"A critical success factor in developing high speed serial data protocols is ramping up on these technologies quickly. We have found that MindShare books, like USB 3.0 Technology, allow engineers to learn new protocols more quickly and efficiently compared to reading the specifications." **Mike Micheletti** - USB Product Manager, Teledyne LeCroy

MINDSHARE
BRINGING LIFE TO KNOWLEDGE

For training, visit **mindshare.com**

MindShare Technology Series

USB 3.0 Technology

Comprehensive Guide to SuperSpeed USB

Donovan Anderson and Jay Trodden | MindShare, Inc.

USB 3.0 TECHNOLOGY

MindShare, Inc.

Donovan (Don) Anderson
Jay Trodden

MindShare Training Courses

Intel Architecture • Intel Haswell Processor • Intel 32/64 Bit x86 Architecture • Intel QuickPath Interconnect (QPI) • Computer Architecture	**Big Data** • Hadoop • Cassandra • ElasticSearch • MongoDB
Mobile Technology • M-PCIe • UFS • MIPI M-PHY • Intel-Based Mobile (Phone/Tablet) • Intel Atom SoC • Mobile DRAM (LPDDRx) • High Speed/EMI for Mobile Platforms	**IO Buses** • PCI Express 3.0 • PCIe IO Virtualization • USB 3.0 / 2.0 • xHCI for USB • USB 3.1 Update
AMD Architecture • AMD Opteron Processor (Bulldozer) • AMD64 Architecture	**Virtualization Technology** • PC Virtualization • IO Virtualization • ARM Virtualization
Firmware Technology • x86 Firmare (UEFI/BIOS) Architecture • BIOS Essentials	**Storage Technology** • SAS • Serial ATA • AHCI for SATA • SATA 3.0 • NVMe
ARM Technology • ARM Architecture • ARM Virtualization	**DRAM Technology** • DDRx & LPDDRx • LPDDRx
Programming • x86 Architecture Programming • x86 Assembly Language Basics • OpenCL Programming	**High Speed Design** • High Speed Design • EMI/EMC

Click here to see a complete list of live and eLearning courses.

training@mindshare.com **1-800-633-1440** **www.mindshare.com**

ARBOR
BY MINDSHARE

The Ultimate Tool to View, Edit and Verify Configuration Settings of Computers

MindShare Arbor is a computer system debug, validation, analysis and learning tool that allows the user to read and write any memory, IO or configuration space address. The data from these address spaces can be viewed in a clean and informative style. In addition, Arbor checks and flags configuration errors and non-optimal settings.

Arbor’s Software Support

MindShare’s Arbor tool supports DOS, Windows and Linux 32- and 64-bit implementations and OS-X is nearing completion. In addition, Arbor supports Intel’s JTag debuggers.

View Reference Info

Arbor can quickly display standard PCI, PCI-X and PCIe configuration registers and memory structures. All the register and field definitions are up-to-date with the PCI Express 3.0 implementation. Arbor also supports other implementations such as the x86 MSRs, USB”s xHCI Host Controller and SATA’s AHCI Host Bus Adapter. Other implementations are planned including, new x86 features and NVMe registers and data structures.

Decoding Standard and Custom Structures from a Live System

MindShare Arbor performs a system scan to record the PCI config space data, relevant memory and memory-mapped structures as well as Model Specific Registers (MSRs) and shows them in a clean and intuitive decoded format. MindShare Arbor also allows users to create their own decode files via XML.

Write Capability

MindShare Arbor provides a very simple interface to directly edit a register in PCI config space, memory address space, IO address space or MSR. This can be done in the decoded view so you see what the meaning of each bit, or by simply writing a hex value to the target location.

Scripting Capability

MindShare Arbor can also be used in a testing environment with compliance test suites, regression testing, system setup for bug evaluation and more. Arbor can run Python scripts which allow users to automate any functionality that can be performed manually from the graphical interface (and even more). MindShare provides several useful python scripts with the purchase of Arbor related to PCIe testing.

Run Rule Checks of Standard and Custom Structures

In addition to capturing and displaying headers and capability structures from PCI config space, Arbor can also check the settings of each field for errors (e.g. violates the spec) and non-optimal values (e.g. a PCIe link trained to something less than its max capability). MindShare Arbor has scores of these checks built in and can be run on any system scan (live or saved). Any errors or warnings are flagged and displayed for easy evaluation and debugging. Users can also define their own rule checks. These rule checks can be for any structure, or set of structures, in PCI config space, memory space or IO space.

Saving System Scans (XML)

After a system scan has been performed, MindShare Arbor allows saving of that system's scanned data (PCI config space, memory space and IO space) all in a single file to be looked at later or sent to a colleague. The scanned data in these Arbor system scan files (.ARBSYS files) are XML-based and can be looked at with any text editor or web browser. Even scans performed with other tools can be easily converted to the Arbor XML format and evaluated with MindShare Arbor.

USB 3.0 TECHNOLOGY

MindShare, Inc.

Donovan (Don) Anderson
Jay Trodden

The authors and publisher have taken care in preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

Library of Congress Cataloging-in-Publication Data

Anderson, Don and Trodden, Jay
USB 3.0 Technology / MindShare, Inc. Donovan Anderson, Jay Trodden....[et al.]

Includes index
ISBN: 978-0-9836465-1-8
1. Computer Architecture 2. Microcomputers - buses.
I. Anderson, Don   II. MindShare, Inc.   III. Title

Library of Congress Number:
ISBN: 978-0-9836465-1-8

Sponsoring Editor:
Project Manager: Maryanne Daves
Cover Design: Greenhouse Creative and MindShare, Inc.

Set in 10 point Palatino Linotype by MindShare, Inc.

First Edition, First Printing,

About This Book

Chapter 1: Motivation for USB 3.0

Chapter 2: USB 2.0 Background

Chapter 3: USB 3.0 Overview

Chapter 4: Introduction to End-to-End Protocol

Chapter 5: End-to-End Packets

Chapter 6: Control Protocol

Chapter 7: Bulk Protocol

Chapter 8: Interrupt Protocol

Chapter 9: Isochronous Protocol

Chapter 10: USB 3.0 Hubs

Chapter 11: Introduction to Port-To-Port Protocol

Chapter 12: LTSSM And the SuperSpeed Link States

Chapter 13: Link Commands

Chapter 14: Header Packet Processing

Chapter 15: Header Packet Flow Control

Chapter 16: Link Errors & Packet Acknowledgement

Chapter 17: Introduction to Chip-To-Chip Protocol

Chapter 18: Physical Layer Logical Functions

Chapter 19: SuperSpeed Reset Events

Chapter 20: Link Training

Chapter 21: Link Recovery and Retraining

Chapter 22: Device Configuration

Chapter 23: SuperSpeed Hub Configuration

Chapter 24: SuperSpeed Power Management

Chapter 25: SuperSpeed Signaling Requirements

Chapter 26: Compliance Testing

Chapter 27: Receiver Loopback Testing

List of Figures

List of Tables

About This Book

Scope

The *USB 3.0 Technology* book is intended as a tutorial and to serve as classroom materials for the training MindShare delivers on this subject. It should be considered a companion to the USB 3.0 specification, helping to clarify and explain the concepts and motivations for decisions that were made in the creation of the specification. This book does include limited reference information from the specification, but in general does not duplicate reference material from the specification. Doing so would potentially introduce errors into information that is rightfully the sole domain of the specification.

The MindShare Architecture Series

The MindShare System Architecture book series includes the publications shown in Table 1.

Table 1: PC Architecture Book Series

Category	Title	Edition	ISBN
Processor Architecture	80486 System Architecture	3rd	0-201-40994-1
	Pentium® Processor System Architecture	2nd	0-201-40992-5
	Pentium® Pro and Pentium® II System Architecture	2nd	0-201-30973-4
	PowerPC System Architecture	1st	0-201-40990-9
	The Unabridged Pentium® 4	1st	0-321-24656-X

Table 1: PC Architecture Book Series (Continued)

Category	Title	Edition	ISBN
Bus Architectures	PCI System Architecture	4th	0-201-30974-2
	Firewire System Architecture: IEEE 1394	2nd	0-201-48535-4
	ISA System Architecture	3rd	0-201-40996-8
	USB 3.0 Technology	1st	978-0-9836465-1-8
	Universal Serial Bus System Architecture	2nd	0-201-30975-0
	PCI-X System Architecture	1st	0-201-72682-3
	PCI Express 3.0 Technology	1st	978-0-9770878-6-0
Network Architecture	InfiniBand Network Architecture	1st	0-321-11765-4
Other Architectures	PCMCIA System Architecture: 16-Bit PC Cards	2nd	0-201-40991-7
	CardBus System Architecture	1st	0-201-40997-6
	Plug and Play System Architecture	1st	0-201-41013-3
	Protected Mode Software Architecture	1st	0-201-55447-X
	AGP System Architecture	1st	0-201-37964-3
Storage Architecture	SAS Storage Architecture	1st	0-977-08780-8
	SATA Storage Technology	1st	978-0-9770878-1-5

Cautionary Note

The reader should keep in mind that MindShare's book series often deals with rapidly evolving technologies. This being the case, it should be recognized that the book is a "snapshot" of the state of the targeted technology at the time that the book was completed. We attempt to update each book on a timely basis to reflect changes in the targeted technology, but, due to various factors (waiting for the next version of the standard to be approved, the time necessary to make the changes, and the time to produce the books and get them out to the distribution channels), there will always be a delay.

The Standard Is the Final Word

As with all of our books, this book represents the authors' interpretations of the specification - in this case the USB 3.0 specification. The authors' are mindful of this fact and attempt to gain clarification from others to minimize this affect. In any case, the specification must be considered the final word!

Documentation Conventions

The conventions used in this book for numeric values are defined in the sections that follow.

Hexadecimal Notation

This section defines the typographical convention used throughout this book. All hex numbers are followed by an "h." Examples:

```
9A4Eh
0100h
```

Binary Notation

All binary numbers are followed by a "b." Examples:

```
0001 0101b
01b
```

1 Motivation for USB 3.0

This Chapter

The Universal Serial Bus (USB) emerged in 1995 to address the many shortcomings of the PC peripheral interfaces, including both technical and end-user concerns. Improvements since the initial USB implementation have been steady in performance and capability. This chapter highlights the key motivations that have led to the latest developments in USB — the SuperSpeed bus.

The Next Chapter

Key elements of the USB 2.0 implementation help in understanding the USB 3.0 and SuperSpeed (SS) bus architectures. The next chapter reviews these key elements for those who want to refresh their USB 2.0 knowledge.

Introduction

USB 3.0 introduces a variety of new features that improve performance, reduce system power consumption, and extend functionality. The key features include:

- Improved performance using new SuperSpeed data rates.
- Improved protocols
  - End-to-End Protocol (Based on Token/Data/Handshake)
    - Data Bursting
    - Bulk Streaming
  - Port-to-Port Protocol (also called Link-to-Link Protocol)
- Enabling of new host controllers that handle all device speeds and protocols while reducing overall link and system power consumption.

These items summarize the most important elements that led to the development of USB 3.0 and the SuperSpeed bus. One additional enabling element for moving to the SuperSpeed bus is maintaining backward compatibility to the existing Low-, Full- and High-Speed devices, and backward compatibility for SuperSpeed devices so they can operate at one or more of the USB 2.0 speeds.

USB 3.0 Host Controllers

In order to support all of the enhancements afforded by the USB 3.0 specification, a new generation of USB 3.0 host controllers is required. While other USB 3.0 host controller implementations are possible, many of the principal USB 3.0 host controllers released thus far are compatible with the eXtensible Host Controller Interface (xHCI) specification. Key motivations for new controllers such as xHCI include:

- Combining support for legacy USB 2.0 low speed (LS), full speed (FS), and high speed (HS) transactions with the new USB 3.0 SuperSpeed (SS) transactions — essential in handling high bandwidth bulk streaming transfers to large-capacity mass storage devices.
- Improved power efficiency at both the bus/device and platform levels. This is becoming critical with the huge growth of mobile and embedded battery-powered platforms.
- Optional virtualization support. Many platforms now support multiple OS's running in virtual machine (VM) environments. a host controller (HC) may leverage the PCI Express Single-Root IO Virtualization (SR-IOV) model and add HC hardware support to lessen the burden on VM software.

Performance

There are few applications that require the SuperSpeed performance at this time. However, solid state drives and video and audio are some of the applications that can currently take advantage of the current SuperSpeed transfer rate of 5 Gigabits/second. In addition, the higher speed enables more demanding applications for the future.

USB Bandwidth Comparison

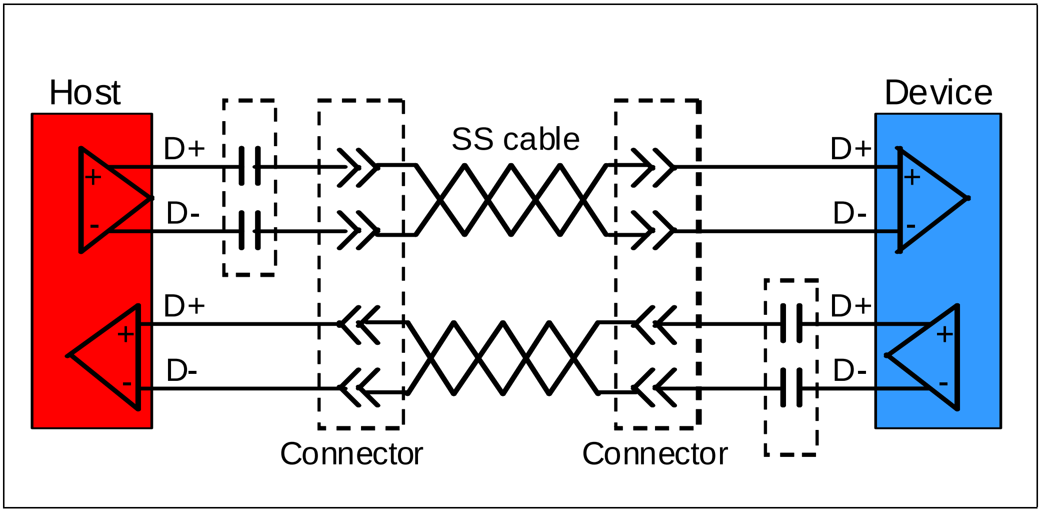
First, let's take a quick look at USB bus bandwidth. Table 1-1 shows the peak bandwidth for USB 2.0's low-, full-, and high-speed transmission rates along with USB 3.0's SuperSpeed transmission rate. Each of the bandwidth numbers are the peak theoretical maximum with a single differential pair. The SuperSpeed bandwidth number can be doubled in some cases as discussed next.

Table 1-1: USB Bandwidth Comparison

USB Speed	Bit Rate	Max Bytes/sec	Introduced
Low Speed	1.5Mb/sec	187.5KB/s	Jan, 1995
Full Speed	12Mb/sec	1.5 MB/sec	Jan, 1995
High Speed	480Mb/sec	60MB/sec	April, 2000
SuperSpeed	5Gb/sec	500MB/sec	Dec, 2008

Figure 1-1 illustrates the physical link associated with SuperSpeed. The SuperSpeed link interface includes two differential pairs; one for transmission and one for reception. This makes the aggregate bandwidth equal to one gigabyte per second. This kind of bandwidth is possible when the host controller sends an OUT transaction while a previous IN transaction is still returning data.

Figure 1-1: SuperSpeed Link Interface



2 USB 2.0 Background

The Previous Chapter

The Universal Serial Bus (USB) emerged in 1995 to address the many shortcomings of the PC peripheral interfaces, including both technical and end-user concerns. Improvements since the initial USB implementation have been steady in performance and capability. The previous chapter highlights the key motivations that have led to the latest developments in USB — the SuperSpeed bus.

This Chapter

Key element of the USB 2.0 implementation help in understanding the USB 3.0 and SuperSpeed bus architecture. This chapter reviews these key elements and is recommended reading for those who need a refresher on USB 2.0.

The Next Chapter

USB 3.0 incorporates the entire USB 2.0 implementation and adds the SuperSpeed bus. The next chapter discusses how compatibility is managed between USB 2.0 and SuperSpeed buses, and summarizes the most important features of the SuperSpeed bus.

Motivations for USB

An understanding of the motivations for USB are central to understanding the nature of the architecture and why it developed as it did. First, let's look at the primary goals of the original implementation listed in Table 2-1 on page 18. As new versions of USB have been added, the fundamental architecture and features listed in Table 2-1 have been created to ensure compatibility with the older implementations while extending performance and capabilities.

Table 2-1: Motivations for USB

Feature	Description
Better Peripheral Interface	Lower Cost and Consumer Friendly — including a serial interface with common connectors that allows sharing of a wide range of device types.
Support range of bandwidth requirements	• Low-speed — 1.5Mb/sec focusing on keyboards, mice, etc. • Full-speed — 12Mb/sec focusing on. • High-speed — 480Mb/sec focusing on mass storage, headphones, printers, etc.
Real Hot Insertion/ Removal	Allow devices to be plugged and unplugged while power is applied, and provide for automatic detection of device insertion and removal, along with the related software setup and tear-down.
Standard IO Connections	Defines standard connectors for attaching USB devices to a host computer and for attaching to the peripheral devices.
Method to Expand Topology	Expand the topology up to 127 devices via root hub ports and external hub ports, while limiting the number of cable hops between a root port and the furthest downstream device to no more than six cables (5 hubs).
Eliminate memory, IO and interrupt resource conflicts	Implement a peripheral bus whose attached peripheral devices require no memory, IO or interrupt resources.
Reduce complexity and cost of USB devices	Create a peripheral bus architecture that concentrates complexity within the host controller, thereby reducing the cost of the peripheral devices.

USB Topologies

While USB started as a peripheral bus for desktop PCs, over time USB found its way into platforms ranging from laptops and servers to embedded systems. Some embedded systems may be resource constrained and implement a subset of USB features (proprietary host controller and software, restricted speeds, custom connectors, etc.).

The foundation for all USB activity is the Host Controller. There are a variety of host controllers, including:

- UHCI (Universal Host Controller Interface) — handles only low- and full-speed devices
- OHCI (Open Host Controller Interface) — handles only low- and full-speed devices
- EHCI (Enhanced Host Controller Interface) — handles high-speed devices directly and low- and full-speed devices that attach beneath high-speed hubs that implement a transaction translator

These host controllers may be used stand-alone or combined with other controllers. The most common implementation is an EHCI controller combined with one or more UHCI or OHCI companion controllers as illustrated in Figure 2-1 on page 20.

USB 2.0 Companion Controllers

USB 2.0 host controllers typically include a companion controller implementation. These controllers normally have a single EHCI controller for handling all root ports to which high-speed devices are attached and one or more UHCI or OHCI controllers to handle all root ports to which low- and full-speed devices are attached. USB host controller implementations vary widely depending on the number of USB root ports desired. Figure 2-1 depicts a companion controller with eight root ports. The four UHCI companion controllers each have two ports to support the possibility of all eight root ports having either low- or full-speed devices attached. EHCI also implements eight ports to handle up to eight high-speed devices.

The Port Routing logic is controlled by EHCI software that tests each port to determine whether a high-speed, full-speed or low-speed device is attached. The EHCI retains all high-speed devices and connects full- and low-speed devices to the UHCI companion controllers.

3 *USB 3.0 Overview*

Previous Chapter

Key elements of the USB 2.0 implementation help to understand the USB 3.0 and SuperSpeed bus architecture. The previous chapter reviewed these key elements and is recommended reading for those who do not have a strong background in USB 2.0.

This Chapter

USB 3.0 incorporates the entire USB 2.0 bus implementation along with the SuperSpeed bus. This chapter discusses how compatibility is managed with USB 2.0 and SuperSpeed buses, and summarizes the features of the new SuperSpeed bus. Later sections of this book provide additional detail regarding these features.

The Next Chapter

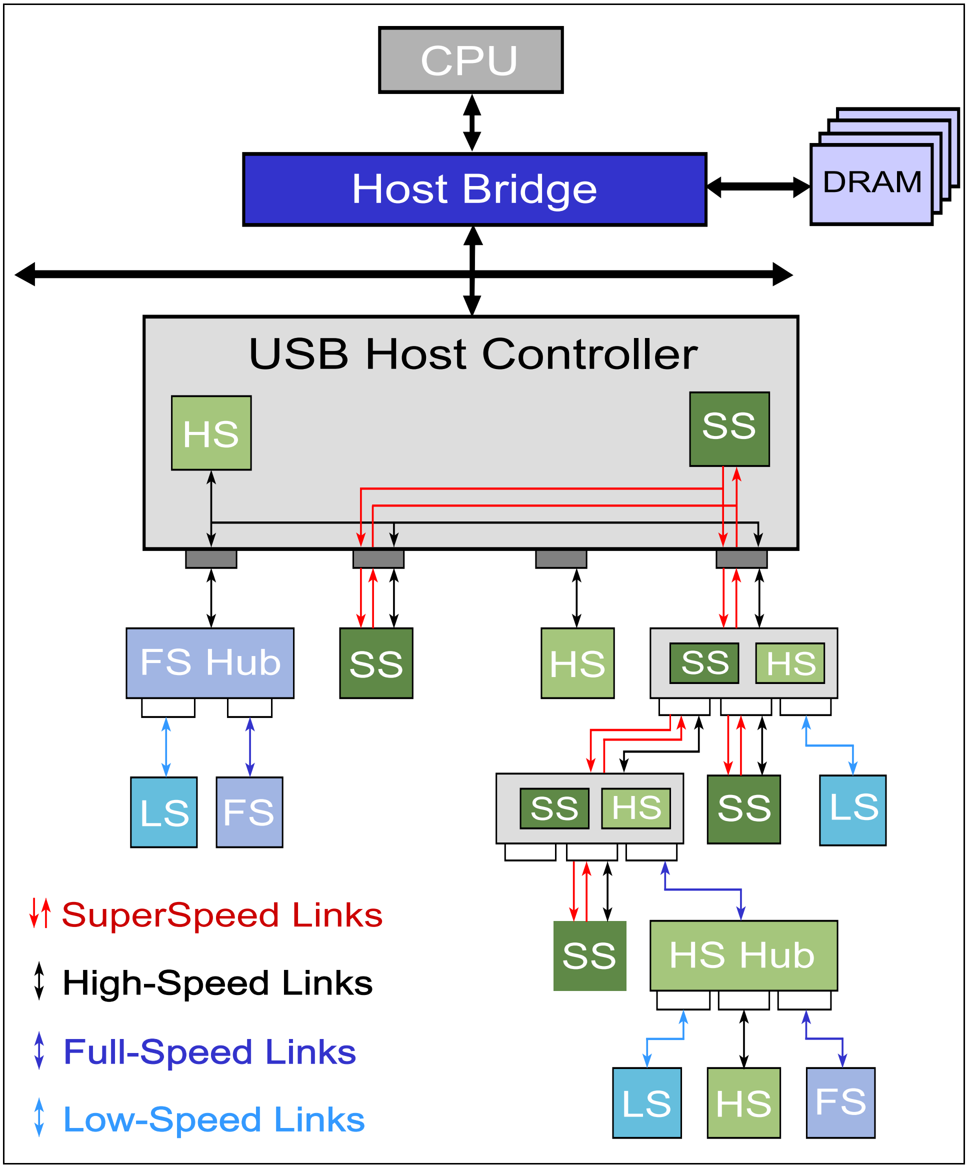
The protocol layer and the related End-to-End SuperSpeed protocol is based on the USB 2.0 Token/Data/Handshake protocol. However, the addition of new features make the End-to-End protocols very different. The next chapter characterizes the major changes brought about by the SS End-to-End protocols.

