Embedded USB2 Version 2.0: Supplement to the USB 2.0 Specification (eUSB2V2)

Apple Inc.
HP Inc.
Intel Corporation
Microsoft Corporation
Renesas Corporation
STMicroelectronics
Texas Instruments

Version 1.0

August 2024

Release History

Version	Comments	Issue Date
1.0	Initial release	August, 2024

Please send comments to techadmin@usb.org.

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

Acknowledgement of Technical Contribution

The authors of this specification would like to recognize the following peo ple who participated in the eUSB2V2 Specification technical work group.

Apple Inc. – Promoter Company Employees			
Brian Baek	Scott Deandrea	William Ferry	Derek Iwamoto
Scott Jackson	Christine Krause	Jongbae Park	Dan Wilson
HP Inc. – Promoter Company Employees			
Isaac Lagnado			
Intel Corporation – Promoter Company Employees			
Jia Jun Lee	Sriram Balasubrahmanyam	Dmitriy Berchanskiy	Daniel Camacho
Huimin Chen	Sourabh Das	Sanjiv Desai	Chee Yong Ew
Venkataramani Gopalakrishnan	Ngek Leong Guok	John Howard	Liana Ilkova
Abdul Ismail	Timo Kaikumaa	Kishore Kasichainula	Raymond Law
Guobin Liu	Uma Medepalli	Aruni Nelson	Vivek Paranjape
Choon Gun Por	Rajaram Regupathy	Amit Srivastava	Karthi Vadivelu
Rafal Wielicki			
Microsoft Corporation – Promoter Company Employees			
Ramamohan Reddy Boreddy	Matt Chung	Philip Froese	Kianoosh Peyvan
Kiran Shastry			
Renesas Corporation – Promoter Company Employees			
Philip Leung	Kiichi Muto		
STMicroelectronics – Promoter Company Employees			
Nathalie Ballot	Indranil Bhattacharya	Richard O'Connor	Jean-Christophe PONT
Anubhav Tripathi			
Texas Instruments – Promoter Company Employees			
Mike Campbell	Win Maung	Anwar Sadat	Neeta Thawani
Deric Waters			

Contributor Company Employees			
Advanced Micro Devices	Ramakanth Akkenepalli	Michael Comai	Raul Gutierrez
	Will Harris	Jason Hawken	Calvin (Hon-Son) Li
	Preetesh Mahendra	Zeeshan Sarwar	Joseph Scanlon
	Niall Sorensen		
Allion Labs, Inc.	Howard Chang	Brian Shih	
Analogix Semiconductor, Inc.	Greg Stewart		
ASMedia Technology, Inc.	Howard Yang		
Cadence Design Systems, Inc.	Mayank Bakshi	Scott Bare	Marcin Behrendt
	Suhang Chen	William Chen	Pawel Eichler
	Thomas Faison	Sanjai Gandhi	Sandeep Grover
	Scott Huss	Kiran Kumar Indrakanti	Poonam Khatri
	Tomasz Klimek	Sanjeet Kumar	Jimmy Li
	Liang Li	Ming Li	Zhenwei Liu
	Andy Mauffet-Smith	Sakthivel Ramaiah	Mohamed Rasith Ali
	Loren Reiss	Arunkumar S	Alexander Sgouros
	David Shin	Fred Stivers	Abishek Subramaniyan
	Kamesh V	Dongdong Wang	Na Wang
	Jiang Wu	Luke Xiang	Marco Zavaroni
	Qian Zhao		
Canova Tech Srl	Andrea Maniero	Antonio Orzelli	Paolo Pilla
	Nicola Scantamburlo		
China Telecommunicatio n Technology Labs	Ming Wei	Weilong Zhan	
Diodes Incorporated	Jao-Ching Chen	Qun Song	Jin-sheng Wang
	Zhinan Zhou		
eEver Technology, Inc.	Chien-Cheng Kuo		
Egis Vision Inc.	Jason Fan	Jeff Lee	Joe Lee

Elka International Ltd.	Roy Ting	Jui-Ming Yang	
Ellisys	Chuck Trefts		
Foxconn	Jason Chou	Shruti Deore	Jon Jacoby
	Terry Little		
Genesys Logic, Inc.	Weddell Lee	Miller Lin	Ryan Lin
	Daniel Tan	Yihsun Wu	Jeffrey Yang
Google Inc.	David Desrosiers	Neil Hendin	Eric Herrmann
	Ryan Hou	Benson Leung	Prashant Malani
Granite River Labs	Nikhil Acharya	Velmurugan Ayyakkannu	Pascal Berten
	Sandy Chang	Bala M	Kristof Mommen
	Krishna Murthy	Vasudev Uttharahalli	
Himax Technologies Limited	Antonio Tsai		
Infineon Technologies	Godwin Gerald Arulappan	Jaeik Lee	Amogh Mahadik
	Venkat Mandagulathur	Venkata Sreenivasa Reddy Namala	Rajagopal Narayanasamy
Kandou Bus SA	David Stauffer		
Keysight Technologies Inc.	Mark Johnson	Biing Lin Lem	Eunice Lim
	Jit Lim	Pedro Merlo	Stephen Muller
	Abhijeet Shinde	Kevin Smith	Xiao Zhang
Kinetic Technologies, Inc.	Ramesh Dandapani	Jay Slivkoff	
Lattice Semiconductor Corp	Satheesh Chellappan		
LeCroy Corporation	Mike Engbretson	Mike Micheletti	
Luxshare-ICT	Daniel Chen	CY Hsu	John Lin
	Stone Lin	Alan Liu	Eric Wen
	Pat Young		
M31 Technology Corp.	Sam Chang	Frost Chen	Ellis Hsiao
	Lin-Chun Hsu	Johnsam Lin	Winson Tsai

	Karl Yao	Eason Yi	
MCCI Corporation	Terry Moore		
MediaTek Inc.	Henry Chen	Akan Lin	Alexyc Lin
	Tung-Sheng Lin	Chiachun Wang	
Microchip Technology Inc.	Richard Petrie		
Nexperia B.V.	Max Guan	Chin-Jui Lin	Stefan Seider
Nordic Semiconductor ASA	Tomasz Moń		
NXP Semiconductors	Ken Jaramillo	Abhijeet Kulkarni	Vijendra Kuroodi
	Bin Lin	Dong Nguyen	Sivakumar Papadasu
	Gabriel Sacripanti	Krishnan TN	Bart Vertenten
	Li Yaoqiao		
ON Semiconductor	Athar Ali	Atul Bhansali	Andrew Yoo
Parade Technologies, Inc.	Tom Burton	Reginald Conley	Eric Wittmayer
	Kevin Yuan	Jingfan zhang	
Qualcomm, Inc	James Goel	Madjid Hamidi	Raja Jagadeesan
	Adi Menachem	George Paparrizos	Richard Wietfeldt
Realtek Semiconductor Corp.	KaiYuan Chang	Chun-Chuan Chen	TsungYu Chen
	Chen Chien Hsun	Hsinliang Chiu	Wei Honggi
	Ming Ying Hsieh	Victor Hsu	Hsiang Huang
	Chunwei Kuo	YingLi Lee	ChihHan Li
	Ying Shun Lien	Terry Lin	Kevin Su
	Po-wei Tsou	Ying Jen Yen	
Resillion (Belgium NV)	Bart Grispen		
Shenzhen Icspring Technology Co., Ltd.	Ken You		
Siemens Industry Software Inc.	Laurence Davies	Ankit Garg	Kiran Kumar
	Manoj Manu	Sriram Bhaskar Teja Mundru	Tray Read
SiliConch Systems Private Limited	Shubham Paliwal		
Sonix Technology Co., Ltd.	Mickey Fan	Ming Chi Fan	Chen-ying Hsu

	Carl Kuo	Jay Liu	Mingfeng Wu
	P.J. Wu		
Specwerkz	Robert Dunstan	Steve McGowan	Brad Saunders
Sumitomo Electric Ind., Ltd.	Mitsuaki Tamura		
Sunplus Innovation Technology Inc.	WJ Kuo	Chien	G.T. Hsiao
	JK Hsu	Jiankai Huang	Y Lee
	Felix Lin	Janice Lin	KC Lin
	MJ Lin	Chuan-Yen Pan	Zonghan Wu
	KH Yu		
Synopsys, Inc.	Subramaniam Aravindhan	Chandrashekar B U	RadosÅ,aw Celmer
	Abhishek Chowdhary	Morten Christiansen	Manuel Duarte
	Arun Elango	Nivin George	Andrzej Grodzicki
	Eric Huang	Tejaswini Koppalkar	Behram Minwalla
	Saleem Mohammad	Balakrishna Nayak	Minh Pham
	Nitin Sharma	Michal Uszynski	
The Silanna Group Pty. Ltd.	Hubertus Notohamiprodjo	Tod Wolf	
UL LLC	Michael Hu	Terry Kao	Dylan Su
VIA Labs, Inc.	Terrance Shih		
VIA Technologies, Inc.	Fong-Jim Wang		
Vivo Mobile Communication Co., Ltd.	Benny Dong	Fangding Luo	Junchen Wei

Contents

Figures

Tables

1 Introduction

1.1 Background

Universal Serial Bus 1.0 was published in 1996. The USB 2.0 specification [1] was published in 2000. In total, there have been approximately 5 major upgrades/enhancements to USB technology to respond demands for more bandwidth and better connectors. As the USB technology becomes ubiquitous in every computing device, its adoption in embedded applications is limited, primarily due to the architectural framework, that is defined to work for all functions. When it comes to embedded applications, where only a specific device function may be required, the architectural philosophy of the USB technology becomes inefficient.

USB cameras are one example. With increasing user expectations of camera resolution and frame rates, the bandwidth constraints of USB 2.0 require today's cameras to aggressively apply compression to fit within its bandwidth limitations. And even with compression, next generation cameras will exceed the USB 2.0 bandwidth limitations.

If USB 3.x [2] technology is used to address the bandwidth need, it is not an optimal solution for these camera devices, because it is a symmetrical link offering the same bandwidth capacity to both downstream and upstream traffic. For a camera device, bandwidth requirement is asymmetrical. A USB 3.x camera is not an optimal solution for embedded applications.

This usage model reveals a technology gap in the platform I/O space yielding an inflection point of opportunity. The technology in this specification was developed to deliver a scalable I/O technology, specifically designed for inside-the-box (embedded) USB applications, optimizing for performance, power and cost. It supports a low-voltage, power efficient PHY with gigabit rates on a USB 2.0 architectural framework, i.e., half-duplex link and transactional protocol.

Note that this specification does not require support for backwards compatibility at the Physical and/or Protocol Level to either USB 2.0 [1] or eUSB2 [3], or how to switch to either USB 2.0 or eUSB2 native mode. However, this specification does define the methodology to support backwards compatibility at the Framework level.

1.2 Objective of the Specification

This **eUSB2 Version 2.0** (eUSB2V2) document defines a performance enhancement to the USB industry-standard, USB 2.0 [1] and eUSB2 [3] specifications. The USB 2.0 specification [1] describes the protocol definition, types of transactions, and bus management required to design and build systems and peripherals that are compliant with this specification. eUSB2V2 is primarily a performance enhancement to eUSB2 native mode to provide more bandwidth for peripherals, such as embedded cameras, by adding significantly higher data rates to USB 2.0, while maintaining a low-voltage electrical interface.

This specification refers to eUSB2V2 as a collection of features or requirements that apply to USB 2.0 and eUSB2 native mode operations. Additionally, where specific differences exist regarding the USB 2.0 and eUSB2 native mode definitions of features or requirements, those differences will be uniquely identified as eUSB2V2 features or requirements. Refer to Section 2.1 for details.

eUSB2V2's goal is to enable peripherals 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 and embedded 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.

1.3 Scope of the Document

The specification is primarily targeted at 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, eUSB2V2 peripherals must implement device framework commands and descriptors as defined in the USB 2.0 Specification. Peripherals operating at the eUSB2V2 speeds must implement the eUSB2V2 enhancements defined in this specification.

1.4 Related Documents

[1] USB Implementers Forum, Universal Serial Bus Specification Revision 2.0, 2000.

[2] USB Implementers Forum, Universal Serial Bus 3.2 Specification, 1.1 ed., 2022.

[3] USB Implementers Forum, Embedded USB2 (eUSB2) Physical Layer Supplement to the USB Revision 2.0 Specificiation, V 1.2, 2018.

1.5 Conventions

1.5.1 Precedence

If there is a conflict between text, figures, and tables, the precedence shall be tables, figures, and then text.

Note: The text, figures, and tables in this specification all contain necessary information for building an implementation and need to be read together to get a full understanding of eUSB2V2 technology.

1.5.2 Keywords

The following keywords differentiate between the levels of requirements and options.

1.5.2.1 Informative

Informative is a keyword that describes information with this specification that intends to discuss and clarify requirements and features as opposed to mandating them.

1.5.2.2 May

May is a keyword that indicates a choice with no implied preference.

1.5.2.3 N/A

N/A is a keyword that indicates that a field or value is not applicable and has no defined value and shall not be checked or used by the recipient.

1.5.2.4 Normative

Normative is a keyword that describes features that are mandated by this specification.

1.5.2.5 Optional

Optional is a keyword that describes features not mandated by this specification. However, if an optional feature is implemented, the feature shall be implemented as defined by this specification (optional normative).

1.5.2.6 Reserved

Reserved is a keyword indicating reserved bits, bytes, words, fields, and code values that are set-aside for future standardization. The use and interpretation of these may be specified by future extensions to this specification and, unless otherwise stated, shall not be utilized or adapted by vendor implementation. A reserved bit, byte, word or field shall be set to zero by the sender and shall be ignored by the receiver. Reserved field values shall not be sent by the sender and, if received, shall be ignored by the receiver.

1.5.2.7 Shall

Shall is a keyword indicating a mandatory (normative) requirement. Designers are mandated to implement all such requirements to ensure interoperability with other compliant devices.

1.5.2.8 Should

Should is a keyword indicating flexibility of choice with a preferred alternative. Equivalent to the phrase "it is recommended that".

1.5.3 Capitalization

Some terms are capitalized to distinguish their definition in the context of this specification from their common English meaning. Words not capitalized have their common English meaning.

1.5.4 Italic Text

Italic text is used to identify variable names, register field and packet field names, or reference document titles.

1.5.5 Numbering

Numbers that are immediately followed by a lowercase "b" (e.g. 01b) are binary values. Numbers that are immediately followed by an uppercase "B" are byte values. Numbers that are immediately followed by a lowercase "h" (e.g. 3Ah) are hexadecimal values. Numbers not immediately followed by either a "b", "B", or "h" are decimal values.

1.5.6 Bit, Byte, DW, and Symbol Conventions

A bit, byte, DW, or Symbol residing in location n within an array is denoted as bit(n), byte(n), DW(n), or Symbol(n).

A sequence of bits, bytes, DWs, or Symbols residing in locations n to m (inclusive) within an array is denoted as bit[m:n], byte[m:n], DW[m:n], or Symbol[m:n].

1.6 Terms and Abbreviations

This section lists and defines the terms and abbreviations used throughout this specification. Note that terms and abbreviations not defined here, use their generally accepted or dictionary meaning.

Term/Abbreviation	Description
Asymmetric	eUSB2V2 USB operation where the upstream (peripheral to host) and the downstream (host to peripheral) bit rates can be different.
CDR	Clock-Data Recovery circuit that extracts the phase and frequency information from the incoming data stream.
Burst sequence	The set of burst transactions executed during a service interval.
Burst transaction	An eUSB2V2 isochronous transaction of a single IN or OUT PID token packet followed by up to consecutive 3 Data packets. Where the DATA2, DATA1, and DATA0 PIDs are assigned in decending order, and DATA0 is always assigned to the last Data packet transferred.
High-Bandwidth	Refers to bits[12:11] of the standard USB2 Endpoint descriptor wMaxPacketSize field.
HSx	Generalized term for the supported eUSB2V2 data rates. Specifically, HS1 is 480Mb/s, HS2 is 960Mb/s, and so on. The maximum data rate is HS10, which is 4.8Gb/s. Note that HS1 is only used in an Asymmetric link configuration.
I/O Request Packet	An identifiable request by a software client to move data between itself (on the host) and an endpoint of a device in an appropriate direction.
IRP	See I/O Request Packet above.
MaxPacketSize	Refers to bits[10:0] of the standard USB2 Endpoint descriptor wMaxPacketSize field.
eUSB2V2	USB operation at either a eUSB2V2 symmetric or asymmetric rate.
HSS*x*-speed	Symmetric USB operation at a eUSB2V2 speed, where *x* is the integer multiple of 480Mb/s. Note, 480Mb/s symmetric links are not supported by eUSB2V2.
HSU*x*-speed	Asymmetric USB operation where the downstream link is operating at 480Mb/s and the upstream link is operating at a multiple of *x* times 480Mb/s.
HSD*x*-speed	Asymmetric USB operation where the upstream link is operating at 480Mb/s and the downstream link is operating at a multiple of *x* times 480Mb/s.
SI	Service Interval. The period between consecutive requests to a USB endpoint to send or receive data.
Symmetric	eUSB2V2 USB operation where both the upstream and downstream bit rates are equal.
Functional Mode	The normal, regular operation mode of an eUSB2V2 link in L0, L1, or L2.
Compliance Mode	The mode of operation specifically defined for compliance testing, a sub-state of Default.
Rx Margining Mode	The mode of operation specifically defined for receive margining, a sub-state of Default.

