Universal Serial Bus Power Delivery Specification

Revision 1.0

5 July 2012

Editor

Bob Dunstan	Intel Corporation

Contributors

Bruce Montag	Dell Inc.	
Gary Verdun	Dell, Inc.	
AJ Yang	Foxconn / Hon Hai	
Steve Sedio	Foxconn / Hon Hai	
Alan Berkema	Hewlett Packard	
Lee Atkinson	Hewlett Packard	
Rahul Lakdawala	Hewlett Packard	
Ron Schooley	Hewlett Packard	
Vaibhav Malik	Hewlett Packard	
Walter Fry	Hewlett Packard	
Bob Dunstan	Intel Corporation	PD Chair/Protocol WG Lead
Brad Saunders	Intel Corporation	
Dan Froelich	Intel Corporation	
David Harriman	Intel Corporation	
Harry Skinner	Intel Corporation	
Jervis Lin	Intel Corporation	
John Howard	Intel Corporation	
Leo Heiland	Intel Corporation	
Maarit Harkonen	Intel Corporation	
Paul Durley	Intel Corporation	
Rahman Ismail	Intel Corporation	System Policy Lead
Sarah Sharp	Intel Corporation	
Scott Brenden	Intel Corporation	
Sridharan Ranganathan	Intel Corporation	
Steve McGowan	Intel Corporation	
Toby Opferman	Intel Corporation	
Kenta Minejima	Japan Aviation Electronics Industry Ltd. (JAE)	
Mark Saubert	Japan Aviation Electronics Industry Ltd. (JAE)	
Toshio Shimoyama	Japan Aviation Electronics Industry Ltd. (JAE)	
Daniel H Jacobs	LeCroy Corporation	
Mike Michelett	LeCroy Corporation	
Roy Chestnut	LeCroy Corporation	

Dave Thompson	LSI Corporation	
Geert Knapen	MCCI Corporation	
Chris Yokum	MCCI Corporation	
Marwan Kadado	Microsoft Corporation	
Randy Aull	Microsoft Corporation	
Yang You	Microsoft Corporation	
Pat Crowe	MQP Electronics Ltd.	
Frank Borngräber	Nokia Corporation	
Pekka Leinonen	Nokia Corporation	
Richard Petrie	Nokia Corporation	PD Vice-Chair/Device Policy Lead
Sten Carlsen	Nokia Corporation	Physical Layer WG Lead
Bart Vertenten	NXP Semiconductors	
Robert de Nie	NXP Semiconductors	
Robert Heaton	Obsidian Technology	
Cor Voorwinden	ON Semiconductor	
Edward Berrios	ON Semiconductor	Power Supply WG Lead
Tom Duffy	ON Semiconductor	
Narendra Mehta	Qualcomm, Inc.	
Terry Remple	Qualcomm, Inc.	
Yoram Rimoni	Qualcomm, Inc.	
Atsushi Mitamura	Renesas Electronics Corp.	
Kiichi Muto	Renesas Electronics Corp.	
Masami Katagiri	Renesas Electronics Corp.	
Patrick Yu	Renesas Electronics Corp.	
Peter Teng	Renesas Electronics Corp.	
Steve Roux	Renesas Electronics Corp.	
Tetsu Sato	Renesas Electronics Corp.	
Alvin Cox	Seagate Technology LLC	Cab Con WG Lead
John Hein	Seagate Technology LLC	
Marc Noblitt	Seagate Technology LLC	
Ronald Rueckert	Seagate Technology LLC	
Tony Priborsky	Seagate Technology LLC	
John Sisto	SMSC	
Ken Gay	SMSC	
Mark Bohm	SMSC	
Richard Wahler	SMSC	

Tim Knowlton	SMSC
Fabien FRIESS	ST-Ericsson
Giuseppe Platania	ST-Ericsson
Jean-Francois Gatto	ST-Ericsson
Nicolas Florenchie	ST-Ericsson
Patrizia Milazzo	ST-Ericsson
Anand Dabak	Texas Instruments
Deric Waters	Texas Instruments
Grant Ley	Texas Instruments
Ingolf Frank	Texas Instruments
Ivo Huber	Texas Instruments
Jean Picard	Texas Instruments
Martin Patoka	Texas Instruments
Srinath Hosur	Texas Instruments
Steven Tom	Texas Instruments
Charles Neumann	Western Digital Technologies, Inc.
Curtis Stevens	Western Digital Technologies, Inc.
John Maroney	Western Digital Technologies, Inc.

Revision History

Revision	Comments	Issue Date
1.0	Initial release	5 July, 2012

Please send comments via electronic mail to techsup@usb.org

Contents

List of Tables

List of Figures

1. Introduction

USB has evolved from a data interface capable of supplying limited power to a primary provider of power with a data interface. Today many devices charge or get their power from USB ports contained in laptops, cars, aircraft or even wall sockets. USB has become a ubiquitous power socket for many small devices such as cell phones, MP3 players and other hand-held devices. Users need USB to fulfill their requirements not only in terms of data but also to provide power to, or charge, their devices simply, often without the need to load a driver, in order to carry out "traditional" USB functions.

There are however, still many devices which either require an additional power connection to the wall, or exceed the USB rated current in order to operate. Increasingly, international regulations require better energy management due to ecological and practical concerns relating to the availability of power. Regulations limit the amount of power available from the wall which has led to a pressing need to optimize power usage. It has the potential to minimize waste as it becomes a standard for charging devices that are not satisfied by ***[BC1.2]***.

Wider usage of wireless solutions is an attempt to remove data cabling but the need for "tethered" charging remains. In addition, industry design requirements drive wired connectivity to do much more over the same connector.

USB Power Delivery is designed to enable the maximum functionality of USB by providing more flexible power delivery along with data over a single cable. Its aim is to operate with and build on the existing USB ecosystem; increasing power levels from existing USB standards, for example Battery Charging, enables new higher power use cases such as USB powered Hard Disk Drives (HDDs) and printers.

With USB Power Delivery the power direction is no longer fixed. This enables the product with the power (Host or Peripheral) to provide the power. For example, a display with a supply from the wall can power, or charge, a laptop. Alternatively, USB power bricks or chargers are able to supply power to laptops and other battery powered devices through their, traditionally power providing, USB ports.

USB Power Delivery enables hubs to become the means to optimize power management across multiple peripherals by allowing each device to take only the power it requires, and to get more power when required for a given application. For example battery powered devices can get increased charging current and then give it back temporarily when the user's HDD requires to spin up. Optionally the hubs can communicate with the PC to enable even more intelligent and flexible management of power either automatically or with some level of user intervention.

Finally, USB Power Delivery allows low power cases such as headsets to negotiate for only the power they require. This provides a simple solution that enables USB devices to operate at their optimal power levels.

1.1 Overview

This specification defines how USB Devices may negotiate for more current and/or higher or lower voltages over the USB cable (V_{BUS}) than are defined in the ***[USB2.0]***, ***[USB3.0]***, or ***[BC1.2]***. It allows Devices with greater power requirements than can be met with today's specification to get the power they require to operate from V_{BUS} and negotiate with external power sources (e.g. wall warts). In addition, it allows a Source and Sink to swap roles such that a Device could supply power to the Host. For example, a display could supply power to a notebook to charge its battery.

The USB Power Delivery Specification is guided by the following principles:

1) Works seamlessly with legacy USB Devices
2) Compatible with existing spec-compliant USB cables
3) Minimizes potential damage from non-compliant cables (e.g. 'Y' cables etc.)
4) Optimized for low-cost implementations

1.2 Purpose

The USB Power Delivery specification defines a power delivery system covering all elements of a USB system including: Hosts, Devices, Hubs and Chargers. This specification describes the architecture, protocols, power supply behavior, connectors and cabling necessary for managing power delivery over USB at up to 100W. This specification is intended to be fully compatible and extend the existing USB infrastructure. It is intended that this specification will allow system OEMs, power supply and peripheral developers adequate flexibility for product versatility and market differentiation without losing backwards compatibility.

USB Power Delivery is designed to operate independently of the existing USB bus defined mechanisms currently used to negotiate power which are:

- ***[USB2.0]***, ***[USB3.0]*** in band requests for high power interfaces.
- ***[BC1.2]*** mechanisms for supplying higher power.

Initial operating conditions remain the default USB states as defined in ***[USB2.0]***, ***[USB3.0]*** or ***[BC1.2]***.

- The Downstream port sources ***vSafe5V*** over V_{BUS}.
- The Upstream port consumes power from V_{BUS}.

1.3 Scope

This specification is intended as an extension to the existing ***[USB2.0]***, ***[USB3.0]*** and ***[BC1.2]*** specifications. It addresses only the elements required to implement enhanced power delivery over V_{BUS}. It is targeted at power supply vendors, manufacturers of ***[USB2.0]***, ***[USB3.0]*** and ***[BC1.2]*** Platforms and Devices.

Normative information is provided to allow interoperability of components designed to this specification. Informative information, when provided, may illustrate possible design implementation.

1.4 Conventions

1.4.1 Precedence

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

1.4.2 Keywords

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

1.4.2.1 May

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

1.4.2.2 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.

1.4.2.3 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).

1.4.2.4 Reserved

Reserved is a keyword indicating reserved bits, bytes, words, fields, and code values that are set-aside for future standardization. Their use and interpretation may be specified by future extensions to this specification and 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 shall be ignored by the receiver.

1.4.2.5 Shall

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

1.4.2.6 Should

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

1.4.3 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 Related Documents

- ***[USB2.0]*** – Universal Serial Bus Specification, Revision 2.0, plus ECN and Errata, July 14, 2011 (this includes the entire document release package including the OTG&EH v2.0 and the Micro connector v1.01 specifications). www.usb.org/developers/docs
- ***[USB3.0]*** – Universal Serial Bus 3.0 Specification, Revision 1 plus ECN and Errata, July 29 2011 (this includes the entire document release package including the OTG&EH v3.0 specification). www.usb.org/developers/docs.
- ***[BC1.2]*** – Battery Charging Specification, Revision 1.2 plus Errata, October 12, 2011 (referred to in this document as the Battery Charging specification). www.usb.org/developers/devclass_docs#approved.
- ***[Maxim37]*** – Maxim Engineering Journal, Volume 37, page 12 http://pdfserv.maxim-ic.com/en/ej/EJ37.pdf.

1.6 Terms and Abbreviations

This section defines terms used throughout this document. For additional terms that pertain to the Universal Serial Bus, see Chapter 2, "Terms and Abbreviations," in ***[USB2.0]***, ***[USB3.0]*** and ***[BC1.2]***.

Table 1-1 Terms and Abbreviations

Term	Description
Adapter	Adapters have one plug and one receptacle. The plug may be mechanically joined either directly to the receptacle or joined via a cable.
Attached	USB Power Delivery ports which are mechanically joined with USB cable.
BIST	Built In Self Test – Power Delivery testing mechanism for the Phy Layer.
Cold Socket	A downstream port receptacle that does not apply ***vSafe5V*** on V_{BUS} until a plug insertion is detected.
Connected	USB Power Delivery ports which are actively communicating using the USB Power Delivery protocol.
Consumer	The capability of a PD Port (typically a Device's upstream port) to sink power from the power conductor (e.g. V_{BUS}).
Consumer/Provider	A Consumer with the additional capability to act as a Provider.
Contract	An agreement reached between a Port Pair as a result of the negotiation process. The agreement may only be altered in the case of a re-negotiation, Hard Reset or failure of the Source.
Dead Battery	A device has a Dead Battery when the battery in a device is unable to power its functions.
Device	When lower cased (device), it refers to any USB product, either device or host.
Device Policy Manager	Module running in a Provider or Consumer that applies Local Policy to each port in the Device via the Policy Engine.
Downstream Port	Either a port in the Host or the ports defined in ***[USB2.0]***, ***[USB3.0]***.
Dual-Role Device	A product containing one or more Dual-Role Ports that are capable of operating as either a Source or a Sink.

Term	Description
Dual-Role Port	A Consumer/Provider or Provider/Consumer capable port that is a port capable of operating as either a Source or Sink.
HDD	A Hard Disk Drive.
Hard Reset	This is initiated by ***Hard Reset*** signaling from either Port Partner. It restores V_{BUS} to the default condition and resets the PD communications engine to its default state.
IR Drop	The voltage drop across the cable and connectors between the Source and the Sink. It is a function of the resistance of the ground wire in the cable plus the contact resistance in the connectors times the current flowing over the path.
Local Policy	Every Device has its own Policy, called the Local Policy, that is executed by its Policy Engine to control its power delivery behavior. The Local Policy at any given time may be the default policy, hard coded or modified by changes in operating parameters or one provided by the system Host or some combination of these. The Local Policy optionally may be changed by a System Policy Manager.
Negotiation	This is the PD process whereby: 1. The Source advertises its capabilities. 2. The Sink requests one of the advertised capabilities. 3. The Source acknowledges the request and alters its output to satisfy the request. The result of the negotiation is a contract for power delivery/consumption between the Port Pair.
PD	USB Power Delivery
PD Capable	A port that supports USB Power Delivery.
PDUSB	USB Device Port or USB Host Port that is PD capable.
PD Connection	A Port Pair with an established contract.
Phy Layer	The Physical Layer responsible for sending and receiving messages across V_{BUS} between a Port Pair.
Policy	Policy defines the behavior of PD capable parts of the system and defines the capabilities it advertises, requests made to (re)negotiate power and the responses made to requests received.
Policy Engine	The Policy Engine interprets the Device Policy Manager's input in order to implement Policy for a given port and directs the Protocol Layer to send appropriate messages.
Port	An interface typically exposed through an A, B or AB receptacle, or via a plug on the end of a hard-wired captive cable. USB Power Delivery defines the interaction between a Port Pair.
Port Pair	Two attached PD Ports.
Port Partner	The USB Power Delivery contract is negotiated between a Port Pair connected by a USB cable. These ports are known as Port Partners.
Power Conductor	The wire delivering power from the Source to Sink. For example USB's V_{BUS}.
Power Consumer	See Consumer
Power Provider	See Provider
Protocol Layer	The entity that forms the messages used to communicate information between Port Partners.

Term	Description
Provider	A capability of a PD Port (typically a Host, Hub, or Wall Wart downstream port) to source power over the power conductor (e.g. V_{BUS}).
Provider/Consumer	A Provider with the additional capability to act as a Consumer.
Re-negotiation	A process wherein one of the Port Partners wants to alter the negotiated contract.
Reserve	Power which is kept back by a Provider in order to ensure that it can meet total power requirements of attached Consumers on at least one port.
Safe Operation	Sources must have the ability to tolerate 5V applied by both Port Partners.
Sink	The port consuming power from V_{BUS}; most commonly a Device.
Single-Role Port	A port that is a port only capable of operating as a Source or Sink, but not both.
Soft Reset	A process that resets the PD communications engine to its default state.
Source	A role a port is currently taking to supply power over V_{BUS}; most commonly a Host or Hub downstream port.
System Policy	Overall system policy generated by the system, broken up into the policies required by each Port Pair to affect the system policy. It is programmatically fed to the individual Devices for consumption by their Policy Engines.
System Policy Manager	Module running on the USB Host. It applies the System Policy through communication with PD capable Consumers and Providers that are also connected to the Host via USB.
Tester	The Tester is assumed to be a piece of test equipment that manages the BIST testing process of a PD UUT.
Unit Under Test (UUT)	The PD device that is being tested by the Tester and responds to the initiation of a particular BIST test sequence.
Upstream Port	Typically a B port on a Device as defined in ***[USB2.0]***, ***[USB3.0]***.
USB Attached State	Synonomous with the ***[USB2.0]***] and ***[USB3.0]*** definition of the attached state
USB Powered State	Synonomous with the ***[USB2.0]*** and ***[USB3.0]*** definition of the powered state.
VI	Same as power (i.e. voltage * current = power)
Wall Wart	A power supply or “power brick” that is plugged into an AC outlet. It supplies DC power to power a device or charge a battery.

2. Overview

2.1 Introduction

In USB Power Delivery, pairs of directly attached ports negotiate voltage, current and/or direction of power flow over the power conductor (V_{BUS}). The mechanisms used, operate independently of other USB methods used to negotiate power. Any contract negotiated using this specification, supersedes any and all previous power contracts established whether from standard ***[USB2.0]***, ***[USB3.0]*** or ***[BC1.2]*** mechanisms.

Each Provider has a Local Policy, governing power allocation to its ports. Sinks also have their own Local Policy governing how they draw power. A System Policy can be enacted over USB that allows modification to these local policies and hence management of overall power allocation in the system.

When PD Devices are attached to each other, the downstream and upstream ports initially default to standard USB operation. The downstream port supplies 5V and the upstream port draws current in accordance with the rules defined by ***[USB2.0]***, ***[USB3.0]*** or ***[BC1.2]*** specification. After Power Delivery negotiation has taken place power can be supplied at higher, or lower, voltages and higher currents than defined in these specifications. It is also possible to swap the power supply roles such that the downstream port receives power and the upstream port supplies power.

2.2 USB Power Delivery Capable Devices

Some examples of USB Power Delivery capable Devices can be seen in Figure 2-1 (a Host, a Device, a Hub, and a Charger). These are given for reference only and do not limit the possible configurations of products that can be built using this specification.

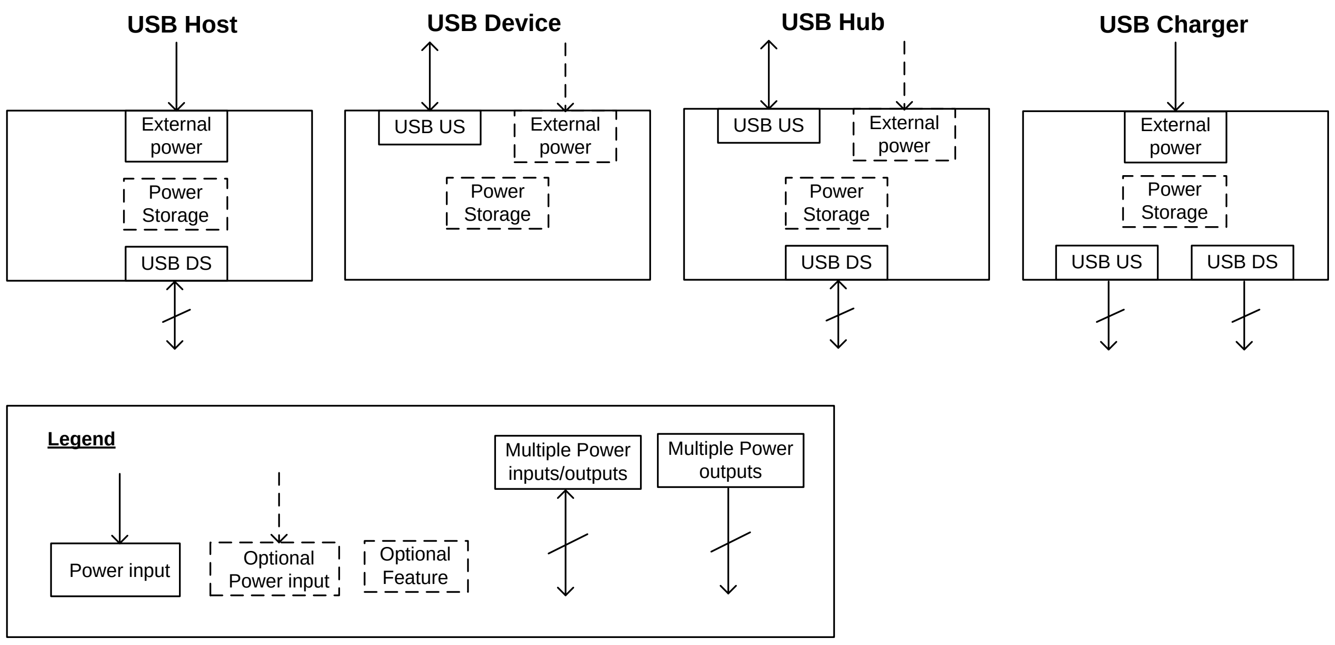


Figure 2-1 Logical Structure of USB Power Delivery Capable Devices

Each USB Power Delivery capable Device is assumed to be made up of at least one port. Providers are assumed to have a Source and Consumers a Sink. Each Device contains one, or more, of the following components:

- Upstream ports that:
    - Sink power (a Consumer).
    - Optionally source power (a Consumer/Provider).
    - Optionally communicate via USB.

- Downstream ports that:
    - Source power (a Provider).
    - Optionally Sink power (a Provider/Consumer).
    - Optionally communicate via USB.
- A Source that may be:
    - An external power source e.g. AC.
    - Power Storage (e.g. battery).
    - Derived from another port (e.g. bus-powered Hub).
- A power sink that may be:
    - Power Storage (e.g. a battery).
    - Used to power internal functions.
    - Used to power devices attached to other devices (e.g. a bus-powered Hub).

2.3 Operational Overview

A USB Power Delivery port supplying power is known as a Source and a port consuming power is known as a Sink. There is only one Source Port and one Sink Port in each PD connection between Port Partners. Where USB products support USB Power Delivery protocols a USB downstream port is initially a Source and a USB upstream port is initially a Sink, although USB Power Delivery also enables these roles to be swapped.

The following sections describe the high level operation of ports taking on the roles of Source and Sink.

2.3.1 Source Operation

The Source Port operates differently depending on attachment status:

- Following Attach
    - The Source Port detects attachment of an A plug.
    - The Source Port should set V_{BUS} to ***vSafe5V***.
- Before PD Connection
    - The Source Port's output voltage on V_{BUS} will be at the USB default of either ***vSafe0V*** or ***vSafe5V***.
    - The Source Port periodically advertises its capabilities by sending ***Capabilities*** messages.
- Establishing PD Connection
    - Presence of a Port Partner is detected either:
        - By receiving a ***GoodCRC*** message in response to a ***Capabilities*** message.
        - By receiving a message from the Sink e.g. ***Get_Source_Cap*** message.
        - By receiving ***Hard Reset*** signaling.
    - If the Source Port's output voltage on V_{BUS} is ***vSafe0V*** , it detects the presence of a Port Partner and sets the output voltage to ***vSafe5V***.
- During PD Connection (normal PD operation)
    - The Source Port detects the type of cabling attached and may alter its advertised capabilities depending on the type of cable detected.
    - The Source informs the Sink whenever its capabilities change, by sending a ***Capabilities*** message.
    - Whether a PD contract is negotiated or not, the Source Port processes and responds to all messages received.
    - The Source Port sends appropriate messages whenever its Local Policy requires.
    - The Source Port, after a period of ***tSourceActivity***, sends out ***Ping*** messages if no other messages are being sent.
- Detach or Communications Failure

    - There are two ways to detect Detach
        - Failure to receive a ***GoodCRC*** message in response to a ***Ping*** message or other message within ***tReceive*** leads to a Soft Reset, followed by a Hard Reset to begin restoring V_{BUS} to default operation within ~50ms. Receiving no response to further attempts at communication is interpreted by the Source Port as a detach event.
        - Detect plug removal and restore V_{BUS} to default operation within ~100ms.
    - The Source returns to unattached behavior including returning its power supply to the USB default of either ***vSafe0V*** or ***vSafe5V*** .
    - If the Source Port is a Consumer/Provider, the Source Port stops supplying power and returns to its default behavior as an unattached Sink Port.
- Error handling
    - Minor protocol errors are handled by a ***Soft Reset*** message issued by either Port Partner, that resets counters, timers and states, but does not change the negotiated voltage and current or the port's role (e.g. Source or Sink).
    - Serious errors are handled by ***Hard Reset*** signaling issued by either Port Partner, that resets protocol as for a Soft Reset but also returns the power supply to USB default ***vSafe0V*** or ***vSafe5V*** output in order to protect the Sink. A Consumer/Provider operating as a Source returns to its default operation as a Sink Port after the Hard Reset.

2.3.2 Sink Operation

The Sink Port operates differently depending on attachment status:

- V_{BUS} Unpowered
    - The Sink Port monitors V_{BUS} for the presence of ***vSafe5V***.
    - The Sink Port may send ***Get_Source_Cap*** messages.
- V_{BUS} Powered
    - The Sink detects the presence of ***vSafe5V*** on V_{BUS} and waits for a ***Capabilities*** message indicating the presence of a PD capable Source.
    - If the Sink Port does not receive a ***Capabilities*** message within ***tSinkActivity*** then it issues ***Hard Reset*** signaling in order to cause the Source port to send a ***Capabilities*** message if the Source port is PD capable.
- Establishing PD Connection
    - The Sink Port receives a ***Capabilities*** message.
    - The Sink Port detects the type of cabling attached.
    - The Sink Port may request one of the capabilities offered by the Source, even if this is the ***vSafe5V*** output offered by ***[USB2.0]***, ***[USB3.0]*** or ***[BC1.2]***, in order to enable future power negotiation.
        - A Sink not requesting any capability (e.g. responding with a ***Request*** message) when the present negotiated power level is still valid results in the retention of the present negotiated power level.
        - A Sink not requesting any capability if the presently negotiated power level is no longer valid, results in an error.
        - A Sink unable to fully operate at the offered capabilities requests the default capability but indicates that it would prefer another power level and provide a physical indication of the failure to the end user (e.g. using an LED).
- During PD Connection (normal PD operation)
    - The Sink Port processes and respond (if a response is required) to all messages received.
    - The Sink Port sends appropriate messages whenever its Local Policy requires.
    - A Sink whose power needs have changed indicates this to the Source with a new ***Request*** message. The Sink Port may request one of the capabilities previously offered by the Source, even if this is the

vSafe5V output offered by ***[USB2.0]***, ***[USB3.0]*** or ***[BC1.2]***, in order to enable future power negotiation.