USB 3.0 Topology and Compatibility

This section discusses the USB 3.0 topology along with the related system and device compatibility issues. Refer to Figure 3-1 on page 48 during the following discussion.

USB 3.0 cables include both the USB 2.0 and SuperSpeed buses. Each 3.0 port supports either a Low-, Full-, or High-Speed device attached to the 2.0 bus or a device attached to the SuperSpeed bus. It should be emphasized that the USB 2.0 bus and SuperSpeed buses are completely independent of each other. This extends to the USB 3.0 hubs that contain a SuperSpeed hub and a USB 2.0 hub.

Figure 3-1: Example USB 3.0 Host Controller and Topology



USB 3.0 Host Controller

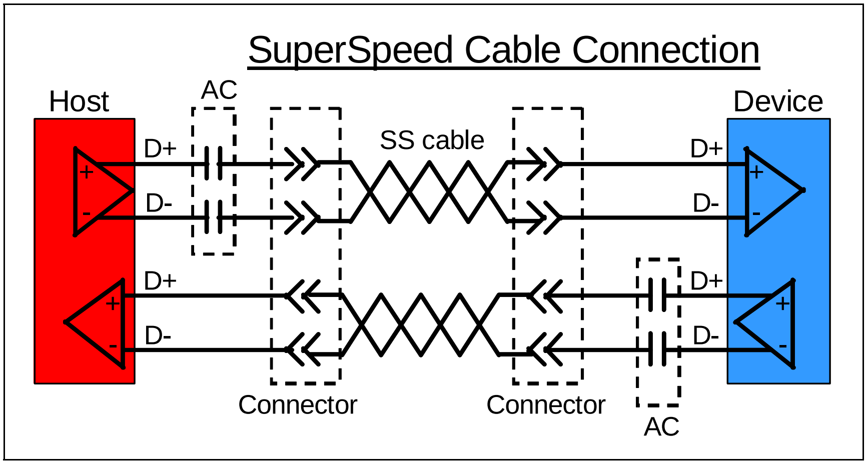
The USB 3.0 host controllers support low-speed (1.5Mb/s), full-speed (12Mb/s), high-speed (480Mb/s) and SuperSpeed (5Gb/s) operation, and do so without any companion controller requirements. This single controller can perform accesses to both USB 2.0 and USB 3.0 SuperSpeed devices simultaneously. All accesses to low- and full-speed devices are handled via a transaction translator that is integrated within the host controller. Also, the host controller may include one or more 2.0-only ports. For example, Figure 3-1 on page 48 illustrates port 1 of the host controller connecting to the HS bus only. All other ports support both HS and SS connections.

Topology

The USB 3.0 topology provides for the SS and HS bus to be implemented side-by-side. The topology maintains the same depth considerations as USB 2.0 with up to five hubs allowed in-line between a root hub port and the furthest device downstream. The upstream port of all USB SS hubs require connecting to both the SuperSpeed and high-speed buses simultaneously. These hubs integrate a SS hub and high-speed hub that contains one or more transaction translators.

The SuperSpeed bus consists of a dual simplex implementation (a transmit differential pair and a receive differential pair) allowing traffic to flow in both directions simultaneously. Figure 3-2 illustrates the SS link interface.

Figure 3-2: Transmit and Receive Differential Pairs



4 Introduction to End-to-End Protocol

Previous Chapter

USB 3.0 incorporates the entire USB 2.0 implementation and adds the SuperSpeed bus. The previous chapter discussed how compatibility is managed between USB 2.0 and SuperSpeed buses, and summarized the features of the new SuperSpeed bus.

This Chapter

The protocol layer and the related End-to-End SuperSpeed protocol is based on the USB 2.0 Token/Data/Handshake protocol. However, the addition of new features make the End-to-End protocols very different. This chapter characterizes the major changes brought about by the SS End-to-End protocols.

The Next Chapter

The role of each header packet that is employed by the End-to-End protocol is described in the next chapter. This chapter serves as a reference for these packets and includes an overview of the role each packet plays in the End-to-End protocol and details the format and descriptions of the fields within each packet. This information is essential for a thorough understanding the operation of the End-to-End protocol.

The Protocols

The End-to-End protocols define a set of protocol layer packets used in transferring information between the host controller and device endpoints. The protocol varies depending on the type of endpoint being accessed and whether the data movement is IN or OUT. Subsequent chapters discuss the specific characteristics and protocols associated with each endpoint type.

The Protocol Packet Types

SuperSpeed USB uses a collection of header packets that originate at and are received by the Protocol layer. These Protocol layer packets are categorized as four primary types by the specification. Table 4-1 lists these packet categories and lists the number of subtypes that exist in each group. Some SubTypes also have additional packet categories that are discussed in the next chapter.

Table 4-1: Protocol Packet Types and Characteristics

Name	Type Code	Sub-Types
Link Management Packets	00000b	6
Transaction Packets	00100b	8
Data Packet	01000b	NA
Isochronous Timestamp Packet	01100b	NA

The Link Management Packets (LMPs) originate and terminate at the Protocol layer, but are only exchanged between link partners. Consequently, the End-to-End protocol does not employ the LMPs and are discussed later in the context of the applications to which they apply. The related LMP functions involve:

- Power Management
- Power Budgeting
- Port Configuration following certain resets
- Vendor Device Testing

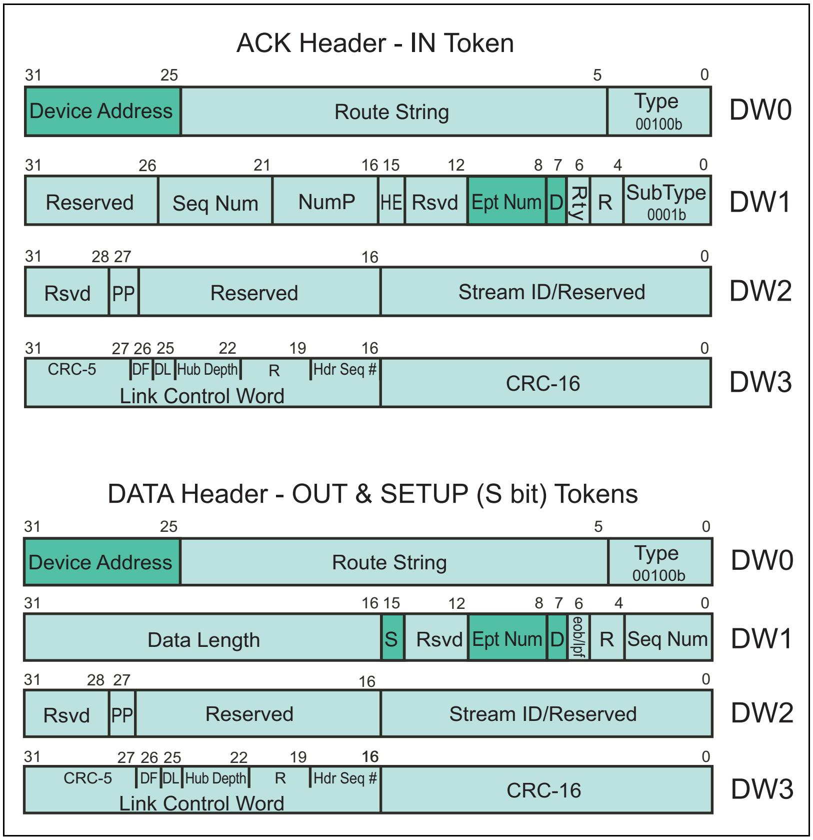
The Token Lives On

Every transaction initiated on the USB 2.0 begins with a token packet whose content is still used by the SS protocol. The specification calls this information the Address Triplet, which uniquely identifies each device endpoint in the topology. The Address Triplet includes:

- Device Address (assigned by USB configuration software)
- Endpoint Number (reported in the device’s endpoint descriptor)
- Direction (D) of data flow (D=1, IN and D=0, OUT)

The End-to-End protocol uses only ACK and DATA header packets when initiating a transaction. Also, the End-to-End protocol uses the IN, OUT and Setup token types that are incorporated into the ACK/Header packets listed in Figure 4-1. The DATA Header used for OUT transactions is converted to a SETUP header when the "S" bit in DW1/Bit 15 is set. Figure 4-1 highlights the token-related information.

Figure 4-1: SuperSpeed Headers Support IN, OUT and Setup Tokens



Bursting

SuperSpeed USB adds a new feature called Data Bursting. Like USB 2.0, the SS End-to-End protocol requires that each data packet transferred must be explicitly acknowledged with an ACK packet. However, Data Bursting allows a limited number of DATA packets to be sent without waiting for an ACK from the

5 *End-to-End Packets*

Previous Chapter

The protocol layer and the related End-to-End SuperSpeed protocol is based on the USB 2.0 Token/Data/Handshake protocol. However, the addition of new features make the End-to-End protocols very different. The previous chapter characterized the major changes brought about by the SS End-to-End protocols.

This Chapter

This chapter describes the role of each packet that is employed by the End-to-End protocol. This chapter serves as a reference for these packets and includes an overview of the role each packet plays in the End-to-End protocol and details the format and descriptions of the fields within each packet. This information is essential for understanding the operation of the End-to-End protocol.

The Next Chapter

Control transfers, also known as message pipes, provide a USB-specific mechanisms that allow requests to be issued to USB devices. These transfers use the bi-directional endpoint zero (called the default endpoint). The next chapter discusses the transfer protocol and example applications associated with Control transfers.

Three Categories of End-To-End Packets

Figure 5-1 on page 80 is a reminder of the various interface layers associated with the SuperSpeed bus and provides some detail regarding the packet types used in the End-to-End protocol. The host controller initiates each transaction sequence by sending a token packet (e.g., ACK packet for IN tokens and DATA packet for OUT tokens). The SuperSpeed End-to-End protocol defines the sequence of packets exchanged between the host controller and targeted USB device. Three categories of protocol layer packets are used in conjunction with the End-to-End protocol:

- Transaction Packets (TPs) — these packets are all ***Header*** packets and include seven SubTypes that in general emulate the token/data/handshake sequence, control data flow and manage End-to-End connectivity.

- Data Packet — this packet is categorized separately from the Transaction Packet because it consists of both a ***Header*** packet and the ***Data Packet Payload*** (DPP). In some cases errors may result in the payload being dropped in which case the End-to-End protocol will detect the missing payload and retransmit the Data packet.
- Isochronous Timestamp Packet (ITP) — this header packet is the only broadcast packet used in the SuperSpeed protocol and is a replacement for the Start of Frame (SOF) packet used by the USB 2.0 bus implementation. As its name indicates the ITP carries bus timing information that is broadcast to all devices in the network that are ready to accept the packet.

Note: Figure 5-1 lists Link Management Packets at the bottom of the Protocol Layer, but they are not used in the End-to-End protocol even though they are defined as protocol layer packets. Link Management packets are associated with a variety of functions covered later.

Figure 5-1: Protocol Layer Overview and Packet Types

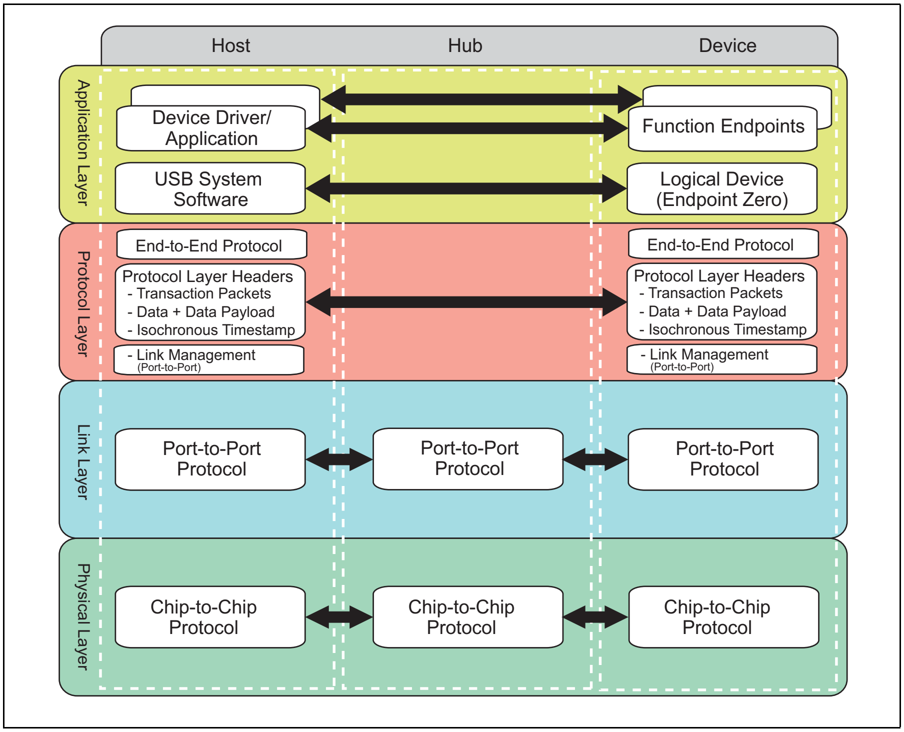


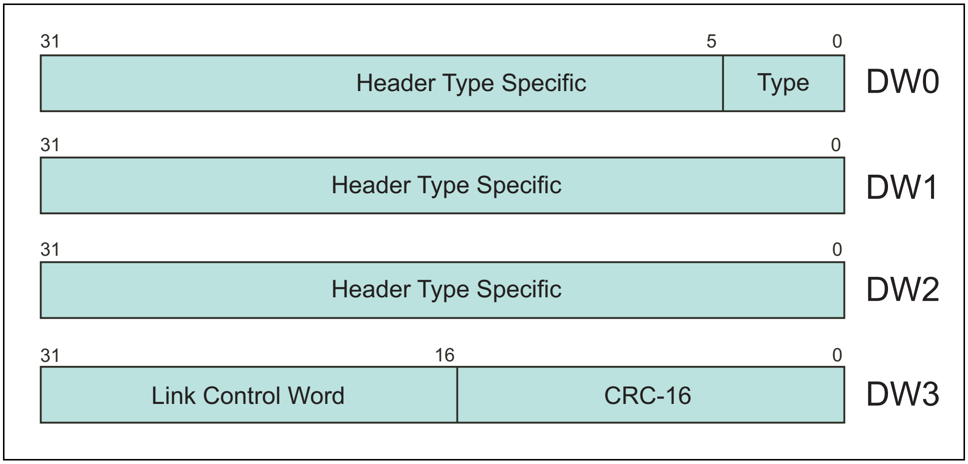
Table 5-1 lists the four Protocol Layer Header Types and indicates packet direction, type code and number of subtypes. All protocol layer packets consist of headers except for the Data Packet Payload. A basic header consists of 12 header bytes, along with a CRC-16 that covers the 12-byte header and a 16-bit Link Control Word for a total of 16 Bytes as shown in Figure 5-2. Header packets are subject to the Port-to-Port protocols that perform two primary functions associated with the Protocol Layer headers:

- Flow Control — verifies that the receive buffer can accept the header
- Header Error Detection and Recovery — verifies that all header packets exchanged between link partners are received without error. Any error condition triggers a recovery process resulting in the header being retried until it is successfully transferred.

Table 5-1: Protocol-Layer Packet Categories and Characteristics

Packet Category	Type Code	Sub-Types	Direction	Protocol
Transaction	00100b	8	Sub-Type Specific	End-to-End
Data (Header + Payload)	01000b	NA	Bi-directional	End-to-End
Isochronous Timestamp	01100b	NA	Downstream Only	Multicast
Link Management	00000b	6	Mixed	Port-to-Port

Figure 5-2: Basic Header Packet



6 Control Protocol

Previous Chapter

The role of each packet employed by the End-to-End protocol is described in the previous chapter. It serves as a reference for these packets and includes an overview of the role each packet plays in the End-to-End protocol and details the format and descriptions of the fields within each packet. This information is essential for understanding the operation of the End-to-End protocol.

This Chapter

Control transfers, sometimes called message pipes, provide a USB-specific mechanisms that allow requests to be issued to USB devices. These transfers use bi-directional endpoint zero (called the default endpoint). This chapter discusses the transfer protocol and example applications associated with Control transfers.

The Next Chapter

Many common devices use the Bulk transfers including printers, scanners, and mass storage devices. The next chapter introduces the capabilities and features associated with the SuperSpeed bulk endpoints, including the new DATA Bursting feature and Bulk Streaming protocols.

Introduction to SuperSpeed Control Transfers

The first transactions targeting a device, once it has been detected, are control transfers. SuperSpeed uses the same control transfer scheme as USB 2.0 and is based on the Token/Data/Handshake sequence. Each SuperSpeed device must implement a default control endpoint (always endpoint zero) used for configuring devices, controlling device states, and other aspects of the device's operation. The control endpoint responds to a variety of USB specific requests that are delivered via control transfers. For example, when a device is detected on the Universal Serial Bus, host software must access the device's descriptors to determine the device type and operational characteristics. The USB specification defines a variety of USB device requests for controlling hub configuration and operation, as well as USB peripheral devices. These requests may be "standard" requests used to perform typical operations required of most devices. Class or vendor-specific requests provide special functionality specific to some device types. See "Standard Device Requests" on page 486 for a detailed list of standard requests supported by SuperSpeed USB.

Control transfers consist of either two or three stages as described below:

- Setup Stage - control transfers always begin with a setup transaction that always delivers 8 bytes of data to endpoint zero. The 8 bytes of data defines the type of request to be performed (e.g., Getting the contents of a device descriptor).
- Data Stage (optional) - this stage is defined only for requests that require data transfers to or from endpoint zero. For example, the GetDescriptor request requires the host controller to perform an IN transaction to read the contents of the requested descriptor information.
- Status Stage - this stage reports the result of the requested operation and is always the final stage of a control transfer.

Following is a list of characteristics implemented by Control Endpoints:

- Max Packet Size - The data packets delivered during the data stage of a control transfer have a maximum payload size of 512 bytes. USB 2.0 restricts the maximum payload to 64 bytes.
- Data Bursting - Control transfers do not support Data Bursting.
- Error Recovery - Control transfers participate in error detection and recovery mechanisms to provide a "best effort" delivery of data based on the USB "three strikes and you're out" policy. In this case, the error is forwarded to software for handling.
- Bus Bandwidth Allocation - control transfer scheduling guarantees at least 10% of the bus bandwidth be reserved for control transfer. If additional bus bandwidth is available, then more than 10% can be allocated to control transfers during a given bus interval.

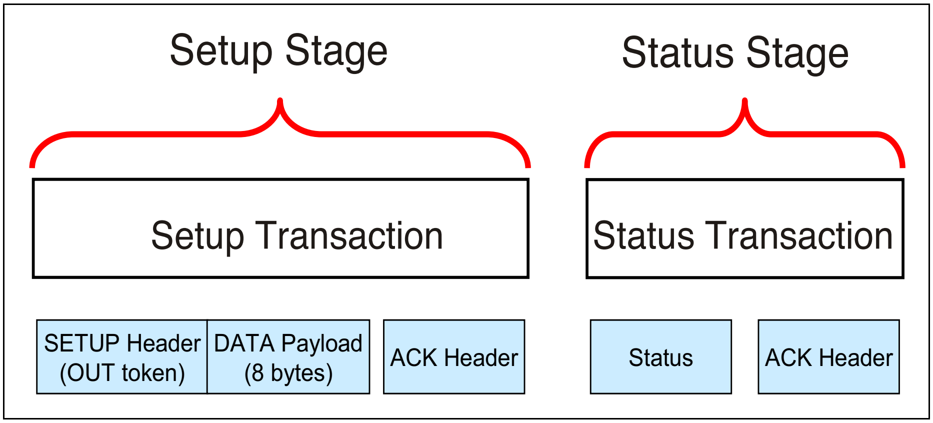
Control Transfer Structures and Examples

All control transfers are performed using either the two-stage or three-stage sequence. This section details the elements and structure of control transfers and provides example applications.

Two-Stage Control Transfer Structure

Two-stage requests commonly deliver commands to a device or hub. Examples include SetAddress, SetConfiguration, SetFunctionWake, and SetPortPower. The two-stage control transfers consist of a Setup and Status Stage as illustrated in Figure 6-1. The Setup Stage is simply a Setup transaction comprising a DATA Header with the ***Setup*** bit set, the 8-byte Data Packet Payload (DPP), both sent by the host controller, and an ACK packet returned from the device. In this example, the 8-byte DPP defines a two-stage control transfer (e.g., a SetAddress request). After the host controller has delivered the Setup request it sends the new SuperSpeed Status header to device endpoint zero and the device returns an ACK header to confirm the request has successfully completed.

