6.4.4.1 and 6.4.4.2. 15) Added sections 6.4.5.3 and 6.4.5.4
1.02	6/27/00	1) Corrected Section 5.22.2.2, the first paragraph now says that XcverSelect is switched at T1 (not T2). 2) Dropped "Disable LineState when XcvrSelect = HS" text in section 5.22. 3) Changed Fig 13 to show PID asserted earlier on DataIn and added

		comment to section 5.13. 4) Corrected DataIn and DataOut signal names in Figures 34, 35, and 36.
1.01	5/25/00	1) Added Vendor Controls, Sections 4.3 and 5.21. 2) Corrected text in sections 4.4 (moved timing descriptions from TXValid description to TXActive Description) and 6.4. In first paragraph, replaced TXValid with TXActive). 3) Dropped all references to HS Only devices. 4) Added Reset qualification to the description of DataBus16_8. 5) Modified description of Mode 2 in section 5.16.
1.0	5/22/00	1.0 Release

2 Introduction

High volume USB 2.0 devices will be designed using ASIC technology with embedded USB 2.0 support. For full-speed USB devices the operating frequency was low enough to allow data recovery to be handled in a vendors VHDL code, with the ASIC vendor providing only a simple level translator to meet the USB signaling requirements. Today's gate arrays operate comfortably between 30 and 60 MHz. With USB 2.0 signaling running at hundreds of MHz, the existing design methodology must change.

As operating frequencies go up it becomes more difficult to compile VHDL code without modification. This document defines the USB 2.0 Transceiver Macrocell Interface (UTMI) and many operational aspects of the USB 2.0 Transceiver Macrocell (UTM). The intent of the UTMI is to accelerate USB 2.0 peripheral development. This document defines an interface to which ASIC and peripheral vendors can develop. ASIC vendors and foundries will implement the UTM and add it to their device libraries. Peripheral and IP vendors will be able to develop their designs, insulated from the high-speed and analog circuitry issues associated with the USB 2.0 interface, thus minimizing the time and risk of their development cycles.

The figure below summarizes a number of concepts expressed throughout this spec. There are assumed to be three major functional blocks in a USB 2.0 peripheral ASIC design: the USB 2.0 Transceiver Macrocell, the Serial Interface Engine (SIE), and the device specific logic.

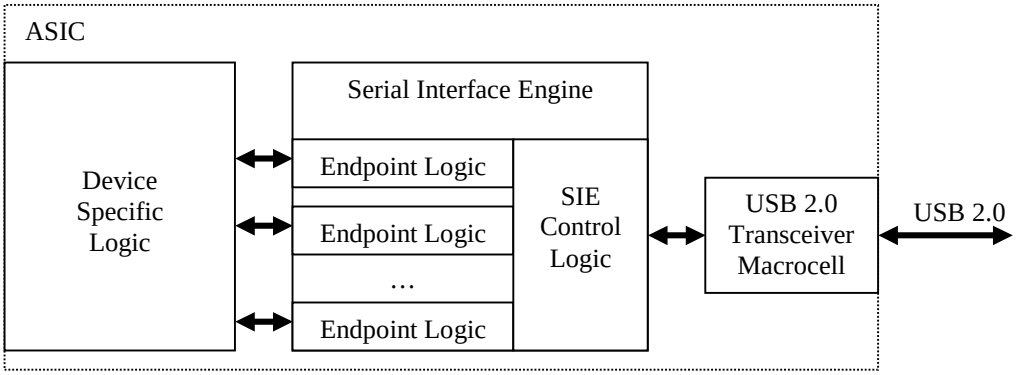


Figure 1: ASIC Functional Blocks

2.1 USB 2.0 Transceiver Macrocell (UTM)

This block handles the low level USB protocol and signaling. This includes features such as; data serialization and deserialization, bit stuffing and clock recovery and synchronization. The primary focus of this block is to shift the clock domain of the data from the USB 2.0 rate to one that is compatible with the general logic in the ASIC.

Some key features of the USB 2.0 Transceiver are:

- Eliminates high speed USB 2.0 logic design for peripheral developers
- Standard Transceiver interface enables multiple IP sources for USB 2.0 SIE VHDL
- Supports 480 Mbit/s "High Speed" (HS)/ 12 Mbit/s “Full Speed” (FS), FS Only and "Low Speed" (LS) Only 1.5 Mbit/s serial data transmission rates.
- Utilizes 8-bit parallel interface to transmit and receive USB 2.0 cable data
- SYNC/EOP generation and checking
- Allows integration of high speed components in to a single functional block as seen by the peripheral designer
- High Speed and Full Speed operation to support the development of "Dual Mode" devices
- Data and clock recovery from serial stream on the USB

- Bit-stuffing/unstuffing; bit stuff error detection
- Holding registers to stage transmit and receive data
- Logic to facilitate Resume signaling
- Logic to facilitate Wake Up and Suspend detection
- Supports USB 2.0 Test Modes
- Ability to switch between FS and HS terminations/signaling
- Single parallel data clock output with on-chip PLL to generate higher speed serial data clocks

The UTMI is designed to support HS/FS, FS Only and LS Only UTM implementations. The three options allow a single SIE implementation to be used with any speed USB transceiver. A vendor can choose the transceiver performance that best meets their needs.

A HS/FS implementation of the transceiver can operate at either a 480 Mb/s or a 12 Mb/s rate. Two modes of operation are required to properly emulate High-speed device connection and suspend/resume features of USB 2.0, as well as Full-speed connections if implementing a Dual-Mode device.

FS Only and LS Only UTM implementations do not require the speed selection signals since there is no alternate speed to switch to.

The USB 2.0 Transceiver can be placed in a low-power mode with the **SuspendM** signal.

2.2 Serial Interface Engine

This block can be further sub-divided into 2 types of sub-blocks; the SIE Control Logic and the Endpoint logic. The SIE Control Logic contains the USB PID and address recognition logic, and other sequencing and state machine logic to handle USB packets and transactions. The Endpoint Logic contains the endpoint specific logic: endpoint number recognition, FIFOs and FIFO control, etc. Generally the SIE Control Logic is required for any USB implementation while the number and types of endpoints will vary as function of application and performance requirements.

SIE logic module can be developed by peripheral vendors or purchased from IP vendors. The standardization of the UTMI allows compatible SIE VHDL to drop into an ASIC that provides the macrocell.

2.3 Device Specific Logic

This is the glue that ties the USB interface to the specific application of the device.

3 Functional Block Diagram

Figure 2 shows the functional block diagram of the USB 2.0 transceiver. Each block is discussed below.

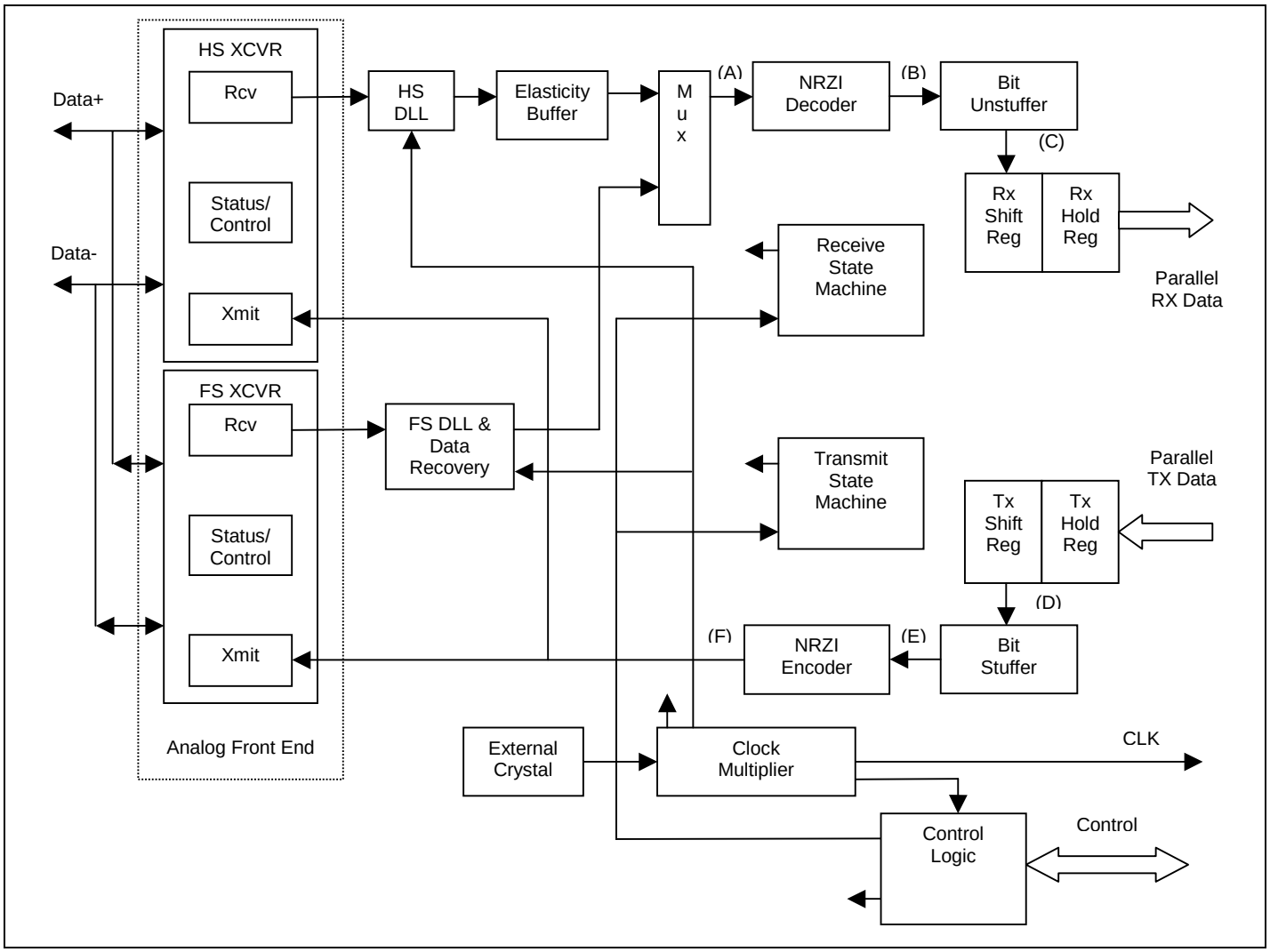


Figure 2: UTM Functional Block Diagram

Note: specific implementations of the UTM may combine, reorganize or otherwise modify the blocks in this diagram. For instance, serial data may be converted to a nibble-wide parallel format immediately and handled as 4-bit data between the HS DLL and the RX Hold Register.

4 UTMI Signal Descriptions

4.1 System Interface Signals

Table 1: System Interface Signals

Name	Direction	Active Level	Description
CLK	Output	Rising-Edge	**Clock**. This output is used for clocking receive and transmit parallel data. 60 MHz HS/FS, with 8-bit interface 30 MHz HS/FS, with 16-bit interface 48 MHz FS Only, with 8-bit interface 6 MHz LS Only, with 8-bit interface See section 4.1.1 for more information on **CLK**.
Reset	Input	High	**Reset.** Reset all state machines in the UTM.
Xcvr Select	Input	N/A	**Transceiver Select.** This signal selects between the FS and HS transceivers: 0: HS transceiver enabled 1: FS transceiver enabled This signal is not provided in FS Only and LS Only transceiver implementations. See section 4.1.2 for more information on **XcvrSelect**.
Term Select	Input	N/A	**Termination Select.** This signal selects between the FS and HS terminations: 0: HS termination enabled 1: FS termination enabled This signal is not provided in FS Only and LS Only transceiver implementations. See section 4.1.3 for more information on **TermSelect**.
SuspendM	Input	Low	**Suspend.** Places the Macrocell in a mode that draws minimal power from supplies. Shuts down all blocks not necessary for Suspend/Resume operation. While suspended, **TermSelect** must always be in FS mode to ensure that the 1.5K pull-up on **DP** remains powered. 0: Macrocell circuitry drawing suspend current 1: Macrocell circuitry drawing normal current
LineState (0-1)	Output	N/A	**Line State.** These signals reflect the current state of the single ended receivers. They are combinatorial until a "usable" **CLK** is available then they are synchronized to **CLK**. They directly reflect the current state of the **DP** (LineState[0]) and **DM** (LineState[1]) signals: DM DP Description 0 0 0: SE0 0 1 1: 'J' State 1 0 2: 'K' State 1 1 3: SE1 See section 4.1.4 for more information on **LineState**.

OpMode (0-1)	Input	N/A	**Operational Mode.** These signals select between various operational modes: [1] [0] Description 0 0 0: Normal Operation 0 1 1: Non-Driving 1 0 2: Disable Bit Stuffing and NRZI encoding 1 1 3: Reserved

4.1.1 CLK

Nominal **CLK** accuracy is ±500ppm for frequency, and 50±5% duty cycle.

No transitions of **CLK** should occur until it is "usable", where usable is defined as a frequency accuracy of ±10%, and a duty cycle accuracy of 50±10%.

Conceptually, there is a "CLKUsable" signal, internal to the UTM, which blocks any transitions of **CLK** until it is "usable". This "CLKUsable" signal is also used to switch the **LineState** output between **CLK** synchronized and combinatorial signaling. See section 5.22.2.3 for further discussion of **CLK**.

4.1.1.1 Options

There are 3 possible implementations for a UTMI device: HS/FS, FS Only, or LS Only. The HS/FS version has 4 interface options: 16-bit unidirectional, 8-bit unidirectional, 16-bit bidirectional/8-bit unidirectional, and 8-bit bi-directional. In each case, when a 16-bit option is selected CLK is at 30 MHz, and when an 8-bit option is selected **CLK** is at 60 MHz.

Note that the 16-bit bidirectional/8-bit unidirectional uses the "DataBus16_8" signal to swtich between them. This signal also switches the CLK frequency.

The FS Only, or LS Only implementations only support 48 MHz and 6 MHz clocks, respectively, and always use 8 bit interfaces (either 8-bit unidirectional or 8-bit bi-directional).

4.1.2 XcvrSelect

XcvrSelect controls a number of transceiver related elements, for instance.

- Selects the receiver (source for the Mux block) in the receive data path.
- It is used as a gating term for enabling the respective HS or FS Transmit Driver.
- Switch internal UTM clocks to shared logic.

4.1.3 TermSelect

TermSelect controls a number of termination related elements, for instance.

- In HS mode the FS Driver is forced to assert an SE0 on the USB, providing the 50 Ohm termination to ground and generating the HS Idle state on the bus.
- In FS Mode **TermSelect** enables the 1.5K pull-up on to the **DP** signal to generate the FS Idle state on the bus.

4.1.4 LineState

The **LineState** signals are used by the SIE for detecting reset, speed signaling, packet timing, and to transition from one behavior to another.

Note: While data packets are being transmitted or received on the USB the **LineState** signals may toggle randomly between the 'J' and 'K' states in FS, and remain in the ‘J’ state in HS. The SIE should ignore these transitions.

4.1.4.1 Synchronization

To minimize unwanted transitions to the SIE during normal operation, the **LineState** is internally synchronized with **CLK**. When synchronized, the setup and hold timing of **LineState** is identical to **DataOut**. The exception to this is when **CLK** is not "usable". If **CLK** is not "usable" then the **LineState**

signals are not synchronized, but driven with combinatorial logic directly from the **DP** and **DM** signal lines. The UTM must multiplex between combinatorial and synchronous LineState output depending on whether CLK is "usable". See section 4.1.1 for a discussion of what a "usable" **CLK** is. See section 4.1.4.3 for an addition method of minimizing LineState transitions in HS mode.

4.1.4.2 Signaling Levels

The voltage thresholds that the **LineState** signals use for comparison on **DP** and **DM** depend on the state of **XcvrSelect**. **LineState** uses HS thresholds when the HS transceiver is enabled (**XcvrSelect** = 0) and FS thresholds when the FS transceiver is enabled (**XcvrSelect** = 1). FS Only and LS Only implementations always use FS thresholds. See first paragraph of section 5.22 for more details.

There is no concept of variable, single ended thresholds in the USB 2.0 specification. The assumption (see Figure 7-1 in the USB 2.0 spec) was that the HS receiver would be used to detect a Chirp K or J, where the output of the HS receiver is always qualified with the "Squelch" signal. If Squelch = 1 then the output of the HS receiver is meaningless.

In the macrocell, as an alternative to using variable thresholds for the single ended receivers the following approach to encoding the LineState outputs can be used.

Mode		Full Speed	High Speed	Chirp	Invalid
XcvrSelect		1	0	0	1
TermSelect		1	0	1	0
Line State	SE0	SE0	Squelch	Squelch	Invalid
	J State	J	!Squelch[1]	! Squelch & HS_Differential_Receiver_Output	Invalid
	K State	K	Invalid	! Squelch & ! HS_Differential_Receiver_Output	Invalid
	SE1	SE1	Invalid	Invalid	Invalid

Note: With this scheme, SE1 is never generated in HS Mode. This is not a problem because SE1 is defined as an illegal bus state in the USB 2.0 specification and is provided by the UTM for debug purposes only.

Note: An SIE attached to a LS Only UTM implementation must interpret the K State as Bus Idle.

4.1.4.3 Minimizing Transitions

In HS mode, 3 ms of no USB activity (Idle state) signals a reset. The SIE monitors **LineState** for Idle state[2]. If in HS mode, **LineState** is simply the output of the HS Differential receiver then **LineState** will toggle randomly while packets are on the USB. To minimize transitions on **LineState** while in HS mode the presence of Squelch can be used to force a **LineState** to a J State.

This scheme allows **LineState** to indicate a J State whenever a packet is on the USB, thus satisfying the requirement that a **LineState** transition occurs when there is activity on the USB, while minimizing the number of **LineState** transitions while there is data on the bus. Using **TermSelect**, rather than **XcvrSelect**, allows the Speed Chirp protocol to complete before enabling this mode.

This approach has the side effect of turning any Chirp K's after the HS terminations are enabled (**TermSelect** = 0. See Figure 25, T7 to T8) into Chirp J's. However, this only occurs after the SIE has determined that it is attached to a HS downstream facing port and the SIE is no longer interested in whether a J or K is on the bus.

[1] Note: This term is optional. See section 4.1.4.3 for a discussion of this term.
[2] Note: When identifying bus "activity", the SIE should not use the data path to monitor SOFs.

4.1.4.4 Bus Packet Timing

LineState must be used by the SIE for the precise timing of packet data on the DP/DM signal lines. The SIE uses **LineState** transitions to identify the beginning and end of receive or transmit packets on the bus. Due to internal UTM buffering and pipeline delays (which are implementation dependent), the receive (**RXActive**) and transmit (**TXValid**) control signals provide only a coarse indication of the actual activity on the bus. **LineState** represents bus activity within 2 or 3 **CLK** times of the actual events on the bus.

HS Mode

When **XcvrSelect** and **TermSelect** are in HS mode, the **LineState** transition from the Idle state (SE0) to a non-Idle state (J) marks the beginning of a packet on the bus. The **LineState** transition from a non-Idle state (J) to the Idle state (SE0) marks the end of a packet on the bus.

FS Mode

When **XcvrSelect** and **TermSelect** are in FS mode, the **LineState** transition from the J State (Idle) to a K State marks the beginning of a packet on the bus. The SIE must then wait for the end of the packet. The **LineState** transition from the SE0 to the J-State marks the end of a FS packet on the bus.

4.1.5 OpMode

When a device generates resume signaling to the host, it switches the **OpMode** to "Disable Bit Stuffing and NRZI Encoding", asserts **TXValid**, and presents the data on the **DataOut** bus. The assertion of OpMode to “Normal” mode at the end of the 1 ms signaling period should occur until after the maximum *TX End Delay* (TXValid has been de-asserted for at least 40 bit times or in FS mode 160 CLKs). See section 0 for a discussion of *TX End Delay*.

SIE designers note: if **OpMode** switched to “Normal” mode before the maximum *TX End Delay* completes, then there is the possibility that the last data still pending in the UTM will be NRZI encoded and bit stuffed (in case 6 1's occur), resulting in K and J transitions on the **DP/DM** signal lines at the end of resume from the device. At this time the downstream facing port will also be propagating back the K state (detected device resume) onto all enabled down stream ports. This creates bus conflict on **DP/DM**.

4.2 USB Interface Signals

Table 2: USB Interface Signals

Name	Direction	Active Level	Description
DP	Bidir	N/A	USB data pin Data+
DM	Bidir	N/A	USB data pin Data–

Note: These signals are listed here for the completeness. They are not actually part of the Transceiver Macrocell interface to the SIE, however they are referred to often in this specification. See the USB 2.0 spec for details of **DP** and **DM** timing and signal levels.

4.3 Vendor Control Signals

These signals are provided for vendor-defined error, status and control information. All the signals are synchronous with **CLK** and use the same setup and hold times as the data signals. These signals are optional for a transceiver, however SIEs are required to make these registers accessible to system software, so that detailed diagnostic and error analysis can be performed.

Table 3: Vendor Control Signals

Name	Direction	Active Level	Description
VControlLoadM	Input	Low	**Vendor Control Load.** Assertion of this signal loads the Vendor Control register: 0: Load Vendor Control Register 1: NOP See section 5.21 for more information.
VControl0-3	Input	Vendor Defined	**Vendor Control**. Vendor defined 4-bit parallel input bus. Note these pins are optional, a macrocell may define partial sets as well. i.e. **VControl0-1**.
VStatus0-7	Output	Vendor Defined	**Vendor Status**. Vendor defined 8-bit parallel output bus. Note these pins are optional, a macrocell may define partial sets as well. i.e. **VStatus0-1**.

4.4 Data Interface Signals

Note: For all multi-bit signal descriptions, bit[0] is always the least significant bit of the referenced value.

Table 4: Data Interface Signals (Transmit)

Name	Direction	Active Level	Description
DataIn0-7	Input	N/A	**DataIn**. 8-bit parallel USB data input bus. When **DataBus16_8** = 1 this bus transfers the low byte of 16-bit transmit data. When **DataBus16_8** = 0 all transmit data is transferred over this bus.
DataIn8-15	Input	N/A	**DataIn**. An 8-bit parallel USB data input bus that transfers the high byte of 16-bit transmit data. These signals are only valid when **DataBus16_8** = 1.
TXValid	Input	High	**Transmit Valid**. Indicates that the **DataIn** bus is valid. The assertion of Transmit Valid initiates SYNC on the USB. The negation of Transmit Valid initates EOP on the USB. In HS (**XcvrSelect** = 0) mode, the SYNC pattern must be asserted on the USB between 8 and 16 bit times after the assertion of **TXValid** is detected by the Transmit State Machine. See section 6.4 for more information. In FS (**XcvrSelect** = 1), FS Only, or LS Only modes, the SYNC pattern must be asserted on the USB no less than 1 CLK and no more than 5 10 CLKs[3] after the assertion of **TXValid** is detected by the Transmit State Machine.
TXValidH	Input	High	**Transmit Valid High**. When **DataBus16_8** = 1, this signal indicates that the **DataIn(8-15)** bus contains valid transmit data. This signal is ignored when **DataBus16_8** = 0. This signal is not provided in 8-Bit transceiver implementations.
TXReady	Output	High	**Transmit Data Ready**. If **TXValid** is asserted, the SIE must always have data available for clocking in to the TX Holding Register on the rising edge of **CLK**. If **TXValid** is TRUE and **TXReady** is asserted at the rising edge of **CLK**, the UTM will load the data on the **DataIn** bus into the TX Holding Register on the next rising edge of **CLK**, at that time, SIE should immediately present the data for next transfer on the **DataIn** bus. If **TXValid** is asserted and **TXReady** is negated, the SIE must hold the previously asserted data on the **DataIn** bus. From the time **TXValid** is negated, **TXReady** is a don't care for the SIE.