    - A Sink not requesting any capability (e.g. responding with a ***Request*** message) when the present negotiated power level is still valid results in the retention of the present negotiated power level.
    - A Sink not requesting any capability if the presently negotiated power level is no longer valid, results in an error.
    - A Sink unable to fully operate at the offered capabilities requests the default capability but indicates that it would prefer another power level and provide a physical indication of the failure to the End User (e.g. using an LED).

- Removal of V_{BUS}
  - The Sink Port detects the removal of V_{BUS} and interpret this as the end of the PD Connection.
  - If the Sink Port is part of a Provider/Consumer the Sink Port stops sinking current and returns to its default behavior as an unattached Source Port.
- Error handling
  - Minor protocol errors are handled by a ***Soft Reset*** message issued by either Port Partner, that resets counters, timers and states, but does not change the negotiated voltage and current or the port's role (e.g. Source or Sink).
  - Serious errors are handled by issuing ***Hard Reset*** signaling that resets protocol as for a Soft Reset but also returns the power supply to its USB default current and voltage. A Provider/Consumer operating as a Sink returns to its USB default operation as a Source Port outputting ***vSafe0V*** or ***vSafe5V*** after the Hard Reset.

2.4 Architectural Overview

This logical architecture is not intended to be taken as an implementation architecture . An implementation architecture is, by definition, a part of product definition and is therefore outside of the scope of this specification.

This section outlines the high level logical architecture of USB Power Delivery referenced throughout this specification. In practice various implementation options are possible based on many different possible types of PD device. PD devices may have many different configurations e.g. USB or non-USB communication, single versus multiple ports, dedicated power supplies versus supplies shared on multiple ports, hardware versus software based implementations etc. The architecture outlined in this section is therefore provided only for reference in order to indicate the high level logic model used by the PD specification. This architecture is used to identify the key concepts and also to indicate logical blocks and possible links between them.

The USB Power Delivery architecture in each USB Power Delivery capable Device is made up of a number of major components.

The communications stack seen in Figure 2-2 consists of:

- A **Device Policy Manager** (see Section 8.2) that exists in all Devices and manages USB Power Delivery resources within the Device across one or more ports based on the Device's Local Policy.
- A **Policy Engine** (see Section 8.3) that exists in each USB Power Delivery port implements the Local Policy for that port.
- A **Protocol Layer** (see Chapter 6) that enables messages to be exchanged between a Source Port and a Sink Port.
- A **Physical Layer** (see Chapter 5) that handles transmission and reception of bits on the wire and handles data transmission.

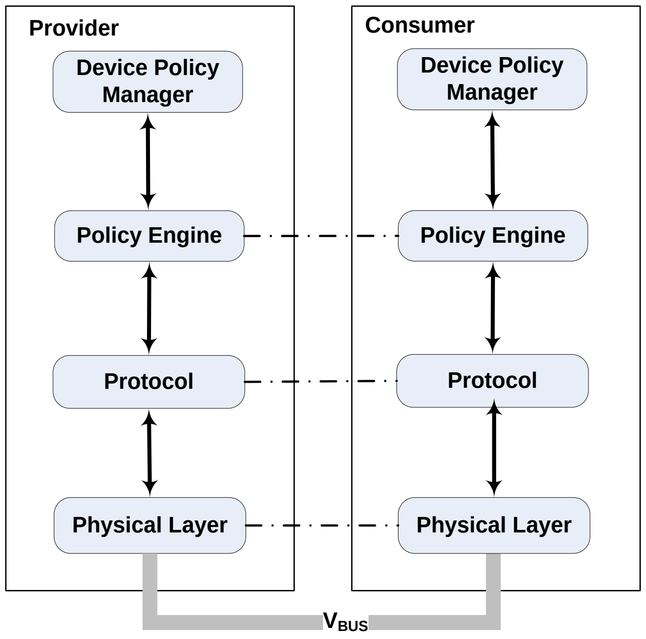


Figure 2-2 USB Power Delivery Communications Stack

Additionally USB Power Delivery devices which can operate as USB devices may communicate over USB (see Figure 2-3). An optional **System Policy Manager** (see Chapter 9) that resides in the USB Host communicates with the PD Device over USB, via the root port and potentially over a tree of USB Hubs. The **Device Policy Manager** interacts with the USB interface in each Device in order to provide and update PD related information in the USB domain. Note that a PD device is not required to have a USB device interface.

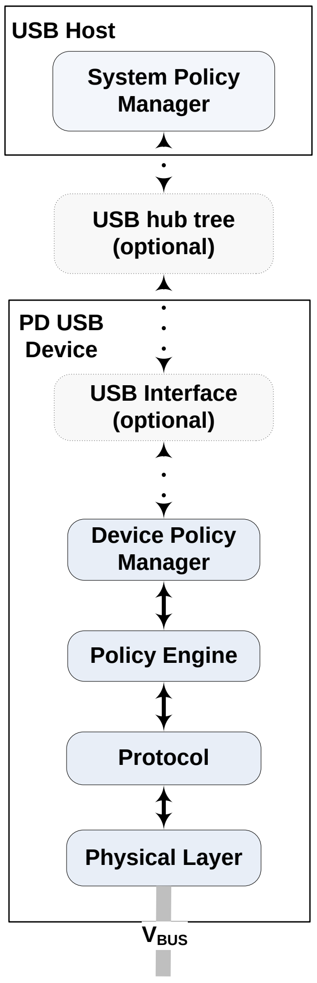


Figure 2-3 USB Power Delivery Communication Over USB

Figure 2-4 shows the logical blocks between two attached PD ports. In addition to the communication stack described above there are also:

- For a Provider or Dual-Role Device: one or more **Sources** providing power to one or more ports.
- For a Consumer or Dual-Role Device: a **Sink** consuming power.
- A **Cable Detection** module (see Section4.4) that:
    - Detects presence of V_{BUS} for Sink Ports
    - Identifies the type of PD cable attached
- The **PD Cabling** (see Section 3) that carries both power and USB Power Delivery signaling over V_{BUS}.
- The **Device Policy Manager** talks to the communication stack, Source/Sink and the cable detection block in order to manage the resources in the Provider or Consumer.

Figure 2-4 illustrates a Provider and a Consumer. Dual-Role Devices such as Provider/Consumers or Consumer/Providers can be constructed by combining the elements of both Provider and Consumer into a single Device. Providers can also contain multiple Source Ports each with their own communications stack and cable detection.

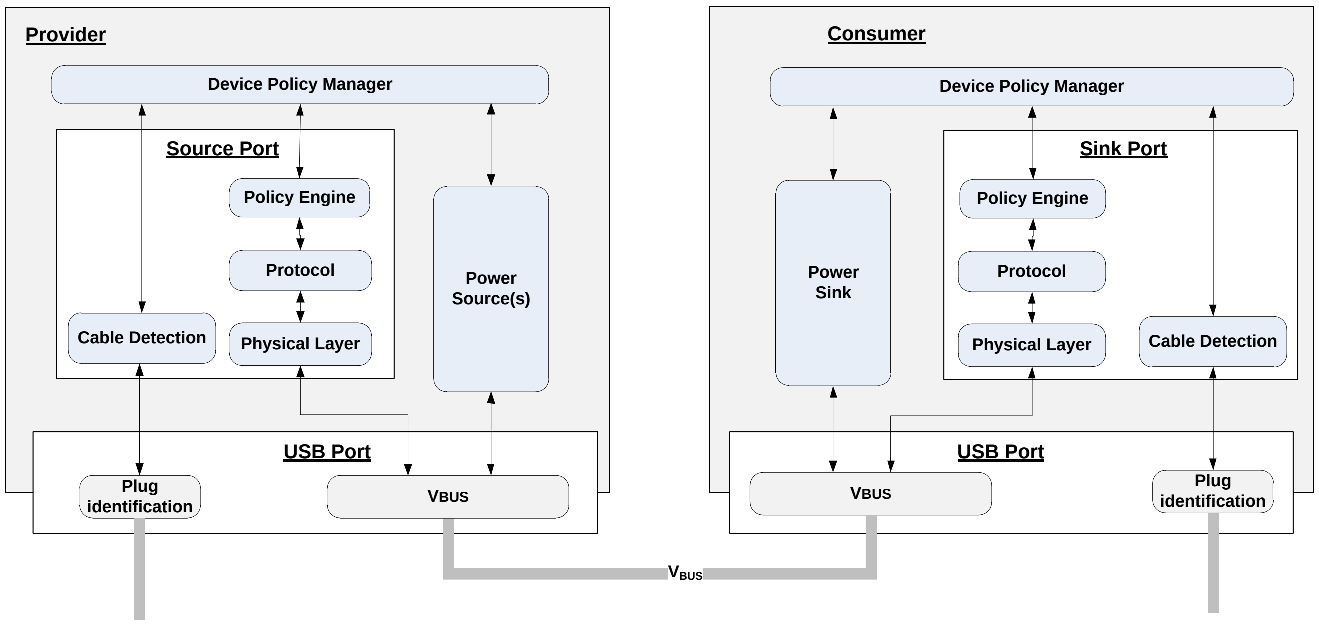


Figure 2-4 High Level Architecture View

2.4.1 Policy

There are two possible levels of Policy:

1) System Policy applied system wide by the System Policy Manager across multiple Providers or Consumers.
2) Local Policy enforced on a Provider or Consumer by the Device Policy Manager.

Policy comprises several logical blocks:

- System Policy Manager (system wide).
- Device Policy Manager (one per Provider or Consumer).
- Policy Engine (one per Source or Sink Port).

2.4.1.1 System Policy Manager

Since the USB Power Delivery protocol is essentially a point to point, implementation of a System Policy requires communication by an additional data communication mechanism i.e. USB. The System Policy Manager monitors and controls System Policy between various Providers and Consumers connected via USB. The System Policy Manager resides in the USB Host and communicates via USB with the Device Policy Manager in each connected Device. Devices without USB data communication capability or are not data connected, will not be able to participate in System Policy.

The System Policy Manager is optional in any given system so USB Power Delivery Providers and Consumers can operate without it being present. This includes systems where the USB Host does not provide a System Policy Manager and may also include "headless" systems without any USB Host. In those cases where a Host is not present, USB Power Delivery is useful for charging purposes, or the powering of devices since useful USB functionality is not possible. Where there is a USB Host but no System Policy Manager, Providers and Consumers can negotiate power between themselves, independently of USB power rules, but are more limited in terms of the options available for managing power.

2.4.1.2 Device Policy Manager

The Device Policy Manager provides mechanisms to monitor and control the USB Power Delivery system within a particular Consumer or Provider. The Device Policy Manager enables Local Policies to be enforced across the system by communication with the System Policy Manager. Local Policies are enacted on a per port basis by the Device Policy Manager's control of the Source/Sink Ports) and by communication with the Policy Engine and Cable Detection for that port.

2.4.1.3 Policy Engine

Providers and Consumers are free to implement their own Local Policies on their directly connected Source or Sink Ports. These will be supported by negotiation and status mechanisms implemented by the Policy Engine for that port. The Policy Engine interacts directly with the Device Policy Manager in order to determine the present Local Policy to be enforced. The Policy Engine will also be informed by the Device Policy Manager whenever there is a change in Local Policy (e.g. a capabilities change).

2.4.2 Message Formation and Transmission

2.4.2.1 Protocol Layer

The Protocol Layer forms the messages used to communicate information between a pair of ports. It is responsible for forming ***Capabilities*** messages, requests and acknowledgements. Additionally it forms messages used to Swap roles and maintain presence. It receives inputs from the Policy Engine indicating which messages to send and indicates the responses back to the Policy Engine.

The basic protocol uses a push model where the Provider pushes it capabilities to the Consumer that in turn responds with a request based on the offering. However, the Consumer may asynchronously request the Provider's present capabilities and may select another voltage/current.

2.4.2.2 Physical Layer

The Physical Layer is responsible for sending and receiving messages across V_{BUS}. It consists of a transceiver that superimposes a signal on the V_{BUS} wire. The Physical Layer is responsible for managing data on the V_{BUS} wire. It tries to avoid collisions on the V_{BUS} wire, recovering from them when they occur. It also detects errors in the messages using a CRC.

2.4.3 Power supply

2.4.3.1 Source

Each Provider will contain one or more Sources that are shared between one or more ports. These Sources are controlled by the Local Policy. Sources start up in USB default operation where the port applies ***vSafe0V*** or ***vSafe5V*** on V_{BUS} and return to this state on detach or after a Hard Reset. If the Source applies ***vSafe0V*** as their default, it monitors V_{BUS} to detect attach events and transitions its output to ***vSafe5V*** upon detecting an attach event.

2.4.3.2 Sink

Consumers are assumed to have one Sink connected to a port. This Sink is controlled by Local Policy. Sinks start up in USB default operation where the port can operate at ***vSafe5V*** with USB default specified current levels and return to this state on detach or after a Hard Reset.

2.4.3.3 Dual-Role Ports

Dual-Role ports are found in Provider/Consumers or Consumer/Providers. A Provider/Consumer will have one or more ports that can operate as either a Source Port (default) or a Sink Port. A Consumer/Provider will have a port that can operate either as a Sink Port (default) or a Source Port.

Combination supplies that support Dual-Role ports follow the default rules as well as supporting a return to their own default role and state on a detach event or Hard Reset.

2.4.3.4 Dead Battery or Lost Power Detection

The USB Power Delivery specification defines a mechanism intended to communicate with and charge a Provider/Consumer with a dead battery. A Provider/Consumer may also use this mechanism to get power when they are disconnected from their normal power source. All Consumer/Providers support this mechanism.

2.4.4 Cable and Connectors

2.4.4.1 Cable Detection

The Cable Detection block provides mechanisms to detect the type of cable attached. This information is provided to the Device Policy Manager. It adjusts the Local Policy accordingly and may also communicate cabling issues to the System Policy Manager for further action.

The USB Power Delivery specification assumes either certified USB cables or detectable PD cables. Both PD signaling and power are delivered via V_{BUS}. The existence of a large number of non-compliant legacy cables, particularly 'Y' and 'W' cables are problematic. These kinds of cables, in combination with the higher voltages that PD can deliver, have the potential to permanently damage the user's equipment. PD defines mechanisms to detect PD capable cables.

2.4.4.2 Ground IR Drop Measurement

The use of non-compliant cables, adapters or combinations may result in excessive IR drop on the Ground which may result in unreliable ***[USB2.0]*** data communications. Every Consumer is recommended to ensure that the IR drop across Ground between itself and the Source is minimized in order to maintain reliable communications. When the Consumer determines the IR drop exceeds allowable limits (see Section 4.3), it reduces its load to stay within those limits and send a ***CableVGO*** message (V_{BUS} Ground Offset) message to the Provider informing it of the over-current event.

2.4.4.3 Interactions between Legacy, BC and PD Devices

In all cases, where either the Source Port or the Sink Port is part of a legacy Device, V_{BUS} operates as defined in the ***[USB2.0]***, ***[USB3.0]*** and ***[BC1.2]*** specifications respectively. There are two primary cases to consider:

- The Host (downstream port) is legacy and as such will not send any advertisements. An attached PD capable Device will not see any advertisements and operates using the rules defined in the ***[USB2.0]***, ***[USB3.0]*** or ***[BC1.2]*** specifications.
- The Device (upstream port) is legacy and as such will not see any advertisements and therefore will not respond. The Host (downstream port) will continue to supply ***vSafe5V*** to V_{BUS} as specified in the ***[USB2.0]***, ***[USB3.0]*** or ***[BC1.2]*** specifications.

USB Power Delivery only operates when two USB Power Delivery Devices are directly connected. When a Device finds itself a mixed environment, where the other Device does not support the USB Power Delivery Specification, the existing contract to supply ***vSafe5V*** albeit at low power (which may be negotiated to high power as defined in the ***[USB2.0]***, ***[USB3.0]*** or ***[BC1.2]*** specifications) remains operative.

2.4.4.4 Power Profiles

Power Profiles are used to define voltages and current ranges that may be offered by USB Power Delivery Sources. Although Profiles are defined for Sources, Sinks may refer to a Profile that will meet their power requirements. See Appendix A for further details.

3. Cable Assemblies and Connectors

USB Power Delivery is designed to operate at voltages and currents outside the specificied ranges defined in ***[USB2.0]***, ***[USB3.0]*** and ***[BC1.2]*** specifications. For this reason it is critical to understand the characteristics of the USB Power Delivery cable assemblies and the connectors used. The following sections define the assumed characteristics of both legacy cable assemblies (unmarked cable assemblies) and USB Power Delivery cable assemblies designed to take advantage of the USB Power Delivery system (marked cable assemblies)

The considerable presence of non-compliant legacy cable assemblies presents an additional challenge to safely deliver voltages other than ***vSafe5V***. 'Y' cable assemblies, for example, provide topologies and connections that are not included in the original USB specifications. As a result, there is a requirement to identify cable assemblies used in support of this specification that deliver more than ***vSafe5V***.

The additional requirements for USB Power Delivery cable assemblies are defined in this section. Unlike typical USB philosophy, both the Device and the Host are responsible for detecting the insertion of a USB Power Delivery cable assembly and make requests for power accordingly.

3.1 Significant Features

This section identifies the significant features of the USB Power Delivery connectors and cable assemblies This section references other parts of the document where further details can be found.

3.1.1 Connectors

The USB PD specification defines the following connectors:

- PD Standard-A plug and receptacle
- PD Standard-B plug and receptacle
- PD Micro-A plug
- PD Micro-AB receptacle
- PD Micro-B plug and receptacle

The USB PD connectors have been significantly improved to increase their current carrying capability (5A for the standard connector and 3A for the micro connector), and to allow them to be electrically identifiable while remaining mechanically compatible with the existing connector set.

For details on the usage of the PD icon refer to Section 3.5.

The PD Micro-A plug, PD Micro-AB receptacle, PD Micro-B plug, and PD Micro-B receptacle are mechanically compatible with their ***[USB2.0]*** or ***[USB3.0]*** versions except for the electrically detected marking. The PD Standard-A plug, PD Standard-A receptacle, PD Standard-B plug, and PD Standard-B receptacle are mechanically different than their ***[USB2.0]*** or ***[USB3.0]*** versions to provide for electrically detected marking. Refer to Section 3.4 for marking details.

Appendix D illustrates mating conditions of Standard-A connector combinations.

Table 3-1 lists the compatible plugs and receptacles and possible roles which may be supported in each case. Note that for AB receptacles it is not required that if the Provider/Consumer role is supported with an A-plug inserted, that the Consumer/Provider role is supported with a B-plug inserted or vice-versa. Standard, non-PD, receptacles used in USB Power Delivery capable products are limited to a nominal 5V at 1.5A as defined in ***[BC1.2]***.

Table 3-1 Plugs Accepted By Receptacles

Receptacle	Plugs Accepted	Possible Role
USB 2.0 Standard-A	USB 2.0 Standard-A, USB 2.0 PD Standard-A, USB 3.0 Standard-A, or USB 3.0 PD Standard-A	Provider, Provider/Consumer
USB 2.0 PD Standard-A	USB 2.0 PD Standard-A, USB 2.0 Standard-A, USB 3.0 PD Standard-A, or USB 3.0 Standard-A	Provider, Provider/Consumer

Receptacle	Plugs Accepted	Possible Role
USB 3.0 Standard-A	USB 3.0 Standard-A, USB 3.0 PD Standard-A, USB 2.0 Standard-A, or USB 2.0 PD Standard-A	Provider, Provider/Consumer
USB 3.0 PD Standard-A	USB 3.0 PD Standard-A, USB 3.0 Standard-A, USB 2.0 PD Standard-A, or USB 2.0 Standard-A	Provider, Provider/Consumer
USB 2.0 Standard-B	USB 2.0 Standard-B or USB 2.0 PD Standard-B	Consumer, Consumer/Provider
USB 2.0 PD Standard-B	USB 2.0 PD Standard-B or USB 2.0 Standard-B	Consumer, Consumer/Provider
USB 3.0 Standard-B	USB 3.0 Standard-B, USB 3.0 PD Standard-B, USB 2.0 Standard-B, USB 2.0 PD Standard-B or USB 3.0 Powered-B	Consumer, Consumer/Provider
USB 3.0 PD Standard-B	USB 3.0 PD Standard-B, USB 3.0 Standard-B, USB 2.0 PD Standard-B, USB 2.0 Standard-B, or USB 3.0 Powered-B	Consumer, Consumer/Provider
USB 3.0 Powered-B	USB 3.0 Powered-B, USB 3.0 Standard-B, USB 3.0 PD Standard-B, USB 2.0 Standard-B, or USB 2.0 PD Standard-B	Consumer, Consumer/Provider
USB 2.0 Micro-B	USB 2.0 Micro-B or USB 2.0 PD Micro-B	Consumer, Consumer/Provider
USB 2.0 PD Micro-B	USB 2.0 PD Micro-B or USB 2.0 Micro-B	Consumer, Consumer/Provider
USB 3.0 Micro-B	USB 3.0 Micro-B, USB 3.0 PD Micro-B, USB 2.0 Micro-B, or USB 2.0 PD Micro-B	Consumer, Consumer/Provider
USB 3.0 PD Micro-B	USB 3.0 PD Micro-B, USB 3.0 Micro-B, USB 2.0 PD Micro-B, or USB 2.0 Micro-B	Consumer, Consumer/Provider
USB 2.0 Micro-AB	USB 2.0 Micro-A, or USB 2.0 PD Micro-A	Provider, Provider/Consumer,
	USB 2.0 Micro-B, USB 2.0 PD Micro-B	Consumer, Consumer/Provider
USB 2.0 PD Micro-AB	USB 2.0 PD Micro-A, or USB 2.0 Micro-A	Provider, Provider/Consumer
	USB 2.0 PD Micro-B, USB 2.0 Micro-B,	Consumer, Consumer/Provider
USB 3.0 Micro-AB	USB 2.0 Micro-A, or USB 2.0 PD Micro-A, USB 3.0 Micro-A, USB 3.0 PD Micro-A	Provider, Provider/Consumer
	USB 2.0 Micro-B, USB 2.0 PD Micro-B, USB 3.0 Micro-B, USB 3.0 PD Micro-B	Consumer, Consumer/Provider
USB 3.0 PD Micro-AB	USB 2.0 PD Micro-A, or USB 2.0 Micro-A USB 3.0 PD Micro-A, USB 3.0 Micro-A	Provider, Provider/Consumer
	USB 2.0 PD Micro-B, USB 2.0 Micro-B, USB 3.0 PD Micro-B, USB 3.0 Micro-B	Consumer, Consumer/Provider

3.1.1.1 USB 2.0 PD Standard-A Connector

The USB 2.0 PD Standard-A connector is defined as the host connector. It has the same mating interface as the USB 2.0 Standard-A connector, but with mechanical differences to provide a means of detecting insertion and PD support. Refer to Section 3.2.3 for mechanical details, pin assignments, and descriptions.

A USB 2.0 PD Standard-A receptacle accepts a USB 2.0 PD Standard-A plug, a USB 2.0 Standard-A plug, a USB 3.0 PD Standard-A plug, or a USB 3.0 Standard-A plug. Similarly, a USB 2.0 PD Standard-A plug may be mated with a USB 2.0 PD Standard-A receptacle, a USB 2.0 Standard-A receptacle, a USB 3.0 PD Standard-A receptacle, or a USB 3.0 Standard-A receptacle.

3.1.1.2 USB 3.0 PD Standard-A Connector

The USB 3.0 PD Standard-A connector is defined as a host connector. It has the same mating interface as the USB 3.0 Standard-A connector, but with mechanical differences to provide a means of detecting insertion and PD support. Refer to Section 3.2.4 for mechanical details, pin assignments, and descriptions.

A USB 3.0 PD Standard-A receptacle accepts a USB 3.0 PD Standard-A plug, a USB 3.0 Standard-A plug, a USB 2.0 PD Standard-A plug, or a USB 2.0 Standard-A plug. Similarly, a USB 3.0 PD Standard-A plug may be mated with a USB 3.0 PD Standard-A receptacle, a USB 3.0 Standard-A receptacle, a USB 2.0 PD Standard-A receptacle, or a USB 2.0 Standard-A receptacle.