2 Architecture

eUSB2V2 is a technology constructed on the USB 2.0 architectural base, including but not limited to:

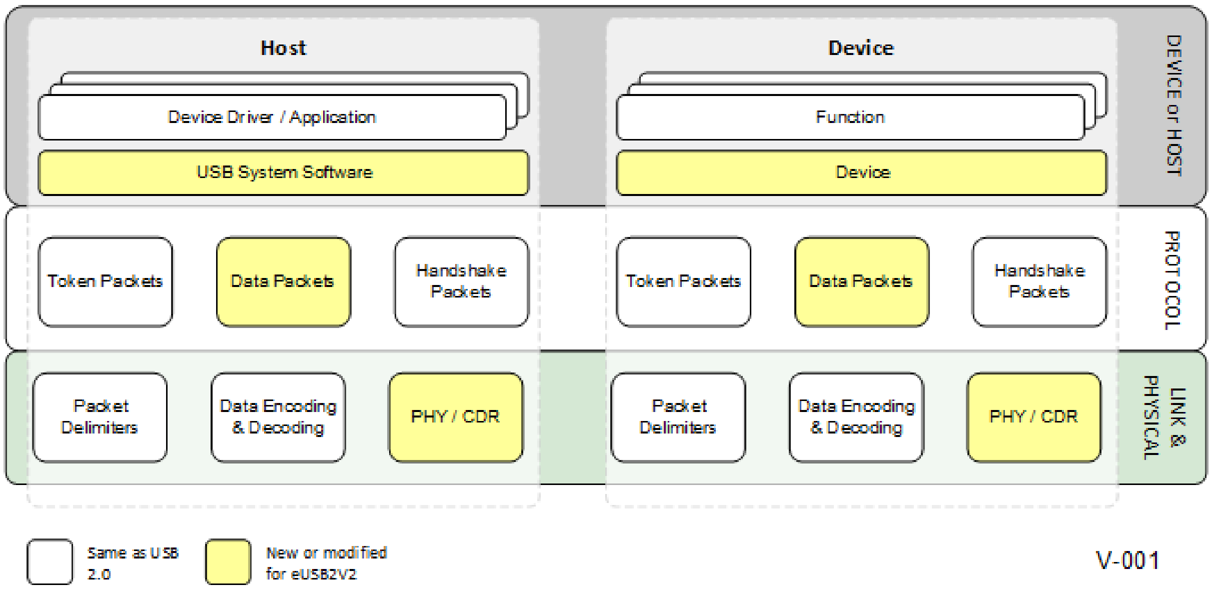
- Half-duplex link
- Host scheduled, transactional protocol
- Device framework

eUSB2V2 provides a low-voltage, power efficient PHY and link that is based on eUSB2 native mode. Refer to eUSB2 [3] for details of the eUSB2 native mode definition. The support of eUSB2 native mode and/or repeater mode defined by the eUSB2 specification is optional.

This section assumes the reader is familiar with the USB 2.0 specification [1] definitions for USB 2.0 packets, transactions, transfer types, etc. They are not repeated in this specification, however some text is included to make this specification easier to read. Please refer to USB 2.0 [1] for more information.

Figure 2-1 illustrates the architectural stack for eUSB2V2.

Figure 2-1. eUSB2V2 I/O Architectural Stack



The illustration above is unique for a USB 2.0 stack in that it borrows nomenclature from the later USB 3.x specification to identify the necessary layering. The architecture is assembled with a principle of isolating changes to areas only where needed to provide the necessary functionality. The premise is to deliver the new capabilities of the technology as transparently as possible to the existing USB 2.0 stack. To that end, the highlights in Figure 2-1 include the areas of change.

- Client applications and device drivers, including all class drivers, shall not require modification to properly function with eUSB2V2 enabled peripherals.
- A low-voltage PHY that can deliver a 4.8Gb/s symmetric or 4.8Gbps/480Mbps asymmetric signaling rates, refer to Chapter 3 for details.
- Host and peripheral controllers require changes to support the eUSB2V2 PHY and protocol.
- New descriptors are defined by the USB framework to expose the eUSB2V2 capabilities that the host and peripheral layers need to be aware of, refer to Chapter 5.
- Data Packets must accommodate new MaxPacketSizes.

The eUSB2V2 changes are purposely composed to minimize the impact on host controllers, peripherals, and system software, while maximizing data throughput and robustness.

2.1 eUSB2V2 Features

This section describes the primary changes introduced by eUSB2V2 to support high bandwidth embedded USB2 applications.

eUSB2V2 is an extension to eUSB2 native mode that targets on-board peripheral interconnects, providing:

- eUSB2V2 transfer rates:
  - HSSx-speed: Symmetric operation where the link is operating at 960Mbps to 4.8Gb/s in both transfer directions.
    - Optionally, if symmetric 480Mb/s operation is desired, it shall be done with via eUSB2 native mode.
  - HSUx-speed: Asymmetric operation where the link is operating at 960Mb/s to 4.8Gb/s in the upstream direction and at 480Mb/s in the downstream direction.
  - HSDx-speed: Asymmetric operation where the link is operating at 960Mb/s to 4.8Gb/s in the downstream direction and at 480Mb/s in the upstream direction.
- Reduced bus turnaround time:
  - Root hub ports are limited to a maximum of 1 re-driver.
  - Only in-box applications are supported, minimizing peripheral interconnect propagation times.
    - Enables shorter interpacket delays.
    - External operation is not supported.
- Increased maximum size of bulk packets when operating with eUSB2V2 compliant software:
  - Bulk endpoint MaxPacketSize increased from 512B to 1024B.
- Enables more efficient isochronous transfers to improve transaction efficiency:
  - More than 3072B of Isochronous packets may be transferred per service interval (SI).
  - Burst transactions are enabled for isochronous transfers, where up to 3 data packets may be transmitted following a single IN or OUT token, reducing bus turnaround times.
  - If transferring more than 1024B of isochronous data per microframe, then all data packets must be 1024B with the remainder in the last packet.
  - The *eUSB2 Isochronous Endpoint Companion* descriptor explicitly defines the dwBytesPerInterval value for isochronous endpoints. dwBytesPerInterval defines the maximum number of bytes an isochronous endpoint may transfer every service interval (SI), replacing the USB2 'high-bandwidth' mechanism for allocating isochronous bandwidth.
- Supports link power management LPM-L1 (L1) and Suspend (L2).
- Supports register access protocol (RAP) for eUSB2V2 peripheral or redriver configurations.
- Fully compatible to USB 2.0 base spec at the protocol layer.
- No change to USB 2.0 software programming model.
- Not compatible with the physical layer defined by USB 2.0 or eUSB2.
- Not compatible with standard USB 2.0 connectors defined by USB 2.0 and its derivatives.

2.2 eUSB2V2 Performance

This section discusses the performance enhancement mechanisms provided by eUSB2V2.

2.2.1 Bursts

eUSB2V2 defines a *Burst* mechanism that modifies the isochronous protocol by allowing a single IN or OUT transaction to transfer up to 3 data packets, thus improving throughput by eliminating bus turnarounds within a Burst.

As defined in the USB 2.0 specification, a typical IN or OUT isochronous transaction is comprised of 2 phases: Token and Data, each consisting of a single packet. The eUSB2V2 Burst mechanism modifies this definition to allow up to 3 packets to be transferred during the Data phase of a transaction, minimizing the number of bus turnarounds associated with the transaction. This feature is particularly important as the bit rates on a half-duplex bus are increased, and turnaround times become the limiting factor in achieving maximum throughput on the bus.

The Burst extension Is necessary to minimize the number of bus turnarounds per transfer.

2.2.2 eUSB2V2 Data Rates

eUSB2V2 supports extended data rates over USB2. The eUSB2V2 extended Symmetric data rates are referred to as *HSSx* speeds, where *x* * 480Mb/s defines the link data rate. eUSB2V2 also supports extended asymmetric data rates, where the rate in the 'slow' direction is always 480Mb/s, and the rate in the 'fast' direction is defined by *x* * 480Mb/s. *HSDx* defines a downstream data rate of *x* * 480Mb/s and an upstream rate of 480Mb/s, and *HSUx* defines an upstream data rate of *x* * 480Mb/s and a downstream rate of 480Mb/s. The variable *x* is an integer from 2 to 10.

Gigabit receivers are complex and expensive to implement, requiring more power, area, and design costs than megabit receivers. eUSB2V2 reduces the cost and complexity of eUSB2V2 peripherals, like cameras, by eliminating the need for them to support a 4.8Gb/s receiver. For instance, HSUx data rates are targeted at reducing the cost of implementing peripherals that only require large upstream data rates, for example, cameras.

2.2.3 Link Configurations

eUSB2V2 links may be configured as symmetric or asymmetric, each with multiple bit rate options. All bit rates are integer multiples of 480Mb/s. The bit rate implemented by an eUSB2V2 link is application dependent.

Table 2-1. eUSB2V2 Bit Rates

Bit Rate (Gb/s)	**480Mb/s Multiplier**
4.8	10
4.32	9
3.84	8
3.36	7
2.88	6
2.4	5
1.92	4
1.44	3
0.96	2

The *Upstream Facing Port* (UFP) and *Downstream Facing Port* (DFP) link configurations supported by eUSB2V2 are defined by the following notation:

- Symmetric *High Speed x* (HSS*x*), where *x* is the multiplier ranges from 2 to 10.
- Asymmetric *HS Upstream x* (HSU*x*), where *x* is the multiplier ranges from 2 to 10, and the Downstream traffic is at 480Mb/s.
- Asymmetric *HS Downstream x* (HSD*x*), where *x* is the multiplier ranges from 2 to 10, and the Upstream traffic is at 480Mb/s.

With asymmetric link configurations, the 'fast' link direction may be any speed from 960Mb/s to 4.8Gb/s supported by eUSB2V2, however the 'slow' direction shall always be 480Mb/s. *Symmetric* link configurations are any case where the upstream and downstream bit rates are identical starting at 960Mb/s.

Symmetric 480Mb/s operation is optionally supported via eUSB2 native mode.

Asymmetric link bit rates are supported to simplify peripheral design and optimize the power and EMI characteristics for a target application.

2.3 Control Messages

The eUSB2 specification [3] defines a generic Control Message Register Access Protocol (CM.RAP) protocol[1], assigning a command message ID of 15 (CM.15) to CM.RAP operations. CM.RAP defines a 64-byte register address space, and Read, Write, Set and Clear messages that can be issued to the registers. However, the eUSB2 specification does not define any usage for the CM.RAP mechanism or assign any register definitions.

eUSB2V2 extends the CM.RAP definition in two ways; 1) assigning separate CM message IDs for communicating with hosts/peripherals and re-drivers, and 2) allocating the registers in the respective address spaces.

- A peripheral device shall implement CM.15 as a receptor.
- A host port shall implement CM.15 as an initiator and a receptor. Note that the requirement for a host to implement CM.15 as a receptor is for the external test equipment to gain access to its PHY configuration and operational mode registers for compliance tests. Refer to Section 3.9 for the host registers accessible by external test equipment.

The encoding of eUSB2 Control Messages is extended as follows:

Table 2-2. Encoding of eUSB2V2 Control Messages

CM[3:0]	Parity	Description	Control Message Name
0-13	X	As defined in the eUSB2 spec.	
14	0	Start of Re-driver register access	CM.14
15	1	Start of host or peripheral register access	CM.15

In this document the term "CM.RAP" is used to generically refer to a CM.14 or a CM.15 message. The CM.14/CM.15 notation is used to refer to a specific target of a CM.RAP format message.

Refer to section 3.9 for more information.

[1] Refer to sections 3.3.8 and 6 in the eUSB2 specification [3] for more information on CM.RAP.

3 Physical Layer

This chapter describes the physical layer of eUSB2V2. It contains transceiver architecture, electrical and channel requirements, and PHY operation. The eUSB2V2 operation is defined for native mode only. Any eUSB2V2 port that optionally supports eUSB2 shall comply with eUSB2 native mode operation.

An eUSB2V2 port includes the following new features for eUSB2V2 operations.

- Extended High-Speed (HSx) data rates: Starting from 960Mb/s up to 4.8Gb/s in steps of 480Mb/s.
- eUSB2V2 peripheral connect.
- Symmetrical and asymmetric data rates between host and peripheral. Depending on application and configuration, the up-stream data rate may be the same or different from the down-stream data rate. See section 2.2.2 for definitions of HSSx, HSDx, and HSUx speeds. For example, an eUSB2V2 camera can send data to the host at 4.8Gb/s but receive data from the host at 480Mb/s.
- Reduced bus turn-around time with direct connect between host and peripheral. An eUSB2V2 re-driver may be included.
- Configurable low power transmitter and Rx CTLE to facilitate received data recovery through over-sampling architecture.
- Scrambling of data.

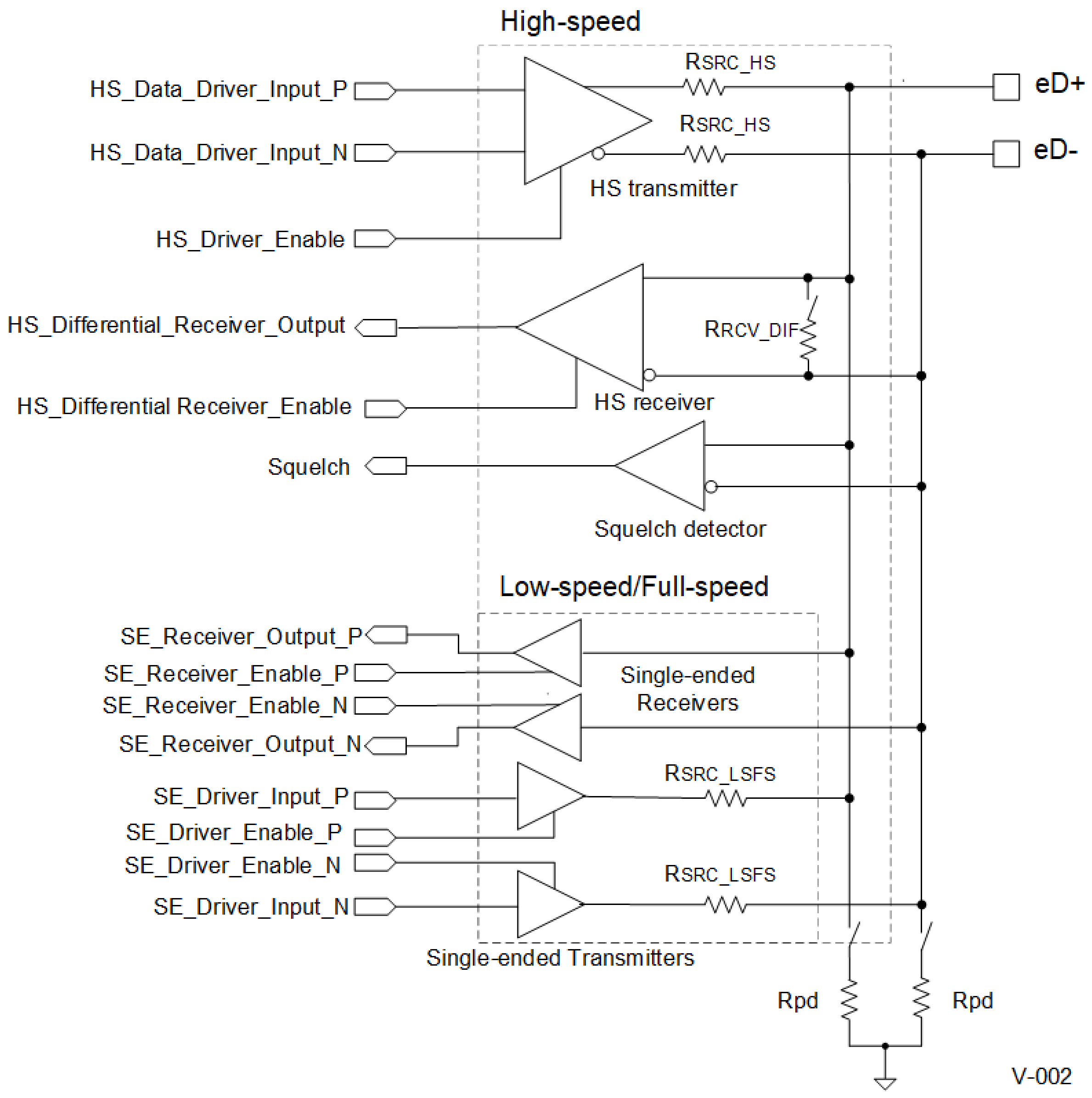
These new features and capabilities are the focus of this chapter and are discussed in detail in the following sections.

This specification does not address any Electromagnetic Compatibility (EMC) regulatory compliance. It is the responsibility of product designers to make sure that their designs comply with all applicable EMC regulatory requirements.

3.1 PHY Architecture

The basic construction of eUSB2V2 PHY is the same as eUSB2. Figure 3-1 below is reproduced from the eUSB2 specification. This includes a differential transceiver, a low power squelch detector, two pairs of single-ended buffers, pull-down resistors, and additional control signals for eUSB2V2 operations. The requirements on the pull-down resistors and the low-speed/full-speed/high-speed transceiver remain the same as eUSB2. Refer to Section 3.1 of the eUSB2 specification for details. The HSx transceiver requirements are described in later sections.

Figure 3-1. eUSB2V2 Physical Layer Transceiver Block Diagram



3.2 HSx Bus State, Signaling and Operations

The HSx bus state representation and HSx signaling are the same as HS. Refer to Section 3.2.3 of the eUSB2 specification.

The operation of the HSx transition to HSx idle is the same as HS. Refer to Section 3.2.4 of the eUSB2 specification.

The HSx squelch entry operation is the same as HS. Refer to Section 3.2.5 of the eUSB2 specification.

The HSx squelch exit operation is different from eUSB2 native mode. Refer to Section 3.6.2 for details.

The mechanism of the HSx analog ping for disconnect detection is deprecated. In an embedded environment, a peripheral connectivity is known and managed by the system. An in-band mechanism for disconnect detection is no longer needed. Note, a peripheral may still perform soft disconnect based on XeSE1.

The HSx Bus Reset is the same as the USB 2.0 Bus Reset. Refer to Section 3.3.4.2.1 of the eUSB2 specification.

The HSx link state transitions include L1 and L2 Suspend entry, Resume or Remote Wake from L1 and L2 Suspend. The operations and signaling of these transitions are the same as HS. Refer to corresponding sections describing native mode operations in Sections 3.3.5, 3.3.6, and 3.3.7 of the eUSB2 specification.

3.3 PHY State Transition and Power Management

The eUSB2v2 operation state machine and power management are identical to eUSB2 native mode. The following sections describe the additional features specific to eUSB2v2 operations in Default, Connect, and Reset states.

3.3.1 Default State

During the Default State, the host may be directed to optionally configure its PHY transceiver. It may also be directed to configure the peripheral transceiver via CM.15 (CM.RAP). Note that other implementation specific methods may also be used, but they are outside the scope of this specification.

There are several sub-states defined in Default for various operations. Shown in Figure 3-2 is the Default sub-state machine with transition conditions. Note that Power-up is an artificial state to indicate power-on initial state.

3.3.1.1 Port Reset

Port Reset is a substate where the host is directed to transmit XeSE1 to reset the link. A peripheral shall enter this substate upon detecting XeSE1. Refer to Section 4.2.1 of the eUSB2 specification [3] for XeSE1 operation.

- A host shall enter Port Reset substate when it is directed to exit from Rx Margining Mode.
- PHY transmitter and receiver configuration registers defined in Table 3-11 with RAP addresses 4 and 7 through 10 shall not be reset when entering this substate. Note that PHY Configuration does not need to be re-programmed.

3.3.1.2 PHY Configuration

PHY Configuration is a substate for the host or peripheral PHY register configurations before entering Port Configuration, Compliance Mode, or Rx Margining Mode. Note that this substate is optional if the host and peripheral PHY registers are pre-configured or programmed through implementation specific method. Note also that peripheral PHY register configuration based on CM.15 has the priority over other register configuration means.

