

"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



For training, visit [mindshare.com](http://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

<b>Intel Architecture</b> <ul style="list-style-type: none"> <li>• Intel Haswell Processor</li> <li>• Intel 32/64 Bit x86 Architecture</li> <li>• Intel QuickPath Interconnect (QPI)</li> <li>• Computer Architecture</li> </ul>	<b>Big Data</b> <ul style="list-style-type: none"> <li>• Hadoop</li> <li>• Cassandra</li> <li>• ElasticSearch</li> <li>• MongoDB</li> </ul>
<b>Mobile Technology</b> <ul style="list-style-type: none"> <li>• M-PCIe</li> <li>• UFS</li> <li>• MIPI M-PHY</li> <li>• Intel-Based Mobile (Phone/Tablet)</li> <li>• Intel Atom SoC</li> <li>• Mobile DRAM (LPDDRx)</li> <li>• High Speed/EMI for Mobile Platforms</li> </ul>	<b>IO Buses</b> <ul style="list-style-type: none"> <li>• PCI Express 3.0</li> <li>• PCIe IO Virtualization</li> <li>• USB 3.0 / 2.0</li> <li>• xHCI for USB</li> <li>• USB 3.1 Update</li> </ul>
<b>AMD Architecture</b> <ul style="list-style-type: none"> <li>• AMD Opteron Processor (Bulldozer)</li> <li>• AMD64 Architecture</li> </ul>	<b>Virtualization Technology</b> <ul style="list-style-type: none"> <li>• PC Virtualization</li> <li>• IO Virtualization</li> <li>• ARM Virtualization</li> </ul>
<b>Firmware Technology</b> <ul style="list-style-type: none"> <li>• x86 Firmware (UEFI/BIOS) Architecture</li> <li>• BIOS Essentials</li> </ul>	<b>Storage Technology</b> <ul style="list-style-type: none"> <li>• SAS</li> <li>• Serial ATA</li> <li>• AHCI for SATA</li> <li>• SATA 3.0</li> <li>• NVMe</li> </ul>
<b>ARM Technology</b> <ul style="list-style-type: none"> <li>• ARM Architecture</li> <li>• ARM Virtualization</li> </ul>	<b>DRAM Technology</b> <ul style="list-style-type: none"> <li>• DDRx &amp; LPDDRx</li> <li>• LPDDRx</li> </ul>
<b>Programming</b> <ul style="list-style-type: none"> <li>• x86 Architecture Programming</li> <li>• x86 Assembly Language Basics</li> <li>• OpenCL Programming</li> </ul>	<b>High Speed Design</b> <ul style="list-style-type: none"> <li>• High Speed Design</li> <li>• EMI/EMC</li> </ul>

[Click here to see a complete list of live and eLearning courses.](#)



## 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*

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designators appear in this book, and MindShare was aware of the trademark claim, the designations have been printed in initial capital letters or all capital letters.

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

Copyright ©2013 by MindShare, Inc.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher.

Printed in the United States of America.

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

## Revision Updates:

### 1.02

Page	Severity	Description
178 (CH8)	Minor	The last word in the caption in Figure 8-8 should be NRDY rather than NDRY.
198 (CH9)	Moderate	<p>The second sentence should read "This makes it difficult for the endpoint <b>to</b> manage transitions to the low-power link states of U1/U2, <b>because</b> it doesn't know .....</p> <p>The first word in the second Paragraph should be <b>Figure</b></p>
216 (CH10)	Minor	Item 5 second sentence should read, "packet <b>is</b> held"
229 (CH11)	Minor	In the paragraph entitled "Power Management Link Commands", the second sentence says that power management link state transitions can't occur without a link partner handshake. Add: There are also link state transitions which occur automatically on certain timeout events during power management handshake attempts.
249 (CH12)	Minor	The chapter reference contained in the second to the last bullet on this page should say: Refer to Chapter 26, entitled "Compliance Testing," on page 617 for details.
390 (CH18)	Major	<p>Table 18-1, entries 5-8 have CRD entries that are incorrect. Corrected values are:</p> <p>K23.7 EPF CRD- = 111010 1000; CRD+ = 000101 0111 K27.7 SHP CRD- = 110110 1000; CRD+ = 001001 0111 K29.7 END CRD- = 101110 1000; CRD+ = 010001 0111 K30.7 SLC CRD- = 011110 1000; CRD+ = 100001 0111</p>
497-498 (CH22)	Major	The second column in Table 22-7 (Request) should have all entries listed as <b>(01h)</b> .
502 (CH22)	Minor	<p>In Table 22-10 under column 4 (Description), in row 7, there are typos in the second sentence. It should read "<b>interfaces associated</b> with....."</p>
502 (CH22)	Minor	<p>In Table 22-10 under column 4 (Description), in row 8, there is a typo in the first sentence. It should read "reports a variety <b>of</b>....."</p> <p>In Table 22-10 under column 3 (Accessed Directly?) all of the No* entries are simplified by removing the asterisks except in the next to the last row.</p> <p>In the last sentence it should read, "...and UUID value, denoted by <b>*</b> below.</p>

560 CH23	Major	Table 23-32: Hub Feature Selectors. The rows labeled PORT_LINK_STATE and PORT_CONFIG_ERROR should be listed as C_PORT_LINK_STATE and CPORT_CONFIG_ERROR
572 (CH24)	Minor	In Table 24-6, the U2 Timeout column 512 and 768 should be 512µs and 768µs. Also FF should be FFh at the bottom of column 1.
583 (CH24)	Minor	In the last paragraph first sentence it should read, “function <b>within</b> a device. Second paragraph should read, “ <b>This is</b> typically associated with a multifunction (i.e. composite) device <b>and is</b> called Function Suspend.
584 (CH24)	Minor	Pages 584 and 585 are reorganized for clarity. This content is the same.

## Table of Contents

---

---

### About This Book

Scope.....	1
The MindShare Architecture Series .....	1
Cautionary Note .....	3
The Standard Is the Final Word .....	3
Documentation Conventions .....	3
Hexadecimal Notation .....	3
Binary Notation.....	3
Decimal Notation .....	4
Bits Versus Bytes Notation .....	4
Bit Fields .....	4
Other Terminology and Abbreviations .....	4
Visit Our Web Site.....	5
We Want Your Feedback.....	6

---

### Chapter 1: Motivation for USB 3.0

Introduction.....	7
USB 3.0 Host Controllers .....	8
Performance.....	8
USB Bandwidth Comparison .....	8
Streaming Video and Audio.....	10
Mass Storage .....	10
Improved Protocols.....	11
End-to-End Protocols .....	11
Token/Data/Handshake Shortcomings and Improvements.....	11
Token/Data/Handshake is Inefficient.....	12
Broadcast Bus Increases Power Consumption .....	13
Polled Flow Control .....	13
Error Handling and Reporting .....	13
Data Bursting.....	14
Bulk Streaming.....	14
Port-to-Port Protocols .....	14
Link Flow Control.....	15
Link Error Detection and Recovery .....	15
Power Management .....	15
SuperSpeed Bus Power Management.....	15
Broadcast Versus Unicast Bus.....	15
Link Power Management .....	16
Function Power Management.....	16
System Power Improvements .....	16

# **USB 3.0 Technology**

---

---

## **Chapter 2: USB 2.0 Background**

<b>Motivations for USB .....</b>	<b>17</b>
<b>USB Topologies .....</b>	<b>18</b>
USB 2.0 Companion Controllers .....	19
Host Controller Registers.....	20
USB Hubs .....	22
Root Hubs .....	22
External Hubs.....	23
Hub Depth Limits .....	25
Broadcast Transactions.....	26
<b>Device Architecture .....</b>	<b>28</b>
Endpoints and Buffers.....	29
Control Endpoints.....	30
Standard Requests .....	31
Optional Endpoints .....	32
<b>USB 2.0 Packets and Protocol.....</b>	<b>33</b>
Token-Data-HandShake Packet Protocol .....	34
IN Transactions .....	34
OUT Transactions .....	35
Low-Speed Transaction Variation.....	35
Control Transfer Stages.....	36
Three-Stage Transfers .....	36
Two-Stage Transfers.....	36
Maximum Payload Sizes .....	37
HS Ping Protocol - For OUT Transactions.....	37
HS Split Transaction Protocol .....	39
<b>Transaction Generation and Scheduling.....</b>	<b>41</b>
Transaction Generation.....	42
Transaction Scheduling.....	43
<b>Device Power .....</b>	<b>45</b>

## **Chapter 3: USB 3.0 Overview**

<b>USB 3.0 Topology and Compatibility .....</b>	<b>47</b>
USB 3.0 Host Controller .....	49
Topology .....	49
USB 2.0/3.0 Compatibility .....	50
Receptacle and Plug Compatibility .....	50
SuperSpeed Device Compatibility .....	51
Software Compatibility .....	51
<b>The SuperSpeed Physical Layer Environment.....</b>	<b>51</b>

## Table of Contents

---

---

USB 3.0 Composite Cable .....	51
SS Link Models.....	52
Shared Bus Power.....	53
USB 3.0 Compliant Cable Assemblies.....	54
General .....	54
USB 3.0 Power-B Connections .....	54
<b>SuperSpeed Layered Interface and Protocols.....</b>	<b>56</b>
Hub Interface Layers .....	58
Layers - Hub Forwarding Packets.....	58
Layers - Hub as Target.....	58
Layers - Hub Downstream Port (Thin Protocol Layer).....	60
Three Protocol Layers.....	61
End-to-End Protocol (Protocol Layer) .....	61
USB 2.0 vs SS IN Transaction Comparison.....	62
USB 2.0 vs SS OUT Transaction Comparison.....	63
Port-to-Port Protocol (Link Layer) .....	64
Chip-to-Chip Protocol (Physical Layer) .....	66
<b>Configuration and Descriptors.....</b>	<b>66</b>
<b>Power Management .....</b>	<b>67</b>
Link Power Management.....	68
Function Power Management .....	69

---

## Chapter 4: Introduction to End-to-End Protocol

<b>The Protocols.....</b>	<b>71</b>
The Protocol Packet Types.....	72
The Token Lives On.....	72
Data Bursting .....	73
Bulk Endpoint Streaming .....	74
Endpoint Characteristics for SuperSpeed .....	76
<b>Port-to-Port Protocol Influence.....</b>	<b>76</b>
Does SuperSpeed Support Parallel Operations? .....	77

---

## Chapter 5: End-to-End Packets

<b>Three Categories of End-To-End Packets.....</b>	<b>79</b>
<b>Transaction Packets (TPs).....</b>	<b>82</b>
ACK Header .....	83
Downstream Moving ACK.....	83
Upstream Moving ACK.....	85
NRDY Header.....	86
ERDY Header .....	88
STATUS Header.....	89

# **USB 3.0 Technology**

---

STALL Header.....	91
Notification Headers .....	92
Function Wake Header .....	92
Latency Tolerance Message Header.....	94
Bus Interval Adjustment Message Header .....	95
PING and PING RESPONSE Header.....	96
<b>Data Packet.....</b>	<b>99</b>
DATA Packet Moving Downstream .....	99
DATA Packet Moving Upstream.....	101
<b>Isochronous Timestamp Packet (ITP) .....</b>	<b>103</b>

---

## **Chapter 6: Control Protocol**

<b>Introduction to SuperSpeed Control Transfers .....</b>	<b>105</b>
<b>Control Transfer Structures and Examples .....</b>	<b>106</b>
Two-Stage Control Transfer Structure.....	107
Three-Stage Control Transfer Structure.....	107
Control IN Example — Single Data Packet.....	108
Control IN Example — Two Data Packets.....	109
Control OUT Example — Two DATA Packets .....	109
<b>Control Transfer Packet Content .....</b>	<b>110</b>
The Setup Transaction.....	110
The DATA Header Packet .....	110
Data Packet Payload — Setup Data .....	111
Setup Response — ACK .....	114
DATA Stage .....	114
Status Stage .....	115
General .....	115
Control Transfer with STALL .....	115
<b>Two Stage Control Transfer Example .....</b>	<b>116</b>
Set Address Request .....	116
Setup Stage — Set Address .....	116
Status Stage — Set Address.....	117
Protocol Details for Set Address Request.....	117
<b>Three Stage IN Control Transfer Examples .....</b>	<b>119</b>
The Get Device Descriptor Request Example .....	119
Setup Transaction — Get Device Descriptor .....	119
Data and Status Stages — Get Device Descriptor .....	120
Protocol Details — Get Device Descriptor Request.....	120
Control Transfer — Variable-Length Data Example 1 .....	122
Control Transfer — Variable-Length Data Example 2 .....	123

---

## **Chapter 7: Bulk Protocol**

## Table of Contents

---

---

<b>Introduction to Bulk Transfers.....</b>	<b>125</b>
<b>Bulk End-to-End Protocol.....</b>	<b>126</b>
Bulk IN SuperSpeed Transaction Protocol.....	127
IN Data Bursting .....	127
Single DATA Packet Burst Example.....	129
Burst IN Example with Four DATA Packets.....	130
Burst IN Example with Long Burst.....	131
End-to-End Flow Control — Bulk IN Transfers.....	132
NRDY Flow Control at Start of IN Transfer .....	132
IN Burst Flow Control Using EOB .....	133
IN Burst Flow Control Using NRDY .....	134
Short packets .....	135
Data IN Transfer Errors and Retry.....	136
Bulk OUT SuperSpeed Transaction Protocol.....	137
OUT Data Bursting .....	138
Single DATA OUT Burst Example.....	138
Burst OUT Example with Four DATA Packets.....	139
Burst OUT Example with Long Burst.....	140
End-to-End Flow Control — Bulk OUT Transfers.....	141
NRDY Flow Control at Start of OUT Transfer .....	142
OUT Burst Flow Control Using NumP=0 .....	143
DATA OUT Transfer Errors.....	144
Timeout conditions and values.....	146
<b>Standard Bulk Vs. Bulk Streaming Endpoints.....</b>	<b>146</b>
<b>Streaming And Endpoint Buffers .....</b>	<b>147</b>
Standard Bulk Endpoint Buffer .....	147
Bulk Streaming Endpoint Buffers.....	148
Stream ID (SID), A Bit More.....	148
Priming the Bulk Streaming Endpoints.....	149
<b>Reporting Bulk Endpoint Streaming Capability .....</b>	<b>150</b>
<b>Standard And Bulk Streaming Examples.....</b>	<b>150</b>
Standard Bulk Endpoints & SCSI BOT Drive .....	151
Standard Device Endpoints.....	152
Host Memory Buffers.....	152
Host Controller (HC).....	152
BOT Disk DMA Read Operation.....	153
Bulk Streaming Endpoints & UAS Drives.....	154
UAS Device Endpoints.....	155
Host Memory Buffers.....	156
Host Controller (HC).....	157
Example UAS and UASP Commands.....	157
The Non-Data UAS Command .....	158

# **USB 3.0 Technology**

---

---

Non Data UASP Command .....	159
Read DMA UAS Command .....	160
Read DMA UASP Command.....	162
Write DMA UAS Command .....	164
Write DMA UASP Command.....	165

---

## **Chapter 8: Interrupt Protocol**

Introduction to Interrupt Transfers .....	167
Interrupt IN SuperSpeed Transaction Protocol .....	169
Single Interrupt Transaction.....	169
Interrupt Burst Transactions (two DATA packets).....	170
Interrupt IN — Errors and Retries.....	173
Interrupt OUT SuperSpeed Transaction Protocol.....	176
Interrupt OUT Data Bursting.....	176
End-to-End Flow Control — Interrupt OUT Transfers.....	177

---

## **Chapter 9: Isochronous Protocol**

Introduction to Isochronous Transfers .....	181
Bus Intervals and Isochronous Service Intervals.....	183
Flexibility of Scheduling .....	184
Isochronous Timestamp Packet Delivery.....	185
General.....	185
The Timestamp Packet .....	186
Delta Value .....	187
Delayed Bit.....	187
Changing the Time Base .....	187
Requirements for Changing the Time Base.....	187
Bus Interval Adjustment Message.....	188
Ping and Ping Response .....	189
Ping Related Timing Parameters .....	190
Maximum Exit Latency (MEL) .....	190
tPingTimeout.....	190
Ping and Ping Response Header Format.....	191
Isochronous End-to-End Protocol .....	192
General Isochronous IN Protocol Rules .....	192
Isochronous IN Protocol .....	192
Isochronous IN with Burst of Two.....	193
Isochronous IN Burst of Two with Zero Payload .....	194
Isochronous IN Burst Size of 16.....	195
Isochronous OUT Transactions.....	196
Isochronous OUT with Burst Size of 16.....	196

## Table of Contents

---

---

Isochronous OUT Burst with Four Service Intervals.....	197
Service Interval N .....	197
Service interval N+1 .....	197
Service Interval N+2 .....	197
Service Interval N+3 .....	197
<b>Smart Isochronous Transaction Scheduling.....</b>	<b>198</b>
Smart Isochronous IN Transaction Example .....	199
Smart Isochronous IN Transactions with No Ping .....	201
Smart Isochronous OUT Transactions.....	202

---

### Chapter 10: USB 3.0 Hubs

<b>Introduction to SS Hubs .....</b>	<b>205</b>
<b>Hub Attachment .....</b>	<b>207</b>
<b>Packet Forwarding .....</b>	<b>207</b>
Packet Forwarding and the Layers.....	209
Packet Routing Across Hub – Link-to-Link.....	209
Packet Routing to Hub Controller.....	210
Packet Routing to Hub Controller with LMP.....	210
Unicast Routing.....	212
Packet Forwarding and Buffer Requirements .....	213
Header Buffers – Store and Forward Model.....	214
Data Payload Buffers – The Repeater Model.....	215
Transaction Deferral .....	216
<b>Hub Error Detection and Handling.....</b>	<b>218</b>
<b>Hub Link Power Management Responsibilities.....</b>	<b>219</b>
<b>Cable Power and Distribution.....</b>	<b>220</b>
<b>Reset Propagation.....</b>	<b>220</b>

---

### Chapter 11: Introduction to Port-To-Port Protocol

<b>Port-To-Port Protocol And The Link Layer .....</b>	<b>221</b>
<b>The Big Picture Revisited .....</b>	<b>222</b>
<b>Port-To-Port Protocol: Header Processing .....</b>	<b>223</b>
General.....	223
Header Fields, Two Groups .....	223
<b>Link Layer Header Processing Elements .....</b>	<b>224</b>
Tx HP Processing .....	225
Tx HP Buffers.....	225
Tx Data CRC-32 Generation .....	225
Rx HP Checks .....	226
Rx HP Buffers .....	226
Rx Data CRC-32 Checking.....	226

# **USB 3.0 Technology**

---

---

<b>Link Management Elements .....</b>	<b>227</b>
LTSSM Functional Block.....	228
Tx, Rx Link Commands.....	228
Link Command Groups .....	228
Packet Acknowledgement Link Commands .....	229
Flow Control Link Commands .....	229
Power Management Link Commands.....	229
Link Up/Link Down Link Commands.....	229
HP Acknowledgement .....	230
HP Flow Control .....	230
<b>Ordered Sets.....</b>	<b>231</b>
Ordered Set Data (D) and Control (K) Symbols.....	231
Four Ordered Set Functional Groups.....	232
Framing Ordered Sets (Delimiters) .....	232
Header Packet Framing .....	232
Data Packet Framing (Valid Data Payload).....	233
Data Packet Framing (Nullified Data Payload) .....	234
Link Command Framing .....	235
Link Training & Retraining Ordered Sets .....	236
TSEQ Ordered Set.....	236
TS1 Ordered Set .....	237
TS2 Ordered Set .....	238
Clock Compensation Ordered Set (Skip Ordered Set).....	239
Loopback Bit Error Rate Test (BERT) Ordered Sets.....	240
General .....	240
BRST Ordered Set .....	241
BERC Ordered Set .....	241
BCNT Ordered Set.....	242

---

## **Chapter 12: LTSSM And the SuperSpeed Link States**

<b>Twelve High Level LTSSM States .....</b>	<b>244</b>
<b>Why Is The LTSSM Needed?.....</b>	<b>245</b>
LTSSM Link Training And Retraining.....	245
Background: USB 2.0 Doesn't Require Training .....	245
LTSSM Coordinates SuperSpeed Link Training .....	246
LTSSM And Link-Level Error Handling .....	246
Background: USB 2.0 Approach to Bus Errors .....	246
USB 3.0 SuperSpeed Approach To Link Errors .....	247
When Does The LTSSM Become Involved?.....	247
LTSSM And SuperSpeed Link Power Management.....	248
Background: USB 2.0 Power Management .....	248
USB 3.0 SuperSpeed Power Management Enhancements.....	248

## **Table of Contents**

---

---

The LTSSM Role In SuperSpeed Link Power Management .....	248
LTSSM And SuperSpeed Link Testing .....	249
Background: USB 2.0 Testing .....	249
USB 3.0 SuperSpeed Link Testing Features .....	249
<b>LTSSM State Transitions.....</b>	<b>250</b>
General.....	250
What Is A Directed LTSSM Transition?.....	250
Handshake Signaling Locks LTSSMs.....	250
LTSSM Handshake Takes Several Forms.....	251
LFPS Signaling Events Used In LTSSM Handshake .....	251
Ordered Sets Used In LTSSM Handshake .....	252
Link Commands Used In LTSSM Handshake.....	253
LTSSM Time-outs.....	254
General .....	254
Timer Characteristics.....	254
<b>LTSSM Reference Section Note .....</b>	<b>256</b>
<b>Summary Of LTSSM Operational States .....</b>	<b>256</b>
U0 .....	256
General Description.....	257
Requirements In The U0 State .....	257
Exit Rules For The U0 State .....	257
U1 .....	259
General Description.....	259
Requirements In The U1 State .....	259
Exit Rules For The U1 State .....	260
U2 .....	261
General Description.....	261
Requirements In The U2 State .....	261
Exit Rules For The U2 State .....	262
U3 (Suspend) .....	263
General Description.....	263
Requirements In The U3 State .....	263
Exit Rules For The U3 State .....	264
<b>Summary Of LTSSM Link Initialization &amp; Training States .....</b>	<b>265</b>
Rx.Detect .....	265
General Description.....	265
Requirements In The Rx.Detect.Reset Substate .....	266
Exit Rules For The Rx.Detect.Reset Substate .....	266
Requirements In The Rx.Detect.Active Substate.....	266
Exit Rules For The Rx.Detect.Active Substate .....	267
Requirements In The Rx.Detect.Quiet Substate .....	267
Exit Rules For The Rx.Detect.Quiet Substate .....	268

# **USB 3.0 Technology**

---

---

Polling.....	268
General Description.....	268
Requirements In The Polling.LFPS Substate .....	269
Exit Rules For The Polling.LFPS Substate .....	269
Requirements In The Polling.RxEQ Substate .....	270
Exit Rules For The Polling.RxEQ Substate.....	270
Requirements In The Polling.Active Substate .....	271
Exit Rules For The Polling.Active Substate.....	271
Requirements In The Polling.Configuration Substate.....	272
Exit Rules For The Polling.Configuration Substate .....	272
Requirements In The Polling.Idle Substate.....	273
Exit Rules For The Polling.Idle Substate .....	273
Recovery .....	274
General Description.....	275
Requirements In Recovery.Active.....	275
Exit Rules For Recovery.Active .....	275
Requirements In Recovery.Configuration .....	276
Exit Rules For Recovery.Configuration .....	276
Requirements In Recovery.Idle.....	277
Exit Rules For Recovery.Idle .....	277
Hot Reset .....	278
General Description.....	278
Requirements In Hot Reset.Active .....	279
Exit Rules For Hot Reset.Active.....	279
Requirements In Hot Reset.Exit.....	280
Exit Rules For Hot Reset.Exit .....	280
<b>Summary Of LTSSM Testing States .....</b>	<b>281</b>
Compliance Mode.....	281
General Description.....	281
Requirements In Compliance Mode .....	282
Exit Rules For Compliance Mode .....	282
Loopback .....	283
General Description.....	283
Requirements In Loopback.Active .....	283
Exit Rules For Loopback.Active .....	284
Requirements In Loopback.Exit.....	284
Exit Rules For Loopback.Exit .....	284
<b>Summary Of Other LTSSM States .....</b>	<b>285</b>
SS.Inactive State.....	285
General Description.....	285
Basic SS.Inactive Requirements .....	285
Requirements In SS.Inactive.Quiet Substate.....	286

## **Table of Contents**

---

---

Exit Rules For SS.Inactive.Quiet Substate .....	286
Requirements In SS.Inactive.Disconnect.Detect.....	286
Exit Rules For SS.Inactive.Disconnect.Detect .....	286
SS.Disabled State.....	287
General Description.....	287
Exit Rules For The SS.Disabled State .....	288
Requirements For The SS.Disabled.Default Substate.....	288
Exit Rules For The SS.Disabled.Default Substate.....	288
Exit Rules For The SS.Disabled.Error Substate.....	288

---

### **Chapter 13: Link Commands**

Four Groups Of Link Commands .....	289
Link Commands On The SuperSpeed Link .....	291
For Additional Details On Link Commands .....	291
Link Command Encoding And Use.....	292
Packet Acknowledgement: LGOOD_n.....	292
Packet Acknowledgement: LBAD .....	293
Packet Acknowledgement: LRTY .....	294
Flow Control: LCRD_x.....	295
Power Management: LGO_Ux (The Request) .....	296
Power Management: LAU (Acceptance) .....	297
Power Management: LXU (Rejection).....	298
Power Management: LPMA (Acknowledgement).....	299
Link Up/Link Down: LDN.....	300
Link Up/Link Down: LUP .....	301
Notes On Link Command Placement.....	302

---

### **Chapter 14: Header Packet Processing**

Link Layer Packet Processing Role .....	303
Link Layer Transmitter Packet Processing.....	305
Header Sequence Number Assignment .....	306
Header Link Control Word CRC-5 Generated .....	307
Header CRC-16 Generated .....	308
Copy Is Placed In A Header Packet Buffer .....	309
Data Packet Payload CRC-32 Generation.....	310
General .....	310
Tx DPP CRC-32 Generation, The Hardware Details .....	311
Framing Added, Packet Sent To Physical Layer .....	312
Receiver Packet Processing .....	313
Inbound Link Layer Traffic .....	313
Key Header Packet Processing Elements .....	314

# **USB 3.0 Technology**

---

Header Packet CRC-16 Checked.....	315
Header Link Control Word CRC-5 Checked .....	316
Header Sequence Number Checked .....	317
Header Packet Buffer Accepts The Header.....	318
Data Packet Payload (DPP) CRC-32 Checked .....	319
General .....	319
Rx DPP CRC-32 Checking, The Hardware Details .....	320

---

## **Chapter 15: Header Packet Flow Control**

<b>Background: Host And Device Flow Control .....</b>	<b>321</b>
USB 2.0 Flow Control, Very Limited.....	322
The SuperSpeed Flow Control Approach .....	322
End To End Flow Control.....	322
Link Level Flow Control Is Also Needed.....	323
<b>SuperSpeed Link Level Flow Control Basics .....</b>	<b>324</b>
<b>Flow Control Elements.....</b>	<b>325</b>
Transmitter Elements .....	326
Tx Header Packet (HP) Buffers.....	326
Remote Rx HP Credit Count .....	326
Credit HP Timer.....	326
Next Rx LCRD_x.....	327
Receiver Elements .....	327
Rx Header Packet (HP) Buffers.....	327
LCRD_x Generation .....	327
<b>Header Packet Flow Control Link Commands.....</b>	<b>328</b>
General.....	328
LCRD_x Link Command Format.....	328
<b>Flow Control Initialization.....</b>	<b>329</b>
Flow Control Logic State After Reset.....	329
Flow Control Logic Initialization.....	330
<b>Flow Control During Normal Operations.....</b>	<b>331</b>
The First Header Is Sent .....	331
Header Packet Reaches Rx HP Buffer .....	332
Emptying An Rx HP Buffer .....	333
Transmitter Receives LCRD_x .....	334
If Tx Receives A Valid LCRD_x .....	335
If Tx Receives An Invalid LCRD_x .....	336

---

## **Chapter 16: Link Errors & Packet Acknowledgement**

<b>Background: USB 2.0 Error Handling .....</b>	<b>339</b>
<b>USB 3.0 SuperSpeed Requires A New Approach .....</b>	<b>340</b>

# Table of Contents

---

---

SuperSpeed Signaling Affects Bit Error Rates .....	340
Complex USB Topologies A Challenge At 5 Gb/s.....	340
Upstream Asynchronous Message Errors.....	341
<b>Scope Of SuperSpeed Link Errors .....</b>	<b>342</b>
Training Sequence Errors.....	343
Errors Occurring While Link is in U0 .....	343
<b>SuperSpeed Link Level Error Correction Approach .....</b>	<b>343</b>
Goals Of Header Packet Acknowledgement .....	344
Reliable Delivery Of Header Packets.....	344
Hand Off The More Serious Errors .....	344
End-To-End Acknowledgement Is Still Needed .....	345
<b>Header Packet Acknowledgement Elements.....</b>	<b>345</b>
Transmitter Elements .....	346
HDR Seq# Assignment.....	346
Next Tx HDR Seq# .....	347
HDR CRC-5.....	347
HDR CRC-16.....	347
Tx HP (Header Packet) Buffers.....	347
Pending_HP_Timer .....	347
Next Rx LGOOD_n.....	348
Receiver Elements .....	349
HDR CRC-5 And CRC-16 Checks .....	349
Retries (Retry attempt counter) .....	349
LBAD Generation .....	350
HDR Seq# Check .....	350
Next Rx HDR Seq# .....	350
Rx Header Packet (HP) Buffers.....	351
LGOOD_n Generation .....	351
<b>Header Packet Acknowledgement Link Commands .....</b>	<b>351</b>
LGOOD_n Link Command Format .....	352
LBAD Link Command Format.....	353
LRTY Link Command Format .....	354
<b>Header Packet Acknowledgement Initialization.....</b>	<b>355</b>
HP Acknowledgement Logic After Reset .....	355
HP Sequence Number Advertisement.....	356
General .....	356
Sequence Of Events .....	357
<b>HP Acknowledgement Sequence: No Retry Required .....</b>	<b>357</b>
First Header Packet Is Sent.....	357
Header CRC-5, CRC-16 Checked By The Receiver .....	359
Header Sequence Number Checked By Receiver .....	360
Receiver Accepts And Acknowledges The Packet.....	361

# **USB 3.0 Technology**

---

---

Transmitter Checks LGOOD_n Acknowledgement .....	362
LGOOD_n Valid: Retire Header Packet .....	363
<b>HP Acknowledgement Sequence: Retry Required.....</b>	<b>364</b>
The Header Packet Is Sent .....	364
Header CRC-5 or CRC-16 Check Fails, LBAD Sent .....	365
Transmitter Receives LBAD, Starts The Retry .....	366
Retry Header Packet CRC-5, CRC-16 Checked .....	367
Retry Header Packet Sequence Number Checked .....	368
Receiver Acknowledges The Retry Header Packet.....	369
Transmitter Checks LGOOD_n Following The Retry .....	370
LGOOD_n Valid: Retire The Header Packet.....	371

---

## **Chapter 17: Introduction to Chip-To-Chip Protocol**

<b>Chip-To-Chip Protocol And The Physical Layer .....</b>	<b>373</b>
The Big Protocol Picture, Once More .....	374
<b>Chip-To-Chip Protocol: PHY Logical Processing.....</b>	<b>375</b>
A Note About D/K Bytes And Symbols.....	375
A Few Common Structure D/K Examples .....	376
<b>The Physical Layer: PHY Logical Processing.....</b>	<b>377</b>
Tx Scrambling .....	378
Tx 8b/10b Encoding .....	378
Tx Serialization.....	378
Rx Clock/Data Recovery .....	378
Rx De-serialization.....	378
Rx Elastic Buffer .....	379
Rx 8b/10b Decoding .....	379
Rx De-scrambling .....	379
Receiver PHY Forwards Traffic to Link Layer .....	379
<b>The Physical Layer: PHY Electrical Specifications.....</b>	<b>379</b>
General Topics .....	381
Transmitter PHY Electrical Topics .....	381
Receiver PHY Electrical Topics .....	381
Low Frequency Periodic Signaling (LFPS).....	381

---

## **Chapter 18: Physical Layer Logical Functions**

<b>Physical Layer Logic Definitions .....</b>	<b>383</b>
<b>Tx Physical Layer (PHY) Logic .....</b>	<b>385</b>
Tx Outbound Bytes And The D/K Flag .....	385
Data Scrambling .....	386
General .....	386
Disabling Scrambling .....	388

## Table of Contents

---

---

8b/10b Encoding .....	388
The Purpose Of Encoding .....	389
Two Key 8b/10b Encoding Rules .....	390
Some 8b/10b Encoding Examples .....	390
8b/10b Encoding And Current Running Disparity .....	391
The AC-Coupled SuperSpeed Link .....	392
Advantages of AC-Coupling .....	392
Challenges of AC-Coupling .....	392
8b/10b Encoding Minimizes Disparity .....	392
An Example .....	393
Is The Transmitter CRD Choice Correct? .....	393
Parallel To Serial Data Conversion .....	394
Differential Transmitter .....	395
<b>Rx Physical Layer (PHY) Logic .....</b>	<b>396</b>
Differential Receiver And Equalization .....	396
Clock And Data Recovery .....	397
Clock Recovery .....	398
Data Recovery Circuit (DRC) .....	398
Serial To Parallel Data Conversion .....	398
Serial To Parallel Conversion and COM Symbol Lock .....	399
Differential Polarity Inversion (If Needed) .....	400
Elastic Buffer .....	401
Tx And Rx Clock Requirements .....	403
5 GHz Base Clock .....	403
Spread Spectrum Clocking .....	403
Implications Of Tx, Rx Clock Difference .....	403
The Result: Two Rx Clock Domains .....	404
Receiver PHY Clock Domain Notes .....	405
Elastic Buffer Architecture .....	406
Skip (SKP) Ordered Sets .....	407
8b/10b Decode .....	408
Error Checking At The Decoder .....	408
Decoded Data (D) And Control (K) Bytes .....	409
Descrambling Of Data (D) Bytes .....	410
Forwarding Traffic To Link Layer .....	411

---

## Chapter 19: SuperSpeed Reset Events

<b>PowerOn Reset .....</b>	<b>413</b>
Software May Enable And Disable VBUS .....	414
PowerOn Reset Impact On Device State .....	415
PowerOn Reset And Self-Powered Devices .....	415
<b>Warm Reset .....</b>	<b>416</b>

# **USB 3.0 Technology**

---

---

How Warm Reset Is Signaled On The Link .....	417
After Warm Reset Completes.....	418
<b>Hot Reset.....</b>	<b>418</b>
How Hot Reset Is Signaled On The Link.....	419
After Hot Reset Completes.....	420
Hot Reset, The LTSSM View .....	420
General Description.....	421
Requirements In Hot Reset.Active .....	421
Exit Rules For Hot Reset.Active.....	421
Requirements In Hot Reset.Exit.....	422
Exit Rules For Hot Reset.Exit .....	422
<b>Notes About Warm Reset And Hot Reset Latency .....</b>	<b>423</b>
Warm Reset Is Time Consuming, But Thorough .....	423
Hot Reset Is Much Faster, Not As Thorough.....	423
<b>How Does Software Know If Hot Reset Is Possible?.....</b>	<b>423</b>
<b>The Two In-Band Reset Hub Requests.....</b>	<b>424</b>
SetFeature (BH_PORT_RESET).....	424
SetFeature (PORT_RESET) .....	424
<b>Hubs Propagate Resets Downstream .....</b>	<b>425</b>
General.....	425
Hub Reset Propagation, Key Rules .....	426

---

## **Chapter 20: Link Training**

<b>Link Training Required Before SuperSpeed Operation .....</b>	<b>427</b>
USB Physical Connections Complicate Link Training .....	428
The Long Channel.....	429
The Short Channel .....	429
SuperSpeed Link Training Is Adaptive .....	429
<b>Which Logic Requires Training?.....</b>	<b>430</b>
General.....	430
Summary Of Key Link Training Elements.....	431
Receiver Equalization.....	431
Clock Recovery.....	432
Data Recovery Circuit (DRC) .....	432
Serial To Parallel Conversion and Symbol Lock .....	433
Differential Polarity Inversion (If Needed).....	433
Elastic Buffer Initialization .....	433
<b>Link Training Is Managed by LTSSM .....</b>	<b>434</b>
<b>The Normal Link Training Sequence .....</b>	<b>435</b>
Rx.Detect.Reset.....	435
Normal Exit From Rx.Detect.Reset .....	436
Other Exits From Rx.Detect.Reset .....	436

## Table of Contents

---

---

Rx.Detect.Active .....	437
SuperSpeed Receivers Enable & Disable Termination.....	438
Receiver Common Mode Input Impedance Range.....	438
The Detection Method.....	439
Normal Exit: From Rx.Detect.Active To Polling .....	441
Alternative Exit: From Rx.Detect.Active To Rx.Detect.Quiet.....	441
Other Exits From Rx.Detect.Active .....	441
Rx.Detect.Quiet .....	442
Polling.LFPS.....	443
The Polling.LFPS Burst Format .....	443
Other Polling.LFPS Requirements And Options .....	444
Normal Exit: From Polling.LFPS To Polling.RxEQ.....	444
Other Exits From Polling.LFPS .....	444
Polling.RxEQ.....	445
Other Requirements In Polling.RxEQ.....	445
The TSEQ Ordered Set.....	446
Two Special Symbols In TSEQ.....	447
Symbol K28.5 (COM) .....	447
Symbol 16-31 D10.2 .....	448
Normal Exit: From Polling.RxEQ To Polling.Active .....	449
Polling.Active .....	450
Other Requirements In Polling.Active .....	450
The TS1 Ordered Set.....	450
Normal Exit: From Polling.Active To Polling.Configuration.....	451
Other Exits From Polling.Active.....	452
Polling.Configuration.....	452
The TS2 Ordered Set.....	453
Normal Exit: From Polling.Configuration To Polling.Idle .....	453
Other Exits From Polling.Configuration .....	454
Polling.Idle .....	454
General .....	454
Other Requirements In Polling.Idle .....	455
LGOOD_n Advertisement.....	455
General .....	455
Format Of LGOOD_n Link Command.....	456
LCRD_x Advertisement.....	457
General .....	457
Format Of LCRD_x Link Command.....	458
Port Capability LMP .....	459
Port Capability LMP Header Format.....	460
Link Speed Field (DW0, bits 15:9) .....	460
Num HP Buffers Field (DW1, bits 7:0).....	460

# **USB 3.0 Technology**

---

---

D (Port Direction Support) Field (DW1, bits 17:16) .....	461
Tiebreaker Field (DW1, bits 23:20) .....	461
Port Configuration LMP .....	462
Port Configuration LMP Header Format .....	463
Selected Link Speed Field (DW0, bits 15:9).....	463
Port Configuration Response LMP.....	464
Port Configuration Response LMP Header Format .....	465
Response Code Field (DW0, bits 15:9) .....	465
<b>Link Training Is Complete, What Now? .....</b>	<b>466</b>
General.....	466
Life In U0, A Few More Rules .....	466

---

## **Chapter 21: Link Recovery and Retraining**

Reasons For Link Recovery And Retraining.....	467
Recovery And Retraining Managed By The LTSSM .....	468
Recovery Is Speedy .....	469
Which Logic Is Retrained In Recovery? .....	470
Entering Recovery: Two Common Examples .....	471
U1-U3 Exit And Transition To Recovery .....	471
U0 Link Layer Error And Transition To Recovery .....	472
The Recovery Substate Events.....	474
Recovery.Active .....	474
Other Requirements In Recovery.Active.....	474
Recovery.Active And The TS1 Ordered Set.....	475
Two Special Symbols In TS1 Ordered Sets .....	475
Symbol K28.5 (COM) .....	475
Symbol 6-15 D10.2 .....	476
Normal Exit: Recovery.Active To Recovery.Configuration .....	477
Other Exits From Recovery.Active .....	478
Recovery.Configuration .....	478
Recovery.Configuration And The TS2 Ordered Set .....	478
Normal Exit: From Recovery.Configuration To Recovery.Idle .....	479
Other Exits From Recovery.Configuration .....	479
Recovery.Idle .....	479
General .....	479
Other Requirements In Recovery.Idle .....	480
LGOOD_n Advertisement.....	480
A Simple Example .....	480
Format Of LGOOD_n Link Command.....	481
LCRD_x Advertisement .....	482
General .....	482
A Simple Example .....	482

## Table of Contents

---

---

For More Details On The LTSSM Recovery ..... 483

---

### Chapter 22: Device Configuration

Overview.....	485
Standard Device Requests.....	486
Device Detection and Reporting.....	488
Device Detection Process — Details.....	489
Status Change Indicator (Endpoint 1).....	489
Hub Port Status and Change Indicators .....	491
Get Port Status Request.....	495
Clear Port Connection Request.....	496
The Device Configuration Process .....	498
Address Device .....	498
Getting the Descriptors .....	499
General .....	499
Standard Descriptors.....	500
Device Descriptor.....	502
Descriptor Length and Type .....	504
DeviceClass/SubClass/Protocol .....	504
Maximum Payload Size for Endpoint Zero (EP0).....	505
String Index Values .....	505
BOS Descriptor .....	506
USB 2.0 Extension .....	507
SuperSpeed USB Device Capability.....	508
Container ID .....	509
Configuration Descriptor.....	510
Number of Interfaces.....	512
Configuration Value.....	512
Attributes and Maximum Power.....	512
Interface Association Descriptor.....	512
Interface Descriptors.....	513
General .....	513
Interface Number and Alternate Setting .....	514
Endpoint Descriptors .....	516
Endpoint Companion Descriptor .....	518
String Descriptors .....	519
Setting Device Configuration.....	521
Additional Requests .....	522
Class Driver Initialization.....	522

---

### Chapter 23: SuperSpeed Hub Configuration

# **USB 3.0 Technology**

---

---

<b>Configuring the Hub .....</b>	<b>523</b>
<b>Standard Device Requests.....</b>	<b>524</b>
<b>Hub Class Requests .....</b>	<b>526</b>
<b>Device Detection and Reporting.....</b>	<b>527</b>
General.....	527
Device Detection Process .....	528
Status Change Endpoint (Endpoint 1).....	528
Hub Port Status and Change Indicators.....	529
Get Port Status Request.....	533
Clear Port Requests .....	534
<b>The Hub Configuration Process.....</b>	<b>536</b>
Addressing the Device .....	536
Getting Descriptors.....	536
General .....	536
Get Device Descriptor .....	539
BOS Descriptor .....	541
USB 2.0 Extension .....	542
SuperSpeed USB Device Capability .....	542
Container ID .....	544
Configuration Descriptor.....	545
Configuration Value.....	546
Bus- or Self-Powered Hub .....	546
Maximum Bus Power Consumed .....	547
Hub's Interface Descriptor.....	547
Status Change Endpoint Descriptor.....	548
General .....	548
Hub Status Change Endpoint Address/Transfer Direction.....	549
Status Change Endpoint Companion Descriptor.....	550
String Descriptors .....	550
Additional Requests .....	551
<b>Hub Driver Initialization .....</b>	<b>551</b>
Hub Class Descriptor .....	551
Power Switching Mode Implemented .....	554
Over-Current Protection Mode.....	555
Maximum Bus Current for Hub Controller .....	555
Device Removable/Non-removable .....	555
Hub-Specific Requests.....	555
Set Hub Depth .....	555
Port U1/U2 Timeout .....	557
Set Port Remote Wake Mask .....	557
Port Link State .....	558
Set Port Power .....	558

## Table of Contents

---

---

Port Reset .....	559
BH Port Reset.....	559
Hub Status Change Endpoint Request .....	560
Reading the Hub Status Field.....	560
Reading Hub Port Status .....	560
<b>Summary of Hub Feature Selectors .....</b>	<b>561</b>

---

## Chapter 24: SuperSpeed Power Management

<b>Principles of SuperSpeed Power Management.....</b>	<b>563</b>
Background.....	564
Power Management Design Goals .....	565
Link Power Management .....	565
Function Power Management.....	566
Suspend/Resume (U3).....	566
<b>SuperSpeed Link Power Management .....</b>	<b>566</b>
The Link Power States.....	566
Link-Power Hierarchy.....	568
<b>Software Role in Link Power Management.....</b>	<b>569</b>
Host Software Triggers .....	569
Hub Inactivity Timers .....	570
U1 Inactivity Timeouts.....	571
U2 Inactivity Timeouts.....	572
U0 → U1 → U2 with Silent Transition .....	573
Device-Specific Inactivity Timers/Algorithms.....	574
Upstream and Downstream U1/U2 Triggers .....	575
<b>The Handshake Sequence .....</b>	<b>576</b>
Entering The U1 or U2 States .....	576
U1 / U2 Handshake and Timer Behavior .....	577
<b>Device Suspend — U3.....</b>	<b>578</b>
General.....	578
Entering SuperSpeed Suspend.....	579
The U3 Handshake .....	579
Device Suspend State .....	580
<b>U1/U2/U3 Exit to U0 .....</b>	<b>580</b>
Triggering U1/U2/U3 Exit.....	580
The Exit Handshake Sequence and Timing .....	581
The HandShake .....	581
Wake Notification .....	583
<b>Function Suspend .....</b>	<b>583</b>
General.....	583
The Function Suspend Request.....	584
<b>Latency Tolerance Message Reporting .....</b>	<b>585</b>

# **USB 3.0 Technology**

---

---

Software Calulates Exit Latencies.....	586
Device Reports Current Latency.....	588

---

## **Chapter 25: SuperSpeed Signaling Requirements**

<b>Physical Layer Electrical Signaling Scope.....</b>	<b>591</b>
<b>Normative And Informative Specifications.....</b>	<b>593</b>
<b>Jitter Budgeting (Informative) .....</b>	<b>593</b>
General.....	593
Jitter Budget Allocations .....	594
<b>SuperSpeed Voltage Levels, Some Definitions .....</b>	<b>595</b>
<b>SuperSpeed Transmitter Requirements .....</b>	<b>596</b>
Tx Electrical Specifications .....	596
Normative Tx Electrical Parameters .....	596
Informative Tx Electrical Parameters.....	598
Transmitter Low Power Option.....	599
Transmitter De-emphasis.....	600
Background.....	600
Transmitter Signal De-emphasis Helps.....	601
Transmitter Spread Spectrum Clocking (SSC).....	603
<b>SuperSpeed Receiver Requirements .....</b>	<b>605</b>
Rx Electrical Specifications .....	605
Normative Rx Parameters .....	606
Informative Rx Parameters.....	607
Rx Equalizer Training.....	608
SuperSpeed Receiver Termination .....	609
Receiver Common Mode Input Impedance Range.....	610
The Detection Method.....	610
<b>Low Frequency Periodic Signaling Requirements .....</b>	<b>612</b>
LFPS Is A Simple Squarewave .....	613
Six LFPS Signaling Event Types .....	613
LFPS Electrical Requirements .....	614
<b>Transmitter And Receiver DC Specifications .....</b>	<b>615</b>
High Impedance Reflections (Normative) .....	615
ESD Protection (Informative) .....	615
Short Circuit Requirements (Informative) .....	616

---

## **Chapter 26: Compliance Testing**

<b>Scope Of USB 3.0 Compliance Testing.....</b>	<b>617</b>
<b>Key Compliance Testing Documents.....</b>	<b>618</b>
Compliance Testing Resources: Documentation.....	619
Compliance Testing Resources: Hardware/Software .....	619

## Table of Contents

---

---

<b>The Compliance Testing Environment.....</b>	<b>620</b>
<b>Key Test Suite 1: USB Command Verifier Compliance .....</b>	<b>620</b>
General Approach USB Command Verification Tests .....	621
Test Assertions .....	621
USB Device State Test Assertions.....	623
Generic Device Operation Test Assertions .....	623
USB Device Request Test Assertions .....	624
USB Descriptor Definition Test Assertions.....	624
Nineteen Tests Used In USB Command Verification .....	624
<b>Key Test Suite 2: Link Layer Compliance .....</b>	<b>626</b>
General Approach In Link Layer Compliance Tests .....	626
Test Assertions .....	626
Link Management & Flow Control Test Assertions .....	627
Link Error Rules & Recovery Test Assertions .....	628
Link Reset Test Assertions.....	628
LTSSM Test Assertions .....	628
Protocol Layer Test Assertions .....	629
Hub Port Test Assertions.....	629
Forty Tests Are Used In Link Layer Testing .....	629
<b>Electrical Compliance Testing.....</b>	<b>632</b>
Factors Affecting USB Electrical Compliance Testing .....	632
Electrical Compliance And The Varied USB Topology.....	632
The Long Channel.....	633
The Short Channel .....	634
USB 3.0 Adds Even More Complexity .....	634
Electrical Compliance Test Support Is Designed-In .....	634
LTSSM Compliance Mode: Device Roles .....	635
Nine Compliance Mode Test Patterns .....	636
Electrical Compliance Testing And The LTSSM .....	637

---

## Chapter 27: Receiver Loopback Testing

<b>Loopback Motivation .....</b>	<b>639</b>
<b>Standard Loopback Configuration .....</b>	<b>639</b>
General Master And Slave Loopback Rules.....	640
The Loopback Test Pattern BERT Data (BDAT).....	641
<b>Optional Loopback Slave Error Counting Support.....</b>	<b>641</b>
<b>Loopback BERT Ordered Sets .....</b>	<b>642</b>
BERT Reset (BRST).....	643
BDAT Error Count Request (BERC).....	643
BDAT Error Count (BCNT) Value .....	644
<b>Slave BERT Command Processing Rules.....</b>	<b>644</b>
General.....	644

# **USB 3.0 Technology**

---

---

BRST & BERC If No Slave Error Counting Support .....	645
<b>Loopback And The LTSSM.....</b>	<b>645</b>
General.....	645
Loopback LTSSM Substates And Transition Events.....	646
Loopback.Active Substate .....	646
Loopback.Exit Substate .....	648
A Note About Differences In Loopback Exit Latency .....	649
<b>Loopback May Also Be Used In Compliance Testing .....</b>	<b>650</b>

## List of Figures

---

---

1-1	SuperSpeed Link Interface .....	9
1-2	USB 2.0 vs SS OUT Transaction Protocols .....	12
2-1	Eight-Port Companion Controller Built into Chipset.....	20
2-2	Companion Controller and Register Sets.....	21
2-3	Example Companion Controller with Root Hubs .....	23
2-4	External Full-Speed and High-Speed Hubs.....	24
2-5	Hub Depth .....	25
2-6	Low-Speed Packet Broadcast .....	26
2-7	HS Broadcast and HS Split Transactions .....	27
2-8	Standard Device Descriptor Tree .....	29
2-9	Device Endpoints and Associated Buffers .....	30
2-10	Token-Data-Handshake Packet Types and Elements .....	34
2-11	Example IN Transaction FS or HS.....	34
2-12	Example OUT Transaction FS or HS.....	35
2-13	Low-Speed Packet from Host Controller .....	35
2-14	Three Stage Control Transfer (Read) .....	36
2-15	Two-Stage Control Transfer .....	37
2-16	OUT Data NAK vs Ping Protocol Solution .....	38
2-17	Ping Protocol Sequence.....	39
2-18	Major High-Speed Hub Functions .....	40
2-19	Split Transaction Sequence - Bulk IN Transfer .....	41
2-20	Transaction Generation.....	43
2-21	Frames and MicroFrames .....	44
2-22	Frame Lists Executed by UHCI Controller .....	44
3-1	Example USB 3.0 Host Controller and Topology .....	48
3-2	Transmit and Receive Differential Pairs.....	49
3-3	The USB 3.0 Cable Interface .....	52
3-4	SS Signaling Environments .....	53
3-5	Powered B Connection Examples .....	55
3-6	SS Interface Layers and Protocols .....	56
3-7	Hub Bypasses the Protocol Layers When Forwarding Packets .....	59
3-8	Hub as Target Device .....	59
3-9	Hub as Target Device with LMP Delivery .....	60
3-10	Example of End-to-End Communication.....	61
3-11	IN Transaction — Token/Data/Handshake versus End-to-End Protocols.....	62
3-12	OUT Transaction — Token/Data/Handshake versus End-to-End Protocols.....	64
3-13	Port-to-Port Protocol Between Link Partners .....	65
3-14	SuperSpeed Descriptors.....	67
3-15	USB 2.0 vs USB SS Bus Idle .....	68
4-1	SuperSpeed Headers Support IN, OUT and Setup Tokens.....	73
4-2	IN Burst Example with Maximum Payload Size of Four .....	74
4-3	Example UASP Structure.....	75

# **USB 3.0 Technology**

---

5-1	Protocol Layer Overview and Packet Types.....	80
5-2	Basic Header Packet .....	81
5-3	Transaction Packet - Basic Format .....	82
5-4	ACK Header Fields - Moving Downstream .....	84
5-5	ACK Header Fields - Moving Upstream.....	85
5-6	NRDY Header Fields.....	87
5-7	ERDY Header Fields .....	88
5-8	Status Header Packet.....	90
5-9	Stall Header Fields.....	91
5-10	Function Wake Notification Header — Header Moves Upstream .....	93
5-11	Latency Tolerance Message Header .....	94
5-12	Bus Interval Adjustment Message Header .....	95
5-13	PING Header Fields — Header Moving Downstream .....	97
5-14	PING RESPONSE Header — Header Moving Upstream.....	98
5-15	Data Header and Payload Moving Downstream .....	100
5-16	Data Header and Payload Moving Upstream.....	102
5-17	Isochronous Timestamp Header .....	104
6-1	Two-Stage Control Transfer .....	107
6-2	Three Stage Control Transfer Structure (IN or OUT).....	108
6-3	Example Three-Stage Transfer with Single IN Transaction .....	108
6-4	Example Two-Packet Data Stage.....	109
6-5	Example Two-Packet OUT Data Stage .....	109
6-6	DATA Header — Setup Request.....	111
6-7	Setup Data Fields Showing Byte Order of Packet Transmission Across the Link ...	
114		
6-8	STATUS Header Format and Content.....	115
6-9	Set Address Request Data .....	117
6-10	Control Transfer Protocol — Two-Stage Transfer .....	118
6-11	Get Device Descriptor Request Data.....	119
6-12	Three-Stage Get Device Descriptor Protocol .....	121
6-13	Variable Length Data During Control Read Operation (with Short Packet).....	122
6-14	Variable Length Data During Control Read Operation (No Short Packet).....	123
7-1	Bulk IN Transaction — Burst of One DATA packet.....	129
7-2	Bulk IN Transfer — Burst of Four DATA Packets .....	130
7-3	IN Burst Transaction with Large Transfer .....	131
7-4	NRDY Flow Control at Start of Transfer .....	133
7-5	IN Burst Flow Control Using EOB .....	134
7-6	IN Burst Flow Control Using NRDY .....	135
7-7	DATA Header and Payload .....	136
7-8	Example Bulk IN Burst with DATA Packet Error .....	137
7-9	Bulk OUT Transaction — Burst of One DATA packet.....	139
7-10	Bulk OUT Burst of Four DATA Packets.....	140

## List of Figures

---

7-11	OUT Burst with Large Transfer .....	141
7-12	NRDY Flow Control at Start of Transfer .....	143
7-13	OUT Transaction Burst Flow Control Example .....	144
7-14	Example Bulk OUT Burst with DATA Packet Error .....	145
7-15	Standard And Bulk Streaming Endpoints .....	147
7-16	BOT Drive Endpoints And System Resources .....	151
7-17	UASP Streaming Drive Endpoints And System Resources .....	155
7-18	UAS Non-Data Command .....	158
7-19	UASP Non-Data Sequence .....	159
7-20	UAS Read DMA Sequence .....	161
7-21	UASP Read DMA Protocol .....	162
7-22	UAS Write DMA Sequence .....	164
7-23	UASP Write DMA Sequence .....	165
8-1	Interrupt with Single DATA Packet Per Interval .....	169
8-2	Interrupt IN Burst Example with Service Interval of 64 .....	170
8-3	Interrupt IN with NRDY .....	171
8-4	Interrupt IN Transaction with EOB .....	172
8-5	Interrupt IN with Retry in Next Service Interval .....	174
8-6	Interrupt IN Retry Example .....	175
8-7	Interrupt OUT Burst .....	177
8-8	Interrupt OUT with NRDY .....	178
8-9	Interrupt OUT Burst with End of Burst .....	179
8-10	Interrupt OUT — Retry .....	180
9-1	Maximum Number of Bursts per Service Interval .....	182
9-2	Isochronous Service Interval Examples .....	183
9-3	Scheduling Options for Load Balancing within an Isochronous Service Interval ... 184	
9-4	Isochronous Timestamp Packet Delivery Example .....	185
9-5	Isochronous Timestamp Packet Format .....	186
9-6	Bus Interval Adjustment Message .....	189
9-7	Ping and Ping Response Packet Format .....	191
9-8	Isochronous IN Sequence with Bursts of Two .....	193
9-9	Isochronous IN with Burst of Two, Zero Payload, and Burst of One .....	194
9-10	Isochronous IN with Maximum Burst Size .....	195
9-11	Isochronous OUT Burst of 16 .....	196
9-12	Isochronous OUT with Four Service Intervals .....	197
9-13	ACK/DATA Header Fields Used to Manage Smart Isochronous Scheduling ...	198
9-14	Isochronous IN Smart Scheduling Example with Pings .....	200
9-15	Smart Scheduling Associated with Isochronous IN Transfers .....	201
9-16	Isochronous OUT Transactions with Smart Scheduling Example .....	203
10-1	Basic Block Diagram of USB 3.0 Hub .....	206
10-2	Hub Connectivity — Downstream and Upstream .....	208

# **USB 3.0 Technology**

---

10-3	Packet Routing — Link Layer to Link Layer .....	209
10-4	Transaction Targeting Hub .....	210
10-5	Transaction Targeting Hub with Port LMP .....	211
10-6	Route String .....	212
10-7	SuperSpeed Unicast Routing Example .....	213
10-8	Buffers Used When Forwarding Packets .....	214
10-9	DATA Payload Buffering Required by Specification .....	215
10-10	Deferred Transaction Example .....	217
10-11	Header Bits Set During Deferral .....	218
11-1	Port-To-Port Protocol And Link Layer Role .....	222
11-2	Link Layer Header Packet Processing, Key Fields .....	223
11-3	Link Layer Header Processing Elements .....	224
11-4	Link Management Elements .....	227
11-5	Header Packet Framing Ordered Set .....	232
11-6	Good Data Packet Framing Ordered Sets .....	233
11-7	Nullified Data Packet Framing Ordered Sets .....	234
11-8	Link Command Framing Ordered Sets .....	235
11-9	TSEQ Ordered Sets Train Receiver Equalizer .....	236
11-10	TS1 Ordered Set .....	237
11-11	TS2 Ordered Set .....	238
11-12	Skip Ordered Set .....	239
11-13	Loopback BERT Ordered Sets .....	240
12-1	Link Training And Status State Machine (LTSSM) .....	244
12-2	LFPS Signaling Used In LTSSM Handshake .....	251
12-3	Ordered Sets Used In LTSSM Transition Handshake .....	252
12-4	Link Commands Used For LTSSM Power Management Transitions .....	253
12-5	LTSSM U0 State .....	256
12-6	LTSSM U1 State .....	259
12-7	LTSSM U2 State .....	261
12-8	LTSSM U3 State .....	263
12-9	LTSSM Rx.Detect State .....	265
12-10	LTSSM Polling State .....	268
12-11	LTSSM Recovery .....	274
12-12	LTSSM Hot Reset .....	278
12-13	LTSSM Compliance Mode .....	281
12-14	LTSSM Loopback .....	283
12-15	LTSSM SS.Inactive State .....	285
12-16	LTSSM SS.Disabled State .....	287
13-1	Link Command Word Fields .....	290
13-2	Framed Link Command On The SuperSpeed Link .....	291
13-3	LGOOD_n Link Command On The SuperSpeed Link .....	292
13-4	LBAD Link Command On The SuperSpeed Link .....	293

## List of Figures

---

13-5	LRTY Link Command On The SuperSpeed Link .....	294
13-6	LCRD_x Link Command On The SuperSpeed Link.....	295
13-7	LGO_Ux Link Command On The SuperSpeed Link.....	296
13-8	LAU Link Command On The SuperSpeed Link.....	297
13-9	LXU Link Command On The SuperSpeed Link .....	298
13-10	LPMA Link Command On The SuperSpeed Link.....	299
13-11	LDN Link Command On The SuperSpeed Link.....	300
13-12	LUP Link Command On The SuperSpeed Link.....	301
14-1	Link Layer Tx And Rx Packet Processing Scope.....	304
14-2	Key Transmitter Packet Processing Functional Blocks .....	305
14-3	Transmitter Assigns Header Packet Sequence Number (HDR Seq#) .....	306
14-4	Transmitter Generates Header Link Control Word CRC-5.....	307
14-5	Transmitter Generates Header CRC-16.....	308
14-6	Header Packet Buffers Hold A Copy Of Each Header.....	309
14-7	Data Packet Payload Requires CRC-32 Generation.....	310
14-8	Transmitter's DPP CRC-32 Generation Logic .....	311
14-9	Transmitter Adds Ordered Set Framing To Outbound Packets.....	312
14-10	Receiver Accepts Inbound Packets And Link Commands.....	313
14-11	Key Receiver Packet Processing Functional Blocks .....	314
14-12	Receiver Checks Header Packet CRC-16.....	315
14-13	Receiver Checks Header Link Control Word CRC-5 .....	316
14-14	Receiver Checks Header Sequence Number .....	317
14-15	Receiver Header Packet Buffet Accepts Header .....	318
14-16	Receiver Checks Data Packet Payload (DPP) CRC-32 .....	319
14-17	Receiver's DPP CRC-32 Checking Logic .....	320
15-1	Asynchronous Traffic May Contribute To Link Flow Control Problems .....	323
15-2	Elements Of USB 3.0 Link Level Flow Control .....	325
15-3	Format Of LCRD_X Link Command Variants .....	328
15-4	Link Level Flow Control Logic After Reset .....	329
15-5	Flow Control Logic Initialization .....	330
15-6	The First Header Packet Is Processed And Sent .....	331
15-7	The First Header Packet Arrives At The Receiver .....	332
15-8	Rx HP Buffer Is Available, Credit Returned.....	333
15-9	Transmitter Receives A Flow Control Credit .....	334
15-10	Transmitter Receives A Valid LCRD_x .....	335
15-11	Transmitter Receives An Invalid LCRD_x.....	336
16-1	Link Errors Are Affected By The Channel.....	341
16-2	Upstream Asynchronous Packet Encounters An Error .....	342
16-3	Elements Of Link Level Header Packet Acknowledgement .....	346
16-4	Format Of LGOOD_n Link Command Variants.....	352
16-5	Format Of The LBAD Link Command .....	353
16-6	Format Of The LRTY Link Command .....	354

# **USB 3.0 Technology**

---

16-7	Link Level Header Packet Acknowledgement Logic After Reset .....	355
16-8	Initial Header Packet Sequence Number Advertisement.....	356
16-9	The First Header Packet Is Processed And Sent .....	358
16-10	Header Packet CRC-5, CRC-16 Checked.....	359
16-11	Header Packet Sequence Number Is Checked .....	360
16-12	Receiver Accepts And Acknowledges A Valid Header Packet .....	361
16-13	Transmitter Checks LGOOD_n Link Command .....	362
16-14	Transmitter Retires A Header Packet .....	363
16-15	A Header Packet Is Processed And Sent .....	364
16-16	Header Packet CRC-5, CRC-16 Check Fails, LBAD Sent.....	365
16-17	Transmitter Receives LBAD, Starts Retry Sequence.....	366
16-18	Retry Header Packet CRC-5, CRC-16 Is Checked.....	367
16-19	Retry Header Packet Sequence Number Is Checked.....	368
16-20	Receiver Accepts And Acknowledges The Retry Header Packet.....	369
16-21	Transmitter Validates LGOOD_n Received After Retry.....	370
16-22	Transmitter Retires The Header Packet After Retry.....	371
17-1	Chip-To-Chip Protocol And Physical Layer Role.....	374
17-2	Common Traffic And Data/Control Bytes .....	376
17-3	Physical Layer Logical Processing: Scope .....	377
17-4	Physical Layer Electrical Specifications: Scope .....	380
18-1	Physical Layer Tx And Rx Logical Section .....	384
18-2	Transmitter Outbound Bytes And The D/K Flag.....	385
18-3	Transmitter PHY Logic Scrambling.....	387
18-4	TS2 Scrambling Disable Bit .....	388
18-5	Transmitter PHY 8b/10b Encoding .....	389
18-6	8b/10b Encoding Supports Link CRD Scheme .....	391
18-7	Transmitter PHY Parallel To Serial Data Conversion .....	394
18-8	The Differential Transmitter .....	395
18-9	The Differential Receiver, Termination, And Equalizer .....	396
18-10	Receiver PHY Clock And Data Recovery Logic.....	397
18-11	Receiver PHY Serial To Parallel Data Conversion.....	399
18-12	COM Is Used To Achieve Symbol Lock .....	400
18-13	Receiver PHY Corrects Differential Polarity Inversion.....	401
18-14	Receiver PHY Elastic Buffer .....	402
18-15	Elastic Buffer Spans Two Receiver Clock Domains.....	404
18-16	Clocking Symbols Into And Out Of The Elastic Buffer.....	406
18-17	Transmitter Inserts Skip (SKP) Ordered Sets .....	407
18-18	Receiver PHY 8b10b Decode.....	408
18-19	Receiver PHY Decoder Use Of The D/K Flag.....	409
18-20	Receiver PHY Descrambling Hardware Details .....	410
18-21	Receiver PHY Forwards Traffic To Link Layer .....	411
19-1	Enabling VBUS Triggers PowerOn Reset.....	414

## List of Figures

---

---

19-2	3.0 Hub Receives A Request To Perform Warm Reset.....	416
19-3	Warm Reset LFPS Signaling Sent And Received .....	417
19-4	3.0 Hub Receives A Request To Perform Hot Reset .....	419
19-5	LTSSM Hot Reset State And Substates.....	420
19-6	USB 3.0 Hub Propagates Resets To Downstream Ports.....	425
20-1	An Example Server System With Short And Long USB 3.0 Channels .....	428
20-2	Link Training & Receiver Physical Layer Logic.....	430
20-3	Link Training Sequence And LTSSM States .....	434
20-4	Link Training Is Delayed Until Warm Reset Ends .....	435
20-5	Warm Reset LFPS Characteristics .....	436
20-6	Checking For A Link partner In Rx.Detect.Active.....	437
20-7	Receiver Enables & Disables Low-Impedance Termination .....	438
20-8	SuperSpeed Transmitter Detects Receiver Termination .....	439
20-9	Rx Detect Equivalent Circuits .....	440
20-10	Conserving Power In Rx.Detect.Quiet If Link Partner Isn't Present.....	442
20-11	Link Traffic Starts With Low Frequency Periodic Signaling (LFPS).....	443
20-12	Characteristics Of The Polling.LFPS Burst Sequence .....	444
20-13	Devices Send SuperSpeed TSEQ Ordered Sets For Receiver Equalization.....	445
20-14	COM Symbol Bit Pattern Is Easily Recognized By Receiver .....	447
20-15	Normal Transmitter And Receiver Differential Polarity .....	448
20-16	Differential Polarity Inversion Corrected By The Receiver .....	449
20-17	TS1 Ordered Sets Appear On The Link In Polling.Active .....	450
20-18	TS1 Ordered Set Format .....	451
20-19	TS2 Ordered Sets In Polling.Configuration Signal End Of Link Training .....	452
20-20	TS2 Ordered Set Format .....	453
20-21	Idle Symbols Confirm That The Next State Is U0 .....	454
20-22	LGOOD_n Link Command Advertises Header Packet Sequence Number .....	455
20-23	Format Of LGOOD_n Link Command.....	456
20-24	LCRD_x Link Commands Initialize Header Packet Flow Control .....	457
20-25	Format Of LCRD_x Link Command .....	458
20-26	Devices Exchange First Header Packets, Port Capability LMP .....	459
20-27	Port Capability LMP Header Format.....	460
20-28	Downstream Facing Port Sends Port Configuration LMP .....	462
20-29	Port Configuration LMP Header Format .....	463
20-30	Upstream Facing Port Sends Port Configuration Response LMP .....	464
20-31	Port Configuration Response LMP Header Format .....	465
21-1	LTSSM Recovery State, Substates, Entry, And Exits .....	468
21-2	Receiver PHY Logic Retrained During Recovery .....	470
21-3	U1-U3 Exit Followed By A Transition To LTSSM Recovery .....	471
21-4	U0 Link Layer Error Causes A Transition To LTSSM Recovery .....	473
21-5	Recovery.Active Employs TS1 Ordered Sets.....	474
21-6	COM Symbol Bit Pattern Is Easily Recognized By Receiver .....	475

# **USB 3.0 Technology**

---

21-7	Normal Transmitter And Receiver Differential Polarity .....	476
21-8	Differential Polarity Inversion Corrected By The Receiver .....	477
21-9	Recovery Configuration Employs TS2 Ordered Sets .....	478
21-10	Logical Idle Symbols Confirm That A Return To U0 Is Next .....	479
21-11	LGOOD_n Link Command Advertises Header Packet Sequence Number .....	481
21-12	Format Of LGOOD_n Link Command.....	481
21-13	LCRD_x Link Commands Initialize Header Packet Flow Control .....	482
22-1	Hub Status Change Indicator.....	489
22-2	Hub Status Change Indicator.....	491
22-3	SuperSpeed Descriptors.....	500
22-4	Descriptor Tree Containing Alternate Interface Settings.....	515
23-1	Hub and Port Status .....	529
23-2	SuperSpeed Descriptors.....	537
23-3	Hub Depth Assignment.....	556
24-1	All SS Links Remain Powered Without Link Power Management.....	564
24-2	LTSSM and Link Power States and Transitions .....	567
24-3	Link Power State Hierarchy .....	568
24-4	Inactivity Timers .....	570
24-5	Silent Transitions — LMP .....	573
24-6	LMP — Inactivity Timeout.....	574
24-7	Link Partner Handshake - U0 to U1 or U2.....	577
24-8	U1/U2/U3 LFPS Exit Signaling .....	582
24-9	Function Wake Notification Packet .....	583
24-10	Function Suspend Request — Setup Transaction Data.....	584
24-11	Path Exit Latency (PEL) and System Exit Latency (SEL) Reported Parameters.	586
24-12	System Exit Latency Calculation .....	587
24-13	Latency Tolerance Message Header .....	589
25-1	Physical Layer Tx And Rx Electrical Section.....	592
25-2	Effects Of Jitter On SuperSpeed Eye .....	593
25-3	Differential And Single-Ended Voltage Levels .....	595
25-4	SuperSpeed Transmitter Elements.....	596
25-5	Inter Symbol Interference (ISI) Example .....	600
25-6	Transmitter De-emphasis Example (Peak-to-Peak Voltage) .....	602
25-7	Comparison Of ISI With, Without De-emphasis .....	603
25-8	Triangular SSC Modulation Example .....	604
25-9	SuperSpeed Receiver PHY Electrical Elements.....	605
25-10	SuperSpeed Receiver Equalization .....	608
25-11	TSEQ Frequency Spectrum .....	609
25-12	Receiver Enables & Disables Low-Impedance Termination .....	610
25-13	SuperSpeed Transmitter Detects Receiver Termination.....	611
25-14	Rx Detect Equivalent Circuits .....	612
25-15	Transmitter And Receiver LFPS Signaling Logic.....	613

## **List of Figures**

---

---

26-1	Scope Of Three Key USB 3.0 SuperSpeed Compliance Tests.....	618
26-2	Compliance Testing Accounts For Long & Short Channels.....	633
26-3	Compliance Mode Test Configuration .....	635
26-4	Compliance LTSSM State .....	637
27-1	Loopback Master And Slave .....	640
27-2	Loopback Ordered Sets & Slave BERT Logic .....	642
27-3	Loopback LTSSM State And Substates.....	645
27-4	Loopback LTSSM Substate Requirements & Transitions .....	646
27-5	TS2 Loopback Entry Signaling.....	647
27-6	Loopback Exit Through Recovery.....	649

## **USB 3.0 Technology**

## List of Tables

---

---

1	PC Architecture Book Series .....	1
1-1	USB Bandwidth Comparison.....	9
1-2	Download Time Comparison .....	10
2-1	Motivations for USB .....	18
2-2	Standard Device Requests .....	31
2-3	Standard Features Supported for Get/Set Feature Request.....	32
2-4	USB 2.0 Maximum Payload Sizes.....	37
3-1	Plug and Respectable Compatibility.....	50
3-2	USB 3.0 Cable Assemblies .....	54
4-1	Protocol Packet Types and Characteristics .....	72
4-2	General Endpoint Characteristics .....	76
5-1	Protocol-Layer Packet Categories and Characteristics.....	81
5-2	Transaction Packet Sub Types .....	82
5-3	ACK Header Field Descriptions — Header Moving Downstream.....	84
5-4	ACK Header Field Descriptions — Header Moving Upstream .....	86
5-5	NRDY Header Field Descriptions — Header Moving Upstream .....	87
5-6	ERDY Header Field Descriptions — Header Moving Upstream .....	89
5-7	STATUS Header Field Descriptions — Header Moving Downstream .....	90
5-8	STALL Header Field Descriptions — Header Moving Upstream.....	92
5-9	Wake Notification Header Field Descriptions — Header Moves Upstream.....	93
5-10	Latency Tolerance Message Header Field Descriptions .....	94
5-11	Bus Interval Adjustment Header Field Descriptions .....	96
5-12	PING Header Field Descriptions — Header Moving Downstream.....	97
5-13	PING RESPONSE Header Field Descriptions .....	98
5-14	DATA Header Field Descriptions — Header Moving Downstream.....	100
5-15	DATA Header Field Descriptions — Header Moving Upstream .....	102
5-16	ISOCHRONOUS TIMESTAMP Header Field Descriptions.....	104
6-1	Setup Stage Request Data .....	112
7-1	SuperSpeed Endpoint Companion Descriptor.....	127
7-2	Timing Parameters for IN and OUT Transaction Protocol.....	146
7-3	Stream ID Values and Descriptions .....	149
7-4	Endpoint Companion Descriptor Streaming Support .....	150
9-1	Descriptions for the Smart Isochronous Fields .....	198
11-1	Control (K) Symbols.....	231
11-2	Loopback BRST Ordered Set.....	241
11-3	Loopback BERL Ordered Set .....	241
11-4	Loopback BCNT Ordered Set .....	242
12-1	LTSSM State Transition Time-outs .....	254
13-1	Link Command Groups.....	290
13-2	LGOOD_n Link Command Information Fields .....	293
13-3	LBAD Link Command Information Fields .....	294
13-4	LRTY Link Command Information Fields .....	295

# **USB 3.0 Technology**

---

13-5	LCRD_x Link Command Information Fields .....	296
13-6	LGO_Ux Link Command Information Fields .....	297
13-7	LAU Link Command Information Fields.....	298
13-8	LXU Link Command Information Fields .....	299
13-9	LPMA Link Command Information Fields .....	300
13-10	LDN Link Command Information Fields .....	301
13-11	LUP Link Command Information Fields .....	302
18-1	8b/10b Data And Control Symbol Encoding Examples .....	390
18-2	8b/10b Encoder Selects The Best CRD Alternative .....	393
18-3	Spread Spectrum Clocking (SSC) Requirements .....	403
20-1	TSEQ Ordered Set Symbol Encoding .....	446
22-1	Standard Device Requests .....	486
22-2	Fields Returned During GetPortStatus.....	492
22-3	Port Change Indicator Definition.....	493
22-4	Port Status Definition.....	494
22-5	Setup Stage Request Data for GetPortStatus .....	496
22-6	ClearPortConnection Request.....	497
22-7	ClearPortConnection Request.....	497
22-8	Set Address Request.....	499
22-9	Get Descriptor Request – Setup Data .....	501
22-10	SuperSpeed Descriptors.....	501
22-11	Get Device Descriptor Request .....	503
22-12	Device Descriptor Definition .....	503
22-13	Get Configuration Descriptor Request .....	506
22-14	BOS Descriptor Definition.....	506
22-15	BOS – USB 2.0 Extension Descriptor.....	507
22-16	BOS – SuperSpeed Device Capability.....	508
22-17	BOS – Container ID .....	510
22-18	Get Configuration Descriptor Request .....	510
22-19	Configuration Descriptor Definition .....	511
22-20	Interface Association Descriptor Definition .....	513
22-21	Interface Descriptor Definition.....	514
22-22	Endpoint Descriptor Definition .....	516
22-23	Endpoint Companion Descriptor Definition .....	519
22-24	GetString Descriptor Request .....	521
22-25	String Descriptor Definition.....	521
22-26	SetConfiguration Request Data .....	522
23-1	Standard Device Requests .....	524
23-2	Hub Class Requests.....	526
23-3	Fields Returned During GetPortStatus.....	530
23-4	Port Change Indicator Definition.....	531
23-5	Port Status Definition.....	532

## List of Tables

---

---

23-6	Setup Stage Data for GetPortStatus .....	534
23-7	Clear Port Connection Request.....	535
23-8	Clear Hub/Port Requests .....	535
23-9	Set Address Request .....	536
23-10	Get Descriptor Request – Setup Data .....	538
23-11	SuperSpeed Descriptors.....	538
23-12	Get Device Descriptor Request.....	539
23-13	Device Descriptor Definition .....	540
23-14	Get BOS Descriptor Request .....	541
23-15	BOS Descriptor Definition.....	541
23-16	BOS – USB 2.0 Extension Descriptor.....	542
23-17	BOS – SuperSpeed Device Capability.....	542
23-18	BOS – Container ID .....	544
23-19	Get Configuration Descriptor Request .....	545
23-20	Hub's Configuration Descriptor.....	545
23-21	Hub Interface Descriptor .....	547
23-22	Hub Status Change – Endpoint Descriptor .....	548
23-23	Hub's Endpoint Companion Descriptor Definition .....	550
23-24	Get Hub Class Descriptor Request.....	551
23-25	Hub Class Descriptor .....	552
23-26	Set Hub Depth Request.....	557
23-27	Set Port U1/U2 Timeout .....	557
23-28	Set Port Remote Wake Mask .....	558
23-29	Set Port Link State.....	558
23-30	Set Port Power.....	559
23-31	Set Port Reset State .....	559
23-32	Set BH Port Reset .....	559
23-33	Hub Feature Selectors .....	561
24-1	Link Power State Conditions .....	567
24-2	Port Link State Transitions .....	569
24-3	Set Feature - Port U1 Timeout Request – Setup Data.....	571
24-4	U1 Inactivity Timeout Value .....	571
24-5	Set Feature - Port U2 Timeout Request – Setup Data.....	572
24-6	U2 Inactivity Timeout Values .....	572
24-7	SetFeature U1 Enable and U2 Enable Requests – Setup Data.....	574
24-8	Summary of U0/U1/U2 Entry Events.....	575
24-9	U1/U2/U3 Exit Timing Parameters .....	582
25-1	GetStatus Request .....	584
25-2	System Exit Latency Request Data.....	588
25-1	Jitter Budget At Silicon Pads (Informative) .....	594
25-2	Transmitter Electrical Parameters (Normative) .....	597
25-3	Transmitter Electrical Parameters (Informative) .....	598

## **USB 3.0 Technology**

---

25-4	Spread Spectrum Clocking (SSC) Modulation .....	604
25-5	Receiver Electrical Parameters (Normative).....	606
25-6	Receiver Electrical Parameters (Informative) .....	607
25-7	Timing Requirements For Six LFPS Event Types .....	614
25-8	LFPS Electrical Specifications (Normative) .....	615
26-1	Test Assertion Examples: USB Command Verifier Compliance Test Specification 622	
26-2	List Of Test Descriptions For The USB Command Verifier Compliance Specifica- 625	
26-3	Test Assertion Examples From Link Layer Test Specification.....	627
26-4	List Of Test Descriptions For The Link Layer Test Specification .....	630
26-5	Compliance Pattern Sequence .....	636
27-1	Loopback BDAT Symbols.....	641
27-2	Loopback BRST Ordered Set.....	643
27-3	Loopback BERC Ordered Set .....	643
27-4	Loopback BCNT Ordered Set .....	644

---

---

# 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

---

---

## Decimal Notation

Numbers without any suffix are decimal. When required for clarity, decimal numbers are followed by a “d.” The following examples each represent a decimal number:

16  
255  
256d  
128d

---

## Bits Versus Bytes Notation

All abbreviations for “bits” use lower case. For example:

- 2.5 Gb/s = 2.5 Gigabits per second.
- 2 Mb = 2 Megabits.

All references to “bytes” are specified in upper case. For example:

- 10MB/s = 10 Megabytes per second.
- 2KB = 2 Kilobytes.

---

## Bit Fields

In many cases, bit fields are documented as [15:8], with this example referring to bits 8 through 15.

---

## Other Terminology and Abbreviations

**Host** — The term Host is used often by the specification and may refer to a USB Host Controller, and in some cases it extends to the CPU, memory and software that plays a role in conveying information to or from the USB devices.

**Device/Hub** — The specification routinely uses the term “USB Device” when referring to devices that are attached to a USB bus. This may include a Hub device. The term Peripheral Device is also used to clarify that a Hub is not being referenced. This text attempts to uniformly use the term Device for Peripheral Devices and use Hub for Hub Devices.

---

---

**Double Word (DWord, DW)** — DWord is used in the body of this book, and DW is employed in some diagrams where space is limited.

**Requests** — A wide range of control transfer requests are supported by USB. The specification denotes requests in a variety of ways as the examples below indicate:

- SET\_ADDRESS or SetAddress
- GET\_STATUS or GetStatus
- GET\_DESCRIPTOR (Configuration) or GetDescriptor (Configuration)
- CLEAR\_FEATURE (PortReset) or ClearFeature (PortReset), etc.

This text uses the GetDescriptor and ClearFeature convention when referring to general descriptors and feature types. When referring to specific requests for features, the following convention is used:

- GetConfiguration
- ClearPortReset

---

## Visit Our Web Site

In addition to listing all of our comprehensive courses, eLearning courses, and books, our web site ([www.mindshare.com](http://www.mindshare.com)) contains:

- Technical information covering questions asked by class participants and readers of our books
- Errata for a number of the books
- Information on MindShare training courses
- Short courses available for viewing online
- Technical papers

---

---

---

---

## We Want Your Feedback

MindShare values your comments and suggestions. You can contact us via mail, phone, fax or email.

**Phone:** (575) 373-0336 and in the U.S. (800) 633-1440.

**Fax:** (303) 957 5464

**Seminars:** E-mail — [training@mindshare.com](mailto:training@mindshare.com)

**Technical questions:** E-mail — [support@mindshare.com](mailto:support@mindshare.com)

**MindShare books:** E-mail — [purchases@mindshare.com](mailto:purchases@mindshare.com)

**Mailing Address:**

MindShare, Inc.  
481 Highway 105, Suite B-246  
Monument, CO 80132

---

---

# 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 Technology**

---

## **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.

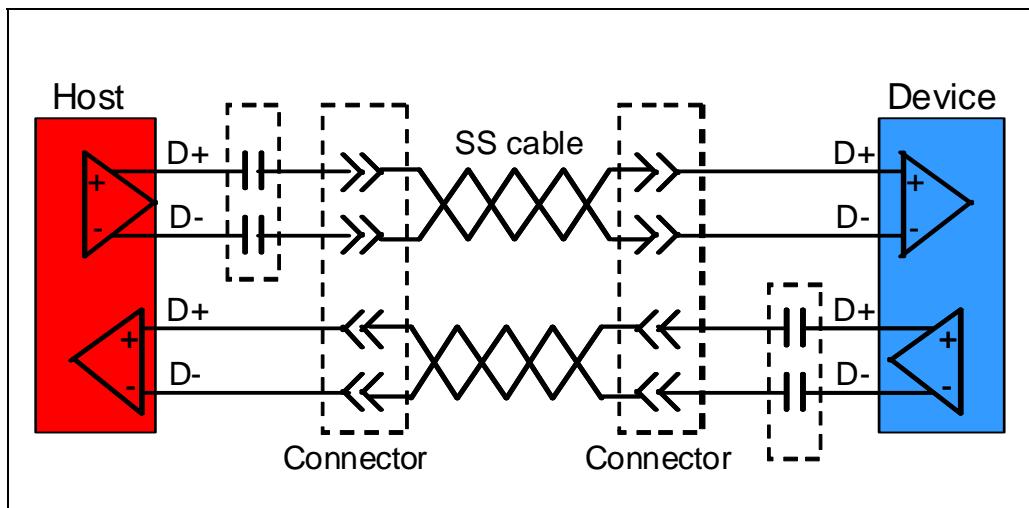
# Chapter 1: Motivation for USB 3.0

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**



# **USB 3.0 Technology**

---

## **Streaming Video and Audio**

Real time video and audio requires guaranteed bandwidth across the USB, likely via isochronous transactions. That is, data must be transferred at regular intervals. The high-speed 60MB/second of bandwidth restricts data resolution due to the restricted number of frames per second that are possible. The Super-Speed bandwidth of 5Gb/s provides much higher resolution.

## **Mass Storage**

Mass storage can also take advantage of the SuperSpeed bandwidth. Usually mass storage applications rely on low priority bulk data transfers, so actual latency is not so important. However, mass storage applications may transfer huge amounts of data, making performance a significant issue. In addition, solid state drives (SSDs) can transfer data at very high peak rates. SuperSpeed's 500MB/s bandwidth reduces file transfer times significantly when compared to many hard drive implementations.

A wide range of mass storage devices, can take advantage of the shorter Super-Speed transfer times, including: camcorders, mp3 players, rotating media drives using queueing and large caches, smart phones, and hand held computers. Table 1-2 compares download times for different USB speeds and files sizes. Notice the significantly better download times with SuperSpeed.

*Table 1-2: Download Time Comparison*

	<b>USB Flash Drive (16 GB)</b>	<b>Standard Definition Movie (6 GB)</b>	<b>High Definition Movie (25 GB)</b>
Full-Speed	~ 6 hours	~ 2 hours	~ 9.25 hours
High-Speed	~ 9 minutes	~ 3.25 minutes	~ 14 minutes
SuperSpeed	~ 54 seconds	~ 20 seconds	~ 70 seconds
Source: USB-IF			

# **Chapter 1: Motivation for USB 3.0**

---

## **Improved Protocols**

SuperSpeed USB employs two new protocols that improve performance and reliability:

- End-to-End Protocols — these protocols are related to the USB 2.0 transaction protocols model commonly called the Token/Data/Handshake protocols. The USB 3.0 specification uses the term “Protocol Layer” to describe the end-to-end protocols.
- Port-to-Port Protocols — these protocols relate to the new SS link interface that are similar to other Gigabit serial interfaces, such as PCI Express.

The collection of features associated with the new SS protocols improve overall performance and efficiency, and are highlighted in the following sections.

---

## **End-to-End Protocols**

This section discusses the improvements made by the SuperSpeed protocols over the USB 2.0 Token/Data/Handshake protocols. The SuperSpeed protocols improve or eliminate the following shortcomings. New features have also been added that improve functionality, performance and efficiency.

The packets that are exchanged between a Root port on the host controller and a SS device are called Protocol-Layer packets. These packets (with one exception) are all 16 bytes in size and called Header packets. The DATA header packet has an associated Data Packet Payload (DPP) that may range from 1 - 1024 bytes in size.

### **Token/Data/Handshake Shortcomings and Improvements**

Four primary limitations of the USB 2.0 protocols are enumerated below:

1. The three packet Token/Data/Handshake sequence is inefficient.
2. USB 2.0 uses a shared broadcast bus that increases power consumption.
3. Polled flow control uses the inefficient NAK protocol.
4. Error handling is limited to three attempts and upon failure, software is notified to manage the problem.

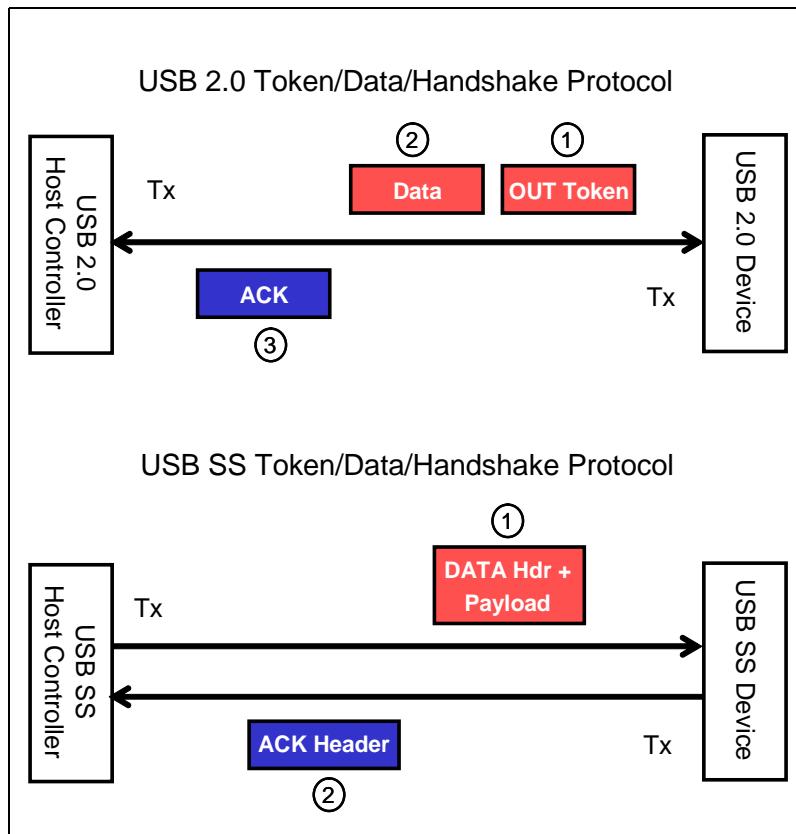
These shortcomings are improved by the SS implementation as described below.

# USB 3.0 Technology

---

**Token/Data/Handshake is Inefficient.** A USB 2.0 transaction typically uses an exchange of three packets (except for isochronous transactions that don't use a handshake packet). SS uses a more efficient two packet per transaction approach, thereby reducing overhead. Figure 1-2 illustrates a USB 2.0 OUT transaction (upper example) and the same OUT transaction is shown using SS protocol (bottom example). The 2.0 OUT transaction begins with the host controller sending a Token packet followed by a DATA packet. The target device responds with an Acknowledge (ACK) packet that confirms successful data delivery. The SuperSpeed transaction begins with the host controller sending a DATA header packet that incorporates the token information and is immediately followed by the Data Packet Payload (DPP).

Figure 1-2: USB 2.0 vs SS OUT Transaction Protocols



## Chapter 1: Motivation for USB 3.0

---

**Broadcast Bus Increases Power Consumption.** USB2.0 broadcasts packets to all devices that may be targeted by a given transaction based on a common speed. For example, when a high-speed token packet is delivered, all USB high-speed links are activated as the packet is sent to all high-speed devices. Each device decodes the address to determine if it's being accessed. If the transaction is an OUT, then data would also be broadcast to these same devices and only the target device would return the handshake packet. If the transaction is an IN, then the target device would return data and the host controller would broadcast the handshake packet to all the devices.

SuperSpeed transactions are unicast directly to the target device and only the links in the path between the root port and target device will see the transaction. Routing information is included within the packet and used by SS hubs to forward the packet to the specified downstream port.

**Polled Flow Control.** The USB 2.0 polling protocol results in very inefficient use of the bus. For example, a typical interrupt endpoint must be polled at the required rate to check if an interrupt is pending. Only when an interrupt is pending will data be returned and acknowledged (via an ACK packet). More typically no interrupt is pending, resulting in repeated polling operations that each return a NAK flow control packet. All other endpoint types (except Isochronous) are also polled, often resulting in large numbers of NAK packets due to successive retries.

Rather than performing repeated retries until a device is ready, SS protocol accesses an endpoint once and if not ready it will send a Not Ready packet (NRDY). When the host controller receives the NRDY packet it deactivates the transaction and waits for the device to return an Endpoint Ready (ERDY) packet, which notifies the host controller to send the transaction again. This could be called a poll-once protocol because there is only one retry performed. In some cases, an event triggers the device to deliver an ERDY thereby notifying the host controller that the device is ready to be serviced, even though the host has not accessed the device.

**Error Handling and Reporting.** USB 2.0 provides for packet error checks and upon failure a transaction retry is performed. Three attempts are allowed for a transaction to complete successfully, but if all three attempts fail then the error is reported to software for handling. Super-Speed end-to-end protocol performs these same checks and implements the same three attempts to transfer the data payload.

# **USB 3.0 Technology**

---

## **Data Bursting**

Another improvement associated with the SS end-to-end protocols is data bursting. SuperSpeed protocol requires that each DATA packet payload be acknowledged with a ACK header packet. Data Bursting permits multiple DATA payloads to be sent prior to the receipt of the first ACK packet. The maximum number of DATA payloads that can be sent prior to receiving an ACK is specified by the Max Burst value defined in the Endpoint Companion descriptors. The effect of bursting is reduced latency when sending or receiving data. The maximum allowed burst size is 16 with 1024 Bytes per DATA payload, or 16 KBs.

## **Bulk Streaming**

Bulk Streaming is called so because this optional feature is only available to bulk endpoints. The goal of the bulk streaming protocol is to manage multiple streams of data between bulk endpoint buffers and corresponding buffers located in main memory.

Standard bulk endpoints have a single buffer for transferring information to or from a single buffer in main memory. Streaming expands the number of buffers associated with a single bulk endpoint and associated main memory to nearly 64k. This is accomplished by targeting a bulk endpoint as usual and then using a 16-bit stream ID (located within the header packets) to select an endpoint buffer to be used for a given transfer. The stream ID value is also used by the host controller to select a corresponding buffer in main memory.

---

## **Port-to-Port Protocols**

As with other serial interfaces that operate in the Gigabits/s transfer rate, SS protocols are defined to ensure successful and efficient movement of Protocol packets across each link. SS performs packet error checks as a packet traverses each link, and each packet must be transferred successfully before being forwarded to the next link. There is no parallel between the SS port-to-port protocols and USB 2.0 token/data/handshake protocols. Three primary features of the port-to-port protocols are:

- Link Flow Control
- Link Error Detection and Recovery
- Link Power Management

# **Chapter 1: Motivation for USB 3.0**

---

## **Link Flow Control**

Link flow control uses a credit-based mechanism to track buffer space availability within the link partner. Link flow control operates in both direction as each link partner reports link credits to the other. See Chapter 15, entitled "Header Packet Flow Control," on page 321 for details associated with link flow control.

## **Link Error Detection and Recovery**

The link error detection and recovery mechanism verifies that each header packet transfers successfully across the link and ensures headers are delivered in the order received. Depending on the nature of the error, the packet will either be retried repeatedly until the transfer succeeds or after a specified time-out period the link may be retrained. Details of the link error detection and recovery mechanisms are discussed in chapter entitled, "Link Errors & Packet Acknowledgement on page 339."

---

## **Power Management**

A major goal of the USB 3.0 specification is to reduce power consumption across all platform implementations. To this end, a variety of changes have been implemented. Note that effective platform power management is also heavily dependent on the capabilities of the USB 3.0 host controller.

---

## **SuperSpeed Bus Power Management**

SuperSpeed power management involves aggressively shutting down links that are not in use and reducing device function power under software control.

## **Broadcast Versus Unicast Bus**

USB 2.0 ports supply power to all attached devices and forward packets using a broadcast mechanism. All devices must decode the packet's address to determine if they are the recipient of a transaction, thereby consuming power even when they are not the target device. This of course causes additional power consumption.

SuperSpeed uses a unicast bus that involves only the links in the direct path between the root port and target device. All other links can enter the electrical idle state and reduce power significantly.

# **USB 3.0 Technology**

---

## **Link Power Management**

The USB cable distributes 5-volt power to all devices and current ranges from 100ma to 500ma for USB 2.0 and 150ma to 900ma SuperSpeed. The fundamental method for conserving power is the simple suspend and resume protocol used by both USB 2.0 and SuperSpeed. The SS bus adds link power management to provide finer granularity thereby reducing power consumption as follows:

- U0 — Link is fully powered and operational
- U1 — Standby, link in electrical idle with fast recovery to U0
- U2 — Standby, link in electrical idle with slow recovery to U0
- U3 — Suspend, link in electrical idle with very slow recovery to U0 and all functions within the device must be suspended such that no more than 2.5ma of current is being drawn from the bus.

Note that transitions to the standby states (U1 and U2) may be managed by hardware. Entry into U3 may only be initiated by software. Details of the SS power savings mechanisms can be found in Chapter 24, entitled "SuperSpeed Power Management," on page 563.

## **Function Power Management**

This feature permits individual functions within a single device to be suspended, while the link remains active to handle other functions. Note that a single-function device may also be placed in a suspended state using this feature. Function power management can be activated only under software control.

---

## **System Power Improvements**

The 2.0 host controllers poll memory periodically to fetch and execute transfer descriptors on a regular basis. This frequently prevents chipsets, main memory, and the host processor from going into low power states. This makes USB a major obstacle when attempting to conserve system power.

A USB 3.0 host controller may reduce overall system power by accessing memory more efficiently. For example, an xHCI host controller doesn't poll main memory to fetch work. Instead, the xHCI relies on the software driver to post work to be performed and then notify the host controller by writing to internal "doorbell registers". These doorbell registers identify the device endpoint requiring service and indirectly specify the memory location containing the work to be performed. The doorbell mechanism reduces traffic to/from memory, allowing system components to more readily enter low-power states.

---

---

# 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.

## **USB 3.0 Technology**

---

*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	<ul style="list-style-type: none"><li>• Low-speed — 1.5Mb/sec focusing on keyboards, mice, etc.</li><li>• Full-speed — 12Mb/sec focusing on.</li><li>• High-speed — 480Mb/sec focusing on mass storage, headphones, printers, etc.</li></ul>
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

## **Chapter 2: USB 2.0 Background**

---

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.

---

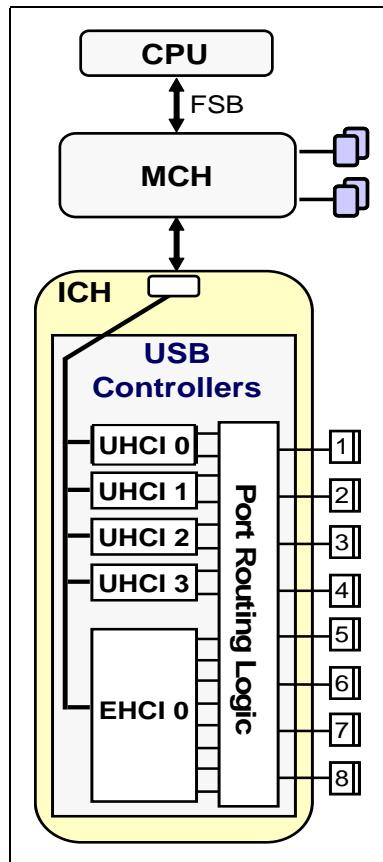
### **USB 2.0 Companion Controllers**

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.

## USB 3.0 Technology

*Figure 2-1: Eight-Port Companion Controller Built into Chipset*



---

### **Host Controller Registers**

USB Host Controllers initiate all USB traffic. Most computers ranging from the smallest mobile systems to the largest servers use PCI software to configure devices within the chipset, including USB host controllers. Figure 2-2 expands on the previous illustrations by depicting an actual chipset implementation.

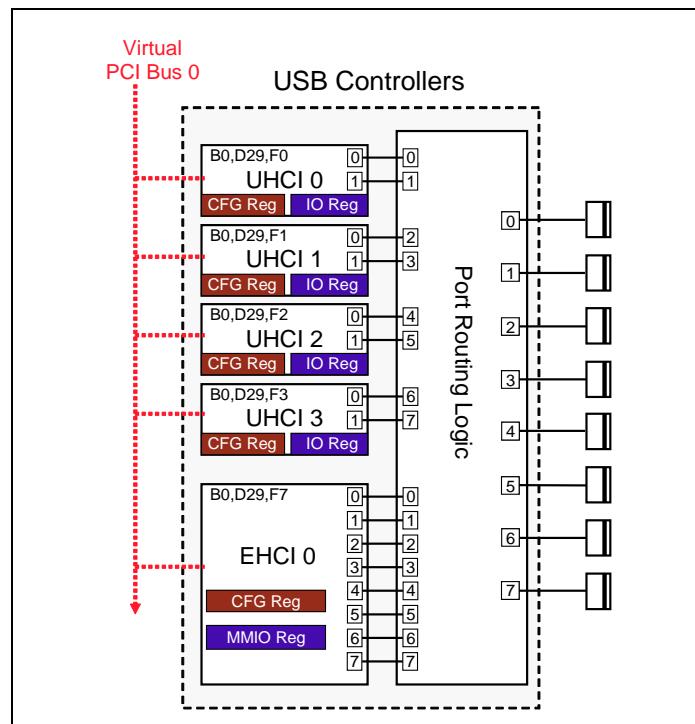
## Chapter 2: USB 2.0 Background

PCI software scans the PCI configuration space by referencing Bus, Device and Function numbers that are implemented by the chipset. For example, Figure 2-2 shows that UHCI 0 has a PCI configuration address of:

*Bus 0 (B0), Device 29 (D29) and Function 0 (F0)*

UHCI 0 - UHCI 3 all share the same bus and device number but each is assigned a unique function number so they can be accessed independently. Notice also that the EHCI also shares Bus 0 and Device 29 and is assigned Function 7.

*Figure 2-2: Companion Controller and Register Sets*



## **USB 3.0 Technology**

---

There are registers associated with each controller and include:

- **PCI Configuration Registers** — these registers, accessed via PCI configuration software, report PCI characteristics and allow software to assign address space for the memory-mapped (EHCI) and IO-mapped (UHCI) controller register blocks, as well as supporting capability registers for Power Management, Interrupt Handling, Advanced Error Reporting, etc.
- **Controller Register Blocks** — these registers reside within the host controller and are mapped into the PCI address space. The registers are accessed by the host controller driver, which has knowledge of the host controller implementation. Collectively the register blocks allow software to set up and manage functions such as:
  - Pointers to USB memory data structures (transfer descriptors, etc.)
  - Power Management policies
  - Error Handling policies
  - Interrupt management
  - Port Status and Control

See MindShare's USB 2.0 System Architecture book for details regarding the UHCI and OHCI register block implementations. See also details regarding EHCI online at <http://www.mindshare.com/learn/?section=11E3171000B5>.

---

## **USB Hubs**

Two primary types of USB hubs (root and external) expand the USB bus by providing additional ports to which devices can attach.

### **Root Hubs**

Figure 2-3 on page 23 shows an example implementation of Root Hub ports that reside within a USB 2.0 host controller. Root hubs perform a variety of actions that are controlled and/or reported via the Port Status and Control registers for each port. This allows software to:

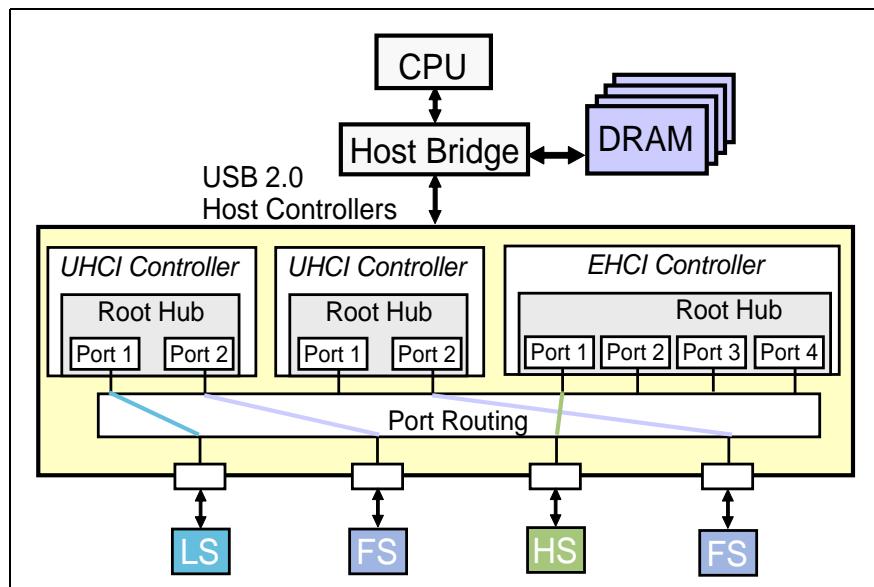
- Apply/remove port power
- Verify device attachment/removal
- Check device speed
- Reset the port and attached device
- Check for over-current conditions
- Perform tests
- etc.

## Chapter 2: USB 2.0 Background

Root hub port registers are typically mapped into the address space of the host bus interface to which the controller is attached (e.g., the PCI bus).

Port routing logic assures that attached devices are associated with the proper host controller and root hub port. The device and host controller type defines the required association as illustrated in Figure 2-3.

Figure 2-3: Example Companion Controller with Root Hubs



### External Hubs

External hubs get detected, reset, assigned a device number and configured like other USB devices. However, the hub function is similar to root hubs.

USB 2.0 supports FS and HS hubs as illustrated in Figure 2-4 on page 24. As discussed previously, full-speed hubs support attachment of low- and full-speed devices only, and if a high-speed device is attached to a full-speed hub it must also function as a full-speed device, but is not required to support full functionality. High-speed hub ports support attachment of LS, FS, and HS devices.

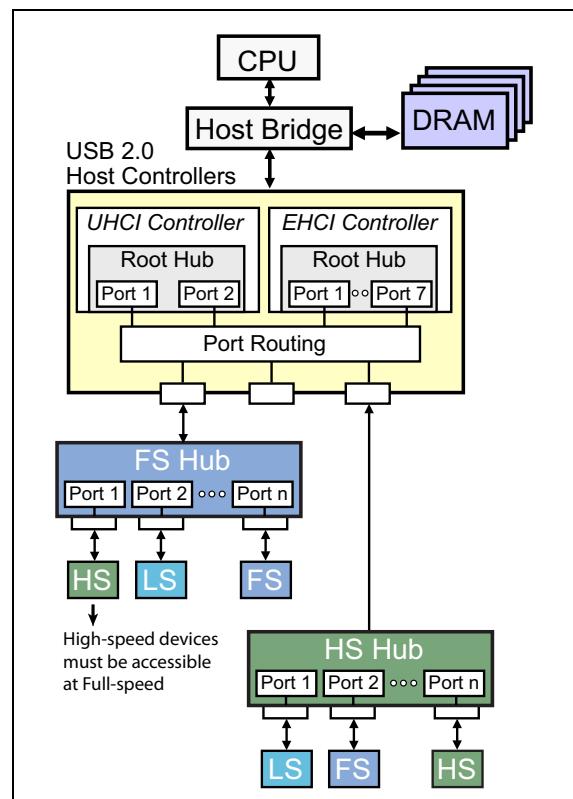
## USB 3.0 Technology

Like the root hub ports, external hub ports have dedicated port status and control registers permitting software to:

- Apply/remove port power
- Reset the port and attached device
- Determine device speed
- Check for change in port status
- Read Port Status
- Set Port Power
- Set Port Suspend, etc.

Host software periodically polls external hubs to detect status change events, and if an event has occurred software issues a control transfer request to read status information to determine the source of the event. Events may be related to the hub or to a specific port. Other commands may be issued to force the hub or port to take a specific action (e.g., Set Port Suspend).

*Figure 2-4: External Full-Speed and High-Speed Hubs*



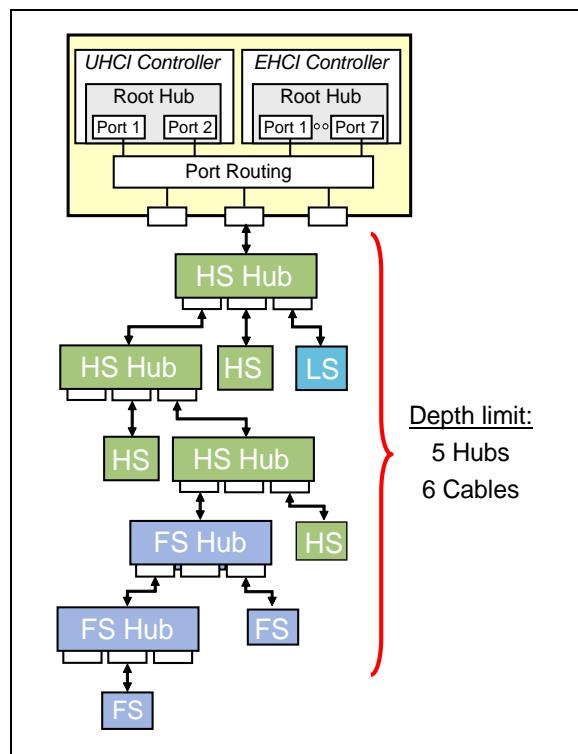
## Chapter 2: USB 2.0 Background

### Hub Depth Limits

The number of hubs that can reside between a root port and the furthest device downstream is limited to five and the maximum number of cables is six. Each cable can be up to 5 meters in length for a total of 30 meters. See Figure 2-5.

The hub depth limit imposes a maximum or worst-case round trip delay associated with the protocol. In short, when a packet is sent and a subsequent packet expected in response, the returning packet should be received within the round-trip delay, which includes the device response time. If the packet does not return within the maximum delay period it must be assumed that an error will be detected.

Figure 2-5: Hub Depth



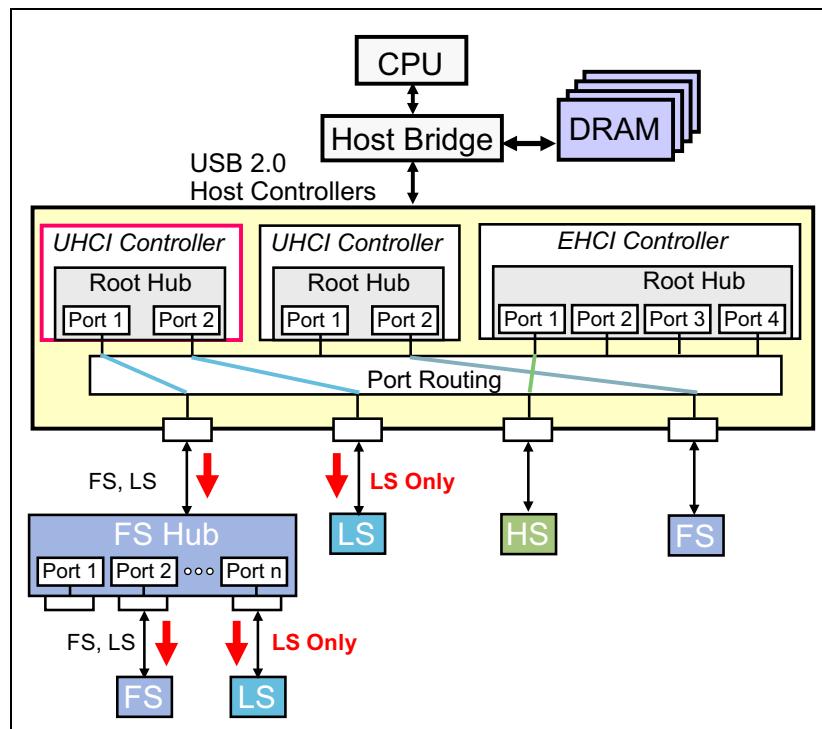
# USB 3.0 Technology

## Broadcast Transactions

The USB 2.0 broadcast scheme simplifies the design of the host controllers and hubs. The nature of broadcast traffic depends on the host controller type: full-speed or high-speed.

FS Hubs forward both LS and FS traffic, but block downstream traffic as needed to assure devices never see speeds exceeding their capabilities. Figure 2-6 on page 26 illustrates a low-speed packet being broadcast to all ports simultaneously. However, when a FS transaction is delivered only FS devices receive the packet.

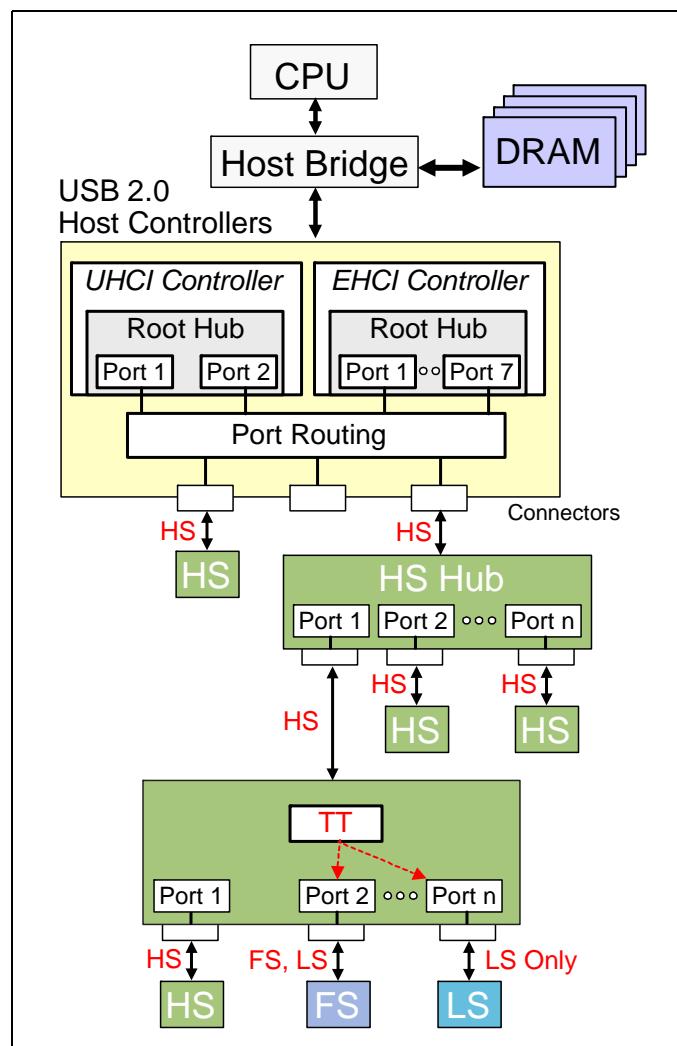
Figure 2-6: Low-Speed Packet Broadcast



## Chapter 2: USB 2.0 Background

High-speed hubs receive only HS traffic and repeat this traffic to all downstream ports that have HS devices attached. Some HS packets received by hubs may be targeting low- or full-speed devices attached to the hub's downstream ports. A transaction translator within 2.0 hubs handle the low- and full-speed transactions without significantly impacting HS bandwidth. See Figure 2-7.

Figure 2-7: HS Broadcast and HS Split Transactions



# **USB 3.0 Technology**

---

## **Device Architecture**

A major aspect of device architecture is the definition of the device's class (e.g., mass storage, audio, human interface device, etc.). The USB Implementers Forum (USB-IF) defines each device class via separate specifications. These class specifications detail the collection of endpoints that represent the programming interface to the device and may also define class-specific control transfer requests along with data formats and other requirements.

Each device reports its class and other pertinent characteristics through a collection of standard device descriptors that host software reads to determine the device type and required resources. The standard descriptors include:

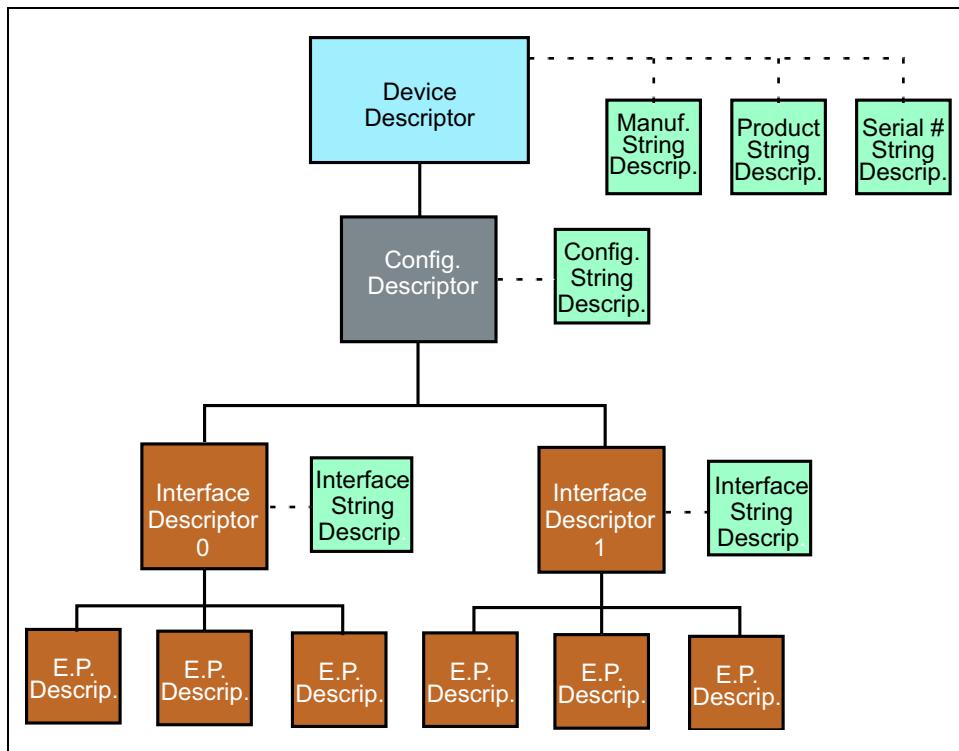
- Device Descriptor — describes the number of configurations supported by the device, vendor-specific information such as manufacturer, serial number, etc., and the device class information.
- Configuration Descriptor(s) — specifies the number of interfaces used and defines attributes associated with this configuration, including source of power, maximum power consumption and remote wakeup capability.
- Interface Descriptor(s) — defines the number of endpoints related to the interface and define attributes associated with the interface, including the number of endpoints used by this interface.
- Endpoint Descriptor(s) — specifies the attributes associated with a given endpoint along with information needed by host software to determine how the endpoint should be accessed.
- String Descriptor(s) — optional descriptors consisting of a UNICODE string that provides human-readable information that can be displayed.
- Class-Specific Descriptor(s) — a given device class may require additional descriptors as defined by a particular device class specification. One example is the hub class descriptor.

Host software reads the descriptor information and creates memory structures for managing access to each of the endpoints. In addition, software locates the appropriate class driver based on information read from the descriptors. The class driver in turn initializes the device for operation.

## Chapter 2: USB 2.0 Background

---

Figure 2-8: Standard Device Descriptor Tree



---

## Endpoints and Buffers

Each device can have a maximum of 31 device endpoints:

- All devices must support EP0 (the bi-directional default control endpoint)
- All remaining endpoints are device-specific and implemented as needed
- Endpoints (except Control) are unidirectional, thus a pair of endpoint addresses are required for bidirectional data transactions

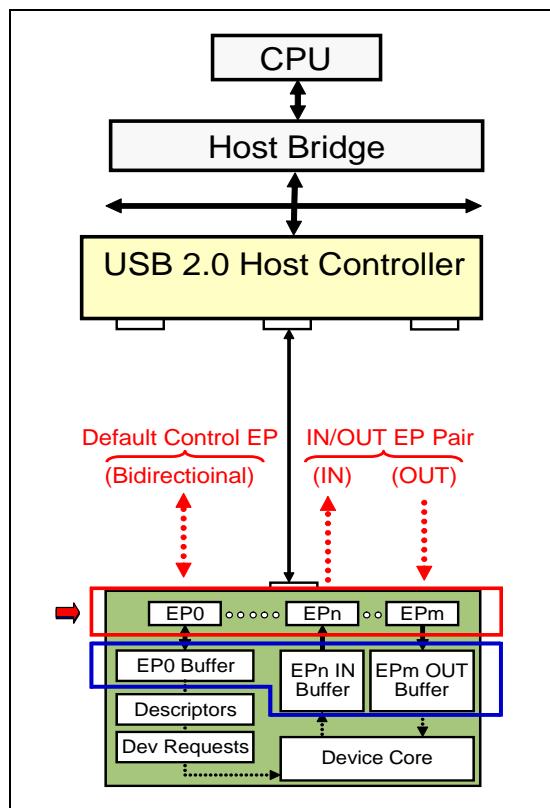
Each endpoint must implement a buffer that supports at least one *maximum-sized* packet as defined by the device's endpoint descriptor. See Figure 2-9.

Device endpoints provide the interface needed to interact with the USB device. That is, the collection of endpoints defined by a device represent the programming interface for controlling and monitoring a peripheral function. For exam-

## USB 3.0 Technology

ple, legacy mass storage devices implement one bulk IN endpoint and one OUT endpoint for handling all SCSI commands, all of which are transported via USB data packets.

*Figure 2-9: Device Endpoints and Associated Buffers*



### **Control Endpoints**

Control Transfers targeting endpoint 0 (called the default control endpoint) deliver standard and class-specific device requests. Every USB device must implement endpoint zero and support most of the standard requests. These transfers use the token-data-handshake scheme but require a sequence of two or more transactions to complete a single control transfer.

## **Chapter 2: USB 2.0 Background**

---

The Default Control Endpoint supports a wide range of activities including:

- Resetting devices attached to each port, thus forcing devices to a known state.
- Reading device capabilities via descriptors located in each device
- Assigning a unique address to each USB device
- Configuring each device
- Performing control and status activities for many USB device classes
- Reading hub port status information in response to a status change event
- Controlling power management activities (suspend and resume)
- and many others

---

### **Standard Requests**

USB devices must support many of the USB standard device requests. Table 2-2 lists and describes the USB 2.0 standard requests and indicates those that are required by all USB devices:

*Table 2-2: Standard Device Requests*

Request	Description	Required?
ClearFeature	Clears the selected feature	Yes
GetConfiguration	Fetches the current configuration setting from the device	Yes
GetDescriptor	Access the descriptor information (i.e., device capability information) stored within the device	Yes
GetInterface	Fetches the current Interface setting for devices that support alternate interface settings	No
GetStatus	Fetches the status information from the device, interface or endpoint	Yes
SetAddress	Assigns the 7-bit device address to which a device will respond	Yes
SetConfiguration	Assigns the 8-bit configuration value specified within the device configuration descriptor	Yes
SetDescriptor	Downloads descriptor data into the selected Device descriptor.	No

# **USB 3.0 Technology**

---

*Table 2-2: Standard Device Requests (Continued)*

<b>Request</b>	<b>Description</b>	<b>Required?</b>
SetFeature	Sets the selected feature (See Figure 2-3 on page 32)	Yes
SetInterface	Sets the selected alternate interface setting within a device	No
SyncFrame	Passes a Frame number to an isochronous endpoint	No

*Table 2-3: Standard Features Supported for Get/Set Feature Request*

<b>Feature Selector</b>	<b>Description</b>	<b>Required?</b>
DEVICE_REMOTE_WAKEUP	Enables or disables the ability of a device to initiate remote wakeup from the suspended state	No
ENDPOINT_STALL	Clears or sets a stall condition for the selected endpoint	Yes
HIGH-SPEED TRANSCEIVER TEST	Sets one of five test modes for compliance testing	Yes

Control endpoints other than endpoint 0 may be implemented, but are rarely used. Applications (Device Classes) that require the use of a control endpoint typically request ownership of the default control endpoint from USB system software.

---

## **Optional Endpoints**

In addition to control endpoints the three remaining endpoint types carry data payloads that are characterized by the application performing the transfer. That is, the nature of the data payload is based on application requirements. Each of the three data transfer types have the following characteristics:

- **Bulk Endpoint**
  - Used for general data movement for mass storage, printers, etc.
  - Handshake verifies correct data delivery
  - No latency guarantees and can be slow under worst case conditions
  - Not available for low speed devices

## **Chapter 2: USB 2.0 Background**

---

- **Interrupt Endpoint**
  - Used for deterministic latency servicing: mouse, keyboard, etc.
  - Handshake verifies correct data delivery
  - Accessed by host at guaranteed service *intervals*
- **Isochronous Endpoint**
  - Used for streaming data applications such as audio, video, etc.
  - Guaranteed bandwidth, no handshake to verify correct data delivery
  - Accessed by host at predetermined intervals with guaranteed data rate
  - Not available for low speed devices

---

## **USB 2.0 Packets and Protocol**

Prior to discussing packet definition and the USB transaction protocol, it might be helpful to define important USB terms:

- **Transfer** — A transfer is the movement of a block of requested data by a software client. Data movement is between a USB device and main memory. The amount of data to be moved may be less than, equal to or greater than the maximum transaction payload size for the target device. Consequently, one or more bus transactions may be needed to complete a transfer.
- **Transaction** — Transactions consist of the movement of a single packet of data between a USB device endpoint (data buffer) and main memory. Consequently, transfers that are less than or equal to the maximum transfer size will complete in a single transaction. When transfers are larger, multiple transactions are needed to complete a single transfer.
- **Pipe** — The communications channel between a device endpoint and a memory buffer supplied by software is termed a communications pipe. Most pipes are unidirectional and transfer information in a single direction (IN or OUT).

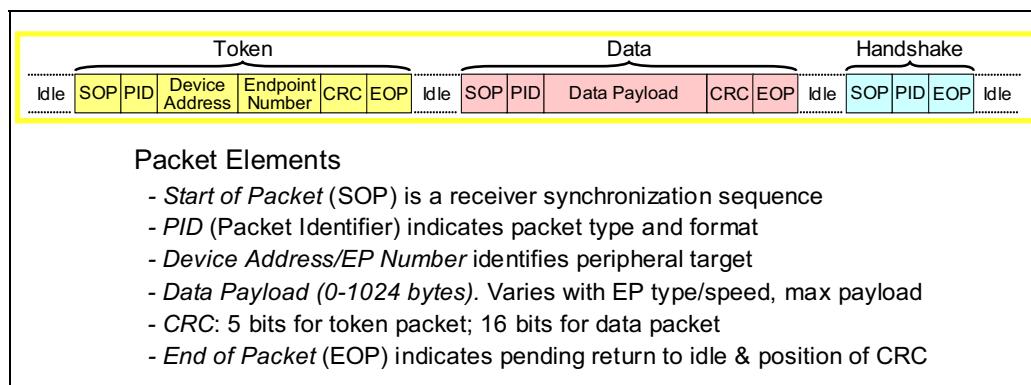
The primary focus of this chapter is on individual transactions that deliver data between the host controller and device endpoints. In addition, there are many variations associated with the basic protocol and those variations are also discussed in this section.

# USB 3.0 Technology

## Token-Data-HandShake Packet Protocol

All endpoint types except isochronous use three packets (token, data and handshake) to complete a transaction successfully as illustrated in Figure 2-10. Isochronous transactions require only the token and data packets. Every USB transaction begins with the host controller broadcasting the token packet and depending on direction of data transfer (IN or OUT) either the device or host sends the data and the recipient returns the handshake.

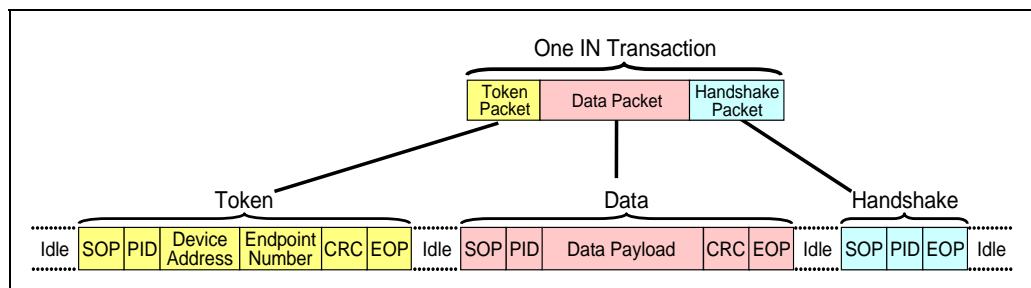
Figure 2-10: Token-Data-Handshake Packet Types and Elements



## IN Transactions

Figure 2-11 illustrates an IN transaction during which data is read from the device's endpoint buffer and delivered to the host controller. The host controller upon receiving the Data packet without error must respond with an ACK packet to confirm success of the transaction.

Figure 2-11: Example IN Transaction FS or HS

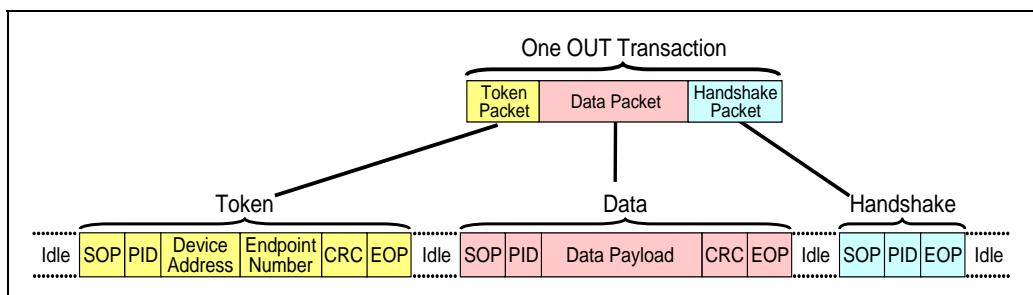


## Chapter 2: USB 2.0 Background

### OUT Transactions

Figure 2-12 on page 35 illustrates an OUT transaction during which data movement is from the host to the device. The device, upon receiving the Data packet without error, must respond with an ACK packet to confirm success of the transaction.

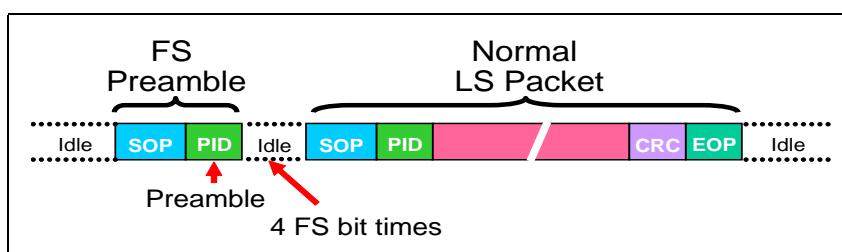
Figure 2-12: Example OUT Transaction FS or HS



### Low-Speed Transaction Variation

By default all hub ports must block all traffic to which low-speed devices are attached. Only when a hub port knows that a low-speed packet is being sent will it allow transactions to pass. The mechanism that notifies hubs that low-speed traffic is being sent is the Preamble packet. Figure 2-13 on page 35 depicts delivery of a low-speed packet from the host. The preamble packet is delivered at full speed to notify all ports that have low-speed devices attached to enable their repeaters. The host must allow a minimum of 4 full-speed bit times for the hub ports to enable their repeater prior to sending the low-speed packet.

Figure 2-13: Low-Speed Packet from Host Controller



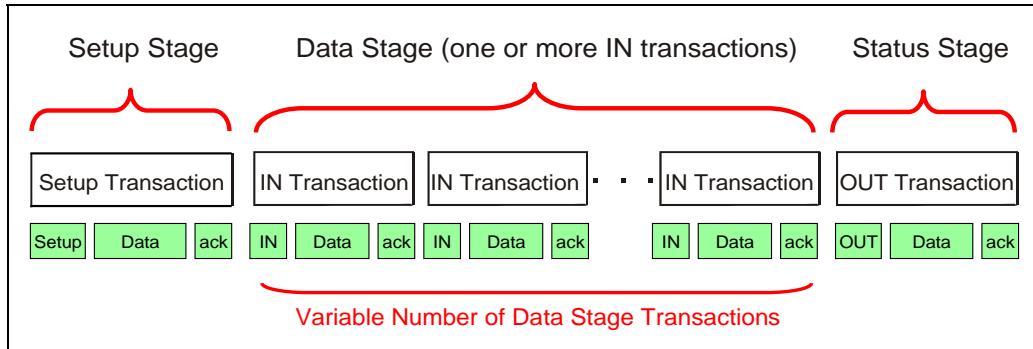
# USB 3.0 Technology

## Control Transfer Stages

- **Setup Stage** — consists of a single transaction that is a special version of an OUT transaction (with its own token PID). The setup transaction always includes an 8 byte Data packet containing the request information.
- **Data Stage** (optional) — this stage consists of one or more IN or OUT transactions for moving data between the host and device. Many device requests do not involve the movement of data, thus the Data Stage is not used in these instances.
- **Status Stage** — All Control transfers conclude with a Status Stage, consisting of a single IN or OUT transaction with a 0-byte payload.

**Three-Stage Transfers.** The data stage is included in a three-stage control transfer. The data stage can consist of a single transaction or a multiple transactions depending on the data payload packet size (up to 64 bytes) and the data transfer size (up to 64KB).

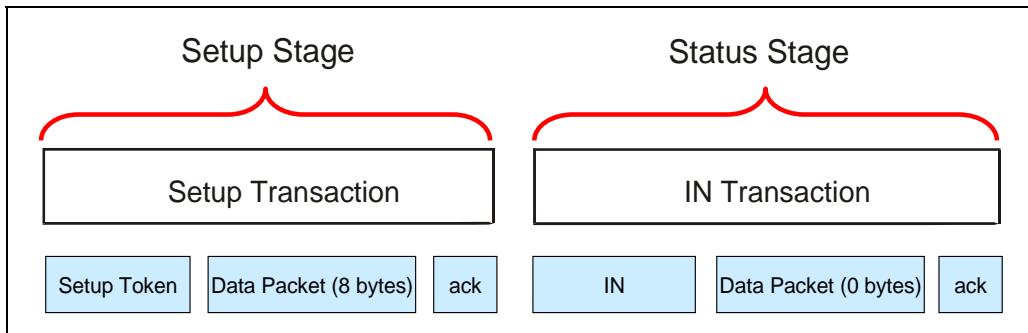
Figure 2-14: Three Stage Control Transfer (Read)



**Two-Stage Transfers.** These transfers omit the data stage because the command or information being delivered is incorporated into the Setup stage (See Figure 2-15). For example, a SetAddress request includes the 7-bit address inside the 8-byte data payload of the setup transaction.

## Chapter 2: USB 2.0 Background

Figure 2-15: Two-Stage Control Transfer



### Maximum Payload Sizes

Table 2-4 lists the maximum payload size for each USB 2.0 endpoint type and speed. In the LS column, the NA entries indicate that bulk and isochronous endpoints are not supported for low-speed devices.

Table 2-4: USB 2.0 Maximum Payload Sizes

Endpoint Type	Maximum Data Payload Size		
	LS	FS	HS
Bulk IN/OUT	NA	64 Bytes	512 Bytes
Interrupt IN/OUT	8 Bytes	64 Bytes	1024 Bytes
Isochronous IN/OUT	NA	1023 Bytes	1024 Bytes
Control (Data Stage) IN/OUT	8 Bytes	64 Bytes	64 Bytes

### HS Ping Protocol - For OUT Transactions

During OUT transaction, the host sends an OUT token packet followed by the data packet. If the target endpoint buffer cannot accept the data, the data must be discarded and the transaction must be retried. Low- and full-speed USB Bulk and Control OUT transactions suffer a bandwidth penalty when a NAK handshake is returned, because the data sent from the host must be discarded by an endpoint buffer if it cannot accept the data. This penalty would have been even

## USB 3.0 Technology

more severe for high-speed devices due to the larger data payload size. Figure 2-16 on page 38 illustrates this potential problem with a high-speed OUT transaction example. In the example an OUT transaction with a 512 byte payload is delivered to an endpoint that cannot accept the data. The endpoint has a full buffer and cannot store the data, so the data is simply dropped and a NAK is returned. This type of behavior is not uncommon and results in large amounts of bandwidth being wasted.

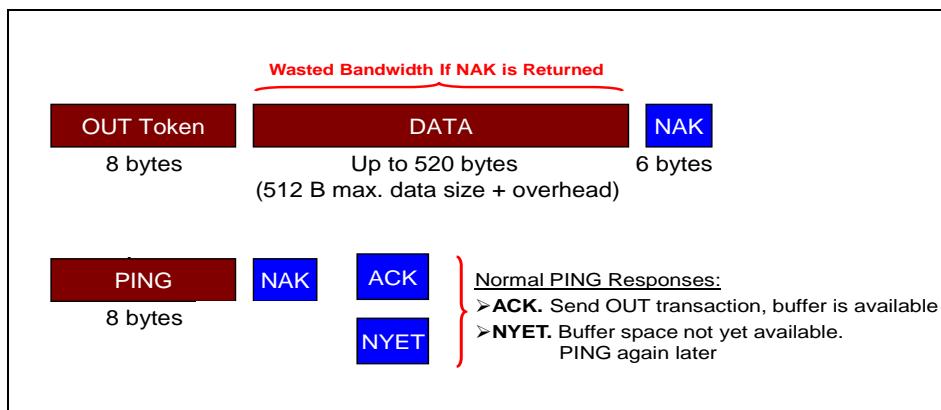
A solution to this problem was devised and released in the USB 2.0 specification, but it is only defined for high-speed devices. Rather than sending large data payloads that are rejected time after time, the Ping protocol requires first checking whether a device is ready to accept the data before sending it. There are two typical responses to a ping packet:

- NYET (Not Yet), telling the host that the target endpoint is not ready to receive data
- ACK (Acknowledge), notifying the host that it is ready to receive the OUT transaction.

The Ping protocol also defines how often a ping packet should be sent so that bandwidth is not wasted by sending back-to-back-to-back ping packets.

Figure 2-17 illustrates an example ping protocol sequence where the first ping response is a NYET packet indicating the device is not ready to accept data. The second ping results in an ACK indicating the device is ready to accept the data, and the host then performs the transaction.

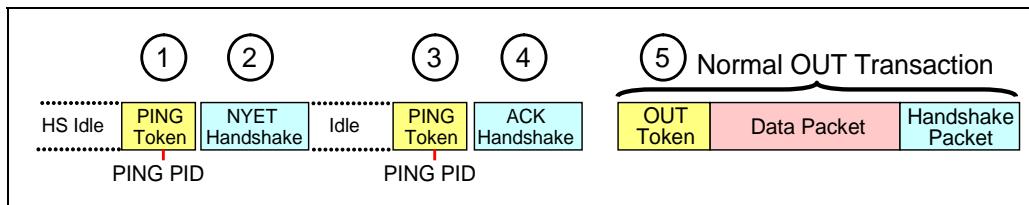
Figure 2-16: OUT Data NAK vs Ping Protocol Solution



## Chapter 2: USB 2.0 Background

---

Figure 2-17: Ping Protocol Sequence



---

### HS Split Transaction Protocol

All traffic to and from high-speed hubs is performed at high speed. If a high-speed device is being accessed the Hub Repeater will forward the HS transaction to all downstream ports to which high-speed devices are attached. If a low- or full-speed device is attached host software directs transactions to these devices by performing split transactions. The HS hub includes a Transaction Translator that emulates a full-speed host controller by performing transactions at the native speed of the target device. Figure 2-18 on page 40 illustrates the primary blocks associated with a high-speed hub. Each block performs the following actions:

- Repeater
  - Receives, buffers, re-clocks, and rebroadcasts downstream HS packets received on its upstream port.
  - Performs same actions but in the opposite direction when a packet is received on one of its downstream ports.
- Transaction Translator (TT)
  - When a *Start Split* packet is received by a HS hub the TT evaluates the hub number field contained within the *start split* packet. If the hub number does not match the number of this hub, the transaction is ignored, otherwise the start split packet is accepted.
  - Upon accepting a HS *Start Split* Packet, the TT then decodes the port number contained within the *Start Split* packet to identify the hub port being targeted.
  - The TT initiates the low- or full-speed transaction to the target port. When the transaction completes either data or completion status will have been received and stored.
  - The TT subsequently receives a *Complete Split* transaction that requests return of the result of the LS- or Full-speed transaction.

## USB 3.0 Technology

- Hub Controller
  - In some cases the hub itself is the target of a transaction. Each Token packet received is evaluated by the hub controller to determine if it's being targeted.
  - When targeted, the hub completes the transaction as required.
  - Note that the repeater forwards HS packets across the hub whether or not the hub is the target.

Figure 2-18: Major High-Speed Hub Functions

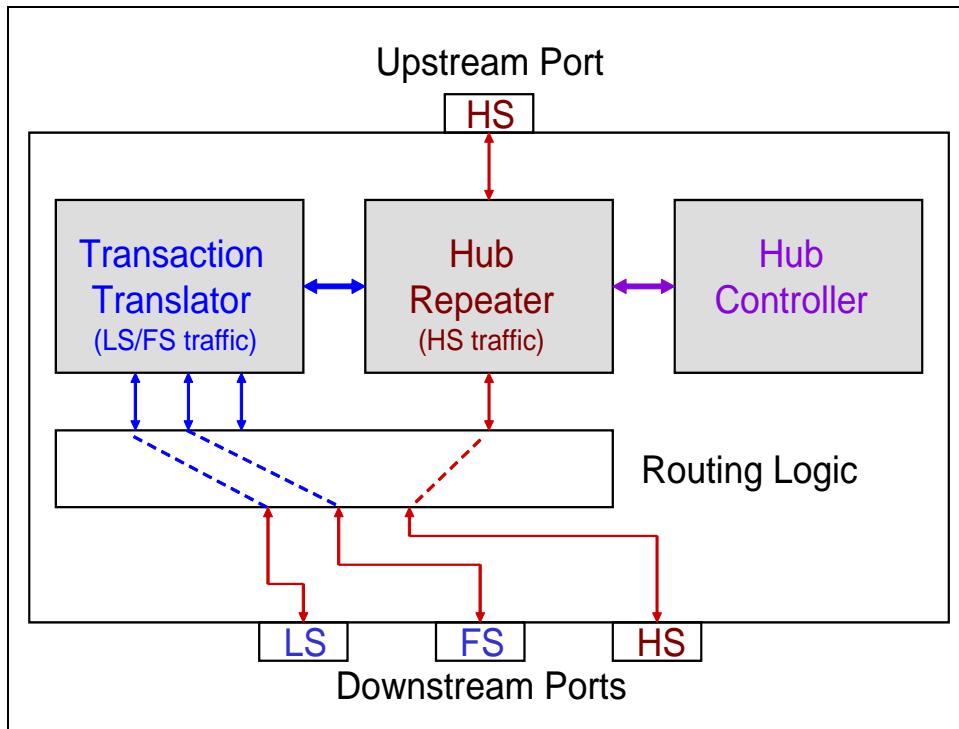


Figure 2-19 on page 41 illustrates the packet sequence associated with a Bulk IN split transaction where the downstream hub ports have low- and a full-speed devices attached. In this example a single transaction translator sends a packet to the full-speed device.

The sequence is as follows:

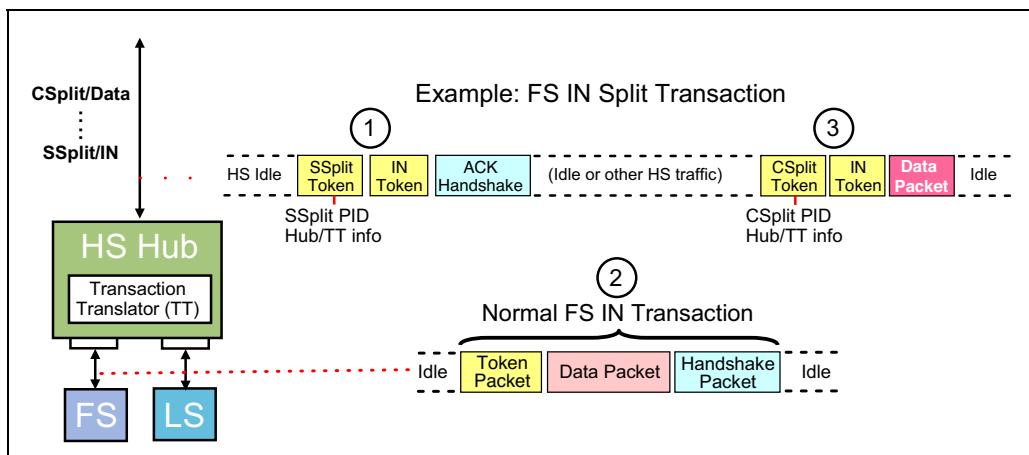
1. Every split transaction begins with the host sending the Start Split (SS) token packet followed by either an IN token packet or an OUT token fol-

## Chapter 2: USB 2.0 Background

lowed by a Data packet. Note that all token packets are send at high speed to the hub.

2. The Transaction Translator (TT) within the hub stores the token packets and data (if an OUT transaction) and starts the transaction at the native speed of the attached device -- full-speed in this example. The TT stores the Data payload returned from the device and awaits the Split Completion transaction from the host.
3. The host sends the Complete Split (CS) transaction consisting of the same token packets sent in the SS transaction to verify the entry being completed. In this example, the data payload is returned to the host via a Data packet, thereby completing the Split transaction.

Figure 2-19: Split Transaction Sequence - Bulk IN Transfer



Please note that the response to the Start Split and Complete Split token packets vary depending on the endpoint type and direction of the transfer.

### Transaction Generation and Scheduling

In the USB 2.0 environment, transaction generation and scheduling is managed by the host controller and its software driver. The driver receives transfer requests originating typically from application software and creates data structures called transfer descriptors (TDs). These TDs are linked together by the driver, thereby creating a schedule of transactions to be performed. This schedule describes the transaction to be performed by the host controller. The following sections detail these operations.

# **USB 3.0 Technology**

---

## **Transaction Generation**

USB transfers involve moving data between a device's endpoint buffers and buffers in memory supplied by the USB device's class driver in response to an application request. The USB host controller software driver receives the transfer requests and creates one or more transfer descriptors (TDs) that are needed to complete the requested data transfer. The essential contents of a transfer descriptor include:

- Target Endpoint — Device Address, Endpoint Number and Direction
- Endpoint Type
- Maximum Data Payload Size
- Transfer Speed (LS, FS, HS)
- Target Memory Buffer's Address Location
- Transaction Results Status (updated by host controller)
- Next TD Pointer

The transfer descriptor contains all of the information necessary to generate a USB transaction. Upon fetching and executing the TD, the host controller fetches the next TD in the linked list.

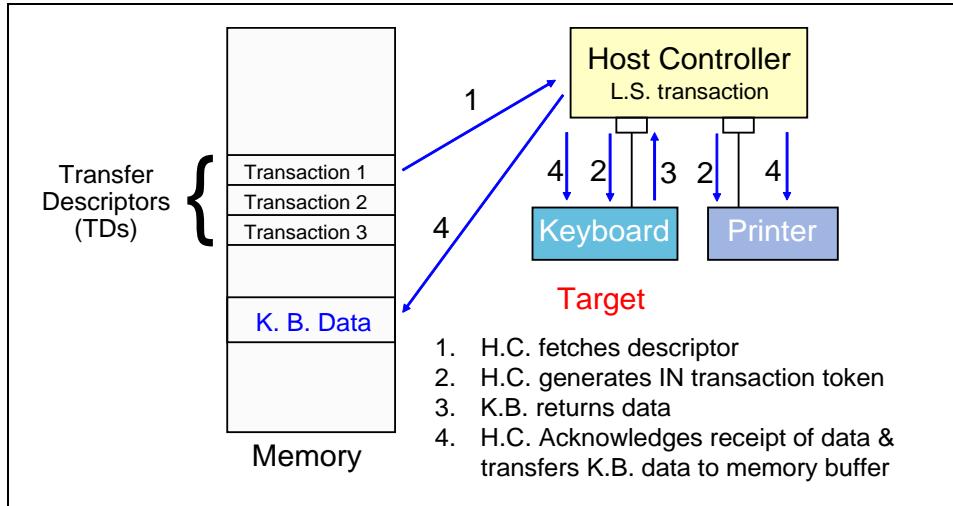
The following general steps are taken when a USB transaction is to be performed. This example involves polling a USB keyboard (See Figure 2-20 on page 43):

1. TD fetched from memory by host controller
2. IN transaction decoded from TD and IN Token is broadcast at low speed (preamble packet not shown)
3. Keyboard returns Data Payload
4. Host Controller performs two actions:
  - ACK broadcast at low speed (data received without error)
  - Data payload delivered to keyboard memory buffer

## Chapter 2: USB 2.0 Background

---

Figure 2-20: Transaction Generation



---

### Transaction Scheduling

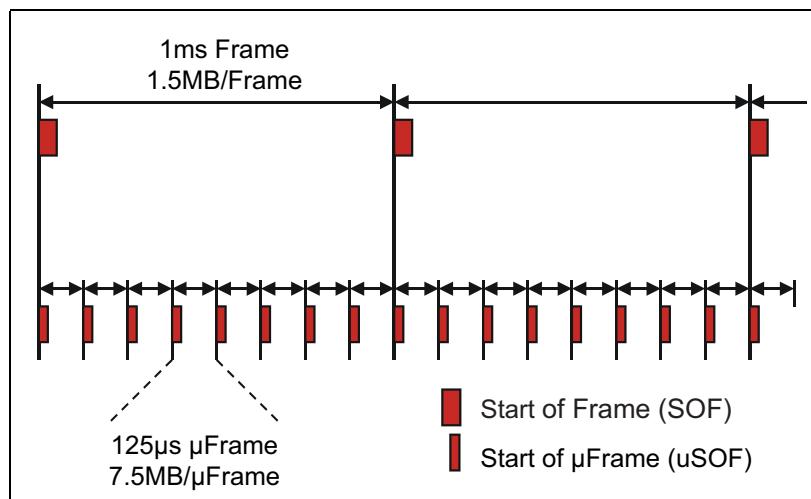
Transfers are scheduled on the basis of 1 ms Frames (LS and FS) and 125us MicroFrames (HS) and whether the transactions are periodic or non-periodic. Typically the host controller driver schedules the periodic transfers first, followed by control transfers and then bulk transfers. Figure 2-21 on page 44 illustrates the frames and the relationship between them. Note also the theoretical maximum bandwidth that can be achieved.

The host controller creates the frame structures and executes the lists of transfer descriptors. As an example, Figure 2-22 on page 44 illustrates the mechanisms used by the UHCI controller to create the 1ms frame lists and to fetch and execute the periodic and non-periodic transfers. The entire schedule of transfer descriptors is referred to as the frame list.

The host controller driver obtains memory space for setting up a frame pointer table in main memory that points to the linked list of transfer descriptors to be executed in each consecutive frame. The host controller's Frame List Base Address register specifies the location of the Frame list Pointer table in memory. The host controller also obtains memory space to set up the frame list. For each frame, the host controller schedules all periodic TDs first followed by the non-periodic TDs.

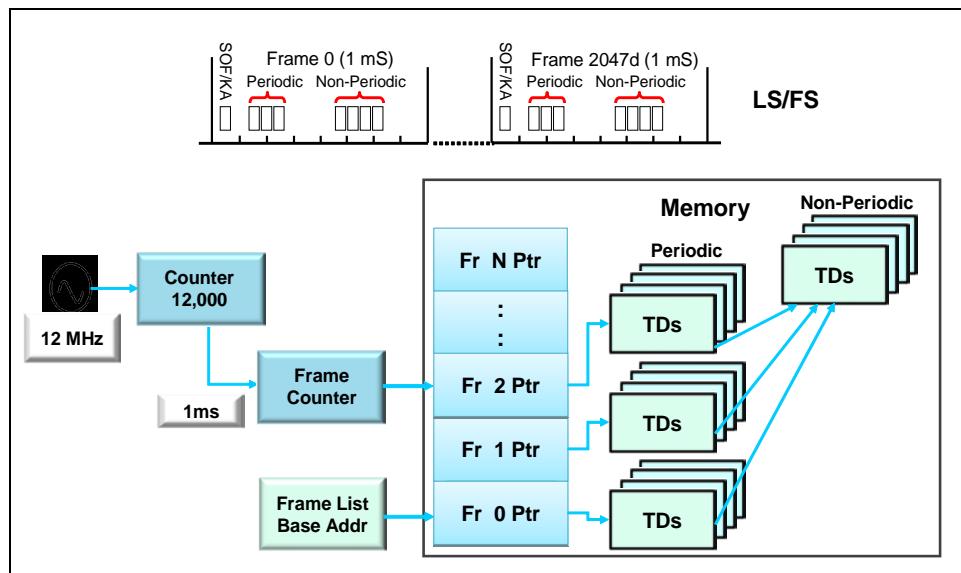
## USB 3.0 Technology

Figure 2-21: Frames and MicroFrames



The frame timing is managed by a 12 MHz clock that increments a counter that rolls over every 12,000 clocks (1 ms at 12 Mbits/s). Every 1ms the Frame Counter increments, thereby advancing the address offset to the next Frame Pointer entry.

Figure 2-22: Frame Lists Executed by UHCI Controller



## **Chapter 2: USB 2.0 Background**

---

### **Device Power**

All USB devices require a small amount of bus power ( $V_{bus}$ ) to permit device detection and removal. Devices may also be entirely powered via bus power. Bus power ranges from 100 ma (minimum) to 500 ma (maximum) at 5 volts. Devices that have their own power supplies are said to be self-powered.

## **USB 3.0 Technology**

---

---

# 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.

---

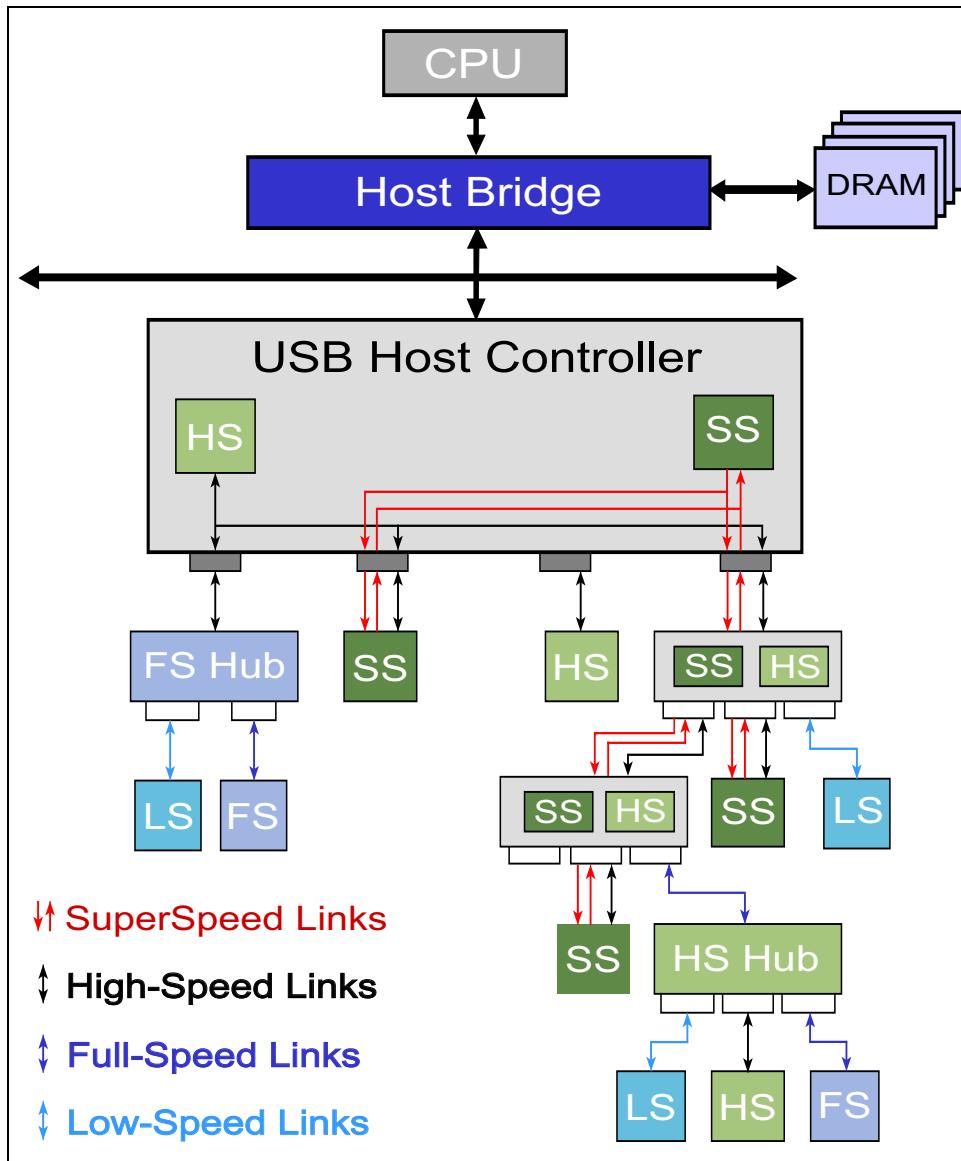
## **USB 3.0 Topology and Compatibility**

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.

## USB 3.0 Technology

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



# Chapter 3: USB 3.0 Overview

---

## USB 3.0 Host Controller

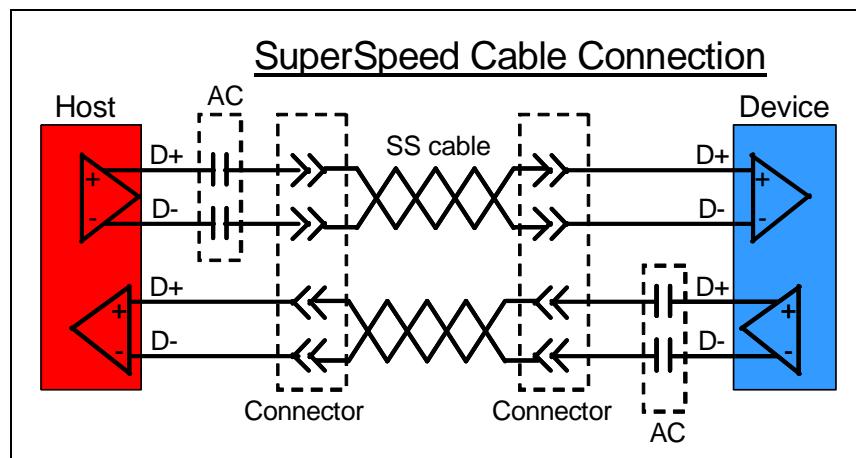
The USB 3.0 host controllers support low-speed (1.5 Mb/s), full-speed (12 Mb/s), high-speed (480 Mb/s) and SuperSpeed (5 Gb/s) operation, and do so without any companion controller requirements. This single controller can perform accesses to both USB 2.0 and USB 3.0 SuperSpeed devices simultaneously. 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



# **USB 3.0 Technology**

---

## **USB 2.0/3.0 Compatibility**

A major goal of USB 3.0 was to provide device compatibility in both directions. That is, all USB 2.0 devices are supported by the USB 3.0 implementation and all SS devices support connection to the USB 2.0 ports. Additionally, USB 3.0 preserves much of the existing investment in software, however, USB host software does require modification to support the SuperSpeed devices and USB 3.0 host controllers (such as xHCI). This section discusses the specific compatibility issues.

### **Receptacle and Plug Compatibility**

USB 3.0 host controllers and associated USB 3.0 connectors provide compatibility with the USB 2.0 implementation; however, in most cases USB 3.0 plugs are not compatible with USB 2.0 receptacles as listed in Table 3-1.

*Table 3-1: Plug and Receptacle Compatibility*

Receptacle	Compatible Plugs
<b>USB 3.0 Standard-A</b>	USB 3.0 Standard-A, USB 2.0 Standard-A
<b>USB 3.0 Standard-B</b>	USB 3.0 Standard-B, USB 2.0 Standard-B
<b>USB 3.0 Powered-B</b>	USB 3.0 Powered-B with Captive Cable (See “Powered B Connection Examples” on page 55 for details), USB 3.0 Standard-B, USB 2.0 Standard-B
<b>USB 3.0 Micro-B</b>	USB 3.0 Micro-B, USB 2.0 Micro-B
<b>USB 3.0 Micro-AB</b>	USB 3.0 Micro-A, USB 3.0 Micro-B USB 2.0 Micro-A, USB 2.0 Micro-B
<b>USB 2.0 Standard-A</b>	USB 2.0 Standard-A, USB 3.0 Standard-A
<b>USB 2.0 Standard-B</b>	USB 2.0 Standard-B
<b>USB 2.0 Micro-B</b>	USB 2.0 Micro-B
<b>USB 2.0 Micro-AB</b>	USB 2.0 Micro-A, USB 2.0 Micro-B

## **Chapter 3: USB 3.0 Overview**

---

### **SuperSpeed Device Compatibility**

SS devices have connections to the SS and USB 2.0 bus; however, simultaneous SuperSpeed and USB 2.0 operation is not allowed by any SuperSpeed devices, except USB 3.0 Hubs. Any USB SS functional device plugged into a FS or HS hub port will connect only on the 2.0 bus.

SuperSpeed devices must support at least one of the USB 2.0 speeds and report which speed or speeds are supported via the **SS Device Capability structure** located in the BOS descriptor (See page 506). Devices also report the lowest speed at which full functionality is provided.

### **Software Compatibility**

SuperSpeed system software inherits many of its requirements from USB 2.0, thereby preserving a sizeable investment in existing code. For example, Super-Speed host software maintains most of the USB 2.0 software infrastructure used for device enumeration and configuration.

Unlike the USB 2.0 host controllers, USB 3.0 host controllers (such as xHCI) perform transaction scheduling for all endpoint types and speeds:

- SuperSpeed
- High-speed
- Full-speed
- Low-speed

In these cases, the host controller schedules transactions in the same manner as USB host software by scheduling periodic transactions first followed by non-periodic transactions as bus time permits.

---

## **The SuperSpeed Physical Layer Environment**

This section discusses and illustrates some of the USB 3.0 cable assemblies and connector types.

---

### **USB 3.0 Composite Cable**

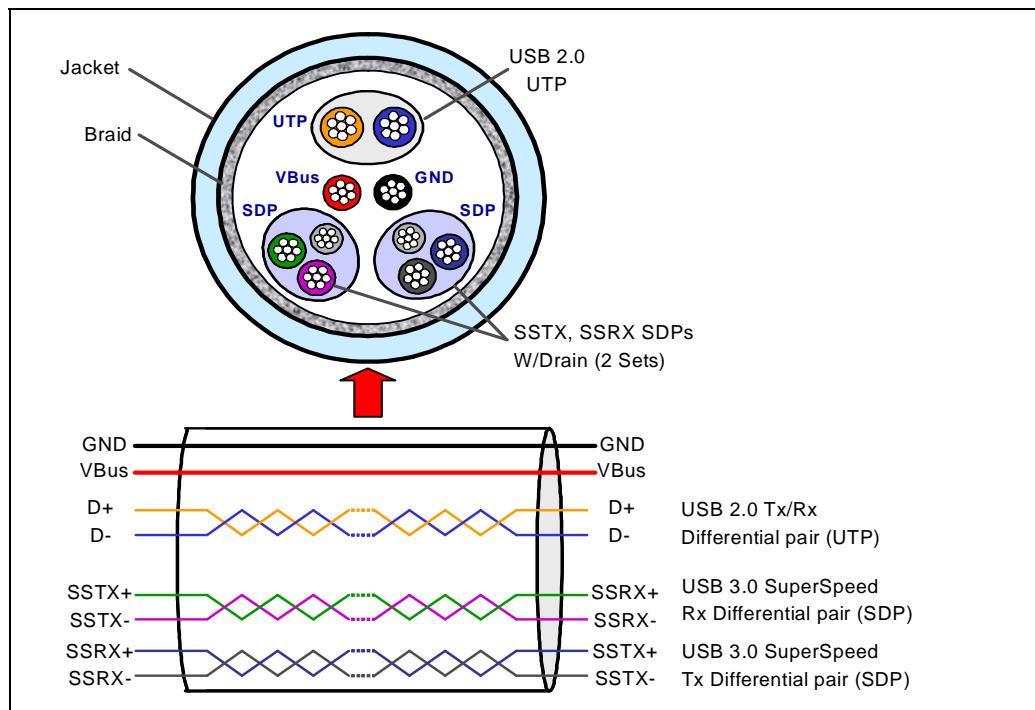
Figure 3-3 on page 52 presents a cross-section view and a representation of the signals implemented in a USB 3.0 cable. The cable interface consists of the following:

## USB 3.0 Technology

---

- USB 2.0 Un-shielded Twisted Differential Pair (UTP) — D+/D-
- SuperSpeed Differential Pairs (SDPs)— SSTX+/SSTX- and SSRX+/SSRX-
- Drain Wires for each SDP — Tied together at plug ends
- Bus Power — V<sub>Bus</sub> and Ground
- Braid — Tin plated copper or aluminum
- Outer Jacket — PVC or halogen free substitute

Figure 3-3: The USB 3.0 Cable Interface



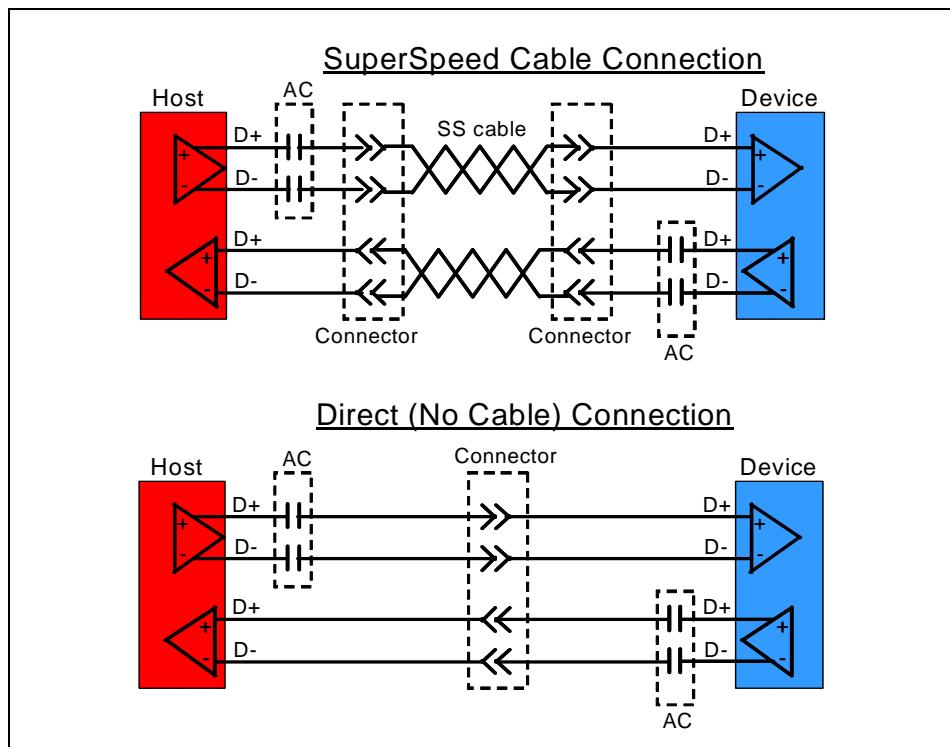
### SS Link Models

One of the benefits of the SS interface is that it can be used in a variety of signaling environments. The most challenging connection is when the two link partners are connected via an external cable as illustrated in Figure 3-4 on page 53. Note that the maximum cable length is not specified and depends on a variety of parameters. USB SS transmitters must use de-emphasis and the receivers must include equalization logic that adjusts to actual conditions. See “Super-Speed Transmitter Requirements” on page 596 and “SuperSpeed Receiver Requirements” on page 605 for more information on signal integrity and the electrical interface.

## Chapter 3: USB 3.0 Overview

The simplest implementation is when the SS link partners are directly connected on a board as illustrated in the bottom section of Figure 3-4. This type of very short connection may require lower transmitter power and may not require de-emphasis. The specification defines both low- and high-power transmitter requirements.

Figure 3-4: SS Signaling Environments



### Shared Bus Power

Remember that in the USB 2.0 implementation a given device can only draw 100 millamps (one unit load) of current when the device is first attached and not yet configured. SuperSpeed devices can draw up to 150 mA (one unit load) prior to being configured. SS also provides up to 6 unit loads or 900 mA maximum, whereas USB 2.0 defines 5 unit loads, or 500 mA.

# **USB 3.0 Technology**

---

## **USB 3.0 Compliant Cable Assemblies**

### **General**

A variety of USB 3.0 cable assemblies are defined by the specification to promote compatibility between the various plug types. Table 3-2 describes these cable assemblies.

*Table 3-2: USB 3.0 Cable Assemblies*

<b>Hub Receptacle Type</b>	<b>Device Plug Type</b>	<b>Comments</b>
Standard A	Standard B	Standard B for SS only
Standard A	Micro B	Micro B for SS only
Micro A	Micro B	Micro B for SS only
Micro A	Standard B	Standard B for SS only
Standard A	Captive Cable	Cable Permanently Attached
Micro A	Captive Cable	Cable Permanently Attached
Powered B	Captive Cable	See Next Section

## **USB 3.0 Power-B Connections**

Historically, USB bus power has only been supplied by root hub port and external hub ports. USB SuperSpeed has added the capability for a self-powered peripheral device to supply power in the upstream direction via a SuperSpeed Powered B receptacle. A Powered-B receptacle has two sets of power pins:

- V<sub>Bus</sub> and GND — this power comes from hub ports and provides power ranging from 150ma to 900ma at 5 volts or from an adapter that uses power from the Powered-B connector and then returns V<sub>Bus</sub> power (150ma) to the downstream peripheral device.
- DPWR and DGND — this power is sourced from a peripheral device that supports the Powered-B receptacle (1 Amp at 5 volts).

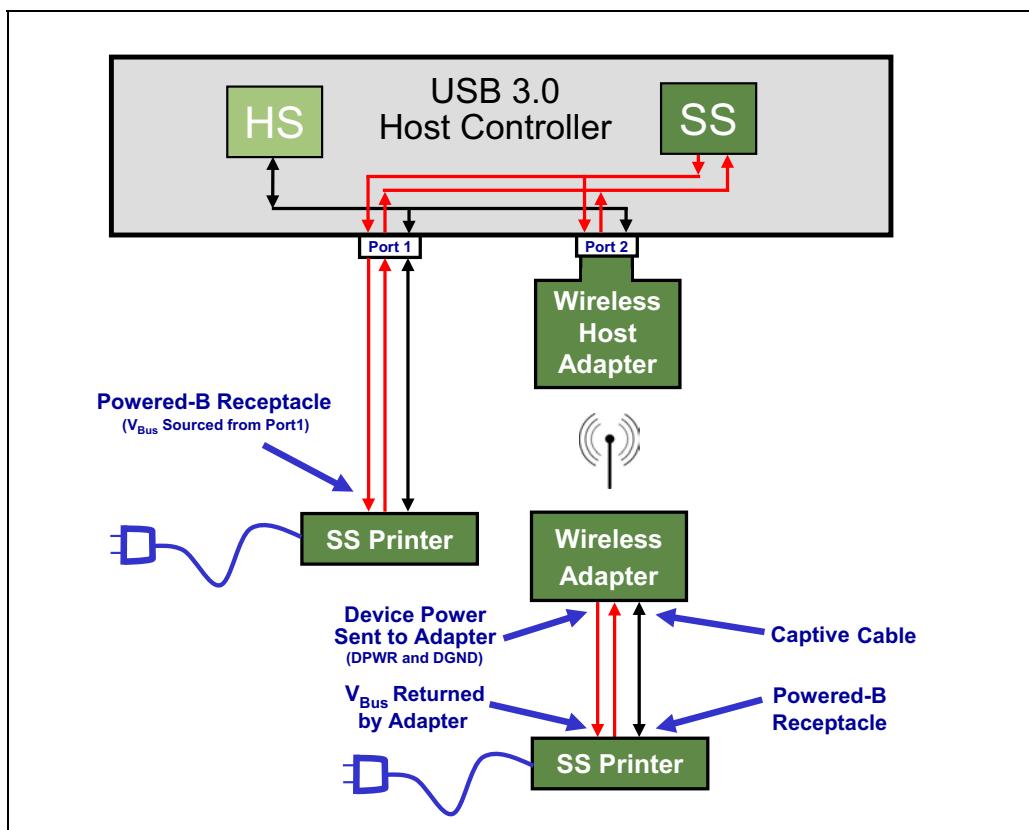
Peripherals that have a Powered-B receptacle have two power modes and methods of connection as illustrated in Figure 3-5 on page 55 and described in the

## Chapter 3: USB 3.0 Overview

example below. Any device that connects to the Powered-B receptacle must have a permanently attached, or captive cable.

1. Port 1 in Figure 3-5 depicts an attached peripheral device (SS Printer) having a Powered-B receptacle. In this configuration the printer is attached in standard fashion with  $V_{Bus}$  power being supplied by root port 1. The cable assembly connecting the peripheral is a USB 3.0 Standard A to Standard B connector that does not support the device power pins (DPWR and DGND). So, device power is available but not connected.
2. Port 2 depicts a printer connected to the host via a wireless connection. The wireless adapter supports the printer's Powered-B receptacle, thus the adapter receives power from the printer via the DPWR and DGND pins. This eliminates the need for a power supply associated with the wireless adapter. The wireless adapter is also required to return  $V_{Bus}$  power to the printer to permit device detection and have a permanently connected cable.

Figure 3-5: Powered B Connection Examples

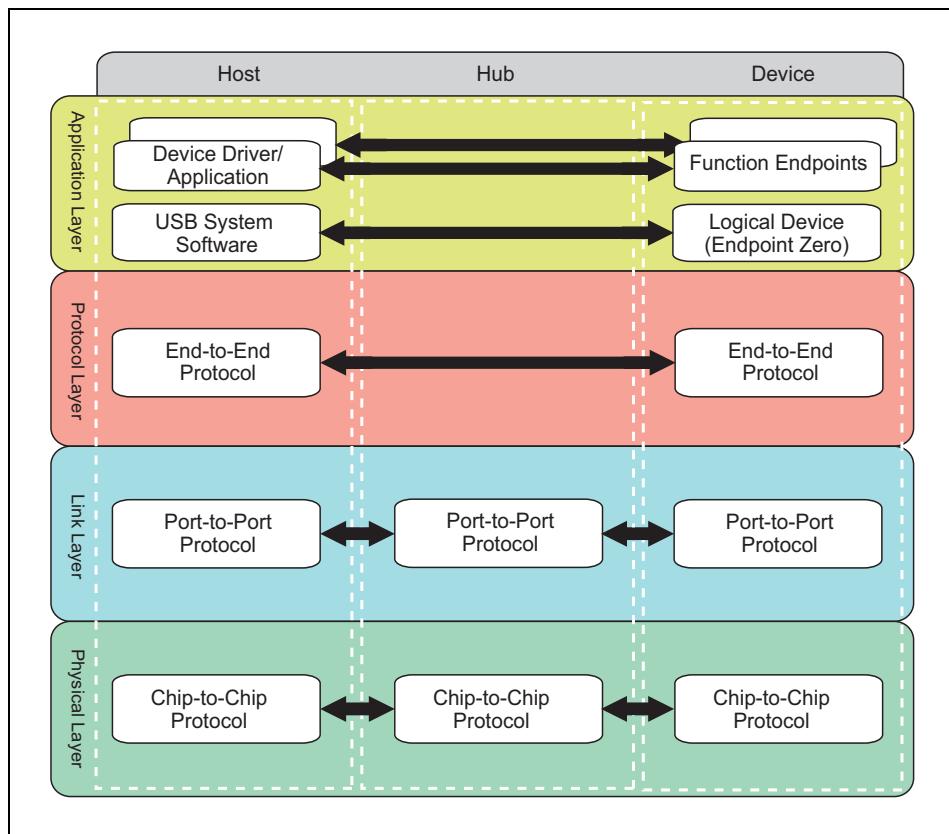


# USB 3.0 Technology

## **SuperSpeed Layered Interface and Protocols**

SS USB defines three interface layers that are discussed in this section. In conjunction with these layers are associated protocols that assist in delivering packets between USB host software and a USB device. The interface layers and protocols are illustrated in Figure 3-6. The diagram shows the host controller, a hub and device and the layers that each use during SuperSpeed communication. Connecting arrows between the host and device represent logical communications between each of the layers. The arrows represented in the Chip-to-Chip protocol represents actual physical communication across the SuperSpeed link.

