Universal Serial Bus 3.0 Specification

HP

Intel Corporation

Microsoft Corporation

NEC Corporation

NXP Semiconductors

Texas Instruments Incorporated

Revision 0.9

July 30, 2008

Revision History

Revision	Comments	Issue Date
0.7	First consolidated review draft.	October 19, 2007
0.75	Updates to Chapters 6, 7, 8, 10, and 11.	November 9, 2007
0.78	Added Chapters 1, 3, and 12 and Appendixes A, B, and C. Updated Chapters 4, 5, 6, 7, 8, 10, and 11.	December 26, 2007
0.85	Moved content into Chapter 2 and Appendix C. Updated all other chapters except Appendixes A and B.	April 4, 2008
0.9RC1	Updated all chapters except Chapter 1 and Appendixes A and B.	June 19, 2008
0.9	Incorporated all approved SCRs	July 30, 2008

INTELLECTUAL PROPERTY DISCLAIMER

THIS SPECIFICATION IS PROVIDED TO YOU "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE. THE AUTHORS OF THIS SPECIFICATION DISCLAIM ALL LIABILITY, INCLUDING LIABILITY FOR INFRINGEMENT OF ANY PROPRIETARY RIGHTS, RELATING TO USE OR IMPLEMENTATION OF INFORMATION IN THIS SPECIFICATION. THE PROVISION OF THIS SPECIFICATION TO YOU DOES NOT PROVIDE YOU WITH ANY LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS.

NOTE: THIS IS A DRAFT SPECIFICATION. The authors of this draft specification may make changes to it at any time, without notice. Product designers should not rely on the specification, and the authors shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes.

Please send comments to *usb3feedback@usb.org*

For industry information, refer to the USB Implementers Forum web page at http://www.usb.org/usb30

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

Copyright © 2007-2008, HP, Intel Corporation, Microsoft Corporation, NEC Corporation, NXP Semiconductors, and Texas Instruments Incorporated.

All rights reserved.

Acknowledgement of USB 3.0 Technical Contribution

Dedication

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

The authors of this specification would like to recognize the following people who participated in the USB 3.0 Bus Specification technical workgroups. We would also like to acknowledge the many others throughout the industry who provided feedback and contributed to the development of this specification.

Promoter Company Employees

Alan Berkema	Hewlett Packard
Anthony Hudson	Hewlett Packard
Arthur Zarnowitz	Hewlett Packard
Huimin Chen	Intel Corporation
Bob Dunstan	Intel Corporation
Dan Froelich	Intel Corporation
Brad Hosler	Intel Corporation
John Howard	Intel Corporation
Rahman Ismail	Intel Corporation
Yun Ling	Intel Corporation
Andy Martwick	Intel Corporation
Steve McGowan	Intel Corporation
Duane Quiet	Intel Corporation
Brad Saunders	Intel Corporation
Micah Sheller	Intel Corporation
Clint Walker	Intel Corporation
Jim Walsh	Intel Corporation
Kai Wang	Intel Corporation
Randy Aull	Microsoft Corporation
Nathan Sherman	Microsoft Corporation
David Wooten	Microsoft Corporation
Nobuyuki Mizukoshi	NEC Corporation
Hajime Nozaki	NEC Corporation
Kenji Oguma	NEC Corporation
Hock Seow	NEC Corporation
"Peter" Chu Tin Teng	NEC Corporation

Socol Constantin	NXP Semiconductors
Linus Kerk	NXP Semiconductors
Martin Klein	NXP Semiconductors
Geert Knapen	NXP Semiconductors
Bart Vertenten	NXP Semiconductors
Julie Nirchi	Texas Instruments
Mitsuru Shimada	Texas Instruments
Sue Vining	Texas Instruments

Contributor Company Employees

Glen Chandler	Acon
John Chen	Acon
Roger Hou	Acon
Charles Wang	Acon
Norman Wu	Acon
Steven Yang	Acon
George Yee	Acon
George Olear	Contech Research
Sophia Liu	Electronics Testing Center, Taiwan (ETC)
William Northey	FCI
Tom Sultzer	FCI
Garry Biddle	Foxconn
Kuan-Yu Chen	Foxconn
Jason Chou	Foxconn
Gustavo Duenas	Foxconn
Bob Hall	Foxconn
Jiayong He	Foxconn
Jim Koser	Foxconn
Joe Ortega	Foxconn
Ash Raheja	Foxconn
James Sabo	Foxconn
Pei Tsao	Foxconn
Kevin Walker	Foxconn
Tsuneki Watanabe	Foxconn
Chong Yi	Foxconn
Taro Hishinuma	Hirose Electric
Kaz Ichikawa	Hirose Electric
Ryozo Koyama	Hirose Electric
Karl Kwiat	Hirose Electric
Tadashi Sakaizawa	Hirose Electric
Shinya Tono	Hirose Electric
Eiji Wakatsuki	Hirose Electric

Takashi Ehara	Japan Aviation Electronics Industry Ltd. (JAE)
Ron Muir	Japan Aviation Electronics Industry Ltd. (JAE)
Kazuhiro Saito	Japan Aviation Electronics Industry Ltd. (JAE)
Hitoshi Kawamura	Mitsumi
Takashi Kawasaki	Mitsumi
Atsushi Nishio	Mitsumi
Yasuhiko Shinohara	Mitsumi
Tom Lu	Molex Inc.
Edmund Poh	Molex Inc.
Scott Sommers	Molex Inc.
Jason Squire	Molex Inc.
Dat Ba Nguyen	NTS/National Technical System
Jan Fahllund	Nokia
Richard Petrie	Nokia
Panu Ylihaavisto	Nokia
Alvin Cox	Seagate
Martin Furuhjelm	Seagate
Julian Gorfajn	Seagate Technology LLC
Marc Hildebrant	Seagate Technology LLC
Tony Priborsky	Seagate Technology LLC
Saleem Mohammad	Synopsys, Inc.
Daniel Weinlader	Synopsys, Inc.
Thomas Grzysiewicz	Tyco Electronics
Masaaki Iwasaki	Tyco Electronics
Kazukiyo Osada	Tyco Electronics
Hiroshi Shirai	Tyco Electronics
Scott Shuey	Tyco Electronics
Masaru Ueno	Tyco Electronics

Contents

Acknowledgement of USB 3.0 Technical Contribution

1 Introduction

1.1 Motivation 1-1
1.2 Objective of the Specification 1-2
1.3 Scope of the Document 1-2
1.4 USB Product Compliance 1-2
1.5 Document Organization 1-3
1.6 Design Goals 1-3
1.7 Related Documents 1-3

2 Terms and Abbreviations

3 SuperSpeed USB Architectural Overview

3.1 USB 3.0 Overview 3-1
3.1.1 SuperSpeed Architecture Overview 3-2
3.1.1.1 Physical Layer 3-2
3.1.1.2 Link Layer 3-3
3.1.1.3 Protocol Layer 3-3
3.1.1.4 Hubs 3-4
3.1.1.5 Power Management 3-5
3.2 USB 3.0 System 3-5
3.2.1 Comparing SuperSpeed USB to USB 2.0 3-5
3.2.2 System Level Topology 3-7
3.2.2.1 Hosts 3-7
3.2.2.2 Hubs 3-8
3.2.2.3 Devices 3-8
3.2.3 Bus Protocol 3-9
3.2.4 Robustness 3-10
3.2.4.1 Error Detection 3-10
3.2.4.2 Error Handling 3-10
3.3 USB Specification Chapter Overview 3-11
3.3.1 Mechanical 3-12
3.3.2 Physical Layer 3-14
3.3.3 Link Layer 3-15
3.3.4 Protocol Layer 3-15
3.3.5 Framework Layer 3-16
3.3.6 Hubs 3-16
3.3.6.1 Hub Architecture 3-17
3.3.6.2 Hub Repeater/Forwarder Architecture 3-18
3.3.6.3 Hubs and Transfers to Power Managed Links 3-18
3.3.7 Performance and Power Efficiency 3-18

3.3.8 SuperSpeed Power Management....3-19
3.3.8.1 Link Power Management....3-19
3.3.8.2 Function and Device Power Management....3-20
3.3.8.3 Platform PM Support....3-21

4. SuperSpeed Data Flow Model

4.1 Implementer Viewpoints....4-1
4.2 SuperSpeed USB Communication Flow....4-1
4.2.1 Pipes....4-2
4.3 SuperSpeed USB Protocol Overview....4-2
4.3.1 Differences with USB 2.0....4-2
4.3.1.1 Comparing USB 2.0 and SuperSpeed USB Transactions....4-2
4.3.1.2 Comparing USB 2.0 and SuperSpeed USB Bus Topology....4-3
4.3.1.3 Introduction to SuperSpeed USB Packets....4-4
4.4 Generalized Transfer Description....4-4
4.4.1 Data Bursting....4-5
4.4.2 IN Transfers....4-6
4.4.3 OUT Transfers....4-7
4.4.4 Power Management and Performance....4-7
4.4.5 Control Transfers....4-8
4.4.5.1 Control Transfer Packet Size....4-8
4.4.5.2 Control Transfer Bandwidth Requirements....4-9
4.4.5.3 Control Transfer Data Sequences....4-9
4.4.6 Bulk Transfers....4-9
4.4.6.1 Bulk Transfer Packet Size....4-10
4.4.6.2 Bulk Transfer Bandwidth Requirements....4-10
4.4.6.3 Bulk Transfer Data Sequences....4-11
4.4.6.4 Bulk Streams....4-11
4.4.7 Interrupt Transfers....4-13
4.4.7.1 Interrupt Transfer Packet Size....4-14
4.4.7.2 Interrupt Transfer Bandwidth Requirements....4-14
4.4.7.3 Interrupt Transfer Data Sequences....4-15
4.4.8 Isochronous Transfers....4-15
4.4.8.1 Isochronous Transfer Packet Size....4-16
4.4.8.2 Isochronous Transfer Bandwidth Requirements....4-16
4.4.8.3 Isochronous Transfer Data Sequences....4-17
4.4.9 Device Notifications....4-17
4.4.10 Reliability....4-18
4.4.10.1 Physical Layer....4-18
4.4.10.2 Link Layer....4-18
4.4.10.3 Protocol Layer....4-18
4.4.11 Efficiency....4-18

5 Mechanical

5.1 Objective....5-1
5.2 Significant Features....5-1
5.2.1 Connectors....5-1
5.2.1.1 USB 3.0 Standard-A Connector....5-2

5.2.1.2 USB 3.0 Standard-B Connector 5-2
5.2.1.3 USB 3.0 Power-B Connector 5-2
5.2.1.4 USB 3.0 Micro-B Connector 5-2
5.2.1.5 USB 3.0 Micro-AB and USB 3.0 Micro-A Connectors 5-3
5.2.2 Compliant Cable Assemblies 5-3
5.2.3 Raw Cables 5-3
5.3 Connector Mating Interfaces 5-4
5.3.1 USB 3.0 Standard-A Connector 5-4
5.3.1.1 Interface Definition 5-4
5.3.1.2 Pin Assignments and Description 5-14
5.3.1.3 USB 3.0 Standard-A Connector Color Code 5-14
5.3.2 USB 3.0 Standard-B Connector 5-14
5.3.2.1 Interface Definition 5-14
5.3.2.2 Pin Assignments and Description 5-19
5.3.3 USB 3.0 Power-B Connector 5-19
5.3.3.1 Interface Definition 5-19
5.3.3.2 Pin Assignments and Descriptions 5-24
5.3.4 USB 3.0 Micro-B Connector Family 5-24
5.3.4.1 Interfaces Definition 5-24
5.3.4.2 Pin Assignments and Description 5-32
5.4 Cable Construction and Wire Assignments 5-34
5.4.1 Cable Construction 5-34
5.4.2 Wire Assignments 5-35
5.4.3 Wire Gauges and Cable Diameters 5-35
5.5 Cable Assemblies 5-36
5.5.1 USB 3.0 Standard-A to USB 3.0 Standard-B Cable Assembly 5-36
5.5.2 USB 3.0 Standard-A to USB 3.0 Micro-B Cable Assembly 5-38
5.5.3 USB 3.0 Micro-A to USB 3.0 Micro-B Cable Assembly 5-40
5.5.4 USB 3.0 Micro-A to USB 3.0 Standard-B Cable Assembly 5-42
5.5.5 Captive Cables 5-42
5.5.6 Cable Assembly Length 5-42
5.6 Electrical Requirements 5-43
5.6.1 High Speed Electrical Requirements 5-43
5.6.1.1 Raw Cable 5-43
5.6.1.2 Mated Connector 5-44
5.6.1.3 Mated Cable Assemblies 5-45
5.6.2 DC Electrical Requirements 5-50
5.6.2.1 Low Level Contact Resistance (EIA 364-23B) 5-50
5.6.2.2 Dielectric Strength (EIA 364-20, Method 301) 5-50
5.6.2.3 Insulation Resistance (EIA 364-21, Method 302) 5-51
5.6.2.4 Contact Current Rating (EIA 364- 70, Method 2) 5-51
5.7 Mechanical and Environmental Requirements 5-51
5.7.1 Mechanical Requirements 5-51
5.7.1.1 Insertion Force (EIA 364-13) 5-51
5.7.1.2 Extraction Force (EIA 364-13) 5-51
5.7.1.3 Durability or Insertion/Extraction Cycles (EIA 364-09) 5-51
5.7.1.4 Cable Flexing (EIA 364-41, Condition I) 5-52

5.7.1.5 Cable Pull-Out (EIA 364-38, Condition A) 5-52
5.7.1.6 Peel Strength (Micro-B Connector Family Only) 5-52
5.7.1.7 4-Axes Continuity Test (USB 3.0 Micro-B Connector Family Only) 5-52
5.7.1.8 Wrenching Strength (Reference, Micro-B Connector Family Only) 5-54
5.7.1.9 Lead Co-Planarity 5-54
5.7.1.10 Solderability 5-54
5.7.1.11 RoHS Compliance 5-54
5.7.2 Environmental Requirements 5-54
5.7.3 Materials 5-55
5.8 Implementation Notes and Design Guides 5-55
5.8.1 Mated Connector Dimensions 5-55
5.8.2 EMI Management 5-58
5.8.3 Stacked Connectors 5-58
5.8.4 Captive Cables 5-58

6 Physical Layer

6.1 Physical Layer Overview 6-1
6.2 Physical Layer Functions 6-1
6.2.1 Measurement Overview 6-4
6.2.2 Channel Overview 6-4
6.3 Symbol Encoding 6-5
6.3.1 Serialization and De-serialization of Data 6-5
6.3.2 Normative 8b/10b Decode Rules 6-6
6.3.3 Data Scrambling 6-6
6.3.4 8b/10b Decode Errors 6-7
6.3.5 Special Symbols for Framing and Link Management 6-8
6.4 Link Initialization and Training 6-8
6.4.1 Normative Training Sequence Rules 6-8
6.4.1.1 Training Control Bits 6-9
6.4.1.2 Training Sequence Values 6-9
6.4.2 Lane Polarity Inversion 6-10
6.4.3 Elasticity Buffer and SKP Ordered Set 6-10
6.4.4 Compliance Pattern 6-11
6.5 Clock and Jitter 6-12
6.5.1 Informative Jitter Budgeting 6-12
6.5.2 Normative Clock Recovery Function 6-12
6.5.3 Normative Spread Spectrum Clocking (SSC) 6-14
6.5.4 Normative Slew Rate Limit 6-15
6.6 Signaling 6-15
6.6.1 Eye Diagrams 6-15
6.6.2 Voltage Level Definitions 6-16
6.6.3 Tx and Rx Input Parasitics 6-17
6.7 Transmitter Specifications 6-18
6.7.1 Transmitter Electrical Parameters 6-18
6.7.2 Transmitter Eye 6-19
6.7.3 Tx Compliance Reference Receiver Equalize Function 6-20
6.7.4 Informative Transmitter De-emphasis 6-21

6.7.5 Entry into Electrical Idle, U1 6-21
6.8 Receiver Specifications 6-22
6.8.1 Receiver Equalization Training 6-22
6.8.2 Informative Receiver CTLE Function 6-22
6.8.3 Receiver Electrical Parameters 6-23
6.8.4 Receiver Loopback 6-24
6.7.4.1 Loopback BERT 6-24
6.8.5 Normative Receiver Tolerance Compliance Test 6-25
6.9 Low Frequency Periodic Signaling (LFPS) 6-27
6.9.1 LFPS Signal Definition 6-27
6.9.2 Example LFPS Handshake for U1/U2 Exit, Loopback Exit, and U3 Wakeup 6-28
6.9.3 Example Warm Reset 6-30
6.10 Transmitter and Receiver DC Specifications 6-30
6.10.1 Informative ESD Protection 6-30
6.10.2 Informative Short Circuit Requirements 6-30
6.10.3 Normative High Impedance Reflections 6-31
6.11 Receiver Detection 6-31
6.11.1 Rx Detect Overview 6-31
6.11.2 Rx Detect Sequence 6-32
6.11.3 Upper Limit on Channel Capacitance 6-32

7 Link Layer

7.1 Link Management and Flow Control 7-2
7.1.1 Packets and Packet Framing 7-2
7.1.1.1 Header Packet Structure 7-2
7.1.1.2 Data Packet Payload Structure 7-6
7.1.2 Link Commands 7-9
7.1.2.1 Link Command Structure 7-9
7.1.2.2 Link Command Word Definition 7-9
7.1.2.3 Link Command Placement 7-12
7.1.3 Logical Idle 7-13
7.1.4 Link Command Usage for Flow Control, Error Recovery and Power Management 7-13
7.1.4.1 Header Packet Flow Control and Error Recovery 7-14
7.1.4.2 Link Power Management and Flow 7-20
7.2 Link Error Rules/Recovery 7-24
7.2.1 Overview of SuperSpeed Bit Errors 7-24
7.2.2 Link Error Types, Detection and Recovery 7-24
7.2.3 Packet Errors 7-24
7.2.3.1 Packet Framing Detection 7-24
7.2.3.2 Header Packet CRC or Length Error 7-25
7.2.3.3 Rx Header Sequence Number Error 7-25
7.2.3.4 ACK Tx Header Sequence Number Error 7-26
7.2.3.5 Header Sequence Number Advertisement Error 7-26
7.2.4 Training Sequence Error 7-26
7.2.5 Link Command Detection and Declarations 7-27
7.2.6 Summary of Error Types and Recovery 7-28

7.3 PowerOn Reset and Inband Reset 7-28
7.3.1 PowerOn Reset 7-28
7.3.2 Inband Reset 7-29
7.4 Link Training and Status State Machine (LTSSM) 7-30
7.4.1 SS.Disabled 7-33
7.4.2 SS.Inactive 7-33
7.4.2.1 SS.Inactive.Quiet 7-34
7.4.2.2 SS.Inactive.Disconnect.Detect 7-34
7.4.3 Rx.Detect 7-35
7.4.3.1 Rx.Detect.Reset 7-35
7.4.3.2 Rx.Detect.Active 7-35
7.4.3.3 Rx.Detect.Quiet 7-36
7.4.4 Polling 7-38
7.4.4.1 Polling.LFPS 7-38
7.4.4.2 Polling.RxEQ 7-39
7.4.4.3 Polling.Active 7-39
7.4.4.4 Polling.Configuration 7-40
7.4.4.5 Polling.Idle 7-40
7.4.5 Compliance Mode 7-42
7.4.6 U0 7-42
7.4.7 U1 7-44
7.4.8 U2 7-45
7.4.9 U3 7-46
7.4.10 Recovery 7-47
7.4.10.1 Recovery.Active 7-48
7.4.10.2 Recovery.Configuration 7-48
7.4.10.3 Recovery.Idle 7-49
7.4.11 Loopback 7-50
7.4.11.1 Loopback.Active 7-51
7.4.11.2 Loopback.Exit 7-51
7.4.12 Hot Reset 7-52
7.4.12.1 Hot Reset.Active 7-52
7.4.12.2 Hot Reset.Exit 7-53

8 Protocol Layer

8.1 USB 3.0 Transactions 8-2
8.2 Packet Types 8-2
8.3 Packet Formats 8-3
8.3.1 Fields Common to all Headers 8-4
8.3.1.1 Packet Revision Field 8-4
8.3.1.2 Packet Type Field 8-4
8.3.1.3 CRC-16 8-5
8.3.1.4 Link Control Word 8-5
8.4 Link Management Packet (LMP) 8-6
8.4.1 Subtype Field 8-6
8.4.2 Set Link Function 8-7
8.4.3 U2 Inactivity Timeout 8-8
8.4.4 Vendor Device Test 8-8
8.4.5 Port Capabilities 8-9

8.4.6 Port Configuration 8-10
8.4.7 Port Configuration ACK 8-12
8.5 Transaction Packet (TP) 8-13
8.5.1 ACK Transaction Packet 8-13
8.5.2 NRDY Transaction Packet 8-15
8.5.3 Endpoint Ready (ERDY) Transaction Packet 8-15
8.5.4 STATUS Transaction Packet 8-16
8.5.5 STALL Transaction Packet 8-16
8.5.6 Device Notification (DEV_NOTIFICATION) Transaction Packet 8-17
8.5.6.1 Function Wake Device Notification 8-18
8.5.6.2 Latency Tolerance Messaging Device Notification 8-18
8.5.6.3 Bus Interval Adjustment Message Device Notification 8-19
8.5.6.4 Best Effort Latency Tolerance Messaging 8-20
8.5.6.5 Bus Interval Adjustment Messaging 8-21
8.5.7 PING Transaction Packet 8-23
8.5.8 PING_RESPONSE Transaction Packet 8-23
8.6 Data Packet (DP) 8-24
8.7 Isochronous Timestamp Packet (ITP) 8-25
8.8 Addressing Triple 8-26
8.9 Route String Field 8-26
8.9.1 Route String Port Field 8-27
8.9.2 Route String Port Field Width 8-27
8.9.3 Port Number 8-27
8.10 Transaction Packet Usages 8-27
8.10.1 Flow Control Conditions 8-28
8.10.2 Burst Transactions 8-28
8.10.3 Short Packets 8-29
8.11 TP Responses 8-29
8.11.1 Device Response to TP Requesting Data 8-29
8.11.2 Host Response to Data Received from a Device 8-30
8.11.3 Device Response to Data Received from the Host 8-31
8.11.4 Device Response to a SETUP DP 8-32
8.12 TP Sequences 8-33
8.12.1 Bulk Transactions 8-33
8.12.1.1 State Machine Notation Information 8-33
8.12.1.2 Bulk IN Transactions 8-34
8.12.1.3 Bulk OUT Transactions 8-35
8.12.1.4 Introduction to Bulk Streaming Protocol 8-38
8.12.2 Control Transfers 8-48
8.12.2.1 Reporting Status Results 8-49
8.12.2.2 Variable-length Data Stage 8-50
8.12.2.3 STALL TPs Returned by Control Pipes 8-50
8.12.3 Bus Interval and Service Interval 8-50
8.12.4 Interrupt Transactions 8-51
8.12.4.1 Interrupt IN Transactions 8-51
8.12.4.2 Interrupt OUT Transactions 8-55

8.12.5 Host Timing Information....8-58
8.12.6 Isochronous Transactions....8-62
8.12.6.1 Host Flexibility in Performing Isochronous Transactions....8-65
8.12.6.2 Device Response to Isochronous IN Transactions....8-65
8.12.6.3 Host Processing of Isochronous IN Transactions....8-65
8.12.6.4 Device Response to an Isochronous OUT Data Packet....8-66
8.13 Timing Parameters....8-67

9 Device Framework

9.1 USB Device States....9-1
9.1.1 Visible Device States....9-1
9.1.1.1 Attached....9-4
9.1.1.2 Powered....9-4
9.1.1.3 Default....9-4
9.1.1.4 Address....9-5
9.1.1.5 Configured....9-5
9.1.1.6 Suspended....9-5
9.1.2 Bus Enumeration....9-5
9.2 Generic USB Device Operations....9-6
9.2.1 Dynamic Attachment and Removal....9-6
9.2.2 Address Assignment....9-6
9.2.3 Configuration....9-7
9.2.4 Data Transfer....9-7
9.2.5 Power Management....9-8
9.2.5.1 Power Budgeting....9-8
9.2.5.2 Changing Device Suspend State....9-8
9.2.5.3 Function Suspend....9-8
9.2.5.4 Changing Function Suspend State....9-9
9.2.6 Request Processing....9-9
9.2.6.1 Request Processing Timing....9-9
9.2.6.2 Reset/Resume Recovery Time....9-10
9.2.6.3 Set Address Processing....9-10
9.2.6.4 Standard Device Requests....9-10
9.2.6.5 Class-specific Requests....9-10
9.2.6.6 Speed Dependent Descriptors....9-11
9.2.7 Request Error....9-11
9.3 USB Device Requests....9-12
9.3.1 bmRequestType....9-12
9.3.2 bRequest....9-13
9.3.3 wValue....9-13
9.3.4 wIndex....9-13
9.3.5 wLength....9-13
9.4 Standard Device Requests....9-14
9.4.1 Clear Feature....9-16
9.4.2 Get Configuration....9-17
9.4.3 Get Descriptor....9-18
9.4.4 Get Interface....9-19

9.4.5 Get Status 9-19
9.4.6 Set Address 9-22
9.4.7 Set Configuration 9-23
9.4.8 Set Descriptor 9-24
9.4.9 Set Feature 9-24
9.4.10 Set Interface 9-27
9.4.11 Set Isochronous Delay 9-27
9.4.12 Set SEL 9-28
9.4.13 Synch Frame 9-29
9.5 Descriptors 9-29
9.6 Standard USB Descriptor Definitions 9-30
9.6.1 Device 9-30
9.6.2 Binary Device Object Store (BOS) 9-32
9.6.2.1 USB 2.0 Extension 9-33
9.6.2.2 SuperSpeed USB Device Capability 9-34
9.6.3 Configuration 9-36
9.6.4 Interface Association 9-38
9.6.5 Interface 9-39
9.6.6 Endpoint 9-41
9.6.7 SuperSpeed Endpoint Companion 9-45
9.6.8 String 9-47
9.7 Device Class Definitions 9-48
9.7.1 Descriptors 9-48
9.7.2 Interface(s) 9-48
9.7.3 Requests 9-48

10 Hub, Host Downstream Port, and Device Upstream Port Specification

10.1 Hub Feature Summary 10-1
10.1.1 SuperSpeed Capable Host with SuperSpeed Capable Software 10-4
10.1.2 USB 2.0 Host 10-4
10.1.3 Hub Connectivity 10-5
10.1.3.1 Packet Signaling Connectivity 10-5
10.1.3.2 Routing Information 10-6
10.1.4 Resume Connectivity 10-8
10.1.5 Hub Fault Recovery Mechanisms 10-8
10.1.6 Hub Header Buffer Architecture 10-9
10.1.6.1 Hub Data Buffer Architecture 10-10
10.2 Hub Power Management 10-10
10.2.1 Link States 10-10
10.2.2 U1/U2 Timers 10-10
10.2.3 Downstream/Upstream Port Link State Transitions 10-11
10.3 Downstream Facing Ports 10-11
10.3.1 Downstream Facing Port State Descriptions 10-13
10.3.1.1 DSPORT.Powered-off 10-13
10.3.1.2 DSPORT.Disconnected (Waiting for SS Connect) 10-14
10.3.1.3 DSPORT.Training 10-14
10.3.1.4 DSPORT.ERROR 10-15
10.3.1.5 DSPORT.Enabled 10-15

10.3.1.6 DSPORT.Resetting 10-15
10.3.2 Disconnect Detect Mechanism 10-16
10.3.3 Labeling 10-16
10.4 Downstream Facing Ports Power Management 10-16
10.4.1 Downstream Facing Port PM Timer 10-16
10.4.2 Downstream Facing Port State Descriptions 10-18
10.4.2.1 Enabled U0 States 10-18
10.4.2.2 Attempt U0 – U1 Transition 10-19
10.4.2.3 Attempt U0 – U2 Transition 10-20
10.4.2.4 Link in U1 10-20
10.4.2.5 Link in U2 10-20
10.4.2.6 Link in U3 10-20
10.5 Upstream Facing Port 10-21
10.5.1 Upstream Facing Port State Descriptions 10-22
10.5.1.1 USPORT.Powered-off 10-22
10.5.1.2 USPORT.Powered-on 10-22
10.5.1.3 USPORT.Training 10-22
10.5.1.4 USPORT.Connected 10-23
10.5.1.5 USPORT.ERROR 10-23
10.5.1.6 USPORT.Enabled 10-23
10.6 Upstream Facing Port Power Management 10-23
10.7 Hub Header Forwarding and Data Repeater 10-24
10.7.1 Hub Elasticity Buffer 10-24
10.7.2 SKP Ordered Sets 10-25
10.7.3 Interpacket Spacing 10-25
10.7.4 Header Buffer Architecture 10-26
10.7.5 Upstream Facing Port TX 10-28
10.7.6 Upstream Facing Port TX State Descriptions 10-29
10.7.6.1 TX Idle 10-29
10.7.6.2 TX Header 10-29
10.7.6.3 TX Data 10-29
10.7.6.4 TX Data Abort 10-29
10.7.6.5 TX Link Command 10-30
10.7.7 Upstream Facing Port RX 10-30
10.7.8 Upstream Facing Port RX State Descriptions 10-31
10.7.8.1 RX Idle 10-31
10.7.8.2 RX Data 10-31
10.7.8.3 RX Header 10-31
10.7.8.4 Process Header 10-32
10.7.8.5 RX Link Command 10-33
10.7.8.6 Process Link Command 10-33
10.7.9 Downstream Facing Port TX 10-33
10.7.10 Downstream Facing Port TX State Descriptions 10-35
10.7.10.1 TX Idle 10-35
10.7.10.2 TX Header 10-35
10.7.10.3 TX Data 10-35
10.7.10.4 TX Data Abort 10-35
10.7.10.5 TX Link Command 10-36
10.7.11 Downstream Facing Port RX 10-36

10.7.12 Downstream Facing Port RX State Descriptions 10-37
10.7.12.1 RX Idle 10-37
10.7.12.2 RX Data 10-37
10.7.12.3 RX Header 10-37
10.7.12.4 Process Header 10-38
10.7.12.5 RX Link Command 10-38
10.7.12.6 Process Link Command 10-38
10.7.13 SuperSpeed Packet Connectivity 10-39
10.8 Suspend and Resume 10-39
10.9 Hub Reset Behavior 10-39
10.10 Hub Port Power Control 10-40
10.10.1 Multiple Gangs 10-40
10.11 Hub Controller 10-41
10.11.1 Endpoint Organization 10-41
10.11.2 Hub Information Architecture and Operation 10-42
10.11.3 Port Change Information Processing 10-43
10.11.4 Hub and Port Status Change Bitmap 10-44
10.11.5 Over-current Reporting and Recovery 10-45
10.11.6 Enumeration Handling 10-46
10.12 Hub Configuration 10-46
10.13 Descriptors 10-48
10.13.1 Standard Descriptors for Hub Class 10-48
10.13.2 Class-specific Descriptors 10-50
10.13.2.1 Hub Descriptor 10-50
10.14 Requests 10-52
10.14.1 Standard Requests 10-52
10.14.2 Class-specific Requests 10-53
10.14.2.1 Clear Hub Feature 10-55
10.14.2.2 Clear Port Feature 10-55
10.14.2.3 Get Hub Descriptor 10-56
10.14.2.4 Get Hub Status 10-56
10.14.2.5 Get Port Error Count 10-58
10.14.2.6 Get Port Status 10-58
10.14.2.7 Set Hub Descriptor 10-65
10.14.2.8 Set Hub Feature 10-65
10.14.2.9 Set Hub Depth 10-65
10.14.2.10 Set Port Feature 10-66
10.15 Host Root (Downstream) Ports 10-71
10.16 Device Upstream Ports 10-71
10.16.1 Peripheral Device Upstream Ports 10-72
10.16.1.1 Peripheral Device Connect State Machine 10-72
10.16.2 Peripheral Device Connect State Descriptions 10-73
10.16.2.1 PCONNECT.Powered-off 10-73
10.16.2.2 PCONNECT.Attempt SS Connect 10-73
10.16.2.3 PCONNECT.Connected on SS 10-73
10.16.2.4 PCONNECT.Connected on USB 2.0 10-73
10.16.2.5 PCONNECT.Connected on USB 2.0 Attempting SS Connection 10-73
10.17 Hub Chapter Parameters 10-74

11 Interoperability and Power Delivery

11.1 USB 3.0 Host Support for USB 2.0 11-1
11.2 USB 3.0 Hub Support for USB 2.0 11-2
11.3 USB 3.0 Device Support for USB 2.0 11-2
11.4 Power Distribution 11-2
11.4.1 Classes of Devices and Connections 11-3
11.4.1.1 Self-powered Hubs 11-4
11.4.1.2 Low-power Bus-powered Devices 11-5
11.4.1.3 High-power Bus-powered Devices 11-5
11.4.1.4 Self-powered Devices 11-6
11.4.2 Steady-State Voltage Drop Budget 11-6
11.4.3 Power Control During Suspend/Resume 11-7
11.4.4 Dynamic Attach and Detach 11-9
11.4.4.1 Inrush Current Limiting 11-9
11.4.4.2 Dynamic Detach 11-10
11.4.5 VBUS Electrical Characteristics 11-10
11.4.6 Powered-B Connector 11-10
11.4.7 Wire Gauge Table 11-11

A Symbol Encoding

B Symbol Scrambling

B.1 Data Scrambling B-1

C Power Management

C.1 SuperSpeed Power Management Overview C-1
C.1.1 Link PM States and Entry & Exit Processes C-2
C.1.1.1 Summary of Link PM States C-2
C.1.1.2 U0 – Link Active C-2
C.1.1.3 U1 – Link Idle with Fast Exit C-3
C.1.1.4 U2 – Link Idle with Slow Exit C-4
C.1.1.5 U3 – Link Suspend C-5
C.1.2 Link PM for Hubs C-6
C.1.2.1 Link PM State Coordination and Management C-6
C.1.2.2 Packet Deferring C-7
C.1.2.3 Software Interface C-7
C.1.3 Other Link PM Support Mechanisms C-9
C.1.3.1 Packet Pending Flag C-9
C.1.3.2 Support for Isochronous Transfers C-9
C.1.3.3 Support for Interrupt Transfers C-10
C.1.4 Device PM C-10
C.1.4.1 Function Suspend C-10
C.1.4.2 Device Suspend C-10
C.1.4.3 Device and Fabric Suspend Entry Process C-11
C.1.4.4 Host Initiated Device and Fabric Wakeup Process C-11
C.1.4.5 Device and Fabric Remote Wakeup Process C-12
C.1.5 Platform PM Support C-13
C.2 SuperSpeed vs. High Speed Interface Selection Considerations for Devices C-14
C.3 Link Power Management Implementation Guidelines C-16

C.3.1 Recommendations for Device Initiated U1 and U2 Entry C-16
C.3.1.1 System Information Available To Devices..... C-16
C.3.1.2 Control Endpoints..... C-17
C.3.1.3 Bulk Endpoints C-17
C.3.1.4 Interrupt Endpoints..... C-18
C.3.1.5 Isochronous Endpoints..... C-19
C.3.1.6 Recommendations for Control Endpoints C-20
C.3.1.7 Recommendations for Bulk Endpoints C-20
C.3.1.8 Recommendations for Interrupt Endpoints..... C-21
C.3.1.9 Recommendations for Isochronous Endpoints C-22
C.3.1.10 Hub Downstream Ports Connected to another Hub..... C-23
C.4 Device Power Management Implementation Guidelines..... C-23
C.4.1 Deciding When to Initiate Selective Suspend..... C-23
C.4.1.1 Background Information..... C-23
C.4.1.2 Implementation Recommendations..... C-24
C.4.2 Latency Tolerance Messaging (LTM) Implementation Guidelines..... C-24
C.4.2.1 Device State Model and State Transition Algorithm..... C-25
C.4.2.2 Other Considerations C-27
C.4.3 Removing Power From Portions of a Device During Suspend..... C-27
C.4.3.1 Devices Which Must Create Their Own Switched Power Rail..... C-28
C.4.3.2 Devices Which Have a Switched Power Rail Provided by System C-29
C.4.3.3 Self Powered Devices..... C-30
C.4.3.4 Identifying What Needs To Remain Powered During Suspend..... C-30
C.5 Calculation Examples for U1 and U2 System Exit Latency Fields C-31
C.5.1 Path 1: Device Connected Directly to Host, Downstream Direction C-32
C.5.2 Path 2: Device Connected Directly to Host, Upstream Direction..... C-32
C.5.3 Path 3: Device Connected to Hub That is Connected to Host, Downstream Direction C-32
C.5.4 Path 4: Device Connected to Hub That is Connected to Host, Upstream Direction..... C-34

Figures

3-1 USB 3.0 SuperSpeed Communications Layers and Power Management Plane Layers 3-2
3-2 USB 3.0 SuperSpeed Communications Layers 3-4
3-3. USB 3.0 System Level Topology 3-7
3-4. USB 3.0 Hub Architecture 3-8
3-5. Examples of Supported SuperSpeed USB Physical Device Topologies 3-9
3-6. SuperSpeed USB Interface and Protocol Layers 3-11
3-7. USB 3.0 Standard-A Cable Connector and Host Receptacle 3-12
3-8. USB 3.0 Micro-B Cable Connector and Device Receptacle 3-12
3-9. USB 3.0 Standard B Cable Connector and Device Receptacle 3-13
3-10. USB 3.0 Cable 3-13
3-11. Bus Expansion Example - Three Levels Of 4-Port Hubs 3-16
3-12. High Level Hub Architecture 3-17
4-1. USB 2.0 style Transfer Mapped onto SuperSpeed USB Transfer 4-3
4-2. SuperSpeed USB IN Transaction Protocol 4-6
4-3. SuperSpeed USB OUT Transaction Protocol 4-7
4-4. USB SS IN Stream Example 4-12
5-1. USB 3.0 Standard-A Receptacle Interface Dimensions 5-8
5-2. USB 3.0 Standard-A Plug Interface Dimensions 5-11
5-3. Reference Footprint for the USB 3.0 Standard-A Receptacle 5-12
5-4. Reference Footprint for the USB 3.0 Double-Stacked Standard-A Receptacle 5-13
5-5. USB 3.0 Standard-B Receptacle Interface Dimensions 5-16
5-6. USB 3.0 Standard-B Plug Interface Dimensions 5-17
5-7. Reference Footprint for the USB 3.0 Standard-B Receptacle 5-18
5-8. USB 3.0 Power-B Receptacle Interface Dimensions 5-21
5-9. USB 3.0 Power-B Plug Interface Dimensions 5-22
5-10. Reference Footprint for USB 3.0 Power-B Receptacle 5-23
5-11. USB 3.0 Micro-B and-AB Receptacles Interface Dimensions 5-26
5-12. USB 3.0 Micro-B and –A Plug Interface Dimensions 5-29
5-13. Reference Footprint for the USB 3.0 Micro-B or -AB Receptacle 5-31
5-14. Illustration of a USB 3.0 Cable Cross-Section 5-34
5-15. USB 3.0 Standard-A to USB 3.0 Standard-B Cable Assembly 5-36
5-16. USB 3.0 Micro-B Plug Cable Overmold Dimensions 5-38
5-17. USB 3.0 Micro-A Cable Overmold Dimensions 5-40
5-18. Impedance limits of a Mated Connector 5-44
5-19. Illustration of Test Points for a Mated Cable Assembly 5-45
5-20. Differential Return Loss Requirement-Reference 5-46
5-21. Differential Insertion Loss Requirement 5-47
5-22. Differential Near-End Crosstalk Requirement between USB 3.0 Pairs 5-48
5-23. Differential Near-End and Far-End Crosstalk Requirement between USB 2.0 Pair and USB 3.0 Pairs 5-49
5-24. Differential to Common Mode Conversion Requirement 5-50
5-25. 4-Axes Continuity Test 5-53
5-26. Mated USB 3.0 Standard-A Connector 5-56
5-27. Mated USB 3.0 Standard-B Connector 5-56
5-28. Mated USB 3.0 Micro-B Connector 5-57

6-1. Super Speed Block Diagram: Physical 6-1
6-2. Transmitter Block Diagram 6-2
6-3. Receiver Block Diagram 6-3
6-4. Channel Models without Cable (Top) and with Cable (Bottom) 6-4
6-5. Character to Symbol Mapping 6-5
6-6. Bit Transmission Order 6-5
6-7. LFSR with Scrambling Polynomial 6-7
6-8. Jitter Filtering – “Golden PLL” and Jitter Transfer Functions 6-13
6-9. “Golden PLL” and Jitter Transfer Functions 6-13
6-10. Period Modulation from Triangular SSC 6-15
6-11. Generic Eye Mask 6-16
6-12. Single-ended and Differential Levels 6-17
6-13. Device Termination Schematic 6-17
6-14. Tx Normative Setup with Reference Channel 6-20
6-15. Tx Compliance Rx EQ Transfer Function 6-20
6-16. De-Emphasis Waveform 6-21
6-17. Frequency Spectrum of TSEQ 6-22
6-18. Rx Tolerance Setup 6-25
6-19. Jitter Tolerance Curve 6-26
6-20. LFPS Signaling 6-27
6-21. U1 Exit, U2 Exit, and U3 Wakeup LFPS Handshake Timing Diagram 6-28
6-22. Example of Warm Reset Out of U3 6-30
6-23. Rx Detect Schematic 6-31
7-1. Link Layer 7-1
7-2. Packet Header with Framing Symbol, CRC-16 and Link Control Word 7-2
7-3. Packet Header 7-3
7-4. CRC-16 Generation 7-4
7-5. Link Control Word 7-5
7-6. CRC-5 Generation 7-6
7-7. Data Packet Payload with CRC-32 and Framing 7-6
7-8. CRC-32 Generation 7-8
7-9. Data Packet with Data Packet Header followed by Data Packet Payload Without Spacing 7-8
7-10. Link Command Structure 7-9
7-11. Link Command Word Structure 7-10
7-11. State Diagram of the Link Training and Status State Machine 7-32
7-12. SS.Inactive Substate Machine 7-34
7-13. Rx.Detect Substate Machine 7-37
7-14. Polling Substate Machine 7-41
7-15. U1 7-44
7-16. U2 7-46
7-17. U3 7-47
7-18. Recovery Substate Machine 7-50
7-19. Loopback Substate Machine 7-52
7-20. Hot Reset Substate Machine 7-54
8-1. Protocol Layer Highlighted 8-1
8-2. Example Transaction Packet 8-3
8-3. Link Control Word Detail 8-5
8-4. Link Management Packet Structure 8-6

8-5. Set Link Function LMP 8-7
8-6. U2 Inactivity Timeout LMP 8-8
8-7. Vendor Device Test LMP 8-8
8-8. Port Capabilty LMP 8-9
8-9. Port Configuration LMP 8-10
8-10. Port Configuration ACK LMP 8-12
8-11. ACK Transaction Packet 8-13
8-12. NRDY Transaction Packet 8-15
8-13. ERDY Transaction Packet 8-15
8-14. STATUS Transaction Packet 8-16
8-15. STALL Transaction Packet 8-16
8-16. Device Notification Transaction Packet 8-17
8-17. Function Wake Device Notification 8-18
8-18. Latency Tolerance Messaging Device Notification 8-18
8-19. Bus Interval Adjustment Message Device Notification 8-19
8-20. PING Transaction Packet 8-23
8-21. PING_RESPONSE Transaction Packet 8-23
8-22. Example Data Packet 8-24
8-23. Isochronous Timestamp Packet 8-26
8-24. Route String Detail 8-27
8-25. Legend for State Machines 8-34
8-26. Sample BULK IN Sequence 8-36
8-27. Sample BULK OUT Sequence 8-37
8-28. General Stream Protocol State Machine (SPSM) 8-38
8-29. Bulk IN Stream Protocol State Machine (ISPSM) 8-40
8-30. IN Move Data State Machine (IMDSM) 8-42
8-31. OUT Stream Protocol State Machine (OSPSM) 8-44
8-32. OUT Move Data State Machine (OMDSM) 8-46
8-33. Control Read and Write Sequences 8-49
8-34. Host Sends Interrupt IN Transaction in Each Service Interval 8-52
8-35. Host Stops Servicing Interrupt IN Transaction Once NRDY is Received 8-53
8-36. Host Resumes IN Transaction after Device Sent ERDY 8-53
8-37. Endpoint Sends STALL TP 8-54
8-38. Host Detects Error in Data and Device Resends Data 8-54
8-39. Host Sends Interrupt OUT Transaction in Each Service Interval 8-56
8-40. Host Stops Servicing Interrupt OUT Transaction Once NRDY is Received 8-56
8-41. Host Resumes Sending Interrupt OUT Transaction after Device Sent ERDY 8-57
8-42. Device Detects Error in Data and Host Resends Data 8-57
8-43. Endpoint Sends STALL TP 8-58
8-44. Host Timing Information for Isochronous OUT Endpoint 8-59
8-45. Host Timing Information for Isochronous IN Endpoint 8-60
8-46. Multiple Active Isochronous Endpoints with Aligned Service Interval Boundaries 8-61
8-47. Isochronous Transaction Format 8-62
8-48. Sample Isochronous IN and OUT Transactions 8-63
8-49. Isochronous IN and OUT Transaction Example 8-64
9-1. Device State Diagram 9-2
9-2. wIndex Format when Specifying an Endpoint 9-13
9-3. wIndex Format when Specifying an Interface 9-13

9-4. Information Returned by a GetStatus() Request to a Device 9-20
9-5. Information Returned by a GetStatus() Request to an Interface 9-21
9-6. Information Returned by a GetStatus() Request to an Endpoint 9-21
9-7. Example of Feedback Endpoint Relationships 9-44
10-1. Hub Architecture 10-2
10-2. SuperSpeed Portion of the Hub Architecture 10-3
10-3. Simple USB 3.0 Topology 10-4
10-4. Hub Signaling Connectivity 10-5
10-5. Route String Example 10-7
10-6. Resume Connectivity 10-8
10-7. Typical Hub Header Buffer Architecture 10-9
10-8. Example Hub Data Buffer Traffic - Header Buffer Only Shown for DS Port 1. 10-10
10-9. Downstream Facing Hub Port State Machine 10-12
10-10. Downstream Facing Hub Port Power Management State Machine 10-17
10-11. Upstream Facing Hub Port State Machine 10-21
10-12. Example Hub Header Buffer Architecture - Downstream Traffic 10-26
10-13. Example Hub Header Buffer Architecture - Upstream Traffic 10-27
10-14. Upstream Facing Port TX State Machine 10-28
10-15. Upstream Facing Port RX State Machine 10-30
10-16. Downstream Facing Port TX State Machine 10-34
10-17. Downstream Facing Port RX State Machine 10-36
10-18. Example Hub Controller Organization 10-41
10-19. Relationship of Status, Status Change, and Control Information to Device States 10-42
10-20. Port Status Handling Method 10-43
10-21. Hub and Port Status Change Bitmap 10-44
10-22. Example Hub and Port Change Bit Sampling 10-45
10-23. Peripheral Device Connect State Machine 10-72
11-1. Compound Self-powered Hub 11-4
11-2. Low-power Bus-powered Function 11-5
11-3. High-power Bus-powered Function 11-5
11-4. Self-powered Function 11-6
11-5. Worst-case Voltage Drop Topology (Steady State) 11-7
11-6. Worst-case Voltage Drop Analysis Using Equivalent Resistance. 11-7
11-7. Typical Suspend Current Averaging Profile 11-8
B-1. Scrambling Spectrum for Data Value of 0 ..... B-6
C-1. Flow Diagram for Host Initiated Wakeup C-12
C-2. SuperSpeed Device Data Transfer C-14
C-3. High Speed Device Data Transfer C-15
C-4. Selection of U1 vs. U2 With Isochronous Endpoints C-19
C-5. Power Rails For Generic Bus Powered Device C-28
C-6. Power Rails For Bus Powered Device With Switched Rail C-28
C-7. Power Rails For PC Application C-29
C-8. Power Rails For Self Powered Device C-30
C-9. Host to Device Path Exit Latency Calculation Examples C-31
C-10. Downstream Host to Device Path Exit Latency with Hub C-33
C-11. Upstream Device to Host Path Exit Latency with Hub C-34

Tables

3-1. Comparing SuperSpeed USB to USB 2.0 3-6
3-2. Link PM States and Characteristics Summary 3-19
5-1. Plugs Accepted By Receptacles 5-2
5-2. USB 3.0 Standard-A Connector Pin Assignments 5-14
5-3. USB 3.0 Standard-B Connector Pin Assignments 5-19
5-4. USB 3.0 Power-B Connector Pin Assignments 5-24
5-7. Cable Wire Assignments 5-35
5-8. Reference Wire Gauges 5-35
5-9. USB 3.0 Standard-A to USB 3.0 Standard-B Cable Assembly Wiring 5-37
5-10. USB 3.0 Standard-A to USB 3.0 Micro-B Cable Assembly Wiring 5-39
5-11. USB 3.0 Micro-A to USB 3.0 Micro-B Cable Assembly Wiring 5-41
5-12. USB 3.0 Micro-A to USB 3.0 Standard-B Cable Assembly Wiring 5-42
5-13. SDP Differential Insertion Loss Examples 5-44
5-14. Environmental Test Conditions 5-54
5-15. Reference Materials[1,2] 5-55
6-1. Special Symbols 6-8
6-2. TSEQ Ordered Set 6-9
6-3. TS1 Ordered Set 6-10
6-4. TS2 Ordered Set 6-10
6-5. Link Functionality Symbols 6-10
6-6. SKP Ordered Set Structure 6-11
6-7. Compliance Pattern Sequences 6-11
6-8. Informative Jitter Budgeting at the Device 6-12
6-9. SSC Parameter 6-14
6-10. Transmitter Normative Electrical Parameters 6-18
6-11. Transmitter Informative Electrical Parameters at Silicon Pads 6-18
6-12. Normative Transmitter Eye Mask 6-19
6-13. Receiver Electrical Parameters 6-23
6-14. BRST 6-24
6-15. BDAT 6-25
6-16. BERC 6-25
6-17. BCNT 6-25
6-18. Jitter Requirements 6-26
6-19. Normative LFPS Electrical Specification 6-27
6-20. LFPS Timing 6-28
6-21. LFPS Handshake Timing for U1/U2 Exit, Loopback Exit, and U3 Wakeup 6-29
7-1. CRC-16 Mapping 7-4
7-2. CRC-32 Mapping 7-7
7-3. Link Command Ordered Set Structure 7-9
7-4. Link Command Bit Definitions 7-10
7-5. Link Command Definitions 7-11
7-6. Logical Idle Definition 7-13
7-7. Transmitter Timers Summary 7-20
7-8. Link Flow Control Timers Summary 7-20
7-9. Valid Packet Framing K-Symbol Order (K = SHP, SDP, END, EDB) 7-25
7-10. Valid Link Command K-Symbol Order 7-27
7-11. Error Types and Recovery 7-28
7-12. Nominal LTSSM Timeouts 7-30

8-1. Packet Revision Field Description....8-4
8-2. Packet Type Field Description....8-4
8-3. Link Control Word Format....8-5
8-4. Link Management Packet Subtype Field....8-6
8-5. Set Link Function....8-7
8-6. U2 Inactivity Timer Functionality....8-8
8-7. Vendor-specific Device Test Function....8-9
8-8. Port Capability LMP Format....8-9
8-9. Port Configuration LMP Format (differences with Port Capability LMP)....8-11
8-10. Port Type Selection Matrix....8-11
8-11. Port Configuration ACK LMP Format (differences with Port Capability LMP)....8-12
8-12. Transaction Packet Subtype Field....8-13
8-13. ACK TP Format....8-14
8-14. NRDY TP Format (differences with ACK TP)....8-15
8-15. ERDY TP Format (differences with ACK TP)....8-16
8-16. STATUS TP Format (differences with ACK TP)....8-16
8-17. STALL TP Format (differences with ACK TP)....8-17
8-18. Device Notification TP Format (differences with ACK TP)....8-17
8-19. Function Wake Device Notification....8-18
8-20. Latency Tolerance Messaging Device Notification....8-19
8-21. Bus Interval Adjustment Message Device Notification....8-20
8-22. PING TP Format (differences with ACK TP)....8-23
8-23. PING_RESPONSE TP Format (differences with ACK TP)....8-24
8-24. Data Packet Format (differences with ACK TP)....8-25
8-25. Isochronous Timestamp Packet Format....8-26
8-26. Device Responses to TP Requesting Data (non-isochronous endpoints)....8-29
8-27. Host Responses to Data Received from a Device (non-isochronous endpoints)....8-30
8-28. Device Responses to OUT Transactions (non-isochronous endpoints)....8-31
8-29. Device Responses to SETUP Transactions (only for control endpoints)....8-32
8-30. Status Stage Responses....8-50
8-31. Device Responses to Isochronous IN Transactions....8-65
8-32. Host Responses to IN Transactions....8-66
8-33. Device Responses to OUT Data Packets....8-66
8-34. Timing Parameters....8-67
9-1. Visible Device States....9-3
9-2. Format of Setup Data....9-12
9-3. Standard Device Requests....9-14
9-4. Standard Request Codes....9-15
9-5. Descriptor Types....9-15
9-6. Standard Feature Selectors....9-16
9-7. Test Mode Selectors....9-25
9-8. Suspend Options....9-26
9-9. Standard Device Descriptor....9-31
9-10. BOS Descriptor....9-32
9-11. Format of a Device Capability Descriptor....9-32
9-12. Device Capability Type Codes....9-33
9-13. USB_30 Device Capabilities Descriptor....9-33
9-14. SuperSpeed Device Capabilities Descriptor....9-34

9-15. Standard Configuration Descriptor 9-36
9-16. Standard Interface Association Descriptor 9-38
9-17. Standard Interface Descriptor 9-40
9-18. Standard Endpoint Descriptor 9-41
9-20. SuperSpeed Endpoint Companion Descriptor 9-45
9-21. String Descriptor Zero, Specifying Languages Supported by the Device 9-47
9-22. UNICODE String Descriptor 9-47
10-1. Downstream Facing Port Signal/Event Definitions 10-13
10-2. Downstream Port VBUS Requirements 10-14
10-3. Upstream Facing Port Signal/Event Definitions 10-22
10-4. Hub Power Operating Mode Summary 10-46
10-5. SuperSpeed Hub Descriptor 10-50
10-6. Hub Responses to Standard Device Requests 10-52
10-7. Hub Class Requests 10-53
10-8. Hub Class Request Codes 10-54
10-9. Hub Class Feature Selectors 10-54
10-10. Hub Status Field, *wHubStatus* 10-57
10-11. Hub Change Field, *wHubChange* 10-57
10-12. Port Status Field, *wPortStatus* 10-59
10-13. Port Change Field, *wPortChange* 10-63
10-14. U1 Timeout Value Encoding 10-66
10-15. U2 Timeout Value Encoding 10-67
10-16. Test Mode Selector Codes 10-69
10-17. Downstream Port Remote Wake Mask Encoding 10-69
10-18. Hub Parameters 10-74
11-1. USB 3.0 and USB 2.0 Interoperability 11-1
11-2. DC Electrical Characteristics 11-10
11-3. VBUS/Gnd Wire Gauge vs. Maximum Length 11-11
A-1. 8b/10b Data Symbol Codes A-1
C-1. Link PM States and Characteristics Summary C-2

1 Introduction

1.1 Motivation

The original motivation for the Universal Serial Bus (USB) came from several considerations, two of the most important being:

- Ease-of-use

  The lack of flexibility in reconfiguring the PC had been acknowledged as the Achilles' heel to its further deployment. The combination of user-friendly graphical interfaces and the hardware and software mechanisms associated with new-generation bus architectures have made computers less confrontational and easier to reconfigure. However, from the end user's point of view, the PC's I/O interfaces, such as serial/parallel ports, keyboard/mouse/joystick interfaces, etc., did not have the attributes of plug-and-play.

- Port Expansion

  The addition of external peripherals continued to be constrained by port availability. The lack of a bi-directional, low-cost, low-to-mid speed peripheral bus held back the creative proliferation of peripherals such as storage devices, answering machines, scanners, PDA's, keyboards, mice, etc. Existing interconnects were optimized for one or two point products. As each new function or capability was added to the PC, a new interface had been defined to address this need.

Initially, USB provided two speeds (12 Mb/s and 1.5 Mb/s) that peripherals could use. As PCs became increasingly powerful and able to process larger amounts of data, users needed to get more and more data into and out of their PCs. This led to the definition of the USB 2.0 specification in 2000 to provide a third transfer rate of 480 Mb/s while retaining backward compatibility. In 2005, with wireless technologies are becoming more and more capable, Wireless USB was introduced to provide a new cable free capability to USB.

USB is the most successful PC peripheral interconnect ever defined and it has migrated heavily into the CE and Mobile segments. In 2006 alone over 2 billion USB devices were shipped and there are over 6 billion USB products in the installed base today. End users "know" what USB is. Product developers understand the infrastructure and interfaces necessary to build a successful product.

USB has gone beyond just being a way to connect peripherals to PCs. Printers use USB to interface directly to cameras. PDAs use USB connected keyboards and mice. The USB On-The-Go definition provides a way for two host-capable devices to be connected and negotiate which one will operate as the "host". USB, as a protocol, is also being picked up and used in many non-traditional applications such as industrial automation.

Now, as technology innovation marches forward, new kinds of devices, media formats and large inexpensive storage are converging. They require significantly more bus bandwidth to maintain the interactive experience users have come to expect. HD Camcorders will have tens of gigabytes of storage that the user will want to move to their PC for editing, viewing and archiving. Furthermore existing devices like still image camera continue to evolve and are increasing their storage capacity to hold even more uncompressed images. Downloading hundreds or even thousands of ten megabyte or larger raw images from a digital camera will be a time consuming process unless the

transfer rate is increased. In addition, user applications demand a higher performance connection between the PC and these increasingly sophisticated peripherals. USB 3.0 addresses this need by adding a even higher transfer rate to match these new usages and devices.

Thus, USB (wired or wireless) continues to be the answer to connectivity for PC, CE and Mobile architectures. It is a fast, bi-directional, isochronous, low-cost, dynamically attachable interface that is consistent with the requirements of the PC platform of today and tomorrow.

1.2 Objective of the Specification

This document defines the next generation USB industry-standard, USB 3.0. The specification describes the protocol definition, types of transactions, bus management, and the programming interface required to design and build systems and peripherals that are compliant with this standard.

USB 3.0's goal remains to enable devices from different vendors to interoperate in an open architecture, while maintaining and leveraging the existing USB infrastructure (device drivers, software interfaces, etc.). The specification is intended as an enhancement to the PC architecture, spanning portable, business desktop, and home environments, as well as simple device-to-device communications. It is intended that the specification allow system OEMs and peripheral developers adequate room for product versatility and market differentiation without the burden of carrying obsolete interfaces or losing compatibility.

1.3 Scope of the Document

The specification is primarily targeted to peripheral developers and platform/adapter developers, but provides valuable information for platform operating system/ BIOS/ device driver, adapter IHVs/ISVs, and system OEMs. This specification can be used for developing new products and associated software.

Product developers using this specification are expected to know and understand the USB 2.0 Specification. Specifically, USB 3.0 devices must implement device framework commands and descriptors as defined in the USB 2.0 specification.

1.4 USB Product Compliance

Adopters of the USB 3.0 specification have signed the USB 3.0 Adopters Agreement, which provides them access to a reasonable and non-discriminatory (RANDZ) license from the Promoters and other Adopters to certain intellectual property contained in products that are compliant with the USB 3.0 specification. Adopters can demonstrate compliance with the specification through the testing program as defined by the USB Implementers Forum. Products that demonstrate compliance with the specification will be granted certain rights to use the USB Implementers Forum logos as defined in the logo license.

1.5 Document Organization

Chapters 1 through 4 provide an overview for all readers, while Chapters 5 through 10 contain detailed technical information defining USB 3.0.

- Peripheral implementers should particularly read Chapters 5 through 10.
- Host Controller implementers should particularly read Chapters 3 through 9.

Readers are also requested to contact operating system vendors for operating system bindings specific to USB 3.0.

1.6 Design Goals

USB 3.0 is the next evolutionary step for wired USB. The goal is that end users view it as the same as USB 2.0, just faster. Several key design areas to meet this goal are listed below.

- Preserve the USB model of smart host and simple device.
- Leverage the existing USB infrastructure. There are a vast number of USB products in use today. A large part of their success can be traced to the existence of stable software interfaces, easily developed software device drivers, and a number of generic standard device class drivers (HID, Mass Storage, audio, etc.). Super Speed USB devices are designed to keep this software infrastructure intact so that developers of peripherals can continue to use the same interfaces and leverage all of their existing development work.
- Significantly improve power management. Reduce the active power when sending data and reduce idle power by provide a richer set of power management mechanisms to allow devices to drive the bus into lower power states.
- Ease of use has always been and remains a key design goal for all varieties of USB.
- Preserve the investment. There are a large number of PCs that only support USB 2.0 in use. There are a larger number of USB 2.0 peripherals in use. Retaining backward compatibility at the type A connector to allow super speed devices to be used albeit at high speed with USB 2.0 PCs and allow high speed devices with their existing cables to be connected to the USB 3.0 super speed type-A connectors.

1.7 Related Documents

USB 2.0 Specification

USB OTG Supplement

Micro-USB Cables and Connectors Supplement to USB 2.0 Specification

EIA-364-1000.01: Environmental Test Methodology for Assessing the Performance of Electrical Connectors and Sockets Used in Business Office Applications

PCI Express Base Specification, Revision 2.0*

2 Terms and Abbreviations

This chapter lists and defines terms and abbreviations used throughout this specification.

Term/Abbreviation	Definition
A-Device	A device with a Type-A plug inserted into its receptacle. The A-device supplies power to VBUS and is host at the start of a session. If the A-device is On-The-Go, it may relinquish the role of host to an On-The-Go B-device under certain conditions.
ACK	Handshake packet indicating a positive acknowledgment.
Active Device	A device that is powered and is not in the Suspend state.
Asynchronous Data	Data transferred at irregular intervals with relaxed latency requirements.
Attached	A downstream device attached to an upstream device when there is a physical cable between the two.
Audio Device	A device that sources or sinks sampled analog data.
AWG#	The measurement of a wire's cross section, as defined by the American Wire Gauge standard.
B-Device	A device with a Type-B plug inserted into its receptacle. The B-device is a peripheral at the start of a session. If the B-device is OTG, it may be granted the role of host from an OTG A-device.
Bandwidth	The amount of data transmitted per unit of time, typically bits per second (b/s) or bytes per second (B/s).
Big Endian	A method of storing data that places the most significant byte of multiple-byte values at a lower storage address. For example, a 16-bit integer stored in big endian format places the least significant byte at the higher address and the most significant byte at the lower address. See also little endian.
Bit	A unit of information used by digital computers. Represents the smallest piece of addressable memory within a computer. A bit expresses the choice between two possibilities and is typically represented by a logical one (1) or zero (0).
b/s	Transmission rate expressed in bits per second.
B/s	Transmission rate expressed in bytes per second.
Buffer	Storage used to compensate for a difference in data rates or time of occurrence of events, when transmitting data from one device to another.
Bulk Transfer	One of the four USB transfer types. Bulk transfers are non-periodic, large bursty communication typically used for a transfer that can use any available bandwidth and can also be delayed until bandwidth is available. See also transfer type.
Bus Enumeration	Detecting and identifying USB devices.
Bus Interval	A 125 µs period that establishes the integral boundary of Service Intervals.
Byte	A data element that is eight bits in size.
Cable	Raw cable with no plugs attached.
Cable assembly	Cable attached with plugs.
Captive cable	Cable assembly with only one end attached with a plug; the other end is permanently attached to a device.
Capabilities	Those attributes of a USB device that are administrated by the host.
CDR	Circuit that performs the clock and data recovery function.

Term/Abbreviation	Definition
Characteristics	Those qualities of a USB device that are unchangeable; for example, the device class is a device characteristic.
Client	Software resident on the host that interacts with the USB System Software to arrange data transfer between a function and the host. The client is often the data provider and consumer for transferred data.
Component	A physical chip or circuit that contains a port.
Configuring Software	Software resident on the host software that is responsible for configuring a USB device. This may be a system configurator or software specific to the device.
Control Endpoint	A pair of device endpoints with the same endpoint number that are used by a control pipe. Control endpoints transfer data in both directions and, therefore, use both endpoint directions of a device address and endpoint number combination. Thus, each control endpoint consumes two endpoint addresses.
Control Pipe	Same as a message pipe.
Connected	A downstream device is connected to an upstream device when it is attached to the upstream device, and when the downstream device has asserted either the D+ or D- data line in order to enter low-speed, full-speed or high-speed signaling.
Control Transfer	One of the four USB transfer types. Control transfers support configuration/command/status type communications between client and function. See also transfer type.
CRC	CRC-5, CRC-16, CRC-32, See Cyclic Redundancy Check.
Cyclic Redundancy Check (CRC)	A check performed on data to see if an error has occurred in transmitting, reading, or writing the data. The result of a CRC is typically stored or transmitted with the checked data. The stored or transmitted result is compared to a CRC calculated for the data to determine if an error has occurred.
D Codes	The data type codes used in 8b/10b encoding.
D+ and D-	Differential pair defined in USB 2.0 specification
Default Address	An address defined by the USB Specification and used by a USB device when it is first powered or reset. The default address is 00H.
Default Pipe	The message pipe created by the USB System Software to pass control and status information between the host and a USB device's endpoint zero.
Descrambling	Restoring the pseudo-random 8 bit character to the original state. See scrambling,
Detached	A downstream device detached to an upstream device when there the physical cable between the two is removed.
Device	A logical or physical entity that performs one or more functions. The actual entity described depends on the context of the reference. At the lowest level, device may refer to a single hardware component, as in a memory device. At a higher level, it may refer to a collection of hardware components that perform a particular function, such as a USB interface device. At an even higher level, device may refer to the function performed by an entity attached to the USB. Devices may be physical, electrical, addressable, and logical. When used as a non-specific reference, a USB device is either a hub or a peripheral device.
Device Address	A seven-bit value representing the address of a device on the USB. The device address is the default address (00H) when the USB device is first powered or the device is reset. Devices are assigned a unique device address by the USB System Software.
Device Endpoint	A uniquely addressable portion of a USB device that is the source or sink of information in a communication flow between the host and device. See also endpoint address.

Term/Abbreviation	Definition
Device Software	Software that is responsible for using a USB device. This software may or may not also be responsible for configuring the device for use.
Disconnected (Unconnected)	A downstream device is disconnected to an upstream device when it is attached to the upstream device, and when the downstream device has not asserted either the D+ or D- data line in order to enter low-speed, full-speed or high-speed signaling.
Downstream	The direction of data flow from the host or away from the host. A downstream port is the port on a hub electrically farthest from the host that generates downstream data traffic from the hub. Downstream ports receive upstream data traffic.
Downstream Facing Port	See Downstream Port.
Downstream Port	A port is one a device connects to. A system's root ports are downstream ports.
Driver	When referring to hardware, an I/O pad that drives an external load. When referring to software, a program responsible for interfacing to a hardware device, that is, a device driver.
DWORD	Double word. A data element that is two words (i.e., four bytes or 32 bits) in size.
Dynamic Insertion and Removal	The ability to attach and remove devices while the host is in operation.
DP	Data Packet which consists of a Data Packet Header followed by a Data Packet Payload.
DPH	Data Packet Header. Contains the data packet's address, route, length and other information about the packet.
DPP	Data Packet Payload. Contains the data packet's data and a 32-bit CRC.
Dual Simplex	Two data paths that each independently carry traffic in one direction.
Endpoint	See device endpoint.
Endpoint Address	The combination of an endpoint number and an endpoint direction on a USB device. Each endpoint address supports data transfer in one direction.
Endpoint Direction	The direction of data transfer on the USB. The direction can be either IN or OUT. IN refers to transfers to the host; OUT refers to transfers from the host.
Endpoint Number	A four-bit value between 0H and FH, inclusive, associated with an endpoint on a USB device.
External Port	See port.
Frame Number	The bus interval counter value (as defined by the ITS) divided (integer division) by 8.
Full-duplex	Computer data transmission occurring in both directions simultaneously.
Full-speed (FS)	USB operation at 12 Mb/s. See also low-speed and high-speed.
Function	A set of one or more related interfaces on a USB device that exposes a capability to a software client.
Gb/s	Transmission rate expressed in gigabits per second (1,000,000,000 bits/second).
GB	Gigabyte or 1,099,511,627,776 bytes.
GB/s	Transmission rate expressed in gigabytes per second.
Handshake Packet	A packet that acknowledges or rejects a specific condition. For examples, see ACK, NRDY or ERDY.
Header	Packet header. DPH, LMP and TP are all headers.
High-speed (HS)	USB operation at 480 Mb/s. See also low-speed and full-speed.
Host	The host computer system where the USB Host Controller is installed. This includes the host hardware platform (CPU, bus, etc.) and the operating system in use.
Host Controller	The host's USB interface.
Hot Reset	Reset mechanism using TS1/TS2 Ordered Sets.

Term/Abbreviation	Definition
Hub	A USB device that provides additional connections to the USB.
Hub Tier	One plus the number of USB links in a communication path between the host and a peripheral device. See Figure 4-1.
ID	Identification. Denotes the pin on the USB 3.0 Micro-B connector family that is used to differentiate a USB 3.0 Micro-A plug from a USB 3.0 Micro-B plug.
Inband Reset	Mechanism that relies on SuperSpeed and/or LFPS signaling to propagate the reset across the link.
Informative	Information given for illustrative purposes only and contains no requirements. See normative.
Interrupt Transfer	One of the four USB transfer types. Interrupt transfer characteristics are small data, non-periodic, low-frequency, and bounded-latency. Interrupt transfers are typically used to handle service needs. See also transfer type.
Isochronous Data	A stream of data whose timing is implied by its delivery rate.
Isochronous Device	An entity with isochronous endpoints, as defined in the USB Specification, that sources or sinks sampled analog streams or synchronous data streams.
Isochronous Sink Endpoint	An endpoint that is capable of consuming an isochronous data stream that is sent by the host.
Isochronous Source Endpoint	An endpoint that is capable of producing an isochronous data stream and sending it to the host.
Isochronous Transfer	One of the four USB transfer types. Isochronous transfers are used when working with isochronous data. Isochronous transfers provide periodic, continuous communication between host and device. See also transfer type.
ITP	Isochronous Timestamp Packet
Jitter	A tendency toward lack of synchronization caused by mechanical or electrical changes. More specifically, the phase shift of digital pulses over a transmission medium.
KB	Kilobyte or 1,024 bytes
K Codes	The control type codes used in 8b/10b encoding. SHP – start header packet SDP – start data packet END – end header or data packet EDB – end of nullified (bad) packet SLC – start link command COM – comma SKP – skip EPF – end packet framing
LFPS	Low frequency periodic signal. Used to communicate information across a link without using SuperSpeed protocol.
LFSR	Linear feedback shift register. Used to create pseudo-random characters for scrambling.
Link Command	Link Command is an 8 Symbol special Ordered Set used for link-level flow control, power management and device removal.
Link Command Word	Two byte word with 11-bits to define the Link command and a 5-bit CRC5 to ensure data integrity.
Link Control Word	Two byte word with 11-bits to define the Link-Level Flow control and a 5-bit CRC5 to ensure data integrity.

Term/Abbreviation	Definition
Little Endian	Method of storing data that places the least significant byte of multiple-byte values at lower storage addresses. For example, a 16-bit integer stored in little endian format places the least significant byte at the lower address and the most significant byte at the next address. See also big endian.
LMP	Link Management Packet. A type of header packet used to communicate information between a pair of links.
Logical Idle	Period of one or more Symbol times when no information (Packets, Special K-Symbols or Link Commands) is being transmitted when link is in U0.
Low-speed	USB operation at 1.5 Mb/s. See also full-speed and high-speed.
LSb	Least significant bit.
LSB	Least significant byte.
LTSSM	Link Training and Status State Machine.
Mb/s	Transmission rate expressed in megabits per second (1,000,000 bits/second).
MB	Megabyte or 1,048,576 bytes.
MB/s	Transmission rate expressed in megabytes per second.
Message Pipe	A bi-directional pipe that transfers data using a request/data/status paradigm. The data has an imposed structure that allows requests to be reliably identified and communicated.
MSb	Most significant bit.
MSB	Most significant byte.
Normative	Required by the specification. See informative.
NRDY	Handshake packet indicating a negative acknowledgment.
OTG	On-The-Go
OTG device	A device with the host and peripheral capabilities
Packet	A bundle of data organized in a group for transmission. Packets typically contain three elements: control information (e.g., source, destination, and length), the data to be transferred, and error detection and correction bits.
Packet Buffer	The logical buffer used by a USB device for sending or receiving a single packet. This determines the maximum packet size the device can send or receive.
Peripheral	A physical entity that is attached to a USB cable and is currently operating as a “device” as defined in the USB Specification, Revision 3.0. The Peripheral responds to low level bus requests from the Host.
Peripheral Device	A non-hub USB device that provides one or more functions to the host, such as a Mass Storage Device.
Persistent	State information (e.g., a descriptor field) with the following characteristics: State is retained and persistent during D3, including transitions into and out of D3.
Phase Locked Loop (PLL)	A circuit that acts as a phase detector to keep an oscillator in phase with an incoming frequency.
Physical Device	A device that has a physical implementation; e.g., speakers, microphones, and CD players.
Pipe	A logical abstraction representing the association between an endpoint on a device and software on the host. A pipe has several attributes; for example, a pipe may transfer data as streams (stream pipe) or messages (message pipe). See also stream pipe and message pipe.
PLL	See Phase Locked Loop.
Plug	Connector attached to the cable, to be mated with the receptacle

Term/Abbreviation	Definition
Port	Point of access to or from a system or circuit. For the USB, the point where a USB device is attached.
Power On Reset (POR)	An event to restore a SuperSpeed Link and return a Port to its power on initial status.
PPM	Parts per million.
PRBS	Pseudo-random bit stream.
Protocol	A specific set of rules, procedures, or conventions relating to format and timing of data transmission between two devices.
RA	See rate adaptation.
Rate Adaptation	The process by which an incoming data stream, sampled at Fs_i, is converted to an outgoing data stream, sampled at Fs_o, with a certain loss of quality, determined by the rate adaptation algorithm. Error control mechanisms are required for the process. Fs_i and Fs_o can be different and asynchronous. Fs_i is the input data rate of the RA; Fs_o is the output data rate of the RA.
Receptacle	Connector mounted on the host or device, to be mated with the plug
Request	A request made to a USB device contained within the data portion of a SETUP packet.
Root Hub	A USB hub directly attached to the Host Controller. This hub (tier 1) is attached to the host.
Root Port	The downstream port on a Root Hub.
Scrambling	The process of changing an 8 bit character in a pseudo-random way. See descrambling.
Service Interval	An integral multiple of bus intervals within which a periodic endpoint must be serviced.
Service Jitter	The deviation of service delivery from its scheduled delivery time.
Service Rate	The number of services to a given endpoint per unit time.
SSC	Spread Spectrum Clock.
Stage	One part of the sequence composing a control transfer; stages include the Setup stage, the Data stage, and the Status stage.
STP	Shielded twist pair (cable)
Stream Pipe	A pipe that transfers data as a stream of samples with no defined USB structure.
SuperSpeed (SS)	USB operation at 5 Gb/s. Features a dual-simplex physical layer, unicast messaging and concurrent bi-directional data flow.
Synchronization Type	A classification that characterizes an isochronous endpoint's capability to connect to other isochronous endpoints.
Termination	Passive components attached at the end of cables to prevent signals from being reflected or echoed.
Timeout	A time interval within which an expected event occurs.
TP	Transaction Packet. A type of header packet used to communicate information between a device and the host.
Training Sequences	Ordered Sets for initializing bit and Symbol alignment and Receiver Equalization. Examples TS1, TS2, TSEQ.
Transaction	The delivery of service to an endpoint: The IN consists of an ACK TP with a response of NRDY or DP The OUT consists of a DP with a response of NRDY or an ACK TP.
Transfer	One or more bus transactions to move information between a software client and its function.
Transfer Type	Determines the characteristics of the data flow between a software client and its function. Four standard transfer types are defined: control, interrupt, bulk, and isochronous.

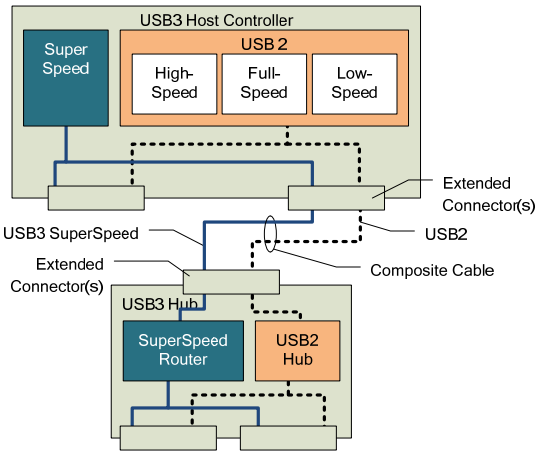
Term/Abbreviation	Definition
Type-A Connector	The standard-A connector defined in the USB 2.0 specification
Universal Serial Bus Driver	The host resident software entity responsible for providing common services to clients that are manipulating one or more functions on one or more Host Controllers.
Upstream	The direction of data flow towards the host. An upstream port is the port on a device electrically closest to the host that generates upstream data traffic from the hub. Upstream ports receive downstream data traffic.
Upstream Port	A port that a device uses to connect to a host. The port on all devices is an upstream port.
Upstream Facing Port	See Upstream Port.
USB 3.0 Standard-A connector	USB 3.0 host connector, supporting SS mode
USB 3.0 Powered-B connector	The Standard Type-B device connector, supporting USB 3.0 SS mode with optional device power delivery
USB 3.0 Standard-B connector	The Standard Type-B device connector, supporting USB 3.0 SS mode
USB 3.0 Micro-A plug	Part of the USB 3.0 Micro-B connector family for OTG use; it can be plugged into a USB 3.0 Micro-AB receptacle; it differs from the USB 3.0 Micro-B plug only in keying and ID pin connection.
USB 3.0 Micro-AB receptacle	Part of the USB 3.0 Micro-B connector family; it accepts either a USB 3.0 Micro-B plug or a USB 3.0 Micro-A plug.
USB 3.0 Micro-B connector	USB 3.0 device connector, supporting SS mode
USB 3.0 Micro connector family	All the receptacles and plugs that are used on devices, including the USB 3.0 Micro-B, USB 3.0 Micro-AB and USB 3.0 Micro-A connectors.
USB 2.0 Standard-A connector	The Type-A connector defined by the USB 2.0 specification
USB 2.0 Standard-B connector	The Standard Type-B connector defined by the USB 2.0 specification
USB-IF	USB Implementers Forum, Inc. is a nonprofit corporation formed to facilitate the development of USB compliant products and promote the technology.
UTP	Unshielded twist pair (cable)
Warm Reset	Reset mechanism using LFPS.
Word	A data element that is two bytes (16 bits) in size.

3 SuperSpeed USB Architectural Overview

This chapter presents an overview of SuperSpeed USB architecture and key concepts with specific emphasis on those aspects of SuperSpeed USB that differ from USB 2.0. Like earlier versions of USB, this version remains focused on implementing a cable bus supporting data exchange between a host computer and a wide range of simultaneously accessible peripherals. The attached peripherals share USB bandwidth through a host-scheduled protocol. The bus allows peripherals to be attached, configured, used, and detached while the host and other peripherals are in operation.

Later chapters describe the various components and specific requirements of SuperSpeed USB in greater detail. This overview has been written with the understanding that the reader already has a fundamental understanding of USB 2.0.

3.1 USB 3.0 Overview



USB3 SuperSpeed Composite Bus

USB 3.0 SuperSpeed introduces a number of significant features for USB I/O. These include:

- A significant speed bump to USB data communications
- Unicast packets to minimize the number of active links
- Asynchronous notifications to remove the need for polling
- Localized link power management as a fundamental part of the bus architecture
- Backward compatibility with USB2 at the driver and physical levels

The speed up incorporates a new PHY architecture that provides a dual simplex communications channel (whereas USB2 uses a half-duplex channel) and a new protocol that takes advantage of the full-duplex nature of the SuperSpeed bus.

Physical backward compatibility with USB2 is accomplished via a composite cable/connector that lays down a physical USB2 bus in parallel with the USB3 SuperSpeed bus (see TBD). The USB2 and USB3 SuperSpeed buses are independent and devices connect to one or the other at the highest rate possible. Logical (software) backward compatibility is accomplished with the new protocols' support for the standard USB Framework.

3.1.1 SuperSpeed Architecture Overview

This section provides an overview of the USB 3.0 SuperSpeed Bus Architecture with a very high level description of the layers that exist on a host and a device. The physical layers are directly connected over the physical medium (wire) to provide the actual chip to chip communication. The link layers are logically connected to construct and manages the flow of packets between pairs of ports. The protocol layers are logically connected to manage the control and flow of data between a host and device. The device framework chapter defines the standard descriptors and requests used by the default control pipe. Device class and Vendor specific requests and descriptors that sit on top of the stack and their definition are outside the bus specification. Power management cuts vertically thorough the layers to support fine grain power management. Figure 3-1 provides a graphical view of the layering.

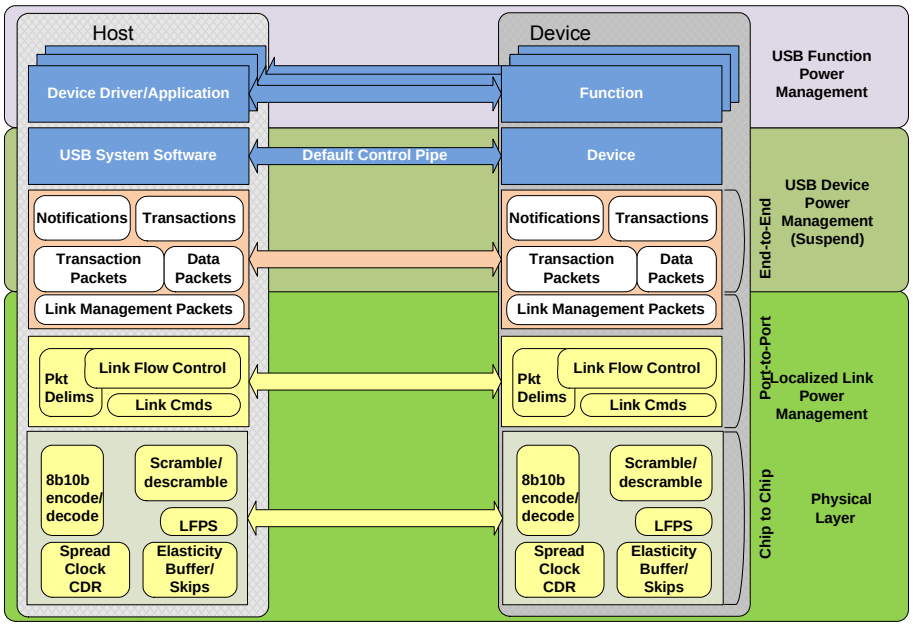


Figure 3-1. USB 3.0 SuperSpeed Communications Layers and Power Management Plane Layers

3.1.1.1 Physical Layer

The function of the physical layer is to reliably exchange information between a pair of ports, typically the host and the device. The physical layer receives 8 bit data from the link layer and scrambles the data to reduce EMI emissions. It then encodes the scrambled 8 bit data into 10 bit symbols for transmission on the wires. The resultant data are sent at a rate that includes spread spectrum to further lower the EMI emissions. The data are received in serial form, one bit at a time. The clock used to sample the data is extracted directly from the data itself, this is called embedded clocking. The bits are assembled into 10 bit symbols, decoded and descrambled. The resultant 8 bit data are then sent to the link layer for further processing.

The link layer also directs the physical layer initialization and power management. The physical layer uses low frequency periodic signaling (LFPS) to signal initialization and power management

information. LFPS is a low frequency differential clock signal that is relatively simple to generate and detect and uses very little power. It is used when the normal signaling speed is not available because the electrical interconnect has not been initialized or is in a low power state.

The link layer and the physical layer have been designed together to provide reliable data transmission. The physical layer is designed to transmit the data with an error rate that does not exceed 1 bit error in every 10^{12} bits on average. The link layer uses error checking to catch these errors and resend the packet further reducing the packet header error rate.

3.1.1.2 Link Layer

The link layer of USB 3.0 is responsible for initializing and maintaining SuperSpeed link-level communications, providing mechanisms to control the flow of packets on the link at a low-level, and managing the link-level operational modes to include entry/exit of power management states.

Initializing a USB bus link starts with detecting when devices are connected across the USB cable. USB SuperSpeed connections involve providing local and detecting remote receiver termination of the two ports that are being connected. The connection process then proceeds to training the link by exchanging a series of special bit streams designed to allow for bit and symbol synchronization.

The link layer also introduces a Low Frequency Periodic Signaling (LFPS) mechanism for communications between the two ports when the link is in a low power management state. The electrical specification of LFPS is such that an LFPS transmitter and an LFPS receiver can be implemented using very little power. The LFPS is used in a variety of applications including exit from a low power state, remote wakeup from a suspended state, initial handshake between the two ports during training, and Warm Reset.

Similar to USB 2.0, clock synchronization across the connected link is a derivative by-product of closely tracking signal transitions on the bus but unlike USB High-Speed, SuperSpeed clocking has to be maintained to a significantly tighter tolerance and is further complicated by the need to support Spread Spectrum clocking for reducing bus-generated EMI. To aid in adapting for clock drift between the two ports, special compensation symbols are inserted between data transmissions so that data buffer overflow or underflow is mitigated in cases where the sending port is operating faster or slower than the receiving port.

To optimize the link operation, a Link Training Status State Machine (LTSSM) is introduced not only to maintain the link training and connectivity, but also to preserve the link power by incorporating a number of low power link states that target for different degrees of power savings based on the link activity. In addition, various compliance and test modes are built in the LTSSM to allow for efficient monitoring and diagnostics of link health. The LTSSM achieves these goals by utilizing special symbols and ordered sets defined in the physical layer.

Finally, link commands are defined for communications between the two link partners to maintain link level packet flow and link power management. The construction of link commands and packet framings is such that they are tolerant to a single bit error.

3.1.1.3 Protocol Layer

The function of the protocol layer is to move data provided by USB system software and device drivers between the host and device and to manage the end to end flow of that data. It defines the structure of the various packet types, defines how those packets are used in support of the various

USB transfer types and the timing parameters that devices must meet to ensure a high level of responsiveness and throughput.

The protocol layer also defines asynchronous notifications that may be used to by devices to request service, send latency tolerance messages, send wake notifications etc.

In support of the isochronous transfer type, SuperSpeed has a periodic timestamp notification mechanism that allows devices to synchronize their clocks with the host. To minimize the effects of power management in isochronous transfers, a Ping/Ping_Response mechanism is defined to put the affected links in an active power state.

Data packets use a 32 bit CRC, sequence numbers and retries to ensure reliable data transfers. Stream IDs have been added to support for multiple commands and out of order execution that allow for example improved performance of storage class devices.

The protocol layer defines link management packets used to configure and manage a link e.g. to enable the debug port or force a specific link power state.

3.1.1.4 Hubs

Hubs are the basic building block of the USB SuperSpeed fabric expanding the number of available ports. They play a number of key roles in USB. Firstly they provide a connection between downstream devices and the host. Secondly they manage the connection for example by sending port status notifications when there is a change on the port, they play a key role in power management for example automatically setting their upstream port's link state based on their downstream ports' link states and deferring traffic routed through a downstream port that is not in the active state. They also replicate forward multicast timestamp notifications sent by the host.

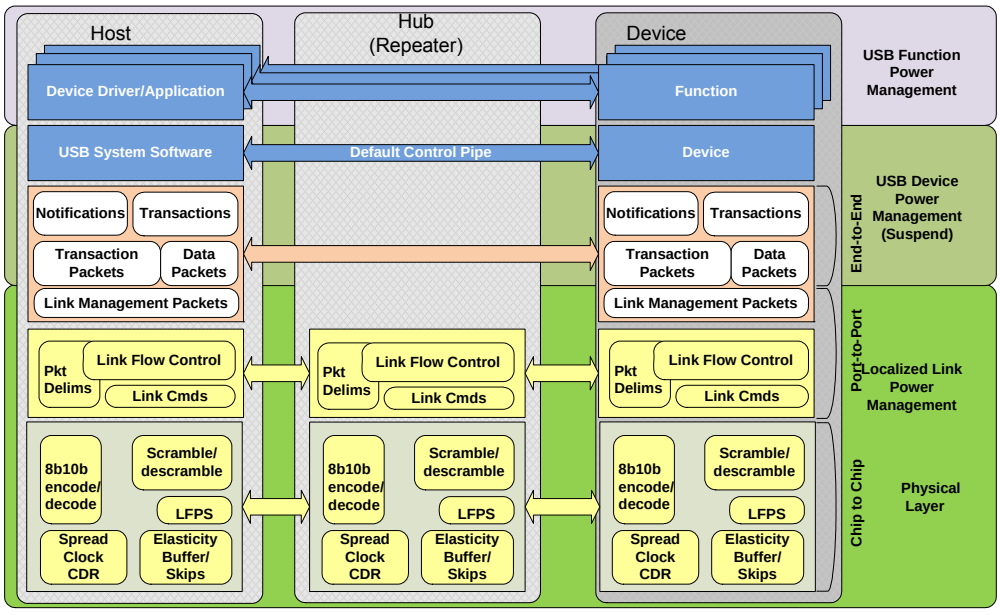


Figure 3-2. USB 3.0 SuperSpeed Communications Layers

The SuperSpeed hub routes downstream packets received on its upstream port to the designated downstream port. It routes upstream packets received on any of its downstream ports to its upstream port. It uses a store and forward mechanism for header packets and a buffered cut through model for data packets.

The repeater part of the hub, operates below the protocol layer as shown in the center portion of Figure 3-2, simply forwards packets, manages link power states due to inactivity and reflects link power states. The hub as a device looks like the right side of Figure 3-2; it configures ports, sets inactivity timer values, changes link power states, under the direction of software.

3.1.1.5 Power Management

USB 3.0 SuperSpeed introduces a top-to-bottom power management plane. These power management layers are not tightly coupled per se but do have certain dependencies. For example, the function power management is not available when the device is suspended and the device suspend state is a lower power state than the link-level power states.

The link power management is driven by the device locally and is propagated upwards by hubs so the host may not be able to track the link-level power management states (of individual links). Due to this, the end-to-end protocol accommodates scenarios where links between the host and specific device are not propagating traffic. An example of this is that the end-to-end packet protocol packets traverse a directed path (i.e. are specifically routed) between the host, through intervening hubs and the device, and if any of the links the packet has to traverse is in a lower power state, the hub will defer the packet and return it to the host.

3.2 USB 3.0 System

The USB 3.0 uses the same basic building blocks as USB 2.0, although they have been significantly upgraded to improve speed, performance and power management.

3.2.1 Comparing SuperSpeed USB to USB 2.0

Table 3-1 summarizes the key architectural differences between SuperSpeed USB and USB 2.0. SuperSpeed USB does not replace the need for USB 2.0; rather it is intended to augment USB 2.0 such that applications requiring higher bandwidth than supported by High-Speed USB can be implemented. SuperSpeed USB is specifically architected to re-use as much of the existing USB 2.0 protocol and software model as possible.

Table 3-1. Comparing SuperSpeed USB to USB 2.0

Characteristic	SuperSpeed USB	USB 2.0
Data Rate	SuperSpeed (5.0 Gbps)	Low-Speed (1.5 Mbps), Full-Speed (12 Mbps) and High-Speed (480 Mbps)
Data Interface	Dual-simplex, four-wire differential signaling separate from USB 2.0 signaling Simultaneous bi-directional data flow	Half-duplex two-wire differential signaling Unidirectional data flow with negotiated directional bus transitions
Cable signal count	Eight: 4 for SS data path 2 for HS data path 2 for power/ground	Four: 2 for LS/FS/HS data path 2 for power/ground
Bus transaction protocol	Host directed, asynchronous traffic flow Packet traffic is explicitly routed	Host directed, polled traffic flow Packet traffic is broadcast to all devices
Power management	Multi-level link power management supporting idle, sleep and suspend states	Port-level suspend with two levels of entry/exit latency
Bus power	Same as for USB 2.0 with a 50% increase for unconfigured power and an 80% increase for configured power.	Support for low/high bus-powered devices with low-power limits for un-configured and suspended devices.

3.2.2 System Level Topology

The USB 3.0 system-level topology is made up of a host platform, a collection of intermediate hubs, and peripheral devices. At this level, USB 3.0 topology is very much the same as USB 2.0 topology. At the bus level, USB 3.0 differs quite dramatically given its simultaneous bi-directional packet routed interface compared to the USB 2.0 half-duplex broadcast bus operation.

Figure 3-3 illustrates the generalized system-level topology for USB 3.0.

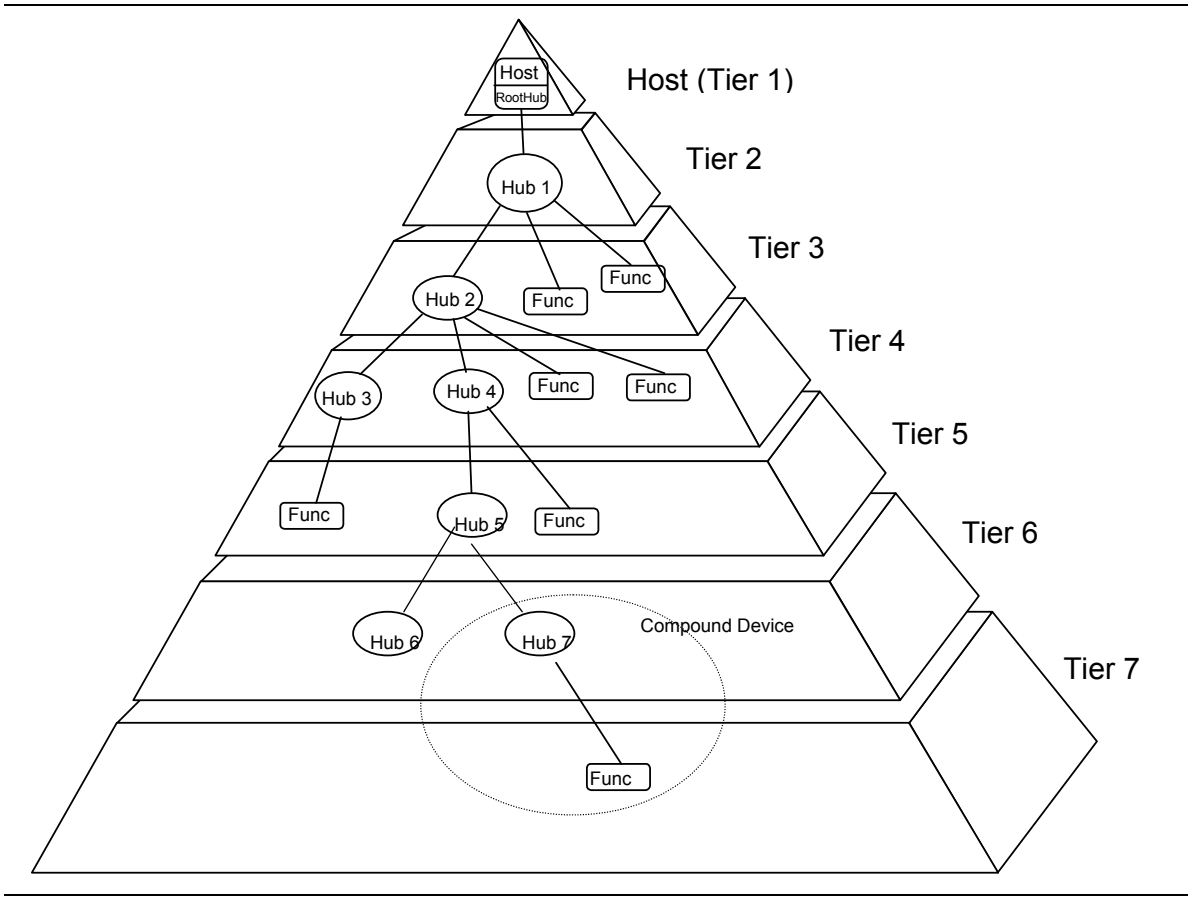


Figure 3-3. USB 3.0 System Level Topology

3.2.2.1 Hosts

The USB 3.0 host platform represents the root (or ultimate upstream port) of the bus topology. SuperSpeed USB operation and traffic flow is dictated by a host controller device as managed by the host platform's system software in support of host applications.

SuperSpeed USB uses the same programming model to communicate with SuperSpeed devices as described in Chapter 10 of the USB 2.0 specification.

3.2.2.2 Hubs

USB 3.0 hubs provide a means to expand the number of devices that can be simultaneously connected to and operated over the SuperSpeed bus. USB 3.0 hubs are also required to support USB 2.0 bus operation for connected devices that don't support SuperSpeed capabilities; this support is accomplished by incorporating the functionality of a USB 2.0 hub in parallel with SuperSpeed hub functionality.

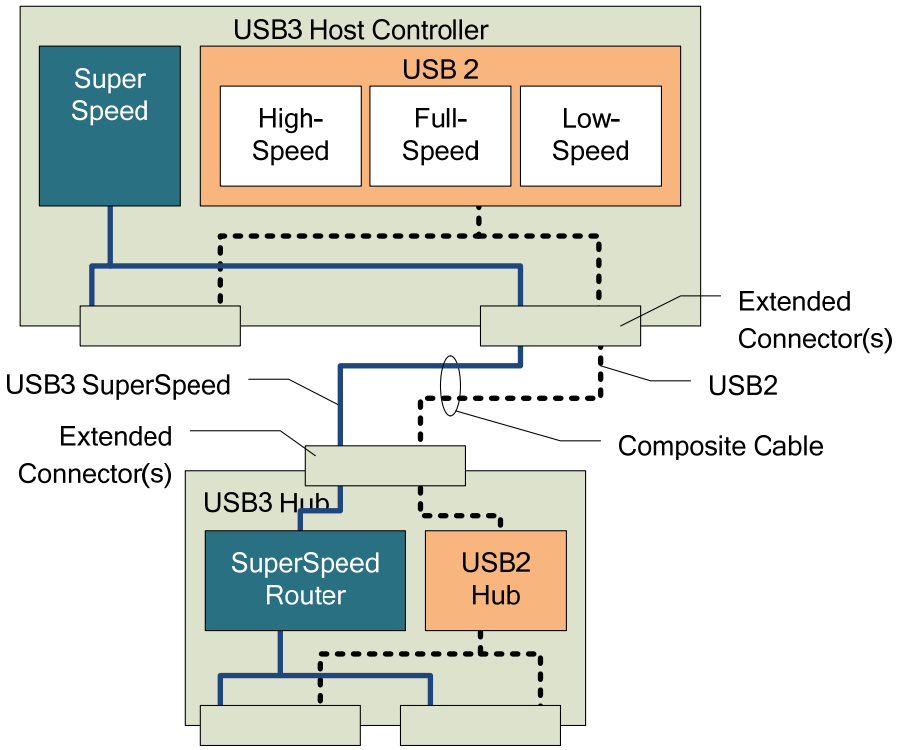


Figure 3-4. USB 3.0 Hub Architecture

Figure 3-4 is illustrative of a USB 3.0 hub. The choice to effectively incorporate an independent USB 2.0 hub controller is arbitrary. Other designs are possible as long as they meet the hub requirements as defined in both the USB 2.0 and 3.0 specifications.

3.2.2.3 Devices

Similar to USB 2.0 devices, SuperSpeed USB devices within a single physical package (i.e., a single peripheral) can consist of a number of functional topologies including single function, multiple functions on a single peripheral device (composite device), and permanently attached peripheral devices behind an integrated hub (compound device).

For any given SuperSpeed USB peripheral device within a single physical package, only one USB connection mode, either SuperSpeed USB or a USB 2.0 speed shall be established for operation with the host. There is no requirement for a SuperSpeed USB capable host to support multiple parallel connections, i.e., simultaneous operation over SuperSpeed USB and USB 2.0, to a given

peripheral device inclusive of all of the interfaces to that device. As described in Section TBD of the Hub Chapter, USB 3.0 hubs are required to support simultaneous operation.

Figure 3-5 illustrates some basic examples of supported SuperSpeed USB physical device topologies including one where a USB 2.0-only peripheral device is incorporated into the physical device based on a compound device implementation. More complex topologies are possible as long as they adhere to the operational requirements at the device and hub unit levels.

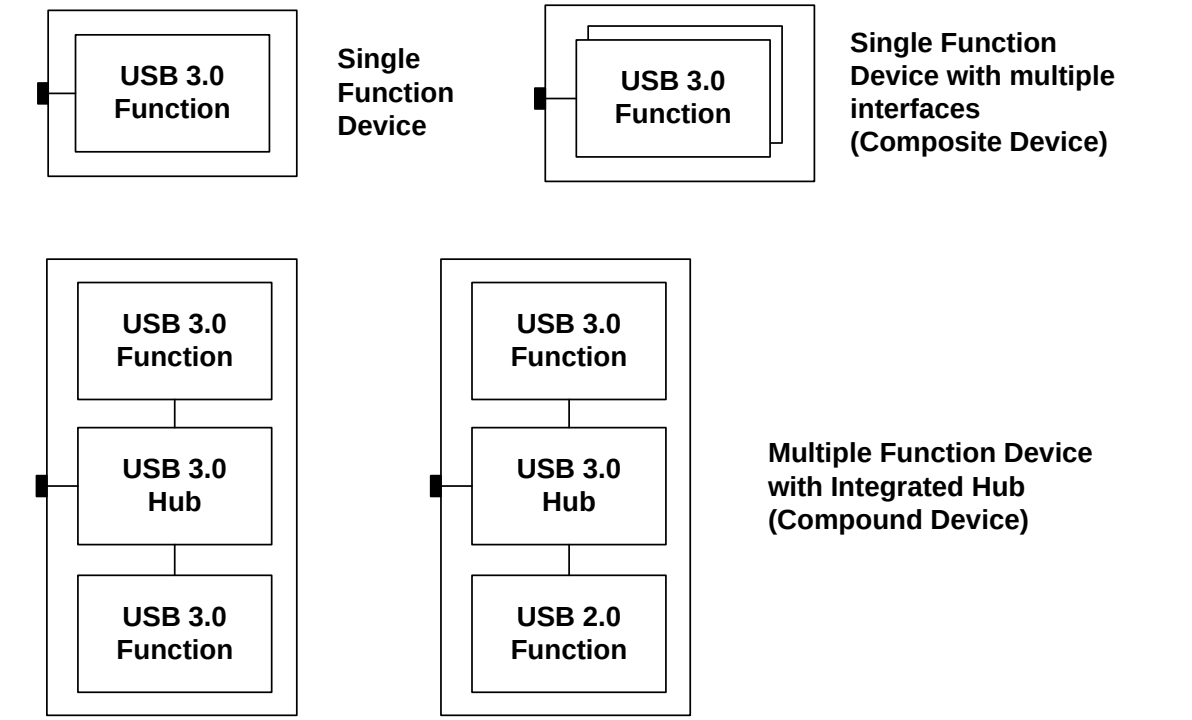


Figure 3-5. Examples of Supported SuperSpeed USB Physical Device Topologies

3.2.3 Bus Protocol

The SuperSpeed USB is a unicast rather than a broadcast bus. The Host Controller initiates all data transfers. Data transfers flow only on the direct route between the host and the target device. If an endpoint cannot complete the request, it tells the host it is not ready and the host moves on to service another endpoint. When the device becomes ready, it asynchronously sends a notification to the host that it is now ready to complete the transaction. The host responds by rescheduling that transaction.

Most bus transactions involve the transmission of two packets. Each OUT transaction begins when the Host Controllers sends a USB data packet to the device with explicit routing information in the data packet header. The device responds with an ACK for the data, a not ready response, or a STALL. Alternatively, each IN transaction begins when the Host Controllers sends an ACK to the device with a request for data along with explicit routing information in the packet header. The device responds with a data packet, a not ready response, or a STALL. The header of the data

packet and ACK describe the type and direction of transaction, the USB device address, and endpoint number.

The USB data transfer model between a source or destination on the host and an endpoint on a device is referred to as a pipe. There are two types of pipes: stream and message. Stream data has no USB-defined structure, while message data does. Additionally, pipes have associations of data bandwidth, transfer service type, and endpoint characteristics like directionality and buffer sizes. Most pipes come into existence when a USB device is configured. One message pipe, the Default Control Pipe, always exists once a device is powered, in order to provide access to the device's configuration, status, and control information.

The transaction schedule allows flow control for some stream pipe types. At the hardware level, this prevents buffers from underrun or overrun situations by using a Not Ready / Endpoint Ready handshake to throttle the data rate. In response to an Endpoint Ready, a transaction is retried when bus time is available. The flow control mechanism permits the construction of flexible schedules that accommodate concurrent servicing of a heterogeneous mix of stream pipes. Thus, multiple stream pipes can be serviced at different intervals and with packets of different sizes.

3.2.4 Robustness

There are several attributes of SuperSpeed USB that contribute to its robustness:

- Signal integrity using differential drivers, receivers, and shielding
- CRC protection for header and data packets
- Link level header packet retries that ensure their reliable delivery
- Detection of attach and detach and system-level configuration of resources
- Data and control pipe constructs for ensuring independence from adverse interactions between functions

3.2.4.1 Error Detection

The physical link bit error rate of the USB SuperSpeed physical medium is expected to be less than one in 10^{12} bits. To provide protection against occasional bit errors, framing and link commands are repeated. Each packet includes a CRC to provide error detection of multiple bit errors. When data integrity is required, such as with lossless data devices, an error recovery procedure may be invoked in hardware or software.

The protocol includes separate CRCs header and data packet. Additionally a link control word is appended to every header packer which has its own CRC. A failed CRC in the header or link control word is considered a serious error which will result in a link level retry to recover from the error. A failed CRC in a data payload is considered to indicate a corrupted data and is handled by the protocol layer with a request to resend the data packet.

3.2.4.2 Error Handling

The protocol allows for error handling in hardware or software. Hardware error handling includes reporting and retry of failed header packets. A USB Host Controller will try a transmission that encounters errors up to three times before informing the client software of the failure. The client software can recover in an implementation-specific way.

3.3 USB Specification Chapter Overview

The USB 3.0 Specification core chapters cover the functional definition of the SuperSpeed USB. The specification is organized as a series of layers as illustrated by Figure 3-6. All of these layers are implemented by both USB hosts and peripherals. For those familiar with the USB 2.0 specification, some aspects of this organization and the specifications within will be more familiar than others. The major differences from USB 2.0 are in the link and physical layers where the specifications have been realigned toward technology specifications that offer multi-gigahertz signaling and simultaneous bi-directional control and data flow.

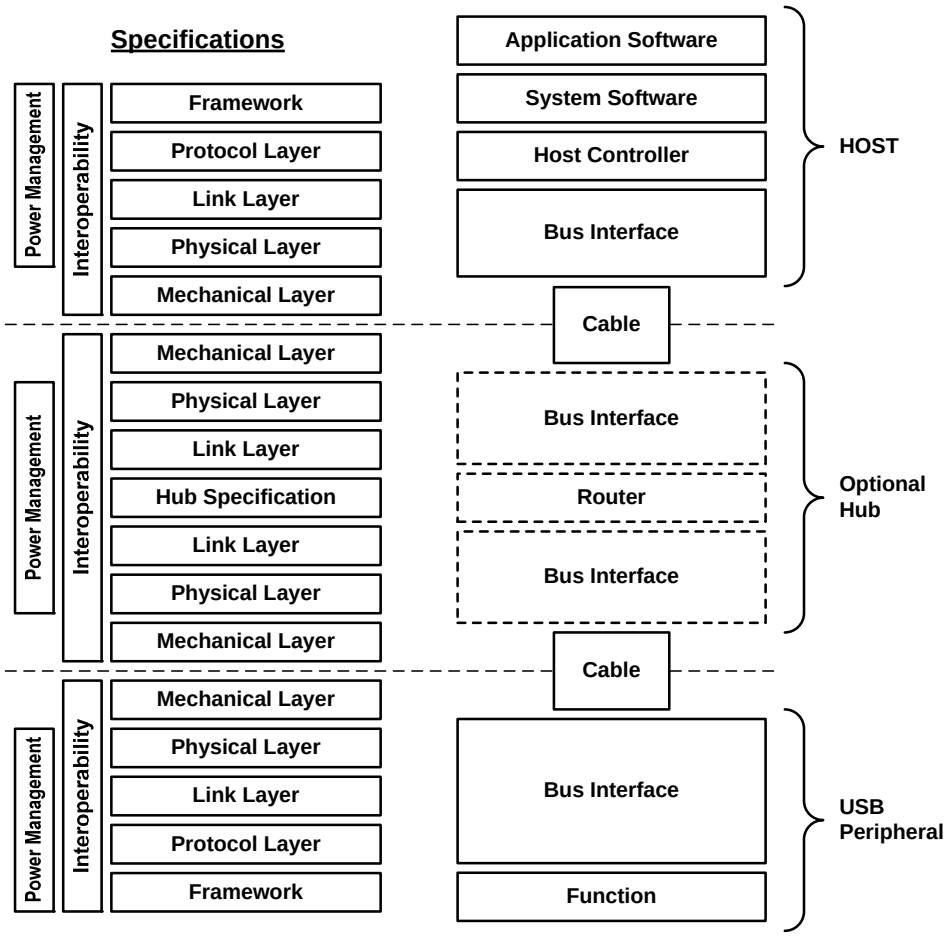


Figure 3-6. SuperSpeed USB Interface and Protocol Layers

SuperSpeed USB is constructed starting with the mechanical layer which defines the requirements for the cables and connectors used to connect hosts with hubs and peripherals. The physical layer adds the lowest-level electrical and signaling requirements for robustly transmitting control and data bits formed as a set of specially defined coding symbols across the mechanical interconnect. The link layer manages setting up and maintaining a logical connection between the ports on each end of the cable. The protocol layer is responsible for organizing bus management and data content into packets and for controlling the transfer of these packets across the bus. The framework

provides a structured way to enable host software to configure and manage hubs and peripherals at the device level, and as needed at the individual port level.

Many aspects of SuperSpeed USB operation spread across multiple layers. The features that make the USB SuperSpeed bus a robust and reliable communications link spread from the mechanical layer and all the way up to the framework. Similarly, power management features require support across most of the functional levels. See the Power Management Appendix for more detailed information.

3.3.1 Mechanical

This specification defines USB 3.0-specific host and device side cable connectors, and a series of USB 3.0-specific standardized cable assemblies. The USB 3.0 Standard-A connector is the principle means of connecting USB devices directly to a host or to the downstream port of a hub.

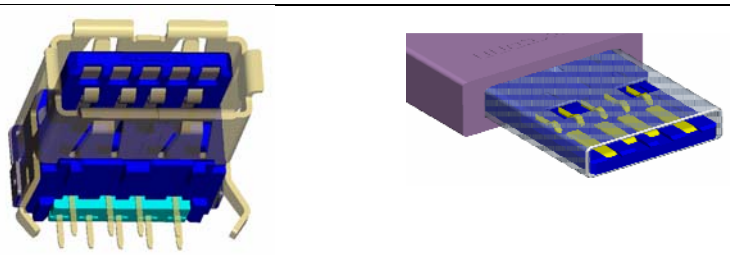


Figure 3-7. USB 3.0 Standard-A Cable Connector and Host Receptacle

The device-side connection is either detachable using the USB 3.0 Standard B, Powered B or Micro-B connector or is captively connected (hardwired) into the device. The USB 3.0 version of the Standard B and Micro-B receptacles are compatible and interoperable with their USB 2.0 B counterparts. To support USB On-the-Go functionality, the specification also defines USB 3.0 Micro-A connector and USB 3.0 Micro-AB receptacle variants; these are essentially the same as USB Micro-B versions but with additional electrical and mechanical keying features.

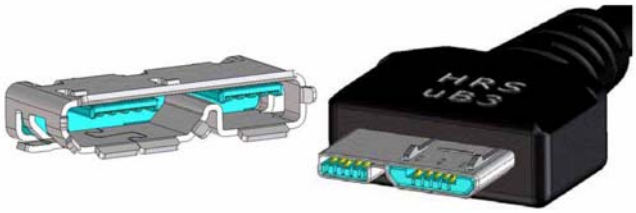


Figure 3-8. USB 3.0 Micro-B Cable Connector and Device Receptacle

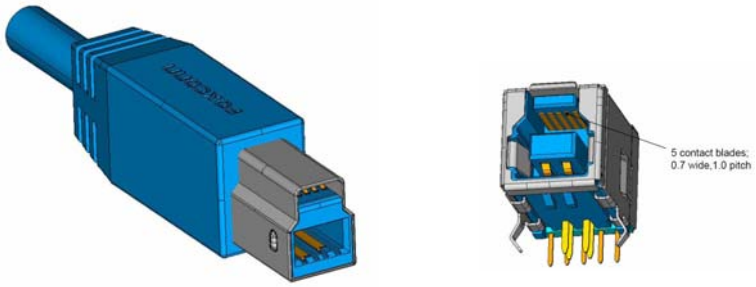


Figure 3-9. USB 3.0 Standard B Cable Connector and Device Receptacle

USB 3.0 cables have eight primary conductors: three twisted signal pairs for USB data paths and a power pair. Figure 3-10 illustrates the basic signal arrangement for the USB 3.0 cable. In addition to the twisted signal pair for USB 2.0 data path, two twisted signal pairs are used to provide the SuperSpeed data path, one for the transmit path and one for the receive path.

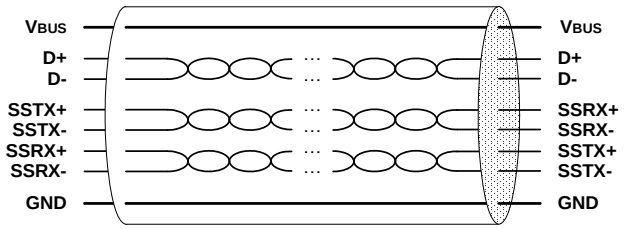


Figure 3-10. USB 3.0 Cable

To enable interoperability with existing USB 2.0 devices, the USB 3.0 Standard-A host receptacles are specified to be backward compatible with USB 2.0 Series "A" cable connectors. To enable interoperability of USB 3.0 devices with USB 2.0 hosts and hubs, the USB 3.0 Standard-A cable connectors are specified to be backward compatible with USB 2.0 Series "A" host receptacles. USB 3.0 cables are not intended to be compatible with USB 2.0 devices.

To aid in recognizing USB 3.0-specific connectors and receptacles, a standardized blue color (Pantone 300C) coding is specified for the plastic portions of the connector and receptacle assemblies.

Electrical (insertion loss, return loss, crosstalk, etc.) performance for USB 3.0 are defined with regard to raw cables, mated connectors and mated cable assemblies; with compliance requirements using industry test specifications established for the latter two categories. Similarly, mechanical (insertion/extraction forces, durability, etc.) and environmental (temperature life, mixed flowing gas, etc.) requirements are defined and compliance established via recognized industry test specifications.

Specification requirements for cables and connectors are provided in the Mechanical Layer Chapter.

3.3.2 Physical Layer

The physical layer of USB 3.0 defines the electrical and signaling requirements for the SuperSpeed bus.

The electrical aspects of each SuperSpeed link in the bus are characterized as a transmitter, channel and receiver; these collectively representing a uni-directional differential link. Each differential link is AC-coupled with capacitors located within the channel on the transmitter side of the connector. The channel also includes consideration for the electrical characteristics of the cables and connectors.

At an electrical level, the SuperSpeed bus connection is initialized by the enabling of receiver termination. The transmitter is responsible for remotely detecting this termination as an indication of a bus connection and informing the link layer so the connect status can be factored into link operation and management. The link goes to U0 (logical idle) automatically on initial connect.

When receiver termination is present but no signaling is occurring on the link, that link is considered to be in electrical idle. When the link is electrically idle, link reset and PM state exit/wake events are signaled across the link using a low-frequency, periodic signal. The interpretation of this signal by the receiver is determined based on the duty cycle and the width of the active period of the signal.

To assist in reducing EMI impact, the transmitter clock source has spread spectrum modulation applied to it.

To signal data and control across the SuperSpeed bus, an ANSI-standard 8b/10b transmission code is used with bits organized as characters and then being encoded as symbols. Control symbols are used for framing data and managing the link, and they special characteristics such that they are uniquely identifiable from data symbols.

Each port on the SuperSpeed bus is expected to have a reference clock domain with Spread Spectrum Clocking (SSC) modulation. Since USB 3.0 does not include a reference clock signal pair in the cable, the reference clock domains on each end of the link are not explicitly linked.

Bit-level timing synchronization relies on each receiver being able to reasonably align its local reference bit recovery clock to that of the transmitter's independent local reference clock by phase-locking to the signal transitions in the received bit stream. To assure that adequate transitions for the receiver to track occur in the bit stream independent of the data content being transmitted, the transmitter applies a scrambler function. Control symbols are defined such that transition balance is inherent in the symbol sequences and therefore are not scrambled.

A number of techniques are employed to improve channel performance. For example, to avoid overdriving and improve eye margin at the receiver, transmitter de-emphasis is applied when multiple bits of the same polarity are sent. Also, channel equalization is applied in the receiver with the characteristics of the equalization profile being established adaptively as part of the link training sequence.

Signal (timing, jitter tolerance, etc.) and electrical (DC characteristics, channel capacitance, etc.) performance of SuperSpeed links are defined with compliance requirements specified in terms of transmit and receive signaling eyes.

Specification requirements for the physical layer are provided in Physical Layer Chapter.

3.3.3 Link Layer

The link layer of USB 3.0 is responsible for initializing and maintaining SuperSpeed link-level communications, providing mechanisms to low-level control the flow of packets on the link, and managing the link-level operational modes including entry/exit of power management states.

Initializing a USB bus link starts with detecting when devices are connected across the USB cable. USB SuperSpeed connections involve providing local and detecting remote receiver termination of the two ports that are being connected. The connection process then proceeds to training the link by exchanging a series of special bit streams designed to allow for bit and symbol synchronization.

Similar to USB 2.0, clock synchronization across the connected link is a derivative by-product of closely tracking signal transitions on the bus but unlike USB High-Speed, SuperSpeed clocking has to be maintained to a significantly tighter tolerance and is complicated by the need to support Spread Spectrum clocking for reducing bus-generated EMI. To aid in adapting for clock drift between the two ports, special compensation symbols are inserted between data transmissions so that data buffer overflow is mitigated in cases where the sending port is operating faster than the receiving port.

To manage the low-level connection, the link layer contains a state machine that defines modal states not only for connecting and training the link, but also for the Ux lower-power idle states, compliance testing states and transitional states necessary for transitioning between idle and active states. State machines make use of physical layer-defined control symbols and symbol sequences to coordinate with the state machine in its link partner. In some cases such as port configuration it is necessary to exchange data between the two port state machines and for this a special message is defined referred to as a Link Management Packet (LMP).

Finally, the link layer defines port-to-port control messages known as link commands which are used to provide low-level flow control of data transfers.

3.3.4 Protocol Layer

The SuperSpeed USB protocol is architected to take advantage of the new dual simplex physical layer. SuperSpeed USB transactions consist of two parts, for an IN, a transaction is made up of a token and data. The token includes the handshake to the previous data packet. For an OUT, a transaction is made up of data and a handshake. The data packet includes the token. The dual simplex physical layer also allows the protocol to support concurrent in and out transactions.

Reducing active power consumption and idle power are primary goals of SuperSpeed USB. Entry into reduced link power states is typically handled by hardware to provide more rapid entry into lower power states than can be achieved by software control; however, software has input into the link power management policies. These link power transitions may be initiated by either the upstream port or the downstream port. The protocol takes the link power management into account to improve performance by introducing the notion of deferred response which is effectively a Not Ready response from a hub allowing the host to defer a transfer delayed by a power managed link in favor of other pending transfers.

SuperSpeed USB is architected to maintain device software compatibility. Care was taken in the definition of the protocol so that at the Framework Layer, vendor specific device drivers and class device drivers would continue to work preserving the existing software investment.

3.3.5 Framework Layer

The framework defined in the USB 3.0 specification builds upon the "USB Device Framework" defined in Chapter 9 of the USB 2.0 Specification. This approach was taken for two reasons:

- SuperSpeed devices must operate as High-speed devices when connected to a USB 2.0 host
- To preserve backward compatibility of software, so that existing class drivers continue to work with SuperSpeed devices

To support the new features defined in SuperSpeed USB new standard commands have been added to support these features and some of the existing standard commands have been extended. The standard commands that SuperSpeed devices need to support are detailed in Framework chapter.

- The SuperSpeed USB framework describes the common attributes and operations to both control the operation of the device as well as to move data between the bus interface and various endpoints on the device. These attributes and operations are used by the function-specific portions of the device to communicate through the bus interface and ultimately with the host.

3.3.6 Hubs

Hubs are an important element in the USB architecture. Hubs simplify USB connectivity for end users by allowing for port expansion and flexibility in port location yet with a reasonably low cost/complexity solution. Each hub has a single upstream port that connects to the downstream port of another hub or a host. USB 3.0 hubs have up to 15 downstream ports that provide additional attachment points in the system. Hubs or peripheral devices can be connected to the downstream ports of a hub. The USB architecture supports up to 5 concatenated hubs between a host port and a device. Figure 3-11 shows an example USB topology with three levels of 4-port hubs. This example topology provides attach points for up to 64 USB devices with 21 4-port hubs. The maximum number of devices supported for any USB topology is 127.

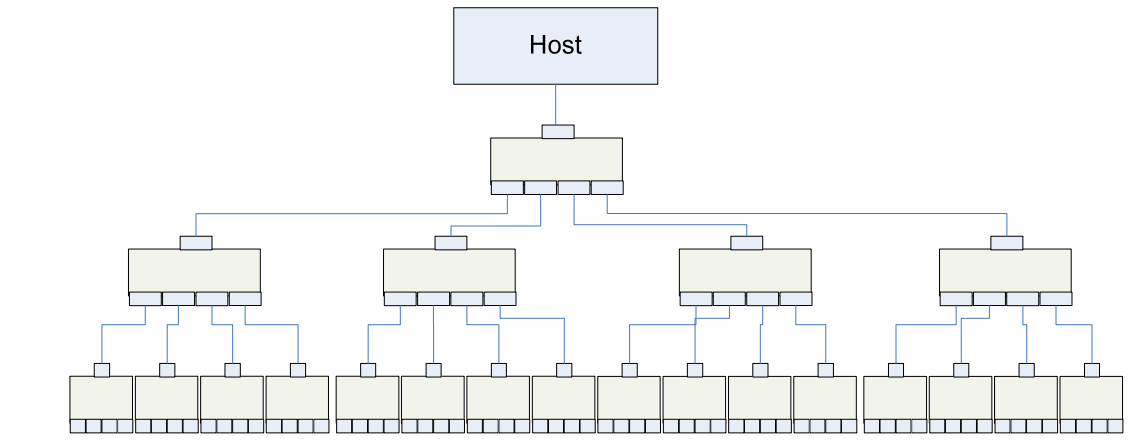


Figure 3-11. Bus Expansion Example - Three Levels Of 4-Port Hubs

Hubs can detect attachment and detachment of devices at each downstream port and enable/disable power distribution to the downstream ports. Each downstream port can be individually enabled and attached to either SuperSpeed, high-speed, full-speed, or low-speed devices.

3.3.6.1 Hub Architecture

A USB 3.0 hub is the logical combination of two hubs: a USB 2.0 hub and a SuperSpeed hub. Each hub operates on a separate data bus as shown in Figure 3-12. The only signal shared between the hubs is VBUS. If either the USB 2.0 hub or SuperSpeed hub controllers requires it, a downstream port must be powered. A USB 3.0 hub connects upstream as both a SuperSpeed and USB 2.0 hub.

A SuperSpeed hub consists of two logical components: A SuperSpeed hub controller, and a SuperSpeed repeater/forwarder. The hub repeater/forwarder is a protocol controlled router between the SuperSpeed hub upstream and downstream ports. It also has hardware support for reset and suspend/resume signaling. The host controller provides communication to/from the host. A hub does not initiate any data transfers. Hub-specific status/control commands permit a host to configure a hub and to monitor and control its ports. The USB 2.0 hub component provides the mechanisms that support high-speed, full-speed, and low-speed devices behind the hub. The USB 2.0 hub component requirements are documented in the *Universal Serial Bus Specification, Revision 2.0*.

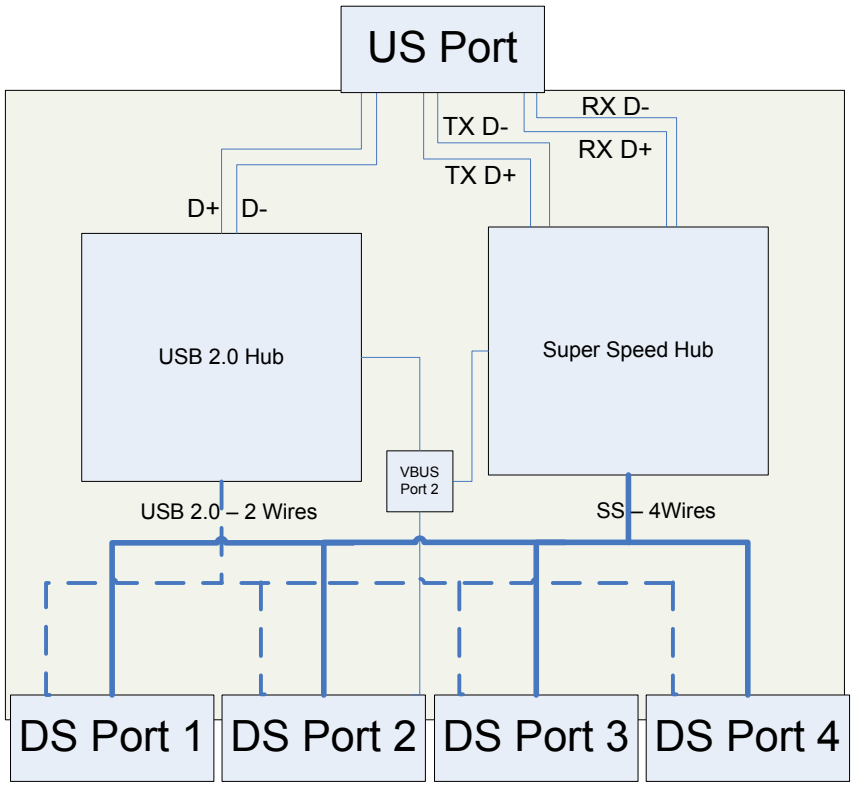


Figure 3-12. High Level Hub Architecture

3.3.6.2 Hub Repeater/Forwarder Architecture

A SuperSpeed hub contains buffering for header and data packets. A SuperSpeed hub does not use the repeater only model used for high speed connectivity in a USB 2.0 hub. This change allows multiple downstream SuperSpeed devices to send asynchronous messages simultaneously without data loss and for some traffic to be stored and delivered when it is directed to downstream ports that are in low power link states. A SuperSpeed hub treats repeater/forwarder treats data and other types of packets differently. Protocol headers are buffered and retried such that header packet loss is extremely unlikely to occur at the link level in the system. Limited buffering is also provided for data packets, such that cases where data is dropped by a link and must retried from the host are rare. The following sections provide a more detailed overview of the hub header and data buffering architectures.

3.3.6.3 Hubs and Transfers to Power Managed Links

The USB 3.0 architecture allows devices to initiate link power management transitions at any time. Therefore, scenarios can exist where the hub receives traffic that is directed at a downstream link that it is in a low power state. In these scenarios the hub discards any data associated with the transaction and modifies the header to indicate the transaction was deferred. The hub sends notification of the deferred transaction back to the host and begins to wakeup the downstream link. When the downstream link is awake the deferred header is delivered. When the device is ready to respond to the deferred transaction it shall sent an ERDY TP to the host.Hubs and Isochronous Transactions

Hubs play an important role in the ability to offer isochronous transfers with any topology. Isochronous timestamps are delivered from the host to devices with isochronous endpoints and allow the isochronous devices to synchronize with the host. Limiting the variation in propagation delay for header packets containing isochronous timestamps is an important hub function. Hubs quickly forward headers downstream when there is no activity with minimal delay variation. In rare cases where other bus traffic delays downstream header delivery, the hubs modify the headers to indicate that they were delayed. This allows isochronous devices to ignore delayed headers for synchronization purposes. Isochronous devices are able to synchronize to the host clock within 200 nanoseconds.

3.3.7 Performance and Power Efficiency

SuperSpeed performance and power efficiency significantly benefits from the simultaneous, bi-directional traffic flows enabled by the USB 3.0 dual-simplex bus architecture. SuperSpeed packets are directly routed rather than broadcast eliminating the need for devices to packet filter. The SuperSpeed protocol incorporates support for power management at the individual link level. When combined with improved power per bit transferred efficiency, SuperSpeed USB performs better even at transfer bandwidths still supportable by USB 2.0 High-Speed.

3.3.8 SuperSpeed Power Management

USB SuperSpeed power management cuts through all layers of the USB 3.0 specification. This section summarized the salient features as they affect the USB SuperSpeed architecture.

The key elements that makeup SuperSpeed PM are:

- Devices send asynchronous busy / ready notifications to the host rather than the host polling devices.
- Packets are unicast (e.g. routed), not broadcast, allowing links that aren't needed to remain in a low power state.
- There are four host or device driven link states with progressively lower power and greater exit latency.
- Function level suspend that allows devices to remove power from portions of their circuitry that are currently not being used.

3.3.8.1 Link Power Management

Links may be placed into low power states (referred to as "U-states" or "Link-states") while they are idle. Deeper power savings can be achieved, progressing from U0 to U1, to U2, and finally to U3 as the link's activity decreases.

Table 3-2. Link PM States and Characteristics Summary

Link State	Description	Characteristics	H/W and/or S/W Directed	Device Clock Gen On/Off	Exit Latency Range
U0	Link active	Link operational state	N/A	On	N/A
U1	Link idle – fast exit	Rx & Tx circuitry quiesced	H/W and S/W	On or Off	µs
U2	Link idle – slow exit	Clock generation circuitry may additionally be quiesced	H/W and S/W	On/Off	µs – ms
U3	Link suspend	Interface (e.g. Physical Layer) power may be removed	Entry: S/W only Exit: H/W and S/W	Off	µs – ms

3.3.8.1.1 U1, U2, and U3 Entry and Exit Processes

U1 and U2 entry may be initiated by either the device or the port it is attached to (root port or hub downstream port). Devices may initiate entry as soon as they determine the link is idle or isn't needed in the near future. Root ports and hub downstream ports may initiate entry after an inactivity timer expires. Link commands are used to transition the link from U0 to U1 or U2. U1 and U2 are exited automatically by either port on a link when there is traffic. The exit process to transition back to U0 consists of a Low Frequency Periodic Signaling (LFPS) handshake followed by link Recovery and training.

U3 (suspend) minimizes power consumption by allowing devices to remove power from circuitry unneeded during suspend. U3 entry is initiated by a downstream port as part of the port suspend process. The host initiates U3 exit by issuing a port command to the downstream port. Devices may initiate U3 exit when the device detects a wakeup event on one of the functions of the device and the host has enabled remote wake-up on that function. The transition back to U0 uses a Low Frequency Periodic Signaling (LFPS) handshake followed by link Recovery and training. When

initiated by a device, it sends a Function Wake packet to the host after the link is restored. The Function Wake message causes an interrupt to software.

3.3.8.1.2 Hub Support for Link PM

Hubs play the following roles in link PM:

- Coordinates the upstream port link PM state with that of its downstream ports.
- Handles packet deferring, where it tells the host that a packet was sent to a downstream port not in U0.

Hubs monitor their downstream port link PM states and adjust their upstream port link PM state to be no higher than the highest downstream port link PM state. This behavior allows a device to control the lowest link PM state of all the links between itself and the host. For example, if the device keeps its link in U0, then all links in the path between the device and host stay in U0.

When the host sends a packet to a device connected to a hub and that device's link is not in U0, the hub responds by sending the host a deferred command and also forwards the packet header to the device with the deferred bit asserted. This helps improve bus performance by eliminating the possible substantial time the host would normally need to wait for a response from the device. The host treats deferred commands similar to having received a not ready (NRDY). When the device receives an IN or an OUT header with the deferred bit set, it prepares for the transfer and when ready sends an endpoint ready (ERDY) message to the host which responds by rescheduling the transfer.

3.3.8.1.3 Other System Support for Link PM

Devices may use the Transfer Pending flag as an indication of pending transfers on the host controller schedule associated with a given endpoint to help the device decide when to place its link in a low power state.

If a link is in a low power state when an isochronous transfer is scheduled, the link exit latency could delay the transfer beyond its isochronous contract. The ping, ping response protocol allows the host to transition all links in the direct path to the device to U0 prior to initiating an isochronous transfer.

3.3.8.2 Function and Device Power Management

The function suspend state is a low power state associated with an individual function which may or may not be part of a composite device. A suspended function may be enabled to do a remote wakeup.

When a remote wakeup event occurs and the device's link is not in U0, then the link is transitioned to U0. Once the link is in U0 (or if it was already in U0) the function sends a Function Wake message to the host. The Function Wake message is used by software to determine which function has transitioned out of the suspended state.

Device suspend is a device-wide state entered and exited intrinsically as a result of a device's upstream port entering and exiting the U3 state.

3.3.8.3 Platform PM Support

LTM allows platforms to make dynamic power vs. performance tradeoffs cooperatively with devices without imposing additional cost. USB devices optionally use the Latency Tolerance Message (LTM) notification containing a Best Effort Latency Tolerance (BELT) value to report how much latency they can tolerate from the system only for best effort services such as bulk. LTM is not used where a guaranteed latency or bandwidth service is needed such as for periodic service.

LTM provides information to the system that may allow it to enter deeper power saving states than would otherwise be reliable. However, this is only possible when all of the devices in the system (not just USB devices) allow it. If any device in the system needs to operate in a high-performance mode at all times, it can require the system to stay in a high state of readiness to service that device with best effort low latency. However, this may incur a substantial cost in terms of platform power and battery life for mobile systems.

4. SuperSpeed Data Flow Model

This chapter presents high-level information on how data and information moves across the SuperSpeed USB. The information in this chapter affects all implementers. Consult the Protocol Chapter for details on the low-level protocol. This chapter provides framework overview information that is further expanded in the Framework Chapter. All implementers should read this chapter so they understand the key concepts of SuperSpeed USB.

4.1 Implementer Viewpoints

SuperSpeed USB is very similar to USB 2.0 in that it provides communication services between a USB Host and attached USB Devices. The communication model view preserves the USB 2.0 layered architecture and basic components of the communication flow (i.e., point-to-point, same transfer types, etc. See Chapter 5 in the Universal Serial Bus Specification Revision 2.0).