3.1.1.3 USB 2.0 PD Standard-B Connector

The USB 2.0 PD Standard-B connector is defined to facilitate deliver of up to 100 Watts. An ID pin supports PD identification. Refer to Section 3.2.5 for mechanical details, pin assignments, and descriptions.

A USB 2.0 PD Standard-B receptacle accepts either a USB 2.0 PD Standard-B plug or a USB 2.0 Standard-B plug. Inserting a USB 3.0 PD Standard-B plug, a USB 3.0 Standard-B plug, or a USB 3.0 Powered-B plug into a USB 2.0 PD Standard-B receptacle is physically disallowed. A USB 2.0 PD Standard-B plug may be mated with a USB 2.0 PD Standard-B receptacle, a USB 2.0 Standard-B receptacle, a USB 3.0 PD Standard-B receptacle, a USB 2.0 Standard-B receptacle, or a USB 3.0 Powered-B receptacle.

3.1.1.4 USB 3.0 PD Standard-B Connector

The USB 3.0 PD Standard-B connector is defined to facilitate delivery of up to 100 Watts. An ID pin supports PD identification. Refer to Section 3.2.6 for mechanical details, pin assignments, and descriptions.

A USB 3.0 PD Standard-B receptacle accepts a USB 3.0 PD Standard-B plug, a USB 3.0 Standard-B plug, a USB 3.0 Powered-B plug, a USB 2.0 PD Standard-B plug, or a USB 2.0 Standard-B plug. Inserting a USB 3.0 PD Standard-B plug or a USB 3.0 Standard-B plug into a USB 2.0 PD Standard-B receptacle or a USB 2.0 Standard-B receptacle is physically disallowed. A USB 3.0 PD Standard-B plug may be mated with a USB 3.0 PD Standard-B receptacle, a USB 3.0 Standard-B receptacle, or a USB 3.0 Powered-B receptacle.

3.1.2 Compliant Cable Assemblies

The USB Power Delivery Specification defines the following cable assemblies:

- USB 3.0 PD Standard-A plug to USB 3.0 PD Standard-B plug
- USB 3.0 PD Standard-A plug to USB 3.0 PD Micro-B plug
- USB 3.0 PD Micro-A plug to USB 3.0 PD Micro-B plug
- USB 3.0 PD Micro-A plug to USB 3.0 PD Standard-B plug
- USB 2.0 PD Standard-A plug to USB 2.0 PD Standard-B plug
- USB 2.0 PD Standard-A plug to USB 2.0 PD Micro-B plug
- USB 2.0 PD Micro-A plug to USB 2.0 PD Micro-B plug
- USB 2.0 PD Micro-A plug to USB 2.0 PD Standard-B plug

PD cable assemblies shall have one plug on each end. PD Cable assemblies with multiple connectors on either end are expressly prohibited due to the danger of back-powering USB ports with high voltages. Any cable combinations not explicitly defined in the list above shall not be permitted.

Connectors on each end of the PD cable assembly shall be labeled with the appropriate USB PD icon defined in Section 3.5.

3.1.3 USB Power Delivery Adapters

Table 3-2 defines USB Power Delivery adapters (PD adapters) compliant with this specification (indicated with a "√"). Charging adapters are not included in this section. The Micro-A plug, Micro-B plug, Micro-AB receptacle, and Micro-B receptacle used for a PD adapter shall be constrained to either the ***[USB2.0]*** or ***[USB3.0]*** version with the exception of electronic marking described in this specification. The PD Standard-A plug and PD Standard-B plug used for a PD adapter shall be constrained to either the ***[USB2.0]*** or ***[USB3.0]*** version defined by this specification. Adapters shall have one plug and one receptacle. Adapters consist of a plug directly connected to a receptacle or connected by a cable. Adapters with multiple plugs or multiple receptacles are expressly prohibited due to the danger of exposing USB ports to higher voltages. The receptacle in the adapter is not required to perform insertion detect. PD adapters shall be labeled with the appropriate USB PD icon defined in Section 3.5.

PD adapters shall connect the Shield, the ID pin and GND through the plug to the receptacle and shall not connect the GND and the Shield together. Detection of IR drop (see Section 4.3) shall be used to determine if PD cable adapters and standard USB cabling have been combined. Note that the use of adapters or adapter cables potentially doubles the IR drop due to the extra connector interfaces and may cause unacceptable ***VGND***_DROP or V_{VBUS_DROP} conditions. Adapter connector combinations not explicitly defined in Table 3-2 shall not be permitted. See Figure 3-1, Figure 3-2, Figure 3-3, Figure 3-4, Figure 3-5, Figure 3-6, and Figure 3-7 for connection diagrams.

Adapters with PD Standard-A connectors require special attention to design since the Standard-A connector does not support the ID function. Implementation, such as circuitry to detect or configure ID pin electrical marking, is vendor specific and represented by simple boxes in the connection diagrams. Refer to Figure 3-1, Figure 3-2, and Figure 3-3. The adapter with the PD Micro-AB receptacle shall detect the ID pin electrical marking and shall only support a Micro-A plug.

3.1.3.1 PD Standard-A Plug to PD Standard-A Receptacle

A PD Standard-A plug to PD Standard-A Receptacle adapter shall be implemented as shown in Figure 3-1. PD communication on VBUS shall be inhibited (e.g., filtered or shunted) when no connector is present or when a Standard-A plug is present in the PD Standard-A receptacle. PD communication on VBUS shall be enabled when a PD Standard-A plug is present in the PD Standard-A receptacle.

3.1.3.2 PD Micro-A Plug to PD Standard-A Receptacle

A PD Micro-A plug to PD Standard-A Receptacle adapter shall be implemented as shown in Figure 3-2. The PD Micro-A plug ID shall indicate, by the resistance value between the ID pin and ground, whether or not a PD Standard-A plug is present in the PD Standard-A receptacle.

3.1.3.3 PD Standard-A Plug to PD Micro-AB Receptacle:

A PD Standard-A plug to PD Micro-AB Receptacle adapter shall be implemented as shown in Figure 3-3. V_{BUS} shall be disconnected between the PD Standard-A plug and the PD Micro-AB receptacle when no connector is present or when a Micro-B plug is present in the PD Micro-AB receptacle. V_{BUS} shall be connected between the PD Standard-A plug and the PD Micro-AB receptacle and PD communication on V_{BUS} shall be inhibited (e.g., filtered or shunted) when a Micro-A plug is present in the PD Micro-AB receptacle. V_{BUS} shall be connected between the PD Standard-A plug and the PD Micro-AB receptacle and PD communication on V_{BUS} shall be enabled when a PD Micro-A plug is present in the PD Micro-AB receptacle.

Table 3-2 Compliant PD Adapter Connector Combinations

Receptacle ► Plug Type ▼	PD Micro-AB	PD Micro-B	PD Standard-A	PD Standard-B
PD Micro-A			√	
PD Micro-B		√		√
PD Standard-A	√		√	
PD Standard-B		√		√

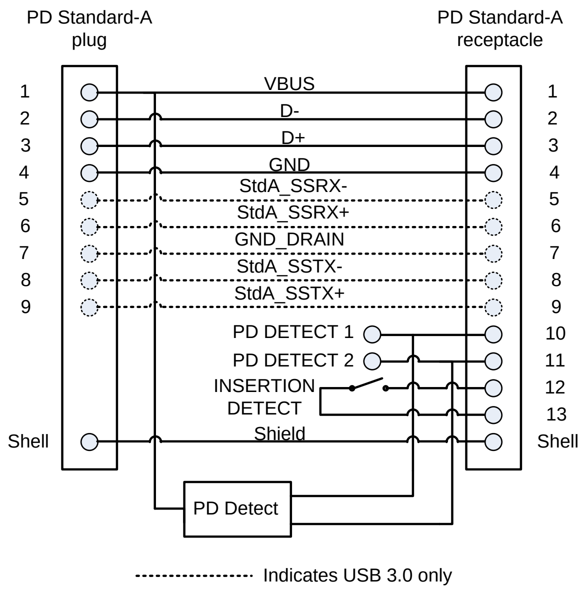


Figure 3-1 PD Standard-A Plug to PD Standard-A Receptacle

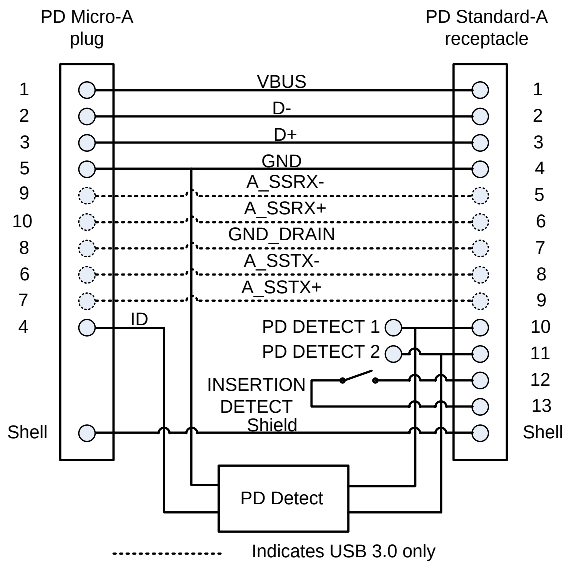


Figure 3-2 PD Micro-A Plug to PD Standard-A Receptacle

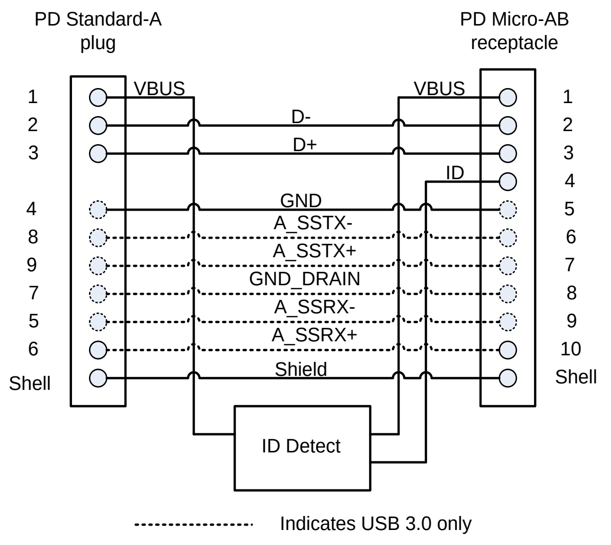


Figure 3-3 PD Standard-A Plug to PD Micro AB Receptacle

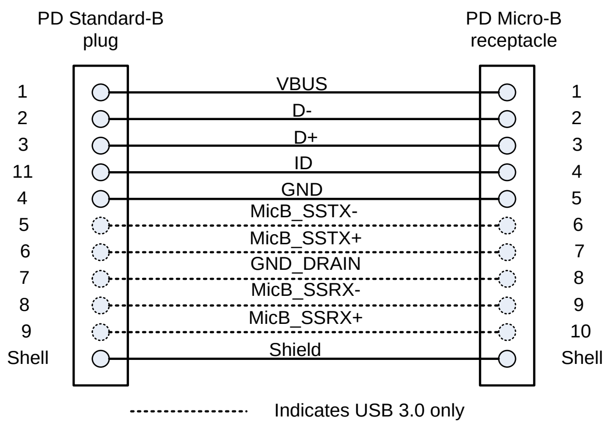


Figure 3-4 PD Standard-B Plug to PD Micro-B Receptacle

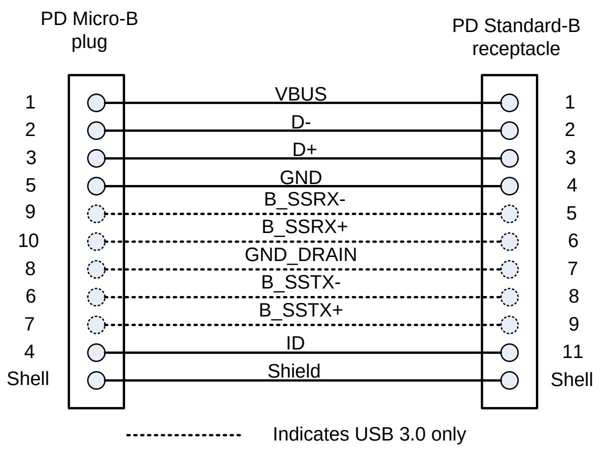


Figure 3-5 PD Micro-B Plug to PD Standard-B Receptacle

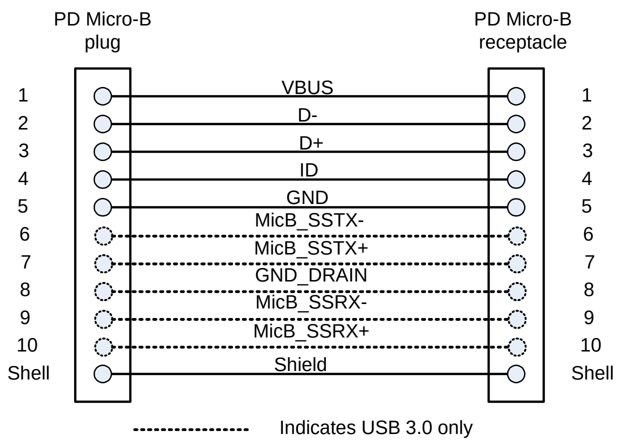


Figure 3-6 PD Micro-B Plug to PD Micro-B Receptacle

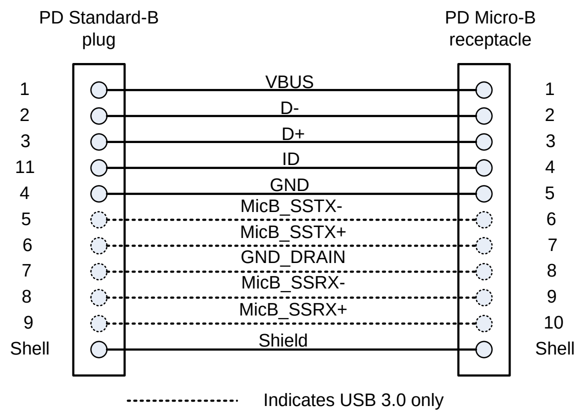


Figure 3-7 PD Standard-B Plug to PD Micro-B Receptacle

3.1.4 Hardwired Captive PD Cable

A hardwired captive PD cable assembly has a single PD plug at one end and a single hardwired (non-removable) connection at the other end. A hardwired captive PD cable assembly is PD compliant if it meets the requirements of this specification at the connector end cable marking or marking detection, negotiation capability, etc. and supports USB PD. A hardwired captive PD cable assembly may support USB signaling. The initial role of a port with a hardwired captive cable assembly may be either a Provider or a Consumer as determined by the type of plug.

The connector on the hardwired captive PD cable assembly shall be labeled with the appropriate USB PD icon defined in Section 3.5.

3.1.5 Standard-A Insertion Detect

Insertion Detect is a feature added to the Standard-A receptacle to support cold socket capability. It may be implemented in a Standard-A receptacle or a PD Standard-A receptacle. Insertion Detect provides an open circuit condition on pin 13 when a plug is not detected in the Standard-A receptacle and a connection to the receptacle shield on pin 13 when a plug is present. Schematic representation is provided in Figure 3-8 to illustrate the electrical implementation of the detection mechanism if pin 12 is present. Implementation is vendor-specific. The Insertion Detect feature shall be implemented for cold socket Standard-A applications and optional for all other Standard-A implementations. See Section 3.2.1 for the mechanical requirements, and Sections 3.6.1, 3.6.2 and 4.3 for the electrical description.

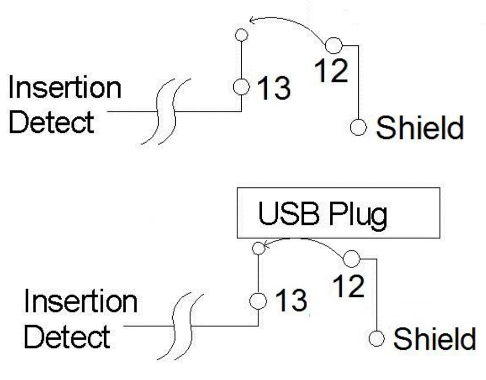


Figure 3-8 Standard-A Insertion Detect Schematic Representation

3.1.6 Standard-A PD Detect

The PD Detect contact is a feature added to the PD Standard-A receptacle to detect the presence of a PD Standard-A cable plug. Implementation of PD Detect shall include either pin 10, pin 11, or both pin 10 and pin 11 in the PD

Standard-A receptacle. If both pins 10 and 11 are present then both shall be connected to the PD Detect logic. Depending on the detection mechanism and location, two pins may be required to ensure PD Detect if the plug is skewed within the receptacle.

PD Detect shall be an open circuit when PD is not detected (refer to Figure 3-9, Figure 3-10 and Figure 3-11) and shall be connected to the receptacle shield when a PD Standard-A plug is present (Figure 3-12). PD Detect shall be a closed circuit to the shield when the PD plug is detected.

The PD Standard-A receptacle shall not connect PD Detect to the receptacle shield when a USB Thin Card is inserted, as shown in Figure 3-11.

Schematic representation is provided (Figure 3-12) to illustrate the electrical implementation of the detection mechanism when the implementation uses the PD Standard-A plug shield as a conductor to complete the detection circuit. The mechanics of the implementation are vendor-specific.

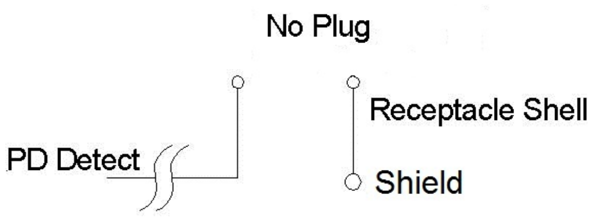


Figure 3-9 PD Standard-A No Plug Detection Circuit

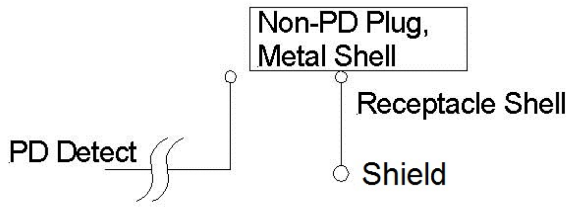


Figure 3-10 Non-PD Plug Standard-A Detection Circuit

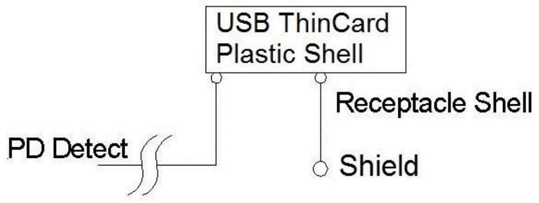


Figure 3-11 USB Thin Card Standard-A Detection Circuit

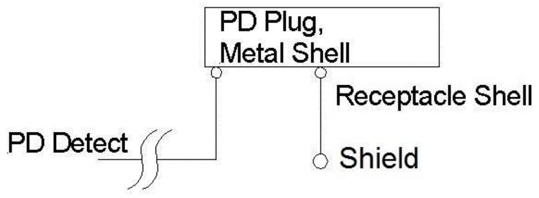


Figure 3-12 PD Plug Standard-A Detection Circuit

See Appendix D for mechanical details of different connector mating situations. See Section 3.6.1 and 3.6.2 for the electrical performance requirements.

3.1.7 Raw Cables

Attention should be given for the possibility of signal interference from V_{BUS}, especially at higher operating current and voltage, in PD cable assemblies that are capable of Hi-Speed or SuperSpeed data communication. See Section 3.6.6 crosstalk requirements.

3.2 Connector Mating Interfaces

This section defines the connector mating interfaces, including the connector interface drawings, pin assignments and descriptions. All dimensions are in mm.

3.2.1 Standard-A Insertion Detect Mechanical Dimensions

The Insertion Detect feature is optional if cold socket is not implemented. Figure 3-13 shows the mechanical dimensions for the zone where the Insertion Detect mechanism on the Standard-A and PD Standard-A receptacle shall activate when any type of Standard-A plug is inserted This zone applies to any USB 2.0 or USB 3.0 versions of receptacles. The detection mechanism shall be designed such that the insertion detect electrical requirements are met, regardless of power being applied. Special consideration should be given to the location of shell features such as retention springs and cutout areas.

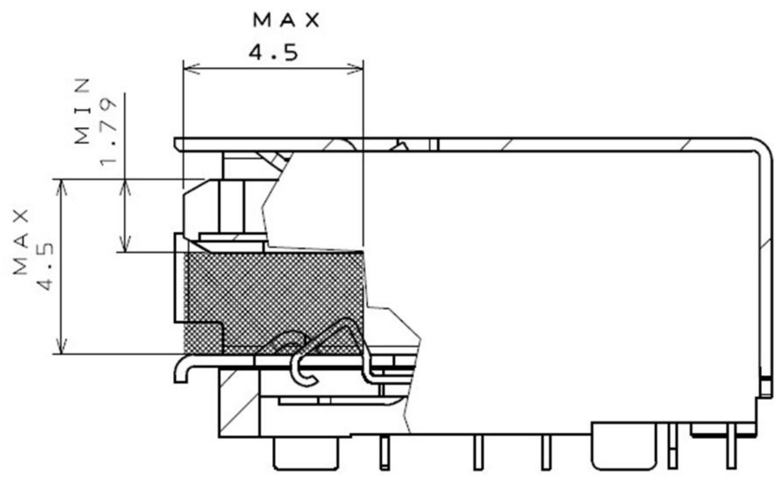


Figure 3-13 Insertion Detect Zone Mechanical Dimensions for the Standard-A Receptacle

3.2.2 USB PD Standard-A PD Detect Mechanical Requirement

Figure 3-14 shows the mechanical dimensions for the range where the PD Detect mechanism on the PD Standard-A receptacle shall indicate PD Detect when a PD Standard-A plug is inserted. The dimensions shall apply to planes parallel to Datum A and shall apply to both USB 2.0 and USB 3.0 versions of PD Standard-A receptacles. Implementation is vendor specific. The sample implementation in the Figure 3-14 is for reference only. PD detect electrical characteristics are defined in Section 3.1.6. The implementation shall also conform to the following requirements:

- The metal shell of the USB PD Standard-A plug is nominally 1.3 mm longer than the USB Standard-A plug to provide a means for PD Detect.
- The metal shell of the USB PD Standard-A plug shall be gold plated on the inner and outer surfaces a minimum of 1.6 mm from the leading edge.
- Contact for PD Detect shall occur on either the inner or the outer surface of the plug shell.
- The insertion of a USB ThinCard shall not indicate PD Detect.

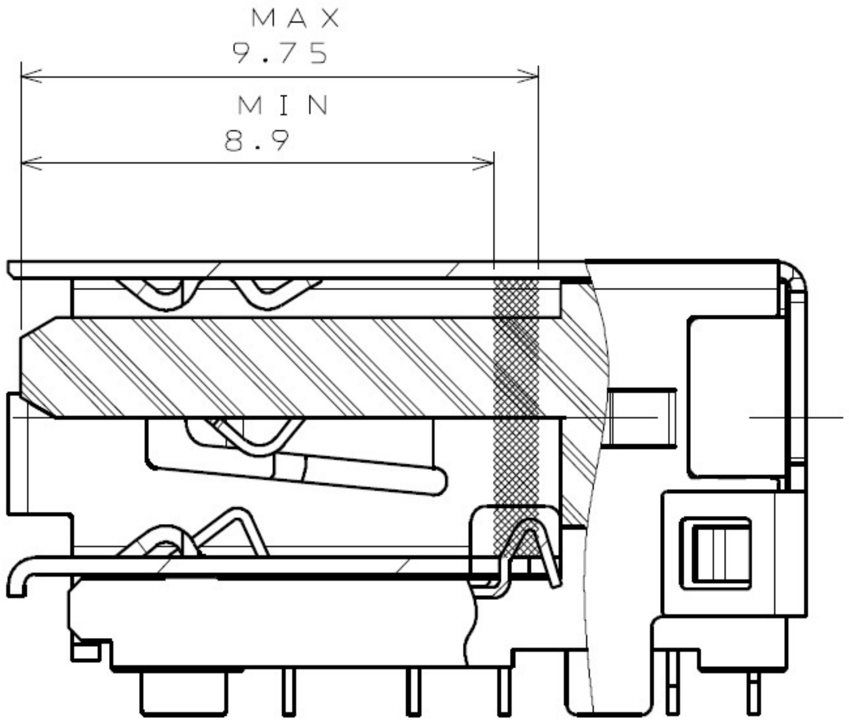


Figure 3-14 PD Detect Plane Location Range for PD Standard-A Receptacles

3.2.3 USB 2.0 PD Standard-A Connector

3.2.3.1 Interface Definition

Figure 3-15, Figure 3-16, and Figure 3-17 show the USB 2.0 PD Standard-A plug, the USB 2.0 PD Standard-A receptacle and a reference footprint for the USB 2.0 PD Standard-A receptacle, respectively.