*Figure 3-6: SS Interface Layers and Protocols*



## **Chapter 3: USB 3.0 Overview**

---

The layers and their primary responsibilities include:

- **Application Layer** — Software applications initiate requests to move data between a memory buffer that software allocates and the target endpoint buffer within the USB device. This is the same action taken by all USB implementations. The logical data flow (represented by the arrows) is between the application/device driver buffers in host memory and the function endpoint buffers within a USB device. USB-specific system software also provides buffers for accessing the device descriptors via endpoint zero. These descriptors give software a logical view of device characteristics including the nature of the function (e.g., printer, keyboard, hard drive, etc.)
- **Protocol Layer** — Transfer requests made by application software result in transactions being sent by the host controller to the protocol layer of the SS interface. Logical data flow is between the transmit buffer of the root port and the endpoint buffer within the target device. (See Figure 3-6.) The protocol layer creates a compliant header packet and manages the End-to-End protocol that is based on the USB 2.0 Token/Data/Handshake protocol. The section entitled “End-to-End Protocol (Protocol Layer)” on page 61 describes the differences between the token/data/handshake sequence and the end-to-end protocol.
- **Link Layer** — The link layer primarily serves to improve data integrity, transfer efficiency, and link power savings. A variety of Link-layer packets are exchanged to support these functions:
  - **Framing** adds a sequence of special characters (called ordered sets) to the front of all headers and link-layer packets to ensure the link partner can accurately detect the start of each packet. Note that Data Payload packets have both start and end framing.
  - **Flow Control** ensures that the transmitter never sends a header to the link partner that will be rejected because of insufficient buffer space. The flow control mechanism exchanges data link packets called flow control credits to update the transmitter regarding the amount of buffer space available.
  - **Error Detection and Recovery** verifies whether headers sent to the link partner arrive without error. The link layer generates and checks CRC values of all headers sent between link partners. Every header is stored in a transmit buffer within the link and when the CRCs are checked by the receiver it returns either a Good or Bad indication to the transmitter. Thus a header is either removed from the buffer (Good), retried (Bad) or the link is retrained in the event of a serious error.
  - **Link Power Management** permits a link to enter and exit low power states to preserve power when the link is not in use. Link Layer packets are exchanged upon entry to the low-power state.

# **USB 3.0 Technology**

---

- **Physical Layer** — Headers, link-layer packets, and ordered sets must be prepared for transmission across the SuperSpeed link. This process involves:
  - scrambling all bytes, except ordered sets, to reduce EMI
  - encoding each byte resulting in a 10-bit symbol
  - sending the data serially at a transmission rate of 5 Gb/sec.

The inverse operations are performed at the receive side.

---

## **Hub Interface Layers**

External hubs use the interface layers differently than root hub ports and peripheral devices, when forwarding packets. This section highlights how the port interface layers are used by the hub under three different circumstances.

### **Layers - Hub Forwarding Packets**

Figure 3-7 illustrates a hub with three downstream-facing ports. In this example the hub is forwarding a header packet from the host to a specific downstream port based on the SS unicast mechanism. When forwarding packets the hub ignores the protocol layer and passes the packet between the link layers.

### **Layers - Hub as Target**

Figure 3-8 illustrates an end-to-end protocol event where the hub is targeted by host software. In this case, the protocol layer of the upstream-facing port is involved in managing the protocol. In this example, the hub controller is used to access one of the hub ports. For example, a Get Hub Port Status request that returns a data packet with two bytes of Status and two bytes of Status Change information.

## Chapter 3: USB 3.0 Overview

Figure 3-7: Hub Bypasses the Protocol Layers When Forwarding Packets

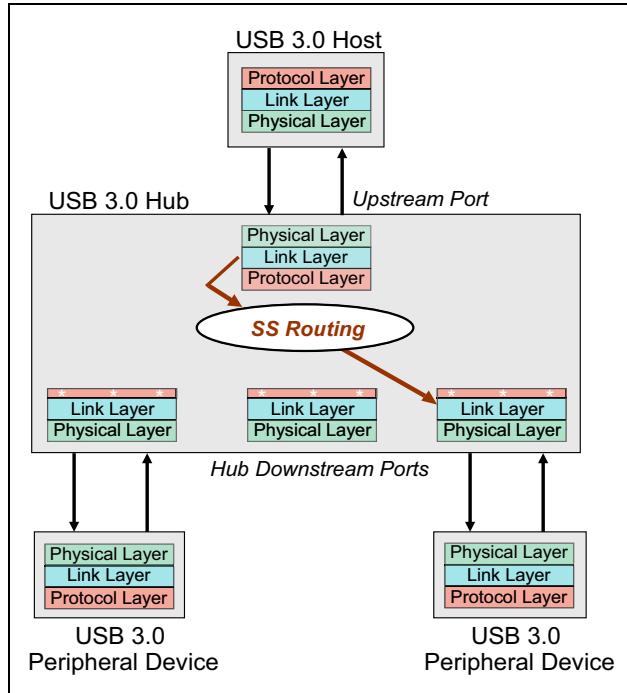
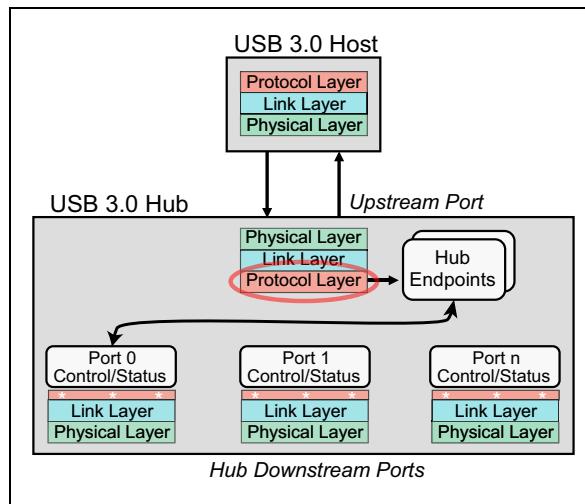


Figure 3-8: Hub as Target Device



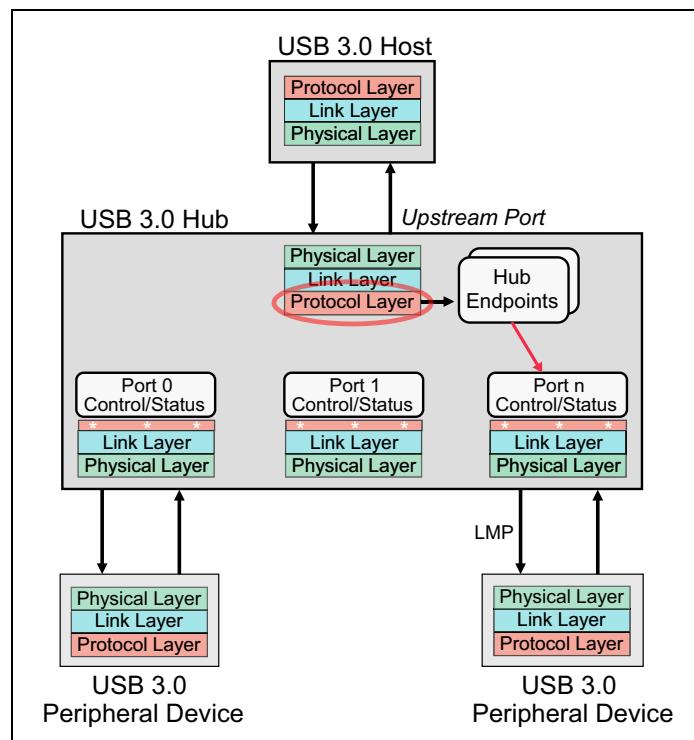
# USB 3.0 Technology

## Layers - Hub Downstream Port (Thin Protocol Layer)

The downstream-facing ports of a hub are accessed via the hub controller and are never accessed directly by any end-to-end transactions. The thin protocol layer shown in the hub diagrams support the exchange of Link Management Packets (LMPs) between the link partners. See Figure 3-9 on page 60. These LMP header packets are categorized as protocol layer packets and use only a small portion of the protocol layer. LMPs are associated with a variety of functions and are discussed later in the context of:

- Link Configuration
- Link Power Management
- Test Features

*Figure 3-9: Hub as Target Device with LMP Delivery*



# Chapter 3: USB 3.0 Overview

## Three Protocol Layers

The three layers of the SuperSpeed port interface each have protocols associated with them. These associated layers and protocols are:

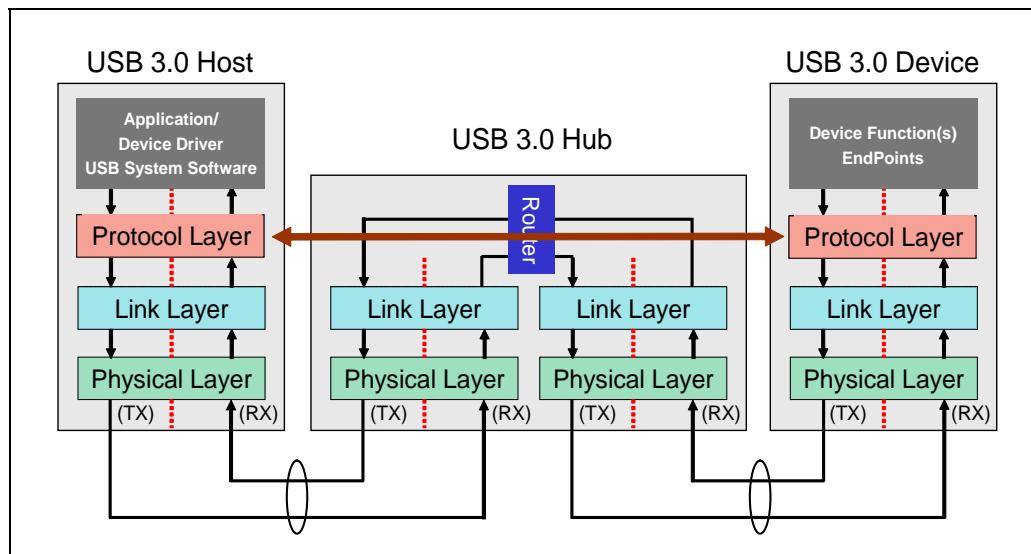
1. Protocol Layer and End-to-End protocol
2. Link Layer and Port-to-Port protocol
3. Physical Layer and Chip-to-Chip protocol

Each of the protocols are introduced in the three following sections.

### End-to-End Protocol (Protocol Layer)

The SuperSpeed bus employs a version of the Token/Data/Handshake sequence called the End-to-End protocol. Figure 3-10 illustrates the logical flow of the high-level Protocol Layer packets. Header packets are routed to the target device using a unicast delivery mechanism. The target device returns DATA in the event of an IN transaction and Acknowledges receipt of DATA during an OUT transaction. The following sections describe the differences between the Token/Data/Handshake sequence and the IN and OUT sequences used by the End-To-End protocol.

Figure 3-10: Example of End-to-End Communication



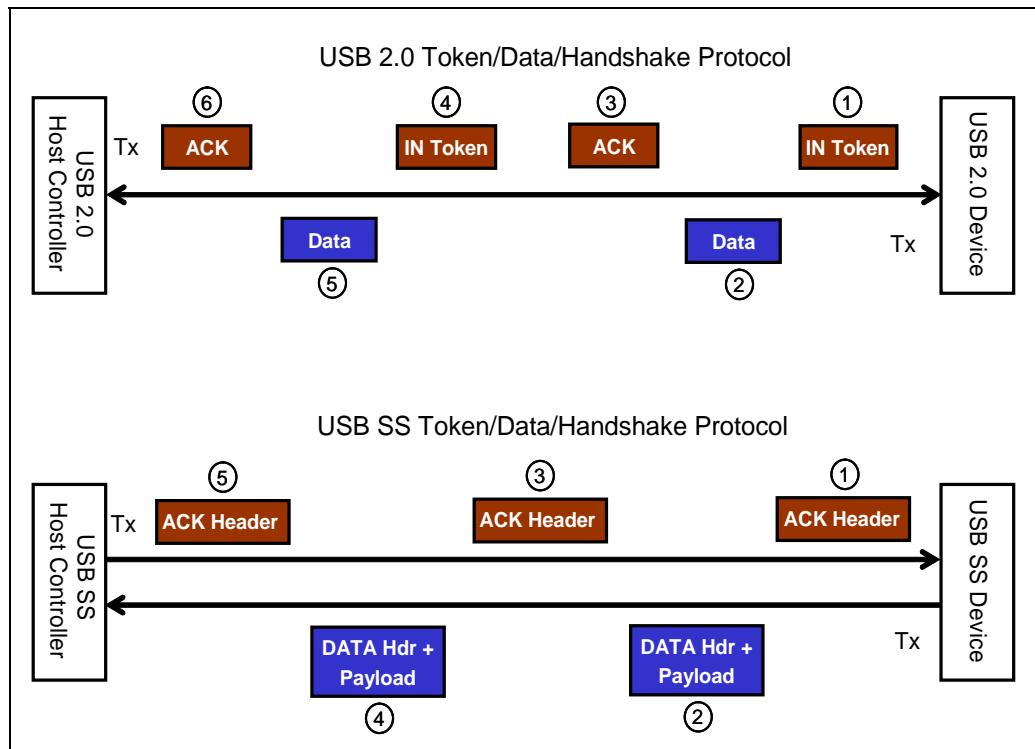
## USB 3.0 Technology

**USB 2.0 vs SS IN Transaction Comparison.** Figure 3-11 on page 62 depicts two typical back-to-back IN transaction sequences. At the top of the diagram is the USB 2.0 Token/Data/Handshake sequence. The operation is straight forward as listed below:

1. Host delivers an IN token packet
2. Device returns a DATA packet
3. Host acknowledges (ACK) successful receipt of the DATA

This same sequence is completed a second time resulting in six packets being exchanged.

Figure 3-11: IN Transaction — Token/Data/Handshake versus End-to-End Protocols



## **Chapter 3: USB 3.0 Overview**

---

The End-to-End sequence at the bottom of Figure 3-11 is performed as follows:

1. Host initiates an IN transaction request by delivering an ACK header packet that incorporates the token-related information (i.e., Device Address, Endpoint Number, and Direction of the transaction)
2. Device returns a DATA header and Data Packet Payload packet to the host
3. Host returns ACK header packet that performs two operations:
  - Acknowledges successful receipt of the DATA packet
  - Delivers a second IN token request
4. Device returns a DATA header and Data Packet Payload
5. Host send ACK, thereby acknowledging successful receipt of data

The two IN transactions require five packets to complete across the dual simplex SS bus.

**USB 2.0 vs SS OUT Transaction Comparison.** Figure 3-12 on page 64 depicts two typical back-to-back OUT transaction sequences for USB 2.0 Token/Data/Handshake (top diagram) and USB SS Token/Data/Handshake (bottom diagram). The USB 2.0 Token/Data/Handshake sequence (top diagram) is straight forward as listed below:

1. Host delivers an OUT token packet
2. Host sends a DATA packet
3. Device acknowledges (ACKs) successful receipt of the data

This same sequence is completed a second time resulting in six packets being exchanged.

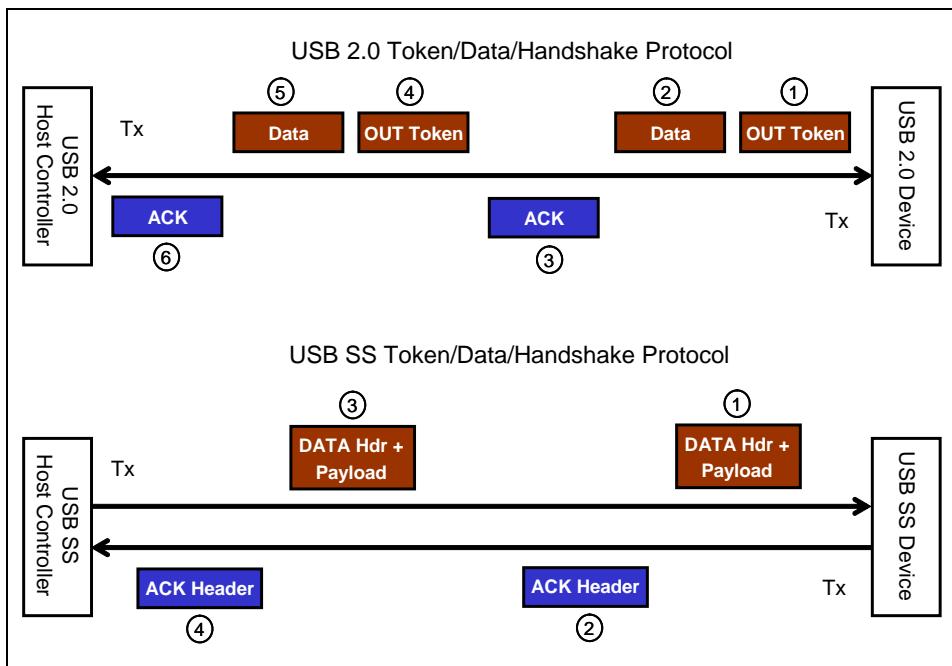
The End-to-End sequence is performed as follows:

1. Host sends a DATA header and Data Packet Payload performing two operations:
  - DATA header includes OUT token packet
  - Data Packet Payload carries the Data
2. Device returns ACK header, thereby acknowledging receipt of data
3. Host sends DATA header and Data Packet Payload again
4. Device returns ACK header to complete the two transactions

The two OUT transactions require four packets to complete across the dual simplex SS bus.

# USB 3.0 Technology

Figure 3-12: OUT Transaction — Token/Data/Handshake versus End-to-End Protocols



## Port-to-Port Protocol (Link Layer)

Every protocol layer packet (called Headers) that traverses a link is subject to the Port-to-Port protocols. The two primary Port-to-Port protocols are link **Flow Control** and **Error Detection and Correction** which are performed at the Link layer. Figure 3-13 illustrates a Header originating at the USB 3.0 Host that traverses a Hub and ends at the USB 3.0 Device. Consequently, two Port-to-Port connections exist between the link partners. The general steps taken between the Host and Device are described below:

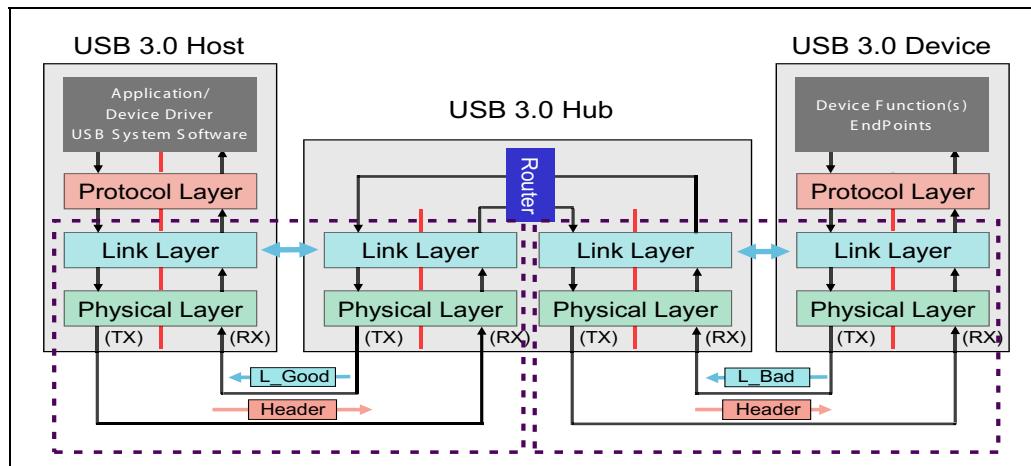
- The Protocol layer prepares the header for delivery to the Link Layer. Upon receipt of the Header, the Link Layer takes the following actions:
  - Flow Control credits have been exchanged between the link partners during initialization, so each knows the number of Header buffers that are available. In this example, the Link layer checks credits and recognizes the Hub has enough buffer space to send the Header.
  - The Link layer also adds a sequence number to the header and generates a CRC value so the link partner can verify whether the Header is

## Chapter 3: USB 3.0 Overview

successfully transferred to the Hub. (Note: The host's Link layer also stores a copy of the Header in its transmission buffer so that an error detected at the Hub's Link layer can be reported back and retried by the Host.)

- When the Header reaches the Link layer at the Hub's ingress port, it checks the CRC value and sequence number, thus verifying the Header has been successfully transferred.
- In response, the Hub's Link layer originates a Link layer packet, called L\_Good (Link Good) and sends it to the Host's Link layer to confirm successful Header delivery.
- The Hub having received the Header packet successfully, forwards the Header to the egress port of the Hub and the final Port-to-Port transfer is performed.
  - As described in the previous example, Flow Control credits are checked by the Hub's Link layer and as before credits are available.
  - The Link layer assigns a sequence number and CRC value to the Header. Then the Header is sent to the link partner to verify successful transmission. (Note: A copy of the Header is stored in the transmission buffer of the Hub's egress port in the event of a failed transmission.)
  - The Header is received at the Link layer of the USB 3.0 Device and a CRC error is detected. The Header is discarded and a Link layer packet, called L\_Bad is sent back to the Hub.
  - Upon detecting the L\_Bad the stored Header is sent to the Device again and this time the transmission is successful.

Figure 3-13: Port-to-Port Protocol Between Link Partners



# **USB 3.0 Technology**

---

## **Chip-to-Chip Protocol (Physical Layer)**

The physical layer prepares every byte for transmission across the link requiring multiple stages. When the link is in a non-communicating state (electrical idle) Low Frequency Periodic Signalling (LFPS) initiates the transition back to the communicating state.

All of these processes define the Chip-to-Chip protocol. The following list introduces the primary elements associated with the Chip-to-Chip protocol. The description focuses on preparing each byte for transmission, and note that the receiving device performs the inverse operations.

- Scrambling — Each byte to be transferred across the link is scrambled to randomize the bit patterns during transmission to reduce Electro-Magnetic Interference (EMI). This is true of all bytes except those that are part of an ordered set.
- 8bit/10bit Encoding — Every byte to be transmitted across the link must be converted into a 10-bit value called a symbol. The symbols fulfill the following beneficial functions:
  - encodes the 5Gb/s clock into the bit stream
  - provide an encoding scheme that permits error detection at the receiver
  - provides unique **control symbols** that are recognized by the receiver and that create ordered sets
- A special ordered set called SKIP
- Parallel to Serial Conversion — All symbols are converted to a serial bit stream and transmitted across the link

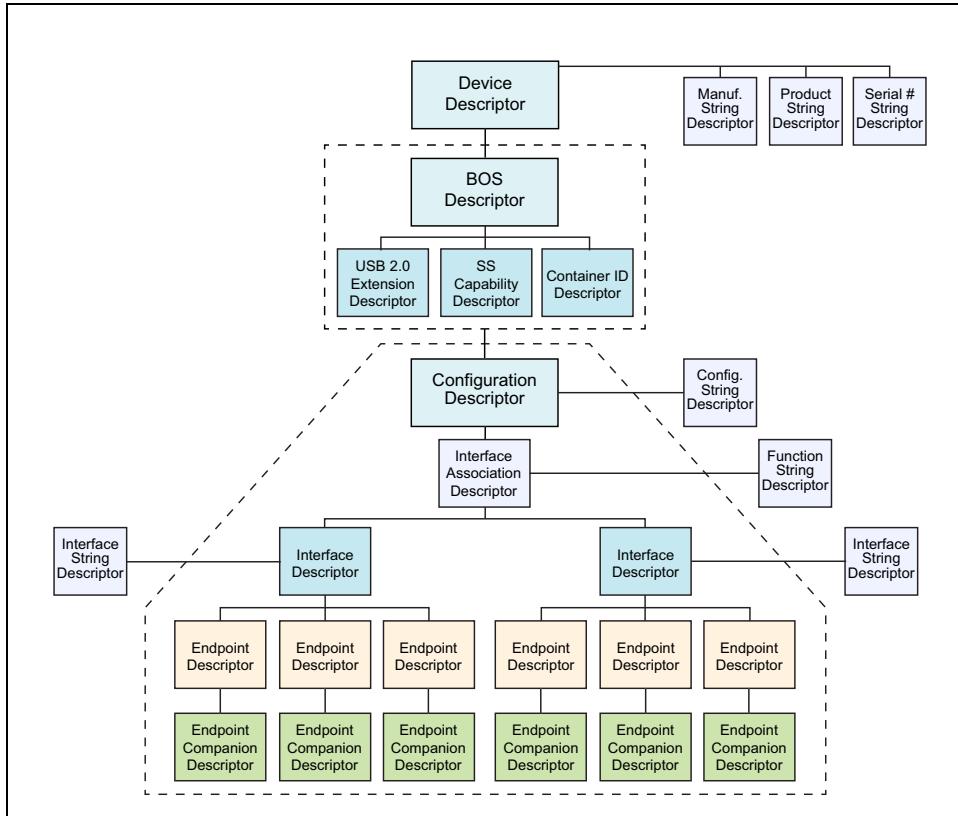
---

## **Configuration and Descriptors**

SuperSpeed configuration is very similar to the USB 2.0 configuration process. The primary differences are in the new features introduced by the SS implementation, which are defined in the device descriptors. Figure 3-14 illustrates the SS descriptors that include the Endpoint Companion and BOS (Binary Object Store) descriptors that were added in the USB 3.0 specification.

## Chapter 3: USB 3.0 Overview

Figure 3-14: SuperSpeed Descriptors



## Power Management

The USB 2.0 power management features are fairly simplistic and have been the source of considerable concern over the years. A major goal of the USB 3.0 specification was to improve power management in a variety of ways. The Super-Speed USB implementation includes 3 primary features for improving overall power management:

- Link Power Management
- Function Power Management
- Suspend and Resume

# USB 3.0 Technology

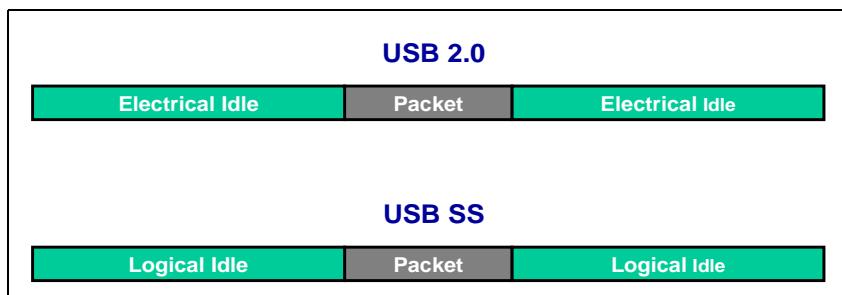
In addition, features of the End-to-End protocol and host controller designs such as xHCI also play an important role in reducing overall system power management by reducing USB bus traffic, along with USB-related traffic to and from host memory.

---

## **Link Power Management**

Prior to introducing the SS power management features let's compare the SS and USB 2.0 buses. Figure 3-15 shows that the USB 2.0 bus is in the electrical idle state until a packet is transmitted. A synchronization sequence at the front end of each USB 2.0 packet allows the receiving device to synchronize to the incoming packet. In contrast, the USB SS bus is initialized for communication immediately following reset and continually transfers information. Even when no packet is being sent, the SS bus continues to transmit (called logical idle) so that it is always ready when a packet is to be sent. Consequently, SS power management must aggressively manage each link by putting it into the electrical idle state as often as possible.

*Figure 3-15: USB 2.0 vs USB SS Bus Idle*



SuperSpeed link power management includes four states as shown below:

- U0 — USB link power state zero means that both link partners are fully powered.
- U1 — USB link power state one means the link is in a standby mode. Standby indicates electrical idle but with fast recovery back to U 0.
- U2 — Link power state two is also standby but conserves considerably more power than U1, and consequently requires more time to recover back to U 0.
- U3 — When the link transitions to U3 even more power is saved and the device enters suspend. Recovery in this particular case is very slow when compared to the standby states.

## **Chapter 3: USB 3.0 Overview**

---

These mechanisms are detailed in chapter entitled, “SuperSpeed Power Management on page 563.”

---

### **Function Power Management**

Power management software may elect to put a single function into a suspended state without impacting other functions within the same device. This is accomplished via a new SS request called Set Function Suspend.

## **USB 3.0 Technology**

---

---

# 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.

# **USB 3.0 Technology**

---

## **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
<b>Link Management Packets</b>	00000b	6
<b>Transaction Packets</b>	00100b	8
<b>Data Packet</b>	01000b	NA
<b>Isochronous Timestamp Packet</b>	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
- Port Configuration following certain resets
- Vendor Device Testing

---

## **The Token Lives On**

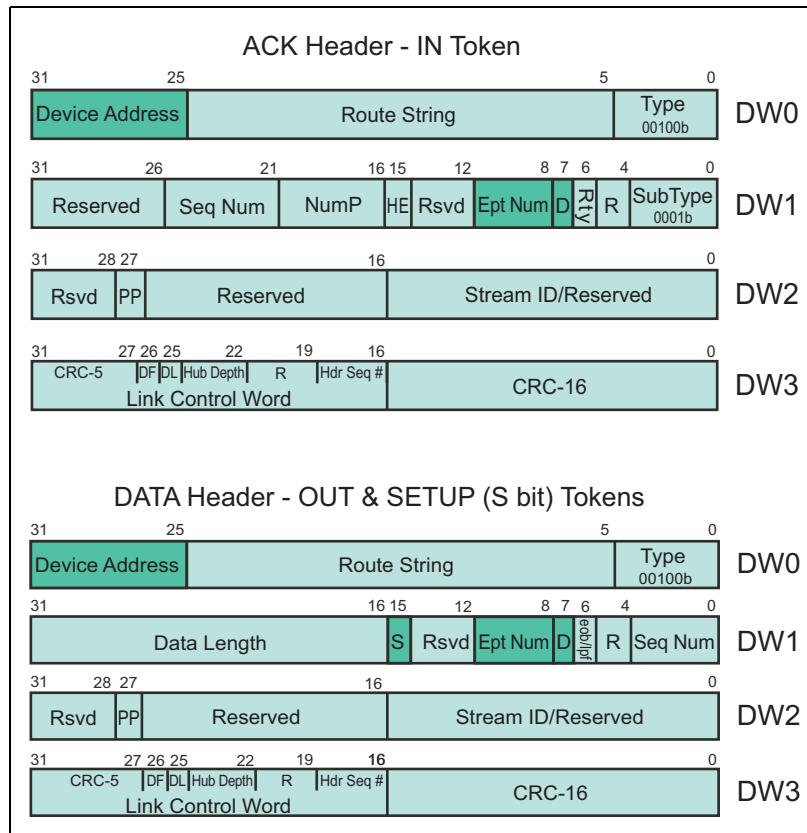
Every transaction initiated on the USB 2.0 begins with a token packet whose content is still used by the SS protocol. The specification calls this information the Address Triplet, which uniquely identifies each device endpoint in the topology. The Address Triplet includes:

- 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)

## Chapter 4: Introduction to End-to-End Protocol

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



## Data 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

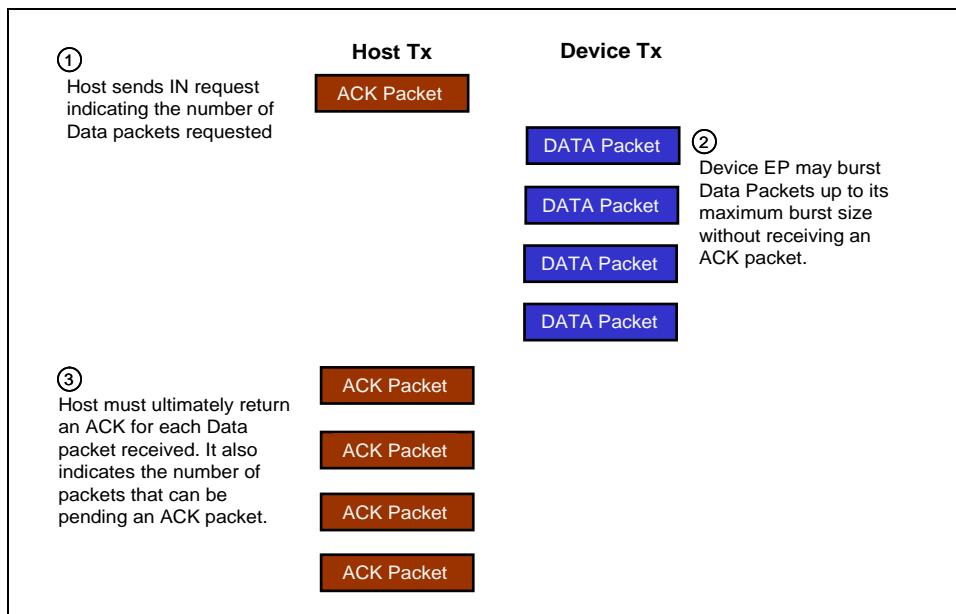
## USB 3.0 Technology

---

previous DATA packet. The maximum number of packets that can be burst is limited by the specification to 16 packets. A device endpoint reports the maximum burst size it supports—2, 4, 8, or 16 DATA packets.

Figure 4-2 illustrates an IN transaction with a target device that supports a maximum payload size of four DATA packets. Notice that prior to receiving the first acknowledgement up to four packets could have been sent. During OUT transactions the Host can also burst DATA to the target device and the same Burst size limitations still apply. At no time can a device or the host have more unacknowledged packets than specified in the Maximum Burst size designation.

Figure 4-2: IN Burst Example with Maximum Payload Size of Four



---

## Bulk Endpoint Streaming

Prior to the USB 3.0 specification a bulk endpoint provided access to a single buffer associated with a *device address, endpoint number and direction* (i.e., an endpoint address). The bulk endpoint streaming feature expands the number of buffers that can be accessed with a single endpoint address. The expansion involves adding a 16-bit StreamID to the endpoint address. This allows nearly 64K individual buffers that can be selected behind a single endpoint.

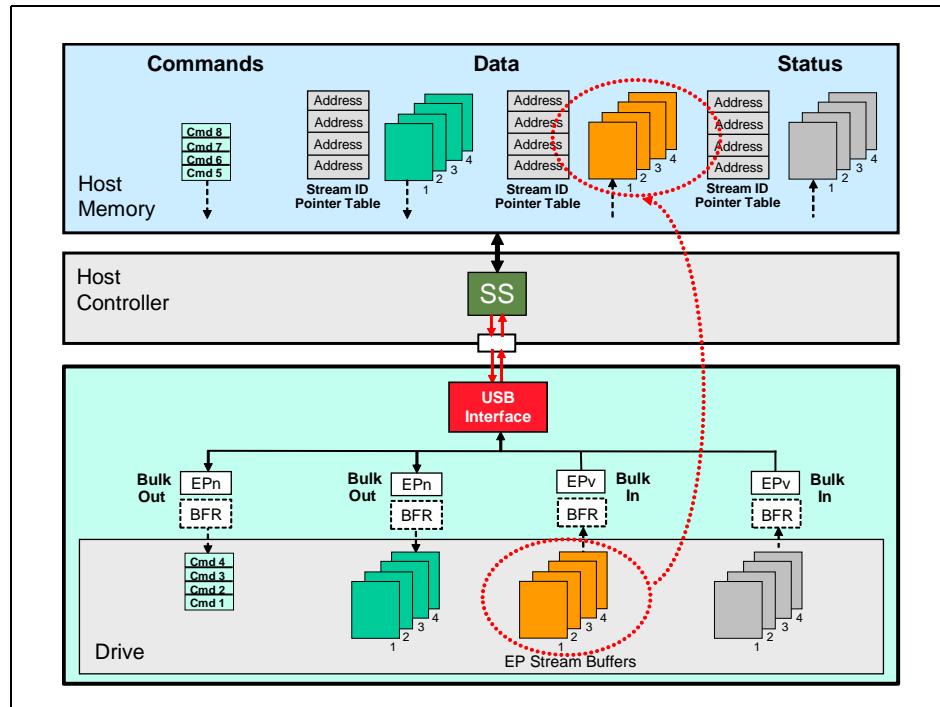
## Chapter 4: Introduction to End-to-End Protocol

Perhaps the next question is what types of applications can take advantage of this feature. One of the most obvious applications of the streaming feature are hard drives. A new USB Attached SCSI (UAS) specification and the related USB-IF protocol called UASP permits a full-SCSI implementation that allows command queuing and out of order execution, thereby reducing overall drive access latency. This SCSI solution can use the bulk streaming solution to manage multi-direction commands and related endpoint buffers. Figure 4-3 illustrates an example UASP implementation where four bulk endpoints are employed as follows:

- One standard Bulk OUT endpoint for queuing commands
- One Bulk OUT Streaming endpoint for transferring data to the drive
- One Bulk IN Streaming endpoint for transferring data from the drive
- One Bulk IN Streaming endpoint for reporting status to software

The stream buffers are selected via a 16-bit Stream ID value that is associated with a particular command and with buffers in memory. This structure allow commands to be processed out-of-order thereby improving overall performance. See “Bulk Streaming Endpoints & UAS Drives” on page 154 for complete details.

Figure 4-3: Example UASP Structure



# **USB 3.0 Technology**

---

## **Endpoint Characteristics for SuperSpeed**

Table 4-2 on page 76 summarizes the endpoint types and their primary characteristics.

*Table 4-2: General Endpoint Characteristics*

EndPoint	Scheduling	Maximum Data Payload	Burst Capable	Max Burst Size	Streaming Capable
<b>Bulk</b>	Non-periodic	1024 bytes	Yes	16 KB	Yes
<b>Interrupt</b>	Periodic	1024 bytes	Yes	3 KB	No
<b>Isochronous</b>	Periodic	1024 bytes	Yes	48 KB	No
<b>Control</b>	Non-periodic	512 bytes	No	NA	No

---

## **Port-to-Port Protocol Influence**

Every header packet is subject to the Port-to-Port protocols. The protocol having the greatest impact on the End-to-End protocol is the error detection and recovery mechanism. This feature ensures that every header packet sent between link partners will be checked for errors and in the event of an error the packet will be retried until delivered successfully. In fact the specification mentions that the Protocol layer assumes all header packets are always transferred successfully under control of the Port-to-Port protocol, which is managed by the link layer.

Even with the Port-to-Port protocol virtually guaranteeing that all headers will be successfully transferred end to end, failures are of course possible. The specification notes that if the host does not receive a response to a transaction request (IN or OUT) within 10µs, it must assume the transaction has failed and halt endpoint processing. In this case software is responsible for error recovery.

Because the Port-to-Port protocol applies to headers only, a DATA header may be successfully retried as a result of errors, but the attached DATA Packet Payload (DPP) is not subject to the Port-to-Port protocols. Instead, the End-to-End Protocol layer takes the responsibility of retrying DATA Packets that have failed.

## **Chapter 4: Introduction to End-to-End Protocol**

---

---

### **Does SuperSpeed Support Parallel Operations?**

This question can be answered in a few ways. The specification clearly indicates the possibility of some parallelism with the following statement:

*"A SuperSpeed host may initiate one or more OUT bus transactions to one or more endpoints while it waits for the completion of the current bus transaction. However a SuperSpeed host shall not initiate another IN bus transaction to any endpoints until the host [verifies the specified requirements]."*

## **USB 3.0 Technology**

---

---

# 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.

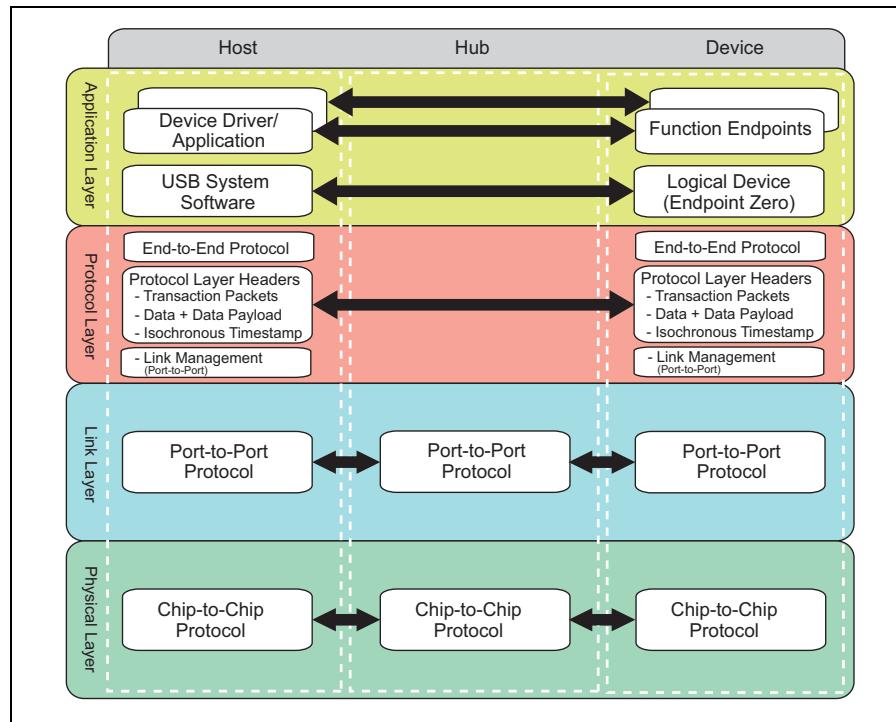
# USB 3.0 Technology

---

- 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



## Chapter 5: End-to-End Packets

---

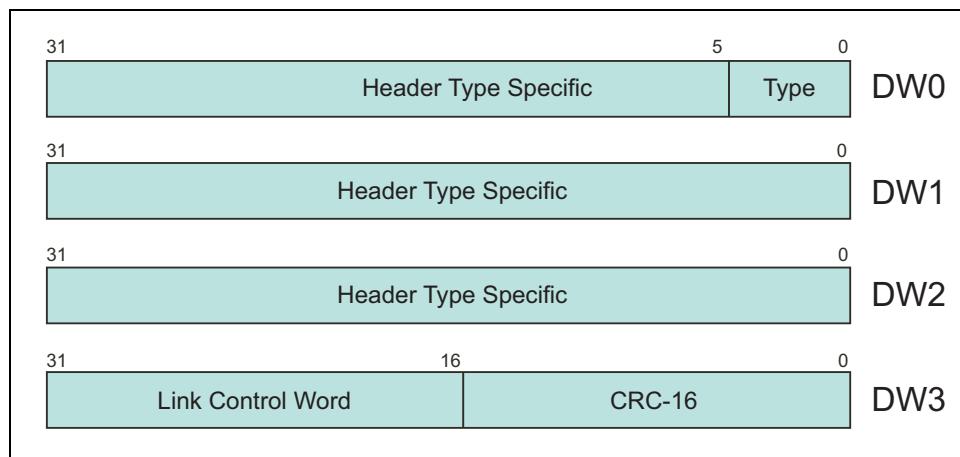
Table 5-1 lists the four Protocol Layer Header Types and indicates packet direction, type code and number of subtypes. All protocol layer packets consist of headers except for the Data Packet Payload. A basic header consists of 12 header bytes, along with a CRC-16 that covers the 12-byte header and a 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
<b>Transaction</b>	00100b	8	Sub-Type Specific	End-to-End
<b>Data (Header + Payload)</b>	01000b	NA	Bi-directional	End-to-End
<b>Isochronous Timestamp</b>	01100b	NA	Downstream Only	Multicast
<b>Link Management</b>	00000b	6	Mixed	Port-to-Port

Figure 5-2: Basic Header Packet



# USB 3.0 Technology

---

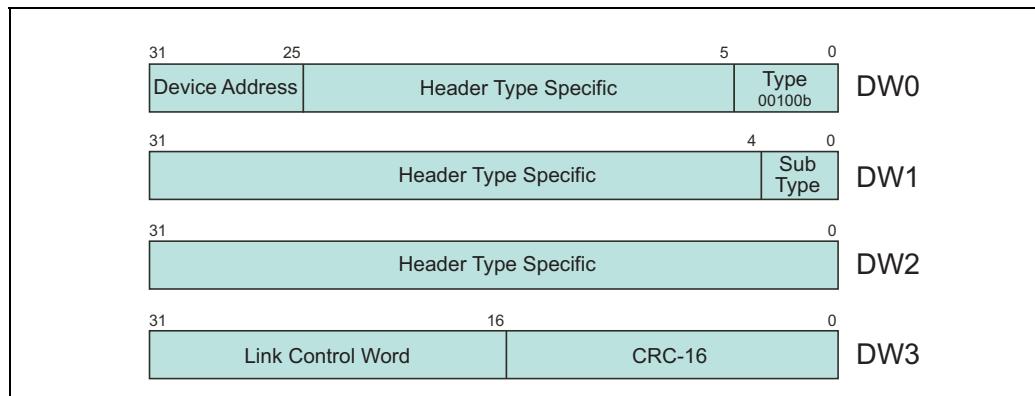
## Transaction Packets (TPs)

The eight TP subtypes are listed in Table 5-2. Note that some packets are bi-directions while others originate only at the USB Host Controller (downstream packets) and others originate only at devices (upstream packets). Figure 5-3 illustrates the basic Header Packet fields used with all Transaction Packets.

Table 5-2: Transaction Packet Sub Types

Name	SubType Code	Direction
Reserved	0000b	NA
ACK	0001b	Bi-directional
NRDY	0010b	Upstream Only
ERDY	0011b	Upstream Only
STATUS	0100b	Downstream Only
STALL	0101b	Upstream Only
NOTIFICATION	0110b	Upstream Only
PING	0111b	Downstream Only
PING RESPONSE	1000b	Upstream Only
Reserved	1001b - 1111b	NA

Figure 5-3: Transaction Packet - Basic Format



## **Chapter 5: End-to-End Packets**

---

### **ACK Header**

The ACK (Acknowledgement) Header is used in two distinct ways and may be sent from the host or from a USB device. The different roles of the ACK Header depend on the direction of Header packet movement as follows:

- ACK Headers being sent from host to device
  - ACK headers have an embedded IN token packet that initiate IN transactions.
  - ACK headers perform the role of the typical ACK handshake packet by notifying the device that data was successfully received by the host.
  - During back-to-back transactions a single ACK header may acknowledge receipt of the previously received data payload and request a new IN transaction at the same time.
  - Indicates the number of data buffers currently available in the host controller.
- ACK Header sent from device to host
  - ACK headers perform the role of the typical ACK handshake packet by notifying the host that data was successfully received by the device.
  - Also indicates the number of receiver buffers now available in the device.

### **Downstream Moving ACK**

Figure 5-4 on page 84 illustrates the header packet format and the fields used when an ACK is sent from the host controller. When initiating an IN transaction the ACK header contains the token information as listed below.

- 7-bit Device Address — DW0 bits 31:25
- 4-bit Endpoint Number — DW1 bits 11:8
- Direction of data movement Device to Host (IN) — DW1 bit 7 = 1

Unlike USB 2.0 the SuperSpeed bus sends transactions based on a unicast rather than a broadcast. The host controller uses a routing mechanism that directs packets to a specific target device. The device then verifies that the transaction is intended for the device by checking that the device address matches.

# USB 3.0 Technology

---

Figure 5-4: ACK Header Fields - Moving Downstream

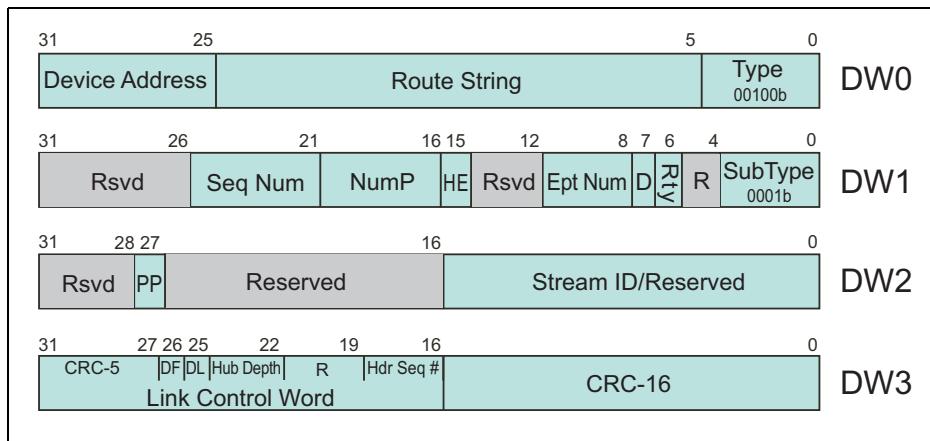


Table 5-3: ACK Header Field Descriptions — Header Moving Downstream

DW#	Name	Offset	Description
DW0	Type	0	Transaction Packet (TP) Code = 00100b
	Route String	5	Five fields of 4 bits each. Used by SS hub to forward ACK onto proper downstream port
	Device Address	25	Address assigned by USB configuration software
DW1	SubType	0	ACK Code = 0001b
	Retry	6	Host detected error in last data payload received. ACK sends SeqNum of failed packet, requests retry
	Direction	7	Direction data movement is IN; D = 1
	Endpoint Number	8	The USB device assigns numbers to all of its endpoints and is reported in the endpoint descriptors
	Host Error	15	Host did not receive data payload due to host controller error — Retry bit must also be set
	Number of Packets	16	Number of Data packet buffers available at host. Must be set to value equal to or less than maximum burst size of endpoint (1-16)
	Sequence Number	21	Indicates End-to-End sequence number for the next Data packet to be transferred

## Chapter 5: End-to-End Packets

---

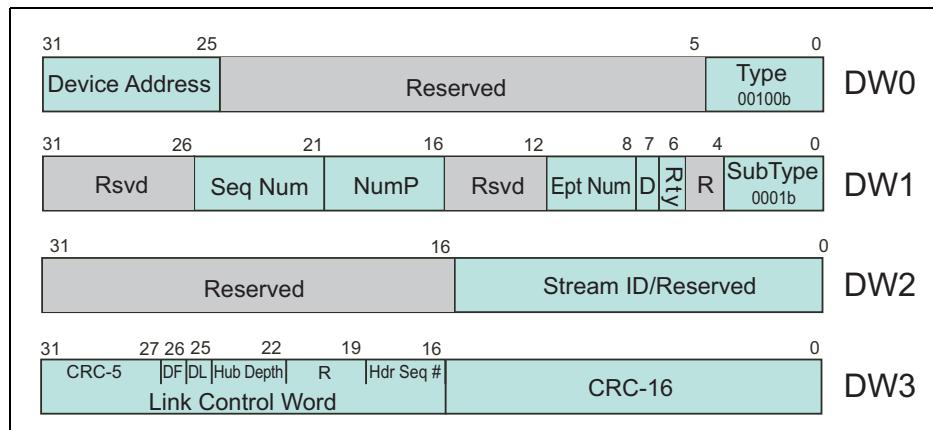
*Table 5-3: ACK Header Field Descriptions — Header Moving Downstream (Continued)*

DW#	Name	Offset	Description
DW2	Stream ID/Reserved	0	Identifies the StreamID value (1-64K) used in SS bulk endpoints that support bulk streaming protocol
	Packet Pending	27	Host sets bit indicating another packet is pending, allowing aggressive power management. Also used in bulk streaming protocol.
DW3	CRC-16	0	CRC value that covers DW0 - DW2
	Link Control Word	16	Sets Link Sequence Number, reports other events and Generates CRC across the Link Control Word.

### Upstream Moving ACK

Figure 5-5 illustrates the ACK Header contents when sent from a device to the host. Table 5-4 on page 86 describes the bit fields associated with an ACK header moving in the upstream direction.

*Figure 5-5: ACK Header Fields - Moving Upstream*



## **USB 3.0 Technology**

---

*Table 5-4: ACK Header Field Descriptions — Header Moving Upstream*

DW#	Name	Offset	Description
DW0	Type	0	Transaction Packet (TP) Code = 00100b
	Device Address	25	Device address, endpoint number, and direction bit identifies the device that is sending the ACK header
DW1	SubType	0	ACK Code = 0001b
	Retry	6	Device detected error in last data payload received. ACK sends SeqNum of failed packet, requests retry
	Direction	7	Direction of data movement is OUT; D = 0
	Endpoint Number	8	Endpoint number, device address and direction bit identifies the device responding
	Host Error	15	Reserved
	Number of Packets	16	Number of Data packet buffers currently available within the device. Must be set to value equal to or less than maximum burst size of endpoint (1-16)
	Sequence Number	21	Indicates End-to-End sequence number for the next Data packet to be transferred
DW2	Stream ID/Reserved	0	Identifies the StreamID value (1-64K) used in SS bulk endpoints that support bulk streaming protocol
	Packet Pending	27	Reserved, during OUT transactions PP is associated with the DATA header
DW3	CRC-16	0	CRC value that covers DW0 - DW2
	Link Control Word	16	Manages link-specific events as the header packet traverses USB links.

---

## **NRDY Header**

Devices send NRDY (Not Ready) headers to the host controller when the device is temporarily unable to return (IN) or accept (OUT) data. The NRDY header packet is similar to the NAK handshake packet in USB 2.0 in that they both perform End-to-End flow control. The primary difference is that NRDY does not result in multiple retries as does NAK. Instead, upon receiving NRDY the host temporarily suspends the current transaction and awaits notification from the device to resume the transaction. The notification occurs when the device sends

## Chapter 5: End-to-End Packets

Endpoint Ready (ERDY), thereby informing the host that it's ready to resume the transaction. In response the host sends the transaction again. In this way, End-to-End retries are limited to just one, thus eliminating a potentially huge amount of bus activity and power consumption.

Figure 5-6 illustrates the fields within a NRDY header. Table 5-5 on page 87 lists and describes the fields used with the NRDY header. The token-related information; Device Address, EndPoint Number and Direction is supplied to identify the target endpoint being suspended.

Figure 5-6: NRDY Header Fields

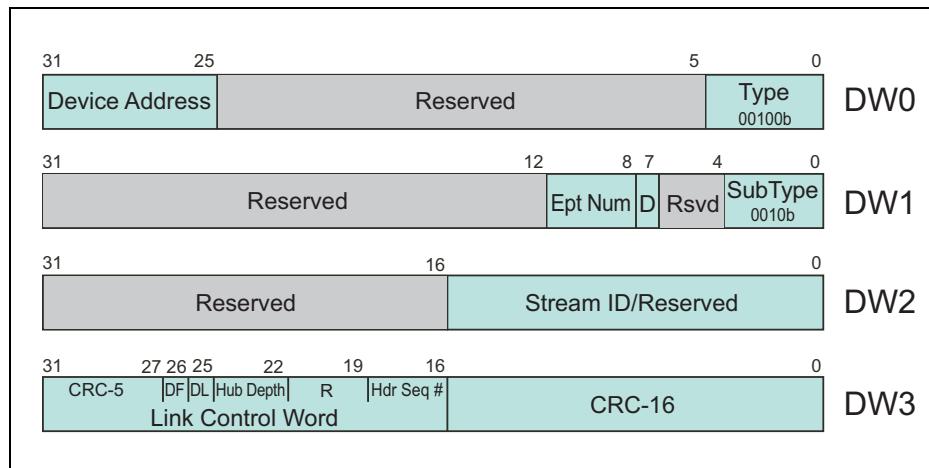


Table 5-5: NRDY Header Field Descriptions — Header Moving Upstream

DW#	Name	Offset	Description
DW0	Type	0	Transaction Packet (TP) Code = 00100b
	Device Address	25	Address assigned by USB configuration software
DW1	SubType	0	NRDY Code = 0010b
	Direction	7	Direction data movement may be IN (1) or OUT (0)
	Endpoint Number	8	Endpoint number, device address and direction bit identifies the device requesting the transaction be temporarily suspended

## **USB 3.0 Technology**

---

*Table 5-5: NRDY Header Field Descriptions — Header Moving Upstream (Continued)*

DW#	Name	Offset	Description
<b>DW2</b>	Stream ID/Reserved	0	Identifies the StreamID value (1-64K) used in SS bulk endpoints that support bulk streaming protocol
<b>DW3</b>	CRC-16	0	CRC value that covers DW0 - DW2
	Link Control Word	16	Manages link-specific events as the header packet traverses USB links

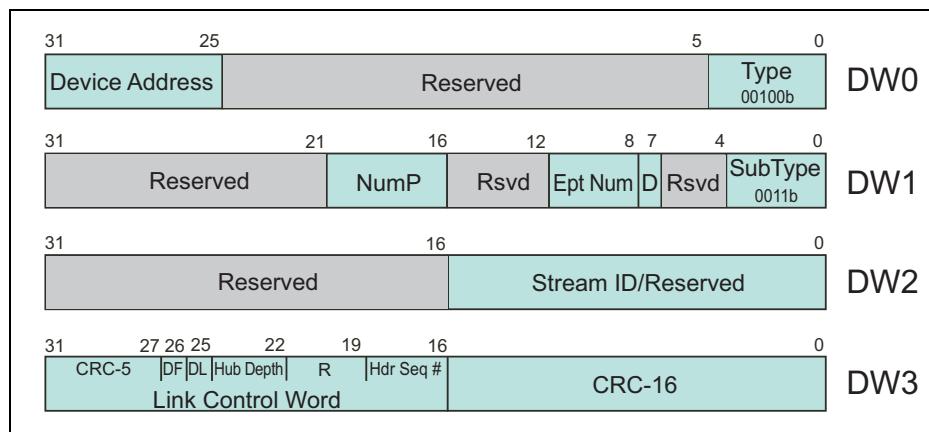
---

## **ERDY Header**

The ERDY (EndPoint Ready) header only moves from a device to the host to notify the host controller that the endpoint is ready to either accept or return a data payload. ERDY may be used in a variety of circumstances in response to a transaction being temporarily suspended.

Figure 5-7 illustrates the ERDY header fields and Table 5-6 on page 89 describes each field.

*Figure 5-7: ERDY Header Fields*



## Chapter 5: End-to-End Packets

---

Table 5-6: ERDY Header Field Descriptions — Header Moving Upstream

DW#	Name	Offset	Description
DW0	Type	0	Transaction Packet (TP) Code = 00100b
	Device Address	25	Address assigned by USB configuration software
DW1	SubType	0	NRDY Code = 0011b
	Direction	7	Direction data movement may be IN (1) or OUT (0)
	Endpoint Number	8	Endpoint number, device address and direction bit identifies the device and endpoint that is ready to be accessed.
	Number of Packets	16	NumP — reports the number of data packet buffers currently available within the device during OUT transactions or the current number of DATA packets ready to send to the host during IN transactions. This value must be equal to or less than maximum burst size of the endpoint (1-16)
DW2	Stream ID/Reserved	0	Identifies the StreamID value (1-64K) used in SS bulk endpoints that support bulk streaming protocol
DW3	CRC-16	0	CRC value that covers DW0 - DW2
	Link Control Word	16	Manages link-specific events as the header packet traverses USB links

---

## STATUS Header

The STATUS header (Figure 5-8 on page 90) is the last packet of any Super-Speed control transfer. It is always sent from the host to a control endpoint. See “STATUS Header Format and Content” on page 115 for details regarding its use. Table 5-7 on page 90 lists and describes each field within the STATUS header.

## USB 3.0 Technology

---

Figure 5-8: Status Header Packet

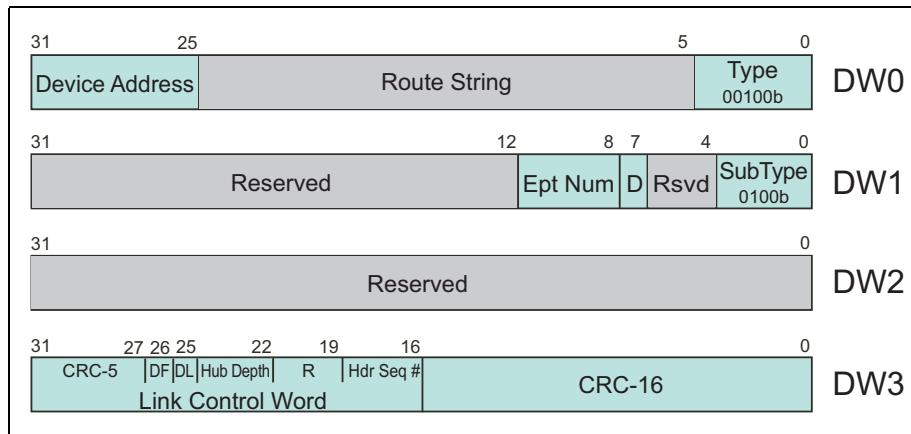


Table 5-7: STATUS Header Field Descriptions — Header Moving Downstream

DW#	Name	Offset	Description
DW0	Type	0	Transaction Packet (TP) Code = 00100b
	Route String	5	Five fields of 4 bits each. Used by SS hub to forward the STATUS header to proper downstream port
	Device Address	25	Address assigned by USB configuration software
DW1	SubType	0	STATUS Code = 0100b
	Direction	7	Direction data movement is either IN or OUT
	Endpoint Number	8	The USB device assigns numbers to all of its endpoints and is reported in the endpoint descriptors
DW2	Reserved	0	NA
DW3	CRC-16	0	CRC value that covers DW0 - DW2
	Link Control Word	16	Manages link-specific events as the header packet traverses USB links.

## Chapter 5: End-to-End Packets

---

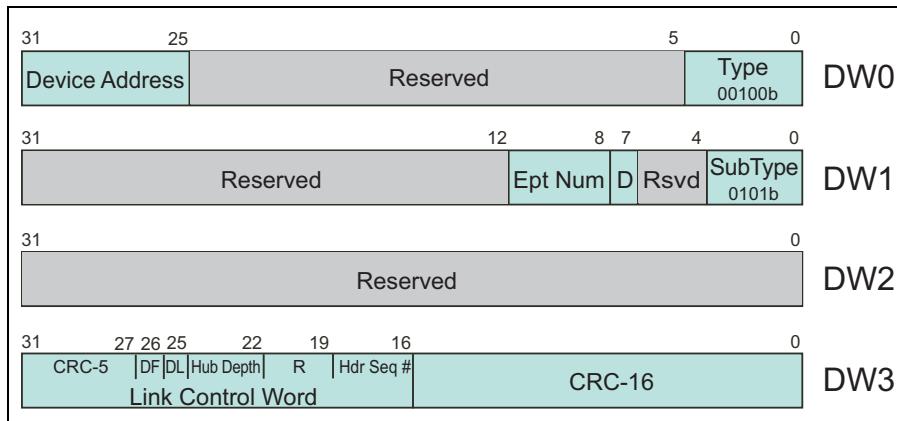
### STALL Header

A target endpoint sends a STALL header to report an error and host software is responsible for clearing the error condition. This is the same action taken with USB 2.0 devices. Stall headers are sent to the host in two occasions:

1. Function stalls are sent by endpoints other than zero and software is required to clear the endpoint stall condition. A ClearEndpointHalt request sent by software clears the function stall condition. Some implementations may simply reset the device.
2. A protocol stall is returned from a control endpoint and may be used to indicate that a particular control transfer request is not supported. This stall is automatically cleared with the start of the new control transfer (i.e., a Setup Transaction).

Figure 5-9 shows the field names and locations and Table 5-8 defines each field.

Figure 5-9: Stall Header Fields



# **USB 3.0 Technology**

---

*Table 5-8: STALL Header Field Descriptions — Header Moving Upstream*

DW#	Name	Offset	Description
DW0	Type	0	Transaction Packet (TP) Code = 00100b
	Device Address	25	Address assigned by USB configuration software
DW1	SubType	0	STALL Code = 0101b
	Direction	7	Direction data movement may be IN or OUT
	Endpoint Number	8	Endpoint number, device address and direction bit identifies the device and endpoint that is stalled
DW2	Stream ID/Reserved	0	Identifies the StreamID value (1-64K) used in SS bulk endpoints that support bulk streaming protocol
DW3	CRC-16	0	CRC value that covers DW0 - DW2
	Link Control Word	16	Manages link-specific events as the header packet traverses each USB link

---

## **Notification Headers**

SuperSpeed USB defines four Notification headers, all of which sent asynchronous messages to the host controller. The notification types are:

- Function Wake
- Latency Tolerance Message
- Bus Interval Adjustment Message
- Host Role Request

Each of the message headers are described in the subsequent sections, except the Host Role Request that is used in SuperSpeed On-The-Go (OTG).

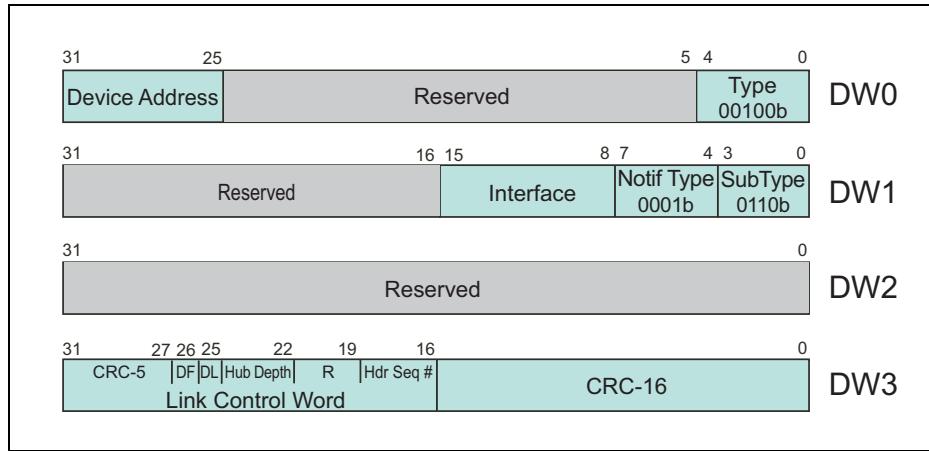
### **Function Wake Header**

One or more functions within a USB SS device may support waking up a device and link so that the function is serviced by software. The header is sent to the host controller to notify software of the wake event. Details regarding Function Wake can be found in the Chapter on Power Management. Figure 5-10 on page 93 illustrates the format and fields associated with the Function Wake header and Table 5-9 on page 93 describes each field.

## Chapter 5: End-to-End Packets

---

*Figure 5-10: Function Wake Notification Header — Header Moves Upstream*



*Table 5-9: Wake Notification Header Field Descriptions — Header Moves Upstream*

DW#	Name	Offset	Description
<b>DW0</b>	Type	0	Transaction Packet (TP) Code = 00100b
	Device Address	25	Device address identifies the device that is sending the WAKE Notification header
<b>DW1</b>	SubType	0	Notification Header Code = 0110b
	Notification Type	4	Wake Notification Code = 0001b
	Interface	8	Identifies the first interface within the function that causes the wake event
<b>DW2</b>	Reserved	0	NA
<b>DW3</b>	CRC-16	0	CRC value that covers DW0 - DW2
	Link Control Word	16	Manages link-specific events as the header packet traverses USB links

# USB 3.0 Technology

---

## Latency Tolerance Message Header

A device may elect to support Latency Tolerance Messages that notify the host of the device's current latency tolerance. For example, a device may indicate in its Latency Tolerance Message the amount of time it can wait between sending ERDY and receiving service from the Host. See "Latency Tolerance Message Reporting" on page 585 for details regarding this implementation.

Figure 5-11 illustrates the Latency Tolerance Message header format and fields and Table 5-10 describes each of the fields.

Figure 5-11: Latency Tolerance Message Header

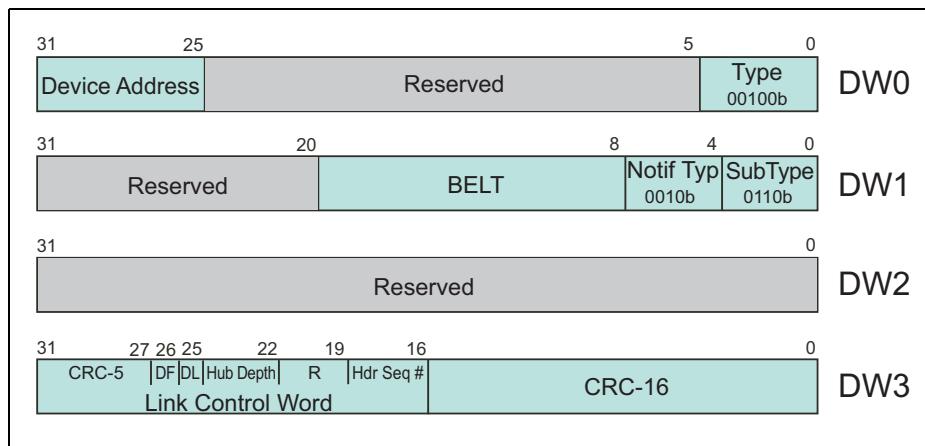


Table 5-10: Latency Tolerance Message Header Field Descriptions

DW#	Name	Offset	Description
<b>DW0</b>	Type	0	Transaction Packet (TP) Code = 00100b
	Device Address	25	Device address identifies the device that is sending the Latency Tolerance Notification header
<b>DW1</b>	SubType	0	Notification Header Code = 0110b
	Notification Type	4	Latency Tolerance Notification Code = 0010b
	BELT	8	Defines the current "Best Effort Latency Tolerance" value

## Chapter 5: End-to-End Packets

Table 5-10: Latency Tolerance Message Header Field Descriptions (Continued)

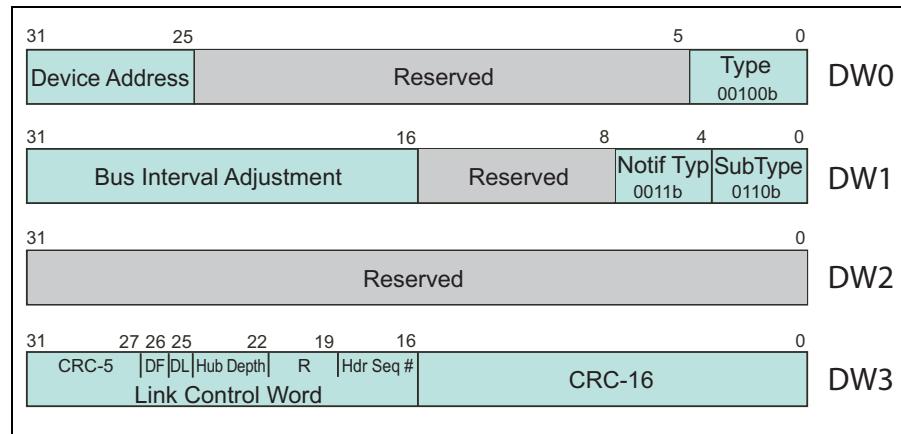
DW#	Name	Offset	Description
DW2	Reserved	0	NA
DW3	CRC-16	0	CRC value that covers DW0 - DW2
	Link Control Word	16	Manages link-specific events as the header packet traverses USB links.

### Bus Interval Adjustment Message Header

The 125μs bus interval may be adjusted when a device sends a Bus Interval Adjustment message to the host. This request may or may not accept the adjustment request. The section entitled, “Bus Interval Adjustment Message Header” on page 95 describes the Bus Interval Adjustment process that an Isochronous Timestamp header may synchronize to.

Figure 5-12 illustrates the format of the Bus Interval Adjustment header and Table 5-11 on page 96 describes the fields.

Figure 5-12: Bus Interval Adjustment Message Header



## **USB 3.0 Technology**

---

*Table 5-11: Bus Interval Adjustment Header Field Descriptions*

DW#	Name	Offset	Description
DW0	Type	0	Transaction Packet (TP) Code = 00100b
	Device Address	25	Device address identifies the device that is sending the Bus Interval Adjustment header
DW1	SubType	0	Notification Header Code = 0110b
	Notification Type	4	Bus Interval Adjustment Code = 0011b
	Bus Interval Adjustment	16	Defines the magnitude of the bus interval adjustment to be made
DW2	Reserved	0	NA
DW3	CRC-16	0	CRC value that covers DW0 - DW2
	Link Control Word	16	Manages link-specific events as the header packet traverses USB links.

---

## **PING and PING RESPONSE Header**

Isochronous transactions must complete within their specified service interval. Because one or more links in the path between the isochronous endpoint and root hub port may be in a low-power state, an isochronous transaction might not get completed in time due to link exit latency. To prevent this, a PING header packet must be sent to the isochronous endpoint prior to its service interval and a PING RESPONSE header must be returned to the host controller.

Figure 5-13 and Table 5-12 on page 97 show the header format and describe the fields associated with the PING header, and Figure 5-14 on page 98 and Table 5-13 on page 98 cover the PING RESEPONSE header.

## Chapter 5: End-to-End Packets

Figure 5-13: PING Header Fields – Header Moving Downstream

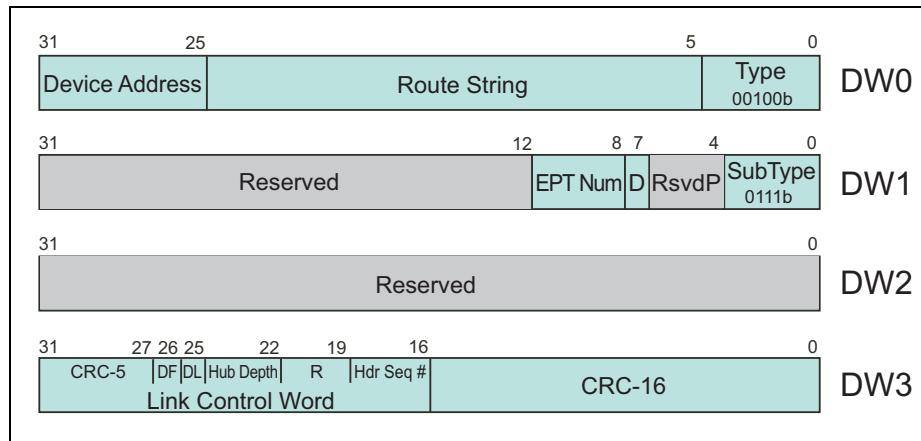


Table 5-12: PING Header Field Descriptions – Header Moving Downstream

DW#	Name	Offset	Description
DW0	Type	0	Transaction Packet (TP) Code = 00100b
	Route String	5	Five fields of 4 bits each. Used by SS hub to forward the PING header to proper downstream port
	Device Address	25	Address used to target the isochronous device with the Ping packet, including Endpoint number and direction bit.
DW1	SubType	0	PING Code = 0111b
	Direction	7	Direction data movement is OUT (0)
	Endpoint Number	8	The USB device assigns numbers to all of its endpoints and is reported in the endpoint descriptors
DW2	Reserved	0	NA
DW3	CRC-16	0	CRC value that covers DW0 - DW2
	Link Control Word	16	Manages link-specific events as the header packet traverses USB links.

## USB 3.0 Technology

---

Figure 5-14: PING RESPONSE Header – Header Moving Upstream

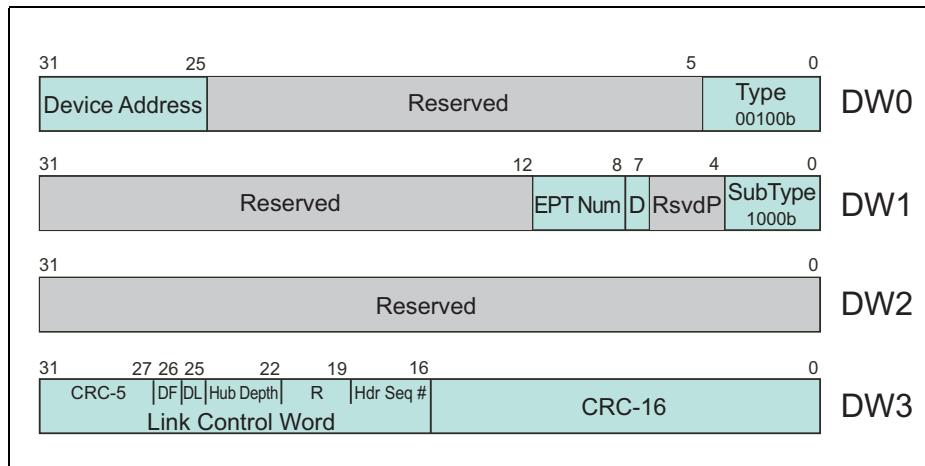


Table 5-13: PING RESPONSE Header Field Descriptions

DW#	Name	Offset	Description
DW0	Type	0	Transaction Packet (TP) Code = 00100b
	Device Address	25	Address returned from the device
DW1	SubType	0	PING RESPONSE Code = 1000b
	Direction	7	Direction bit must be set to the same value of the PING header (D=0)
	Endpoint Number	8	Endpoint number, device address and direction bit identifies the isochronous endpoint
DW2	Stream ID/Reserved	0	Reserved
DW3	CRC-16	0	CRC value that covers DW0 - DW2
	Link Control Word	16	Manages link-specific events as the header packet traverses USB links

## **Chapter 5: End-to-End Packets**

---

### **Data Packet**

A DATA packet consists of two separate components:

- DATA Packet Header (DPH) — 12-Byte header, CRC-16 and LCW
- DATA Packet Payload (DPP) — 0 - 1024 Bytes, CRC-32

The DATA header has different roles depending on the direction of Header packet movement as follows:

- DATA Header being sent from host to device
  - DATA headers can embed an OUT token packet that initiates OUT transactions.
  - DATA headers can embed a Setup token packet that initiates Control transfers.
  - DATA headers have an accompanying DATA payload that is sent to a device OUT endpoint.
- DATA Header sent from device to host
  - During IN transactions, DATA headers and the associated DATA Payload return data to the host controller in response to an ACK header from the host.

Use of the Data Packet Header (DPH) fields also varies depending on whether an IN or OUT transaction is in progress. These differences are covered in the following sections.

The DATA Packet Payload may be dropped due to certain error conditions, leaving only the DATA header to be forwarded on to the recipient. These conditions are covered in later chapters.

---

### **DATA Packet Moving Downstream**

Figure 5-15 on page 100 illustrates the fields associated with a DATA packet moving in the downstream direction. Table 5-14 on page 100 describes each field in the DATA header, except reserved fields are not listed.

The DATA Packet Payload may have a zero byte payload in which case the DPP consists of the CRC-32. Also, the payload is not required to be aligned on four-byte boundaries and can end on any byte boundary.

## USB 3.0 Technology

---

Figure 5-15: Data Header and Payload Moving Downstream

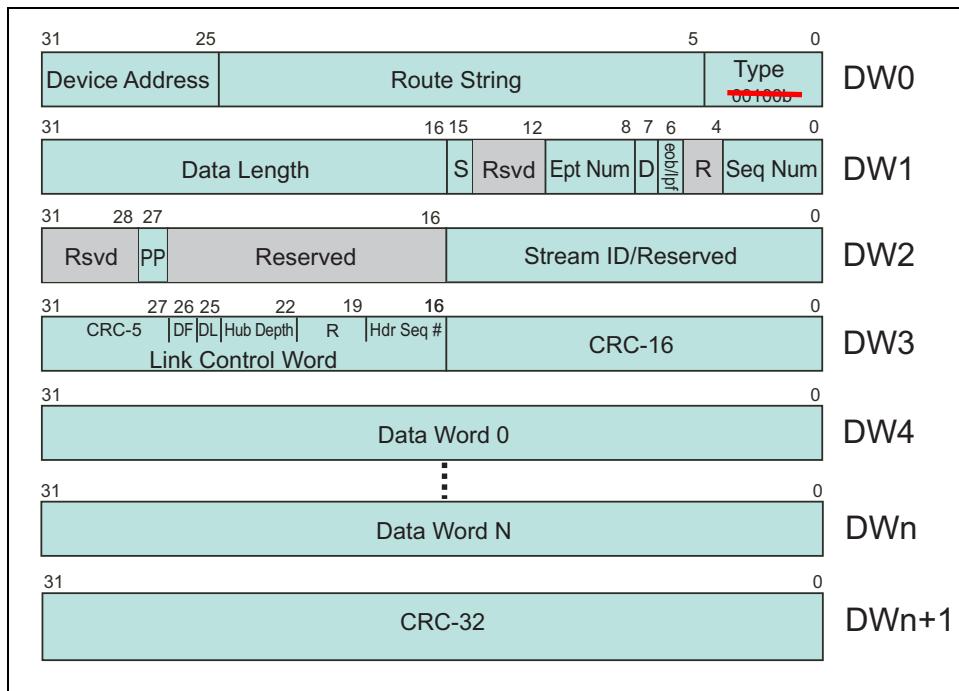


Table 5-14: DATA Header Field Descriptions — Header Moving Downstream

DW#	Name	Offset	Description
DW0	Type	0	DATA Packet (TP) Code = 01000b
	Route String	5	Five fields of 4 bits each. Used by SS hub to forward the DATA header to proper downstream port
	Device Address	25	Address assigned by USB configuration software

## Chapter 5: End-to-End Packets

---

Table 5-14: DATA Header Field Descriptions — Header Moving Downstream (Continued)

DW#	Name	Offset	Description
DW1	Sequence Number	0	A 5-bit sequence number is assigned to each DATA header and may roll over from 31 back to 0 and allows confirmation of successful data delivery or not via the ACK packet.
	eob/lpf	6	<ul style="list-style-type: none"><li>• eob (end of burst) provides flow control during DATA burst transfers for bulk and interrupt endpoints.</li><li>• lpf (last packet flag) is only set in the DATA header of the last data packet of an isochronous service interval.</li></ul>
	Direction	7	Direction data movement is OUT; D = 0
	Endpoint Number	8	The USB device assigns numbers to all of its endpoints and is reported in the endpoint descriptors
	Setup	15	When S=1, the DATA header is sending a Setup token packet and Data Length must be 8 bytes
	Data Length	16	Number of bytes in the DPP
DW2	Stream ID/Reserved	0	Identifies the StreamID value (1-64K) used in SS bulk endpoints that support bulk streaming protocol
	Packet Pending	27	Host sets bit indicating another packet is pending, allowing aggressive power management. Also used in bulk streaming protocol.
DW3	CRC-16	0	CRC value that covers DW0 - DW2
	Link Control Word	16	Manages link-specific events as the header packet traverses USB links.

---

## DATA Packet Moving Upstream

DATA packets moving in the upstream direction go directly to the root hub port and the Route String field is reserved. Figure 5-16 on page 102 illustrates the format and fields and Table 5-15 on page 102 describes all fields except those that are reserved.

## USB 3.0 Technology

Notice that the DATA Packet Payload in Figure 5-16 shows a payload that does *not* end on a DW boundary to reinforce the principle of ending on any byte boundary.

Figure 5-16: *Data Header and Payload Moving Upstream*

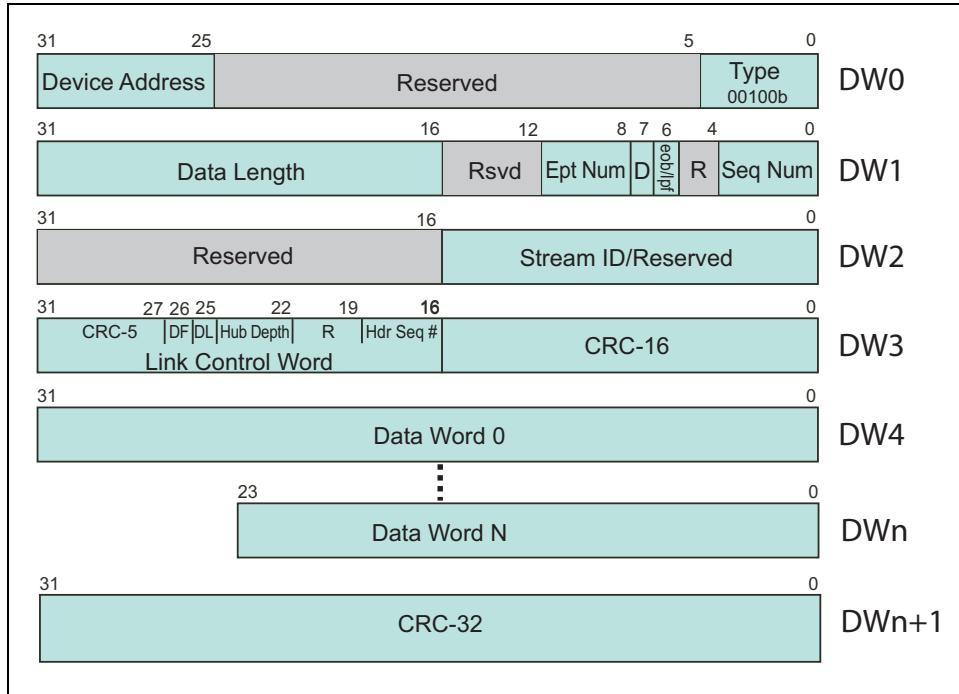


Table 5-15: DATA Header Field Descriptions — Header Moving Upstream

DW#	Name	Offset	Description
DW0	Type	0	DATA Packet (TP) Code = 01000b
	Device Address	25	Address assigned by USB configuration software

## Chapter 5: End-to-End Packets

---

Table 5-15: DATA Header Field Descriptions — Header Moving Upstream (Continued)

DW#	Name	Offset	Description
DW1	Sequence Number	0	A 5-bit sequence number is assigned to each DATA header and may roll over from 31 back to 0 and allows confirmation of successful data delivery or not via the ACK packet.
	eob/lpf	6	<ul style="list-style-type: none"><li>• eob (end of burst) provides flow control during DATA burst transfers for bulk and interrupt endpoints.</li><li>• lpf (last packet flag) is only set in the DATA header of the last data packet of an isochronous service interval.</li></ul>
	Direction	7	Direction data movement is OUT; D = 0
	Endpoint Number	8	The USB device assigns numbers to all of its endpoints and is reported in the endpoint descriptors
	Data Length	16	Number of bytes in the DPP
DW2	Stream ID/Reserved	0	Identifies the StreamID value (1-64K) used in SS bulk endpoints that support bulk streaming protocol
DW3	CRC-16	0	CRC value that covers DW0 - DW2
	Link Control Word	16	Manages link-specific events as the header packet traverses USB links.

---

### Isochronous Timestamp Packet (ITP)

This header packet replaces the Start of Frame (SOF) packet used in USB 2.0. Rather than sending a packet precisely on a 125 $\mu$ s bus interval boundary, the Isochronous Timestamp Packet is sent within an 8 $\mu$ s window at the beginning of each bus interval. The ITP broadcast header conveys bus interval timing that targets all links and devices that are not in a low-power state. Upon receiving the bus timing information a device may lock its internal timebase to the USB bus timing.

Figure 5-17 on page 104 depicts the Isochronous Timestamp Packet and identifies each field within the header. Table 5-16 on page 104 describes each of the fields except the reserved fields that are not listed.

# USB 3.0 Technology

---

Figure 5-17: Isochronous Timestamp Header

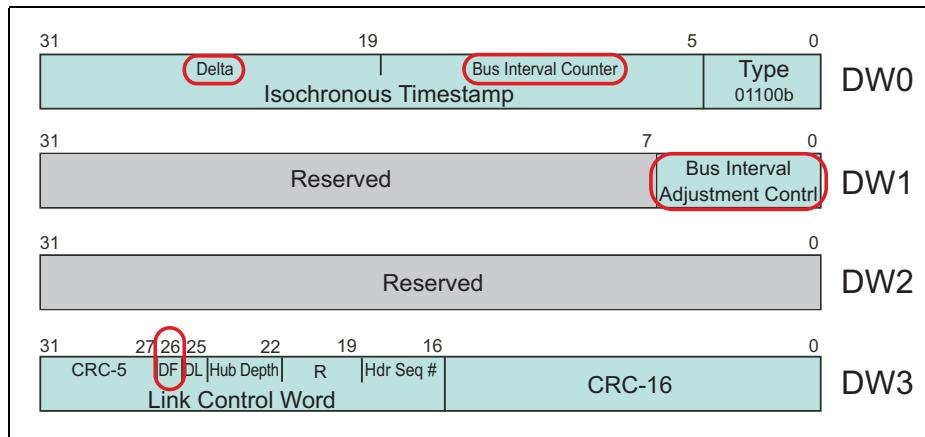


Table 5-16: ISOCHRONOUS TIMESTAMP Header Field Descriptions

DW#	Name	Offset	Description
<b>DW0</b>	Type	0	Isochronous Timestamp Packet (ITP) Code = 01100b
	Isochronous Timestamp	25	<ul style="list-style-type: none"><li>Bus Interval Counter — reports the current value of the 125µs bus interval counter that ranges from 0-16383 intervals</li><li>Delta — the delay between the aligned bus interval boundary and ITP transmission</li></ul>
<b>DW1</b>	Bus Interval Adjustment Control	0	This field contains the address of the device that previously sent a Bus Interval Adjustment Message, if the host has granted the adjustment request. A value of zero indicated the request has been rejected.
<b>DW2</b>	Reserved	0	NA
<b>DW3</b>	CRC-16	0	This CRC value covers DW0 - DW2
	Link Control Word	16	Manages link-specific events as the header packet traverses USB links

---

---

# 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 specifica-

## **USB 3.0 Technology**

---

tion 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.

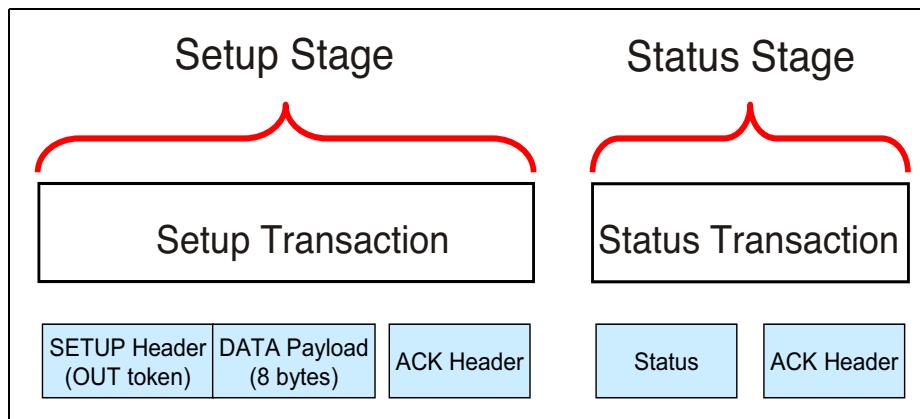
## Chapter 6: Control Protocol

---

### 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

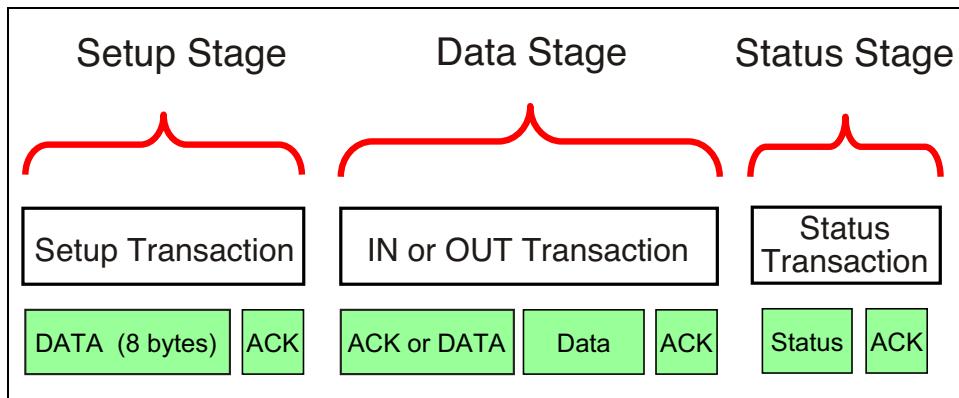
Three stage control transfers consist of the Setup Stage followed by the Data stage that consists of one or more IN or OUT transactions depending on the specific request and ends with the Status Stage (See Figure 6-2). The most common types of three-stage control transfer are getting descriptor or status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

## USB 3.0 Technology

maximum payload size of a SuperSpeed Data packet is eight times larger than the USB 2.0 payload size (512 vs 64 bytes). Consequently, the number of data transactions within the Data Stage will be fewer with SuperSpeed than in USB 2.0.

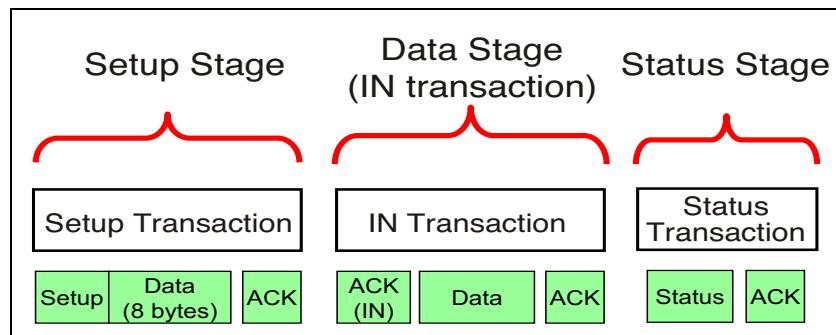
*Figure 6-2: Three Stage Control Transfer Structure (IN or OUT)*



### **Control IN Example — Single Data Packet**

Figure 6-3 on page 108 illustrates a three-stage transfer in which a single transaction is performed in the data stage. The ACK header initiates the IN token request and the Data Header and DPP are returned from the device. Subsequently an ACK packet sent from the host acknowledges successful receipt of the data.

*Figure 6-3: Example Three-Stage Transfer with Single IN Transaction*



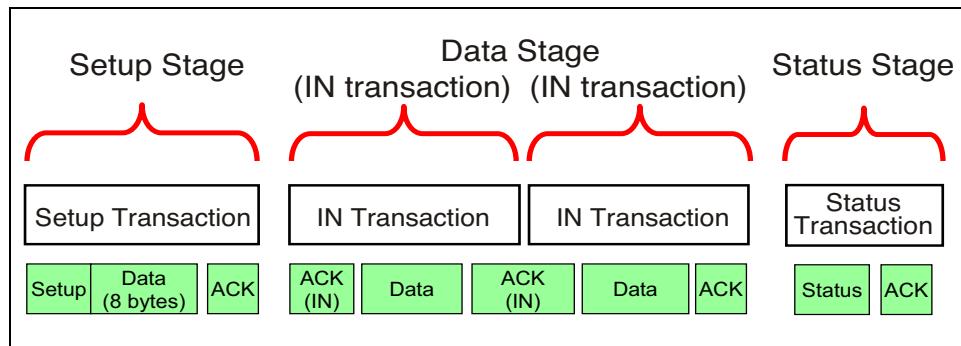
# Chapter 6: Control Protocol

## Control IN Example — Two Data Packets

Figure 6-4 illustrates the transaction associated with three-stage control transfers. Notice that this example shows two IN transactions associated with the data stage. The Data Stage begins with the host controller sending an ACK (IN token) and the device returns the first Data packet (includes Header and DPP). The host sends the next ACK that performs two functions:

1. Acknowledges reception of the first Data packet and
2. Issues an IN token request for the next Data packet

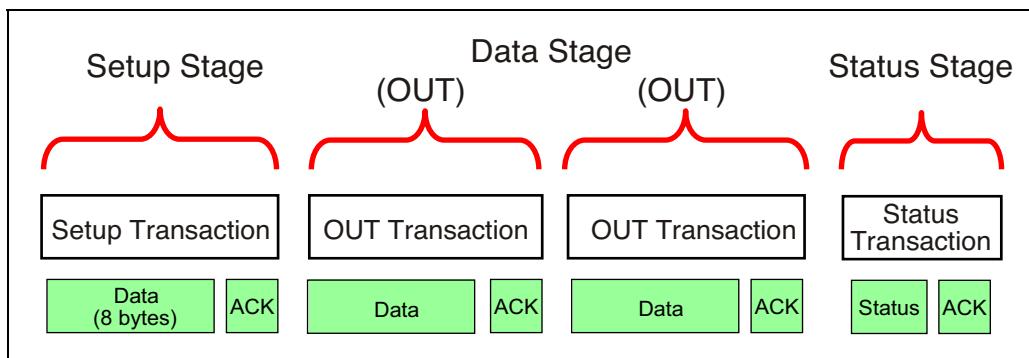
Figure 6-4: Example Two-Packet Data Stage



## Control OUT Example — Two DATA Packets

The Control OUT transaction examples are straight forward in that each Data transaction begins with the host sending a Data Header and DPP and the target device responds with an ACK packet that verifies successful receipt of the data.

Figure 6-5: Example Two-Packet OUT Data Stage



# **USB 3.0 Technology**

---

---

## **Control Transfer Packet Content**

The previous sections discussed the primary structure of the control protocols. This section details the content of the Setup, Data and Status stages.

---

### **The Setup Transaction**

The Setup transaction defines the action to be performed by the target device. This section defines the requirements and options that are available to software for performing control transfers. Devices must support a collection of device descriptors required for configuration and basic operation.

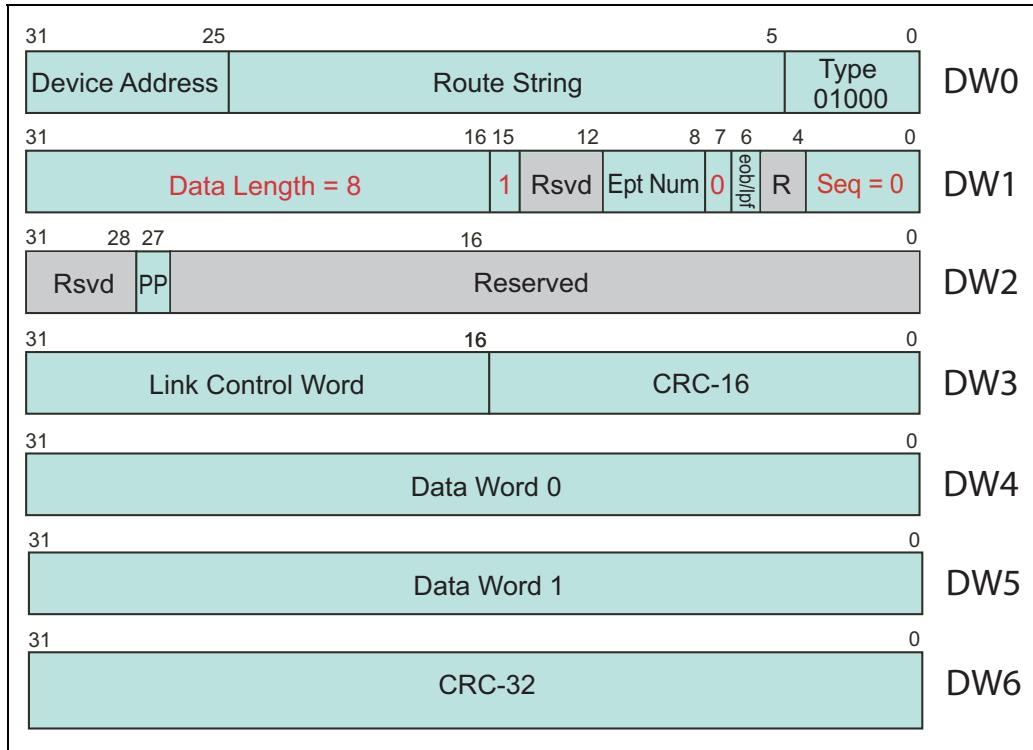
#### **The DATA Header Packet**

Control transfers always begin with a DATA header. The following list of parameters must be applied to the DATA header to define the DATA header as a Setup transaction. Also see Figure 6-6 on page 111.

- The Data Header packet associated with the Setup Stage must have DW1, bits 3:0 cleared to zero so that every Setup transaction begins with sequence number zero.
- The Data Header must have DW1, bit 15 set (1) so the device recognizes this Data header as the beginning of a Setup transaction.
- The direction bit (DW1, bit 7) must be cleared (0) to indicate data is moving toward the device.
- DW1, bits 31:16 must report a value of 8. The eight bytes of data define the contents of the Data Packet Payload and the request to be performed.

## Chapter 6: Control Protocol

Figure 6-6: DATA Header — Setup Request



### Data Packet Payload — Setup Data

The 8-byte Data Packet Payload (DPP) defines the request to be performed. Table 6-1 on page 112 defines the fields within the 8-byte request structure. Each field is further described in the bullet list that follows.

## **USB 3.0 Technology**

---

*Table 6-1: Setup Stage Request Data*

Offset	Field Name	Size	Value	Description
<b>0</b>	Request Type	1	Bitmap	Request Characteristics  D7 - Data Transfer Direction: 0 = Host to device 1 = Device to host  D6:5 - Request Type: 0 = Standard 1 = Class 2 = Vendor 3 = Reserved  D4:0 - Recipient 0 = Device 1 = Interface 2 = Endpoint 3 = Other (e.g., port) 4-31 = Reserved
<b>1</b>	Request	1	Value	Request Code
<b>2</b>	Value	2	Value	Value is request dependent
<b>4</b>	Index	2	Index/Offset	Content varies by request. Typically an offset or index
<b>6</b>	Length	2	Count	Number of bytes to transfer 0 = No Data Stage 1-15 = Data transfer size (1 to 64Kbytes)

- **Request Type** field description:

- D7; The movement of the data during a control transfer is specified in this bit field. Two-stage transfers always involve data move from host to device. The direction during three-stage transfers depend on the direction of the Data stage packets.
- D6:5; The type of request depends on the software that is performing the operation as follows:

## Chapter 6: Control Protocol

---

- **Type 0;** The software performing a control transfer may be the USB system software responsible for the configuration and other standard operations common to all devices. These requests are called **Standard** requests. See Table 22-1 on page 486 for a list of standard transfers.
- **Type 1;** All device attached to USB have one or more functions, called classes (e.g., Mass Storage, Hub, Audio and many more) When software performs a control transfer related to a specific function it is called a **Class** specific request.
- **Type 2;** Finally, implementations created by a manufacturer that are not defined by any USB class, are referred to as **Vendor** specific requests.
- D4:0; The recipient identifies the entity being targeted by a request.
  - **Recipient 0;** If the request is related to overall device operation (e.g., Set Address) then the **device** is the recipient.
  - **Recipient 1;** Similarly, the **interface** is targeted when a SetInterface or SuspendFunction request is sent.
  - **Recipient 2;** **Endpoints** are the recipient when a ClearEndpointHalt request is sent.
  - **Recipient 3;** Finally, the category of **other** relates to class- or vendor-specific request that target some other entity such as a hub port.
- **Request** field; This entry contains the request code. A given request may require the Value, Index, and Length fields be loaded with information.
- **Value** field; The contents of this field varies according to the request code. For example, if a GetDescriptor code (06h) was loaded into the Request field, the value field must specify the type of descriptor being requested (Device, Configuration, String, etc.).
- **Index** field; The content of this field also varies with the request type code. Sometimes this field is used as an index (e.g., the Language ID value is required by the String descriptor) or it may be used as an offset to specific a port number for a variety of hub-specific requests.
- **Length** field; This field specifies the size of the buffer that software has setup for sending or receiving data during the Data stage of a control transfer. The maximum payload is limited to 64Kbytes due to the 16-bit field.

The eight bytes of setup data is also summarized in Figure 6-7 on page 114. The byte values are consistent with the transmission order of packets across each link, with the least significant byte sent first. Consequently, all two-byte fields show the low byte on the left and the high byte on the right. As a side note, this view of the byte order matches USB protocol analyzer presentations of the packets.

## **USB 3.0 Technology**

---

*Figure 6-7: Setup Data Fields Showing Byte Order of Packet Transmission Across the Link*

LSB			MSB		
Request Type	Request (1 byte)	Value (2 bytes)	Index (2 bytes)	Length (2 bytes)	Data Stage
10000000B	GET_DESCRIPTOR (06h)	Descriptor Type (0001h)	Zero	Length (1200h)	Descriptor Data

### **Setup Response — ACK**

The only legal response to a Setup Transaction is an ACK header. Only one control transfer resource exists and it targets only the default control endpoint. Thus, a NRDY response is not needed because EP0 will always be ready as a result of the last Status transaction having completed. Similarly no STALL handshake is permitted because if the control transfer is malformed the STALL will be returned with the next host access to EP0.

---

### **DATA Stage**

The Data stage of control transfers is optional and is simply a series of IN or OUT transactions. The Data stage always begins with sequence number zero. The maximum payload size of each packet is 512 bytes and the total payload is specified in the Length field of the Setup Data packet as follows:

- The host controller will always send the exact amount of data specified in the Length field during control OUT transactions.
- The host controller will always read the exact amount of data specified in the Length field during control IN transactions, unless the actual amount of data returned is less than the total Length, in which case a short packet is reported by the target device.
- During the Data stage, a STALL header may be returned in response to a host initiated IN or OUT transaction. In these cases, the control transfer simply ends with the STALL header. A subsequent Setup transaction automatically clears the Stall condition.

# Chapter 6: Control Protocol

---

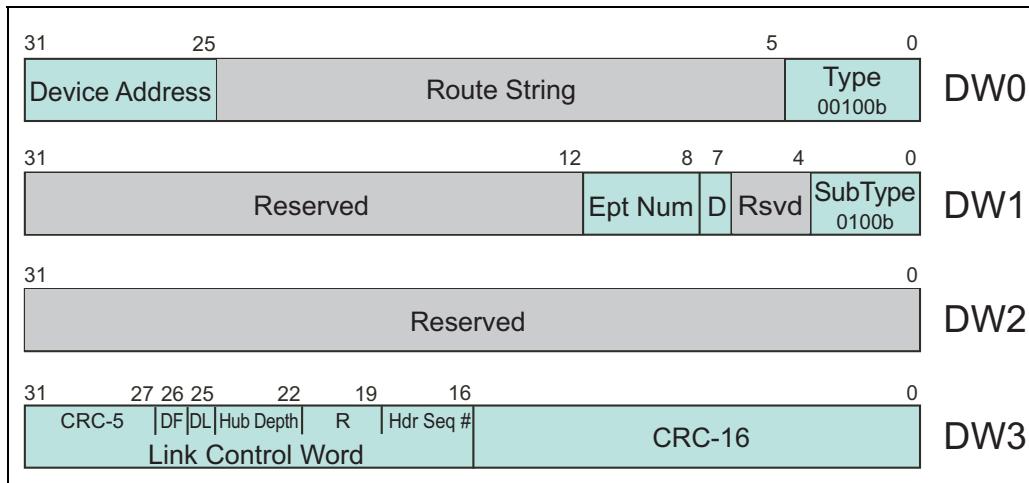
## Status Stage

### General

The Status stage of all SuperSpeed control transfers begin with the host controller sending a Status header packet (See Figure 6-8). In response, control EP0 may return one of four header packets:

- **ACK** header indicating the control transfer completed successfully
- **NRDY** header notifying the host that the control transfer is pending completion
- **ERDY** header notifying the host that the control transfer has completed and to get final status (ACK or STALL) by sending another Status header.
- **STALL** header notifying the host controller that the control transfer has failed

Figure 6-8: STATUS Header Format and Content



### Control Transfer with STALL

STALL header packets returned during the Status stage may indicate that this particular request is not supported. These stall conditions are called **protocol stalls** and are cleared by the next Setup token.

# **USB 3.0 Technology**

---

## **Two Stage Control Transfer Example**

There are many types of two-stage control transfers that request the device, interface, endpoint or other entity (e.g., port) within a device to perform some action. Examples include:

- Set Port Power (hub specific)
- Set Address
- Set Isochronous Delay
- Set Inactivity Timeout value
- Clear Endpoint Stall (all endpoints other than EP0)
- Clear Suspend
- Clear Function Suspend

The required Set Address Request has been chosen to detail the operation of the two-stage control transfers.

---

## **Set Address Request**

Following reset a SuperSpeed device responds only to device address zero. Upon successful completion of the Set Address request, the device will have been assigned a unique address assigned by the host controller at the request of host software. Subsequent transactions to this device will use the new address.

### **Setup Stage — Set Address**

The Setup transaction simply transfers the 7-bit Device Address to the target device via the 8-byte data payload. The Status Stage includes the Data header and Data Packet Payload (DPP) and the ACK header response. The Setup Token within the Data header will specify a device address of zero, an endpoint number of zero, and the direction bit will be zero (OUT). The payload size must be eight bytes as shown in Figure 6-9. The contents of the SetAddress request data is defined below:

- Request Type
  - Bit 7: Direction = Host to Device (0)
  - Bits 6:5: Standard Request (00)
  - Bits 4:0: Recipient = Device (00000)
- Request = SetAddress (05h)
- Value = Address to be applied (03h in this example) in the low byte position
- Index and Length fields = 0000h (not used in this request)

## Chapter 6: Control Protocol

---

Figure 6-9: Set Address Request Data

LSB			MSB		
Request Type	Request (1 byte)	Value (2 bytes)	Index (2 bytes)	Length (2 bytes)	Data Stage
00000000B	SET_ADDRESS (05h)	Address (0300h)	Zero	Zero	None

### Status Stage — Set Address

A successful Status transaction verifies that the device is ready to respond to the new address. If the device has not applied the address when the Status transaction is sent by the host controller, the device must reply with an NRDY header. Subsequently, endpoint zero will return an ERDY header packet to notify the host that the address has been applied. In response, the host controller will send the Status header again and the endpoint zero will return ACK.

---

### Protocol Details for Set Address Request

Figure 6-10 on page 118 illustrates the protocol associated with the Set Address Request described previously. The following discussion enumerates each step in the control transfer protocol.

Setup Stage:

1. Host Tx — The Setup Stage of a Control transfer always uses a Data packet with a sequence number of zero (Seq=0), must have DW1; bit 15 set indicating this packet defines a Setup Token packet (Setup=1), and DW1 bits 31:16 must specify a DPP of eight bytes.
2. Device Tx — Upon receiving the 8-byte Data packet, the target device always returns an ACK packet. Seq=1 acknowledges receipt of the DATA packet and NumP=1 indicates the DPP has been received.

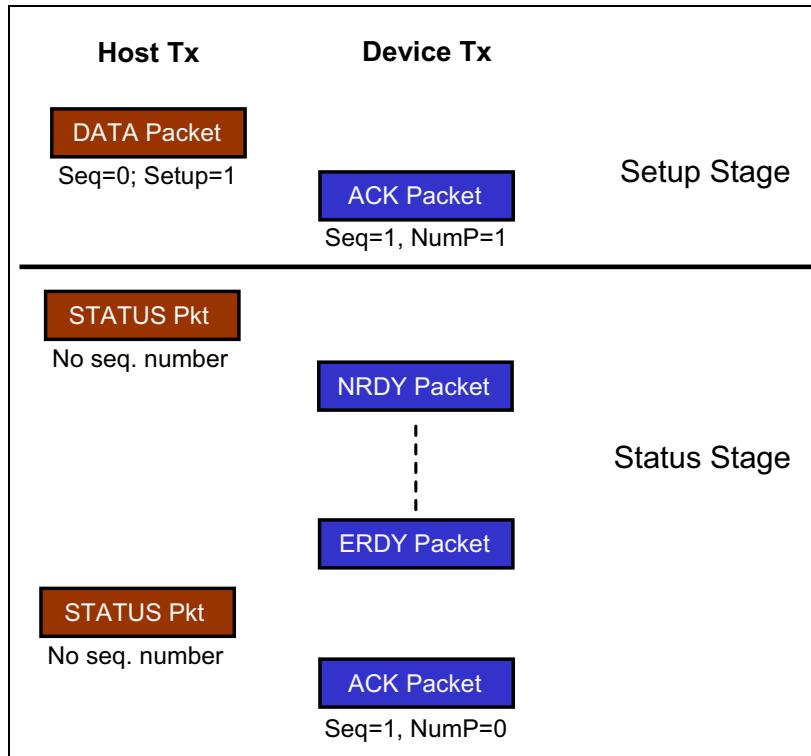
Status Stage:

1. Host Tx — The host controller transmits the STATUS packet that has no sequence number defined.
2. Device Tx — Upon receiving the STATUS packet, EP0 will return the current status of the request. In this example a NRDY packet is sent to the host controller because the device is not ready for access using the new address.

## USB 3.0 Technology

3. The host controller temporarily deactivates the transfer and is allowed to perform other transactions in the schedule until the device is ready to complete the control transfer.
4. **Device Tx — When the address has been successfully applied, the device notifies the host controller by sending an ERDY packet.**
5. Host Tx — The host controller sends the STATUS packet again upon receiving ERDY.
6. Device Tx — Device sends ACK packet to host and must set Seq=1 and NumP=0 to confirm completion of the control transfer.
7. The host controller retires the control transfer upon receiving a valid ACK packet from the device.

Figure 6-10: Control Transfer Protocol — Two-Stage Transfer



## **Chapter 6: Control Protocol**

---

---

### **Three Stage IN Control Transfer Examples**

This section includes four examples of three-stage control IN transfers.

- Get Device Descriptor Request
- Get String Descriptor Request — Data size unknown with short packet
- Get String Descriptor Request — Data size unknown no short packet detected

---

### **The Get Device Descriptor Request Example**

This request returns the contents of the Device Descriptor from the target device. Low-level protocol details are described for this example.

#### **Setup Transaction — Get Device Descriptor**

Figure 6-11 illustrates the contents of the eight bytes of request data during the GetDescriptor request. The last column specifies the content returned in the Data stage. Each field is filled with the appropriate data content corresponding to a device descriptor request. The value field contains a 01h in the upper byte that defines the descriptor as a Device descriptor. The amount of data to be transferred in the data stage is specified by the length field. In this example, the lower byte contains a 12h (18 bytes decimal).

*Figure 6-11: Get Device Descriptor Request Data*

						MSB
Request Type	Request (1 byte)	Value (2 bytes)	Index (2 bytes)	Length (2 bytes)	Data Stage	
10000000B	GET_DESCRIPTOR (06h)	Descriptor Type (0001h)	Zero	Length (1200h)	Descriptor Data	

# **USB 3.0 Technology**

---

## **Data and Status Stages — Get Device Descriptor**

The device descriptor request defined in the Setup stage of the control transfer requested all 18 bytes of the device descriptor. The host controller simply performs an IN transaction to fetch the 18 bytes of data from EP0. Once the DATA packet has been sent and received, the host controller returns an ACK header, thereby completing the data stage.

The Status stage consists confirms from the device perspective that the transfer of the descriptor to the host controller has completed as expected.

## **Protocol Details — Get Device Descriptor Request**

Figure 6-12 on page 121 illustrates the protocol associated with the Get Device Descriptor Request described previously. The following discussion enumerates each step in the control transfer protocol.

### **Setup Stage**

1. The Setup Stage of a Control transfer always uses a Data packet with a sequence number of zero and must have DW1, bit 15 set indicating this packet defines a Setup Token packet.
2. Upon receiving the 8-byte Data packet, the target device must always return an ACK packet with Seq=1 and NumP=1.

### **Data Stage**

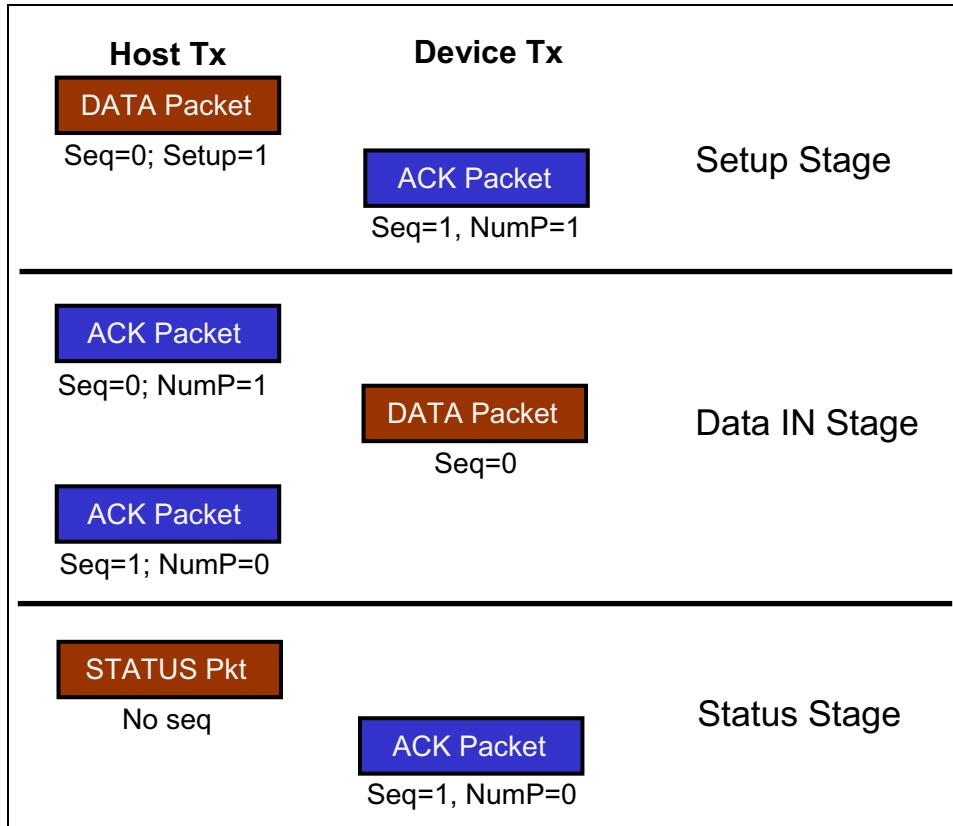
1. The Data Stage performs an IN transaction to fetch the contents of the Device descriptor (18 bytes). The ACK header must specify Seq=0 for the first transaction of any Data stage. Note also that the ACK packet requests a single data packet (NumP=1) because burst transactions are not supported by control transfers. The maximum data payload size of 512 bytes easily handles the Device descriptor content.
2. The Data packet must have a Sequence number of zero and it returns the requested 18-bytes from the device descriptor.
3. The host controller receives the data without error and confirms the acknowledgement by returning Seq=1. Also, the ACK packet returns NumP=0 indicating the end of the Data stage transfer.

## Chapter 6: Control Protocol

Status Stage:

1. The host controller sends a Status header to initiate the Status stage. Notice that the Status header does not contain a sequence number.
2. Endpoint zero confirms completion of the control transfer by returning an ACK packet to the host controller. The ACK packet must report a Sequence number of one and NumP=0.

*Figure 6-12: Three-Stage Get Device Descriptor Protocol*



# USB 3.0 Technology

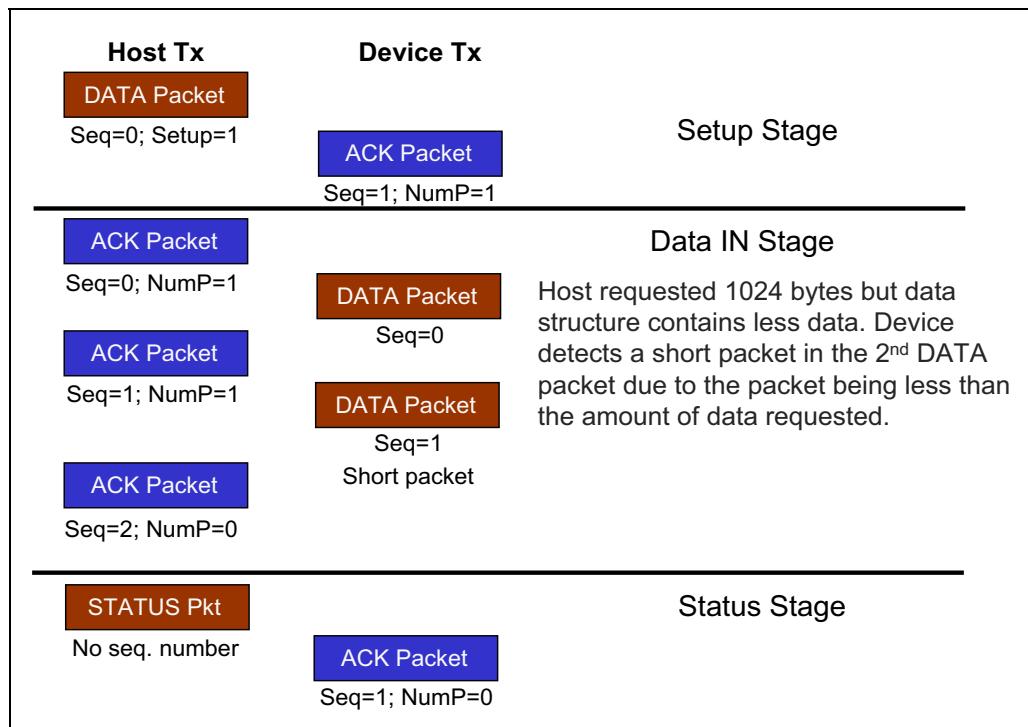
## Control Transfer — Variable-Length Data Example 1

Some applications may have a block of data to read but don't know the exact size of the data structure being read.

The length field in the Setup data specifies a transfer size of 1024 bytes and knows that the data will not exceed that length.

During the Data Stage, the first DATA packet requested by the host controller transfers the maximum payload size of 512 bytes. The next DATA packet is prepared when the data transfer ends. In response the device sends the available data to the host controller, which is detected as a short packet. This causes the control transfer to end. The associated protocols are illustrated in Figure 6-13.

Figure 6-13: Variable Length Data During Control Read Operation (with Short Packet)



## Chapter 6: Control Protocol

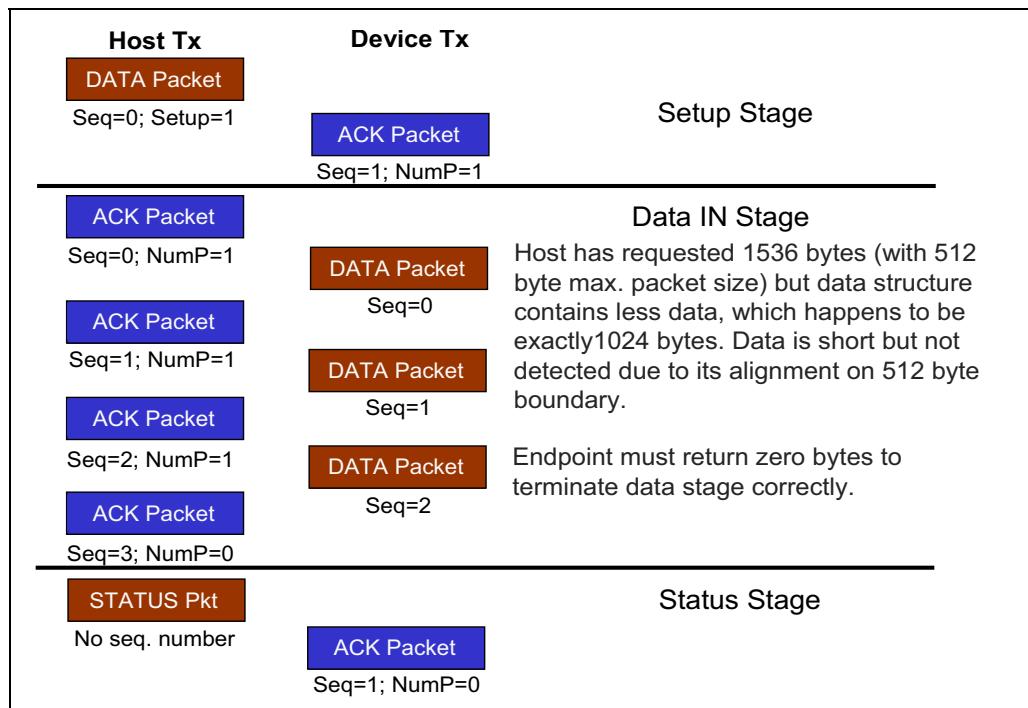
### Control Transfer — Variable-Length Data Example 2

Refer to Figure 6-14 during the following discussion. This example describes the protocol associated with software performing a Control Read transfer request without knowing the exact amount of data to be read. The transfer length is set to 1536 bytes but the actual data size is 1024 bytes. The DATA IN Stage of the transaction proceeds with the host controller sending back-to-back IN transaction requests.

1. The target device returns a 512 bytes DATA packet with Seq=0.
2. The second IN request results in another 512 byte DATA packet with Seq=1
3. Target device returns a 512 byte DATA packet that ends the data transfer, but the host doesn't know it.

Host controller receives the second DATA packet, the host controller doesn't know that the end of data has been reached, so it sends another ACK request. Upon receiving the ACK packet, the target endpoint returns a DATA packet with a payload of zero bytes.

Figure 6-14: Variable Length Data During Control Read Operation (No Short Packet)



## **USB 3.0 Technology**

---

---

# 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

## **USB 3.0 Technology**

---

- 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

# Chapter 7: Bulk Protocol

---

## 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

## **USB 3.0 Technology**

---

The following items define characteristics and behaviors associated with IN Data Bursting.

- DATA packets can be sent continuously as long as the Device can send them and the host can receive them.
- Each ACK header packet includes a Sequence Number (5-bit field) to indicate the next expected DATA packet.
- During endpoint configuration a **SetConfiguration** request must be performed. This forces the sequence number to an initialized value of zero. Consequently the host expects the first IN DATA packet from a bulk endpoint to have a DATA packet sequence number of zero. Note: the **SetInterface** and **ClearStall** requests also clear the sequence numbers to zero.
- DATA packet sequence numbers increment from 0-31 and roll back to zero.
- The maximum number of DATA packets that can be sent, but not acknowledged, is limited to the maximum burst size supported by the endpoint.
- All DATA packets must be maximum-sized payloads during a burst. Only the last DATA packet of a burst transfer can be less than maximum size.
- The NumP (Number of Packets) field reports the number of packet buffers available at the host for receiving DATA packets or the number of packets remaining in the transfer.
- The NumP value may be incremented by any value up to the maximum payload size (MaxPacketSize) for a given bulk endpoint.
- A NumP value of zero signifies the end of a burst transfer.
- NumP may decrement by no greater than one count for each subsequent ACK packet sent by the host. The only exceptions are:
  - The target device sets the End of Burst (EOB) bit in the DATA packet being returned. This indicates that the device temporarily cannot send data. The host must respond by returning an ACK packet with NumP set to zero.
  - The target device sends a DATA packet with a payload less than the maximum size. Upon receiving a short packet the host controller returns NumP=0. Following a short packet, if the host has additional data to fetch it may return a NumP value greater than zero.

Following are three example burst transactions to help clarify the protocol requirements of IN burst transfers. These examples include:

- Single DATA packet burst
- Four DATA packet burst
- Long DATA packet burst

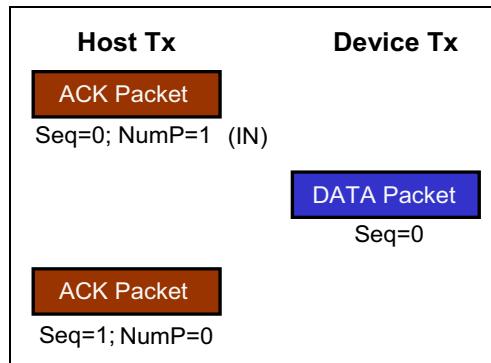
## Chapter 7: Bulk Protocol

---

**Single DATA Packet Burst Example.** An endpoint may be able to burst only one packet at a time. Figure 7-1 on page 129 illustrates the protocol used when an endpoint only sends a single DATA packet. The essential aspects of the protocol are enumerated below:

1. The host sends an ACK header that includes the IN Token information. Two additional fields are important in managing the protocol:
  - Seq=0 specifies the sequence number of the DATA packet to be returned to the host.
  - NumP=1 indicates a host request for a single DATA packet.
2. The target device receives the ACK header (IN transaction) and returns the DATA packet (header and payload) to the host, which includes the requested sequence number of zero (Seq=0).
3. Next, the host receives the DATA packet without error. CRC checks are made within the DATA header and Data Packet Payload. The host also verifies that the header sequence number is in the correct order.
4. The host increments the sequence number to Seq=1 and returns an ACK header. In addition, the NumP value is decremented by one (NumP=0) indicating end of transfer.
5. Upon receipt of the ACK header, the target device verifies the DATA packet was received without error because the sequence number returned from the host is one greater than the DATA packet previously sent. NumP=0 also tells the target device that the transfer has ended. The next transfer targeting this endpoint starts with Seq=1.

Figure 7-1: Bulk IN Transaction — Burst of One DATA packet



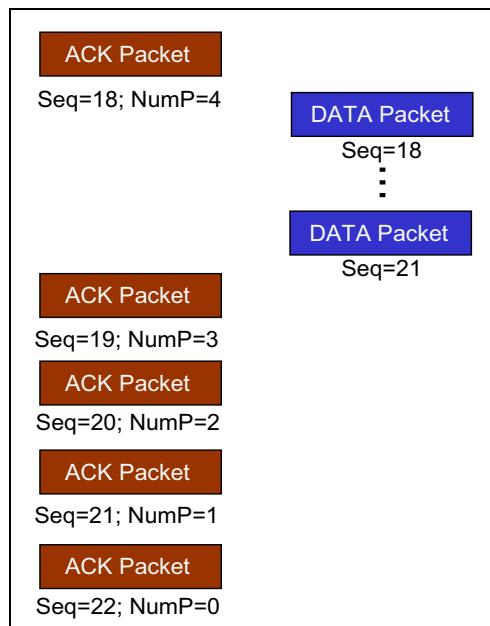
## **USB 3.0 Technology**

---

**Burst IN Example with Four DATA Packets.** The host in this example and shown in Figure 7-2 on page 130, needs to fetch four DATA packets of information. The burst of four packets indicates that the target device supports a Maximum Burst Size of at least 4 DATA packets, and the host has knowledge of this capability. Notice the ACK packet specifies a DATA packet sequence number of 18 and a NumP value of four.

Upon receiving the first ACK packet the target device responds by returning a burst of four DATA packets with sequence numbers of 18-21. The host receives each DATA packet and ultimately returns ACK packets for each DATA packet received. Notice that with each subsequent ACK header packet the NumP value decrements by one. This prevents the target device from sending additional DATA packets. For example, the second ACK header specifies Seq=19, which acknowledges receipt of DATA packet 18, leaving three packets (Seq numbers 19-21) yet to be acknowledged. NumP=3 indicates that the host is requesting three DATA packets, which in this example have already been sent by the device. The host ultimately acknowledges the remaining three packets and decrements the NumP value. Thus, all four packets are acknowledged and the NumP goes to zero.

*Figure 7-2: Bulk IN Transfer — Burst of Four DATA Packets*



## Chapter 7: Bulk Protocol

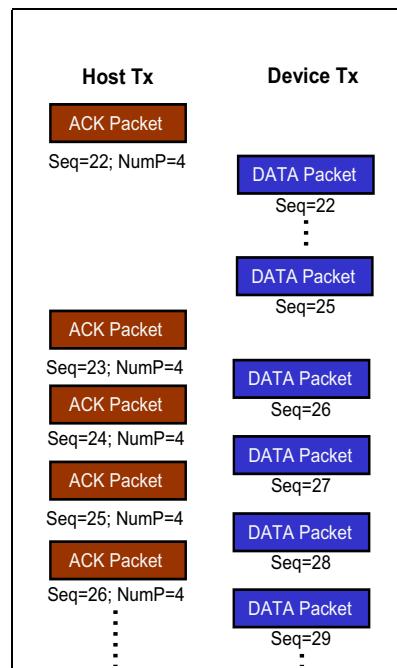
---

**Burst IN Example with Long Burst.** This example shows how burst transactions can continue as long as the host requests additional data and as long as the target device can return data. Refer to Figure 7-3 on page 131 during the following discussion.

This transaction is a continuation of the previous example. The transaction begins with the host sending an ACK header packet advertising Seq=22 and NumP=4. The target device receives the ACK and delivers a burst of four DATA packets (Seq 22-25). Once again the Max Payload Size is 4, so the device waits for an acknowledgement before sending more DATA packets.

The first ACK header acknowledges receipt of DATA packet 22 by returning Seq=23 and also advertises NumP=4. Currently, three DATA packets are still awaiting acknowledgement and the host has requested four packets (NumP=4). Therefore, the device can now send an additional DATA packet (Seq=26) so that once again there are four packets not yet acknowledged. In short, the number of DATA packets not yet acknowledged can never be greater than the Max burst size supported. In the example, NumP is always four so each DATA packet acknowledged allows delivery of the next Data packet and the burst transfer continues.

Figure 7-3: IN Burst Transaction with Large Transfer



# **USB 3.0 Technology**

---

## **End-to-End Flow Control — Bulk IN Transfers**

When a USB SS device cannot return data there are actions that a device takes to flow control the transfer. The flow control mechanism temporarily causes the host to suspend the transfer, and when the device is ready to send data again it notifies the host to resume the transfer.

Three header packets are used in the End-to-End flow control mechanism:

- NRDY (Not Ready) header — device returns NRDY to notify the host controller that it cannot return the requested data at this time.
- DATA with eob (end of burst) set — device returns a DATA packet and also sets the eob bit to notify the host that it cannot return the next DATA packet at this time.
- ERDY (Endpoint Ready) — device notifies the host controller that it is now ready to send data. This triggers the host to resume the transactions where it left off previously.

Three examples follow, each of which describe a different flow control scenario.

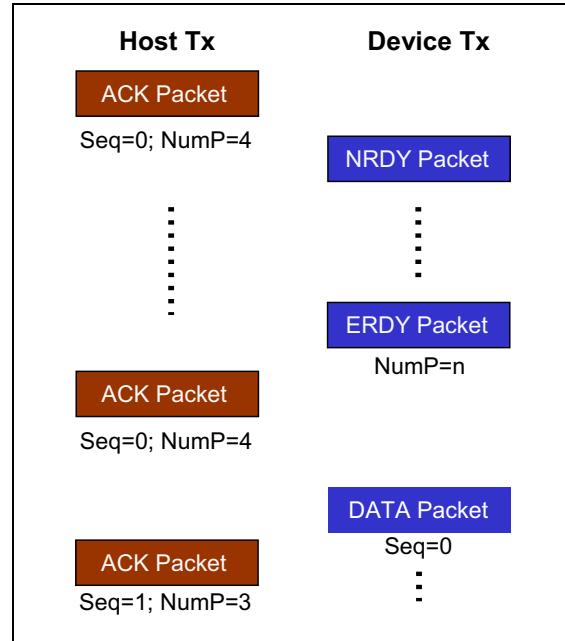
**NRDY Flow Control at Start of IN Transfer.** Figure 7-4 on page 133 depicts a flow control sequence at the beginning of the burst transactions. In this example the host's request for data cannot be fulfilled within the 250 ns timeout period (See Table 7-2 on page 146) so the device has no choice other than to flow control the transfer using NRDY. The sequence of events is detailed below.

1. Target device receives IN request from host and must respond within the specified time. Because the DATA packet is not ready for delivery, the device returns the NRDY flow control packet.
2. Upon receiving NRDY the host controller deactivates the transaction and waits for the device to request reactivation of the transaction.
3. The device sends ERDY to notify the host that it is ready to send data. The NumP field within the ERDY header reports the number of DATA packets that are ready for transmission, up to the MaxBurstSize.
4. When the host controller receives the ERDY header it restarts the transaction by re-sending the ACK packet. No timing parameter is defined for the host response to ERDY because bulk transfers have no guaranteed latency.

## Chapter 7: Bulk Protocol

---

Figure 7-4: NRDY Flow Control at Start of Transfer



**IN Burst Flow Control Using EOB.** Devices may encounter conditions that prevent burst transactions from continuing. There are two primary scenarios that can cause this condition:

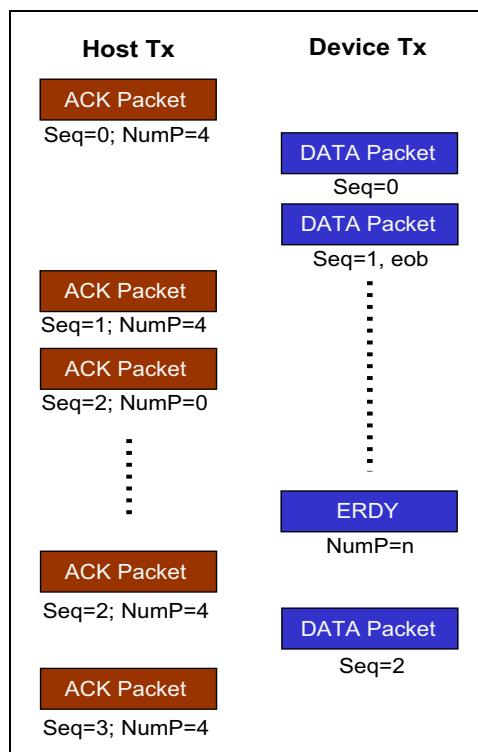
- The host may request a block of data (up to the MaxBurstSize of the device) be returned when it sends the ACK packet (e.g., NumP=4). If the device can only return a portion of the requested data, it must set the **eob** bit in the last DATA packet it can supply.
- A device may recognize that a high priority event must be handled during a burst transfer. This may prevent the device from sending a data packet within the 100ns MaxBurstInterval. Consequently, the device can set **eob** in the last DATA packet prior to the expected delay.

Figure 7-5 on page 134 illustrates the burst flow control protocols using eob. In this example, the host initially requests four DATA packets and the device responds with two DATA packets with eob set in the 2nd packet. The host responds to the 2nd DATA packet with an ACK packet that verifies successful receipt of the packet with Seq=2 and sets NumP=0 to verify the end of burst. The host places the transfer into the inactive state and waits for the device to signal ERDY.

## USB 3.0 Technology

The device eventually sends ERDY and indicates the number of DATA packets ready to be sent. NumP may be any number from 1 to the MaxBurstSize (4 in this example). Upon receiving ERDY the host activates the burst transfer by repeating the last ACK packet with Seq=2 but sets NumP=4, thereby requesting a burst of four packets.

Figure 7-5: IN Burst Flow Control Using EOB

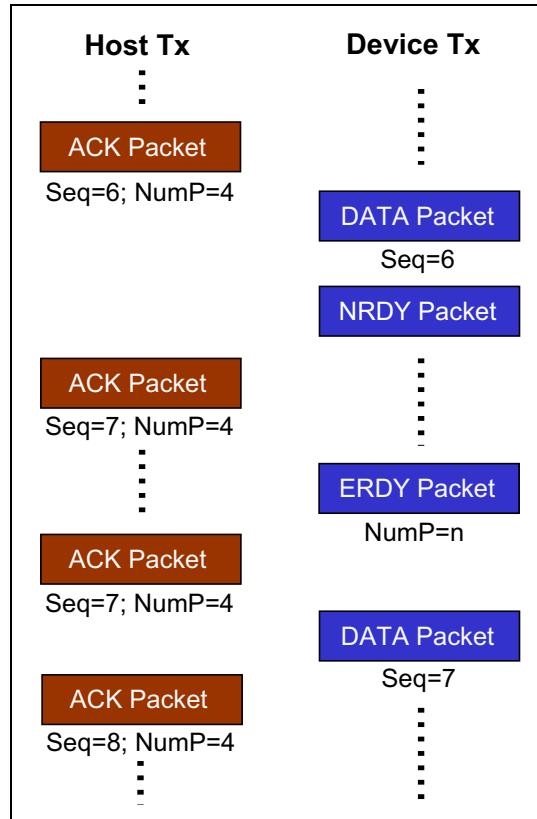


**IN Burst Flow Control Using NRDY.** The example shown in Figure 7-6 is a variation of the previous example. In this case let's assume the device has four DATA packets ready for transmission but a high priority event within the device prevents a second DATA packet from being sent within the MaxBurstInterval timeout (100ns). When this occurs the only option for the device is to send NRDY. Upon receipt of the NRDY header the host delivers an ACK packet with Seq=7 to acknowledge the successful transfer of DATA packet Seq= 6, and then it transitions the burst transfer to the inactive state. When the device is ready to resume the transfer it sends ERDY to the host. When ERDY is received by the host it sends the previous ACK packet again and the burst continues.

## Chapter 7: Bulk Protocol

---

Figure 7-6: IN Burst Flow Control Using NRDY



**Short packets.** During burst transactions a DATA packet may be sent with a data length less than the MaxPacketSize. This is called a **short packet** and always signifies the end of a transfer. Optionally, the device may set the EOB bit in the short DATA packet. When a short packet is sent the following actions take place:

- A device that returns a short packet during an Bulk IN transfer must stop sending DATA packets until another transfer is initiated.
- Upon receiving a short packet the host controller must return an ACK packet with NumP=0, unless there is a DATA packet error. (see next section for DATA errors and Retry).
- If the host receives a short packet and another transfer is pending, the host controller may initiate the next transfer by sending an ACK packet with NumP>0 and =< MaxBurstSize.

## USB 3.0 Technology

---

**Data IN Transfer Errors and Retry.** The Port-to-Port protocol ensures that all protocol layer header packets are successfully sent between a root hub port and device endpoint. However, the Data Packet Payload (DPP) is not handled by the Port-to-Port protocol, thus, the End-to-End protocol must manage DATA packet errors.

Figure 7-7 illustrates a DATA header and DPP and the associated CRC values. As a DATA packet is transferred from link-to-link a variety of CRC checks are made by each hub and ultimately at the root hub port. Several types of errors can lead to DATA transfer errors, including:

- **CRC-32 Errors** — errors detected within the DPP by hubs or the root hub port. These errors result in the DPP being forwarded as a nullified packet. The application can choose to either discard or use the corrupted data.
- **DPP Framing Errors** — these errors (DPPSTART and DPPEND) prevent the DPP from being properly recognized, causing the DPP to be dropped, leaving only the DATA header to be forwarded to the destination.
- **DATA Header Errors** — these errors result in the Port-to-Port protocol retrying the transaction until it successfully transfers. This also results in the DPP being dropped.

Figure 7-7: DATA Header and Payload

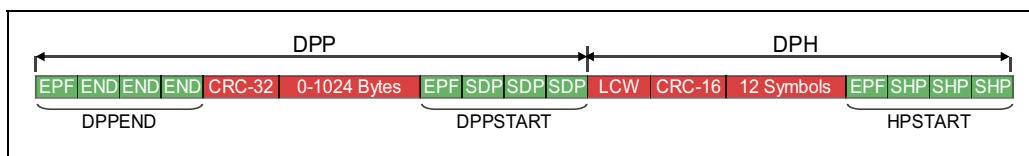


Figure 7-8 illustrates the End-to-End protocol solution for recognizing and retrying DATA packet errors. The sequence of events is as follows:

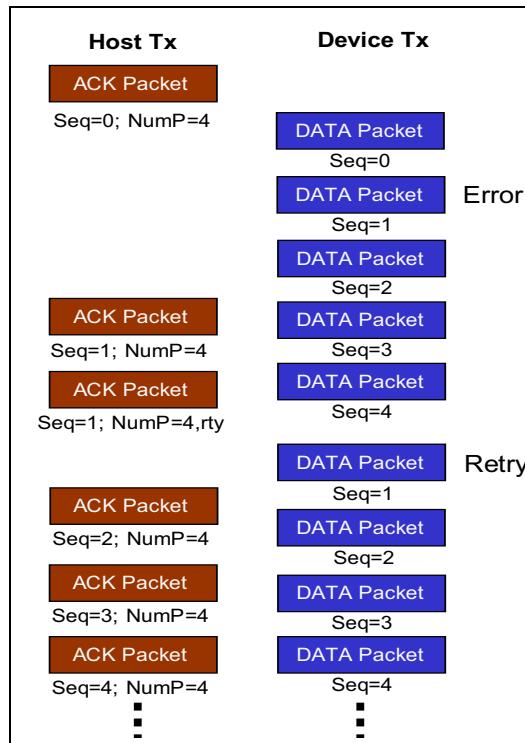
1. The ACK packet requests a burst of four DATA packets starting with Seq=0
2. In response, the device sends a burst of four Data packets.
3. DATA packet zero is acknowledged by the host via the 2nd ACK packet with Seq=1 and NumP=4.
4. The target device recognizes a request for 4 packets and sends DATA packet 4, leaving four DATA packets (1-4) still unacknowledged.

## Chapter 7: Bulk Protocol

---

5. Upon receiving DATA packet 1 the host controller detects an error in the DPP and discards packets 1-4.
6. The host returns an ACK packet with Seq=1, NumP=4 and Retry=1 to request the target re-send DATA packets 1-4.
7. The device bursts DATA packets 1-4 and burst of transactions continue.

Figure 7-8: Example Bulk IN Burst with DATA Packet Error



---

### Bulk OUT SuperSpeed Transaction Protocol

The host issues DATA packets to initiate an OUT token transaction. OUT transaction protocol involves the following areas of focus:

- OUT Data bursting
- End-to-End flow control
- Data transfer errors and retry
- Timeout conditions and values

## **USB 3.0 Technology**

---

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

### **OUT Data Bursting**

The following items define characteristics and behaviors associated with OUT Data Bursting.

- DATA packets can be sent continuously as long as the host can send them and the device can receive them.
- Each DATA header includes a Sequence Number (5-bit field) indicating the DATA packet being sent. The Data Packet Payload immediately follows the header.
- The maximum number of DATA packets that can be sent by the host without having been acknowledged is limited to the maximum burst size (MaxBurstSize) supported by the device endpoint.
- During a burst transfer All DATA packets must have maximum-sized payloads, except the last DATA packet of a burst transfer.
- The NumP field returned by the device reports the number of buffers available at the device for receiving DATA packets.
- The NumP value may be incremented by any value up to the maximum payload size (MaxPacketSize).
- NumP may decrement by no greater than one count for each subsequent ACK packet sent by the device.

Following are three example burst transactions to help clarify the protocol requirements of OUT burst transactions. These examples include:

- Single DATA packet burst
- Four DATA packet burst
- Long DATA packet burst

**Single DATA OUT Burst Example.** An endpoint may be able to accept only one packet at a time. Figure 7-9 illustrates the protocol used when the host sends a single DATA packet. The essential aspects of the protocol are enumerated below:

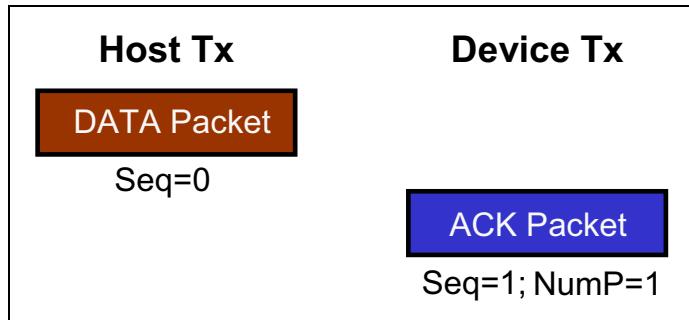
1. The host sends one DATA header knowing the device supports reception of a single DATA packet. The DATA header includes the OUT Token information and the sequence number of the packet (zero in this example).
2. The target device receives the DATA packet (header and payload) and verifies there are no errors. This includes CRC checks within both the DATA header and the Data Packet Payload.

## Chapter 7: Bulk Protocol

---

3. The device then returns an ACK header with a sequence number that is one greater than the sequence number received in the DATA packet, thereby acknowledging successful packet transfer. The ACK packet also includes NumP=1 indicating a single packet was received. Note that NumP=0 is used for burst flow control during OUT transactions.
4. Upon receipt of the ACK header, the host verifies the packet was received and terminates the transaction.

Figure 7-9: Bulk OUT Transaction — Burst of One DATA packet



**Burst OUT Example with Four DATA Packets.** The host in this example needs to send four DATA packets. The burst of four packets indicates that the target device supports a MaxBurstSize of at least 4 DATA packets, and the host has knowledge of this capability. Notice that the first DATA packet has a sequence number of 29. Refer to Figure 7-10 on page 140 during the following discussion.

The sequence of events is as follows:

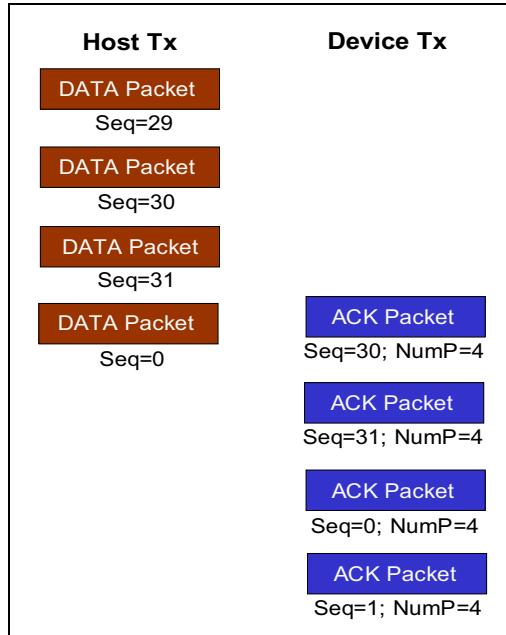
1. The host controller initiates the burst transfer by sending a burst of four DATA packets (header and payload).
2. The device receives each DATA packet and ultimately returns an acknowledgement for each packet received. Notice that with each subsequent ACK header the NumP value stays at four. The device processes the DATA packets very efficiently or implements a buffer larger than four entries.
3. The device sends the ACK header with Seq=30, which acknowledges receipt of DATA packet 29. The device also sets NumP=4 giving the host permission to send another DATA packet, but the host does not initiate another packet knowing that it only requires a 4 KB transfer.

## **USB 3.0 Technology**

---

4. During the next three transactions the device ACKs each DATA packet by incrementing the sequence number and continuing to specify NumP=4. Notice that the sequence number in the third ACK packet has rolled over from Seq=31 to Seq=0, thereby acknowledging successful transmission of DATA packet Seq=31.
5. Upon successfully completing the fourth DATA packet transmission, the host controller terminates the transfer.

*Figure 7-10: Bulk OUT Burst of Four DATA Packets*



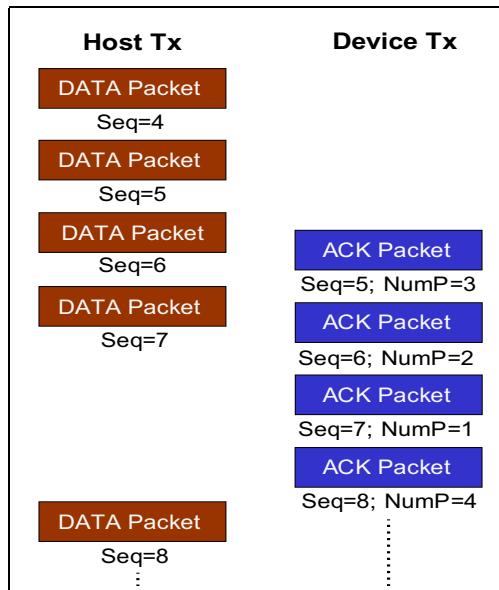
**Burst OUT Example with Long Burst.** This example shows how burst transactions can continue as long as the host sends additional data and as long as the target device has buffer space to receive data. Also, the example shows possible behavior of NumP being incremented and decremented. Refer to Figure 7-11 during the following discussion. This transaction is a continuation of the previous example with Seq=4 being the next DATA packet.

## Chapter 7: Bulk Protocol

The sequence is as follows:

1. Host bursts four DATA packets knowing MaxBurstSize=4.
2. Device acknowledges receipt of DATA packet 4 by returning ACK with Seq=5 along with NumP=3, reporting that 3 of the 4 buffers are available.
3. Host cannot send next packet because the device has reported 3 buffers available and 3 packets are still currently outstanding (not yet ACKed).
4. Device sends 2nd ACK with Seq=6 and NumP=2 (2 of 4 buffers avail.).
5. Host cannot send next packet because 2 packets are still outstanding.
6. Device sends 3rd ACK with Seq=7 and NumP=1 (1 of 4 buffers avail.).
7. Host cannot send next packet because 1 packet still outstanding.
8. Device sends 4th ACK with Seq=8 and NumP=4 (4 of 4 buffers avail.).
9. Host can now burst 4 more DATA packets and continue the burst.

Figure 7-11: OUT Burst with Large Transfer



### End-to-End Flow Control — Bulk OUT Transfers

When a USB SS device cannot accept data there are actions that the device takes to flow control the transfer. The flow control mechanism temporarily causes the host to deactivate the burst of transactions, and when the device is ready to accept data again it notifies the host to resume the transfer.

## **USB 3.0 Technology**

---

There are three header packets that can be used by the OUT End-to-End flow control mechanism:

- NRDY (Not Ready) header — device returns NRDY to notify the host controller that it cannot return the requested data at this time.
- NumP=0 (Number of Packet buffers available) — device returns NumP=0 in the ACK packet to indicate that it has no buffer space available temporarily, to accept data. This forces an end-of-burst event.
- ERDY (Endpoint Ready) — device notifies the host controller that it is now ready to send data. This causes the host to resume the transactions where it left off previously.

Two examples follow, each of which describe a different flow control scenario.

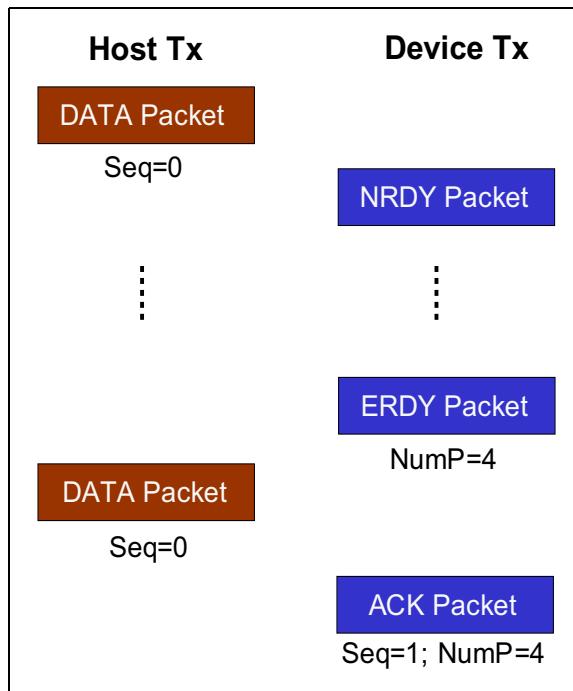
**NRDY Flow Control at Start of OUT Transfer.** Figure 7-12 on page 143 depicts a flow control sequence at the beginning of the burst transactions. In this example the host's DATA packet cannot be accepted so the device has no choice other than to flow control the transfer using NRDY. The sequence of events is detailed below.

1. Target device receives DATA packet with OUT token from host and cannot receive the data due to a buffer full condition or a busy condition within the device. The device returns the NRDY flow control packet to notify the host of the condition.
2. Upon receiving NRDY the host controller deactivates the transaction and waits for the device to request reactivation of the transaction.
3. The device sends ERDY to notify the host that it is ready to receive data. The NumP field within the ERDY header reports the number of buffers available for receiving DATA packets.
4. When the host controller receives the ERDY header it restarts the transaction by re-sending the DATA packet. No timing parameter is defined for the host response to ERDY because bulk transfers have no guaranteed latency.

## Chapter 7: Bulk Protocol

---

Figure 7-12: NRDY Flow Control at Start of Transfer



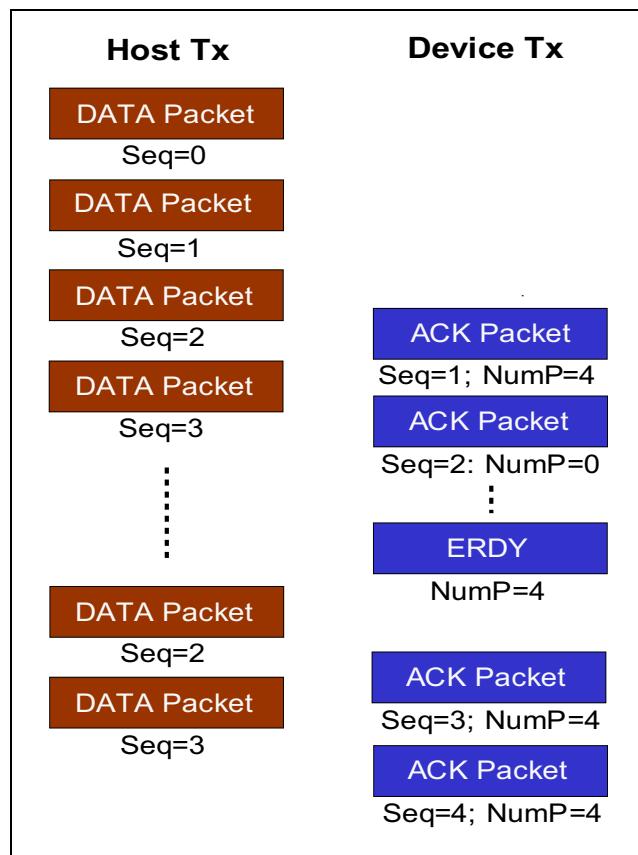
**OUT Burst Flow Control Using NumP=0.** Devices may encounter conditions that prevent OUT burst transactions from continuing. For example, the host may send a block of data (up to the MaxBurstSize of the device). If the device can only accept a portion of the requested data, it must set NumP=0 in the ACK packet associated with the last DATA packet it can accept. Figure 7-13 on page 144 illustrates the burst flow control protocol using NumP=0.

1. Host bursts four DATA packets (sequence numbers 0-3) to the device.
2. Device accepts the first DATA packet and returns ACK to the host with Seq=1 and NumP=4
3. The target device responds to the 2nd DATA packet with an ACK packet that verifies successful receipt of the packet with Seq=2 and sets NumP=0 to signal the end of burst.
4. The host places the burst transfer into the inactive state and waits for the device to signal ERDY.
5. When DATA packets 2 and 3 arrive at the device, it simply discards these packets.

## USB 3.0 Technology

6. The device eventually sends ERDY and indicates the number of buffers it has available to receive data (NumP=4).
7. Upon receiving ERDY the host activates the burst transfer by sending DATA packets at the continuation point (DATA packet Seq=2).

Figure 7-13: OUT Transaction Burst Flow Control Example



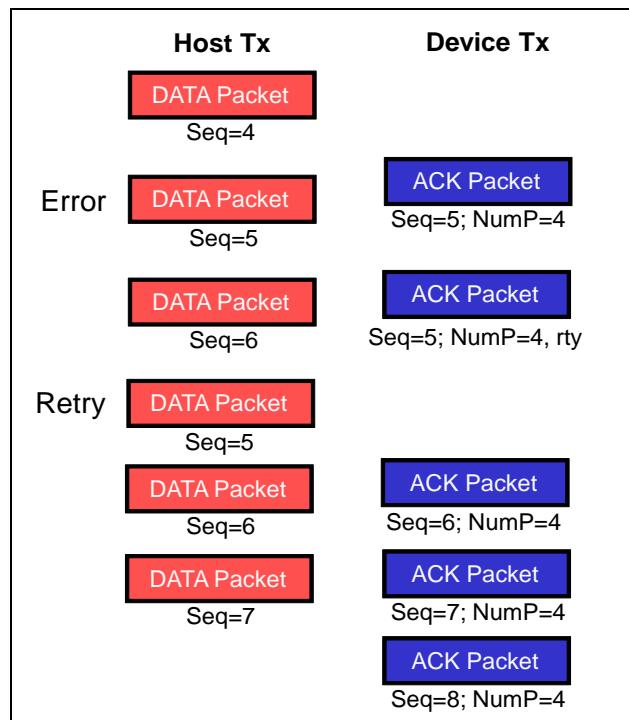
**DATA OUT Transfer Errors.** Figure 7-8 illustrates the End-to-End protocol solution for recognizing and retrying OUT DATA packet errors. The sequence of events is as follows:

1. The host starts a burst of 4 DATA packets that start with Seq=4.
2. The device upon detecting the DATA packet returns an ACK packet with Seq=5, which acknowledges success receipt of DATA packet Seq=4.

## Chapter 7: Bulk Protocol

3. Next, DATA packet Seq=5 is received by the device but an error is detected. In response, the device returns an ACK packet with Seq=5, NumP=4 and retry (rty), thereby requesting that DATA packet 5 be retried.
4. DATA packet Seq=6 was already in flight when the ACK retry was signaled, so the device discards DATA packet 6, which must also be retried.
5. Next, the host performs the retry of DATA packet 5 along with DATA packets 6 and 7, and the device acknowledges each packet.
6. The host has no more data to send to the transaction ends.

Figure 7-14: Example Bulk OUT Burst with DATA Packet Error



## **USB 3.0 Technology**

---

### **Timeout conditions and values**

Table 7-2 lists the timing parameters related to IN and OUT transactions.

*Table 7-2: Timing Parameters for IN and OUT Transaction Protocol*

Parameter	Description	Value
tNRDY	Maximum time between a device receiving the last symbol of an ACK or DATA packet and first framing symbol of the NRDY packet response	250 ns
tDPResponse	Maximum time between a device receiving the last symbol of an ACK packet and first framing symbol of the DATA packet response	250 ns
tACKResponse	Maximum time between a device receiving the last symbol of a DATA packet and first framing symbol of the ACK packet response	250 ns
tHostACKResponse	Maximum time between a host receiving the last framing symbol of a DATA packet and first framing symbol of the ACK packet response	3 $\mu$ s
tMaxBurstInterval	Maximum time between a device or host sending the last symbol of a DATA packet and first framing symbol of the next DATA packet during a burst transfer	100 ns

---

### **Standard Bulk Vs. Bulk Streaming Endpoints**

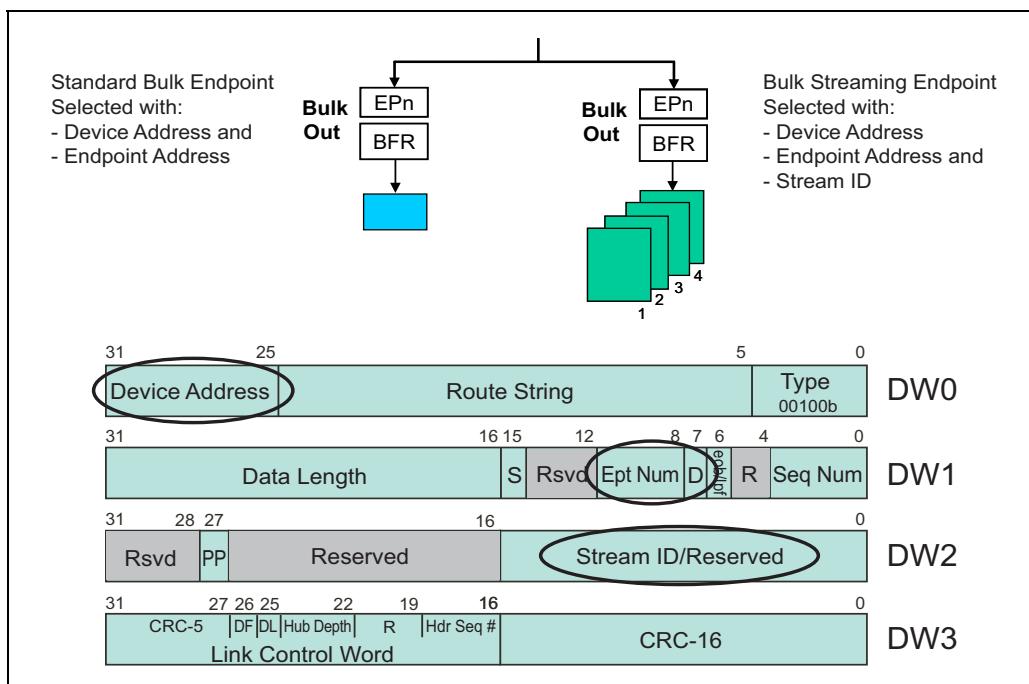
While the End-to-End protocol rules just described apply to all SuperSpeed bulk endpoints, if a bulk endpoint supports the streaming, there is a significant difference in how it is managed and the performance of applications designed to take advantage of the streaming capability. The remainder of this chapter describes differences between bulk streaming and standard (non-streaming) bulk endpoints while presenting application examples for each.

## Chapter 7: Bulk Protocol

### Streaming And Endpoint Buffers

At the upper left of Figure 7-15 is a standard bulk (non-streaming) endpoint and the buffer (BFR) supporting it. At the upper right is bulk streaming endpoint and the buffers supporting it (four in this example). At the lower part of the illustration is a packet header highlighting key fields used for targeting endpoints and buffers. Refer to this illustration in the discussion that follows.

Figure 7-15: Standard And Bulk Streaming Endpoints



### Standard Bulk Endpoint Buffer

As indicated at the left in Figure 7-15 on page 147, a standard bulk endpoint has a single buffer that is large enough to accommodate the maximum packet size reported by the endpoint. If data packet bursting is supported, the size of the buffer is the maximum packet size multiplied by the burst size (up to 16 KB for a standard bulk endpoint buffer).

## **USB 3.0 Technology**

---

The fact that there is a single buffer behind the endpoint has two implications:

- Packets targeting the endpoint only have one possible destination. In the packet header, the only important token address information is Device Address and Endpoint Address (Endpoint Number plus Direction). The Stream ID (SID) field is reserved.
- Packets moving through the endpoint must be handled one at a time. If there is a flow control condition, all endpoint packet traffic is paused. ERDY and NRDY packets sent by the device apply to the endpoint as a whole.

---

### **Bulk Streaming Endpoint Buffers**

As indicated at the right in Figure 7-15 on page 147, a bulk streaming endpoint has multiple buffers; each is large enough to accommodate the maximum packet size multiplied by the burst size reported by the endpoint (up to 16KB per streaming buffer).

The fact that there are multiple buffers (up to 64K) behind the endpoint has several implications:

- Packets targeting the endpoint have as many possible destinations as there are streaming buffers. Referring again to the packet header depicted in Figure 7-15 on page 147, the important token address information now includes: Device Address, Endpoint Address (Endpoint Number plus Direction), and the Stream ID (SID) field.
- If there is a flow control condition at the buffer for one stream, other streams may still be actively moving packets. ERDY and NRDY packets sent by the device include Stream ID (SID) and apply on a stream-by-stream basis.
- Expanding the number of buffers behind a bulk streaming endpoint and providing independent flow control for each stream (based on Stream ID) is an enabling feature for the command queueing and out-of-order data packet processing now possible with USB 3.0 SuperSpeed bulk devices.

---

### **Stream ID (SID), A Bit More**

The Stream ID (SID) field within headers may either be reserved (zeros) or contain SID information. Only four Header packets use the SID field as shown below. Table 7-3 defines the content of the SID field within the header packets.

## Chapter 7: Bulk Protocol

---

- ACK
- DATA
- NRDY
- ERDY

Table 7-3: Stream ID Values and Descriptions

Values	Description
0000h	Reserved — used by standard Bulk endpoints (non-streaming)
0001h - FFFDh	Stream n — Available SID values for use during stream operations
FFFEh	Prime — used to initialize the streaming state machines
FFFFh	NoStream — indicates this endpoint supports streaming, but that no stream is associated with this particular packet.

---

### Priming the Bulk Streaming Endpoints

As indicated in Table 7-3, the Stream ID (SID) field in ACK, DATA, NRDY, and ERDY header packets normally will be:

- 0000h for packets targeting standard (non-streaming) bulk or endpoints that are not bulk.
- 0001h-FFFCh for packets targeting bulk streaming endpoints

The second to the last entry in Table 7-3 is FFFEh. It is used after a bulk endpoint is discovered to have streaming capabilities, but before it is used. Prior to issuing a command, each streaming endpoint must be primed so that it is ready for the initial access. Priming involves initiating a transaction to the endpoint with FFFEh (prime) in the SID field. Because no operation is pending within the device when priming occurs, the endpoint returns NRDY with FFFEh in the SID field. When the device endpoint is ready to perform streaming transfers, it issues ERDY.

## **USB 3.0 Technology**

---

### **Reporting Bulk Endpoint Streaming Capability**

Bulk endpoints report streaming capability in the SuperSpeed Endpoint Companion descriptor. Table 7-4 on page 150 shows the first four fields of the Endpoint Companion descriptor. Streaming support is reported in bits 4:0 of the Attributes bitmap at Offset 3. As indicated Table 7-4, a zero in the MaxStreams field indicates no stream support; a non-zero value indicates the maximum number of streams (as a power of 2). It is permitted (and very common) for devices with bulk streaming endpoints to also implement standard (non-streaming) bulk endpoints.

*Table 7-4: Endpoint Companion Descriptor Streaming Support*

Offset	Field	Size	Value	Description
0	Length	1	06h	Size of this descriptor is 6 bytes.
1	Descriptor-Type	1	30h	SuperSpeed Endpoint Companion descriptor type is 48.
2	MaxBurst	1	Number	Zero-based Values can range from 0-15 and specify the maximum size of the burst (1-16). (Not valid for Control endpoints)
3	Attributes	1	Bitmap	<b>Bulk Endpoints:</b> Bits 4:0    MaxStreams Values 0h = Streams not supported 1-Fh = 2 Bits 7:5    Reserved (zeros)

---

### **Standard And Bulk Streaming Examples**

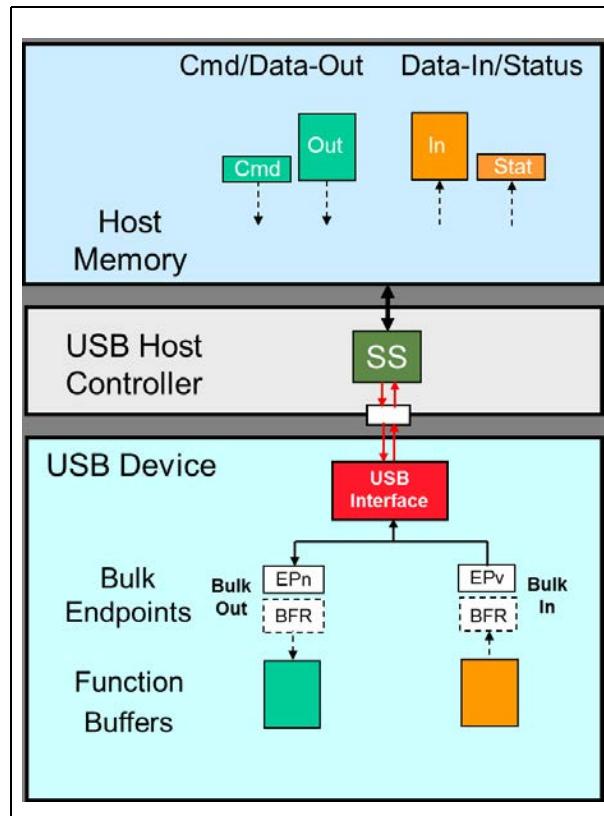
Many USB 3.0 SuperSpeed applications that rely on bulk endpoints are associated with mass storage. The two examples that follow illustrate some of the differences between managing mass storage devices based on standard (non-streaming) bulk endpoints and bulk streaming endpoints.

## Chapter 7: Bulk Protocol

### Standard Bulk Endpoints & SCSI BOT Drive

This example assumes that a SuperSpeed Bulk Only Transport (BOT) drive is attached to a root hub port of the Host Controller (HC). Figure 7-16 depicts system resources required to support the drive.

Figure 7-16: BOT Drive Endpoints And System Resources



The roles of the system resources required to support a BOT drive are depicted in Figure 7-16 on page 151 and described in the sections that follow. Briefly, the three groups of resources include:

# **USB 3.0 Technology**

---

## ***Standard Device Endpoints***

In addition to the default control endpoint, EP0, a BOT drive requires two standard (non-streaming) bulk endpoints.

- **Bulk OUT (Command & Data) Endpoint** — this standard (non-streaming) Bulk OUT endpoint is the target of USB transactions carrying SCSI commands or DMA Write data to drive. The single endpoint buffer handles both SCSI commands and 1KB data packets (maximum burst = 16).
- **Bulk IN (Data & Status) Endpoint** — this standard (non-streaming) Bulk IN endpoint is the target of USB transactions returning DMA Read data or SCSI status from the drive. The single endpoint buffer handles both SCSI commands and 1KB data packets (maximum burst = 16).

## ***Host Memory Buffers***

This is generally system main memory DRAM. As indicated in Figure 7-16 on page 151, four memory buffers are required to support a BOT drive interface.

- **Command (CMD) Buffer** — a single main memory Command (CMD) buffer is used to enqueue SCSI commands that will ultimately target the device Bulk OUT endpoint. The number of bytes in a command is SCSI-defined (example: 31 bytes).
- **Out Data Buffer** — one main memory Out Data buffer is required. Data in this buffer becomes the payload for USB OUT transactions transporting data are sent by the host controller to the Bulk OUT endpoint.
- **In Data Buffer** — one main memory In Data buffer is required. The host controller fills this buffer with the contents of data packet payloads returned during USB IN transactions transporting data from the Bulk IN endpoint.
- **Status Buffer** — one main memory Status buffer is required. The host controller fills this buffer with the contents of data packet payloads returned during USB IN transactions transporting SCSI status from the Bulk IN endpoint. The number of status bytes is defined by SCSI (example: 13 bytes).

## ***Host Controller (HC)***

The host controller provides the interface between the USB software stack and the downstream devices. In the discussion that follows, the role of the host controller and its software driver are presented in general terms; details on specific host controllers, such as those based on the eXtensible Host Controller Interface--xHCI, are described in separate specifications.

## Chapter 7: Bulk Protocol

---

Key services the host controller provides in supporting a BOT drive include:

- **USB Transaction Generation** — Based on software requests, the host controller fetches Transfer Descriptors for each endpoint and executes the required USB SuperSpeed Bulk IN/OUT transactions.
- **Main Memory Access (DMA)** — As needed, the host controller performs DMA read/write operations to access main memory buffers for the purposes of fetching SCSI commands and outbound data as well as storing inbound data and SCSI status.
- **End-to-End Flow Control** — The host controller manages header packet flow control and device NRDY and ERDY flow control packets. Service is paused and resumed on an endpoint basis.
- **Protocol Layer Error Handling** — In addition to the link level error handling required of all SuperSpeed link partners, the host controller performs all of the Protocol Layer data packet checks and packet acknowledgement.
- **Other Services** — The host controller also notifies the host in the event of device attachment/removal, forwards Device Notifications upstream, etc.

### BOT Disk DMA Read Operation

In this example, software intends to issue a 31-byte SCSI DMA Read command to the BOT drive. The command includes starting location on the disk, amount of data to transfer, the transfer direction, etc. When the transfer completes, the BOT drive is expected to return 13 bytes of SCSI status.