This chapter describes the differences (from USB 2.0) of how data and control information is communicated between a SuperSpeed USB Host and its attached SuperSpeed USB Devices. In order to understand SuperSpeed USB data flow, the following concepts are useful:

- Communication Flow Models: Section 4.2 described how communication flows between the host and devices through the SuperSpeed USB bus.
- SuperSpeed USB Protocol Overview: Section 4.3 gives a high level overview of the SuperSpeed USB Protocol and its differences as compared to the USB 2.0 protocol.
- Generalized Transfer Description: Section 4.4 provides an overview of how data transfers work in SuperSpeed USB and subsequent sections define the operating constraints for each transfer type.
- Device Notifications: Section 4.4.9 provides an overview of Device Notifications, a feature which allows a device to asynchronously notify its host of events or status on the device.
- Media Reliability and Efficiency: Sections 4.4.10 and 4.4.11 summarizes the information and mechanisms available in SuperSpeed USB.

4.2 SuperSpeed USB Communication Flow

SuperSpeed USB retains the familiar concepts and mechanisms and support for Endpoints, Pipes, and Transfer types, please refer to the USB Specification, Revision 2.0 for details. As in USB 2.0 the ultimate consumer/producer of data is an Endpoint.

The endpoint's characteristics (Max Packet Size, Burst Size etc.) are reported in the endpoint descriptor and the SuperSpeed Endpoint Companion Descriptor and as in USB 2.0 the endpoint is identified using an addressing triple {DeviceAddress, Endpoint Number, Direction}.

All SuperSpeed USB devices must implement at least the Default Control Pipe (Endpoint zero). The Default Control Pipe is a Control pipe as defined in the USB 2.0 specification.

4.2.1 Pipes

A SuperSpeed USB pipe is an association between an endpoint on a device and software on the host. Pipes represent the ability to move data between software on the host via a memory buffer and an endpoint on a device and have the same behavior as defined in the USB 2.0 Specification. The main difference is that when a non Isochronous endpoint in SuperSpeed USB is busy it returns a Not Ready (NRDY) response and must send an Endpoint Ready (ERDY) notification when it wants to be serviced again. The host will then reschedule the transaction at the next available opportunity within the constraints of the transfer type.

4.3 SuperSpeed USB Protocol Overview

As mentioned in the Architecture Overview Chapter, the SuperSpeed USB protocol is architected to take advantage of the new dual simplex physical layer. All the USB 2.0 transfer types are supported by the SuperSpeed USB protocol. The differences between the USB 2.0 protocol and the SuperSpeed USB protocol are first discussed followed by a brief description of the packets used in SuperSpeed USB.

4.3.1 Differences with USB 2.0

SuperSpeed USB is backward compatible with USB 2.0 at the framework level. However there are some fundamental differences between the SuperSpeed USB protocol and USB 2.0:

- USB 2.0 uses a three-part transaction (Token, Data and Handshake) while SuperSpeed USB uses the same three parts differently. For OUTs the token is incorporated in the data packet while for INs the Token is replaced by a handshake.
- USB 2.0 limits bursts to a fixed number of packets while SuperSpeed USB supports continuous bursting.
- USB 2.0 is broadcast half-duplex bus while SuperSpeed USB is dual-simplex unicast bus.
- USB 2.0 uses a polling model while SuperSpeed USB uses asynchronous notifications.

4.3.1.1 Comparing USB 2.0 and SuperSpeed USB Transactions

Figure 4-1 shows a USB 2.0 style transfer mapped to a SuperSpeed USB transfer. The dual-simplex physical layer allows information to travel simultaneously in both directions. The SuperSpeed USB protocol takes advantage by allowing the handshake for one data packet to overlap the next. For OUT transfers, the USB 2.0 Token is incorporated in the data packet header so a separate Token is not required. For IN transfers, a handshake sent to the device to request data. The device may respond by either returning data, returning a STALL handshake, or by returning a Not Ready (NRDY) handshake to defer the transfer until the device is ready. The ability to simultaneously send traffic in both directions as can be seen in Figure 4-1 where the handshake in response to the OUT data and the handshake requesting IN data from the device overlap.

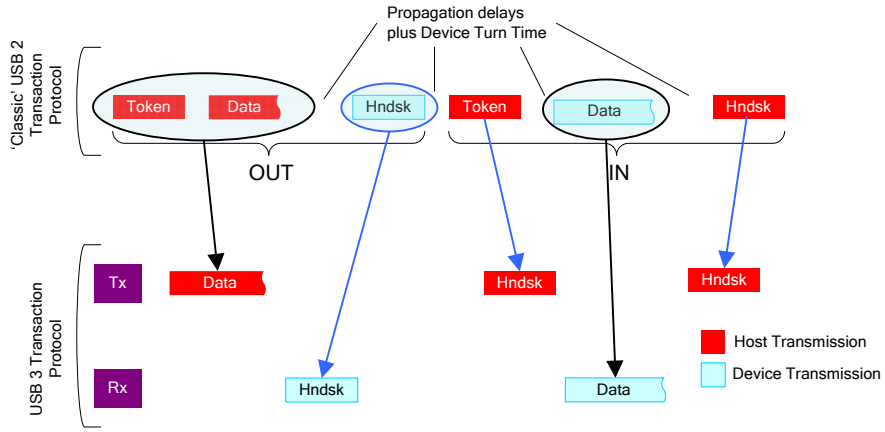


Figure 4-1. USB 2.0 style Transfer Mapped onto SuperSpeed USB Transfer

4.3.1.2 Comparing USB 2.0 and SuperSpeed USB Bus Topology

The USB 2.0 broadcasts packets to all enabled downstream ports. Every device is required to decode the address triple (device address, endpoint and direction) of each packet to determine if it needs to respond. SuperSpeed USB unicasts the packets - downstream packets are sent over a directed path between the host and the targeted device while upstream packets are sent over the direct path between the device and the host. SuperSpeed USB packets contain routing information that the hubs use to determine which downstream port the packet needs to traverse to reach the device. Unicasting the packets allows for improved link power management as the only links used to forward a packet need to be in the active power state.

USB 2.0 style polling has been replaced with asynchronous notifications. The SuperSpeed USB transaction is initiated by the host making a request followed by a response from the device. If the device can honor the request, it either accepts or sends data. If the endpoint is halted, the device shall respond with a STALL handshake. If it cannot honor the request due to lack of buffer space or data, it responds with a Not Ready (NRDY) to tell the host that it is not able to process the request at this time. The device then at a later time, when it can honor the request, sends an Endpoint Ready (ERDY) to the host which will then reschedule the transaction.

The move to unicasting the packets together with asynchronous notifications allow links that are not actively passing packets to be put into reduced power states. Upstream and downstream ports cooperate to place their link into a reduced power state that hubs will propagate upstream. Allowing link partners to control their independent link power state and a hub's propagating the highest link power state seen on any of its downstream ports to its upstream port, puts the bus into the lowest allowable power state rapidly.

4.3.1.3 Introduction to SuperSpeed USB Packets

The SuperSpeed USB protocol is based on packets. All packets start with a 16-byte header. Some packets consist of a header only. The header is structured to minimize the amount of processing needed in order to maximize performance. All headers begin with the Revision and Packet Type information used by the receiving entity (e.g., host, hub or device) to decide how to handle the packet. The header is protected by a 16-bit CRC (CRC-16) and ends with a 2- byte link control word. Depending on the Type, most packets contain routing information (Route String) and a device address triple (device address, endpoint number and direction). The Route String is used to direct packets sent by the host in a directed fashion through the topology. Packets sent by the device are implicitly routed as the hub always forwards a packet seen on any downstream port to its upstream port. There are four basic types of packets: Link Management Packets, Transaction Packets , Data Packets and Isochronous Timestamp Packets:

- A Link Management Packet (LMP) only traverses a pair of directly connected ports and is primarily used to manage that link.
- A Transaction Packet (TP) traverses all the links in the path directly connecting the host and a device. It is used to control the flow of data packets, configure devices and hubs etc. Note a Transaction Packet does not have a data payload.
- A Data Packet (DP) traverses all the links in the path directly connecting the host and a device. Data Packets consist of two parts: a Data Packet Header (DPH) which is similar to a TP and a Data Packet Payload (DPP) which consists of the data block plus a 32-bit CRC (CRC-32) used to ensure the data's integrity.
- Isochronous Timestamp Packet (ITP) is multicast packet sent by the host and forwarded to all the links in U0 from the host to one or more devices**.**

4.4 Generalized Transfer Description

Each non-isochronous data packet sent to a receiver is acknowledged by a handshake (called an ACK transaction packet), however due to the fact that SuperSpeed USB has independent transmit and receive paths, the transmitter does not have to wait for an explicit handshake for each data packet transferred before sending the next packet.

SuperSpeed USB preserves all of the basic Data Flow and Transfer concepts defined in USB 2.0, including the Transfer Types, Pipes and basic data flow model. The differences with USB 2.0 discussed in this section, starting with description of differences at the protocol level, then the differences in transfer type constraints.

The USB 2.0 specification utilizes a serial transaction model. This essentially means that a host starts and completes one bus 'transaction' {Token, Data, Handshake}, before starting the next transaction. 'Split' transactions also adhere to this same model as they are comprised of complete high-speed transactions {Token, Data, Handshake} that are completed under the same model as all other transactions.

SuperSpeed USB maps the USB 2.0 transaction protocol but optimizes the transactions by making use of the fact that SuperSpeed USB uses independent transmit and receive paths.

The USB 2.0 protocol completes an entire IN or OUT transaction (Token, Data and Handshake phases) before continuing to the next bus transaction for the next scheduled function endpoint. All transmissions from the host are essentially broadcast on the USB 2.0 bus. In contrast, the

SuperSpeed USB protocol does not broadcast any packets (except for ITPs which are multicast) and packets traverse only the links needed to reach the intended recipient. The host starts all transactions and devices respond either with data or acknowledgements. If the device cannot accept the data or does not have data available then it responds with a packet that states that it is not able to do so. Subsequently when the device is ready to either receive or transmit data it sends a notification to the host that indicates that it is ready to resume transactions. In addition SuperSpeed USB also provides the ability to transition links into and out of specific low power states. Lower power Link states are entered either under software control or under autonomous hardware control after being enabled by software. Mechanisms are provided for the host to initiate a transition from a non-U0 to U0 for all links in the path to a device. A SuperSpeed USB Host determines when to start transactions on the bus in order to satisfy the needs (and priorities) of the applications controlling the devices on the SuperSpeed USB bus.

Endpoint maximum packet sizes, exposed to the host via the endpoint descriptor, in this specification indicate 'application' data payloads only. They do not include any of the overhead for link and protocol level. See Section 9.6.6 for details. In addition, bandwidth allocation for SuperSpeed USB is very similar to wired USB 2.0. Details of how the basic transfer types are implemented in SuperSpeed USB can be found in the Protocol Chapter.

4.4.1 Data Bursting

The USB 2.0 protocol allows a maximum of three data packets per USB transaction for high-bandwidth endpoints and a maximum of one data packet per USB transaction for all other transactions. However as the bus speed increases, waiting for individual packet acknowledgements adds unacceptable turn-around time delays. In addition, since the SuperSpeed USB bus has independent transmit and receive paths, a device can continuously send/receive data to/from the host and acknowledgements for the same are transmitted asynchronously by the receiver to the transmitter. This new capability of SuperSpeed USB is referred to as Data Bursting and allows for better efficiency because it eliminates the need for the turn around time while waiting for acknowledgements for data packets transmitted.

Each endpoint on a SuperSpeed USB device indicates the number of packets that it can send/receive before it has to wait for an explicit handshake. Maximum data burst size is an individual endpoint capability which depends on the endpoint's current configuration. A host determines an endpoint's maximum data burst size from the SuperSpeed USB Endpoint Companion descriptor associated with this endpoint (see Section 9.6.7). The size of each data packet in a data burst must be the configured endpoint's maximum packet size except for the last which can be any size up to and including the endpoint's maximum packet size. The host may dynamically change the burst size on a per-transaction basis (up to the configured maximum burst size).

The host may use any burst size up to the configured maximum burst size. Examples of when a host may use different burst sizes include (but are not limited to) a fairness policy on the host, retries for an interrupt stream, etc. When the endpoint is an OUT, the host can easily control the burst size (receiver must always be able to manage a transaction burst size). Note that the host must observe the configured maximum endpoint sequence range as defined below, regardless of the actual burst size it is using. When the endpoint is an IN, the host can limit the burst size for the endpoint on a per-transaction basis via a field in the acknowledgment packet sent to the device. Note that a host may override the configured burst size by specifying a value less than the configured maximum burst size of the endpoint.

4.4.2 IN Transfers

For device to host transactions, the host directs the device as to the number of packets it can send and the sequence numbers of the packets expected from the device. The device must transmit data phase packets based on the current configured characteristics of the endpoint. Configured characteristics in this context are associated with the currently active (configured) device/interface configuration characteristics. The host must adhere to the constraints of the transfer type and advertised capabilities of the endpoint. A host initiates a transfer by sending an acknowledgement packet (IN) to the device. This acknowledgement packet contains the addressing information required to route the packet to the intended recipient. It also includes the sequence number of data packet(s) expected from the device. In response the device will transmit data packet(s) with the appropriate sequence numbers back to the host. The Acknowledgement packet also implicitly acknowledges the previous data packet that was received successfully.

Note that even though the host is required to send an acknowledgement packet for every data packet received, the device can send up to the number of data packets requested in the Acknowledgement to the host without waiting for the acknowledgement packet.

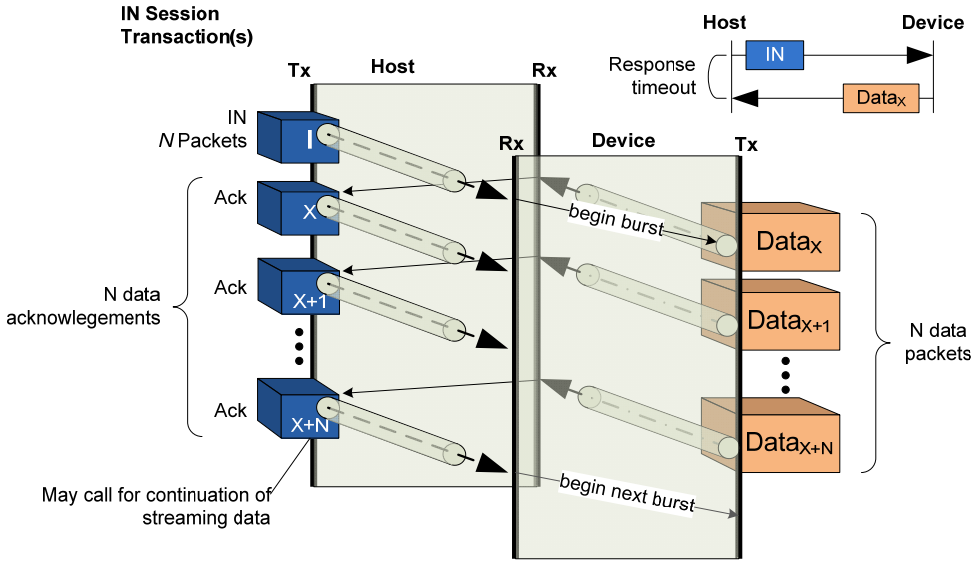


Figure 4-2. SuperSpeed USB IN Transaction Protocol

4.4.3 OUT Transfers

For host to device transactions, a host must transmit data phase packets based on the current configured characteristics of the endpoint. The host must adhere to the constraints of the transfer type and advertised capabilities of the endpoint. A host initiates a transfer by sending data packets to the device. Each data packet contains the addressing information required to route the packet to the intended recipient. It also includes the sequence number of data packet. For non-isochronous transfers, the device acknowledges the packets by sending back an Acknowledgement packet that includes the sequence number of the next expected data packet. The Acknowledgement packet also implicitly acknowledges the previous data packets that was received successfully.

Note that even though the device is required to send an acknowledgement packet for every data packet received, the host can send up to the max burst size number of data packets to the device without waiting for an acknowledgement.

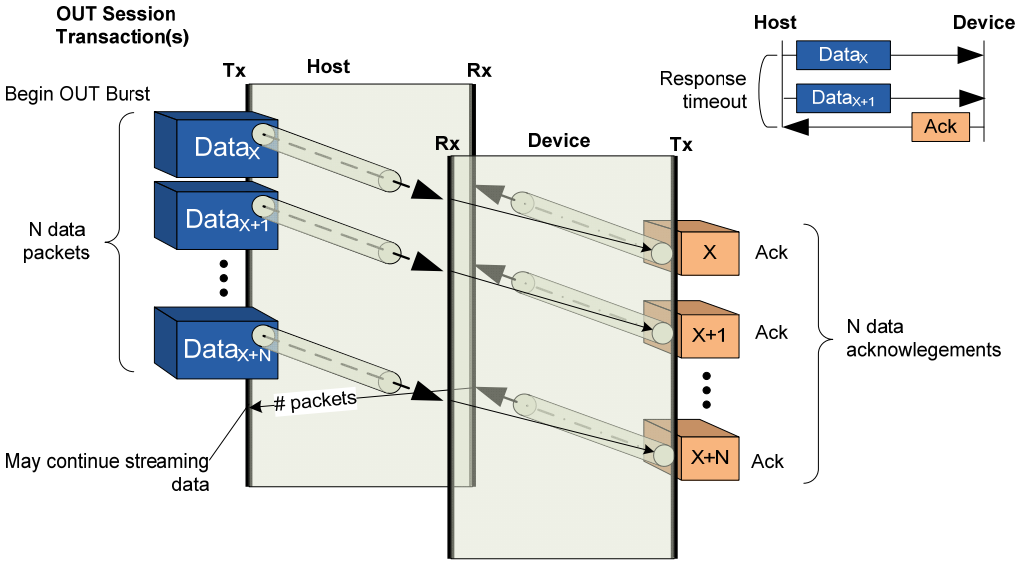


Figure 4-3. SuperSpeed USB OUT Transaction Protocol

4.4.4 Power Management and Performance

The use of inactivity timers and device driven link power management adds a new level of complexity to SuperSpeed USB. While it provides the ability for very aggressive power management, it is not visible to the host except on its root ports. So for example when the host does an IN to a device behind a hub with a power managed port, the IN will not be able to traverse the power managed link(s) until they return to the active state. The host will not be able to start another transaction until that IN completes. The effect of this behavior could have a significant impact on overall performance.

However to balance power management with good performance, the concept of a deferred request (to both INs and OUTs) is introduced. So like the NRDY a device uses to tell the host that it is not able to complete the request, a deferred command is issued by the hub as a way to tell the host that

this particular path is in a reduced power managed state and that the host should go on to schedule other transactions.

There is a Deferred bit flag in the packet that is set by a hub when the packet is destined for a downstream port that is not in an active state (e.g. U0). It also sets the Hub Depth field in the header. In addition to queuing the packet to forward when the link returns to an active state, the hub returns a copy of the packet to the host. Upon receipt of the deferred packet, the host knows the transfer request was effectively NRDYed by the hub. A device, upon receipt of the deferred packet, knows the host was requesting a transfer and will respond by sending an ERDY if it can service the request or take no action if it cannot. The deferred mechanism largely negates any performance impacts that result from link power management.

4.4.5 Control Transfers

The purpose and characteristics of Control Transfers are identical to those defined in USB 2.0 (Section 5.5 of the USB 2.0 Specification). The Protocol Chapter of this specification describes the details of the packets, bus transactions and transaction sequences used to accomplish Control transfers. The Framework Chapter of this specification defines the complete set of standard command codes used for devices.

Each device is required to implement the Default Control Pipe as a message pipe. This pipe is intended to be used for device initialization and management. This pipe is to be used to access device descriptors and to make requests of the device to manipulate its behavior (at a device-level). Control transfers must adhere to the same USB data structure definitions described in USB 2.0.

The SuperSpeed USB system will make a “best effort” to support delivery of control transfers between the host and devices. As with USB 2.0, a function and its client software cannot request specific bandwidth for control transfers.

4.4.5.1 Control Transfer Packet Size

Control endpoints have a fixed maximum control transfer data payload size of 512-bytes and have a maximum burst size of one (1). These maximums apply to all data transactions during the data stage of the control transfer. Refer to Section 8.12.2 for detailed information on the Setup and Status stages of a control transfer in SuperSpeed USB.

A SuperSpeed USB device must report a value of 09H in the bMaxPacketSize field of its Device Descriptor. The rule for decoding the default max packet size for the Default Control Pipe is given in Section 9.6.1. The Default Control Pipe must support a maximum sequence value of 32 (i.e., sequence values in the range [0-31] are used).

The requirements for data delivery and completion of device to host and host to device Data stages are generally not changed between USB 2.0 and SuperSpeed USB (see Section 5.5.3 in the USB 2.0 Specification). A control pipe may have a variable-length data phase in which the host requests more data than is contained in the actual data structure. When all of the data is returned to the host, the function must indicate that the Data stage is ended by returning a short packet.

4.4.5.2 Control Transfer Bandwidth Requirements

A device has no way to indicate the desired bandwidth for a control pipe. A host balances the bus access requirements of all control pipes and pending transactions on those pipes to provide a "best effort" delivery between client software and functions on the device. This policy is unchanged with regards to USB 2.0.

SuperSpeed USB requires that part of the USB bus be reserved to be available for use by control transfers as follows:

- The transactions of a control transfer may be scheduled coincident with transactions for other function endpoints of any defined transfer type.
- Retries of control transfers may be scheduled with equal priority to all new control and bulk transactions.
- If there are control and bulk transfers pending for multiple endpoints, control transfers for different endpoints are selected for service according to a fair access policy that is Host Controller implementation-dependent.
- When a control endpoint delivers a flow control event (as defined in Section 8.10.1), the host will remove the endpoint from the actively scheduled endpoints. The device must transmit an Endpoint Ready (ERDY) device notification to notify the host that it is ready to resume data streaming to/from the endpoint.

These requirements allow control transfers between a host and devices to regularly move data across the SuperSpeed USB bus with "best effort". The basic System Software discretionary behavior defined in USB 2.0 (Section 5.5.4) applies equally to SuperSpeed USB Control Transfers.

4.4.5.3 Control Transfer Data Sequences

SuperSpeed USB preserves the message format and general stage sequencing of control transfers defined in USB 2.0 (see Section 5.5.5). The SuperSpeed USB protocol defines some changes to the Setup and Status stages of a control transfer, however all of the sequencing requirements for normal and error recovery scenarios defined in USB 2.0 Section 5.5.5 directly map to the SuperSpeed USB Protocol.

4.4.6 Bulk Transfers

The purpose and characteristics of Bulk Transfers is similar to that defined in USB 2.0 (Section 5.8 of the USB 2.0 Specification). Section 8.12.1 of this specification describes the details of the packets, bus transactions and transaction sequences used to accomplish Bulk transfers. Bulk transfer type is intended to support devices that want to communicate relatively large amounts of data at highly variable times where the transfer can use any available SuperSpeed USB bandwidth. A SuperSpeed (SS) USB Bulk function endpoint provides the following:

- Access to the SuperSpeed USB bus on a bandwidth available basis
- Guaranteed delivery of data, but no guarantee of bandwidth or latency

Bulk transfers occur only on a bandwidth-available basis. With large amount of available bus time, bulk transfers may happen relatively quickly; for conditions with little bus time available, bulk transfers may take a long time.

SuperSpeed USB retains the following characteristics of bulk pipes:

- No data content structure is imposed on communication flow for bulk pipes
- A bulk pipe is a stream pipe, and therefore always has communication flow either into or out of the host for any pipe instance. If an application requires a bi-directional bulk communication flow, two bulk pipes must be used (one IN and one OUT).

Standard USB bulk pipes provide the ability to move a stream of data. SuperSpeed adds the notion of streams that provide a protocol-level support for a multi-stream model.

4.4.6.1 Bulk Transfer Packet Size

An endpoint for bulk transfers specifies the maximum data packet payload size and burst size that the endpoint can accept from or transmit on the SuperSpeed USB bus. The only allowable maximum data payload size for all bulk endpoints is 1024. The maximum allowable burst size a bulk endpoint may specify is any value in the range 1 to 16. All SuperSpeed USB bulk endpoints support a maximum sequence value of 32 (i.e., sequence values in the range [0-31] are used).

A host is required to support any SuperSpeed USB bulk endpoint. A host must support all combinations of bulk packet sizes and bulk burst sizes. No host is required to support larger than maximum packet sizes. The host ensures that no data payload of any data packet in a transaction burst will be sent to the endpoint that is larger than the reported maximum packet size and it will not send more data packets than the reported maximum burst size.

A bulk function endpoint must always transmit data payloads with data fields less than or equal to the endpoint's maximum packet size. If the bulk transfer has more information than will fit into the Maximum packet size for the transaction, all data payloads in the data burst are required to be Maximum packet size except for the last data payload in the burst, which may contain the remaining data. A bulk transfer may span multiple bus transactions. A bulk transfer is complete when the endpoint does one of the following:

- Has transferred exactly the amount of data expected
- Transfers a data packet with a payload less than Maximum packet size.
- Responds with a STALL handshake.

4.4.6.2 Bulk Transfer Bandwidth Requirements

As with USB 2.0 a bulk function endpoint has no way to indicate a desired bandwidth for a bulk pipe. Bulk transactions occur on the SuperSpeed USB bus only on a bandwidth available basis. SuperSpeed USB provides a "good effort" delivery of bulk data between client software and device functions. Moving control transfers over the SuperSpeed USB bus has priority over moving bulk transactions. When there are bulk transfers pending for multiple endpoints, the host will provide transaction opportunities to individual endpoints according to a fair access policy, which is host implementation dependent.

All bulk transfers pending in a system contend for the same available bus time. An endpoint and its client software cannot assume a specific rate of service for bulk transfers. Bus time made available to a software client and its endpoint can be changed as other devices are inserted into and removed from the system or also as bulk transfers are requested for other function endpoints. Client software cannot assume ordering between bulk and control transfers; i.e., in some situations, bulk transfers can be delivered ahead of control transfers.

The host can use any burst size between 1 and the reported maximum in transactions with a bulk endpoint to more effectively utilize the available bandwidth. For example, there may be more bulk transfers than bandwidth available, so a host can employ a policy of using smaller data bursts per transactions to provide fair service to all pending bulk data streams.

When a bulk endpoint delivers a flow control event (as defined in Section 8.10.1) the host will remove it from the actively scheduled endpoints. The device must transmit an Endpoint Ready (ERDY) device notification to the host to notify it that the associated bulk endpoint has bulk data or bulk buffer space available and is ready to resume data streaming.

4.4.6.3 Bulk Transfer Data Sequences

Bulk transactions use the standard burst sequence for reliable data delivery protocol defined in Section 8.10.2. Bulk endpoints are initialized to the initial transmit or receive window condition (see Section 8.12.1.2 and Section 8.12.1.3) by an appropriate control transfer (SetConfiguration, SetInterface, ClearEndpointFeature). A host likewise assumes the initial transmit or receive window state for bulk pipes after it has successfully completed the appropriate control transfer as mentioned above.

Halt conditions for a SuperSpeed USB bulk pipe have the identical side effects as defined for a USB 2.0 bulk endpoint. Recovery from halt conditions are also identical to the USB 2.0 specification, see Section 5.8.5 in the USB 2.0 specification. A bulk pipe halt condition includes a STALL handshake response to a transaction or exhaustion of the host's transaction retry policy due to transmission errors.

4.4.6.4 Bulk Streams

A standard USB Bulk Pipe represents the ability to move single stream of (FIFO) data between the host and a device via a host memory buffer and a device endpoint. **Streams** provide protocol-level support for a multi-stream model.

Streams are managed between the host and a device using the *Stream Protocol*. Each Stream is assigned a *Stream ID* (SID).

The Stream Protocol defines a handshake, which allows the device or host to establish the *Current Stream* (CStream) ID associated with an endpoint. The host uses CStream ID to select the command or operation-specific Endpoint Buffer(s) that will be used for subsequent data transfers on the pipe (see the figure below). The device uses CStream ID to select the Function Data buffer(s) that will be used (see the figure below).

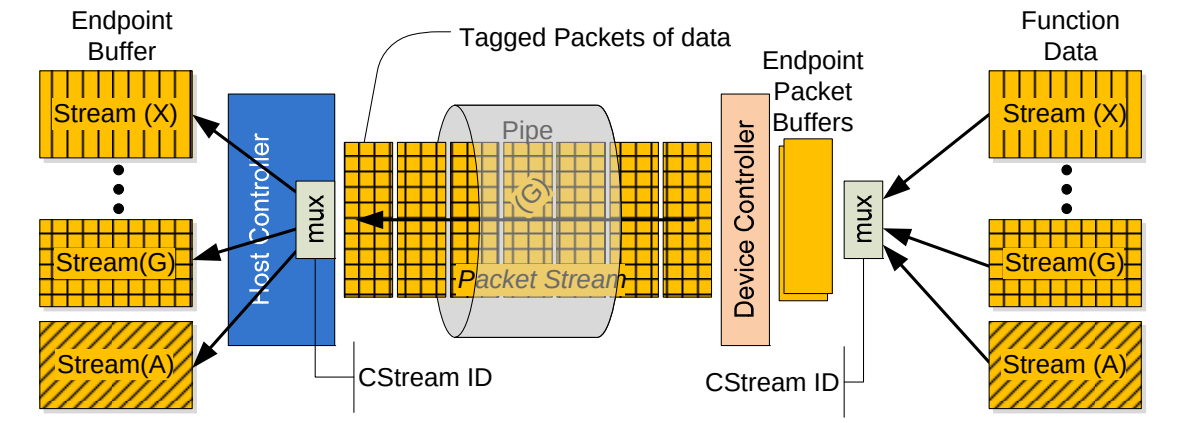


Figure 4-4. USB SS IN Stream Example

The example above represents an IN Bulk pipe, where a large number of Streams have been established. Associated with each Stream in host memory is one or more Endpoint Buffers to receive the Stream data, and in the device there is a corresponding command or operation-specific Function Data to be transmitted to the host.

When the device has data available for a specific Stream (G in this example), it issues an ERDY tagged with the CStream ID, and the host will begin issuing IN ACK TP's to the device that are tagged with the CStream ID. The device will respond by returning DPs that contain the Function Data associated with the CStream ID that are also tagged with the CStream ID. When the host receives the data, it uses the CStream ID to select the set of Endpoint Buffers that will receive the data.

When the Function Data is exhausted, the device terminates the Stream. The host is also allowed to terminate the Stream if it runs out of Endpoint Buffer space.

Streams may be used, for example, to support out-of-order data transfers required for mass storage device command queuing.

Streams only apply to SS Bulk USB endpoints.

A standard bulk endpoint has a single set of Endpoint Buffers associated with it. Streams extend the number of host buffers accessible by an endpoint from 1, to up to 65533. There is a 1:1 mapping between a host buffer and a *Stream ID*.

Device Class defined methods are used for coordinating the Stream IDs that are used by host to select Endpoint Buffers and the device to select Function Data associated with a particular Stream. Typically this is done via an out-of-band mechanism (e.g. another endpoint) that is used to pass the list of valid Stream IDs between the host and the device.

The selection of the Current Stream may be initiated by the host or the device, and in either case the Stream Protocol provides a method for a selection to be rejected. For example, the host may reject a Stream selection initiated by the device if it has no Endpoint Buffers available for it. Or the device may reject a Stream selection initiated by the host if it has no Function Data available for it. The Device Class defines when a stream may be selected by the host or the device, and the actions that will be taken when a Stream is rejected.

A combination of vendor and Device Class defined algorithms determine how Streams are scheduled by a device. The Stream Protocol provides methods for starting, stopping and switching Streams.

Mechanisms defined by the Stream Protocol allow the device or the host to flow control a Stream. These mechanisms overlap with the standard bulk flow control mechanism (NRDY, ERDY, etc.).

The host also may start or stop a Stream. For instance, the host will stop a Stream if it runs out of buffer space for the Stream. When the host controller informs the device of this condition, the device may switch to another Stream or wait and continue the same Stream when the host receives more buffers.

The Stream Protocol also provides a mechanism which allows the host to asynchronously inform the device when Endpoint Buffers have been added to the pipe. This is useful in cases where the host must terminate a stream because it ran out of Endpoint Buffers; however the device still has more Function Data to transfer. Without this mechanism, the device would have to periodically retry starting the Stream (impacting power management), or a long latency out-of-band method would be required.

Since Streams are run over a standard bulk pipe, an error will halt the pipe, stopping all stream activity. Removal of the halt condition is achieved via software intervention through a separate control pipe as it is for a standard bulk pipe.

Finally, Streams significantly increase the functionality of a bulk endpoint, while having a minimal impact on the additional hardware required to support the feature in hosts and devices.

4.4.7 Interrupt Transfers

The purpose and characteristics of Interrupt Transfers are similar to those defined in USB 2.0 (Section 5.7 of the USB 2.0 Specification). The SuperSpeed USB interrupt transfer types are intended to support devices that want a high reliability method to communicate a small amount of data with a bounded service interval. The Protocol Chapter of this specification describes the details of the packets, bus transactions and transaction sequences used to accomplish Interrupt transfers. The SuperSpeed USB Interrupt transfer type nominally provides the following:

- Guaranteed maximum service interval.
- Guaranteed retry of transfer attempts the next service interval.

Interrupt transfers are attempted each service interval for an interrupt endpoint. Bandwidth is reserved to guarantee a transfer attempt each service interval. Once a transfer is successful, another transfer attempt is not made until the next service interval. If the endpoint responds with a Not Ready (NRDY) or an ACK indicating that it cannot accept any more packets the host will not attempt another transfer to that endpoint until it receives an Endpoint Ready (ERDY) device notification. The host must then service the endpoint within twice the guaranteed service interval. The requested service interval for the endpoint is described in its endpoint descriptor.

SuperSpeed USB retains the following characteristics of interrupt pipes:

- No data content structure is imposed on communication flow for interrupt pipes
- An interrupt pipe is a stream pipe, and therefore is always unidirectional

4.4.7.1 Interrupt Transfer Packet Size

An endpoint for interrupt transfers specifies the maximum data packet payload size that it can accept from or transmit on the SuperSpeed USB bus. The only allowable maximum data payload size for interrupt endpoints is 1024 bytes for interrupt endpoints that support a Burst size greater than one and can be any size from 1 to 1024 for an interrupt endpoint with a Burst size equal to one. The maximum allowable burst size for interrupt endpoints is three. All SuperSpeed USB interrupt endpoints support a maximum sequence value of 32 (i.e., sequence values in the range [0-31] are used).

SuperSpeed USB interrupt endpoints are only intended for moving small amounts of data with a bounded service interval. The SuperSpeed USB protocol does not require the interrupt transactions to be maximum size.

A host is required to support any SuperSpeed USB interrupt endpoint. A host must support all combinations of interrupt packet sizes and burst sizes. No host is required to support larger than maximum packet sizes. The host ensures that no data payload of any data packet in a transaction burst will be sent to the endpoint that is larger than the reported maximum packet size and it will not send more data packets than the reported maximum burst size.

An interrupt endpoint must always transmit data payloads with data fields less than or equal to the endpoint's maximum packet size. If the interrupt transfer has more information than will fit into the maximum packet size for the endpoint, all data payloads in the data burst are required to be maximum packet size except for the last data payload in the burst, which may contain the remaining data. An interrupt transfer may span multiple bus transactions.

An interrupt transfer is complete when the endpoint does one of the following:

- Has transferred exactly the amount of data expected has been transferred
- Transfers a data packet with a payload less than Maximum packet size
- Responds with a STALL handshake

4.4.7.2 Interrupt Transfer Bandwidth Requirements

Periodic endpoints can be allocated at most 80% of the SuperSpeed USB total available bandwidth.

An endpoint for an interrupt pipe specifies its desired service interval bound via its endpoint descriptor. An interrupt endpoint can specify a desired period (2bInterval-1) x 125 μs, where bInterval is in the range 1 to (including) 16. The USB System Software will use this information during configuration to determine a period that can be sustained. The period provided by the system may be shorter than that desired by the device up to the shortest period defined by the SuperSpeed USB (125 μs which is also referred to as a Bus interval). Note that errors on the bus can prevent an interrupt transaction from being successfully delivered over the bus and consequently exceed the desired period.

A SuperSpeed USB interrupt endpoint can move up to 3072 bytes per a bus interval (or 192 Mb/s). Interrupt transfers are moved over the USB by accessing an interrupt endpoint every service interval. For interrupt endpoints, the host has no way to determine whether the endpoint will source/sync data without accessing the endpoint and requesting an interrupt transfer. If an interrupt IN endpoint has no interrupt data to transmit or an interrupt OUT endpoint has insufficient buffer to accept data when accessed by the host, it responds with a flow control response.

An endpoint should only provide interrupt data when it has an interrupt data pending to avoid having a software client erroneously notified of a transfer completion. A zero-length data payload is a valid transfer and may be useful for some implementations. The host may access an endpoint at any point during the service interval. The interrupt endpoint should not assume a fixed spacing between transaction attempts. The interrupt endpoint can assume only that it will receive a transaction attempt within the service interval bound. Errors can prevent the successful exchange of data within the service interval bound and a host is not required to retry the transaction in the same service interval and is only required to retry the transaction in the next service interval.

4.4.7.3 Interrupt Transfer Data Sequences

Interrupt transactions use the standard burst sequence for reliable data delivery protocol defined in Section 8.10.2. Interrupt endpoints are initialized to the initial transmit or receive window condition (see Section 8.12.4.1 and Section 8.12.4.2) by an appropriate control transfer (SetConfiguration, SetInterface, ClearEndpointFeature). A host sets the initial transmit or receive window state for interrupt pipes after it has successfully completed the appropriate control transfer.

As mentioned earlier, if the endpoint responds with a flow control response, the host will not attempt another transfer to that endpoint until it receives an Endpoint Ready (ERDY) device notification. The host must then service the endpoint within twice the guaranteed service interval.

Halt conditions for a SuperSpeed USB interrupt pipe have the identical side effects as defined for a USB 2.0 interrupt endpoint. Recovery from halt conditions are also identical to the USB 2.0 specification, see Section 5.7.5 in the USB 2.0 specification. An interrupt pipe halt condition includes a STALL handshake response to a transaction or exhaustion of the host's transaction retry policy due to transmission errors.

4.4.8 Isochronous Transfers

The purpose of SuperSpeed USB Isochronous Transfers is similar to those defined in USB 2.0 (Section 5.6 of the USB 2.0 Specification). As in the USB 2.0 specification, the SuperSpeed USB Isochronous Transfer type is intended to support streams that want to perform error tolerant, periodic transfers within a bounded service interval. SuperSpeed USB does not transmit start of frames as on the USB 2.0 bus and timing information is transmitted to devices via ITPS. The Protocol Chapter of this specification describes the details of the packets, bus transactions and transaction sequences used to accomplish isochronous transfers as well how the timing information is conveyed to the devices that support isochronous transfers. The SuperSpeed USB Isochronous transfer type provides the following:

- Guaranteed bandwidth for transaction attempts on the SuperSpeed USB bus with bounded latency.
- Guaranteed data rate through the pipe as long as data is provided to the pipe.

Isochronous transactions are attempted each service interval for an isochronous endpoint. Isochronous endpoints that are admitted on the SuperSpeed USB bus are guaranteed the bandwidth they require on the bus. The host can request data from the device or send data to the device at any time during the service interval for a particular endpoint on that device. The requested service interval for the endpoint is described in its endpoint descriptor. The SuperSpeed USB isochronous transfer type is designed to support sources and destinations that produce and consume data at the same average rate. Software using the SuperSpeed USB isochronous transfer type actually may not be isochronous in nature.

A SuperSpeed USB isochronous pipe is a stream pipe and is always unidirectional. The endpoint description identifies whether a given isochronous pipe's communication flow is into or out of the host. If a device requires bi-directional isochronous communication flows two isochronous pipes must be used, one in each direction.

SuperSpeed USB power management may interfere with isochronous transfers whenever an isochronous transfer needs to traverse a link in a reduced power state. The resultant delay could result in the data not arriving within the service interval. To overcome this issue, SuperSpeed USB defines a PING (see Section 8.5.7) and PING_RESPONSE (see Section 8.5.7) mechanism. Before initiating an isochronous transfer the host may send a PING to the device. The device responds with a PING_RESPONSE that tells the host that all the links in the path to the device are in the active state.

4.4.8.1 Isochronous Transfer Packet Size

An endpoint for isochronous transfers specifies the maximum data packet payload size that the endpoint can accept from or transmit on SuperSpeed USB. The only allowable maximum data payload size for isochronous endpoints is 1024 bytes for isochronous endpoints that support a Burst size greater than one and can be any size from 0 to 1024 for an isochronous endpoint with a Burst size equal to one. The maximum allowable burst size for isochronous endpoints is sixteen. However an isochronous endpoint can request up to 3 bursts in the same service interval.

The SuperSpeed USB protocol does not require the isochronous data packets to be maximum size. If an amount of data less than the maximum packet size is being transferred it does not need to be padded.

A host is required to support any SuperSpeed USB isochronous endpoint. A host must support all combinations of isochronous packet sizes and burst sizes. No host is required to support larger than maximum packet sizes. The host ensures no data payload of any data packet in a transaction burst will be sent to the endpoint that is larger than the reported maximum packet size and it will not send more data packets than the reported maximum burst size and max number of bursts.

An isochronous endpoint shall always transmit data payloads with data fields less than or equal to the endpoint's maximum packet size. If the isochronous transfer has more information than will fit into the maximum packet size for the endpoint, all data payloads in the data burst are required to be maximum packet size except for the last data payload in the burst, which may contain the remaining data. An isochronous transfer may span multiple bus transactions.

4.4.8.2 Isochronous Transfer Bandwidth Requirements

Periodic endpoints can be allocated up to 80% of the SuperSpeed USB total available bandwidth.

An endpoint for an isochronous pipe specifies its desired service interval bound via its endpoint descriptor. An isochronous endpoint can specify a desired period (2bInterval-1) x 125 μs, where bInterval is in the range 1 to 16. The USB System Software will use this information during configuration to determine a period that can be sustained. Note that errors on the bus can prevent an isochronous transaction from being successfully delivered over the bus.

A SuperSpeed USB isochronous endpoint can move up to 49152 bytes per bus interval (or maximum 375 MB/s). Isochronous transfers are moved over the USB by accessing an isochronous endpoint every service interval. The host will send data or request data to or from the endpoint

every service interval. Note, if an endpoint has no isochronous data to transmit when accessed by the host, it will send back a zero length packet in response to the request for data.

The host may access an endpoint at any point during the appropriate service interval. The isochronous endpoint should not assume a fixed spacing between transaction attempts. The isochronous endpoint can assume only that it will receive a transaction attempt within the service interval bound. Errors may prevent the successful exchange of data within the service interval bound, however since the packets in an isochronous transaction are not acknowledged, a host has no way of knowing which packets were not received successfully and hence will not retry any packet that was not received successfully.

4.4.8.3 Isochronous Transfer Data Sequences

Isochronous transactions use the standard burst sequence to deliver data as defined in Section 8.10.2.. Isochronous endpoints always transmit data packets starting with sequence number zero in each service interval that they are polled. Each successive data packet transmitted in the same service interval is sent with the next higher sequence number up to the max number of packets an endpoint can send/receive per interval. Isochronous endpoints do not support retries and can not respond with flow control responses.

4.4.9 Device Notifications

Device Notifications are a standard method for a device to communicate asynchronous, device and Bus-level event information to the host. This communication method is a bus-level feature that does not map to the pipe model defined for the standard transfer types. Device Notifications are always initiated by a device and the flow of data information is always device to host. Each packet transmitted is called a Device Notification or simply Notification.

Notifications are message-oriented data communications that have a specific data format structure as defined in Section 8.5.6. Notifications do not have any payload. Devices can send a Notification at any time due to the fact that the SuperSpeed USB bus is dual simplex bus.

4.4.10 Reliability

To ensure reliable operation, several layers of protection are used. This provides reliability for both flow control and data end to end.

4.4.10.1 Physical Layer

The SuperSpeed USB Physical Layer provides bit error rates of about 1 bit in 10e12 bits.

4.4.10.2 Link Layer

The SuperSpeed USB Link Layer provides Header Packets unrecoverable/undetectable bit error rates of about 1 bit in 10e20 bits. This is done using a number of techniques including sending packet framing ordered sets, sending link level flow control commands for header packets and including a retry mechanism. This allows the Link Layer to ensure reliable end-to-end delivery of all header packets.

4.4.10.3 Protocol Layer

The SuperSpeed USB Protocol Layer depends on a 32-bit CRC in the Data Payload and a timeout coupled with retries to ensure that reliable data is provided to application.

4.4.11 Efficiency

SuperSpeed USB efficiency is dependent a number of factors including 8b10b symbol encoding, packet structure and framing, link level flow control, and protocol overhead. At 5Gbps signaling rate with 8b10b encoding, the raw throughput is 500MBps. When link flow control, packet framing and, protocol overhead are considered, it is realistic for 400MBps or more to be delivered to an application.

5 Mechanical

This chapter defines form, fit and function of the USB 3.0 connectors and cable assemblies. It contains the following contents:

- Connector mating interfaces
- Cables and cable assemblies
- Electrical requirements
- Mechanical and environmental requirements
- Implementation notes and guidelines

The intention of this chapter is to enable connector, system, and device designers and manufacturers to build, qualify and use the USB 3.0 connectors, cables, and cable assemblies.

5.1 Objective

The mechanical layer specification has been developed with the following objectives:

- Supporting 5 Gbps data rate
- Backward compatible with USB 2.0
- Minimizing connector form factor variations
- Managing EMI
- Supporting On-the-Go (OTG)
- Low cost

5.2 Significant Features

This section identifies the significant features of the USB 3.0 connectors and cable assemblies specification. The purpose of this section is not to present all the technical details associated with each major feature, but rather to highlight its existence. Where appropriate, this section references other parts of the document where further details can be found.

5.2.1 Connectors

The USB 3.0 specification defines the following connectors:

- USB 3.0 Standard-A plug and receptacle
- USB 3.0 Standard-B plug and receptacle
- USB 3.0 Power-B plug and receptacle
- USB 3.0 Micro-B plug and receptacle
- USB 3.0 Micro-A plug
- USB 3.0 Micro-AB receptacle

Table 5-1 below lists the compatible plugs and receptacles.

Table 5-1. Plugs Accepted By Receptacles

Receptacle	Plugs Accepted
USB 2.0 Standard-A	USB 2.0 Standard-A or USB 3.0 Standard-A
USB 3.0 Standard-A	USB 3.0 Standard-A or USB 2.0 Standard-A
USB 2.0 Standard-B	USB 2.0 Standard-B
USB 3.0 Standard-B	USB 3.0 Standard-B or USB 2.0 Standard-B
USB 3.0 Power-B	USB 3.0 Power-B, USB 3.0 Standard-B, or USB 2.0 Standard-B
USB 2.0 Micro-B	USB 2.0 Micro-B
USB 3.0 Micro-B	USB 3.0 Micro-B or USB 2.0 Micro-B
USB 2.0 Micro-AB	USB 2.0 Micro-B or USB 2.0 Micro-A
USB 3.0 Micro-AB	USB 3.0 Micro-B, USB 3.0 Micro-A, USB 2.0 Micro-B, or USB 2.0 Micro-A

5.2.1.1 USB 3.0 Standard-A Connector

The USB 3.0 Standard-A connector is defined as the (PC) host connector, supporting the Super Speed (SS) mode. It has the same mating interface as the USB 2.0 Standard-A connector, but with additional pins for 2 more differential pairs. See Section 5.3.1.2 for pin assignments and descriptions.

A USB 3.0 Standard-A receptacle accepts either a USB 3.0 Standard-A plug or a USB 2.0 Standard-A plug. In the former case, the link operates in a SS mode with a 5 Gbps data rate, using the two added differential pairs, or operates in the SS and HS/LS modes simultaneously, while in the latter case, the link runs in a USB 2.0 or 1.0 mode using the D+ and D- pins.

Similarly, a USB 3.0 Standard-A plug can be mated with either a USB 3.0 Standard-A receptacle or a USB 2.0 Standard-A receptacle.

A unique color coding is required for the USB 3.0 Standard-A connector housing to help users distinguish the USB 3.0 Standard-A connector from the USB 2.0 Standard-A connector- see Section 5.3.1.3 for detail.

5.2.1.2 USB 3.0 Standard-B Connector

The USB 3.0 Standard-B connector is defined for relatively large, stationary peripherals, such as external hard drives and printers. It is so defined that the USB 3.0 Standard-B receptacle accepts either a USB 3.0 Standard-B plug or a USB 2.0 Standard-B plug. Inserting a USB 3.0 Standard-B plug into a USB 2.0 Standard-B receptacle is physically disallowed. See Section 5.3.2 for detail.

5.2.1.3 USB 3.0 Power-B Connector

The USB 3.0 Power-B connector is defined to allow a USB 3.0 device to provide power to a USB adaptor without the need for an external power supply. It is identical to the USB 3.0 Standard-B connector in form factor, but has two more pins for power.

5.2.1.4 USB 3.0 Micro-B Connector

The USB 3.0 Micro-B connector is defined for small handheld devices. It is compatible with the USB 2.0 Micro-B connector, i.e., a USB 2.0 Micro-B plug works in a USB 3.0 Micro-B receptacle. Section 5.3.4 defines the USB 3.0 Micro-B connector family.

5.2.1.5 USB 3.0 Micro-AB and USB 3.0 Micro-A Connectors

The USB 3.0 Micro-AB receptacle is similar to the USB 3.0 Micro-B receptacle, except for keying difference. It accepts a USB 3.0 Micro-A plug, a USB 3.0 Micro-B plug, a USB 2.0 Micro-A plug, or a USB 2.0 Micro-B plug. The USB 3.0 Micro-AB receptacle is only allowed on OTG products, which may function as either a host or device. All other uses of the USB 3.0 Micro-AB receptacle are prohibited.

The USB 3.0 Micro-A plug is similar to the USB 3.0 Micro-B plug, except for keying difference and ID pin connections. The USB 3.0 Micro-A plug, the USB 3.0 Micro-AB receptacle, and the USB 3.0 Micro-B receptacle and plug all belong to the USB 3.0 Micro-B connector family since their interfaces differ only in keying. Similar to the USB 2.0 Micro-A plug, the USB 3.0 Micro-A plug is defined for OTG applications only.

5.2.2 Compliant Cable Assemblies

The USB 3.0 specification defines the following cable assemblies:

- USB 3.0 Standard-A plug to USB 3.0 Standard–B plug
- USB 3.0 Standard-A plug to USB 3.0 Micro–B plug
- USB 3.0 Micro-A plug to USB 3.0 Micro-B plug
- USB 3.0 Micro-A plug to USB 3.0 Standard-B plug
- Captive cable with USB 3.0 Standard-A plug
- Captive cable with USB 3.0 Micro-A plug
- Captive cable with USB 3.0 Power-B plug

The captive cable is the cable assembly with only one end attached with a plug while the other end is permanently attached to a device. This specification does not define how the permanent attachment shall be done on the device side. Section 5.5.5 offers more discussion on the captive cables.

No other types of cable assemblies are allowed by the USB 3.0 specification. Section 5.5 provides detailed discussion on USB 3.0 cable assemblies.

5.2.3 Raw Cables

Due to EMI and signal integrity requirements, each cable differential pair used for the SS mode in a USB 3.0 cable assembly must be shielded; the UTP (unshielded twist pair) used for USB 2.0 is NOT allowed for SS. Section 5.4 defines the cable construction for USB 3.0.

5.3 Connector Mating Interfaces

This section defines the connector mating interfaces, including the connector interface drawings, pin assignments and descriptions.

5.3.1 USB 3.0 Standard-A Connector

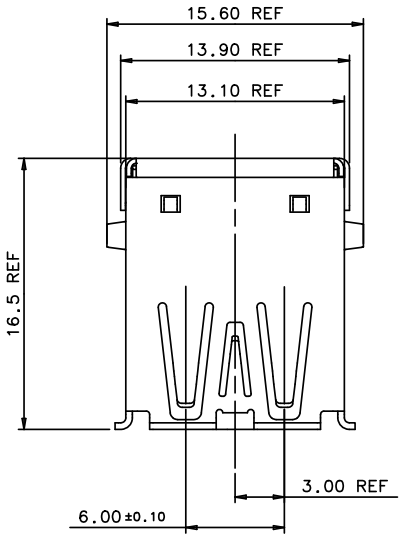
5.3.1.1 Interface Definition

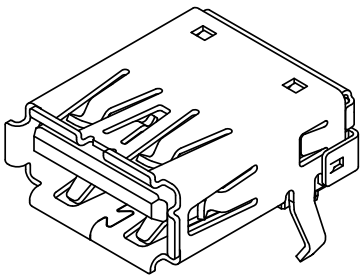
Figure 5-1 to Figure 5-4 show, respectively, the USB 3.0 Standard-A receptacle and plug interface dimensions, as well as the reference footprints for the USB 3.0 Standard-A receptacle. Note that only the dimensions that govern the mating interoperability are specified. All the REF dimensions are provided for reference only, not hard requirements.

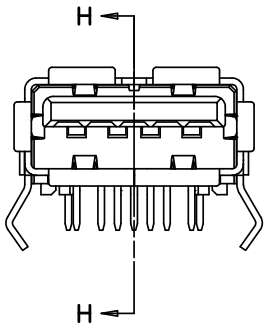
Although the USB 3.0 Standard-A connector has basically the same form factor as the USB 2.0 Standard-A connector, it has significant differences inside. Below are the key features and design areas that need attentions:

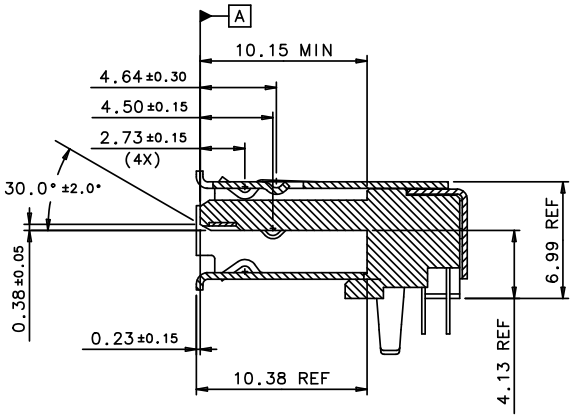
- Besides the VBUS, D-, D+ and GND pins that are required for USB 2.0/1.0, the USB 3.0 Standard-A connector includes five more pins- two differential pairs plus one GND. The two added differential pairs are for SS data transfer, supporting the unidirectional, half-duplex SS signaling topology; the added GND pin is for drain wire termination, managing signal integrity and EMI performance.
- The contact areas of the five SS pins are located towards the front of the receptacle as the blades, while the four USB 2.0 pins towards the back of the receptacle as the beams or springs. Accordingly in the plug, the SS contacts, as the beams, seat behind the USB 2.0 blades. In other words, the USB 3.0 Standard-A connector has a two-tier contact system.
- The tiered-contact approach within the Standard-A connector form factor inevitably results in less contact area to work with, as compared to the USB 2.0 Standard-A connector. The connector interface dimensions take considerations of contact mating requirements between the USB 3.0 Standard-A receptacle and USB 3.0 Standard-A plug, the USB 3.0 Standard-A receptacle and USB 2.0 Standard-A plug, and the USB 2.0 Standard-A receptacle and USB 3.0 Standard-A plug. Connector designers should carefully consider those aspects in design details.
- The connector interface definition comprehends the need to avoid shorting between the SS and USB 2.0 pins during insertion when plugging a USB 2.0 Standard-A plug into a USB 3.0 Standard-A receptacle, or a USB 3.0 Standard-A plug into a USB 2.0 Standard-A receptacle. But connector designers should be conscious about this when detailing out designs.
- There may be some increase in the USB 3.0 Standard-A receptacle connector depth (into a system board) to support the two-tiered-contacts, as compared to the USB 2.0 Standard-A receptacle.
- The through-hole footprints in Figure 5-3 and Figure 5-4 are shown for reference. Other footprints, such as SMT (surface mount) are also allowed.

- Drawings for stacked USB 3.0 Standard-A receptacles are not shown in this specification. But they are allowed, as long as they meet all the electrical and mechanical requirements defined in this specification. In fact, a double-stacked USB 3.0 Standard-A receptacle is expected to be a common application just like the double-stacked USB 2.0 Standard-A receptacle that has been widely used in PC's. When designing a stacked USB 3.0 Standard-A receptacles, efforts must be made to minimize impedance discontinuity of the top connector in the stack because of its long electrical length. Figure 5-4 shows a reference footprint for a double-stacked Standard-A receptacle connector. Note that pins 1 to 9 correspond to the lower stack while pins 10 to 18 the upper stack. Section 5.8.3 offers further discussion on the stacked connector.
- Attentions must be paid to the high speed electrical design of the USB 3.0 Standard-A connector. Besides minimizing the connector impedance discontinuities, the crosstalks among the SS pins and USB 2.0 pins should also be minimized.





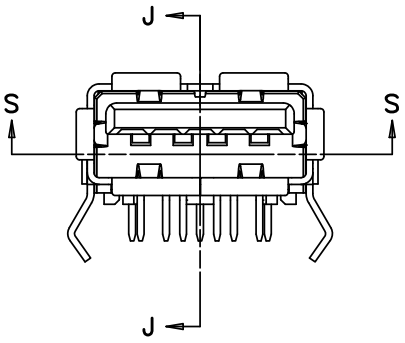


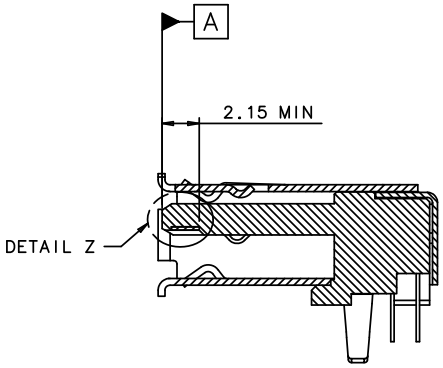


SECTION H-H

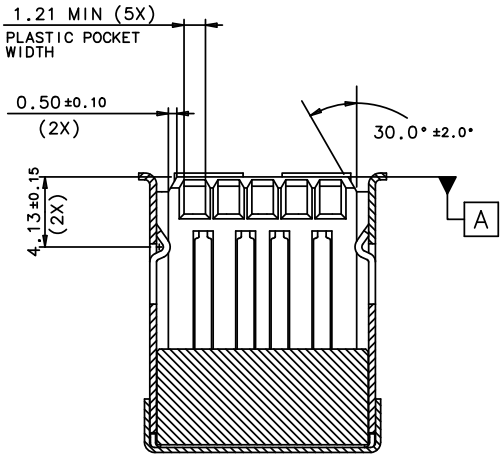
NOTES:
1) NON-DIMENSIONED GEOMETRY FOR REFERENCE ONLY, SUBJECT TO CHANGE.
2) DRAWING FOR MATING INTERFACE DIMENIONS ONLY. VIEWS MAY NOT SHOW REALISTIC MANUFACTURING CONDITION.

continued

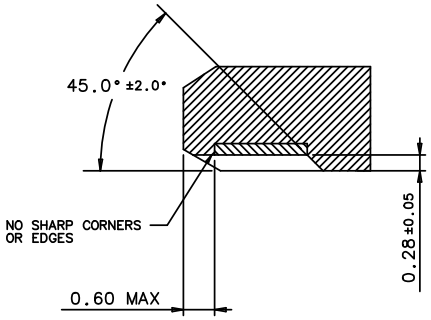




SECTION J-J



SECTION S-S



DETAIL Z

continued

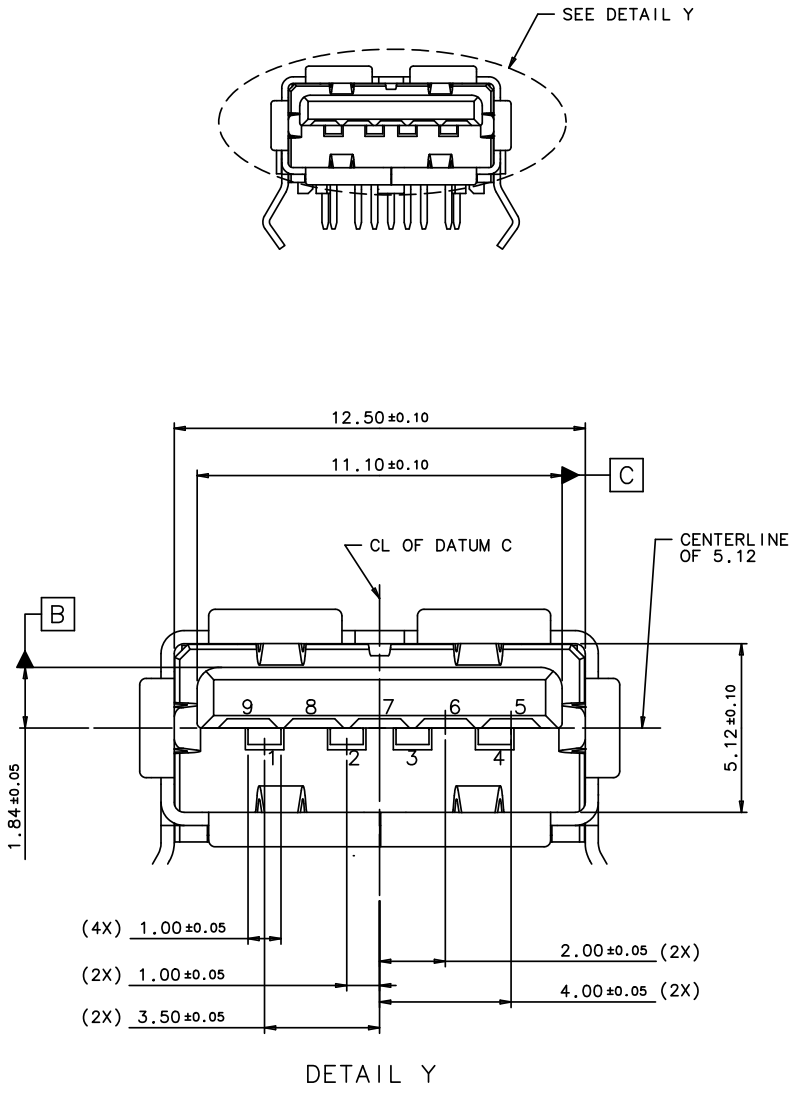
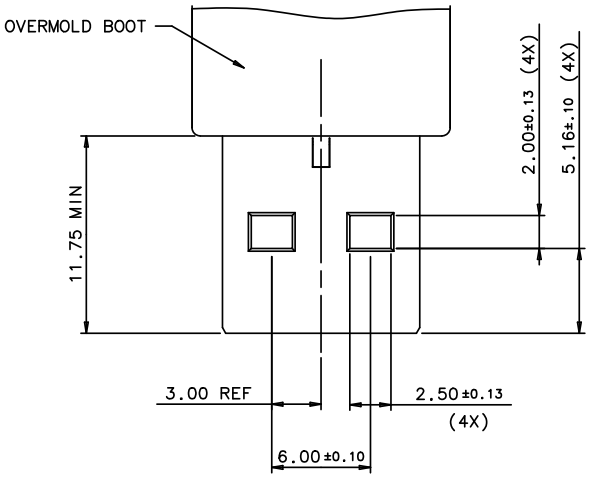
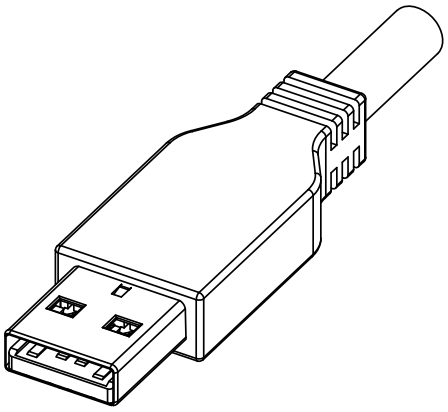
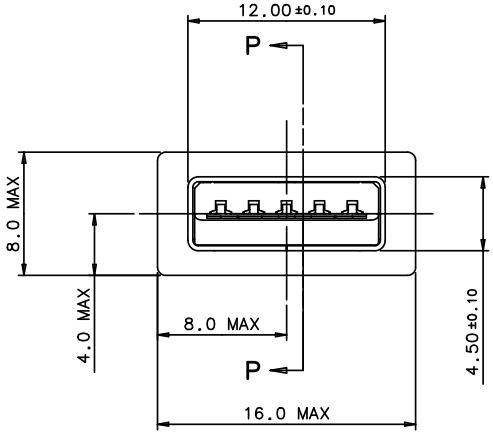
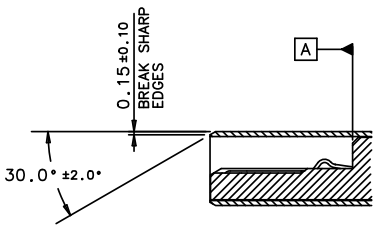


Figure 5-1. USB 3.0 Standard-A Receptacle Interface Dimensions





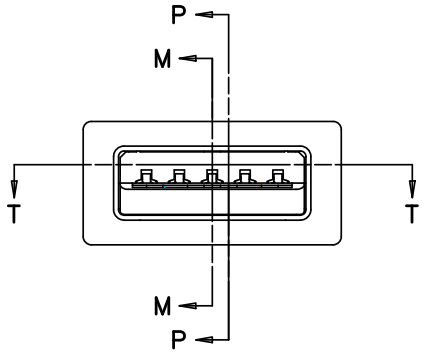


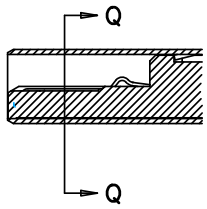


SECTION P-P

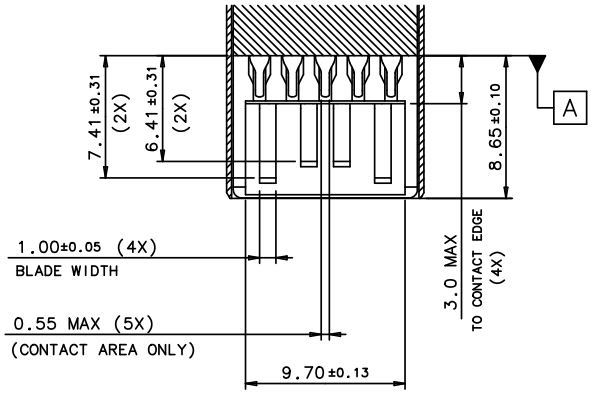
NOTES:
1) NON-DIMENSIONED GEOMETRY FOR REFERENCE ONLY, SUBJECT TO CHANGE
2) DRAWING FOR MATING INTERFACE DIMENSIONS ONLY. VIEWS MAY NOT SHOW REALISTIC MANUFACTURING CONDITION.

continued

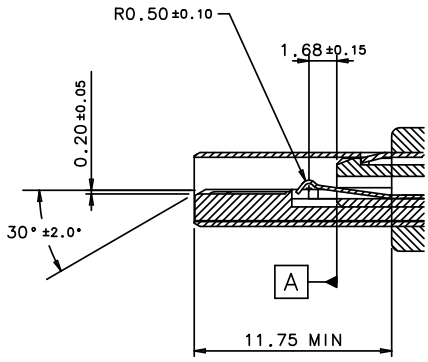




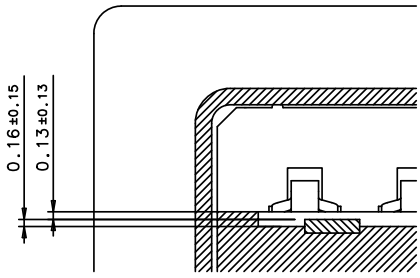
SECTION P-P



SECTION T-T



SECTION M-M



SECTION Q-Q

continued

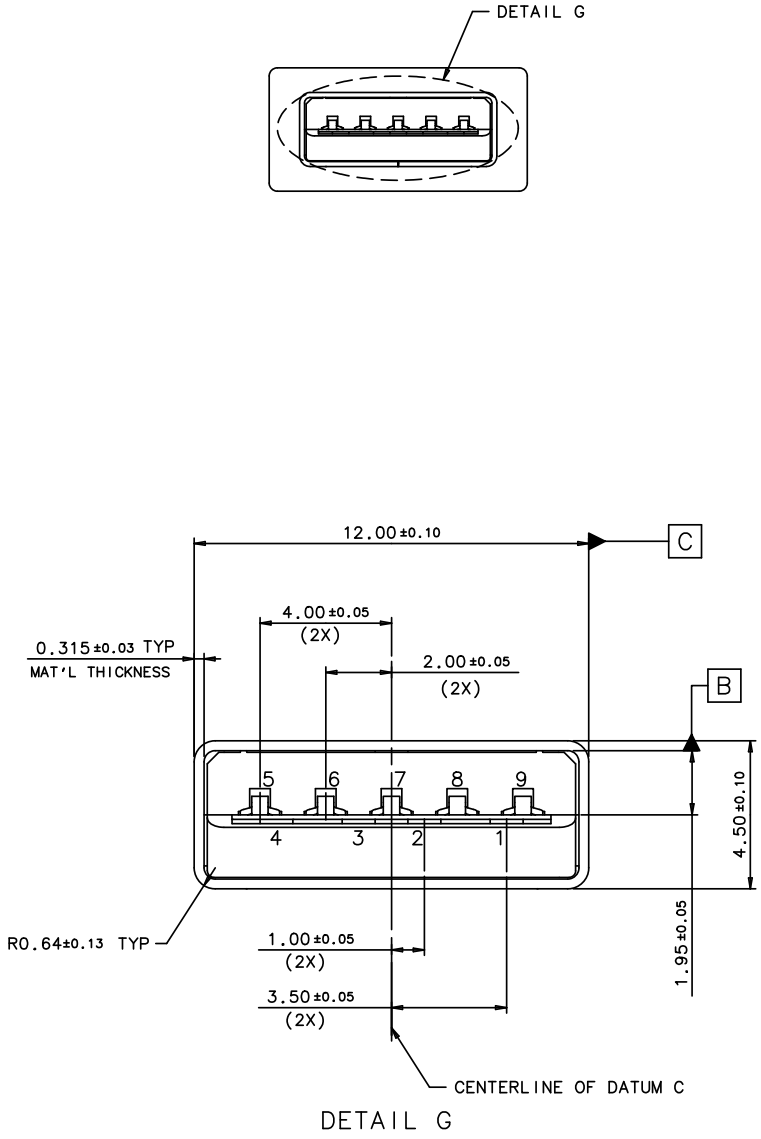


Figure 5-2. USB 3.0 Standard-A Plug Interface Dimensions

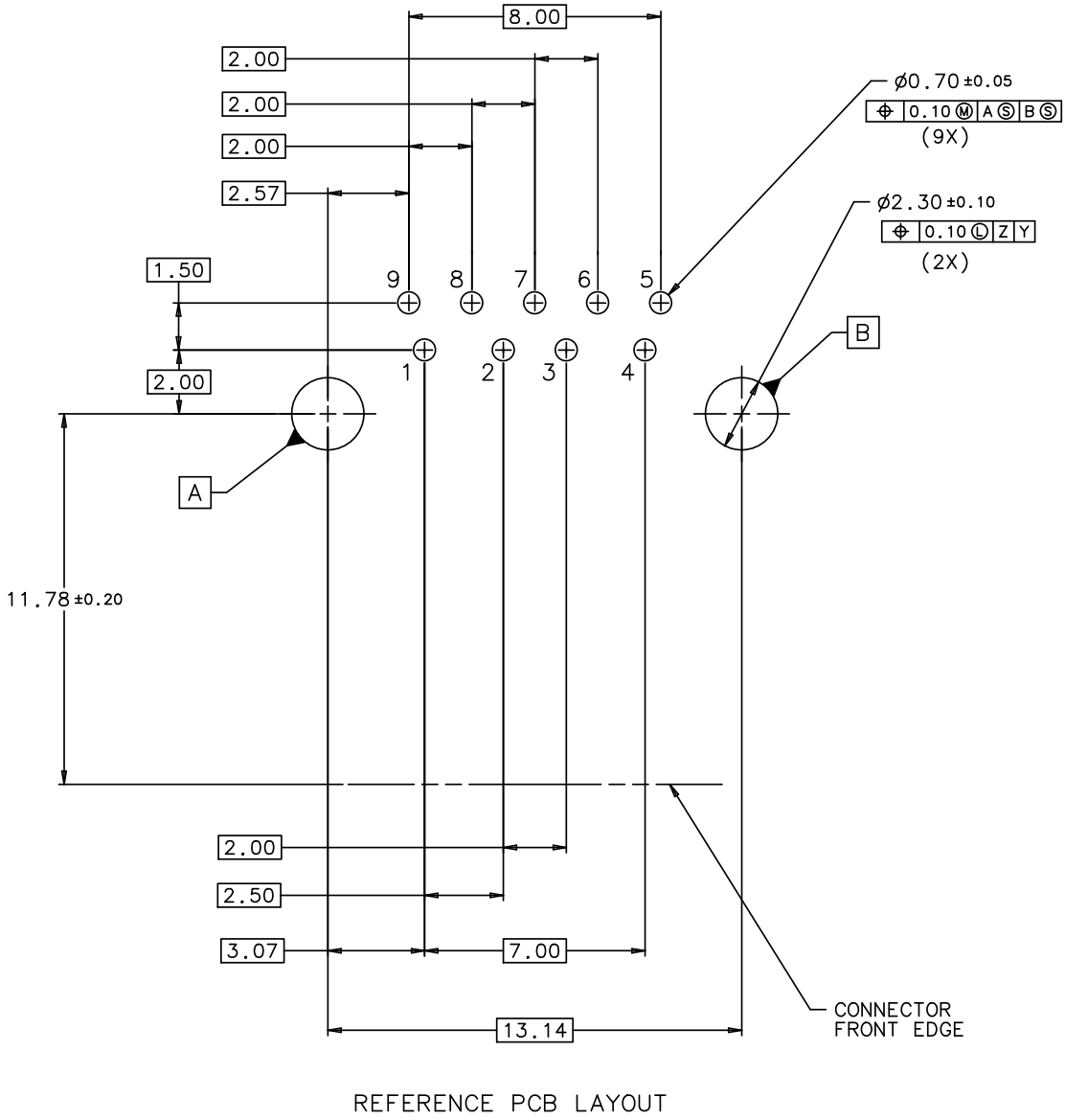


Figure 5-3. Reference Footprint for the USB 3.0 Standard-A Receptacle

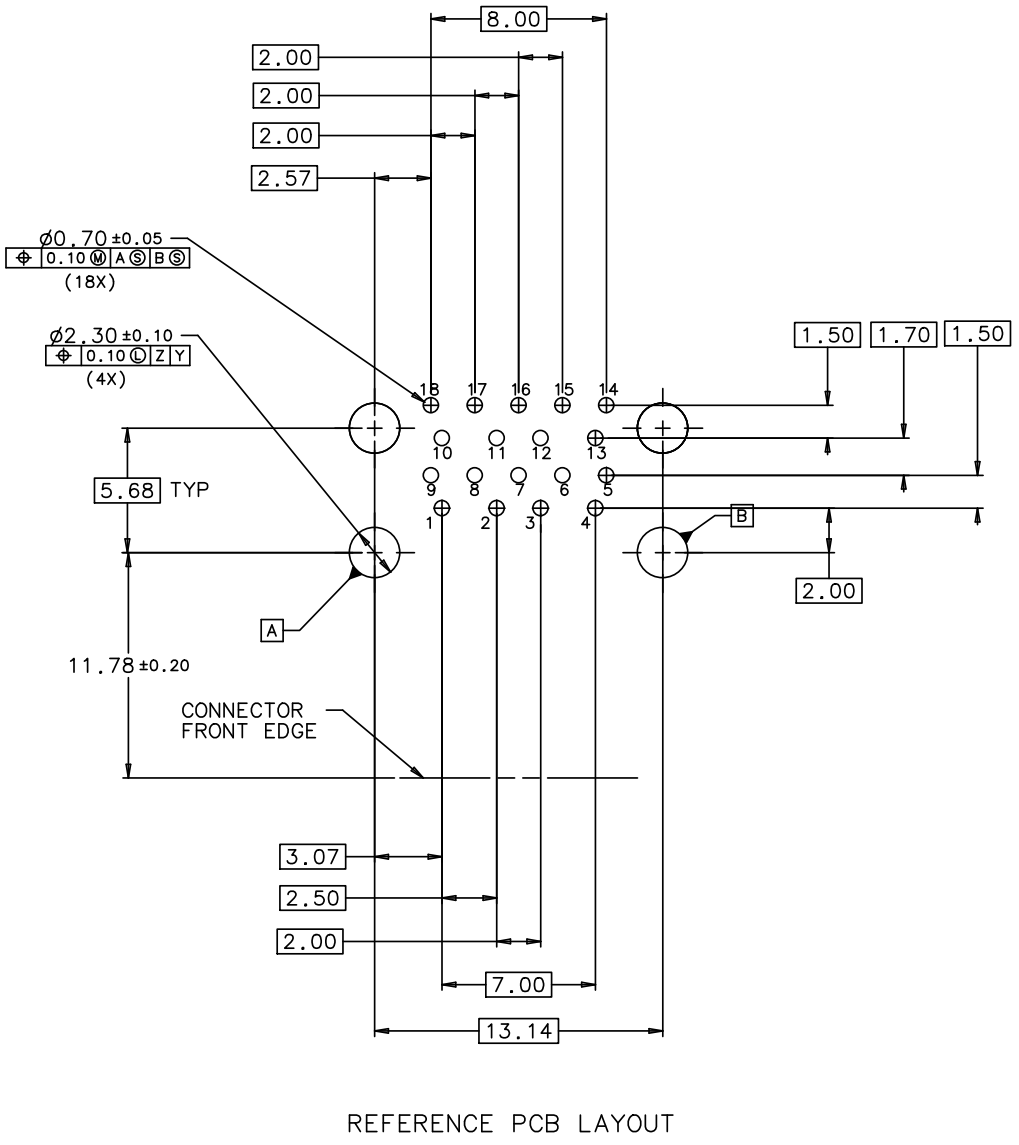


Figure 5-4. Reference Footprint for the USB 3.0 Double-Stacked Standard-A Receptacle

5.3.1.2 Pin Assignments and Description

The usage and assignments of the nine pins in the USB 3.0 Standard-A connector are defined in the following table.

Table 5-2. USB 3.0 Standard-A Connector Pin Assignments

Pin Number	Signal Name	Description	Mating Sequence
1	VBUS	Power	Second
2	D-	USB 2.0 differential pair	Third
3	D+		
4	GND	Ground for power return	Second
5	StdA_SSRX-	Super Speed receiver differential pair	Last
6	StdA_SSRX+		
7	GND	Ground for signal return	
8	StdA_SSTX-	Super Speed transmitter differential pair	
9	StdA_SSTX+		
Shell	Shield	Connector metal shell	First

Note: TX and RX are defined from the host perspective

The physical location of the pins in the connector is illustrated in Figure 5-1 to Figure 5-4. Note that pins 1 to 4 are referred to as the USB 2.0 pins, while pins 5 to 9 the SS pins.

5.3.1.3 USB 3.0 Standard-A Connector Color Code

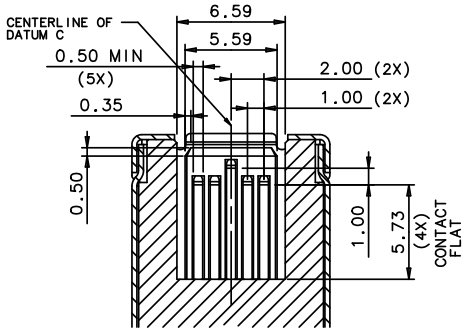
Since both the USB 2.0 Standard–A and USB 3.0 Standard-A receptacles may co-exist in a platform, color coding is required for the USB 3.0 Standard-A connector (receptacle and plug) housings to help users distinguish it from the USB 2.0 Standard-A connector.

BLUE (Pantone 300C) is the required color for the USB 3.0 Standard-A receptacle and plug plastic housings. Connector manufacturers and system integrators must make sure the blue-colored receptacle housing be visible to users.

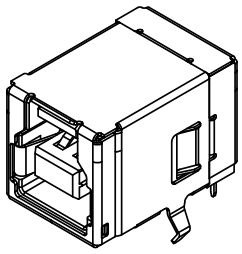
5.3.2 USB 3.0 Standard-B Connector

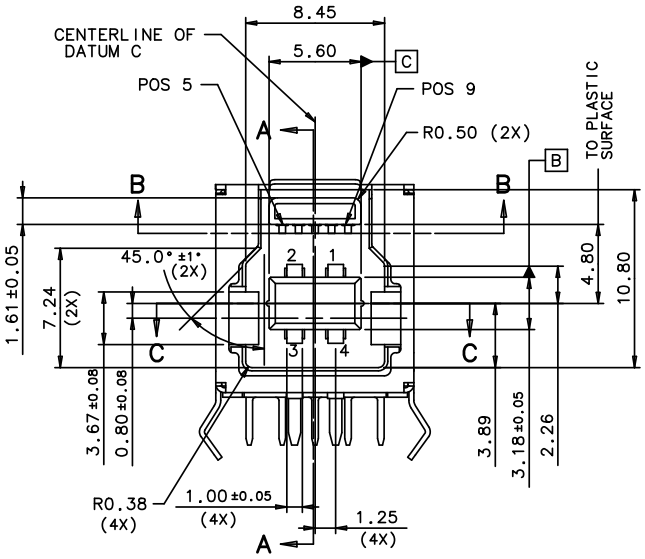
5.3.2.1 Interface Definition

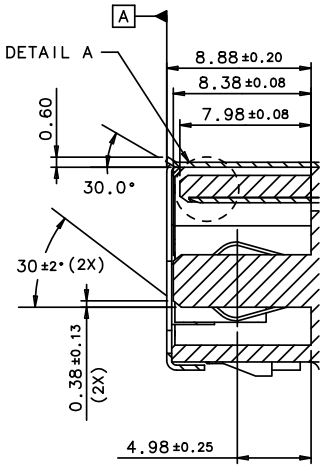
Figure 5-5 to Figure 5-7 show, respectively, the USB 3.0 Standard-B receptacle and plug interface dimensions, as well as the reference footprint.



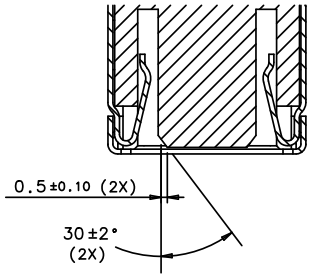
SECTION B-B



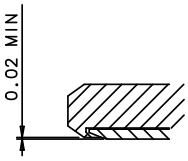




SECTION A-A

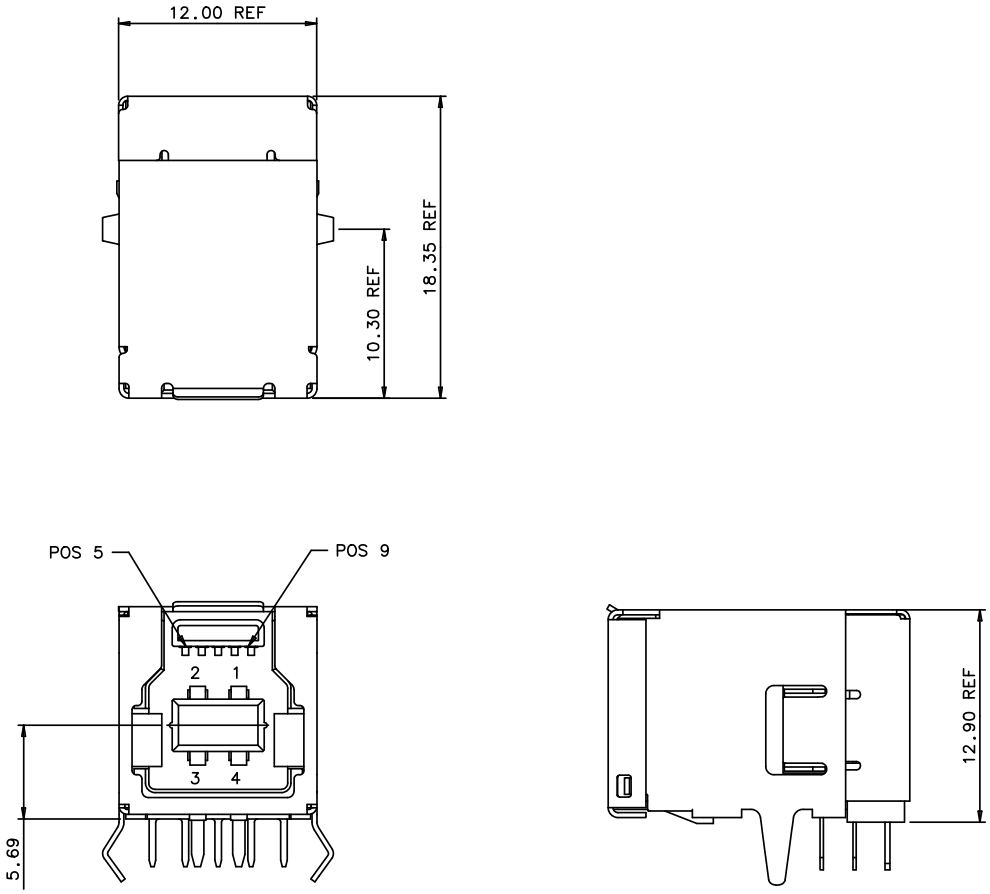


SECTION C-C



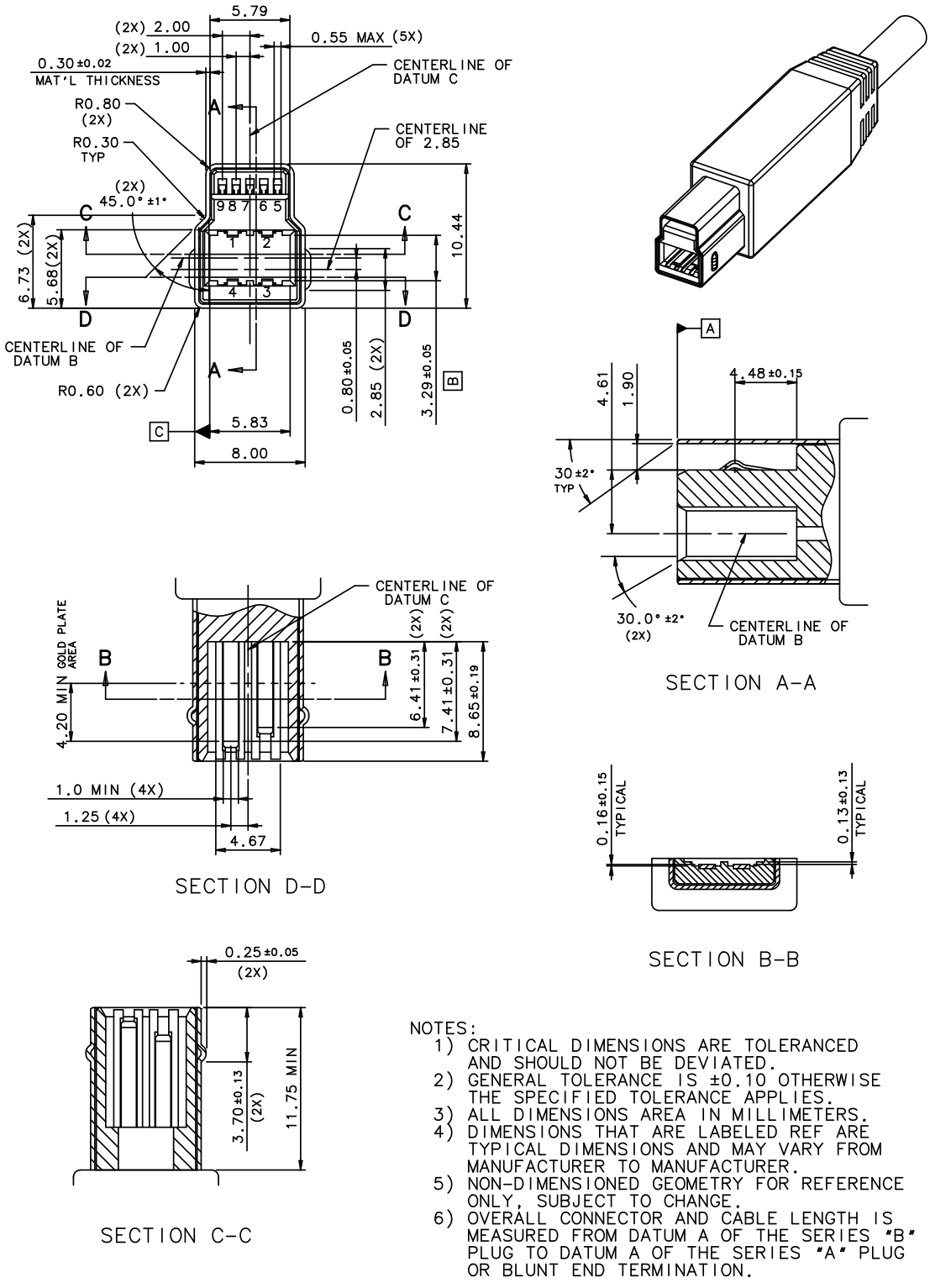
DETAIL A

continued



NOTES:
1) CRITICAL DIMENSIONS ARE TOLERANCED AND SHOULD NOT BE DEVIATED.
2) GENERAL TOLERANCE IS ±0.10 OTHERWISE THE SPECIFIED TOLERANCE APPLIES.
3) ALL DIMENSIONS AREA IN MILLIMETERS.
4) DIMENSIONS THAT ARE LABELED REF ARE TYPICAL DIMENSIONS AND MAY VARY FROM MANUFACTURER TO MANUFACTURER.
5) NON-DIMENSIONED GEOMETRY FOR REFERENCE ONLY, SUBJECT TO CHANGE.

Figure 5-5. USB 3.0 Standard-B Receptacle Interface Dimensions



NOTES:
1) CRITICAL DIMENSIONS ARE TOLERANCED AND SHOULD NOT BE DEVIATED.
2) GENERAL TOLERANCE IS ±0.10 OTHERWISE THE SPECIFIED TOLERANCE APPLIES.
3) ALL DIMENSIONS AREA IN MILLIMETERS.
4) DIMENSIONS THAT ARE LABELED REF ARE TYPICAL DIMENSIONS AND MAY VARY FROM MANUFACTURER TO MANUFACTURER.
5) NON-DIMENSIONED GEOMETRY FOR REFERENCE ONLY, SUBJECT TO CHANGE.
6) OVERALL CONNECTOR AND CABLE LENGTH IS MEASURED FROM DATUM A OF THE SERIES "B" PLUG TO DATUM A OF THE SERIES "A" PLUG OR BLUNT END TERMINATION.

Figure 5-6. USB 3.0 Standard-B Plug Interface Dimensions

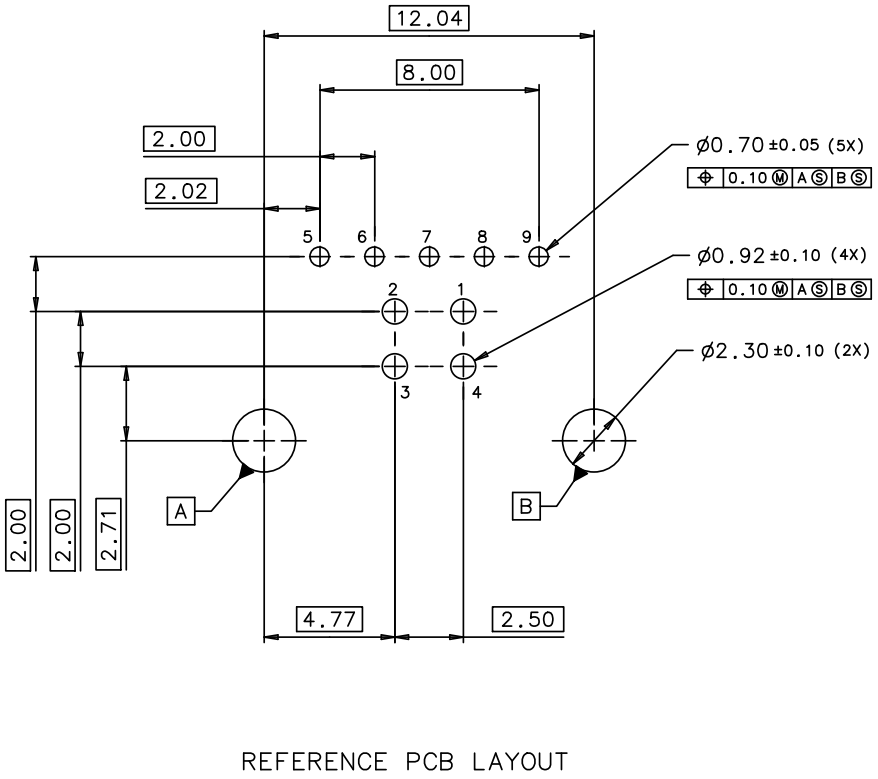


Figure 5-7. Reference Footprint for the USB 3.0 Standard-B Receptacle

The USB 3.0 Standard-B receptacle interfaces have two portions: the USB 2.0 interface and the USB 3.0 SS interface. The USB 2.0 interface consists of pins 1 to 4 while the USB 3.0 SS interface pins 5 to 9.

When a USB 2.0 Standard-B plug is inserted into the USB 3.0 Standard-B receptacle, only the USB 2.0 interface is engaged, and the link will operate in the HS, FS, or LS mode, not taking the full advantage of the SS capability. However, since the USB 3.0 SS portion is visibly not mated when a USB 2.0 Standard-B plug is in the USB 3.0 Standard-B receptacle, users will get the visual feedback that the cable plug is not quite matched with the receptacle. Only when a USB 3.0 Standard-B plug is inserted into the USB 3.0 Standard-B receptacle, is the interface completely visibly engaged.

5.3.2.2 Pin Assignments and Description

The usage and assignments of the nine pins in the USB 3.0 Standard-B connector are defined in Table 5-3.

Table 5-3. USB 3.0 Standard-B Connector Pin Assignments

Pin Number	Signal Name	Description	Mating Sequence
1	VBUS	Power	Second
2	D-	USB 2.0 differential pair	Third or beyond
3	D+		
4	GND	Ground for power return	Second
5	StdB_SSTX-	Super Speed transmitter differential pair	Third or beyond
6	StdB_SSTX+		
7	GND	Ground for signal return	
8	StdB_SSRX-	Super Speed receiver differential pair	
9	StdB_SSRX+		
Shell	Shield	Connector metal shell	First

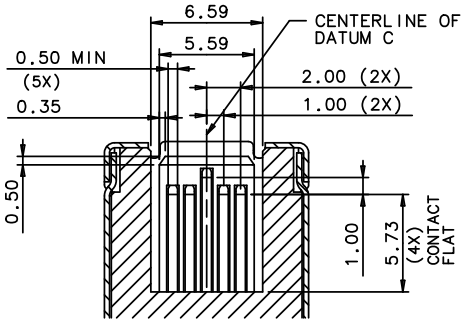
Note: TX and RX are defined from the device perspective

The physical location of the pins in the connector is illustrated in Figure 5-5 to Figure 5-7.

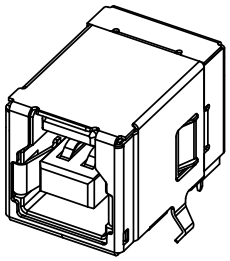
5.3.3 USB 3.0 Power-B Connector

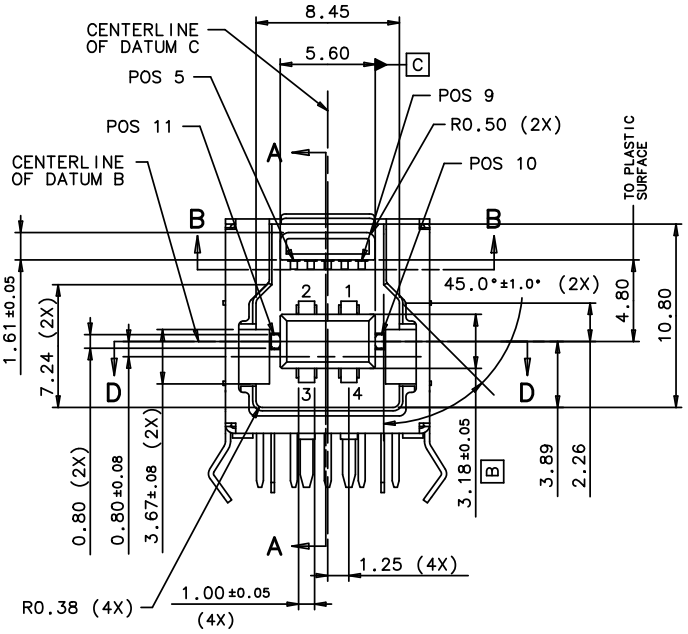
5.3.3.1 Interface Definition

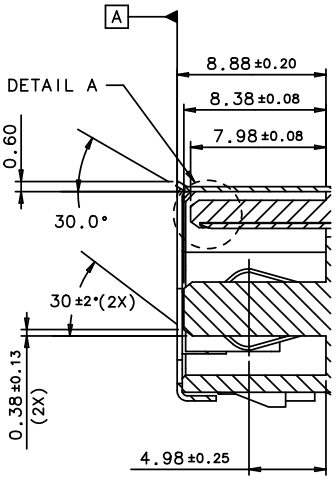
Figure 5-8 to Figure 5-10 show, respectively, the USB 3.0 Power-B receptacle and plug interface dimensions, as well as the reference footprint.



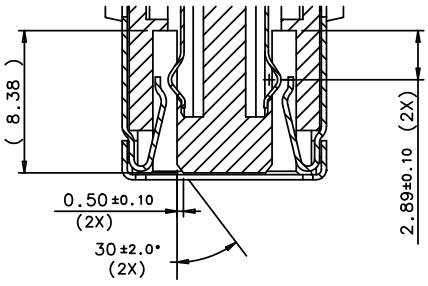
SECTION B-B



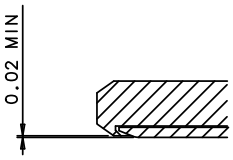




SECTION A-A

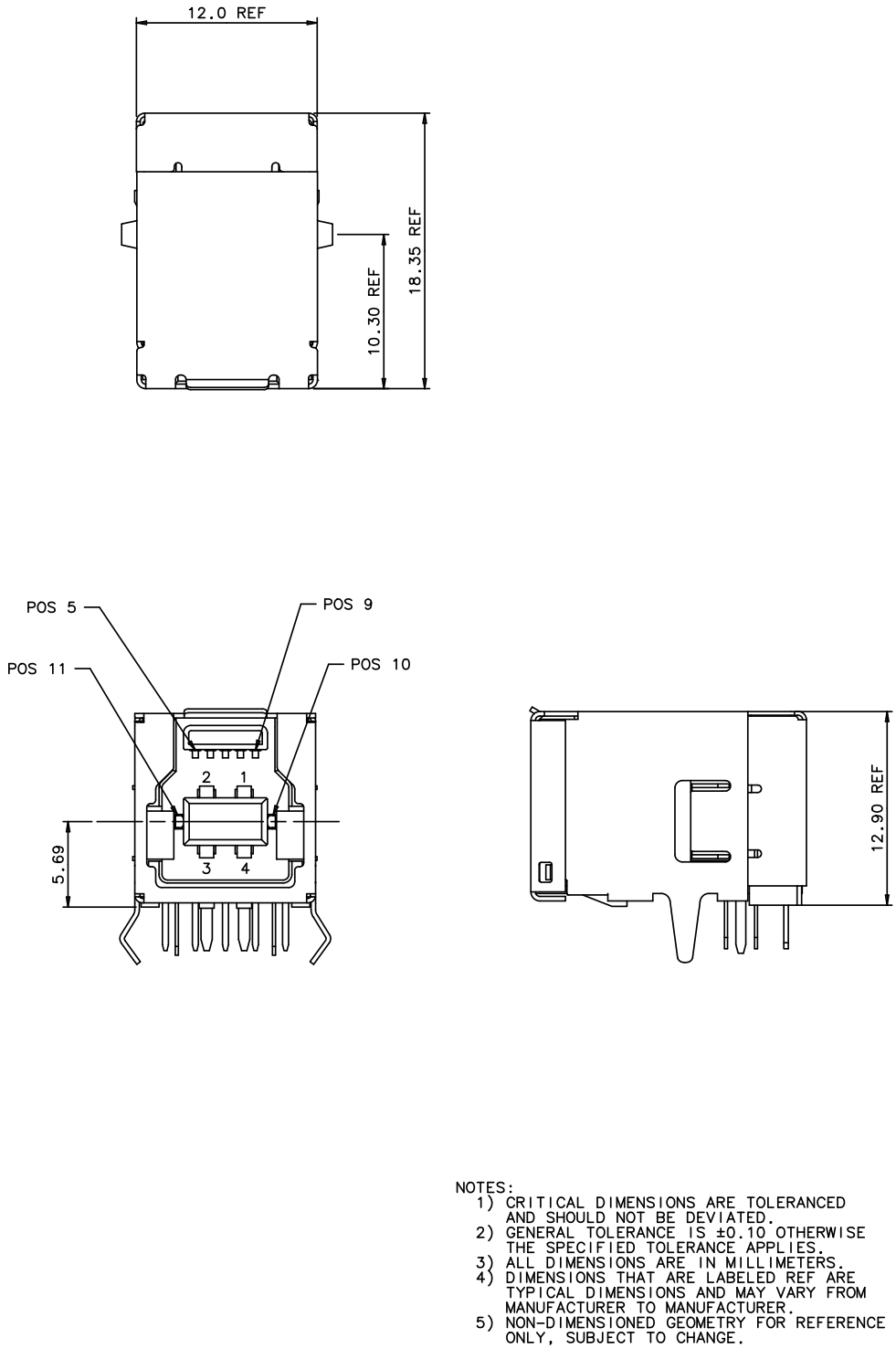


SECTION D-D



DETAIL A

continued



NOTES:
1) CRITICAL DIMENSIONS ARE TOLERANCED AND SHOULD NOT BE DEVIATED.
2) GENERAL TOLERANCE IS ±0.10 OTHERWISE THE SPECIFIED TOLERANCE APPLIES.
3) ALL DIMENSIONS ARE IN MILLIMETERS.
4) DIMENSIONS THAT ARE LABELED REF ARE TYPICAL DIMENSIONS AND MAY VARY FROM MANUFACTURER TO MANUFACTURER.
5) NON-DIMENSIONED GEOMETRY FOR REFERENCE ONLY, SUBJECT TO CHANGE.

Figure 5-8. USB 3.0 Power-B Receptacle Interface Dimensions

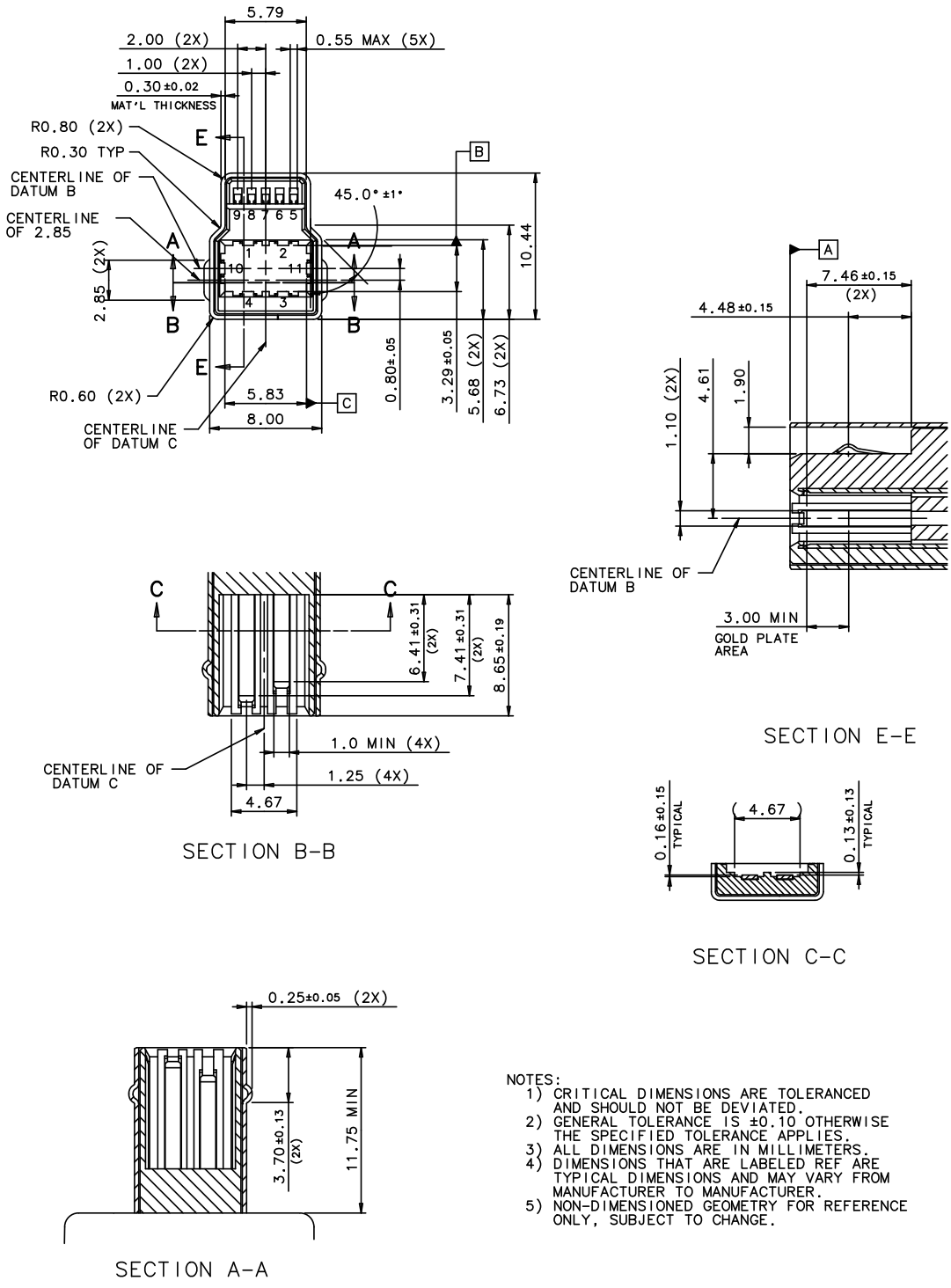


Figure 5-9. USB 3.0 Power-B Plug Interface Dimensions

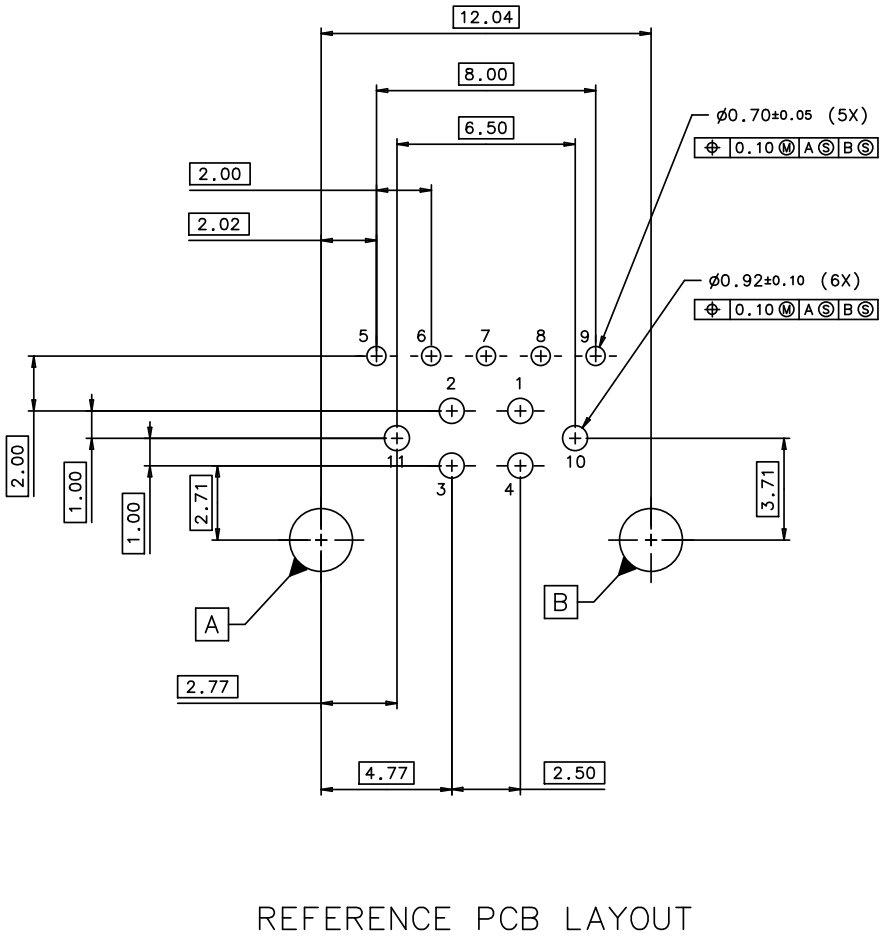


Figure 5-10. Reference Footprint for USB 3.0 Power-B Receptacle

5.3.3.2 Pin Assignments and Descriptions

The usage and assignments of the nine pins in the USB 3.0 Power-B connector are defined in Table 5-4.

Table 5-4. USB 3.0 Power-B Connector Pin Assignments

Pin Number	Signal Name	Description	Mating Sequence
1	VBUS	Power	Second
2	D-	USB 2.0 differential pair	Third or beyond
3	D+		
4	GND	Ground for power return	Second
5	StdB_SSTX-	Super Speed transmitter differential pair	Third or beyond
6	StdB_SSTX+		
7	GND	Ground for signal return	
8	StdB_SSRX-	Super Speed receiver differential pair	
9	StdB_SSRX+		
10	DPWR	Power provided by device	
11	DGND	Ground Return for DPWR	
Shell	Shield	Connector metal shell	First

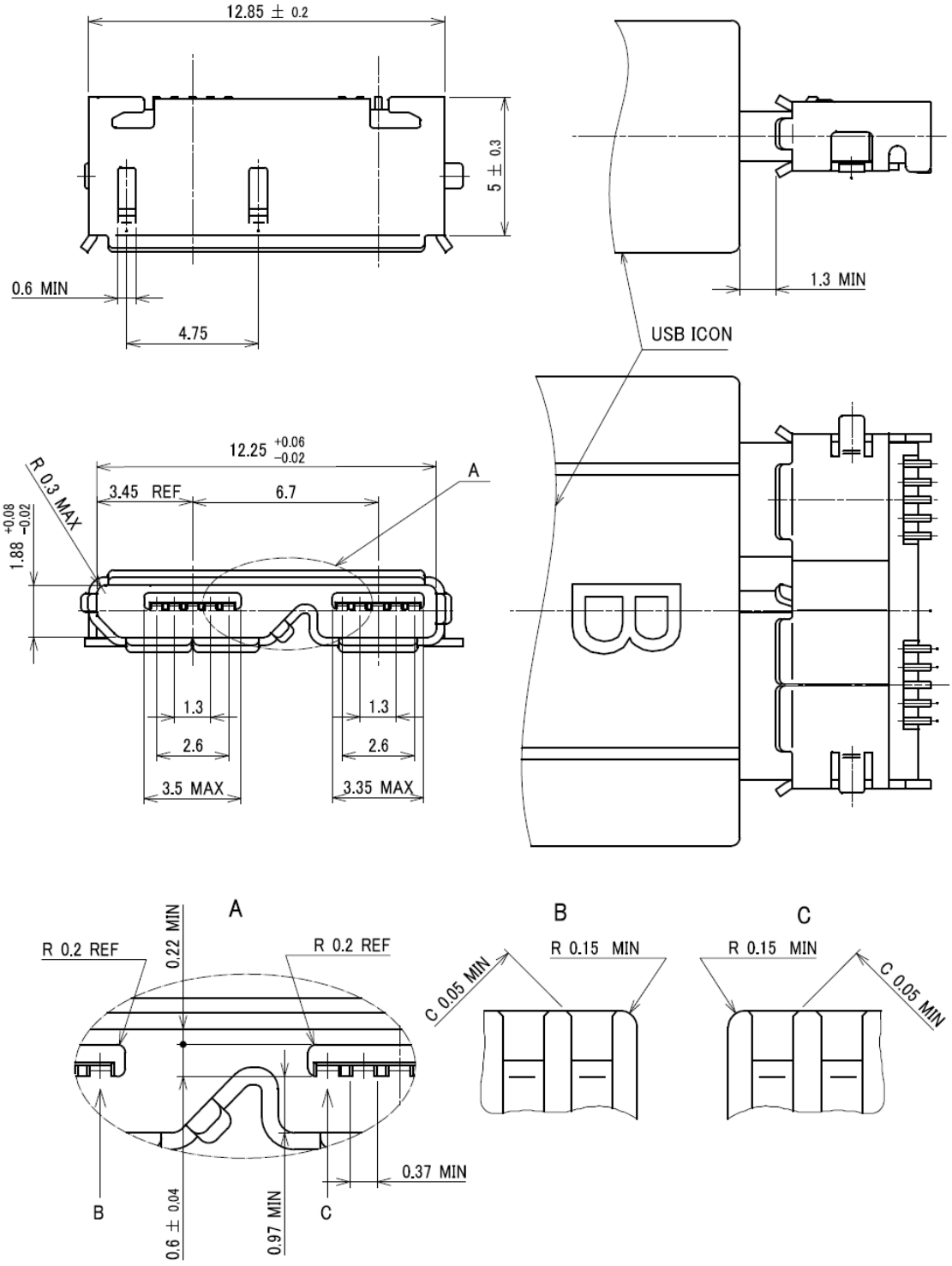
Note: TX and RX are defined from the device perspective

The physical location of the pins in the connector is illustrated in Figure 5-8 to Figure 5-10.

5.3.4 USB 3.0 Micro-B Connector Family

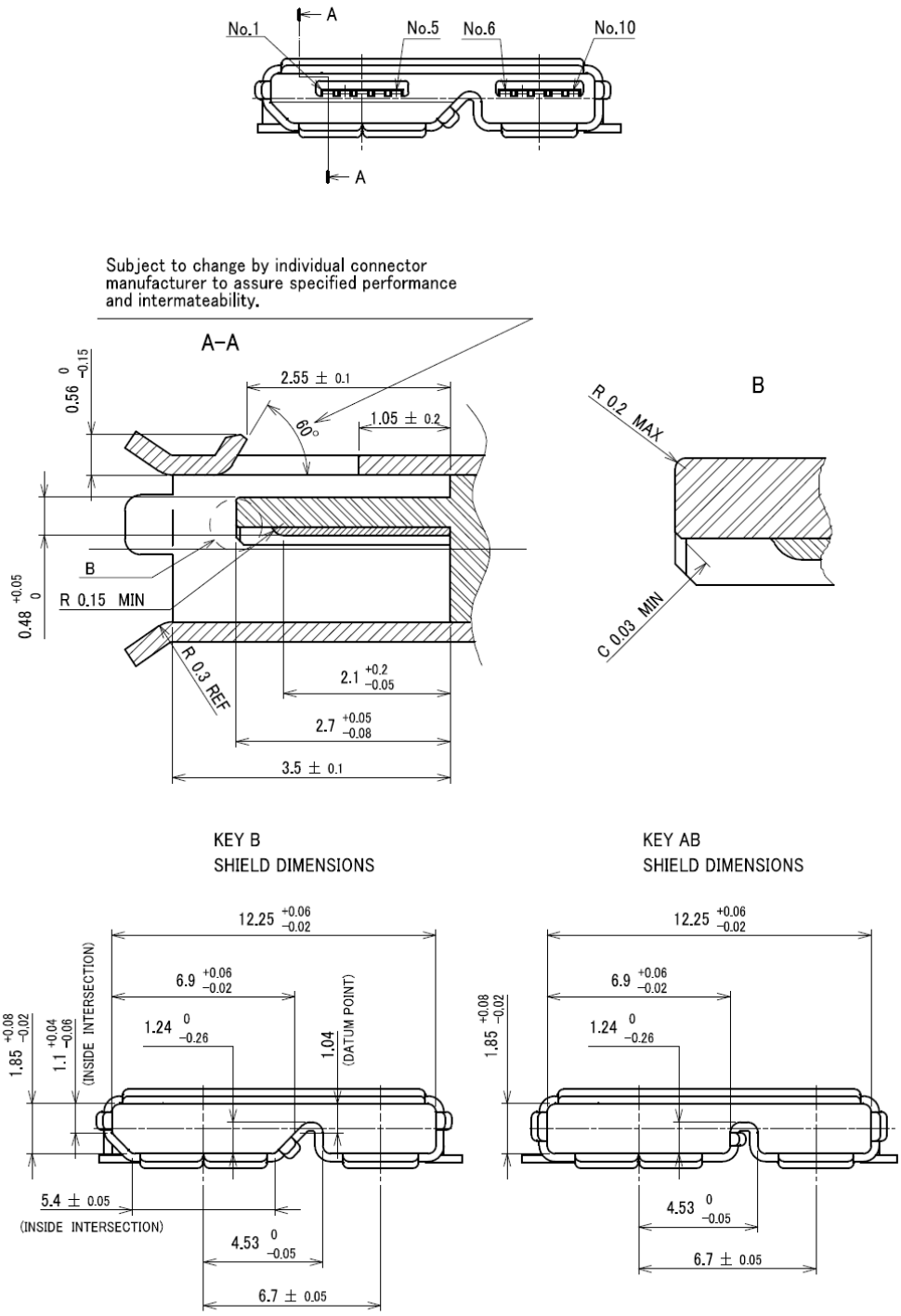
5.3.4.1 Interfaces Definition

The USB 3.0 Micro-B connector family consists of the USB 3.0 Mirco-B receptacle, USB 3.0 Mirco-AB receptacle, USB 3.0 Micro-B plug and USB 3.0 Micro-A plug. Figure 5-11 and Figure 5-12 show the USB 3.0 Micro-B family receptacle and plug interface dimensions. Note that only the dimensions that govern the mating interoperability are specified.



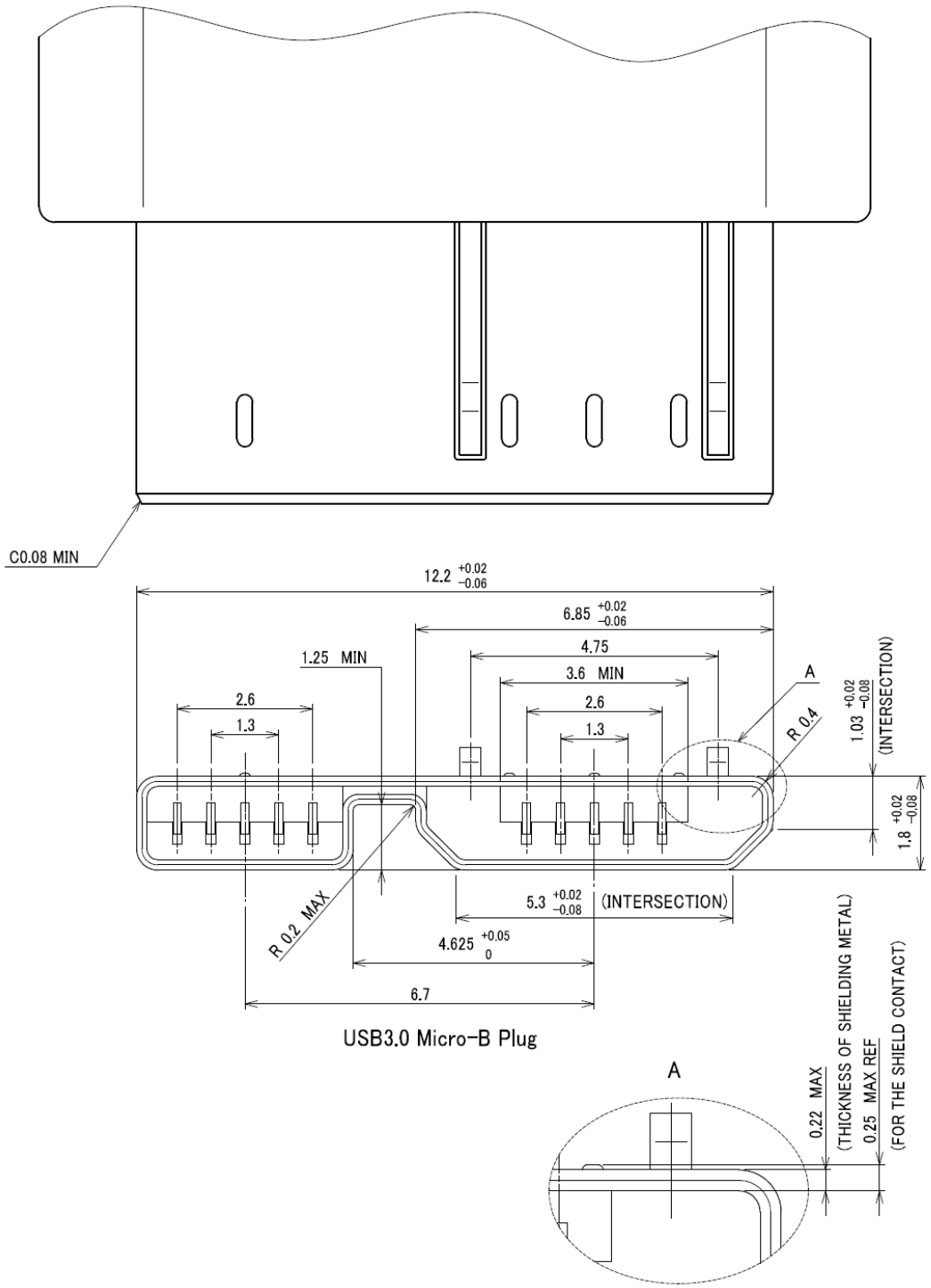
NOTE : General tolerance is +/− 0.05 mm, otherwise the specified tolerances apply.

continued



NOTE : Chamfer metals are optional.

Figure 5-11. USB 3.0 Micro-B and-AB Receptacles Interface Dimensions

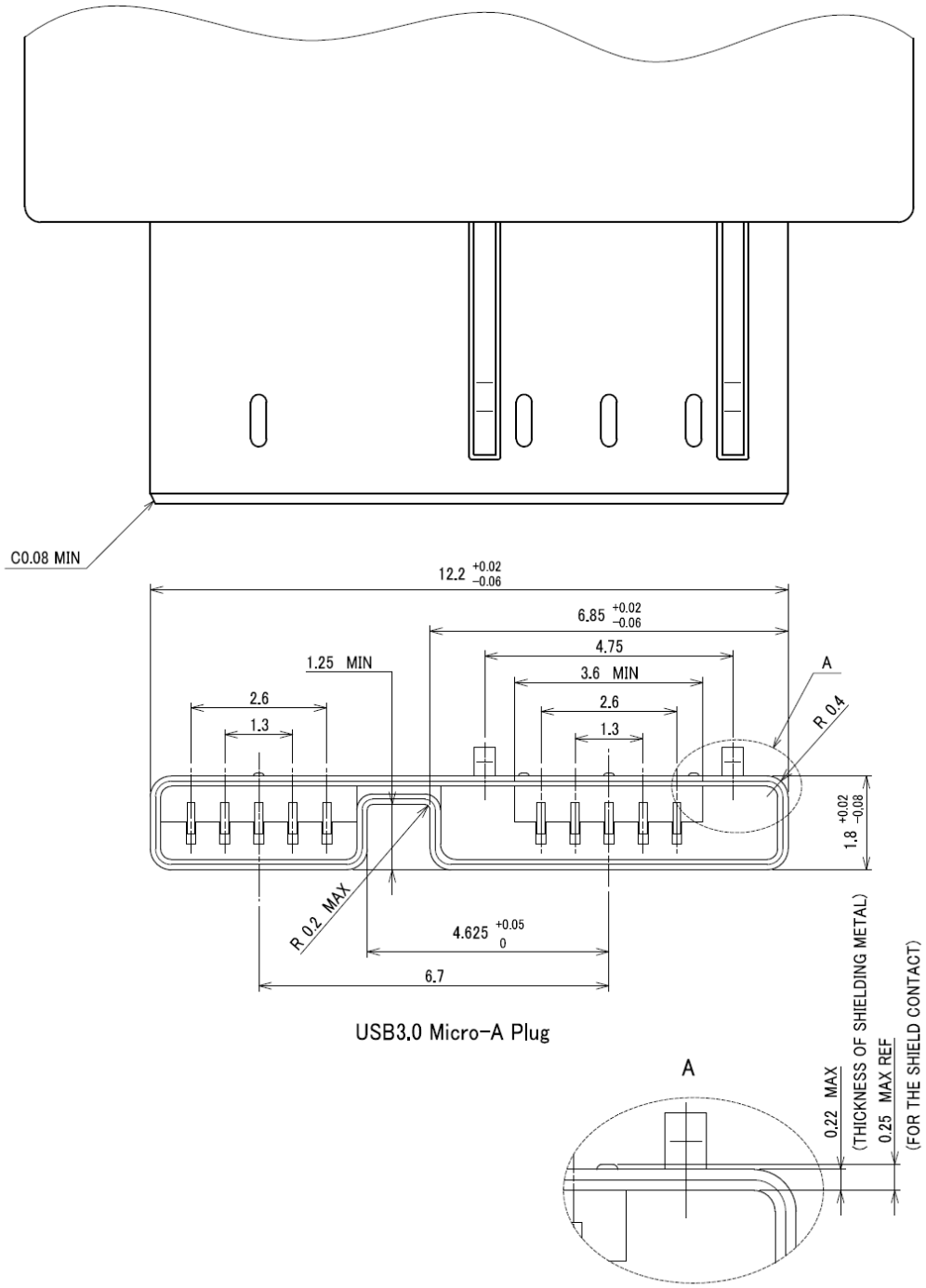


USB3.0 Micro-B Plug

NOTES:

1. Dimensions that are labeled REF may vary from manufacturer to manufacturer.
2. General tolerance is +/−0.05mm, otherwise the specified tolerances apply.

continued



USB3.0 Micro-A Plug

NOTES:

1. Dimensions that are labeled REF may vary from manufacturer to manufacturer.
2. General tolerance is +/−0.05mm, otherwise the specified tolerances apply.

continued

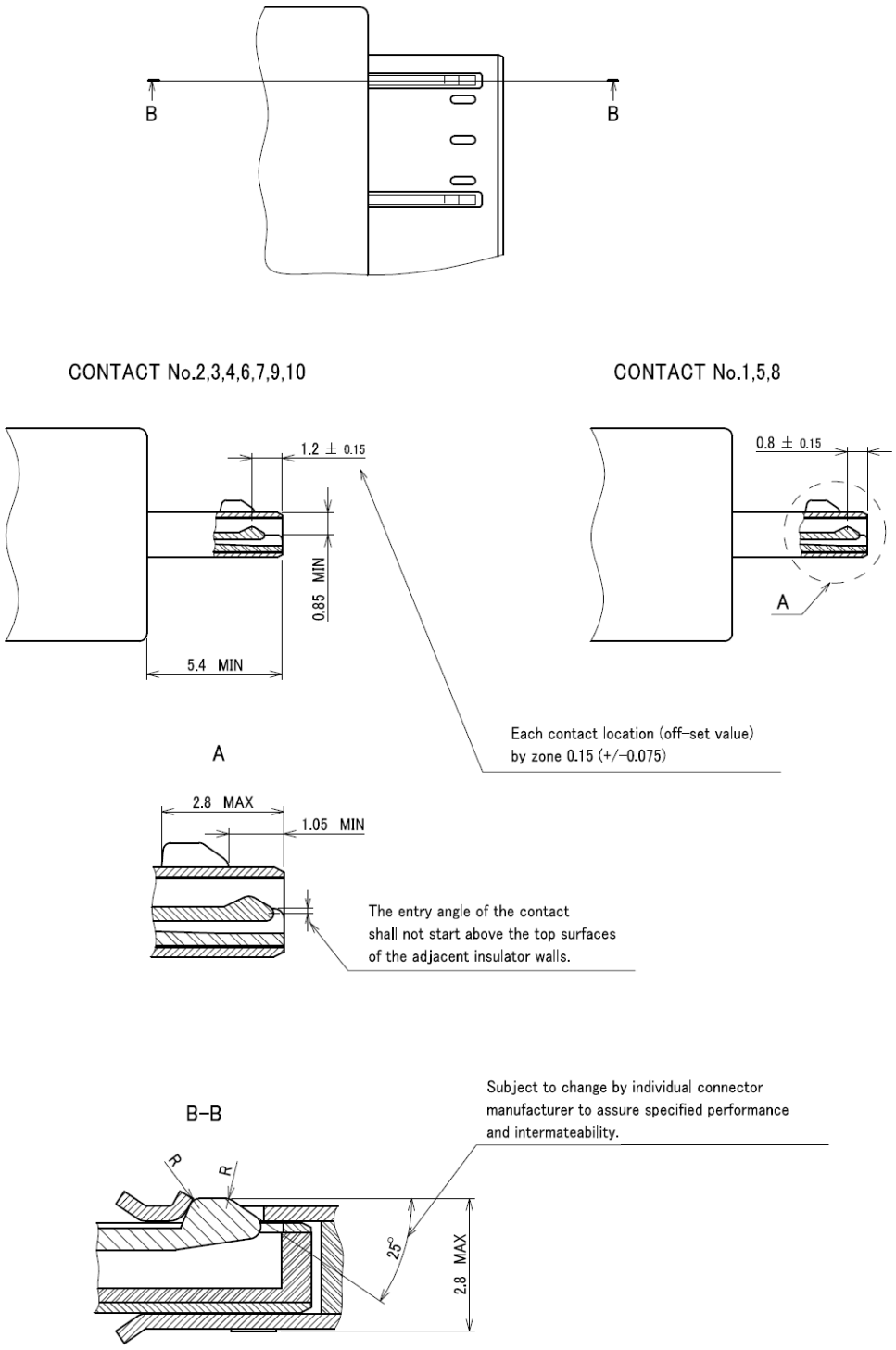
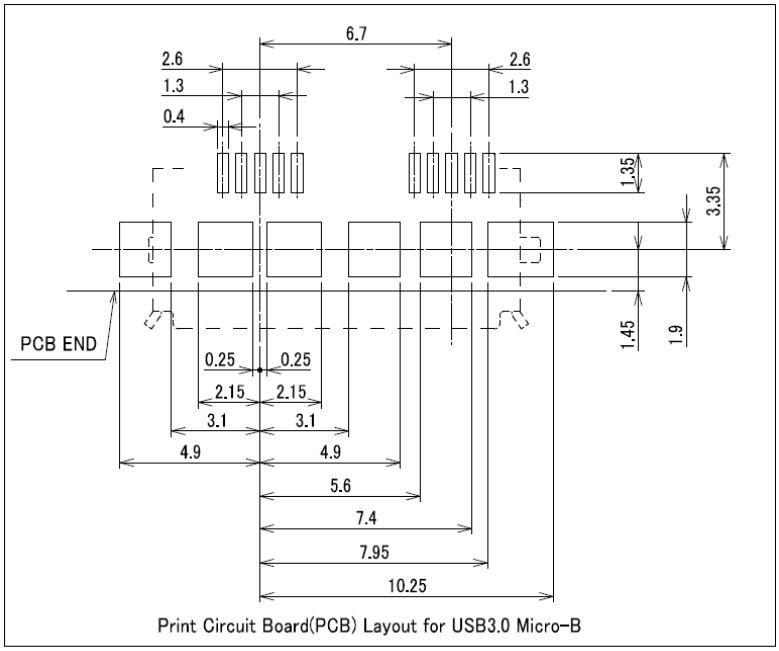
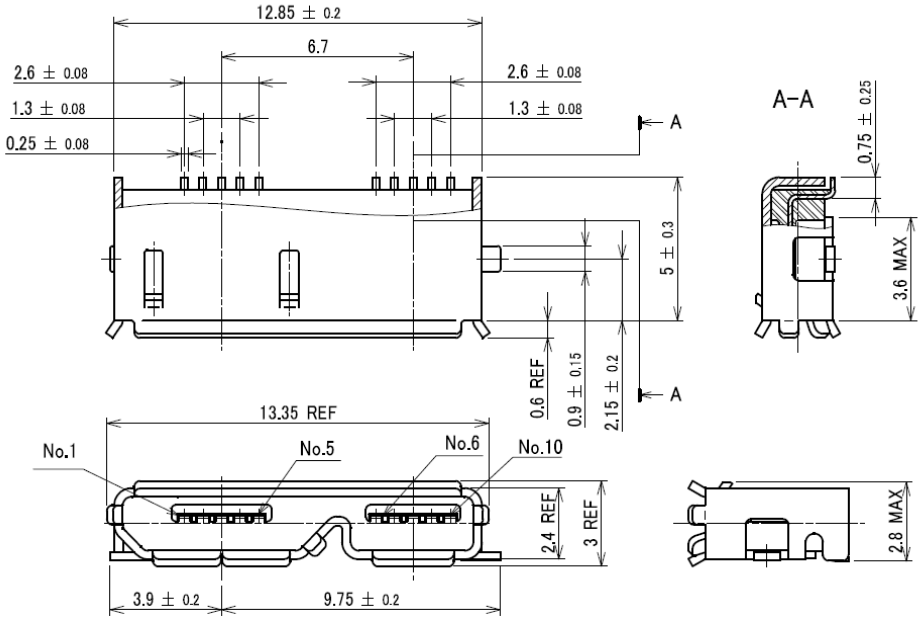


Figure 5-12. USB 3.0 Micro-B and –A Plug Interface Dimensions



Print Circuit Board(PCB) Layout for USB3.0 Micro-B

Standard-Surface Mount-Version Drawing

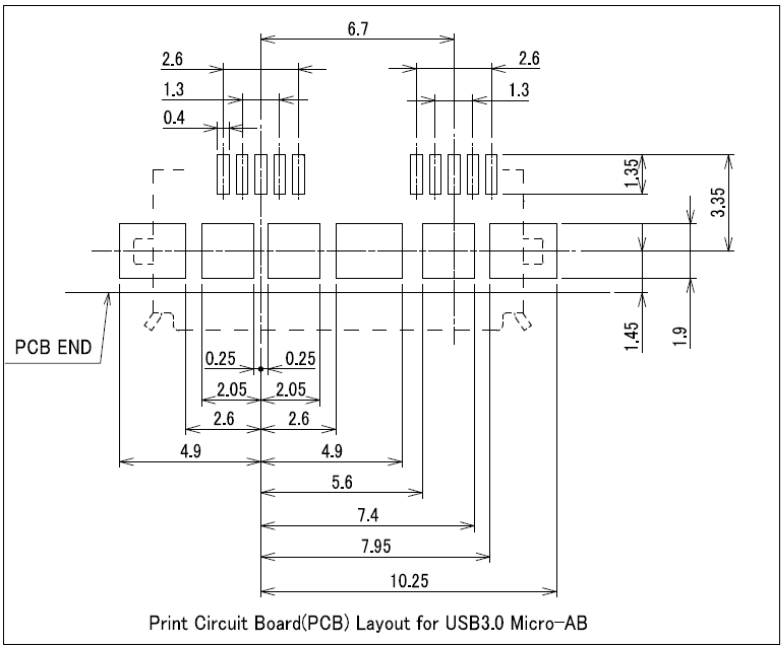


USB3.0 Micro-B RECEPTACLE

NOTES :

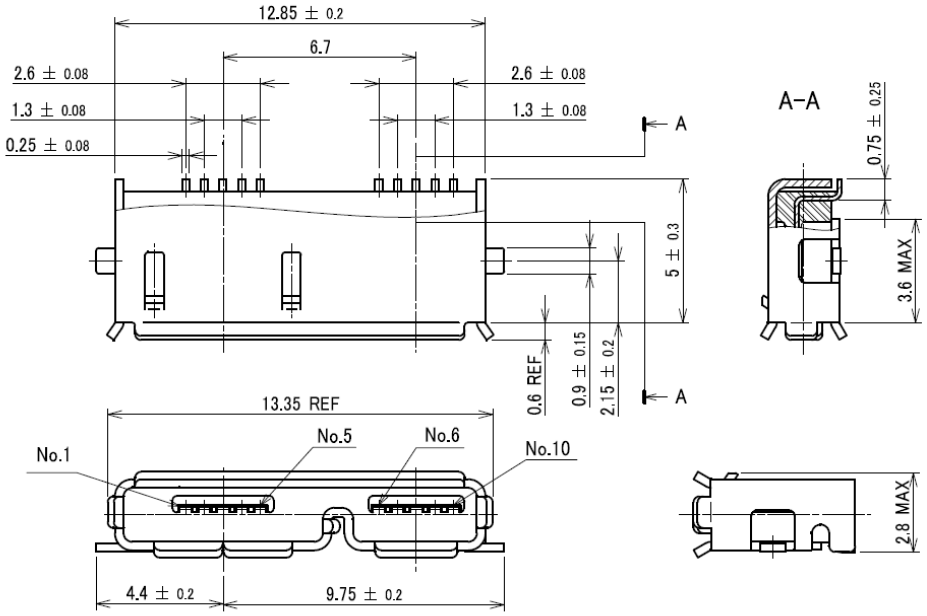
1. Critical Dimensions are TOLERANCED and should not be deviated.
2. Dimensions that labeled REF are typical dimensions and may vary from manufacturer to manufacturer.
3. General tolerance is +/− 0.05 mm, otherwise the specified tolerances apply.
4. Chamfer metals are optional.

continued



Print Circuit Board(PCB) Layout for USB3.0 Micro-AB

Standard-Surface Mount-Version Drawing



USB3.0 Micro-AB RECEPTACLE

NOTES :

1. Critical Dimensions are TOLERANCED and should not be deviated.
2. Dimensions that labeled REF are typical dimensions and may vary from manufacturer to manufacturer.
3. General tolerance is +/- 0.05 mm, otherwise the specified tolerances apply.
4. Chamfer metals are optional.