Mechanical requirements governing mating interoperability of the USB 2.0 PD Standard-A receptacle and USB PD Standard-A plug are included in this specification. For the USB 2.0 PD receptacle, Datum A was moved from the front edge of the receptacle shell to the front of the tongue to be consistent with **[USB3.0]** and to provide proper dimensioning for PD Detect features. Similarly for the USB 2.0 PD Standard-A plug, Datum A was moved from the front edge of the plug shell to the back of the tongue.

Figure 3-15 defines the mechanical requirements that govern the mating interoperability that shall be followed for the USB 2.0 Standard-A plug. See Section 3.2.2 for additional USB PD Standard-A plug requirements.

Figure 3-16 defines the normative mechanical requirements of the USB 2.0 PD Standard-A receptacle that govern the mating interoperability and informative dimensions for solder tail locations, a sample Insertion Detect implementation, and a sample PD Detect implementation. Dimensions associated with solder tails, the sample Insertion Detect feature, and the sample PD Detect features are not normative. The solder pin locations are vendor-specific and included in the drawings for reference only. See Section 3.2.1 and Section 3.2.2 for normative Insertion Detect and PD Detect mechanical requirements, respectively.

The through-hole footprint in Figure 3-17 is shown as an example. Other footprints are allowed.

General considerations:

- There may be some increase in the USB 2.0 PD Standard-A receptacle connector depth (into a system board) as compared to the USB 2.0 Standard-A receptacle to accommodate detection of the Standard-A plug's longer shell.
- Drawings for stacked USB 2.0 PD Standard-A receptacles are not shown in this specification. They are allowed, as long as they meet all the electrical requirements defined in ***[USB2.0]*** and the mating

interoperability mechanical and electrical requirements defined in this specification. When designing stacked USB 2.0 PD Standard-A receptacles, PD Detect contacts shall be provided in all PD capable receptacles.

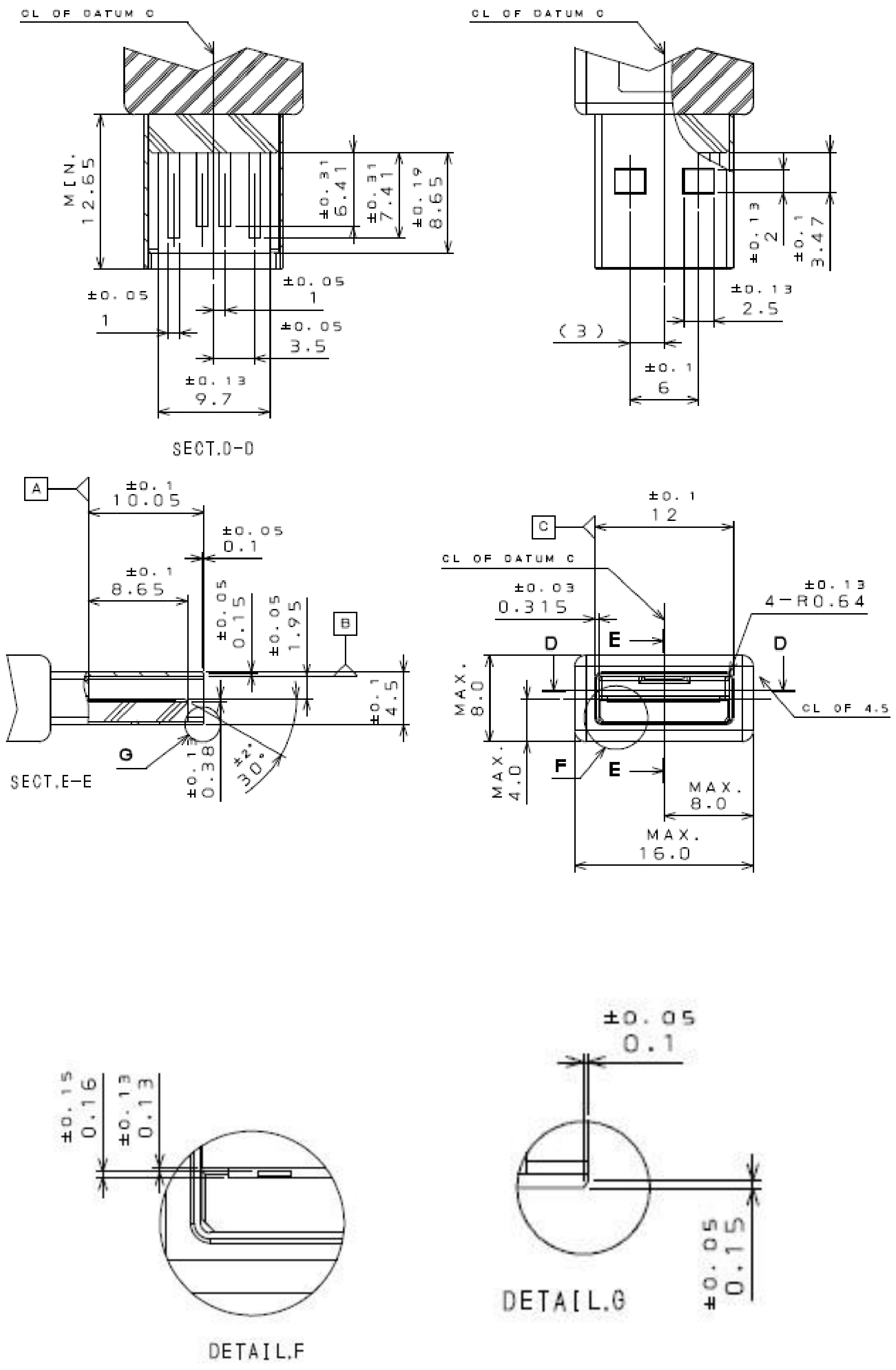
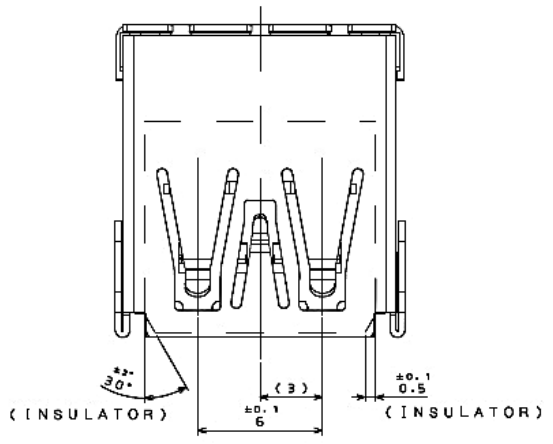
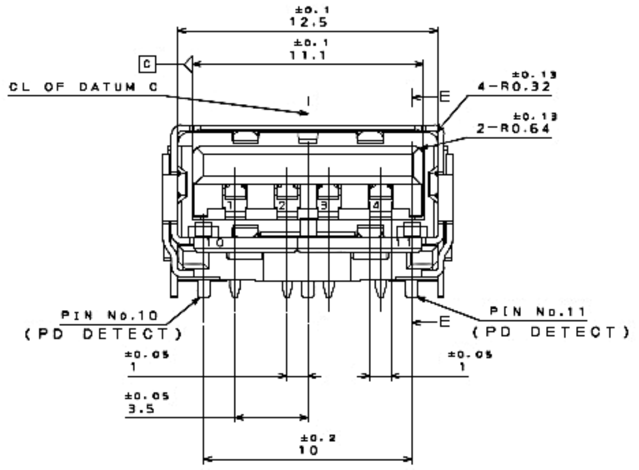
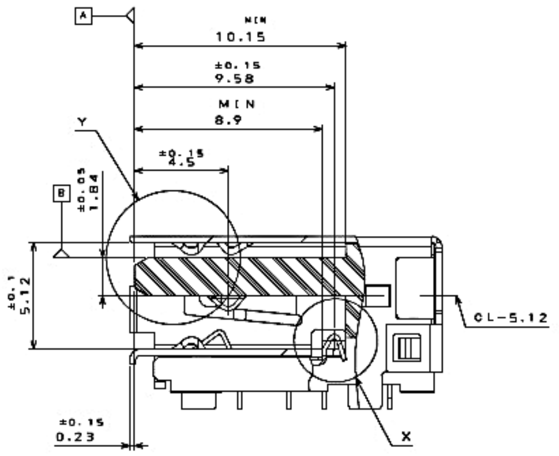


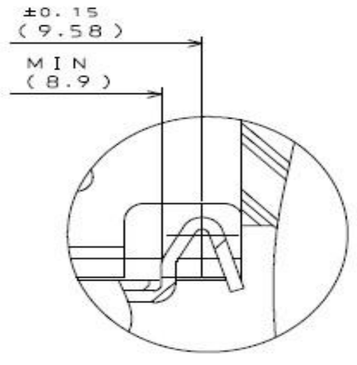
Figure 3-15 PD Standard-A Plug Interface Dimensions



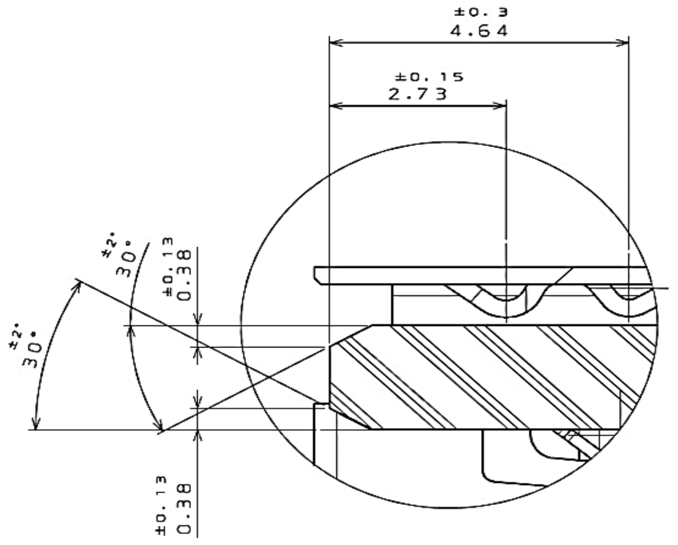




SECT. E-E



DETAIL X



DETAIL Y

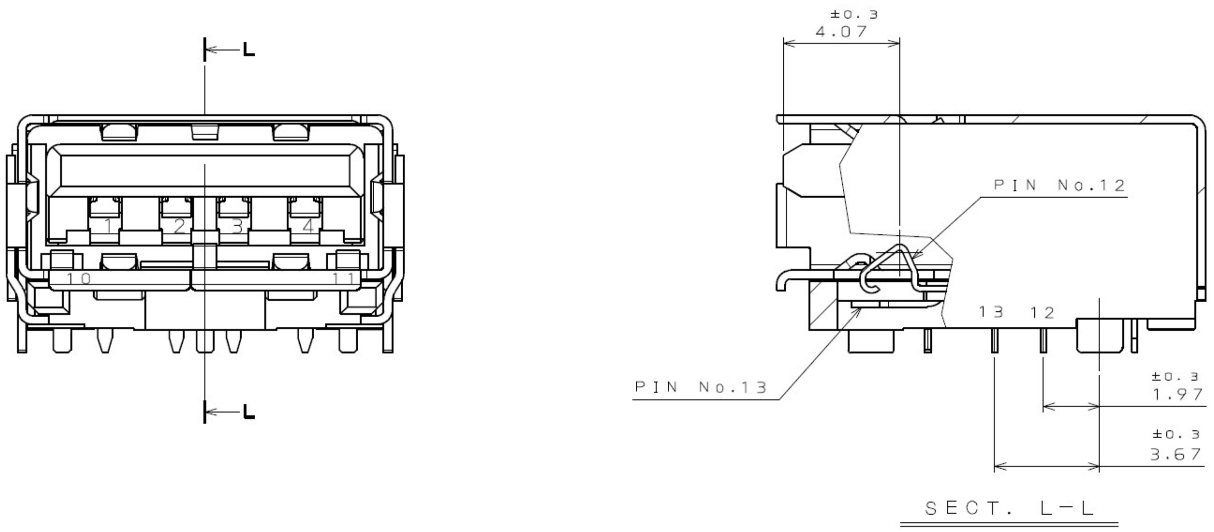


Figure 3-16 USB 2.0 PD Standard-A Receptacle Interface Dimensions

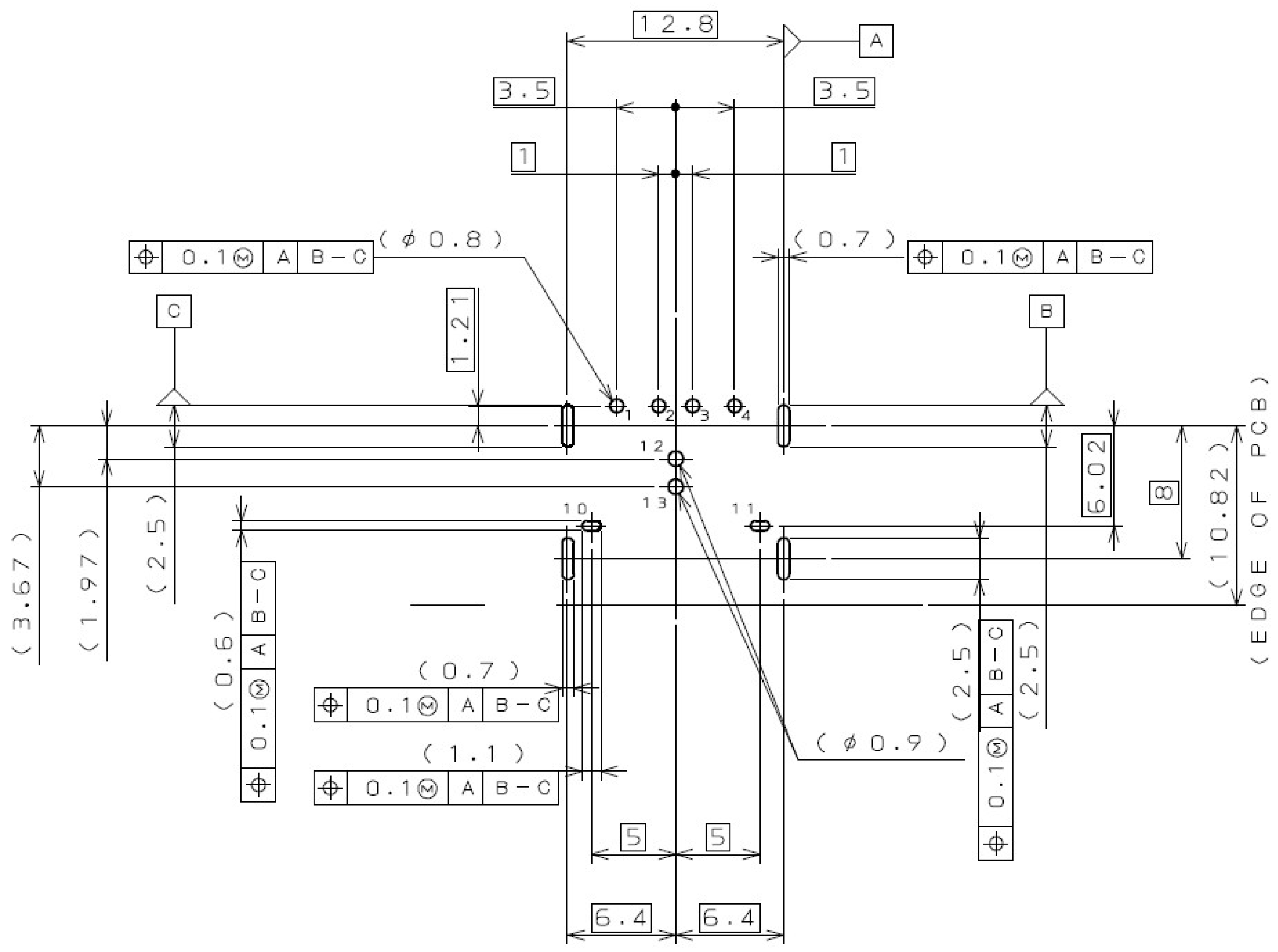


Figure 3-17 Reference Footprint for the USB 2.0 PD Standard-A Top Mount Single Receptacle (Informative)

3.2.3.2 Pin Assignments and Description

The usage and assignments of the pins that shall be used in the USB 2.0 PD Standard-A connector are defined in Table 3-3.

Table 3-3 USB 2.0 PD Standard-A Connector Pin Assignments

Pin Number[1]	Signal Name	Description	Mating Sequence
1	V_{BUS}	Power	Third
2	D-	Differential pair as defined in *[USB2.0]*	Fourth
3	D+		
4	GND	Ground for power return	Third
10[2]	PD DETECT 1	Contact in PD receptacle to detect a PD plug	Last
11[2]	PD DETECT 2	Contact in PD receptacle to detect a PD plug	Last

Pin Number[1]	Signal Name	Description	Mating Sequence
12[3],13	INSERTION DETECT	Receptacle only. Detects insertion of a plug into the receptacle. Optional except for cold socket applications.	Second
Shell	Shield	Connector metal shell	First
Note 1: Pin numbers not included in this table do not have contacts present. Pin numbering is consistent with location across multiple USB connector types. Note 2: Implementation of PD DETECT shall include: a) either pin 10 or pin 11. b) both pin 10 and pin 11. Note 3: Pin 12, if present, shall be connected to Shield.			

The physical location of the pins in the connector is illustrated inFigure 3-16.

3.2.4 USB 3.0 PD Standard-A Connector

3.2.4.1 Interface Definition

Figure 3-18, Figure 3-19, and Figure 3-20 show the USB 3.0 PD Standard-A plug, the USB 3.0 PD Standard-A receptacle and a reference footprint for the USB 3.0 PD Standard-A receptacle, respectively.

Figure 3-18 defines the mechanical requirements of the USB 3.0 PD Standard-A plug that govern the mating interoperability that are different than the dimensions in the ***[USB2.0]***and **[USB3.0]** specifications, that shall be followed, for the USB 3.0 Standard-A plug. See Section 3.2.1 for additional USB PD Standard-A plug requirements.

Figure 3-19 provides an informative example of a USB 3.0 PD Standard-A receptacle including dimensions for solder tail locations, a sample Insertion Detect implementation, and a sample PD Detect implementation. The dimensions in Figure 3-19 associated with solder tails, the sample Insertion Detect feature, and the sample PD Detect features are not normative. The solder pin locations are vendor-specific and included in the drawings for reference only. The USB 3.0 PD Standard-A receptacle mating interface shall comply with the dimensions defined in the ***[USB2.0***] and **[USB3.0]** specifications for the USB 3.0 Standard-A receptacle. See Section 3.2.1 and Section 3.2.2 for normative Insertion Detect and PD Detect mechanical requirements, respectively.

The through-hole footprint in Figure 3-20 is shown as an example. Other footprints are allowed.

General considerations:

- Drawings for stacked USB 3.0 PD Standard-A receptacles are not shown in this specification. They are allowed, as long as they meet all the electrical and mechanical requirements defined in ***[USB2.0]***, ***[USB3.0]***, and this specification. When designing a stacked USB 3.0 PD Standard-A receptacle, PD Detect contacts shall be provided in all PD capable receptacles.

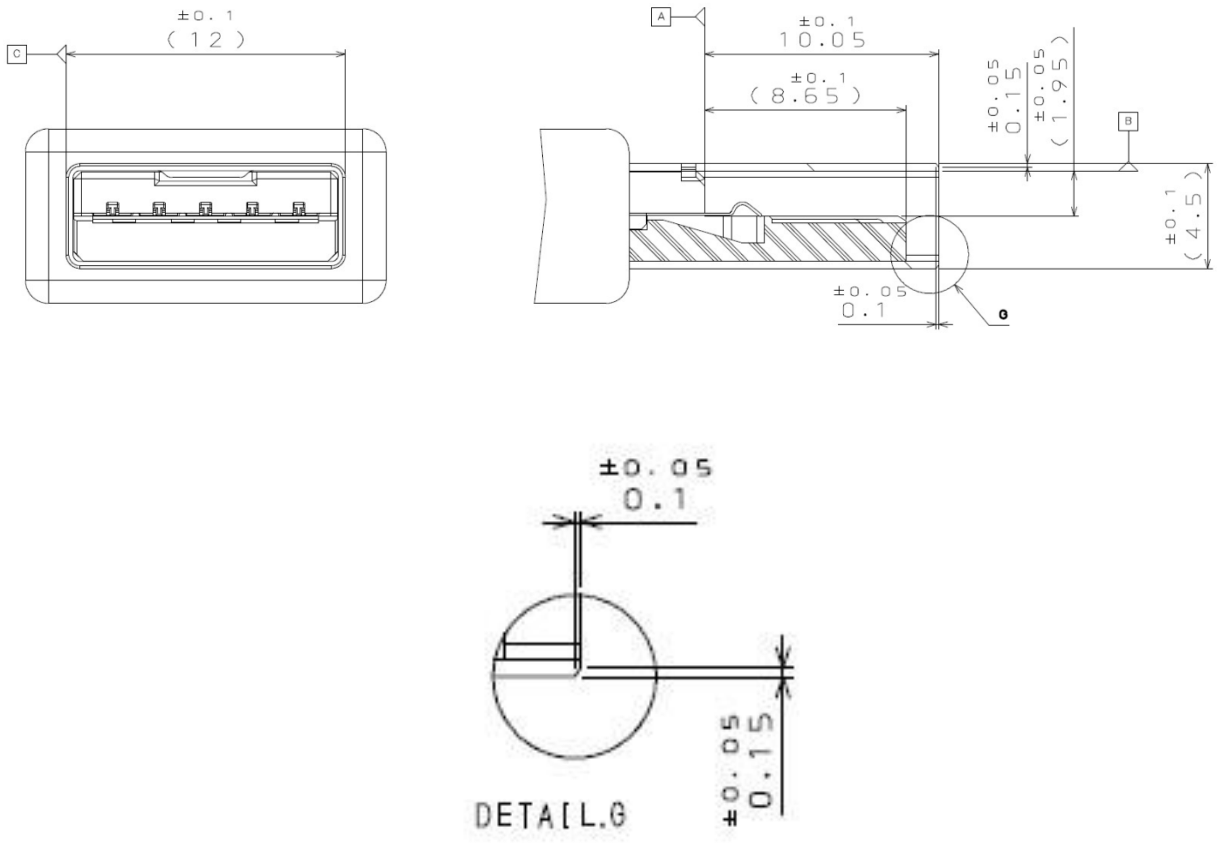


Figure 3-18 USB 3.0 PD Standard-A Plug Interface Dimensions

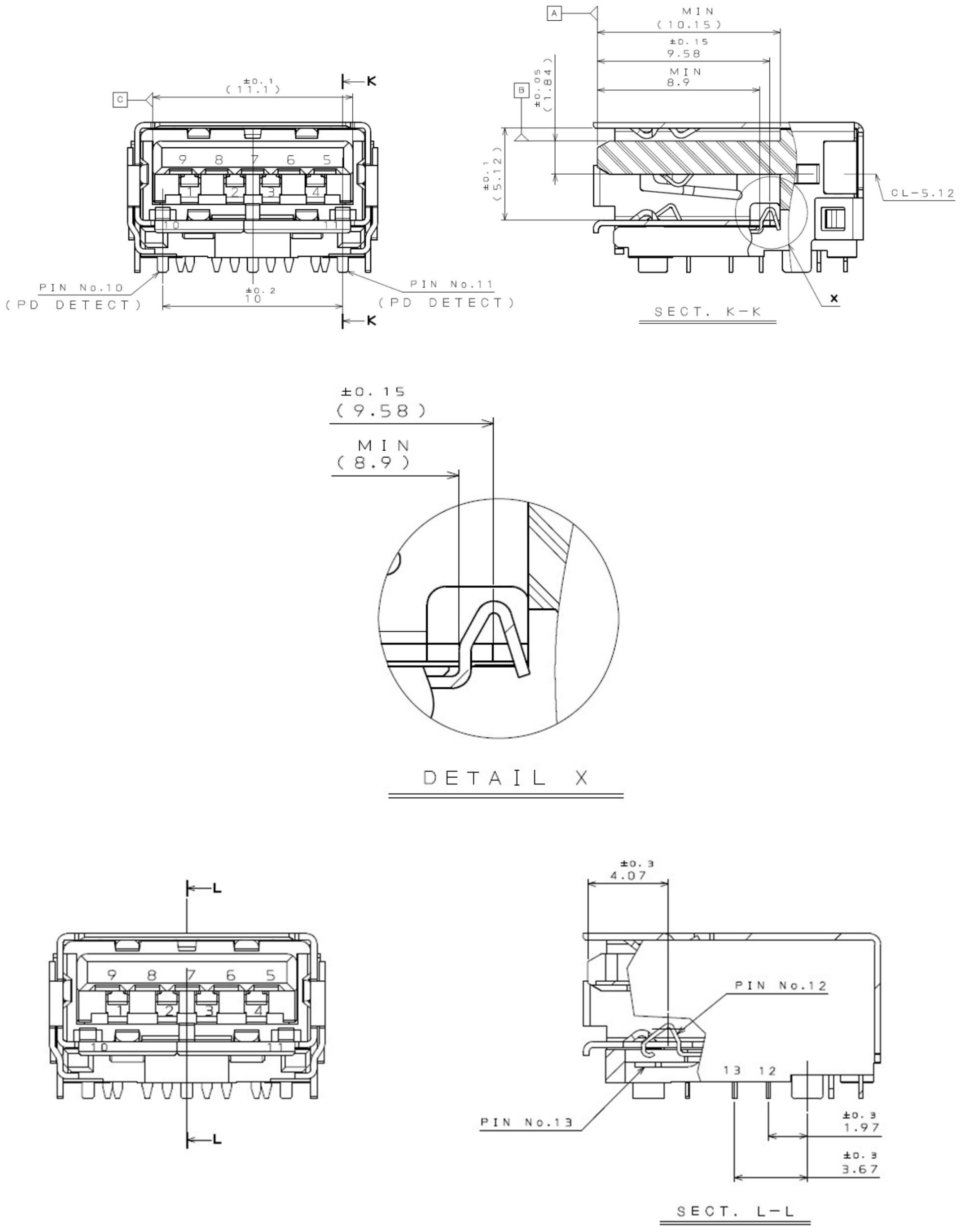


Figure 3-19 Reference USB 3.0 PD Standard-A Receptacle Interface Dimensions (Informative)