Referring again to Figure 7-16 on page 151, the general steps required to perform this BOT disk DMA Read operation include:

1. **Command (CMD) Buffer** — The 31-byte SCSI DMA Read command is enqueued in the main memory Command (CMD) buffer.
2. **In Data Buffer** — The main memory In Data buffer is allocated. The size should be large enough to store all inbound data packet payloads for the SCSI DMA Read transfer size. Because of paging and scatter-gather, the In Data buffer may actually be made up of multiple smaller buffers.
3. **Status Buffer** — A 13-byte Status buffer is allocated for the status which will be retrieved from the drive once the command completes.
4. Software requests the HC to perform a USB transaction targeting the BOT drive Bulk OUT endpoint. Transfer length is 31 bytes and the HC is provided an address pointer to the main memory Command (CMD) buffer.
5. The HC performs a read of the 31 bytes from SCSI Command buffer and schedules a USB transaction targeting the BOT drive Bulk OUT endpoint.
6. When the SCSI command is successfully delivered to the BOT drive Bulk OUT endpoint, the HC may generate an interrupt to the host.
7. At the drive, the 13-byte SCSI command is decoded and the drive controller

## **USB 3.0 Technology**

---

- starts moving data into the Bulk IN endpoint buffer. ERDY may be sent upstream when at least one data packet is available for transfer.
8. Software requests the HC to perform USB transactions targeting the BOT drive Bulk IN endpoint. Depending on the DMA Read transfer length, the transfer may actually require many USB Bulk IN transactions.
  9. The HC initiates as many USB Bulk IN transactions as needed to satisfy the transfer requirements. As data packets are read from the BOT drive Bulk IN endpoint, HC moves data packet payloads to the main memory In Data buffers allocated for the transfer.
  10. At the completion of all required USB Bulk IN transactions, the HC may generate an interrupt informing software that the transfer is complete.
  11. To retrieve SCSI status at the end of the transfer, software requests the HC to perform a final USB transaction targeting the BOT drive Bulk IN endpoint. Transfer length is 13 bytes and the HC is provided an address pointer to the main memory SCSI Status buffer.
  12. After the USB IN transaction, the HC updates the main memory Status buffer
  13. Host controller may generate another interrupt.
  14. The BOT drive DMA Read is complete.

---

## **Bulk Streaming Endpoints & UAS Drives**

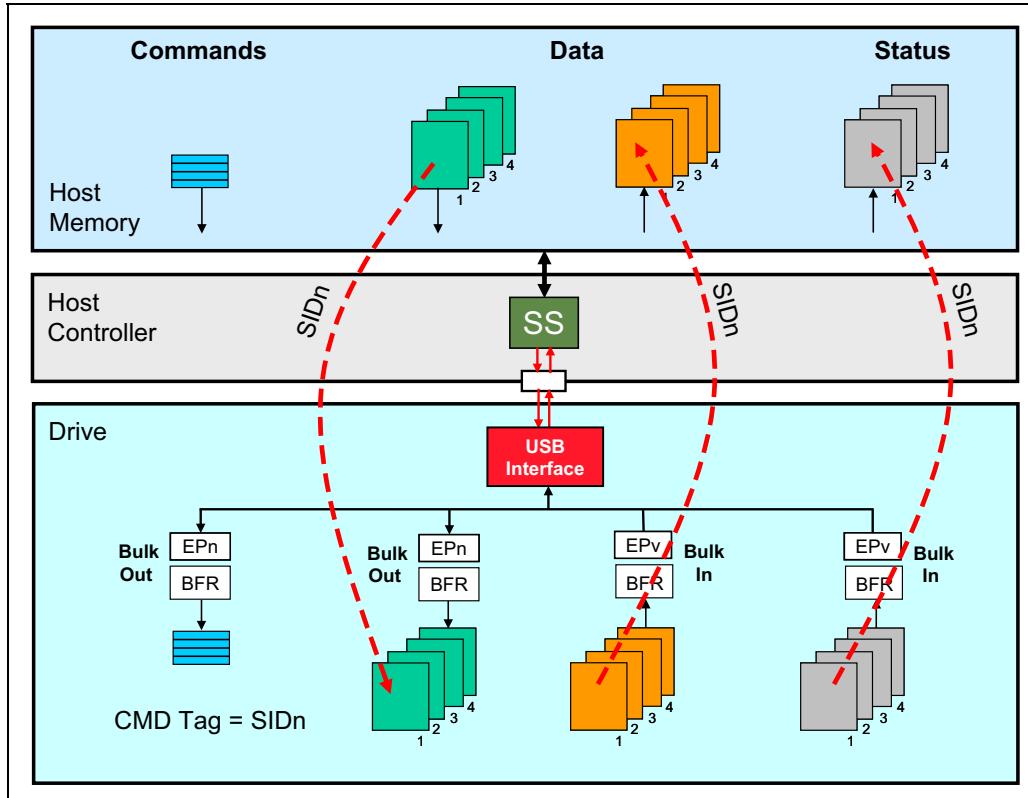
Bulk streaming provides the capability to enqueue multiple commands and execute them out-of-order to improve overall performance. Two key specifications provide the framework for SCSI-compatible USB bulk streaming:

- UAS (USB Attached SCSI) — defines the commands and protocols associated with the full SCSI Architecture Model 4 (SAM 4) command set.
- UASP (USB Attached SCSI with SuperSpeed Protocols) — this specification applies the SuperSpeed protocols to the UAS implementation.

In the example depicted in Figure 7-17, a USB Attached SCSI (UAS) drive with a command queuing capability of four (Queue Depth = 4) is attached to a root hub port of a USB 3.0 host controller. Note the four streaming buffer sets.

## Chapter 7: Bulk Protocol

Figure 7-17: UASP Streaming Drive Endpoints And System Resources



The resources and behaviors required to manage the UAS drive depicted in Figure 7-17 on page 155 are described in the sections that follow.

### UAS Device Endpoints

In addition to the default control endpoint, EP0, a UAS drive requires three bulk streaming endpoints and one standard (non-streaming) bulk endpoint.

- **Bulk OUT (Command) Endpoint** — this standard (non-streaming) bulk OUT endpoint is the target of USB transactions carrying SCSI commands to the UAS drive. The single endpoint buffer associated with the Command endpoint has one entry for each supported stream (up to 64 K).

## **USB 3.0 Technology**

---

- **Bulk OUT (Data) Endpoint** — this bulk streaming OUT endpoint is the target of USB transactions carrying DMA Write data to the UAS drive. Stream ID (SID) field in packet headers routes packets between main memory Out Data stream buffer and corresponding endpoint stream buffer. MaxPacket size: 1024 bytes; bursting: 1-16 data packets.
- **Bulk IN (Data) Endpoint** — this bulk streaming IN endpoint is the target of USB transactions returning DMA Read data from the UAS drive. Stream ID (SID) field in packet headers routes packets between endpoint stream buffer and corresponding main memory In Data stream buffer. MaxPacket size: 1024 bytes; bursting: 1-16 data packets.
- **Bulk IN (Status) Endpoint** — this bulk streaming IN endpoint is the target of USB transactions requesting the return of SCSI status for previously executed SCSI commands. Stream ID (SID) field in packet headers is used to associate Bulk IN (Status) endpoint stream buffer with corresponding main memory Status stream buffer.

### **Host Memory Buffers**

As indicated in Figure 7-17 on page 155, the total number of main memory buffers required for a UAS drive interface depends on the number of streams.

- **Command (CMD) Buffer** — a single main memory Command (CMD) buffer is used to enqueue SCSI commands that will ultimately target the device Bulk OUT (Command) endpoint. The number of bytes in a command is SCSI-defined (example: 31 bytes).
- **Out Data Buffer (per stream)** — one main memory Out Data buffer is required for each Bulk OUT data stream. Data in this buffer becomes the data packet payload for USB OUT transactions sent by the host controller to a Bulk OUT (Data) endpoint stream buffer.
- **In Data Buffer (per stream)** — one main memory In Data buffer is required for each Bulk IN data stream. The host controller fills this buffer with the contents of data packet payloads returned during USB IN transactions sent by the host controller to the corresponding Bulk IN (Data) endpoint stream buffer.
- **Status Buffer (per stream)** — one main memory Status buffer is required for each supported stream. The UAS drive returns status for each command as it completes and Stream ID (SID) information determines which Status buffer receives the status. The number of status bytes, and the meaning, is defined by SCSI (example: 13 bytes)

## Chapter 7: Bulk Protocol

---

### Host Controller (HC)

The host controller provides the interface between the USB software stack and the downstream devices. In the discussion that follows, the role of the host controller and its software driver are presented in general terms; details on specific host controllers, such as those based on the eXtensible Host Controller Interface--xHCI, are described in separate specifications.

Key services the host controller provides in supporting bulk streaming and the UAS drive include:

- **USB Transaction Generation** — Based on software requests, the host controller fetches Transfer Descriptors for each data stream and executes the required USB SuperSpeed Bulk IN/OUT transactions.
- **Main Memory Access (DMA)** — As needed, the host controller performs DMA read/write operations to access main memory buffers for the purposes of fetching SCSI commands and outbound data as well as storing inbound data and SCSI status.
- **End-to-End Flow Control** — The host controller manages header packet flow control and device NRDY and ERDY flow control packets. Service of data streams is paused and switched as needed on a stream basis.
- **Protocol Layer Error Handling** — In addition to the link level error handling required of all SuperSpeed link partners, the host controller performs all of the Protocol Layer data packet checks and packet acknowledgement.
- **Other Services** — The host controller also notifies the host in the event of device attachment/removal, forwards Device Notifications upstream, etc.

---

### Example UAS and UASP Commands

This section illustrates and describes the execution of three USB Attached SCSI (UAS) command types. This is followed by the same command execution but with the SuperSpeed USB protocols (UASP) applied. Note that the USAP examples presume that the streaming endpoints have already been primed.

- Non-Data Command
- Read DMA Command
- Write DMA Command

The UAS commands use SCSI terminology where an Initiator and Target perform the commands. This is followed by a depiction of the SuperSpeed USB protocol that encapsulates the SCSI packets within USB Data packets.

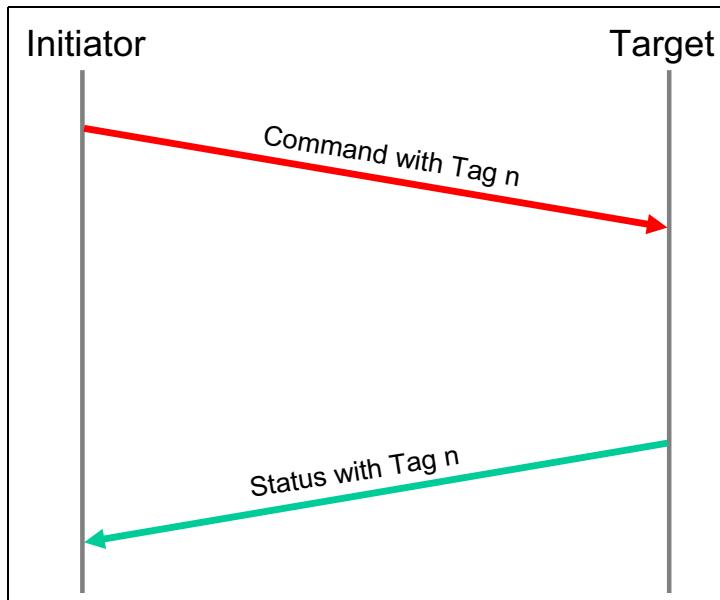
## **USB 3.0 Technology**

---

### **The Non-Data UAS Command**

Figure 7-18 represents a Non-Data command in which a SCSI Initiator delivers the command (including a Tag) to the SCSI Target. The Target decodes and executes the command, and simply returns a SCSI Status packet (including the Tag) to report the result.

*Figure 7-18: UAS Non-Data Command*

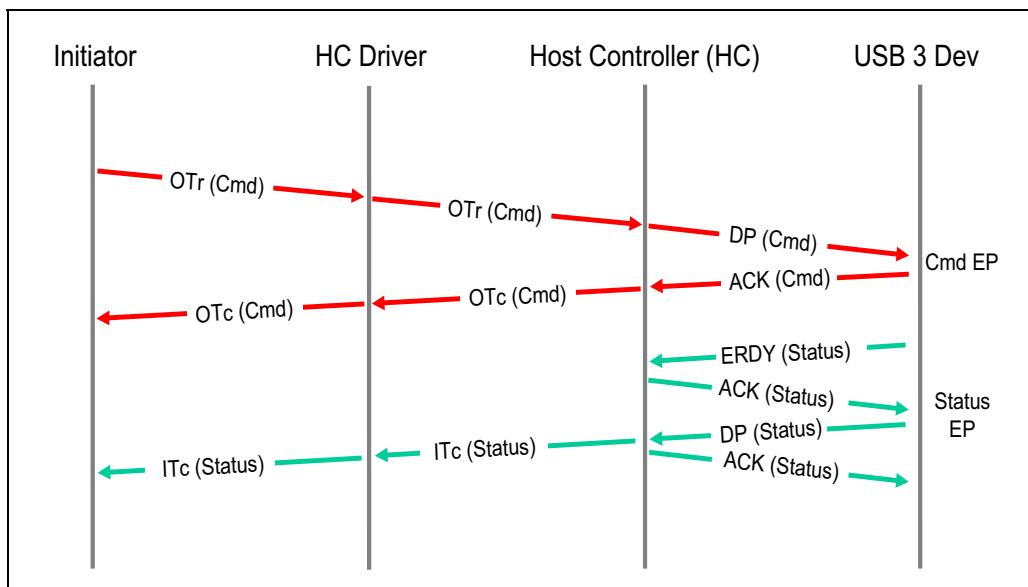


# Chapter 7: Bulk Protocol

## Non Data UASP Command

Figure 7-19 applies the UASP protocol to the previous UAS example.

Figure 7-19: UASP Non-Data Sequence



The sequence below is based on Figure 7-19 and depicts events following a host controller (HC) driver request to issue a SCSI command and perform a non-data operation. The descriptions below define actions taken by the SCSI client driver, HC driver and Host Controller (HC) hardware. The actions described are generic and do not imply any specific hardware implementation.

1. The Initiator in Figure 7-19 represents the software initiating delivery of a SCSI command to the drive. The SCSI client driver allocates a Command queue memory buffer and loads the command into the buffer.
2. Next, the client driver passes the command request (including the 16-bit Tag) to USB software. The HC driver receives the request and sets up a transfer of the SCSI command to the UAS drive Bulk Out (Command) endpoint. The host controller is notified of the pending transaction.
3. HC performs the OUT transaction of the SCSI command. In Figure 7-19, this event is tagged OTr (CMD). The protocol actions include:
  - The HC transmits a DATA packet (containing the SCSI command).
  - Upon receiving the command, the Command Endpoint returns an ACK packet. The drive decodes and starts executing the command.

## **USB 3.0 Technology**

---

- An HC interrupt is generated to notify the HC driver and SCSI client that a command has been successfully received by the UAS drive. In Figure 7-19 on page 159, this event is tagged OTc (CMD)
- 4. SCSI client allocates a main memory Status buffer and initiates a request (including the 16-bit Tag) to the HC driver. The HC driver receives the request and sets up a transfer of the SCSI status from the UAS drive Bulk IN (Status) endpoint. The host controller is notified of the pending transaction.
- 5. ERDY returned by Bulk IN (Status) endpoint when command completes and status is available.
- 6. HC performs the IN transaction of SCSI status. Protocol actions include:
  - The HC transmits an ACK packet with the SID value
  - Status DATA packet transmission is performed with a single packet.
  - The HC returns an ACK to the drive verifying successful transmission.
- 7. Upon conclusion of the status transfer an interrupt is generated to notify the HC driver and SCSI client that the data (Status) is available. In Figure 7-19 on page 159, this event is tagged ITc (Status).

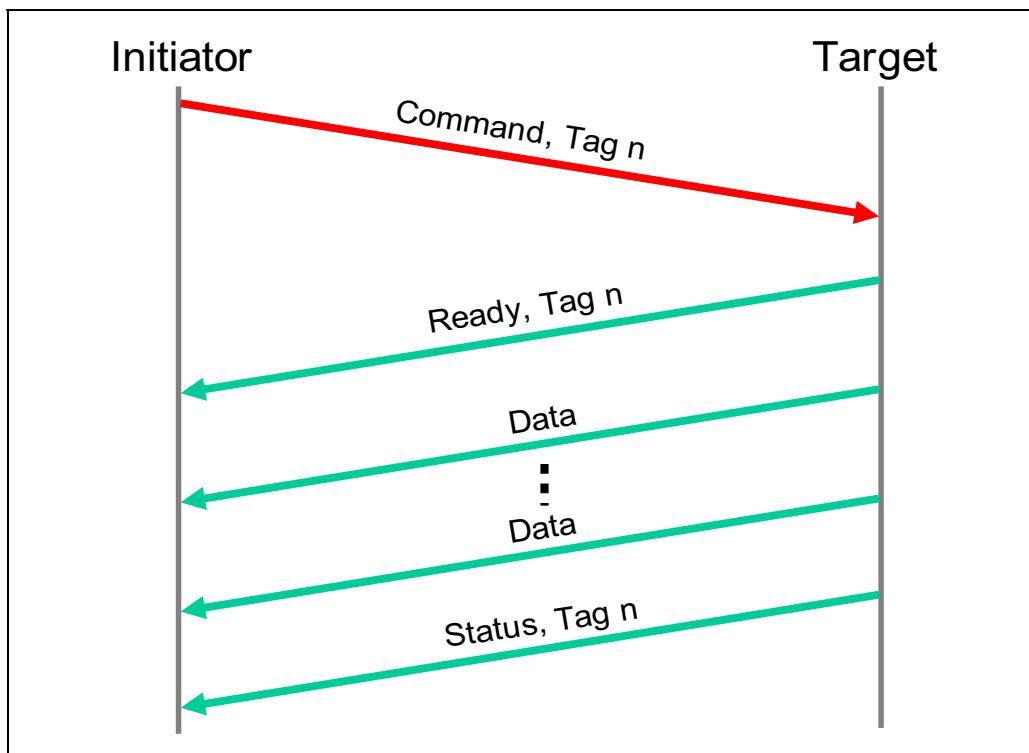
### **Read DMA UAS Command**

Figure 7-20 on page 161 represents a UAS Read DMA sequence in which a SCSI Initiator delivers the command (including a Tag) to the SCSI Target. The Target detects a DMA Read Operation and performs the requested read from disk. When the UAS drive is ready to deliver data it sends a Ready packet (including the Tag) to notify the Initiator that data is ready for delivery. Next, the Target delivers Data packets until the requested Data has been transferred. Finally, the Target transmits the Status packet (including the Tag) and the command completes.

## Chapter 7: Bulk Protocol

---

Figure 7-20: UAS Read DMA Sequence

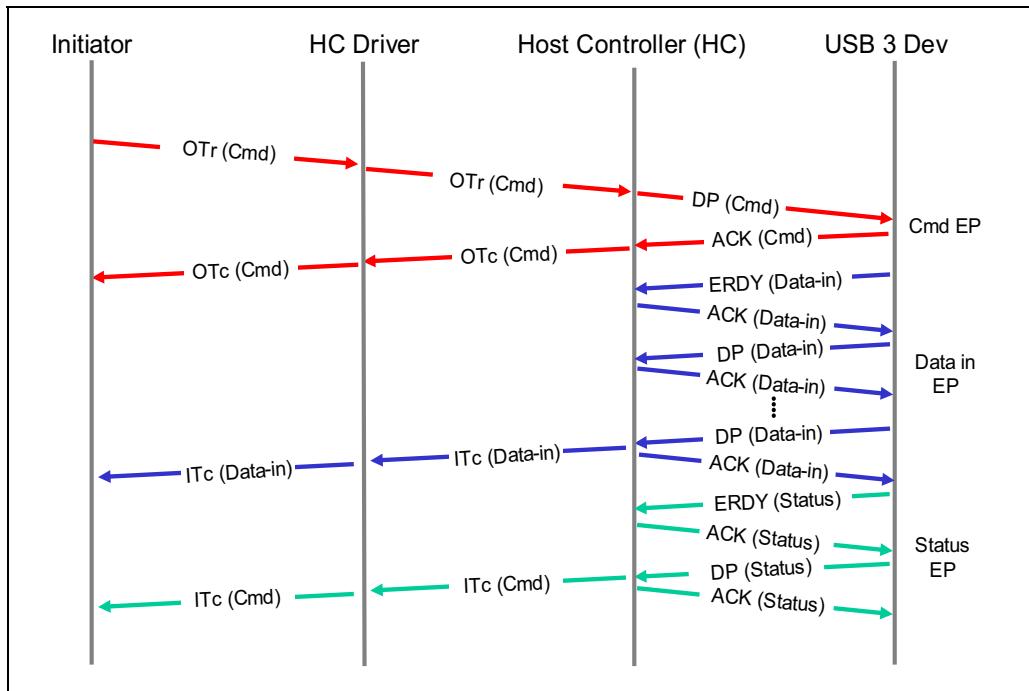


## USB 3.0 Technology

### Read DMA UASP Command

Figure 7-21 on page 162 applies the UASP protocol to the previous UAS Read DMA example.

Figure 7-21: UASP Read DMA Protocol



The sequence below is based on Figure 7-21 and describes events following a host controller (HC) driver request to issue a SCSI command and perform a UAS Read DMA operation. The descriptions define actions taken by the SCSI client driver, HC driver and Host Controller (HC) hardware. Actions described are generic and do not imply any specific hardware implementation.

1. The Initiator in Figure 7-21 on page 162 represents the software initiating delivery of a SCSI command to the drive. The SCSI client driver allocates a main memory Command (CMD) queue buffer and loads the command into the buffer.
2. Next, the client driver passes the command request (including the 16-bit Tag) to USB software. The HC driver receives the request and sets up a

## **Chapter 7: Bulk Protocol**

---

- transfer of the SCSI command to the UAS drive Bulk OUT (Command) endpoint. The host controller is notified of the pending transaction.
3. HC performs the OUT transaction of the SCSI command. In Figure 7-21 on page 162, this event is tagged OTr (CMD). The protocol actions include:
    - The HC transmits a DATA packet (containing the SCSI command).
    - Upon receiving the command, the Bulk OUT (Command) Endpoint returns an ACK packet. The drive decodes and starts executing the command.
    - An HC interrupt is generated to notify the HC driver and SCSI client that the command has been successfully received by the UAS drive. In Figure 7-21 on page 162, this event is tagged OTc (CMD)
  4. SCSI client allocates a main memory In Data buffer and initiates a transfer request (including the 16-bit Tag) to the HC driver. The HC driver receives the request and sets up required data transactions targeting one of the UAS Bulk IN (Data) endpoint streams (using SID). Host controller is notified of the pending transaction. The protocol actions include:
    - After the UAS drive controller processes the Read DMA command, data is retrieved and ERDY is returned by Bulk IN (Data) endpoint stream buffer when at least one data packet is available for transfer.
    - Upon receiving ERDY, the host controller commences sending ACK tokens to, and receiving data packets from, the Bulk IN (Data) endpoint stream buffer using normal USB flow control and packet acknowledgement mechanisms. This exchange continues until all requested data is returned.
    - When the last data packet is received by the host controller, an interrupt is generated to notify the HC driver and SCSI client that the data transfer has completed. In Figure 7-21 on page 162, this event is tagged ITc (Data).
  5. SCSI client allocates a main memory Status buffer and initiates a request (including the 16-bit Tag) to the HC driver. The HC driver receives the request and sets up a transfer of the SCSI status from the UAS drive Bulk IN (Status) endpoint. The host controller is notified of the pending transaction.
  6. ERDY returned by Bulk IN (Status) endpoint when command completes and status is available.
  7. HC performs the IN transaction of SCSI status. Protocol actions include:
    - The HC transmits an ACK packet with the SID value
    - Status DATA packet transmission is performed with a single packet.
    - The HC returns an ACK to the drive verifying successful transmission.
    - HC updates main memory Status buffer.
  8. Upon completion of the status transfer, an interrupt is generated to notify the HC driver and SCSI client that SCSI status for the UAS Read DMA operation is available. In Figure 7-21 on page 162, this event is tagged ITc (Cmd).

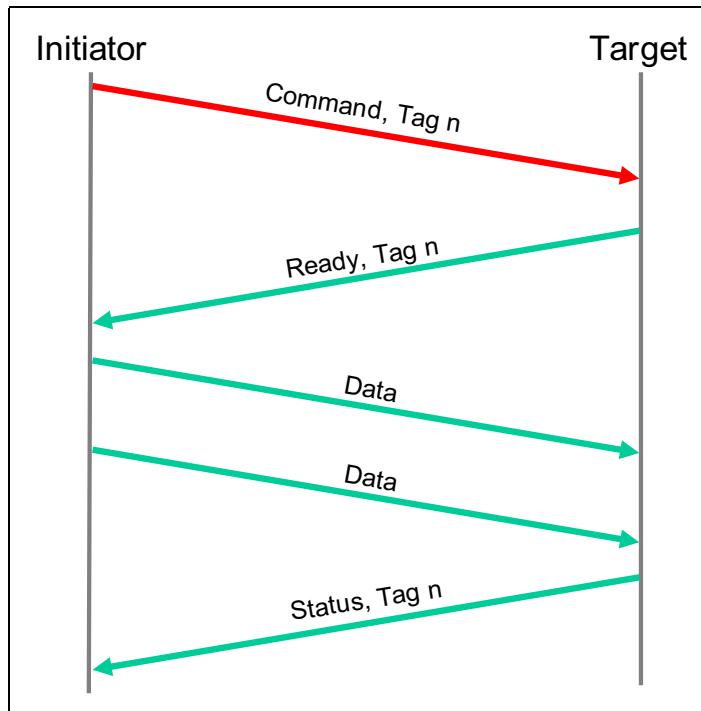
## **USB 3.0 Technology**

---

### **Write DMA UAS Command**

Figure 7-22 on page 164 represents a UAS Write DMA sequence in which a SCSI Initiator delivers the command (including a Tag) to the SCSI Target. The Target detects a DMA Write Operation and prepares to accept data from the initiator and write it to the disk. When the UAS drive is ready to accept data it sends a Ready packet (including the Tag) to notify the Initiator. Next, the Initiator delivers Data until the transfer size is satisfied. At this point, the Target transmits the Status packet (including the Tag) and the command completes.

*Figure 7-22: UAS Write DMA Sequence*

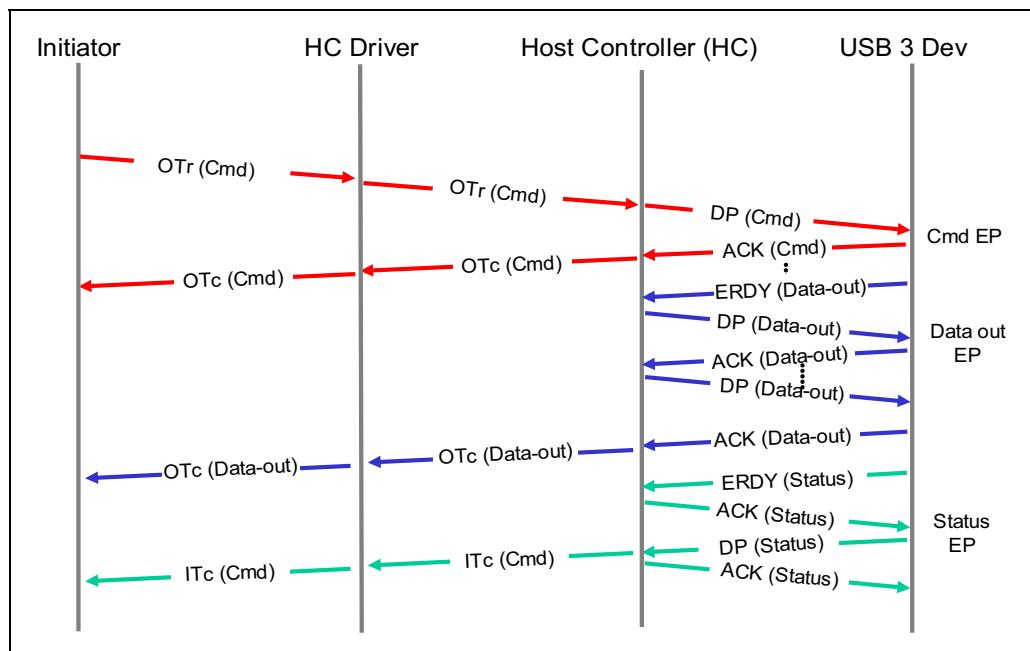


# Chapter 7: Bulk Protocol

## Write DMA UASP Command

Figure 7-23 applies UASP protocol to the previous UAS Write DMA example.

Figure 7-23: UASP Write DMA Sequence



The sequence below is based on Figure 7-23 and describes events following a host controller (HC) driver request to issue a SCSI command and perform a UAS Write DMA operation. The descriptions define actions taken by the SCSI client driver, HC driver and Host Controller (HC) hardware. Actions described are generic and do not imply any specific hardware implementation.

1. The Initiator in Figure 7-23 represents the software initiating delivery of a SCSI command to the drive. The SCSI client driver allocates a main memory Command (CMD) queue buffer and loads the command into the buffer.
2. Next, the client driver passes the command request (including the 16-bit Tag) to USB software. The HC driver receives the request and sets up a transfer of the SCSI command to the UAS drive Bulk OUT (Command) endpoint. The host controller is notified of the pending transaction.
3. HC performs the OUT transaction of the SCSI command. In Figure 7-23, this event is tagged OTr (CMD). The protocol actions include:

## **USB 3.0 Technology**

---

- The HC transmits a DATA packet (containing the SCSI command).
  - Upon receiving the command, the Bulk OUT (Command) Endpoint returns an ACK packet. The drive decodes and starts executing the command.
  - An HC interrupt is generated to notify the HC driver and SCSI client that the command has been successfully received by the UAS drive. In Figure 7-23 on page 165, this event is tagged OTc (CMD)
4. SCSI client allocates a main memory Out Data buffer and loads data to be transferred to the target UAS device OUT endpoint. Client initiates a transfer request (including the 16-bit Tag) to the HC driver. The HC driver receives the request and sets up required data transactions targeting the UAS Bulk OUT (Data) endpoint stream (using SID). Host controller is notified of the pending transaction. The protocol actions include:
    - After the UAS drive controller processes the Write DMA command, ERDY is returned by Bulk OUT (Data) endpoint stream buffer when at least one data packet can be accepted.
    - Upon receiving ERDY, the host controller commences sending Data Packets to, and receiving ACK packets from, the Bulk OUT (Data) endpoint stream buffer using normal USB flow control and packet acknowledgement mechanisms. This exchange continues until all requested data is sent.
    - When the last data packet is sent by the host controller, an interrupt is generated to notify the HC driver and SCSI client that the data transfer has completed. In Figure 7-23 on page 165, this event is tagged OTc (Data).
  5. SCSI client allocates a main memory Status buffer and initiates a request (including the 16-bit Tag) to the HC driver. The HC driver receives the request and sets up a transfer of the SCSI status from the UAS drive Bulk IN (Status) endpoint. The host controller is notified of the pending transaction.
  6. ERDY returned by Bulk IN (Status) endpoint when command completes and status is available.
  7. HC performs the IN transaction of SCSI status. Protocol actions include:
    - The HC transmits an ACK packet with the SID value
    - Status DATA packet transmission is performed with a single packet.
    - The HC returns an ACK to the drive verifying successful transmission.
    - HC updates main memory Status buffer.
  8. Upon completion of the status transfer, an interrupt is generated to notify the HC driver and SCSI client that SCSI status for the UAS Write DMA operation is available. In Figure 7-23 on page 165, this event is tagged ITc (Cmd).

---

---

# 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.

## USB 3.0 Technology

- Endpoint descriptors specify a service interval (sometimes called the polling interval) that defines the frame or  $\mu$ frame 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:

$$2^{\text{interval}-1}$$

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

$$2^{7-1} \rightarrow 2^6 \rightarrow 64 \text{ } \mu\text{frames} = 8\text{ms}$$

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.

# Chapter 8: Interrupt Protocol

---

## Interrupt IN SuperSpeed Transaction Protocol

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

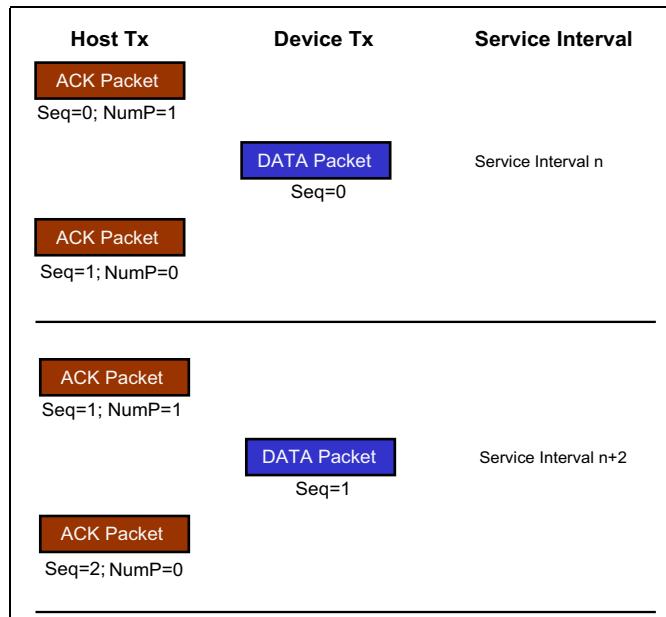
- Single Interrupt Transaction
- Burst transaction sequence (two DATA packet burst)
- Transaction with NRDY
- Transaction with eob
- Burst transaction with DATA packet error and Retry in same interval
- Burst transaction with DATA packet error and Retry in next interval

---

### Single Interrupt Transaction

The example in Figure 8-1 illustrates an interrupt transactions that transfers data during every other  $\mu$ frame. 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*



# USB 3.0 Technology

## **Interrupt Burst Transactions (two DATA packets)**

Figure 8-2 on page 170 illustrates an interrupt IN burst of transfer with a MaxBurstSize of two DATA packets. Note that the host controller requests two DATA packets by setting NumP=2. As each DATA packet is acknowledged NumP then decrements as each DATA packet complete successfully.

The example further illustrates a service interval of 64. Once the burst of transactions complete during Bus Interval n, the host controller will schedule additional DATA packets during Bus Interval n+64.

*Figure 8-2: Interrupt IN Burst Example with Service Interval of 64*

<b>Host Tx</b>	<b>Device Tx</b>	<b>Service Interval</b>
ACK Packet	DATA Packet Seq=0	Bus Interval n
Seq=0; NumP=2		
ACK Packet		
Seq=1; NumP=1		
ACK Packet		
Seq=2; NumP=0		
ACK Packet	DATA Packet Seq=2	Bus Interval n+64
Seq=2; NumP=2		
ACK Packet		
Seq=3; NumP=1		
ACK Packet	DATA Packet Seq=3	
Seq=4; NumP=0		

## Chapter 8: Interrupt Protocol

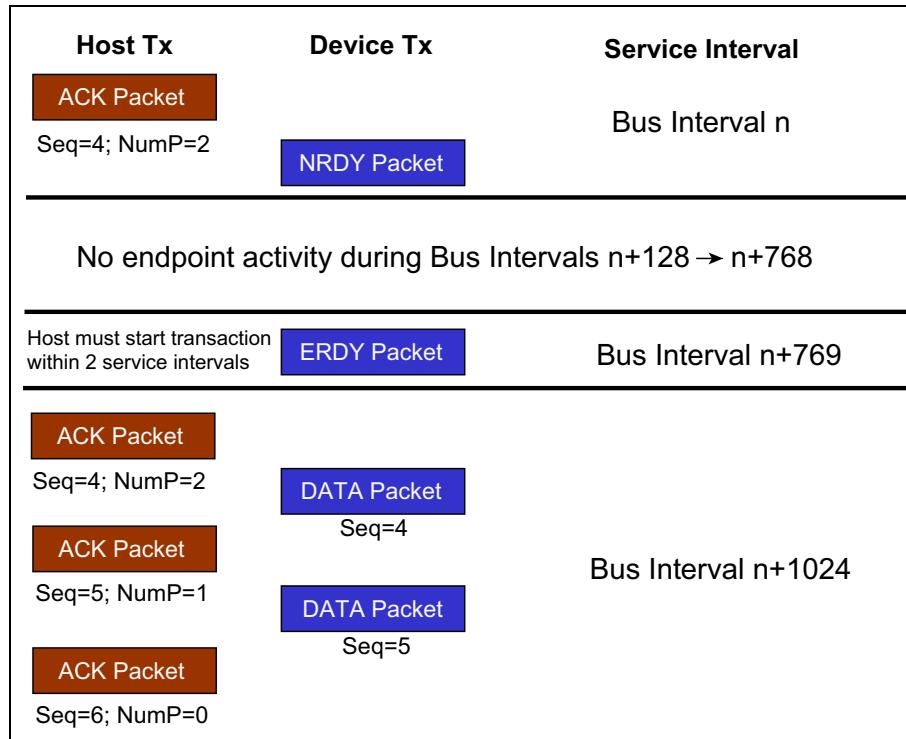
End-to-End flow control during IN transactions occur when NRDY or EOB are asserted. When the endpoint has data to deliver, the endpoint asserts ERDY, thus notifying the host to re-initiate the transaction.

- NRDY — used when a DATA packet cannot be sent at this time
- EOB bit set in the DATA packet — used to notify the host that the next DATA packet temporarily cannot be sent.

In response to these flow control events, the host controller deactivates the current transaction or burst of transactions.

The next example (Figure 8-3) illustrates a continuation of the previous example (Figure 8-2 on page 170), but the endpoint has send NRDY to the host, thereby temporarily stopping the transaction during bus intervals n+128 through n+768. During bus interval 769 the endpoint sent ERDY to notify the host that the endpoint was again ready to send data. The host restarts the transaction during bus interval n+1024, which is within the required two service intervals.

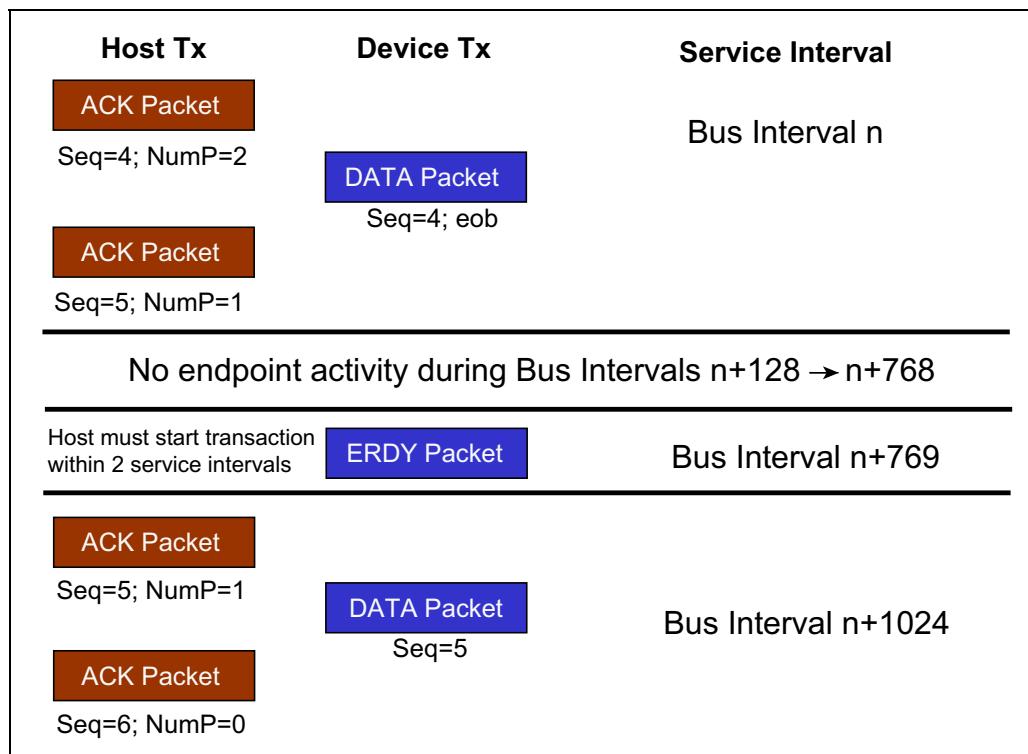
Figure 8-3: Interrupt IN with NRDY



## USB 3.0 Technology

This example is the same as the previous with the exception that end of burst (**eob**) is set within the DATA packet. In this case, the data is transferred to the host and the eob indicates to the host that the transaction is temporarily deactivated. As in the previous example there is no activity during bus interval n+128 through n+768, but during n+769 the endpoint sends ERDY, thereby notifying the host that the transaction can be re-initiated. As required the host sends an ACK packet within two service intervals following ERDY be sent.

Figure 8-4: Interrupt IN Transaction with EOB



## Chapter 8: Interrupt Protocol

---

### Interrupt IN — Errors and Retries

Errors may result from a variety of scenarios depending on the timing of the error, bandwidth requirements of the endpoint and behaviors of the host. The specification notes that in the event of an error the host is not required to retry the transaction in the current service interval. However, the host must retry the transaction no later than the next service interval.

This leaves open the question: Under what conditions might the host decide to retry a transaction or not in the current service interval? A possible rationale for making this decision is the bandwidth allocation per service interval. For example, if the bandwidth allocation is 3 KBs/service interval, and the first of the 3 DATA packets incurs an error, a successful retry and a second DATA packet transfer would not exceed the allotted bandwidth. That is, three DATA packets would have been sent but the third packet scheduled would not have been sent. Conversely, if the first two DATA packets had transferred successfully and the last once failed, all of the allotted bandwidth would have already been consumed and the retry would be scheduled in the next service interval.

The specification is unclear as to how the host controller should notify the target device that DATA packet 7 failed and must be retried without it being performed during the current interval. If the host controller simply returns a typical ACK with **retry=1** and **NumP=1**, the device would immediately retry the failed DATA packet. There seems to be two possible solutions:

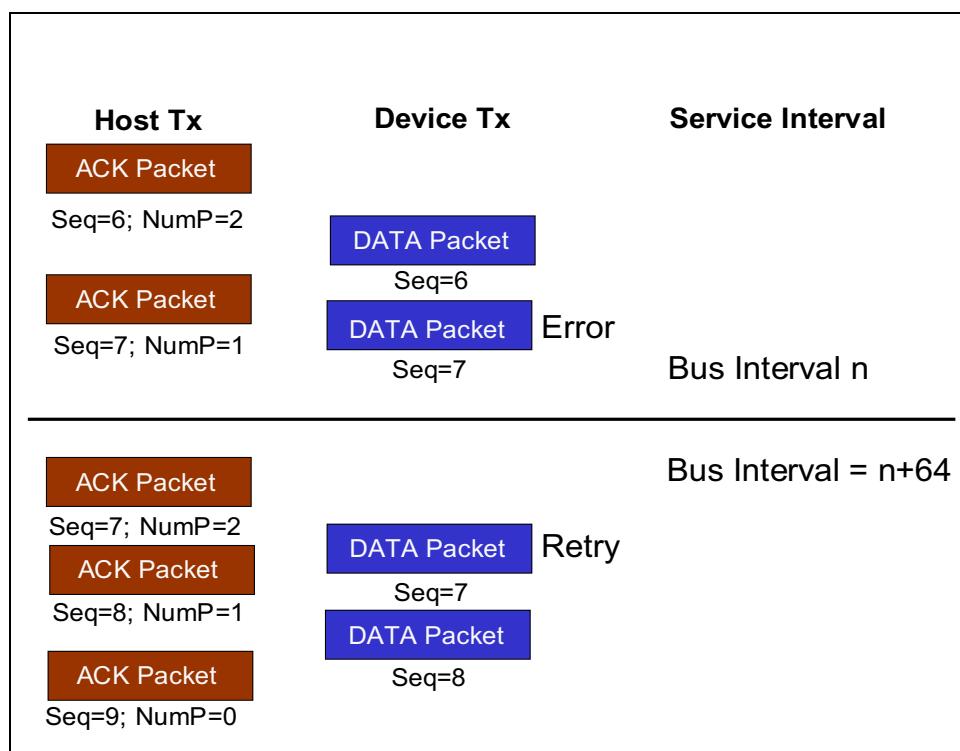
1. The host controller could return an ACK with retry=1 (indicating the DATA packet must be retried) and NumP=0 (indicating the current burst sequence has completed). This introduces a condition (retry=1 and NumP=0) not described in the specification.
2. The host controller could decide to not return any packet to the target endpoint in the current service interval. Failing to detect an ACK the device would retain the data and wait for ACK to be send in the next interval. However, the specification clearly states that the host controller, in response to a failed DATA packet, must return an ACK packet with retry=1.

Two sample error conditions are illustration on the following pages.

## USB 3.0 Technology

Figure 8-5 on page 174 depicts a burst of two transactions scheduled during service interval n with an error detected in DATA packet 7. The host controller decides to perform the retry in the next service interval (n+64), perhaps because the periodic bandwidth is near its 90% limit. In this example, the host initiates a retry during the next bus interval (n+64).

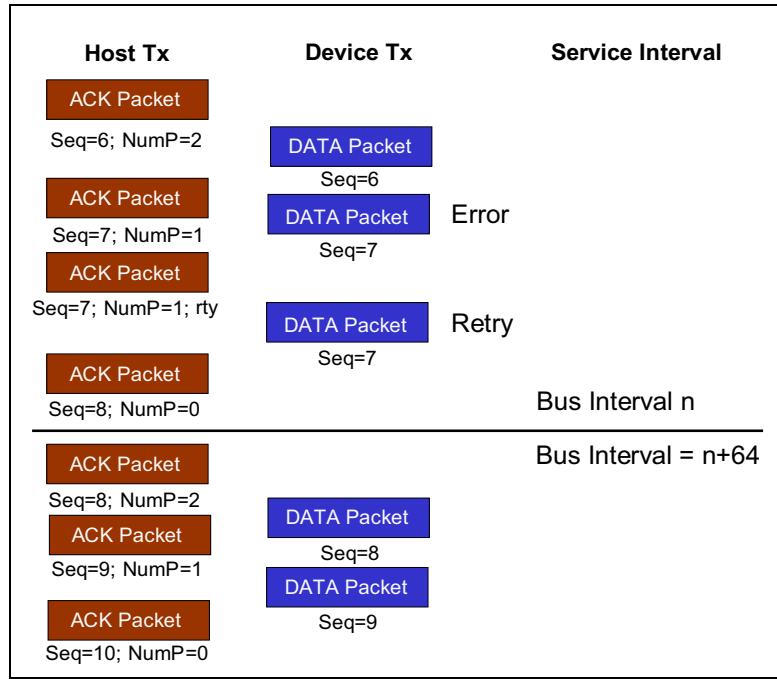
*Figure 8-5: Interrupt IN with Retry in Next Service Interval*



## Chapter 8: Interrupt Protocol

Figure 8-6 on page 175 illustrates a burst of two transactions scheduled during service interval n, however DATA packet 7 incurs an error and the host returns an ACK packet notifying the target device that a retry is needed. Even though the scheduled bandwidth has been consumed for the interrupt service interval, the host controller is aware that additional periodic bandwidth is available. Consequently, there is no harm in retrying the transaction.

Figure 8-6: Interrupt IN Retry Example



# **USB 3.0 Technology**

---

## **Interrupt OUT SuperSpeed Transaction Protocol**

The host issues DATA Headers and the associated DATA Packet Payloads (DPP) to initiate OUT transactions. Interrupt OUT protocol is the same as bulk OUT protocol with the exception for the three items listed below:

- Data burst size is limited to a burst of three
- Interrupt OUT transactions must be delivered during the service interval specified by the device endpoint descriptor
- Burst flow control, data transfer errors and retries must be tied to the service interval

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

### **Interrupt OUT Data Bursting**

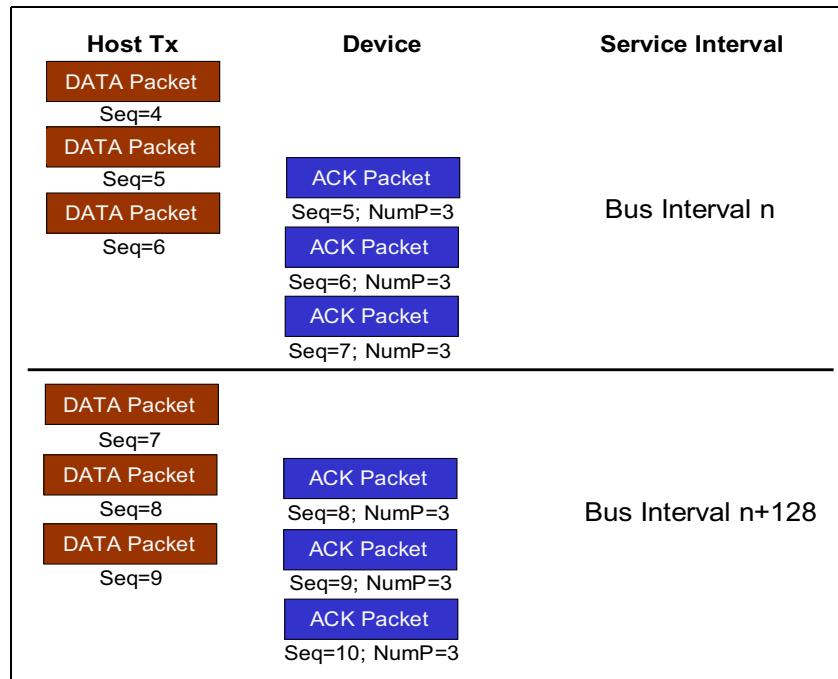
Interrupt OUT transaction characteristics include the following:

- Whether or not an interrupt endpoint supports bursting is reported in the Endpoint Companion descriptor and is limited to a burst of three. Table 7-1 on page 127 shows a portion of the descriptor that defines data bursting.
- Interrupt OUT transactions continue from one service interval to the next as long as DATA packets continue to be ACKed by the endpoint.
- Upon receiving a DATA packet, an Interrupt OUT endpoint that temporarily cannot receive the DATA must respond with an NRDY (or STALL in the event of device errors).
- The host resumes DATA transmission after receiving an ERDY packet and must do so within two service intervals.

## Chapter 8: Interrupt Protocol

The following example illustrates an Interrupt OUT burst of transactions with a MaxBurstSize of three DATA packets. The example further illustrates a service interval of 128. During interval n there is a burst of three DATA packets all acknowledged by the device without error. This is followed by a burst of three DATA packets during interval n+128 that also complete without error.

Figure 8-7: Interrupt OUT Burst



### End-to-End Flow Control — Interrupt OUT Transfers

Interrupt OUT endpoints use the same End-to-End flow control mechanisms that are used in Bulk transfers. In summary, interrupt flow control consists of:

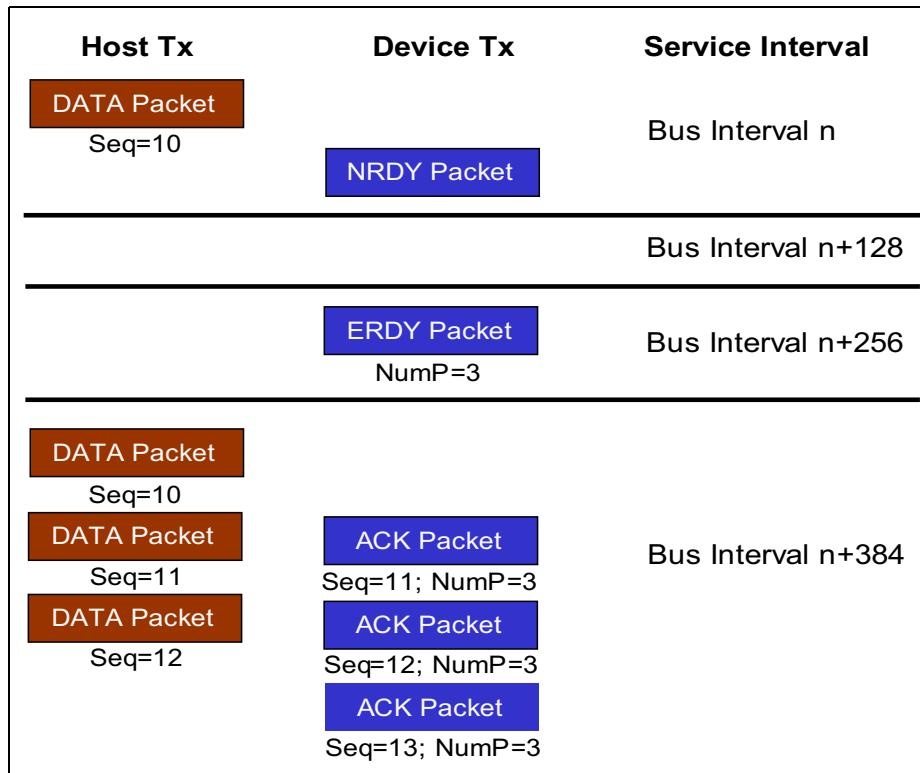
- NRDY — used when a DATA packet cannot be sent at this time
- NumP=0 set in the ACK packet — used to notify the host of an end of burst condition

## USB 3.0 Technology

In response to these flow control events, the host controller deactivates the current transaction or burst of transactions.

The next example (Figure 8-8) illustrates a continuation of the previous example, but the activity associated with the endpoint has temporarily stopped due to an NRDY being returned by the target device. This condition may last for an extended period of time depending on the nature of the application.

*Figure 8-8: Interrupt OUT with NRDY*

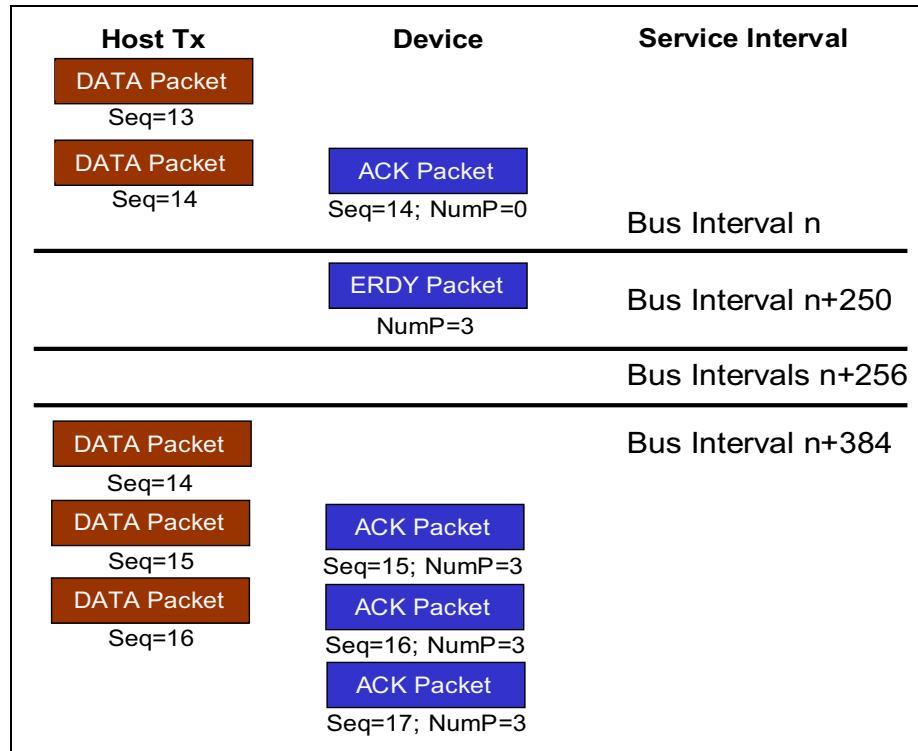


## Chapter 8: Interrupt Protocol

The next example illustrates an end of burst (NumP=0) during an Interrupt OUT burst transaction (Figure 8-9 on page 179). The host controller intends to burst three DATA packets during bus interval n. However, the device returns an ACK packet with Seq=14 (indicating DATA packet 13 has been accepted) and NumP=0 (indicating end of burst). DATA packet 14 was sent by the host controller prior to it receiving the end of burst indication, so the target device must discard this packet.

The host controller, upon detecting NumP=0, deactivates the interrupt OUT burst transactions and waits for the target device to send ERDY. In the example, the target device signals ERDY during bus interval n+128 and the host controller reactivates the burst transactions within two bus intervals (n+384).

Figure 8-9: Interrupt OUT Burst with End of Burst

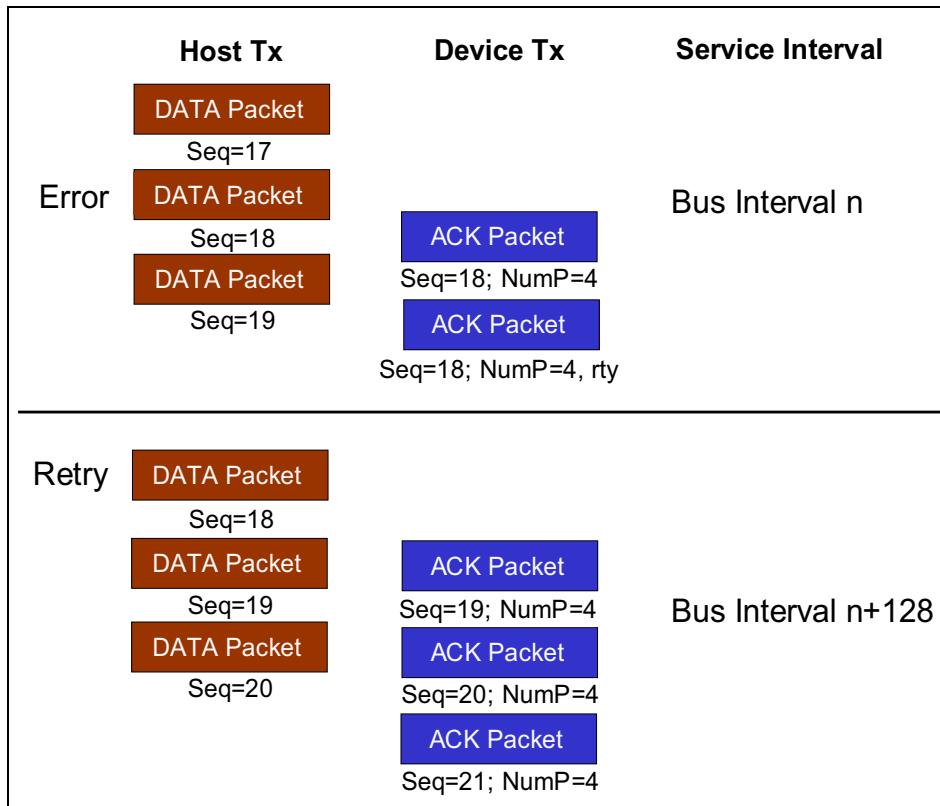


# USB 3.0 Technology

## **Interrupt OUT — Errors, Retry and Service Intervals**

The discussion regarding error detection and retries for interrupt IN transactions also applies to the interrupt OUT transactions (See “Interrupt IN — Errors and Retries” on page 173). Figure 8-10 on page 180 illustrates an example interrupt burst in which the last DATA packet of the service interval incurs an error. The host controller detects the ACK packet with retry and defers the retransmission of DATA packets 18 and 19 until the next service interval.

*Figure 8-10: Interrupt OUT — Retry*



---

---

# 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.

## USB 3.0 Technology

- 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



- 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).

## Chapter 9: Isochronous Protocol

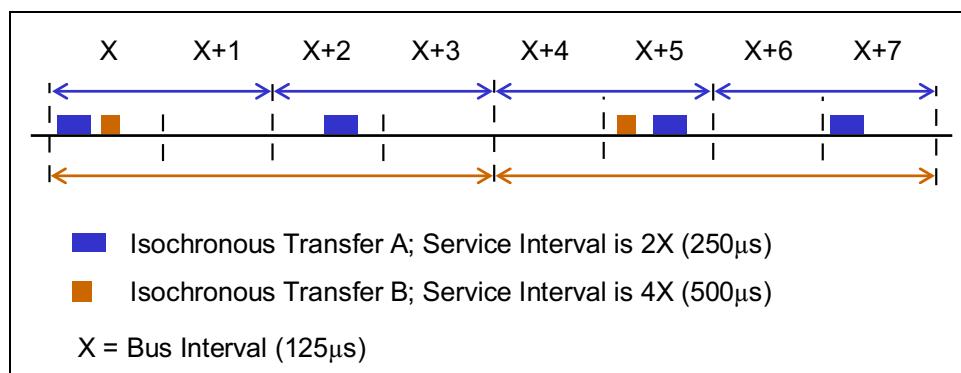
### Bus Intervals and Isochronous Service Intervals

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 ( $\mu$ SOF) 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  $\mu$ SOF 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



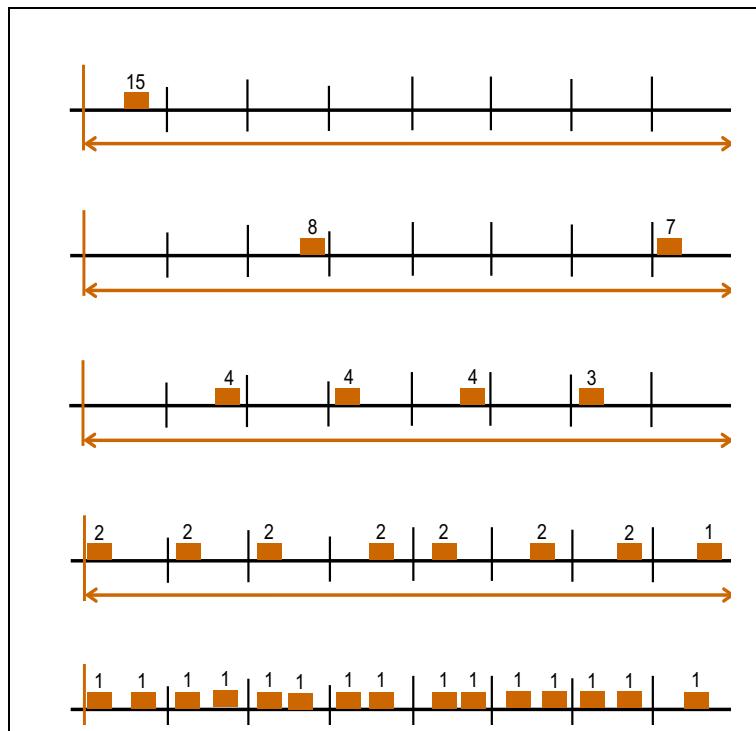
# USB 3.0 Technology

## Flexibility of Scheduling

Host controllers are allowed to schedule isochronous transactions in a variety of ways. The simplest approach is to transfer all of the isochronous data in a single large burst within a single bus interval (e.g. Maximum-sized burst of 48 KB). However load balancing might be important and to this end the specification allows flexibility in scheduling. For example, if the total isochronous burst size is 15, then the options permitted are as follows:

- Single burst of 15
- A burst of 8 and burst of 7
- Three bursts of 4 and burst of 3
- Seven bursts of 2 and burst of 1
- Fifteen bursts of 1

*Figure 9-3: Scheduling Options for Load Balancing within an Isochronous Service Interval*



In addition, the Mult field may specify the number of bursts (1, 2, or 3) that can be performed in a single service interval.

# Chapter 9: Isochronous Protocol

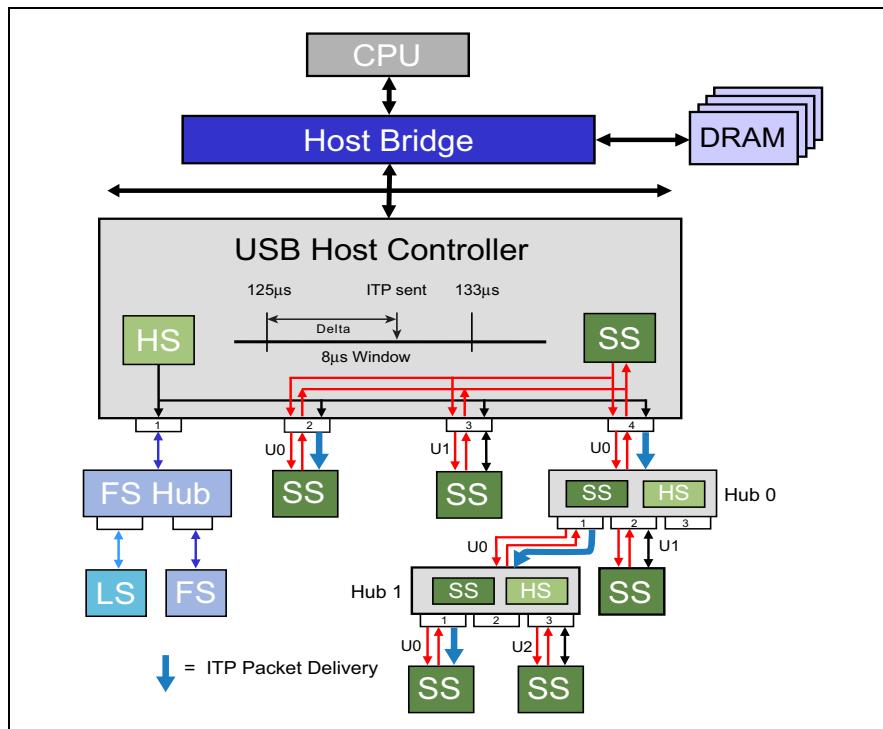
## Isochronous Timestamp Packet Delivery

### General

Unlike the  $\mu$ SOF packets, the ITPs are delivered within an  $8\mu\text{s}$  window following the aligned  $125\mu\text{s}$  boundary. Consequently, ITPs report the Delta from the aligned  $125\mu\text{s}$  boundary to the time at which the root ports start the transmission of ITPs as illustrated in Figure 9-4.

SuperSpeed root hubs multicast the ITPs to all ports that are currently in the U0 state (ports 2 and 4 in this example) and are forwarded downstream to all active devices. Ports not in the U0 state (U1 and U2) do not receive an ITP during the current bus interval because these states indicate the port and associated link are in the electrical idle state. Thus, any device needing to receive ITPs for synchronization must ensure its link is active when the ITP is delivered.

Figure 9-4: Isochronous Timestamp Packet Delivery Example



# USB 3.0 Technology

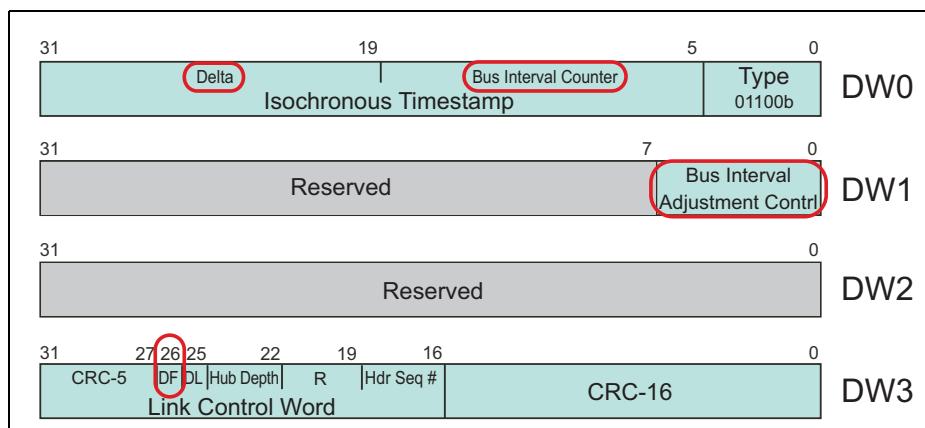
---

## The Timestamp Packet

Figure 9-5 illustrates the format of the Isochronous Timestamp Packet. The packet contains four fields:

- Delta — this value reports the delay between the bus interval boundary and start of ITP transmission by the root hub ports. The Delta value count may be zero if the ITPs start at the aligned bus interval boundary, and otherwise consists of a number of Isochronous Timestamp Granularity units. One unit is defined as 16.67 ns (8 HS bit times).
- Bus Interval Counter — keeps track of bus interval numbers from zero to 16383d. The bus interval count increments every aligned 125  $\mu$ s and rolls over every 2.04 seconds.
- Delayed Bit (DL) — this bit indicates that an ITP has been delayed when being forwarded across a hub.
- Bus Interval Adjustment Control — this field contains zeros until a Bus Interval Adjustment request is made by a device. If the host controller accepts the request, it places the address of the requesting device into this field as a confirmation to the device. (See “Requirements for Changing the Time Base” on page 187.)

Figure 9-5: Isochronous Timestamp Packet Format



## **Chapter 9: Isochronous Protocol**

---

### **Delta Value**

The 13-bit Delta value delivered in an ITP is valid at the moment of ITP transmission begins from the root hub port. Additional delay occurs when the ITP is forwarded to a target device. The propagation delays across each link and each hub must be known so that a true timestamp value can be derived by devices. To accomplish this adjustment, host software calculates the one-way travel time from the root hub ports and the devices that reside downstream. This value is called the Isochronous Delay and is forwarded to each device during initialization via a SetIsochronousDelay request.

### **Delayed Bit**

When ITPs are multicast a hub may incur a delay in forwarding the header packet, causing the hub to set the **delayed (DL)** bit in the Link Control Word. Consequently, the Delta value is now inaccurate and should be discarded.

---

### **Changing the Time Base**

The original 1.0 version of the USB specification included a feature that allowed a device driver to change the frequency of the USB bus. Later versions of the USB specification removed this feature, but some Host Bus Controllers may still permit adjusting the bus clock. The USB specification has added a similar feature for the SuperSpeed bus. A single device may request that the SuperSpeed timebase (nominally 8KHz) be changed in small increments to increase or decrease the bus interval. The likely reason for a device wanting to change the bus frequency to match its own local clock.

Although the specification does not describe the intended application, the authors suspect this feature will be used in some embedded implementations where a single isochronous endpoint could benefit from adjusting the bus frequency to match its sample clock. General PC applications are more likely to have multiple isochronous devices with conflicting sample clocks.

---

### **Requirements for Changing the Time Base**

Unlike the USB 1.0 bus interval adjustment mechanism that required software to initiate a change in bus clock frequency, the SuperSpeed solution requires the device to initiate a change. The request to change the bus frequency is made via a Bus Interval Adjustment (BIA) Message (See Figure 9-6 on page 189) and only one device at a time is allowed to use this mechanism.

## **USB 3.0 Technology**

---

The minimum request is one Bus Interval Adjustment (BIA) Granularity Unit or 4.0690104 ps. When a device requests one BIA Granularity Unit, this tells the Host Controller to apply this unit across 4096 bus intervals. This equates to 8 high-speed bit times or 16.666 ns. The host controller determines how it will apply the change and is allowed to make the adjustment in a single bus interval. When larger requests are made, the host controller is required to apply the change evenly across multiple bus intervals. The maximum adjustment that can be made is +/-13.333  $\mu$ s.

Prior to any requested Bus Interval Adjustment, the BIA Control field within the ITP contains zeros (See Figure 9-5 on page 186). When a device issues a Bus Interval Adjustment request, the host controller may place the address of the requesting device into the BIA Control field. The device monitors subsequent ITPs to see if the BIA control field holds the device address thereby confirming the adjustment or if it contains zero indicating adjustment rejection.

The following list defines the requirements associated with changing the bus frequency.

- The host must support the entire adjustment range from -37268 to 37267 BIA Granularity Units and is reported as a two's complement value within the BIA Control field.
- A device must not attempt a Bus Interval Adjustment request that asks for more than  $\pm 4096$  units.
- A device is restricted to requesting Bus Interval Adjustments only once in every eight bus intervals.
- Subsequent BIA requests cannot be sent until the device has observed the affect of the previous request by checking the timestamp value in the ITP.

---

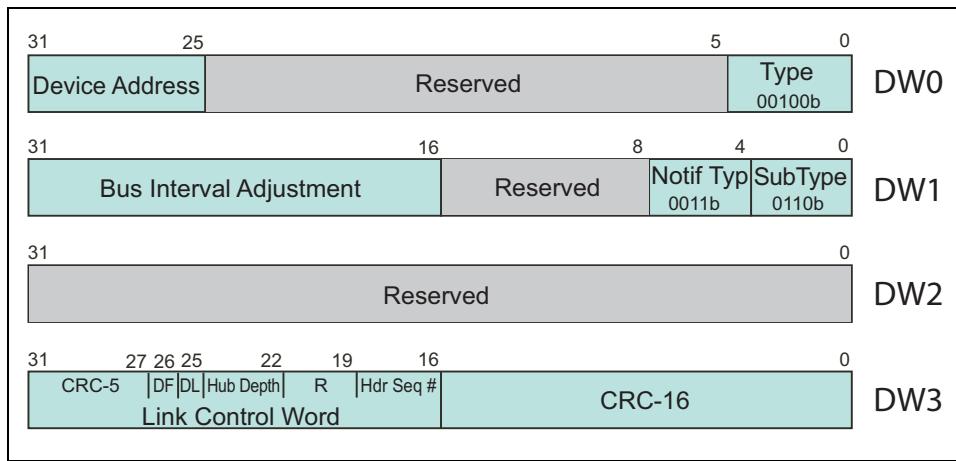
### **Bus Interval Adjustment Message**

Figure 9-6 shows the format of the Bus Interval Adjustment Message header. The specification categorizes this message as a Device Notification, meaning that it can be sent asynchronously by a device to notify the host controller of a particular event. In this example a request to change the duration of the bus interval is being made by the device.

## Chapter 9: Isochronous Protocol

---

Figure 9-6: Bus Interval Adjustment Message



---

### Ping and Ping Response

The SuperSpeed bus can conserve power by placing links into the electrical idle state after a period of idle time. This aggressive power management could leave links between the root port and the isochronous device in an electrical idle state while the isochronous endpoint waits for the next service interval. When the service interval arrives the host initiates an Isochronous transaction that will be delayed while the idled links recover back to the fully powered and operational state (U0). The recovery time may be in the range of hundreds of microseconds, thus potentially preventing the isochronous transfer from completing within the service interval.

To solve this problem the host controller must ensure that all links between the root port and isochronous device are active prior to the next service interval. The host controller accomplishes this by sending a PING packet that targets the isochronous device, causing each link in the path to transition to the U0 state. The device in turn sends a PING RESPONSE packet back to the host controller to confirm all links in the path are in U0.

# **USB 3.0 Technology**

---

## **Ping Related Timing Parameters**

### **Maximum Exit Latency (MEL)**

The host controller knows the worst case round trip delay time required to complete a Ping sequence associated with a given device, called the Maximum Exit Latency. This information helps the host controller in scheduling pings to each isochronous device.

The total MEL consists of four timing parameters:

- tMEL1 — the time required to transition all links in the path between the root port and the target device to transition to U0. This value is derived from the exit latency information provided in the BOS descriptors and includes the Hub Header Decode Latency for all hubs in the path.
- tMEL2 — the time required to send the PING packet from the root port to the target device. This parameter includes the propagation delay across each link (20 symbol times/link) plus the hub delay value (read from the Hub descriptor) for each hub in the path.  
$$tMEL2 = (\text{all } wHubDelay \text{ values}) + (20 \text{ symbol times} * (\text{number of hub} + 1))$$
- tMEL3 — the turn-around time between the device receiving the PING packet and delivering the PING RESPONSE packet. This parameter is called the tPingResponse and is specified to be within 400ns. The timing is further specified as the interval between the last framing symbol of the PING packet received by the endpoint and the first framing symbol of the PING RESPONSE packet sent by the endpoint.
- tMEL4 — the time required to send PING RESPONSE packet from the device to the host. This is the same delay as tMEL2, but may include additional delay if the response gets queued behind a maximum sized DATA packet. This results in a worst case value of:  $tMEL4 = tMEL2 + 2.1 \mu\text{s}$

### **tPingTimeout**

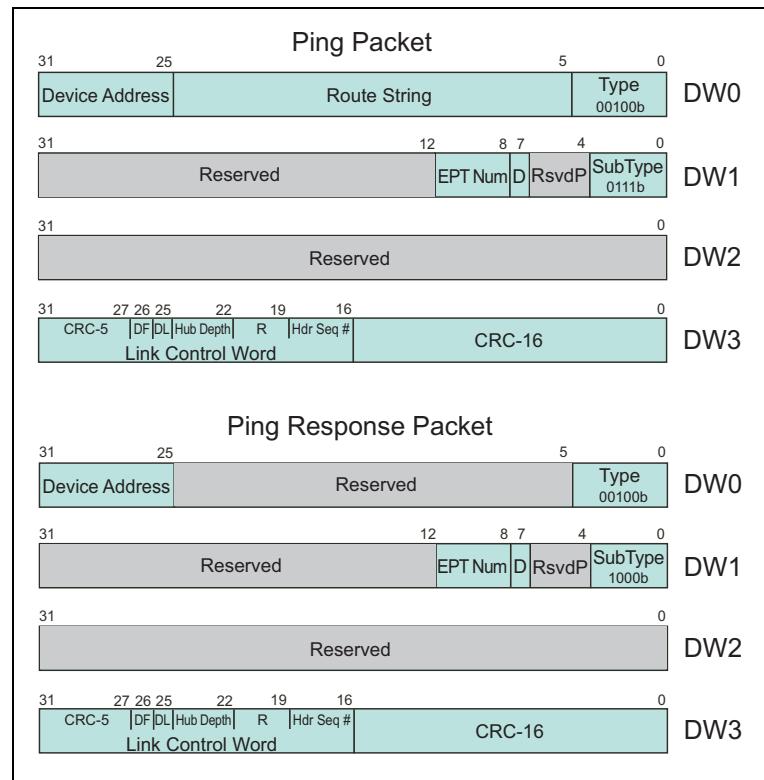
Upon receiving a PING packet the isochronous endpoint must keep its link in the U0 state until a subsequent packet is received from the host. In the unlikely event that a subsequent packet is not received as expected, the tPingTimeout expires allowing transitions to U1 or U2. The tPingTimeout value is set to the maximum of all service intervals for all isochronous endpoints within the device.

## Chapter 9: Isochronous Protocol

### Ping and Ping Response Header Format

Figure 9-7 on page 191 depicts the Ping and Ping Response packets. The host controller delivers the Ping Packet, which is routed to the target device. When the Ping packet arrives the address is checked but the endpoint number and direction bits are ignored. The entire token (Address, EP number and Direction bit) is copied to the Ping Response packet that the device returns to the host controller.

Figure 9-7: Ping and Ping Response Packet Format



# **USB 3.0 Technology**

---

## **Isochronous End-to-End Protocol**

Isochronous transactions begin the same as bulk and interrupt transactions, with the host controller initiating ACK headers for IN transactions and DATA headers for OUT transactions. However, Isochronous transactions do not include handshake packets to confirm the validity of data.

Isochronous Data Bursting supports either 1, 2, or 3 bursts within a single service interval and each burst may transfer up to sixteen 1 KB Data Packets if supported by the endpoint. This yields a maximum transfer size of 48 Data packets (48 KB) per service interval when three maximum sized bursts are performed.

---

## **General Isochronous IN Protocol Rules**

The following requirements apply to Isochronous transfers:

- The first data packet sent during a service interval always begins with sequence number zero.
- Sequence numbers for subsequent data packets continue to increment in order from 0 - 31 and on the 32nd packet the sequence rolls over to zero and may continue to increment up to sequence number 15 (the largest transfer size permitted, 3 bursts x 16 packets x 1024 bytes/packet).
- The last Data Packet of a burst transactions must have the last packet flag (**lpf**) set.
- Isochronous protocol allows data packets to be less than the maximum size including a zero data payload.
- When a data payload is larger than the maximum payload size then maximum sized payloads must be used, with the exception that the last packet of the burst may be less than the maximum payload.
- No flow control or retries are permitted

---

## **Isochronous IN Protocol**

This section details the protocol associated with Isochronous IN transactions and illustrates variations that are allowed.

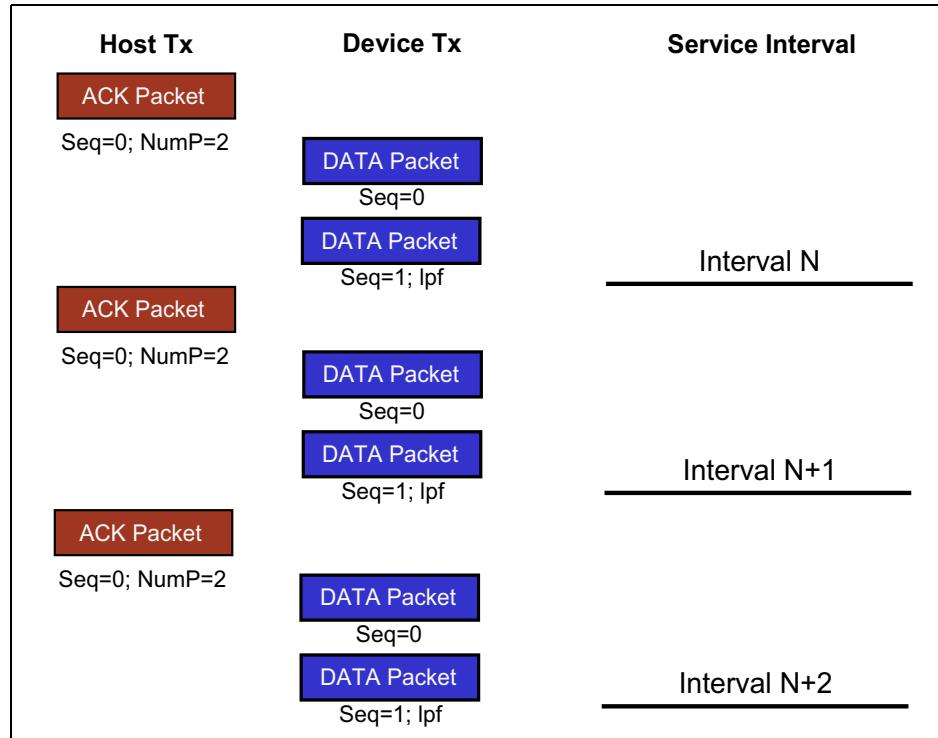
# Chapter 9: Isochronous Protocol

## Isochronous IN with Burst of Two

Figure 9-8 on page 193 illustrates a transfer with three service intervals and the isochronous endpoint supports a Maximum Burst of two DATA packets. In this example, the transaction is performed as follows:

1. The host controller delivers an ACK packet with a requested sequence number of zero and NumP=2 that initiates the IN Isochronous transaction.
2. In response to the ACK header, the endpoint returns a burst of two data packets. The first data packet must use the sequence number of zero that was advertised in the ACK packet and have a maximum payload size of 1024 bytes. Note that the last data packet may have a payload that is smaller than the maximum payload size.
3. The second packet must use the next consecutive sequence number of 1 and because its the last packet of the burst, the **last packet flag** (lpf) must be set.
4. The following two service intervals use the same protocol as the first.

Figure 9-8: Isochronous IN Sequence with Bursts of Two



# USB 3.0 Technology

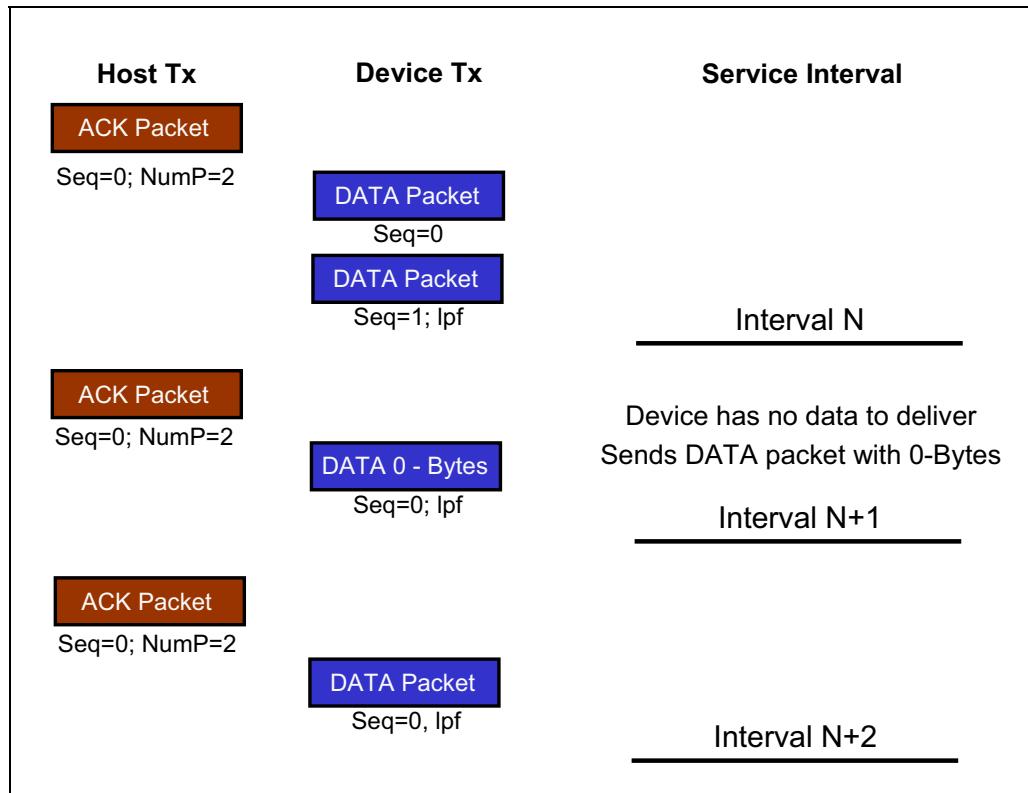
## Isochronous IN Burst of Two with Zero Payload

This example shows a variation permitted by the isochronous protocol. The sequence is the same as the previous example for service interval N but includes a zero payload during Interval N+1 and a single data packet for Interval N+2.

During Interval N+1 the endpoint returns a Data packet with Sequence number zero, a data payload of zero and the lpf set indicating the last packet of Interval N+1. This notifies the host that no samples are available for transmission and that the burst transfer is continuing.

During Interval N+2 a single data packet is delivered, with of course the required sequence number zero indicating the first packet and lfp indication end of the current burst. The non-zero data payload could be from 1 to 1024 bytes.

Figure 9-9: Isochronous IN with Burst of Two, Zero Payload, and Burst of One

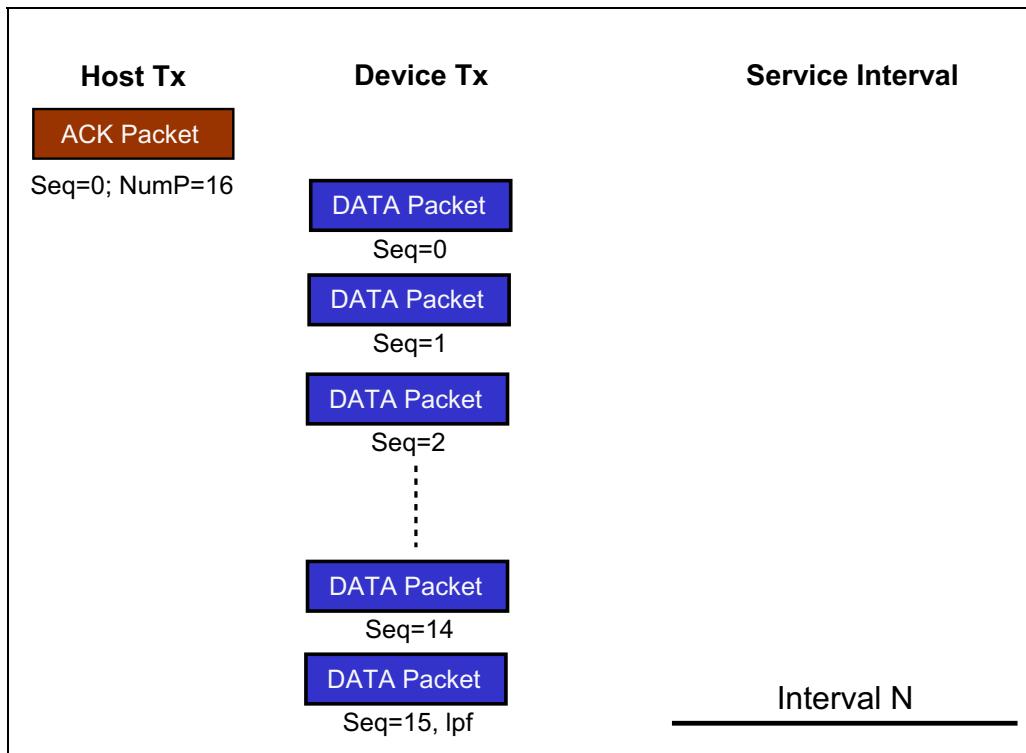


## Chapter 9: Isochronous Protocol

### Isochronous IN Burst Size of 16

This example illustrates the maximum burst of 16 DATA packets. The ACK packet advertises sequence zero and indicates a request of 16 Data packets. The endpoint returns the first packet with the required sequence number zero, and sends subsequent packets with the numbers in sequence. Packets 0-14 must use the maximum payload of 1024 bytes, while Packet 15 may have a payload size ranging from 1-1024 bytes. Packet 15 must also have it's lpf set.

Figure 9-10: Isochronous IN with Maximum Burst Size



# USB 3.0 Technology

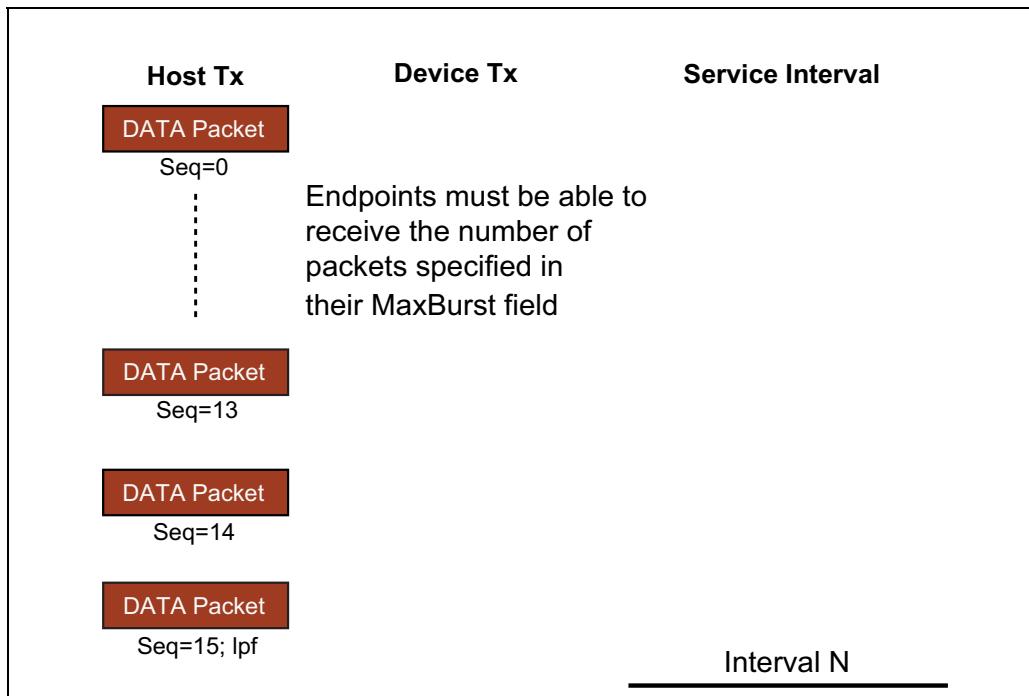
## Isochronous OUT Transactions

### Isochronous OUT with Burst Size of 16

Figure 9-11 illustrates a maximum burst size of 16 DATA packets. The protocol is based on the same principles used during IN transactions, except the host controller delivers the data to the isochronous endpoint. The requirements include:

- The first DATA packet must have a sequence number of zero
- Each successive DATA packet must increment the sequence number
- The last packet of the burst must have the lpf set to one

*Figure 9-11: Isochronous OUT Burst of 16*



# Chapter 9: Isochronous Protocol

## Isochronous OUT Burst with Four Service Intervals

The example depicts a sequence of four service intervals with different characteristics. The endpoint supports a burst of two.

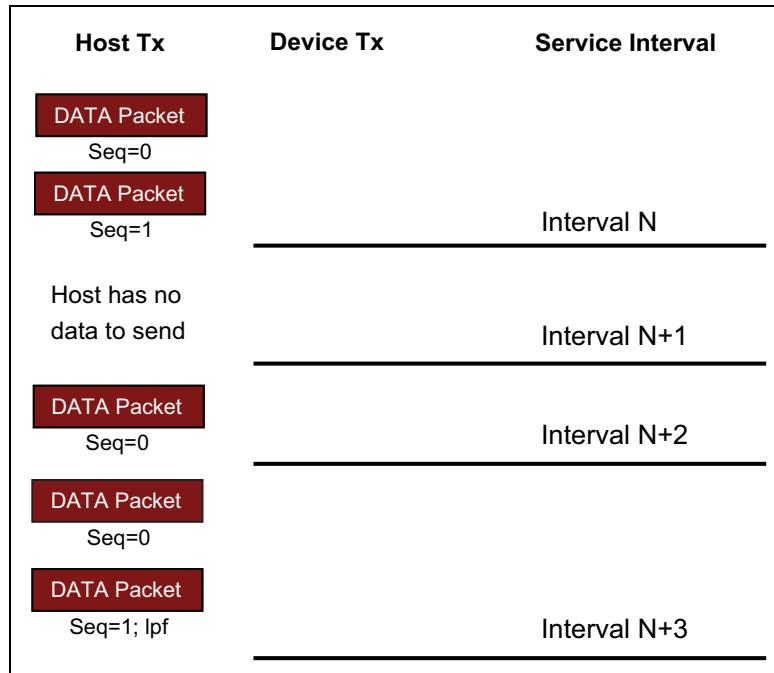
**Service Interval N.** During service interval N The host controller delivers the first DATA packet with sequence number zero and the second packet with sequence number one. As required, the second DATA packet also has the lpf set to one.

**Service interval N+1.** During service interval N there is no data for the host controller to deliver, so not DATA packet is sent.

**Service Interval N+2.** During Service Interval N+2 the host controller only has a single DATA packet to sent with Seq=0 and lpf=1.

**Service Interval N+3.** Finally, in service interval N+3 the host controller has two DATA packets. The first DATA packet with Seq=0 and the second DATA packet with Seq=1 and lpf=1.

Figure 9-12: Isochronous OUT with Four Service Intervals



# USB 3.0 Technology

---

## Smart Isochronous Transaction Scheduling

Isochronous transactions are allowed to be scheduled any time during the service interval. This makes it difficult for the endpoint to manage transitions to the low-power link states of U1/U2, because it doesn't know the host controller's behavior. Smart Isochronous Scheduling helps by communicating to endpoints the number of bus intervals that will be idle before the next Isochronous Transfer is performed. To accomplish this, new fields are defined in the ACK packet header (for IN transactions) and the DATA packet header (for OUT transactions).

Figure 9-13 highlights the fields within the ACK and DATA packet headers used by the host controller to manage smart Isochronous scheduling. Table 9-1 lists and describes the fields used for Smart Isochronous Scheduling.

Figure 9-13: ACK/DATA Header Fields Used to Manage Smart Isochronous Scheduling

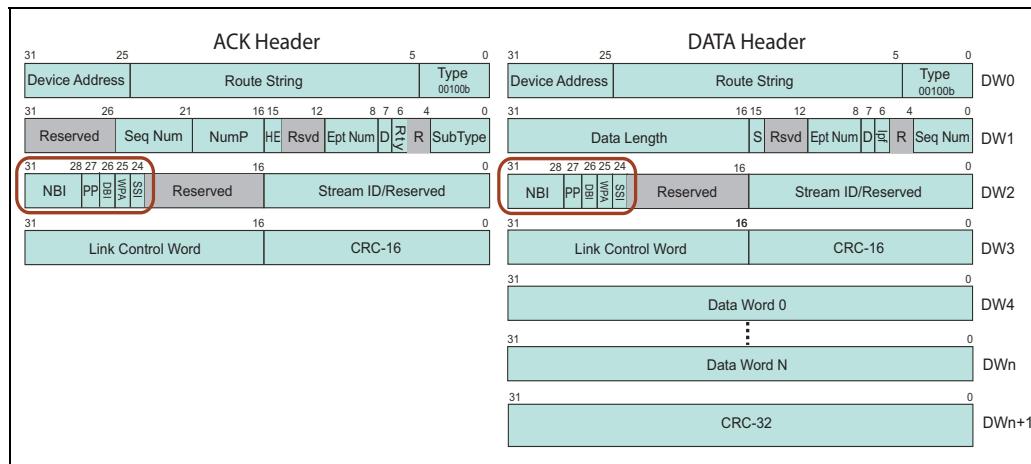


Table 9-1: Descriptions for the Smart Isochronous Fields

Field	Size	Name	Description
SSI	1 bit	Support Smart Isochronous	SSI=1 — Indicates support for Smart Isoc Scheduling for Isoc endpoints. During OUT transactions this bit can be set only if lpf is zero. During IN transactions this bit is ignored if the endpoint returns data with lpf set to one. NBI, DMI and WPA may be selectively enabled by SSI. SSI=0 — Indicates no support for Smart Isoc Scheduling.

## **Chapter 9: Isochronous Protocol**

---

*Table 9-1: Descriptions for the Smart Isochronous Fields (Continued)*

Field	Size	Name	Description
<b>WPA</b>	<b>1 bit</b>	Will Ping Again	WPA=1—is only valid when SSI=1. This bit notifies Isoc endpoints that the host controller will send a PING packet before servicing the endpoint again. WPA=0 — No PING packet scheduled.
<b>DBI</b>	<b>1 bit</b>	Data in current Bus Interval is finished	DBI=1—is only valid when SSI=1. This bit indicates the host controller will have finished all transactions for the current bus interval when the current transaction completes. When WPA=1, the endpoint can ignore DBI notification because the host controller will send PING to the device before continuing transactions.
<b>NBI</b>	<b>4 bits</b>	Number of Bus Intervals	NBI is only valid when SSI=1, WPA=0 and DBI=1. NBI value is zero based and specifies the number of consecutive bus intervals during which the host controller will not send transactions to this endpoint. The host controller will continue servicing the endpoint in the bus interval as follows: current bus interval (n) + value in NBI (0-15) + 1. For example, assume the current bus interval is 34d and NBI is 9, then the host controller would resume transmission during bus interval 44.

---

### **Smart Isochronous IN Transaction Example**

The following sequence depicts the behavior of the host controller and isochronous endpoint when Smart Isochronous Scheduling is employed. (Figure 9-14 on page 200.) The endpoint has a service interval of 8. The events are as follows:

**Interval N-1**— PING packet is send by the host controller to transition each link in the path between the root port and the Isoc device to U0.

**Interval N**— IN burst initiated by ACK header that specifies the following fields:

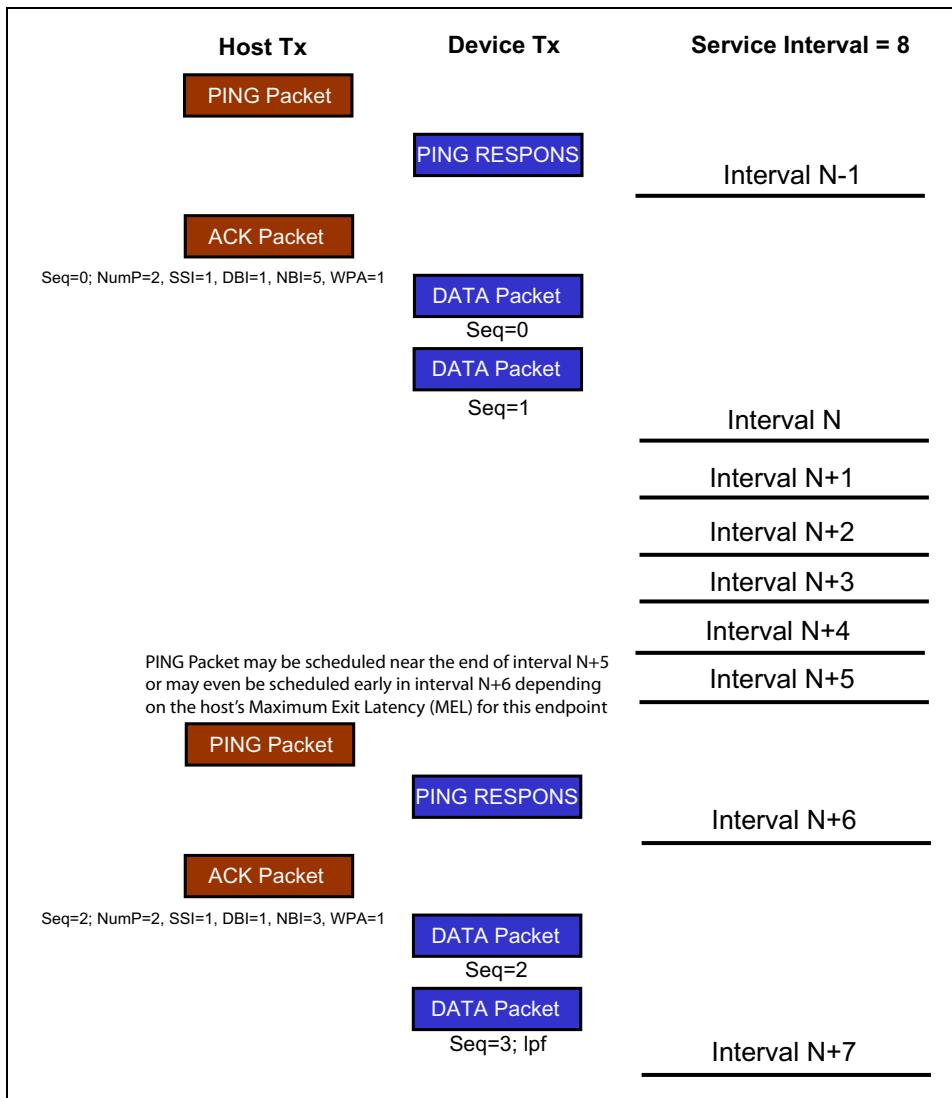
- Seq=0 — Start Seq Number =0
- NumP=2 — host controller is requesting 2 DATA packets
- SSI=1, — Smart Scheduling of Isoch transactions is supported
- DBI=1 — No more transactions in current bus interval after current burst
- NBI=5 — No transactions to endpoint during the next 6 bus intervals
- WPA=1 — Host controller will Ping again before initiating transactions

## USB 3.0 Technology

**Interval N+1 through N+6** — No transactions are performed during these bus intervals. A Ping packet is also performed during bus interval N+6.

**Interval N+7** — This is the last transfer associated with this service interval as indicated by the lpf being set in the last DATA packet.

*Figure 9-14: Isochronous IN Smart Scheduling Example with Pings*

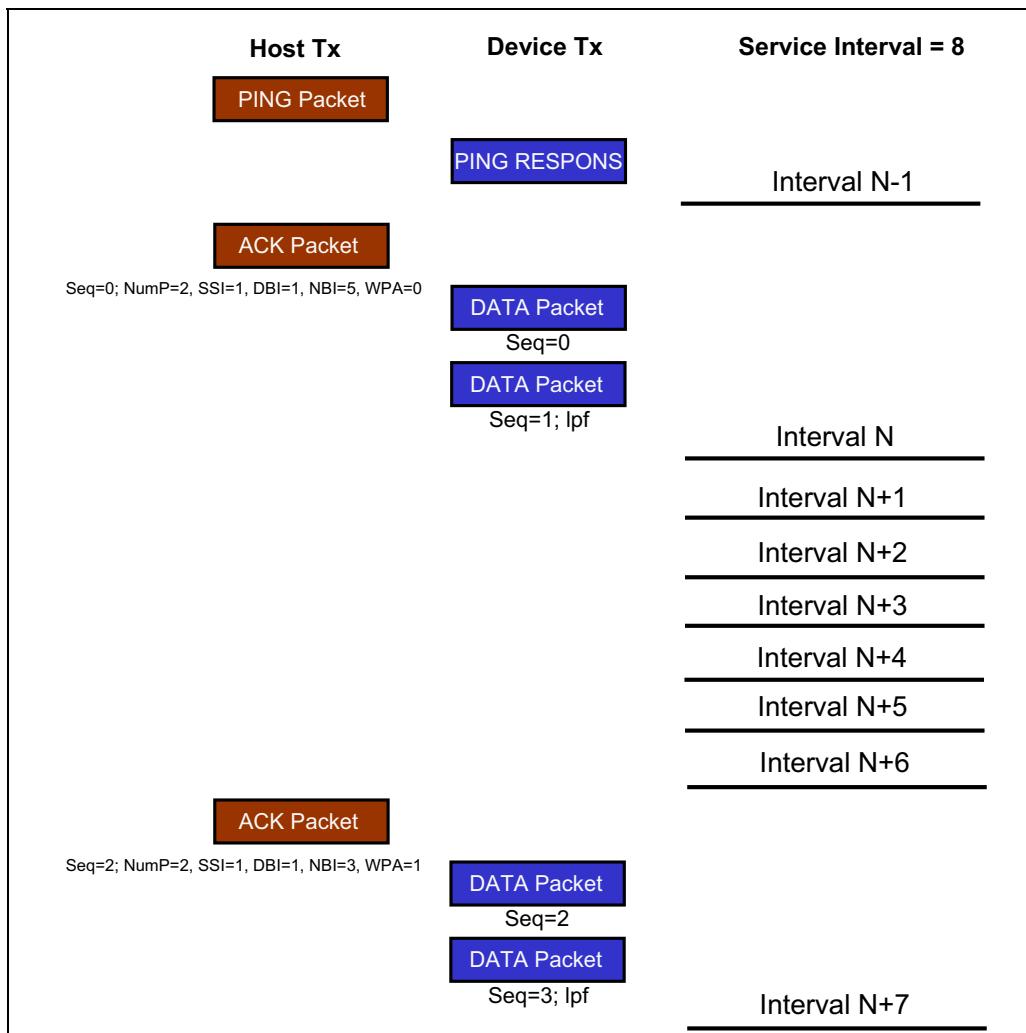


## Chapter 9: Isochronous Protocol

### Smart Isochronous IN Transactions with No Ping

Figure 9-15 illustrates an Isochronous IN Transactions that has a service interval of eight. This transaction is identical to the previous example except that no PING packet is send during the service interval as indicated by WPA=0 being set in Interval N. Note that Interval N+7 has WPA=1, indicating the host controller will Ping so start the next service interval.

Figure 9-15: Smart Scheduling Associated with Isochronous IN Transfers



## **USB 3.0 Technology**

---

### **Smart Isochronous OUT Transactions**

Figure 9-16 on page 203 emphasizes some of the Smart Scheduling feature associated with OUT Isochronous transactions during a service interval.

**Interval N-1**— The PING packet is send by the host controller to transition each link in the path between the root port and the Isoc device to U0.

**Interval N**— OUT Isochronous burst is initiated by host controller delivering a DATA packet. The Smart Isoc fields are described for DATA packet zero below:

- Seq=0 — Indicates the Start Seq Number for the DATA packet
- SSI=1, — Smart Scheduling of Isoch transactions is supported
- DBI=0 — Following the current DATA packet, more transactions are scheduled within the current bus interval.
- NBI=x — Field is invalid when DBI=0.
- WPA=0 — Host controller will not Ping before sending another transaction.

The second DATA packet sent by the host controller provides additional information regarding Smart Scheduling:

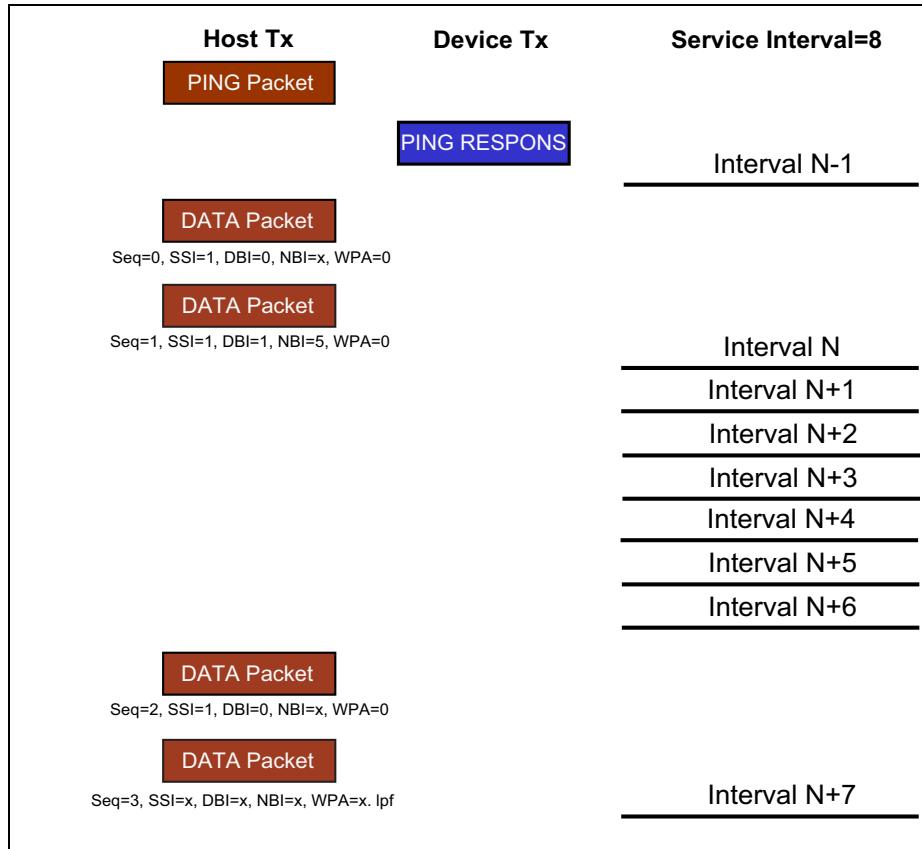
- Seq=1 — Indicates the Start Seq Number for the DATA packet
- SSI=1, — Smart Scheduling of Isoch transactions is supported
- DBI=1 — Following the current DATA packet no more transactions are scheduled within current bus interval.
- NBI=5 — This valid specifies that no transactions will be scheduled to the endpoint during the following six bus intervals.
- WPA=0 — Indicates the host controller will not Ping before initiating additional transactions to this endpoint.

**Interval N+1 through N+6** — No transactions are performed during these bus intervals. A Ping packet is also performed during bus interval N+6.

**Interval N+7** — The first DATA packet in interval N\_7 has the same Smart Scheduling bits set as in Interval N. The second DATA packet has the lpf field set, so the service interval has ended, so the Smart Scheduling bits are irrelevant.

## Chapter 9: Isochronous Protocol

Figure 9-16: Isochronous OUT Transactions with Smart Scheduling Example



## **USB 3.0 Technology**

---

---

# **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.

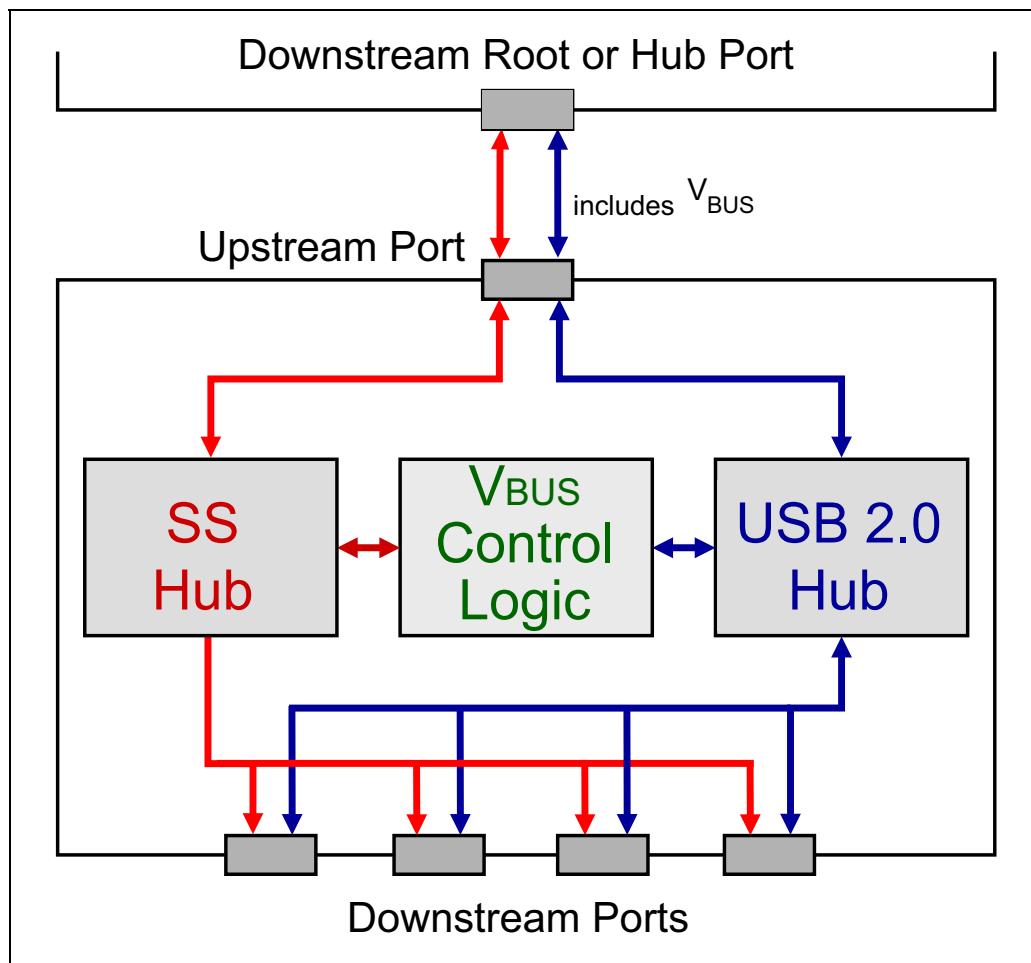
The upstream port of a hub attaches directly to a root port or to the downstream port of another hub. The upstream port must attach to both the SuperSpeed bus and the high-speed USB 2.0 bus when possible. The USB cable delivers  $V_{BUS}$  power that is shared by the SuperSpeed and USB 2.0 sides of the hub.  $V_{BUS}$  con-

## USB 3.0 Technology

trol 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<sub>BUS</sub> 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**

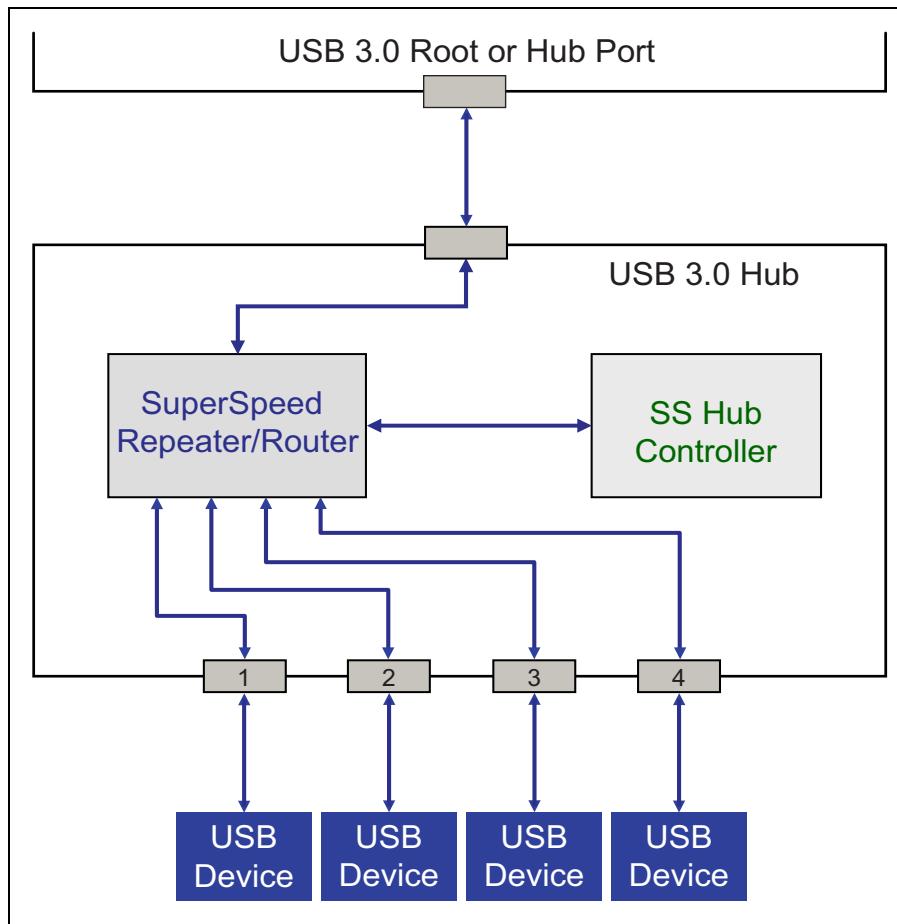
One of the primary functions associated with Hubs is to forward packets in both directions while performing end-to-end transfers between the root port and target devices. Figure 10-2 on page 208 illustrates this requirement. Packets moving in the downstream direction must be routed by the hub to support 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)

## USB 3.0 Technology

Each packet moving in the downstream direction includes a routing field that provides the instructions for forwarding the packet to a specific port (except for broadcast packets). The next section details this unicast routing mechanism.

*Figure 10-2: Hub Connectivity – Downstream and Upstream*



Packets received on a downstream hub port are simply forwarded to the upstream port. To manage the flow of traffic moving across hubs, the specification defines a minimum set of buffers. (See Table 10-8 on page 214.)

# Chapter 10: USB 3.0 Hubs

## Packet Forwarding and the Layers

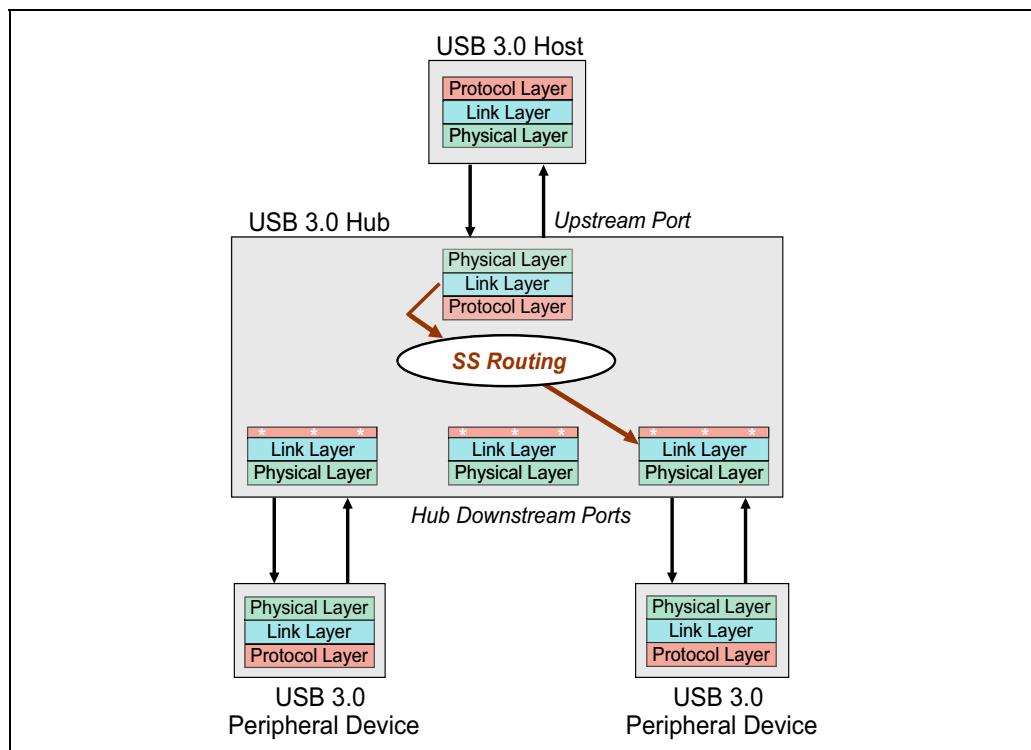
This section describes three examples associated with the layered implementation and routing. The examples include:

- Packet Routing directly across the hub
- Packet Routed to hub controller that accesses a local port (e.g., port status)
- Packet routed to the hub controller that results in an LMP being sent to the attached device.

### Packet Routing Across Hub — Link-to-Link

Figure 10-3 depicts a hub with the layers exposed. Notice that packets being forwarded do not involve the protocol layer because the hub controller is not the recipient. Instead, packets are transferred from link layer to link layer. This applies to packets moving in both directions.

Figure 10-3: Packet Routing — Link Layer to Link Layer



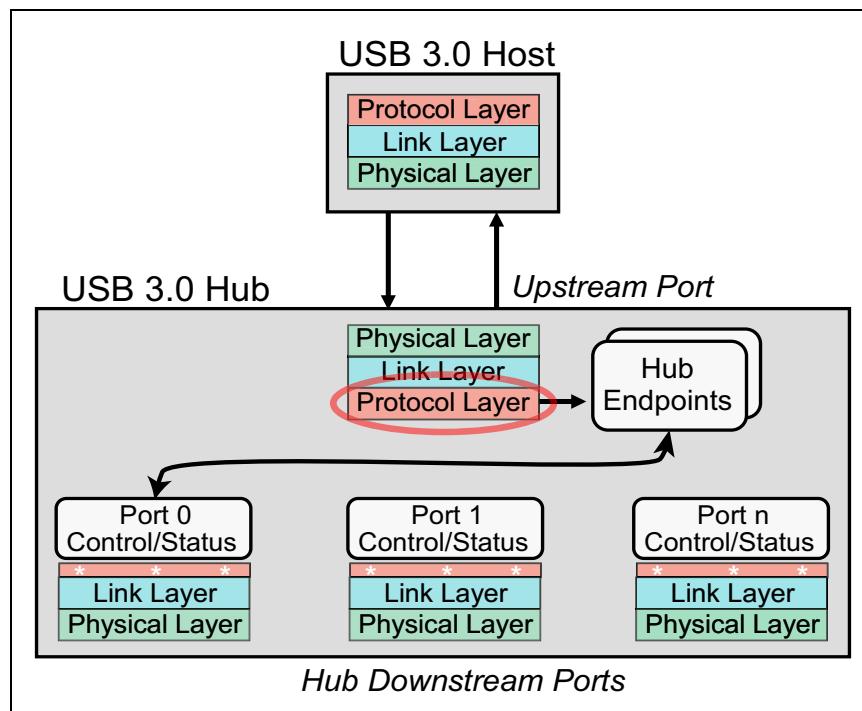
# **USB 3.0 Technology**

---

## **Packet Routing to Hub Controller**

Figure 10-4 illustrates a packet being routed to the hub where the transaction terminates. When the packet targets the local hub controller, the packet is forwarded to the protocol layer. The transaction may target the hub controller directly (e.g., SetHubAddress) or may target one of the downstream ports (e.g., GetPortStatus). This example illustrates a hub port 0 being accessed.

*Figure 10-4: Transaction Targeting Hub*

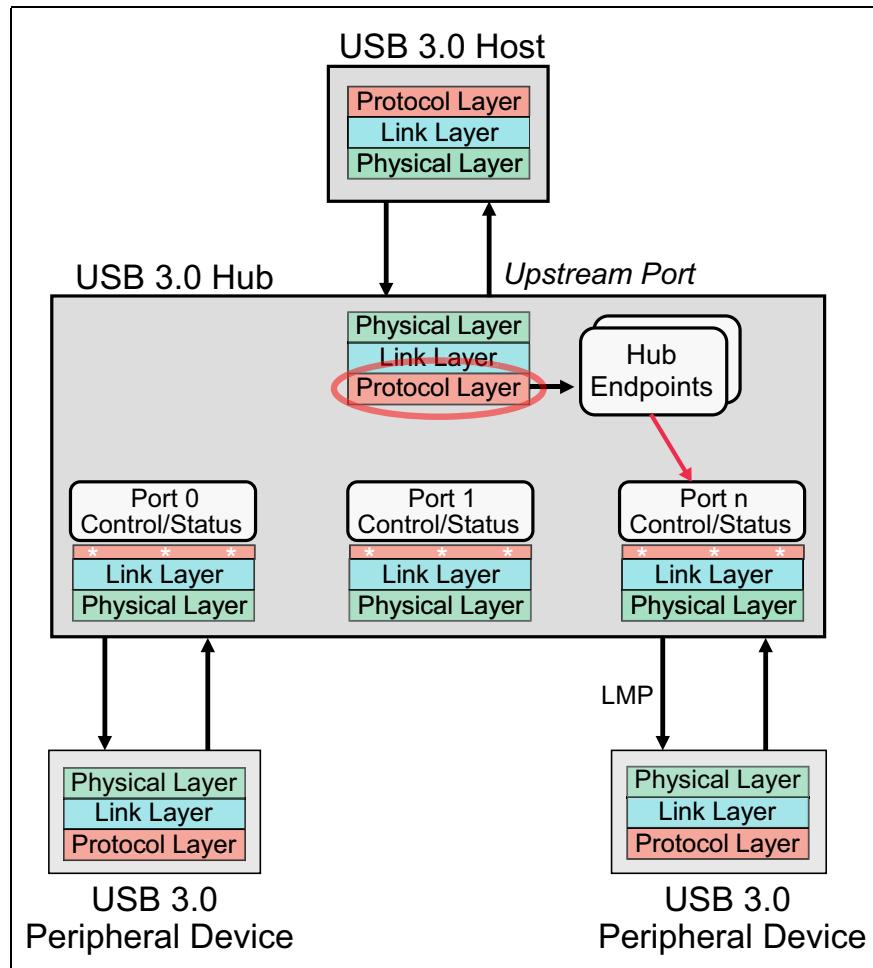


## **Packet Routing to Hub Controller with LMP**

The final example illustrates a packet being routed to the hub controller. This packet triggers the target port to deliver a Link Management Packet (LMP) to the downstream device. (See Figure 10-5 on page 211.) There are only a few LMPs and each will be covered in later chapters. The LMPs primarily relate to Link Power Management, Port Configuration and Testing.

## Chapter 10: USB 3.0 Hubs

Figure 10-5: Transaction Targeting Hub with Port LMP



# **USB 3.0 Technology**

---

## **Unicast Routing**

SuperSpeed USB employs a Unicast routing mechanism to direct packets from a root hub port, across one or more USB 3.0 hubs and finally to the target device. Host software has knowledge of the topology and builds the routing structure for each device detected. The routing structure consists of five nibbles of information called the Route String, and is located in DW0 of all header packets that are moving in the downstream direction (see Figure 10-6 and Figure 10-7 on page 213). Each nibble represents routing information for up to 5 hubs that can reside between the root hub port and the furthest device downstream. Each nibble provides 16 possible combinations with values defined as follows:

- 0h = Upstream Hub Port is Target — this means that the hub itself is the recipient of the header packet
- 1h - 15h = One of 15 Downstream Hub Ports is the target — the header packet is directed to the target port specified and sent to the attached hub or peripheral device.

This routing string implementation limits all USB hubs to no more than 15 downstream ports.

Each Hub must know the depth at which it resides in the topology so it can identify which nibble of the routing table it should use. Host software assigns the Hub Depth information during device configuration. A Hub Depth Value of 0-4 is assigned to each hub in the topology using the SetHubDepth request. This permits hubs to index into the proper nibble within the route string by multiplying the Hub Depth Value x 4.

*Figure 10-6: Route String*

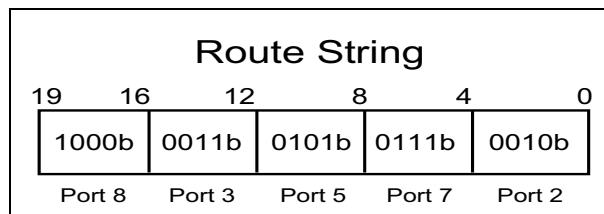
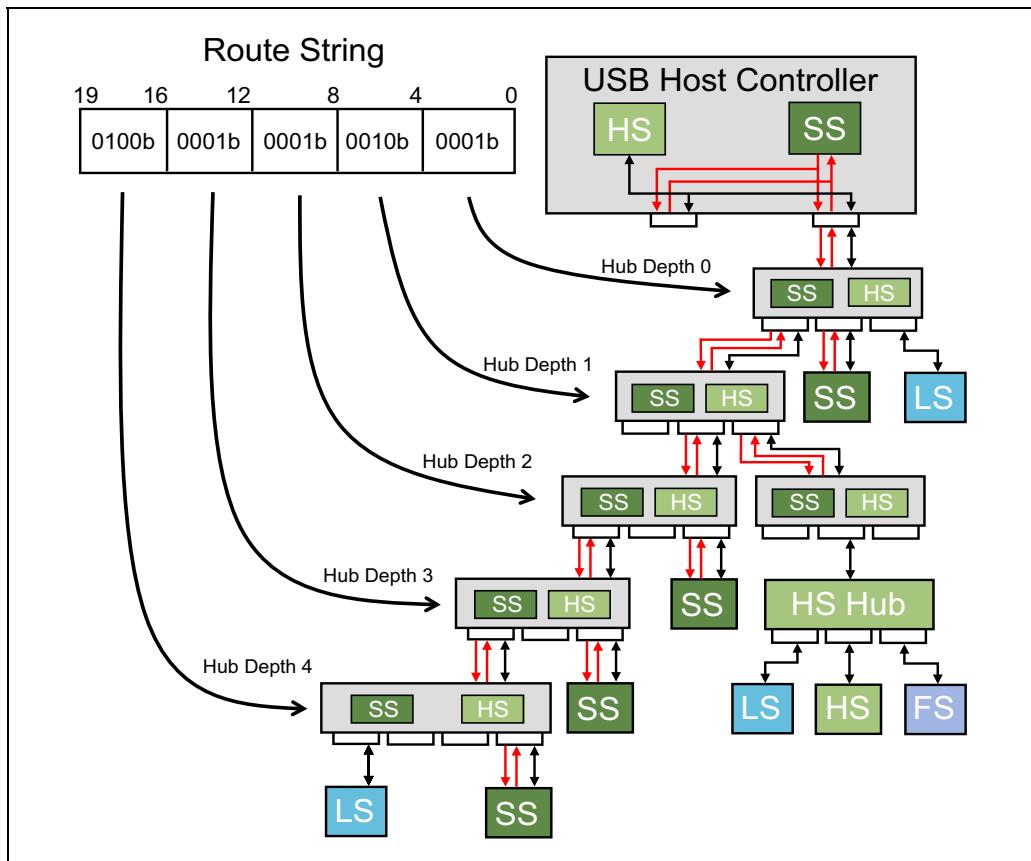


Figure 10-7 illustrates a routing sequence that spans the maximum of 5 hubs. This example assumes that hub ports are numbered consecutively from left to right starting with port 1. Each hub that receives the header packet indexes into the route string to identify the port that will deliver the header packet further downstream.

## Chapter 10: USB 3.0 Hubs

Figure 10-7: SuperSpeed Unicast Routing Example



### Packet Forwarding and Buffer Requirements

USB 2.0 implements a repeater model where all packets and data payloads are simply repeated in one direction at a time across the hub. In addition, USB 2.0 has a single differential pair that is used in a half duplex implementation. SuperSpeed of course uses two differential pairs (one transmit and one receive) to support burst transfers. In addition SS hubs use two packet forwarding implementations discussed in the following sections.

1. **Store and Forward Model - Header Packets**
2. **Repeater Model - Data Payloads**

# USB 3.0 Technology

## **Header Buffers — Store and Forward Model**

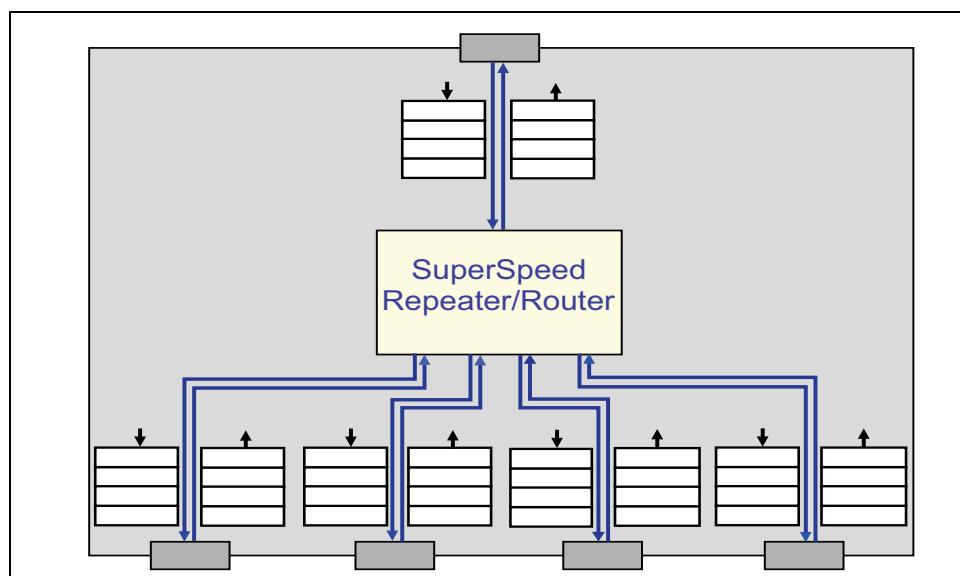
Hubs implement the store and forward model for all buffer Headers -- this includes buffering in the receive and transmit port buffers. The motivation for this approach includes:

- Ability to support asynchronous message traffic (e.g., ERDY) from multiple devices while handling end-to-end transaction traffic, including bursting.
- Handle delays associated with target links that may be in a power managed mode.
- Support for port-to-port protocols, such as flow control and header packet retries (discussed in subsequent chapters).

Figure 10-8 depicts buffers used by the 16 byte header packets. Each port interface must implement four receive buffers and four transmit buffers primarily for handling transaction bursting. A variety of requirements are specified:

- The upstream port must forward header packets immediately to the target port, if there is room in the buffer.
- SS hubs must hold 8 header packets targeting the same DS port when starting with buffers empty and flow control credits must not run out.
- Header packets at an ingress (receiving) port must be transmitted in the same order by the egress (transmitting) port.

*Figure 10-8: Buffers Used When Forwarding Packets*



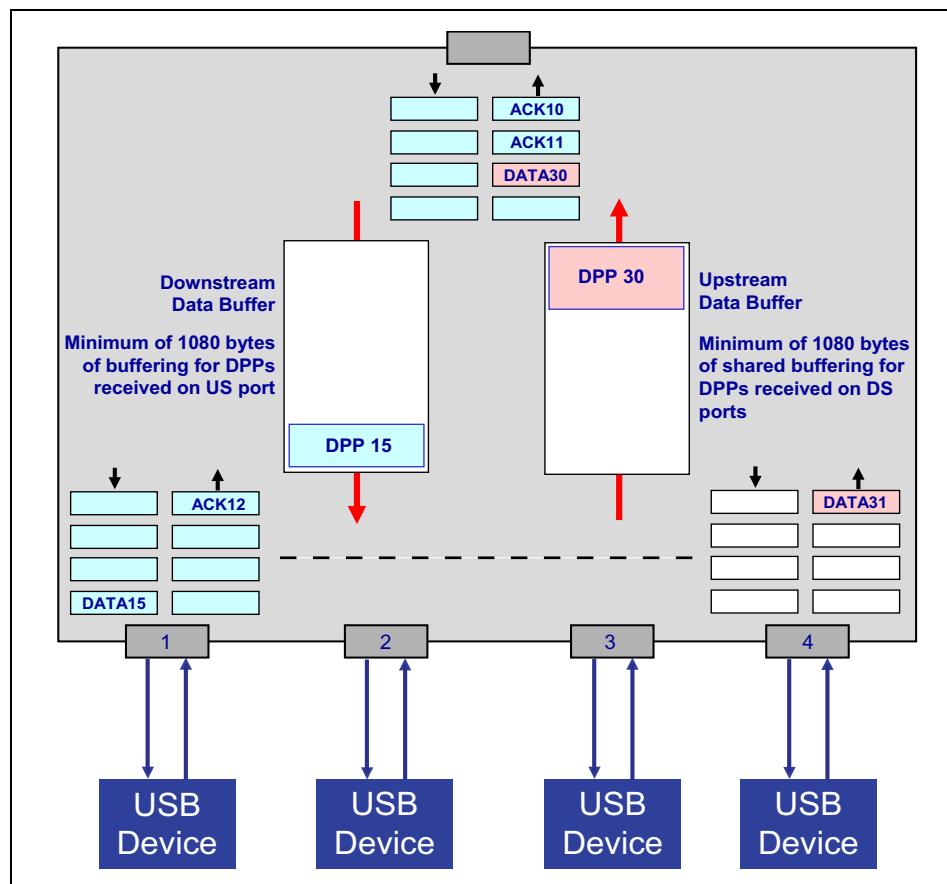
## Chapter 10: USB 3.0 Hubs

### Data Payload Buffers — The Repeater Model

Data Packet Payloads (DPPs) associated with DATA Header packets use a repeater implementation, but some buffer space is required. During data bursting the payload size is 1024 bytes and would require a large amount of buffering within a hub if the entire payload was buffered. Instead, when a Data header packet arrives at the receiving port it is forwarded immediately to the transmit buffer for delivery across the next link. If the header is not yet being sent, a small portion of the data payload will require buffering prior to transmission.

Figure 10-9 pictures an OUT transfer being performed on port 1 and an IN transfer being performed on port 4 both showing a small amount of buffering.

Figure 10-9: DATA Payload Buffering Required by Specification



## **USB 3.0 Technology**

---

### **Transaction Deferral**

A key aspect of the SuperSpeed implementation is conservation of power. This is primarily accomplished by aggressively placing links into an electrical-idle state, called standby. Consequently, when a transaction is routed to a link in the standby state the transaction is delayed until the link can recover back to the active state. In addition, other transactions pending delivery may also stall because of the inactive link.

To improve performance associated with inactive links, a variation in the protocol, called Transaction Deferral is employed. The result of Transaction Deferral is that the stalled transaction will be temporarily suspended, allowing other transactions to complete while the inactive link recovers.

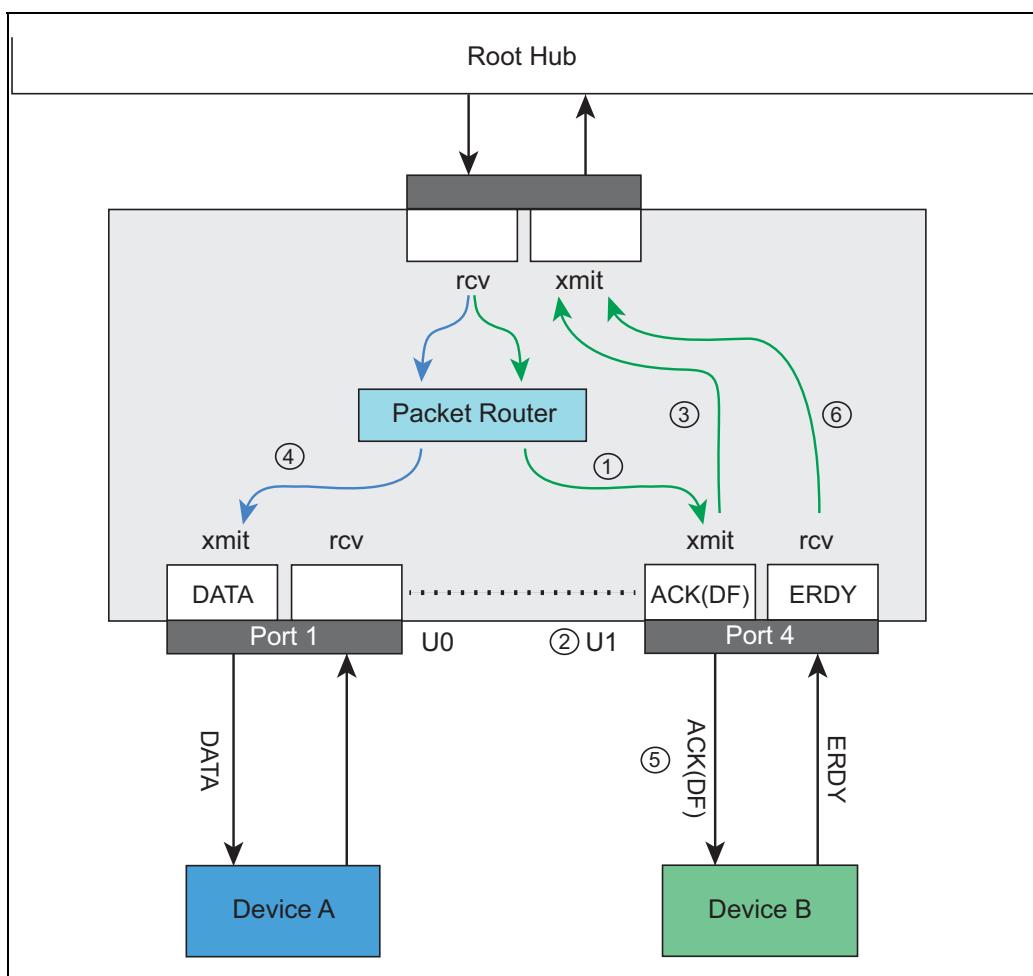
Figure 10-10 on page 217 depicts a transaction deferral scenario and summarizes the deferred operation performed by the hub. The actions described in the illustration are discussed below:

1. The Host Controller (HC) initiates an IN transaction by sending an ACK packet to Device B. The packet arrives at the Hub and determines that the packet should be routed to Port 4.
2. Because Port 4 is in an inactive state (U1), when the ACK packet arrives it is deferred by setting bit fields within the packet. Figure 10-11 on page 218 highlights the Deferred bit (DF), Delayed Bit (DL) and Hub Depth within the Link Control Word of the ACK packet.
3. The deferred ACK packet is then sent back to the Host Controller. Upon receiving the deferred packet, the Host Controller temporarily deactivates the IN transaction. This allows other pending transactions to be performed while the inactive link recovers back to the active state.
4. The Host Controller initiates an OUT transaction by sending a DATA packet to Device A. When the hub receives the DATA packet it is routed to Port 1 and delivered to the device.
5. Even though the IN transaction has been deactivated, the Deferred ACK packet is still held in the hub. When the link finally recovers back to the active state, the Deferred ACK packet is delivered to Device B. This allows the device to process the request and prepare to send data back to the host.
6. When the device is ready to deliver data it notifies the Host Controller by sending an Endpoint Ready (ERDY) Packet. Though not illustrated, when the current transaction to Device A completes, the HC initiates the IN transaction again. When the ACK packet is forwarded to Port 4 and an active link is discovered the ACK is delivered and a DATA packet is returned.

## Chapter 10: USB 3.0 Hubs

OUT transactions are also deferred under the same circumstances as described in the IN transaction example.

*Figure 10-10: Deferred Transaction Example*

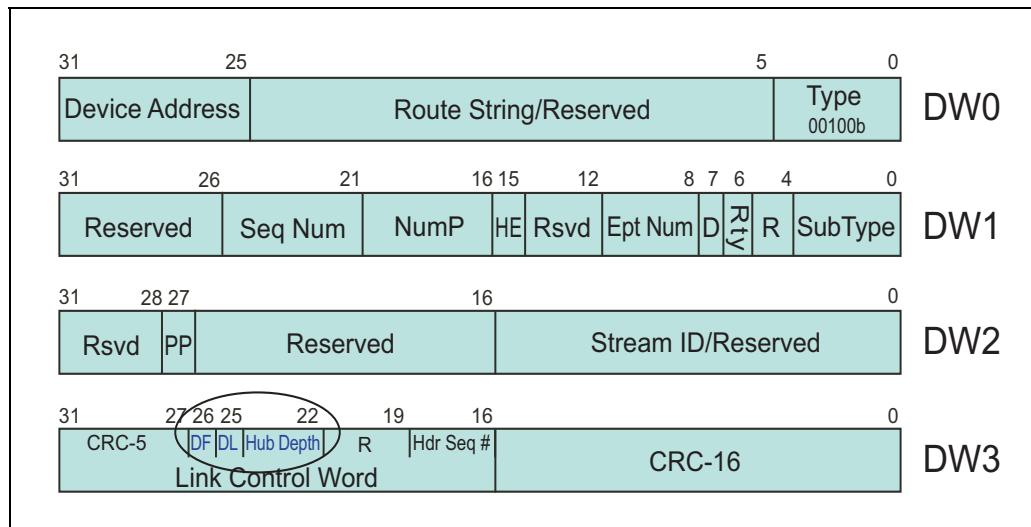


## **USB 3.0 Technology**

---

Figure 10-11 on page 218 highlights the fields that are set when a deferred header is returned to the host controller and the target device.

*Figure 10-11: Header Bits Set During Deferral*



---

## **Hub Error Detection and Handling**

Hub error detection focuses on receiving incoming Header packets and verifying successful reception by evaluating the CRC. In addition, every header delivered must be recognized within the logical idle data stream. This is managed by sending a sequence of unique patterns (ordered sets) called “Start Header Framing.” The Data Packet Payload (DPP) that follows a Data Header requires both “Start Framing and End Framing.”

The occasional Header packet error is typically managed by the Port-to-Port protocols via retransmission of the failed packet. These protocols are covered in subsequent chapters. A Data Packet Payload (DPP) also implements CRC, but no retransmission of the payload is allowed. This means that a DPP will likely be forwarded as a damaged packet, but there are a number of possible circumstances and outcomes described below:

- A Header CRC error is reported back to the transmitter and the Header Packet is retried. If a DATA Header and DPP is received and the DATA

## **Chapter 10: USB 3.0 Hubs**

---

Header incurs a CRC error, the DATA Header is retried causing the DPP to be discarded because retransmitting a DPP is not allowed.

- Unrecoverable Start Header Packet framing results in the Header packet being lost. This results in a timeout and the Header packet is retried. If a DPP follows the header, it is lost.
- Unrecoverable DPPSTART framing results in the DPP being dropped and the Data Header will be forwarded alone.
- An error within DPP causes the DPP to be aborted immediately followed by End Bad Packet framing.
- Unrecoverable DPPEND Framing causes the DPP to abort followed by End Bad Packet framing.
- Babbling is a condition in which a DPP exceeds the maximum payload. In short, if DPP starts and DPEND/DPPABORT framing not detected within 1030 symbols, the DPP immediately aborts followed by End Bad Packet framing.

---

### **Hub Link Power Management Responsibilities**

As described in the USB SuperSpeed overview chapter, Link Power has one full-power state and three low-power states defined as:

- U0 — Full power and operation
- U1 — Standby with fast recovery back to U0
- U2 — Standby with slow recovery back to U1
- U3 — Suspend with very long recovery time

Hubs play pivotal roles in Link Power Management that include:

- Implementing U1 and U2 inactivity timers on the downstream ports along with a U2 inactivity on the hub upstream port. These timers are set up by software and count idle time (inactivity) associated with the link. When a counter expires it triggers entry into the specified low power state. Details associated with these timers can be found in the section entitled, “Hub Inactivity Timers” on page 570.
- Hubs must also ensure that the link power hierarchy is maintained. This simply means that a hub must never allow its upstream link to enter a power state that is lower than any of its downstream links.

## **USB 3.0 Technology**

---

---

### **Cable Power and Distribution**

A hub can be implemented with bus-power only or may optionally include a power supply making it a self-power hub. Maximum bus power delivered from a self-powered USB SuperSpeed port is 900 ma at 5 vdc. The maximum current that can be drawn from a hub or device prior to being configured is 150ma. When a hub is initially connected the downstream ports have no power applied. During configuration, software uses the SetPortPower request to apply power to the ports, making it possible to detect the presence of any devices.

---

### **Reset Propagation**

Hubs are required to propagate reset as discussed in the section entitled, “Hubs Propagate Resets Downstream” on page 425.

---

---

# **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 its 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.

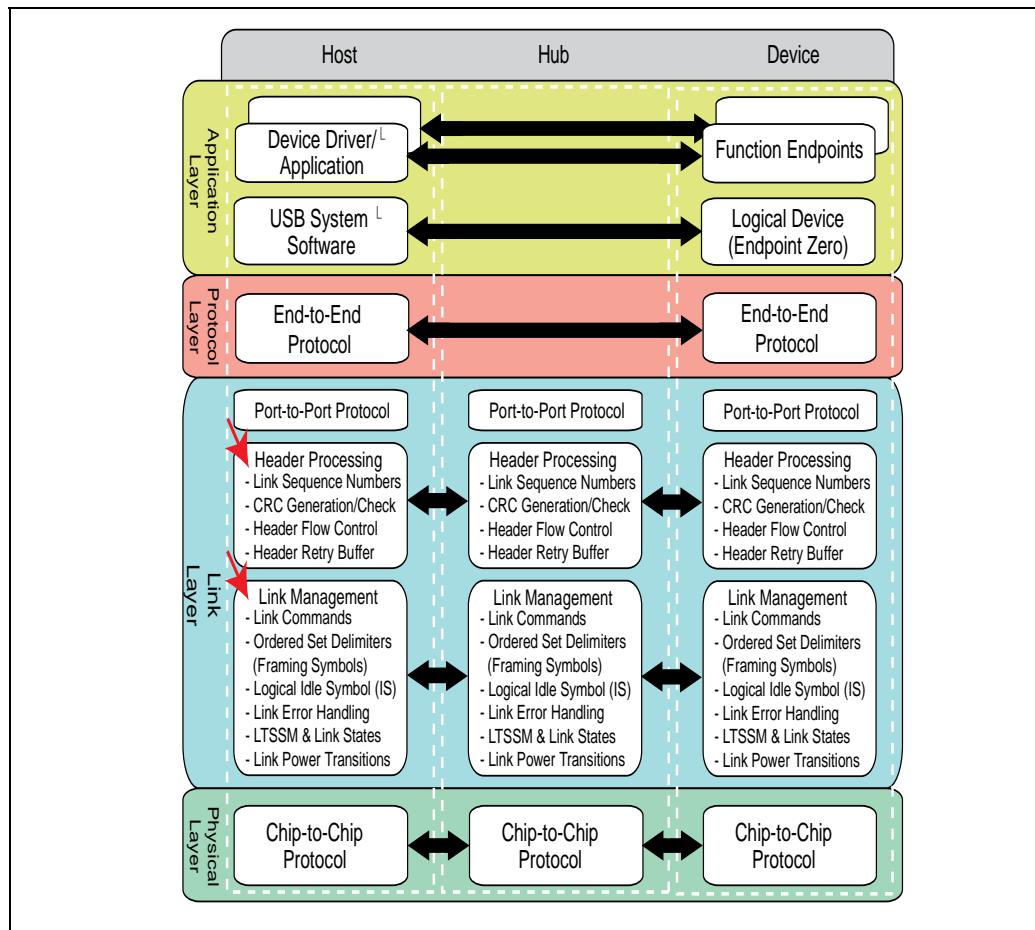
# USB 3.0 Technology

## 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



# Chapter 11: Introduction to Port-To-Port Protocol

## Port-To-Port Protocol: Header Processing

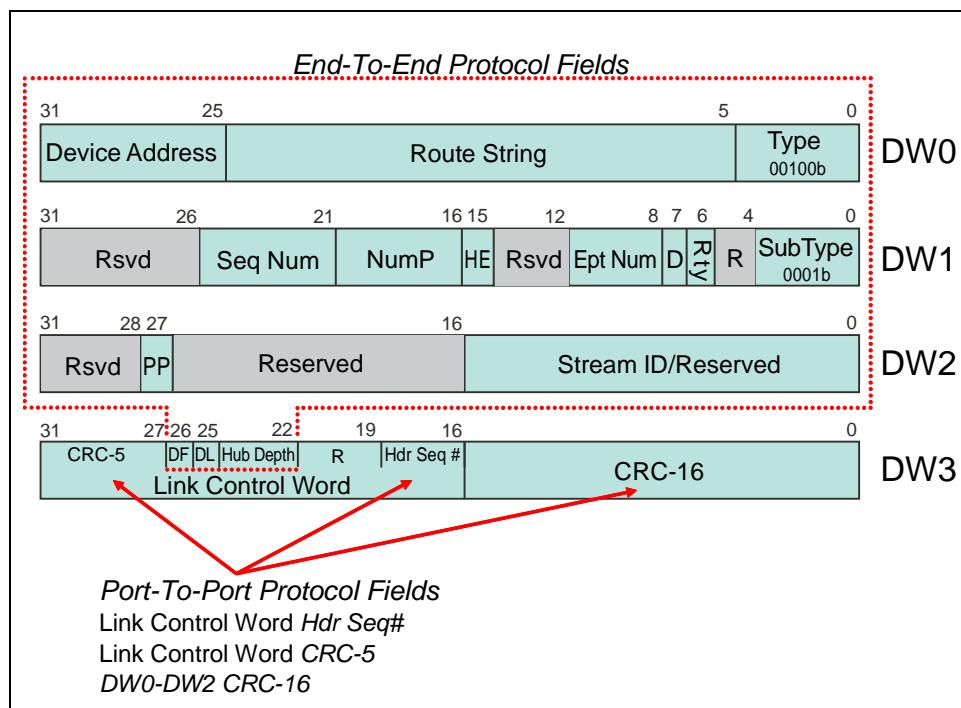
### General

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

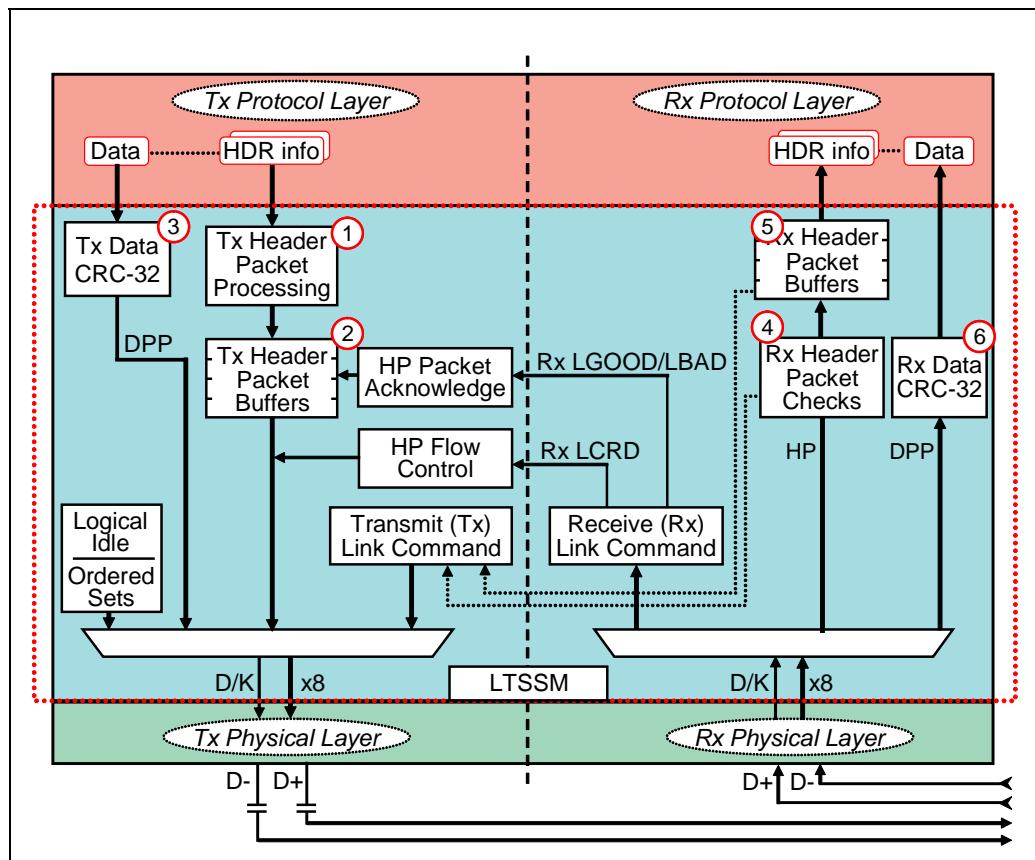


# USB 3.0 Technology

## Link Layer Header Processing Elements

The following section identifies the link layer elements related to header packet processing. Six key elements are tagged in Figure 11-3 on page 224. The role of each is summarized here and described in detail in subsequent chapters.

Figure 11-3: Link Layer Header Processing Elements



## **Chapter 11: Introduction to Port-To-Port Protocol**

---

### **Tx HP Processing**

Note (1) in Figure 11-3 on page 224. For each outbound header packet, the transmitter (Tx) is responsible for assigning a header packet sequence number, generating a CRC-5 and CRC-16, and using them to complete the 16-byte header sent from the protocol layer. The reason this is done at the link layer is because the link sequence number is specific to each link and, in the event that the header fails CRC checks at the receiver, it is the Tx link layer that will perform the header packet retry.

The details of header processing are covered in Chapter 14, entitled "Header Packet Processing," on page 303.

---

### **Tx HP Buffers**

Note (2) in Figure 11-3 on page 224. A copy of each completed 16-byte header is maintained in one of the Tx HP Buffers until it is acknowledged by the receiver. In the current USB 3.0 specification, there are four Tx HP Buffers to hold copies of header packets. Each time a header packet is acknowledged, the transmitter discards it from the buffer and reuses the buffer for another outbound header packet.

Details of Tx HP Buffer management, including initialization, acknowledgement, header packet retry, and LTSSM Recovery (in the case of unrecoverable errors), are covered in:

- Chapter 14, entitled "Header Packet Processing," on page 303
- Chapter 16, entitled "Link Errors & Packet Acknowledgement," on page 339
- Chapter 20, entitled "Link Training," on page 427
- Chapter 21, entitled "Link Recovery and Retraining," on page 467

---

### **Tx Data CRC-32 Generation**

Note (3) in Figure 11-3 on page 224. This functional block is only used if the outbound header is for a data packet. For each transmitted data packet, the data packet header (DPH) processing is the same as for other header packets. The DPH is immediately followed by the data packet payload (DPP). A four-byte CRC-32 is generated and sent to the physical layer immediately after the last byte of data. Additional details may be found in Chapter 14, entitled "Header Packet Processing," on page 303.

## **USB 3.0 Technology**

---

### **Rx HP Checks**

Note (4) in Figure 11-3 on page 224. For each inbound header packet, the receiver (Rx) link layer performs three header packet checks to verify error-free delivery. The first two checks are CRC-5 and CRC-16. If either are incorrect, the receiver generates a link command (LBAD) requesting the transmitter to retry the header packet. If both CRCs are correct, the header packet sequence number is checked. If valid, the header packet is accepted and an acknowledgement link command (LGOOD) is sent to the transmitter. If not, a transition to LTSSM Recovery is initiated to correct the problem.

Additional details on receiver header packet checks may be found in:

- Chapter 14, entitled "Header Packet Processing," on page 303
- Chapter 16, entitled "Link Errors & Packet Acknowledgement," on page 339
- Chapter 21, entitled "Link Recovery and Retraining," on page 467

---

### **Rx HP Buffers**

Note (5) in Figure 11-3 on page 224. Assuming that the header packet CRC and sequence number checks were successful, the receiver moves the header packet into the next sequential Rx HP Buffer before forwarding it up to the protocol layer. When the header packet moves out of the Rx HP Buffer and up to the protocol layer, a flow control credit link command (LCRD) is also sent to the transmitter.

Additional details of Rx HP Buffer management, including initialization, acknowledgement, header packet retry, and LTSSM Recovery (in the case of unrecoverable errors), are covered in:

- Chapter 14, entitled "Header Packet Processing," on page 303
- Chapter 16, entitled "Link Errors & Packet Acknowledgement," on page 339
- Chapter 21, entitled "Link Recovery and Retraining," on page 467

---

### **Rx Data CRC-32 Checking**

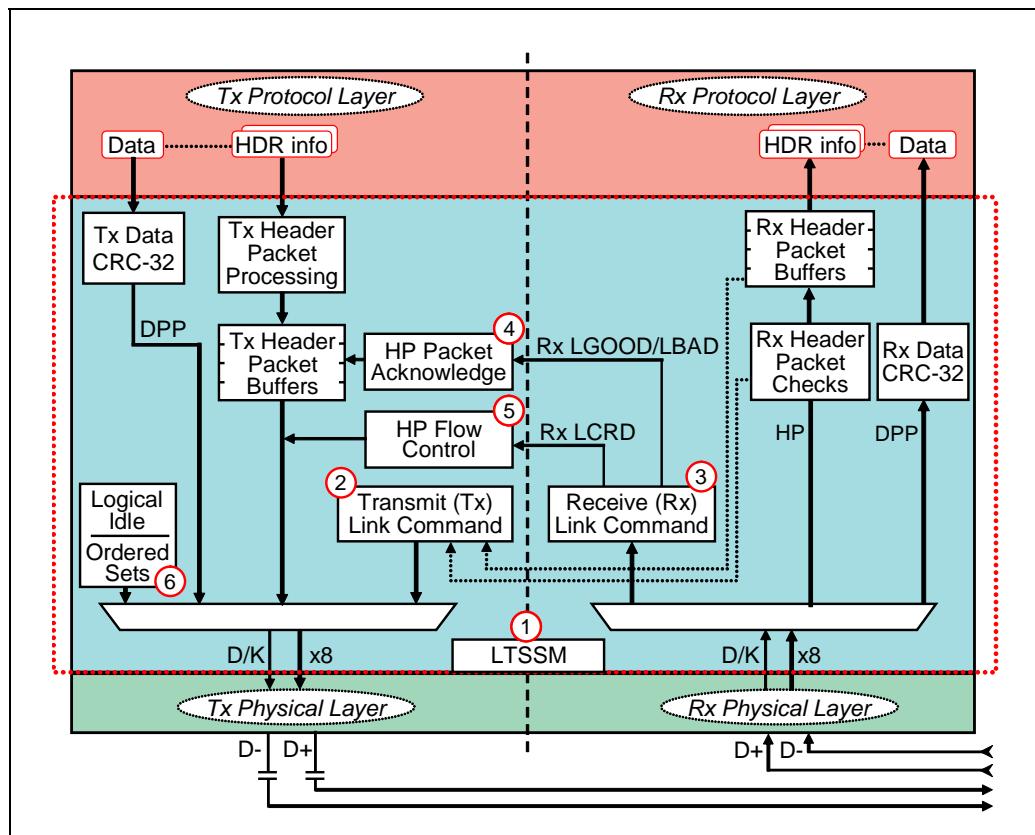
Note (6) in Figure 11-3 on page 224. This functional block is only used for inbound data packets. After checking the data packet header (DPH), the receiver compares the data packet payload (DPP) CRC-32 sent by the transmitter with the CRC-32 it calculates locally. The result (CRC-32 good or bad) is forwarded to the protocol layer. Additional details may be found in Chapter 14, entitled "Header Packet Processing," on page 303.

# Chapter 11: Introduction to Port-To-Port Protocol

## Link Management Elements

The following section describes the link layer elements related to Port-to-Port protocol link management. Six key elements are identified in Figure 11-4 on page 227. The role of each is summarized here and described in detail in subsequent chapters.

Figure 11-4: Link Management Elements



# **USB 3.0 Technology**

---

## **LTSSM Functional Block**

Note (1) in Figure 11-4 on page 227. The discussion of Port-to-Port protocol link management elements starts here because the Link Training and State Machine (LTSSM) is responsible for managing several new (or improved) USB features related to:

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

By monitoring link layer error recovery attempts, header packet flow control and packet acknowledgement time-outs, power management inactivity timers, etc., the LTSSM is continuously checking the health of the link and the responsiveness of the link partner. As required, it initiates (or responds to) handshakes with the link partner to coordinate LTSSM state transitions.

Additional coverage of the LTSSM is provided in the following chapters:

- Chapter 12, entitled "LTSSM And the SuperSpeed Link States," on page 243
- Chapter 20, entitled "Link Training," on page 427
- Chapter 21, entitled "Link Recovery and Retraining," on page 467
- Chapter 26, entitled "Compliance Testing," on page 617
- Chapter 27, entitled "Receiver Loopback Testing," on page 639
- Chapter 24, entitled "SuperSpeed Power Management," on page 563

---

## **Tx, Rx Link Commands**

Note (2), Note (3) in Figure 11-4 on page 227. Link commands are used for link layer communication between pairs of link partners. On the link, link commands are always a total of eight symbols; they are never forwarded to other links.

---

## **Link Command Groups**

The link command types fall into four groups:

- Packet Acknowledgement
- Flow Control
- Power management
- Link Up/Link Down

# **Chapter 11: Introduction to Port-To-Port Protocol**

---

For a detailed descriptions of link commands, refer to the following chapters:

- Chapter 13, entitled "Link Commands," on page 289
- Chapter 15, entitled "Header Packet Flow Control," on page 321
- Chapter 16, entitled "Link Errors & Packet Acknowledgement," on page 339
- Chapter 24, entitled "SuperSpeed Power Management," on page 563

## **Packet Acknowledgement Link Commands**

Packet acknowledgement link commands include LGOOD\_n, LBAD, and LRTY and are used by the receiver and transmitter to acknowledge receipt of valid header packets or to request a retry by the transmitter in the event of failed CRC checks at the receiver.

## **Flow Control Link Commands**

Flow control link commands are sent by each receiver to indicate availability of Rx HP Buffer space at the link layer. As the receiver is required to implement four buffers and use them in order (Buffer A, B, C, D, A, etc.). The flow control link commands are similarly encoded: LCRD\_A, LCRD\_B, LCRD\_C, LCRD\_D. For each flow control link command received, the transmitter increments its credit counter (1 credit = 1 header).

## **Power Management Link Commands**

There are four link commands used by link partners to negotiate transitions from the U0 link state to one of the power management link states, U1-U3. The transitions normally occur following a handshake sequence; the link commands used for this purpose are: LGO\_Ux, LAU, LXU, and LPMA. Note that there are also transitions that occur automatically on certain timeout events during power management handshake attempts.

## **Link Up/Link Down Link Commands**

In the absence of other packets while the link is in the U0 state, devices are required to periodically (every 10uS) send a link command informing the link partner that the device is still present and still in the U0 state. The downstream facing ports of the host controller and external hubs send the LDN link command; upstream facing ports of devices and external hubs send the LUP link command.

## **USB 3.0 Technology**

---

### **HP Acknowledgement**

Note (4) in Figure 11-4 on page 227. Port-to-Port protocol requires the transmitter to maintain a copy of each header packet in its local Tx HP Buffer until it is acknowledged by the receiver. For each header packet received and checked, the receiver sends an acknowledgement link command. There are two cases:

- If both CRC checks and the sequence number check are OK, the receiver sends the LGOOD\_n link command ("n" is the sequence number of the good header packet). The transmitter then flushes its local copy of the header packet.
- If a CRC check fails, then an LBAD link command is returned to the transmitter requiring a retry of the oldest unacknowledged header packet in the Tx HP Buffer.

Up to three retries of the same header packet may be attempted before the LTSSM will force a transition to recovery to correct the problem.

Additional details on header packet acknowledgement may be found in:

- Chapter 14, entitled "Header Packet Processing," on page 303
- Chapter 16, entitled "Link Errors & Packet Acknowledgement," on page 339
- Chapter 21, entitled "Link Recovery and Retraining," on page 467

---

### **HP Flow Control**

Note (5) in Figure 11-4 on page 227. Each time the link layer receiver forwards a header packet from its Rx HP Buffer to the protocol layer, it is required to send the corresponding LCRD\_x flow control link command to the transmitter. In addition, an initial "advertisement" of flow control credits is required on each entry to the U0 link state. These advertisements always report which of the four Rx HP Buffers (A,B,C,D) are currently free for use.

Each time the transmitter receives an LCRD\_x flow control link command, it increments its local credit counter (1 credit = 1 header).

Additional details on header packet flow control may be found in:

- Chapter 14, entitled "Header Packet Processing," on page 303
- Chapter 15, entitled "Header Packet Flow Control," on page 321
- Chapter 21, entitled "Link Recovery and Retraining," on page 467

## **Chapter 11: Introduction to Port-To-Port Protocol**

---

### **Ordered Sets**

Note (6) in Figure 11-4 on page 227. One of the most important components in Port-to-Port protocol link management is the use of ordered sets. Ordered sets are not packets or link commands, but do appear on the link as fixed groups of 10-bit symbols. The number of symbols in an ordered set varies from two to thirty two.

---

### **Ordered Set Data (D) and Control (K) Symbols**

Before reviewing the format and use of ordered sets, it is worth noting that they are comprised of 10-bit symbols output from the 8b10b encoder. Some ordered sets are made up exclusively of control (K) symbols and some are a combination of control (K) and data (D) symbol types. For reference in the discussion that follows, Table 11-1 on page 231 lists the control (K) symbols.

*Table 11-1: Control (K) Symbols*

<b>Encoding</b>	<b>Symbol</b>	<b>Name</b>	<b>Usage</b>
K28.1	SKP	Skip	Clock compensation
K28.2	SDP	Start Data Packet	Marks start of Data Packet Payload
K28.3	EDB	End Bad	Marks end of nullified Data Packet Payload
K28.4	SUB	Symbol Substitute	Replaces invalid symbol decode
K28.5	COM	Comma	Symbol Alignment
K28.6	Rsvd	Rsvd	Reserved for future use
K27.7	SHP	Start Header Packet	Marks start of Data Packet Header (DPH)
K29.7	END	End	Marks end of valid Data Packet Payload
K30.7	SLC	Start Link Command	Marks start of Link Command
K23.7	EPF	End Packet Framing	Marks end of Packet or Link Command framing

# USB 3.0 Technology

---

## Four Ordered Set Functional Groups

There are twelve defined ordered set types, in four functional groups:

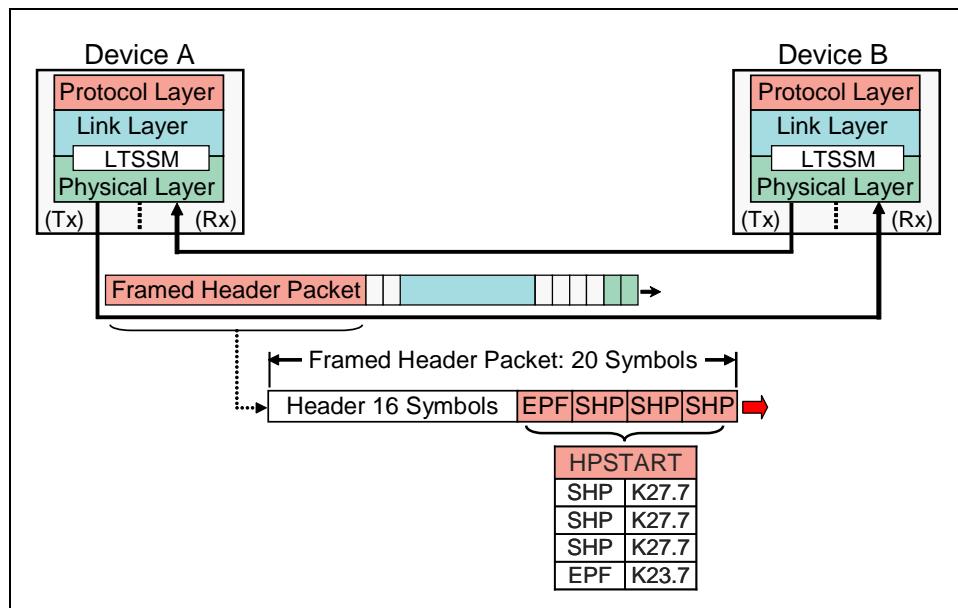
- Framing (delimiter) ordered sets
- Link training and retraining ordered sets
- Clock compensation ordered sets
- Loopback Bit Error Rate Test (BERT) ordered sets

### Framing Ordered Sets (Delimiters)

In order to simplify the receiver task of recognizing boundaries between packets and link commands, framing ordered sets are added. There are five variants of packet and link command framing.

**Header Packet Framing.** As depicted in Figure 11-5 on page 232, outbound 16-byte headers are preceded by the four-symbol header packet start (HPSTART) framing. There is no framing at the back of the header packet.

Figure 11-5: Header Packet Framing Ordered Set

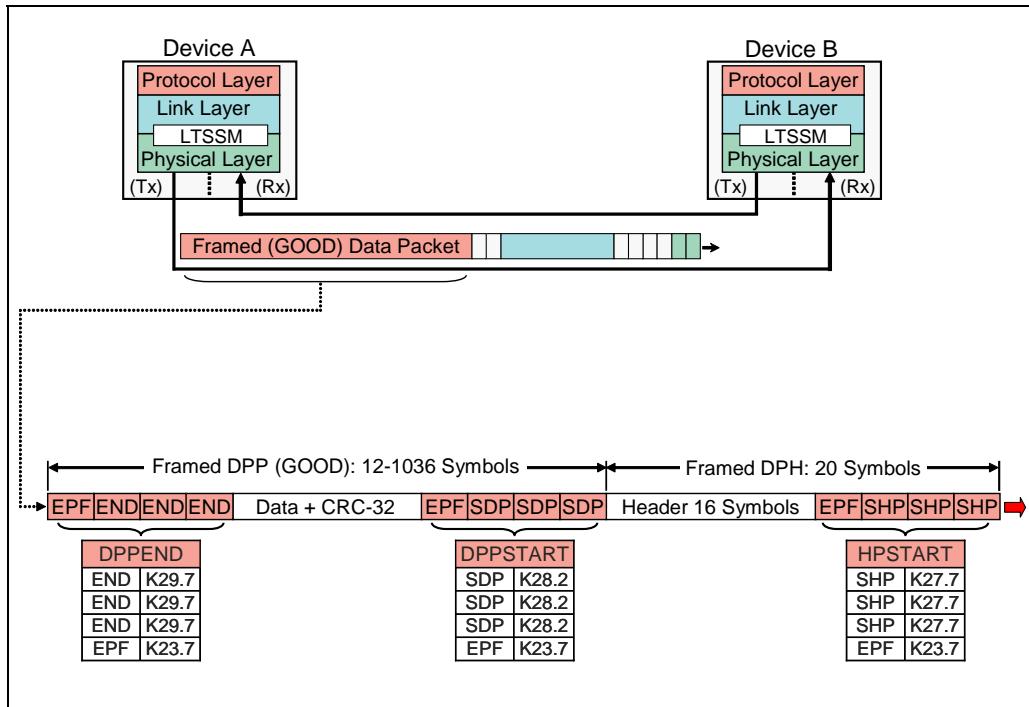


## Chapter 11: Introduction to Port-To-Port Protocol

**Data Packet Framing (Valid Data Payload).** As shown in Figure 11-6 on page 233, data packet framing requires three ordered sets. The data packet header (DPH) is framed at the front with four symbol HPSTART ordered set in the same way as any other header packet. The data packet payload (DPP) is framed at the front with the four symbol DPPSTART framing and at the back with the four symbol DPPEND framing. There is no inter-packet gap between the data packet header and the data packet payload.