[3] Note that the number of CLKs depends on the bus and interface options:
FS 16-bit interface (CLK = 30 MHz) Max TX Start Delay = 5 CLKs
FS 8-bit interface (CLK = 60 MHz) Max TX Start Delay = 10 CLKs
FS Only 8-bit interface (CLK = 48 MHz) Max TX Start Delay = 8 CLKs
LS Only 8-bit interface (CLK = 6 MHz) Max TX Start Delay = 8 CLKs

Table 5: Data Interface Signals (Receive)

Name	Direction	Active Level	Description
DataOut0-7	Output	N/A	**DataOut**. 8-bit parallel USB data output bus. When **DataBus16_8** = 1 this bus transfers the low byte of 16-bit receive data. When **DataBus16_8** = 0 all receive data is transferred over this bus.
DataOut8-15	Output	N/A	**DataOut**. An 8-bit parallel USB data output bus that transfers the high byte of 16-bit receive data. These signals are only valid when **DataBus16_8** = 1.
RXValid	Output	High	**Receive Data Valid**. Indicates that the **DataOut** bus has valid data. The Receive Data Holding Register is full and ready to be unloaded. The SIE is expected to latch the **DataOut** bus on the clock edge.
RXValidH	Output	High	**Receive Data Valid High**. When **DataBus16_8** = 1 this signals indicates that the **DataOut(8-15)** bus is presenting valid receive data. This signal is ignored when **DataBus16_8** = 0. This signal is not provided in 8-Bit transceiver implementations.
RXActive	Output	High	**Receive Active**. Indicates that the receive state machine has detected SYNC and is active. **RXActive** is negated after a Bit Stuff Error or an EOP is detected. See the RX State Machine (Figure 6) for more details on the negation conditions for **RXActive**. In HS mode (**XcvrSelect** = 0), **RXActive** must be negated no less than 3 and no more that 8 **CLK**s after an Idle state is detected on the USB. And **RXActive** must be negated for at least 1 **CLK** between consecutive received packets. See sections 4.4.1 and 6.4 for more information. In FS (**XcvrSelect** = 1), FS Only, or LS Only modes, **RXActive** must be negated no more than 2 **CLK**s after a FS Idle state[4] is detected on the USB. And **RXActive** must be negated for at least 4 **CLK**s between consecutive received packets.
RXError	Output	High	**Receive Error**. 0 Indicates no error. 1 Indicates that a receive error has been detected. Possible sources of errors are discussed in section 5.8.1. This output is clocked with the same timing as the **DataOut** lines and can occur at anytime during a transfer. If asserted, it will force the negation of **RXValid** on the next rising edge of **CLK**. See sections 4.4.1 and 5.8 for more information on the interaction of **RXError** with the Receive State Machine.

4.4.1 Receive Active

RXActive is used by the SIE to time inter-packet gaps. It is important that **RXActive** accurately reflects the state of the USB. For instance, HS implementations should not simply negate **RXActive** as soon as a forced

[4] A FS Idle state follows a FS EOP, where a FS EOP is an SE0 asserted for 2 FS times followed by a J asserted for 1 FS bit time. See Figure 37: FS Receive to transmit inter-packet delayFigure 37.

bit stuff error (EOP) is detected. Two HS cases in particular should be considered: 1) dribble bits[5] introduced by hubs and 2) long EOPs at the end of SOF packets. In both cases, the initial bit stuff error that signals the EOP is followed by several additional bits before the USB returns to the Idle state. The deassertion of **RXActive** under normal conditions must reflect the USB being in the Idle state, not simply timed off the recognition of EOP. The exception is if an error is detected during a receive. In this case RXValid will be negated after the next rising edge of CLK after the error is detected. And RXActive may be negated immediately or after a Bus Idle condition is detected. See section 5.8 for more information.

It is recommended that for HS packets, the internal "squelch" signal of the UTM be used to qualify the negation of **RXActive** under normal conditions, because squelch indicates an SE0 (HS Idle State) on the bus.

Table 6: Data Interface Signals (16-bit Bi-directional)

Name	Direction	Active Level	Description
Data0-7	Bidir	N/A	**Data**. 8-bit parallel USB data input bus when **DataBus16_8** = 0. Low byte of bi-directional parallel USB data bus when **DataBus16_8** = 1.
Data8-15	Bidir	N/A	**Data**. 8-bit parallel USB data output bus when **DataBus16_8** = 0. High byte of bi-directional parallel USB data bus when **DataBus16_8** = 1. **DataBus16_8** may only be changed while **Reset** is asserted.
ValidH	Bidir	High	**ValidH**. This signal indicates that the high order 8 bits of a 16-bit data word presented on the **Data** bus are valid. When **DataBus16_8** = 1 and **TXValid** = 0, **ValidH** is an output, gating **RXValidH** to the SIE, indicating that the high order receive data byte on the **Data** bus is valid. When **DataBus16_8** = 1 and **TXValid** = 1, **ValidH** is an input and indicates that the high order transmit data byte, presented on the **Data** bus by the transceiver is valid. When **DataBus16_8** = 0, **ValidH** is undefined. The status of the low order data byte is determined by **TXValid** and **RXValid**.

[5] Each hub is allowed to introduce up to 4 dribble bits to the end of a packet. i.e. A device that is attached 5 hubs deep will see up to 20 dribble bits.

Table 7: Data Interface Signals (Other)

Name	Direction	Active Level	Description
DataBus 16_8	Input	High	**Data Bus 16 - 8**. Selects between 8 and 16 bit data transfers. 1 16-bit data path operation enabled. DataIn(8-15), DataOut(8-15), TXValidH, and RXValidH operational. CLK = 30 MHz. 0 8-bit data path operation enabled. DataIn(8-15), DataOut(8-15), TXValidH, and RXValidH undefined. CLK = 60 MHz. Note that 16 bit operation is only an option for a HS/FS transceiver implementation. Note **DataBus16_8** is only sampled by the macrocell on the negation of **Reset**.

5 Block level Descriptions

This section provides descriptions of each of the blocks shown in Figure 2. These blocks represent high level functionality that is required to exist in the Macrocell.

5.1 Clock Multiplier

This module generates the appropriate internal clocks for the UTM and the **CLK** output signal. All data transfer signals are synchronized with the **CLK** signal.

The UTM vendor determines the frequency of the external crystal. The Clock Multiplier circuit and the External Crystal must meet the requirements defined in the USB 2.0 specification.

After the release of **SuspendM**, the **CLK** signal generated by the transceiver must meet the following requirements:

1) Produce the first **CLK** transition no later than 5.6 ms after the negation of **SuspendM**.

2) The **CLK** signal frequency error must be less than 10% (±6.00 MHz)

3) The **CLK** must fully meet the required accuracy of ±500 ppm (±30.0 KHz), no later than 1.4ms after the first transition of **CLK**.

5.1.1 Clocking

5.1.1.1 HS/FS operation

In HS mode there is one **CLK** cycle per byte time. The frequency of **CLK** does not change when the UTMI is switched between HS to FS modes. In FS mode there are 5 **CLK** cycles per FS bit time, typically 40 **CLK** cycles per FS byte time. If a received byte contains a stuffed bit then the byte boundary can be stretched to 45 **CLK** cycles, and two stuffed bits would result in a 50 **CLK** delay between bytes.

Figure 3 shows the relationship between **CLK** and the receive data transfer signals in FS mode. **RXActive** "frames" a packet, transitioning only at the beginning and end of a packet, however transitions of **RXValid** may take place any time 8 bits of data are available. Figure 3 also shows how **RXValid** is only asserted for one **CLK** cycle per byte time even though the data may be presented for the full byte time. The Macrocell is required to present valid data for only for one clock cycle (while **RXValid** is asserted), although it may be presented until new data is received.

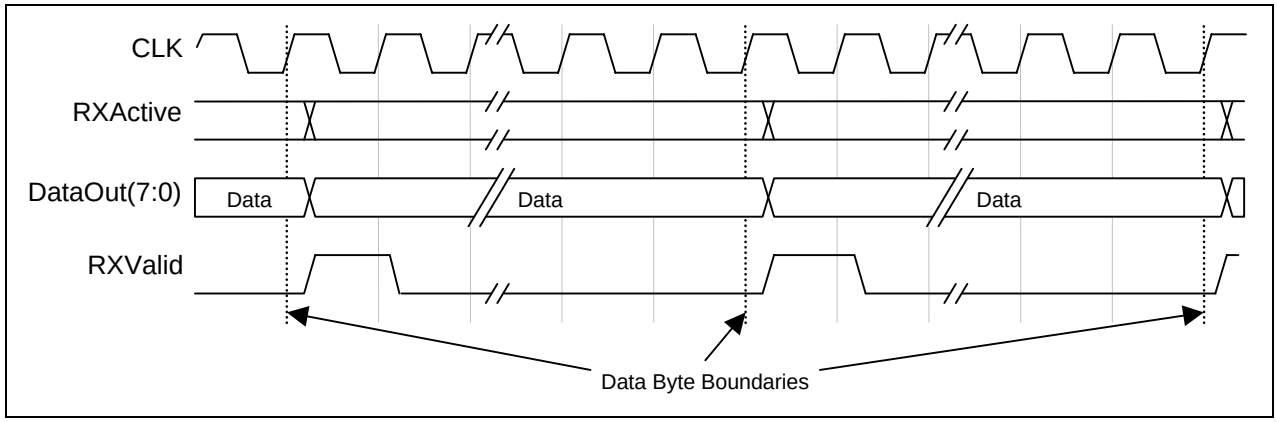


Figure 3: FS CLK Relationship to Receive Data and Control Signals

Figure 4 shows relationship between **CLK** and the transmit data transfer signals in FS mode. **TXReady** is only asserted for one **CLK** per byte time. This signal acknowledges to the SIE that the data on the **DataIn**

lines has been read by the Macrocell (small arrows above DataIn signal. The SIE must present the next data byte on the **DataIn** bus after it detects **TXReady** high on a rising edge of **CLK**.

Transitions of **TXValid** must meet the defined setup and hold times relative to **CLK**. The delay between the assertion of **TXValid** and the first assertion of **TXReady** is Macrocell implementation dependent.

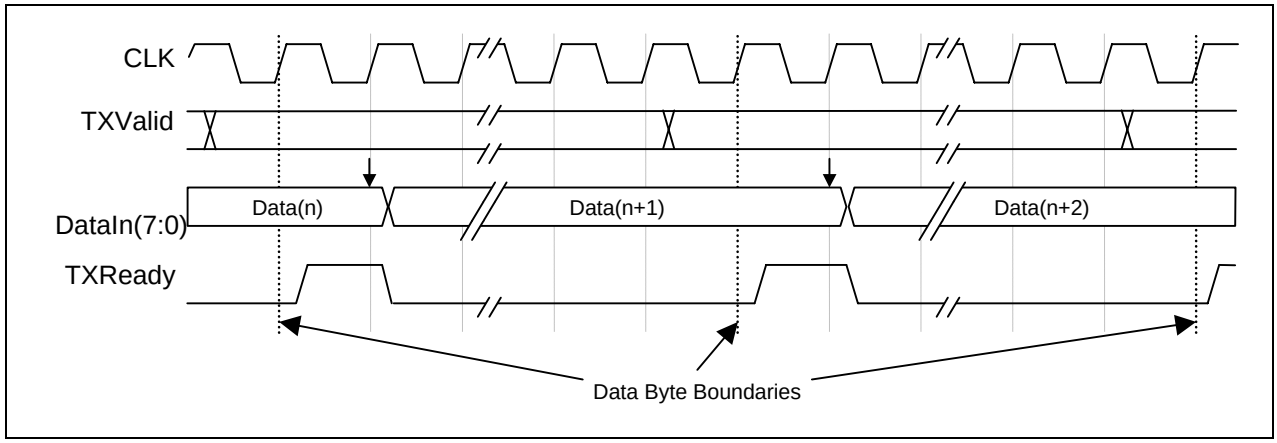


Figure 4: FS CLK Relationship to Transmit Data and Control Signals

The **XcvrSelect** signal determines whether the HS or FS timing relationship is applied to the data and control signals.

5.1.1.2 FS Only operation

A "FS Only" implementation of the UTM would provide 32 **CLK** cycles per byte time. The frequency of **CLK** would be 48.0 MHz. Timing is similar to a HS/FS UTM operating in FS mode.

5.1.1.3 LS Only operation

A "LS Only" implementation of the UTM would provide 32 **CLK** cycles per byte time. The frequency of **CLK** would be 6.0 MHz. Timing is similar to a HS/FS UTM operating in FS mode.

5.2 HS DLL (High Speed Delay Line PLL)

The delay line PLL extracts clock and data from the data received over the USB 2.0 interface for reception by the Receive Deserializer. A vendor defined number of delayed clock taps are be used to sample the received data. The data output from the DLL is synchronous with the local clock.

5.3 Elasticity Buffer

This buffer is used to compensate for differences between transmitting and receiving clocks. The USB specification defines a maximum clock error of +/-500 ppm. When the error is calculated over the maximum packet size up to +/- 12 bits of drift can occur. The elasticity buffer is filled to a threshold prior to enabling the remainder of the down stream receive logic. This block may be integrated into the DLL block. An example that will meet these requirements is a 24 bit deep, 1 bit wide FIFO with a threshold set at the midpoint.

Overflow or underflow conditions detected in the elasticity buffer can be reported with the **RXError** signal.

5.4 Mux

The bulk of the logic in the transceiver can be used with HS or FS operations. The Mux block allows the data from the HS or FS receivers to be routed to the shared receive logic. The state of the Mux is determined by the **XcvrSelect** input.

5.5 NRZI Decoder

This is a standard USB 1.X compliant serial NRZI decoder module, which can operate at FS or HS USB data rates.

5.6 Bit Unstuff Logic

This is a standard USB 1.X compliant serial bit unstuff module, which can operate at FS or HS USB data rates. The bit unstuff logic is a state machine, which strips a stuffed 0 bit from the data stream and detects bit stuff errors. In FS mode bit stuff errors assert the **RXError** signal. In HS mode bit stuff errors are used to generate the EOP signal so the **RXError** signal is not asserted.

The bit rate on USB is constant, however the bit rate as presented by the UTMI to the SIE is slightly reduced due to the extraction of inserted 1 bits. Normally a byte of data is presented on the **DataOut** bus for every 8 bits received, however after eight stuffed bits are eliminated from the data stream a byte time is skipped in the **DataOut** stream. Figure 5 shows how **RXValid** is used to skip bytes in the **DataOut** byte stream.

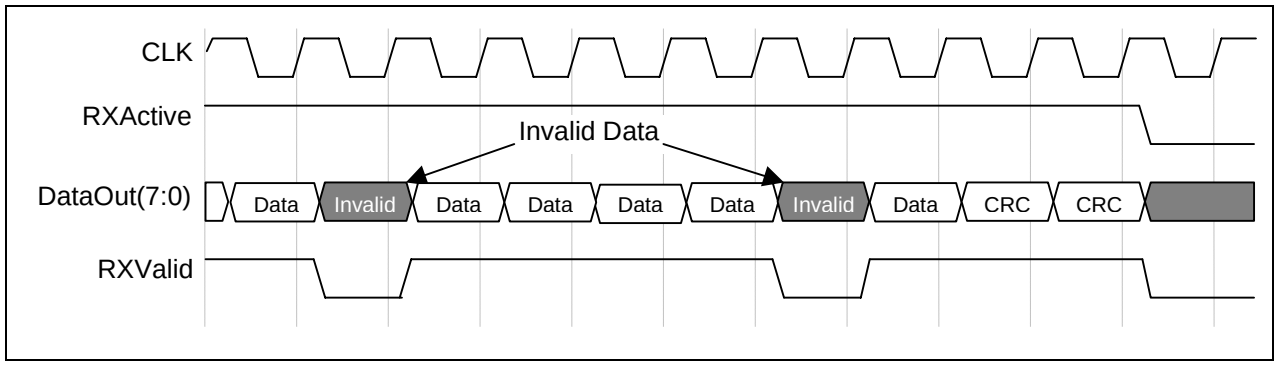


Figure 5: Receive Timing for Data with after Unstuffing Bits

Note that the timing in Figure 5 is for HS mode.

In FS mode, if a bit stuff error is detected then the Receive State Machine will assert **RXError**. See section 5.8.1 for more information.

5.7 Rx Shift/Hold Register

This module is responsible for de-serializing received data recovered by the DLL and transmitting 8-bit parallel data to the application bus interface. This module consists of an 8-bit primary shift register for serial to parallel conversion and an 8-bit Hold register used to buffer the last deserialized data byte.

5.8 Receive State Machine

The behavior of the Receive State Machine is described below and illustrated in Figure 6.

The assertion of **Reset** will force the Receive State Machine into the *Reset* state. The *Reset* state negates **RXActive** and **RXValid**. When the **Reset** signal is negated the Receive State Machine enters the *RX Wait* state and starts looking for a SYNC pattern on the USB. When a SYNC pattern is detected the state machine will enter the *Strip SYNC* state and assert **RXActive**. The length of the received SYNC pattern varies and can be up to 32 bits long. As a result, the state machine may remain in the *Strip SYNC* state for several byte times before capturing the first byte of data and entering the *RX Data* state.

After 8 bits of valid serial data is received the state machine enters the *RX Data* state, where the data is loaded into the RX Holding Register on the rising edge of **CLK** and **RXValid** is asserted. The SIE must clock the data off the **DataOut** bus on the next rising edge of **CLK**.

Stuffed bits are stripped from the data stream. Each time 8 stuffed bits are accumulated the state machine will enter the *RX Data Wait* state, negating **RXValid** thus skipping a byte time.

When the EOP is detected the state machine will enter the *Strip EOP* state and negate **RXActive** and **RXValid**. After the EOP has been stripped the Receive State Machine will reenter the *RX Wait* state and begin looking for the next packet.

If a Receive Error is detected, the *Error* State is entered and **RXError** is asserted. Then either the *Abort 1* State is entered where **RXActive**, **RXValid**, and **RXError** are negated, or the *Abort 2* State is entered where only **RXValid**, and **RXError** are negated. The *Abort 1* State proceeds directly to the *RX Wait* State, while *Abort 2* State proceeds to the *Terminate* State after an Idle bus state is detected on **DP** and **DM**. The *Terminate* State proceeds directly to the *RX Wait* State.

When the last data byte is clocked off the **DataOut** bus the SIE must also capture the state of the **RXError** signal.

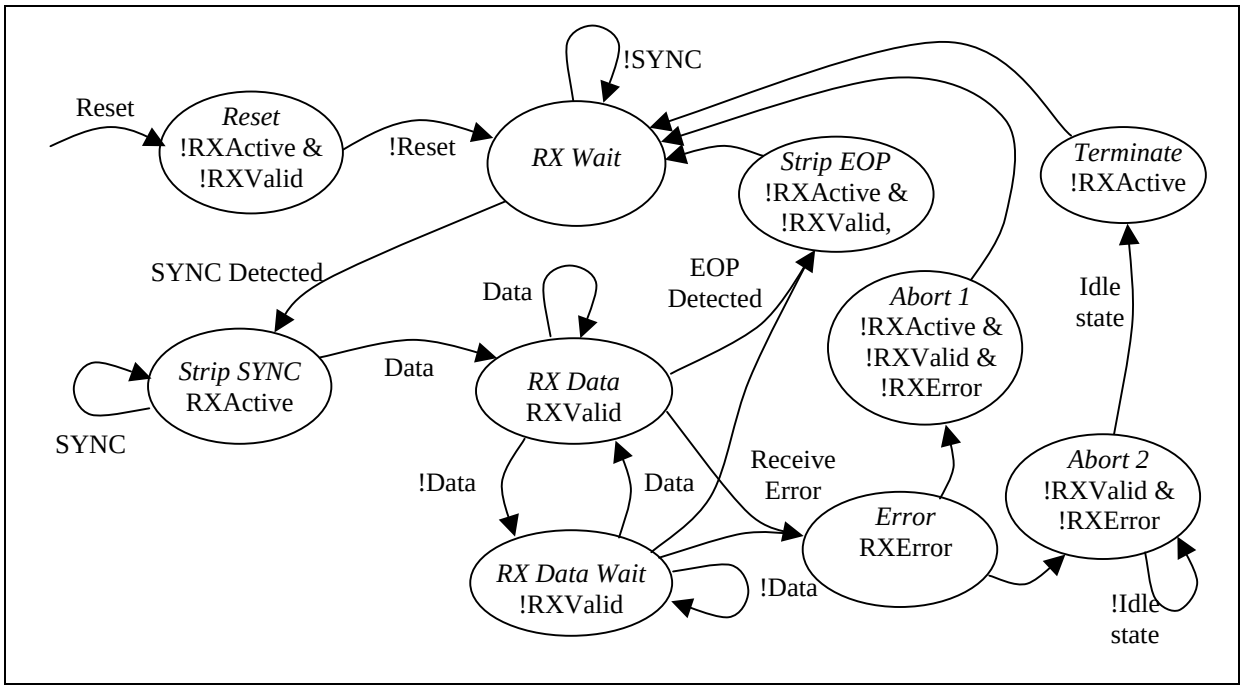


Figure 6: Receive State Diagram

- **RXActive** and **RXValid** are sampled on the rising edge of **CLK**.
- In the *RX Wait* state the receiver is always looking for SYNC.
- The Macrocell asserts **RXActive** when SYNC is detected (Strip SYNC state).
- The Macrocell negates **RXActive** when an EOP is detected (Strip EOP state).
- When **RxActive** is asserted, **RXValid** will be asserted if the RX Holding Register is full.
- **RXValid** will be negated if the RX Holding Register was not loaded during the previous byte time. This will occur if 8 stuffed bits have been accumulated.
- The SIE must be ready to consume a data byte if **RXActive** and **RXValid** are asserted (RX Data state).
- In FS mode, if a bit stuff error is detected then the Receive State Machine will negate **RXActive** and **RXValid**, and return to the *RXWait* state.

Figure 7 shows the timing relationship between the received data (**DP**/**DM**) , **RXValid**, **RXActive**, **RXError** and **DataOut** signals.

Note that the USB 2.0 Transceiver does NOT decode Packet ID's (PIDs). They are passed to the SIE for decoding.

Note: Figure 7, Figure 8 and Figure 9 are timing examples of a HS/FS UTM when it is in HS mode. When a HS/FS UTM is in FS Mode there are approximately 40 **CLK** cycles every byte time. The Receive State Machine assumes that the SIE captures the data on the **DataOut** bus if **RXActive** and **RXValid** are asserted. In FS mode, **RXValid** will only be asserted for one **CLK** per byte time. See section 5.1.1 for more information on FS clocking. The clocking of a HS/FS UTM in FS mode is similar to FS Only and LS Only implementations, except that for FS Only and LS Only implementations there are only 8 **CLK** cycles per byte time.

Note: The receive and transmit sections of the transceiver operate independently. The receiver will "receive" any packets on the USB. The transceiver does not identify whether the packet that it is receiving is from the upstream or the downstream port. The SIE must ignore receive data while it is transmitting.