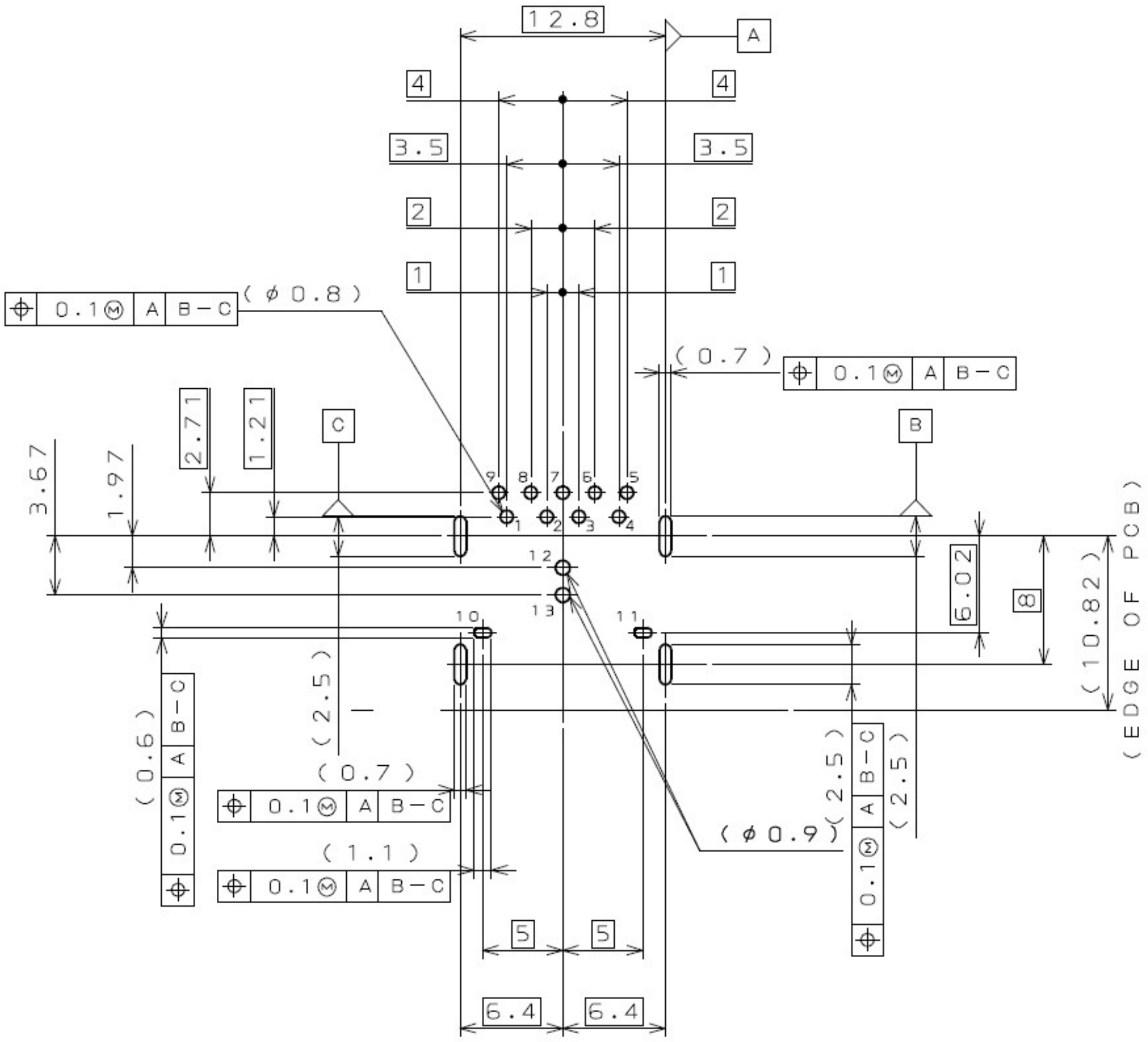


Figure 3-20 Reference Footprint for the USB 3.0 PD Standard-A Top Mount Single Receptacle (Informative)

3.2.4.2 Pin Assignments and Description

The usage and assignments of the pins that shall be used in the USB 3.0 PD Standard-A connector are defined in Table 3-4.

Table 3-4 USB 3.0 PD Standard-A Connector Pin Assignments

Pin Number[1]	Signal Name[2]	Description	Mating Sequence
1	V_{BUS}	Power	Third
2	D-	Differential pair as defined in *[USB2.0]*	Fourth
3	D+		
4	GND	Ground for power return	Third
5	StdA_SSRX-	SuperSpeed receiver differential pair	Fifth
6	StdA_SSRX+		

Pin Number[1]	Signal Name[2]	Description	Mating Sequence
7	GND_DRAIN	Ground for signal return	
8	StdA_SSTX-	SuperSpeed transmitter differential pair	
9	StdA_SSTX+		
10[3]	PD DETECT 1	Contact in PD receptacle to detect a PD plug	Last
11[3]	PD DETECT 2	Contact in PD receptacle to detect a PD plug	Last
12[4], 13	INSERTION DETECT	Receptacle only. Detects insertion of a plug into the receptacle. Optional except for cold socket applications.	Second
Shell	Shield	Connector metal shell	First

Note 1: Pin numbers not included in this table do not have contacts present. Pin numbering is consistent with location across multiple USB connector types.

Note 2: Tx and Rx are defined from the host perspective.

Note 3: Implementation of PD DETECT shall include :

a) either pin 10 or pin 11.

b) both pin 10 and pin 11.

Note 4: Pin 12, if present, shall be connected to Shield.

The physical location of the pins in the connector is illustrated in Figure 3-19. Note: pins 1 to 4 are referred to as the USB 2.0 pins, while pins 5 to 9 are referred to as the SuperSpeed pins.

3.2.5 USB 2.0 PD Standard-B Connector

3.2.5.1 Interface Definition

Figure 3-22, Figure 3-21, and Figure 3-23 show the USB 2.0 PD Standard-B plug, the USB 2.0 PD Standard-B receptacle, and a reference footprint for the USB 2.0 PD Standard-B receptacle, respectively. Solder pin locations on the USB 2.0 PD Standard-B receptacle are vendor-specific and are included in the drawings for reference only. The views in the plug and receptacle figures correspond to those shown in the ***[USB2.0]*** base specification and non-reference dimensions define the ID pin for PD applications. The USB 2.0 PD Standard-B plug and receptacle shall comply with the mechanical requirements specified in the ***[USB2.0]*** base specification and the ID pin as specified in Figure 3-22 and Figure 3-21.

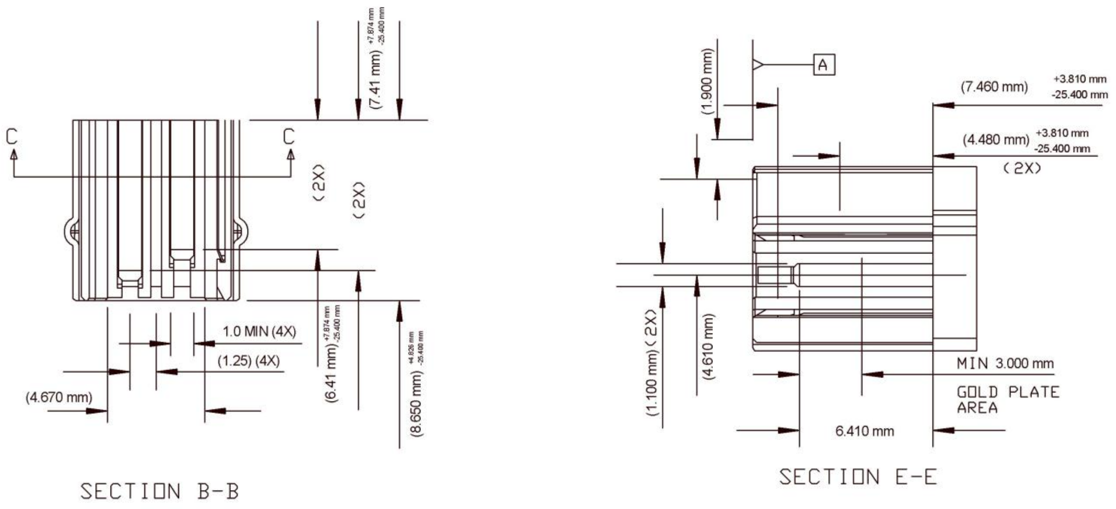

C
C
(7.41 mm) +7.874 mm -25.400 mm
(2X)
(2X)
1.0 MIN (4X)
(1.25) (4X)
(4.670 mm)
(6.41 mm) +7.874 mm -25.400 mm
(8.650 mm) +4.826 mm -25.400 mm
SECTION B-B
(1.900 mm)
A
(7.460 mm) +3.810 mm -25.400 mm
(4.480 mm) +3.810 mm -25.400 mm
(2X)
(1.100 mm) (2X)
(4.610 mm)
MIN 3.000 mm
GOLD PLATE AREA
6.410 mm
SECTION E-E

Figure 3-21 USB 2.0 PD Standard-B Plug Interface Dimensions

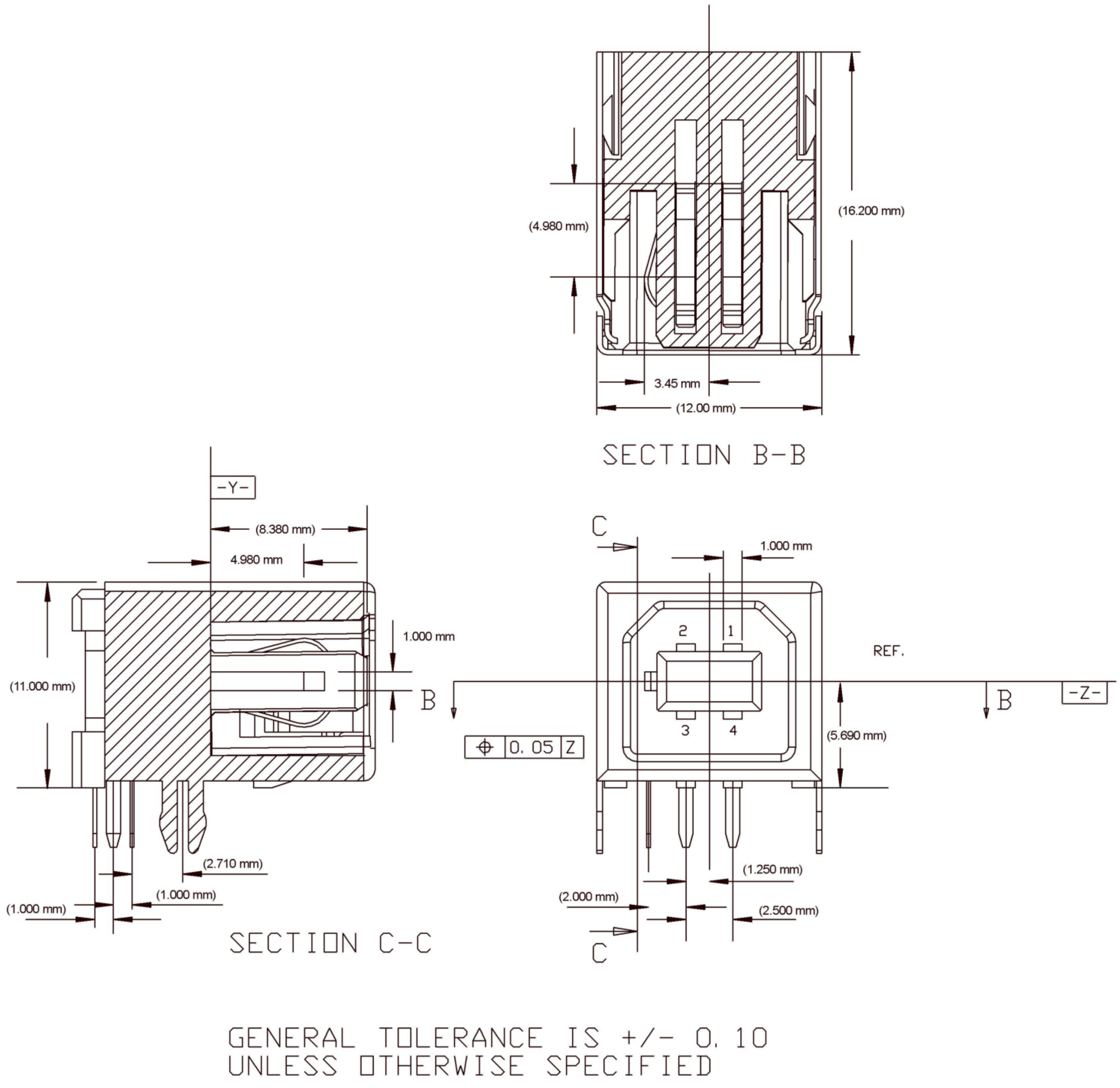


Figure 3-22 USB 2.0 PD Standard-B Receptacle Interface Dimensions

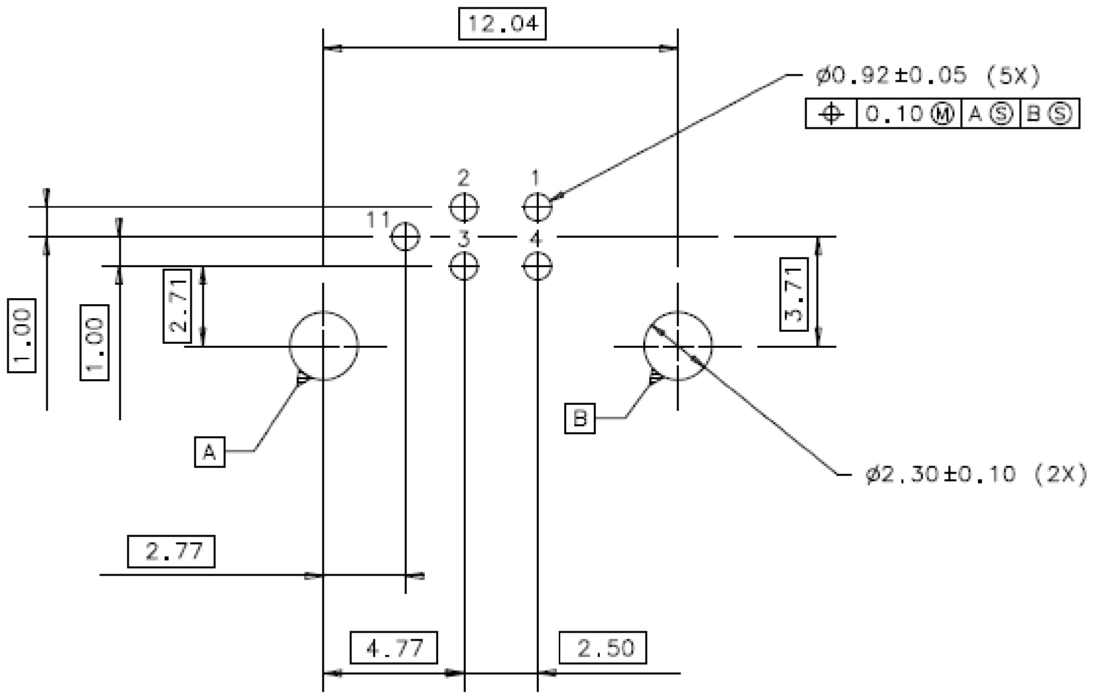


Figure 3-23 Reference Footprint for the USB 2.0 PD Standard-B Receptacle

3.2.5.2 Pin Assignments and Descriptions

The usage and assignments of the pins that shall be used in the USB 2.0 PD Standard-B connector are defined in Table 3-5.

Table 3-5 USB 2.0 PD Standard-B Connector Pin Assignments

Pin Number[1]	Signal Name	Description	Mating Sequence
1	V_{BUS}	Power	Second
2	D-	Differential pair as defined in *[USB2.0]*	Third or beyond
3	D+		
4	GND	Ground for power return	Second
11[2]	ID	Identification of PD capability	Third or beyond
Shell	Shield	Connector metal shell	First
Note 1: Pin numbers not included in this table do not have contacts present. Pin numbering is consistent with location across multiple USB connector types. Note 2: ID pin as defined in core USB specifications and extended by the USB Power Delivery specifications. See Section 3.4.3. Pin 11 was defined by *[USB3.0]* as the Ground Return for DPWR (power supplied by the device) for the *[USB3.0]* Powered-B connector.			

The physical locations of the pins in the connector are illustrated in Figure 3-22 and Figure 3-21.

3.2.6 USB 3.0 PD Standard-B Connector

3.2.6.1 Interface Definition

Figure 3-24 , Figure 3-25, and Figure 3-26 show the USB 3.0 PD Standard-A plug, the USB 3.0 PD Standard-A receptacle, and a reference footprint for the USB 3.0 PD Standard-A receptacle, respectively. Solder pin locations on the USB 3.0 PD Standard-B receptacle are vendor-specific and included in the drawings for reference only. The views in the plug and receptacle figures correspond to those shown in the *[USB3.0]* base specification and non-reference dimensions define the ID pin for PD applications. The USB 3.0 PD Standard-B plug and receptacle shall comply with all other mechanical requirements specified in the *[USB3.0]* base specification and the ID pin as specified in Figure 3-24 and Figure 3-25.

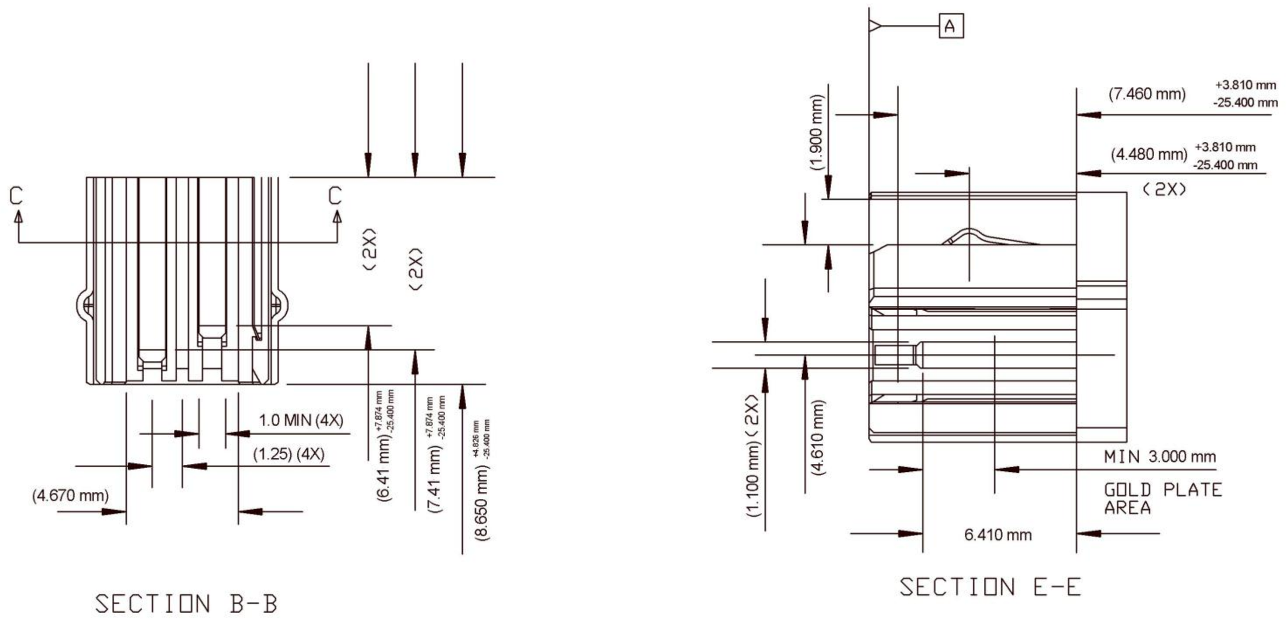
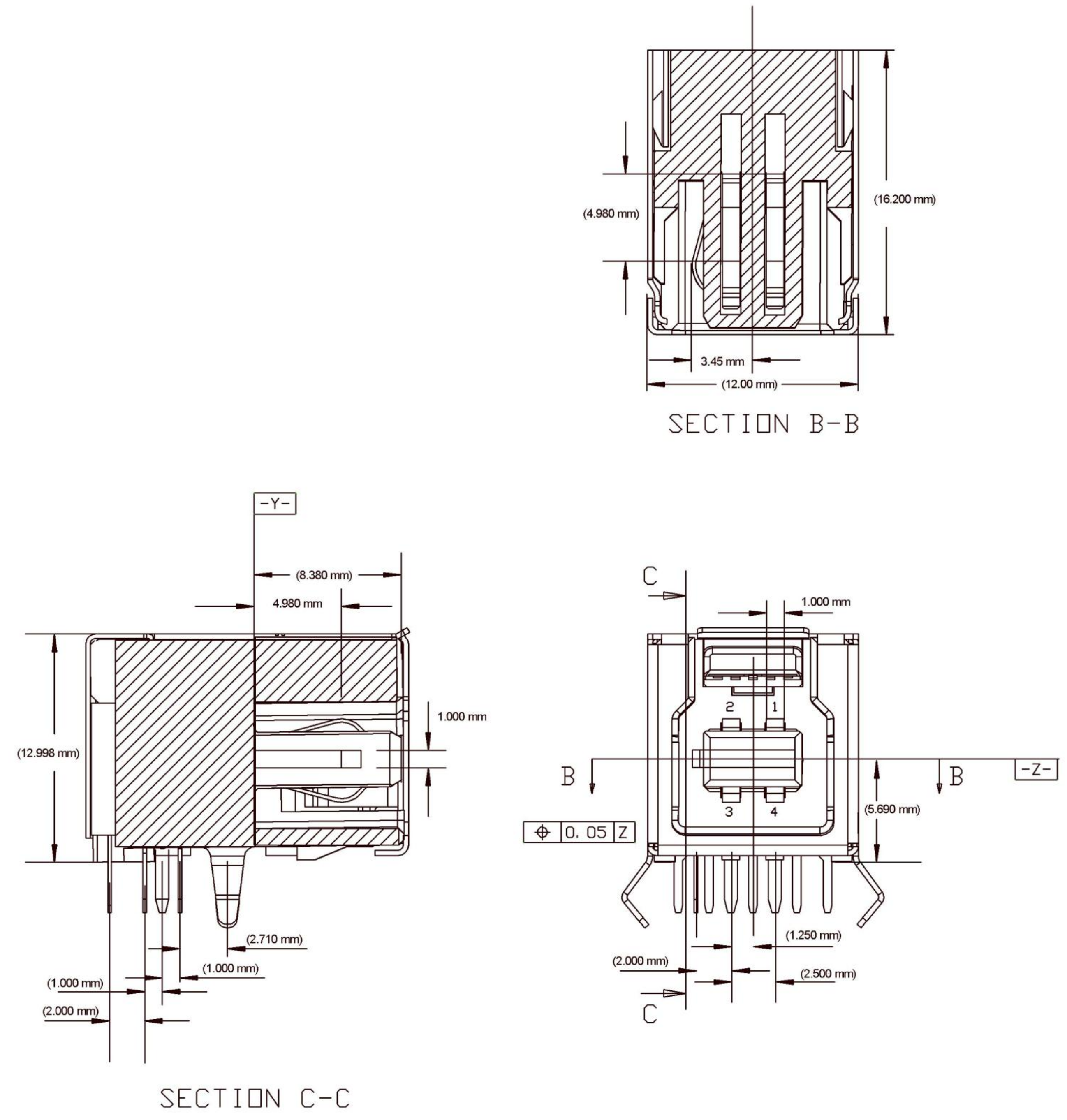