Figure 5-13. Reference Footprint for the USB 3.0 Micro-B or -AB Receptacle

The USB 3.0 Micro-B connector family has the following characteristics:

- The USB 3.0 Micro-B connector may be considered a combo of USB 2.0 Micro-B interface plus the USB 3.0 SS contacts. The USB 3.0 Micro-B receptacle accepts a USB 2.0 Micro-B plug, maintaining backward compatibility.
- The USB 3.0 Micro-B connector maintains the same connector height and contact pitch as the USB 2.0 Micro-B connector.
- The USB 3.0 Micro-B connector uses the same, proven latch design as the USB 2.0 Micro-B connector.
- The USB 3.0 Micro-AB receptacle is identical to the USB 3.0 Micro-B receptacle except for keying difference in the connector shell outline.
- The USB 3.0 Micro-A plug is similar to the USB 3.0 Micro-B plug with different keying and ID pin connections. Section 5.3.4.2 discusses the ID pin connections.
- There is no required footprint for the USB 3.0 Micro-B connector family. Figure 5-13 shows reference Micro-B and -AB connector footprints.

5.3.4.2 Pin Assignments and Description

Table 5-5 and Table 5-6 show the pin assignments for the USB 3.0 Micro-B connector family.

Table 5-5. USB 3.0 Micro-B Connector Pin Assignments

Pin Number	Signal Name	Description	Mating Sequence
1	VBUS	Power	Second
2	D-	USB 2.0 differential pair	Last
3	D+		
4	ID	OTG identification	
5	GND	Ground for power return	Second
6	MicB_SSTX-	Super Speed transmitter differential pair	Last
7	MicB_SSTX+		
8	GND	Ground for SS signal return	Second
9	MicB_SSRX-	Super Speed receiver differential pair	Last
10	MicB_SSRX+		
Shell	Shield	Connector metal shell	First

Note: TX and RX are defined from the device perspective

Table 5-6. USB 3.0 Micro-AB/-A Connector Pin Assignments

Pin Number	Signal Name	Description	Mating Sequence
1	VBUS	Power	Second
2	D-	USB 2.0 differential pair	Last
3	D+		
4	ID	OTG identification	
5	GND	Ground for power return	Second
6	MicA_SSRX-	Super Speed receiver differential pair	last
7	MicA_SSRX+		
8	GND	Ground for SS signal return	Second
9	MicA_SSTX-	Super Speed transmitter differential pair	Last
10	MicA_SSTX+		
Shell	Shield	Connector metal shell	First

Note: TX and RX are defined when an OTG device serves as a host.

The physical location of the pins in the connector is illustrated in Figure 5-11 to Figure 5-13.

5.4 Cable Construction and Wire Assignments

This section discusses the USB 3.0 cables, including cable construction, wire assignments, and wire gauges. The performance requirements will be specified in Section 5.6.1.1.

5.4.1 Cable Construction

Figure 5-14 illustrates a USB 3.0 cable cross-section. There are three groups of wires: UTP (unshielded twist pair) signal pair, SDP (shielded differential pair, twist or twinax) signal pairs, and power and ground return wires.

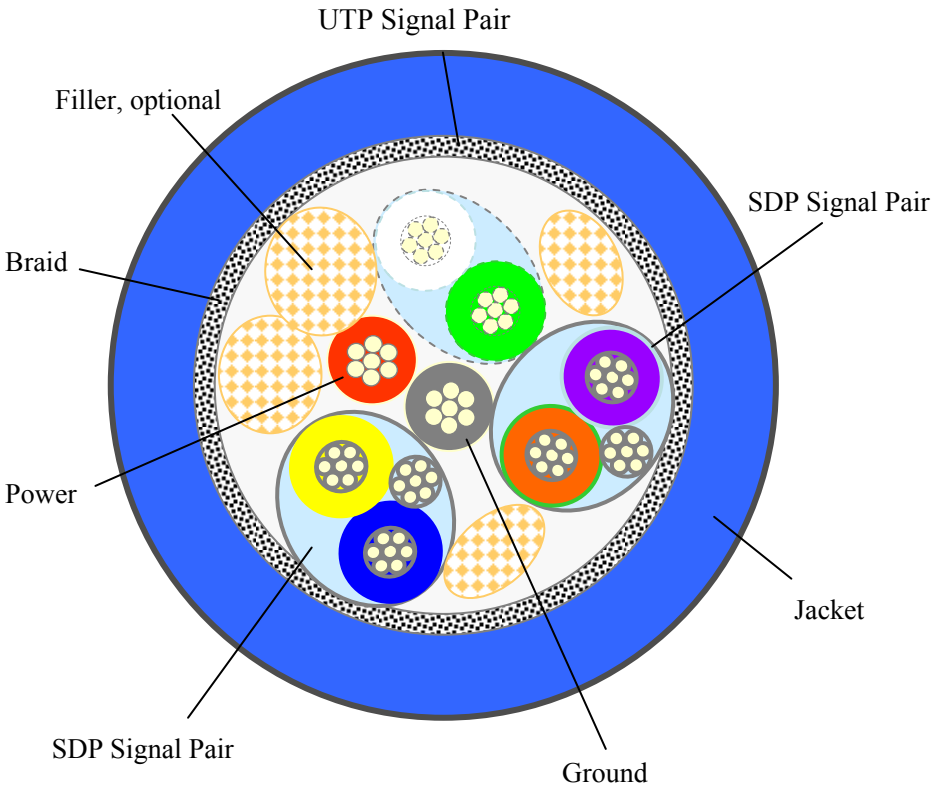


Figure 5-14. Illustration of a USB 3.0 Cable Cross-Section

The UTP is intended to transmit the USB 2.0 signaling while the SDP's are for the SS use; the shield is needed for the SS differential pairs for signal integrity and EMI performance. Each SDP is attached with a drain wire, which is eventually connected to the system ground through the GND pin(s) in the connector.

A metal braid is required to enclose all the wires in the USB 3.0 cable. The braid is to be terminated to the plug metal shells, as close to 360° as possible, to contain EMI.

5.4.2 Wire Assignments

Table 5-7 defines the wire number, signal assignments and colors.

Table 5-7. Cable Wire Assignments

Wire Number	Signal Name	Description	Color
1	PWR	Power	Red
2	UTP_D-	Unshielded twist pair, negative	White
3	UTP_D+	Unshielded twist pair, positive	Green
4	GND_PWRrt	Ground for power return	Black
5	SDP1-	Shielded differential pair1, negative	Blue
6	SDP1+	Shielded differential pair1, positive	Yellow
7	SDP1_Drain	Drain wire for SDP1	
8	SDP2-	Shielded differential pair2, negative	Purple
9	SDP2+	Shielded differential pair2, positive	Orange
10	SDP2_Drain	Drain wire for SDP2	
Braid	Shield	Cable external braid to be 360° terminated on to plug metal shell	

5.4.3 Wire Gauges and Cable Diameters

This specification chooses not to specify wire gauges. Table 5-8 lists the typical wire gauges ***for reference***. A large gauge wire incurs less loss, but at the cost of cable flexibility. One should choose the smallest possible wire gauges that meet the cable assembly electrical requirements.

To maximize cable flexibility, all wires are required to be stranded and the cable outer diameter should be minimized as much as possible. A typical USB 3.0 cable outer diameter may range from 3 to 6 mm.

Table 5-8. Reference Wire Gauges

Wire Number	Signal Name	Wire Gauge (AWG)
1	PWR	20-28
2	UTP_D-	28-34
3	UTP_D+	28-34
4	GND_PWRrt	20-28
5	SDP1-	26-34
6	SDP1+	26-34
7	SDP1_Drain	28-34
8	SDP2-	26-34
9	SDP2+	26-34
10	SDP2_Drain	28-34

5.5 Cable Assemblies

5.5.1 USB 3.0 Standard-A to USB 3.0 Standard-B Cable Assembly

Figure 5-15 shows a USB 3.0 Standard-A to USB 3.0 Standard-B cable assembly.

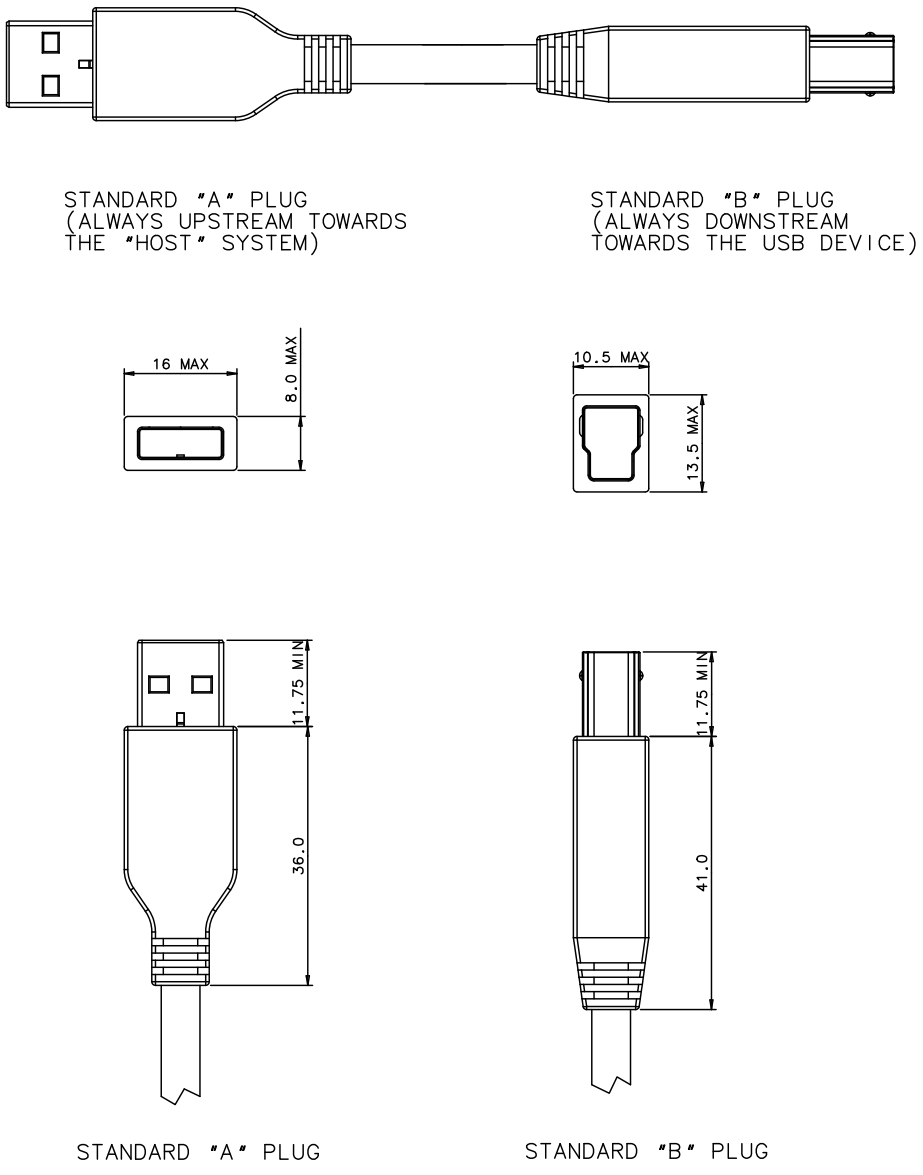


Figure 5-15. USB 3.0 Standard-A to USB 3.0 Standard-B Cable Assembly

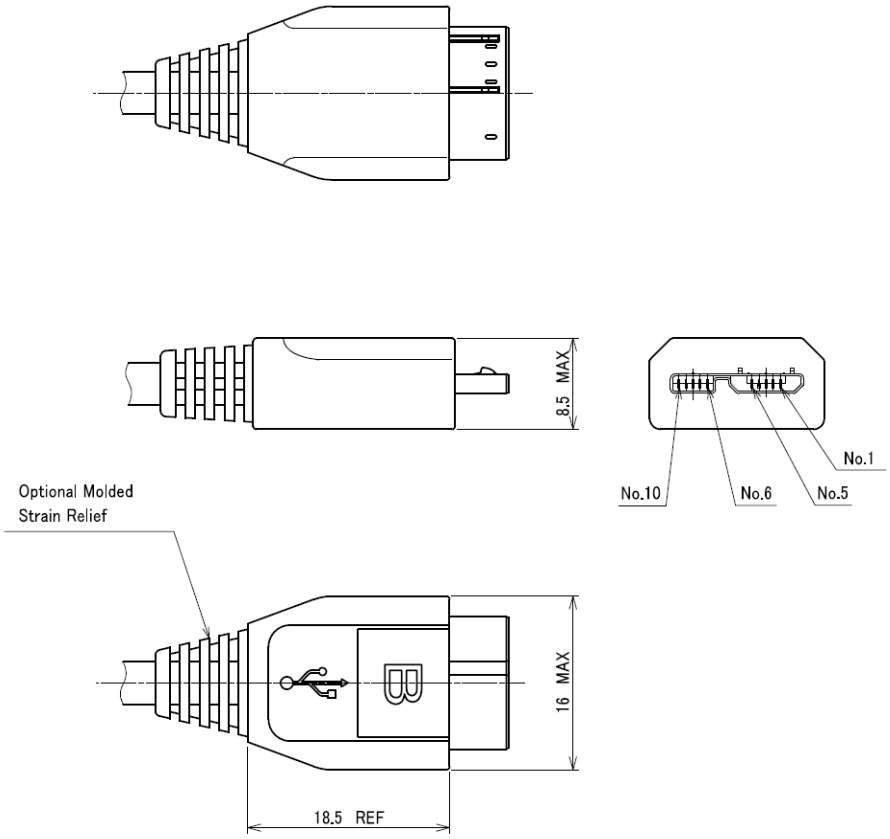
Table 5-9 defines the wire connections for the USB 3.0 Standard-A to USB 3.0 Standard-B cable assembly.

Table 5-9. USB 3.0 Standard-A to USB 3.0 Standard-B Cable Assembly Wiring

USB 3.0 Standard-A Plug		Wire		USB 3.0 Standard-B Plug	
Pin Number	**Signal Name**	**Wire Number**	**Wire Name**	**Pin Number**	**Signal Name**
1	VBUS	1	PWR	1	VBUS
2	D-	2	UTP_D-	2	D-
3	D+	3	UTP_D+	3	D+
4	GND	4	GND_PWRrt	4	GND
5	StdA_SSRX-	5	SDP1-	5	StdB_SSTX-
6	StdA_SSRX+	6	SDP1+	6	StdB_SSTX+
7	GND	7 & 10	SDP1_Drain SDP2_Drain	7	GND
8	StdA_SSTX-	8	SDP2-	8	StdB_SSRX-
9	StdA_SSTX+	9	SDP2+	9	StdB_SSRX+
Shell	Shield	Braid	Braid	Shell	Shield

5.5.2 USB 3.0 Standard-A to USB 3.0 Micro-B Cable Assembly

Figure 5-16 shows the USB 3.0 Micro-B plug overmold dimensions for a USB 3.0 Standard-A to USB 3.0 Micro-B cable assembly. The USB 3.0 Standard-A plug overmold dimensions can be found in Figure 5-15.



NOTES:
1. Any surface can have texturing up to 0.3mm below the surface.
2. A square area around the letter 'B' can be lowered by as much as 0.5mm.
3. USB authorized logo mark, connector type letter designation– (A or B), color of the insulator body and maximum dimensions are mandatory. Overmolding outer configuration, color and final shape are reference.
4. Pin 4 is not connected to pin 5 inside the plug.

Figure 5-16. USB 3.0 Micro-B Plug Cable Overmold Dimensions

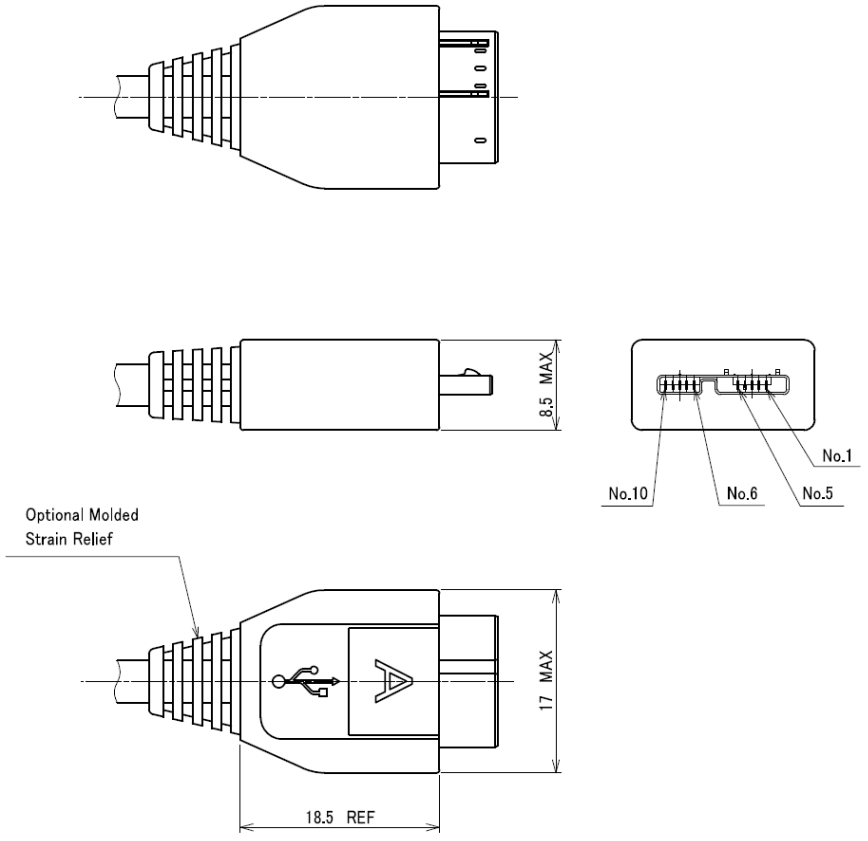
Table 5-10 shows the wire connections for the USB 3.0 Standard-A to USB 3.0 Micro-B cable assembly. Note that the ID pin in the USB 3.0 Micro-B plug should be a no-connect, leaving open.

Table 5-10. USB 3.0 Standard-A to USB 3.0 Micro-B Cable Assembly Wiring

USB 3.0 Standard-A Plug		Wire		USB 3.0 Micro-B Plug	
Pin Number	**Signal Name**	**Wire Number**	**Wire Name**	**Pin Number**	**Signal Name**
1	VBUS	1	PWR	1	VBUS
2	D-	2	UTP_D-	2	D-
3	D+	3	UTP_D+	3	D+
4	GND	4	GND_PWRrt	5	GND
5	StdA_SSRX-	5	SDP1-	6	MicB_SSTX-
6	StdA_SSRX+	6	SDP1+	7	MicB_SSTX+
7	GND	7 & 10	SDP1_Drain SDP2_Drain	8	GND
8	StdA_SSTX-	8	SDP2-	9	MicB_SSRX-
9	StdA_SSTX+	9	SDP2+	10	MicB_SSRX+
				4	ID
Shell	Shield	Braid	Braid	Shell	Shield

5.5.3 USB 3.0 Micro-A to USB 3.0 Micro-B Cable Assembly

Figure 5-17 shows the USB 3.0 Micro-A plug cable overmold dimensions in a USB 3.0 Micro-A to USB 3.0 Micro-B cable assembly. The USB 3.0 Micro-B plug cable overmold dimensions are shown in Figure 5-16.



NOTES:
1. Any surface can have texturing up to 0.3mm above the surface.
2. A square area around the letter 'A' can be raised as much as 0.5mm above the surface.
3. USB authorized logo mark, connector type letter designation– (A or B), color of the insulator body and maximum dimensions are mandatory. Overmolding outer configuration, color and final shape are reference.
4. Pin 4 is connected to pin 5 inside the plug.

Figure 5-17. USB 3.0 Micro-A Cable Overmold Dimensions

Table 5-11 shows the wire connections for the USB 3.0 Micro-A to USB 3.0 Micro-B cable assembly. The ID pin on a USB 3.0 Micro-A plug shall be connected to the GND pin. The ID pin on a USB 3.0 Micro-B plug is not connected or is connected to ground by a resistance of greater than Rb_PLUG_ID (100kΩ MIN). An On-The-Go device is required to be able to detect whether a USB 3.0 Micro-A or USB 3.0 Micro-B plug is inserted by determining if the ID pin resistance to ground is less than Ra_PLUG_ID (10Ω MAX) or if the resistance to ground is greater than Rb_PLUG_ID. Any ID resistance less than Ra_PLUG_ID shall be treated as ID = FALSE and any resistance greater than Rb_PLUG_ID shall be treated as ID = TRUE.

Table 5-11. USB 3.0 Micro-A to USB 3.0 Micro-B Cable Assembly Wiring

USB 3.0 Micro-A Plug		Wire		USB 3.0 Micro-B Plug	
Pin Number	**Signal Name**	**Wire Number**	**Wire Name**	**Pin Number**	**Signal Name**
1	VBUS	1	PWR	1	VBUS
2	D-	2	UTP_D-	2	D-
3	D+	3	UTP_D+	3	D+
4	ID (see Note 1)	No Connect		4	ID (See Note 2)
5	GND	4	GND_PWRrt	5	GND
6	MicA_SSRX-	5	SDP1-	6	MicB_SSTX-
7	MicA_SSRX+	6	SDP1+	7	MicB_SSTX+
8	GND	7 & 10	SDP1_Drain SDP2_Drain	8	GND
9	MicA_SSTX-	8	SDP2-	9	MicB_SSRX-
10	MicA_SSTX+	9	SDP2+	10	MicB_SSRX+
Shell	Shield	Braid	Braid	Shell	Shield

Notes:

1. Connect to the GND or Shield
2. No connect

5.5.4 USB 3.0 Micro-A to USB 3.0 Standard-B Cable Assembly

A USB 3.0 Micro-A to USB 3.0 Standard-B cable assembly is also allowed. Figure 5-17 and Figure 5-15 show, respectively, the UBS 3.0 Micro-A cable overmold and the USB 3.0 Standard-B cable overmold.

Table 5-12 shows the wire connections for the USB 3.0 Micro-A to USB 3.0 Standard-B cable assembly.

Table 5-12. USB 3.0 Micro-A to USB 3.0 Standard-B Cable Assembly Wiring

USB 3.0 Micro-A Plug		Wire		USB 3.0 Standard-B Plug	
Pin Number	**Signal Name**	**Wire Number**	**Wire Name**	**Pin Number**	**Signal Name**
1	VBUS	1	PWR	1	VBUS
2	D-	2	UTP_D-	2	D-
3	D+	3	UTP_D+	3	D+
4	ID (see Note 1)	No Connect			
5	GND	4	GND_PWRrt	4	GND
6	MicA_SSRX-	5	SDP1-	5	MicB_SSTX-
7	MicA_SSRX+	6	SDP1+	6	MicB_SSTX+
8	GND	7 & 10	SDP1_Drain SDP2_Drain	7	GND
9	MicA_SSTX-	8	SDP2-	8	MicB_SSRX-
10	MicA_SSTX+	9	SDP2+	9	MicB_SSRX+
Shell	Shield	Braid	Braid	Shell	Shield

Notes:

1. Connect to the GND or Shield

5.5.5 Captive Cables

Three captive cables are allowed in this specification: the captive cable with the USB 3.0 Standard-A plug, the captive cable with the USB 3.0 Micro-A plug, and the captive cable with the USB 3.0 Power-B plug. All other captive cables are prohibited.

5.5.6 Cable Assembly Length

This specification does not specify cable assembly lengths. A USB 3.0 cable assembly can be of any length, as long as it meets all the requirements defined in this specification. The cable assembly loss budget defined in Section 5.6.1.1.3 and the cable voltage drop budget defined in Chapter 11.4.2 will limit the cable assembly length.

5.6 Electrical Requirements

This section covers the electrical requirements for USB 3.0 raw cables, mated connectors, and mated cable assemblies. USB 3.0 signals, known as Super Speed (SS) 5 Gb/s, are governed by USB 3.0 specification. The USB 2.0 signals, Full Speed (FS) 12 Mb/s and High Speed (HS) 480 Mb/s, are governed by the USB 2.0 specification.

USB 3.0 specification compliance is established through normative requirements of mated connectors and mated cable assemblies. USB 3.0 (SS) requirements are specified mainly in terms of S-parameters, using industry test specification with supporting details when required. USB 3.0 DC requirements, such as contact resistance and current carrying capability are also specified in this section.

Any informative specification for cable and connector products is for the purpose of design guidelines and manufacturing control.

In conjunction with performance requirements, the required test method is referenced for the parameter stated. A list of the industry standards for DC requirements is found in the Section 5.6.2. Additional supporting test procedures can be found in "USB 3.0 Connectors and Cable Assemblies Compliance Document".

5.6.1 High Speed Electrical Requirements

The high speed requirements apply only to the SS signals. The requirements for the USB 2.0 signals are governed by the USB 2.0 specification.

5.6.1.1 Raw Cable

Informative raw cable electrical performance targets are provided here to help cable assembly manufacturers manage raw cable suppliers. *Those targets are not part of the USB 3.0 compliance items; the ultimate requirements will be the mated cable assembly performance to be specified in Section 5.6.1.3.*

5.6.1.1.1 Characteristic Impedance

The differential characteristic impedance for the SDP pairs is recommended to be within 90+/- 7 ohms. It should be measured with a TDR in a differential mode using a 200 ps (10-90%) rise time.

5.6.1.1.2 Intra-Pair Skew

The intra-pair skew for the SDP pairs is recommended to be less than 15 ps per meter. It should be measured with a TDT in a differential mode using a 200 ps (10-90%) rise time with a crossing at 50% of the input voltage.

5.6.1.1.3 Differential Insertion Loss

Cable loss depends on wire gauges and dielectric materials. Table 5-13 lists the examples of differential insertion losses for the SDP pairs. Note that the differential loss values are referenced to a 90-ohm differential impedance.

Table 5-13. SDP Differential Insertion Loss Examples

Frequency	34AWG	30AWG	28AWG	26AWG
0.625 GHz	2.7 dB/m	1.3 dB/m	1.0 dB/m	0.9 dB/m
1.25 GHz	3.3 dB/m	1.9 dB/m	1.5 dB/m	1.3 dB/m
2.50 GHz	4.4 dB/m	3.0 dB/m	2.5 dB/m	1.9 dB/m
5.00 GHz	6.7 dB/m	4.6 dB/m	3.6 dB/m	3.1 dB/m
7.50 GHz	9.0 dB/m	5.9 dB/m	4.7 dB/m	4.2 dB/m

5.6.1.2 Mated Connector

The mated connector impedance is required to control impedance mismatch. The differential impedance of a mated connector shall be within 90+/-13.5 ohms, as seen from a 50 ps (20-80%) risetime of a differential TDR. Figure 5-18 illustrates the impedance limits of a mated connector. The impedance profile of a mated connector must fall within the limits shown in Figure 5-18. Note that the impedance profile of the mated connector is defined from the receptacle footprints through the plug cable termination area. In the case the plug is directly attached to a device PCB, the mated connector impedance profile includes the path from the receptacle footprints to the plug footprints.

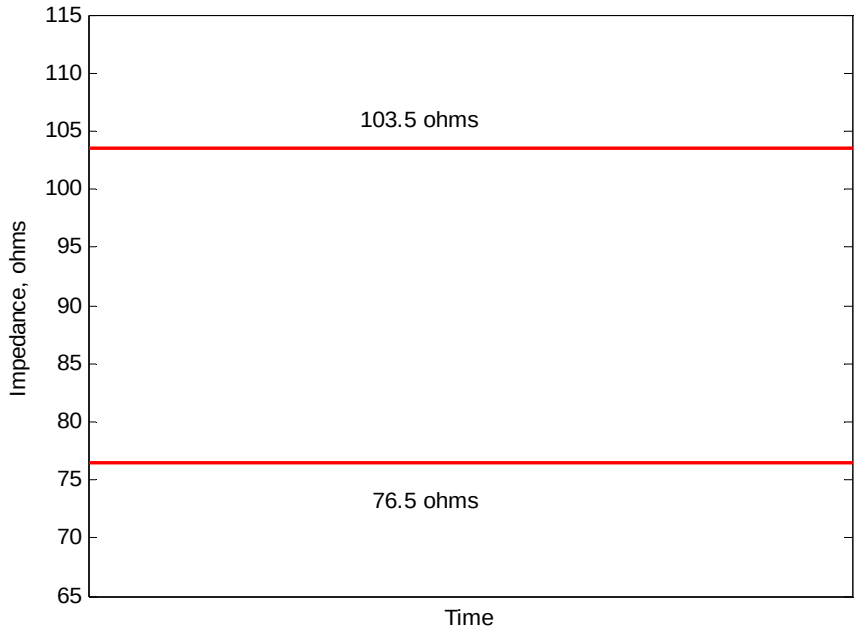


Figure 5-18. Impedance limits of a Mated Connector

5.6.1.3 Mated Cable Assemblies

A mated cable assembly refers to a cable assembly mated with the corresponding receptacles mounted on a test board fixture at the both ends. The requirements are for the entire signal path of the mated cable assembly, from the host receptacle contact solder pads or through-holes on the host system board to the device receptacle contact solder pads or through holes on the device system board, not including PCB traces, as illustrated in Figure 5-19; the measurement is between TP1 (test point 1) and TP2 (test point 2).

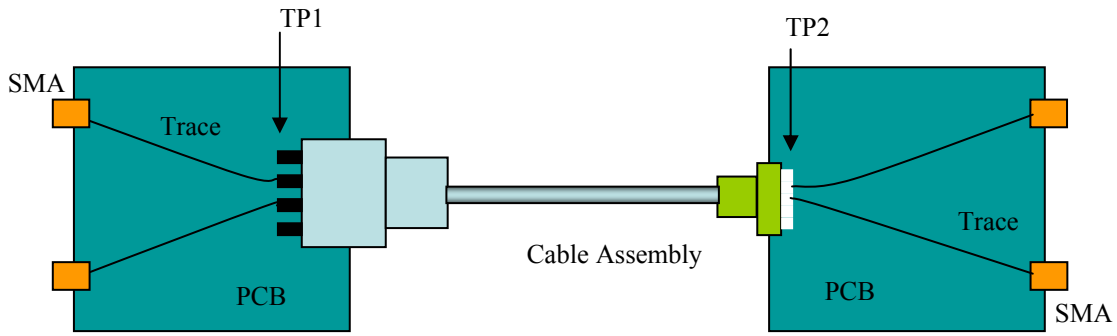


Figure 5-19. Illustration of Test Points for a Mated Cable Assembly

For proper measurements, the receptacles shall be mounted on a test fixture. The test fixture shall have uncoupled access traces from SMA or microprobe launches to the test points, preferably with 50 Ω +/-7% ohms single-ended characteristic impedance. The test fixture shall have appropriate calibration structures to calibrate out the fixturing effect. All non-ground pins that are adjacent but not connected to measurement ports shall be terminated with 50 Ω loads.

A reference USB 3.0 mated cable assembly test fixture will be defined in the "USB 3.0 Connectors and Cable Assemblies Compliance Document", in which the detailed testing procedures will be given.

5.6.1.3.1 Differential Return Loss (EIA-360-108) (Reference)

The differential return loss, SDD11, measures the impedance mismatch in frequency domain. The differential return loss envelop is defined by following vertices: (100MHz, -15 dB), (f0/2, -15 dB), (f0, -12 dB), (2*f0, -5 dB) and (3*f0, -2 dB), where f0=2.5 GHz being the fundamental frequency. This is shown in Figure 5-20; the measured differential return loss of a mated cable assembly must be below or on the differential return loss envelope.

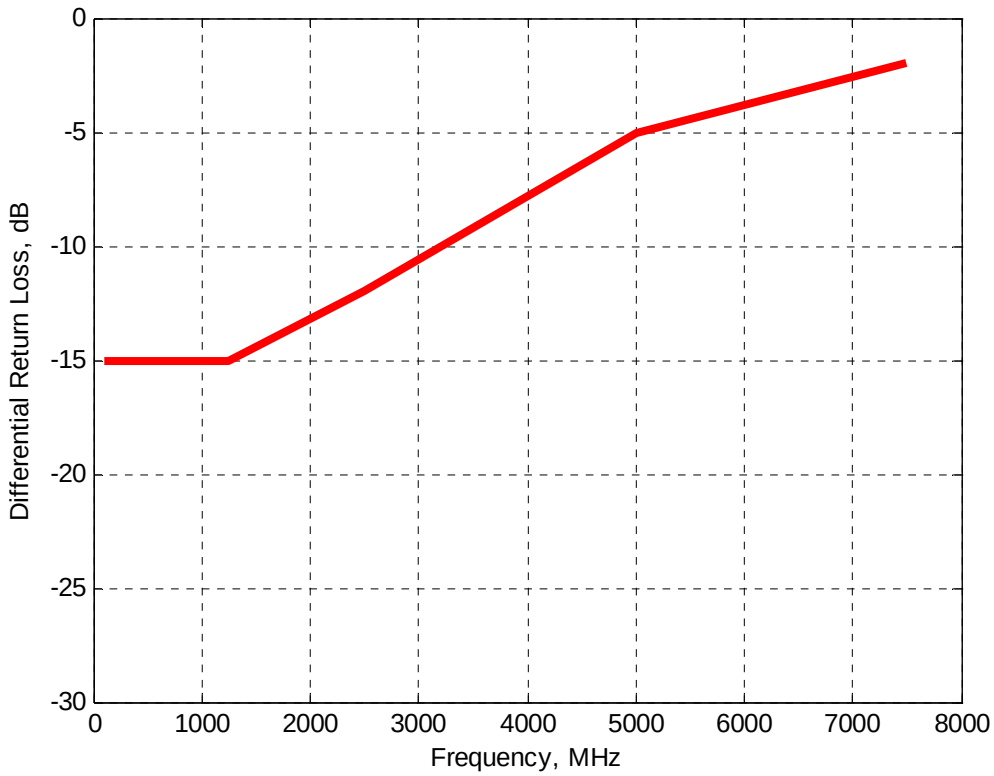


Figure 5-20. Differential Return Loss Requirement-Reference

To be consistent with the USB 3.0 channel nominal differential characteristic impedance requirement of 90 ohms, the differential return loss shall be normalized with a 90-ohm reference differential impedance. Most VNA software allows normalization of measured S-parameters to a different reference impedance. For example in PLTS, one can set the port impedance to 45 ohms to normalize the measured 50-ohm single-ended S-parameters to 45 ohms; this will result in a 90-ohm differential S-parameters after the singled-ended-to-differential conversion.

5.6.1.3.2 Differential Insertion Loss (EIA-360-101)

The differential insertion loss, SDD12, measures the differential signal energy transmitted through the mated cable assembly. Figure 5-21 shows the differential insertion loss limit, which is normalized with 90-ohm differential impedance and defined by the following vertices: (100MHz, -1.5 dB), (f0/2, -4.5 dB), (f0, -7.5 dB), and (3*f0, -25 dB). The measured differential return loss of a mated cable assembly must be above or on the differential insertion loss limit.

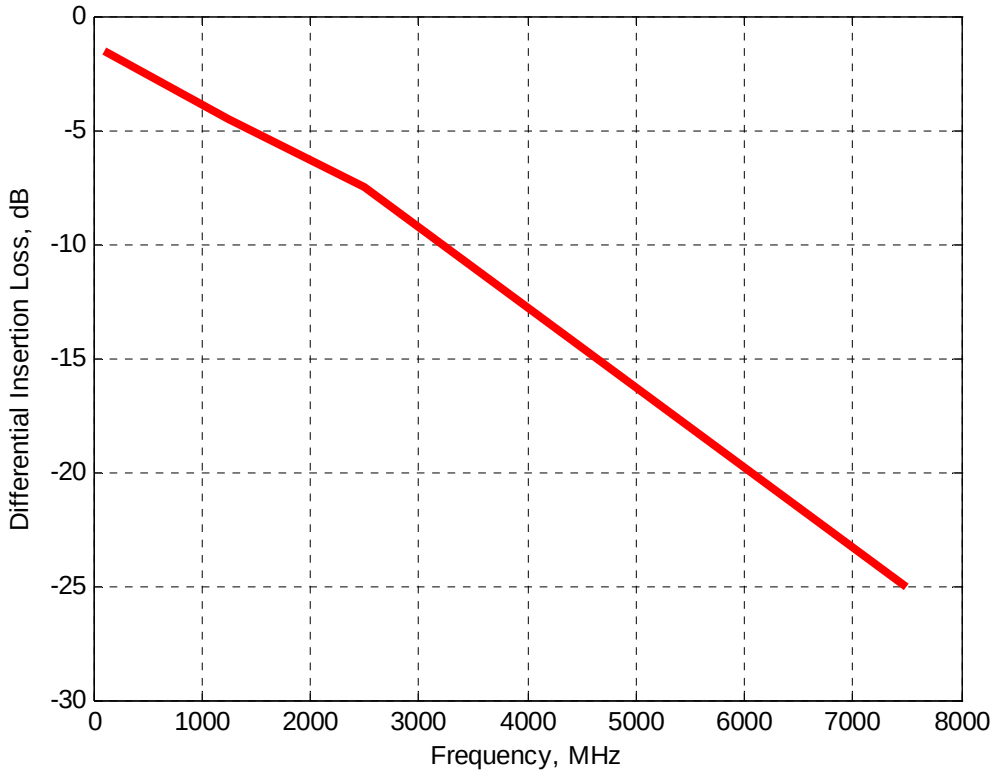


Figure 5-21. Differential Insertion Loss Requirement

5.6.1.3.3 Differential Near-End Crosstalk between USB 3.0 Pairs (EIA-360-90)

The differential crosstalk measures the unwanted coupling between differential pairs. Since the TX pair is right next to the RX pair for USB 3.0, only the differential near-end crosstalk (DDNEXT) is specified, as shown in Figure 5-22. The mated cable assembly meets the DDNEXT requirement if its DDENXT is below or on the limits shown in Figure 5-22; the vertices that defines the DDNEXT limits are: (100MHz, -32 dB), (f0, -32 dB), (2*f0,-25 dB) and (3*f0, -25 dB).

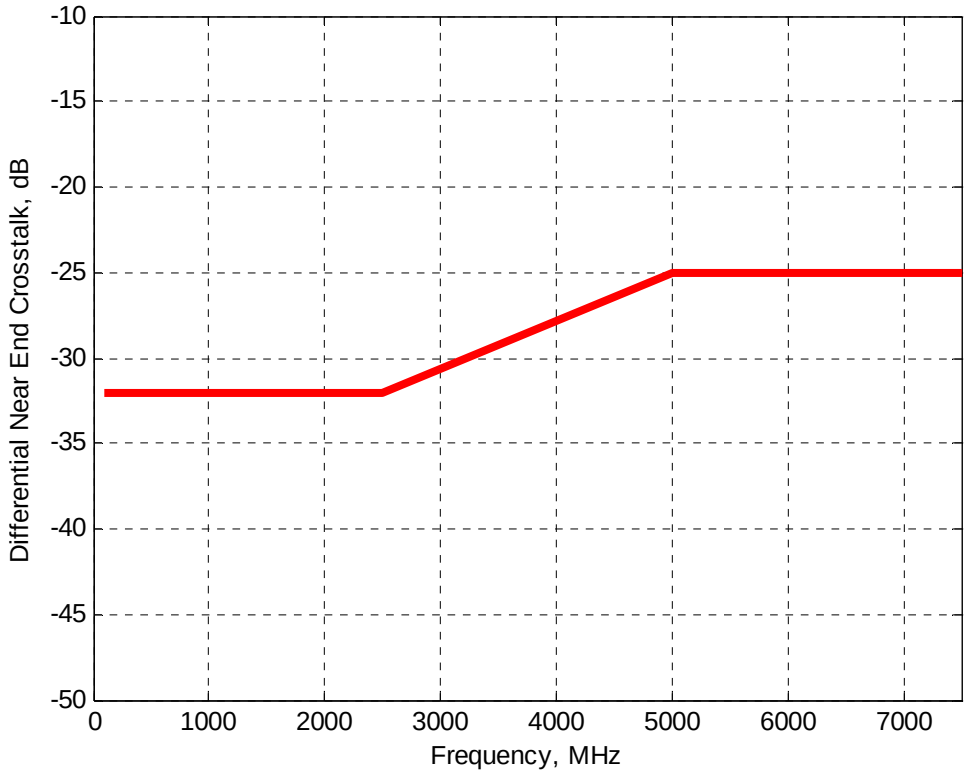


Figure 5-22. Differential Near-End Crosstalk Requirement between USB 3.0 Pairs

5.6.1.3.4 Differential Crosstalks between USB 3.0 and USB 2.0 Pairs (EIA-360-90)

The differential near-end and far-end crosstalk between the USB 2.0 pairs (D+/D-) and the USB 3.0 pairs (SSTX+/SSTX- or SSRX+/SSRX-) shall be managed to be less than the limit shown in Figure 5-23; the vertices that defines the DDNEXT limits are: (100MHz, -23 dB), (f0, -23 dB), (2*f0,-15 dB) and (3*f0, -15 dB). This requirement applies only to the USB 3.0 Standard-A connector.

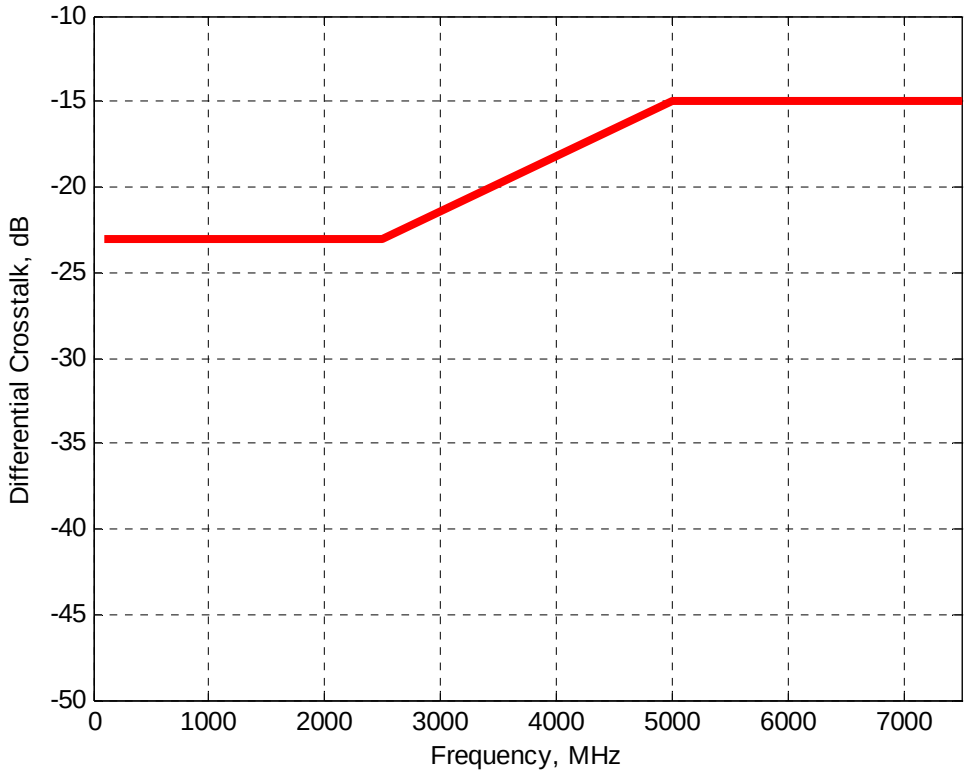


Figure 5-23. Differential Near-End and Far-End Crosstalk Requirement between USB 2.0 Pair and USB 3.0 Pairs

5.6.1.3.5 Differential to Common Mode Conversion

Since the common mode current is directly responsible for EM emission, limiting the differential-to-common-mode conversion, SCD12, will limit EM emission. Figure 5-24 illustrates the SCD12 requirement; a mated cable assembly passes the SCD12 requirement if its SCD12 is less than or equal to -20 dB across the frequency range shown in Figure 5-24.

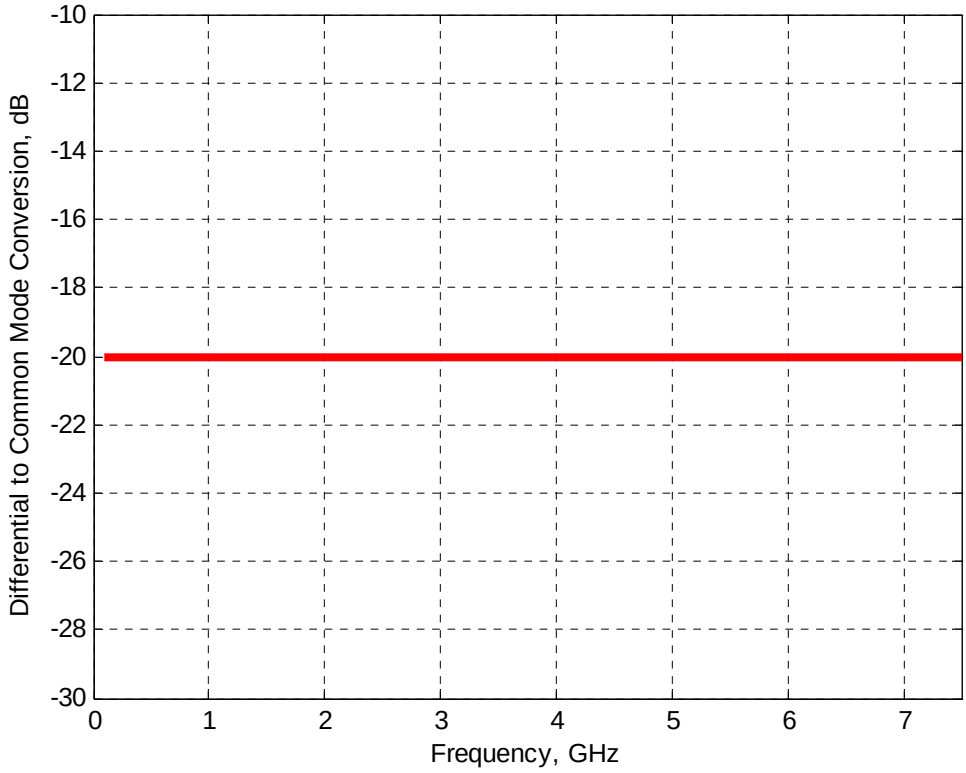


Figure 5-24. Differential to Common Mode Conversion Requirement

5.6.2 DC Electrical Requirements

5.6.2.1 Low Level Contact Resistance (EIA 364-23B)

The following requirement applies to both the signal contacts and the connector shell contacts:

- 30 mΩ (Max) initial when measured at 20 mV (Max) open circuit at 100mA. Maximum change (delta) of +10 mΩ after environmental stresses. See Section 5.7.2 for environmental requirements and test sequences.

5.6.2.2 Dielectric Strength (EIA 364-20, Method 301)

No breakdown under the following conditions:

- Apply 100 Volts AC (RMS) between adjacent contacts of unmated and mated connectors.

5.6.2.3 Insulation Resistance (EIA 364-21, Method 302)

100 mega ohms minimum between adjacent contacts of unmated and mated connectors.

5.6.2.4 Contact Current Rating (EIA 364- 70, Method 2)

1.8A for VBUS and its corresponding GND pins (Pin 1 and Pin 4 for the USB 3.0 Standard-A and Standard-B connectors; Pin 1 and Pin 5 for the USB 3.0 Micro-B connector family) and at the same time 0.25A for all the other contacts minimum when measured at an ambient temperature of 25 degrees Celsius. With power applied to the contacts, the delta temperature must not exceed +30 degrees Celsius at any point in the USB 3.0 connectors under test.

USB 3.0 Power-B only: 2.0A for devicePWR and its corresponding GND pins (Pin 10 and Pin 11 for USB 3.0 Power-B connector) and at the same time 0.25A for all the other contacts minimum when measured at an ambient temperature of 25 degrees Celsius. With power applied to the contacts, the delta temperature must not exceed +30 degrees Celsius at any point in the USB 3.0 connectors under test.

5.7 Mechanical and Environmental Requirements

5.7.1 Mechanical Requirements

5.7.1.1 Insertion Force (EIA 364-13)

35 N maximum at a maximum rate of 12.5 mm(0.492") per minute.

It is recommended to use a non-silicon based lubricant on the latching mechanism to reduce wear. If used, the lubricant may not affect any other characteristic of the system.

5.7.1.2 Extraction Force (EIA 364-13)

10 N min initial and 8 N min after the specified insertion/extraction, or durability cycles (at a maximum rate of 12.5mm (0.492") per minute).

No burs or sharp edges are allowed on top of locking latches (hook surfaces which will rub against receptacle shield).

It is recommended to use a non-silicon based lubricant on the latching mechanism to reduce wear. If used the lubricant may not affect any other characteristic of the system.

5.7.1.3 Durability or Insertion/Extraction Cycles (EIA 364-09)

Three different durability ratings are specified depending on applications:

- USB 3.0 Standard-A connector: 1500 cycles min or 5000 cycles min
- USB 3.0 Standard-B connector: 1500 cycles min or 5000 cycles min
- USB 3.0 Power-B connector: 1500 cycles min or 5000 cycles min
- USB 3.0 Micro-B connector family: 10,000 cycles min

The durability test shall be done at a max rate of 200 cycles per hour. No physical damage to any part of the connector and cable assembly. Meet requirements of additional tests as specified in the test sequences in Section 5.7.2.

5.7.1.4 Cable Flexing (EIA 364-41, Condition I)

Dimension X=3.7x cable diameter and 100 cycles in each of two planes.

No physical damage. No discontinuity over 1 ms during flexing.

5.7.1.5 Cable Pull-Out (EIA 364-38, Condition A)

Subject the cable assembly to a 50 N axial load for a min of one minute while clamping one end of the cable plug.

No physical damage.

5.7.1.6 Peel Strength (Micro-B Connector Family Only)

Minimum 150N when a soldered receptacle is pulled up from PCB in the vertical direction. No visible physical damages.

5.7.1.7 4-Axes Continuity Test (USB 3.0 Micro-B Connector Family Only)

USB 3.0 Micro-B connector family shall be tested for continuity under stress using the test configurations shown below. Plugs must be supplied in a cable assembly with a representative overmold. A USB 3.0 Micro-B or -AB receptacle shall be mounted on a 2-layer printed circuit board (PCB) of between 0.8 and 1.0 mm thickness. The PCB shall be clamped on either side of the receptacle no further than 5 mm away from the solder tails. The PCB shall initially be placed in a horizontal plane, and an 8-Newton tensile force shall be applied to the cable in a downward direction, perpendicular to the axis of insertion, for a period of at least 10 seconds.

The continuity across each contact shall be measured throughout the application of the tensile force. The PCB shall then be rotated 90 degrees such that the cable is still inserted horizontally and the 8-Newton tensile force will be applied again in the downward direction and continuity measured as before. This test will be repeated for 180 and 270 degree rotations. Passing cables will exhibit no discontinuities of greater than 1 microsecond duration in any of the four orientations.

One method for measuring the continuity through the contacts is to short all the wires at the end of the cable pigtail. Then on the PCB apply a voltage through a pull-up to each of VBus, D+, D-, ID, and the SS pins, with the GND pins connected to ground.

When testing a USB 3.0 Micro-A plug, all the sense resistors must stay pulled down for the length of the test. When testing a USB 3.0 Micro-B plug, the ID pin must stay high and the other pins must remain low for the duration of the test. Test houses may use another method to verify continuity through all pins if they choose.

The 4-axes continuity tests shall be done with a USB 3.0 Micro-B/-A plug in a USB 3.0 Micro-B/-AB receptacle AND with a USB 2.0 Micro-B/-A plug in a USB 3.0 Micro-B/-AB receptacle, as illustrated in Figure 5-25 below.

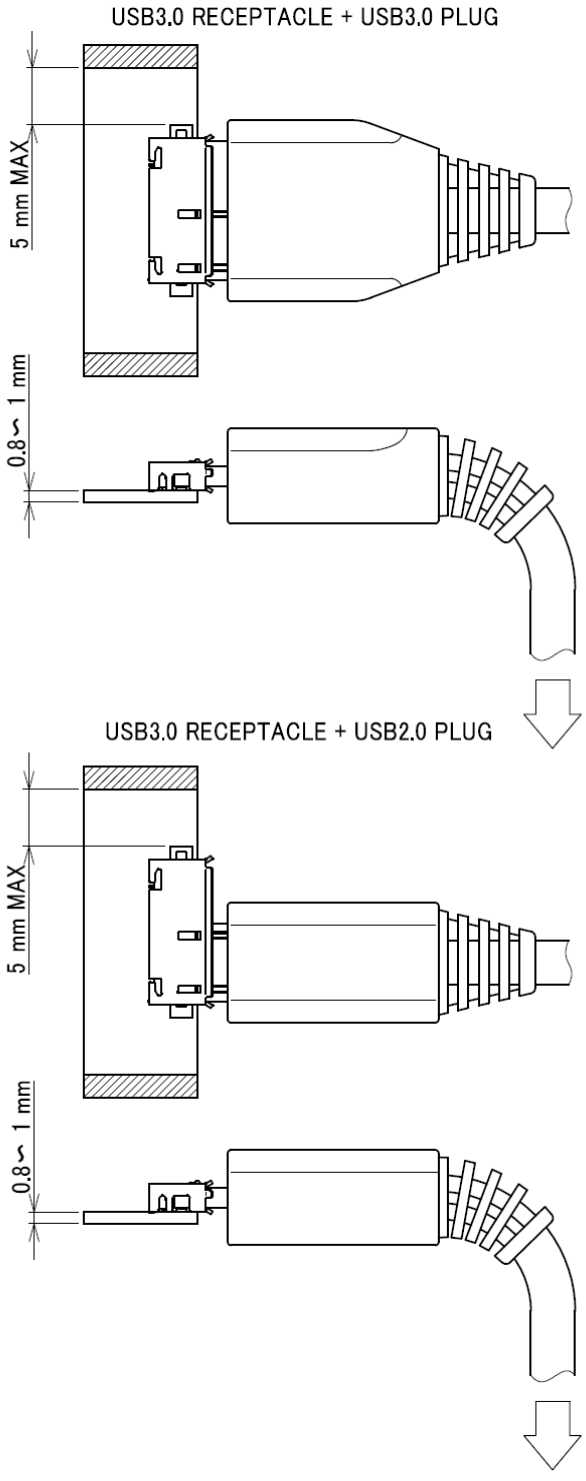


Figure 5-25. 4-Axes Continuity Test

5.7.1.8 Wrenching Strength (Reference, Micro-B Connector Family Only)

Perpendicular Force Test: This test shall be performed using virgin parts. Perpendicular forces (Fp) are applied to a plug when inserted at a distance (L) of 15mm from the edge of the receptacle. Testing conditions & method should be agreed with all parties. These forces are applied to four directions (left, right, up, down). Compliant connectors will meet the following force thresholds with the following results:

- No plug or receptacle damage: 0 - 25N
- The plug can be damaged, but in such a way that the receptacle does not sustain damage: 25 - 50N

5.7.1.9 Lead Co-Planarity

Co-planarity of all SMT leads shall be within 0.08mm range.

5.7.1.10 Solderability

Solder shall cover a minimum of 95% of the surface being immersed, when soldered at temperature 255℃ +/-5℃ for in immersion duration 5s.

5.7.1.11 RoHS Compliance

Component should to be RoHS compliant. Lead Free plug and receptacle materials should conform to Directive 2002/95/EC of January 27,2003 on Restriction of Hazardous Substances (RoHS), or other regulatory directives.

5.7.2 Environmental Requirements

The connector interface environmental tests shall follow EIA-364-1000.01, Environmental Test Methodology for Assessing the Performance of Electrical Connectors and Sockets Used in Business Office Applications. The test groups/sequences and durations shall be derived from the following requirements:

- Durability (mating/un-mating) rating of 1500, 5000, or 10,000 cycles
- Field temperature: 65 ºC
- Field life: 7 years

Since the connector defined has far more than 0.127 mm wipe length, Test Group 6 in EIA-364-1000.01 is not required. The temperature life test duration and the mixed flowing gas test duration values are derived from EIA 364-1000.01 based on the field temperature per the following.

Table 5-14. Environmental Test Conditions

Temperature Life test temperature and duration	105 °C for 120 hours
Temperature Life test temperature and duration for preconditioning	105 °C for 72 hours
Mixed flowing gas test duration	7 days

The pass/fail criterion for the low level contact resistance (LLCR) is as defined Section 5.6.2.1.

5.7.3 Materials

This specification does not specify materials for connectors and cables. Connector and cable manufactures shall select appropriate materials based on performance requirements. Table 5-15 below is provided for reference only.

Table 5-15. Reference Materials[1,2]

Component	Materials	Comments
Cable	Conductor: copper with tin plating	
	SP Shield: AL foil	
	Braid: Tin plated copper or aluminum	
	Jacket: PVC or halogen free substitute material	
Cable Overmold	Thermoset	
Connector Shell	Copper alloy or stainless steel, depending on durability requirement	
Contact	Base material: Phosphor bronze	
	Under-plating: 2.0 µm Ni	
	Contact area plating: (Min) 0.05 µm Au + (Min) 0.75 µm Ni-Pd	
	Solder tail plating: (Min) 0.05 µm Au	
Housing	Thermoplastics capable of withstanding lead-free soldering temperature.	The USB 3.0 Standard connector housings are required to be Pantone 300C (BLUE)

[1] Halogen free materials should be considered for all plastics.

[2] If an application of the part requires a flammability rating less flammable than HB a suitable flame retardant compound should be considered for plastics in this part.

5.8 Implementation Notes and Design Guides

This section discusses a few implementation notes and design guides to help users design and use the USB 3.0 connectors and cables.

5.8.1 Mated Connector Dimensions

Figure 5-26, Figure 5-27 and Figure 5-28 show the mated plugs and receptacles for the USB 3.0 Standard-A, USB 3.0 Standard-B, and USB 3.0 Micro-B connectors, respectively. The distance between the receptacle front surface to the cable overmold should be noticed by system designers to avoid interference between the system enclosure and the cable plug overmold.

Provisions must be made in connectors and chassis to ground the connector metal shells to the metal chassis to contain EMI emission.

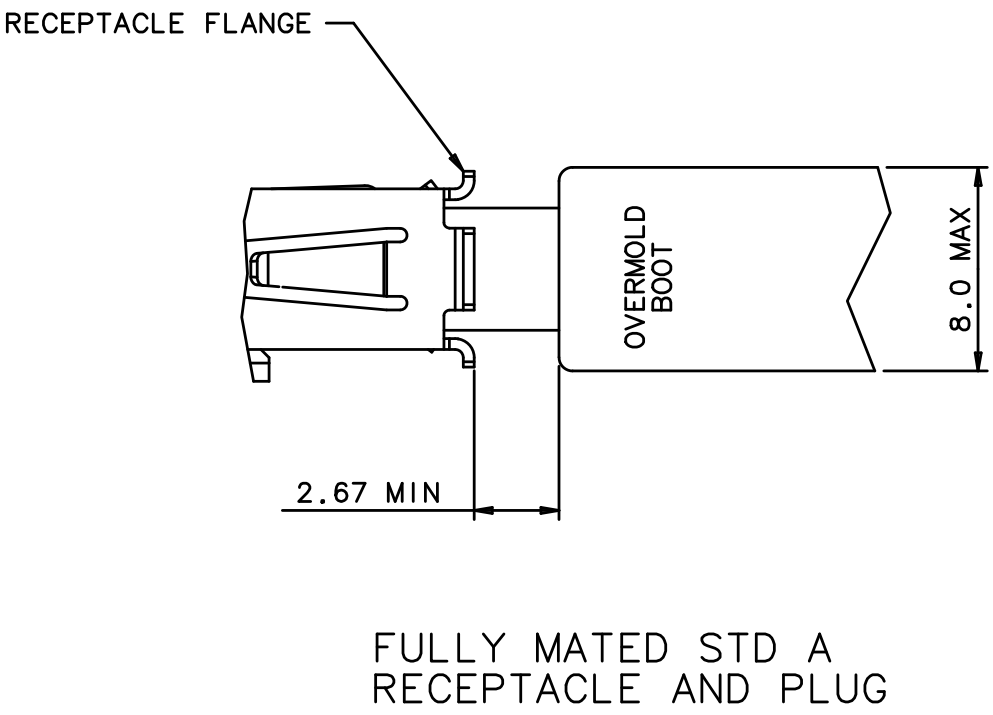


Figure 5-26. Mated USB 3.0 Standard-A Connector

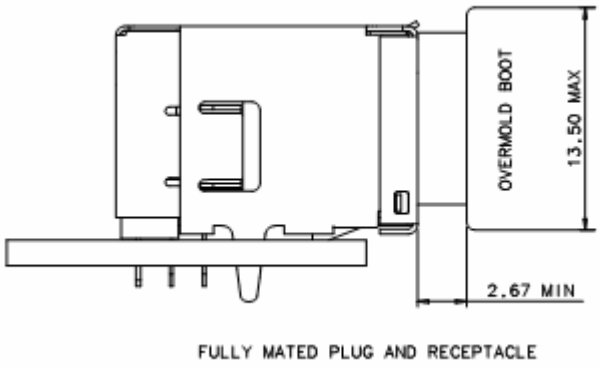


Figure 5-27. Mated USB 3.0 Standard-B Connector

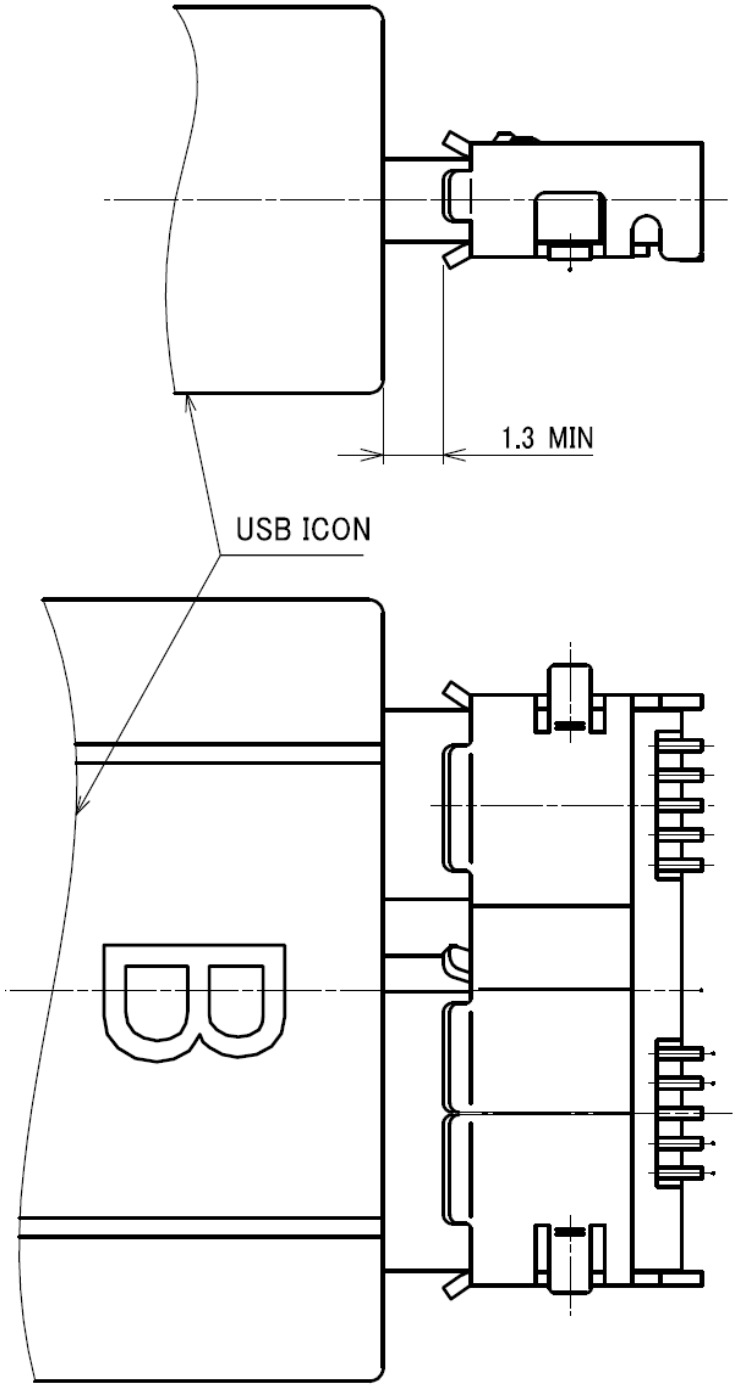


Figure 5-28. Mated USB 3.0 Micro-B Connector

5.8.2 EMI Management

Systems that include the USB 3.0 connectors and cable assemblies must meet the relevant EMI/EMC regulations. Because of the complex nature of EMI, it is difficult to specify a component level EMI test for the cable assemblies. However, connector and cable assembly designers, as well as system implementers must pay a great attention to receptacle and cable plug shielding to ensure a low impedance grounding path. The followings are a few guidelines for EMI management:

- The quality of raw cables must be ensured. The intra-pair skew or the differential to common mode conversion of the SS pairs has a significant impact on cable EMI performance and it must be controlled within the limit of this specification.
- The cable external braid should be terminated to the cable plug metal shell as close to 360˚ as possible. Without appropriate shielding termination, even a perfect cable with zero intra-pair skew can break EMI.
- If not done properly, the wire termination can contribute a lot to differential-to-common-mode conversion. The breakout distance for the wire termination should be kept as small as possible for both EMI and signal integrity. If possible, always maintain symmetry for the two lines within a differential pair.
- The mating interface between the receptacle and cable plug should have a sufficient number of grounding fingers, or springs to provide a continuous return path from the cable plug to system ground. The presence of the friction lock features should not compromise ground return connections.
- It is recommended that the receptacle connectors be designed with a back-shield, as a part of the receptacle connector metal shell. The back-shield should be designed with a short return path to the system ground plane.
- System implementers should ensure that the receptacle connectors be connected to metal chassis or enclosures through grounding fingers, screws, or any other way to mitigate EMI.

5.8.3 Stacked Connectors

Stacked USB connectors are commonly used in PC systems. Though this specification does not explicitly define the stacked USB 3.0 Standard-A receptacles, they are certainly supported. The following are a few points that should be noticed when designing a stacked USB 3.0 connector:

- A stacked connector introduces additional crosstalks between the stacks. Such crosstalks should be minimized when designing a stacked USB 3.0 connector. The differential NEXT and FEXT should be managed within -32 dB (up to the fundamental frequency 2.5 GHz) between differential pairs across stacks.
- Due to the additional electrical length, the connector on the top stack will generally perform worse than the one on the bottom. Connector designers should carefully design the top contact geometries and materials to minimize impedance discontinuity. Regardless how many stacks one may choose to design, the same electrical requirements defined in Section 5.6 must be met.

5.8.4 Captive Cables

The captive cables must still meet the mated connector requirements specified in Section 5.6.1.2. But a captive cable is not considered a stand-alone component. For electrical budgeting purposes, a captive cable is considered part of a device. The electrical PHY chapter defines the electrical budgets for devices attached with captive cables.

6 Physical Layer

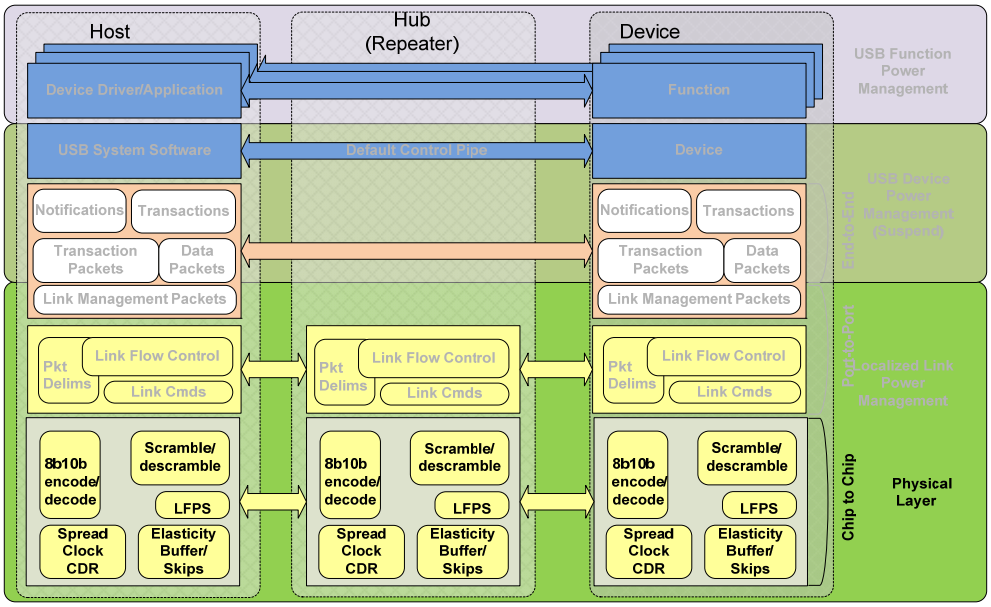


Figure 6-1. Super Speed Block Diagram: Physical

6.1 Physical Layer Overview

The physical layer defines the signaling technology for the SuperSpeed bus. This chapter defines the electrical requirements of the Super Speed physical layer.

This section defines the electrical-layer parameters required for interoperability between Super Speed components. Normative specifications are required. Informative specifications may assist product designers and testers understand the intended behavior of the Superspeed bus.

6.2 Physical Layer Functions

The functions of the physical layer are shown in the following figures.

Phy Layer Tx Block Diagram

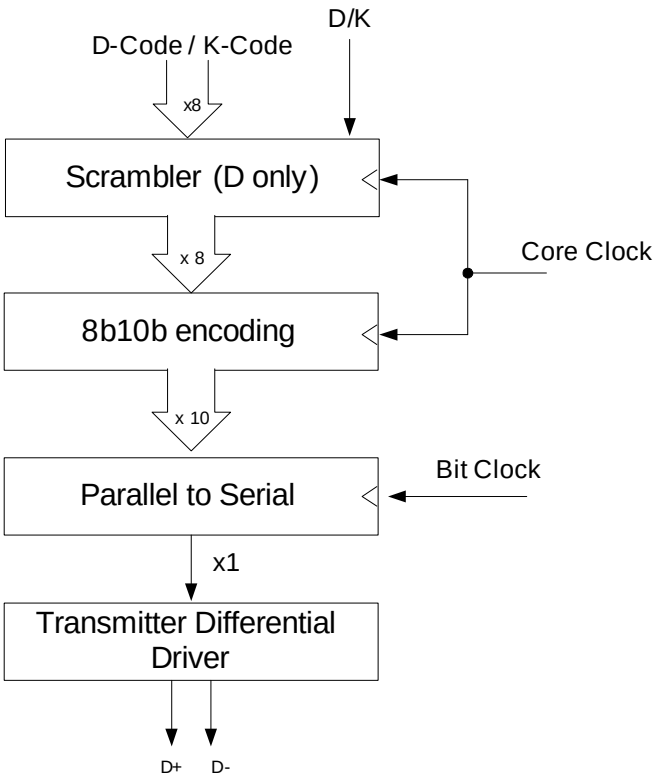


Figure 6-2. Transmitter Block Diagram

Phy Layer Rx Block Diagram

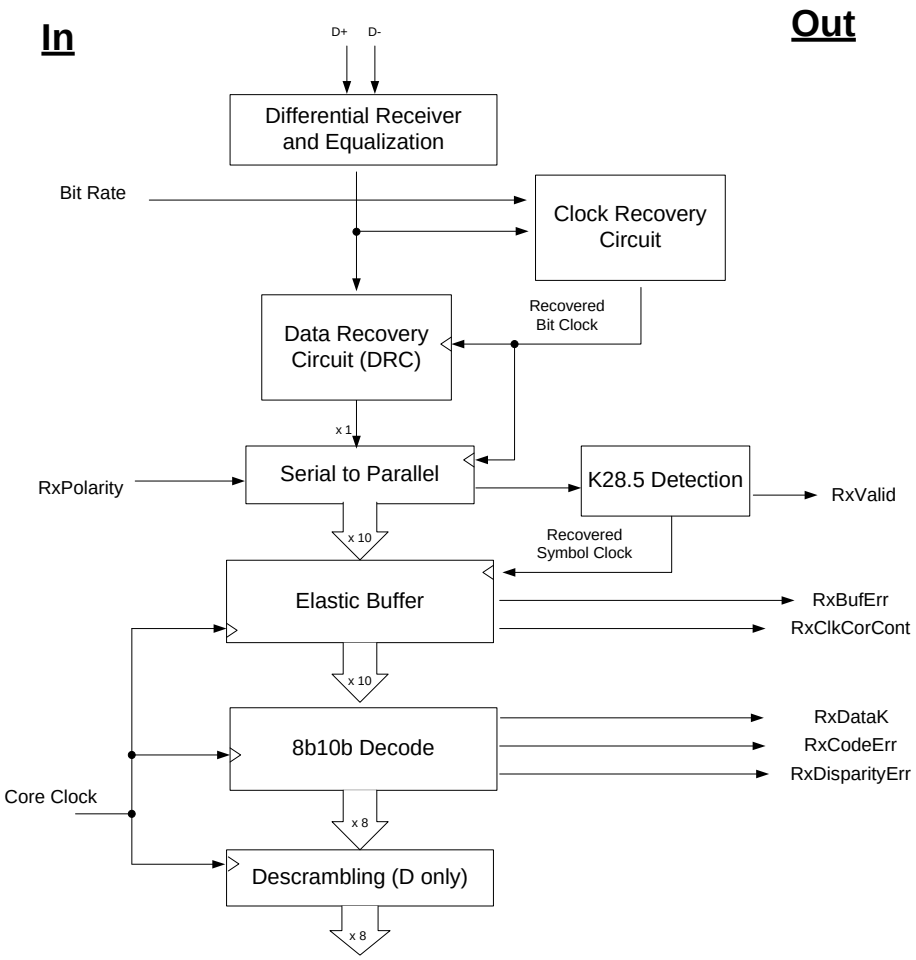


Figure 6-3. Receiver Block Diagram

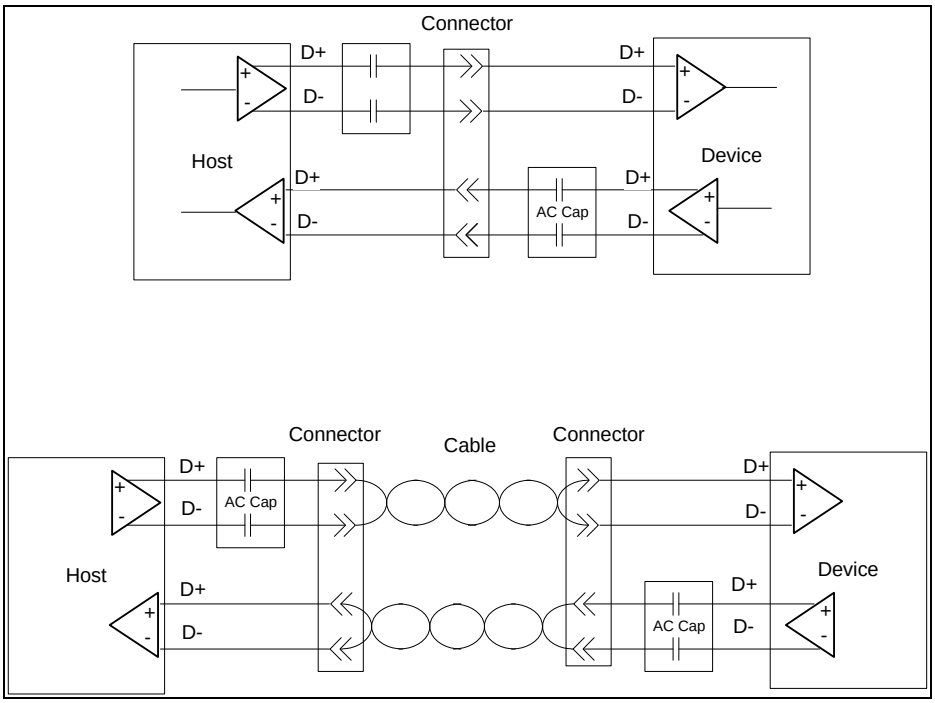


Figure 6-4. Channel Models without Cable (Top) and with Cable (Bottom)

6.2.1 Measurement Overview

The normative SuperSpeed eye diagram is to be measured through a compliance channel that represents the sum of a long channel, a short channel and a 3 meter cable. These reference channels are described in Section TBD. The eye diagram is measured using the clock recovery function described in Section 6.5.2.

For the long channel case the eye diagram at the receiver is completely closed. An informative receiver equalization function is provided in Section 6.8.2 that is optimized to the compliance channel and used to open the receiver eye.

The details of this method will be described in a future compliance document. This specification methodology allows a silicon vendor to design the channel and the component as a matched pair. It is expected that a silicon component will have layout guidelines that must be followed in order for the component to meet the overall specification and the eye diagram at the end of the compliance channel.

6.2.2 Channel Overview

A transmitter and receiver that operate together and are located on the same component are called a port. A channel connects two Ports together with two uni-directional differential pairs of pins for a total of 4 wires. The ports are required to be AC coupled. The AC coupling capacitors are associated with the transmitter.

6.3 Symbol Encoding

SuperSpeed uses the 8b/10b transmission code. The definition of this transmission code is identical to that specified in ANSI X3.230-1994 (also referred to as ANSI INCITS 230-1994), clause 11. As shown in Figure 6-5, ABCDE maps to abcdei and FGH maps to fghj.

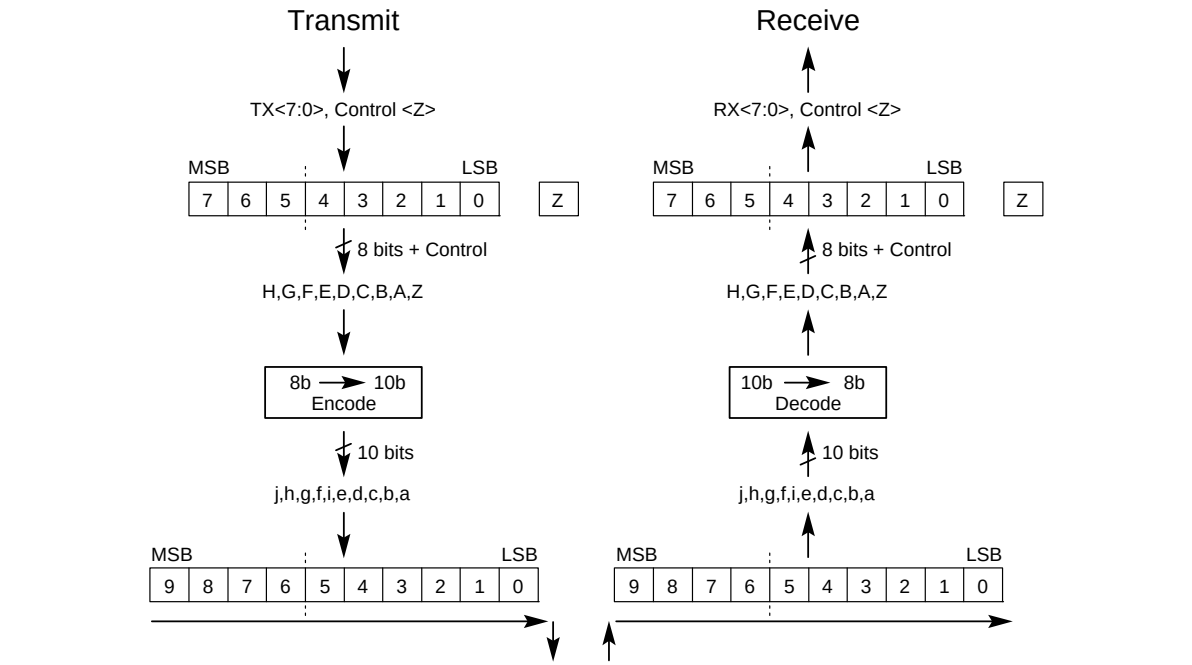


Figure 6-5. Character to Symbol Mapping

6.3.1 Serialization and De-serialization of Data

The bits of a Symbol are placed starting with bit "a" and ending with bit "j." This is shown in Figure 6-6.

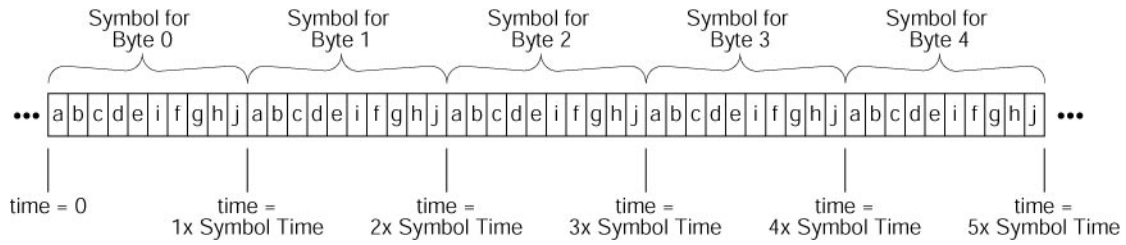


Figure 6-6. Bit Transmission Order

6.3.2 Normative 8b/10b Decode Rules

1. A Transmitter is permitted to pick any disparity when first transmitting differential data after being in an Electrical Idle state. The Transmitter shall then follow proper 8b/10b encoding rules until the next Electrical Idle state is entered.
2. The initial disparity for a Receiver is the disparity of the first Symbol used to obtain Symbol lock.
3. Disparity may also be-reinitialized if Symbol lock is lost and regained during the transmission of differential information due to an implementation specific number of errors.
4. All following received Symbols after the initial disparity is set shall be in the proper column corresponding to the current running disparity.
5. Receive disparity errors do not directly cause the link to retrain.

6.3.3 Data Scrambling

The scrambling function is implemented using a free running Linear Feedback Shift Register (LFSR.) On the Transmit side, scrambling is applied to characters prior to the 8b/10b encoding. On the Receive side, de-scrambling is applied to characters after 8b/10b decoding. The LFSR is reset whenever a COM symbol is sent or received.

The LFSR is graphically represented in Figure 6-7. Scrambling or unscrambling is performed by serially XORing the 8-bit (D0-D7) character with the 16-bit (D0-D15) output of the LFSR. An output of the LFSR, D15, is XORed with D0 of the data to be processed. The LFSR and data register are then serially advanced and the output processing is repeated for D1 through D7. The LFSR is advanced after the data is XORed.

The mechanism to notify the Physical Layer to disable scrambling is implementation specific and beyond the scope of this specification.

The data scrambling rules are as follows:

1. The LFSR implements the polynomial: $G(X)=X^{16}+X^{5}+X^{4}+X^{3}+1$
2. The LFSR value shall be advanced eight serial shifts for each Symbol except for SKP.
3. All 8b/10b D-codes, except those within the Training Sequence Ordered Sets shall be scrambled.
4. K codes shall not be scrambled.
5. The initialized value of an LFSR seed (D0-D15) shall be FFFFh. After COM leaves the Transmitter LFSR, the LFSR on the transmit side shall be initialized. Every time COM enters the Receive LFSR, the LFSR on the Receive side shall be initialized.

IMPLEMENTATION NOTE

Disabling Scrambling

Disabling scrambling is intended to help simplify test and debug equipment. Control of the exact data patterns is useful in a test and debug environment. Since scrambling is reset at the Physical Layer there is no reasonable way to reliably control the state of the data transitions through software. The Disable Scrambling bit is provided in the training sequence for this purpose.

The mechanism(s) and/or interface(s) used to notify the Physical Layer to disable scrambling is component implementation specific and beyond the scope of this specification.

For more information on scrambling, see Appendix B.

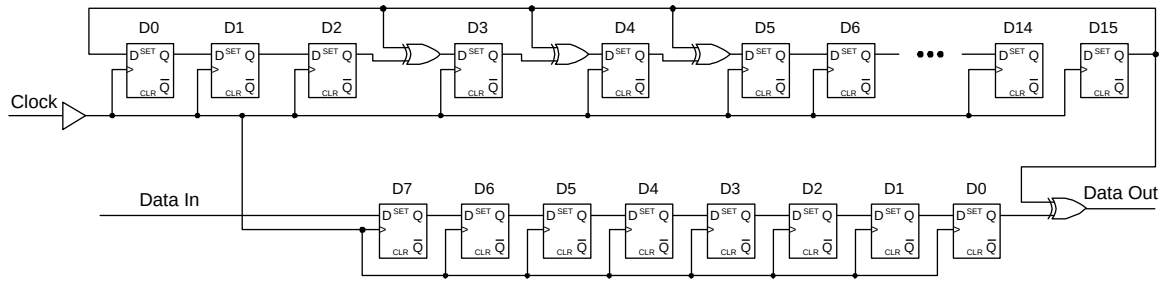


Figure 6-7. LFSR with Scrambling Polynomial

6.3.4 8b/10b Decode Errors

An 8b/10b error shall occur when a received Symbol does not match any of the valid 8b/10b Symbols listed in Appendix B. 8b/10b errors may not directly initiate Recovery.

6.3.5 Special Symbols for Framing and Link Management

The 8b/10b encoding scheme provides Special Symbols that are distinct from the Data Symbols used to represent Characters. These Special Symbols are used for various Link Management mechanisms described later. Table 6-1 lists the Special Symbols used and provides a brief description for each. Special Symbols must follow the proper 8b10b disparity rules.

Table 6-1. Special Symbols

Encoding	Symbol	Name	Description
K28.1	SKP	Skip	Compensates for different bit rates between two communicating ports. SKPs may be dynamically inserted or removed from the data stream.
K28.2	SDP	Start Data Packet	Marks the start of a Data Packet Payload
K28.3	EDB	End Bad	Marks the end of a nullified Packet
K28.5	COM	Comma	Used for symbol alignment.
K28.6	-----	-----	Reserved
K27.7	SHP	Start Header Packet	Marks the start of a Data Packet, Transaction Packet or Link Management Packet
K29.7	END	End	Marks the end of a packet
K30.7	SLC	Start Link Command	Marks the start of a Link Command
K23.7	EPF	End Packet Framing	Marks the end of a packet framing

6.4 Link Initialization and Training

This section defines the sequences that are used for configuration and initialization. The sequences are used by the Initialization State Machine (see Chapter 7) for the following functions:

- Configuring and initializing the link
- Bit-lock and symbol lock
- Rx equalization training
- Lane polarity inversion

6.4.1 Normative Training Sequence Rules

Training sequences are composed of Ordered Sets used for initializing bit alignment, symbol alignment and receiver equalization.

The following rules apply to the training sequences:

- Training sequence Ordered Sets shall be 8b/10b encoded.
- Transmission of a TS1 or TS2 Ordered Sets shall not be interrupted by SKP Ordered Sets.
  SKP Ordered Sets shall be inserted before, or after, completion of any TS1, or TS2 Ordered set.
- No SKP Ordered sets are to be transmitted during the entire TSEQ time.

Additional rules for the use of TSEQ, TS1 and TS2 Ordered Sets can be found in Chapter 7.

6.4.1.1 Training Control Bits

The training control bits are found in the Link Functionality symbol within the TS1 and TS2 ordered sets. They are described in Table 6-5.

Only one bit in the Link Functionality symbol may be set by the transmitting component at a time. If a receiver detects more than one bit is set in the received Link Functionality symbol then all of the training control bits must be ignored. This includes reserved bits.

The training control bits are not error checked. In order for a training control bit to be valid at least 3 control consecutive bits must be received.

6.4.1.2 Training Sequence Values

Table 6-2. TSEQ Ordered Set

Symbol Number	Name	Value
0	K28.5	COM (Comma)
1	D31.7	0xFF
2	D23.0	0x17
3	D0.6	0xC0
4	D20.0	0x14
5	D18.5	0xB2
6	D7.7	0xE7
7	D2.0	0x02
8	D2.4	0x82
9	D18.3	0x72
10	D14.3	0x6E
11	D8.1	0x28
12	D6.5	0xA6
13	D30.5	0xBE
14	D13.3	0x6D
15	D31.5	0xBF
16-31	D10.2	0x5A

Table 6-3. TS1 Ordered Set

Symbol Number	Encoded Values	Description
0-3	K28.5	COM (Comma)
4	D0.0	Reserved for future use
5	See Table 6-5	Link Functionality
6-15	D10.2	TS1 Identifier

Table 6-4. TS2 Ordered Set

Symbol Number	Encoded Values	Description
0-3	K28.5	COM (Comma)
4	D0.0	Reserved
5	See Table 6-5	Link Functionality
6-15	D5.2	TS2 Identifier

Table 6-5. Link Functionality Symbols

	TS1 Symbol 5	Description
Bit 0	0 = Normal Training 1 = Reset	Reset is set by the Host only in order to reset the device
Bit 1	0 = SS enable 1 = SS disable	Device sets to 0 Host sets to 1 to disable Device.
Bit 2	0 = Loopback de-asserted 1 = Loopback asserted	When set, the receiving component enters digital loopback
Bit 3	0 = Disable Scrambling de-asserted 1 = Disable Scrambling asserted	When set, the receiving component disables scrambling
Bit 4:7	Set to 0	Reserved for future use.

6.4.2 Lane Polarity Inversion

During the TSEQ training sequence, the Receiver must use the D10.2 Symbol within the TSEQ Ordered Set to determine lane polarity inversion (D+ and D- are swapped). If polarity inversion has occurred, the D10.2 symbols within the TSEQ ordered set will be received as D21.5 instead of D10.2 and the receiver must invert the polarity of the received bits. This must be done before the TSEQ symbols 1-15 are used since these symbols are not all symmetric under inversion in the 8b10b domain. If the receiver does not use the TSEQ training sequence then the polarity inversion may be checked against the D10.2 symbol in the TS1 ordered set.

6.4.3 Elasticity Buffer and SKP Ordered Set

The SuperSpeed architecture supports a separate clock domain on each side of the Link. The accuracy each clock is required to be within +-300 ppm. This gives a maximum frequency difference between the two devices of the link of +- 600 ppm. In addition, SSC creates a frequency delta that has a maximum difference of 5000 ppm. The total magnitude of the frequency delta can range from -5300 to 300 ppm. This frequency delta is managed by an elasticity buffer that consumes or inserts SKP ordered sets.

SKP Ordered Sets shall be used to compensate for frequency differences between the two ends of the Link. The transmitter must send one SKP ordered set every 354 symbols. However, SKP ordered sets must only be inserted at packet boundaries. The transmitter is allowed to buffer the SKP ordered sets up to a maximum of 4 SKP ordered sets. The receiver must implement an elasticity buffer capable of buffering (or starving) 8 symbols of data.

Table 6-6. SKP Ordered Set Structure

Symbol Number	Encoded Values	Description
0	K28.1	SKP
1	K28.1	SKP

6.4.4 Compliance Pattern

Entry to the Polling.Compliance substate is described in Chapter 7. This initiates the transmission of the PRBS compliance sequence. SKPs are not sent during the compliance pattern. The compliance pattern shall be transmitted continuously or until a ping LFPS is detected at the receiver. Upon detection of a ping LFPS, the compliance pattern shall advance to the next compliance pattern. Upon detection of a reset LFPS the compliance pattern shall be terminated. The compliance pattern sequences are given in Table 6-7.

Table 6-7. Compliance Pattern Sequences

Compliance Pattern	Value	Description
CP0	D0.0 scrambled	A psuedo-random data pattern that is exactly the same as logical idle (see Chapter 7) but does not include SKP sequences
CP1	D10.2	Nyquist frequency
CP2	D24.3	Nyquist / 2
CP3	K28.5	COM pattern
CP4	LFPS	The low frequency periodic signaling pattern
CP5	K28.7	With de-emphasis
CP6	K28.7	Without de-emphasis
CP7	50-250 1’s and 0’s	With de-emphasis. Repeating 50-250 1’s and then 50-250 0’s.
CP8	50-250 1’s and 0’s	With without de-emphasis. Repeating 50-250 1’s and then 50-250 0’s.

Note: Unless otherwise noted scrambling is disabled for compliance patterns.

6.5 Clock and Jitter

6.5.1 Informative Jitter Budgeting

The jitter for USB 3.0 is budgeted among the components that comprise the end to end connections: the transmitter, channel (including packaging, connectors, and cables), and the receiver. The jitter budget is derived at the pads of the device. The Dj distribution is the dual Dirac method. Table 6-8 lists Tx, Rx, and channel jitter budgets.

Table 6-8. Informative Jitter Budgeting at the Device

Jitter Contribution (ps)	Rj[1,2]	Dj[3]	Tj[4] at 10^{-12}
Tx[6]	2.42	41	75
Media[5]	2.13	45	75
Rx	2.42	57	91
Total:	4.03	143	200

Notes:

1. Rj is the sigma value assuming a Gaussian distribution
2. Rj Total is computed as the Root Sum Square of the individual Rj components
3. Dj budget is using the Dual Dirac method
4. Tj at a 10^{-12} BER is calculated as 14.068 * Rj + Dj
5. The media budget includes the cancellation of ISI from the appropriate Rx equalization function
6. Tx is measured after JTF

6.5.2 Normative Clock Recovery Function

The Tx Phase jitter measurement is performed using a standard clock recovery, shown in Figure 6-8. For information on the golden PLL measurement see www.t11.org.

The clock recovery function is given by Equations 1-3. A schematic of the general clock recovery function is shown in Figure 6-8. As shown, the clock recovery circuit has a low pass response. After the recovered clock is compared (subtracted) to the data the overall clock recovery becomes a high pass function. This is shown in Figure 6-9.

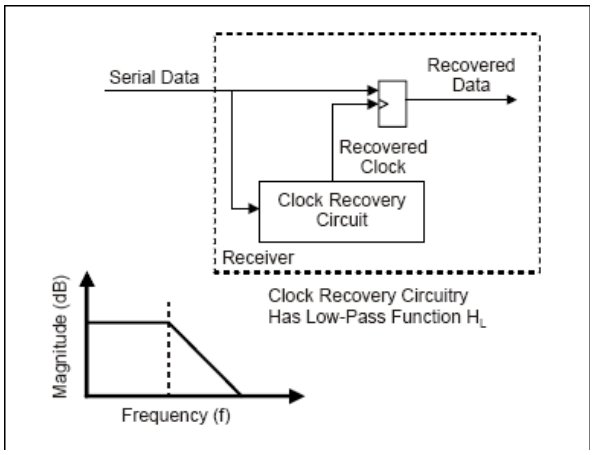


Figure 6-8. Jitter Filtering – "Golden PLL" and Jitter Transfer Functions

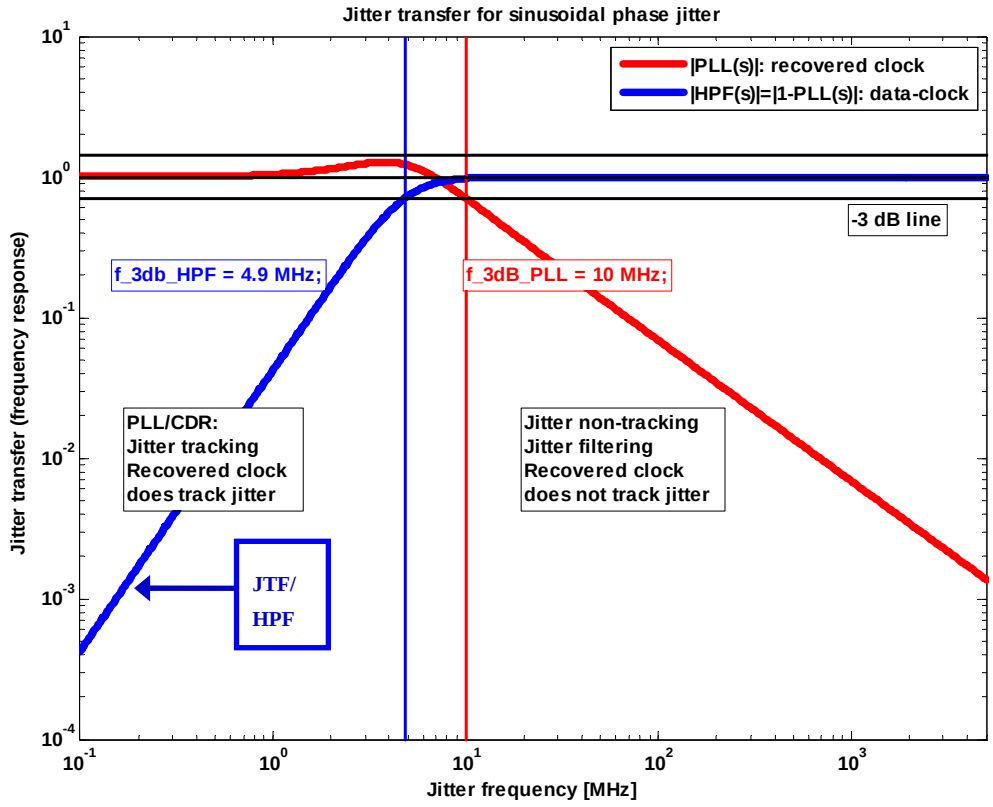


Figure 6-9. "Golden PLL" and Jitter Transfer Functions

The equations for these functions are

(1) $$H_{CDR}(s) = \frac{2s\zeta\omega_n + \omega_n^2}{s^2 + 2s\zeta\omega_n + \omega_n^2}$$

and

(2) $$JTF(s) = \frac{s^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

where ω_n is the natural frequency and ξ is the damping factor. The relationship to the 3 dB frequency is

(3) $$\omega_{3dB} = \omega_n\left(1 + 2\zeta^2 + \left[\left(1 + 2\zeta^2\right)^2 + 1\right]^{\frac{1}{2}}\right)^{\frac{1}{2}}$$

As shown in figure 5-7, the corner frequency $\omega_{3dB} = 2\pi 10^7$ and $\zeta = 0.707$. This transfer function has a maximum peaking of 2 dB.

6.5.3 Normative Spread Spectrum Clocking (SSC)

All ports are required to have Spread Spectrum Clocking (SSC) modulation. Providing the same SSC clock to two different components is allowed but not required, the SSC can be generated asynchronously. The SSC profile is not specified and is vendor specific. The SSC modulation requirement is given in table 6-10. The SSC modulation may not violate the phase slew rate described in Section 6.5.4.

Table 6-9. SSC Parameter

Symbol	Description	Limits		Units	Note
		Min	Max		
$T_{SSC\text{-}MOD\text{-}RATE}$	Modulation Rate	30	33	kHz	
$T_{SSC\text{-}FREQ\text{-}DEVIATION}$	SSC deviation	+0.0 / -0.4	+0.0/-0.5	%	1

Note:

1. The data rate is modulated from +0% to -0.5% of the nominal data rate frequency and scales with data rate.
2. This is measured below 1 MHz only

An example of the period modulation from triangular SSC is shown in Figure 6-10 .

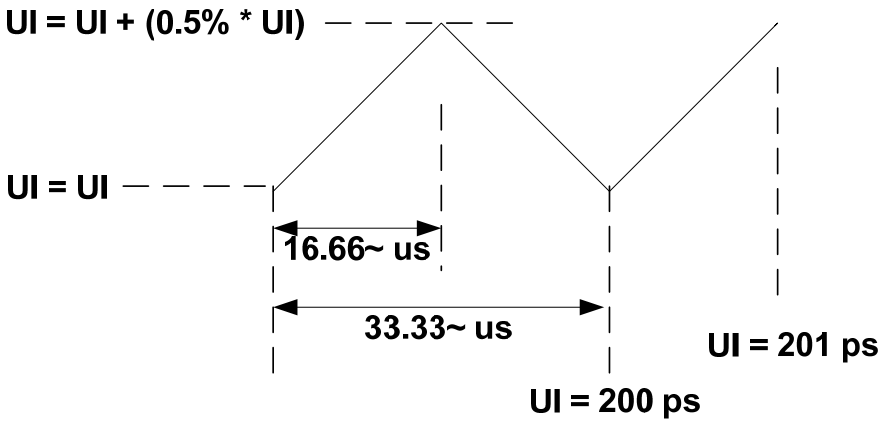


Figure 6-10. Period Modulation from Triangular SSC

6.5.4 Normative Slew Rate Limit

The CDR is a slew rate limited phase tracking device. The combination of SSC and all other jitter sources within the bandwidth of the CDR must not exceed the maximum allowed slew rate.

This measurement is performed by filtering the phase jitter with the CDR transfer function and taking the first difference of the phase jitter to obtain the filtered period jitter. The peak of the period jitter must not exceed $T_{CDR_SLEW_MAX}$ given in Table 6-10.

Additional details on the slew rate measurement will be available in a future white paper.

6.6 Signaling

6.6.1 Eye Diagrams

The eye diagrams are a graphical representation of the voltage and time limits of the signal. This eye mask applies to jitter after the application of the appropriate jitter transfer function and reference receiver equalization. In all cases, the eye is to be measured for 10^6 UI. The budget for the link is derived assuming a total 10^{-12} bit error rate and is extrapolated to a measurement of 10^6 UI assuming the random jitter is Gaussian.

Figure 6-11 shows the eye mask used for all eye diagram measurements. Referring to the figure, the time t, is measured from the crossing points of D+ and D-. The time is called the eye width, and the voltage is the eye height. The eye height is to be measured at the maximum opening (at the center of the eye width +/- 0.05 UI).

The eye diagrams are to be centered using the jitter transfer function (JTF). The recovered clock is obtained from the data and processed by the JTF. The center of the recovered clock is used to position the center of the data in the eye diagram.

The eye diagrams are to be measured into a 50 ohm load.

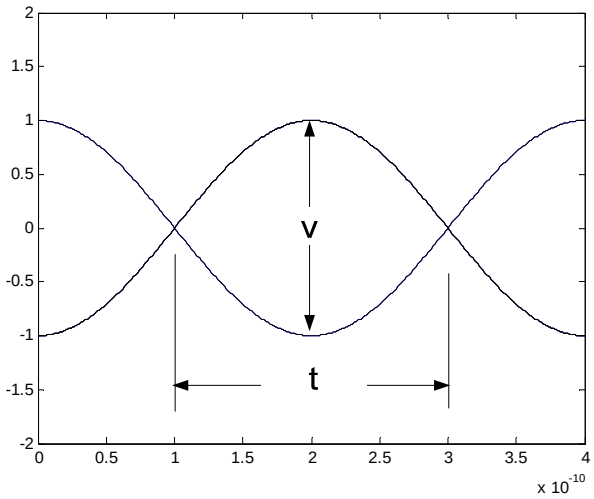


Figure 6-11. Generic Eye Mask

6.6.2 Voltage Level Definitions

Referring to Figure 6-12, the differential voltage, V_{DIFF}, is the voltage on D+ with respect to D-. V_{DIFF} is the same voltage as the swing on the single signal of one conductor. The differential voltage is

(4) $V_{DIFF} = D+ - D-$

The total differential voltage swing is the peak to peak differential voltage, $V_{DIFF-PP.}$ This is twice the differential voltage. The peak to peak differential voltage is

(6) $V_{DIFF-PP}=2 * V_{DIFF}$

The Common Mode Voltage (V_{CM}) is the average voltage present on the same differential pair with respect to ground. This is measured, with respect to ground, as

(7) $V_{CM}=(D+ + D-) / 2.$

DC is defined as all frequency components below F_{DC} = 30 kHz. AC is defined as all frequency components at or above F_{DC} = 30 kHz. These definitions pertain to all voltage and current specifications.

An example waveform is shown in Figure 6-12. In this waveform the peak to peak differential voltage, $V_{DIFF-PP}$ is 800 mV. The differential voltage, V_{DIFF}, is 400 mVPP. Note that while the center crossing point for both D+ and D- is shown at 300 mV, the corresponding crossover point for the differential voltage is at 0.0 V. The center crossing point at 300 mV is also the common mode voltage, V_{CM}.

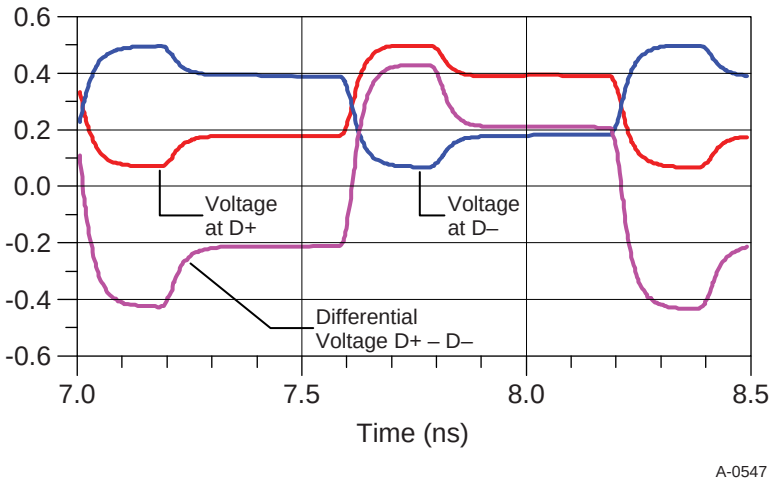


Figure 6-12. Single-ended and Differential Levels

6.6.3 Tx and Rx Input Parasitics

Tx and Rx input parasitics are specified by the lumped circuit shown in Figure 6-13.

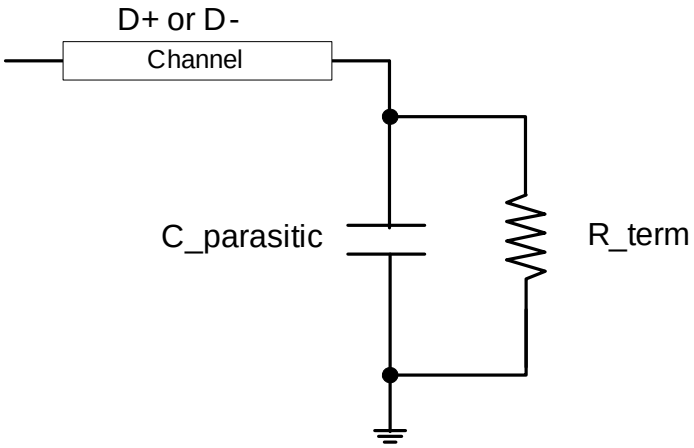


Figure 6-13. Device Termination Schematic

In this circuit, the input buffer is simplified to a termination resistance in parallel with a parasitic capacitor. This simplified circuit is the load impedance.

6.7 Transmitter Specifications

6.7.1 Transmitter Electrical Parameters

Peak (P) and peak- peak (PP) are defined in Section 6.6.2.

Table 6-10. Transmitter Normative Electrical Parameters

Symbol	Parameter	5.0 GT/s	Units	Comments
UI	Unit Interval	199.94 (min) 200.06 (max)	ps	The specified UI is equivalent to a tolerance of ±300 ppm for each device. Period does not account for SSC induced variations.
$V_{TX-DIFF-PP}$	Differential p-p Tx voltage swing	0.8 (min) 1.2 (max)	V	Nominal is 1 V p-p
$V_{TX-DE-RATIO}$	Tx de-emphasis	3.0 (min) 4.0 (max)	dB	Nominal is 3.5 dB
$R_{TX-DIFF-DC}$	DC differential impedance	72 (min) 120 (max)	Ω	
$V_{TX-RCV-DETECT}$	The amount of voltage change allowed during Receiver Detection	0.6 (max)	V	Detect voltage transition should be an increase in voltage on the pin looking at the detect signal to avoid a high impedance requirement when an “off” receiver’s input goes below ground. See Section 1.2.5.6 and Note 9 for details
$C_{AC-COUPLING}$	AC Coupling Capacitor	75 (min) 200 (max)	nF	All Transmitters shall be AC coupled. The AC coupling is required either within the media or within the transmitting component itself.
$T_{CDR_SLEW_MAX}$	Max slew rate	10	ms/sec	See the jitter white paper for details on this measurement.

The values in the following table are informative and not normative. They are included in this document to provide some guidance beyond the normative requirements in Table 6-10 for transmitter design and development. A transmitter can be fully compliant with the normative requirements of the spec and not meet all the values in this table (many of which are immeasurable in a finished product). Similarly, a transmitter that meets all the values in this table is not guaranteed to be in full compliance with the normative part of this spec.

Table 6-11. Transmitter Informative Electrical Parameters at Silicon Pads

Symbol	Parameter	5.0 GT/s	Units	Normative	Comments
$T_{MIN-PULSE-Dj}$	Deterministic min pulse	0.96	UI		Tx pulse width variation that is deterministic.
$T_{MIN-PULSE-Tj}$	Tx min pulse	0.90	UI		Min Tx pulse at 10^{-12} including Dj and Rj.
T_{TX-EYE}	Transmitter Eye	0.625 (min)	UI		Includes all jitter sources
$T_{TX—DJ-DD}$	Tx deterministic jitter	0.19 (max)	UI		Deterministic jitter only assuming the Dual Dirac distribution
$C_{TX-PARASITIC}$	Tx input capacitance for return loss	1.25 (max)	pf		Parasitic capacitance to ground

Symbol	Parameter	5.0 GT/s	Units	Normative	Comments
R_{TX-DC}	Transmitter DC common mode impedance	18 (min) 30 (max)	Ω		DC impedance limits to guarantee Receiver detect behavior. Measured with respect to AC ground over a voltage of 0-500mV.
$I_{TX-SHORT}$	Transmitter short-circuit current limit	60 (max)	mA		The total current Transmitter can supply when shorted to ground.
$V_{TX-DC-CM}$	Transmitter DC common-mode voltage	0 (min) 2.2 (max)	V		The instantaneous allowed DC common-mode voltages at the connector side of the AC coupling capacitors.
$V_{TX-CM-AC-PP_ACTIVE}$	Tx AC common mode voltage active	100 mv	mVPP		Max mismatch from D+ D- for both time and amplitude. While signaling.
$V_{TX-CM-DC-ACTIVE-IDLE-DELTA}$	Absolute Common Mode Voltage between *U1* and *U0*	200 (max)	mV		peak
$V_{TX-IDLE-DIFF-AC-pp}$	Electrical Idle Differential Peak – Peak Output Voltage	0 (min) 10 (max)	mV		
$V_{TX-IDLE-DIFF-DC}$	DC Electrical Idle Differential Output Voltage	0 (min) 10 (max)	mV		Voltage must be low pass filtered to remove any AC component. This limits the common mode error when resuming U1 to U0

6.7.2 Transmitter Eye

The eye mask is measured using the compliance data pattern CD1 described in Section 6.4.4. Eye Height is measured from 10^6 UI. Jitter is extrapolated from 10^6 UI to 10^{-12}.

Table 6-12. Normative Transmitter Eye Mask

Signal Characteristic	Minimal	Nominal	Maximum	Units	Note
Eye Height	100		1200	mV	2
Dj			93	ps	1,2,3
Rj			60	ps	1,2,3
Tj			132	ps	1,2,3

Notes:

1. Measured over 10^6 UI and extrapolated to 10^{-12} BER
2. Measured after receiver equalization function
3. Measured at end of reference channel and cables at TP1 figure 6-14

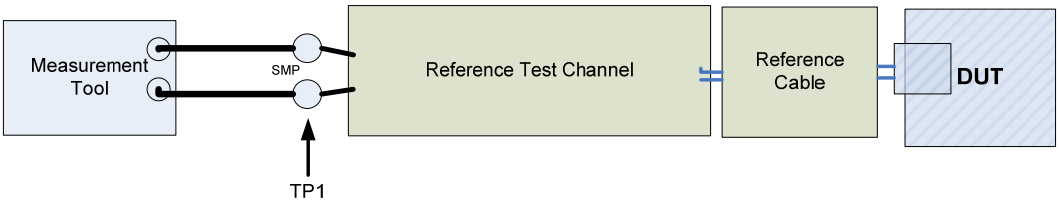


Figure 6-14. Tx Normative Setup with Reference Channel

6.7.3 Tx Compliance Reference Receiver Equalize Function

The normative transmitter eye is captured at the end of the reference channel. At this point the eye may be closed. To open the eye so it can be measured a reference Rx equalizer is defined.

(10) $$H(s) = \frac{A_{dc}\omega_{p1}\omega_{p2}}{\omega_z} \cdot \frac{s+\omega_z}{(s+\omega_{p1})(s+\omega_{p2})}$$

Plotted in: Figure 6-15 is the Compliance EQ transfer function.

(11)
$$A = 0.8$$
$$p1 = 2\pi 3.75x10^9$$
$$p2 = 2\pi 5x10^9$$
$$z1 = 2\pi 750x10^6$$

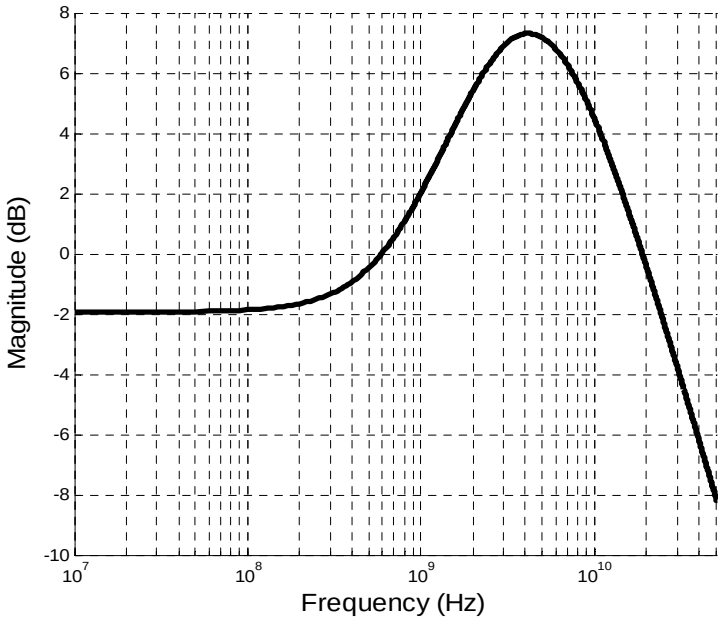


Figure 6-15. Tx Compliance Rx EQ Transfer Function

6.7.4 Informative Transmitter De-emphasis

The channel budgets and eye diagrams were derived using a $V_{TX\text{-}DE\text{-}RATIO}$ of transmit de-emphasis for both the Host and the Device reference channels. The differential peak-peak de-emphasis waveform is shown in Figure 6-16.

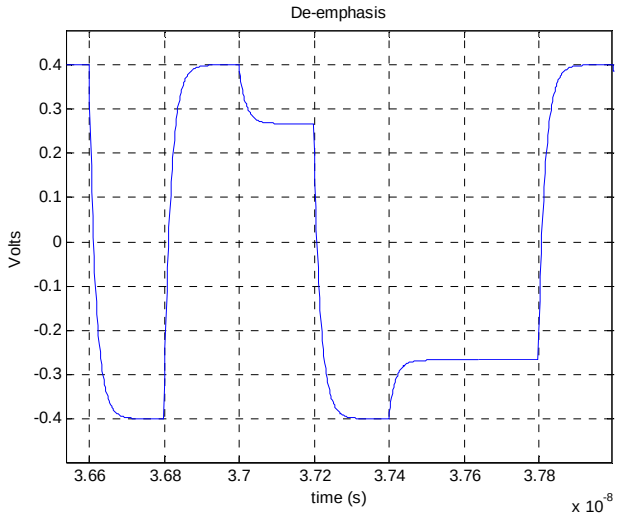


Figure 6-16. De-Emphasis Waveform

6.7.5 Entry into Electrical Idle, U1

Electrical Idle is a steady state condition where the Transmitter D+ and D- voltages are held constant at the same value and the Receiver Termination is within the range specified by $Z_{RX\text{-}DC}$. Electrical Idle is used in the power saving state of U1.

The low impedance common mode and differential Receiver terminations values (see Table 6-13) must be met in Electrical Idle. The Transmitter can be in either a low or high impedance mode during Electrical Idle.

6.8 Receiver Specifications

6.8.1 Receiver Equalization Training

The receiver equalization training sequence, detailed in Section 6.3.1, can be used to train the receiver equalizer. The TSEQ training sequence is designed to provide a spectrally rich data pattern that is useful for training typical receiver equalization architectures. In addition, a high edge density pattern is interleaved with the data to help the CDR maintain bit lock. The TSEQ training sequence repeats 65536 times to allow for testing many coefficient settings. No SKPs are inserted during the TSEQ training sequence. The frequency spectrum of the TSEQ sequence is shown in Figure 6-17.

Receiver equalization training is implementation specific.

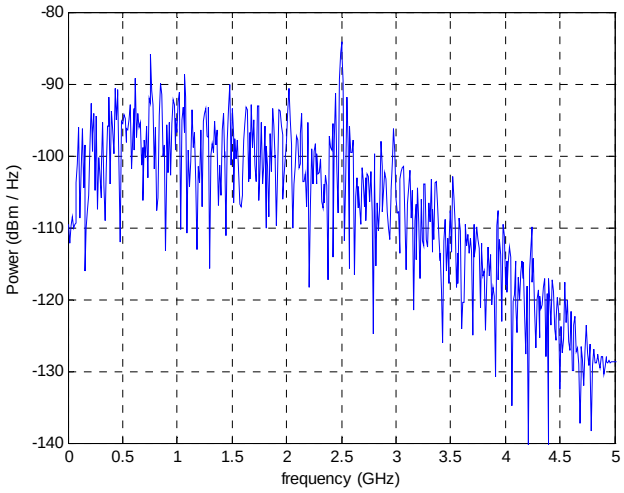


Figure 6-17. Frequency Spectrum of TSEQ

6.8.2 Informative Receiver CTLE Function

USB 3.0 requires receiver equalization to meet system timing and voltage margins. For long cables and channels the eye at the Rx is closed, and there is no meaningful eye without first applying an equalization function. The Rx equalizer may be required to adapt to different channel losses using the Rx EQ training period. The exact Rx equalizer and training method is implementation specific.

The equation for the CTLE used to develop spec is the compliance EQ function, see Section 6.7.3.

6.8.3 Receiver Electrical Parameters

Normative specifications are to be measured at the connector. Peak (P) and peak- peak (PP) are defined in Section 6.6.2.

Table 6-13. Receiver Electrical Parameters

Symbol	Parameter	5.0 GT/s	Units	Normative	Comments
UI	Unit Interval	*199.94* (min) *200.06* (max)	ps	Normative	UI does not account for SSC caused variations.
$V_{RX-DIFF-PP-POST-EQ}$	Differential Rx peak-peak voltage	*30* (min)	mV		Measured after the Rx EQ function, Section 6.8.2
T_{RX-TJ}	Max Rx inherent timing error	*0.45* (max)	UI		Measured after the Rx EQ function, Section 6.8.2
$T_{RX-DJ-DD}$	Max Rx inherent deterministic timing error	*0.3* (max)	UI		Max Rx inherent deterministic timing error
$C_{RX-PARASITIC}$	Rx input capacitance for return loss	1.1 (max)	pf		
R_{RX-DC}	Receiver DC common mode impedance	18 (min) 30 (max)	Ω	Normative	DC impedance limits are needed to guarantee Receiver detect. Measured with respect to ground over a voltage of 500 mV max.
$R_{RX-DIFF-DC}$	DC differential impedance	72 (min) 120 (max)	Ω	Normative	
$V_{RX-CM-AC-P}$	Rx AC common mode voltage	150 (max)	mV Peak		Measured at Rx pins into a pair of 50 Ω terminations into ground. Includes Tx and channel conversion, AC range up to 5 GHz
$V_{RX-CM-DC-ACTIVE-IDLE-DELTA_P}$	Rx AC common mode voltage during the U1 to U0 transition	200 (max)	mV Peak		Measured at Rx pins into a pair of 50 Ω terminations into ground. Includes Tx and channel conversion, AC range up to 5 GHz
$Z_{RX-HIGH-IMP-DC-POS}$[1]	DC Input CM Input Impedance for V>0 during Reset or power down	25 k (min)	Ω	Normative	Rx DC CM impedance with the Rx terminations not powered, measured over the range 0 – 500 mV with respect to ground.
$V_{RX-LFPS-DET-DIFFp-p}$	LFPS Detect Threshold	100 (min) 300 (max)	mV	Normative	Below the min is noise Must wake up above the max.

Note

1. Only DC Input CM Input Impedance for V>0 is specified. DC Input CM Input Impedance for V<0 is not guaranteed and could be as low as 0 ohm.

6.8.4 Receiver Loopback

The entry and exit process for receiver loopback is described in chapter 7.

Receiver loopback must be retimed. Direct connection from the Rx amplifier to the transmitter is not allowed for loopback mode. The receiver must continue to process SKPs as appropriate. SKP symbols must be consumed or inserted as required for proper clock tolerance compensation. Over runs or under runs of the clock tolerance buffers will reset the buffers to the neutral position.

During loopback the receiver must process the Bit Error Rate Test (BERT) commands.

Loopback must occur in the 10bit domain. No error correction is allowed. All symbols must be transmitted as received with the exception of SKP and BERT commands.

6.7.4.1 Loopback BERT

During loopback the receiver processes the BERT ordered sets BRST, BDAT, and BERC. These ordered sets are given in Table 6-14 through Table 6-17. BRST and BDAT are looped back as received. BERC ordered sets are not looped back but are replaced with BCNT ordered sets. Anytime a BRST is received the error count register EC is set to 0 and the scrambling LFSR is set to 0FFFF h. Any number of consecutive BRST ordered sets may be received.

BRST followed by BDAT starts the bit error rate test. BDAT sequence is the output of the scrambler and is equivalent to the logical idle sequence. It consists of scrambled 0 as described in Appendix B. As listed in Appendix B, the first 16 characters of the sequence are reprinted here:

FF	17	C0	14	B2	E7	02	82	72	6E	28	A6	BE	6D	BF	8D

The receiver must compare the received data to the BDAT sequence. Errors increment the error count register (EC) by 1. EC may not rollover but must be held at FFFF h. The LFSR is advanced once for every character except SKPs. The LFSR rolls over after 2^16 symbols. SKPs must be inserted or deleted as necessary for clock tolerance compensation.

The BERC command does not increment the error count register. The LFSR is advanced. The BERC ordered set is replaced by the BCNT ordered set. The BCNT ordered set includes the 8b10b encoded error count (EC) register.

BERC may be sent multiple times. The EC register is not cleared by BERC ordered sets.

BERT continues until the loopback mode is terminated as described in Chapter 7.

Table 6-14. BRST

Symbol Number	Encoded Values	Description
0	K28.5	COM
1	K28.7	BRST

Table 6-15. BDAT

Symbol Number	Encoded Values	Description
D0.0 <0:n>	Electrical Idle see appendix B	Scrambled 0 Rolls over after 2^16 symbols

Table 6-16. BERC

Symbol Number	Encoded Values	Description
0	K28.3	BERC
1	K28.3	BERC
2	K28.3	BERC
3	K28.3	BERC

Table 6-17. BCNT

Symbol Number	Encoded Values	Description
0	K28.5	COM
EC<0:7>	DCODE	Error count LSB
EC<8:15>	DCODE	Error count MSB
K29.7	END	End EC

6.8.5 Normative Receiver Tolerance Compliance Test

The receiver tolerance test is tested in the compliance reference channel. A pattern generator will send a compliance test pattern and jitter through the compliance reference channels to the receiver. The receiver will loop back the data and any difference in the pattern sent from the pattern generator and returned will be an error. Additional details on the receiver compliance test will be made available in a future document, “USB Compliance Methodology”.

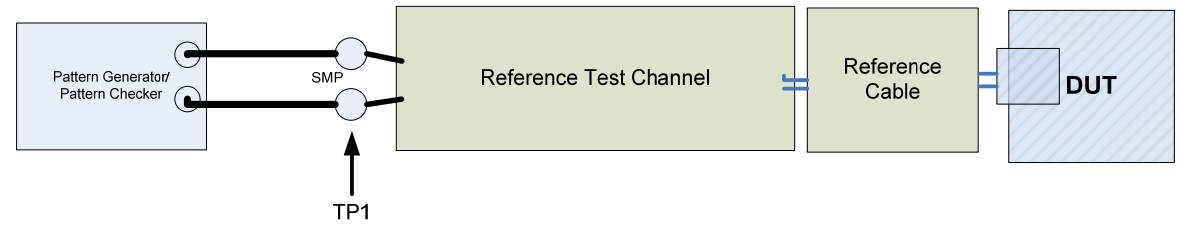


Figure 6-18. Rx Tolerance Setup

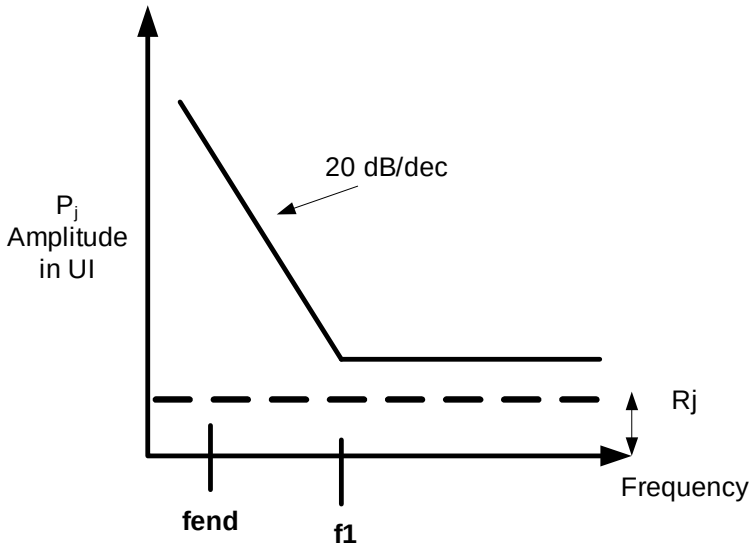


Figure 6-19. Jitter Tolerance Curve

The jitter used to test the receiver shall meet the requirements of Table 6-18.

Table 6-18. Jitter Requirements

Symbol	Parameter	Value	Units	Notes
F1	Tolerance corner	4.9	MHz	
J_{Rj}	Random Jitter	0.0121	UI rms	
$J_{Rj_p\text{-}p}$	Random Jitter peak- peak at 10^{-12}	0.17	UI p-p	
J_{Pj_500kHZ}	Sinusoidal Jitter	8	UI p-p	
J_{Pj_1Mhz}	Sinusoidal Jitter	4	UI p-p	
J_{Pj_2MHz}	Sinusoidal Jitter	2	UI p-p	
J_{Pj_f1}	Sinusoidal Jitter	0.2	UI p-p	
J_{Pj_50MHz}	Sinusoidal Jitter	0.2	UI p-p	
V_full_swing	Transition bit differential voltage swing	0.75	V p-p	
V_EQ_level	Non transition bit voltage (equalization)	-3	dB	

Notes:

1. All parameters measured at TP1
2. Due to time limitations at compliance testing only a subset of frequencies can be tested. However, the Rx is required to tolerate Pj at all frequency in-between the compliance test points.
3. During Rx tolerance test SSC is generated by test equipment and present at all times. Each J_{Pj} source is then added and tested to spec limit one at a time.

6.9 Low Frequency Periodic Signaling (LFPS)

Low frequency periodic signaling (LFPS) is used for side band communication between the two Ports across a link that is either in a low power state or an error state.

6.9.1 LFPS Signal Definition

Table 6-19 defines the LFPS electrical specification at the Transmitter. A typical LFPS waveform is shown in Figure 6-20. T_period is the period of an LFPS signal. An LFPS burst is the transmission of continuous LFPS signal over a period of time defined by T_burst. An LFPS sequence is defined by the transmission of an LFPS burst over a period of time defined by T_ repeat.

An LFPS message is defined based on the variation of T_burst and T_ repeat. The LFPS messages include Polling.LFPS, LFPS.Ping, U1/U2 exit, U3 wakeup, and Warm Reset. The detailed use of LFPS is specified in Chapter 7.

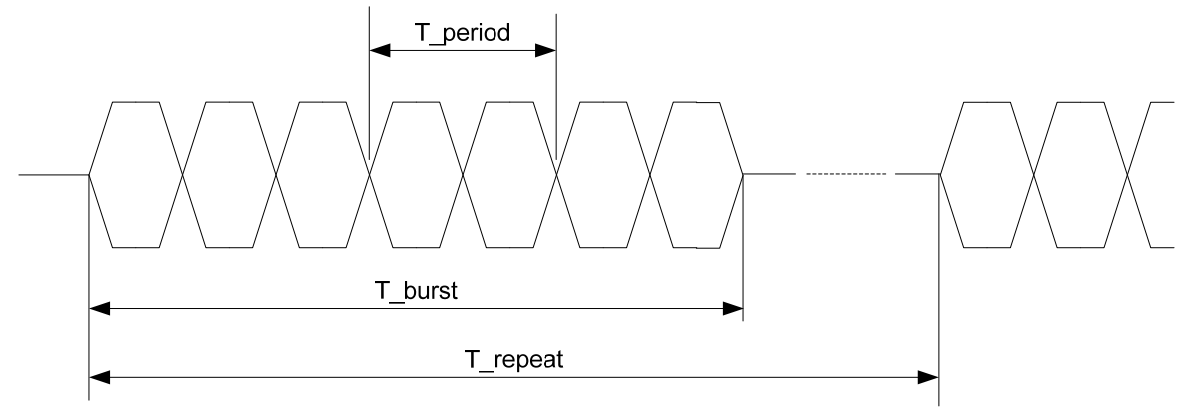


Figure 6-20. LFPS Signaling

Table 6-19. Normative LFPS Electrical Specification

Symbol	Minimum	Nominal	Max	Units
T_period	20	---	100	ns
V_CM_AC_LFPS (see Table 6-10)			$V_{TX\text{-}CM\text{-}AC\text{-}PP_ACTIVE_RMS}$	mV
V_CM_LFPS_active			10	mV
Amplitude (Vpp differential)	800		1200	mV
t_rise_fall_20_80			0.2 * T_Period	s

Table 6-20. LFPS Timing

	T_burst			T_repeat [2]		
	Min	Typ	Max	Min	Typ	Max
Polling.LFPS	0.8 us		1.2 us	8 us		12 us
Ping.LFPS		2*T_Period		160 ms		240 ms
TReset	80 ms		120 ms			
U1 Exit[1]			900 us			
U2 / Loopback Exit[1]	80 us		2 ms			
U3 Exit[1]	80 us		10 ms			

Notes:

1. LFPS must be transmitted for the burst minimum time. The LFPS handshake process and timing defined in Section 6.9.2 apply. If no LFPS is detected within the maximum time while polling for the LFPS handshake then it is a failure to exit.
2. The number of repeats is defined in Chapter 7.

6.9.2 Example LFPS Handshake for U1/U2 Exit, Loopback Exit, and U3 Wakeup

The LFPS signal used for U1/U2 exit, Loopback Exit, and U3 wakeup is defined the same as continuous LFPS signals with the exception of timeout values defined in Table 6-21. The handshake process for U1/U2 exit and U3 wakeup is illustrated in Figure 6-21. Link partner 1 and link partner 2 are connected together on a single link. The timing requirements are different for U1 exit, U2exit, and U3 wakeup. They are listed in Table 6-21.

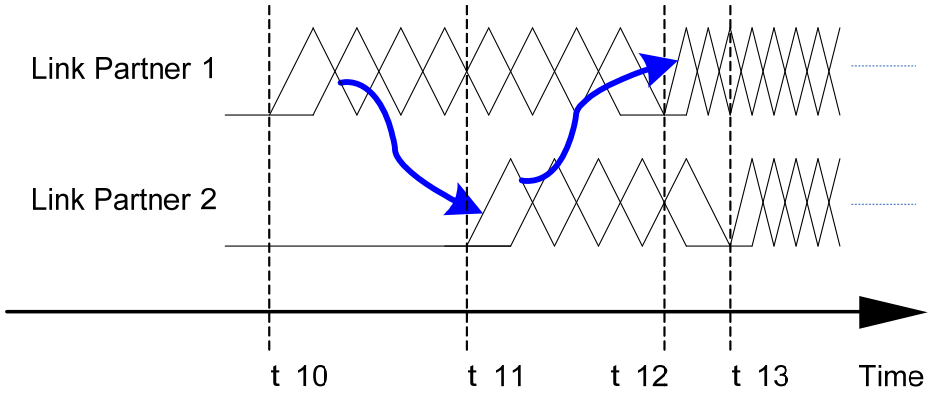


Figure 6-21. U1 Exit, U2 Exit, and U3 Wakeup LFPS Handshake Timing Diagram

The handshake process is as follows:

- Link partner 1 initiates exit by transmitting LFPS at time t10 (see Figure 6-21). LFPS transmission shall continue until the handshake is declared either successful or failed.
- Link partner 2 detects valid LFPS on its receiver, and responds by transmitting LFPS at time t11. LFPS transmission shall continue until the handshake is declared either successful or failed.
- A successful handshake is declared for Link partner 1 if it receives valid LFPS from Link partner 2 within before the maximum time.