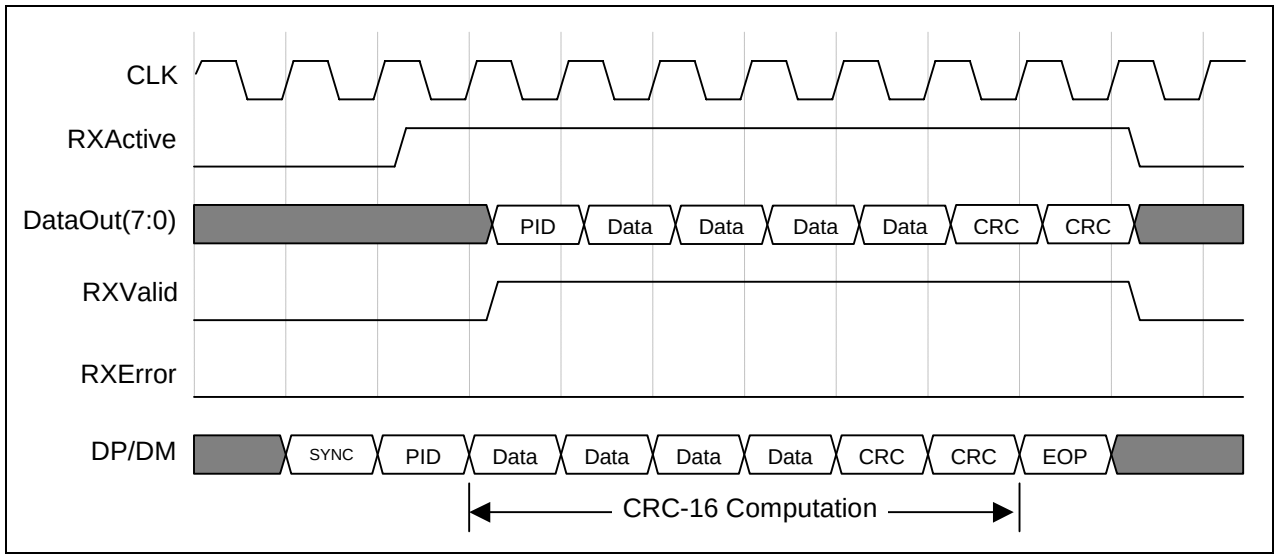


Figure 7: Receive Timing for Data Packet (with CRC-16)

Note: In Figure 7, Figure 8 and Figure 9 the SYNC pattern on **DP**/**DM** is shown as one byte long. The SYNC pattern received by a device can vary in length, these figures assume that all but the last 12 bits have been consumed by the hubs between the device and the host controller.

Note: In Figure 7, Figure 8 and Figure 9 the packet displayed on **DP**/**DM** may be pipelined by the UTM and occur several bit times, or even several byte times earlier, relative to the UTMI signal transitions (RXActive, DataOut, RXValid, etc.). Macrocell implementations should minimize internal latencies.

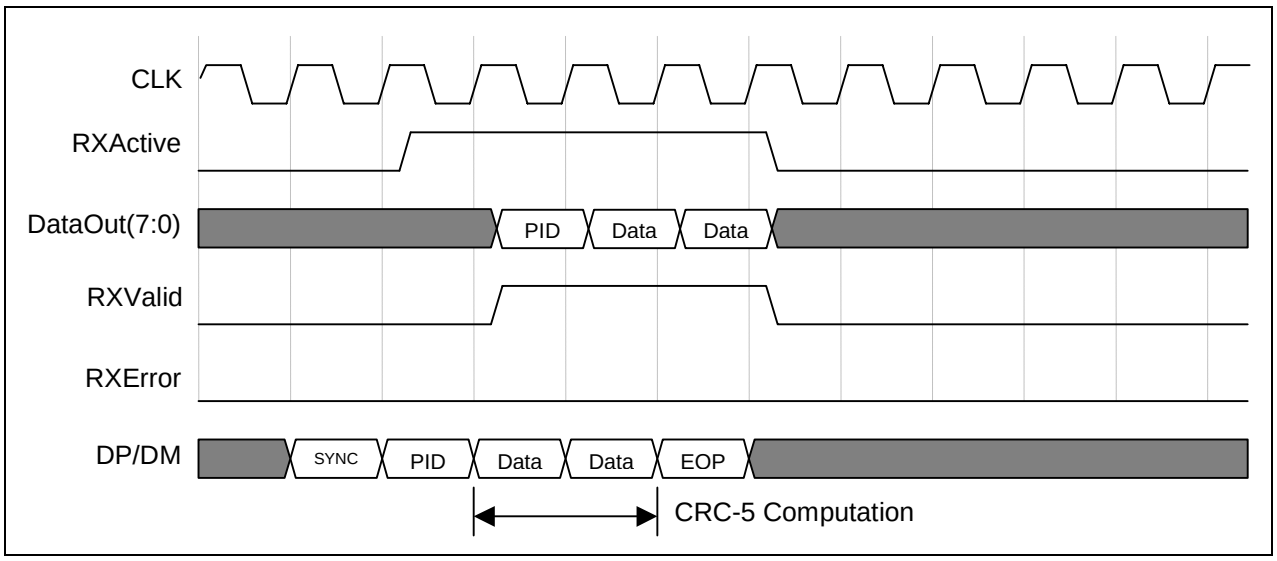


Figure 8: Receive Timing for Setup Packet

Note that the 2 "Data" bytes (16 bits) displayed in Figure 8 are really divided into 3 fields: Address (7 bits), Endpoint (4 bits), and CRC5 (5 bits). The CRC-5 is actually calculated over the Address and Endpoint fields. See section 8.4.1 in the USB 2.0 Specification for more information on Token Packets.

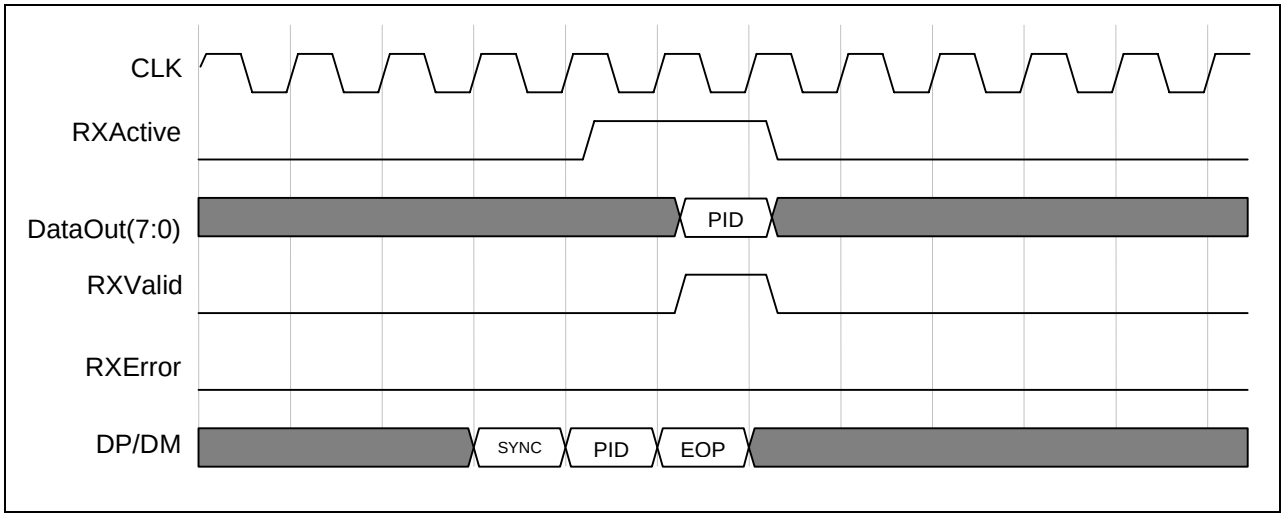


Figure 9: Receive Timing for a Handshake Packet (no CRC)

5.8.1 Receive Error Reporting

If an error is detected by the receiver the receive state machine will enter the *Error* state and assert **RXError**. It will then transition to the *Abort1* or *Abort2/Terminate* state, terminating the receive operation. Any data received while **RXError** is asserted should be ignored. See Figure 10. The Receive State Machine will then enter the *RX Wait* state and start looking for a valid SYNC pattern.

Possible sources of receive errors.

- Bit stuff error has been detected during a FS receive operation
- Elasticity Buffer overrun
- Elasticity Buffer underun
- Loss of sync by the DLL
- Alignment error, EOP not on a byte boundary
- Vendor Specific errors

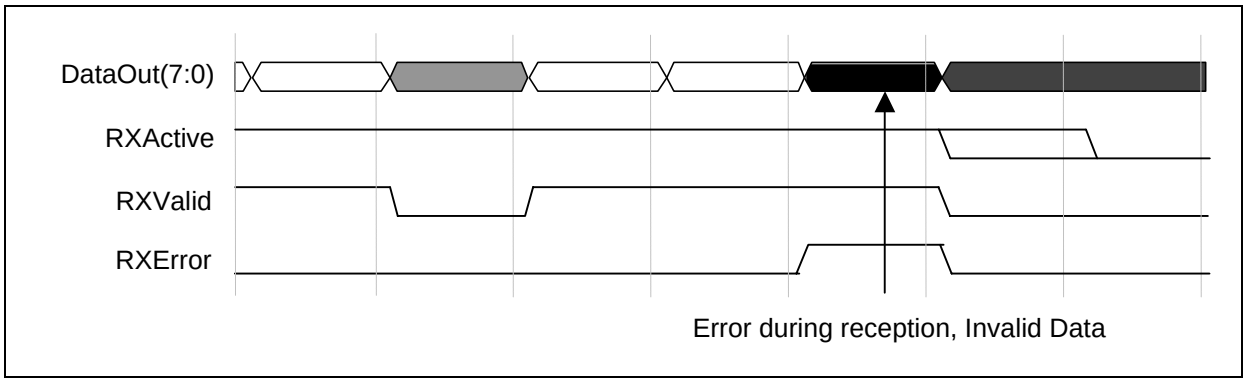


Figure 10: RXError Timing diagram

5.8.1.1 Bit Suff Error Reporting

Consider the case where an error occurs in the middle of a packet. Depending on the UTM implementation, **RXActive** may be negated at the same time as **RXValid** and **RXError**, or later. For instance, if a bitstuff error occurs in the middle of a FS receive packet it will generate a receive error, however packet data is still on the bus so **RXActive** will not be negated until an Idle state is detected on the bus.

By definition, a "bit stuff error" during a HS packet is automatically interpreted as an EOP. In this case **RXError** is not asserted. However if the bit stuff error was a true bit stuff error vs. an EOP-forced bit stuff error, then there will continue to be packet data on the bus and an EOP-forced bit stuff error will occur at the end of the packet. The SIE should know that some problem occurred during the packet because the CRC for the packet will be incorrect, however it is conceivable that the last 2 bytes of data received before the true bit stuff error match the correct CRC and an error is not detected by CRC. To maximize the robustness of the error detection, the UTM should flag a framing error for any bit stuff error that does not occur at the expected byte boundary.

The USB 2.0 specification does not specifically state that the receive state machine must detect an Idle state before beginning the search for a SYNC pattern, however one should consider it a requirement for a robust design. In the case of a true bit stuff error in the middle of a long packet, this will prevent the possibility of interpreting data as a SYNC pattern and incorrectly beginning the reception of one or more false packets.

There is always the possibility of multiple bit stuff errors occurring during a packet.

If a true bit stuff error occurred during the Data packet of an OUT transaction, then the SIE must not handshake thus allowing the transaction to timeout.

To prevent this collision and maintain the correct inter-packet delay, the Receive State Machine can be implemented in one of two ways:

1) If the Receive State Machine negates **RXActive** immediately, it must internally block **TXValid** to the Transmit State Machine until the USB is back to an Idle state and the minimum inter-packet delay, as defined by the USB 2.0 specification, has transpired.

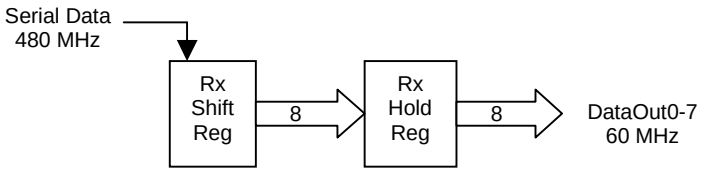
2) The Receive State Machine can hold **RXActive** asserted until an Idle state is detected on the bus. Thus, holding off the SIE until the bus is Idle. In this case the SIE is responsible for timing the inter-packet delay.

It is recommended that for HS packets, the internal "squelch" signal of the UTM be used to qualify the negation of **RXActive**.

Note: Figure 10 shows **RXValid** and **RXError** asserted at the same time. The state of **RXValid** is a function of data flow control, while **RXError** is asserted when an error is detected.

5.9 Rx Shift/Hold Registers

This module is responsible for converting serial data received from the USB to parallel data. This module consists of an 8-bit primary RX Shift Register for serial to parallel conversion and an 8-bit RX Hold Register used to buffer received data bytes and present them to the **DataOut** bus.



5.10 NRZI Encoder

This is a standard USB 1.X compliant serial NRZI encoder module, which can operate at full-speed or high-speed USB data rates.

5.11 Bitstuff Logic

In order to ensure adequate signal transitions, bit stuffing is employed when sending data on USB. A zero is inserted after every six consecutive ones in the data stream before the data is NRZI encoded, to enforce a transition in the NRZI data stream. Bit stuffing is enabled beginning with the SYNC Pattern and through the entire transmission. The data "one" that ends the SYNC Pattern is counted as the first one in a sequence.

In FS mode bit stuffing by the transmitter is always enforced, without exception. If required by the bit stuffing rules, a zero bit is inserted even after the last bit before the TXValid signal is negated.

After 8 bits are stuffed into the USB data stream **TXReady** is negated for one byte time to hold up the data stream on the **DataIn** bus. Figure 11 show the timing relationship between **TXReady** and **DataIn**.

See Figure 11 for an example of bit stuffing on USB.

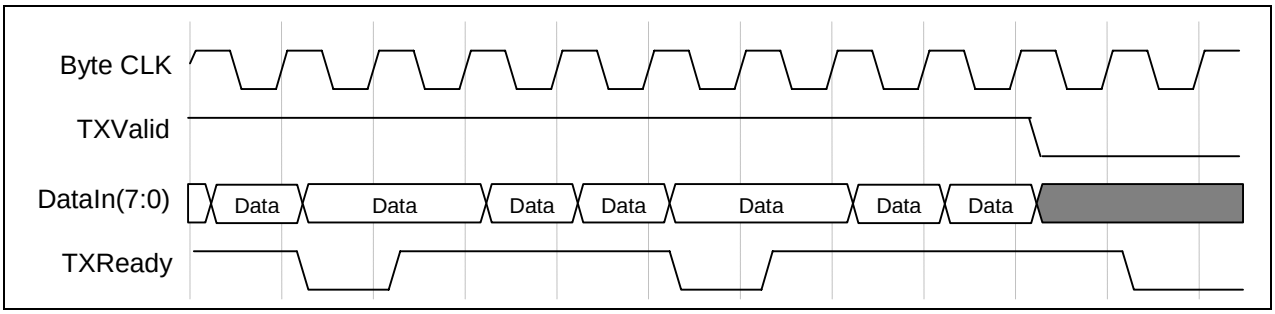
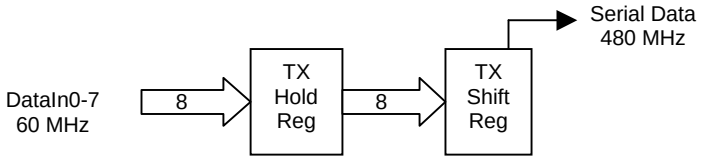


Figure 11: Transmit Timing delays due to Bit Stuffing

Note that the Byte CLK in Figure 11 is identical to the **CLK** signal in HS mode and **CLK**/40 in FS mode.

5.12 Tx Shift/Hold Register

This module is responsible for reading parallel data from the parallel application bus interface upon command and serializing for transmission over USB. This module consists of an 8-bit primary shift register for parallel/serial conversion and an 8-bit Hold register used to buffer the next data to serialize.



5.13 Transmit State Machine

The behavior of the Transmit State Machine is described below and illustrated in Figure 12.

The **Reset** signal forces the state machine into the *Reset* state which negates **TXReady**. When **Reset** is negated the transmit state machine will enter the *TX Wait* state.

In the *TX Wait* state, the transmit state machine looks for the assertion of **TXValid**. When **TXValid** is detected, the state machine will enter the *Send SYNC* state and begin transmission of the SYNC pattern.

When the transmitter is ready for the first byte of the packet (PID), it will enter the *TX Data Load* state, assert **TXReady** and load the TX Holding Register. The state machine may enter the *TX Data Wait* state while the SYNC pattern transmission is completed.

TXReady is used to throttle transmit data. The state machine will remain in the *TX Data Wait* state until the TX Data Holding register is available for more data. In the *TX Data Load* state, the state machine loads the Transmit Holding register. The state machine will remain in the *TX Data Load* state as long as the transmit state machine can empty the TX Holding Register before the next rising edge of **CLK**.

When **TXValid** is negated the transmit state machine enters the *Send EOP* state where it sends the EOP. While the EOP is being transmitted **TXReady** is negated and the state machine will remain in the *Send EOP* state. After the EOP is transmitted the Transmit State Machine returns to the *TX Wait* state, looking for more work.

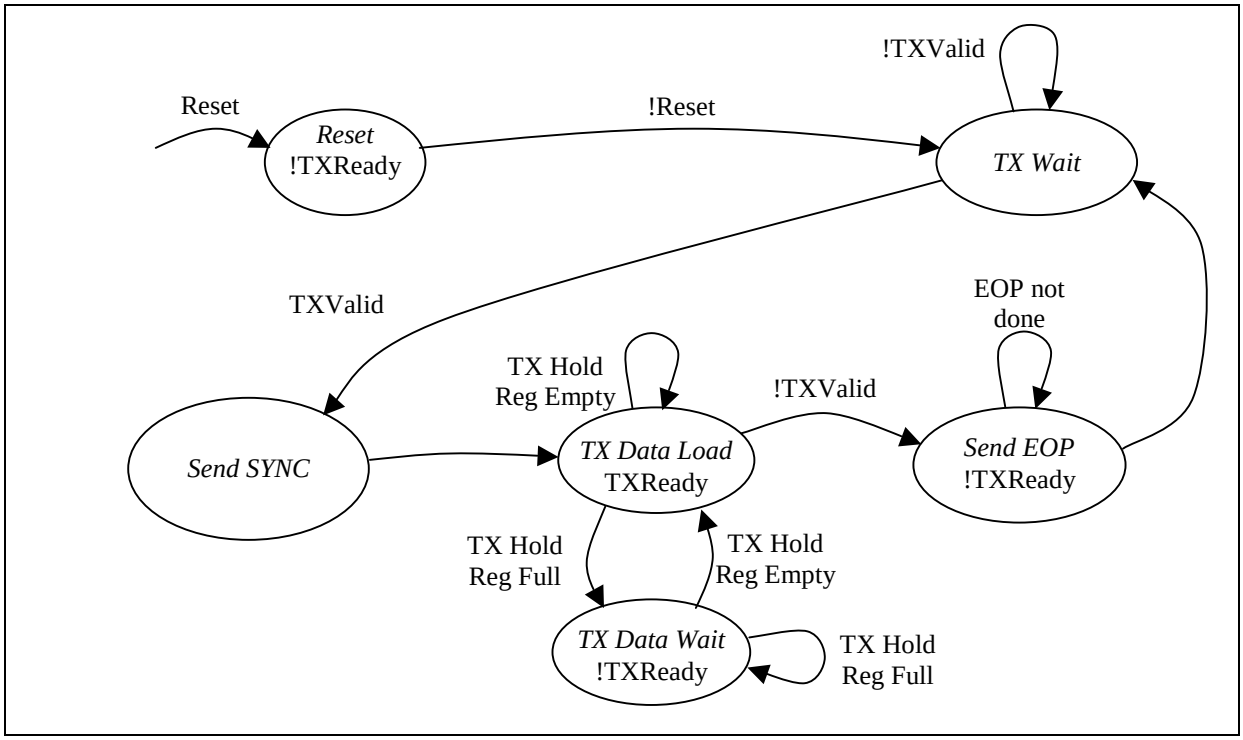


Figure 12: Transmit State Diagram

- **Transmit** must be asserted to enable any transmissions.
- The SIE asserts **TXValid** to begin a transmission.
- The SIE negates **TXValid** to end a transmission.

- After the SIE asserts **TXValid** it can assume that the transmission has started when it detects **TXReady** asserted.
- The SIE assumes that the UTM has consumed a data byte if **TXReady** and **TXValid** are asserted.
- The SIE must have valid packet information (PID) asserted on the **DataIn** bus coincident with the assertion of **TXValid**. Depending on the UTM implementation, **TXReady** may be asserted by the Transmit State Machine as soon as one CLK after the assertion of **TXValid**.
- **TXValid** and **TXReady** are sampled on the rising edge of **CLK**.
- The Transmit State Machine does NOT automatically generate Packet ID's (PIDs) or CRC. When transmitting, the SIE is always expected to present a PID as the first byte of the data stream and if appropriate , CRC as the last bytes of the data stream.
- The SIE must use **LineState** to verify a Bus Idle condition before asserting **TXValid** in the *TX Wait* state.
- The state of **TXReady** in the *TX Wait* and *Send SYNC* states is undefined. An MTU implementation may prepare for the next transmission immediately after the *Send EOP* state and assert **TXReady** in the *TX Wait* state. An MTU implementation may also assert **TXReady** in the *Send SYNC* state. The first assertion of TXReady is Macrocell implementation dependent. The SIE must prepare **DataIn** for the first byte to be transmitted before asserting **TXValid**.

Figure 13 shows the timing relationship between **TXValid**, **DataIn**, **TXReady** and the transmitted data (**DP**/**DM**).

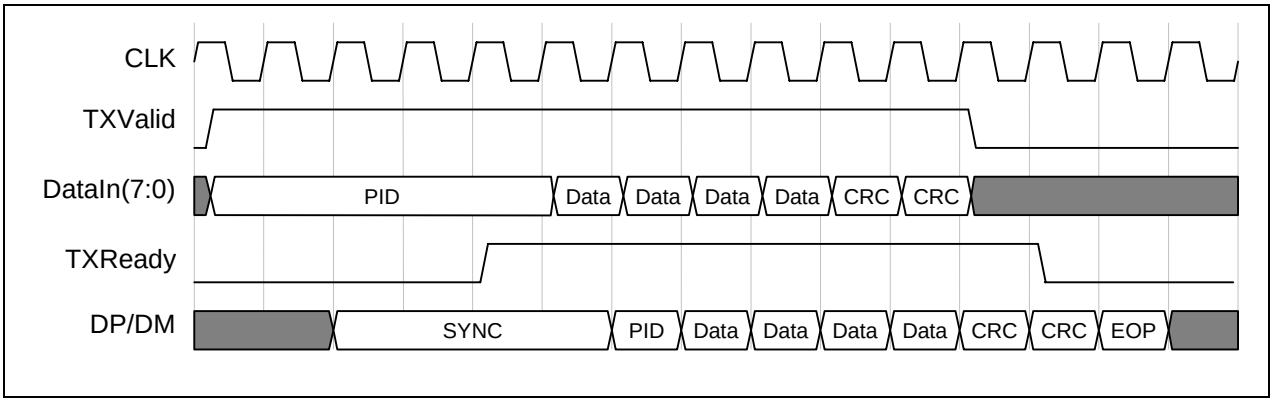


Figure 13: Transmit Timing for a Data packet

The SIE negates **TXValid** to complete a packet. Once negated, the Transmit State Machine will never re-assert **TXReady** until after the EOP has been loaded into the Transmit Shift Register. Note that the UTM Transmit State Machine can be ready to start another packet immediately, however the SIE must conform to the minimum inter-packet delays identified in the USB 2.0 Specification.

5.13.1 Transmit Error Reporting

In FS mode if the data transmitter is unable, because of problems such as a buffer underrun condition, to transmit the identical amount of data as was in the original data packet, it must terminate the transaction by generating a bit stuffing violation, followed by an EOP. To accomplish this the SIE must switch the **OpMode** to "Disable Bit Stuffing and NRZI Encoding" and load 0's into the **DataIn** lines for at least one byte time before negating **TXValid**.

In HS mode, if an error condition occurs during transmission, the current transmit stream must be terminated by the transmission of a complemented version of the CRC, followed by an EOP. In this case the SIE will be responsible for presenting the complemented CRC to the **DataIn** lines before negating **TXValid**.

In either mode the negation of **TXValid** will cause the UTM to terminate the packet with the appropriate EOP.