Figure 6-1: Two-Stage Control Transfer



Three-Stage Control Transfer Structure

Three stage control transfers consist of the Setup Stage followed by the Data stage that consists of one or more IN or OUT transactions depending on the specific request and ends with the Status Stage (See Figure 6-2). The most common types of three-stage control transfer are getting descriptor or status information from peripheral devices and hubs.

The 8-bytes of data associated with a Setup Stage of a three-stage control transfer specifies the amount of data to be transferred during the Data Stage. The

7 Bulk Protocol

Previous Chapter

Control transfers, sometimes called message pipes, provide a USB-specific mechanisms that allow requests to be issued to USB devices. These transfers use bi-directional endpoint zero (called the default endpoint). The previous chapter discusses the transfer protocol and example applications associated with Control transfers.

This Chapter

Many common devices use the Bulk transfers including printers, scanners, and mass storage devices. This chapter introduces the capabilities and features associated with the SuperSpeed bulk endpoints, including the DATA Bursting and Bulk Streaming protocols.

The Next Chapter

Interrupt transactions provide periodic access to an endpoint and limit the amount of data that can be transferred during a bus interval. The next chapter discusses the applications, capabilities, and behaviors associated with interrupt transfers.

Introduction to Bulk Transfers

SuperSpeed Bulk endpoint characteristics include several important new features, but in general have the same characteristics as USB 2.0 Bulk transfers. The attributes of SuperSpeed Bulk endpoints include:

- End-to-End reliable data transport between the host and device is accomplished through an error detection and retry mechanism. Three failed attempts to deliver data results in software handling the error recovery.
- No reservation for bus bandwidth is made for bulk transfers. Consequently, there are no latency or bandwidth guarantees.
- Maximum payload size increases from 512 bytes to 1024 bytes

- Data Bursting provides for up to 16 DATA packets (16 KB) that can be sent or received without having received an ACK. This capability can significantly reduce data transfer latency.
- Bulk Streaming provides nearly 64K of addressable buffers associated with a single bulk endpoint. This is managed via the 16-bit StreamID field.

Bulk End-to-End Protocol

The End-to-End protocol associated with bulk endpoints involves the exchange of several Protocol Layer packets and further focusses on several critical fields within these packets. This chapter explains the protocol requirements and provides a number of example transfers for both IN and OUT operations.

Common fields related to both IN and OUT transfers include:

- ***Route String*** — Used by hubs to forward packets to the destination device
- ***Address Triple*** — Device Address, Endpoint Number, and Direction that uniquely each endpoint within the topology (aka, token information)
- ***Seq Num*** — End-to-End Sequence Numbers ensure DATA packets arrive in order
- ***NumP*** — Number of packet buffers available for receiving or sending DATA packets
- ***Retry*** — Indicates failed transfer of a Data Packet that must be retried
- ***PP*** — Packet Pending bit from host notifies the device that this packet is the last currently scheduled for delivery

Bulk IN SuperSpeed Transaction Protocol

The host issues ACK packets to initiate IN transactions. IN transaction protocol involves several areas of focus as listed below:

- Data bursting
- End-to-End flow control
- Short packets
- Data transfer errors and retry
- Host and device responses to all valid conditions
- Timeout conditions and values

The protocol requirements associated with each of the topics is covered in the following sections.

IN Data Bursting

Whether or not a bulk endpoint supports bursting is reported in the Endpoint Companion descriptor. Table 7-1 shows the first three fields of the descriptor including MaxBurst size supported.

Table 7-1: SuperSpeed Endpoint Companion Descriptor

Offset	Field Definition	Field Size	Description
0	Length	1	Size of the Endpoint Companion Descriptor in bytes
1	Descriptor Type	1	Descriptor Type = 48
2	MaxBurst	1	Maximum number of DATA packets this endpoint can send or receive that have not been acknowledged. Values range from 0 to 15 and translate to 1 - 16 packets

8 *Interrupt Protocol*

Previous Chapter

Many common devices use the Bulk transfers including printers, scanners, and mass storage devices. The previous chapter introduces the capabilities and features associated with the SuperSpeed bulk endpoints, including the new DATA Bursting feature and Bulk Streaming protocols.

This Chapter

Interrupt transactions provide periodic access to an endpoint and limit the amount of data that can be transferred during a bus interval. This chapter discusses the applications, capabilities, and behaviors associated with interrupt transfers.

The Next Chapter

Some devices require constant data delivery normally handled as synchronous transfers, such as the connection between an MP3 player and headphones. However, USB does not support synchronous transfers, so an alternate solution called isochrony is employed. The next chapter details the application, capabilities and characteristics of Isochronous transfers.

Introduction to Interrupt Transfers

SuperSpeed interrupt endpoint characteristics are similar to USB 2.0 interrupt endpoint characteristics, but include several new features. The attributes of SuperSpeed interrupt endpoints include the following:

- Periodic transfers target device endpoints that require occasional access. Classical interrupt devices like keyboards and mice may not be able to send data to the host any faster than every 10 ms and may not be in use for long periods of time.
- Interrupt transfers use a reliable transport mechanism to ensure data is successfully transferred between the host and device. This is accomplished through CRC error detection and end-to-end retry. Three failed attempts to deliver data results in software being notified to handle error recovery.

- Endpoint descriptors specify a service interval (sometimes called the polling interval) that defines the frame or μframe during which data is transferred. The ***interval*** value reported in the endpoint descriptor is based on a value, ranging from 1-16, that is used as an exponent as follows:

  $2^{interval-1}$

  For example, a keyboard with a polling interval of 8ms, requires an ***interval*** value of 7.

  $2^{7-1} \rightarrow 2^{6} \rightarrow 64$ *μframes = 8ms*

  A polling interval may range from 1 to 32,768 on power's of 2 boundaries.

- ***MaxBurstSize*** is reported in the Endpoint Companion descriptor. Interrupt endpoints may have the MaxBurstSize value set as follows:
  MaxBurstSize = 0; supports 1 DATA packet
  MaxBurstSize = 1; supports 2 DATA packet burst
  MaxBurstSize = 2; supports 3 DATA packet burst
- ***MaxPacketSize*** is reported in the Endpoint descriptor as follows:
  - if MaxBurstSize=0; the MaxPacketSize value in the Endpoint descriptor may be a value from 1-1024
  - if MaxBurstSize=1 or 2; the MaxPacketSize must be 1024
- Two types of interrupt endpoint transfers exist in SuperSpeed as defined by the Interrupt Endpoint descriptors:
  - Standard periodic transfers
  - SuperSpeed asynchronous notifications (including WAKE, LATENCY TOLERANCE messages and BUS INTERVAL ADJUSTMENT messages.
- The periodic nature of interrupt transfers requires the host to access the endpoint during the specified service interval. Beyond that difference, Interrupt IN endpoints use the same End-to-End flow control mechanisms that are used in Bulk transfers. Other variations associated with interrupt transfers are reviewed below:
  - Following a burst flow control event ERDY notifies the host when the endpoint is ready to resume transactions. Unlike Bulk transfers the host must re-initiate burst transactions within two service intervals.
  - The host is not required to retry failed DATA packets during the same service interval.
  - Interrupt transactions are limited to a MaxBurstSize of three DATA packets.

Interrupt IN SuperSpeed Transaction Protocol

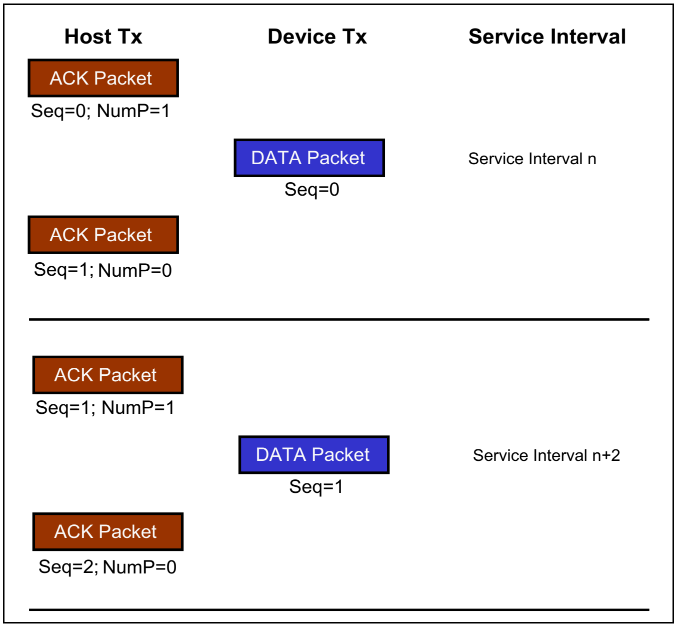
This section illustrates and discusses variations of the Interrupt IN protocol, including:

- Single Interrupt Transaction
- Burst transaction sequence (two DATA packet burst)
- Transaction with NRDY
- Transaction with eob
- Burst transaction with DATA packet error and Retry in same interval
- Burst transaction with DATA packet error and Retry in next interval

Single Interrupt Transaction

The example in Figure 8-1 illustrates an interrupt transactions that transfers data during every other µframe. As always, the host issues ACK packets to initiate IN transactions delivered to the target device. The device responds with a DATA header and payload (1024 Bytes) that are returned to the host. In turn, the host delivers an ACK packet to confirm the successful transaction.

Figure 8-1: Interrupt with Single DATA Packet Per Interval



9 Isochronous Protocol

Previous Chapter

Interrupt transactions provide periodic access to an endpoint and limit the amount of data that can be transferred during a bus interval. The previous chapter discusses the applications, capabilities, and behaviors associated with interrupt transfers.

This Chapter

Some devices require constant data delivery that is typically handled as synchronous transfers, such as the connection between an MP3 player and earplugs. However, the USB bus does not support synchronous transfers, so an alternate solution call isochrony is employed. The next chapter details the application, capabilities and characteristics of Isochronous transfers.

The Next Chapter

USB SuperSpeed hubs support both SuperSpeed and USB 2.0 operation. The next chapter covers the detailed operation of the SuperSpeed side of the hub, including the major hub responsibilities.

Introduction to Isochronous Transfers

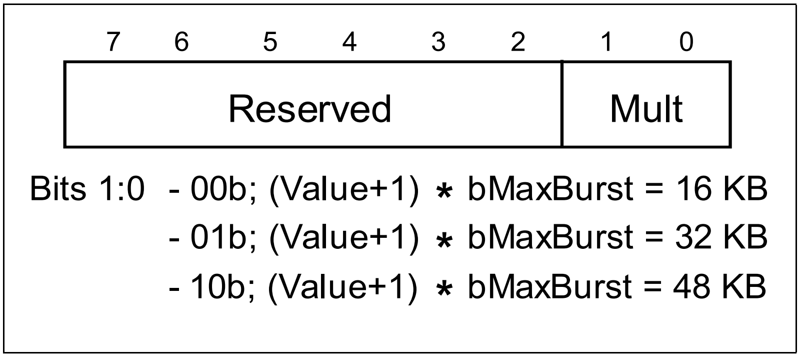
Isochronous transaction protocols implemented in the SS environment behave very similar to the USB 2.0 implementation. The primary characteristics of isochronous transfers are summarized in the following bullet list.

- Used by devices that require synchronous operation:
  - USB Headphones
  - Streaming Video, etc.

  Note: USB is not a synchronous bus but Isochrony aids hardware and software in managing synchronous delivery of data.
- Host provides guaranteed bandwidth and bounded latency per service interval.
- Periodic endpoints (isochronous and interrupt) are allocated up to 90% of the available bandwidth.

- Isochronous data delivery takes priority over the validity of data. Consequently, handshake packets are not defined for the Isochronous transfer protocol and no flow control or retries are allowed.
- The Endpoint descriptor defines the Maximum Packet Size (i.e., maximum size of a DATA payload). This value can vary depending on the Maximum Burst Size supported as defined below:
  - — If bMaxBurstSize = 0; bMaxPacketSize can be any value from 0-1024 bytes
  - — If bMaxBurstSize = 1-15; bMaxPacketSize must be 1024 bytes
- The Maximum Burst Size is defined in the Endpoint Companion descriptor as follows:
  - — 0 = endpoint supports burst of 1 DATA packet
  - — 1-15 = bursts of 2 to 16 DATA packets
- The Multiplier (Mult) field defines the Maximum number of Bursts supported during a service interval. The Mult value is located in the Endpoint Companion descriptor within the *bmAttributes* field (see Figure 9-1). The Mult field is zero based and has a maximum value of 2. The example shows the maximum burst size of 16 packets for each of the three multipliers.

Figure 9-1: Maximum Number of Bursts per Service Interval



Bits 1:0 - 00b; (Value+1) ∗ bMaxBurst = 16 KB
- 01b; (Value+1) ∗ bMaxBurst = 32 KB
- 10b; (Value+1) ∗ bMaxBurst = 48 KB

- The number of bytes an isochronous endpoint will transfer during a service interval is reported in the ***wBytesPerInterval*** field in the SS Endpoint Companion descriptor. This value is intended to facilitate Isochronous scheduling by reserving the required bus bandwidth.
- The ***binterval*** value for an isochronous endpoint defines the number of bus intervals contained within a single service interval. The *binterval* value is reported in the Endpoint descriptor as follows:
  - — The possible values range from 1 to 32,768, and expressed as: $2^{bInterval-1}$; where *bInterval* is a value from 1-16.
  - — Service interval duration ranges from 125µs – 4.096s (power's of 2).

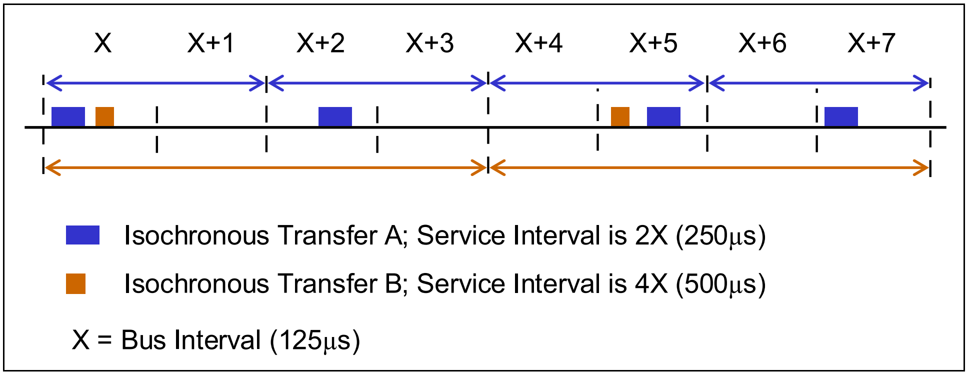
Bus Intervals and Isochronous Service Intervals

Some USB 2.0 isochronous endpoints maintain synchronization with the USB bus clocks as a means of delivering data at a constant rate (nominally 1KHz for full speed or 8KHz for high speed). This synchronization is maintained via the Start of Frame (SOF) or Micro Start of Frame (μSOF) packets that are broadcast at the beginning of every bus interval boundary. SOF packets also report the frame count that increments with each frame and rolls back to zero after 2048 frames.

SuperSpeed timing is also based on the same nominal 8KHz timing used in High-Speed USB. However, the SS bus uses the Isochronous Timestamp Packet (ITP) rather than μSOF packets to report the host bus interval timing.

Figure 9-2 depicts two isochronous transfers (A and B) with different service intervals. Transfer A uses a service interval duration of two bus intervals, while transfer B has a service interval spanning four bus intervals. Note in this example that during each service interval the same amount of data is being transferred. This indicates that the sample clocks associated with both transfer A and B are synchronous to the USB bus clock.

Figure 9-2: Isochronous Service Interval Examples


X
X+1
X+2
X+3
X+4
X+5
X+6
X+7
Isochronous Transfer A; Service Interval is 2X (250μs)
Isochronous Transfer B; Service Interval is 4X (500μs)
X = Bus Interval (125μs)

10 USB 3.0 Hubs

The Previous Chapter

Some devices require constant data delivery (sometimes called streaming) normally handled as synchronous transfers, such as the connection between an MP3 player and earplugs. However, the USB bus does not support synchronous transfers, so an alternate solution call isochrony is employed. The next chapter details the application, capabilities and characteristics of Isochronous transfers.

This Chapter

USB SuperSpeed hubs support both SuperSpeed and USB 2.0 operation. This chapter covers the detailed operation of the SuperSpeed side of the hub, including the major hub responsibilities.

The Next Chapter

In the three levels of USB 3.0 SuperSpeed protocol, Port-To-Port is the middle layer. This chapter is an overview of Port-To-Port topics covered in greater detail in subsequent chapters: transmitter and receiver header packet processing, CRC generation/checking, Link Commands, header packet flow control, and packet acknowledgement.

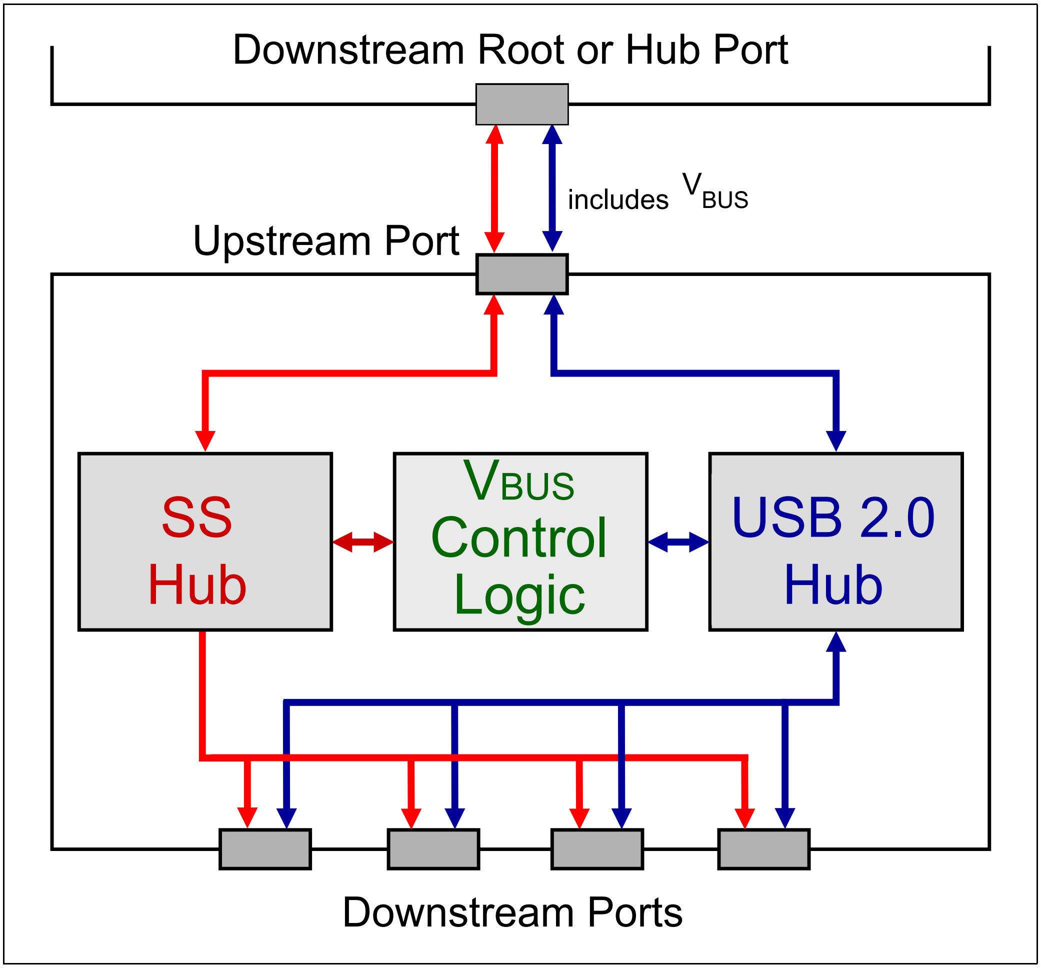
Introduction to SS Hubs

SuperSpeed hubs contain two separate hub functions: one for SuperSpeed operations and one for the USB 2.0 (i.e., Low-, Full- and High-Speed) operations. The USB 2.0 hub functions exactly as it did previously. (Please see MindShare's USB 2.0 book for details). Figure 10-1 on page 206 illustrates a SS Hub implementation showing the primary interconnects.

The upstream port of a hub attaches directly to a root port or to the downstream port of another hub. The upstream port must attach to both the SuperSpeed bus and the high-speed USB 2.0 bus when possible. The USB cable delivers V_{BUS} power that is shared by the SuperSpeed and USB 2.0 sides of the hub. V_{BUS} control logic may be required to manage potential port power conflicts if both SuperSpeed and USB 2.0 software attempt to change port power simultaneously or inappropriately.

Hub downstream ports provide the attachment points for connecting USB 2.0 devices (Low-, Full- and High-speed devices) and SuperSpeed devices. A description of hub upstream port and downstream port connections are discussed later in this chapter.

Figure 10-1: Basic Block Diagram of USB 3.0 Hub



Chapter 10: USB 3.0 Hubs

This chapter focuses on SuperSpeed hub functionality as listed below:

- Attachment to a downstream root or hub port.
- Responding to Software Requests including:
  - GetDescriptor (Device, Configuration, String, Hub, etc.)
  - SetAddress
  - SetConfiguration
  - SetPortPower
  - GetPortStatus, etc.
- Attachment/Detachment to upstream ports of device/hubs.
- Forwarding Packet in both directions.
- Managing packet routing in the downstream direction.
- Managing the link power hierarchy
- Transaction Deferral
- Hub Error Detection and Handling

Hub Attachment

Hubs like other USB devices cannot attach to a root or other hub port without V_{BUS} power being detected. This is true of both the USB 2.0 bus connection and the SuperSpeed connection. Once power is detected each bus will attempt to establish a connection to the downstream ports of the link partner.