In the special case of a zero-byte transfer, the DPP would consist only of the DPPSTART framing, CRC-32, and the DPPEND framing (a total of 12 symbols).

Figure 11-6: Good Data Packet Framing Ordered Sets

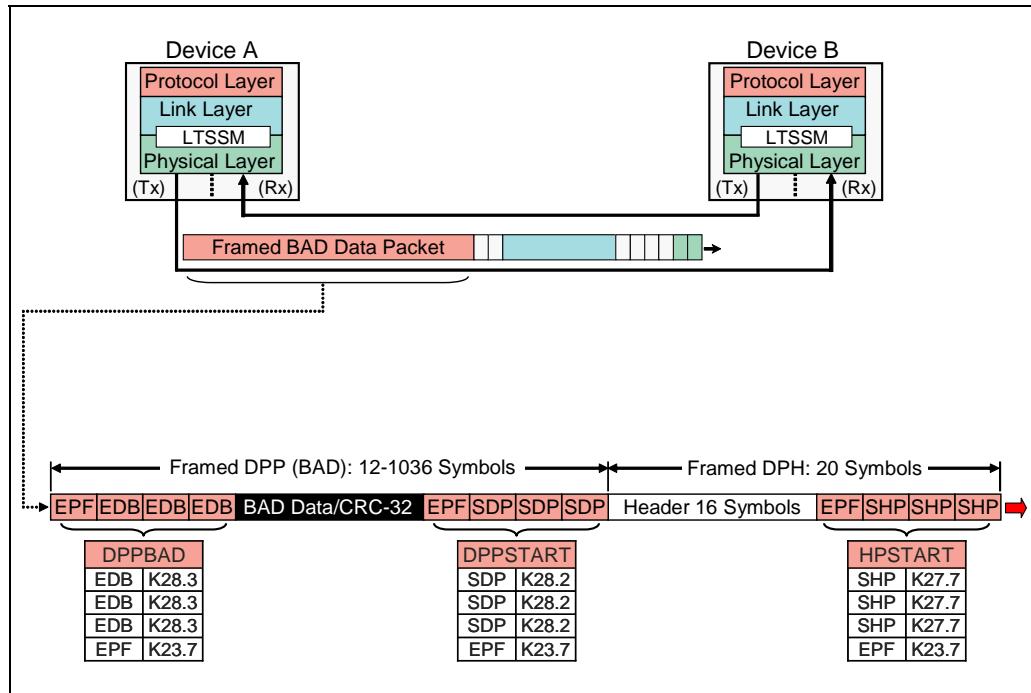


## USB 3.0 Technology

**Data Packet Framing (Nullified Data Payload).** It is possible that a data packet is corrupted in flight as it crosses a hub from one link to another. If the data packet payload (DPP) is known to be corrupted, the forwarding device may send a nullified data packet. As depicted in Figure 11-7 on page 234, an nullified data packet payload is framed at the front with four symbol DPPSTART framing and at the rear with four symbol DPPBAD framing. In addition, the CRC-32 is inverted.

For endpoint types other than isochronous, when a nullified data packet is delivered to the ultimate recipient, an ACK Transaction Packet is returned with a Sequence Number (Seq Num) that indicates the data packet should be sent again.

Figure 11-7: Nullified Data Packet Framing Ordered Sets

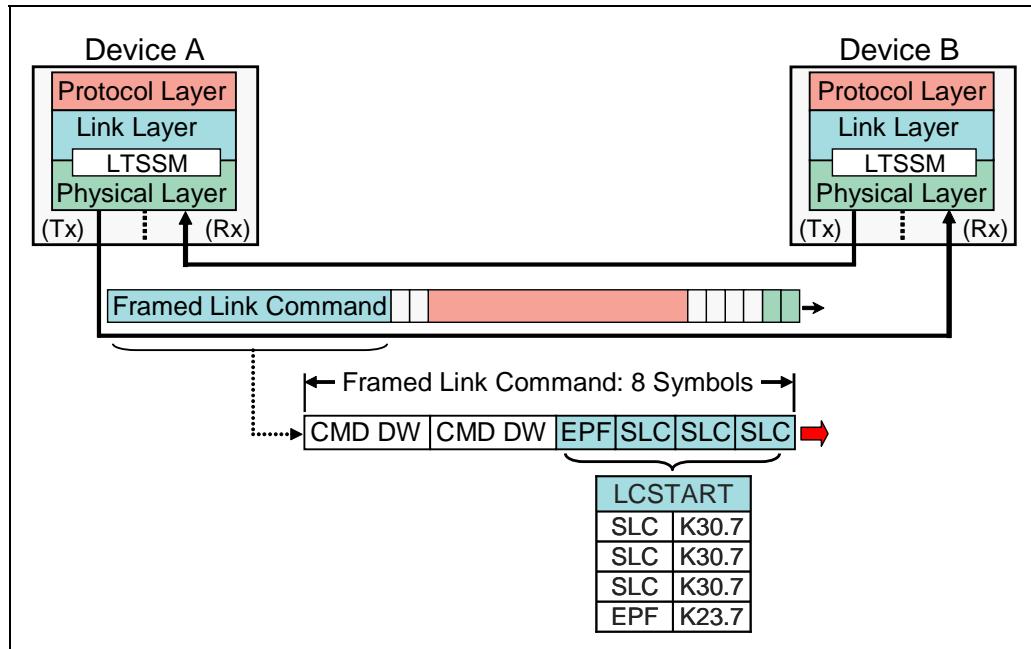


## Chapter 11: Introduction to Port-To-Port Protocol

**Link Command Framing.** There are four categories of link commands used for header packet acknowledgement, flow control, link power management handshake, and periodic link up/down indication signaling. Link commands are not flow controlled and must be accepted when sent. To simplify recognition of link commands interleaved with header and data packet traffic, a distinctive ordered set framing is used.

As depicted in Figure 11-8 on page 235, a total of eight symbols are transmitted: first is the four symbol link command start (LCSTART) framing, followed by the Link Command Word (CMD DW in the illustration) which is always sent twice.

Figure 11-8: Link Command Framing Ordered Sets



# USB 3.0 Technology

## Link Training & Retraining Ordered Sets

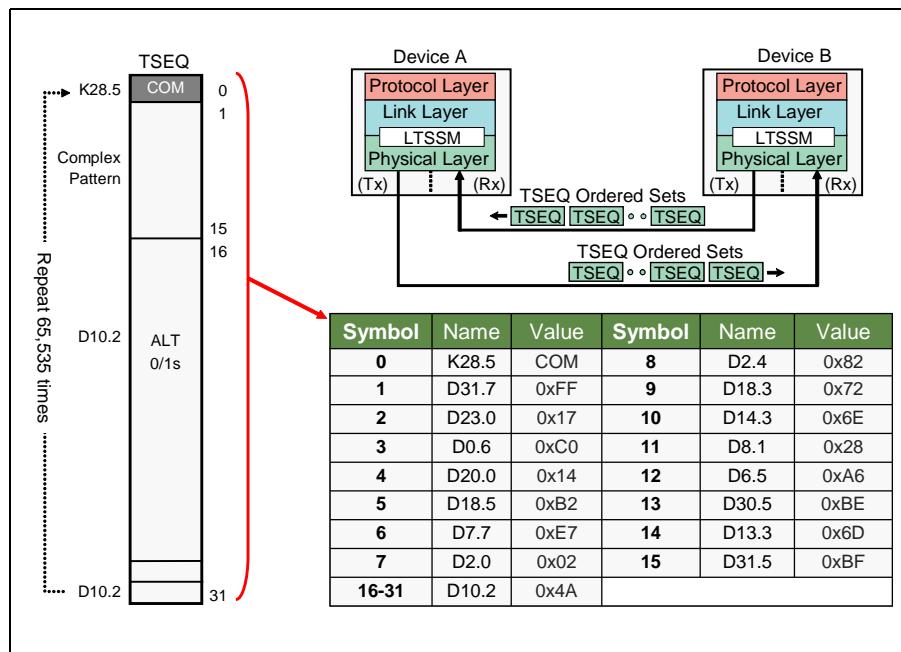
There are three ordered sets used for the purposes of SuperSpeed link training as well as for recovery retraining following serious link errors or an exit from U1-U3 power management states:

- TSEQ
- Training Sequence 1 (TS1)
- Training Sequence 2 (TS2)

**TSEQ Ordered Set.** During full link training, transmitters send the TSEQ ordered set 65,536 times to enable the link partner receiver PHY to set up its equalizer parameters and compensate for real-world signal conditions at its differential inputs.

As indicated in Figure 11-9 on page 236, the TSEQ ordered set is comprised of thirty two 10-bit symbols which were selected for frequency content and bit patterns useful in training receiver equalization, achieving bit and symbol lock, etc.

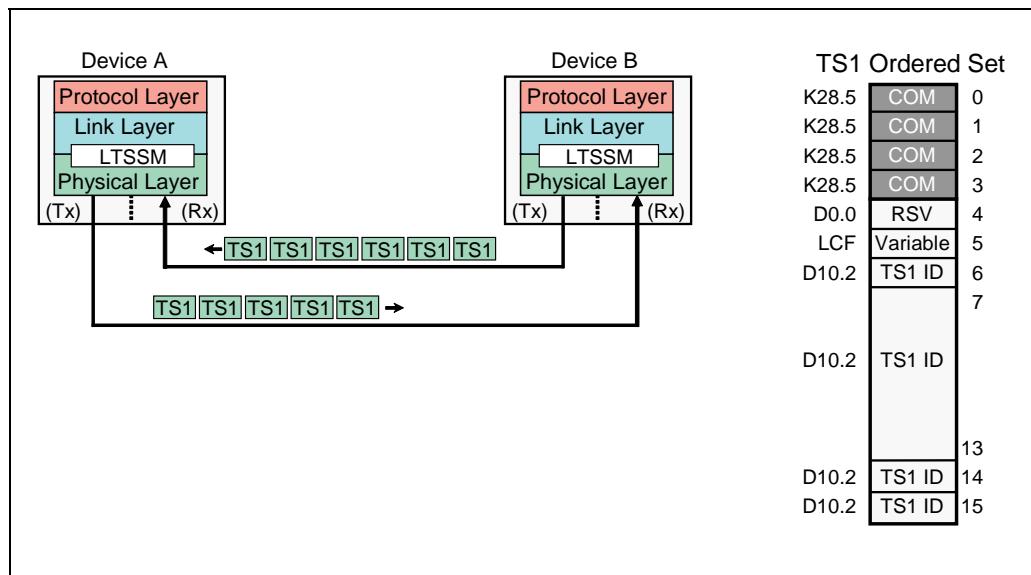
Figure 11-9: TSEQ Ordered Sets Train Receiver Equalizer



## Chapter 11: Introduction to Port-To-Port Protocol

**TS1 Ordered Set.** TS1 ordered sets are 16 symbols used for handshake purposes during full link training and retraining during exits from link power management states, recovery from serious link errors, etc. Figure 11-10 on page 237 depicts the format of the TS1 ordered set. The D10.2 symbols (TS1 symbols 6-15) contain a high frequency bit pattern of alternating 0's and 1's used by receivers to confirm that equalizer parameters are correct. The D10.2 symbols are also used to detect differential signal polarity inversion and correct it internally, if needed. The first four TS1 symbols are COM and have a distinctive bit pattern used by receivers to detect the boundaries between 10-bit symbols (referred to as symbol lock).

Figure 11-10: TS1 Ordered Set

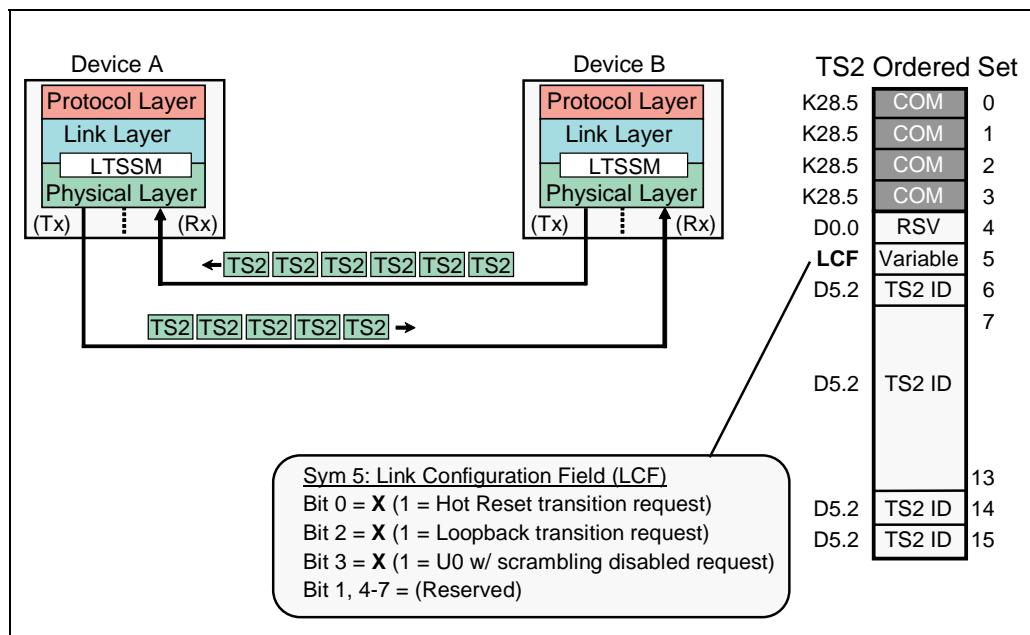


## USB 3.0 Technology

**TS2 Ordered Set.** TS2 ordered sets are 16 symbols each and are exchanged during the final stages of link training or recovery/retraining. Figure 11-11 on page 238 depicts the TS2 ordered set 16-byte format. The purpose of TS2 ordered sets is twofold. Receipt of TS2's informs the link partner that a device has completed any required physical layer clock/data recovery and symbol lock training and is prepared to exit link training (or retraining).

The second function of the TS2 ordered sets is to provide an opportunity for link partners to signal a request to change link behavior in the next state. Aside from the normal transition into the U0 link state, three options are available; one of the options may be selected by setting the corresponding bit in the TS2 Symbol 5 Link Configuration Field (LCF). As indicated in Figure 11-11 on page 238, the choices are: a transition to Hot Reset, a transition to receiver loopback testing, or a transition to U0 with scrambling disabled.

Figure 11-11: TS2 Ordered Set

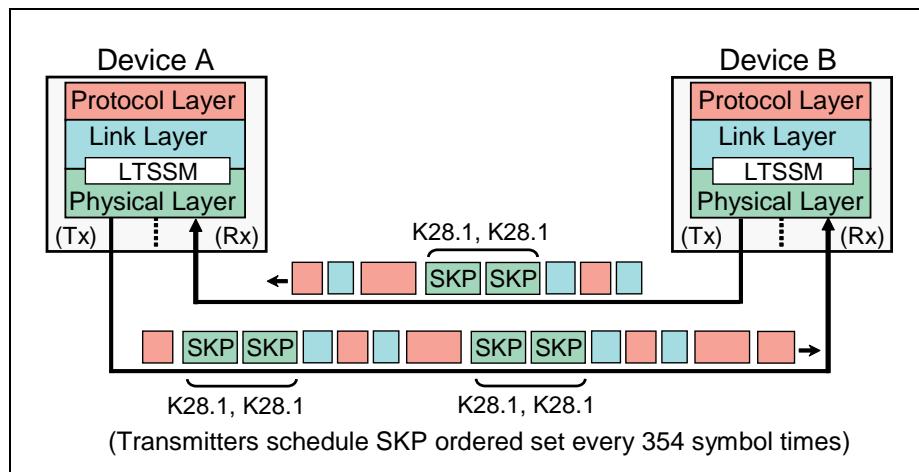


# Chapter 11: Introduction to Port-To-Port Protocol

## Clock Compensation Ordered Set (Skip Ordered Set)

The skip ordered set consists of a pair of SKP control (K) symbols, K28.1, sent periodically by SuperSpeed link partner transmitters any time the SuperSpeed link is in the U0 state. As indicated in Figure 11-2 on page 223, the skip ordered set symbols are scheduled to be sent every 354 symbol times, but if a transfer is in progress at the scheduled time, the transmitter waits until an appropriate packet or link command boundary to insert them into the outbound data stream.

Figure 11-12: Skip Ordered Set



The purpose of the skip ordered sets is to enable each link partner receiver to compensate for the slight mismatch in rates between the clock recovered from the transmitter's incoming bit stream and the receiver's local clock. The mismatch is caused by the fact that transmitter and receivers have independent local clocks that are permitted to be within +/- 300 ppm of each other. In addition, transmitters employ spread spectrum clocking (SSC) to reduce EMI; SuperSpeed SSC requires the transmitter's bit rate to slowly range between +/- 5000 ppm from the nominal rate which causes an additional mismatch between recovered clock and local clock at the receiver that changes dynamically.

To avoid over flow and underflow conditions, each receiver employs an elastic buffer at the physical layer that temporarily stores incoming symbols. A pair of SKP ordered sets is discarded from the elastic buffer if the level starts to rise beyond a designed threshold. Similarly, the receiver may gate the output clock

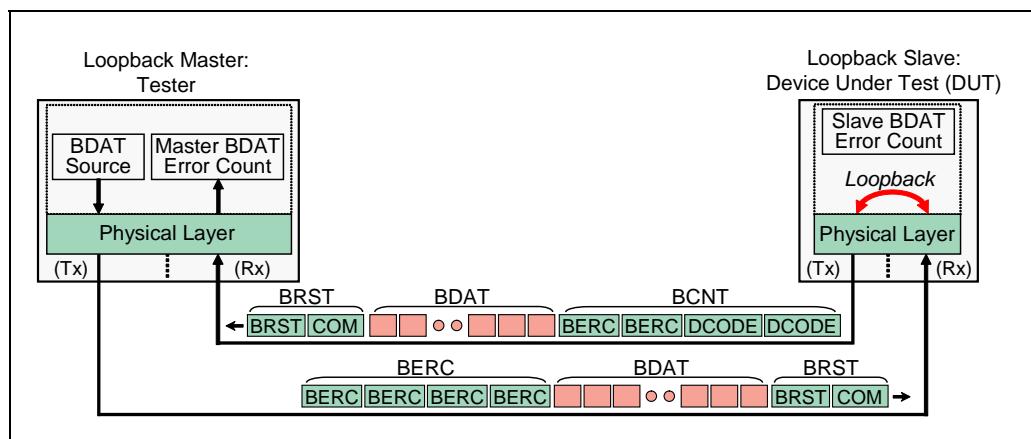
# USB 3.0 Technology

or insert SKP pairs into the buffer at times when the transmitter rate is on the low side causing the elastic buffer level to fall.

## Loopback Bit Error Rate Test (BERT) Ordered Sets

**General.** The Link Training and Status State Machine (LTSSM) supports a loopback mode that simplifies performance of a SuperSpeed link bit error rate test (BERT). As shown in Figure 11-13 on page 240, the loopback configuration consists of a test fixture (protocol analyzer, etc.) acting as loopback master connected to the loopback slave device under test (DUT).

*Figure 11-13: Loopback BERT Ordered Sets*



In the simplest usage model, the loopback master transitions the DUT into the LTSSM loopback mode by sending TS2 ordered sets during link training with the TS2 symbol 5 loopback bit set = 1. Once the DUT is in loopback mode, the loopback master starts transmitting the BERT data pattern (BDAT) consisting of pseudo-random data (scrambled and encoded NOP symbols). The receiver physical layer (PHY) simply loops the pattern back to the master where the incoming symbols are checked for errors and a count is accumulated. The error count is then correlated to the link target bit error rate (BER) of  $10^{-12}$ .

An optional capability for USB 3.0 devices is support for slave error counting by the device under test as it receives and loops back the test pattern to the master. Referring again to Figure 11-13 on page 240, the three ordered sets required to support slave BERT counting are shown. They include:

## **Chapter 11: Introduction to Port-To-Port Protocol**

---

- BRST
- BERC
- BCNT

**BRST Ordered Set.** Loopback BRST is an ordered set command requesting the slave to clear its 16-bit BDAT error counter. This ordered set is only sent by the loopback master and consists of the two 10-bit symbols shown in Table 11-2 on page 241. BRST may be repeated as often as necessary by the master. As with other loopback traffic, BRST is retransmitted by the slave.

*Table 11-2: Loopback BRST Ordered Set*

Symbol Number	Encoding	Description
0	K28.5	COM symbol
1	K28.7	BRST symbol. Used in Bit Error Rate Test (BERT) protocol to reset error count at device under test

**BERC Ordered Set.** Loopback BER C is an ordered set command requesting the slave to return the error count accumulated for received BDAT symbols since the last BRST was received. This ordered set is sent by the master; format is shown in Table 11-3 on page 241.

*Table 11-3: Loopback BER C Ordered Set*

Symbol Number	Encoding	Description
0	K28.3	BERC symbol
1	K28.3	BERC symbol
2	K28.3	BERC symbol
3	K28.3	BERC symbol

## **USB 3.0 Technology**

---

**BCNT Ordered Set.** Upon receipt of the BERC ordered set, a loopback slave that supports BERT state machine and error counting returns the BCNT ordered set. BCNT consists of the four 10-bit symbols shown in Table 11-4. Note that the first two symbols are BERC (K28.3) and the last two symbols contain the 16-bit (two byte) error count. The error count symbols are labeled *DCODE* in the table and are not scrambled.

*Table 11-4: Loopback BCNT Ordered Set*

Symbol Number	Encoding	Description
0	K28.3	BERC symbol
1	K28.3	BERC symbol
2	DCODE	BCNT. First byte of 16-bit error count
3	DCODE	BCNT. Second byte of 16-bit error count

For more information on this topic, refer to Chapter 27, entitled "Receiver Loopback Testing," on page 639.

---

---

# 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 its 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.

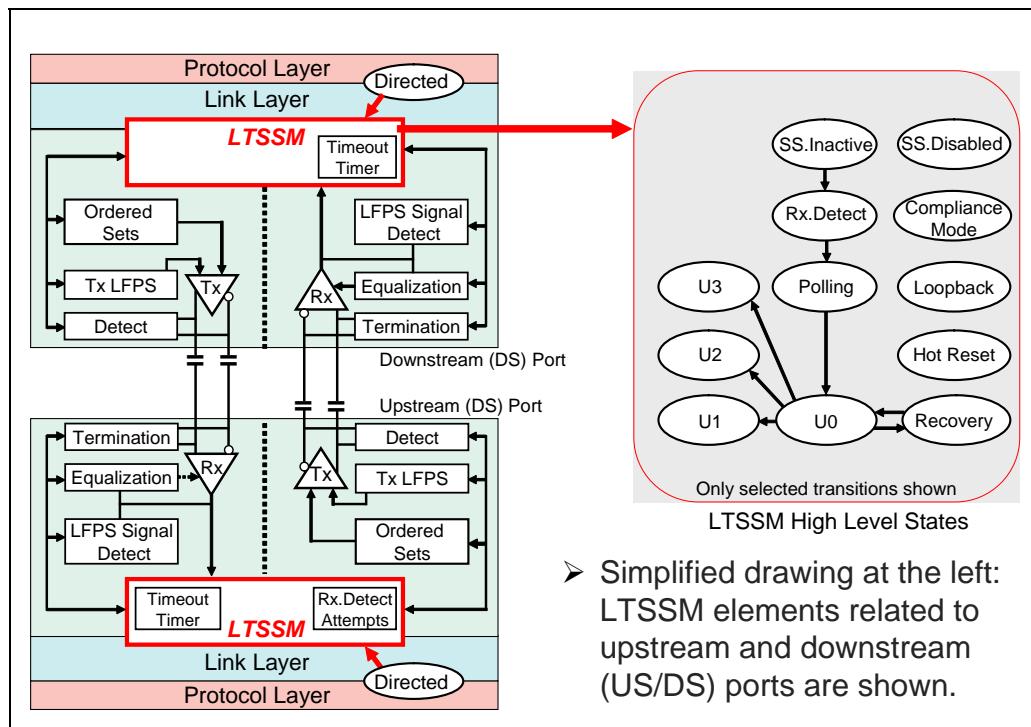
# USB 3.0 Technology

## 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:  $U_0, U_1, U_2, U_3$
- 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)



## **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 more responsibilities to each pair of link partners. 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 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.

# **USB 3.0 Technology**

---

## **LTSSM Coordinates SuperSpeed Link Training**

A number of features of USB 3.0 make the use of the LTSSM essential in the training and retraining of a SuperSpeed link:

- In the move from USB 2.0 to USB 3.0 SuperSpeed, a major effort was made to preserve the USB cable and connector model. At one extreme are short channels ranging from dongles to chip-to-chip connections. At the other extreme, connections involving pc board traces, multiple connectors, and cables lengths of up to 3 meters or so are possible. At 5 Gb/s SuperSpeed, receivers are required to use link training to adapt equalizers to real-world link conditions. LTSSMs coordinate the link training sequence.
- Once link training equalization enables the receiver PHY to recover a usable differential signal “eye”, the remainder of training permits clock and data recovery, 10-bit symbol lock, elastic buffer initialization, and correction of differential polarity inversion (if needed).

After initialization, each transition out of U1, U2, or U3 link power management states requires the LTSSM to manage the recovery and link retraining sequence. The LTSSM also manages recovery and retraining in the event of link errors that cannot be corrected by other means. One of the significant benefits of recovery and retraining is that the lengthy Warm Reset signaling isn't needed and equalization parameters are preserved.

---

## **LTSSM And Link-Level Error Handling**

Several of the key improvements brought by USB 3.0 SuperSpeed protocol are related to error handling. A brief summary of shortfalls in USB 2.0 error handling is presented here, followed by a description of the approach taken by USB 3.0 for SuperSpeed links.

### **Background: USB 2.0 Approach to Bus Errors**

One of the limitations of legacy USB 2.0 is a heavy reliance on the host controller and the USB software to handle most aspects of communication and link management. The simplicity of USB 2.0 promotes low cost peripherals and hubs, but also results in a distinct lack of capabilities with regard to link level error handling. In general, if a USB 2.0 device detects an error on its bus (e.g. token, data, handshake packet CRC, etc.) it doesn't respond. This means that the host controller monitors expected inbound traffic for each endpoint and informs software in the event an error results in a timeout condition.

## **Chapter 12: LTSSM And the SuperSpeed Link States**

Because USB 2.0 employs a broadcast bus model, with no peer-to-peer or DMA traffic, there is only one transaction in progress at a time per host controller. This makes pausing for error correction under host controller management fairly straightforward, but slow.

### **USB 3.0 SuperSpeed Approach To Link Errors**

For USB 3.0 SuperSpeed protocol, the level of host controller and software involvement seen in USB 2.0 link error handling would be problematic. SuperSpeed USB replaces the broadcast bus model with unicast bus and dual simplex signaling. While peer-to-peer and DMA transfers are still not permitted, USB 3.0 devices operating at SuperSpeed are allowed to send asynchronous notifications upstream to the host controller at any time. Because there can be independent, simultaneous downstream and upstream packet traffic (either or both of which could encounter errors), there is no practical way to retain the simple host controller managed error correction scheme of USB 2.0.

Instead, USB 3.0 SuperSpeed relies on link layer CRC generation and checking and adds automatic, hardware-based header packet *retry* by the transmitter for any header packet received with a CRC error at the receiver. If successful, such header packet retries do not require the link to leave the U0 operational state or any involvement by software or the LTSSM.

### **When Does The LTSSM Become Involved?**

The target bit error rate (BERT) for each USB 3.0 SuperSpeed link is  $10^{-12}$ . Link layer CRC generation/checking and header packet replay handles the occasional (and expected) bit-flip errors without requiring an exit from U0 or intervention by the LTSSM. There are other occasions when more serious or repetitive link errors occur that cannot be handled with packet replay. These types of errors may be attributable to an incorrect packet sequence or flow control problems at the transmitter, receiver, or both.

In cases like these, the LTSSM assumes responsibility for restoring reliable link operations. Time-outs or excessive replay attempts trigger the LTSSM to issue TS2 ordered sets (initiating a transition of the link to the Recovery state), followed by the start of link retraining. If successful, the link partners quickly return to the U0 state and normal packet transfers resume. Generally, all of this occurs without the need for host controller or software involvement.

# **USB 3.0 Technology**

---

## **LTSSM And SuperSpeed Link Power Management**

### **Background: USB 2.0 Power Management**

Another limitation of legacy USB 2.0 is the restrictive set of bus power management capabilities. In the absence of any bus traffic, devices on a 2.0 bus segment automatically enter electrical idle--the *suspend* power conservation state--after 3mS of idle time. Software may globally or selectively suspend hub downstream facing ports to force this condition.

A fairly lengthy resume signaling protocol is used during an exit from 2.0 suspend--reducing the usefulness of the mechanism in low-latency environments. Later USB 2.0 implementations optionally support the 2.0 Link Power Management (LPM) extensions which address some of these issues.

### **USB 3.0 SuperSpeed Power Management Enhancements**

USB 3.0 SuperSpeed links support a collection of features intended to improve overall device and platform power savings. SuperSpeed link power management includes four link power states and provides an opportunity for software to program U1 and U2 inactivity timers on a per-link basis. Link partners then coordinate power management transitions under LTSSM hardware control.

### **The LTSSM Role In SuperSpeed Link Power Management**

If an inactivity timer in one of the SuperSpeed link partners expires, the LTSSMs automatically coordinate the link power management transition handshake between the two devices--without the need for software involvement. Of course, software always has the option to invoke a power management transition directly by targeting a hub port with the appropriate hub-class power request. In this case, the LTSSM for the hub downstream facing port would initiate the handshake signaling.

# **Chapter 12: LTSSM And the SuperSpeed Link States**

---

## **LTSSM And SuperSpeed Link Testing**

### **Background: USB 2.0 Testing**

Because of the bit rates involved, devices operating at USB 2.0 low speed (LS) and full speed (FS) do not require (or support) a built-in compliance mode test feature for verifying electrical signal levels and timing requirements. Physical layer validation relies on protocol testers, scopes, and other external test methods.

For USB 2.0, peripheral and hub devices operating at 480 Mb/s high speed (HS) must support the HS compliance test mode and the Port\_Test USB Device Request. This mode enables a test platform to command a device under test (DUT) to output a variety of test patterns which may then be checked for compliance with voltage/timing, impedance, and other physical layer specifications.

### **USB 3.0 SuperSpeed Link Testing Features**

USB 3.0 SuperSpeed (SS) greatly extends this capability by defining two special LTSSM test states:

- Compliance Mode enables a tester to command the device under test to output nine test patterns, ranging from 5 Gb/s toggling 0's and 1's and pseudo-random data (with or without transmitter de-emphasis) to Low Frequency Periodic Signaling (LFPS). Voltage and timing measurements at the transmitter outputs and receiver inputs can then be made. Refer to Chapter 26, entitled "Compliance Testing," on page 617 for details.
- Loopback Mode is the second LTSSM test mode. It enables a tester (loopback master) to verify the actual bit error rate (BER) for a SuperSpeed Link. When Loopback is entered, the tester outputs a known test pattern (e.g. pseudo-random data) which is retransmitted (looped back) by the physical layer of the device under test. The tester detects symbols received in error and calculates the actual BER for the link. Optionally, the device under test (DUT) may implement additional state machine logic enabling it to detect and count errors as it performs the loopback. If this is supported, the tester fetches and compares the slave error count to its own and can determine BER for each link direction. Refer to Chapter 27, entitled "Receiver Loopback Testing," on page 639.

# **USB 3.0 Technology**

---

---

## **LTSSM State Transitions**

---

### **General**

An important element of LTSSM behavior is the fact that link partners are required to coordinate transitions from one state or substate to another. Some transitions are only initiated by downstream facing ports, some by upstream facing ports, some by either type of port.

LTSSM state transitions result from a number of events, including:

- Device attachment/removal
- Cycling V<sub>BUS</sub> power (which triggers a Power On Reset)
- Warm Reset or Hot Reset
- The link training sequence
- Link power management transitions
- Link errors which are uncorrectable in U0
- Any of the defined link layer time-outs
- Directed transitions

---

### **What Is A Directed LTSSM Transition?**

The last bullet in the list above refers to *directed transitions*. Many LTSSM state transitions occur automatically in response to bus events. An example is the transition to SS.Disabled when a peripheral device times out during the Polling State while attempting link training with its partner. Directed transitions, on the other hand, result because of a request from higher protocol layers. An example is a hub receiving a request to perform a Hot Reset on one of its downstream facing ports. In response, the hub would transition to the Hot Reset state and perform the required handshake to bring its link partner to the same state.

---

### **Handshake Signaling Locks LTSSMs**

The LTSSM is specified in such a way that link partners transition together from state to state in a lock-step manner. When a downstream facing port is directed to initiate an event, or an upstream facing port detects an event requiring a state change, its LTSSM transitions to the new state, initiates handshake signaling, and waits for a response from its partner indicating that it too has transitioned to the new state. Until the appropriate handshake response is received from the

## **Chapter 12: LTSSM And the SuperSpeed Link States**

partner, the transition is not complete. Requiring link partners to use handshake signaling in order to proceed through state changes assures that completion occurs as quickly as possible for each device, while allowing for varying response times resulting from hardware design differences.

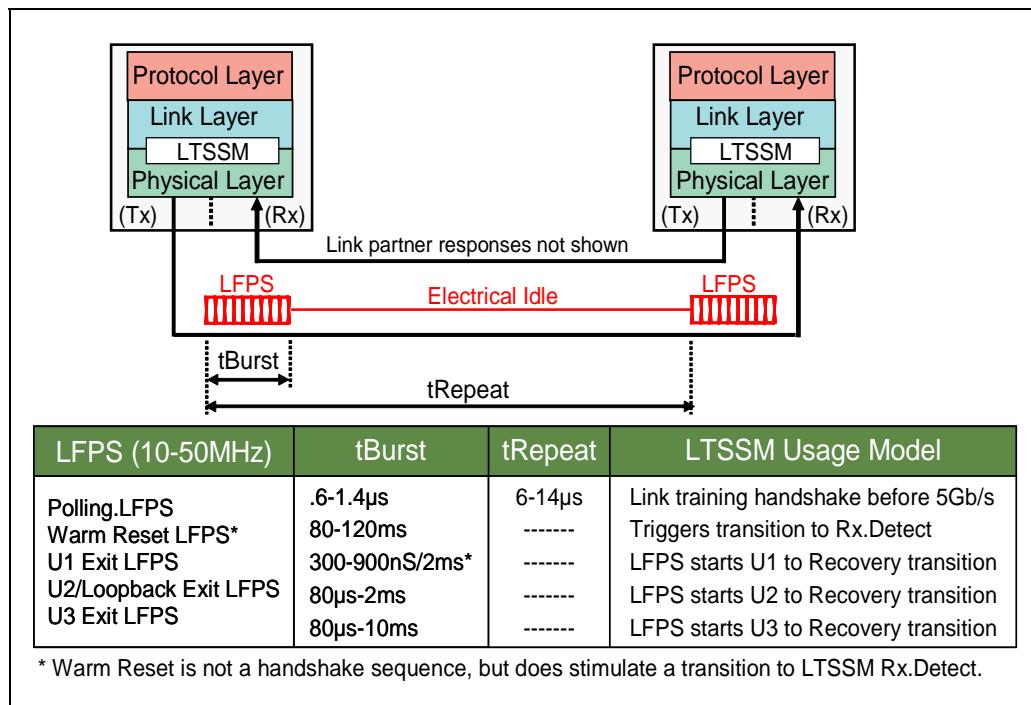
### **LTSSM Handshake Takes Several Forms**

The type of LTSSM handshake signaling performed depends on both the current and next expected state of the link. Generally, if link has been previously trained and is functioning properly, then LTSSM handshake signaling will be done at 5 Gb/s SuperSpeed. If the link isn't operational at 5 Gb/s, then low frequency periodic signaling (LFPS) signaling is used.

#### **LFPS Signaling Events Used In LTSSM Handshake**

There are six LFPS signaling events. Five of these are employed in LTSSM state transitions and are shown in Figure 12-2 on page 251.

*Figure 12-2: LFPS Signaling Used In LTSSM Handshake*



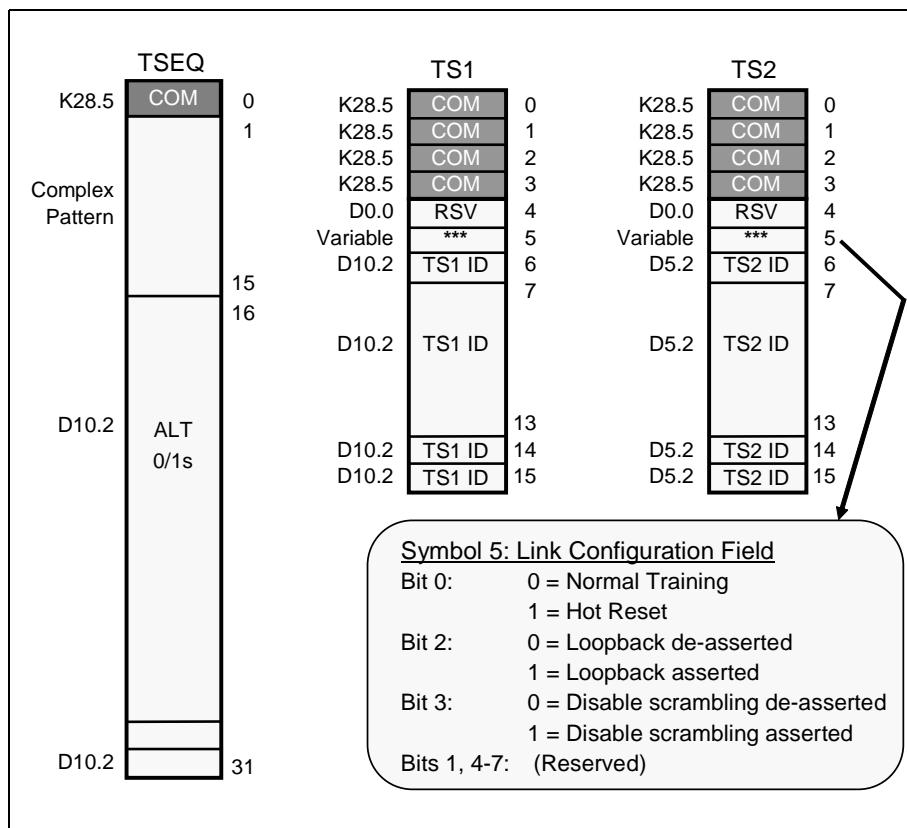
# USB 3.0 Technology

## Ordered Sets Used In LTSSM Handshake

During link training, TSEQ, TS1, and TS2 Ordered Sets are used in two ways. First, they enable the receiver physical layer to adapt to the electrical interface connecting it to the link partner's SuperSpeed transmitter. This includes receiver equalization, clock and serial data recovery, elastic buffer management, etc. Second, as link training proceeds, the exchange of these ordered sets is used to advance the LTSSM states until entry into the U0 link state.

After link training has enabled 5 Gb/s Superspeed operations, TS2 ordered sets are then used to initiate requests and respond during LTSSM transitions to Recovery, Hot Reset, Loopback, etc. In Figure 12-3 on page 252, notice the Hot Reset, Loopback, and Disable Scrambling bits in the *Link Configuration Field* (Symbol 5) of TS1/TS2s.

Figure 12-3: Ordered Sets Used In LTSSM Transition Handshake



## Chapter 12: LTSSM And the SuperSpeed Link States

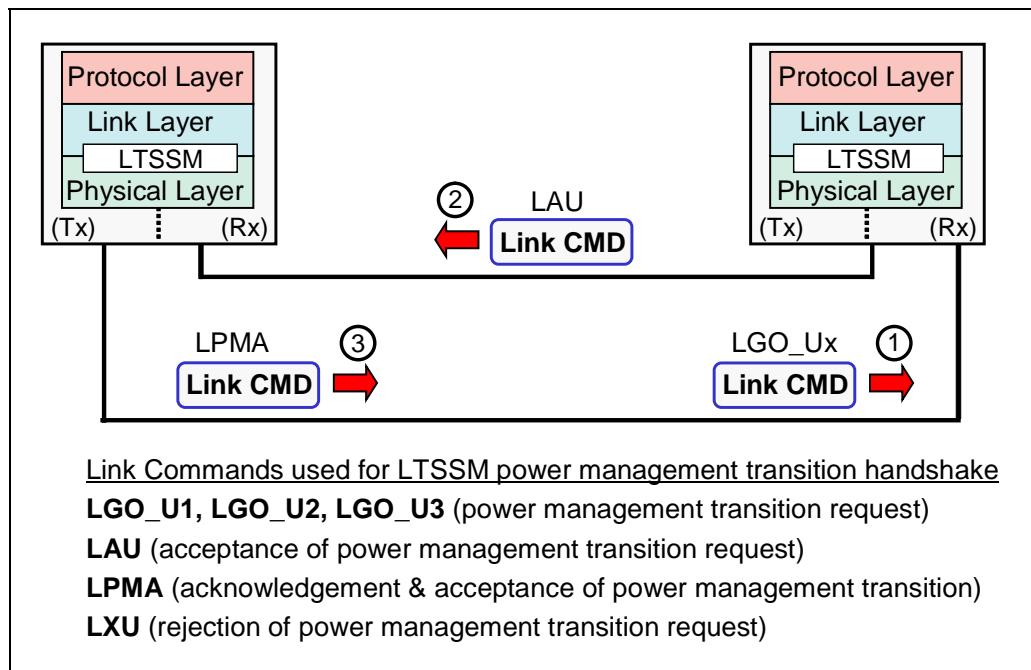
### **Link Commands Used In LTSSM Handshake**

One of the big advantages of SuperSpeed USB over the earlier USB 2.0 protocol is improved link power management. Once enabled by software, LTSSM managed power adjustments can be made dynamically on a link-by-link basis.

USB 3.0 defines a set of power management link commands to be used by link partners when requesting a transition from U0 to U1, U2, or U3. Using link commands instead of TS1/TS2 ordered sets has two advantages, both related to speed. First, link commands are each 8 bytes in length (vs. 16 bytes for TS1/TS2s). Second, using link commands saves additional time as it avoids the need to transition through Recovery (as required when using TS2 ordered sets).

Figure 12-4 depicts a power management link command exchange.

*Figure 12-4: Link Commands Used For LTSSM Power Management Transitions*



Note: LGO\_Ux link commands are used to transition into power management states U1-U3 from U0. Exits from link states U1-U3 require an LFPS handshake to exit electrical idle and a transition through Recovery in order to return to U0.

# **USB 3.0 Technology**

---

---

## **LTSSM Time-outs**

### **General**

To allow for link partner LTSSM response time differences while still assuring acceptable latency during LTSSM state changes, the USB 3.0 Specification defines a maximum timeout for transitions between each LTSSM state/substate. To this end, a set of more than 25 link layer time-outs are defined to track the time spent waiting for various bus events and handshake responses.

### **Timer Characteristics**

Accuracy of link layer timers (other than U1/U2 Inactivity Timers) is -0% to +50%. Each timer is implemented in hardware and loaded with its initial value on each Warm Reset or Hot Reset. In general, on entry into each LTSSM sub-state, the appropriate timer is allowed to start decrementing. If a response is received in time, a timeout is avoided. If a handshake timeout does occur, the LTSSM determines the action to be taken, often involving a transition to SS.Inactive, SS.Disabled, or Rx.Detect state.

The time-outs shown in Table 12-1 indicate the initial state and the state to which the LTSSM will transition if a timeout occurs. Some of the table entries are referenced in the summary discussion of the LTSSM states and transitions that follows.

*Table 12-1: LTSSM State Transition Time-outs*

<b>Timeout Name</b>	<b>Initial State</b>	<b>Next State(s)</b>	<b>Timeout Period</b>
tSISInactiveQuietTimeout	SS.Inactive.Quiet	SS.Inactive.Disconnectc.Detect	12 ms
tRxDetectQuietTimeout	Rx.Detect.Quiet	Rx.Detect.Active	12 ms
tPollingLFPSTimeout	Polling.LFPS <sup>1</sup>	Compliance/ Rx.Detect/ SS.Disabled	360 ms
tPollingActiveTimeout	Polling.Active <sup>1</sup>	Rx.Detect/ SS.Disabled	12 ms

## Chapter 12: LTSSM And the SuperSpeed Link States

*Table 12-1: LTSSM State Transition Time-outs (Continued)*

Timeout Name	Initial State	Next State(s)	Timeout Period
tPollingConfigurationTimeout	Polling.Configuration <sup>1</sup>	Rx.Detect/ SS.Disabled	12 ms
tPollingIdleTimeout	Polling.Idle <sup>1</sup>	Rx.Detect/ SS.Disabled	2 ms
tU0RecoveryTimeout	U0	Recovery	1 ms
tU0LTimeout	U0	U0	10 µs
tNoLFPSResponseTimeout	U1	SS.Inactive	2 ms
PORT_U2_Timeout	U1	U2	Note 2
tU1PingTimeout	U1	Rx.Detect	300 ms
tNoLFPSResponseTimeout	U2	SS.Inactive	2 ms
tNoLFPSResponseTimeout	U3	U3	10 ms
tRecoveryActiveTimeout	Recovery.Active	SS.Inactive,Rx.Detect	12 ms
tRecoveryConfigurationTimeout	Recovery.Configuration	SS.Inactive,Rx.Detect	6 ms
tRecoveryIdleTimeout	Recovery.Idle	SS.Inactive	2 ms
tLoopbackExitTimeout	Loopback.Exit	SS.Inactive	2 ms
tHotResetActiveTimeout	Hot.Reset.Active	SS.Inactive	12 ms
tHotResetExitTimeout	Hot.Reset.Exit	SS.Inactive	2 ms
tU3WakeUpRetryDelay	U3	U3	100 ms
tU2RxdetDelay	U2	U2	100 ms
tU3RxdetDelay	U3	U3	100 ms
Notes:			
1. Timeout during Polling causes transitions to different states (refer to specification)			
2. PORT_U2_Timeout depends on programming of U2 Inactivity Timer.			

# USB 3.0 Technology

---

## LTSSM Reference Section Note

The remainder of this chapter may be viewed as reference information for readers requiring low-level details about each LTSSM state, substate, and transition event.

## Summary Of LTSSM Operational States

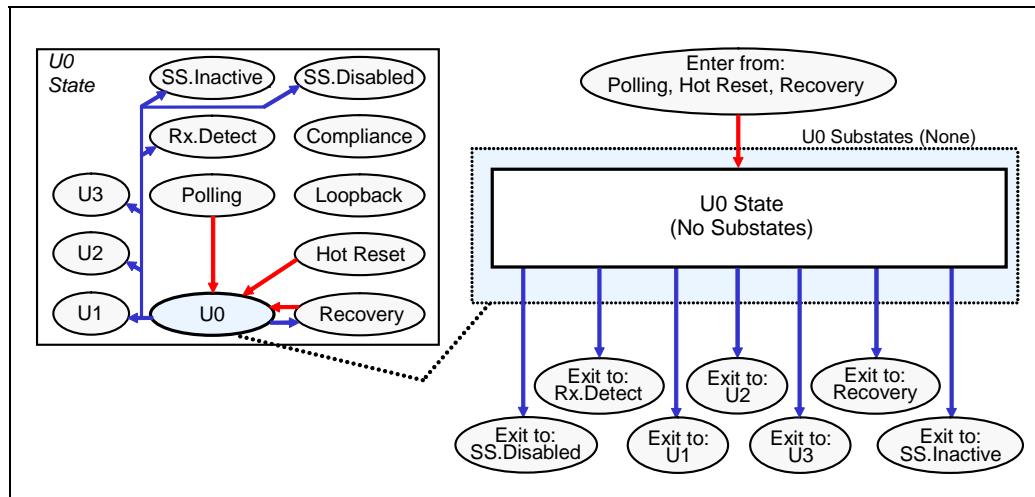
The following section summarizes the U0, U1, U2, and U3 LTSSM Operational States. Included are requirements for each substate as well as the normal and timeout exit paths from the substates. Note: Refer to Chapter 24, entitled "SuperSpeed Power Management," on page 563 for more thorough coverage of USB 3.0 power management and additional details about U1, U2, and U3 link states.

---

### U0

Figure 12-5 on page 256 depicts the LTSSM U0 operational state as well as the possible transitions into and out of the state. There are no substates for U0.

Figure 12-5: LTSSM U0 State



# **Chapter 12: LTSSM And the SuperSpeed Link States**

## **General Description**

U0 is the active, functional state of the SuperSpeed connection. Packets and link commands are sent and received with no latencies resulting from link power management. In this state, symbols are always in flight; in the absence of other traffic, NOP (logical idle) symbols and clock compensation SKP ordered sets enable receivers to maintain lock with transmitters.

## **Requirements In The U0 State**

The following requirements apply while an LTSSM is in the U0 state. Note that there are slightly different requirements for downstream (DS) facing ports and upstream (US) facing ports.

- Ports must observe all U0 transmitter and receiver electrical specifications, including low impedance receiver termination,  $R_{RX-DC}$ .
- The 1 ms  $tU0RecoveryTimeout$  timer is started to track the interval between consecutive inbound link commands. Timer restarts each time a link command is received.
- The 10  $\mu s$   $tU0LTimeout$  timer is also started on entry to U0. This timer tracks the bus idle interval between successive Link commands sent to the link partner. Timer is reset on transmission of the first symbol of an outbound link command and starts counting down as the last symbol is sent and the link goes into logical idle.
- If there is no other link traffic and the 10  $\mu s$   $tU0LTimeout$  timer (see the previous bullet) expires, downstream facing ports transmit an LDN link command; upstream facing ports transmit a LUP link command. These link commands reaffirm that the link is still functioning and the partners remain in U0.

## **Exit Rules For The U0 State**

There are seven possible exits from the U0 State (see Figure 12-5 on page 256).

Exit rules observed by both upstream (US) and downstream (DS) facing ports:

- Ports transition to U1 if LGO\_U1 handshake completes successfully.
- Ports transition to U2 if LGO\_U2 handshake completes successfully.
- Ports transition to U3 if LGO\_U3 handshake completes successfully. Note that only downstream facing ports, operating under software control, may initiate transitions to U3 (Suspend)
- Ports transition to Recovery if directed
- Ports transition to Recovery upon detection of a TS1 ordered set
- Ports transition to Recovery for a variety of error conditions defined in the specification related to header packet flow control and error handling. This includes PENDING\_HP\_TIMER, CREDIT\_HP\_TIMER time-outs, Replay

## **USB 3.0 Technology**

---

count rollover, etc.

- Ports transition to SS.Inactive when PENDING\_HP\_TIMER times out for the fourth consecutive time.

Exit rules observed only by downstream (DS) facing ports:

- Downstream facing ports transition to SS.Inactive when directed
- Downstream facing ports transition to SS.Inactive if U3 entry handshake fails three times.
- Downstream facing ports transition to SS.Disabled when directed.
- Downstream facing ports transition to Rx.Detect state when directed to issue a Warm Reset on the link.
- Downstream facing ports transition to Recovery if  $tU0Recovery$  timer expires indicating no packets or link commands (including LUP) received for 1 ms.

Exit rules observed only by upstream (US) facing ports:

- Upstream facing ports in self-powered devices transition to SS.Disabled when V<sub>BUS</sub> is off.
- Upstream facing ports transition to Rx.Detect when Warm Reset is detected.
- Upstream facing ports transition to SS.Disabled when directed. This results when entry into U0 was not followed by receipt of Port Capability LMP within the time allowed.
- Upstream facing ports transition to Recovery if  $tU0Recovery$  timer expires indicating no packets or link commands (including LDN) received for 1 ms.

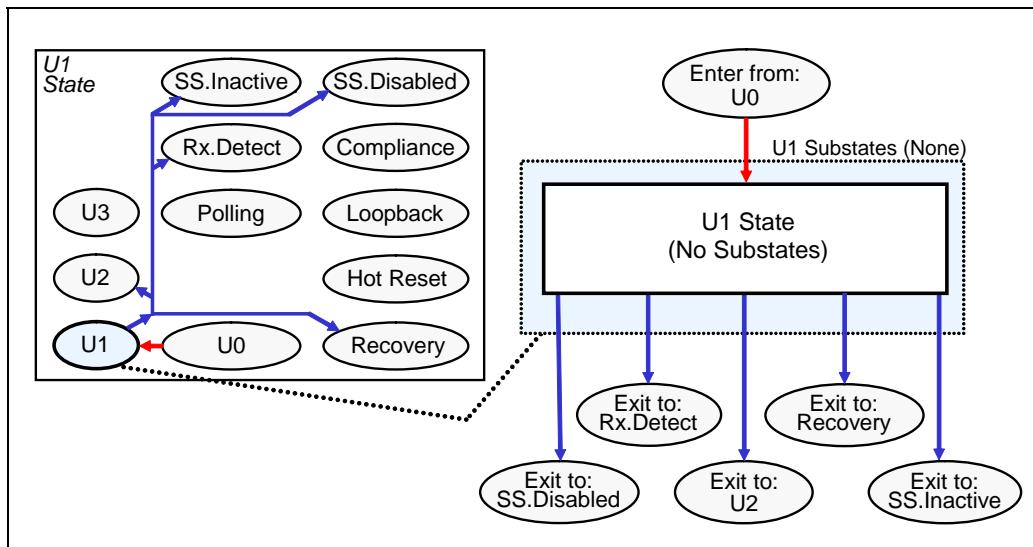
## **Chapter 12: LTSSM And the SuperSpeed Link States**

---

### **U1**

Figure 12-6 on page 259 depicts the LTSSM U1 operational state as well as the possible transitions into and out of the state. There are no substates for U1.

*Figure 12-6: LTSSM U1 State*



### **General Description**

U1 is a power management state in which the link is disabled in order to conserve power during idle times. The U1 state is assumed to conserve less power than U2, but with a shorter “exit latency” when a return to the U0 state required. USB 3.0 descriptors enable software to determine U1 and U2 exit latencies for each device. This information enables power management software to make informed power management decisions for each link.

### **Requirements In The U1 State**

The following requirements apply while an LTSSM is in the U1 state. Note that there are slightly different requirements for downstream (DS) facing ports and upstream (US) facing ports.

- Port receivers maintain low-impedance receiver termination  $R_{RX-DC}$ .
- Port transmitters must maintain SuperSpeed transmitter DC common mode

## **USB 3.0 Technology**

---

voltage as specified by  $V_{TX-CM-DC-ACTIVE-DELTA}$ .

- Ports keep LFPS detection enabled so U1 Exit LFPS signaling is recognized.
- Upstream facing ports also must detect Warm Reset LFPS signaling.
- Ports enable LFPS transmitters when exit from U1 is initiated.
- Ports enable U2 Inactivity Timer countdown on entry to U1 (if U2 Inactivity Timer value is >0).
- Downstream facing ports remain enabled for detection of Ping.LFPS and enable the 300mS  $tU1PingTimeout$  timer to verify that Ping.LFPS is regularly received from the link partner.
- Upstream facing ports transmit Ping.LFPS every 160-240 ms.

### **Exit Rules For The U1 State**

As depicted in Figure 12-6 on page 259, there are five possible exit states from the U1 State.

Exit rules observed by both upstream (US) and downstream (DS) facing ports:

- Ports silently transition to U2 if U2 Inactivity Timer is enabled and expires.
- Ports transition to Recovery if U1 Exit LFPS handshake signaling completes successfully.
- Ports transition to SS.Inactive if U1 Exit LFPS handshake signaling fails and the 2 ms  $tNoLFPSResponseTimeout$  expires. Exit rules are observed only by downstream (DS) facing ports:
  - Downstream facing ports transition to SS.Disabled when directed.
  - Downstream facing ports transition to Rx.Detect state when Ping.LFPS has not been received and 300 ms  $tU1PingTimeout$  expires.
  - Downstream facing ports transition to Rx.Detect state when directed to issue a Warm Reset on the link.

Exit rules observed only by upstream (US) facing ports:

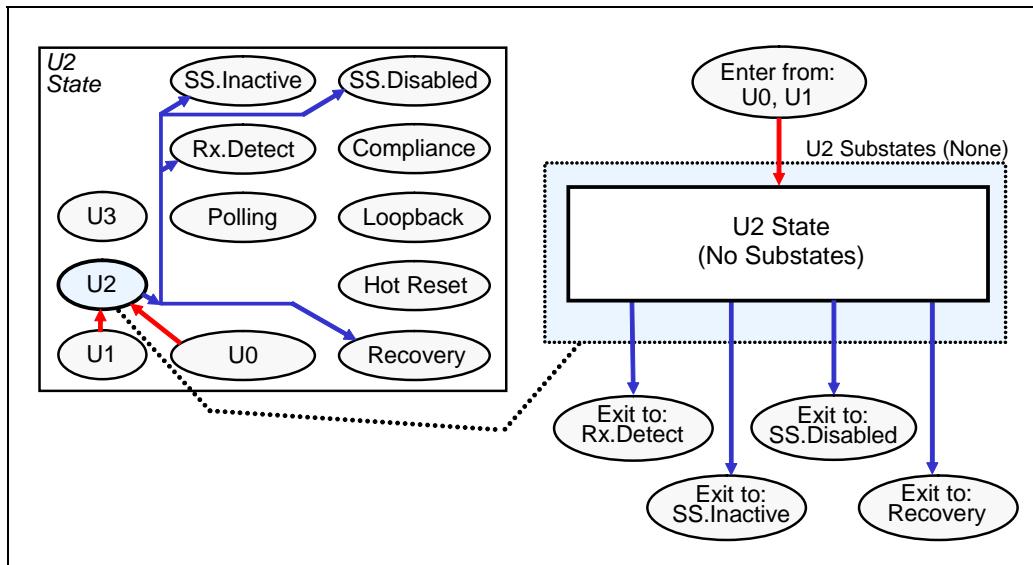
- Upstream facing ports in self-powered devices transition to SS.Disabled when  $V_{BUS}$  is off.
- Upstream facing ports transition to Rx.Detect when Warm Reset is detected.

## Chapter 12: LTSSM And the SuperSpeed Link States

### **U2**

Figure 12-7 on page 261 depicts the LTSSM U2 operational state as well as the possible transitions into and out of the state. There are no substates for U2.

*Figure 12-7: LTSSM U2 State*



#### **General Description**

U2 is a power management state in which the link is disabled in order to conserve power during idle times. The U2 state is assumed to conserve more power than U1, but with a longer “exit latency” when a return to the U0 state required. USB 3.0 descriptors enable software to determine U1 and U2 exit latencies for each device. This information enables power management software to make informed power management decisions for each link.

#### **Requirements In The U2 State**

The following requirements apply while an LTSSM is in the U2 state. Note that there are slightly different requirements for downstream (DS) facing ports and upstream (US) facing ports.

- Port transmitters are not required to maintain SuperSpeed transmitter DC

## **USB 3.0 Technology**

---

common mode voltage within specification  $V_{TX-CM-DC-ACTIVE-DELTA}$ .

- Port receivers maintain low-impedance receiver termination  $R_{RX-DC}$ .
- Ports keep LFPS detection enabled so U2 Exit LFPS signaling is recognized.
- Upstream facing ports also must detect Warm Reset LFPS signaling.
- Ports enable LFPS transmitters when exit from U2 is initiated.
- Downstream facing ports must recognize that link partner may be in either U1 or U2 state because the U2 Inactivity transition is silent. If partner is in U1, it will send Ping.LFPS every 200ms; downstream facing ports must distinguish between Ping.LFPS and U1 LFPS Exit signaling.
- Downstream facing ports perform far-end receiver termination detection every 100 ms (using  $tU2RxdetDelay$  timeout)

### **Exit Rules For The U2 State**

As depicted in Figure 12-7 on page 261, there are four possible exit states from the U2 State.

Exit rules observed by both upstream (US) and downstream (DS) facing ports:

- Ports transition to Recovery if U2 Exit LFPS handshake signaling completes successfully.
- Ports transition to SS.Inactive if U2 Exit LFPS handshake signaling fails and the 2 ms  $tNoLFPSResponseTimeout$  expires.

Exit rules observed only by downstream (DS) facing ports:

- Downstream facing ports transition to SS.Disabled when directed.
- Downstream facing ports transition to Rx.Detect state if far-end high impedance termination  $Z_{RX-HIGH_IMP_DC_POS}$  is detected.
- Downstream facing ports transition to Rx.Detect state when directed to issue a Warm Reset on the link.

Exit rules observed only by upstream (US) facing ports:

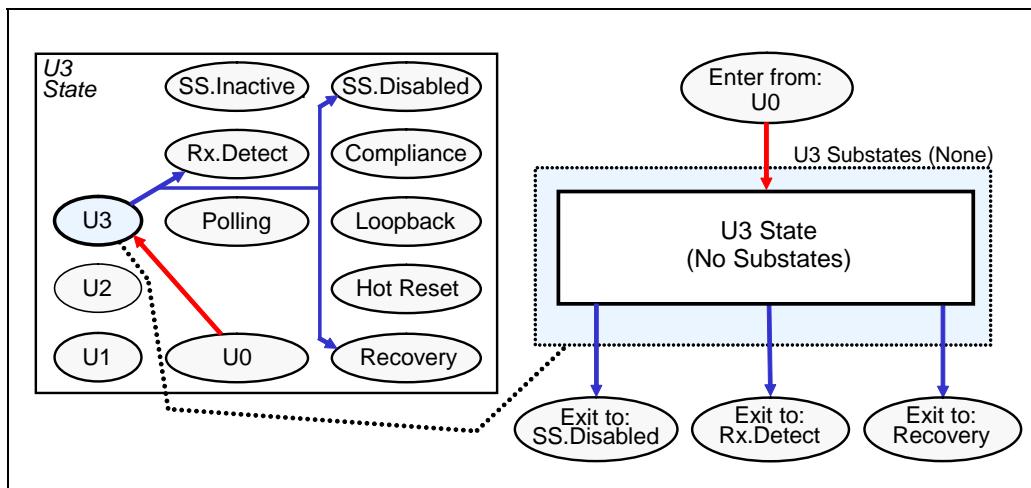
- Upstream facing ports in self-powered devices transition to SS.Disabled when  $V_{BUS}$  is off.
- Upstream facing ports transition to Rx.Detect when Warm Reset is detected.

## **Chapter 12: LTSSM And the SuperSpeed Link States**

### **U3 (Suspend)**

Figure 12-8 on page 263 depicts the LTSSM U3 operational state as well as the possible transitions into and out of the state. There are no substates for U3.

*Figure 12-8: LTSSM U3 State*



#### **General Description**

U3 is the suspend power management state, providing the greatest power savings, but involving the longest exit latency. Unlike transitions to the U1 and U2 power management states, which may be enabled to occur under hardware control following a programmed period of inactivity, U3 state entry is always software-initiated. Requests to enter U3 originate at downstream facing ports.

#### **Requirements In The U3 State**

The following requirements apply while an LTSSM is in the U3 state. Note that there are slightly different requirements for downstream (DS) facing ports and upstream (US) facing ports.

- Port transmitters are not required to maintain SuperSpeed transmitter DC common mode voltage within specification  $V_{TX-CM-DC-ACTIVE-DELTA}$ .
- Port receivers maintain low-impedance receiver termination  $R_{RX-DC}$ .
- Ports keep LFPS detection enabled so U3 Exit LFPS signaling is recognized.

## **USB 3.0 Technology**

---

- Upstream facing ports also must detect Warm Reset LFPS signaling.
- Ports enable LFPS transmitters when exit from U3 is initiated.
- Upstream facing ports do not send Ping.LFPS
- Downstream facing ports disable Ping.LFPS detection
- Downstream facing ports perform far-end receiver termination detection every 100 ms (using  $tU3RxdetDelay$  timeout)
- Unlike U1 and U2 states, failure to respond to U3 Exit LFPS signaling within the 10 ms  $tNoLFPSResponseTimeout$  does not result in a transition to SS.Inactive. If a port is unable to respond to U3 Exit LFPS signaling from its link partner within this time, it may initiate U3 LFPS wakeup when ready to return to U0. There is a delay of 100 ms that should be observed before re-attempting a wakeup ( $tU3WakeupRetryDelay$ ).

### **Exit Rules For The U3 State**

As depicted in Figure 12-8 on page 263, there are three possible exit states from the U3 State.

Exit rules observed by both upstream (US) and downstream (DS) facing ports:

- Ports transition to Recovery if U3 Exit LFPS handshake signaling completes successfully.
- Ports remain in U3 state if U3 Exit LFPS handshake fails and 10 ms LFPS handshake timeout ( $tNoLFPSResponseTimeout$ ) expires. After a delay of 100 ms ( $tU3WakeupRetryDelay$ ), the device may optionally send U3 LFPS wakeup signaling to wake up the host when it is ready to return to U0.

Exit rules observed only by downstream (DS) facing ports:

- Downstream facing ports transition to SS.Disabled when directed.
- Downstream facing ports transition to Rx.Detect state if far-end high impedance termination  $Z_{RX-HIGH\_IMP\_DC\_POS}$  is detected.
- Downstream facing ports transition to Rx.Detect state when directed to issue a Warm Reset on the link.

Exit rules observed only by upstream (US) facing ports:

- Upstream facing ports in self-powered devices transition to SS.Disabled when  $V_{BUS}$  is off.
- Upstream facing ports transition to Rx.Detect when Warm Reset is detected.

## **Chapter 12: LTSSM And the SuperSpeed Link States**

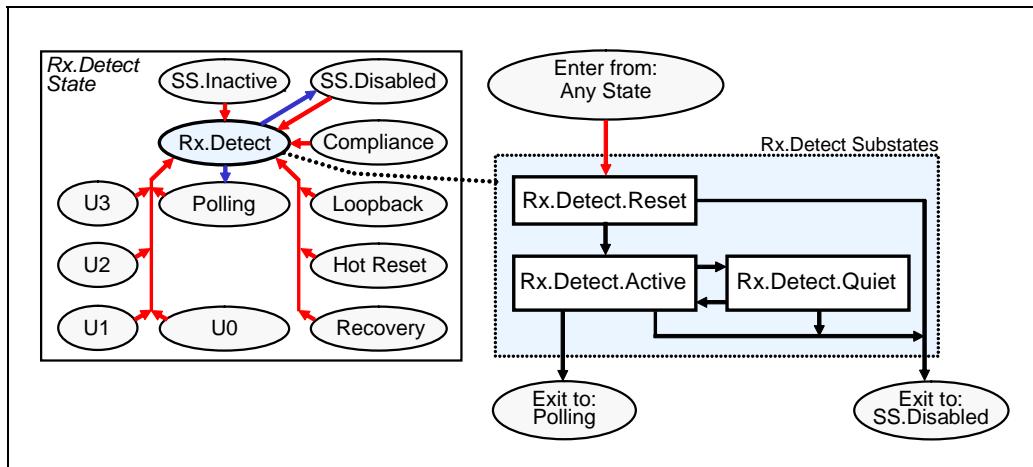
### **Summary Of LTSSM Link Initialization & Training States**

This section summarizes the Rx.Detect, Polling, Recovery, and Hot Reset LTSSM Link Initialization & Training States. Included are requirements for each substate as well as the normal and timeout exit paths from the substates. Refer to Chapter 20, entitled "Link Training," on page 427 for a more conceptual discussion of link training, recovery, and an expanded example of a generic link training sequence.

#### **Rx.Detect**

Figure 12-9 on page 265 depicts the LTSSM Rx.Detect link training state as well as the possible transitions into and out of each of its three substates.

*Figure 12-9: LTSSM Rx.Detect State*



#### **General Description**

The initial state of link training, devices start in Rx.Detect after a PowerOn Reset. It is also possible to transition to Rx.detect from other LTSSM states following certain time-outs or a Warm Reset. In the LTSSM Rx.Detect state, periodic attempts are made by each SuperSpeed transmitter to detect a link partner by checking for far-end receiver low impedance termination.

# **USB 3.0 Technology**

---

## **Requirements In The Rx.Detect.Reset Substate**

In the Rx.Detect.Reset substate, devices are waiting for PowerOn Reset initialization to complete before proceeding with link training. The following requirements apply while an LTSSM is in the Rx.Detect.Reset state. Note that there are slightly different requirements for downstream (DS) facing ports and upstream (US) facing ports.

- If entry into this substate was the result of a PowerOn Reset, then upstream and downstream ports complete the required internal initialization.
- If entry into this substate was the result of Warm Reset, the downstream facing port continues driving Warm Reset LFPS signaling for 80-100mS (*tReset*).
- If entry into this substate was the result of Warm Reset, the upstream facing port remains in this substate until Warm Reset signaling ends.

## **Exit Rules For The Rx.Detect.Reset Substate**

- If entry into this substate was the result of a PowerOn Reset, then when upstream and downstream ports complete the required internal initialization, they proceed to Rx.Detect.Active substate (no handshake)
- If entry into this substate was the result of Warm Reset, the downstream facing port transitions to Rx.Detect.Active substate (no handshake) when it completes 80-100 ms Warm Reset LFPS signaling.
- If entry into this substate was the result of Warm Reset, the upstream facing port transitions to Rx.Detect.Active substate (no handshake) Warm Reset LFPS signaling ends.

## **Requirements In The Rx.Detect.Active Substate**

In the Rx.Detect.Active substate devices attempt to confirm the presence of a SuperSpeed link partner. The scheme involves electrical detection of far-end receiver termination impedance by each transmitter. The following requirements apply while an LTSSM is in the Rx.Detect.Active state. Note that there is a slightly different requirement for downstream (DS) facing ports and upstream (US) facing ports.

- All port transmitters attempt to sense the far-end receiver termination. Specifics of the technique are described in Chapter 20, entitled "Link Training," on page 427.
- Upstream facing ports are required to count the detection attempts and will transition to SS.Disabled after eight attempts. If this occurs, the device will "fail over" and attempt connection at a USB 2.0 speed.

## **Chapter 12: LTSSM And the SuperSpeed Link States**

### **Exit Rules For The Rx.Detect.Active Substate**

As depicted in Figure 12-9 on page 265, there are three possible exit states from the Rx.Detect.Active substate.

Exit rules observed by both upstream (US) and downstream (DS) facing ports:

- Ports transition to Polling state if far-end low impedance receiver termination ( $R_{RX-DC}$ ) is detected.

Exit rules observed only by downstream (DS) facing ports:

- Downstream facing ports transition to SS.Disabled when directed.
- Downstream facing ports transition to Rx.Detect.Quiet if far-end low impedance ( $R_{RX-DC}$ ) is not detected.

Exit rules observed only by upstream (US) facing ports:

- Upstream facing hub ports transition to Rx.Detect.Quiet if far-end low impedance ( $R_{RX-DC}$ ) is not detected
- Upstream facing port of a peripheral transitions to Rx.Detect.Quiet if far-end low impedance ( $R_{RX-DC}$ ) is not detected and if number of detection attempts is less than eight.
- Upstream facing port of a peripheral transitions to SS.Disabled if far-end low impedance ( $R_{RX-DC}$ ) is not detected and the number of detection attempts is eight. The specification indicates that the eight attempt rule enables a device to “fail over” to a USB 2.0 speed within approximately 80 ms.

### **Requirements In The Rx.Detect.Quiet Substate**

In the Rx.Detect.Quiet substate devices attempt to conserve power between successive attempts to detect the presence of a link partner. This is useful because of the USB dynamic attachment/removal environment. Instead of remaining powered and checking continuously, devices in Rx.Detect transition between the Rx.Detect.Active and Rx.Detect.Quiet substates every 12 ms until a link partner is seen. During Rx.Detect.Quiet, transmitters turn off detect logic and wait until the timer expires, then try again. The following requirements apply to both upstream and downstream ports while the LTSSM is in Rx.Detect.Quiet.

- Far-end receiver termination detection is disabled
- A 12 ms timer,  $t_{RxDetectQuietTimeout}$ , is started on entry to the substate.

# USB 3.0 Technology

## Exit Rules For The Rx.Detect.Quiet Substate

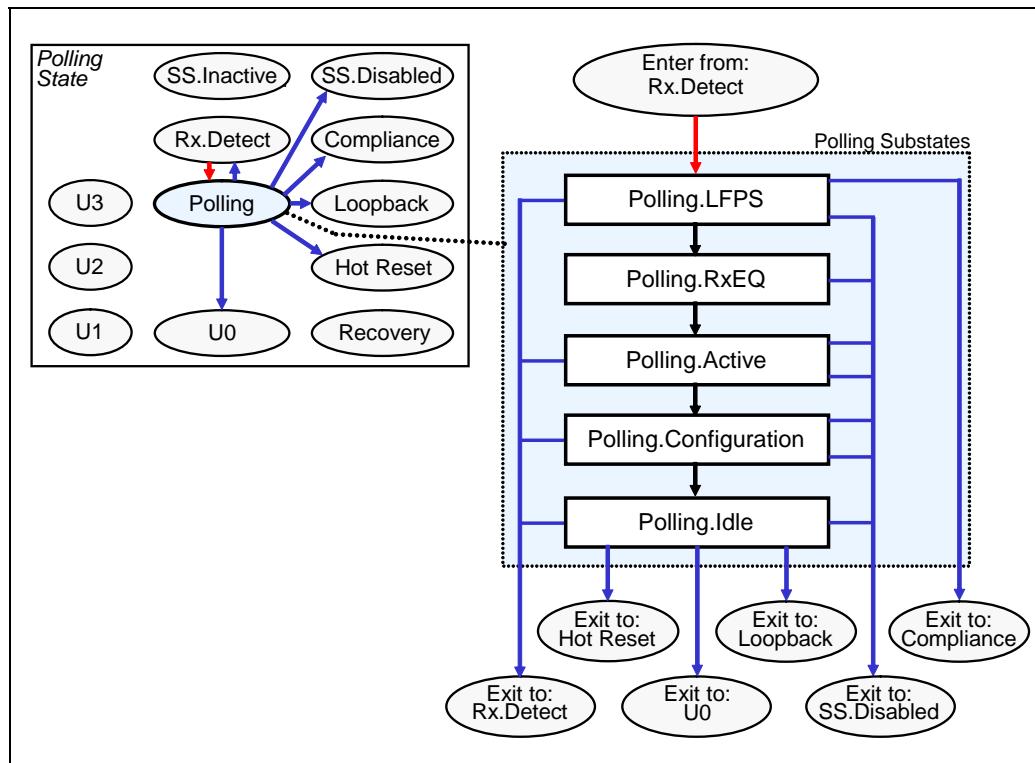
As depicted in Figure 12-9 on page 265, there are two possible exit states from the Rx.Detect.Quiet substate. The exit rules are:

- All ports transition to Rx.Detect.Active each time the 12 ms timer expires
- Downstream facing ports transition to SS.Disabled when directed.

## Polling

Figure 12-10 on page 268 depicts the LTSSM Polling link training state as well as the possible transitions into and out of each of its five substates.

Figure 12-10: LTSSM Polling State



## General Description

Polling, the second step in link training, allows devices to adapt their Super-Speed transmitters and receivers for 5 Gb/s operation under actual link con-

## **Chapter 12: LTSSM And the SuperSpeed Link States**

ditions. LFPS bursts and TSEQ, TS1, TS2 ordered sets are exchanged, enabling receiver equalization, clock and data recovery, differential signal inversion, etc.

### **Requirements In The Polling.LFPS Substate**

In the Polling.LFPS substate, devices establish receiver PHY DC operating point using the simple 10-50MHz LFPS square wave. The exchange of Polling.LFPS also starts the handshake process that will carry them through the remainder of link training. The following requirements apply to both upstream and downstream ports while LTSSMs are in the Polling.LFPS substate.

- On entry to Polling.LFPS substate all ports must enable LFPS receiver detection.
- Once a receiver detects Polling.LFPS signaling, it must establish DC operating point within 80  $\mu$ s. Receiver may optionally also use Polling.LFPS to start equalizer training.
- All ports start a 360 ms timer ( $t_{PollingLFPSTimeout}$ ) on entry to the Polling.LFPS substate.

### **Exit Rules For The Polling.LFPS Substate**

Rules observed by both upstream (US) and downstream (DS) facing ports:

- Ports transition to Polling.RxEQ when 1) at least 16 Polling.LFPS bursts have been sent, 2) two consecutive Polling/LFPS bursts received, and 3) four consecutive Polling.LFPS bursts are sent after receiving at least one Polling.LFPS burst from link partner.
- Ports transition to Compliance Mode state if the 360 ms Polling.LFPS timer ( $t_{PollingLFPSTimeout}$ ) expires and two other conditions are met: 1) the port has never completed Polling.LPFS since the last PowerOn Reset and 2) Successful Polling.LFPS handshake allowing transition to Polling.RxEQ has not occurred.

Exit rules observed only by downstream (DS) facing ports:

- Downstream facing ports transition to SS.Disabled when directed.
- Downstream facing ports transition to Rx.Detect when directed to issue a Warm Reset.
- Downstream facing ports transition to Rx.Detect if the 360 ms Polling.LFPS timer ( $t_{PollingLFPSTimeout}$ ) expires and link has been trained at least once since PowerOn Reset.

## **USB 3.0 Technology**

---

Exit rules observed only by upstream (US) facing ports:

- Upstream facing hub ports transition to Rx.Detect if the 360 ms Polling.LFPS timer ( $t_{PollingLFPSTimeout}$ ) expires and link has been trained at least once since PowerOn Reset.
- Upstream facing ports of peripherals transition to SS.Disabled if the 360 ms Polling.LFPS timer ( $t_{PollingLFPSTimeout}$ ) expires and link has been trained at least once since PowerOn Reset.
- Upstream facing ports transition to Rx.Detect when Warm Reset is detected.

### **Requirements In The Polling.RxEQ Substate**

In the Polling.RxEQ substate, devices exchange the first 5 Gb/s traffic on the SuperSpeed link. 65,536 TSEQ ordered sets (32-bytes per TSEQ ordered set) are sent and received allowing receiver PHYs to complete equalizer training. The following requirements apply to both upstream and downstream ports while LTSSMs are in the Polling.RxEQ substate.

- If lane polarity inversion is detected, each receiver must correct it.
- Ports transmit 65,536 TSEQ ordered sets.
- Receivers train equalizer logic based on received TSEQ ordered sets.

### **Exit Rules For The Polling.RxEQ Substate**

Rules observed by both upstream (US) and downstream (DS) facing ports:

- Ports automatically transition to Polling.Active after receiver equalization is complete and 65,536 TSEQ ordered sets have been sent to link partner.

Exit rules observed only by downstream (DS) facing ports:

- Downstream facing ports transition to SS.Disabled when directed.
- Downstream facing ports transition to Rx.Detect when directed to issue a Warm Reset.

Exit rules observed only by upstream (US) facing ports:

- Upstream facing ports transition to Rx.Detect when Warm Reset is detected.

## **Chapter 12: LTSSM And the SuperSpeed Link States**

### **Requirements In The Polling.Active Substate**

In the Polling.Active substate, receiver equalization is complete and devices proceed with the remainder of link training. TS1 ordered sets are used in this substate and TS2 ordered sets may also be seen as one device may transition into the next substate, Polling.Configuration, before the other exits this substate. Note that there are slightly different requirements for downstream (DS) facing ports and upstream (US) facing ports while in Polling.Active.

- Ports start a 12 ms timer (*tPolling.ActiveTimeout*) on entry.
- Ports transmit TS1 ordered sets to the link partner.
- Receivers complete any training tasks that remain then prepare to transition to Polling.Configuration substate.

### **Exit Rules For The Polling.Active Substate**

As depicted in Figure 12-10 on page 268, there are three possible exit states from the Polling.Active substate.

Exit rules observed by both upstream (US) and downstream (DS) facing ports:

- Ports transition to Polling.Configuration substate when eight consecutive and identical TS1 or TS2 ordered sets have been received.

Exit rules observed only by downstream (DS) facing ports:

- Downstream facing ports transition to SS.Disabled when directed.
- Downstream facing ports transition to Rx.Detect when directed to issue a Warm Reset.
- Downstream facing ports transition to Rx.Detect if the 12 ms timer (*tPollingActiveTimeout*) expires and requirements to transition to Polling.Configuration have not been met.

Exit rules observed only by upstream (US) facing ports:

- Upstream facing hub ports transition to Rx.Detect if the 12 ms timer (*tPollingActiveTimeout*) expires and requirements to transition to Polling.Configuration have not been met.
- Upstream facing port of peripheral transitions to SS.Disabled if the 12 ms timer (*tPollingActiveTimeout*) expires and requirements to transition to Polling.Configuration have not been met.
- Upstream facing ports transition to Rx.Detect when Warm Reset is detected.

# **USB 3.0 Technology**

---

## **Requirements In The Polling.Configuration Substate**

In the Polling.Configuration substate, the link partners signal that they have completed link training. In addition, in this substate an opportunity is provided to set TS2 Reset, Loopback, or Scrambling Disable bits. If one of these bits is set, then a transition to LTSSM states other than U0 is possible. Note that there are slightly different requirements for downstream (DS) facing ports and upstream (US) facing ports while in Polling.Configuration.

- Ports start a 12 ms timer (*tPollingConfigurationTimeout*) on entry.
- Ports transmit TS2 ordered sets to the link partner. Optionally, ports may set the (Hot) Reset bit in symbol 5 of the TS2 ordered set. Note: TS2 Reset bit is only set in TS2s sent by upstream facing ports in response detecting the Reset bit set in TS2s received from the downstream facing port.
- Ports also may optionally set the Loopback or Scrambling Disable bits in symbol 5 of TS2 ordered sets.

## **Exit Rules For The Polling.Configuration Substate**

As depicted in Figure 12-10 on page 268, there are three possible exit states from the Polling.Configuration substate.

- All ports transition to Polling.Idle substate when eight consecutive and identical TS2 ordered sets have been received and Sixteen TS2 ordered sets have been sent after receiving at least 8 consecutive and identical TS2s.

Exit rules observed only by downstream (DS) facing ports:

- Downstream facing ports transition to SS.Disabled when directed.
- Downstream facing ports transition to Rx.Detect when directed to issue a Warm Reset.
- Downstream facing ports transition to Rx.Detect if the 12 ms timer (*tPollingConfigurationTimeout*) expires and requirements to transition to Polling.Idle have not been met.

Exit rules observed only by upstream (US) facing ports:

- Upstream facing hub ports transition to Rx.Detect if the 12 ms timer (*tPollingConfigurationTimeout*) expires and requirements to transition to Polling.Idle have not been met.
- Upstream facing port of peripheral transitions to SS.Disabled if the 12 ms timer (*tPollingConfigurationTimeout*) expires and requirements to transition to Polling.Idle have not been met.
- Upstream facing ports transition to Rx.Detect if Warm Reset is detected.

## **Chapter 12: LTSSM And the SuperSpeed Link States**

### **Requirements In The Polling.Idle Substate**

In the Polling.Idle substate, the link partners decode the TS2s received in Polling.Configuration and determine what the next LTSSM state should be: U0, Loopback, or Hot Reset. Note that there are slightly different requirements for downstream (DS) facing ports and upstream (US) facing ports in Polling.Idle.

- Ports start a 2 ms timer ( $t_{PollingIdleTimeout}$ ) on entry.
- Ports decode TS2s received earlier in Polling.Configuration substate
- Downstream facing ports reset the Link Error Count (LEC)
- Upstream facing ports reset port configuration information
- Ports enable scrambling unless it was explicitly disabled in received TS2s
- If the next LTSSM state is U0, port starts transmitting (Logical) Idle symbols
- Ports prepare to receive Header Sequence Number and Flow Control Credit advertisement from the link partner.

### **Exit Rules For The Polling.Idle Substate**

As depicted in Figure 12-10 on page 268, there are five possible exit states from the Polling.Idle substate.

Exit rules observed only by both upstream (US) and downstream (DS) facing ports:

- Port transitions to Loopback when directed and device is capable of acting as Loopback Master
- Port transitions to Loopback when TS2 ordered set is received during Polling.Configuration with Loopback bit set and device is capable of acting as Loopback Slave.
- Port transitions to U0 if Loopback and Reset bits were not set in TS2 ordered sets and two conditions are met: 1) eight consecutive Idle symbols have been received and 2) sixteen consecutive Idle symbols were sent after receiving at least one Idle symbol from the link partner.

Exit rules observed only by downstream (DS) facing ports:

- Downstream facing ports transition to SS.Disabled when directed.
- Downstream facing ports transition to Rx.Detect when directed to issue a Warm Reset.
- Downstream facing ports transition to Hot Reset when directed.
- Downstream facing ports transition to Rx.Detect if the 2 ms timer ( $t_{PollingIdleTimeout}$ ) expires and requirements to transition to U0 have not been met.

# USB 3.0 Technology

---

Exit rules observed only by upstream (US) facing ports:

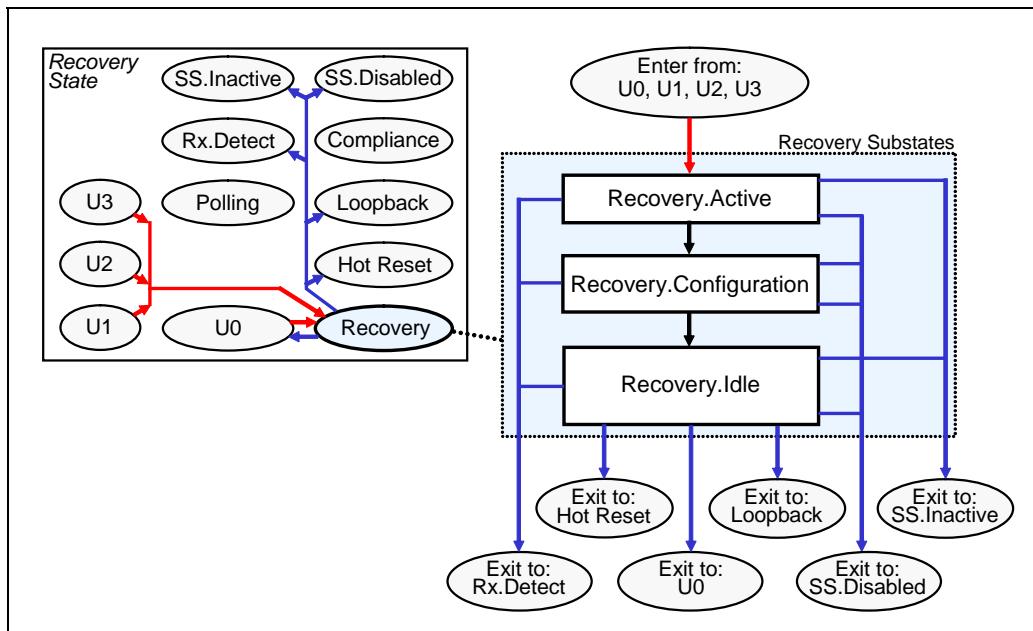
- Upstream facing hub ports transition to Rx.Detect if the 2 ms timer ( $tPollingIdleTimeout$ ) expires and requirements to transition to U0 have not been met.
- Upstream facing port of peripheral transitions to SS.Disabled if the 2 ms timer ( $tPollingIdleTimeout$ ) expires and requirements to transition to U0 have not been met.
- Upstream facing ports transition to Rx.Detect when Warm Reset is detected.

---

## Recovery

Figure 12-11 on page 274 depicts the LTSSM Recovery as well as the possible transitions into and out of the state. There are three substates in Recovery.

Figure 12-11: LTSSM Recovery



# **Chapter 12: LTSSM And the SuperSpeed Link States**

---

## **General Description**

Recovery offers low-latency retraining and return to U0 for links which have previously completed full link training. This state is entered on power management exits, because of an unrecoverable link error, etc. Because receiver equalization and other parameters are preserved, Recovery is a low latency event.

## **Requirements In Recovery.Active**

The following requirements apply while an LTSSM is in the Recovery.Active:

- Ports maintain normal transmitter and receiver specifications, including low impedance receiver termination ( $R_{RX-DC}$ ).
- A 12 ms timer ( $tRecoveryActiveTimeout$ ) is started on entry to this substate.
- Ports transmit TS1 ordered sets on entry to this state
- Ports retrain receivers using received TS1 or TS2 ordered sets

## **Exit Rules For Recovery.Active**

As depicted in Figure 12-11 on page 274, there are four possible exit states from the Recovery.Active substate.

Exit rules observed by both upstream (US) and downstream (DS) facing ports:

- Ports transition to Recovery.Configuration when eight consecutive and identical TS1 or TS2 ordered sets are received

Exit rules observed by downstream (DS) facing ports:

- Downstream facing ports transition to SS.Inactive when either Ux Exit Timer or 12 ms  $tRecoveryActiveTimeout$  expire AND the transition to Recovery was not to attempt Hot Reset.
- Downstream facing ports transition to Rx.Detect when either Ux Exit Timer or 12 ms  $tRecoveryActiveTimeout$  expire AND the transition to Recovery was to attempt Hot Reset.
- Downstream facing ports transition to Rx.Detect when directed to issue a Warm Reset
- Downstream facing ports transition to SS.Disabled when directed.

Exit rules observed by upstream (US) facing ports:

- Upstream facing ports transition to SS.Inactive when either Ux Exit Timer or 12 ms  $tRecoveryActiveTimeout$  expire.
- Upstream facing ports transition to Rx.Detect when Warm Reset is detected

# **USB 3.0 Technology**

---

## **Requirements In Recovery.Configuration**

Following requirements apply if an LTSSM is in the Recovery.Configuration:

- Ports maintain normal transmitter and receiver specifications, including low impedance receiver termination ( $R_{RX-DC}$ ).
- A 6 ms timer ( $tRecoveryConfigurationTimeout$ ) is started on entry to this substate.
- Ports transmit identical TS2 ordered sets
- If directed, downstream facing port sets one of the following bits in TS2s it sends: Reset, Loopback, or Scrambling Disable.
- If directed, upstream facing port sets either Loopback or Scrambling Disable bit in TS2s it transmits.
- Ports transmit TS1 ordered sets on entry to this state
- Ports retrain receivers using received TS1 or TS2 ordered sets

## **Exit Rules For Recovery.Configuration**

As depicted in Figure 12-11 on page 274, there are four possible exit states from the Recovery.Active substate.

Exit rules observed by both upstream (US) and downstream (DS) facing ports:

- Ports transition to Recovery.Idle if eight consecutive and identical TS2 ordered sets are received AND sixteen TS2 ordered sets were sent after receiving the first of the eight consecutive and identical TS2 ordered sets.

Exit rules observed by downstream (DS) facing ports:

- Downstream facing ports transition to SS.Inactive when either Ux Exit Timer or 6 ms  $tRecoveryConfigurationTimeout$  expire AND the transition to Recovery was not to attempt Hot Reset.
- Downstream facing ports transition to Rx.Detect when either Ux Exit Timer or 6 ms  $tRecoveryConfigurationTimeout$  expire AND the transition to Recovery was to attempt Hot Reset.
- Downstream facing ports transition to Rx.Detect when directed to issue a Warm Reset
- Downstream facing ports transition to SS.Disabled when directed.

Exit rules observed by upstream (US) facing ports:

- Upstream facing ports transition to SS.Inactive when either Ux Exit Timer or 6 ms  $tRecoveryConfigurationTimeout$  expire.
- Upstream facing ports transition to Rx.Detect when Warm Reset is detected

## **Chapter 12: LTSSM And the SuperSpeed Link States**

### **Requirements In Recovery.Idle**

Following requirements apply if an LTSSM is in the Recovery.Idle:

- A 2 ms timer ( $tRecoveryIdleTimeout$ ) is started on entry to this substate.
- Ports transmit Idle symbols if next state is U0. May optionally transmit Idle symbols if next state is Hot Reset or Loopback.
- Ports Use information decoded in TS2s link configuration fields during Recovery.Configuration to determine if next state is Loopback, Hot Reset, or U0. Transition to that state.
- Ports enable scrambling by default (unless TS2 Scrambling Disable bit was set in TS2s)
- Port prepares to receiver Header Sequence Number advertisement from link partner (if next state is U0).

### **Exit Rules For Recovery.Idle**

As depicted in Figure 12-11 on page 274, there are six possible exit states from the Recovery.Idle substate.

Exit rules observed by both upstream (US) and downstream (DS) facing ports:

- Ports transition to U0 if next state is not Loopback, Hot Reset and two conditions are met: 1) eight consecutive Idle symbols are received and 2) sixteen consecutive Idle symbols have been sent after receiving at least one.
- Port transitions to Loopback as the Loopback Slave if Loopback bit was set in TS2s during Recovery.Configuration substate.
- Port transitions to Loopback as the Loopback Master if directed and port is capable of acting as the master.
- Port transitions to SS.Inactive when either the Ux Exit timer or 2 ms tRecoveryIdleTimeout expires and requirements to transition to U0 have not been met.

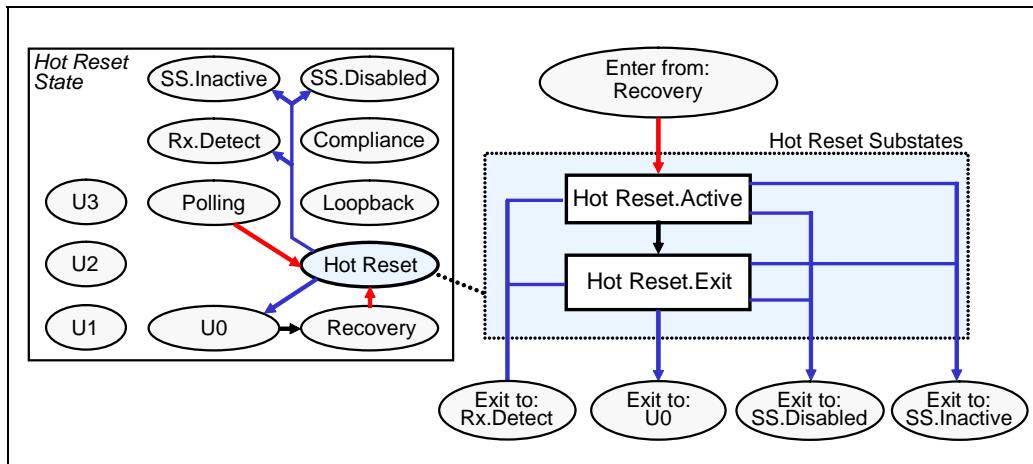
# USB 3.0 Technology

---

## Hot Reset

Figure 12-12 on page 278 depicts the LTSSM Hot Reset state as well as the possible transitions into and out of the state. There are two substates in Hot Reset.

Figure 12-12: LTSSM Hot Reset



## General Description

Hot Reset provides a low latency reset method for links that have previously been through full link training. Hot Reset avoids the lengthy Warm Reset and transition to Rx.Detect for full link training by assuming that certain physical layer (PHY) parameters established earlier are still valid. If Hot Reset is successful, link partners perform a handshake using TS2 ordered sets, initialize internal logic, and quickly return to U0. Hot Reset is always initiated by under software control by a downstream facing port.

During the Hot Reset, all ports will maintain normal transmitter and receiver electrical specifications, including the low impedance receiver termination ( $R_{RX-DC}$ ). Downstream facing port will also reset the Link Error Count (LEC), power management timers, and U1/U2 Inactivity timers.

## **Chapter 12: LTSSM And the SuperSpeed Link States**

### **Requirements In Hot Reset.Active**

The following requirements apply while an LTSSM is in Hot Reset.Active:

- A 12 ms timer (*tHotResetActiveTimeout*) is started on entry to this substate.
- Upon entry to this substate, downstream facing port will transmit at least 16 TS2s with the Reset bit asserted.
- When Hot Reset signaling is recognized by upstream facing port, it will transition to LTSSM Hot Reset state and return TS2s with Reset bit asserted.
- The exchange of TS2s with Reset bit asserted will continue until the upstream facing port has completed internal initialization and starts sending TS2s with the Reset bit de-asserted.
- When the downstream facing port recognizes TS2s with Reset de-asserted, it responds sends TS2s with Reset de-asserted in response. Ports transmit TS1 ordered sets on entry to this state

### **Exit Rules For Hot Reset.Active**

As depicted in Figure 12-12 on page 278, there are four possible exit states from the Hot Reset.Active substate.

Exit rules observed by both upstream (US) and downstream (DS) facing ports:

- Ports transition to Hot Reset.Exit when the following is true: 1) at least 16 TS2s with Reset asserted were transmitted, 2) at least 2 consecutive TS2s with Reset de-asserted were received, and 3) four consecutive TS2s with Reset de-asserted were sent after receiving at least one TS2 with Reset de-asserted.
- Ports transition to SS.Inactive if the 12 ms *tHotResetActiveTimeout* expires.

Exit rules observed by downstream (DS) facing ports:

- Downstream facing ports transition to SS.Inactive when either Ux Exit Timer or 12 ms *tRecoveryActiveTimeout* expire AND the transition to Recovery was not to attempt Hot Reset.
- Downstream facing ports transition to Rx.Detect when directed to issue a Warm Reset.
- Downstream facing ports transition to SS.Disabled when directed.

Exit rules observed by upstream (US) facing ports:

- Upstream facing ports transition to Rx.Detect when Warm Reset is detected.

# **USB 3.0 Technology**

---

## **Requirements In Hot Reset.Exit**

Following requirements apply if an LTSSM is in HotResetExit substate:

- A 2 ms timer ( $tHotResetExitTimeout$ ) is started on entry to this substate.
- Ports transmit Idle symbols and prepare for HDR Seq# advertisement.

## **Exit Rules For Hot Reset.Exit**

As depicted in Figure 12-12 on page 278, there are four possible exit states from the Hot Reset.Exit substate.

Exit rules observed by both upstream (US) and downstream (DS) facing ports:

- Ports transition to U0 when 1) eight consecutive Idle symbols are received and 2) Sixteen Idle symbols were sent after receiving at least one Idle.
- Ports transition to SS.Inactive if the 2 ms  $tHotResetExitTimeout$  expires.

Exit rules observed by downstream (DS) facing ports:

- Downstream facing ports transition to Rx.Detect when directed to issue a Warm Reset.
- Downstream facing ports transition to SS.Disabled when directed.

Exit rules observed by upstream (US) facing ports:

- Upstream facing ports transition to Rx.Detect when Warm Reset is detected.

## **Chapter 12: LTSSM And the SuperSpeed Link States**

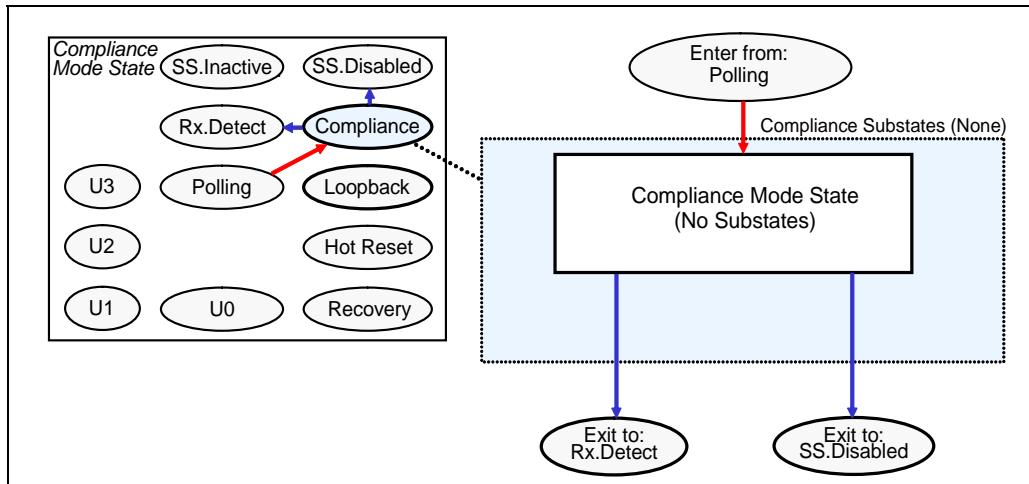
### **Summary Of LTSSM Testing States**

The following section summarizes the Compliance Mode and Loopback LTSSM testing states.

#### **Compliance Mode**

Figure 12-13 on page 281 depicts the LTSSM Compliance Mode as well as the possible transitions into and out of the state. There are no substates for Compliance.

*Figure 12-13: LTSSM Compliance Mode*



#### **General Description**

Compliance mode is entered from the Polling.LFPS substate and used to verify that SuperSpeed transmitters meet specifications for voltage and timing in the wide range of possible USB 3.0 physical channels.

# **USB 3.0 Technology**

---

## **Requirements In Compliance Mode**

The following requirements apply while an LTSSM is in Compliance Mode. Note that compliance testing generally involves a test fixture and a device under test (DUT). The DUT is the source of the nine test patterns; its link partner is responsible for sending Ping.LFPS bursts each time the test pattern is to be advanced (CP0-CP8).

- Ports maintain low-impedance receiver termination  $R_{RX-DC}$ .
- Ports enable LFPS receivers
- The port responsible for sending the compliance test patterns is required to meet the SuperSpeed transmitter DC common mode voltage specification,  $V_{TX-CM-DC-CM}$  before sending the first compliance pattern (CP1).
- Port transmits next compliance pattern each time a Ping.LFPS is detected.
- When the last compliance pattern (CP8) is active and another Ping.LFPS is received, the first pattern (CP0) is sent continuously until Compliance Mode exit is detected.

## **Exit Rules For Compliance Mode**

As depicted in Figure 12-13 on page 281, there are two possible exit states from the Compliance Mode state.

- Downstream facing ports transition to SS.Disabled when directed.
- Downstream facing ports transition to Rx.Detect state when directed to issue a Warm Reset on the link.
- Upstream facing ports transition to Rx.Detect when Warm Reset is detected.

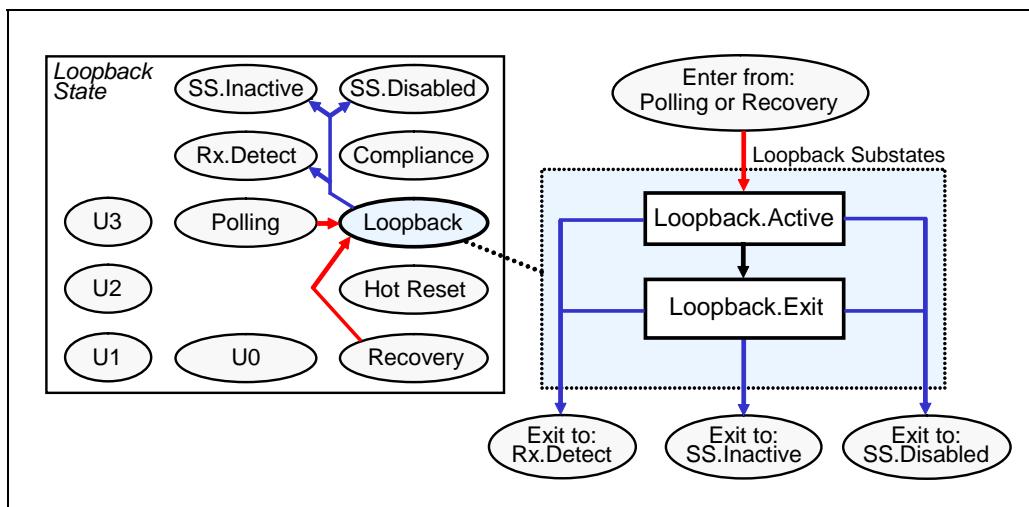
For a more detailed discussion of compliance mode testing, refer to Chapter 26, entitled "Compliance Testing," on page 617.

## Chapter 12: LTSSM And the SuperSpeed Link States

### Loopback

Figure 12-14 on page 283 depicts the LTSSM Loopback as well as the possible transitions into and out of the state. There are two substates in Loopback.

Figure 12-14: LTSSM Loopback



### General Description

Loopback testing provides a standard way to quantify the bit error rate (BER) for SuperSpeed links and help isolate errors occurring on the link itself from those related to internal device hardware problems, protocol violations, etc. There is one master and one slave in the loopback configuration. Master starts loopback by setting Loopback bit in TS2, then sources the loopback data. Slave retransmits loopback data sent to it. Master checks returned data for errors.

### Requirements In Loopback.Active

The following requirements apply while an LTSSM is in the Loopback.Active:

- Ports maintain normal transmitter and receiver specifications, including low impedance receiver termination ( $R_{RX-DC}$ ).
- Ports enable LFPS receiver
- The loopback master sends valid scrambled and encoded 10-bit symbols.

## **USB 3.0 Technology**

---

These normally consist of logical 0 (NOP) symbols.

- Slave retransmits the received 10-bit symbols as received, without modification with two exceptions: 1) receiver will correct differential lane polarity inversion and 2) receiver may add or drop SKP ordered sets as needed.
- If slave BERT state machine is supported, slave will also respond to BERT ordered sets from master which allow resetting, enabling, and retrieving 16-bit slave error count.

### **Exit Rules For Loopback.Active**

As depicted in Figure 12-14 on page 283, there are three possible exit states from the Loopback.Active substate.

- When directed, loopback master transitions to Loopback.Exit substate
- Loopback slave transitions to Loopback.Exit when Loopback Exit LFPS is detected. Loopback Exit LFPS is a 60 $\mu$ s-2mS LFPS burst.

### **Requirements In Loopback.Exit**

The following requirements apply while an LTSSM is in the Loopback.Exit:

- A 2 ms timer ( $t_{LoopbackExitTimeout}$ ) is started.
- LFPS transmitter and receiver remain enabled
- Ports transmit and receive Loopback Exit LFPS (60  $\mu$ s to 2 ms LFPS burst)

### **Exit Rules For Loopback.Exit**

As depicted in Figure 12-14 on page 283, there are three possible exit states from the Loopback.Exit substate.

- Exit rules observed by both upstream (US) and downstream (DS) facing ports:
- Ports transition to Rx.Detect if Loopback Exit handshake is successful
- Ports transition to SS.Inactive if the 2 ms timer ( $t_{LoopbackExitTimeout}$ ) expires and requirements to transition to Rx.Detect have not been met

Exit rules observed by downstream (DS) facing ports:

- Downstream facing port transitions to SS.Disabled when directed
- Downstream facing ports transition to Rx.Detect state when directed to issue a Warm Reset on the link.

Exit rules observed by Upstream (US) facing ports:

- Upstream facing ports transition to Rx.Detect when Warm Reset detected.

## **Chapter 12: LTSSM And the SuperSpeed Link States**

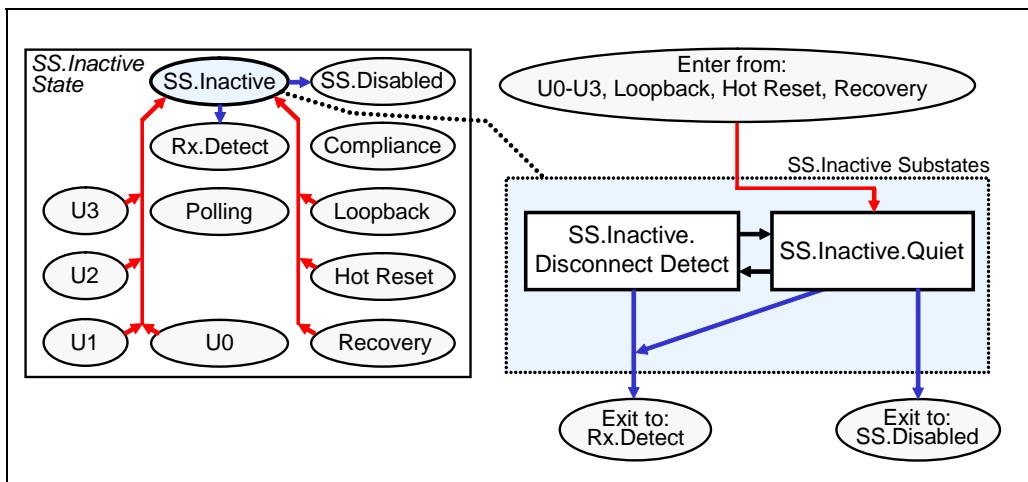
### **Summary Of Other LTSSM States**

The following section summarizes the remaining two high-level LTSSM states: SS.Inactive and SS.Disabled.

#### **SS.Inactive State**

Figure 12-15 on page 285 depicts the LTSSM SS.Inactive State as well as the possible transitions into and out of each of the two SS.Inactive substates.

*Figure 12-15: LTSSM SS.Inactive State*



#### **General Description**

The SS.Inactive state is entered as a result of an error which renders the link non-operational and software intervention is required. While link is in SS.Inactive, each port checks periodically for far-end low impedance receiver termination, ( $R_{RX-DC}$ ). If not present, a device may have been removed.

#### **Basic SS.Inactive Requirements**

- $V_{BUS}$  remains valid.
- Low impedance receiver terminations ( $R_{RX-DC}$ ) are enabled
- Transmitter common mode voltage not required to be within specification.

# **USB 3.0 Technology**

---

## **Requirements In SS.Inactive.Quiet Substate**

In this substate, link partners conserve power by switching off far-end receiver termination detection logic while waiting for software action. The following requirements apply while an LTSSM is in the SS.Inactive.Quiet substate:

- Ports start a 12 ms timer (*tSS.InactiveQuietTimeout*) on entry to this state.
- Ports disable far-end receiver termination detection logic.

## **Exit Rules For SS.Inactive.Quiet Substate**

Rules observed by both upstream (US) and downstream (DS) facing ports:

- Ports transition to SS.Inactive.Disconnect.Detect when 12 ms timer (*tSS.InactiveQuietTimeout*) expires.
- Downstream facing ports transition to SS.Disabled when directed.
- Downstream facing ports transition to Rx.Detect when directed to issue a Warm Reset.
- Upstream facing ports transition to Rx.Detect when Warm Reset is detected.

## **Requirements In SS.Inactive.Disconnect.Detect**

In this substate, link partners switch on far-end receiver termination detection logic to determine if link partner is still present. The following requirements apply while an LTSSM is in the SS.Inactive.Disconnect.Detect substate:

- Ports perform far-end receiver low impedance detect.

## **Exit Rules For SS.Inactive.Disconnect.Detect**

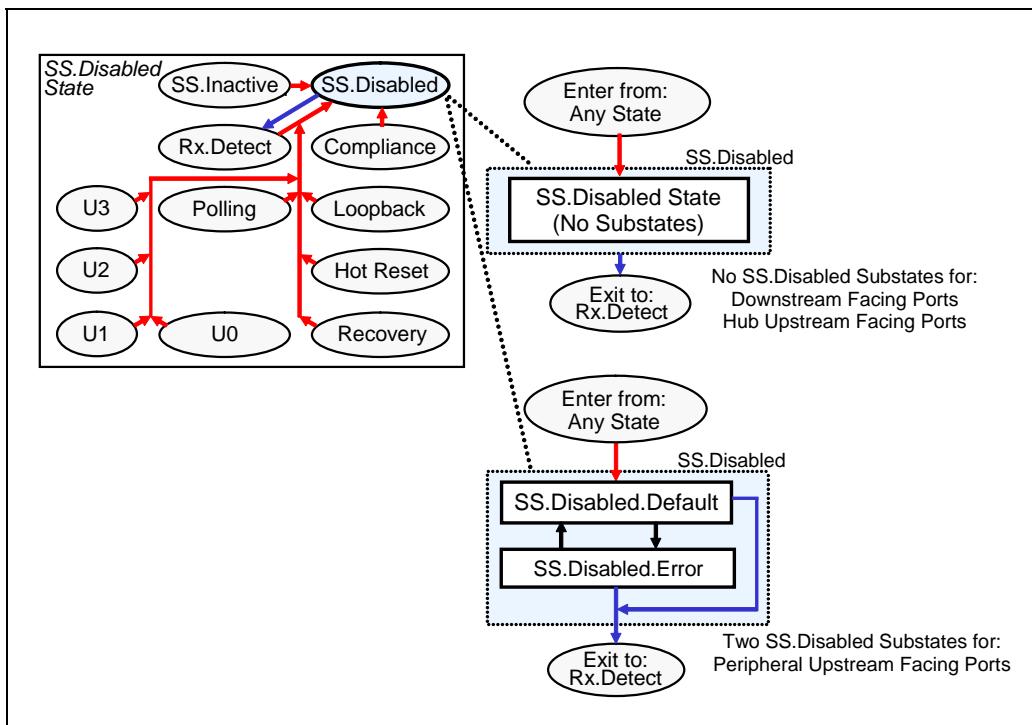
- If far-end receiver impedance meets specification R<sub>RX-DC</sub> then device is still present. Transition to SS.Inactive.Quiet Check will be made again in 12 ms.
- If far-end receiver impedance does not meet specification R<sub>RX-DC</sub> then device is no longer present. Transition to Rx.Detect.

## Chapter 12: LTSSM And the SuperSpeed Link States

### **SS.Disabled State**

Figure 12-16 on page 287 depicts the LTSSM SS.Disabled State. Note that SS.Disabled has two substates for upstream facing external hub ports.

*Figure 12-16: LTSSM SS.Disabled State*



### **General Description**

In the SS.Disabled state, SuperSpeed connectivity is disabled and the port removes its low impedance receiver termination. A downstream facing port may be directed to enter SS.Disabled from any other LTSSM state. For upstream facing peripheral ports, entry into SS.Disabled occurs during link training after eight attempts to connect. Any USB 2.0 reset causes upstream facing peripheral ports to retry SuperSpeed link training three times. If this fails, the link remains disabled for SuperSpeed. The USB 2.0 interface is still available.

## **USB 3.0 Technology**

---

Downstream ports and upstream hub ports in LTSSM SS.DISABLED observe the following rules:

- $V_{BUS}$  may be present
- Port receiver termination is high impedance ( $Z_{RX-HIGH-IMP-DC-POS}$ )
- Port is disabled from driving and receiving LFPS or 5 Gb/s signals.

### **Exit Rules For The SS.Disabled State**

- Downstream facing ports transition to Rx.Detect when directed.
- Upstream facing hub ports transition to Rx.Detect if upstream  $V_{BUS}$  transitions to valid or a USB 2.0 Reset is detected on the upstream bus.

### **Requirements For The SS.Disabled.Default Substate**

There are two SS.Disabled substates for Peripherals and the following rules apply if the LTSSM state is SS.Disabled.Default:

- $V_{BUS}$  may be present
- Port receiver termination is high impedance ( $Z_{RX-HIGH-IMP-DC-POS}$ )
- Port is disabled from driving and receiving LFPS or 5 Gb/s signals.
- Port implements a counter,  $tDisabledCount$ . This counter tracks the number of attempts the peripheral makes to connect at SuperSpeed following detection of a USB 2.0 Reset.
- $tDisabledCount$  is reset each time  $V_{BUS}$  transitions to valid or each time a USB 2.0 Reset is detected.
- $tDisabledCount$  is incremented each time SS.Disabled.Default is entered.

### **Exit Rules For The SS.Disabled.Default Substate**

- Peripheral upstream facing port transitions to Rx.Detect if  $V_{BUS}$  transitions to valid or USB 2.0 Reset is detected and  $tDisabledCount$  is less than 3.
- Peripheral upstream facing port transitions to SS.Disabled.Error if  $tDisabledCount$  is 3.

### **Exit Rules For The SS.Disabled.Error Substate**

- Peripheral upstream facing port transitions to Rx.Detect on PowerOn Reset.
- Self-powered peripheral upstream facing port transitions to SS.Disabled.Default if  $V_{BUS}$  becomes invalid.
- Peripheral upstream facing port remains in SS.Disabled.Error if a USB 2.0 Reset is detected.

---

---

# 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 its 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:

## USB 3.0 Technology

---

- 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.

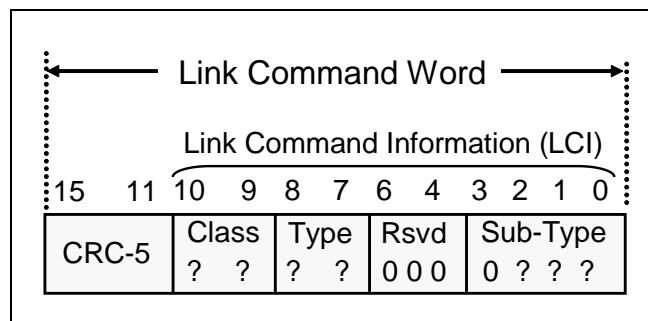
Table 13-1: Link Command Groups

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



## Chapter 13: Link Commands

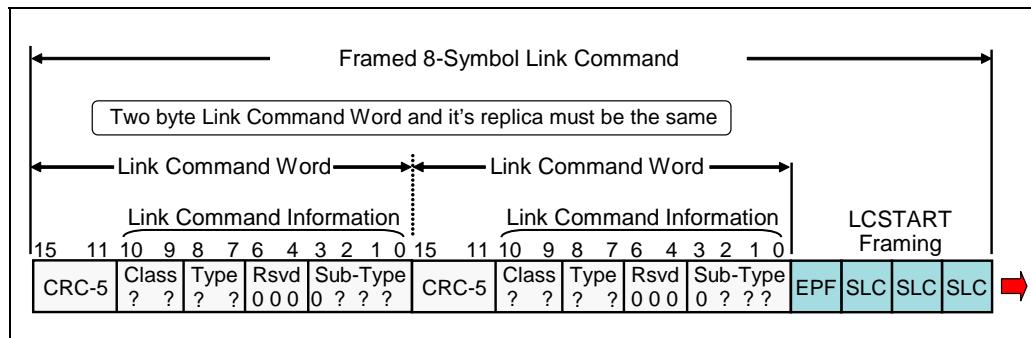
### Link Commands On The SuperSpeed Link

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
- Refer to Chapter 20, entitled "Link Training," on page 427 for details on LUP and LDN link commands.

# USB 3.0 Technology

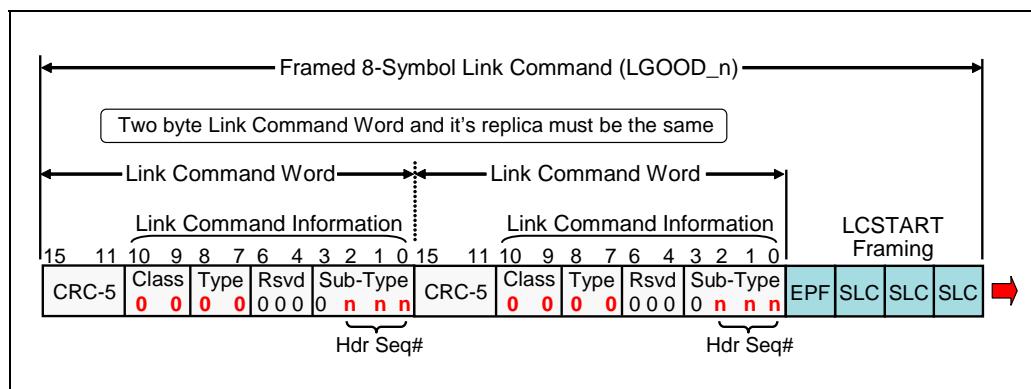
## **Link Command Encoding And Use**

The following section describes the Class, Type, and Sub-Type LCI encoding as well as the general usage model for each link command.

### **Packet Acknowledgement: LGOOD\_n**

This link command is sent by the receiver link layer to the link partner's transmit link layer to report successful delivery of a header packet. The LGOOD value "n" has a range of 0-7, indicating the Hdr Seq# of the header being acknowledged. When received, it is used by the transmitter to locate and flush the local copy from its header packet buffer (retry buffer). Figure 13-3 on page 292 depicts the LGOOD\_n link command as it appears on the SuperSpeed link.

*Figure 13-3: LGOOD\_n Link Command On The SuperSpeed Link*



The Link Command Information (LCI) fields for an LGOOD\_n link command are shown in Table 13-2 on page 293.

## Chapter 13: Link Commands

Table 13-2: LGOOD\_n Link Command Information Fields

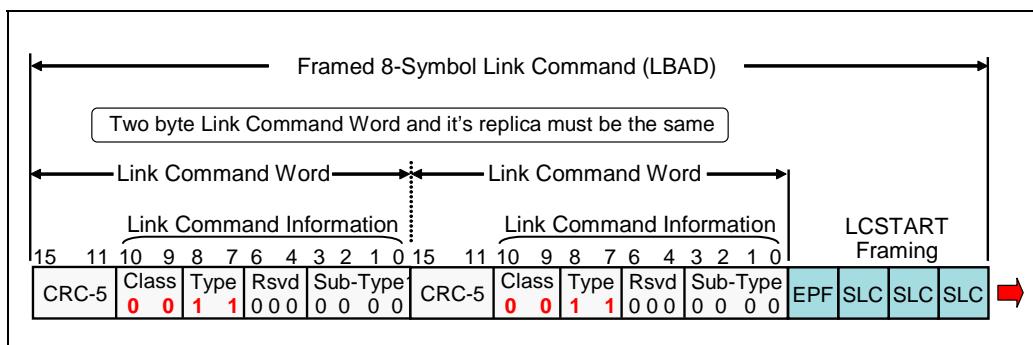
Class	Type	Sub-Type	Notes
00b	00b	R000b-R11b	The Sub-Type field identifies the Hdr Seq# of the latest header received without error. Valid range in this field is 0-7. Upper bit of LCI Sub-Type field (bit 3) of is reserved (R) and set = 0.

### Packet Acknowledgement: LBAD

This link command is sent by the receiver link layer to the link partner's transmit link layer to report that the oldest unacknowledged header in its retry buffer failed link layer CRC-5 and/or CRC-16 checks and must be resent.

When LBAD is returned by the receiver because of a failed header, the expectation is that the transmitter will re-send (retry) as quickly as possible; any packets that were in flight after the failed header will also have to be resent. Figure 13-4 on page 293 depicts the LBAD link command as it appears on the Super-Speed link.

Figure 13-4: LBAD Link Command On The SuperSpeed Link



The Link Command Information (LCI) fields for an LBAD link command are shown in Table 13-3 on page 294.

# USB 3.0 Technology

---

Table 13-3: LBAD Link Command Information Fields

Class	Type	Sub-Type	Notes
00b	11b	RRRRb	The entire Sub-Type field is reserved (R) and set = 0 in LBAD link command. Receipt of LBAD link command implies that the oldest unacknowledged header in the transmitter header packet buffer is the one received in error. Transmitter will re-send (retry) starting with the bad header.

---

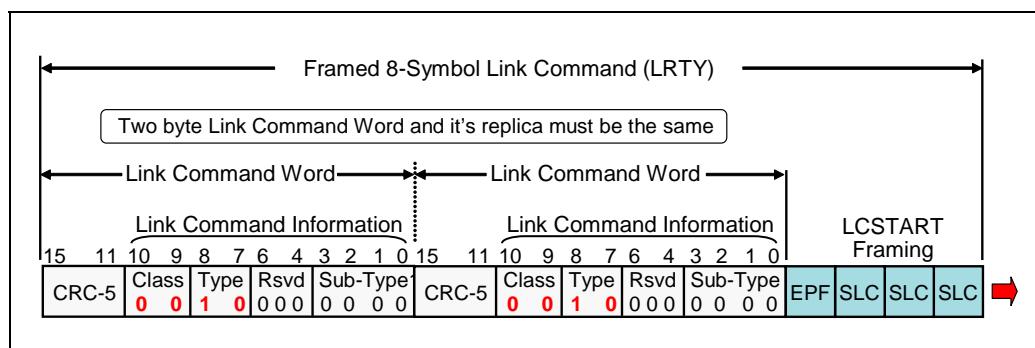
## Packet Acknowledgement: LRTY

The LRTY link command is only seen on the link if an earlier header failed CRC-5 and/or CRC-16 checks at the receiver and an LBAD link command was returned to the transmitter.

Once the transmitter recognizes an LBAD, it searches its header packet (retry) buffer for the oldest unacknowledged header and the sends LRTY link command just ahead of the header retransmission (retry). Sending LRTY helps the receiver recognize the boundary between header packets which may have been in flight after the one that failed and the beginning of the retry sequence.

Figure 13-5 on page 294 depicts the LRTY link command as it appears on the SuperSpeed link.

Figure 13-5: LRTY Link Command On The SuperSpeed Link



## Chapter 13: Link Commands

The Link Command Information (LCI) fields for an LRTY link command are shown in Table 13-4 on page 295.

Table 13-4: LRTY Link Command Information Fields

Class	Type	Sub-Type	Notes
00b	10b	RRRRb	The entire Sub-Type field is reserved (R) and set = 0 in LRTY link command. Receipt of LRTY link command at the receiver indicates that transmitter is commencing a header retry (following LBAD). The first header received after the KRTY must be the one that failed.

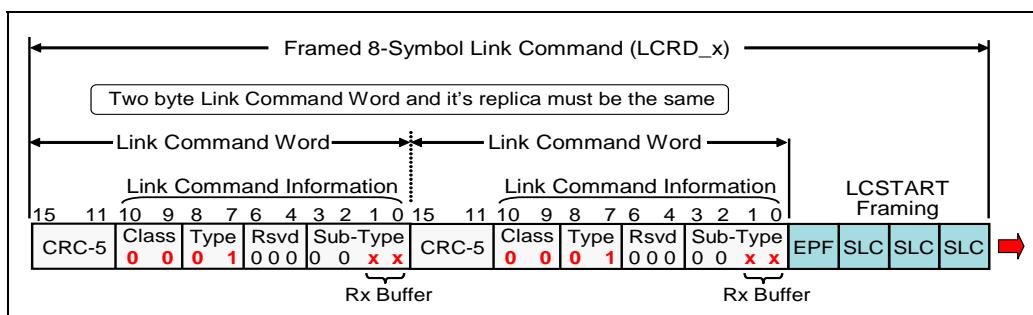
### Flow Control: LCRD\_x

It is a requirement of the USB 3.0 specification that header packets must be flow controlled. This requires receivers to advertise flow control credits to link partner transmitters using the LCRD\_x link command in several circumstances:

- Each time the link transitions to the U0 state following PowerOn Reset or Warm Reset, the receivers in link partners perform an initial advertisement of four flow control credits (corresponding to the required four Rx header packet buffers).
- Each time the link transitions to U0 from Recovery, receivers advertise flow control credits reflecting the current free space in their Rx header packet buffers (any number might be free or occupied)
- While in U0, the receiver sends a flow control credit each time a buffer becomes free (header packet has been forwarded up to protocol layer)

Figure 13-6 on page 295 depicts the LCRD\_x link command as it appears on the SuperSpeed link.

Figure 13-6: LCRD\_x Link Command On The SuperSpeed Link



# USB 3.0 Technology

The Link Command Information (LCI) fields for an LCRD\_x link command are shown in Table 13-5 on page 296.

Table 13-5: LCRD\_x Link Command Information Fields

Class	Type	Sub-Type	Notes
00b	01b	RR00b-RR11b	The Sub-Type field identifies the Rx Buffer which has become available (0 =Buffer A, 1=Buffer B, etc.). Valid range in this field is 0-3. Upper two bits of Sub-Type field (bits 3:2) are reserved (R) and set = 0.

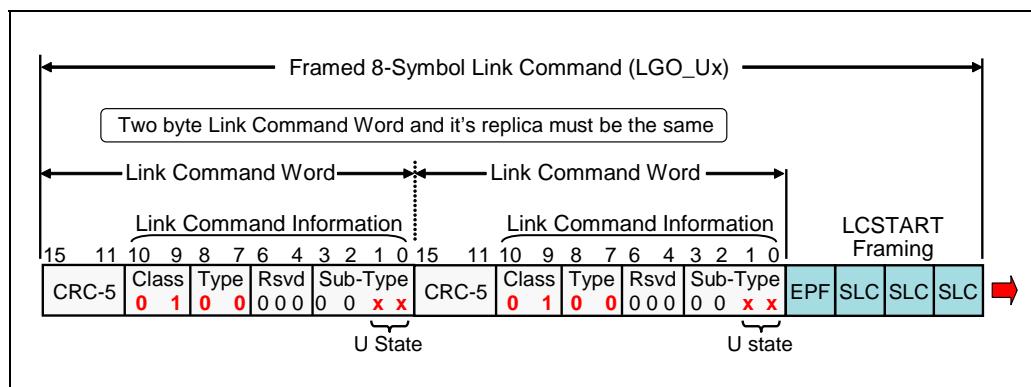
## Power Management: LGO\_Ux (The Request)

An important enhancement in USB 3.0 is link-level power management. Software initializes link partners to carry out a scheme that may enable either, both, or neither device to request a transition from U0 to U1, U2, or U3 based on a programmed period of inactivity.

Even if devices are not enabled to initiate power management transitions based on inactivity, software may command the downstream facing port of a hub to request the transition. In either case, a link command handshake is required that starts with the LGO\_Ux link command request to leave U0.

Figure 13-7 on page 296 depicts the LGO\_Ux link command as it appears on the SuperSpeed link.

Figure 13-7: LGO\_Ux Link Command On The SuperSpeed Link



## Chapter 13: Link Commands

The Link Command Information (LCI) fields for an LGO\_Ux link command are shown in Table 13-6 on page 297.

Table 13-6: LGO\_Ux Link Command Information Fields

Class	Type	Sub-Type	Notes
01b	00b	RR00b-RR11b	The Sub-Type field identifies the target U-state that the sender of this LGO_Ux link command would like to enter from the current U0 state. Valid values are 1 (U1), 2 (U2), or 3 (U3 Suspend).

### Power Management: LAU (Acceptance)

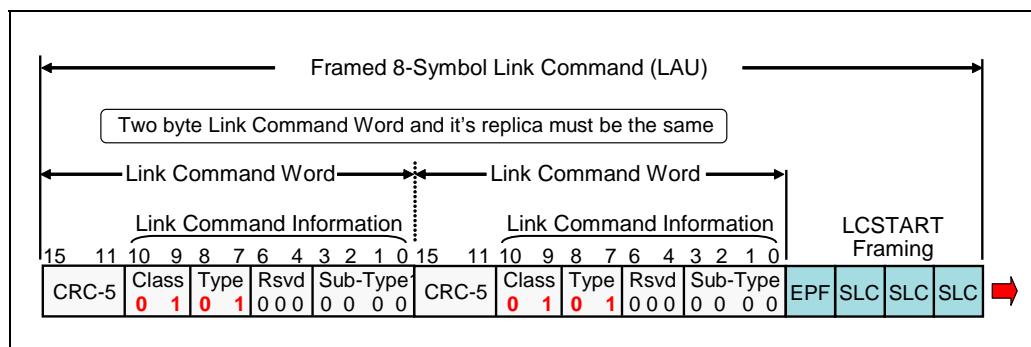
The second step in the power management transition handshake between link partners is a response from the recipient of the LGO\_Ux link command:

- The link partner may accept the transition request and indicate this choice by sending the LAU link command. This is the case shown here.
- The link partner may reject the transition from U0 to the requested state because of pending work or other reasons. In this case, it sends the LXU link command instead of LAU. This case is described next.

Note: it is always possible that a higher priority event occurs during the power management handshake (e.g. Warm Reset, Hot Reset, Recovery transition, etc.). Barring cases such as these, the handshake would continue as described here.

Figure 13-8 on page 297 depicts the LAU link command as it appears on the SuperSpeed link if the power management transition request is accepted.

Figure 13-8: LAU Link Command On The SuperSpeed Link



## USB 3.0 Technology

The Link Command Information (LCI) fields for an LAU link command are shown in Table 13-7 on page 298.

Table 13-7: LAU Link Command Information Fields

Class	Type	Sub-Type	Notes
01b	01b	RRRRb	The entire Sub-Type field is reserved (R) and set = 0 in LAU link command. Sending LAU informs the sender of LGO_Ux that the link partner is prepared to proceed with the transition to the new power management state.

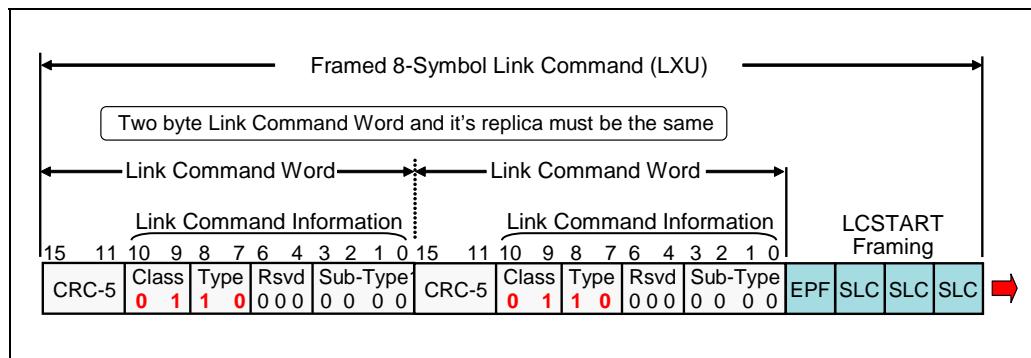
### Power Management: LXU (Rejection)

In the event that the recipient of the LGO\_Ux link command is not prepared to accept the transition from U0 (because of pending work or other reasons), it may return the LXU link command rejecting the request.

Note: it is always possible that a higher priority event occurs during the power management handshake (e.g. Warm Reset, Hot Reset, Recovery transition, etc.). Barring cases such as these, the handshake would continue as described here.

Figure 13-9 on page 298 depicts the LXU link command as it appears on the SuperSpeed link if the power management transition request is rejected.

Figure 13-9: LXU Link Command On The SuperSpeed Link



## Chapter 13: Link Commands

The Link Command Information (LCI) fields for an LXU link command are shown in Table 13-8 on page 299.

Table 13-8: LXU Link Command Information Fields

Class	Type	Sub-Type	Notes
01b	10b	RRRRb	The entire Sub-Type field is reserved (R) and set = 0 in LXU link command. Sending LXU informs the sender of LGO_Ux that the link partner is not prepared to proceed with the transition to the new power management state.

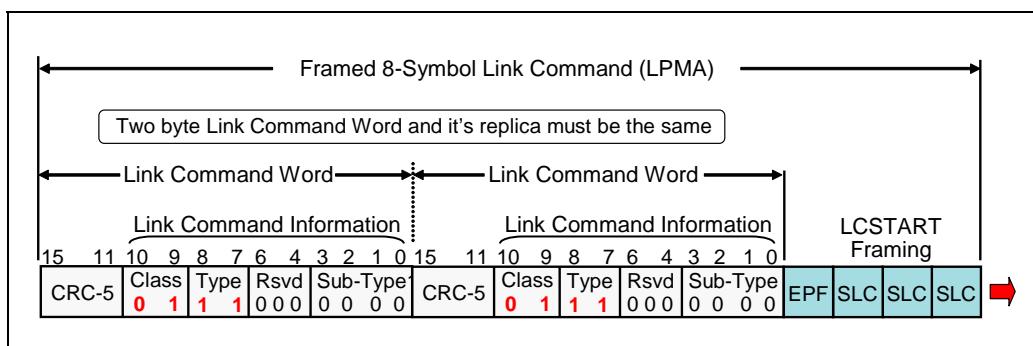
### Power Management: LPMA (Acknowledgement)

The final link command in the power management transition request handshake is LPMA. This link command is returned by the original requester (sender of LGO\_Ux) to its link partner after it has received either the LAU or LXU link command response. LPMA indicates that the requester recognized and will honor the response.

In the event that the recipient of the LGO\_Ux link command responded with the LAU link command (accepting the request), receipt of LPMA indicates that the exit to U1, U2, or U3 can now be made.

Figure 13-10 on page 299 depicts the LPMA link command as it appears on the SuperSpeed link. Again, LPMA is the final link command of the sequence.

Figure 13-10: LPMA Link Command On The SuperSpeed Link



## USB 3.0 Technology

---

The Link Command Information (LCI) fields for an LPMA link command are shown in Table 13-9 on page 300.

Table 13-9: LPMA Link Command Information Fields

Class	Type	Sub-Type	Notes
01b	11b	RRRRb	The entire Sub-Type field is reserved (R) and set = 0 in LPMA link command. Sending LPMA informs the sender of LAU or LXU that the original requester has recognized and will honor the response.

---

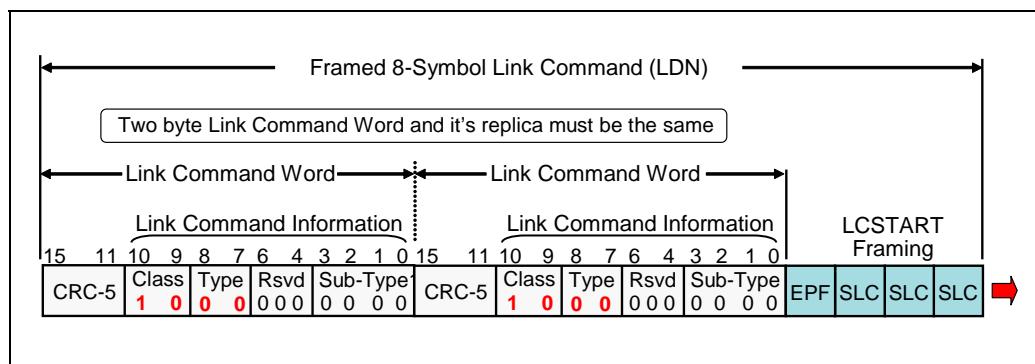
### Link Up/Link Down: LDN

The LDN (link downstream) link command is sent by downstream facing ports of root and external hubs to indicate that the port is still active and in the U0 state.

LDN is only sent in the absence of other packets and link commands; it is triggered on a 10uS timeout without any link layer or protocol layer traffic.

Figure 13-11 on page 300 depicts the LDN link command as it appears on the SuperSpeed link.

Figure 13-11: LDN Link Command On The SuperSpeed Link



The Link Command Information (LCI) fields for an LDN link command are shown in Table 13-10 on page 301.

# Chapter 13: Link Commands

Table 13-10: LDN Link Command Information Fields

Class	Type	Sub-Type	Notes
10b	00b	RRRRb	The entire Sub-Type field is reserved (R) and set = 0 in LDN link command. If there is no other link traffic, LDN is sent every 10uS to inform the downstream link partner that the upstream device is still active and is in the U0 state.

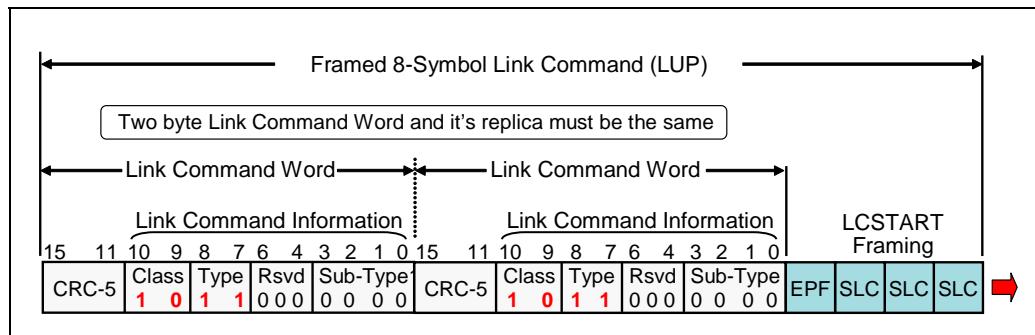
## Link Up/Link Down: LUP

The LUP (link upstream) link command is the complement to the LDN link command described previously. It is sent by upstream facing ports of peripherals and external hubs to indicate that the port is still active and in the U0 state.

LUP is only sent in the absence of other packets and link commands; it is triggered on a 10uS timeout without any link layer or protocol layer traffic.

Figure 13-12 on page 301 depicts the LUP link command as it appears on the SuperSpeed link.

Figure 13-12: LUP Link Command On The SuperSpeed Link



## **USB 3.0 Technology**

---

The Link Command Information (LCI) fields for an LUP link command are shown in Table 13-11 on page 302.

*Table 13-11: LUP Link Command Information Fields*

Class	Type	Sub-Type	Notes
10b	11b	RRRRb	The entire Sub-Type field is reserved (R) and set = 0 in LUP link command. If there is no other link traffic, LUP is sent every 10uS to inform the upstream link partner that the downstream device is still active and is in the U0 state.

---

### **Notes On Link Command Placement**

On a SuperSpeed link, link commands are interleaved with header packets, data packets, SKP ordered sets, other link commands, and logical idle symbols. A few rules apply when link commands are sent with other traffic:

- Link commands (and other packets) are never placed within the structures of header packets, data packets, or other link commands.
- Link commands may be sent back to back
- Link commands may not be placed between a data packet header (DPH) and its data packet payload (DPP).
- Link commands may not be transmitted until any scheduled SKP ordered sets have been sent.

---

---

# 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.

---

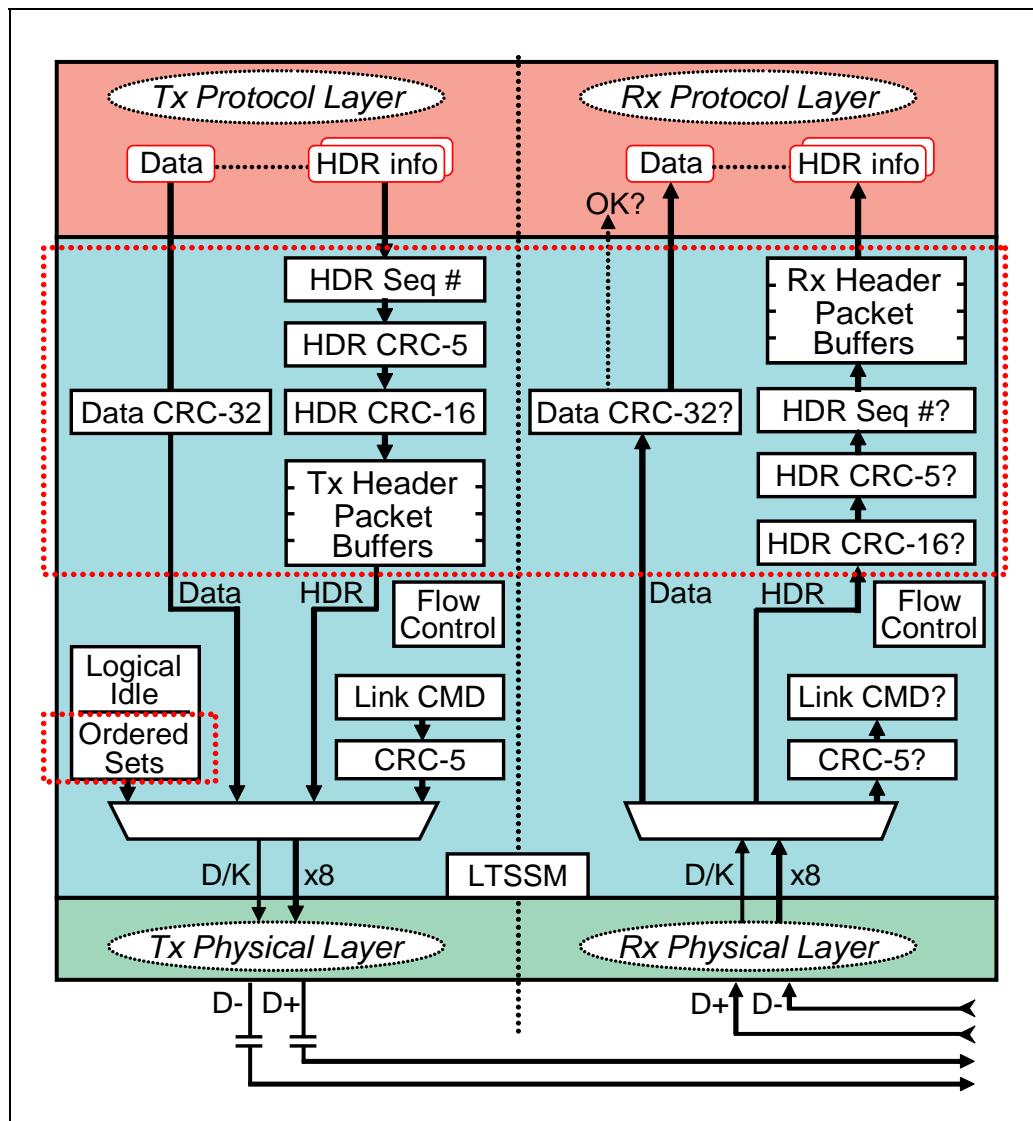
## Link Layer Packet Processing Role

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.

## USB 3.0 Technology

---

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



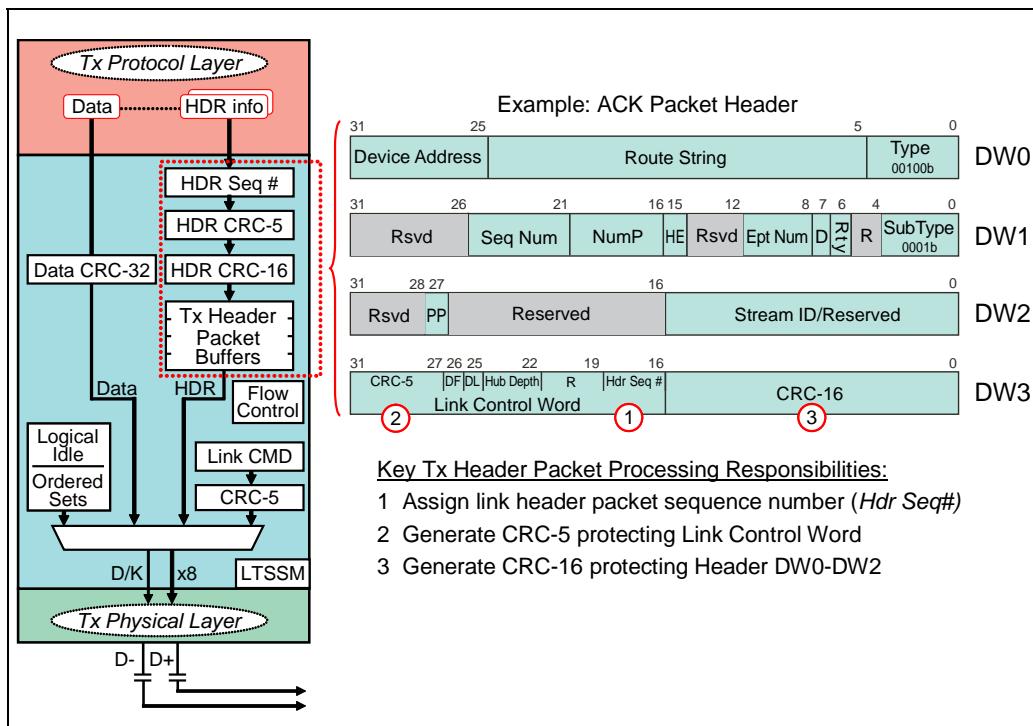
## Chapter 14: Header Packet Processing

### Link Layer 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 be 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



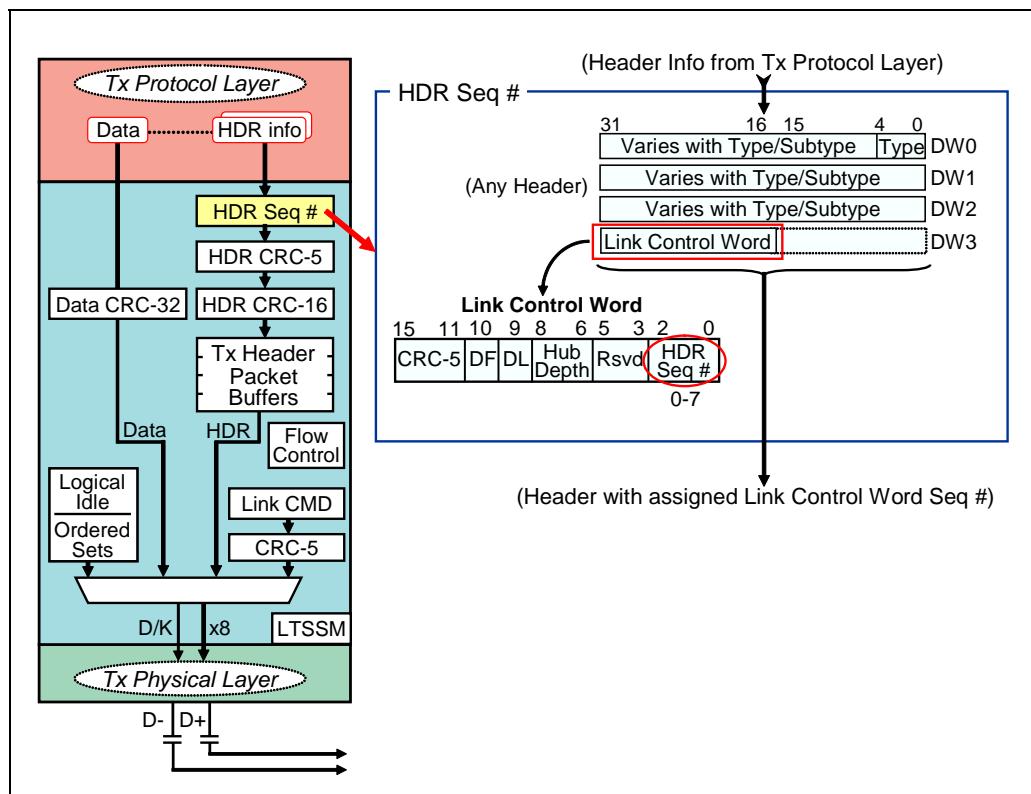
# USB 3.0 Technology

## Header Sequence Number Assignment

As indicated in Figure 14-3 on page 306, one of the first Tx processing steps is assignment of the 3-bit header packet link sequence number (Hdr Seq#). This number starts at 0, increments by one for each header packet sent, and rolls over to 0 for each eight header packets sent. It is a central component in link level error handling, as the receiver must acknowledge that each header packet was valid and received in order.

In the event of an error transmitting a header packet, the receiver will send a link command (LBAD) indicating which specific packet failed and requesting a retry. For details on link level error handling, refer to Chapter 16, entitled "Link Errors & Packet Acknowledgement," on page 339.

Figure 14-3: Transmitter Assigns Header Packet Sequence Number (HDR Seq#)

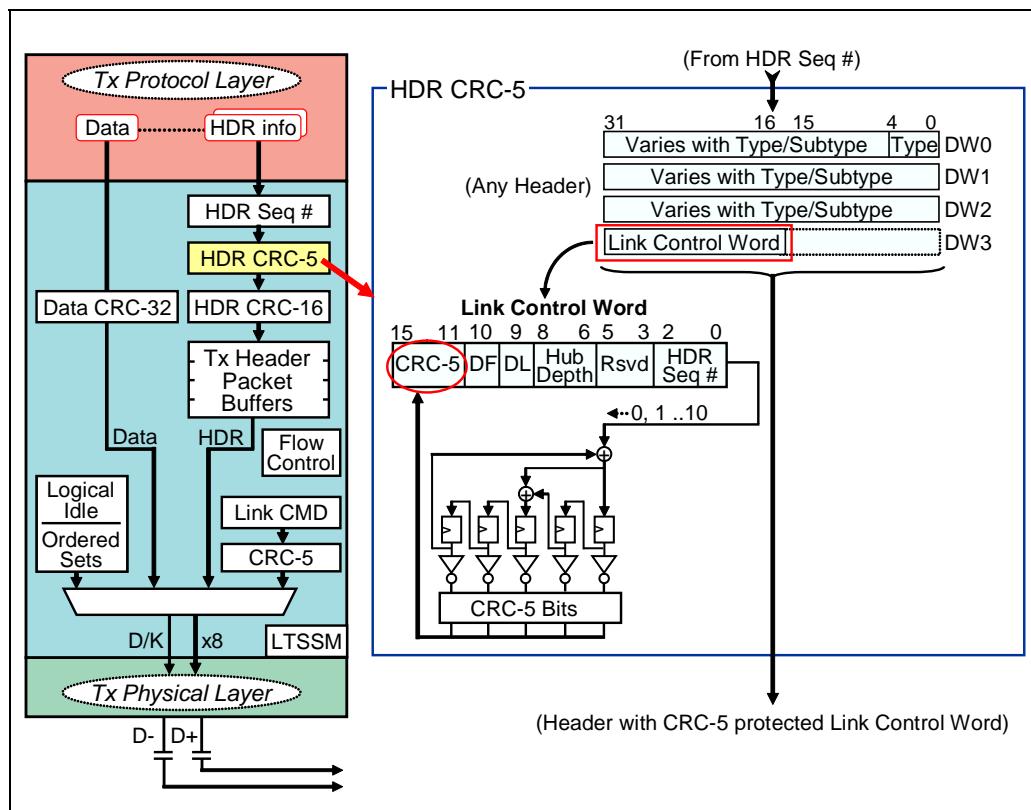


## Chapter 14: Header Packet Processing

### Header Link Control Word CRC-5 Generated

In this step, the transmitter generates a CRC (Cycle Redundancy Check) protecting bits 0-10 of the Link Control Word in header DW3. As depicted in Figure 14-4 on page 307, this CRC-5 is generated in hardware. Once it is calculated, the CRC-5 is placed in bits 11-15 of the Link Control Word.

Figure 14-4: Transmitter Generates Header Link Control Word CRC-5



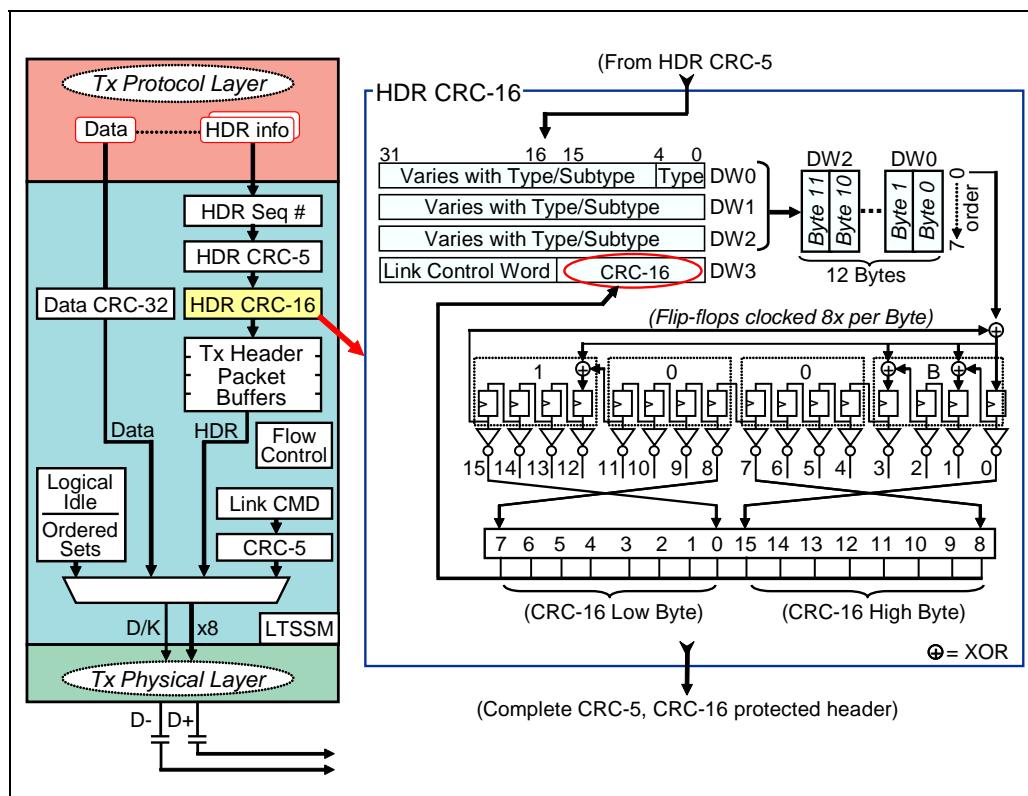
# USB 3.0 Technology

## Header CRC-16 Generated

Here, the transmitter generates a CRC -16 protecting the lower 12 bytes (DW0-DW2) of the header. As depicted in Figure 14-5 on page 308, CRC-16 is also generated in hardware. There is an advantage in having an independent CRC-5 protecting the Link Control Word and this CRC-16 protecting DW0-DW2. In cases where hubs are present and the header packet must travel across multiple links, the Link Control Word must be updated at the link layer of each port to reflect the Hdr Seq# for that link (the CRC-5 must also be regenerated on each link).

On the other hand, the information in DW0-DW2 is associated with End-To-End protocol and does not change if the header packet crosses another link (the CRC-16 also is unchanged).

Figure 14-5: Transmitter Generates Header CRC-16



## Chapter 14: Header Packet Processing

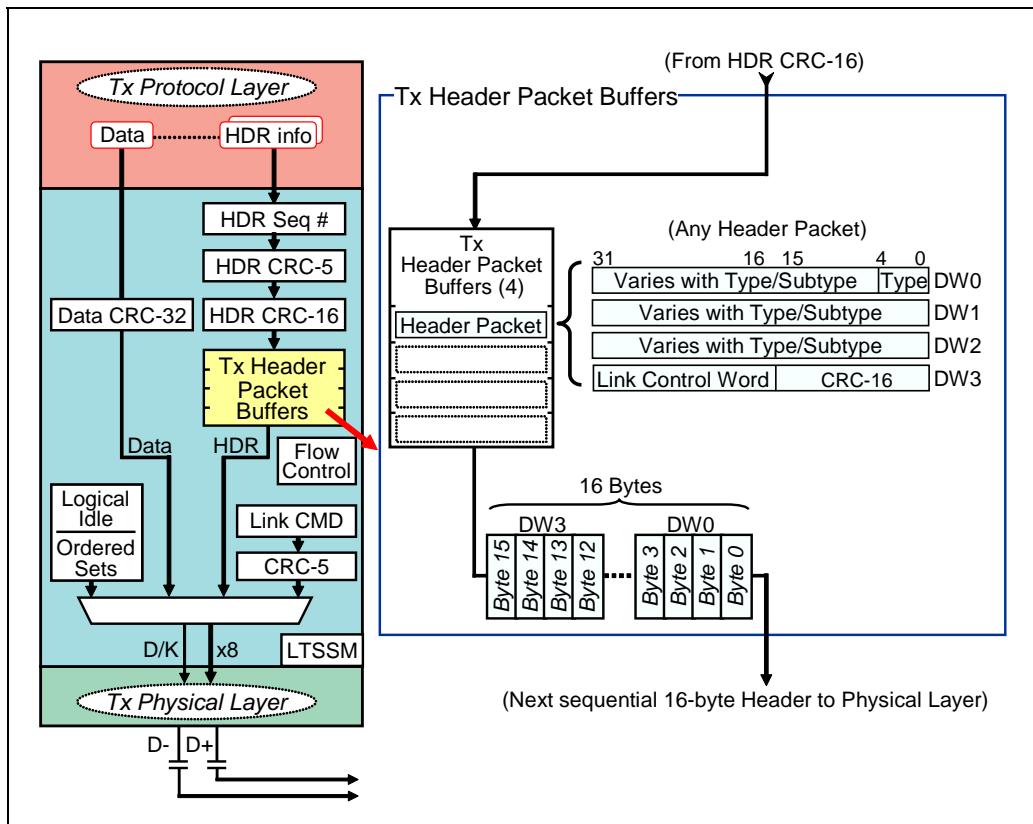
### Copy Is Placed In A Header Packet Buffer

At this point, assembly of the header packet is complete. The Tx link layer is obligated to maintain a copy until the header packet is acknowledged by the link partner receiver. In the event of an error, the link layer transmitter may retry sending a failed header up to three times before the link must be retrained.

The USB 3.0 specification requires that transmitters must be capable of buffering up to four header packets. Figure 14-6 on page 309 depicts the transmitter header packet buffers (also referred to as Tx HP Buffers).

Refer to Chapter 16, entitled "Link Errors & Packet Acknowledgement," on page 339 for details on transmitter header packet buffer management.

Figure 14-6: Header Packet Buffers Hold A Copy Of Each Header



# USB 3.0 Technology

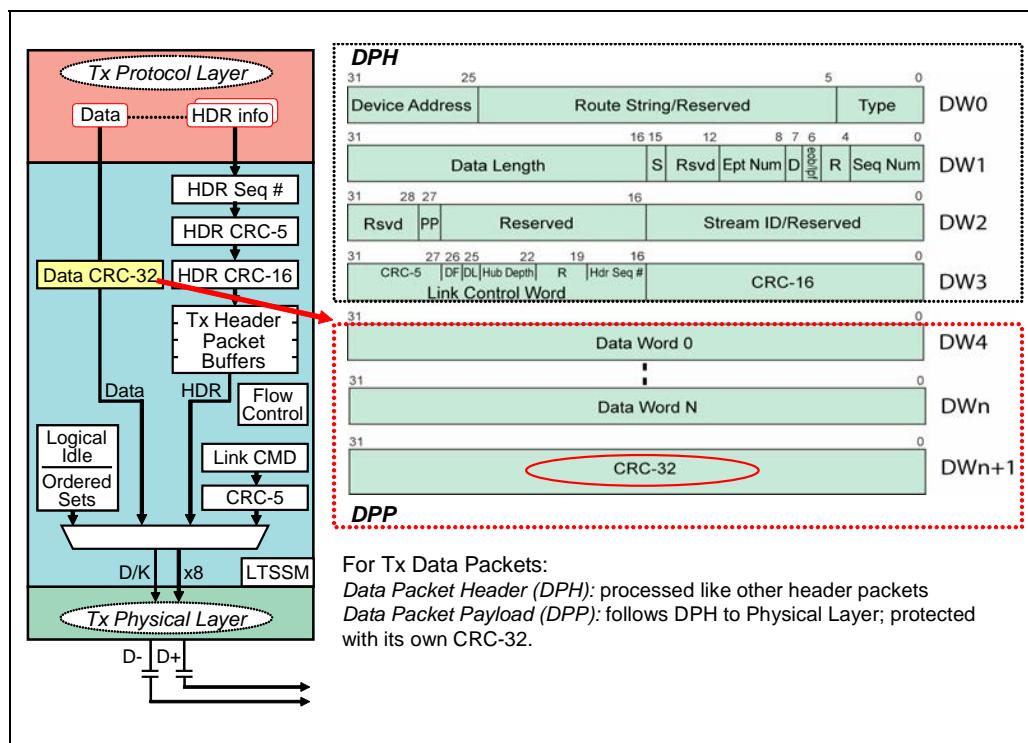
## Data Packet Payload CRC-32 Generation

### General

As indicated in Figure 14-7 on page 310, a data packet has both a data packet header (DPH) and data packet payload (DPP). The DPH has much the same format and is processed in the same way as other headers in terms of the Hdr Seq#, CRC-5, and CRC-16. When the transmitter recognizes that a data packet is being processed, it is known that the DPP will immediately follow the DPH to the link (no gap is permitted between the DPH and the DPP)

A CRC-32 must be generated for each data packet payload (DPP). The position of the CRC-32 in the DPP is immediately after the last Data Word (DW). While the USB 3.0 specification indicates that it is a protocol layer responsibility to acknowledge data payloads, the hardware logic to generate and check CRC-32 is shown here (and in the specification) at the link layer.

Figure 14-7: Data Packet Payload Requires CRC-32 Generation

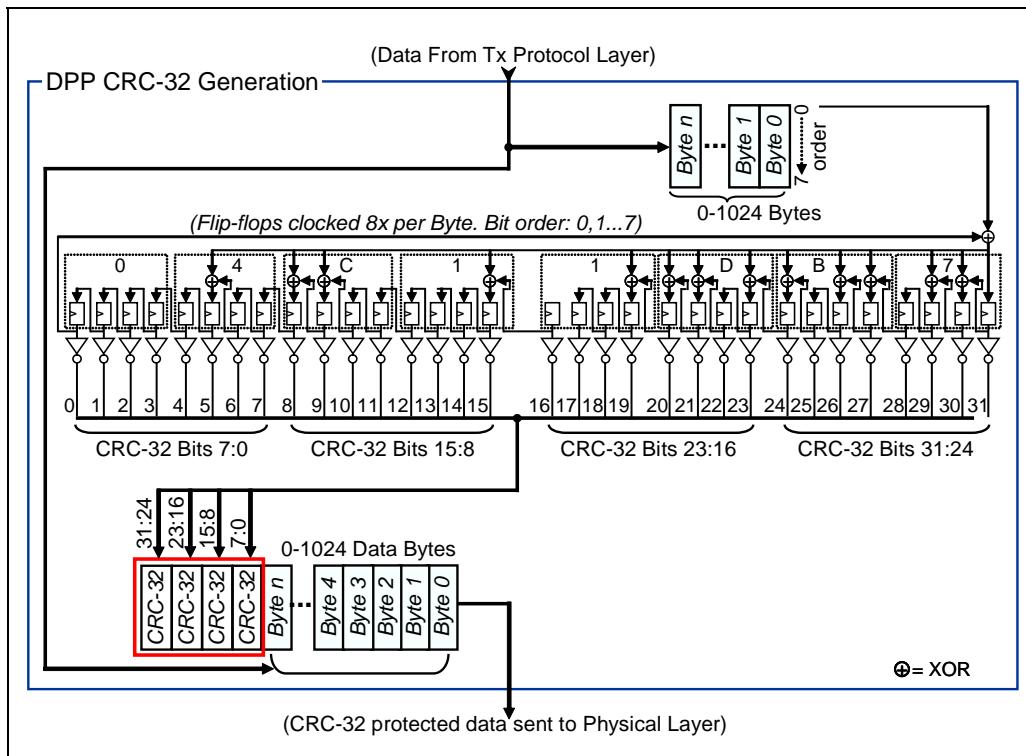


## Chapter 14: Header Packet Processing

### Tx DPP CRC-32 Generation, The Hardware Details

Figure 14-8 on page 311 illustrates the hardware defined in the specification for generating the DPP CRC-32. As shown, each DW of the payload is shifted into the logic, starting with bit 0 of the low byte in DW0, and ending with bit 7 of the last byte, last DW. The four bytes of CRC-32 then follow the data to the physical layer for transmission.

Figure 14-8: Transmitter's DPP CRC-32 Generation Logic

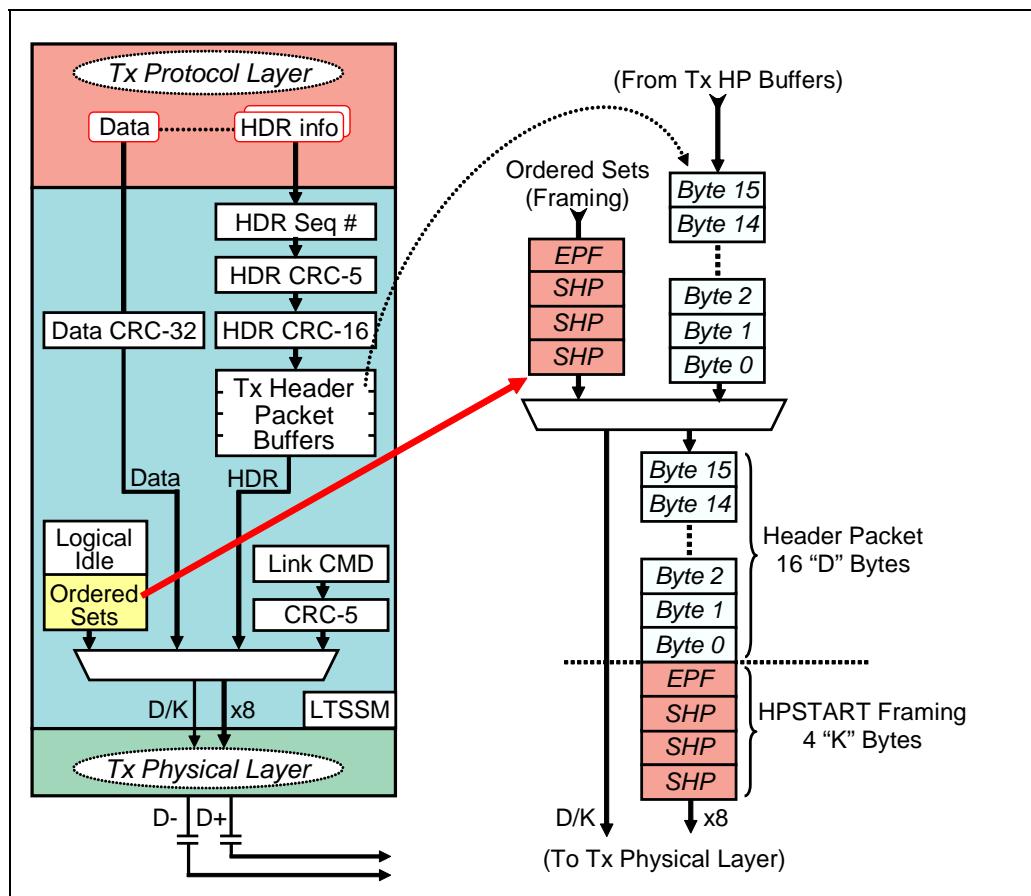


# USB 3.0 Technology

## Framing Added, Packet Sent To Physical Layer

Regardless of whether a header or data packet is to be transmitted, the final processing step by the transmitter link layer is to append the packet framing. Header packet start framing (HPSTART) is a set of four ordered set “K” symbols: SHP, SHP, SHP, and EPF. Figure 14-9 on page 312 depicts a framed header packet. The transmitter forwards the framed header packet to the physical layer one byte at a time with an extra bit, “D/K”. The D/K bit indicates which bytes are “K” (control) and which are “D” (data--the 16 header bytes).

Figure 14-9: Transmitter Adds Ordered Set Framing To Outbound Packets



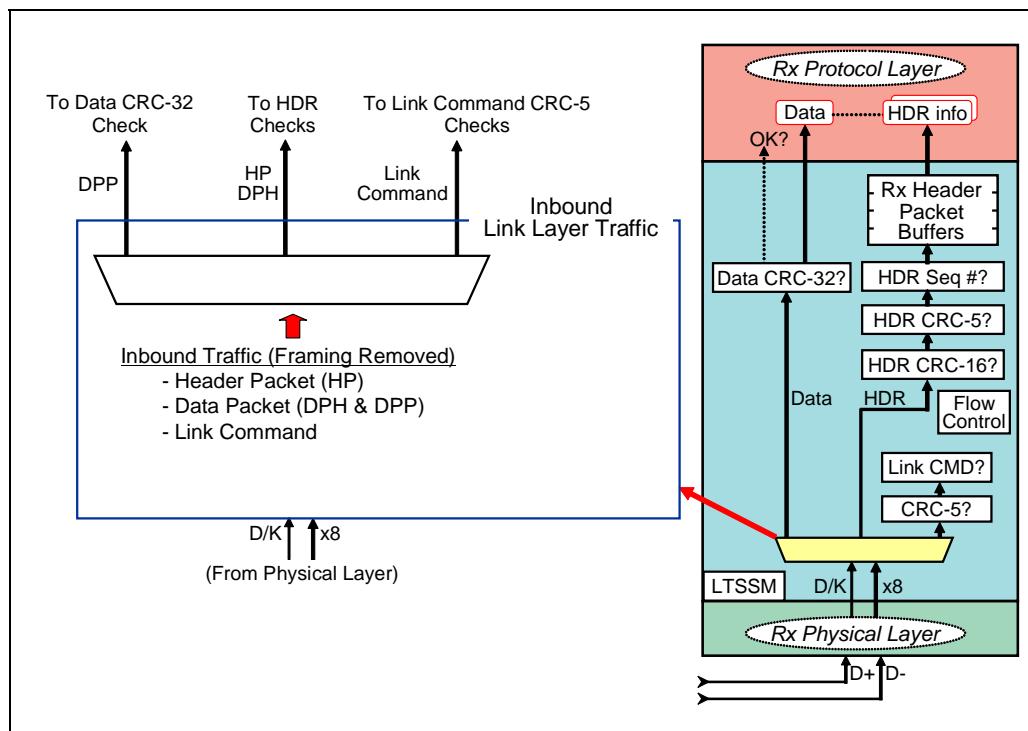
# Chapter 14: Header Packet Processing

## Receiver Packet Processing

### Inbound Link Layer Traffic

While the focus in this chapter is on header packet processing, it is still useful to remember that traffic on the link consists of a mix of interleaved header packets, data packets, link commands, ordered sets, and logical idle symbols. The physical layer removes framing and filters other ordered set "K" symbols. As shown in Figure 14-10 on page 313, it ultimately delivers header packets, data packets, and link commands to the receiver link layer for processing.

Figure 14-10: Receiver Accepts Inbound Packets And Link Commands



# USB 3.0 Technology

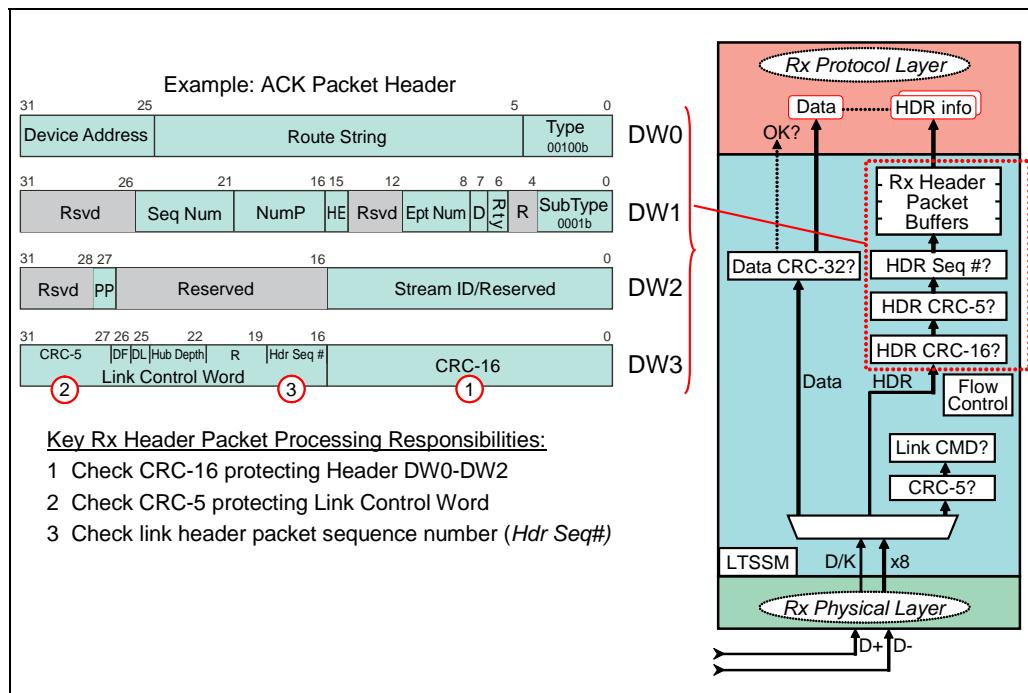
## Key Header Packet Processing Elements

Based on HPSTART framing, the receiver is aware of where headers start and end. It processes and attempts to recover the 16-byte header packet structure sent by the link partner. If the physical layer detects DPPSTART framing immediately after a header is processed, a data packet is in progress and the receiver prepares to check the DPP CRC-32; results will be passed to the protocol layer.