3.3.1.3 Port Configuration

Port Configuration is a substate for host to synchronize and conclude the link initialization before Connect. Refer to section 4.2.2 of the eUSB2 Rev1.2 specification for Port Configuration operation.

- A host shall initiate Port Configuration handshake by enabling the receiver termination.

3.3.1.4 Rx Margining Mode

Rx Margining Mode is a substate where a host or a peripheral is directed to perform receive margining. Note that Rx margining for HS1 receiver is optional.

- For host Rx margining, bits [d2:d0] of the host and peripheral Operational Mode registers shall be set to 010. Note that the traffic direction bit (d2) decides if Rx margining is performed towards the host or the peripheral. Refer to Section 3.8.6 for details of Rx Margining operations. Note also that the peripheral Operational Mode register is programmed based on CM.15.
- For peripheral Rx margining, bits [d2:d0] of the host and peripheral Operational Mode registers shall be set to 110.

Note also that in this substate, a host may program its PHY for different transmitter and receiver configurations. The peripheral PHY registers may also be programmed based on CM.15. If those register values are different from the ones in functional mode, they shall be restored by margining software to the values for functional mode before exiting Rx Margining Mode.

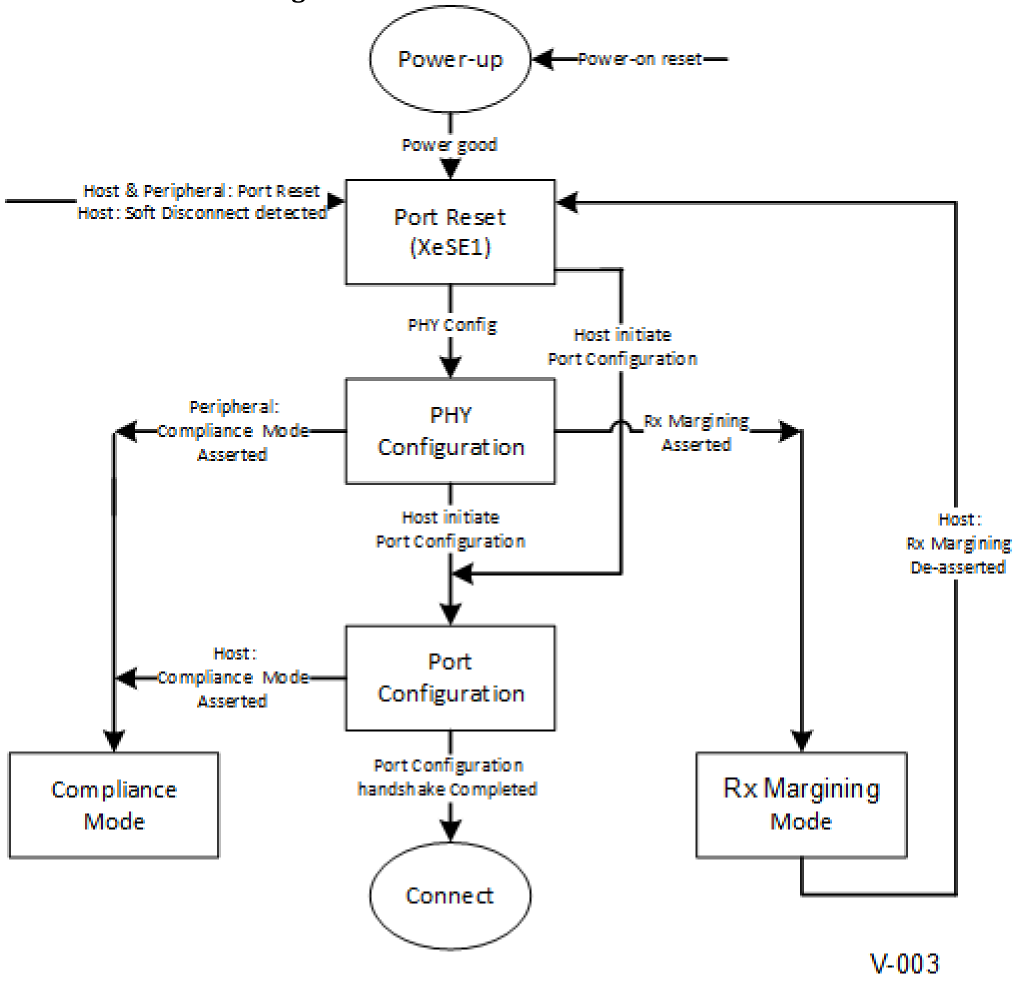
3.3.1.5 Compliance Mode

Compliance Mode is a substate where a host or a peripheral is directed to enter Compliance Mode.

- A host shall be directed to enter Compliance Mode.
- A peripheral shall enter Compliance Mode if bits [d1:d0] of its Operational Mode register are set to 01b via CM.15.

Specifically for a host, upon entry to this sub-state, it shall enable its CM.15 as a receptor, and subsequent operations for compliance test are directed by external test equipment based on CM.15 as an initiator. Refer to Sections 3.7.5 and 3.8.5 for Compliance Mode operations.

Figure 3-2. Default Sub-state Machine


Power-up
Power-on reset
Power good
Host & Peripheral: Port Reset
Host: Soft Disconnect detected
Port Reset (XeSE1)
PHY Config
Host initiate Port Configuration
PHY Configuration
Peripheral: Compliance Mode Asserted
Rx Margining Asserted
Host initiate Port Configuration
Host: Rx Margining De-asserted
Port Configuration
Host: Compliance Mode Asserted
Compliance Mode
Port Configuration handshake Completed
Rx Margining Mode
Connect
V-003

3.3.2 Connect and Reset States

The HSx configuration is based on the assumptions that the link data rate, the host and peripheral PHY settings are pre-determined according to the specific system implementation, which is out of the scope of this specification.

- A host and peripheral shall follow the HS connect and Bus Reset process of the eUSB2 native mode as defined in the eUSB2 specification [3]. Note that HSx data rate is already configured prior to Connect, no HSx speed negotiation is needed during Bus Reset.
- Upon completion of Bus Reset, the link shall enter L0 and start eUSB2V2 operation.

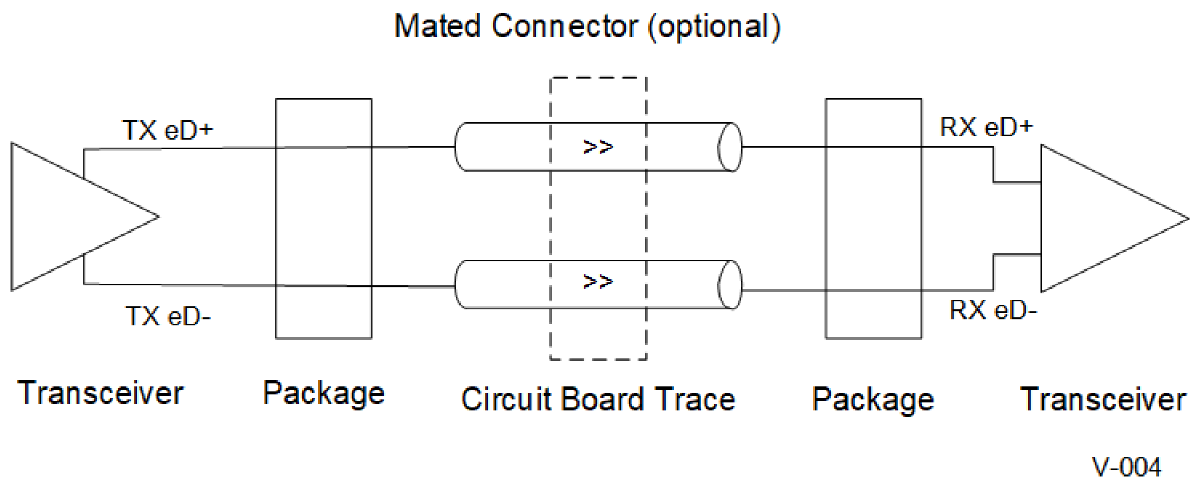
3.4 HSx Data Rate and Transceiver Configurations

Since eUSB2V2 targets in-box applications, the channel loss between a host and peripheral is known beforehand and therefore, the transceiver configuration to compensate for channel loss can be pre-configured given an HSx data rate. If needed, further configuration may be done via CM.15 (CM.RAP), see sections 3.7, 3.8, and 3.9 for further details.

3.5 Channel Requirements

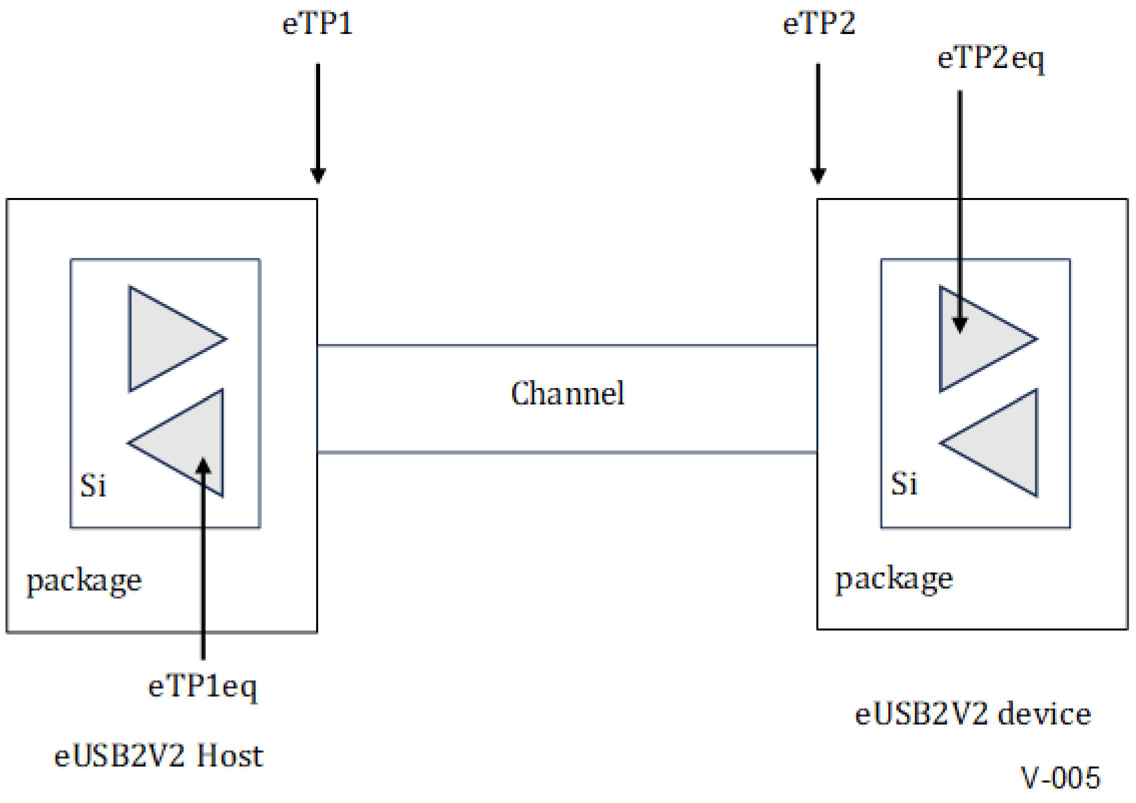
As eUSB2V2 is intended to support in-box applications in native mode, the physical channel is relatively simple with both the host and the peripheral being inside the system's physical boundary. An external cable or connector is not supported, however an internal cable and connector is allowed. Figure 3-3 below is reproduced from the eUSB2 specification illustrating the channel.

Figure 3-3. eUSB2V2 Channel Topology



Following the eUSB2 Rev 1.2 specification, the host's package pin is the eTP1 test point while the peripheral package pin is the eTP2 test point. New for eUSB2V2 is the eTP1eq and eTP2eq test points. These are test points on the receiver paths with reference package loss and receive equalization included. Figure 3-4 below illustrates the idea.

Figure 3-4. Measurement Plane for eUSB2V2



With the channel and test points defined as such, Table 3-1below shows the informative channel loss and impedance an eUSB2V2 transceiver is expected to support.

Table 3-1. eUSB2V2 Channel Loss and Impedance Expectation (Informative)

Parameter	Symbol	Value	Notes
Host package loss	IL_{HOST}	-3.0 dB	1
Peripheral package loss	$IL_{PERIPHERAL}$	-1.5 dB	1
Channel loss between eTP1 and eTP2	$IL_{CHANNEL}$	-15.5 dB	2
Motherboard channel impedance	Z_{MB}	85Ω	3
Cable impedance	Z_{CABLE}	100Ω	3
Notes: 1. Maximum loss at 2.5GHz, silicon die load included. 2. Maximum loss at 2.5GHz. Using common PCB and FPC cable as reference, including 2 vias and 2 connectors. The total of 20dB loss corresponds to packages plus roughly 30" of physical length at worst case corner. 3. Tolerance is +/- 10%.			

3.6 Signaling

The eUSB2V2 signaling scheme is the familiar differential mode signaling, and the usual definitions listed below are all applicable.

Differential voltage:	$V_{DIFF} = V_p – V_n$
Differential voltage peak-to-peak:	$V_{DIFF\text{-}PP} = \max(V_{DIFF}) -- \min(V_{DIFF})$
Common mode voltage:	$V_{CM} = (V_p + V_n)/2$

There is one important difference between eUSB2 and eUSB2V2 signaling schemes, which is eUSB2V2 does not allow un-terminated receiver. Furthermore, since eUSB2V2 supports multi-Gb/s data rates, the approach of using hexagonal eye mask at certain test point to determine the goodness of a transmitter or the IO link is too simplistic. An approach similar to USB3's should be used, and the details for eUSB2V2 will be described in the next few sections.

3.6.1 Scrambling and Bit Stuffing

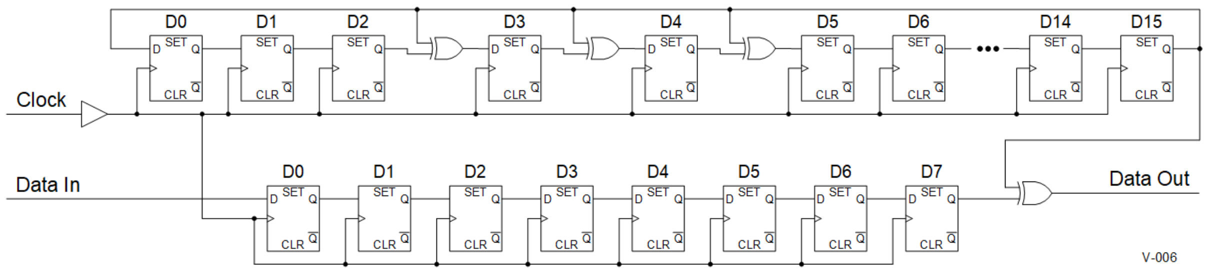
To minimize EMI/RFI concerns with potential data sequence that may be periodic, scrambling is required for all packets in both directions. The scrambling function is implemented using a free running Linear Feedback Shift Register (LFSR).

The LFSR is graphically represented in Figure 3-5. 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 scrambling rules are as follows:

- The scrambling shall start with the first byte after the PID, and end with the last byte before EOP.
- The LFSR implements the polynomial: $G(X)=X^{16}+X^5+X^4+X^3+1$
- The initialized value of an LFSR seed (D0-D15) shall be FFFFh.
- The LFSR value shall be advanced eight serial shifts for each Symbol.

Figure 3-5. LFSR with Scrambling Polynomial



Bit stuffing operation is the same as the USB 2.0 specification [1], except for when it starts.

- Bit stuffing shall be applied after scrambling and start with the first J of the SYNC pattern.

3.6.2 HSx SYNC Pattern and EOP of SOF

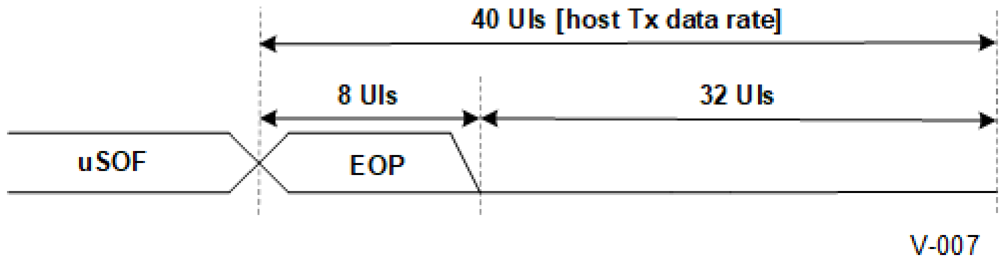
The HSx SYNC pattern shall be 40 bits long, with the first 24 bits of consecutive Ks, followed by seven pairs of KJ and ended with two Ks. The consecutive Ks in the preceding SYNC pattern is specified to reduce the squelch detector sensitivity based on a half differential comparator. (Note that entering squelch still needs a fully differential detector).

The receivers of host, peripheral, and eUSB2V2 re-driver may each consume up to 8 Ks to exit from the HSx idle state.

With analog ping removed, the EOP of SOF remains 8 UIs. To ensure interoperability with the host controller, the host PHY shall perform the following.

- It shall report to the host controller of 40 UIs of EOP, while only transmitting 8 UIs of EOP. Shown in Figure 3-6 is the timing diagram of the SOF EOP.

Figure 3-6. Timing illustration of SOF EOP



3.6.3 HSx Inter-packet Delay, Turn-around Time, and Transaction Timeout

HSx inter-packet delays are measured from the time when the line returns to HSx idle state at the end of one packet to the time when the line leaves the HSx idle state at the start of the next packet. A minimum amount of delay is required for each packet to be transferred and processed between the PHY and its controller.

The minimum inter-packet delay of the two consecutive transmitted packets shall be at least $T_{HSXIPDSD}$, which is referred to the transmitter bit-time. This is called the inter-packet delay for packets travelling in the same direction, and it is meant to ensure enough time for the receiver to enter the HSx idle state after receiving the first packet and ready to exit from the HSx idle state to receive the next packet. Example: A host sends an OUT token, waits for at least $T_{HSXIPDSD}$ bit times, then sends the data packet.

When a peripheral device is transmitting, the delay between two consecutive packets shall not exceed $T_{HSXRSPIPD1}$. Furthermore, once this limit is exceeded, the peripheral shall not send more data packets within the transaction.

When transmitting after receiving a packet, a host or peripheral shall provide an inter-packet delay of at least $T_{HSXIPDOD}$, that is referred to the bit time of the transmitted packet. This is called the inter-packet delay for packets travelling in the opposite direction. Example: A peripheral shall provide at least $T_{HSXIPDOD}$ of delay after receiving the IN token and before transmitting the data.

The maximum bus turn-around time, or the maximum host/peripheral response time ($T_{HSXRSPIPD1}$), is similar to the inter-packet delay for packets travelling in the opposite direction. When required, a host or peripheral shall respond (turn the bus around) no longer than $T_{HSXRSPIPD1}$.