Attachment to the SS and USB 2.0 buses are completely independent and attachment procedures are very different. Once the connections have been made, the buses remain completely independent. Also, when the hub connects to the USB 2.0 side only, SS Hub functions are disabled. Details associated with the USB 2.0 bus attachment can be found in MindShare's USB 2.0 book.

Packet Forwarding

One of the primary functions associated with Hubs is to forward packets in both directions while performing end-to-end transfers between the root port and target devices. Figure 10-2 on page 208 illustrates this requirement. Packets moving in the downstream direction must be routed by the hub to support the SS unicast routing mechanism. When a packet arrives at the upstream hub port the hub determines the destination of the incoming packet. As highlighted in the Figure 10-2 example. There are five possibilities:

- the transaction may be routed to one of the four downstream ports
- the hub itself may be the recipient (i.e., SS Hub Controller)

11 *Introduction to Port-To-Port Protocol*

Previous Chapter

USB 3.0 hubs support both SuperSpeed and USB 2.0 operation. The previous chapter covers the operation and major features of the SuperSpeed side of the hub.

This Chapter

In the three levels of USB 3.0 SuperSpeed protocol, Port-to-Port is the middle layer. This chapter is an overview of Port-to-Port topics covered in greater detail in subsequent chapters: transmitter and receiver header packet processing, CRC generation/checking, Link Commands, header packet flow control, and packet acknowledgement.

The Next Chapter

The next chapter describes the Link Training and Status State Machine (LTSSM) and the twelve SuperSpeed link states it supports. The LTSSM logic resides, conceptually, at the link layer and is used to reduce the traditional USB software burden related to managing link connectivity, power management, and testing. The responsibilities of the LTSSM and each of it's states and substates are summarized to provide background for later chapters that refer to the LTSSM's role during reset events, link training/recovery, power management transitions, etc.

Port-To-Port Protocol And The Link Layer

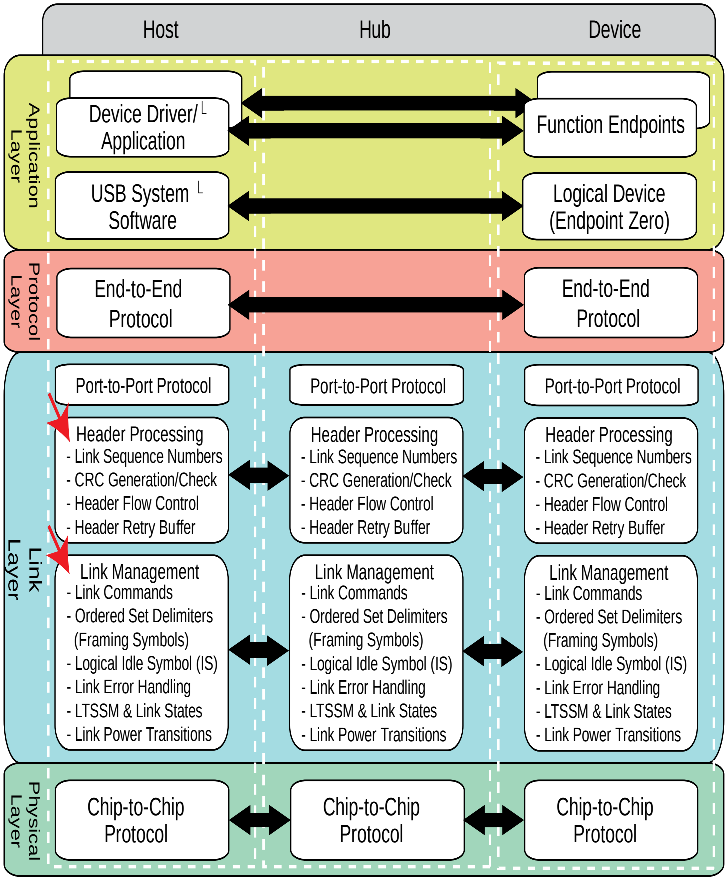
As traffic moves between link partners, the link layer enforces Port-to-Port protocol rules that assure that link integrity is maintained, header packets are processed and transported between link partners without error, link power management transitions occur transparently, etc.

The Big Picture Revisited

As indicated in Figure 11-1 on page 222, the link layer functions required of all host, hub, and peripheral devices to support Port-to-Port protocol fall into two groups:

- Header Processing
- Link Management

Figure 11-1: Port-To-Port Protocol And Link Layer Role



Port-To-Port Protocol: Header Processing

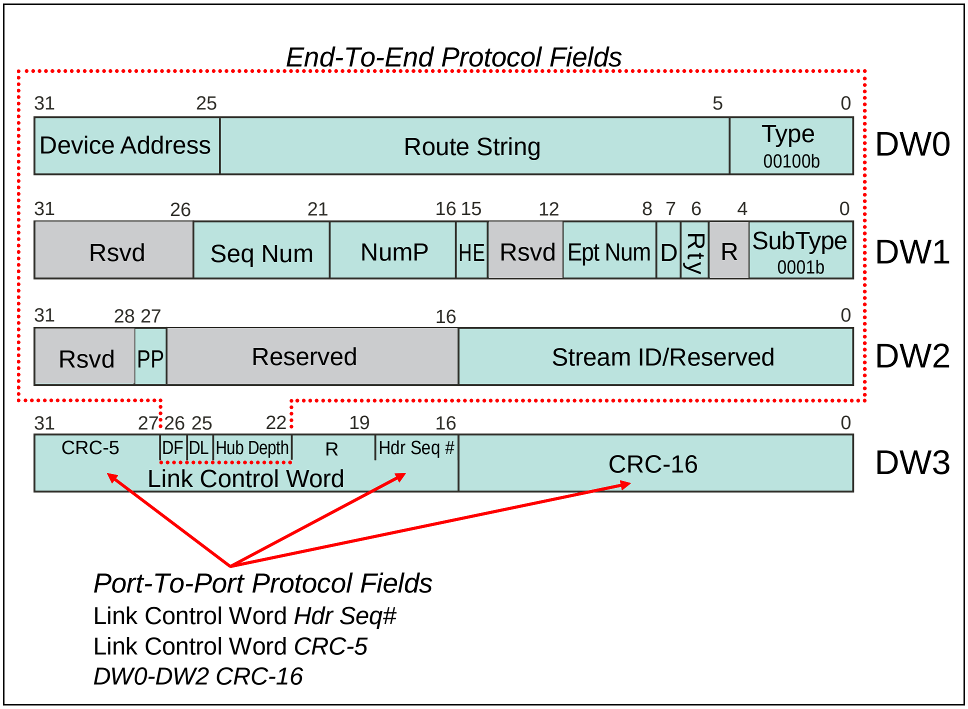
General

Headers are 16 byte structures that originate at the protocol layer of the transmitter and terminate at the protocol layer of the receiver. Some fields in the header are related to End-To-End protocol, other header fields are managed by the link layer and related to Port-to-Port protocol. As the headers pass through the link layer, CRC and header sequence numbers are generated by the transmitter and checked by the receiver to assure that no errors have occurred and that packets were received in the intended order.

Header Fields, Two Groups

DW3 of each protocol layer packet includes key fields associated with the Port-to-Port protocol; these fields are identified in Figure 11-2 on page 223. The other fields (within the dashed line) are related to the End-To-End protocol and are passed through unchanged by the link layer.

Figure 11-2: Link Layer Header Packet Processing, Key Fields



12 *LTSSM And the SuperSpeed Link States*

The Previous Chapter

In the three levels of SuperSpeed protocol, Port-to-Port is the middle layer. The USB 3.0 specification defines Port-to-Port protocol in terms of link layer responsibilities of transmitters and receivers. The previous chapter provided an overview of link layer responsibilities, including header processing, packet framing, flow control, header packet acknowledgement, and the use of link commands. All of these are described in more detail in subsequent chapters in the context of their use.

This Chapter

This chapter introduces the Link Training and Status State Machine (LTSSM) and the twelve SuperSpeed link states it supports. The responsibilities of the LTSSM and each of it's states and substates are summarized here to provide background for later chapters that refer to the LTSSM's role during reset events, link training/retraining, power management transitions, etc.

The Next Chapter

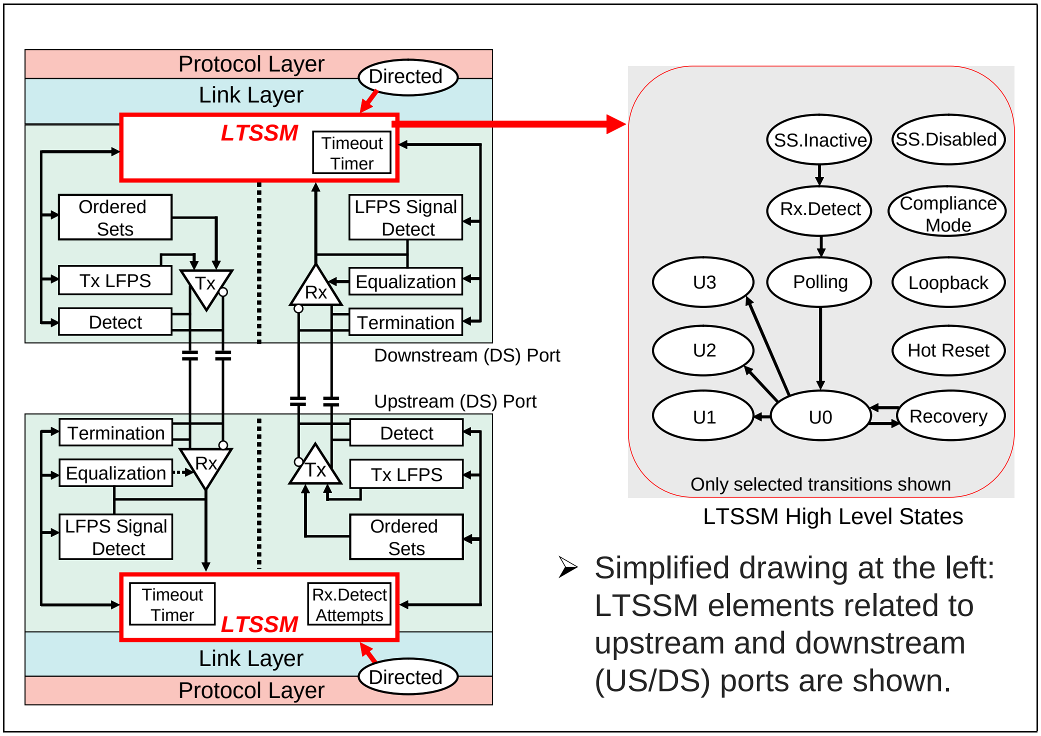
The next chapter describes the format and use of each SuperSpeed link command. Collectively, these eight-symbol commands are used for header packet flow control and packet acknowledgement, power management transition handshake, and for indicating continued presence in the U0 link state in the absence of other traffic.

Twelve High Level LTSSM States

As indicated in Figure 12-1, the Link Training and Status State Machine (LTSSM) resides at the link layer and supports twelve high level states. The USB 3.0 specification organizes the twelve high level LTSSM states into four functional groups:

- Operational States: *U0, U1, U2, U3*
- Link Initialization & Training States: *Rx.Detect, Polling, Hot Reset, Recovery*
- Testing States: *Compliance Mode, Loopback*
- Other States: *SS.Inactive, SS.Disabled*

Figure 12-1: Link Training And Status State Machine (LTSSM)



➢ Simplified drawing at the left: LTSSM elements related to upstream and downstream (US/DS) ports are shown.

Chapter 12: LTSSM And the SuperSpeed Link States

Why Is The LTSSM Needed?

USB has always relied on software and host controller hardware to manage the collection of simple hubs and peripherals that reside downstream. For USB 3.0 SuperSpeed, addition of the LTSSM helps reduce some of this burden by transferring many responsibilities to hardware. The LTSSM responsibilities include:

- Link training and retraining
- Link-level error handling
- Link power management
- Link test management

The following sections summarize major elements of each of these LTSSM responsibilities. All are covered in greater detail in other chapters.

LTSSM And Link Training And Retraining

Background: USB 2.0 Doesn’t Require Training

USB 2.0 does not specify a particular initialization sequence when preparing the physical layer for low speed (LS), full speed (FS), or High Speed (HS) operations. At attachment, the downstream device (upstream facing port) attaches a pull-up resistor to either the D- or D+ bus signals indicating to the hub it is attached to whether it is LS or FS/HS capable.

When software polls the hub and discovers the attachment, a bus reset is performed. During reset, a high speed device is permitted to negotiate a speed change from FS to HS using *chirp* protocol. If chirp is successful, HS terminations are switched in by the link partners. That’s it; the electrical specifications for the bus, cables, connectors, etc. assure that:

- No receiver equalization is required at any USB 2.0 speed
- No attempt is made to establish a continuous bit/symbol lock between link partner transmitter and receivers.
- At the start of a new packet, a special Start Of Packet (SOP) bit transition sequence is sent to re-establish lock between transmitter and receiver.
- At the end of each USB 2.0 packet, the bus returns to electrical idle.

LTSSM Coordinates SuperSpeed Link Training

A number of features of USB 3.0 make the use of the LTSSM essential in the

13 Link Commands

Previous Chapter

The previous chapter introduced the Link Training and Status State Machine (LTSSM) and the twelve SuperSpeed link states it supports. The LTSSM is used to reduce the traditional USB software burden related to managing link connectivity, power management, and testing. The responsibilities of the LTSSM and each of it's states and substates were summarized to provide background for subsequent chapters that refer to the LTSSM's role during reset events, link training/recovery, power management transitions, etc.

This Chapter

This chapter describes the format and use of each SuperSpeed link command. Collectively, these eight-symbol commands are used for header packet flow control, packet acknowledgement, power management transition handshake, and for indicating continued presence in the U0 link state in the absence of other traffic.

The Next Chapter

The next chapter presents a conceptual view of transmitter and receiver logic involved in link layer processing of outbound and inbound 16-byte headers required for each data packet (DP), isochronous timestamp packet (ITP), transaction packet (TP), and link management packet (LMP). Topics include generation and checking of link sequence number, header CRC-5/CRC-16, and the use of link layer header packet buffers holding a copy of each header packet until it is checked and acknowledged by the receiver.

Four Groups Of Link Commands

Unlike protocol layer End-To-End packets, link commands are part of the Port-to-Port protocol and used for communication between link partners; they are never routed to other links. They also are not subject to flow control and, when sent by the transmitter link layer, must be accepted by the receiver link layer. As indicated in Table 13-1 on page 290, there are four link command groups:

- Packet Acknowledgement — three types, sub-types vary with type. Two types are sent by the receiver link layer to inform the transmitter link layer whether or not a header packet was received without error. The third type is used by the transmitter link layer to inform the receiver that a header packet is being resent due to an earlier failure (also referred to as a retry).
- Flow Control — one type, four sub-types. Sent by the receiver to inform the transmitter each time a link layer header packet buffer becomes available.
- Power Management — one type, four sub-types. Exchanged by link partners during link power management transitions from U0 to U1, U2, or U3.
- Link Up/Link Down — one type, two sub-types. Periodically sent in the absence of other traffic to inform partner of continued presence in U0 state.

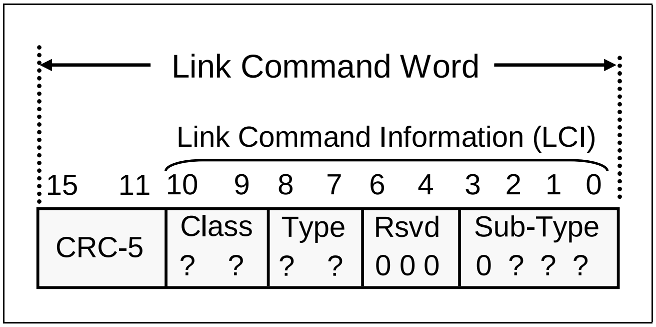
Table 13-1: Link Command Groups

Packet Category	Class Code	Type Codes	Sub-Types
Packet Acknowledgement	00b	00b, 02b, 3b	Varies with Type
Flow Control	00b	01b	4
Power Management	01b	01b	4
Link Up/Link Down	10b	10b	2

As depicted in Figure 13-1 on page 290, the basic content of a link command is contained in a 16-bit Link Command Word structure consisting of:

- Link Command Information (LCI) in bits 10:0. Note the three valid LCI fields: Class, Type, and Sub-Type
- A CRC-5 field in bits 15:11 (protecting the LCI)

Figure 13-1: Link Command Word Fields



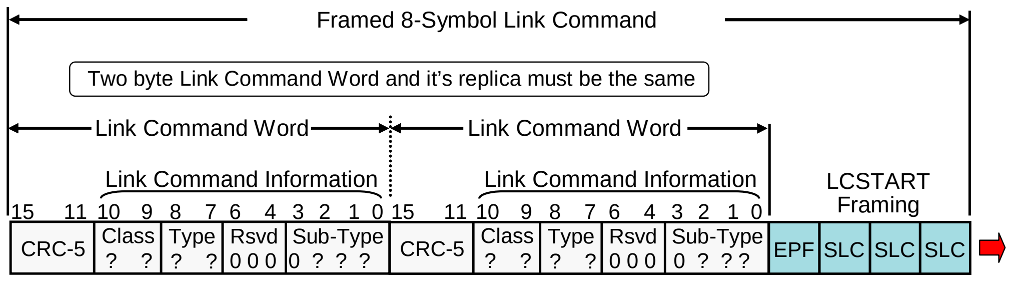
Link Commands On The SuperSpeed Link

As described previously, a link command contains a 16-bit Link Command Word consisting of the 11-bit Link Command Information (LCI) field and the 5-bit CRC. To assure that link command information is valid at the receiver, two things are done by the transmitter:

- The 16-bit (2 byte) Link Command Word is replicated--each with its own LCI and CRC-5. The two Link Command Word copies follow one another.
- The link command is preceded with the 4 “K” symbol LCSTART framing for a total of 8 symbols.

At the receiver link layer, both copies of the Link Command Word are verified to be identical before the link command is processed. Figure 13-2 on page 291 illustrates a framed link command on the SuperSpeed link.

Figure 13-2: Framed Link Command On The SuperSpeed Link



For Additional Details On Link Commands

Additional details on the use of link commands may be found in other chapters:

- Refer to Chapter 24, entitled "SuperSpeed Power Management," on page 563 for additional details on LGO_Ux, LAU, LXU, LPMA Link commands.
- Refer to Chapter 15, entitled "Header Packet Flow Control," on page 321 for details on LCRD_x link commands
- Refer to Chapter 16, entitled "Link Errors & Packet Acknowledgement," on page 339 for details on LGOOD_n, LBAD, and LRTY link commands

14 Header Packet Processing

The Previous Chapter

The previous chapter described the format and use of each SuperSpeed link command. Collectively, these eight-symbol commands are used for header packet flow control and packet acknowledgement, power management transition handshake, and for indicating continued presence in the U0 link state in the absence of other traffic.

This Chapter

This chapter presents a conceptual view of transmitter and receiver logic involved in link layer processing of outbound and inbound 16-byte headers required for each data packet (DP), isochronous timestamp packet (ITP), transaction packet (TP), and link management packet (LMP). Topics include generation and checking of link sequence number, header CRC-5/CRC-16, and the use of link layer header packet buffers holding a copy of each header packet until it is checked and acknowledged by the receiver.

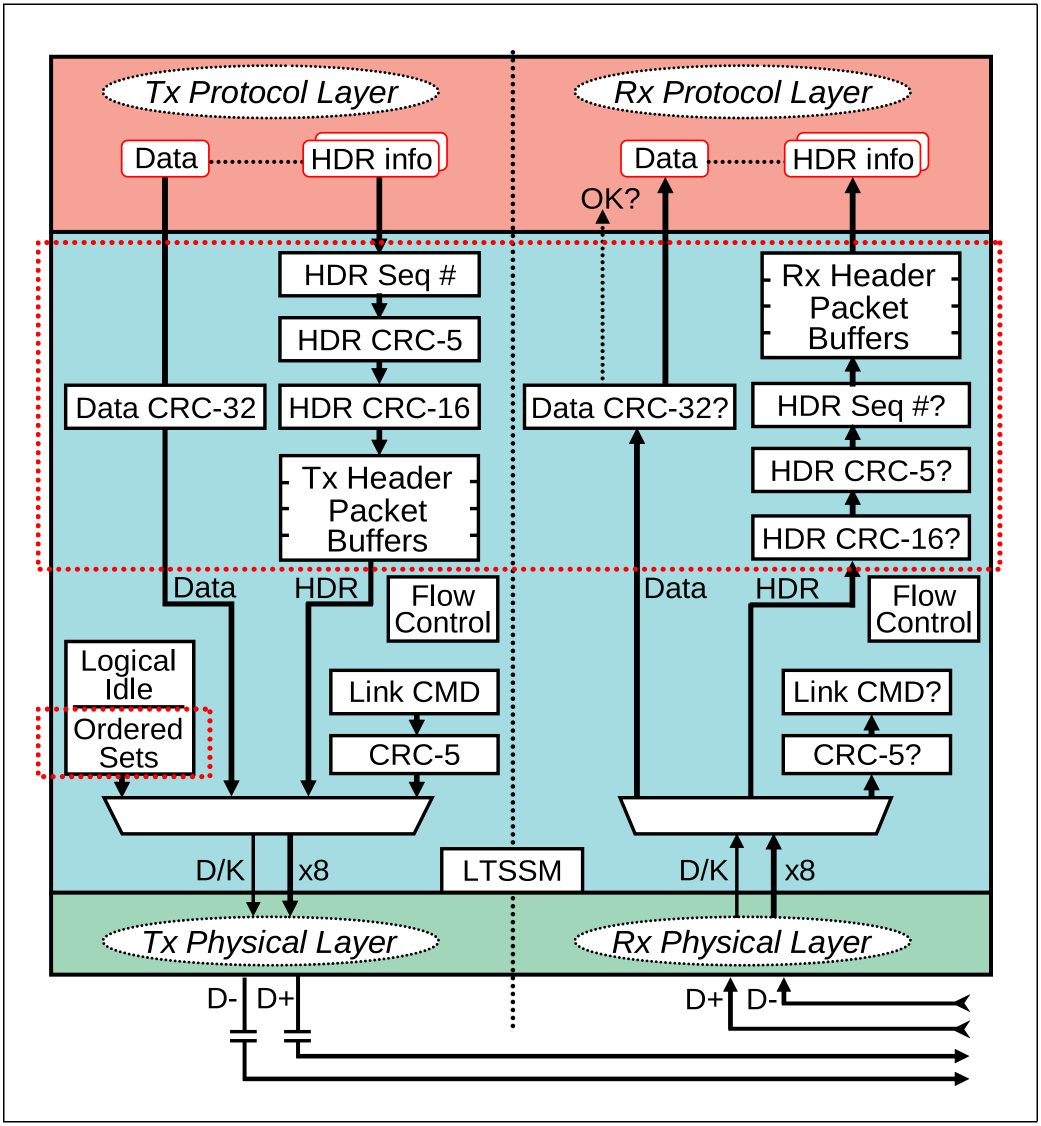
The Next Chapter

SuperSpeed USB employs link level header packet flow control to assure that header packet transmission is never attempted if link layer buffer space is unavailable at the receiver. The next chapter describes the motivation for link layer header packet flow control and a conceptual view of the buffers, credit counters, timers, and link commands required to support it.