C
C
(2X)
(2X)
1.0 MIN (4X)
(1.25) (4X)
(4.670 mm)
(6.41 mm) +7.874 mm -25.400 mm
(7.41 mm) +7.874 mm -25.400 mm
(8.650 mm) +4.826 mm -25.400 mm
SECTION B-B
A
(7.460 mm) +3.810 mm -25.400 mm
(4.480 mm) +3.810 mm -25.400 mm
(2X)
(1.900 mm)
(1.100 mm) (2X)
(4.610 mm)
MIN 3.000 mm
GOLD PLATE AREA
6.410 mm
SECTION E-E

Figure 3-24 USB 3.0 PD Standard-B Plug Interface Dimensions



GENERAL TOLERANCE IS +/- 0.10
UNLESS OTHERWISE SPECIFIED

Figure 3-25 USB 3.0 PD Standard-B Receptacle Interface Dimensions

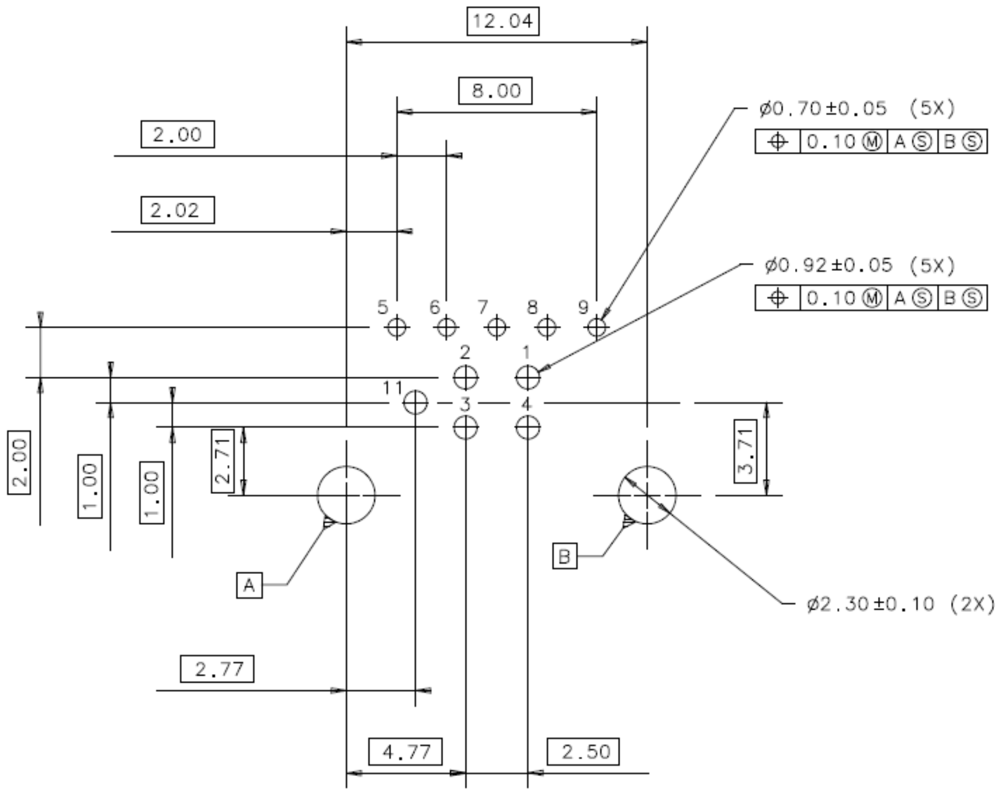


Figure 3-26 Reference Footprint for the USB 3.0 PD Standard-B Receptacle

3.2.6.2 Pin Assignments and Descriptions

The usage and assignments of the pins that shall be used in the USB 3.0 PD Standard-B connector are defined in Table 3-6.

Table 3-6 USB 3.0 PD Standard-B Connector Pin Assignments

Pin Number[1]	Signal Name[2]	Description	Mating Sequence
1	V_{BUS}	Power	Second
2	D-	Differential pair as defined in *[USB2.0]*	Third or beyond
3	D+		
4	GND	Ground for power return	Second
5	StdB_SSTX-	SuperSpeed transmitter differential pair	Third or beyond
6	StdB_SSTX+		
7	GND_DRAIN	Ground for signal return	
8	StdB_SSRX-	SuperSpeed receiver differential pair	
9	StdB_SSRX+		
11[3]	ID	Identification of PD capability	
Shell	Shield	Connector metal shell	First
Note 1: Pin numbers not included in this table do not have contacts present. Pin numbering is consistent with location across multiple USB connector types. Note 2: Tx and Rx are defined from the host perspective. Note 3: ID pin as defined in core USB specifications and extended by the USB Power Delivery specifications. See Section 3.4.3. Pin 11 was defined by *[USB3.0]* as the Ground Return for DPWR (power supplied by the device) for the USB 3.0 Powered-B connector.			

The physical locations of the pins in the connector are illustrated in Figure 3-23 and Figure 3-24.

3.3 Cable Assemblies

USB Power Delivery introduces the concept of an electronically marked cable assembly. The particular marking denotes the cable assembly's characteristics and is electronically detected. This provides Devices on each end of the cable assembly a means to detect and identify the cable assembly capabilities. The ID pin definition is extended to electronically mark cable assemblies. An ID pin has been added to the Standard-B Connectors to create PD Standard-B Connectors as defined in Section 3.2.5 and Section 3.2.6. Since only the Device can detect the ID pin, the Device shall detect the cable assembly and make requests that are consistent with the cable assembly's capabilities. Similarly, the Standard-A Connector shell has been modified to create a PD Standard-A Connector to allow detection of cable insertion and to identify if the cable assembly is PD-capable. The combination of these PD cable assembly markings provide a robust system to safely deliver high power across the USB cable assembly.

The portion of this specification that allows the negotiation of voltages other than the default ***vSafe5V*** and currents in excess of 1.5A shall apply only to Devices with marked PD cable assemblies.

3.3.1 Non-marked Cable Assemblies

Limitations are placed on the use of legacy cable assemblies (i.e., ones not marked). Legacy cable assemblies include all cable assemblies with USB 2.0 or USB 3.0 Standard-A Connectors, USB 2.0 or USB 3.0 Standard-B Connectors, USB 3.0 Powered-B Connectors, and USB 2.0 or USB 3.0 Micro-B Connectors. Devices shall only use these unmarked legacy cable assemblies at ***vSafe5V*** and up to 1.5A as described by ***[USB2.0]***, ***[USB3.0]*** and ***[BC1.2]***.

3.3.2 Marked Cable Assemblies

Marking allows the Device to detect the insertion of a USB Power Delivery cable assembly and its electrical characteristics. The Device shall not negotiate for currents in excess of the electrical characteristics indicated by the cable's marking.

3.4 PD Cable Assembly and PD Cable Adaptor Marking

Paragraphs in this section described the markings for the plug connectors at each end of the PD cable assembly or PD cable adaptor. References to cable assembly in this section apply to both cable assemblies and PD cable adaptors which include a PD plug connector.

3.4.1 Marker for PD Standard-A Connectors

The PD Standard-A Connector shell is 1.3 mm longer than the shell of the legacy Standard-A Connector. The PD Standard-A Connector shell shall be detected by the downstream Device connector using a PD Standard-A receptacle connector and associated marking detection circuitry to indicate that the cable assembly is a PD cable assembly. Marking detection circuitry is vendor specific.

3.4.2 Electronic Markers for Micro-A Plugs

In the standard cable assembly with a Micro-A plug, the ID-pin is grounded. The data and shield connections shall be made per the ***[USB2.0]*** and the ***[USB3.0]*** specifications.

3.4.2.1 Legacy Micro-A Plug

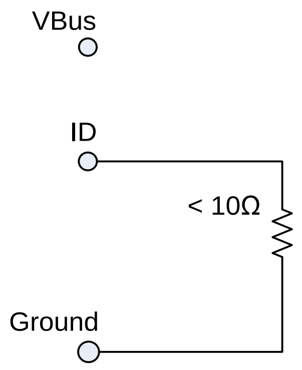


Figure 3-27 Schematic of a Micro-A Plug Legacy Termination

Figure 3-27 shows the schematic diagram of the Micro-A plug termination in a legacy cable assembly. Note the ID pin is connected to ground through a low impedance to indicate that the plug is an A plug. For PD to remain backward compatible, the low impedance to ground shall be maintained. However, any value less than 1kΩ will still be interpreted by a legacy port as a Micro-A plug.

PD uses two new markers to allow the detection of a low power cable assembly and PD cable assembly in addition to the detection of a legacy cable.

3.4.2.2 Low Power Micro-A Plug

Devices with a hardwired captive cable with a Micro-A plug connector supporting a one cell Lithium battery as their power source or where very low power consumption is important may terminate the Micro-A Plug's ID pin to ground with a 1kΩ resistor and a 0.01µF capacitor to V_{BUS}. This termination may be detected electrically. See 4.4.5 for additional information regarding low power Device characteristics.

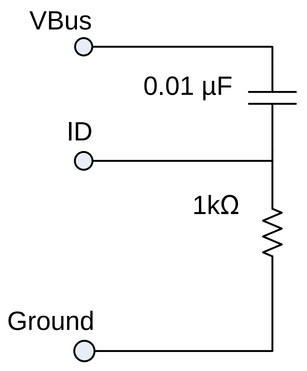


Figure 3-28 Schematic of a Micro-A Plug Marker Indicating Low Power Capability

3.4.2.3 PD Micro-A plug

PD cable assemblies with Micro-A plug connectors shall be marked with a resistor terminating the ID pin to ground. See Figure 3-29 for a schematic diagram of the connector termination in a PD cable assembly using a Micro-A plug.

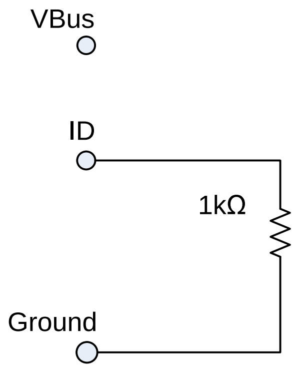


Figure 3-29 Schematic of a Micro-A PD Plug

3.4.3 Electronic Markers for PD Standard-B Plugs and Micro-B Plugs

Electronic markers for PD cable assemblies with PD Standard-B plug connectors and Micro-B plug connectors are specified in this section. A legacy Micro-B connector's ID pin exhibits a very high resistance to ground. To maintain backward compatibility, capacitors are used as markers.

Note the 3A and 5A markers are mutually exclusive; hence both markers shall not be present at the same time.

3.4.3.1 3A Capable PD B Plug

The schematic diagram for a 3A-capable PD cable assembly detailing how a PD Standard-B plug connector or a Micro-B plug connector shall be marked is shown in Figure 3-30.

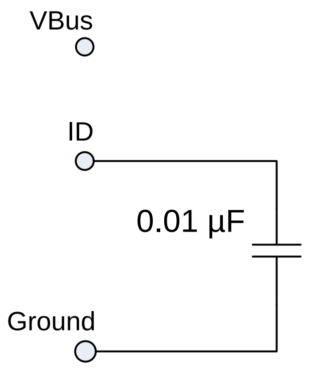


Figure 3-30 Schematic of a B Plug Connector Marker Indicating 3A Capability

3.4.3.2 5A Capable PD B Plug

The schematic diagram for a 5A-capable PD cable assembly detailing how a PD Standard-B plug connector shall be marked is shown in Figure 3-31.

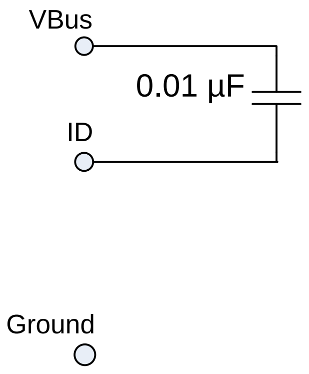


Figure 3-31 Schematic of a B Plug Connector Marker Indicating 5A Capability

3.5 USB PD Icon

The USB PD cable assemblies shall display the USB PD Icon illustrated in Figure 3-32. Note that the icons illustrated in Figure 3-32 are given as examples only and are not dimensioned or proportioned correctly A dimensioned drawing and allowable usage of the icon are supplied with the license from the USB-IF.

USB 2.0 PD icon:
USB 3.0 PD icon:
SS

Figure 3-32 USB PD Icon

The USB PD Icon will be present and visible near a USB PD receptacle. This provides easy user recognition and facilitates alignment during the mating process.

3.6 USB Power Delivery Cable Requirements

The methods used to mark the USB PD cable assemblies and USB PD cable adaptors are found in Section 3.4. The USB PD connector family shall conform to all requirements in Section 3, in addition to the requirements specified in the ***[USB2.0]***, ***[USB3.0]*** or ***[BC1.2]*** specifications. Test sequences shall conform to EIA-364-1000.001 as specified in the Environmental Requirements section of ***[USB3.0]***.

Some USB PD cable assemblies may be designed for use as power only (i.e., no USB data communication). Requirements that relate to the transfer of USB data may not apply to these cables.

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

The following requirement applies to the power contacts of a 3A PD cable assembly:

- 20mΩ (Max) initial for V_{BUS} and GND contacts.
- Maximum change (delta) of +10mΩ after environmental stresses.
- Measure at 20mV (Max) open circuit at 100mA.

The following requirement applies to the power contacts of a 5A PD cable assembly:

- 20mΩ (Max) initial for V_{BUS} and GND contacts.
- Maximum change (delta) of +10mΩ after environmental stresses.
- Measure at 20mV (Max) open circuit at 100mA.

The following requirements apply, independent of power being applied, to the resistance of a PD Standard-A plug receptacle shell to the PD Standard-A receptacle PD DETECT contact with a PD Standard-A plug in the mated condition or to the resistance between the INSERTION DETECT contacts when a Standard-A plug is present in the Standard-A receptacle supporting Insertion Detect. For PD Detect, there are two contact interfaces to the plug shell included in series in this measurement:

- 200mΩ (Max) initial.
- Maximum change (delta) of 300mΩ after environmental stresses.
- Measure at 20mV (Max) open circuit at 100mA.

3.6.2 Open Circuit Resistance

The following requirements apply, independent of power being applied, to the resistance of the PD Standard-A receptacle shell to the PD Standard-A receptacle PD DETECT contact(s) when no PD Standard-A plug is inserted in the mated condition or a non-PD Standard–A plug is inserted in the mated condition or to the resistance between the INSERTION DETECT contacts of a Standard-A receptacle supporting Insertion Detect when a Standard-A plug is not inserted:

- ≥ 10MΩ initial.
- ≥ 10MΩ after environmental stress.

3.6.3 Dielectric Strength (EIA 364-20)

No breakdown shall occur when 100VAC (RMS) is applied between adjacent contacts of unmated and mated connectors.

3.6.4 Insulation Resistance (EIA 364-21)

A minimum of 100MΩ insulation resistance is required between adjacent contacts of unmated and mated connectors.

3.6.5 Contact Current Rating

3.6.5.1 3A PD Connector Mated Pair (EIA 364-70, Method 2)

A current of 3.0 A shall be applied to V_{BUS} pin and its corresponding GND pin (e.g., pin 1 and pin 5 of the Micro-B Connector). When the current is applied to the contacts, the delta temperature shall not exceed +30°C at any point on the connectors under test, when measured at an ambient temperature of 25°C.

3.6.5.2 5A PD Connector Mated Pair (EIA 364-70, Method 2)

A current of 5.0A shall be applied to V_{BUS} pin and its corresponding GND pin (e.g., pin 1 and pin 4 of the PD Standard-B Connector). When the current is applied to the contacts, the delta temperature shall not exceed +30°C at any point on the connectors under test, when measured at an ambient temperature of 25°C.

3.6.6 Differential Crosstalk between V_{BUS} and the D+/D- Pair (EIA-360-90)

The differential, near-end, and far-end, crosstalk between the D+/D- pair and V_{BUS} shall be managed not to exceed the limit shown in Figure 3-33.

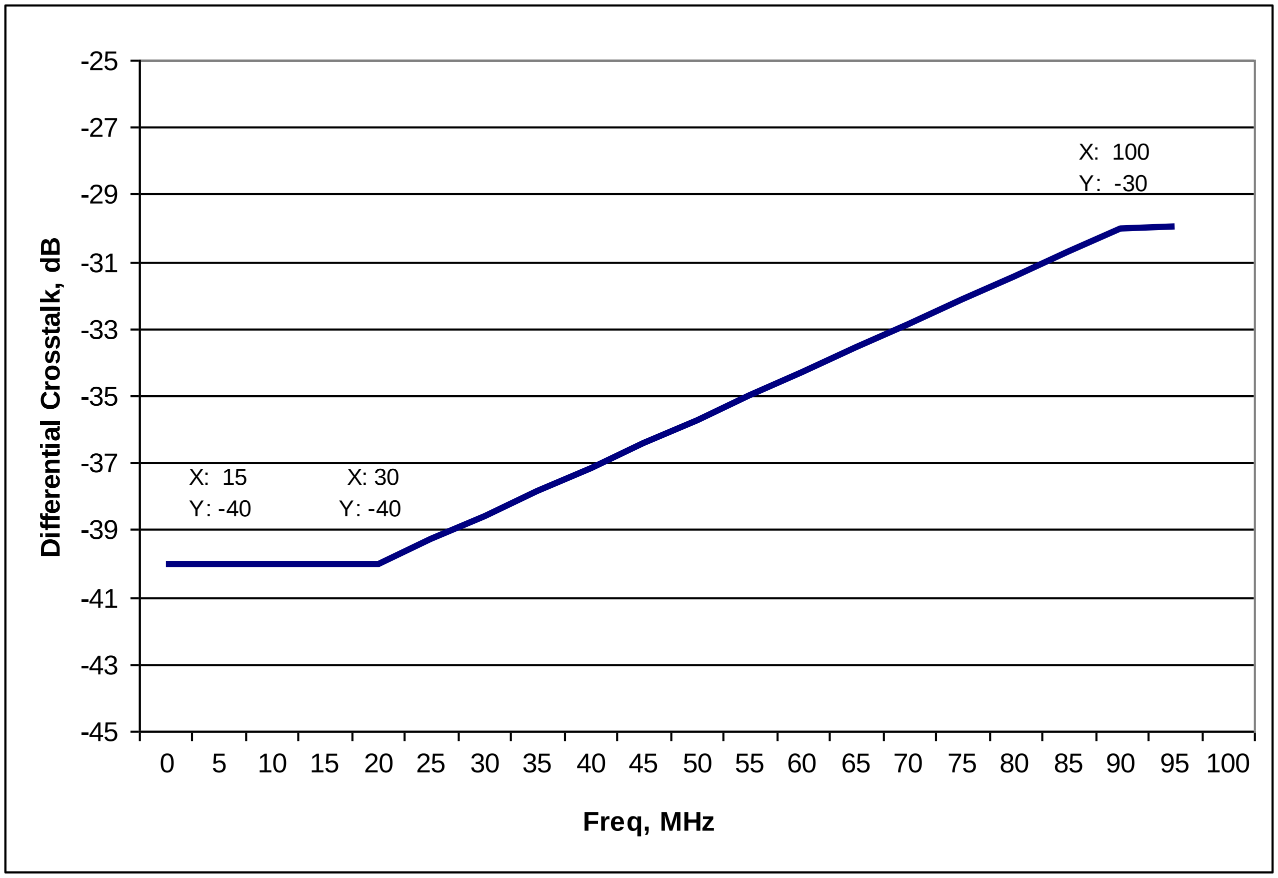


Figure 3-33 Differential Near-End and Far-End Crosstalk Requirement between the D+/D- Pair and V_{BUS}

3.6.7 PD Cable Assembly Shielding Connectivity

The shield conductor shall be attached to the connector shell at both ends of the cable and shall provide a resistance of no greater than 1.0Ω from end to end. The shield shall not be connected to ground within the cable.

3.6.8 PD Cable V_{BUS} Impedance

The cable impedance shall meet the requirements specified in Table 5-11.

3.6.9 PD Cable IR Drop Considerations

The maximum voltage drop between the Source and Sink ports is defined to:

- Insure signal integrity of the USB 2.0 signal wires.
- Quantify the worst case voltage seen at the input of a Sink.

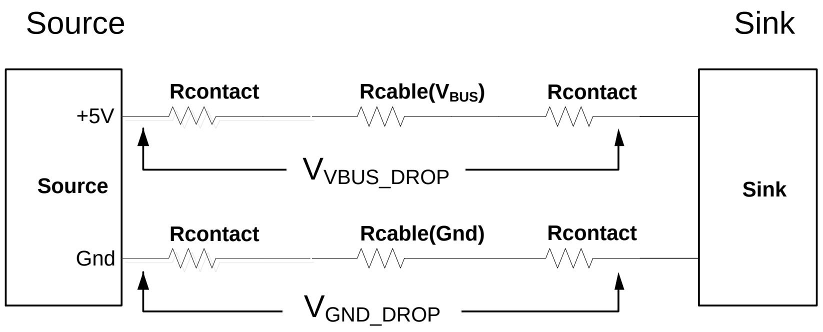


Figure 3-34 Voltage Drop Measurement

As shown in Figure 3-34, voltage drop across the cable is measured independently on both the GND and the V_{BUS} connections. It is inclusive of the cable and connectors at both ends.

3.6.9.1 Voltage Drop on the Ground

A PD cable assembly shall ensure ***V_{GND_DROP}*** does not exceed the maximum value listed in Table 3-7 for the signal ground reference between the Source's power source connection to the USB receptacle and the Sink's connection to its internal power conversion block, if present. ***V_{GND_DROP}*** shall be measured at ***iCable*** and measured either:

- At the receptacle's solder pad where it attaches to the printed circuit board
- Or where the captive cable is attached to the printed circuit board in the device.

V_{GND_DROP} shall include the effects of the following:

1) The mated contact resistance of both the A-side and the B-side (if present) connections for GND.
2) The series resistance of the cable's GND wire.
3) The rated current of the cable (***iCable***).

V_{GND_DROP} for removable cables equals (2 * Rcontact + Rcable(GND)) * ***iCable***. ***V_{GND_DROP}*** for captive cables equals (1 * Rcontact + Rcable(GND)) * ***iCable***.

The only variable in the above equations is Rcable(GND). The wire size and its length are the major contributors to V_{GND_DROP}. The ground wire size shall be selected so that ***V_{GND_DROP}*** at a current of ***iCable*** does not exceed the maximum value listed in Table 3-7

3.6.9.2 Overall IR Drop between a Source and a Sink

A PD cable assembly shall provide a maximum voltage drop of ***vIRDrop_Cable*** for V_{BUS} to GND at the Sink end of the cable. The V_{BUS} to GND voltage drop shall include ***V_{VBUS_DROP}*** plus ***V_{GND_DROP}***.

V_{VBUS_DROP} shall be measured at a current of ***iCable*** and measured either:

- At the receptacle's solder pad where it attaches to the printed circuit board
- Or where the captive cable is attached to the printed circuit board in the device.

V_{VBUS_DROP} shall include the effects of the following

- The mated contact resistance of both the A-side and B-side (if present) connections for V_{BUS}.
- The series resistance of cable's V_{BUS} wire.
- The rated current of the cable (***iCable***).

V_{VBUS_DROP} for removable cables equals (2 * Rcontact + Rcable(V_{BUS})) * ***iCable***. ***V_{VBUS_DROP}*** for captive cables equals (1 * Rcontact + Rcable(V_{BUS})) * ***iCable***.

The only variable in the above equations is Rcable(V_{BUS}). The wire size and its length are the major contributors to ***V_{VBUS_DROP}***. The V_{BUS} wire size shall be selected so that ***V_{VBUS_DROP}*** at a current of ***iCable*** does not exceed the maximum value listed in Table 3-7.

3.6.9.3 Example Calculation of Overall IR Drop

The goal of this example is to design a compliant PD Cable with a Standard A and a Micro B plug (e.g. ***iCable*** = 3A) that is 1m in length.

The allowable Rcable(GND) to achieve this is found by:

1) R=E/I or R = IR -> Rcable(GND) = the maximum value for ***VGND_DROP*** listed in Table 3-7/***iCable*** -> 0.375/3 = 125mΩ.
2) Mated contact resistance is defined as 30mΩ (Max) times 2 mated pair connections or 60mΩ.
3) Rcable(GND) = 125mΩ – 60mΩ = 65mΩ (the budget for the ground wire in the cable).