A host or peripheral shall assume transaction timeout has occurred after $T_{TIMEOUT}$. For a host, this timeout may occur when a peripheral fails to respond, or the delay between data packets during an isochronous IN burst exceeds the limit. For a peripheral, this timeout may occur when the host fails to response.

For Isochronous Burst transfers, transaction timeout counter will be reset and restart after every packet received in a burst.

During an Isochronous IN transaction when the host is expecting to receive more than 1 data packet (dwBytesPerInterval > 1024), if the host receives corrupted data packet, and it is still expecting to receive further data packet, then the host shall not start a new transaction until $T_{TIMEOUT}$ is detected.

During an Isochronous OUT transaction, if the host runs out of data, it is allowed to start a new transaction immediately (complying to $T_{HSXIPDSD}$). In this scenario, the peripheral shall consider the uncompleted transaction as corrupted and start processing the new transaction.

Table 3-2. Timing Parameters

Parameter	Symbol	Min	Max	Units
Host/peripheral response time at HSx speed	$T_{HSXRSPIPD1}$		1920	HS10 bit time
Inter-Packet Delay for packets traveling in the Same Direction	$T_{HSXIPDSD}$	32		Transmit bit time
Inter-Packet Delay for packets traveling in the Opposite Direction	$T_{HSXIPDOD}$	32		Responding PHY transmit bit time
Transaction timeout	$T_{TIMEOUT}$	1		Micro-second

3.6.4 Radio Frequency Interference (RFI) Consideration

One important aspect of implementing high-speed IO inside a modern PC formfactor, such as a clam-shell laptop, is the consideration of RFI and its mitigation.

For example, eUSB2V2 can be used to connect the user-facing camera (peripheral) located at the top of a clam-shell screen to the SOC (host) on the motherboard in the base. A typical system implementation of such a link involves both cable and motherboard route. It is also typical for wireless (such as Wi-Fi) antenna to be placed at the top of the screen or inside the hinge. This means the antenna can be placed near the camera or the cable, thus posting significant RFI risk.

There are multiple effective RFI mitigation strategies. The exact details depend on the specific system design.

3.7 Transmitter Specification

This section provides the details of the eUSB2V2 transmitter. As already mentioned before, the range of data rate and channel insertion loss supported by eUSB2V2 is far beyond the original eUSB2, and thus an eUSB2V2 transmitter requires equalization capability, and its jitter also needs to be carefully controlled.

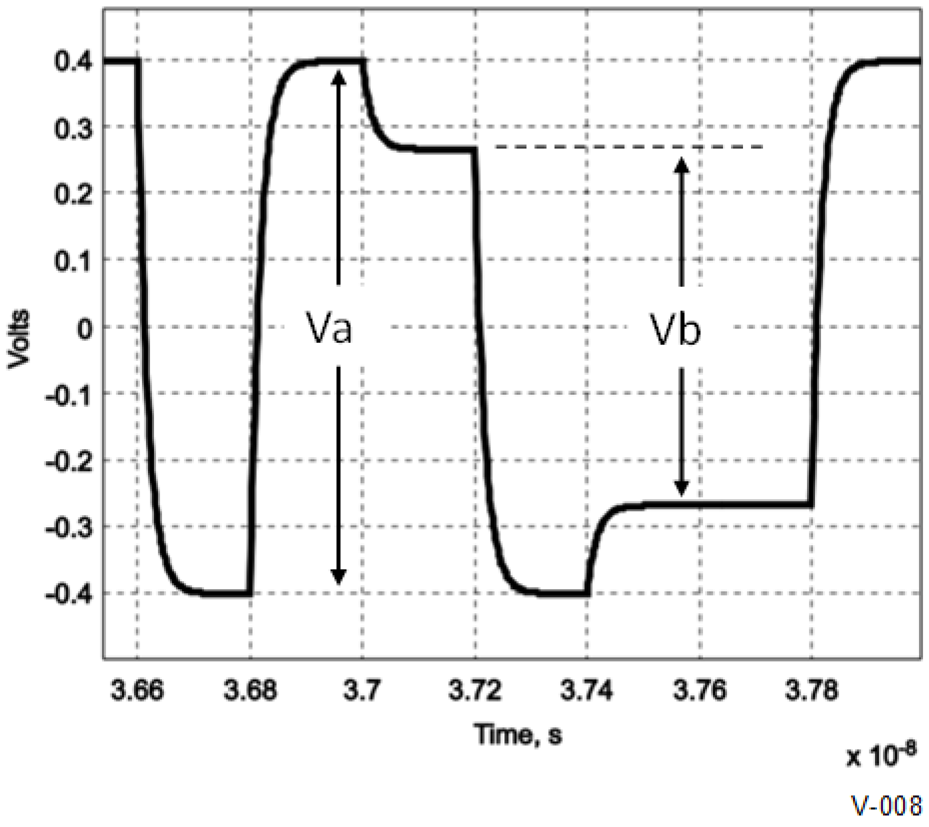
3.7.1 Transmitter Output Voltage Swing

Consistent with the original eUSB2 low power operation, and to mitigate potential RFI issue, support of low Tx voltage swing is required. Transmit voltage measured near the pin with termination shall be programmable between 360 to 800 mV-differential-peak-to-peak.

3.7.2 Transmitter De-emphasis

Similar to USB3 Gen1, an eUSB2V2 transmitter shall support transmit de-emphasis. Figure 3-7 below shows a differential waveform having roughly -125mV de-emphasis amplitude.

Figure 3-7. De-emphasis Waveform



De-emphasis amplitude, C1 = -(Va-Vb)/2

De-emphasis boost = 20log(Vb/Va)

The de-emphasis boost is the ratio of the de-emphasized voltage to the full voltage swing expressed in dB, see formula above. The Tx de-emphasis should be programmable between 0 to -8dB. Recommend a minimum support of 0, -3.5dB, -6dB, and -8dB de-emphasis.

Since eUSB2V2 supports a very wide range of transmit voltage swing, care must be taken in the implementation of the de-emphasis circuity. For example, to achieve -6dB boost at 800mV swing, the de-emphasis amplitude, C1, is -200mV. For the same -6dB boost at 400mV swing, however, C1 would be -100mV only.

3.7.3 Transmitter Electrical Parameters

Table 3-3. Transmitter Electrical Parameters

Symbol	Parameter	Value	Unit	Comments
UI	Unit Interval	208.33 (4.8Gb/s) to 2083.33 (480Mb/s)	ps	Programmable data rate. UI tolerance is +/- 300ppm for each HSx data rate. This tolerance is <u>not</u> included in the values listed.
$V_{TX\text{-}DIFF\text{-}PP}$	Differential p-p Tx voltage swing	360 (min) 800 (max)	mV	See note 1.
$Z_{TX\text{-}DIFF}$	Differential output impedance	65 (min) 80 (nominal) 95 (max)	Ω	Programmable, recommend step size of 10Ω.
C_{TX}	Tx pad capacitance	1.5 (nominal) 2.0 (max)	pF	
$V_{TX\text{-}DE\text{-}RATIO}$	Tx de-emphasis boost	0 (min) -8 (max)	dB	See note 2.
$T_{RISE/FALL}$	Signal rise and fall time	0.1 (min)	UI	Zero to 100%. Rise and fall times match to within 5%.
$T_{P\text{-}N\text{-}SKEW}$	P/N skew	0.025 (max)	UI	
$V_{TX\text{-}CM}$	Transmit common mode voltage	170 (min) 420 (max)	mV	Does not include AC noise. Un-terminated.
$V_{TX\text{-}CM\text{-}AC}$	Tx AC common mode voltage active	30 (max)	+/- mV peak	
Notes: 1. Measured near pins, terminated, using long-1 and long-0 bit pattern. Programmable, recommend step size of 100mV or less. 2. Recommend minimum support of 0, -3.5, -6, and -8dB. Must be programmable.				

Table 3-4. Transmitter Informative Jitter Budget at the Die Pad

Symbol	Parameter	Value	Unit
Rj_{TX}	Transmit random jitter	0.71	%UI-rms
Dj_{TX}	Transmit deterministic jitter	15	%UI-pp
Tj_{TX}	Transmit total jitter	25	%UI-pp
Note: Tj at 10^{-12} BER is calculated as 14.068 * Rj + Dj			

3.7.4 Transmitter Eye Test

For compliance purposes, the transmitter output shall meet the eye mask requirement specified in this section. The measurements shall be performed at the end of a calibrated test channel with appropriate insertion loss using the scramble-zero bit pattern.

Reference receiver equalization shall be applied and optimized using measurement software. Furthermore, reference jitter transfer function shall be used to reject low frequency jitter before calculating the eye opening, see equation (2) in section 3.8.3.

Transmit equalization shall be optimized for best eye opening. Both eye height and eye width are measured for 3 million consecutive UI, and the eye width is then extrapolated to 10^{-12} bit-error-rate.

Table 3-5. Normative Transmitter Eye Mask

Parameter	Value	Unit	Note
Eye Height	85 (min)	mV	1, 2, 4
Eye Width	0.4 (min)	UI	1, 3, 4
Notes: 1. Measured at the end of calibrated test channel, 17dB of loss at 2.5GHz for testing host, 18.5dB of loss at 2.5GHz for testing peripheral. 2. Measured over 3 million consecutive UI. 3. Measured over 3 million consecutive UI and extrapolated to 10^{-12} BER. 4. Measured minimum opening at the center of the UI.			

3.7.5 Transmit Compliance Mode

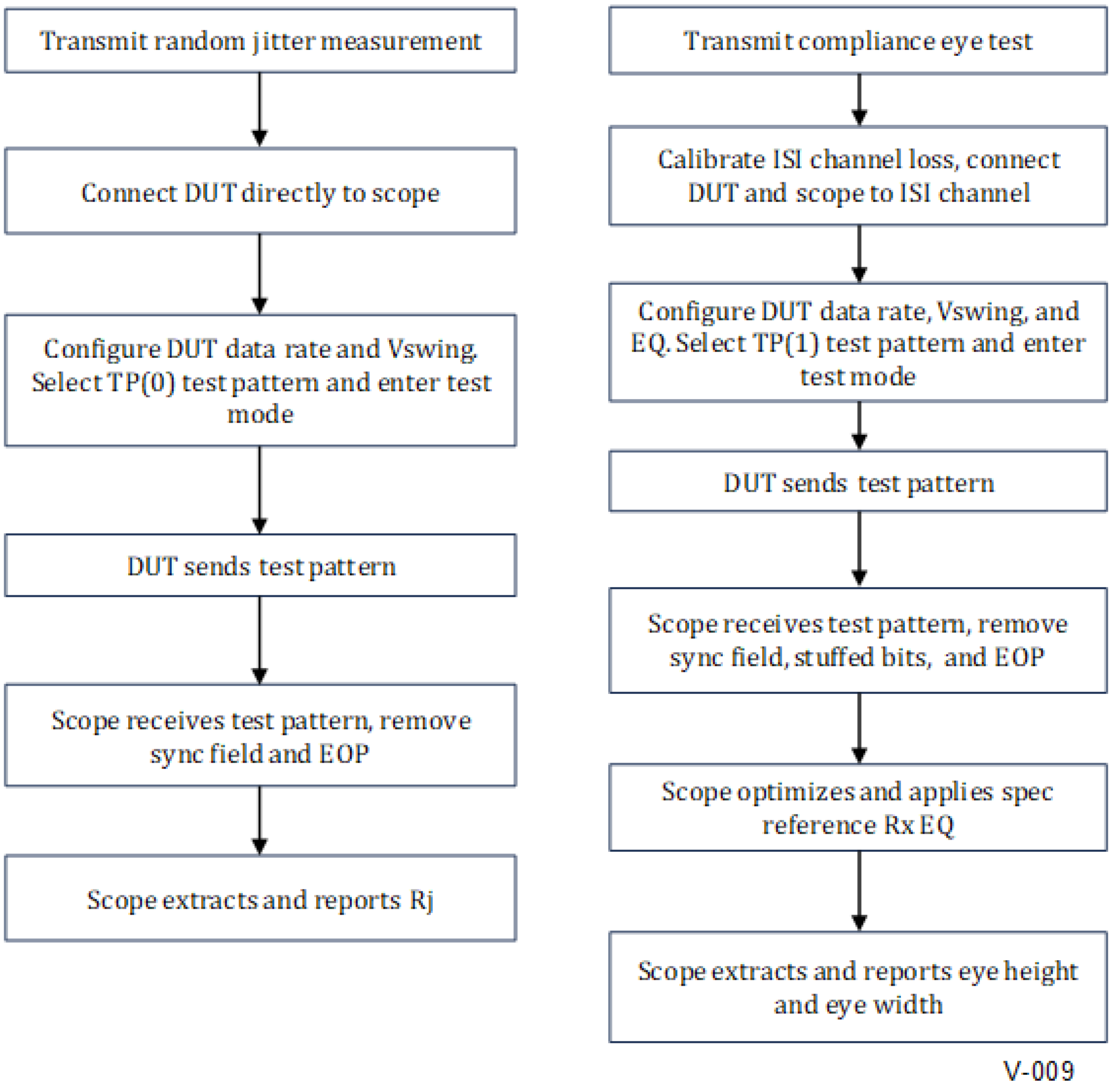
To facilitate HSx compliance test, an eUSB2V2 transceiver shall support applicable Configuration and Operational Mode Registers detailed in section 3.9. For a host and a peripheral device, these registers shall be accessible via CM.15. It is expected test equipment will support these registers and be able to act as CM receptor or initiator, depending on the part under test. Note that a part may be a host port or a peripheral port. Note also that this compliance test methodology applies to both transmitter and receiver compliance test.

When a host or peripheral enters Compliance Mode, a test equipment is used for test control purposes. The equipment shall issue CM.15 write to configure the host or peripheral's transmitter. It shall also select the proper test pattern and direct the PHY under test into the proper Compliance Mode via CM.15 write to the host or peripheral's Operational Mode register. The PHY shall transmit the test pattern after being directed into the Compliance Mode with the Trigger bit asserted.

Figure 3-8 below shows the Tx test flow.

All features required to support electrical testing, including test pattern generation and scrambling, shall be supported at the PHY level.

Figure 3-8. Transmit Test Flow Diagrams


Transmit random jitter measurement
Connect DUT directly to scope
Configure DUT data rate and Vswing. Select TP(0) test pattern and enter test mode
DUT sends test pattern
Scope receives test pattern, remove sync field and EOP
Scope extracts and reports Rj
Transmit compliance eye test
Calibrate ISI channel loss, connect DUT and scope to ISI channel
Configure DUT data rate, Vswing, and EQ. Select TP(1) test pattern and enter test mode
DUT sends test pattern
Scope receives test pattern, remove sync field, stuffed bits, and EOP
Scope optimizes and applies spec reference Rx EQ
Scope extracts and reports eye height and eye width
V-009

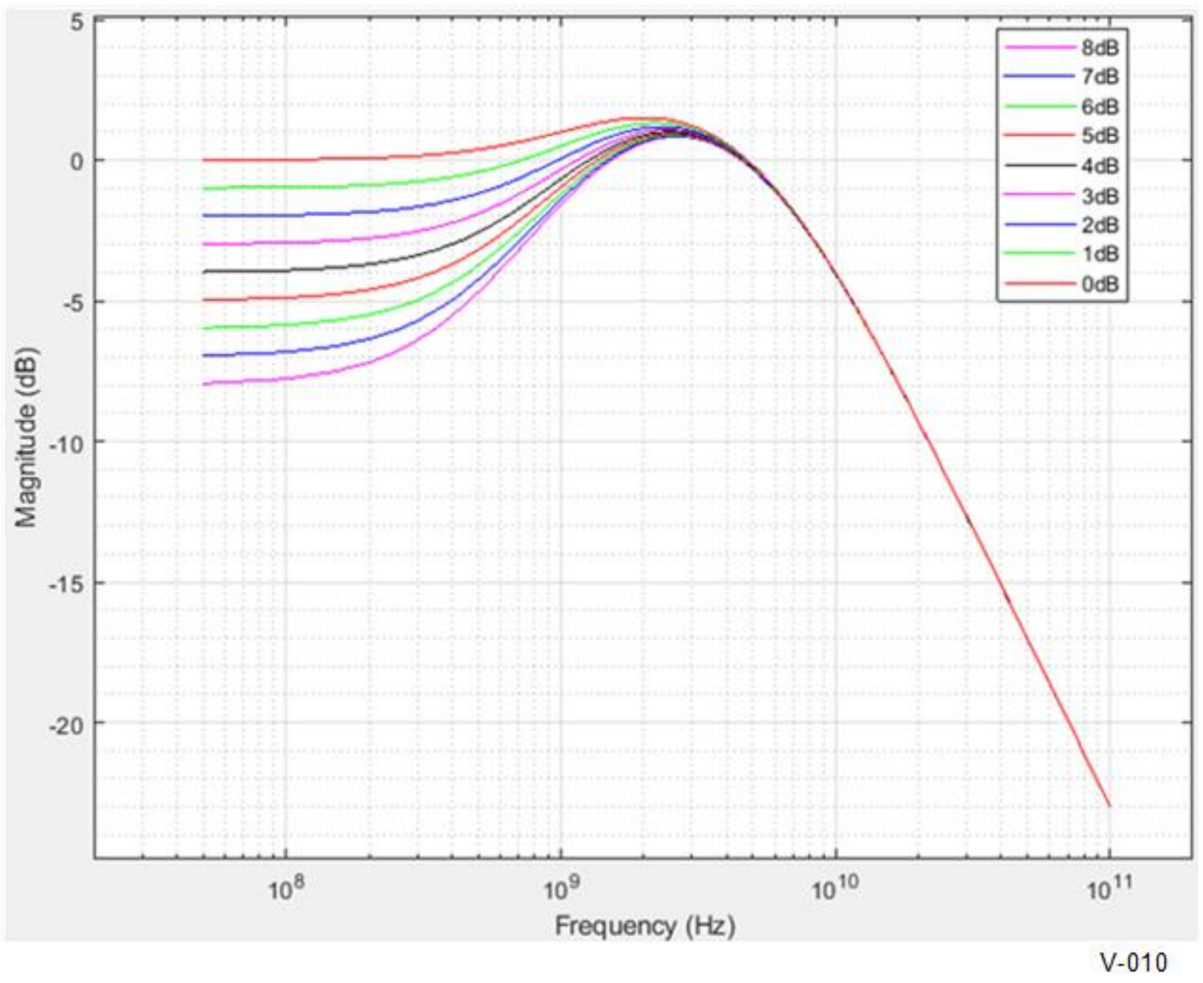
3.8 Receiver Electrical Specification

This section provides the details of the eUSB2V2 receiver. Again, to support multi-Gb/s data rates and high channel insertion loss, the receiver requires equalization capability. Furthermore, an eye mask requirement for receiver test is also defined in Section 3.8.4.