Figure 14-11 on page 314 illustrates key elements of the Rx header packet processing. In the example, an ACK transaction packet is to be checked before being accepted into the Rx Header Packet Buffers. At some point, each valid header will be forwarded from the Rx Header Packet Buffers to the receiver's protocol layer. At the protocol layer, the fields of interest include:

- The contents of header DW0-DW2
- Three fields in the Link Control Word: *Hub Depth*, *DL*, and *DF*. Of course, if a data packet was sent, the DPP is also forwarded to the protocol layer.

Figure 14-11: Key Receiver Packet Processing Functional Blocks



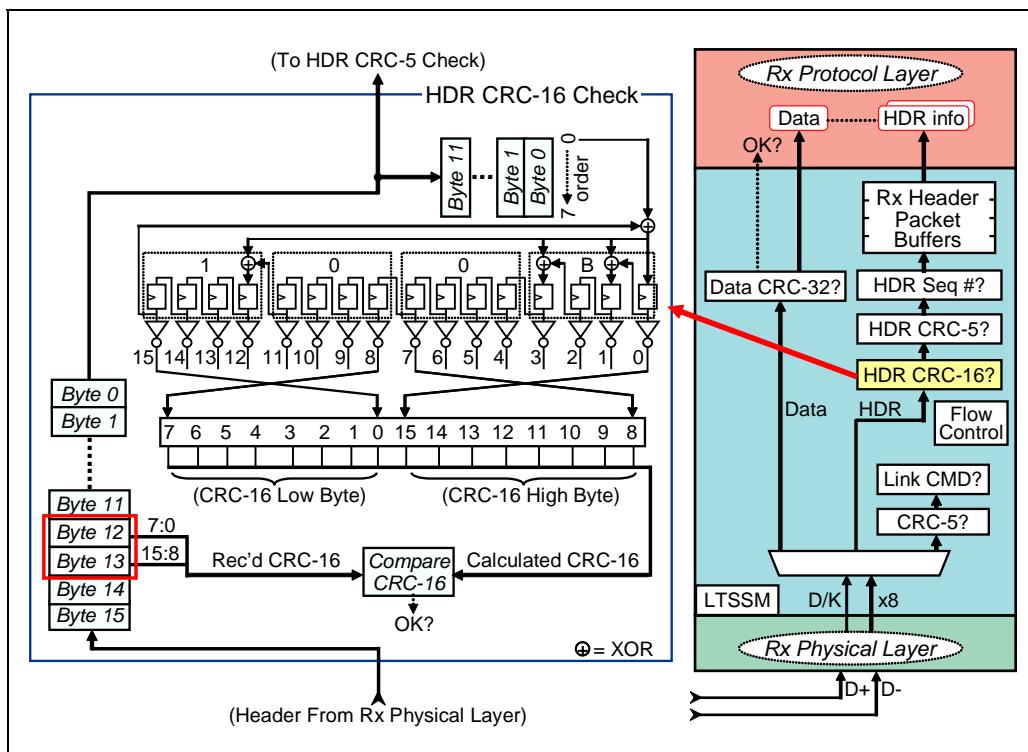
# Chapter 14: Header Packet Processing

## Header Packet CRC-16 Checked

The first step in header packet processing at the link layer of the receiver is to verify that no bit errors occurred on the link while the packet was in flight. As indicated in Figure 14-12 on page 315, one of these checks is CRC-16. The check is done in the same manner as the CRC-16 generation was done by the transmitter. Each byte of the header DW0-DW2 is streamed through the logic to produce a local CRC-16 result. This value is then compared with the CRC-16 field in the header sent by the transmitter.

If the CRC-16 values are the same, the test passes and the CRC-5 test will also be done. If the CRC-16 values are not the same, the receiver will inform the transmitter by way of an LBAD link command that a retry is required of the same header packet. For details, refer to Chapter 16, entitled "Link Errors & Packet Acknowledgement," on page 339.

Figure 14-12: Receiver Checks Header Packet CRC-16



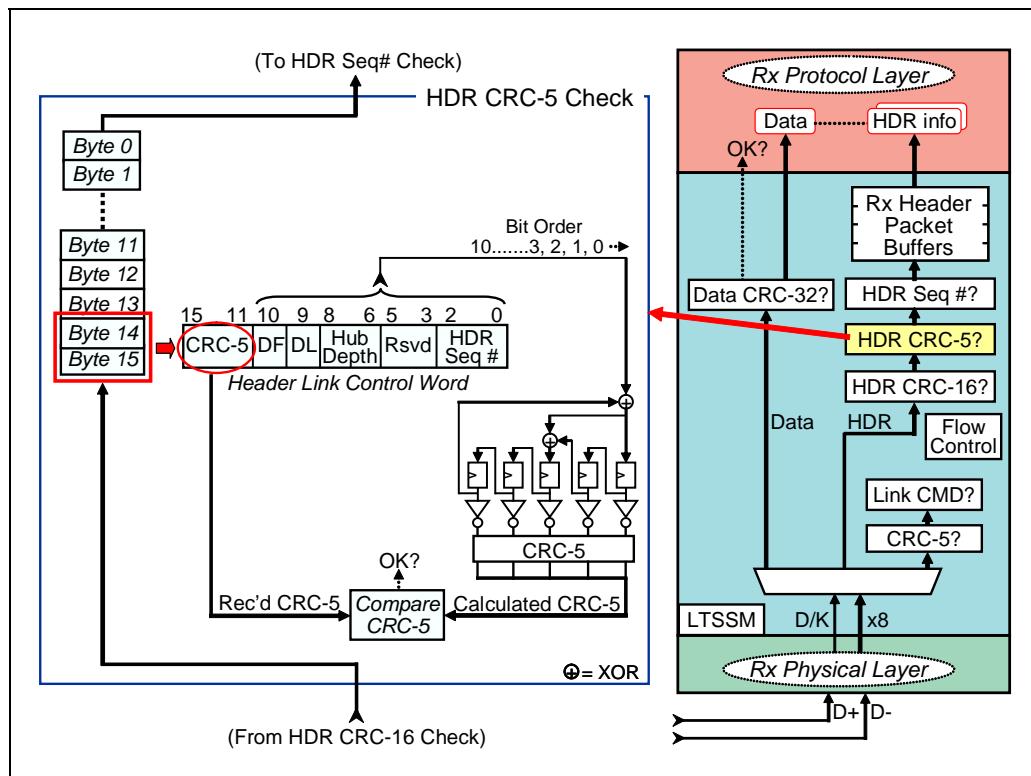
# USB 3.0 Technology

## Header Link Control Word CRC-5 Checked

The other required CRC check is of the Link Control Word in header DW3. This is a CRC-5, and it is done in hardware in the same way as it was carried out by the transmitter. As illustrated in Figure 14-13 on page 316 the lower 11 bits (bits 0-10) are streamed serially through the CRC-5 logic to obtain a local result which is then compared with the CRC-5 field in the header.

If the CRC-5 values are the same, the test passes. If the CRC-5 values are not the same, the receiver will inform the transmitter by way of an LBAD link command that a retry is required of the same header packet. For details, refer to Chapter 16, entitled "Link Errors & Packet Acknowledgement," on page 339.

Figure 14-13: Receiver Checks Header Link Control Word CRC-5

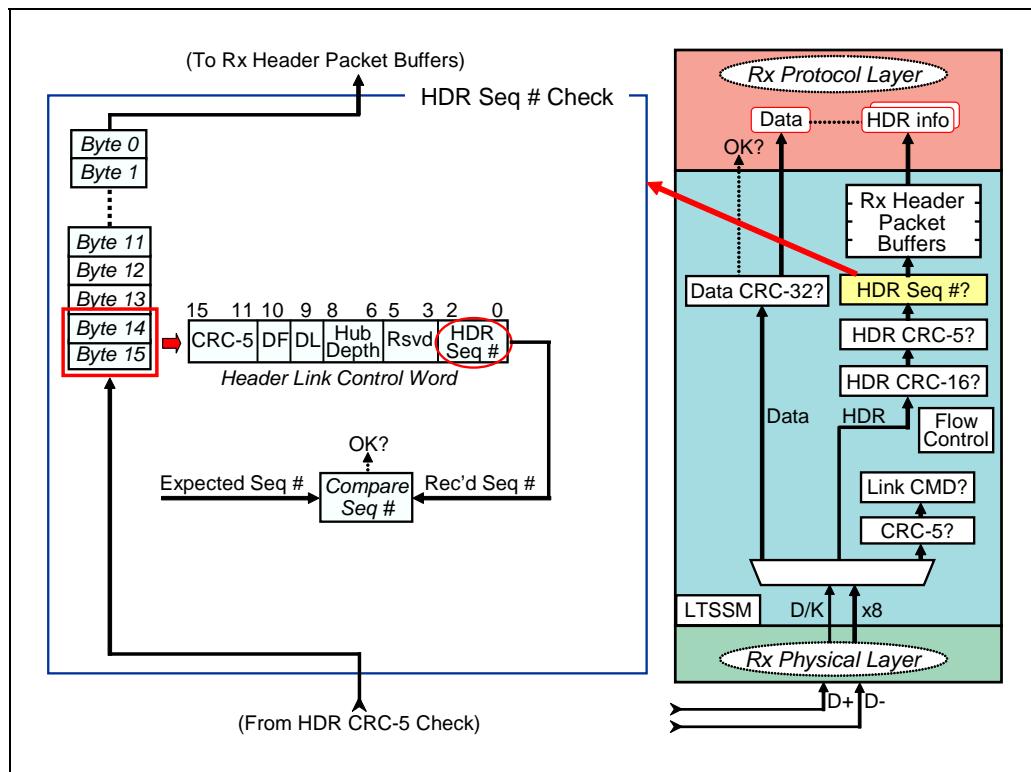


# Chapter 14: Header Packet Processing

## Header Sequence Number Checked

The final header packet check done by the link layer receiver is of the header packet sequence number (Hdr Seq#) assigned by the transmitter link layer to each outbound header packet. As shown in Figure 14-14 on page 317, the receiver link layer maintains a counter tracking the expected Hdr Seq# for each inbound header packet. The expected value starts at 0 and increments by one for each valid header packet that is received. The count is sequential and if the expected Seq# value differs from the actual Hdr Seq# in the header Link Control Word, it is likely that a header packet was corrupted in flight. In any case, the receiver will initiate a transition from U0 to Recovery to attempt to correct the problem. For details, refer to Chapter 16, entitled "Link Errors & Packet Acknowledgement," on page 339.

Figure 14-14: Receiver Checks Header Sequence Number



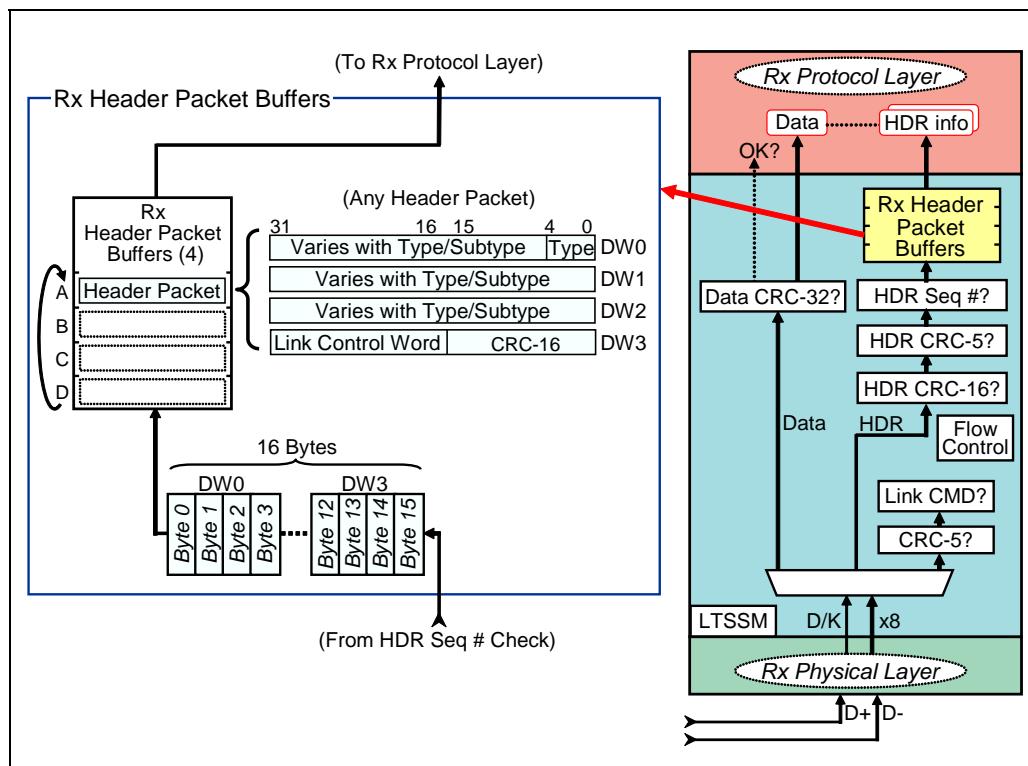
# USB 3.0 Technology

## Header Packet Buffer Accepts The Header

In the current USB 3.0 specification, the receiver is required implement four link layer header packet buffers to help support the link header packet flow control scheme. On Figure 14-15 on page 318, it can be seen that the Rx Header Packet Buffer is in the path of the checks; an entry in the buffer will not be consumed unless a header packet has passed all CRC-5, CRC-16, and Hdr Seq# checks.

A header packet accepted into the Rx Header Packet Buffers remains there until it is forwarded up to the protocol layer of the device. Once that occurs, the receiver will inform the transmitter with a flow control link command that another buffer entry has become available. The burden is on each transmitter to avoid sending header packets unless it has at least one credit (indicating that a buffer is available). For more details, refer to Chapter 15, entitled "Header Packet Flow Control," on page 321.

Figure 14-15: Receiver Header Packet Buffet Accepts Header



# Chapter 14: Header Packet Processing

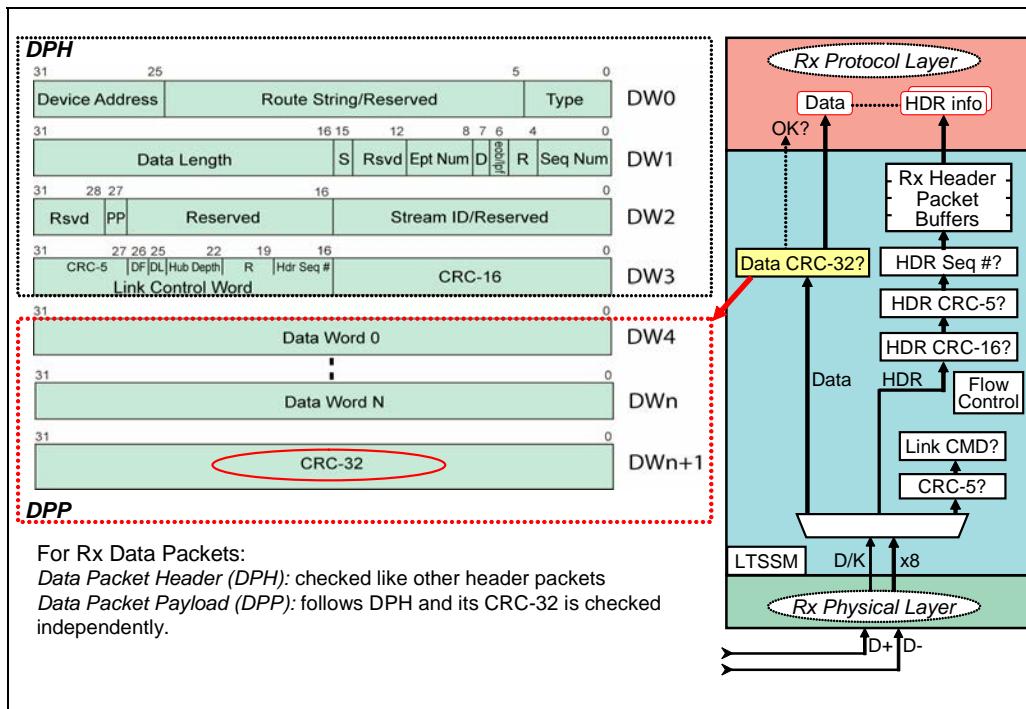
## Data Packet Payload (DPP) CRC-32 Checked

### General

As indicated in Figure 14-16 on page 319, an incoming data packet is arriving at the link layer. Assuming the data packet header (DPH) passes the CRC-16, CRC-5, and Hdr Seq# checks, the DPH will be briefly stored in the Rx Header Packet Buffer. On the link, the data packet payload (DPP) immediately follows the header; if the receiver physical layer detects the DPPSTART framing (for the DPP), the receiver prepares to perform the required CRC-32 check.

As indicated in Figure 14-16, the position of the CRC-32 in the DPP is immediately after the last Data Word (DW). While the USB 3.0 specification indicates that it is a protocol layer responsibility to acknowledge data payloads, the hardware logic to generate and check CRC-32 is shown here (and in the specification) at the link layer.

Figure 14-16: Receiver Checks Data Packet Payload (DPP) CRC-32

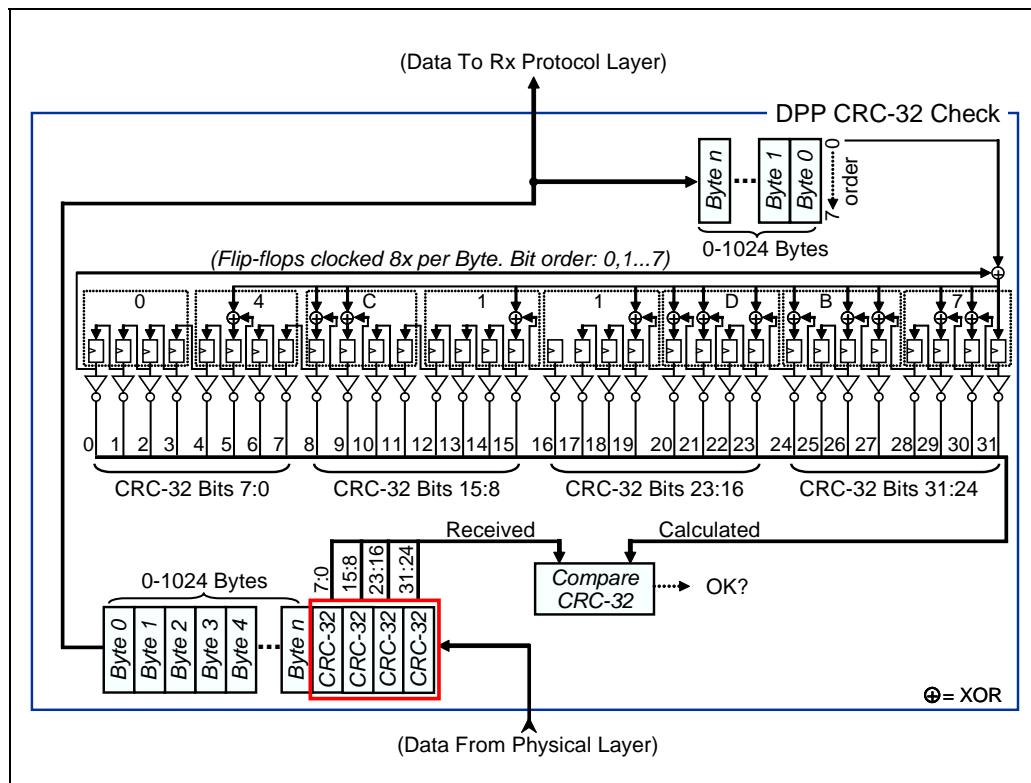


# USB 3.0 Technology

## Rx DPP CRC-32 Checking, The Hardware Details

Figure 14-17 on page 320 illustrates the hardware defined in the specification for generating and checking the DPP CRC-32. As shown, each DW of the payload is shifted into the logic, starting with bit 0 of the low byte in DW0, and ending with bit 7 of the last byte, last DW. In the case of the receiver, the last four bytes of the received DPP (the CRC-32 sent by the transmitter) are then compared with the calculated value.

Figure 14-17: Receiver's DPP CRC-32 Checking Logic



---

---

# 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 3.0 Technology**

---

## **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 it 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

## Chapter 15: Header Packet Flow Control

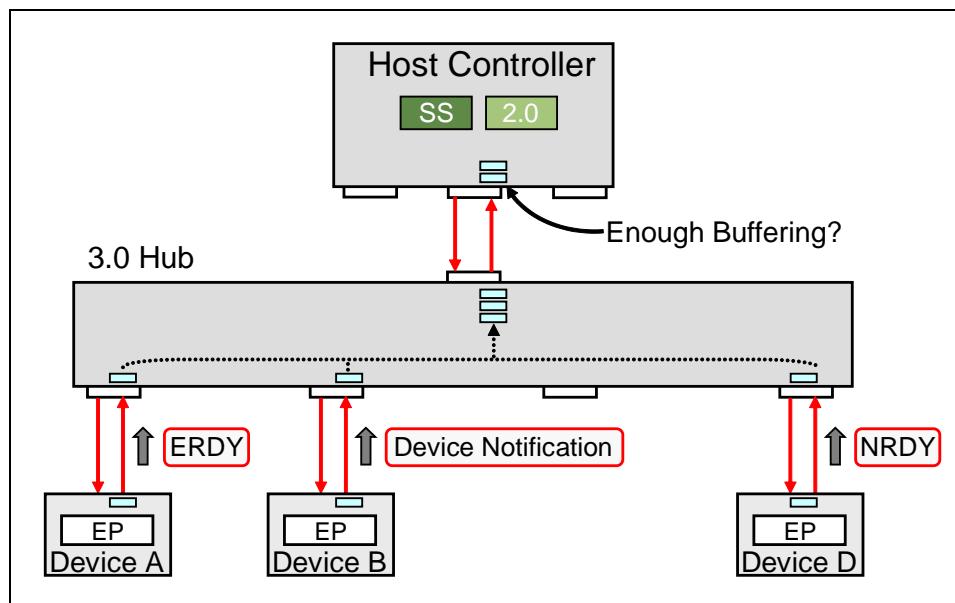
its readiness to continue sending data. As long as the host and endpoint are both ready and willing to proceed, data transfers can continue.

### 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



# **USB 3.0 Technology**

---

## **SuperSpeed Link Level Flow Control Basics**

USB 3.0 defines a credit-based, link level flow control scheme as part of the SuperSpeed Port-to-Port protocol. Link layer logic at the transmitter and receiver in each pair of link partners is responsible for assuring that no header packet is sent which cannot be accepted at the receiver because of a lack of link layer buffering.

It is important to note that link level flow control is completely independent of the End-To-End flow control managed at the protocol layers of the initiator and ultimate recipient. For example, an ACK transaction packet sent by the host controller targeting an IN endpoint is an End-To-End protocol layer packet. It will be routed down the path to the target. As it encounters the downstream facing ports of hubs, it crosses the next link subject to available flow control credits. When it reaches the target, the ACK may or may not receive a NRDY response from the target--independent of link flow control events during its delivery.

Note that some traffic is subject to link level flow control, and some is not:

- Header packets are always flow controlled, including all Transaction Packets (TPs), Link Management Packets (LMPs), Isochronous Timestamp packets (ITPs), and Data Packet Headers (DPH).
- Data Packet Payloads (DPPs) are not flow controlled at the link layer. The assumption is that if the sender has a header credit for the DPH, then the DPP can also be sent. This is appropriate because it is a protocol layer responsibility to acknowledge and flow control data packets using methods defined in the End-To-End protocol.
- Link Commands, Ordered Sets, and logical idle traffic is never flow controlled; when sent, it must be accepted by the link partner.

SuperSpeed link level flow control places most of the burden of avoiding link partner receiver buffer over-runs on the transmitter. The credit scheme requires each receiver link layer to "advertise" credits indicating initial and on-going buffer availability to the transmitter; it also requires the transmitter to track credits consumed as headers are sent. If no credits remain, the transmitter may not send headers until the receiver awards more. Time-outs assure that failures in the flow credit mechanism are recognized; most are recoverable at the link layer through the LTSSM Recovery state.

The next section describes, conceptually, the transmitter and receiver buffers, counters, timers, and link commands required to support link flow control.

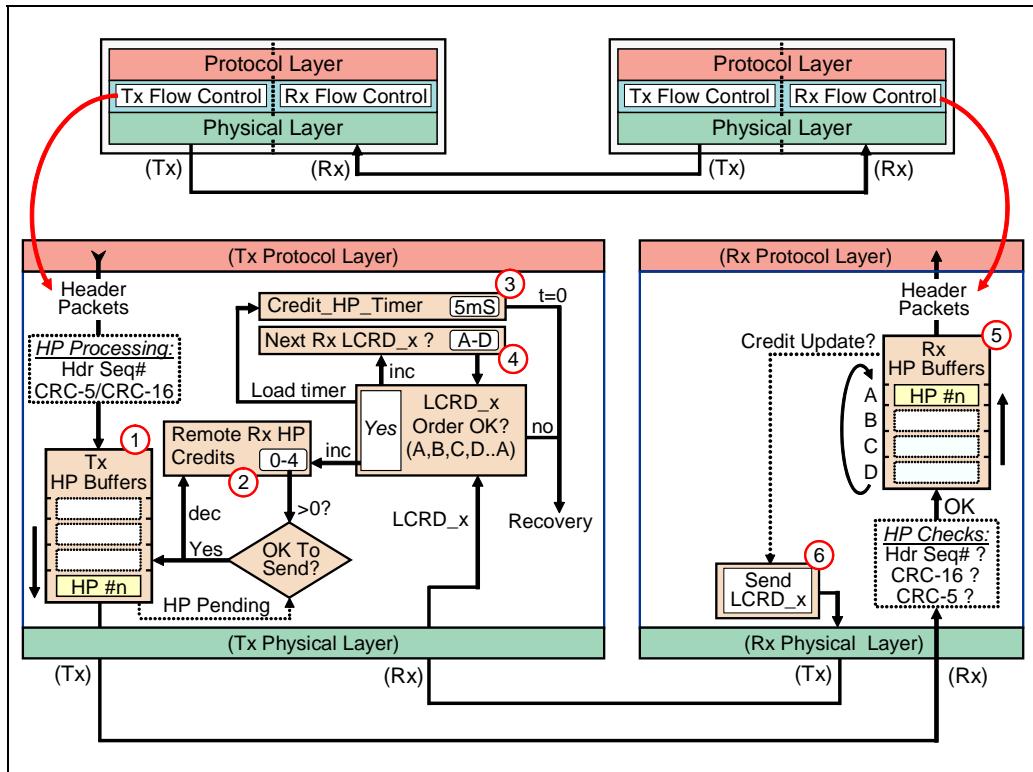
# Chapter 15: Header Packet Flow Control

## Flow Control Elements

SuperSpeed link level flow control defines transmitter and receiver hardware logic as well as four LCRD\_x flow control link command variants. The transmitter and receiver functional requirements are described in this section.

Figure 15-2 is a conceptual view of the link layer flow control elements. Note that only one direction is depicted. The transmit interface of one device is on the left; the receive interface for its link partner is on the right. The logic must be replicated to support header packet flow control in the other direction.

Figure 15-2: Elements Of USB 3.0 Link Level Flow Control



# **USB 3.0 Technology**

---

## **Transmitter Elements**

At the left in Figure 15-2 on page 325, the transmitter buffer, counter, timer, and decision logic related to link layer flow control is shown. Each is summarized below. References to illustration notes are included.

### **Tx Header Packet (HP) Buffers**

Note (1) in Figure 15-2 on page 325. In the current specification, there are four Tx Header Packet (HP) Buffers used by the transmitter to store a local copy of each 16-byte header packet until it is later acknowledged as valid by the link partner receiver via receipt of an LGOOD\_n link command. More on header packet acknowledgement in the next chapter.

### **Remote Rx HP Credit Count**

Note (2) in Figure 15-2 on page 325. This counter is maintained by the transmitter and tracks the number of available link layer header packet receive buffers in its link partner. The count starts at zero and increments by 1 each time a valid LCRD\_x link command is received. It decrements by 1 each time a header packet is sent. In the current USB 3.0 specification, the receiver must implement four receive buffers, so the valid count ranges from 0-4.

A credit check is made by the transmitter before sending a header packet over the link. If a header packet is pending, the check verifies that the current Remote Rx HP Credit Count is >0.

### **Credit HP Timer**

Note (3) in Figure 15-2 on page 325. This timer is implemented to assure that flow control credits are returned promptly by the receiver as long as there are header packets outstanding. Unnecessary delays in returning credits to the transmitter can have the effect of creating back pressure or delaying entry into link power management states U1, U2, or U3 (all credits must be returned before entry into a power management state).

The timeout period for the Credit\_HP\_Timer is 5mS. Basic operation of the timer can be summarized as follows:

- Timer is active when the link is in the U0 state and the Remote Rx Header Buffer Credit count is <4 (at least one outstanding header packet).
- Timer starts to decrement when a new or retried outbound header packet leaves the link layer (is forwarded to the physical layer for transmission).

## **Chapter 15: Header Packet Flow Control**

---

- Timer is reset (reloaded to 5mS) each time a valid LCRD\_x link command is received. It will start to decrement immediately if the Remote Rx Header Count is <4 (at least one more header is still outstanding)
- Port will automatically transition from U0 to Recovery if the Credit\_HP\_Timer times out and header packet credits are still outstanding.

### **Next Rx LCRD\_x**

Note (4) in Figure 15-2 on page 325. This logic tracks the next expected “x” value carried by an LCRD\_x link command. Because of the strict order of use for Rx HP buffers, the value in this register is A, B, C, D, A, etc. Each time an LCRD\_x link command is received, it is compared against the current value in this counter; if not the same, a transition to Recovery is initiated. Each return to U0 from Recovery forces another advertisement of credits by receivers; this assures that the link partners will be in agreement when header packets are again sent.

---

### **Receiver Elements**

At the right in Figure 15-2 on page 325, the receiver logic related to link layer flow control is shown. The key functional blocks are summarized below.

#### **Rx Header Packet (HP) Buffers**

Note (5) in Figure 15-2 on page 325. There are four Rx Header Packet Buffers used by the receiver to temporarily store header packets until they are forwarded up to the protocol layer. The four buffers, referred to as Buffer A, B, C, and D are used in that order. Each time a buffer becomes available, the corresponding LCRD\_x flow control credit link command is sent to the transmitter.

#### **LCRD\_x Generation**

Note (6) in the illustration. Each time an Rx HP Buffer becomes free, the receiver credit update logic requests the transmit interface to send the corresponding LCRD\_x flow control link command to the transmitter. On entry to U0 link state, the receiver sends one of LCRD\_x for each available buffer:

- On U0 entry from reset, four buffers are available (and 4 credits are sent).
- On U0 entry from Recovery, from 0-4 buffers may be free. Receiver sends one LCRD\_x for each available buffer.

# USB 3.0 Technology

## Header Packet Flow Control Link Commands

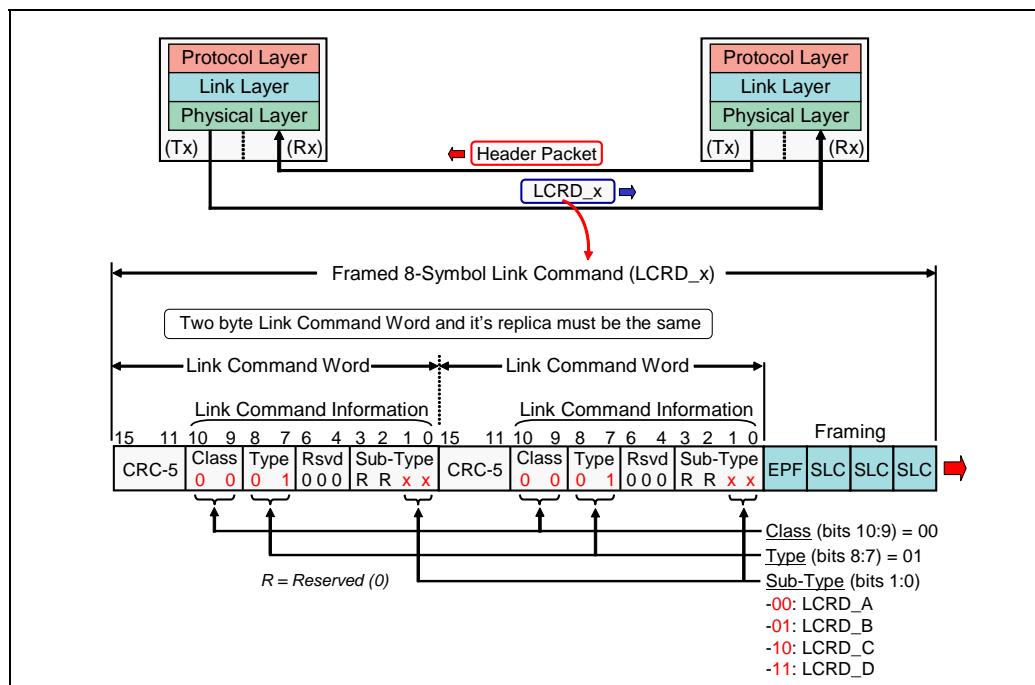
### General

As each header packet is processed by the receiver, the LCRD\_x flow control link command is used to inform the transmitter that an additional credit has become available because a packet has been removed from the Rx HP Buffer and forwarded up to the protocol layer. In addition, an initial buffer availability credit advertisement using LCRD\_x link commands is made each time the SuperSpeed link enters the U0 state from link training or from Recovery.

### LCRD\_x Link Command Format

The LCRD\_x link command has four variants (Sub-Types) which correspond to the available buffer being reported: A, B, C, D. The link command format is illustrated in Figure 15-3 on page 328. As with all link commands, the Link Command Word is 16 bits (2 bytes) and is sent twice.

Figure 15-3: Format Of LCRD\_X Link Command Variants



# Chapter 15: Header Packet Flow Control

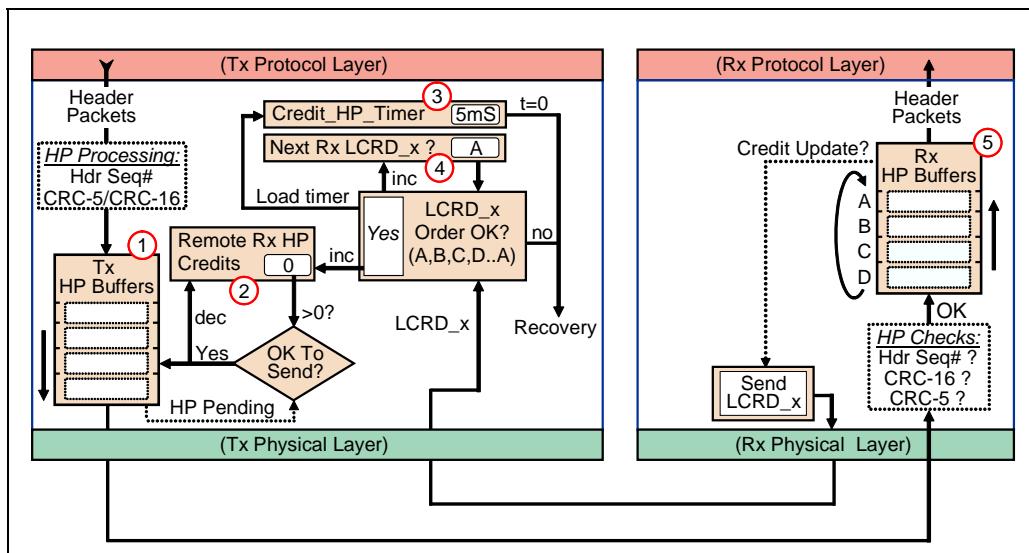
## Flow Control Initialization

This section describes the default state of link layer flow control logic followed by the link partner initialization sequence required to exchange credits.

## Flow Control Logic State After Reset

Figure 15-4 depicts the initial state of transmitter and receiver link layer flow control logic following a reset, but before flow control initialization.

Figure 15-4: Link Level Flow Control Logic After Reset



Numbers in the following list refer to the circled numbers in Figure 15-4.

1. Transmitter Tx HP Buffers are flushed of header packets (if any). Four buffers will be available when normal traffic commences.
2. Transmitter Remote Rx HP Credit Count is reset = 0. This helps avoid the generation of link traffic before link layer flow control is initialized.
3. Transmitter Credit\_HP\_Timer is initialized to 5 ms, but won't start counting down until header packets are sent.

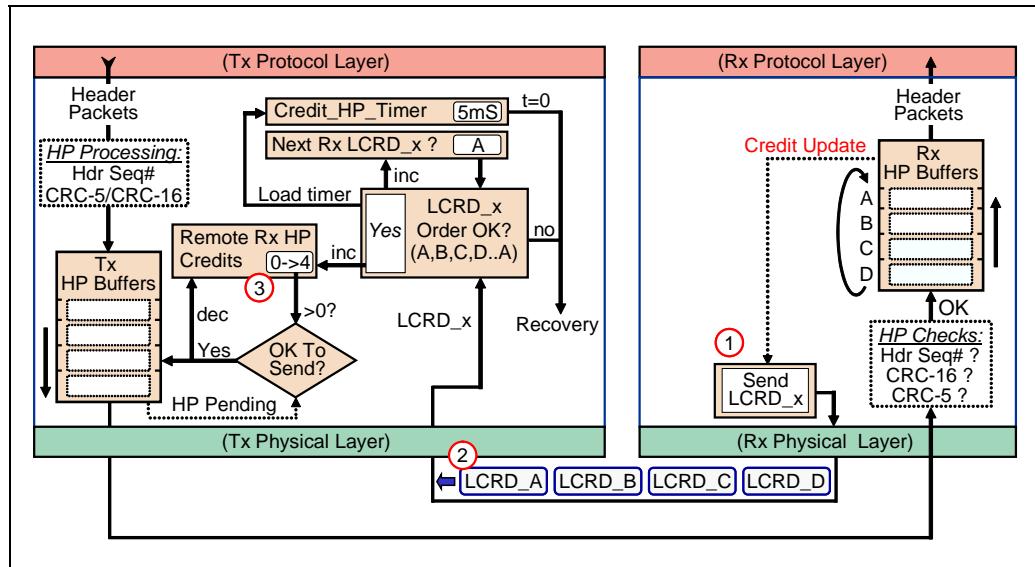
# USB 3.0 Technology

4. Transmitter Next Rx LCRD\_x Count is initialized to 0 (the numerical value corresponding to Rx HP Buffer A). This means that the first expected LCRD\_x to be returned by the receiver is LCRD\_A.
5. Receiver Rx HP Buffers are flushed of header packets (if any). Four available buffers will be advertised as being available before normal traffic commences.

## Flow Control Logic Initialization

Figure 15-5 highlights transmitter and receiver flow control logic to be initialized near the end of the link training, following a reset.

Figure 15-5: Flow Control Logic Initialization



Numbers in the following list refer to the circled numbers in Figure 15-5.

1. The receiver credit update logic is required to perform an initial advertisement of flow control credits at the end of link training. The current USB 3.0 specification requires that receivers implement (and advertise credits for) four buffers.
2. The receiver sends the required four flow control link commands in order: LCRD\_A, LCRD\_B, LCRD\_C, and LCRD\_D.

# Chapter 15: Header Packet Flow Control

3. The transmitter's Remote Rx HP Credit Count increments by one each time a valid LCRD\_x link command is received. The initial credit advertisement by the receiver (four of these link commands) causes this count to rise to the maximum value of four.

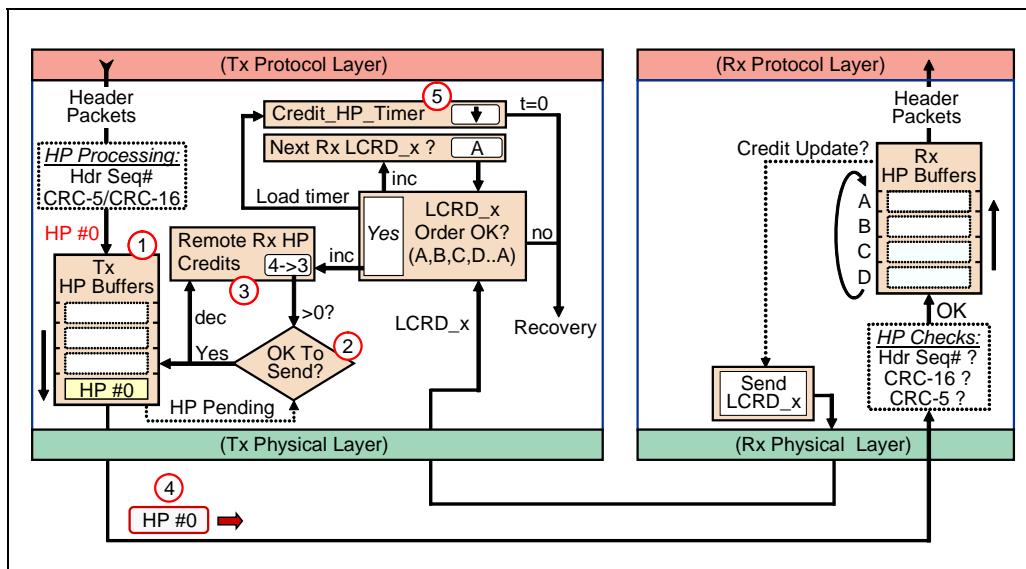
## Flow Control During Normal Operations

This section describes the general sequence of flow control events during normal operations on a SuperSpeed link. The sequence starts when an outbound header packet arriving from the protocol layer is placed in the transmitter's link layer Tx HP Buffer. The sequence ends after the header packet is received, checked, and forwarded up to the protocol layer, and an LCRD\_x flow control link command is returned by the receiver.

### The First Header Is Sent

Refer to Figure 15-6 during the following discussion of events that occur when an outbound header is delivered to the transmitter link layer, processed, and sent to the link.

Figure 15-6: The First Header Packet Is Processed And Sent



# USB 3.0 Technology

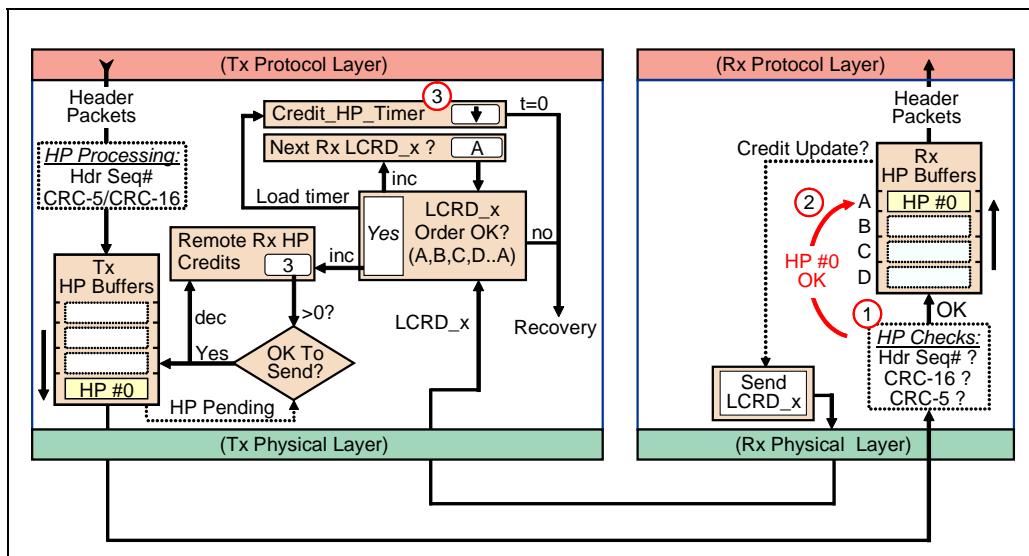
Numbers in the following list refer to the circled numbers in Figure 15-6 on page 331.

1. The 16 byte (4 DW) header sent to the transmitter link layer from the protocol layer is processed before being placed in the Tx HP Buffer. Processing includes assignment of a Hdr Seq# and generation of the CRC-5 and CRC-16. For the purposes of this flow control discussion, assume that the header has been assigned Hdr Seq# 0, CRC-5 and CRC-16 have been generated, and the header (HP #0) is placed in the Tx HP buffer.
  2. Next, the transmitter checks flow control credits. In this example, the Remote HP Credits Count was = 4, so the header packet will be sent.
  3. Transmitter deducts 1 credit from the Remote HP Credit Count (now = 3).
  4. The transmitter forwards the header packet to the physical layer.
  5. The transmitter activates the Credit\_HP\_Timer which will timeout in 5mS unless an LCRD\_x link command returning a credit is received first.
- The transmitter may send more header packets (subject to available credits).

## Header Packet Reaches Rx HP Buffer

Refer to Figure 15-7 during the following discussion of events that occur when an outbound header arrives at the link layer of the receiver.

Figure 15-7: The First Header Packet Arrives At The Receiver



## Chapter 15: Header Packet Flow Control

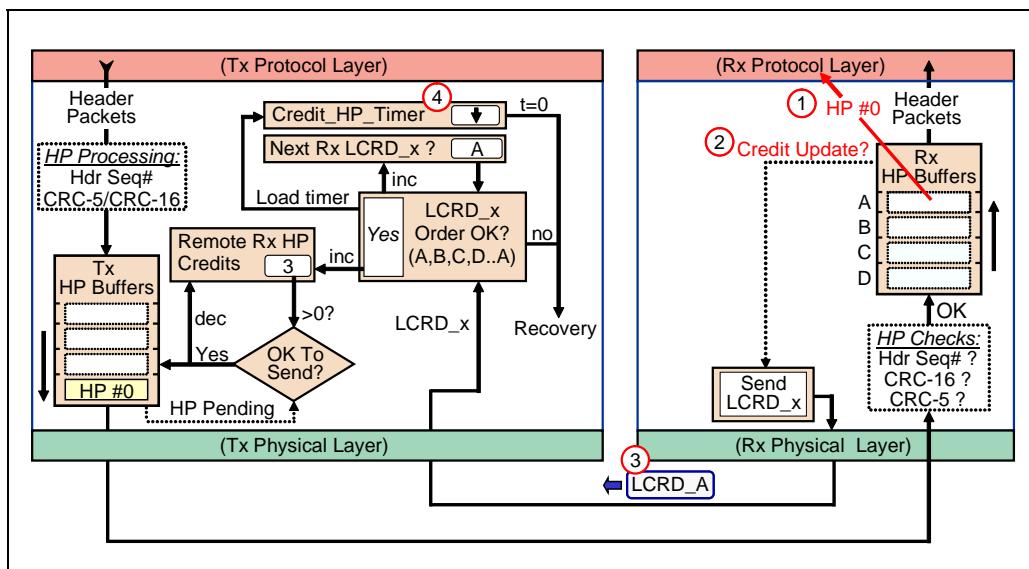
Numbers in the following list refer to the circled numbers in Figure 15-7 on page 332.

1. Each header packet (HP) arriving at the receiver link layer is subjected to header CRC-5, CRC-16, and Hdr Seq# checks to determine if it was delivered over the SuperSpeed link without error. The details of the checks and the link commands used to inform the transmitter of the results are described in Chapter 16, entitled "Link Errors & Packet Acknowledgement," on page 339. For the purposes of this flow control discussion, assume that there were no errors.
2. Because the header passed the checks, it is forwarded to the next sequential Rx HP Buffer. Because this is the first header packet transferred, it will be placed in Buffer A until it is forwarded up to the protocol layer.
3. Note that during this time, the transmitter Credit\_HP\_Timer continues to count down towards 0.

### Emptying An Rx HP Buffer

Refer to Figure 15-8 during the following discussion of events that occur when a header leaves the Rx Header Packet Buffer for the protocol layer.

Figure 15-8: Rx HP Buffer Is Available, Credit Returned



# USB 3.0 Technology

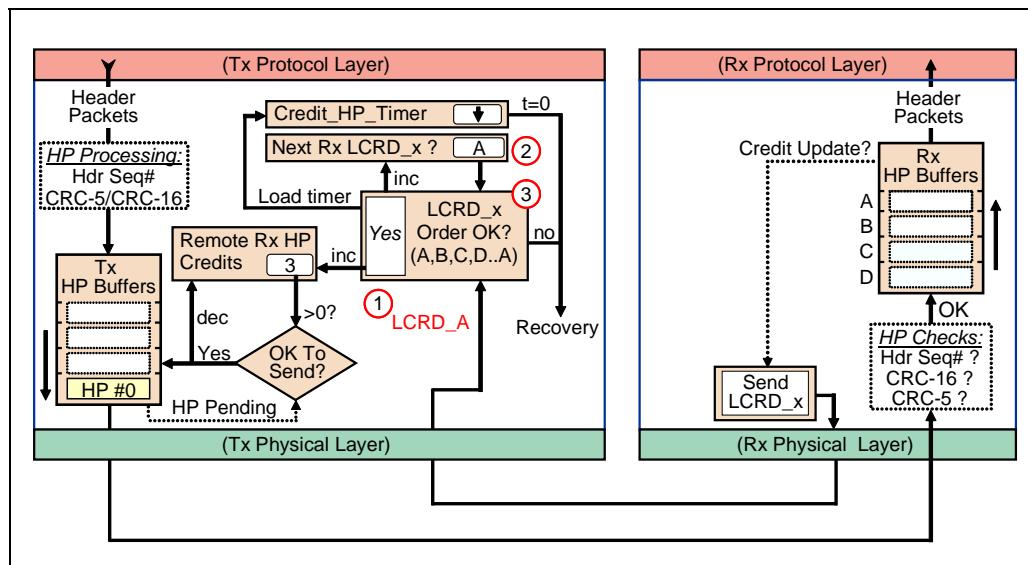
Numbers in the following list refer to the circled numbers in Figure 15-8 on page 333.

1. The Rx HP Buffer is used to store headers temporarily until the receiver removes it and forwards it up to the protocol layer.
2. As headers are removed from the link layer Rx HP Buffer, a credit update is required to inform the transmitter of the event. One credit is reported for each buffer that again becomes available.
3. In this case, an LCRD\_A link command is sent to the transmitter.
4. Note that during this time, the transmitter Credit\_HP\_Timer continues to count down towards 0.

## Transmitter Receives LCRD\_x

Refer to Figure 15-9 during the following discussion of events that occur when the transmitter receives and checks the flow control link command from the receiver.

Figure 15-9: Transmitter Receives A Flow Control Credit



## Chapter 15: Header Packet Flow Control

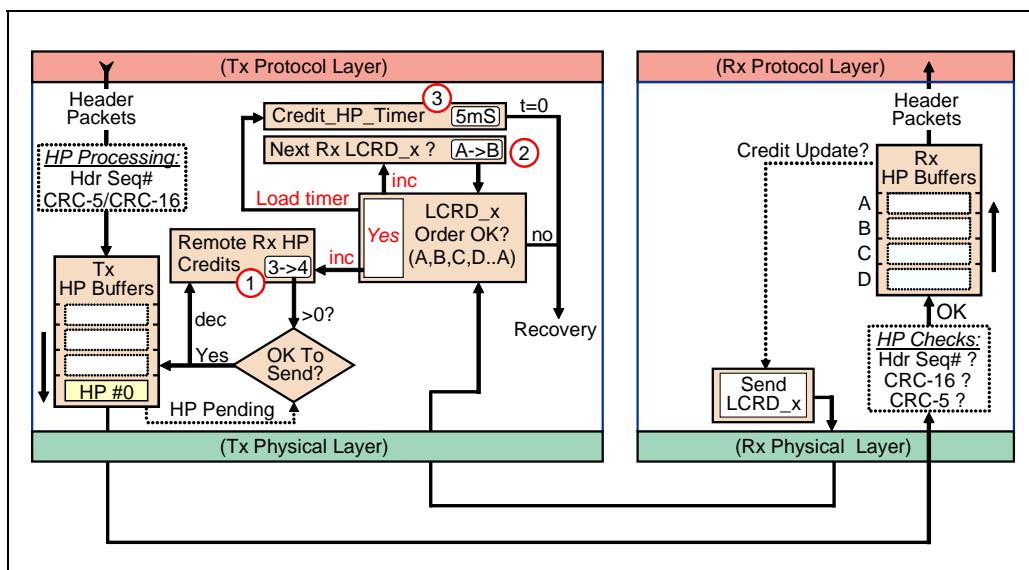
Numbers in the following list refer to the circled numbers in Figure 15-9 on page 334.

1. Each time the transmitter receives an LCRD\_x flow control link command, it is required to check its validity. As with all link commands, the first check confirms that both copies of the Link Command Word are the same, and CRC-5 fields are valid (if not, the entire link command is discarded).
2. Assuming that the link command itself was not corrupted in flight, a second check is made using the current value in the transmitter's Next Rx LCRD\_x counter to determine the expected value of "x" (A-D)
3. The LCRD\_A arriving from the link is compared to the expected value. If the value is correct, the credit update will occur; if not, the transmitter will force a link transition to Recovery to fix the problem. The two cases are described next.

### If Tx Receives A Valid LCRD\_x

Refer to Figure 15-10 during the following discussion of events that occur when the transmitter receives a valid LCRD\_x flow control link command from the receiver.

Figure 15-10: Transmitter Receives A Valid LCRD\_x



## USB 3.0 Technology

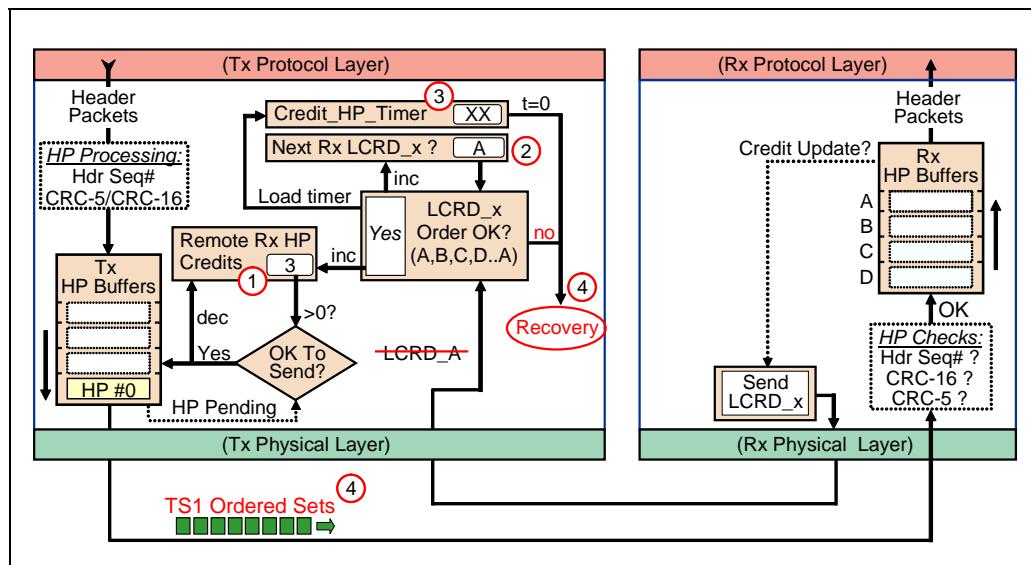
Numbers in the following list refer to the circled numbers in Figure 15-10 on page 335.

1. In this example, the transmitter was expecting (and received) a valid LCRD\_A flow control link command. Because of this, the Remote Rx HP Count increments by 1 (from 3 to 4). The transmitter once again has its full complement of flow control credits.
2. The transmitter also increments the Next RX LCRD\_x Count (from A to B)
3. The Credit\_HP\_Timer is also reloaded with 5mS. Because there are no other header packets outstanding, this timer holds (does not count down).

### If Tx Receives An Invalid LCRD\_x

Refer to Figure 15-11 during the following discussion of events that occur when the transmitter receives an invalid LCRD\_x flow control link command from the receiver or the Credit\_HP\_Timer expires while waiting for a valid LCRD\_x.

Figure 15-11: Transmitter Receives An Invalid LCRD\_x



## **Chapter 15: Header Packet Flow Control**

The numbers in the following list refer to the circled numbers in Figure 15-11 on page 336.

1. In this example, the transmitter expected LCRD\_A, but the one received did not match the expected value in next Rx LCRD\_x Counter. In this case, the Remote Rx HP Credit Count is not updated.
2. The Next Rx LCRD\_x Count is also left unchanged (still = A)
3. The Credit\_HP\_Timer does not continue counting down during the transition to Recovery that follows.
4. The transmitter sends TS1 ordered sets to initiate a transition to Recovery to provide another opportunity for the receiver to advertise the current state of its Rx HP Buffer availability.

On re-entry to U0, the receiver would advertise that all Rx HP Buffers are free, the transmitter would update counters, and the link partners would again be in agreement. More on Recovery in Chapter 21, entitled "Link Recovery and Retraining," on page 467.

## **USB 3.0 Technology**

---

---

# 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

## **USB 3.0 Technology**

---

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**

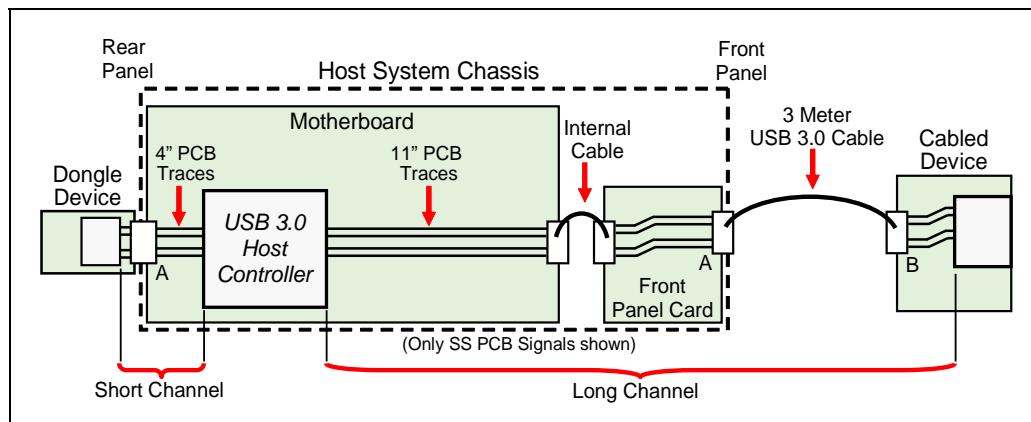
Over the years, as USB added new speeds, users have maintained high expectations regarding plug-and-play, backward compatibility, and low cost. One of the biggest challenges of USB 3.0 SuperSpeed design is preserving this user experience while assuring reliable 5 Gb/s operations in the wide range of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

## Chapter 16: Link Errors & Packet Acknowledgement

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.

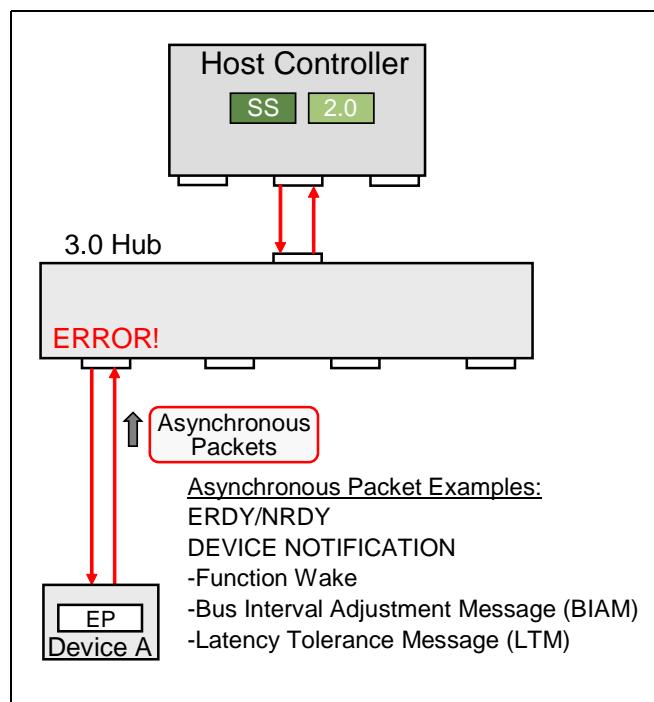
### Upstream Asynchronous Message Errors

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.

## USB 3.0 Technology

As depicted in Figure 16-2, asynchronous traffic is initiated by the endpoint device and moves upstream towards the host. For a device residing below a hub, a link error during transmission of an unsolicited upstream packet means that the message would never reach the host if there was no provision for link level error correction.

Figure 16-2: Upstream Asynchronous Packet Encounters An Error



### **Scope Of SuperSpeed Link Errors**

While the primary focus of this chapter is on the mechanism of header packet acknowledgement, it is worth noting that as symbols cross the 5 Gb/s link, an error can occur anywhere within a header packet, data packet, link command, or one of the ordered set symbols used for framing, clock compensation, training sequences, etc. The approach taken by USB 3.0 for handling SuperSpeed link errors depends on the symbol type and position, whether the link error occurs during link training or in link state U0, etc. Some of the cases are summarized below.

# **Chapter 16: Link Errors & Packet Acknowledgement**

---

## **Training Sequence Errors**

Errors occurring in training sequence ordered sets are expected and tolerated as long as error-free handshake completes within a specified timeout period. If a timeout expires, the device transitions to an LTSSM state that is dependent on the port type. This is described in Chapter 20, entitled "Link Training," on page 427.

## **Errors Occurring While Link is in U0**

There are many possibilities:

- On the link, header packets and link commands are preceded with the four "K" symbol HPSTART or LCSTART framing. Data packet payloads (DPPs) are framed at the front with four "K" symbol DPPSTART and at the back with the four "K" symbol DPPEND framing. Framing errors are automatically corrected at the receiver physical layer as long as only one symbol out of the four framing symbols in a set is corrupted.
- A link command carries two copies of the 16-bit Link Command Word. At the receiver, if both copies of the Link Command Words (including CRC-5) are identical and contain a valid encoding, the link command is carried out. If not, the link command is dropped.
- Bit errors within a header packet will be detected during link layer CRC checks and automatically corrected when the receiver requests the transmitter to retry sending the header. This case is a primary focus for the remainder of this chapter.
- Repeated errors transmitting the same header packet, or errors in header packet ordering, require a link transition from U0 to the Recovery state. This is followed by a retraining of the link and a return to U0. Because this is a longer latency process than retry, the downstream facing port of each hub implements a link error counter (LEC) to track the number of times link errors have forced a transition to Recovery.

---

## **SuperSpeed Link Level Error Correction Approach**

SuperSpeed links replace the traditional USB broadcast model with unicast packets and per-link error correction. At the heart of the link level error handling is header packet acknowledgement and retry which focuses only on header packet error detection and correction. Packet framing, link command, and training sequence ordered set errors are handled as described above.

# **USB 3.0 Technology**

---

## **Goals Of Header Packet Acknowledgement**

The purpose of link level header packet acknowledgement and retry mechanism is two-fold.

### **Reliable Delivery Of Header Packets**

Header packets are exchanged between link partners without error because link layer transmitters protect headers with CRC-5 and CRC-16 and add a sequential header packet sequence number (Hdr Seq#). Receivers are obligated to check the CRCs and link header sequence number to assure that no bit errors occurred and that packets are processed in the order sent.

If a header packet error does occur, the first attempts to recover from the error will be made locally (at the link where the error occurred), without assistance from software or the host controller, and without impacting other links (except for the minor performance impact associated with each retry).

### **Hand Off The More Serious Errors**

Although the header packet acknowledgement and retry mechanism will handle the vast majority of link errors caused by a corrupted 10-bit symbol, there are many cases where header packet retransmission (retry) can't possibly correct the problem. Some examples:

- Marginal signal quality in a long channel causing multiple packet errors or corrupted header packet acknowledgement, flow control credit, or power management handshake link commands.
- Power supply glitches or brownout conditions causing loss of physical, link, or protocol layer state information in one or both of the link partners.

The role of transmitter and receiver header packet acknowledgement logic in such cases is to recognize that excessive retries are occurring, header packets or link commands are being lost and arriving out of order, or an unresponsive link partner is causing timeout conditions, etc.

At this point, a transition to Recovery is invoked and an attempt is made to reestablish the proper link state, return to U0, and resume normal traffic.

## **Chapter 16: Link Errors & Packet Acknowledgement**

---

### **End-To-End Acknowledgement Is Still Needed**

Note that link level header packet acknowledgement is completely independent of the End-To-End data packet acknowledgement managed at the protocol layers of the initiator and transaction target. For example, a data packet sent by the host controller targeting an OUT endpoint is an End-To-End packet. It will be routed down the path to the target and may or may not encounter data packet header errors that cause the header to be retried.

In either case, the response of the target protocol layer will be based on data packet payload CRC-32 and other factors unrelated to link level events in the path between the device and the host controller root hub.

---

### **Header Packet Acknowledgement Elements**

While the USB 3.0 specification does not impose a particular hardware implementation, it defines link layer functional responsibilities and the link commands needed to manage header packet acknowledgement and retry.

This section identifies the role of each major element of header packet acknowledgement and retry, including:

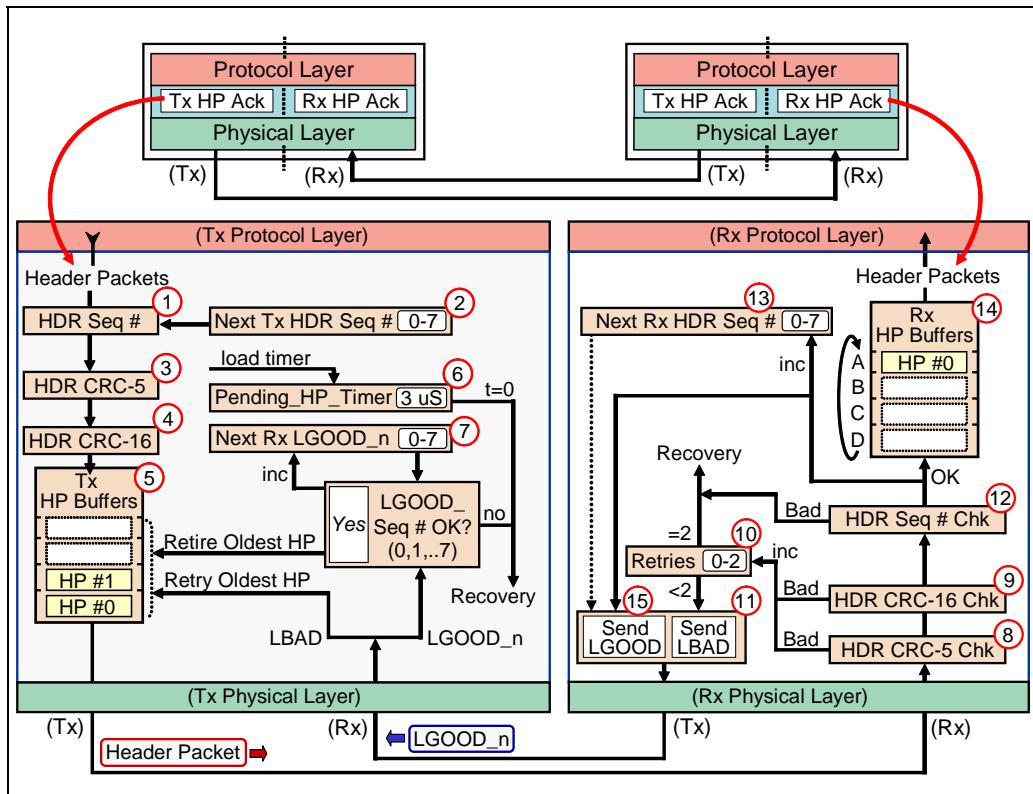
- Transmitter and receiver link layer buffers, counters, timers
- The format of the three header packet acknowledgement and retry link commands: LGOOD\_n, LBAD, and LRTY

Figure 16-3 on page 346 is a conceptual view of the link layer header packet acknowledgement elements. Only one direction is depicted. The transmit interface of one device is on the left; the receive interface for its link partner is on the right. The logic must be replicated to support header packet acknowledgement for the other direction.

## USB 3.0 Technology

---

Figure 16-3: Elements Of Link Level Header Packet Acknowledgement



## Transmitter Elements

At the left in Figure 16-3 on page 346, the transmitter buffer, counter, timer, and decision-making logic related to header packet acknowledgement is shown. Each functional element is highlighted in Figure 16-3 with a reference number and summarized below.

### HDR Seq# Assignment

Note (1) in Figure 16-3 on page 346. As each outbound header arrives at the transmitter link layer, it is assigned the next sequential header sequence number (Hdr Seq#). The 3-bit Hdr Seq# field is in the Link Control Word of the

## **Chapter 16: Link Errors & Packet Acknowledgement**

---

header DW3. Refer to Chapter 14, entitled "Header Packet Processing," on page 303 for details concerning the format and use of header fields.

### **Next Tx HDR Seq#**

Note (2) in Figure 16-3 on page 346. This logic tracks the link header sequence number (Hdr Seq#) to be assigned by the transmitter to the next outbound header packet. The number is initialized to zero, increments by 1 for each header, and rolls over at seven and starts again.

### **HDR CRC-5**

Note (3) in Figure 16-3 on page 346. After assigning the link sequence number for the header packet and adding it to the Hdr Seq# field of the Link Control Word, the transmitter calculates CRC-5 to protect the lower 11 bits of the Link Control Word and places it in the upper 5 bits of the Link Control Word.

### **HDR CRC-16**

Note (4) in Figure 16-3 on page 346. Before copying the completed 16-byte header into the Tx HP Buffer, a CRC-16 protecting the 12-byte core of the header (DW0-DW2) is calculated and placed in the lower 16 bits of DW3.

### **Tx HP (Header Packet) Buffers**

Note (5) in Figure 16-3 on page 346. To support the header packet replay mechanism, the transmitter is required to save a local copy of each completed outbound header packet until it is later acknowledged as valid by the link partner receiver. In the current specification, there are four Tx Header Packet (HP) Buffers.

### **Pending\_HP\_Timer**

Note (6) in Figure 16-3 on page 346. This timer is implemented to assure that the receiver acknowledges header packets promptly. It also protects against the possibility that a header packet acknowledgement LGOOD\_n, LBAD, or LRTY link command was corrupted in flight. Unnecessary delays in packet acknowledgement can have the effect of creating back pressure at the transmitter or delaying entry into link power management states U1, U2, or U3 (all header packets must be acknowledged before entry into a power management state).

The timeout period for the Pending\_HP\_Timer is 3uS. The rules for managing the timer can be summarized as follows.

## **USB 3.0 Technology**

---

The Pending\_HP\_Timer is active only when the link is in the U0 state and one the following is true:

- A Header Sequence Number Advertisement (starting LGOOD\_n value) is expected from its link partner. This occurs on entry to U0 from the link training LTSSM Polling state and on entry to U0 from Recovery.
- There is at least one unacknowledged header packet in the Tx HP Buffers.

The Pending\_HP\_Timer is started (counts down) if one the following is true:

- A Header Sequence Number Advertisement (starting LGOOD\_n value) is expected from its link partner.
- The first header packet is sent (no other outstanding header packets in the Tx HP Buffers).
- A retry sequence begins (following earlier receipt of an LBAD response).

The Pending\_HP\_Timer is reset, but stopped, if any of the following are true:

- A Header Sequence Number Advertisement (starting LGOOD\_n value) was received from its link partner but no outbound header packets have yet been processed.
- An LGOOD\_n header packet acknowledgment link command was received and there are no more outstanding header packets in the Tx HP Buffers.
- An LBAD is received in response to a header packet transmission. The timer will remain stopped while the transmitter prepares to retry the failed header packet (it starts again when the retry occurs).

The retry mechanism is intended to handle failed header packet transmission and allow the link partners to remain in the U0 state. If retry is not successful, a port will transition from U0 to Recovery if two things are true:

- The Pending\_HP\_Timer decrements to 0 (times out) AND
- Any outbound header or data packet that was in progress when the timeout occurred has been allowed to complete.

### **Next Rx LGOOD\_n**

Note (7) in Figure 16-3 on page 346. This counter tracks the Hdr Seq# of the last header to be acknowledged by the receiver. Each time a valid LGOOD\_n link command is received, its value is compared to the current value in this counter. If they are the same, the value in this counter is incremented and the Tx HP Buffer header packet with the “oldest” Hdr Seq# is retired.

# **Chapter 16: Link Errors & Packet Acknowledgement**

---

## **Receiver Elements**

At the right in Figure 16-3 on page 346, the receiver logic related to link layer header packet acknowledgement is shown. Each functional element is highlighted in Figure 16-3 with a reference number and summarized below.

### **HDR CRC-5 And CRC-16 Checks**

Note (8) and (9) in Figure 16-3 on page 346. As each inbound header packet arrives at the receiver link layer, two header CRC checks are performed locally:

- CRC-16 is generated for the first 12 bytes (DW0-2) of the header
- CRC-5 is generated for the lower 11-bits of the Link Control Word (in DW3)

If both CRC-5 and CRC-16 generated locally agree with the CRC-5 and CRC-16 fields in the header delivered by the transmitter, the CRC check was successful. The header packet is forwarded to the Hdr Seq# check.

If either or both of the CRC-5 and CRC-16 values generated locally do not agree with the CRC-5 and CRC-16 in the header, the check failed. The header packet is discarded and one of two actions is taken:

- If the retry limit for this header packet has not been reached, an LBAD is sent to request the transmitter to send it again.
- If the retry limit for this header packet has been reached, a transition to Recovery is initiated.

See the next note for more on retry limits.

### **Retries (Retry attempt counter)**

Note (10) in Figure 16-3 on page 346. The link layer header packet retry mechanism limits the number of attempts to successfully transmit a header packet (same Hdr Seq#) to three tries. Each time an inbound header packet fails a CRC-5 or CRC-16 check, the header packet is discarded and the receiver checks the number of retry attempts already made. Based on the check, there are two possibilities:

- If the Retries count is <2, the count is incremented by 1 and an LBAD link command informing the transmitter that the header packet should be sent again.

## **USB 3.0 Technology**

---

- If Retries count = 2, then the maximum transmission attempts for this header packet has been reached (first attempt plus two retries). At this point, the receiver triggers a transition from U0 to Recovery. On the return to U0, the link partners will again exchange LGOOD\_n link commands to come to an agreement on the current Hdr Seq# value for each link direction.

### **LBAD Generation**

Note (11) in Figure 16-3 on page 346. If an inbound header packet fails a CRC-5 or CRC-16 check and if the retry attempt count is not already = 2, the receiver sends an LBAD link command informing the transmitter that the oldest unacknowledged header packet (based on Hdr Seq#) in its Tx HP Buffer must be sent again (retried).

### **HDR Seq# Check**

Note (12) in Figure 16-3 on page 346. If an inbound header packet passes both CRC-5 and CRC-16 checks, the receiver verifies that the link sequence number (Hdr Seq#) agrees with the next expected value. A comparison is made between the Hdr Seq# of the inbound header packet and the current value in the receiver's Next Rx HDR Seq# counter.

If the Hdr Seq# and Next Rx HDR Seq# values agree:

- The header packet is placed in the Rx HP Buffer
- The Next Rx HDR Seq# is incremented by 1

If the Hdr Seq# and Next Rx HDR Seq# values don't agree:

- The header packet is discarded
- The Next Rx HDR Seq# is not incremented
- An LBAD link command will be sent to the transmitter requesting a retry

### **Next Rx HDR Seq#**

Note (13) in Figure 16-3 on page 346. This logic tracks the expected header sequence number (Hdr Seq#) for the next inbound header packet. The range of this counter is 0-7 and the rules for loading it are:

- On entry to U0 from the link training Polling.Idle state, the value = is 0.
- On entry to U0 from Recovery, the value in this counter is the same as it was the last time the link was in U0.
- Each time a valid header packet is processed, the number in this counter increments by 1 (the count rolls over at 7).

## **Chapter 16: Link Errors & Packet Acknowledgement**

---

### **Rx Header Packet (HP) Buffers**

Note (14) in Figure 16-3 on page 346. There are four Rx Header Packet Buffers used by the receiver to temporarily store header packets until they are forwarded up to the protocol layer. The four buffers, referred to as Buffer A, B, C, and D are used in that order. The next entry in the Rx HP Buffers is used when an inbound header packet has passed the required CRC-5, CRC-16, and Hdr Seq# checks. At that time, the receiver also returns an LGOOD\_n link command to the transmitter acknowledging receipt of the valid header packet.

Some time later, each header packet in the Rx HP Buffers is forwarded up to the receiver protocol layer (in the same order it arrived). As each Rx Buffer again becomes free, the receiver sends the corresponding LCRD\_x flow control credit link command to the transmitter. Details on this may be found in Chapter 15, entitled "Header Packet Flow Control," on page 321.

### **LGOOD\_n Generation**

Note (15) in Figure 16-3 on page 346. Each time an inbound header packet passes the required CRC-5, CRC-16, and Hdr Seq# checks it is placed in the next Rx HP Buffer and the receiver sends an LGOOD\_n link command with the Hdr Seq# corresponding to that of the valid header packet being acknowledged.

---

### **Header Packet Acknowledgement Link Commands**

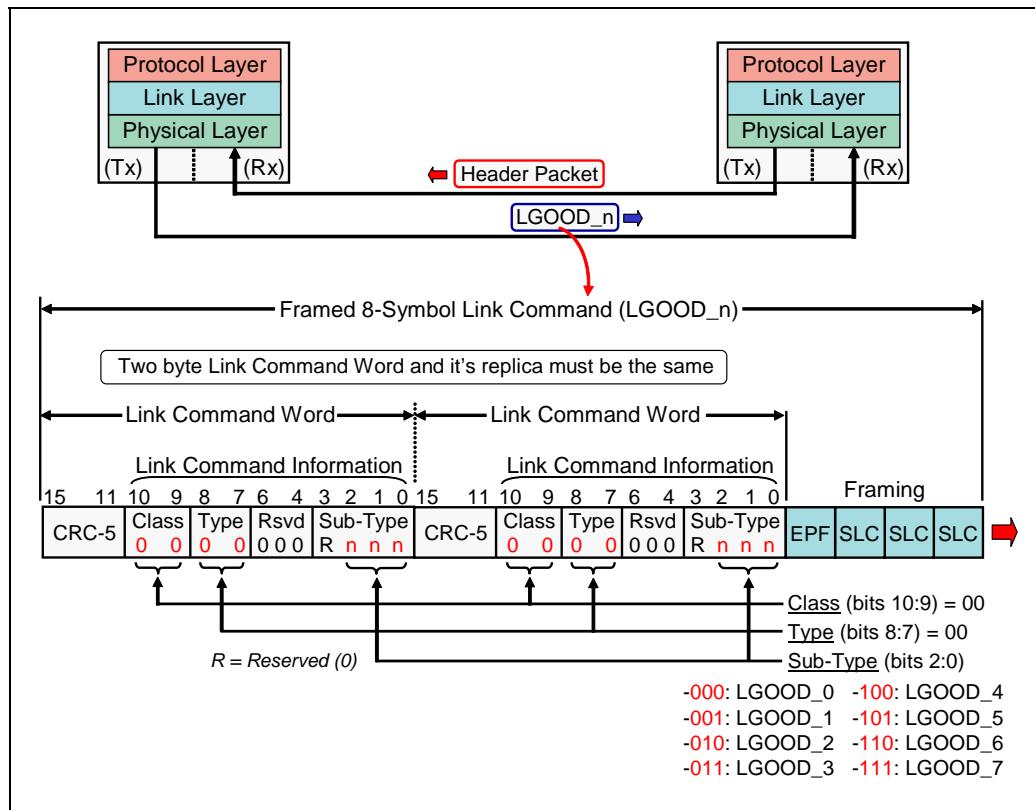
As each header packet is processed by the receiver, one of two header packet acknowledgement link commands is sent to inform the transmitter whether the header was received successfully and the transmitter's local copy may be deleted, or if there was CRC error requiring the transmitter to retry the transmission.

# USB 3.0 Technology

## LGOOD\_n Link Command Format

An LGOOD\_n link command has eight variants (Sub-Types) which correspond to the header sequence number (Hdr Seq#) of the valid header most recently processed by the receiver. The LGOOD\_n link command format is illustrated in Figure 16-4 on page 352. As with all link commands, the Link Command Word is 16 bits (2 bytes) and is sent twice.

Figure 16-4: Format Of LGOOD\_n Link Command Variants

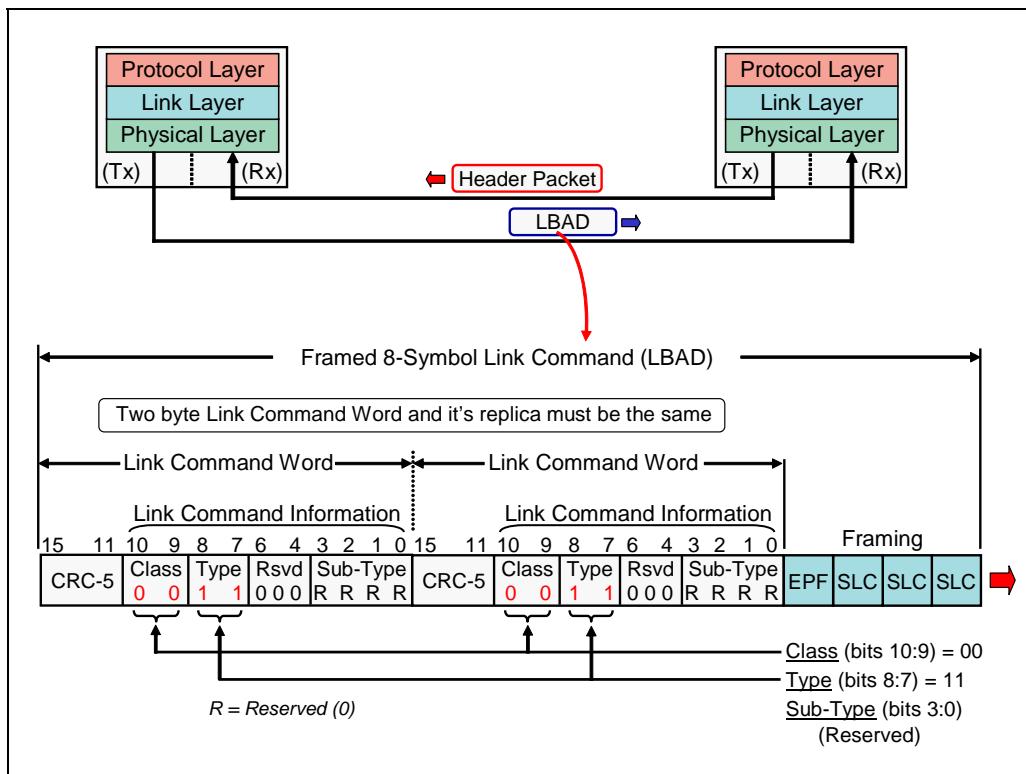


# Chapter 16: Link Errors & Packet Acknowledgement

## LBAD Link Command Format

The LBAD link command, depicted in Figure 16-5 on page 353, has only one variant (Sub-Type). When LBAD is sent, the receiver is indicating that the oldest unacknowledged header packet in the transmitter's Tx HP Buffers was corrupted in flight, was discarded, and must be sent again. To assure proper ordering, packets that arrive after the failed one, but before the replay commences, have higher Hdr Seq#s and will also be discarded.

Figure 16-5: Format Of The LBAD Link Command

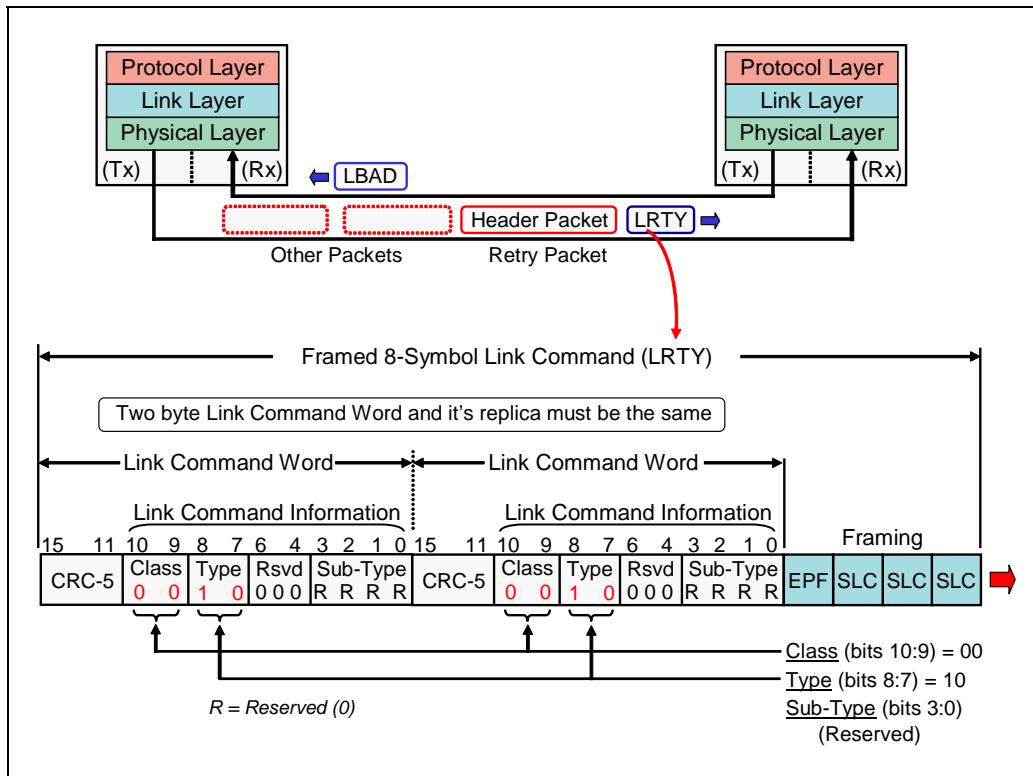


# USB 3.0 Technology

## LRTY Link Command Format

The LRTY link command, depicted in Figure 16-6 on page 354, also has only one variant (Sub-Type) and is sent by the transmitter at the start of a retry sequence. When LRTY is sent, the transmitter is identifying the point in link traffic where the header packet that failed previously, and is being retried, is to be found. The first header packet after LRTY is always the retry packet, but the transmitter may follow it with all other unacknowledged header packets in the Tx HP Buffers.

Figure 16-6: Format Of The LRTY Link Command



# Chapter 16: Link Errors & Packet Acknowledgement

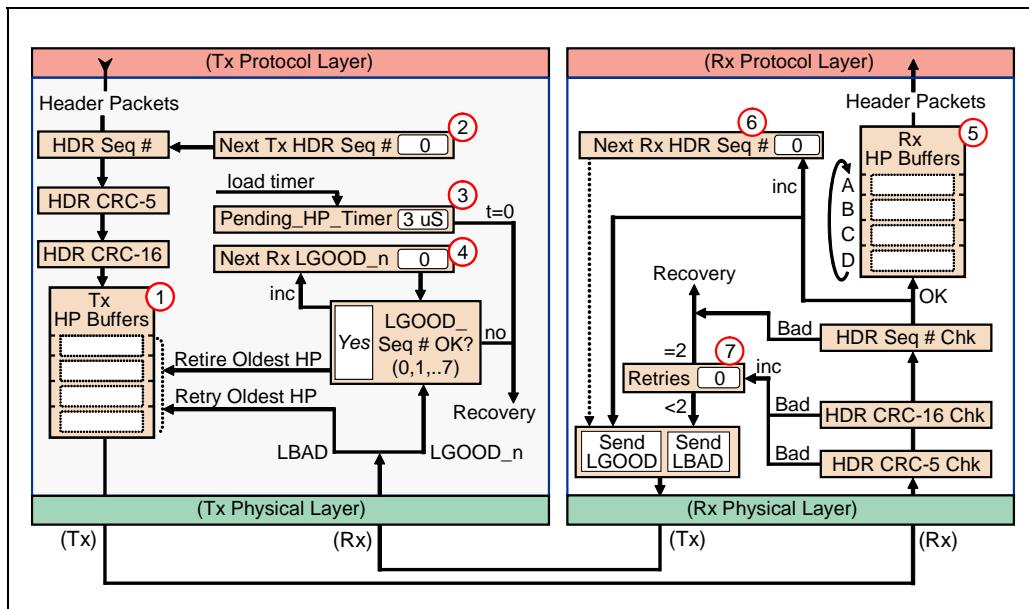
## Header Packet Acknowledgement Initialization

This section describes the default state of link layer header packet acknowledgement logic after reset and then the link partner initialization sequence required to establish the starting header packet link sequence number (Hdr Seq#).

## HP Acknowledgement Logic After Reset

Figure 16-7 on page 355 depicts the initial state of transmitter and receiver link layer header packet acknowledgement logic on entry to U0 following reset and link training, but before the initial Header Packet Sequence Number Advertisement.

Figure 16-7: Link Level Header Packet Acknowledgement Logic After Reset



Numbers in this list refer to the circled numbers in Figure 16-7 on page 355.

1. Transmitter Tx HP Buffers are flushed of header packets (if any). Four buffers will be available when normal traffic commences.
2. Next Tx HDR Seq# is initialized = 0. This will be the starting header packet link sequence number (Hdr Seq#) for outbound headers.

# USB 3.0 Technology

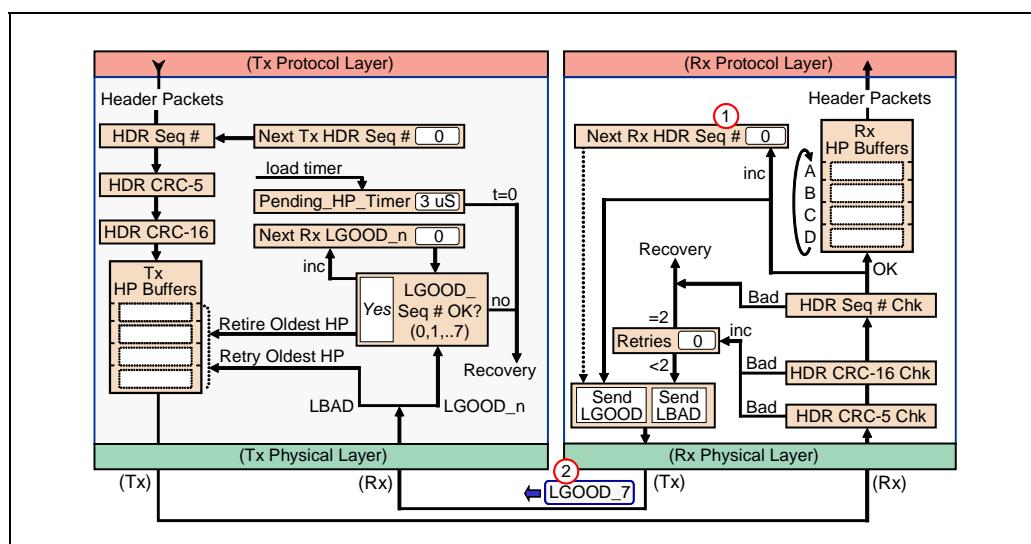
3. Transmitter Pending\_HP\_Timer is initialized to 3uS and starts counting down in anticipation of receiving the initial Header Sequence Number Advertisement LGOOD\_n from its link partner. If the initial LGOOD\_n is not received before the timer expires, a transition to Recovery will occur.
4. Transmitter Next Rx LGOOD\_n counter is initialized = 0. This means that the first expected header packet to be acknowledged by the receiver will be Hdr Seq# 0 (LGOOD\_0).
5. Receiver Rx HP Buffers are flushed of header packets. Four available buffers will be advertised as being available using LCRD\_x flow control link commands before normal traffic commences. Flow control is covered in detail in Chapter 15, entitled "Header Packet Flow Control," on page 321.
6. Receiver Next Rx HDR Seq# is initialized = 0. This indicates that first header packet received is expected to have a link header sequence number (Hdr Seq#) of 0.
7. Retries counter is initialized = 0. Starting with the first, each inbound header packet is allowed up to 3 attempts at successful delivery before the link will transition from U0 to Recovery for retraining.

## HP Sequence Number Advertisement

### General

Figure 16-8 on page 356 highlights the final step in header packet acknowledgement initialization, the initial Header Packet Sequence Number advertisement.

Figure 16-8: Initial Header Packet Sequence Number Advertisement



## **Chapter 16: Link Errors & Packet Acknowledgement**

---

### **Sequence Of Events**

Numbers in this list refer to the circled numbers in Figure 16-8 on page 356.

1. On entry to U0 from link training or Recovery, each receiver is required to send a Header Packet Sequence Number advertisement consisting of an LGOOD\_n link command indicating the starting Hdr Seq# the transmitter should use for header packets it sends. The advertised number is always the Hdr Seq# last header packet acknowledged. One method the receiver can use to determine the correct LGOOD\_n value is to deduct 1 from the value in the Next Rx HDR Seq# counter (Next Rx HDR Seq# -1). Because the Next Rx HDR Seq# is initialized = 0 after reset, the advertised value will be 7 (0-1=7 in the modulo-8 Hdr Seq# counting scheme)
2. Receiver uses its transmit interface to send the LGOOD\_7 link command.

Note that on an entry to U0 from reset and link training, the transmitter loads its Next Rx LGOOD\_n counter = 0 regardless of the initial Header Packet Sequence Number advertisement. On an entry to U0 from Recovery, the transmitter does use the Header Packet Sequence Number advertisement LGOOD\_n number to determine the value to load into this counter.

---

### **HP Acknowledgement Sequence: No Retry Required**

This section describes the general sequence of header packet acknowledgement events during normal operations on a SuperSpeed link. The sequence starts when an outbound header is delivered from the protocol layer to the link layer transmitter for processing. The sequence ends when the receiver link layer has received the header packet, checked CRC-5, CRC-16, and the Hdr Seq#, and returned the corresponding LGOOD\_n header packet acknowledgement link command.

---

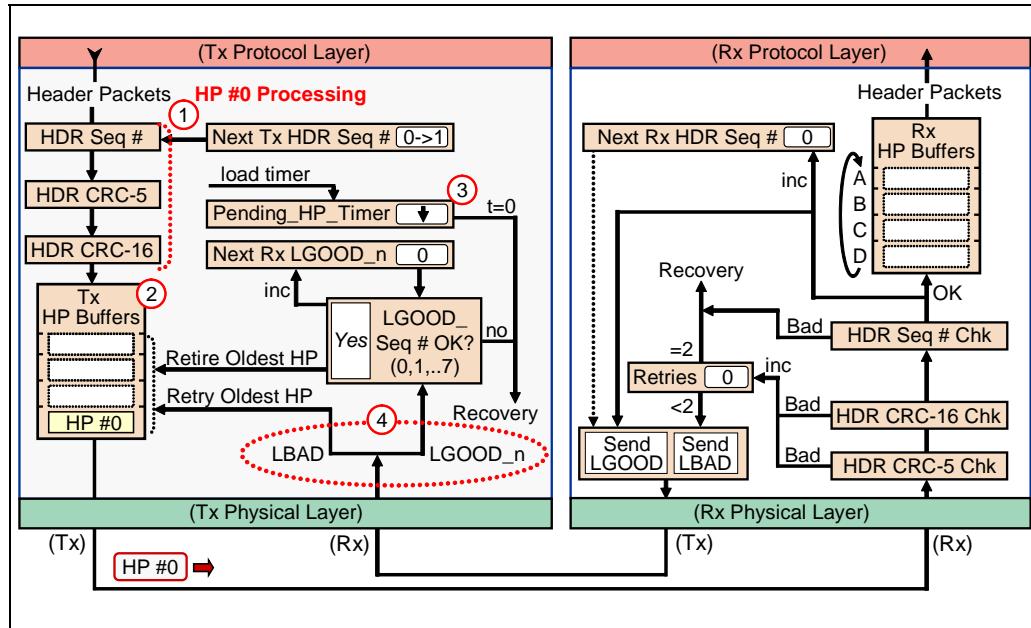
### **First Header Packet Is Sent**

Refer to Figure 16-9 on page 358 during the following discussion of events that occur when an outbound header is delivered to the transmitter link layer, processed, and sent to the link.

## USB 3.0 Technology

---

Figure 16-9: The First Header Packet Is Processed And Sent



Numbers in the following list refer to the circled numbers in Figure 16-9 on page 358.

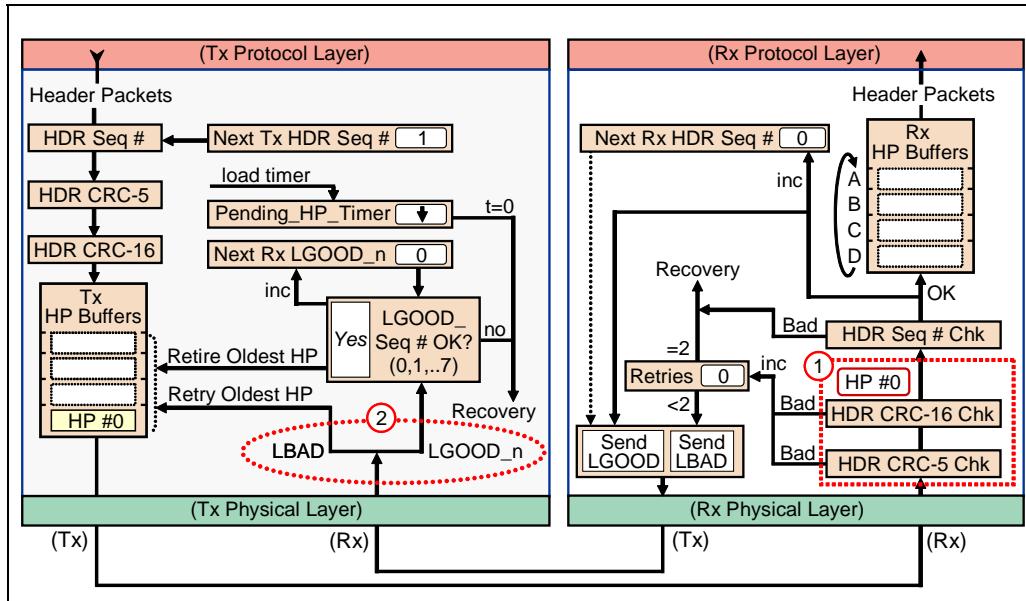
1. Each header arriving from the protocol layer is processed: Hdr Seq# is assigned and CRC-5/CRC-16 are generated. For this example, assume that the header has been assigned Hdr Seq# 0; CRC-5 and CRC-16 have been generated and added to the header (which is now complete).
2. A copy of the completed header (HP #0) is placed in the Tx HP buffer. Next, flow control credits are checked (assume that credits are available).
3. The transmitter sends the header packet and activates the 3uS Pending\_HP\_Timer.
4. At this point the transmitter is waiting for a header packet acknowledgement. There will be a Pending\_HP\_Timer timeout in 3uS unless an LGOOD\_n or LBAD link command is returned by the receiver first.

# Chapter 16: Link Errors & Packet Acknowledgement

## Header CRC-5, CRC-16 Checked By The Receiver

Refer to Figure 16-10 on page 359 during the following discussion of events that occur when the header arrives at the receiver and is checked for CRC errors.

Figure 16-10: Header Packet CRC-5, CRC-16 Checked



Numbers in the following list refer to the circled numbers in Figure 16-10 on page 359.

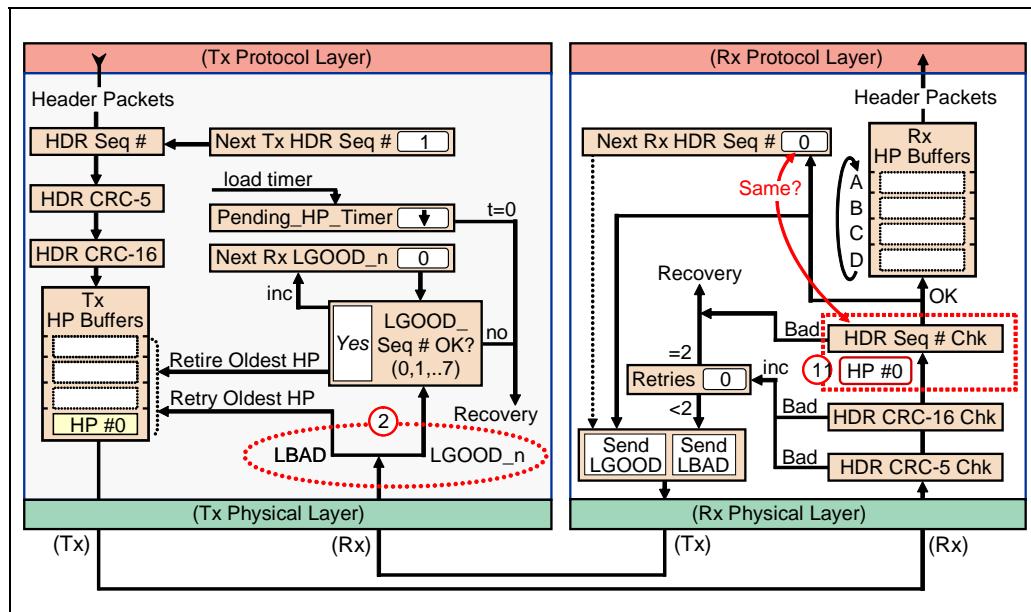
1. Each header packet (HP) arriving at the receiver link layer is subjected to header CRC-5 and CRC-16 checks to determine if it was delivered over the SuperSpeed link without bit errors. In this example, assume that both checks were successful. The receiver prepares to check the header sequence number (Hdr Seq#)
2. The transmitter's Pending\_HP\_Timer continues counting down while it waits for the receiver to return an LGOOD\_n or LBAD response. The transmitter may also send more header packets if it has flow control credits.

# USB 3.0 Technology

## Header Sequence Number Checked By Receiver

Refer to Figure 16-11 on page 360 during the following discussion of events that occur when the packet passes CRC-5 and CRC-16 checks and the link header packet sequence number (Hdr Seq#) is validated.

Figure 16-11: Header Packet Sequence Number Is Checked



Numbers in the following list refer to the circled numbers in Figure 16-11 on page 360.

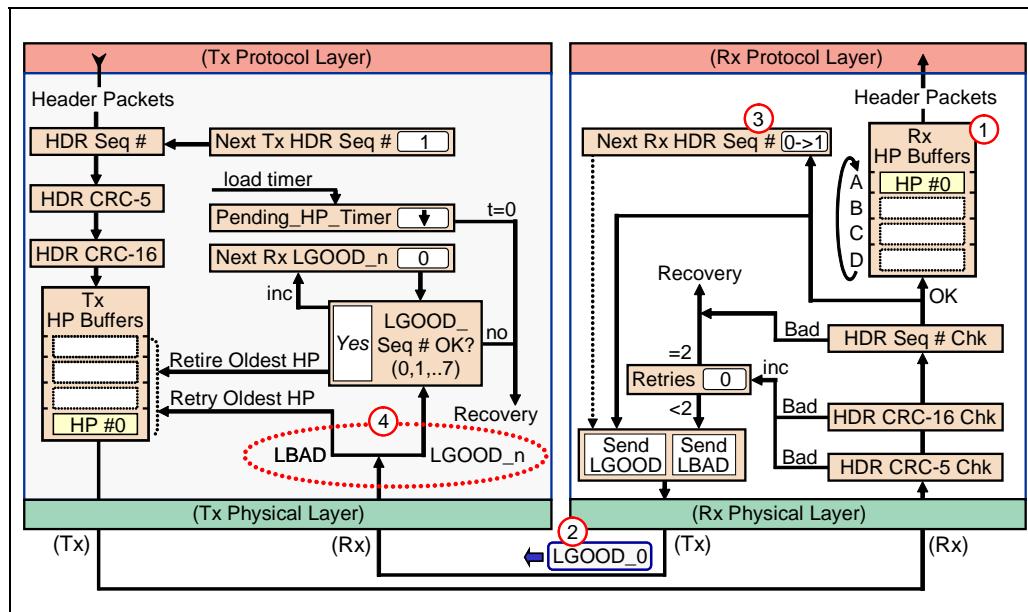
1. The last test before the receiver accepts the header packet is the link header packet sequence number (Hdr Seq#) check. A comparison is made between the actual Hdr Seq# value in the Link Control Word of the header delivered by the transmitter and the expected value tracked by the receiver's Next Rx HDR Seq# counter. If the sequence numbers are the same, the check is successful. If not, a transition from U0 to Recovery is required. In this example, the values match (Hdr Seq# = 0). All that remains is the acknowledgement.
2. The transmitter's Pending\_HP\_Timer continues counting down while it waits for the receiver to return an LGOOD\_n or LBAD response. The transmitter may also send more header packets if it has flow control credits.

# Chapter 16: Link Errors & Packet Acknowledgement

## Receiver Accepts And Acknowledges The Packet

Refer to Figure 16-12 on page 361 during the following discussion of events that occur when the receiver accepts and acknowledges receipt of a valid header packet.

Figure 16-12: Receiver Accepts And Acknowledges A Valid Header Packet



Numbers in the following list refer to the circled numbers in Figure 16-12 on page 361.

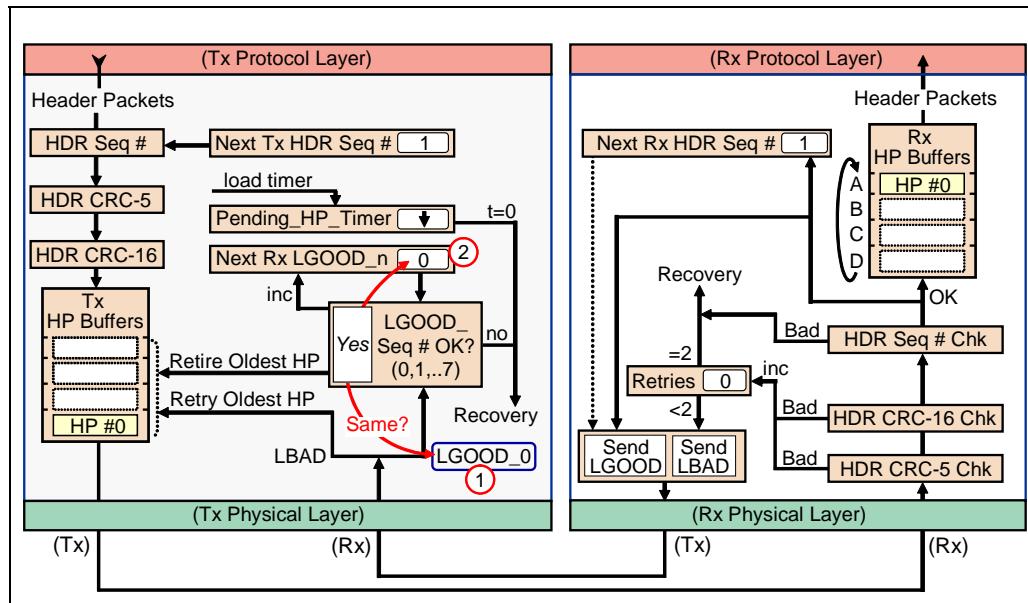
1. Because the inbound header packet (HP #0) passed the CRC-5, CRC-16, and Hdr Seq# checks, it is moved into the next sequential Rx HP Buffer. As this is the first header packet after a reset, this would be Rx HP Buffer A.
2. The receiver uses its transmit interface to send a header packet acknowledgement. Here, the valid header packet was HP #0, so LGOOD\_0 is sent.
3. The receiver also increments the Next Rx HDR Seq# from 0 to 1 in preparation for the next expected inbound header packet (HP #1).
4. The transmitter's Pending\_HP\_Timer continues counting down while it waits for the receiver to return an LGOOD\_n or LBAD response. The transmitter may also send more header packets if it has flow control credits.

# USB 3.0 Technology

## Transmitter Checks LGOOD\_n Acknowledgement

Refer to Figure 16-13 on page 362 during the following discussion of events that occur when the transmitter receives and checks the LGOOD\_n link command from the receiver.

Figure 16-13: Transmitter Checks LGOOD\_n Link Command



Numbers in the following list refer to the circled numbers in Figure 16-13 on page 362.

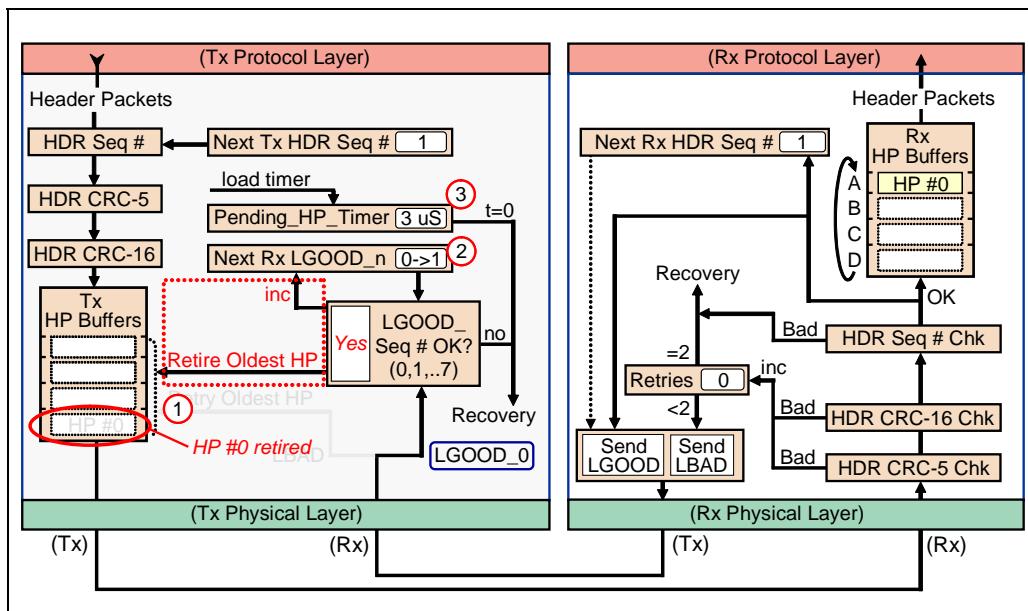
1. Each time the transmitter receives an LGOOD\_n header packet acknowledgement link command, it is required to check its validity. As with all link commands, the first check confirms that both copies of the Link Command Word are the same (if not, the entire link command is discarded).
2. Assuming that the link command was not corrupted, it is decoded and a second check is made of the Next Rx LGOOD\_n counter to determine the expected value of "n" (0-7). In this example, LGOOD\_0 was received and it agrees with the expected value in the Next Rx LGOOD\_n counter so the header packet will be retired. If it had been incorrect, a transition to Recovery would have been required.

# Chapter 16: Link Errors & Packet Acknowledgement

## LGOOD\_n Valid: Retire Header Packet

Refer to Figure 16-14 on page 363 during the following discussion of events that occur when the transmitter receives a valid LGOOD\_n header packet acknowledgement and retires the oldest header packet in the Tx HP Buffers.

Figure 16-14: Transmitter Retires A Header Packet



Numbers in the following list refer to the circled numbers in Figure 16-14 on page 363.

1. Because the LGOOD\_0 header packet acknowledgement link command was expected and received, the transmitter is now permitted to discard its local copy of the header packet. In this case, HP #0 is deleted from the Tx HP Buffers.
2. The transmitter increments the Next RX LGOOD\_n counter (from 0 to 1). This value is now the next expected Hdr Seq# to be acknowledged by the receiver.
3. The Pending\_HP\_Timer is also reloaded with 3uS. Because there are no other header packets outstanding, this timer holds (does not count down).

# USB 3.0 Technology

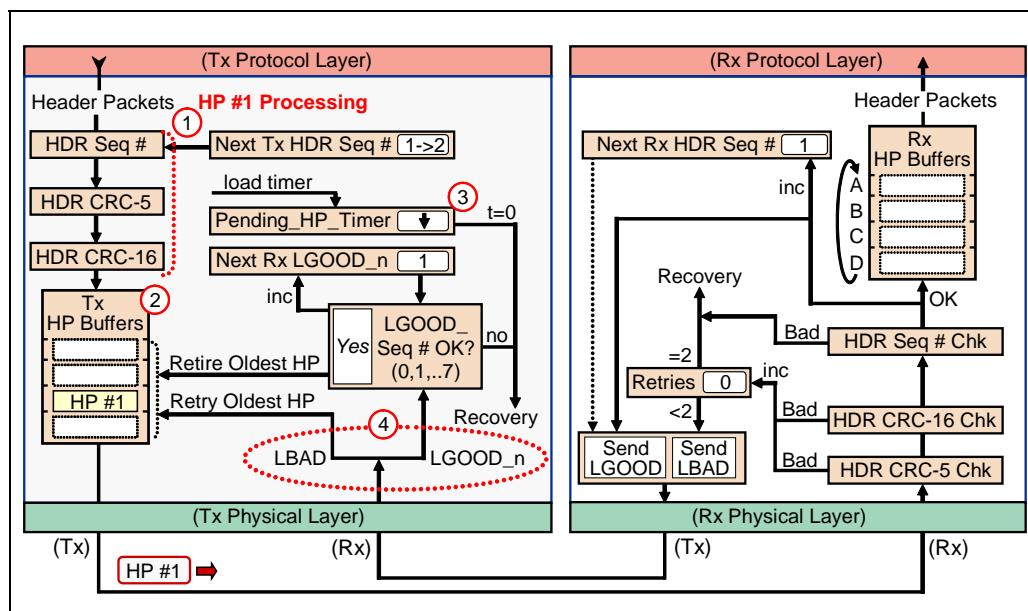
## HP Acknowledgement Sequence: Retry Required

This section describes the general sequence of header packet acknowledgement events when a header packet is corrupted in flight, a CRC error is detected by the receiver, LBAD is signaled, and a retry is performed. Ultimately, the retry results in a successful transfer and the LGOOD\_n header packet acknowledgement from the receiver.

### The Header Packet Is Sent

Refer to Figure 16-15 on page 364 during the following discussion of events that occur when an outbound header is delivered to the transmitter link layer, processed, and sent to the link.

Figure 16-15: A Header Packet Is Processed And Sent



## Chapter 16: Link Errors & Packet Acknowledgement

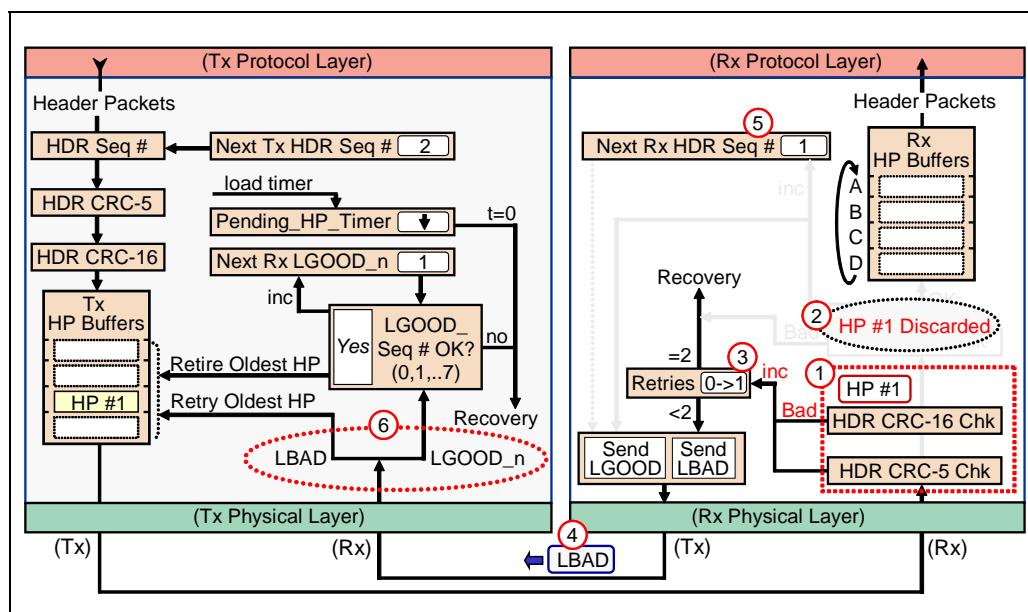
Numbers in the following list refer to the circled numbers in Figure 16-15 on page 364.

1. Each header arriving from the protocol layer is processed at the link layer: the Hdr Seq# is assigned and CRC-5/CRC-16 are generated. For this example, assume that the header has been assigned Hdr Seq# 1; CRC-5 and CRC-16 have been generated and added to the header (which is now complete).
2. A copy of the completed header (HP #1) is placed in the Tx HP buffer. Next, flow control credits are checked (assume that credits are available).
3. The transmitter sends the header packet and activates the 3uS Pending\_HP\_Timer.
4. At this point the transmitter is waiting for a header packet acknowledgement. There will be a Pending\_HP\_Timer timeout in 3uS unless an LGOOD\_n or LBAD link command is returned by the receiver first.

### Header CRC-5 or CRC-16 Check Fails, LBAD Sent

Refer to Figure 16-16 on page 365 during the following discussion of events that occur when the receiver header packet CRC-5 or CRC-16 check fails and an LBAD link command is sent to the transmitter requesting a retry.

Figure 16-16: Header Packet CRC-5, CRC-16 Check Fails, LBAD Sent



## USB 3.0 Technology

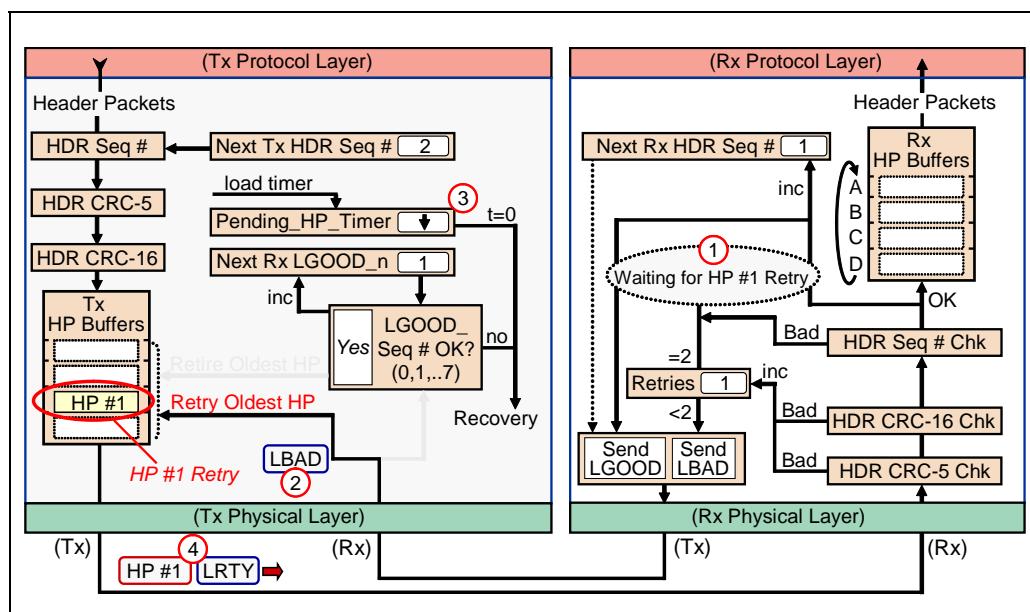
Numbers in the following list refer to the circled numbers in Figure 16-16 on page 365.

1. Each header packet is subjected to header CRC-5 and CRC-16 checks. In this example, assume that the CRC-16 check fails.
2. The header packet is discarded. Any other packets in flight with higher Hdr Seq#s will also be discarded.
3. Next, the receiver checks the number of retry attempts already made for this header packet. Because this will be the first retry attempt for this header packet, the current count = 0 and will increment to 1.
4. An LBAD link command is sent to the transmitter requesting a retry.
5. Next Rx HDR Seq# is not incremented (the next expected Hdr Seq# is still =1). Until the retry completes, any inbound packet with a Hdr Seq# >1 will be discarded.
6. The transmitter's Pending\_HP\_Timer continues counting down while it waits for the receiver to return an LGOOD\_n or LBAD response

### Transmitter Receives LBAD, Starts The Retry

Refer to Figure 16-17 on page 366 during the following discussion of events that occur when the transmitter receives LBAD and starts the header packet retry sequence.

Figure 16-17: Transmitter Receives LBAD, Starts Retry Sequence



## Chapter 16: Link Errors & Packet Acknowledgement

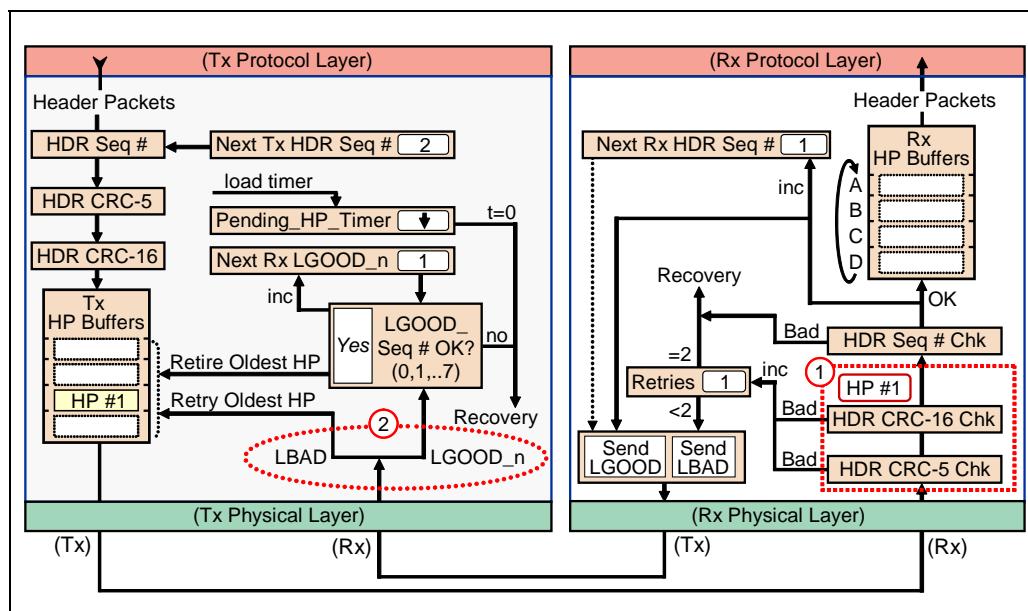
Numbers in the following list refer to the circled numbers in Figure 16-17 on page 366.

1. From the time LBAD is sent, the receiver is waiting for the retry sequence to start. On the link, the start of retry will be signaled by the arrival of the LRTY link command.
2. When the transmitter detects that a valid LBAD has been received, it prepares to retry the oldest unacknowledged header packet in its Tx HP Buffers. Any other header packets in the Tx HP Buffer may follow (in order).
3. Because a retry is to be started, the Pending HP Timer is reloaded with 3uS, but remains halted until the retry commences.
4. When the LRTY link command appears on the link, it is immediately followed by the retry header packet and any other unacknowledged packets in the Tx HP Buffers. The 3uS Pending\_HP\_Timer is again restarted.

### Retry Header Packet CRC-5, CRC-16 Checked

Refer to Figure 16-18 on page 367 during the following discussion of events that occur when the retry header packet is checked at the receiver for CRC errors.

Figure 16-18: Retry Header Packet CRC-5, CRC-16 Is Checked



## USB 3.0 Technology

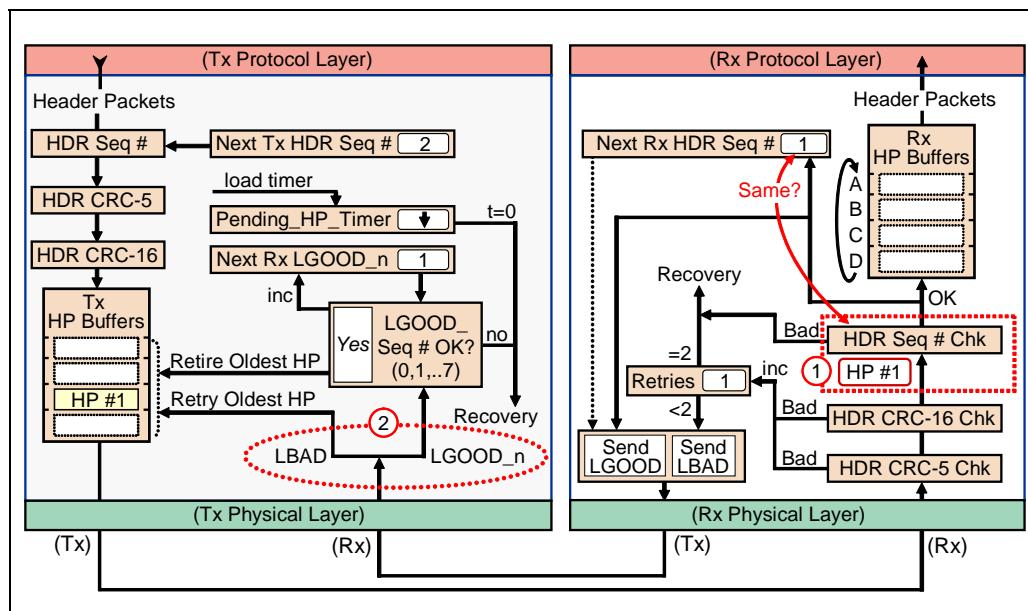
Numbers in the following list refer to the circled numbers in Figure 16-18 on page 367.

1. When the retry header packet arrives at the receiver link layer, CRC-5 and CRC-16 checks are made to determine if it was delivered over the Super-Speed link without bit errors. In this example, assume that both checks are now successful. The receiver prepares to check the header sequence number (Hdr Seq#).
2. The transmitter's Pending\_HP\_Timer continues counting down while it waits for the receiver to return an LGOOD\_n or LBAD response. The transmitter may also send more header packets if it has flow control credits.

### Retry Header Packet Sequence Number Checked

Refer to Figure 16-19 on page 368 during the following discussion of events that occur when the retry header packet passes CRC-5 and CRC-16 checks and the link header packet sequence number (Hdr Seq#) is validated.

Figure 16-19: Retry Header Packet Sequence Number Is Checked



## Chapter 16: Link Errors & Packet Acknowledgement

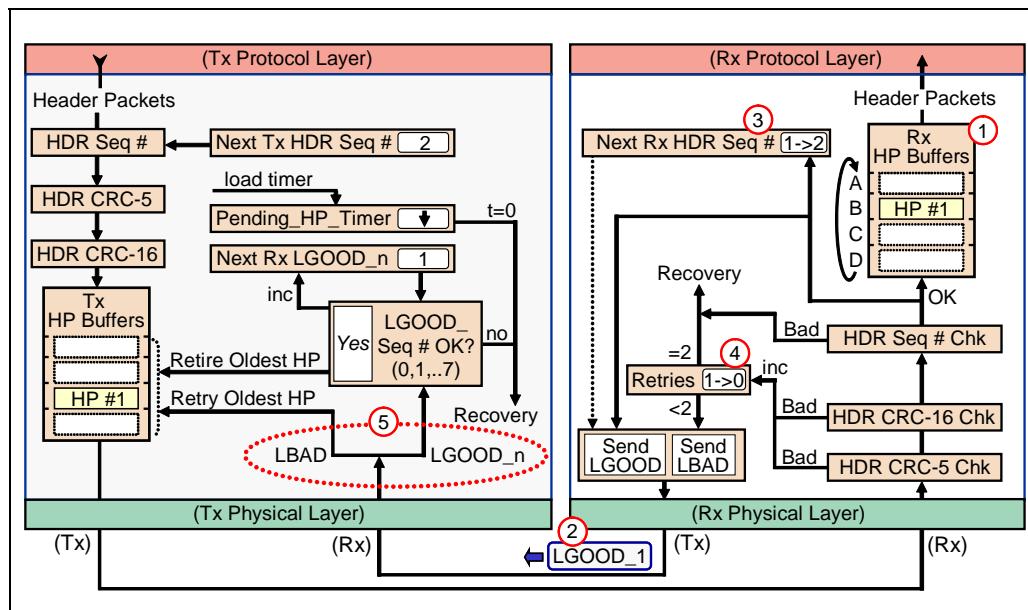
Numbers in the following list refer to the circled numbers in Figure 16-19 on page 368.

1. Next is the link header packet sequence number (Hdr Seq#) check. A comparison is made between the actual Hdr Seq# value in the Link Control Word of the header delivered by the transmitter and the expected value tracked by the receiver's Next Rx HDR Seq# counter. If the sequence numbers are the same, the check is successful. If not, a transition from U0 to Recovery is required. In this example, the values match (Hdr Seq# = 1).
2. The transmitter's Pending\_HP\_Timer continues counting down while it waits for the receiver to return an LGOOD\_n or LBAD response. The transmitter may also send more header packets if it has flow control credits.

### Receiver Acknowledges The Retry Header Packet

Refer to Figure 16-20 on page 369 during the following discussion of events that occur when the receiver accepts and acknowledges the valid retry header packet.

Figure 16-20: Receiver Accepts And Acknowledges The Retry Header Packet



# USB 3.0 Technology

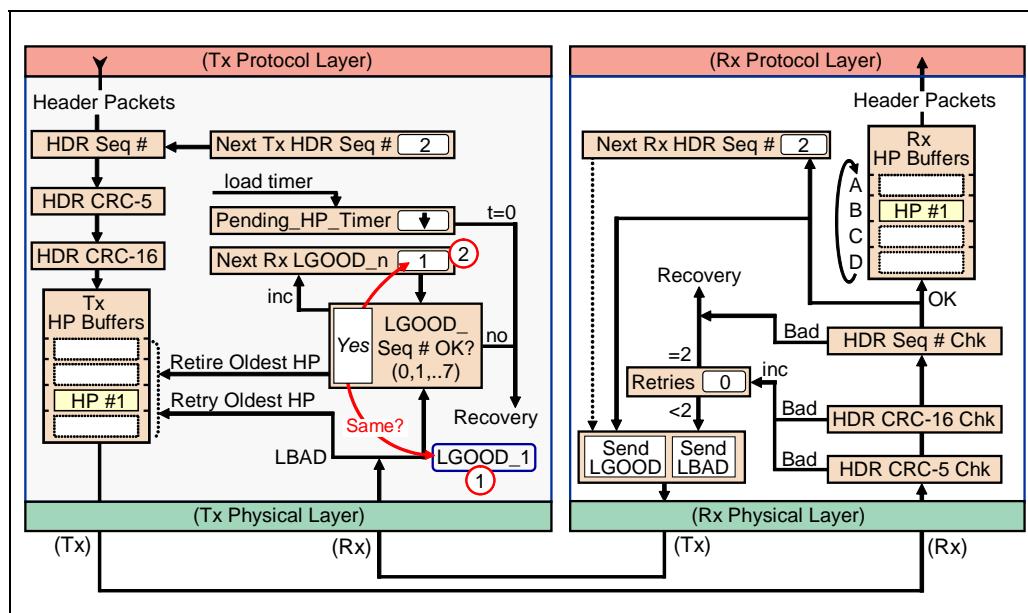
Numbers in the following list refer to the circled numbers in Figure 16-20 on page 369.

1. The retry header packet (HP #1) passed the CRC-5, CRC-16, and Hdr Seq# checks and is moved into the next Rx HP Buffer (Buffer B in this example).
  2. The receiver uses its transmit interface to send a header packet acknowledgement. Here, the valid header packet was HP #1, so LGOOD\_1 is sent.
  3. The receiver also increments the Next Rx HDR Seq# from 1 to 2 in preparation for the next expected inbound header packet (HP #2).
  4. Because the retry was successful, the Retries counter is reset = 0 for the next inbound header packet.
  5. The transmitter's Pending\_HP\_Timer continues counting down while it waits for the inbound LGOOD\_n or LBAD response. The transmitter may also send more header packets if it has flow control credits.

## **Transmitter Checks LGOOD\_n Following The Retry**

Refer to Figure 16-21 on page 370 during the following discussion of events that occur when the transmitter validates the LGOOD\_n link command.

Figure 16-21: Transmitter Validates LGOOD  $n$  Received After Retry



## Chapter 16: Link Errors & Packet Acknowledgement

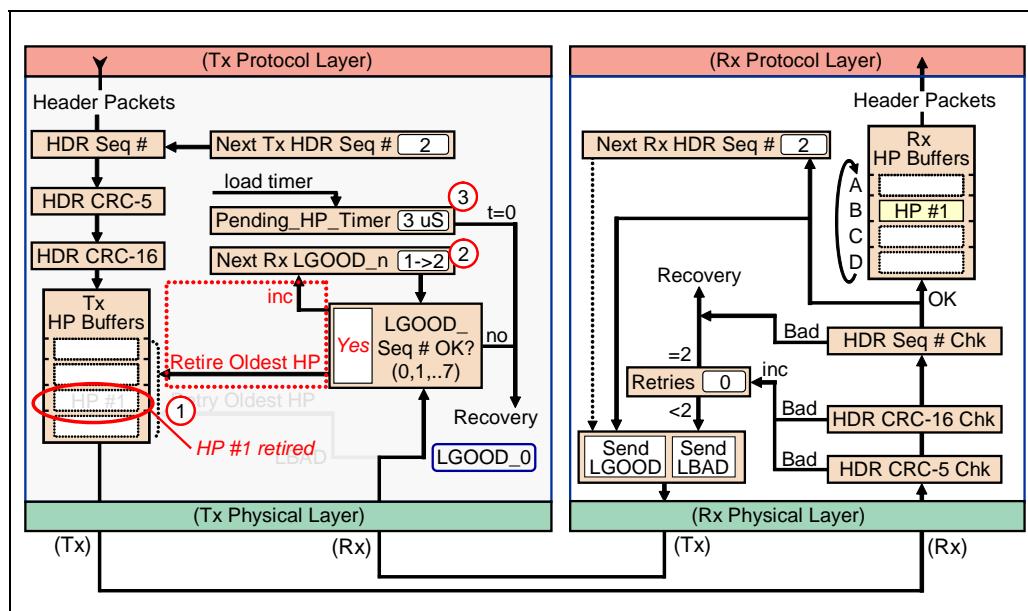
Numbers in the following list refer to the circled numbers in Figure 16-21 on page 370.

1. Following the retry, the transmitter receives an LGOOD\_n header packet acknowledgement link command and checks its validity. As with all link commands, the first check confirms that both copies of the Link Command Word are the same (if not, the entire link command is discarded).
2. Assuming that the link command was not corrupted, it is decoded and a second check is made of the Next Rx LGOOD\_n counter to determine the expected value of "n" (0-7). In this example, LGOOD\_1 was received and it agrees with the expected value in the Next Rx LGOOD\_n counter so the header packet (HP #1) will be retired. If it had been incorrect, a transition to Recovery would have been required.

### LGOOD\_n Valid: Retire The Header Packet

Refer to Figure 16-22 on page 371 during the following discussion of events that occur when the transmitter receives a valid LGOOD\_n and retires the oldest header packet in the Tx HP Buffers (the one it was requested to retry).

Figure 16-22: Transmitter Retires The Header Packet After Retry



## **USB 3.0 Technology**

---

Numbers in the following list refer to the circled numbers in Figure 16-22 on page 371.

1. Because the LGOOD\_1 header packet acknowledgement link command was expected and received, the transmitter is now permitted to discard its local copy of the header packet. In this case, HP #1 is deleted from the Tx HP Buffers.
2. The transmitter increments the Next RX LGOOD\_n counter (from 1 to 2). This value is now the next expected Hdr Seq# to be acknowledged by the receiver.
3. The Pending\_HP\_Timer is also reloaded with 3uS. Because there are no other header packets outstanding, this timer holds (does not count down).

---

---

# **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.

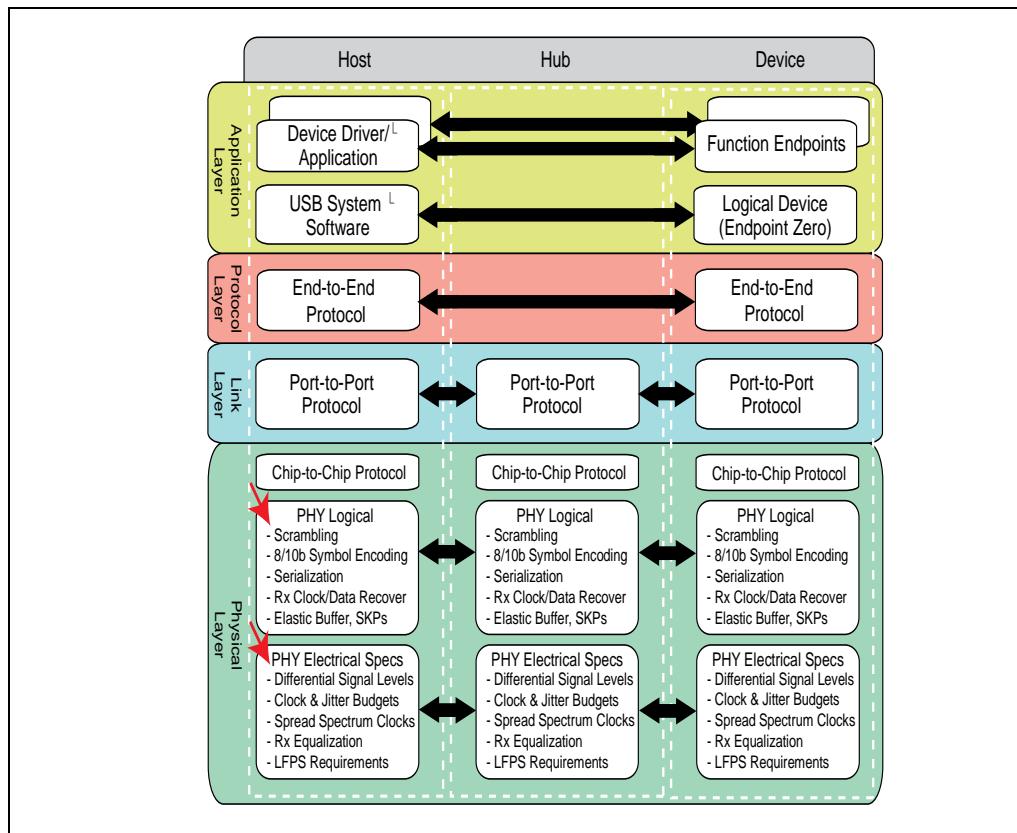
# USB 3.0 Technology

## 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



## **Chapter 17: Introduction to Chip-To-Chip Protocol**

---

### **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:

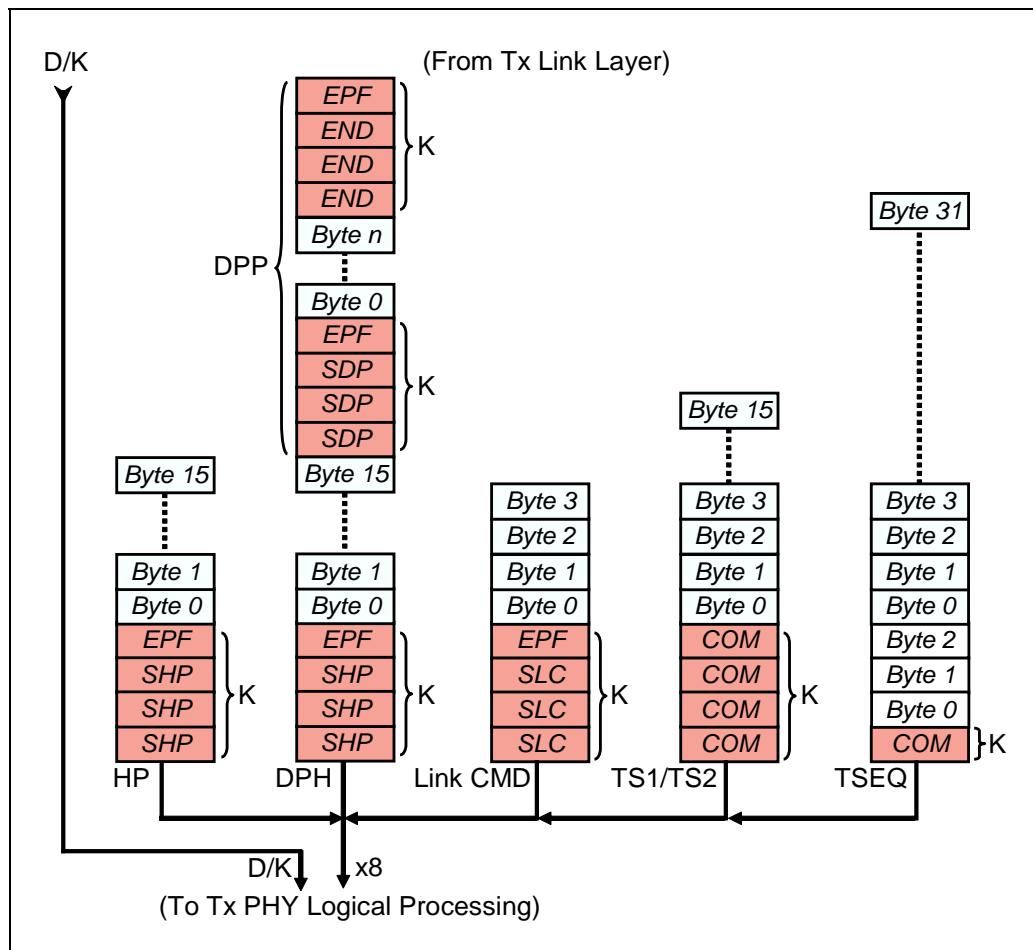
- A byte is an 8-bit value before it is encoded by the transmitter; depending on where in the logical processing sequence it is, the byte may or may not be scrambled. At the receiver, a byte is an 8-bit value after it is decoded; again, it may or may not have been de-scrambled.
- A symbol is the 10-bit 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.

# USB 3.0 Technology

## A Few Common Structure D/K Examples

Some common structures to be processed by the physical (PHY) layer logical section are shown in Figure 17-2 on page 376. Note the fact that the COM and framing ordered sets will all be processed as control (K) bytes; the contents of header packets, data packets, and even training sequences TS1, TS2, and TSEQ will all be processed as data (D) bytes.

Figure 17-2: Common Traffic And Data/Control Bytes

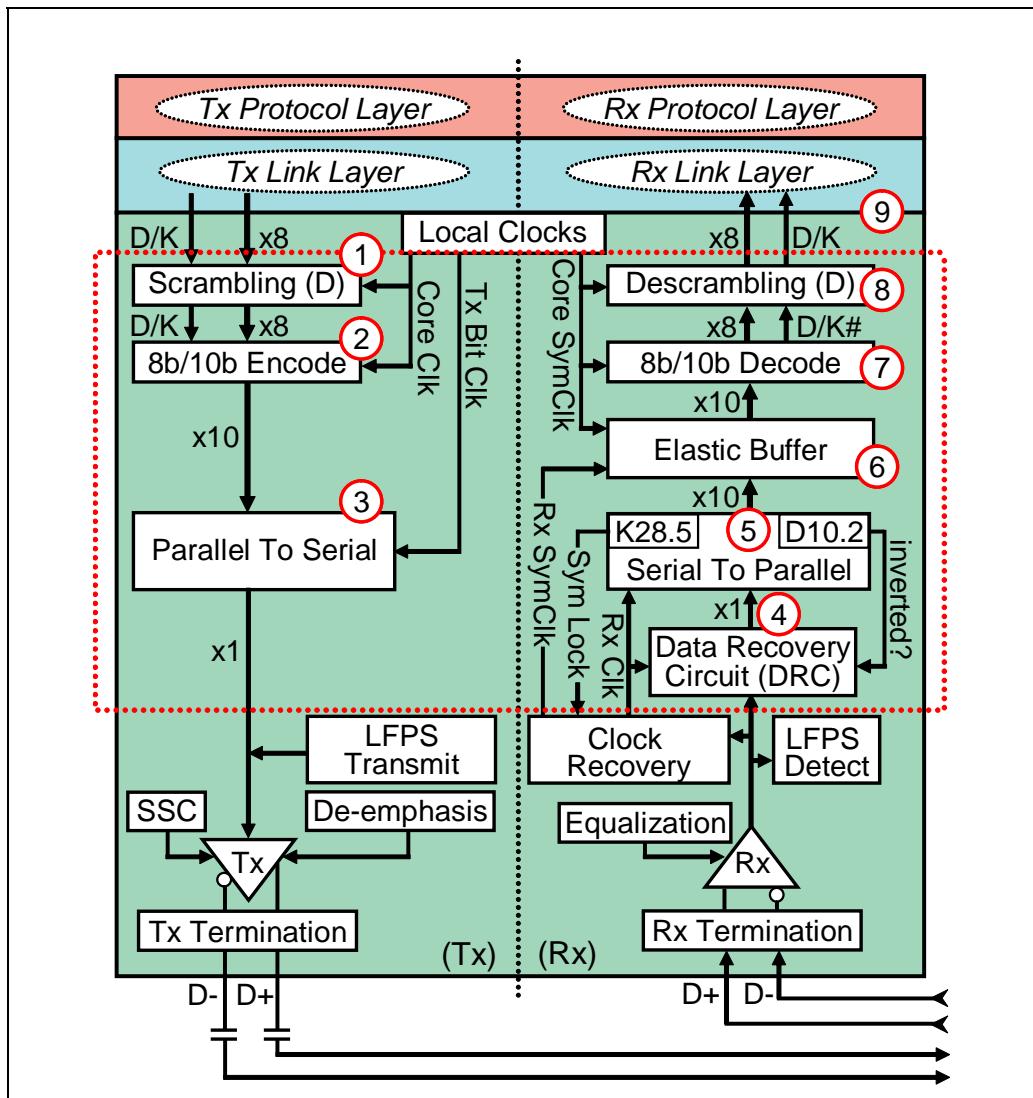


## Chapter 17: Introduction to Chip-To-Chip Protocol

### The Physical Layer: PHY Logical Processing

This section identifies functional blocks related to PHY logical processing. Elements in Figure 17-3 on page 377 are summarized here and described in Chapter 18, entitled "Physical Layer Logical Functions," on page 383.

Figure 17-3: Physical Layer Logical Processing: Scope



## **USB 3.0 Technology**

---

---

### **Tx Scrambling**

Note (1) in Figure 17-3 on page 377. The purpose of scrambling is to reduce EMI by randomizing the outbound bytes before transmission. Bytes which will become “D” symbols are scrambled, bytes which will become “K” symbols are not scrambled.

---

### **Tx 8b/10b Encoding**

Note (2) in Figure 17-3 on page 377. After scrambling, each byte is encoded into a 10 bit symbol. The 10-bit values allow unique encodings for “D” and “K” symbols and simplify recognition of packet delimiters and ordered sets at the receiver.

---

### **Tx Serialization**

Note (3) in Figure 17-3 on page 377. The 10-bit symbol output from encoding is then serialized and shifted onto the 5 Gb/s SuperSpeed link, one bit at a time by the transmitter’s differential driver.

---

### **Rx Clock/Data Recovery**

Note (4) in Figure 17-3 on page 377. When operating in link state U0, all traffic is sent and received serially at 5 Gb/s. During link training, receivers train equalization logic so that a valid input signal level is available and the recovered clock is positioned such that each bit is sampled at the correct time.

---

### **Rx De-serialization**

Note (5) in Figure 17-3 on page 377. Assuming symbol lock was achieved during link training/retraining, each 10 bits from the 5 Gb/s SuperSpeed link at the differential receiver is a new 10-bit symbol. The receiver must recover 10-bit symbols in order to perform a decode.

## **Chapter 17: Introduction to Chip-To-Chip Protocol**

---

### **Rx Elastic Buffer**

Note (6) in Figure 17-3 on page 377. The elastic buffer at the physical layer of each receiver helps compensate for the expected mismatch between the recovered transmitter clock and the local clock of the receiver. If functioning correctly, symbols may be clocked into and out of the elastic buffer at slightly different rates without loss of any meaningful information.

---

### **Rx 8b/10b Decoding**

Note (7) in Figure 17-3 on page 377. As each 10-bit symbol is shifted out of the elastic buffer, 8b/10b decode logic produces one byte accompanied by a "D/K" indication. In addition, the receiver PHY checks that the symbol meets disparity requirements and decodes into a valid value.

---

### **Rx De-scrambling**

Note (8) in Figure 17-3 on page 377. The final processing step is de-scrambling. Only "D" symbols are descrambled; the purpose is to recover the original logical byte from the scrambled value.

---

### **Receiver PHY Forwards Traffic to Link Layer**

Note (9) in Figure 17-3 on page 377. At this point, framing symbols are stripped off and the header packet, data packet, or link command is forwarded to the receiver's link layer for CRC and other checks. Each byte is accompanied by a "D/K" flag indication.

---

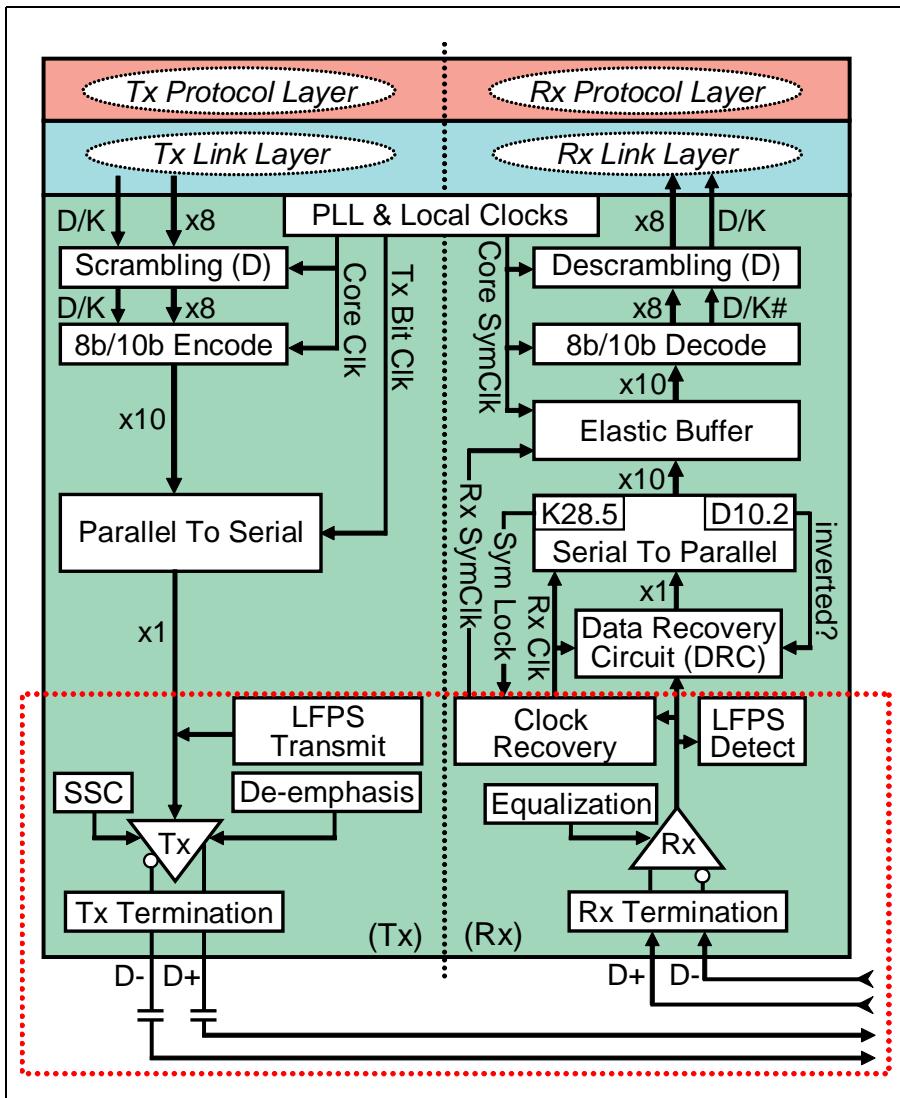
## **The Physical Layer: PHY Electrical Specifications**

As indicated in Figure 17-4 on page 380, this section lists physical layer topics related to Chip-to-Chip PHY electrical specifications. For details on these topics, refer to Chapter 25, entitled "SuperSpeed Signaling Requirements," on page 591.

## USB 3.0 Technology

---

Figure 17-4: Physical Layer Electrical Specifications: Scope



# **Chapter 17: Introduction to Chip-To-Chip Protocol**

---

## **General Topics**

- Normative (required) and informative specifications
  - Jitter Budget allocation for transmitters, receivers, and the interconnection media (cables, connectors, etc.)
- 

## **Transmitter PHY Electrical Topics**

For more details on any of the following topics, refer to “SuperSpeed Transmitter Requirements” on page 596

- Transmitter SuperSpeed voltage and timing requirements
  - Transmitter low power option
  - Transmitter de-emphasis requirements
  - Transmitter spread spectrum clocking (SSC)
- 

## **Receiver PHY Electrical Topics**

For more details on any of the following topics, refer to “SuperSpeed Receiver Requirements” on page 605

- Receiver SuperSpeed voltage and timing requirements
  - Receiver equalizer training
  - Receiver termination
  - Receiver termination detection (Rx.Detect) method
- 

## **Low Frequency Periodic Signaling (LFPS)**

For more details on any of the following topics, refer to “Low Frequency Periodic Signaling Requirements” on page 612

- LFPS signaling events
- LFPS timing requirements
- Electrical specifications

## **USB 3.0 Technology**

---

---

# 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<sub>BUS</sub> 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.

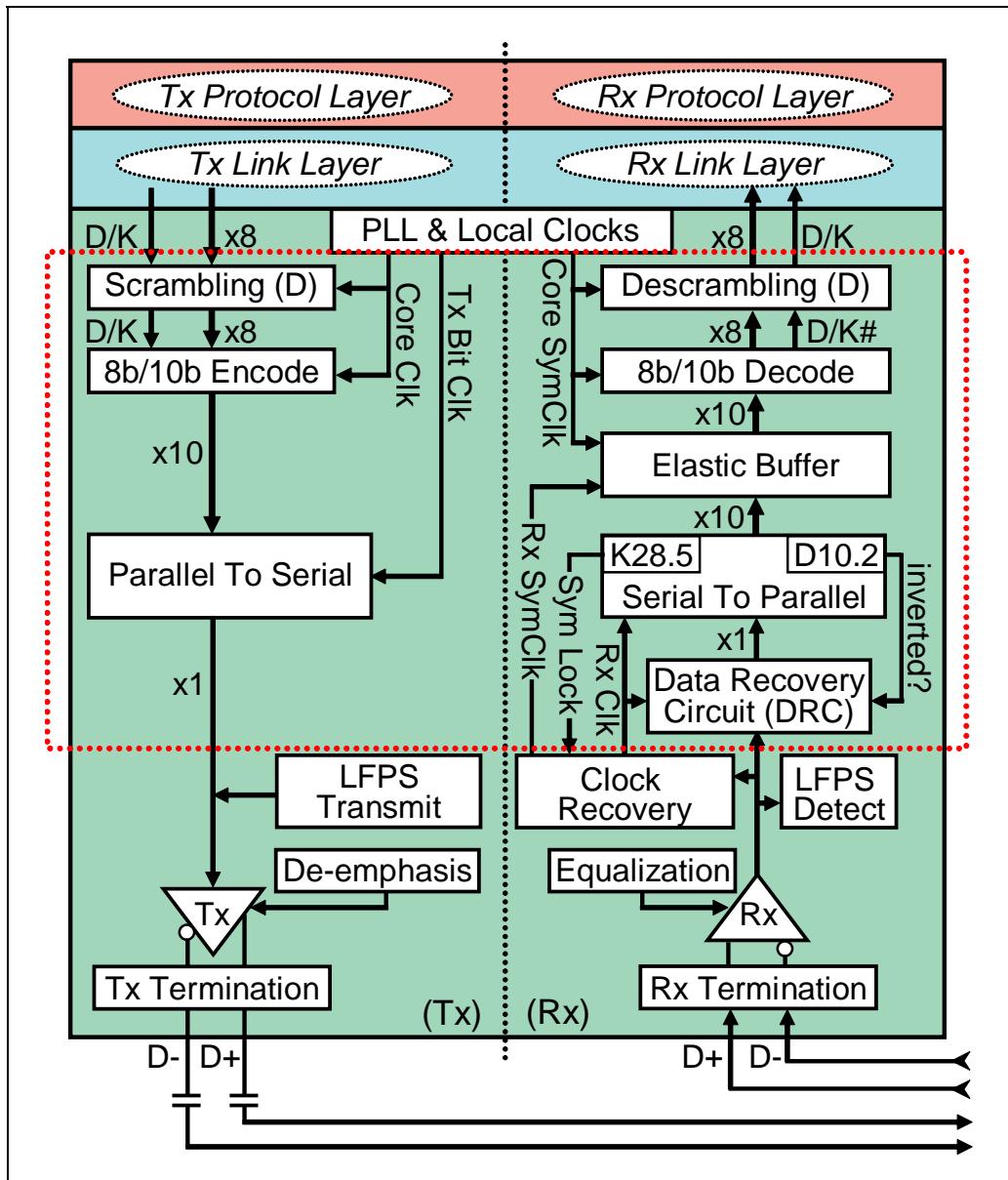
---

## 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.

## USB 3.0 Technology

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



## Chapter 18: Physical Layer Logical Functions

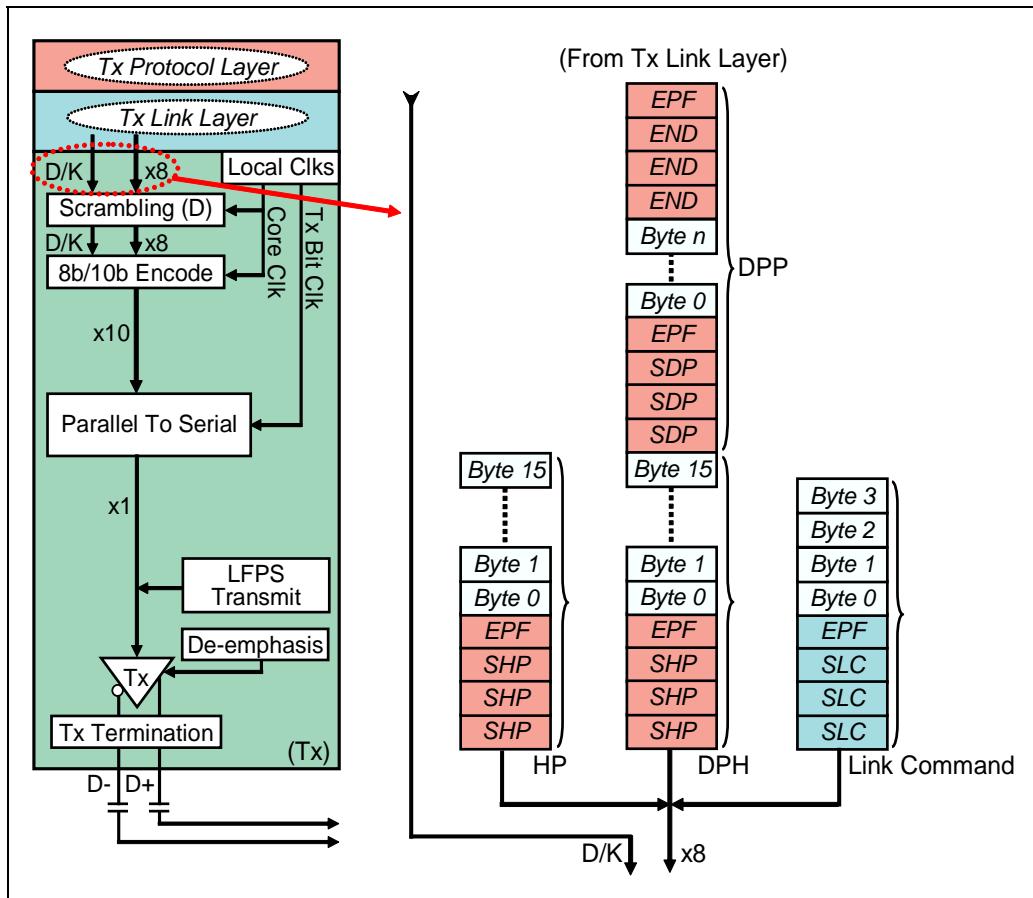
## **Tx Physical Layer (PHY) Logic**

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*



## **USB 3.0 Technology**

---

Outbound traffic from the link layer is passed into the physical layer and processed. Each byte is accompanied by the D/K flag bit indicating whether it will become a data (D) byte or a control (K) byte. As can be seen in Figure 18-2 on page 385, the traffic arrives as:

- Framed Header Packets. These include sixteen bytes of header information preceded by the four control byte HPSTART framing.
- Framed Data Packets. The data packet header (DPH) is sixteen bytes and preceded by the four control byte HPSTART framing. The data packet payload (DPP) immediately follows the DPH and is preceded by the four control byte DPPSTART framing and framed at the back by the four control byte DPEND framing.
- Framed Link Commands. These consist of 4 bytes of Link Command Word information (actually two Link Command Words, repeated twice) preceded by the four control byte LCSTART framing.

Not shown here, but always present, are other outbound bytes including skip (SKP) ordered sets and logical idle.

It is a Tx PHY logic responsibility to track the D/K flag state for each of the bytes in the outbound traffic as control (K) bytes and data (D) bytes are handled differently at the scrambling and 8b/10b stages.

---

## **Data Scrambling**

### **General**

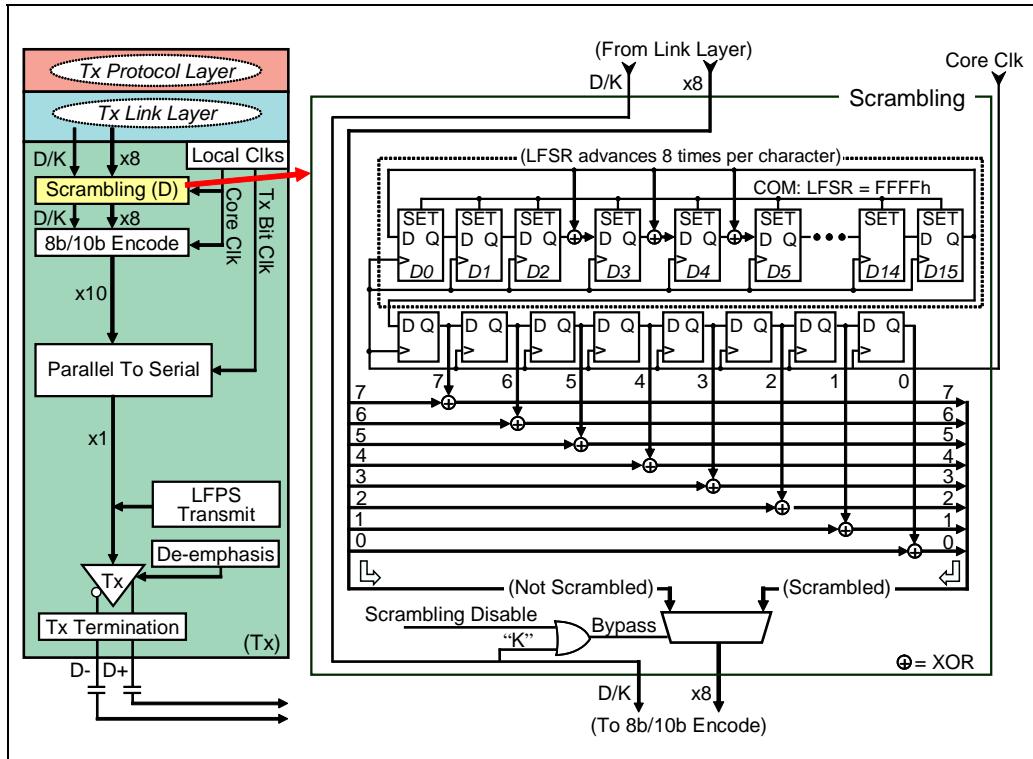
In this step, the Tx PHY logic scrambles outbound data (D) bytes to randomize information and reduce electromagnetic interference (EMI). Note that the term data (D) byte is used loosely here and taken to mean any byte that is not a control (K) byte. Data (D) bytes don't include framing, but do include:

- All bytes in a 16-byte header
- All bytes in a data packet payload, including CRC-32
- Both bytes in the link command Link Command Word field

As can be seen conceptually in Figure 18-3 on page 387, the scrambler is a free running LFSR (Linear Feedback Shift Register) implemented in hardware. In the illustration, the bypass path for control (K) symbols is also indicated. Note that each time a COM byte is recognized, the scrambler LFSR is reset to FFFFh.

## Chapter 18: Physical Layer Logical Functions

Figure 18-3: Transmitter PHY Logic Scrambling



While the details of hardware implementation are left up to the designers, the basic operation of the scrambling LFSR is as follows:

- Scrambling of an incoming data (D) byte is done by serially XORing (a logical exclusive OR) the 8-bit data byte with the 16-bit output of the LFSR
- LFSR output bit 15 is XORed with data bit 0 (D0) of the data byte, then the LFSR is advanced and the step repeated for data bits 1-7.

Some additional rules concerning the scrambler LFSR

- The scrambler polynomial is:  $G(X)=X^{16} + X^5 + X^4 + X^3 + 1$
- All data (D) bytes, except those used in training sequences, are scrambled
- The LFSR advances by 8 serial shifts for each byte processed, except SKP
- Control (K) bytes are never scrambled. A primary reason for this is the receiver PHY must recognize framing symbols to properly handle incoming

# USB 3.0 Technology

---

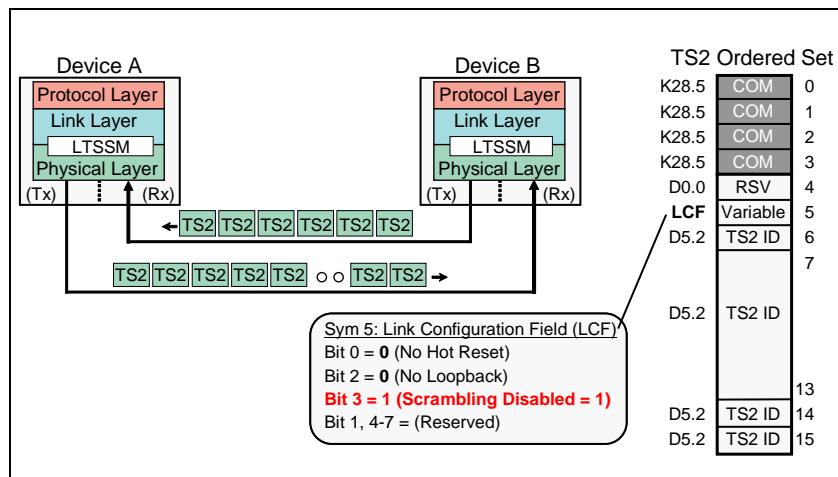
packets and link commands (descrambling logic is much later in the Rx PHY logic pipeline).

- The initial value of the LFSR is FFFFh and is initialized each time an out-bound COM is processed by the transmitter.

## Disabling Scrambling

As shown in Figure 18-4 on page 388, The capability to disable scrambling is exposed as a bit in the TS2 training sequence ordered set (byte 5). During the link training Polling.Configuration substate and the Recovery.Configuration substate, link partners exchange TS2s and can disable scrambling at that time.

Figure 18-4: TS2 Scrambling Disable Bit



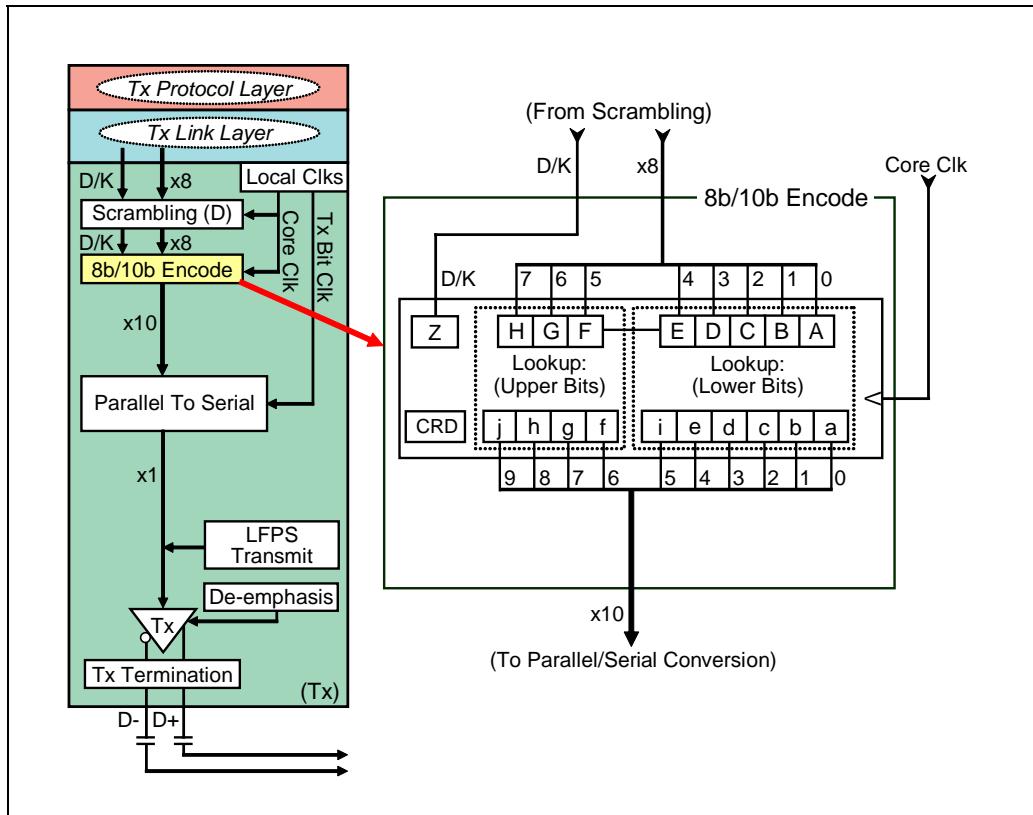
---

## 8b/10b Encoding

Next, the Tx PHY logic encodes each data (D) and control (K) byte into a 10-bit symbol. The encoding is compliant with ANSI INCITS 230-1994, clause 11 and the conceptual logic to accomplish this is shown in Figure 18-5 on page 389.

## Chapter 18: Physical Layer Logical Functions

Figure 18-5: Transmitter PHY 8b/10b Encoding



### The Purpose Of Encoding

Expanding 8-bit data (D) and control (K) bytes into 10-bit symbols represents a 25% performance impact, but has important benefits including:

- 1024 symbol set enables unique encoding of data (D) and control (K) values. Symbol 10-bit patterns also support DC-balance and the required current running disparity (CRD) correction scheme.
- Symbols have 10-bit patterns with sufficient transition density to embed clock information recoverable by the receiver.

# **USB 3.0 Technology**

---

## **Two Key 8b/10b Encoding Rules**

- Each 10-bit encoded symbol contains six 1's and four 0's, six 0's and four 1's, or five 1's and five 0's. Symbol encodings with more than six bits of one polarity are not valid.
- The D/K bit accompanying each byte into the encoder assures that control (K) symbols and data (D) symbols are unique sets. Even if a data byte and control byte are the same numerically, their symbol encodings are different.

## **Some 8b/10b Encoding Examples**

Referring to Figure 18-5 on page 389 above and to Table 18-1 below, some examples of 8b/10b encoding and the 10-bit results can be seen. The first symbols in Table 18-1 are control (K); the last two entries in Table 18-1 are data (D) symbol encodings. On the right side of the table are the two encodings for each symbol; alternative encodings support the balanced running disparity scheme described next.

*Table 18-1: 8b/10b Data And Control Symbol Encoding Examples*

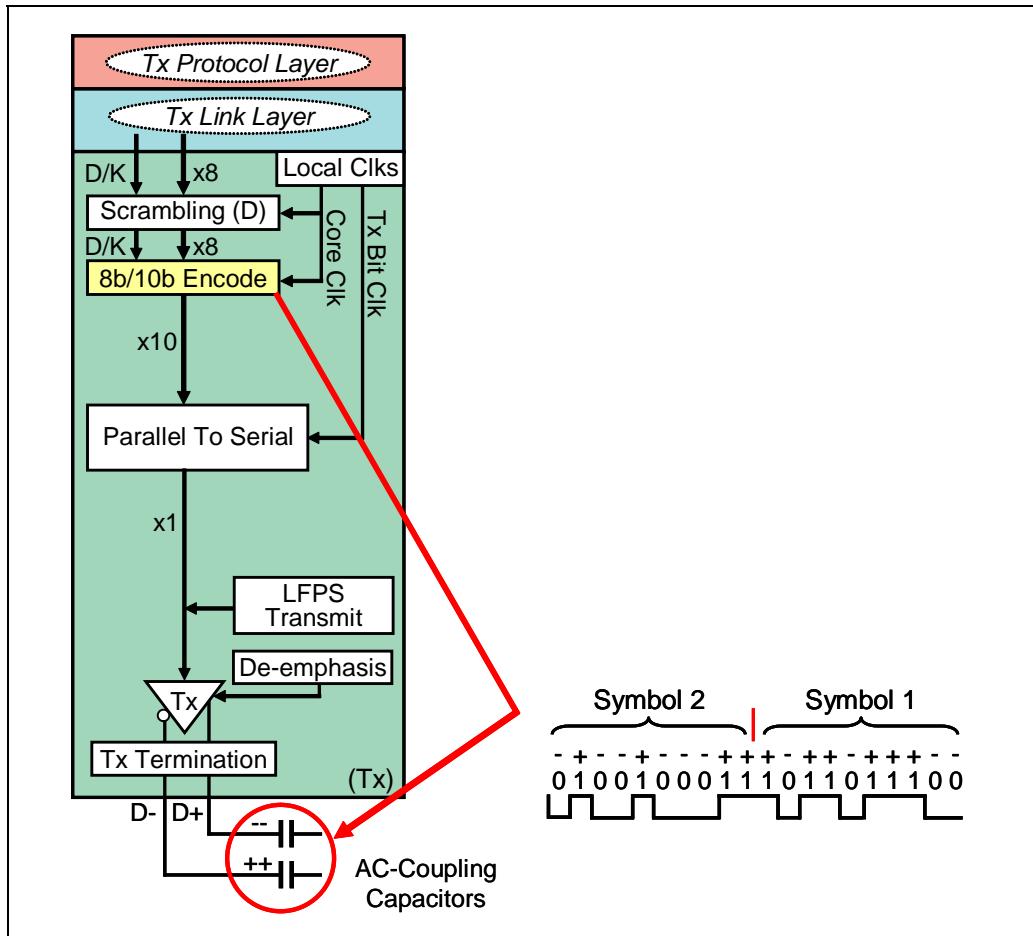
Encoding	Symbol Name	Logical (hex)	Input Bits HGF EDCBA	CRD- Symbol abcdeifghj	CRD+ Symbol abcdeifghj
Control (K) Symbol Encoding Examples					
K28.1	SKP	3C	001 11100	001111 1001	110000 0110
K28.2	SDP	5C	010 11100	001111 0101	110000 1010
K28.3	EDB	7C	011 11100	001111 0011	110000 1100
K28.4	SUB	9C	100 11100	001111 0010	110000 1101
K23.7	EPF	F7	111 10111	111010 1000	000101 0111
K27.7	SHP	FB	111 11011	110110 1000	001001 0111
K29.7	END	FD	111 11101	101110 1000	010001 0111
K30.7	SLC	FE	111 11110	011110 1000	100001 0111
Data (D) Symbol Encoding Examples					
D0.0	00	00	000 00000	100111 0100	011000 1011
D5.0	05	05	000 00101	101001 1011	101001 0100

# Chapter 18: Physical Layer Logical Functions

## 8b/10b Encoding And Current Running Disparity

As described in the previous section, the 8b/10b encoding scheme actually supports two possible encoded values for each data and control byte. This is because transmitters are required to assure that outbound serial data, over time, contains an equal (balanced) number of 0's and 1's. The on-going difference in the number of transmitted bits of each polarity is referred to as the running disparity. Tracking the current running disparity (CRD) and correcting any imbalance is critical on the AC-coupled SuperSpeed link and one of the important motivations for 8b/10b encoding.