Link Layer Packet Processing Role

While the link layer has a wide range of Port-to-Port protocol responsibilities, the focus of this chapter is on the link layer transmitter and receiver roles in header and data packet processing. The highlighted regions in Figure 14-1 on page 304 indicate the scope of link layer topics covered here.

Figure 14-1: Link Layer Tx And Rx Packet Processing Scope

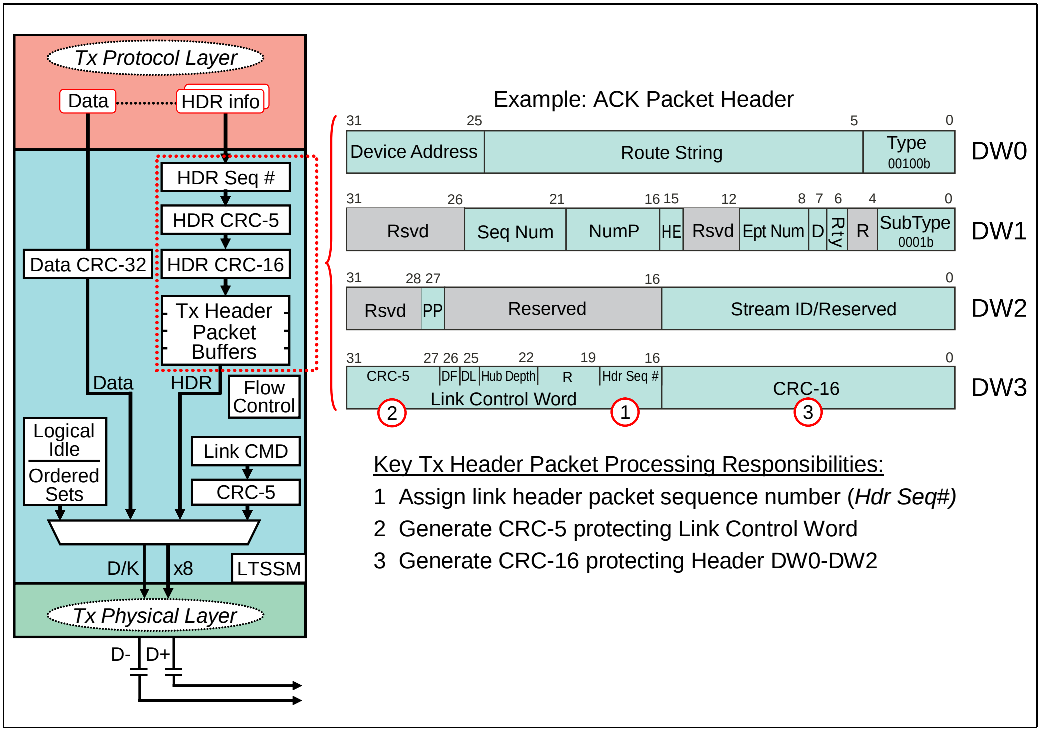


Transmitter Packet Processing

The transmitter link layer processes outbound header and data packets that originate at the device protocol layer. Each header for a transaction packet (TP), link management packet (LMP), isochronous timestamp packet (ITP), and data packet (DPH) arrives at the transmitter (Tx) link layer as a partially completed 16-byte structure. The Tx link layer performs the final processing, including assignment of a link sequence number and generation of two CRCs to protect fields within the header. In cases when a data packet is to be sent, an additional CRC-32 will generated and appended to the data packet payload (DPP) to protect it.

Before looking at the details, Figure 14-2 on page 305 provides a quick review of header packet format and the key Tx link layer header packet processing responsibilities. In the example, an ACK transaction packet has arrived at the link layer for final processing buffering before transmission to the link partner.

Figure 14-2: Key Transmitter Packet Processing Functional Blocks



Key Tx Header Packet Processing Responsibilities:

1 Assign link header packet sequence number (*Hdr Seq#)*

2 Generate CRC-5 protecting Link Control Word

3 Generate CRC-16 protecting Header DW0-DW2

15 Header Packet Flow Control

The Previous Chapter

The previous chapter presented a conceptual view of transmitter and receiver logic involved in link layer processing of outbound and inbound 16-byte headers required for each data packet (DP), isochronous timestamp packet (ITP), transaction packet (TP), and link management packet (LMP). Topics included generation and checking of link sequence number, header CRC-5/CRC-16, and the use of link layer header packet buffers holding a copy of each header packet until it is checked and acknowledged by the receiver.

This Chapter

SuperSpeed USB employs link level header packet flow control to assure that header packet transmission is never attempted if link layer buffer space is unavailable at the receiver. This chapter describes the motivation for link layer header packet flow control and a conceptual view of the buffers, credit counters, timers, and link commands required to support it.

The Next Chapter

The next chapter summarizes common SuperSpeed link errors encountered during transmission of header packets, data packets, and link commands. It also describes one of the primary components of link error handling, link level header packet acknowledgement and, if necessary, retransmission (referred to as a *retry)*. Topics covered include the use transmitter and receiver counters, timers, header CRC and link sequence number generation/checking, and three link commands used in the packet acknowledgement handshake.

Background: Host And Device Flow Control

Generally speaking, flow control enables an initiator or target device to pause data transfers or other traffic, usually because of temporary buffer full/empty conditions. Flow control by an initiator is often simpler than for the target because it can withhold new requests until it is ready.

USB 2.0 Flow Control, Very Limited

A number of features associated with the previous generation USB 2.0 severely limit flow control opportunities for low speed, full speed, and high speed peripheral devices and hubs:

- The simple master-slave broadcast bus requires that all transactions must be initiated by the host controller under software control.
- The USB 2.0 physical layer (PHY) is built on a half-duplex interface which renders upstream asynchronous messages, for purposes such as device initiated flow control, impossible. This also means that no DMA or peer-to-peer transactions may be initiated.
- While a USB host controller is capable of generating CPU interrupts, there is no support for peripheral interrupts in any generation of USB.

Given these limitations, the best USB 2.0 can offer in the way of flow control is an endpoint polling scheme in which the host provides the target device endpoint an opportunity to return a NAK handshake response to an IN or OUT token. The NAK indicates that the endpoint is not ready to start (or continue) with data transactions. When this occurs in USB 2.0, considerable bandwidth may be wasted because of transaction re-attempts triggered not by errors, but because of IN endpoint buffer empty, or OUT endpoint buffer full conditions.

The SuperSpeed Flow Control Approach

End To End Flow Control

While USB 3.0 SuperSpeed protocol is still based on a token-data-handshake model, it includes several enhancements that collectively provide far better end to end flow control for both host and devices:

- The USB 2.0 NAK response is replaced with the USB 3.0 SuperSpeed NRDY/ERDY responses. In this scheme, once the device returns a NRDY packet indicating is not ready to start (or continue) data transfers, the host controller will suspend service to the endpoint until the device later sends an ERDY packet (this eliminates the repeated NAKs seen in USB 2.0)
- For OUT endpoint flow control, each ACK header packet returned by the device endpoint includes a *NumP* (number of additional packets) field indicating its readiness to continue accepting data.
- For IN endpoint flow control, each data packet header returned by the device includes the *NumP* field in the DPH (data packet header) indicating

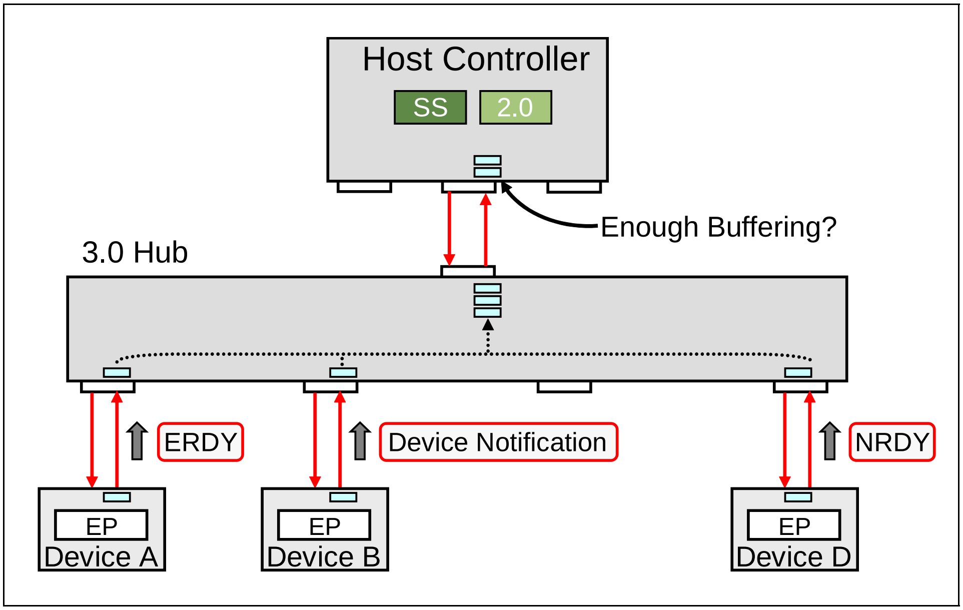
its readiness to continue sending data. As long as the host and endpoint are both ready and willing to proceed, data transfers can continue.

Link Level Flow Control Is Also Needed

USB 2.0 makes no provision for link layer flow control because the master-slave broadcast bus and half-duplex link means that there can only be one transaction outstanding at a time beneath the host controller and no possibility that packets could be moving upstream and downstream at the same time. The host controller and current target endpoint simply take over all parts of the topology until the transaction is complete.

With the USB 3.0 SuperSpeed dual-simplex links, host packets and asynchronous upstream device packets may occur at the same time. A simple example is depicted in Figure 15-1 on page 323. Here, Device A,B, and D attached to a USB 3.0 hub are each sending asynchronous transaction packets upstream. On its upstream port, the hub must forward the three packets to the host controller. The question of whether the downstream facing port of the host controller can accept all of them at the highest possible rate depends on the number of receiver buffers currently available. The remainder of this chapter describes the USB 3.0 solution to this type of link level flow control problem.

Figure 15-1: Asynchronous Traffic May Contribute To Link Flow Control Problems



16 Link Errors & Packet Acknowledgement

The Previous Chapter

SuperSpeed USB employs link level header packet flow control to assure that header packet transmission is never attempted if link layer buffer space is unavailable at the receiver. The previous chapter describes the motivation for link layer header packet flow control and a conceptual view of the buffers, credit counters, timers, and link commands required to support it.

This Chapter

This chapter summarizes common SuperSpeed link errors encountered during transmission of header packets, data packets, and link commands. It also describes one of the primary components of link error handling, link level header packet acknowledgement and, if necessary, retransmission (referred to as a *retry)*. Topics covered include the use transmitter and receiver counters, timers, header CRC and link sequence number generation/checking, and three link commands used in the packet acknowledgement handshake.

The Next Chapter

In the three levels of SuperSpeed protocol, Chip-To-Chip resides below End-To-End and Port-to-Port protocols. The USB 3.0 specification defines Chip-to-Chip protocol in terms of physical (PHY) layer responsibilities of the transmitter and receiver in each link partner. The next chapter provides an overview of the Chip-to-Chip protocol and two key groups of physical layer responsibilities: PHY logical functions and the electrical signaling requirements. Both of these are described in more detail in subsequent chapters.

Background: USB 2.0 Error Handling

Previous generation USB 2.0 relies heavily on host controller time-outs and transaction rescheduling to deal with bus level errors. Even if bit errors are fairly infrequent, when they do occur a considerable amount of potential bandwidth is wasted waiting for timeout requirements to be met before correcting problems and moving on to other transactions. Placing the burden for error handling and most other aspects of USB 2.0 protocol on software and the host controller reduces the complexity and cost of USB hubs and peripheral devices, but increases the overall latency in moving packets when errors occur.

USB 3.0 SuperSpeed Requires A New Approach

A number of factors helped drive the move away from the traditional USB 2.0 model of host based error detection and correction towards link-level error detection and handling. A few of the key factors include the following.

SuperSpeed Signaling Affects Bit Error Rates

- Moving from USB 2.0 480 Mb/s High Speed to USB 3.0 5 Gb/s Superspeed results in a much narrower signal eye. This complicates receiver clock and data recovery (CDR) and increases the likelihood of bit error rates above the SuperSpeed target bit error rate (BER) of 10^{-12}.
- SuperSpeed transmitter spread-spectrum clocking (SSC), needed to reduce EMI, also adds to the complexity of clock recovery at the receiver.
- Frequency-dependent losses when transmitting at 5 Gb/s further degrade signal quality and the ability of a SuperSpeed receiver to discriminate between logic levels. A usable signal eye may not be recoverable without aggressive receiver equalization.
- Transmitter de-emphasis (3.5 dB), required to reduce inter-symbol interference (ISI) at the receiver, also has the general negative effect of lowering signal levels at the receiver.

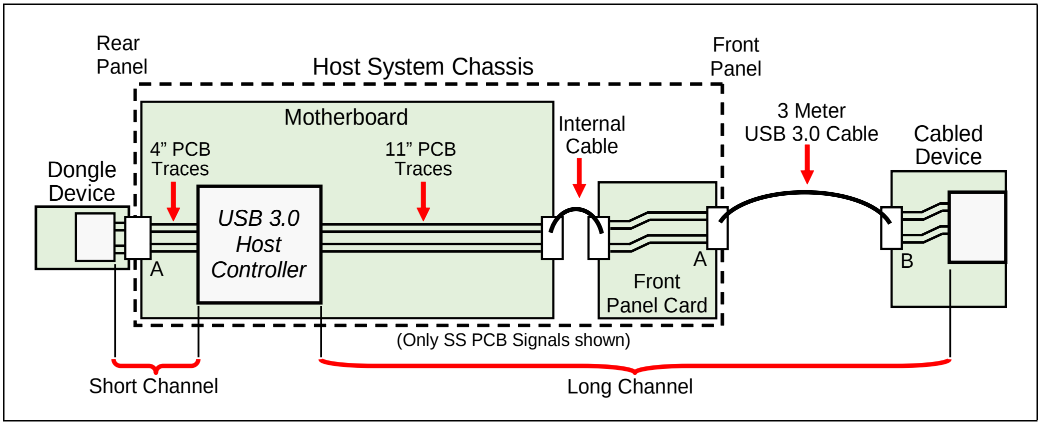
Complex USB Topologies A Challenge At 5 Gb/s

Over the years, as USB added new speeds, users have maintained high expectations regarding plug-and-play, backward compatibility, and low cost. One of the biggest challenges of USB 3.0 SuperSpeed design is preserving this user experience while assuring reliable 5 Gb/s operations in the wide range of USB topologies encountered in systems ranging from mobile devices to PCs, workstations, and servers.

Figure 16-1 on page 341 depicts some extremes of physical connections that are possible in the point-to-point, USB cabled bus topology (the example shown is based on a rack-mounted server chassis). The USB 3.0 specification refers to connection extremes as *short channels* and *long channels*.

- Short channel example. As indicated in Figure 16-1, the "dongle" device attached at the left is directly connected to the host controller with 4 inch printed circuit board (PCB) traces.
- Long channel example. The device at the right of Figure 16-1 is at the end of a very long channel that includes lengthy PCB traces, two cables, and multiple connectors.

Figure 16-1: Link Errors Are Affected By The Channel



While both devices in Figure 16-1 may have been designed with great care, real-world differences in USB topologies can, and do, impact signal quality (and the bit error rate) at SuperSpeed receivers.

Upstream Asynchronous Message Errors

Another motivation for moving away from the host-based error handling employed in USB 2.0 involves upstream message traffic. While device initiated upstream traffic is not possible with the half-duplex USB 2.0 physical connections, the USB 3.0 SuperSpeed link is full-duplex and devices are permitted to send asynchronous protocol layer packets to gain the attention of the host.

17 Introduction to Chip-To-Chip Protocol

Previous Chapter

The previous chapter summarizes common SuperSpeed link errors encountered during transmission of header packets, data packets, and link commands. It also described one of the primary components of link error handling, link level header packet acknowledgement and, if necessary, retransmission (referred to as a *retry)*. Topics covered included the use transmitter and receiver counters, timers, header CRC and link sequence number generation/checking, and three link commands used in the packet acknowledgement handshake.

This Chapter

In the three levels of SuperSpeed protocol, Chip-to-Chip resides below End-To-End and Port-to-Port protocols. The USB 3.0 specification defines Chip-to-Chip protocol in terms of physical (PHY) layer responsibilities of the transmitter and receiver in each link partner. This chapter provides an overview of the Chip-to-Chip protocol and two key groups of physical layer responsibilities: PHY logical functions and the electrical signaling requirements. Both of these are described in more detail in subsequent chapters.

The Next Chapter

The next chapter describes logical functions performed by physical layer transmitters and receivers during 5 Gb/s SuperSpeed operations. For the transmitter, a description of hardware scrambling, 8b/10b encoding, and serialization is provided. For the receiver, equalization, clock/data recovery, symbol lock, elastic buffer, 8b/10b decoding, and descrambling are covered.

Chip-To-Chip Protocol And The Physical Layer

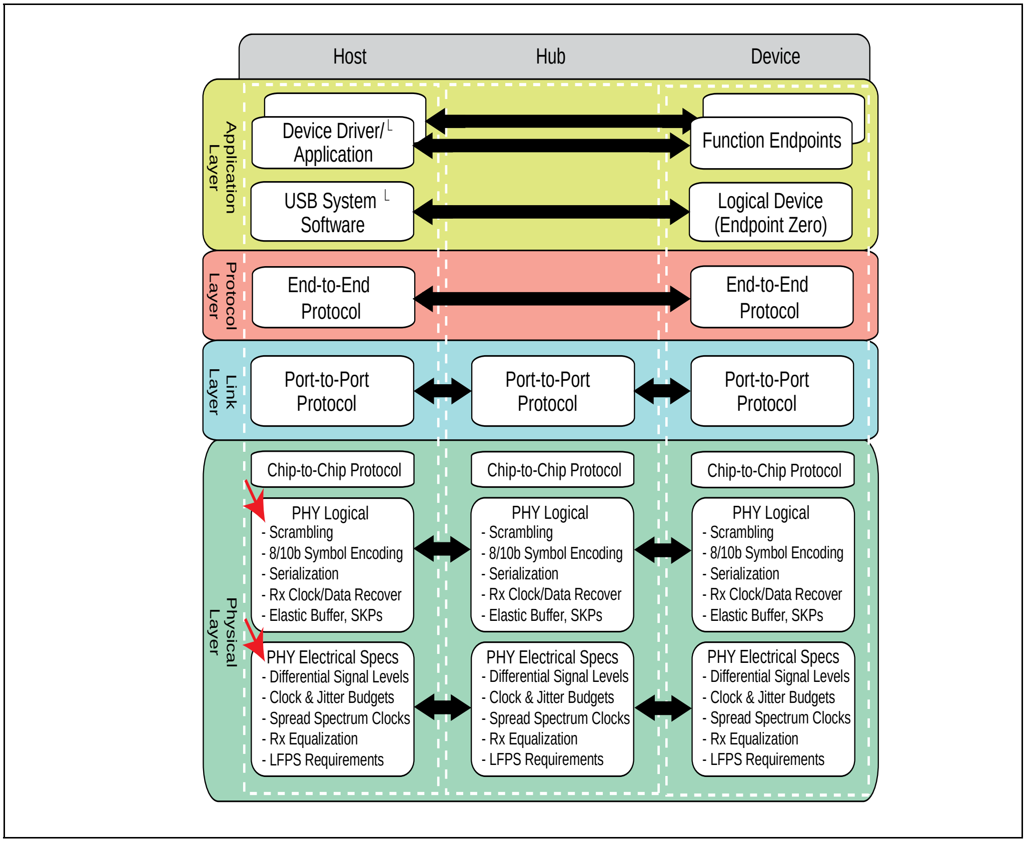
As traffic moves between link partners, the physical layer enforces Chip-to-Chip protocol rules that assure that packet, link command, ordered set, and logical idle symbols are processed and transported across the 5 Gb/s SuperSpeed link without error. In addition, the physical layer employs Chip-to-Chip protocol low frequency periodic signaling for a number of cases when SuperSpeed operations are not possible on the link.