3.8.1 Receiver CTLE Function

Similar to USB3, an eUSB2V2 receiver shall support receive CTLE equalization. For flexibility and programmability, it is recommended to follow the USB3 gen2 reference CTLE (section 6.8.2.2 of the USB3.2 specification) and add two additional higher boost settings. See the 7dB and 8dB curves in Figure 3-9 below.

Figure 3-9. Reference CTLE Curves



Furthermore, to support low voltage swing operation, eUSB2V2 receiver shall have Variable Gain Amplifier (VGA). The VGA should be programmable to provide 0 to 2.5x of signal amplification.

3.8.2 Receiver Electrical Parameters

Table 3-6. Receiver Electrical Parameters

Symbol	Parameter	Value	Unit	Notes
UI	Unit Interval	208.33 (4.8Gb/s) to 2083.33 (480Mb/s)	ps	Programmable data rate. UI tolerance is +/- 300ppm for each HSx data rate. This tolerance is not included in the values listed..
$Z_{RX\text{-}DIFF}$	Differential input impedance	65 (min) 80 (nominal) 95 (max)	Ω	Programmable, recommend step size of 10Ω.
C_{RX}	Rx pad capacitance	1.5 (nominal) 2.0 (max)	pF	
$T_{RX\text{-}TJ}$	Max Rx inherent jitter	0.25 (max)	UI	Peak-to-peak jitter at 1e-12 BER.
$V_{SQUELCH\text{-}DIFF}$	Squelch detect threshold (peak differential)	25 (min) 35 (max)	mV	
$V_{RX\text{-}CM}$	Receive common mode voltage	70 (min) 250 (max)	mV	Include Tx to Rx impedance mismatch, but not AC noise.
$V_{RX\text{-}CM\text{-}AC}$	Rx AC common mode voltage	60 (max)	+/- mV peak	

3.8.3 Reference Jitter Transfer Functions

Refer to section 6.5.2 of the USB3 specification [2], the receiver's CDR transfer function and the Jitter Transfer Function (JTF) are:

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

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

With the damping factor $\zeta = 0.707$, Table 3-7 below shows the natural frequency ω_n, the CDR 3dB frequency, and the JTF 3dB frequency at all the HSx data rates.

Table 3-7. CDR and JTF Characteristic Frequencies

Bit Rate (Gb/s)	Natural Frequency ($2\pi 10^6$ rad/s)	CDR 3dB frequency (MHz)	JTF 3dB frequency (MHz)
4.80	4.9	10	4.9
4.32	4.42	9.1	4.42
3.84	3.94	8.1	3.94
3.36	3.40	7	3.40
2.88	2.92	6	2.92
2.40	2.43	5	2.43
1.92	1.94	4	1.94
1.44	1.46	3	1.46
0.96	0.97	2	0.97
0.48	0.49	1	0.49

3.8.4 Receiver Compliance Tests

For compliance purposes, a receiver shall be tested with a calibrated compliance test channel. A pattern generator shall send selected Test Pattern (Section 3.9.3) with jitter calibrated through the reference channel to the receiver. The receiver shall utilize the Compliance Mode to detect and report bit errors.

Table 3-8. Normative Receiver Stressed Eye Parameters

Parameter	Value	Unit	Note
Random Jitter	0.71	%UI-rms	1
Eye Height	85 (nominal)	mV	2, 3, 5
Eye Width	0.4 (nominal)	UI	2, 4
Notes: 1. Measured using clock pattern at the output of pattern generator over 3 million consecutive UI and extrapolated to 10^{-12} BER. 2. Measure at the end of calibrated test channel, 17dB of loss at 2.5GHz for testing host, 18.5dB of loss at 2.5GHz for testing peripheral. 3. Measured over 3 million consecutive UI. 4. Measured over 3 million consecutive UI and extrapolated to 10^{-12} BER. 5. Eye height is to be measured at the minimum opening at the center of the UI.			

A receiver shall also be tested for its jitter tolerance capabilities. Figure 3-10 below shows the jitter tolerance masks, and the corner frequencies are the JTF 3dB frequencies listed on Table 3-7 above. Table 3-9 below contains the input jitter requirements.

Figure 3-10. Jitter Tolerance Mask

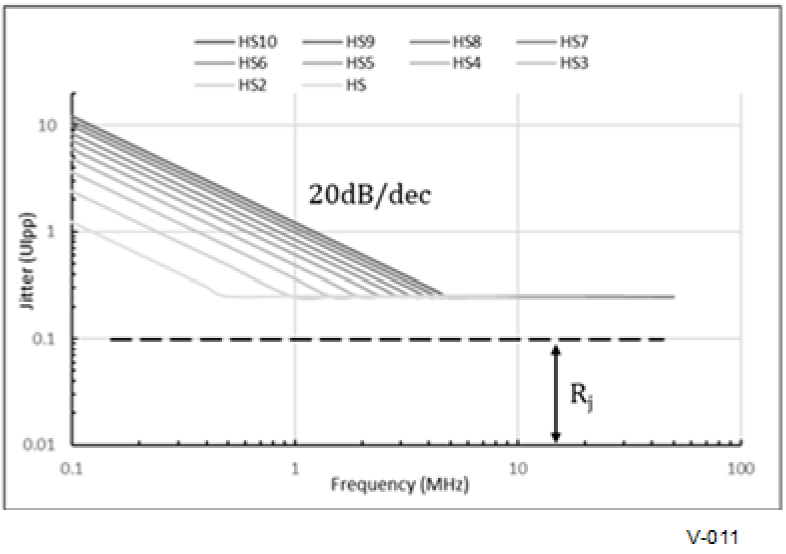


Table 3-9. Input Jitter Requirements for Rx Tolerance Testing

Symbol	Parameter	Value	Units
f1	Tolerance corner (JTF 3dB freq.)	See Table 3-7	MHz
J_{Rj}	Random Jitter	0.0071	UI-rms
J_{Rj_p-p}	Random Jitter pk-pk at 1e-12 BER	0.1	UI pk-pk
J_{Pj_fx}	Periodic Jitter at fx = x*f1, where x= 0.75, 0.5, 0.25, and 0.125	J_{Pj_fx} = 0.25/x – 0.1	UI pk-pk
J_{Pj_f1}	Periodic Jitter at f1	0.15	UI pk-pk
J_{Pj_50MHz}	Periodic Jitter at 50MHz	0.15	UI pk-pk
Notes: All parameters measured at BERT output. BERT Tx voltage, equalization, and DUT Rx equalization settings follow the optimized settings during the Stressed Eye Test. These settings shall be noted in the test report.			

3.8.5 Receive Compliance Mode

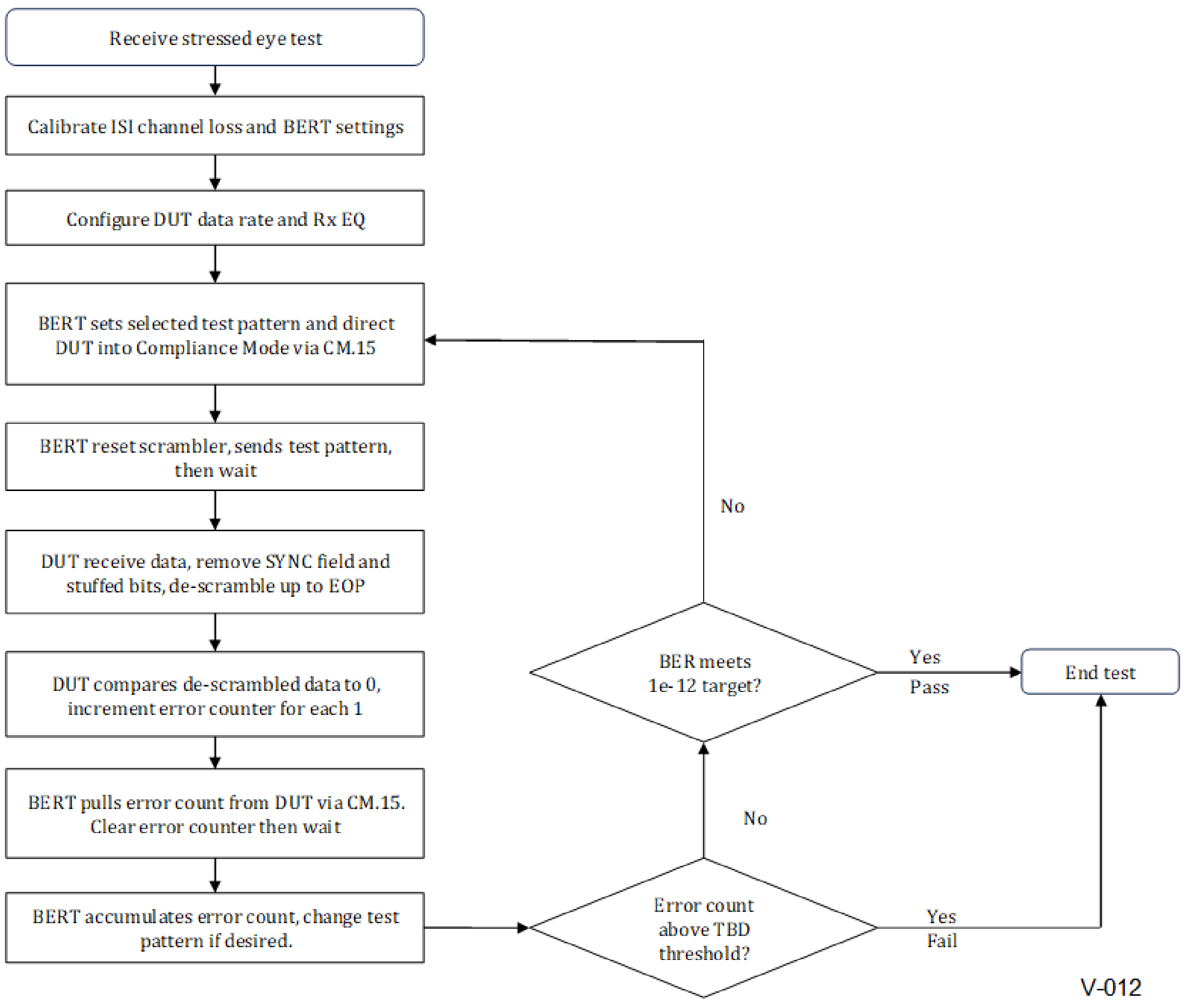
When a host or a peripheral enters Compliance Mode, the test equipment is used for test control purposes. The test equipment shall issue CM.15 write to configure the host or peripheral's receive data rate and equalization. The receiver test steps are the following:

1. Equipment selects test pattern and issues CM.15 write to PHY Operational Mode Register to direct the PHY under test into the proper Compliance Mode. The test equipment shall wait for 1 microsecond.
2. The PHY under test prepares its error detector according to the test pattern and resets its error counter.
3. After the wait time, the test equipment transmits the test pattern. The equipment shall wait for 1 microsecond.
4. After the test pattern is received, the PHY under test shall remove the stuffed bits, de-scramble the data, and count bit errors.
5. After the wait time, equipment issues a CM.15 read to capture the error count from the PHY's Error Count register.
6. Repeat from step 1 until either the target bit-error-rate is achieved, or the PHY fails the test.

Figure 3-11 below shows the Rx test flow. This process is also applicable to the jitter tolerance test.

All features required to support electrical testing, including descrambling, error detection, and error count reporting, etc., shall be supported at the PHY level.

Figure 3-11. Receive Test Flow Diagram


Receive stressed eye test
Calibrate ISI channel loss and BERT settings
Configure DUT data rate and Rx EQ
BERT sets selected test pattern and direct DUT into Compliance Mode via CM.15
BERT reset scrambler, sends test pattern, then wait
DUT receive data, remove SYNC field and stuffed bits, de-scramble up to EOP
DUT compares de-scrambled data to 0, increment error counter for each 1
BERT pulls error count from DUT via CM.15. Clear error counter then wait
BERT accumulates error count, change test pattern if desired.
Error count above TBD threshold?
No
Yes
Fail
BER meets 1e-12 target?
No
Yes
Pass
End test
V-012

3.8.6 Receive Margining

To facilitate system debug and tuning, eUSB2V2 receiver shall provide voltage and timing margining capability. Registers are defined in Sections 3.9.7 and 3.9.8. These registers allow software to control the receiver behavior for margining purpose. Margining shall be performed in the Default state using specific test pattern.

Hardware-based receive margining is optional, and its implementation details are vendor specific and outside the scope of this specification.

For voltage margining, a receiver sampler is offset from the nominal sampling position in the voltage (vertical) dimension. Independent voltage margining in the positive (high) and negative (low) directions is required. A receiver shall use the range and step size in Table 3-10 below for voltage margining.

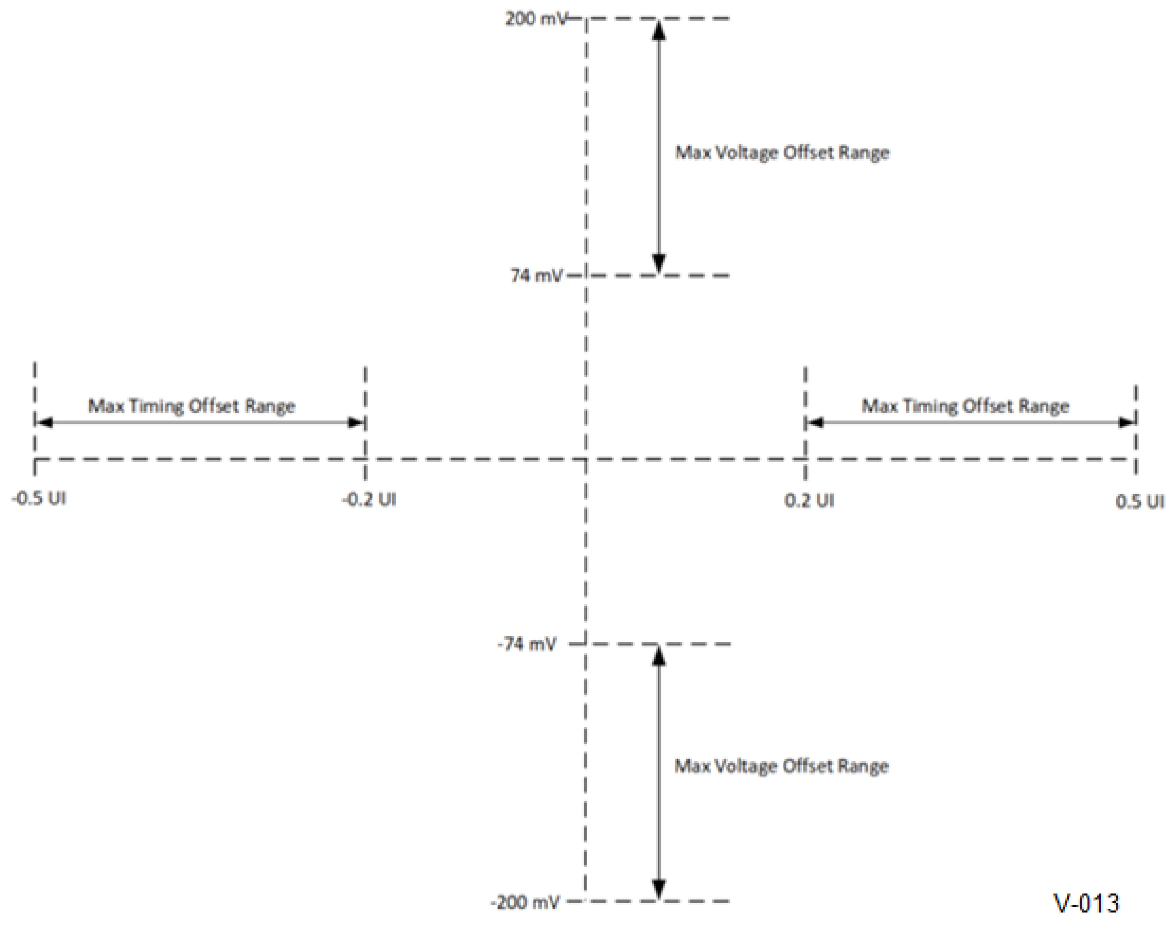
For timing margining, a receiver sampler is offset from the nominal sampling position in the timing (horizontal) dimension. Independent margining in the positive (right) and negative (left) directions is optional but recommended. Timing margining can be implemented using a jitter injection circuit to inject jitter onto the receive sampling clock to offset the data sampler from the nominal position. A receiver shall use the range and step size in Table 3-10 below for timing margining.

Table 3-10. Receive Margining Voltage and Timing Requirements

Parameter	Min	Max	Units
Voltage Margining Range	+/- 74	+/- 200	mV
Voltage Margining Step Size	Note 1	3	mV
Voltage Margining Steps (per direction)	Note 2		
Timing Margining Range	+/- 0.2	+/- 0.5	UI
Timing Margining Step Size	Note 3	0.03	UI
Timing Margining Steps (per direction)	Note 4		

Notes:
1. The minimum Voltage Margining Step Size is bounded by Min(Voltage Margining Range)/Max(Voltage Margining Steps).
2. The minimum Voltage Margining Steps (per direction) is bounded by ceiling(Min(Voltage Margining Range)/Max(Voltage Margining Step Size)).
3. The minimum Timing Margining Step Size is bounded by Min(Timing Margining Range)/Max(Timing Margining Steps).
4. The minimum Timing Margining Steps (per direction) is bounded by ceiling(Min(Timing Margining Range)/Max(Timing Margining Step Size)).