The allowable Rcable(V_{BUS}) to achieve this can be found by:

1) R=E/I or R = IR -> Rcable(V_{BUS}) = the maximum value for ***VVBUS_DROP*** listed in Table 3-7/***iCable*** -> 0.625/3 = 208mΩ.
2) Mated contact resistance is defined as 30mΩ (Max) times 2 mated pair connections or 60mΩ.
3) Rcable(V_{BUS}) = 208mΩ – 60mΩ = 148mΩ (the budget for the V_{BUS} wire in the cable).

In this example, a 22AWG @ 52.94mΩ /m used for Ground and a 26AWG @ 134mΩ /m used for V_{BUS} meets the ***vIRDrop_Cable*** requirements for a 1m cable.

3.6.9.4 Effects of Adapters on Overall IR Drop

Introduction of an adapter has significant impact on the overall IR Drop budget as it introduces two additional mated pairs of contact resistances to consider. Use of an adapter may result in violations of IRDrop_Cable and/or more ***V_{GND_DROP}***. Violation of either may compromise a device's performance.

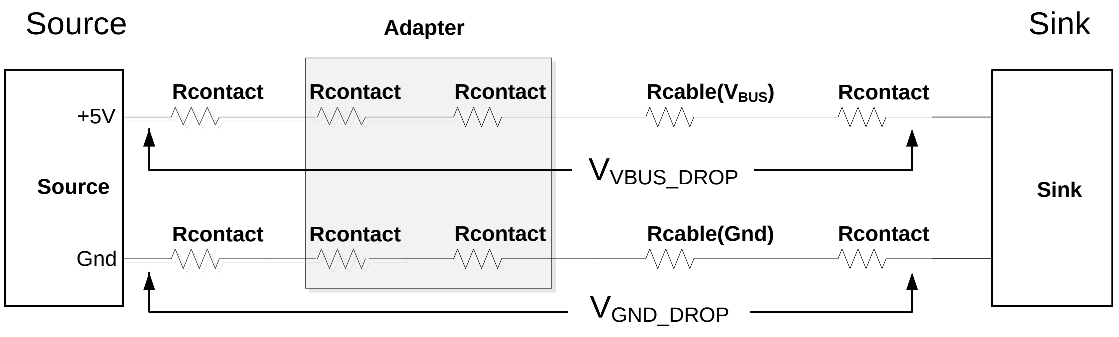


Figure 3-35 Effects of an Adapter on Overall IR Drop

Table 3-7 Electrical Parameters

Parameter	Minimum	Maximum	Units	Description
V_{VBUS_DROP}[1]		625	mV	
V_{GND_DROP}[1]		375	mV	Note: this value is the same as V_{GND_OFFSET} as defined in ***[BC1.2]***.
vIRDrop_Cable		1	V	Sum of ***V_{VBUS_DROP}*** and ***V_{GND_DROP}***.

Parameter	Minimum	Maximum	Units	Description
iCable	Current carrying capacity depends on the type of connectors deployed between the attached ports. Hence the current is defined as: • 5A if only Standard connectors are used. • 3A if a micro connector is present at any point.			

Note 1: For Charging Only cables (e.g. cables without data lines), the maximum value of either V_{VBUS_DROP} or V_{GND_DROP} may be exceeded, however V_{VBUS_DROP} plus V_{GND_DROP} shall not exceed ***vIRDrop_Cable.***

4. Electrical Requirements

This chapter covers the Platform's electrical requirements for implementing USB Power Delivery.

USB Power Delivery may be implemented alongside the ***[USB2.0]***, ***[USB3.0]*** and ***[BC1.2]*** specifications. In the case where a Device requests power via the Battery Charging Specification and then the USB Power Delivery Specification, it shall follow the USB Power Delivery Specification. If the USB Power Delivery connection is lost, the port shall return to its default state, see Section 6.7.2.

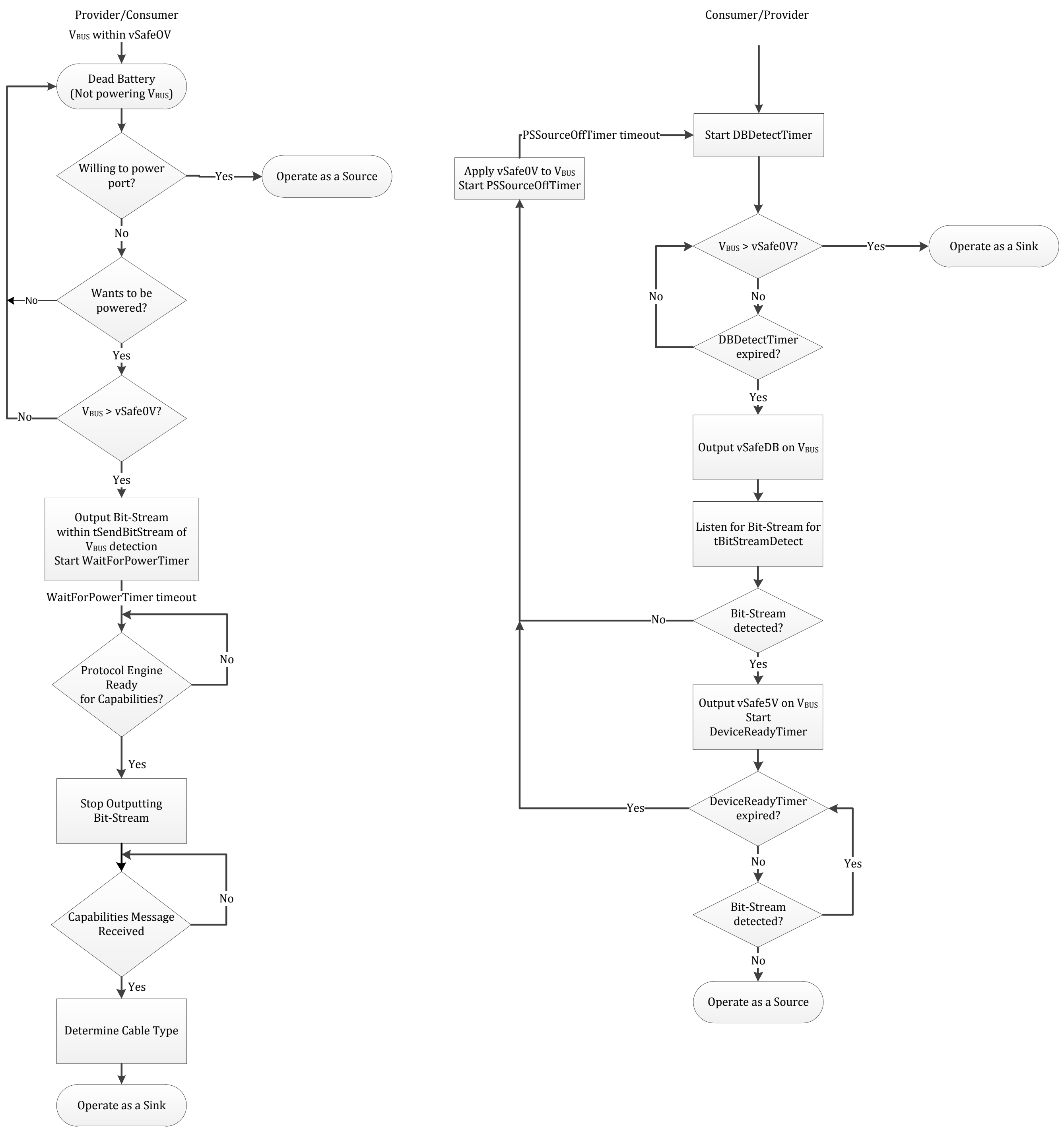
4.1 Dead Battery Detection / Unpowered Port Detection

The USB Power Delivery specification defines a mechanism for a Consumer/Provider to provide power to a Provider/Consumer under the following circumstances:

- The Provider/Consumer has a dead battery that requires charging or
- The Provider/Consumer has lost its power source.

A Consumer/Provider shall back power the Provider/Consumer at a reduced current level. This enables the Provider/Consumer's PD transmitter on its downstream port to send a continuous stream of alternating '0s' and '1s' (referred to in this section as a Bit Stream) that the Consumer/Provider's upstream port's PD receiver can readily detect. The use of a current limited Source (***vSafeDB***) to back power V_{BUS} is intended to minimize the risk of damage to legacy USB downstream ports. At the start of the process when the Consumer/Provider sees no voltage on V_{BUS}, it probes the bus by back powering V_{BUS} to see if its Port Partner has a dead battery or unpowered port that it wants powered. If so, the Consumer/Provider outputs ***vSafe5V*** on V_{BUS} and becomes the defacto power Provider.

When the process is complete, both ports will have performed an implicit role swap without the use of the ***Swap*** message (see Section 8.3.2.5) where the Consumer/Provider is now operating as the Source while the Provider/Consumer operates as a Sink. With communications and cable capabilities established, the Provider/Consumer may negotiate for a voltage/current combination to charge its battery or to operate.



Provider/Consumer
V_{BUS} within vSafe0V
Dead Battery (Not powering V_{BUS})
Willing to power port?
Yes
Operate as a Source
No
Wants to be powered?
No
Yes
V_{BUS} > vSafe0V?
No
Yes
Output Bit-Stream within tSendBitStream of V_{BUS} detection Start WaitForPowerTimer
WaitForPowerTimer timeout
Protocol Engine Ready for Capabilities?
No
Yes
Stop Outputting Bit-Stream
Capabilities Message Received
No
Yes
Determine Cable Type
Operate as a Sink
Consumer/Provider
PSSourceOffTimer timeout
Start DBDetectTimer
Apply vSafe0V to V_{BUS} Start PSSourceOffTimer
V_{BUS} > vSafe0V?
Yes
Operate as a Sink
No
No
DBDetectTimer expired?
Yes
Output vSafeDB on V_{BUS}
Listen for Bit-Stream for tBitStreamDetect
Bit-Stream detected?
No
Yes
Output vSafe5V on V_{BUS} Start DeviceReadyTimer
DeviceReadyTimer expired?
Yes
No
Yes
Bit-Stream detected?
No
Operate as a Source

Figure 4-1 Dead Battery / Unpowered Port Detection Flow

Figure 4-1 illustrates the flow for Dead Battery/Unpowered Port detection for both Provider/Consumer ports and Consumer/Provider ports. To ensure consistent behavior all Consumer/Provider ports shall have the ability to detect a Provider/Consumer with a Dead Battery or Unpowered Port.

The Consumer/Provider may use other means to detect the presence of a Port Partner before applying the method described in the following paragraph.

In operation, a Consumer/Provider port that does not detect ***vSafe5V*** on V_{BUS} shall periodically apply a current limited five volt supply (***vSafeDB***) to V_{BUS} in an attempt to ascertain the presence of a Provider/Consumer port that wants to be powered. In response to ***vSafeDB*** on V_{BUS}, a Provider/Consumer port that wants power shall transmit the Bit Stream that a Consumer/Provider can detect indicating the presence of a port that wants to be powered. When the Consumer/Provider detects a Provider/Consumer port wanting to be powered, it shall apply ***vSafe5V*** to V_{BUS}. In all other cases, for example when connected to a legacy port or an unpowered Provider port, there will be no response and the Consumer/Provider port shall not apply full power (***vSafe5V***) to V_{BUS}. The Consumer/Provider shall continue to periodically probe for the presence of a Provider/Consumer port that wants power. The limitations on the current and power applied to back power V_{BUS} and its controlled duration shall be applied in order to prevent damage to legacy ports.

4.2 Operational Details for the Provider/Consumer

The Provider/Consumer with a dead battery shall begin (as it must if its battery is truly dead) by outputting nothing on V_{BUS}. When power is applied to V_{BUS} (e.g. back powered), V_{BUS} is used by the Provider/Consumer to power its transmitter and to output the Bit Stream within the limited power offered by ***vSafeDB***.

The Provider/Consumer shall begin outputting the Bit Stream on V_{BUS} within ***tSendBitStream***. This is necessary because the application of full V_{BUS} power is dependent on the time it takes the Consumer/Provider to detect the presence of activity on V_{BUS} and to adjust its power supply.

The Provider/Consumer may have additional circuitry that requires more power than is available from V_{BUS} when powered by ***vSafeDB***. The Provider/Consumer port can assume that full ***vSafe5V*** power is available ***tWaitForPower*** after it detected the presence of greater than zero volts on V_{BUS}. After ***tWaitForPower***, ***vSafe5V*** will be available to the Provider/Consumer to bring up any remaining logic required to process PD messages on V_{BUS}. When this logic is ready, the Provider/Consumer port shall signal its Port Partner that is ready to operate as Consumer by stopping the Bit Stream.

In summary the Provider/Consumer that requires power shall:

1) Output vSafe0V on V_{BUS}.
2) When a voltage greater than vSafe0V is detected on V_{BUS}, start sending the Bit Stream within tSendBitStream using vSafeDB to power its PD interface.
3) After sending the Bit Stream for tWaitForPower, assume vSafe5V is present on V_{BUS}. (Note there is no positive handshake for this transition.)
4) Use vSafe5V to bring up its Protocol and Policy engines and when it's ready to receive and process Capabilities messages stop sending the Bit Stream.
5) Operate as Consumer awaiting receipt of a ***Capabilities*** message.

There is an implicit contract between the Consumer/Provider and Provider/Consumer based on the time between application of ***vSafeDB*** to V_{BUS}, the beginning of the Bit Stream on V_{BUS} and the time the Consumer/Provider takes to recognize the Bit Stream and apply ***vSafe5V***. These times are intentionally long to provide a good deal of margin.

The Consumer/Provider, when it detects V_{BUS} is at ***vSafe0V*** for ***tDBDetect***, shall:

1) Output ***vSafeDB***
2) Detect at least 128 alternating '0s' and '1s' from the Bit Stream within ***tBitStreamDetect***.
    a) If the Bit Stream is not detected within ***tBitStreamDetect,*** remove ***vSafeDB***from V_{BUS}, wait ***tDBDetect*** and go to step 1.
    b) Output ***vSafe5V*** and start DeviceReadyTimer.
3) While the Bit Stream is present
    a) ***tDeviceReady*** not exceeded, go to 4.
    b) ***tDeviceReady*** exceeded, remove ***vSafe5V*** from V_{BUS}, wait ***tDBDetect*** and go to step 1.
4) Send a ***Capabilities*** message and operate as a Provider.

4.3 Cable IR Ground Drop

Every PD Sink Port capable of USB communications should have the capability to detect excessive IR drop (e.g. greater than ***V_{GND_DROP}***) on the Ground between itself and the Source. This ensures that the voltage drop across ground does not fall out of the acceptable common mode range for the USB Hi-Speed transceivers data lines due to excessive current draw. If this occurs, unreliable USB communication may result.

Sink ports with data lines drawing more than 1.5A shall have the capability to detect IR drop greater than ***vIRDrop_Cable***.

An estimate of ***V_{GND_DROP}*** can be made by a Sink, by:

- Assuming a compliant cable that meets ***vIRDrop_Cable.***
- Measuring V_{BUS} while operating.

If the measured value is less than 1V below the negotiated voltage with the Source, then ***V_{GND_DROP}*** will be within acceptable limits. Otherwise, the Sink shall send a ***CableVGO*** message and attempt to reduce its operating current.

Other methods of performing this measurement may be employed.

4.4 Cable Type Detection

4.4.1 Plug Type Detection

Non-compliant cables, such as 'Y' and 'W' cables create the potential to damage hardware were the PD system to allow more than ***vSafe5V*** to be placed on V_{BUS}. To prevent this, all PD plug assemblies are made in a way that can be electronically detected. Only cables that are marked for PD shall be used for voltages higher than ***vSafe5V*** or current levels higher than 1.5A.

Providers and Provider/Consumers shall detect the type of attached cable and either limit the Capabilities they offer or operate in a low power mode based on:

- The receptacle type (e.g. Standard-A or Micro-AB) and current carrying capability.
- The A-plug type (e.g. Standard or PD or Low Power).

Consumers and Consumer/Providers shall detect the type of attached cable and limit their requests based on:

- The receptacle type (e.g. Standard-B or Micro-AB) and current carrying capability.
- The detected B-plug type (e.g. Standard or PD).
- The current carrying capability indicated by the PD B-plug.

4.4.2 Plug Insertion Detect

The USB Power Delivery specification defines an Insertion Detect mechanism for the Standard-A plug. It consists of a contact that connects with the shield when a Standard-A plug or a PD Standard-A plug is inserted into the receptacle. The shield is essentially connected to ground through at most a 1KΩ resistance.

The Micro-A plug's ID pin is used for Insertion Detect. It is essentially connected to ground through at most a 1KΩ resistance. This is the default defined in the Micro-A connector specification for ***[USB2.0]*** and ***[USB3.0]***. See Section 3.4.2 for more details.

The Insertion Detect should be used as follows to:

- Indicate when to apply power to V_{BUS} when a plug is present for cold socket applications.
- Indicate to the PD logic to start sending ***Capabilities*** messages when a plug is present.
- Indicate to the PD logic to put V_{BUS} back to the default state when the plug is removed.

The Insertion Detect feature for Standard-A receptacles shall be present for cold socket but is optional for all other Standard-A applications.

Cable type detection is a multi-step process that both the Provider and Consumer perform. This section provides flow of the cable type detection based on the electronic markings defined in Section 3.4. The Source shall limit maximum capabilities it offers so as not to exceed to the capabilities of the type of A-plug detected. Requests made by the Sink shall not exceed the capabilities of the type of B plug detected.

The cable detection process is usually run when the Source is powered up or when power is applied to a Sink. The exact method used to detect these events is up to the manufacturer and shall meet the following requirements:

- Sources shall run the cable detection process prior to the Source sending ***Capabilities*** messages offering voltages in excess of ***vSafe5V*** or currents in excess of 1.5A.
- Sinks shall run the cable detection process prior to making a request for power.
- Provider/Consumers with dead batteries shall wait until after receipt of the first ***Capabilities*** message before running the cable detection process and making a request for power.

4.4.3 Plug Type Determination

Figure 4-2 shows the flow for the first portion of the multi-step process that shall be used to determine the kind of cable. It begins by determining the kind of plug inserted into the receptacle. Specifically the plug type: Standard-A, Standard-B, Micro-A or Micro-B, is determined.

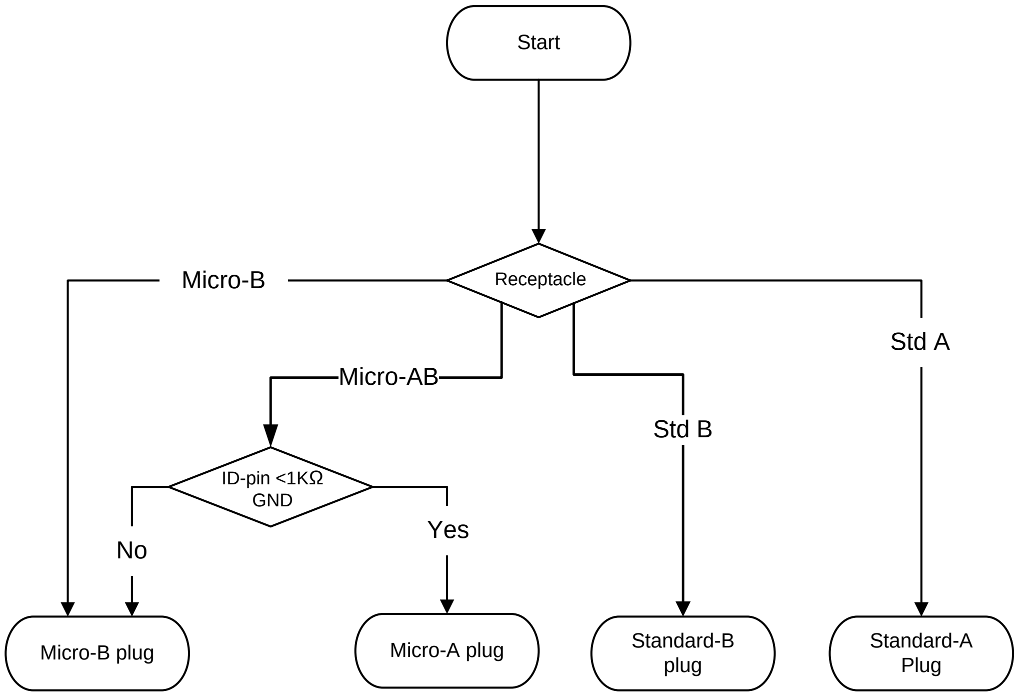


Figure 4-2 Plug Type Determination

4.4.4 Detecting the PD Capabilities of the Standard-A Connector

The PD version of the Standard-A receptacle has one or two additional contacts that are used to:

- Optional detect if Standard-A plug is inserted in or removed from the receptacle.
- Required detect if the Standard-A plug is PD Capable or not.

The following description assumes that the two switches in the Standard-A receptacle are detected by having pull-up resistors to a positive voltage and looking at the voltage, so a high voltage indicates no connection in the switch and a low voltage indicates a connection in the switch. Other detection circuits may be used.

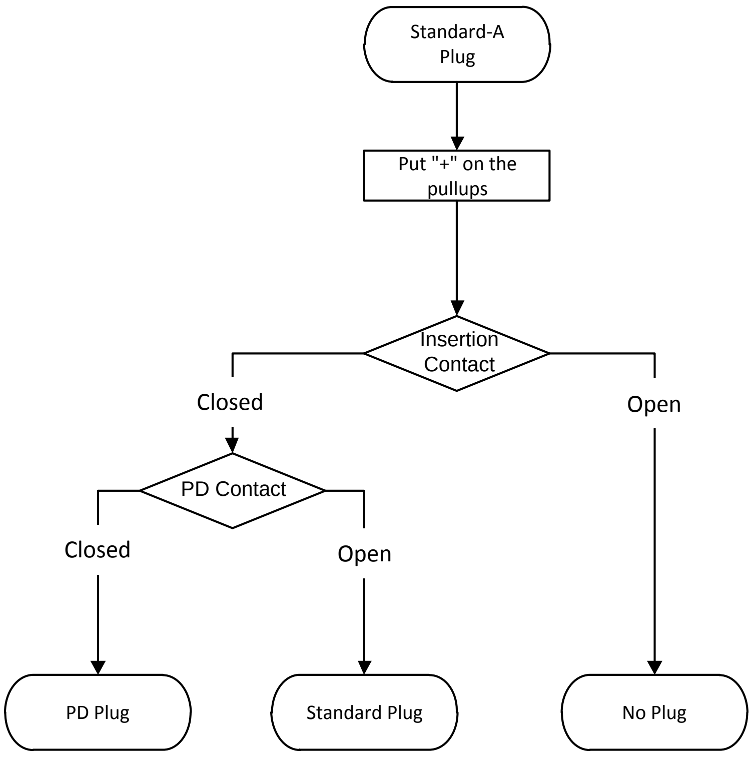


Figure 4-3 Standard-A Plug PD Capabilities Flow

Figure 4-3 illustrates how the detection contacts in the A receptacle shall be used. Insertion Detect is an optional feature (see Section 3.1.5). When not present, the path through the Figure 4-3 follows the Insertion Detect 'Closed' arc.

4.4.5 Plug Type Detection except Standard-A

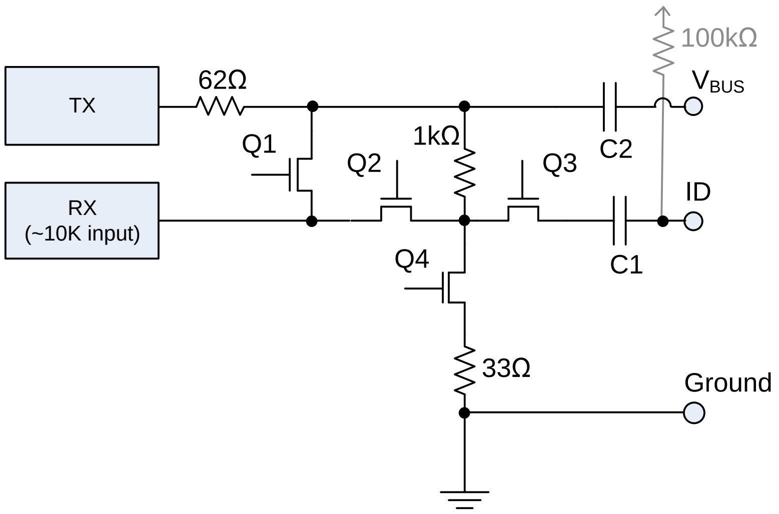


Figure 4-4 Plug Type Detection Circuit

The example circuit shown in Figure 4-4 is used to detect the electronic markings on the ID pin indicating the type of Micro connector.

The TX is used to put a carrier signal on the V_{BUS} line and the RX is used to detect whether a signal is present or not (typically the Squelch is used for this purpose). Q1 - Q4 are used to create a series of circuits where the voltage output (as measured by the RX) in each step is used to determine the configuration of the plug in accordance with to Table 4-1.

In normal operation Q1 is conducting (turned ON) and Q2, Q3 and Q4 are not conducting (turned OFF).