The Big Protocol Picture, Once More

As indicated in Figure 17-1 on page 374, the physical layer functions required of all host, hub, and peripheral devices to support Chip-to-Chip protocol fall into two groups:

- Physical layer (PHY) Logical Processing
- Physical layer (PHY) Electrical Specifications

Figure 17-1: Chip-To-Chip Protocol And Physical Layer Role



Chip-To-Chip Protocol: PHY Logical Processing

As the header packets, data packets, and link commands move from the link layer to the SuperSpeed link, transmitter physical layer (PHY) hardware performs logical processing on each data and control byte in a series of pipelined stages including scrambling, 8b/10b encoding, and serialization before shifting each bit onto the 5 Gb/s SuperSpeed link with its differential driver.

At the link partner receiver, similar logical processing is performed in the reverse order to recover each data and control byte as well as the packet or link command structure to which it belongs.

A Note About D/K Bytes And Symbols

The term “data byte” is used loosely in the context of physical layer logical processing. It includes any byte associated with a header packet, data packet, or link command that is not a framing symbol. It also includes logical idle (D0.0) as well as any data (D) bytes within the link training TSEQ, TS1, and TS2 ordered sets. The distinction between data (D) and control (K) bytes is important because physical layer logical processing is different in the two cases.

Another distinction work noting here is the use of the terms “byte” and “symbol”. In this book, the difference is:

- A byte is an 8-bit value before it is encoded by the transmitter; depending on where in the logical processing sequence it is, the byte may or may not be scrambled. At the receiver, a byte is an 8-bit value after it is decoded; again, it may or may not have been de-scrambled.
- A symbol is the 10-bit value assigned to a byte by the 8b/10b encoder at the transmitter. The encoded values for data (D) and control (K) symbols are mutually exclusive; a D/K flag bit accompanying each byte into the encoder assures that the 10-bit encoding belongs to either the data symbol group or the control symbol group.

18 Physical Layer Logical Functions

The Previous Chapter

In the three levels of SuperSpeed protocol, Chip-to-Chip resides below End-To-End and Port-to-Port protocols. The USB 3.0 specification defines Chip-to-Chip protocol in terms of physical (PHY) layer responsibilities of the transmitter and receiver in each link partner. The chapter provides an overview of the Chip-to-Chip protocol and two key groups of physical layer responsibilities: PHY logical functions and the electrical signaling requirements.

This Chapter

This chapter describes logical functions performed by physical layer transmitters and receivers during 5Gb/s SuperSpeed operations. For the transmitter, a description of hardware scrambling, 8b/10b encoding, and serialization is provided. For the receiver, equalization, clock/data recovery, symbol lock, elastic buffer, 8b/10b decoding, and descrambling are covered.

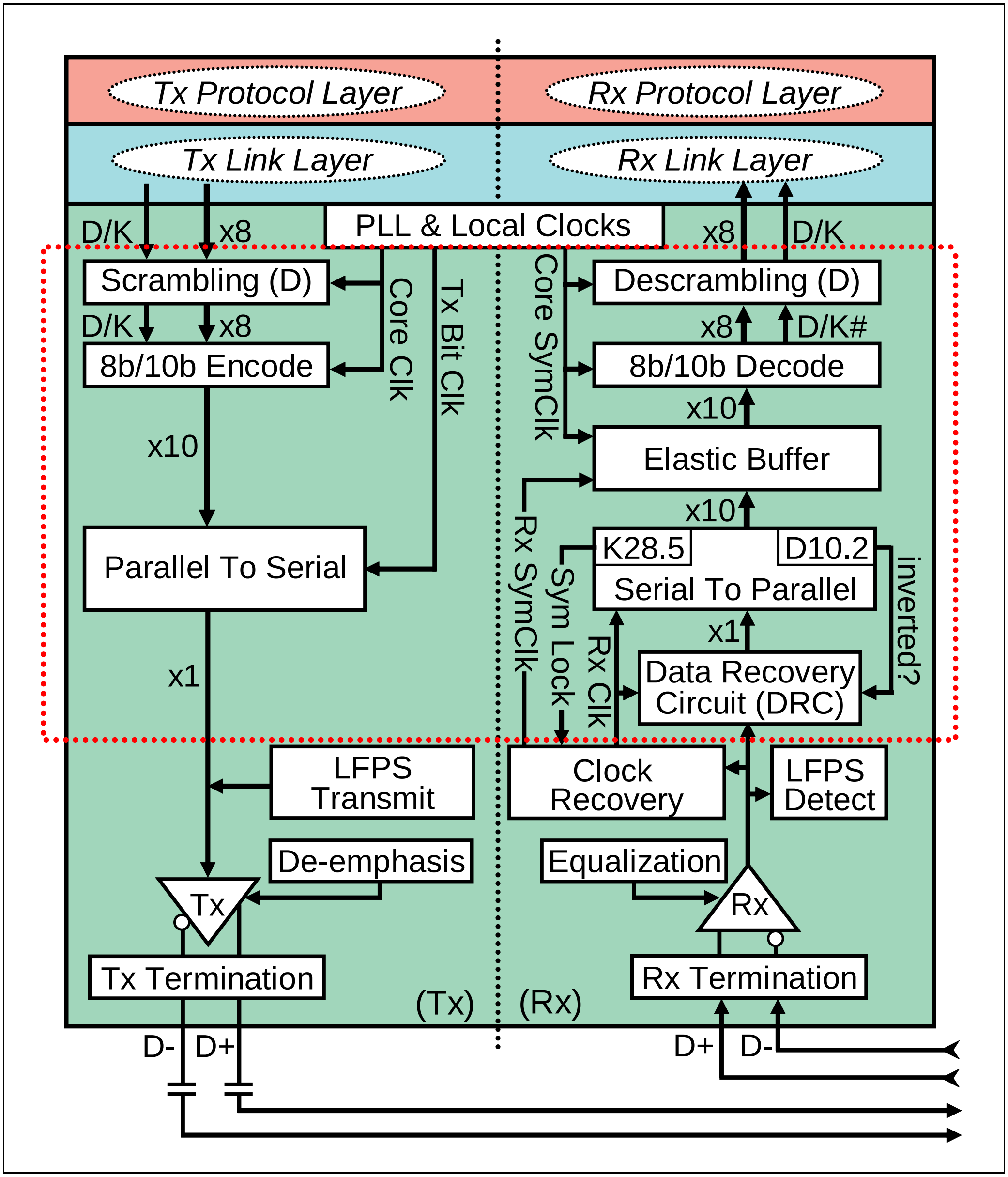
The Next Chapter

The next chapter describes three reset types defined in USB SuperSpeed: PowerOn Reset, Warm Reset, and Hot Reset. PowerOn Reset occurs automatically when V_{BUS} link power transitions to a valid voltage level. Warm Reset and Hot Reset are referred to as *inband* resets, occurring when software directs a downstream facing port to assert reset signaling to its attached device.

Physical Layer Logic Definitions

The SuperSpeed physical layer (PHY) consists of a transmitter and receiver residing in the same package and supporting two sets of unidirectional AC-coupled differential signal pairs. The physical layer can be subdivided into logical and electrical signaling sections. This chapter covers the logical section of the PHY shown in the highlighted region of Figure 18-1 on page 384. In the illustration, the transmit (Tx) PHY logical section is on the left and the receive (Rx) PHY logical section on the right.

Figure 18-1: Physical Layer Tx And Rx Logical Section



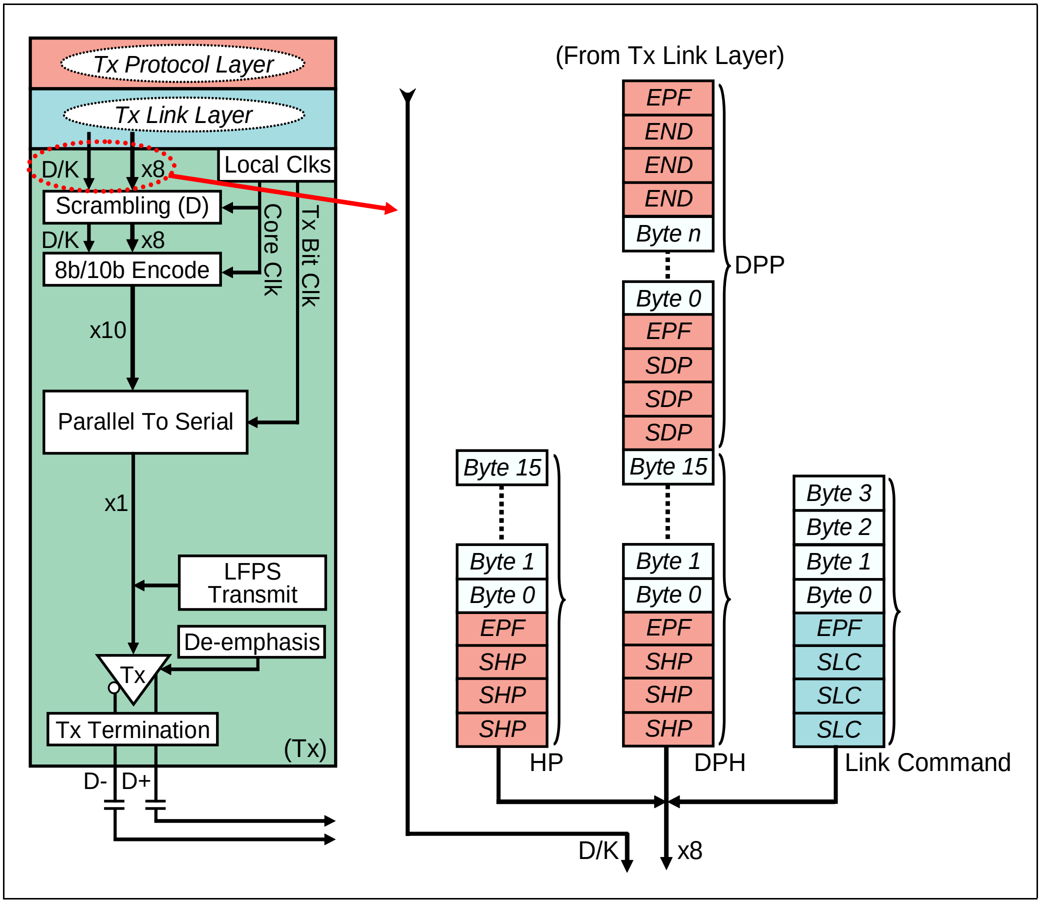
Tx Physical Layer (PHY) Logic

Anytime a link is operating SuperSpeed, The Tx physical layer logic continually processes outbound packets, link commands, and ordered sets. In the absence of all other traffic, logical idle symbols are sent in order to maintain reliable clock/data recovery at the receiver of the link partner.

Tx Outbound Bytes And The D/K Flag

The first step in Tx PHY logic processing is shown in Figure 18-2 on page 385.

Figure 18-2: Transmitter Outbound Bytes And The D/K Flag



19 *SuperSpeed Reset Events*

The Previous Chapter

The previous chapter described logical functions performed by physical layer transmitters and receivers during 5 Gb/s SuperSpeed operations. For the transmitter, a description of hardware scrambling, 8b/10b encoding, and serialization was provided. For the receiver, equalization, clock/data recovery, symbol lock, elastic buffer, 8b/10b decoding, and descrambling was covered.

This Chapter

This chapter describes three reset types defined in USB SuperSpeed: PowerOn Reset, Warm Reset, and Hot Reset. PowerOn Reset occurs automatically when V_{BUS} link power transitions to a valid voltage level. Warm Reset and Hot Reset are referred to as *inband* resets, occurring when software directs a downstream facing port to assert reset signaling to its attached device. The methods used by software to invoke resets and the impact on device state are also covered.

The Next Chapter

The next chapter covers SuperSpeed link training and port initialization, a process that starts automatically following a PowerOn or Warm Reset. Chapter topics focus on device responsibilities and the handshake sequence required to transition a link from Rx.Detect, through Polling, and into U0. The initialization of device ports upon entry into U0 through the exchange of Header Sequence Number and Rx Header Buffer Credit Advertisement link commands is also discussed.

PowerOn Reset

PowerOn Reset occurs automatically each time a device attached to a root or external hub port detects a V_{BUS} transition to a valid voltage level (a minimum of 4.0 V). Upstream facing ports monitor V_{BUS} continuously and are required to perform an internal PowerOn Reset each time V_{BUS} transitions from invalid to valid--even if the device is self powered.

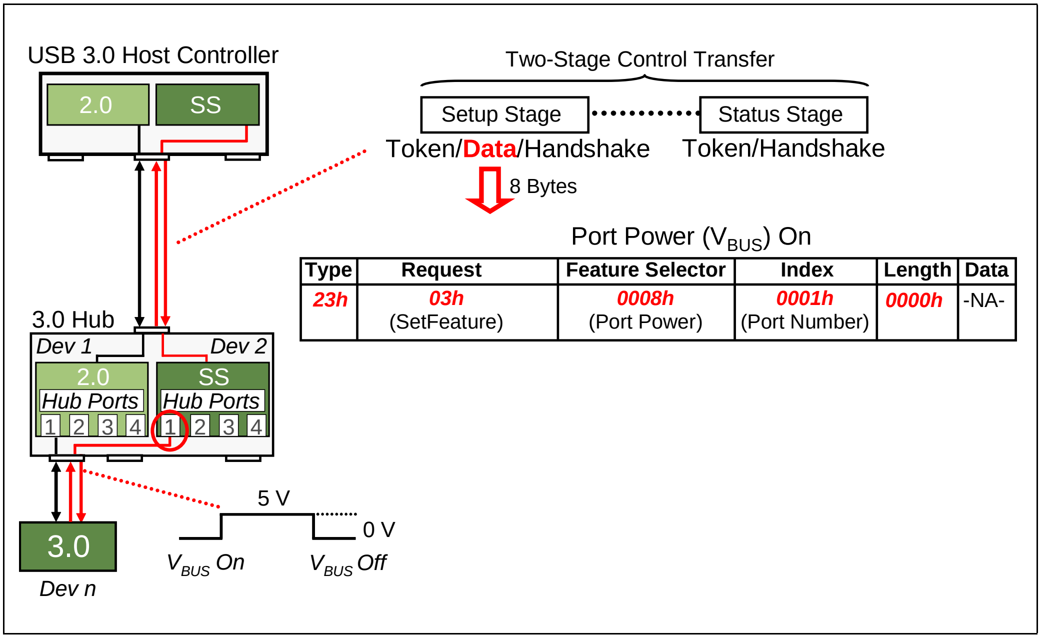
Software May Enable And Disable V_{BUS}

In addition to the PowerOn Reset that occurs automatically each time a device detects a V_{BUS} transition to a valid voltage, software may selectively disable or enable V_{BUS} while a device remains physically attached. For external hubs, this is done by targeting the default control endpoint (EP0) of the hub with the appropriate hub-class request to enable or disable port power.

In the example shown in Figure 19-1 on page 414, software has caused the hub-class request *SetFeature (PortPower)* to be sent. The 8-byte Setup stage of the request includes the target hub device address and port number (Device 2, Port 1 in this case). Once the Control Transfer Setup and Status stages complete successfully, the hub port performs the requested action and enables V_{BUS}.

As shown in the example, V_{BUS} transitions to a valid level and PowerOn Reset commences within the device. This causes a fundamental reset of all internal USB state and device and partners prepare for SuperSpeed link training.

Figure 19-1: Enabling V_{BUS} Triggers PowerOn Reset



Type	Request	Feature Selector	Index	Length	Data
23h	03h (SetFeature)	0008h (Port Power)	0001h (Port Number)	0000h	-NA-

PowerOn Reset Impact On Device State

Detection of PowerOn Reset returns the device USB interface to its initial state. Memory, registers, and other storage elements related to USB Protocol, Link, and Physical layers are restored to default values:

- USB device address returns to 0 and any previously selected configuration/ interface information is lost.
- Link layer Tx and Rx header packet buffers, counters, timers are cleared
- Once V_{BUS} is again valid and PowerOn Reset completes, a USB 3.0 device that detects SuperSpeed low impedance receiver terminations for its link partner may draw up to 150mA of V_{BUS} current (one unit load). Otherwise, it may only draw up to 100mA of V_{BUS} current as required in the USB 2.0 specification.
- If a SuperSpeed link partner was detected, the Link Training and Status State Machine (LTSSM) transitions to the Rx.Detect state and prepares for link training. The LTSSM states were introduced in the last chapter.

Note that each device is required to handle its own internal timing when initializing itself during PowerOn Reset.

PowerOn Reset And Self-Powered Devices

Bus-powered (aka cable-powered) devices are completely powered down when V_{BUS} is removed. Full initialization is assured because all USB and other hardware logic is powered up each time V_{BUS} is reapplied. In addition, bus-powered devices present a high impedance receiver termination while power is removed, assuring that they are “not visible” to downstream facing hub ports until the port is again powered.

Because self-powered devices have a local supply derived from an AC or DC power source, they are capable of maintaining some logical state even while V_{BUS} is off. To help assure consistent USB PowerOn Reset behavior between bus-powered and self-powered devices, the USB 3.0 specification includes some requirements for self-powered devices during and after a PowerOn Reset:

The following rules apply while PowerOn Reset is active (V_{BUS} is invalid):

- Receiver termination must be in the high-impedance state. This means that when a device detects PowerOn Reset is active, it must remove its receiver low impedance termination and appear as “not visible” on the SuperSpeed link. High impedance termination, $Z_{RX\text{-}HIGH\text{-}IMP\text{-}DC\text{-}POS,}$ must be 25 KΩs or more.

20 *Link Training*

The Previous Chapter

The previous chapter described the three reset types defined in USB SuperSpeed: PowerOn Reset, Warm Reset, and Hot Reset. PowerOn Reset occurs automatically when V_{BUS} link power transitions to a valid voltage level. Warm Reset and Hot Reset are referred to as *inband* resets; they occur when software directs a hub downstream port to assert reset signaling to its attached device. The methods used by software to invoke resets and the impact on device state for each reset type were also covered.

This Chapter

This chapter covers SuperSpeed link training and port initialization, a process that starts automatically following a PowerOn or Warm Reset. Chapter topics focus on device responsibilities and the handshake sequence required to transition a link through LTSSM *Rx.Detect* and *Polling* states, and into *U0*. The initialization of device ports upon entry into U0 through the exchange of Header Sequence Number and Header Buffer Credit Advertisement Link Commands, as well as Port Capability/Configuration LMPs, is also discussed.

The Next Chapter

The next chapter describes link recovery & retraining, a low-latency alternative to full link training, but only used for links which have previously been in the U0 state. A transition to LTSSM Recovery state followed by retraining is employed each time an exit is signaled from U1, U2, or U3 power management states or to recover from link errors which could not be corrected while in U0. Chapter topics include the impact of retraining on device state and the handshake signaling required to enter and exit LTSSM Recovery state.

Link Training Required Before SuperSpeed Operation

Before SuperSpeed links are enabled for 5 Gb/s transfers, an initialization process called link training is required to establish a reliable connection between the physical layer (PHY) transmitters and receivers for each pair of link partners. In cases when external hubs are present, all links in the path between a host controller *root hub port* and a target device at any level in the topology must have completed link training before any End-to-End SuperSpeed packets can be exchanged. Full link training occurs automatically, on a link by link basis, each time PowerOn Reset or Warm Reset is initiated by a downstream facing port or detected by an upstream facing port.

USB Physical Connections Complicate Link Training

USB is found in systems ranging from embedded platforms to PCs, workstations, and servers. In addition to the usual high speed serial bus concerns related to signal quality and clock/data recovery, SuperSpeed USB 3.0 presents some additional challenges. It is an IO bus that can be easily extended with hubs and permits a wide range of physical connections between devices, including combinations of cables, connectors, and chip-to-chip pc board traces. The goal of link training is to assure that a reliable 5 Gb/s signal is delivered by transmitters to receivers across the full range of permitted physical interfaces.

USB 3.0 refers to the connection between link partners as the *channel.* A channel is often described as long or short, and the term has both physical and electrical implications. Because of variations in motherboard layout, location of USB connectors, and the presence of user supplied cables, hubs, and peripherals, most real-world systems have a mix of long and short channels.

Figure 20-1: An Example Server System With Short And Long USB 3.0 Channels

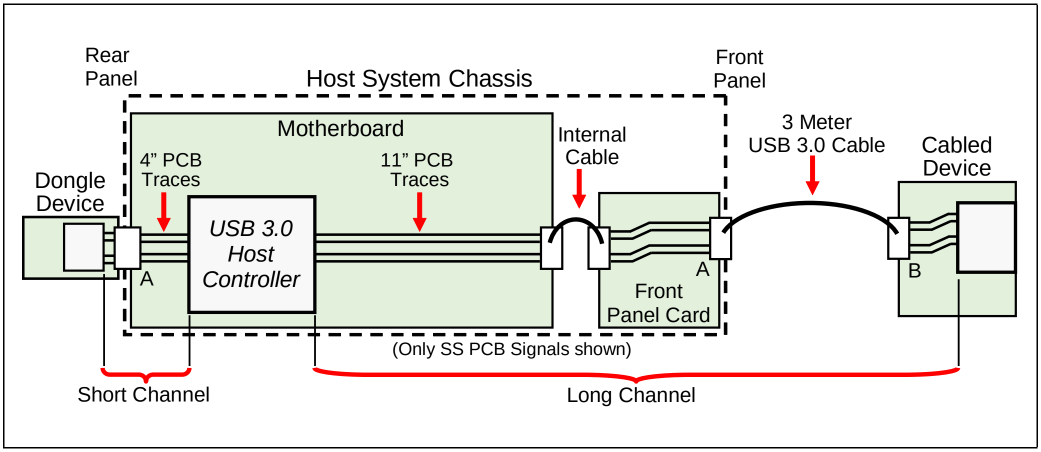


Figure 20-1 on page 428 illustrates long and short channel extremes which may be encountered in systems such as rack-mounted servers. In the server example shown, USB 3.0 connectors are exposed at both the back and front of the chassis. A brief discussion of the long and short channel depicted in the example system follows.