- A successful handshake is declared for Link partner 1 if the following conditions are met within No-LFPS-Response-Timeout after t10 (see Figure 6-21 and Table 6-21):
  1. Valid LFPS is received from Link partner 2.
  2. For U1 exit, U2 exit and U3 Wakeup Link partner 1 is ready to transmit valid training sequences. This does not apply to Loopback exit.
- A successful handshake is declared for Link partner 2 if the following conditions are met within No-LFPS-Response-Timeout after t11:
  1. Link partner 2 has transmitted the minimum LFPS defined as (t13 – t11) in Table 6-21.
  2. For U1 exit, U2 exit, and U3 Wakeup Link partner 2 is ready to transmit valid training sequences. This does not apply to Loopback exit.
- Link partner 1 declares a failed handshake if its successful handshake conditions were not met.
- Link partner 2 declares a failed handshake if its successful handshake conditions were not met.
- For U1 exit, U2 exit and U3 Wakeup for a given port the maximum duration between LFPS transmission and training sequence transmission is 20ns.

Table 6-21. LFPS Handshake Timing for U1/U2 Exit, Loopback Exit, and U3 Wakeup

	U1 Exit[1]		U2/Loopback Exit[1]		U3 Wakeup	
	Min	**Max**	**Min**	**Max**	**Min**	**Max**
t11 – t10	0.3 us	0.9 us	0.3 us	2 ms	0.3 us	10 ms
t13 – t10				2 ms		20 ms
t12 – t11	0	0.9 us	0	2 ms	0	10 ms
t13 – t11	0.6 us	0.8 us	80 us	2 ms	80 us	10 ms
t12 – t10				2 ms		10 ms
No-LFPS-Response-Timeout	Min	Max	Min	Max	Min	Max

Note:

If one link partner is in U1 and the other is in U2, then U2 timing shall apply.

6.9.3 Example Warm Reset

A Warm Reset is a reset issued only by a Downstream Port to an Upstream Port when the link is in an error state.

Figure 6-22 shows the timing diagram of warm_reset generation and detection.

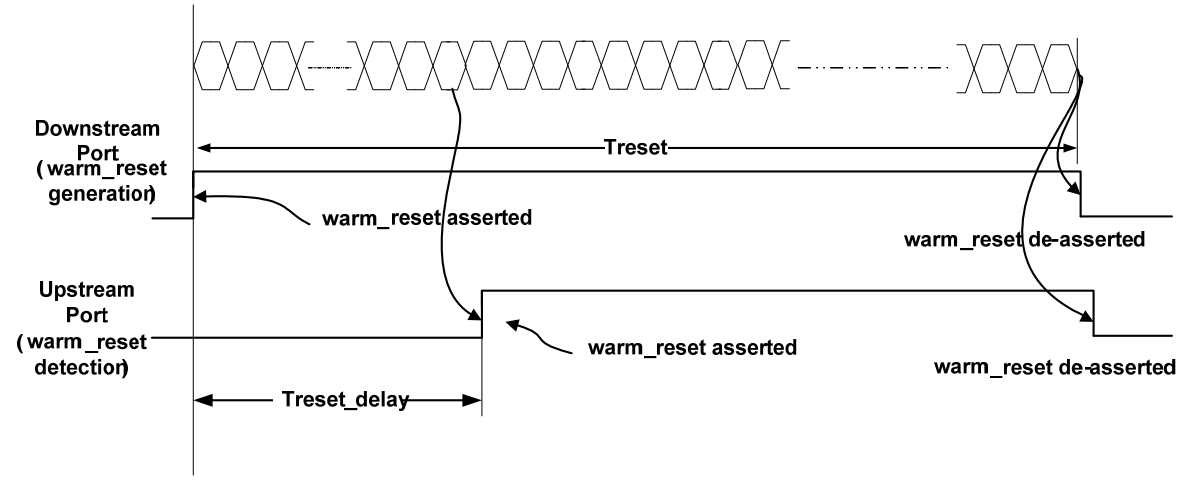


Figure 6-22. Example of Warm Reset Out of U3

6.10 Transmitter and Receiver DC Specifications

6.10.1 Informative ESD Protection

All signal and power pins must withstand 2000 V of ESD using the human body model and 500 V using the charged device model without damage. Class 2 per JEDEC JESE22-A114-A. This ESD protection mechanism also helps protect the powered down Receiver from potential common mode transients during certain possible reset or surprise insertion situations.

6.10.2 Informative Short Circuit Requirements

All Transmitters and Receivers must support surprise hot insertion/removal without damage to the component. The Transmitter and Receiver must be capable of withstanding sustained short circuit to ground of D+ and D-.

6.10.3 Normative High Impedance Reflections

During an asynchronous reset event one device may be reset while the other device is transmitting. The device under reset is required to disconnect the receiver termination. During this time the device under reset may be receiving active data. Since the data is not terminated the differential voltage into the receiver will be doubled. For a short channel, the receiver may experience a total of 2* $V_{DIFF.}$

The receiver must tolerate this doubling of the negative voltage that can occur if the Rx termination is disconnected. A part must tolerate a 20 ms event that doubles the voltage on the receiver input when the termination is disconnected 10,000 times over the life time of the part.

6.11 Receiver Detection

6.11.1 Rx Detect Overview

The Receiver Detection circuit is implemented as part of a Transmitter and must correctly detect whether a load impedance equivalent to a DC impedance Z_{RX-DC} (Table 6-13) is present. The Rx detection operates on the principle of the RC time constant of the circuit. This time constant changes based on the presence of the receiver termination. This is conceptually illustrated in Figure 6-23. In this figure, R_detect is the implementation specific charging resistor. C_AC is the AC capacitor that is in the circuit only if R_Term is also present, otherwise, only C_Parasitic is present.

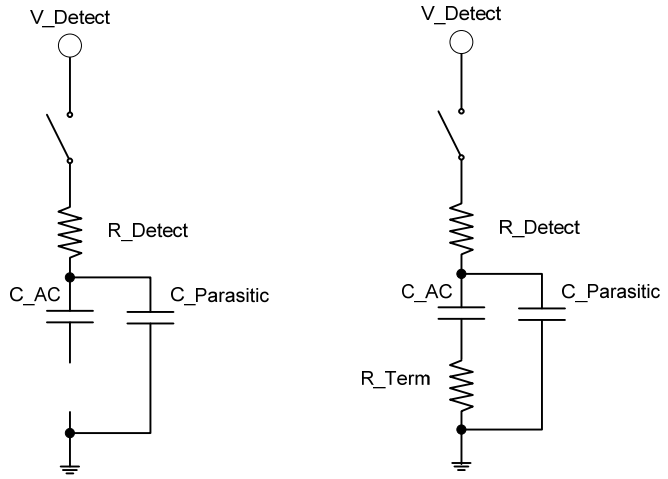


Figure 6-23. Rx Detect Schematic

The left side of Figure 6-23 shows the Receiver Detection circuit with no termination present and the right side is the same circuit with termination.

Detect voltage transition must be common mode. Detect voltage transition must conform to $V_{TX_RCV_DETECT}$. as described in Table 6-10.

The receiver detect sequence must be in the positive common mode direction only. Negative receiver detection is not allowed.

6.11.2 Rx Detect Sequence

The recommended behavior of the Receiver Detection sequence is:

1. A Transmitter must start at a stable voltage prior to the detect common mode shift.
2. A Transmitter changes the common mode voltage on D+ and D- consistent with detection of Receiver high impedance which is bounded by parameter $Z_{RX\text{-}HIGH\text{-}IMP\text{-}DC\text{-}POS}$ given in Table 6-13.
3. A Receiver is detected based on the rate that the lines change to the new voltage.
   - The Receiver is not present if the voltage at the Transmitter charges at a rate dictated only by the Transmitter impedance and the capacitance of the interconnect and series capacitor.
   - The Receiver is present if the voltage at the Transmitter charges at a rate dictated by the Transmitter impedance, the series capacitor, the interconnect capacitance, and the Receiver termination.

Any time Electrical Idle is exited the detect sequence does not have to execute or may be aborted. During the Device connect, the Device receiver has to guarantee it is always in high impedance state while its power plane is stabilizing. This is required to avoid the Host falsely detecting the Device and starting the training sequence before the Device is ready. Similarly a disabled port has to keep its receiver termination in high impedance which is bounded by parameters $Z_{RX\text{-}HIGH\text{-}IMP\text{-}DC\text{-}POS}$ until directed by higher layer to exit from the Disabled state. In contrast, a port which is at U1/U2/U3 Electrical Idle must have it Receiver Termination turned on and met the $Z_{RX\text{-}DC}$ specification.

6.11.3 Upper Limit on Channel Capacitance

The interconnect total capacitance to ground seen by the Receiver Detection circuit must not exceed 3 nF to ground, including capacitance added by attached test instrumentation. This limit is needed to guarantee proper operation during Receiver detect. Note that this capacitance is separate and distinct from the AC coupling capacitance value.

7 Link Layer

The Link layer's responsibilities are to maintain the Link connectivity so that successful packet transactions between the two Ports across the Link are ensured. To increase the error tolerance due to bit errors introduced in the physical layer and channel, special coding schemes are added into the Link layer packet framings and packet data such that packet framings have a single-bit error tolerance and packet data integrity can be also verified when a packet is received.

The Link layer also facilitates the Link training, the Link testing/debugging, and the Link power management.

The focus of this chapter is to define the packet and Link command structures, the Link-level control and power management functions, the Link Training Status State Machine (LTSSM) definitions and the Link error recovery policies.

The chapter addresses the following in detail:

- Packet Framing
- Link command
- Link Management and Flow Control
- Link Error Rules/Recovery
- Resets
- Link State Machine descriptions

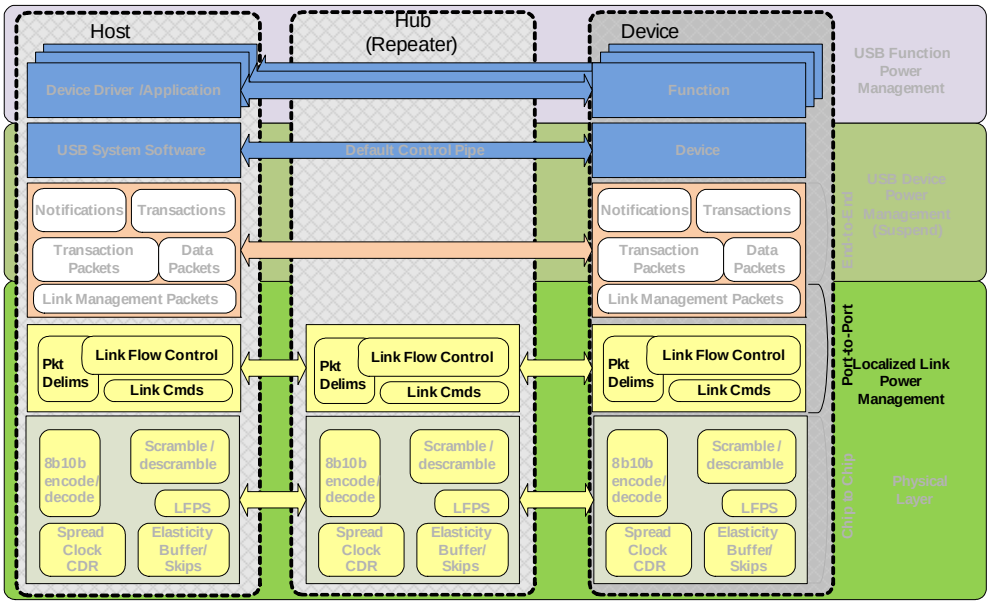


Figure 7-1. Link Layer

7.1 Link Management and Flow Control

This section contains information regarding the Link management and the Link flow control.

- The Packet and Packet Framing section defines packet types, packet structures and CRC requirements for each Packet.
- The Link Command section defines special Link Command structures that control various functionalities at the Link Level.
- The Logical Idle defines a special Symbol used in U0.
- The Flow control defines a set of handshake rules for Packet Transactions.

7.1.1 Packets and Packet Framing

SuperSpeed uses Packets to transfer information. Detailed Packet formats for Link Management Packets, Transaction Packets, Isochronous Time Stamp Packets, and Data Packets are defined in Section 8.2.

7.1.1.1 Header Packet Structure

All Header Packets are 20-Symbols long, as is formatted in Figure 7-2. This includes Link Management Packets, Transaction Packets, Isochronous Time Stamp Packets, and Data Packet Headers. A Header Packet consists of three parts including a Header Packet Framing, a Packet Header, and a Link control word.

7.1.1.1.1 Header Packet framing

This is a 4-symbol Packet Start Framing Ordered Set based on K-symbols. It is defined as three consecutive symbols of SHP followed by a K-symbol EPF. A Header packet shall always begin with a Packet Start Framing Ordered Set. The construction of the Header Packet Framing is to achieve a single-bit error tolerance.

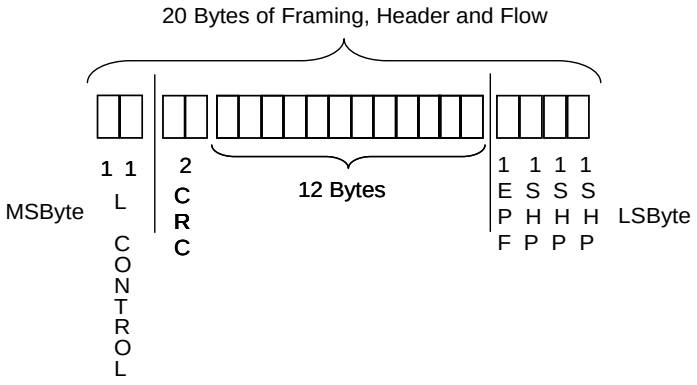


Figure 7-2. Packet Header with Framing Symbol, CRC-16 and Link Control Word

7.1.1.1.2 Packet Header

A Packet Header consists of 14-bytes as formatted in Figure 7-3. It includes a 12-byte Header information and a 2-byte CRC-16. CRC-16 is used to detect the data integrity of the 12-byte Header information.

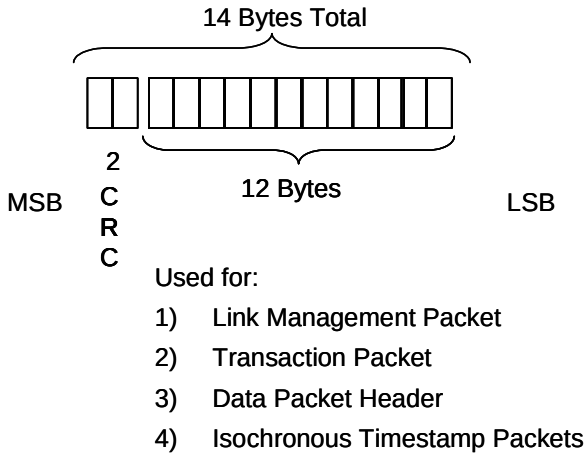


Figure 7-3. Packet Header

The implementation of CRC-16 on the Packet Header is defined below:

- The polynomial for CRC-16 shall be = 100Bh
- The initial value of CRC-16 shall be = FFFFh
- CRC-16 shall be calculated for all 12 Bytes of the Header, not inclusive of any packet framing Symbols.
- CRC-16 calculation shall begin at Byte 0 bit 0 and continue to bit 7 of each of the 12 Bytes.
- The remainder of CRC-16 shall be complemented.

Note: The CRC-16 polynomial is not the same as the one used for USB 2.0.

Figure 7-4 is an illustration of CRC-16 generation. The output bit ordering is concluded in Table 7-1.

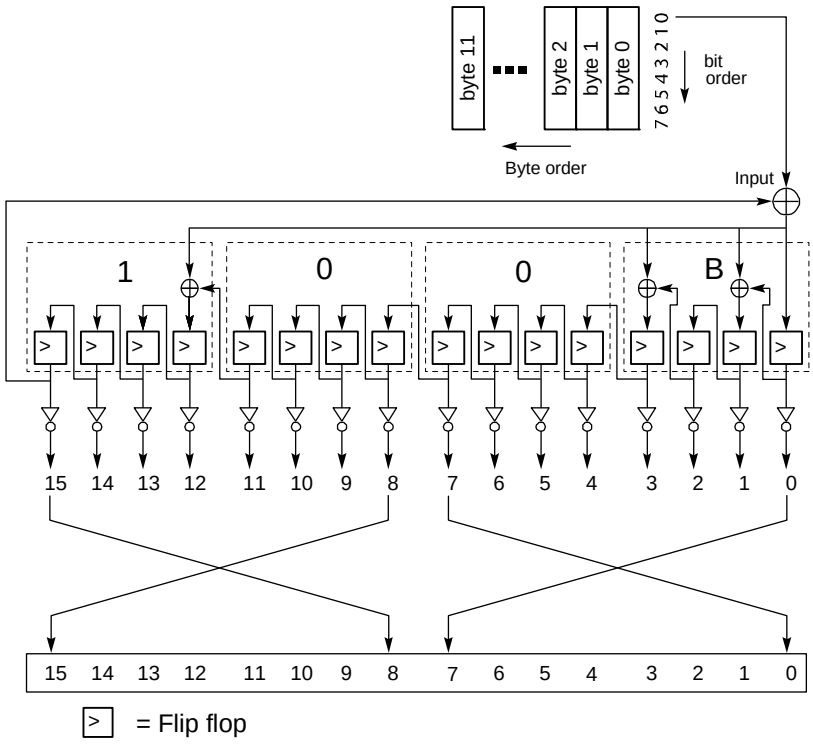


Figure 7-4. CRC-16 Generation

Table 7-1. CRC-16 Mapping

CRC-16 Result bit	Position in CRC-16 Field
0	7
1	**6**
2	**5**
3	4
4	3
5	2
6	1
7	0
8	15
9	14
10	13
11	12
12	11
13	10
14	9
15	8

7.1.1.1.3 Link Control Word

The 2-Byte Link Control Word is formatted as shown in Figure 7-5. It is used for both Link-Level and End-to-End flow control. Detailed descriptions of each bit are given in Section 7.3.1.1.

The Link Control Word shall contain a 3-bit Header Sequence number, 3-bit Reserved, a 3-bit Hub depth index, a Delayed bit (DL), a Deferred bit (DF) and a 5-bit CRC-5.

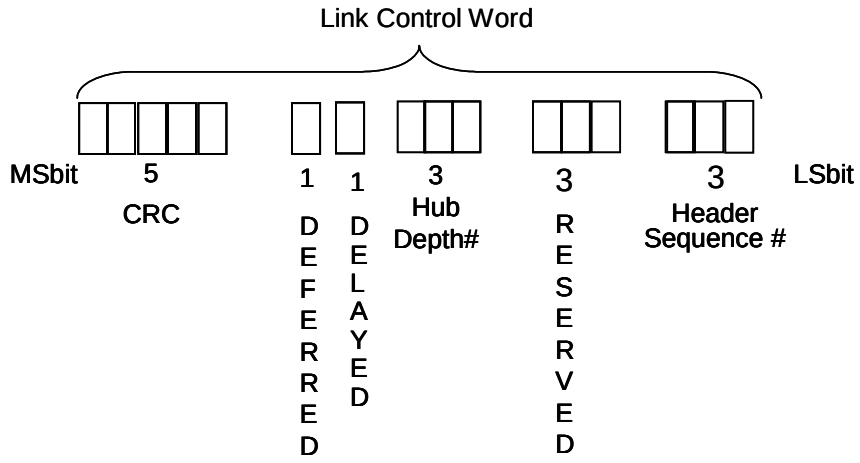


Figure 7-5. Link Control Word

CRC-5 protects the data integrity of the Link Control Word. The implementation of CRC-5 is defined below:

- The CRC-5 Polynomial shall be = 00101b
- The Initial value for the CRC-5 shall be = 11111b
- CRC-5 is calculated for the remaining 11 bits of the Link Control Word.
- CRC-5 calculation shall begin at bit 0 and proceed to bit 10.
- The remainder of CRC-5 shall be complemented.

Figure 7-6 is an illustration of CRC-5 generation.

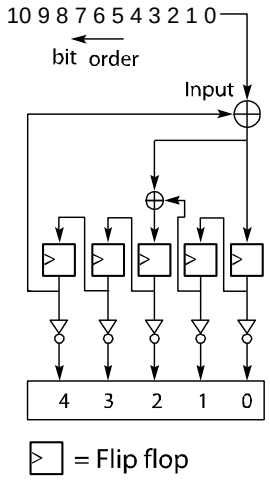


Figure 7-6. CRC-5 Generation

7.1.1.2 Data Packet Payload Structure

Data Packets are a special type of Packet consisting of a Data Packet Header (DPH) and a Data Packet Payload (DPP). Data Packet Header is defined in Section 7.1.1.1. Data Packet Payload, on the other hand, consists of a data packet payload framing, and a variable length of data followed by 4-bytes of CRC-32. Figure 7-7 describes the format of a Data Packet Payload.

7.1.1.2.1 Data Packet Payload Framing

Data Packet Payload Framing consists of total of 8 symbols, a 4-symbol Data Packet Payload Starting Frame Ordered Set and a 4-symbol Data Packet Payload Ending Frame Ordered Set. As indicated by Figure 7-7, a Data Packet Payload Starting Frame Ordered Set consists of three consecutive K-symbols of SDP followed by a single K-symbol of EPF. A Data Packet Payload Ending Frame Ordered Set consists three consecutive K-symbol of END or EDB (End of nullified packet) followed by a single K-symbol of EPF.

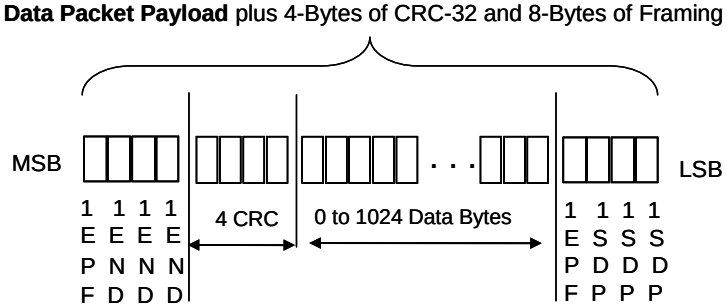


Figure 7-7. Data Packet Payload with CRC-32 and Framing

7.1.1.2.2 Data Packet Payload

The Data Packet Payload section consists of 0 to 1024 data bytes followed by 4-bytes CRC-32. Any premature termination of a Data Payload Packet shall contain an EDB Data Packet Payload Ending Frame. A Data Packet Payload shall immediately follow its corresponding Data Packet Header with no spacing in between.

CRC-32 protects the data integrity of the Data Payload. CRC-32 is defined below:

- The CRC-32 Polynomial shall be = 04C1 1DB7h
- The CRC-32 Initial value shall be = FFFF FFFFh
- CRC-32 shall be calculated for all Bytes of the Data Packet Payload, not inclusive of any packet framing Symbols.
- CRC-32 calculation shall begin at Byte 0 bit 0 and continue to bit 7 of each of the Bytes of the Data Packet Payload.
- The remainder of CRC-32 shall be complemented.

Figure 7-8 is an illustration of CRC-32 generation. The output bit ordering is listed in Table 7-2.

Table 7-2. CRC-32 Mapping

CRC-32 Result bit	Position in CRC-32 Field
0	7
1	**6**
2	**5**
3	4
4	3
5	2
6	1
7	0
8	15
9	14
10	13
11	12
12	11
13	10
14	9
15	8
16	23
17	22
18	21
19	20
20	19
21	18
22	17
23	16
24	31
25	30
26	29
27	28
28	27
29	26
30	25
31	24

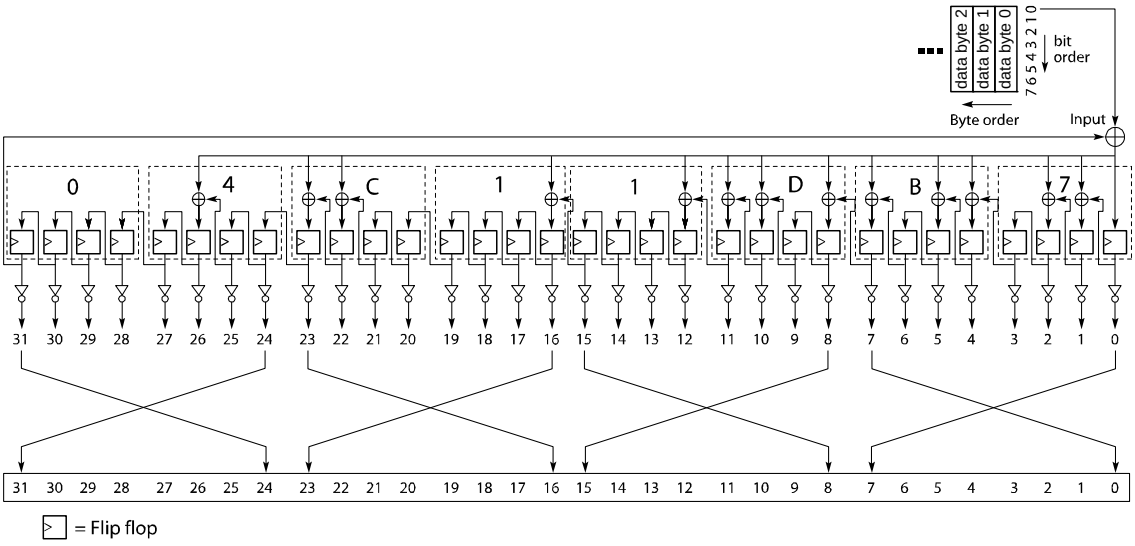


Figure 7-8. CRC-32 Generation

7.1.1.2.3 Spacing Between Data Header Packet and Data Payload Packet

There shall be no spacing between a Data Packet Header and its corresponding Data Packet Payload. This is illustrated in Figure 7-9.

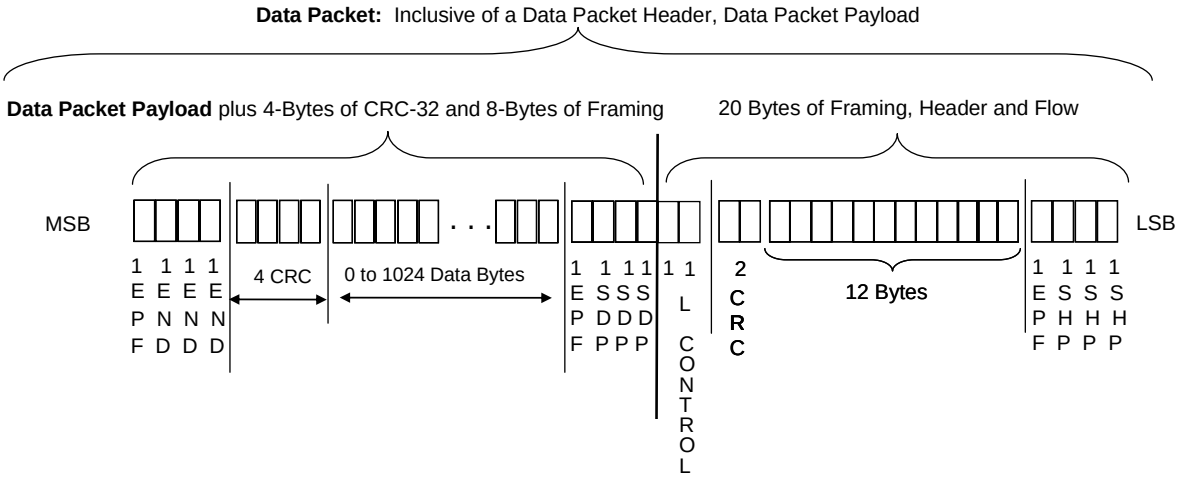


Figure 7-9. Data Packet with Data Packet Header followed by Data Packet Payload Without Spacing

Additional details on how Header Packets are transmitted and received at the Link Level are described in Section 7.1.4.

7.1.2 Link Commands

Link Commands are special Ordered Sets used for Link-level packet management such as flow control and Link-level power management. Link Commands are a fixed length of eight symbols and contain repeated Symbols to increase the error tolerance. See Section 7.2 for more details. Link Command names have the L-preface to differentiate their Link-level usage and to avoid confusion with Transaction Packets.

7.1.2.1 Link Command Structure

Link Command Ordered Sets shall be 8 Symbols long and constructed with the following format shown in Figure 7-10. The first four Symbols are Special K-Symbols consisting of three consecutive SLCs followed by EPF. The second four Symbols consist of a 2-Symbol Link Command Word and its replica. Both Link Command Words are scrambled. Table 7-3 summarizes the Link command structure.

Table 7-3. Link Command Ordered Set Structure

Symbol Number	Description
0	SLC (Start Link Command)
1	SLC (Start Link Command)
2	SLC (Start Link Command)
3	EPF
4~5	Link Command Word
6~7	Link Command Word

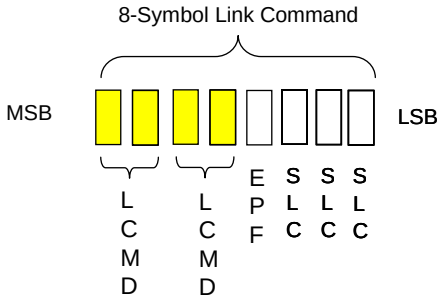


Figure 7-10. Link Command Structure

7.1.2.2 Link Command Word Definition

Link Command Word is 16-bits long with the 11-bit Link Command information protected by a 5-bit CRC-5. The 11-bit Link Command information is defined in Table 7-4. The calculation of CRC-5 is the same as Link Control Word illustrated in Figure 7-6.

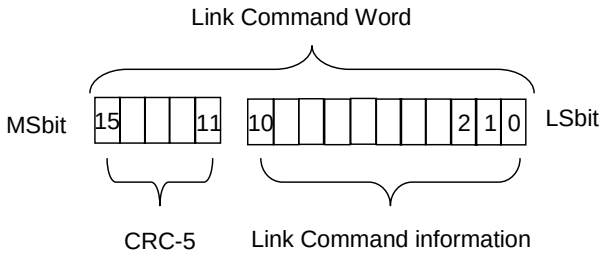


Figure 7-11. Link Command Word Structure

Table 7-4. Link Command Bit Definitions

Class		Type	b6~4	Sub-Type
b10~9	Link Command	b8~7		b3~0
00	LGOOD_n LRTY LBAD LCRD_x	00: LGOOD_n	Reserved (000)	b3: reserved. b2~0: HP Sequence number. 000: LGOOD_0; 001: LGOOD_1; 010: LGOOD_2; 011: LGOOD_3; 100: LGOOD_4; 101: LGOOD_5; 110: LGOOD_6; 111: LGOOD_7.
		01: LCRD_x		b3~2: reserved. b1~0: Rx Header buffer credit. 00: LCRD_A; 01: LCRD_B; 10: LCRD_C; 11: LCRD_D.
		10: LRTY 11: LBAD		Reserved (0000)
01	LGO_Ux LAU LXU LPMA	00: LGO_Ux		0001: LGO_U1; 0010: LGO_U2; 0011: LGO_U3; Others: reserved.
		01: LAU 10: LXU 11: LPMA		Reserved (0000)
10	LUP	00: LUP Others: reserved		Reserved (0000)
11: reserved	Reserved	Reserved (0000)		Reserved (0000)

Note: For the field that is reserved, a transmitter shall set the reserved field to 0.

Link Commands are defined for three usage cases. First, Link Commands are used to ensure the successful transfer of a Packet. Second, Link Commands are used for Link Power management. And third, a special Link Command is defined for an Upstream Port to signal its presence in U0.

Successful Header Packet transactions between the two Link Partners require proper Header Packet acknowledgement and Rx Header Buffer credit exchange. Header Packet acknowledgement and Rx Header buffer credit are separate. LGOOD_n (n= 0 to 7) and LBAD are used to acknowledge whether a Header Packet has been received properly or not. LCRD_A, LCRD_B, LCRD_C, and LCRD_D are used to signal the availability of Rx Header buffers in terms of credit. LRTY is used to signal that a Header Packet is re-sent. See Table 7-5 for details.

LGOOD_n uses an explicit numerical index called Header Sequence Number to represent the sequencing of a Header Packet. The Header Sequence Number starts from 0 and is incremented by one based on modulo-8 with each Header Packet. The index corresponds to the received Header Sequence number and is used for flow control and detection of lost or corrupted Header Packets.

LCRD_x uses an explicit alphabetical index. The index A, B, C, D, A, B, C… is advanced by one with each Header packet but does not correspond to the Header Sequence number. The index is used to ensure Rx Header Buffer credits are returned in order.

LBAD and LRTY do not use indexes.

LGO_U1, LGO_U2, LGO_U3, LAU, LXU and LPMA are Link Commands used for Link Power management.

LUP is a special Link Command used for an Upstream Port to indicate its Port presence in U0. The usage of LUP is described in Table 7-5.

Additional requirements and examples on the use of Link Commands are found in Section 7.4.1.

Table 7-5. Link Command Definitions

Link Command	Definition – See Sections 7.1.4.1, 7.1.4.2 and 7.4.6 for detailed use and requirements.
LGOOD_n	n = 0, 1, 2, ...7. Sent by a Port receiving a Header packet when all of the following conditions are true: • The Header Packet has a valid structure and can be recognized by the Receiver. • CRC-5 and CRC-16 are valid. • Length is valid. • Header Sequence number in the Received Header Packet matches the expected. • Buffer space in the Receiver is available for storing the received Header Packet. The numerical Index n is the Header Sequence number n. Receipt of LGOOD_n without the expected Header Sequence number relating to the corresponding Header Packet will cause the Transmitter to enter Recovery.
LBAD	Invalid Packet Length or CRC for either Link Control Word and/or Header Packet. Sent by a Port receiving the Header Packet in response to an invalid Header Packet. Packet that was received has corrupted CRC-5 and/or CRC-16, or bad length. Receipt of LBAD will cause a Port to resend all Header Packets after the last Header packet that has been acknowledged with LGOOD_n.

Link Command	Definition – See Sections 7.1.4.1, 7.1.4.2 and 7.4.6 for detailed use and requirements.
LCRD_x	x = A, B, C, D. Signifies that a single Receive Header buffer has been made available. The Rx Header Buffer credit can be updated. Sent by a Port after receiving a Header Packet that meets the following criteria • LGOOD_n is sent. • The Header Packet has been processed, and the corresponding Rx Header buffer is available to receive another Header Packet. The index letter does not correspond to the Header Sequence number. The Port receiving the Header Packet must send one individual LCRD_x for one Header Packet received. LCRD_x is sent in the index order of A, B, C, D, and back to A without skipping. A skipping of LCRD_x will cause the Link to transition to Recovery. e same Header Packet received implies missing of
LRTY	Sent by a Port before resending the first Header Packet in response to receipt of LBAD.
LGO_U1	Sent by a Port requesting the Link to enter U1.
LGO_U2	Sent by a Port requesting the Link to enter U2.
LGO_U3	Sent by a Downstream Port requesting the Link to enter U3. An Upstream Port must accept the request.
LAU	Sent by a Port accepting the request to enter U1, U2, or U3.
LXU	Sent by a Port rejecting the request to enter U1 or U2.
LPMA	Sent by a Port upon receiving LAU. Used in conjunction with LGO_Ux and LAU handshake to guarantee both Ports are in the same state.
LUP	Link Upstream Port. Link Command sent once after every 50 consecutive Idle symbols in U0 by the Upstream Port as a ping to inform its Port presence.

7.1.2.3 Link Command Placement

The Link Command placement shall meet the following rules.

- Link Commands shall not be placed inside Header Packet structures (i.e., within Link Management Packets, Transaction Packets, Isochronous Time Stamp Packet Header, or the Data Packet Header.)
- Link Commands shall not be placed within the Data Packet Payload of a Data Packet Structure.
- Link Commands shall not be placed between the Data Packet Header and Data Packet Payload.
- Link Commands may be placed before and after a Header Packet.
- Multiple Link Commands are allowed to be transmitted back to back.
- Link Commands shall not be sent until all scheduled SKP Ordered Sets have been transmitted.

  Note: Additional rules regarding scheduling of Link Commands are found in Section 10.7.5 to Section 10.7.12.

7.1.3 Logical Idle

Logical Idle is defined to be a period of one or more Symbol Times when no information (Packets, special K-Symbols or Link Commands) is being transferred on the Link. A special D-Symbol (00h), defined as Idle Symbol(IS), shall be transmitted by a Port during any time in U0 meeting the Logical Idle definition.

The Idle Symbol shall be scrambled according to rules described in Section 6.4.3.

Table 7-6. Logical Idle Definition

Symbol	Data Byte Name	Data Byte Value	Definition
IS	D0.0	00h	Represents Idle state on the bus.

7.1.4 Link Command Usage for Flow Control, Error Recovery and Power Management

Link Commands are used for Link Level Header Packet flow control, to identify lost/corrupted Header Packets and to initiate/acknowledge Link-Level power management transitions. The construction and descriptions for each Link Command are found in Section 7.1.2.

7.1.4.1 Header Packet Flow Control and Error Recovery

Header Packet flow control is used for all Header Packets. It requires each side of the Link to follow specific Header buffer and transmission ordering constraints to guarantee a successful packet transfer and Link interoperability. This section describes in details the rules of packet flow control.

7.1.4.1.1 Initialization

The Link initialization refers to initialization of a Port once a Link transitions to U0 from Polling, Recovery, or Hot Reset. The initialization includes the Header Sequence Numbers Advertisement and the Remote Rx Header Buffer Credit Count Advertisement between the two Ports before a Header Packet can be transmitted.

- The following requirements shall be applied to a Port.
  1. A Port shall maintain two Tx Header Sequence Numbers. The first Tx Header Sequence Number is defined as the Header Sequence number that will be added before the transmission of a Header Packet. The second Tx Header Sequence Number is ACK Tx Header Sequence Number. It is defined as the expected Header Sequence Number to be matched with the Header Sequence Number of LGOOD_n that is sent by a Port receiving the Header Packet.
  2. A Port shall have an Rx Header Sequence Number. It is defined as the expected Header Sequence Number when a Header Packet is received.
  3. A Port shall maintain two Header Buffer credit counts. The first is the Local Rx Header Buffer credit count. It is defined as the number of the available Rx Header Buffer credits of its receiver. The second is the Remote Rx Header Buffer credit count. It is defined as the number of the available Rx Header Buffer credits from its link partner.
  4. A Port shall have enough Tx Header Buffers in its transmitter to hold up to four unacknowledged Header Packets.
  5. A Port shall not transmit any Header Packet if its Remote Rx Header Buffer Credit Count is 0.
  6. A Port shall have enough Rx Header Buffers in its receiver to receive up to four Header Packets.
  7. Upon entry to U0 the following shall be performed in the sequence presented.
     a. A Port shall start the PENDING_HP timer and CREDIT_HP timer in expectation of Header Sequence number and Rx Header Buffer Credit Advertisement.
     b. A Port shall initiate the Header Sequence Number Advertisement
     c. A Port shall initiate the Rx Header Buffer Credit Advertisement.

- The Header Sequence Number Advertisement refers to Tx Header Sequence Number initialization by exchanging Header Sequence Numbers between the two Ports. This Header Sequence Number is the Header Sequence Number of the last Header Packet a Port has received and acknowledged with LGOOD_n. The main purpose of the Header Sequence Number Advertisement is to maintain the Link flow before and after Recovery such that a Port upon re-entry to U0 is aware what the last Header Packet is that was sent successfully prior to Recovery, and decides what Header Packets in its Tx Header Buffers that can be flushed or need to be re-transmitted. The following rules shall be applied during the Header Sequence Number Advertisement.
  1. A Port shall set its initial Rx Header Sequence Number defined in the following.
     a. If a Port enters U0 from Polling or Hot Reset, the Rx Header Sequence Number is 0;
     b. If a Port enters U0 from Recovery, the Rx Header Sequence Number is the same as the Rx Header Sequence Number before Recovery.
  2. A Port shall initiate the Header Sequence Number Advertisement by transmitting LGOOD_n with "n" equal to the Rx Header Sequence Number minus one.

     Note: The decrement is based on modulo-8 operation.
  3. A Port shall set its initial Tx Header Sequence number to the Sequence Number received during the Rx Header Sequence Number Advertisement plus one.
  4. A Port shall set its initial ACK Tx Header Sequence number to the Sequence Number received during the Rx Header Sequence Number Advertisement plus one.

     Note: The increment is based on modulo-8 operation.
  5. A Port shall not send any Header Packet or any other Link Commands except LGOOD_n until the Header Sequence Number Advertisement has been received and a Remote Rx Header Buffer Credit is available.
- The Remote Rx Header Buffer Credit Count Advertisement refers to Remote Rx Header Buffer Credit Count Initialization by exchanging the number of available Local Rx Header Buffers between the two Ports. The following rules shall be applied during the Rx Header Buffer Credit Advertisement.
  1. A Port shall initiate Rx Header Buffer Credit Advertisement upon completion of Header Sequence Number Advertisement.
  2. A Port shall initialize the following before sending the Rx Header Buffer Credit.
     a. A Port shall initialize its Tx Header Buffer Credit index to A.
     b. A Port shall initialize its Rx Header Buffer Credit index to A.
     c. A Port shall initialize its Remote Rx Header Buffer Credit Count to 0.
     d. A Port shall set its Local Rx Header Buffer Credit Count defined in the following.
        1. If a Port enters U0 from Polling or Hot Reset, its Local Rx Header Buffer Credit Count is 4;
        2. If a Port enters U0 from Recovery, its Local Rx Header Buffer Credit Count is the number of Rx Header Buffers available for incoming Header Packets.
  3. A Port shall perform the Remote Rx Header Buffer Credit Advertisement by transmitting LCRD_x to notify its link partner. A Port shall transmit one of the following based on its Local Rx Header Buffer Credit Count:
     a. LCRD_A if the Local Rx Header Buffer Credit Count is one.
     b. LCRD_A and LCRD_B if the Local Rx Header Buffer Credit Count is two.

c. LCRD_A, LCRD_B, and LCRD_C if the Local Rx Header Buffer Credit Count is three.
d. LCRD_A, LCRD_B, LCRD_C and LCRD_D if the Local Rx Header Buffer Credit Count is four.

4. A Port receiving LCRD_x from its Link Partner shall increment its Remote Rx Header Buffer Credit Count by one each time an LCRD_x is received.
5. A Port shall not transmit any Header Packet if its Remote Rx Header Buffer Credit Count is 0.

- The following rules shall be applied when a Port enters U0 from Recovery.
  1. A Port shall retain all the Header Packets in its Tx Header Buffers that have not been acknowledged by LGOOD_n.
  2. A Port shall drop all the Header Packets in its Rx Header Buffers that have not been acknowledged by LGOOD_n prior to entry to Recovery.
  3. A Port shall continue to process those Header Packets in its Rx Header Buffers that have been acknowledged with LGOOD_n prior to entry to Recovery, and update the Local Rx Header Buffer Credit Count.
  4. A Port shall maintain its Rx Header Sequence Number.
  5. A Port sending LBAD before Recovery shall not expect to receive LRTY before a re-tried Header Packet from its Link Partner upon entry to U0.
  6. A Port receiving LBAD before Recovery shall not send LRTY before a re-tried Header Packet to its Link partner upon entry to U0

  Note: there exists a situation where an LBAD was sent by a Port before Recovery and it may or may not be received properly by its Link partner. Under this situation, the rules of LBAD/LRTY do not apply. See Section 7.1.4.1.8 for details.
- Upon entry to Recovery and the next state is Hot Reset or Loopback, a Port may optionally continue its processing of all the packets acknowledged by LGOOD_n. A Port shall drop all packets that have not been acknowledged by LGOOD_n.

7.1.4.1.2 General Rules of LGOOD_n and LCRD_x Usage

- The Rx Header Buffer Credit shall be transmitted in the alphabetical order of LCRD_A, LCRD_B, LCRD_C, LCRD_D, and back to LCRD_A. LCRD_x received out of alphabetical order is considered as missing of a Link Command, and transition to Recovery shall be initiated.
- Header Packets shall be sent with the Header Sequence Number in the numerical order from 0 to 7, and back to 0. LGOOD_n received out of the numerical order is considered as missing of a Link Command, and the transition to Recovery shall be initiated.
- Transmitted packets not acknowledged by LGOOD_n prior to Recovery, shall be marked with Delayed bit set in the Link Control Word when resent after Recovery. Hosts shall be exempt from this requirement.
- When Packets to be transmitted are pending in a Port during Recovery, upper layers may continue to request a Port to transmit additional Packets. These newly submitted Packets shall be marked with the Delayed bit set. Hosts shall be exempt from this requirement if they have updated the timestamp prior to re-transmission of a Packet containing a timestamp.

7.1.4.1.3 Transmitting Header Packets

- Before sending a Header Packet, a Port shall add the Tx Header Sequence number corresponding to the Header Sequence number field in the Link Control Word.
- Transmission of a Header Packet shall consume a Tx Header Buffer. Accordingly, the Tx Header Sequence Number shall be incremented by one after the transmission or roll-over to 0 if the maximum Header sequence number is reached.
- Transmission of a retried Header Packet shall not consume an additional Tx Header Buffer and the Tx Header Sequence Number shall remain unchanged.
- Upon receiving LBAD, a Port shall send LRTY followed by resending all the Header Packets that have not been acknowledged with LGOOD_n.
- Prior to re-sending a Header Packet, a Port shall set the Delay bit within the Link Control word and re-calculate CRC-5.

  Note: CRC-16 within Header Packet remains unchanged.
- The Remote Rx Header Buffer Credit Count shall be incremented by one if a valid LCRD_x is received.
- The Remote Rx Header Buffer Credit Count shall be decremented by one if a non retried Header Packet is sent.
- The Remote Rx Header Buffer Credit Count shall not be changed when a Header Packet is retried.

7.1.4.1.4 Receiving Header Packets

A Header Packet is defined as “received properly” when it has all of the following:

1. Valid CRC-5
2. Valid CRC-16
3. Valid length
4. An Rx Header buffer is available to store the Header Packet, or the Header Packet has been processed.

- If a Header Packet has not been received properly, the following shall occur:
  1. If the Header Packet has one or more CRC-5 or CRC-16 errors, or length error, a Port shall send LBAD.
  2. If a Port does not have a Rx Header buffer available to store the Header Packet, a transition to Recovery shall be initiated.
- Receipt of a Header Packet shall consume one Rx Header Buffer.
- If a Header Packet has been received properly, the Header Sequence Number in the received Header Packet and the Rx Header Sequence number shall be compared.
  1. If the Header Sequence number in the received Header Packet and the Rx Header Sequence number are equal, a Port receiving the Header Packet shall increment the Rx Header Sequence number by one (or roll over to 0 if the maximum Header Sequence number is reached). The following two actions shall be taken.
     a. A Port shall transmit a single LGOOD_n with ”n” corresponding to the received Header Sequence number if a Header Packet is received properly. The Local Rx Header Buffer Credit Count shall be decremented by one.

b. A Port shall free an Rx Header Buffer after the received Header Packet within that buffer has been processed and a single LCRD_x is sent. A Port shall advance the Credit index orderly (and roll over to A if the Header Buffer Credit index of D is reached). The Local Rx Header Buffer Credit Count shall be incremented by one.

Note: The LCRD_x index is used to ensure Rx Header Buffer credits are returned in order.

2. If the Header Sequence Number in the received Header Packet and the Rx Header Sequence Number are not equal, a Port shall transition to Recovery.

- If the Header Packet has not been received properly, a Port shall ignore all Header Packets received between the corrupted Header Packet causing the LBAD and the LRTY.
- After transmitting LBAD, a Port shall not transmit any additional LBAD or LGOOD_n until it processes a Header Packet following an LRTY Link Command.
- After transmitting LBAD, a Port shall continue to send LCRD_x if an Rx Header Buffer Credit is made available.
- A Port shall transition directly to Recovery if it fails to receive the same Header packet three consecutive times. A Port shall not transmit the third LBAD upon the third error.

7.1.4.1.5 Receiving Data Packet Payload

The processing of Data Packet Payload shall meet the following rules.

- A Data Packet Payload shall be forwarded if the following two conditions are met.
  1. A Data Packet Header is received properly.
  2. A Data Packet Payload Starting Framing is received properly.
- The Data Packet Payload forwarding shall be terminated if one of the following conditions is met.
  1. A Data Packet Payload Ending Frame is detected and forwarded.
  2. Data Packet Payload of length exceeding “sDataSymbolsBabble” (see Table 10-18) has been reached and no Data Packet Payload Ending Frame is detected.
  3. A non Data Packet Payload Ending Frame is detected.

  Note: Both 2 and 3 are cases of error conditions.
- A Data Packet Payload shall be dropped if its Data Packet Header is corrupted.
- The CRC-32/length check of a Data Packet Payload is optional for a packet being forwarded by a hub. Checking the CRC-32/length of a Data Packet Payload of a packet to be consumed (that is, a packet which has reached its target device) is required.

7.1.4.1.6 Receiving LGOOD_n

- A Port shall maintain every Header Packet transmitted within its Tx Header Buffer until it receives an LGOOD_n with its Header Sequence Number matching its ACK TX Header Sequence Number.
  1. If the Header Sequence Number in LGOOD_n matches the ACK TX Header Sequence Number, a Port shall flush the Tx Header Buffer that stores that Header Packet and increment the ACK TX Header Sequence Number by one based on modulo-8 operation.
  2. If the Header Sequence Number in LGOOD_n does not match the ACK TX Header Sequence Number, a Port shall transition to Recovery. The ACK TX Header Sequence Number shall be unchanged.

     Note: A Port that has received an out of order LGOOD_n implies a loss or corrupted Link Command and must initiate transition to Recovery.

7.1.4.1.7 Receiving LCRD_x

- A Port shall adjust its Remote Rx Header Buffer Credit Count based on the received LCRD_x.
  1. A Port shall increment its Remote Rx Header Buffer Credit Count by one upon receipt of an LCRD_x.
  2. A Port shall transition to Recovery if it receives an out of order LCRD_x.

     Note: A Port that has received an out of order credit implies a loss or corrupted Link Command and must transition to Recovery.

7.1.4.1.8 Receiving LBAD

- Upon receipt of LBAD, a Port shall send a single LRTY before re-transmitting all the Header Packets in the Tx Header Buffers that have not been acknowledged with LGOOD_n. A Port shall set the Delayed bit in the Link Control Word on all resent Header Packets and recalculate CRC-5.

  Note: Resending an ITP invalidates the isochronous timestamp value. CRC-16 is unchanged in a retried Header Packet.
- Upon receipt of LBAD, a Port shall send a single LRTY if there is no unacknowledged Header Packet in the Tx Header Buffers.

  Note: This is an error condition where LBAD is created due to Link error.

7.1.4.1.9 Transmitter Timers

- A Port shall have a PENDING_HP timer that is active only in U0 and there are pending Header Packets; it shall be disabled and reset in all other states. The PENDING_HP timer shall be active when a Header Packet is pending or transmitted. The PENDING_HP timer shall be reset when an LGOOD_n or a LBAD is received. The PENDING_HP timer shall be restarted when a Header Packet is unacknowledged with an LGOOD_n and there are still pending Header Packets. The timeout value for the PENDING_HP timer is listed in Table 7-7.
- A Port shall initiate a transition to the Recovery state if the following two conditions are met.
  1. PENDING_HP timer times out.
  2. The transmission of an outgoing HP is completed or a DPP is terminated.

  Note: This is to allow a graceful transition to Recovery without a HP being truncated.
- A Port shall have a CREDIT_HP timer that is active in only U0 and the transmitter's Credit Count is less than 4; it shall be disabled and reset in all other states. The CREDIT_HP timer shall be started when a Header Packet or a retried Header Packet is sent. The CREDIT_HP timer shall be reset when a valid LCRD_x is received. The CREDIT_HP timer shall be re-started if the transmitter's Credit Count is less than 4. The timeout value for the CREDIT_HP timer is listed in Table 7-7.
- A Port shall initiate a transition to the Recovery state if the following two conditions are met.
  1. CREDIT _HP timer times out.
  2. The transmission of an outgoing HP is completed or a DPP is terminated.

Note: This is to allow a graceful transition to Recovery without a HP being truncated.

Table 7-7. Transmitter Timers Summary

Timers	Timeout Value (us)	Tolerance
PENDING_HP timer	3	0 ~ +50%
CREDIT_HP timer	5000	

7.1.4.2 Link Power Management and Flow

Requests to transition to low power Link states are done at the Link Level during U0. Link Commands LGO_U1, LGO_U2 and LGO_U3 are sent by a Port as a request to enter a low power Link state. LAU, LXU are sent by the other Port as the response. LPMA is sent by a Port in response only to LAU. Details on exit/wake from a power management state are described in Sections 7.4.7, 7.4.8, and 7.4.9.

7.1.4.2.1 Power Management Link Timers

A Link Command PM timer (PM_LC_TIMER) on a Port requesting to enter a low power Link state is required to ensure that the process of Low Power Link State negotiation is done in a prompt manner.

- The PM_LC_TIMER timer shall be started after the last Symbol of an LGO_Ux Link Command is sent.
- The PM_LC_TIMER timer shall be disabled and reset upon receipt of the LAU or LXU Symbol at the receiving Port of the requestor. The timeout value for the PM_LC_TIMER is listed in Table 7-8. Upon a PM_LC_TIMER timeout, a Port shall initiate Recovery.

A PM Entry timer (PM_Entry_TIMER) is required on a Port receiving LGO_Ux. It is required to ensure that both sides of the link are in the same PM state regardless if the LAU or LPMA is lost or corrupted. The timeout value for the PM_Entry_TIMER is listed in Table 7-8.

- The PM_Entry_TIMER timer shall be started after the last Symbol of an LAU is sent.
- The PM_Entry_TIMER timer shall be disabled and reset upon receipt of the LPMA, or a TS1 Ordered Set, or a LFPS meeting Ux exit signaling specified in Section 6.9.2.

Table 7-8. Link Flow Control Timers Summary

Timers	Timeout Value (us)	Tolerance
PM_LC_TIMER	3	0 ~ +50%
PM_Entry_TIMER	4	
Ux_Exit	6000	

7.1.4.2.2 Low Power Link State Initiation

- A Port shall not send a LGO_U1, LGO_U2 or LGO_U3 unless it meets all of the following:
  1. It transmitted an LGOOD_n, LCRD_x sequence for all packets received.
  2. It received an LGOOD_n, LCRD_x sequence for all packets transmitted.
  3. It has no pending packets for transmission.

4. It is directed by a higher layer to initiate entry. Examples of when a higher layer may direct the Link layer to initiate entry: (a) The U1 or U2 inactivity timeout expires (see PORT_U1_TIMEOUT, PORT_U2_TIMEOUT in the Hub chapter); (b) Reception of a SetPortFeature(PORT_LINK_STATE) request; (c) Device implementation specific mechanisms.
5. It has met higher layer conditions for initiating entry. Examples are (a) U1_enable/U2_enable is set or U1_TIMEOUT/U2_TIMEOUT is not equal zero; (b) Device has received an ACK TP for each and every previously transmitted packet; and (c) Device is not waiting for a TP following a PING. For these and any other examples, see the protocol chapter.

Note: This implies all credits must be received and returned before a Port can initiate a transition to a power management state.

- A Port shall do one of the following in response to receiving an LGO_U1 or LGO_U2:
  1. A Port shall send an LAU if the Force Link PM Accept field is asserted due to having received a Set Link Functionality LMP.
  2. A Port shall send an LAU if all of the following conditions are met.
     a. It has transmitted an LGOOD_n, LCRD_x sequence for all packets received.
     b. It has received an LGOOD_n, LCRD_x sequence for all packets transmitted.
     c. It has no pending packets for transmission.
     d. It is not directed by a higher layer to reject entry. Examples of when a higher layer may direct the Link layer to reject entry: (1) U1 or U2 is not enabled (e.g. PORT_U1_TIMEOUT or PORT_U2_TIMEOUT set to zero); (2) When a device has not received an ACK Transaction Packet (TP) for a previously transmitted packet (see the Protocol chapter); and (3) When a device receives a ping TP (see the ping packet definition in the Protocol chapter for more information).
  3. A Port shall send an LXU if any of the above conditions are not met.

  Note: This implies all credits must be received and returned before a Port can transition to a low power Link state.

7.1.4.2.3 U1/U2 Entry Flow

Either a Downstream Port or an Upstream Port may initiate U1/U2 entry or exit. Entry to a Low Power U1 or U2 Link state is accomplished by using Link Commands defined in Table 7-5.

- A Port shall send a single LGO_U1, LGO_U2 Link Command to request a transition to a low power Link state.
- Upon issuing LGO_Ux, a Port shall start PM_LC_TIMER.
- A Port shall respond to LGO_Ux with a single LAU Link Command and remain in U0 if it accepts LGO_Ux.
- A Port shall reject LGO_U1 or LGO_U2 with a single LXU Link Command and continue its Packet processing.
- Upon sending LGO_U1 or LGO_U2, a Port shall continue the process of receiving Packets until LAU is received.
- Upon receiving LXU, a Port shall remain in U0.
- A Port in U1 shall enter U2 directly if the following two conditions are met.
  1. The Port's U2 inactivity timer is enabled
  2. The U2 inactivity timer times out and no U1 LFPS exit signal is received

- A Port shall initiate transition to Recovery if a single LAU or LXU Link command is not received upon PM_LC_TIMER timeout.
- Upon issuing LAU, a Port shall not send any packets or Link commands.
- Upon issuing LAU or LPMA, a Port shall start PM_ENTRY_TIMER.
- Upon receiving LAU, a Port shall send a single LPMA Link Command and then enter the requested low power Link state.
- A Port that sends LAU shall enter the corresponding low power Link state upon receipt of LPMA before PM_ENTRY_TIMER timeout.
- A Port that sends LAU shall enter the requested low power Link state upon PM_ENTRY_TIMER timeout and if all of the following conditions are met.
  1. LPMA is not received
  2. No TS1 Ordered Set is received
  3. No LFPS Ux_Exit signal is received

  Note: this implies LPMA is corrupted and the Port issuing LGO_Ux has entered Ux.
- A Port that has sent LAU shall enter Recovery before PM_ENTRY_TIMER timeout if aTS1 Ordered Set is received.

  Note: this implies LAU was corrupted and the Port issuing LGO_Ux has entered Recovery.
- A Port that has sent LAU shall respond with Ux LFPS exit handshake defined in Section 6.9.2 before PM_ENTRY_TIMER timeout and if LFPS Ux_Exit signal is received.

  Note: this implies LPMA was corrupted and the Port issuing LGO_Ux has initiated Ux exit.

7.1.4.2.4 U3 Entry Flow

Only a Downstream Port can initiate U3 entry. An Upstream Port can not reject U3 entry.

- Upon Directed, a Downstream Port shall initiate U3 entry process by sending LGO_U3.
- Upon issuing LGO_U3, a Downstream Port shall start PM_LC_TIMER.
- An Upstream Port shall send LAU Link Command in response to a LGO_U3 request by a Downstream Port.
- An Upstream Port shall not send any packets or Link commands subsequent to sending an LAU.
- Upon issuing LGO_U3, a Downstream Port shall ignore any Packets sent by an Upstream Port.

  Note: This is a corner condition that an Upstream Port is sending a Header Packet before receiving LGO_U3.
- Upon Receiving LGO_U3, an Upstream Port shall respond with an LAU. The processing of all the unacknowledged Packets shall be aborted.
- Upon issuing LAU, an Upstream Port shall start PM_ENTRY_TIMER.
- A Downstream Port shall send a single LPMA and then transition to U3 if LAU is received.
- A Downstream Port shall transition to Recovery and re-initiate U3 entry after recover if all of the following three conditions is met.
  1. The PM_LC_TIMER times out.
  2. LAU is not received.
  3. The number of consecutive U3 entry attempts is less than three.

- An Upstream Port shall transition U3 if one of the following two conditions is met.
  1. LPMA is received
  2. The PM_ENTRY_TIMER times out and LPMA is not received

7.1.4.2.5 Concurrent Low Power Link Management Flow

Concurrent Low Power Link Management flow applies to situations where a Downstream Port and an Upstream Port both issue a request to enter a Low Power Link State.

- If a Downstream Port has sent an LGO_U1, LGO_U2, or LGO_U3 and also received an LGO_U1 or LGO_U2, it shall send an LXU.
- If an Upstream Port has sent an LGO_U1 or LGO_U2 and also received an LGO_U1, LGO_U2, it shall wait until reception of an LXU and then send either an LAU or LXU.
- If an Upstream Port has sent an LGO_U1 or LGO_U2 and also received an LGO_U3 from a Downstream Port, it shall wait until the reception of a LXU and then send an LAU.
- If a Downstream Port is directed by a higher layer to initiate a transition to U3, and a transition to U1 or U2 has been initiated but not yet completed, the Port shall first complete the in process transition to U1 or U2, then transition to U0 and initiate a transition to U3.

7.1.4.2.6 Concurrent Low Power Link Management and Recovery Flow

Concurrent Low Power Link Management and Recovery flow applies to situations where a Port issues a Low Power Link state entry and another Port issues Recovery. The Port that issues the Low Power Link state entry shall meet the following rules.

- Upon issuing LGO_Ux, a Port shall transition to Recovery if a TS1 Ordered Set is received.
- A Port shall re-initiate Low Power Link state entry process described in Sections 7.1.4.2.3 and 7.1.4.2.4 upon re-entry to U0 from Recovery.

7.1.4.2.7 Low Power Link State Exit Flow

Exit from a Low Power Link State refers to exit from U1 or U2, or U3 wakeup. It is accomplished by LFPS exit signaling defined in Section 6.9.2. A successful LFPS handshake process will lead both a Downstream Port and an Upstream Port to Recovery.

An Ux_Exit Timer is defined to cover the period of time from when a Port starts exit from U1 or U2 by sending LFPS Exit handshake signal, to when the Link is in U0. The timeout value is defined in Table 7-8.

- Ux_Exit Timer shall only apply to U1 or U2 exit. It shall not apply to U3 wakeup.

The Exit from a U1 or U2 shall meet the following flow. The U3 wakeup follows the same flow with the exception that no Ux_exit Timer is applied.

- An Ux_Exit timer shall be started and Ux_Exit/U3 wakeup LFPS signal defined in Section 6.9.2 shall be sent upon a Port initiating Ux Exit/U3 Wakeup.
- A Port that has received Ux_Exit/U3 wakeup LFPS signal shall start Ux exit by responding with Ux_Exit/U3 wakeup LFPS signal defined in Section 6.9.2.
- Upon a successful LFPS handshake before “No-LFPS-Response-Timeout” defined in Table 6-14, a Port shall transition to Recovery.
- A Port shall transition to SS.Inactive if one of the following two conditions is met,
  1. Upon “No-LFPS-Response-Timeout” and the condition of a successful LFPS handshake is not met,
  2. Upon Ux_Exit timer timeout, the Link has not transitioned to U0,

7.2 Link Error Rules/Recovery

7.2.1 Overview of SuperSpeed Bit Errors

The SuperSpeed timing budget is based on the statistical bit error probability of a Link being less than 1e-12. Header Packets and Link Commands are designed to be tolerant to a single bit error without retraining the Link. Details on errors detected with Link Level Flow Control are described in Section 7.1.4.

7.2.2 Link Error Types, Detection and Recovery

The LFSR is incremented as normal for all errors that do not create a COM Symbol. See section (PHY reference) for details on the LFSR response to COM and SKP Symbols. Beyond that, the Link layer will detect 3 categories of errors: Packet, Link Command and Training Sequence. Logical Idle Symbols not equal to 00 have no consequences and shall be ignored.

7.2.3 Packet Errors

Several types of packet errors must be detected. They are:

1. Missing of a Header Packet
2. Invalid Header Packet due to CRC and/or length errors
3. Mismatch of a Header Sequence number

Regardless, the Link Error Count is incremented for only one type of the error in the Link Layer, and that is an error which will cause the Link to transition to Recovery.

Note: not all detected errors will increment the error counter.

7.2.3.1 Packet Framing Detection

Packet Framing Ordered Set is constructed such that any single K-symbol corruption within the Ordered Set will not corrupt its Packet Framing recognition.

Header Packet framing and Data Packet Payload framing are all constructed using 4 special K-symbol Ordered Set. A Header Packet Framing contains only one Packet Framing Ordered Set at the beginning of the packet defined in 7.1.1. A Data Packet Payload contains one Start Packet Framing Ordered Set at the beginning of the packet and one End Packet Framing Ordered Set at the end of the Packet defined in 7.1.2.

Packet Framing Detection:

- A Valid Packet Framing shall be declared if the following two conditions are met.
  1. At least three of the four K-symbols in the four consecutive symbol periods are valid Packet Framing K-symbols.
  2. The four Symbols are in an order defined in Table 7-9.
- If a Packet Framing has two or more K-symbols corrupted, a Packet will not be detectable and therefore result in missing of a Packet.

Note: Missing of a Header Packet can not be detected directly.

Table 7-9. Valid Packet Framing K-Symbol Order (K = SHP, SDP, END, EDB)

Symbol 0	Symbol 1	Symbol 2	Symbol 3	Comment
K	K	K	EPF	all K-symbols are valid
Corrupt	K	K	EPF	First K corrupted
K	Corrupt	K	EPF	Second K corrupted
K	K	Corrupt	EPF	Third K corrupted
K	K	K	Corrupt	EPF corrupted

- Any appearance of single or double occurrences of K Symbols within a Packet shall be ignored.

Error Path Recovery:

- Missing of a Header Packet will result in the Link to transition to Recovery depending on which one of the following two conditions becomes true first.
  1. A Port transmitting the Header Packet upon its PENDING_HP_Timer timeout
  2. A Port receiving the Header Packet upon detection of a Header Sequence Number error

Error Counting:

- The error count shall be incremented by one each time a transition to Recovery occurs.

7.2.3.2 Header Packet CRC or Length Error

Error Detection:

- A Header Packet CRC Error shall be declared if the following conditions are met.
  1. A valid Header Packet framing is detected
  2. No valid Packet Framing is detected within a Header Packet.
  3. Either CRC-5 or CRC-16 check fails as defined in section 7.1.1
- A Header Packet Length Error shall be declared if the following two conditions are met.
  1. A valid Header Packet framing is detected
  2. A valid Packet Framing is detected within a Header Packet as defined in section 7.1.1

Error Path Recovery:

- A Port receiving the Header Packet shall send an LBAD as defined in Section 7.1.4.1 if it detects a Header Packet CRC or length error.

Error Counting:

- The error count shall remain unchanged.

7.2.3.3 Rx Header Sequence Number Error

Detection:

- An Rx Header Sequence Number Error shall occur if the following conditions are met.
  1. A Valid Header Packet framing is detected to allow the packet to be parsed
  2. All CRC's must be valid and the packet must be the correct length
  3. The Header Sequence Number in the received Header Packet does not match the Rx Header Sequence Number.

Error Path Recovery:

- A Port detecting an Rx Header Packet Sequence Number Error shall transition to Recovery.

Error Counting:

- The error count shall be incremented by one each time a transition to Recovery occurs.

7.2.3.4 ACK Tx Header Sequence Number Error

Detection:

- A Tx Header Sequence Number Error shall occur if the following conditions are met.
  1. A Valid LGOOD_n is received
  2. The Header Sequence Number in the received LGOOD_n does not match the ACK Tx Header Sequence Number.

Error Path Recovery:

- A Port detecting a ACK Tx Header Sequence Number Error shall transition to Recovery.

Error Counting:

- The Link Error Count shall be incremented by one each time a transition to Recovery occurs.

7.2.3.5 Header Sequence Number Advertisement Error

Detection:

- A Header Sequence Number Advertisement Error shall occur if one of the following conditions is true.
  1. Upon PENDING_HP_Timer timeout and LGOOD_n not received
  2. A Header Packet received before receiving LGOOD_n
  3. Any Link Commands except LGOOD_n received before LGOOD_n

Error Path Recovery:

- A Port detecting any Header Sequence Number Advertisement Error shall transition to Recovery.

Error Counting:

- The Link Error Count shall be incremented by one each time a transition to Recovery occurs.

7.2.4 Training Sequence Error

Symbol corruptions during the TS1 and TS2 Ordered Sets in Polling.Active, Polling.Configuration, Recovery.Active, and Recovery.Configuration sub-states are expected until the requirements are met to transition to the next state. A timeout from any one of these sub-states is considered Training Sequence Error.

Detection:

- A Timeout from either Polling.Active, or Polling.Configuration, or Recovery.Active, or Recovery.Configuration sub-state to the SS.Inactive shall result in a Training Sequence error.

Error Path Recovery:

- An Error in Training Sequence shall result in a transition to the SS.Inactive state.
- A Port shall transition SS.Inactive if a Training Sequence Error occurs during Recovery.
- A Downstream Port shall transition to Rx.Detect if a Training Sequence Error occurs during Polling.

- An Upstream Port of a Hub shall transition to Rx.Detect if a Training Sequence Error occurs during Polling.
- An Upstream Port of a peripheral Device shall transition to SS.Disabled if a Training Sequence Error occurs during Polling.

Error Counting:

- The Link Error Count shall remain unchanged.

7.2.5 Link Command Detection and Declarations

A Link Command is constructed such that any single K-symbol corruption will not invalidate the recognition of a Link Command, and any single-bit error in the two scrambled Link Command Words will not corrupt the correct parsing of a Link Command.

Detection and Parsing:

- A detection of a Link Command shall be declared if the following two conditions are met.
  1. At least three of the four K-symbols in four consecutive symbol periods are valid Link Command K-symbols.
  2. The four symbols are in the order described in Table 7-10.
- A valid Link Command is declared upon detection of a Link Command and one of the following two conditions is met.
  1. Both Link Command Words are the same. They both contain valid Link Command information as defined in Table 7-4. They both pass the CRC-5 check.
  2. The two Link Command Words are not the same, with one Link Command Word containing valid Link Command information as defined in Table 7-4, passing the CRC-5 check, and the other Link Command Word either failing CRC-5 check, or not containing valid Link Command information as defined in Table 7-4.
- An invalid Link Command is declared upon detection of a Link Command and the conditions to meet a Valid Link Command are not met.

Error Path Recovery:

- An invalid Link Command shall result in a receiving Port transitioning to Recovery.

  Note: The missing of a Link Command is not detected by a receiving Port. A transmitting Port will detect the missing of a Link Command and issue the transition to Recovery.

Table 7-10. Valid Link Command K-Symbol Order

Symbol 0	Symbol 1	Symbol 2	Symbol 3	Comment
SLC	SLC	SLC	EPF	all K-symbols are valid
Corrupt	SLC	SLC	EPF	First SLC corrupted
SLC	Corrupt	SLC	EPF	Second SLC corrupted
SLC	SLC	Corrupt	EPF	Third SLC corrupted
SLC	SLC	SLC	Corrupt	EPF corrupted

7.2.6 Summary of Error Types and Recovery

Table 7-11 summarizes the Link error types, error count, and different error paths to restore the Link.

- The Link error shall be counted each time a Link transitions to Recovery.
- The Link error count shall be reset upon a PowerOn Reset, Warm Reset, or a Hot Reset.

Table 7-11. Error Types and Recovery

Error Type	Description/Example	Error Recovery Path	Update Link Error Count?
Missing Packet Framing	Only a Valid Packet Framing Ordered Set will be declared in the receiver side	Delayed transition to Recovery	Yes
Header Packet CRC or Length Error	Any Header packet CRC is bad or the packet length is not correct	Header packet retry process	No.
Header Sequence Number Error	Header Sequence number is not expected by receiver	Recovery	Yes
Training Sequence Error	Timeout from Polling or Recovery to SS.Inactive without reaching U0	Timeout to SS.Inactive. Software required to exit SS.Inactive	No
Invalid Link Command	Valid Link command framing but invalid Link Command Word	Recovery.	Yes
Missing Link command	No Valid Link command framing is detected	Delayed transition to Recovery	Yes

7.3 PowerOn Reset and Inband Reset

There are two categories of reset associated with a Link. The first, PowerOnReset, is associated with the application of a valid Vbus specified in Section 11.4.2. Upon PowerOnReset, the LTSSM (described in Section 7.4) will bring a Port to its initial state of Rx.Detect. The second, Inband Reset relies on SuperSpeed and/or LFPS signaling to propagate the reset across the Link. There are two mechanisms to complete an Inband Reset; Hot Reset and Warm Reset. Upon completion of either a PowerOn Reset or an Inband Reset, the Link shall transition to U0 as described in Section 7.3.2.

A generic Reset mentioned in other chapters refers to an Inband Reset unless otherwise specified.

7.3.1 PowerOn Reset

PowerOn Reset is initiated by the application of VBUS. PowerOn Reset applies only when VBUS is present. A Port must be responsible for its own internal Reset signaling and timing.

The following shall occur when PowerOnReset is asserted or while VBUS is off:

1. Receiver terminations shall meet specifications defined in Table 6-12.
2. Transmitters shall hold a constant DC common mode voltage ($V_{TX\text{-}DC\text{-}CM}$) defined in Table 6-10.
3. The LTSSM and the PHY level variables (such as Rx equalization settings) shall be reset to their default values.

4. A Port shall be in one of the following states.
   a. An Upstream Port of a peripheral Device shall be in “Attached” state defined in Section 9.1.1.1.
   b. An Upstream Port of a Hub shall be in “USPORT.Power-off” state defined in Section 10.5.1.1.
   c. A Downstream Port shall be in “DSPORT. Power-off” state defined in Section 10.3.1.1.

The following shall occur when PowerOn Reset is completed:

1. The LTSSM of a Port shall be initialized to Rx.Detect.
2. A Port shall be in one of the following states.
   a. An Upstream Port of a peripheral Device shall be in “Powered” state defined in Section 9.1.1.2.
   b. An Upstream Port of a Hub shall be in “USPORT.Power On” state defined in Section 10.5.1.2.
   c. A Downstream Port shall be in “DSPORT. Disconnected” state defined in Section 10.3.1.2.
3. The LTSSM and the PHY level variables (such as Rx equalization settings) shall be reset to their default values.
4. The receiver impedance termination of a Port shall meet the $R_{RX\text{-}DC}$ specifications defined in Table 6-12..

Note: Rx termination will always be maintained throughout operation except for SS.disabled

7.3.2 Inband Reset

An Inband Reset shall be generated by a Downstream Port only when it is Directed.

There are two mechanisms to generate an Inband Reset. The first mechanism; Hot Reset, is defined by sending TS2 Ordered Sets with the Reset bit asserted. A Hot Reset shall cause the LTSSM to transition to the Hot Reset state. Upon completion of Hot Reset, the following shall occur.

- A Downstream Port shall reset its Link Error Count and return to “DSPort.Enabled” described in Section 10.3.1.5.
- An Upstream Port shall return to “Default” described in Section 9.1.1.3 of the Framework Chapter.
- The Port Configuration information of an Upstream Port shall remain unchanged. Refer to Section 8.4.5.
- The LTSSM of a Port shall transition to U0.

The second mechanism of an Inband Reset is Warm Reset. The signaling of a Warm Reset is defined as a Low Frequency Periodic Signal meeting the T_{reset} requirements (see Chapter 6). A Warm Reset will cause the LTSSM to transition to Rx.Detect, re-train the Link including the Receiver Equalizer, reset an Upstream Port, and then transition to U0. A completion of a Warm Reset shall result in the following.

- A Downstream Port shall reset its Link Error Count and return to “DSPort.Enabled” described in Section 10.3.1.5 of the Hub Chapter.
- An Upstream Port shall return to “Default” described in Section 9.1.1.3 of the Framework Chapter.
- Port Configuration information of an Upstream Port shall be reset to default values. Refer to Section 8.4.5.

- The LTSSM of a Port shall transition to U0.

If a Reset is directed with Warm Reset not asserted, a Downstream Port shall issue the Inband Reset based on following conditions.

- If the Downstream Port is SS.Inactive, it shall use a Warm Reset.
- If the Downstream Port is in U0, it shall use Hot Reset.
- If the Downstream Port is in U1, U2, or U3, it shall exit U1, U2 or U3 using the appropriate LFPS exit handshake, transition to Recovery and then transition to Hot Reset.
- If a Downstream Port is in a transitory state of Polling or Recovery, it shall use Hot Reset.
- If a Hot Reset fails due to a LFPS handshake timeout, a Downstream Port shall transition to SS.Inactive until software intervention or upon detection of removal of an Upstream Port.
- If a HotReset fails due to a TS1/TS2 handshake timeout, a Downstream Port shall transition to SS.Inactive and attempt a Warm Reset.
- If the Downstream Port is in SS.Disabled, an Inband Reset is prohibited.

If a Reset is directed with Warm Reset asserted, the following shall occur.

- A Downstream Port shall initiate a Warm Reset in all the Link states except SS.Disabled and then transition to Rx.Detect.
- An Upstream Port shall enable its LFPS receiver and Warm Reset detector in all the Link states except SS.Disabled.
- An Upstream Port receiving a LFPS signal that meets the T_{reset} requirements shall transition to the Rx.Detect.

7.4 Link Training and Status State Machine (LTSSM)

Configuration information and requests to initiate State Machine transitions are controlled by software. All LTSSM references to “directed” refers to upper layer mechanisms fully described in Chapter 10.

All timeout values specified in the Link Training and Status State Machine (LTSSM) timeout values are minus 0 seconds and plus 50% unless explicitly stated otherwise . All timeout values must be set to the specified values after power-on/reset. All counter values must be set to the specified values after power-on/reset.

Table 7-12. Nominal LTSSM Timeouts

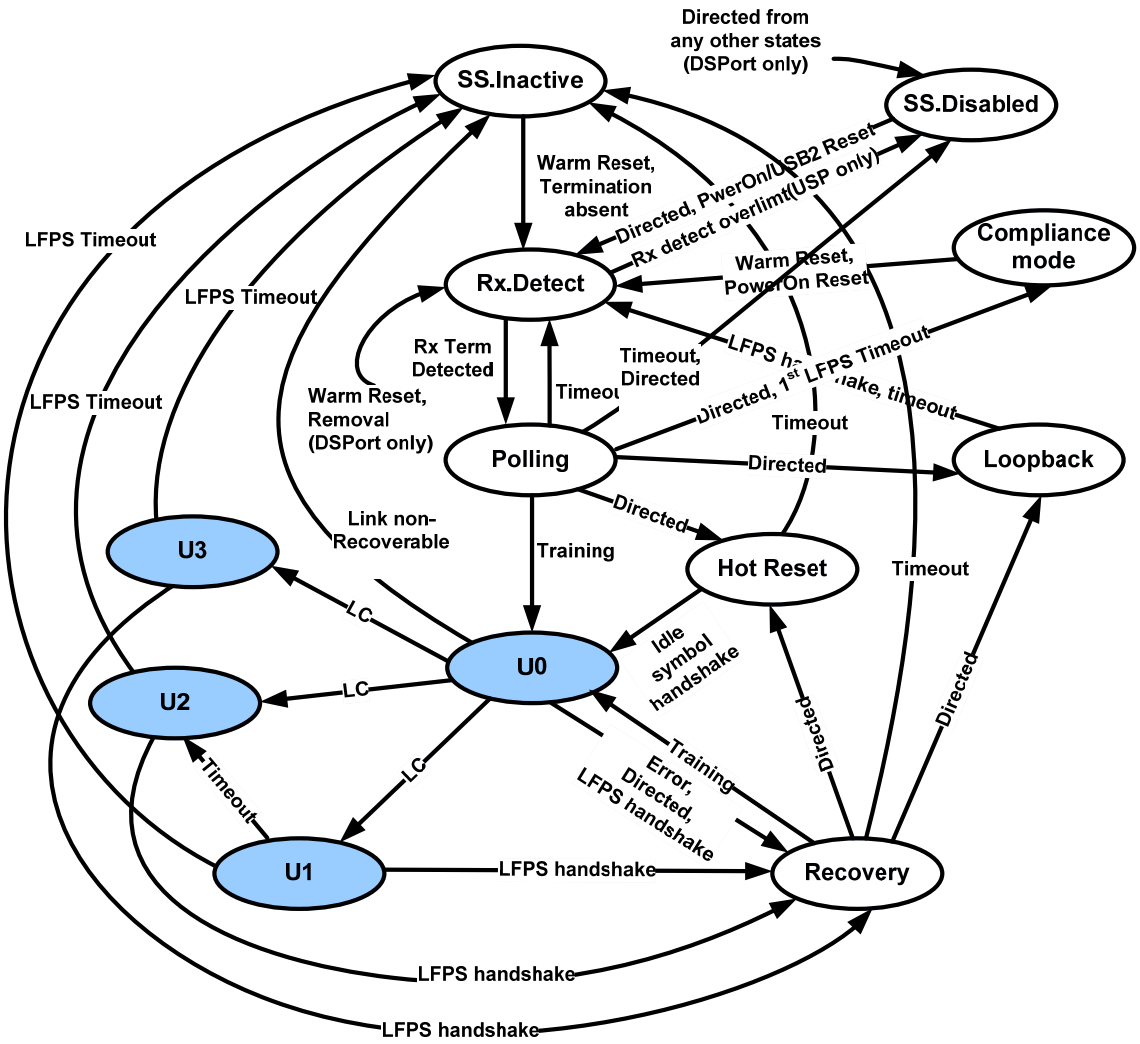
Initial State	Timeout to State	Timeout Value	Tolerance
Rx.Detect.Quiet	Rx.Detect.Active	12 ms	0 ~ +50%
Polling.LFPS **	Compliance	360 ms	
Polling.LFPS **	SS.Inactive	24 ms	
Polling.Active	SS.Inactive	12 ms	
Polling.Configuration	SS.Inactive	12 ms	
U1	SS.Inactive	2 ms	
U1	U2	U1 Inactivity field set in LMP (see protocol chapter)	

Initial State	Timeout to State	Timeout Value	Tolerance
Rx.Detect.Quiet	Rx.Detect.Active	12 ms	
Polling.LFPS **	Compliance	360 ms	
Polling.LFPS **	SS.Inactive	24 ms	
Polling.Active	SS.Inactive	12 ms	
Polling.Configuration	SS.Inactive	12 ms	
U1	SS.Inactive	2 ms	
U2	SS.Inactive	2 ms	
U3	SS.Inactive	20 ms	
Hot Reset	SS.Inactive	2 ms	
Polling.Idle	SS.Inactive	2 ms	
Recovery.Idle	SS.Inactive	2 ms	
Recovery.Active	SS.Inactive	12 ms	
Recovery.Configuration	SS.Inactive	12 ms	

**Note: If Polling.LFPS has not successfully received any valid LFPS signals it shall timeout to Compliance. If valid LFPS signals have been received, the timeout shall go to SS.Inactive. See Section 7.4.4.1 for details.

The state machine rules for configuring and operating a SuperSpeed Link are defined in the following sections.

All state machines diagrams have descriptions for transition conditions. These descriptions are for informational use only.



Note: Transition conditions are illustrative only.

Figure 7-11. State Diagram of the Link Training and Status State Machine

7.4.1 SS.Disabled

SS.Disabled is a state with a Port's low-impedance termination removed. It is a state where a Port's SuperSpeed connectivity is disabled. Refer to Section 10.16 for details regarding the behavior of a Peripheral Device. Refer to Section TBD of the Hub Chapter for details regarding a Hub's Upstream Port and Downstream Port.

SS.Disabled is also a Logical power-off state for a self-powered Upstream Port.

SS.Disabled Substate Machines:

SS.Disabled does not contain any substate.

SS.Disabled Requirements:

- Vbus may be present during SS.Disabled.
- The Port's Receiver termination shall present high impedance to ground.
- The Port shall be disabled for transmitting and receiving LFPS and SuperSpeed signals.

Exit from SS.Disabled:

A Downstream Port shall transition to Rx.Detect if directed.

An Upstream Port shall transition to Rx.Detect only if a PowerOn Reset or a USB2 bus reset is applied.

7.4.2 SS.Inactive

SS.Inactive is a state where a Link has failed SuperSpeed Operation. A Downstream Port can only exit from this state if directed, or upon detection of an absence of a far-end receiver termination (R_{RX-DC}) specified in Table 6-12, or upon a Warm Reset. An Upstream Port can only exit to Rx.Detect upon a Warm Reset, or upon detection of an absence of a far-end receiver termination (R_{RX-DC}) specified in Table 6-12.

During SS.Inactive, a Port periodically performs the termination detection of a far-end receiver. The purpose of this detection is to determine if the entry to SS.Inactive is due to disconnect of a Port. If a disconnection is detected, a Port will return to Rx.Detect. If a disconnection is not detected, a Link will stay in SS.Inactive until software intervention.

SS.Inactive Substate Machines:

SS.Inactive contains the following substate machines shown in Figure 7-13.

- SS.Inactive.Disconnect.Detect
- SS.Inactive.Quiet

SS.Inactive Requirements:

- Vbus shall be present during SS.Inactive.
- The Port's Receiver shall meet the termination requirement (R_{RX-DC}) specified in Chapter 6.
- The Transmitter common mode is not required to be within specification during this state.

7.4.2.1 SS.Inactive.Quiet

SS.Inactive.Quiet Requirements:

- The function of termination detection shall be disabled.
- The 12 ms timer shall be started.

Exit from SS.Inactive.Quiet:

- A Downstream Port shall transition to SS.Inactive.Disconnect.Detect after a 12 ms timeout.
- A Downstream Port shall transition to SS.Disabled if directed.
- A Downstream Port shall transition to Rx.Detect if a Warm Reset is issued.
- An Upstream Port shall transition to Rx.Detect upon detection of a Warm Reset.

7.4.2.2 SS.Inactive.Disconnect.Detect

SS.Inactive.Disconnect.Detect Requirements:

- The Transmitter shall perform the termination detection described in Section 6.11.

Exit from SS.Disconnect.Detect.Active:

- A Port shall transition to Rx.Detect if a far-end low-impedance receiver termination (R_{RX-DC}) is detected meeting specification defined in Table 6-12.
- A Port shall transition to SS.Inactive.Quiet if a far-end low-impedance receiver termination (R_{RX-DC}) meeting specification defined in Table 6-12 is not detected.

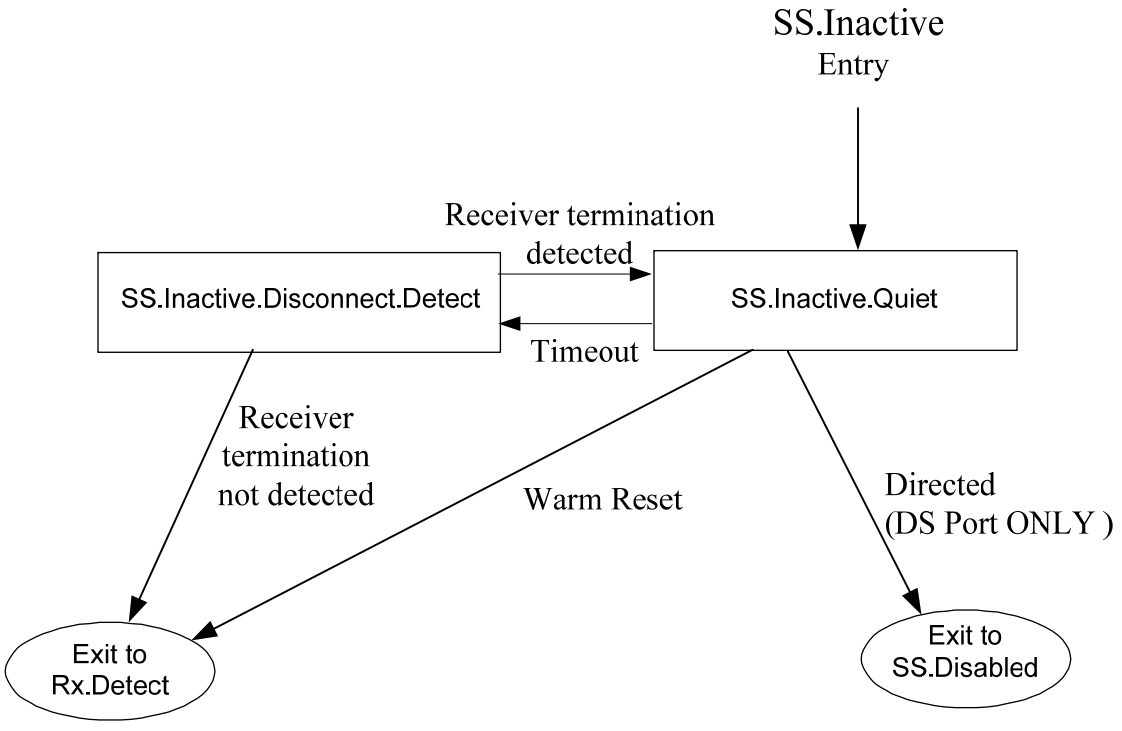


Figure 7-12. SS.Inactive Substate Machine

7.4.3 Rx.Detect

Rx.Detect is the initial power-on state of the LTSSM for both a Downstream Port and an Upstream Port. It is also the initial state for an Upstream Port upon detection of a Warm Reset from any other Link state except SS.Disabled. The purpose of Rx.Detect is to detect the presence of a far-end termination to ground. Rx.Detect.Reset is a default reset state used for the two Ports to synchronize to the Warm Reset. Rx.Detect.Active is a substate for termination detection. Rx.Detect.Quiet is a power saving substate in which the termination detection function is disabled.

Rx.Detect Substate Machines:

Rx.Detect contains the following substate machines shown in Figure 7-13.

- Rx.Detect.Reset
- Rx.Detect.Active
- Rx.Detect.Quiet

Rx.Detect Requirements:

- The Transmitter common mode is not required to be within specification during this state.
- The Rx termination defined in Table 6-12 shall be maintained.

7.4.3.1 Rx.Detect.Reset

Rx.Detect.Reset Requirements:

- An Upstream Port shall monitor the Warm Reset event.
- A Port shall complete the Warm Reset as defined in Section 6.9.3.

Note: a Warm Reset has duration of 100 ms. An Upstream Port needs to detect the occurrence of the Warm Reset and monitor it until its completion.

Exit from Rx.Detect.Reset:

- A Port shall transition to Rx.Detect.Active if a Warm Reset is not present.

Note: Warm Reset is not present during power-on.

- A Downstream Port shall transition to Rx.Detect.Active when it completes its Warm Reset.
- An Upstream Port shall transition to Rx.Detect.Active when it receives no more LFPS Warm Reset signals from the Downstream Port as defined in Section 6.9.3.

7.4.3.2 Rx.Detect.Active

Rx.Detect.Active Requirements:

- The Transmitter shall initiate a termination detection event. described in Section 6.11.
- The number of termination detection events shall be counted. One termination detection event is defined by a Transmitter enabling the function of termination detection.

Exit from Rx.Detect.Active:

- A Port shall transition to Polling upon detection of a far-end low-impedance receiver termination ($R_{RX\text{-}DC}$) defined in Table 6-12.
- A Downstream Port shall transition to Rx.Detect.Quiet if a far-end low-impedance receiver termination ($R_{RX\text{-}DC}$) defined in Table 6-12 is not detected.
- A Downstream Port shall transition to SS.Disabled if directed.

- An Upstream Port of a Hub shall transition to Rx.Detect.Quiet if a far-end low-impedance receiver termination (R_{RX-DC}) defined in Table 6-12 is not detected.
- An Upstream Port of a Peripheral Device shall transition to Rx.Detect.Quiet if the following two conditions are met.
  1. A far-end low-impedance receiver termination (R_{RX-DC}) defined in Table 6-12.
  2. The number of Rx termination Detect event is less than 8.
- An Upstream Port of a Peripheral Device shall transition to SS.Disabled if the following two conditions are met.
  1. A far-end low-impedance receiver termination (R_{RX-DC}) defined in Table 6-12 is not detected.
  2. The number of Rx termination Detect event has reached 8.

  Note: This event limit of the Rx termination detection is to allow a SuperSpeed Peripheral Device on a legacy platform to transition to USB2 after a reasonable amount of time.

7.4.3.3 Rx.Detect.Quiet

Rx.Detect.Quiet Requirements:

- The function of termination detection shall be disabled.

Exit from Rx.Detect.Quiet:

- A Port shall transition to Rx.Detect.Active after a 12 ms timeout.
- A Downstream Port shall transition to SS.Inactive if directed.

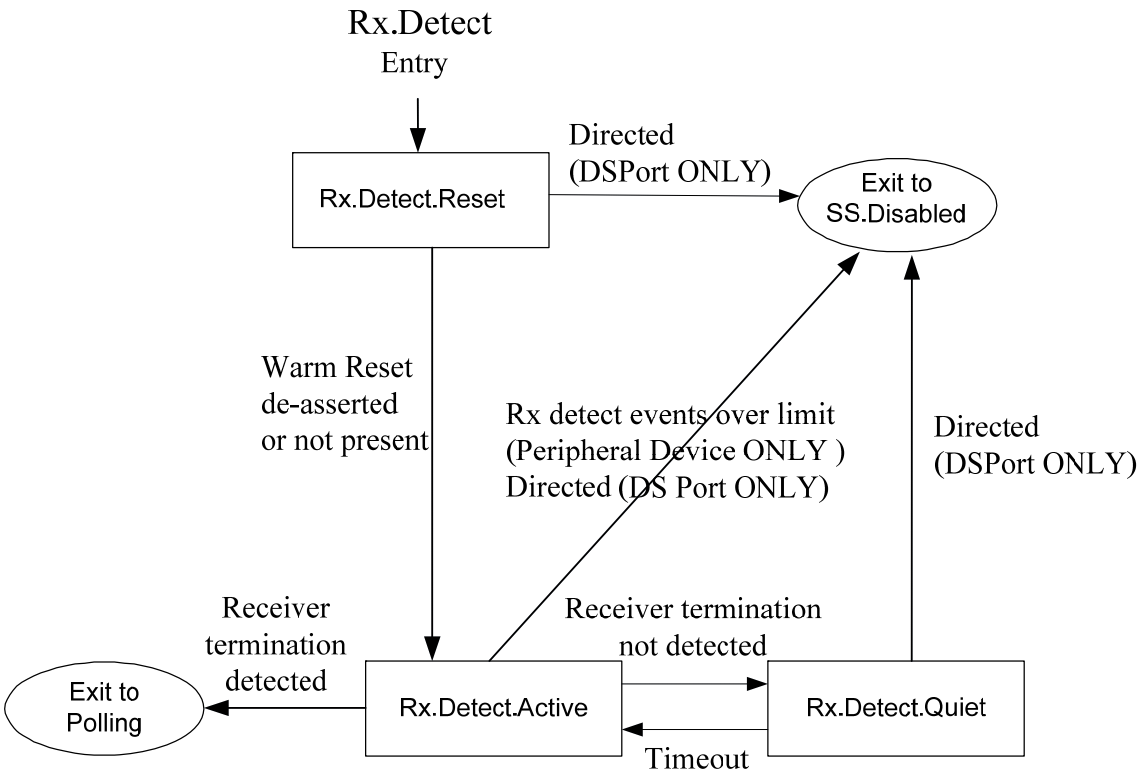


Figure 7-13. Rx.Detect Substate Machine

7.4.4 Polling

Polling is a state for Link training. During Polling, an LFPS handshake will be realized between the two Ports before the SuperSpeed training. Bit-Lock, Symbol-Lock, and Rx Equalization trainings are achieved through TSEQ, TS1, and TS2 training Ordered Sets.

Polling Substate Machines:

Polling contains the following substates shown in Figure 7-14.

- Polling.LFPS
- Polling.RxEQ
- Polling.Active
- Polling.Configuration
- Polling.Idle

Polling Requirements:

- Receiver Termination shall remain in low impedance.

7.4.4.1 Polling.LFPS

Polling.LFPS Requirements:

- Upon entry, a LFPS receiver shall be enabled to receive the Polling.LFPS signals defined in Section 6.9.1.
- Upon entry, a LFPS transmitter shall be enabled to transmit the Polling.LFPS signals.
- A 360 ms timer shall be started.
- A SuperSpeed receiver may optionally be enabled to receive TSEQ Ordered Sets for receiver equalizer training.

  Note: A Port first entering into Polling.RxEQ will start transmitting TSEQ Ordered Sets while another Port is still in Polling.LFPS. Disabling a SuperSpeed Receiver in Polling.LFPS may result in a slightly short period for Rx Equalizer training. A Port must complete its Receiver Equalizer training at Polling.RxEQ

Exit from Polling.LFPS:

- Polling.LFPS shall transition to Polling.RxEQ if the following three conditions are met:
  1. At least 8 consecutive Polling.LFPS bursts meeting the Polling.LFPS signal specification are sent.
  2. Two consecutive Polling.LFPS bursts are received.
  3. Four consecutive Polling.LFPS bursts are sent after receiving one Polling.LFPS burst.
- If it has never successfully completed Polling.LFPS after PowerOn Reset, a Port shall transition to Compliance upon the 360 ms timeout and the conditions to transition to Polling.RxEQ are not met.

  Note: If the very first attempt in Polling.LFPS handshake fails after PowerOn Reset, it implies that a passive test load maybe present and Compliance test should be initiated. If the very first attempt in Polling.LFPS handshake succeeds after PowerOn Reset, it implies the presence of two SuperSpeed Ports across the Link and no Compliance test is intended. Therefore, any subsequent handshake timeout in Polling.LFPS when the Link is re-trained is only a indication of Link training failure, not a signal to enter Compliance mode.
- A Downstream Port shall transition to Rx.Detect upon the 360 ms timeout after having trained once since PowerOn Reset and the conditions to transition to Polling.RxEQ are not met.

- An Upstream Port of a Hub shall transition to Rx.Detect upon the 360 ms timeout after having trained once since PowerOn Reset and the conditions to transition to Polling.RxEQ are not met.
- An Upstream Port of a Peripheral Device shall transition to SS.Disabled upon the 360 ms timeout after having trained once since PowerOn Reset and the conditions to transition to Polling.RxEQ are not met.

7.4.4.2 Polling.RxEQ

Polling.RxEQ is a substate for Rx equalizer training.

Polling.RxEQ Requirements:

- The detection and correction of the lane polarity inversion shall be enabled, as is described in Section 6.4.2.
- The Transmitter shall send the TSEQ Ordered Set.
- The Receiver shall complete Rx Equalizer training within this state.

Exit from Polling.RxEQ:

- A Port shall transition to Polling.Active after 65536 consecutive TSEQ Ordered Sets defined in Table 6-2 are transmitted.
- A Downstream Port shall transition to SS.Disabled if Directed.
- A Downstream Port shall transition to Rx.Detect if a Warm Reset is Directed.
- An Upstream Port shall transition to Rx.Detect if a Warm Reset is detected.

7.4.4.3 Polling.Active

Polling.Active is a substate to start the Link's SuperSpeed training after Receiver equalization.

Polling.Active Requirements:

- A Port transmits TS1 Ordered Sets.
- A Port receives TS1 or TS2 Ordered Sets.
- A 12 ms timer shall be started.

Exit from Polling. Active:

- A Port shall transition to Polling.Configuration upon receiving eight consecutive and identical TS1 or TS2 Ordered Sets.
- A Downstream Port shall transition to Rx.Detect upon 12 ms timeout and the conditions to transition to Polling. Configuration are not met.
- An Upstream Port of a Hub shall transition to Rx.Detect 12 ms timeout and the conditions to transition to Polling.Configuration are not met.
- An Upstream Port of a Peripheral Device shall transition to SS.Disabled upon 12 ms timeout and the conditions to transition to Polling.Configuration are not met.
- A Downstream Port shall transition to SS.Disabled if directed.
- A Downstream Port shall transition to Rx.Detect if a Warm Reset is Directed.
- An Upstream Port shall transition to Rx.Detect if a Warm Reset is detected.

7.4.4.4 Polling.Configuration

Polling.Configuration Requirements:

- A Port transmits TS2 Ordered Sets.
- A Port receives TS2 Ordered Sets.
- If directed, a Downstream Port shall set Reset bit in the TS2 Ordered Set.
- If directed, a Port shall set Loopback bit in the TS2 Ordered Set.

Exit from Polling.Configuration:

- A Port shall transition to Polling.Idle after the following two conditions are met:
  1. Eight consecutive and identical TS2 Ordered Sets are received.
  2. 16 TS2 Ordered Sets are sent after receiving the first of the eight consecutive and identical TS2 Ordered Sets.
- A Downstream Port shall transition to Rx.Detect upon 12 ms timeout and the conditions to transition to Polling.Idle are not met.
- An Upstream Port of a Hub shall transition to Rx.Detect upon 12 ms timeout and the conditions to transition to Polling. Idle are not met.
- An Upstream Port of a Peripheral Device shall transition to SS.Disabled upon 12 ms timeout and the conditions to transition to Polling. Idle are not met.
- A Downstream Port shall transition to SS.Disabled if directed.
- A Downstream Port shall transition to Rx.Detect if a Warm Reset is Directed.
- An Upstream Port shall transition to Rx.Detect if a Warm Reset is detected.

7.4.4.5 Polling.Idle

Polling.Idle Requirements:

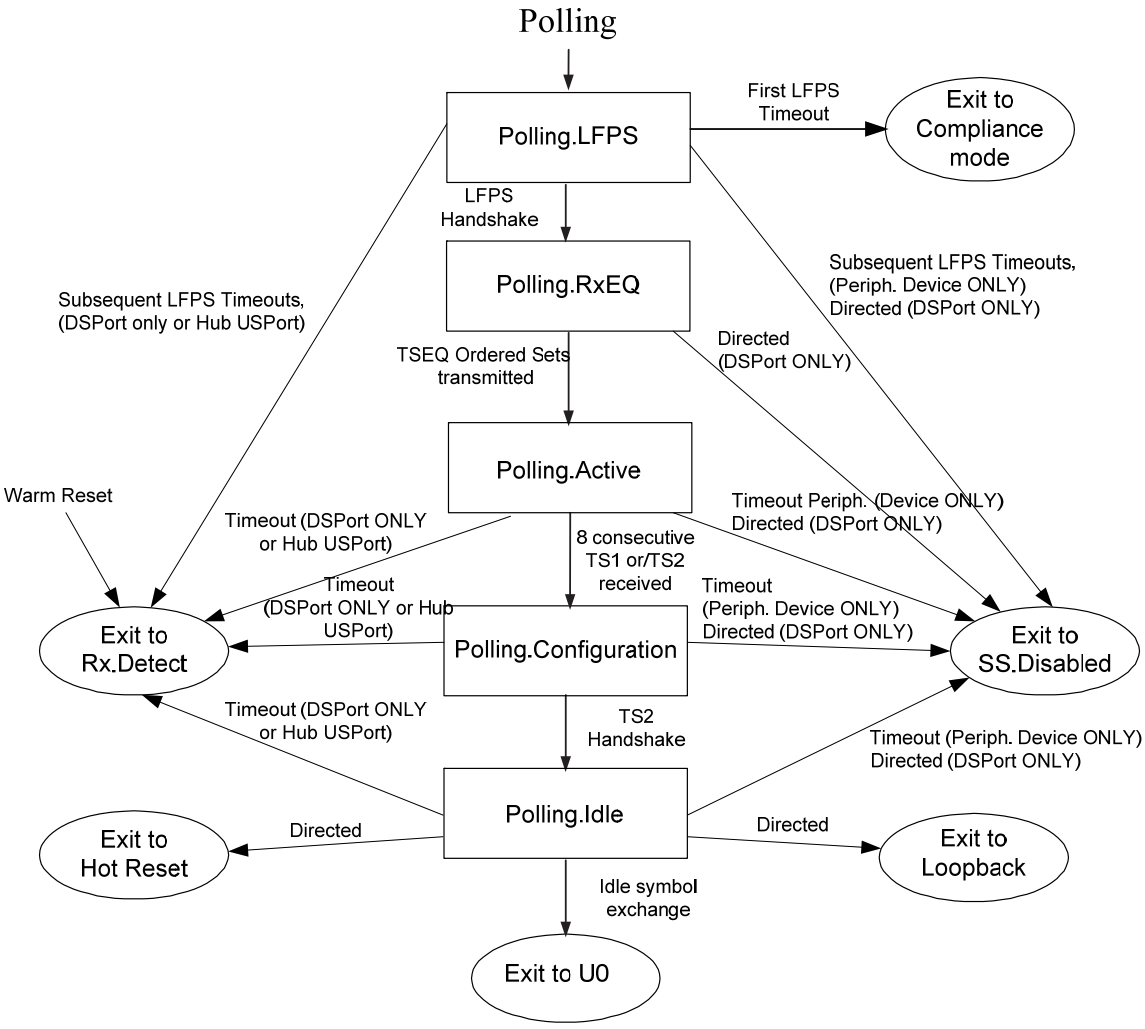
- A Port shall decode the TS2 Ordered Set received in Polling.Configuration.
- A Port shall perform a Reset as described in Section 7.3.2.
- A Port shall transmit Idle symbols.
- A 2 ms timer shall be started.
- A Port shall expect to receive Link Commands.

  Note: the skew between the two Ports entering U0 will result in one Port transmitting Link Command while the other Port is still in Polling.Idle.

Exit from Polling.Idle:

- A Port shall transition to Loopback as a Loopback master if directed and the Port is capable of being a Loopback master.
- A Port shall transition to Loopback as a Loopback slave if the Loopback bit is asserted in the TS2 Ordered Set.
- A Downstream Port shall transition to Hot Reset if directed.
- An Upstream Port shall transition to Hot Reset if Reset bit is asserted in the TS2 Ordered Set.
- A Port shall transition to U0 if the following two conditions are met:
  1. Eight consecutive Logical Idle symbols are received.
  2. 16 Logical Idle symbols are sent after receiving one Logical Idle symbol.
- A Downstream Port shall transition to SS.Disabled if directed.

- A Downstream Port shall transition to Rx.Detect upon 2 ms timeout and the conditions to transition to U0 are not met.
- An Upstream Port of a Hub shall transition to Rx.Detect upon 2 ms timeout and the conditions to transition to U0 are not met.
- An Upstream Port of a Peripheral Device shall transition to SS.Disabled upon 2 ms timeout and the conditions to transition to U0 are not met.
- A Downstream Port shall transition to Rx.Detect if a Warm Reset is Directed.
- An Upstream Port shall transition to Rx.Detect if a Warm Reset is detected.



Note: Transition conditions are illustrative only.

Figure 7-14. Polling Substate Machine

7.4.5 Compliance Mode

Compliance mode is for Transmitter test only. It is used to test if the Transmitter is compliant with the voltage and timing specifications. Several different test patterns are defined in Table 6-7.

Compliance Mode Substate Machines:

Compliance Mode does not contain any substate machines.

Compliance Mode Requirements:

- A Port shall enable its LFPS receiver.
- The transmitter shall wait until its Tx Common Mode settles, then shall send the first compliance test pattern defined Table 6-7 upon entry into Compliance mode.
- A Port shall transmit the next compliance test patterns upon detection of a Ping.LFPS defined in Section 6.9.1..
- A Port shall transmit the first compliance test pattern upon detection of a Ping.LFPS and the test pattern has reached the final test pattern.

Exit from Compliance Mode:

- A Port shall transition to Rx.Detect upon detection of a Warm Reset or a PowerOn Reset.
- A Downstream Port shall transition to SS.Disabled if directed.

7.4.6 U0

U0 is the normal operational state where packets can be transmitted and received.

U0 Substate Machines:

U0 does not contain any substate machines.

U0 Requirements:

- Upon entry into U0, a device shall reset the running count of transmitted Symbols since the last SKP Ordered Set. See Section 6.4.3 for details.
- All Tx and Rx specifications defined in Tables 6-10 and 6-12 shall be maintained.
- LFPS receiver shall be enabled.

Exit from U0:

- A Port shall transition to U1 upon successful completion of LGO_U1 entry sequence. See Section 7.1.4.2 for details.
- A Port shall transition to U2 upon successful completion of LGO_U2 entry sequence. See Section 7.1.4.2 for details.
- A Port shall transition to U3 upon successful completion of LGO_U3 entry sequence. See Section 7.1.4.2 for details.
- A Port shall transition to Recovery upon any error stated in Section 7.2 that causes a transition to the Recovery state.
- A Port shall transition to Recovery upon detection of a TS1 Ordered Set.
- A Downstream Port will transition to Recovery if directed.

- A Port shall transition to SS.Inactive if the following two conditions are met.
  1. Not receiving LGOOD_n corresponding to a Header Packet upon PENDING_HP timer timeout.
  2. Three consecutive transitions to Recovery have been tried due to not receiving LGOOD_n for the same Header Packet.
- A Downstream Port shall transition to SS.Disabled if Directed.
- A Downstream Port shall transition to Rx.Detect upon receiving no LUP within 120 consecutive Idle symbols.
- A Downstream Port shall transition to Rx.Detect if a Warm Reset is Directed.
- An Upstream Port shall transition to Rx.Detect if a Warm Reset is detected.
- An Upstream Port shall transition to SS.Disabled upon detection of VBUS off.

Note: this condition only applies to a self-powered Upstream Port. SS.Disabled is a logical power-off state for a self -powered Upstream Port.

7.4.7 U1

U1 allows a power saving opportunity over U0.

U1 Substate Machines:

U1 does not contain any substates. Transitions to other states are shown in Figure 7-15.

U1 Requirements:

- The Transmitter DC common mode voltage shall be within specification $V_{TX\text{-}CM\text{-}DC\text{-}ACTIVE\text{-}IDLE\text{-}DELTA}$.
- Rx Termination shall remain in low impedance.
- U1 resume functionality using LFPS signaling defined in Section 6.9.2 shall be maintained.
- LFPS Ping detection shall be enabled.
- An Upstream Port shall transmit a LFPS Ping every 200 ms.

Exit from U1:

- A Downstream Port shall transition to SS.Disabled if Directed.
- A Downstream Port shall transition to Rx.Detect receiving no valid LFPS Ping defined in Section 6.8 of the Physical Layer Chapter within 300 ms.
- A Downstream Port shall transition to Rx.Detect if a Warm Reset is Directed.
- An Upstream Port shall transition to Rx.Detect if a Warm Reset is detected.
- An Upstream Port shall transition to SS.Disabled upon detection of VBUS off.
- Note: this condition only applies to a self-powered Upstream Port. SS.Disabled is a logical power-off state for a self -powered Upstream Port. A Port shall transition to U2 upon the timeout of the U2 inactivity timer defined in Section 8.2.
- A Port shall transition to Recovery upon successful completion of a LFPS handshake meeting the U1 Exit LFPS handshake specification in Section 6.9.2.
- A Port shall transition to SS.Inactive upon the 2 ms LFPS handshake timeout and an un-successful LFPS handshake.

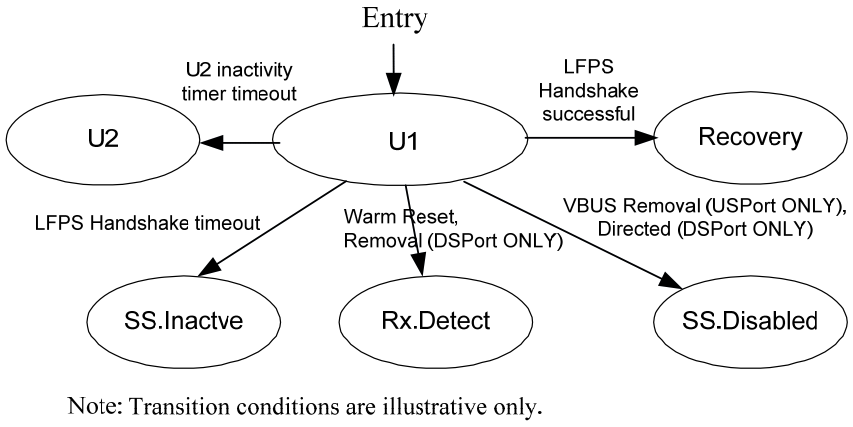


Figure 7-15. U1

7.4.8 U2

U2 is a Link power saving state similar to U1, but allows the opportunity for additional power reduction.

U2 Substate Machines:

U2 does not contain any substates. The transitions to other states are shown in Figure 7-16.

U2 Requirements:

- The DC common mode voltage does not need to be within specification $V_{TX\text{-}CM\text{-}DC\text{-}ACTIVE\text{-}IDLE\text{-}DELTA}$.
- Rx Termination shall remain in low impedance.
- LFPS Ping detection shall be disabled.
- U2 resume functionality using LFPS signaling see Section 6.9.2shall be maintained.
- A Downstream Port shall perform the receiver termination detection every 100 ms.

Exit from U2:

- A Downstream Port shall transition to SS.Inactive if one of the following two conditions is met.
  1. Upon detection of a far-end high impedance receiver termination
  2. If directed
- A Downstream Port shall transition to SS.Disabled if Directed.
- A Downstream Port shall transition to Rx.Detect upon detection of the high-impedance termination of a far-end receiver defined in Table 6-10.
- A Downstream Port shall transition to Rx.Detect if a Warm Reset is Directed.
- An Upstream Port shall transition to Rx.Detect if a Warm Reset is detected.
- An Upstream Port shall transition to SS.Disabled upon detection of VBUS off.

Note: this condition only applies to a self-powered Upstream Port. SS.Disabled is a logical power-off state for a self -powered Upstream Port.

- A Port shall transition to Recovery upon successful completion of a handshake meeting the U2 LFPS Exit specification defined in Section 6.9.2.
- A Port shall transition to SS.Inactive upon the 2 ms LFPS handshake timeout and an un-successful LFPS handshake.

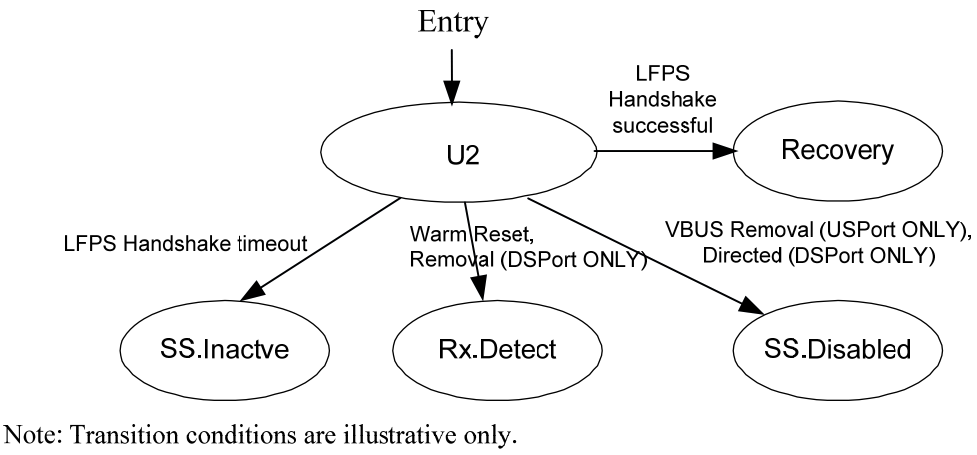


Figure 7-16. U2

7.4.9 U3

U3 allows the opportunity for aggressive power management.

U3 Substate Machines:

U3 does not contain any substates. Transitions to other states are shown in Figure 7-17.

U3 Requirements:

- The DC common mode voltage does not need to be within specification $V_{TX\text{-}CM\text{-}DC\text{-}ACTIVE\text{-}IDLE\text{-}DELTA}$.
- Rx Termination shall remain in low impedance.
- LFPS Ping detection shall be disabled.
- U3 resume functionality using LFPS signaling defined in Section 6.9.2.shall be maintained.
- A Downstream Port shall perform the receiver termination detection every 100 ms.

Exit from U3:

- A Downstream Port shall transition to SS.Inactive if one of the following two conditions is met.
  1. Upon detection of a far-end high impedance receiver termination
  2. if Directed
- A Downstream Port shall transition to SS.Disabled if Directed.
- A Downstream Port shall transition to Rx.Detect upon detection of the high-impedance termination of a far-end receiver defined in Table 6-10.
- A Downstream Port shall transition to Rx.Detect if a Warm Reset is Directed.
- An Upstream Port shall transition to Rx.Detect if a Warm Reset is detected.
- An Upstream Port shall transition to SS.Disabled upon detection of VBUS off.

Note: this condition only applies to a self-powered Upstream Port. SS.Disabled is a logical power-off state for a self -powered Upstream Port.

- A Port shall transition to Recovery upon successful completion of a handshake meeting the U3 LFPS Exit specification defined in Section 6.9.2 .
- A Port shall transition to SS.Inactive upon the 10 ms LFPS handshake timeout and an un-successful LFPS handshake meeting the U3 exit specification.

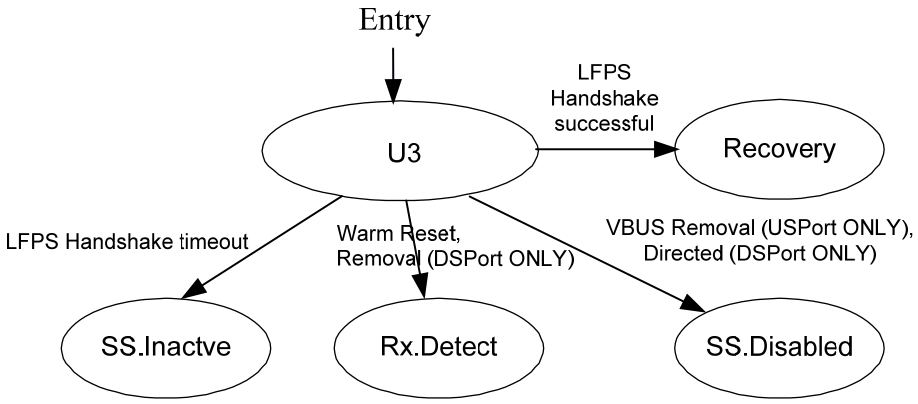


Figure 7-17. U3

7.4.10 Recovery

During Recovery, the Ports exchange TS1 and TS2 Ordered Sets. It allows the Link to be re-trained with minimum latency. It also allows the Link to be re-configured into different operation modes.

Recovery Substate Machines:

Recovery contains the following substate machines shown in Figure 7-18.

- Recovery.Active
- Recovery.Configuration
- Recovery.Idle

Recovery Requirements:

- Rx Termination shall remain in low impedance.
- Upon entry into Recovery all unacknowledged Tx Header Packets shall be maintained.
- Upon entry into Recovery all acknowledged Rx Header Packets and not processed shall be maintained.
- Upon entry into Recovery all unacknowledged Rx Header Packets shall be flushed.

  Note: The PENDING_HP and CREDIT_HP timers are disabled and reset.

7.4.10.1 Recovery.Active

Recovery,Active Requirements:

- The transmitter shall send TS1 Ordered Sets upon entry into Recovery.Active.
- A 12 ms timer shall be active.

Exit from Recovery.Active:

- A Port shall transition to Recovery.Configuration after eight consecutive and identical TS1 or TS2 Ordered sets are received.
- A Port shall transition to SS.Inactive if one of the following timers timeout and the condition to transition to Recovery.Configuration is not met.
  1. U1_exit timer,
  2. U2_exit timer,
  3. 12 ms timer.
- A Downstream Port shall transition to SS.Disabled if directed.
- A Downstream Port shall transition to Rx.Detect if a Warm Reset is Directed.
- An Upstream Port shall transition to Rx.Detect if a Warm Reset is detected.

7.4.10.2 Recovery.Configuration

Recovery.Configuration Requirements:

- If directed, a Port shall set Reset bit in the TS2 Ordered Set.
- If directed, a Port shall set Loopback bit in the TS2 Ordered Set.
- The Transmitter shall transmit TS2 Ordered Sets.
- A 12 ms timer shall be active.

Exit from Recovery.Configuration:

- A Port shall transition to Recovery.Idle after the following two conditions are met:
  1. Eight consecutive and identical TS2 Ordered Sets are received.
  2. 16 TS2 Ordered Sets are sent after receiving the first of the eight consecutive and identical TS2 Ordered Sets.
- A Port shall transition to SS.Inactive if one of the following timers times out and the condition to transition to Recovery.Configuration is not met.
  1. U1_exit timer,
  2. U2_exit timer,
  3. 12 ms timer.
- A Downstream Port shall transition to SS.Disabled if directed.
- A Downstream Port shall transition to Rx.Detect if a Warm Reset is Directed.
- An Upstream Port shall transition to Rx.Detect if a Warm Reset is detected.

7.4.10.3 Recovery.Idle

Recovery.Idle Requirements:

- A Port shall transmit and receive Idle symbols.
- A Port shall decode the TS2 Ordered Sets received in Recovery.Configuration.
- A 2 ms timer shall be active.
- A Port shall expect to receive Link Commands.

  Note: The skew between the two Ports entering U0 will result in one Port in U0 transmitting Link Command while the other Port is still in Recovery.Idle.

Exit from Recovery.Idle:

- A Port shall transition to Loopback as a Loopback master if directed and the Port is capable of being a Loopback master.
- A Port shall transition to Loopback as a Loopback slave if the Loopback bit is asserted in TS2 Ordered Sets.
- A Port shall transition to U0 if the following two conditions are met:
  1. 8 consecutive Logical Idle symbols are received.
  2. 16 Logical Idle symbols are sent after receiving one Logical Idle symbol.
- A Port shall transition to SS.Inactive if one of the following timers times out and the condition to transition to Recovery.Configuration is not met.
  1. U1_exit timer,
  2. U2_exit timer,
  3. 2 ms timer.
- A Downstream Port shall transition to Hot Reset if directed.
- A Downstream Port shall transition to SS.Disabled if directed.
- A Downstream Port shall transition to Rx.Detect if a Warm Reset is Directed.
- An Upstream Port shall transition to Rx.Detect if a Warm Reset is detected.
- An Upstream Port shall transition to Hot Reset if the Reset bit is asserted in TS2 Ordered Sets.

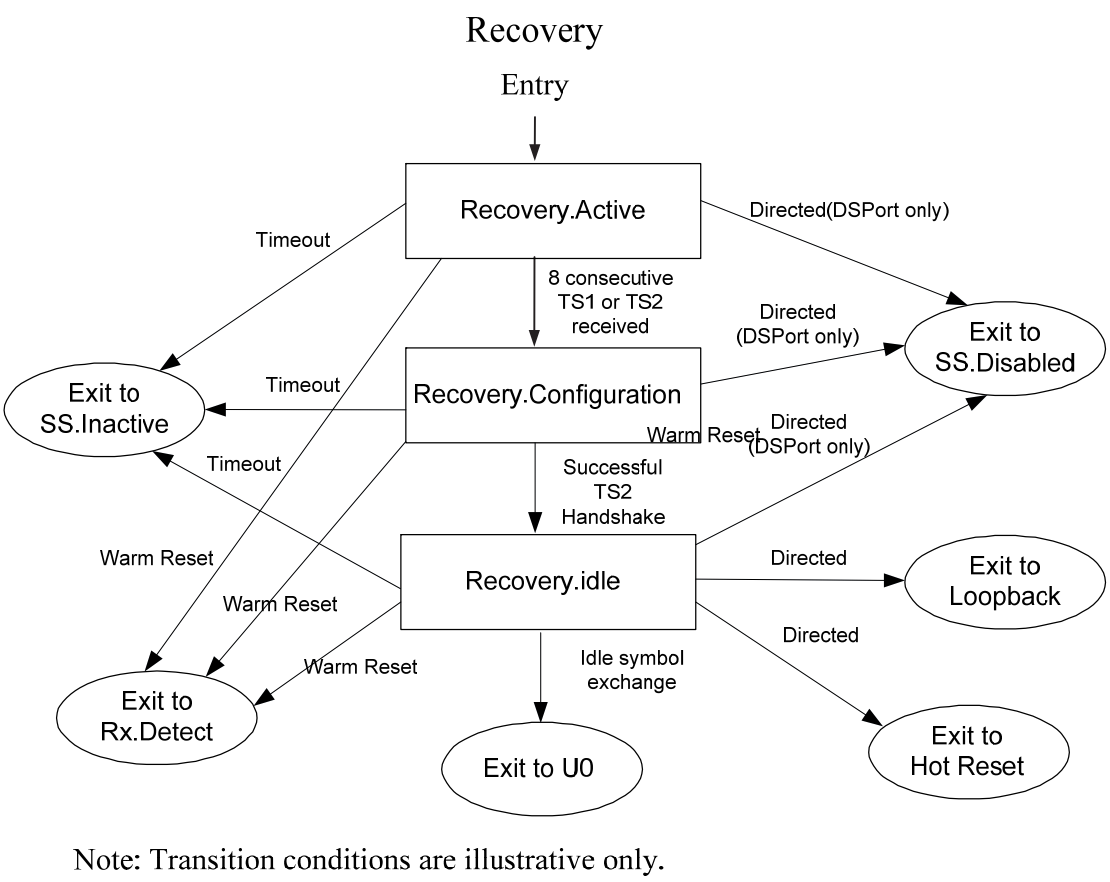


Figure 7-18. Recovery Substate Machine

7.4.11 Loopback

Loopback is intended for test and fault isolation use. Loopback includes a bit error rate test (BERT) state machine, described in Chapter 6.

A Loopback master is the component requesting Loopback. A Loopback slave is the component that retransmits the symbols received from the Loopback master.

During loopback active, the loopback slave must support the BERT protocol described in Chapter 6. The slave must respond to the command for BERT error counter reset and BERT report error count. The loopback slave must check the incoming data for the loopback data pattern.

Loopback Substate Machines:

Loopback contains the following substate machines shown in Figure 7-19.

- Loopback.Active
- Loopback.Exit

Loopback Requirements:

- There shall be one Loopback master and one Loopback slave. The Loopback master is the Port that has the loopback bit asserted in TS2 Ordered Set.

7.4.11.1 Loopback.Active

Loopback.Active Requirements:

- The Loopback master shall send valid 8b/10b data with SKPs as necessary.
- The Loopback slave shall retransmit the received 10-bit information as received, with lane polarity inversion determined in polling applied, while continuing to perform clock tolerance compensation.
- The Loopback slave shall not make modifications, other than the polarity inversion and SKP Symbols, which may be added or dropped as required, of the received 10-bit data.
  Note: this implies that the Loopback slave should disable or bypass its own 8b/10b encoder/decoder, and scrambler/de-scrambler.
- The loopback slave must process the BERT commands as appropriate.
- The LFPS transmitter and the LFPS receiver shall be enabled.

Exit from Loopback.Active:

- A Downstream Port shall transition to SS.Disabled if directed.
- A Downstream Port shall transition to Rx.Detect if a Warm Reset is Directed.
- An Upstream Port shall transition to Rx.Detect if a Warm Reset is detected.
- If directed, the Loopback master shall transition to Loopback.Exit.
- The Loopback slave shall transition to Loopback.Exit upon detection of LFPS exit signal meeting Loopback exit signaling defined in Section 6.9.2.

7.4.11.2 Loopback.Exit

Loopback.Exit Requirements:

- A 2 ms timer shall be active.
- A Port shall transmit and receive LFPS signals defined in Section 6.9.2.

Exit from Loopback.Exit:

- A Port shall transition to Rx.Detect upon a successful handshake defined in Section 6.9.2.
- A Port shall transition to Rx.Detect upon 2 ms timeout.
- A Downstream Port shall transition to SS.Disabled if directed.
- A Downstream Port shall transition to Rx.Detect if a Warm Reset is Directed.
- An Upstream Port shall transition to Rx.Detect if a Warm Reset is detected.

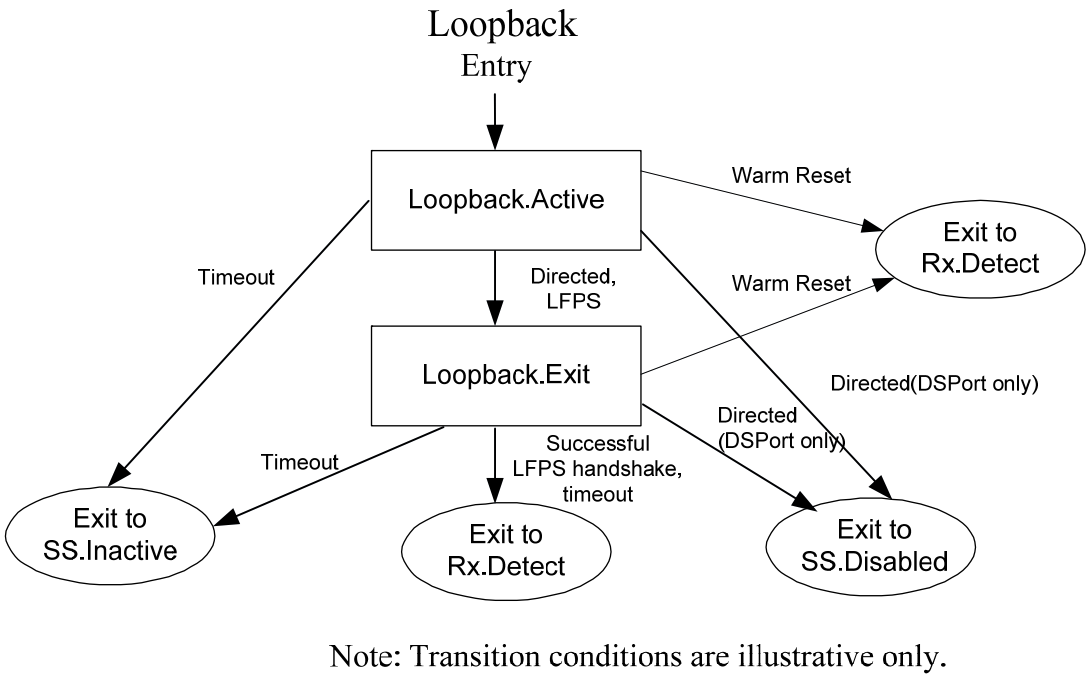


Figure 7-19. Loopback Substate Machine

7.4.12 Hot Reset

Only a Downstream Port can be directed to initiate a Hot Reset.

Hot Reset substate Machines:

Hot Reset contains the following substate machines shown in Figure 7-20.

- Hot Reset Active
- Hot Reset.Exit

Hot Reset Requirements:

- An Upstream Port shall return to “default” state defined in Section 9.1.1.3.
- A Downstream Port shall return to “DSPort.Enabled” defined in Section 10.3.1.5.
- The Port Configuration information shall remain unchanged (refer to Section 8.4.5 for details).

7.4.12.1 Hot Reset.Active

Hot Reset.Active Requirements:

- A Downstream Port shall transmit at least 16 TS2 Ordered Sets continuously with the Reset bit asserted.
- An Upstream Port shall transmit TS2 Ordered Sets with the Reset bit asserted.
- A 2 ms Hot Reset timer shall be active.
- A Port shall perform Hot Reset described in Hot Reset Requirement of this section.

Note: the 2 ms timer is started in Hot Reset.Active and will be active throughout the Hot Reset sub-states without reset.

Exit from Hot Reset.Active:

- A Port shall transition to SS.Inactive upon 2 ms timeout and the conditions to transition to Hot Reset.Exit are not met.
- A Downstream Port shall transition to SS.Disabled if directed.
- A Downstream Port shall transition to Rx.Detect if a Warm Reset is Directed.
- An Upstream Port shall transition to Rx.Detect if a Warm Reset is detected.
- A Downstream Port shall transition to Hot Reset.Exit upon completion of the transmitting 16 TS2 Ordered Sets with the Reset bit asserted.
- An Upstream Port shall transition to Hot Reset.Exit upon completion of its Port reset.

7.4.12.2 Hot Reset.Exit

Hot Reset.Exit Requirements:

- A Port shall transmit Logical Idle symbols.
- A Port shall receive Logical Idle symbols.
- The 2 ms timer shall be enabled.

Exit from Hot Reset.Exit:

- A Port shall transition to U0 if the following two conditions are met:
  1. Eight consecutive Logical Idle symbols are received.
  2. 16 Logical Idle symbols are sent after receiving one Logical Idle symbol.
- A Port shall transition to SS.Inactive upon 2 ms Timeout and the conditions to transition to U0 are not met.
- A Downstream Port shall transition to SS.Disabled if directed.
- A Downstream Port shall transition to Rx.Detect if a Warm Reset is Directed.
- An Upstream Port shall transition to Rx.Detect if a Warm Reset is detected.

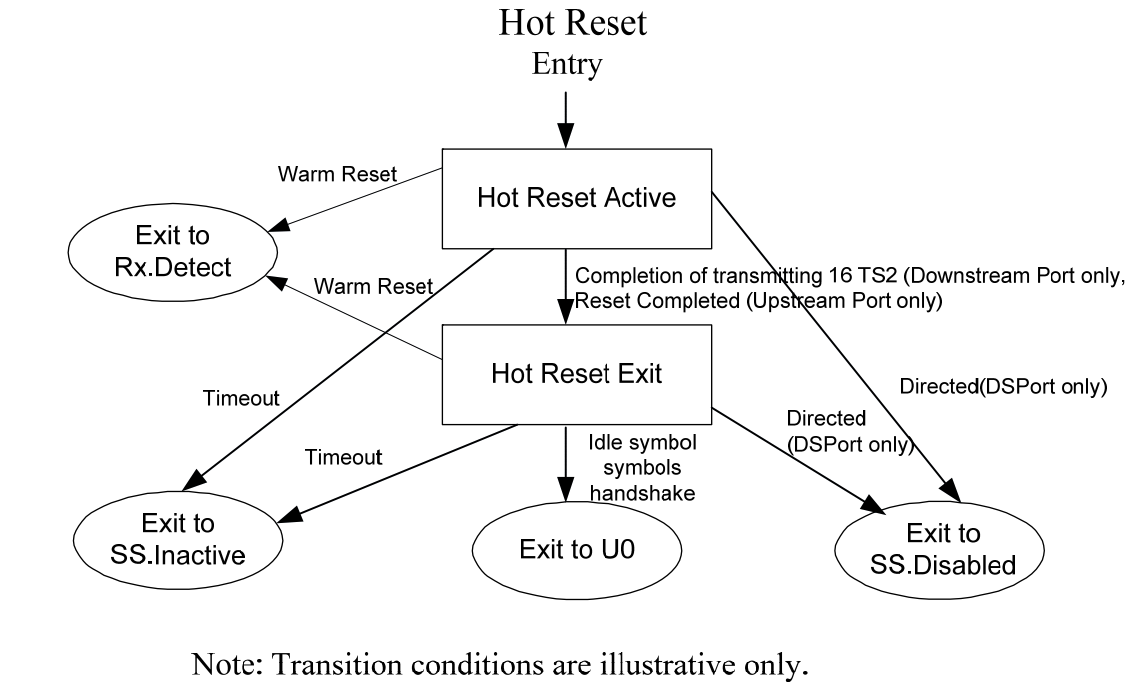


Figure 7-20. Hot Reset Substate Machine

8 Protocol Layer

The protocol layer manages the end to end flow of data between a device and its host. This layer is built on the assumption that the link layer guarantees delivery of certain types of packets and this layer adds on end to end reliability for the rest of the packets depending on the transfer type.