5.14 USB Full Speed XCVR

The FS receiver includes the logic necessary to send and receive the FS data on USB, as well as support the Reset, Suspend and Resume detection functions as described in the 1.1 spec. See the Reset Detection, Suspend Detection and Resume Detection sections below for more detail.

5.14.1 Transmit Driver

When enabled, data from the transmit data path will be driven onto the **DP**/**DM** signal lines. The FS transmit driver is active only when Transmit is asserted, the **XcvrSelect** input is in FS transceiver enabled mode and the Transmit State Machine has data to send.

5.14.2 Receive Buffer

When enabled, received FS data will be multiplexed through the receive data path to the receive shift and hold registers. The USB 2.0 receive buffer is active only when the **XcvrSelect** input is in FS transceiver enabled mode.

5.15 USB2.0 XCVR

This block contains the low-level analog circuitry required to physically interface USB 2.0 signaling to the USB **DP**/**DM** signal lines. It contains the HS Differential Data transmitter and receiver, performs HS Transmission Envelope Detection and disables the 1.5K Ohm FS indicator resistor.

5.15.1 Transmit Driver

When enabled, data from the transmit data path will be driven on to the **DP**/**DM** signal lines. The HS transmit driver is active only when Transmit is asserted, the **XcvrSelect** input is in HS transceiver enabled mode and the Transmit State Machine has data to send.

5.15.2 Receive Buffer

When enabled, received HS data will be multiplexed through the receive data path to the receive shift and hold registers. The USB 2.0 receive buffer is active only when the **XcvrSelect** input is in HS transceiver enabled mode.

5.15.3 Other Components of Transceiver

5.15.3.1 Transmission Envelope Detector

The quiescent state of a HS link is for the **DP** and **DM** lines to be balanced near ground with the differential receivers listening for Start of Packet. The Transmission Envelope Detector is evoked to prevent spurious signals (e.g., noise, crosstalk, or oscillation) from triggering the Start of Packet detection process (to "squelch" the receiver).

This envelope detector is used to disable or "squelch" the HS receiver when the amplitude of the differential signal falls below the minimum required level for data reception, preventing noise from propagating through the receive logic.

5.15.3.2 Full-Speed Indicator Control

In full-speed mode, a 1.5K Ohm pull-up on the **DP** signal line is used to indicate to an upstream port that a full-speed device is attached. In high-speed mode this resistor would introduce about a 50 mV error into the received signal so it must be removed. When it is enabled, the HS XCVR contains the circuitry to electrically detach the full-speed indicator resistor from the **DP** signal line.

5.16 Operational Modes

The **OpMode** signals are capable of inhibiting normal operation of the transceiver and evoking special test modes. These modes take effect immediately and take precedence over any pending data operations. The transmission data rate when in **OpMode** depends on the state of the **XcvrSelect** input. There are 3 test modes:

- Normal Operation (0)
- Non-Driving (1)
- Disable Bit Stuffing and NRZI encoding (2)

Mode 0 allows the transceiver to operate with normal USB data decoding and encoding.

Mode 1 allows the transceiver logic to support a soft disconnect feature which tri-states both the HS and FS transmitters, and removes any termination from the USB making it appear to an upstream port that the device has been disconnected from the bus.

Mode 2 disables Bit Stuff and NRZI encoding logic so 1's loaded from the **DataIn** bus becomes 'J's on the **DP**/**DM** lines and 0's become 'K's. Note that this mode affects the automatic SYNC Pattern and EOP generation by **TXValid**. It is disabled so that Chirps can be generated on the USB. The operation of the receiver is undefined.

Note that the **OPMode** signals are normally changed only when the transmitter and the receiver are quiescent, i.e. when entering a test mode or for a device initiated resume, the **OPMode** is set and then **TXValid** is asserted. In this case, the SYNC pattern and EOP are not transmitted by the UTM.

The only exception is when the **OPMode** signals are set to mode 2 while **TXValid** is asserted (the transceiver is transmitting a packet), in order to flag a transmission error. See section 5.13.1 for more information. In this case, the SYNC pattern has already been transmitted by the UTM so upon the negation of **TXValid** the EOP must also be transmitted to properly terminate the packet.

Changing the **OPMode** signals under all other conditions, while the transceiver is receiving or transmitting data will generate undefined results.

5.16.1 USB 2.0 Test Mode Generation

Note that the USB Specification defines additional test modes. These are accomplished with the following techniques.

To force an SE0 State on the bus the UTM is placed in test mode 0 (Normal Operation) and no data is transmitted. This results in an HS Idle mode on the bus, which is SE0.

To force a 'J' State on the bus the UTM is placed in test mode 2 (Disable Bit Stuffing and NRZI encoding) and all 1's are loaded into the Transmit Data Holding register.

To force a 'K' State on the bus the UTM is placed in test mode 2 (Disable Bit Stuffing and NRZI encoding) and all 0's are loaded into the Transmit Data Holding register.

To generate a Test Packet on the bus the UTM is placed test mode 0 (Normal Operation) and all the Test Packet data (as defined in Chapter 5 of the USB 2.0 specification) is loaded into the transmit data register.

Table 8: USB 2.0 Test Mode to Macrocell Mapping

USB 2.0 Test Modes	Macrocell Setup		
	Operational Mode	Transmitted data	XcvrSelect & TermSelect
SE0_NAK	Normal	No transmit	HS
J	Disable	All '1's	HS
K	Disable	All '0's	HS
Test_Packet	Normal	Test Packet data	HS

Note that the "Test Force_Enable" mode described in the USB 2.0 Specification does not apply to upstream ports.

5.17 Speed Selection

The **XcvrSelect** and **TermSelect** signals determine whether the device is in HS or FS mode, enabling the respective transceiver and termination. The HS Detection Handshake protocol requires independent control of transceivers and terminations, where the device enables FS terminations but is required to drive and listen to the bus with the HS transceiver. In all other cases the state of the **XcvrSelect** and **TermSelect** signals are identical.

FS Only and LS Only UTM implementations do not require the **XcvrSelect** and **TermSelect** speed selection signals since there is no alternate speed to switch to.

5.18 Bi-directional 8-bit Interface

An option for the UTMI is to attach the block described below to the 8-bit interface. This option provides a 8-bit bi-directional bus, minimizing the connection count. This bi-directional option is typically used by a discrete implementation of a UTMI compliant transceiver that attaches to an ASIC.

When this option is applied, 8 data lines will be presented by the transceiver, where **Data0-7** is a bi-directional data bus.

If **TXValid** is asserted (1) then the signals **Data0-7** accept transmit data from the SIE. If **TXValid** is negated (0) then the signals **Data0-7** present received data to the SIE.

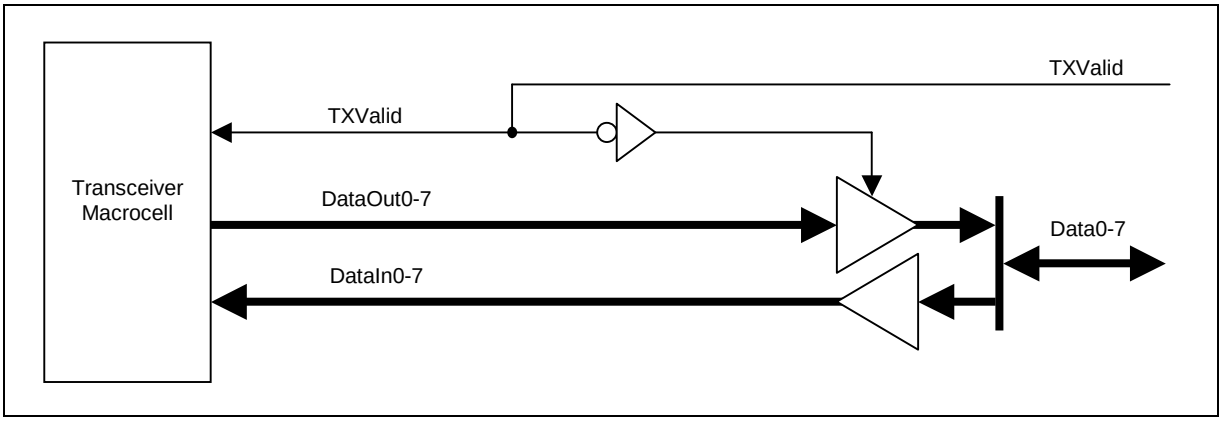


Figure 14: 8-Bit Bi-directional Data Bus Interface

5.19 16-Bit Interface

To support slower SIE logic implementations the UTMI also supports a 16-bit data interface. The **DataBus16_8** signal identifies whether data is handled as 16 or 8 bit. When 16-bit operation is enabled the changes to the 8-bit implementation are:

- **CLK** will run at half the rate of the equivalent 8-bit implementation (30 MHz). The 16-bit interface is only defined for HS/FS transceiver implementations (not for FS Only or LS Only).
- Additional signals (**RXValidH** and **TXValidH**) are provided to identify whether the high byte of the respective 16-bit data word is valid.
- Additional data lines are provided (**DataIn** 8-15 and **DataOut** 8-15).
- The **TXReady** signal will drop low for one clock time each time 16 stuffed bits are accumulated vs. after the accumulation of 8 stuffed bits with the 8-bit interface.
- The **RXValid** and **RXValidH** signals will simultaneously drop low for one clock time each time 16 stuffed bits are accumulated vs. **RXValid** dropping low after the accumulation of 8 stuffed bits with the 8-bit interface.

Note that **DataBus16_8** controls data bus width and the frequency of CLK. It is only sampled by the macrocell on the negation of Reset.

Note that the other sections of this document assume that the Transceiver is operating with an 8-bit interface. The timings need to be adjusted appropriately for operation using 16-bit interface.

5.19.1 16-Bit Transmit Timing

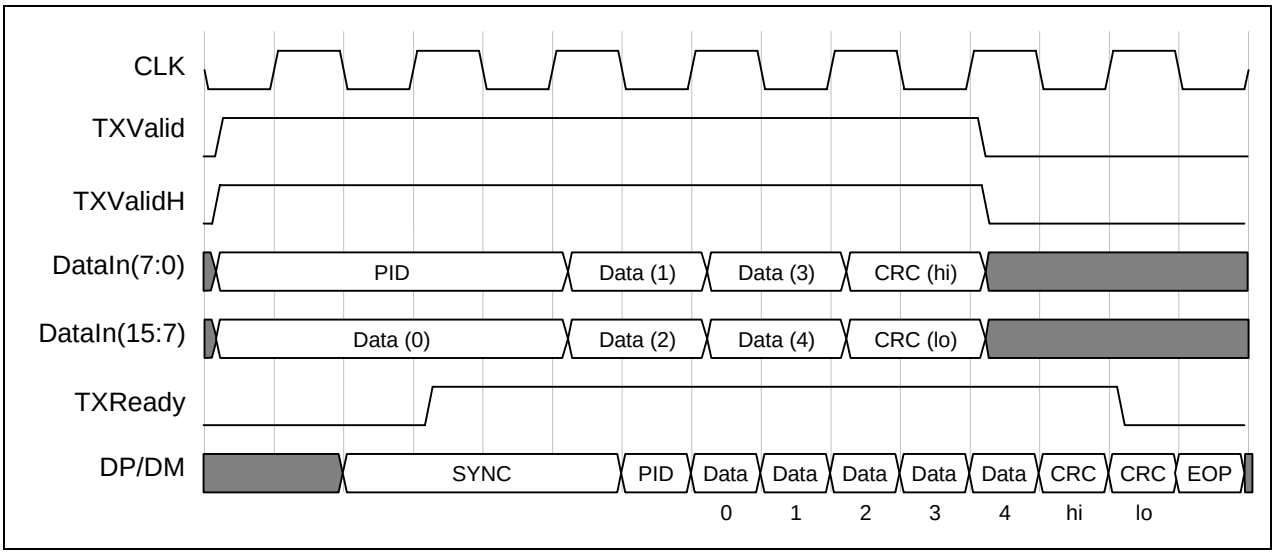


Figure 15: Transmit Timing for 16-bit Data, Even Byte Count

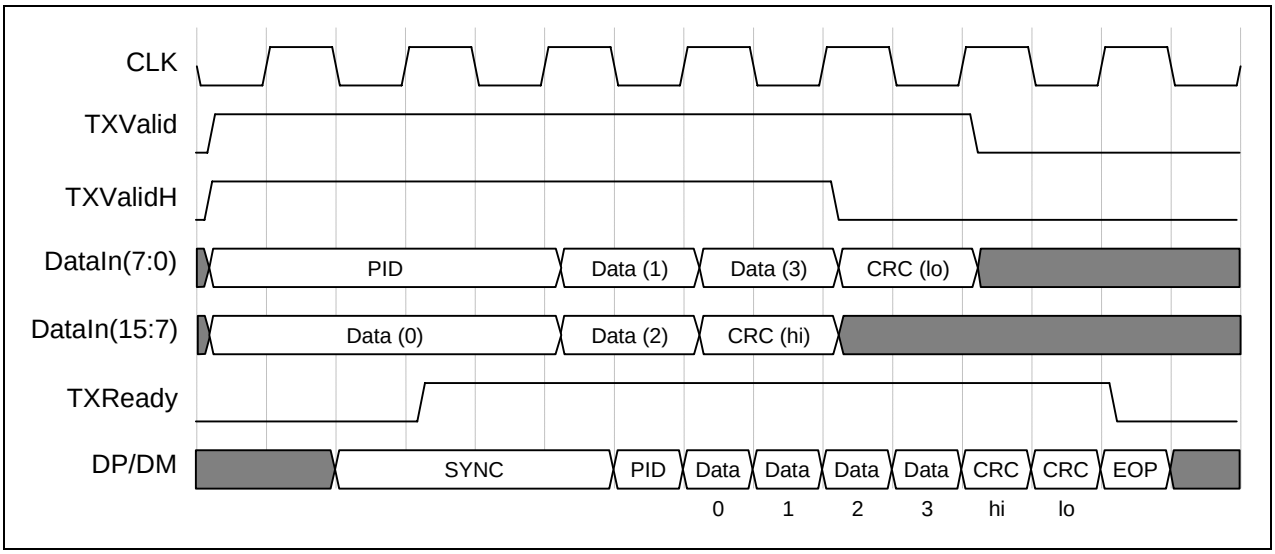


Figure 16: Transmit Timing for 16-bit Data, Odd Byte Count

Note: The CRC must be transmitted MSb first, however the UTM does not distinguish CRC from data on the **DataIn** byte lanes, so the SIE is responsible for placing the CRC bytes on the correct **DataIn** byte lane. In Figure 15 and Figure 16 the high and low byte of the CRC16 must be swapped between the **DataIn** byte lanes depending in whether the byte count is even or odd.

5.19.2 16-Bit Receive Timing

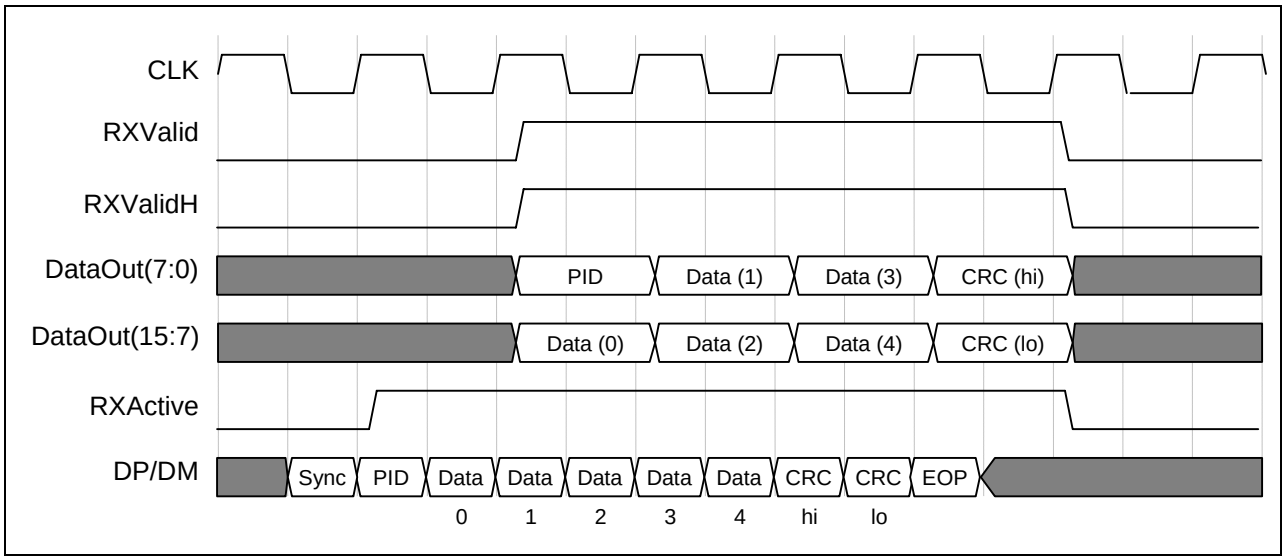


Figure 17: Receive Timing for 16-bit Data, Even Byte Count

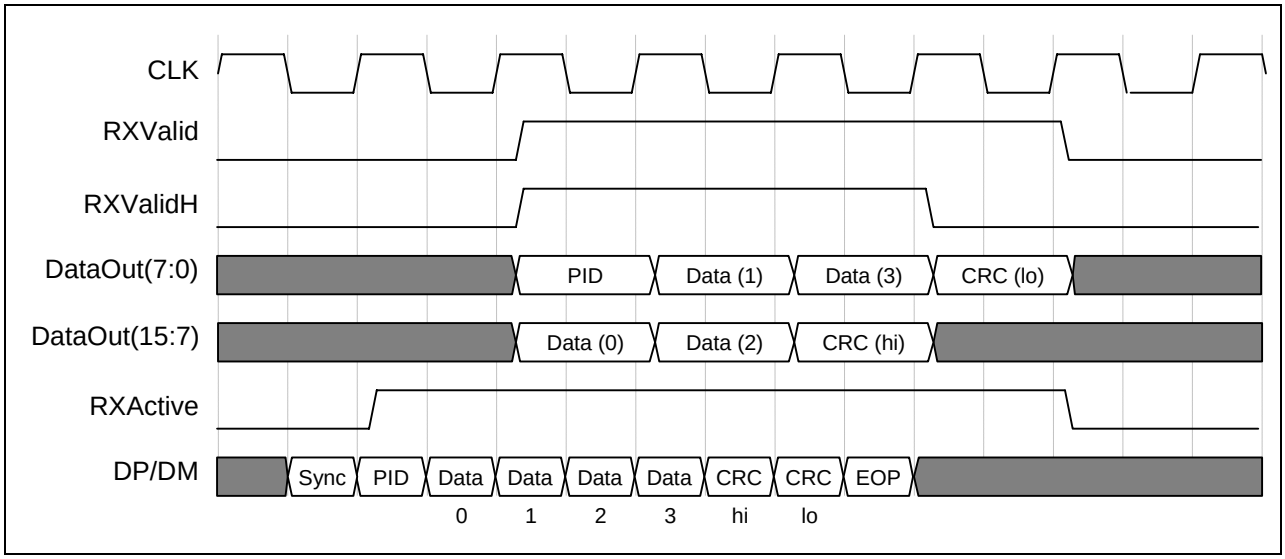


Figure 18: Receive Timing for 16-bit Data, Odd Byte Count

Note: The CRC must be transmitted MSb first, however the UTM does not distinguish CRC from data on the **DataOut** byte lanes, so the SIE is responsible for receiving the CRC bytes on the correct **DataIn** byte lane. In Figure 17and Figure 18 the high and low byte of the CRC16 are swapped between the **DataIn** byte lanes depending in whether the byte count is even or odd.

5.20 Bi-directional 16-bit Interface

An option for the UTMI is to attach the block described below to the 16-bit interface. This option provides a 16-bit bi-directional bus, minimizing the connection count. This bi-directional option is typically used by a discrete implementation of a UTMI compliant transceiver that attaches to an FPGA or a RAM based gate array.

When this option is applied, 16 data lines will be presented by the transceiver. When **DataBus16_8** is low (0), the 16 data signals act as two 8-bit data buses, where **Data0-7** is an input bus for receive data, identical to the **DataIn0-7** bus and **Data8-15** is an output bus for transmit data, identical to the **DataOut0-7** bus.

When **DataBus16_8** is high (1), transceiver is placed in 16-bit mode, where **Data0-15** is a bi-directional data bus. If **TXValid** is asserted (1) then the signals **Data0-15** accept transmit data from the SIE. If **TXValid** is negated (0) then the signals **Data0-15** present received data to the SIE.

To additionally reduce the signal count the **TXValidH** and **RXValidH** signals are multiplexed onto the **ValidH** signal. Note that the **ValidH** signal is undefined if **DataBus16_8** = 0.

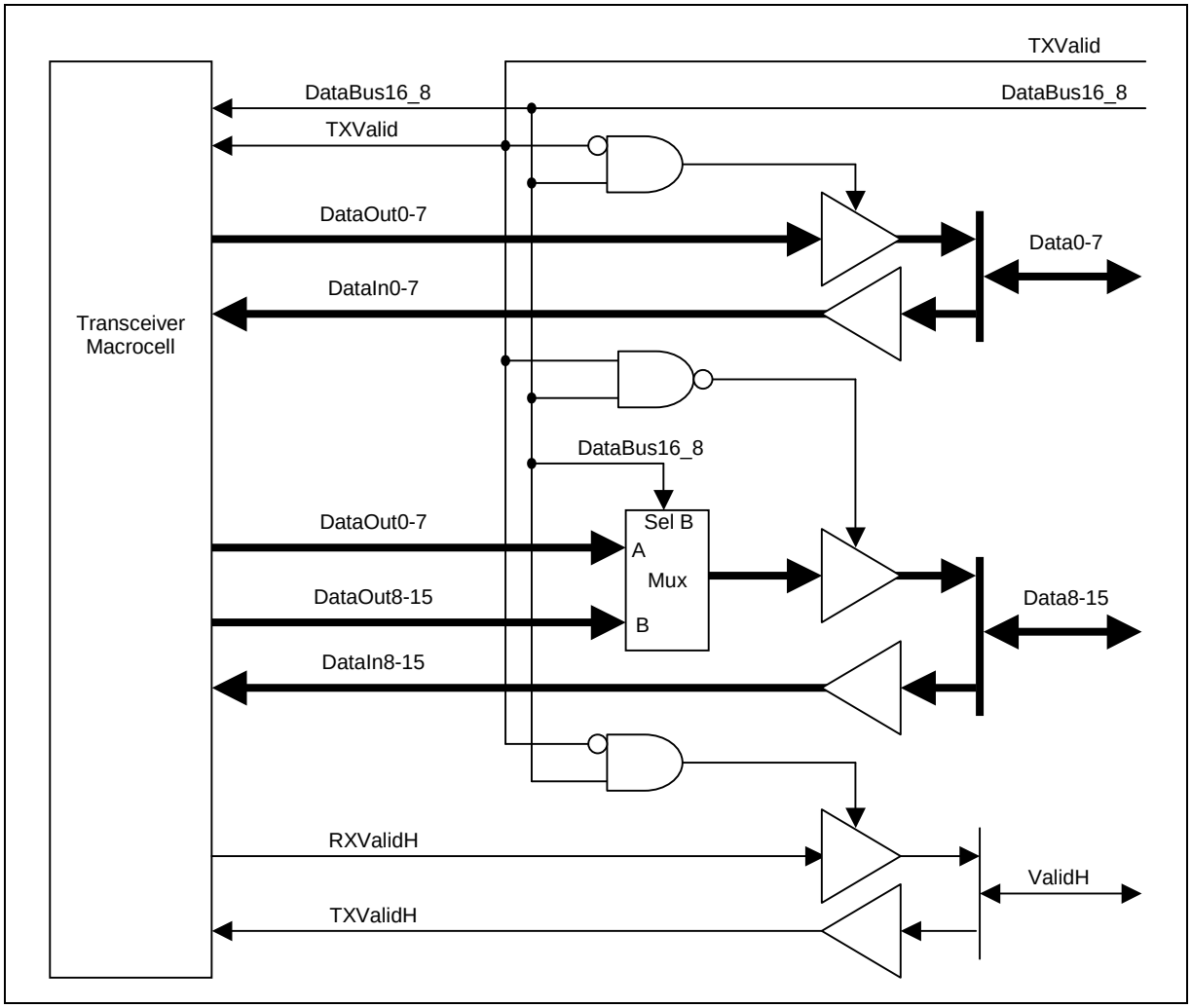


Figure 19: 16-bit Bi-directional Data Bus Interface

5.21 Vendor Controls

The UTMI provides for optional vendor-defined error, status and control information to be presented through a standard interface. The vendor control interface consists of two registers. All the registers are synchronous with **CLK**. SIEs are required to make these registers accessible to system software, so that detailed diagnostic and error analysis can be performed. The 2 registers are:

1) An 8-bit "Vendor Status" register that is an output of the macrocell. A macrocell vendor can use this port to present internal transceiver information to the SIE. Typical examples are: the internal macrocell signals like "CLK Usable" or "Squelch", error codes, etc.