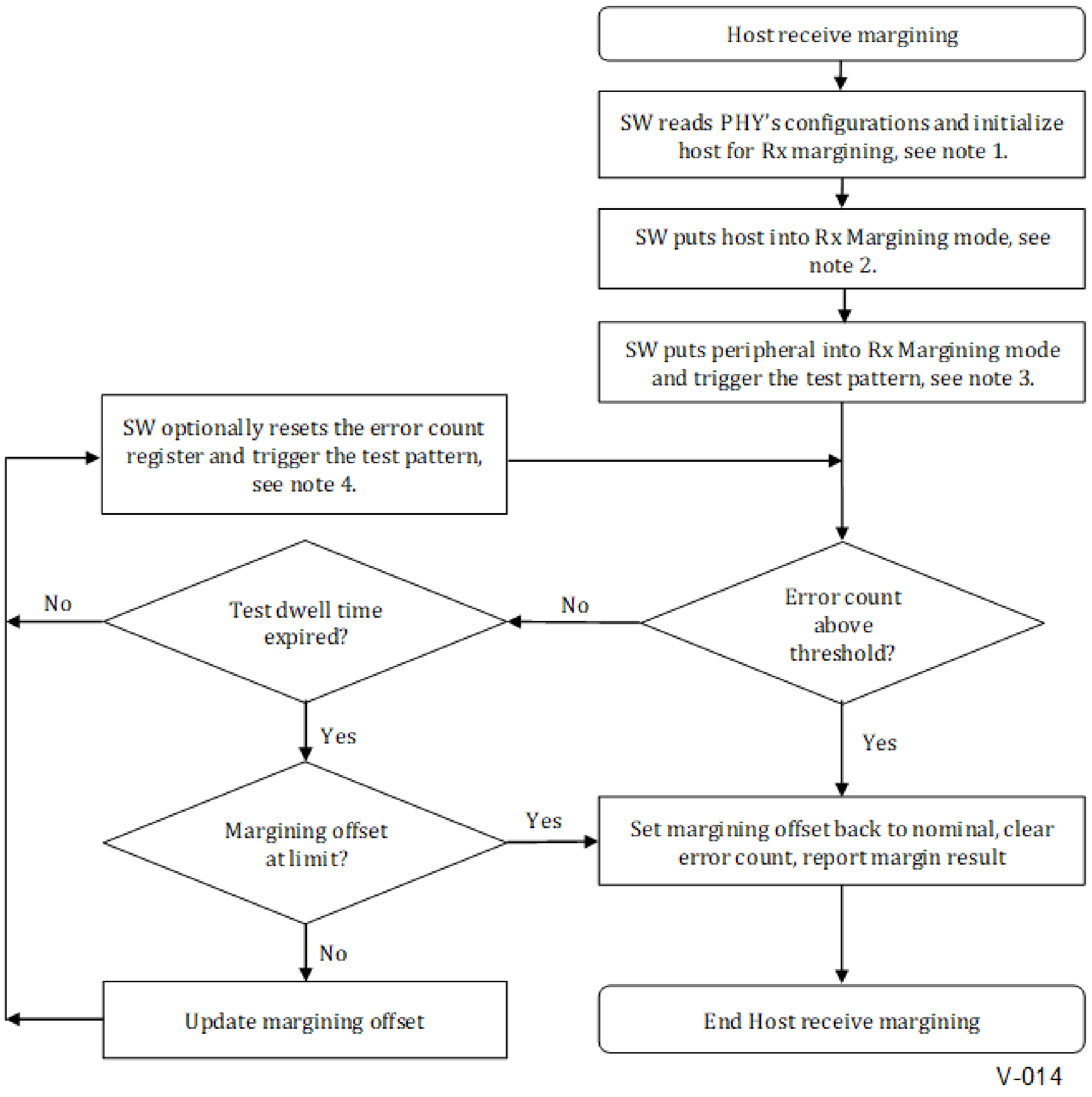
Figure 3-12. Receive Margining Range Requirements



3.8.6.1 Host Receive Margining

An eUSB2V2 host receiver shall support receive margining in the Default state. Figure 3-13 below shows an example flow of software driven host receive margining. Note that both the host and peripheral are put into the Rx Margining Mode. The Direction bits of the Operational Mode registers are set to 0 such that the host is receiving while the peripheral is transmitting. The test pattern should be TP(1), which is 3 million bits of PRBS16 pattern. The Trigger bit is used to direct the peripheral to transmit the test pattern. Note that dwell time is defined by the margining software. It is the time allocated to count a desired number of bits before changing the margining offset. The longer the dwell time, the lower the bit error rate margining can achieve.

Figure 3-13. Software Driven Host Receive Margining Example Flow


Host receive margining
SW reads PHY's configurations and initialize host for Rx margining, see note 1.
SW puts host into Rx Margining mode, see note 2.
SW puts peripheral into Rx Margining mode and trigger the test pattern, see note 3.
SW optionally resets the error count register and trigger the test pattern, see note 4.
Error count above threshold?
No
Test dwell time expired?
No
Yes
Yes
Margining offset at limit?
Yes
No
Set margining offset back to nominal, clear error count, report margin result
Update margining offset
End Host receive margining
V-014

Notes:

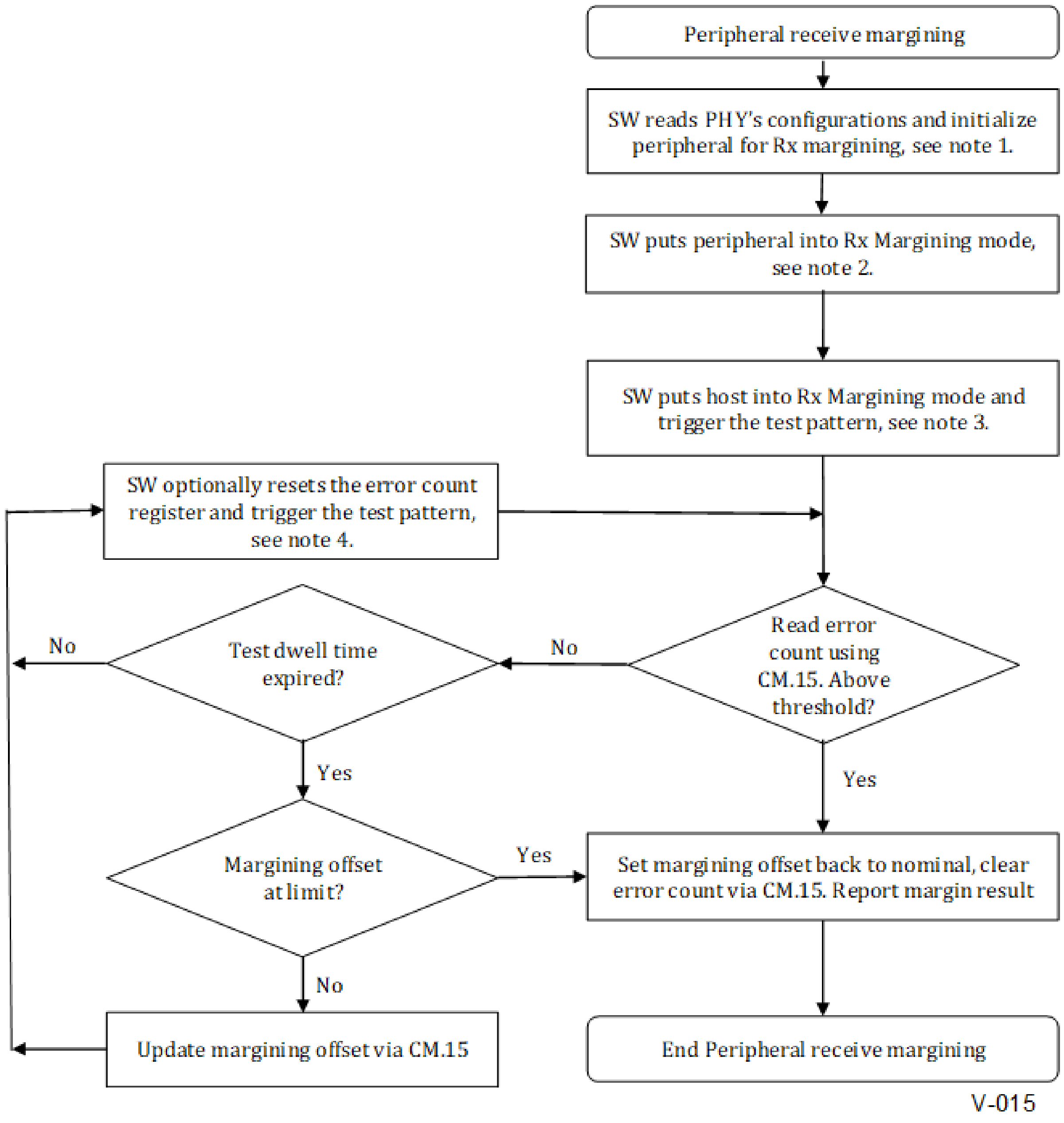
1. SW direct link into Default, reads host and peripheral (via CM.15) configuration registers for setup purpose. To initialize the host, SW clears its error count and Rx Voltage and Timing Margining registers.

2. SW sets host's Operational Mode register d7:d0 to 00001010b to select upstream direction, TP(1), and directs host to enter Rx Margining mode.
3. SW, via CM.15, sets peripheral's Operational Mode register d7:d0 to 10001010b to select upstream direction, TP(1), and directs peripheral to enter Rx Margining mode. Peripheral waits $T_{HSXIPDSD}$ and sends TP(1) and then resets d7 back to 0.
4. Depending on margining target BER and threshold, SW may optionally reset the error count register. SW sets peripheral's Operational Mode register d7 to 1 (all other bits remain the same) to trigger the sending of TP(1) again. Peripheral waits $T_{HSXIPDSD}$ and sends TP(1). Once the pattern is sent, the peripheral resets the bit to 0.

3.8.6.2 Peripheral Receive Margining

An eUSB2V2 peripheral receiver shall support receive margining in the Default state. Figure 3-14 below shows an example flow of software driven peripheral receive margining. Note that both the host and peripheral are put into the Rx Margining Mode. The Direction bits of the Operational Mode registers are set to 1 such that the host is transmitting while the peripheral is receiving. The test pattern should be TP(1), which is 3 million bits of PRBS16 pattern. The Trigger bit is used to direct the host to transmit the test pattern. Note that dwell time is defined by the margining software. It is the time allocated to count a desired number of bits before changing the margining offset. The longer the dwell time, the lower the bit error rate the margining can achieve.

Figure 3-14. Software Driven Peripheral Receive Margining Example Flow


Peripheral receive margining
SW reads PHY's configurations and initialize peripheral for Rx margining, see note 1.
SW puts peripheral into Rx Margining mode, see note 2.
SW puts host into Rx Margining mode and trigger the test pattern, see note 3.
SW optionally resets the error count register and trigger the test pattern, see note 4.
Read error count using CM.15. Above threshold?
No
Test dwell time expired?
No
Yes
Margining offset at limit?
Yes
No
Update margining offset via CM.15
Yes
Set margining offset back to nominal, clear error count via CM.15. Report margin result
End Peripheral receive margining
V-015

Notes:

1. SW directs link into Default, reads host and peripheral (via CM.15) configuration registers for setup purpose. To initialize the peripheral, SW clears its error count and Rx Voltage and Timing Margining registers.
2. SW sets peripheral's Operational Mode register d7:d0 to 00001110b to select downstream direction, TP(1), and directs peripheral to enter Rx Margining mode.
3. SW sets host's Operational Mode register d7:d0 to 10001110b to select downstream direction, TP(1), and directs host to enter Rx Margining mode. Host waits $T_{HSXIPDSD}$ and sends TP(1) and then resets d7 back to 0.
4. Depending on margining target BER and threshold, SW may optionally reset the error count register. SW sets host's Operational Mode register d7 to 1 (all other bits remain the same) to trigger sending of TP(1) again. Host waits $T_{HSXIPDSD}$ and sends TP(1). Once the pattern is sent, the host resets the bit to 0.

3.9 Configuration and Operational Mode Registers

To support eUSB2V2 normal operation and electrical tests, the following configuration and operational mode registers are defined.

These registers are mandatory for host, re-drivers and peripherals, and they shall be accessed via CM.14 for re-drivers and CM.15 for host and peripherals. Refer to Section 2.3.

When testing a host, a test equipment shall start as a CM receptor and acknowledge CM.15 messages initiated by a host. The equipment, however, shall not complete Port Configuration handshake. Furthermore, it is expected a test personnel will use host specific method to put it into Compliance Mode and flip it to a CM receptor. Then the equipment can be turned into a CM initiator and proceed with testing.

When testing a peripheral, a test equipment shall start as a CM initiator and periodically send CM.15 message, e.g. Read the Data Rate register, to the peripheral under test. The equipment shall proceed with testing upon acknowledgement from the peripheral.

Note: The CM.RAP definition does not include error detection mechanism. Developers are advised to read back the register for debug purposes, if needed.

3.9.1 Register Addresses

The register Address column referred to in Table 3-11 conforms to the 6-bit address field defined in the eUSB2 specification [3], section 6. Table 3-11 below shows the details of the eUSB2V2 assignments.

Table 3-11. Configuration and Operational Mode Register Definitions

Address	Host	Peripheral	Re-driver	Register Description	Type	Note
0	N/A	Y	Y	Vendor ID Low	Read-only	
1	N/A	Y	Y	Vendor ID High	Read-only	
2	N/A	Y	Y	Product ID Low	Read-only	
3	N/A	Y	Y	Product ID High	Read-only	
4	Y	Y	N/A	Data Rate (Section 3.9.2)	Read/Write	
5	Y	Y	N/A	Operational Mode (Section 3.9.3)	Read/Write	1
6	Y	Y	N/A	Error Count	Read/Write	
7	Y (Downstream)	Y (Upstream)	Y (Upstream)	Tx Configuration (Section 3.9.5)	Read/Write	
8	N/A	N/A	Y (Downstream)	Tx Configuration (vendor defined)	Read/Write	
9	N/A	N/A	Y (Upstream)	Rx Configuration (vendor defined)	Read/Write	
10	Y (Upstream)	Y (Downstream)	Y (Downstream)	Rx Configuration (Section 3.9.6)	Read/Write	
11	Y	Y	N/A	Rx Voltage Margining (Section 3.9.7)	Read/Write	
12	Y	Y	N/A	Rx Timing Margining (Section 3.9.8)	Read/Write	
13:31				Reserved	Read/Write	
32:63				Vendor defined	Read/Write	

Notes:
1. Test Mode write shall trigger entry into test mode.

Table 3-12. CM Register Field Assignment

RAP	**Data**								**Host**	**Peripheral**	**Re-driver**	**Register**
Address	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**CM.15**	**CM.15**	**CM.14**	
0	Vendor ID Low								N/A	Y	Y	Vendor ID
1	Vendor ID High								N/A	Y	Y	
2	Product ID Low								N/A	Y	Y	Product ID
3	Product ID High								N/A	Y	Y	
4	DS Rate				US Rate				Y	Y	N/A	Data Rate[1]
5	Trig	DScr	TP			Dir	Mode		Y	Y	N/A	Operational Mode[2]
6	Count								Y	Y	N/A	Error Count[2]
7				DE		Vswing			Y (Downstream)	Y (Upstream)	Y (Upstream)	TX Config
8									N/A	N/A	Y (Downstream)	TX Config
9									N/A	N/A	Y (Upstream)	RX Config
10		VGA			CTLE				Y (Upstream)	Y (Downstream)	Y (Downstream)	RX Config
11	Voltage Margin Offset								Y	Y	N/A	Rx Voltage Margining[2]
12	Timing Margin Offset								Y	Y	N/A	Rx Timing Margining[2]
31:13	Reserved											Reserved
63:32	Vendor Defined											Vendor Defined
	Note 1: This shall only apply during Compliance Mode or Rx Margining Mode. Note 2: Default value is 0.											

Table 3-13. Host Configuration and Operational Mode Register Set Applicability

Register Fields	Compliance Mode	Rx Margining Mode (for Host)	Rx Margining Mode (for Peripheral)	Functional Mode	Debug Operation in Functional Mode
Register Access Interface	CM.RAP	Implementation specific	Implementation specific	Implementation specific	Implementation specific
PHY State when registers are configured	Default. Compliance Mode	Default. Rx Margining Mode	Default. Rx Margining Mode	Default. PHY Configuration	Default. PHY Configuration
US Rate	Yes	Yes	Yes	No	No
DS Rate	Yes	Yes	Yes	No	No
Mode	Yes (Compliance Mode)	Yes (Functional Mode, Rx Margining Mode)	Yes (Functional Mode, Rx Margining Mode)	No (Functional Mode)	No (Functional Mode)
Dir	Yes	Yes (Upstream)	Yes (Downstream)	No	No
TP	Yes	Yes	Yes	No	No
DScr	Ignored	Ignored	Ignored	No	Yes
Trig	Yes	No	Yes	No	No
Count	Yes	Yes	No	No	No
Vswing	Yes	Yes	Yes	Yes	Yes
DE	Yes	Yes	Yes	Yes	Yes
CTLE	Yes	Yes	Yes	Yes	Yes
VGA	Yes	Yes	Yes	Yes	Yes
Voltage Margin offset	No	Yes	No	No	No
Timing Margin offset	No	Yes	No	No	No
Vendor Defined	Vendor Defined	Vendor Defined	Vendor Defined	Vendor Defined	Vendor Defined
	Notes: 1. “Yes” = Register setting may be changed 2. “No” = Register setting shall not be changed 3. Content in parenthesis is the expected register setting 4. “Ignored” = Register setting is ignored by the PHY				

Table 3-14. Peripheral Configuration and Operational Mode Register Set Applicability

Register Fields	Compliance Mode	Rx Margining Mode (for Host)	Rx Margining Mode (for Peripheral)	Functional Mode	Debug Operation in Functional Mode
Register Access Interface	CM.RAP	CM.RAP	CM.RAP	CM.RAP (Optional)	CM.RAP (Optional)
PHY State when registers are configured	Default. Compliance Mode	Default. Rx Margining Mode	Default. Rx Margining Mode	Default. PHY Configuration	Default. PHY Configuration
US Rate	Yes	Yes	Yes	No	No
DS Rate	Yes	Yes	Yes	No	No
Mode	Yes (Compliance Mode)	Yes (Rx Margining Mode)	Yes (Rx Margining Mode)	No (Functional Mode)	No (Functional Mode)
Dir	Yes	Yes (Downstream)	Yes (Upstream)	No	No
TP	Yes	Yes	Yes	No	No
DScr	Ignored	Ignored	Ignored	No	Yes
Trig	Yes	No	Yes	No	No
Count	Yes	Yes	No	No	No
Vswing	Yes	Yes	Yes	Yes	Yes
DE	Yes	Yes	Yes	Yes	Yes
CTLE	Yes	Yes	Yes	Yes	Yes
VGA	Yes	Yes	Yes	Yes	Yes
Voltage Margin offset	No	Yes	No	No	No
Timing Margin offset	No	Yes	No	No	No
Vendor Defined	Vendor Defined	Vendor Defined	Vendor Defined	Vendor Defined	Vendor Defined
	Notes: 1. “Yes” = Register setting may be changed 2. “No” = Register setting shall not be changed 3. Content in parenthesis is the expected register setting 4. “Ignored” = Register setting is ignored by the PHY				

3.9.2 Data Rate Register

Since eUSB2V2 supports asymmetric data rate, all 8 bits are needed to fully address the transceiver's data rates.