Figure 18-6: 8b/10b Encoding Supports Link CRD Scheme



# **USB 3.0 Technology**

---

## **The AC-Coupled SuperSpeed Link**

A SuperSpeed link is comprised of two sets of uni-directional signals. The physical layer transmitter in each of the link partners drives serial data onto the link differentially as indicated in Figure 18-6 on page 391. The physical link connection employs a 75nF-200nF capacitor in series with each signal line; the capacitors are in close proximity to (or integrated within) the transmitter. This connection method is referred to as AC-coupling and it has significant advantages in the USB 3.0 SuperSpeed 5Gb/s cabled signaling environment, but also presents a few challenges:

**Advantages of AC-Coupling.** The AC-coupling capacitors eliminate DC common mode voltage sharing and enable transmitters to operate at independent common mode voltage levels. Device power and ground planes are isolated from those of the link partner.

**Challenges of AC-Coupling.** A significant challenge of AC-coupling on the SuperSpeed link, and the one addressed by the current running disparity (CRD) scheme described in this section, is caused by data-dependent offset voltage on AC-coupling capacitors resulting from signal lines spending more time in one logic state than in the other. For example, if thousands of bits were transmitted and the vast majority were logical 1's, a positive charge would build up on the coupling capacitor in the signal path. As this DC voltage offset caused by the cumulative disparity builds, the ability of the capacitor to pass AC signals is diminished. Ideally, for each logical "1" sent, a logical "0" would be sent as well--maintaining a DC balance.

## **8b/10b Encoding Minimizes Disparity**

As described previously, each symbol out of the 8b/10b encoder has one of the following 10-bit properties:

- It is comprised of six 1's and four 0's (a positive disparity) OR
- It is comprised of six 0's and four 1's (a negative disparity) OR
- It is comprised of five 1's and five 0's (a neutral disparity)

As each 10-bit symbol is output from the 8b/10b encoder, the encoder's current running disparity tracking is updated; the range of CRD = 0, +2, or -2 based on the last symbol processed. If the first 10-bit symbol sent had six 1's and four 0's, then CRD = +2 and when the next symbol is sent, the encoder would attempt to correct the CRD by selecting the more negative encoding alternative to bring the CRD back in balance. Note that if the next symbol is neutral (five 1's and five 0's), it has no impact on the current running disparity.

# Chapter 18: Physical Layer Logical Functions

## An Example

Assume that the CRD = +2 (based on the previous symbol processed by the encoder). The selection of the encoding alternative for the next symbol is dependent on the CRD and follows the guidelines summarized in Table 18-2 on page 393 (note the CRD- and CRD+ headings at the top right of the table):

- If the next symbol is to be K28.1, then the fact that both of its encodings are unbalanced requires the selection of the encoding alternative at the far right to be used as it has more 0's than 1's and brings CRD back into balance.
- Instead, if the next symbol is to be D0.0, then the fact that both encodings are neutral means that neither would have an impact on CRD, but the USB 3.0 specification still says that if CRD is currently positive (CRD+), then the encoding at the far right should be used.

Table 18-2: 8b/10b Encoder Selects The Best CRD Alternative

Encoding	Symbol Name	Logical (hex)	Input Bits HGF EDCBA	CRD- Symbol abcdeifghj	CRD+ Symbol abcdeifghj
Control (K) Symbol Encoding Example					
K28.1	SKP	3C	001 11100	001111 1001	110000 0110
Data (D) Symbol Encoding Example					
D0.0	00	00	000 00000	100111 0100	011000 1011

## Is The Transmitter CRD Choice Correct?

Because both of the alternative encodings for each symbol are correct logically, the question of whether the transmitter is observing the CRD rules arises. If the transmitter isn't following the CRD rules, link reliability will eventually be compromised.

The USB 3.0 specification requires the receiver to check CRD as symbols are processed at the physical layer; it should report CRD violations to its link layer. Because detected CRD violations are often the result of a flipped bit on the link, the problem may be resolved at the receiver's upper layers:

- For CRD violations that occur during the transmission of a header packet, the receiver link layer will detect a failed CRC-5 or CRC-16.
- For CRC violations that occur during the transmission of a framing symbol, the link layer generally recovers silently if all other symbols in the framing ordered set are correct.

# USB 3.0 Technology

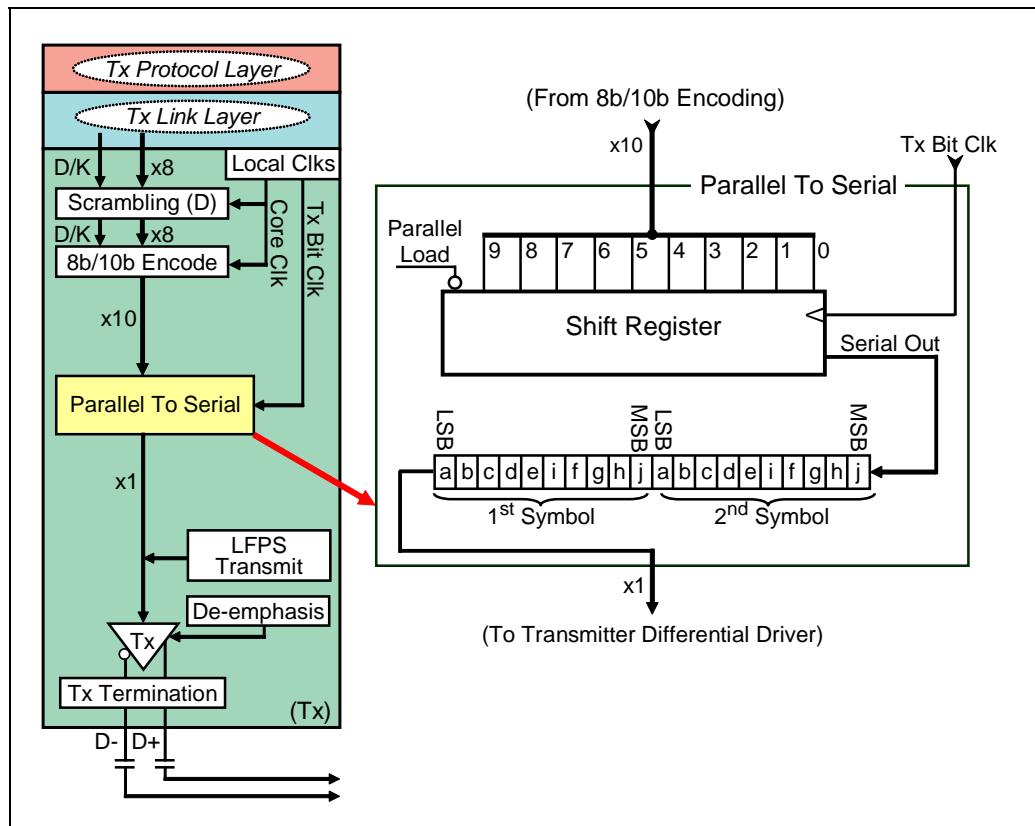
- If the CRD violation occurs during the sending of a data packet payload, the CRC-32 check performed by the receiver's protocol layer will result in an ACK with the indication that an End-To-End packet retry is necessary.

## Parallel To Serial Data Conversion

As indicated in Figure 18-7 on page 394, the Tx PHY logic serializes each outbound 10-bit symbol for transmission on the 5Gb/s SuperSpeed link. The input to serialization logic runs at 1/10th of the serial bus rate as each symbol is loaded. The serial output is clocked at the Tx bit rate, nominally 5Gb/s, but subject to spread-spectrum clocking (SSC) modulation.

In the illustration, note the transmission bit order for each symbol sent to the link (least significant bit first).

Figure 18-7: Transmitter PHY Parallel To Serial Data Conversion

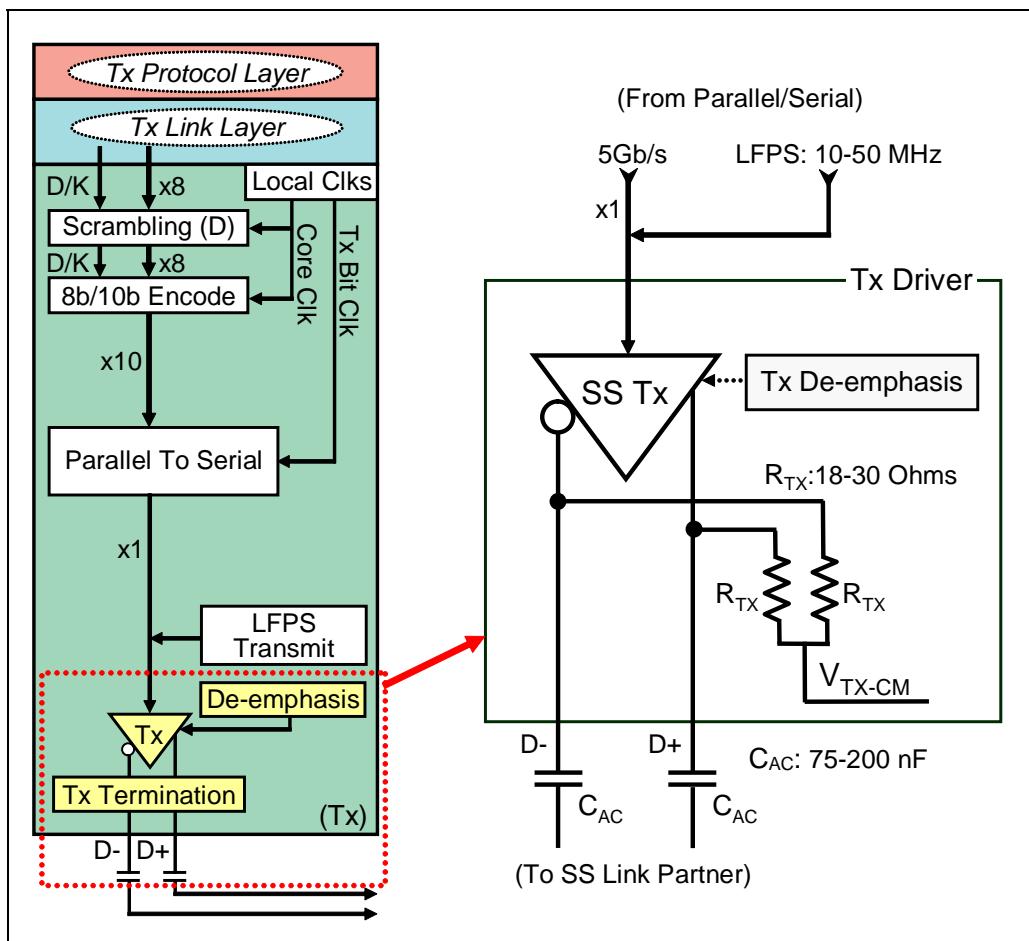


# Chapter 18: Physical Layer Logical Functions

## Differential Transmitter

The serial output is then clocked onto the SuperSpeed link using the differential transmitter as shown in Figure 18-8 on page 395. The details of the differential transmitter electrical specifications are described in Chapter 25, entitled "Super-Speed Signaling Requirements," on page 591.

Figure 18-8: The Differential Transmitter



# USB 3.0 Technology

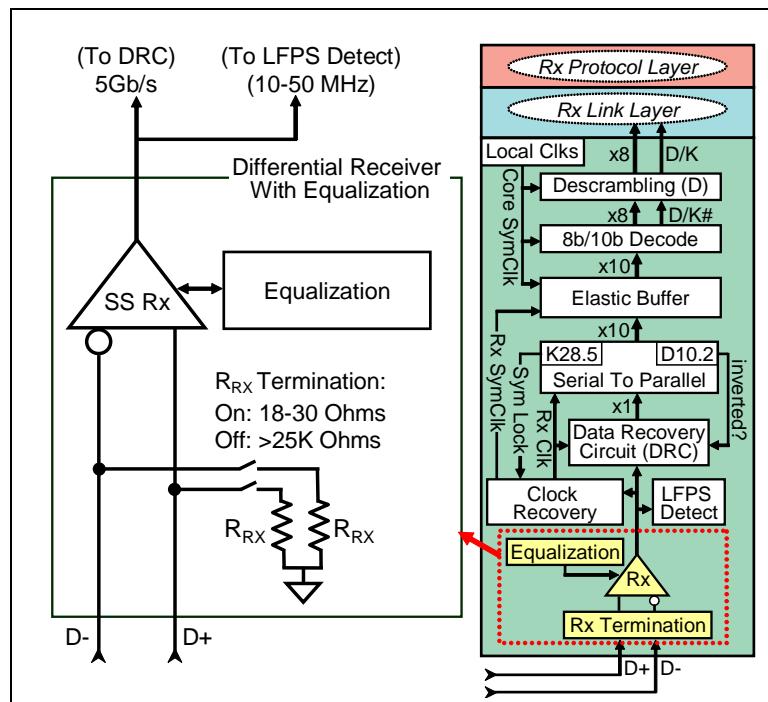
## Rx Physical Layer (PHY) Logic

When the link is operating at 5Gb/s SuperSpeed, the receiver continually processes inbound header packets, link commands, ordered sets, and logical idle symbols. The following sections describe the elements of receiver PHY logic. For context, the differential receiver and some other elements of the PHY electrical section are mentioned here, but are described in more detail later.

### Differential Receiver And Equalization

The differential receiver is shown conceptually in Figure 18-9 on page 396. During link training, the Rx equalizer is trained and parameters are saved and reused during link power management exits and other transitions to LTSSM Recovery. The details of the differential receiver, termination, and equalizer electrical specifications are described in Chapter 25, entitled "SuperSpeed Signaling Requirements," on page 591.

Figure 18-9: The Differential Receiver, Termination, And Equalizer



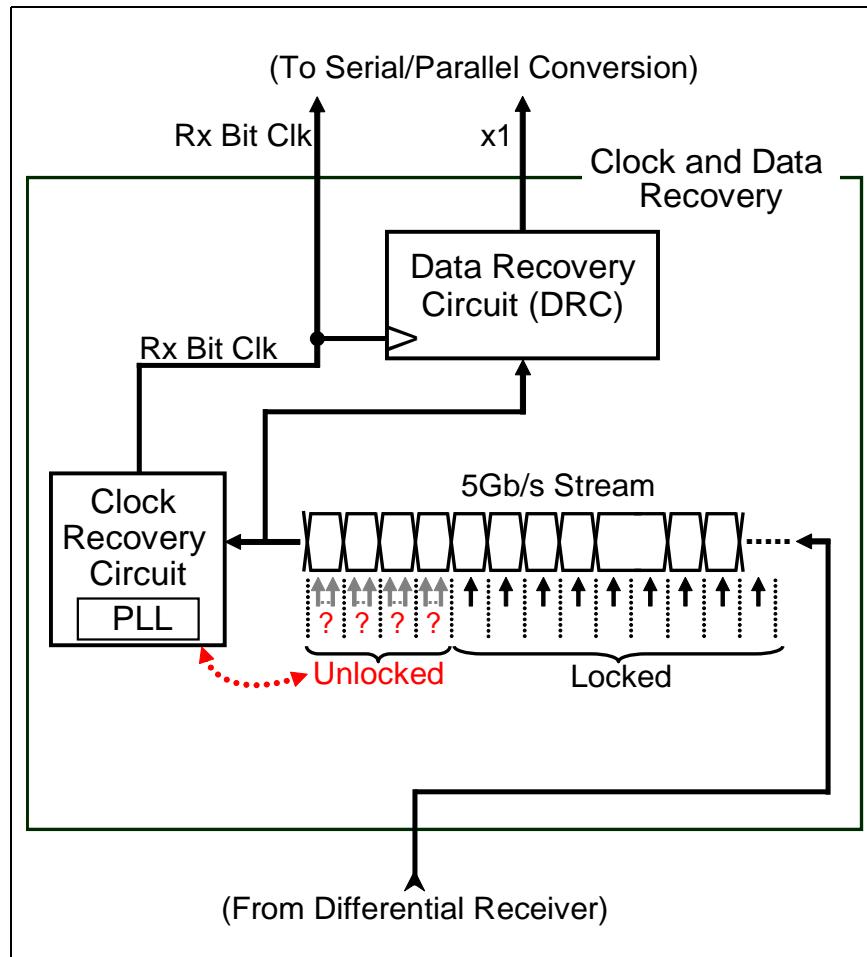
## Chapter 18: Physical Layer Logical Functions

### Clock And Data Recovery

The next stage of the receiver PHY logic section is clock and data recovery. As may be seen in Figure 18-10 on page 397, and as described in the following sections, this part of the receiver PHY logic relies on two mechanisms working together:

- Clock recovery circuit
- Data recovery circuit (DRC)

Figure 18-10: Receiver PHY Clock And Data Recovery Logic



# **USB 3.0 Technology**

---

## **Clock Recovery**

Once equalization is set up during link training to guarantee usable signal levels at the differential receiver, clock recovery extracts and maintains a copy of the transmitter clock by detecting transitions in the incoming serial bit stream and locking its receive clock to it. This process is depicted conceptually in Figure 18-10 on page 397. During full link training, clock recovery is done using TSEQ ordered sets. During LTSSM Recovery, TS1 ordered sets are used to re-initialize clock recovery logic during exits from U1, U2, and U3 power management states, etc.

## **Data Recovery Circuit (DRC)**

Using the recovered clock, the receiver samples the incoming serial bit stream data. As shown in Figure 18-10 on page 397, the receiver positions the recovered clock in the center of each bit time to help compensate for the expected clock jitter--defined as the difference between the ideal and actual clock position. At this point, the receiver has achieved *bit lock*. The recovered serial data is forwarded, one bit at a time, to the receiver's Serial To Parallel conversion logic.

---

## **Serial To Parallel Data Conversion**

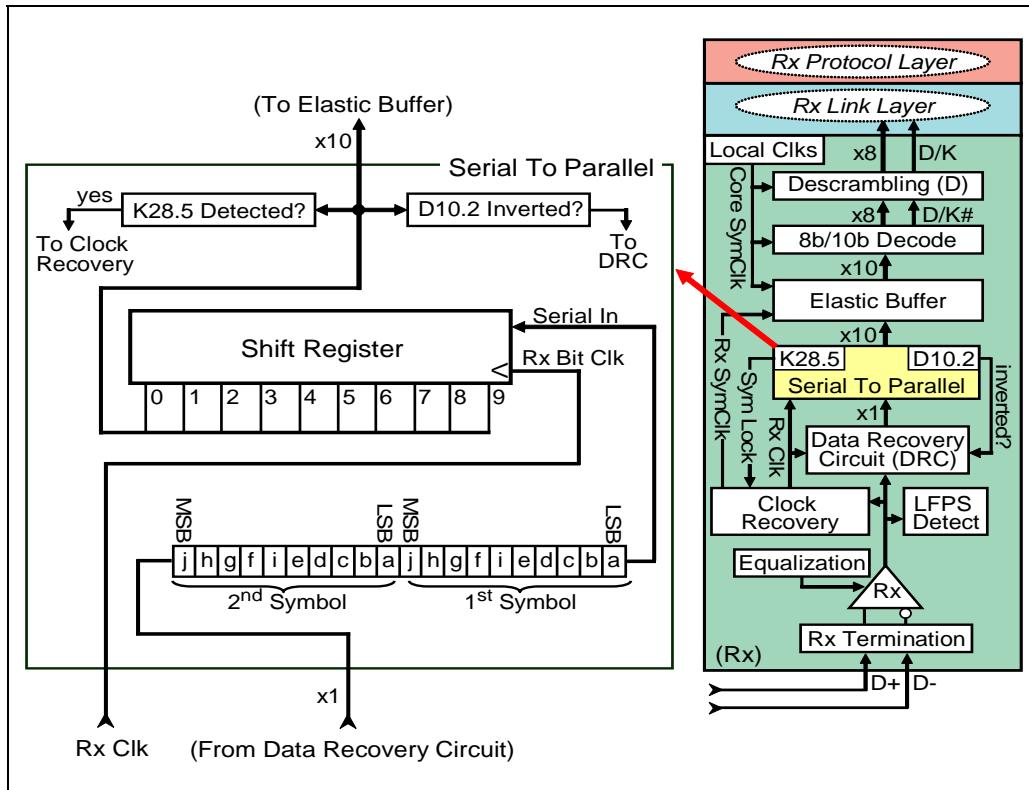
Figure 18-11 on page 399 depicts the receiver PHY Serial To Parallel data conversion logic. Aside from de-serialization of 5 Gb/s data arriving from the link, two other tasks are associated with this logic:

- COM symbol detection. This symbol is used by the receiver PHY to achieve symbol lock.
- D10.2 symbol detection. This symbol is used by the receiver to check for differential polarity inversion.

These roles are described in the following section.

# Chapter 18: Physical Layer Logical Functions

Figure 18-11: Receiver PHY Serial To Parallel Data Conversion



## Serial To Parallel Conversion and COM Symbol Lock

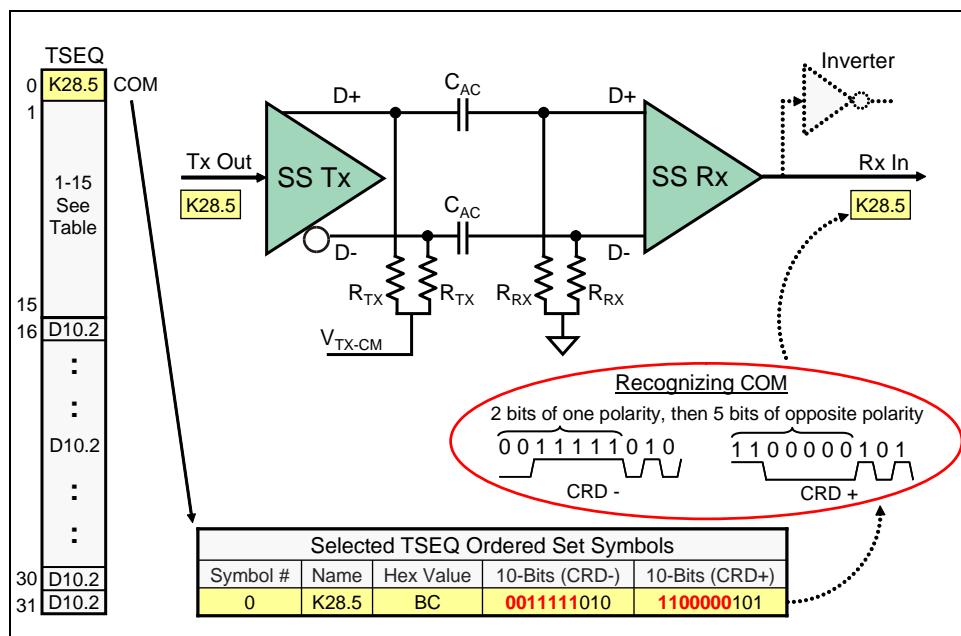
Once the data recovery circuit (DRC) is correctly sampling each inbound bit with its recovered clock, the next step for the receiver PHY is to determine the boundary between each 10-bit symbol. During link training, the first symbol in each of the 65,536 TSEQ ordered sets is the COM symbol (K28.5). When 10 bits from DRC are shifted into the Serial To Parallel shift register, a check is made to see if the 10 bit value is K28.5 (COM). This is called COM detect; there are two possibilities:

- A COM is not detected. In this case, another bit is shifted in and the oldest bit is dropped; the COM detect is attempted again (until successful)
- A COM is detected (the 10-bit value is K28.5). Once COM detect is successful (referred to as *symbol lock*), each 10 bits entering the Serial To Parallel logic is a new symbol and is forwarded to the Elastic Buffer.

## USB 3.0 Technology

Note that during LTSSM Recovery (for example, on an exit from link state U1-U3), there are no TSEQ ordered sets exchanged by link partners. Instead, TS1 order set COM symbols are used to achieve symbol lock using the same scheme as just described for TSEQ. The case of COM detection using TSEQ ordered sets is shown in Figure 18-12 on page 400.

Figure 18-12: COM Is Used To Achieve Symbol Lock



### Differential Polarity Inversion (If Needed)

Each receiver is required to check during link training for differential signal polarity inversion and correct the problem if detected. Each differential transmitter (Tx) employs two output pins, referred to as transmitter D+ and D-. Similarly, each differential receiver (Rx) has two corresponding input pins, receiver D+ and D-. The expectation is that for each pair of link partners:

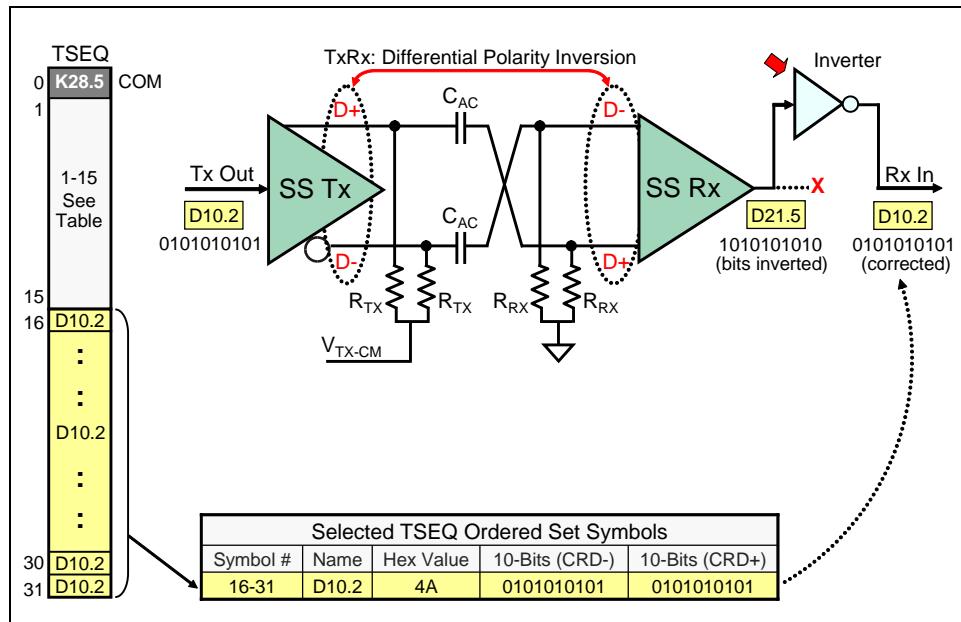
- Transmitter D+ is connected to receiver D+
- Transmitter D- is connected to receiver D-

During link training, the TSEQ ordered set D10.2 symbol is used by the receiver to determine if polarity inversion has occurred. If D10.2 symbols are received as D21.5, polarity is inverted and the receiver activates an internal inverter to correct it. This case is depicted in Figure 18-13 on page 401.

## Chapter 18: Physical Layer Logical Functions

Note that during LTSSM Recovery, no TSEQ ordered sets are exchanged so the D10.2 symbol in TS1 Ordered sets are used instead to check for differential polarity inversion. If inversion is detected, the receiver activates its internal inverter to correct it.

Figure 18-13: Receiver PHY Corrects Differential Polarity Inversion



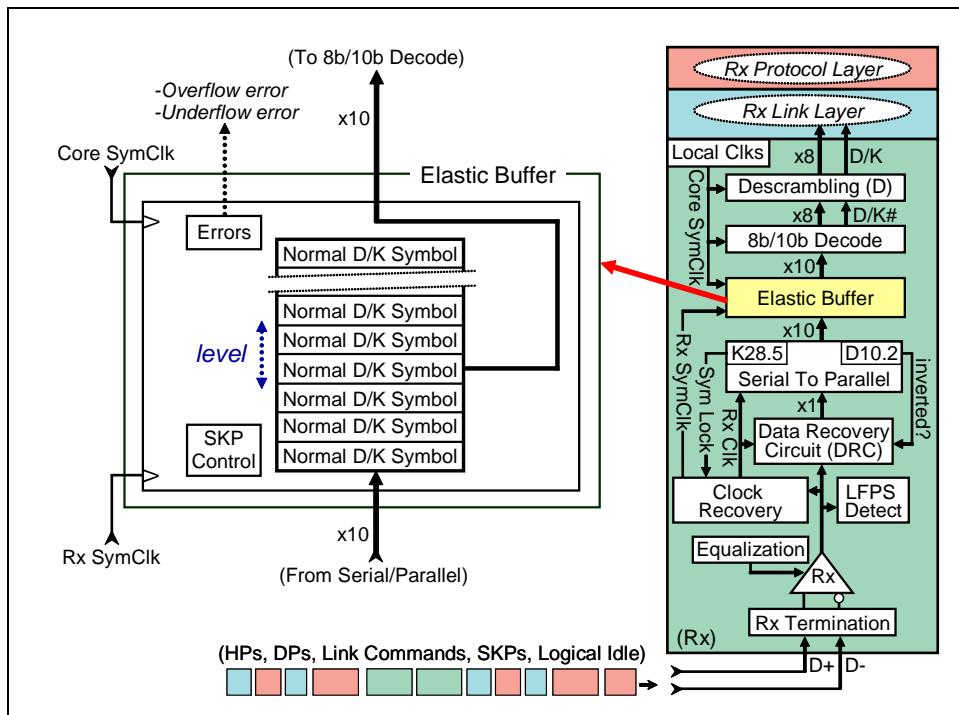
### Elastic Buffer

On links in the U0 state, a continuous stream of 10-bit symbol data is sent from transmitter to receiver. This section describes potential problems arising from the slight difference between the transmitter/receiver clock rates and the role of the elastic buffer in compensating for them. The elastic buffer (also referred to as elasticity buffer) is depicted in Figure 18-14 on page 402.

## USB 3.0 Technology

---

Figure 18-14: Receiver PHY Elastic Buffer



Key topics in this section include:

- Transmitter (Tx) and receiver (Rx) SuperSpeed clock requirements.
- The two clock domains managed by the receiver physical layer (PHY). One domain is based on the recovered Tx clock and the other domain is based on the local clock.
- Receiver use of the elastic buffer and SKP ordered sets in managing the hazards arising from the different rates in the two domains: data overflow and underflow.

# Chapter 18: Physical Layer Logical Functions

## Tx And Rx Clock Requirements

All generations of USB support independent local clocks in each device. Two factors impact the clock rate differences between link partners:

**5 GHz Base Clock.** When generating internal (local) clocks, accuracy must be within +/- 300 ppm (parts per million) of the SuperSpeed 5 GHz base clock rate. Assuming worst case, there could be a difference in link partner local clock frequencies of 600 ppm.

**Spread Spectrum Clocking.** In addition to the small difference in base clock rate attributed to clock tolerance, SuperSpeed spread spectrum clocking (SSC) requires each transmitter to slowly modulate the rate at which serial data is sent to the link partner. Modulation is only in the downward direction from its base clock rate. The modulation rate and frequency range for SSC are shown in Table 18-3 on page 403.

Table 18-3: Spread Spectrum Clocking (SSC) Requirements

Parameter	Description	Limits		
		Min	Max	Notes
t <sub>SSC-MOD-RATE</sub>	Modulation Rate	30kHz	33kHz	
t <sub>SSC-FREQ-DEVIATION</sub>	Modulation Rate	+0/-4000 ppm	+0/-5000 ppm	1
Notes		1. Data rate modulated from 0 ppm to -5000 ppm of nominal bit rate (5GT/s, etc.)		

## Implications Of Tx, Rx Clock Difference

Looking at the two extremes of possible clock rate mismatch reveals that:

- If Tx clock is -300 ppm, the Rx clock is +300 ppm, and SSC rate is at the low point (-5000 ppm), total mismatch is 5600 ppm. Every 178 symbol times (1M/5600) the transmitter will fall behind by one symbol. Unless addressed, the receiver will encounter a data underflow (no symbols to process).
- If the Tx clock is +300 ppm, the Rx clock is -300 ppm, and SSC rate is at the high point (+0 ppm), total mismatch is 600 ppm. Every 1666 symbol times (1M/600) the transmitter will have sent one extra symbol. Unless addressed, the receiver will encounter a data overflow (a dropped symbol during processing).

# USB 3.0 Technology

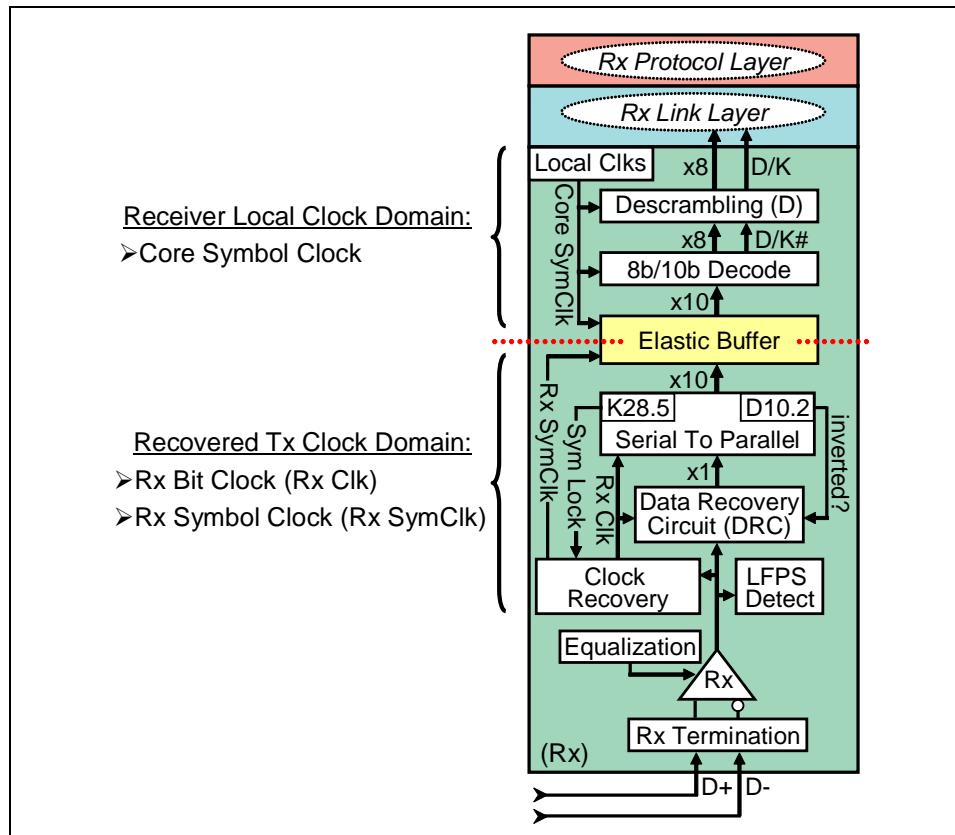
## The Result: Two Rx Clock Domains

The mismatch in transmitter and receiver clock rates requires each receiver to use the elastic buffer to manage two clock domains. The receiver will:

- Recover a clock from the transmitter's incoming data stream for the purposes of sampling each received bit at the proper time.
- Use its elastic buffer and SKP ordered sets to adjust for the rate difference between its local clock and the recovered clock from the transmitter. Because of SSC modulation, this compensation is dynamic and the level in the elastic buffer can be expected to rise and fall.

Figure 18-15 on page 404 depicts the two receiver clock domains. The notes that follow describe major elements in each clock domain.

Figure 18-15: Elastic Buffer Spans Two Receiver Clock Domains



## **Chapter 18: Physical Layer Logical Functions**

---

**Receiver PHY Clock Domain Notes.** There a few details worth noting about the receiver clock domains shown in Figure 18-15 on page 404 before looking at the elastic buffer and the use of SKP ordered sets:

- In the lower portion of the illustration, the Rx Bit Clock (Rx Clk) recovered by the receiver from the incoming bit stream sent by the transmitter can be seen. Rx Clock runs at the transmitter's rate.
- Once the receiver's clock recovery circuit achieves bit lock, Rx Clk is used by the data recovery circuit (DRC) to sample each incoming bit and clock it into the serial-to-parallel data conversion logic.
- Assuming that the COM symbol (K28.5) was detected during link training (or retraining) and symbol lock has been achieved, each ten bits shifted out in parallel from the serial-to-parallel data conversion is clocked into the elastic buffer.
- Note that the clocking from the output of serial-to-parallel data conversion logic into the elastic buffer is done using the recovered symbol clock (Rx Sym Clk in Figure 18-15 on page 404). The recovered symbol clock is also based on the transmitter's recovered clock, but operates at 1/10 of the Rx Clk rate because the data is now parallel (10 bits).
- As indicated, the clock domains change starting from the output of the elastic buffer. Now, the receiver is using its local symbol clock (Rx Sym Clk in the illustration) to move symbols through the remainder of the PHY logic to the link layer.

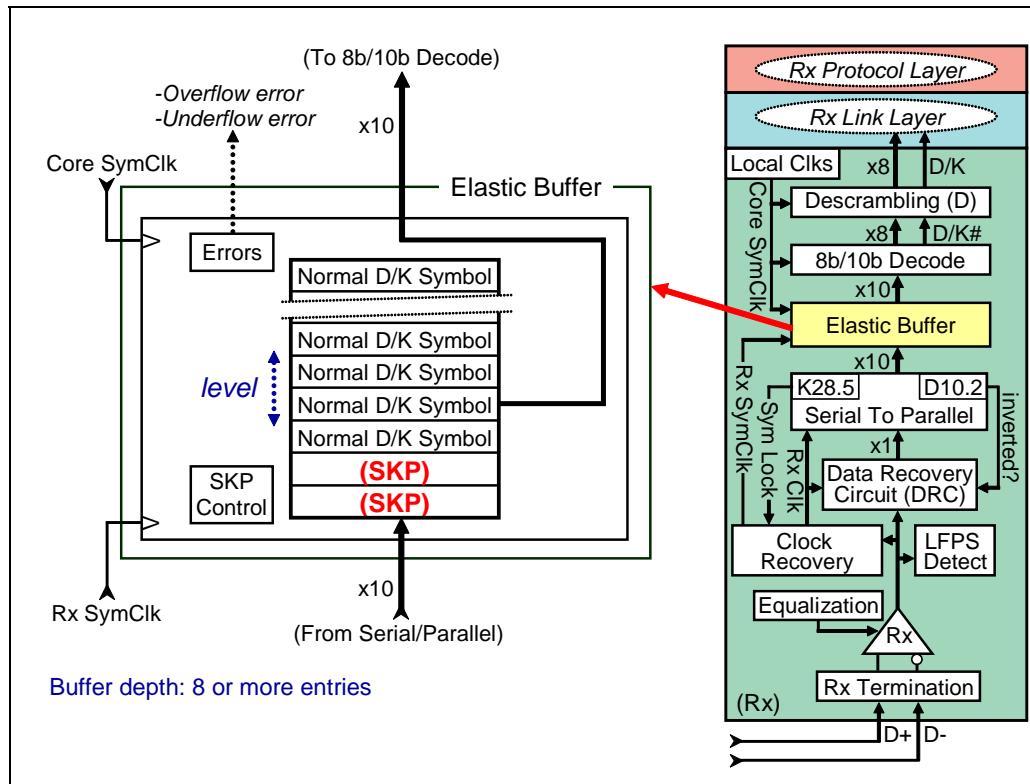
# USB 3.0 Technology

## Elastic Buffer Architecture

A conceptual drawing of the elastic buffer architecture can be seen in Figure 18-16 on page 406. In the illustration, symbols are clocked into the elastic buffer using the symbol clock recovered from the transmitter (Rx SymClk). Symbols are shifted out of the buffer using the receiver's local symbol clock (Core SymClk). The USB 3.0 specification provides some rules, but leaves most elastic buffer hardware implementation details up to the designers, including:

- **Elastic buffer depth** — must be capable of buffering at least 8 symbols
- **Skip (SKP) insertion** — depending on the elastic buffer design, the receiver may insert SKP ordered set pairs as needed to manage low levels in the elastic buffer.
- **Clock Gating** — an alternative to inserting SKP ordered sets to manage low buffer levels is the use of clock gating. This technique disables clocking symbols out of the elastic buffer anytime it is empty.

Figure 18-16: Clocking Symbols Into And Out Of The Elastic Buffer



# Chapter 18: Physical Layer Logical Functions

## Skip (SKP) Ordered Sets

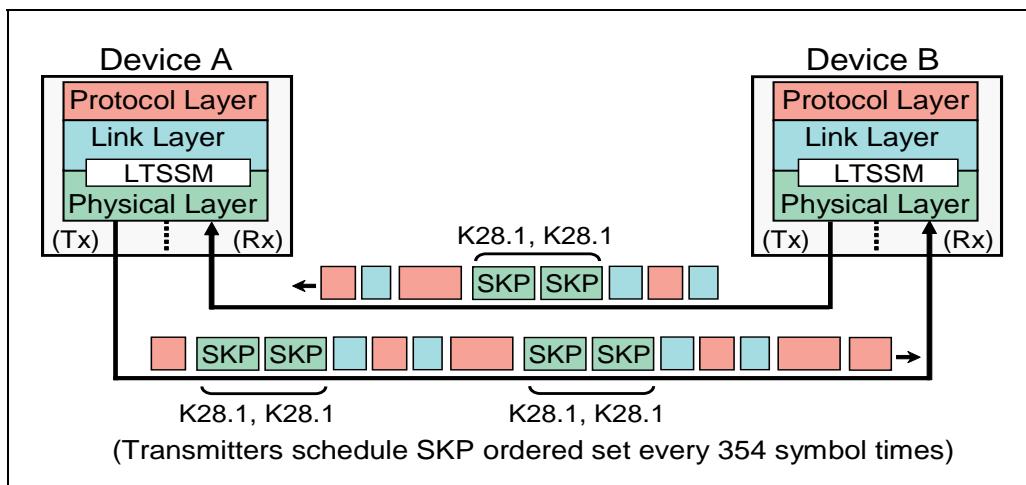
As just described, the elastic buffer is used to compensate for the slight mismatch between the recovered Tx clock used to receive data from the link and the receiver local clock used to transport data through 8b/10b decoding, de-scrambling, and up to the link layer. The conceptual drawing shown in Figure 18-16 on page 406 shows normal D/K symbols and a pair of SKP ordered set symbols are currently being clocked through the elastic buffer.

During periods when the receiver local clock is faster than the transmitter's clock, the receiver can manage the slowly falling elastic buffer level by inserting SKP ordered sets into the buffer or through the use of clock gating (when the buffer is empty). To handle the times when the transmitter clock may be slightly faster than the receiver's clock, the receiver must wait for the arrival of SKP ordered sets into the elastic buffer which can then be discarded (in pairs) to lower the buffer level.

To assure that the elastic buffer never overflows, each transmitter is required to schedule delivery of SKP ordered sets as shown in Figure 18-17 on page 407, using the following rules:

- SKP symbols (K28.1) are always inserted in pairs. A pair is scheduled every 354 symbol times, but are never inserted within packets, link commands, or other ordered sets.
- Transmitter may buffer (accumulate) up to four pairs of skip (SKP) ordered sets while waiting for an opportunity to send them across the link.

Figure 18-17: Transmitter Inserts Skip (SKP) Ordered Sets



# USB 3.0 Technology

## 8b/10b Decode

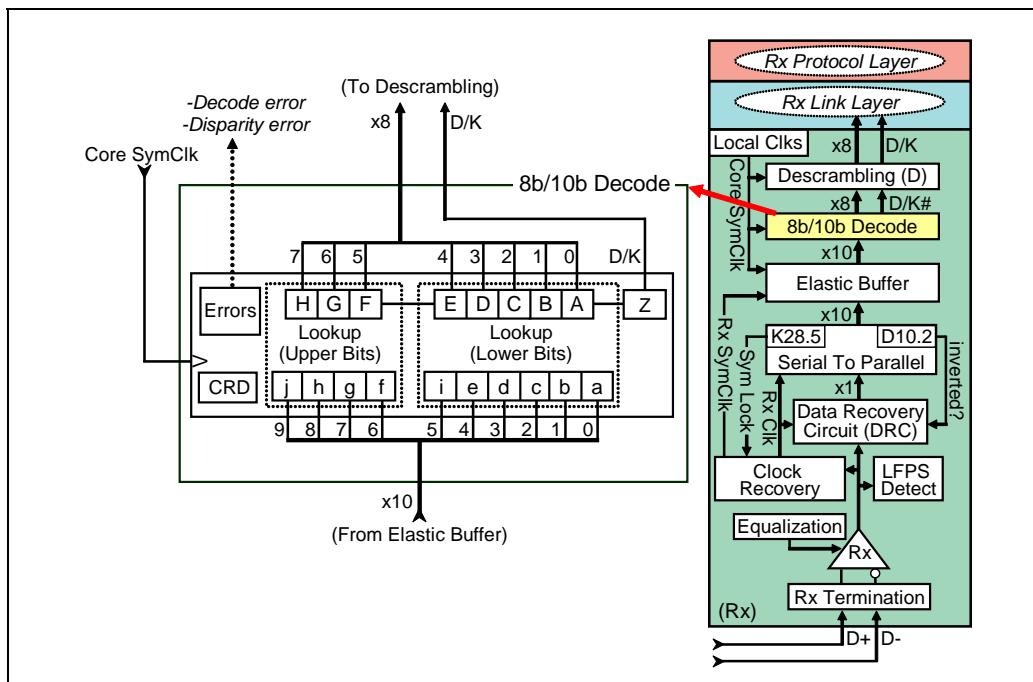
As indicated in Figure 18-18 on page 408, symbols are clocked out of the elastic buffer and into the 8b/10b decode logic using the receiver's local symbol clock (Core Sym Clk in the illustration). For each 10-bit symbol into the decoder, one byte and a D/K flag bit is forwarded to the de-scrambling logic.

### Error Checking At The Decoder

Other responsibilities of the receiver PHY 8b/10b decode logic include checking inbound symbols for errors caused by corrupted bits on the link:

- Symbol disparity is checked against CRD to ensure that DC-balance is maintained on the link. In the event of a CRD error, the physical layer informs the link layer.
- Symbol value is compared with for valid data (D) or control (K) encoding. In the case of an illegal 10-bit symbol, the 8b/10b decode logic inserts the SUB symbol (K28.4) in its place before forwarding it upstream.

Figure 18-18: Receiver PHY 8b10b Decode

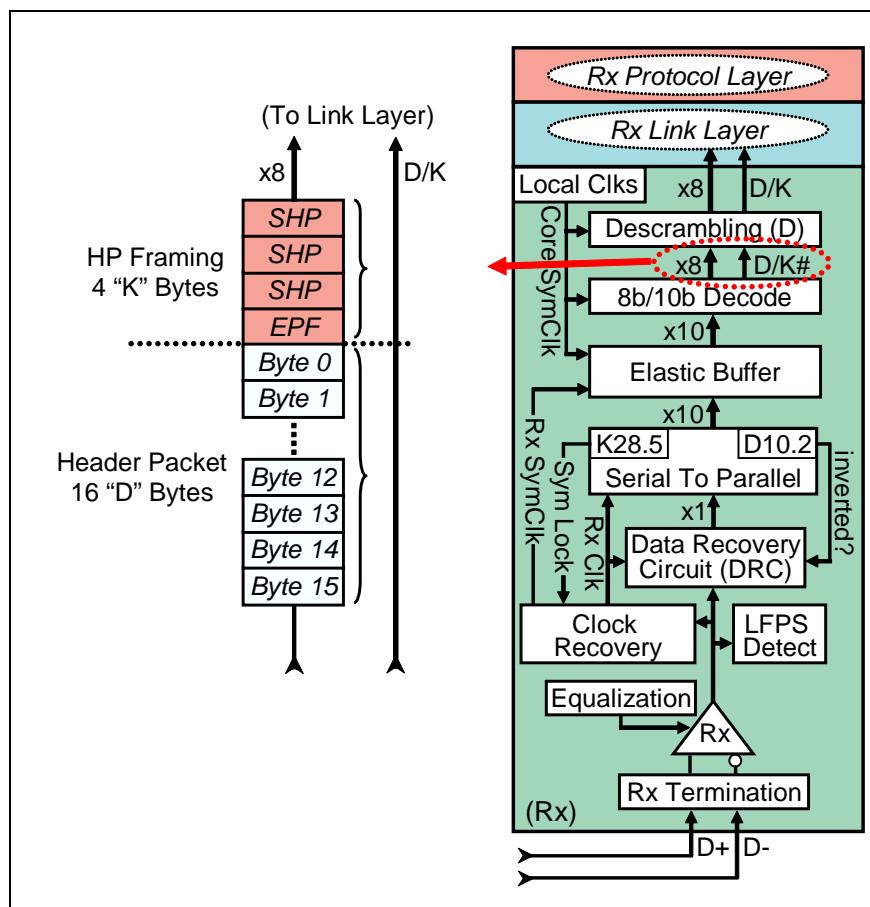


# Chapter 18: Physical Layer Logical Functions

## Decoded Data (D) And Control (K) Bytes

Figure 18-19 on page 409 illustrates the use of the D/K flag as each byte is forwarded to the de-scrambling logic. In the example shown, it is a header packet being processed; the four bytes that comprise the HPSTART framing are identified as control (K), and the 16 bytes that comprise the header are identified as data (D). While not shown in Figure 18-19, the same handling of bytes and the D/K flag would be applied to inbound data packet headers (DPH), data packet payload (DPP), and Link Commands.

Figure 18-19: Receiver PHY Decoder Use Of The D/K Flag



## USB 3.0 Technology

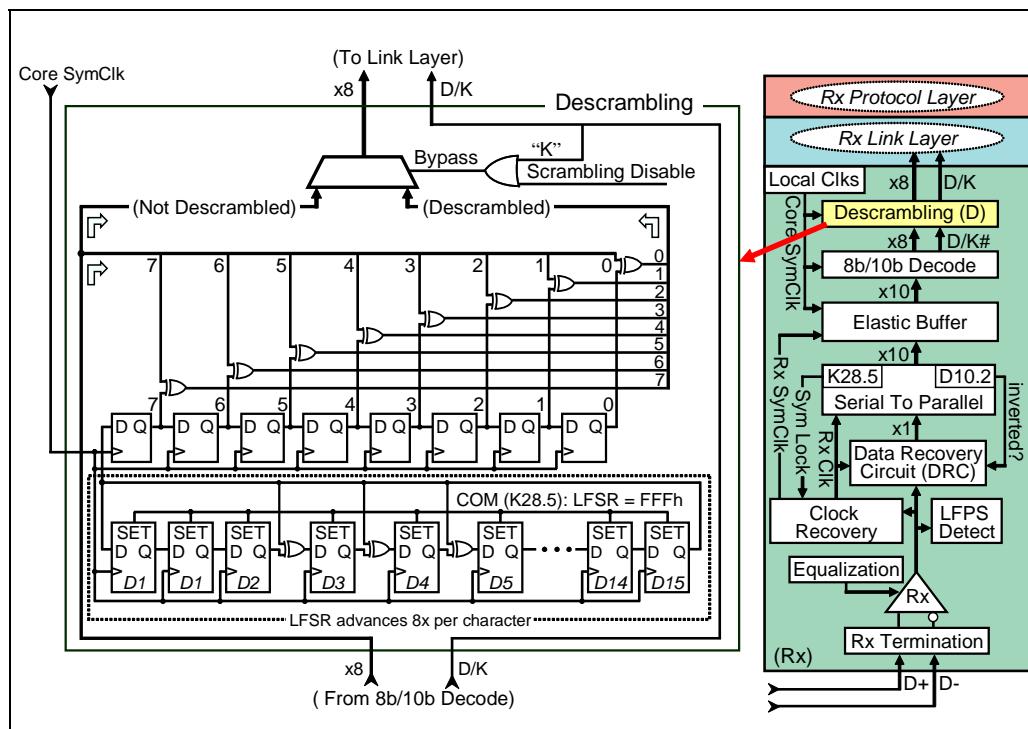
## **Descrambling Of Data (D) Bytes**

In this step, the Rx PHY logic de-scrambles inbound data (D) bytes to recover logical values for each outbound data (D) byte scrambled by the transmitter to reduce EMI. Data (D) bytes to be de-scrambled don't include framing, but do include:

- All bytes in a 16-byte header
  - All bytes in a data packet payload, including CRC-32
  - Both bytes in the link command Link Command Word field

As can be seen in Figure 18-20 on page 410, the scrambler is the same free running LFSR (Linear Feedback Shift Register) hardware employed by the transmitter. In the illustration, the bypass path for control (K) symbols is also shown. Note that each time a COM byte is recognized, the scrambler LFSR is reset to FFFFh.

*Figure 18-20: Receiver PHY Descrambling Hardware Details*

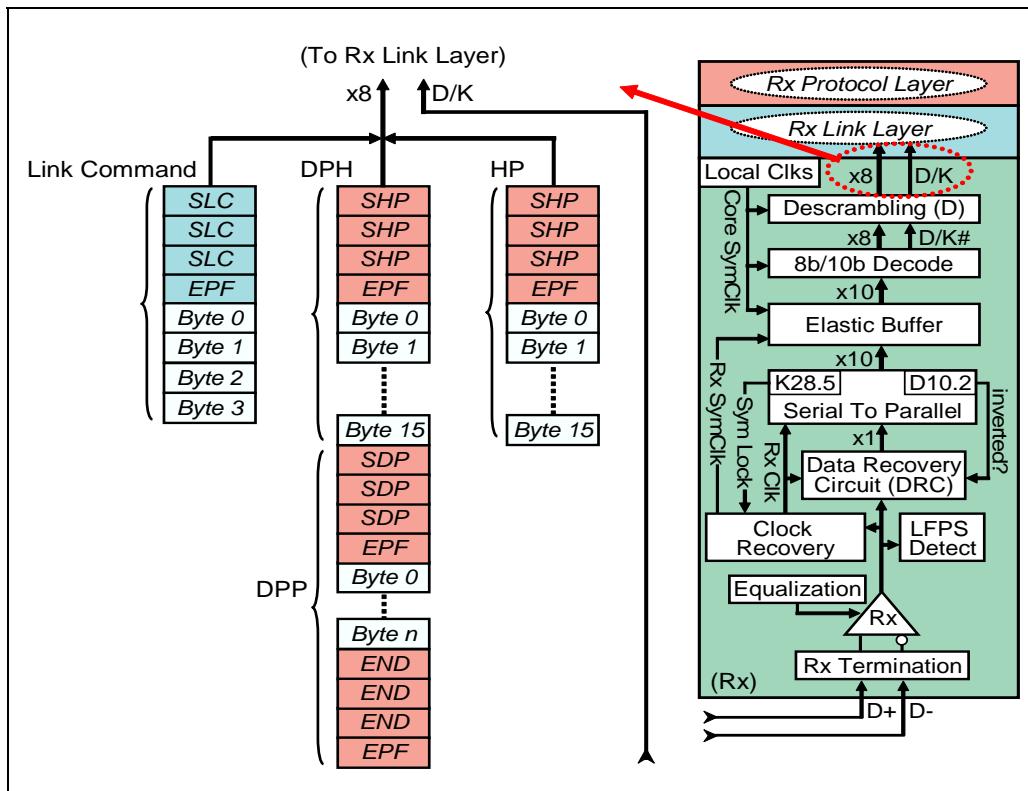


# Chapter 18: Physical Layer Logical Functions

## Forwarding Traffic To Link Layer

The final step in Rx PHY logic processing is shown Figure 18-21 on page 411.

Figure 18-21: Receiver PHY Forwards Traffic To Link Layer



Inbound traffic from the link processed by the receiver PHY is passed to the link layer for CRC and other checks. Each byte is accompanied by the D/K flag bit indicating whether it is a data (D) byte or a control (K) byte. As can be seen in Figure 18-21 on page 411, the inbound traffic consists of:

- Framed Header Packets. These include sixteen bytes of header information preceded by the four control byte HPSTART framing.

## **USB 3.0 Technology**

---

- **Framed Data Packets.** The data packet header (DPH) is sixteen bytes and preceded by the four control byte HPSTART framing. The data packet payload (DPP) immediately follows the DPH and is preceded by the four control byte DPPSTART framing and framed at the back by the four control byte DPPEND framing.
- **Framed Link Commands.** These consist of 4 bytes of Link Command Word information (actually two Link Command Words, repeated twice) preceded by the four control byte LCSTART framing.

---

---

# 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<sub>BUS</sub> 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<sub>BUS</sub> transition to a valid voltage level (a minimum of 4.0 V). Upstream facing ports monitor V<sub>BUS</sub> continuously and are required to perform an internal PowerOn Reset each time V<sub>BUS</sub> transitions from invalid to valid--even if the device is self powered.

# USB 3.0 Technology

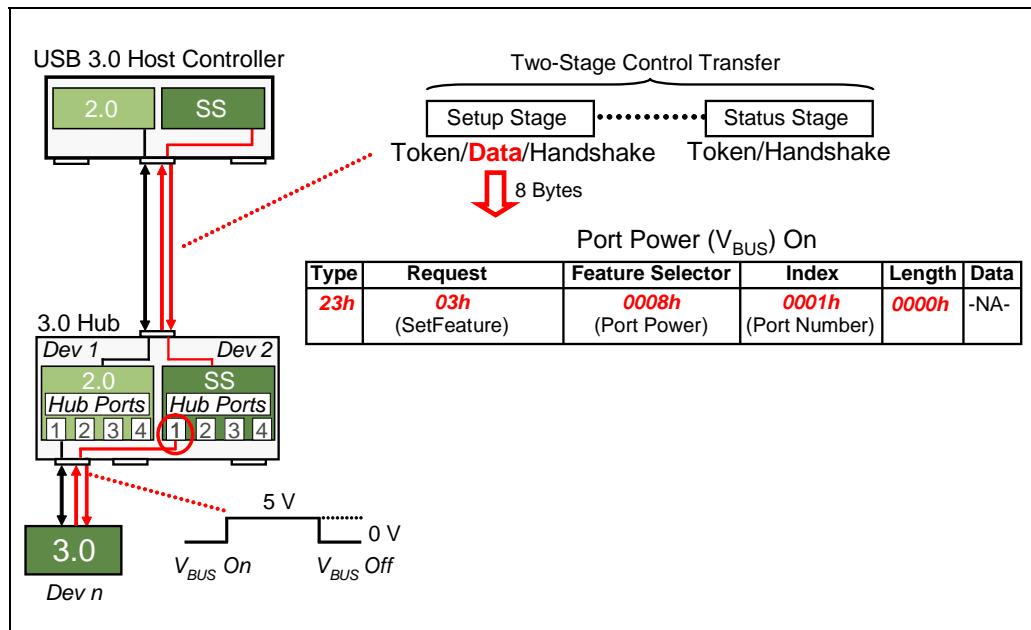
## Software May Enable And Disable V<sub>BUS</sub>

In addition to the PowerOn Reset that occurs automatically each time a device detects a V<sub>BUS</sub> transition to a valid voltage, software may selectively disable or enable V<sub>BUS</sub> 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<sub>BUS</sub>.

As shown in the example, V<sub>BUS</sub> 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<sub>BUS</sub> Triggers PowerOn Reset



## Chapter 19: SuperSpeed Reset Events

---

### 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,  $Z_{RX-HIGH-IMP-DC-POS}$ , must be 25 K $\Omega$ s or more.

# USB 3.0 Technology

- USB 3.0 SuperSpeed transmitters maintain constant DC common mode voltage:  $V_{TX-DC-CM} = 0-2.2V$ .

When PowerOn Reset completes and  $V_{BUS}$  is again valid, these rules apply:

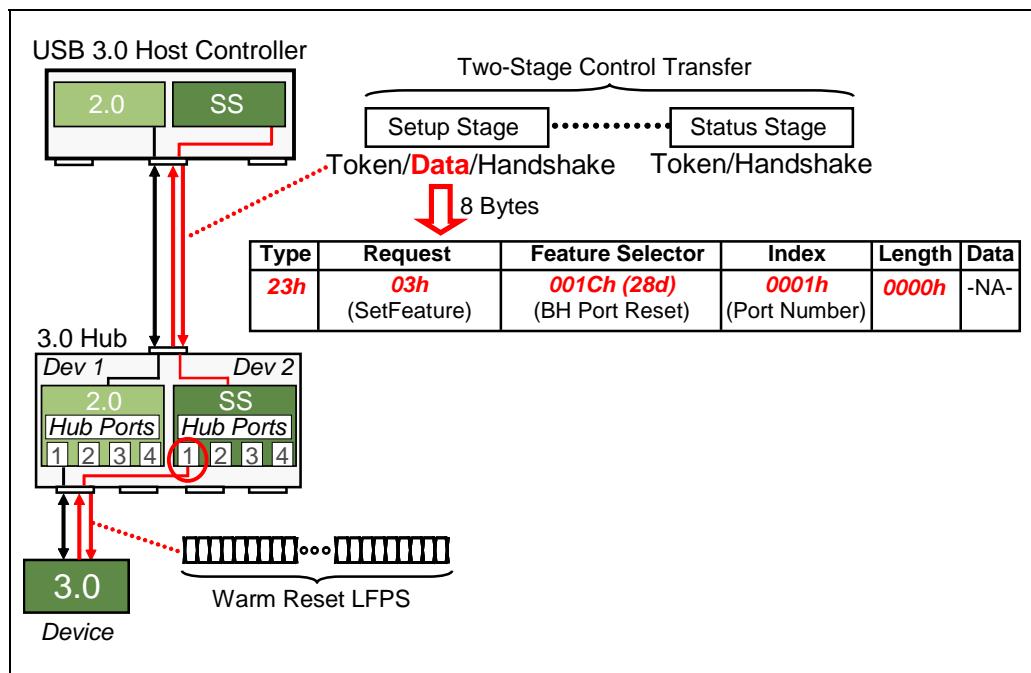
- Device transitions LTSSM state to Rx.Detect
- Timers, counters, and other LTSSM variables are restored to initial values
- Receiver equalization parameters are also restored to initial values
- Low impedance receiver termination is active:  $R_{RX-DC} = 18-30 \Omega$ s.

## Warm Reset

Warm Reset is one of the two inband reset types and is used to re-initialize device state, start link training, and prompt a return to link state U0. This is a fundamental reset and has similar effects to PowerOn Reset, but does not require a cycling of  $V_{BUS}$  power.

As shown in Figure 19-2, Warm Reset occurs under software control; the required LFPS signaling is initiated by downstream facing hub ports. The downstream facing port may signal, and the upstream facing port must detect, a Warm Reset while in any LTSSM state except SS.Disabled.

Figure 19-2: 3.0 Hub Receives A Request To Perform Warm Reset



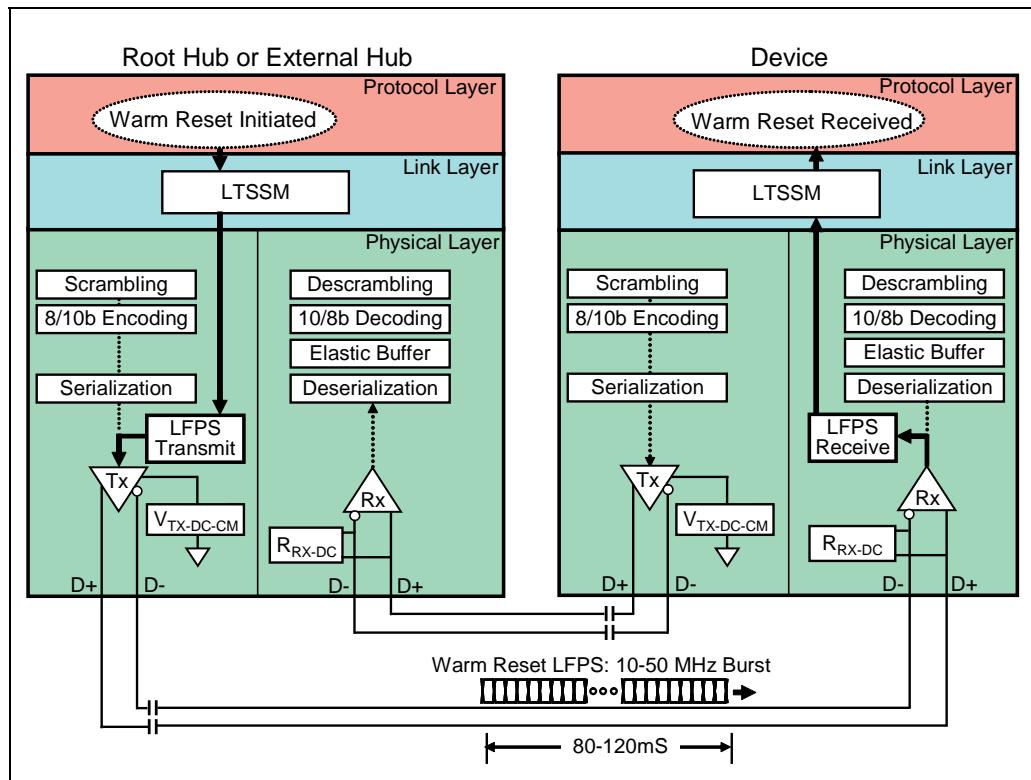
# Chapter 19: SuperSpeed Reset Events

## How Warm Reset Is Signaled On The Link

As shown in Figure 19-3 on page 417, Warm Reset is signaled when the downstream facing port drives a single low frequency periodic signaling (LFPS) burst that meets Warm Reset timing requirements:  $t_{Reset} = 80\text{-}120\text{mS}$ .

Note: A Warm Reset may be directed by software anytime the link is in any state, other than SS.Disabled. One of the implications of this is that upstream facing ports are generally always enabled to detect LFPS signaling by their hub port link partners.

Figure 19-3: Warm Reset LFPS Signaling Sent And Received



# **USB 3.0 Technology**

---

## **After Warm Reset Completes**

At the completion of Warm Reset, the following rules apply:

Detection of Warm 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 Warm Reset signaling ends, 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.

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

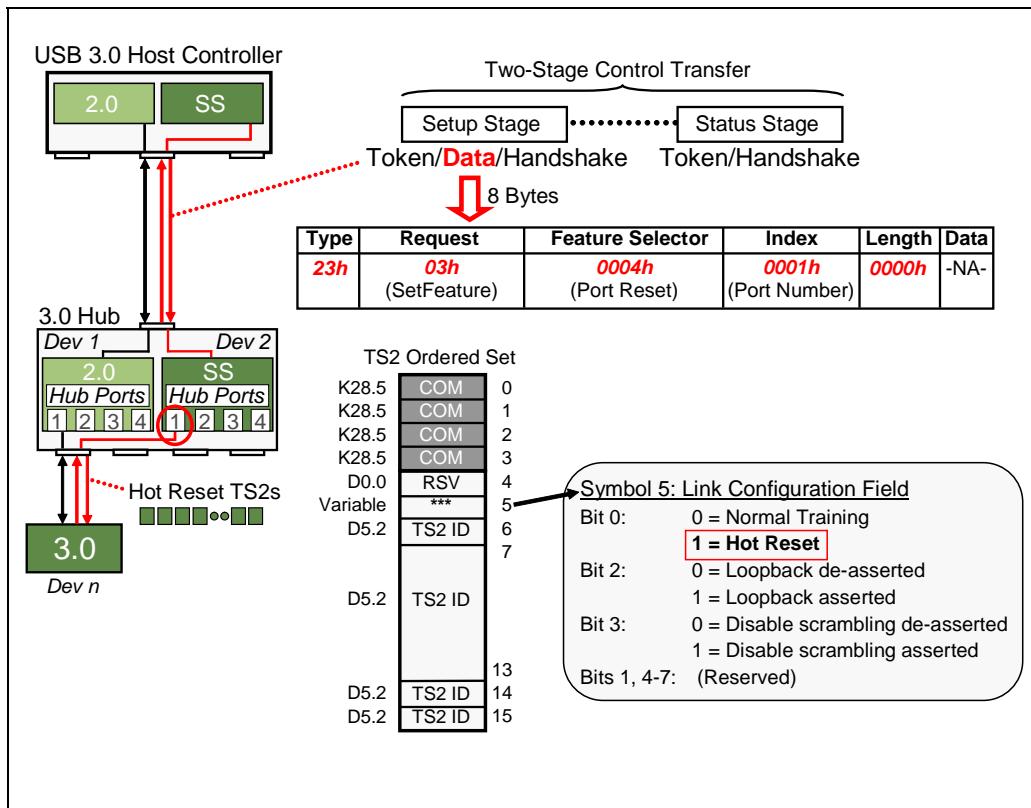
---

## **Hot Reset**

Hot Reset is the second in-band reset type. As shown in Figure 19-4 on page 419, Hot Reset occurs under software control and is initiated by downstream facing hub ports. Because Hot Reset employs SuperSpeed TS2 ordered sets, it may only be used on a functioning SuperSpeed interface in link state U0 or in the Polling.Idle substate (at the end of link training). If a link is operational, but in the power management states U1, U2, or U3, an exit from electrical idle occurs before the Hot Reset appears on the link.

## Chapter 19: SuperSpeed Reset Events

Figure 19-4: 3.0 Hub Receives A Request To Perform Hot Reset



### How Hot Reset Is Signaled On The Link

As depicted in Figure 19-4 on page 419, Hot Reset is signaled when the downstream facing port issues TS2 ordered sets with the “Hot Reset” bit set. The general sequence is:

- Downstream facing port transmits TS2's with Reset bit set = 1 continuously until the other device responds.
- Upstream facing port returns TS2's with Reset Bit also set = 1 until it has completed internal reset operations. Upstream facing port then sends TS2's with Reset bit cleared (=0) indicating it is ready to exit.
- Downstream facing port completes handshake by returning TS2's with Reset bit cleared.
- Devices exchange logical idle symbols and prepare to return to the U0 link state.

# USB 3.0 Technology

---

## After Hot Reset Completes

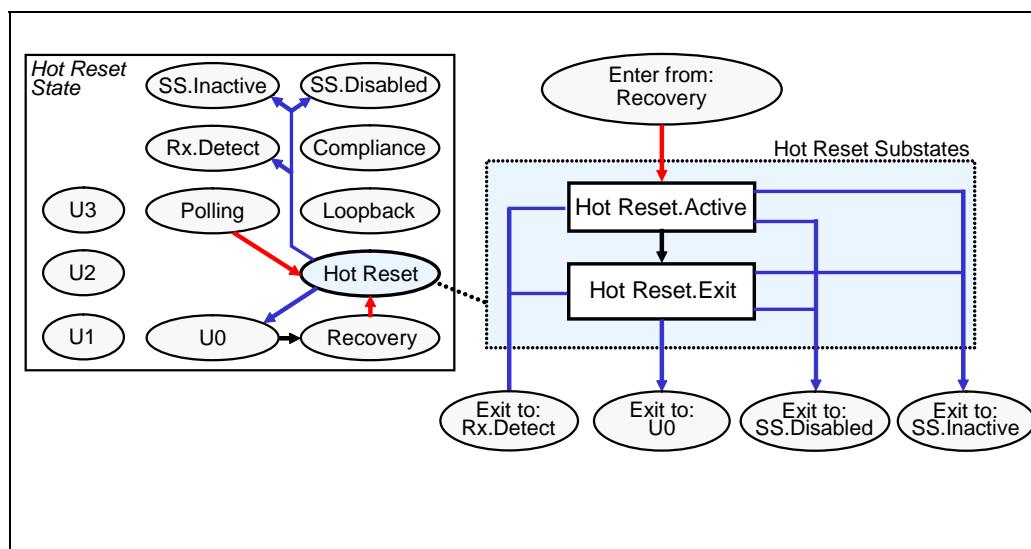
At the completion of Hot Reset, the following rules apply:

- The Link Error Count (LEC) for the downstream facing port is cleared.
- Saved port configuration values for upstream facing ports are reapplied. These include: number of header packet buffers (four in the current specification) and link speed (5 GT/s in the current specification). reusing these saved parameters eliminates the need to exchange Port Capability and Configuration Link Management Packets on U0 re-entry after a Hot Reset.
- Receiver equalization parameters are preserved.
- Link layer Tx and Rx header packet buffers, counters, timers are cleared
- USB specified registers and other storage elements are at default values.
- USB device address returns to 0
- Receivers maintain their low impedance termination ( $R_{RX-DC}$ )
- Devices normally return to the U0 link state, but transitions to Rx.Detect, SS.Inactive, or SS.Disabled are also possible if Hot Reset does not complete successfully.

## Hot Reset, The LTSSM View

Figure 19-5 on page 420 depicts the LTSSM Hot Reset state as well as the possible transitions into and out of the state. There are two substates in Hot Reset.

Figure 19-5: LTSSM Hot Reset State And Substates



# **Chapter 19: SuperSpeed Reset Events**

---

## **General Description**

Hot Reset provides a low latency reset method for links that have previously been through full link training. Hot Reset avoids the lengthy Warm Reset and transition to Rx.Detect for full link training by assuming that device context and physical layer parameters established earlier are still valid. If Hot Reset is successful, link partners perform a handshake using TS2 ordered sets, initialize internal logic, and quickly return to U0. Hot Reset is always initiated under software control by a downstream facing port.

During the Hot Reset, all ports will maintain normal transmitter and receiver electrical specifications, including the low impedance receiver termination ( $R_{RX-DC}$ ). Downstream facing port will also reset the Link Error Count (LEC), power management timers, and U1/U2 Inactivity timers.

## **Requirements In Hot Reset.Active**

The following requirements apply while an LTSSM is in Hot Reset.Active:

- A 12 ms timer ( $tHotResetActiveTimeout$ ) is started on entry to this substate.
- Upon entry to this substate, downstream facing port will transmit at least 16 TS2s with the Reset bit asserted.
- When Hot Reset signaling is recognized by upstream facing port, it will transition to LTSSM Hot Reset state and return TS2s with Reset bit asserted.
- The exchange of TS2s with the Reset bit asserted will continue until the upstream facing port has completed internal initialization and starts sending TS2s with the Reset bit de-asserted.
- When the downstream facing port recognizes TS2s with Reset de-asserted, it responds sends TS2s with Reset de-asserted in response. Ports transmit TS1 ordered sets on entry to this state

## **Exit Rules For Hot Reset.Active**

As depicted in Figure 19-5 on page 420, there are four possible exit states from the Hot Reset.Active substate.

Exit rules observed by both upstream (US) and downstream (DS) facing ports:

- Ports transition to Hot Reset.Exit when the following is true: 1) at least 16 TS2s with Reset asserted were transmitted, 2) at least 2 consecutive TS2s with Reset de-asserted were received, and 3) four consecutive TS2s with Reset de-asserted were sent after receiving at least one TS2 with Reset de-asserted.
- Ports transition to SS.Inactive if the 12 ms  $tHotResetActiveTimeout$  expires.

## **USB 3.0 Technology**

---

Exit rules observed by downstream (DS) facing ports:

- Downstream facing ports transition to SS.Inactive when either Ux Exit Timer or 12 ms  $tRecoveryActiveTimeout$  expire AND the transition to Recovery was not to attempt Hot Reset.
- Downstream facing ports transition to Rx.Detect when directed to issue a Warm Reset
- Downstream facing ports transition to SS.Disabled when directed.

Exit rules observed by upstream (US) facing ports:

- Upstream facing ports transition to Rx.Detect when Warm Reset is detected

### **Requirements In Hot Reset.Exit**

Following requirements apply if an LTSSM is in HotResetExit substate:

- A 2 ms timer ( $tHotResetExitTimeout$ ) is started on entry to this substate.
- Ports transmit Idle symbols
- Ports prepare to receiver Header Sequence Number advertisement.

### **Exit Rules For Hot Reset.Exit**

As depicted in Figure 19-5 on page 420, there are four possible exit states from the Hot Reset.Exit substate.

Exit rules observed by both upstream (US) and downstream (DS) facing ports:

- Ports transition to U0 when 1) eight consecutive Idle symbols are received and 2) Sixteen Idle symbols were sent after receiving at least one Idle.
- Ports transition to SS.Inactive if the 2 ms  $tHotResetExitTimeout$  expires.

Exit rules observed by downstream (DS) facing ports:

- Downstream facing ports transition to Rx.Detect when directed to issue a Warm Reset
- Downstream facing ports transition to SS.Disabled when directed.

Exit rules observed by upstream (US) facing ports:

- Upstream facing ports transition to Rx.Detect when Warm Reset is detected

## **Chapter 19: SuperSpeed Reset Events**

---

---

### **Notes About Warm Reset And Hot Reset Latency**

---

#### **Warm Reset Is Time Consuming, But Thorough**

Warm Reset signaling is fairly lengthy process as it requires an 80-120mS burst of LFPS so that it may be clearly distinguished from other types of LFPS signaling events (e.g., exit from U1, U2, U3 power management states). Furthermore, receiver equalization parameters must be re-established during the full link training that follows a PowerOn or Warm Reset. On the other hand, the Warm Reset re-initializes all USB state and is very effective in clearing most types of USB 3.0 SuperSpeed link error conditions.

---

#### **Hot Reset Is Much Faster, Not As Thorough**

The Hot Reset handshake employs TS2 ordered sets and may conclude in approximately 1uS if both devices are prepared to proceed (a minimum of 16 TS2 ordered sets with Hot Reset bit set are exchanged followed by a return to U0). In addition, receiver equalization parameters are preserved during Hot Resets. This results in an additional savings of approximately 4.2mS (65,536 TSEQ ordered sets are sent and received during receiver equalization setup following PowerOn Reset or Warm Resets)

---

### **How Does Software Know If Hot Reset Is Possible?**

With respect to the instantaneous state of a link, it doesn't. As described previously, there are advantages and disadvantages in using each of the inband reset types on a SuperSpeed link. Hot Reset is much faster, but the fact that it employs TS2 ordered sets means it may only be used on a functioning Super-Speed link in one of the following states: U0, U1, U2, Polling.Idle, Recovery.Idle.

So, while a Hot Reset is attractive because of its low latency, there would be a serious burden on software if the instantaneous link state had to be known in order to issue a hub request for a port Hot Reset. To mitigate this problem, the USB 3.0 Specification provides the two in-band reset hub request commands and requires the target hub port to help determine whether a Hot Reset request can be honored.

# **USB 3.0 Technology**

---

---

## **The Two In-Band Reset Hub Requests**

The two hub-class requests used by software to generate inband resets on a downstream facing USB 3.0 hub port are:

- SetFeature (BH\_PORT\_RESET).
- SetFeature (PORT\_RESET).

The usage model for each request follows.

---

### **SetFeature (BH\_PORT\_RESET)**

This request is used if software requires that a Warm Reset is to be used (regardless of current link state).

The rules the hub follows when SET\_FEATURE (BH\_PORT\_RESET) is received:

- If the link is in any state other than SS.Disabled and BH\_PORT\_RESET is directed, a Warm Reset will be issued, followed by a transition to Rx.Detect and full link training.

The SetFeature (BH\_PORT\_RESET) request and the Warm Reset signaling were shown in the example in Figure 19-2 on page 416.

---

### **SetFeature (PORT\_RESET)**

This request is used if software prefers a Hot Reset. If the link state permits it, Hot Reset will be issued. If not, a Warm Reset will be issued instead. When the SET\_FEATURE (PORT\_RESET) request is received, the hub uses the following strategy in deciding whether to issue a Warm or Hot Reset on the downstream link:

- If the link is in U0, Hot Reset is issued
- If the link is in U1 or U2, the link will use LFPS exit handshake to transition to Recovery and U0; then Hot Reset will be issued.
- If the link is in the last substate of link training Polling State (Polling.Idle) or of Recovery (Recovery.Idle), Hot Reset will be issued.
- If the link is in U3, SS.Loopback, SS.Compliance, or SS.Inactive, then Warm Reset will be issued instead of Hot Reset.
- If the link is in SS.Disabled, an inband reset on the SuperSpeed link is not allowed (neither Warm or Hot Reset will be issued).

## Chapter 19: SuperSpeed Reset Events

- If Hot Reset handshake fails during link Recovery, a Warm Reset will automatically be attempted by the hub. If the Warm Reset succeeds, normal transition to Rx.Detect and full link training will occur.

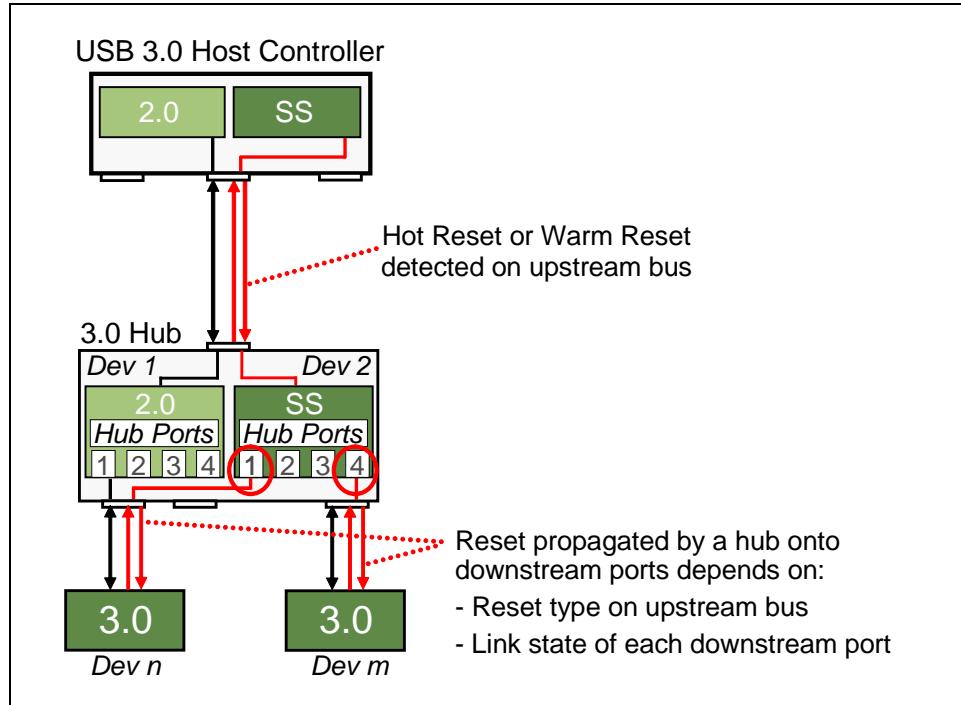
The SetFeature (PORT\_RESET) hub request and the resulting link Hot Reset signaling were shown in the example in Figure 19-4 on page 419.

### Hubs Propagate Resets Downstream

#### General

As indicated in Figure 19-6 on page 425, USB 3.0 hubs propagate resets detected on the upstream port to its downstream ports. The type of reset propagated downstream depends on whether the upstream port detected a Warm or Hot Reset as well as the current link state of the downstream port itself.

Figure 19-6: USB 3.0 Hub Propagates Resets To Downstream Ports



## **USB 3.0 Technology**

---

---

### **Hub Reset Propagation, Key Rules**

- If a SuperSpeed downstream link is in any LTSSM State other than disabled (SS.Disabled), a Warm Reset is propagated downstream as Warm Reset.
- A Hot Reset detected on the upstream port will be forwarded to downstream ports as a Hot Reset providing the downstream link has been through link training and is in LTSSM State U0, U1, U2, etc.
- Hot Reset on upstream port will be forwarded downstream as Warm Reset if the downstream port is in LTSSM state U3 (Suspend), SS.INACTIVE, SS.COMPLIANCE, or SS.LOOPBACK.
- A Hot Reset detected on the upstream port will be silently dropped at each downstream port that is not powered or in the SS.Disabled state.

---

---

# 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<sub>BUS</sub> 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

## USB 3.0 Technology

---

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.

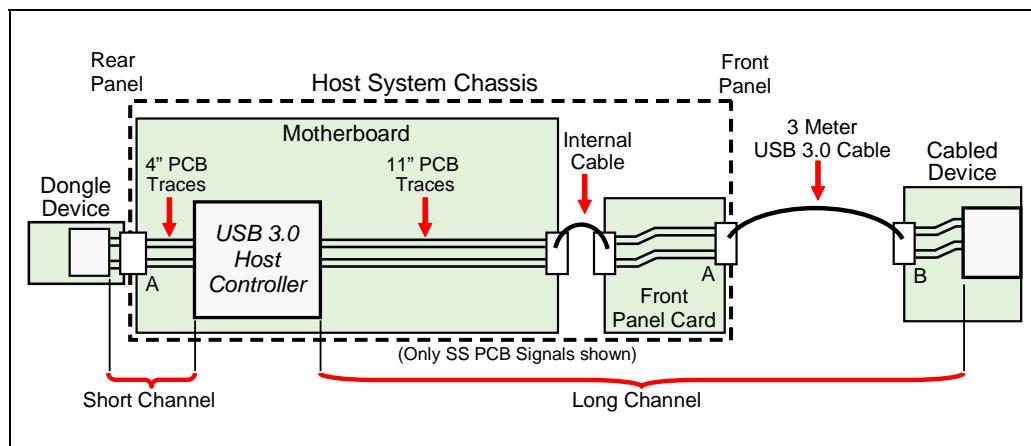
---

### 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



## **Chapter 20: Link Training**

---

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

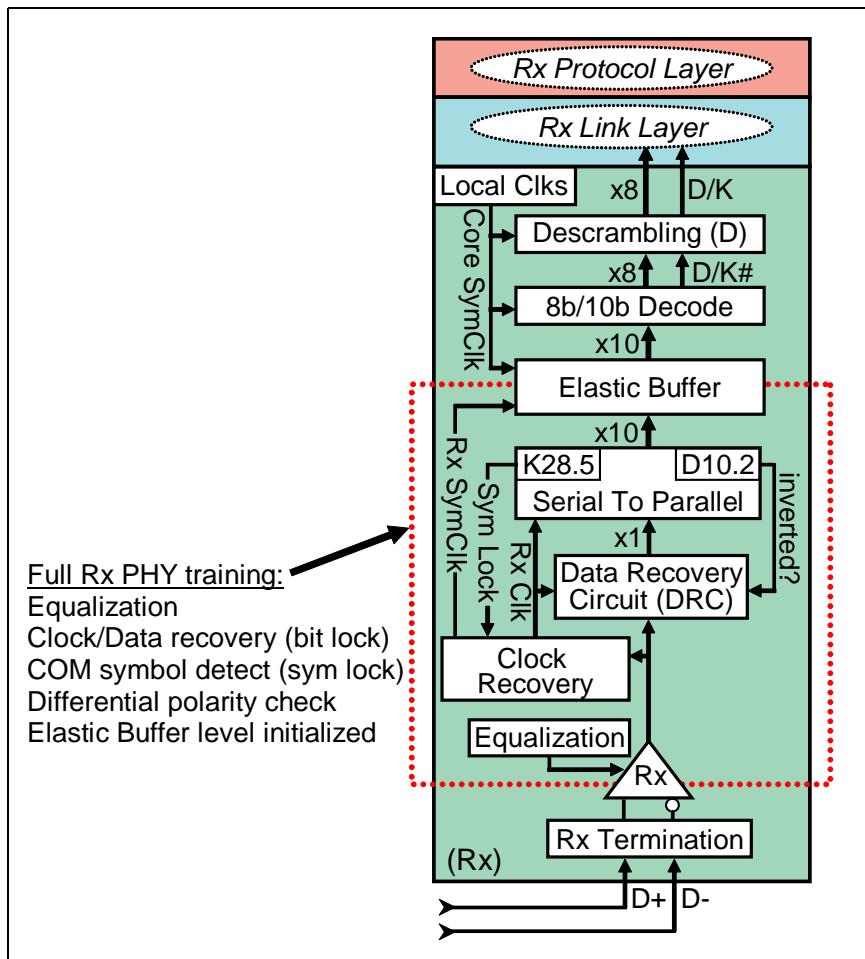
# USB 3.0 Technology

## Which Logic Requires Training?

### General

Link training places most of the burden of adapting to real-world link signal conditions on the receiver physical layer (PHY) logic in each link partner. Key elements of receiver PHY logic affected by link training are shown in Figure 20-2 on page 430.

Figure 20-2: Link Training & Receiver Physical Layer Logic



## **Chapter 20: Link Training**

---

### **Summary Of Key Link Training Elements**

In the sections that follow, link training is described in terms of the sequence of events required to transition a link from its initial state (after a Warm Reset or PowerOn Reset) to the U0 operational state where 5 Gb/s SuperSpeed packet transfers are permitted.

Much of the link training discussion is concerned with the portion of the receiver physical layer logic highlighted in Figure 20-2 on page 430. For a more comprehensive discussion of all receiver physical layer logic, refer to Chapter 18, entitled "Physical Layer Logical Functions," on page 383. A brief summary of the logic highlighted in Figure 20-2 on page 430, and directly affected by the link training process, follows. Key elements include:

- Receiver Equalization
- Clock Recovery
- Data Recovery Circuit (DRC)
- Serial To Parallel Conversion and Symbol lock
- Differential Polarity Inversion (if needed)
- Elastic Buffer initialization

#### **Receiver Equalization**

The responsibility for signal compensation on SuperSpeed links falls on both transmitters and receivers. The compensation for frequency-dependent signal loss is generally referred to as equalization. Transmitters use voltage de-emphasis on outbound bits to combat expected inter-symbol interference (ISI) effects at the receiver. At the receiver, equalization logic compensates for the sum of detrimental effects of its channel on the 5 Gb/s signal delivered from transmitter to receiver inputs.

As shown in Figure 20-2 on page 430, link equalization logic is associated with each SuperSpeed differential receiver. While the USB 3.0 specification does not impose a particular scheme to be used for receiver equalization, it does define minimum voltage and timing requirements at the receiver inputs, target bit error rate (BER) for the link, as well the wide range of physical connections over which compliance should be checked.

Initializing receiver equalization logic is one of the most critical steps in the link training scheme used in SuperSpeed USB 3.0. The scheme is based on the transmitter sending known bit patterns and the receiver applying the appropriate compensation based on signal voltage and timing detected at its inputs. The

## **USB 3.0 Technology**

---

equalization test patterns are specially selected for their varied frequency content; it is assumed that once equalizers have been trained, any pattern of bits received during normal operations will be recovered correctly by the receiver.

Setup of receiver equalization logic is accomplished during link training in two steps. Transmitters first send 10-50 MHz Polling.LFPS bursts to alert the link partner's LTSSM that training is starting and allowing receivers to set up low frequency parameters (also loosely referred to as *DC* parameters). Because of its low frequency, a valid LFPS squarewave signal should be recoverable at the receiver, regardless of the channel length. Next, the transmitter sends 5 Gb/s SuperSpeed training sequence (TSEQ) ordered sets, allowing the receiver to establish high frequency equalizer parameters.

While training of receiver equalization enables devices to adapt to the full range of channel types and signal conditions, it does have a drawback--it is slow. In order to allow a wide range of hardware equalization solutions, transmitters are required repeat the 32-symbol TSEQ ordered set 65,536 times--consuming approximately 4mS of link training time. To mitigate the amount of time this consumes, equalizers only must be trained after a Warm Reset, PowerOn Reset, or if a timeout requires the full link training to be repeated. The receiver saves equalizer parameters on each full link training and reapplies them during the lower latency link recovery and retraining mechanism employed during exits from power management states, recovery from link errors, etc.

### **Clock Recovery**

Once equalization is set up and can assure usable signal levels at the differential receiver, clock recovery attempts to extract a usable copy of the transmitter (Tx) clock by detecting transitions in the incoming serial bit stream. During full link training, clock recovery is done using TSEQ ordered sets received during equalization. Later, TS1/TS2 ordered sets are used for clock recovery during exits from U1, U2, and U3 power management states, etc.

### **Data Recovery Circuit (DRC)**

Using the recovered clock, the receiver samples the incoming serial bit stream data. Ideally, the receiver positions the recovered clock in the center of each bit time to help compensate for the expected clock jitter--defined as the difference between the ideal and actual clock position. At this point, the receiver has achieved *bit lock*.

Referring again to Figure 20-2 on page 430, the recovered serial data is forwarded, one bit at a time, to receiver's Serial To Parallel conversion logic.

# **Chapter 20: Link Training**

---

## **Serial To Parallel Conversion and Symbol Lock**

The next step for a receiver is to determine the boundary between each 10-bit symbol. During link training, the first symbol in each of the 65,536 TSEQ ordered sets is the COM symbol (K28.5). When 10 bits from DRC are shifted into the Serial To Parallel shift register, a check is made to see if the 10 bit value is K28.5 (COM). This is called COM detect; there are two possibilities:

- A COM is not detected. In this case, another bit is shifted in and the oldest bit is dropped; the COM detect is attempted again (until successful)
- A COM is detected (the 10-bit value is K28.5). Once COM detect is successful (referred to as *symbol lock*), each 10 bits entering the Serial To Parallel logic is a new symbol and is forwarded to the Elastic Buffer.

## **Differential Polarity Inversion (If Needed)**

Each receiver is required to check during link training for differential signal polarity inversion and correct the problem if detected. Each differential transmitter (Tx) employs two output pins, referred to as transmitter D+ and D-. Similarly, each differential receiver (Rx) has two corresponding input pins, receiver D+ and D-. The expectation is that for each pair of link partners:

- Transmitter D+ is connected to receiver D+
- Transmitter D- is connected to receiver D-

During link training, the TSEQ ordered set D10.2 symbol is used by the receiver to determine if polarity inversion has occurred. If D10.2 symbols are received as D21.5, polarity is inverted and the receiver activates an internal inverter to correct it. Differential polarity inversion (and correction) is covered in greater detail shortly, in the link training section on LTSSM Polling.EQ.

## **Elastic Buffer Initialization**

Finally, link training provides an opportunity to initialize the receiver's elastic buffer. Symbols are shifted into the elastic buffer using recovered Tx symbol clock, and shifted out using the receiver's local symbol clock. Depending on the design, each time SuperSpeed symbol traffic commences, the receiver may require that some number of symbols are clocked into the elastic buffer before the first symbol is clocked out and forwarded to the next stage (8b/10b decoding). Once initialized, the elastic buffer and SKP ordered sets should avoid any overflow or underflow conditions.

## USB 3.0 Technology

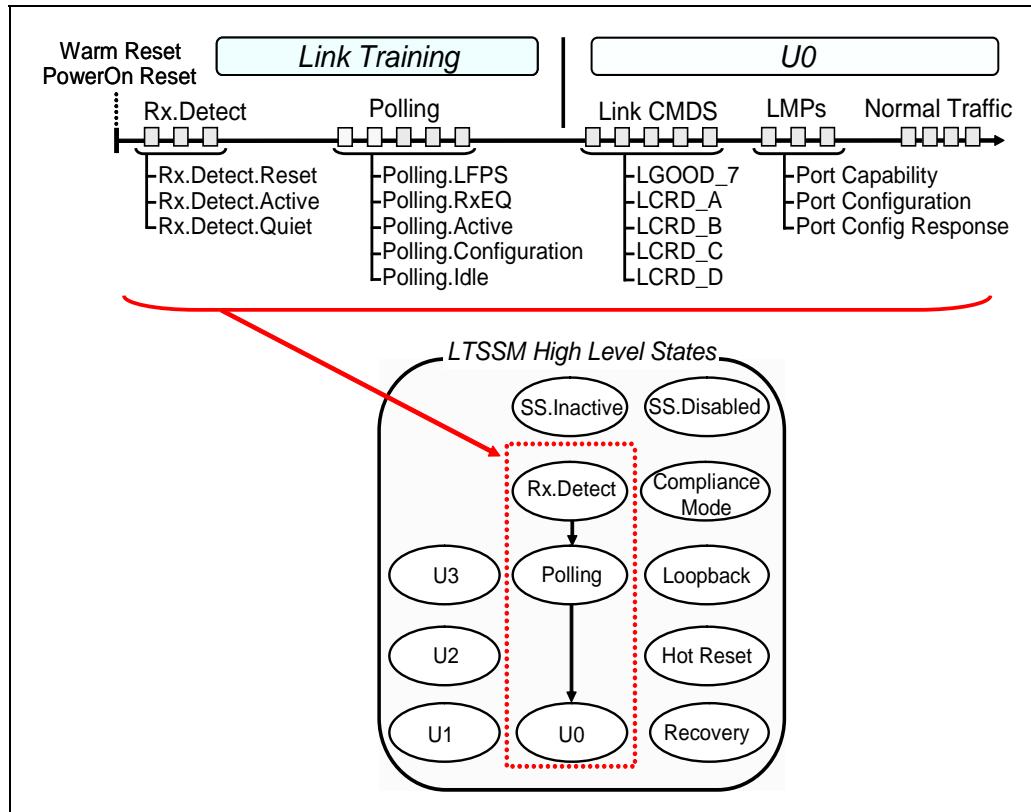
### Link Training Is Managed by LTSSM

The lower portion of Figure 20-3 on page 434 depicts the twelve high level LTSSM states. As highlighted, two LTSSM states are required during link training to transition a link to the U0 operational state:

- Rx.Detect
- Polling

At the top of Figure 20-3 on page 434, a time line is shown that summarizes the sequence of steps to be completed during link training. Note in the illustration that the two high level LTSSM states required for link training each have substates. The Rx.Detect state has three substates and the Polling state has five substates.

Figure 20-3: Link Training Sequence And LTSSM States



# Chapter 20: Link Training

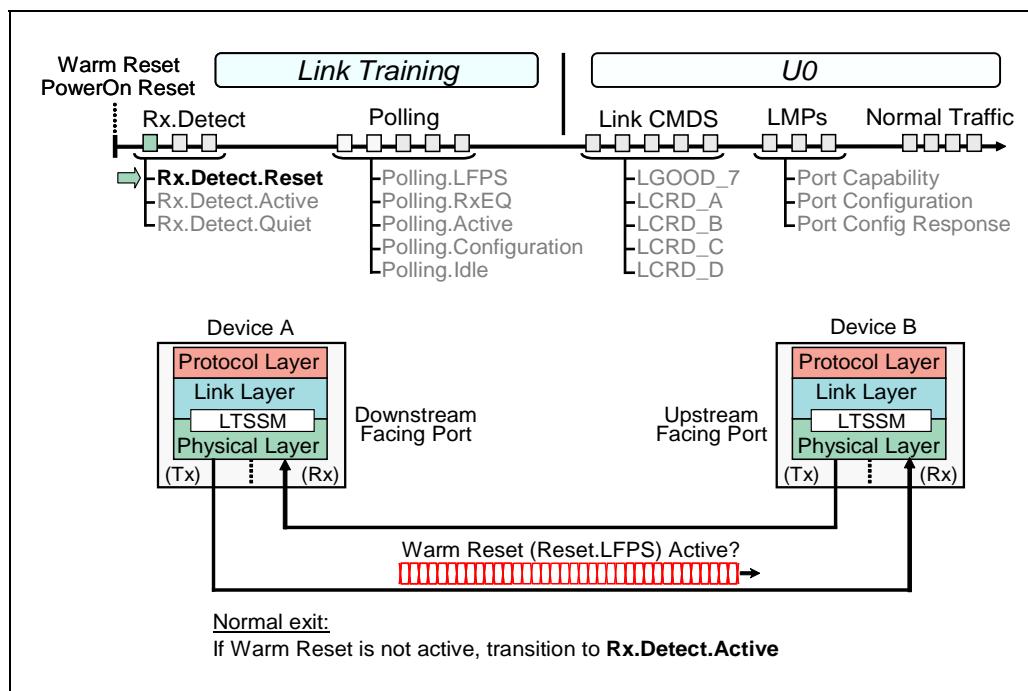
## The Normal Link Training Sequence

The following section describes the purpose of each Rx.Detect and Polling substate in the normal link training sequence. Starting with PowerOn or Warm Reset at the left and ending with entry into the U0 operational state at the right, link partners proceed together through training. Handshake signaling is used on the link to indicate when they have completed the requirements of the current substate and are ready to proceed to the next. *Note:* this link training sequence assumes that there are no errors or time-outs.

### Rx.Detect.Reset

As indicated in Figure 20-4 on page 435, Rx.Detect.Reset is the first substate in link training. Link partners are required to pause in this substate before proceeding with training if a Warm Reset was the reason the Rx.Detect state was entered, and Warm Reset is still active on the link.

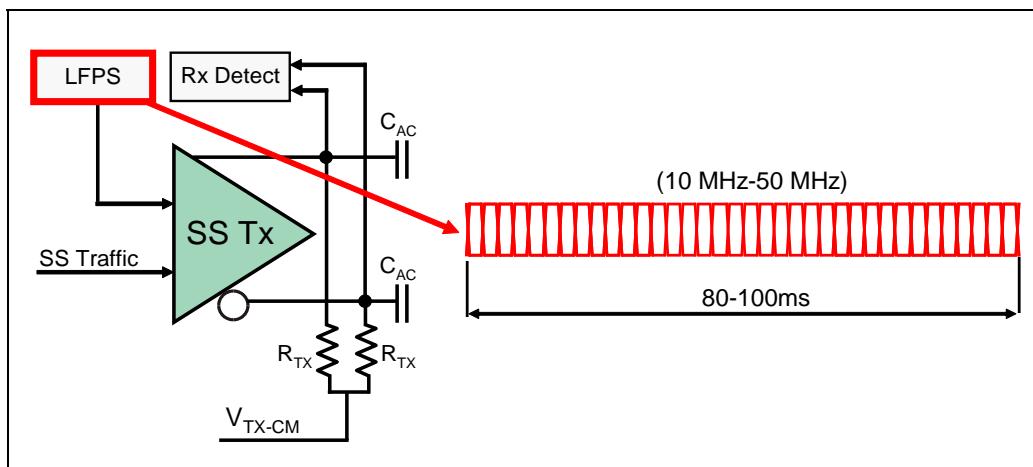
Figure 20-4: Link Training Is Delayed Until Warm Reset Ends



## USB 3.0 Technology

---

Figure 20-5: Warm Reset LFPS Characteristics



Only downstream facing ports are permitted to assert a Warm Reset, and only if directed to do so by software. As indicated in Figure 20-5, the Warm Reset LFPS duration is approximately 100mS.

### Normal Exit From Rx.Detect.Reset

The normal exit from the Rx.Detect.Reset substate is to Rx.Detect.Active. Conditions enabling this transition depend on the port type and the circumstances that originally caused the transition to Rx.Detect:

- For a downstream facing port that asserts Warm Reset, a transition automatically occurs after the port has met the 80-100mS Reset.LFPS assertion time. The port de-asserts Warm Reset and transitions to Rx.Detect.Active.
- For an upstream facing port that detected a Warm Reset, a transition to Rx.Detect.Active occurs when detects Warm Reset de-asserted.
- In cases where the transition to Rx.Detect was the result of an event other than Warm Reset, an exit to Rx.Detect.Active occurs automatically for all port types. The most common example of this case is a PowerOn Reset.

### Other Exits From Rx.Detect.Reset

It is always possible (even during link training) that a downstream facing hub port is targeted by software with a command to disable the port. If this occurs, the command to disable the port takes precedence over link training and the downstream facing port would disable its receiver termination and become "not visible" to its SuperSpeed link partner.

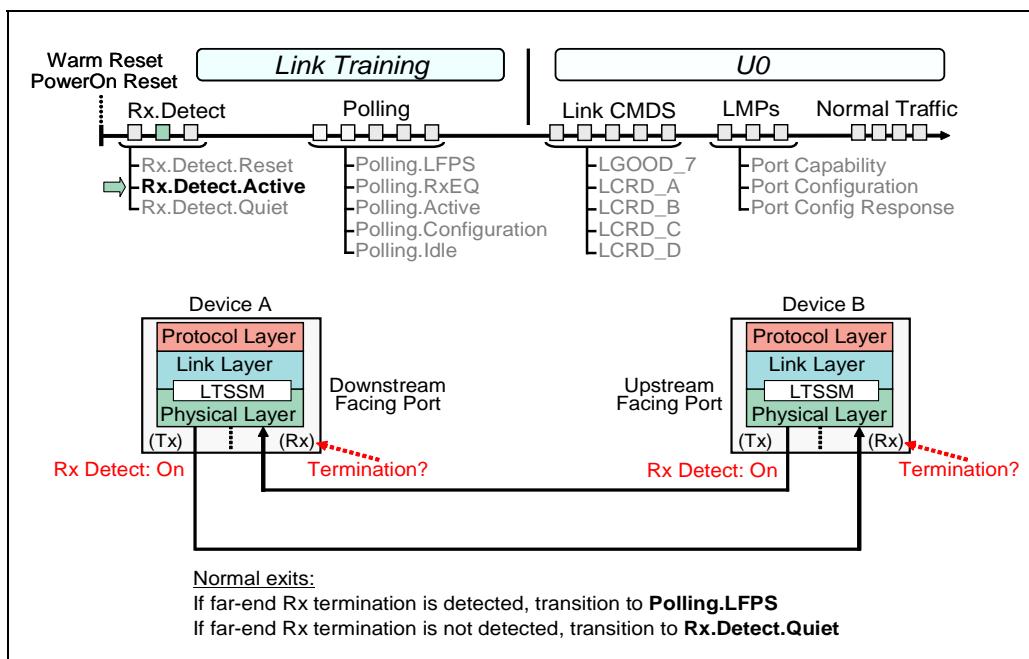
# Chapter 20: Link Training

## Rx.Detect.Active

This is the second substate in the Rx.Detect link training state. The purpose of this state is to provide an opportunity for attached devices to detect the presence of a SuperSpeed-capable link partner before attempting link training. This is important for a variety of reasons which include:

- It may be that there is no attached device
- USB 3.0 cables and connectors support both SuperSpeed and USB 2.0 signals and a device may be attached that is USB 2.0-only
- Two USB 3.0 devices may be connected, but software may have disabled SuperSpeed operation in the downstream facing port. Because the downstream facing port will automatically remove its receiver termination when it is disabled, the link partner connected to it won't proceed beyond the Rx.Detect.Active state, and will quickly timeout and "fall back" to USB 2.0 operations (SuperSpeed interface will not be used).

Figure 20-6: Checking For A Link partner In Rx.Detect.Active

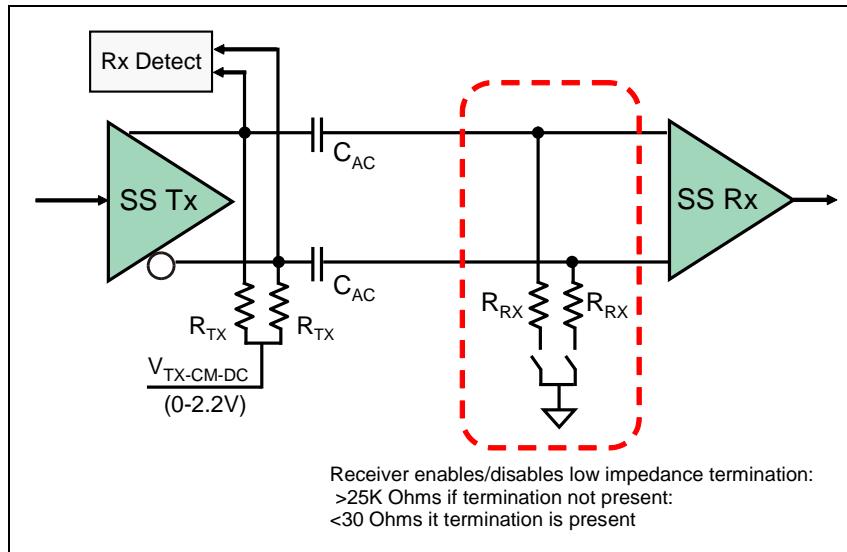


# USB 3.0 Technology

## SuperSpeed Receivers Enable & Disable Termination

As indicated in Figure 20-7 on page 438, SuperSpeed receivers are capable of enabling and disabling their low-impedance termination resistors. Whether or not the low-impedance termination is enabled depends on the LTSSM state. For example, if the link state is U0, receiver termination is always enabled to provide the required differential signal termination needed when 5 Gb/s SuperSpeed traffic is on the bus. On the other hand, if  $V_{BUS}$  is removed by the downstream facing port then the bus is in the disabled state, and link partners are both required to disable the low-impedance receiver termination (even if they are self-powered).

Figure 20-7: Receiver Enables & Disables Low-Impedance Termination



## Receiver Common Mode Input Impedance Range

From the perspective of a transmitter “looking into” the attached SuperSpeed receiver, either a low or high far-end DC common mode impedance is seen.

- $R_{RX-DC}$  is receiver DC common mode input impedance when termination is enabled ( $18-30 \Omega$ s)
- $Z_{RX-HIGH-IMP-DC-POS}$  is receiver DC common mode input impedance when termination not enabled ( $25 K\Omega$ s, min.)

## Chapter 20: Link Training

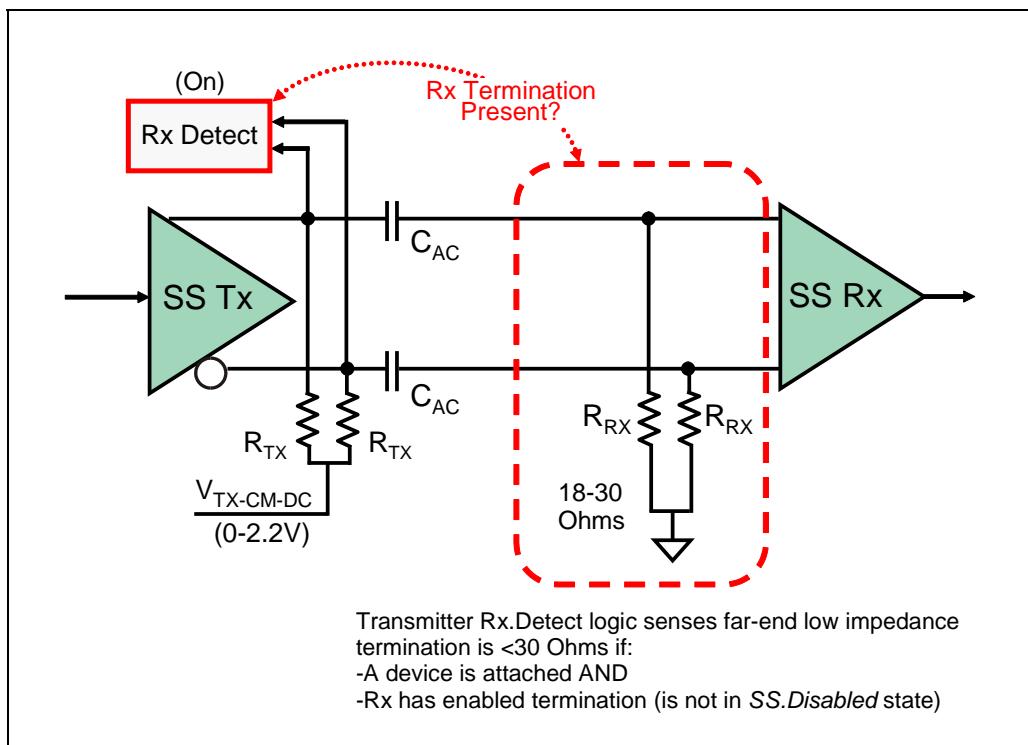
### The Detection Method

The capability of receivers to enable and disable low-impedance termination is used during link training to provide a simple way for device transmitters to determine whether a SuperSpeed-capable link partner is connected.

In all substates of Rx.Detect, SuperSpeed receivers enable their low impedance termination:  $R_{RX-DC}$  equal to 18-30  $\Omega$ s. The detection scheme, as depicted in Figure 20-8 on page 439, requires the SuperSpeed Transmitter to enable its integrated Rx Detect logic and follow the following general sequence:

- Transmitter establishes a stable common mode voltage ( $V_{TX-CM-DC}$ )
- Transmitter steps common mode voltage on D+/D- in a positive direction
- Transmitter Rx Detect logic measures the time required to raise Tx D+/D- outputs to the new voltage level. Rise time is proportional to load, a function of transmitter impedance, series capacitor ( $C_{AC}$ ), the interconnect capacitance, and the receiver termination (if enabled).

Figure 20-8: SuperSpeed Transmitter Detects Receiver Termination

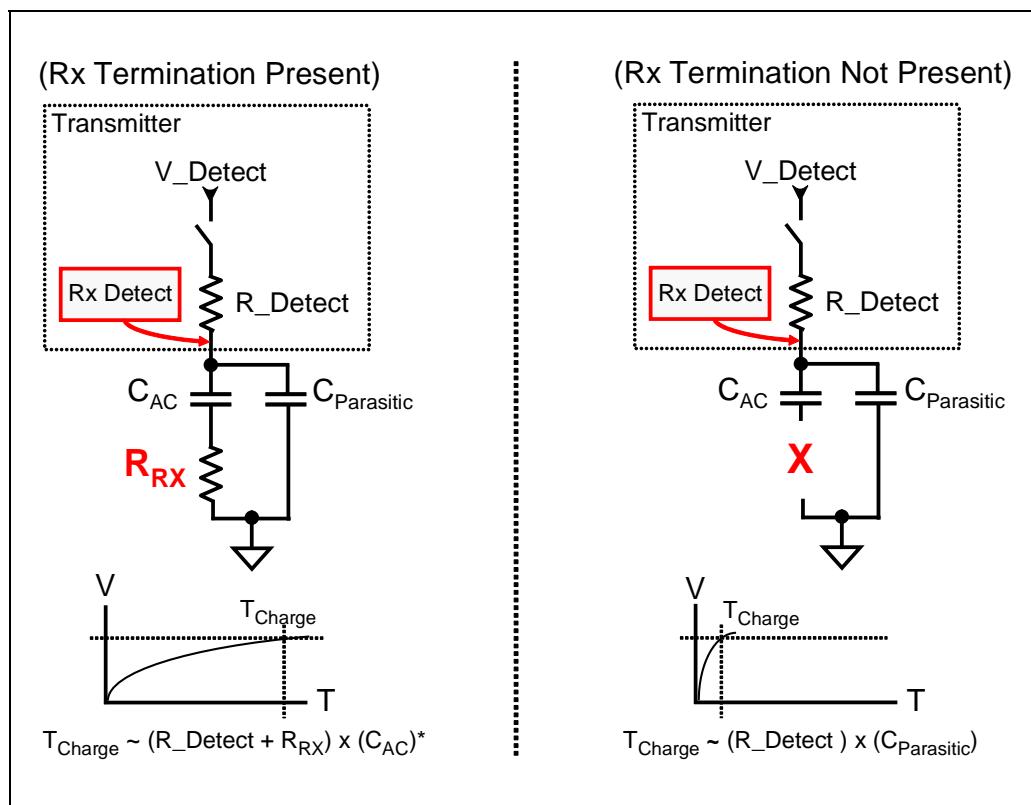


## USB 3.0 Technology

The method for detecting receiver termination is based on the behavior of a simple resistor-capacitor circuit attached to a dc power supply. The time required to charge the capacitor when voltage is applied is proportional to the size of the resistors multiplied by the size of the capacitors ( $T = R \times C$ ). Figure 20-9 on page 440 depicts equivalent circuits for the two Rx.Detect.Active cases.

- On the left is the case where the transmitter performs the common mode voltage shift, measures the rise time (relatively long), and determines that SuperSpeed receiver termination is present in the link partner. Note that the resistance consists of transmitter impedance ( $R_{\text{Detect}}$ ) and  $R_{\text{RX}}$ ; capacitance includes  $C_{\text{AC}}$  and a small amount of parasitic capacitance (which is ignored in the calculation shown)
- On the right of Figure 20-9, the same procedure is followed, but no receiver termination is detected (rise time is short). Here, resistance consists only of transmitter impedance ( $R_{\text{Detect}}$ ); capacitance is small, and only parasitic.

Figure 20-9: Rx Detect Equivalent Circuits



## **Chapter 20: Link Training**

---

### **Normal Exit: From Rx.Detect.Active To Polling**

The normal exit from the Rx.Detect.Active substate, assuming that receiver termination was detected, is to the Polling state. In the Polling state, link training will commence.

### **Alternative Exit: From Rx.Detect.Active To Rx.Detect.Quiet**

If far-end termination was not detected, then another Rx.Detect substate, called Rx.Detect.Quiet, is used to conserve power until Rx.Detect.Active is re-entered and another attempt at receiver termination detection is made. Repeated attempts to start and complete link training are important:

- Because of USB hot plug, downstream facing ports of the root hub or external hubs must be capable of recognizing attachment and starting link training at any time (unless software or a hardware error has disabled the port).
- A USB 3.0 peripheral is only permitted to connect at one protocol at a time (SuperSpeed or USB 2.0). Because the default connection is SuperSpeed, a peripheral is required to reattempt SuperSpeed link training up to eight times, then “fall back” and connect at a USB 2.0 speed (SuperSpeed interface is then disabled).
- Unlike a peripheral device, a hub attempts to connect at both USB 2.0 and SuperSpeed on its upstream interface. It is not limited to eight attempts at SuperSpeed connection on its upstream facing port and will continue to transition between Rx.Detect.Active and Rx.Detect.Quiet.

### **Other Exits From Rx.Detect.Active**

In this link training sequence example, we are assuming the normal exit from Rx.Detect.Active to the Polling state is to be taken--with possible transitions to and from Rx.Detect.Quiet until a link partner is detected. For a discussion of other cases of exits from this substate, refer to Chapter 12, entitled "LTSSM And the SuperSpeed Link States," on page 243.

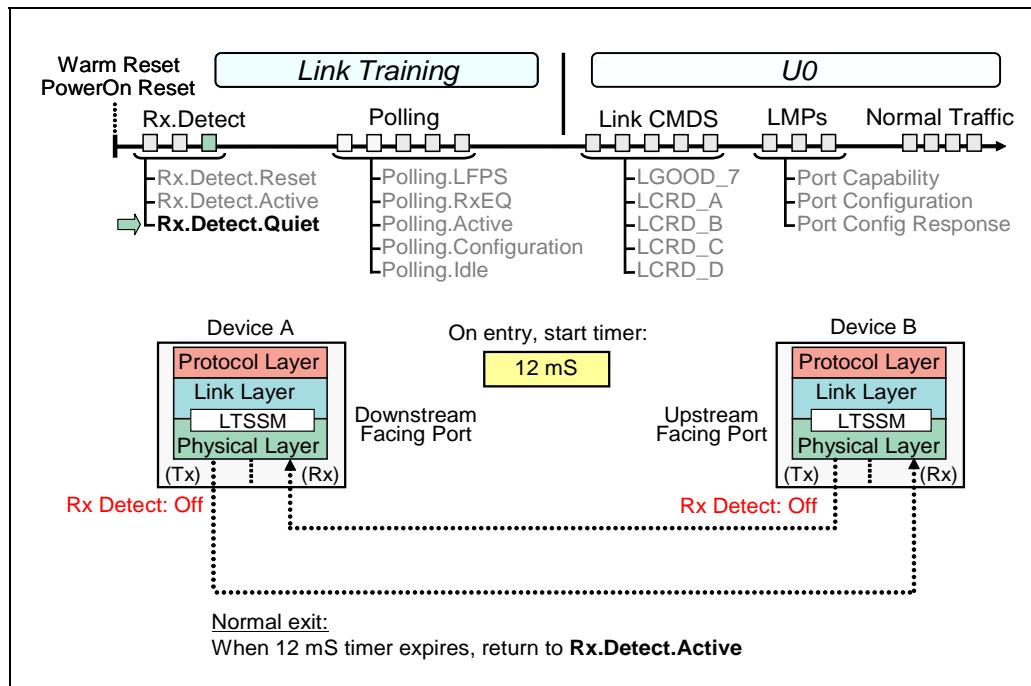
# USB 3.0 Technology

## Rx.Detect.Quiet

This is the Rx.Detect substate that is only entered in the event that the previous transition into Rx.Detect.Active resulted in a failure to detect far-end low impedance receiver termination. The transition to Rx.Detect.Quiet substate provides an opportunity for the transmitter to disable its Rx Detect logic and conserve power before returning to Rx.Detect.Active and trying again.

As indicated in Figure 20-10 on page 442, transmitters in Rx.Detect.Quiet disable Rx Detect logic and start a 12 ms timer.

Figure 20-10: Conserving Power In Rx.Detect.Quiet If Link Partner Isn't Present



## Normal Exit: From Rx.Detect.Quiet To Rx.Detect.Active

When the 12 ms timer expires, a return to Rx.Detect.Active is triggered.

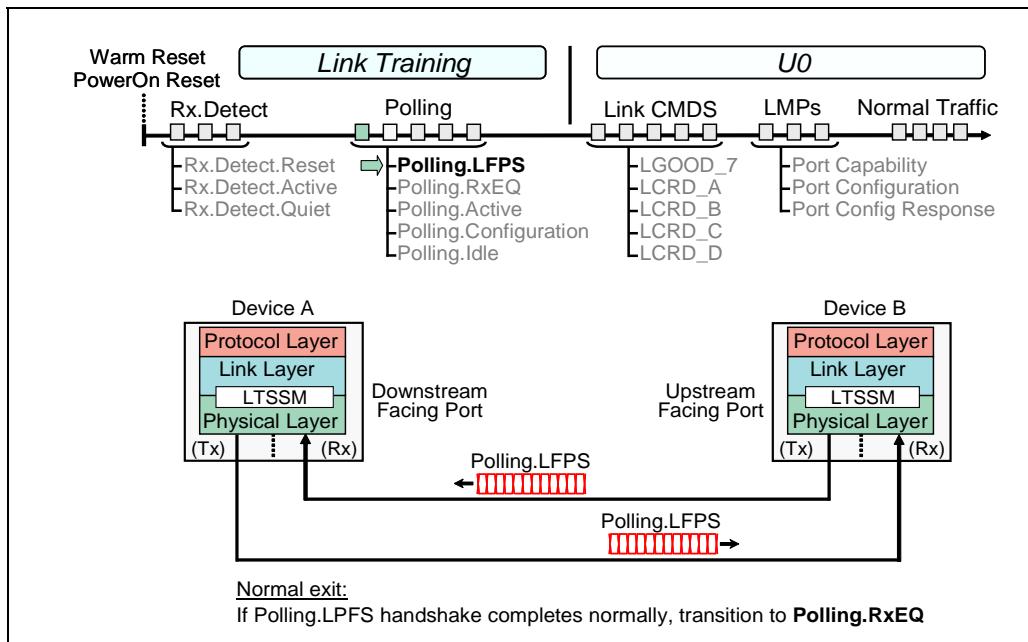
## Chapter 20: Link Training

### Polling.LFPS

Polling has five substates and is entered only if far-end low impedance receiver termination was detected in Rx.Detect.Active. In the Polling.LFPS substate, link partners exchange 10MHz-50MHz low frequency periodic signaling (LFPS) bursts which accomplishes two things:

- Alerts the link partner receiver that link training is commencing
- Enables each receiver to establish DC operating point for equalization logic

Figure 20-11: Link Traffic Starts With Low Frequency Periodic Signaling (LFPS)

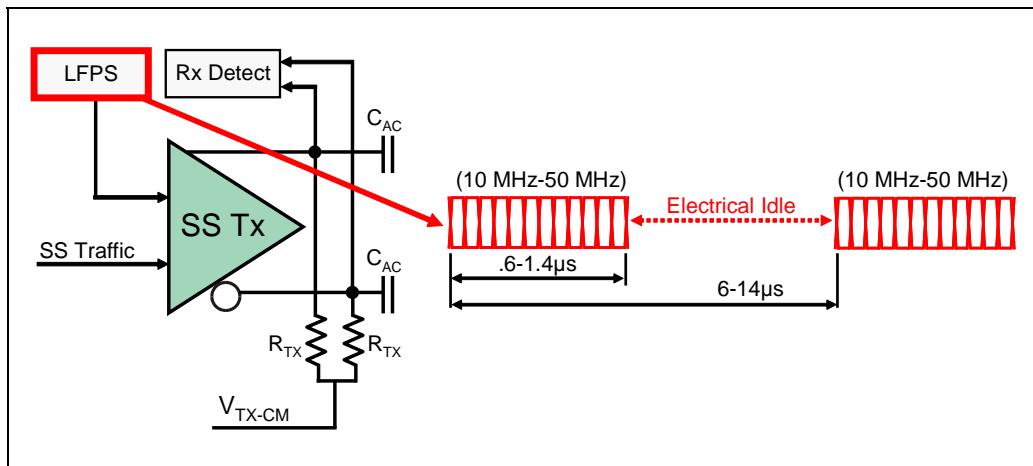


### The Polling.LFPS Burst Format

As indicated in Figure 20-12 on page 444, the Polling.LFPS is delivered in repeating bursts. The duration of each 10 MHz to 50 MHz burst is nominally 1μs and repeats every 6-14 μs. The period between LPFS bursts is electrical idle.

## USB 3.0 Technology

Figure 20-12: Characteristics Of The Polling.LFPS Burst Sequence



### **Other Polling.LFPS Requirements And Options**

There are a number of requirements and options covering device behavior in the Polling.LFPS substate:

- On entry to polling.LFPS state from Rx.Detect.Active, all devices must be enabled for LFPS communications within 80  $\mu$ s
- During Polling.LFPS, device may enable SuperSpeed receiver in preparation for 5 Gb/s TSEQ ordered sets which follow in next substate.
- On entry to Polling.LFPS, a 360 ms timer is started.

### **Normal Exit: From Polling.LFPS To Polling.RxEQ**

The normal exit from the Polling.LFPS substate is to Polling.RxEQ where 5 Gb/s link training will start. This exit will occur if:

- Device has transmitted at least 16 Polling.LFPS bursts
- Device has received at least 2 consecutive Polling.LFPS bursts
- Device has transmitted at least four consecutive Polling.LFPS bursts after receiving one

### **Other Exits From Polling.LFPS**

In this link training sequence example, we are assuming the normal exit from Polling.LFPS to Polling.RxEQ is taken. There are a number of other exits from Polling.LFPS that vary with port type, whether the 360 ms timer has expired,

# Chapter 20: Link Training

whether the link has been previously trained since the last Warm Reset, etc. Refer to Chapter 12, entitled "LTSSM And the SuperSpeed Link States," on page 243 for a description of each of these cases.

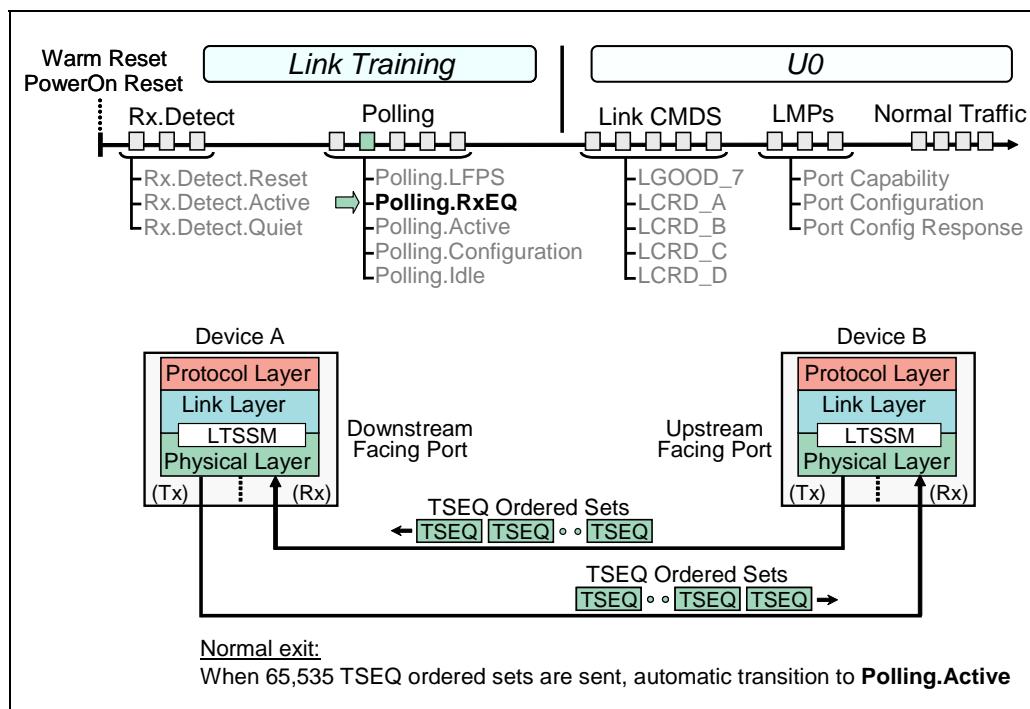
## Polling.RxEQ

This substate is used to complete the training of receiver equalizer logic. It is also the first substate in which 5 Gb/s bit rate traffic moves across the link. As shown in Figure 20-13 on page 445, link partners exchange TSEQ ordered sets. Note that this substate does not require a positive handshake to exit; each device sends at least 64K (65,536) TSEQ ordered sets, then transitions to the next substate when it has completed its own equalization training using TSEQs supplied by the link partner.

### Other Requirements In Polling.RxEQ

Another aspect of device behavior in the Polling.RxEQ substate is that TSEQ ordered sets are used to detect and correct differential polarity inversion (if needed). Differential polarity inversion (and its correction) is described shortly.

Figure 20-13: Devices Send SuperSpeed TSEQ Ordered Sets For Receiver Equalization



# **USB 3.0 Technology**

---

## **The TSEQ Ordered Set**

Repeated 65,536 times, the TSEQ ordered set exchanged during Polling.RxEQ is comprised of thirty two 10-bit symbols. The symbols within the TSEQ ordered set were selected for frequency content and bit patterns useful in training receiver equalization, achieving bit and symbol lock, etc. Table 20-1 on page 446 lists each of the 10-bit symbols that form the TSEQ ordered set. Also shown in the table are the pairs of 10-bit encodings available to the transmitter when maintaining a balanced *running disparity* (0's and 1's) on the link.

*Table 20-1: TSEQ Ordered Set Symbol Encoding*

<b>Symbol #</b>	<b>Name</b>	<b>Hex Value</b>	<b>10-Bits (CRD-)</b>	<b>10-Bits (CRD+)</b>
0	K28.5	COM	0011111010	1100000101
1	D31.7	0xFF	1010110001	0101001110
2	D23.0	0x17	1110100100	0001011011
3	D0.6	0xC0	1001110110	0110000110
4	D20.0	0x14	0010111011	0010110100
5	D18.5	0xB2	0100111010	0100111010
6	D7.7	0xE7	1110001110	0001110001
7	D2.0	0x02	1011010100	0100101011
8	D2.4	0x82	1011010010	0100101101
9	D18.3	0x72	0100111100	0100110011
10	D14.3	0x6E	0111001100	0111000011
11	D8.1	0x28	1110011001	0001101001
12	D6.5	0xA6	0110011010	0110011010
13	D30.5	0xBE	0111101010	1000011010
14	D13.3	0x6D	1011001100	1011000011
15	D31.5	0xBF	1010111010	0101001010
16-31	D10.2	0x4A	0101010101	0101010101

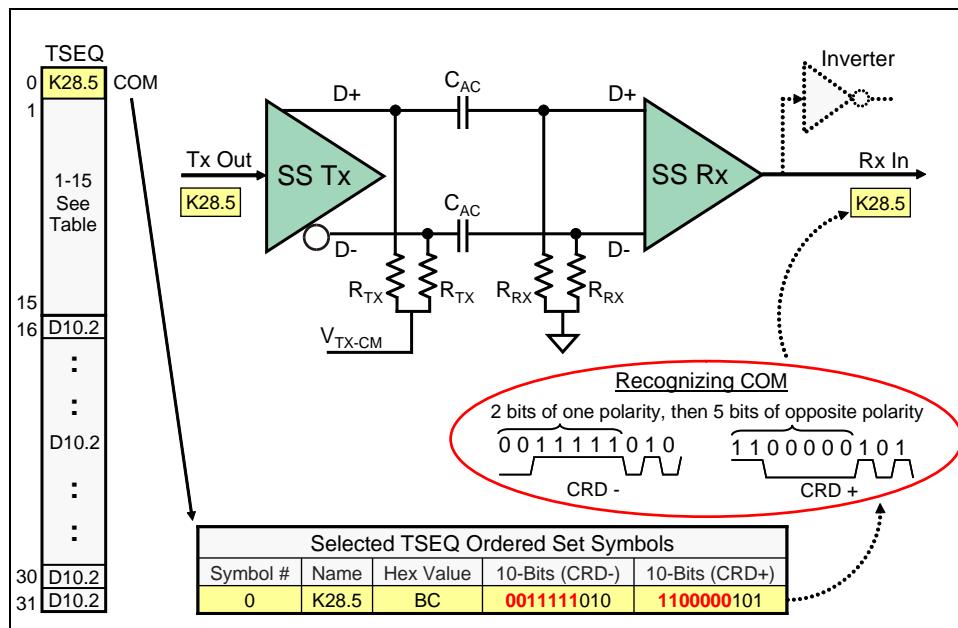
## Chapter 20: Link Training

### Two Special Symbols In TSEQ

Two of the TSEQ symbol types shown in Table 20-1 on page 446 have special uses worth noting here.

**Symbol K28.5 (COM).** The COM (Comma) is the first symbol of each 32-byte TSEQ ordered set. During link training (and retraining) it is the only 10-bit symbol with a bit pattern that includes 2 bits of one polarity, followed by 5 bits of the opposite polarity. Encoding of other USB 3.0 Super-Speed symbols during these times also assures this special “2 and 5” pattern doesn’t occur because of bit combinations resulting from two neighboring symbols. The uniqueness of the COM symbol makes it easily recognizable. During link training, the receiver samples the incoming bit stream in search of the COM “2 and 5” bit pattern. Once COM is detected, symbol lock has been achieved and every ten bits thereafter is a new symbol. Because the TSEQ ordered set is repeated 65,536 times during link training, the receiver has a lengthy opportunity to recover the transmitter clock, sample the TSEQ bit stream, recognize COM, and achieve symbol lock.

Figure 20-14: COM Symbol Bit Pattern Is Easily Recognized By Receiver



## USB 3.0 Technology

---

Figure 20-14 on page 447 depicts the transmission\reception of the COM symbol. As with all 10-bit symbols, there are two versions of COM. The version of COM sent depends on the *current running disparity* (CRD) for bits sent previously; note that both versions of COM contain the special bit pattern of two bits of one polarity followed by five bits of the other ("2 and 5").

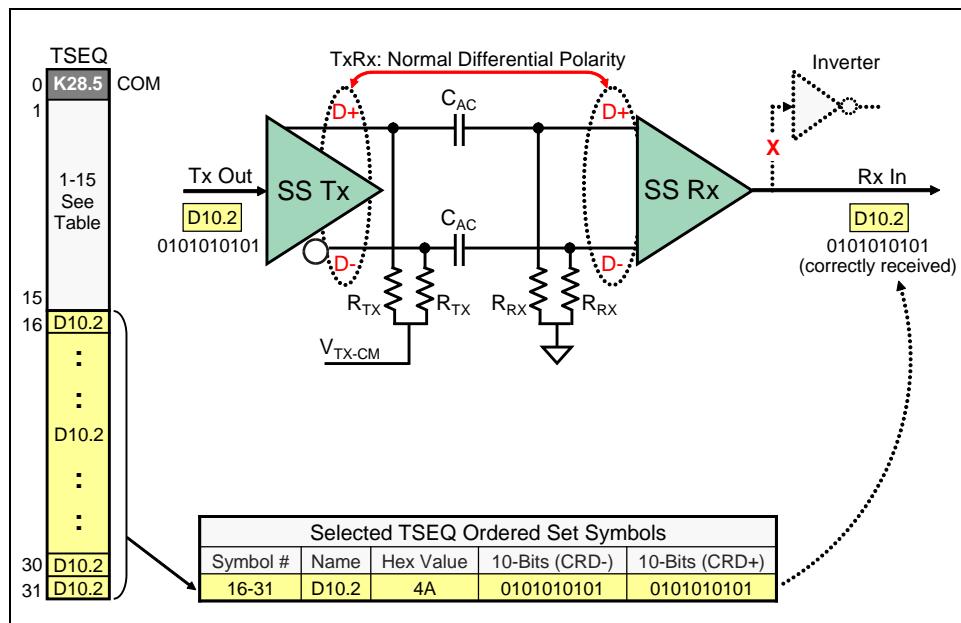
**Symbol 16-31 D10.2.** The last 16 symbols of each 32-byte TSEQ ordered set are D10.2. This symbol type has two important uses:

The first use is to help train receiver equalization logic. Looking at Table 20-1 on page 446, it can be seen that the bit pattern for D10.2 is toggling 0's and 1's. When the last 16 symbols of TSEQ are sent back-to-back, the result is a 2.5 GHz squarewave enabling the receiver to set up highest frequency equalization coefficients.

The second key use of the TSEQ D10.2 symbols is to enable the receiver to perform the required check during link training for differential polarity inversion, and correct it if necessary. A differential transmitter employs two output pins (Tx D+/D-). Differential receivers have two input pins (Rx D+/D-). The expectation is that for each pair of link partners:

- Transmitter D+ is connected to receiver D+
- Transmitter D- is connected to receiver D-

Figure 20-15: Normal Transmitter And Receiver Differential Polarity



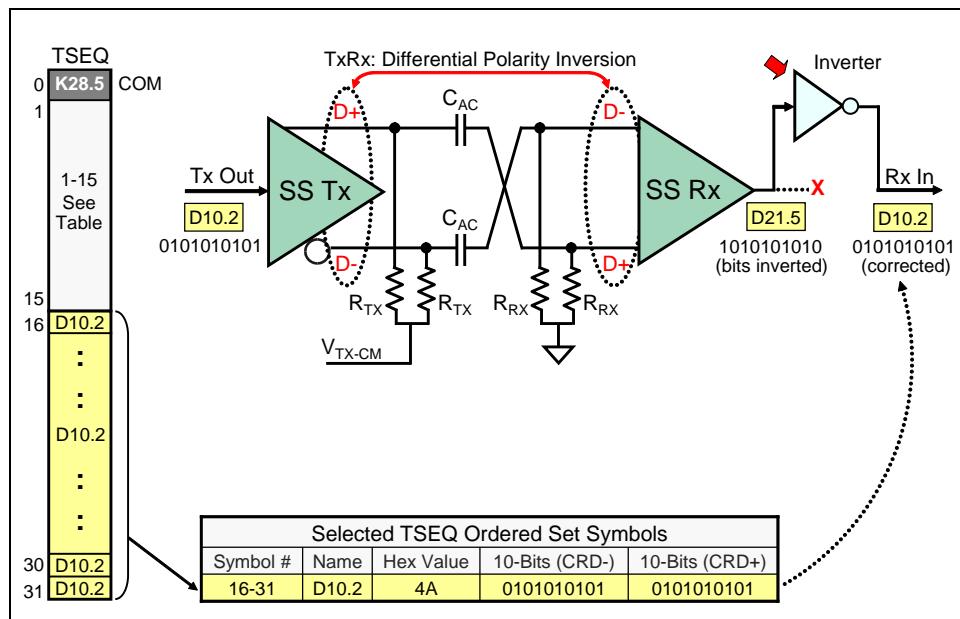
## Chapter 20: Link Training

Figure 20-15 on page 448 depicts a SuperSpeed transmitter and receiver connected with normal (correct) Tx/Rx differential polarity. Note in the illustration that the D10.2 symbols are received as sent.

In some cases, device pinout, pc board trace routing, or connector layout results in a swap of channel differential signal pairs in the path from transmitter to receiver. During link training, if the D10.2 symbols (TSEQ symbols 16-31) are received as D21.5, differential polarity is inverted and the receiver activates an internal inverter to correct it.

Figure 20-16 on page 449 illustrates the case where a SuperSpeed transmitter and receiver are connected with inverted differential signal polarity. Notice that the D10.2 symbol sent by the transmitter is received as D21.5, then internally corrected by enabling the inverter shown in at the right of the illustration.

Figure 20-16: Differential Polarity Inversion Corrected By The Receiver



### Normal Exit: From Polling.RxEQ To Polling.Active

The normal exit from the Polling.RxEQ substate is to Polling.Active where any link training not completed in Polling.RxEQ will complete. This exit occurs automatically when a port has transmitted 65,536 TSEQ ordered sets.

# USB 3.0 Technology

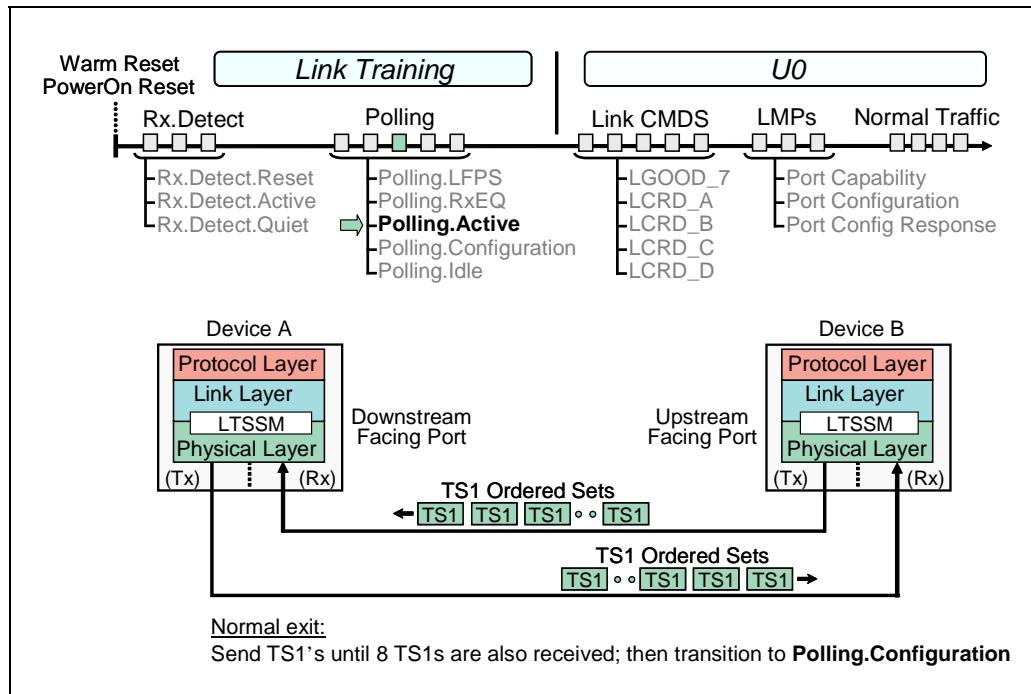
## Polling.Active

In this state, devices continue with any link training that was not completed in Polling.RxEQ. While in this state, devices exchange 16-byte TS1 ordered sets. Because ports train at different rates, one generally transitions to the next substate (Polling.Configuration) before the other.

### Other Requirements In Polling.Active

On entry to the Polling.Active substate, devices start a 12 ms timer.

Figure 20-17: TS1 Ordered Sets Appear On The Link In Polling.Active

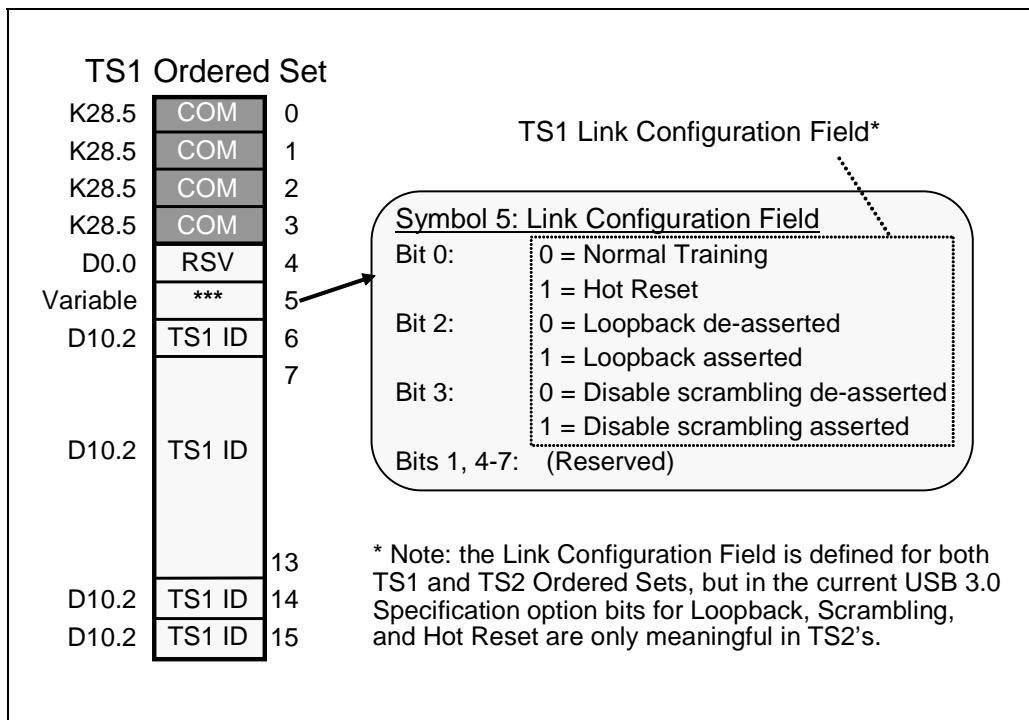


## The TS1 Ordered Set

TS1 is *Training Sequence 1*. When received, it informs the link partner that the attached device has completed work with (or doesn't use) the TSEQ ordered set and has transitioned to Polling.Active to complete link training. Figure 20-18 on page 451 depicts the format of the TS1 ordered set.

## Chapter 20: Link Training

Figure 20-18: TS1 Ordered Set Format



Note that, as with TSEQ ordered sets, TS1s include K28.5 (COM) and D10.2 which can be used for bit/symbol lock and receiver equalization. The USB 3.0 specification indicates that use of TSEQ ordered sets for link training is optional; a device that trains quickly could simply ignore incoming TSEQs and wait until the arrival of TS1's to set up equalization, and achieve bit/symbol lock. Regardless, all SuperSpeed devices must send 65,536 TSEQ ordered sets to their link partners.

### Normal Exit: From Polling.Active To Polling.Configuration

The normal exit from the Polling.Active substate is to Polling.Configuration. This exit occurs when a port completes training and receives 8 consecutive, identical TS1s or TS2s.

# USB 3.0 Technology

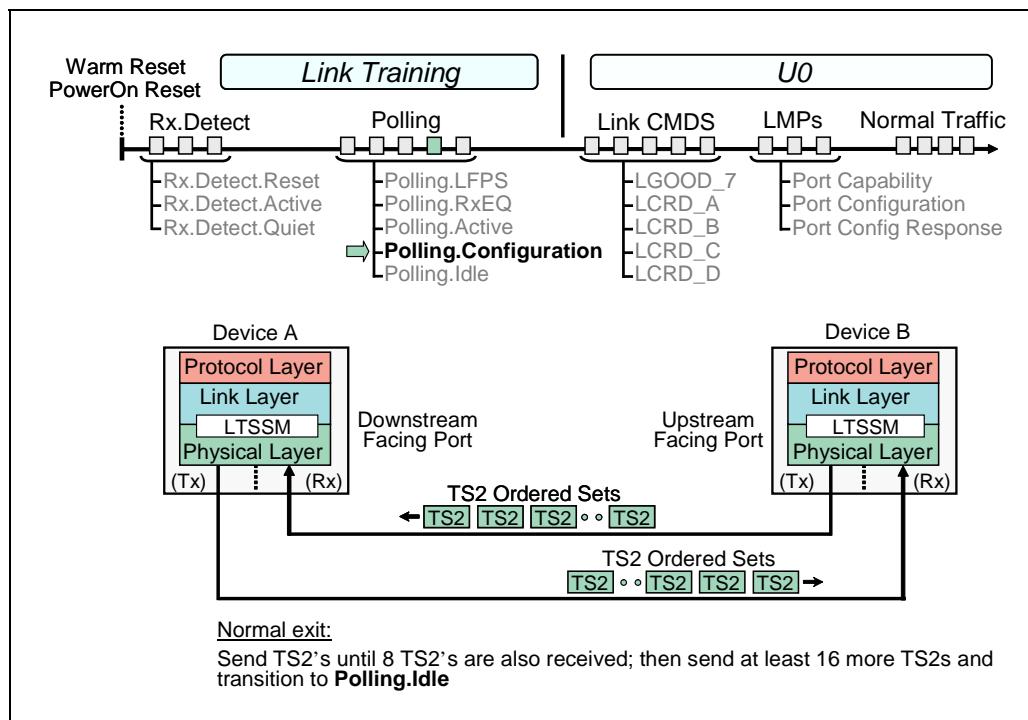
## Other Exits From Polling.Active

In this link training sequence example, we are assuming the normal exit from Polling.Active to Polling.Configuration state is to be take. For a discussion of other cases of exits from this substate, refer to Chapter 12, entitled "LTSSM And the SuperSpeed Link States," on page 243.

## Polling.Configuration

As shown in Figure 20-19 on page 452, devices in Polling.Configuration exchange TS2 ordered sets to signal that they have completed link training and are prepared to exit. Normally, they indicate a pending transition to the U0 active link state. While it doesn't apply in this generic link training example, Polling.Configuration is also an opportunity for devices to use the TS2 ordered sets to indicate that they intend to transition to Hot Reset, Loopback, or would like to disable scrambling for debug purposes.

Figure 20-19: TS2 Ordered Sets In Polling.Configuration Signal End Of Link Training

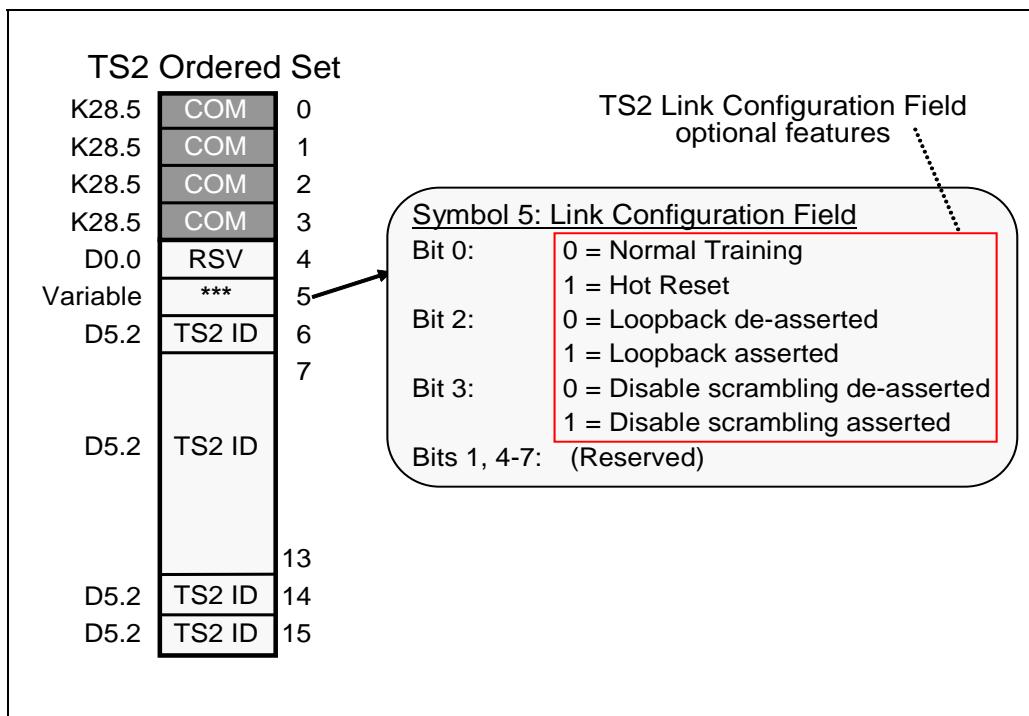


# Chapter 20: Link Training

## The TS2 Ordered Set

TS2 is *Training Sequence 2* and Figure 20-20 on page 453 depicts the TS2 ordered set 16-byte format. The TS2 Link Configuration Field (in symbol 5) is used to indicate which option is requested: a transition to U0 (with/without scrambling enabled), Hot Reset, or Loopback.

Figure 20-20: TS2 Ordered Set Format



## Normal Exit: From Polling.Configuration To Polling.Idle

The normal exit from the Polling.Configuration substate is to Polling.Idle. This exit occurs only if the TS2 Link Configuration Field (in symbol 5) did not indicate that a transition to Hot Reset or Loopback was requested.

The port makes the transition to Polling.Idle when:

- 8 consecutive and identical TS2s were received (with no Loopback or Hot Reset indication)
- 16 TS2s were sent after receiving 8 consecutive, identical TS2s.

# USB 3.0 Technology

## Other Exits From Polling.Configuration

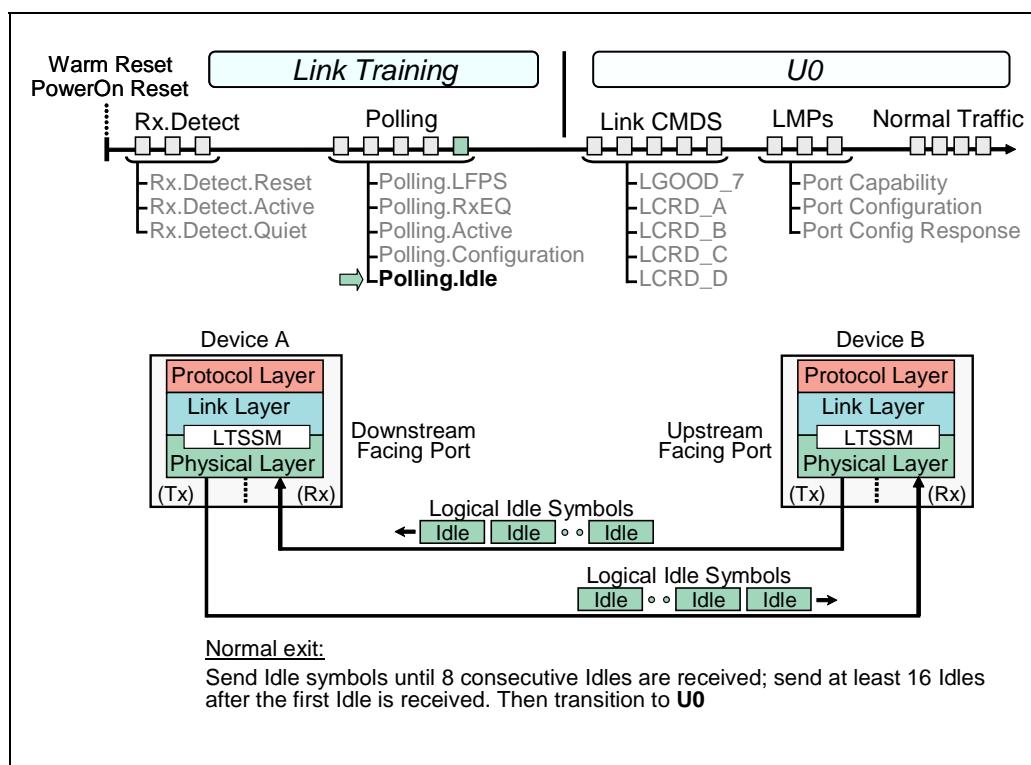
In this link training sequence example, we are assuming the normal exit from Polling.Configuration to Polling.Idle state is to be taken. As described previously, there are alternative exits to Loopback or Hot Reset. For all of the exit possibilities, refer to Chapter 12, entitled "LTSSM And the SuperSpeed Link States," on page 243.

## Polling.Idle

### General

In this state, ports have already decoded TS2s received in Polling.Configuration and will exchange logical idle (D0.0) symbols if they are about to enter U0. If scrambling disable was requested in TS2s, it is applied now.

Figure 20-21: Idle Symbols Confirm That The Next State Is U0



# Chapter 20: Link Training

## Other Requirements In Polling.Idle

On entry to the Polling.Idle, the link is about to become available for normal SuperSpeed traffic. In preparation for entry into U0, the following rules apply:

- Downstream facing ports reset their link error counters (LECs)
- Upstream facing ports reset port configuration information to default value
- Ports prepare to receive Header Sequence Number advertisement. Normal Exit: From Polling.Idle To U0

Normal exit from Polling.Idle substate: U0. The port makes the transition when:

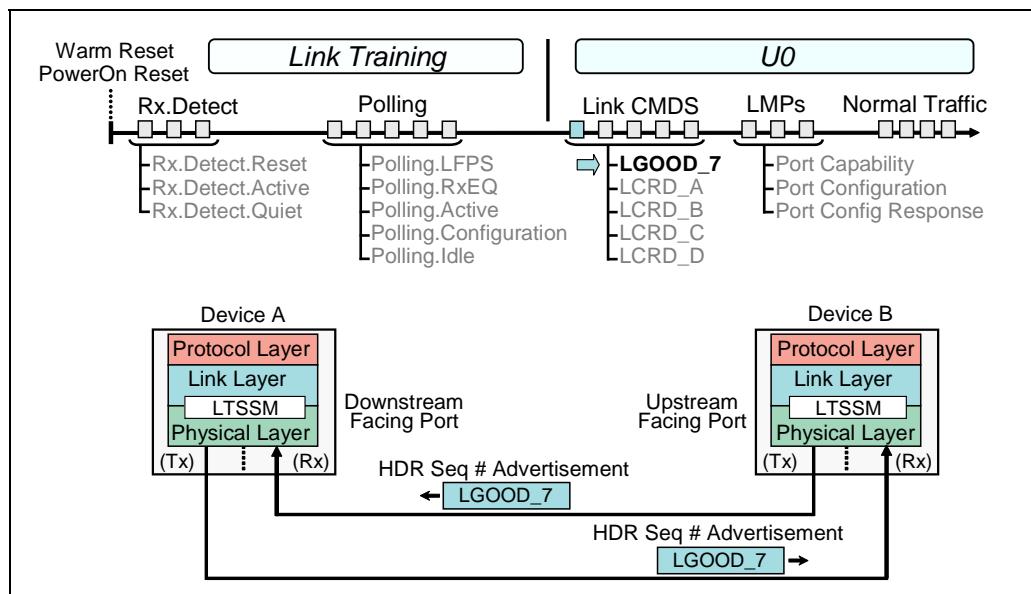
- 8 Idle symbols are received and
- 16 Idle symbols have been sent after receiving at least 1 Idle symbol

## LGOOD\_n Advertisement

### General

On entry into U0, link partners inform one another of the starting link sequence number for header packets. Link sequence numbers range from 0-7; on entry into U0 following PowerOn Reset or Warm Reset, the starting link sequence number is always = 0. The sequence number advertisement is one less than actual value, so as indicated in Figure 20-22 on page 455, link partners send the link command LGOOD\_7.

Figure 20-22: LGOOD\_n Link Command Advertises Header Packet Sequence Number

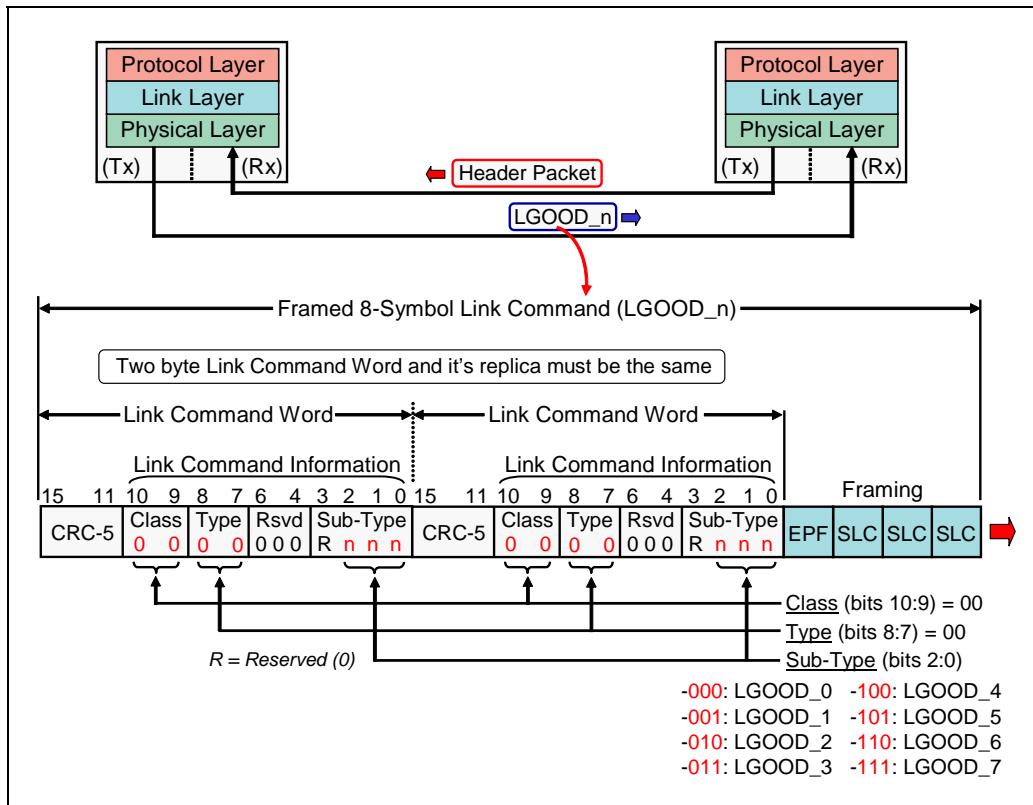


# USB 3.0 Technology

## Format Of LGOOD\_n Link Command

Figure 20-23 on page 456 illustrates the format of an LGOOD\_n link command used to advertise the starting header packet sequence number. On entry into U0 from PowerOn or Warm Reset, the value "n" should be = 7.

Figure 20-23: Format Of LGOOD\_n Link Command



# Chapter 20: Link Training

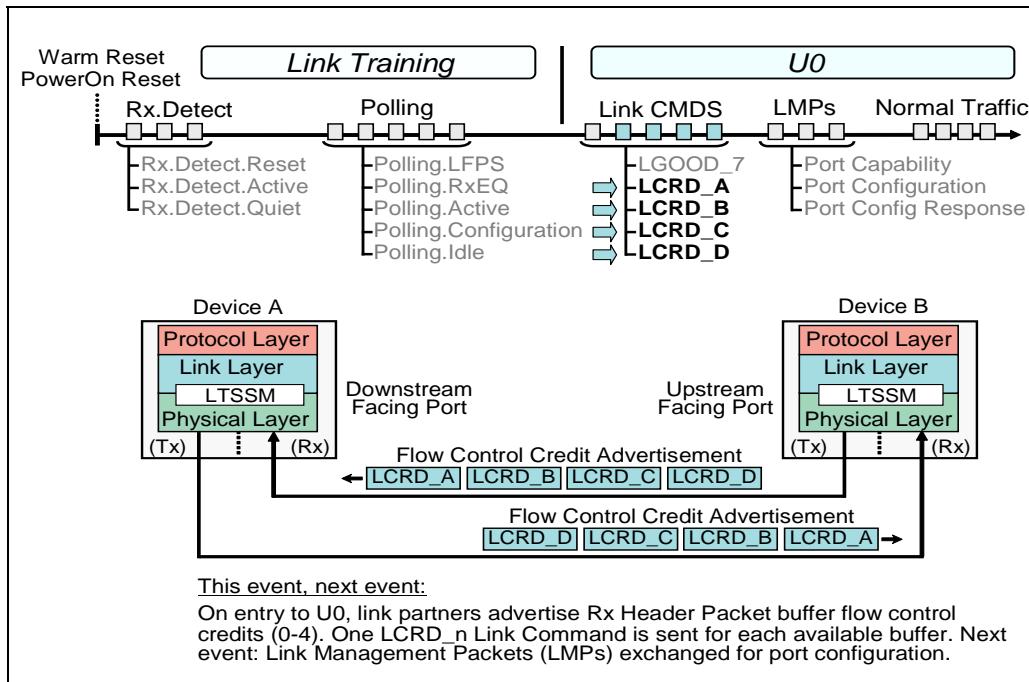
## LCRD\_x Advertisement

### General

The other link command advertisement required on entry to U0 is for flow control credits. Link partners always send one flow control credit link command for each available link layer receive buffer. The current USB 3.0 specification requires four receive buffers (referred to as A-D).

As shown in Figure 20-24 on page 457, on entry to U0 following Warm Reset or PowerOn Reset, four flow control link commands are sent in the order shown.

Figure 20-24: LCRD\_x Link Commands Initialize Header Packet Flow Control

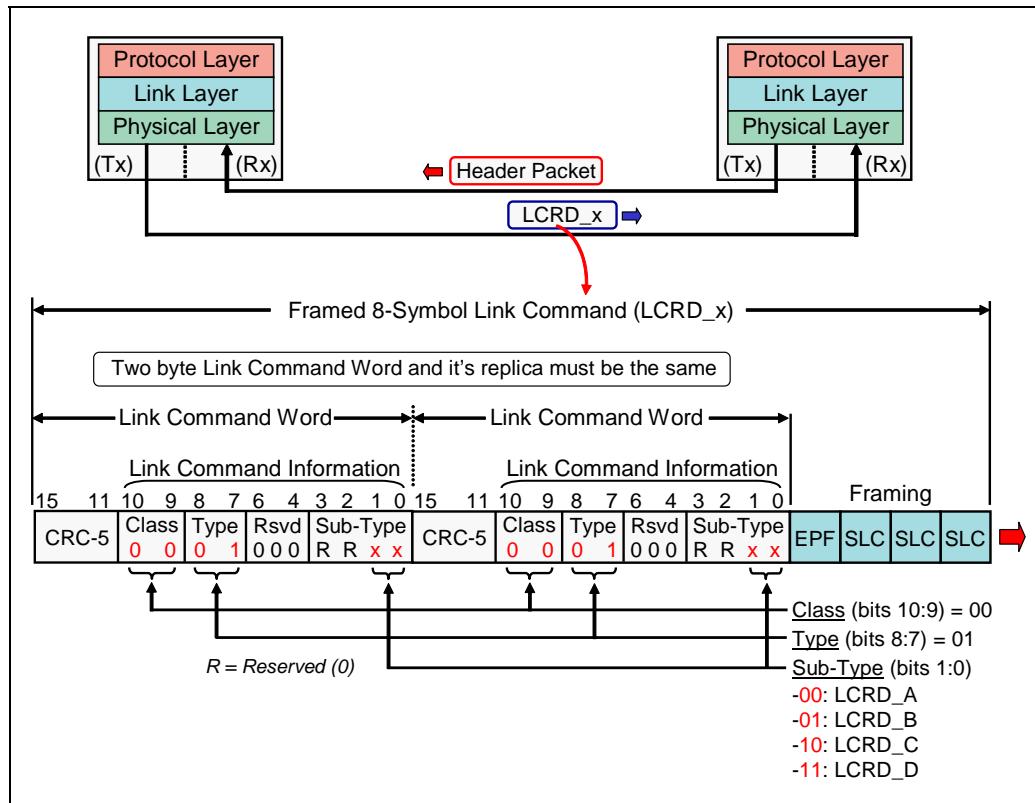


# USB 3.0 Technology

## Format Of LCRD\_x Link Command

Figure 20-25 on page 458 illustrates the format of an LCRD\_x link command used to advertise the starting link flow control credits. On entry into U0 from PowerOn or Warm Reset, four of these link commands are sent. Note that LCRD\_A uses Sub-Type = 0, LCRD\_B uses Sub-Type = 1, etc.

Figure 20-25: Format Of LCRD\_x Link Command



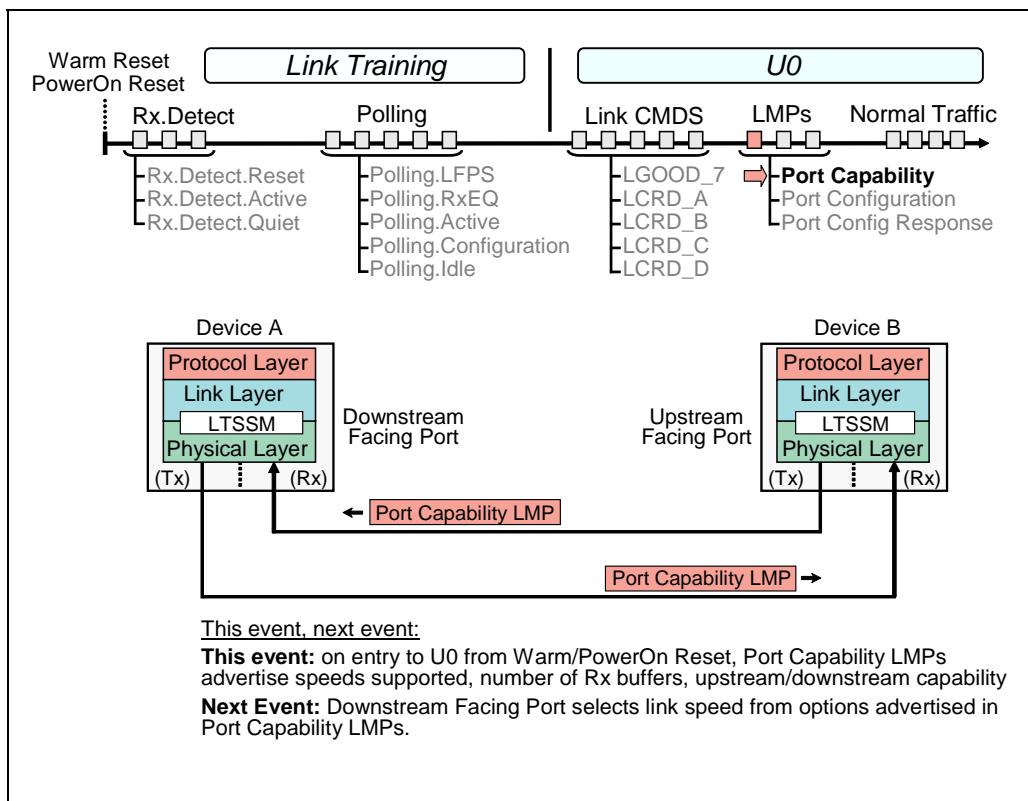
# Chapter 20: Link Training

## Port Capability LMP

At this point, all that remains is port initialization. This involves the exchange of Link Management Packets (LMPs), the first header packets seen on the link following a Warm Reset or PowerOn Reset. As indicated in Figure 20-26 on page 459, the first of this three-packet LMP series is the *Port Capability LMP*. Each device transmits this LMP to inform the link partner of three capabilities defined in the current specification:

- The number of link layer receive buffers implemented
- Speeds it supports (on the USB 3.0 SuperSpeed signal set)
- Whether it is capable of acting as a downstream facing port, upstream facing port, or both.

Figure 20-26: Devices Exchange First Header Packets, Port Capability LMP

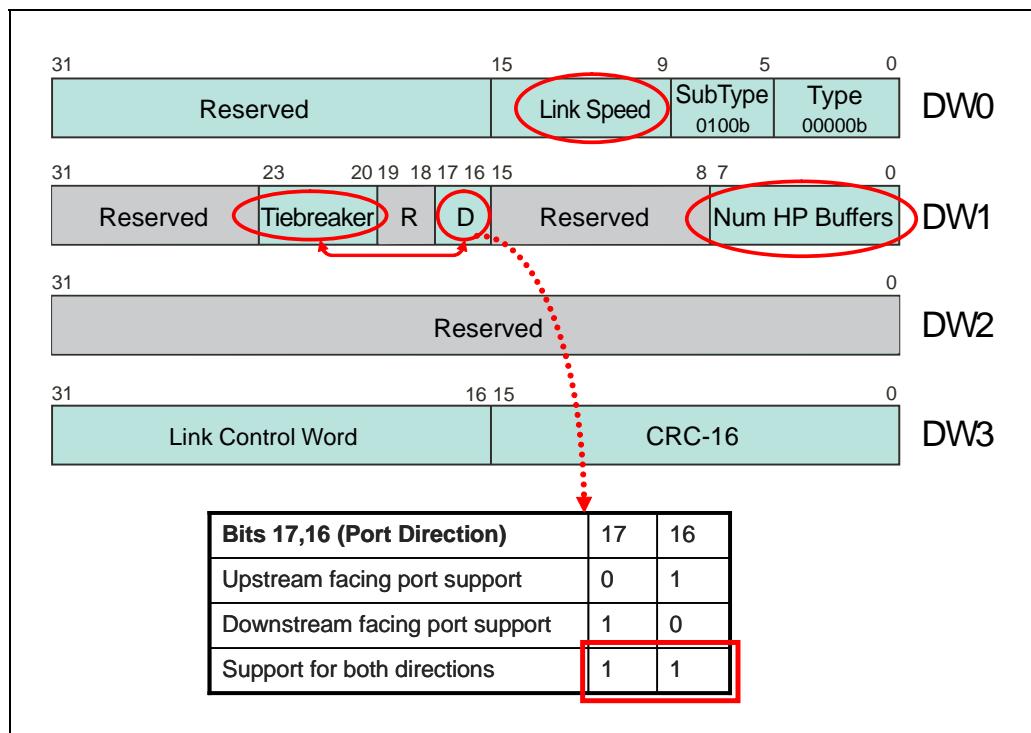


# USB 3.0 Technology

## Port Capability LMP Header Format

Figure 20-27 on page 460 illustrates the Port Capability LMP 16-byte (four dword) header format.

Figure 20-27: Port Capability LMP Header Format



### Link Speed Field (DW0, bits 15:9)

This field is a bit mask indicating all speeds the link partner is capable of on the SuperSpeed signal set. Currently, only 5 Gb/s is permitted, so all devices would return the first bit in this field (bit 9) set = 1. Bits 15:10 are reserved for future generations of SuperSpeed USB.

### Num HP Buffers Field (DW1, bits 7:0)

Here, each link partner reports the number of flow-controlled link layer header packet (HP) buffers implemented in hardware. The current specification requires exactly four buffers--which is the value reported in this field.

## **Chapter 20: Link Training**

---

### **D (Port Direction Support) Field (DW1, bits 17:16)**

Based on application, hardware designers decide whether USB 3.0 silicon will be capable of acting as downstream facing port, upstream facing port, or both. In many cases, a device always acts as a peripheral upstream facing port or as the downstream facing port of a host controller root hub or external hub. Because of applications such as USB On-The-Go and its dual role-devices, it is also acceptable to report capability to support either of the two roles.

Until the Port Capability, Port Configuration, and Port Configuration Response LMPs are exchanged, link training appears identical on either side of each link. Of course, when port configuration completes, there can only be one downstream and one upstream facing port.