The chapter describes the following in detail:

- Types of packets
- Format of the packets
- Expected responses to packets sent by a host and a device
- Describes the four USB 3.0 transaction types
- Support for streaming support for the Bulk transfer type
- The timing parameters for the various responses and packets a device/host may receive/transmit

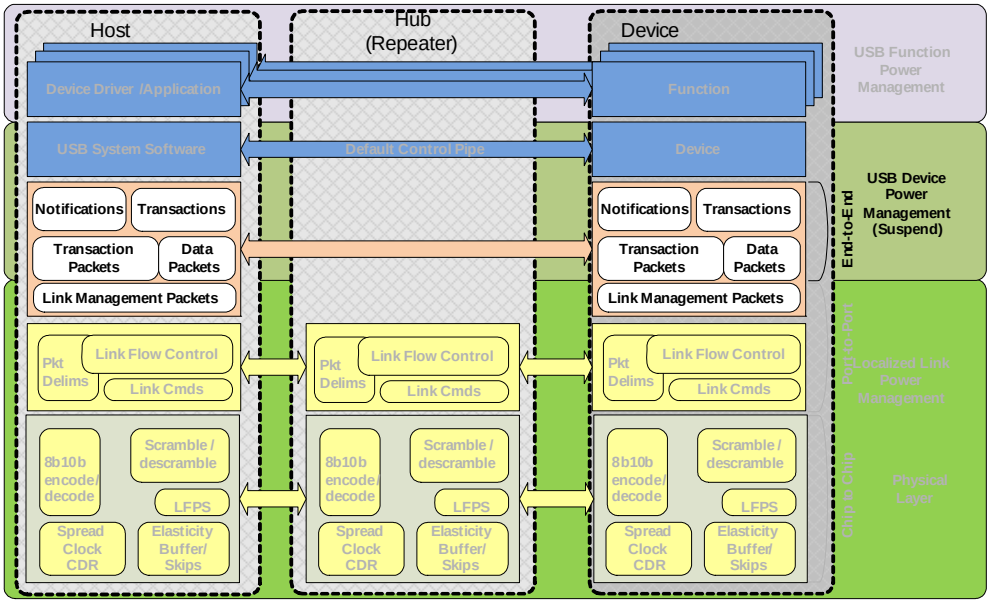


Figure 8-1. Protocol Layer Highlighted

8.1 USB 3.0 Transactions

SuperSpeed USB transactions are initiated by the host when it either requests or sends data to a device and are completed when the device sends the data or acknowledges receipt of the data. Note: A transfer is a request for data made by a device application to the host which then breaks it up into a series of transactions. For non-isochronous transfers, the device may respond to a valid request by:

- returning an NRDY Transaction Packet
- accepting it by returning an ACK Transaction Packet in the case of an OUT transaction
- returning one or more data packets in the case of an IN transaction.
- returning a STALL Transaction Packet if there is an internal endpoint error

In the case of an NRDY Transaction Packet response, there will be no further activity between the host and that endpoint on the device until the endpoint notifies the host that it is ready. This allows the links between the device and the host to be placed in a reduced power state until the device is ready to receive or send data. When ready, the device asynchronously sends a notification (ERDY Transaction Packet) to the host to tell it that it is now ready to move data and the host responds by rescheduling the request. Note Isochronous transfers do not use ERDYs as they are serviced by the host at periodic intervals. Additionally, data packets sent to or received from an isochronous endpoint are not acknowledged, i.e., no ACK Transaction Packets are sent to acknowledge the receipt of a data packet.

Devices only respond to requests made by the host. The host is responsible for scheduling transactions on the bus and maintaining the priority and fairness of the data movement on the bus; it does this by the timing and ordering of IN and OUT requests. Transactions are not broadcast; packets traverse a direct path between the host and device. Any unused links may be placed into reduced power states making the bus amenable to aggressive power management.

8.2 Packet Types

SuperSpeed USB uses four basic packet types each with one or more subtypes. The four packet types are:

- **Link Management Packet** (LMP) which only travels between a pair of links (e.g., a pair of directly connected ports) and is primarily used to manage that link.
- **Transaction Packet** (TP) which traverses all the links directly connecting the host to the device. It is used to control the flow of data packets, configure devices and hubs etc. Transaction Packets have no data payload.
- **Data Packet** (DP) which traverses all the links directly connecting the host to the device. Data Packets have two parts: a Data Packet Header (DPH) and a Data Packet Payload (DPP).
- **Isochronous Timestamp Packet** (ITP) which is multicast all the links in U0 from the host to one or more devices**.**

All packets start with a 16-byte header. TPs, LMPs, and ITPs consist of a header only. The header is structured to minimize the amount of processing needed to interpret it and maximize performance. All headers begin with two common fields that are used by the receiving entity (e.g., host, hub or device) to determine how to process the packet. All headers are protected by a 2-byte

CRC (CRC-16) which is followed by a 2-byte Link Control Word at the end of the packet. Note that the CRC-16 does not protect the 2-byte Link Control Word as it has its own CRC (CRC-5).

All devices (including hubs) and the host consume the LMPs they receive. Hubs have the additional responsibility to forward DPs, ITPs and TPs to the downstream port nearer the device or to the upstream port nearer the host. Note that ITPs are only sent by a host and received by devices.

Packet headers have several characteristics in common:

- They are of the same fixed length - 16 bytes
- Link Commands (see Section 7.1.2) are used to ensure that all headers (LMPs, TPs, ITPs and DPHs) have an uncorrectable/undetectable error rate less than one error in 10e20 bits.
- All packets except LMPs are forwarded by hubs unless the packet is routed to the hub itself. Additional rules for forwarding ITPs are described in Section 8.7. Note the Link Control Word in a TP, ITP or DPH may be changed by a hub after it receives the packet and before it forwards it.

The Link Control Word is always sent last and the fields in the Link Control Word are described in Section 8.3.1.1.

If the **Packet Type** field is set to *Transaction Packet* or *Data Packet Header* Type, the **Route String** and **Device Address** fields follow the **Packet Type** field. The **Route String** field is used by hubs to identify and route packets which appear on their upstream port to the appropriate downstream port. The routing of packets flowing from a device to the host is fixed as they are always routed from the hub's downstream port to its upstream port. The **Device Address** field is provided to the host so that it can identify the source of a packet and associate it with the appropriate buffer.

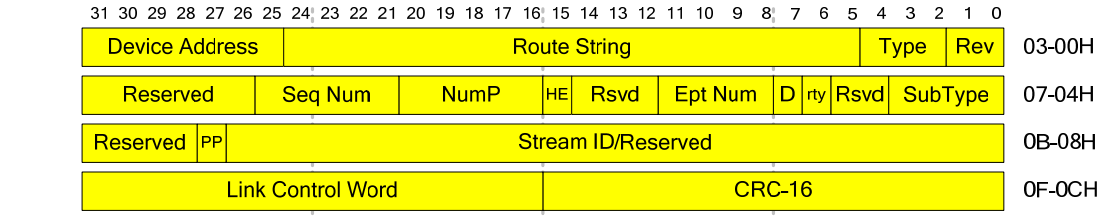


Figure 8-2. Example Transaction Packet

Data Packets include additional information in the header that describes the data block. The Data Block is always followed by a 4-byte CRC-32 used to determine the correctness of the data. The Data Block and the CRC-32 together are referred to as the Data Packet Payload.

8.3 Packet Formats

This section defines the SuperSpeed USB packets. It defines the fields that make up the varying packet type and subtypes.

Note: Packet byte and bit definitions in this section are described in an un-encoded data format. The effects of symbols added to the serial stream (i.e., to frame packets or control/modify the link), bit encoding, bit scrambling and link level framing have been removed for the sake of clarity. See

the Physical Layer and Link Layer Chapters for detailed information. In cases where bus performance, efficiency or timing is discussed, the effects of these lower level operations may be discussed to provide additional context.

8.3.1 Fields Common to all Headers

All SuperSpeed USB headers start with two fields that are used to determine how to interpret the packet. At a high level this tells the recipient of the packet what to do with it: either to use it to manage the link or to move and control the flow of data between the device and the host. In addition all packets end with a 2-byte Link Control Word.

A receiver shall ignore any packet that has reserved values in defined fields. However, the receiver shall acknowledge the packet and return credit for the same as per the requirement specified in Section 7.1.4.1.

8.3.1.1 Packet Revision Field

The **Revision** field is a two-bit field that identifies the USB Specification revision level that the packet conforms to. The presence of a **Revision** field provides extensibility so that features may be added to the USB packet definition over time. Note that any changes to future revisions will be done in a manner that maintains backward compatibility. A device or host that receives a packet with a reserved value in the revision field shall ignore the packet.

Table 8-1. Packet Revision Field Description

Field Name	Width	Offset	Description
Revision	2b	00b	These 2-bits identify the USB Specification Revision number the packets conform to. **Value** — **Description** 00b — USB 3.0. 01b – 11b — Reserved

8.3.1.2 Packet Type Field

The **Type** field is a three-bit field that identifies the packet type. The type is used to determine how the packet is to be used or forwarded by intervening links.

Table 8-2. Packet Type Field Description

Field Name	Width	Offset	Description
Type	3b	02b	These 3-bits identify the packet's Type **Value** — **Description** 000b — Link Management Packet. 001b — Transaction Packet 010b — Data Packet Header 011b — Isochronous Timestamp Packet 100b – 111b — Reserved

8.3.1.3 CRC-16

All header packets have a 16 bit CRC field. This field is the CRC over the preceding 12 bytes in the header packet. Refer to Section 7.1.1.1.2 for the polynomial used to calculate this value.

8.3.1.4 Link Control Word

The usage of this word is defined in the Link Layer chapter, see Section 7.1.1.1.3.

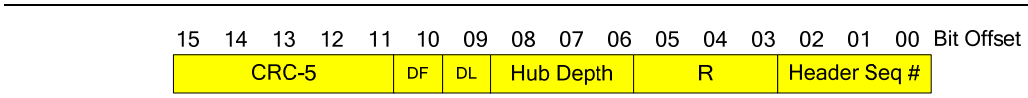


Figure 8-3. Link Control Word Detail

Table 8-3. Link Control Word Format

Field Name	Width	Offset	Description
Header Sequence Number	3b	112b	The valid values in this field are 0 through 3. All other values are reserved.
Reserved (R)	3b	115b	These bits are reserved and shall be set to zero.
Hub Depth	3b	118b	This field is only valid when the **Deferred** bit is set and identifies to the host the hierarchical on the USB that the hub is located at in the deferred TP or DPH returned to the host. This informs the host that the port on which the packet was supposed to be forwarded on is currently in a low power state (either U1 or U2). The only valid values in this field are 1 through 5.
Delayed (DL)	1b	121b	This bit may only be set by a hub. The Hub Chapter provides more details on when this bit shall be set. This bit shall not be reset by any subsequent hub that this packet traverses.
Deferred (DF)	1b	122b	This bit may only be set by a hub. This bit shall be set when the downstream port on which the packet needs to be sent is in a power managed state. This bit shall not be reset by any subsequent hub that this packet traverses.
CRC-5	5b	123b	This field is the CRC used to verify the correctness of the preceding 11-bits in this word. Refer to Section 7.1.1.1.3 for the polynomial used to calculate this value.

8.4 Link Management Packet (LMP)

All packets including LMPs start with the two common fields: **Revision** (see Section 8.3.1.1) and **Type** (see Section 8.3.1.2). Packets identified by the **Type** field set to *Link Management Packet* are referred to as LMPs and are used to manage a single link. They carry no addressing information and as such are not routable. They may be generated and placed on the bus as the result of hub port commands, for example a hub port command used to set the U2 inactivity timeout. In addition they may be used to exchange port capability information or for test purposes.

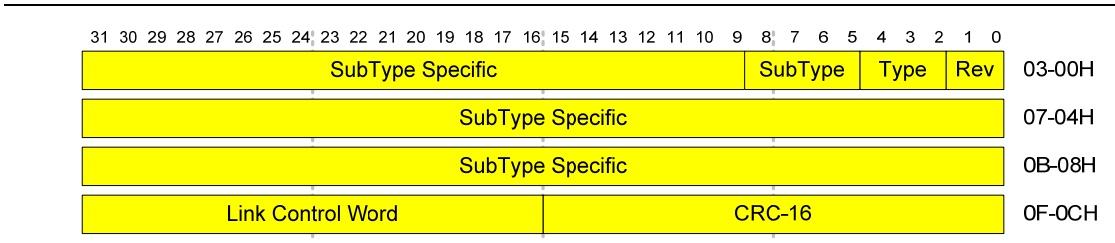


Figure 8-4. Link Management Packet Structure

8.4.1 Subtype Field

The numeric value in the LMP **Subtype** field further identifies the contents of the LMP.

Table 8-4. Link Management Packet Subtype Field

Field Name	Width	Offset	Description
Subtype	4b	05b	These 4 bits identify the Link Packet Subtype: **Value** — **Type of LMP** 0000b — Reserved 0001b — Set Link Function 0010b — U2 Inactivity Timeout 0011b — Vendor Device Test 0100b — Port Capability 0101b — Port Configuration 0110b — Port Config Ack All codes not explicitly assigned are reserved and must not be used.
CRC-16	16b	96b	This field is the CRC over the preceding 12 bytes. Refer to Section 7.1.1.1.2 for the polynomial used to calculate this value.

8.4.2 Set Link Function

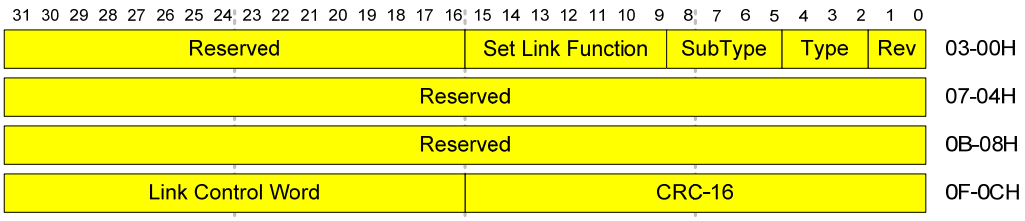


Figure 8-5. Set Link Function LMP

The **Set Link Function** LMP shall be used to configure functionality that can be changed without leaving the active (U0) state.

Upon receipt of a LMP with the Force_LinkPM_Accept bit asserted, the port shall accept all LGO_U1 and LGO_U2 Link Commands until the bit is de-asserted or VBUS is removed.

Note: Improper use of the Force_LinkPM_Accept functionality can impact the performance of the link significantly. This capability shall only be used for compliance and testing purposes. Software must ensure that there are no pending packets at the link level before issuing Set Port Feature command that generates an LGO_U1 or LGO_U2 link command.

This LMP is sent by a hub to the device connected on a specific port when it receives a SetPortFeature (FORCE_LINKPM_ACCEPT) command. See Section 10.4.2.2 and Section 10.4.2.9 for more details.

Table 8-5. Set Link Function

Field Name	Width	Offset	Description
SubType	4b	05b	This field shall be set to *Set Link Function* for a Set Link Functionality LMP
Set Link Function	7b	09b	These 7 bits identify the Set Link Function **Bits** — **Description** 0 — RESERVED. Default to 0b. 1 — Force_LinkPM_Accept **Value** — **Meaning** 0 — De-assert 1 — Assert 6:2 — The bits are Reserved and shall be set to zero.

8.4.3 U2 Inactivity Timeout

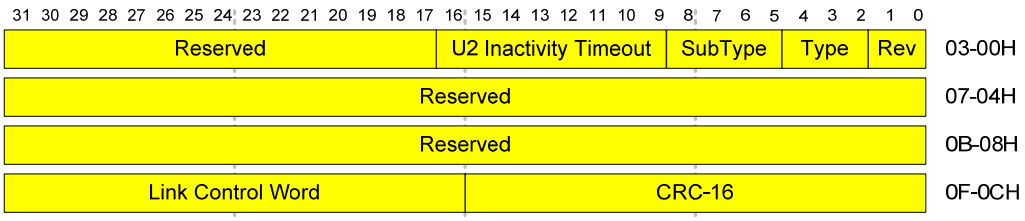


Figure 8-6. U2 Inactivity Timeout LMP

The **U2 Inactivity Timeout** LMP shall be used to define the timeout from U1 to U2. See Section 10.4.2.1 for details on this LMP.

Table 8-6. U2 Inactivity Timer Functionality

Field Name	Width	Offset	Description
SubType	4b	05b	This field shall be set to *U2 Inactivity Timeout* for a U2 Inactivity Timeout LMP
U2 Inactivity Timeout	8b	09b	These 8 bits identify the desired U2 Inactivity Timeout values. The value placed in this field is the same value that is sent to the hub in a Set Port Feature (PORT_U2_TIMEOUT) command. See Section 10.14.2.9 for details on the encoding of this field. The timer in the upstream facing port shall be accurate to +/-1us. This is required so that both link partners transition from U1 to U2 at approximately the same time.

8.4.4 Vendor Device Test

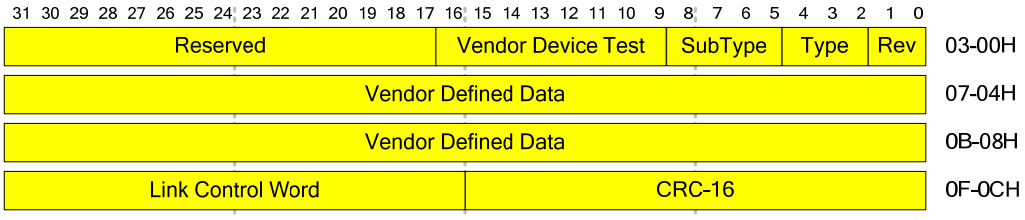


Figure 8-7. Vendor Device Test LMP

Use of this LMP byte is intended to simplify device testing. The default state of all bits in the Vendor-specific device test field shall be 0b. The functional definition of all enabled (set to 1) bits is implementation specific, and shall not be used for the normal operation of the link.

Table 8-7. Vendor-specific Device Test Function

Field Name	Width	Offset	Description
SubType	4b	05b	This field shall be set to *Vendor Device Test* for a Vendor Device Test LMP
Vendor-specific device test	8b	09b	The functionality of these 8 bits is implementation specific and all the values are vendor specific.

8.4.5 Port Capabilities

This LMP is sent after the successful completion of training by both the link partners to each other. This LMP is used to describe each device's link capabilities to its link partner. All SuperSpeed capable devices must send this LMP within tPortConfiguration time after a warm reset or power on reset (see Section 7.3) has successfully completed and the link is in U0.

If a link partner does not receive this LMP within tPortConfiguration time then:

- If the link partner has downstream capability it shall signal an error as described in Section 10.14.2.5.
- If the link partner only supports upstream capability then it shall try and connect at the other speeds this device supports.

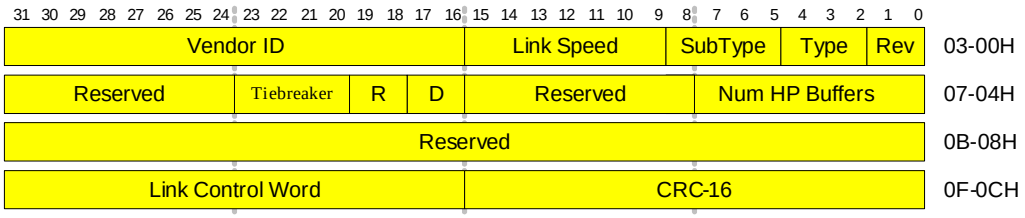


Figure 8-8. Port Capabilty LMP

Table 8-8. Port Capability LMP Format

Width	Offset	Description
4b	05b	**SubType.** This field shall be set to *Port Capability* for a Port Capability LMP
7b	09b	**Link speed**. This field describes the link speeds supported by this device. **Bits** — **Description** 0 — This bit is set to 1 then this device supports signaling at 5Gbs (Default). 6:1 — These bits are reserved and shall be set to zero by devices that are compliant to this revision of the specification.
16b	16b	**Vendor ID**. This field shall be set to the USB Vendor ID of the device sending this LMP.
8b	32b	**Num HP Buffers**. This field specifies the number of header packet buffers (in each direction Transmit or Receive) this device has. All devices that are compliant to this revision of the specification shall return a value of 4 in this field. All other values are reserved.
8b	40b	**Reserved (R)**. This field is reserved and shall be set to zero.

Width	Offset	Description
2b	48b	**Direction (D)**. This field is used to identify the upstream/downstream capabilities of the port. All ports shall have at least one of these bits set. **Bits** — **Description** 0 — If this bit is set to 1 then this port can be configured to be a downstream port. 1 — If this bit is set to 1 then this port can be configured to be an upstream port
2b	50b	**Reserved (R)**. This field is reserved and shall be set to zero
4b	52b	**Tiebreaker.** This field is only valid when the both bits 0 and 1 of the **Direction** field are set. This field is used to determine the port type when two devices with both upstream and downstream capability are connected to each other. See Table 8-10 for details. This field must be set to zero in all other cases.
40b	56b	**Reserved**. This field is reserved and shall be set to zero.

8.4.6 Port Configuration

After exchanging Port Capability LMPs, the link partners will determine which of the link partners will be configured as the downstream facing port as specified in Table 8-10. The link partner that will be configured as the downstream facing port shall send the Port Configuration LMP to the link partner that will be configured as the upstream facing port to complete the configuration of the link partners within tPortConfiguration time after a warm reset or power on reset (see Section 7.3) has successfully completed and the link is in U0. The link partner sending this LMP sets the fields to values that it expects the upstream facing port to operate at. Only the fields that are different from the Port Capability LMP are described in this section.

All SuperSpeed ports that support downstream port capability shall be capable of sending this LMP.

If the port that was to be configured in the upstream facing mode does not receive this LMP within tPortConfiguration time then it shall try and connect at the other speeds this device supports.

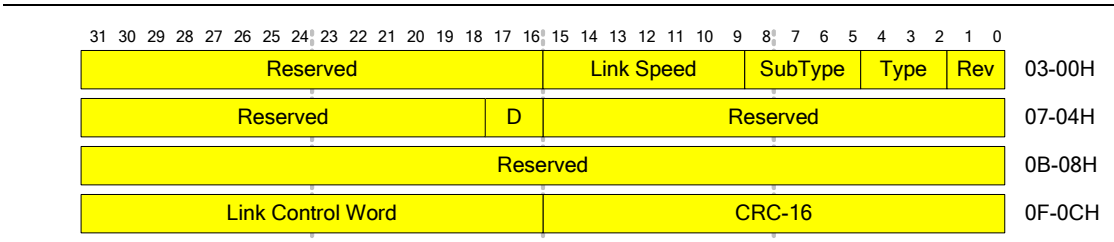


Figure 8-9. Port Configuration LMP

Table 8-9. Port Configuration LMP Format (differences with Port Capability LMP)

Width	Offset	Description
4b	05b	**SubType.** This field shall be set to *Port Configuration* for a Port Configuration LMP
7b	09b	**Link speed**. This field describes the link speed the upstream port shall operate at. Only one of the bits in this field shall be set in the Port Configuration LMP sent by the link partner configured in the downstream mode. **Bits** — **Description** 0 — If this bit is set to 1 then this device shall operate at 5Gbs (Default). 6:1 — These bits are reserved and shall be set to zero by devices that are compliant to this revision of the specification.
24b	32b	**Reserved**. This field is reserved and shall be set to zero.
2b	48b	**Direction (D)**. This field must always be set to 10b in a Port Configuration LMP sent by the downstream facing port.
62b	46b	**Reserved**. This field is reserved and shall be set to zero.

A port's Port Type is selected as determined by Table 8-10.

Table 8-10. Port Type Selection Matrix

		Device's Link Partner Port Capability		
		Upstream Only	**Downstream Only**	**Both**
Device Port Capability	**Upstream Only**	Not Allowed	Automatic	Devices with both capabilities will automatically become the downstream port
	Downstream Only	Automatic	Not Allowed	Devices with both capabilities will automatically become the upstream port
	Both	Devices with both capabilities will become automatically the downstream port	Devices with both capabilities will automatically become the upstream port	The device with the higher value in the **Tiebreaker** field will become the downstream port[1]

Note: [1]If the **TieBreaker** field contents are equal, then the two link partners shall exchange Port Capability LMPs again with new and different value in the **TieBreaker**field. The sequence of TieBreaker field values generated by a port shall be either unpredictably random or deterministic. If the sequence is deterministic it shall be different than the sequence generated by all potential link partners.

A port configured in the downstream mode shall send the Port Configuration LMP to the upstream port. The port sending this LMP must set the **Direction** field to upstream only (10b) and select only one bit for the **Link Speed** field.

If a downstream capable port cannot work with its link partner then the downstream capable port must signal an error as described in Section 10.14.2.5.

8.4.7 Port Configuration ACK

This LMP is sent by the device configured in the upstream mode to the device configured in the downstream mode in response to a Port Configuration LMP within tPortConfiguration time after a warm reset or power on reset (see Section 7.3) has successfully completed and the link is in U0. It is used to indicate acceptance/rejection of the Port Configuration LMP. Only the fields that are different from the Port Capability LMP are described in this section.

All SuperSpeed ports that support upstream port capability shall be capable of sending this LMP.

If the downstream facing port does not receive this LMP within tPortConfiguration time then it shall signal an error as described in Section 10.14.2.5.

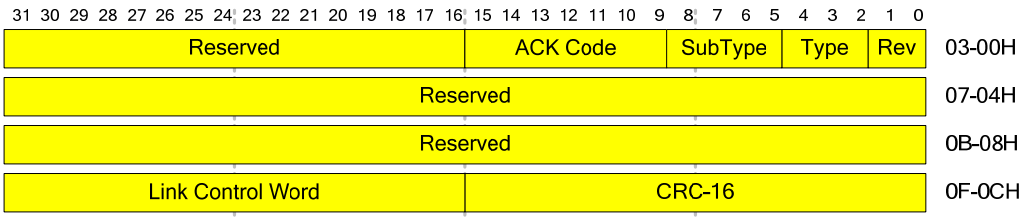


Figure 8-10. Port Configuration ACK LMP

Table 8-11. Port Configuration ACK LMP Format (differences with Port Capability LMP)

Width	Offset	Description
4b	05b	**SubType.** This field shall be set to *Port Configuration ACK* for a Port Configuration ACK LMP
7b	09b	**ACK Code**. This field indicates the settings that were accepted in the Port Configuration LMP that was sent to the device. **Bits** — **Description** 0 — If this bit is set to 1, then this device accepted the Link Speed setting 1 — If this bit is set to 1, then this device accepted the Direction setting 6:2 — These bits are reserved and shall be set to zero
80b	16b	**Reserved**. This field is reserved and shall be set to zero.

If the ACK Code indicates that any of the configuration options was rejected by the upstream port then:

- The downstream facing port shall signal an error as described in Section 10.14.2.5.
- The upstream facing port shall try and connect at the other speeds this device supports.

8.5 Transaction Packet (TP)

Transaction Packets (TPs) traverse a path between the host and device. A TP is identified by the **Type** field set to *Transaction Packet*. A TP also includes address and routing information. The routing information, which is present in the **Route String** field, is used by hubs to route a packet that appears on its upstream port to the appropriate downstream port for packets going from the host to the device. The **Device Address** field contains the address of the device that is the intended recipient of the TP. When a device sends TPs back to the host then the **Device Address** field is used by the host to identify the source of the TP. TPs are used to control data flow between devices and the host as well as to manage the end to end connection. The **SubType** field type in a TP is used by the recipient to determine the format and usage of the TP.

Table 8-12. Transaction Packet Subtype Field

Field Name	Width	Offset	Description
SubType	4b	32b	The subtype field is used to identify a specific type of TP

Value	Type of TP
0000b	Reserved
0001b	ACK
0010b	NRDY
0011b	ERDY
0100b	STATUS
0101b	STALL
0110b	DEV_NOTIFICATION
0111b	PING
1000b	PING_RESPONSE
1001b – 1111b	Reserved

8.5.1 ACK Transaction Packet

This TP is used for two purposes:

- for IN endpoints, this TP is sent by a host to request data from a device as well as to acknowledge any previously received data
- for OUT endpoints, this TP is sent by a device to acknowledge receipt of Data Packets sent by the host as well as to inform of the host of the number of Data Packets buffers it has available after receipt of this packet.

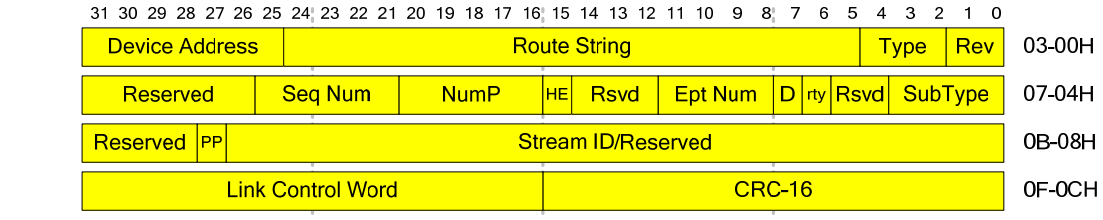


Figure 8-11. ACK Transaction Packet

Table 8-13. ACK TP Format

Width	Offset	Description
20b	5b	**Route String**. This field is only used by hubs. In conjunction with a Route String Index, it is used to route a packet to the correct downstream port. See Section 8.9 for details.
7b	25b	**Device Address**. This field specifies the device, via its address, that is the recipient of the TP. A total of 127 addresses may be specified. Upon reset and power-up, a device's address defaults to a value of zero and shall be programmed by the host during the enumeration process. Device address zero is reserved as the default address and may not be assigned to any other use.
4b	32b	**SubType**. This field shall be set to *ACK* for an ACK TP.
2b	36b	**Reserved (Rsvd)**. This field is reserved and shall be set to zero.
1b	38b	**Retry Data Packet (rty)**. This field is used to signal that the host or device did not receive a Data Packet and requests the transmitter to resend one or more Data Packets starting at the specified Seq Num.
1b	39b	**Direction (D)**. This field defines the direction of the data flow **Value** — **Direction of Data Flow** 0b — Host to Device 1b — Device to Host
4	40b	**Endpoint Number (Ept Num)**. This field determines the Endpoint within the device that is the recipient of this TP. SuperSpeed USB devices may support up to a maximum of 15 IN and OUT endpoints apart from the required default control endpoint with an Ept Num set to zero.
3	44b	**Reserved (Rsvd)**. This field is reserved and shall be set to zero.
1b	47b	**Host Error (HE)**. This field is only valid when the ACK TP is sent from the host to a device. This bit shall be set if the host was unable to accept a valid Data Packet due to internal Host issues. When a host sets this field to a one, it must also set the **Retry Data Packet** field to a one.
5b	48b	**Number of Packets (NumP)**. This field is used to indicate the number of Data Packet buffers that the receiver can accept. The value in this field shall be less than or equal to the Max Burst supported by the endpoint as determined by the value in the **Burst Size** field in the Endpoint Companion Descriptor (see Section 9.6.7in the Framework Chapter).
5b	53b	**Sequence Number (Seq Num)**. This field is used to identify the next packet requested and by implication ACKing packets with previous Sequence Numbers. Note that the sequence number wraps around at 31. Hence, if the Seq Num is 0, then the ACK TP acknowledges any previous packets that may have been sent by the transmitter with sequence numbers 31 and below.
6b	58b	**Reserved**. This field is Reserved and shall be set to zero.
27b	64b	**Stream ID/Reserved**. If this ACK TP is targeted at a Bulk endpoint then the field is interpreted as follows: **Bits** — **Description** 15:0 — This is a one based Stream ID field. The usage of this field is class dependent. 26:16 — These bits are reserved and shall be set to zero. If this ACK TP is targeted at a Control, Isochronous or Interrupt endpoint then this field shall be set to zero. See Section 8.12 for further details.
1b	91b	**Packets Pending (PP)**. This field can only be set by a Host. If the field is set the host has another packet available for the endpoint identified by the **Endpoint Number and Direction** field. If no endpoints on this device have packets pending, then the device can use this information to aggressively power manage its upstream link e.g., set the link to a lower power U1 or U2 state.
4b	92b	**Reserved**. This field is Reserved and shall be set to zero.

8.5.2 NRDY Transaction Packet

This TP can only be sent by a device for a non-isochronous endpoint. An OUT endpoint sends this TP to the host if it has no packet buffer space available to accept the DP sent by a host. An IN endpoint sends this TP to the host if it cannot return a DP in response to an ACK TP sent by a host.

Only the fields that are different from an ACK TP are described in this section.

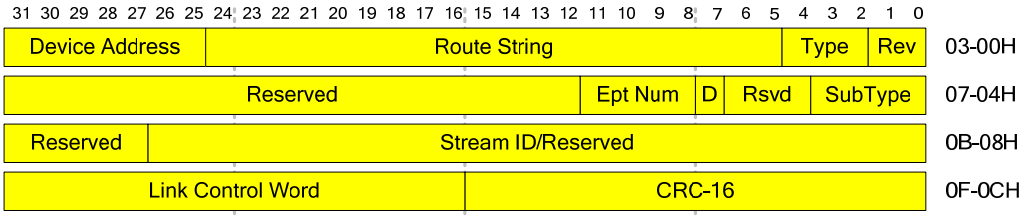


Figure 8-12. NRDY Transaction Packet

Table 8-14. NRDY TP Format (differences with ACK TP)

Width	Offset	Description
4b	32b	**SubType**. This field shall be set to *NRDY* for an NRDY TP.
3b	36b	**Reserved (Rsvd)**. This field is reserved and shall be set to zero.
20b	44b	**Reserved**. This field is reserved and shall be set to zero.
5b	90b	**Reserved**. This field is reserved and shall be set to zero.

8.5.3 Endpoint Ready (ERDY) Transaction Packet

This TP can only be sent by a device for a non-isochronous endpoint. It is used to inform the host that the endpoint is ready to send or receive Data Packets. Only the fields that are different from an ACK TP are described in this section.

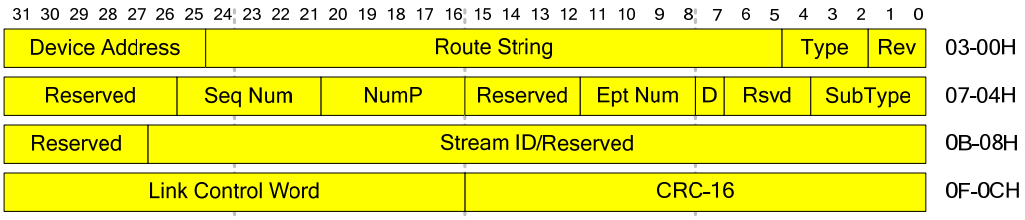


Figure 8-13. ERDY Transaction Packet

Table 8-15. ERDY TP Format (differences with ACK TP)

Width	Offset	Description
4b	32b	**SubType**. This field shall be set to *ERDY* for an ERDY TP.
3b	36b	**Reserved (Rsvd)**. This field is reserved and shall be set to zero.
4b	44b	**Reserved (Rsvd)**. This field is reserved and shall be set to zero.
5b	90b	**Reserved**. This field is reserved and shall be set to zero.

8.5.4 STATUS Transaction Packet

This TP can only be sent by a host. It is used to inform the device that the host has initiated the Status stage of a control transfer. This TP shall only be sent to a Control endpoint. Only the fields that are different from an ACK TP are described in this section.

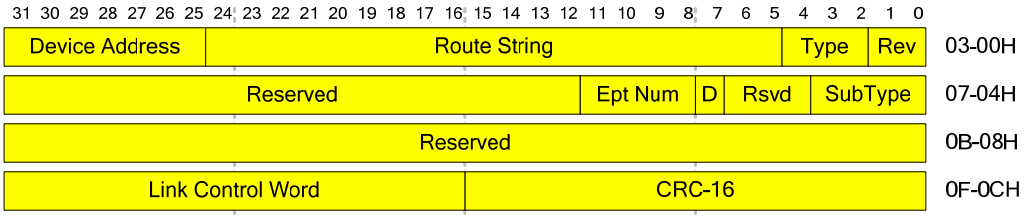


Figure 8-14. STATUS Transaction Packet

Table 8-16. STATUS TP Format (differences with ACK TP)

Width	Offset	Description
4b	32b	**SubType**. This field shall be set to *STATUS* for a STATUS TP.
3b	36b	**Reserved (Rsvd)**. This field is reserved and shall be set to zero.
52b	44b	**Reserved**. This field is reserved and shall be set to zero.

8.5.5 STALL Transaction Packet

This TP can only be sent by a device. It is used to inform the host that an endpoint on the device is halted or that a control transfer is invalid. Only the fields that are different from an ACK TP are described in this section.

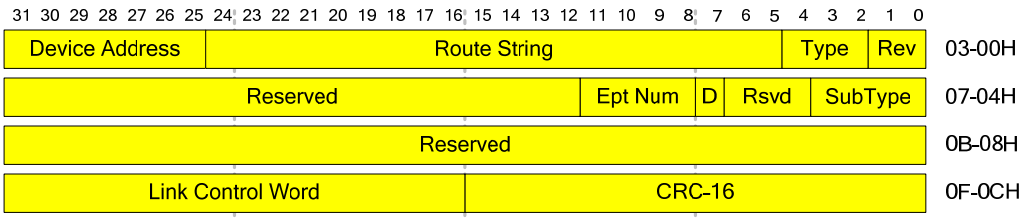


Figure 8-15. STALL Transaction Packet

Table 8-17. STALL TP Format (differences with ACK TP)

Width	Offset	Description
4b	32b	**SubType**. This field shall be set to *STALL* for an STALL TP.
3b	36b	**Reserved (Rsvd)**. This field is reserved and shall be set to zero.
52b	44b	**Reserved**. This field is reserved and shall be set to zero.

8.5.6 Device Notification (DEV_NOTIFICATION) Transaction Packet

This optional TP can only be sent by a device. It is used by devices to inform the host of an asynchronous change in the device/interface state e.g., to identify the function in the device that caused the device to perform a remote wake operation. This TP is not sent from a particular endpoint but from the device in general. Only the fields that are different from an ACK TP are described in this section.

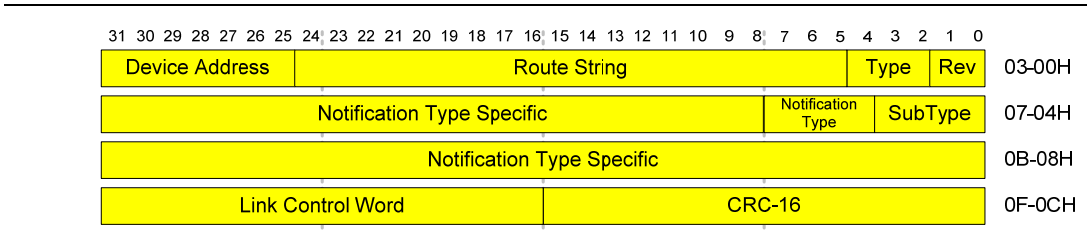


Figure 8-16. Device Notification Transaction Packet

Table 8-18. Device Notification TP Format (differences with ACK TP)

Width	Offset	Description
4b	32b	**SubType**. This field shall be set to *DEV_NOTIFICATION* for a Device Notification TP.
4b	36b	**Notification Type**. The field identifies the type of the device notification **Value** — **Type of Notification Packet** 0000b — Reserved 0001b — FUNCTION_WAKE 0010b — LATENCY_TOLERANCE_MESSAGE 0011b — BUS_INTERVAL_ADJUSTMENT_MESSAGE 0100b – 1111b — Reserved

8.5.6.1 Function Wake Device Notification

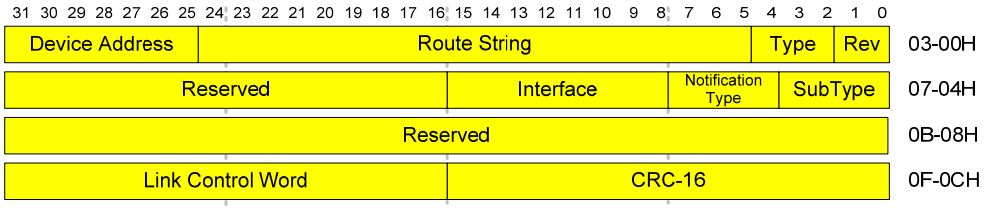


Figure 8-17. Function Wake Device Notification

Table 8-19. Function Wake Device Notification

Width	Offset	Description
4b	32b	**SubType**. This field shall be set to *DEV_NOTIFICATION* for a Device Notification TP.
4b	36b	**Notification Type**. FUNCTION_WAKE
8b	40b	**Interface**. This field identifies the first interface in the function that caused the device to perform a remote wake operation.
48b	48b	**Reserved.** This field is reserved and shall be set to zero.

8.5.6.2 Latency Tolerance Messaging Device Notification

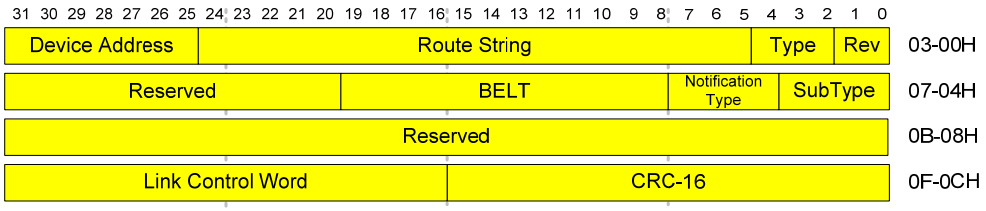


Figure 8-18. Latency Tolerance Messaging Device Notification

Table 8-20. Latency Tolerance Messaging Device Notification

Width	Offset	Description
4b	32b	**SubType**. This field shall be set to *DEV_NOTIFICATION* for a Device Notification TP.
4b	36b	**Notification Type**. LATENCY_TOLERANCE_MESSAGE
12b	40b	**BELT**. The field describes the Best Effort Latency Tolerance value **Bits** — **Description** 9:0 — LatencyValue 11:10 — LatencyScale. 2 bits with the following encoding: **Value** — **Meaning** 00b — Reserved 01b — LatencyValue is to be multiplied by 1024 ns 10b — LatencyValue is to be multiplied by 32,768 ns 11b — LatencyValue is to be multiplied by1,048,576 ns
44b	48b	**Reserved.** This field is reserved and shall be set to zero.

8.5.6.3 Bus Interval Adjustment Message Device Notification

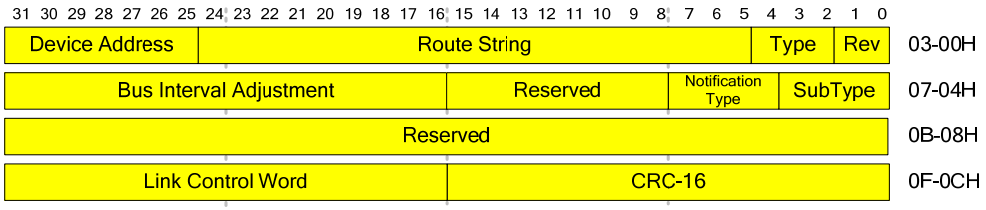


Figure 8-19. Bus Interval Adjustment Message Device Notification

Table 8-21. Bus Interval Adjustment Message Device Notification

Width	Offset	Description
4b	32b	**SubType**. This field shall be set to *DEV_NOTIFICATION* for a Device Notification TP.
4b	36b	**Notification Type**. BUS_INTERVAL_ADJUSTMENT_MESSAGE
8b	40b	**Reserved.** This field is reserved and shall be set to zero.
16b	48b	**Bus Interval Adjustment.** This field is a signed word encoded in two's complement that indicates the Bus Interval Adjustment value as follows: **Value** **Meaning** FFFFH Bus Interval decrease of 1 BusIntervalAdjustmentGranularity unit … 8000H Bus Interval decrease of 32768 BusIntervalAdjustmentGranularity units 7FFFH Bus Interval increase of 32767 BusIntervalAdjustmentGranularity units. … 0000H No change to bus interval The Notification Type field is set to *BUS_INTERVAL_ADJUSTMENT_MESSAGE*.

8.5.6.4 Best Effort Latency Tolerance Messaging

Devices indicate whether they are capable of sending LTM TPs using the **LTM Capable** field in the SuperSpeed_USB Device Capability in the BOS descriptor (see Section 9.6.2 for further information). The BELT applies across an entire device. Devices consider all functions and all endpoints in determining their BELT value, generally taking the smallest one. LTM TPs are only originated by devices, not including hubs. Hubs simply forward LTM TPs toward the host. The LTM Enable (see Section 9.4.10) feature selector is used to enable or disable a device from sending LTM TPs. A device shall send an LTM TP with a value of tBELTdefault in the **BELT** field in response to any change in state of LTM Enable.

The minimum BELT is tBELTmin (see Table 8-34).

In order to avoid LTM TPs impacting hub buffering, devices shall not send more than two LTM TPs within any tBELTRepeat period. The reason to allow up to two per period is that a device may transition to idle and very quickly see new activity. In this case the device may need to send an LTM TP with an active state BELT very quickly after having sent an LTM TP with an idle state BELT.

Prior to sending an LTM TP, a device's default BELT is assumed by the system to be tBELTdefault (see Table 8-34).

The BELT value represents a best effort amount of time between when the host receives an ERDY, and when the host responds by initiating transactions to the endpoint associated with that ERDY. It is important to stress that the BELT is best effort, and strictly for the purpose of power management. LTM TPs apply to all endpoint types except for isochronous endpoints. For interrupt endpoints the subscribed service interval applies at all times except after receipt of an ERDY from

the endpoint, in which case the BELT supersedes the subscribed service interval (if and only if the BELT value is greater than the service interval) but only for the first transaction following the ERDY.

If a device allows its link to enter U1, it shall subtract the U1 System Exit Latency (U1SEL) from its total latency tolerance and send the remaining value as the **BELT** field value in an LTM TP. If a device allows its link to enter U2, it shall subtract the U2 System Exit Latency (U2SEL) from its total latency tolerance and send the remaining value as the **BELT** field value in an LTM TP.

If a device's idle state BELT will become less than tBELTdefault as a result of allowing its link to enter U2, the device shall not enable U2. A device shall not allow its link to enter U2 unless its last reported BELT is tBELTdefault or larger.

IMPLEMENTATION NOTE

Practical Use of LTM with U1 and U2

It would be impractical for a device to send an LTM every time U1 or U2 were entered or exited.

For U1 a simple implementation could always subtract U1SEL prior to sending any LTM. The device could then allow its link to enter and exit U1 at any time without having to send an LTM to update its BELT.

For U2 the U2SEL may overshadow a device's active state BELT. For example, if a device's active state BELT is 200 µs, subtracting a millisecond range U2SEL value would overshadow the BELT. In this case the device may wish to dynamically disable U2 entry while in an active state, so that it does not need to subtract off U2SEL from its BELT while active, and then dynamically re-enable U2 when it returns to an idle state. Additionally, drivers for such devices may wish to disable direct U1 to U2 transitions, but still allow U2 to be initiated by the device, by programming the **U2Timeout** field to 0xFF.

8.5.6.5 Bus Interval Adjustment Messaging

This optional device notification may be sent by a device to request an increase or decrease in the length of the bus interval. This would typically be used by a device trying to synchronize the USB host bus interval clock with an external clock. Bus interval adjustment requests are relative to the current bus interval. For example, if a device requests an increase of one BusIntervalAdjustmentGranularity unit and then later requests an increase of two BusIntervalAdjustmentGranularity units the overall increase by the host would be three BusIntervalAdjustmentGranularity units from the default bus interval length.

The host shall support adjustments through an absolute range of -37268 to 37267 BusIntervalAdjustmentGranularity units. A device shall not request adjustments more than once every 8 bus intervals. A device shall not send a bus interval adjustment request until it has waited long enough to accurately observe the result of a previous bus interval adjustment request. A device shall not make a single BusIntervalAdjustment request for more than +/- 4096 units. A device may make multiple BusIntervalAdjustment requests over time for a combined total of more than 4096 units. A device shall not request a bus interval adjustment unless the device received an

isochronous timestamp within the past 125 μs and the isochronous timestamp contained a **Bus Interval Adjustment Control** field with a value equal to zero or the device's address.

Only one device can control the bus interval length at a time. A host controller implements a first come first serve policy for handling bus interval adjustment requests as described in this section. When the host controller begins operation it shall transmit ITPs with the **Bus Interval Adjustment Control** field set to zero. When the host controller first receives a bus interval adjustment control request, it shall set the **Bus Interval Adjustment Control** field in subsequent ITPs to the address of the device that sent the request. The host shall ignore bus interval adjustment requests from all other devices once the **Bus Interval Adjustment Control** field is set to a non-zero address. If the controlling device is disconnected the host controller shall reset the **Bus Interval Adjustment Control** field to zero. A host controller may provide a way for software to override default bus interval adjustment control field behavior and select a controlling device. A host controller shall begin applying bus interval adjustments within two bus intervals from when the bus interval adjustment request is received.

The smallest bus interval adjustment (1 BusIntervalAdjustmentGranularity) requires the host to make an average adjustment of 8 high speed bit times every 4096 bus intervals. A host is allowed to make this adjustment in a single bus interval such that the clock used to generate ITP times and bus interval boundaries does not need a period smaller than 8 high speed bit times. A host shall make bus interval adjustments at regular intervals. When a host is required to make an average of one or more 8 high speed bit time adjustments every 4096 bus intervals the adjustments shall be evenly distributed as defined by the following constraints:

- Intervals that contain one more 8 high speed bit time adjustment than other intervals are referred to as maximum adjustment bus intervals.
- The number of 8 high speed bit time adjustments made in any bus interval shall not be more than one greater than the number of high speed bit time adjustments made in any other bus interval.
- The distance in bus intervals between consecutive maximum adjustment bus intervals shall not vary by more than one bus interval.

The even distribution and average adjustment requirements for bus interval adjustments shall apply from one bus interval after a bus interval adjustment request is received by the host until the bus interval where a subsequent valid bus interval adjustment request is received by the host.

The following is an example of valid host behavior for a specific bus interval adjustment request. After power on, the host receives a bus interval adjustment request for a bus interval decrease of 10 BusIntervalAdjustmentGranularity units in bus interval X-1. The host controller uses a clock with a period of 8 high speed bit times to drive a counter that produces timestamps and bus interval boundaries. The host controller adds an extra 8 high speed bit time clock tick to its counter in each of the following bus intervals: X+409, X+819, X+1228, X+1638, X+2048, X+2457, X+2867, X+3276, X+3686, X+4096, X+4505,…

8.5.7 PING Transaction Packet

This TP can only be sent by a host. It is used by the host to transition all links in the path to a device back to U0 prior to initiating an isochronous transfer. Please refer to the Power Management appendix for details on the usage of this TP. Only the fields that are different from an ACK TP are described in this section.

The device shall respond to the PING TP by sending a PING_RESPONSE TP (see Section 8.5.8) to the host within tPingResponse (see Table 8-34).

The device shall keep its link in U0 until it receives a subsequent packet from the host, or until tPingTimeout (see Table 8-34) elapses.

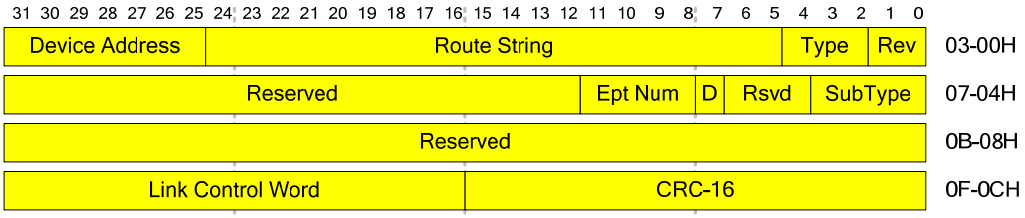


Figure 8-20. PING Transaction Packet

Table 8-22. PING TP Format (differences with ACK TP)

Width	Offset	Description
4b	32b	**SubType**. This field shall be set to *PING* for a PING TP.
3b	36b	**Reserved (Rsvd)**. This field is reserved and shall be set to zero.
52b	44b	**Reserved**. This field is reserved and shall be set to zero.

8.5.8 PING_RESPONSE Transaction Packet

This TP can only be sent by a device. A device sends this TP in response to a PING TP sent by a host. Please refer to the Power Management appendix for details on the usage of this TP. Only the fields that are different from an ACK TP are described in this section.

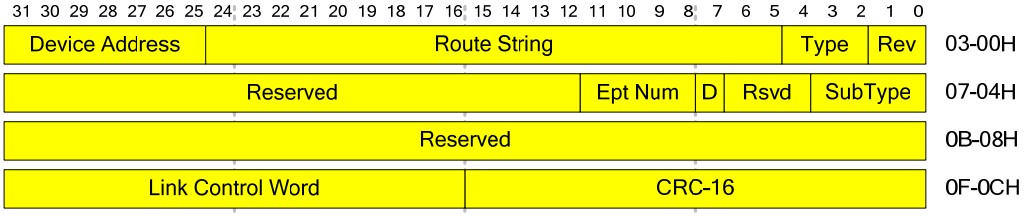


Figure 8-21. PING_RESPONSE Transaction Packet

Table 8-23. PING_RESPONSE TP Format (differences with ACK TP)

Width	Offset	Description
4b	32b	**SubType**. This field shall be set to *PING_RESPONSE* for a PING_RESPONSE TP.
3b	36b	**Reserved (Rsvd)**. This field is reserved and shall be set to zero.
52b	44b	**Reserved**. This field is reserved and shall be set to zero.

8.6 Data Packet (DP)

This packet can be sent by either a host or a device. A host uses this packet to send data to a device. Devices use this packet to return data to a host in response to an ACK TP. Data Packets start with the first four fields used by Transaction Packets. The additional information present in a Data Packet is defined in this section. All Data packets are comprised of a Data Packet Header and a Data Packet Payload. Only the fields that are different from an ACK TP are described in this section.

Data packets traverse a direct path between the host and device. Note it is permissible to send a Data Packet with a zero length data block; however it shall have a CRC-32.

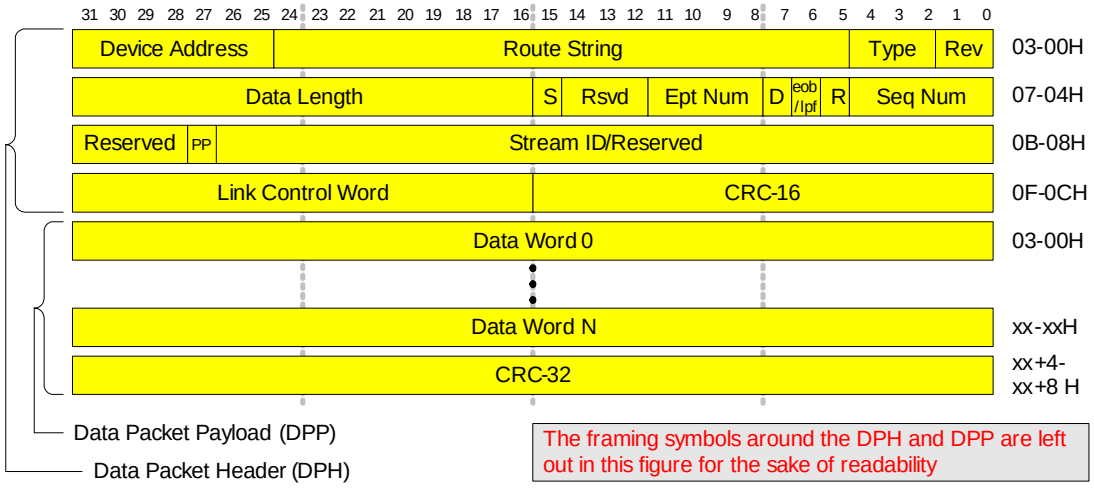


Figure 8-22. Example Data Packet

Table 8-24. Data Packet Format (differences with ACK TP)

Width	Offset	Description
5b	32b	**Sequence Number (Seq Num)**. This field is used to identify the sequence number of the DP. Note that the sequence number wraps around at 31.
1b	37b	**Reserved (R)**. This field is reserved and shall be set to zero.
1b	38b	**End Of Burst (eob)/Last Packet Flag (lpf)**. For non-isochronous endpoints this field is referred as eob and for isochronous endpoints this field is referred as lpf. For non-isochronous endpoints this field is used to identify that this is the last packet of a burst. Setting eob to one is equivalent to sending an NRDY. When the device is ready to continue the transfer it shall send a Notification (Endpoint Ready) to signal the host. eob can be set by a device only and the value of this field is ignored by the device in DPs sent by a host. For control type endpoints eob shall be set to zero For isochronous endpoints this field is used to identify that this is the last packet of the last burst in the current service interval. lpf can be set by device and host. Please refer to Section 8.12.6 for the usage of this field when the target/source of this DP is an isochronous endpoint.
4b	40b	**Endpoint Number (Ept Num)**. This field determines the Endpoint within the device that is the recipient of this TP. SuperSpeed USB devices may support up to a maximum of 15 IN and OUT endpoints apart from the required default control endpoint with an Ept Num set to zero.
3b	44b	**Reserved (R)**. This field is reserved and shall be set to zero.
1b	47b	**Setup (S)**. This field is set by a host to indicate this Data Packet is a Setup packet. This field can only be set by a host.
16b	48b	**Data Length**. This field is used to indicate the number of bytes in the Data Payload excluding the Data CRC-32.
32b	xxb	**CRC-32**. The data CRC is a polynomial applied over the Data block of the data payload. Refer to Section 7.1.1.2.1 for the polynomial used to calculate this value.

8.7 Isochronous Timestamp Packet (ITP)

An ITP is identified by the **Type** field set to *Isochronous Timestamp Packet* and is used to deliver a timestamp from the host to a device. However, ITPs carry no addressing or routing information and as such are not routable. ITPs are forwarded (multicast) by hubs to all downstream ports with links in the U0 state. A device receiving an ITP shall process it like any other header packet at the link level. A device shall not respond to an ITP at the protocol level. ITPs are used to provide host timing information to devices for synchronization. Note that any device or hub may receive an ITP. A host shall transmit an ITP on a root port link if and only if the link is already in U0. Only a host shall initiate an ITP transmission. A host shall not bring a root port link to U0 for the purpose of transmitting an ITP. A host shall transmit an ITP in every bus interval within tTimestampWindow from a bus interval boundary if the root port link is in U0. A host shall begin transmitting ITPs within tIsochronousTimestampStart from when the host root port's link enters U0 from the polling state. An ITP may be transmitted in between packets in a burst. If a device receives an ITP with the delayed flag (DL) set in the link control word the timestamp value may be significantly inaccurate and may be ignored by the device.

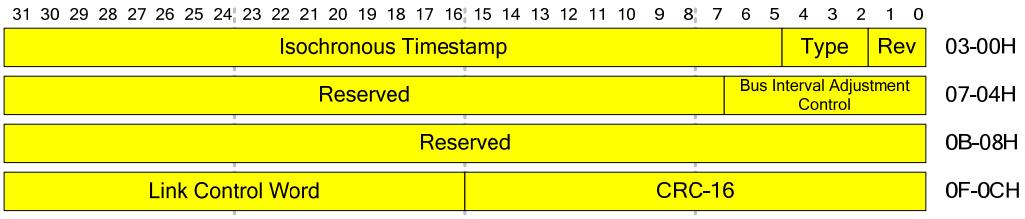


Figure 8-23. Isochronous Timestamp Packet

Table 8-25. Isochronous Timestamp Packet Format

Width	Offset	Description
27b	5b	**Isochronous Timestamp (ITS)**. The isochronous timestamp packet is used to identify the current time value from the perspective of the host transmitting the ITP. The timestamp is split into two fields: **Bits** — **Description** 13:0 — Bus interval counter. The current 1/8 of a millisecond counter. The count value rolls over to zero when the value reaches 0x3FFF and continues to increment. 26:14 — The time delta from the start of the current ITP packet to the previous bus interval boundary. This value is a number of tIsochTimestampGranularity units. The value used shall specify the delta that comes closest to the previous bus interval boundary without going before the boundary. Note: If a packet starts exactly on a bus interval boundary the delta time is set to 0.
7b	32b	**Bus Interval Adjustment Control**. This field specifies the device, via its address, that is in control of the bus interval adjustment mechanism. Upon reset, power-up, or if the device is disconnected, a host shall set this field to zero.
57b	32b	**Reserved**. This field is reserved and shall be set to zero.

The value of the Isochronous Timestamp field in the ITP shall be within +/- 1 tIsochTimestampGranularity units of the value of the host clock (for ITP generation) measured when the first framing symbol of the ITP is transmitted by the host.

8.8 Addressing Triple

Data Packets and most Transaction Packets provide access to specific data flow using a composite of three fields. They are the **Device Address**, the **Endpoint Number**, and the **Direction** fields.

8.9 Route String Field

The **Route String** is a 20-bit field in every packet header the hub uses to direct a downstream flowing packet to the designated downstream port. It explicitly routes a packet on the direct path between the host and device. It is composed of a concatenation of the downstream port numbers (4 bits per hub) for each hub traversed to reach the device. The hub uses a Hub Depth value left shifted by two as an offset into the Route String to locate the bits it uses to determine the

downstream port number. The Hub Depth value is determined and assigned to every hub during the enumeration process.

Note that this field is only valid in packets sent by the host. Devices shall set the **Route String** to zero in any packet that contains a route string that it sends to the host.

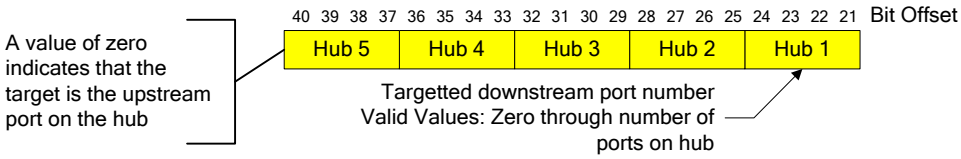


Figure 8-24. Route String Detail

8.9.1 Route String Port Field

This 4-bit wide field in the Route String represents the port in the hub being addressed.

8.9.2 Route String Port Field Width

The Route String Port field width is fixed at 4-bits, limiting the maximum number of ports a hub may support to 15. The upstream port of the hub is accessed by a Route String Port field value of zero and the hub's downstream ports are numbered sequentially starting at 1. The maximum value that may be legally set in this field is determined by the number of downstream ports on the hub.

8.9.3 Port Number

The specific port on a hub that the packet is directed at is identified by the value in the Route String Port field. The hub's upstream port is always represented by a zero while the downstream ports begin with one and count up sequentially.

8.10 Transaction Packet Usages

TPs are used to report the status of data transactions and can return values indicating successful reception of data packets, command acceptance or rejection, flow control, and halt conditions. Control, Bulk and Interrupt data transfer types have guaranteed delivery and return TPs. Devices and hosts do not return TPs for Isochronous transfers.

8.10.1 Flow Control Conditions

Only Bulk, Control and Interrupt endpoints may send flow control responses. Isochronous endpoints cannot send flow control responses.

An IN endpoint shall be considered to be in a flow control condition if it returns one of the following responses to an ACK TP:

- responding with an NRDY TP
- sending a DP with the **EOB** field set to 1 in the DPH.

An OUT endpoint shall be considered to be in a flow control condition if it returns one of the following responses to a DP:

- responding with an NRDY TP
- sending an ACK TP with the **NumP** field set to 0.

The **Packets Pending** field is only valid when set by a host and does no effect whether or not an endpoint enters the flow control state.

When an endpoint is in a flow control condition it shall send an ERDY TP to be moved back into the active state. When an endpoint is not in a flow control condition it shall not send an ERDY TP unless the endpoint is a Bulk endpoint that supports streams. Refer to Section 8.12.1.4.2 and Section 8.12.1.4.3 for further information when a Bulk endpoint that supports streams can send an ERDY TP. Note that a host may resume transactions to any endpoint – even if the endpoint had not returned an ERDY TP after returning a flow control response.

8.10.2 Burst Transactions

The SuperSpeed USB protocol allows a host to continually send data to a device or receive data from a device as long as the device can receive the data or transmit the data. The number of packets an endpoint on a device can send or receive at a time without an intermediate acknowledgement packet is reported by the device in the endpoint companion descriptor (see Section 9.6.7) for that endpoint. An endpoint that reports more than one packet in its Maximum Burst capability is considered to be able to support ‘Burst’ Transactions.

While bursting the following rules apply:

- The maximum number of packets that can be sent in a burst without an explicit acknowledgement is limited to the Maximum Burst capability of the endpoint.
- Each individual packet in the burst shall have a Data Payload of maximum packet size. Only the last packet in a Burst may be of a size smaller than the reported maximum packet size. If the last one is smaller then the same rules for short packet apply to a short packet at the end of a burst (See Section 8.10.3).
- The burst transaction continues as long as the **NumP** field in the ACK TP is not set to zero.
- The **NumP** field can be incremented at any time by the device or host sending the ACK TP as long as the device or host wants to continue receiving data. The only requirement is that the **NumP** field shall not have a value greater than the maximum burst supported by the device.

- A device or host sending an ACK TP can only decrement the **NumP** field by one packet in each ACK TP it sends. E.g. if the previous ACK TP had a value of five in **NumP** field then the next ACK TP to acknowledge the next packet received may have a value of four in **NumP** field. The only exceptions to this rule are:
  - — If the device can receive the data but cannot accept any more data then it shall send an ACK TP with the **NumP** field set to zero
  - — The host may send an ACK TP with **NumP** field set to zero in response to a device sending a DP with the **EOB** field set or with a length shorter than the maximum packet size.

8.10.3 Short Packets

SuperSpeed USB retains the semantics of short packet behavior that USB 2.0 supported. When a host or device receives a DP with the **Data Length** field shorter than the maximum packet length for that endpoint it shall deem that that transfer is complete.

In the case of an IN transfer, the device shall stop sending DPs after sending a short DP. The host shall respond to the short DP with an ACK TP with the **NumP** field set to zero. The host shall resume transactions to the endpoint on the device when a new transfer is posted for that endpoint.

In the case of an OUT transaction the host may stop sending DPs after sending a short DP. The host can resume transactions to the endpoint on the device when a new transfer is posted for that endpoint.

8.11 TP Responses

Transmitting and receiving devices shall return TPs as detailed in Table 8-26 through Table 8-28. Not all TPs are allowed, depending on the transfer type and whether the TP is being issued by a device or by the host.

8.11.1 Device Response to TP Requesting Data

Table 8-26 shows the possible responses a device may make in response to a TP requesting data for non-isochronous endpoints. If the device is unable to send data, due to a halt or a flow control condition, it issues a STALL or NRDY TP, respectively. If the device is able to issue data and the Deferred bit is not set in the TP, it does so. If the received TP has invalid values (incorrect Device Address or the endpoint number and direction does not refer to an endpoint that is part of the current configuration or an invalid sequence number), the device shall not return a response.

Table 8-26. Device Responses to TP Requesting Data (non-isochronous endpoints)

Invalid TP Received	TP Received with Deferred Bit Set	Device Tx Endpoint Halt Feature Set	Device Ready to Transmit Data	Action Taken
Yes	Do not care	Do not care	Do not care	The device shall ignore the TP
No	Yes	Yes	Do not care	The device shall send and ERDY TP
No	Yes	No	No	The device shall not respond. It shall send an ERDY TP when it is ready to resume.

Invalid TP Received	TP Received with Deferred Bit Set	Device Tx Endpoint Halt Feature Set	Device Ready to Transmit Data	Action Taken
No	Yes	No	Yes	The device shall send an ERDY TP indicating that it is ready to send data
No	No	Yes	Do not care	Issue STALL TP
No	No	No	No	Issue NRDY TP
No	No	No	Yes	Start transmitting DPs with sequence numbers requested by host

8.11.2 Host Response to Data Received from a Device

Table 8-27 shows the host responses to data received from a device for non-isochronous endpoints. The host is able to return only one type of TP: ACK. If the host receives a DP with a DPH that has invalid values (unknown Device Address or the endpoint number and direction does not refer to an endpoint that is part of the current configuration or an invalid sequence number) then host shall ignore the packet. If the host receives a corrupted DP or the host cannot accept data from a device (due to problems such as internal buffer overrun), it discards the data and will send a request for the DP again. If the host is able to accept data and the DP is received error-free, the host accepts the data and issues an ACK TP.

Table 8-27. Host Responses to Data Received from a Device (non-isochronous endpoints)

DPH has Invalid Values	Data Packet Payload Corrupted[1]	Host Can Accept Data	TP Returned by Host
Yes	Do not care	Do not care	Discard data and do not send any TP
No	Yes	Do not care	Discard data; send an ACK TP with the **Retry** bit set requesting for one or more DPs starting from the sequence number of the packet that was corrupted.
No	No	No	Discard data; send an ACK TP with the **Retry** bit set requesting for one or more DPs starting from the sequence number of the packet that the host was unable to receive. The ACK TP shall have the **Host Error** bit set to one to indicate that the host was unable to accept the data.
No	No	Yes	Accept data; send an ACK TP requesting for zero or more DPs starting from the sequence number of the next packet expected. This is also an implicit acknowledgement that this packet was received successfully.

Note:

1. Either the DPP is corrupted or the DPP was not received after receiving the DPH.

8.11.3 Device Response to Data Received from the Host

TP responses by a device to data received from the host for non-isochronous endpoints are shown in Table 8-28. Assuming successful data packet decode, a device, upon receiving a DP, may return any one of the three TPs. If the DP was corrupted, the device returns an ACK TP with **Retry** bit set indicating that it did not receive that DP and informing the host of the data buffers it has available to receive DPs from the host. If the DP was received error-free and the endpoint on the device is halted, the device returns a STALL TP. If the device receives a DP with a DPH that has invalid values (incorrect Device Address or the endpoint number and direction does not refer to an endpoint that is part of the current configuration or an invalid sequence number) then the device shall ignore the packet. If the device can accept the data and has received the DP error-free, it returns an ACK TP. If the device cannot accept the data packet due to flow control reasons, it returns an NRDY TP.

Note receipt of an ACK TP indicates to the host the DP with the previous sequence number was successfully received by the device as well as the number of data packet buffers the device has available to receive any pending DPs the host has. The device shall send an ACK TP for each DP received.

Table 8-28. Device Responses to OUT Transactions (non-isochronous endpoints)

DPH has Invalid Values	DPH has Deferred Bit Set	Receiver Halt Feature Set	Data Packet Payload Corrupted[1]	Device Can Accept Data	TP Returned by Device
Yes	Do not care	Do not care	Do not care	Do not care	Discard DP
No	Yes	Yes	Do not care	Do not care	The device shall send an ERDY TP
No	Yes	No	Do not care	No	The device shall not respond. It shall send an ERDY TP when it is ready to resume
No	Yes	No	Do not care	Yes	The device shall send an ERDY TP
No	No	Yes	Do not care	Do not care	The device shall send a STALL TP
No	No	No	Yes	Do not care	Discard DP, send an ACK TP with the sequence number of the DP expected (thereby indicating that the DP was not received), the **Retry** bit set and the number of DPs that the device can receive for this endpoint.

DPH has Invalid Values	DPH has Deferred Bit Set	Receiver Halt Feature Set	Data Packet Payload Corrupted[1]	Device Can Accept Data	TP Returned by Device
No	No	No	No	Yes	Send an ACK TP indicating the sequence number of the next DP expected (thereby indicating that this DP was received successfully) and the number of DPs that the device can receive for this endpoint.
No	No	No	No	No	Discard DP, send an NRDY TP

[1] Either the DPP is corrupted or the DPP was not received after receiving the DPH.

8.11.4 Device Response to a SETUP DP

A SETUP DP is a special DP that is identified by the **Setup** field set to one and addressed to any control endpoint. SETUP is a special type of host-to-device data transaction that permits the host to initiate a command that the device shall perform. Upon receiving a SETUP DP, a device shall respond as shown in Table 8-29.

Note a SETUP DPH shall be considered invalid if it has any one of the following:

- incorrect Device Address
- endpoint number and direction does not refer to an endpoint that is part of the current configuration
- endpoint number does not refer to a control endpoint
- non-zero sequence number

Table 8-29. Device Responses to SETUP Transactions (only for control endpoints)

DPH has Invalid Values	DPH has Deferred Bit Set	Data Packet Payload Corrupted[1]	TP Returned by Device
Yes	N/A	N/A	Discard DP
No	Yes	N/A	The device shall send an ERDY TP indicating that it is ready to receive the SETUP DP.
No	No	Yes	Discard SETUP DP, send an ACK TP with the sequence number set to zero, the rty bit set and the NumP field set to one.
No	No	No	Send an ACK TP with the sequence number set to one (thereby indicating that this SETUP DP was received successfully) and the NumP field set to one.

[1] Either the SETUP DPP is corrupted or the SETUP DPP was not received after receiving the SETUP DPH.

8.12 TP Sequences

The packets that comprise a transaction vary depending on the endpoint type. There are four endpoint types: bulk, control, interrupt, and isochronous.

8.12.1 Bulk Transactions

Bulk transaction types are characterized by the ability to guarantee error-free delivery of data between the host and a device by means of error detection and retry. Bulk transactions use a two-phase transaction consisting of TPs and DPs. Under certain flow control and halt conditions, the data phase may be replaced with a TP.

8.12.1.1 State Machine Notation Information

This section shows detailed host and device state machines required to advance the Protocol on an IN or OUT pipe. The diagrams should not be taken as a required implementation, but to specify the required behavior.

Figure 8-25 shows the legend for the state machine diagrams. A circle with a three line border indicates a reference to another (hierarchical) state machine. A circle with a two line border indicates an initial state. A circle with a single line border is a simple state.

A diamond (joint) is used to join several transitions to a common point. A joint allows a single input transition with multiple output transitions or multiple input transitions and a single output transition. All conditions on the transitions of a path involving a joint must be true for the path to be taken. A path is simply a sequence of transitions involving one or more joints.

A transition is labeled with a block with a line in the middle separating the (upper) condition and the (lower) actions. The condition is required to be true to take the transition. The actions are performed if the transition is taken. The syntax for actions and conditions is VHDL. A circle includes a name in bold and optionally one or more actions that are performed upon entry to the state.

Transitions using a solid arrow are generated by the host. Transitions using a dashed arrow are generated by the device. Transitions using a dot-dot-dash arrow are generated by the either the device or the host.

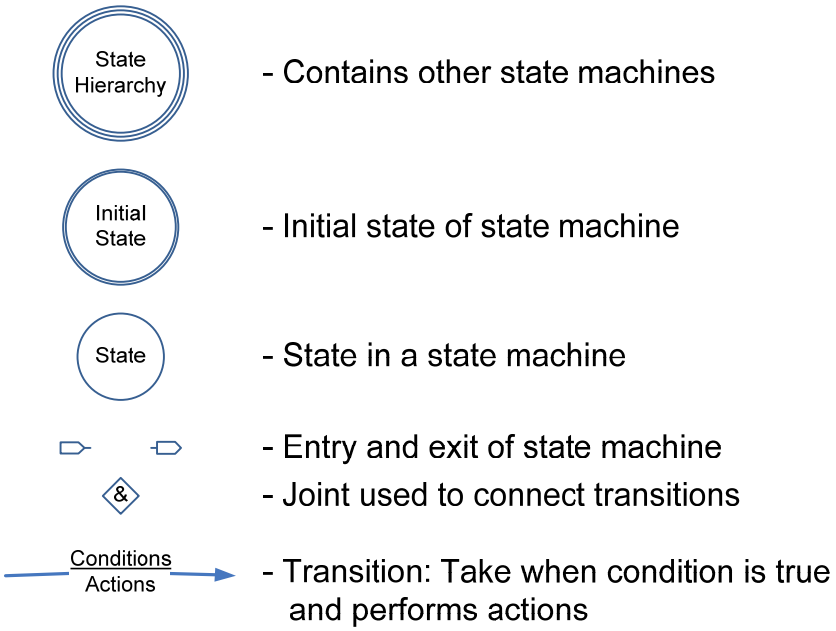


Figure 8-25. Legend for State Machines

8.12.1.2 Bulk IN Transactions

When the host is ready to receive bulk data, it sends an ACK TP to the device indicating the sequence number and number of packets it expects from the device. If the **Deferred** bit is not set in the ACK TP then the targeted endpoint on the device responds by returning either the DPs requested by the host or, should it be unable to return data, an NRDY or STALL TP. An NRDY TP indicates that the device is temporarily unable to return data, while a STALL TP indicates that the endpoint is permanently halted and requires USB System Software intervention. If the **Deferred** bit is set in the ACK TP then the device shall send an ERDY TP to indicate to the host that the endpoint is ready to transmit data.

The host will send an ACK TP for each valid DP it receives from the device. A device does not need to wait for the ACK TP to send the next DP to the host if the previous ACK TP indicated that the host expected the device to send more than one DP (depending on the value of the **Number of Packets** field in the TP). The ACK TP implicitly acknowledges the last DP with the previous sequence number as being successfully received by the host and also indicates to the device the next DP with the sequence number and number of packets the host expects from the device. If the host detects an error while receiving any of the DPs, it will send an ACK TP with the sequence number value set to the first DP that was received with an error with the Retry bit set, even if subsequent packets in the burst asked for by the host were received without error. The device is required to re-send all DPs starting from the sequence number set in the ACK TP in which the Retry bit set.

The host expects the first DP to have a sequence number set to zero when it starts the first transfer from an endpoint after the endpoint has been initialized (via a Set Configuration or Set Interface or a ClearFeature (STALL) command – See the Framework chapter for details on these commands). The second DP sent by the device from that endpoint shall have a sequence number set to one; the third DP has a sequence number set to two and so on till sequence number thirty one. The next DP

after sequence number thirty one uses a sequence number of zero. The device keeps incrementing the sequence number of the packets it transmits unless it receives an ACK TP with the Retry bit set to one that indicates that it has to retransmit an earlier DP.

If the host asks for multiple DPs from the device and the device does not have that number of DPs available to send at the time, the device shall send the last DP with the **End Of Burst** Flag in the DPH set to one. Note that it is not necessary to set the **End Of Burst** flag if the DP sent to the host has a payload that is less than the MaxPacketSize for that endpoint.

A transfer is complete when the device sends all the data that is expected by the host or it sends a DP with a payload that is less than the MaxPacketSize. When the host wants to start a new transfer it shall send another ACK TP with the next sequence number and number of DPs expected from the device. E.g., if the DP with the payload less than MaxPacketSize was 2, the host will initiate the next transfer by sending an ACK TP with the expected sequence number set to 3.

8.12.1.3 Bulk OUT Transactions

When the host is ready to transmit bulk data, it sends one or more DPs to the device. If DPH with valid values (valid device address, endpoint number and direction as well as the expected sequence number) is received by the device, it will return one of following three TPs:

- An ACK TP with zero in the **Retry** bit indicates that the DP was received without error and informs the host about the amount of buffering it has in terms of number of DPs via the **Number of Packets** field in the ACK TP. A host does not need to wait for each ACK TP to send the next DP to the device if the previous ACK TP indicated that the device had buffering available for more than one DP. If there was an error detected in the Data Packet payload then the sequence number in the ACK TP would indicate the sequence number of the DP received in error and the Retry bit will be set in that ACK TP. If the device cannot accept any more DPs from the host then it shall sent the **NumP** field in the ACK TP to zero and it shall send an ERDY TP when it is ready to receive DPs from the host.
- An NRDY TP indicates that the DP was received without error but that the host should resend the DP at a later time because the device was in a temporary condition preventing it from accepting the DP (e.g., buffer full). When the device is ready to accept data, it will send an ERDY TP to the host indicating the number of DPs it has buffer space available for.
- If the **Deferred** bit is set in the DPH, then the device shall send and ERDY TP when its ready to accept data from the host.
- If the endpoint was halted, a STALL TP is returned to indicate that the endpoint is permanently halted and requires USB System Software intervention.

The host always initializes the first DP sequence number to zero in the first transfer it performs to an endpoint after the endpoint is initialized (via a Set Configuration or Set Interface or ClearFeature (STALL) command – See the Framework Chapter for details on these commands). The second DP has a sequence number set to one; the third DP has a sequence number set to two and so on till thirty one. The next DP after sequence number thirty one uses a sequence number of zero. The host keeps incrementing the sequence number of the DPs it transmits unless it receives an ACK TP with the Retry bit set to one that indicates that it has to retransmit an earlier packet.

A transfer is complete when the host sends all the data it has to the device; however the last DP of the transfer may or may not have a payload which is equal to the MaxPacketSize of the endpoint. When the host wants to start a new transfer it shall send another DP, with the next sequence number, targeted at the endpoint in the device.

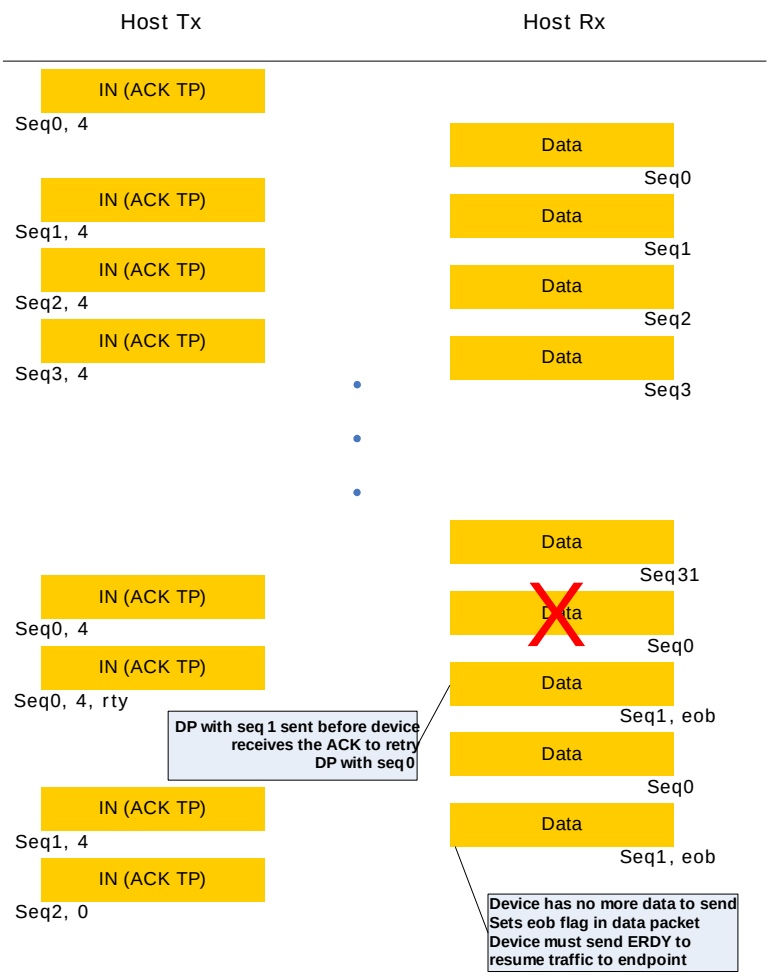


Figure 8-26. Sample BULK IN Sequence

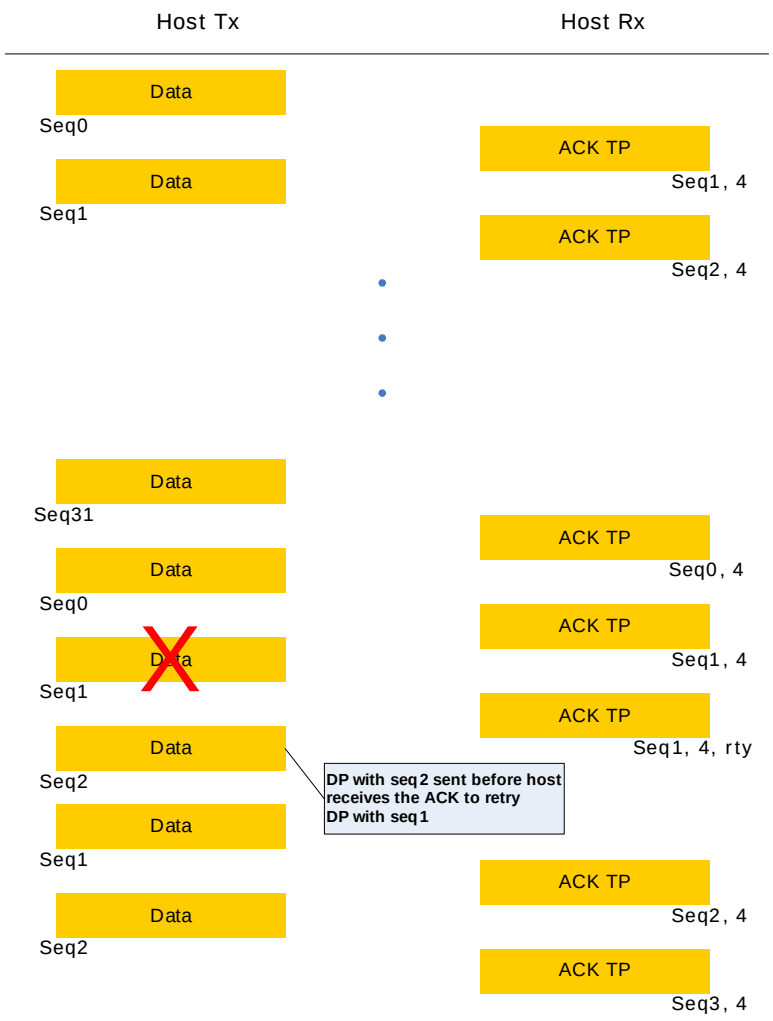


Figure 8-27. Sample BULK OUT Sequence

8.12.1.4 Introduction to Bulk Streaming Protocol

The Stream Protocol adheres to the semantics of the standard SuperSpeed Bulk protocol, so the packet exchanges on a SuperSpeed bulk pipe that supports Streams are indistinguishable from a SuperSpeed bulk pipe that doesn't. The Stream Protocol is managed strictly through manipulation of the *Stream ID* field in the packet header.

Note: As described in this section, the Stream Protocol applies to the state of the "pipe" and is described as single entity. In reality the Stream Protocol is being tracked independently by the host at one end of the pipe and the device at the other. So at any instant in time the two ends may momentarily be out of phase due to packet propagation delays between the host and the device.

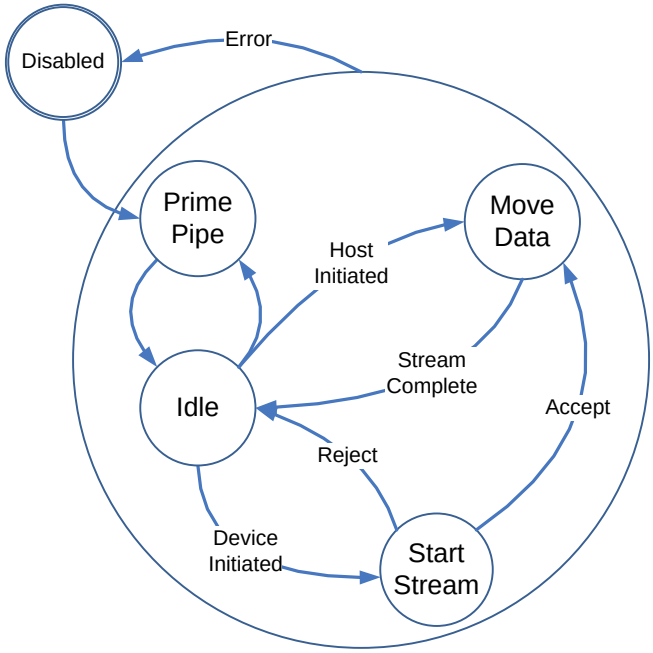


Figure 8-28. General Stream Protocol State Machine (SPSM)

Figure 8-28 illustrates the basic state transitions of the Stream Protocol State Machine (SPSM). This section describes the general transitions of the SPSM as they apply to both IN and OUT endpoints. Detailed operation of the SPSM for IN and OUT endpoints is described in subsequent sections.

Disabled – This is the initial state of the pipe after it is configured, as well as the state that is transitioned to if an error is detected in any of the other states. The first time an Endpoint Buffer is assigned to the pipe, the host will transition the SPSM to the Prime Pipe state. If the Disabled state was entered due to an error, then the error condition must be removed via software intervention before the state may be exited. Note an error (Stall, timeout, etc.) shall transition any SPSM state to the Disabled state.

Prime Pipe – This state is always initiated by host, and informs the device that an Endpoint Buffer set has been added or modified by software.

Idle – This state indicates that there is no Current Stream selected. In this state, the SPSM is waiting for Prime Pipe or Host Initiated transition to Move Data, or a Device Initiated transition to Start Stream. The object of the Host and Device Initiated transitions is to start a Stream (setting the Current Stream, by the Host or Device, respectively) and begin moving data.

Start Stream – This state is always initiated by Device when it wants to select a Stream and start a data transfer. If the device selected Stream is accepted by the host; the Current Stream is set and the pipe enters the Move Data state. If the device selected Stream is rejected by the host; the pipe returns to Idle state.

Move Data – In this state Stream data is transferred. If this state is entered by a Host Initiated Stream selection the Current Stream will be set by the host. If this state is entered from the Start Stream state, the Current Stream selection will have already been set by the device. The SPSM transitions back to the Idle state when the Stream transfer is complete, or if the host or the device decides to terminate the Stream transfer. The transition to Idle invalidates the Current Stream for the pipe.

8.12.1.4.1 Stream IDs

A 16-bit field *Stream ID* field is reserved in DP headers and in ACK, NRDY, and ERDY TPs for passing SIDs between the host and the device. The bullets below identify specific SID values that are reserved by the Stream Protocol and other SID notations.

- **NoStream** – This SID indicates that no Stream ID is associated with the respective bus packet and the Stream ID field should not be interpreted as referencing a valid Stream. The *NoStream* SID value is FFFFh.
- **Prime** – This SID is used to define transitions into and out of the Prime Pipe state. As with *NoStream*, no Stream ID is associated with the respective bus packet and the Stream ID field should not be interpreted as referencing a valid Stream. The *Prime* SID value is FFFEh.
- **Stream n** – Where n is a value between 1 and 65533 (FFFDh). This notation is used to reference a valid Stream ID. The Stream ID field in the packet header is valid if it uses this notation. Valid *Stream n* SID values are between 1 and 65533 (FFFDh), where the numeric value is identical to *n*.
- **Stream 0** – This value is reserved and not used by a pipe that supports Streams. The *Stream 0* SID value is 0000h. Its use is required by a standard bulk pipe.
- **CStream** represents the value of the "Current" Stream ID assigned to the pipe. A *CStream* value is maintained by both the host and the device. The Stream Protocol ensures that the *CStream* values are consistent in the host and the device. Valid values are *NoStream* or *Stream n*.
- **LCStream** represents the value of the CStream SID assigned to the pipe before the last state transition. An *LCStream* value is maintained by the host. Valid values are *Prime, NoStream* or *Stream n*. e.g. While the pipe in the Move Data state CStream = Stream n, when the pipe transitions from Move Data to Idle state, LCStream is set to *Stream n* and CStream is set to *NoStream*, thus LCStream records the "Last CStream" value.

Stream n SID values are assigned by the host and passed to the device (typically through an out-of-band, Device Class defined method). The value of a *Stream n* SID shall be treated as a "logical value" by a device, i.e. the device should not infer any meaning from the value or modify it.

Note: The Bulk IN and OUT Stream Protocols below describe simplified state machines that do not explicitly detail the burst feature of SS endpoints which allows DPs to be sent without receiving an ACK. An implementation shall extend these state machines to manage bursting.

8.12.1.4.2 Bulk IN Stream Protocol

This section defines the SuperSpeed packet exchanges that transition the Stream Protocol from one state to another on an IN bulk endpoint.

For an IN pipe, Endpoint Buffers in host receive Function Data from the device.

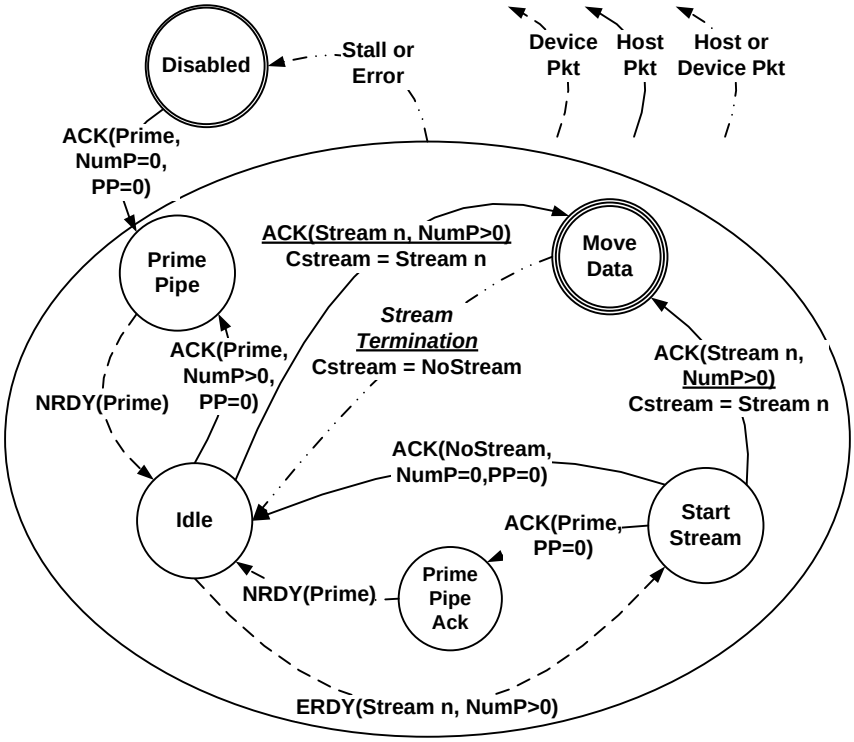


Figure 8-29. Bulk IN Stream Protocol State Machine (ISPSM)

After an endpoint is configured, the pipe is in the **Disabled** state. The host shall transition the pipe to the **Prime Pipe** state by issuing a DP with the Stream ID field set to *Prime*. This transition occurs after Endpoint Buffers are assigned to the pipe by system software.

The device shall cause the pipe to exit the **Prime Pipe** state and transition to the **Idle** state by asserting an NRDY TP with its Stream ID field set to *Prime*.

Note: If an intermediate hub deferred the ACK TP, the host and the device shall act as if the device sent an NRDY TP. i.e. the host shall transition to the **Idle** state when it receives the Deferred Response. The device shall transition to the **Prime Pipe** state when it receives the Deferred ACK TP and then it shall immediately transition to the **Idle** state as if it has sent the NRDY TP with its Stream ID field set to *Prime*.

In the **Idle** state the pipe is waiting for a Stream selection (e.g. a transition to **Start Stream** or **Move Data**) or a notification from the host that an Endpoint Buffer as been added or modified for the pipe (transition to **Prime Pipe**). In the **Idle** state, Stream selection initiated by the host is identified by an ACK TP with its Stream ID set to *Stream n* and a NumP value > 0. This packet shall transition the ISPSM from the **Idle** state to the **Move Data** state. The host shall initiate this transition only if the last ISPSM transition was from **Start Stream** or **Move Data**, and the Endpoint Buffer posted to the pipe was for *LCStream*. In the **Idle** state, Stream selection initiated by the device is identified by an ERDY TP with its Stream ID set to *Stream n* and a NumP value > 0. This packet shall transition the ISPSM from the **Idle** state to the **Start Stream** state. The device shall initiate this transition when it wishes to start a Stream transfer regardless of whether it is in a flow control condition or not.

In the **Start Stream** state the pipe is waiting for the host to accept or reject the Stream selection proposed by the device. The host shall indicate the acceptance of a device initiated Stream selection by asserting an ACK TP with the following field settings; Stream ID = *Stream n* and NumP > 0. This packet shall transition the ISPSM from the **Start Stream** state to the **Move Data** state. The host shall indicate the rejection of a device initiated Stream selection by asserting an ACK TP with the following field settings; Stream ID = *NoStream*, NumP = 0, and Packet Pending (PP) = 0. This packet shall transition the ISPSM from the **Start Stream** state to the **Idle** state. The host shall reject a stream selection if there are no Endpoint Buffers available for the device selected SID.

The ISPSM is executing independently in the host and the device. A race condition occurs if the device issues an ERDY to the host and enters the **Start Stream** state, at the same time that the host issues a ACK(Prime,PP=0) to the device and enters the **Prime Pipe** state. To recover from this condition, if the device receives a ACK(Prime,PP=0) while in the **Start Stream** state it shall transition to the **Prime Pipe Ack** state and issue an NRDY(Prime) to the host, to complete the **Prime Pipe** to **Idle** transition for the host, and the **Prime Pipe Ack** to **Idle** transition for the device.

In the **Move Data** state CStream is set at both ends of the pipe and the pipe is actively moving data to the host. The details of the bus transactions executed in the Move Data state and its exit conditions are defined in the IN Move Data State Machine defined below.

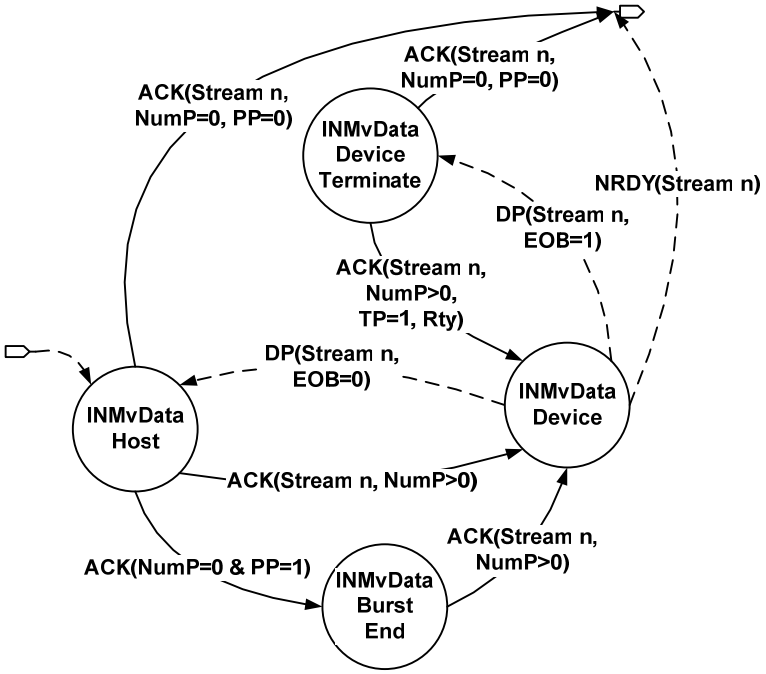


Figure 8-30. IN Move Data State Machine (IMDSM)

The IN Move Data State Machine (IMDSM) is entered from the **Start Stream** or **Idle** states as described above. The entry into the IMDSM immediately transitions to the **INMvData Device** state. The Stream ID field of all packets generated by the IMDSM shall be Stream n.

Each time the **INMvData Device** state is entered, the device performs the following actions to advance the IMDSM.

if (Device Function Data bytes > Max Packet Size)

The device shall generate a DP with EOB field = 0, which shall cause the pipe to transition to **INMvData Host** state.

else (Device Function Data bytes <= Max Packet Size)

The device shall generate a DP with EOB field = 1, which shall cause the pipe to transition to **INMvData Device Terminate** state.

Optionally, the device may generate an NRDY TP with the Stream ID set to *Stream n*, which terminates the stream, and shall cause the pipe to exit the IMDSM and transition to **Idle** state. The use of this transition is Device Class defined.

Note: If an intermediate hub deferred the ACK TP, the host and the device shall act as if the device sent an NRDY TP. i.e. the host shall transition to the **Idle** state when it receives the Deferred Response. The device shall exit the IMDSM and transition to the **Idle** state when it receives the Deferred ACK TP as if it has sent an NRDY TP with its Stream ID field set to *Stream n*. If the device accepts the host initiated Stream ID, it shall send an ERDY with its Stream ID field set to *Stream n*. If the device rejects the host initiated Stream ID, it shall stay the **Idle** state and wait for next Stream selection either by the host or the device.

The **INMvData Host** state is entered because the device has more Function Data to send, so the host performs the following actions to advance the IMDSM.

if (Another DP can be scheduled in this burst)

if (The host has more Endpoint Buffer space available)

Generate an ACK TP with NumP > 0, which shall cause the pipe to transition to **INMvData Device** state.

else the host is out of Endpoint Buffer space.

Generate a Terminating ACK TP with NumP = 0 and PP = 0, which shall cause the pipe to exit the IMDSM and transition to the **Idle** state.

else (the last DP of the burst has just been received)

Terminate the burst.

if (The host has more Endpoint Buffer space available)

Inform the device that the burst is complete (NumP = 0) and another burst shall be scheduled by the host (PP = 1) for CStream. Generate an ACK TP with NumP = 0 and PP = 1, which shall cause the pipe to transition to **INMvData Burst End** state.

else the host is out of Endpoint Buffer space.

Generate a Terminating ACK TP with NumP = 0 and PP = 0, which shall cause the pipe to exit the IMDSM and transition to the **Idle** state.

In the **INMvData Burst End** state, the host shall generate an ACK with NumP > 0 and PP = 1 to initiate the next burst and cause the pipe to transition to the **INMvData Device** state.

The pseudo code describing the IMDSM assumes that the received DP is valid. If it is invalid, an ACK TP shall be generated, which shall transition the pipe to the **INMvData Device** state. The Sequence Number in the ACK TP shall not advance, however the retry will consume a burst opportunity, decrementing transmitted NumP value. If NumP = 0 and there are still Endpoint Buffers available in the host, PP shall be set to 1, otherwise PP is set to 0.

The **INMvData Device Terminate** state is entered because the device has no more Function Data to be sent, so the host shall generate a Terminating ACK TP with NumP = 0 and PP = 0, which shall cause the pipe to exit the IMDSM and transition to the **Idle** state. If the host detects an error on the last DP sent by the device, it shall respond with a Retry ACK TP (Steam n, NumP>0, Rty) and the IMDSM shall transition to the **INMvDataDevice** state.

Note: if a DP error is detected in the **INMvDataHost** state the host shall generate an ACK TP with NumP > 0 and Rty = 1, which shall cause the pipe to transition to **INMvDataDevice** state and resend the packet.

8.12.1.4.3 Bulk OUT Stream Protocol

This section defines the SuperSpeed packet exchanges that transition the Stream Protocol from one state to another on an OUT bulk endpoint.

For an OUT pipe, Endpoint Data in host is transmitted to Function Buffers in the device. And unless otherwise stated, a DP will contain Endpoint Data.

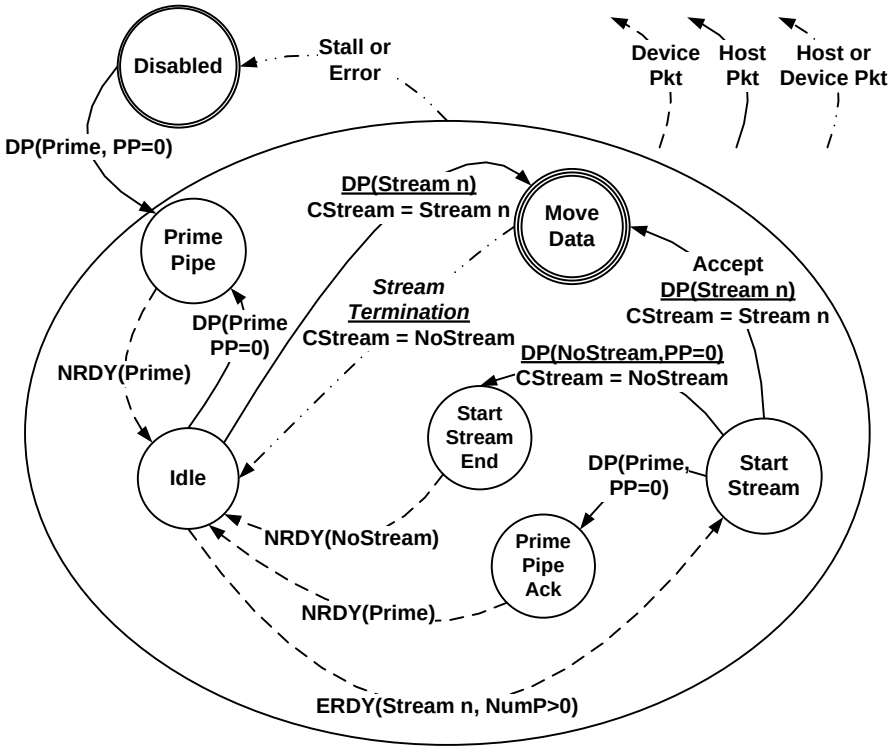


Figure 8-31. OUT Stream Protocol State Machine (OSPSM)

After an endpoint is configured, the pipe is in the **Disabled** state. The host shall transition the pipe to the **Prime Pipe** state by issuing aDP with the Stream ID field set to *Prime*. This transition occurs after Endpoint Buffers are assigned to the pipe by system software.

The device shall cause the pipe to exit the **Prime Pipe** state and transition to the **Idle** state by asserting an NRDY TP with its Stream ID field set to *Prime*.

Note: If an intermediate hub deferred the DP, the host and the device shall act as if the device sent an NRDY TP. i.e. the host shall transition to the **Idle** state when it receives the Deferred Response. The device shall transition to the **Prime Pipe** state when it receives the Deferred DPH. Then it shall immediately transition to the **Idle** state as if it has sent an NRDY TP with its Stream ID field set to *Prime*.

In the **Idle** state the pipe is waiting for a Stream selection (e.g. a transition to **Start Stream** or **Move Data**) or a notification from the host that Endpoint Data has been added or modified for the pipe (transition to **Prime Pipe**). In the **Idle** state, Stream selection initiated by the host is identified by a DP with its Stream ID set to *Stream n*. The value of PP will depend on whether the host has

one (PP = 0) or more (PP = 1) packets to transfer. This packet shall transition the OSPSM from the **Idle** state to the **Move Data** state. The host shall initiate this transition only if the last OSPSM transition was from **Start Stream End** or **Move Data**, and the Endpoint Data posted to the pipe was for *LCStream*. In the **Idle** state, Stream selection initiated by the device is identified by an ERDY TP with its Stream ID set to *Stream n* and a NumP value > 0. This packet shall transition the OSPSM from the **Idle** state to the **Start Stream** state. The device shall initiate this transition when it wishes to start a Stream transfer regardless of whether it is in a flow control condition or not.

In the **Start Stream** state the pipe is waiting for the host to accept or reject the Stream selection proposed by the device. The host shall indicate the acceptance of a device initiated Stream selection by sending a DP with the following field settings; Stream ID = *Stream n*. The value of PP will depend on whether the host has one (PP = 0) or more (PP = 1) packets to transfer. This packet shall transition the OSPSM from the **Start Stream** state to the **Move Data** state. The host shall indicate the rejection of a device initiated Stream selection by asserting a DP with the following field settings; Length = 0, Stream ID = *NoStream* and PP = 0. This packet shall transition the OSPSM from the **Start Stream** state to the **Start Stream End** state. The host shall reject a stream selection if there is no Endpoint Data Buffer available for the device selected SID.

The OSPSM is executing independently in the host and the device. A race condition occurs if the device issues an ERDY to the host and enters the **Start Stream** state, at the same time that the host issues a DP(Prime,PP=0) to the device and enters the **Prime Pipe** state. To recover from this condition, if the device receives a DP(Prime,PP=0) while in the **Start Stream** state it shall transition to the **Prime Pipe Ack** state and issue an NRDY(Prime) to the host, to complete the **Prime Pipe** to **Idle** transition for the host, and the **Prime Pipe Ack** to **Idle** transition for the device.

The **Start Stream End** state is an intermediate state for exiting the **Start Stream End** state when a selection is rejected. The device shall always force a transition to the **Idle** state by issuing an NRDY TP, with Stream ID = *NoStream*. This transition fulfills the requirement that a device must respond to a DP from the host.

In the **Move Data** state CStream is set at both ends of the pipe and the pipe is actively moving data to the device. The details of the bus transactions executed in the Move Data state and its exit conditions are defined in the OUT Move Data State Machine defined below.

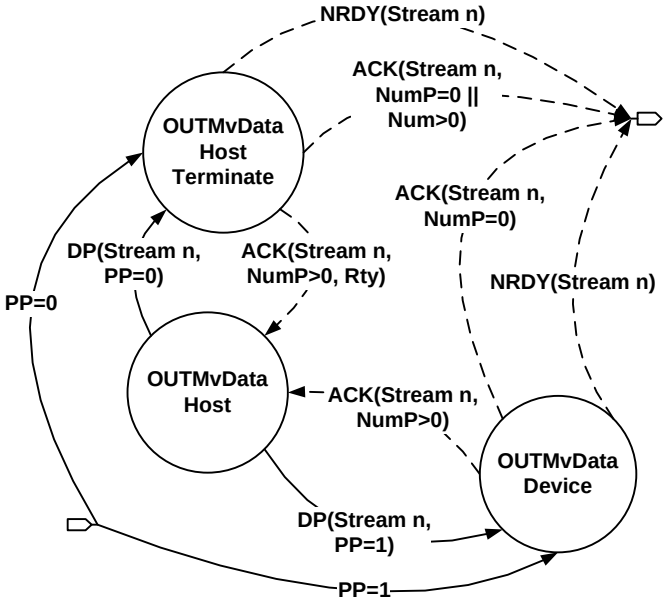


Figure 8-32. OUT Move Data State Machine (OMDSM)

The OUT Move Data State Machine (OMDSM) is entered from the **Start Stream** or **Idle** states as described above. The Stream ID field of all packets generated by the OMDSM shall be Stream n.

Upon entering the OMDSM, the value of the PP field in the received DP is evaluated. PP = 1 transitions the OMDSM to the **OUTMvData Device** state. PP = 0 transitions the OMDSM to the **OUTMvData Host Terminate** state.

Each time the **OUTMvData Device** state is entered, the device performs the following actions to advance the OMDSM:

if (Device Function Buffer space > Host Endpoint Data size)

The device shall generate an ACK TP with the NumP field > 0, which shall cause the pipe to transition to **OUTMvData Host** state.

else Device Function Buffer space <= Host Endpoint Data size

The device shall generate a Terminating ACK TP with the NumP field = 0, which shall cause the pipe to exit the OMDSM, and transition to **Idle** state.

Optionally, the device may generate an NRDY TP with the Stream ID set to *Stream n*, which terminates the stream, and shall cause the pipe to exit the OMDSM and transition to **Idle** state. The use of this transition is Device Class defined.

Note: If an intermediate hub deferred the DP, the host and the device shall act as if the device sent an NRDY TP. i.e. the host shall transition to the **Idle** state when it receives the Deferred Response. The device shall exit the OMDSM and transition to the **Idle** state when it receives the Deferred DP as if it has sent an NRDY TP with its Stream ID field set to *Stream n*. If the device accepts the host initiated Stream ID, it shall send an ERDY TP with its Stream ID field set to *Stream n*. If the device rejects the host initiated Stream ID, it shall stay the **Idle** state and wait for next Stream selection either by the host or the device.

The **OUTMvData Host** state is entered because the device has more Function Buffer space available for receiving data, so the host performs the following actions to advance the OMDSM.

if (Host Endpoint Data size > Max Packet Size)

Generate a DP with PP = 1, which shall cause the pipe to transition to **OUTMvData Device** state.

else the Host Endpoint Data size <= Max Packet Size

Generate a DP with PP = 0, which shall cause the pipe to exit the IMDSM and transition to the **OUTMvData Host Terminate** state.

The pseudo code describing the OMDSM is independent of whether the received ACK TP indicates a retry or not. An ACK TP with Retry shall affect the transmitted DP Sequence Number and cause Endpoint Data to be retransmitted.

The **OUTMvData Host Terminate** state is entered because the host has no more Endpoint Data to be sent (PP = 0), so the device shall generate a Terminating ACK TP with NumP = 0, which shall cause the pipe to exit the OMDSM and transition to the **Idle** state. If the device detects an error on the last DP sent by the host, it shall respond with a Retry ACK TP (Steam n, NumP>0, Rty=1) and the IMDSM shall transition to the **OUTMvData Host** state.

Note if a DP error is detected in the **OUTMvData Device** state the device shall generate an ACK TP with NumP > 0 and Rty=1, which shall cause the pipe to transition to **OUTMvData Host** state and resend the packet.

8.12.2 Control Transfers

Control transfers minimally have two transaction stages: Setup and Status. A control transfer may optionally contain a Data stage between the Setup and Status stages. The direction of the Data stage is indicated by the **bmRequestType** field which is present in the first byte of the data payload of the Setup packet. During the Setup stage, a SETUP transaction is used to transmit information to the control endpoint of a device. SETUP transactions are similar in format to a Bulk OUT transaction but have the **Setup** field set to one in the DPH along with the **Data Length** field set to eight. In addition, a Setup packet always uses a Data sequence number of zero. The device receiving a Setup packet shall accept the Setup packet and respond with an ACK TP. If the packet is corrupted, the device shall discard the data and return an ACK TP with the sequence number of zero and the Retry bit set to inform the host to resend the Setup packet.

Note that a device may return an ACK TP with the **NumP** field set to zero in response to a SETUP packet if it wants to place the endpoint in a flow control state. The device must send an ERDY to start the Data or Status stage.

The Data stage, if present, of a control transfer consists of one or more IN or OUT transactions and follows the same protocol rules as bulk transfers with a burst set to one. The Data stage always starts with the sequence number set to zero. All the transactions in the Data stage shall be in the same direction (i.e., all INs or all OUTs). The maximum amount of data to be sent during the data stage and its direction are specified during the Setup stage. If the amount of data exceeds the data packet size, the data is sent in multiple data packets that carry the maximum packet size. Any remaining data is sent as a residual in the last data packet.

Note that all control endpoints only support a burst of one and hence the host can only send/receive one packet at a time to/from a control endpoint.

The Status stage of a control transfer is the last transaction in the sequence. The status stage transaction is identified by a TP with the SubType set to *STATUS*. In response to a STATUS TP with zero in the **Deferred** bit, a device will send an NRDY, STALL or ACK TP. If it sends an NRDY TP the host will wait for the device to send an ERDY TP for that control endpoint before sending another STATUS TP to the device. If the **Deferred** bit is set in the STATUS TP then the device shall send an ERDY TP to indicate to the host that is ready to complete the status stage of the control transfer.

Figure 8-33 shows the transaction order, the data sequence number value, and the data packet types for control read and write sequences.

Figure 8-33. Control Read and Write Sequences

When a STALL TP is sent by a control endpoint in either the Data or Status stages of a control transfer, a STALL TP shall be returned on all succeeding accesses to that endpoint until a SETUP DP is received. The endpoint is not required to return a STALL TP after it receives a subsequent SETUP DP. For control endpoints, if an ACK TP is returned for the SETUP transaction, the host expects that the endpoint has automatically recovered from the condition that caused the STALL and the endpoint shall operate normally.

8.12.2.1 Reporting Status Results

During the Status stage, the device reports to the host the outcome of the previous Setup and Data stages of the transfer. Three possible results may be returned:

- The command sequence completed successfully.
- The command sequence failed to complete.
- The device is still busy completing the command.

Status reporting is always in the device-to-host direction. Table 8-30 summarizes the type of responses required for each. All Control transfers return status in the TP that is returned to the host in response to STATUS TP transaction.

Table 8-30. Status Stage Responses

Status Response	TP Sent by device
Request completes	ACK TP
Request has an error	STALL TP
Device is busy	NRDY TP

The host shall send a STATUS TP to the control pipe to initiate the Status stage. The pipe's handshake response to this TP indicates the current status. An NRDY TP indicates that the device is still processing the command and that the device will send an ERDY TP when it completes the command. An ACK TP indicates that the device has completed the command and is ready to accept a new command. A STALL TP indicates that the device has an error that prevents it from completing the command.

The **NumP** field the ACK TP sent by the control endpoint on the device must be set to zero. However this does not move the endpoint into the flow controlled state.

If during a Data stage a control pipe is sent more data or is requested to return more data than was indicated in the Setup stage, it shall return a STALL TP. If a control pipe returns a STALL TP during the Data stage, there will be no Status stage for that control transfer.

8.12.2.2 Variable-length Data Stage

A control pipe may have a variable-length data phase in which the host requests more data than is contained in the specified data structure. When all of the data structure is returned to the host, the device indicates that the Data stage is ended by returning a DP that has a payload less than the MaxPacketSize for that endpoint.

8.12.2.3 STALL TPs Returned by Control Pipes

Control pipes have the unique ability to return a STALL TP due to problems in control transfers. If the device is unable to complete a command, it returns a STALL TP in the Data and/or Status stages of the control transfer. Unlike the case of a functional stall, protocol stall does not indicate an error with the device. The protocol STALL condition lasts until the receipt of the next SETUP DP, and the device will return a STALL TP in response to any IN or OUT transaction on the pipe until the SETUP DP is received. In general, protocol stall indicates that the request or its parameters are not understood by the device and thus provides a mechanism for extending USB requests.

SuperSpeed USB devices do not support functional stall on a control pipe.

8.12.3 Bus Interval and Service Interval

For all periodic (interrupt and isochronous) endpoints, the interval at which an endpoint must be serviced is called a 'Service Interval'. In this specification the term 'Bus Interval' is used to refer to a 125 µs period.

8.12.4 Interrupt Transactions

The interrupt transfer type is used for infrequent data transfers with a bounded service period. It supports a reliable data transport with guaranteed bounded latency. It offers guaranteed constant data rate as long as data is available. If an error is detected in the data delivered, the host is not required to retry the transaction in the same service interval. However, if the device is momentarily unable to transmit or receive the data (i.e., responds with an NRDY TP), the host will resume transactions to the endpoint only after it receives an ERDY TP from that device for that endpoint.

Interrupt transactions are very similar to bulk transactions – but are limited to a burst of three DPs in each service interval. A host shall continue to perform transactions to the endpoint at the agreed upon service interval as long as the device accepts data (in the case of OUT endpoints) or returns data (in the case of IN endpoints). The host is required to send an ACK TP for every DP successfully received in the service interval even if it is the last DP in that service interval. The final ACK TP will acknowledge the last DP received and shall have the **Number of Packets** field set to zero.

Once the transfer is completed or when the device returns a flow control response, the host will not perform any more transactions to the endpoint except as noted in the above paragraph.

8.12.4.1 Interrupt IN Transactions

When a host wants to start an Interrupt IN transaction to an endpoint, it sends an ACK TP to the endpoint with the expected sequence number and the number of packets it expects to receive from the endpoint. If the endpoint is able to send data in response to the ACK TP from host, it may send up to the number of packets requested by the host within the same service interval. The host shall respond to each of the DPs with an ACK TP indicating successful reception of the data or retry the packet in case the data is corrupted.

Note that the host expects the first DP to have its sequence number set to zero when it starts the first transfer from a specific endpoint, after the endpoint has been initialized (via a Set Configuration or Set Interface or ClearFeature (STALL) command – See the Framework Chapter for details on these commands).

As long as the device returns data in response to the host sending ACK TPs and the transfer is not complete, the host shall continue to send ACK TPs to the device during every service interval for that endpoint. As noted previously, the host shall acknowledge the successful reception of the DP or retry the transaction if the DP was corrupted.

The host shall stop performing transactions to the endpoint on the device when any of the following happen:

- The endpoint responds with an NRDY/STALL TP
- All the data for the transfer is successfully received
- The endpoint sets the EOB flag in the last DP sent to the host

If an error is detected in the DP sent by the endpoint, the host shall respond with an ACK TP with Retry bit set. The endpoint is required to resend the same DP with same sequence number.

When the endpoint receives an ACK TP from the host and cannot respond by sending data, it shall send an NRDY (or STALL in case of an internal endpoint/device error) TP to the host. The host will not perform any more transactions to the endpoint in subsequent service intervals.

A host will resume interrupt transactions to an endpoint that responded with an NRDY TP (or with the EOB flag set in the last DP) sent to the host in a previous service interval only after it receives an ERDY TP from the endpoint. This notifies the host about the endpoint's readiness to transmit data again. Once the host receives the ERDY TP, it shall send an IN request (via an ACK TP) to the endpoint no later than twice the maximum service delay as determined by the value of the **bInterval** field in the interrupt endpoint descriptor. The endpoint responds by returning either the DP (the sequence number of the packet being one more than the sequence number of the last successful data sent) or, should it be unable to return data, an NRDY or STALL TP. The host will send an ACK TP when it receives a valid DP. If the host detects an error while receiving any of the DPs, it will send an ACK TP with the sequence number set to the sequence number of the packet it wants the endpoint to resend, with the Retry bit set.

If a device receives a deferred interrupt IN TP, and the device needs to send interrupt IN data, the device shall respond with an ERDY TP and keep its link in U0 until it receives the subsequent interrupt transaction from the host, or until tPingTimeout (see Table 8-34) time elapses.

As in the case of Bulk transactions, the sequence number is continually incremented with each packet sent by the endpoint. When the sequence number reaches 31 it wraps around to zero.

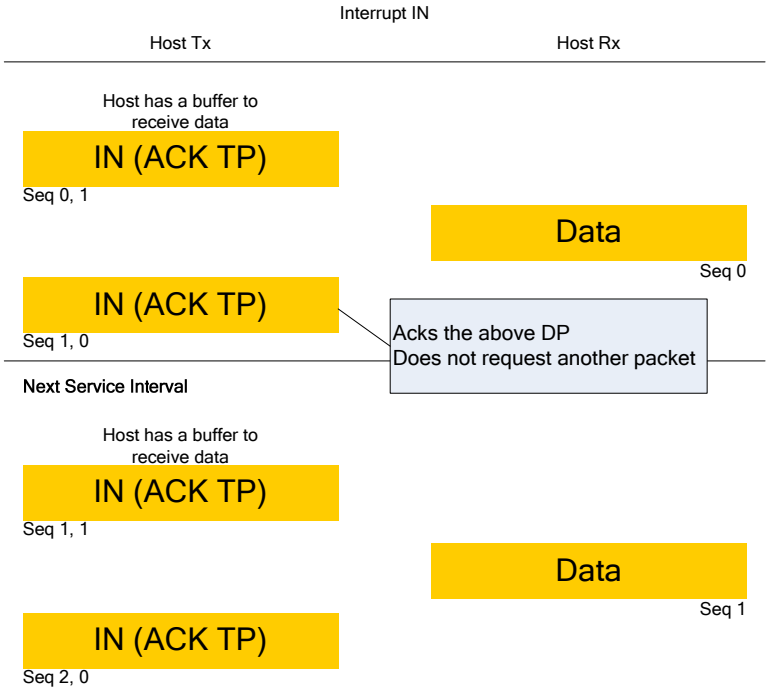


Figure 8-34. Host Sends Interrupt IN Transaction in Each Service Interval

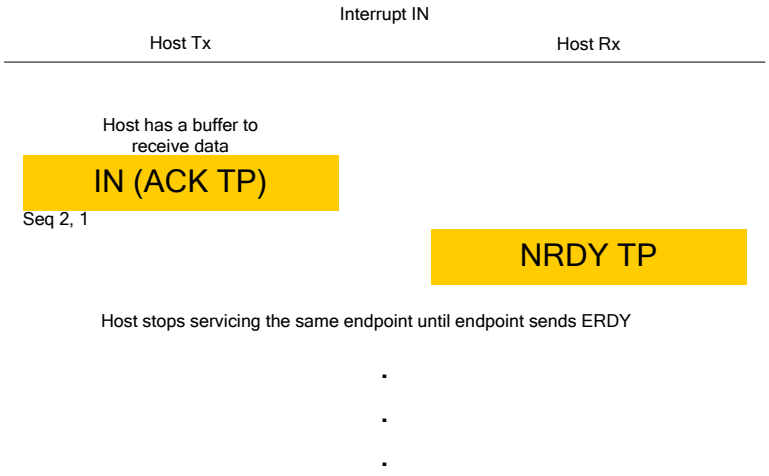


Figure 8-35. Host Stops Servicing Interrupt IN Transaction Once NRDY is Received

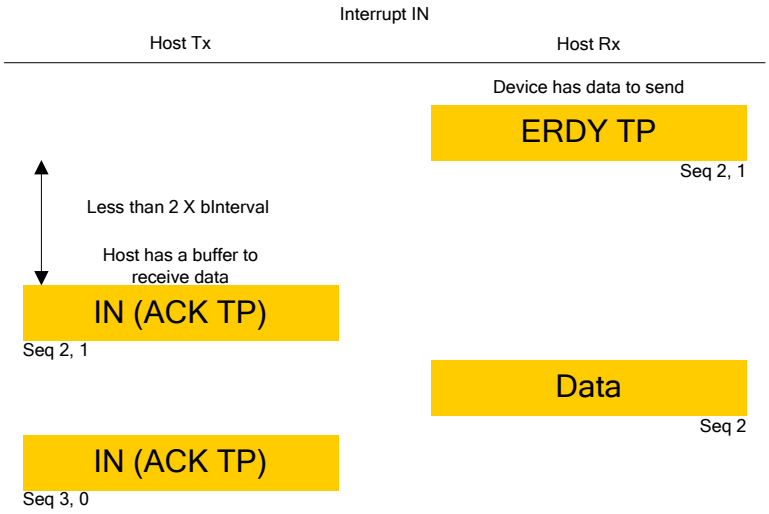


Figure 8-36. Host Resumes IN Transaction after Device Sent ERDY

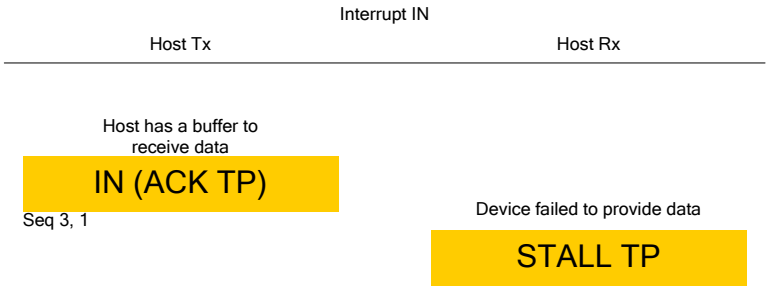


Figure 8-37. Endpoint Sends STALL TP

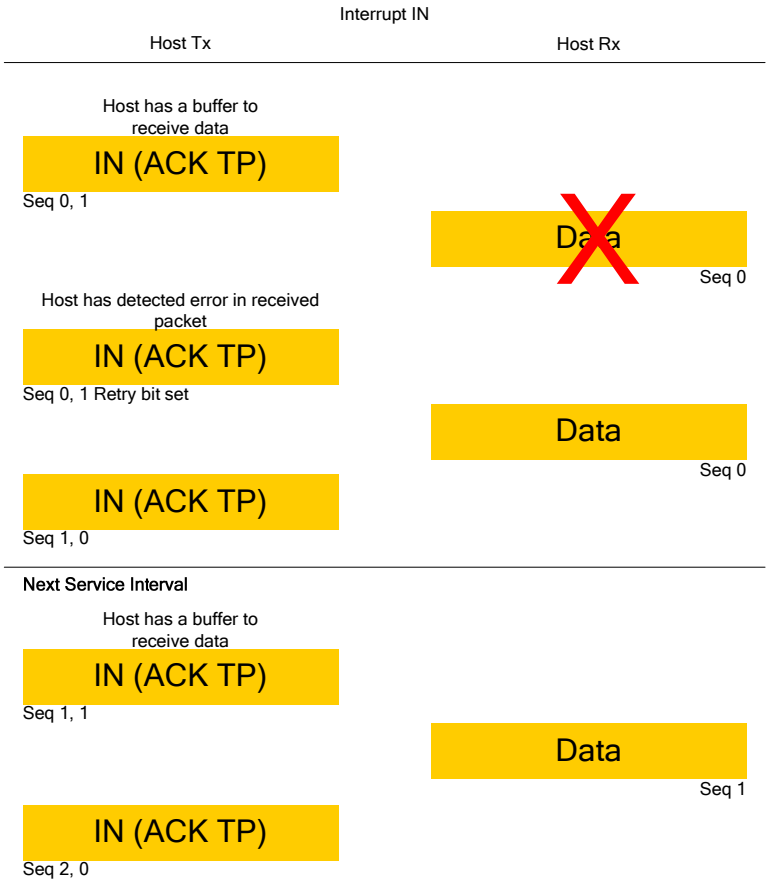


Figure 8-38. Host Detects Error in Data and Device Resends Data

8.12.4.2 Interrupt OUT Transactions

When the host wants to start an Interrupt OUT transaction to an endpoint, it sends the first DP with the expected sequence number. The host may send more packets to the endpoint in the same service interval if the endpoint supports more than a burst of one. If the endpoint is able to receive data from the host, it sends an ACK TP for acknowledging the successful receipt of data.

Note that the host always initializes the first DP sequence number to zero in the first transfer it performs to an endpoint after the endpoint is initialized (via a Set Configuration or Set Interface or ClearFeature (STALL) command – See the Framework Chapter for details on these commands).

As long as the device returns ACK TPs in response to the host sending data packets and the transfer is not complete, the host shall continue to send data to the device during every service interval for that endpoint. The device shall acknowledge the successful reception of the DP or ask the host to retry the transaction if the data packet was corrupted. Note that the host is not required to retry the transaction until the next service interval.

In response to the OUT data sent by the host with a valid DPT, the endpoint shall return one of the following three TPs:

- An ACK TP with the Retry bit clear indicates that the DP was received without error. If the DP was received with an error, the Retry bit will be set. Note that the host is not required to retransmit the DP in the same service interval.
- An NRDY TP indicates that the DP was received without error but that the host should resend the DP at a later time because the endpoint was in a temporary condition preventing it from accepting the DP (e.g., buffer full).
- If the endpoint was halted, a STALL TP is returned to indicate that the endpoint is permanently halted and requires USB System Software intervention. Note that STALL has priority over ACK or NRDY.

When the endpoint receives data from the host, and it cannot receive data momentarily, it shall send an NRDY (or STALL in case of an internal endpoint/device error) TP to the host. The host will not perform any more transactions to the endpoint in subsequent service intervals.

A host will only resume interrupt transactions to an endpoint that responded with an NRDY TP (or an ACK TP with the number of buffers available set to zero) after it receives an ERDY TP from the endpoint. This notifies the host about the endpoint's readiness to receive data again. Once the host receives an ERDY TP, the host shall transmit the data packet to the endpoint no later than twice the maximum service interval as determined by the value of the **bInterval** field in the interrupt endpoint descriptor for that endpoint. The host will then send a DP to the endpoint with the sequence number of the packet being one more than the sequence number of the last successful data sent. The endpoint will send an ACK TP when it receives a valid DP. If the device detects a DPP error while receiving any of the DPs, it will send an ACK TP with the sequence number set to the sequence number of the DP it wants the host to resend, with the Retry bit set.

If a device receives a deferred interrupt OUT DPH, and the device needs to receive interrupt OUT data, the device shall respond with an ERDY TP and keep its link in U0 until it receives the subsequent interrupt transaction from the host, or until tPingTimeout (see Table 8-34) elapses.

As in the case of Bulk transactions, the sequence number is continually incremented with each packet sent by host. When the sequence number reaches 31 it wraps around to zero.

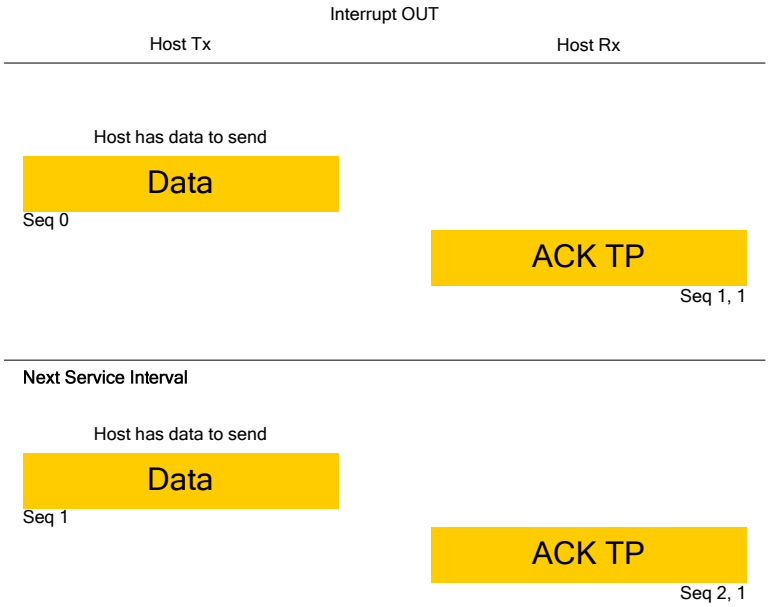


Figure 8-39. Host Sends Interrupt OUT Transaction in Each Service Interval

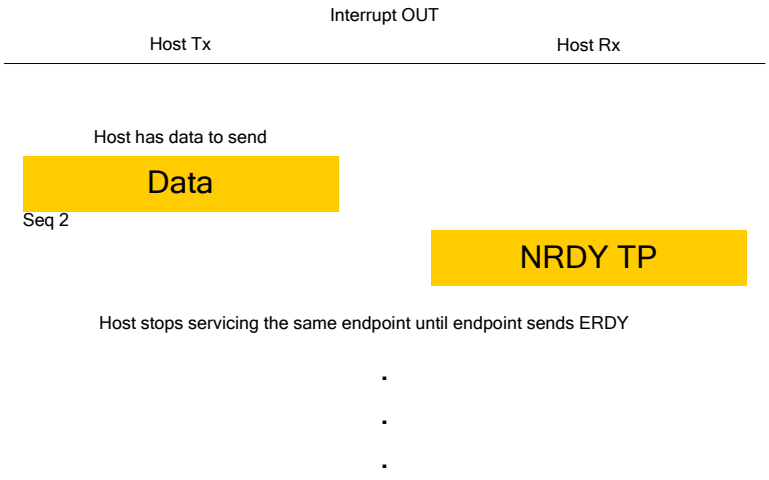


Figure 8-40. Host Stops Servicing Interrupt OUT Transaction Once NRDY is Received

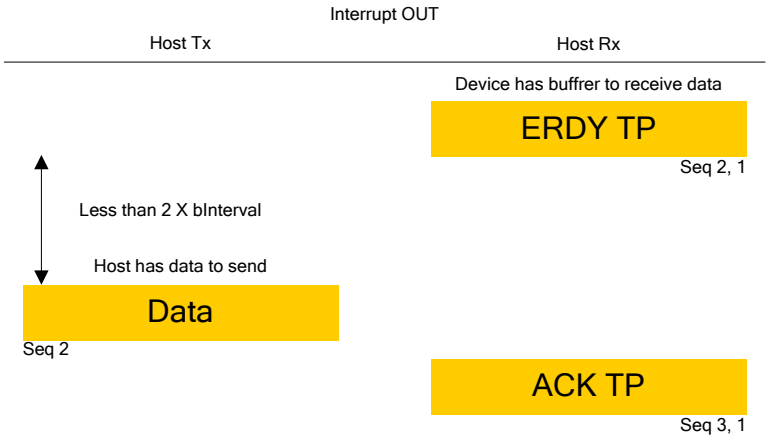


Figure 8-41. Host Resumes Sending Interrupt OUT Transaction after Device Sent ERDY

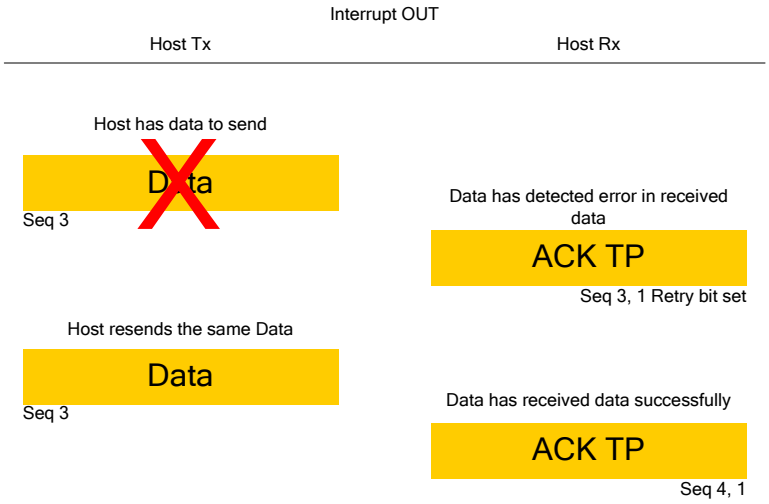


Figure 8-42. Device Detects Error in Data and Host Resends Data

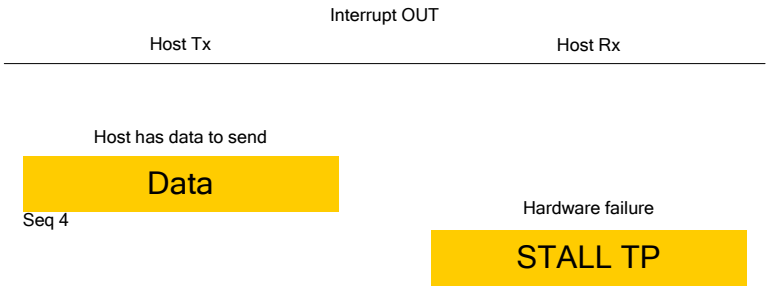


Figure 8-43. Endpoint Sends STALL TP

8.12.5 Host Timing Information

USB 3.0 host controllers do not broadcast regular start of frame (SOF) packets to all devices on a SuperSpeed USB link. Host timing information is only sent by the host via isochronous timestamp packets (ITP) when the root port link is in U0 around a bus interval boundary. Hubs forward isochronous timestamp packets to any downstream port with a link in U0. A host shall provide isochronous timestamps based on a non-spread clock. Devices are responsible for keeping the link in U0 around bus interval boundaries when isochronous timestamps are required for device operation. A device should never keep the link in U0 for the sole purpose of receiving timestamps unless the timestamps are required for device operation.