2) A 4-bit "Vendor Control" register that is an input to the macrocell. A macrocell vendor can use this register to enable special test modes, like digital or analog loopback, select the information that is presented on the Vendor Status port, etc. Macrocells will use Reset to initialize this register to values that allow normal operation. This way if the SIE does not drive these signals the macrocell will still be fully functional.

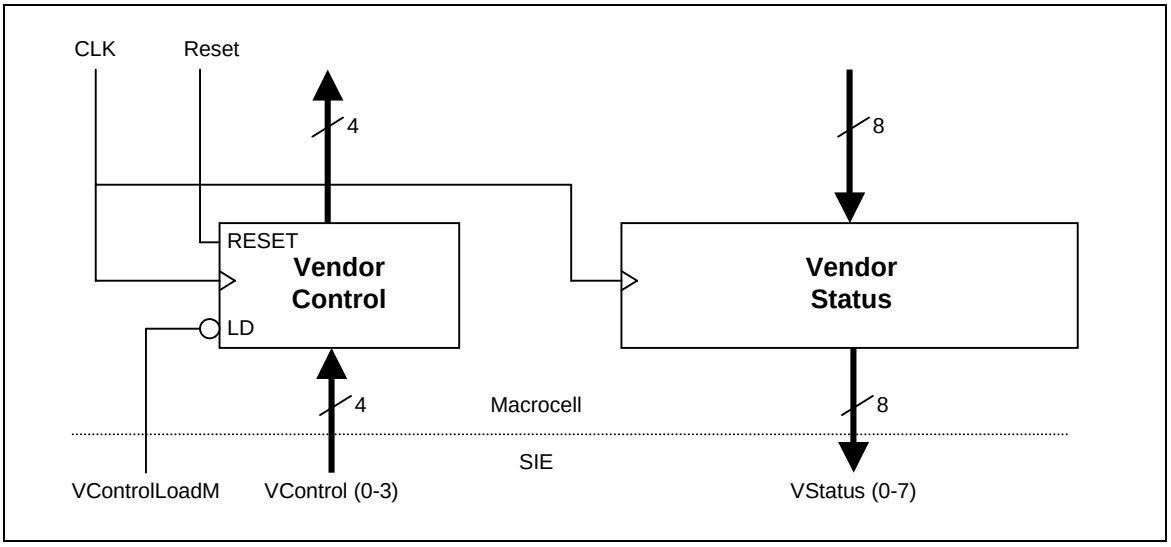


Figure 20: Vendor Control Register Block Diagram

5.22 Other Functions

The **LineState** signals are used for many functions. The **LineState** signals reflect the current state of the **DP**/**DM** signal lines. The thresholds used by the **LineState** to determine the state of **DP**/**DM** depend on the value of **XcvrSelect**. **LineState** uses HS thresholds when the HS transceiver is enabled (**XcvrSelect** = 0) and FS thresholds when the FS transceiver is enabled (**XcvrSelect** = 1). FS Only and LS Only implementations always use FS thresholds.

The following sections make a distinction between "soft" SE0 and "driven" SE0. Soft SE0 is the bus signaling that results from the **DP** and **DM** signal lines being pulled down exclusively by the 15K pull-down resistors (Rpd). Driven SE0 is the result of generating a SE0 condition by enabling the FS Transmitter. In this case the **DP** and **DM** signal lines are being pulled down by the 45 Ohm serial (Rs) termination resistors.

Note: Rpd and Rs are defined in Figure 7.1 of the USB 2.0 Specification.

5.22.1 SE0 handling

For low-speed and full-speed operation, Idle is a J state on the bus and SE0 is used as part of the EOP or to indicate reset. When asserted in an EOP, SE0 is never asserted on the bus for more than 2 low-speed bit times (1.3 µs). The assertion of SE0 for more than 2.5 µs is interpreted as a reset by device operating in low-speed or full-speed.

For high-speed operation, Idle is an SE0 state on the bus. SE0 is also used to reset a high-speed device. A high-speed device cannot use the 2.5 µs assertion of SE0 (as defined for FS operation) to indicate reset since the bus is often in this state between packets. If no bus activity (Idle) is detected for more than 3 ms. a high-speed device must determine whether the downstream port is signaling a suspend or a reset. Sections 5.22.1.1 and 5.22.1.2 detail how this determination is made. If a reset is signaled the high-speed device will then initiate the HS Detection handshake protocol, as defined in section 5.22.2.

Note that the initial assertion of SE0 on the bus is referred to in the core specification and this specification as "HS Reset T0" (see Section 7.1.7.3 "Reset Signaling" of the USB 2.0 Specification).

5.22.1.1 Suspend Detection

If a HS device detects SE0 asserted on the bus for more than 3 ms. (T1) its UTM is placed in FS mode (**XcvrSelect** and **TermSelect** = 1). This enables the FS pull-up on the **DP** line, asserting a continuous FS 'J' state on the bus. The SIE must then check the **LineState** signals for an 'J' State condition. If 'J' State condition is asserted at time T2, then the upstream port is asserting a Soft SE0 and the USB is in a 'J' state indicating a suspend condition. By time T4 the device must be fully suspended.

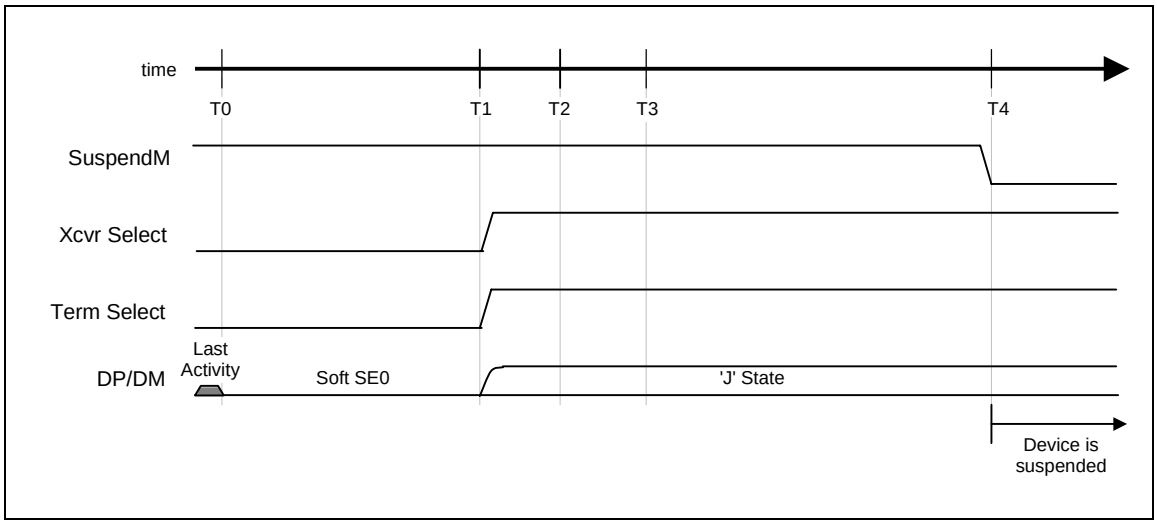


Figure 21: Suspend Timing Behavior (HS Mode)

Table 9: Suspend Timing Values (HS Mode)

Timing Parameter	Description	Value
HS Reset T0	End of last bus activity, signaling either a reset or a suspend.	0 (reference)
T1	The time at which the device must place itself in FS mode after bus activity stops.	HS Reset T0 + 3.0 ms < T1 {T_{WTREV}} < HS Reset T0 + 3.125 ms
T2	SIE samples LineState. If LineState = 'J', then the initial SE0 on the bus (T0 - T1) had been due to a Suspend state and the SIE remains in HS mode.	T1 + 100 µs < T2 {$T_{WTWRSTHS}$} < T1 + 875 µs
T3	The earliest time where a device can issue Resume signaling.	HS Reset T0 + 5ms {T_{WTRSM}}
T4	The latest time that a device must actually be suspended, drawing no more than the suspend current from the bus.	HS Reset T0 + 10ms {T_{2SUSP}}

Note: USB 2.0 core specification timing values are referenced in curly braces {}.

5.22.1.2 Reset Detection

If a device in HS mode detects bus inactivity for more than 3 ms. (T1) its UTM is placed in FS mode (**XcvrSelect** = 1 and **TermSelect** = 1). This enables the FS pull-up on the **DP** line to attempt to assert a continuous FS 'J' state on the bus (dotted line in Figure 22). The SIE must then check the **LineState** signals for the SE0 condition. If SE0 is asserted at time T2, then the upstream port is forcing the reset state to the device (i.e. Driven SE0). The device will then initiate the HS Detection Handshake protocol.

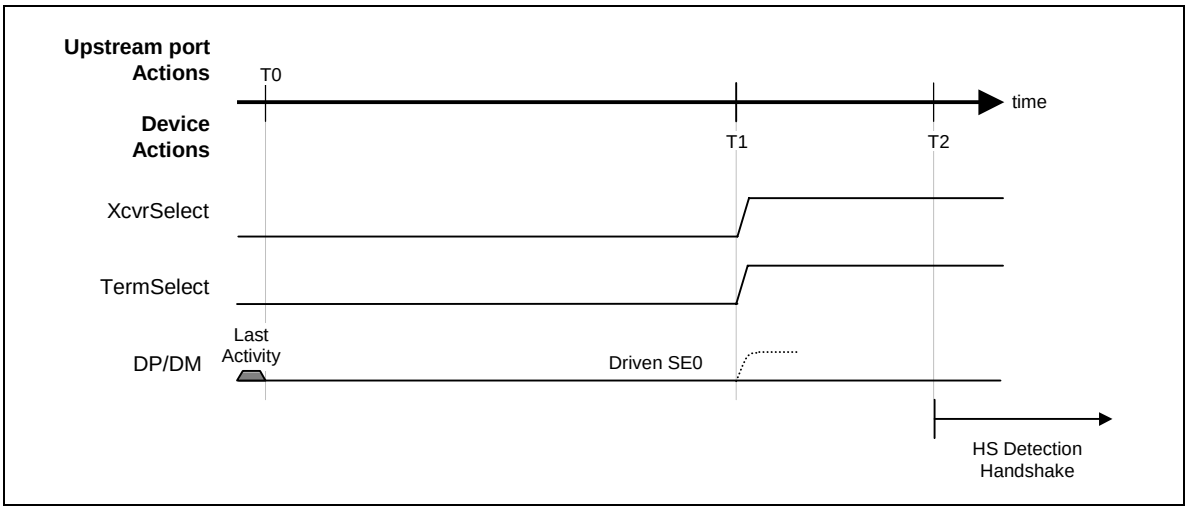


Figure 22: Reset Timing Behavior (HS Mode)

Table 10: Reset Timing Values (HS Mode)

Timing Parameter	Description	Value
HS Reset T0	Bus activity ceases, signaling either a reset or a suspend.	0 (reference)
T1	Earliest time at which the device may place itself in FS mode after bus activity stops.	HS Reset T0 + 3.0 ms < T1 {T_{WTREV}} < HS Reset T0 + 3.125 ms
T2	SIE samples LineState. If LineState = SE0, then the SE0 on the bus is due to a Reset state. The device now enters the HS Detection Handshake protocol.	T1 + 100 µs < T2 {$T_{WTWRSTHS}$} < T1 + 875 µs

5.22.2 HS Detection Handshake

The High-Speed Detection Handshake process is entered from one of three states: suspend, active FS or active HS. The downstream port asserting an SE0 state on the bus initiates the HS Detection Handshake. Depending on the initial state, an SE0 condition can be asserted from 0 to 4 ms before initiating the HS Detection Handshake. These states are described in section 7.1.7.3 of the USB 2 specification (State 3 of the "Reset Protocol for HS capable hubs and devices").

There are three ways in which a device may enter the HS Handshake Detection process:

1) If the device is suspended and it detects an SE0 state on the bus it may immediately enter the HS handshake detection process.

2) If the device is in FS mode and an SE0 state is detected for more than 2.5 µs. it may enter the HS handshake detection process.

3) If the device is in HS mode and an SE0 state is detected for more than 3.0 ms. it may enter the HS handshake detection process. In HS mode, a device must first determine whether the SE0 state is signaling a suspend or a reset condition. To do this the device reverts to FS mode by placing **XcvrSelect** and **TermSelect** into FS mode. The device must not wait more than 3.125 ms before the reversion to FS mode. After reverting to FS mode, no less than 100 µs. and no more than 875 µs. later the SIE must check the **LineState** signals. If a J state is detected the device will enter a suspend state. If an SE0 state is detected, then the device will enter the HS Handshake detection process.

In each case, the assertion of the SE0 state on the bus initiates the reset interval (referred to in this section as "HS Reset T0"). The minimum reset interval is 10 ms. Depending on the previous mode that the bus was in, the delay between the initial assertion of the SE0 state (HS Reset T0) and entering the HS Handshake detection process (T0 in Table 11, Table 12, and

Table 13) can be from 0 to 4 ms.

This transceiver design pushes as much of the responsibility for timing events on to the SIE as possible, and the SIE requires a stable CLK signal to perform accurate timing. In cases 2 and 3 above CLK has been running and is stable, however in case 1 the UTM is reset from a suspend state, and the internal oscillator and clocks of the transceiver are assumed to be powered down. A device has up to 6 ms after the release of SuspendM (HS Reset T0) to assert a minimum of a 1 ms Chirp K. To meet these timing constraints and provide reasonably relaxed clock stability requirements for the transceiver, several requirements are placed on the transceiver clock generator. See section 5.1 for details. Given these constraints the SIE can reliably generate a 1ms Chirp K.

5.22.2.1 FS Downstream Facing Port

This is an example of the UTM behavior when the downstream facing port that it is attached to does not support HS operation.

Upon entering the HS Detection process (T0) **XcvrSelect** and **TermSelect** are in FS mode. The D+ pull-up is asserted and the HS terminations are disabled. The SIE then sets **OpMode** to *Disable Bit Stuffing and NRZI encoding*, and begins the transmission of all 0's data, which asserts a HS K (chirp) on the bus (T1). The device chirp must last at least 1.0ms and must end no later than 7.0ms after HS Reset T0. At time T1 the SIE sets **XcvrSelect** to HS mode and begins listening for a chirp sequence from the downstream port.

If the downstream port is a not HS capable, then the HS K asserted by the device is ignored and the alternating sequence of HS Chirp K's and J's is not generated. If the downstream chirps are not detected (T4) the device will enter FS mode by returning **XcvrSelect** to FS mode.

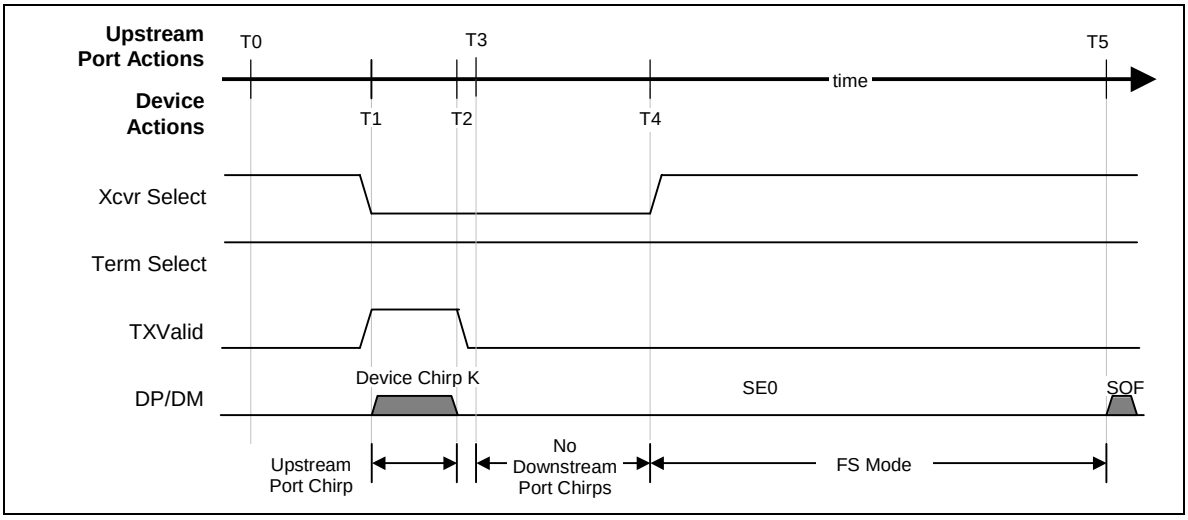


Figure 23: HS Detection Handshake Timing Behavior (FS Mode)

Table 11: HS Detection Handshake Timing Values (FS Mode)

Timing Parameter	Description	Value
T0	HS Handshake begins. D+ pull-up enabled, HS terminations disabled.	0 (reference)
T1	Device enables HS Transceiver and asserts Chirp K on the bus.	T0 < T1 < HS Reset T0 + 6.0 ms
T2	Device removes Chirp K from the bus. 1 ms minimum width.	T1 + 1.0 ms {T_{UCH}} < T2 < HS Reset T0 + 7.0 ms {T_{UCHEND}}
T3	Earliest time when downstream port may assert Chirp K on the bus.	T2 < T3 < T2+100 µs {T_{WTDCH}}
T4	Downstream port chirp not detected by the device. Device reverts to FS default state and waits for end of reset.	T2 + 1.0 ms < T4 {T_{WTFS}} < T2 + 2.5 ms
T5	Earliest time at which downstream port may end reset	HS Reset T0 + 10 ms {T_{DRST} (Min)}

Note: T0 may occur to 4 ms after HS Reset T0.
Note: The SIE must assert the Chirp K for 66000 **CLK** cycles to ensure a 1ms minimum duration.

5.22.2.2 HS Downstream Facing Port

This is an example of the UTM behavior when the downstream facing port that it is attached to supports HS operation.

Upon entering the HS Detection process (T0) **XcvrSelect** and **TermSelect** are in FS mode. The D+ pull-up is asserted and the HS terminations are disabled. The SIE then sets **OpMode** to *Disable Bit Stuffing and NRZI encoding*, and begins the transmission of all 0's data, which asserts a HS K (chirp) on the bus (T1). The device chirp must last at least 1.0ms, and must end no later than 7.0ms after HS Reset T0. At time T1 the SIE sets **XcvrSelect** to HS mode and asserts a Chirp K on the bus. After the Chirp K is complete the SIE begins listening for the chirp sequence from the downstream port.

If the downstream port is HS capable then it will begin generating an alternating sequence of Chirp K's and Chirp J's (T3) after the termination of the chirp from the device (T2). After the device sees the valid downstream port chirp sequence Chirp K-J-K-J-K-J (T6), it will enter HS mode by setting **TermSelect** to HS mode (T7). Figure 24 provides a state diagram for Chirp K-J-K-J-K-J validation. Prior to the end of reset (T9) the upstream port must terminate the sequence of Chirp K's and Chirp J's (T8) and assert SE0 (T8-T9). Note that the sequence of Chirp K's and Chirp J's constitutes bus activity.

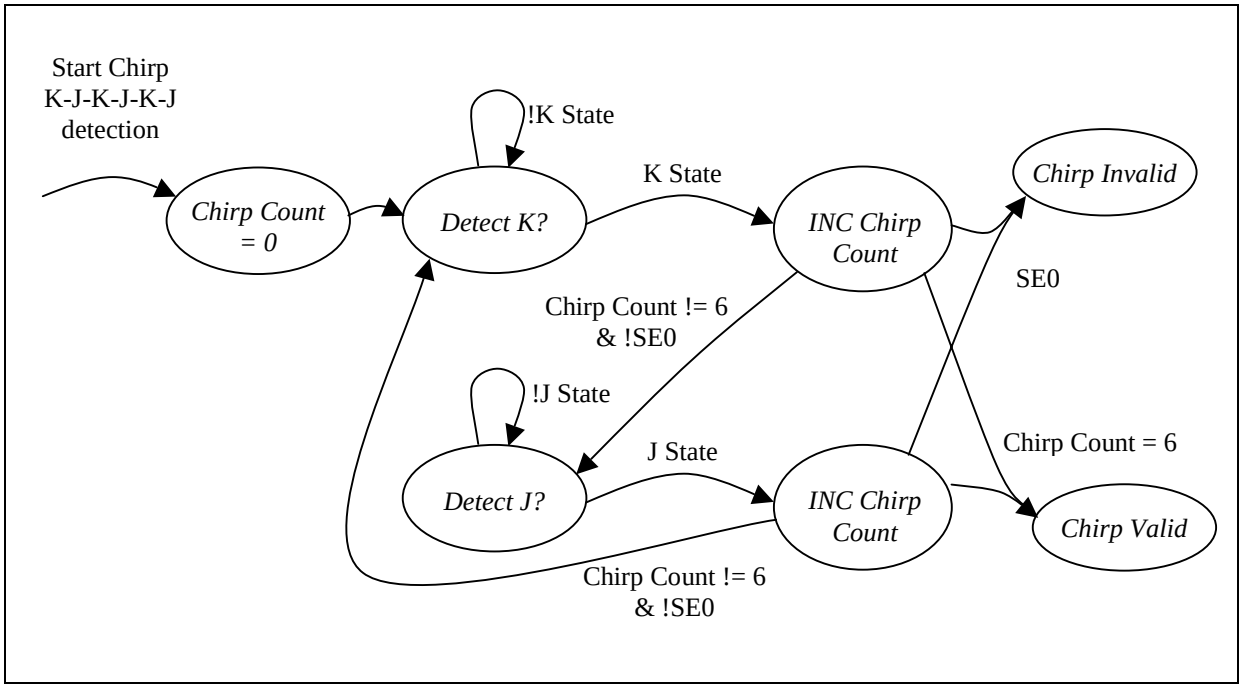


Figure 24: Chirp K-J-K-J-K-J Sequence Detection State Diagram

The Chirp K-J-K-J-K-J sequence occurs too slow to propagate through the serial data path, therefore **LineState** signal transitions must be used by the SIE to step through the Chirp K-J-K-J-K-J state diagram, where "K State" is equivalent to **LineState** = K State and "J State" is equivalent to **LineState** = J State. The SIE must employ a counter (Chirp Count) to count the number of Chirp K and Chirp J states. Note that **LineState** does not filter the bus signals so the requirement that a bus state must be "continuously asserted for 2.5 µs" must be verified by the SIE sampling the **LineState** signals.