Here are few rules concerning this field:

- If one device only reports downstream facing port capability (D = 10b), then the other must support upstream facing port capability (D=01b) or both (D=11b).
- Similarly, if one device only has upstream facing port capability (D = 10b), then the other must support downstream facing port capability (D=10b) or both (D=11b).
- In the event that both link partners report capability to act as either downstream OR upstream facing port (D=11b), this must be resolved before the exchange of Port Capability LMPs ends.

Referring again to Figure 20-27 on page 460 the *Tiebreaker* field is used to cover this case.

### **Tiebreaker Field (DW1, bits 23:20)**

Any device that is only capable of acting as upstream facing or downstream facing port (D = 01b or D=10b) is not required to implement the Tiebreaker field and the bits are treated as reserved (0). Devices that report support for either role must implement this field and include a random number here each time the Port Capability LMP is sent. If both devices are capable of either role, then the sender with highest random number in the Tiebreaker field “wins” and will thereafter act as downstream facing port; the other partner will assume upstream facing port responsibilities.

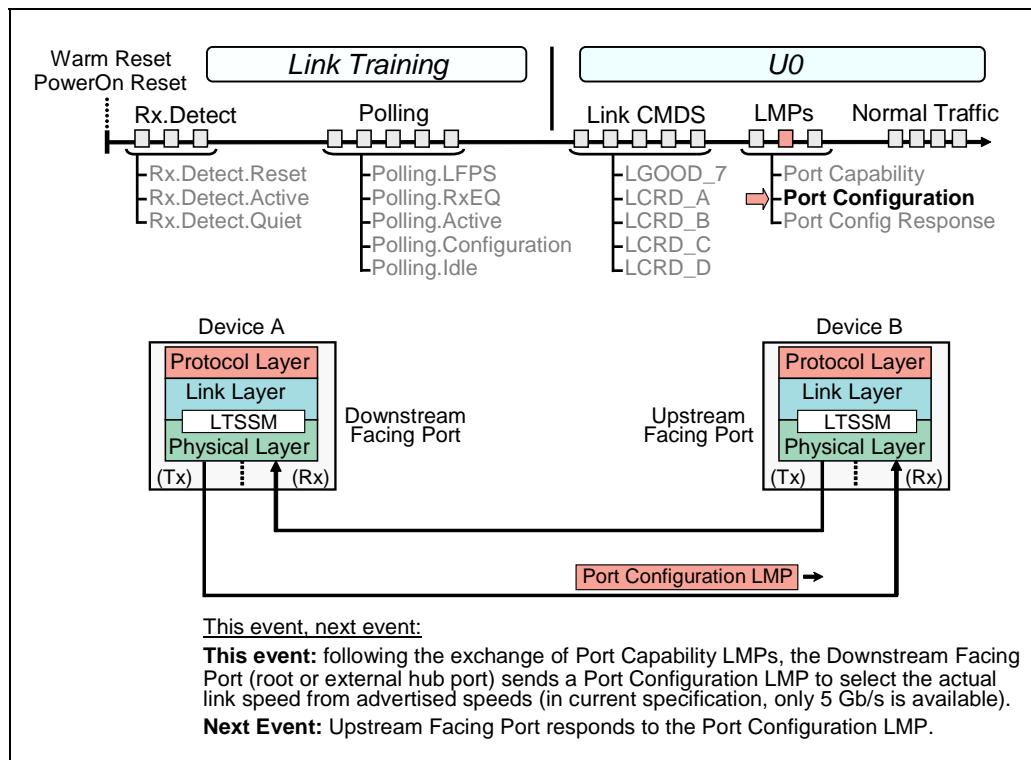
If, by chance, both partners are capable of either role and send the same random number in the Tiebreaker field, the Port Capability LMPs will be exchanged again (a new random number must be generated and sent each time).

# USB 3.0 Technology

## Port Configuration LMP

The second type of LMP employed during port initialization is only sent by the downstream facing port, as shown in Figure 20-28 on page 462. It is used to inform the upstream facing port of the speed option selected.

Figure 20-28: Downstream Facing Port Sends Port Configuration LMP

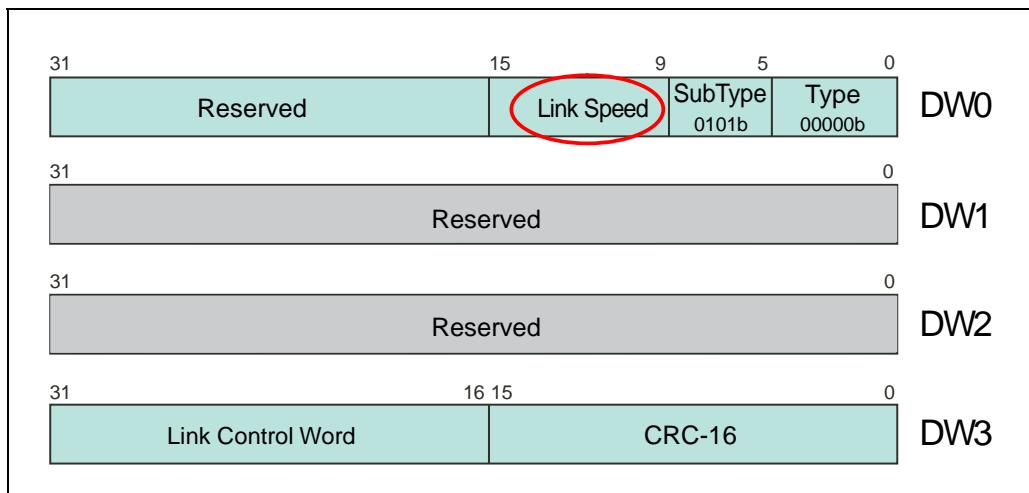


## Chapter 20: Link Training

### Port Configuration LMP Header Format

Figure 20-29 on page 463 illustrates the Port Configuration LMP 16-byte (four dword) header format.

Figure 20-29: Port Configuration LMP Header Format



#### **Selected Link Speed Field (DW0, bits 15:9)**

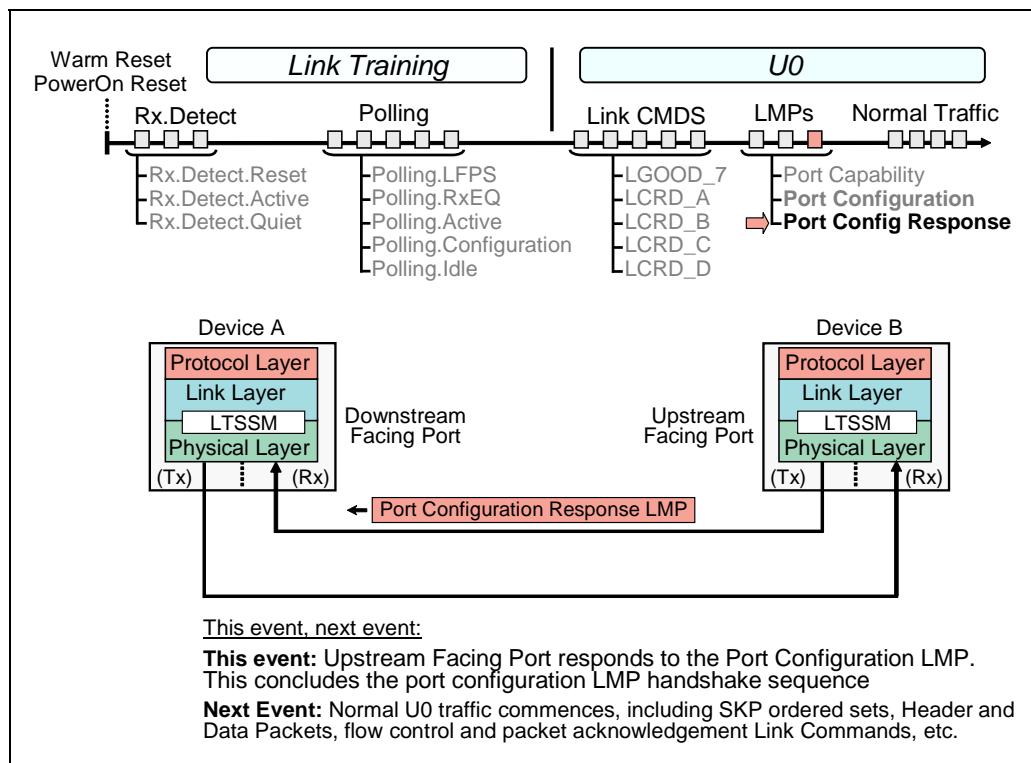
This field will have one bit set = 1 indicating the link speed to be used until the next link training event. As only 5 Gb/s is currently used, bit 9 = 1; bits 15:10 must be = 0 (0000001b).

# USB 3.0 Technology

## Port Configuration Response LMP

The third and final LMP employed during port initialization is only sent by the upstream facing port, as shown in Figure 20-30 on page 464. It is used to indicate acceptance of the speed selection chosen by the downstream facing port and acts as a handshake indicating that port initialization is complete.

Figure 20-30: Upstream Facing Port Sends Port Configuration Response LMP

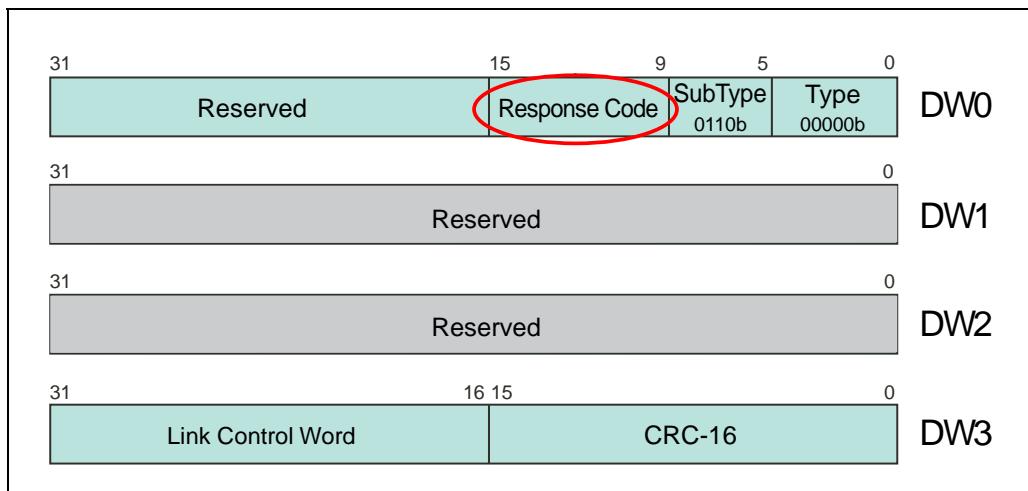


## Chapter 20: Link Training

### Port Configuration Response LMP Header Format

Figure 20-31 on page 465 illustrates the Port Configuration Response LMP 16-byte (four dword) header format.

Figure 20-31: Port Configuration Response LMP Header Format



#### Response Code Field (DW0, bits 15:9)

The upstream facing port uses this field to report compliance with the port speed request conveyed in the Port Configuration LMP sent by its downstream facing port link partner.

- A value of 1 in this field (0000001b) indicates acceptance
- Other values in this field are reserved and indicate an error condition. The error will be reported by the downstream facing (hub) port as a PORT\_CONFIGURATION\_ERROR when port status is checked.

# **USB 3.0 Technology**

---

---

## **Link Training Is Complete, What Now?**

---

### **General**

U0 is the active, functional state of 5 Gb/s SuperSpeed links. In the absence of other traffic, NOP (logical idle) symbols and clock compensation SKP ordered sets enable receivers to maintain proper elastic buffer levels and bit/symbol lock with transmitters.

---

### **Life In U0, A Few More Rules**

The following requirements apply while an LTSSM is in the U0 state. Note that there are slightly different requirements for downstream (DS) facing ports and upstream (US) facing ports.

- Ports must observe all U0 transmitter and receiver electrical specifications, including low impedance receiver termination,  $R_{RX-DC}$ .
- The 1 ms  $tU0RecoveryTimeout$  timer tracks the interval between consecutive inbound link commands. Timer restarts each time a Link Command is received.
- The 10  $\mu$ s  $tU0LTimeout$  timer tracks the bus idle interval between successive Link Commands sent to the link partner. Timer is reset on transmission of the first symbol of an outbound Link Command and starts counting down as the last symbol is sent and the link goes into logical idle.
- If there is no other link traffic and the 10  $\mu$ s  $tU0LTimeout$  timer (see the previous bullet) expires, downstream facing ports transmit an LDN Link Command; upstream facing ports transmit a LUP Link Command. These Link Commands reaffirm that the link is still functioning and the partners remain in U0.

---

---

# **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 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.

## USB 3.0 Technology

---

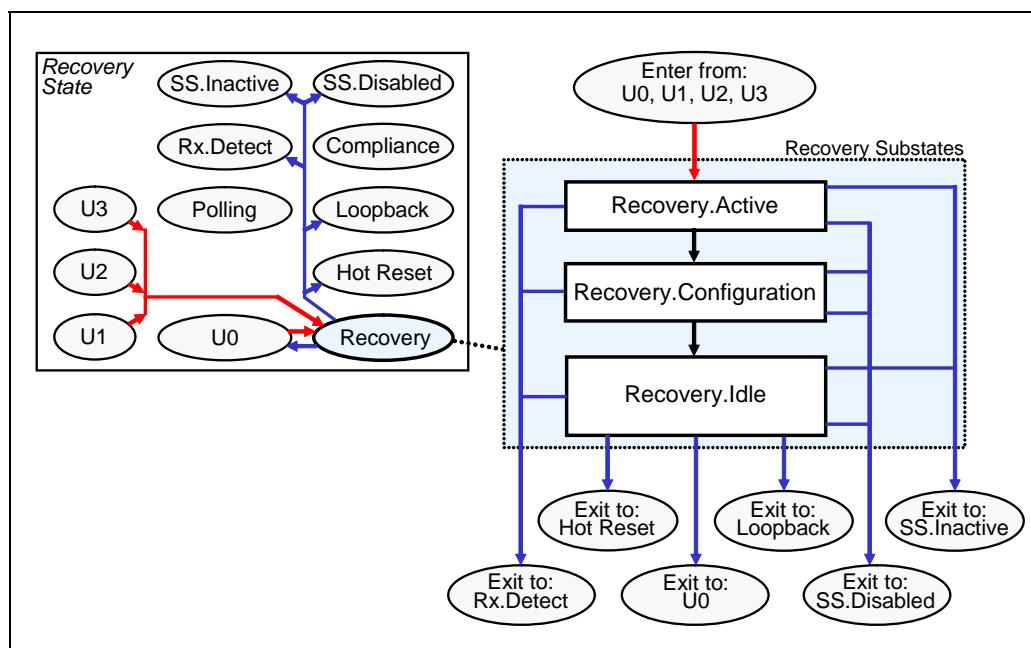
- Recovery transitions required to correct link layer or physical layer error conditions which can't be repaired while the U0 state. In this case, the transition to LTSSM Recovery is used to exchange training sequences, confirm receiver "bit lock" and "symbol lock", and re-initialize elastic buffer levels. 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



## **Chapter 21: Link Recovery and Retraining**

---

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.

# USB 3.0 Technology

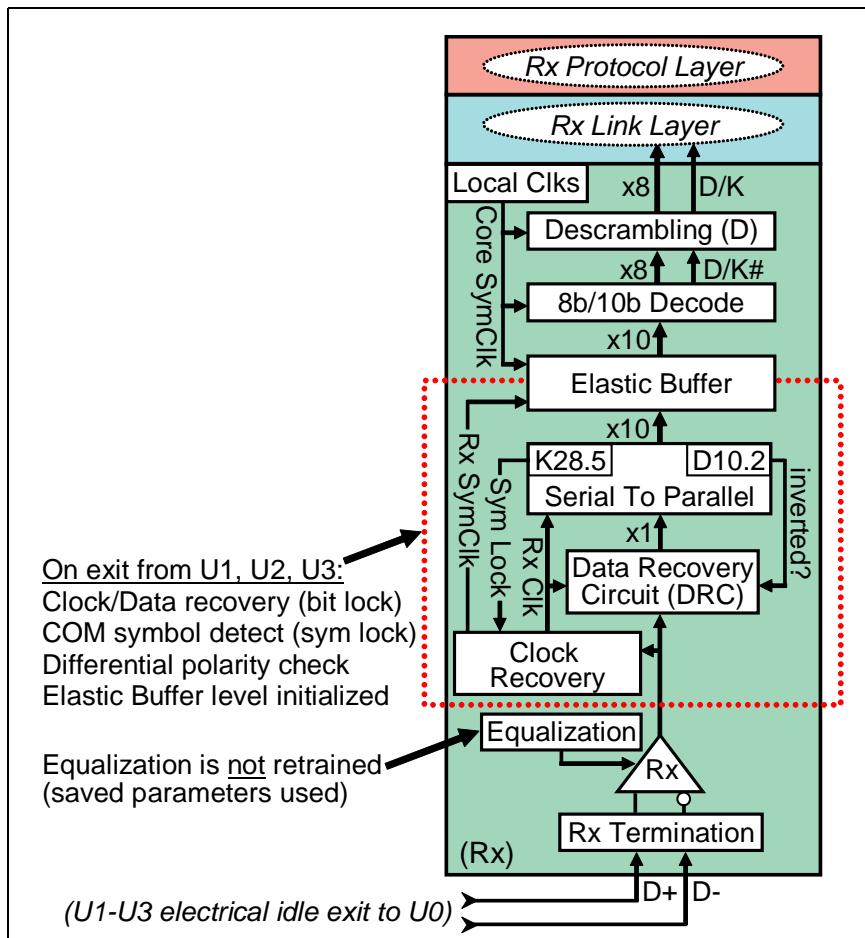
## Which Logic Is Retrained In Recovery?

On entry to the LTSSM Recovery state, link partners start exchanging Super-Speed ordered sets intended to accomplish two things:

- Alert the link partner that a transition to LTSSM Recovery is underway
- Start the process of retraining the receiver physical layer (in cases such as exits from U1-U3 electrical idle where the receive clock has been lost).

Figure 21-2 on page 470 depicts key receiver PHY logic to be retrained.

Figure 21-2: Receiver PHY Logic Retrained During Recovery



# Chapter 21: Link Recovery and Retraining

As indicated in Figure 21-2 on page 470, while receiver equalization is not retrained during link recovery, the following sections of the receiver PHY are:

- Clock Recovery
- Data Recovery Circuit (DRC)
- Serial To Parallel COM symbol and D10.2 Symbol detect
- Elastic Buffer

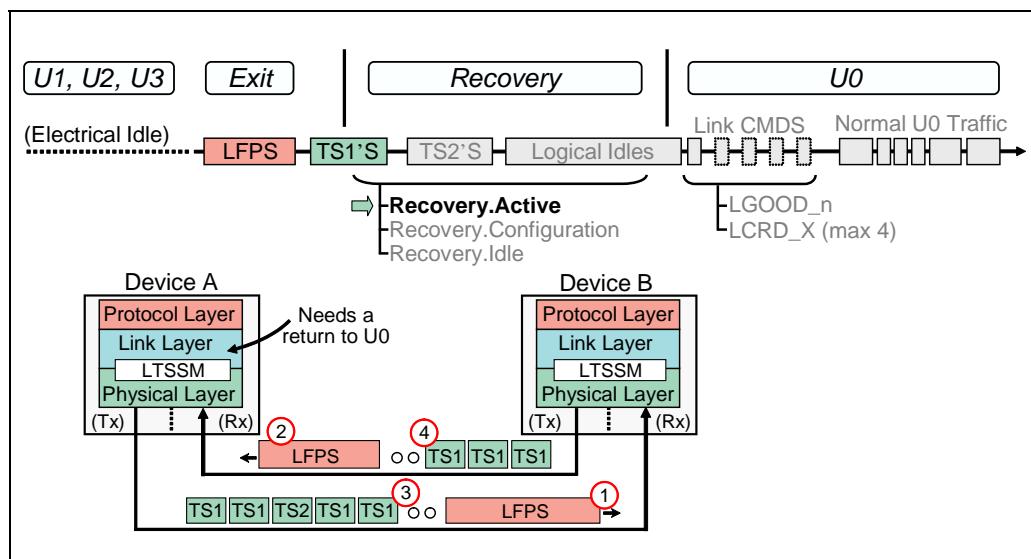
## Entering Recovery: Two Common Examples

As with other LTSSM state transitions, the link partner handshake for coordinating entry into the LTSSM Recovery state depends on the previous state of the link. The following two sections describe differences in entering the LTSSM Recovery state starting from link electrical idle on an exit from the U1-U3 power management states vs. entry from U0 due to an uncorrectable link error.

## U1-U3 Exit And Transition To Recovery

As depicted in Figure 21-3 on page 471, entry into LTSSM Recovery state from U1-U3 power management states is a two step process. Both are described below.

Figure 21-3: U1-U3 Exit Followed By A Transition To LTSSM Recovery



## **USB 3.0 Technology**

---

In the example shown above, Device A requires that the link return to U0 from U1, U2, or U3 power management states. The general sequence of events follows and numbers in this list refer to the circled numbers in Figure 21-3 on page 471.

1. The first event in the transition from U1-U3 to LTSSM Recovery occurs when Device A enables its transmitter and drives U1 Exit, U2 Exit, or U3 Exit LFPS to its link partner. The duration of the burst ranges from 300nS (U1 Exit) to as much as 10mS (U3 Exit).
2. Receivers are required to keep LFPS detect circuitry active when the link is in any of the power management states. Device B will detect the incoming LFPS burst and verify by the duration that it is a power management exit and not a Warm Reset. Note that a Warm Reset takes precedence over all other LTSSM state transitions and Warm Reset LFPS is easily recognized as it has a duration of 80-120 ms. For this example, assume U1, U2, or U3 Exit LFPS signaling is confirmed; Device B enables its transmitter and returns the same LFPS Ux Exit burst.
3. After the initiator (Device A) detects the LFPS has been returned by its link partner, it ceases LFPS and commences sending TS1 (Training Sequence 1) ordered sets at 5 Gb/s SuperSpeed. At this point, the initiator is in the LTSSM Recovery.Active substate and waiting for the link partner to make the same transition.
4. When the link partner (Device B) also stops sending LFPS and starts transmitting TS1 ordered sets, it too has transitioned to LTSSM Recovery.Active substate.

---

## **U0 Link Layer Error And Transition To Recovery**

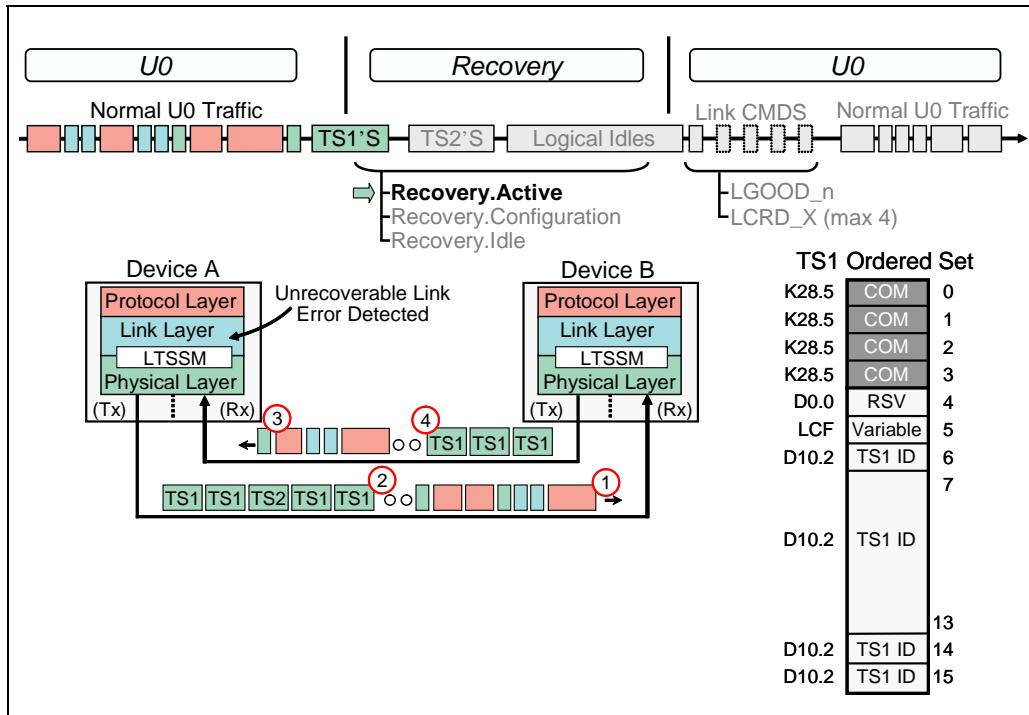
The second common reason for transitions into the LTSSM Recovery state are related to link layer errors. In the example shown in Figure 21-4 on page 473, Device A has detected a link layer error that can't be corrected using the preferred header packet retry mechanism. Examples of link layer errors that require such LTSSM Recovery transitions include the following:

- A timeout of 10uS without receipt of any link commands
- A header packet is received out of order (Hdr Seq# is not sequential)
- A timeout of the Credit\_HP\_Timer at the transmitter while waiting for a LCRD\_x link command awarding a flow control credit.
- A timeout of the transmitter's Pending\_HP\_Timer waiting for an LGOOD\_n or LBAD link command response to a header packet sent earlier.

The general sequence of events for entering LTSSM Recovery from U0 in such cases is depicted Figure 21-4 on page 473 and described below.

## Chapter 21: Link Recovery and Retraining

*Figure 21-4: U0 Link Layer Error Causes A Transition To LTSSM Recovery*



In the example shown above, Device A detects an unrecoverable link layer error and initiates a transition to LTSSM Recovery to correct the problem (retrain the link). If successful, the link will quickly return to the U0 operational state. Numbers in this list refer to the circled numbers in Figure 21-4 on page 473.

1. Device A is sending and receiving normal U0 link traffic, including header packets, data packets, link commands, skip ordered sets, and logical idles.
2. At some point, Device A detects an uncorrectable error associated with either its link layer transmit or receive interface. At this point, its link layer stops processing new inbound and outbound packets. Device A sends TS1 ordered sets to the link partner and transitions to LTSSM Recovery state.
3. As indicated in Figure 21-4 on page 473, Device B is also processing normal U0 traffic when it detects the TS1 ordered sets from Device A.
4. When Device B recognizes a transition to LTSSM Recovery is underway, it also transitions to LTSSM Recovery and returns TS1 ordered sets. (Note that one device usually transitions a bit more quickly to a new LTSSM state than its link partner).

# USB 3.0 Technology

## The Recovery Substate Events

Regardless of the reason for entering LTSSM Recovery, the events that occur are similar. The following section describes the events associated with each of the three LTSSM recovery substates. This example assumes that the reason for entering LTSSM Recovery was related to a power management exit or link error, and not associated with the LTSSM Recovery state role in Hot Reset, entering Loopback mode, or disabling scrambling.

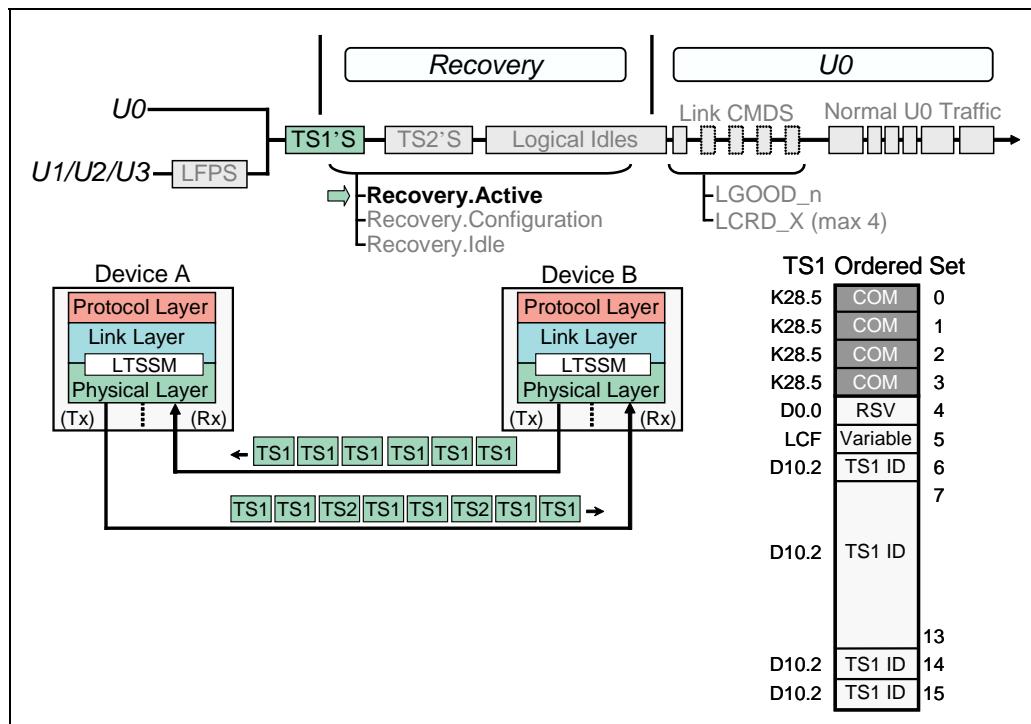
### Recovery.Active

In this state, devices exchange 16-byte TS1 ordered sets. Because ports train at different rates, one generally transitions to the next substate (Recovery.Configuration) before the other.

#### Other Requirements In Recovery.Active

On entry to the Recovery.Active substate, devices start a 12 ms timer.

Figure 21-5: Recovery.Active Employs TS1 Ordered Sets



# Chapter 21: Link Recovery and Retraining

## Recovery.Active And The TS1 Ordered Set

TS1 (Training Sequence 1) can be seen at the right in Figure 21-5 on page 474. When received, it informs the link partner that the attached device has transitioned to Recovery.Active to start link retraining. As shown above, TS1s include D10.2 and K28.5 (COM) used by receivers to reacquire bit and symbol lock.

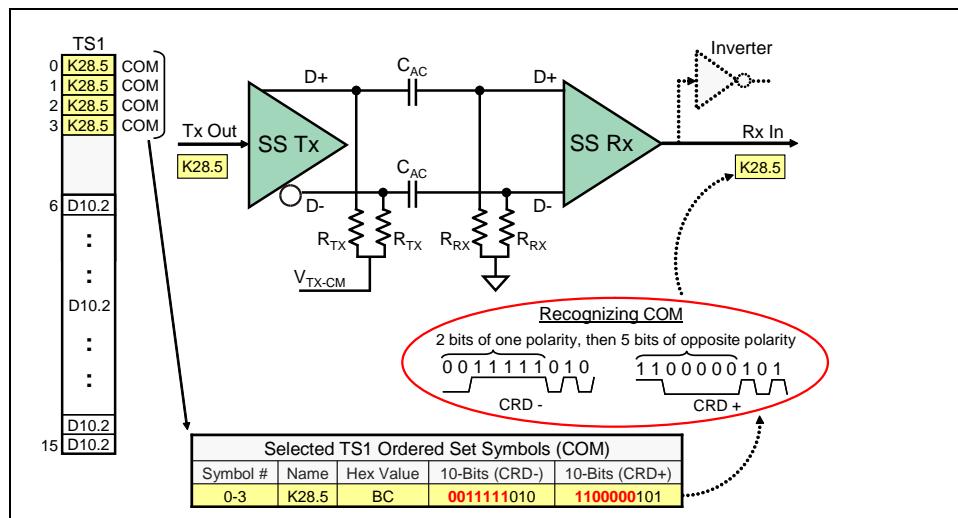
## Two Special Symbols In TS1 Ordered Sets

Two of the 16-byte TS1 ordered set symbols have special uses in link retraining:

- K28.5 (COM)
- D10.2

**Symbol K28.5 (COM).** The COM (Comma) is repeated four times at the start of each 16-byte TS1 ordered set. It is the only 10-bit symbol that occurs during link training with a bit pattern that includes 2 bits of one polarity, followed by 5 bits of the opposite polarity. Encoding of other USB 3.0 SuperSpeed symbols assures this special “2 and 5” pattern doesn’t occur because of bit combinations resulting from two neighboring symbols. The uniqueness of the COM symbol makes it easily recognizable by the receiver. During link retraining (and link training), the receiver samples the incoming bit stream in search of the COM “2 and 5” bit pattern. Once COM is detected, symbol lock has been achieved and every ten bits thereafter is a new symbol. Because each TS1 ordered set has four COM symbols and the TS1s are sent repeatedly during link retraining, the receiver has plenty of opportunities to recover the transmitter clock, sample the TS1 bit stream, recognize COM, and achieve symbol lock.

Figure 21-6: COM Symbol Bit Pattern Is Easily Recognized By Receiver



## USB 3.0 Technology

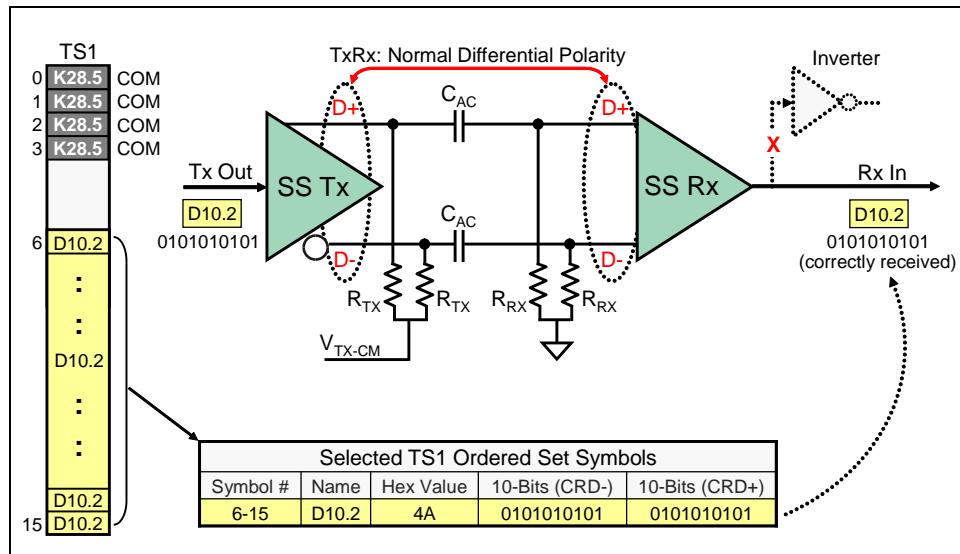
Figure 21-6 on page 475 depicts the transmission and reception of the COM symbol. As with all 10-bit symbols, there are two versions of COM. The version of COM that is sent depends on the *current running disparity* (CRD) for bits sent previously; note that both versions of COM contain the special bit pattern of two bits of one polarity followed by five bits of the other ("2 and 5").

**Symbol 6-15 D10.2.** The last 8 symbols of each 16-byte TS1 ordered set are D10.2. During LTSSM Recovery and link retraining, this symbol has an important use.

The TS1 D10.2 symbols are used to enable the receiver to perform the required check during link training for differential polarity inversion, and correct it if necessary. A differential transmitter employs two output pins (Tx D+/D-). Differential receivers have two input pins (Rx D+/D-). The expectation is that for each pair of link partners:

- Transmitter D+ is connected to receiver D+
- Transmitter D- is connected to receiver D-

Figure 21-7: Normal Transmitter And Receiver Differential Polarity



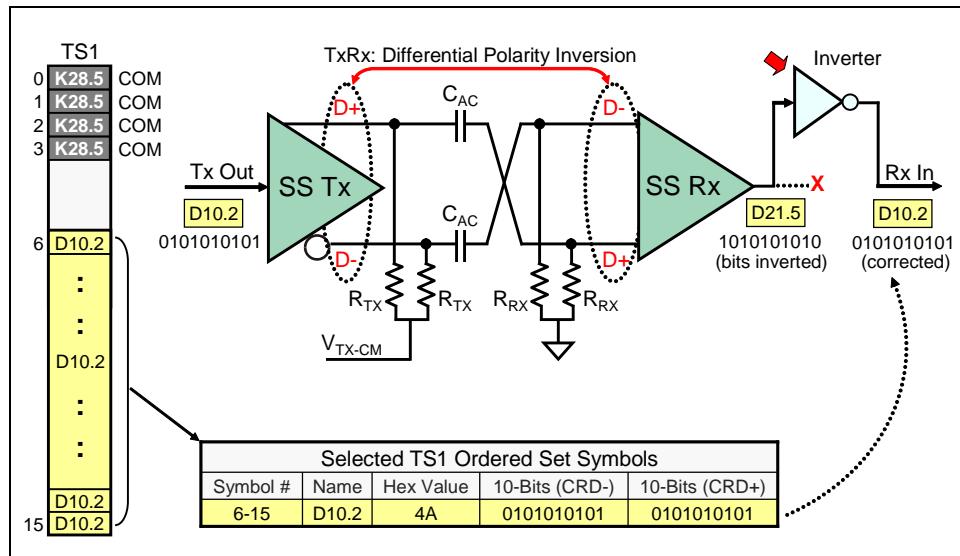
## Chapter 21: Link Recovery and Retraining

Figure 21-7 on page 476 depicts a SuperSpeed transmitter and receiver connected with normal (correct) Tx/Rx differential polarity. Note in the illustration that the D10.2 symbols are received as sent.

In some cases, device pinout, pc board trace routing, or connector layout results in a swap of channel differential signal pairs in the path from transmitter to receiver. During link retraining (and training), if the D10.2 symbols (TS1 symbols 6-15) are received as D21.5, differential polarity is inverted and the receiver activates an internal inverter to correct it.

Figure 21-8 on page 477 illustrates the case where a SuperSpeed transmitter and receiver are connected with inverted differential signal polarity. Notice that the D10.2 symbol sent by the transmitter is received as D21.5, then internally corrected by enabling the inverter shown in at the right of the illustration.

Figure 21-8: Differential Polarity Inversion Corrected By The Receiver



### Normal Exit: Recovery.Active To Recovery.Configuration

The normal exit from the Recovery.Active substate is to Recovery.Configuration. This exit occurs when a port completes retraining internally and has received 8 consecutive, identical TS1s or TS2s.

# USB 3.0 Technology

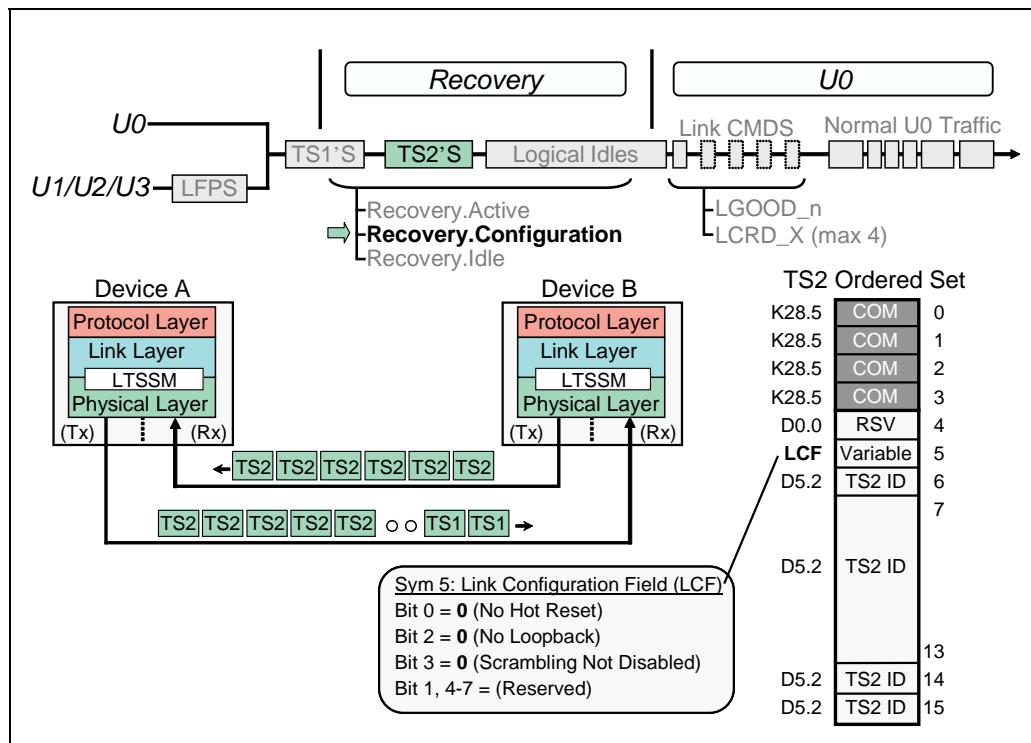
## Other Exits From Recovery.Active

For a discussion of other cases of exits from this substate, refer to Chapter 12, entitled "LTSSM And the SuperSpeed Link States," on page 243.

## Recovery.Configuration

As shown in Figure 21-9 on page 478, devices in Recovery.Configuration exchange TS2 ordered sets. While not shown in this example, Recovery.Configuration is also an opportunity for to use the TS2 ordered sets to indicate that a transition to Hot Reset or Loopback is requested.

Figure 21-9: Recovery.Configuration Employs TS2 Ordered Sets



## Recovery.Configuration And The TS2 Ordered Set

TS2 is Training Sequence 2. Figure 21-9 depicts the 16-byte TS2 ordered set. The Link Configuration Field (LCF in symbol 5) selects options: Hot Reset, Loopback, Scrambling Disable. If no LCF bits are set =1, a return to U0 (with scrambling enabled) is requested.

# Chapter 21: Link Recovery and Retraining

## Normal Exit: From Recovery.Configuration To Recovery.Idle

The normal exit from the Recovery.Configuration substate is to Recovery.Idle. This exit occurs only if the TS2 Link Configuration Field (in symbol 5) did not indicate that a transition to Hot Reset or Loopback was requested.

The port makes the transition to Recovery.Idle when:

- 8 consecutive and identical TS2s were received (with no Loopback or Hot Reset indication)
- 16 TS2s were sent after receiving 8 consecutive, identical TS2s.

## Other Exits From Recovery.Configuration

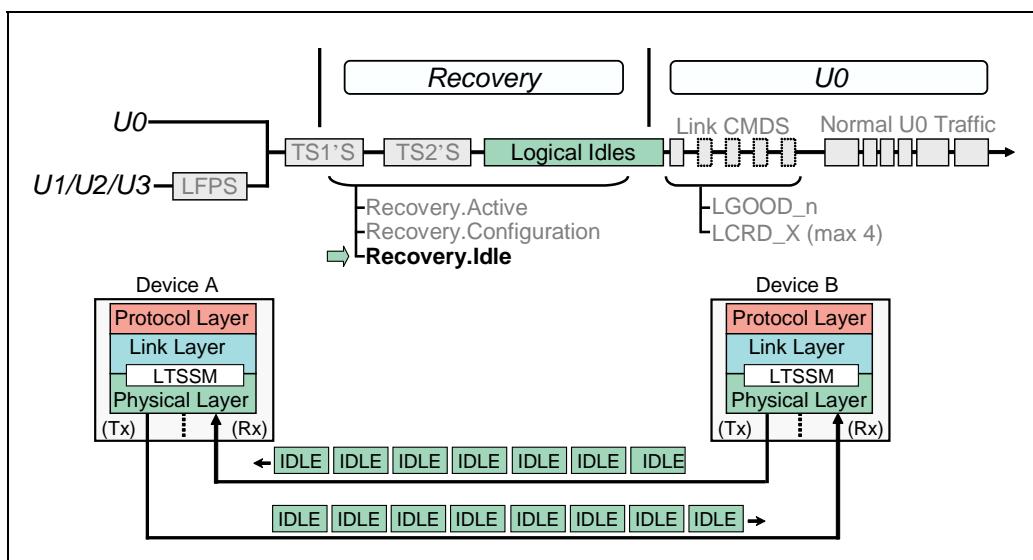
For all of the exit possibilities, refer to Chapter 12, entitled "LTSSM And the SuperSpeed Link States," on page 243.

## Recovery.Idle

### General

In this state, ports have already decoded TS2s received in Recovery.Configuration and will exchange logical idle (D0.0) symbols if they are about to enter U0. If scrambling disable was requested in TS2s, it is applied now.

Figure 21-10: Logical Idle Symbols Confirm That A Return To U0 Is Next



# **USB 3.0 Technology**

---

## **Other Requirements In Recovery.Idle**

On entry to the Recovery.Idle substate, the link is about to exit LTSSM recovery and resume normal SuperSpeed traffic. In preparation for entry into U0, the following rules apply:

- Downstream facing ports reset their link error counters (LECs)
- All ports prepare to exchange Header Sequence Number advertisements and Flow Control Credit advertisements.

The normal exit from the Recovery.Idle substate is to U0. The port makes the transition to U0 when:

- 8 Idle symbols are received and
- 16 Idle symbols have been sent after receiving at least 1 Idle symbol

---

## **LGOOD\_n Advertisement**

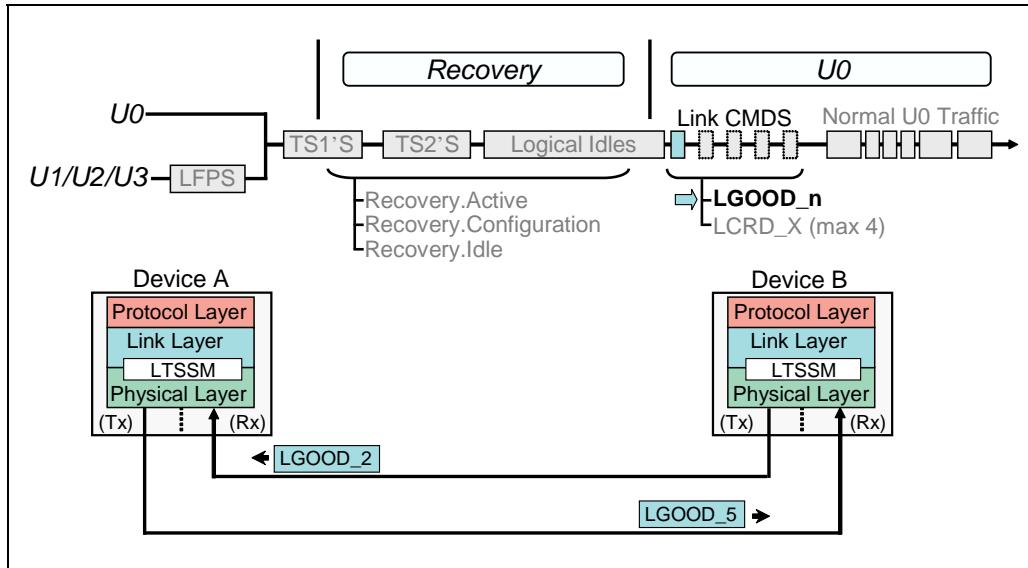
On re-entry into U0 from LTSSM Recovery, link partners exchange the starting header packet sequence number (Hdr Seq#) for each link direction; this link sequence number has a range of 0-7 and is conveyed using the LGOOD\_n link command. The value sent indicates the last valid packet acknowledged by the receiver before the transition to link recovery occurred. The link layer transmit interface uses the value to confirm that all acknowledged packets have been flushed from the retry buffer and that header packets not yet acknowledged are re-sent.

## **A Simple Example**

In Figure 21-11 on page 481, Device A is advertising a starting header packet sequence number of five by sending LGOOD\_5 to the link partner. This indicates that the first header packet sent by Device B after the return to U0 should be Hdr Seq#6. Device B is sending LGOOD\_2; the first header packet it expects to receive should carry Hdr Seq#=3.

## Chapter 21: Link Recovery and Retraining

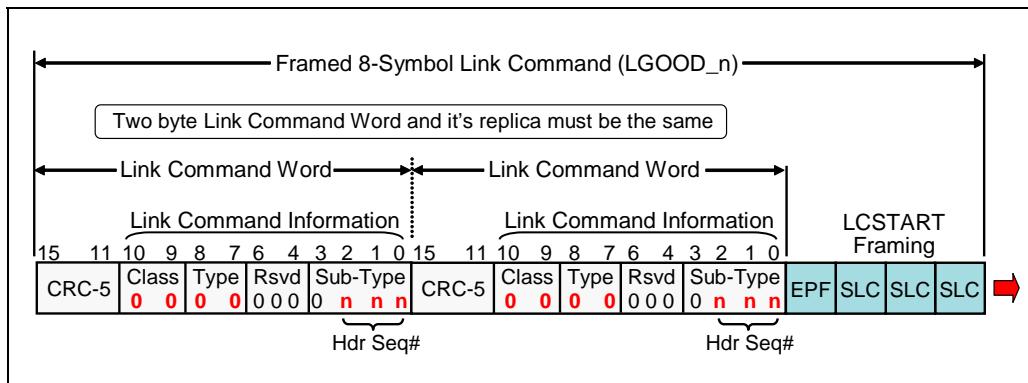
Figure 21-11: LGOOD\_n Link Command Advertises Header Packet Sequence Number



### Format Of LGOOD\_n Link Command

Figure 21-12 on page 481 illustrates the format of an LGOOD\_n link command used to advertise the starting header packet sequence number.

Figure 21-12: Format Of LGOOD\_n Link Command



# USB 3.0 Technology

---

## LCRD\_x Advertisement

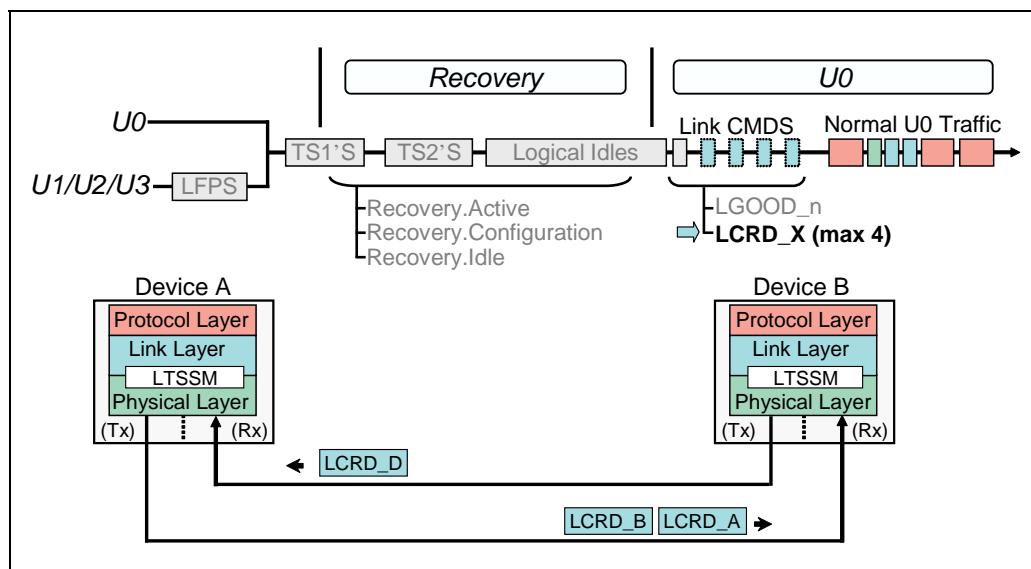
### General

The other link command advertisement required on entry to U0 is for flow control credits. There are four link layer receive header packet buffers (referred to as Rx HP Buffers A-D) and any number may have been in use when the transition to LTSSM Recovery occurred. During the credit advertisement, link partners send one flow control credit link command, LCRD\_x, for each available link layer receive buffer. Buffers are always used in order A-D and the LCRD\_x link commands must also be sent in order (if more than one buffer is free).

### A Simple Example

In the example shown in Figure 21-13 on page 482, Device A had two available header packet buffers (A,B) available when the transition to LTSSM Recovery occurred. Device B had only one header packet buffer (D) available when the transition to LTSSM Recovery occurred. As shown in Figure 21-13, Device A sends two credits using LCRD\_A, LCRD\_B; Device B sends one credit using LCRD\_D. Note that the exchange of header packet sequence numbers and flow control credits completes the link recovery process and normal U0 traffic resumes.

Figure 21-13: LCRD\_x Link Commands Initialize Header Packet Flow Control



## **Chapter 21: Link Recovery and Retraining**

---

### **For More Details On The LTSSM Recovery**

Refer to Chapter 12, entitled "LTSSM And the SuperSpeed Link States," on page 243 for more details on LTSSM Recovery, including:

- Substate requirements
- Time-outs
- Alternative exits

## **USB 3.0 Technology**

---

---

# 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 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

# **USB 3.0 Technology**

---

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 transferred 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 100000011B	GetStatus (00h)	Zero	Zero Interface Endpoint Other	Two	Device Interface, Endpoint Status or Other (port)
00000000B 00000001B 00000010B	ClearFeature (01h)	Feature Selector	Zero Interface Endpoint	Zero	None

## Chapter 22: Device Configuration

---

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

# **USB 3.0 Technology**

---

## **Device Detection and Reporting**

This section describes the discovery process used to detect each device (including hubs) and specifies the actions for reporting the event to host software. Hub client software plays a pivotal role in this process. Root or external hub ports detect the presence of each device (via hub client software), which triggers the configuration process that proceeds one port at a time. The primary events associated with device detection are summarized below and detailed in subsequent sections:

1. The hub client software applies power to the port (if not already powered), causing the link partners to enter their reset state.
2. When a device exits reset it attempts to detect the presence of its link partner and the hub port performs the same action.
3. When the link partners detect each other's presence several things occur:
  - The Link Training and Initialization process is triggered.
  - The hub port sets port status and the change indicator bits.
  - Next, the status change indicator (Endpoint 1) within the hub sets a bit associated with the port causing the status change.
4. When Link Training and Initialization completes the link enters the U0 state and Link Flow Control is initialized, thereby permitting header packets to be transmitted.
5. Having detected device attachment, the first header packet transferred may be an ERDY (or some other mechanism for root ports) to notify the host that Endpoint 1 is ready has incurred an event and should be read.
6. The hub client driver performs an IN transaction to read Endpoint 1 (the hub Status Change indicator) and detects that a port has detected some event change.
7. The hub client driver performs a GetPortStatus request to hub, this time reading the contents of the port status and change indicator bits and detecting device attachment.
8. Having detected the device, the hub client notifies configuration software that a device is ready to be configured.

The host controller driver software creates an abstraction of the root hub so that the standard USB hub client driver can make requests to the host controller's software driver. Because of this abstraction, the same process is described for both the root hub and external USB hubs. The actual mechanisms used to access status information from the root port involves accessing memory-mapped registers within the host controller.

## Chapter 22: Device Configuration

---

### Device Detection Process — Details

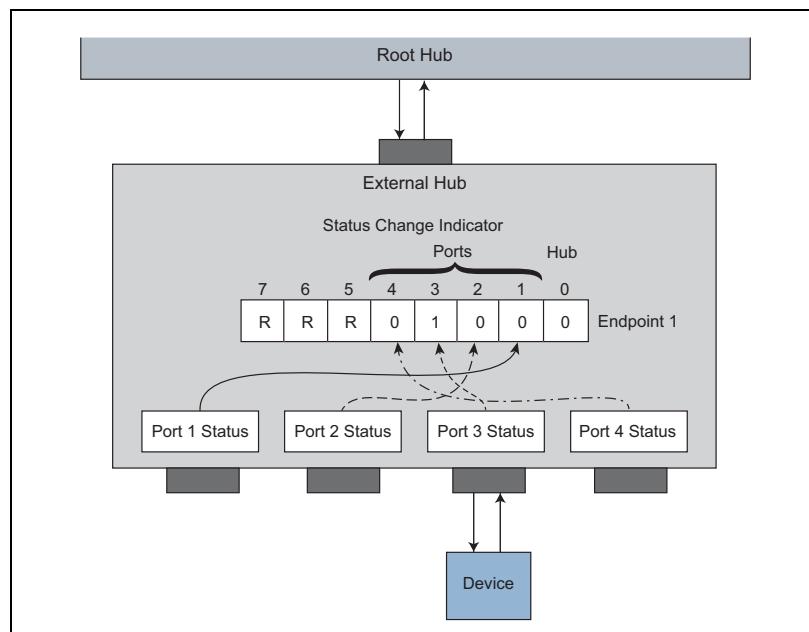
Device detection begins with the application of  $V_{BUS}$  from a downstream root port or an external downstream hub port. When power is applied both link partners enter the Reset state, and the receivers apply hi-impedance terminations of 25 K $\Omega$ s or greater. When reset is removed low-impedance terminations in the range of 18 to 30  $\Omega$ s are applied. This causes entry to the LTSSM's RxDetect state. The section entitled, "The Detection Method" on page 439 describes the mechanism used by the link partners to detect device presence.

Having detected that a device is attached, Link Training begins and the downstream hub port (root or external port) sets port status bits along with the hub's Status Change Indicator (EP1 for external hubs) thus indicating one or more ports have incurred a status change.

### Status Change Indicator (Endpoint 1)

Figure 22-1 illustrates the hub's Status Change Indicator. In this example, a device is detected at Port 3, port status has been updated and bit 3 of Endpoint 1 has been set. Next, the change indicator event must be communicated to the host.

*Figure 22-1: Hub Status Change Indicator*



## **USB 3.0 Technology**

---

The notification method depends on whether the device is attached to a root port or external hub port as described below:

- Root Hub Port Attachment — The change notification can cause the Host Controller to generate an interrupt to call the Host Controller driver to read the Status Change Indicator.
- External Hub Port Attachment — The change notification can cause the hub to deliver an ERDY header to the host controller, which in turn sends an interrupt that calls the hub client driver to initiate a read from Endpoint 1.

Hubs implement this interrupt endpoint to track status change events that occur within the hub and hub ports. The format of the Status Change Endpoint is shown with Bit 0 reserved for global hub events and bits 1-n to identify the port numbers. In the example, four ports are implemented so bits five through seven are reserved. Notice also that port 3 has a device attached and bit 3 of the status change endpoint has been set by hardware.

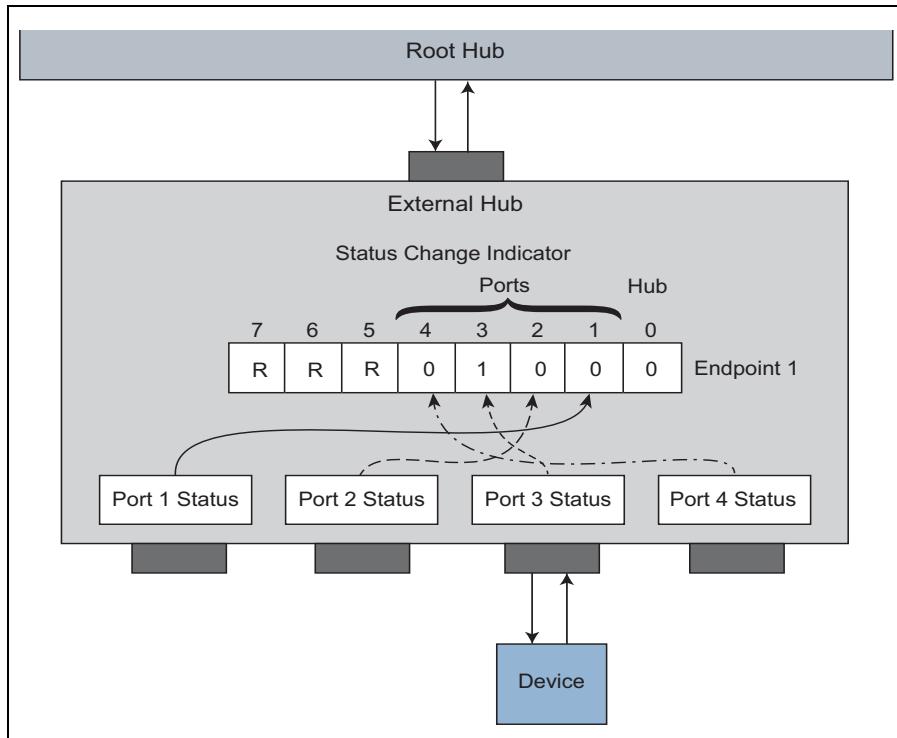
Configuration software is aware of the number of ports supported by each hub (15 ports maximum), and therefore is aware of the size of the bitmap returned when the hub's status change endpoint is accessed. Status is reported in byte-sized fields with zeros returned in the bit fields corresponding to ports that are not implemented and that have not had a status change.

When software accesses the status change endpoint, the bitmap illustrated in Table 22-2 on page 492 is returned only if a status change has occurred. If no status information is present, the hub returns NRDY during the IN transaction. When an event does occur a status bit is set and the hub sends ERDY to notify software that the endpoints needs servicing.

## Chapter 22: Device Configuration

---

Figure 22-2: Hub Status Change Indicator



---

### Hub Port Status and Change Indicators

When hub client software detects one or more ports have incurred an event, software fetches port status information to determine what specific event or events have occurred. Fetching port status is done via the GetPortStatus request that returns the contents of the Port Status and Change Indicators. In the previous example, a device was connected to port 3 resulting in changes to the Port Status and Change Indicator represented in Table 22-2. The changes include:

- Port Change Indicator
  - Bit 0; Connect Change = 1
- Port Status
  - Bit 0; Current Connect Status = 1
  - Bits 12:10; Negotiated Speed = 000b (5Gbs)

Note: The change indicator bits prevent software from having to store previous status in order to detect which bit has changed.

## **USB 3.0 Technology**

---

Table 22-3 lists and described each of the Port Change Indicators. These bits are normally cleared to zero and are set when a status change has occurred. After performing a GetPortStatus request, the hub client driver must clear the change indicator bits back to zero. This ensures that a subsequent event will be detected. (See page 496 for details regarding the Clear Change Indicator request.)

*Table 22-2: Fields Returned During GetPortStatus*

Port Change Indicators																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved	Reserved	Reserved	Reserved: all zeros returned on read								Port Config Error	Port Link State Change	BH Port Reset Change	Port Reset Change	Over-Current Indicator Change	Reserved
Port Status																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved	Reserved	Reserved	Negotiated Speed: 000 = 5Gb/s 001 = Resrvd 002 = Resrvd 003 = Resrvd 004 = Resrvd 005 = Resrvd 006 = Resrvd 007 = Resrvd				Port Power				Port Link State				Port Reset	Over-Current Indicator
															Reserved	Port Enabled/Disabled
															Current Connect Status = 1	Connect Change = 1

## Chapter 22: Device Configuration

---

Table 22-3: Port Change Indicator Definition

Bit	Name	Description
0	Connect Change	This field reflects a change in the current connection state of the port. When set to one Connection Status has changed.
2:1	Reserved	
3	Port Over-current Change	This field applies only to hub ports that report over-current conditions on an individual port basis. When set to one Over-current status has changed.
4	Port Reset Change	When set to one, this bit indicates that a reset on this port has completed.
5	BH Reset Change	When set to one this bit indicates that a warm reset has completed.
6	Port Link State Change	When set to one this field indicates a change in the Port Link Status.
7	Port Config Error	This bit is set to one when the link partners are not successful in completing the Port Configuration process. This process is associated with the Link Management Packet (LMP) exchange that occurs after Flow Control Initialization.
15:8	Reserved	

Table 22-4 lists and describes the status bits that are also returned during the Get Port Status request.

## **USB 3.0 Technology**

---

*Table 22-4: Port Status Definition*

Bit	Name	Description
0	Current Connect Status	This field reflects the current connection state of the port. 0=No device attached 1=Device attached
1	Port Enabled/Disabled	Indicates whether the port is currently enabled. A port can be disabled due to fault conditions or system software 0 = Port disabled 1 = Port enabled
2	Reserved	
3	Port Over-current	Indicates that this port has incurred an over-current condition. This bit remains set until the over-current condition is removed. This bit is required for all self-power hubs. 0 = No over-current condition 1 = Over-current condition exists
4	Port Reset	Indicates that the port is reset due to a Warm or Hot Reset. 0 = No port reset 1 = Port reset is active
8:5	Port Link State	Specifies the current state of the link attached to this port. After the current state is complete a transition to the next state can be reported. 00h = U0 state 01h = U1 state 02h = U2 state 03h = U3 state 04h = SS.Disabled State 05h = Rx.Detect State 06h = SS.Inactive State 07h = Polling State 08h = Recovery State 09h = Hot Reset State 0Ah = Compliance Mode State 0Bh = Loopback State
9	Port Power	Reflects this port's power control state. Hubs can implement different power switching methods and may not represent whether power is actually applied or not. 0 = Port is in Powered-off state 1 = Port not in Powered-off state

## Chapter 22: Device Configuration

---

Table 22-4: Port Status Definition (Continued)

Bit	Name	Description
12:10	Negotiated Speed	This value indicates the highest common speed that the link partners support. This is negotiated during U0 via the Port Configuration LMP handshake 0 = 5 Gbps 1-7 = Reserved
15:13	Reserved	

---

### Get Port Status Request

The hub client uses the default control pipe to perform a GetPortStatus request to the hub. In this case the control transfer consists of:

- The Setup Stage — This setup transaction delivers 8 bytes of data that specify the GetPortStatus request.
- The Data Stage — This IN transaction is used to return port status to the hub client.
- The Status Stage — This Status transaction sent by the host controller verifies that the operation was successful, with the return of ACK.

Table 22-5 on page 496 defines the 8 bytes of data that are sent during the setup transaction. This table is used as reference information. Subsequent control transfers are compressed into a single row. The first byte (Request Type) is a bit-map that is described in Table 22-5. The other field values are also described in the table. Note that the last column within the table specifies that four bytes of data will be returned and consists of the “Port Status” and “Change Indicators.”

## **USB 3.0 Technology**

---

*Table 22-5: Setup Stage Request Data for GetPortStatus*

Offset	Field Name	Size	Value	Description
0	Request Type	1	Bitmap	D7 - Data Transfer Direction: 0 = Host to device 1 = Device to host  D6:5 - Request Type: 0 = Standard 1 = Hub Class Specific 2 = Vendor 3 = Reserved  D4:0 - Recipient 0 = Device 1 = Interface 2 = Endpoint 3 = Other (Port) 4-31 = Reserved
1	Request	1	Value = 00h	Request Code = GetDescriptor
2	Value	2	Value = 0000h	Value is request dependent
4	Index	2	Index = 0003h	Index value defines Port Number
6	Size	2	Count = 0004h	Defines bytes to transfer

---

### **Clear Port Connection Request**

Following the GetPortStatus request, the hub client must clear the Status Change bit in order for the port to detect another connection event. This is done via a control transfer defined as the ClearFeature command with the feature field set to clear port connections (C\_Port\_Connections). The 8 bytes delivered by the Setup transaction defines the ClearPortConnection request as shown in Table 22-6 on page 497.

## Chapter 22: Device Configuration

---

The fields are defined as follows:

- Request Type
  - Direction of transfer is from host to device
  - Request is Class Specific
  - Recipient is other (port)
- Request = ClearFeature (00h)
- Value = Feature Selector (16d)
- Index = Port Number (3 in this example)
- Length = Zero (Data Stage not used)

Table 22-6: *ClearPortConnection Request*

Request Type (1 Byte)	Request (1 Byte)	Value (2 Bytes)	Index (2 Bytes)	Length (2 Bytes)	Data
00100011B	ClearFeature (01h)	Feature Selector (16d)	Port Number (0003h)	Zero	None

Other hub and port status changes also result in change indicators being set and must also be cleared. Table 22-7 lists the change indicators and related the Setup Data.

Table 22-7: *ClearPortConnection Request*

Request Type (1 Byte)	Request (1 Byte)	Value (2 Bytes) Feature Selector	Index (2 Bytes)	Length (2 Bytes)	Data
00100000B	ClearFeature (01h)	C_Hub_Local_Power (0d)	Zero	Zero	None
00100000B	ClearFeature (01h)	C_Hub_Local_Power (1d)	Zero	Zero	None
00100011B	ClearFeature (01h)	C_Port_Connection (16d)	Port Number	Zero	None
00100011B	ClearFeature (01h)	C_Port_Over_Current (19d)	Port Number	Zero	None

## USB 3.0 Technology

---

Table 22-7: ClearPortConnection Request (Continued)

Request Type (1 Byte)	Request (1 Byte)	Value (2 Bytes) Feature Selector	Index (2 Bytes)	Length (2 Bytes)	Data
00100011B	ClearFeature (01h)	C_Port_Reset (20d)	Port Num- ber	Zero	None
00100011B	ClearFeature (01h)	C_Port_Link_State (25d)	Port Num- ber	Zero	None
00100011B	ClearFeature (01h)	C_Port_Configuration_Error (26d)	Port Num- ber	Zero	None
00100011B	ClearFeature (01h)	C_BH_Port_Reset (29d)	Port Num- ber	Zero	None

---

## The Device Configuration Process

Virtually all of the configuration process involves control transfers (via default endpoint zero) targeting the downstream hub port and the attached device. See Chapter 6, entitled "Control Protocol," on page 105 for a review of control transfers.

The following describes the general sequence of events performed by system software and tracks the state transitions that occur when configuring a device. As discussed earlier some portions of the device configuration sequence may be performed differently depending on operating environment.

---

### Address Device

During Reset devices force their address register to zero and the first access to a device will be performed using address zero. This is the device's default state. The first step in the configuration process is to assign a unique address to the device. This is accomplished with a SetAddress request that delivers a 7-bit address to the device. Table 22-8 on page 499 shows the Setup data that is delivered during the SetAddress request. This actions places the device in it's address state. The device can only change its address after the status stage of the control transfer has completed. Note that only one device can be configured at a time.

## Chapter 22: Device Configuration

---

Table 22-8: Set Address Request

Request Type (1 Byte)	Request (1 Byte)	Value (2 Bytes)	Index (2 Bytes)	Length (2 Bytes)	Data
0000000B	SetAddress (05h)	Device Address (0001h)	Zero	Zero	None

---

## Getting the Descriptors

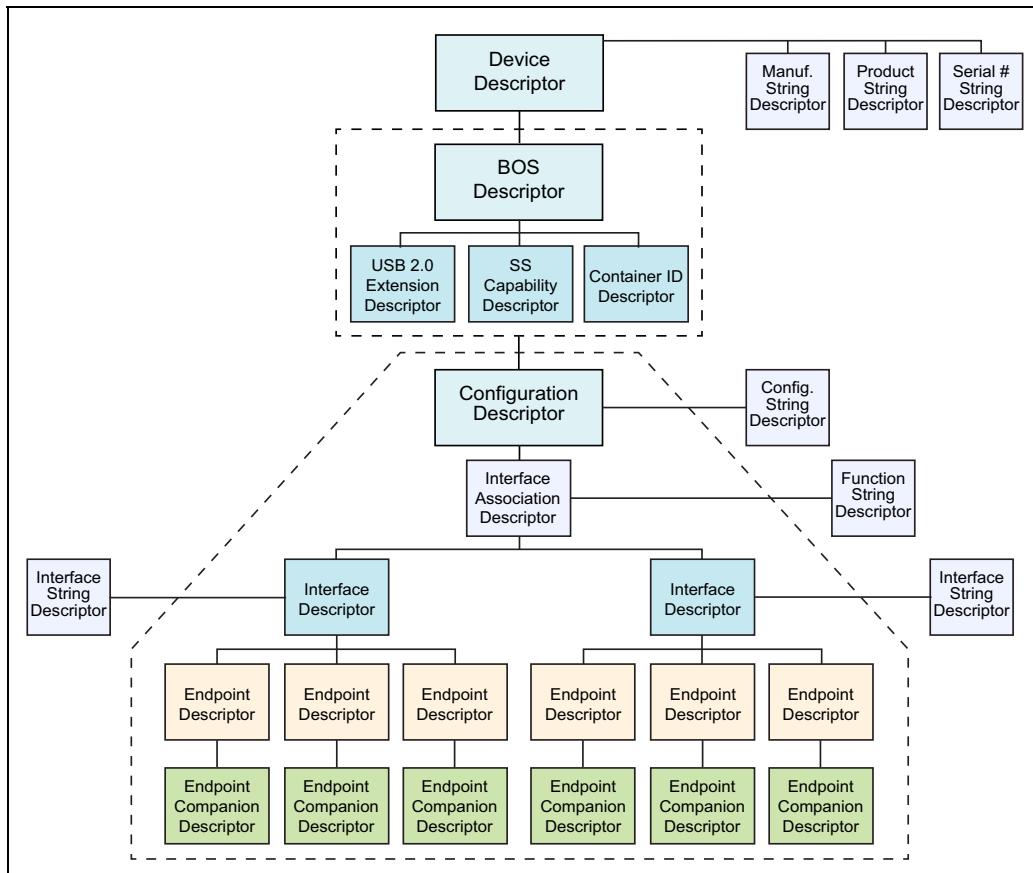
### General

Device implementers must create descriptors to reflect the characteristics and behaviors of the device. This section provides the definition and format of the standard descriptors that typically accompany a SuperSpeed device. Figure 22-3 on page 500 illustrates the descriptor tree structure. Note that some of the descriptors are optional, including:

- String Descriptors
- Interface Association Descriptors
- Endpoint and Endpoint Companion Descriptors (It is possible to implement a device that only uses the default control endpoint (zero) but in the author's opinion this is not likely to be implemented with a SuperSpeed device.)

# USB 3.0 Technology

Figure 22-3: SuperSpeed Descriptors



## Standard Descriptors

Every USB device contains standard descriptors that are read and parsed by the USB configuration software. The descriptors determine the device's capabilities and characteristics that enable software to determine if the device can be supported based on power and bandwidth requirements, along with other device-specific information.

Configuration software performs the GetDescriptor request to access all descriptor types. Table 22-9 shows the format of the 8-byte data payload that is delivered in the setup transaction. The request field specifies GetDescriptor and

## Chapter 22: Device Configuration

---

the value field identifies which descriptor is being fetched by software. The length field defines the memory buffer size used for holding the descriptor contents. Software may choose to fetch only a portion of the descriptor.

Table 22-9: Get Descriptor Request – Setup Data

Request Type (1 Byte)	Request (1 Byte)	Value (2 Bytes)	Index (2 Bytes)	Length (2 Bytes)	Data
10000000B	GetDescriptor (06h)	Descriptor Type	Zero	Size of Descriptor	Descriptor Content

Table 22-10 lists the standard descriptors, identifies the descriptor values and specifies which descriptors can be fetched directly via the GetDescriptor request. For example, the Interface, Endpoint, and Endpoint Companion descriptors cannot be fetched directly. Instead they are fetched via the Configuration descriptor, which can return all the descriptors that are part of the complete configuration. Column three in Table 22-10 specifies which descriptors can be accessed directly and which cannot.

Table 22-10: SuperSpeed Descriptors

Descriptor Type	Value decimal	Accessed directly?	Description
Device	1	Yes	Includes general information about the device. Reports number of Configurations, vendor and device class information
Configuration	2	Yes	Reports number of interfaces, power information and other attributes associated with this configuration. This descriptor also returns the contents of all descriptors that are part of this particular configuration, denoted by *.
String	3	Yes	Optional UNICODE strings reporting manufacturer, serial number, device type and other human readable information

## **USB 3.0 Technology**

---

*Table 22-10: SuperSpeed Descriptors*

Descriptor Type	Value decimal	Accessed directly?	Description
Interface	4	No	Reports number of endpoints and attributes associated with this interface.
Endpoint	5	No	Reports endpoint address, direction of data and other attributes.
Reserved	6 - 7	NA	
Interface Power	8	Yes	Related to the <i>USB Interface Power Management Specification</i> but it is unclear whether this specification and descriptor are actually implemented.
On-The-Go	9	Yes	Reserved for OTG and Embedded Host.
Debug	10 (Ah)	Yes	Optional Host Controller feature (i.e., xHCI).
Interface Assoc.	11 (Bh)	No	Required in cases where there are two or more interfaces associated with the same function.
BOS	15 (Fh)	Yes	Binary Object Store — reports a variety of Device Capability entries, including USB 2.0 Link Power Management, SuperSpeed capabilities and UUID value, denoted by * below.
Device Capability	16 (10h)	No*	This Descriptor applies to all of the Device Capability entries included under the BOS descriptor.
SS USB Endpoint Companion	48 (30h)	No	Reports new capabilities that may be used by SuperSpeed devices, including Data Bursting and Bulk Streaming.

---

## **Device Descriptor**

Table 22-11 shows the Setup data used when performing a GetDeviceDescriptor request. The 1d value resides in the upper byte of the Value field and indicates the Device descriptor is being accessed.

## Chapter 22: Device Configuration

---

Table 22-11: Get Device Descriptor Request

Request Type (1 Byte)	Request (1 Byte)	Value (2 Bytes)	Index (2 Bytes)	Length (2 Bytes)	Data
10000000B	GetDescriptor (06h)	Descriptor Type (1d)	Zero	12h	18 Bytes

The contents of the Device descriptor is shown in Table 22-12. The following sections discuss how host software evaluates the device descriptor during the configuration process. Some fields in Table 22-12 are not discussed further since the definition of these fields is obvious and does not benefit from additional verbiage (in the judgment of the author).

Table 22-12: Device Descriptor Definition

Offset	Field	Size (bytes)	Value	Description
0	Length	1	12h	Size of this descriptor in bytes = 18d
1	Descriptor Type	1	01h	DEVICE Descriptor Type = 1
2	USB	2	BCD	USB Specification Release Number in Binary-Coded Decimal (0x0300). This field identifies the release of the USB specification with which the device and its descriptors are compliant (3.0 in this example).
4	Device-Class	1	Class	Class code (assigned by USB). 0 = each interface defines its own class 1-FEh = indicates aggregate interface used FFh = device class is vendor specific
5	Device Subclass	1	Subclass	Subclass code (assigned by USB) All values qualified by Class field. 0 — when class field is set to zero FFh — when class field is set to FFh 1-FEh — Reserved by USB-IF

## **USB 3.0 Technology**

---

*Table 22-12: Device Descriptor Definition (Continued)*

Offset	Field	Size (bytes)	Value	Description
6	Device Protocol	1	Protocol	Protocol code (assigned by USB) All values qualified by Class field. 0 — when class field is set to zero FFh — when class field is set to FFh 1-FEh — Reserved by USB-IF
7	MaxPacket-Size0	1	09h	Maximum packet size for endpoint zero. $2^{\text{bMaxPacketSize0}}$ $\text{bMaxPacketSize0} = 09h$ (512B) for SS
8	Vendor	2	ID	Vendor ID (assigned by USB-IF).
10	Product	2	ID	Product ID (assigned by manufacturer).
12	Device	2	BCD	Device release number in binary-coded decimal.
14	Manufacturer	1	Index	Index of manufacturer string descriptor (optional).
15	Product	1	Index	Index of product string descriptor (optional).
16	Serial Number	1	Index	Index of device serial number string descriptor (optional).
17	NumConfigurations	1	Number	Number of configurations supported.

### **Descriptor Length and Type**

The first two fields of every descriptor are:

1. Length Field — defines the size of the current descriptor, which is 18 Bytes for the Device descriptor
2. Descriptor Type Field — that defines the current descriptor, which of course is the Device descriptor in this example.

### **DeviceClass/SubClass/Protocol**

These fields are included in the Device descriptor and also in the Interface descriptor. The definition for these fields is specified below:

## **Chapter 22: Device Configuration**

---

- DeviceClass
  - A value of zero (0) indicates that each Interface associated with a configuration has its own Class definition and each interface functions independently.
  - Any Value in the range of 1-FEh indicates aggregate interfaces are used. Also, each interface has its own Class definition but may interact with one or more interfaces within the configuration.
  - A value of FFh indicates the Device Class is vendor specific.
- DeviceSubClass — Value is dependent on the DeviceClass field
  - When the DeviceClass has a zero (0) the SubClass must also have a value of zero.
  - A value of FFh
  - When the DeviceClass field is not FFh, the SubClass values are reserved.
- DeviceProtocol — Value is dependent upon the DeviceClass and SubClass
  - A value of zero (0) means that Class-specific protocols are not reported at the device level, but Class-specific protocols may be defined at the interface level.
  - Values other than 0 and FFh indicate that class-specific protocols are used on a device basis.
  - A value of FFh indicates the device uses vendor-specific protocol on a device basis.

### **Maximum Payload Size for Endpoint Zero (EP0)**

Endpoint zero is called the default endpoint because every device must use EP0 to perform the initial access to the device. USB 2.0 permits some devices to choose a their maximum payload size for EP0 within a given range and the value selected is reported in offset 6 of the Device descriptor. However, Super-Speed USB requires the Maximum payload size to be 512 Bytes.

### **String Index Values**

The Device descriptor has three index values (offset 14, 15 and 16) used to access the Manufacturer, Product and Serial Number String Descriptors. Technically, these index values are optional but most devices do implement at least two and sometimes all three of these index values.

# **USB 3.0 Technology**

---

## **BOS Descriptor**

The BOS (Binary Object Store) descriptor provides access to three Device Capability descriptors that are returned in order, as follows:

1. USB 2.0 Extension — This extension requires all SuperSpeed devices to support the USB 2.0 Link Power Management protocols.
2. SUPERSPEED\_USB — Provides device level capability information including exit latency values, USB speeds supported and more.
3. CONTAINER\_ID — Provides a unique ID across all operating modes.

Table Table 22-13 defines Setup Data associated with the Get BOS Descriptor.

*Table 22-13: Get Configuration Descriptor Request*

Request Type (1 Byte)	Request (1 Byte)	Value (2 Bytes)	Index (2 Bytes)	Length (2 Bytes)	Data
10000000B	GetDescriptor (06h)	Descriptor Type (15d)	Zero	0009h	9 Bytes

Table 22-14 shows or information contained with the BOS descriptor. There are additional Device Capability descriptors that can only be accessed via the BOS descriptor. Currently, there are 3 of these descriptors and each are discussed below. A BOS descriptor is typically accessed one time to determine the size of the buffer and a second time to fetch the BOS and Device Capability descriptors that have been implemented by the device.

*Table 22-14: BOS Descriptor Definition*

Offset	Field	Size	Value	Description
0	Length	1	05h	Size of this descriptor is 5 bytes.
1	Descriptor-Type	1	0Fh	BOS descriptor type is 15.

## Chapter 22: Device Configuration

---

Table 22-14: BOS Descriptor Definition (Continued)

Offset	Field	Size	Value	Description
2	TotalLength	2	Number	Length of the BOS descriptor and the Device Capability descriptors that follow.
4	NumDevice-Capabilities	1	Number	The number of Device Capabilities implemented below the BOS descriptor

### USB 2.0 Extension

The first Device Capability descriptor is the USB 2.0 Extension. This descriptor relates to the requirement that all SuperSpeed devices must support at least one USB 2.0 speed. The USB 2.0 Extension specifically refers to the USB 2.0 Link Power Management (LPM) feature. All SuperSpeed devices are required to support USB 2.0 LPM when operating at either the low-, full- or high-speed operation. Table 22-15 on page 507 lists the entries associated with the USB 2.0 LPM Device Capability.

Table 22-15: BOS — USB 2.0 Extension Descriptor

Offset	Field	Size	Value	Description
0	Length	1	7h	Size of this descriptor is 7 bytes.
1	DescriptorType	1	10h	Device Capability descriptor type is 16.
2	DevCapability-Type	1	02h	Device Capability Type is 02h: USB 2.0 Extension
3	Attributes	4	Bitmap	Support for device level features. A value of 1 indicates support.  Bit 0 = Reserved (zero) Bit1 = Link Power Management (LPM) Support is required and Bit 1 must be set to 1. Bits 31:2 = Reserved and set to zero

# USB 3.0 Technology

---

## SuperSpeed USB Device Capability

This descriptor specifies whether Latency Tolerance Monitoring is supported by this device, it reports a bitmap indicating which USB 2.0 speeds are supported, specifies the lowest speed that maintains full functionality and reports U1 and U2 exit latencies back to U0.

Table 22-16: BOS — SuperSpeed Device Capability

Offset	Field	Size	Value	Description
0	Length	1	0Ah	Size of this descriptor is 10 bytes.
1	DescriptorType	1	10h	Device Capability descriptor type is 16.
2	DevCapability-Type	1	03h	Device Capability Type is 03h: SuperSpeed USB Device Capability
3	Attributes	1	Bitmap	Support for device level features. A value of 1 indicates support.  Bit 0 = Reserved (zero) Bit 1 = LTM Capable, setting this bit to 1 indicates this device supports Latency Tolerance Reporting. Bits 7:2 = Reserved (zero)
4	SpeedsSupported	2	Bitmap	This bitmap indicates the speeds supported by the device. A one indicates support for the speed.  Bit 0 = Device supports low-speed Bit 1 = Device supports full-speed Bit 2 = Device supports high-speed Bit 3 = Device supports SS (5 Gbps) Bits 15:4 Reserved (zero)
6	Functionality Support	1	Number	Specifies the lowest speed at which the device supports full functionality. The value is based on the speeds defined in the previous row. For example a value of 1 indicates full-speed is the lowest fully functional speed.

## Chapter 22: Device Configuration

Table 22-16: BOS — SuperSpeed Device Capability (Continued)

Offset	Field	Size	Value	Description
7	U1DevExitLatency	1	Number	<p>The value in the field indicates the worst case latency for the device port when transitioning from U1 back to U0. The value presumes the link partner does not limit the value.</p> <p>00h = zero 01h = Less than 1μs 02h = Less than 2μs 03h = Less than 3μs ---- ----- 0Ah = Less than 10μs 0B-FFh = Reserved</p>
8	U2DevExitLatency	2	Number	<p>The value in the field indicates the worst case latency for the device port when transitioning from U2 back to U0. The value presumes the link partner does not limit the value. The value applies to all ports on a device (e.g.. Hub)</p> <p>0000h = zero 0001h = Less than 1μs 0002h = Less than 2μs 0003h = Less than 3μs ---- ----- 07FFh = Less than 2047μs 0800-FFFFh = Reserved</p>

### Container ID

This device level descriptor must be implemented by all hubs and is optional for other devices. The Container ID value is a 128-bit number defined as a Universal Unique Identifier (UUID). This value is intended to be universal in that it applies to all modes of use (e.g., a hub operating in SuperSpeed and in USB 2.0 modes). The UUID also applies to devices that implement connections other than USB. For example, a USB Disc Drive that also has a Firewire connector. Table 22-17 depicts the content of the Container ID descriptor.

## **USB 3.0 Technology**

---

*Table 22-17: BOS — Container ID*

Offset	Field	Size	Value	Description
0	Length	1	14h	Size of this descriptor is 20 bytes.
1	DescriptorType	1	10h	Device Capability descriptor type is 16.
2	DevCapability-Type	1	04h	Device Capability Type is 04h: ContainerID
3	Reserved	1	Number	Reserved
4	ContainerID	16	UUID	A 128-bit unique number that can be used universally across all device instances and modes of operation.

---

## **Configuration Descriptor**

Table 22-18 shows the Setup data used when performing a GetConfiguration-Descriptor request. The 2d value in the upper byte of the Value field indicates that this descriptor is defined as a Configuration descriptor. Software reads a configuration descriptor to obtain global information regarding a given configuration option.

*Table 22-18: Get Configuration Descriptor Request*

Request Type (1 Byte)	Request (1 Byte)	Value (2 Bytes)	Index (2 Bytes)	Length (2 Bytes)	Data
10000000B	GetDescriptor (06h)	Descriptor Type (2d)	Zero	0009h	9 Bytes

Table 22-19 on page 511 lists and describes the fields associated with the Configuration descriptor. The subsequent sections discuss some of the fields that may need additional explanation.

## Chapter 22: Device Configuration

---

Table 22-19: Configuration Descriptor Definition

Offset	Field Name	Size (bytes)	Value	Description
0	Length	1	09h	Size of this descriptor is 9 Bytes
1	DescriptorType	1	02	Configuration value = 02h.
2	TotalLength	2	Number	Total length of data returned for this configuration. Includes the combined length of all descriptors (configuration, interface, endpoint, and class or vendor specific) returned for this configuration.
4	Num Interfaces	1	Number	Number of interfaces supported by this configuration.
5	Configuration Value	1	Number	Value to use as an argument to Set Configuration to select this configuration.
6	Configuration	1	Index	Index of string descriptor describing this configuration.
7	Attributes	1	Bitmap	Configuration characteristics  D7 Reserved (set to 1) D6 Self Powered D5 Remote Wakeup D4:0 Reserved (reset to 0)  If the device is self-powered D6 = 1 If a device configuration supports remote wakeup, D5 is set to 1.
8	MaxPower	1	mA	Maximum power consumption of USB device from the bus in this specific configuration when the device is fully operational. Expressed in 8 mA units (i.e., 50 = 400 mA) with SuperSpeed.

# **USB 3.0 Technology**

---

## **Number of Interfaces**

A given device may have two or more interfaces that require separate class drivers. An interface consists of a collection of endpoints through which a given device driver would control and communicate with its device. The NumInterfaces field specifies the number of interfaces that are implemented in this configuration.

## **Configuration Value**

Once configuration software has selected one of the configurations defined by the device, the device must be configured. Each configuration descriptor has a unique configuration value (offset 5) that is used to configure the device. Until the configuration value is written to the device, it must not consume more than 150mA of current and is not fully operational.

System software configures a device by using the Set Configuration request. The configuration value is loaded into the “value” field of the setup transaction of the Set Configuration request. Once configured, the device takes on the characteristics defined by the selected configuration.

## **Attributes and Maximum Power**

The configuration attributes define how a device is powered and if it supports remote wakeup. A device configuration reports whether the configuration is bus-powered or self-powered. Device status reports whether the device is currently self-powered. If a device is disconnected from its external power source, it updates device status to indicate the device is no longer self-powered.

A device is limited to the amount of bus power that it specifies in the MaxPower field of the Configuration descriptor. This is true even if the device loses its external power source.

If a device can continue to operate when disconnected from its external power source, it continues to do so. If the device cannot continue to operate, the device will drop operations that it can no longer support. Host software may determine the cause of the failure by checking status and noting the loss of the device’s power source.

---

## **Interface Association Descriptor**

The Interface Association descriptor was defined in the USB 3.0 specification. This descriptor can only be read as part of the GetConfigurationRequest. The descriptor associates two or more interfaces that are part of the same function.

## Chapter 22: Device Configuration

---

The specification requires that an Interface Association descriptor be placed immediately before the collection of interfaces (including interfaces associated with alternate settings) which are part of the same function. Table 22-20 describes the content of Interface Association descriptor.

*Table 22-20: Interface Association Descriptor Definition*

Offset	Field	Size	Value	Description
0	Length	1	08h	Size of this descriptor is 8 bytes
1	DescriptorType	1	0Bh	Interface Association descriptor type = 11
2	FirstInterface	1	Number	First interface number associated with this function
3	InterfaceCount	1	Number	Total number of interfaces comprising the function
4	FunctionClass	1	Class	Class Code of First Interface
5	Function SubClass	1	SubClass	SubClass of First Interface
6	Function Protocol	1	Protocol	Protocol of First Interface
7	Function	1	Index	Index of string describing this function

---

## Interface Descriptors

### General

An Interface is a collection of endpoints that define a function (class). The software driver that manages the function is called a Class Driver, that initializes and controls the function. Some devices may define a single interface while others may require two or more interfaces (typically called a composite device). Table 22-21 lists the format and definition of the interface descriptor. The subsequent sections discuss some of the fields that might benefit by having additional explanation.

# **USB 3.0 Technology**

---

*Table 22-21: Interface Descriptor Definition*

Offset	Field	Size	Value	Description
0	Length	1	09h	Size of this descriptor is 9 bytes
1	Descriptor-Type	1	04h	Interface descriptor type = 4
2	Interface Number	1	Number	Number of the interface number associated with this function (zero based)
3	Alternate Setting	1	Number	Optional value used to define the setting for modifying the interface setting
4	Number of Endpoints	1	Number	Number of endpoints associated with this interface. If zero, the interface only uses only Endpoint zero (default control pipe).
5	InterfaceClass	1	Class	0 = reserved for future use 1-FEh = Class values assigned by USB-IF FFh = Class is Vendor-Specific
6	Interface SubClass	1	SubClass	The SubClass values are qualified by the Class value.  0 = if Class value is zero 1-FEh = SubClass value is assigned by USB-IF FFh = Class is Vendor-Specific
7	Interface Protocol	1	Protocol	The Protocol values are qualified by Interface-Class and SubClass values  0 = Class-specific protocol not used 1-FEh = Protocol value is assigned by Class Specification FFh = Device uses Vendor-Specific protocol
8	Function	1	Index	Index of string describing this function

## **Interface Number and Alternate Setting**

The “interface number” and “alternate settings” fields within the interface descriptor support the alternate setting feature. Devices may define alternate interfaces within the same configuration. The intent is to permit adjustments to a configuration during normal operation, after the initial configuration has been completed. A device that supports alternate settings will include one or more sets of additional interface and endpoint descriptors with the same interface,

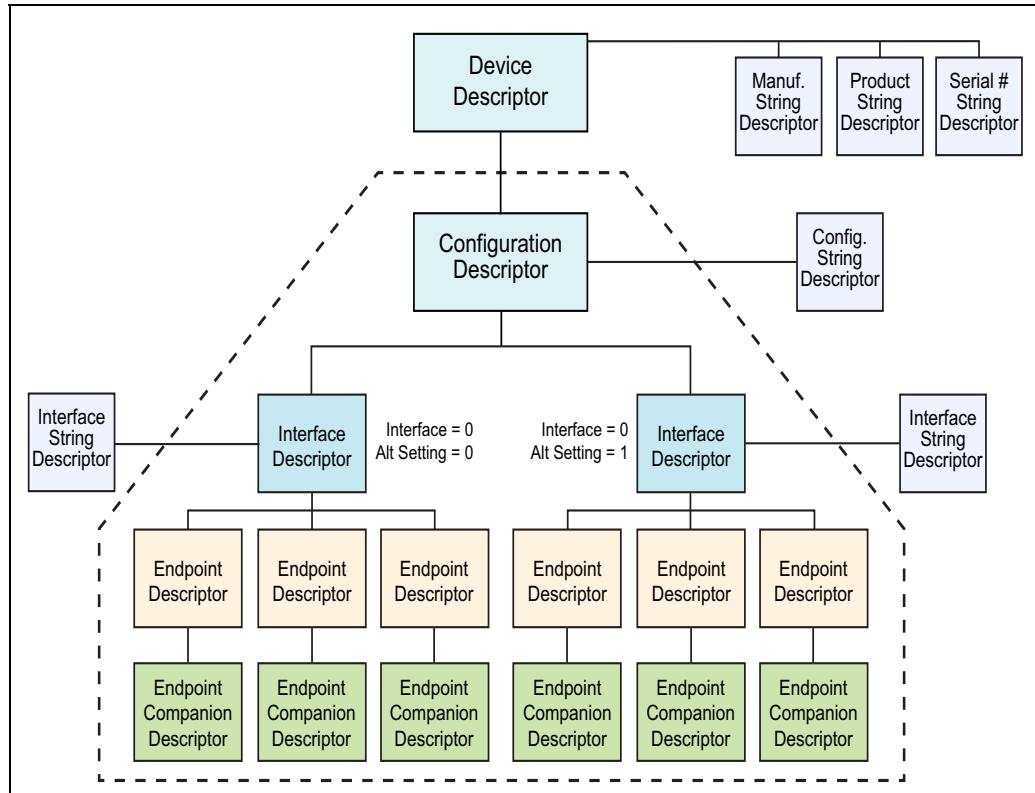
## Chapter 22: Device Configuration

but that contain different alternate settings.

As an example, consider the descriptor tree in Figure 22-4 on page 515. Note that both interface descriptors have an “interface number” of zero, specifying that each defines settings for interface zero. However, the “alternate settings” field of each interface descriptor is different. During configuration, alternate setting zero is used by default. The other setting (one) can be chosen after configuration to “fine tune” the configuration. Host software uses the alternate setting value to select the alternate interface of choice.

When the client driver chooses to change the alternate setting it performs a SetInterface request. The Alternate setting value selects the interface it wants to use.

Figure 22-4: Descriptor Tree Containing Alternate Interface Settings



# **USB 3.0 Technology**

---

## **Endpoint Descriptors**

The endpoint descriptor defines the actual registers that are implemented within a given device. Table 22-22 on page 516 shows the format and definition of an endpoint descriptor. These descriptors define the capabilities of each register and specify information such as the:

- type of transfer required by this endpoint
- direction of the transfer (IN or OUT)
- bandwidth needed
- polling interval

For periodic endpoint types, configuration software must determine if the USB system can support the endpoint based on its bandwidth requirements (specified in the MaxPacketSize field). If the endpoint bandwidth requirements exceed the capabilities, then the device is not configured and the user is notified.

Note that in some cases bi-directional communication is required. For example, if an input/output register is implemented within a function, two endpoint descriptors must be created -- one IN endpoint and one OUT endpoint. The exception to this approach is a control endpoint. Control transfers are the only USB transfers that support bidirectional data flow.

Endpoint descriptors are designed for supporting all of the transfer types — Bulk, Interrupt, Isochronous and Control. Offset 3 in Table 22-22 specifies the type of transfer associated with this endpoint. The Synchronization Type and Usage Type are only relevant for isochronous endpoints and are ignored otherwise.

*Table 22-22: Endpoint Descriptor Definition*

Offset	Field	Size	Value	Description
0	Length	1	07h	Size of this descriptor is 7 bytes.
1	DescriptorType	1	05h	Endpoint descriptor type is 5.

## Chapter 22: Device Configuration

---

*Table 22-22: Endpoint Descriptor Definition (Continued)*

Offset	Field	Size	Value	Description																			
2	EndpointAddress	1	Endpoint	<p>The address of the endpoint on the USB device described by this descriptor. The address is encoded as follows:</p> <p>Bit 0:3 Endpoint number      Bit 4:6 Reserved, reset to zero      Bit 7 Direction:          0 = OUT endpoint          1 = IN endpoint          (Ignored for Control endpoints)</p>																			
3	Attributes	1	Bitmap	<p>This field describes the endpoint's attributes when it is configured using the ConfigurationValue.</p> <p><b>Endpoint Transfer Types</b></p> <table> <tr><td>Bits 1:0 Transfer Type</td></tr> <tr><td>00 Control</td></tr> <tr><td>01 Isochronous</td></tr> <tr><td>10 Bulk</td></tr> <tr><td>11 Interrupt</td></tr> </table> <p><b>Interrupt Endpoints:</b></p> <table> <tr><td>Bits 3:2 Reserved</td></tr> </table> <p><b>Bits 5:4 Usage Type</b></p> <table> <tr><td>00 Periodic</td></tr> <tr><td>01 Notification</td></tr> <tr><td>10 Reserved</td></tr> <tr><td>11 Reserved</td></tr> </table> <p><b>Isochronous Endpoints</b></p> <table> <tr><td>Bits 3:2 Synchronization Type</td></tr> <tr><td>00 No Synchronization</td></tr> <tr><td>01 Asynchronous</td></tr> <tr><td>10 Adaptive</td></tr> <tr><td>11 Synchronous</td></tr> </table> <p><b>Bits 5:4 Usage Type</b></p> <table> <tr><td>00 Data endpoint</td></tr> <tr><td>01 Feedback endpoint</td></tr> <tr><td>10 Implicit feedback data EP</td></tr> <tr><td>11 Reserved</td></tr> </table> <p><b>Bits 7:6 Reserved (must be zero)</b></p>	Bits 1:0 Transfer Type	00 Control	01 Isochronous	10 Bulk	11 Interrupt	Bits 3:2 Reserved	00 Periodic	01 Notification	10 Reserved	11 Reserved	Bits 3:2 Synchronization Type	00 No Synchronization	01 Asynchronous	10 Adaptive	11 Synchronous	00 Data endpoint	01 Feedback endpoint	10 Implicit feedback data EP	11 Reserved
Bits 1:0 Transfer Type																							
00 Control																							
01 Isochronous																							
10 Bulk																							
11 Interrupt																							
Bits 3:2 Reserved																							
00 Periodic																							
01 Notification																							
10 Reserved																							
11 Reserved																							
Bits 3:2 Synchronization Type																							
00 No Synchronization																							
01 Asynchronous																							
10 Adaptive																							
11 Synchronous																							
00 Data endpoint																							
01 Feedback endpoint																							
10 Implicit feedback data EP																							
11 Reserved																							

## USB 3.0 Technology

---

Table 22-22: Endpoint Descriptor Definition (Continued)

Offset	Field	Size	Value	Description
4	MaxPacketSize	2	Number	<p>Maximum packet size for this endpoint.</p> <p><b>Control endpoints</b> = 512 Bytes <b>Bulk endpoints</b> = 1024 Bytes <b>Interrupt endpoints</b></p> <ul style="list-style-type: none"><li>— 1024 Bytes when MaxBurst &gt; zero</li><li>— 1-1024 Bytes when MaxBurst = zero</li></ul> <p><b>Isochronous endpoints</b></p> <ul style="list-style-type: none"><li>— 1024 Bytes when MaxBurst &gt; zero</li><li>— 0-1024 Bytes when MaxBurst = zero</li></ul>
6	bInterval	1	Number	<p>Interval for servicing endpoint. Expressed in 125µs units.</p> <p>Values are specified by this formula: <math>2^{b\text{Interval}-1}</math></p> <p><b>bInterval value:</b></p> <ul style="list-style-type: none"><li>1-16 Isochronous and Periodic Interrupt EPs</li><li>8-16 Notification Interrupt EPs</li></ul>

---

## Endpoint Companion Descriptor

The Endpoint descriptor as defined by USB 2.0 has insufficient space for reporting new features associated with SuperSpeed USB. Consequently, the Endpoint Companion descriptor was implemented to report some of the new capabilities that include:

- Data Bursting
- Bulk Streaming

Table 22-23 on page 519 describes the content of the Endpoint Companion descriptor. Following the descriptor there are sections that explain the descriptors more fully.

## Chapter 22: Device Configuration

---

Table 22-23: Endpoint Companion Descriptor Definition

Offset	Field	Size	Value	Description
0	Length	1	06h	Size of this descriptor is 6 bytes.
1	Descriptor-Type	1	30h	SuperSpeed Endpoint Companion descriptor type is 48.
2	MaxBurst	1	Number	Zero-based Values can range from 0-15 and specify the maximum size of the burst (1-16). (Not valid for Control endpoints)
3	Attributes	1	Bitmap	<b>Bulk Endpoints:</b> Bits 4:0 MaxStreams Values 0h = Streams not supported 1-Fh = $2^{\text{MaxStreams}}$ Bits 7:5 Reserved (zeros)  <b>Control or Interrupt Endpoint:</b> Bits 7:0 Reserved (zeros)  <b>Isochronous Endpoint:</b> Bits 1:0 Mult - zero based value determines max packets within service interval and equals $(\text{MaxBurst} + 1) \times (\text{Mult} + 1)$ Bits 7:2 Reserved
4	BytesPer-Interval	2	Number	Periodic endpoints only — Total bytes transferred during service interval. Used for reserving time for scheduling isochronous traffic.

---

## String Descriptors

String descriptors provide Unicode characters used to display information to users. All string descriptors are considered to be optional, however, most devices implement String descriptors associated with the Device descriptor. The Device descriptor includes three index values used to access the Manufacturer, Product and Serial Number String Descriptors. (See offsets 14, 15, and 16 in Table 22-12 on page 503 and 256..) Most devices implement the Manufacturer and Product index values and sometimes all three are implemented.

## **USB 3.0 Technology**

---

Only four descriptor types include a string index field and all have a single String index field except for the Device descriptor.

- Device
- Configuration
- Interface Association
- Interface

The index values are typically assigned sequentially beginning with one (Manufacturer ID), then two (Product ID). Any index value of zero (e.g., Serial Number) indicates the string is not supported.

String descriptors may also support multiple languages, which are referenced with a 16-bit Language ID. These language IDs can be found at:

<http://www.usb.org/developers/docs/> (near the bottom of the page)

GetStringDescriptor uses an index value of zero to acquire a list of all Languages supported by a device. Information returned may include a single language ID or a string of Language IDs or no IDs at all.

Accessing a String descriptor involves:

1. Software Selects the Index value associated with the String descriptor that is to be read (e.g., Device Manufacturer). Exception: A value of zero is used to get a string of supported Language IDs.
2. Configuration software prepares the GetStringDescriptor request by Preparing the Setup Data.
3. The GetStringDescriptor request is performed, thereby returning the requested string.

Table 22-24 displays the fields associated with the Setup Data and each field is described below:

- The *Request Type* indicates a standard request targeting the device
- The *Request* field specifies GetDescriptor
- The *Value* field contains a value of 3 specifying a String descriptor
- The *Index* field holds the index value previously read from the Device descriptor
- The *Length* field specifies the amount of data to be fetched from the String descriptor. The length of the String descriptor is not known until the length field of the descriptor is read. Consequently, it is common for a GetStringDescriptor to be read twice.

## Chapter 22: Device Configuration

---

Table 22-24: Get String Descriptor Request

Request Type (1 Byte)	Request (1 Byte)	Value (2 Bytes)	Index (2 Bytes)	Length (2 Bytes)	Data
10000000B	GetDescriptor (06h)	Descriptor Type (3d)	Index Value	Total Length	String

Table 22-25 shows the contents of the String descriptor. The same basic format is used when accessing the Language IDs, except that offset 2 is the beginning of the Language ID list that consists of one or more 2 byte LANGIDs, rather than a simple Unicode string.

Table 22-25: String Descriptor Definition

Offset	Field	Size	Value	Description
0	Length	1	Number	Size of the string descriptor
1	Descriptor-Type	1	03h	String descriptor type is 3
2	String	N	Number	Unicode string or List of 16-bit Language IDs

---

## Setting Device Configuration

When configuration software parses the descriptors it verifies that all of the resources necessary to support the device are available, including bus power and bandwidth. Software then performs the Set Configuration request that delivers the configuration value to the device. This action fully enables the device, thereby permitting maximum power consumption as specified in the Configuration descriptor. The device transitions to the Configured state, but the device function(s) will likely need to be initialized and configured by one or more class drivers.

Table 22-26 lists the Setup Data associated with the SetConfiguration request. The lower byte contains the Configuration Value that is read from the Configuration descriptor.

## **USB 3.0 Technology**

---

*Table 22-26: SetConfiguration Request Data*

Request Type (1 Byte)	Request (1 Byte)	Value (2 Bytes)	Index (2 Bytes)	Length (2 Bytes)	Data
00000000B	SetConfiguration (09h)	Configuration Value	Zero	Total Length	None

---

## **Additional Requests**

Configuration software may perform other requests during the configuration process. These requests include enabling features and forwarding information to devices (including hub devices):

- Set Isochronous Delay — Used by devices that require reception of the Isochronous Time Stamp packet. Set Isochronous Delay allows adjustment of the time stamp value to account for propagation delay from root port to the device.
- SEL (System Exit Latency) — the SEL request specifies the SEL and PEL (Path Exit Latency) information for U1 and U2. This information can be used by SuperSpeed devices to adjust the Latency Tolerance Message, if supported.
- LTM\_ENABLE (Latency Tolerance Message enable) — this enables a device to generate Latency Toerance Message to the host to reduce system power consumption and increase transaction latency. See previous bullet item.
- U1\_ENABLE — this request enables a device to initiate link power transitions to the U1 state.
- U2\_ENABLE — this request enables a device to initiate link power transitions to the U2 state.

---

## **Class Driver Initialization**

The process of Class Driver initialization is a very broad topic given the large number of device classes that exist. Consequently, this topic is beyond the scope of this book. MindShare classes do cover the most common Class-specific implementations upon request.

---

---

# **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

# **USB 3.0 Technology**

---

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

## Chapter 23: SuperSpeed Hub Configuration

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
100000010B	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

# **USB 3.0 Technology**

---

## **Hub Class Requests**

Hubs have specific requests that manage Hubs and handle the downstream ports.

*Table 23-2: Hub Class Requests*

<b>Request Type</b>	<b>Request</b>	<b>Value (2 bytes)</b>	<b>Index (2 bytes)</b>	<b>Length (2 bytes)</b>	<b>Data</b>
10100000B 10100011B	GetHubStatus GetPortStatus (00h)	Zero Zero	Zero Port	4 bytes 4 bytes	Hub Status Port Status
00100000B 00100011B	ClearHubFeature ClearPortFeature (01h)	Feature Feature	Zero Port	Zero Zero	None
00000000B 00100011B	SetHubFeature SetPortFeature (03h)	Feature Feature	Zero Timeout, Port Selector	Zero	None
10100000B	GetHubDescriptor (06h)	Descriptor Type and Descriptor Index	Zero or Language ID	Descriptor Length	Descriptor
00100000B	SetHubDescriptor (07h)	Descriptor Type and Descriptor Index	Zero or Language ID	Descriptor Length	Descriptor
0010000B	SetHubDepth (12h)	Zero	Zero	Zero	None
10100011B	GetPortErrorCount (13h)	Zero	Port	Two	None

# **Chapter 23: SuperSpeed Hub Configuration**

---

---

## **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 536 to skip this section.

This section describes how devices detect each other’s presence and specifies the actions that must take place to report the event. Hub client software plays a pivotal role in this process. Each device attached to a root or external hub port must be detected by the hub client software, which triggers the configuration process. The primary events associated with device detection are summarized below and detailed in subsequent sections:

1. The hub client software applies power to the port, causing the link partners to enter their reset state.
2. On reset exit, a device attempts to detect the presence of its link partner.
3. When the link partners detect each other’s presence two things occur:
  - The Link Training and Initialization process is triggered.
  - The hub port sets port status bits indicating the attachment event, and sets the Status Change indicator to specify the port to which the device is attached.
4. When Link Training and Initialization completes, the link enters the U0 state and transactions can be performed.
5. The hub client driver performs an access to the hub’s Status Change indicator and detects port status changes.
6. The hub client driver accesses the device again, but this time it reads the contents of the port status bits to verify device attachment.
7. Having detected the device, the hub client notifies configuration software that a device is ready to be configured.

The host controller driver software creates an abstraction of the root hub so that the standard USB hub client driver can make requests to the host controller’s software driver. Because of this abstraction, the same process is described for both the root hub and external USB hubs. The actual mechanisms used to access status information from the hub and root ports involves accessing memory-mapped registers within the host controller. This is based on a PCI or PCIe host controller interface.

# **USB 3.0 Technology**

---

## **Device Detection Process**

Device detection begins with the application of  $V_{BUS}$  from a downstream root port or an external downstream hub port. When power is applied both link partners enter the Reset state, and the receivers apply hi-impedance terminations of  $25K\Omega$ s or greater. When reset is removed low-impedance terminations in the range of 18 to  $30\Omega$ s are applied. This causes entry to the LTSSM's RxDetect state. Chapter 20, entitled "Link Training," on page 427 describes the mechanism used by the link partners to detect device presence.

Having detected device attachment, Link Training begins with the downstream hub port (root or external port) setting port status bits, along with the hub's Status Change Endpoint (EP1 for external hubs), thus indicating which ports have incurred a status change. Subsequently, software checks the Status Change Endpoint to detect if one or more ports have incurred a change and then reads port status. (See Figure 23-1 on page 529.)

### **Status Change Endpoint (Endpoint 1)**

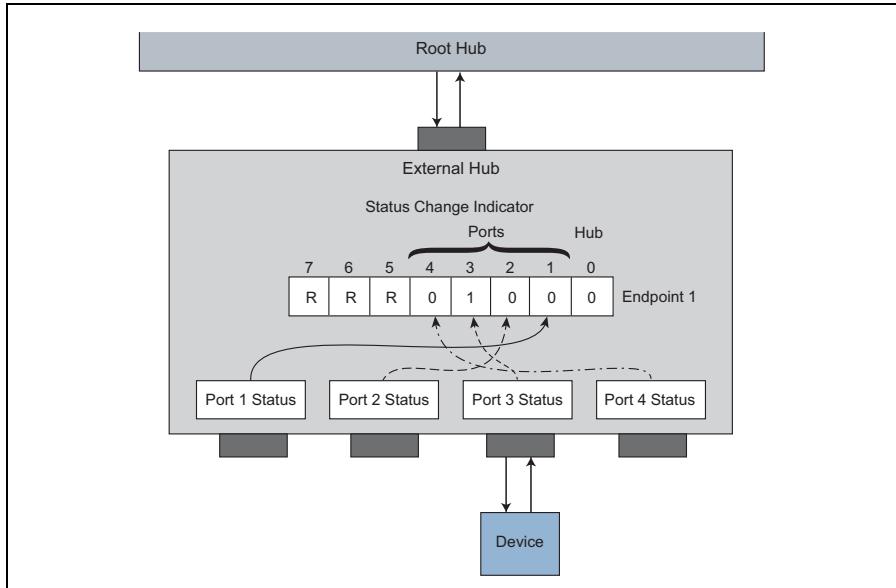
The first step for software is to determine whether a device has been attached to a port is to monitor the hub port. Upon detection of device attachment, the hub's status change endpoint sets a bit, indicating a status change has occurred within the port. In response, the port triggers an Endpoint Ready (ERDY) packet that is sent to the root port. This notifies the host controller to perform an IN transaction to fetch the status change information. Figure 23-1 on page 529 illustrates a hub attached to a root port. The format of the Status Change Endpoint (Bit 0) is reserved for global hub events. Bits 1-n identify the port numbers. In the example, four ports are implemented so bits five-to- seven are reserved. Notice also that port 3 has a device attached and bit 3 of the status change endpoint has been set by hardware.

Configuration software is aware of the number of ports supported by each hub (15 ports maximum), and therefore is aware of the size of the bitmap returned when the hub's status change endpoint is accessed. Status is reported in byte-sized fields with zeros returned in the bit fields corresponding to ports that are not implemented and that have not had a status change.

When software accesses the status change endpoint, the bitmap illustrated in Table 23-3 on page 530 is returned only if a status change has occurred. If no status information is present, the hub returns NRDY during the IN transaction. When an event does occur a status bit is set and the hub sends ERDY to notify software that the endpoint needs servicing.

## Chapter 23: SuperSpeed Hub Configuration

Figure 23-1: Hub and Port Status



### Hub Port Status and Change Indicators

When hub client software detects one or more ports have incurred an event, software fetches port status information to determine what specific event(s) has occurred. Fetching port status is done via the `GetPortStatus` request that returns the contents of the Port Status and Change Indicators. In the previous example, a device was connected to port 3 resulting in changes to the Port Status and Change Indicator represented in Table 23-3 on page 530. The changes include:

- Port Change Indicator
  - Bit 0; Connect Change = 1
- Port Status
  - Bit 0; Current Connect Status = 1
  - Bits 12:10; Negotiated Speed = 000b (5Gbs)

Note: The change indicator bits prevent software from having to store previous status in order to detect which bit has changed.

# USB 3.0 Technology

*Table 23-3: Fields Returned During GetPortStatus*

Port Change Indicators																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Port Config Error	Port Link State Change	BH Port Reset Change	Port Reset Change	Over-Current Indicator Change	Reserved	Reserved	Connect Change = 1	
Reserved: all zeros returned on read																
Port Status																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved	Reserved	Reserved	Negotiated Speed: 000 = 5Gb/s 001 = Resvd 002 = Resvd 003 = Resvd 004 = Resvd 005 = Resvd 006 = Resvd 007 = Resvd	Port Power	Port Link State						Port Reset	Over-Current Indicator	Reserved	Port Enabled/Disabled	Current Connect Status = 1	

## **Chapter 23: SuperSpeed Hub Configuration**

---

Table 23-4 lists and describes each of the Port Change Indicators. These bits are normally cleared to zero and are set when a status change has occurred. After performing a GetPortStatus request, the hub client driver must clear the change indicator bits back to zero. This ensures that a subsequent event will be detected. (See page 534 for details regarding the Clear Change Indicator request.)

*Table 23-4: Port Change Indicator Definition*

<b>Bit</b>	<b>Name</b>	<b>Description</b>
0	Connect Change	This field reflects a change in the current connection state of the port. When set to one Connection Status has changed.
2:1	Reserved	
3	Port Over-current Change	This field applies only to hub ports that report over-current conditions on an individual port basis. When set to one Over-current status has changed.
4	Port Reset Change	When set to one, this bit indicates that a reset on this port has completed.
5	BH Reset Change	When set to one this bit indicates that a warm reset has completed.
6	Port Link State Change	When set to one this field indicates a change in the Port Link Status.
7	Port Config Error	This bit is set to one when the link partners are not successful in completing the Port Configuration process. This process is associated with the Link Management Packet (LMP) exchange that occurs after Flow Control Initialization.
15:8	Reserved	

## **USB 3.0 Technology**

---

Table 23-5 lists and describes the status bits that are also returned during the Get Port Status request.

*Table 23-5: Port Status Definition*

Bit	Name	Description
0	Current Connect Status	This field reflects the current connection state of the port. 0=No device attached 1=Device attached
1	Port Enabled/Disabled	Indicates whether the port is currently enabled. A port can be disabled due to fault conditions or system software 0 = Port disabled 1 = Port enabled
2	Reserved	
3	Port Over-current	Indicates that this port has incurred an over-current condition. This bit remains set until the over-current condition is removed. This bit is required for all self-power hubs. 0 = No over-current condition 1 = Over-current condition exists
4	Port Reset	Indicates that the port is reset due to a Warm or Hot Reset. 0 = No port reset 1 = Port reset is active
8:5	Port Link State	Specifies the current state of the link attached to this port. After the current state is complete a transition to the next state can be reported. 00h = U0 state 01h = U1 state 02h = U2 state 03h = U3 state 04h = SS.Disabled State 05h = Rx.Detect State 06h = SS.Inactive State 07h = Polling State 08h = Recovery State 09h = Hot Reset State 0Ah = Compliance Mode State 0Bh = Loopback State
9	Port Power	Reflects this port's power control state. Hubs can implement different power switching methods and may not represent whether power is actually applied or not. 0 = Port is in Powered-off state 1 = Port not in Powered-off state

## **Chapter 23: SuperSpeed Hub Configuration**

---

*Table 23-5: Port Status Definition (Continued)*

Bit	Name	Description
12:10	Negotiated Speed	This value indicates the highest common speed that the link partners support. This is negotiated during U0 via the Port Configuration LMP handshake 0 = 5 Gbps 1-7 = Reserved
15:13	Reserved	

### **Get Port Status Request**

The hub client uses the default control pipe to perform a GetPortStatus request to the hub. In this case the control transfer consists of:

- The Setup Stage — This setup transaction delivers 8 bytes of data that specify the GetPortStatus request.
- The Data Stage — This IN transaction is used to return port status to the hub client.
- The Status Stage — This Status transaction sent by the host controller verifies that the operation was successful, with the return of ACK.

Table 23-6 on page 534 defines the 8 bytes of data that are sent during the setup transaction. This table is used as reference information. Subsequent control transfers are compressed into a single row. The first byte (Request Type) is a bitmap that is described in Table 23-6. The other field values are also described in the table. Note that the last column within the table specifies that four bytes of data will be returned and consists of the “Port Status” and “Change Indicators.”

## **USB 3.0 Technology**

---

*Table 23-6: Setup Stage Data for GetPortStatus*

Offset	Field Name	Size	Value	Description
0	Request Type	1	Bitmap	<p>Request Characteristics</p> <p>D7 - Data Transfer Direction: 0 = Host to device 1 = Device to host</p> <p>D6:5 - Request Type: 0 = Standard 1 = Hub Class Specific 2 = Vendor 3 = Reserved</p> <p>D4:0 - Recipient 0 = Device 1 = Interface 2 = Endpoint 3 = Other (Port) 4-31 = Reserved</p>
1		1	Value (00h)	Request Code = GetStatus
2		2	Value = 0000h	Value is request dependent
4		2	Index = 0003h	Index value defines Port Number
6		2	Count = 0004h Bytes	Defines bytes to transfer Port Status and Change Indicators

### **Clear Port Requests**

Following the GetPortStatus request, the hub client must clear the Status Change bit(s) in order for the port to detect a subsequent event (e.g., port connection change). This is done via a control transfer defined as the ClearFeature command. In this example the feature field is set to clear port connection (C\_Port\_Connection). The 8 bytes of data delivered by the Setup transaction defines the ClearPortConnection request as shown in Table 23-7 on page 535. The fields are defined as follows:

- Request Type
  - Direction of transfer is from host to device
  - Request is Class Specific
  - Recipient is other (port)

## Chapter 23: SuperSpeed Hub Configuration

- Request = ClearFeature (01h)
- Value = Feature Selector (16d)
- Index = Port Number (3 in this example)
- Length = Zero (Data Stage not used)

Table 23-7: Clear Port Connection Request

Request Type (1 Byte)	Request (1 Byte)	Value (2 Bytes)	Index (2 Bytes)	Length (2 Bytes)	Data
00100011B	ClearFeature (01h)	Feature Selector (16d)	Port Number (0003h)	Zero	None

In addition to the Hub and port status change indicators, Table 23-8 lists other functions requiring change indicators.

Table 23-8: Clear Hub/Port Requests

Request Type (1 Byte)	Request (1 Byte)	Value (2 Bytes) Feature Selector	Index (2 Bytes)	Length (2 Bytes)	Data
00100000B	ClearFeature (01h)	C_Hub_Local_Power (0d)	Zero	Zero	None
00100000B	ClearFeature (01h)	C_Hub_Over_Current (1d)	Zero	Zero	None
00100011B	ClearFeature (01h)	C_Port_Connection (16d)	Port Number	Zero	None
00100011B	ClearFeature (01h)	C_Port_Over_Current (19d)	Port Number	Zero	None
00100011B	ClearFeature (01h)	C_Port_Reset (20d)	Port Number	Zero	None
00100011B	ClearFeature (01h)	C_Port_Link_State (25d)	Port Number	Zero	None
00100011B	ClearFeature (01h)	C_Port_Configuration_Error (26d)	Port Number	Zero	None
00100011B	ClearFeature (01h)	C_BH_Port_Reset (29d)	Port Number	Zero	None

# **USB 3.0 Technology**

---

## **The Hub Configuration Process**

Virtually all of the configuration process involves control transfers (via default endpoint zero) targeting the hub port and the attached device being configured. In fact, prior to a device being configured only endpoint zero is accessible. As discussed earlier some portions of the device configuration sequence may be performed differently depending on operating environment.

### **Addressing the Device**

During Reset devices force their address register to zero, thus the first access to a device will be performed using address zero. This is the device's default state. The first step in the configuration process is to assign a unique address to the device. This is accomplished with a SetAddress request that delivers a 7-bit address to the device. Table 23-9 on page 536 shows the Setup data that is delivered during the SetAddress request. This action places the device in its address state. The device can only change its address after the status stage of the control transfer has completed. Note that only one device can be configured at a time.

*Table 23-9: Set Address Request*

Request Type (1 Byte)	Request (1 Byte)	Value (2 Bytes)	Index (2 Bytes)	Length (2 Bytes)	Data
0000000B	SetAddress (05h)	Device Address (0001h)	Zero	Zero	None

---

## **Getting Descriptors**

### **General**

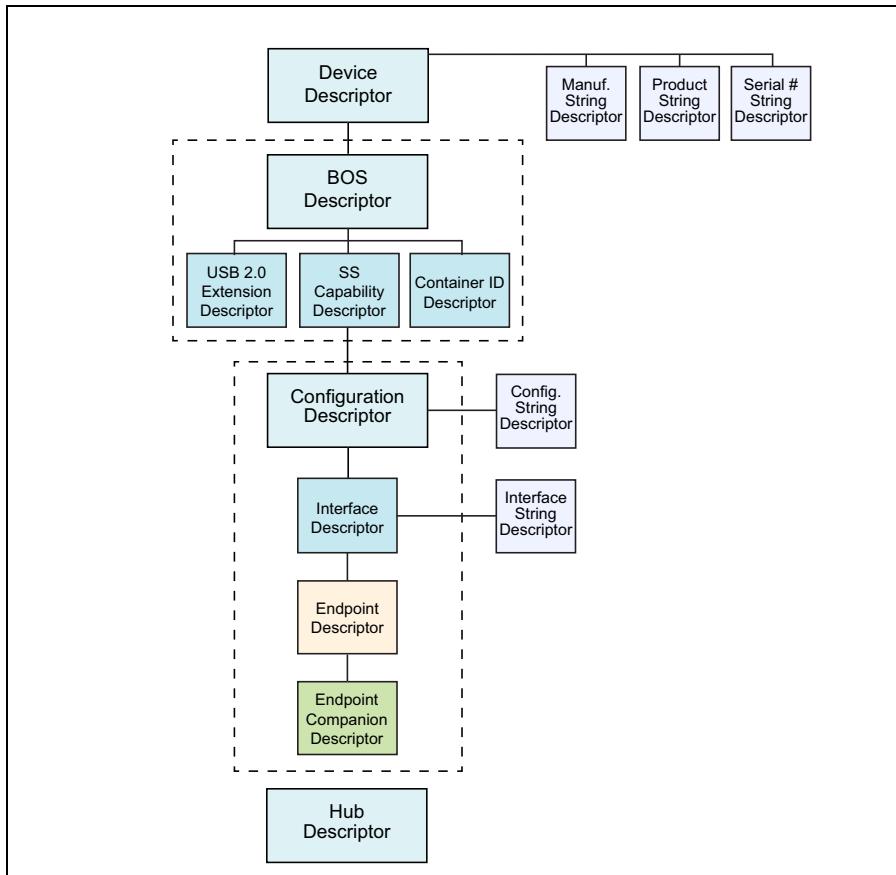
Hubs, like other devices, must also implement standard descriptors that must be read for determining the nature of the device. Standard descriptors are read by configuration software via the standard GetDescriptor request.

Hubs have a class-specific descriptor called the hub descriptor. This descriptor contains information about the hub implementation, such as number of downstream ports. The hub class descriptor is read by the hub class driver via a hub-class specific "GetDescriptor" request.

Hubs contain the following standard descriptors, as illustrated in Figure 23-2.

## Chapter 23: SuperSpeed Hub Configuration

Figure 23-2: SuperSpeed Descriptors



Configuration software performs a GetDescriptor request when accessing descriptors. Table 23-10 shows the format of the 8-byte data payload that is delivered in the setup transaction. The request field specifies GetDescriptor and the value field identifies which descriptor is being fetched. The length field defines the memory buffer size used for holding the descriptor contents.

## **USB 3.0 Technology**

---

*Table 23-10: Get Descriptor Request — Setup Data*

Request Type (1 Byte)	Request (1 Byte)	Value (2 Bytes)	Index (2 Bytes)	Length (2 Bytes)	Data
10000000B	GetDescriptor (06h)	Descriptor Type	Zero	Size of Descriptor	Descriptor Content

Table 23-11 on page 538 lists the standard descriptors, identifies the descriptor values and specifies which descriptors can be fetched directly via the GetDescriptor request. For example, the Interface, Endpoint, and Endpoint Companion descriptors cannot be fetched directly. Instead they are fetched via the Configuration descriptor, which can return all the descriptors that are part of the overall configuration. Column three in Table 23-11 specifies which descriptors can be accessed directly and which cannot.

*Table 23-11: SuperSpeed Descriptors*

Descriptor Type	Value decimal	Accessed directly?	Description
Device	1	Yes	Includes general information about the device. Reports number of Configurations, vendor and device class information
Configuration	2	Yes	Reports number of interfaces, power information and other attributes associated with this configuration. This descriptor also returns the contents of all descriptors that are part of this particular configuration
String	3	Yes	Optional UNICODE strings reporting manufacturer, serial number, device type and other human readable information
Interface	4	No	Reports number of endpoints and attributes associated with this interface
Endpoint	5	No	Reports endpoint address, direction of data and other attributes
Reserved	6 - 7	NA	

## Chapter 23: SuperSpeed Hub Configuration

Table 23-11: SuperSpeed Descriptors (Continued)

Descriptor Type	Value decimal	Accessed directly?	Description
BOS	15 (0Fh)	Yes	Binary Object Store — reports a variety on Device Capability entries, including USB 2.0 Link Power Management, SuperSpeed capabilities, and UUID value
Device Capability	16 (10h)	No	This Descriptor applies to all of the Device Capability entries included under the BOS descriptor
SS USB Endpoint Companion	48 (30h)	No	Reports new capabilities that may be used by Super-Speed devices, including Data Bursting and Bulk Streaming

### Get Device Descriptor

Table 23-12 shows the Setup data used when performing a GetDeviceDescriptor request. The 1d value resides in the upper byte of the Value field and indicates the Device descriptor is being accessed.

Table 23-12: Get Device Descriptor Request

Request Type (1 Byte)	Request (1 Byte)	Value (2 Bytes) Descriptor Type	Index (2 Bytes)	Length (2 Bytes)	Data
10000000B	GetDescriptor (06h)	Device Descriptor (1d)	Zero	12h	18 Bytes

The contents of the Device descriptor are shown in Table 23-13. The following sections discuss how host software evaluates the device descriptor during the configuration process. Some fields in Table 23-13 are not discussed further since the definition of these fields is obvious and does not benefit from additional verbiage (in the judgment of the authors).

## **USB 3.0 Technology**

---

*Table 23-13: Device Descriptor Definition*

Offset	Field	Size (bytes)	Value	Description
0	Length	1	12h	Size of this descriptor in bytes = 18d
1	Descriptor Type	1	01h	DEVICE Descriptor Type = 1
2	USB	2	300h	USB Specification Release Number in Binary-Coded Decimal (0x0300). This field identifies the release of the USB specification with which the device and its descriptors are compliant (3.0 in this example).
4	DeviceClass	1	09h	Hub Class Code
5	Device Subclass	1	00h	Hub SubClass code = 00h
6	Device Protocol	1	03h	Hub Protocol = 03h
7	MaxPacket-Size0	1	09h	Maximum packet size for endpoint zero. $2^{b\text{MaxPacketSize}0}$ $b\text{MaxPacketSize}0 = 09h$ (512B) for SS
8	Vendor	2	ID	Vendor ID (assigned by USB-IF).
10	Product	2	ID	Product ID (assigned by manufacturer).
12	Device	2	BCD	Device release number in binary-coded decimal.
14	Manufacturer	1	Index	Index of manufacturer string descriptor (optional).
15	Product	1	Index	Index of product string descriptor (optional).
16	Serial Number	1	Index	Index of device serial number string descriptor (optional).
17	NumConfigurations	1	01h	Number of configurations limited to one.

# Chapter 23: SuperSpeed Hub Configuration

---

## BOS Descriptor

The BOS (Binary Object Store) descriptor provides access to three Device Capability descriptors that are returned in order, as follows:

1. USB 2.0 Extension — This extension requires all SuperSpeed devices to support the USB 2.0 Link Power Management protocols.
2. SUPERSPEED\_USB — Provides device level capability information including exit latency values, USB speeds supported and more.
3. CONTAINER\_ID — Provides a unique ID across all operating modes.

Table 23-14 defines Setup Data associated with the Get BOS Descriptor.

*Table 23-14: Get BOS Descriptor Request*

Request Type (1 Byte)	Request (1 Byte)	Value (2 Bytes) Descriptor Type	Index (2 Bytes)	Length (2 Bytes)	Data
10000000B	GetDescriptor (06h)	BOS Descriptor (0Fh)	Zero	05h	5 Bytes

Table 23-15 shows the information contained with Hub's BOS descriptor. There are three additional Device Capability descriptors that must be implemented by Hubs. A BOS descriptor is typically accessed one time to determine the size of the buffer and a second time to fetch the BOS and Device Capability descriptors that have been implemented by the device.

*Table 23-15: BOS Descriptor Definition*

Offset	Field	Size	Value	Description
0	Length	1	05h	Size of this descriptor is 5 bytes
1	Descriptor-Type	1	0Fh	BOS descriptor type is 15
2	TotalLength	2	2Ah	Length of the BOS descriptor and all three Device Capability descriptors is 42 bytes
4	NumDevice-Capabilities	1	03h	The number of Device Capabilities implemented below the BOS descriptor

# **USB 3.0 Technology**

---

## **USB 2.0 Extension**

The first Device Capability descriptor is the USB 2.0 Extension. This descriptor relates to the requirement that all SuperSpeed devices must support at least one USB 2.0 speed. The USB 2.0 Extension specifically refers to the USB 2.0 Link Power Management (LPM) feature. All SuperSpeed devices are required to support USB 2.0 LPM when operating at either the low-, full- or high-speed operation. Table 23-16 on page 542 lists the entries associated with the USB 2.0 LPM Device Capability.

*Table 23-16: BOS — USB 2.0 Extension Descriptor*

Offset	Field	Size	Value	Description
0	Length	1	7h	Size of this descriptor is 7 bytes.
1	DescriptorType	1	10h	Device Capability descriptor type is 16.
2	DevCapability-Type	1	02h	Device Capability Type is 02h: USB 2.0 Extension
3	Attributes	4	02h	Support for device level features. A value of 1 indicates support.  Bit 0 = Reserved (zero) Bit 1 = Link Power Management (LPM) Support is required and Bit 1 must be set to 1. Bits 31:2 = Reserved and set to zero

## **SuperSpeed USB Device Capability**

This descriptor specifies whether Latency Tolerance Monitoring is supported by this device, it reports a bitmap indicating which USB 2.0 speeds are supported, specifies the lowest speed that maintains full functionality and reports U1 and U2 exit latencies back to U0.

*Table 23-17: BOS — SuperSpeed Device Capability*

Offset	Field	Bytes	Value	Description
0	Length	1	0Ah	Size of this descriptor is 10 bytes.
1	DescriptorType	1	10h	Device Capability descriptor type is 16.

## Chapter 23: SuperSpeed Hub Configuration

---

*Table 23-17: BOS – SuperSpeed Device Capability (Continued)*

Offset	Field	Bytes	Value	Description
2	DevCapabilityType	1	03h	Device Capability Type is 03h: SuperSpeed USB Device Capability
3	Attributes	1	Bitmap	<p>Support for device level features. A value of 1 indicates support.</p> <p>Bit 0 = Reserved (zero)          Bit 1 = LTM Capable, setting this bit to 1 indicates this device supports Latency Tolerance Reporting.          Bits 7:2 = Reserved (zero)</p>
4	SpeedsSupported	2	0Eh	<p>This bitmap indicates the speeds supported by the device. A one indicates support for that speed.</p> <p>Hubs must support bits 1, 2 and 3</p> <p>Bit 0 = Device supports low-speed          Bit 1 = Device supports full-speed          Bit 2 = Device supports high-speed          Bit 3 = Device supports SS (5 Gbps)          Bits 15:4 Reserved (zero)</p>
6	Functionality Support	1	1	A value of 1 indicates full-speed is the lowest fully functional speed.
7	U1DevExitLatency	1	Number	<p>The value in the field indicates the worst case latency for the device port when transitioning from U1 back to U0. The value presumes the link partner does not limit the value.</p> <p>00h = zero          01h = Less than 1µs          02h = Less than 2µs          03h = Less than 3µs          -----          0Ah = Less than 10µs          0B-FFh = Reserved</p>

## USB 3.0 Technology

---

Table 23-17: BOS – SuperSpeed Device Capability (Continued)

Offset	Field	Bytes	Value	Description
8	U2DevExitLatency	2	Number	<p>The value in the field indicates the worst case latency for the device port when transitioning from U2 back to U0. The value presumes the link partner does not limit the value. The value applies to all ports on a device (e.g., Hub)</p> <p>0000h = zero 0001h = Less than 1µs 0002h = Less than 2µs 0003h = Less than 3µs</p> <p>-----</p> <p>07FFh = Less than 2047µs 0800-FFFFh = Reserved</p>

### Container ID

This device level descriptor must be implemented by all hubs and is optional for other devices. The Container ID value is a 128-bit number defined as a Universal Unique Identifier (UUID). This value is intended to be universal in that it applies to all modes of use (e.g., a hub operating in SuperSpeed and in USB 2.0 modes). The UUID also applies to devices that implement connections other than USB. For example, a USB Disc Drive that also has a Firewire connector. Table 23-18 depicts the content of the Container ID descriptor.

Table 23-18: BOS – Container ID

Offset	Field	Size	Value	Description
0	Length	1	14h	Size of this descriptor is 20 bytes.
1	DescriptorType	1	10h	Device Capability descriptor type is 16.
2	DevCapability-Type	1	04h	Device Capability Type is 04h: ContainerID
3	Reserved	1	Zero	Reserved
4	ContainerID	16	UUID	A 128-bit unique number that can be used universally across all device instances and modes of operation.

# Chapter 23: SuperSpeed Hub Configuration

---

## Configuration Descriptor

The Configuration descriptor has the same definition as it does for standard USB devices. Configuration software accesses this descriptor to obtain the following information:

- number of interfaces supported by this configuration
- the configuration value used to configure the hub
- whether the hub is bus powered or self powered
- maximum amount of bus power consumed by the hub in this configuration

Table 23-9 on page 536 shows the Setup data that is delivered during the Get-Configuration Descriptor request. The Configuration descriptor fetches all the descriptors that are part of the configuration. Consequently, software does not know the total size of all the descriptors that must be fetched. The Configuration descriptor must be read the first time to get the **Total Length** (See Offset 2 in Table 23-20 on page 545). This value specifies the buffer size needed to hold all of the descriptors associated with the configuration.

Table 23-19: Get Configuration Descriptor Request

Request Type (1 Byte)	Request (1 Byte)	Value (2 Bytes)	Index (2 Bytes)	Length (2 Bytes)	Data
10000000B	GetDescriptor (06h)	Descriptor Type (2d)	Zero	1st access (09h) 2nd access (total length)	9 Bytes total length

Table 23-20 defines the standard configuration descriptor fields as implemented by a hub device.

Table 23-20: Hub's Configuration Descriptor

Offset	Field	Size	Value	Description
0	Length	1	09h	Size of this descriptor is 9 bytes.
1	Descriptor-Type	1	02h	CONFIGURATION.
2	TotalLength	2	31	Total length of data returned for this configuration. Includes the combined length of all descriptors (configuration, interface, endpoint, and class or vendor specific) returned for this configuration. Total length is 31 bytes

# USB 3.0 Technology

---

Table 23-20: Hub's Configuration Descriptor (Continued)

Offset	Field	Size	Value	Description
4	Num Interfaces	1	01h	Number of interfaces supported by this configuration is one. This interface is the Status Change Endpoint.
5	Config Value	1	Number	Value to use as an argument to Set Configuration to select this configuration.
6	Config Index	1	Index	Index of string descriptor describing this configuration.
7	Attributes	1	Bitmap	<p>Configuration characteristics</p> <p>D7 Must be 1 D6 Self Powered D5 Remote Wakeup D4:0 Reserved (reset to 0)</p> <p>A device configuration that uses power from the bus and a local source at runtime may be determined using the GetStatus device request.</p>
8	MaxPower	1	8mA increments	Maximum amount of bus power this hub will consume in this configuration when the device is fully operational. This value includes the hub controller, all embedded devices, and all ports (value based on 8mA increments).

## Configuration Value

The configuration value at offset 5 is the number used when configuring the device via the *Set Configuration* request. This also enables access to the hub's Status Change endpoint.

## Bus- or Self-Powered Hub

Offset 7 within the configuration descriptor is defined as device attributes. These attributes specify whether the hub is self-powered, and bus powered. If both bit fields are set, then the hub is a hybrid-powered device. If the hub is bus-powered, the maximum bus power field can be used by configuration software to determine the amount of power available at each port. See details regarding issues related to bus- and self-powered hubs.

# Chapter 23: SuperSpeed Hub Configuration

---

## Maximum Bus Power Consumed

This field is only valid for hubs that are bus- or hybrid-powered. It specifies the maximum amount of current this hub will draw from the USB 3.0 bus. Configuration software can use this value to deduce the maximum amount of current available at each port. This value includes power consumed by pull-up resistors, hub controller, all embedded devices, and all ports.

---

## Hub's Interface Descriptor

The Interface descriptor is returned as part of the GetConfigurationDescriptor request. An Interface typically defines a collection of endpoints that define a function or class. The software driver that manages the function is called a Class Driver, that initializes and controls the function. In the hub application a single interrupt endpoint defines the single function, called the Status Change endpoint. This endpoint is used by the Hub Client Driver that handles both hub and hub port events. Only one interface descriptor is allowed for Hubs. Table 23-21 on page 547 shows the fields of the Interface Descriptor.

*Table 23-21: Hub Interface Descriptor*

Offset	Field	Size	Value	Description
0	Length	1	09h	Size of this descriptor in bytes.
1	DescriptorType	1	04h	INTERFACE Descriptor Type = 4.
2	InterfaceNumber	1	00h	Must be zero for hubs
3	AlternateSetting	1	00h	Alternate interfaces no allowed for hubs
4	NumEndpoints	1	01h	Number of endpoints used by this interface (excluding endpoint zero) is 1
5	InterfaceClass	1	09h	Class code assigned by USB-IF is 9.
6	InterfaceSubclass	1	00h	Subclass code assigned by USB-IF is zero for hubs
7	InterfaceProtocol	1	00h	Protocol code assigned by USB is zero for hubs.
8	Interface	1	Number	Index of string descriptor.

# **USB 3.0 Technology**

---

## **Status Change Endpoint Descriptor**

### **General**

This descriptor is returned in the GetConfiguration descriptor request. The hub function defines a single endpoint called the status change endpoint. Table 23-5 shows the definition of the status endpoint descriptor. Configuration software obtains the following information from the endpoint descriptor:

- Status change endpoint address
- Direction of transfer
- Transfer type supported by this endpoint
- Maximum data packet size supported
- Interval at which the endpoint should be accessed

*Table 23-22: Hub Status Change — Endpoint Descriptor*

Offset	Field	Size	Value	Description
0	Length	1	07h	Size of this descriptor is 7 bytes.
1	DescriptorType	1	05h	Endpoint descriptor type is 5
2	Endpoint Address	1	81h	The endpoint number for the hub's Status Change Endpoint is one and the direction is always IN, as follows:  Bit 3:0 Endpoint number = 1 Bit 6:4 Reserved, reset to zero Bit 7 Direction: 0 = OUT endpoint 1 = IN endpoint

## Chapter 23: SuperSpeed Hub Configuration

---

*Table 23-22: Hub Status Change – Endpoint Descriptor (Continued)*

Offset	Field	Size	Value	Description										
3	Attributes	1	13h	<p>This field describes the Hub Status Change endpoint attributes. The EP type is interrupt and the usage type is Notification. The field value is 19 decimal.</p> <p>Endpoint Transfer Types</p> <table> <tr><td><b>Bits 1:0 Transfer Type</b></td></tr> <tr><td>00 Control</td></tr> <tr><td>01 Isochronous</td></tr> <tr><td>10 Bulk</td></tr> <tr><td><b>11 Interrupt</b></td></tr> </table> <p>Interrupt Endpoints:</p> <table> <tr><td><b>Bits 3:2 Reserved</b></td></tr> </table> <p><b>Bits 5:4 Usage Type</b></p> <table> <tr><td>00 Periodic</td></tr> <tr><td><b>01 Notification</b></td></tr> <tr><td>10 Reserved</td></tr> <tr><td>11 Reserved</td></tr> </table> <p><b>Bits 7:6 Reserved (must be zero)</b></p>	<b>Bits 1:0 Transfer Type</b>	00 Control	01 Isochronous	10 Bulk	<b>11 Interrupt</b>	<b>Bits 3:2 Reserved</b>	00 Periodic	<b>01 Notification</b>	10 Reserved	11 Reserved
<b>Bits 1:0 Transfer Type</b>														
00 Control														
01 Isochronous														
10 Bulk														
<b>11 Interrupt</b>														
<b>Bits 3:2 Reserved</b>														
00 Periodic														
<b>01 Notification</b>														
10 Reserved														
11 Reserved														
4	MaxPacketSize	2	2 Bytes	<p>Maximum packet size for this endpoint.</p> <p>Isochronous endpoints</p> <ul style="list-style-type: none"> <li>— 1024 Bytes when MaxBurst &gt; zero</li> <li>— <b>0-1024 Bytes when MaxBurst = zero</b></li> </ul>										
6	bInterval	1	Number=8	<p>Interval for servicing endpoint. Expressed in 125µs units.</p> <p>Values are specified by this formula:  <math>2^{b\text{Interval}-1}</math></p> <p><b>bInterval value:</b>  8 - Notification Interrupt EPs</p>										

### Hub Status Change Endpoint Address/Transfer Direction

The status change endpoint number is specified within the field at offset 2 of the endpoint descriptor. This field specifies the endpoint number (bits 0:3), which is determined at design time and hardwired into the device's address decoder and is implementation dependent. Bit 7 within the same field specifies the direction of the data transfers associated with the endpoint. This bit must be set to indicate that the direction of transfers is from the device (Status Change Endpoint) to the host (i.e., IN transactions).

# USB 3.0 Technology

---

## Status Change Endpoint Companion Descriptor

The Endpoint descriptor as defined by USB 2.0 has insufficient space for reporting new features associated with SuperSpeed USB. Consequently, the Endpoint Companion descriptor was implemented to report new capabilities. However, the status change endpoint transfers single transaction requiring no more than 16-byte payload, so it does not need bursting nor any of the other features. Table 23-23 describes the less than exciting content of the Hub's Endpoint Companion descriptor.

*Table 23-23: Hub's Endpoint Companion Descriptor Definition*

Offset	Field	Size	Value	Description
0	Length	1	06h	Size of this descriptor is 6 bytes.
1	Descriptor-Type	1	30h	SuperSpeed Endpoint Companion descriptor type is 48.
2	MaxBurst	1	Number = 0	Maximum Burst Size for Hub Status Change endpoint = 0
3	Attributes	1	Bitmap = 0	Interrupt Endpoint: Bits 7:0 Reserved (zeros)
4	BytesPerInterval	2	Number = 2	Periodic endpoints only — Total bytes transferred during service interval. Used for reserving time for scheduling isochronous traffic. This field is set to 2 for the Status Change EP.

---

## String Descriptors

String Descriptors are associated with the following descriptor types and all are limited to a single string descriptor except for the Device Descriptor:

- Device Descriptor — Up to three string descriptors can be implemented (Manufacturer, Product, & Serial Number String Descriptors).
- Configuration Descriptor
- Interface Descriptor(s)

The Hub's implementation of string descriptors is exactly the same as other devices. Please see "String Descriptors" on page 519 for details.

# **Chapter 23: SuperSpeed Hub Configuration**

---

## **Additional Requests**

Configuration software may perform other requests during the configuration process. These requests include enabling features and forwarding information to devices (including hub devices):

- SEL (System Exit Latency) — the SEL request specifies the SEL and PEL (Path Exit Latency) information for U1 and U2. This information can be used by SuperSpeed devices to adjust the Latency Tolerance Message., if supported. See Chapter 24, entitled "SuperSpeed Power Management," on page 563 for details.
- LTM\_ENABLE (Latency Tolerance Message enable) — this enables a device to generate Latency Tolerance Message to the host to reduce system power consumption and increase transaction latency. See previous bullet item.

---

## **Hub Driver Initialization**

Once a hub has been configured, the hub client driver completes the initialization process, thereby preparing the hub and hub ports for normal operation. The first action typically taken after configuration is evaluating the Hub Class descriptor.

---

## **Hub Class Descriptor**

A class-specific descriptor is defined for hub devices. This descriptor is read via the hub class-specific “GetDescriptor” request. Table 23-24 on page 551 lists the Setup Data that defines the Class-specific Hub descriptor.

*Table 23-24: Get Hub Class Descriptor Request*

Request Type (1 Byte)	Request (1 Byte)	Value (2 Bytes) Descriptor Type	Index (2 Bytes)	Length (2 Bytes)	Data
10100000B	GetDescriptor (06h)	Hub Class (2A00h)	Zeros	000Ch	12 Bytes

## **USB 3.0 Technology**

---

The Hub Client driver reads the hub class descriptor to determine the following information:

- Power switching mode implemented.
- Whether hub is part of compound device or not.
- Whether device implements ganged, individual port, or no over-current protection.
- The time delay from software requesting power be applied to a port until power is valid.
- Maximum bus current required by the hub controller.
- Whether the device attached to a port is removable or not.
- Whether a port is powered in ganged mode or individual ports.

The hub class descriptor fields are defined in Table 23-25 and include brief descriptions of each field. Following the table are more complete descriptions of selected fields.

*Table 23-25: Hub Class Descriptor*

Offset	Field	Size	Description
0	DescLength	1	Number of bytes in the descriptor, including this byte.
1	DescriptorType	1	Descriptor Type = 2Ah (hub class descriptor).
2	NbrPorts	1	Number of downstream ports that this hub supports = 15
3	HubCharacteristics	2	D1:D0 Power Switching Mode. 00 Ganged power switching (all ports powered at once) 01 Individual port power switching 1X Reserved D2 Identifies a Compound Device 0 Hub is not part of a compound device 1 Hub is part of a compound device D4:D3 Over-current Protection Mode. 00 Global Over-Current Protection. The hub reports over-current as a summation of all ports' current draw. 01 Individual Port Over-Current Protection. The hub reports over-current on a per port basis. Each port has an over-current indicator. 1X No Over-Current Protection. This option is only allowed for bus-powered hubs that don't implement over-current protection.  D15:D5 Reserved

## Chapter 23: SuperSpeed Hub Configuration

---

*Table 23-25: Hub Class Descriptor (Continued)*

Offset	Field	Size	Description
5	PwrOn2PwrGood	1	Time (in 2ms intervals) from the time power-on sequence begins on a port until power is good on that port. System software uses this value to determine how long to wait before accessing a powered-on port. A zero value indicates the hub does not support power switching.
6	HubContrCurrent	1	<p>Maximum current requirements for the SuperSpeed and USB 2.0 hub controller electronics, expressed as a Current Unit or 4mA. The value reported in this field is multiplied by 4mA (i.e., a value of 55 * 4mA = 220mA).</p> <p>When the USB 3.0 hub is operating in USB 2.0 mode only, hub controller current requirements are reported in the USB 2.0 hub descriptor.</p>
7	Hub Header Decode Latency (HubHdrDecLat)	1	<p>Header Decode Latency (Downstream Packets only)</p> <p>The Hub's upstream port is in U0 and the target downstream Hub port is in a low-power link state. A HEADER or DATA packet is received on the upstream hub port and is forwarded to the target downstream port. The value reported is the worst case exit latency beginning with the last symbol received on the upstream port until the hub starts LFPS on the downstream port.</p> <p>00h = Much less than 0.1μs      01h = 0.1μs      02h = 0.2μs      03h = 0.3μs      04h = 0.4μs      05h = 0.5μs      06h = 0.6μs      07h = 0.7μs      08h = 0.8μs      09h = 0.9μs      0Ah = 1.0μs      0Bh - 0Fh = Reserved</p>

# **USB 3.0 Technology**

---

*Table 23-25: Hub Class Descriptor (Continued)*

Offset	Field	Size	Description
8	HubDelay	2	<p>This field defines the average nanosecond delay introduced by the hub when packet are moving in either direction and the links are in the U0 state. The value reported is the time between the last symbol of a packet received and the first symbol at the transmitting port.</p> <p>Additional restrictions include:</p> <ul style="list-style-type: none"><li>• No other packets of any kind are in flight</li><li>• Flow Control Credit Count is not zero at the transmitting port</li><li>• The transmitting port has no packets in it's transmit buffer</li></ul> <p>Maximum allowed value in this field is: tHubDelay (400ms)</p>
10	DeviceRemovable	2	<p>This bit map specifies if a given port has a removable connection. A value of 1 indicates a removable port connection and 0 indicates a permanent port connection.</p> <p>The format of the bit map is implemented as follows:</p> <p>Bit 0 — Reserved Bit 1 — Port 1 Bit 2 — Port 2 ----- Bit 15 — Port 15</p>

## **Power Switching Mode Implemented**

A hub may control power to ports in two ways. Offset 3, bits D1:D0, of the hub class descriptor defines which method is employed by the current hub:

- Ganged power switching — power is switched to all ports at the same time.
- Individual port power switching — power is applied to each port separately. The “Set Port Power” Feature request applies power to the individual port that has been selected.

The specification also allows self-powered hubs to apply power to all downstream ports without a power-switching request. The power can be available as long as the hub’s power supply is on. The intent is to support applications like battery charging from the USB ports.

# **Chapter 23: SuperSpeed Hub Configuration**

---

## **Over-Current Protection Mode**

A hub may choose to implement over-current protection in different ways, as long as it conforms to the safety requirement of allowing no more than 5 Amps of current to be drawn by a given port. Offset 3, bits D4:D3, specify which over-current protection mode is implemented for this hub.

## **Maximum Bus Current for Hub Controller**

The Configuration descriptor defines the Maximum power consumption of the USB 3.0's hub controller (both sides) when it is fully operational. If a hub is bus powered, the hub client driver must know how much current is available for the downstream ports. Offset 6 within the hub class descriptor specifies the maximum amount of current that the hub controller consumes, thus the hub client driver can deduct hub controller current from the total to get the amount of current available to the ports.

## **Device Removable/Non-removable**

The DeviceRemovable field at offset 7 provides a bitmap of all ports supported by this hub. Each bit position corresponds to a given port (i.e., bit 1 specifies port 1). If the bit field corresponding to a port is cleared, the device is removable, and if the bit is set, the device is permanently attached (e.g., an embedded device). The field size is one byte for hubs that support from one to seven ports. Another byte must be added to support the maximum of 15 ports.

---

## **Hub-Specific Requests**

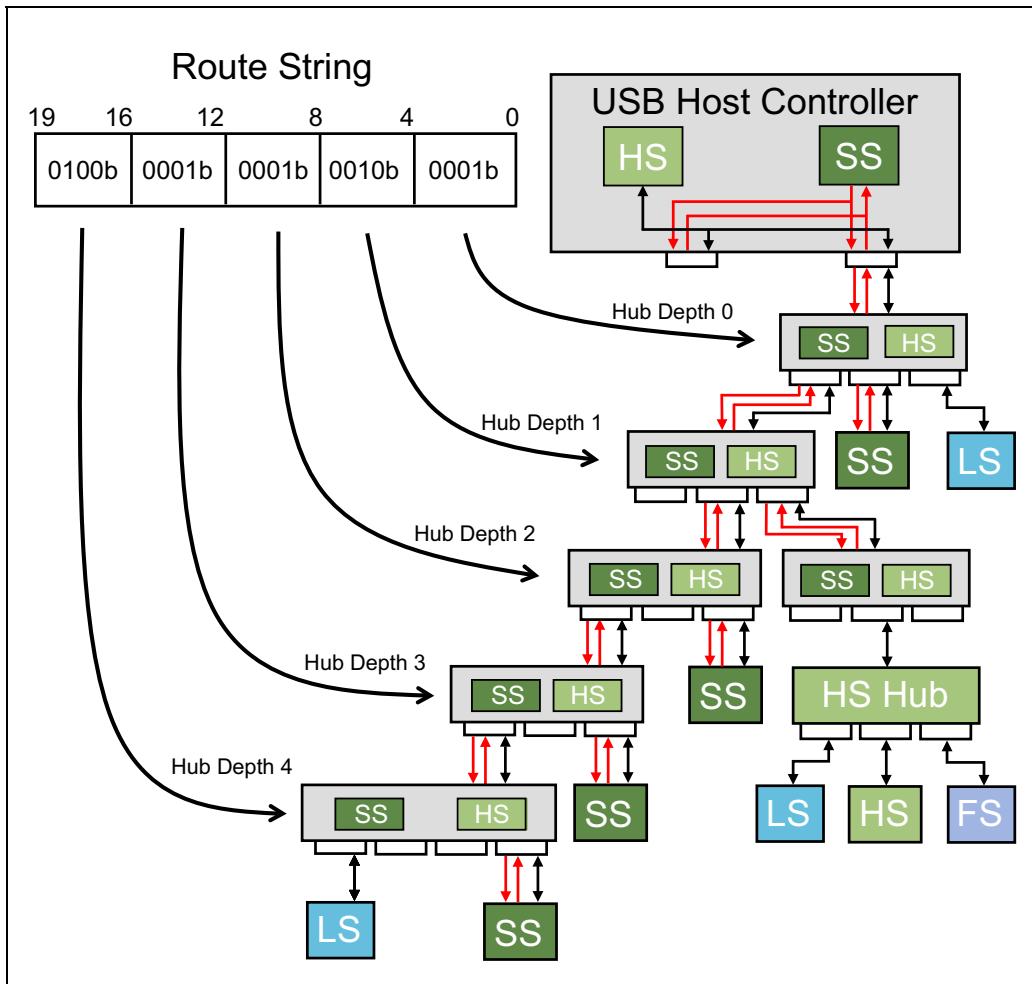
The Hub client software is responsible for setting up and enabling a variety of features that are managed via hub-specific requests. Some of the requests are required and some are optional. Each of the requests are defined in the following sections.

### **Set Hub Depth**

Each Hub must know the depth at which it resides in the topology so it can identify which nibble of the routing table it should use. Host software assigns the Header packet route string during device configuration. A Hub Depth value is assigned to each hub based on its position in the USB hierarchy. Figure 23-3 on page 556 is a reminder of the USB 3.0 topology and hub depth values of 0-4. This permits hubs to index into the proper nibble within the route string. Hub Depth values range from 0 to 4 (i.e., Hub Depth Value x 4) as listed in Figure 23-3.

## USB 3.0 Technology

Figure 23-3: Hub Depth Assignment



## **Chapter 23: SuperSpeed Hub Configuration**

---

The hub class-specific “Set Hub Depth” request in Table 23-26 on page 557 lists the Setup Data that defines this request.

*Table 23-26: Set Hub Depth Request*

Request Type (1 Byte)	Request (1 Byte)	Value (2 Bytes) Descriptor Type	Index (2 Bytes)	Length (2 Bytes)	Data
00100000B	Set Hub Depth (12h)	Hub Depth (value 0 - 4)	Zeros	Zeros	None

### **Port U1/U2 Timeout**

U1/U2 Inactivity Timers trigger a link to enter a low-power link state (U1 or U2) to conserve power. Details associated with these timers can be found in “Software Role in Link Power Management” on page 569. The Hub-Specific request is listed in Table 23-27.

*Table 23-27: Set Port U1/U2 Timeout*

Request Type (1 Byte)	Request (1 Byte)	Value (2 Bytes) Descriptor Type	Index (2 Bytes)	Length (2 Bytes)	Data
00100011B	Set Feature (03h)	Feature Selectors U1 Timeout = 23 U2 Timeout = 24	Selected Timeout Value	Zeros	None

### **Set Port Remote Wake Mask**

A hub may signal a wakeup event on its upstream port if one or more downstream ports have been conditioned to trigger the wakeup. Three specific events can cause the trigger when enabled:

- Device Connected
- Device Disconnected
- Over Current Condition

## **USB 3.0 Technology**

---

The Set Port Remote Wake Mask permits software to selectively enable or mask a port's ability to respond to these events. The Index field highlights the events and reminds us that any event can be masked or enabled.

*Table 23-28: Set Port Remote Wake Mask*

Request Type (1 Byte)	Request (1 Byte)	Value (2 Bytes) Descriptor Type	Index (2 Bytes)	Length (2 Bytes)	Data
00100011B	Set Feature (03h)	Feature Selectors (27)	Connect 0 = mask Disconnect 1 = enable OverCurrent	Zeros	None

### **Port Link State**

Software may choose to transition a link using values from 0 - 5 (all other values are invalid).

- A value of 0 forces the link to the U0 state
- Values of 1 - 3 cause transitions to U1, U2, and U3 respectively
- A value of 4 causes a transition to the Disabled state
- A value of 5 causes a transition to the RxDetect state

*Table 23-29: Set Port Link State*

Request Type (1 Byte)	Request (1 Byte)	Value (2 Bytes) Descriptor Type	Index (2 Bytes)	Length (2 Bytes)	Data
00100011B	Set Feature (03h)	Feature Selector (0-5)	Zeros	Zeros	None

### **Set Port Power**

Once a hub has been configured, power may need to be applied to the downstream ports. Configuration software detects the port power mode when reading the hub class descriptor. If power must be applied manually, then the software issues a "SetPortPower" request usually targeting a specific port. However, a hub may be designed to apply power to all ports simultaneously (ganged) when the "SetPortPower" request is performed.

## Chapter 23: SuperSpeed Hub Configuration

---

Table 23-30: Set Port Power

Request Type (1 Byte)	Request (1 Byte)	Value (2 Bytes) Descriptor Type	Index (2 Bytes)	Length (2 Bytes)	Data
00100011B	Set Feature (03h)	Feature Selector (8d)	Port Number	Zeros	None

### Port Reset

Software may trigger a Reset event by delivering a Set Port Reset Request to a downstream root or hub port. The action taken depends on the current state of the port. See Chapter 19, entitled "SuperSpeed Reset Events," on page 413 for details regarding Port Reset. Table 23-31 depicts the Set Port Reset request.

Table 23-31: Set Port Reset State

Request Type (1 Byte)	Request (1 Byte)	Value (2 Bytes) Descriptor Type	Index (2 Bytes)	Length (2 Bytes)	Data
00100011B	Set Feature (03h)	Feature Selector (4d)	Zeros	Zeros	None

### BH Port Reset

Table 23-32 shows the specific request data required to deliver a BH Port Reset request to a target port. When the BH Port Reset request is delivered to a port, the port initiates a Warm Reset sequence. See "How Warm Reset Is Signaled On The Link" on page 417 for details.

Table 23-32: Set BH Port Reset

Request Type (1 Byte)	Request (1 Byte)	Value (2 Bytes) Descriptor Type	Index (2 Bytes)	Length (2 Bytes)	Data
00100011B	Set Feature (03h)	Feature Selector (0028d)	Port	Zeros	None

# **USB 3.0 Technology**

---

## **Hub Status Change Endpoint Request**

One or more port status bits get set when a hub port change is detected. This triggers a status change event causing the hub to deliver an ERDY header, thus notifying software to check the status. Configuration software polls the status change endpoint to detect which ports have incurred a status change event. The status change endpoint merely reports whether the hub or a hub port has incurred a status change but does not identify the nature of the change. Specific requests must be made to the hub or port to determine the:

1. Specific event(s) that has occurred.
2. Current state of the item causing the event.

---

## **Reading the Hub Status Field**

If a hub status change is detected, software can perform the *Get Hub Status* request to obtain the source of the change. Hub status changes consist of:

- Local power change
- Over-current change

Once the specific hub status item that has changed is detected, software can take the appropriate action (e.g., by notifying the user of the event). Software must also acknowledge and clear the hub status change field by using the related ClearHubFeature request. For example, if a local power change has occurred, software uses the ClearHubLocalPower request.

---

## **Reading Hub Port Status**

When a port status change occurs, software uses the “Get Port Status” request to determine which port feature or features have experienced a change. The possible sources of port changes are:

- **Connect Status Change** — device either connected or disconnected from port
- **Port Enable/Disable Change** — change caused by hardware event
- **Suspend Change** — changed indicated when resume has completed
- **Over-Current Indicator Change** — used only by hubs that report over-current on a per port basis
- **Reset Change** — change set when reset processing is completed

## Chapter 23: SuperSpeed Hub Configuration

---

Consider the following example. A Connect Status Change is indicated when power is applied to a port that currently has a device attached. The port status change field will be set and the current status field will indicate that a device is currently attached. Configuration software recognizes that a device has been connected and acknowledges the change by performing a “Clear Port Feature” request.

---

### Summary of Hub Feature Selectors

The following table is a general summary of the hub port states and the transitions that take place for given signaling events or control requests. See the 3.0 specification for a complete listing of port states and transitions.

*Table 23-33: Hub Feature Selectors*

Feature Selector	Recipient	Value
C_HUB_LOCAL_POWER	Hub	0
C_HUB_OVER_CURRENT	Hub	1
PORT_CONNECTION	Port	2
PORT_OVER_CURRENT	Port	3
PORT_RESET	Port	4
PORT_LINK_STATE	Port	5
PORT_POWER	Port	8
C_PORT_CONNECTION	Port	16
C_PORT_OVER_CURRENT	Port	19
C_PORT_RESET	Port	20
RESERVED (USED IN USB 2.0)	Port	21
PORT_U1_TIMEOUT	Port	23
PORT_U2_TIMEOUT	Port	24
C_PORT_LINK_STATE	Port	25
C_PORT_CONFIG_ERROR	Port	26

## **USB 3.0 Technology**

---

*Table 23-33: Hub Feature Selectors (Continued)*

<b>Feature Selector</b>	<b>Recipient</b>	<b>Value</b>
POR T_REMOTE_WAKE_MASK	Port	27
BH_PORT_RESET	Port	28
C_BH_PORT_RESET	Port	29
FORCE_LINKPM_ACCEPT	Port	30

---

---

# **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**

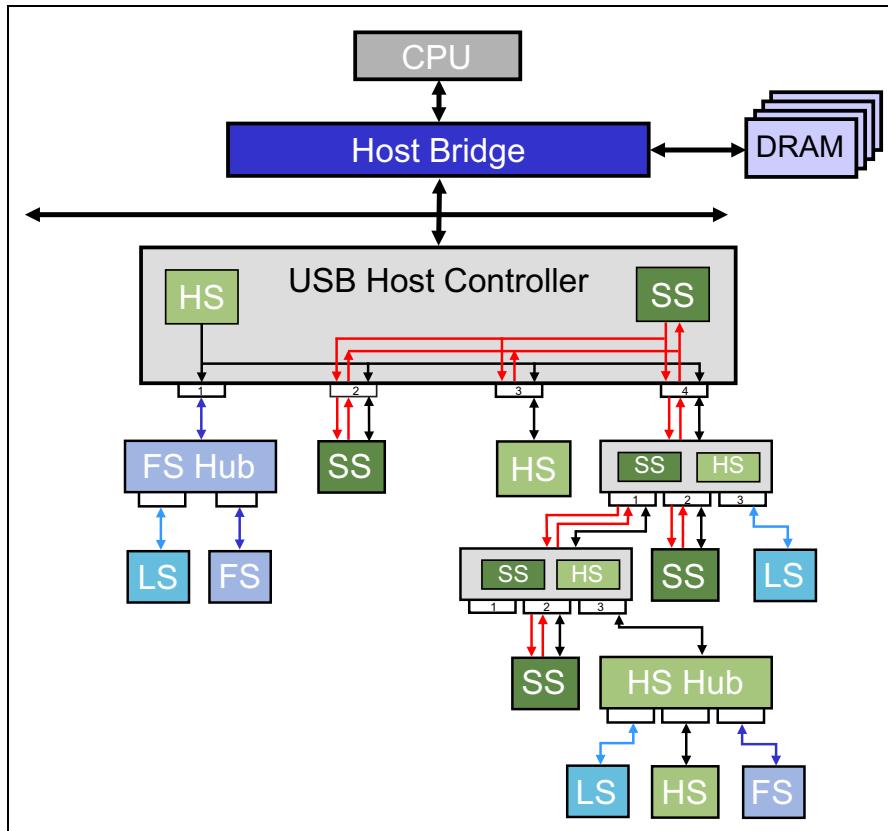
This section provides background information, concepts and principles associated with Link Power Management. Later chapter discuss the various mechanisms defined by the specification for managing link and device power.

# USB 3.0 Technology

## 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



# **Chapter 24: SuperSpeed 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.

# **USB 3.0 Technology**

---

## **Function Power Management**

This feature permits individual functions within a single device to be suspended under software control. The link must remain active to handle other functions within the device.

### **Suspend/Resume (U3)**

USB 3.0 SuperSpeed root ports and external hub ports supply 5-volt power to all devices with current ranging from 150 mA to 900 mA. The fundamental method for conserving bus power is the software initiated suspend state (also used by the USB 2.0 bus). When in the suspend state devices must not sink more than 2.5 mA of current from the bus.

---

## **SuperSpeed Link Power Management**

---

### **The Link Power States**

The Link Training Status State Machine (LTSSM) defines the states and state transitions associated with the link interface power. The four power states are:

- U0: Link and Link Partners are fully powered and in the operational state.
- U1: Link has transitioned to “Standby” with fast recovery back to U0.
- U2: Link has transitioned to “Standby” and is conserving more power than U1, with slow recovery back to U0.
- U3: Link has transitioned to “Suspend” (the lowest power state) and power consumption is reduced to no more than 2.5 mA, with very slow recovery back to U0. Only software can cause a transition to U3.

Figure 24-2 on page 567 illustrates the LTSSM, the related power states and the typical transitions for entering and exiting the low power states. There are a variety of triggers that may cause a transition from U0 to U1, U2 or U3. A trigger may be initiated by either link partner (except U3 which requires software) and always results in a link-layer handshake. The handshake either completes causing a transition to the selected low-power state or results in a rejection by the receiving device. Exit from the low-power state back to U0 requires two handshakes:

- Low Frequency Periodic Signaling (LFPS) handshake between the link partners, followed by
- TS1 ordered set handshake between the link partners, causing a transition to the Recovery state where link communications are re-established, resulting in the transition back to U0.

Table 24-1 on page 567 summarizes the link power state conditions.

## Chapter 24: SuperSpeed Power Management

Figure 24-2: LTSSM and Link Power States and Transitions

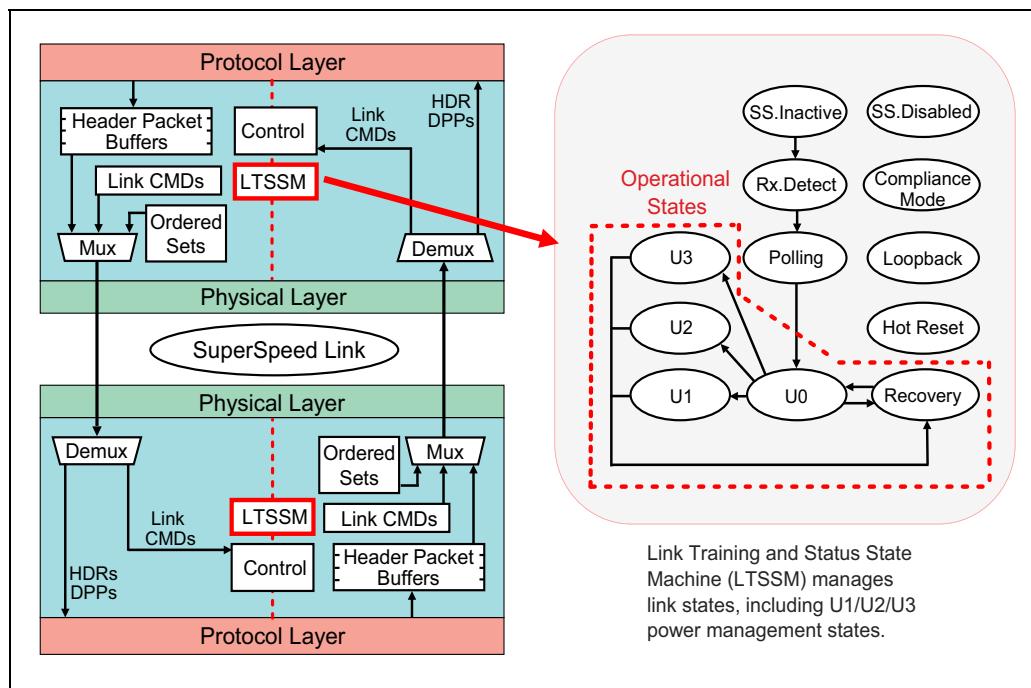


Table 24-1: Link Power State Conditions

Link State	Description	Characteristics	State Transition Initiator	Device Clock Gen on/off	Typical Exit Latency Range
<b>U0</b>	Link Active	Operational State	NA	On	NA
<b>U1</b>	Link idle Fast exit	Rx and Tx circuitry quiesced	Hardware	On or Off	$\mu$ s
<b>U2</b>	Link idle Slower exit	Clock gen circuitry may be quiesced	Hardware	On or Off	$\mu$ s - ms
<b>U3</b>	Link Suspend	Some device power may be removed, need Wake & Warm Reset	Entry: software only Exit: Hardware or Software	Off	ms

# USB 3.0 Technology

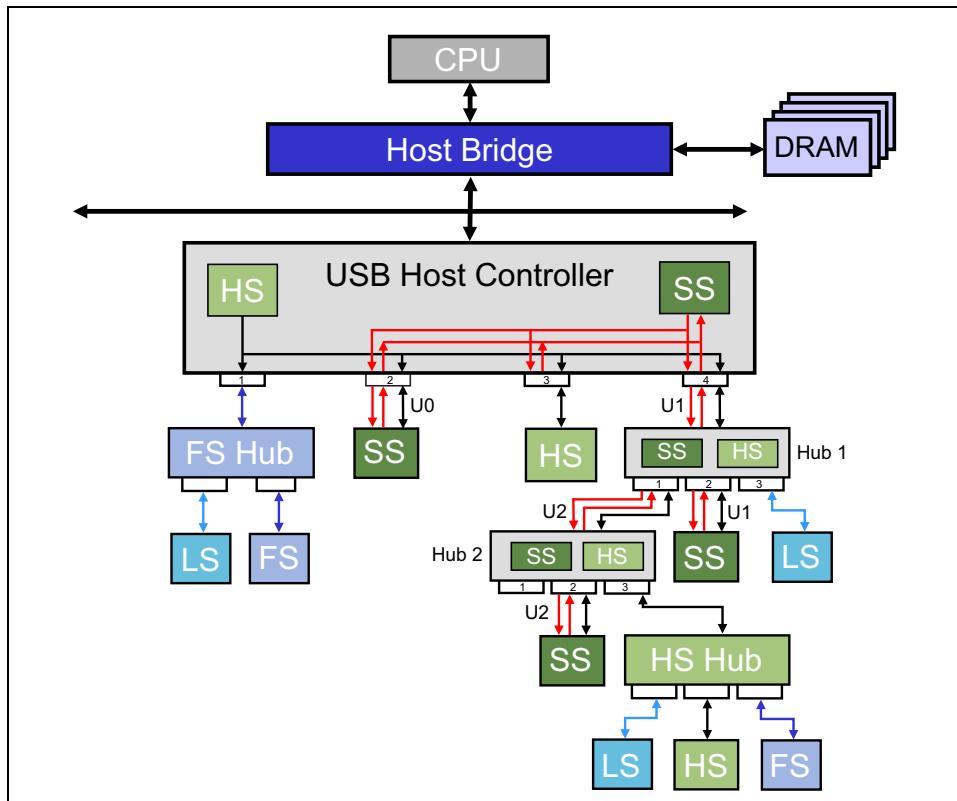
## Link-Power Hierarchy

Figure 24-3 on page 568 illustrates the hierarchical nature of link power management. In short, a hub must never allow its upstream link to enter a power state that is lower than any of its downstream links. The example in Figure 24-3 lists power states (U0, U1 or U2) for each SS link and labels each root hub port and each USB 3.0 downstream hub port for reference.

Notice that Hub 2 has a single SS device attached with a link power state of U2. The upstream port of Hub 2 must not be in a lower power state than U2. Moving upstream to hub 1 shows port 1 in the U2 state and port 2 in the U1 state. This means that the upstream port of hub 1 must not be in a lower power state than U1.

The entire link power hierarchy that must be considered in the topology includes: U0, U1, U2, U3, Rx.Detect and SS.Disabled.