Note: A device will receive an isochronous timestamp if the device's link is in U0 around a bus interval boundary. This means that devices without any isochronous endpoints or need for synchronization may discard isochronous timestamp packets without negative side effects. If a device implements an inactivity timer for deciding when to drive the link into a lower power state, the device may choose to not reset the inactivity timer upon receiving an isochronous timestamps.

Figure 8-44 shows how host timing information is sent to a device with an active isochronous OUT endpoint. The service interval is one bus interval and the link is assumed to be in U0 continuously to keep the example simple. The timing information is sent in an isochronous timestamp packet around each bus interval boundary and communicates the current bus interval and the time from the start of the timestamp packet to the previous bus interval. Isochronous endpoints request a service interval of $125 * 2^n$ µs, where n is an integer value from 0 to 15 inclusive.

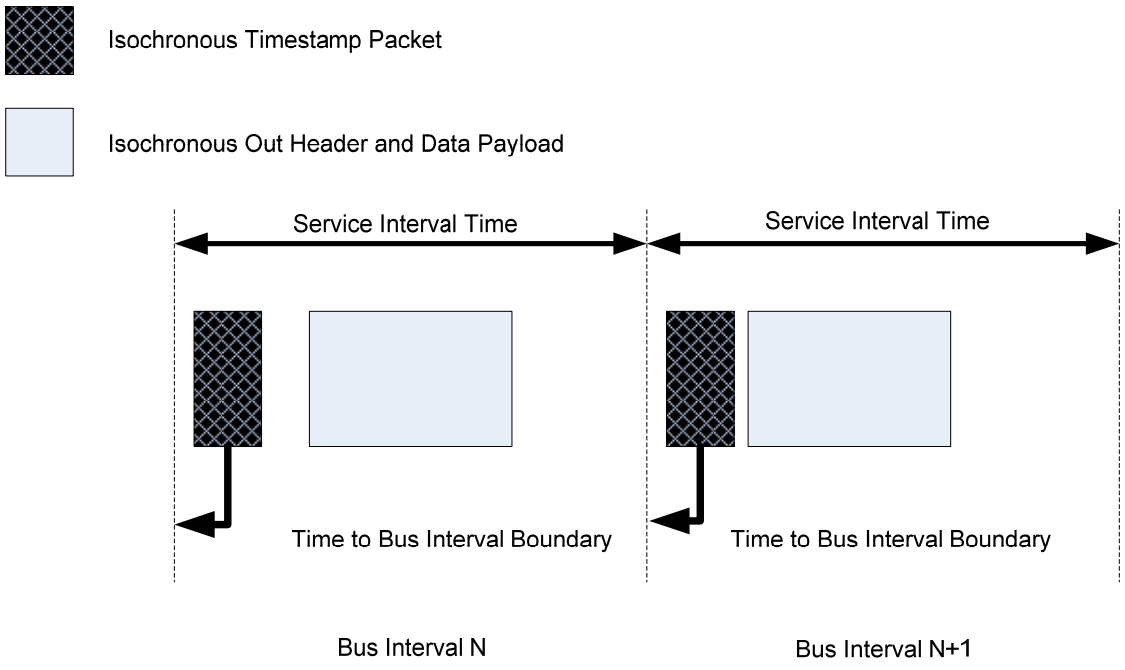


Figure 8-44. Host Timing Information for Isochronous OUT Endpoint

Figure 8-45 shows how host timing information is sent to a device with an active isochronous IN endpoint. The timing information is sent in an isochronous timestamp packet around each bus interval boundary and communicates the current bus interval and the time from the start of the timestamp packet to the previous bus interval boundary. The service interval is one bus interval and the link is assumed to be in U0 continuously to keep the example simple.

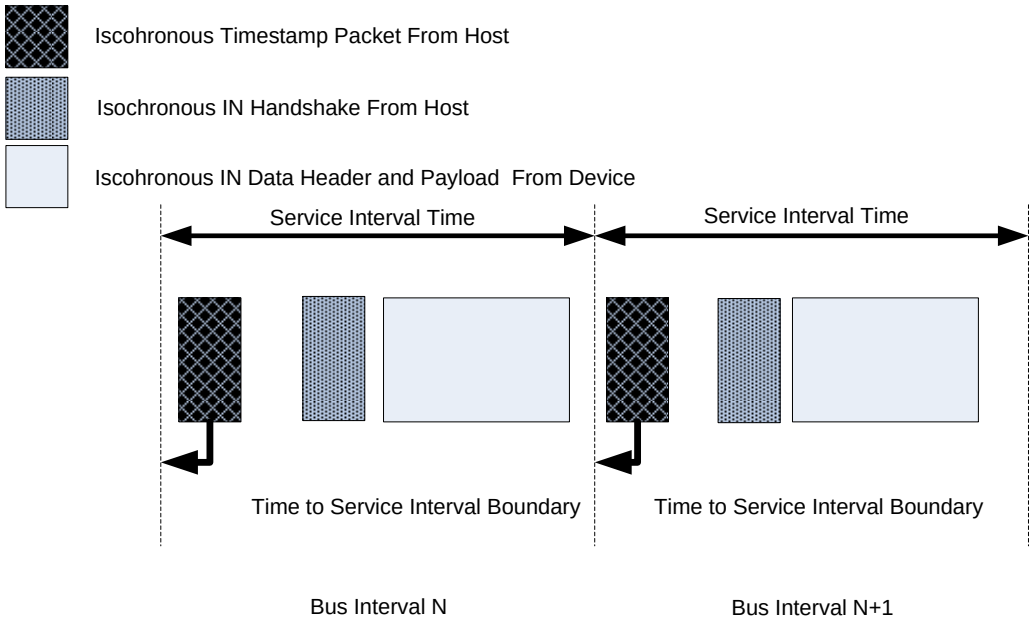


Figure 8-45. Host Timing Information for Isochronous IN Endpoint

When there are multiple active isochronous endpoints connected to the same USB 3.0 host – the host shall communicate timing information such that the bus interval boundaries for all the isochronous endpoints are aligned. A host shall keep bus interval boundaries aligned for all devices and endpoints at all times unless the host link enters U3. Isochronous timestamps issued after the host root port link exits U3 may be aligned with boundaries from before the host root port link entered U3. If ITPs issued after a host root port link exits U3 are not aligned with boundaries from before the host root port link entered U3, A host shall begin transmitting ITPs within tIsochronousTimestampStart from when the host root port's link enters U0 after the link was in U3. Figure 8-46 shows an example with an active isochronous IN endpoint and isochronous OUT endpoint connected below the same USB 3.0 host controller. The service interval for the isochronous IN endpoint is X and the service interval for the isochronous OUT endpoint is 2X. The bus interval boundaries are aligned as required. Note that the USB 3.0 host is free to schedule an isochronous IN (via an ACK TP) or isochronous OUT data anywhere within the appropriate service interval. A device shall not assume that transactions will occur at the same location within each service interval. A host shall schedule isochronous transactions such that they do not cross service interval boundaries.

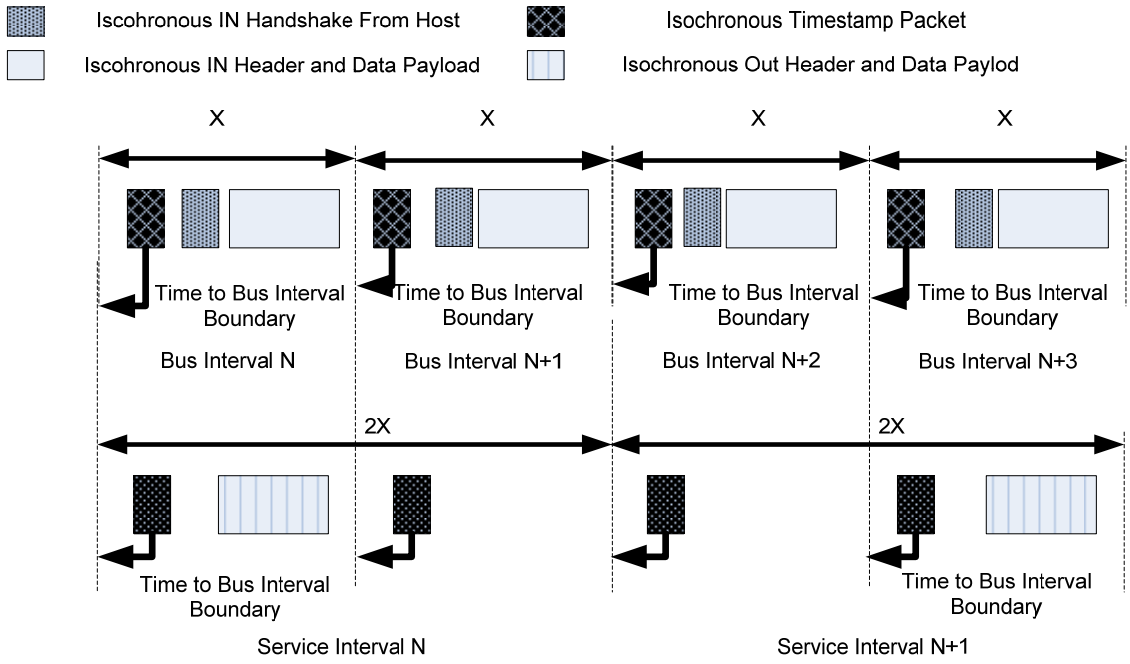


Figure 8-46. Multiple Active Isochronous Endpoints with Aligned Service Interval Boundaries

8.12.6 Isochronous Transactions

IN and OUT isochronous transactions are shown in Figure 8-48. For INs the host issues an ACK TP followed by a data phase in which the endpoint transmits data for INs. For OUTs the host simply transmits data when there is data to be sent in the current service interval. Isochronous transactions do not support retry capability.

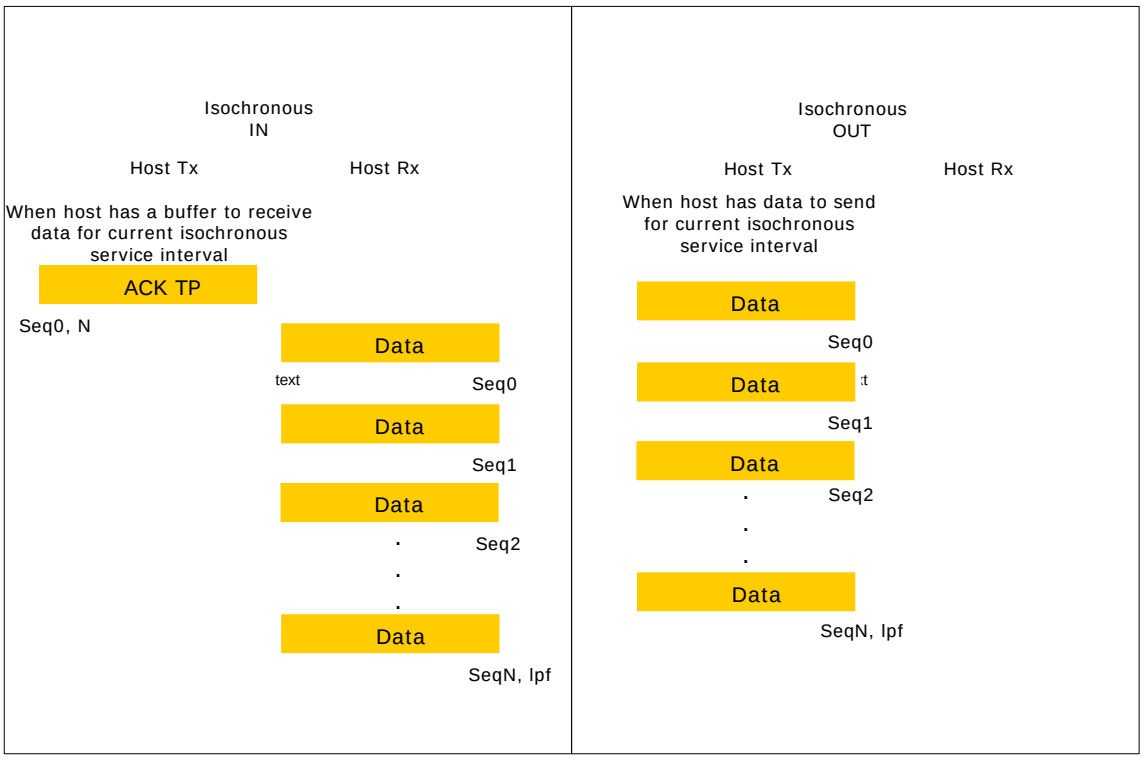


Figure 8-47. Isochronous Transaction Format

SuperSpeed USB isochronous transactions with a single data packet per service interval always use sequence number zero. For isochronous transactions that include multiple data packets in a service interval the sequence number is increased by one for each subsequent data packet. The first data packet sent in any service interval always uses sequence number zero. A SuperSpeed USB host controller shall be able to accept and send up to 48 data packets (DP) per service interval. The first DP in each service interval shall start with the sequence number set to 0. The second DP shall have a sequence number set to one; the third DP has a sequence number set to two and so on until sequence number 31. The next DP after sequence number 31 uses a sequence number of zero. A SuperSpeed USB device with an isochronous endpoint shall be able to send or receive the number of packets (with sequence numbers 0 – N) as indicated in its endpoint and endpoint companion descriptors.

If there is no data to send to an isochronous OUT endpoint during a service interval the host does not send anything during the interval. If a device with an isochronous IN endpoint does not have data to send when an isochronous IN ACK TP is received from the host – it shall send a zero length data packet. The host shall send one and only one ACK TP to an isochronous IN endpoint each

service interval. Figure 8-48 shows sample isochronous IN and OUT transactions for endpoints than have requested 2000 bytes of bandwidth per service interval (no more than 2 packets can be sent or received each service interval).

If a host is not able to send isochronous OUT data during the specified interval due to an error condition, the host discards the data and notifies host software of the error. If a host is not able to send an isochronous ACK TP during the specified service interval due to an error condition the host notifies host software of the error.

Figure 8-48. Sample Isochronous IN and OUT Transactions

Figure 8-49. Isochronous IN and OUT Transaction Example

8.12.6.1 Host Flexibility in Performing Isochronous Transactions

A host is allowed some flexibility in performing isochronous service during a service interval. A host may perform the isochronous transaction for an endpoint as a single burst or it may split the service into bursts of 2, 4, or 8 followed by any remainder. A host shall not perform isochronous transaction in any other way. For isochronous endpoints that have a multiplier value greater than one these rules apply to the transaction associated with each multiplier value separately. A device shall support all possible host transactions allowed by these rules. For example – if an isochronous OUT endpoint requests a maximum number of packets in a burst of 11 and the host has 11 packets to send to the endpoint during a service interval there are 4 possible ways the host could perform the transaction:

- A single burst of 11 packets.
- A burst of 8 followed by a burst of three.
- Two bursts of 4 followed by a burst of three.
- Five bursts of 2 followed by a burst of one.

The host shall not reset the sequence number within the service interval. The host shall set the lpf flag only on the last packet in the last burst within the service interval.

8.12.6.2 Device Response to Isochronous IN Transactions

Table 8-31 shows the possible responses a device may make in response to an ACK TP. If the device is unable to send data it sends a zero byte data packet. If the device is able to issue data, it does so and shall send MaxPacketSize bytes for each packet except the last. If the received ACK TP is invalid, the device returns no response.

Table 8-31. Device Responses to Isochronous IN Transactions

ACK TP Received Invalid	Device Can Transmit Data	Action Taken
Yes	Do not care	Return no response
No	No	Return zero length data packet with sequence number 0.
No	Yes	Return N data packets with sequence numbers 0 to N-1. Each packet except the last shall be MaxPacketSize bytes. The last packet shall have the lpf flag set.

8.12.6.3 Host Processing of Isochronous IN Transactions

Table 8-32 shows the host processing of data from an IN transaction. The host never returns a response to isochronous IN data received. If the host receives a corrupted data packet, it discards the data and informs host software of the error. If the host cannot accept data from a device, (due to problems such as internal buffer overrun) this condition is considered to be an error and the host returns no response to the device (note that this is an error condition that should never occur). If the host is able to accept data and the data packet is received error-free, the host accepts the data and sends no response.

Table 8-32. Host Responses to IN Transactions

Data Packet Corrupted	Host Can Accept Data	Host Data Processing
Yes	N/A	Discard data.
No	No. (This should never happen for a compliant host implementation.)	Discard data.
No – Data Packet Has Expected Sequence Number	Yes	Accept data.
No – Data Packet Does Not Have Expected Sequence Number.	Yes	Accept data, record missing sequence numbers.

8.12.6.4 Device Response to an Isochronous OUT Data Packet

Table 8-33 shows the device processing of data from an OUT data packet. The device never returns a TP in response.

Table 8-33. Device Responses to OUT Data Packets

Data Corrupted	Expected Sequence Number	Device Can Accept Data	Device Data Processing
Yes	Do not care	Do not care	Discard data.
No	Yes	Yes	Accept data.
No	Yes	No	Data discarded.
No	No	No	Data discarded. Device may discard any additional data for current service interval.
No	No	Yes	Data discarded. Device may discard all data for the rest of current service interval.

8.13 Timing Parameters

Table 8-34 lists the minimum and/or maximum times a device shall adhere to when responding to various types of packets it receives. It also lists the default and minimum times a device may set in Latency tolerance messages as well as the minimum time after receipt of certain TPs and when it can initiate a U1/U2 entry. In addition it lists the maximum time between DPs a device must adhere to while bursting.

Note that all txxxResponse (e.g. tNRDYResponse) times are all timings that a device shall meet when the device has nothing else to send on its upstream link.

Table 8-34. Timing Parameters

Name	Description	Min	Max	Units
tPortConfiguration	Maximum time after a successful warm reset or a power on reset that the link partners must complete the port configuration process. This includes the time to exchange port capabilities, send the port configuration and receive acknowledgement.		20	µs
tPingTimeout	Timeout after a device receives a ping from the host and when it can initiate U1 or U2. This parameter is measured in terms of the maximum of all the service intervals for all isochronous endpoints within the device.	2		Service intervals
tPingResponse	Time between device reception of the last framing symbol of a ping and the first framing symbol of the ping_response.		250	ns
tBELTdefault	Default for best effort latency tolerance	1		ms
tBELTmin	Minimum value of best effort latency tolerance allowed in a Latency Tolerance Message.	125		µs
tNRDYResponse	Time between device reception of the last framing symbol for an ACK TP or a DPP and the first framing symbol of an NRDY response.		250	ns
tDPResponse	Time between device reception of the last framing symbol for an ACK TP and the first framing symbol of a DP response.		250	ns
tACKResponse	Time between device reception of the last framing symbol for a DPP and the first framing symbol of an ACK response.		250	ns
tHostACKResponse	Time between host reception of the last framing symbol for a DPP and the first framing symbol of an ACK response.		3	µs
tERDYTimeout	Timeout after a device sends an ERDY to the host and when it can initiate U1 or U2 if not serviced.	500		ms
tNotification	Rate at which a device shall send a function wake notification if the device hasn't been accessed (since sending the last function wake notification)	2500		ms

Name	Description	Min	Max	Units
tMaxBurstInterval	Time between DPs when a device or host is bursting. Note that there is an additional requirement for devices. If the device cannot meet this max time then it shall set the **eob** flag in the last DP it sends.		100	ns
tTimestampWindow	A host shall transmit an isochronous timestamp from a bus interval boundary to tTimestampWindow after the bus interval boundary if the root port's link is in U0.	0	8	μs
tIsochTimestampGranularity	The granularity of isochronous timestamps.	8	8	USB 2.0 High-speed bit times
BusIntervalAdjustmentGranularity	The adjustment unit for device requested changes to the bus interval.		4.0690104[1]	ps
tIsochronousTimestampStart	Time by which a host shall start transmitting isochronous timestamps after a root port link enters U0 from polling or after the root port link enters U0 after the link was in U3.		250	μs

[1] (tIsochTimestampGranularity/4096)

Note: If a host does not see a response to a Data Transaction (either IN or OUT) within 10 μs it shall assume that the transaction has failed and halt the endpoint. No retries shall be performed.

9 Device Framework

A USB device may be divided into three layers:

- The bottom layer is a bus interface that transmits and receives packets.
- The middle layer handles routing data between the bus interface and various endpoints on the device. As in USB 2.0, the endpoint is the ultimate consumer or provider of data. It may be thought of as a source or sink for data. The characteristics of an endpoint, e.g., the endpoint's transfer type, the maximum payload (MaxPacketSize), and the number of packets (Burst Size) it can receive or send at a time are described in the endpoint's descriptor.
- The top layer is the functionality provided by the serial bus device, for instance, a mouse or video camera interface.

This chapter describes the common attributes and operations of the middle layer of a USB device. These attributes and operations are used by the function-specific portions of the device to communicate through the bus interface and ultimately with the host.

9.1 USB Device States

A USB device has several possible states. Some of these states are visible to the USB and the host, while others are internal to the USB device. This section describes those states.

9.1.1 Visible Device States

This section describes USB device states that are externally visible (see Figure 9-1). Table 9-1 summarizes the visible device states.

NOTE

USB devices perform a reset operation in response to reset signaling on the upstream facing port. When reset signaling has completed, the USB device is reset. The reset signaling depends on the link state. See Section 7.3 for details.

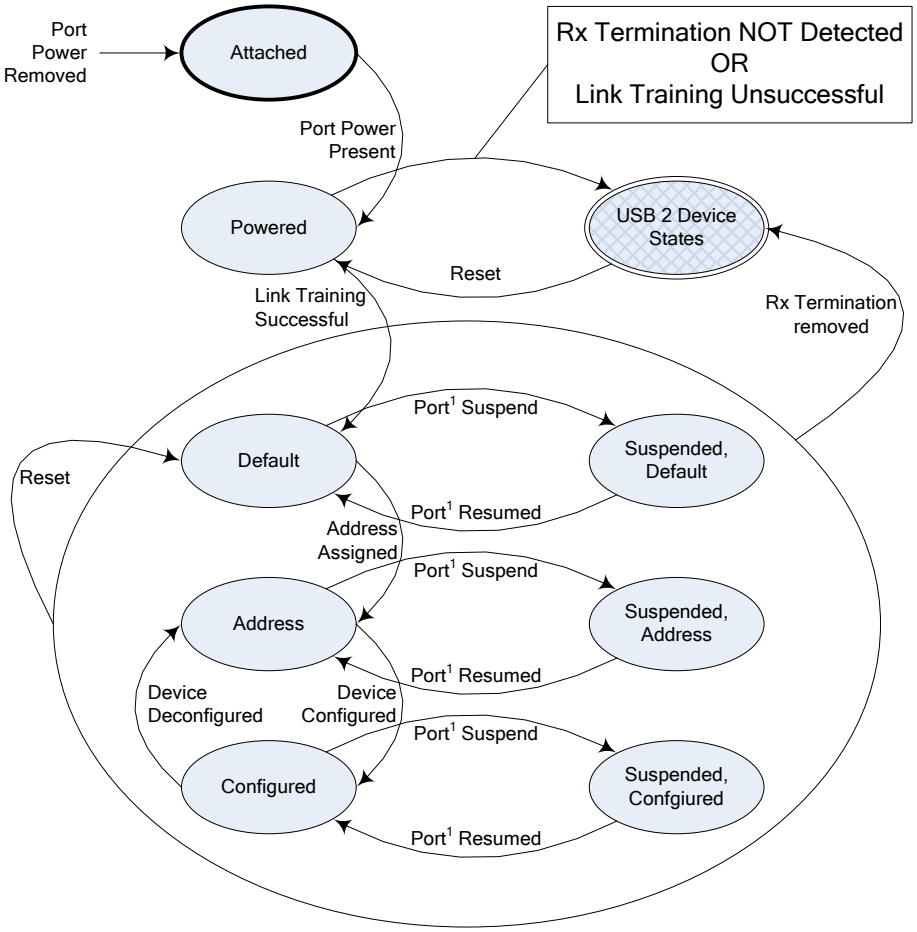


Figure 9-1. Device State Diagram

[1] Port refers to the device's upstream port

Table 9-1. Visible Device States

Attached	Powered	Default	Address	Configured	Suspended[1]	State
No	--	--	--	--	--	Device is not attached to the USB. Other attributes are not significant.
Yes	No	--	--	--	--	Device is attached to the USB, but is not powered. Other attributes are not significant.
Yes	Yes	No	--	--	--	Device is attached to the USB and powered and its upstream link has not successfully completed training.
Yes	Yes	Yes	No	--	--	Device is attached to the USB and powered and has been reset, but has not been assigned a unique address. Device responds at the default address.
Yes	Yes	Yes	Yes	No	--	Device is attached to the USB, powered, has been reset, and a unique device address has been assigned. Device is not configured.
Yes	Yes	Yes	Yes	Yes	No	Device is attached to the USB, powered, has been reset, has a unique address, is configured, and is not suspended. The host may now use the function provided by the device.
Yes	Yes	Yes	--	--	Yes	Device is, at minimum, in the Default State (Attached to the USB, is powered and its upstream link has been successfully trained) and its upstream link has been set to U3 by its upstream link partner. It may also have a unique address and be configured for use. However, because the device is suspended, the host may not use the device's function.

[1]Suspended from the Default, Address or Configured state

9.1.1.1 Attached

A USB device may be attached or detached from the USB. The state of a USB device when it is detached from the USB is not defined by this specification. This specification only addresses required operations and attributes once the device is attached.

9.1.1.2 Powered

USB devices may obtain power from an external source and/or from the USB through the hub to which they are attached. Externally powered USB devices are termed self-powered. Although self-powered devices may already be powered before they are attached to the USB, they are not considered to be in the Powered state until they are attached to the USB and VBUS is applied to the device.

A device may support both self-powered and bus-powered configurations. Some device configurations support either power source. Other device configurations may be available only if the device is self-powered. Devices report their power source capability through the configuration descriptor. The current power source is reported as part of a device's status. Devices may change their power source at any time, e.g., from self- to bus-powered. If a configuration is capable of supporting both power modes, the power maximum reported for that configuration is the maximum the device will draw from VBUS in either mode. The device shall observe this maximum, regardless of its mode. If a configuration supports only one power mode and the power source of the device changes, the device will lose its current configuration and address and return to the Powered state. If a device operating in SuperSpeed mode is self-powered and its current configuration requires more than 150 mA, then if the device switches to being bus-powered, it shall return to the Address state. Self-powered hubs that use VBUS to power the Hub Controller are allowed to remain in the Configured state if local power is lost. Note that the max power draw for a SuperSpeed device operating in the High-speed mode is governed by the limits set in the USB 2.0 specification.

A hub port shall be powered in order to detect port status changes, including attach and detach. Bus-powered hubs do not provide any downstream power until they are configured, at which point they will provide power as allowed by their configuration and power source. A USB device shall be able to be addressed within a specified time period from when power is initially applied (refer to Section TBD in the Logical Chapter). After an attachment to a port has been detected, the host may reset the port, which will also reset the device attached to the port.

9.1.1.3 Default

When operating in SuperSpeed mode, after the device has been powered, it shall not respond to any bus transactions until its link has successfully trained. The device is then addressable at the default address.

A device that is capable of SuperSpeed operation determines whether it will operate at SuperSpeed as a part of the connection process (see the Device Connection State Diagram in the Hub Chapter for more details).

A USB 3.0 device shall reset successfully at one of the supported USB 2.0 speeds when in an USB 2.0 only electrical environment. After the device is successfully reset, the device shall also respond successfully to device and configuration descriptor requests and return appropriate information as per the requirements laid out in the USB 2.0 specification. The device may or may not be able to support its intended functionality when operating in the USB 2.0 mode.

9.1.1.4 Address

All USB devices use the default address when initially powered or after the device has been reset. Each USB device is assigned a unique address by the host. A USB device maintains its assigned address while suspended.

A USB device responds to requests on its default pipe whether the device is currently assigned a unique address or is using the default address.

9.1.1.5 Configured

Before a USB device's function may be used, the device shall be configured. From the device's perspective, configuration involves correctly processing a SetConfiguration() request with a non-zero configuration value. Configuring a device or changing an alternate setting causes all of the status and configuration values associated with all the endpoints in the affected interfaces to be set to their default values. This includes resetting the data sequence of any endpoint in the affected interfaces to zero. On initial entry into the configured state a device shall default into the fully functional D0 State.

9.1.1.6 Suspended

In order to conserve power, USB devices automatically enter the Suspended state (one of Suspended Default, Address or Configured) when they observe that their upstream link is being driven to the U3 state (see Section 7.1.4.2.4). When suspended, the USB device maintains any internal status, including its address and configuration.

Attached devices shall be prepared to suspend at any time from the Default, Address or Configured states. A USB device shall enter the Suspended state when the hub port it is attached to is set to go into U3. This is referred to as selective suspend.

A USB device exits suspend mode when it observes wake-up signaling (see Section 7.4.9) on its upstream port. A USB device may also request the host to exit suspend mode or selective suspend by driving resume signaling (see Section 7.4.9) on its upstream link to indicate remote wakeup. The ability of a device to signal remote wakeup is optional. If a USB device is capable of remote wakeup signaling, the device shall support the ability of the host to enable and disable this capability. When the device is reset, remote wakeup signaling shall be disabled.

9.1.2 Bus Enumeration

When a USB device is attached to or removed from the USB, the host uses a process known as bus enumeration to identify and manage the device state changes necessary. When a USB device is attached to a powered port, the following actions are taken:

1. The hub to which the USB device is now attached informs the host of the event via a reply on its status change pipe (see Section 10.11.1. At this point, the USB device has been reset, is in the Default state and the port to which it is attached is enabled and ready to respond to control transfer requests on the Default Control Pipe.
2. The host determines the exact nature of the change by querying the hub.
3. Now that the host knows the port to which the new device has been attached, the host then may reset the device again if it so wishes but it is not required to do so.
4. If the host resets the port, the hub performs the required reset processing for that port (see Section 10.3.1.6). When the reset is completed, the port will be back in the enabled state.

5. The USB device is now in the Default state and can draw no more than 150 mA from VBUS. All of its registers and state have been reset and it answers to the default address.
6. The host assigns a unique address to the USB device, moving the device to the Address state.
7. Before the USB device receives a unique address, its Default Control Pipe is still accessible via the default address. The host reads the device descriptor to determine what the actual maximum data payload size this USB device's default pipe can use.
8. The host reads the configuration information from the device by reading each configuration from zero to *n*-1, where *n* is the number of configurations. This process may take several milliseconds to complete.
9. Based on the configuration information and how the USB device will be used, the host assigns a configuration value to the device. The device is now in the Configured state and all of the endpoints in this configuration have taken on their described characteristics. The USB device may now draw the amount of VBUS power described in its descriptor for the selected configuration. From the device's point of view, it is now ready for use.

When the USB device is detached, the hub again sends a notification to the host. Detaching a device disables the port to which it had been attached and the port moves into the Disconnected state (see Section 10.3.1.2). Upon receiving the detach notification, the host will update its local topological information.

9.2 Generic USB Device Operations

All USB devices support a common set of operations. This section describes those operations.

9.2.1 Dynamic Attachment and Removal

USB devices may be attached or detached at any time. The hub provides the attachment point or downstream port and is responsible for reporting any change in the state of the port.

The hub resets and enables the hub downstream port where the device is attached upon detection of an attachment, which also has the effect of resetting the device. A reset USB device has the following characteristics:

- Its USB address is set to zero (the default USB address)
- It is not configured
- It is not suspended

When a device is removed from a hub port, the hub disables the port where the device was attached, the port moves into the Dis-connected state (see Section 10.3.1.2) and notifies the host of the removal.

9.2.2 Address Assignment

When a USB device is attached, the host is responsible for assigning a unique address to the device. Before assigning an address, the host may explicitly reset the device; however note that the device implicitly gets reset during the connection process before the host is notified of a device being attached to the port.

9.2.3 Configuration

A USB device shall be configured before its function(s) may be used. The host is responsible for configuring a USB device. The host typically requests configuration information from the USB device to determine its capabilities.

As part of the configuration process, the host sets the device configuration and, where necessary, selects the appropriate alternate settings for the interfaces.

Within a single configuration, a device may support multiple interfaces. An interface is a related set of endpoints that present a single feature or function of the device to the host. The protocol used to communicate with this related set of endpoints and the purpose of each endpoint within the interface may be specified as part of a device class or vendor-specific definition.

In addition, an interface within a configuration may have alternate settings that redefine the number or characteristics of the associated endpoints. If this is the case, the device shall support the GetInterface() request to report the current alternate setting for the specified interface and SetInterface() request to select the alternate setting for the specified interface.

Within each configuration, each interface descriptor contains fields that identify the interface number and the alternate setting. Interfaces are numbered from zero to one less than the number of concurrent interfaces supported by the configuration. Alternate settings range from zero to one less than the number of alternate settings for a specific interface. The default setting when a device is initially configured is alternate setting zero.

In support of adaptive device drivers that are capable of managing a related group of USB devices, the device and interface descriptors contain *Class*, *SubClass*, and *Protocol* fields. These fields are used to identify the function(s) provided by a USB device and the protocols used to communicate with the function(s) on the device. A class code is assigned to a group of related devices that has been characterized as a part of a USB Class Specification. A class of devices may be further subdivided into subclasses, and, within a class or subclass, a protocol code may define how the Host Software communicates with the device.

NOTE

The assignment of class, subclass, and protocol codes shall be coordinated but is beyond the scope of this specification.

9.2.4 Data Transfer

Data may be transferred between an endpoint within a USB device and the host in one of four ways. Refer to the Data Flow Chapter for the definition of the four types of transfers. An endpoint number may be used for different types of data transfers in different alternate settings. However, once an alternate setting is selected (including the default setting of an interface), a USB device endpoint uses only one data transfer method until a different alternate setting is selected.

9.2.5 Power Management

Power management on USB devices involves the issues described in the following sections.

9.2.5.1 Power Budgeting

USB bus power is a limited resource. During device enumeration, a host evaluates a device's power requirements. If the power requirements of a particular configuration exceed the power available to the device, Host Software shall not select that configuration.

USB devices shall limit the power they consume from VBUS to one unit load or less until configured. When operating in SuperSpeed mode, 150 mA equals one unit load. Suspended devices, whether configured or not, shall limit their bus power consumption as to the suspend mode power requirements in the USB 2.0 specification. Depending on the power capabilities of the port to which the device is attached, a SuperSpeed USB device operating in SuperSpeed mode may be able to draw up to six unit loads from VBUS after configuration. The amount of current draw for SuperSpeed devices are increased to 150 mA for low-power devices and 900 mA for high-power devices when operating in SuperSpeed mode.

Device power management is comprised of suspend and function suspend. Suspend refers to a device wide state. Function suspend refers to a state of an individual function within a device. For single function devices suspend and function suspend are synonymous. Suspending a device with more than one function effectively suspends all the functions within the device.

9.2.5.2 Changing Device Suspend State

Device suspend is entered and exited intrinsically as part of the suspend entry and exit processes. Some device state information may be retained in a persistent state during suspend. The minimum set of said information that shall be retained is listed below:

- Port status change (downstream ports)
- Device address
- Device configuration value
- Function suspend and function remote wake enable state

Device classes may require additional information to be retained during suspend, beyond what is identified in this specification and is beyond the scope of this specification. Devices can optionally remove power from circuitry that is not needed while in suspend.

9.2.5.3 Function Suspend

The function suspend state is a reduced power state associated with an individual function. The function may or may not be part of a composite device.

A function may be placed into function suspend independently of other functions within a composite device. A device may be transitioned into device suspend regardless of the function suspend state of any function within the device. Function suspend state is retained while in device suspend and throughout the device suspend entry and exit processes.

9.2.5.4 Changing Function Suspend State

Functions are placed into function suspend using the FUNCTION_SUSPEND feature selector (see Table 9-6). The FUNCTION_SUSPEND feature selector also controls whether the function may initiate a function remote wakeup. Whether a function is capable of initiating a Function Remote Wake is determined by the status returned when the first interface in that function is queried using a Get Status command (see Section 9.4.5).

Remote wakeup (i.e., wakeup from a device suspend state) is enabled when any function within a device is enabled for function remote wakeup. (Note the terminology distinction between "function remote wake" and "remote wake.") The DEVICE_REMOTE_WAKEUP feature selector is ignored and not used by SuperSpeed USB devices.

A function may request exit from function suspend by sending a remote wake message to the host if it is enabled for function remote wakeup. This applies to single function devices as well as multiple function (i.e. composite) devices. If the link is in a non-U0 state then the device must transition the link to U0 prior to sending the remote wake message. If a remote wake event occurs in multiple functions, each function shall send a remote wake message.

When all functions within a device are in function suspend and the PORT_U2_TIMEOUT field (see Section 10.14.2.9) is programmed to 0xFF, the device shall initiate U2 after 10 ms of link inactivity.

9.2.6 Request Processing

With the exception of SetAddress() requests (see Section 9.4.6), a device may begin processing of a request as soon as the device returns the ACK following the Setup. The device is expected to "complete" processing of the request before it allows the Status stage to complete successfully. Some requests initiate operations that take many milliseconds to complete. For requests such as this, the device class is required to define a method other than Status stage completion to indicate that the operation has completed. For example, a reset on a hub port may take multiple ms to complete depending on the status of the link attached to the port. The SetPortFeature(PORT_RESET) (see Section 10.14.2.9) request "completes" when the reset on the port is initiated. Completion of the reset operation is signaled when the port's status change is set to indicate that the port is now enabled. This technique prevents the host from having to constantly poll for a completion when it is known that the request will take a relatively long period of time.

9.2.6.1 Request Processing Timing

All devices are expected to handle requests in a timely manner. USB sets an upper limit of 5 seconds for any command to be processed. This limit is not applicable in all instances. The limitations are described in the following sections. It should be noted that the limitations given below are intended to encompass a wide range of implementations. If all devices in a USB system used the maximum allotted time for request processing, the user experience would suffer. For this reason, implementations should strive to complete requests in times that are as short as possible.

9.2.6.2 Reset/Resume Recovery Time

After a port is successfully reset or resumed, the USB System Software is allowed to access the device attached to the port immediately and it is expected to respond to data transfers. The device may ignore any data transfers during the recovery interval.

9.2.6.3 Set Address Processing

After the reset/resume recovery interval(0), if a device receives a SetAddress() request, the device shall be able to complete processing of the request and be able to successfully complete the Status stage of the request within 50 ms. In the case of the SetAddress() request, the Status stage successfully completes when the device sends an ACK Transaction Packet in response to Status stage IN Transaction Packet.

After successful completion of the Status stage, the device shall be able to accept Setup packets addressed to the new address. Also, after successful completion of the Status stage, the device shall not respond to tokens sent to the old address (unless, of course, the old and new address is the same).

9.2.6.4 Standard Device Requests

For standard device requests that require no Data stage, a device shall be able to complete the request and be able to successfully complete the Status stage of the request within 50 ms of receipt of the request. This limitation applies to requests targeted to the device, interface, or endpoint.

For standard device requests that require data stage transfer to the host, the device shall be able to return the first data packet to the host within 500 ms of receipt of the request. For subsequent data packets, if any, the device shall be able to return them within 500 ms of successful completion of the transmission of the previous packet. The device shall then be able to successfully complete the status stage within 50 ms after returning the last data packet.

For standard device requests that require a data stage transfer to the device, the 5-second limit applies. This means that the device shall be capable of accepting all data packets from the host and successfully completing the Status stage if the host provides the data at the maximum rate at which the device can accept it. Delays between packets introduced by the host add to the time allowed for the device to complete the request.

9.2.6.5 Class-specific Requests

Unless specifically exempted in the class document, all class-specific requests shall meet the timing limitations for standard device requests. If a class document provides an exemption, the exemption may only be specified on a request-by-request basis.

A class document may require that a device respond more quickly than is specified in this section. Faster response may be required for standard and class-specific requests.

9.2.6.6 Speed Dependent Descriptors

A device capable of operation at SuperSpeed shall be capable of operating at one of the USB 2.0 defined speeds. The device always knows its operational speed as part of connection processing (See Section 10.5for more details on the connection process). A device operates at a single speed after completing the reset sequence. In particular, there is no speed switch during normal operation. However, a SuperSpeed capable device may have configurations that are speed dependent. That is, it may have some configurations that are only possible when operating at SuperSpeed or some that are only possible when operating at high-speed. SuperSpeed capable devices shall support reporting the speeds they can operate at. Note that a USB 3.0 hub is the only device that is allowed to operate at both USB2.0 and SuperSpeed simultaneously.

A SuperSpeed capable device responds with descriptor information that is valid for the current operating speed. For example, when a device is asked for configuration descriptors, it only returns those for the current operating speed (e.g., high-speed). When operating in SuperSpeed mode, the device shall report the other speeds it can operate via its BOS descriptor (see Section 9.6.2).

Note that when operating at USB 2.0 speeds, the device shall report the other USB 2.0 speeds it supports using the standard mechanism defined in the USB 2.0 specification in addition to reporting the other speeds supported by the device in its BOS descriptor. Devices with a value of at least 0210H Devices in the *bcdUSB* field of their device descriptor shall support GetDescriptor (BOS Descriptor) requests.

NOTE

These descriptors are not retrieved unless the host explicitly issues the corresponding GetDescriptor requests.

9.2.7 Request Error

When a request not defined for the device, is inappropriate for the current setting of the device, or has values that are not compatible with the request is received, a Request Error exists. The device deals with the Request Error by returning a STALL Transaction Packet in response to the next Data stage transaction or in the Status stage of the message. It is preferred that the STALL Transaction Packet be returned at the next Data stage transaction to avoid unnecessary bus activity.

9.3 USB Device Requests

All USB devices respond to requests from the host on the device's Default Control Pipe. These requests are made using control transfers. The request and the request's parameters are sent to the device in the Setup packet. The host is responsible for establishing the values passed in the fields listed in Table 9-2. Every Setup packet has eight bytes.

Table 9-2. Format of Setup Data

Offset	Field	Size	Value	Description
0	*bmRequestType*	1	Bitmap	Characteristics of request: D7: Data transfer direction 0 = Host-to-device 1 = Device-to-host D6...5: Type 0 = Standard 1 = Class 2 = Vendor 3 = Reserved D4...0: Recipient 0 = Device 1 = Interface 2 = Endpoint 3 = Other 4...31 = Reserved
1	*bRequest*	1	Value	Specific request (refer to Table 9-3)
2	*wValue*	2	Value	Word-sized field that varies according to request
4	*wIndex*	2	Index or Offset	Word-sized field that varies according to request; typically used to pass an index or offset
6	*wLength*	2	Count	Number of bytes to transfer if there is a Data stage

9.3.1 bmRequestType

This bitmapped field identifies the characteristics of the specific request. In particular, this field identifies the direction of data transfer in the second phase of the control transfer. The state of the *Direction* bit is ignored if the *wLength* field is zero, signifying there is no Data stage.

The USB 3.0 Specification defines a series of standard requests that all devices shall support. These are enumerated in Table 9-3. In addition, a device class may define additional requests. A device vendor may also define requests supported by the device.

Requests may be directed to the device, an interface on the device, or a specific endpoint on a device. This field also specifies the intended recipient of the request. When an interface is specified, the *wIndex* field identifies the interface. When an endpoint is specified, the *wIndex* field identifies the endpoint.

9.3.2 bRequest

This field specifies the particular request. The *Type* bits in the *bmRequestType* field modify the meaning of this field. This specification defines values for the *bRequest* field only when the bits are reset to zero, indicating a standard request (refer to Table 9-3).

9.3.3 wValue

The contents of this field vary according to the request. It is used to pass a parameter to the device, specific to the request.

9.3.4 wIndex

The contents of this field vary according to the request. It is used to pass a parameter to the device, specific to the request.

The *wIndex* field is often used in requests to specify an endpoint or an interface. Figure 9-2 shows the format of *wIndex* when it is used to specify an endpoint.

D7	D6	D5	D4	D3	D2	D1	D0
Direction	Reserved (Reset to zero)			Endpoint Number			
D15	D14	D13	D12	D11	D10	D9	D8
Reserved (Reset to zero)							

Figure 9-2. wIndex Format when Specifying an Endpoint

The *Direction* bit is set to zero to indicate the OUT endpoint with the specified *Endpoint Number* and to one to indicate the IN endpoint. In the case of a control pipe, the request should have the *Direction* bit set to zero but the device may accept either value of the *Direction* bit.

Figure 9-3 shows the format of *wIndex* when it is used to specify an interface.

D7	D6	D5	D4	D3	D2	D1	D0
Interface Number							
D15	D14	D13	D12	D11	D10	D9	D8
Reserved (Reset to zero)							

Figure 9-3. wIndex Format when Specifying an Interface

9.3.5 wLength

This field specifies the length of the data transferred during the second phase of the control transfer. The direction of data transfer (host-to-device or device-to-host) is indicated by the *Direction* bit of the *bmRequestType* field. If this field is zero, there is no data transfer phase.

On an input request, a device shall never return more data than is indicated by the *wLength* value; it may return less. On an output request, *wLength* will always indicate the exact amount of data to be sent by the host. Device behavior is undefined if the host should send more or less data than is specified in *wLength.*

9.4 Standard Device Requests

This section describes the standard device requests defined for all USB devices. Table 9-3 outlines the standard device requests, while Table 9-4 and Table 9-5 give the standard request codes and descriptor types, respectively.

USB devices shall respond to standard device requests, even if the device has not yet been assigned an address or has not been configured.

Table 9-3. Standard Device Requests

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00000000B 00000001B 00000010B	CLEAR_FEATURE	Feature Selector	Zero Interface Endpoint	Zero	None
10000000B	GET_CONFIGURATION	Zero	Zero	One	Configuration Value
10000000B	GET_DESCRIPTOR	Descriptor Type and Descriptor Index	Zero or Language ID	Descriptor Length	Descriptor
10000001B	GET_INTERFACE	Zero	Interface	One	Alternate Interface
10000000B 10000001B 10000010B	GET_STATUS	Zero	Zero Interface Endpoint	Two	Device, Interface, or Endpoint Status
00000000B	SET_ADDRESS	Device Address	Zero	Zero	None
00000000B	SET_CONFIGURATION	Configuration Value	Zero	Zero	None
00000000B	SET_DESCRIPTOR	Descriptor Type and Descriptor Index	Zero or Language ID	Descriptor Length	Descriptor
00000000B 00000001B 00000010B	SET_FEATURE	Feature Selector	Zero Interface Endpoint	Zero	None
00000001B	SET_INTERFACE	Alternate Setting	Interface	Zero	None
00000000B	SET_ISOCH_DELAY	Delay in ns	Zero	Zero	None
00000000B	SET_SEL	U1 or U2 System Exit Latency	Zero	Zero	None
10000010B	SYNCH_FRAME	Zero	Endpoint	Two	Frame Number

Table 9-4. Standard Request Codes

bRequest	Value
GET_STATUS	0
CLEAR_FEATURE	1
Reserved for future use	2
SET_FEATURE	3
Reserved for future use	4
SET_ADDRESS	5
GET_DESCRIPTOR	6
SET_DESCRIPTOR	7
GET_CONFIGURATION	8
SET_CONFIGURATION	9
GET_INTERFACE	10
SET_INTERFACE	11
SYNCH_FRAME	12
SET_SEL	48
SET_ISOCH_DELAY	49

Table 9-5. Descriptor Types

Descriptor Types	Value
DEVICE	1
CONFIGURATION	2
STRING	3
INTERFACE	4
ENDPOINT	5
Reserved	6
Reserved	7
INTERFACE_POWER[1]	8
OTG	9
DEBUG	10
INTERFACE_ASSOCIATION	11
BOS	15
DEVICE CAPABILITY	16
SUPERSPEED_USB_ENDPOINT_COMPANION	48

[1] The INTERFACE_POWER descriptor is defined in the current revision of the *USB Interface Power Management Specification*.

Feature selectors are used when enabling or setting features, such as function remote wakeup, specific to a device, interface, or endpoint. The values for the feature selectors are given in Table 9-6.

Table 9-6. Standard Feature Selectors

Feature Selector	Recipient	Value
ENDPOINT_HALT	Endpoint	0
FUNCTION_SUSPEND	Interface	0
TEST_MODE	Device	2
U1_ENABLE	Device	48
U2_ENABLE	Device	49
LTM_ENABLE	Device	50

If an unsupported or invalid request is made to a USB device, the device responds by returning a STALL Transaction Packet in the Data or Status stage of the request. If the device detects the error in the Setup stage, it is preferred that the device returns a STALL Transaction Packet at the earlier of the Data or Status stage. Receipt of an unsupported or invalid request does NOT cause the *Halt* feature on the control pipe to be set. If for any reason, the device becomes unable to communicate via its Default Control Pipe due to an error condition, the device shall be reset to clear the condition and restart the Default Control Pipe.

9.4.1 Clear Feature

This request is used to clear or disable a specific feature.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00000000B 00000001B 00000010B	CLEAR_FEATURE	Feature Selector	Zero Interface Endpoint	Zero	None

Feature selector values in wValue shall be appropriate to the recipient. Only device feature selector values may be used when the recipient is a device, only interface feature selector values may be used when the recipient is an interface, and only endpoint feature selector values may be used when the recipient is an endpoint.

Refer to Table 9-6 for a definition of which feature selector values are defined for which recipients.

A ClearFeature() request that references a feature that cannot be cleared, that does not exist, or that references an interface or a endpoint that does not exist, will cause the device to respond with a Request Error.

If *wLength* is non-zero, then the device behavior is not specified.

Default state: Device behavior when this request is received while the device is in the Default state is not specified.

Address state: This request is valid when the device is in the Address state; references to interfaces or to endpoints other than the Default Control Pipe shall cause the device to respond with a Request Error.

Configured state: This request is valid when the device is in the Configured state.

NOTE

The Test_Mode feature cannot be cleared by the ClearFeature() request. In addition the device shall process a Clear Feature (U1_Enable or U2_Enable or LTM_Enable) only if the device is in the configured state.

9.4.2 Get Configuration

This request returns the current device configuration value.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
10000000B	GET_CONFIGURATION	Zero	Zero	One	Configuration Value

If the returned value is zero, the device is not configured.

If *wValue*, *wIndex*, or *wLength* are not as specified above, then the device behavior is not specified.

Default state: Device behavior when this request is received while the device is in the Default state is not specified.

Address state: The value zero shall be returned.

Configured state: The non-zero *bConfigurationValue* of the current configuration shall be returned.

9.4.3 Get Descriptor

This request returns the specified descriptor if the descriptor exists.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
10000000B	GET_DESCRIPTOR	Descriptor Type and Descriptor Index	Zero or Language ID (refer to Section 9.6.7)	Descriptor Length	Descriptor

The wValue field specifies the descriptor type in the high byte (refer to Table 9-5) and the descriptor index in the low byte. The descriptor index is used to select a specific descriptor (only for configuration and string descriptors) when several descriptors of the same type are implemented in a device. For example, a device can implement several configuration descriptors. For other standard descriptors that can be retrieved via a GetDescriptor() request, a descriptor index of zero shall be used. The range of values used for a descriptor index is from 0 to one less than the number of descriptors of that type implemented by the device.

The wIndex field specifies the Language ID for string descriptors or is reset to zero for other descriptors. The wLength field specifies the number of bytes to return. If the descriptor is longer than the wLength field, only the initial bytes of the descriptor are returned. If the descriptor is shorter than the wLength field, the device indicates the end of the control transfer by sending a short packet when further data is requested.

The standard request to a device supports four types of descriptors: device, configuration, BOS (Binary device Object Store) and string. As noted in Section 9.2.6.6 a device operating in SuperSpeed mode reports the other speeds it supports via the BOS descriptor and shall not support the device_qualifier and other_speed_configuration descriptors. A request for a configuration descriptor returns the configuration descriptor, all interface descriptors, endpoint descriptors and endpoint companion descriptors (when operating in SuperSpeed mode) for all of the interfaces in a single request. The first interface descriptor follows the configuration descriptor. The endpoint descriptors for the first interface follow the first interface descriptor. In addition, SuperSpeed devices shall return Endpoint Companion descriptors for each of the endpoints in that interface to return the endpoint capabilities required for SuperSpeed capable devices, which would not fit inside the existing endpoint descriptor footprint. If there are additional interfaces, their interface descriptor, endpoint descriptors and endpoint companion descriptors (when operating in SuperSpeed mode) follow the first interface's endpoint and endpoint companion (when operating in SuperSpeed mode) descriptors.

This specification also defines a flexible and extensible framework for describing and adding device-level capabilities to the set of USB standard specifications. The BOS descriptor (Binary device Object Store, see Section 9.6.2) defines a root descriptor that is similar to the configuration descriptor, and is the base descriptor for accessing a family of related descriptors. A host can read a BOS descriptor and learn from the *wTotalLength* field the entire size of the device-level descriptor set, or it can read in the entire BOS descriptor set of device capabilities. There is no way for a host to read individual device capability descriptors. The entire set can only be accessed via reading the BOS descriptor with a GetDescriptor() request and using the length reported in the *wTotalLength* field.

Class-specific and/or vendor-specific descriptors follow the standard descriptors they extend or modify.

All devices shall provide a device descriptor and at least one configuration descriptor. If a device does not support a requested descriptor, it responds with a Request Error.

Default state: This is a valid request when the device is in the Default state.

Address state: This is a valid request when the device is in the Address state.

Configured state: This is a valid request when the device is in the Configured state.

9.4.4 Get Interface

This request returns the selected alternate setting for the specified interface.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
10000001B	GET_INTERFACE	Zero	Interface	One	Alternate Setting

Some USB devices have configurations with interfaces that have mutually exclusive settings. This request allows the host to determine the currently selected alternate setting.

If *wValue* or *wLength* are not as specified above, then the device behavior is not specified.

If the interface specified does not exist, then the device responds with a Request Error.

Default state: Device behavior when this request is received while the device is in the Default state is not specified.

Address state: A Request Error response is given by the device.

Configured state: This is a valid request when the device is in the Configured state.

9.4.5 Get Status

This request returns status for the specified recipient.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
10000000B 10000001B 10000010B	GET_STATUS	Zero	Zero Interface Endpoint	Two	Device, Interface, or Endpoint Status

The *Recipient* bits of the bmRequestType field specify the desired recipient. The data returned is the current status of the specified recipient. If the recipient is an endpoint then the lower byte of *wIndex* identifies the endpoint whose status is being queried.

If *wValue* or *wLength* are not as specified above, or if *wIndex* is non-zero for a device status request, then the behavior of the device is not specified.

If an interface or an endpoint is specified that does not exist, then the device responds with a Request Error.

Default state: Device behavior when this request is received while the device is in the Default state is not specified.

Address state: If an interface or an endpoint other than the Default Control Pipe is specified, then the device responds with a Request Error.

Configured state: If an interface or an endpoint that does not exist is specified, then the device responds with a Request Error.

A GetStatus() request to a device returns the information shown in Figure 9-4.

D7	D6	D5	D4	D3	D2	D1	D0
Reserved (Reset to zero)				U2 Enable	U1 Enable	Remote Wakeup	Self Powered
D15	D14	D13	D12	D11	D10	D9	D8
Reserved (Reset to zero)							

Figure 9-4. Information Returned by a GetStatus() Request to a Device

The Self Powered field indicates whether the device is currently self-powered. If D0 is reset to zero, the device is bus-powered. If D0 is set to one, the device is self-powered. The Self Powered field may not be changed by the SetFeature() or ClearFeature() requests.

The Remote Wakeup field is reserved and must be set to zero by SuperSpeed devices. SuperSpeed devices use the Function Remote Wake enable/disable field to indicate whether they are enabled for Remote Wake.

The U1 Enable field indicates whether the device is currently enabled to initiate U1 entry. If D2 is set to zero, the device is disabled from initiating U1 entry otherwise it is enabled to initiate U1 entry. The U1 Enable field can be modified by the SetFeature() and ClearFeature() requests using the U1_ENABLE feature selector. This field is reset to zero when the device is reset.

The U2 Enable field indicates whether the device is currently enabled to initiate U2 entry. If D3 is set to zero, the device is disabled from initiating U2 entry otherwise it is enabled to initiate U2 entry. The U2 Enable field can be modified by the SetFeature() and ClearFeature() requests using the U2_ENABLE feature selector. This field is reset to zero when the device is reset.

A GetStatus() request to the first interface in a function returns the information shown in Figure 9-5.

D7	D6	D5	D4	D3	D2	D1	D0
Reserved (Reset to zero)						Function Remote Wakeup	Function Remote Wake Capable
D15	D14	D13	D12	D11	D10	D9	D8
Reserved (Reset to zero)							

Figure 9-5. Information Returned by a GetStatus() Request to an Interface

The *Function Remote Wake Capable* field indicates whether the function supports remote wake up. The *Function Remote Wakeup* field indicates whether the function is currently enabled to request remote wakeup. The default mode for functions that support function remote wakeup is disabled. If D1 is reset to zero, the ability of the function to signal remote wakeup is disabled. If D1 is set to one, the ability of the function to signal remote wakeup is enabled. The *Function Remote Wakeup* field can be modified by the SetFeature() requests using the *FUNCTION_SUSPEND* feature selector. This *Function Remote Wakeup* is reset to zero when the function is reset.

A GetStatus() request to the any other interface in a function shall return all zeros.

A GetStatus() request to an endpoint returns the information shown in Figure 9-6.

D7	D6	D5	D4	D3	D2	D1	D0
Reserved (Reset to zero)							Halt
D15	D14	D13	D12	D11	D10	D9	D8
Reserved (Reset to zero)							

Figure 9-6. Information Returned by a GetStatus() Request to an Endpoint

The *Halt* feature is required to be implemented for all interrupt and bulk endpoint types. If the endpoint is currently halted, then the Halt feature is set to one. Otherwise, the Halt feature is reset to zero. The Halt feature may optionally be set with the SetFeature(ENDPOINT_HALT) request. When set by the SetFeature() request, the endpoint exhibits the same stall behavior as if the field had been set by a hardware condition. If the condition causing a halt has been removed, clearing the *Halt* feature via a ClearFeature(ENDPOINT_HALT) request results in the endpoint no longer returning a STALL Transaction Packet. Regardless of whether an endpoint has the *Halt* feature set, a ClearFeature(ENDPOINT_HALT) request always results in the data sequence being reinitialized to ZERO, and if Streams are enabled, the Stream State Machine shall be reinitialized to the *Disabled* state. The *Halt* feature is reset to zero after either a SetConfiguration() or SetInterface() request even if the requested configuration or interface is the same as the current configuration or interface.

SuperSpeed USB devices do not support functional stall on a control endpoints and hence do not require the *Halt* feature be implemented for any Control endpoints.

9.4.6 Set Address

This request sets the device address for all future device accesses.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00000000B	SET_ADDRESS	Device Address	Zero	Zero	None

The wValue field specifies the device address to use for all subsequent accesses.

The Status stage after the initial Setup packet assume the same device address as the Setup packet. The USB device does not change its device address until after the Status stage of this request is completed successfully. Note that this is a difference between this request and all other requests. For all other requests, the operation indicated shall be completed before the Status stage.

If the specified device address is greater than 127, or if *wIndex* or *wLength* is non-zero, then the behavior of the device is not specified.

Default state: If the address specified is non-zero, then the device shall enter the Address state; otherwise, the device remains in the Default state (this is not an error condition).

Address state: If the address specified is zero, then the device shall enter the Default state; otherwise, the device remains in the Address state but uses the newly-specified address.

Configured state: Device behavior when this request is received while the device is in the Configured state is not specified.

9.4.7 Set Configuration

This request sets the device configuration.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00000000B	SET_CONFIGURATION	Configuration Value	Zero	Zero	None

The lower byte of the wValue field specifies the desired configuration. This configuration value shall be zero or match a configuration value from a configuration descriptor. If the configuration value is zero, the device is placed in its Address state. The upper byte of the *wValue* field is reserved.

If *wIndex*, *wLength*, or the upper byte of *wValue* is non-zero, then the behavior of this request is not specified.

Default state: Device behavior when this request is received while the device is in the Default state is not specified.

Address state: If the specified configuration value is zero, then the device remains in the Address state. If the specified configuration value matches the configuration value from a configuration descriptor, then that configuration is selected and the device enters the Configured state. Otherwise, the device responds with a Request Error.

Configured state: If the specified configuration value is zero, then the device enters the Address state. If the specified configuration value matches the configuration value from a configuration descriptor, then that configuration is selected and the device remains in the Configured state. Otherwise, the device responds with a Request Error.

9.4.8 Set Descriptor

This request is optional and may be used to update existing descriptors or new descriptors may be added.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00000000B	SET_DESCRIPTOR	Descriptor Type and Descriptor Index	Language ID (refer to Section 9.6.7) or zero	Descriptor Length	Descriptor

The wValue field specifies the descriptor type in the high byte (refer to Table 9-5) and the descriptor index in the low byte. The descriptor index is used to select a specific descriptor (only for configuration and string descriptors) when several descriptors of the same type are implemented in a device. For example, a device can implement several configuration descriptors. For other standard descriptors that can be set via a SetDescriptor() request, a descriptor index of zero shall be used. The range of values used for a descriptor index is from 0 to one less than the number of descriptors of that type implemented by the device.

The wIndex field specifies the Language ID for string descriptors or is reset to zero for other descriptors. The wLength field specifies the number of bytes to transfer from the host to the device.

The only allowed values for descriptor type are device, configuration, and string descriptor types.

If this request is not supported, the device will respond with a Request Error.

Default state: Device behavior when this request is received while the device is in the Default state is not specified.

Address state: If supported, this is a valid request when the device is in the Address state.

Configured state: If supported, this is a valid request when the device is in the Configured state.

9.4.9 Set Feature

This request is used to set or enable a specific feature.

bmRequestType	bRequest	wValue	wIndex		wLength	Data
00000000B 00000001B 00000010B	SET_FEATURE	Feature Selector	Test Selector Suspend Options	Zero Interface Endpoint	Zero	None

Feature selector values in wValue shall be appropriate to the recipient. Only device feature selector values may be used when the recipient is a device; only interface feature selector values may be used when the recipient is an interface, and only endpoint feature selector values may be used when the recipient is an endpoint. If the recipient is an endpoint then the lower byte of *wIndex* identifies the endpoint.

Refer to Table 9-6 for a definition of which feature selector values are defined for which recipients.

The FUNCTION_SUSPEND feature is only defined for an interface recipient. The lower byte of wIndex shall be set to the first interface that is part of that function.

The TEST_MODE feature is only defined for a device recipient (i.e., bmRequestType = 0) and the lower byte of wIndex shall be zero. Setting the TEST_MODE feature puts the device upstream facing port into test mode. The device will respond with a request error if the request contains an invalid test selector. The transition to test mode shall be complete no later than 3 ms after the completion of the status stage of the request. The transition to test mode of an upstream facing port shall not happen until after the status stage of the request. The power to the device shall be cycled to exit test mode of an upstream facing port of a device. See the Link Layer Chapter for definitions of each test mode. A device shall support the TEST_MODE feature when in the Default, Address, or Configured SuperSpeed device states.

The U1/U2_ENABLE feature is only defined for a device recipient and wIndex shall be set to zero. Setting the U1/U2_ENABLE feature allows the device to initiate U1/U2 entry respectively. A device shall support the U1/U2ENABLE feature when in the Configured SuperSpeed state only. System software must not enable the device to initiate U1 if the time for U1 System Exit Latency initiated by Host plus one Bus Interval time is greater than the minimum of the service intervals of any periodic endpoints in the device. In addition, System software must not enable the device to initiate U2 if the time for U2 System Exit Latency initiated by Host plus one Bus Interval time is greater than the minimum of the service intervals of any periodic endpoints in the device.

The LTM_ENABLE feature is only defined for a device recipient and wIndex shall be set to zero. Setting the LTM_ENABLE feature allows the device to send Latency Tolerance Messages. A device shall support the LTM_ENABLE feature if it is in the Configured SuperSpeed state and supports the LTM capability.

A SetFeature() request that references a feature that cannot be set or that does not exist causes a STALL Transaction Packet to be returned in the Status stage of the request.

Table 9-7. Test Mode Selectors

Value	Description
00H – 05H	Reserved
06H	COMPLIANCE_MODE
07H	LOOPBACK_MODE
08H-3FH	Reserved for standard test selectors
3FH-BFH	Reserved
C0H-FFH	Reserved for vendor-specific test modes.

If the feature selector is *TEST_MODE*, then the most significant byte of *wIndex* is used to specify the specific test mode. The recipient of a SetFeature(TEST_MODE…) shall be the device; i.e., the lower byte of *wIndex* shall be zero and the *bmRequestType* shall be set to zero. The device shall have its power cycled to exit test mode. The valid test mode selectors are listed in Table 9-7. See the Link Layer Chapter for more information about the specific test modes.

Table 9-8. Suspend Options

Bit	Description
0	**Value** **Meaning** 0 Normal operation state (Default) 1 Low power suspend state
1	**Value** **Meaning** 0 Function Remote Wake Disabled (Default) 1 Function Remote Wake Enabled
2-7	Reserved

If the feature selector is *FUNCTION_SUSPEND*, then the most significant byte of *wIndex* is used to specify Suspend options. The recipient of a SetFeature (FUNCTION_SUSPEND…) shall be the first interface in the function; and hence the *bmRequestType* shall be set to one. The valid encodings for the *FUNCTION_SUSPEND* suspend options are listed in Table 9-8.

If *wLength* is non-zero, then the behavior of the device is not specified.

If an endpoint or interface is specified that does not exist, then the device responds with a Request Error.

Default state: A device shall be able to accept a SetFeature(TEST_MODE, TEST_SELECTOR) request when in the Default State. Device behavior for other SetFeature requests while the device is in the Default state is not specified.

Address state: If an interface or an endpoint other than the Default Control Pipe is specified then the device responds with a Request Error. If the device receives a SetFeature(U1/U2 Enable or LTM Enable or FUNCTION_SUSPEND) then the device responds with a Request Error.

Configured state: This is a valid request when the device is in the Configured state.

9.4.10 Set Interface

This request allows the host to select an alternate setting for the specified interface.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00000001B	SET_INTERFACE	Alternate Setting	Interface	Zero	None

Some USB devices have configurations with interfaces that have mutually exclusive settings. This request allows the host to select the desired alternate setting. If a device only supports a default setting for the specified interface, then a STALL Transaction Packet may be returned in the Status stage of the request. This request cannot be used to change the set of configured interfaces (the SetConfiguration() request shall be used instead).

If the interface or the alternate setting does not exist, then the device responds with a Request Error. If *wLength* is non-zero, then the behavior of the device is not specified.

Default state: Device behavior when this request is received while the device is in the Default state is not specified.

Address state: The device shall respond with a Request Error.

Configured state: This is a valid request when the device is in the Configured state.

9.4.11 Set Isochronous Delay

This request informs the device of the average delay from the time a host transmits a packet to the time it is received by the device.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00000000B	SET_ISOCH_DELAY	Delay in ns	Zero	Zero	None

The wValue field specifies a delay from 0 to 65535 nanoseconds. This delay represents the average time from when the host starts transmitting the first framing symbol of the packet to when the device receives the first framing symbol of that packet.

If *wIndex* or *wLength* is non-zero, then the behavior of this request is not specified.

Default state: Device behavior when this request is received while the device is in the Default state is not specified.

Address state: Device behavior when this request is received while the device is in the Address state is not specified.

Configured state: This is a valid request when the device is in the Configured state.

9.4.12 Set SEL

This request sets the System Exit Latency.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00000000B	SET_SEL	U1/U2 SEL and System Exit Latency	Zero	Zero	None

The wValue field contains the System Exit Latency value in the lower three nibbles (bits 11:0). The encoding of these bits depends on the value of the upper nibble (bits 15:12).

If the Upper nibble is 0 then the host software is setting the worst case latency to transition all links in the path between device and host from U1 to U0, when initiated by device. The following are the permissible values for the lower three nibbles:

Value	Meaning
00H	Zero.
01H	Less than 1 µs
02H	Less than 2 µs
03H	Less than 3 µs
04H	Less than 4 µs
…	…
0FH	Less than 15 µs
10H – FFFH	Reserved

If the Upper nibble is 1, then the host software is setting the worst case latency to transition all links in the path between device and host from U2 to U0, when initiated by device. The following are the permissible values for the lower three nibbles:

Value	Meaning
00H	Zero.
01H	Less than 1 µs
02H	Less than 2 µs
03H	Less than 3 µs
04H	Less than 4 µs
…	…
07FFH	Less than 2047 µs
0800H – FFFH	**Reserved**

If *wIndex*, *wLength*, or the upper nibble of *wValue* is not set to 0 or 1, then the behavior of this request is not specified.

Default state: Device behavior when this request is received while the device is in the Default state is not specified.

Address state: Device behavior when this request is received while the device is in the Address state is not specified.

Configured state: This is a valid request when the device is in the Configured state.

9.4.13 Synch Frame

This request is used to set and then report an endpoint's synchronization frame.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
10000010B	synch_frame	Zero	Endpoint	Two	Frame Number

When an endpoint supports isochronous transfers, the endpoint may also require per-frame transfers to vary in size according to a specific pattern. The host and the endpoint must agree on which frame the repeating pattern begins. The number of the frame in which the pattern began is returned to the host.

If a SuperSpeed device supports the Synch Frame request, it shall internally synchronize itself to the zeroth microframe and have a time notion of classic frame. Only the frame number is used to synchronize and reported by the device endpoint (i.e., no microframe number). The endpoint must synchronize to the zeroth microframe.

This value is only used for isochronous data transfers using implicit pattern synchronization. If *wValue* is non-zero or *wLength* is not two, then the behavior of the device is not specified.

If the specified endpoint does not support this request, then the device will respond with a Request Error.

Default state: Device behavior when this request is received while the device is in the Default state is not specified

Address state: The device shall respond with a Request Error.

Configured state: This is a valid request when the device is in the Configured state.

9.5 Descriptors

USB devices report their attributes using descriptors. A descriptor is a data structure with a defined format. Each descriptor begins with a byte-wide field that contains the total number of bytes in the descriptor followed by a byte-wide field that identifies the descriptor type.

Using descriptors allows concise storage of the attributes of individual configurations because each configuration may reuse descriptors or portions of descriptors from other configurations that have the same characteristics. In this manner, the descriptors resemble individual data records in a relational database.

Where appropriate, descriptors contain references to string descriptors that provide displayable information describing a descriptor in human-readable form. The inclusion of string descriptors is optional. However, the reference fields within descriptors are mandatory. If a device does not support string descriptors, string reference fields shall be reset to zero to indicate no string descriptor is available.

If a descriptor returns with a value in its length field that is less than defined by this specification, the descriptor is invalid and should be rejected by the host. If the descriptor returns with a value in its length field that is greater than defined by this specification, the extra bytes are ignored by the host, but the next descriptor is located using the length returned rather than the length expected.

A device may return class- or vendor-specific descriptors in two ways:

1. If the class or vendor specific descriptors use the same format as standard descriptors (e.g., start with a length byte and followed by a type byte), they shall be returned interleaved with standard descriptors in the configuration information returned by a GetDescriptor(Configuration) request. In this case, the class or vendor-specific descriptors shall follow a related standard descriptor they modify or extend.
2. If the class or vendor specific descriptors are independent of configuration information or use a non-standard format, a GetDescriptor() request specifying the class or vendor specific descriptor type and index may be used to retrieve the descriptor from the device. A class or vendor specification will define the appropriate way to retrieve these descriptors.

9.6 Standard USB Descriptor Definitions

The standard descriptors defined in this specification may only be modified or extended by revision of this specification.

9.6.1 Device

A device descriptor describes general information about a USB device. It includes information that applies globally to the device and all of the device's configurations. A USB device has only one device descriptor.

The DEVICE descriptor of a SuperSpeed capable device has a version number of 3.0 (0300H).

The *bcdUSB* field contains a BCD version number. The value of the *bcdUSB* field is 0xJJMN for version JJ.M.N (JJ – major version number, M – minor version number, N – sub-minor version number), e.g., version 2.1.3 is represented with value 0x0213 and version 3.0 is represented with a value of 0x0300.

The *bNumConfigurations* field indicates the number of configurations at the current operating speed. Configurations for the other operating speed are not included in the count. If there are specific configurations of the device for specific speeds, the *bNumConfigurations* field only reflects the number of configurations for a single speed, not the total number of configurations for both speeds.

If the device is operating at SuperSpeed, the *bMaxPacketSize0* field shall be set to 09H (see Table 9-9) indicating a 512 byte maximum packet. SuperSpeed operation does not allow other maximum packet sizes for the Default Control Pipe (Endpoint 0) control endpoint.

All USB devices have a Default Control Pipe. The maximum packet size of a device's Default Control Pipe is described in the device descriptor. Endpoints specific to a configuration and its interface(s) are described in the configuration descriptor. A configuration and its interface(s) do not include an endpoint descriptor for the Default Control Pipe. Other than the maximum packet size, the characteristics of the Default Control Pipe are defined by this specification and are the same for all SuperSpeed devices.

The bNumConfigurations field identifies the number of configurations the device supports. Table 9-9 shows the standard device descriptor.

Table 9-9. Standard Device Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of this descriptor in bytes
1	*bDescriptorType*	1	Constant	DEVICE Descriptor Type
2	*bcdUSB*	2	BCD	USB Specification Release Number in Binary-Coded Decimal (i.e., 2.10 is 210H). This field identifies the release of the USB Specification with which the device and its descriptors are compliant.
4	*bDeviceClass*	1	Class	Class code (assigned by the USB-IF). If this field is reset to zero, each interface within a configuration specifies its own class information and the various interfaces operate independently. If this field is set to a value between 1 and FEH, the device supports different class specifications on different interfaces and the interfaces may not operate independently. This value identifies the class definition used for the aggregate interfaces. If this field is set to FFH, the device class is vendor-specific.
5	*bDeviceSubClass*	1	SubClass	Subclass code (assigned by the USB-IF). These codes are qualified by the value of the *bDeviceClass* field. If the *bDeviceClass* field is reset to zero, this field shall also be reset to zero. If the *bDeviceClass* field is not set to FFH, all values are reserved for assignment by the USB-IF.
6	*bDeviceProtocol*	1	Protocol	Protocol code (assigned by the USB-IF). These codes are qualified by the value of the *bDeviceClass* and the *bDeviceSubClass* fields. If a device supports class-specific protocols on a device basis as opposed to an interface basis, this code identifies the protocols that the device uses as defined by the specification of the device class. If this field is reset to zero, the device does not use class-specific protocols on a device basis. However, it may use class-specific protocols on an interface basis. If this field is set to FFH, the device uses a vendor-specific protocol on a device basis.
7	*bMaxPacketSize0*	1	Number	Maximum packet size for endpoint zero. The bMaxPacketSize0 value is used as the exponent for a $2^{bMaxPacketSize0}$ value; e.g., a bMaxPacketSize0 of 4 means a Max Packet size of.16 ($2^4 \rightarrow 8$). 09H is the only valid value in this field when operating in SuperSpeed mode.
8	*idVendor*	2	ID	Vendor ID (assigned by the USB-IF)
10	*idProduct*	2	ID	Product ID (assigned by the manufacturer)
12	*bcdDevice*	2	BCD	Device release number in binary-coded decimal
14	*iManufacturer*	1	Index	Index of string descriptor describing manufacturer
15	*iProduct*	1	Index	Index of string descriptor describing product
16	*iSerialNumber*	1	Index	Index of string descriptor describing the device's serial number
17	*bNumConfigurations*	1	Number	Number of possible configurations

9.6.2 Binary Device Object Store (BOS)

This section defines a flexible and extensible framework for describing and adding device-level capabilities to the set of USB standard specifications. As mentioned above, there exists a device descriptor, but all device-level capability extensions are defined using the following framework.

The BOS descriptor defines a root descriptor that is similar to the configuration descriptor, and is the base descriptor for accessing a family of related descriptors. A host can read a BOS descriptor and learn from the *wTotalLength* field the entire size of the device-level descriptor set, or it can read in the entire BOS descriptor set of device capabilities. The host accesses this descriptor using the GetDescriptor() request. The descriptor type in the GetDescriptor() request is set to BOS (see Table 9-10). There is no way for a host to read individual device capability descriptors. The entire set can only be accessed via reading the BOS descriptor with a GetDescriptor() request and using the length reported in the *wTotalLength* field.

Table 9-10. BOS Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of descriptor
1	*bDescriptorType*	1	Constant	BOS Descriptor type
2	*wTotalLength*	2	Number	Length of this descriptor and all of its sub descriptors
4	*bNumDeviceCaps*	1	Number	The number of separate device capability descriptors in the BOS

Individual technology-specific or generic device-level capabilities are reported via Device Capability descriptors. The format of the Device Capability descriptor is defined in Table 9-11. The Device Capability descriptor has a generic header, with a sub-type field (*bDevCapabilityType*) which defines the layout of the remainder of the descriptor. The codes for *bDevCapabilityType* are defined in Table 9-12.

Table 9-11. Format of a Device Capability Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of this descriptor.
1	*bDescriptorType*	1	Constant	Descriptor type: DEVICE CAPABILITY Type.
2	*bDevCapabilityType*	1	Number	Valid values are listed in Table 9-12.
3	*Capability-Dependent*	Var	Variable	Capability-specific format.

Device Capability descriptors are always returned as part of the BOS information returned by a GetDescriptor(BOS) request. A Device Capability cannot be directly accessed with a GetDescriptor() or SetDescriptor() request.

Table 9-12. Device Capability Type Codes

Capability Code	Value	Description
Wireless_USB	01H	Defines the set of Wireless USB-specific device level capabilities.
USB 2.0 EXTENSION	02H	USB 2.0 Extension Descriptor.
SUPERSPEED_USB	03H	Defines the set of SuperSpeed USB specific device level capabilities.
Reserved	00H, 04-FFH	Reserved for future use.

The following section defines the USB_30 device capability and the USB 2.0 Extension Descriptor a SuperSpeed device shall return.

9.6.2.1 USB 2.0 Extension

A SuperSpeed device shall include the USB 2.0 Extension descriptor and shall support LPM when operating in USB 2.0 High-speed mode.

Table 9-13. USB_30 Device Capabilities Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of descriptor
1	*bDescriptorType*	1	Constant	DEVICE CAPABILITY Descriptor type
2	*bDevCapabilityType*	1	Constant	Capability type: USB 2.0 EXTENSION
3	*bmAttributes*	1	Bitmap	Bitmap encoding of supported device level features. A value of one in a bit location indicates a feature is supported; a value of zero indicates it is not supported. Encodings are: Bit — Encoding 0 — Reserved. Shall be set to zero. 1 — LPM. A value of one in this bit location indicates that this device supports the Link Power Management protocol. SuperSpeed devices shall set this bit to one. 7:2 — Reserved. Shall be set to zero.

9.6.2.2 SuperSpeed USB Device Capability

This section defines the required device-level capabilities descriptor which shall be implemented by all SuperSpeed devices. This capability descriptor cannot be directly accessed with a GetDescriptor() or SetDescriptor() request.

Table 9-14. SuperSpeed Device Capabilities Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of descriptor
1	*bDescriptorType*	1	Constant	DEVICE CAPABILITY Descriptor type
2	*bDevCapabilityType*	1	Constant	Capability type: SUPERSPEED_USB
3	*bmAttributes*	1	Bitmap	Bitmap encoding of supported device level features. A value of one in a bit location indicates a feature is supported; a value of zero indicates it is not supported. Encodings are: Bit — Encoding 0 — Reserved. Shall be set to zero. 1 — LTM Capable. A value of one in this bit location indicates that this device has is capable of generating Latency Tolerance Messages. 7:2 — Reserved. Shall be set to zero.
4	*wSpeedsSupported*	2	Bitmap	Bitmap encoding of the speed supported by this device when operating in SuperSpeed mode. Bit — Encoding 0 — If this bit is set, then the device supports operation at low speed USB. 1 — If this bit is set, then the device supports operation at full speed USB. 2 — If this bit is set, then the device supports operation at high speed USB. 3 — If this bit is set, then the device supports operation at 5 Gbs. 15:4 — Reserved. Shall be set to zero.
6	*bFunctionalitySupport*	1	Number	The lowest speed at which all the functionality supported by the device is available to the user. For example if the device supports all its functionality when connected at full speed and above then it sets this value to 1. Refer to the **wSpeedsSupported** field for valid values that can be placed in this field.

Offset	Field	Size	Value	Description
7	*bU1DevExitLat*	1	Number	U1 Device Exit Latency. Worst case latency to transition from U1 to U0, assuming the latency is limited only by the device and not the device's link partner. This field applies only to the exit latency associated with an individual port, and does not apply to the total latency through a hub (e.g., from downstream port to upstream port). The following are permissible values: Value — Meaning 00H — Zero. 01H — Less than 1 µs 02H — Less than 2 µs 03H — Less than 3 µs 04H — Less than 4 µs … — … 0AH — Less than 10 µs 0BH – FFH — Reserved For hubs this is the value for both its upstream and downstream ports.
8	*wU2DevExitLat*	2	Number	U2 Device Exit Latency. Worst case latency to transition from U2 to U0, assuming the latency is limited only by the device and not the device's link partner. Applies to all ports on a device. The following are permissible values: Value — Meaning 0000H — Zero 0001H — Less than 1 µs 0002H — Less than 2 µs 0003H — Less than 3 µs 0004H — Less than 4 µs … — … 07FFH — Less than 2047 µs 0800H – FFFFH — Reserved For hubs this is the value for both its upstream and downstream ports.
10	*wReserved*	2	Number	This field is reserved and shall be set to zero.
12	*guidID*	16	GUID	This is a unique 128-bit number that shall be returned by a SuperSpeed device in both SuperSpeed and High-speed modes. The value though unique must be the same number reported when operating in either mode. This field is required for hubs and is optional and may be set to zero for all other devices.

9.6.3 Configuration

The configuration descriptor describes information about a specific device configuration. The descriptor contains a bConfigurationValue field with a value that, when used as a parameter to the SetConfiguration() request, causes the device to assume the described configuration.

The descriptor describes the number of interfaces provided by the configuration. Each interface may operate independently. For example, a Video Class device might be configured with two interfaces, each providing 64 MB/s bi-directional channels that have separate data sources or sinks on the host. Another configuration might present the Video Class device as a single interface, bonding the two channels into one 128 MB/s bi-directional channel.

When the host requests the configuration descriptor, all related interface, endpoint and endpoint companion descriptors are returned (refer to Section 9.4.3).

A USB device has one or more configuration descriptors. Each configuration has one or more interfaces and each interface has zero or more endpoints. An endpoint is not shared among interfaces within a single configuration unless the endpoint is used by alternate settings of the same interface. Endpoints may be shared among interfaces that are part of different configurations without this restriction.

Once configured, devices may support limited adjustments to the configuration. If a particular interface has alternate settings, an alternate may be selected after configuration. Table 9-15 shows the standard configuration descriptor.

Table 9-15. Standard Configuration Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of this descriptor in bytes
1	*bDescriptorType*	1	Constant	CONFIGURATION Descriptor Type
2	*wTotalLength*	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	*bNumInterfaces*	1	Number	Number of interfaces supported by this configuration
5	*bConfigurationValue*	1	Number	Value to use as an argument to the SetConfiguration() request to select this configuration
6	*iConfiguration*	1	Index	Index of string descriptor describing this configuration

Offset	Field	Size	Value	Description
7	*bmAttributes*	1	Bitmap	Configuration characteristics D7: Reserved (set to one) D6: Self-powered D5: Remote Wakeup D4...0: Reserved (reset to zero) D7 is reserved and shall be set to one for historical reasons. A device configuration that uses power from the bus and a local source reports a non-zero value in *bMaxPower* to indicate the amount of bus power required and sets D6. The actual power source at runtime may be determined using the GetStatus(DEVICE) request (see Section 9.4.5). If a device configuration supports remote wakeup, D5 is set to one.
8	*bMaxPower*	1	mA	Maximum power consumption of the USB device from the bus in this specific configuration when the device is fully operational. Expressed in 2-mA units when the device is operating in High-Speed mode and in 8ma units when operating in SuperSpeed mode. (i.e., 50 = 100 mA in High-Speed mode and 50 = 400 mA in SuperSpeed mode). Note: 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 that it is no longer self-powered. A device may not increase its power draw from the bus, when it loses its external power source, beyond the amount reported by its configuration. 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, it fails operations it can no longer support. The USB System Software may determine the cause of the failure by checking the status and noting the loss of the device's power source.

9.6.4 Interface Association

The Interface Association Descriptor is used to describe that two or more interfaces are associated to the same function. An ‘association’ includes two or more interfaces and all of their alternate setting interfaces. A device must use an Interface Association descriptor for each device function that requires more than one interface. An Interface Association descriptor is always returned as part of the configuration information returned by a GetDescriptor(Configuration) request. An interface association descriptor cannot be directly accessed with a GetDescriptor() or SetDescriptor() request. An interface association descriptor must be located before the set of interface descriptors (including all alternate settings) for the interfaces it associates. All of the interface numbers in the set of associated interfaces must be contiguous. Table 9-15 shows the standard interface association descriptor. The interface association descriptor includes function class, subclass and protocol fields. The values in these fields can be the same as the interface class, subclass and protocol values from any one of the associated interfaces. The preferred implementation, for existing device classes, is to use the interface class, subclass and protocol field values from the first interface in the list of associated interfaces.

Table 9-16. Standard Interface Association Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of this descriptor in bytes.
1	*bDescriptorType*	1	Constant	INTERFACE ASSOCIATION Descriptor.
2	*bFirstInterface*	1	Number	Interface number of the first interface that is associated with this function.
3	*bInterfaceCount*	1	Number	Number of contiguous interfaces that are associated with this function.
4	*bFunctionClass*	1	Class	Class code (assigned by USB-IF). A value of zero is not allowed in this descriptor. If this field is FFH, the function class is vendor-specific. All other values are reserved for assignment by the USB-IF.
5	*bFunctionSubClass*	1	SubClass	Subclass code (assigned by USB-IF). If the *bFunctionClass* field is not set to FFH all values are reserved for assignment by the USB-IF.
6	*bFunctionProtocol*	1	Protocol	Protocol code (assigned by USB-IF). These codes are qualified by the values of the *bFunctionClass* and *bFunctionSubClass* fields.
7	*iFunction*	1	Index	Index of string descriptor describing this function.

NOTE

Since this particular feature was not included in earlier versions of the USB specification, there is issue with how existing USB OS implementations will support devices that use this descriptor. It is strongly recommended that device implementations utilizing the interface association descriptor *use the Multi-interface Function Class codes in the device descriptor. This allows simple and easy identification of these devices and allows on some operating systems, installation of an upgrade driver that can parse and enumerate configurations that include the Interface Association Descriptor. The Multi-interface Function Class code is documented on the http://www.usb.org/developers/docs website.*

9.6.5 Interface

The interface descriptor describes a specific interface within a configuration. A configuration provides one or more interfaces, each with zero or more endpoint descriptors. When a configuration supports more than one interface, the endpoint descriptors for a particular interface follow the interface descriptor in the data returned by the GetConfiguration() request. As mentioned earlier in this chapter, SuperSpeed devices shall return Endpoint Companion descriptors for each of the endpoints in that interface to return additional information about its endpoint capabilities. The Endpoint Companion descriptor shall immediately follow the endpoint descriptor it is associated with in the configuration information. An interface descriptor is always returned as part of a configuration descriptor. Interface descriptors cannot be directly accessed with a GetDescriptor() or SetDescriptor() request.

An interface may include alternate settings that allow the endpoints and/or their characteristics to be varied after the device has been configured. The default setting for an interface is always alternate setting zero. The SetInterface() request is used to select an alternate setting or to return to the default setting. The GetInterface() request returns the selected alternate setting.

Alternate settings allow a portion of the device configuration to be varied while other interfaces remain in operation. If a configuration has alternate settings for one or more of its interfaces, a separate interface descriptor and its associated endpoint and endpoint companion (when reporting its SuperSpeed configuration) descriptors are included for each setting.

If a device configuration supported a single interface with two alternate settings, the configuration descriptor would be followed by an interface descriptor with the *bInterfaceNumber* and *bAlternateSetting* fields set to zero and then the endpoint and endpoint companion (when reporting its SuperSpeed configuration) descriptors for that setting, followed by another interface descriptor and its associated endpoint and endpoint companion descriptors. The second interface descriptor's *bInterfaceNumber* field would also be set to zero, but the *bAlternateSetting* field of the second interface descriptor would be set to one.

If an interface uses only the Default Control Pipe, no endpoint descriptors follow the interface descriptor. In this case, the *bNumEndpoints* field shall be set to zero.

An interface descriptor never includes the Default Control Pipe in the number of endpoints. Table 9-17 shows the standard interface descriptor.

Table 9-17. Standard Interface Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of this descriptor in bytes
1	*bDescriptorType*	1	Constant	INTERFACE Descriptor Type
2	*bInterfaceNumber*	1	Number	Number of this interface. Zero-based value identifying the index in the array of concurrent interfaces supported by this configuration.
3	*bAlternateSetting*	1	Number	Value used to select this alternate setting for the interface identified in the prior field
4	*bNumEndpoints*	1	Number	Number of endpoints used by this interface (excluding the Default Control Pipe). If this value is zero, this interface only uses the Default Control Pipe.
5	*bInterfaceClass*	1	Class	Class code (assigned by the USB-IF). A value of zero is reserved for future standardization. If this field is set to FFH, the interface class is vendor-specific. All other values are reserved for assignment by the USB-IF.
6	*bInterfaceSubClass*	1	SubClass	Subclass code (assigned by the USB-IF). These codes are qualified by the value of the *bInterfaceClass* field. If the *bInterfaceClass* field is reset to zero, this field shall also be reset to zero. If the *bInterfaceClass* field is not set to FFH, all values are reserved for assignment by the USB-IF.
7	*bInterfaceProtocol*	1	Protocol	Protocol code (assigned by the USB). These codes are qualified by the value of the *bInterfaceClass* and the *bInterfaceSubClass* fields. If an interface supports class-specific requests, this code identifies the protocols that the device uses as defined by the specification of the device class. If this field is reset to zero, the device does not use a class-specific protocol on this interface. If this field is set to FFH, the device uses a vendor-specific protocol for this interface.
8	*iInterface*	1	Index	Index of string descriptor describing this interface

9.6.6 Endpoint

Each endpoint used for an interface has its own descriptor. This descriptor contains the information required by the host to determine the bandwidth requirements of each endpoint. An endpoint descriptor is always returned as part of the configuration information returned by a GetDescriptor(Configuration) request. An endpoint descriptor cannot be directly accessed with a GetDescriptor() or SetDescriptor() request. There is never an endpoint descriptor for endpoint zero. Table 9-18 shows the standard endpoint descriptor.

Table 9-18. Standard Endpoint Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of this descriptor in bytes
1	*bDescriptorType*	1	Constant	ENDPOINT Descriptor Type
2	*bEndpointAddress*	1	Endpoint	The address of the endpoint on the USB device described by this descriptor. The address is encoded as follows: Bit 3...0: The endpoint number Bit 6...4: Reserved, reset to zero Bit 7: Direction, ignored for control endpoints 0 = OUT endpoint 1 = IN endpoint

Offset	Field	Size	Value	Description
3	*bmAttributes*	1	Bitmap	This field describes the endpoint's attributes when it is configured using the *bConfigurationValue*. Bits 1..0: Transfer Type 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt If an interrupt endpoint, bits 5..2 are defined as follows: Bits 3..2: Reserved Bits 5..4: Usage Type 00 = Notification 01 = Periodic 10 = Reserved 11 = Reserved If isochronous, they are defined as follows: Bits 3..2: Synchronization Type 00 = No Synchronization 01 = Asynchronous 10 = Adaptive 11 = Synchronous Bits 5..4: Usage Type 00 = Data endpoint 01 = Feedback endpoint 10 = Implicit feedback Data endpoint 11 = Reserved If not an isochronous or interrupt endpoint, bits 5..2 are reserved and shall be set to zero. All other bits are reserved and shall be reset to zero. Reserved bits shall be ignored by the host.
4	*wMaxPacketSize*	2	Number	Maximum packet size this endpoint is capable of sending or receiving when this configuration is selected. There are only two legal values for this field. For control endpoints this field shall be set to 512. For all other endpoint types this field shall be set to 1024.
6	*bInterval*	1	Number	Interval for polling endpoint for data transfers. Expressed in 125-µs units. For SuperSpeed isochronous and interrupt endpoints, this value shall be in the range from 1 to 16. The bInterval value is used as the exponent for a $2^{(bInterval-1)}$ value; e.g., a bInterval of 4 means a period of.8 ($2^{(4-1)} \rightarrow 2^3 \rightarrow 8$). For SuperSpeed bulk/control OUT endpoints, the bInterval is not used.

The *bmAttributes* field provides information about the endpoint's Transfer Type (bits 1..0) and Synchronization Type (bits 3..2). In addition, the Usage Type bit (bits 5..4) indicate whether this is an endpoint used for normal data transfers (bits 5..4 = 00B), whether it is used to convey explicit feedback information for one or more data endpoints (bits 5..4 = 01B) or whether it is a data endpoint that also serves as an implicit feedback endpoint for one or more data endpoints

(bits 5..4=10B). Bits 5..2 are only meaningful for isochronous endpoints and shall be reset to zero for all other transfer types.

If the endpoint is used as an explicit feedback endpoint (bits 5..4 = 01B), then the Transfer Type shall be set to isochronous (bits1..0 = 01B) and the Synchronization Type shall be set to No Synchronization (bits 3..2 = 00B).

A feedback endpoint (explicit or implicit) needs to be associated with one (or more) isochronous data endpoints to which it provides feedback service. The association is based on endpoint number matching. A feedback endpoint always has the opposite direction from the data endpoint(s) it services. If multiple data endpoints are to be serviced by the same feedback endpoint, the data endpoints shall have ascending ordered–but not necessarily consecutive–endpoint numbers. The first data endpoint and the feedback endpoint shall have the same endpoint number (and opposite direction). This ensures that a data endpoint can uniquely identify its feedback endpoint by searching for the first feedback endpoint that has an endpoint number equal or less than its own endpoint number.

Example: Consider the extreme case where there is a need for five groups of OUT asynchronous isochronous endpoints and at the same time four groups of IN adaptive isochronous endpoints. Each group needs a separate feedback endpoint and the groups are composed as shown in Table 9-19.

Table 9-19. Example of Feedback Endpoint Numbers

OUT Group	Nr of OUT Endpoints	IN Group	Nr of IN Endpoints
1	1	6	1
2	2	7	2
3	2	8	3
4	3	9	4
5	3		

The endpoint numbers can be intertwined as illustrated in Figure 9-7.

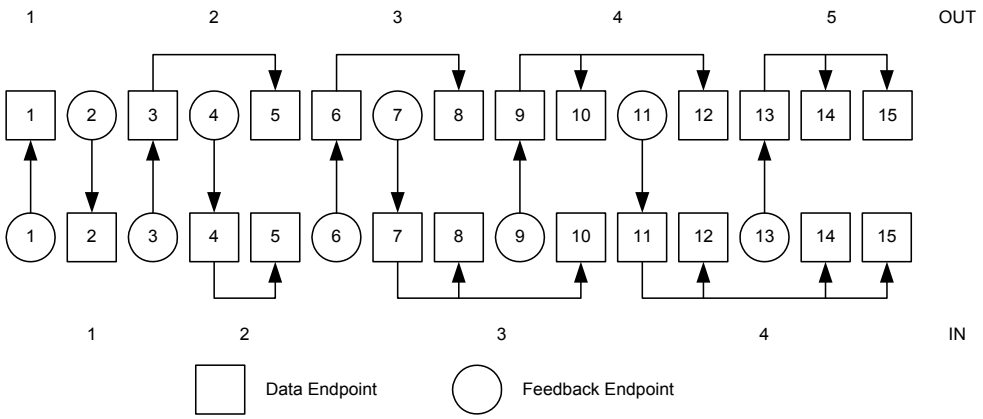


Figure 9-7. Example of Feedback Endpoint Relationships

For high-speed bulk and control OUT endpoints, the *bInterval* field is only used for compliance purposes; the host controller is not required to change its behavior based on the value in this field.

9.6.7 SuperSpeed Endpoint Companion

Each SuperSpeed endpoint described in an interface is followed by a SuperSpeed Endpoint Companion descriptor. This descriptor contains additional endpoint characteristics that are only defined for SuperSpeed endpoints. This descriptor is always returned as part of the configuration information returned by a GetDescriptor(Configuration) request and cannot be directly accessed with a GetDescriptor() or SetDescriptor() request. The Default Control Pipe does not have an Endpoint Companion descriptor. The Endpoint Companion descriptor shall immediately follow the endpoint descriptor it is associated with in the configuration information.

Table 9-20. SuperSpeed Endpoint Companion Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of this descriptor in bytes
1	*bDescriptorType*	1	Constant	SUPERSPEED_USB_ENDPOINT_COMPANION Descriptor Type
2	*bMaxBurst*	1	Number	The maximum number of packets the endpoint can send or receive as part of a burst. Valid values are from 0 to 15. A value of 0 indicates that the endpoint can only burst one packet at a time and a value of 15 indicates that the endpoint can burst up to sixteen packets at a time. For endpoints of type control this shall be set to 0.

Offset	Field	Size	Value	Description
3	*bmAttributes*	1	Bitmap	If a Bulk Endpoint Bits: Description 4:0 **MaxStreams**. The maximum number of streams this endpoint supports. Valid values are from 0 to 16, where a value of 0 indicates that the endpoint does not define streams. For the values 1 to 16, the number of streams supported equals to $2^{MaxStream}$. This field is only valid for Bulk endpoints. 7:5 **Reserved**. These bits are reserved and shall be set to zero. If a Control or Interrupt Endpoint Bits: Description 7:0 **Reserved**. These bits are reserved and shall be set to zero. If an isochronous endpoint: Bits: Description 1:0 **Mult**. A zero based value that indicates the maximum number of bursts within a service interval that this endpoint supports. This field is only valid for Isochronous endpoints. A value of zero indicates that the device supports one Burst of bMaxBurst packets per service interval. The max value that can be set in this field is 2. 7:2 **Reserved. These bits are reserved and shall be set to zero.** 7 **Reserved**. This bit is reserved and shall be set to zero.
4	*wBytesPerInterval*	2	Number	The total number of bytes this endpoint will transfer every service interval. This field is only valid for periodic endpoints. For isochronous endpoints, this value is used to reserve the bus time in the schedule, required for the frame data payloads per 125 µs. The pipe may, on an ongoing basis, actually use less bandwidth than that reserved. The device reports, if necessary, the actual bandwidth used via its normal, non-USB defined mechanisms.

9.6.8 String

String descriptors are optional. As noted previously, if a device does not support string descriptors, all references to string descriptors within device, configuration, and interface descriptors shall be reset to zero.

String descriptors use UNICODE encodings as defined by *The Unicode Standard, Worldwide Character Encoding, Version 3.0*, The Unicode Consortium, Addison-Wesley Publishing Company, Reading, Massachusetts (URL: http://www.unicode.com). The strings in a USB device may support multiple languages. When requesting a string descriptor, the requester specifies the desired language using a sixteen-bit language ID (LANGID) defined by the USB-IF. The list of currently defined USB LANGIDs can be found at http://www.usb.org/developers/docs.html. String index zero for all languages returns a string descriptor that contains an array of two-byte LANGID codes supported by the device. Table 9-21 shows the LANGID code array. A USB device may omit all string descriptors. USB devices that omit all string descriptors shall not return an array of LANGID codes.

The array of LANGID codes is not NULL-terminated. The size of the array (in bytes) is computed by subtracting two from the value of the first byte of the descriptor.

Table 9-21. String Descriptor Zero, Specifying Languages Supported by the Device

Offset	Field	Size	Value	Description
0	*bLength*	1	N+2	Size of this descriptor in bytes
1	*bDescriptorType*	1	Constant	STRING Descriptor Type
2	*wLANGID[0]*	2	Number	LANGID code zero
...	...	...	...	...
N	*wLANGID[x]*	2	Number	LANGID code *x*

The UNICODE string descriptor (shown in Table 9-22) is not NULL-terminated. The string length is computed by subtracting two from the value of the first byte of the descriptor.

Table 9-22. UNICODE String Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of this descriptor in bytes
1	*bDescriptorType*	1	Constant	STRING Descriptor Type
2	*bString*	N	Number	UNICODE encoded string

9.7 Device Class Definitions

All devices shall support the requests and descriptor definitions described in this chapter. Most devices provide additional requests and, possibly, descriptors for device-specific extensions. In addition, devices may provide extended services that are common to a group of devices. In order to define a class of devices, the following information shall be provided to completely define the appearance and behavior of the device class.

9.7.1 Descriptors

If the class requires any specific definition of the standard descriptors, the class definition shall include those requirements as part of the class definition. In addition, if the class defines a standard extended set of descriptors, they shall also be fully defined in the class definition. Any extended descriptor definitions shall follow the approach used for standard descriptors; for example, all descriptors shall begin with a length field.

9.7.2 Interface(s)

When a class of devices is standardized, the interfaces used by the devices shall be included in the device class definition. Devices may further extend a class definition with proprietary features as long as they meet the base definition of the class.

9.7.3 Requests

All of the requests specific to the class shall be defined.

10 Hub, Host Downstream Port, and Device Upstream Port Specification

This chapter describes the architectural requirements for a USB 3.0 hub. The chapter also describes any differences between functional requirements for a host downstream port and a hub downstream port and any differences between a device upstream port and a hub upstream port. The chapter contains a description of two of the three principal sub-blocks: the super speed hub repeater/forwarder and the SuperSpeed hub controller. The USB 2.0 hub sub-block is described in the *Universal Serial Bus Specification, Revision 2.0*. This chapter also describes the hub's operation for error recovery, reset, and suspend/resume. The second half of the chapter defines hub request behavior and hub descriptors.

The hub specification chapter along with the *Universal Serial Bus Specification, Revision 2.0* supply sufficient additional information to permit an implementer to design a hub that conforms to the USB 3.0 specification.

10.1 Hub Feature Summary

Hubs provide the electrical interface between USB devices and the host. Hubs are directly responsible for supporting many of the attributes that make USB user friendly and hide its complexity from the user. Listed below are the major aspects of USB functionality that hubs shall support:

- Connectivity behavior
- Power management
- Device connect/disconnect detection
- Bus fault detection and recovery
- SuperSpeed and USB 2.0 (high speed, full speed, and low speed) device support

A USB 3.0 hub incorporates a USB 2.0 hub and a SuperSpeed hub consisting of two principal components: the SuperSpeed Hub Repeater/Forwarder and the SuperSpeed Hub Controller. The USB 2.0 hub is described in the USB 2.0 specification. All subsequent references in this specification are to components of the SuperSpeed hub unless otherwise noted. The Hub Repeater/Forwarder is responsible for connectivity setup and tear-down. It also supports exception handling, such as bus fault detection and recovery and connect/disconnect detect. The Hub Controller provides the mechanism for host-to-hub communication. Hub-specific status and control commands permit the host to configure a hub and to monitor and control its individual downstream facing ports.

Figure 10-2 shows a high level block diagram of a four port USB 3.0 hub and the locations of its upstream and downstream facing ports. A USB 3.0 hub is the logical combination of two hubs: a USB 2.0 hub and a Super Speed hub. Each hub operates on a separate data bus. Typically, the only signal shared between them is VBUS. If either the USB 2.0 hub or Super Speed hub controllers requires, a downstream port is powered. A USB 3.0 hub connects on both interfaces upstream whenever possible. All exposed downstream ports on a USB 3.0 hub shall support both SuperSpeed and USB 2.0 connections.

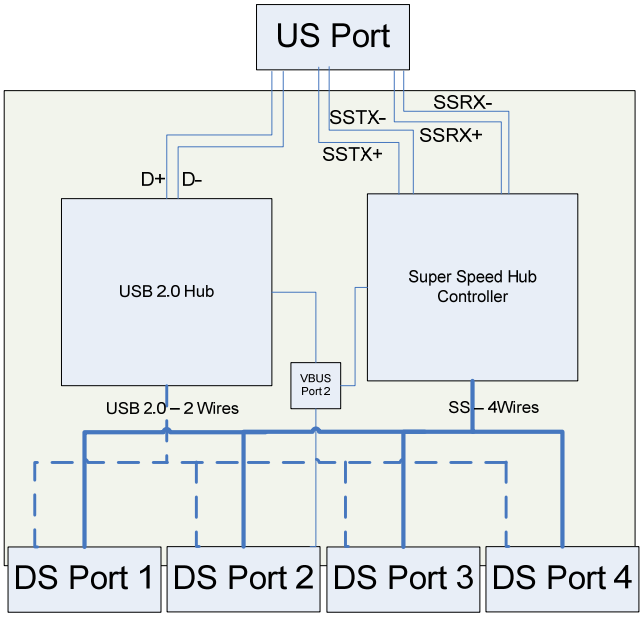


Figure 10-1. Hub Architecture

Figure 10-2 shows the Super Speed portion of a USB 3.0 hub consisting of a Hub Repeater/Forwarder section and a Hub Controller section.

The USB 2.0 portion of a USB 3.0 hub shall meet all requirements of the USB 2.0 specification unless specific exceptions are noted.

The Hub Repeater/Forwarder is responsible for managing connectivity between upstream and downstream facing ports which are operating Super Speed. The Hub Controller provides status and control and permits host access to the Super Speed Hub.

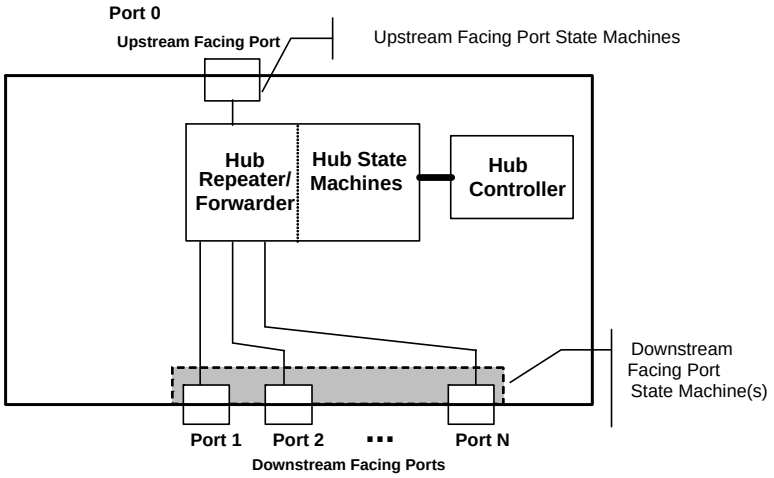


Figure 10-2. SuperSpeed Portion of the Hub Architecture

When the hub upstream facing port is attached to an electrical environment that is only operating at high-speed, SuperSpeed connectivity is not available to devices attached to downstream facing ports.

Unlike other USB 3.0 devices, a USB 3.0 hub is required to connect upstream on both the USB 3.0 and USB 2.0 buses. Hardware support for SuperSpeed connections may be enabled/disabled under the control of software/firmware for USB 3.0 downstream ports. If the hub upstream SuperSpeed connection is not supported by the root/hub port hardware to which the USB 3.0 hub is connected, the hub disables SuperSpeed support on all of its downstream ports. If a USB 3.0 hub upstream port is not connected, the hub does not provide power to the downstream ports unless it supports charging applications. Refer to Section 10.3.1.1 for a detailed discussion on when a hub is allowed to remove VBUS from a downstream facing port. The USB 3.0 specification allows self-powered and bus-powered hubs. The following sections present the typical flow for connection management in various types of systems for the simple topology shown in Figure 10-3 when the host system is first powered on.

Note: These connection examples outline cases where the system operates as expected. The handling of error cases are specified later in this chapter.

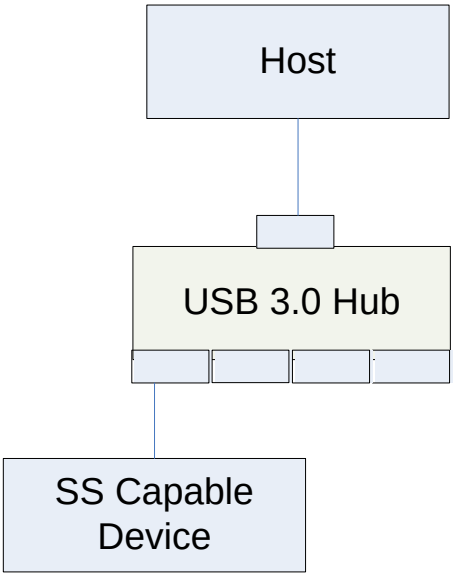


Figure 10-3. Simple USB 3.0 Topology

10.1.1 SuperSpeed Capable Host with SuperSpeed Capable Software

Host is powered off. Hub does not provide power to its downstream ports.[1]

Host is powered on with SuperSpeed support enabled on its downstream ports by default.

Hub detects VBUS and SuperSpeed support and powers its downstream ports with SuperSpeed enabled.

Hub connects both as a SuperSpeed and as a high-speed device.

Device detects VBUS and SuperSpeed support and connects as a SuperSpeed device.

Host system begins hub enumeration at high-speed and SuperSpeed.

Host system begins device enumeration at SuperSpeed.

10.1.2 USB 2.0 Host

Host is powered off. Hub does not provide power to its downstream ports. [2]

Host is powered on. There is no SuperSpeed hardware support.

Hub detects VBUS and connects as a high-speed device.

[1] Unless the hub supports charging applications. Refer to section 10.3.1.1for more details.

[2] Unless the hub supports charging applications. Refer to section 10.3.1.1for more details.

Host system begins hub enumeration at high-speed.

Hubs power downstream ports when directed by software with SuperSpeed support disabled.

Device connects at high-speed.

Host system begins device enumeration at high-speed.

10.1.3 Hub Connectivity

Hubs exhibit different connectivity behavior depending on whether they are propagating data packet header/data packet payload packet traffic, other packet traffic, resume signaling, or are in an Idle state.

10.1.3.1 Packet Signaling Connectivity

The Hub Repeater/Forwarder contains one port that shall always connect in the upstream direction (referred to as the upstream facing port) and one or more downstream facing ports. Upstream connectivity is defined as being towards the host, and downstream connectivity is defined as being towards a device. A Super Speed hub controller contains buffering for header and data packets. A Super Speed hub controller does not use the repeater only model used for high speed connectivity in a USB 2.0 hub. This change allows multiple downstream devices to send asynchronous messages simultaneously without data loss and for some traffic to be stored and delivered when it is directed to downstream ports when the link is not in U0. Figure 10-4 shows the high level packet signaling connectivity behavior for hubs in the upstream and downstream directions. Later sections describe the hub internal buffering and connectivity in more detail. A hub also has an Idle state, during which the hub makes no connectivity. When in the Idle state, all of the hub's ports (upstream plus downstream) are U1, U2 or in U0 receiving and transmitting logical idles waiting for the start of the next packet.

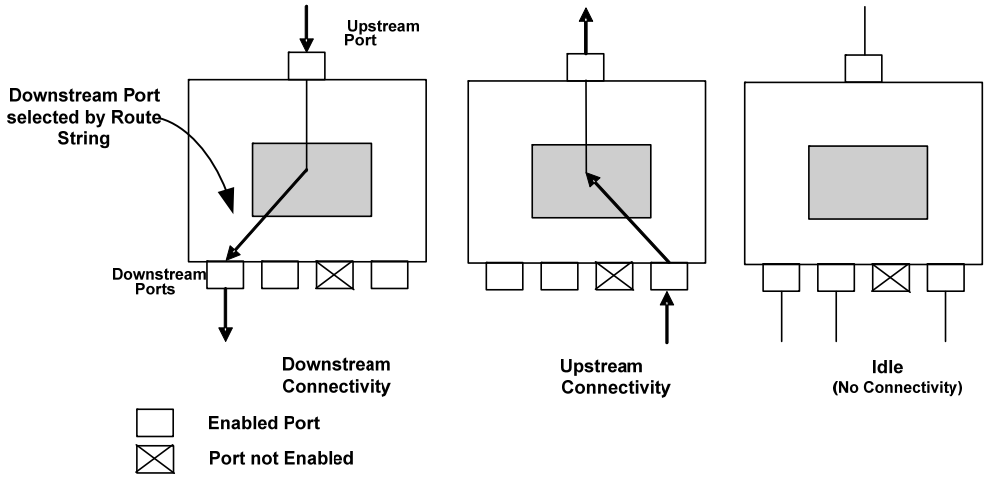


Figure 10-4. Hub Signaling Connectivity

If a downstream facing port is enabled (i.e., in a state where it can propagate signaling through the hub), and the hub detects the start of a packet on that port, the hub begins to store the packet header. Connectivity is established in an upstream direction to the upstream facing port of that hub

whenever a valid header has been received on a downstream port. The hub transmits valid headers received on downstream ports upstream, but not to any other downstream facing ports. This means that when a device or a hub transmits a packet upstream, only those hubs in line between the transmitting device and the host will see the packet.

All packets except Isochronous Timestamp Packets are unicast in the downstream direction; hubs operate using a direct connectivity model. This means that when the host or hub transmits a packet downstream, only those hubs in line between the host and intended recipient device will see the packet. When a hub detects the start of a packet on its upstream facing port the hub begins to store the packet header. It uses the Route String in the packet header and a Route String Index assigned when enumerated to establish connectivity only to the port indicated. If the indicated port is not enabled, it does not propagate packet signaling downstream. The hub shall silently drops a header that is routed to a downstream port that is not enabled. The hub shall perform link level acknowledgement of the header normally in this case. Hub behavior is undefined if a header is routed to a downstream port that does not exist.

10.1.3.2 Routing Information

Packets received on the hub upstream port are routed based on information contained in a 20-bit field (Route String) in the packet header. It is used in conjunction with a hub depth value by the hub to identify the target port for a downstream directed packet. The hub depth value is assigned by software using the Set Hub Depth request. The hub ignores the route string and assumes all packets are routed directly to the hub until the hub enters the configured state. The hub's upstream port shall be represented by port number zero while the downstream ports shall begin with port number one and count up sequentially. Whenever a hub controller responds to a packet routed to the hub with a packet containing a route string or originates a packet (except for a packet the hub is deferring) the hub shall set the route string to zero.

Figure 10-5 illustrates the use of route strings in an example topology with 5 levels of four port USB 3.0 hubs. The hub depth value for each level of hub is illustrated in the figure. Each hub and each device in the topology contains the route string that would be used to route a packet to that device/hub. For each hub depth, the octet in the route string that determines the routing target at that hub depth is a different color than the rest of the route string. The host root port is not included in the 20 bit route string.

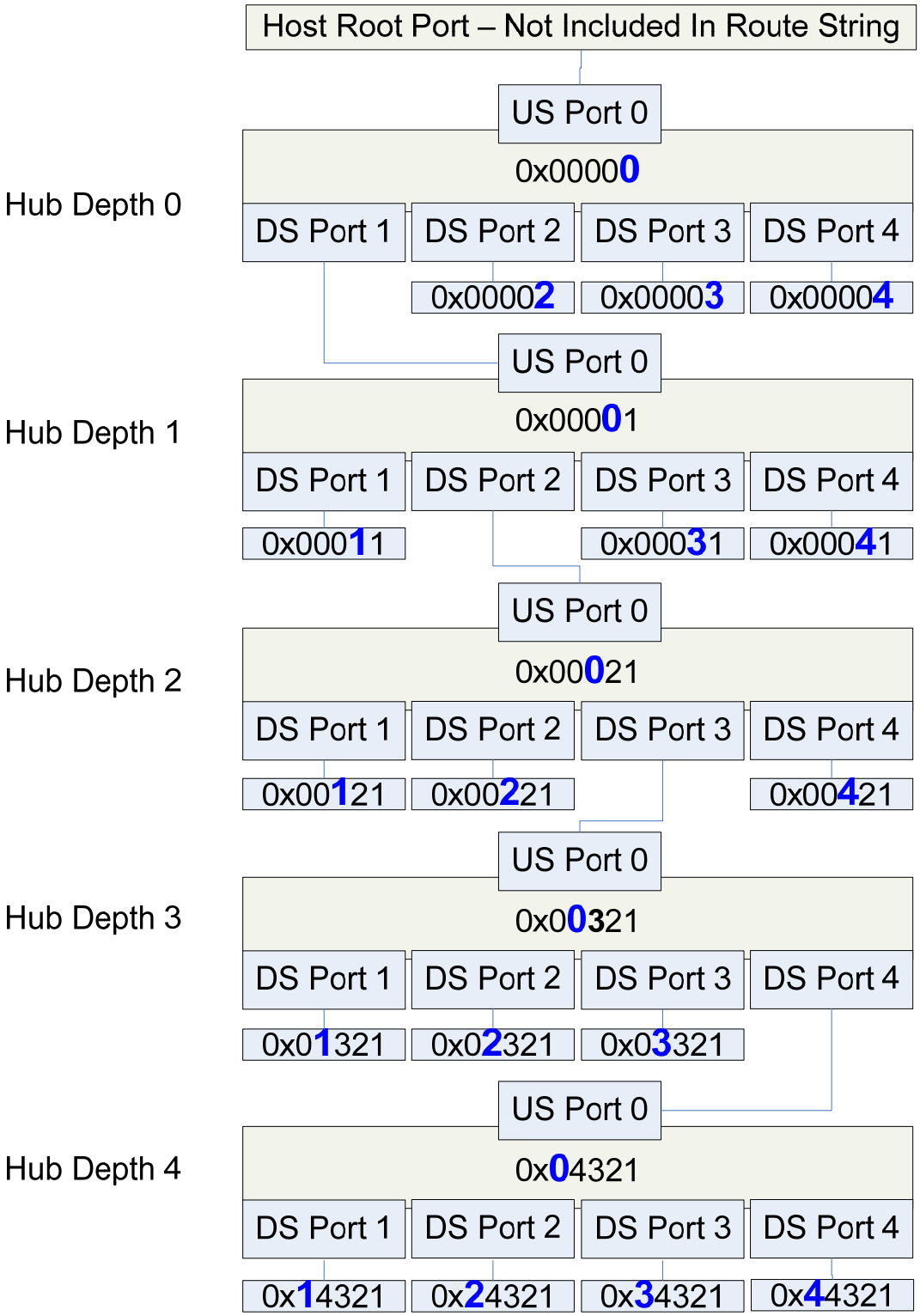


Figure 10-5. Route String Example

10.1.4 Resume Connectivity

Hubs exhibit different connectivity behaviors for upstream- and downstream-directed resume signaling. A hub that is suspended (e.g. ,its upstream facing port is in U3/D3) does not reflect resume signaling from its upstream facing port to any of its enabled downstream facing ports unless a downstream facing port is suspended and has received resume signaling since it was suspended. Figure 10-6 illustrates hub upstream and downstream resume connectivity.

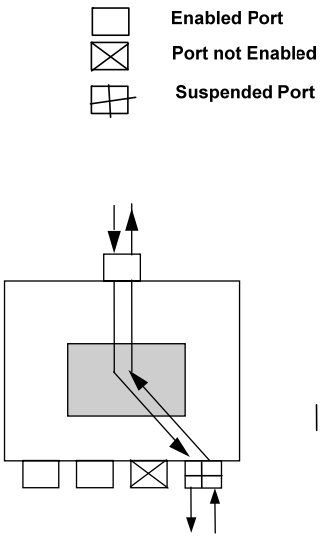


Figure 10-6. Resume Connectivity

If a hub is suspended and detects resume signaling from a suspended downstream facing port, the hub reflects that signaling upstream. It does not reflect that signaling to any downstream facing ports. Note that software shall not initiate a transition to U3 on the upstream port of a hub unless it has already initiated transitions to U3 on all enabled downstream ports. A detailed discussion of resume connectivity appears in Section 10.8.

10.1.5 Hub Fault Recovery Mechanisms

Hubs are the essential USB component for establishing connectivity between the host and other devices. It is vital that any connectivity faults be detected and prevented from occurring.

Hubs must also be able to detect and recover from lost or corrupted packets that are addressed to the Hub Controller. Because the Hub Controller is, in fact, another USB device, it shall adhere to the same timeout rules as other USB devices, as described in the Protocol Chapter.

10.1.6 Hub Header Buffer Architecture

Figure 10-7 shows the logical representation of a typical header buffer implementation for a SuperSpeed hub. Logically, a SuperSpeed hub has separate header buffers associated with each port for both upstream and downstream traffic. When a hub receives a header on its upstream port, it routes the header to the appropriate downstream header buffer for transmission (unless the header is for the hub). When the hub receives a header on a downstream port, it routes the header to the upstream port header buffer for transmission. Headers are kept in the hub header buffers after transmission until link level acknowledgement (LGOOD_n) for the header is received. This allows the hub to retry the headers if necessary to ensure that headers are received correctly at the link level. The header buffers also allow a hub to store the headers until they can be forwarded when the header is directed to a downstream link that is a low power link state. Hubs store the header and deliver it once the link becomes active.

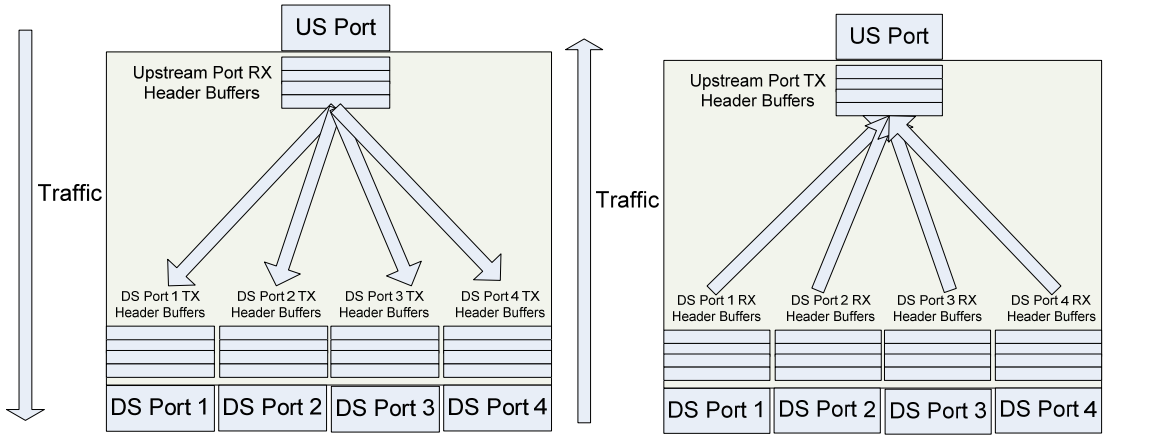


Figure 10-7. Typical Hub Header Buffer Architecture

10.1.6.1 Hub Data Buffer Architecture

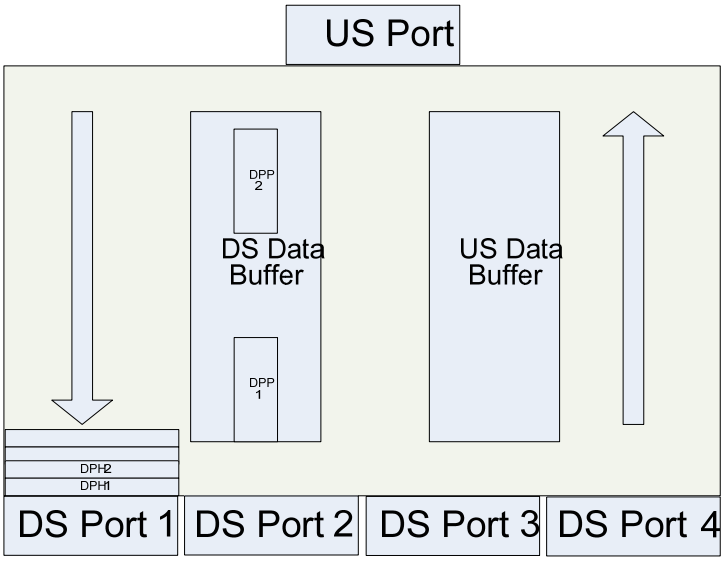


Figure 10-8. Example Hub Data Buffer Traffic
Header Buffer Only Shown for DS Port 1.

Figure 10-8 shows the logical representation of the data buffer architecture in a typical SuperSpeed hub. SuperSpeed hubs provide independent buffering for data packets in both the upstream and downstream directions. The USB 3.0 Architecture allows concurrent transactions to occur in both the upstream and downstream directions. In the figure two packets are in progress in the downstream direction. The hub can store more than one data packet payload at the same time. In rare occurrences where packet payloads are discarded if buffering is unavailable, end-to-end protocol will retry to recover.

10.2 Hub Power Management

10.2.1 Link States

The hub is required to support U0, U1, U2, and U3 on all ports (upstream and downstream).

10.2.2 U1/U2 Timers

The hub is required to have inactivity timers for both U1 and U2 on each downstream port. The timeout values are programmable and may be set by the host software. The timeout value zero means the timeout is disabled. The default value for the U1/U2 timeouts is zero. The downstream port state machines presented in this chapter describe the specific operational rules when U1 and/or U2 timeouts are enabled.

- Hub downstream ports shall accept U1 or U2 entry initiated by a link partner except when the corresponding U1/U2 timeout is set to zero or there is pending traffic directed to the downstream port.

Note: If a hub has received a valid packet on its upstream port that is routed to a downstream port it shall reject U1 or U2 link entry attempts on the downstream port until the packet has been successfully transmitted. A hub may also reject U1 or U2 link entry attempts on downstream ports if the hub is receiving a packet but has not determined the packet destination. A hub implementation shall ensure no race condition exists where a header that has not been deferred is queued for transmission on a downstream port with a link that is in U1, U2, or U3 or is in the process of entering U1, U2, or U3.

- Hub downstream ports shall reject all U1 and U2 entry requests if the corresponding timeout is set to zero.
- The hub inactivity timers for U1 and U2 shall not be reset by an isochronous timestamp packet.

10.2.3 Downstream/Upstream Port Link State Transitions

The hub shall evaluate link power state changes made to its downstream ports such that it propagates the highest U-state (U0 > U1 >U2 > U3 > Disabled) of all downstream ports to its upstream port when there is no pending upstream traffic. If an upstream port link state transition would result in an upstream port link state that has been disabled by software, the hub shall transition the upstream port link to the next highest U-state that is enabled.

The downstream port state machines presented in this chapter provide the specific timing requirements for changing the upstream port link state in response to downstream port link state changes.

The hub also shall initiate a link state transition whenever it receives a packet that is routed to downstream port that is not in U0. The hub upstream port state machines provided in this chapter provide the specific timing requirements for these transitions.

If enabled, port status change interrupts, e.g., due to a connect event on a downstream port, will cause the upstream link to initiate a transition to U0.

10.3 Downstream Facing Ports

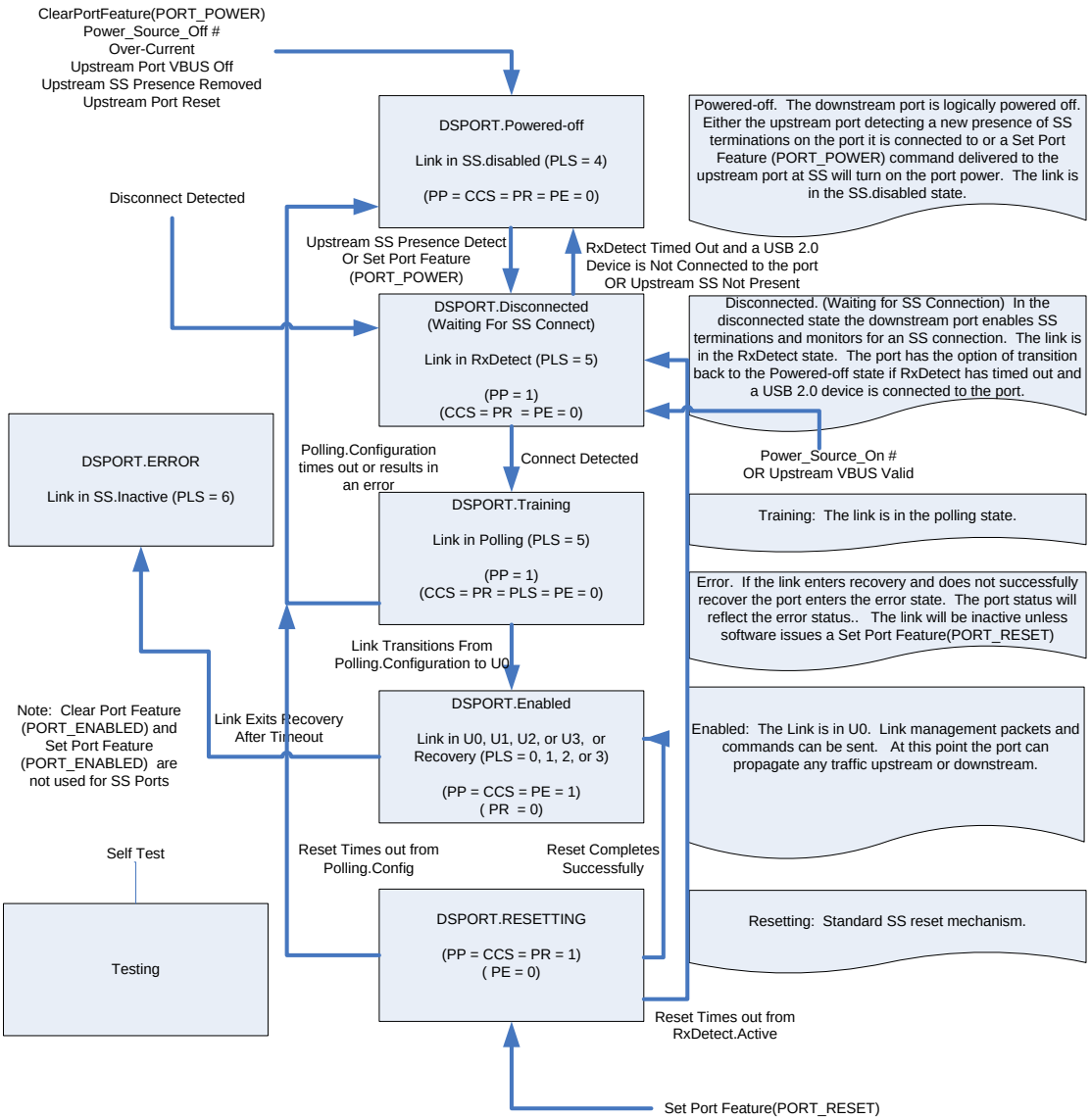
The following sections provide a functional description of a state machine that exhibits the correct behavior for a downstream facing port.

Figure 10-9 is an illustration of the downstream facing port state machine. Each of the states is described in Section 10.4.2. In the diagram below, some of the entry conditions into states are shown without origin. These conditions have multiple origin states and the individual transitions lines are not shown so that the diagram can be simplified. The description of the entered state indicates from which states the transition is applicable.

TBD – Specifically say hub upstream port everywhere that upstream SS detect, etc is referenced.

NOTE

For the root hub, the signals from the upstream facing port state machines are implementation dependent.



Port Status Field:

Notation	Field Name
PP	PORT_POWER
CCS	PORT_CONNECTION
PR	PORT_RESET
PLS	PORT_LINK_STATE
PE	PORT_ENABLE

Note: GetPortStatus Requests will return all zeros until a Set Configuration Request for a non-zero config and a subsequent Set Port Feature(PORT_POWER) request have occurred for the port.

Figure 10-9. Downstream Facing Hub Port State Machine

Table 10-1. Downstream Facing Port Signal/Event Definitions

Signal/Event Name	Event/Signal Source	Description
Power_source_off	Implementation- dependent	Power to the port not available due to over-current or termination of source power (e.g., external power removed)
Over-current	Hub Controller	Over-current condition exists on the hub or the port
Disconnect_Detect	Internal	Disconnect seen at port
Port_Error	Internal	Error condition detected (see Section 10.3.1.4)
SetTest	Hub Controller	TBD

10.3.1 Downstream Facing Port State Descriptions

10.3.1.1 DSPORT.Powered-off

The DSPORT.Powered-off state is a logical powered off state. The hub may still be required or choose to provide VBUS for a downstream port in the DSPORT.Powered-off state. Detailed requirements for presence of VBUS are covered later in this section.

A port shall transition into this state if any of the following situations occur:

- From any state when the hub receives a ClearPortFeature(port_power) request for this port. In this case power is only removed from the port if it would not impact the LS, FS, or HS operation on any of the downstream ports on the hub and would not impact SS operation on any ports other than the target port.
- From any state when power is lost to the port or an over-current condition exists.
- From any state when VBUS is removed from the upstream hub port.
- From any state if the hub's upstream port link transitions to the SS.disabled state.
- From any state if the upstream hub port's link has attempted 8 consecutive RxDetect events without detecting terminations.
- From the DSPORT.Disconnected state if RxDetect times out and a USB 2.0 device is not connected to the port.
- From the DSPORT.Training state if the port's link times out from the polling.config state.

Note that the transitions from the DSPORT.Disconnected and DSPORT.Training link states are not direct transitions to the SS.disabled link state. Refer to the link chapter for more details on the steps involved in these transitions.

A port will enter this state due to an over-current condition on another port if that over-current condition may have caused the power supplied to this port to drop below specified limits for port.

If a hub was configured while the external power supply was present, and then if external power is lost, the hub shall place all ports in the Powered-off state if power remains to run the hub controller.

In this state, the port's link is in the SS.disabled state.

Table 10-2 shows the allowed state of VBUS for downstream hub ports for possible states of the upstream hub port and logical port power for a downstream port. The table covers the case where the hub has adequate power to provide power for the downstream ports (external power source is present). For a hub that does not implement per port power control all downstream ports that will be affected by removing VBUS shall be in a state where power may be off before the hub removes

VBUS. Note that a hub may provide power to all its downstream ports all of the time to support applications such as battery charging from a USB port. If a hub provides power on a downstream port when VBUS is not present on the hub upstream port it shall stop providing VBUS for tDownstreamVBUS Drop when VBUS becomes valid on the hub upstream port.

Table 10-2. Downstream Port VBUS Requirements

Hub Upstream Port Connection Status	Downstream Port SuperSpeed Port Power Off (PORT_POWER = 0) USB 2.0 Port Power On (PORT_POWER = 1)	Downstream Port USB 2.0 Port Power Off (PORT_POWER = 0) Downstream Port SuperSpeed Port Power On (PORT_POWER = 1)	Downstream Port USB 2.0 and SuperSpeed Port Power Off (PORT_POWER = 0)
SuperSpeed	On[3]	On	May be off
USB 2.0	On	May be off	May be off
SuperSpeed and USB 2.0	On	On	May be off
No VBUS	May be off	May be off	May be off

10.3.1.2 DSPORT.Disconnected (Waiting for SS Connect)

This is the default state when local power is valid (self-powered) or VBUS becomes valid (bus-powered. A port transitions to this state in any of the following situations:

- From the DSPORT.Powered-off state when the hub receives a SetPortFeature(PORT_POWER) request and the link for the hub upstream port is in any state other than SS.disabled or SS.RxDetect.
- From any state except the DSPORT.Powered-off state when the port detects a disconnect.
- From the DSPORT.Powered-off state when the hub upstream port's link transitions from RxDetect to the polling state.