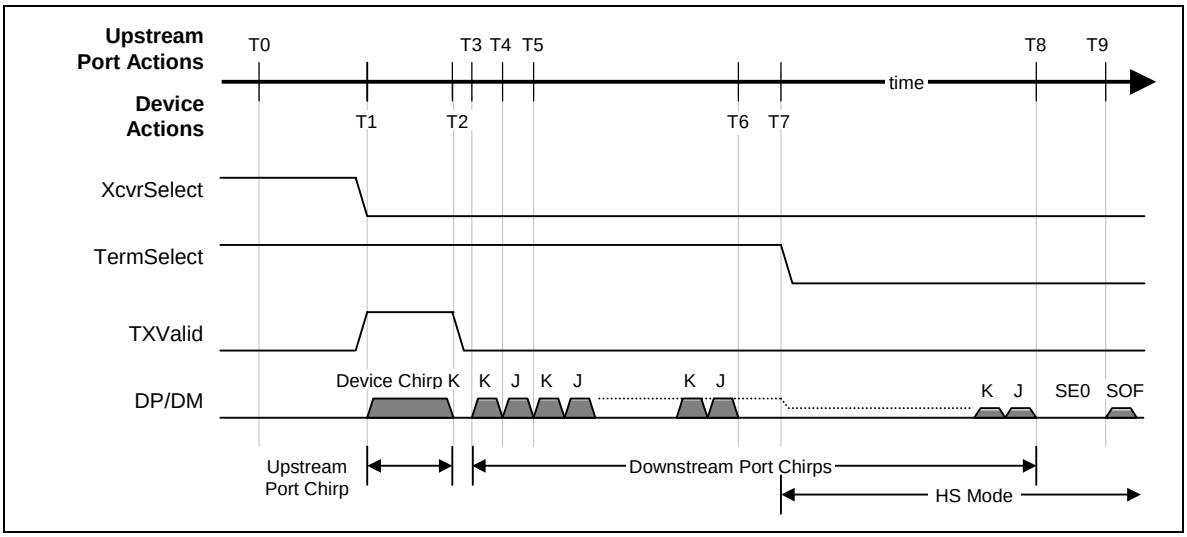


Figure 25: HS Detection Handshake Timing Behavior (HS Mode)

Table 12: Reset Timing Values

Timing Parameter	Description	Value
T0	HS Handshake begins. D+ pull-up enabled, HS terminations disabled.	0 (reference)
T1	Device asserts Chirp K on the bus.	T0 < T1 < HS Reset T0 + 6.0 ms {T_{UCHEND} - T_{UCH}}
T2	Device removes Chirp K from the bus. 1 ms minimum width.	T0 + 1.0 ms {T_{UCH}} < T2 < HS Reset T0 + 7.0 ms {T_{UCHEND}}
T3	Downstream port asserts Chirp K on the bus.	T2 < T3 < T2+100 µs {T_{WTDCH}}
T4	Downstream port toggles Chirp K to Chirp J on the bus.	T3 + 40 µs {T_{DCHBIT} (min)} < T4 < T3 + 60 µs {T_{DCHBIT} (max)}
T5	Downstream port toggles Chirp J to Chirp K on the bus.	T4 + 40 µs {T_{DCHBIT} (min)} < T5 < T4 + 60 µs {T_{DCHBIT} (max)}
T6	Device detects downstream port chirp.	T6
T7	Downstream port chirp detected by the device. Device removes D+ pull-up and asserts HS terminations, reverts to HS default state and waits for end of reset.	T6 < T7 < T6 + 500 µs {T_{WTHS}}
T8	Terminate downstream port Chirp K-J sequence (Repeating T4 and T5)	T9 - 500 µs {T_{DCHSE0} (max)} < T8 < T9 - 100 µs {T_{DCHSE0} (min)}
T9	The earliest time at which downstream port may end reset. The latest time at which the device may remove the D+ pull-up and assert the HS terminations, reverts to HS default state.	HS Reset T0 + 10 ms {T_{DRST} (Min)}

Note: T0 may be up to 4 ms after HS Reset T0.
Note: The SIE must use **LineState** to detect the downstream port chirp sequence.
Note: Due to the assertion of the HS termination on the downstream port and FS termination on the upstream port, between T1 and T7 the signaling levels on the bus are higher than HS signaling levels and are less than FS signaling levels.

5.22.2.3 Suspend Timing

If reset is entered from a suspended state, the internal oscillator and clocks of the transceiver are assumed to be powered down. Figure 26 shows how **CLK** is used to control the duration of the chirp generated by the device.

When reset is entered from a suspended state (J to SE0 transition reported by **LineState**), **SuspendM** is combinatorially negated at time T0 by the SIE. Depending on the implementation, it may take several milliseconds for the transceiver's oscillator to stabilize. The UTM must not generate any transitions of the **CLK** signal until it is "usable", where "usable" is defined as stable to within ±10% of the nominal frequency and the duty cycle accuracy 50±10%. After **CLK** is "usable", the SIE must initialize a timer (T1) and look for SE0 to be asserted for at least 2.5 us. If the test is TRUE (T1 >= Tfiltse0) then start the reset handshake protocol. If the test is FALSE, the latest time that you could successfully start the Chirp sequence was exceeded and the SIE never saw SE0 for at least 2.5 us (T0 >= Tuchend - Tuch & T1 < Tfiltse0), then return to the suspend state (assert SuspendM).

The first transition of **CLK** occurs at T1. The SIE must assert a Chirp K for 66000 **CLK** cycles to ensure a 1ms minimum duration. If **CLK** is 10% fast (66 mHz) then Chirp K will be 1.0 ms. If **CLK** is 10% slow (54 mHz) then Chirp K will be 1.2 ms. The 5.6ms (T1) requirement for the first **CLK** transition after **SuspendM**, provides 200 ns for the SIE to initialize itself (T1 to T2) and ensures enough time to assert a 1ms Chirp K and still complete before T3 with a worst case **CLK**. Once the Chirp K is completed (T3) the SIE can begin looking for downstream chirps and use **CLK** to time the process.

To detect the assertion of the downstream Chirp K's and Chirp J's for 2.5 us {T_{FILT}}, the SIE must see the appropriate **LineState** signals asserted continuously for 165 CLK cycles.

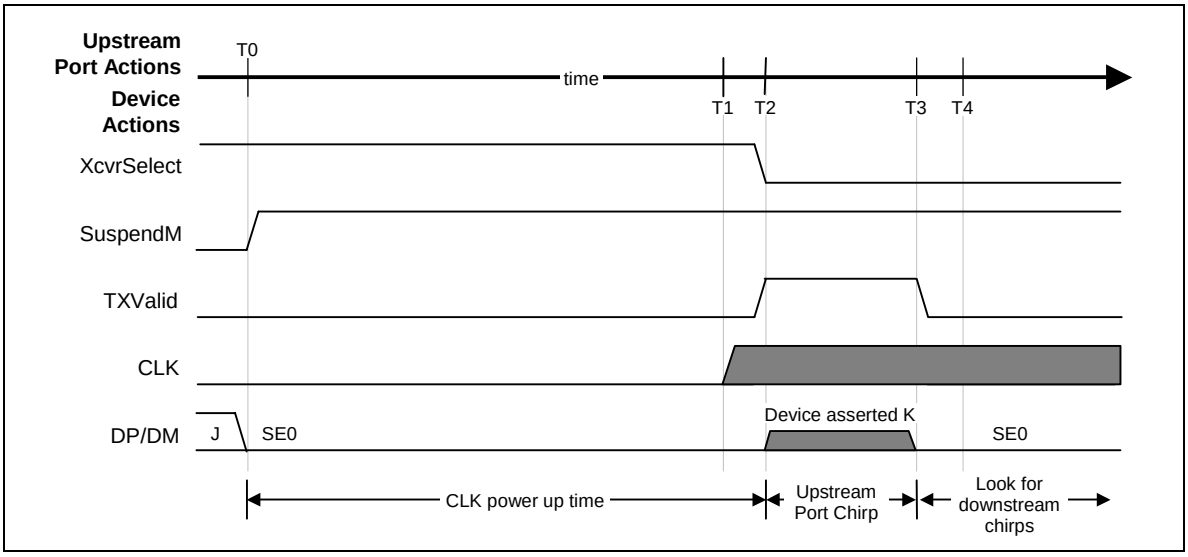


Figure 26: HS Detection Handshake Timing Behavior from Suspend

Table 13: HS Detection Handshake Timing Values from Suspend

Timing Parameter	Description	Value
T0	While in suspend state an SE0 is detected on the USB. HS Handshake begins. D+ pull-up enabled, HS terminations disabled, SuspendM negated.	0 (HS Reset T0)
T1	First transition of CLK. CLK "Usable" (frequency accurate to ±10%, duty cycle accurate to 50±10%).	T0 < T1 < T0 + 5.6 ms
T2	Device asserts Chirp K on the bus.	T1 < T2 < T0 + 5.8 ms
T3	Device removes Chirp K from the bus. (1 ms minimum width) and begins looking for downstream chirps.	T2 + 1.0 ms {T_{UCH}} < T3 < T0 + 7.0 ms {T_{UCHEND}}
T4	CLK "Nominal" (CLK is frequency accurate to ±500 ppm, duty cycle accurate to 50±5%).	T1 < T3 < T0 + 20.0 ms { T_{DRST} (Min) + T_{RSMRCY}}

5.22.3 Assertion of Resume

In this case, an event internal to the device initiates the resume process. A device with remote wake-up capability must wait for at least 5 ms after the bus is in the idle state before sending the remote wake-up resume signaling. This allows the hubs to get into their suspend state and prepare for propagating resume signaling.

The device has 10 ms. where it can draw a non-suspend current before it must drive resume signaling. At the beginning of this period the SIE may negate **SuspendM**, allowing the transceiver (and its oscillator) to power up and stabilize.

Figure 27 illustrates the behavior of a device returning to HS mode after being suspended. At T4, a device that was previously in FS mode would maintain **TermSelect** and **XcvrSelect** high.

To generate resume signaling (FS 'K') the UTM is placed in the "Disable Bit Stuffing and NRZI encoding" **OpMode** must be in “Disable Bit Stuffing and NRZI encoding” mode, **TermSelect** and **XcvrSelect** must be in FS mode, **TXValid** asserted, and all 0's data is presented on the **DataIn** bus for at least 1 ms (T1 - T2). See section 4.1.5 for a discussion of **OpMode** operation.

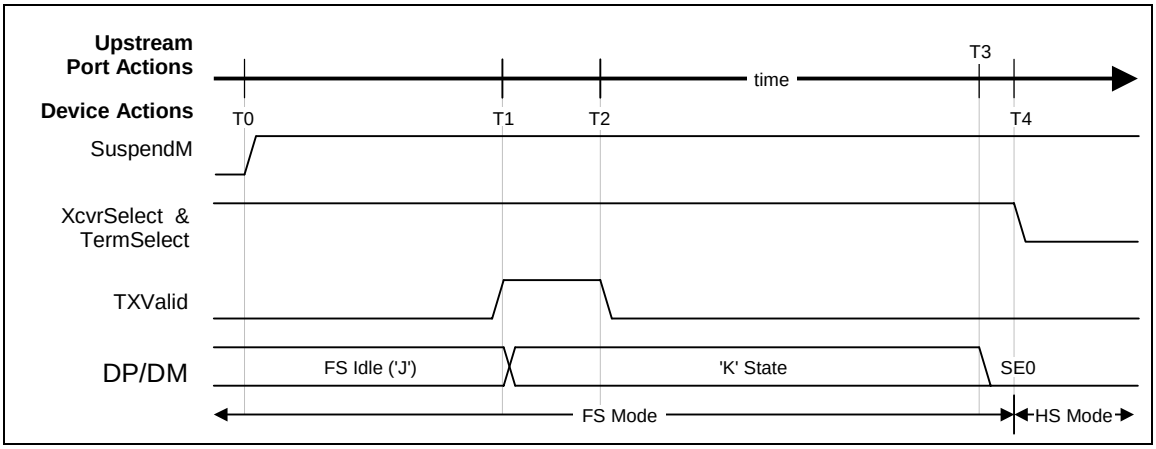


Figure 27: Resume Timing Behavior (HS Mode)

Table 14: Resume Timing Values (HS Mode)

Timing Parameter	Description	Value
T0	Internal device event initiating the resume process	0 (reference)
T1	Device asserts FS 'K' on the bus to signal resume request to downstream port	T0 < T1 < T0 + 10 ms. {T_{DRSMDN}}
T2	The device releases FS 'K' on the bus, however by this time the 'K' state is held by downstream port.	T1 + 1.0 ms {T_{DRSMUP} (min)} < T2 < T1 + 15 ms {T_{DRSMUP} (max)}
T3	Downstream port asserts SE0.	T1 + 20 ms {T_{DRSMDN}}
T4	Latest time at which a device, which was previously in HS mode, must restore HS mode after bus activity stops.	T3 + 1.33 µs {2 Low-speed bit times}

5.22.4 Detection of Resume

Resume signaling always takes place in FS mode (**TermSelect** and **XcvrSelect** = FS enabled), so the behavior for a HS device is identical to that of a FS device. The SIE uses the **LineState** signals to determine when the USB transitions from the 'J' to the 'K' state and finally to the terminating low-speed EOP (SE0 for 1.25-1.5 µs.).

The resume signaling (FS 'K') will be asserted for at least 20 ms. At the beginning of this period the SIE may negate **SuspendM**, allowing the transceiver (and its oscillator) to power up and stabilize.

The low-speed EOP condition is relatively short. SIEs that simply look for an SE0 condition to exit suspend mode do not necessarily give the transceivers clock generator enough time to stabilize. It is recommended that all SIE implementations key off the 'J' to 'K' transition for exiting suspend mode (**SuspendM** = 1). And within 1.25 µs after the transition to the SE0 state (low-speed EOP) the SIE must enable normal operation, i.e. enter HS or FS mode depending on the mode the device was in when it was suspended.

If the device was in FS mode before the suspend: then the SIE leaves the FS terminations enabled. After the SE0 expires, the downstream facing port will assert a J state for one low-speed bit time, and the bus will enter a FS Idle state (J state, maintained by the FS terminations).

If the device was in HS mode before the suspend: then the SIE must switch to the HS terminations before the SE0 expires (< 1.25 µs). After the SE0 expires, the bus will then enter a HS Idle state (maintained by the HS terminations).

Note: Handling glitches on the USB during suspend.
When a device is suspended, a J State is on the bus and the SIE should be looking for a K (resume) or an SE0 (reset). In either case, a glitch that looks like a K or an SE0 will cause the macrocell oscillator to start up. The macrocell must hold off any **CLK** transitions until **CLK** is "usable". Once **CLK** is running (typically several ms. later), the SIE must check **LineState**. If a J State is asserted then the SIE returns to the suspend state, if a K State is asserted then the SIE starts the resume process, and if an SE0 is asserted then the SIE starts the reset process.

5.22.5 HS Device Attach

Figure 28 demonstrates the timing of the UTMI control signals during a device attach event. When a HS device is attached to an upstream port, power is asserted to the device and the device sets **XcvrSelect** and **TermSelect** to FS mode (time T1).

$\mathbf{V_{BUS}}$ is the +5V power available on the USB. Device Reset in Figure 28 indicates that $\mathbf{V_{BUS}}$ is within normal operational range as defined in the USB 2.0 specification. The assertion of Device Reset (T0) will initialize the device and cause the SIE state machine to set the **XcvrSelect** and **TermSelect** signals to FS mode (T1). Note that Device Reset is not the same as the UTMI **Reset** signal. Device Reset is an input to the SIE, which in turn can use it to assert **Reset** to the UTM.

Device Reset is combinatorially asserted to the SIE after $\mathbf{V_{BUS}}$ is in normal operating range. The SIE should then combinatorially release **Reset** and **SuspendM** to the to the UTM, allowing the UTM clocks to spin up. **CLK** will begin oscillating after the internal clock is stable. See section 5.22.2.3 for a discussion of what a "stable" clock is.

The standard FS technique of using a pull up on **DP** to signal the attach of a FS device is employed. The SIE must then check the **LineState** signals for SE0. If **LineState** = SE0 is asserted at time T2 then the upstream port is forcing the reset state to the device (i.e. Driven SE0). The device will then reset itself before initiating the HS Detection Handshake protocol.

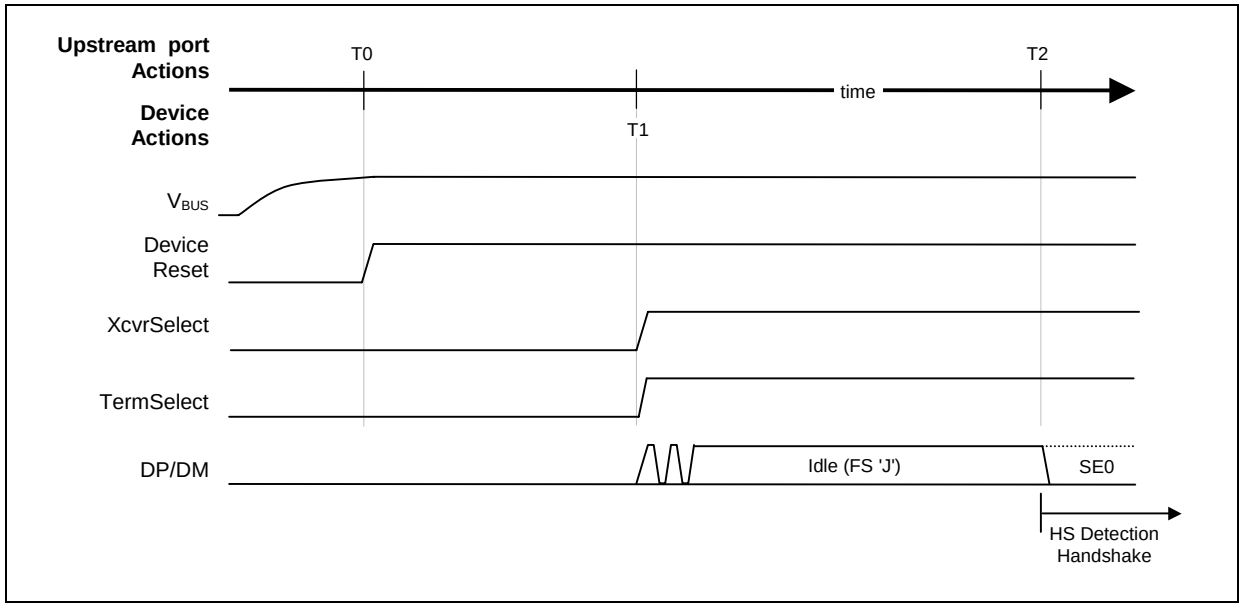


Figure 28: Device Attach Behavior

Table 15: Attach and Reset Timing Values

Timing Parameter	Description	Value
T0	Vbus Valid.	0 (reference)
T1	Maximum time from Vbus valid (> 4.01 V) to when the device must signal attach.	T0 + 100 ms {T_{SIGATT}} > T1
T2 (HS Reset T0)	Debounce interval. The device now enters the HS Detection Handshake protocol.	T1 + 100 ms {T_{ATTDB}} < T2

6 Appendix

6.1 FS Operations

The following sections provide examples of FS SYNC and EOP generation by the Transmit State Machine.

6.1.1 FS Start Of Packet

The assertion of the TXValid signal initiates packet transmission. With the bus initially in the Idle state ('J' state), a SYNC pattern ("KJKJKJKK") in its NRZI encoding is generated on the USB. To generate this pattern the SYNC data pattern (0x80) is forced into the Transmit Data Shift Register by the Transmit State Machine. TXValid will remain asserted if valid packet data bytes are available to be loaded into the Transmit Data Holding Register.

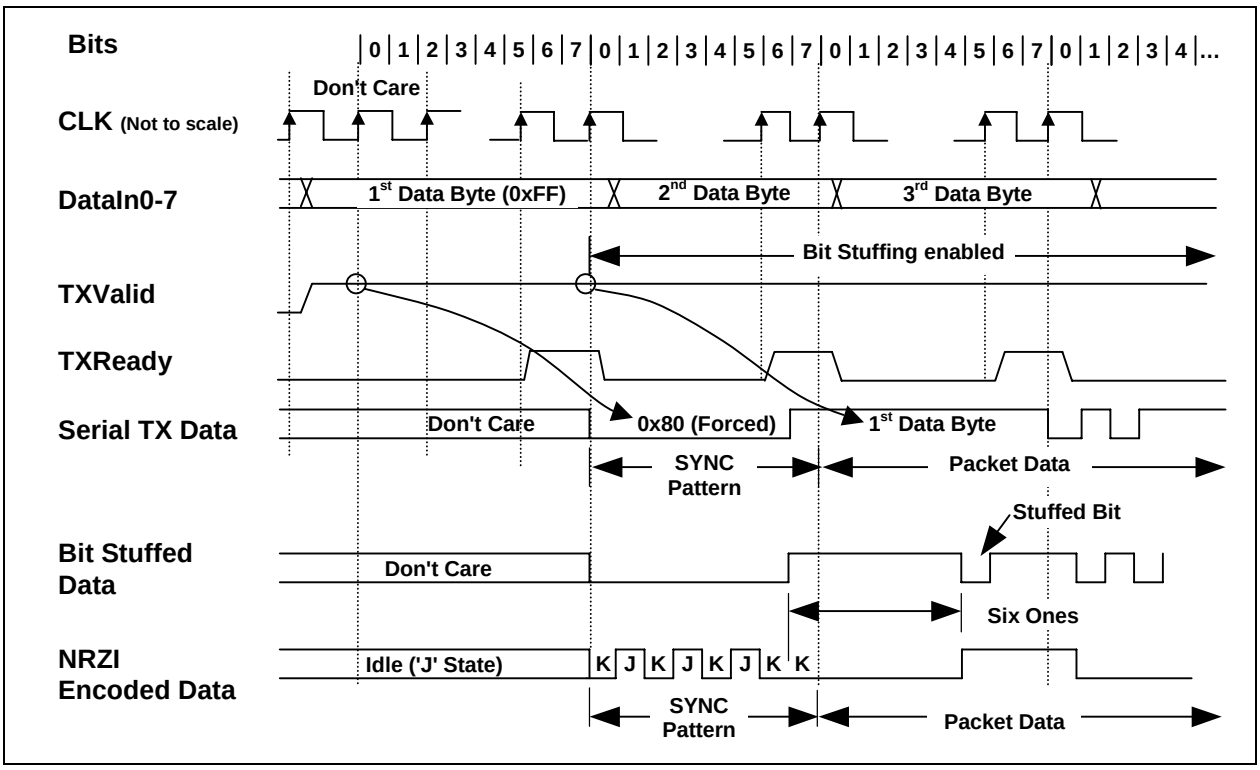


Figure 29: Data Encoding Sequence: FS SYNC

The **CLK** signal is marked "not to scale" because cycles are dropped in Figure 29 to show how **CLK** edges align with the byte boundaries of the FS data transmission.

Note: the "Serial TX Data" is the output of the TX Shift Reg block (D) in Figure 2. The "Bit Stuffed Data" is the output of the Bit Stuffer block (E) in Figure 2. The "NRZI Encoded Data" is the output of the NRZI Encoder block (F) in Figure 2.

6.1.2 FS End Of Packet

In FS mode, a single ended zero (SE0) state is used to indicate EOP and a 'J' state indicates Idle. The SE0 state is asserted for 2 bit times then a 'J' state is asserted for 1 bit time. The bus will be held in the Idle state by the FS pull-up on the bus.

Negating the **TXValid** signal initiates the FS EOP process; bit stuffing will cease, the bit stuff state machine will be reset, and the FS EOP (two bit times of SE0 followed by a single 'J' bit) will be asserted on the bus.

TXReady is negated after **TXValid** is detected false and cannot be reasserted (dashed line) until after the EOP pattern and 'J' state bit are transmitted. The delay between the assertion of **TXValid** and the first assertion of **TXReady** is UTM implementation dependent.