*Figure 24-3: Link Power State Hierarchy*



# **Chapter 24: SuperSpeed Power Management**

---

## **Software Role in Link Power Management**

SuperSpeed power management provides three methods for triggering transitions to low-power link states. These are:

1. Host Software
2. Hub Port Inactivity Timers Expire (initialized by software)
3. Device-Specific Algorithms or Timers Expire (enabled by software)

The most direct method of managing transitions to low-power states is software commands. However, the hardware mechanisms (Hub and Devices) typically provide more efficient power transitions, and Host Software must initialize and enable the hardware mechanisms before they can trigger a transition. All of the triggering mechanisms, initiate a handshake between the link partners, that typically results in transitions to a low-power link state. In some cases, a triggering mechanism may not have visibility to pending transactions that must be performed. In these cases, the handshake is cancelled and the pending transactions are performed. Subsequent sections detail each mechanism.

---

### **Host Software Triggers**

Software uses the Set\_Port\_Feature (PORT\_LINK\_STATE) request to transaction links to the low-power states and back. These requests target only root and external hub ports. When a PORT\_LINK\_STATE request is made, software passes a value indicating the power state to which the target port will transition (See Table 24-2).

*Table 24-2: Port Link State Transitions*

Requested Value	Transition
1	U0 → U1
2	U0 → U2
3	U0 → U3
0	U1 → U0 U2 → U0 U3 → U0

# USB 3.0 Technology

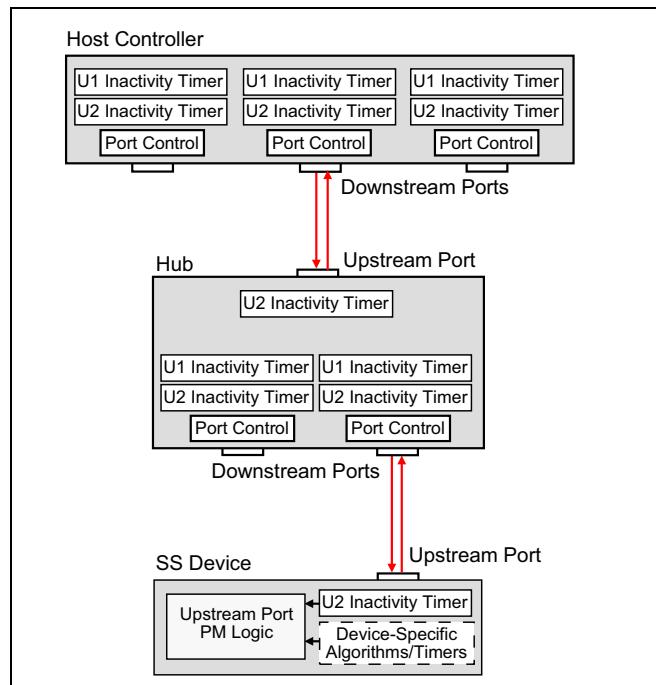
The specification suggests that software's role in normal circumstances would be limited to transitioning power to and from U3, while software transitions to and from the U1 and U2 states are intended only for test purposes.

## **Hub Inactivity Timers**

The primary triggering mechanisms used to initiate entry into the U1 or U2 state are the inactivity timers. These inactivity timers are reset when a packet of any type (except Isochronous Timestamp Packets) is sent or received by either link partner and prevent the inactivity timers from expiring. However, following a period of logical idle, an inactivity timer may expire. Inactivity timer expiration initiates a handshake sequence between the link partners typically resulting in a transition to U1 or U2 low-power state.

The root ports and the downstream ports of all external hubs must implement U1 and U2 inactivity timers. In addition, the upstream port of peripheral devices and external hubs must implement a U2 inactivity timer to support silent transitions to the U2 state, discussed later. Figure 24-4 illustrates the various locations of the inactivity timers.

*Figure 24-4: Inactivity Timers*



# Chapter 24: SuperSpeed Power Management

---

## U1 Inactivity Timeouts

The transition from U0 to U1 can be triggered by the hub port's U1 inactivity timer expiring. Host software establishes the timeout interval by delivering a PORT\_U1\_TIMEOUT request to the target hub. Table 24-3 lists the Setup DATA for this hub request and Table 24-4 lists the U1 timeout values and associated timeout intervals defined by the specification.

Table 24-3: Set Feature - Port U1 Timeout Request — Setup Data

Request Type (1 Byte)	Request (1 Byte)	Value (2 Bytes)	Index (2 Bytes)	Length (2 Bytes)	Data
0000000B	SetFeature (03h)	Feature Selector 23 (17h) = PortU1Timeout	U1 Timeout Value	Zero	NA

Table 24-4: U1 Inactivity Timeout Value

Value	U1 Timeout
00h	Zero (default)
01h	1μs
02h	2μs
03h	3μs
.....	.....
7Fh	127μs
80h - FEh	Reserved
FFh	Infinite

The default value of zero disables the inactivity timers and rejects any requests to enter U1. Values from 01h - 7Fh (1-127d) correspond to U1 timeout values, which have a granularity of one microsecond. Software may set up a timer condition called *Infinite* (value FFh) that prevents the hub downstream port from triggering entry to U1, but accepts requests from the link partner for entering the U1 state.

# **USB 3.0 Technology**

---

## **U2 Inactivity Timeouts**

The transition from U0 to U2 can be triggered when the U2 inactivity timer expires. By default the U2 inactivity timer is disabled, thereby preventing entry to the U2 state and rejecting requests from the upstream port of the link partner to enter U2. Like the U1 inactivity timer, software establishes a timeout interval by delivering a PORT\_U2\_TIMEOUT request. Table 24-5 lists the Setup DATA for this hub request and Table 24-6 lists the U2 timeout values and associated timeout intervals defined by the specification.

*Table 24-5: Set Feature - Port U2 Timeout Request — Setup Data*

Request Type (1 Byte)	Request (1 Byte)	Value (2 Bytes)	Index (2 Bytes)	Length (2 Bytes)	Data
00000011B	SetFeature (03h)	Feature Selector 24 (18h) = PortU2Timeout	U2 Timeout Value	Zero	NA

*Table 24-6: U2 Inactivity Timeout Values*

Value	U2 Timeout
00h	Zero (default)
01h	256μs
02h	512μs
03h	768μs
.....	.....
FEh	65.024ms
FFh	Infinite

The U2 timeout interval has a granularity of 256μs, ranging to 65.024μs. The infinite setting prevents the inactivity timer from triggering entry into U2, but accepts requests from the link partner that may trigger entry into the U2 state.

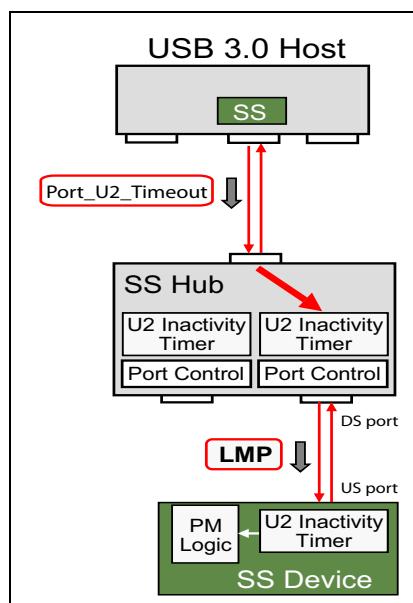
## Chapter 24: SuperSpeed Power Management

### **U0 → U1 → U2 with Silent Transition**

Software may choose to set up a sequence of transitions from U0 → U1 → U2 where the transition from U1 to U2 is silent (i.e., no bus activity is needed for the transition). To initiate the sequence software sets up the U1 inactivity timer using the PORT\_U1\_TIMEOUT request. Next software delivers the PORT\_U2\_TIMEOUT request that results in the following actions (Refer to Figure 24-5):

1. The PORT\_U2\_TIMEOUT request causes the U2 inactivity timeout value to be loaded into U2 inactivity timer.
2. Next, the hub places the U2 timeout value into a Link Management Packet (LMP) that is delivered to the attached device. See Figure 24-6 on page 574
3. The LMP initializes the device's U2 Inactivity timer, which sets up the same U2 timeout value at each end of the link.
4. During normal operation, the U1 inactivity timer expires and the downstream hub port initiates the handshake sequence with the device.
5. The handshake completes with the link partners transitioning to U1.
6. When the link enters U1 each link partner starts its U2 timer.
7. When the U2 Inactivity Timers both expire the link partners silently transition to U2. Note that the U2 timeouts will not be precisely synchronous.

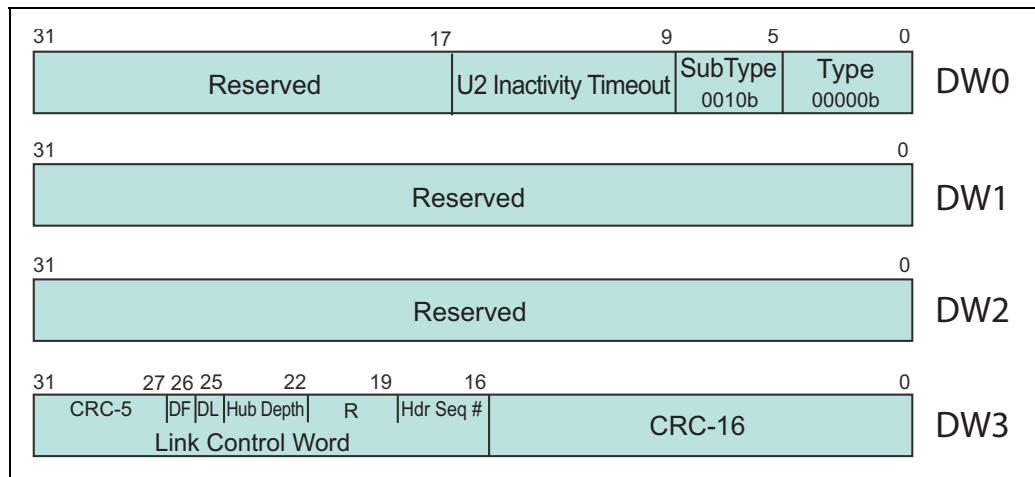
Figure 24-5: Silent Transitions — LMP



## USB 3.0 Technology

---

Figure 24-6: LMP — Inactivity Timeout



---

### Device-Specific Inactivity Timers/Algorithms

Devices may implement algorithms and timers that, when enabled by software, can trigger entry into the U1 and/or U2 states. Devices may be better suited to manage power transitions more aggressively than inactivity timers. For example, a given device may know the nature of the traffic associated with its function and trigger entry to U1 or U2 immediately following a burst transaction, knowing there will be no additional traffic for some time.

Software must specifically enable a device to trigger entry into the U1 or U2 states via a SetFeature U1\_Enable and U2\_Enable request. Table 24-7 shows the Setup Data associated with this request.

Table 24-7: SetFeature U1 Enable and U2 Enable Requests — Setup Data

Request Type (1 Byte)	Request (1 Byte)	Value (2 Bytes)	Index (2 Bytes)	Length (2 Bytes)	Data
0000000B	SetFeature (03h)	Feature Selector 48 (30h) = U1_Enable 49 (31h) = U2_Enable	Zero	Zero	NA

## **Chapter 24: SuperSpeed Power Management**

---

There are conditions that prevent software from enabling a device's ability to initiate transitions to U1 and/or U2. This restriction is related to periodic transfers.

---

### **Upstream and Downstream U1/U2 Triggers**

Downstream (DS) and Upstream (US) Ports may use inactivity timers to trigger possible entry to the U1 or U2 link state, but must be enabled by software to do so. Table 24-8 summarizes the conditions under which these triggers are allowed or rejected. The table does not cover every possible variation but describes the most common implementations.

*Table 24-8: Summary of U0/U1/U2 Entry Events*

<b>U1 Timer</b>	<b>U2 Timer</b>	<b>Device</b>	<b>Transition</b>	<b>Initiating Port</b>
00h	00h	Disabled	U0 only	Neither
01h-7Fh	00h	Disabled	U0 → U1	DS Port only
00h	01h - FEh	Disabled	U0 → U2	DS Port only
01h-7Fh	01h - FEh	Disabled	U0 → U1→U2	DS Port only
FFh	FFh	Disabled	U0 only	Neither
00h	00h	Enabled	U0 only	Neither
01h-7Fh	00h	Enabled	U0 → U1	Either Port
00h	01h - FEh	Enabled	U0 → U2	Either Port
01h-7Fh	01h - FEh	Enabled	U0 → U1→U2	DS Port only
FFh	FFh	Enabled	U0→ U1/U2	UP Port only

# **USB 3.0 Technology**

---

---

## **The Handshake Sequence**

---

### **Entering The U1 or U2 States**

Each link partner implements the Link Training Status State Machine (LTSSM) that manages transitions from U0 to U1/U2 states when an appropriate triggering mechanism is detected. There are a variety of events that can trigger a transition to a low-power state. Software may trigger the transition itself or may have enabled one or both of the link partners to triggering a power transition.

- Host software requests (not recommended) — Set\_Port\_Feature (Link State U1 or U2)
- Hub inactivity timer expiration — U1 or U2 inactivity timers
- Device inactivity timer expiration — U2
- Device-specific algorithms and/or inactivity timers

Regardless of the mechanism used to trigger entry into U1 or U2 the link partners must negotiate entry to the low-power state. The negotiation process can involve four Link-Layer packets:

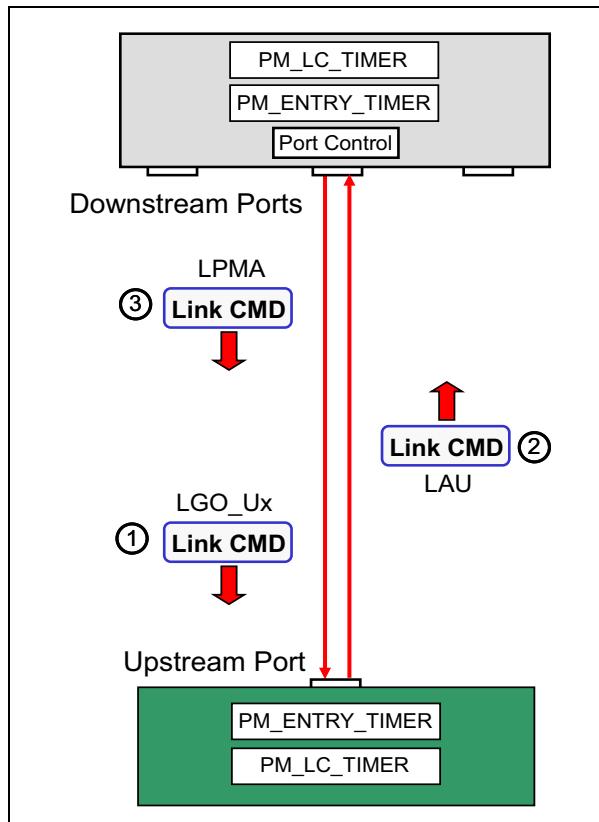
- LGO\_Ux — Used to request entry into Ux state (x=1 or 2)
- LAU — Link Accept entry into U state
- LXU — Link Accept Rejection
- LPMA — Link Power Management Acknowledgement

Figure 24-7 on page 577 illustrates a successful handshake sequence, with the downstream port initiating the entry into a low-power link state. Upstream ports may also trigger entry to a low-power state. The LTSSM implements two timers associated with this handshake:

- PM\_LC\_TIMER
- PM\_ENTRY\_TIMER

## Chapter 24: SuperSpeed Power Management

Figure 24-7: Link Partner Handshake - U0 to U1 or U2



### U1 / U2 Handshake and Timer Behavior

All trigger sources may be used when transitioning to U1 or U2. The following bullet items describe the operation and behavior associated with the timers.

Note that the LTSSM does not permit sending LGO\_Ux unless the following conditions are satisfied:

1. Successful delivery and receipt of all recent headers have been confirmed and associated credits have been delivered and received.
2. There are no pending packets to send.
3. A valid triggering event has occurred.

# **USB 3.0 Technology**

---

The timers have the following characteristics:

- PM\_LC\_TIMER — used by initiating port to ensure prompt entry into the low-power state. This timer starts when the last symbol of the LGO\_Ux packet is delivered and is cleared upon receipt of either the LAU or LXU packet. If neither packet is received within 3 $\mu$ s the port must transmit TS1 ordered sets resulting in a transition to the Recovery state.
- PM\_ENTRY\_TIMER — used by the port receiving the request to enter the low-power state and ensures that the link partners are in the same state if an LAU or LPMA are not received.
  - During normal operation:
    - The timer starts when the last symbol of the LAU is sent.
    - When the LAU is received by the initiating port, it sends an LPMA and enters the low-power state.
    - When the receiving port detects LPMA it also transitions to the low-power state, providing the PM\_ENTRY TIMER has not timed out (6 $\mu$ s).
  - When LAU reception fails:
    - The receiving port detects the TS1 Ordered Sets, and
    - The receiving port enters Recovery and the timer is cleared.
  - When LPMA fails:
    - The receiving port will not detect TS1 Ordered Sets.
    - The receiving port enters the low-power state, when the 6 $\mu$ s PM\_ENTRY TIMER expires.

---

## **Device Suspend — U3**

---

### **General**

Only host software can initiate entry into the Device Suspend state by performing a Port\_Link State\_U3 request. Suspend permits software to place devices into a deep power conservation state, while allowing recovery back to U0 within milliseconds of time. Devices must be prepared to enter suspend at any time (i.e., from the default, address or configured states). As with USB 2.0, SuperSpeed devices must limit power consumption to 2.5mA. Bus-powered hubs are allowed to draw 12.5mA, which includes the hub and the four devices attached to the hub's downstream ports. (Note: Bus-powered hubs are limited to four downstream ports.)

## **Chapter 24: SuperSpeed Power Management**

---

### **Entering SuperSpeed Suspend**

When software initiates the transition to U3, this triggers the LGO\_U3/LAU/ LPMA handshake between the targeted hub downstream port and the upstream port of the attached device. The target device may also choose to reject the request to enter U3 by returning a LXU handshake to the hub port.

Unlike the USB 2.0 Suspend and Resume implementation, SuperSpeed has no global suspend capability that allows software to suspend all devices within the topology with a single command. Instead, software places SuperSpeed devices into low-power states one device at a time. Additionally, software cannot place a hub into the suspended state unless all devices downstream of the hub are already suspended. Thus, a hub that has any active devices attached to its downstream ports must reject any attempted request to enter suspend.

#### **The U3 Handshake**

When software performs a Set\_Port\_Feature (LINK\_PORT\_STATE=3) the handshake sequence begins. The handshake sequence and the timers are the same as used in the U1/U2 transition.

- PM\_LC\_TIMER — used by the downstream port to ensure prompt entry into the U3 state. This timer starts when the last symbol of the LGO\_U3 packet is delivered and is cleared upon receipt of the LAU packet. The upstream port must not send LXU. If LAU is not received within  $3\mu s$  the downstream port must transmit TS1 ordered sets resulting in a transition to the Recovery state. Following a successful Recovery to U0, the hub port retires entry to U3 again. If three consecutive failures occur the LTSSM transitions to the SS.Inactive state.
- PM\_ENTRY\_TIMER — used by the upstream port to ensure that the link partners are in the same state if an LAU or LPMA are not received.
  - During normal operation:
    - The timer starts when the last symbol of the LAU is sent by the upstream port
    - When the LAU is received by the downstream port, it sends an LPMA and enters U3.
    - When the upstream port detects LPMA it also transitions to U3, providing the PM\_ENTRY TIMER has not timed out.
  - When LAU reception fails:
    - The timer starts when the last symbol of the LAU is sent by the upstream port.
    - The downstream port's PM\_LC\_TIMER expires resulting in TS1 Ordered Sets being sent by the downstream port.

## **USB 3.0 Technology**

---

- The upstream port detects TS1s, enters Recovery and clears the PM\_ENTRY TIMER.
- When LPMA fails:
  - The upstream port does not detect TS1 Ordered Sets, and
  - When the 6 $\mu$ s PM\_ENTRY TIMER expires the port enters U3.

### **Device Suspend State**

When suspended devices must preserve the following information:

- Device address
- Device configuration value
- Port status change information (downstream ports of hubs)
- Function suspend enable state (if supported)
- Function remote wake enable state (if supported)

---

### **U1/U2/U3 Exit to U0**

This section discusses the exit from the low-power link states back to U0. The process of exiting from U1/U2/U3 involves the same process but there are timing differences that will be discussed later in this section.

---

### **Triggering U1/U2/U3 Exit**

The mechanisms used to trigger the recovery back to U0 differ as suggested by the terminology used; U1/U2 Exit versus U3 Wakeup discussed below:

U1/U2 Exit — may be triggered by host software or by upstream or downstream port events. These events include:

- Transaction targeting the downstream hub port
- Device needing to send ERDY on its upstream port
- Host software issuing a Port\_Link\_State\_U0 request to the downstream hub port

## **Chapter 24: SuperSpeed Power Management**

---

A triggering event causes one of the link partners to initiate an LFPS handshake sequence that begins the recovery process.

U3 Wakeup — Exiting Suspend (U3) can be performed in two ways that depend partially on whether a device supports Remote Wakeup as follows:

- Host Initiated Wakeup — The host sends a Set\_Port\_Link\_U0 request that targets the downstream hub port to which the suspended device is attached. This triggers the recovery handshake sequence.
- Device Initiated Wakeup — A suspended device that supports Remote Wakeup and has been enabled by software, may signal resume to the hub by initiating the recovery handshake sequence.

---

### **The Exit Handshake Sequence and Timing**

The handshake sequence begins with a continuous burst of Low Frequency Periodic Signaling (LFPS) that is followed by TS1 ordered sets. The TS1s ordered sets initiate entry to the recovery state and when recovery completes, the link enters U0.

Figure 24-8 on page 582 depicts the handshake sequence and Table 24-9 on page 582 defines the specific timing values that must be met for the U1, U2 and U3 recovery.

#### **The HandShake**

Link Partner 1:

1. LFPS Burst is initiated for the minimum duration t10-t12.
2. A valid LFPS is received from Link Partner 2 (t11). If there is no LFPS returned by Link Partner 2 within 2ms, the handshake fails.
3. Link Partner 1 is ready to transmit TS1s (t12).

Link Partner 2:

1. LFPS Burst is initiated within the allowed t10-t11 timing.
2. The Burst duration must be met (t11-t13).
3. Link Partner 2 is ready to transmit TS1s (t13) and observes the 20ns maximum gap between the end of LFPS and start of TS1s.

The TS1s initiate entry to Recovery and when completed the Link Partner will have transitioned the link to L0.

## USB 3.0 Technology

---

Figure 24-8: U1/U2/U3 LFPS Exit Signaling

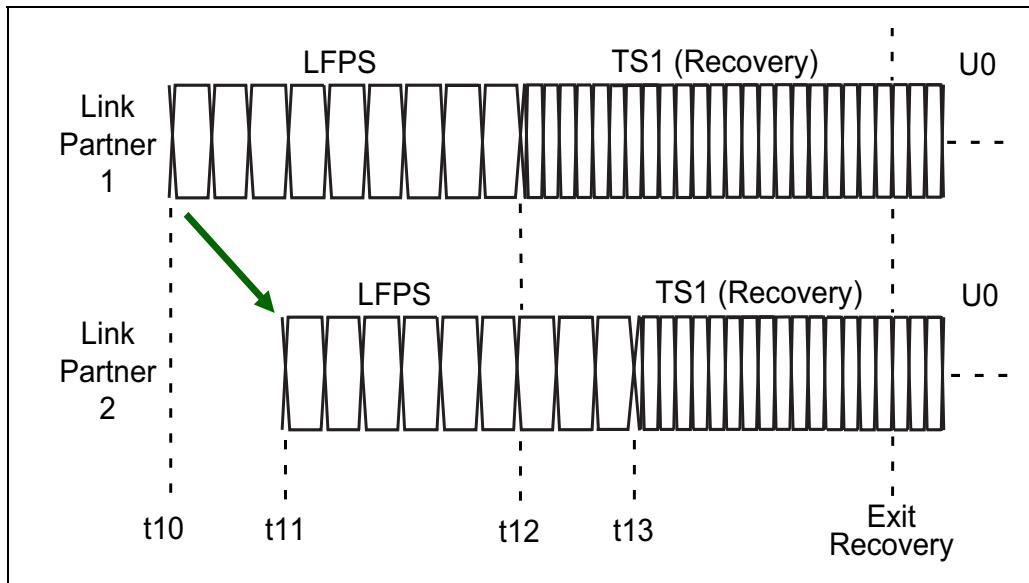


Table 24-9: U1/U2/U3 Exit Timing Parameters

Time	U1 Exit		U2 Exit		U3 Wakeup	
	Min	Max	Min	Max	Min	Max
t10-t11	0.3μs	0.9μs	0.3μs	2ms	0.3μs	10ms
t10-t13	0.9μs	2ms		2ms		20ms
t11-t12	0.3μs	0.9μs	0	2ms	0	10ms
t11-t13	0.6μs	0.9μs	80μs	2ms	80μs	10ms
t10-t12	0.6μs	2ms	80μs	2ms	80μs	10ms
No LFPS Response Timeout		2ms		2ms		10ms

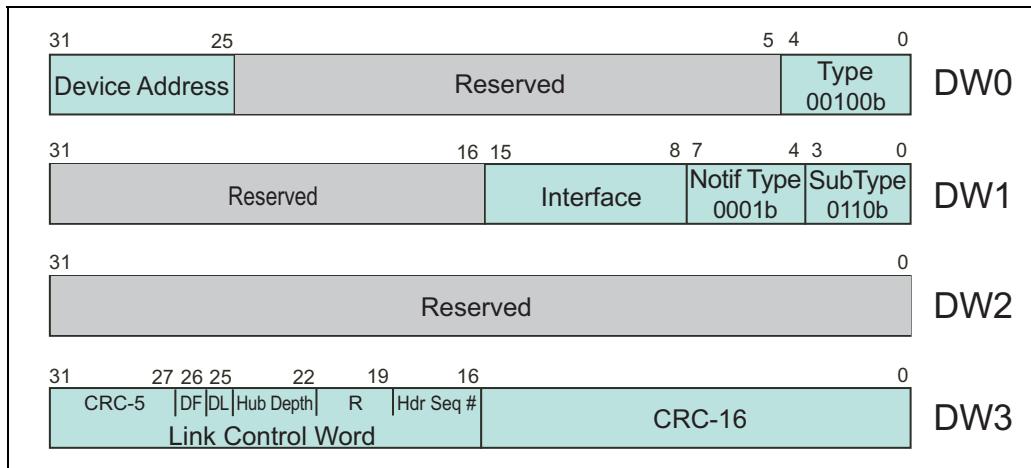
## Chapter 24: SuperSpeed Power Management

---

### Wake Notification

Once the wake sequence has completed and the link is in the U0 state, the device must send a Wake Notification Message to the host controller. This notification provides the Device Address and Interface Number (See Figure 24-9).

Figure 24-9: Function Wake Notification Packet



Note: Following reset, Remote Wakeup is disabled until software sends a Function\_Suspend request. (See Figure 24-10 on page 584) Many devices do not support Remote Wakeup, so they have no ability to initiate the wakeup sequence and only the hub downstream port can initiate wakeup.

---

### Function Suspend

---

#### General

SuperSpeed devices may support suspending a single function within a device. This is typically associated with a multifunction (i.e., composite) device and is called Function Suspend. This feature can also apply to a single function. When a function is placed in a suspended state, the power consumption associated with that function is reduced significantly.

# USB 3.0 Technology

---

The first step associated with a Function Suspend may be for software to execute a GetStatus Request targeting the device's interface to determine if the device supports Function Remote Wakeup (See Table 25-1). This request provides two bits:

- **Bit 0** = Indicates the Interface is Function Remote Wake Capable, when set.
- **Bit 1** = Indicates Function Remote Wakeup is enabled, when set.

Table 25-1: *GetStatus Request*

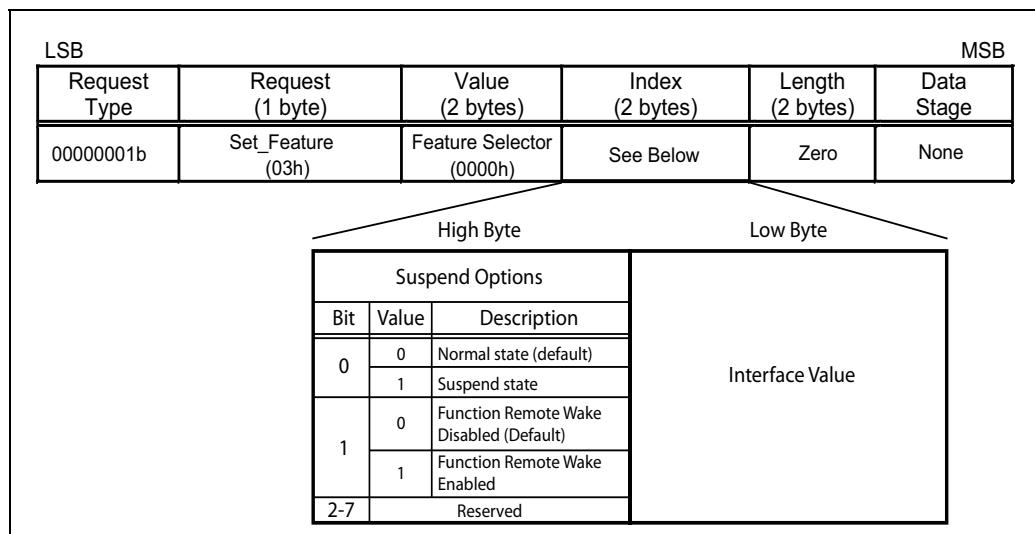
Request Type (1 Byte)	Request (1 Byte)	Value (2 Bytes)	Index (2 Bytes)	Length (2 Bytes)	Data
00100011B	GetStatus (00h)	(0000h)	Interface Number	0002h	None

---

## The Function Suspend Request

This feature is controlled by host software via the Function\_Suspend request that is delivered directly to the target device. Figure 24-10 illustrates the DATA fields within the Setup transaction associated with the Function\_Suspend request.

Figure 24-10: *Function Suspend Request — Setup Transaction Data*



## **Chapter 24: SuperSpeed Power Management**

---

The critical fields associated with the Function Suspend request are:

- Request Type – 01= Recipient is Interface
- Request – Set Feature request
- Value – 0000h defines the request as Function\_Suspend
- Index – Low Byte: First Interface of Function to be suspended  
High Byte: Bit 0 = 1 Suspend state is active  
Bit 1 = 1 Function Remote Wake is enabled

---

### **Latency Tolerance Message Reporting**

SuperSpeed peripheral devices may choose to implement Latency Tolerance Messages that make it possible for the platform to make dynamic adjustments to balance best performance & power savings.

These messages notify the host of whether the device needs to be serviced imminently or has no immediate requirements. By tracking service requirements a peripheral device can conserve power more aggressively. Peripheral devices can track their latency requirements and share this information with the host controller. For example, if software has accessed a device and it has no data to deliver, the device will return NRDY to the host. Devices use Latency Tolerance Messages to tell host how long they can tolerate lack of service. A variety of factors must be considered when reporting latency information, including:

- Periodicity of transfers
- Size of Internal buffering
- Worst-case Latency between device and host
- Endpoint with the worst latency tolerance

The device is unaware of the system delays that increase the overall latency. This includes:

- Number of hubs in path to host
- Link exit latencies
- Hub delays
- Host processing time

# USB 3.0 Technology

## Software Calculates Exit Latencies

U1 and U2 exit latencies are reported in the Device Capabilities for each port:

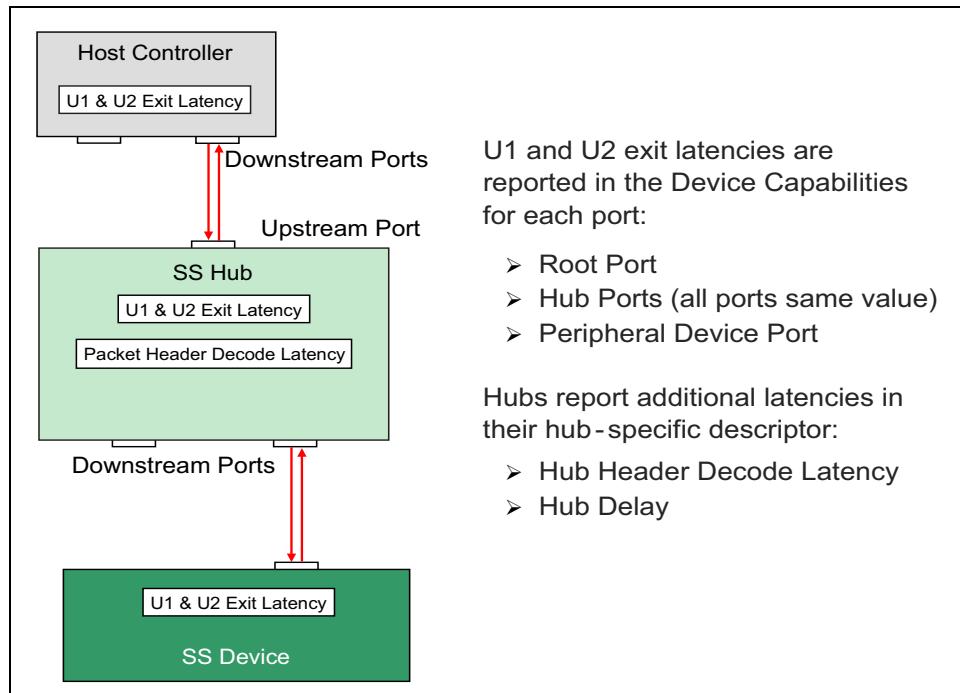
- Root Port
- Hub Ports (all ports same value)
- Peripheral Device Port

Hubs report additional latencies in their hub-specific descriptor:

- Hub Header Decode Latency
- Hub Delay

Figure 24-11 on page 586 shows how software calculates the exit latency information. The exit latency from the device to the downstream port of the root hub is termed the *path exit latency (PEL)*. Note that if the links in Figure 24-11 are in the U1 state the exit latency will be less than if the links are in the U2 state. Both sets of values must be calculated.

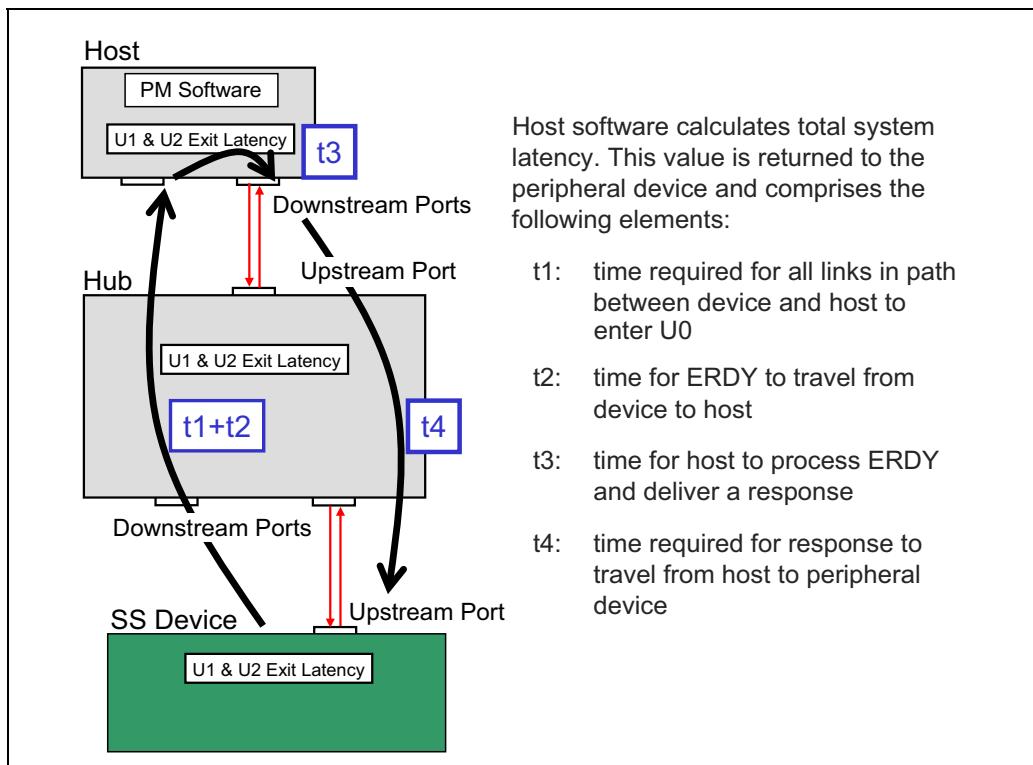
Figure 24-11: Path Exit Latency (PEL) and System Exit Latency (SEL) Reported Parameters



## Chapter 24: SuperSpeed Power Management

Figure 24-12 on page 587 illustrates the round trip latency, which is termed the *system exit latency (SEL)*. The SEL and PEL values are reported via software and must be subtracted from device's latency tolerance value. Alternatively, a device may assume SEL based on worst-case exit latency of 5 hubs between device and host.

Figure 24-12: System Exit Latency Calculation



Host software calculates total system latency. This value is returned to the peripheral device and comprises the following elements:

- t1: when device initiates transaction, the time required for all links in path to host to enter U0
- t2: time for ERDY to travel from device to host
- t3: time for host to process ERDY and deliver a response
- t4: time required for response to travel from host to peripheral device

## **USB 3.0 Technology**

---

The SEL request consists of a three-stage control transfer that includes a 6-byte data payload containing system and path exit latency data. Table 24-12 on page 587 lists the Request Data.

*Table 25-2: System Exit Latency Request Data*

Offset	Name	Description
0	U1SEL	System Exit Latency for U1 in $\mu$ sec (8 bits)
1	U1PEL	Path Exit Latency for U1 in $\mu$ sec (8 bits)
2	U2SEL	System Exit Latency for U2 in $\mu$ sec (16 bits)
3	U2PEL	Path Exit Latency for U2 in $\mu$ sec (16 bits)

---

## **Device Reports Current Latency**

The Device reports its exit latency thus notifying the host of the current latency.

Latency Tolerance Message Device Notification:

- Notifies host of change in service latency requirements. (e.g. a device at idle may have less restrictive latency requirements than when active.)
- LTM indicates worst case latency device can tolerate from host receipt of subsequent ERDY until it initiates a response.

Latency Tolerance Message:

- Included in the LTM DN is a Best Effort Latency Tolerance (BELT) value which indicates the maximum time the device can wait for service from the time a subsequent ERDY is received by the host.
- A device's latency tolerance must be adjusted to include the system exit latency value. Because more latency is being introduced, the SEL value must be subtracted from the device latency tolerance.
- The LTM Device Notification is optional; it is used by devices to inform the host of changes in service latency requirements

Miscellaneous Issues:

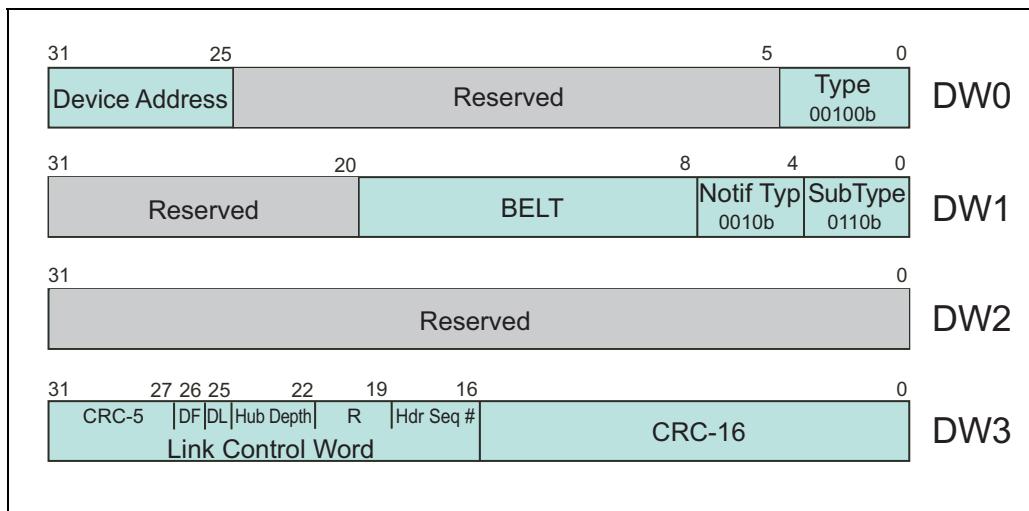
- Host defaults to BELT of 1ms for all devices
- If a device supports LTM, it is reported in descriptors.
- An LTM cannot be repeated more than 2 times during 1ms interval (tBEL-TRepeat)

## Chapter 24: SuperSpeed Power Management

Sending LTM by a device also requires:

- All endpoints in a device share the minimum service latency requested in the LTM BELT fields
- Minimum BELT request is 125µs
- LTMs are only sent when there is a BELT change to report.

Figure 24-13: Latency Tolerance Message Header



## **USB 3.0 Technology**

---

---

# **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 interoperability and to receive USB certification.

---

## **Physical Layer Electrical Signaling Scope**

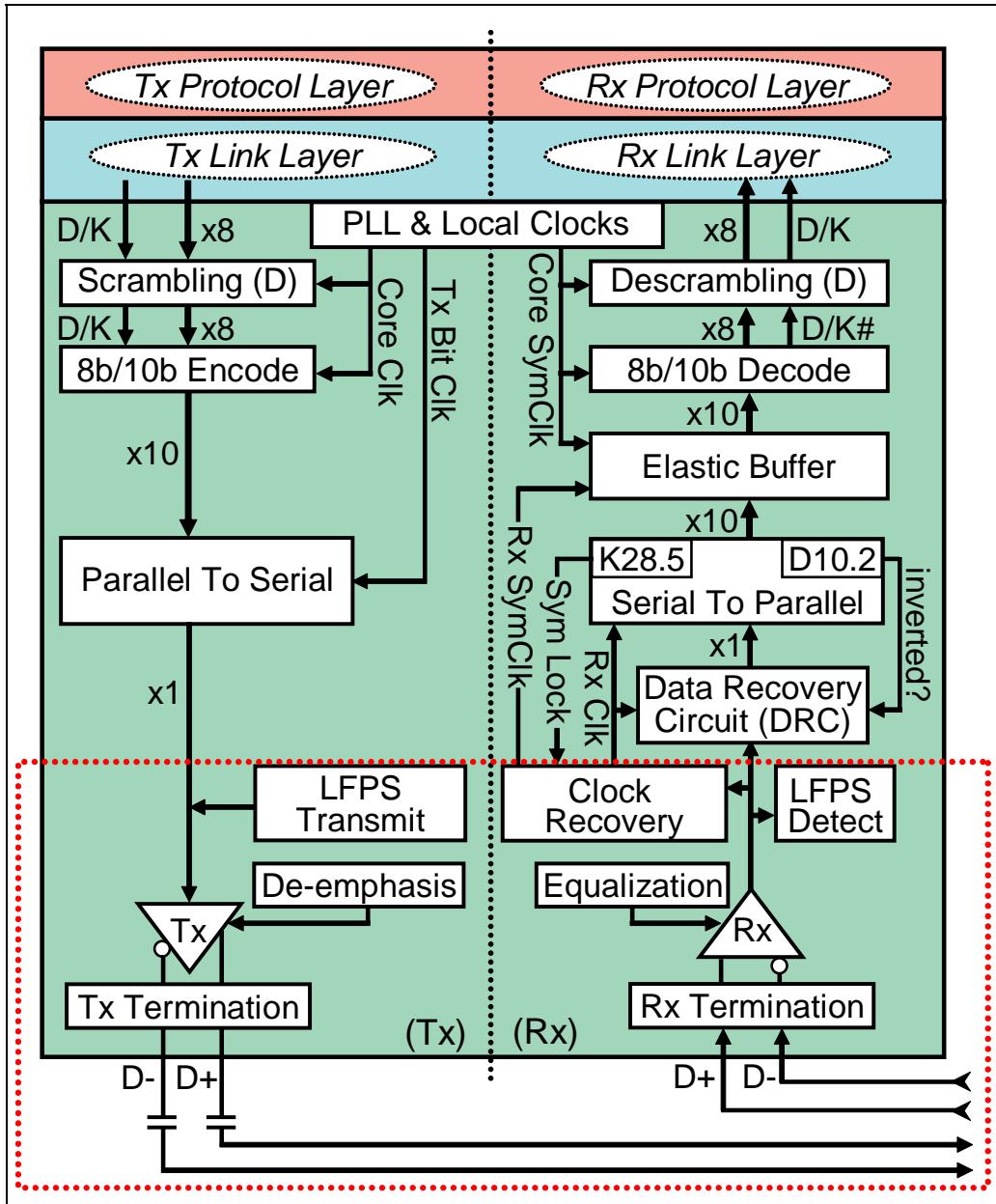
The SuperSpeed physical layer (PHY) can be divided into a logical and electrical signaling sections. This chapter covers the transmitter and receiver electrical specifications for the signaling elements in the highlighted region at the bottom of Figure 25-1 on page 592, including:

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

## USB 3.0 Technology

---

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



# **Chapter 25: SuperSpeed Signaling Requirements**

---

## **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 Super-Speed bus”.

---

## **Jitter Budgeting (Informative)**

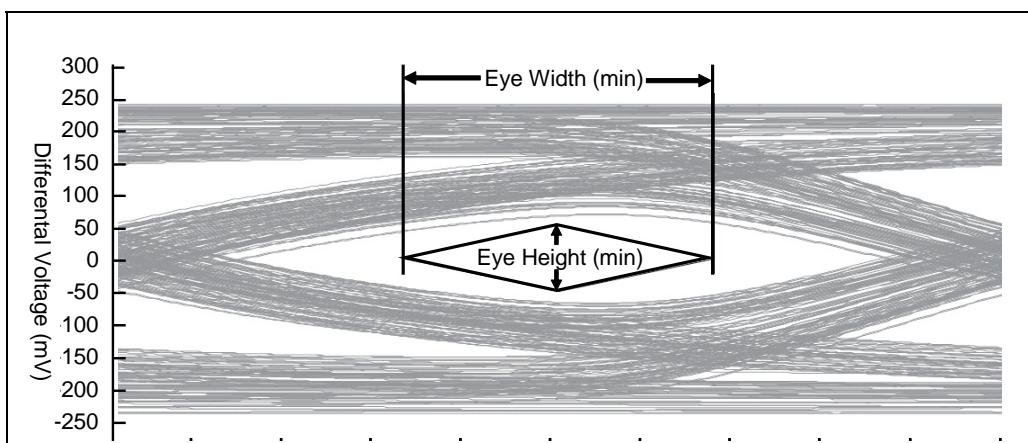
---

### **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 ( $D_j$ ) cause by transmission line effects
- Data-dependent random jitter ( $R_j$ ) 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*



## **USB 3.0 Technology**

---

At the receiver, jitter causes the recovered clock to move within some range relative to the ideal position and impacts sampling setup or hold times. The overall effect is to reduce the width of the usable portion of the eye to a mask region in the center. As indicated in the eye diagram shown in Figure 25-2 on page 593, the usable eye mask is bounded vertically by frequency-dependent attenuation and transmitter de-emphasis, and horizontally by jitter effects. Reliable operation is only assured within the mask area.

---

### **Jitter Budget Allocations**

The USB 3.0 SuperSpeed jitter budgeting allocates a portion of the total jitter ( $T_j$ ) budget, assuming a bit error rate (BER) of  $10^{-12}$ , to the three link elements:

- The transmitter
- The channel media (including packaging, cables, and connectors)
- The receiver

Table 25-1 on page 594 summarizes the portion of total jitter budget for the three elements (measured at silicon pads).

*Table 25-1: Jitter Budget At Silicon Pads (Informative)*

Jitter Contribution (ps)	$R_j^{1,2}$	$D_j^3$	$T_j^4$ at $10^{-12}$ Bit Error Rate (BER)
Tx <sup>6</sup>	2.42	41	75
Media <sup>5</sup>	2.13	45	75
Rx	2.42	57	91
Total	4.03	143	200

Notes:

1.  $R_j$ =sigma value assuming a Gaussian distribution
2.  $R_j$  total is calculated as Root Sum Square (RSS) of individual  $R_j$  components
3.  $D_j$  budget calculated using Dual Dirac method
4.  $T_j$  at  $10^{12}$  BER is calculated to be  $14.068 \times R_j + D_j$
5. Media budget assumes ISI cancellation using Rx equalization function
6. Tx measured after JTF (Jitter Transfer Function) has been applied

## Chapter 25: SuperSpeed Signaling Requirements

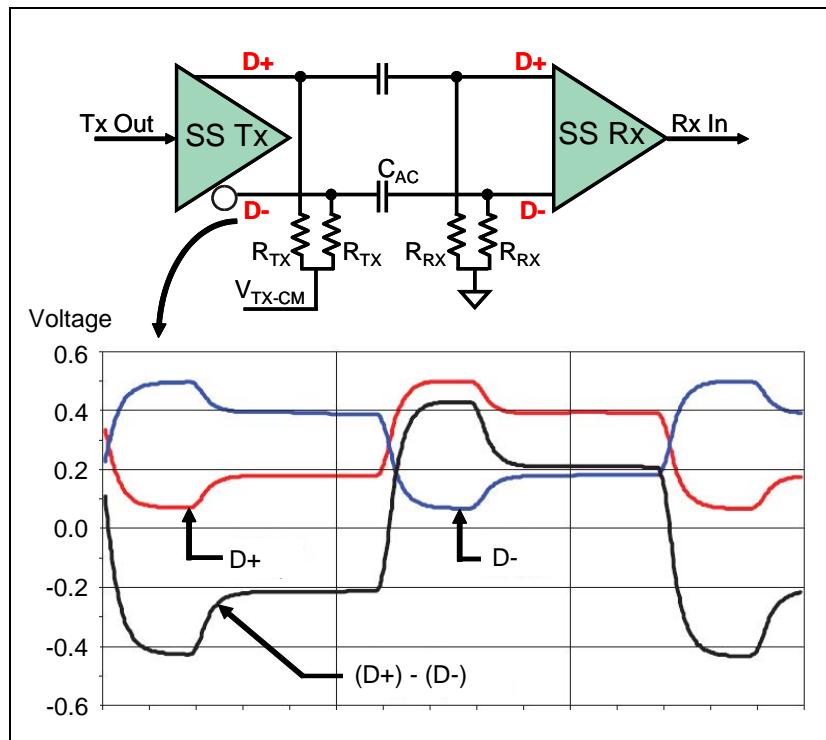
### SuperSpeed Voltage Levels, Some Definitions

The illustration in Figure 25-3 on page 595 depicts common views of Super-Speed differential and single-ended voltage levels:

- The differential voltage,  $V_{\text{DIFF}} = (D+) - (D-)$
- The total differential voltage swing,  $V_{\text{DIFF-PP}} = 2 * V_{\text{DIFF}}$
- The Common Mode Voltage,  $V_{\text{CM}}$ , is the average voltage on the differential pair with respect to ground.  $V_{\text{CM}} = (D+) + (D-)/2$ .

In the example shown in Figure 25-3,  $V_{\text{DIFF-PP}} = 800\text{mV}$ ,  $V_{\text{DIFF}} = 400\text{mVPP}$ , and  $V_{\text{CM}} = 300\text{mV}$ .

Figure 25-3: Differential And Single-Ended Voltage Levels



# USB 3.0 Technology

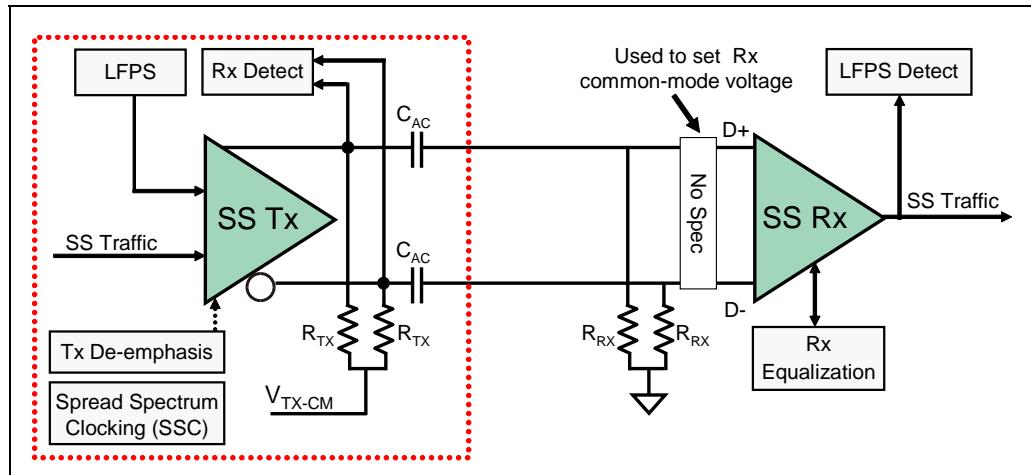
---

## SuperSpeed Transmitter Requirements

Following processing by the transmitter PHY logical section, serial data is driven onto the SuperSpeed link using the differential transmitter. Figure 25-4 on page 596 depicts major elements of SuperSpeed transmitter hardware:

- The differential driver with de-emphasis
- Transmitter termination and AC-coupling capacitors
- Far end receiver termination detection (Rx Detect) logic
- Spread spectrum clocking (SSC)

Figure 25-4: SuperSpeed Transmitter Elements



---

## Tx Electrical Specifications

SuperSpeed transmitter has both normative (required) and informative (optional) electrical specifications.

### Normative Tx Electrical Parameters

Table 25-2 on page 597 summarizes normative requirements for the differential transmitter. These parameters define timing and voltage parameters as measured at the transmitter outputs.

## **Chapter 25: SuperSpeed Signaling Requirements**

---

*Table 25-2: Transmitter Electrical Parameters (Normative)*

Symbol	Parameter	5GT/s	Units	Comments
UI	Unit Interval	199.94 (min) 200.06 (max)	ps	Corresponds to +/- 300 ppm. Does not include SSC variations
V <sub>TX-DIFF-PP</sub>	Differential Tx p-p voltage	0.8 (min) 1.2 (max)	V	1 V p-p nominal
V <sub>TX-DIFF-PP-LOW</sub>	Low Power Differential Tx p-p voltage	0.4 (min) 1.2 (max)	V	De-emphasis not required in Low Power Mode
V <sub>TX-DE-RATIO</sub>	Tx De-emphasis level	3.0 (min) 4.0 (max)	dB	3.5 dB nominal
R <sub>TX-DIFF-DC</sub>	DC Differential Impedance	72 (min) 120 (max)	$\Omega$	
V <sub>TX-RCV-DETECT</sub>	Tx voltage change allowed during Receiver Detection	0.6 (max)	V	
C <sub>AC-COUPLING</sub>	AC Coupling Capacitor value	75 (min) 200 (max)	nf	Tx provides coupling. May be internal or external
t <sub>CDR-SLEW-MAX</sub>	Maximum slew rate	10	ms/s	

# USB 3.0 Technology

---

## Informative Tx Electrical Parameters

The informative transmitter electrical parameters are summarized in Table 25-3 on page 598.

Table 25-3: Transmitter Electrical Parameters (Informative)

Symbol	Parameter	5GT/s	Units	Comments
$t_{MIN-PULSE\_Dj}$	Deterministic minimum pulse	0.96	UI	Tx pulse width variation (deterministic)
$t_{MIN-PULSE\_Tj}$	Tx minimum pulse	0.90	UI	Min Tx pulse width at $10^{-12}$ BER; includes Dj & Rj
$t_{TX-EYE}$	Transmitter Eye	0.625 (min)	UI	Includes jitter sources
$t_{TX-DJ-DD}$	Tx Deterministic Jitter	0.205 (min)	UI	Deterministic (assuming Dual Dirac distribution)
$C_{TX-PARASITIC}$	Tx input capacitance for return loss	1.25 (max)	pF	Parasitic capacitance to ground
$R_{TX-DC}$	Tx DC common mode impedance	18 (min) 30 (max)	$\Omega$	Measured with respect to AC ground over 0-500mV range
$I_{TX-SHORT}$	Tx short circuit current limit	60 (max)	mA	Total current Tx supplies when shorted to ground
$V_{TX-DC-CM}$	Tx DC common mode voltage	0 (min) 2.2 (max)	V	Instantaneous DC voltages allowed at connector side of AC capacitors
$V_{TX-AC-PP\_ACTIVE}$	Tx AC common mode voltage active	100	mVp-p	Maximum mismatch from $T_{xp}/T_{xn}$ for both time and amplitude

## Chapter 25: SuperSpeed Signaling Requirements

Table 25-3: Transmitter Electrical (Continued)Parameters (Informative)

Symbol	Parameter	5GT/s	Units	Comments
V <sub>TX-CM-DC-ACTIVE-IDLE-DELTA</sub>	Absolute DC common mode voltage between U <sub>1</sub> and U <sub>0</sub>	200 (max)	mV	
V <sub>TX-IDLE-DIFF-AC-pp</sub>	Electrical Idle differential peak-peak output voltage	0 (min) 10 (max)	mV	
V <sub>TX-IDLE-DIFF-DC</sub>	DC Electrical Idle differential output voltage	0 (min) 10 (max)	mV	Low pass filter is applied to remove AC component
Notes				
1. Values in this table are informative. 2. Transmitter may be fully compliant and not meet all values in this table or transmitter may meet all values in this table and not be in full compliance with the normative part of the specifications.				

### Transmitter Low Power Option

The USB 3.0 specification provides an option for low power transmitter operation in cases where the system may be sensitive to power consumption or electrical interference and the channel characteristics are deterministic. Examples include:

- Link partners connected with printed circuit board (PCB) traces. In this case, there are no connectors or cables present.
- Internal USB 3.0 cable connections between motherboard and USB 3.0 connector at the front or rear panel of a computer chassis where the cable length is minimal.

In such cases, the transmitter can operate at lower output voltages and without employing de-emphasis. The electrical requirements for transmitter operation in low power mode can be seen in Table 25-2 on page 597. Note that the requirements at the receiver do not change; there is only one set of requirements for signals at the receiver differential input.

# USB 3.0 Technology

## Transmitter De-emphasis

Transmitters operating at the standard power levels (not the low power option just described) are required to employ signal de-emphasis as part of the overall link equalization scheme.

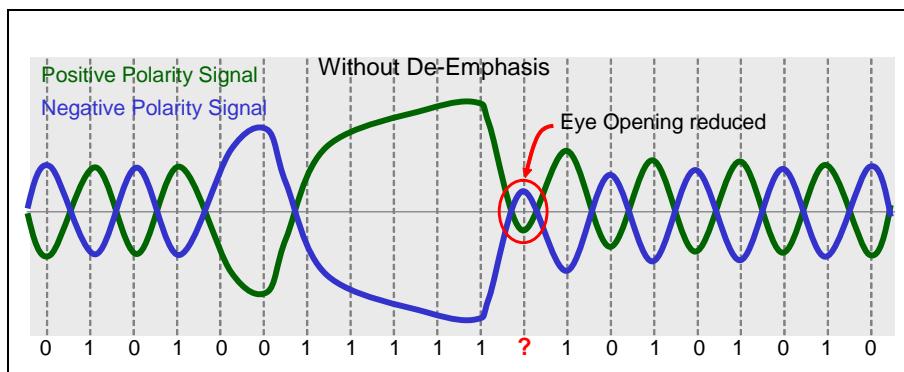
## Background

At high serial data bit rates, frequency-dependent losses become problematic for the receiver. In the case of the 5 Gb/s SuperSpeed link, a long channel (pc board traces, connectors, cable, etc.) may act as a very aggressive low pass filter:

- Data patterns with multiple bits of the same polarity appear as low frequency traffic and are readily passed to the receiver without much attenuation (we would say that the bandwidth of the channel is higher than that of that of the data).
  - Data with rapidly alternating “0” and “1” patterns appear as very high frequency traffic and is attenuated by the channel because there is insufficient time for the voltage to reach its maximum amplitude before the next bit transition (In this case, the bandwidth of the data is higher than that of the channel).

The problem for the receiver is that real-world data contains a mix of bit patterns and successive bits of the same polarity (low frequency) may impact the ability of the receiver to properly interpret the first bit that follows each transition (zero to one or one to zero). This is referred to as inter-symbol interference (ISI) and can be seen in the illustration in Figure 25-5 on page 600.

Figure 25-5: Inter Symbol Interference (ISI) Example



## **Chapter 25: SuperSpeed Signaling Requirements**

---

From left to right in the example shown in Figure 25-5 on page 600, note the following events:

- The transmitter sent four bits of alternating 0's and 1's. Note that these bits are toggling at the maximum rate and appear as a short duration 2.5 GHz waveform. This is the highest frequency pattern possible. Note that the bits never reach the same voltage levels as do bits in the lower frequency patterns that follow.
- Next, the transmitter sent two successive bits of the opposite polarity (0's); note here that the voltage level continued to build in the second bit time.
- Next, the transmitter drove five successive bits of the opposite polarity (1's); again, the voltage level continued to build with each successive bit. Because the 8b/10b encoding scheme doesn't permit symbols with more than 5 successive bits of the same polarity, this is the lowest frequency pattern possible on the link. Even this low frequency pattern tends to create a significant ISI problem at the next bit transition if the receiver is at the end of a long channel.
- As shown in the highlighted portion of Figure 25-5 on page 600, when the transmitter attempted to drive the first bit of the opposite polarity after the long run of five sequential bits, the voltage level was required to transition from the highest possible level to the opposite state in one bit time. Based on the illustration, the transmitter may not have been successful delivering a sufficient differential eye opening for the receiver.

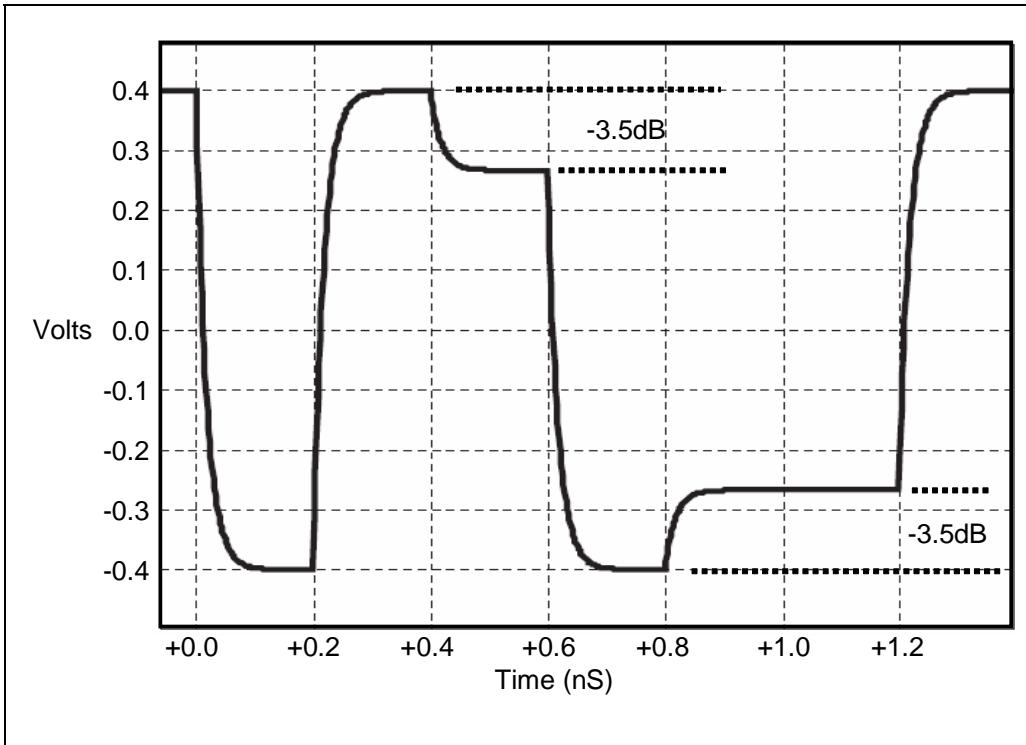
### **Transmitter Signal De-emphasis Helps**

To address this problem, the transmitter monitors the outbound bit stream and only drives full voltage at a transition; it de-emphasizes (lowers) the output voltage by -3.5 dB for successive bits of the same polarity--until the next transition.

De-emphasis has the effect of boosting the high frequency components (transitions) in the bit stream, equalizing the signal levels of high frequency and low frequency bit patterns at the receiver inputs, and ultimately reducing ISI. The de-emphasized signal level is nominally -3.5dB and is shown in Figure 25-6 on page 602. Note that in the illustration, the voltage levels are peak-to-peak.

## USB 3.0 Technology

Figure 25-6: Transmitter De-emphasis Example (Peak-to-Peak Voltage)

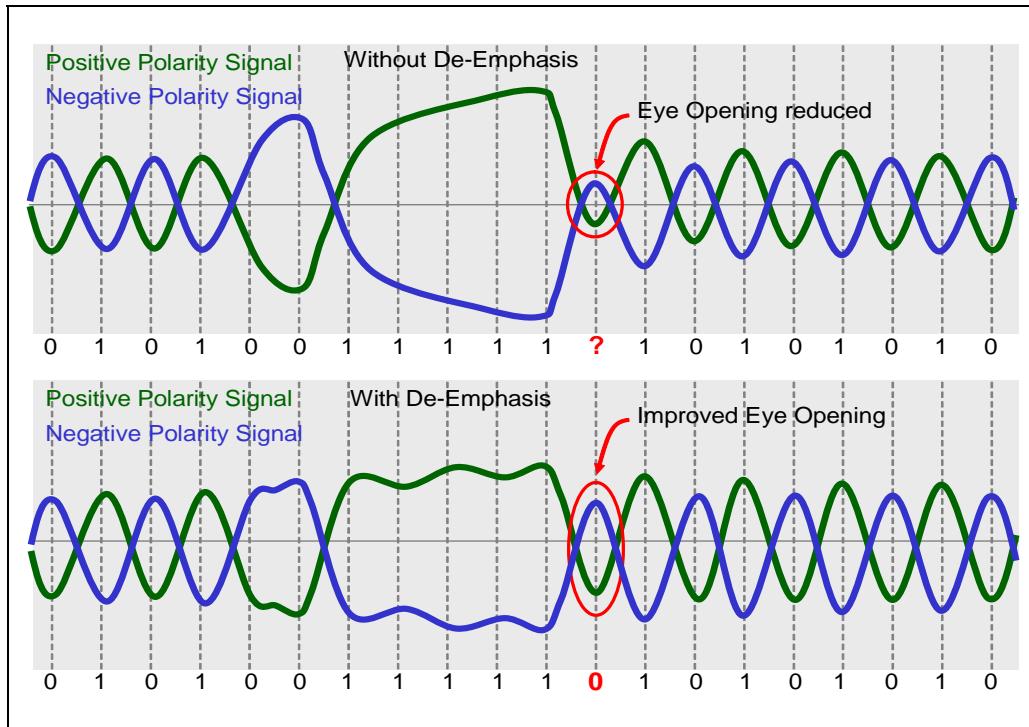


### De-emphasis: A Comparison

In Figure 25-7 on page 603, a comparison of ISI effects with and without de-emphasis is depicted. In the lower illustration, the transmitter de-emphasized the last four bits in the long run before the transition and the voltage level never reached the level it would have otherwise. At the highlighted transition, a significantly improved signal eye opening is seen at the receiver inputs.

## Chapter 25: SuperSpeed Signaling Requirements

Figure 25-7: Comparison Of ISI With, Without De-emphasis



### Transmitter Spread Spectrum Clocking (SSC)

In addition to the basic clock requirement of 5 Gb/s +/- 300 ppm, the USB 3.0 specification requires transmitters to support spread spectrum clocking (SSC) as a method for reducing EMI. SSC involves modulating the clock frequency of the transmitter at a rate that spreads the frequency spectrum of radiated energy and reduces large spikes at one frequency. The modulation rate, between 30KHz and 33KHz, is slow enough to avoid rapid changes in the receiver elastic buffers or in the management of clock compensation skip (SKP) ordered sets.

As illustrated in the example shown in Figure 25-8 on page 604, SSC modulation is only in the downward direction from the 5 Gb/s rate (0 ppm to -5000 ppm). Restricting the modulation to lower frequencies also helps prevent the tendency to overrun the receiver elastic buffer when SSC is at the upper frequency limit.

## USB 3.0 Technology

---

Figure 25-8: Triangular SSC Modulation Example

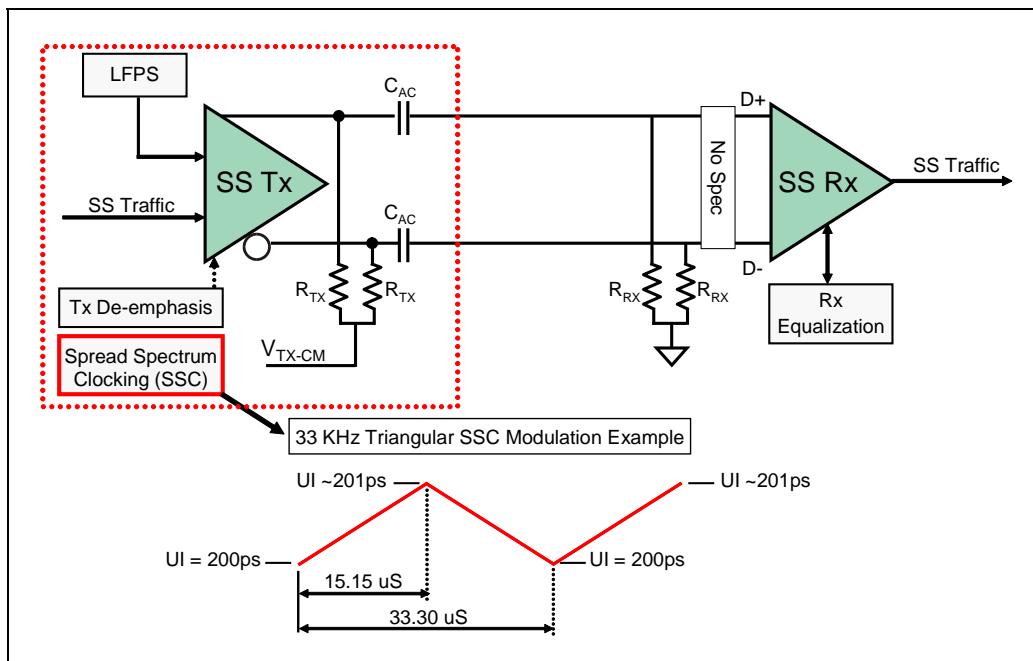


Table 25-4: Spread Spectrum Clocking (SSC) Modulation

Parameter	Description	Limits		
		Min	Max	Notes
t <sub>SSC-MOD-RATE</sub>	Modulation Rate	30kHz	33kHz	
t <sub>SSC-FREQ-DEVIATION</sub>	Modulation Rate	+0/ -4000 ppm	+0/ -5000 ppm	1, 2
Notes				
1. Data rate modulated from 0 ppm to -5000 ppm of nominal bit rate (5GT/s, etc.) 2. SSC Frequency Deviation in ppm is measured using 2MHz or lower frequency				

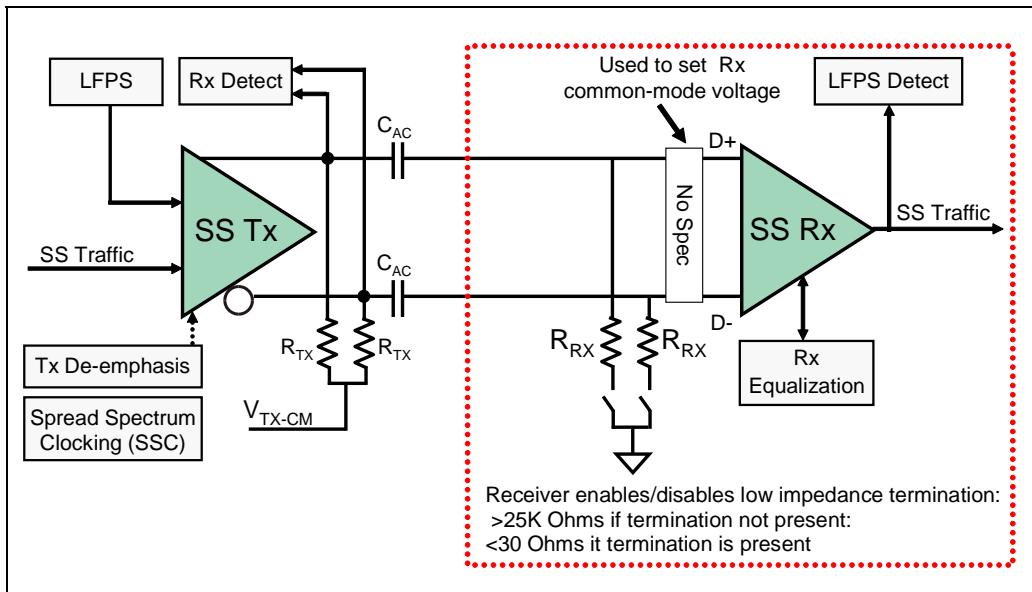
## Chapter 25: SuperSpeed Signaling Requirements

### SuperSpeed Receiver Requirements

The SuperSpeed differential receiver must compensate for negative channel effects on the incoming transmitter bits stream and recover usable clock and data information. In addition, the receiver must be capable of responding to Warm Reset and other low frequency periodic signaling (LFPS) events initiated by its link partner. Key receiver physical layer electrical elements are shown in Figure 25-9 on page 605, and include:

- Switched low impedance receiver termination
- Receiver equalization
- LFPS detect logic

Figure 25-9: SuperSpeed Receiver PHY Electrical Elements



### Rx Electrical Specifications

SuperSpeed receivers have both normative (required) and informative (optional) electrical specifications.

# **USB 3.0 Technology**

---

## **Normative Rx Parameters**

Table 25-5 on page 606 summarizes normative requirements for the differential receiver.

*Table 25-5: Receiver Electrical Parameters (Normative)*

<b>Symbol</b>	<b>Parameter</b>	<b>5GT/s</b>	<b>Units</b>	<b>Comments</b>
UI	Unit Interval	199.94 (min) 200.06 (max)	ps	Corresponds to +/- 300 ppm. Does not include SSC variations
R <sub>RX-DC</sub>	Receiver DC common mode impedance	18 (min) 30 (max)	Ω	Measured with respect to ground over voltage range of 500mV maximum
R <sub>RX-DIFF-DC</sub>	DC Differential Impedance	72 (min) 120 (max)	Ω	Measured with respect to ground over max range of differential signal (1200mV)
Z <sub>RX-HIGH-IMP-DC-POS</sub>	DC input common mode input impedance for voltage >0 during reset or power down <sup>2</sup>	25 K (min)	Ω	RX terminations not powered. Measured with respect to ground over range 0-500mV
V <sub>RX-LFPS-DET-DIFFp-p</sub>	Low Frequency Differential Signaling (LFPS) detect threshold	100 (min) 300 (max)	mV	Below minimum is noise. Receiver must wake up if above maximum
Notes:				
1. Normative specifications listed above measured at connector				
2. DC input common mode input impedance is specified for V>0. If V= 0, impedance not guaranteed (may be as low as 0 Ω)				

## **Chapter 25: SuperSpeed Signaling Requirements**

---

### **Informative Rx Parameters**

The informative receiver electrical parameters are summarized in Table 25-6 on page 607.

*Table 25-6: Receiver Electrical Parameters (Informative)*

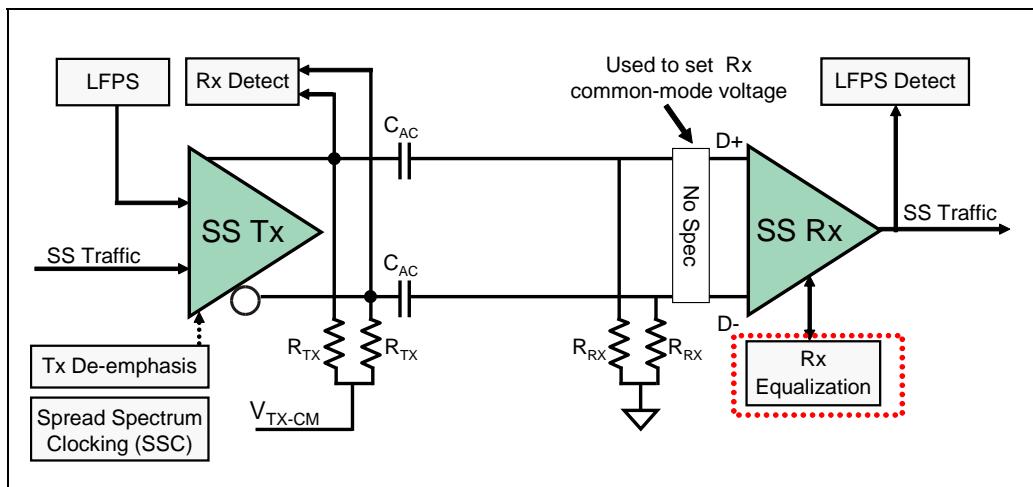
<b>Symbol</b>	<b>Parameter</b>	<b>5GT/s</b>	<b>Units</b>	<b>Comments</b>
$V_{RX-DIFF-PP-POST-EQ}$	Differential Rx peak-to-peak voltage	30 (min)	mV	Measured after Rx Equalization applied
$t_{RX-TJ}$	Max Rx inherent timing error	0.45 (max)	UI	Measured after Rx Equalization applied
$t_{RX-TJ-DD}$	Max Rx inherent deterministic timing error	0.285 (max)	UI	Measured Rx inherent deterministic timing error
$C_{RX-PARASITIC}$	Rx input capacitance for return loss	1.1 (max)	pF	
$V_{RX-CM-AC-P}$	Rx AC common mode voltage	150 (max)	mV Peak	Measured at Rx pins into $50\ \Omega$ terminations to ground. AC range up to 5GHz
$V_{RX-CM-DC-ACTIVE-IDLE-DELTA-P}$	Rx AC common mode voltage during U1-U0 power management state transitions	200 (max)	mV Peak	Measured at Rx pins into $50\ \Omega$ terminations to ground. AC range up to 5GHz

# USB 3.0 Technology

## Rx Equalizer Training

The responsibility for signal compensation on SuperSpeed links falls on both transmitters and receivers. The compensation for frequency-dependent signal loss is generally referred to as equalization. As described earlier in this chapter, and depicted in Figure 25-10 on page 608, transmitters use voltage de-emphasis on outbound data bits to combat inter-symbol interference (ISI) effects at the receiver. At the receiver, equalization logic compensates for the sum of detrimental effects of its channel on the 5 Gb/s signal delivered from transmitter to receiver inputs.

Figure 25-10: SuperSpeed Receiver Equalization

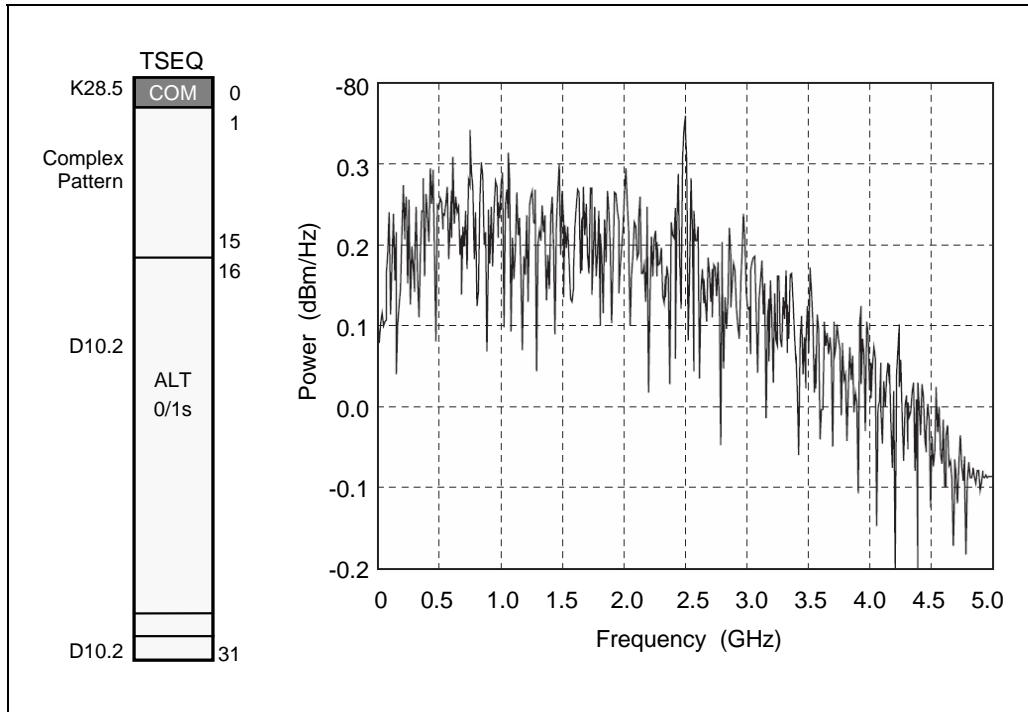


Initializing receiver (Rx) equalization logic is adaptive, and one of the most critical steps in SuperSpeed link training. The transmitter sends 65,536 training sequence equalization (TSEQ) ordered sets and the receiver applies the appropriate compensation based on signal voltage and timing detected at its inputs. The equalization test patterns are specially selected for their varied frequency content; it is assumed that once equalizers have been trained, any pattern of bits received during normal operations will be recovered correctly by the receiver.

The format of the training sequence (TSEQ) ordered set and the broad frequency spectrum contained in the 32 symbol pattern are shown in Figure 25-11 on page 609. Note that symbols 16-31 in the TSEQ ordered set is D10.2, which consists of the highest frequency pattern of alternating 0's and 1's. The effects of this can be seen as a large spike in the frequency spectrum at 2.5 GHz.

## Chapter 25: SuperSpeed Signaling Requirements

Figure 25-11: TSEQ Frequency Spectrum



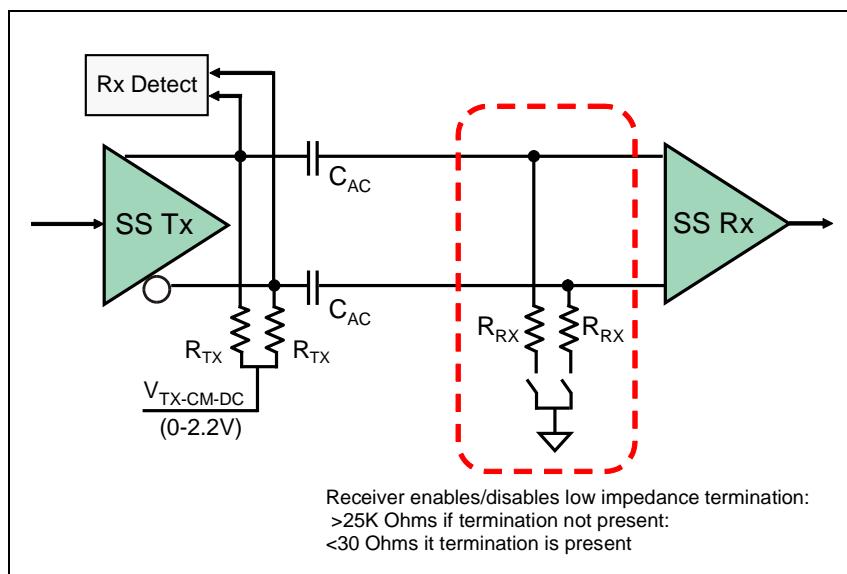
### SuperSpeed Receiver Termination

As indicated in Figure 25-12 on page 610, SuperSpeed receivers are capable of enabling and disabling their low-impedance termination resistors. For example, if the link state is U0, receiver termination is always enabled to provide the required differential signal termination needed when 5 Gb/s SuperSpeed traffic is on the bus. On the other hand, if  $V_{BUS}$  is removed by the downstream facing port then the bus is in the disabled state, and link partners are both required to disable the low-impedance receiver termination (even if they are self-powered).

## USB 3.0 Technology

---

Figure 25-12: Receiver Enables & Disables Low-Impedance Termination



### Receiver Common Mode Input Impedance Range

From the perspective of a transmitter “looking into” the attached SuperSpeed receiver, either a low or high far-end DC common mode impedance is seen.

- $R_{RX-DC}$  is receiver DC common mode input impedance when termination is enabled ( $18-30 \Omega$ s)
- $Z_{RX-HIGH\_IMP\_DC\_POS}$  is receiver DC common mode input impedance when termination not enabled ( $25 K\Omega$ s, min.)

### The Detection Method

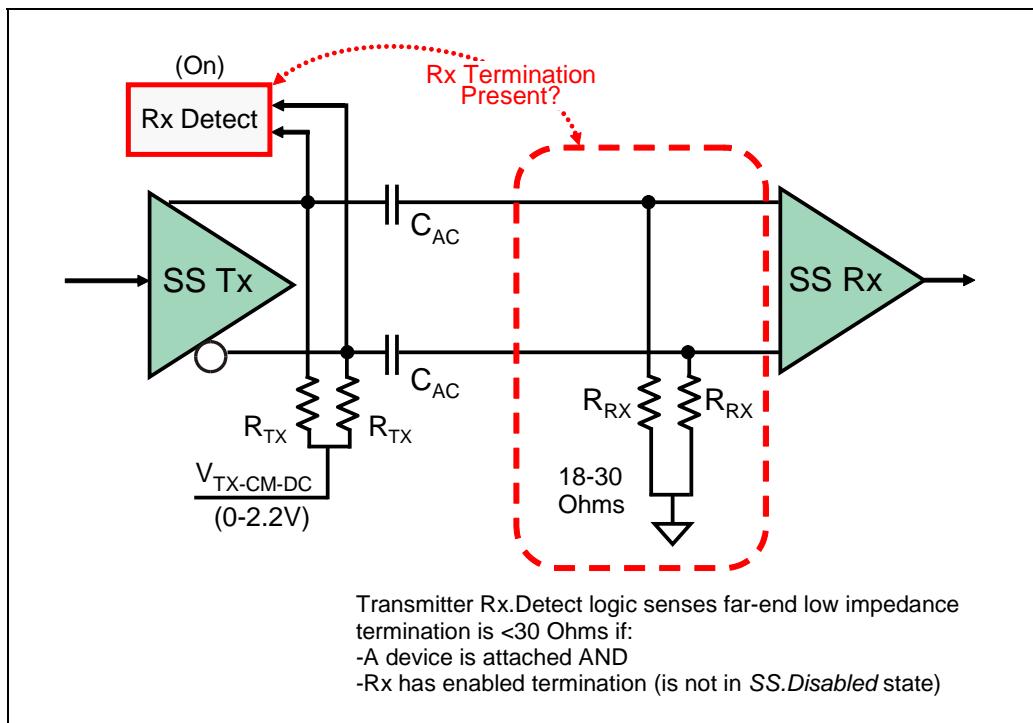
The capability of receivers to enable and disable low-impedance termination is used during link training to provide a simple way for device transmitters to determine whether a SuperSpeed-capable link partner is connected. The detection scheme, as depicted in Figure 25-13 on page 611, requires the SuperSpeed Transmitter to enable its integrated Rx Detect logic and follow the following general sequence:

- Transmitter establishes a stable common mode voltage ( $V_{TX-CM-DC}$ )
- Transmitter steps common mode voltage on D+/D- in a positive direction

## Chapter 25: SuperSpeed Signaling Requirements

- Transmitter Rx Detect logic measures the time required to raise Tx D+/D- outputs to the new voltage level. Rise time is proportional to load, a function of transmitter impedance, series capacitor ( $C_{AC}$ ), the interconnect capacitance, and the receiver termination (if enabled)

Figure 25-13: SuperSpeed Transmitter Detects Receiver Termination



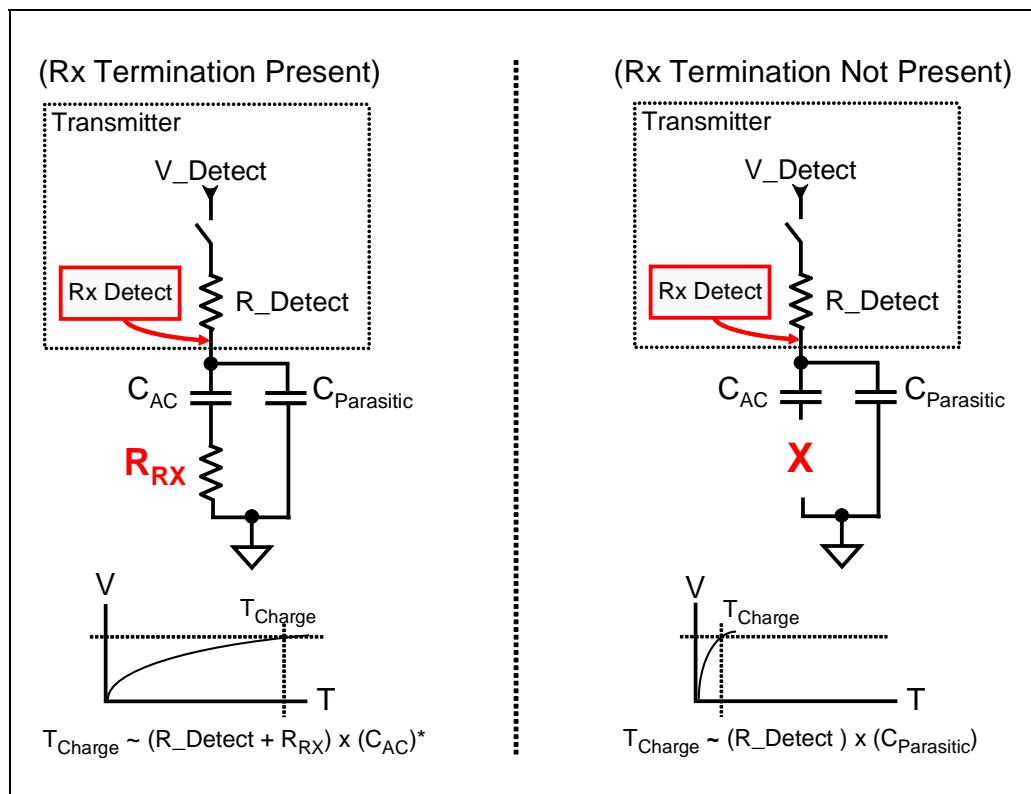
The method for detecting receiver termination is based on the behavior of a simple resistor-capacitor circuit attached to a dc power supply. The time required to charge the capacitor when voltage is applied is proportional to the size of the resistors multiplied by the size of the capacitors ( $T = R \times C$ ). Figure 25-14 on page 612 depicts equivalent circuits for the two Rx.Detect.Active cases.

- On the left is the case where the transmitter performs the common mode voltage shift, measures the rise time (relatively long), and determines that SuperSpeed receiver termination is present in the link partner. Note that the resistance consists of transmitter impedance ( $R_{Detect}$ ) and  $R_{RX}$ ; capacitance includes  $C_{AC}$  and a small amount of parasitic capacitance (which is ignored in the calculation shown)

## USB 3.0 Technology

- On the right of Figure 25-14, no receiver termination is detected (rise time is short). Here, resistance consists only of transmitter impedance ( $R_{\text{Detect}}$ ); capacitance is small, and only parasitic.

Figure 25-14: Rx Detect Equivalent Circuits



### Low Frequency Periodic Signaling Requirements

Low frequency periodic signaling (LFPS) is used for primitive link communications between link partners when the 5 Gb/s SuperSpeed link has not yet been trained or is in a state where its reliability is in question. As indicated in Figure 25-15 on page 613, the transmitter and receiver employ separate logic for sending and receiving LFPS. Once the link is trained and functioning properly in the operational states U0-U3, LFPS is primarily used to:

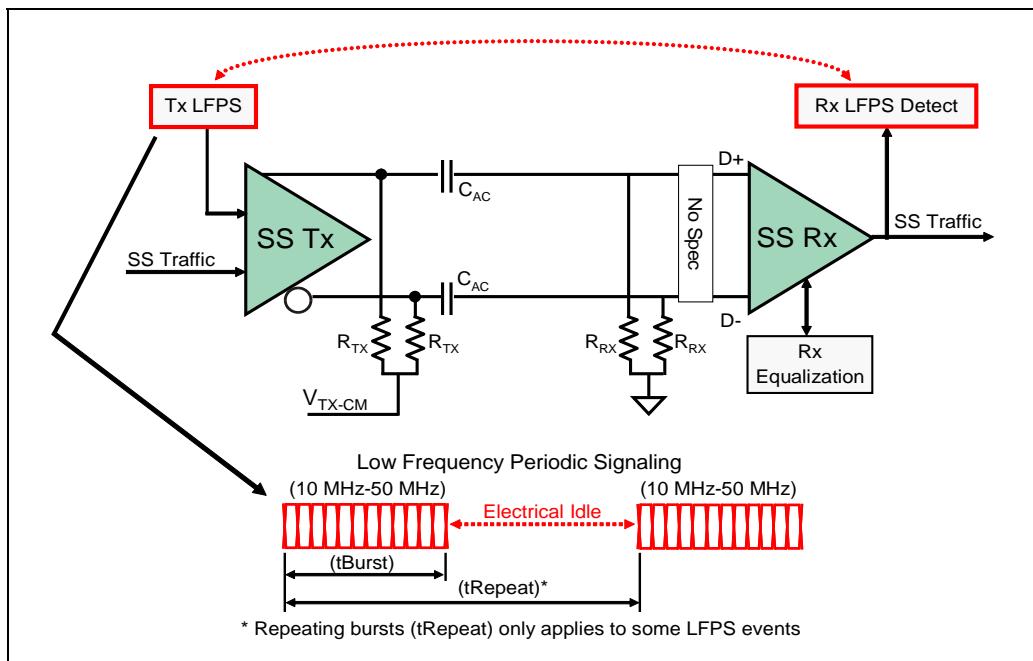
## Chapter 25: SuperSpeed Signaling Requirements

- Signal an exit from power management states U1-U3 or to verify continued device attachment while the link is in a power management state
- Initiate a Warm Reset

### LFPS Is A Simple Squarewave

LFPS is a 10 MHz-50 MHz squarewave sent by the transmitter. The low frequency assures receiver detection even if the link has not yet been trained.

Figure 25-15: Transmitter And Receiver LFPS Signaling Logic



### Six LFPS Signaling Event Types

As shown in Figure 25-15 on page 613, the information carried by LFPS is based on the LFPS burst length ( $t_{Burst}$ ) and whether or not the LFPS pattern repeats ( $t_{Repeat}$ ). In all, there are six LFPS signaling events:

## **USB 3.0 Technology**

---

- Polling.LFPS
- Ping.LFPS
- Warm Reset LFPS
- U1 Exit LFPS
- U2/Loopback Exit LFPS
- U3 Wakeup (Resume) LFPS

*Table 25-7: Timing Requirements For Six LFPS Event Types*

	tBurst				tRepeat		
	Min	Typ	Max	LFPS Cycles	Min	Typ	Max
Polling.LFPS	0.6 $\mu$ s	1.0 $\mu$ s	1.4 $\mu$ s		6 $\mu$ s	10 $\mu$ s	14 $\mu$ s
Ping.LFPS	40 ns		200 ns	2	160 ms	200 ms	240 ms
tReset	80 ms	100 ms	120 ns				
U1 Exit	600 ns		2 ms				
U2/ Loopback Exit	80 $\mu$ s		2 ms				
U3 Wakeup	80 $\mu$ s		10 ms				

The use of each of the six LFPS event types is described in other chapters in the context of their use:

- See Chapter 20, entitled "Link Training," on page 427 for details on Polling.LFPS
- See Chapter 24, entitled "SuperSpeed Power Management," on page 563 for details on U1-U3 Exit LFPS and Ping.LFPS
- See Chapter 19, entitled "SuperSpeed Reset Events," on page 413 for Warm Reset LFPS details

---

## **LFPS Electrical Requirements**

The following table summarizes electrical signaling requirements for transmitters when sending LFPS.

## **Chapter 25: SuperSpeed Signaling Requirements**

---

*Table 25-8: LFPS Electrical Specifications (Normative)*

Symbol	Min	Typ	Max	Units	Comments
tPeriod	20		100	ns	
V <sub>CM-AC-LFPS</sub>			100	mV	
V <sub>CM-LFPS-Active</sub>			10	mV	
V <sub>TX-DIFF-PP-LFPS</sub>	800		1200	mV	LFPS p-p differential amplitude
V <sub>TX-DIFF-PP-LFPS-LP</sub>	400		600	mV	Low power Tx LFPS p-p differential amplitude
tRiseFall2080			4	ns	Note 1
Duty cycle	40		60	%	Note 1

---

## **Transmitter And Receiver DC Specifications**

The following are miscellaneous normative and informative transmitter and receiver DC specifications

---

### **High Impedance Reflections (Normative)**

In the event a device is receiving SuperSpeed data and a reset is received, it is required to disconnect its low impedance terminations. When this occurs, there may be a period when its termination is disconnected (as required) and SuperSpeed traffic is still being received from its link partner. In this case, the absence of termination will cause the differential voltage levels at the receiver inputs to rise by a factor of about 2. The receiver must be capable of tolerating up to 20 ms of this condition without damage for up to 10,000 times over the lifetime of the part.

---

### **ESD Protection (Informative)**

Signal and power pins must withstand 2000 Volts of ESD without damage in compliance with Class 2 JEDEC JESD2-A114F) as well as withstanding 500 Volts

## **USB 3.0 Technology**

---

without damage using the charged device model in compliance with Class 3 JEDEC JESD22-C101D.

---

### **Short Circuit Requirements (Informative)**

Transmitters and receivers must tolerate hot ( $V_{BUS}$  present) insertion and removal without physical damage. Sustained short circuit conditions between the differential transmitter outputs (Txn/Txp) and ground must be tolerated. The same is true for differential receiver inputs (Rxn/Rxp) and ground.

---

---

# 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.

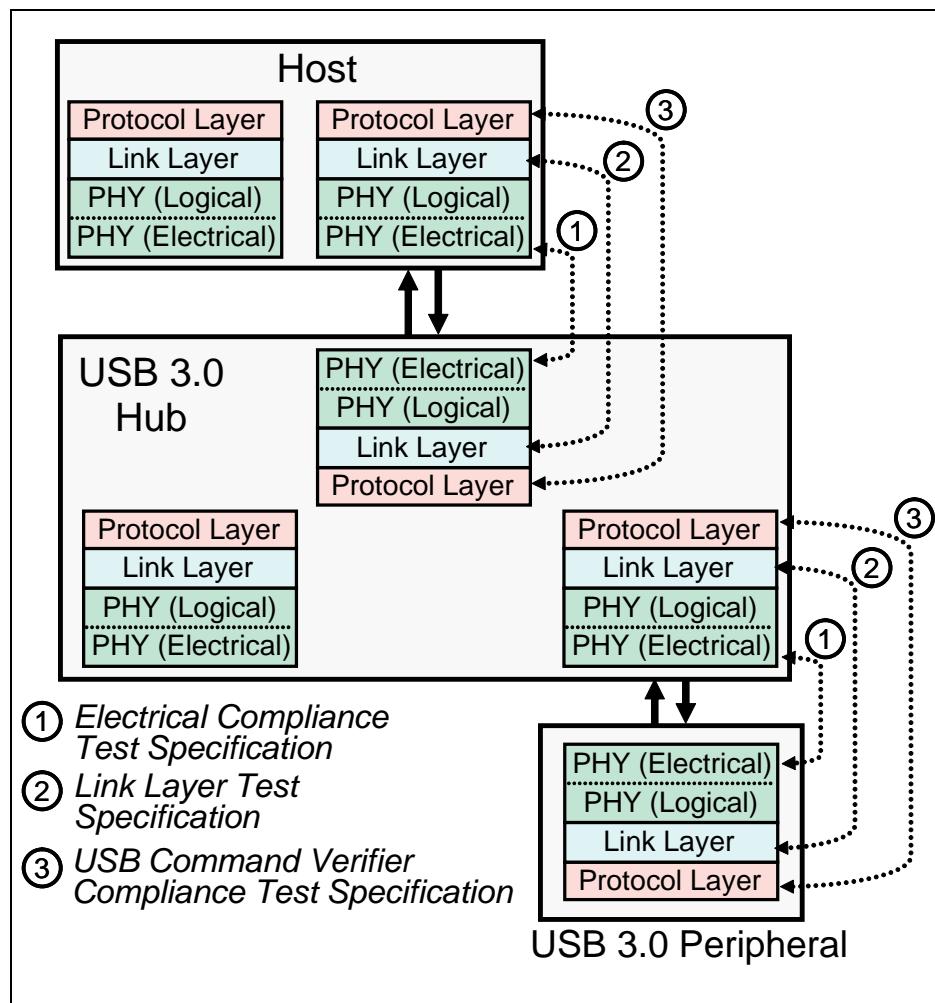
The USB Implementers Forum (USBIF) website, [usb.org](http://usb.org), provides a set of supplementary specifications and white papers covering the full range of compliance testing. Generally speaking, the compliance test specifications describe items to be verified, procedures used to test them, and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

# USB 3.0 Technology

## 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



## **Chapter 26: Compliance Testing**

---

### **Compliance Testing Resources: Documentation**

Compliance testing specifications, white papers, and other documents are available from the [usb.org](http://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](http://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. This 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.

# **USB 3.0 Technology**

---

## **The Compliance Testing Environment**

The process of verifying that host, hub, and peripheral devices meet the requirements of the USB 3.0 Specification involves thorough testing of both logical and electrical behavior. The compliance test environment requires specialized hardware, software, and test fixtures capable of creating hundreds of specific test conditions and evaluating the response of the particular device under test (DUT).

In USB, many aspects of behavior are dependent on the device *port type*. USB 3.0 ports are classified as either downstream facing ports or upstream facing ports and fall into the following categories:

- Host controller root hub ports (these are always downstream facing ports)
- Peripheral ports (these are always upstream facing ports)
- USB 3.0 hub ports (hubs have one upstream facing port and a device-specific number of downstream facing ports).

The distinction between the four port types is important in testing because some compliance checks apply to all ports, other checks apply to only certain port types. This also means that compliance test equipment must be capable of assuming the role of the corresponding link partner as each port type is being tested.

---

## **Key Test Suite 1: USB Command Verifier Compliance**

Governing document: *USB Command Verifier Compliance Test Specification*.

The first of the three key test specifications shown in Figure 26-1 on page 618 is the *USB Command Verifier Compliance Test Specification*. The specification for Command Verification Testing is structured much like the Link Layer Compliance Tests described in the next section, but these tests focus on USB 3.0 Protocol Layer compliance and make the assumption that the port physical layer and link layer are functional.

The compliance tests in this suite verify the response of the device under test to standard USB 3.0 requests. Defined and reserved fields in USB Device, Interface Association, Interface, Configuration, Endpoint/Endpoint Companion, and BOS descriptors are checked. Power management related compliance checks include Suspend/Resume, Remote Wakeup, and Latency Tolerance Message (LTM). Reset behavior is also verified.

# **Chapter 26: Compliance Testing**

---

## **General Approach USB Command Verification Tests**

The group of tests described in this specification apply to all host, hub, and peripheral implementations. This specification is organized in two parts:

- **Test Assertions** — The first section of the *USB Command Verifier Compliance Test Specification* contains a table of compliance criteria to be checked. A set of rules is extracted from content in the Protocol Layer and Device Framework sections of the USB 3.0 Specification. Each test assertion (rule) in the table includes a reference to the chapter/section/subsection in the USB 3.0 Specification where the rule is defined.
- **Test Descriptions** — The second section of the *USB Command Verifier Compliance Test Specification* contains descriptions each of the available tests along with the test assertion(s) to be checked. Some tests are used to verify compliance with multiple test assertions.

---

## **Test Assertions**

There are approximately three hundred test assertions (rules to be checked) listed in the table occupying the first part of the *USB Command Verifier Compliance Test Specification*. The general format of the Test Assertion table is shown in Table 26-1 on page 622. The table also includes a few examples of test assertions; the test assertions shown are all related to the GetDescriptor request as described in Device Framework section (Chapter 9) of the USB 3.0 Specification. A few notes about Table 26-1:

- The left hand column (Assertion #) provides a pointer to USB 3.0 Specification chapter/section where the rule originates. For example: 9.4.3 #1 indicates the first rule associated with USB Specification Chapter 9 (Device Framework), Section 4.3.
- The right hand column indicates which test in this suite checks compliance with the rule. *TI* is the Test Initialization sequence.

## **USB 3.0 Technology**

---

*Table 26-1: Test Assertion Examples: USB Command Verifier Compliance Test Specification*

<b>Assertion #</b>	<b>Assertion Description</b>	<b>Test Number</b>
<b>Subsection Reference: 9.4.3 Get Descriptor</b>		
<b>9.4.3 #1</b>	Devices must respond with a Request Error to GetDescriptor() requests that specify unsupported descriptor types	9.10
<b>9.4.3 #2</b>	Devices must support a valid GetDescriptor(Device) request.	9.1, 9.10, 9.11, 9.14
<b>9.4.3 #3</b>	Devices must support a valid GetDescriptor(DeviceQualifier) request when operating at a USB 2.0 speed.	Chapter 9 tests for USB 2.0 devices
<b>9.4.3 #4</b>	Devices must return the EP companion descriptor in addition to configuration, interface, endpoint descriptors in response to a single GetDescriptor(Configuration) request if operating in SS mode.	9.5, 9.6, 9.9
<b>9.4.3 #5</b>	Devices must support a valid GetDescriptor(String) request.	TI, 9.8
<b>9.4.3 #6</b>	Devices must support a valid GetDescriptor(Configuration) request.	TI, 9.2, 9.3, 9.4, 9.5, 9.6, 9.14, 9.15
<b>9.4.3 #7</b>	High Speed capable devices must support a valid GetDescriptor(OtherSpeedRequest) request.	Chapter 9 tests for USB 2.0 devices
<b>9.4.3 #9</b>	Devices must support a valid GetDescriptor(BOS) request.	9.7
<b>9.4.3 #10</b>	SuperSpeed device shall not report device_qualifier descriptor when operating in SuperSpeed mode	TI, 9.2
<b>9.4.3 #11</b>	SuperSpeed device shall not report other_speed_configuration descriptor when operating in SuperSpeed mode	TI, 9.2

## **Chapter 26: Compliance Testing**

---

Refer to the *USB Command Verifier Compliance Test Specification* for a complete listing of test assertions, but the major groups are summarized below.

### **USB Device State Test Assertions**

Twelve test assertions from Subsection 9.1.1 of the Device Framework chapter of the USB 3.0 Specification. USB Device State test assertions include:

- SS devices must support at least one USB 2.0 speed
- SS device must operate at SuperSpeed if attached to SS host
- USB 3.0 compliant device must reset successfully at a USB 2.0 environment
- Devices must use default address (0) on power up or reset
- Devices must enter suspended state when upstream link enters U3
- Devices must exit suspend when upstream link signaling is detected
- If device is capable of remote wake, it must support host enable/disable
- On a device reset, remote wakeup must be disabled

### **Generic Device Operation Test Assertions**

Twenty two test assertions from Subsection 9.2 of the Device Framework chapter of the USB 3.0 Specification. Examples include:

- Device configuration must complete before any of its functions may be used
- Devices are limited to a max of one V<sub>BUS</sub> unit load until configuration.
- SuperSpeed devices are limited to six V<sub>BUS</sub> unit loads (900 mA) after configuration. SS devices in 2.0 environment are limited to 500mA, maximum.
- Devices maintain USB device address, configuration, and on resume from suspend state.
- If link is not in U0 on exit from function suspend, device shall transition link to U0 state to send remote wake message.
- If remote wake occurs in multiple device functions, each must send a Function Wake
- Devices must not respond to old device address after Status Stage of SetAddress command completes
- Devices must process any command in no more than 5 seconds
- For standard device requests with no Data Stage, device must complete Status Stage in less than 50mS
- For standard device requests with a Data Stage transfer to the host, device must return first data packet within 500mS. After final data packet, device must complete Status Stage in less than 50mS.

# **USB 3.0 Technology**

---

## **USB Device Request Test Assertions**

Test assertions from Subsection 9.3 and 9.4 of the Device Framework chapter of the USB 3.0 Specification. Device request test assertion groups include:

- Usage of STALL Transaction Packet
- ClearFeature Requests
- GetConfiguration Requests
- GetDescriptor Requests
- GetInterface Requests
- GetStatus Requests
- SetAddress Requests
- SetConfiguration Requests
- Set Descriptor Requests
- SetFeature Requests
- SetInterface Requests
- SetIsochronousDelay Requests
- SetSEL (System Exit Latency) Requests

## **USB Descriptor Definition Test Assertions**

This group of test assertions are from Subsection 9.6 of the Device Framework chapter of the USB 3.0 Specification. Used in descriptor format checks.

- Device Descriptor fields
- Binary Device Object Store (BOS) Descriptor types and fields
- Configuration Descriptor fields
- Interface Descriptor fields
- Endpoint Companion Descriptor fields
- String Descriptor fields

---

## **Nineteen Tests Used In USB Command Verification**

The following tests are described in the *USB Command Verifier Compliance Test Specification* and used to verify compliance with the list of test assertions described in the previous section. Each test is defined in a *test description* which includes the test description (TD) number, the assertion(s) to be checked, and a set of test procedures. The specification makes the point that the test procedure steps in the specification are listed mainly to provide insight into the general sequence of events--the details are left up to the test software and hardware implementation.

A list of test descriptor numbers and names is shown in Table 26-2.

## **Chapter 26: Compliance Testing**

---

*Table 26-2: List Of Test Descriptions For The USB Command Verifier Compliance Specification*

<b>Test Descriptor Number (TD #)</b>	<b>Test Name</b>
<b>Physical Layer Test Descriptors</b>	
<b>TD.9.1</b>	Device Descriptor Test
<b>TD.9.2</b>	Standard Configuration Descriptor Test
<b>TD.9.3</b>	Standard Interface Association Descriptor Test
<b>TD.9.4</b>	Standard Interface Descriptor Test
<b>TD.9.5</b>	Endpoint Descriptor Test
<b>TD.9.6</b>	SuperSpeed Endpoint Companion Descriptor Test
<b>TD.9.7</b>	BOS and Device Capability Descriptor Test
<b>TD.9.8</b>	String Descriptor Test
<b>TD.9.9</b>	Halt Endpoint Test
<b>TD.9.10</b>	Bad Descriptor Test
<b>TD.9.11</b>	Bad Feature Test
<b>TD.9.12</b>	Remote Wakeup Test
<b>TD.9.13</b>	Set Configuration Test
<b>TD.9.14</b>	Suspend/Resume Test
<b>TD.9.15</b>	Function Remote Wakeup Test
<b>TD.9.16</b>	Enumeration Test
<b>TD.9.17</b>	LTM Test
<b>TD.9.18</b>	Reset Device Test
<b>TD.9.19</b>	Control Transfer Timing Test
<b>XXXXXX</b>	Other Tests: SS devices must be tested at all supported USB 2.0 speeds SS devices must also pass any class-specific tests.

# **USB 3.0 Technology**

---

## **Key Test Suite 2: Link Layer Compliance**

Governing document: *USB Serial Bus 3.0 Link Layer Test Specification*

The second of the three key test specifications shown in Figure 26-1 on page 618 that guide compliance testing of USB 3.0 host, hub, and peripheral ports is the *USB Serial Bus 3.0 Link Layer Test Specification*. Link layer testing is a very comprehensive area of compliance verification and includes many types of tests applying to all port types. As the specification title suggests, most of the tests are related to the USB 3.0 SuperSpeed Link Layer, but some checks are also made of Protocol Layer and Physical Layer (logical) compliance criteria.

---

## **General Approach In Link Layer Compliance Tests**

The group of tests described in this specification apply to all host, hub, and peripheral implementations. The specification is organized in two parts:

- **Test Assertions.** The first section of the *Link Layer Test Specification* is comprised of a table containing compliance criteria to be checked. This is a set of rules based on key content in the USB 3.0 Specification. Each test assertion (rule) in the table includes a reference to the chapter/section/subsection in the USB 3.0 Specification where the rule is defined.
- **Test Descriptions.** The second section of the *Link Layer Test Specification* describes each of the available compliance tests along with the test assertion(s) it checks. Some tests are used to verify compliance with multiple test assertions.

---

## **Test Assertions**

There are approximately two hundred test assertions (rules to be checked) listed in the table occupying the first part of the *Link Layer Test Specification*. The general format of the Test Assertion table is shown in Table 26-3. The table also includes a few examples of test assertions; the assertions shown are all related to data packet payload formatting rules found in Link Layer chapter of the USB 3.0 Specification. A few notes about Table 26-3:

- The left hand column (Assertion #) provides a pointer to USB 3.0 Specification chapter/section where the rule originates. For example: 7.2.1.2.1 #1 indicates the first rule associated with USB Specification Chapter 7 (Link Layer), Section 2.1.2.1.

## **Chapter 26: Compliance Testing**

---

- The right hand column (Test Number) indicates which of the link layer Tests checks compliance with the rule. Often this is a number; in the example shown, *IOP* is the Inter operability Test suite; *NT* indicates an item not explicitly tested or tested indirectly by other compliance checks.

*Table 26-3: Test Assertion Examples From Link Layer Test Specification*

<b>Assertion #</b>	<b>Assertion Description</b>	<b>Test Number</b>
<b>Subsection Reference: 7.2.1.2.1 Data Packet Payload Framing</b>		
7.2.1.2.1 #1	Data packet payload (DPP) shall always begin with the DPPSTART ordered set	IOP
7.2.1.2.1 #2	DPP shall always end with the DPPEND ordered set	IOP
7.2.1.2.1 #3	DPP with abnormal ending shall end with the DPPABORT ordered set	NT
<b>Subsection Reference: 7.2.1.2.2 Data Packet Payload</b>		
7.2.1.2.2 #1	Data packet payload (DPP) shall consist of 0 to 1024 bytes immediately followed by the 4-byte CRC-32.	IOP
7.2.1.2.2 #2	CRC-32 for DPP shall be calculated as specified when transmitted.	IOP
7.2.1.2.2 #3	CRC-32 for DPP with a 0-byte payload shall be the value 0x00000000.	NT
7.2.1.2.2 #4	DPP shall immediately follow its header (DPH) with no symbols in between.	IOP

The remainder of the test assertions from the *Link Layer Test Specification* are summarized below, by functional group.

### **Link Management & Flow Control Test Assertions**

Over one hundred test assertions from Subsection 7.2 of the Link Layer chapter of the USB 3.0 Specification. Link Management & Flow Control test assertion categories include:

# **USB 3.0 Technology**

---

- Header packet (HP) assembly and framing
- Header packet *link control word* usage
- Data packet payload (DPP) format & framing
- Link command usage: assembly, framing, placement
- Header packet flow control, timers, initialization, LCRD\_x usage
- Header packet acknowledgement, LGOOD\_n, LBAD usage
- Header packet sequence number & flow control credit advertisements
- Link power management, U1/U2 entry, inactivity timers
- Link power management, U3 entry and exit

## **Link Error Rules & Recovery Test Assertions**

Test assertions from Subsection 7.3 of the Link Layer chapter of the USB 3.0 Specification. Link Error Rules test assertion categories include:

- Link command errors
- ACK Tx header sequence number errors
- Header sequence number advertisement errors
- Rx header buffer credit advertisement errors
- Training sequence errors

## **Link Reset Test Assertions**

Test assertions from Subsection 7.4 of the Link Layer chapter of the USB 3.0 Specification. Link Reset test assertion categories include:

- PowerOn reset
- In-band resets (warm reset and hot reset)

## **LTSSM Test Assertions**

Nearly two hundred test assertions related to the Link Training and Status State machine (LTSSM) operation are based on Subsection 7.5 of the Link Layer of the USB 3.0 Specification. Some of the LTSSM test assertion categories include:

- Rx.Detect substate requirements and entry/exit
- Polling substate requirements and entry/exit
- U0 state requirements and entry/exit
- U1 state requirements and entry/exit
- U2 state requirements and entry/exit
- U3 state requirements and entry/exit
- Recovery substate requirements and entry/exit
- SS.Inactive substate requirements and entry/exit

## **Chapter 26: Compliance Testing**

---

- SS.Disabled state requirements and entry/exit
- Loopback substate requirements and entry/exit
- Hot Reset substate requirements and entry/exit
- Compliance Mode state requirements and entry/exit

### **Protocol Layer Test Assertions**

Test assertions from Subsection 8.3, 8.4, and 8.6 of the Protocol Layer chapter of the USB 3.0 Specification. Protocol Layer test assertion categories include:

- Receiver handling of reserved Header Packet fields
- Handling of received Force\_LinkPM\_Accept bit LMPs
- Usage of Vendor Device Test LMP at run time
- Port Capability LMP transmission and reception
- Port Configuration LMP transmission and reception
- Port Configuration Response LMP transmission and reception

### **Hub Port Test Assertions**

Test assertions from Subsection 10.2, 10.3, 10.4 of the Hub chapter of the USB 3.0 Specification. Hub and hub downstream/upstream port test assertion categories include:

- Hub power management and handling of downstream port U1/U2 entry attempts if packets from upstream port are pending.
- Hub avoidance of race conditions when header is queued for transmission on downstream port at the same time U1/U2 entry is in progress.
- Hub recovery from failed Hot Reset on downstream port by automatically issuing Warm Reset.
- Hub transition of downstream facing port to Disconnected if Rx.Detect timer expires after Warm Reset.
- Hub avoidance of race condition on downstream port if SetPortFeature(PortLinkState) is received at the same time link partner requests U1/U2 entry.

---

### **Forty Tests Are Used In Link Layer Testing**

Over forty tests are described in the *Link Layer Test Specification* and used to verify compliance with the extensive list of test assertions described in the previous section. To check a port under test (PUT) for compliance, a series of tests are performed. Each test is defined in a *test description* which includes the test description (TD) number, the assertion(s) to be checked, and a set of test proce-

## **USB 3.0 Technology**

---

dures. The specification makes the point that the test procedure steps in the specification are listed mainly to provide insight into the general sequence of events--the details are left up to the test software and hardware implementation.

A list of test descriptor numbers and names is shown in Table 26-4.

*Table 26-4: List Of Test Descriptions For The Link Layer Test Specification*

<b>Test Descriptor Number (TD #)</b>	<b>Test Name</b>
<b>Physical Layer Test Descriptors</b>	
TD.6.1	Lane Polarity Inversion Test
TD.6.2	Skip (SKP) Ordered Set Test
TD.6.3	Elasticity Buffer Test
TD.6.4	LFPS Frequency Test
TD.6.5	Polling_LFPS Duration Test
<b>Link Layer Test Descriptors</b>	
TD.7.1	Link Bring-up Test
TD.7.2	Link Commands Framing Robustness Test
TD.7.3	Link Commands CRC-5 Robustness Test
TD.7.4	Invalid Link Commands Test
TD.7.5	Header Packet Framing Robustness Test
TD.7.6	Data Packet Payload Framing Robustness Test
TD.7.7	Rx Header Packet Retransmission Test
TD.7.8	Tx Header Packet Retransmission Test
TD.7.9	PENDING_HP_TIMER Deadline Test
TD.7.10	CREDIT_HP_TIMER Deadline Test

## **Chapter 26: Compliance Testing**

---

*Table 26-4: List Of Test Descriptions For The Link Layer Test Specification (Continued)*

<b>Test Descriptor Number (TD #)</b>	<b>Test Name</b>
<b>TD.7.11</b>	PENDING_HP_TIMER Timeout Test
<b>TD.7.12</b>	CREDIT_HP_TIMER Timeout Test
<b>TD.7.13</b>	Wrong Header Sequence Test
<b>TD.7.14</b>	Wrong LGOOD_N Sequence Test
<b>TD.7.15</b>	Wrong LCRD_X Sequence Test
<b>TD.7.16</b>	Link Command Missing Test
<b>TD.7.17</b>	tPortConfiguration Time Timeout Test
<b>TD.7.18</b>	Low Power Initiation For U1 Test
<b>TD.7.19</b>	Low Power Initiation For U2 Test
<b>TD.7.20</b>	PM_LC_TIMER Deadline Test
<b>TD.7.21</b>	PM_LC_TIMER Timeout Test
<b>TD.7.22</b>	PM_ENTRY_TIMER Timeout Test
<b>TD.7.23</b>	Accepted PM Transaction For U1 Test
<b>TD.7.24</b>	Accepted PM Transaction For U2 Test
<b>TD.7.25</b>	Accepted PM Transaction For U3 Test
<b>TD.7.26</b>	Transition To U0 From Recovery Test
<b>TD.7.27</b>	Hot Reset Detection In Polling Test
<b>TD.7.28</b>	Hot Reset Detection In U0 Test
<b>TD.7.29</b>	Hot Reset Initiation In U0 Test
<b>TD.7.30</b>	Recovery Transition On 3 Consecutive Rx Header Packets Test
<b>TD.7.31</b>	Hot Reset Failure (trigger automatic Warm Reset) Test
<b>TD.7.32</b>	Warm Reset On Rx.Detect Timeout Test
<b>TD.7.33</b>	Exit Compliance Mode Test (upstream facing ports)

## **USB 3.0 Technology**

---

*Table 26-4: List Of Test Descriptions For The Link Layer Test Specification (Continued)*

<b>Test Descriptor Number (TD #)</b>	<b>Test Name</b>
<b>TD.7.34</b>	Exit Compliance Mode Test (downstream facing ports)
<b>TD.7.35</b>	Exit U3 By Reset Test (downstream facing ports)
<b>TD.7.36</b>	Exit U3 Test (host ports only)
<b>TD.7.37</b>	Packet Pending Test (upstream facing ports only)

---

## **Electrical Compliance Testing**

The remainder of this chapter deals with compliance testing of the USB 3.0 SuperSpeed physical layer electrical interface.

---

### **Factors Affecting USB Electrical Compliance Testing**

USB was originally a peripheral bus for desktop PCs. Over time, it also became widely used in other platform types ranging from embedded systems to laptops, workstations, and servers. Electrical compliance testing is complicated by the fact that requirements are slightly different for hosts, hubs, and devices as well as because of several other factors, including:

- The wide range of USB physical connection schemes that are possible between the host controller and connectors the platform exposes for external USB device attachment. For example, a desktop or laptop may require a short run of pc board traces between the host controller and each of the USB connectors. A server may require motherboard pc trace routing to rear panel connectors as well as pc traces to an internal connector which is cabled to the a second pc board that manages devices attached to front panel USB connectors.
- The attachment method between USB connectors and devices, ranging from dongles to cables up to 5 meters for USB 2.0 and 3 meters for USB 3.0.

---

### **Electrical Compliance And The Varied USB Topology**

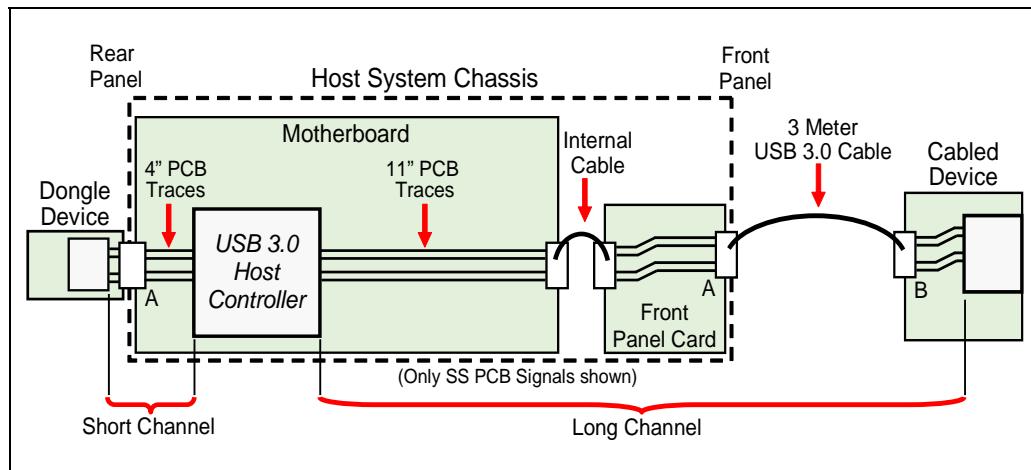
As described earlier in Chapter 20, entitled "Link Training," on page 427, USB is

## Chapter 26: Compliance Testing

found in systems ranging from embedded platforms to PCs, workstations, and servers. As with link training, electrical compliance testing must thorough enough to account for the wide range of possible physical connections between devices, including combinations of cables, connectors, and chip-to-chip pc board traces.

USB 3.0 refers to the connection between link partners as the *channel*. Depending on pc board trace length, presence of connectors and cables, etc., a channel may be described as long or short. The typical system has a combination of long and short channels as shown in Figure 26-2 on page 633. Electrical compliance testing should verify that a device anywhere in the topology will function reliably.

Figure 26-2: Compliance Testing Accounts For Long & Short Channels



### The Long Channel

The device at the right in Figure 26-2 on page 633 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, that 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.

## **USB 3.0 Technology**

---

### **The Short Channel**

By contrast, the USB “dongle” device at the left in Figure 26-2 on page 633 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.

---

### **USB 3.0 Adds Even More Complexity**

USB 3.0 SuperSpeed features add to the complexity of, and time required to perform, electrical compliance testing. Some of these include:

- The tighter timing requirements of the new 5GT/s data rate
- Validation of both directions of the AC-coupled, dual-simplex link
- Spread-spectrum clocking (SSC)
- Low Frequency Periodic Signaling (LFPS)
- Transmitter de-emphasis and receiver equalization

---

### **Electrical Compliance Test Support Is Designed-In**

An early step in device validation is verification that SuperSpeed device transmitter timing and voltage specifications are within the limits allowed in USB 3.0. This involves performing measurements at the transmitter output using self-generated data patterns specifically chosen for this purpose. Once the transmitter has been checked, similar measurements can be made at receiver inputs to determine the effects of a particular SuperSpeed channel and whether minimum requirements for receiver timing and voltage are met.

The USB 3.0 specification defines a standard set of transmitter compliance test patterns and a simple protocol for a controlling device, acting as the link partner, to initiate compliance mode entry/exit and sequence through the test patterns as measurements are made. Compliance Mode is one of the special states of the link training and Status State Machine (LTSSM) and its major features are described in the following section.

## Chapter 26: Compliance Testing

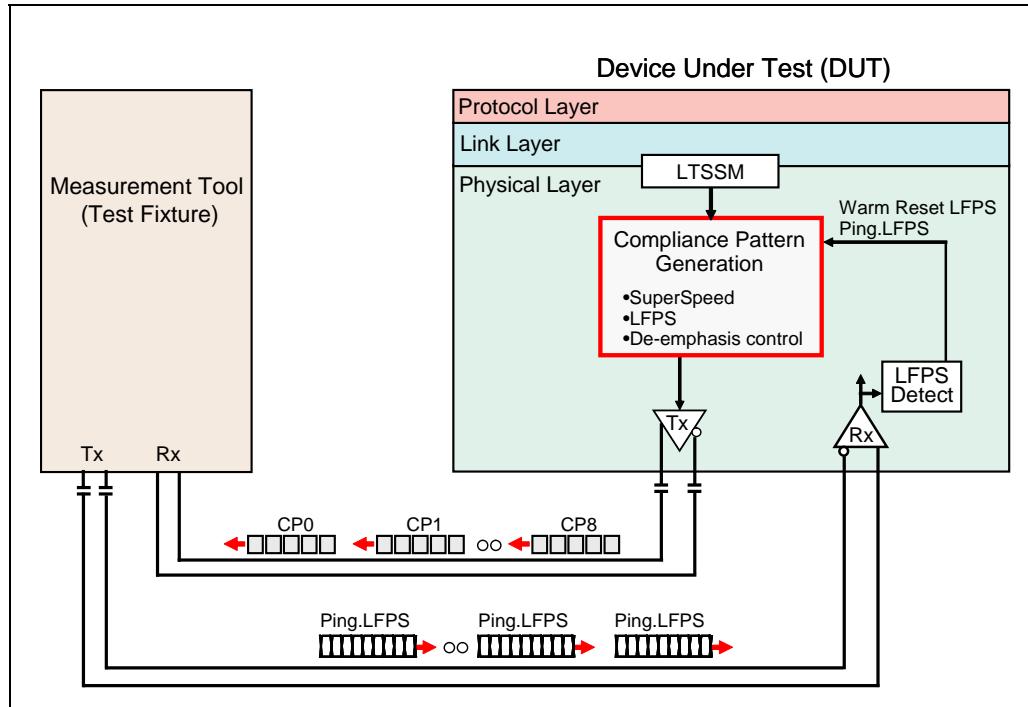
### LTSSM Compliance Mode: Device Roles

Figure 26-3 on page 635 illustrates the relationship between the device under test (transmitter of test patterns) and its link partner responsible for managing the compliance test (referred to here as the tester). All ports must support compliance mode as the device under test (DUT).

Some things to note in Figure 26-3 on page 635:

- The device under test will enter compliance mode and transmit compliance patterns CP0-CP8 under control of the tester.
- Once compliance mode is entered, the tester will use Low Frequency Periodic Signaling transmitter Ping.LFPS bursts to indicate readiness for the next test pattern.
- Voltage and timing measurements can made at the transmitter output as well as the receiver input.

Figure 26-3: Compliance Mode Test Configuration



# **USB 3.0 Technology**

---

## **Nine Compliance Mode Test Patterns**

The data patterns generated as part of compliance mode testing consist of nine variants which occur in a fixed sequence. Compliance patterns CP0-CP8 include pseudo-random data, high and low frequency square waves, and symbols sent with and without transmitter de-emphasis.

CP0 is pseudo-random data made up of scrambled and encoded NOPs. The remaining compliance patterns, CP1-CP8 are not scrambled and are intended to yield consistent transmit bit patterns.

Compliance patterns do not include SKPs (Skip) ordered sets as compliance testing does not include receiver processing of data. Table 26-5 summarizes compliance pattern features and the order in which they appear in the fixed test sequence.

*Table 26-5: Compliance Pattern Sequence*

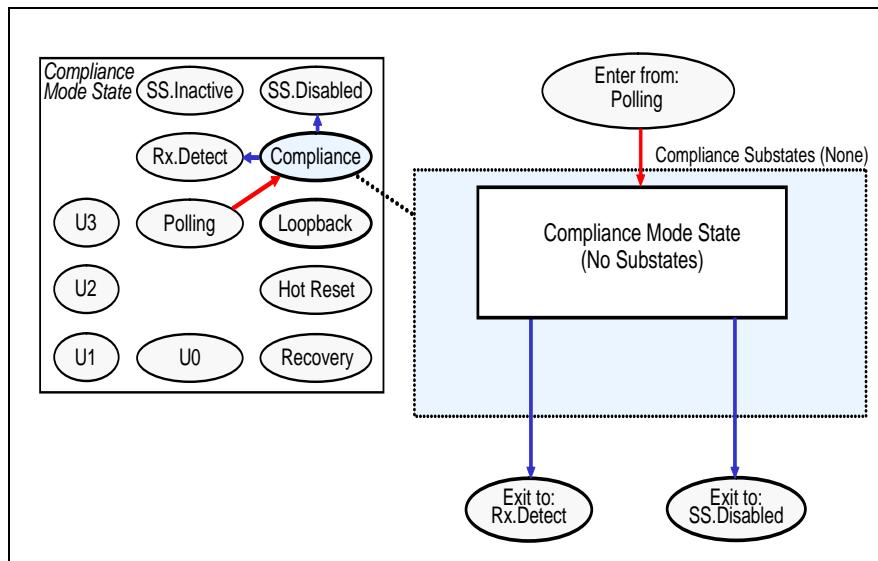
<b>Pattern</b>	<b>Description</b>	<b>Notes</b>
CP0	D0.0 Logical Idle Symbol (NOP)	Only pattern that is scrambled
CP1	D10.2 Symbol	Nyquist toggling 0's and 1's (2.5GHz square wave)
CP2	D24.3 Symbol	Nyquist/2 toggling 0's and 1's
CP3	D28.5 Symbol	COM symbols
CP4	LFPS Square Wave	10-50MHz Low Frequency Periodic Signaling
CP5	K28.7 Symbol	Nyquist/3 with de-emphasis
CP6	K28.7 Symbol	Nyquist/3 without de-emphasis
CP7	50 to 250 repeating 1's, then 50-250 repeating 0's	Repeating 1-5MHz square wave pattern without de-emphasis.
CP8	50 to 250 repeating 1's, then 50-250 repeating 0's	Repeating 1-5MHz square wave pattern with de-emphasis

## Chapter 26: Compliance Testing

### Electrical Compliance Testing And The LTSSM

As depicted in Figure 26-4, Compliance is one of twelve high level states of the Link Training and Status State Machine (LTSSM) that, conceptually, resides at the link layer of each USB 3.0 device. Note that here are no substates associated with Compliance.

Figure 26-4: Compliance LTSSM State



#### Entry from Polling-

A device enters Compliance Mode from Polling.LFPS if, on the first attempt at link training after PowerOn Reset, the 360 ms timer initialized on entry to Polling has expired without successful completion of Polling.LFPS handshake. If the Polling.LFPS burst is not received during the first Polling.LFPS opportunity, a passive ( $50 \Omega$ ) load is assumed to be present on the differential signal lines; this is an indication to the receiving device that it should enter Compliance Mode and transmit the first test pattern.

## **USB 3.0 Technology**

---

*Requirements while in Compliance Mode-*

- Both devices maintain low-impedance receiver termination ( $R_{RX-DC}$ ).
- LFPS receiver of the device sending the test patterns remains enabled to allow the other device to signal when the compliance pattern should be advanced.
- Device sending compliance test patterns must wait until its transmitter common mode voltage specification ( $V_{TX-DC-CM}$ ) is met.
- Each time a valid Ping.LFPS burst is received, device transmitting compliance test patterns advances to the next pattern. When CP8, the last test pattern, is active and valid Ping.LFPS is received, the pattern rolls over to CP0 and holds until compliance testing ends.

*Exit from Compliance Mode-*

- Downstream facing port transitions to Rx.Detect when it is directed to send a Warm Reset.
- Upstream facing port transitions to Rx.Detect when valid Warm Reset LPFS signaling (18-120 ms) is detected.
- Downstream facing port may also be directed to SS.Disabled during compliance testing.

---

---

# 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 interoperability 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.

---

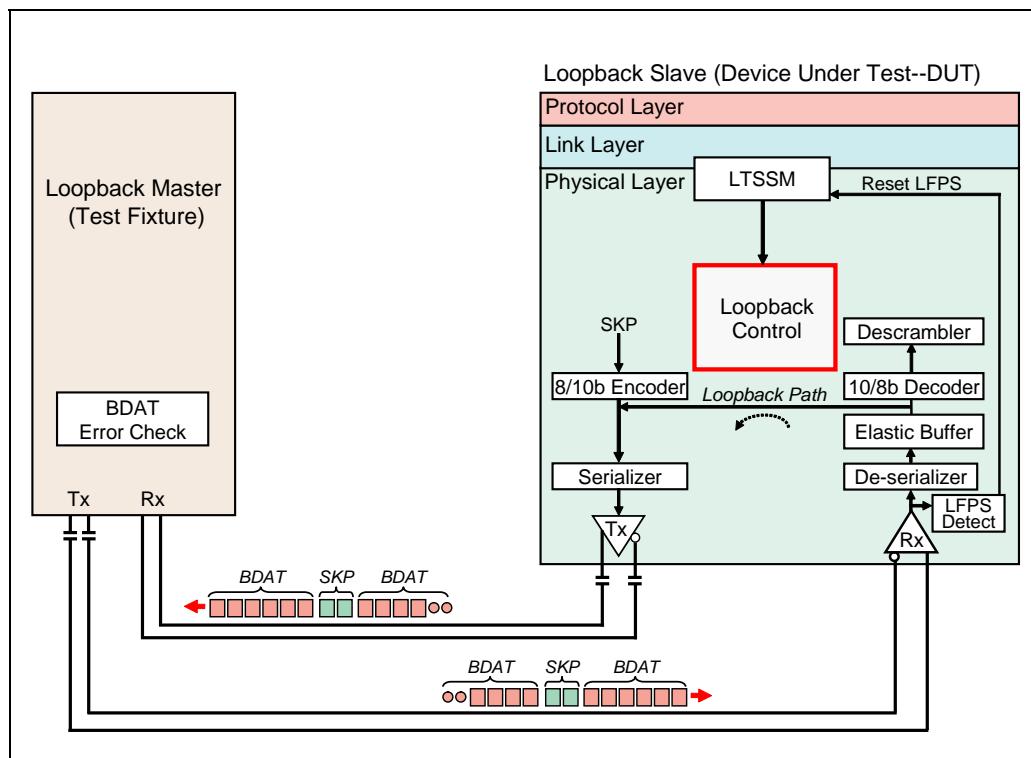
## **Standard Loopback Configuration**

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  $10^{-12}$ .

## USB 3.0 Technology

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.

## Chapter 27: Receiver Loopback Testing

---

- 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

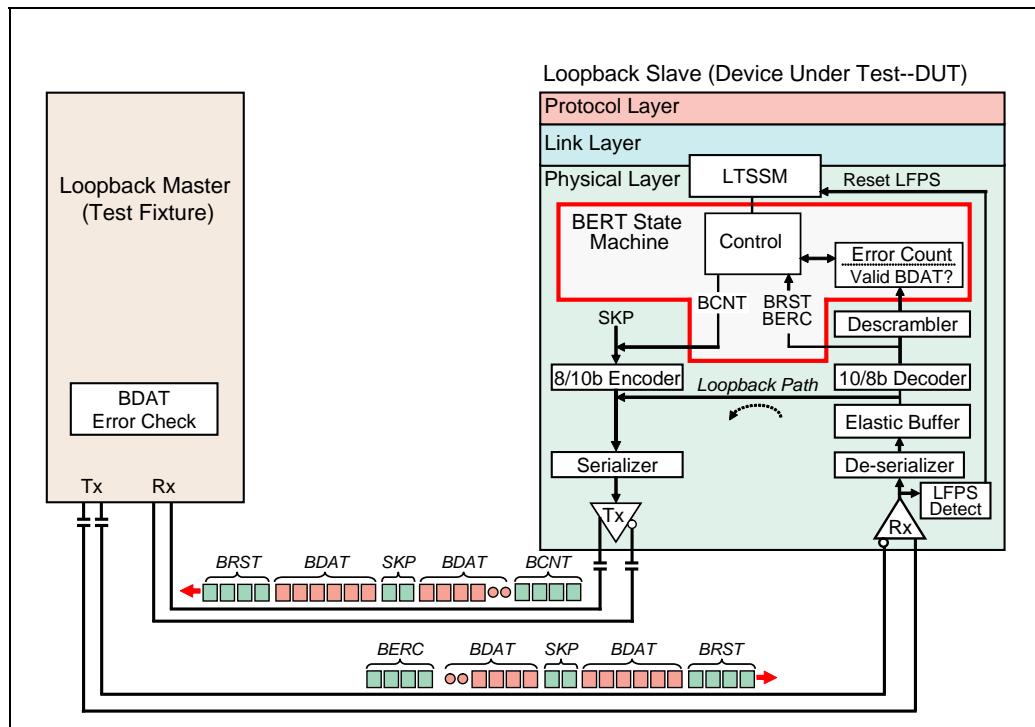
## USB 3.0 Technology

---

sets used by the master to clear or request the accumulated slave error count. By comparing the error count reported by the loopback slave to the error count it tracks for “looped back” data, the master can quantify the bit error rate (BER) for each link direction.

Figure 27-2 on page 642 depicts a SuperSpeed device with support for the optional loopback slave bit error rate test (BERT) state machine and error counting.

Figure 27-2: Loopback Ordered Sets & Slave BERT Logic



---

### Loopback BERT Ordered Sets

As depicted in Figure 27-2 on page 642, the loopback master and slave exchange BERT test pattern data (BDAT) and three ordered sets used by the loopback master to manage the slave BERT state machine and retrieve the slave BDAT error count.

## **Chapter 27: Receiver Loopback Testing**

---

### **BERT Reset (BRST)**

Loopback BRST is an ordered set command requesting the slave to clear its 16-bit BDAT error counter. This ordered set is only sent by the loopback master and consists of the two 10-bit symbols shown in Table 27-2. BRST may be repeated as often as necessary by the master. As with other loopback traffic, BRST is retransmitted by the slave.

*Table 27-2: Loopback BRST Ordered Set*

Symbol Number	Encoding	Description
0	K28.5	COM symbol
1	K28.7	BRST symbol. Used in Bit Error Rate Test (BERT) protocol to reset error count at device under test

---

### **BDAT Error Count Request (BERC)**

Loopback BERC is an ordered set command requesting the slave to return the error count accumulated for received BDAT symbols since the last BRST was received. This ordered set is sent by the master; format is shown in Table 27-3.

*Table 27-3: Loopback BERC Ordered Set*

Symbol Number	Encoding	Description
0	K28.3	BERC symbol
1	K28.3	BERC symbol
2	K28.3	BERC symbol
3	K28.3	BERC symbol

# **USB 3.0 Technology**

---

## **BDAT Error Count (BCNT) Value**

Upon receipt of the BERC ordered set, a loopback slave that supports BERT state machine and error counting returns the BCNT ordered set. BCNT consists of the four 10-bit symbols shown in Table 27-4. Note that the first two symbols are BERC (K28.3) and the last two symbols contain the 16-bit (two byte) error count. The error count symbols are labeled *DCODE* in the table and are not scrambled.

*Table 27-4: Loopback BCNT Ordered Set*

Symbol Number	Encoding	Description
0	K28.3	BERC symbol
1	K28.3	BERC symbol
EC	DCODE	BCNT. First byte of 16-bit error count
EC	DCODE	BCNT. Second byte of 16-bit error count

---

## **Slave BERT Command Processing Rules**

---

### **General**

- Receipt of BRST causes the error count to be cleared (set = 0) and the scrambling LFSR to be set to its initial value: 0FFFFh.
- Receipt of BRST followed by BDAT informs the slave that loopback bit error rate testing (BERT) has started.
- During the BERT, the slave uses the scrambler output to validate received BDAT NOP symbols.
- Error count (EC) increments by one for each BDAT symbol received in error. While in loopback, the slave 16-bit error count is not permitted to roll over and holds at 0FFFFh until a BRST is received. BERC requests from the master do not clear the count and may be sent as often as required.

## Chapter 27: Receiver Loopback Testing

### BRST & BERC If No Slave Error Counting Support

If BRST and BERC ordered sets are received by a device that does not support the slave BERT state machine and BDAT error counting feature, it simply loops back the ordered sets and does not return BCNT.

### Loopback And The LTSSM

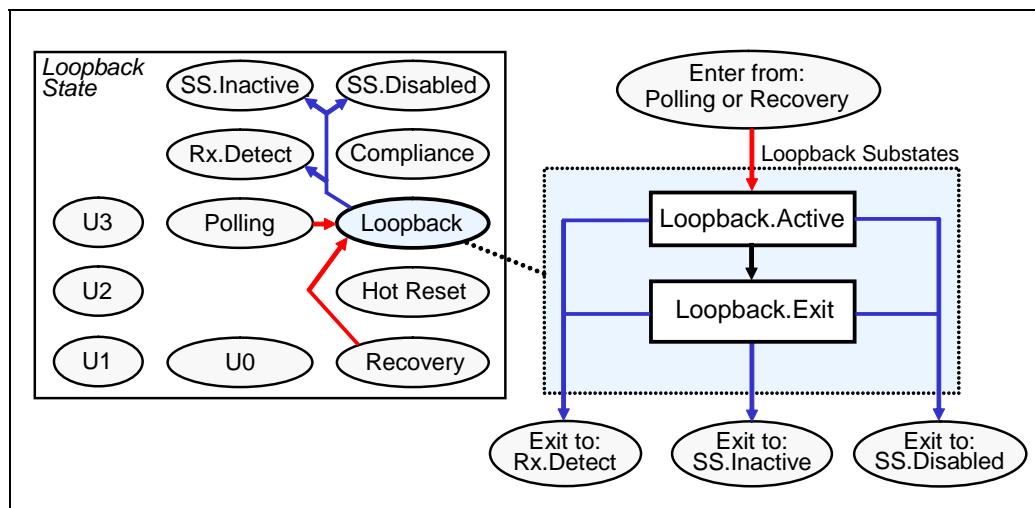
#### General

As depicted in Figure 27-3 on page 645, the Loopback State is one of twelve high level states of the Link Training and Status State Machine (LTSSM) that, conceptually, resides at the link layer of each USB 3.0 device.

The LTSSM State Machine provides a convenient way to view Loopback entry, exits, handshake signaling, and time-outs for the two Loopback substates:

- Loopback.Active
- Loopback.Exit

Figure 27-3: Loopback LTSSM State And Substates

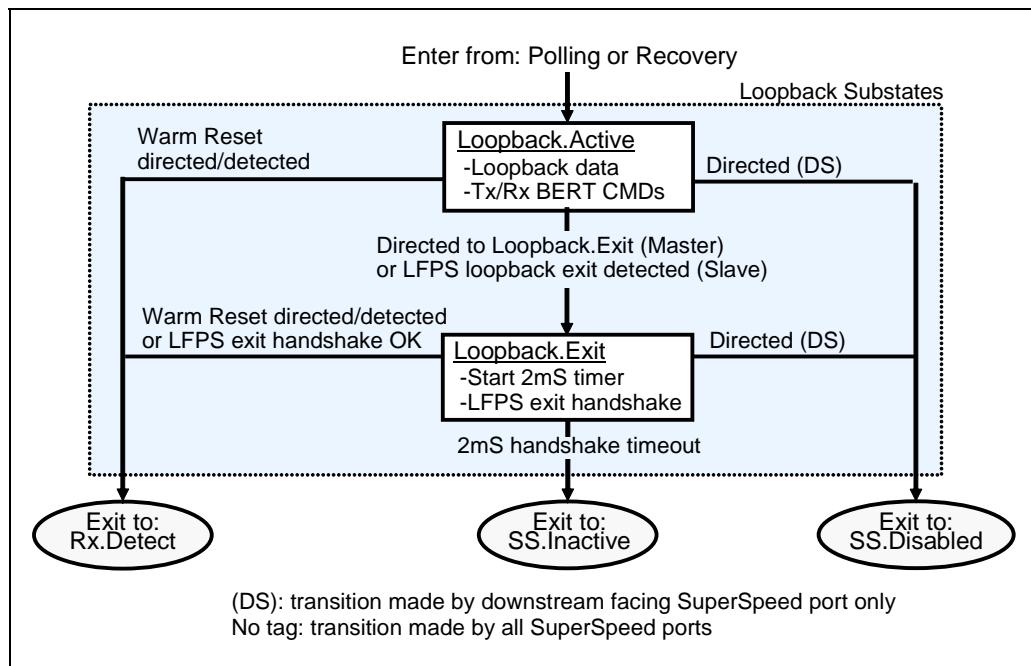


# USB 3.0 Technology

## Loopback LTSSM Substates And Transition Events

Refer to Figure 27-4 on page 646 in the following discussion of Loopback LTSSM substate requirements and transition events.

Figure 27-4: Loopback LTSSM Substate Requirements & Transitions



### Loopback.Active Substate

#### Entry from Polling-

If a link partner sets the loopback bit = 1 in the Polling.Configuration TS2s (loopback is bit 2 in TS2 Symbol 5), the other device recognizes that a transition to loopback will occur during the last substate of Polling (Polling.Idle). The device detecting the loopback bit set will act as loopback slave.

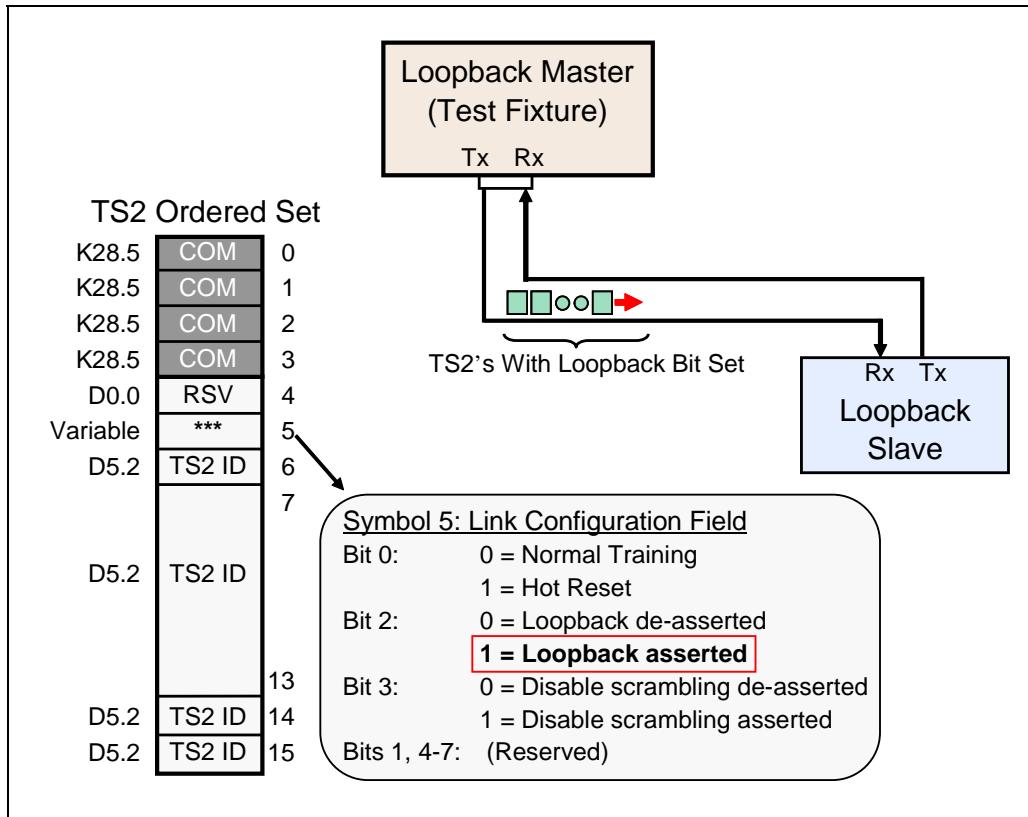
#### Entry from Recovery-

During the Recovery.Configuration substate, link partners always exchange TS2 ordered sets in a SuperSpeed handshake; this provides another opportunity to set the TS2 loopback bit. If the TS2 loopback bit is set = 1, the LTSSM transitions from Recovery to Loopback.Active.

## Chapter 27: Receiver Loopback Testing

Figure 27-5 on page 647 depicts the loopback bit in TS2 Symbol 5. Note that TS2 Symbol 5 is also referred to as the *Link Configuration Field*.

Figure 27-5: TS2 Loopback Entry Signaling



Requirements while in Loopback.Active substate-

- Loopback master sends pseudo-random 10-bit loopback data consisting of scrambled and encoded logical 0's (NOPs). Clock compensation SKP ordered sets are inserted into the pattern using standard rules.
- Slave retransmits the received loopback data symbols. It may not modify received symbols, other than correcting differential polarity inversion and inserting/deleting elastic buffer SKPs as necessary.
- Slave must also recognize and process BERT commands interleaved with loopback data.

## **USB 3.0 Technology**

---

- LFPS receiver must remain enabled so Loopback LFPS exit signaling can be recognized.

*Exit from Loopback.Active substate-*

As depicted in Figure 27-4 on page 646, there are three exits from Loopback.Active substate

- Downstream facing ports transition to SS.Disabled when directed.
- Downstream facing ports transition to Rx.Detect state when directed to issue a Warm Reset on the link.
- Upstream facing ports transition to Rx.Detect state when the Warm Reset is detected.
- The loopback master transitions to Loopback.Exit substate when directed.
- The loopback slave transitions to Loopback.Exit when Loopback LFPS exit signaling is detected.

### **Loopback.Exit Substate**

In this substate, the loopback master has completed testing, signaled an exit from loopback, and is waiting for the Loopback LFPS exit handshake with its link partner to complete.

*Requirements While In Loopback.Exit-*

- A 2mS timer is started on entry into the Loopback.Exit substate
- LFPS transmitter and receiver remain enabled
- Master and slave complete Loopback.LFPS exit signaling

*Exit from Loopback.Exit-*

As depicted in Figure 27-4 on page 646, there are three possible exits from the Loopback.Exit substate

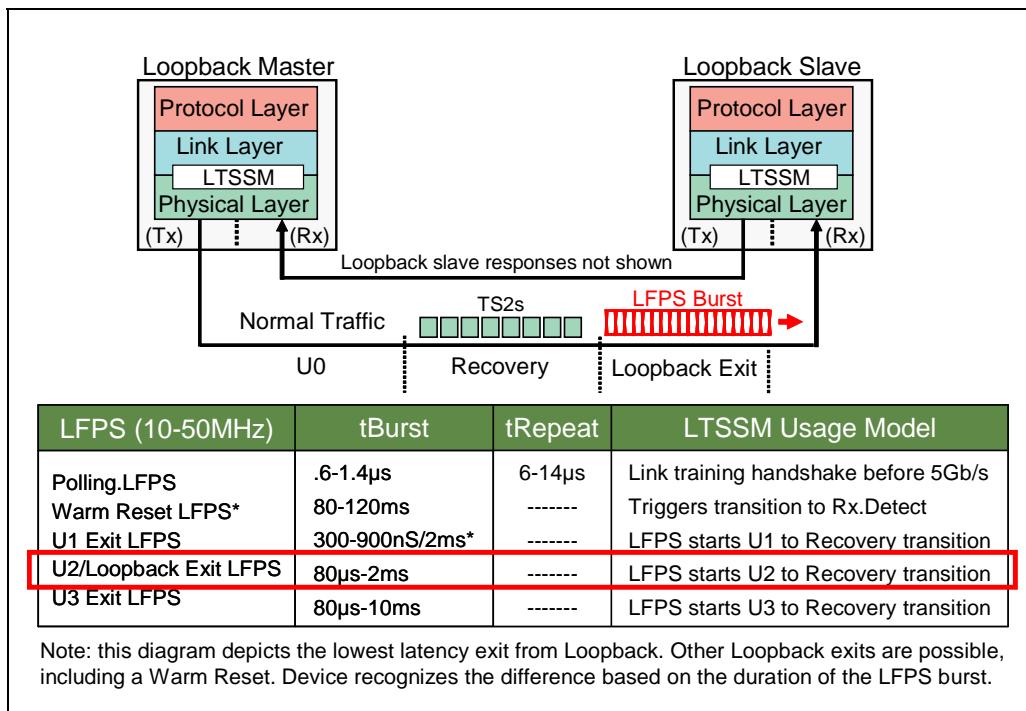
- Ports transition to Rx.Detect if Loopback.LFPS exit handshake is successful
- Ports transition to SS.Inactive if the 2mS timeout expires and Loopback.LFPS exit handshake is not yet successful.
- Downstream facing ports transition to SS.Disabled when directed.
- Downstream facing ports transition to Rx.Detect state when directed to issue a Warm Reset on the link.
- Upstream facing ports transition to Rx.Detect state when the Warm Reset is detected.

# Chapter 27: Receiver Loopback Testing

## A Note About Differences In Loopback Exit Latency

As just described, there are several possible exits from Loopback Mode once the testing is complete. In some types of validation and debug testing, Loopback is entered and exited frequently and the transition times are important. As depicted in Figure 27-6 on page 649, the lowest latency exit is through Recovery; a return to link state U0 and normal operations can be accomplished within a milliseconds, or so. While a Warm Reset will also cause an exit from Loopback, but it requires 80-120 ms of LFPS signaling followed by a transition to Rx.Detect and full link training.

Figure 27-6: Loopback Exit Through Recovery



## **USB 3.0 Technology**

---

---

### **Loopback May Also Be Used In Compliance Testing**

In the standard BERT loopback configuration, the master generates a fixed test pattern (BDAT) consisting of scrambled and encoded NOPs which are checked after retransmission by the slave. If the slave BERT error counting option is supported, slave checks for inbound BDAT errors and returns the accumulated error count when requested by the loopback master.

The loopback capability may also be used in electrical compliance testing. In this case, a compliance tester puts a slave in loopback mode and then may generate and check a variety of other test patterns while adjusting Tx amplitude, jitter, de-emphasis, and other variables.

# **Index**

---

---

Numerics  
8b/10b Decoding 379, 407, 408  
8b/10b Encoding 375, 378, 388, 389, 390, 391, 392, 393, 601

A  
AC Coupling Capacitor 597  
ACK Header 83, 84, 85, 86, 108, 115, 129, 193, 199  
Alternate Settings 514  
Application Layer 57  
Asynchronous Messages 92  
Asynchronous Notifications 168, 247

B  
BCNT Ordered Set 242  
BELT 589  
BER 240, 249, 283, 340, 594, 639  
BERC Ordered Set 241  
BERT 641, 642  
BERT Data Test Pattern (BDAT) 641  
BERT Ordered Sets 240  
BERT Reset 643  
BH Port Reset 424, 492, 530, 559, 562  
Bit Error Rate Test (BERT) 642  
BOS Descriptor 506, 541  
Broadcast Transactions 26  
BRST Ordered Set 241  
Bulk Endpoint 32, 147  
Bulk Endpoint Streaming 74, 150  
Bulk IN SuperSpeed Transactions 127  
Bulk OUT SuperSpeed Transactions 137  
Bulk OUT Transfers 141  
Bulk Streaming 14, 74  
Bulk Streaming Endpoints 146  
Bulk Transfers 125  
Bus Interval 95, 103, 183  
Bus Interval Adjustment Message 188  
Bus Interval Adjustment Message Header 95  
Bus-powered Devices 415

# **USB 3.0 Technology**

---

## C

Change Indicators 491  
Chip-to-Chip Protocol 66, 373, 375  
Class Driver Initialization 522  
Class-Specific Descriptor 28  
Clear Change Indicator request 492  
Clear Hub Feature request 560  
Clear Hub Local Power Feature Request 560  
Clear Hub/Port Requests 535  
Clear Port Connection Request 496  
Clear Port Request 534  
Clock And Data Recovery 397  
Clock Compensation Ordered Set 239  
Clock Recovery 432  
Clock/Data Recovery 378  
Companion Controllers 19  
Compliance Mode 281, 635  
Compliance Mode Test Patterns 636  
Compliance Testing 617  
Configuration Descriptor 28, 510, 545  
configuration value 512, 546  
Container ID - BOS 509, 544  
Control (K) Symbols 231  
Control Endpoints 30  
Control Transfer Stages 36  
Control Transfers - SuperSpeed 105  
CRC-16 (Header) 308  
CRC-16 Checked (Header) 315  
CRC-32 Check (DPP) 319  
CRC-32 Generation (DPP) 310  
CRC-5 (Link Control Word) 307  
CRC-5 Checked (Link Control Word) 316  
CRD (Current Running Disparity) 391  
Credit HP Timer 326  
Current Running Disparity 391

## **Index**

---

---

D

- Data (D) Symbols 231
- Data Bursting 14, 73
- DATA Header Packet 110
- DATA OUT Bursts 138
- Data Packet 99
  - Data Packet Framing 233, 234
  - DATA Packet Header 99, 198, 233, 322
  - Data Packet Header (DPH) 310
  - DATA Packet Payload 11, 63, 76, 80, 99, 102, 111, 310
  - Data Payload - Repeater Model 215
  - Data Recovery Circuit 432
  - Data Scrambling 386
  - DATA Stage 36, 106, 114
  - DC Differential Impedance 597
  - De-scrambling 379
  - Detecting Receiver Termination 611
  - Device Configuration 485
  - Device Configuration Process 498
  - Device Descriptor 28, 502, 503, 539
  - Device Descriptor Definition 540
  - Device Detection 488, 527
  - Device Detection Process 528
  - Device Initiated Wakeup 581
  - Device Suspend - U3 578
  - Device-Specific Timers 574
  - Differential Receiver 396
  - Differential Transmitter 395
  - Differential Tx p-p voltage 597
  - Disabling Scrambling 388
  - DPP CRC-32 Checking 320
  - DWord 5

# **USB 3.0 Technology**

---

E

- EHCI 19
  - Elastic Buffer 379, 401
  - Elastic Buffer Architecture 406
  - Elastic Buffer Initialization 433
  - Endpoint Characteristics 76
  - Endpoint Companion Descriptor 518, 550
  - Endpoint Descriptor 28, 516
  - Endpoints 29
  - End-to-End Flow Control 141
  - End-to-End Protocol 11, 61, 71
  - EOB IN Flow Control 133
  - Equalization 52, 396, 431
  - ERDY Header 88
  - Errors, DATA IN Transfers 136
  - Errors, DATA OUT Transfers 144
  - Errors, Interrupt IN Transactions 173
  - Errors, Interrupt OUT Transactions 180
  - ESD Protection 615
  - Exit Latencies 586
  - Exiting U1-U3 471
  - eXtensible Host Controller 8
  - External Hubs 23
- F
- Flow Control 321
  - Flow Control - Normal Operations 331
  - Flow Control Credits 334
  - Flow Control Elements 325
  - Flow Control Initialization 329
  - Flow Control Link Commands 229
  - Flow Control Logic Initialization 330
  - Framing 312
  - Framing Ordered Sets 232
  - Function Power Management 69, 566
  - Function Suspend 584
  - Function Wake Header 92

# **Index**

---

---

## G

Ganged Power Switching 554  
Get Descriptor 501, 537  
Get Descriptor Request 536, 538  
Get Descriptor Requests 501  
Get Device Descriptor 539  
Get Hub Descriptor request 551, 557  
Get Hub Status request 560  
Get Port Status Request 492, 495, 533, 560  
Getting Descriptors 536  
Getting the Descriptors 499

## H

Hdr Seq# 306  
Hdr Seq# Check 317  
Header Buffers - Store and Forward Model 214  
Header CRC Checks 359  
Header CRC-16 Generation 308  
Header Fields 223  
Header Packet Flow Control 230  
Header Packet Framing 232  
Header Processing 223  
High Impedance Reflections 615  
Host Initiated Wakeup 581  
Hot Reset 278, 418  
Hot Reset Signaling 419  
Hot Reset state 420  
HP Sequence Number 356  
HS Split Transaction Protocol 39  
Hub Attachment 207  
Hub Buffer Requirements 213  
Hub Class Descriptor 551, 552  
Hub Configuration 523  
Hub Depth Limits 25  
Hub Descriptor 536  
Hub Driver Initialization 551  
Hub Error Detection and Handling 218

## **USB 3.0 Technology**

---

Hub Feature Selectors 561  
Hub Inactivity Timers 570  
Hub Interface Descriptor 547  
Hub Interface Layers 58  
Hub Link Power Management 219  
Hub Over-Current Protection 555  
Hub Packet Forwarding 207  
Hub Port States 561  
Hub Port Status 529, 560  
Hub Status Change Endpoint 528  
Hub Status Change Indicator 489  
Hub Status Field 560  
Hub-Specific Requests 555  
Hybrid-Powered Device 547  
**I**  
IN Burst 134  
IN Data Bursting 127  
IN SuperSpeed Transaction Protocol 127  
IN Transactions 34  
Interface Association Descriptor 512  
Interface Descriptor 28, 513  
Interface Number 514  
Interrupt Burst Transactions 170  
Interrupt Endpoint 33  
Interrupt IN SuperSpeed Transaction Protocol 169  
Interrupt OUT Data Bursting 176  
Interrupt OUT Transfers 177  
Interrupt Transfers - SuperSpeed 167  
Inter-Symbol Interference (ISI) 600  
Introduction to Isochronous Transfers 181  
ISI With De-emphasis 603  
Isochronous Endpoint 33  
Isochronous End-to-End Protocol 192  
Isochronous IN Protocol 192  
Isochronous IN Protocol Rules 192  
Isochronous IN with Burst 193

## **Index**

---

---

Isochronous OUT Burst 197  
Isochronous OUT Transactions 196  
Isochronous OUT with Burst 196  
Isochronous Service Intervals 183  
Isochronous Timestamp Packet 80, 103, 185  
ITP Isochronous Timestamp Packet 80  
**J**  
Jitter Budget Allocations 594  
Jitter Budgeting 593  
**L**  
Latency Tolerance Message 588  
Latency Tolerance Message Header 94, 589  
LAU 298  
LAU (Link Accept USB Power State) 297  
LBAD (Link Bad) 294  
LBAD Link Command 365  
LCRD\_x 295, 327, 482  
LCRD\_x (Link Credit) 295, 328  
LCRD\_x Advertisement 457  
LCRD\_x Generation 327  
LCRD\_X Link Command 328  
LDN (Link Up/Link Down) 299  
LFPS 612  
LFPS (Low Frequency Periodic Signaling) 381  
LFPS Signaling 416, 613  
LFPS Signaling Events 251, 613  
LFPS\_Ux Exit burst 472  
LFSR 387  
LGO\_Ux (Link Go to USB Power State) 296  
LGOOD\_n 352, 371  
LGOOD\_n (Link Good) 292  
LGOOD\_n Advertisement 455, 480  
Link Bad (LBAD) 293  
Link Command Framing 235  
Link Command Groups 290  
Link Command Word 291

## **USB 3.0 Technology**

---

Link Commands 228, 289, 291  
Link Control Word 81, 85, 307, 308, 314, 316  
Link Down (LDN) 300  
Link Error Count (LEC) 420  
Link Good n (LGOOD\_n) 292  
Link Layer 57  
Link Layer Packet Processing 303  
Link Layer Packet Processing - Tx 305  
Link Layer Receiver Packet Processing 313  
Link Layer Transmitter Packet Processing 305  
Link Level Flow Control 323  
Link Power Management 68, 564  
Link Power State Conditions 567  
Link Power State Transitions 567  
Link Power States 566  
Link Training 428  
Link Training Elements 431  
Link Training Sequence 434, 435  
Link Up (LUP) 301  
Link Up/Link Down Link Commands 229  
Link-Power Hierarchy 568  
LMP - Inactivity Timeout 574  
Long Channel 429, 633  
Loopback 283, 639  
Loopback BERT Ordered Sets 642  
Loopback Bit Error Rate Test 240  
Loopback Test Pattern 641  
Low Frequency Periodic Signaling 605  
Low Frequency Periodic Signaling (LFPS) 381  
Low Power Differential Tx p-p voltage 597  
LPMA (Link Power Management Ack) 299  
LRTY 295  
LRTY (Link Retry) 294  
LRTY Link Command 354  
LTM BELT 589  
LTM\_ENABLE 522

# **Index**

---

---

LTSSM 245, 281  
LTSSM Functional Block 228  
LTSSM State Transition Time-outs 254  
LTSSM State Transitions 250  
LTSSM states 468  
LTSSM Time-outs 254  
LUP 301, 302  
LXU (Link Rejects USB Power State) 298  
**M**  
Maximum Payload Sizes - USB 2.0 37  
Maximum slew rate 597  
Modulation Rate 604  
**N**  
Notification Headers 92  
Notifications 82  
NRDY Header 86  
NRDY IN Flow Control 132, 134  
NRDY OUT Flow Control 142  
NumP=0 OUT Flow Control 143  
**O**  
OHCI 19  
Ordered Sets 231  
OUT Data Bursting 138  
OUT SuperSpeed Transaction Protocol 137  
OUT Transactions 35  
Over-Current Protection 555  
**P**  
Packet Acknowledgement Link Commands 229  
Packet Forwarding 213  
Packet Framing 312  
Packet Routing Across Hubs 209  
Packet Routing to Hub Controller 210  
Packet Routing to Hub Controller with LMPs 210  
Path Exit Latency (PEL) 586  
PCI Configuration Registers 22  
PEL 587

## **USB 3.0 Technology**

---

Pending HP Timer 347  
Physical Layer 58, 377  
Physical Layer Logic 383, 430  
Ping and Ping Response 189  
PING and PING RESPONSE Headers 96  
Ping Maximum Exit Latency 190  
Ping Timing Parameters 190  
Ping.LFPS 614  
Polarity Inversion 400, 401, 433  
Polling.Active 450  
Polling.Configuration 452  
Polling.Idle 454  
Polling.LFPS 443, 614  
Polling.RxEQ 445  
Port Capability LMP 459  
Port Capability LMP Header 460  
Port Change Indicator Definition 493  
Port Change Indicators 492, 529  
Port Config Response LMP 464  
Port Configuration LMP 462  
Port Link State 558  
Port Reset 559  
Port Status 491  
Port Status Definition 494, 532  
Port U1/U2 Timeout 557  
PORT\_RESET 424  
Port-to-Port Protocol 11, 14, 64, 76, 221, 223  
Power Management 67  
Power Management Link Commands 229  
Power Management Transitions 253  
PowerOn Reset 413, 415  
Priming the Bulk Streaming Endpoints 149  
Protocol Layer 57, 61  
Protocol Layer Overview 80  
Protocol Packet Types 72

# **Index**

---

---

## R

Receive Header Packet Checks 226  
Receiver DC common mode impedance 606  
Receiver Equalization 431  
Receiver Error Checks 349  
Receiver Packet Processing 314  
Recovery 274  
Recovery State 400, 469  
Recovery Substate Events 474  
Recovery.Active 474  
Recovery.Configuration 478  
Recovery.Idle 479  
Remote Rx HP Credits 326  
Remote Wakeup 583  
Repeater Model 215  
Retries (Headers) 349  
Root Hubs 22  
Routing String 212  
Rx DC Differential Impedance 606  
Rx Equalizer Training 608  
Rx PHY Logic 396  
Rx.Detect 416, 434, 436  
Rx.Detect State 265, 435  
Rx.Detect.Active 437  
Rx.Detect.Quiet 442  
Rx.Detect.Reset 435  
S  
Scrambler Polynomial 387  
Scrambling 378  
SEL (System Exit Latency) 522  
SEL request 588  
Serial To Parallel Conversion 433  
Set Address request 536  
Set Configuration request 546  
Set Hub Depth 555  
Set Isochronous Delay 522

## **USB 3.0 Technology**

---

Set Port Power 558  
Set Port Power Feature request 554, 558  
Set Port Remote Wake Mask 557  
SetAddress request 498  
Setting Device Configuration 521  
Setup Stage 36, 106  
Setup Transaction 110  
Short Channel 429, 634  
Short Circuit Tolerance 616  
Short Packets 135  
Skip Ordered Set 239  
SKP Ordered Sets 402, 404, 406, 407  
Smart Isochronous OUT Transactions 202  
Smart Isochronous Transaction Scheduling 198  
Software Role in LPM 569  
Spread Spectrum Clocking 403, 603  
Spread-Spectrum Clocking (SSC) 394  
SS Link Error Correction 343  
SS USB Device Capability - BOS 508  
SSC 394, 603  
SSC Modulation 604  
STALL Header 91  
Standard Bulk Endpoints 151  
Standard Descriptors 499, 536  
Standard Device Requests 486, 524  
Standard Requests 31  
Status Change Endpoint Descriptor 548  
Status Change Indicator 489  
STATUS Header 89  
Status Stage 36, 106  
Status Stage - SuperSpeed 115  
Store and Forward Model 214  
Stream ID 148  
Stream IN Values 149  
Streaming Bulk Endpoints 154  
String Descriptor 28

## **Index**

---

---

String Descriptors 519, 550  
SuperSpeed Bulk Endpoints 125  
SuperSpeed Descriptors 67, 501, 538  
SuperSpeed End To End Flow Control 322  
SuperSpeed Flow Control 322  
SuperSpeed Hubs 205  
SuperSpeed Layered Interface 56  
SuperSpeed Link Level Flow Control 324  
SuperSpeed Receiver Termination 609  
SuperSpeed Suspend 579  
SuperSpeed Transaction Timeout Values 146  
SuperSpeed Voltage Levels 595  
Suspend/Resume 566  
Symbol Lock 433  
System Exit Latency 588  
System Exit Latency (SEL) 587  
System Exit Latency Calculation 587  
T  
The Physical Layer 379  
Three-Stage Control Transfers 107  
Three-Stage Transfers 36  
Time Base Changes 187  
Timestamp Packet 186  
Token/Data/Handshake 11, 62  
Token-Data-HandShake 34  
tPingTimeout 190  
Training Sequence Equalization 608  
Training Sequence Errors 343  
Transaction Deferral 216  
Transaction Packet Sub Types 82  
Transaction Packets 82  
Transmit Header Packet Processing 225  
Transmitter De-emphasis 600  
Transmitter Error Checks 346  
Transmitter Low Power Option 599  
tReset 614

## **USB 3.0 Technology**

---

TS1 Ordered Set 237, 450  
TS1 Ordered State 252  
TS2 Ordered Set 238, 252, 453  
TSEQ 608  
TSEQ Ordered Set 236, 252, 446  
Two-Stage Control Transfer 107, 116  
Two-Stage Transfers 36  
Tx De-emphasis level 597  
Tx DPP CRC-32 Generation 311  
Tx PHY Logic 385  
U  
U0 State 256, 566  
U1 Exit 472, 614  
U1 Inactivity Timeouts 571  
U1 State 259, 566  
U1 to U2 Silent Transition 573  
U1/U2/U3 Exit to U0 580  
U1\_ENABLE 522  
U2 Exit 472  
U2 Inactivity Timeouts 572  
U2 State 566  
U2/ Loopback Exit 614  
U2\_ENABLE 522  
U3 Exit 472  
U3 State 566  
U3 State (Suspend) 263  
U3 Wakeup 581, 614  
UAS (USB Attached SCSI) 154  
UAS Commands - Non Data 158  
UAS Commands - Read DMA 160  
UAS Commands - Write DMA 164  
UAS Device Endpoints 155  
UASP (USB Attached SCSI with SuperSpeed Protocols) 154  
UASP Commands - Read DMA 162  
UASP Commands - Write DMA 165  
UHCI 19

## **Index**

---

---

Unicast Routing 212  
Unit Interval 597, 606  
Upstream Asynchronous Message Errors 341  
USB 2.0 Error Handling 339  
USB 2.0 Extension - BOS 507, 542  
USB 2.0 Flow Control 322  
USB 2.0/3.0 Compatibility 50  
USB 3.0 Power-B Connections 54  
USB 3.0 Topology 48  
USB Attached SCSI 154  
USB Attached SCSI with SuperSpeed Protocols 154  
USB SuperSpeed Error Handling 340  
V  
VBUS 414  
W  
Wake Notification Message 583  
Warm Reset 416, 417, 435, 605  
X  
xHCI 8



# USB 3.0 Technology A Comprehensive Guide to SuperSpeed USB



## LIVE COURSES:

- Comprehensive PCI Express
- PCI Express IO Virtualization
- Comprehensive NVM Express



## eLEARNING COURSES:

- Comprehensive PCI Express
- Fundamentals of PCI Express
- Comprehensive NVM Express

**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.