In this state the port's link shall be in the RxDetect state.

Note: The port's link shall still perform connection detection normally from the RxDetect if the upstream hub port's link is in U3.

10.3.1.3 DSPORT.Training

A port transitions to this state from the DSPORT.Disconnected state when SuperSpeed terminations are detected.

In this state the port's link shall be in the Polling state.

[3] If the hub upstream port is unable to connect on the USB 2.0 bus the downstream port VBUS may be off in this state.

10.3.1.4 DSPORT.ERROR

A port transitions to this state only when an SS capable device is present and a serious error condition occurred while attempting to operate the link. A port transitions to this state in any of the following situations:

- From the DSPORT.Enabled state if the link enters recovery and times out without recovering.

A downstream port exits the error state only if software issues a SetPortFeature(PORT_RESET) request..

In this state the port's link shall be in the SS.Inactive state.

10.3.1.5 DSPORT.Enabled

A port transitions to this state in any of the following situations:

- From the Training state when the port's link successfully enters U0.

A port in the Enabled state will propagate transaction packets in either the upstream or the downstream direction.

When the hub downstream port first transitions to the Enabled state after a power on reset, the value for the U1 and U2 timers shall be reset to zero.

The link shall be in U0 when the enabled state is entered.

If the hub upstream port's link is in U3 when the downstream port enters DSPORT.Enabled and the hub is not enabled for remote wakeup, the downstream port shall initiate a transition to U3 on its link within tDSPortEnabledToU3.

Section 10.4 provides a state machine that shows a functionally correct implementation for a downstream port managing different link states within the DSPORT.ENABLED state.

10.3.1.6 DSPORT.Resetting

A downstream port transition to the DSPORT.Resetting state when it receives a SetPortFeature(PORT_RESET) request unless the port is in the DSPORT.Powered-off or DSPORT.Disconnected state. If the downstream port is in the DSPORT.Powered-off or DSPORT.Disconnected state and received a SetPortFeature(PORT_RESET) request it does not change states.

If the port state when the SetPortFeature(PORT_RESET) request is received is DSPORT.Error the port shall send a warm reset on the downstream port link within tDSPortResetToLFPS. If the port state when the SetPortFeature(PORT_RESET) request is received is DSPORT.Enabled the port shall initiate a hot reset on the link within tDSPortResetToHotReset.

When the downstream port link enters RxDetect.active during a warm reset the hub shall start a timer to count the time is in RxDetect.active. If this timer exceeds tTimeForResetError while the link remains in RxDetect.active the port shall transition to the DSPORT.disconnected state.

10.3.2 Disconnect Detect Mechanism

Disconnect detection mechanisms are covered in the link chapter in Section 7.4.6.

10.3.3 Labeling

USB system software uses port numbers to reference an individual port with a ClearPortFeature or SetPortFeature request. If a vendor provides a labeling to identify individual downstream facing ports, then each port connector shall be labeled with its respective port number. The port numbers assigned to a specific port by the hub shall be consistent between the USB 2.0 hub and SuperSpeed hub controller.

10.4 Downstream Facing Ports Power Management

The following sections provide a functional description of a state machine that exhibits the correct link power management behavior for a downstream facing port.

Figure 10-10 is an illustration of the downstream facing port power management state machine. Each of the states is described in Section 10.4.2. In the diagram below, some of the entry conditions into states are shown without origin. These conditions have multiple origin states and the individual transitions lines are not shown so that the diagram can be simplified. The description of the entered state indicates from which states the transition is applicable.

10.4.1 Downstream Facing Port PM Timer

Each downstream port maintains logical inactivity timers for keeping track of when U1 and U2 timeouts are exceeded. The timers shall be reset every time a packet of any type except an isochronous timestamp packet is sent or received by the port's link. The timer shall be accurate to +/- 1 uS. Other requirements for the timer are defined in the downstream port PM state machine descriptions.

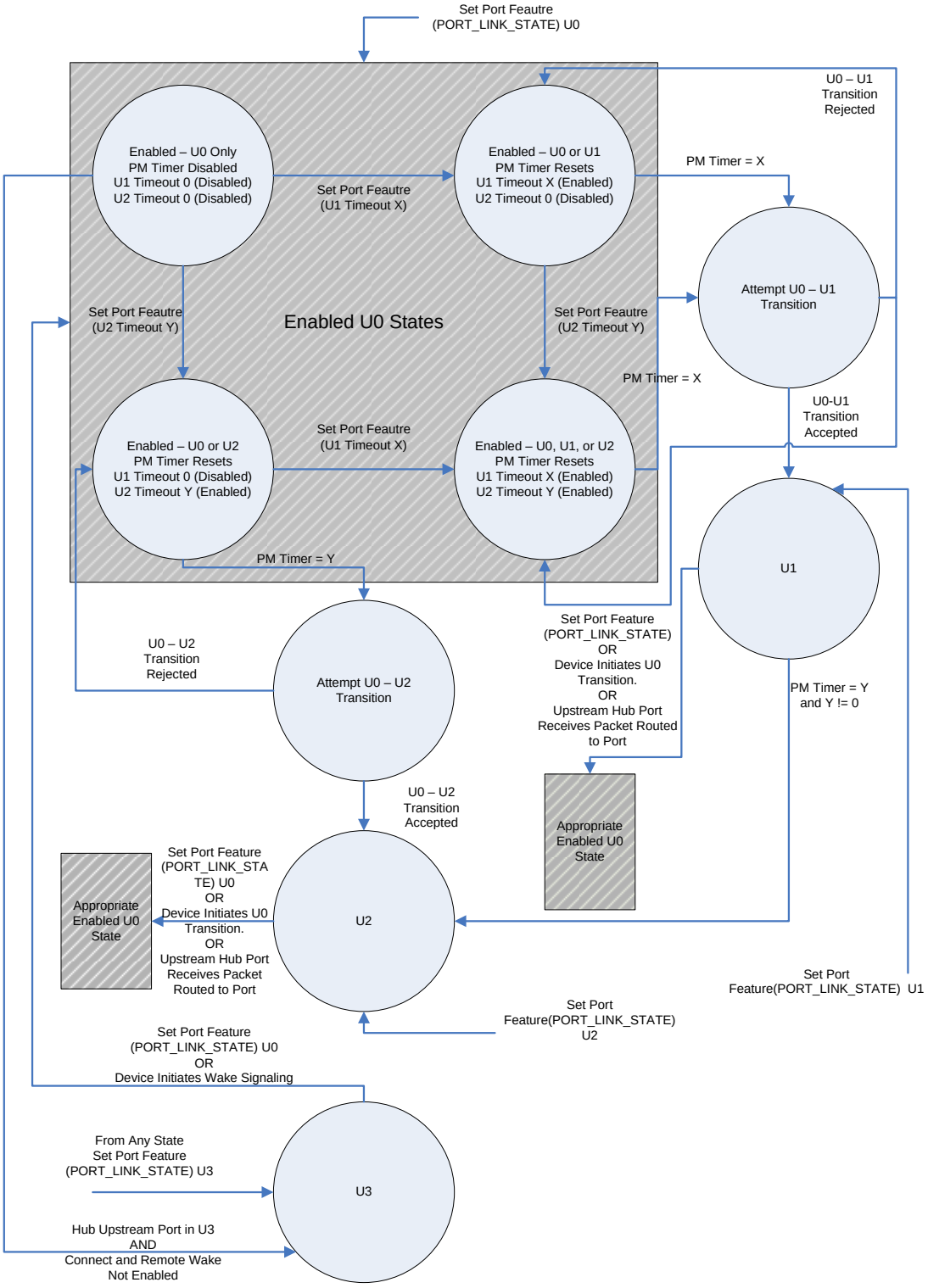


Figure 10-10. Downstream Facing Hub Port Power Management State Machine

10.4.2 Downstream Facing Port State Descriptions

10.4.2.1 Enabled U0 States

There are 4 enabled U0 states that differ only in the values that are configured for the U1 and U2 timeouts. The U1 and U2 timeout values for the port reset to zero when the port enters an enabled U0 state for the first time after a power on reset. The U1 or U2 timeout values may be set by software with a SetPortFeature(PORT_U1_TIMEOUT) or SetPortFeature(PORT_U2_TIMEOUT) command at any time. The port behaves as follows for the various combinations of U1 and U2 timeout values:

U1_TIMEOUT = 0, U2_TIMEOUT = 0

- This is the default state before the hub has received any SetPortFeature(PORT_U1/U2_TIMEOUT) requests for the port.
- The port's link shall reject all U1 or U2 transition requests by the link partner.
- The PM timer may be disabled and the PM timer value shall be ignored.
- The port's link shall not attempt to initiate transitions to U1 or U2.

U1_TIMEOUT = X > 0 , U2_TIMEOUT = 0

- The port's link shall reject all U2 transition requests by the link partner.
- The PM timer shall be reset when this state is entered and is active.
- The port's link shall accept U1 entry requests by its link partner unless the hub has one or more packets/link commands to transmit on the port.
- If the U1 timeout is 0xFF, the port shall be disabled from initiating U1 entry but shall accept U1 entry requests by the link partner unless the hub has one or more packets/link commands to transmit on the port.
- If the U1 timeout is not 0xFF and the PM timer reaches X the port's link shall initiate a transition to U1.

U1_TIMEOUT = 0, U2_TIMEOUT = Y > 0

- The port's link shall reject all U1 transition requests by the link partner.
- The PM timer shall be reset when this state is entered and is active.
- The port's link shall accept U2 entry requests by its link partner unless the hub has one or more packets/link commands to transmit on the port.
- If the U2 timeout is 0xFF, the port shall be disabled from initiating U2 entry but shall accept U2 entry requests by the link partner unless the hub has one or more packets/link commands to transmit on the port.
- If the U2 timeout is not 0xFF and the PM timer reaches Y the port's link shall initiate a direct transition from U0 to U2. In this case PORT_U2_TIMEOUT represents an amount of inactive time in U0.

U1_TIMEOUT =X > 0, U2_TIMEOUT = Y > 0

- The PM timer is reset when this state is entered and is active.
- The port's link shall accept U1 or U2 entry requests by its link partner unless the hub has one or more packets/link commands to transmit on the port.
- If the U1 timeout is 0xFF, the port shall be disabled from initiating U1 entry but shall accept U1 entry requests by the link partner unless the hub has one or more packets/link commands to transmit on the port.

- If the U1 timeout is not 0xFF and the PM timer reaches X the port's link shall initiate a transition to U1.
- If the U2 timeout is not 0xFF and the PM timer reaches Y the port's link shall initiate a direct transition from U1 to U2. In this case PORT_U2_TIMEOUT represents an amount of time in U1.

The U1 and U2 timeout values may be set by software at any time.

A port transitions to one of the Enabled U0 states (depending on the U1 and U2 Timeout values) in any of the following situations:

- From any state if the hub receives a SetPortFeature(PORT_LINK_STATE) U0 request.
- From U1 if the link partner successfully initiates a transition to U0.
- From U2 if the link partner successfully initiates a transition to U0.
- From U1 if the hub successfully initiates a transition to U0 after receiving a packet routed to the port.
- From U2 if the hub successfully initiates a transition to U0 after receiving a packet routed to the port
- From an attempt to transition from the U0 to the U1 state if the downstream port's link partner rejects the transition attempt
- From an attempt to transition from the U0 to the U2 state if the downstream port's link partner rejects the transition attempt
- From U3 if the upstream port of the hub receives wakeup signaling and the downstream port received wakeup signaling while it was in U3.

Whenever a downstream port or its link partner initiates a transition from U1 to one of the Enabled U0 states and the upstream port is not in U0, the hub shall initiate a transition to U0 on the upstream port within tHubPort2PortExitLat of when the transition was initiated on the downstream port. If the upstream port is in U0, it shall remain in U0 while the downstream port transitions to U0.

Whenever a downstream port or its link partner initiates a transition from U2 to one of the Enabled U0 states and the hub upstream port is not in U0:

- If the hub upstream port's link is in U2, the hub shall initiate a transition to U0 on the upstream port's link within tHubPort2PortExitLat of when the transition was initiated on the downstream port.
- If the hub upstream port's link is in U1, the hub upstream port shall be in U0 or initiate a transition to U0 within tHubPort2PortExitLat + U2DevExitLat-U1DevExitLat of when the transition was initiated on the downstream port.

Note: If the upstream port of the hub receives a packet that is routed to a downstream port that is in U3 the hub behavior is undefined.

10.4.2.2 Attempt U0 – U1 Transition

In this state the port attempts to transition its link from the U0 state to the U1 state.

A port shall attempt to transition to the U1 state in any of the following situations:

- The PM timer reaches the U1 timeout value.
- The hub receives a SetPortFeature(PORT_LINK_STATE) U1 request.
- The downstream port's link partner initiates a U0-U1 transition.

If the transition attempt fails, the port returns to the appropriate enabled U0 state. However, if this state was entered due to a SetPortFeature request, the port continues to attempt the U0-U1 transition on its link. Note that a SetPortFeaturue request would typically only be used for U1 entry for test purposes.

10.4.2.3 Attempt U0 – U2 Transition

In this state the port attempts to transition the link from the U0 state to the U2 state.

A port shall attempt to transition to the U2 state in any of the following situations:

- The PM timer reaches the U2 timeout value.
- The hub receives a SetPortFeature(PORT_LINK_STATE) U2 request.
- The downstream port's link partner initiates a U0-U2 transition.

If the transition attempt fails, the port returns to the appropriate enabled U0 state. However, if this state was entered due to a SetPortFeature request, the port continues to attempt the U0-U2 transition. Note that a SetPortFeaturue request would typically only be used for U2 entry for test purposes.

10.4.2.4 Link in U1

Whenever a downstream port enters U1 and all downstream ports are now in the U1 or a lower power state the hub shall initiate a transition to U1 on the upstream port within tHubPort2PortExitLat if the upstream port is enabled for U1.

10.4.2.5 Link in U2

The following rules apply when a downstream port enters U2:

- If all downstream ports are now in the U2 or a lower power state, the hub shall initiate a transition to U2 on the upstream port within tHubPort2PortExitLat, if the upstream port is enabled for U2. If U2 is not enabled on the upstream port, but U1 is enabled, the hub shall initiate a transition to U1 with the same timing requirements.
- If all downstream ports are now in the U1 or lower power state, the hub shall initiate a transition to U1 on the upstream port within tHubPort2PortExitLat, if the upstream port is enabled for U1.

10.4.2.6 Link in U3

The following rules apply when a downstream port enters U3:

- If all downstream ports are now in the U3 or a lower power state, the hub shall initiate a transition to the lowest enabled power state on the upstream port within tHubPort2PortExitLat.
- If all downstream ports are now in the U2 or lower power state, the hub shall initiate a transition to the lowest enabled power state above U3 on the upstream port within tHubPort2PortExitLat.
- If all downstream ports are now in the U1 or lower power state, the hub shall initiate a transition to U1 on the upstream port within tHubPort2PortExitLat, if the upstream port is enabled for U1.

10.5 Upstream Facing Port

The following sections provide a functional description of a state machine that exhibits correct behavior for a hub upstream facing port. These sections also apply to the upstream facing port on a device unless exceptions are specifically noted. An upstream port shall only attempt to connect to the Superspeed and USB 2.0 interfaces as described by the upstream port state machine in the following sections.

Figure 10-11 is an illustration of the upstream facing port state machine. Each of the states is described in section 10.5.x. In the diagram below, some of the entry conditions into states are shown without origin. These conditions have multiple origin states and the individual transitions lines are not shown so that the diagram can be simplified. The description of the entered state indicates from which states the transition is applicable.

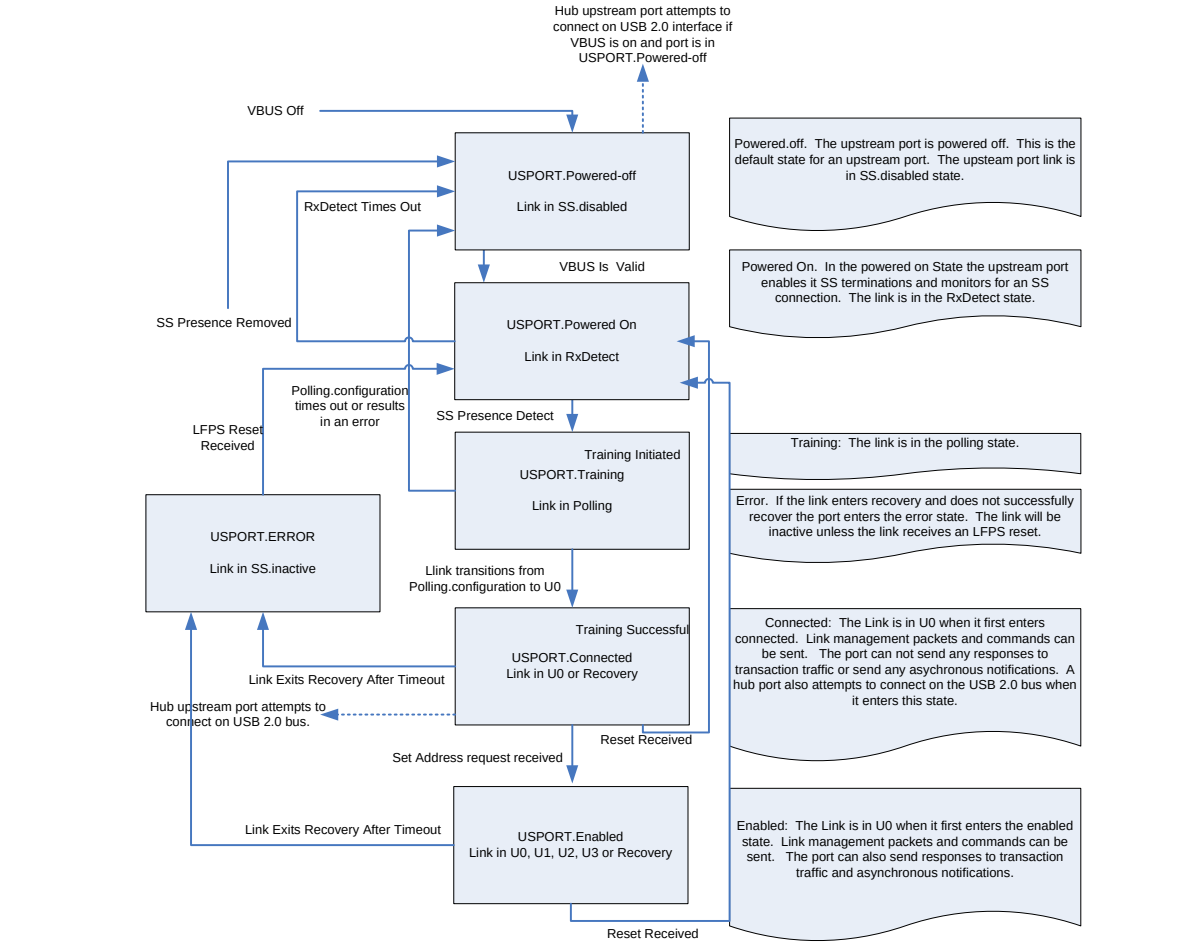


Figure 10-11. Upstream Facing Hub Port State Machine

Table 10-3. Upstream Facing Port Signal/Event Definitions

Signal/Event Name	Event/Signal Source	Description
Power_source_off	Implementation-dependent	Power to the port not available due to over-current or termination of source power (e.g., external power removed)
Disconnect_Detect	Internal	Disconnect seen at port
Port_Error	Internal	Error condition detected

10.5.1 Upstream Facing Port State Descriptions

10.5.1.1 USPORT.Powered-off

The USPORT.Powered-off state is the default state for an upstream facing port.

A port shall transition into this state if any of the following situations occur:

- From any state when VBUS is removed.
- From the USPORT.Training state if the port's link times out from the polling.config state.
- From the USPORT.Powered-on if RxDetect times out.

In this state, the port's link shall be in the SS.disabled state.

If the port enters the USPORT.Powered-off state and VBUS is still present the port shall attempt to connect on the USB 2.0 bus.

10.5.1.2 USPORT.Powered-on

A port shall transition into this state in any of the following situations:

- From the USPORT.Powered-off state when VBUS is valid.
- From the USPORT.Error state when the link receives a warm reset.
- Form the USPORT.Connected state when the link receives any reset.
- From the USPORT.Enabled state when the link receives any reset.

In this state the port's link shall be in the SS.RxDetect state. While in this state, if the USB 2.0 portion of the hub enters the suspended state, the total hub current draw from VBUS shall meet the suspend current limit.

10.5.1.3 USPORT.Training

A port transitions to this state from the USPORT.Powered-on state when SuperSpeed terminations are detected.

In this state the port's link shall be in the polling state.

10.5.1.4 USPORT.Connected

A port transitions to this state from the USPORT.Training state when link enters U0 from polling.configuration.

In this state the port's link shall be in the U0 or recovery states.

The port may send link management packets or link commands but shall not transmit any other packets while in the connected state.

If the port enters the USPORT.Connected state and is not connected on the USB 2.0 bus it shall attempt to connect on the USB 2.0 bus. This requirement only applies to upstream ports on hubs.

10.5.1.5 USPORT.ERROR

A port transitions to this state when a serious error condition occurred while attempting to operate the link. A port transitions to this state in any of the following situations:

- From the USPORT.Connected state if the link enters recovery and times out without recovering.
- From the USPORT.Enabled state if the link enters recovery and times out without recovering.

In this state the port's link shall be in the SS.Inactive state.

A port exits the error state only if an LFPS reset is received on the link.

10.5.1.6 USPORT.Enabled

A port transitions to this state from the USPORT.Connected state when a Set Address request is received and the Ack TP response to the status stage of the Set Address has been sent and successfully acknowledged at the link level.

In this state the port's link shall be in the U0, U1, U2, U3 or recovery states.

The port may send any type of packet while in the enabled state.

The link shall be in U0 when the enabled state is entered.

10.6 Upstream Facing Port Power Management

If enabled, port status change interrupts, e.g., due to a connect event on a downstream port, will cause the upstream link to initiate a transition to U0.

10.7 Hub Header Forwarding and Data Repeater

The Hub uses a store and forward model for headers and a repeater model for data that combined provide the following functionality:

Downstream

- Validates data packet headers
- Sets up connection to selected downstream port
- Forwards packet header to downstream port
- Forwards data packet payload to downstream port if present
  - — Sets up and tears down connectivity on packet boundaries
  - — Ensures orderly entry into and out of the Suspend state, including proper handling of remote wakeups

Upstream

- Validates data packet header
- Sets up connection to upstream port
- Forwards packet header to upstream port
- Forwards data packet payload to upstream port if present
  - — Sets up and tears down connectivity on packet boundaries
  - — Ensures orderly entry into and out of the Suspend state, including proper handling of remote wakeups

10.7.1 Hub Elasticity Buffer

There are no direct specifications for elasticity buffer behavior in a hub. However, note that a hub must meet the requirements in Section 10.7.3 for the maximum variation in propagation delay for headers that are forwarded from the hub upstream port to a downstream port.

10.7.2 SKP Ordered Sets

A hub transmits SKP ordered sets following the rules for all transmitters in the link chapter for all transmissions.

10.7.3 Interpacket Spacing

When a hub originates packets it shall follow the rules for interpacket spacing set in the protocol chapter. Minimum interpacket spacing rules are set to allow hubs opportunities to insert link commands and/or headers between packets while forwarding a constant stream of packets.

When a hub is forwarding packets upstream it shall forward packets following the priority rules in the link chapter. Data packet headers and data packet payloads shall be forwarded contiguously without any other symbols between the data packet header and datapacket payload. No other spacing rules apply for packets forwarded upstream by a hub. When a hub is forwarding packets downstream it shall forward packets following the priority rules in the link chapter. Datapacket headers and data packet payloads shall be forwarded contiguously without any other symbols between the data packet header and data packet payload. A hub shall not forward a pair of packets downstream with a smaller gap between the last framing symbol of the first packet and the first framing symbol of the second packet then the gap when the packets were received on the hub upstream port. A hub shall only increase the received gap between a pair of packets forwarded downstream to insert link commands. Additional examples are covered in the Hub TX and RX state machines in this chapter.

10.7.4 Header Buffer Architecture

The specification does not require a specific architecture for the header buffers in a hub. A specific architecture that meets the functional requirements of this specification is shown in Figure 10-12 and Figure 10-13 to provide an example for the purposes of better illustrating the functional behavior of a hub. Figure 10-12 shows a hub with a four header RX buffer for the upstream port and a four header TX buffer for each of the downstream ports. Figure 10-13 shows a four header RX buffer for each of the downstream ports and a four header TX buffer for the upstream port. The buffers shown in Figure 10-12 and Figure 10-13 are independent physical buffers.

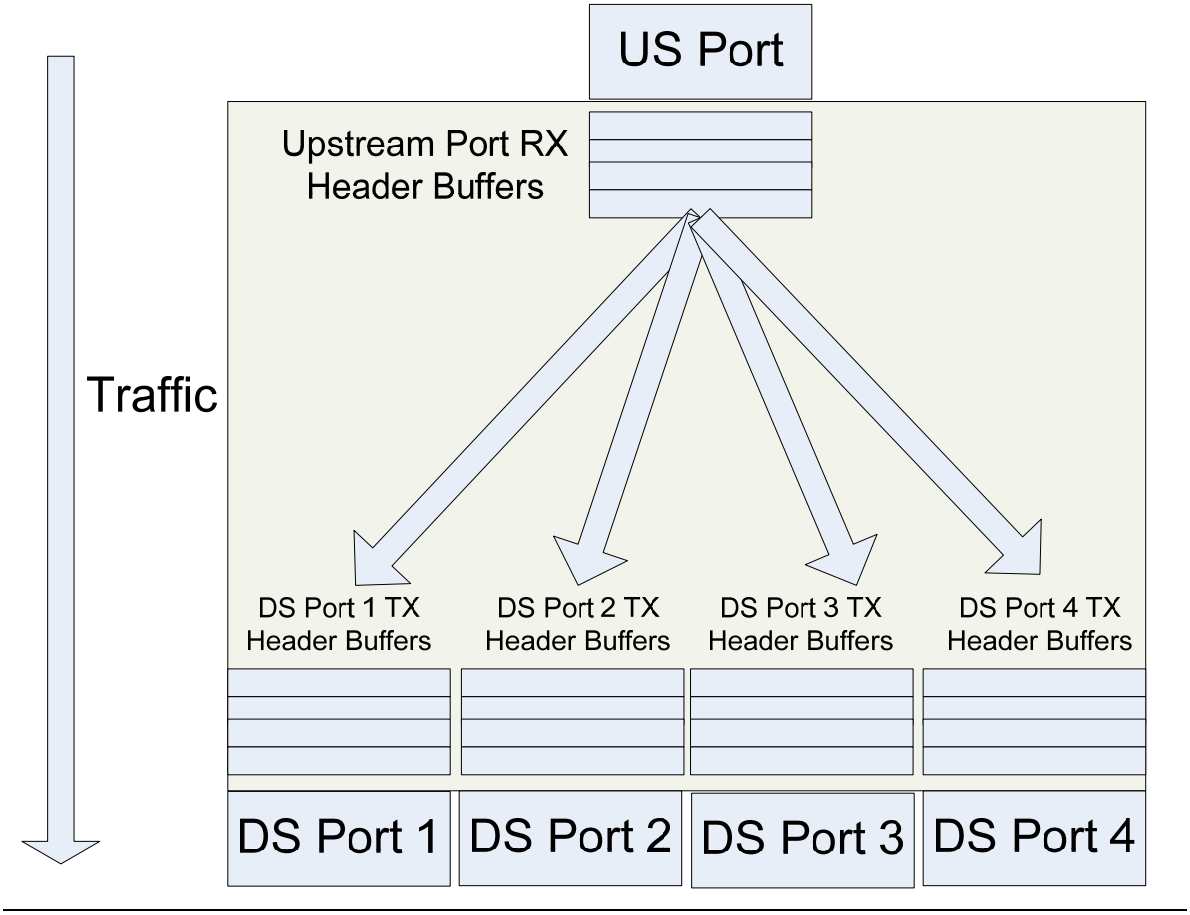


Figure 10-12. Example Hub Header Buffer Architecture - Downstream Traffic

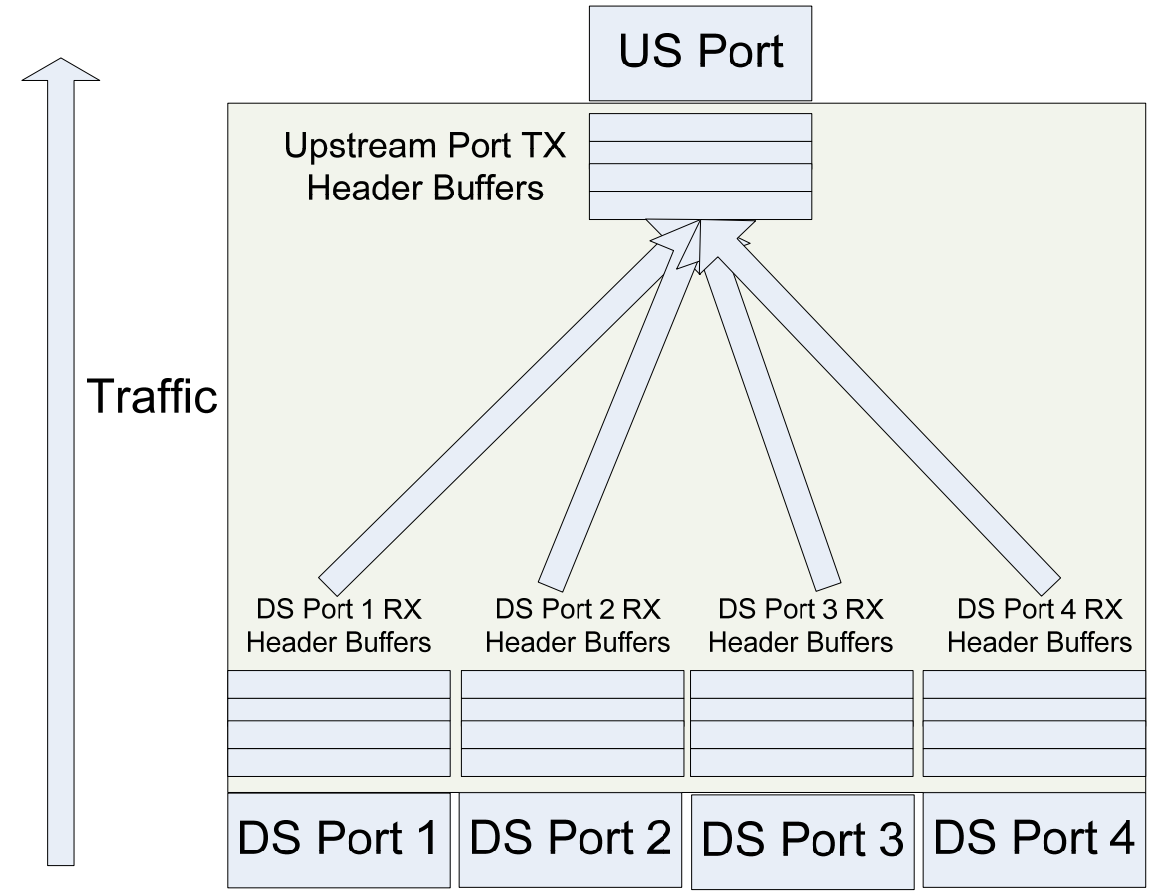


Figure 10-13. Example Hub Header Buffer Architecture - Upstream Traffic

The following lists functional requirements for a hub buffer architecture:

- A hub starting with all header buffers empty shall be able to receive at least eight headers directed to the same downstream port that is not in U0 before its upstream port runs out of header flow control credits.
- A hub starting with all header buffers empty shall be able to receive at least four headers directed to the same downstream port that is not in U0 before downstream header traffic to other ports is blocked behind these headers.
- A hub starting with all header buffers empty shall be able to receive at least eight headers on the same downstream port directed for upstream transmission when the upstream port is not in U0.
- Headers transmitted by a downstream port shall be transmitted in the order they were received on the upstream port.
- Headers transmitted by an upstream port from the same downstream port shall be transmitted in the order they were received on the downstream port.

Sections 10.7.6, 10.7.8, 10.7.10, and 10.7.12 provide detailed functional state machines for the upstream and downstream port TX and RX header buffers in a hub implementation.

10.7.5 Upstream Facing Port TX

This section describes the functional requirements of the upstream facing port TX state machine.

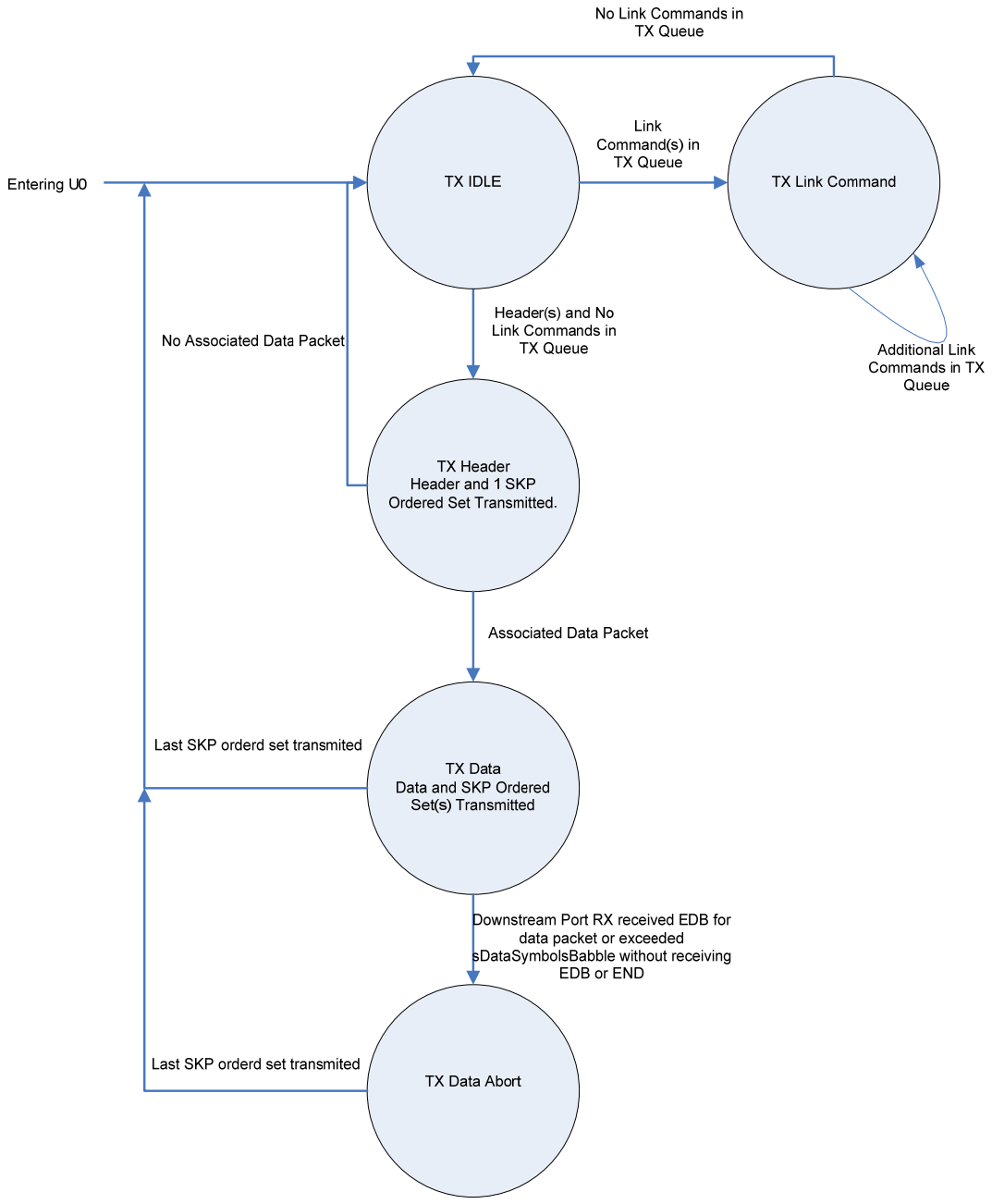


Figure 10-14. Upstream Facing Port TX State Machine

10.7.6 Upstream Facing Port TX State Descriptions

10.7.6.1 TX Idle

In the TX Idle state the upstream port transmitter is actively transmitting idle symbols. A port shall transition to the TX Idle state in any of the following situations:

- From the TX Data, TX Data Abort, or TX Header state after the last required SKP ordered set is transmitted
- From the TX Link Command state after a link command is transmitted
- As the default state when the link enters U0

When the upstream port transmitter enters the TX idle state a TX idle timer is set to zero. The TX idle timer increments by one each time an idle symbol is transmitted. When the TX idle timer reaches 50 the transmitter shall transmit an LUP link command and reset the TX idle timer to zero.

10.7.6.2 TX Header

In the TX Header state the upstream port transmitter is actively transmitting a header.

Note: A hub shall not abort the transmission of a header with Abnormal Data Packet Payload Ending Frame Ordered Set.

A port shall transition to the TX Header state in any of the following situations:

- From the TX IDLE state when there are one or more headers queued for transmission and there are no link commands queued for transmission.

10.7.6.3 TX Data

In the TX Data state the upstream port transmitter is actively transmitting a data packet payload. After transmitting the end framing symbols for the data packet and required SKP ordered sets the port may remove the data packet from hub storage. A hub shall not retransmit a DPP packet under any circumstances.

A port shall transition to the TX Data state from the TX Header state when there is a data packet payload associated when the data packet header that was transmitted. The data packet payload transmission shall begin immediately after transmission of the last symbol of the header.

10.7.6.4 TX Data Abort

In the TX Data abort state the upstream port transmitter aborts the normal transmission of a data packet payload by transmitting Abnormal Data Packet Payload Ending Frame Ordered Set framing symbols and required SKP ordered sets. The port then removes the data packet payload from hub storage.

A port shall transition to the TX Data Abort state from the TX Data state when the downstream port receiving the data packet payload receives an Abnormal Data Packet Payload Ending Frame Ordered Set or has received sDataSymbolsBabble symbols without receiving a valid Data Packet Payload Ending Frame Ordered Set or Abnormal Data Packet Payload Ending Frame Ordered Set.

10.7.6.5 TX Link Command

In the TX Link Command state the upstream port transmitter is actively transmitting a link command. If multiple link commands are transmitted in the TX Link Command state, they shall be transmitted without a gap.

A port shall transition to the TX Link Command state in any of the following situations:

- From the TX IDLE state when there are one or more link commands queued for transmission.
- From the TX Link Command state when there are additional link commands queued for transmission

10.7.7 Upstream Facing Port RX

This section describes the functional requirements of the upstream facing port RX state machine.

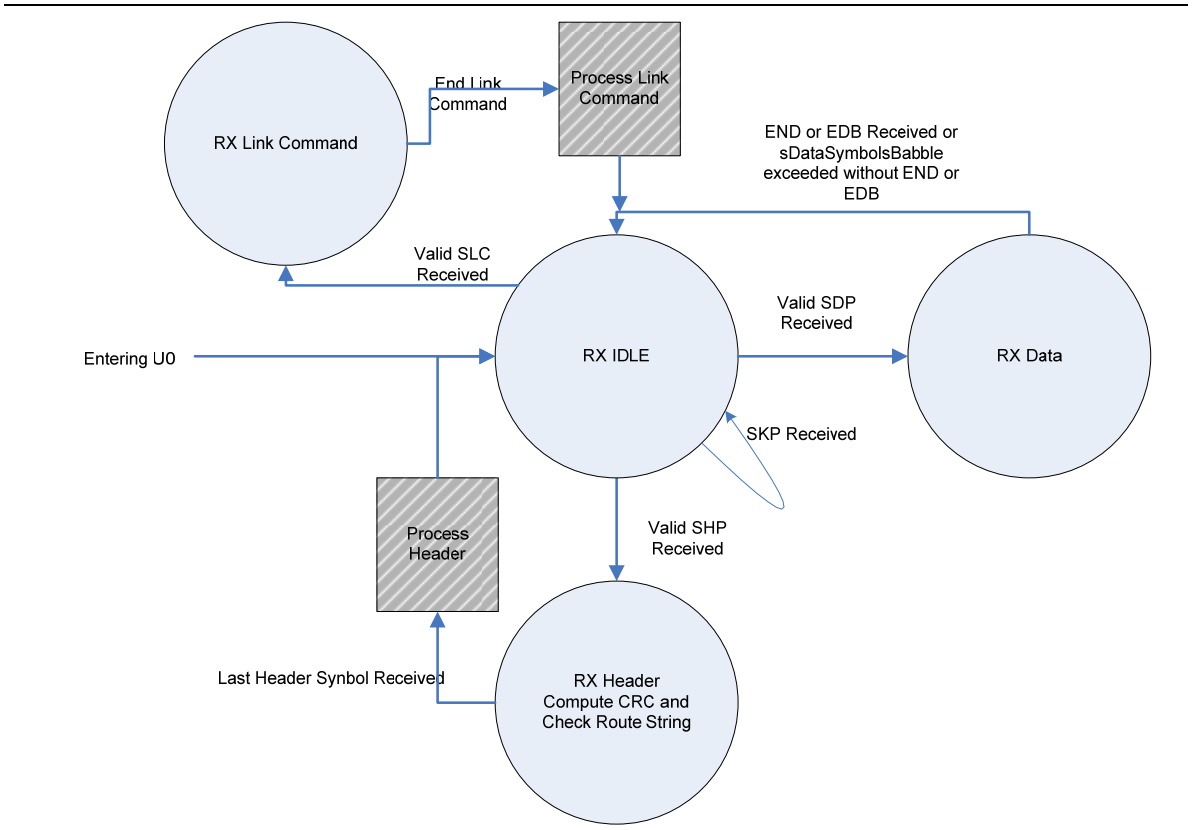


Figure 10-15. Upstream Facing Port RX State Machine

10.7.8 Upstream Facing Port RX State Descriptions

10.7.8.1 RX Idle

In the RX Idle state, the upstream port receiver is actively processing received symbols and looking for Data Packet Payload Starting Frame Ordered Set, Packet Start Framing Ordered Set, or SLC framing symbols to begin receiving a packet.

A port shall transition to the RX Idle state in any of the following situations:

- From the RX Data state when an Data Packet Payload Ending Frame Ordered Set or Abnormal Data Packet Payload Ending Frame Ordered Set is received
- From the RX Header state when the last symbol in a header is received
- From the Rx Data state when sDataSymbolsBabble is reached without receiving an Data Packet Payload Ending Frame Ordered Set or Abnormal Data Packet Payload Ending Frame Ordered Set.
- After receiving a link command
- As the default state when the link enters U0

10.7.8.2 RX Data

In the RX Data state the upstream port receiver is actively processing received symbols and looking for an Data Packet Payload Ending Frame Ordered Set or Abnormal Data Packet Payload Ending Frame Ordered Set. The receiver shall start a count of received symbols at zero when entering this state. The first symbol counted is the first symbol after the SDP.

A port shall transition to the RX Data state when it receives a valid Data Packet Payload Starting Frame Ordered Set.

If the port receives an Abnormal Data Packet Payload Ending Frame Ordered Set or detects a Data Babble Error, it may clear the data from its buffers once it has transmitted received DPP including the Abnormal Data Packet Payload Ending Frame Ordered Set on the appropriate downstream port. The hub shall transmit the received symbols for a data packet payload including the Abnormal Data Packet Payload Ending Frame Ordered Set on the appropriate downstream port when an Abnormal Data Packet Payload Ending Frame Ordered Set is received or a Data Babble Error is detected on the upstream port. Note that this requirement applies even when the hub receives an Abnormal Data Packet Payload Ending Frame Ordered Set or detects a Data Babble Error before beginning to transmit the DPP packet on the hub downstream port. Otherwise it transmits an Abnormal Data Packet Payload Ending Frame Ordered Set on the downstream port to terminate the data transmission.

The hub shall have at least 1080 bytes of buffering for data packets received on the upstream port..

10.7.8.3 RX Header

In the RX header state the upstream port receiver is actively processing received symbols until the last header symbol is received. The receiver shall start a count of received symbols at zero when entering this state. The first symbol counted is the first symbol after the Packet Start Framing Ordered Set.

A port shall transition to the RX Header state when it receives a valid Packet Start Framing Ordered Set.

Note: The port shall finish calculating the CRC-16, the Link Control Word CRC-5, and check the route string within four symbol times after the last symbol of the header is received. Implementations may have to begin the CRC calculation as symbols are received and check the route string before the header is verified to meet this requirement.

10.7.8.4 Process Header

Once the final symbol for a header is received the port shall perform all additional processing necessary for the header. Any such processing shall not block the port from immediately returning to the RX idle state and continuing to process received symbols.

A port performs additional header processing in any of the following situations. The additional processing steps in each situation are described.

- When the final header symbol is received in the RX header state and the header CRC-16 and Link Control Word CRC-5 are valid.
  - — If the header is routed to a downstream port that is not in U0:
    1. The hub initiates U0 entry on the appropriate downstream port link. U0 entry shall be initiated tDownLinkStateChange from when the hub received the route string of the header.
    2. The header is marked deferred (if it is not already marked deferred) and the correct Link Control Word CRC-5 is re-calculated for modified header.
    3. If the header was marked deferred in step 2, a deferred header including the hub depth and correct CRC-5 is queued into the upstream port TX header buffer.
    4. The header is queued for transmission on the appropriate downstream port.
    5. If the upstream port RX header buffer has at least four free slots the appropriate LGOODx link command is queued for transmission by the upstream port. Otherwise, the appropriate LGOODx link command is queued for transmission once the RX header buffer slot used for the header is available.
    6. If the header is a data packet header the next packet is discarded if it is a data packet payload.
  - — If the header is routed to a downstream port that is in U0:
    1. If the downstream port TX header buffer queue is not empty (there is at least one header in the queue that has not been completely transmitted), the header is marked delayed and the correct Link Control Word CRC-5 is re-calculated for modified header.
    2. The header is queued for transmission on the appropriate downstream port.

       Note: If the queue for the appropriate downstream port is full the header is queued as soon as a space is available in the appropriate downstream port queue. The hub shall still process subsequent headers normally while a downstream port queue is full if they are directed to a different downstream port.

    3. If the upstream port RX header buffer has at least four free slots the appropriate LGOODx link command is queued for transmission by the upstream port. Otherwise, the appropriate LGOODx link command is queued for transmission once the RX header buffer slot used for the header is available.

— If the header is not routed to a downstream port.

If the upstream port RX header buffer has at least four free slots the appropriate LGOODx link command is queued for transmission by the upstream port. Otherwise, the appropriate LGOODx link command is queued for transmission once the RX header buffer slot used for the header is available.

- When the final header symbol is received in the RX header state and the header CRC-16 or Link Control Word CRC-5 is invalid.
  - — The header is removed from the RX header buffer.
  - — An LBAD link command is queued for transmission by the upstream port.
  - — Note: All subsequent headers are silently discarded until a retry of the invalid header is received.

TBD – Add example of what a hub would return for a specific header in the case of a deferred header.

10.7.8.5 RX Link Command

In the RX Link Command state the upstream port receiver is actively processing received symbols and looking for the end of a link command (second instance of link command symbol).

A port transitions to the RX Link Command state when it receives a valid SLC.

10.7.8.6 Process Link Command

Once the final symbol for a link command is received the port shall perform all additional processing necessary for the link command. Any such processing shall not block the port from immediately returning to the RX idle state and continuing to process received symbols.

10.7.9 Downstream Facing Port TX

This section describes the functional requirements of the downstream facing port TX state machine.

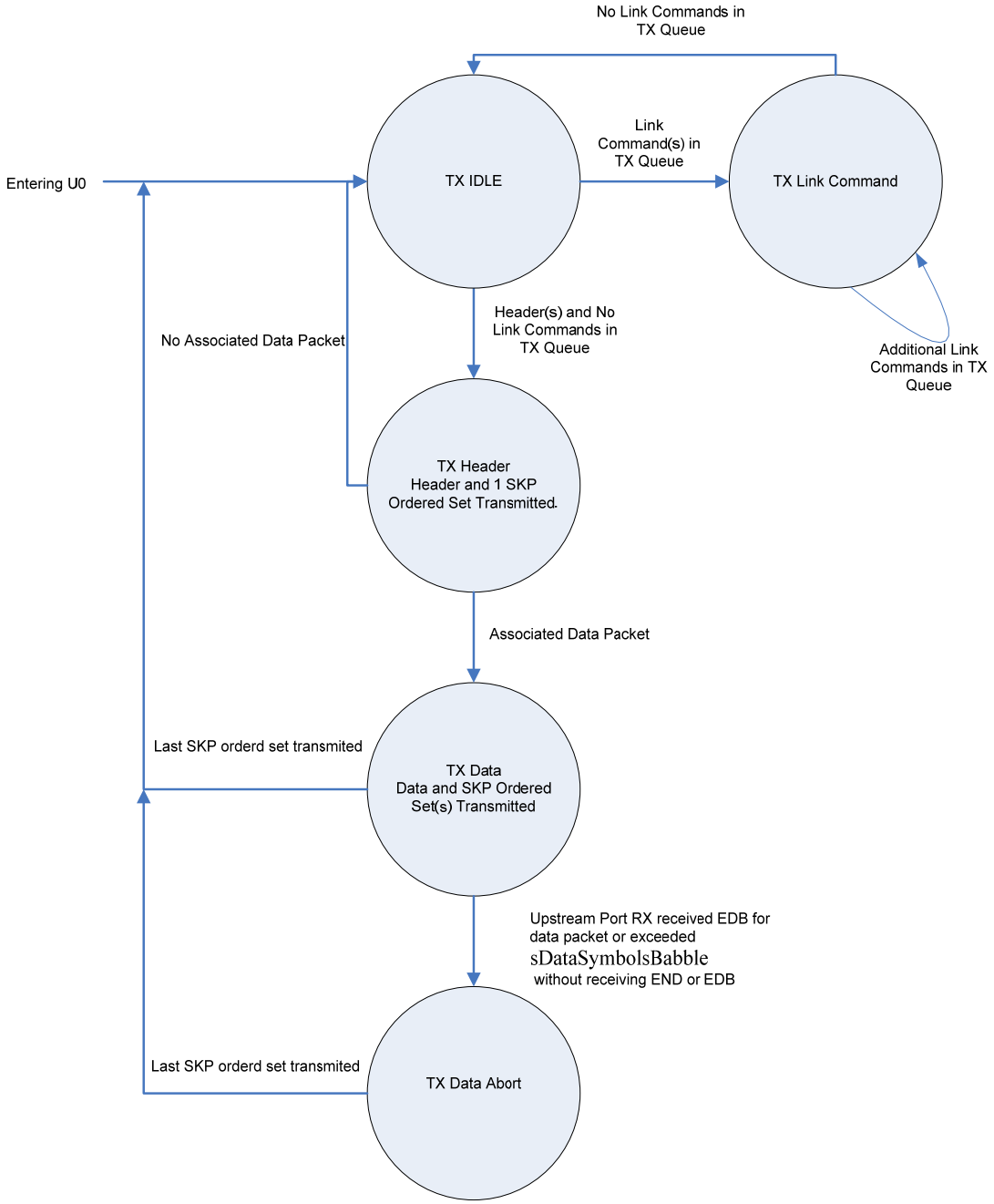


Figure 10-16. Downstream Facing Port TX State Machine

10.7.10 Downstream Facing Port TX State Descriptions

10.7.10.1 TX Idle

In the TX Idle state the downstream port transmitter is actively transmitting idle symbols. A port shall transition to the TX Idle state in any of the following situations:

- From the TX Data, TX Data Abort, or TX Header state after the last required SKP ordered set is transmitted
- From the TX Link Command state after a link command is transmitted
- As the default state when the link enters U0

10.7.10.2 TX Header

In the TX Header state the downstream port transmitter is actively transmitting a header.

Note: A hub shall not abort the transmission of a header with Abnormal Data Packet Payload Ending Frame Ordered Set.

A port shall transition to the TX Header state in any of the following situations:

- From the TX IDLE state when there are one or more headers queued for transmission and there are no link commands queued for transmission

10.7.10.3 TX Data

In the TX Data state the downstream port transmitter is actively transmitting a data packet payload. After transmitting the end framing symbols for the data packet and required SKP ordered sets the port may remove the data packet from hub storage. A hub shall not retransmit a DPP packet under any circumstances.

A port shall transition to the TX Data state from the TX Header state when there is a data packet payload associated when the data packet header that was transmitted. The data packet payload transmission shall begin immediately after transmission of the last symbol of the header.

10.7.10.4 TX Data Abort

In the TX Data abort state the downstream port transmitter aborts the normal transmission of a data packet payload by transmitting Abnormal Data Packet Payload Ending Frame Ordered Set framing symbols and required SKP ordered sets. The port then removes the data packet payload from hub storage.

A port transitions to the TX Data Abort state from the TX Data state when the upstream port receiving the data packet payload receives an Abnormal Data Packet Payload Ending Frame Ordered Set or has received sDataSymbolsBabble symbols without receiving a valid Data Packet Payload Ending Frame Ordered Set or Abnormal Data Packet Payload Ending Frame Ordered Set.

10.7.10.5 TX Link Command

In the TX Link Command state the downstream port transmitter is actively transmitting a link command. If multiple link commands are transmitted in the TX Link Command state, they shall be transmitted without a gap.

A port shall transition to the TX Link Command state in any of the following situations:

- From the TX IDLE state when there are one or more link commands queued for transmission.
- From the TX Link Command state when there are additional link commands queued for transmission.

10.7.11 Downstream Facing Port RX

This section describes the functional requirements of the downstream facing port RX state machine.

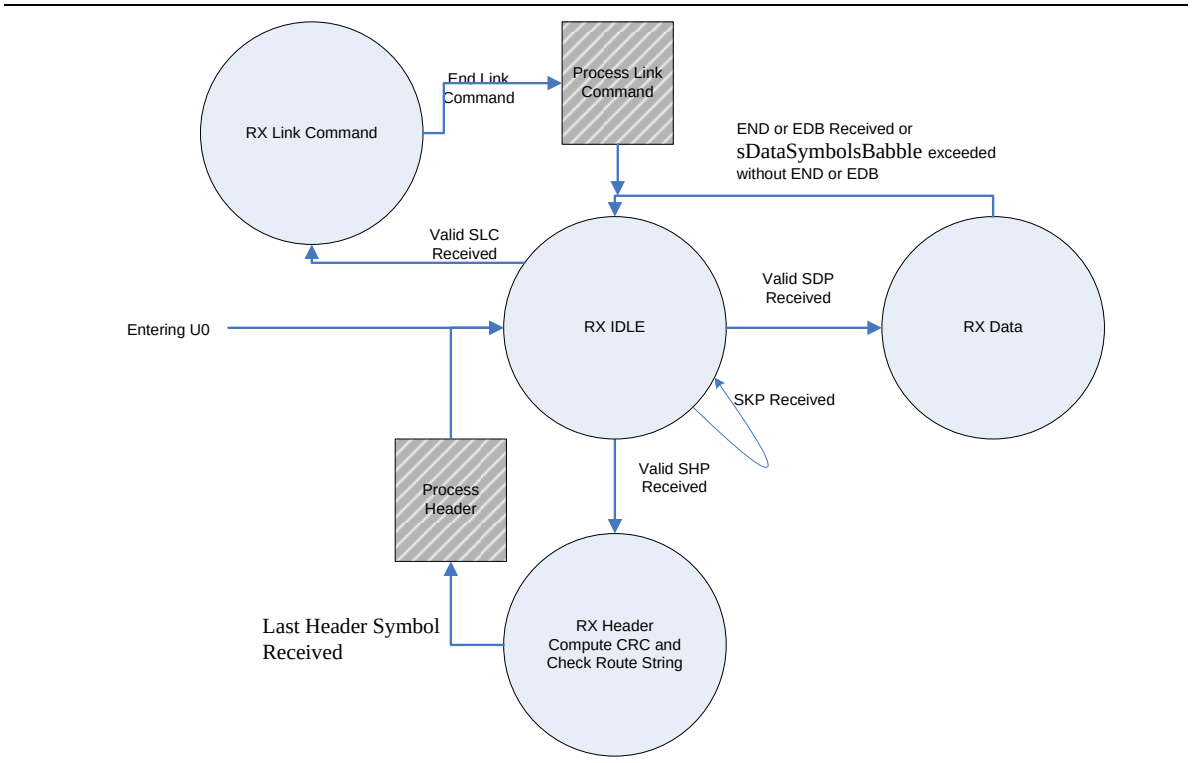


Figure 10-17. Downstream Facing Port RX State Machine

10.7.12 Downstream Facing Port RX State Descriptions

10.7.12.1 RX Idle

In the RX Idle state, the downstream port receiver is actively processing received symbols and looking for Data Packet Payload Starting Frame Ordered Set, Packet Start Framing Ordered Set, or SLC framing symbols to begin receiving a packet.

A port shall transition to the RX Idle state in any of the following situations:

- From the RX Data state when an Data Packet Payload Ending Frame Ordered Set or Abnormal Data Packet Payload Ending Frame Ordered Set is received
- From the RX Header state when the last symbol in a header is received
- From the Rx Data state when sDataSymbolsBabble is reached without receiving an Data Packet Payload Ending Frame Ordered Set or Abnormal Data Packet Payload Ending Frame Ordered Set.
- After receiving a link command
- As the default state when the link enters U0

10.7.12.2 RX Data

In the RX Data state the downstream port receiver is actively processing received symbols and looking for an Data Packet Payload Ending Frame Ordered Set, or Abnormal Data Packet Payload Ending Frame Ordered Set. The receiver shall start a count of received symbols at zero when entering this state. The first symbol counted is the first symbol after the Data Packet Payload Starting Frame Ordered Set.

A port shall transition to the RX Data state when it receives a valid Data Packet Payload Starting Frame Ordered Set.

If the port receives an Abnormal Data Packet Payload Ending Frame Ordered Set or detects a Data Babble Error it may clear the data from its buffers once it has transmitted received DPP including the Abnormal Data Packet Payload Ending Frame Ordered Set on the upstream port. The hub shall transmit the received symbols for a data packet payload including the Abnormal Data Packet Payload Ending Frame Ordered Set on its upstream port when an Abnormal Data Packet Payload Ending Frame Ordered Set is received or a Data Babble Error is detected on the downstream. Note that this requirement applies even when the hub receives an Abnormal Data Packet Payload Ending Frame Ordered Set or detects a Data Babble Error before beginning to transmit the DPP packet on the hub upstream port. The hub shall have at least 1080 bytes of shared buffering for data packets received on all downstream ports.

10.7.12.3 RX Header

In the RX header state the downstream port receiver is actively processing received symbols until the last header symbol is received. The receiver shall start a count of received symbols at zero when entering this state. The first symbol counted is the first symbol after the Packet Start Framing Ordered Set.

A port shall transition to the RX Header state when it receives a valid Packet Start Framing Ordered Set.

Note: The port shall finish calculating the CRC-16 and the Link Control Word CRC-5 within 4 symbol times after the last symbol of the header is received. Implementations may have to begin the CRC calculation as symbols are received to meet this requirement.

10.7.12.4 Process Header

Once the final symbol for a header is received the port shall perform all additional processing necessary for the header. Any such processing shall not block the port from immediately returning to the RX idle state and continuing to process received symbols.

A port performs additional header processing in any of the following situations. The additional processing steps in each situation are described.

- When the final header symbol is received in the RX header state and the header CRC-16 and Link Control Word CRC-5 is valid:
  1. The header is queued for transmission on the upstream port.

     Note: If the queue for the upstream port is full the header is queued as soon as a space is available in the upstream port queue. The hub shall still process subsequent headers normally while the upstream port queue is full. If headers have been received on more than one downstream port when a space becomes available in the upstream port header queue, the hub shall prioritize a data packet header if one is waiting at the front of a downstream queue. Otherwise, the arbitration algorithm the hub uses is not specified.

  2. If the downstream port RX header buffer has at least four free slots, the appropriate LGOODx link command is queued for transmission by the downstream port. Otherwise, the appropriate LGOODx link command is queued for transmission once the RX header buffer slot used for the header is available.
- When the final header symbol is received in the RX header state and the header CRC-16 or Link Control Word CRC-5 is invalid:
  - The header is removed from the RX header buffer.
  - An LBAD link command is queued for transmission by the downstream port.
  - Note: All subsequent headers are silently discarded until a retry of the invalid header is received.

10.7.12.5 RX Link Command

In the RX Link Command state the downstream port receiver is actively processing received symbols and looking for the end of a link command (second instance of link command symbol).

A port shall transition to the RX Link Command state when it receives a valid SLC.

10.7.12.6 Process Link Command

Once the final symbol for a link command is received the port shall perform all additional processing necessary for the link command. Any such processing shall not block the port from immediately returning to the RX idle state and continuing to process received symbols.

10.7.13 SuperSpeed Packet Connectivity

The SuperSpeed hub packet repeater/forwarder must relock the packets in both directions. Relocking means that the repeater extracts the data from the received stream and retransmits the stream using its own local clock.

10.8 Suspend and Resume

Hubs must support suspend and resume both as a USB device and in terms of propagating suspend and resume signaling. Global suspend/resume refers to the entire bus being suspended or resumed without affecting any hub's downstream facing port states; selective suspend/resume refers to a downstream facing port of a hub being suspended or resumed without affecting the hub state. Superspeed hubs only support selective suspend and resume. They do not support global suspend and resume. Selective suspend/resume is implemented via requests to a hub. Device-initiated resume is called remote-wakeup.

The hub follows the same suspend requirements as a SuperSpeed device on its upstream facing port.

When a hub downstream port link is in the U3 state the following requirements apply to the hub if it receives wakeup signaling from its link partner on a downstream port:

- If the hub upstream port's link is not in U3 the hub shall drive remote wakeup signaling on the downstream link where the wakeup signaling was received in tHubDriveRemoteWakeDownstream .
- If the hub upstream port's link is in U3 the hub shall drive wakeup signaling on its upstream port in tHubPropRemoteWakeUpstream.

When a hub upstream port's link is in the U3 state and it receives wakeup signaling from its link partner on the hub upstream port's link the hub shall automatically drive remote wakeup to any downstream ports that are in U3 and have received remote wakeup signaling since entering U3.

When the hub receives a SetPortFeature(PORT_LINK_STATE) U0 for a downstream port with a link in U3 the hub shall drive remote wakeup signaling on the link in tHubDriveRemoteWakeDownstream.

10.9 Hub Reset Behavior

Reset signaling to a hub is defined only in the downstream direction, which is at the hub's upstream facing port. The reset signaling mechanism required of the hub is described in the physical layer chapter.

A suspended hub shall interpret the start of reset as a wakeup event; it shall be awake and have completed its reset sequence by the end of reset signaling.

After completion of the reset sequence, a hub is in the following state:

- Hub Controller default address is 0.
- Hub status change bits are set to zero.
- Hub Depth is reset to zero.
- Downstream facing ports are in the DSPORT.Powered-off state.

10.10 Hub Port Power Control

Self-powered hubs may have power switches that control delivery of power to downstream facing ports but it is not required. A hub with power switches can switch power to all ports as a group/gang, to each port individually, or have an arbitrary number of gangs of one or more ports.

A hub indicates whether or not it supports power switching by the setting of the Logical Power Switching Mode field in *wHubCharacteristics*. If a hub supports per-port power switching, then the power to a port is turned on when a SetPortFeature(PORT_POWER) request is received for the port. Port power is turned off when the port is in the DSPORT.Powered-off state. If a hub supports ganged power switching, then the power to all ports in a gang is turned on when any port in a gang receives a SetPortFeature(PORT_POWER) request. The power to a gang is not turned off unless all ports in a gang are in the Powered-off or Not Configured states. Note, the power to a port is not turned on by a SetPortFeature(PORT_POWER) if both C_HUB_LOCAL_POWER and Local Power Status (in *wHubStatus*) are set to 1B at the time when the request is executed and the PORT_POWER feature would be turned on. Note that a hub which supports charging applications may keep power on at other times. Refer to section 10.3.1.1 for more details on allowed behavior for a hub that supports charging applications.

Although a self-powered hub is not required to implement power switching, the hub shall support the Powered-off state for all ports. Additionally, the hub shall implement the *PortPwrCtrlMask* (all bits set to 1B) even though the hub has no power switches that can be controlled by the USB System Software.

For a hub with no power switches, *bPwrOn2PwrGood* shall be set to zero.

10.10.1 Multiple Gangs

A hub may implement any number of power and/or over-current gangs. A hub that implements more than one over-current and/or power switching gang shall set both the Logical Power Switching Mode and the Over-current Reporting Mode to indicate that power switching and over-current reporting are on a per port basis (these fields are in *wHubCharacteristics*). Also, all bits in *PortPwrCtrlMask* shall be set to 1B.

When an over-current condition occurs on an over-current protection device, the over-current is signaled on all ports that are protected by that device. When the over-current is signaled, all the ports in the group are placed in the Powered-off state, and the C_PORT_OVER_CURRENT field is set to 1B on all the ports. When port status is read from any port in the group, the PORT_OVER_CURRENT field will be set to 1B as long as the over-current condition exists. The C_PORT_OVER_CURRENT field shall be cleared in each port individually.

When multiple ports share a power switch, setting PORT_POWER on any port in the group will cause the power to all ports in the group to turn on. It will not, however, cause the other ports in that group to leave the Powered-off state. When all the ports in a group are in the Powered-off state or the hub is not configured, the power to the ports is turned off.

If a hub implements both power switching and over-current, it is not necessary for the over-current groups to be the same as the power switching groups.

If an over-current condition occurs and power switches are present, then all power switches associated with an over-current protection circuit shall be turned off. If multiple over-current protection devices are associated with a single power switch, then that switch will be turned off when any of the over-current protection circuits indicates an over-current condition.

10.11 Hub Controller

The Hub Controller is logically organized as shown in Figure 10-18.

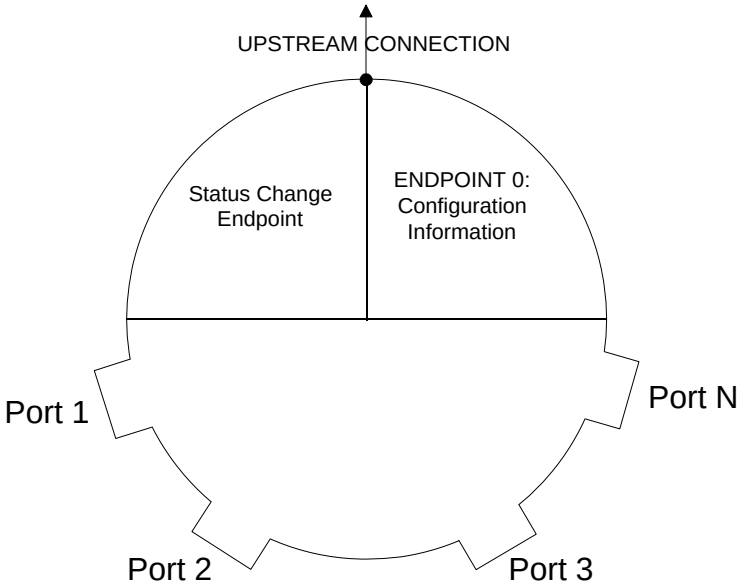


Figure 10-18. Example Hub Controller Organization

10.11.1 Endpoint Organization

The Hub Class defines one additional endpoint beyond the Default Control Pipe, which is required for all hubs: the Status Change endpoint. This endpoint has the maximum burst size set to one. The host system receives port and hub status change notifications through the Status Change endpoint. The Status Change endpoint is an interrupt endpoint. If no hub or port status change bits are set, then the hub returns an NRDY when the Status Change endpoint receives an IN (via and ACK TP) request. When a status change bit is set, the hub will send an ERDY TP to the host. The host will subsequently asks the Status Change endpoint for the data, which will indicate the entity (hub or port) with a change bit set. The USB System Software can use this data to determine which status registers to access in order to determine the exact cause of the status change interrupt.

10.11.2 Hub Information Architecture and Operation

Figure 10-19 shows how status, status change, and control information relate to device states. Hub descriptors and Hub/Port Status and Control are accessible through the Default Control Pipe. The Hub descriptors may be read at any time. When a hub detects a change on a port or when the hub changes its own state, the Status Change endpoint transfers data to the host in the form specified in Section 10.11.4.

Hub or port status change bits can be set because of hardware or Software events. When set, these bits remain set until cleared directly by the USB System Software through a ClearPortFeature() request or by a hub reset. While a change bit is set, the hub continues to report a status change when the Status Change endpoint is read until all change bits have been cleared by the USB System Software.

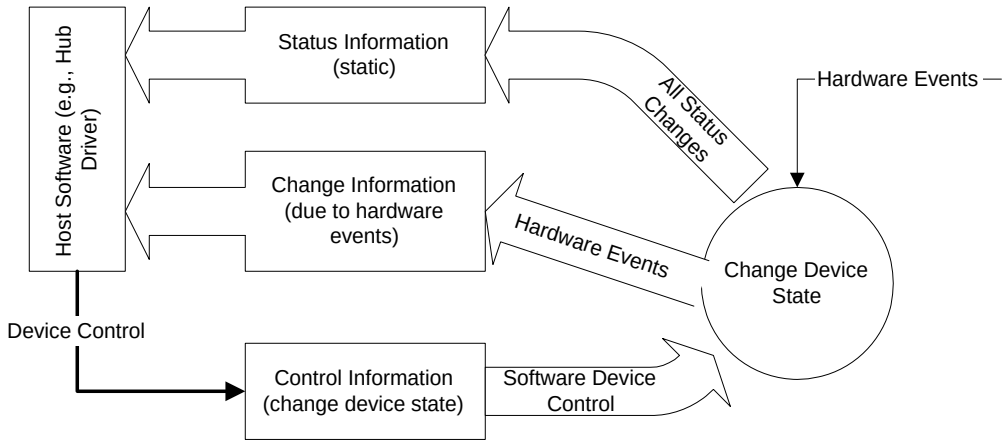


Figure 10-19. Relationship of Status, Status Change, and Control Information to Device States

The USB System Software uses the interrupt pipe associated with the Status Change endpoint to detect changes in hub and port status.

10.11.3 Port Change Information Processing

Hubs report a port's status through port commands on a per-port basis. The USB System Software acknowledges a port change by clearing the change state corresponding to the status change reported by the hub. The acknowledgment clears the change state for that port so future data transfers to the Status Change endpoint do not report the previous event. This allows the process to repeat for further changes (see Figure 10-20).

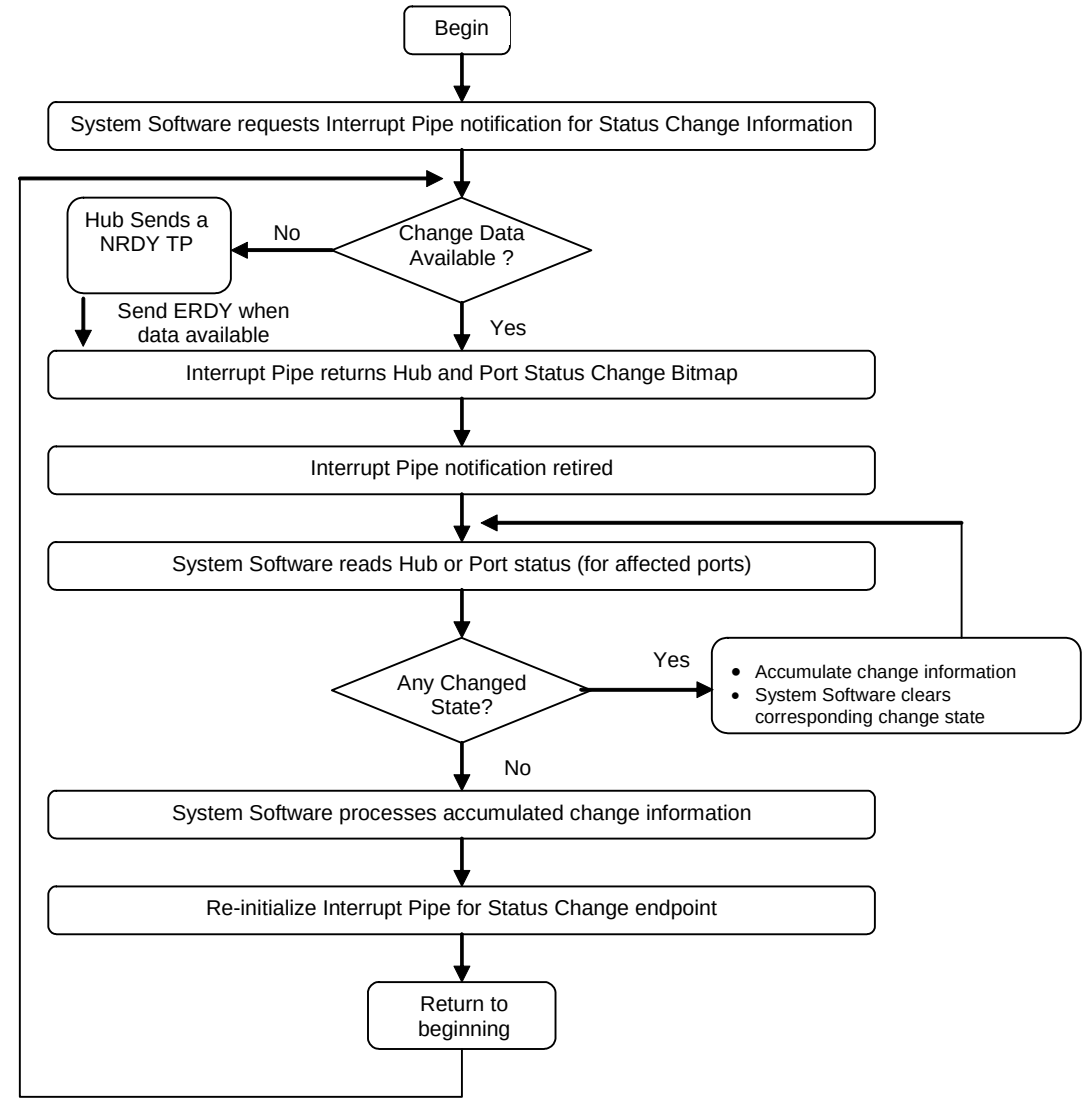


Figure 10-20. Port Status Handling Method

10.11.4 Hub and Port Status Change Bitmap

The Hub and Port Status Change Bitmap, shown in Figure 10-21, indicates whether the hub or a port has experienced a status change. This bitmap also indicates which port(s) has had a change in status. The hub returns this value on the Status Change endpoint. Hubs report this value in byte-increments. That is, if a hub has six ports, it returns a byte quantity, and reports a zero in the invalid port number field locations. The USB System Software is aware of the number of ports on a hub (this is reported in the hub descriptor) and decodes the Hub and Port Status Change Bitmap accordingly. The hub reports any changes in hub status in bit zero of the Hub and Port Status Change Bitmap.

The Hub and Port Status Change Bitmap size varies from a minimum size of one byte to a maximum of two bytes. Hubs report only as many bits as there are ports on the hub, subject to the byte-granularity requirement (i.e., round up to the nearest byte). A USB 3.0 hub may have no more than nMaxHubPorts.

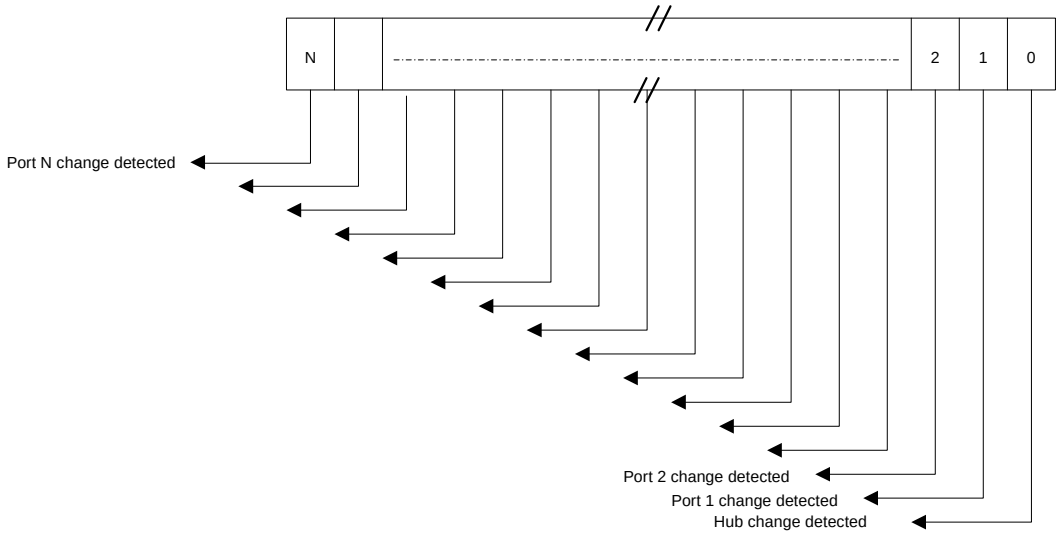


Figure 10-21. Hub and Port Status Change Bitmap

Any time any of the Status Changed bits are non-zero, an ERDY is returned (if an NRDY was previously sent) notifying the host that the Hub and Port Status Change Bitmap has changed. Figure 10-22 shows an example creation mechanism for hub and port change bits.

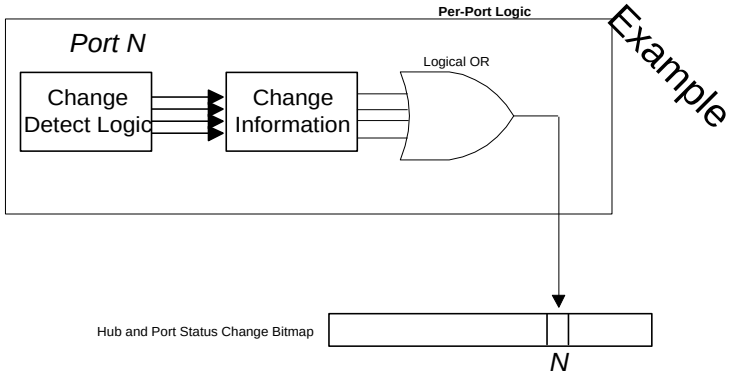


Figure 10-22. Example Hub and Port Change Bit Sampling

10.11.5 Over-current Reporting and Recovery

USB devices shall be designed to meet applicable safety standards. Usually, this will mean that a self-powered hub implements current limiting on its downstream facing ports. If an over-current condition occurs, it causes a status and state change in one or more ports. This change is reported to the USB System Software so that it can take corrective action.

A hub may be designed to report over-current as either a port or a hub event. The hub descriptor field *wHubCharacteristics* is used to indicate the reporting capabilities of a particular hub (see Section 10.13.2.1). The over-current status bit in the hub or port status field indicates the state of the over-current detection when the status is returned. The over-current status change bit in the Hub or Port Change field indicates if the over-current status has changed.

When a hub experiences an over-current condition, it shall place all affected ports in the Powered-off state. If a hub has per-port power switching and per-port current limiting, an over-current condition on one port may still cause the power on another port to fall below specified minimums. In this case, the affected port is placed in the Powered-off state and C_PORT_OVER_CURRENT is set for the port, but PORT_OVER_CURRENT is not set. If the hub has over-current detection on a hub basis, then an over-current condition on the hub will cause all ports to enter the Powered-off state. However, in this case, neither C_PORT_OVER_CURRENT nor PORT_OVER_CURRENT is set for the affected ports.

Host recovery actions for an over-current event should include the following:

1. Host gets change notification from hub with over-current event.
2. Host extracts appropriate hub or port change information (depending on the information in the change bitmap).
3. Host waits for over-current status bit to be cleared to 0.
4. Host cycles power to on for all of the necessary ports (e.g., issues a SetPortFeature(PORT_POWER) request for each port).
5. Host re-enumerates all affected ports.

10.11.6 Enumeration Handling

The hub device class commands are used to manipulate its downstream facing port state. When a device is attached, the device attach event is detected by the hub and reported on the status change interrupt endpoint. The host will accept the status change report and may request a SetPortFeature(PORT_RESET) on the port. The Get_Status(PORT) request invoked by the host will return a PORT_SUPER_SPEED indication that the downstream facing port is operating at SuperSpeed.

When the device is detached from the port, the port reports the status change through the status change endpoint. Then the process is ready to be repeated on the next device attach detect.

10.12 Hub Configuration

Hubs are configured through the standard USB device configuration commands. A hub that is not configured behaves like any other device that is not configured with respect to power requirements and addressing. If a hub implements power switching and does not support charging, no power is provided to the downstream facing ports while the hub is not configured. Refer to section 10.3.1.1 for details on power for a hub that supports charging applications. Configuring a hub enables the Status Change endpoint. Part of the configuration process is setting the hub depth which is used to compute an index (see Section 10.14.2.8) into the RouteString (see Section 8.9). The hub depth is used to derive the offset into the RouteString (in a TP or DP) that the hub shall use to route packets received on the hubs upstream port. The USB System Software may then issue commands to the hub to switch port power on and off at appropriate times.

The USB System Software examines hub descriptor information to determine the hub's characteristics. By examining the hub's characteristics, the USB System Software ensures that illegal power topologies are not allowed by not powering on the hub's ports if doing so would violate the USB power topology. The device status and configuration information can be used to determine whether the hub can be used in a given topology. Table 10-4 summarizes the information and how it can be used to determine the current power requirements of the hub.

Table 10-4. Hub Power Operating Mode Summary

Configuration Descriptor		Hub Device Status (Self Power)	Explanation
MaxPower	**bmAttributes (Self Powered)**		
0	0	N/A	N/A This is an illegal set of information.
0	1	0	N/A A device which is only self-powered, but does not have local power cannot connect to the bus and communicate.
0	1	1	Self-powered only hub and local power supply is good. Hub status also indicates local power good.. Hub functionality is valid anywhere depth restriction is not violated.
>0	0	N/A	Bus-powered only hub. Downstream facing ports may not be powered unless allowed in current topology. Hub device status reporting self-powered is meaningless if *bmAttributes.self-powered* is zero.

Configuration Descriptor		Hub Device Status (Self Power)	Explanation
MaxPower	bmAttributes (Self Powered)		
>0	1	0	This hub is capable of both self- and bus-powered operating modes. It is currently only available as a bus-powered hub.
> 0	1	1	This hub draws power from both the bus and its local power supply. It is currently available as a self-powered hub.

A self-powered hub has a local power supply, but may optionally draw one unit load from its upstream connection. This allows the interface to function when local power is not available (see Section 7.2.1.2). When local power is removed (either a hub-wide over-current condition or local supply is off), a hub of this type remains in the Configured state but transitions all ports (whether removable or non-removable) to the Powered-off state. While local power is off, all port status and change information read as zero and all SetPortFeature() requests are ignored (request is treated as a no-operation). The hub will use the Status Change endpoint to notify the USB System Software of the hub event (see Section 11.24.2.6 for details on hub status).

The *MaxPower* field in the configuration descriptor is used to report to the system the maximum power the hub will draw from VBUS when the configuration is selected. The external devices attaching to the hub will report their individual power requirements.

A compound device may power both the hub electronics and the permanently attached devices from VBUS. The entire load may be reported in the hubs' configuration descriptor with the permanently attached devices each reporting self-powered, with zero *MaxPower* in their respective configuration descriptors.

A bus powered hub shall be able to supply any power not used by the hub electronics or permanently attached devices for the selected configuration to the exposed downstream ports. The hub shall be able to provide the power with any split across the exposed downstream ports. (i.e., If the hub can provide 600 mA to two exposed downstream ports it must be able to provide 450 ma to one and 150 ma to the other, 300 ma to each, etc.)

Note: Software shall ensure that at least 150 mA is available for each exposed downstream port on a bus powered hub.

10.13 Descriptors

Hub descriptors are derived from the general USB device framework. Hub descriptors define a hub device and the ports on that hub. The host accesses hub descriptors through the hub's default pipe.

The USB specification (refer to Chapter 8) defines the following descriptors:

- Device Level Descriptors
- Configuration
- Interface
- Endpoint
- String (optional)

The hub class defines additional descriptors (refer to Section 10.13.2). In addition, vendor-specific descriptors are allowed in the USB device framework. Hubs support standard USB device commands as defined in Chapter 8.

A Hub is the only device that is allowed to function at both High Speed and Super Speed at the same time. This specification only defines the descriptors a hub shall report when it is operating in SuperSpeed mode.

Note that a SuperSpeed hub shall always support the Get Descriptor (BOS) (see Section 9.6.2) in both SuperSpeed and High Speed modes.

10.13.1 Standard Descriptors for Hub Class

The hub class pre-defines certain fields in standard USB descriptors. Other fields are either implementation-dependent or not applicable to this class.

A hub has a device descriptor with a bDeviceProtocol field set to three (3) and an interface descriptor with a bInterfaceProtocol field set to zero(0).

NOTE

For the descriptors and fields shown below, the bits in a field are organized in a little-endian fashion; that is, bit location 0 is the least significant bit and bit location 7 is the most significant bit of a byte value.

Hub Descriptors in SuperSpeed Mode

Device Descriptor (SuperSpeed information):

bLength	12H
bDescriptorType	1
bcdUSB	300H
bDeviceClass	HUB_CLASSCODE (09H)
bDeviceSubClass	0
bDeviceProtocol	3
bMaxPacketSize0	09H
bNumConfigurations	1

Configuration Descriptor (SuperSpeed information):

bLength	09H
bDescriptorType	2
wTotalLength	N
bNumInterfaces	1
bConfigurationValue	X
iConfiguration	Y
bmAttributes	Z
bMaxPower	The maximum amount of bus power the hub will consume in this configuration

Interface Descriptor:

bLength	09H
bDescriptorType	4
bInterfaceNumber	0
bAlternateSetting	0
bNumEndpoints	1
bInterfaceClass	HUB_CLASSCODE (09H)
bInterfaceSubClass	0
bInterfaceProtocol	0
iInterface	I

Endpoint Descriptor (for Status Change Endpoint):

bLength	07H
bDescriptorType	5
bEndpointAddress	Implementation-dependent; Bit 7: Direction = In(1)
bmAttributes	Transfer Type = Interrupt (00000011B)
wMaxPacketSize	1024
bInterval	FFH (Maximum allowable interval)

Endpoint Companion Descriptor (for Status Change Endpoint):

bLength	04H
bDescriptorType	18
bMaxBurst	1

The hub class driver retrieves a device configuration from the USB System Software using the GetDescriptor() device request. The only endpoint descriptor that is returned by the GetDescriptor() request is the Status Change endpoint descriptor.

10.13.2 Class-specific Descriptors

10.13.2.1 Hub Descriptor

Table 10-5 outlines the various fields contained by the hub descriptor.

Table 10-5. SuperSpeed Hub Descriptor

Offset	Field	Size	Description
0	*bDescLength*	1	Number of bytes in this descriptor, including this byte
1	*bDescriptorType*	1	Descriptor Type, value: 2AHfor SuperSpeed hub descriptor
2	*bNbrPorts*	1	Number of downstream facing ports that this hub supports. The maximum number of ports of ports a hub can support is 15.
3	*wHubCharacteristics*	2	D1...D0: Logical Power Switching Mode 00: Ganged power switching (all ports' power 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, without a breakdown of individual port over-current status. 01: Individual Port Over-current Protection. The hub reports over-current on a per-port basis. Each port has an over-current status. 1X: No Over-current Protection. This option is allowed only for bus-powered hubs that do not implement over-current protection. D15...D5: Reserved
5	*bPwrOn2PwrGood*	1	Time (in 2 ms intervals) from the time the power-on sequence begins on a port until power is good on that port. The USB System Software uses this value to determine how long to wait before accessing a powered-on port.
6	*bHubContrCurrent*	1	Maximum current requirements of the Hub Controller electronics when the hub is operating on both USB 2.0 and SuperSpeed expressed in units of aCurrentUnit. (i.e., 50 = 50*aCurrentUnit mA). Note that the encoding of this field is different than the encoding used in the USB 2.0 specification for USB 2.0 hubs. A USB 3.0 hub shall report the current requirements when it is only operating on USB 2.0 (not SuperSpeed) in the USB 2.0 hub descriptor.

Offset	Field	Size	Description
7	*bHubHdrDecLat*	1	Hub Packet Header Decode Latency. Worst case latency for hubs whose upstream link is in U0 to decode the header of a downstream flowing TP or DP packet and initiate a transition to U0 on the relevant downstream port. This field is used to calculate the total path exit latency through a hub. The following are permissible values: Value — Meaning 00H — Much less than 0.1 µs. 01H — 0.1 µs 02H — 0.2 µs 03H — 0.3 µs 04H — 0.4 µs 05H – FFH — Reserved
8	*wHubDelay*	2	This field defines the delay in nano-seconds a hub introduces on downstream flowing packets that it receives before forwarding them when both its upstream link and the downstream link on which it forwards the packet are in the U0 state and no Link Commands are in flight. The time is measured from receipt of the first framing symbol of the packet by the upstream port until the downstream port sends the first framing symbol of the packet.
10	*DeviceRemovable*	Variable, depending on number of ports on hub	Indicates if a port has a removable device attached. This field is reported on byte-granularity. Within a byte, if no port exists for a given location, the field representing the port characteristics returns 0. Bit value definition: 0B - Device is removable. 1B - Device is non-removable. This is a bitmap corresponding to the individual ports on the hub: Bit 0: Reserved for future use Bit 1: Port 1 Bit 2: Port 2 .... Bit *n*: Port *n* (implementation-dependent, up to a maximum of 15 ports).

10.14 Requests

10.14.1 Standard Requests

Hubs have tighter constraints on request processing timing than specified in Section 9.2.6 for standard devices because they are crucial to the "time to availability" of all devices attached to USB. The worst case request timing requirements are listed below (apply to both Standard and Hub Class requests):

- Completion time for requests with no data stage: 50 ms
- Completion times for standard requests with data stage(s):
  - Time from setup packet to first data stage: 50 ms
  - Time between each subsequent data stage: 50 ms
  - Time between last data stage and status stage: 50 ms

As hubs play such a crucial role in bus enumeration, it is recommended that hubs average response times be less than 5 ms for all requests.

Table 10-6 outlines the various standard device requests.

Table 10-6. Hub Responses to Standard Device Requests

bRequest	Hub Response
CLEAR_FEATURE	Standard
GET_CONFIGURATION	Standard
GET_DESCRIPTOR	Standard
GET_INTERFACE	Undefined. Hubs are allowed to support only one interface.
GET_STATUS	Standard
SET_ADDRESS	Standard
SET_CONFIGURATION	Standard
SET_DESCRIPTOR	Optional
SET_FEATURE	Standard
SET_INTERFACE	Undefined. Hubs are allowed to support only one interface.
SYNCH_FRAME	Undefined. Hubs are not allowed to have isochronous endpoints.

Optional requests that are not implemented shall return a STALL in the Data stage or Status stage of the request.

10.14.2 Class-specific Requests

The hub class defines requests to which hubs respond, as outlined in Table 10-7. Table 10-8 defines the hub class request codes. All requests in the table below except SetHubDescriptor() are mandatory.

Table 10-7. Hub Class Requests

Request	bmRequestType	bRequest	wValue	wIndex	wLength	Data
ClearHubFeature	00100000B	CLEAR_FEATURE	Feature Selector	Zero	Zero	None
ClearPortFeature	00100011B	CLEAR_FEATURE	Feature Selector	Port	Zero	None
GetHubDescriptor	10100000B	GET_DESCRIPTOR	Descriptor Type and Descriptor Index	Zero or Language ID	Descriptor Length	Descriptor
GetHubStatus	10100000B	GET_STATUS	Zero	Zero	Four	Hub Status and Change Status
GetPortStatus	10100011B	GET_STATUS	Zero	Port	Four	Port Status and Change Status
GetPortErrorCount	10000000B	GET_PORT_ERR_ COUNT	Zero	Port	Two	Number of Link Errors on this port
SetHubDescriptor	00100000B	SET_DESCRIPTOR	Descriptor Type and Descriptor Index	Zero or Language ID	Descriptor Length	Descriptor
SetHubFeature	00100000B	SET_FEATURE	Feature Selector	Zero	Zero	None
SetHubDepth	00100000B	SET_HUB_DEPTH	Hub Depth	Zero	Zero	None
SetPortFeature	00100011B	SET_FEATURE	Feature Selector	Selector, Timeout, Port	Zero	None

Table 10-8. Hub Class Request Codes

bRequest	Value
GET_STATUS	0
CLEAR_FEATURE	1
RESERVED (used in previous specifications for GET_STATE)	2
SET_FEATURE	3
Reserved for future use	*4-5*
GET_DESCRIPTOR	6
SET_DESCRIPTOR	7
RESERVED (used in USB 2.0 specification)	8-11
SET_HUB_DEPTH	12
GET_PORT_ERR_COUNT	13

Table 10-9 gives the valid feature selectors for the hub class. See Section 10.14.2.4 and Section 10.14.2.6 for a description of the features.

Table 10-9. Hub Class Feature Selectors

	Recipient	Value
C_HUB_LOCAL_POWER	Hub	0
C_HUB_OVER_CURRENT	Hub	1
PORT_CONNECTION	Port	0
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
PORT_TEST	Port	21
PORT_U1_TIMEOUT	Port	23
PORT_U2_TIMEOUT	Port	24
C_PORT_LINK_STATE	Port	25
C_PORT_CONFIG_ERROR	Port	26
PORT_REMOTE_WAKE_MASK	Port	27
BH_PORT_RESET	Port	28
C_BH_PORT_RESET	Port	29

10.14.2.1 Clear Hub Feature

This request resets a value reported in the hub status.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00100000B	CLEAR_FEATURE	Feature Selector	Zero	Zero	None

Clearing a feature disables that feature; refer to Table 10-9 for the feature selector definitions that apply to the hub as a recipient. If the feature selector is associated with a status change, clearing that status change acknowledges the change. This request format is used to clear either the c_hub_local_power or c_hub_over_current features.

It is a Request Error if *wValue* is not a feature selector listed in Table 10-9 or if *wIndex* or *wLength* are not as specified above.

If the hub is not configured, the hub's response to this request is undefined.

10.14.2.2 Clear Port Feature

This request resets a value reported in the port status.

bmRequestType	bRequest	wValue	wIndex		wLength	Data
00100011B	CLEAR_FEATURE	Feature Selector	Selector	Port	Zero	None

The port number shall be a valid port number for that hub, greater than zero. The port field is located in bits 7..0 of the *wIndex* field.

Clearing a feature disables that feature or starts a process associated with the feature; refer to Table 10-9 for the feature selector definitions. If the feature selector is associated with a status change, clearing that status change acknowledges the change. This request format is used to clear the following features:

- PORT_POWER
- C_PORT_CONNECTION
- C_PORT_RESET
- C_PORT_OVER_CURRENT
- C_PORT_LINK_STATUS
- C_PORT_CONFIG_ERROR
- C_BH_PORT_RESET

Clearing the PORT_POWER feature causes the port to be placed in the Powered-off state and may, subject to the constraints due to the hub’s method of power switching, result in power being removed from the port. When in the power off state, the only requests that are valid when this port is the recipient are Get Port Status (see Section 10.14.2.6) and Set Port Feature (PORT_POWER) (see Section 10.14.2.10).

Clearing the FORCE_LINKPM_ACCEPT feature causes the port to de-assert the Force_LinkPM_Accept bit in Set Link Function LMPs. If the Force_LinkPM_Accept bit is not asserted on the port, the hub shall treat this request as a functional no-operation.

It is a Request Error if *wValue* is not a feature selector listed in Table 10-9, if *wIndex* specifies a port that does not exist, or if *wLength* is not as specified above. It is not an error for this request to try to clear a feature that is already cleared (the hub shall treat this as a functional no-operation).

If the hub is not configured, the hub's response to this request is undefined.

10.14.2.3 Get Hub Descriptor

This request returns the hub descriptor.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
10100000B	GET_DESCRIPTOR	Descriptor Type and Descriptor Index	Zero	Descriptor Length	Descriptor

The GetDescriptor() request for the hub class descriptor follows the same usage model as that of the standard GetDescriptor() request (refer to Chapter 9). The standard hub descriptor is denoted by using the value *bDescriptorType* defined in Section 10.13.2.1. All hubs are required to implement one hub descriptor, with descriptor index zero.

If *wLength* is larger than the actual length of the descriptor, then only the actual length is returned. If *wLength* is less than the actual length of the descriptor, then only the first *wLength* bytes of the descriptor are returned; this is not considered an error even if *wLength* is zero.

It is a Request Error if *wValue* or *wIndex* are other than as specified above.

If the hub is not configured, the hub's response to this request is undefined.

10.14.2.4 Get Hub Status

This request returns the current hub status and the states that have changed since the previous acknowledgment.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
10100000B	GET_STATUS	Zero	Zero	Four	Hub Status and Change Status

The first word of data contains wHubStatus (refer to Table 10-10). The second word of data contains wHubChange (refer to Table 10-11).

It is a Request Error if *wValue*, *wIndex*, or *wLength* are other than as specified above.

If the hub is not configured, the hub's response to this request is undefined.

Table 10-10. Hub Status Field, *wHubStatus*

Bit	Description
0	**Local Power Source:** This is the source of the local power supply. This field indicates whether hub power (for other than the SIE) is being provided by an external source or from the USB. This field allows the USB System Software to determine the amount of power available from a hub to downstream devices. 0 = Local power supply good 1 = Local power supply lost (inactive)
1	**Over-current:** If the hub supports over-current reporting on a hub basis, this field indicates that the sum of all the ports' current has exceeded the specified maximum and all ports have been placed in the Powered-off state. If the hub reports over-current on a per-port basis or has no over-current detection capabilities, this field is always zero. The hub should only report over-current if it is physically unable to meet the sum of all ports' current draws. For more details on over-current protection, see the USB 2.0 Specification Section 7.2.1.2.1. 0 = No over-current condition currently exists. 1 = A hub over-current condition exists.
2-15	**Reserved** These bits return 0 when read.

There are no defined feature selector values for these status bits and they can neither be set nor cleared by the USB System Software.

Table 10-11. Hub Change Field, *wHubChange*

Bit	Description
0	**Local Power Status Change:** (C_HUB_LOCAL_POWER) This field indicates that a change has occurred in the hub's Local Power Source field in *wHubStatus*. This field is initialized to zero when the hub receives a bus reset. 0 = No change has occurred to Local Power Status. 1 = Local Power Status has changed.
1	**Over-Current Change**: (C_HUB_OVER_CURRENT) This field indicates if a change has occurred in the Over-Current field in *wHubStatus*. This field is initialized to zero when the hub receives a bus reset. 0 = No change has occurred to the Over-Current Status. 1 = Over-Current Status has changed.
2-15	**Reserved** These bits return 0 when read.

Hubs may allow setting of these change bits with SetHubFeature() requests in order to support diagnostics. If the hub does not support setting of these bits, it should either treat the SetHubFeature() request as a Request Error or as a functional no-operation. When set, these bits may be cleared by a ClearHubFeature() request. A request to set a feature that is already set or to clear a feature that is already clear has no effect and the hub will not fail the request.

10.14.2.5 Get Port Error Count

This request returns the number of link errors detected by the hub on the port indicated by wIndex. This value is reset to zero whenever the device goes through a Reset (see Section 7.3) or at power up.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
10000000B	GET_PORT_ERR_COUNT	Zero	Port	Two	Number of Link Errors

The port number shall be a valid port number for that hub, greater than zero.

It is a Request Error if *wValue* or *wLength* are other than as specified above or if *wIndex* specifies a port that does not exist.

If the hub is not configured, the behavior of the hub in response to this request is undefined.

10.14.2.6 Get Port Status

This request returns the current port status and the current value of the port status change bits.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
10100011B	GET_STATUS	Zero	Port	Four	Port Status and Change Status

The port number shall be a valid port number for that hub, greater than zero.

The first word of data contains wPortStatus (refer to Table 10-12). The second word of data contains wPortChange (refer to Table 10-11).

The bit locations in the wPortStatus and wPortChange fields correspond in a one-to-one fashion where applicable.

It is a Request Error if *wValue* or *wLength* are other than as specified above or if *wIndex* specifies a port that does not exist.

If the hub is not configured, the behavior of the hub in response to this request is undefined.

10.14.2.6.1 Port Status Bits

Table 10-12. Port Status Field, *wPortStatus*

Bit	Description
0	Current Connect Status: (PORT_CONNECTION) This field reflects whether or not a device is currently connected to this port. **Value** **Meaning** 0 No device is present 1 A device is present on this port
1	Port Enabled/Disabled: This field indicates whether the port is enabled. Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by the USB System Software. **Value** **Meaning** 0 Port is disabled 1 Port is enabled
2	Reserved This bit returns 0 when read
3	Over-current: (PORT_OVER_CURRENT) If the hub reports over-current conditions on a per-port basis, this field will indicate that the current drain on the port exceeds the specified maximum. 0 = No over-current condition exists on this port. 1 = An over-current condition exists on this port.
4	Reset: (PORT_RESET) This field is set when the host wishes to reset the attached device. It remains set until the reset signaling is turned off by the hub. **Value** **Meaning** 0 Reset signaling not asserted 1 Reset signaling asserted
5-8	Port Link State: (PORT_LINK_STATE) This field reflects the current state of the link attached to this port. A new state is not reflected until the link state transition to that state is complete. **Value** **Meaning** 0x00 Link is in the U0 State 0x01 Link is in the U1 State 0x02 Link is in the U2 State 0x03 Link is in the U3 State 0x04 Link is in the Disabled State 0x05 Link is in the RxDetect State 0x06 Link is in the Inactive State 0x07 Link is in the Polling State 0x08 Link is in the Recovery State 0x09 Link is in the Hot Reset State 0xA Link is in the Compliance Mode State 0xB Link is in the Loopback State 0xC-0xF Reserved

Bit	Description
9	Port Power: (PORT_POWER) This field reflects a port's logical, power control state. Because hubs can implement different methods of port power switching, this field may or may not represent whether power is applied to the port. The device descriptor reports the type of power switching implemented by the hub. **Value** **Meaning** 0 This port is in the Powered-off state 1 This port is not in the Powered-off state
10-12	Negotiated Speed of the SuperSpeed Device Attached to this port: (PORT_SPEED) This field is valid only if a device is attached. **Value** **Description** 0 5Gbs 1 Reserved 2 Reserved 3 Reserved 4 Reserved 5 Reserved 6 Reserved 7 Reserved
13-14	Reserved These bits return 0 when read.
15	Port Test Mode : (PORT_TEST) This field reflects the status of the port's test mode. Software uses the SetPortFeature() and ClearPortFeature() requests to manipulate the port test mode. **Value** **Meaning** 0 This port is not in the Port Test Mode 1 This port is in Port Test Mode

PORT_CONNECTION

When the Port Power bit is one, this bit indicates whether or not a device is attached and has successfully completed link training. This field reads as one when a device is attached and has successfully completed link training; it reads as zero otherwise. This bit is reset to zero when the port is in the Powered-off state or the Disconnected states. It is set to one when the port is in the Powered state, a device attach is detected, and the downstream port successfully completed link training and transitions to the DSPORT..enabled state.

SetPortFeature(PORT_CONNECTION) and ClearPortFeature(PORT_CONNECTION) requests shall not be used by the USB System Software and shall be treated as no-operation requests by hubs.

PORT_ENABLE

This bit is set when the port is allowed to forward packet data upstream or downstream. When the downstream port moves to the DSPORT.Enabled state, this bit will be set.

This bit may be cleared as the result of any of the following:

- The port being in the DSPORT.Powered-off state
- Port_Error detection
- Disconnect detection
- When the port enters the DSPORT.Resetting state as a result of receiving the SetPortFeature(PORT_RESET) request

PORT_OVER_CURRENT

This bit is set to one while an over-current condition exists on the port. This bit is cleared when an over-current condition does not exist on the port.

If the voltage on this port is affected by an over-current condition on another port, this bit is set and remains set until the over-current condition on the affecting port is removed. When the over-current condition on the affecting port is removed, this bit is reset to zero.

Over-current protection is required on self-powered hubs (it is optional on bus-powered hubs) as outlined in Section 10.10.

The SetPortFeature(PORT_OVER_CURRENT) and ClearPortFeature(PORT_OVER_CURRENT) requests shall not be used by the USB System Software and may be treated as no-operation requests by hubs.

PORT_RESET

This bit is set while the port is in the Resetting state. A SetPortFeature(PORT_RESET) request will initiate the Resetting state if the conditions in Section 10.3.1.6 are met. This bit is set to zero while the port is in the Powered-off state.

The ClearPortFeature(PORT_RESET) request shall not be used by the USB System Software and may be treated as a no-operation request by hubs.

PORT_LINK_STATE

This field reflects the current state of the link.

The SetPortFeature(PORT_LINK_STATE) request may be issued by the USB System Software at any time but will have an effect only as specified in Section 10.14.2.10.

The ClearPortFeature(PORT_LINK_STATE) requests shall not be used by the USB System Software and may be treated as no-operation requests by hubs.

PORT_POWER

This bit reflects the current power state of a port. This bit is implemented on all ports whether or not actual port power switching devices are present.

While this bit is zero, the port is in the Powered-off state. Similarly, anything that causes this port to go to the Power-off state will cause this bit to be set to zero.

A SetPortFeature(PORT_POWER) will set this bit to one unless both C_HUB_LOCAL_POWER and Local Power Status (in *wHubStatus*) are set to one in which case the request is treated as a functional no-operation.

This bit may be cleared under the following circumstances:

- Hub receives a ClearPortFeature(PORT_POWER).
- An over-current condition exists on the port.
- An over-current condition on another port causes the power on this port to be shut off.

The SetPortFeature(PORT_POWER) and ClearPortFeature(PORT_POWER) requests may be issued by the USB System Software whenever the port is not in the DSPORT.Powered-off state.

PORT_SPEED

This field has meaning only when the PORT_ENABLE bit is set. A value of zero in this field indicates that the SuperSpeed device attached to this port is operating at 5Gbs. All other values in this field are reserved.

This field can only be read by USB System Software.

PORT_TEST

When the Port Test Mode bit is set to a one (1B), the port is in test mode. The specific test mode is specified in the SetPortFeature(PORT_TEST) request by the test selector. The hub provides no standard mechanism to report the specific test mode; therefore, system software must track which test selector was used. See Section 10.14.2.9 for more information about using SetPortFeature to control test mode.

This field may only be set as a result of a SetPortFeature(PORT_TEST) request. A port PORT_TEST request is only valid to a port that is in the Disabled, Disconnected, or Suspended states. If the port is not in one of these states, the hub shall respond with a request error.

This field may be cleared as a result of resetting the hub.

10.14.2.6.2 Port Status Change Bits

Port status change bits are used to indicate changes in port status bits that are not the direct result of requests. Port status change bits can be cleared with a ClearPortFeature() request or by a hub reset. Hubs may allow setting of the status change bits with a SetPortFeature() request for diagnostic purposes. If a hub does not support setting of the status change bits, it may either treat the request as a Request Error or as a functional no-operation. Table 10-13 describes the various bits in the *wPortChange* field.

Table 10-13. Port Change Field, *wPortChange*

Bit	Description
0	Connect Status Change: (C_PORT_CONNECTION) Indicates a change has occurred in the port's Current Connect Status. The hub device sets this field as described in Section 10.3.1. **Value** **Meaning** 0 No change has occurred to Current Connect status 1 Current Connect status has changed
1-2	Reserved These bits return 0 when read.
3	Over-Current Indicator Change: (C_PORT_OVER_CURRENT) This field applies only to hubs that report over-current conditions on a per-port basis (as reported in the hub descriptor). **Value** **Meaning** 0 No change has occurred to Over-Current Indicator 1 Over-Current Indicator has changed If the hub does not report over-current on a per-port basis, then this field is always zero.
4	Reset Change: (C_PORT_RESET) This field is set when reset processing on this port is complete. **Value** **Meaning** 0 No change 1 Reset complete
5	BH Reset Change: (C_BH_PORT_RESET) This field is set when a warm reset processing on this port is complete. **Value** **Meaning** 0 No change 1 Reset complete
6	Port Link State Change: (C_PORT_LINK_STATE) This field is set when the port link status has changed. **Value** **Meaning** 0 No change 1 Link Status has changed
7	Port Config Error: (C_PORT_CONFIG_ERROR) This field is set when the port fails to config its link partner. **Value** **Meaning** 0 Port Link Configuration was successful 1 Port Link Configuration was unsuccessful
8-15	Reserved These bits return 0 when read.

C_PORT_CONNECTION

This bit is set when the PORT_CONNECTION bit changes because the hub port downstream port successfully completed training and transitioned to the DSPORT.enabled state. This bit will be cleared to zero by a ClearPortFeature(C_PORT_CONNECTION) request or while the port is in the Powered-off state.

C_PORT_OVER_CURRENT

This bit is set when the PORT_OVER_CURRENT bit changes from zero to one or from one to zero. This bit is also set if the port is placed in the Powered-off state due to an over-current condition on another port.

This bit will be cleared when the port is in the DSPORT.Powered-off state or by a ClearPortFeature(C_PORT_OVER_CURRENT) request.

C_PORT_RESET

This bit is set when the port transitions from the Resetting state (or, if present, the Speed_eval state) to the Enabled state.

This bit will be cleared by a ClearPortFeature(C_PORT_RESET) request, or while the port is in the Powered-off state.

C_PORT_LINK_STATE

This bit is set when the port's Link completes a transition from the U3 state to the U0 as a result of a SetPortFeature(Port_Link_State) request or completes a transition from any of the U states to SS.disabled, RX.Detect, or SS.Inactive. This bit is not set due to transitions from U3 to U0 as a result of remote wakeup signaling received on a downstream facing port.

This bit will be cleared by a ClearPortFeature(C_PORT_LINK_STATE) request, or while the port is in the Powered-off state.

C_PORT_CONFIG_ERROR

This bit is set if the Link connected to the port could not be successfully configured, e.g. if two downstream only capable ports are connected to each other or if link configuration could not be completed. In addition, the port must transition to the Disabled state when this occurs.

This bit will be cleared by a ClearPortFeature(C_PORT_CONFIG_ERROR) request, or while the port is in the Powered-off state.

10.14.2.7 Set Hub Descriptor

This request overwrites the hub descriptor.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00100000B	SET_DESCRIPTOR	Descriptor Type and Descriptor Index	Zero	Descriptor Length	Descriptor

The SetDescriptor request for the hub class descriptor follows the same usage model as that of the standard SetDescriptor request (refer to the framework chapter). The standard hub descriptor is denoted by using the value *bDescriptorType* defined in Section 10.13.2.1. All hubs are required to implement one hub descriptor with descriptor index zero.

This request is optional. This request writes data to a class-specific descriptor. The host provides the data that is to be transferred to the hub during the data transfer phase of the control transaction. This request writes the entire hub descriptor at once.

Hubs shall buffer all the bytes received from this request to ensure that the entire descriptor has been successfully transmitted from the host. Upon successful completion of the bus transfer, the hub updates the contents of the specified descriptor.

It is a Request Error if *wIndex* is not zero or if *wLength* does not match the amount of data sent by the host. Hubs that do not support this request respond with a STALL during the Data stage of the request.

If the hub is not configured, the hub's response to this request is undefined.

10.14.2.8 Set Hub Feature

This request sets a value reported in the hub status.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00100000B	SET_FEATURE	Feature Selector	Zero	Zero	None

Setting a feature enables that feature; refer to Table 10-9 for the feature selector definitions that apply to the hub as recipient. Status changes may not be acknowledged using this request.

It is a Request Error if *wValue* is not a feature selector listed in Table 10-9 or if *wIndex* or *wLength* are not as specified above.

If the hub is not configured, the hub's response to this request is undefined.

10.14.2.9 Set Hub Depth

This request sets the value that the hub uses to determine the index into the Route String Index for the hub.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00100000B	SET_HUB_DEPTH	Hub Depth	Zero	Zero	None

wValue has the value of the Hub Depth. The Hub Depth left shifted by two is the offset into the Route String that identifies the lsb of the Route String Port Field for the hub.

It is a Request Error if *wValue* is greater than 5 or if *wIndex* or *wLength* are not as specified above.

If the hub is not configured, the hub's response to this request is undefined.

10.14.2.10 Set Port Feature

This request sets a value reported in the port status.

bmRequestType	bRequest	wValue	wIndex		wLength	Data
00100011B	SET_ FEATURE	Feature Selector	Selector or Timeout Value or Remote Wake Mask	Port	Zero	None

The port number shall be a valid port number for that hub, greater than zero. The port number is in the least significant byte (bits 7..0) of the *wIndex* field. The most significant byte of *wIndex* is zero, except when the feature selector is PORT_TEST or PORT_U1_TIMEOUT or PORT_U2_TIMEOUT or PORT_LINK_STATE or PORT_REMOTE_WAKE_MASK.

Setting a feature enables that feature or starts a process associated with that feature; see Table 10-9 for the feature selector definitions that apply to a port as a recipient. Status change may not be acknowledged using this request. Features that can be set with this request are:

- PORT_RESET
- PORT_POWER
- PORT_TEST
- PORT_U1_TIMEOUT
- PORT_U2_TIMEOUT
- PORT_LINK_STATE
- PORT_REMOTE_WAKE_MASK
- BH_PORT_RESET

When the feature selector is PORT_U1_TIMEOUT, the most significant byte (bits 15..8) of the *wIndex* field specifies the Timeout value for the U1 inactivity timer. The port's link shall send an LMP to its link partner with the specified timeout value after receiving a Set Port Feature request with the PORT_U1_TIMEOUT feature selector. Refer to Section 10.4.2.1 for a detailed description of how the U1 inactivity timer value is used.

The following are permissible values:

Table 10-14. U1 Timeout Value Encoding

Value	Description
00H	Zero (Default)
01H	1 µs
02H	2 µs
03H	3 µs
…	…
7FH	127 µs
80H-FEH	Reserved
FFH	Infinite

When the feature selector is PORT_U2_TIMEOUT, the most significant byte (bits 15..8) of the *wIndex* field specifies the Timeout value for the U2 inactivity timer. The port's link shall send an LMP to its link partner with the specified timeout value after receiving a Set Port Feature request with the PORT_U2_TIMEOUT feature selector. Refer to Section 10.4.2.1 for a detailed description of how the U2 inactivity timer value is used.

The following are permissible values:

Table 10-15. U2 Timeout Value Encoding

Value	Description
00H	Zero (Default)
01H	256 µs
02H	512 µs
03H	768 µs
…	…
FEH	65.024 ms
FFH	Infinite

When the feature selector is PORT_LINK_STATE, the most significant byte (bits 15..8) of the *wIndex* field specifies the U State the host software wants to put the link connected to the port into This request is only valid when the PORT_ENABLE bit is set and the PORT_LINK_STATE is not set to Disabled, RxDetect or Inactive.

- If the value is 0 then the hub shall transition the link to a U0 state from any of the U states.
- If the value is 1 then host software wants to transition the link to the U1 State. The hub shall attempt to transition the link to U1 from U0. If the link is in any state other than U0 when a request is received with a value of 1 the behavior is undefined.
- If the value is 2 then the host software wants to transition the link to the U2 State. The hub shall attempt to transition the link to U2 from U0. If the link is in any state other than U0 when a request is received with a value of 1 the behavior is undefined.
- If the value is 3 then host software wants to selectively suspend the device connected to this port. The hub shall attempt to transition the link to U3 from any of the other U states using allowed link state transitions. While this state is active, the hub does not propagate downstream-directed traffic to this port, but the hub will respond to resume signaling from the port.
- The hub shall respond with a Request Error if it sees any other value in the upper byte of the *wIndex* field.

When the feature selector is PORT_TEST, the most significant byte (bits 15..8) of the *wIndex* field is the selector identifying the specific test mode.

Table 10-16 lists the test selector definitions. Refer to Section TBD <This is not currently listed in the link chapter>in the Link Chapter for definitions of each test mode. Test mode of a downstream facing port can only be used in a well defined sequence of hub states. This sequence is defined as follows:

1. All enabled downstream facing ports of the hub containing the port to be tested shall be (selectively) suspended via the SetPortFeature(PORT_LINK_STATE) U3 request. Each downstream facing port of the hub shall be in the disabled, disconnected, or suspended state
2. A SetPortFeature(PORT_TEST) request shall be issued to the downstream facing port to be tested. Only a single downstream facing port can be in test_mode at a time. The transition to test mode shall be complete no later than 3 ms after the completion of the status stage of the request.
3. The downstream facing port under test can now be tested.
4. During test_mode, a port disconnect or resume status change on one of the suspended ports (not including the port under test) shall cause a status change (C_PORT_CONNECTION or C_PORT_LINK_STATE) report (See Section 10.11.3 and 10.14.2.6.2) from the hub. Note: Other status changes may or may not be supported in a hub with a downstream facing port in test mode. The reporting of these status changes can allow a test application to restore normal operation of a root hub without requiring a non-USB keyboard or mouse for user input. For example. a USB device attached to the root hub can be disconnected to notify the test application to restore normal root hub operation.
5. During test_mode, the state of the hub downstream facing ports shall not be changed by the host (i.e., hub class requests other than the Get_Port_Status() request shall not be issued by the host). Note: The hub shall also be reset before a SetPortFeature(PORT_TEST) can be used to place the port into another test mode.
6. After the test is completed, the hub (with the port under test) shall be reset by the host or user. This shall be accomplished by manipulating the port of the parent hub to which the hub under test is attached. This manipulation can consist of one of the following:
   a. Issuing a SetPortFeature(PORT_RESET) to the port of the parent hub to which the hub under test is attached.
   b. Issuing a ClearPortFeature(PORT_POWER) and SetPortFeature(PORT_POWER) to cycle power of a parent hub that supports per port power control.
   c. Disconnecting and re-connecting the hub under test from its parent hub port.
   d. For a root hub under test, a reset of the Host Controller may be required as there is no parent hub of the root hub.
7. Behavior of the hub under test and its downstream facing ports is undefined if these requirements are not met.

Table 10-16. Test Mode Selector Codes

Bit	Test Mode Description
0-2	**Value** **Meaning** 0 Reserved 1 Enter to Loopback 2 Enter Compliance 3 Vendor Specific 4-7 Reserved
3-7	These 5 bits specify any additional information about the above test mode. These bits are currently only valid when in the *Enter Compliance* Test Mode. **Bit** **Meaning** 3 Add SKPs to Compliance Pattern (Compliance SOS) 4 Send Compliance Pattern 2 5 Send Vendor Specific Compliance Pattern 6-7 Reserved. Must be set to zero

When the feature selector is PORT_REMOTE_WAKE_MASK, the most significant byte (bits 15..8) of the *wIndex* field specifies the conditions that would cause the hub to signal a remote wake event on its upstream port. The encoding for the port remote wake mask is given below:

Table 10-17. Downstream Port Remote Wake Mask Encoding

Bit	Description
0	**Conn_RWEnable** **Value** **Meaning** 0 The hub is disabled from signaling a remote wakeup due to a connect event on this port; connect events that occur during suspend must still be detected and reported after the resume process has completed (due to some other event) as a C_PORT_CONNECTION port status change. 1 The hub is enabled to signal a remote wakeup due to a connect event on the port.
1	**Disconn_RWEnable** **Value** **Meaning** 0 The hub is disabled from signaling a remote wakeup due to a disconnect event on this port; disconnect events that occur during suspend must still be detected and reported after the resume process has completed (due to some other event) as a C_PORT_CONNECTION port status change. 1 The hub is enabled to signal a remote wakeup due to a disconnect event on the port.

Bit	Description
2	**OC_RWEnable** **Value** **Meaning** 0 The hub is disabled from signaling a remote wakeup due to an over-current event on this port; over-current events that occur during suspend must still be detected and reported after the resume process has completed (due to some other event) as a C_PORT_OVER_CURRENT port status change. Note that a hub that does not support per-port overcurrent detection/reporting will signal remote-wakeup for an over-current event unless all ports have OC-RWEnable set to 0. 1 The hub is enabled to signal a remote wakeup due to an over-current event on the port.
3-7	These bits are reserved and must be set to zero.

Note that after power on or after the hub is reset, the remote wake mask is set to zero (i.e. the mask is enabled).

The hub shall meet the following requirements:

- If the port is in the Powered-off state, the hub shall treat a SetPortFeature(PORT_RESET) request as a functional no-operation.
- If the port is not in the Enabled state, the hub shall treat a SetPortFeature(PORT_LINK_STATE) U3 request as a functional no-operation.
- If the port is not in the Enabled state, the hub shall treat a SetPortFeature(PORT_U1_TIMEOUT) request as a functional no-operation.
- If the port is not in the Enabled state, the hub shall treat a SetPortFeature(PORT_U2_TIMEOUT) request as a functional no-operation.
- If the port is not in the Powered-off state, the hub shall treat a SetPortFeature(PORT_POWER) request as a functional no-operation.
- If the port is not in the Enabled state, the hub shall treat a SetPortFeature(FORCE_LINKPM_ACCEPT) request as a functional no-operation.

When the feature selector is BH_PORT_RESET, the hub shall initiate a warm reset (see Section 7.3) on the port that is identified by this command. The state of the port after this reset shall be the same as the state after a SetPortFeature(PORT_RESET). On completion of a BH_PORT_RESET, the hub shall set the C_BH_PORT_RESET field to one in the PortStatus for this port.

It is a Request Error if *wValue* is not a feature selector listed in Table 10-9, if *wIndex* specifies a port that does not exist, or if *wLength* is not as specified above.

If the hub is not configured, the hub's response to this request is undefined.

10.15 Host Root (Downstream) Ports

The root ports of a USB 3.0 host have similar functional requirements to the downstream ports of a USB 3.0 hub. This section summarizes which requirements also apply to the root port of a host and identifies any additional or different requirements.

A host root port shall follow the requirements for a downstream facing hub port in section 10.2 except for Section 10.2.3.

A host root port shall follow the requirements for a downstream facing hub port in section 10.3 with the following exceptions and additions:

- None of the transitions and/or transition conditions based on the state of the upstream hub port apply to a root port.
- A host shall have control mechanisms in the host interface that allow software to achieve equivalent behavior to downstream hub port behavior in response to SetPortFeature or ClearPortFeature requests documented in Section 10.3.
- A host shall implement port status bits consistent with the downstream port state descriptions in Section 10.3.
- If a host supports a USB 2.0 only mode the host is required to be able to correlate each port in the USB2.0 only mode with any SuperSpeed mode port that shares the same physical connector. Note this is similar to the requirement for USB 3.0 hubs in Section 10.3.3.

A host root port shall follow the requirements for a downstream facing hub port in Section 10.4 with the same general exceptions already noted in this section.

A host shall implement port status bits through the host interface that are consistent with all port status bit definitions in this chapter.

A host shall have mechanisms to achieve equivalent control over its root ports as provided by the SetPortFeature, ClearPortFeature and GetPortStatus requests documented in this chapter.

10.16 Device Upstream Ports

The upstream port of a USB 3.0 device has similar functional requirements to the upstream port of a USB 3.0 hub. This section summarizes which requirements also apply to the upstream port of a device and identifies any additional or different requirements.

A device shall follow the requirements for an upstream facing hub port in Section 10.5 with the following exceptions and additions:

- A device shall not attempt to connect on the USB 2.0 interface when the port is in the USPORT.Connected state.
- A device shall not attempt to connect on the USB 2.0 interface unless the port has entered the USPORT.Powered-off state and VBUS is still present as shown in Figure 10-11.
- If a device is connected on the USB 2.0 interface and it receives a USB 2.0 bus reset the device shall enter the USPORT.Powered-On state within tCheckSuperSpeedOnReset time.

- If a device is connected on the USB 2.0 interface and the SuperSpeed port enters the USPORT.Training state the device shall disconnect on the USB 2.0 interface within tUSB2SwitchDisconnect time.

10.16.1 Peripheral Device Upstream Ports

The upstream port of a USB 3.0 device has similar functional requirements to the upstream port of a USB 3.0 hub. This section summarizes which requirements also apply to the upstream port of a device and identifies any additional or different requirements.

A device shall follow the requirements for an upstream facing hub port in Section 10.5 with the following exceptions and additions:

- A device shall not attempt to connect on the USB 2.0 interface when the port is in the USPORT.Connected state.
- A device shall not attempt to connect on the USB 2.0 interface unless the port has entered the USPORT.Powered-off state and VBUS is still present as shown in Figure 10-11.
- If a device is connected on the USB 2.0 interface and it receives a USB 2.0 bus reset the device shall enter the USPORT.Powered-On state within tCheckSuperSpeedOnReset time.
- If a device is connected on the USB 2.0 interface and the SuperSpeed port enters the USPORT.Training state the device shall disconnect on the USB 2.0 interface within tUSB2SwitchDisconnect time.

10.16.1.1 Peripheral Device Connect State Machine

The following sections provide a functional description of a state machine that exhibits correct peripheral device behavior for when to connect on Superspeed or USB 2.0. Figure 10-23 is an illustration of the peripheral device connect state machine. Each of the states is described in Section 10.16.2.

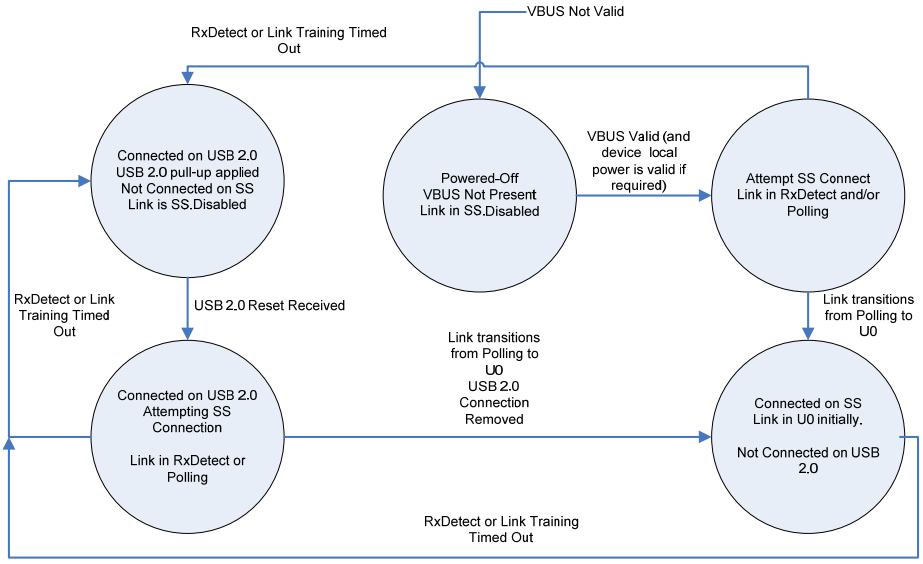


Figure 10-23. Peripheral Device Connect State Machine

10.16.2 Peripheral Device Connect State Descriptions

10.16.2.1 PCONNECT.Powered-off

The PCONNECT.Powered-off state is the default state for a peripheral device. A peripheral device shall transition into this state if any of the following situations occur.

- From any state when VBUS is removed.

In this state, the port's link shall be in the SS.disabled state and the USB 2.0 pull-up is not applied.

10.16.2.2 PCONNECT.Attempt SS Connect

A port shall transition into this state if any of the following situations occur:

- From the PCONNECT.Powered-off state when VBUS becomes valid (and local power is valid if required).

In this state the SuperSpeed link is in RxDetect or Polling and the USB 2.0 pull-up is not applied.

10.16.2.3 PCONNECT.Connected on SS

A port shall transition into this state if any of the following situations occur:

- From the PCONNECT.Attempt SS Connect when the link transitions from polling to U0.
- From the PCONNECT.Connected on USB 2.0 state when the SS link transitions from polling to U0.

In this state the SuperSpeed link is in U0, U1, U2, U3, Inactive, RxDetect, Recovery, or Polling and the USB 2.0 pull-up is not applied.

10.16.2.4 PCONNECT.Connected on USB 2.0

A port shall transition into this state if any of the following situations occur:

- From the PCONNECT.Attempt SS Connect, PCONNECT.Connected on SS, or PCONNECT.Connected on USB 2.0 Attempting SS states if any RxDetect or Polling substate times out.

In this state the SuperSpeed link is in SS.Disabled and the USB 2.0 pull-up is applied.

10.16.2.5 PCONNECT.Connected on USB 2.0 Attempting SS Connection

A port shall transition into this state if any of the following situations occur:

- From the PCONNECT.Connected on USB 2.0 state when a USB 2.0 reset is received.

In this state the SuperSpeed link is in SS.RxDetect or SS.Polling states and the USB 2.0 pull-up is applied.

While in this state, a device shall remove USB 2.0 pull-up if the SuperSpeed link transitions from polling to U0.

10.17 Hub Chapter Parameters

Table 10-18 includes a complete list of the parameters used in the hub chapter.

Table 10-18. Hub Parameters

Name	Description	Min	Max	Units
tDownLinkStateChange	Time from receiving the route string of a header directed to a downstream port that is in a low power link state to initiating a return to U0 on the downstream link.	0	TBD	ns
sDataSymbolsBabble	The number of symbols in a data packet payload after the Data Packet Payload Starting Frame Ordered Set without and Data Packet Payload Ending Frame Ordered Set or Abnormal Data Packet Payload Ending Frame Ordered Set that shall cause a device to detect the packet is invalid.	1030	N/A	symbols
tHubPropRemoteWakeUpstream	Time from start of remote wakeup signaling on the downstream port a hub to when the hub must propagate the remote wakeup signaling on its upstream port if the upstream port link is in U3.	0	1	ms
tHubDriveRemoteWakeDownstream	Time from receiving a SetPortFeature(PORT_LINK_STATE) U0 for a downstream port with a link in U3 to driving remote wakeup signaling on the link.	0	1000	ns
tHubPort2PortExitLat	Time from a downstream port's link initiating a U-state change to when a hub must initiate a U-state change on the upstream port's link (when required).	0	1	uS
tDownstream VBUS Drop	Time that a hub must stop providing VBUS on a downstream port if the hub was providing VBUS on the downstream port and VBUS becomes valid on the upstream port.	100	200	mS
aCurrentUnit	Unit for reporting the current draw of hub controller circuitry in the hub descriptor.		4	mA
nMaxHubPorts	Maximum number of ports on a USB 3.0 hub.		15	Ports
tTimeForResetError	If the downstream port link remains in RxDetec.active for this length of time during a warm reset the reset is considered to have failed.	100	200	mS
tCheckSuperSpeedOnReset	Time from when a device (not a hub) detects a USB 2.0 bus reset to when the device port must enter the USPORT.Powered-On state.	0	1	mS
tUSB2SwitchDisconnect	Time from when a device (not a hub) enters USPORT.Training to when the device must disconnect on the USB 2.0 interface if the device is connected on USB 2.0.	0	1	mS

11 Interoperability and Power Delivery

This chapter defines interoperability and power delivery requirements for USB 3.0. Areas covered include USB 3.0 host and device support for USB 2.0 operation, and USB 3.0 V_{BUS} power consumption limits.

Table 11-1 lists the compatibility matrix for USB 3.0 and USB 2.0. The implication of identifying a host port as supporting USB 3.0 is that both hardware and software support for USB 3.0 is in place; otherwise the port shall only be identified as a USB 2.0 port.

Table 11-1. USB 3.0 and USB 2.0 Interoperability

USB Host Port	USB Device Capability	Connected Mode
USB 2.0	USB 2.0	USB 2.0 High-Speed, Full-Speed or Low-Speed
	USB 3.0	USB 2.0 High-Speed
USB 3.0	USB 2.0	USB 2.0 High-Speed, Full-Speed or Low-Speed
	USB 3.0	USB 3.0 SuperSpeed

It should be noted that USB 3.0 devices are not required to be backward compatible with USB 1.1 host ports although supporting Full-Speed and Low-speed modes are allowed.

11.1 USB 3.0 Host Support for USB 2.0

USB 3.0-capable ports on hosts shall also support USB 2.0 operation in order to enable backward compatibility with USB 2.0 devices. It should be noted however that USB 3.0-capable hosts are not required to support USB 3.0 operation on all of the ports available on the host, i.e., some USB 3.0-capable hosts may have a mix of USB 2.0-only and USB 3.0-capable ports.

To address the situation where a USB 3.0 device is connected to a USB 2.0-only port on a USB 3.0-capable host, after establishing a USB 2.0 High-Speed connection with the device, it is recommended that the host shall inform the user that the device will support SuperSpeed operation if it is moved to a USB 3.0-capable port on the same host. If a USB 3.0 device is connected to a USB 3.0-capable host via a USB 2.0 hub, it is recommended that the host shall inform the user that the device will support SuperSpeed operation if it is moved to an appropriate host port or if the hub is replaced with a USB 3.0 hub.

When a USB 3.0 hub is connected to a host's USB 3.0-capable port, both USB 3.0 SuperSpeed and USB 2.0 High-Speed bus connections shall be allowed to connect and operate in parallel. There is no requirement for a USB 3.0-capable host to support multiple parallel connections to peripheral devices.

The USB 2.0 capabilities of a USB 3.0 host shall be designed to the USB 2.0 specification and shall meet the USB 2.0 compliance requirements.

11.2 USB 3.0 Hub Support for USB 2.0

All ports, both upstream and downstream, on USB 3.0 hubs shall support USB 2.0 operation in order to enable backward compatibility with USB 2.0 devices.

When another USB 3.0 hub is connected in series with a USB 3.0 hub, both USB 3.0 SuperSpeed and USB 2.0 High-Speed bus connections shall be allowed to connect and operate in parallel. There is no requirement for a USB 3.0 hub to support multiple parallel connections to peripheral devices.

Within a USB 3.0 hub, both the USB 3.0 and USB 2.0 hub devices shall implement in the hub framework a common standardized device unique ID (DUID) to enable software to identify the physical relationship of the hub devices.

The USB 2.0 capabilities of a USB 3.0 hub shall be designed to the USB 2.0 specification and shall meet the USB 2.0 compliance requirements.

11.3 USB 3.0 Device Support for USB 2.0

In most cases, backward compatible operation at USB 2.0 High-Speed signaling is supported by USB 3.0 devices in order that higher capability devices are still useful with lesser capable hosts and hubs. For product installations where support for USB 3.0 operation can be independently assured between the device and the host, such as internal devices that are not user accessible, device support for USB 2.0 may not be necessary. USB 3.0 device certification requirements require support for USB 2.0 for all user attached devices.

For any given USB 3.0 peripheral device within a single physical package, only one USB connection mode, either USB 3.0 SuperSpeed or a USB 2.0 speed but not both, shall be established for operation with the host.

Within a physical device, all the individual peripheral devices shall implement in the device framework a common standardized device unique ID (guidUID) to enable software to identify all of the functional components of a specific device and independent of which speed bus it appears on.

The USB 2.0 capabilities of a USB 3.0 device shall be designed to the USB 2.0 specification and shall meet the USB 2.0 compliance requirements.