The Long Channel

The device at the right in Figure 20-1 on page 428 is connected to the front panel of the server system chassis by a 3 meter USB 3.0 cable. A front panel pc card inside the chassis then connects to the motherboard using an internal cable. Because of the location of the USB 3.0 host controller on the motherboard in this example, 11” of PCB traces are then required to attach to the host controller and the internal connector. In a long channel such as this, which includes multiple cables and connectors, frequency-dependent losses at SuperSpeed rates result in poor signal quality at the receiver and a *signal eye* that may or may not be recoverable.

The Short Channel

By contrast, the USB “dongle” device at the left in Figure 20-1 on page 428 is directly attached to a rear panel motherboard USB 3.0 connector and routed to the USB 3.0 Host Controller using only 4” of PCB trace length. In a short channel such as this, signal losses are less of a problem than signal reflections.

SuperSpeed Link Training Is Adaptive

Designers of USB peripherals and hubs generally have no idea of the actual topologies where the device will be used. Because of hot plug, a device may be swapped between long and short channels without warning. For the most part, the same uncertainty applies to the host controller. While the motherboard layout establishes fixed connections between host controller root hub ports and USB connectors, the topology beyond the connector is unknown and may be changed by the user at any time).

The variability in USB channel characteristics is handled automatically by requiring that full link training is performed if any of the following occurs:

- V_{BUS} transitions from invalid to valid (4.0 V minimum). This typically occurs at attachment or when hub is commanded to apply port power
- On any Warm Reset invoked by software
- In the event of certain LTSSM timeout conditions

21 *Link Recovery and Retraining*

The Previous Chapter

The previous chapter covered SuperSpeed link training and port initialization, a process that starts automatically following a PowerOn or Warm Reset. Chapter topics included device responsibilities and the handshake sequence required to transition a link through LTSSM *Rx.Detect* and *Polling* states, and into *U0*. The initialization of device ports upon entry into U0 through the exchange of Header Sequence Number and Header Buffer Credit Advertisement Link Commands, as well as Port Capability/Configuration LMPs, was also discussed.

This Chapter

This chapter covers link recovery & retraining, a lower-latency alternative to full link training for links that have previously been in the U0 state. Transitions to recovery occur for a variety of reasons, primarily related to related to power management exits and the handling of serious link errors. Chapter topics include the impact of retraining on device state as well the handshake signaling required to enter and exit recovery.

The Next Chapter

The next chapter discusses the detection and configuration of USB devices that are attached to a USB port of any speed. Device descriptors and other characteristics and features that relate to configuring a device are also discussed.

Reasons For Link Recovery And Retraining

There are a number of events which may cause a SuperSpeed link transition to the Recovery LTSSM state. These events fall into the three groups listed here and described in the sections that follow:

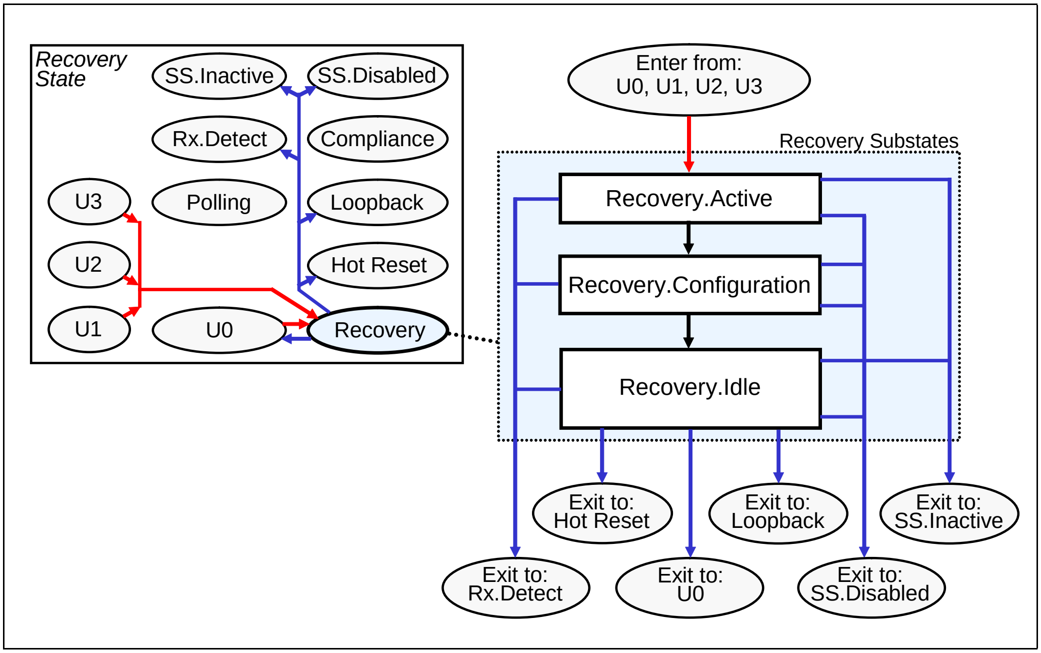
- Recovery transitions required on each return to U0 from link power management states U1, U2, or U3. The link must exit electrical idle and enter Recovery to retrain the physical layer (PHY) for 5 Gb/s operations.

- Recovery transitions required to correct link layer or physical layer error conditions which can't be repaired while the U0 state. In this case, the transition to LTSSM Recovery is used to exchange training sequences, confirm receiver "bit lock" and "symbol lock", and re-initialize elastic buffer levels. If the sequence is successful, then a return to U0 is negotiated followed by the re-advertisement of link layer flow control credits and header sequence number information. Various LTSSM Recovery time-outs handle the cases where a link partner is unresponsive; if so, the link may transition to the inactive state (SS.Inactive).
- Transitions to LTSSM Recovery as an intermediate step in a subsequent transition to Hot Reset, Loopback mode, or to disable link data scrambling.

Recovery And Retraining Managed By The LTSSM

Figure 21-1 on page 468 depicts the twelve high-level states of the USB 3.0 SuperSpeed Link Training and Status State Machine (LTSSM). Recovery is one of the high level LTSSM states and its relationship to the other high level states can be seen at the left in the illustration.

Figure 21-1: LTSSM Recovery State, Substates, Entry, And Exits



On the right of Figure 21-1 on page 468 is a more detailed illustration showing the three LTSSM Recovery substates as well as the entry path from U0, U1, U2, or U3 states. After entering LTSSM Recovery, there are six possible exit paths to other states--depending on the reason for entering LTSSM Recovery, the response of the link partner, time-outs, etc. The exit paths are:

- To Rx.Detect state if a timeout or other condition requires full link training to be performed again.
- To the SS.Inactive state if the link is non-operational.
- To the Loopback state for bit error rate (BER) testing
- To the Hot Reset state to force initialization of most device USB 3.0 logic to the default state.
- To the SS.Disabled state for ports not permitted to operate at SuperSpeed
- To the U0 state for a return to normal 5 Gb/s SuperSpeed link operations

Recovery Is Speedy

As mentioned previously, a transition to LTSSM Recovery as a way to re-establish reliable 5 Gb/s transport is extremely fast when compared to full link training. The primary reason for this is because link partners entering LTSSM Recovery are required to retain (and apply) receiver equalization parameters established earlier during the last full link training.

Receiver equalizer training is, by far, the most time-consuming part of the full link training sequence. During this process, devices exchange 64K TSEQ ordered sets, each 32 symbols in length. On the 5 Gb/s SuperSpeed serial link, this requires approximately 4.2 ms. If this process was repeated for each exit from a U1 or U2 link power management state, much of the benefit associated with link level power management would be lost. Instead, receiver equalizers are trained and parameters are reused in all cases except errors or other conditions requiring full link training to be repeated.

The following discussion describes link recovery events related to the two most common reasons for entering the LTSSM Recovery state: exits from link power management states U1-U3 and handling link layer errors that could not be corrected with three attempts at header packet retry. Note that the handshake required to enter LTSSM Recovery is different in the two cases and is described in the following sections. On the other hand, events within the LTSSM Recovery substates are much the same for both cases.

22 Device Configuration

The Previous Chapter

The previous chapter covered link recovery & retraining, a lower-latency alternative to full link training for links that have previously been in the U0 state. Transitions to recovery occur for a variety of reasons, primarily related to related to power management exits and the handling of serious link errors. Chapter topics include the impact of retraining on device state as well the handshake signaling required to enter and exit recovery.

This Chapter

This chapter discusses the detection and configuration of USB devices that are attached to any USB port. The process is virtually the same for devices of any speed. Device descriptors and other characteristics and features that relate to configuring a device are also discussed.

The Next Chapter

Hub devices must be configured like any other device attached to a USB port. Hub configuration differs from other devices in that it involves reporting whether or not other devices are attached to the downstream ports. The next chapter reviews the hub configuration process and discusses the role of the hub client driver in performing the setup and initialization for normal hub operation.

Overview

Please note that many of the actions associated with Device configuration must be performed exactly as described herein, while other operations can be performed in a variety of ways.

During system initialization, all devices (including hubs) will be detected, configured and initialized. Following configuration and initialization, devices may be removed or new devices may be connected. It is the hub client software that checks USB hub ports to detect device attachment or removal. If a status change results from a new device having been attached, the configuration process is triggered for that device. Similarly when device removal is detected resources associated with the device are released.

This chapter discusses the primary operations performed during a typical configuration sequence. SuperSpeed device configuration is typically performed one device at a time, similar to the USB 2.0 approach. Configuration begins with detection of an attached device and ends with the device having been configured and initialized. The primary operations include:

- Device Detection and Reporting
- Assigning a unique Device address
- Determining Device Capabilities (via Device Descriptors)
- Reporting User Information (via String Descriptors)
- Configuring Device (via SetConfiguration Request)
- Locating and loading associated Client (Device Class) Software
- Client-specific device initialization

Standard Device Requests

Almost all of the Configuration process involves Device Requests and are in the form of control transfers. The setup transaction of a Device Request includes the 8 bytes of data that specifies the action to be performed. Table 22-1 lists the Standard Requests including the setup data. The last column (Data) indicates the type of data being transfered to or from the device, resulting in a 3-stage control transfer. When “None” is reported a 2-stage control transfer is performed. (See “Control Transfer Structures and Examples” on page 106 to review the control transfer process.)

Table 22-1: Standard Device Requests

Request Type	Request	Value (2 bytes)	Index (2 bytes)	Length (2 bytes)	Data
100000000B 100000001B 100000010B	GetStatus (00h)	Zero	Zero Interface Endpoint	Two	Device Interface, or Endpoint Status
00000000B 00000001B 00000010B	ClearFeature (01h)	Feature Selector	Zero Interface Endpoint	Zero	None

Table 22-1: Standard Device Requests (Continued)

Request Type	Request	Value (2 bytes)	Index (2 bytes)	Length (2 bytes)	Data
00000000B 00000001B 00000010B	SetFeature (03h)	Feature Selector	Zero Interface Endpoint	Zero	None
00000000B	SetAddress (05h)	Device Address	Zero	Zero	None
10000000B	GetDescriptor (06h)	Descriptor Type and Descriptor Index	Zero or Language ID	Descriptor Length	Descriptor
00000000B	SetDescriptor (07h)	Descriptor Type and Descriptor Index	Zero or Language ID	Descriptor Length	Descriptor
10000000B	GetConfiguration (08h)	Zero	Zero	One	Configuration Value
00000000B	SetConfiguration (09h)	Configuration Value	Zero	Zero	None
100000001B	GetInterface (10h)	Zero	Interface	One	Alternate Interface
00000001B	SetInterface (11h)	Alternate Setting	Interface	Zero	None
10000010B	SyncFrame (12h)	Zero	Endpoint	Two	Frame Number
00000000B	SetSEL (48h)	Zero	Zero	Six	Exit Latency Values
00000000B	SetIsochDelay (49)	Delay in ns	Zero	Zero	None

23 *SuperSpeed Hub Configuration*

The Previous Chapter

The previous chapter discussed the configuration of USB devices that are attached to any USB port. The process is virtually the same for devices of any speed. Device descriptors and other characteristics and features that relate to configuring the device were also detailed and discussed.

This Chapter

Hub devices are configured like any other device attached to a USB port. Hub configuration differs in that it involves reporting whether or not other devices are attached to the downstream ports. This chapter reviews the hub configuration process with the focus on the issues related to extending the bus through the hub's downstream facing ports.

The Next Chapter

The SuperSpeed bus implements a high-speed serial bus that requires constant transmission of information (logical idle) to ensure the link between devices are ready to deliver packets with very low latency. However, this constant transmission of logical idle creates constant power consumption. Consequently, the SuperSpeed bus is architected to manage link power aggressively during logical idle by entering an electrical idled state. It must also be able to recover back to the operational state with relatively low latencies. The next chapter describes the various features and mechanisms used to reduce power consumption.

Configuring the Hub

Hubs are the only USB device types that connect to both the SuperSpeed bus and the USB 2.0 bus. This discussion focuses only on the SuperSpeed bus and hub configuration. The USB 2.0 configuration is covered in MindShare's USB 2.0 book.

Hubs must be configured like any other device. When a hub is attached to a root or other hub port software does not know the nature of the device until the device descriptors are read. These descriptors reveal to software the device's function, or class. Consequently, hubs go through the same process as any other device during the initial stages of configuration. This process consists of the following events and includes the associated device states. The device in the following bullet list is a hub.

- Device is attached to hub port (Attached State)
- Power is applied to device (Powered State)
- Power triggers device Reset Assertion and Deassertion
- Attachment detected and Link Training is performed (Default State)
- Software assigns device Address (Address State)
- Software determines device characteristics via Descriptors
- Software reports user information via String descriptors
- Device features are enabled
- Device is Configured (Configured State)

This chapter focuses on the hardware and software interactions associated with a root hub port detecting a device (a hub in this case) and software configuring it for normal operation. Once configuration software has configured a device, class-specific software completes the initialization. From a hub perspective this means preparing hub features and downstream ports for normal operation.

Standard Device Requests

Almost all of the configuration process involves Device Requests performed via control transfers. For reference purposes, Table 23-1 lists the standard requests including the 8 bytes of setup data that defines each request. The shaded rows indicate the request is not supported by Hubs.

Table 23-1: Standard Device Requests

Request Type	Request	Value (2 bytes)	Index (2 bytes)	Length (2 bytes)	Data
100000000B 100000001B 100000010B	GetStatus (00h)	Zero	Zero Interface Endpoint	Two	Device Interface, or Endpoint Status
00000000B 00000001B 00000010B	ClearFeature (01h)	Feature Selector	Zero Interface Endpoint	Zero	None
00000000B 00000001B 00000010B	SetFeature (03h)	Feature Selector	Zero Interface Endpoint	Zero	None

Table 23-1: Standard Device Requests (Continued)

Request Type	Request	Value (2 bytes)	Index (2 bytes)	Length (2 bytes)	Data
00000000B	SetAddress (05h)	Device Address	Zero	Zero	None
10000000B	GetDescriptor (06h)	Descriptor Type and Descriptor Index	Zero or Language ID	Descriptor Length	Descriptor
00000000B	SetDescriptor (07h)	Descriptor Type and Descriptor Index	Zero or Language ID	Descriptor Length	Descriptor
10000000B	GetConfiguration (08h)	Zero	Zero	One	Configuration Value
00000000B	SetConfiguration (09h)	Configuration Value	Zero	Zero	None
100000001B	**GetInterface (10h)**	**Zero**	**Interface**	**One**	**Alternate Interface**
00000001B	**SetInterface (11h)**	**Alternate Setting**	**Interface**	**Zero**	**None**
10000010B	SyncFrame (12h)	Zero	Endpoint	Two	Frame Number
00000000B	SetSel (48h)	Zero	Zero	Six	Exit Latency Values
00000000B	SetIsochDelay (49)	Delay in ns	Zero	Zero	None

Device Detection and Reporting

General

Note: The reader may prefer to skip this section because it is virtually identical to the discussion covered in the previous chapter. This discussion is included here in the event the previous chapter was skipped. See "The Hub Configuration Process" on page 534 to skip this section.

24 *SuperSpeed Power Management*

The Previous Chapter

Hub devices are configured like any other device attached to a USB port. Hub configuration differs in that it involves reporting whether or not other devices are attached to the downstream ports. The previous chapter reviews the hub configuration process with the focus on the issues related to extending the bus through the hub's downstream facing ports.

This Chapter

The SuperSpeed bus implements a high-speed serial bus that requires constant transmission of information (logical idle) to ensure the link between devices are ready to deliver packets with very low latency. However, this constant transmission of logical idle creates constant power consumption. Consequently, the SuperSpeed bus is architected to manage link power aggressively during logical idle by entering an electrical idled state. It must also be able to recover back to the operational state with relatively low latencies. This chapter describes the various features and mechanisms used to reduce power consumption.

The Next Chapter

The next chapter discusses the SuperSpeed link electrical interface and signaling. Major topics include SuperSpeed clocks and transmitter/receiver electrical specifications as well as Low Frequency Periodic Signaling (LFPS).

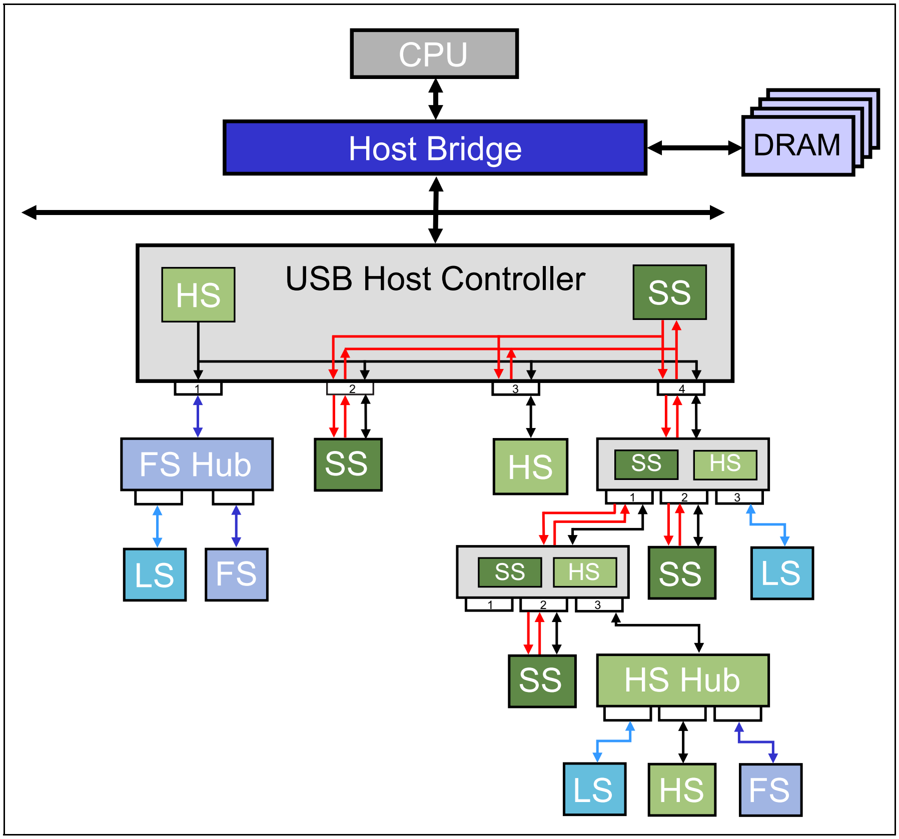
Principles of SuperSpeed Power Management

This section provides background information, concepts and principles associated with Link Power Management. Later chapter discuss the various mechanisms defined by the specification for managing link and device power.

Background

The primary goal of SuperSpeed bus power management is to conserve as much power as possible without adversely affecting responsiveness and performance. It is obvious that battery-powered systems require the greatest emphasis on power savings, but with today's high energy costs, systems of all types come under scrutiny. Like many high-speed serial bus implementations (e.g., PCI Express, Serial Attached SCSI, and InfiniBand), SuperSpeed USB can support a large number of links in the topology, and these high-speed links natively require constant transmission to maintain synchronization needed for low-latency transmission of packets. This means that every SuperSpeed link in the topology continuously consumes power if there is no method for placing these links into low power states. See Figure 24-1.

Figure 24-1: All SS Links Remain Powered Without Link Power Management



Power Management Design Goals

This section focuses on three features associated with SuperSpeed power management provide improved power management over the USB 2.0 solutions. These features include:

1. Link Power Management
2. Improved Suspend/Resume
3. Function Power Management.

Link Power Management

The SuperSpeed bus includes features that allow aggressive link power management by placing links into low-power states as often as possible and for the longest duration possible without significantly impacting performance. Clearly, this approach involves trade-offs between conserving power and maintaining short latencies. The approach to conserving link power may depend upon platform characteristics. Two extremes come to mind:

- Server platforms requiring low latencies and high performance may favor performance over power conservation.
- Battery powered systems may manage power very aggressively, thereby favoring power savings over low latencies and high performance.

Link power management relies on bus idle time, during which links can be placed into low-power states and then recover quickly when a packet must be delivered. Several features introduced by SuperSpeed USB help to create more bus idle time, including:

- Unicast transactions — only links between the root port and target device are active, while all other links remain idle.
- Fast completion of transactions (5 Gb/s) — higher transmission rates means the links are active for shorter periods of time.
- More efficient transaction protocols — protocol improvements lead to more bus idle time.
- Improved end-to-end flow control — the SuperSpeed bus is not polled. Instead devices notify the host when they are ready to be accessed, thereby reducing bus traffic and increasing idle time.

The USB 3.0 specification defines the mechanisms used to trigger link transitions to the low power states and are discussed later in this chapter.

25 *SuperSpeed Signaling Requirements*

The Previous Chapter

The previous chapter described link and platform level power enhancements introduced in USB 3.0.

This Chapter

This chapter discusses the SuperSpeed link electrical interface and signaling. Major topics include SuperSpeed clocks and transmitter/receiver electrical specifications as well as Low Frequency Periodic Signaling (LFPS).