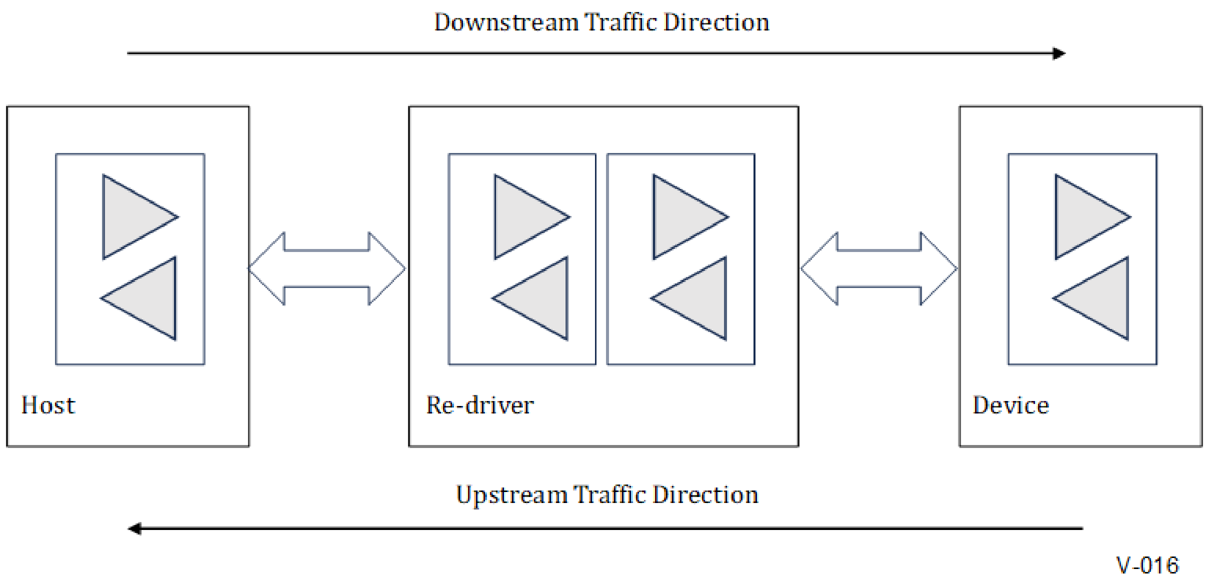
Table 3-15. Downstream Data Rate (DS Rate)

d7	d6	d5	d4	Downstream Data Rate
0	0	0	0	Reserved
0	0	0	1	HS1
0	0	1	0	HS2
0	0	1	1	HS3
0	1	0	0	HS4
0	1	0	1	HS5
0	1	1	0	HS6
0	1	1	1	HS7
1	0	0	0	HS8
1	0	0	1	HS9
1	0	1	0	HS10
1011 and higher				Reserved

Table 3-16. Upstream Data Rate (US Rate)

d3	d2	d1	d0	Upstream Data Rate
0	0	0	0	Reserved
0	0	0	1	HS1
0	0	1	0	HS2
0	0	1	1	HS3
0	1	0	0	HS4
0	1	0	1	HS5
0	1	1	0	HS6
0	1	1	1	HS7
1	0	0	0	HS8
1	0	0	1	HS9
1	0	1	0	HS10
1011 and higher				Reserved

Figure 3-15. Traffic Direction



For example, if a peripheral data rate register is d7:d0 = 00011010b, it is receiving at HS1 speed and transmitting at HS10 speed. The same register setting for the host, however, means it is receiving at HS10 and transmitting at HS1. Since a re-driver simply re-drives the signal to achieve longer channel length, data rate setting is not applicable.

Note: d7:d0 = 00010001b is invalid. Also, the default values of the host and the peripheral Data Rate registers are implementation specific, and it is the system designer's responsibility to properly configure them. Furthermore, changing data rate likely requires changing the PHY's PLL frequency and thus the operation requires some time to finish. Developers shall allow at least 10ms before the next operation.

3.9.3 Operational Mode Register

The Operational Mode register is used to set a component into different modes of operation. Mode of Operation (Mode) is a 2-bit field that identifies the mode of operation, Dir is a 1-bit field that is used for identifying upstream vs. downstream traffic direction for a test, TP is a 3-bit field that identifies the test pattern, DScr is a 1-bit field that disables the scrambler/de-scrambler (only applicable in Functional Mode). TP Trigger is a 1-bit field that triggers the test pattern to be sent. This bit shall be set to 1 whenever the selected test pattern needs to be sent, and it shall be reset back to 0 by the PHY once the transmission of the test pattern is complete.

Note that there exists a debug operation in functional mode, where the link enters L0 with special PHY configurations, that includes, but is not limited to, disabling the scrambler and changing the transceiver registers.

Table 3-17. Mode of Operation (Mode)

d1	d0	Mode of Operation
0	0	Functional Mode (default)
0	1	Compliance Mode
1	0	Rx Margining Mode
1	1	Reserved

Table 3-18. Traffic Direction (Dir)

d2	Traffic direction
0	Upstream (Peripheral Tx and Host Rx)
1	Downstream (Host Tx and Peripheral Rx)

Table 3-19. Test Pattern (TP)

d5	d4	d3	Test Pattern	Scrambling	Bit-stuffing	NRZI
0	0	0	3 million bits of 0 and 1, 1 shot.	No	No	No
0	0	1	PRBS16 + Bit-stuffing + NRZI, 3 million bits, 1 shot.	Yes	Yes	Yes
0	1	0	(PRBS16 + Bit-stuffing + NRZI, 8192 bits) repeat 1k times with $T_{HSXIPDSD}$ inter-packet delay in between.	Yes	Yes	Yes
0	1	1	(PRBS7, 1k bits) repeat 10k times with $T_{HSXIPDSD}$ inter-packet delay in between.	Yes	No	No
1	0	0	(PRBS7, 8 bits) repeat 100k times with $T_{HSXIPDSD}$ inter-packet delay in between.	Yes	No	No
1	0	1	(64 bits of 0 + 64 bits of 1) repeat 1k times, 1 shot.	No	No	No
1	1	0	Reserved			
1	1	1	Reserved			

Notes:
All patterns begin with the Sync field and end with EOP.
For PRBS 16, the scrambling function is $G(X)=X^{16}+X^5+X^4+X^3+1$ with FFFFh initialization seed. The input bit pattern is all zero .
For PRBS7, the scrambling function is $G(X)=X^7+X^6+1$ with FFh initialization seed. The input bit pattern is all zero.

Table 3-20. Disable Scramble/De-scrambler (DScr)

d6	Disable Scrambler/De-scrambler
0	Enabled (default)
1	Disabled

Table 3-21. TP Trigger

d7	TP Trigger
0	Idle (The PHY reset this bit to here after it was set to 1 and the test pattern is sent)
1	Triggered
Notes: During Tx compliance test, test equipment may set this bit to 1 via CM.15 to trigger the selected test pattern to be sent. During host Rx margining, software may direct the host PHY to set this bit on the peripheral's register to trigger the test pattern to be sent.	

3.9.4 Error Count Register

This 8-bit register is defined to store error count during Rx compliance test or Rx margining.

Table 3-22. Error Count

Bits	Field Description	Note
7:0	Error count	Does not roll over. Once it reaches the maximum value of FFh, it does not change until being reset by the PHY or explicitly written to 0.

3.9.5 Transmit Configuration Register

This 8-bit register configures the transmitter's voltage swing (Vswing) and de-emphasis settings. Three bits, d7, d6, and d5 are reserved.

Table 3-23. Transmit Voltage Swing (Vswing)

d2	d1	d0	Vswing (mVdiff-pp)
0	0	0	360
0	0	1	400
0	1	0	500
0	1	1	600
1	0	0	700
1	0	1	800
1	1	0	Reserved
1	1	1	Reserved
Note: While these settings are aligned with values in Table 3-3, they are informative.			

Table 3-24. Transmit De-emphasis (DE)

d4	d3	Tx De-emphasis (dB)
0	0	0
0	1	-3.5
1	0	-6
1	1	-8
Note: While these settings are aligned with values in Table 3-3, they are informative.		

3.9.6 Receive Configuration Register

This 8-bit register configures the receiver's equalization. Bit d7 is reserved.

Table 3-25. Receive CTLE Boost (CTLE)

d3	d2	d1	d0	CTLE Boost (dB)
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
Note: 1001 to 1111 are reserved. While these settings are aligned with values on Table 3-6, they are informative.				

Table 3-26. Receive Variable Gain Amplifier (VGA)

d6	d5	d4	VGA gain multiplier
0	0	0	1
0	0	1	1.25
0	1	0	1.5
0	1	1	1.75
1	0	0	2
1	0	1	2.25
1	1	0	2.5
1	1	1	Reserved
Note: While these settings are aligned with values on Table 3-6, they are informative.			

Note: To accommodate different design complexities of the receive equalization circuity, the settings of Table 3-25 and Table 3-26 are recommendation only as long as the strongest 2.5x VGA gain plus 8dB CTLE boost can be met by the specific design implementation.

3.9.7 Rx Voltage Margining Register

This 8-bit register provides the mechanism for software-controlled receive voltage margining capability.

Table 3-27. Rx Voltage Margining Register

Bits	Field Description	Values
7:0	Voltage Margin Offset (signed)	Vendor specific

3.9.8 Rx Timing Margining Register

This 8-bit register provides the mechanism for software-controlled receive timing margining capability.

Table 3-28. Rx Timing Margining Register

Bits	Field Description	Values
7:0	Timing Margin Offset (signed)	Vendor specific

4 eUSB2V2 Topology and Transfer Types

This section defines the enhancements necessary to the USB2 specification to support eUSB2V2 functionality.

4.1 Bus Topology

The bus topology of eUSB2V2 is similar the USB 2.0 model as described in section 5.2 of the USB 2.0 specification [1]. The significant difference is with regards to the eUSB2V2 physical topology limitations.

4.1.1 Physical Bus Topology

USB 2.0 provides for fanout and distance via hub devices. Each hub provides fanout to more devices. And the cabling between host and each hub level enables long distances between a device and a host port.

eUSB2V2 targets embedded platform use cases where large fanouts are not needed because the total number of embedded peripherals will tend to be small. In addition, the distance available with a single physical link is sufficient for typical mobile platform embedded use cases.

The physical bus topology options supported by eUSB2V2 are illustrated in Figure 4-1.

An eUSB2V2 *device* may be a re-driver or peripheral.

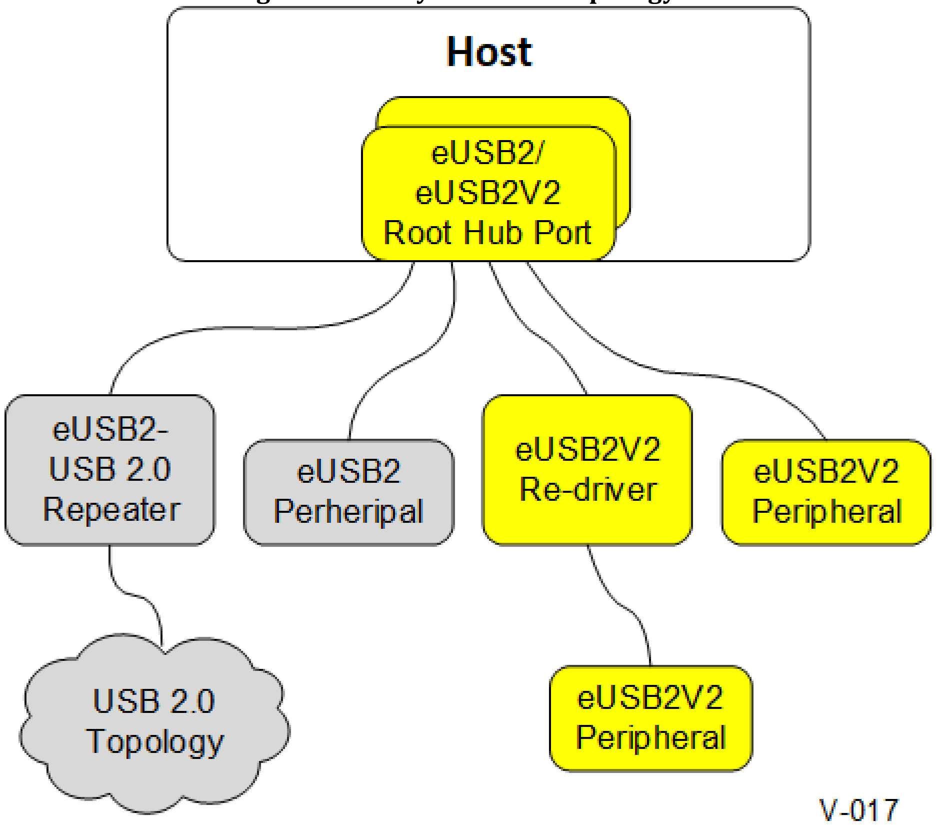
To reduce Inter-packet and Bus Turnaround delays, eUSB2V2 supports physical topologies of at most one re-driver between the host and peripherals.

eUSB2V2 re-driver enables longer distances between an eUSB2V2 host and a peripheral.

Multiple functions may be packaged together in what appears to be a single eUSB2V2 physical device as a composite device.

A composite device presents multiple interfaces that are controlled independently of each other as separate functions. A composite device has only a single device address.

Figure 4-1. Physical Bus Topology



The host shall consider each downstream facing eUSB2V2 port as a full eUSB2V2 bus instance for scheduling and bandwidth calculations. Each eUSB2V2 port shall support the maximum eUSB2V2 bandwidth (depending on the link data rate) for the respective transfer types, as described in Section 4.3. And traffic on a eUSB2V2 port is completely independent from other host ports.

There is no change in eUSB2V2 to the client software-to-function relationship as defined in the USB 2.0 specification, section 5.2.4.

4.1.2 Logical Bus Topology

The eUSB2V2 logical bus is identical to the logical Bus Topology described in section 5.2.4 of the USB 2.0 specification [1].

4.2 eUSB2V2 Communication Flow

The eUSB2V2 communication flow is identical to the USB communications flow described in the USB2 specification with the following exceptions for eUSB2V2:

- Increases the overall data throughput by providing a data rate of up to 4.8Gb/s. eUSB2V2 links are also referred to as HSSx-speed, HSUx-speed, or HSDx-speed links.
- Increases the isochronous endpoint bandwidth over USB2 by allowing the transfer of over 3072B/microframe.
- Extends packet size characteristics:
    - If transferring more than 1024B of data, eUSB2V2 isochronous endpoints are required to use 1024B data packets, except for the last one.
    - The MaxPacketSize for eUSB2V2 bulk pipes are increased to 1024B when operating with eUSB2V2 compliant software.

Like USB2, no more than 90% of any frame shall be allocated for periodic (isochronous and interrupt) transfers for eUSB2V2 endpoints, ensuring that non-periodic endpoints are not starved. The system software allocates this bandwidth across all configured Isochronous and Interrupt EPs attached to a port.

4.3 Periodic Bandwidth Allocation

eUSB2V2 uses the *eUSB2 Isochronous Endpoint Companion* descriptor to define the bandwidth allocation of an isochronous endpoint. The descriptor allows definition of bandwidth allocations greater than 3072B per service interval (SI) for isochronous endpoints.

Section 5.9 in the USB2 specification [1] defines the concept of ‘High-Speed, High Bandwidth Endpoints’, where the bandwidth per SI of a high-speed periodic endpoint is defined by the standard Endpoint descriptor as; “The lower 11 bits of wMaxPacketSize indicate the size of the data payload for each individual transaction while bits 12..11 indicate the maximum number of required transactions possible.” Bits[12:11] define a multiplier that is applied to bits[10:0] to determine the periodic bandwidth allocation of an endpoint. For further discussion in this document, bits[10:0] of the wMaxPacketSize field shall be referred to as the *MaxPacketSize* field, and the 2-bit field (bits[12:11]) shall be referred to as the *High-Bandwidth* field.

This approach overloaded the USB 1.1 standard Endpoint descriptor definition of wMaxPacketSize field to enable high-speed periodic bandwidth allocations of greater than 1024B, however, the definition can result in inefficient data transfers. For instance, to allocate 1536B of periodic bandwidth, the MaxPacketSize is set to 768 and the High-Bandwidth field set to 1, i.e., up to two 768 B packets per SI. To define a 1536B periodic bandwidth requirement, meant defining a sub-optimal MaxPacketSize value, causing the endpoint to generate a second transaction when transferring between 769B and 1024B.

eUSB2V2 uses the *eUSB2 Isochronous Endpoint Companion* descriptor dwBytesPerInterval field to explicitly define the bandwidth requirements of an isochronous endpoint and requires the value of the endpoint companion wMaxPacketSize field to be set to 1024B for any endpoint that requires the allocation of more than 1024B of periodic bandwidth. This approach minimizes the number of transactions when variable size data transfers are generated.

For eUSB2V2 peripherals, refer to section 5.1.2 for how to configure the values of the MaxPacketSize and High-Bandwidth fields of the standard Endpoint descriptor.

The *eUSB2 Isochronous Endpoint Companion* descriptor is described in section 5.1.4.

4.4 Control Transfers

All aspects of eUSB2V2 control endpoints are identical to those defined by the USB2 specification.

4.5 Bulk Transfers

Bulk transfers operate identical to those defined in the USB2 specification [1], except for the MaxPacketSize when operating with eUSB2V2 compliant software. Refer to section 5.1.

4.6 Isochronous Transfers

eUSB2V2 isochronous transfers use the *eUSB2 Isochronous Endpoint Companion* descriptor dwBytesPerInterval field to define its periodic bandwidth requirement.

Refer to section 5.1.1 for how to set the values of the MaxPacketSize and *High-Bandwidth* fields of a standard isochronous Endpoint descriptor.

Two eUSB2V2 features enable isochronous endpoints to take advantage of its higher bandwidth capabilities:

- The ability to allocate more than 3072B of bandwidth per microframe.
- Burst transactions.

The USB2 specification limited isochronous bandwidth allocation to 3072B per microframe. eUSB2V2 endpoints use the *eUSB2 Isochronous Endpoint Companion* descriptor to define bandwidth allocations for isochronous transfers. Refer to section 5.1.4 for more information on the *eUSB2 Isochronous Endpoint Companion* descriptor.

eUSB2V2 isochronous endpoints execute burst transactions. A burst transaction allows a single transaction to transfer up to 3 consecutive data packets, thus eliminating n/3 of the bus turnarounds that are associated with transferring n data packets using standard isochronous transactions. eUSB2V2 burst transaction management is handled entirely by the host controller. They are automatically enabled for data transactions greater than 1024B. Refer to section 6.4 for more information on isochronous burst transactions.

4.6.1 Burst Transactions

To minimize bus turnarounds, eUSB2V2 defines Isochronous Burst transactions. An eUSB2V2 Burst transaction can transfer 1, 2 or 3 data packets in a single transaction, while a standard USB2 transaction only transfers 1 data packet per transaction.

Burst IN transactions cycle through DATA2, DATA1, and DATA0 PIDs in descending order to allow the consecutive data packets within a transaction to be distinguished, where DATA0 is always assigned to the last data packet transferred. Burst OUT transactions always send MDATA PIDs until the last data packet which can be one of DATA2/DATA1/DATA0.