11.4 Power Distribution

This section describes the USB 3.0 power distribution specification. Note the USB 2.0 power distribution requirements still apply when a USB 3.0 device is operating at high-speed, full-speed or low-speed.

11.4.1 Classes of Devices and Connections

USB 3.0 provides power over two connectors: the Standard-A connector and the Powered-B connector. The following sections focus on the power delivery requirements for the Standard-A connector.

The power source and sink requirements of different device classes can be simplified with the introduction of the concept of a unit load. A unit load for USB 3.0 has been redefined to be 150 mA. The number of unit loads a device can draw is an absolute maximum, not an average over time. A device may be either low-power at one unit load or high-power, consuming up to six unit loads. All devices default to low-power when first powered. The transition to high-power is under software control. It is the responsibility of software to ensure adequate power is available before allowing devices to consume high-power.

The USB supports a range of power sourcing and power consuming agents; these include the following:

- **Root port hubs**: Are directly attached to the USB Host Controller. Hub power is derived from the same source as the Host Controller. Systems that obtain operating power externally, either AC or DC, must be capable of supplying at least six unit loads to each port. Such ports are called high-power ports. Battery-powered systems may supply either one or six unit loads. Ports that can supply only one unit load are termed low-power ports.
- **Self-powered hubs**: Power for the internal functions and downstream facing ports does not come from VBUS. However, the USB interface of the hub may draw up to one unit load from VBUS on its upstream facing port to allow the interface to function when the remainder of the hub is powered down. Hubs that obtain operating power externally (not from VBUS) must supply six unit loads to each port. Battery-powered hubs may supply either one or six unit loads per port.
- **Low-power bus-powered devices**: All power to these devices comes from VBUS. They may draw no more than one unit load at any time.
- **High-power bus-powered devices**: All power to these devices comes from VBUS. They must draw no more than one unit load upon power-up and may draw up to six unit loads after being configured.
- **Ports support the USB Charging Spec V 2.0: TBD**
- **Self-powered devices**: May draw up to one unit load from VBUS to allow the USB interface to function when the remainder of the function is powered down. All other power comes from an external (not from VBUS) source.

No device shall supply (source) current on VBUS at its upstream facing port at any time. From VBUS on its upstream facing port, a device may only draw (sink) current. Devices must also ensure that the maximum operating current drawn by a device is one unit load, until configured.

11.4.1.1 Self-powered Hubs

Self-powered hubs have a local power supply that furnishes power to any non-removable functions and to all downstream facing ports, as shown in Figure 11-1. Power for the Hub Controller, however, may be supplied from the upstream VBUS (a "hybrid" powered hub) or the local power supply. The advantage of supplying the Hub Controller from the upstream supply is that communication from the host is possible even if the device's power supply remains off. This makes it possible to differentiate between a disconnected and an unpowered device. If the hub draws power for its upstream facing port from VBUS, it may not draw more than one unit load.

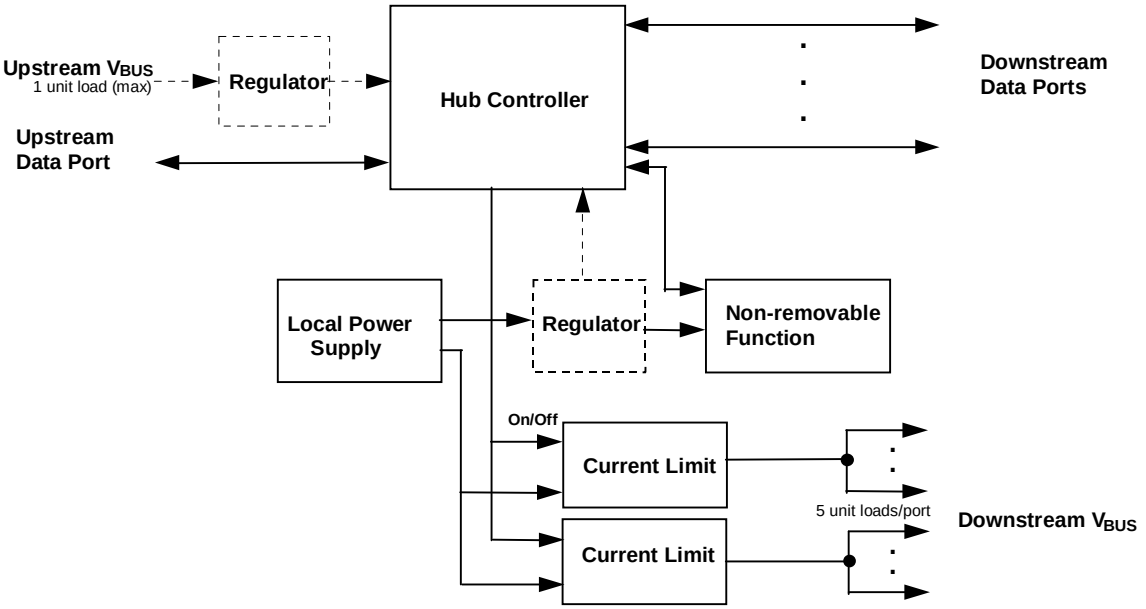


Figure 11-1. Compound Self-powered Hub

The maximum number of ports that can be supported is limited to fifteen and the capability of the local VBUS supply.

Self-powered hubs may experience loss of power. This may be the result of disconnecting the power cord or exhausting the battery.

11.4.1.1.1 Over-current Protection

The host and all self-powered hubs must implement over-current protection for safety reasons, and the hub must have a way to detect the over-current condition and report it to the USB software. Should the aggregate current drawn by a gang of downstream facing ports exceed a preset value, the over-current protection circuit removes or reduces power from all affected downstream facing ports. The over-current condition is reported through the hub to Host Controller, as described in Section TBD. The preset value cannot exceed 5.0 A and must be sufficiently higher than the maximum allowable port current or time delayed such that transient currents (e.g., during power up or dynamic attach or reconfiguration) do not trip the over-current protector. If an over-current condition occurs on any port, subsequent operation of the USB is not guaranteed, and once the

condition is removed, it may be necessary to reinitialize the bus as would be done upon power-up. The over-current limiting mechanism must be resettable without user mechanical intervention. Polymeric PTCs and solid-state switches are examples of methods, which can be used for over-current limiting.

11.4.1.2 Low-power Bus-powered Devices

A low-power device is one that draws up to one unit load from the USB cable when operational. Figure 11-2 shows a typical bus-powered, low-power device, such as a mouse. Low-power regulation can be integrated into the device silicon. Low-power devices must be capable of operating with input VBUS voltages as low as 4.00 V, measured at the plug end of the cable.

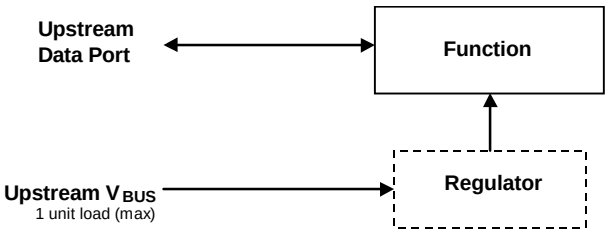


Figure 11-2. Low-power Bus-powered Function

11.4.1.3 High-power Bus-powered Devices

A device is defined as being high-power if, when fully powered, it draws over one but no more than six unit loads from the USB cable. A high-power device requires staged switching of power. It must first come up in a reduced power state of less than one unit load. At bus enumeration time, its total power requirements are obtained and compared against the available power budget. If sufficient power exists, the remainder of the device may be powered on. A typical high-power device is shown in Figure 11-3. The device's electronics have been partitioned into two sections. The device controller contains the minimum amount of circuitry necessary to permit enumeration and power budgeting. The remainder of the device resides in the function block. High-power devices must be capable of operating in their low-power (one unit load) mode with an input voltage as low as 4.00 V, so that it may be detected and enumerated even when plugged into a bus-powered hub. They must also be capable of operating at full power (up to six unit loads) with an input voltage of 4.00 V, measured at the device side of the B-series receptacle.

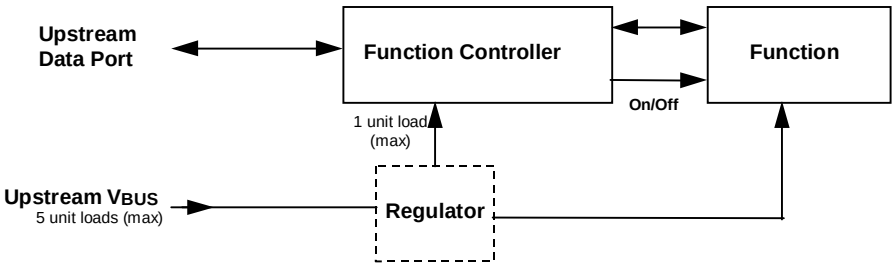


Figure 11-3. High-power Bus-powered Function

11.4.1.4 Self-powered Devices

Figure 11-4 shows a typical self-powered device. The device controller is powered either from the upstream bus via a low-power regulator or from the local power supply. The advantage of the former scheme is that it permits detection and enumeration of a self-powered device whose local power supply is turned off. The maximum upstream power that the device controller can draw is one unit load, and the regulator block must implement inrush current limiting. The amount of power that the device block may draw is limited only by the local power supply. Because the local power supply is not required to power any downstream bus ports, it does not need to implement current limiting, soft start, or power switching.

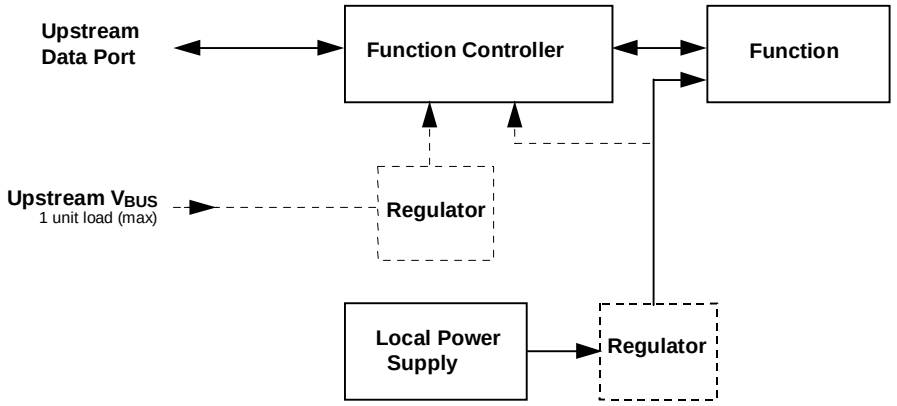


Figure 11-4. Self-powered Function

11.4.2 Steady-State Voltage Drop Budget

The steady–state voltage drop budget is determined from the following:

- The voltage supplied by the internal power supply is 4.75 V to 5.25 V.
- The voltage supplied by high-powered hub ports is 4.45 V to 5.25 V.
- The maximum voltage drop (for detachable cables) between the A-series plug and B-series plug on VBUS is 171 mV.
- The maximum current for the calculations is 0.9 A.
- The maximum voltage drop for all cables between upstream and downstream on GND is 171 mV.
- The maximum voltage drop for all mated connectors is 27 mV.
- All hubs and peripheral devices must be able to provide configuration information with as little as 4.00 V at the device end of their B-series receptacle. Both low and high-power devices need to be operational with this minimum voltage.

Figure 11-5 shows the minimum allowable voltages. Note under transient conditions, the supply at the device can drop to 3.67 V for a brief moment.

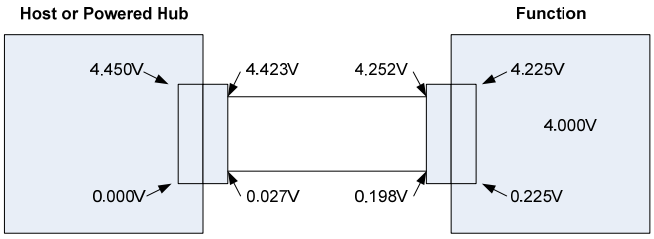


Figure 11-5. Worst-case Voltage Drop Topology (Steady State)

Note: the following assumptions were used:

- 3 meter cable assembly with A-series and B-series plugs
- #22AWG wire used for power and ground (0.019 Ω/foot)
- A-series and B-series plug/receptacle pair have a contact resistance of 30 mΩ
- Wire ~380 mΩ series resistance
- IR Drop at device = (((2 * 30 mΩ) + 190 mΩ) * 900 mA) * 2 or 0.450 V

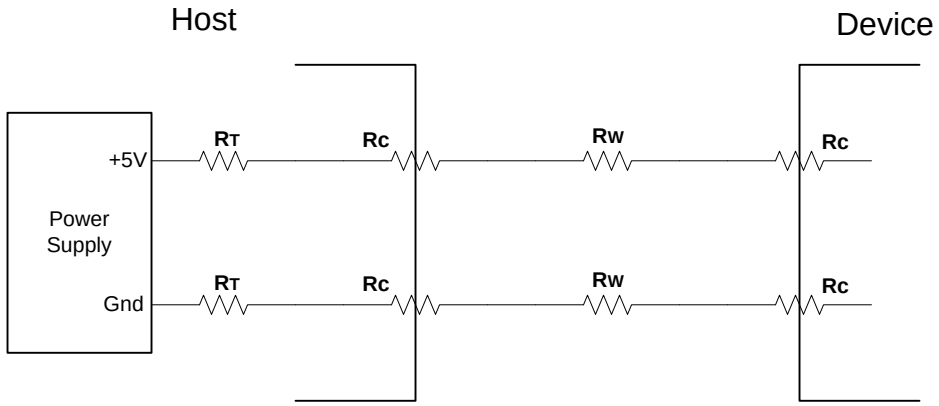


Figure 11-6. Worst-case Voltage Drop Analysis Using Equivalent Resistance.

Rt	Host Trace Resistance:	0.167 Ω
Rc	Mated Connector Resistance:	0.030 Ω
Rw	Cable Resistance:	0.190 Ω

11.4.3 Power Control During Suspend/Resume

Suspend current is a function of unit load allocation. All USB devices initially default to low-power. SuperSpeed devices operating at low-power are limited to 500 μA of suspend current. If the device is configured for high-power and enabled as a remote wakeup source, it may draw up to 2.5 mA during suspend. If a hub is not configured, it is operating as a low-power device and must limit its suspend current to 500 μA.

While in the Suspend state, a device may briefly draw more than the average current. The amplitude of the current spike cannot exceed the device power allocation 150 mA (or 900 mA). A

maximum of 1.0 second is allowed for an averaging interval. The average current cannot exceed the average suspend current limit (I_{CCSH} or I_{CCSL}, see Table 11-2) during any 1.0-second interval ($T_{SUSAVG1}$). The profile of the current spike is restricted so the transient response of the power supply (which may be an efficient, low-capacity, trickle power supply) is not overwhelmed. The rising edge of the current spike must be no more than 100 mA/µs. Downstream facing ports must be able to absorb the 900 mA peak current spike and meet the voltage droop requirements defined for inrush current during dynamic attach (see Section TBD). Figure 11-7 illustrates a typical example profile for an averaging interval.

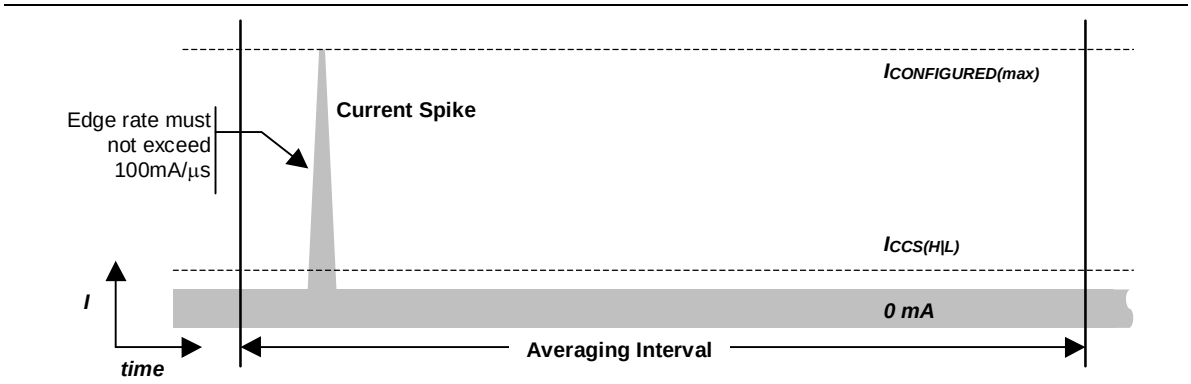


Figure 11-7. Typical Suspend Current Averaging Profile

Devices are responsible for handling the bus voltage reduction due to the inductive and resistive effects of the cable. When a hub is in the Suspend state, it must still be able to provide the maximum current per port (six unit loads per port for self-powered hubs). This is necessary to support remote wakeup-capable devices that will power-up while the remainder of the system is still suspended. Such devices, when enabled to do remote wakeup, must drive resume signaling upstream within 10 ms of starting to draw the higher, non-suspend current. Devices not capable of remote wakeup must not draw the higher current when suspended.

When devices wakeup, either by themselves (remote wakeup) or by seeing resume signaling, they must limit the inrush current on V_{BUS}. The device must have sufficient on-board bypass capacitance or a controlled power-on sequence such that the current drawn from the hub does not exceed the maximum current capability of the port at any time while the device is waking up.

11.4.4 Dynamic Attach and Detach

The act of plugging or unplugging a hub or peripheral device must not affect the functionality of another device on other segments of the network. Unplugging a device will stop the transaction between that device and the host. However, the hub to which this device was attached will recover from this condition and will alert the host that the port has been disconnected.

11.4.4.1 Inrush Current Limiting

When a peripheral device or hub is plugged into the network, it has a certain amount of on-board capacitance between VBUS and ground. In addition, the regulator on the device may supply current to its output bypass capacitance and to the device as soon as power is applied. Consequently, if no measures are taken to prevent it, there could be a surge of current into the device which might pull the VBUS on the hub below its minimum operating level. Inrush currents can also occur when a high-power device is switched into its high-power mode. This problem must be solved by limiting the inrush current and by providing sufficient capacitance in each hub to prevent the power supplied to the other ports from going out of tolerance. An additional motivation for limiting inrush current is to minimize contact arcing, thereby prolonging connector contact life.

The maximum droop possible in the hub VBUS is 330 mV. In order to meet this requirement, the following conditions must be met:

- The maximum load (C_{RPB}) that can be placed at the downstream end of a cable is 10 μF in parallel with as small as a 27 Ω resistance. The 10 μF capacitance represents any bypass capacitor directly connected across the VBUS lines in the device plus any capacitive effects visible through the regulator in the device. The 27 Ω resistance represents one unit load of current drawn by the device during connect.
- If more bypass capacitance is required in the device, then the device must incorporate some form of VBUS surge current limiting, such that it matches the characteristics of the above load.
- The hub downstream facing port VBUS power lines must be bypassed (C_{HPB}) with no less than 120 μF of low-ESR capacitance per hub. Standard bypass methods should be used to minimize inductance and resistance between the bypass capacitors and the connectors to reduce droop. The bypass capacitors themselves should have a low dissipation factor to allow decoupling at higher frequencies.

The upstream facing port of a hub is also required to meet the above requirements.

A high-power bus-powered device that is switching from a lower power configuration to a higher power configuration must not cause droop > 330 mV on the VBUS at its upstream hub. The device can meet this by ensuring that changes in the capacitive load it presents do not exceed 10 μF.

11.4.4.2 Dynamic Detach

When a device is detached from the network with power flowing in the cable, the inductance of the cable will cause a large flyback voltage to occur on the open end of the device cable. This flyback voltage is not destructive. Proper bypass measures on the hub ports will suppress any coupled noise. This will require some low capacitance, very low inductance bypass capacitors on each hub port connector. The flyback voltage and the noise it creates is also moderated by the bypass capacitance on the device end of the cable. Also, there must be some minimum capacitance on the device end of the cable to ensure that the inductive flyback on the open end of the cable does not cause the voltage on the device end to reverse polarity. A minimum of 1.0 μF is recommended for bypass across V_{BUS}.

11.4.5 V_{BUS} Electrical Characteristics

Table 11-2. DC Electrical Characteristics

Parameter	Symbol	Conditions	Min.	Max.	Units
Supply Voltage:					
High-power Port	V_{BUS}		4.45	5.25	V
Low-power Port	V_{BUS}		4.70	5.25	V
Supply Current:					
High-power Hub Port (out)	I_{CCPRT}		900		mA
Low-power Hub Port (out)	I_{CCUPT}		150		mA
High-power Peripheral Device (in)	I_{CCHPF}			900	mA
Low-power Peripheral Device (in)	I_{CCLPF}			150	mA
Unconfigured Device (in)	I_{CCINIT}			150	mA
Suspended High-power Device	I_{CCSH}			2.5	mA
Suspended Low-power Device	I_{CCSL}			500	μA

11.4.6 Powered-B Connector

The Powered-B connector was defined to allow devices such as printers to connect to and power devices such as Wireless USB adapters. This improves usability by eliminating the need for an external power supply (e.g. 'wall-wart') to power the adapter.

The Powered-B receptacle shall meet the following electrical requirements:

- The Powered-B receptacle shall supply 5 V +/- 10% over the entire load range (0-1 A).
- The Powered-B receptacle shall be capable of supplying 1 A.
- The Powered-B receptacle shall implement over-current protection for safety reasons.
- The Powered-B receptacle shall deliver full power whenever the device is turned on or in standby.
- A device that exposes a Powered-B receptacle shall operate as a low-power device.
- A device that is powered (in whole or in part) by a Powered-B receptacle shall not expose any Standard-A ports

11.4.7 Wire Gauge Table

Table 11-3 is a table of VBUS/Gnd wire gauges showing the relationship between gauge and maximum length in order to achieve the previously cited voltage drop values. The user should note that these lengths are the maximum length possible to meet the voltage drop budget, thus gauges smaller and lengths greater than the table values will fail to deliver the expected voltage value.

Table 11-3. VBUS/Gnd Wire Gauge vs. Maximum Length

American Stranded Wire Gauge (AWG) on VBUS/Gnd	Ohms Per 100 Meters (Maximum)	Maximum Cable Length (Meters)
28	23.20	0.8
26	14.60	1.3
24	9.09	2.0
22	5.74	3.0
20	3.58	5.3

A Symbol Encoding

Table A-1 shows the byte-to-Symbol encodings for data characters. Table A-2 shows the Symbol encodings for the Special Symbols used for TLP/DLLP Framing and for interface management. RD- and RD+ refer to the Running Disparity of the Symbol sequence on a per-Lane basis.

Table A-1. 8b/10b Data Symbol Codes

Data Byte Name	Data Byte Value (hex)	Bits HGF EDCBA (binary)	Current RD- abcdei fghj(binary)	Current RD+ abcdei fghj (binary)
D0.0	00	000 00000	100111 0100	011000 1011
D1.0	01	000 00001	011101 0100	100010 1011
D2.0	02	000 00010	101101 0100	010010 1011
D3.0	03	000 00011	110001 1011	110001 0100
D4.0	04	000 00100	110101 0100	001010 1011
D5.0	05	000 00101	101001 1011	101001 0100
D6.0	06	000 00110	011001 1011	011001 0100
D7.0	07	000 00111	111000 1011	000111 0100
D8.0	08	000 01000	111001 0100	000110 1011
D9.0	09	000 01001	100101 1011	100101 0100
D10.0	0A	000 01010	010101 1011	010101 0100
D11.0	0B	000 01011	110100 1011	110100 0100
D12.0	0C	000 01100	001101 1011	001101 0100
D13.0	0D	000 01101	101100 1011	101100 0100
D14.0	0E	000 01110	011100 1011	011100 0100
D15.0	0F	000 01111	010111 0100	101000 1011
D16.0	10	000 10000	011011 0100	100100 1011
D17.0	11	000 10001	100011 1011	100011 0100
D18.0	12	000 10010	010011 1011	010011 0100
D19.0	13	000 10011	110010 1011	110010 0100
D20.0	14	000 10100	001011 1011	001011 0100
D21.0	15	000 10101	101010 1011	101010 0100
D22.0	16	000 10110	011010 1011	011010 0100
D23.0	17	000 10111	111010 0100	000101 1011
D24.0	18	000 11000	110011 0100	001100 1011
D25.0	19	000 11001	100110 1011	100110 0100
D26.0	1A	000 11010	010110 1011	010110 0100
D27.0	1B	000 11011	110110 0100	001001 1011
D28.0	1C	000 11100	001110 1011	001110 0100
D29.0	1D	000 11101	101110 0100	010001 1011
D30.0	1E	000 11110	011110 0100	100001 1011
D31.0	1F	000 11111	101011 0100	010100 1011

Data Byte Name	Data Byte Value (hex)	Bits HGF EDCBA (binary)	Current RD- abcdei fghj(binary)	Current RD+ abcdei fghj (binary)
D0.1	20	001 00000	100111 1001	011000 1001
D1.1	21	001 00001	011101 1001	100010 1001
D2.1	22	001 00010	101101 1001	010010 1001
D3.1	23	001 00011	110001 1001	110001 1001
D4.1	24	001 00100	110101 1001	001010 1001
D5.1	25	001 00101	101001 1001	101001 1001
D6.1	26	001 00110	011001 1001	011001 1001
D7.1	27	001 00111	111000 1001	000111 1001
D8.1	28	001 01000	111001 1001	000110 1001
D9.1	29	001 01001	100101 1001	100101 1001
D10.1	2A	001 01010	010101 1001	010101 1001
D11.1	2B	001 01011	110100 1001	110100 1001
D12.1	2C	001 01100	001101 1001	001101 1001
D13.1	2D	001 01101	101100 1001	101100 1001
D14.1	2E	001 01110	011100 1001	011100 1001
D15.1	2F	001 01111	010111 1001	101000 1001
D16.1	30	001 10000	011011 1001	100100 1001
D17.1	31	001 10001	100011 1001	100011 1001
D18.1	32	001 10010	010011 1001	010011 1001
D19.1	33	001 10011	110010 1001	110010 1001
D20.1	34	001 10100	001011 1001	001011 1001
D21.1	35	001 10101	101010 1001	101010 1001
D22.1	36	001 10110	011010 1001	011010 1001
D23.1	37	001 10111	111010 1001	000101 1001
D24.1	38	001 11000	110011 1001	001100 1001
D25.1	39	001 11001	100110 1001	100110 1001
D26.1	3A	001 11010	010110 1001	010110 1001
D27.1	3B	001 11011	110110 1001	001001 1001
D28.1	3C	001 11100	001110 1001	001110 1001
D29.1	3D	001 11101	101110 1001	010001 1001
D30.1	3E	001 11110	011110 1001	100001 1001
D31.1	3F	001 11111	101011 1001	010100 1001
D0.2	40	010 00000	100111 0101	011000 0101
D1.2	41	010 00001	011101 0101	100010 0101
D2.2	42	010 00010	101101 0101	010010 0101
D3.2	43	010 00011	110001 0101	110001 0101
D4.2	44	010 00100	110101 0101	001010 0101
D5.2	45	010 00101	101001 0101	101001 0101
D6.2	46	010 00110	011001 0101	011001 0101
D7.2	47	010 00111	111000 0101	000111 0101

Data Byte Name	Data Byte Value (hex)	Bits HGF EDCBA (binary)	Current RD- abcdei fghj(binary)	Current RD+ abcdei fghj (binary)
D8.2	48	010 01000	111001 0101	000110 0101
D9.2	49	010 01001	100101 0101	100101 0101
D10.2	4A	010 01010	010101 0101	010101 0101
D11.2	4B	010 01011	110100 0101	110100 0101
D12.2	4C	010 01100	001101 0101	001101 0101
D13.2	4D	010 01101	101100 0101	101100 0101
D14.2	4E	010 01110	011100 0101	011100 0101
D15.2	4F	010 01111	010111 0101	101000 0101
D16.2	50	010 10000	011011 0101	100100 0101
D17.2	51	010 10001	100011 0101	100011 0101
D18.2	52	010 10010	010011 0101	010011 0101
D19.2	53	010 10011	110010 0101	110010 0101
D20.2	54	010 10100	001011 0101	001011 0101
D21.2	55	010 10101	101010 0101	101010 0101
D22.2	56	010 10110	011010 0101	011010 0101
D23.2	57	010 10111	111010 0101	000101 0101
D24.2	58	010 11000	110011 0101	001100 0101
D25.2	59	010 11001	100110 0101	100110 0101
D26.2	5A	010 11010	010110 0101	010110 0101
D27.2	5B	010 11011	110110 0101	001001 0101
D28.2	5C	010 11100	001110 0101	001110 0101
D29.2	5D	010 11101	101110 0101	010001 0101
D30.2	5E	010 11110	011110 0101	100001 0101
D31.2	5F	010 11111	101011 0101	010100 0101
D0.3	60	011 00000	100111 0011	011000 1100
D1.3	61	011 00001	011101 0011	100010 1100
D2.3	62	011 00010	101101 0011	010010 1100
D3.3	63	011 00011	110001 1100	110001 0011
D4.3	64	011 00100	110101 0011	001010 1100
D5.3	65	011 00101	101001 1100	101001 0011
D6.3	66	011 00110	011001 1100	011001 0011
D7.3	67	011 00111	111000 1100	000111 0011
D8.3	68	011 01000	111001 0011	000110 1100
D9.3	69	011 01001	100101 1100	100101 0011
D10.3	6A	011 01010	010101 1100	010101 0011
D11.3	6B	011 01011	110100 1100	110100 0011
D12.3	6C	011 01100	001101 1100	001101 0011
D13.3	6D	011 01101	101100 1100	101100 0011
D14.3	6E	011 01110	011100 1100	011100 0011
D15.3	6F	011 01111	010111 0011	101000 1100

Data Byte Name	Data Byte Value (hex)	Bits HGF EDCBA (binary)	Current RD- abcdei fghj(binary)	Current RD+ abcdei fghj (binary)
D16.3	70	011 10000	011011 0011	100100 1100
D17.3	71	011 10001	100011 1100	100011 0011
D18.3	72	011 10010	010011 1100	010011 0011
D19.3	73	011 10011	110010 1100	110010 0011
D20.3	74	011 10100	001011 1100	001011 0011
D21.3	75	011 10101	101010 1100	101010 0011
D22.3	76	011 10110	011010 1100	011010 0011
D23.3	77	011 10111	111010 0011	000101 1100
D24.3	78	011 11000	110011 0011	001100 1100
D25.3	79	011 11001	100110 1100	100110 0011
D26.3	7A	011 11010	010110 1100	010110 0011
D27.3	7B	011 11011	110110 0011	001001 1100
D28.3	7C	011 11100	001110 1100	001110 0011
D29.3	7D	011 11101	101110 0011	010001 1100
D30.3	7E	011 11110	011110 0011	100001 1100
D31.3	7F	011 11111	101011 0011	010100 1100
D0.4	80	100 00000	100111 0010	011000 1101
D1.4	81	100 00001	011101 0010	100010 1101
D2.4	82	100 00010	101101 0010	010010 1101
D3.4	83	100 00011	110001 1101	110001 0010
D4.4	84	100 00100	110101 0010	001010 1101
D5.4	85	100 00101	101001 1101	101001 0010
D6.4	86	100 00110	011001 1101	011001 0010
D7.4	87	100 00111	111000 1101	000111 0010
D8.4	88	100 01000	111001 0010	000110 1101
D9.4	89	100 01001	100101 1101	100101 0010
D10.4	8A	100 01010	010101 1101	010101 0010
D11.4	8B	100 01011	110100 1101	110100 0010
D12.4	8C	100 01100	001101 1101	001101 0010
D13.4	8D	100 01101	101100 1101	101100 0010
D14.4	8E	100 01110	011100 1101	011100 0010
D15.4	8F	100 01111	010111 0010	101000 1101
D16.4	90	100 10000	011011 0010	100100 1101
D17.4	91	100 10001	100011 1101	100011 0010
D18.4	92	100 10010	010011 1101	010011 0010
D19.4	93	100 10011	110010 1101	110010 0010
D20.4	94	100 10100	001011 1101	001011 0010
D21.4	95	100 10101	101010 1101	101010 0010
D22.4	96	100 10110	011010 1101	011010 0010
D23.4	97	100 10111	111010 0010	000101 1101

Data Byte Name	Data Byte Value (hex)	Bits HGF EDCBA (binary)	Current RD- abcdei fghj(binary)	Current RD+ abcdei fghj (binary)
D24.4	98	100 11000	110011 0010	001100 1101
D25.4	99	100 11001	100110 1101	100110 0010
D26.4	9A	100 11010	010110 1101	010110 0010
D27.4	9B	100 11011	110110 0010	001001 1101
D28.4	9C	100 11100	001110 1101	001110 0010
D29.4	9D	100 11101	101110 0010	010001 1101
D30.4	9E	100 11110	011110 0010	100001 1101
D31.4	9F	100 11111	101011 0010	010100 1101
D0.5	A0	101 00000	100111 1010	011000 1010
D1.5	A1	101 00001	011101 1010	100010 1010
D2.5	A2	101 00010	101101 1010	010010 1010
D3.5	A3	101 00011	110001 1010	110001 1010
D4.5	A4	101 00100	110101 1010	001010 1010
D5.5	A5	101 00101	101001 1010	101001 1010
D6.5	A6	101 00110	011001 1010	011001 1010
D7.5	A7	101 00111	111000 1010	000111 1010
D8.5	A8	101 01000	111001 1010	000110 1010
D9.5	A9	101 01001	100101 1010	100101 1010
D10.5	AA	101 01010	010101 1010	010101 1010
D11.5	AB	101 01011	110100 1010	110100 1010
D12.5	AC	101 01100	001101 1010	001101 1010
D13.5	AD	101 01101	101100 1010	101100 1010
D14.5	AE	101 01110	011100 1010	011100 1010
D15.5	AF	101 01111	010111 1010	101000 1010
D16.5	B0	101 10000	011011 1010	100100 1010
D17.5	B1	101 10001	100011 1010	100011 1010
D18.5	B2	101 10010	010011 1010	010011 1010
D19.5	B3	101 10011	110010 1010	110010 1010
D20.5	B4	101 10100	001011 1010	001011 1010
D21.5	B5	101 10101	101010 1010	101010 1010
D22.5	B6	101 10110	011010 1010	011010 1010
D23.5	B7	101 10111	111010 1010	000101 1010
D24.5	B8	101 11000	110011 1010	001100 1010
D25.5	B9	101 11001	100110 1010	100110 1010
D26.5	BA	101 11010	010110 1010	010110 1010
D27.5	BB	101 11011	110110 1010	001001 1010
D28.5	BC	101 11100	001110 1010	001110 1010
D29.5	BD	101 11101	101110 1010	010001 1010
D30.5	BE	101 11110	011110 1010	100001 1010
D31.5	BF	101 11111	101011 1010	010100 1010

Data Byte Name	Data Byte Value (hex)	Bits HGF EDCBA (binary)	Current RD- abcdei fghj(binary)	Current RD+ abcdei fghj (binary)
D0.6	C0	110 00000	100111 0110	011000 0110
D1.6	C1	110 00001	011101 0110	100010 0110
D2.6	C2	110 00010	101101 0110	010010 0110
D3.6	C3	110 00011	110001 0110	110001 0110
D4.6	C4	110 00100	110101 0110	001010 0110
D5.6	C5	110 00101	101001 0110	101001 0110
D6.6	C6	110 00110	011001 0110	011001 0110
D7.6	C7	110 00111	111000 0110	000111 0110
D8.6	C8	110 01000	111001 0110	000110 0110
D9.6	C9	110 01001	100101 0110	100101 0110
D10.6	CA	110 01010	010101 0110	010101 0110
D11.6	CB	110 01011	110100 0110	110100 0110
D12.6	CC	110 01100	001101 0110	001101 0110
D13.6	CD	110 01101	101100 0110	101100 0110
D14.6	CE	110 01110	011100 0110	011100 0110
D15.6	CF	110 01111	010111 0110	101000 0110
D16.6	D0	110 10000	011011 0110	100100 0110
D17.6	D1	110 10001	100011 0110	100011 0110
D18.6	D2	110 10010	010011 0110	010011 0110
D19.6	D3	110 10011	110010 0110	110010 0110
D20.6	D4	110 10100	001011 0110	001011 0110
D21.6	D5	110 10101	101010 0110	101010 0110
D22.6	D6	110 10110	011010 0110	011010 0110
D23.6	D7	110 10111	111010 0110	000101 0110
D24.6	D8	110 11000	110011 0110	001100 0110
D25.6	D9	110 11001	100110 0110	100110 0110
D26.6	DA	110 11010	010110 0110	010110 0110
D27.6	DB	110 11011	110110 0110	001001 0110
D28.6	DC	110 11100	001110 0110	001110 0110
D29.6	DD	110 11101	101110 0110	010001 0110
D30.6	DE	110 11110	011110 0110	100001 0110
D31.6	DF	110 11111	101011 0110	010100 0110
D0.7	E0	111 00000	100111 0001	011000 1110
D1.7	E1	111 00001	011101 0001	100010 1110
D2.7	E2	111 00010	101101 0001	010010 1110
D3.7	E3	111 00011	110001 1110	110001 0001
D4.7	E4	111 00100	110101 0001	001010 1110
D5.7	E5	111 00101	101001 1110	101001 0001
D6.7	E6	111 00110	011001 1110	011001 0001
D7.7	E7	111 00111	111000 1110	000111 0001

Data Byte Name	Data Byte Value (hex)	Bits HGF EDCBA (binary)	Current RD- abcdei fghj(binary)	Current RD+ abcdei fghj (binary)
D8.7	E8	111 01000	111001 0001	000110 1110
D9.7	E9	111 01001	100101 1110	100101 0001
D10.7	EA	111 01010	010101 1110	010101 0001
D11.7	EB	111 01011	110100 1110	110100 1000
D12.7	EC	111 01100	001101 1110	001101 0001
D13.7	ED	111 01101	101100 1110	101100 1000
D14.7	EE	111 01110	011100 1110	011100 1000
D15.7	EF	111 01111	010111 0001	101000 1110
D16.7	F0	111 10000	011011 0001	100100 1110
D17.7	F1	111 10001	100011 0111	100011 0001
D18.7	F2	111 10010	010011 0111	010011 0001
D19.7	F3	111 10011	110010 1110	110010 0001
D20.7	F4	111 10100	001011 0111	001011 0001
D21.7	F5	111 10101	101010 1110	101010 0001
D22.7	F6	111 10110	011010 1110	011010 0001
D23.7	F7	111 10111	111010 0001	000101 1110
D24.7	F8	111 11000	110011 0001	001100 1110
D25.7	F9	111 11001	100110 1110	100110 0001
D26.7	FA	111 11010	010110 1110	010110 0001
D27.7	FB	111 11011	110110 0001	001001 1110
D28.7	FC	111 11100	001110 1110	001110 0001
D29.7	FD	111 11101	101110 0001	010001 1110
D30.7	FE	111 11110	011110 0001	100001 1110
D31.7	FF	111 11111	101011 0001	010100 1110

Table A-2. 8b/10b Special Character Symbol Codes

Data Byte Name	Data Byte Value	Bits HGF EDCBA	Current RD - abcdei fghj	Current RD + abcdei fghj
K28.0	1C	000 11100	001111 0100	110000 1011
K28.1	3C	001 11100	001111 1001	110000 0110
K28.2	5C	010 11100	001111 0101	110000 1010
K28.3	7C	011 11100	001111 0011	110000 1100
K28.4	9C	100 11100	001111 0010	110000 1101
K28.5	BC	101 11100	001111 1010	110000 0101
K28.6	DC	110 11100	001111 0110	110000 1001
K28.7	FC	111 11100	001111 1000	110000 0111
K23.7	F7	111 10111	111010 1000	000101 0111
K27.7	FB	111 11011	110110 1000	001001 0111
K29.7	FD	111 11101	101110 1000	010001 0111
K30.7	FE	111 11110	011110 1000	100001 0111

B Symbol Scrambling

B.1 Data Scrambling

The following subroutines encode and decode an 8-bit value contained in “inbyte” with the LFSR. This is presented as one example only; there are many ways to obtain the proper output. This example demonstrates how to advance the LFSR eight times in one operation and how to XOR the data in one operation. Many other implementations are possible but they must all produce the same output as that shown here.

The following algorithm uses the “C” programming language conventions, where “<<” and “>>” represent the shift left and shift right operators, “>” is the compare greater than operator, and “^” is the exclusive or operator, and “&” is the logical “AND” operator.

```
/*
   this routine implements the serial descrambling algorithm in parallel form
   for the LSFR polynomial:  x^16+x^5+x^4+x^3+1
   this advances the LSFR 8 bits every time it is called
   this requires fewer than 25 xor gates to implement (with a static register)

   The XOR required to advance 8 bits/clock is:
bit   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15

      8   9  10  11  12  13  14  15   0   1   2   3   4   5   6   7
                  8   9  10  11  12  13  14  15
                      8   9  10  11  12  13  14  15
                          8   9  10  11  12  13  14  15

   The serial data is just the reverse of the upper byte:
bit   0   1   2   3   4   5   6   7
     15  14  13  12  11  10   9   8
*/
```

```
int scramble_byte(int inbyte)
{

static int scrambit[16];
static int bit[16];
static int bit_out[16];
static unsigned short lfsr = 0xffff; // 16 bit short for polynomial
int i, outbyte;

    if (inbyte == COMMA)   // if this is a comma
    {
        lfsr = 0xffff;     // reset the LFSR
        return (COMMA);    // and return the same data
    }

    if (inbyte == SKIP)    // don't advance or encode on skip
        return (SKIP);

    for (i=0; i<16;i++)    // convert LFSR to bit array for legibility
        bit[i] = (lfsr >> i) & 1;

    for (i=0; i<8; i++)    // convert byte to be scrambled for legibility
        scrambit[i] = (inbyte >> i) & 1;

    // apply the xor to the data
    if (! (inbyte & 0x100) &&    // if not a KCODE, scramble the data
        ! (TrainingSequence == TRUE))  // and if not in the middle of
    {                                  // a training sequence
        scrambit[0] ^= bit[15];
        scrambit[1] ^= bit[14];
        scrambit[2] ^= bit[13];
        scrambit[3] ^= bit[12];
        scrambit[4] ^= bit[11];
        scrambit[5] ^= bit[10];
        scrambit[6] ^= bit[9];
        scrambit[7] ^= bit[8];
    }
```

```
    // Now advance the LFSR 8 serial clocks
    bit_out[ 0] = bit[ 8];
    bit_out[ 1] = bit[ 9];
    bit_out[ 2] = bit[10];
    bit_out[ 3] = bit[11] ^ bit[ 8];
    bit_out[ 4] = bit[12] ^ bit[ 9] ^ bit[ 8];
    bit_out[ 5] = bit[13] ^ bit[10] ^ bit[ 9] ^ bit[ 8];
    bit_out[ 6] = bit[14] ^ bit[11] ^ bit[10] ^ bit[ 9];
    bit_out[ 7] = bit[15] ^ bit[12] ^ bit[11] ^ bit[10];
    bit_out[ 8] = bit[ 0] ^ bit[13] ^ bit[12] ^ bit[11];
    bit_out[ 9] = bit[ 1] ^ bit[14] ^ bit[13] ^ bit[12];
    bit_out[10] = bit[ 2] ^ bit[15] ^ bit[14] ^ bit[13];
    bit_out[11] = bit[ 3]           ^ bit[15] ^ bit[14];
    bit_out[12] = bit[ 4]                     ^ bit[15];
    bit_out[13] = bit[ 5];
    bit_out[14] = bit[ 6];
    bit_out[15] = bit[ 7];
    lfsr = 0;
    for (i=0; i <16; i++) // convert the LFSR back to an integer
        lfsr += (bit_out[i] << i);

    outbyte = 0;
    for (i= 0; i<8; i++) // convert data back to an integer
        outbyte += (scrambit[i] << i);

    return outbyte;
}

/*  NOTE THAT THE DESCRAMBLE ROUTINE IS IDENTICAL TO THE SCRAMBLE ROUTINE
    this routine implements the serial descrambling algorithm in parallel form
    this advances the lfsr 8 bits every time it is called
    this uses fewer than 25 xor gates to implement (with a static register)
    The XOR tree is the same as the scrambling routine
*/
```

```
int unscramble_byte(int inbyte)
{

    static int descrambit[8];
    static int bit[16];
    static int bit_out[16];
    static unsigned short lfsr = 0xffff;  // 16 bit short for polynomial
    int outbyte, i;

    if (inbyte == COMMA)   // if this is a comma
    {
       lfsr = 0xffff;      // reset the LFSR
       return (COMMA);     // and return the same data
    }

    if (inbyte == SKIP)   // don't advance or encode on skip
        return (SKIP);

    for (i=0; i<16;i++)   // convert the LFSR to bit array for legibility
        bit[i] = (lfsr >> i) & 1;

    for (i=0; i<8; i++)   // convert byte to be de-scrambled for legibility
        descrambit[i] = (inbyte >> i) & 1;

    // apply the xor to the data
    if (! (inbyte & 0x100) &&    // if not a KCODE, scramble the data
        ! (TrainingSequence == TRUE))  // and if not in the middle of
    {                                  // a training sequence
        descrambit[0] ^= bit[15];
        descrambit[1] ^= bit[14];
        descrambit[2] ^= bit[13];
        descrambit[3] ^= bit[12];
        descrambit[4] ^= bit[11];
        descrambit[5] ^= bit[10];
        descrambit[6] ^= bit[9];
        descrambit[7] ^= bit[8];
    }
```

```
    // Now advance the LFSR 8 serial clocks
    bit_out[ 0] = bit[ 8];
    bit_out[ 1] = bit[ 9];
    bit_out[ 2] = bit[10];
    bit_out[ 3] = bit[11] ^ bit[ 8];
    bit_out[ 4] = bit[12] ^ bit[ 9] ^ bit[ 8];
    bit_out[ 5] = bit[13] ^ bit[10] ^ bit[ 9] ^ bit[ 8];
    bit_out[ 6] = bit[14] ^ bit[11] ^ bit[10] ^ bit[ 9];
    bit_out[ 7] = bit[15] ^ bit[12] ^ bit[11] ^ bit[10];
    bit_out[ 8] = bit[ 0] ^ bit[13] ^ bit[12] ^ bit[11];
    bit_out[ 9] = bit[ 1] ^ bit[14] ^ bit[13] ^ bit[12];
    bit_out[10] = bit[ 2] ^ bit[15] ^ bit[14] ^ bit[13];
    bit_out[11] = bit[ 3]           ^ bit[15] ^ bit[14];
    bit_out[12] = bit[ 4]                     ^ bit[15];
    bit_out[13] = bit[ 5];
    bit_out[14] = bit[ 6];
    bit_out[15] = bit[ 7];
    lfsr = 0;
    for (i=0; i <16; i++) // convert the LFSR back to an integer
        lfsr += (bit_out[i] << i);

    outbyte = 0;
    for (i= 0; i<8; i++) // convert data back to an integer
        outbyte += (descrambit[i] << i);

    return outbyte;
}
```

The initial 16-bit values of the LFSR for the first 128 LFSR advances following a reset are listed below:

	0, 8	**1, 9**	**2, A**	**3, B**	**4, C**	**5, D**	**6, E**	**7, F**
00	FFFF	E817	0328	284B	4DE8	E755	404F	4140
08	4E79	761E	1466	6574	7DBD	B6E5	FDA6	B165
10	7D09	02E5	E572	673D	34CF	CB54	4743	4DEF
18	E055	40E0	EE40	54BE	B334	2C7B	7D0C	07E5
20	E5AF	BA3D	248A	8DC4	D995	85A1	BD5D	4425
28	2BA4	A2A3	B8D2	CBF8	EB43	5763	6E7F	773E
30	345F	5B54	5853	5F18	14B7	B474	6CD4	DC4C
38	5C7C	70FC	F6F0	E6E6	F376	603B	3260	64C2
40	CB84	9743	5CBF	B3FC	E47B	6E04	0C3E	3F2C
48	29D7	D1D1	C069	7BC0	CB73	6043	4A60	6FFA
50	F207	1102	01A9	A939	2351	566B	6646	4FF6
58	F927	3081	85B0	AC5D	478C	82EF	F3F2	E43B
60	2E04	027E	7E72	79AE	A501	1A7D	7F2A	2197
68	9019	0610	1096	9590	8FCD	D0E7	F650	46E6
70	E8D6	C228	3AB2	B70A	129F	9CE2	FC3C	2B5C
78	5AA3	AF6A	70C7	CDF0	E3D5	C0AB	B9C0	D9C1

An 8-bit value of 0 repeatedly encoded with the LFSR after reset produces the following consecutive 8-bit values:

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	FF	17	C0	14	B2	E7	02	82	72	6E	28	A6	BE	6D	BF	8D
10	BE	40	A7	E6	2C	D3	E2	B2	07	02	77	2A	CD	34	BE	E0
20	A7	5D	24	B1	9B	A1	BD	22	D4	45	1D	D3	D7	EA	76	EE
30	2C	DA	1A	FA	28	2D	36	3B	3A	0E	6F	67	CF	06	4C	26
40	D3	E9	3A	CD	27	76	30	FC	94	8B	03	DE	D3	06	52	F6
50	4F	88	80	95	C4	6A	66	F2	9F	0C	A1	35	E2	41	CF	27
60	74	40	7E	9E	A5	58	FE	84	09	60	08	A9	F1	0B	6F	62
70	17	43	5C	ED	48	39	3F	D4	5A	F5	0E	B3	C7	03	9D	9B
80	8B	0D	8E	5C	33	98	77	AE	2D	AC	0B	3E	DA	0B	42	7A
90	7C	D1	CF	A8	1C	12	EE	41	C2	3F	38	7A	0D	69	F4	01
A0	DA	31	72	C5	A0	D7	93	0E	DC	AF	A4	55	E7	F0	72	16
B0	68	D5	38	84	DD	00	CD	18	9E	CA	30	59	4C	75	1B	77
C0	31	C5	ED	CF	91	64	6E	3D	FE	E8	29	04	CF	6C	FC	C4
D0	0B	5E	DA	62	BA	5B	AB	DF	59	B7	7D	37	5E	E3	1A	C6
E0	88	14	F5	4F	8B	C8	56	CB	D3	10	42	63	04	8A	B4	F7
F0	84	01	A0	01	83	49	67	EE	3E	2A	8B	A4	76	AF	14	D5
100	4F	AC	60	B6	79	D6	62	B7	43	E7	E5	2A	40	2C	6E	7A
110	56	61	63	20	6A	97	4A	38	05	E5	DD	68	0D	78	4C	53
120	8B	D6	86	57	B2	AA	1A	80	18	DC	BA	FC	03	A3	4B	30

Scrambling produces the power spectrum (in the 10-bit domain) shown in Figure B-1.

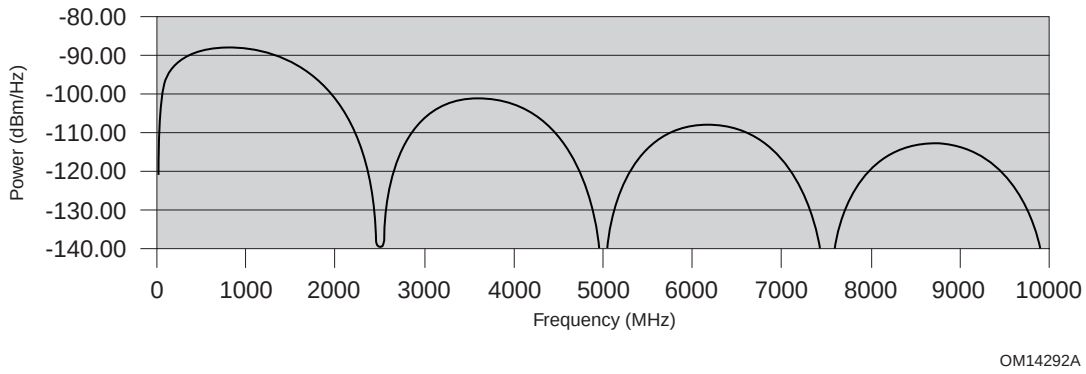


Figure B-1. Scrambling Spectrum for Data Value of 0

C Power Management

This appendix offers power management implementation guidelines for USB 3.0 SuperSpeed devices. It is intended to aide in the development of power efficient devices. The following topics are covered:

- PM overview
- SuperSpeed vs. High Speed Interface Selection Considerations for Devices
- Link PM implementation guidelines
- Device PM implementation guidelines
- Calculation Examples for U1 and U2 System Exit Latency Fields

The intended audience is developers of USB 3.0 devices.

C.1 SuperSpeed Power Management Overview

USB 3.0 SuperSpeed power management (PM) requirements are distributed throughout this specification. This section ties it all together as a description of a coherent power management system.

The key elements of PM in USB 3.0 are:

- Devices send asynchronous busy / ready notifications to the host rather than the host polling devices. This eliminates wasting power from polling devices even when there is no real work to be done.
- Packets are unicast (e.g. routed), not broadcast, allowing links that are not needed to remain in a low power state.
- Four link states with progressively lower power and greater exit latency.
- Host and device driven link PM mechanisms that facilitate placing links into progressively lower power states.
- Function level suspend and device wide suspend which allow devices to remove power from the portions of their circuitry that are not in use.

This PM overview is broken down into the following subsections:

- Link PM states and entry & exit processes
- Link PM for hubs
- Other mechanisms in support of link PM, e.g. for isochronous transfers
- Device suspend and function suspend
- Platform PM support (Latency Tolerance Messaging)

C.1.1 Link PM States and Entry & Exit Processes

Link PM enables links to be placed into low power states while idle. As a link becomes progressively less active deeper power savings can be achieved, progressing from U0 (link active) to U1 (link standby with fast exit), to U2 (link standby with slow exit), and finally to U3 (suspend).

After being configured by software, U1 and U2 are typically entered and exited automatically under hardware control, with U2 providing more power savings than U1 at the cost of additional exit latency. U3 is entered under software control, typically after a software implemented inactivity timeout, and exit can be initiated by either software (host initiated exit) or hardware (remote wakeup). U3 is directly coupled to the device suspend state (see Section C.1.4).

The following sub-sections first provide a summary of the link PM states, and then give an overview of each link PM state, how each are entered and exited, and the associated software interface.

C.1.1.1 Summary of Link PM States

A summary of link PM states and their characteristics is given in Table C-1. Each link PM state is described in more detail in the subsections that follow.

Table C-1. Link PM States and Characteristics Summary

Link State	Description	Characteristics	H/W and/or S/W Directed[1]	Device Clock Gen On/Off	Exit Latency Range
U0	Link active	Link operational state	N/A	On	N/A
U1	Link idle – fast exit	Rx & Tx circuitry quiesced	H/W and S/W	On or Off	µs
U2	Link idle – slow exit	Clock generation circuitry may additionally be quiesced	H/W and S/W	On/Off[2]	µs – ms
U3	Link suspend	Interface (e.g. Physical Layer) power may be removed	Entry: S/W only[3] Exit: H/W and S/W	Off	µs – ms

Notes:

1. S/W directed means that entry and exit can be initiated by software. H/W directed means that entry and exit can be initiated by hardware.
2. Except where required by implementation, it is intended for devices to quiescent their clock generation circuitry (e.g., PLL) during U2. An example of the exception is with hubs, which may share one PLL across all ports, and as a result it may not make sense for its PLL to be shut off until all of its ports are in U2 or a lower power state.
3. An exception is when a device is connected to a hub that is in suspend with remote wakeup due to a connect event disabled. See the Hub chapter for further information.

C.1.1.2 U0 – Link Active

U0 is the normal link operational state. All packet communication, whether for control or data transfer, occurs in this state.

C.1.1.3 U1 – Link Idle with Fast Exit

U1 is a low exit latency standby state. The electrical characteristics of this state allow substantial power savings in comparison with U0. Exit latency is dominated by the time to achieve receiver symbol lock and the link training process.

Both devices (i.e., upstream ports) and hub downstream ports may initiate U1 entry. Devices may initiate entry as soon as they determine that the link is idle or is not needed in the near future, e.g. using the Packet Pending field (see Section C.1.3.1 below). Each hub downstream port has an independent inactivity timer. If the port is in U0 and neither transmits nor receives any packets (Transaction Packets, Data Packets, or Link Management Packets, but not including timestamp packets nor link commands) during a period of time specified by the inactivity timeout, then the downstream port initiates U1 entry. The downstream port's link partner may either accept or reject the U1 entry request, e.g. if the link partner has some activity which will shortly require a packet transmission then it may reject the U1 entry request. A rejection to an entry request simply resets and restarts the inactivity timer with the programmed timeout. Specific recommendations for when devices should initiate U1 entry, and for how to set the U1 inactivity timeout, are provided in Section TBD.

When a device or downstream port initiates U1 entry, its link partner may either accept or reject. The link level U1 entry process consists of one port transmitting a LGO_U1 link command, and its link partner responding with either a LAU (link accept U-state) or LXU (link reject U-state) command (see the Link chapter for further information).

Either a device or its link partner can initiate U1 exit. U1 exit is typically initiated when a packet needs to be transmitted, e.g., an IN message from the host or an ERDY message from a device. The exit process (i.e., transition back to U0) consists of a Low Frequency Periodic Signaling (LFPS) handshake (see PHY chapter) followed by link Recovery and training (see Link chapter).

The software interface associated with U1 is used to enable the device to initiate U1 entry (see the U1_Enable feature in the Framework chapter), to enable U1 on hub downstream ports and set the inactivity timeout (see the PORT_U1_TIMEOUT feature in the Hub chapter), and to inform the device what the total latency is to transition the entire path of links between the device and host from U1 to U0 under worst case circumstances when the transition is initiated by the device (see the SET_SEL standard device request in the Framework chapter, and also Section C.5 for example calculations). Devices also report their maximum U1 exit latency using the U1DevExitLat field in the Device Capabilities Descriptor (see the Framework chapter).

The U1_Enable feature controls whether a device upstream port may initiate U1 entry, but does not control whether the device upstream port accepts or rejects U1 entry requests by its link partner. A device upstream port is always enabled to accept U1 entry requests, though a device may choose to reject an entry request based on e.g. activity within the device. More information about programming the PORT_U1_TIMEOUT feature is given in Section C.1.2.3.

Composite devices should consider the needs of all functions within the device prior to initiating entry to a low power link state.

C.1.1.4 U2 – Link Idle with Slow Exit

U2 is a low to medium range exit latency standby state. Exit latencies are the same as for U1, plus clock generation (e.g., PLL) startup time if the clock generation circuitry is quiesced during U2.

The purpose of U2 is to allow clock generation circuitry (e.g., PLLs) to be quiesced in order to save additional power in comparison with U1. Devices are encouraged to quiescent clock generation circuitry whenever possible to save power, however under some implementation specific circumstances this may not make sense. For example, a hub may save substantial cost by sharing one PLL across all of its ports, and as a result its PLL can not be quiesced unless all of its ports are in U2 or a lower power state, but when all of its ports are in U2 or a lower power state then the PLL can be quiesced.

U2 is typically entered directly from U1, though U2 may also be entered from U0. If the U1 inactivity timeout is non-zero, then the U2 inactivity timeout represents an amount of time that a link is in U1, i.e. the U2 inactivity timer begins ticking when a link enters U1, and when the timeout expires both link partners silently drop from U1 to U2. Neither link partner is able to reject entry. Both link partners implement a timer, and use the exact same timeout value. A consistent timeout between the two link partners is accomplished by only giving software access to programming the U2 inactivity timeout in downstream ports. Downstream ports automatically send a Link Management Packet containing the timeout value to their link partner whenever the inactivity timeout is changed (see the Hub chapter for further information, or the Protocol chapter for the LMP packet definition). The accuracy of these timers is also specified (see Hub chapter), but it should be noted that this accuracy can never be perfect so there can be brief periods of time where one port is in U1 and its link partner is in U2, however the U1 exit and U2 exit processes are compatible with each other so as to accommodate these corner conditions seamlessly.

If the U1 inactivity timeout is zero, then the U2 inactivity timeout represents an amount of time that a link is inactive, i.e. no packets (TPs, DPs, or LMPs, but not including timestamp packets nor link commands) are transmitted or received by a port. In this case, when the U2 inactivity timer expires the downstream port initiates U2 entry from U0. A downstream port may also initiate U2 entry from U0 as a result of a SetPortFeature(PORT_LINK_STATE) U2 request.

Devices (i.e. upstream ports) may also initiate U2 entry from U0. Note however that if a device's U2 inactivity timer is enabled, and its link is already in U1, it is intended for the direct U1 to U2 transition to be used as opposed to transitioning from U1 to U0 and then to U2. When a port initiates U2 entry from U0, its link partner may either accept or reject based on whether it has packets to transmit (assuming U2 is enabled). The link level U0 to U2 transition process consists of one port transmitting a LGO_U2 link command, and its link partner responding with either a LAU (link accept U-state) or LXU (link reject U-state) command (see the Link chapter for further information).

Either a device or its link partner can initiate U2 exit. U2 exit is initiated when a packet needs to be transmitted, e.g., an IN message from the host or an ERDY message from a device. The exit process (i.e. transition back to U0) consists of a Low Frequency Periodic Signaling (LFPS) handshake (see PHY chapter) followed by link Recovery and training (see Link chapter).

The software interface associated with U2 is used to enable the device to initiate U2 entry (see the U2_Enable feature in the Framework chapter), to enable U2 on hub downstream ports and set the inactivity timeout (see the PORT_U2_TIMEOUT feature in the Hub chapter), and to inform the device what the total latency is to transition the entire path of links between the device and host

from U2 to U0 under worst case circumstances when the transition is initiated by the device (see the SET_SEL standard device request in the Framework chapter, and also Section C.5 for example calculations). Devices also report their maximum U2 exit latency using the U2DevExitLat field in the Device Capabilities Descriptor (see the Framework chapter).

The U2_Enable feature controls whether a device upstream port may initiate U2 entry from U0. More information about programming the PORT_U2_TIMEOUT feature is given in Section C.1.2.3.

Composite devices should consider the needs of all functions within the device prior to initiating entry to a low power link state.

C.1.1.5 U3 – Link Suspend

U3 is a deep power saving state where interface (e.g., Physical Layer) power may be removed, except as needed to perform the following functions:

- For upstream ports in devices and hubs:
  - Reset signaling detection
  - Wakeup signaling detection (for host initiated wakeup)
  - Wakeup signaling transmission (for remote wakeup capable devices)
- For downstream ports in hubs and hosts:
  - Reset signaling generation
  - Disconnect event detection
  - Wakeup signaling detection (for remote wakeup)
  - Wakeup signaling transmission (for host initiated wakeup)

The purpose of U3 is to minimize power consumption during suspend. Devices may remove power from most circuitry while retaining power for circuitry needed during suspend. For the two components connected on a given link, each component may switch off interface power independently.

VBUS remains on during U3. Any power rail switching by devices is implementation specific and beyond the scope of this specification. (See Section C.4.2 for implementation guidelines re. power rail switching.)

U3 entry may only be initiated by a downstream port as part of the port suspend process (see the SetPortFeature hub class request and PORT_LINK_STATE feature in the Hub chapter). Software (e.g. function driver) typically implements an inactivity timeout for the purpose of placing a function into suspend, independent of other functions or devices in the system (see Section C.1.4 for further information about device suspend and function suspend). Specific recommendations for when functions (devices) should initiate suspend (U3) entry are provided in Section C.4.1.

When a downstream port initiates U3 entry, its link partner is not allowed to reject the request. The link level U3 entry process consists of one port transmitting a LGO_U3 link command, and its link partner responding with a LAU (link accept U-state) command (see the Link chapter for further information).

Either a device or the host can initiate U3 exit (wakeup). The host initiates U3 exit by issuing a SetPortFeature(PORT_LINK_STATE) request to the desired downstream port. Devices initiate U3 exit due to a remote wakeup event on the device, e.g. a connect event on a hub downstream port, if

enabled to do so by the host. The exit process (i.e. transition back to U0) consists of a Low Frequency Periodic Signaling (LFPS) handshake (see PHY chapter) followed by link Recovery and training (see Link chapter). If exit was initiated by a device, the specific function(s) within the device that had a remote wakeup event sends a Function Wake device notification packet (see the Protocol chapter for further information) to the host. The Function Wake message causes an interrupt to software (see the appropriate host controller specification for further information). Also see Section C.1.4.5 for a description of how remote wakeup works across a fabric of hubs and devices.

For host initiated U3 exit (wakeup), the LFPS handshake process allows devices up to 20ms to complete, allowing sufficient time for a device to turn on a switched power rail if implemented. The LFPS handshake takes 20ms in total if the device begins sending LFPS 10ms after receiving LFPS, plus waiting an additional 10ms after beginning LFPS to begin link Recovery and training. See the timing defined in the PHY chapter for further information.

The software interface associated with U3 consists of a set of port controls and function controls. The port controls, e.g. initiating U3 on a hub downstream port, are described in Section C.1.2.3. The function controls, e.g. enabling a function (device) for remote wakeup, are described in Section C.1.4.1.

C.1.2 Link PM for Hubs

Hubs play multiple critical roles in link PM:

- Coordinates the upstream port link PM state with that of its downstream ports.
- Handles packet deferring, where it tells the host that a packet was sent to a downstream port not in U0.
- Provides inactivity timers on downstream ports to initiate U1 and U2 entry.

C.1.2.1 Link PM State Coordination and Management

Hubs monitor their downstream port link PM states and keep their upstream port in the lowest power link PM state they can without allowing it to be in a lower power state than any of its downstream ports (assuming the upstream port's link partner always accepts U1 and U2 entry requests, which will normally be the case).

This hub link PM behavior allows a device to control the link PM state of all the links between itself and the host. For example, if the device keeps its link in U0, then the hub immediately above the device will keep its upstream port in U0, which results in all links in the path between the device and host to stay in U0 as long as the device keeps its link in U0. A similar example could be given with U1, where the device allows U1 but not U2, thereby limiting the latency to transition all links in the path to the host back to U0.

This hub link PM behavior also has the affect that when a device places its link in a low power state, this low power state can propagates upstream through hubs to the host, limited of course by the link PM state of other devices in the fabric.

When a device initiates a transition from a low power link PM state back to U0 on a hub downstream port, the hub begins transitioning its upstream port to U0 simultaneously with the downstream port. This minimizes the total time to transition all links in the path to the host back to U0, for upstream flowing packets. For downstream flowing packets, on the other hand, hubs will first receive the packet and decode which downstream port the packet is targeted for, and then only

initiate a transition to U0 on that particular downstream port. This saves power by only powering up links that are needed to perform useful work.

C.1.2.2 Packet Deferring

When the host sends a packet to a device connected to a hub and the device's link is not in U0, packet deferring allows the host to perform other transfers while waiting for the device's link to return to U0. When a hub receives a packet targeted to a downstream port in a non-U0 state, the hub informs the host of this by sending a deferred command back to the host, and also sends the original packet header to the device with the deferred field asserted. The host treats this deferred command similar to a NRDY, and may then initiate transfers with other devices instead of waiting for the link to return to U0. (See the Hub chapter for further information.)

When a device receives an IN or an OUT header with the deferred field asserted it prepares for the transfer, sends an ERDY to the host when ready, and keeps its link in U0 until the transfer occurs (or until an error timeout occurs). The host responds to the ERDY by rescheduling the original transfer.

C.1.2.3 Software Interface

The software interface for hub link PM consists of the following port controls and status fields:

- the PORT_LINK_STATE feature and port status field,
- the PORT_REMOTE_WAKE_MASK feature,
- the C_PORT_LINK_STATE port status change bit,
- the PORT_U1_TIMEOUT feature, and
- the PORT_U2_TIMEOUT feature.

Other hub port controls can impact link PM behavior, e.g. the PORT_RESET feature, but are not covered here. (See the Hub chapter for detailed information about the above features and port status change bit, and other hub port controls.)

The PORT_LINK_STATE feature is used to request a link PM state change from any U-state to any U-state, though its primary intended use is between U0 and U3. The value of the feature sent in the SetPortFeature request indicates which link PM state is requested, as defined in the Hub chapter. There is also a PORT_LINK_STATE port status field that can be read with a GetPortStatus request and indicates the current U-state state for a port's link.

The PORT_REMOTE_WAKE_MASK feature is used to mask each remote wakeup event that might be originated at a hub downstream port. For example, a connect event may be unmasked, i.e., enabled to cause a remote wakeup event, while a disconnect event may be masked, i.e., disabled from causing a remote wakeup event. If connect and disconnect events are disabled from generating a remote wakeup event, these events are still detected by the hub and reported as port status change events after the hub is resumed. This saves power by allowing these events to be serviced by software at a more convenient time instead of resuming a computer only for the purpose of servicing these events and returning to suspend right away.

The PORT_REMOTE_WAKE_MASK feature does not control whether a hub forwards upstream remote wakeup signaling it received from a downstream device. Hubs always forward upstream remote wakeup signaling received on a downstream port. (See the Hub chapter for further information.)

The C_PORT_LINK_STATE port status change bit is used to signal completion of a transition from U3 to U0, resulting from a host initiated wakeup (i.e. a SetPortFeature(PORT_LINK_STATE) request), on a hub downstream port. This port status change event causes an interrupt to software that the resume process for the hub port, and the device attached to the port, has been completed. Note that C_PORT_LINK_STATE is not asserted due to a remote wakeup. For remote wakeup, the function (device) originating the remote wakeup sends a Function Wake device notification packet to the host, which also causes a software interrupt (see Section C.1.4.1).

The PORT_U1_TIMEOUT feature is used to enable and disable U1 on hub downstream ports, and also to set an inactivity timeout for initiating U1 entry. When a downstream port is disabled for U1 entry, i.e., PORT_U1_TIMEOUT set to zero, then the downstream port will neither initiate entry nor accept entry requests. A PORT_U1_TIMEOUT of 0xFF disables the downstream port from initiating U1 entry, but enables it to accept entry requests. Values in between zero and 0xFF set an inactivity timeout.

The PORT_U2_TIMEOUT feature is used to enable and disable U2 on hub downstream ports, and also to set an inactivity timeout for initiating a transition to U2. When a downstream port is disabled for U2 entry, i.e. PORT_U2_TIMEOUT set to zero, then the downstream port will neither initiate entry nor accept entry requests. A PORT_U2_TIMEOUT of 0xFF disables the direct U1 to U2 transition and U2 inactivity timeout, but enables the downstream port to accept U2 entry requests while in U0. Values in between zero and 0xFF set an inactivity timeout.

Some information that may be useful in determining inactivity timeouts is listed below:

- U1 and U2 device exit latencies (i.e., U1DevExitLat and U2DevExitLat fields in the Device Capabilities Descriptor, see the Framework chapter for further information).
- U1 and U2 System Exit Latency fields (see the SET_SEL standard device request in the Framework chapter, and also Section C.5 for example calculations) , or the total U1 and U2 exit latencies for all the links in the path between the device and the host.

Also see Section TBD for specific recommendations on determining appropriate inactivity timeouts.

It should also be noted that link PM at the fabric level will not work properly unless software programs hubs in accordance with the following:

- Any given link should not be enabled for a lower power link state than any links hierarchically below it. For example, U2 should only be enabled on a hub upstream port if (a) U2 is enabled on all of its downstream ports, or (b) the downstream ports are in a lower power state such as Disabled.
- A device should not be placed in a lower power state than any device below it in the hierarchy. For example, a hub should not be placed into suspend (upstream port in U3) unless all of its downstream ports are in U3 or a lower power state such as Disabled.
- A hub downstream port should not be programmed to enable direct U1 to U2 transitions if the device connected under the hub downstream port can not tolerate the exit latency associated with U2, or if it impairs the device's use of Latency Tolerance Messaging (see Section C.1.5). Software implementations may offer a mechanism for devices to influence hub programming of PORT_U1_TIMEOUT and PORT_U2_TIMEOUT features.
- If a downstream port of a hub (e.g., Hub1) is connected to another hub (e.g., Hub2), the U2 inactivity timeout value for Hub2's upstream port should be programmed to be at least a couple microseconds greater than the largest U2 inactivity timeout for all of Hub2's downstream ports.

This is done for margin due to timer inaccuracies, defined in the Hub and Framework chapters for hub downstream ports and devices respectively. Note that Hub2's upstream port U2 inactivity timeout is actually programmed in a Hub1 downstream port.

C.1.3 Other Link PM Support Mechanisms

C.1.3.1 Packet Pending Flag

Devices may use the Packet Pending flag (see the Protocol chapter) to help decide when to place their link in a low power state. The Packet Pending flag provides an indication of whether the host controller has any additional packets to transfer on the schedule associated with a given endpoint. When there are no more packets pending for all endpoints on a device, the device may be able to place its link in a low power state immediately. See Section C.3.1 for device implementation recommendations for when to initiate low power link states.

C.1.3.2 Support for Isochronous Transfers

If a link is in a low power state when an isochronous transfer is scheduled, the latency to transition to U0 could potentially delay the transfer beyond its subscribed isochronous service contract. The ping, ping response protocol allows the host to transition all links in the direct path to the device to U0 prior to initiating an isochronous transfer. Host controllers are provided with information about how long it can take (worst case) to transition all links in the path between the host and device to U0 when the transition is initiated by the host (see the appropriate host controller specification for further information). This allows the host to schedule ping messages far enough ahead of time to ensure meeting the service contract.

The ping, ping response process consists of the following:

- The host sends a ping message to a device.
- Hubs route the ping message toward the targeted device.
- The device responds to the ping message by sending a ping response message to the host.
- The device keeps its link in U0 until it receives a subsequent packet from the host (up to an error timeout). This results in all links in the path between the device and host staying in U0, due to the rules for hub link states (see Section C.1.2.1).

After sending a ping message to one device and prior to receiving a ping response message, the host may transfer data with other devices. Once the host receives a ping response from a given device, the host knows all the links in the path are in U0 and can then schedule a transfer with that device without the data transmission incurring any link state exit delays along the path to the device.

Timestamps convey timing information for isochronous endpoints. Timestamps are sent by the host every bus interval when a root port is in U0. Hubs propagate timestamps downstream on each port in U0. In order to receive timestamp information a device needs to transition all links in the direct path to the host to U0 prior to a bus interval boundary. Hub downstream port inactivity timers ignore (e.g. are not reset due to) timestamp packets.

C.1.3.3 Support for Interrupt Transfers

If a link is in a low power state when an interrupt transfer is scheduled, the latency to transition to U0 could potentially delay the transfer beyond the subscribed interrupt service contract. Such problems are mitigated by the host initiating the transfer ahead of time to compensate for the time it takes to transition all links in the path to the device back to U0. Host controllers are provided with information about how long it can take (worst case) to transition all links in the path between the host and device to U0 when the transition is initiated by the host (see the appropriate host controller specification for further information). This allows the host to schedule interrupt transfers far enough ahead of time to ensure meeting the service contract.

C.1.4 Device PM

Device power management is primarily under software control, with various hardware mechanisms to support it. Device PM consists of some function level mechanisms plus some device and hub mechanisms.

In the following sub-sections, first function level suspend is described. Then device and hub mechanisms are described, including how it all ties together in a fabric (e.g., how remote wakeup occurs across a fabric).

C.1.4.1 Function Suspend

The function suspend state is a low power state associated with an individual function. The function may or may not be part of a composite device. A function may be placed into function suspend independently of other functions using the FUNCTION_SUSPEND feature (see the Framework chapter). The FUNCTION_SUSPEND feature is also used to enable function remote wakeup. The Function Remote Wake Capable field (see Framework Chapter) indicates whether a function is capable of function remote wakeup.

If a composite device has one function in suspend but another function in use, and the link is in e.g. U0, a mechanism is needed for the function in suspend to signal a remote wakeup. This is done with the Function Wake device notification packet (see the Protocol chapter). When a remote wakeup event occurs, if the device's link is in a non-U0 state then it is transitioned to U0 using the appropriate process, and then the function in which the remote wakeup event occurred sends a Function Wake message to the host. The Function Wake message generates an interrupt to software (see the appropriate host controller specification for further information) and indicates which function initiated the remote wakeup. Note the DEVICE_REMOTE_WAKEUP feature from USB 2.0 is not used by USB 3.0 devices.

C.1.4.2 Device Suspend

Device suspend is a device-wide state entered and exited intrinsically as a result of a device's upstream port entering and exiting the U3 state. A device may be transitioned into device suspend regardless of the function suspend state of any function within the device.

Devices may implement a switched power rail and remove power from large portions of the device while in suspend. Implementation recommendations in this regard are provided in Section C.4.3. Some device state information must be retained in a persistent state during suspend (see the Framework chapter for further information).

C.1.4.3 Device and Fabric Suspend Entry Process

A device is transitioned into suspend using the following process:

- If remote wakeup is to be enabled, at least one function is enabled for function remote wakeup.
- A SetPortFeature(PORT_LINK_STATE) request is issued to the downstream port which the device is directly connected.
- The downstream port initiates U3 entry by sending a LGO_U3 link command.
- The device sends an LAU link command.
- Both link partners transition their transmitters to electrical idle and enter the U3 state. (See the Link chapter for further information about the link level U3 entry process.)

A fabric of devices and hubs, or a subset thereof, is placed into suspend on a device by device basis under software control. First all leaf devices in the fabric (or subset thereof) are placed into suspend. Then the hub(s) connected to the leaf devices is (are) placed into suspend. This is repeated up the hierarchy until the desired fabric has been placed in suspend.

C.1.4.4 Host Initiated Device and Fabric Wakeup Process

A host initiated suspend exit (wakeup) process is illustrated in Figure C-1 and consists of the following steps, after an event occurs within the system which requires the device to be woken up from suspend:

1. All hubs and downstream ports in the path to the device are successively woken up from suspend, if they were in suspend, starting with the hub nearest the host.
   a. At each hierarchical level, if a downstream port is in U3 then the PORT_LINK_STATE feature is set to U0.
      - For hubs this is accomplished by software issuing a SetPortFeature(PORT_LINK_STATE) request to the port. Refer to the Hub chapter for further information.
      - For root ports please refer to the relevant host controller specification.
   b. In response to the PORT_LINK_STATE feature being set to U0 the downstream port transmits wakeup signaling to its link partner.
   c. The link and downstream port complete the transition to U0 as defined in the Link and PHY chapters. This completes the wakeup process for the link and downstream port.
   d. After the wakeup process has completed for any given downstream port, the port generates a C_PORT_LINK_STATE port status change interrupt. This informs software that the wakeup process has completed for that port.
   e. Software repeats the above process until all links in the path to the device have returned to U0.
2. The fabric wakeup process is completed when the device's link has transitioned to U0.

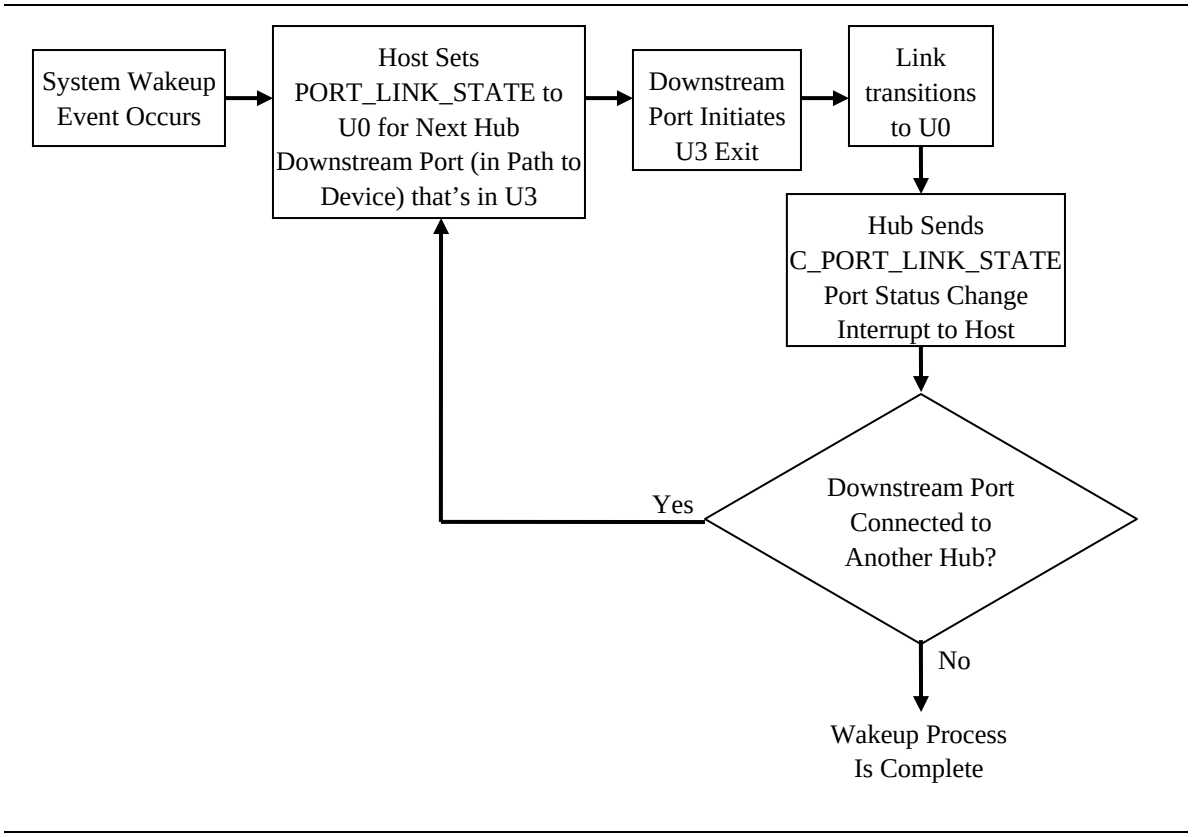


Figure C-1. Flow Diagram for Host Initiated Wakeup

C.1.4.5 Device and Fabric Remote Wakeup Process

A device initiated suspend exit (remote wakeup) process consists of the following steps, after an event occurs within the device which requires the device to be woken up from suspend:

1. The device transmits (LFPS) wakeup signaling (see PHY chapter) to its link partner.
2. Wakeup signaling is propagated up the hierarchy by hubs until reached by the Controlling Hub. The Controlling Hub is the hub not in suspend hierarchically closest to the device, or, in cases where there are no intermediate hubs or all intermediate hubs are in suspend then the Controlling Hub is the host (root hub).
3. The Controlling Hub transmits (LFPS) wakeup signaling on its downstream port which received wakeup signaling. Each hub in the direct path to the remote wakeup device propagates the wakeup signaling downstream on the hub downstream port(s) that received wakeup signaling. See the Hub chapter for hub timing requirements associated with this process. As the wakeup signaling is propagated downstream, each link completes the LFPS handshake (see PHY chapter) and transitions to U0.
4. After all of the links between the Controlling Hub and the remote wakeup device transition to U0, the function within the remote wakeup device that originated the remote wakeup sends a Function Wake device notification packet to the host, which causes a software interrupt. Software then clears the function suspend state (see FUNCTION_SUSPEND feature) associated with the function that originated the remote wakeup.

C.1.5 Platform PM Support

Latency Tolerance Messaging (LTM) allows platforms to make dynamic power vs. performance tradeoffs cooperatively with devices without imposing additional cost (e.g., no additional device buffering is needed to enable the system to enter deeper power saving states). Devices may use LTM to help platforms (systems) save power by informing the system when devices are idle vs. active vs. very active, with some level of granularity.

Using an LTM notification containing a Best Effort Latency Tolerance (BELT) value (see Protocol chapter), a device reports how much latency it can tolerate from the system, but only for best effort services such as bulk. LTM is not used where a guaranteed latency or bandwidth service is needed, such as for isochronous service.

LTM provides information to the system that may allow it to enter deeper power saving states than would otherwise be reliable, but only when all of the devices in the system allow it. Any device in the system may choose to operate in a high performance mode at all times, and require the system to stay in a high state of readiness to service that device with low latency, however this may incur a substantial cost in terms of platform power and battery life for mobile systems.

Upon receipt of an LTM packet the host identifies the smallest BELT across all devices in the system. The smallest BELT is used along with other system information to determine what platform sleep state may be entered at any given point in time.

LTM support is optional for devices. For devices that do not support LTM a default BELT may be assumed by the system (see tBELTdefault in the Protocol chapter).

Devices that do support LTM send an LTM packet with an updated BELT value whenever their activity level changes significantly. For example, when a device becomes idle (e.g. has been inactive for some period), the device reports a larger BELT. When a device becomes active it reports a smaller BELT. One BELT applies across an entire device, so the needs of all functions and endpoints should be considered in determining the BELT value, generally taking the smallest one. Specific BELT values are device implementation dependent as well as platform implementation dependent.

The BELT value is also impacted by use of link PM. The BELT represents an amount of latency allocated to the system and does not include the latency to transition links back to U0 from U1 or U2. The latency to transition all links between a device and the host back to U0, when initiated by the device, is programmed into the U1SystemExitLat and U2SystemExitLat fields in each device using the SET_SEL request (see Framework chapter). Different implementations may be used, however generally speaking when a device is active (i.e. not totally idle) it should subtract the U1SystemExitLat from its total latency tolerance and send the remaining value as its BELT, and when a device is idle it should subtract the U2SystemExitLat (assuming U2 is enabled). This allows a device's link to go in and out of U1 and U2 at any time without needing to send a new LTM every time the link state changes.

The software interface for LTM consists of the LTM Capable field in the SuperSpeed Device Capabilities descriptor, and the LTM_Enable feature (see Framework chapter).

C.2 SuperSpeed vs. High Speed Interface Selection Considerations for Devices

SuperSpeed versus High Speed interface selection for a new device design can have a major impact on system power consumption for many system implementations. While a device is actively transferring data, system components are also transferring that data and may consume larger amounts of power during the transfer. In many systems the power consumption of these system components can dwarf that of the entire USB device. In such systems, under typical circumstances the faster the data transfer completes the faster the system components can return to a low power state, which can save power on average over time. Examples of these system components may include a host controller, a core logic component (e.g. a DRAM controller), DRAM components, a microprocessor with a cache that needs to snoop DRAM accesses, etc.

Figure C-2 and Figure C-3 illustrate an example of a device that has an average data transfer rate of 10MBytes/s when actively in use with SuperSpeed and High Speed interfaces respectively. The system power figures are roughly representative of typical notebook computer systems. The difference in system power consumption between the two systems is due to the interface power of the device and its link partner (no hubs present). While no data transfers are taking place the difference in system power consumption between the two systems is small, and is approximated as zero. Link power management considerations are ignored for simplicity of illustration.

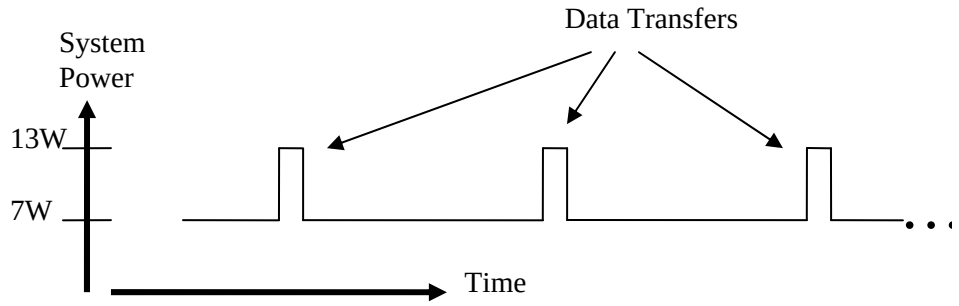


Figure C-2. SuperSpeed Device Data Transfer

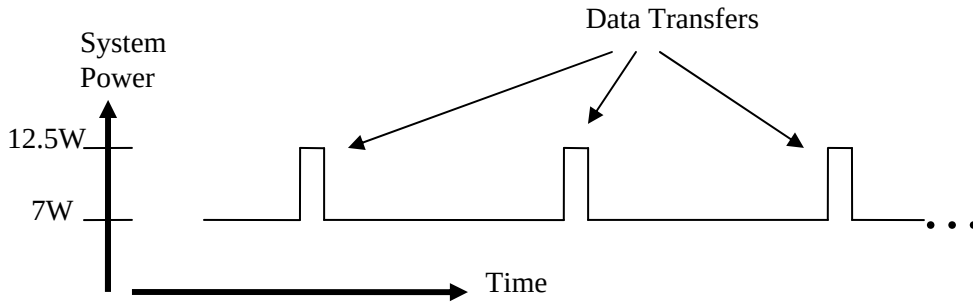


Figure C-3. High Speed Device Data Transfer

With a SuperSpeed interface the average percentage of time data is being actively transferred is roughly 3% (10MBytes/s divided by 333MBytes/s effective bandwidth[1]). This results in an average system power of (3%)(13W) + (97%)(7W) = 7.2W.

With a High Speed interface the average percentage of time data is being actively transferred is roughly 25% (10MBytes/s divided by 40MBytes/s effective bandwidth[1]). This results in an average system power of (25%)(12.5W) + (75%)(7W) = 8.4W.

For a 10MBytes/s device the system power difference is roughly 1.2W. This is a substantial percentage of system power. For larger bandwidth devices this power difference is larger.

Different system and device implementations can have different power characteristics, resulting in different system power impact tradeoffs. However, as the above example illustrates, the interface choice of one single device in a computer system can have a very substantial impact on system power consumption, and battery life in mobile systems.

[1] 333MBytes/s and 40MBytes/s are based on 67% of raw bit rate for SuperSpeed and High Speed interfaces respectively. Based on SuperSpeed protocol simulations 333MBytes/s represents a conservatively low estimate.

C.3 Link Power Management Implementation Guidelines

Power savings resulting from the effective use of link PM can have a significant impact on system power consumption. For example, not using link PM can decrease the average battery life of a typical notebook computer by 10%.[2] Other applications can benefit also.

The following topics are discussed in this section:

- Recommendations for device initiated U1 and U2 entry
- Recommendations for U1 and U2 inactivity timeouts

Both devices and hub downstream ports may initiate U1 and U2 entry. Hub downstream ports have inactivity timers used for initiating U1 and U2 entry, with timeouts programmed by software. Devices have additional information available which can be used to initiate U1 or U2 more intelligently than simple inactivity timers.

C.3.1 Recommendations for Device Initiated U1 and U2 Entry

All of a device's endpoints need to be considered in determining if and when a device should place its link in U1 or U2. The approach discussed herein is based on a combination of detecting when non-periodic endpoints are idle, and predicting whether periodic endpoints will need the link in the near future. If no endpoints within a device need the link then the link should be placed in U1 or U2.

The following subsections discuss specific criteria for determining when endpoints are idle or predicting whether the link will be needed in the near future. Which link PM state would be most appropriate to initiate (i.e., U1 or U2) is also discussed.

C.3.1.1 System Information Available To Devices

The following host controller and fabric information is available to all devices for use in determining when to initiate U1 or U2:

- Packet Pending flag.
- U1 and U2 System Exit Latency fields.

The Packet Pending flag is a bit field in Transaction Packets sent from the host to devices (see the Protocol chapter for more detailed information about this field). This field indicates whether any additional packets are pending for transfer on the host controller schedule for the associated endpoint. If there are no transfers pending on the host controller schedule for all of the endpoints in a device, then the device is idle as far as the host controller is concerned.

The U1 and U2 System Exit Latency fields are device fields programmed by software (see the SET_SEL command in the Framework chapter for more detailed information about these fields). These fields indicate the worst case latency to transition all links in the path between the device and host from U1 or U2 to U0, when this transition is initiated by the device. This transition latency is different when initiated by the device versus by the host due to the link PM behavior of hubs. For

[2] Assumes two SuperSpeed devices in U2 each with roughly 250mW of savings per PHY, two PHYs per link, and two links in the platform for a one watt potential savings to an average platform power of roughly 10W while running the Mobile Mark Office Productivity suite.

upstream flowing traffic hubs transition their upstream port to U0 in parallel with the transition on a downstream port. For downstream flowing traffic hubs first receive the packet, decode which downstream port the packet is targeted for, and then begin to transition the targeted downstream port to U0.

C.3.1.2 Control Endpoints

Control endpoints are generally used for device configuration and status purposes, e.g. during enumeration. Control transfers are highly tolerant of link exit latencies, and it is recommended to enable use of U1 and U2 (see Section TBD) for when a control endpoint is idle. For control endpoints, prediction of whether the endpoint will need the link in the near future is based on whether the endpoint is idle.

Once a control transfer is completed the endpoint may be considered idle for purposes of link PM. After this idle condition is met, if all other device endpoints are idle then the device is recommended to initiate U1 after a small link inactivity timeout, e.g. in the 10us range. Once in U1, U2 should be entered after PORT_U2_TIMEOUT (see Section C.1.2.3 for PORT_U2_TIMEOUT recommendations, and the Hub and Link chapters for further information about the PORT_U2_TIMEOUT field and the direct U1 to U2 transition process).

C.3.1.3 Bulk Endpoints

Bulk endpoints are by definition tolerant of substantial link exit latencies. It is recommended to enable use of U1 and U2 (see Section TBD) for when a bulk endpoint is idle.

Determining when a bulk endpoint is idle depends on whether the device is capable of asynchronously requesting a USB transfer due to activity on its application side interface.

A Network Interface Card (NIC), for example, may asynchronously request the host controller to schedule a USB transfer in response to receiving some packets from its network interface. The host will establish an IN transfer that the device NRDYs, and the device may not send an ERDY for a long period (e.g., minutes) until network interface activity occurs. Since network interface activity can occur independently of any USB or host system activity, it represents an asynchronous activity.

A mass storage device, on the other hand, will only perform a USB transfer in response to a request from the host. Devices like this do not have application side activity independent from host system activity.

C.3.1.3.1 Devices That Do Not Asynchronously Request USB Transfers

The endpoint is idle if the following conditions are met:

- There are no transfers pending in the Host Controller (HC) schedule for the endpoint, as indicated by the Packet Pending field.
- There is no application specific activity occurring on the device. An example of an application specific activity is a mass storage device that received a command that causes it to seek a particular track. The USB transfer (command) may have completed but the command caused device activity that is expected to be followed up on within a brief period of time. (Note that some commands may take longer times to complete, e.g. to spin up a drive's spindle, during which time a device may be able to temporarily place its link in a low power state.)

After these endpoint idle conditions are met, if all other device endpoints are either idle or do not need the link in the near future then the device is recommended to initiate U1 after a small link

inactivity timeout, e.g. in the 10us range. Once in U1, U2 should be entered after PORT_U2_TIMEOUT.

C.3.1.3.2 Devices That Asynchronously Request USB Transfers

Devices such as NICs will normally have a pending IN transfer in the host controller schedule that may be left pending for long periods (e.g. minutes) until there is data to transfer. As a result such endpoints are not able to utilize the Packet Pending field effectively in determining when endpoints are idle.

For these endpoints idle conditions are application specific. The U1 and U2 System Exit Latency fields may be used to set device inactivity timeouts more aggressively than those set in hub downstream ports. For some devices the hub U1 and U2 inactivity timers may be sufficient for initiating U1 and U2 entry.

Other devices may offer opportunities to be more aggressive with power management. For example, some wireless NICs have over-the-air protocols that let the NIC know it will not have any activity for periods in the hundreds of milliseconds because its partner is turning its radio off, in which case the NIC knows it is idle and can immediately place its link in U2.

C.3.1.4 Interrupt Endpoints

Interrupt endpoints may at times need to be serviced at a subscribed periodic rate, and at other times may be idle, e.g. after sending a NRDY. While an interrupt endpoint continues to transfer data in each consecutive interval it is operating in a periodic mode, but after a NRDY is sent the endpoint begins operating in a non-periodic mode until the next ERDY is sent. Many interrupt endpoints can be effectively treated as always operating in a non-periodic mode or notification mode, for example hubs, which generate interrupts due to port status change events (e.g., a connect or disconnect event).

C.3.1.4.1 Periodic Mode of Operation

While the endpoint is being serviced at a periodic rate, as soon as a transfer is completed for any given service interval the endpoint knows the link will not be needed for nearly an entire service interval period. If all other endpoints in the device are either idle or do not need the link in the near future, it is recommended for the device to immediately initiate either U1 or U2, depending on the length of the service interval (see Section C.3.1.8 for more discussion on this particular aspect).

Note that hosts are required to initiate interrupt transfers far enough ahead of a transfer window in order to meet subscribed service requirements. See the appropriate Host Controller specification for information about how the HC is informed how far ahead of time the transfers need to be sent (e.g., the Max Exit Latency field in device context for xHCI).

C.3.1.4.2 Non-periodic or Notification Mode of Operation

If the endpoint sends a NRDY in response to a request from the host, the endpoint is idle. If all other endpoints in the device are either idle or do not need the link in the near future, it is recommended for the device to immediately initiate either U1 or U2, depending on the length of the service interval (see Section C.3.1.8 for more discussion on this particular aspect).

After the device sends an ERDY on behalf of the endpoint, the link should be kept in U0 until the host sends a request in response to the ERDY (or until an error timeout occurs). (See the USB 3.0 specification for error conditions under which the host does not respond to an ERDY.)

C.3.1.5 Isochronous Endpoints

Isochronous endpoints are always serviced at a subscribed periodic rate. As soon as a transfer is completed for any given service interval the endpoint knows the link will not be needed until the next service interval. The Last Packet flag (see Protocol chapter) indicates when a transfer has completed for any given interval. If all other endpoints in the device are either idle or do not need the link in the near future, it is recommended for the device to immediately initiate either U1 or U2, depending on the following:

- The length of the service interval (see Section C.3.1.9 for more discussion on this particular aspect).
- The exit latency for the entire path of links between host and device when exit is initiated by the host. (See Section C.5 for example exit latency calculations.)
- The amount of time to transfer data within a given service interval.

To determine whether to use U1 or U2, define X to equal the service interval length minus the amount of time to transmit the subscribed amount of data (bytes) within a given interval. X is roughly the amount of time that the link will be idle in each interval, ignoring e.g. host controller scheduling variances, etc. This is illustrated in Figure C-4. If the U1 exit latency (for the entire path of links between host and device when exit is initiated by the host) is less than roughly 50% of X, then it is recommended for U1 to be initiated after each transfer within a given interval. If the U2 exit latency is less than roughly 50% of X, then use of U2 is recommended instead of U1.

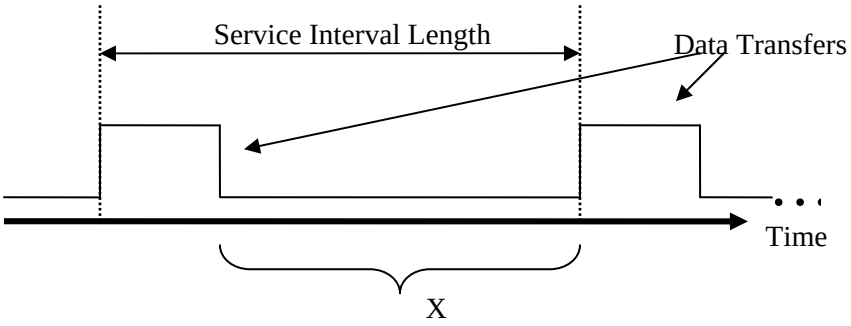


Figure C-4. Selection of U1 vs. U2 With Isochronous Endpoints

Note that hosts are required to send a ping message far enough ahead of a transfer window in order to meet subscribed service requirements. See the appropriate Host Controller specification for information about how the HC is informed how far ahead of time the transfers need to be sent (e.g., the Max Exit Latency field in device context for xHCI).

If a device needs timestamp information the device transitions its link to U0 prior to a bus interval boundary. The device must track when an appropriate bus interval boundary will occur, and in part can use the information programmed with the SET_SEL request to determine when to begin this transition to U0 in order to minimize the duration in U0.

For many devices timestamps are not needed more often than data is delivered. For such devices one power efficient way to receive timestamps is to first allow that a timestamp may be received in

between a ping and the data, and if a timestamp is not received then place the link in U0 just prior to the next bus interval. Recommendations for U1 and U2 Inactivity Timeouts in Hubs

Hub downstream ports are programmed by software with inactivity timeouts for initiating U1 and U2 entry using the PORT_U1_TIMEOUT and PORT_U2_TIMEOUT feature selectors (see the Hub Chapter). What to set these inactivity timeouts to depends on the types of devices (endpoints) hierarchically connected under them.

The following subsections offer recommendations for whether U1 or U2 should be enabled, and if so what an appropriate inactivity timeout value should be, based on endpoint type. For any given downstream port, the largest value across all device endpoints hierarchically under the port should be chosen.

The timeout value recommendations herein are guidelines. An optimal timeout value depends on device class, system implementation, and device implementation.

For most endpoint types the recommendations offered herein are based on setting the inactivity timeout to a multiple of the latency to transition links to U0. This latency loosely represents the "cost" or performance impact of putting a link into a low power state.

Some of the following subsections classify specific timeout values as aggressive, moderate, or lax. These are qualitative classifications. Some equipment such as cell phones may choose to use more aggressive policies in order to save more power. Other equipment such as desktop computers may choose to use lax policies.

C.3.1.6 Recommendations for Control Endpoints

Control endpoints are highly tolerant of link exit latencies. It is recommended to enable both U1 and U2 inactivity timeouts for control endpoints.

The inactivity timeouts are recommended to be set to three times the exit latency for the total path of links between device and host. More specifically:

- The PORT_U1_TIMEOUT is recommended to be set to three times the U1 System Exit Latency field.
- The PORT_U2_TIMEOUT is recommended to be set to three times the U2 System Exit Latency field.

A multiple of three represents an aggressive policy, but not so aggressive that the link will be constantly going into U1 or U2 in between successive control transfers, e.g. during device enumeration and initialization.

C.3.1.7 Recommendations for Bulk Endpoints

Bulk endpoints are generally tolerant of high link exit latencies when idle. It is recommended to enable both U1 and U2 inactivity timeouts for bulk endpoints.

The inactivity timeouts are recommended to be set to five to ten times the exit latency for the total path of links between device and host. More specifically:

- The PORT_U1_TIMEOUT is recommended to be set to five to ten times the U1 System Exit Latency field.
- The PORT_U2_TIMEOUT is recommended to be set to five to ten times the U2 System Exit Latency field.

A multiple of five represents a moderate policy. A multiple of ten represents a lax policy. In some applications it may be appropriate to use an aggressive multiple of three.

C.3.1.8 Recommendations for Interrupt Endpoints

Whether to enable U1 or U2 inactivity timeouts for an interrupt endpoint depends on the following:

- The subscribed interval period.
- The exit latency for the entire path of links between device and host.
- The nature of the interrupt application, i.e. whether it uses a periodic mode of operation as indicated by the Usage Type bits in the Endpoint Descriptor bmAttributes field (see Framework Chapter) (also see Section C.3.1.4 for a description of periodic mode of operation).

If the Usage Type is Periodic, then the same criteria as for isochronous endpoints should be used in determining whether U1 or U2 should be enabled, and in determining what the inactivity timeouts should be set to.

If the Usage Type is Notification, then the endpoint is fairly tolerant of link exit latency and interrupts will occur relatively infrequently. In this case it is recommended to enable the U1 inactivity timeout.

Whether to also enable the U2 inactivity timeout depends on how long the subscribed service interval is, or more accurately how much latency the endpoint can tolerate for the host to initiate the transfer, compared to the exit latency for the entire path of links between host and device. For interrupt IN endpoints the exit latency of concern is for when exit is initiated by the device. For interrupt OUT endpoints the exit latency of concern is for when exit is initiated by the host. If the interrupt endpoint service requirements allow for it to tolerate the U2 exit latency, then it is recommended for the U2 inactivity timeout to be enabled. If enabled, the inactivity timeouts are recommended to be set the same as the control endpoints.

Note that for interrupt endpoints it is appropriate for devices to initiate U1 or U2 very quickly after a transfer is completed for any given service interval (see Section C.3.1.4). The inactivity timeouts should generally be used as a backup in this case.

C.3.1.9 Recommendations for Isochronous Endpoints

Whether to enable U1 or U2 inactivity timeouts for an isochronous endpoint, and what to set the timeouts to, depends on the following:

- The subscribed interval period.
- The exit latency for the entire path of links between host and device when exit is initiated by the host. (See Section C.5 for example path exit latency calculations.)

As described in Section C.3.1.5, instead of using U1 or U2 inactivity timeouts, it is far more efficient for isochronous devices to initiate U1 or U2 immediately after a data transfer is completed for any given interval. The below recommendations assume isochronous devices adhere to the recommendations offered in Section C.3.1.5.

Typically an isochronous transfer will occur in every service interval, and the device may put its link in U1 or U2 right after a transfer is completed in each interval. Under these circumstances the inactivity timeouts will only be used when service intervals are skipped, i.e. when an application does not need to transfer data in one or more consecutive intervals, or when some sort of error condition interrupts an application's data transfers.

The U1 and U2 inactivity timeouts can be enabled and used to place links in lower power states under these circumstances, with proper choice of timeout values. The timeout values should be chosen to place a link in a lower power state after it can be determined that an entire interval has been skipped.

For U1 it is recommended that the PORT_U1_TIMEOUT be set to the larger of the following values:

- Five to ten percent longer than the service interval period. (The additional five to ten percent is chosen only to provide some design margin. For endpoints that transfer larger amounts of data within a given interval, less margin is necessary.)
- Twice the U1 exit latency for the entire path of links between host and device when exit is initiated by the host. (See Section C.5 for example path exit latency calculations.)

For U2 it is recommended that the inactivity timeout be disabled unless the device (under the hub port) is enabled to initiate U2 (see Section C.3.1.5 for a discussion of whether to enable U2 for an isochronous device). The rationale behind this is if the U2 inactivity timeout is enabled but the device does not normally initiate U2 in between successive isochronous service intervals, then the U2 path exit latency will still need to be used by the HC in determining when to send ping messages, and basing that off of the U2 exit latency when U1 is normally used will cause the link to stay in U0 (instead of U1) far longer than necessary, such that it is simply better to disable U2 altogether.

If U2 is enabled to be initiated by the device (per Section C.3.1.5), then it is recommended that the PORT_U2_TIMEOUT be set to the larger of the following values:

- Five percent longer than the service interval period. (The additional five percent is chosen only to provide some design margin.)
- Twice the U2 exit latency for the entire path of links between host and device when exit is initiated by the host. (See Section C.5 for example path exit latency calculations.)

C.3.1.10 Hub Downstream Ports Connected to another Hub

The PORT_U1_TIMEOUT and PORT_U2_TIMEOUT associated with hub upstream ports should be equal to or larger than the respective maximum values across all of the hub's downstream ports. A minor complication of this is the fact that these timeouts on a hub's upstream port are actually set within its link partner, which is really the downstream port of another hub (or root hub). As a result, when programming these inactivity timeouts for hub downstream ports, whether the downstream port is connected to another hub below needs to be considered.

C.4 Device Power Management Implementation Guidelines

C.4.1 Deciding When to Initiate Selective Suspend

Devices or functions that are idle or not in use should be placed in selective suspend. (See the IAD ECN to the USB 2.0 Specification, available from the USB SIG web site, for a definition of the term function as it is used in this Section.)

The following subsections first give some relevant background information about USB 3.0, and then offer implementation recommendations.

C.4.1.1 Background Information

USB 3.0 SuperSpeed has some enhancements in comparison to USB 2.0 that facilitate more aggressive use of selective suspend. These enhancements reduce the amount of time it takes to enter and exit selective suspend, and include the following features:

- Function level suspend and remote wake messaging. A function can enter and exit selective suspend independently of whether it is part of a composite device, and independently of other functions even when it is part of a composite device.
- Reduced overhead in suspend entry and exit processes. Once software receives an interrupt due to a remote wake message, software can immediately access the function. If woken by remote wakeup, no intermediate hub port status change events will be generated and so do not need to be cleared, helping to minimize the overall resume latency.

C.4.1.2 Implementation Recommendations

The simplest way to decide when to place a function/device into selective suspend is based on an inactivity timeout implemented in software, e.g. in the function driver. An inactivity timeout should be chosen as a function of the following things:

- The hardware level latency to both suspend and resume the function/device. This is primarily dependent on function/device implementation. Hardware suspend latency is typically small, on the order of a handful of microseconds, in comparison to the resume latency for USB 3.0 SuperSpeed. Typical hardware level resume latencies are on the order of a handful of milliseconds.
- The software level resume latency. This is heavily dependent on function/device (driver) software implementation and system software implementation. For functions/devices that retain all state during selective suspend (i.e. do not need to save / restore any state from / to the function/device), typical software level suspend and resume latencies are each on the order of a handful of milliseconds.
- How much power is saved by placing the function/device in selective suspend.
- How much performance is impacted by placing the function/device in selective suspend.

The inactivity timeout is recommended to be chosen in the range of ten to one hundred times the total (hardware + software) resume latency, depending on the power and performance impacts. For devices that use selective suspend very dynamically, a timeout of 100ms or less is recommended. For devices that do not use selective suspend very dynamically, a timeout of 1s or less is recommended.

Minimizing a function's suspend exit latency has a substantial impact on how aggressively selective suspend can be used. Another important factor is minimizing power consumption while in suspend because the function is likely to be in suspend the vast majority of the time in the overall operational life of the device. In making tradeoffs between saving more power during suspend versus decreasing resume latency, a balance needs to be struck. For battery powered equipment the more important factor is likely to be minimizing suspend mode power. One mechanism for helping accomplish this is discussed in Section C.4.2.

C.4.2 Latency Tolerance Messaging (LTM) Implementation Guidelines

Computer systems typically maintain a high state of readiness to service devices even when the computer is idle. LTM provides a mechanism for a system to reduce its state of readiness with the cooperation of USB 3.0 devices, resulting in substantial system power savings without requiring additional cost to devices. See section C.1.5 for an overview of LTM.

This section provides device implementation guidelines for LTM support. The algorithm described herein is based on a model using two device Latency Tolerance states, an active state and an idle state. Each state has a different Best Effort Latency Tolerance (BELT). Other implementations are also possible, for example using a three LT-state model.

C.4.2.1 Device State Model and State Transition Algorithm

Two device Latency Tolerance states (LT-states) are defined: LT-idle and LT-active. In the LT-idle state a device is idle and can tolerate a larger latency from the system. In the LT-active state a device is either actively transferring data with the host or anticipating a need for the system to service the device with a shorter latency.

The recommended BELT value for the LT-idle state is tBELTdefault (see the Protocol chapter). The recommended BELT value for the LT-active state is tBELTmin (see the Protocol chapter). Other BELT values may be used and are dependent on platform (system) implementation, however these implementation guidelines assume the above recommended values are used.

When a device transitions from LT-idle to LT-active it sends an LTM Transaction Packet (TP) with a BELT of tBELTmin. When a device transitions from LT-active to LT-idle it sends an LTM TP with a BELT of tBELTdefault.

The default LT-state of a device after reset is LT-idle. For all devices the default BELT assumed by the system is tBELTdefault.

C.4.2.1.1 Transitioning From LT-idle to LT-active

Devices transition from LT-idle to LT-active when the device determines or predicts a sustained data transfer is to occur in the near future (i.e. within the last reported BELT). This determination may be done in different ways depending on the nature of the device and also on circumstances. Some examples are given below:

- The host initiates a bulk OUT transfer with the Transfer Pending flag asserted to a flash drive device. As a result of receiving this OUT request, the device should predict that a sustained transfer has been initiated by the host, transition to LT-active, and send an updated BELT to the host.
- The host initiates a bulk IN transfer with the Transfer Pending flag asserted to a hard disk drive (HDD) device with is spindle currently spun down and the requested data not in a cache on the HDD. The device should predict that a sustained transfer has been initiated by the host. However the device will service the IN request after its spindle spins up, which may take substantially longer than the last reported BELT. When the device determines that the spindle will complete its spin up within the last reported BELT, the device transitions to LT-active and sends an updated BELT.
- A NIC device begins receiving data on its network interface that requires a bulk IN data transfer with the host. The device should predict that a sustained transfer to the host will be needed. The device should transition to LT-active, transition its link to U0 (if not already in U0), and send both an ERDY and an updated BELT to the host.
- The host initiates an IN transfer to a device's default control endpoint, e.g. a GetStatus request. A device should not predict a sustained data transfer is to occur in the near future based on this IN request.

C.4.2.1.2 Transitioning From LT-active to LT-idle and Selection of LT-timeout

Devices transition from LT-active to LT-idle when the device determines or predicts that no transfers will occur within the near future. One method for determining this condition is an inactivity timeout LT-timeout. If no packets are transmitted or received by a device for LT-timeout, and no future sustained data transfer is determined or predicted, then the device transitions to LT-idle and sends an updated BELT to the host. A software programmable LT-timeout is recommended, programmed by the function (device) driver.

The LT-timeout value should be determined based on the following:

- The difference in system power consumption between when devices report a BELT of tBELTdefault versus tBELTmin. This system power consumption difference is dependent on system implementation.
- The performance impact resulting from the device reporting a BELT of tBELTdefault. This is dependent on device and system implementations.
- If the U2 inactivity timeout is enabled on a device's link, LT-timeout must be shorter than PORT_U2_TIMEOUT (see Hub chapter). This is needed to satisfy the requirement that U2 not be entered unless a device's last reported BELT is greater than or equal to tBELTdefault (see Protocol chapter).

Selecting an LT-timeout value essentially represents making a power versus performance tradeoff. The power benefit of devices reporting a BELT of tBELTdefault may be derived from a CPU and other system components entering lower power states, i.e. allowing the system to enter deeper system sleep states. These deeper system sleep states may result in a far larger power benefit than for U2, depending on system implementation, warranting a more aggressive inactivity timeout than for U2.

The LT-timeout value is typically chosen as a multiple of the exit latency "cost" associated with the LT-idle state BELT value. A multiple of one to two (i.e. an LT-timeout of tBELTdefault to twice tBELTdefault) is typically chosen for battery powered equipment such as notebook computers. A multiple of five to ten is typically chosen for equipment not powered by batteries.

C.4.2.2 Other Considerations

The following are additional considerations associated with device support of LTM:

- The BELT represents a latency tolerance for an entire device. The BELT value must be aggregated across all endpoints within the device, including all functions within a composite device. The smallest BELT value across all endpoints should be selected. For the purpose of determining the aggregated BELT, a BELT of tBELTdefault should be associated with isochronous endpoints.
- If a device needs to send an ERDY to request the host schedule a transfer, the total latency for the device to receive the transfer request will include both a link PM state exit latency and a system sleep state exit latency. If a device will not operate reliably when both U2 and LTM are enabled, but will operate reliably when only one of U2 and LTM are enabled, it is recommended that priority be given to enabling LTM for platforms that support LTM because the power savings will be greater (assuming a platform would not implement LTM support unless there were a benefit to doing so).
- If LTM is supported by a function, LTM should be disabled prior to placing the function into function suspend (see the FUNCTION_SUSPEND feature selector in the Framework chapter) and prior to placing a device into suspend (see the PORT_LINK_STATE feature selector in the Hub chapter). Devices send an updated LTM when LTM is enabled or disabled, as defined in the Protocol chapter. Disabling LTM in this way ensures that a suspended device does not keep the system in a high state of readiness, wasting power.

C.4.3 Removing Power From Portions of a Device During Suspend

In some applications a device may remain powered during suspend. It is important to minimize power consumption in this mode. The USB 3.0 Specification allows for devices to remove power from portions of the device during suspend. For example, the Low Frequency Periodic Signal (LFPS) handshake process for wakeup (U3 exit) gives sufficient time for a device to turn on a power supply or switched power rail (see the PHY chapter for LFPS handshake timing).

Generally a device needs to create its own switched power rail from VBUS, power some circuitry off VBUS and power the rest of its circuitry off the switched power rail. In some applications such as devices on PC motherboards, a switched power rail may already be available to devices – essentially for free in terms of device Bill Of Material cost. In the following subsections the general case is described first, and then a specific case for PC motherboard applications is given. This is followed by a discussion of what needs to remain powered during suspend.

C.4.3.1 Devices Which Must Create Their Own Switched Power Rail

As a point of reference, Figure C-5 illustrates the power rails for a generic bus powered device. Only one power rail (VBUS) is included.

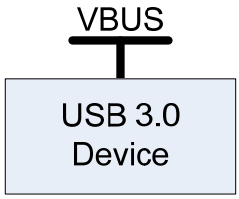


Figure C-5. Power Rails For Generic Bus Powered Device

Figure C-6 illustrates a bus powered device that creates a new switched power rail called Vmain. Vmain is switched off while the device is in suspend. A simple power switch (FET) is illustrated, however a more complex implementation may instead turn on or off a Switch Mode Power Supply (SMPS) controller that would generate Vmain at a different voltage level(s) than VBUS.

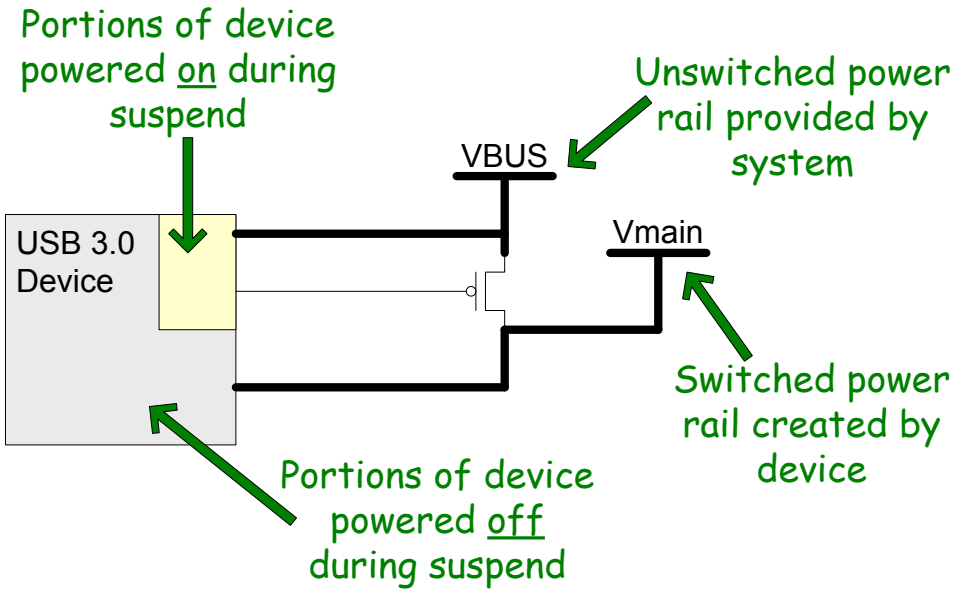


Figure C-6. Power Rails For Bus Powered Device With Switched Rail

A device implementation may also leave Vmain on for selective suspend, but shut off Vmain for global suspend. This would be configured by the device/function driver as part of the suspend process.

C.4.3.2 Devices Which Have a Switched Power Rail Provided by System

Figure C-7 illustrates a device that uses the power rails offered by a PC system board. Vaux is always on while a power source is available (e.g., from batteries or an A/C wall outlet), including while the computer is in suspend (i.e., S3 or S4 ACPI states). Vmain is switched off while the computer is in suspend (i.e., S3 or S4).

When the device is placed in selective suspend, Vmain may remain on. As a result suspend mode device power consumption off Vmain still needs to be minimized in order to meet the USB 3.0 Specification, though this does not diminish additional device power savings that occur when Vmain is shut off.

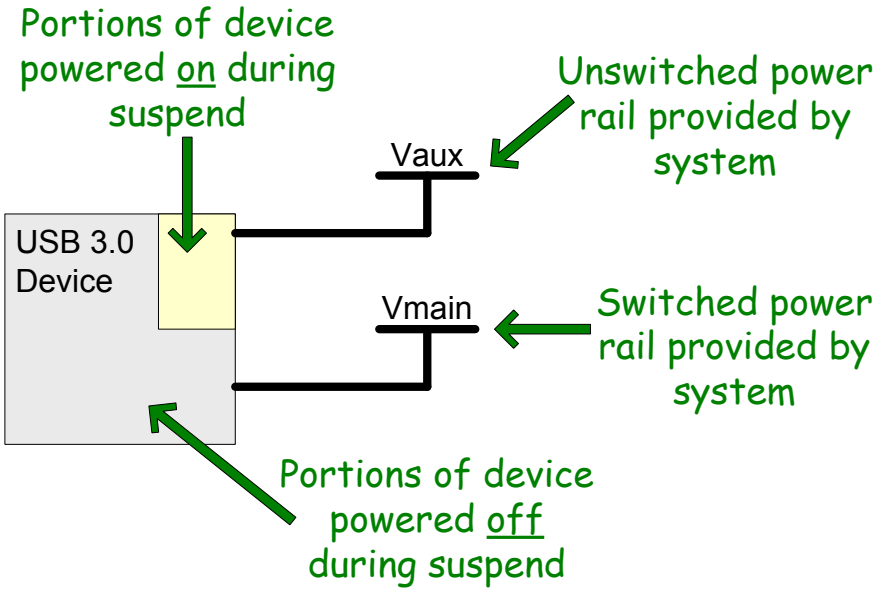


Figure C-7. Power Rails For PC Application

C.4.3.3 Self Powered Devices

Figure C-8 illustrates a self powered device that creates a switched power rail called Vmain. Vmain is switched off while the device is in suspend. The power supply within the device is turned on or off by logic that remains powered off VBUS during suspend. In this example VBUS is created by the system, however some devices may benefit from an alternate implementation where the device supplies its own power during suspend. Devices that support On The Go (OTG) may also require a more complex implementation.

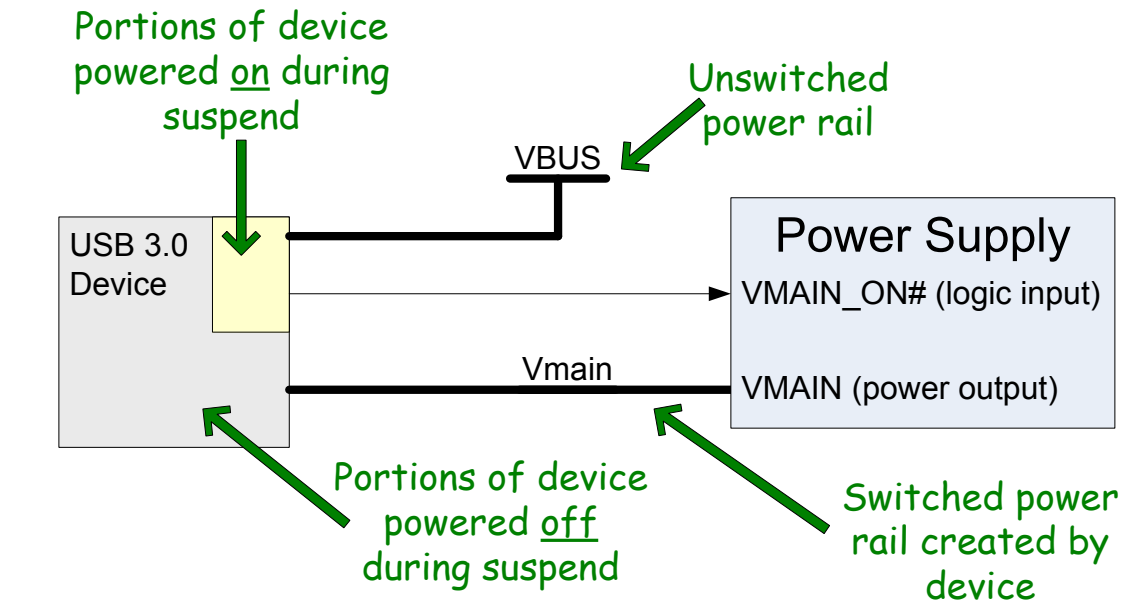


Figure C-8. Power Rails For Self Powered Device

C.4.3.4 Identifying What Needs To Remain Powered During Suspend

The amount of circuitry powered during suspend should be minimized in order to save power. What exactly needs to remain powered is heavily application dependent. The following list may help identify the circuitry that needs to remain powered during suspend:

- Circuitry to detect wakeup signaling from the host.
- Circuitry to detect reset signaling from the host.
- Circuitry to switch on/off power to the portions of the device that are powered off during suspend.
- Circuitry to send remote wakeup signaling to the host, if the device is capable of remote wakeup and the device is enabled for remote wakeup.
- The necessary state of the circuitry powered off during suspend. This state may need to be stored in some form of non-volatile storage (either on the device itself or external to the device, e.g. using a service from operating system software) during the suspend process, and then restored during the resume process.
- Application specific circuitry.

C.5 Calculation Examples for U1 and U2 System Exit Latency Fields

This section provides examples of how to calculate the exit latency (i.e., time to transition from a non-U0 state to a U0 state) for a total path between a device and the host. Examples are given for both device initiated exit and host initiated exit.

Example calculations for U1 System Exit Latency (U1SEL) and U2 System Exit Latency (U2SEL) fields are also included. These are associated with device initiated transitions from non-U0 to U0.

Figure C-9 depicts a USB 3.0 hierarchy and calls out the relevant parameters used in the calculations that follow.

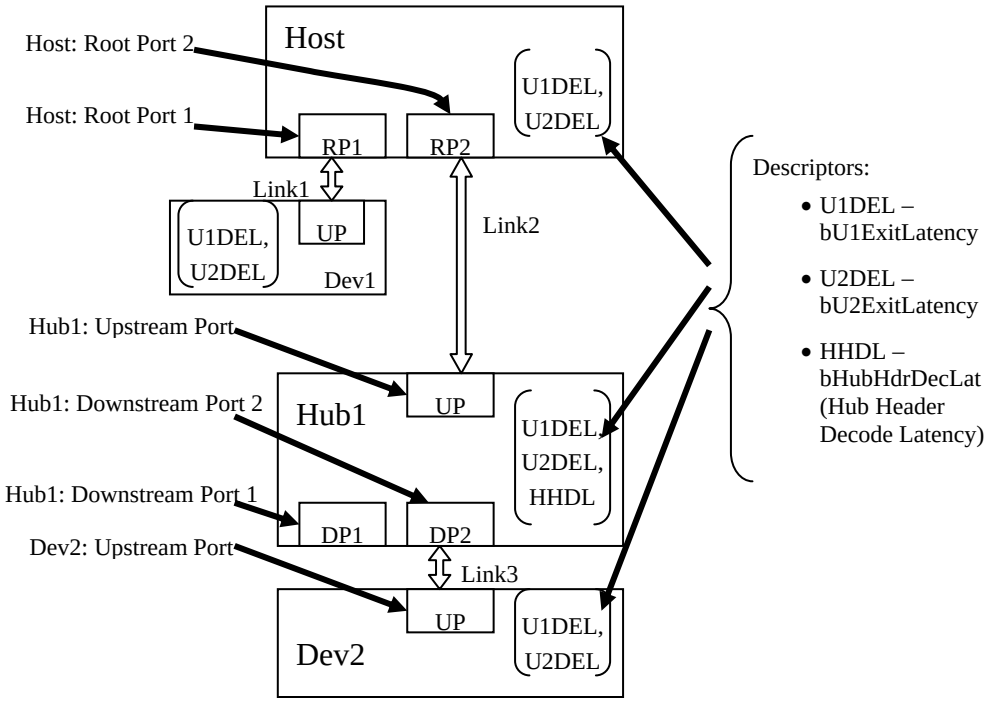


Figure C-9. Host to Device Path Exit Latency Calculation Examples

C.5.1 Path 1: Device Connected Directly to Host, Downstream Direction

An example calculation of the total latency to transition Link1 from U1 to U0, as initiated by the host due to a need to send a downstream flowing packet, is given below.

If U1 is enabled in either Dev1 or RP1, then
  Link1 U1 to U0 transition latency:
    Link1_EL = Max(Dev1:U1DEL, RP:U1DEL)
else
  Total path exit latency: 0

An example calculation of the total latency to transition Link1 from U2 to U0, as initiated by the host due to a need to send a downstream flowing packet, is given below. Note that if U2 is not enabled for the link then the transition latency is zero regardless of whether U1 is enabled.

If U2 is enabled in either Dev1 or RP1, then
  Link1 U2 to U0 transition latency:
    Link1_EL = Max(Dev1:U2DEL, RP:U2DEL)
  Total path exit latency: Link1_EL
else
  Total path exit latency: 0

C.5.2 Path 2: Device Connected Directly to Host, Upstream Direction

This calculation is the same as for the downstream direction. U1SEL and U2SEL are equal to the respective total path exit latency calculation.

C.5.3 Path 3: Device Connected to Hub That is Connected to Host, Downstream Direction

An example calculation of the total latency to transition Link2 and Link3 from U1 to U0, as initiated by the host due to a need to send a downstream flowing packet, is given below. Note that if U1 is disabled for Link3 then the transition latency is zero for the entire path.

If U1 is enabled in either Dev2 or Hub1-DP2, then
  Link3 U1 to U0 transition latency:
    Link3_EL = Max(Dev2:U1DEL, Hub1:U1DEL)
  Latency for hub to receive and decode packet header and initiate U1 to U0 transition on the appropriate downstream port: HHDL
  Link2 U1 to U0 transition latency:
    If U1 is enabled in either Hub1-UP or RP2, then
      Link2_EL = Max(Hub1:U1DEL, RP:U1DEL)
    else
      Link2_EL = 0
  Total path exit latency: Link2_EL + HHDL + Link3_EL
else
  Total path exit latency: 0

A timing diagram depicting the above process is provided in Figure C-10. The diagram assumes U1 is enabled on each link. The above calculation assumes the Host Scheduling Delay (i.e., the delay by the host to begin transmitting the packet after the link completes the transition to U0) shown in the diagram is zero.

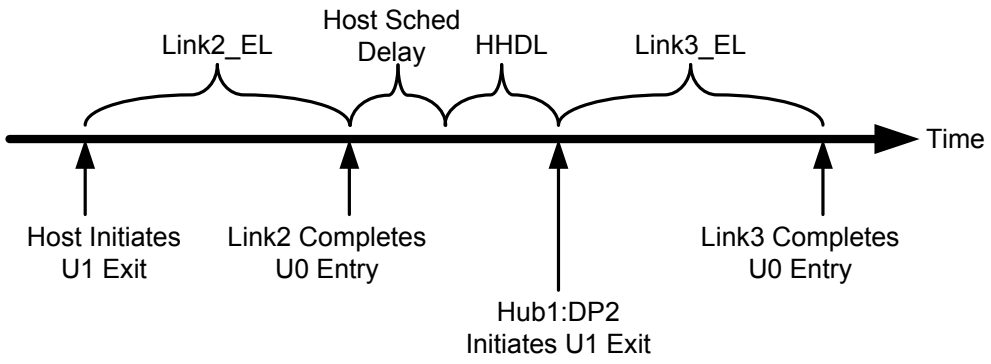


Figure C-10. Downstream Host to Device Path Exit Latency with Hub

An example calculation of the total latency to transition Link2 and Link3 from U2 to U0, as initiated by the host due to a need to send a downstream flowing packet, is given below. Note that for Link3 if U2 is disabled then the transition latency is zero regardless of whether U1 is enabled. However if U2 is enabled for Link3 then Link2 may be disabled for U2 but enabled for U1 and its U1 exit latency may contribute to the total path U2 exit latency.

```
If U2 is enabled in either Dev2 or Hub1-DP2, then
  Link3 U2 to U0 transition latency:
        Link3_EL = Max(Dev2:U2DEL, Hub1:U2DEL)
  Latency for hub to receive and decode packet header and initiate U2 to U0 transition on the
  appropriate downstream port: HHDL
  Link2 U2 to U0 transition latency:
        If U2 is enabled in either Hub1 or RP2, then
              Link2_EL = Max(Hub1:U2DEL, RP:U2DEL)
        else if U1 is enabled in either Dev1 or RP1, then
              Link2_EL = Max(Hub1:U1DEL, RP:U1DEL)
        else
              Link2_EL = 0
  Total path exit latency: Link2_EL + HHDL + Link3_EL
else
  Total path exit latency: 0
```

C.5.4 Path 4: Device Connected to Hub That is Connected to Host, Upstream Direction

An example calculation of the total latency to transition Link3 and Link2 from U1 to U0, as initiated by device Dev2 due to a need to send an upstream flowing packet, is given below. Note that if U1 is disabled for Link3 then the transition latency is zero for the entire path. U1SEL is equal to the total path exit latency calculation.

```
If U1 is enabled in either Dev2 or Hub1-DP2, then
  Link3 U1 to U0 transition latency:
        Link3_EL = Max(Dev2:U1DEL, Hub1:U1DEL)
  Link2 U1 to U0 transition latency:
        If U1 is enabled in either Hub1-UP or RP2, then
              Link2_EL = Max(Hub1:U1DEL, RP:U1DEL)
        else
              Link2_EL = 0
  Total path exit latency: Max(Link3_EL, Link2_EL + tHubPort2PortU1ExitLat)
else
  Total path exit latency: 0
U1SEL = total path exit latency
```

See the Hub chapter for information about tHubPort2PortExitLat.

A timing diagram depicting the above process is provided in Figure C-11. The diagram assumes U1 is enabled on each link.

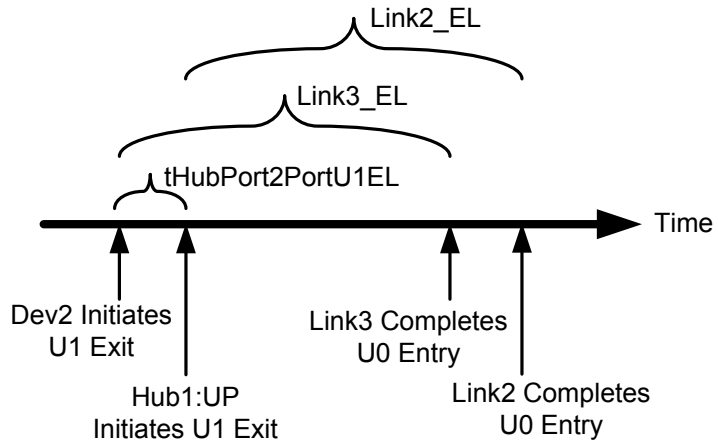


Figure C-11. Upstream Device to Host Path Exit Latency with Hub

An example calculation of the total latency to transition Link3 and Link2 from U2 to U0, as initiated by device Dev2 due to a need to send an upstream flowing packet, is given below:

```
If U2 is enabled in either Dev2 or Hub1-DP2, then
  Link3 U2 to U0 transition latency:
        Link3_EL = Max(Dev2:U2DEL, Hub1:U2DEL)
  Link2 U2 to U0 transition latency:
        If U2 is enabled in either Hub1 or RP2, then
              Link2_EL = Max(Hub1:U2DEL, RP:U2DEL)
        else if U1 is enabled in either Dev1 or RP1, then
              Link2_EL = Max(Hub1:U1DEL, RP:U1DEL)
        else
              Link2_EL = 0
  Total path exit latency: Max(Link3_EL, Link2_EL + tHubPort2PortU2ExitLat)
else
  Total path exit latency: 0
U2SEL = total path exit latency
```

See the Hub chapter for information about tHubPort2PortExitLat.

电子发烧友
www.elecfans.com

“电子发烧友”网站是一个面向广大电子爱好者、大专院校学生、中小型企业工程技术人员的电子技术应用、推广专业网站。主要内容有：电子技术应用交流，器件资料、电子设计软件下载，电子技术支持服务，电子产品发布、转让和引进等信息。

本资料由"电子发烧友"网站收集整理，版权属原作者

在使用本资料或软件时，有什么问题，欢迎到“电子发烧友”网站内的 BBS“技术论坛”中发表，我站的热心网友会帮助你的。

电子技术论坛 http://bbs.elecfans.com

需要更多的电子技术相关资料或软件，欢迎到“电子发烧友”网站下载。

“电子发烧友”网站：http://www.elecfans.com