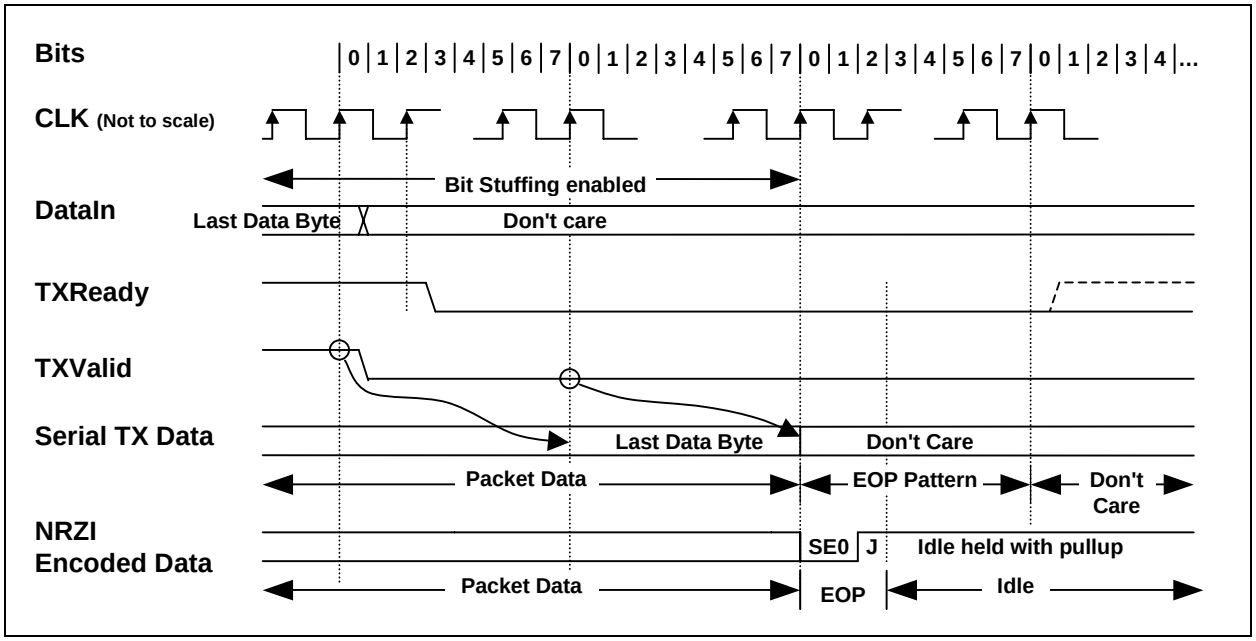


Figure 30: Data Encoding Sequence: FS EOP

Note: the "Serial TX Data" is the output of the TX Shift Reg block (D) in Figure 2. The "Bit Stuffed Data" is the output of the Bit Stuffer block (E) in Figure 2. The "NRZI Encoded Data" is the output of the NRZI Encoder block (F) in Figure 2.

6.2 HS Operation

The following sections provide examples of HS SYNC and EOP generation by the Transmit State Machine.

6.2.1 HS Start Of Packet

The assertion of **TXValid** will initiate the generation of the HS SYNC pattern on the USB by the Transmit State Machine. A SYNC pattern is the 32-bit binary string "KJKJK…JKJKJKK", in its NRZI encoding. To generate this pattern four bytes of SYNC data pattern (0x00, 0x00, 0x00, 0x80) are forced into the Transmit Shift Register by the Transmit State Machine. As long as **TXValid** is asserted the Transmit State Machine will assume that valid packet data is available on the **DataIn** bus to be loaded into the Transmit Holding Register.

Figure 31 demonstrates how the detection of a zero to one transition of **TXValid** forces the transmit state machine of the transceiver to send a 32-bit Sync pattern then begin transmission with bit stuffing enabled.

TXReady is negated to indicate to the SIE that the data on the **DataIn** bus has not been loaded into the Transmit Data Holding Register.

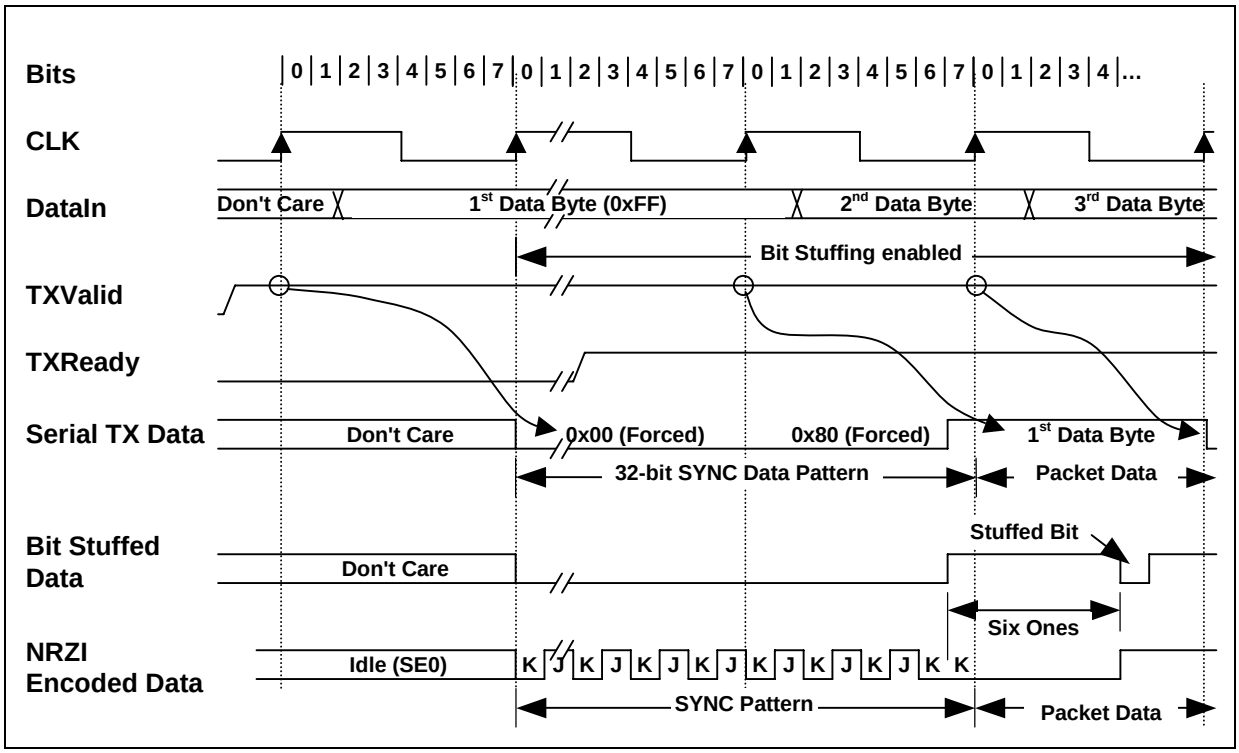


Figure 31: Data Encoding Sequence: HS SYNC

Note: the "Serial TX Data" is the output of the TX Shift Reg block (D) in Figure 2. The "Bit Stuffed Data" is the output of the Bit Stuffer block (E) in Figure 2. The "NRZI Encoded Data" is the output of the NRZI Encoder block (F) in Figure 2.

6.2.2 HS End Of Packet

The negation of TXValid will initiate the EOP sequence in the Transmit State Machine.

If required by the bit stuffing rules, a zero bit is inserted even after the last data bit.

TXReady is negated for at least one byte time while the EOP pattern is being transmitted.

TXReady is negated after **TXValid** is detected false and cannot be reasserted (dashed line) until after the EOP pattern is transmitted. The delay between the assertion of **TXValid** and the first assertion of **TXReady** is UTM implementation dependent.

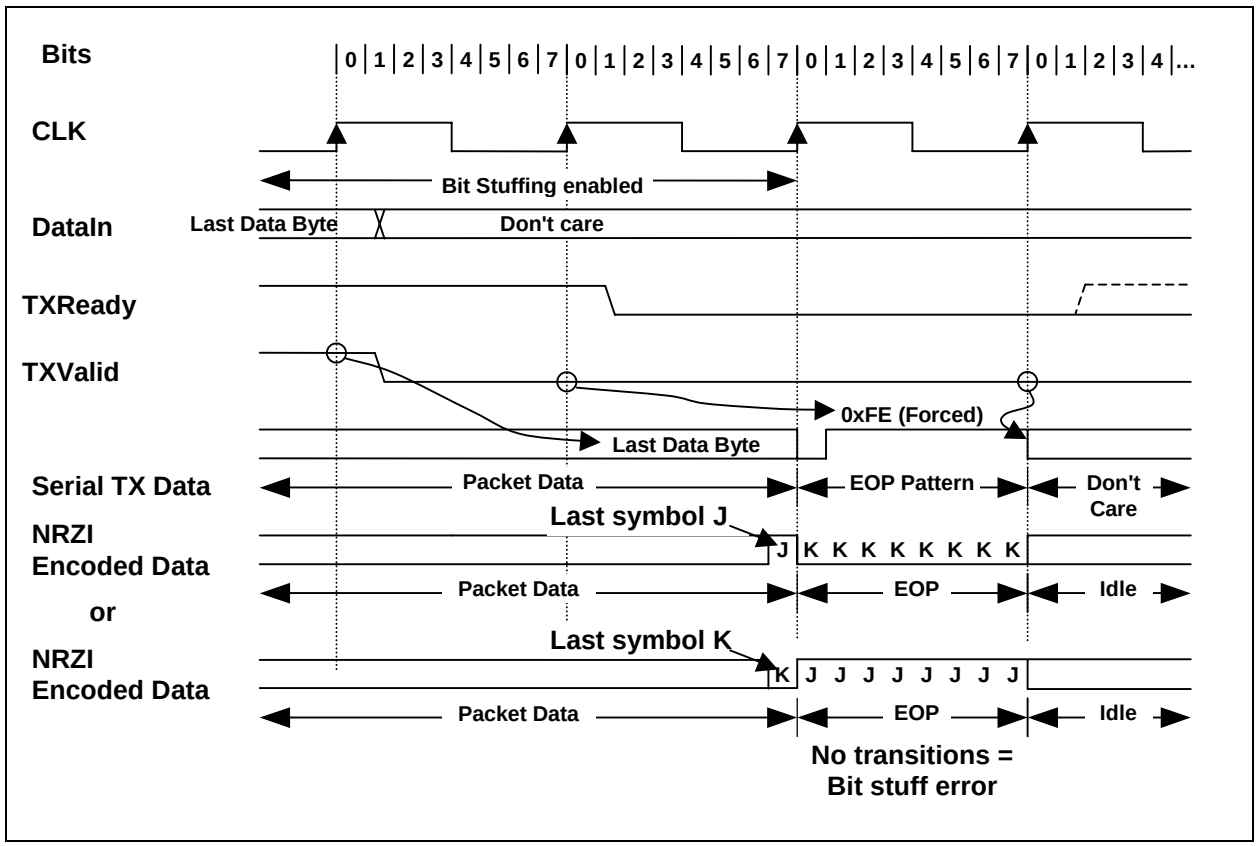


Figure 32: Data Encoding Sequence: HS EOP

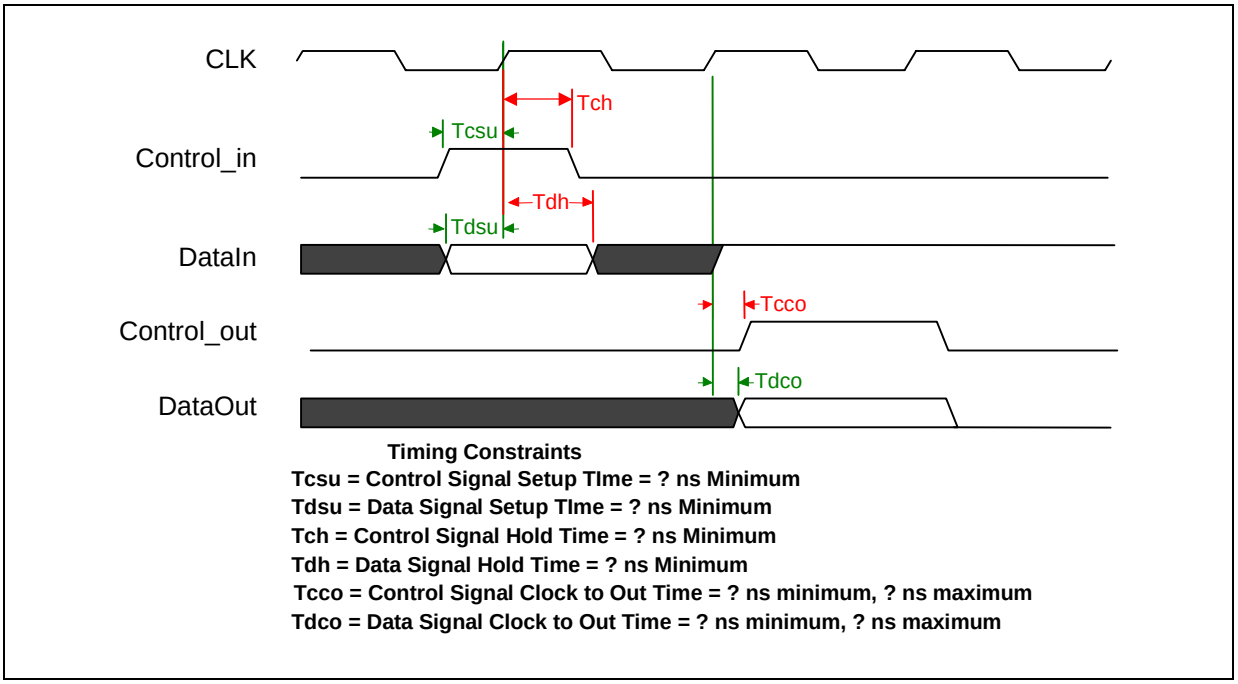
Note: the "Serial TX Data" is the output of the TX Shift Reg block (D) in Figure 2. The "Bit Stuffed Data" is the output of the Bit Stuffer block (E) in Figure 2. The "NRZI Encoded Data" is the output of the NRZI Encoder block (F) in Figure 2.

Note: The Serial TX Data is shifted out least significant bit first.

6.3 Timing Constraints

The following figure must be filled in by the UTM vendor.

Figure 33: Timing Constraints



Notes:
1. DataIn represents the signals **DataIn00** through **DataIn07** (or **DataIn15** if a 16-bit implementation).
3. Control_in represents all remaining System and Data Interface input signals.
4. DataOut represents the signals **DataOut00** through **DataOut07** (or **DataIn15** if a 16-bit implementation).
5. Control_out represents all remaining System and Data Interface output signals.

6.4 Inter-Packet Delay Overview

This section discusses the issues related to inter-packet delays when operating in HS mode. In HS mode these examples assume that the receiver contains an elasticity buffer.

6.4.1 HS Inter-packet delay for a receive followed by a transmit

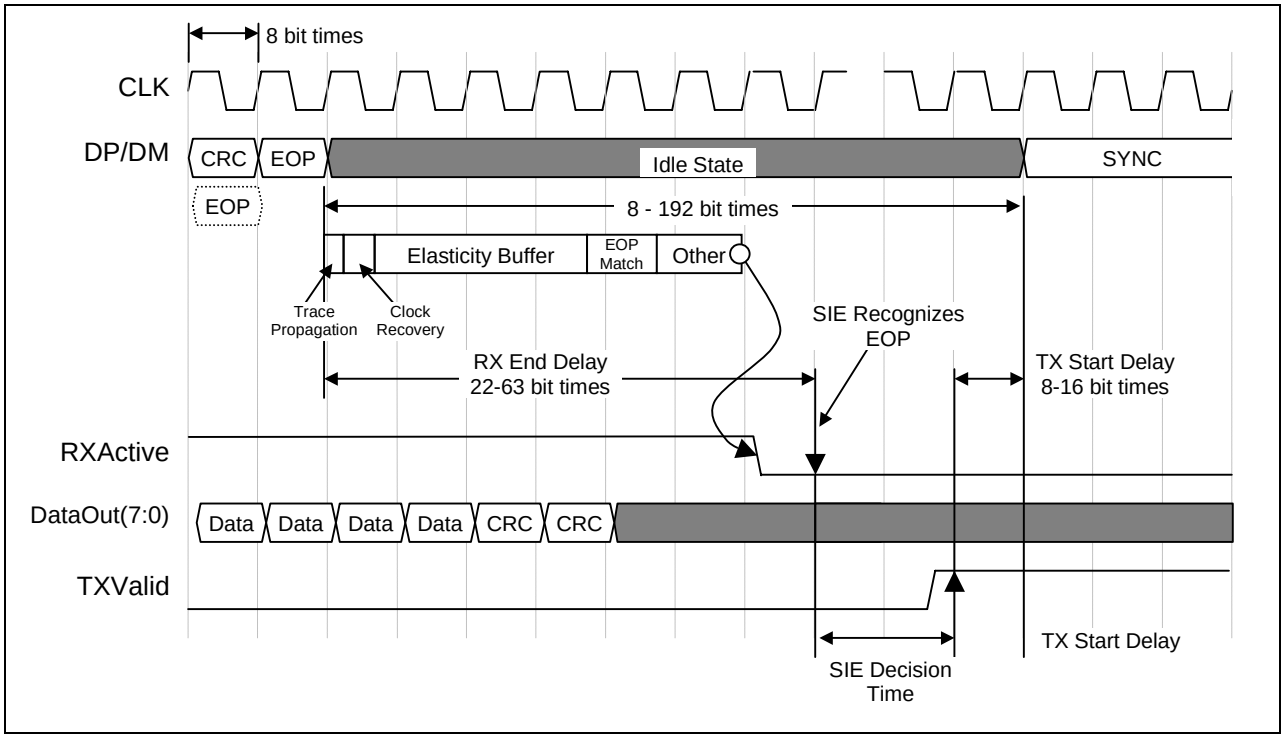


Figure 34: HS Receive to transmit inter-packet delay

DP/DM represents the serial data on the bus. Received data is synchronized to a **CLK** boundary by the UTM. The EOP on the **DP/DM** signal lines represents an EOP on the USB that arrives perfectly synchronized with **CLK**. The EOP in the dotted box represents an EOP that completes one bit time too late to be synchronized with **CLK**, representing an additional 7-bit time delay.

Assuming a worst case where the elasticity buffer is full, it will take 4 byte times to unload it. Also assuming that it takes the UTM 1 **CLK** time to decode the EOP (shifted out of the elasticity buffer) and one **CLK** time for the change in the **RXActive** signal, the total worst case delay is 63 bit times.

If EOP arrives on a **CLK** edge and the elasticity buffer is empty then the best case delay is 16 bit times.

Table 16: Receive End Delay Components

Bit Times	Reason
0-7	Data Synchronization Offset
0-2	Trace Propagation
4	Clock Recovery
0-32	Elasticity Buffer purge[6]
8	EOP Decode
10	Other miscellaneous delays
22-63	Total Receive End Delay

The Data Synchronization Offset is delay between the end of EOP and the synchronization with **CLK**, representing an additional 0 to 7-bit time delay.

Note that the Receive End Delay budget discussed in the table above is a representative example. The actual contribution of each component of the Receive End Delay is implementation specific. However, in all cases a device must meet the constraints of the Total Receive End Delay (22-63 bit times). The exception is if **RXError** is asserted and **RXActive** is negated immediately (RX *Abort 1* state).

The *Receive End Delay* is the time between the beginning of the Idle state on the bus after a receive packet and the CLK edge that the SIE detects the negation of **RXActive**.

The *Transmit Start Delay* is the time between the CLK edge that the UTM transmit state machine detects the assertion of **TXValid** and the assertion of the SYNC pattern on the bus.

The *SIE Decision Time* is the delay between the SIE detecting the negation of **RXActive** and the UTM detecting the assertion of **TXValid**.

If the *Receive End Delay* is 3 to 8 **CLK** times and the *Transmit Start Delay* is 1 to 2 **CLK**s, then the *SIE Decision Time* must be between 0 and 14 **CLK**s to meet the requirements of the USB spec.

The worst case *SIE Decision Time* assumes that the elasticity buffer is full, the EOP just missed synchronizing with **CLK**, maximum Trace Propagation, and 32 bits are in the Elasticity Buffer (8 **CLK**s), and it takes 2 **CLK**s between the assertion of **TXValid** and the beginning of the SYNC pattern. the *SIE Decision Time* must not take more than 14 **CLK**s to ensure that it does not exceed the inter-packet delay maximum of 192 bit times (24 **CLK**s).

In this example, the best case inter-packet delay assumes that the elasticity buffer is empty, the EOP is perfectly synchronized with **CLK**, 0 Trace Propagation, and 0 bits are in the Elasticity Buffer (4 **CLK**s), it takes 1 **CLK** between the assertion of **TXValid** and the beginning of the SYNC pattern, and the *SIE Decision Time* takes 1 **CLK**. In this case there will be a minimum of 38 bit times (5 **CLK**s) between EOP and SYNC. This timing exceeds the 8 bit time minimum inter-packet delay required by the spec. UTM implementations that minimize the *Receive End Delay* will provide better performance and more closely meet the minimum 8 bit time delay.

The list of delays in Table 16 should remain basically unchanged whether the interface is running with a 60 or 30 MHz **CLK**. When considering a 16-bit interface (30 MHz **CLK**), the only difference is synchronizing the serial input stream with **CLK**. *Data Synchronization Offset* will be 0-15 for a 30 MHz **CLK**. This results in a *Total Receive End Delay* of 22-81 bit times or 2-6, 30MHz **CLKs**.

[6] Some implementations may not pre-fill the elasticity buffer, and instead use the flow control provided by **RXValid** to handle cases where the downstream port clocking rate is slower than that of the upstream port. In this case the Elasticity Buffer purge time would be 0-20 bit times.

HS Inter-packet delay for a Transmit followed by a Receive
In this case the downstream port determines the inter-packet delay.

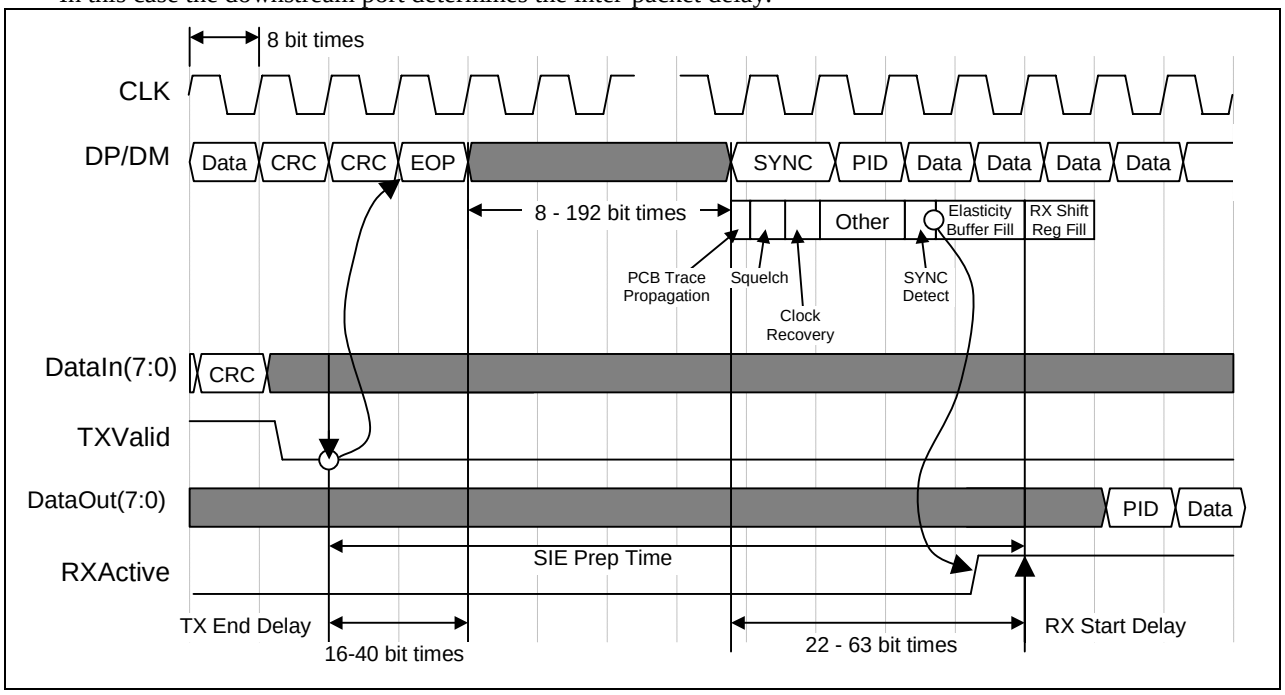


Figure 35: HS Transmit to Receive inter-packet delay

Note: To meet the minimum inter-packet delays between transmitting a packet and receiving the next, it is recommended that the receiver be disabled while transmitting. Whether this is necessary will depend on the implementation. In HS mode this can be accomplished by asserting the "squelch" signal if the HS_Drive_Enable signal is asserted. See Figure 7-1 of the USB 2.0 Specification for an explanation of the squelch and HS_Drive_Enable signals.