The maximum number of data packets that a host or peripheral can generate in a burst transaction (Burst Count) is not defined in the *eUSB2 Isochronous Endpoint Companion* descriptor. But it is derived from the number of bytes remaining for the transfer, where:

If number of bytes remaining <= 1024, then Burst Count = 1
If number of bytes remaining > 1024 and <= 2048, then Burst Count = 2
If number of bytes remaining > 2048, then Burst Count = 3

One or more burst transactions may be executed to transfer isochronous data during a service interval (SI). The set of burst transactions executed during a service interval is referred to as the ‘burst sequence’.

All burst transactions of a burst sequence shall contain 3 data packets except for the last burst transaction. The last burst transaction may be a ‘short burst transaction’. A ‘short burst transaction’ is a burst transaction which transfers less than 3072B of data.

All data packets of a burst sequence shall be 1024B in size, except for the last data packet of a short burst transaction. The last data packet of a short burst transaction shall be a ‘short packet’. A ‘short packet’ is a data packet that is not wMaxPacketSize (1024B) bytes in size. (Note that a special case of a short packet is a packet with zero-length payload.) Refer to section 5.3.2 in the USB2 specification for more information on ‘short’ packets.

A data error is indicated in a service interval if any of the following conditions are true:

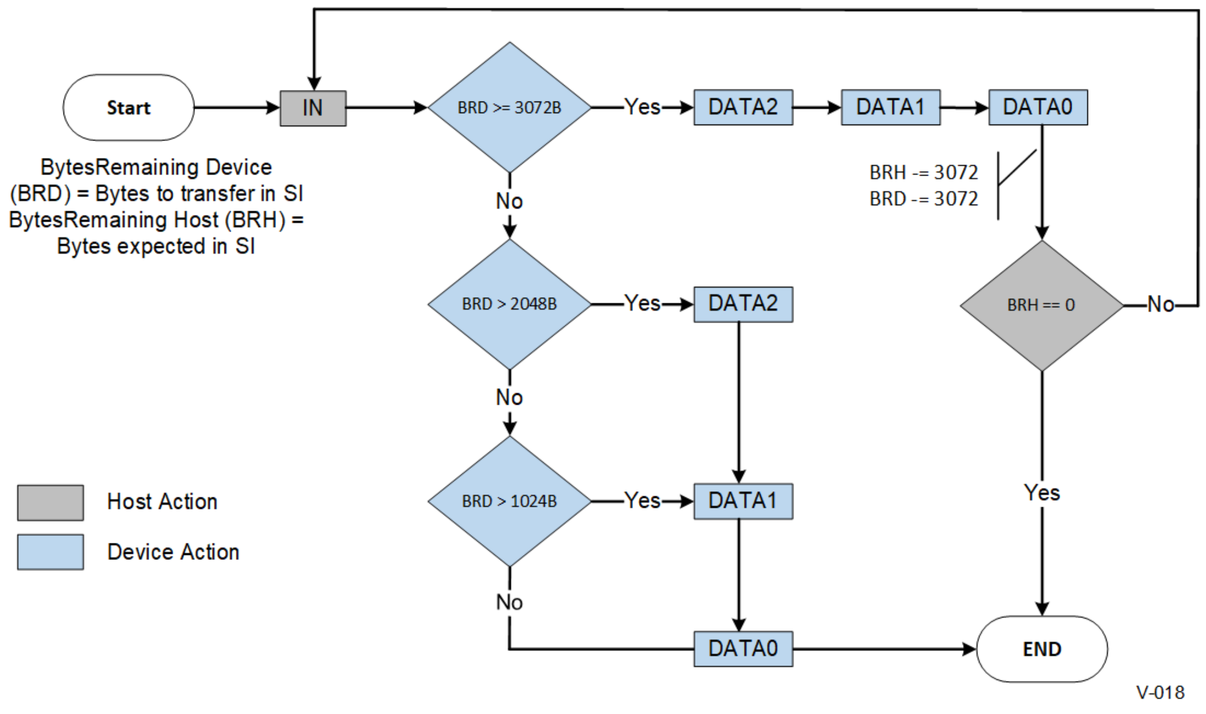
- Any of the error conditions described in section 8.7 of the USB2 specification [1].
- A missing data PID in the sequence of expected data packets within a burst transaction.
- The service interval ends without a terminating short burst transaction being received for a burst sequence, and the number of bytes received is less than dwBytesPerInterval.

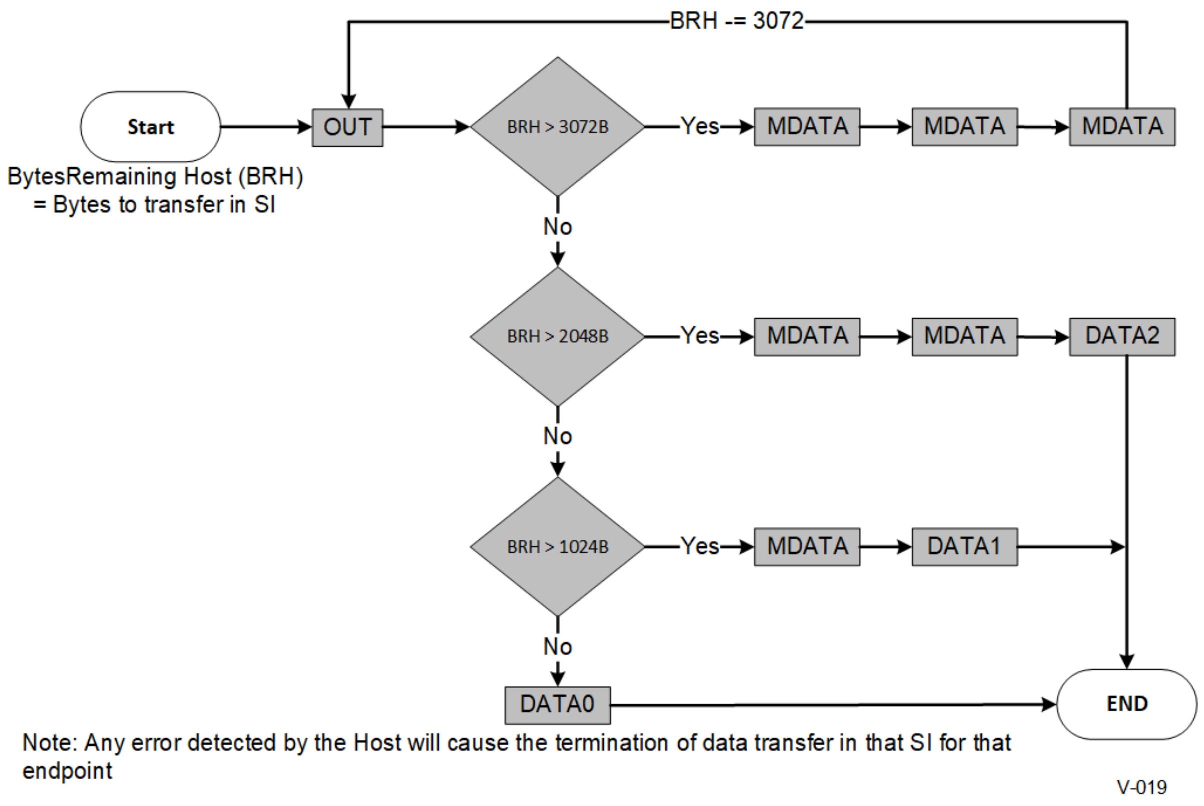
If an error is detected, then all data for the service interval is considered bad and dropped.

The last data PID of a IN burst sequence shall be DATA0. The last data PID of a OUT burst sequence shall be DATA2/DATA1/DATA0.

Figure 4-2 illustrates the packets of an IN or OUT burst transaction to an endpoint with a 3 data packet burst.

Figure 4-2. Isochronous Burst Transactions




BRH -= 3072
Start
OUT
BRH > 3072B
Yes
MDATA
MDATA
MDATA
BytesRemaining Host (BRH)
= Bytes to transfer in SI
No
BRH > 2048B
Yes
MDATA
MDATA
DATA2
No
BRH > 1024B
Yes
MDATA
DATA1
No
DATA0
END
Note: Any error detected by the Host will cause the termination of data transfer in that SI for that endpoint
V-019

If a PID Check, Bit Stuff, or Data CRC error is detected on a data packet, the endpoint shall drop the packet data and transition to the HSx idle state and drop all subsequent data packets until the end of the service interval. This is the 'Error' path in Figure 4-2.

The data packet of a burst transaction <= 1024B shall use a DATA0 PID.

For IN transactions, if 1025B to 2048B are transferred by a burst transaction, then only DATA1 and DATA0 data packets shall follow the Token packet. For OUT transactions, a host shall send an MDATA packet followed by a DATA1 packet.

For IN transaction, If 2049B to 3072B are transferred by a burst transaction, then only DATA2, DATA1 and DATA0 data packets shall follow the Token packet. For OUT transactions, a host shall send two MDATA packets followed by a DATA2 packet.

The last burst transaction of an isochronous IN burst sequence shall be a short burst sequence unless dwBytesPerInterval have been transferred. The last burst transaction for an isochronous OUT burst sequence shall include a DATA2/DATA1/DATA0 packet.

4.7 Interrupt Transfers

All aspects of eUSB2V2 interrupt endpoints are identical to those defined by the USB2 specification.

5 USB Device Framework

This section defines the framework differences between standard USB and eUSB2V2.

5.1 Standard USB Descriptor Definitions

This section describes the descriptors introduced by eUSB2V2. It defines an approach that enables a peripheral to present a single set of configuration descriptors that will work with legacy or eUSB2V2 aware system software.

Note, the expectation is that the peripheral knows what speed that it connects at, and adjusts the configurations that it presents accordingly, i.e. different configuration bandwidth settings are defined depending on whether the peripheral is connected at HSSx-speed, HSUx-speed, or HSDx-speed.

5.1.1 Standard Device Descriptor

The bcdUSB field in a eUSB2V2 compliant peripheral shall be set to 0x230.

The maximum packet size for endpoint zero (bMaxPacketSize0) field shall be set to 64B.

5.1.2 Standard Endpoint Descriptor

For control, bulk, and interrupt endpoints, the standard Endpoint descriptor shall follow the standard Endpoint descriptors definitions as defined in the USB2 specification.

For isochronous endpoint, the standard Endpoint descriptor shall declare a zero bandwidth setting, i.e., the wMaxPacketSize field shall be set to 0, and the actual maximum packet size and bandwidth requirements shall be defined in the eUSB2 Isochronous Endpoint Companion descriptor wMaxPacketSize and dwBytesPerInterval fields, respectively.

5.1.3 Standard Interface Descriptor

To support legacy software, an eUSB2V2 peripheral (with isochronous endpoints) shall include at least one Alternate Interface that shall declare all isochronous endpoints within it to have a maximum packet size of 1024B with a Mult of 3. Note that these isochronous endpoints shall follow the eUSB2V2 protocol even when transferring less than 3072B of data when this Alternate Interface is selected.

5.1.4 eUSB2 Isochronous Endpoint Companion Descriptor

This descriptor may only be returned by isochronous endpoints of eUSB2V2 peripherals.

The standard Endpoint descriptor shall be followed by an *eUSB2 Isochronous Endpoint Companion* descriptor for each Isochronous Endpoint. An *eUSB2 Isochronous Endpoint Companion* descriptor for an endpoint shall be declared before the next endpoint or interface descriptor is declared. This descriptor is 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 wMaxPacketSize and the dwBytesPerInterval field values have the following descriptions in a *eUSB2 Isochronous Endpoint Companion* descriptor when used in a eUSB2V2 peripheral.

Table 5-1: eUSB2 Isochronous Endpoint Companion Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor in bytes (8).
1	bDescriptorType	1	Constant	EUSB2_ISO_ENDPOINT_COMPANION Descriptor Type.
2	wMaxPacketSize	2	Number	The maximum packet size this endpoint is capable of sending or receiving when this configuration is selected. For isochronous endpoints this field shall be set to 1024 unless the isochronous endpoint is transferring less than 1024B per Service Interval (SI).
4	dwBytesPerInterval	4	Number	The maximum number of bytes this endpoint will transfer per Service Interval. This value is used to reserve the bus time in the schedule, required for the frame data payloads per SI. The pipe may, on an ongoing basis, actually use less bandwidth than that reserved. The peripheral reports, if necessary, the actual bandwidth used via its normal, non-USB defined mechanisms. If this field is greater than 1024B, then the SI shall be set to 1 microframe.

5.2 Standard Device Requests

A eUSB2V2 peripheral shall support all the standard USB 2.0 device requests and features. In addition, all eUSB2V2 peripherals that support Bulk endpoints shall support the BULK_MAX_PACKET_UPDATE Feature Selector.

This request is used to set the maximum packet size of all Bulk endpoints in the peripheral to 1024B.

Table 5-2: SetFeature (BULK_MAX_PACKET_UPDATE)

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00000000B	SET_FEATURE	BULK_MAX_PACKET_UPDATE	0	Zero	None

Default State:	This request is valid in the Default state.
Address State:	This request is valid in the Addressed state.
Configured State:	This request is not valid in the Configured state.

A eUSB2v2 Bulk endpoint shall revert the maximum packet size to 512 bytes when any of the following happens:

- USB 2.0 Bus Reset
- Disconnect
- Deconfiguration

Note that the wMaxPacketSize filed shall always be set to 512 in the Endpoint descriptor, independent of the current maximum packet size being 512 or 1024.

5.3 Standard Device Feature Selector

Table 5-3: Standard Device Feature Selector

Feature Selector	Recipient	Value
BULK_MAX_PACKET_UPDATE	Device	8

6 Data Flow Model

This chapter presents information about how data is moved across the USB by eUSB2V2. The information in this chapter affects all implementers. The information presented is at a level above the signaling and protocol definitions of the system.

6.1 USB Communication Flow

The communication flow in eUSB2V2 is the same as defined for USB 2.0 in section 5.3 of the USB 2.0 specification.

6.2 Transfer Types

eUSB2V2 preserves the transfer type architecture and data flow as defined in section 5.4 of the USB 2.0 specification.

6.3 Control Transfers

The semantics of eUSB2V2 control transfers is identical to those described in the USB 2 specification [1], section 5.5.

6.4 Isochronous Transfers

Isochronous transfers in this specification follow most of the architecture of USB 2.0 isochronous transfers described in section 5.6 of the USB 2.0 Specification. This section focuses on the small set of differences.

The isochronous transfer type provides the same architectural guaranteed access to USB bandwidth as was provided with USB 2.0. The differences include:

- Isochronous burst transactions may have 1 to 3 data packets in the data phase of the transaction.
- The upper limit of assignable bandwidth to a single isochronous pipe is significantly increased.
- Multiple burst transactions are allowed during a service interval.

An isochronous pipe allows multiple burst transactions per microframe if necessary to meet the isochronous endpoint's bandwidth requirements. The eUSB2V2 bandwidth allocation is defined by the *eUSB2 Isochronous Endpoint Companion* descriptor dwBytesPerInterval field, not the standard Endpoint descriptor wMaxPacketSize field.

The IRP determines the amount of data to be moved during a service interval. If one isochronous data packet associated with an IRP is in error, then the IRP for the service interval is retired.

6.4.1 Isochronous Transfer Size Constraints

Data packet payloads for isochronous pipes are allowed to be up to 1024B. The maximum data packet size for an isochronous endpoint is declared in the *eUSB2 Isochronous Endpoint Companion* descriptor.

Each endpoint in each configuration for an isochronous pipe specifies the maximum amount of data that it can transmit or receive during a service interval. If there is sufficient bus time for the maximum transfer size (dwBytesPerInterval) for all periodic endpoints associated with a configuration, the configuration is established; if not, the configuration is not established.

As described previously, the configuration for an isochronous pipe essentially expresses that dwBytesPerInterval that have been allowed for in the bandwidth grant, i.e., when the isochronous endpoint is configured. The host shall schedule sufficient transactions to deliver the opportunity for all the allocated data packets. The data transmitter will transmit all data packets in *eUSB2 Isochronous Endpoint Companion* descriptor wMaxPacketSize quantity, with the last data packet being 'short', i.e., containing fewer than 1024B, unless dwBytesPerInterval bytes are transferred, and it is a multiple of 1024B.

6.4.2 Isochronous Transfer Bus Access Constraints

eUSB2V2 requires that no more than 90% of any microframe be allocated for periodic (isochronous and interrupt) transfers.

The amount of data a single eUSB2V2 isochronous endpoint can move up depends on the link data rate, and the link and protocol overheads. A eUSB2V2 isochronous endpoint that requires more than 1024B per service interval shall define a wMaxPacketSize of 1024 and the total number of bytes to transfer in the dwBytesPerInterval field.

An eUSB2V2 isochronous endpoint may use multiple burst transactions per microframe to transfer data. See Section 4.5 for more information about multiple transactions per microframe for eUSB2V2 endpoints.

6.4.3 Isochronous Transfer Data Sequences

Isochronous transfers do not support data retransmission in response to errors on the bus. A receiver can determine that a transmission error occurred via the application of the PID Check Field or CRC16 field on the trailing end of a data packet. The low-level USB protocol does not allow handshakes to be returned to the transmitter of an isochronous pipe. Handshakes are returned by 'reliable data' links to inform the transmitter whether a packet was successfully received or not and to enable the retransmission of lost data. Isochronous transfers do not support 'reliable data', because they do not return a handshake. For isochronous transfers timeliness is more important than correctness/retransmission, and, given the low error rates expected on the bus, the isochronous protocol is optimized by assuming transfers normally succeed. Isochronous receivers can determine if, and how much data, is lost during a service interval. Refer to section 5.12 of the USB 2.0 Specification [1] as it describes these USB mechanisms in more detail.

To meet the Service Interval bandwidth requirements of an endpoint, eUSB2V2 isochronous transfers may require multiple burst transactions of multiple data packets. If an error is detected in any packet transferred during a Service Interval, then all the data associated with that endpoint for that Service Interval shall be invalidated. This includes where only a single packet in the Service Interval is corrupted.

The data PID sequencing mechanism across multiple data packets transiting the bus during an isochronous transaction is described in Section 4.6.1.

6.5 Interrupt Transfers

The semantics of eUSB2V2 interrupt transfers is identical to those described in the USB 2 specification [1], section 5.7.

6.6 Bulk Transfers

The semantics of eUSB2V2 bulk transfers is identical (except for the MaxPacketSize, when operating with eUSB2V2 compliant software) to those described in the USB 2 specification [1], section 5.8.

6.7 Bus Access for Transfers

eUSB2V2 has a goal to preserve as much of the established architecture and implementations as possible. At this writing there are no new or different requirements for eUSB2V2 that deviate from section 5.11 of the USB 2.0 Specification.