The Next Chapter

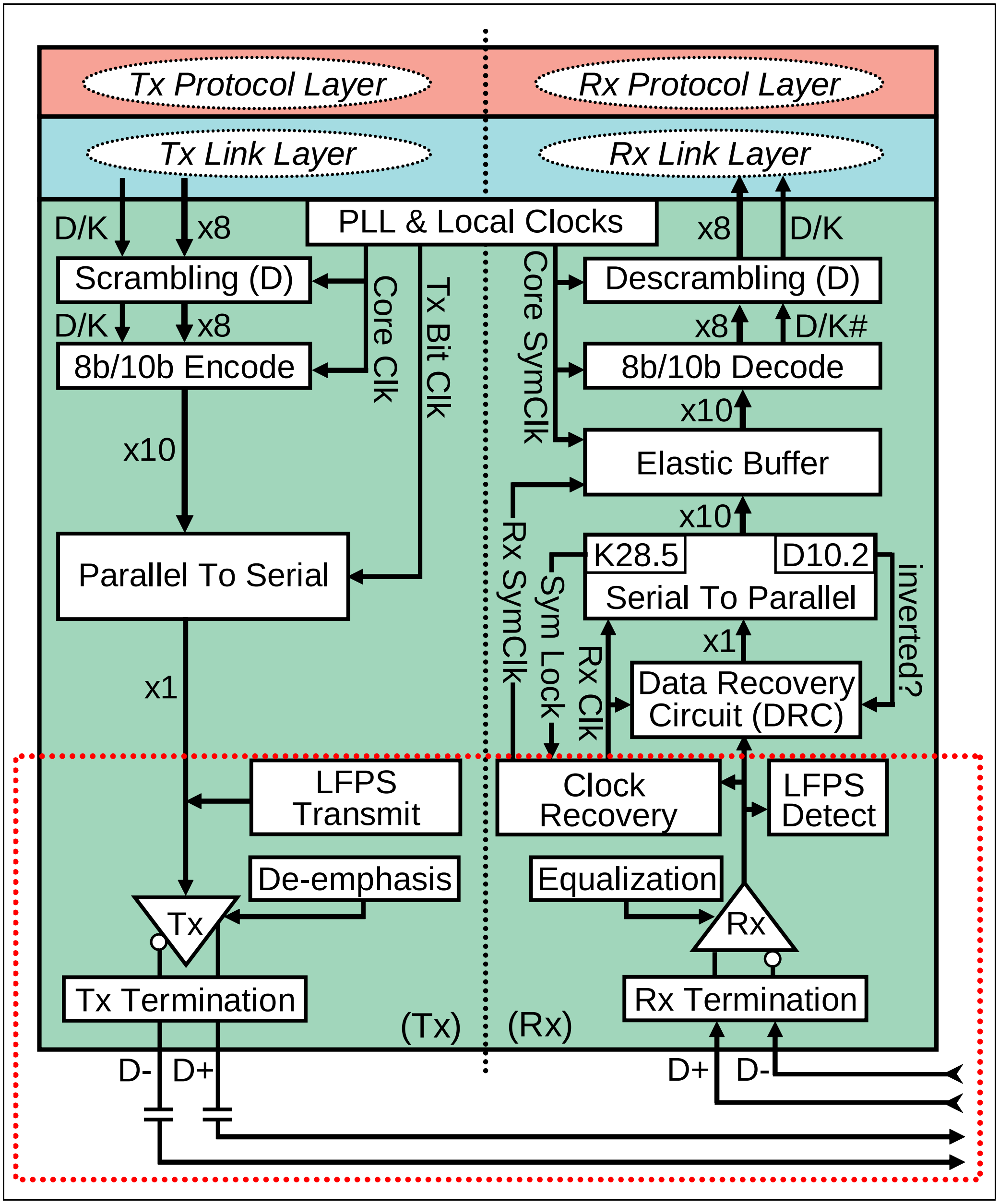
The next chapter describes key features of compliance testing, a collection of checks designed to verify conformance with USB 3.0 Specification protocol, link, and physical layer requirements. Compliance testing is slightly different for hosts, hubs, and peripherals and is required in order to assure device inter operability and to receive USB certification.

Physical Layer Electrical Signaling Scope

The SuperSpeed physical layer (PHY) can be divided into a logical and electrical signaling sections. This chapter covers the transmitter and receiver electrical specifications for the signaling elements in the highlighted region at the bottom of Figure 25-1 on page 592, including:

- Jitter budgeting
- SuperSpeed transmitter requirements
- SuperSpeed receiver requirements
- Low frequency periodic signaling (LFPS)

Figure 25-1: Physical Layer Tx And Rx Electrical Section



Normative And Informative Specifications

The USB 3.0 specification and related support documents define SuperSpeed transmitter and receiver electrical requirements as well as the testing methods to be used to verify compliance. A distinction is made between *normative* and *informative* requirements.

- Normative specifications must be observed and are intended to assure component inter-operability.
- Informative specifications are optional and intended to "assist product designers and testers in understanding the intended behavior of the SuperSpeed bus".

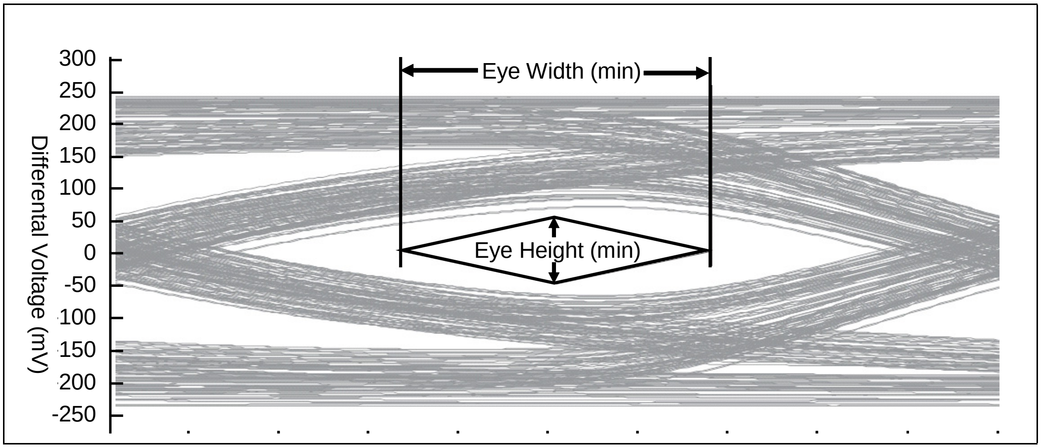
Jitter Budgeting (Informative)

General

As indicated in Figure 25-2 on page 593, when serial data moves across the SuperSpeed link at 5 Gb/s, a number of factors affect the *signal eye* quality at the receiver inputs and the receiver's ability to recover a stable clock for sampling the incoming bit stream. These factors include:

- Deterministic jitter (Dj) cause by transmission line effects
- Data-dependent random jitter (Rj) caused by data patterns
- Noise induced into the channel by other signal sources
- Frequency-dependent attenuation of the channel

Figure 25-2: Effects Of Jitter On SuperSpeed Eye



26 *Compliance Testing*

The Previous Chapter

The previous chapter discussed the SuperSpeed link electrical interface and signaling. Major topics included clocks and transmitter/receiver electrical specifications, as well as Low Frequency Periodic Signaling (LFPS).

This Chapter

This chapter describes key features of compliance testing, a collection of tests designed to verify conformance with USB 3.0 Specification protocol, link, and physical layer requirements. Compliance testing is slightly different for hosts, hubs, and peripherals and is required in order to assure device inter operability and receive USB certification.

The Next Chapter

The next chapter covers receiver loopback and link bit error rate testing (BERT), including loopback master generation of the test pattern and re-transmission by the slave. In addition, the LTSSM view of loopback and optional BERT state machine and slave error counting/reporting protocol are also described.

Scope Of USB 3.0 Compliance Testing

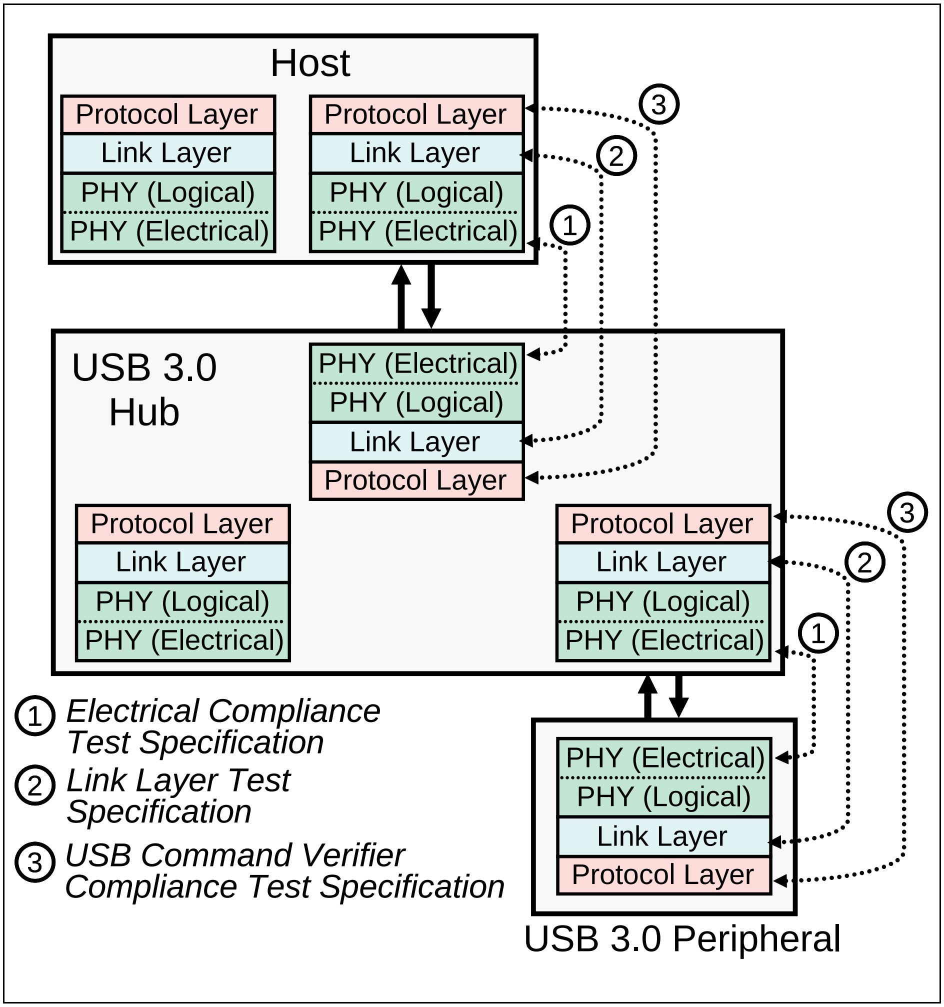
The term *compliance testing* in the context of USB 3.0 is sometimes used in a very restricted way to refer to electrical compliance testing. Electrical compliance testing defines methods for verifying SuperSpeed voltage and timing parameters are within the limits imposed by the USB 3.0 specification, but it is actually just one component in a larger suite of compliance tests.

The USB Implementers Forum (USBIF) website, usb.org, provides a set of supplementary specifications and white papers covering the full range of compliance testing. Generally speaking, the compliance test specifications describe items to be verified, procedures used to test them, and the outcome required to demonstrate compliance with the USB 3.0 Specification.

Key Compliance Testing Documents

Compliance testing is based on requirements in the base USB 3.0 Specification. Test criteria and procedures for checking SuperSpeed compliance of host, hub, and peripherals are contained in the three key compliance testing specifications identified in Figure 26-1. For clarity, the USB 2.0 interfaces and the separate specification defining compliance testing for USB 3.0 cables are not shown.

Figure 26-1: Scope Of Three Key USB 3.0 SuperSpeed Compliance Tests



Compliance Testing Resources: Documentation

Compliance testing specifications, white papers, and other documents are available from the usb.org website. These include:

- USB Command Verifier Compliance Test Specification, SuperSpeed USB
- USB 3.0 Link Layer Test Specification
- USB 3.0 Electrical Compliance Test Specification
- USB 3.0 Cable and Connector Compliance Document

If you are working with a stand-alone or integrated xHCI-compliant host controller, additional compliance checks are required and the following documents may also be useful:

- xHCI Inter operability Test Procedures For Peripherals, Hubs, and Hosts
- xHCI Backwards Compatibility Test Procedures For Hosts

Compliance Testing Resources: Hardware/Software

Check the usb.org website and USB-IF eStore to view a wide range of hardware and software products related to SuperSpeed USB compliance testing.

Some examples:

- USB30CV, the USB 3.0 Command Verifier, is the official software used to run device and hub framework tests that are required for certification. USB30CV includes compliance drivers and other software for xHCI host controllers.
- Lists of known good devices (KGD) and known good hubs (KGH) are also available to help during the compliance testing process.
- Signal Test (Sigtest) Tool is available for download. The is the official tool used during voltage and jitter electrical compliance testing. It may be used in conjunction with the SuperSpeed electrical test fixture available in the USB-IF eStore.
- Several Peripheral Development Kits (PDKs), including xHCI-based kits.
- USB Low Speed, Full Speed, High Speed, and SuperSpeed devices which are certified as compliant. The devices are very useful when checking downstream facing port compliance of a USB 3.0 host controller root hub or an external USB 3.0 hub.

27 *Receiver Loopback Testing*

The Previous Chapter

The previous chapter described key features of compliance testing, a collection of checks designed to verify conformance with USB 3.0 specification protocol, link, and physical layer requirements. Compliance testing is slightly different for hosts, hubs, and peripherals and is required in order to assure device inter operability and to receive USB certification.

This Chapter

This chapter covers receiver loopback and link bit error rate testing (BERT), including loopback master generation of the test pattern and re-transmission by the slave. In addition, the LTSSM view of loopback and optional BERT state machine and slave error counting/reporting protocol are also described.

Loopback Motivation

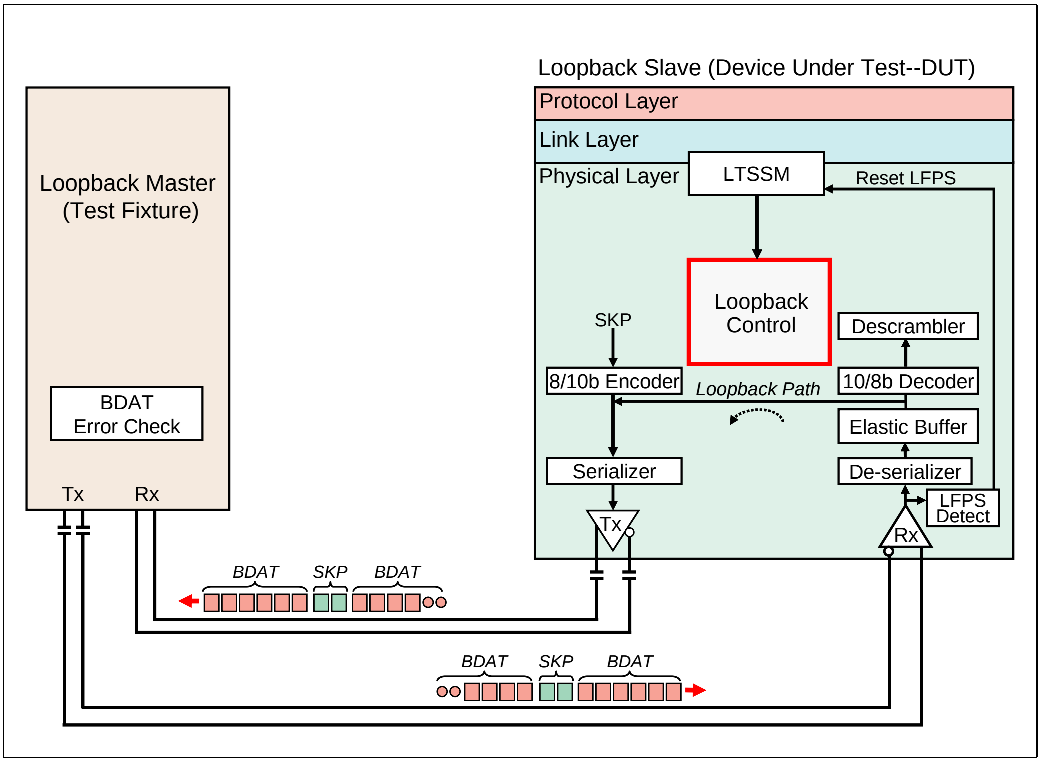
The target bit error rate (BER) for each USB 3.0 SuperSpeed link is 10^{-12}. Because of the dual simplex signaling, it is certain that errors will occur in each direction of the link. Loopback provides a standard way to quantify the actual bit error rate and isolate errors occurring on the 5GT/s link itself from those related to internal device hardware problems, protocol violations, etc. The USB 3.0 specification includes a special LTSSM Loopback state for performing the bit error rate test (BERT) and defines the role of each link partner when loopback is in use.

Standard Loopback Configuration

There is one master and one slave in a loopback configuration. The master, often a protocol exerciser or other test fixture, sources the loopback data test pattern symbols, checks for errors in symbols “looped back” (retransmitted) to it by the slave, and accumulates an error count that may then be correlated to the target link BER of 10^{-12}.

Figure 27-1 on page 640 is a conceptual diagram depicting the general loopback configuration. In this case, the slave does not support the optional BERT state machine and error counting/reporting capability.

Figure 27-1: Loopback Master And Slave



General Master And Slave Loopback Rules

- While in loopback, devices maintain the same transmitter and receiver electrical specifications as for U0 (low impedance receiver termination, etc.)
- The master uses normal rules in sending 10 bit symbols, including SKP ordered sets.
- Receiver loopback requires the slave to re-time (reclock) the incoming BERT data test pattern (BDAT) symbols before returning them to the master. As mentioned previously and illustrated in Figure 27-1 on page 640, the loopback path in the receiver resides completely at the Physical Layer.

- The specification indicates that "Loopback must occur in the 10-bit domain", meaning that the receiver must deserialize BDAT stream and perform loopback operations using 10-bit symbols. This also implies the receiver disables or bypasses its own 8b/10b encoder/decoder and scrambler/descrambler logic for the transmit side of the BDAT loopback data path.
- Other than differential signal polarity inversion, the loopback slave does not attempt to correct any errors. Symbols are sent as received, except for SKPs which may be added or dropped as needed by the slave in the management of its elastic buffer.

The Loopback Test Pattern BERT Data (BDAT)

BDAT, the bit error rate test (BERT) data test pattern, consists of scrambled and encoded NOPs as shown in Table 27-1. The duration of the BDAT transmit sequence is implementation specific but, because of the 16-stage LFSR Scrambler, an extended BDAT pattern is pseudo-random and repeats every 64K symbols.

Table 27-1: Loopback BDAT Symbols

Symbol Number	Encoding	Description
All <0:n>	D0.0 (NOP)	BDAT symbols. Bit Error Rate Test (BERT) protocol uses scrambled and encoded NOP symbols as loopback data.

Optional Loopback Slave Error Counting Support

A limitation of the standard BERT loopback scheme is that the test pattern is only sourced and checked by the loopback master, making it difficult for the test fixture to determine whether the outbound, inbound, or both directions of the SuperSpeed dual simplex link are encountering bit errors.

The specification provides an option for loopback slaves to perform a local check of received test pattern symbols as they are retransmitted to the master. To this end, a slave BERT state machine is defined as is a collection of ordered

USB 3.0 Technology A Comprehensive Guide to SuperSpeed USB

USB 3.0 SuperSpeed provides the next step in the ever-improving performance of the Universal Serial Bus, with nearly ten times the performance of USB 2.0. As expected USB 3.0 maintains compatibility with the USB 2.0 low-, full- and high-speed devices by incorporating the actual USB bus into the USB 3.0 cable along with the SuperSpeed bus. In addition SuperSpeed devices operate when plugged into USB 2.0 ports. Major improvements in USB power management is achieved through a combination of USB link power management and the new xHCI host controller.

Like all MindShare books, our USB 3.0 book takes the hard work out of deciphering the specs and provides a thorough description of the SuperSpeed USB bus. The book also contains numerous practical examples that illustrate the concepts and implementations. Written in a tutorial style, this book is ideal for anyone new to USB SuperSpeed bus, and our ***USB 2.0*** book covers the other side of USB 3.0 bus. The thorough coverage of detail in these books also make them an essential resource for seasoned veterans.

Essential topics in USB 3.0 include:

- Motivations for USB 3.0
- End-to-End Protocols
- Protocol Packet Types and Fields
- Transfer Types
- Bulk Streaming
- Port-to-Port Protocols
- Link Packet Types and Fields
- Link and Physical Layer Hardware
- Link Flow Control
- Link Error Detection and Handling
- 8b/10b Encoding/Decoding
- Ordered Sets
- Link Power Management
- SuperSpeed Reset
- Link Initialization
- USB Configuration
- USB 3.0 Hubs
- Physical Layer Electrical
- Compliance Testing

Donovan (Don) Anderson has worked with MindShare for over 21 years and has conducted courses on a wide range of topics including USB and has authored/coauthored 14 MindShare books including USB 2.0 System Architecture and PCI Express.

Jay Trodden has been a computer hardware designer for many years and a MindShare instructor for nearly 14 years. His design and teaching experience includes work with many processors and a number of bus architectures, including PCI, PCI-X, PCI Express, HyperTransport, and USB. Jay co-authored MindShare's HyperTransport System Architecture book.

MindShare is a world-renowned training and publishing company that sets a high standard of excellence in training and enables high-tech companies to adopt, implement, and roll out cutting-edge technologies quickly and confidently. We bring our knowledge to life through a wide variety of flexible learning methods and delivery options. MindShare now goes beyond the classroom to deliver engaging interactive eLearning, both in a virtual classroom and an online module format. Visit **www.mindshare.com** to learn more about our enthusiastic and experienced instructors, courses, eLearning, books and other training delivery options.

Contact MindShare at **training@mindshare.com** or **1-800-633-1440**