In this example *TX End Delay* is fixed at 16 bit times, however the worst case *TX End Delay* can be 40 bit times or longer, depending on the implementation. The 16 bit time *TX End Delay* assumes that the last byte (2nd CRC byte in Figure 35) of the packet and the EOP will be shifted out after **TXValid** is negated.

The *RX Start Delay* is calculated below.

Table 17: Receive Start Delay Components

Bit Times	Reason
0-7	Data Synchronization Offset
0-2	Trace Propagation
4	Squelch Detection
4	Clock Recovery
0-12	Elasticity Buffer fill[7]
4-24	SYNC Decode (SYNC = 12-32 bits)
10	Other miscellaneous delays
22-63	Total Receive Start Delay

[7] Some implementations may not pre-fill the elasticity buffer, and instead use the flow control provided by **RXValid** to handle cases where the downstream port clocking rate is slower than that of the upstream port.

Note that the Receive Start Delay budget discussed in the table above is a representative example. The actual contribution of each component of the Receive Start Delay is implementation specific. However, in all cases a device must meet the constraints of the Total Receive Start Delay.

Best case *RX Start Delay* (22 bit times) assumes a minimum length SYNC pattern (12 bit times), the end of the SYNC pattern is synchronized with **CLK** in a manner that will result in minimum delay, and a 0 bit time Trace Propagation delay (3 **CLK**s).

Worst case *RX Start Delay* (63 bit times) assumes a maximum length SYNC pattern (32 bit times), the end of the SYNC pattern is synchronized with **CLK** in a manner that will result in maximum delay, and a 2 bit time Trace Propagation delay (8 **CLK**s).

TX End Delay is implementation dependent and may vary from 16 to 40 bit times (2 to 5 **CLK**s).

SIE Prep Time is the delay between negating **TXValid** and detecting the assertion of **RXActive**. Assuming the best case *TX End Delay* of 16 bit times (2 **CLK**s), a minimum inter-packet gap of 8 bit times (1 **CLK** time), and a best case *RX Start Delay* of 22 bit times (3 **CLK**s), then the *SIE Prep Time* will be as short as 6 **CLK**s.

Assuming the worst case *TX End Delay* of 40 bit times (5 **CLK**s), a maximum inter-packet gap of 192 bit times (24 **CLK**s), and a worst case *RX Start Delay* (8 **CLK**s), then the *SIE Prep Time* will be up to 37 **CLK**s.

6.4.2 HS Inter-packet delay for a receive followed by a receive

In this case the minimum 32 bit time delay is guaranteed by the USB specification.

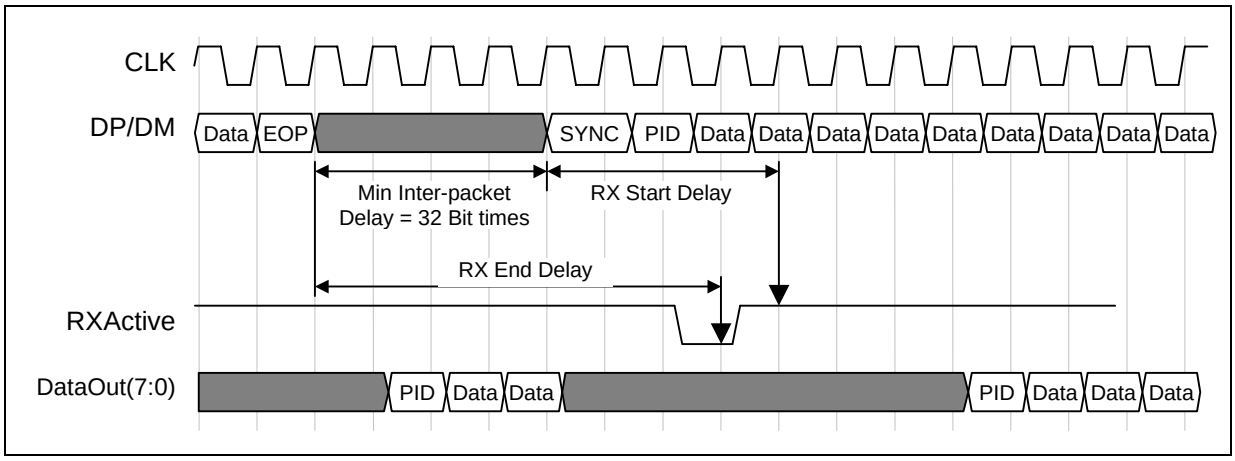


Figure 36: HS Back to back receives with minimum inter-packet delay.

In all implementations the *Rx Start Delay* + minimum inter-packet gap must be greater than *RX End Delay*. In the case where a UTM implementation pre-fills the elasticity buffer the *Rx Start Delay* (best case, 34 bit times 5 CLKs) + minimum inter-packet gap (32 bit times, 4 CLKs) will be greater than *RX End Delay* (worst case, 63 bit times, 8 CLKs). If the UTM implementation does not pre-fills the elasticity buffer then the *Rx Start Delay* (best case, 22 bit times 4 CLKs) + minimum inter-packet gap (32 bit times, 4 CLKs) will be still greater than *RX End Delay* (worst case, 51 bit times, 7 CLKs).

6.4.3 FS Inter-packet delay for a Receive followed by a Transmit

6.4.3.1 HS/FS UTM is running in Full Speed mode

For a HS/FS UTM is running in FS mode there are 40 CLK cycles per data byte or 5 CLK cycles per full speed bit time.

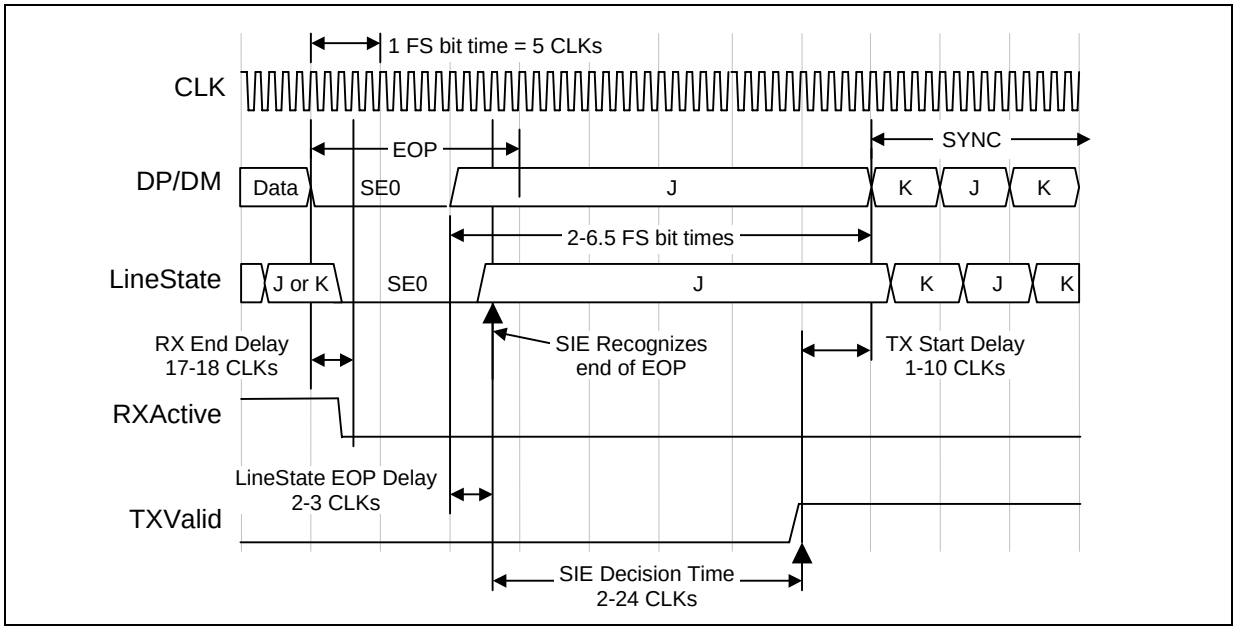


Figure 37: FS Receive to transmit inter-packet delay

LineState delay	2-3 CLKs[8]
TXValid to first K of SYNC	1-10 CLKs

Macrocell overhead	3-13 CLKs
Spec inter-packet Gap time	32 CLKs (6.5 bit times[9])

For timing the inter-packet delay the SIE must utilize LineState to determine the EOP transition from SE0 to the J-State. RXActive must also be negated before TXValid can be asserted because the SIE must parse the received data to determine what data to return with the following transmit operation.

FS SIE Decision Time must be between 7-19 CLKs[10] to ensure that 6.5 bit times FS inter-packet gap is met.

Note: In FS mode, the *RX End Delay* is relaxed. RXActive must be negated by the end of the EOP + 2 CLKs, so the best case *RX End Delay* is 17 CLKs. And worst case *RX End Delay* is 18 CLKs assuming that some clock synchronization delay.

[8] The LineState delay assumes that the DP/DM signals are double clocked to prevent metastability problems.

[9] The USB specification (section 7.1.18.1) states that a device with a detachable cable must not exceed a 6.5 bit time delay and a device with a detachable cable must not exceed a 7.5 bit time delay. Not knowing which implementation that the UTMI will be used in, the 6.5 bit time max is assumed.

[10] The minimum of 2 CLKs is required so that there is no collision on the bus. This is because the downstream facing port will force a J State for one bit time to complete the EOP. Best case LineState Delay + best case TXValid to first K of Sync + 2 = 1 bit time (5).

Note: The timing in Figure 37 allows approximately two FS bit times from the detection of TXValid asserted, to the assertion of SYNC on the USB. So if the UTMI is running with a 16 bit interface (30 MHz CLK), then the TXStart Delay is a minimum of 1 CLK and a maximum of 3 CLKs.

6.4.3.2 FS Only or LS Only UTMs

There are 32 CLK cycles per data byte in FS Only or LS Only UTM implementations, or 4 CLK cycles per bit time.

LineState delay	2-3 CLKs
TXValid to SYNC	1-8 CLKs

Macrocell overhead	3-11 CLKs
Spec inter-packet Gap time	26 CLKs (6.5 bit times)

The FS SIE Decision Time must be between 4 and 15 CLKs to ensure that 6.5 bit times FS inter-packet gap is met.

6.4.4 FS Inter-packet delay for a Transmit followed by a Receive

6.4.4.1 HS/FS UTM is running in Full Speed mode

For a HS/FS UTM running in FS mode there are 40 CLK cycles per data byte or 5 CLK cycles per full speed bit time.

For a transmit followed by a receive the downstream port determines the inter-packet delay. The SIE must measure the inter-packet delay to determine whether a timeout has occurred or not. In this case, **TXValid** and **RXActive** are not used to measure the inter-packet delay because **TXValid** is negated long before the EOP is sent and **RXActive** will not be asserted until after the SYNC pattern is recognized. The SIE must use **LineState** to measure inter-packet delays.

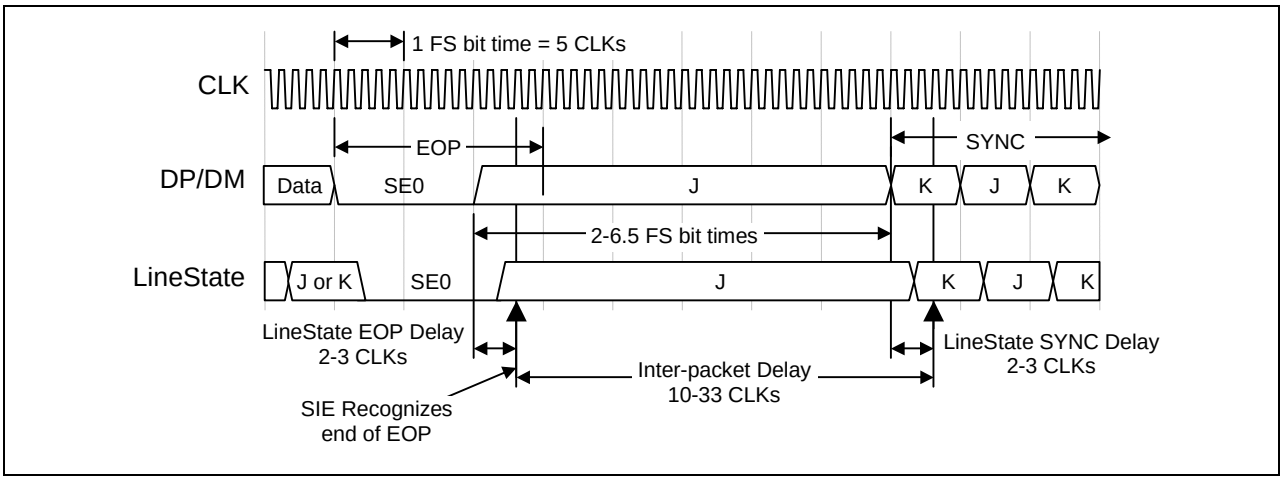


Figure 38: FS transmit to receive or receive to receive inter-packet delay

LineState EOP delay (SE0 to J State)	2-3 CLKs
LineState SYNC delay (J State to K State)	2-3 CLKs

The worst case situation is if LineState takes 2 CLKs to identify EOP and 3 CLKs to identify SYNC (or vice versa). In all other cases the LineState delays cancel.

The USB specification declares that a device must not timeout before 16 bit times (80 CLKs) and shall timeout after 18 bit times (90 CLKs).

If the inter-packet delay exceeds 89 CLKs a timeout error has occurred. Where 89 - 2 (min LineState EOP delay) + 3 (max LineState SYNC delay) = 90.

6.4.4.2 FS Only or LS Only UTMs

There are 32 CLK cycles per data byte in FS Only or LS Only UTM implementations, or 4 CLK cycles per bit time.

LineState delay (SE0 to J State)	2-3 CLKs
LineState delay (J State to K State)	2-3 CLKs

Macrocell overhead	4-6 CLKs

The USB specification declares that a device must not timeout before 16 bit times (64 CLKs) and not after 18 bit times (72 CLKs).

If the inter-packet delay exceeds 63 CLKs an error has occurred. Where 63 - 2 (min LineState EOP delay) + 3 (max LineState SYNC delay) = 64.

6.4.4.3 Full Speed Transmit

For a HS/FS UTM is running in FS mode there are 40 CLK cycles per data byte or 5 CLK cycles per full speed bit time. The SIE asserts TXValid to initiate a transmit operation. Depending on the implementation, a UTM may be capable of loading several bytes of data before TXReady is negated to throttle the data flow.

For very short packets, like an ACKnowledgemnent, only one byte needs to be loaded. In this case the SIE will assert **TXValid**, and the UTM will assert **TXReady** and load the byte into the Transmit Holding Register. The SIE having moved the last byte to the SIE will then immediately drop **TXValid**, signaling the end of the packet. This complete handshake can occur in less than one FS bit time, meanwhile the handshake packet will take approximately 19 FS bit times (8 SYNC bits, 8 data bits, 3 bit times for the EOP). This means that any inter-packet timing must use **LineState** to accurately determine when the packet is on the bus. The figure below shows the beginning of the FS handshake transmission.

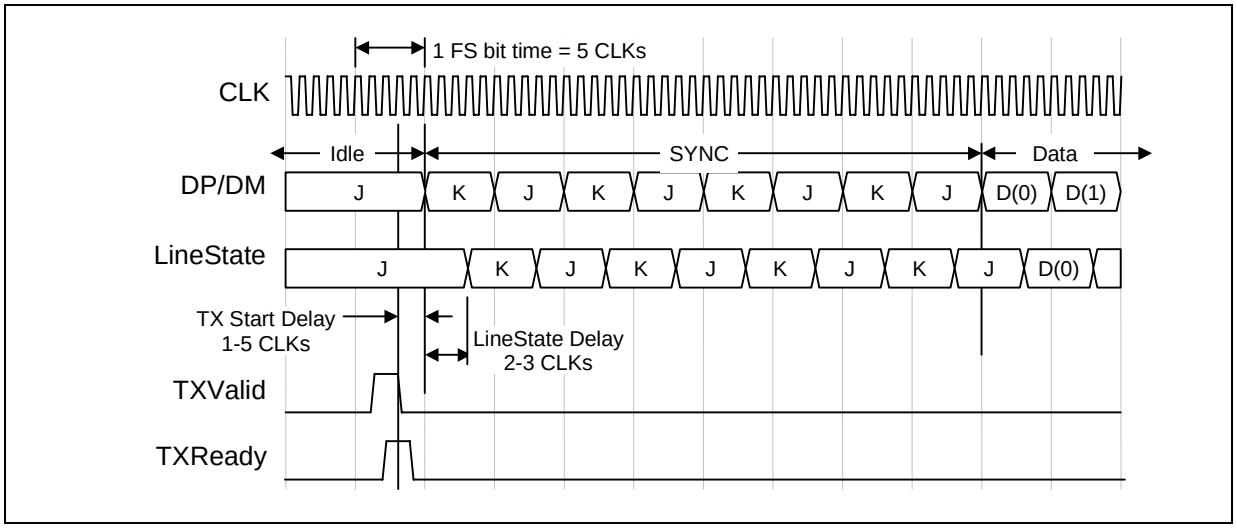


Figure 39: Start of FS handshake transmit

6.4.4.4 Full Speed Receive

RXActive more closely represents the packet on the bus, however **LineState** is still the most accurate representation of when the FS packet is on the bus.

6.5 UTM Entity Diagrams

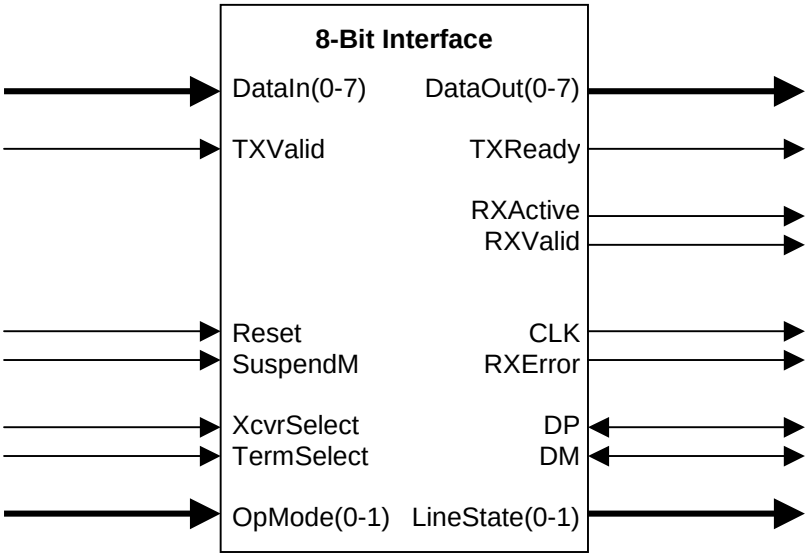


Figure 40: 8-Bit Interface Entity Diagram

Note: XcvrSelect and **TermSelect** do not exist in FS Only or LS Only implementations.

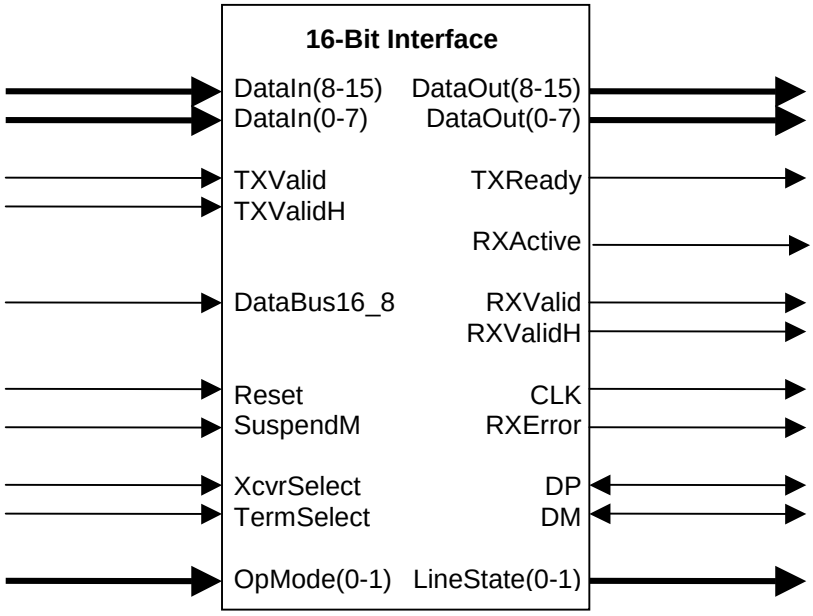


Figure 41: 16-Bit Interface Entity Diagram

Note: XcvrSelect and **TermSelect** do not exist in FS Only or LS Only implementations.

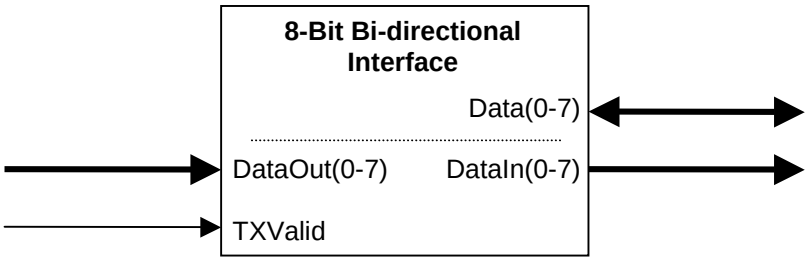


Figure 42: 8-Bit Bi-directional Interface Entity Diagram

Note: That the 16-Bit Bi-directional Interface block shown below, attaches to the 16-Bit Interface Entity Diagram above to provide a 16-bit bi-directional interface for the UTM. For the internals of this block see Figure 19.

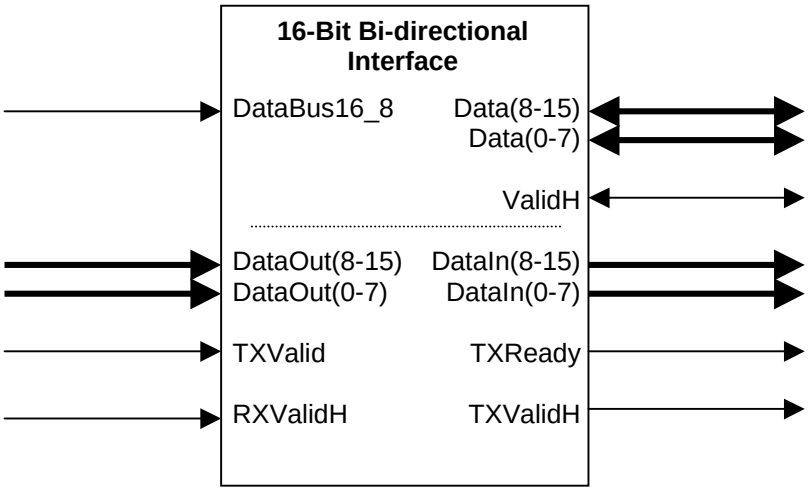


Figure 43: 16-Bit Bi-directional Interface Entity Diagram