In order to check the plug type, a series of steps shall be performed; the result of each step is recorded as a "0" or "1". The steps are:

1) Q1, Q3 and Q4 not conducting (turned OFF)
2) Q2 conducting (turned ON)
3) Check Squelch -> open - "1", else "0" -> bit 1
4) Q3 conducting (turned ON)
5) Check Squelch -> open - "1", else "0" -> bit 2
6) Q4 conducting (turned ON)
7) Check Squelch -> open - "1", else "0" -> bit 3

Table 4-1 summarizes the results.

Table 4-1 Plug Type Determination

bit 1	bit 2	bit 3	Micro-A plug	Micro-B or PD Standard-B plug	Approximate level at RX when detecting bit 1	Approximate level at RX when detecting bit 2	Approximate level at RX when detecting bit 3
1	1	1	Low Power	PD (5A)	~ 0dB	~ 0dB	~ -9dB
1	1	0	PD	Legacy	~ 0dB	~ -6dB (PD) ~ 0dB (Legacy)	~ -30dB
1	0	1	Fault	Fault			
1	0	0	Legacy	PD (3A)	~ 0dB	~ -40dB	~ -40dB
0	1	1	Fault	Fault			
0	1	0	Fault	Fault			
0	0	1	Fault	Fault			
0	0	0	Fault	Fault			

4.5 Low Power Devices

Sources powered by a single cell Li battery that want to minimize the power they output over V_{BUS} may take advantage of the Low Power Device feature without the requirement to fully support the PD negotiation process. Low Power (Sink) devices shall be able to operate normally when powered by any voltage between 2.5V and ***vSafe5V***.

The entry process into low power operation is as follows:

1) The Source detects a Low Power device is attached (see Table 4-1)
2) The Source is then allowed to supply any voltage between 2.5V and vSafe5V, typically it directly connects the battery to V_{BUS} avoiding any power conversion losses.
3) The Source may monitor the voltage and when it falls below 2.5V reapply ***vSafe5V*** in an attempt to continue operating. Alternatively, the Source may not monitor V_{BUS} and when it falls to this level, both it and the Low Power device likely fail for lack of power.

4.6 Electrical Parameters

Table 4-2 shows the parameters used in this section.

Table 4-2 Electrical Parameters

	Minimum	Nominal	Maximum	Units	Section
tSendBitStream			0.2	s	8.3.3.6.7
tBitStreamDetect			0.3	s	8.3.3.6.7
tWaitForPower	10			ms	8.3.3.6.7
tDeviceReady	60		90	s	8.3.3.6.7
tDBDetect		10	15	s	8.3.3.6.7

Table 4-3 Electrical Timers

Timer	Parameter	Used By	Section
BitStreamDetectTimer	***tBitStreamDetect***	PolicyEngine	8.3.3.6.7
WaitForPowerTimer	***tWaitForPower***	Policy Engine	8.3.3.6.7
DeviceReadyTimer	***tDeviceReady***	PolicyEngine	8.3.3.6.7
DBDetectTimer	***tDBDetect***	PolicyEngine	8.3.3.6.7

5. Physical Layer

5.1 Physical Layer Overview

The Physical Layer (PHY) defines the signaling technology for USB Power Delivery. This chapter defines the electrical requirements and parameters of the PD Physical Layer required for interoperability between USB PD Devices.

5.2 Physical Layer Functions

The USB PD Physical Layer consists of a pair of transmitters and receivers that communicate across a single signal wire, V_{BUS}. All communication is half duplex. The PHY shall practice collision avoidance to minimize communication errors on the channel.

The transmitter performs the following functions:

- Receive packet data from the protocol layer
- Calculate and append a CRC
- Encode the packet data including the CRC (i.e. the payload)
- Transmit the bit stream (preamble, ***SOP***, payload, CRC and ***EOP***) across the channel using a Frequency Shift Keying (FSK) modulated carrier

The receiver performs the following functions:

- Detect the FSK modulated carrier from the channel, recover the clock and lock onto the bit stream from the preamble
- Detect the ***SOP***
- Decode the received data including the CRC
- Detect the ***EOP*** and validate the CRC
  - If the CRC is valid, deliver the packet data to the protocol layer.
  - If the CRC is not valid, flush the received data.

The PHY functions are shown in Figure 5-1, Figure 5-2, and Figure 5-3 The PHY is expected to keep power consumption low, especially when only the squelch detector is required to be active. In the active mode, where any of the functions listed above may be executed, the Phy Block power consumption should be minimized. In the squelch mode, when only the squelch detector is required, the power consumption should be minimized.

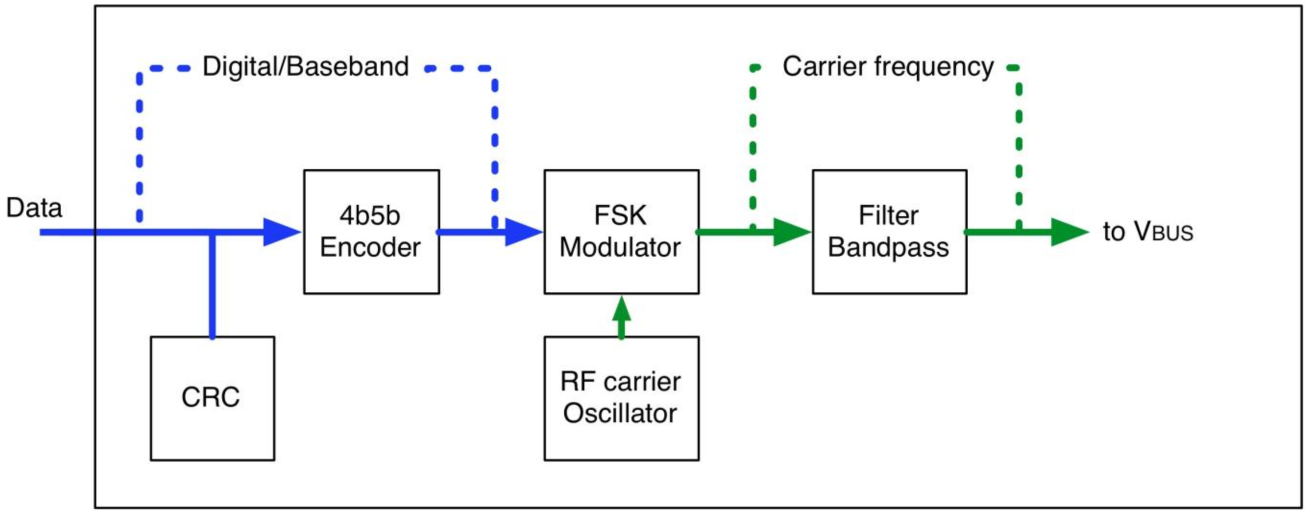


Figure 5-1 Transmitter Block Diagram

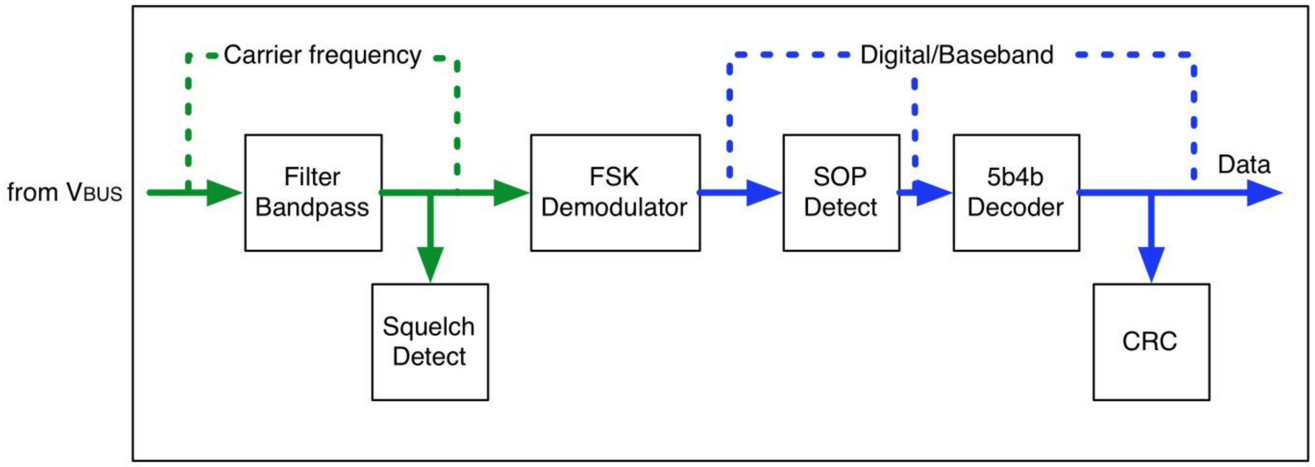


Figure 5-2 Receiver Block Diagram

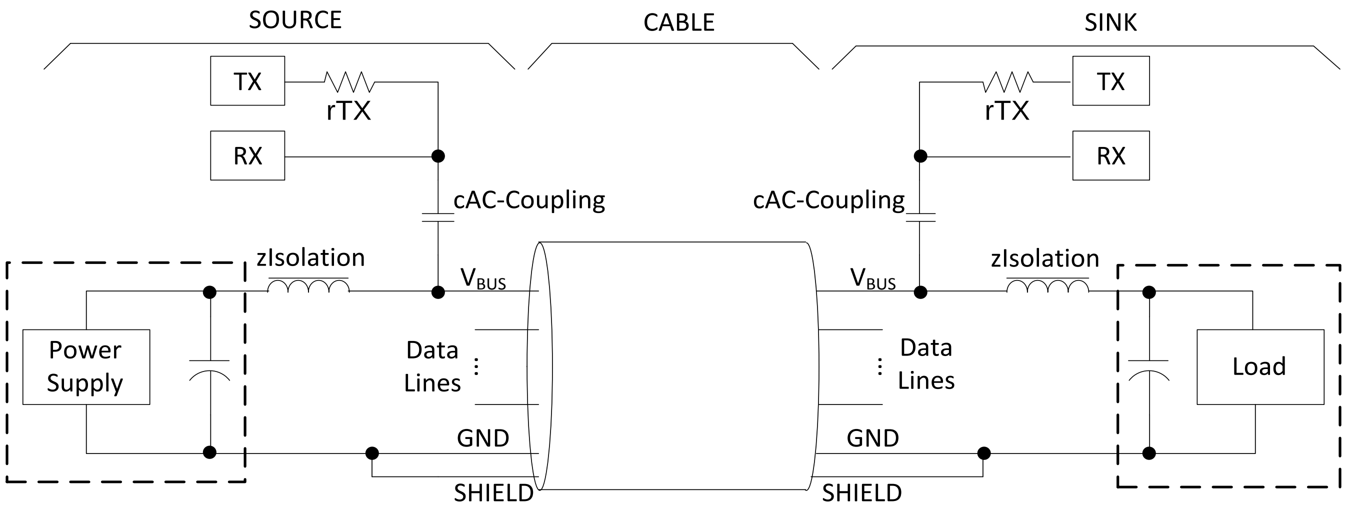


Figure 5-3 Channel Diagram (Cable Type Detection not shown)

5.2.1 Channel Overview

The channel connects two PHYs together via a single-ended serial signal. The PD signal uses USB V_{BUS} as the channel. The PHYs shall be AC coupled through a capacitor ***cAC-Coupling*** at each end. A resistor, ***rTX,*** separates the transmitter and V_{BUS}. While transmitting, the Source, or Sink, shall apply an AC signal with amplitude ***vTX*** to V_{BUS} as measured between ***rTX*** and ***cAC-Coupling***.

5.2.2 Transceiver isolation Impedance

The Source and Sink shall place an isolation impedance between the V_{BUS} wire bulk capacitance and the V_{BUS} pin on the connector to allow the AC coupled USB Power Delivery transceiver to communicate over V_{BUS}.

The isolation impedance shall have a reactance of ***zIsolation*** at any frequency in the band from ***fLow*** to ***fHigh.*** See Sections 7.1.8 and 7.2.6 for additional detail.

5.2.3 Transceiver AC Coupling Capacitance

The Source and Sink shall place an AC coupling capacitance ***cAC-Coupling*** between the V_{BUS} pin on the connector and the transceiver as indicated in Figure 5-3.

5.3 Bit Encoding

PD uses a carrier of ***fCarrier*** modulated with the information to avoid the noise from the power supplies. Frequency Shift Keying (FSK) shall be used to encode bits for transmission on the channel. A signal of amplitude ***vTX*** shall be injected onto V_{BUS} using a carrier frequency, ***fCarrier***. The following logic states shall be used:

- Logic 0 is indicated by a frequency ***fCarrier*** - ***fDeviation***.
- Logic 1 is indicated by a frequency ***fCarrier*** + ***fDeviation***.

5.4 Symbol Encoding

Except for the preamble, all communications on the line shall be encoded with a line code to ensure a reasonable level of DC-balance and a suitable number of transitions. This encoding makes receiver design less complicated and allows for more variations in the receiver design.

4b5b line code shall be used. This encodes 4-bit data to 5-bit symbols for transmission and decodes 5-bit symbols to 4-bit data for consumption by the receiver.

The 4b5b code provides data encoding along with special symbols. Special symbols are used to signal ***Hard Reset***, and delineate packet boundaries.

Table 5-1 4b5b Symbol Encoding Table

Name	4b	5b Symbol	Description
0	0000	11110	hex data 0
1	0001	01001	hex data 1
2	0010	10100	hex data 2
3	0011	10101	hex data 3
4	0100	01010	hex data 4
5	0101	01011	hex data 5
6	0110	01110	hex data 6
7	0111	01111	hex data 7
8	1000	10010	hex data 8
9	1001	10011	hex data 9
A	1010	10110	hex data A
B	1011	10111	hex data B
C	1100	11010	hex data C
D	1101	11011	hex data D
E	1110	11100	hex data E
F	1111	11101	hex data F
Sync-1	K-code	11000	Startsynch #1
Sync-2	K-code	10001	Startsynch #2
RST-1	K-code	00111	Hard Reset #1
RST-2	K-code	11001	Hard Reset #2
EOP	K-code	01101	EOP End Of Packet
Reserved	Error	00000	Do Not Use

Name	4b	5b Symbol	Description
Reserved	Error	00001	Do Not Use
Reserved	Error	00010	Do Not Use
Reserved	Error	00011	Do Not Use
Reserved	Error	00100	Do Not Use
Reserved	Error	00101	Do Not Use
Reserved	Error	00110	Do Not Use
Reserved	Error	01000	Do Not Use
Reserved	Error	01100	Do Not Use
Reserved	Error	10000	Do Not Use
Reserved	Error	11111	Do Not Use

5.5 Ordered sets

Ordered sets shall be interpreted according to Figure 5-4.

An ordered set consists of 4 K-codes sent as shown below.

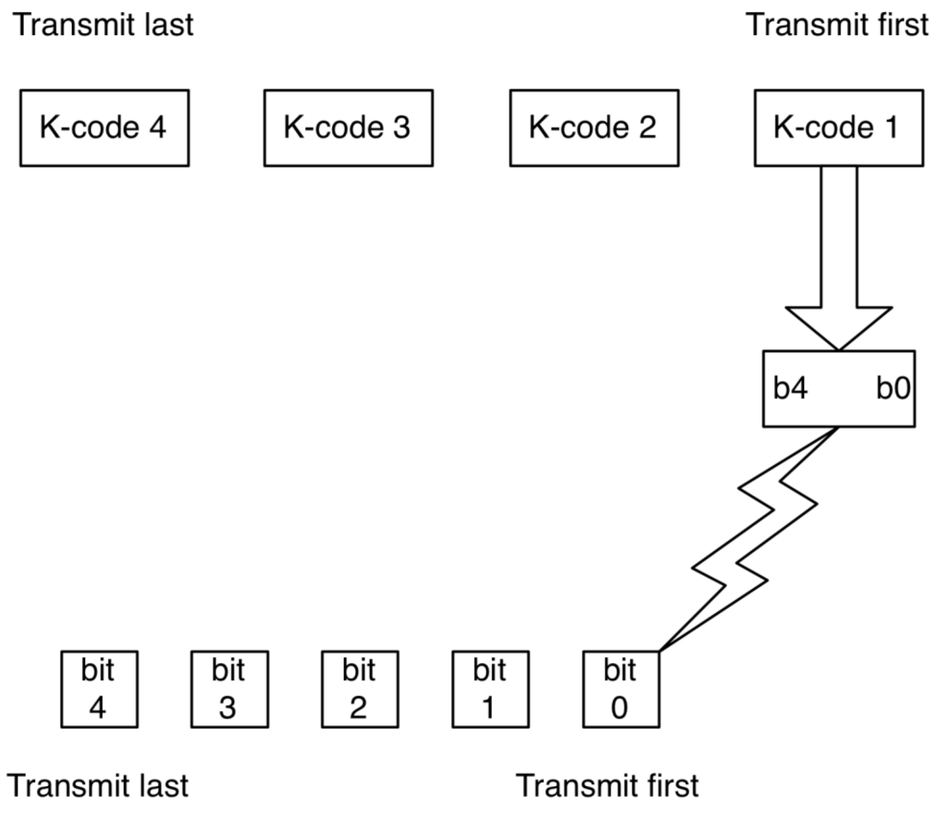


Figure 5-4 Interpretation of ordered sets

A list of the ordered sets used by USB Power Delivery can be seen in Table 5-2.

Table 5-2 Ordered Sets

Ordered Set	Section
SOP	5.7.1.2
Hard Reset	5.7.4

5.6 Transmitted Bit Ordering

This section describes the order of bits on the wire that shall be used when transmitting data of varying sizes. Table 5-3 shows the different data sizes that are possible. Figure 5-5 shows the transmission order that shall be followed.

Table 5-3 Data Size

	Unencoded	Encoded
Byte	8-bits	10-bits
Word	16-bits	20- bits
DWord	32-bits	40-bits

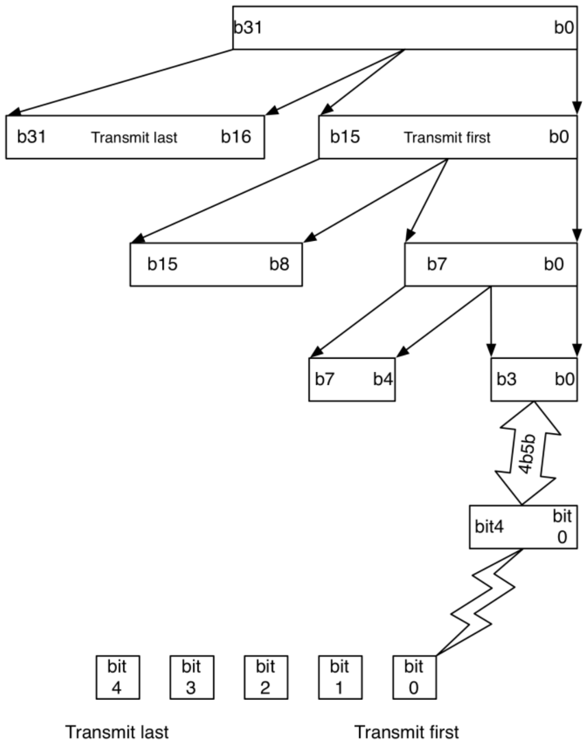


Figure 5-5 Transmit Order for Various Sizes of Data

5.7 Packet Format

The packet format shall consist of a preamble, an ***SOP***, (see Section 5.7.1.2), packet data including header, a CRC and an ***EOP*** (see Section 5.7.1.5). The packet format is shown in Figure 5-6.

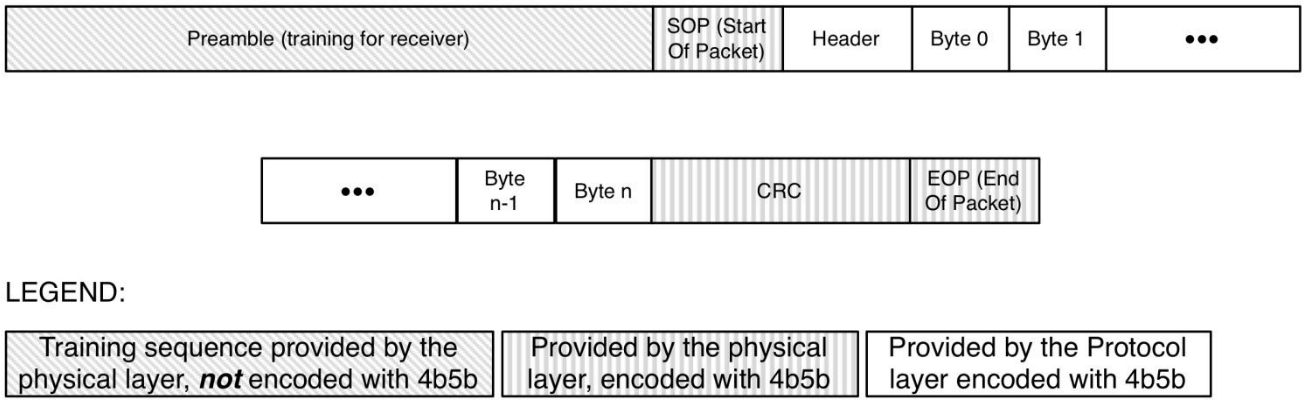

Preamble (training for receiver)
SOP (Start Of Packet)
Header
Byte 0
Byte 1
•••
•••
Byte n-1
Byte n
CRC
EOP (End Of Packet)
LEGEND:
Training sequence provided by the physical layer, not encoded with 4b5b
Provided by the physical layer, encoded with 4b5b
Provided by the Protocol layer encoded with 4b5b

Figure 5-6 USB Power Delivery Packet Format

5.7.1 Packet Framing

The transmission shall start with a preamble that is used to allow the receiver to lock onto the carrier. It shall be followed by a ***SOP*** (Start of Packet). The packet shall be terminated with an ***EOP*** (End of Packet) K-code.

5.7.1.1 Preamble

The preamble shall be used to achieve lock in the receiver by presenting an alternating series of "0s" and "1s", so the average frequency is the carrier frequency. Unlike the rest of the packet, the preamble shall not be encoded.

The preamble shall consist of a 64-bit sequence of alternating 0s and 1s. The preamble shall start with a "0" and shall end with a "1".

5.7.1.2 Start of Packet Sequence (SOP)

SOP is an ordered set. The ***SOP*** ordered set is defined as: three ***Sync-1*** K-codes followed by one ***Sync-2*** K-code.

Table 5-4 SOP ordered set

K-code number	K-code in code table
1	***Sync-1***
2	***Sync-1***
3	***Sync-1***
4	***Sync-2***

Table 5-5 Validation of SOP sequence

	1st code	2nd code	3rd code	4th code
Valid	Error	***Sync-1***	***Sync-1***	***Sync-2***
Valid	***Sync-1***	Error	***Sync-1***	***Sync-2***
Valid	***Sync-1***	***Sync-1***	Error	***Sync-2***
Valid	***Sync-1***	***Sync-1***	***Sync-1***	Error
Valid (perfect)	***Sync-1***	***Sync-1***	***Sync-1***	***Sync-2***

	1st code	2nd code	3rd code	4th code
Not Valid (example)	***Sync-1***	Error	***Sync-1***	Error

The receiver shall search for all four K-codes and when it finds at least three in the correct order, it shall interpret it as a valid start of packet.

5.7.1.3 Packet Payload

The packet payload is delivered from the protocol layer (Section 6.2) and shall be encoded with the hex data codes from Table 5-1.

5.7.1.4 CRC

The CRC shall be inserted just after the payload. It is described in Section 5.7.2.

5.7.1.5 End of Packet (EOP)

The end of packet marker shall be a single ***EOP*** K-code as defined inTable 5-1. This shall mark the end of the CRC. After the ***EOP***, the CRC-residual shall be checked. If the CRC is not good, the whole transmission shall be discarded, if it is good, the packet shall be delivered to the Protocol Layer.

5.7.2 CRC

The packet header and data shall be protected by a 32-bit CRC.

CRC-32 protects the data integrity of the data payload. CRC-32 is defined as follows:

- The CRC-32 polynomial shall be = 04C1 1DB7h.
- The CRC-32 Initial value shall be = FFFF FFFFh.
- CRC-32 shall be calculated for all bytes of the payload not inclusive of any packet framing symbols (i.e. excludes the preamble, ***SOP***, ***EOP***).
- CRC-32 calculation shall begin at byte 0 bit 0 and continue to bit 7 of each of the bytes of the packet.
- The remainder of CRC-32 shall be complemented.
- The residual of CRC-32 shall be C704 DD7Bh.

Note: This inversion of the CRC-32 remainder adds an offset of FFFF FFFFh that will create a constant CRC-32 residual of C704 DD7Bh at the receiver side.

Note: The CRC implementation is identical to the one used in ***[USB3.0]***.

Figure 5-7 is an illustration of CRC-32 generation. The output bit ordering is listed in Table 5-6.

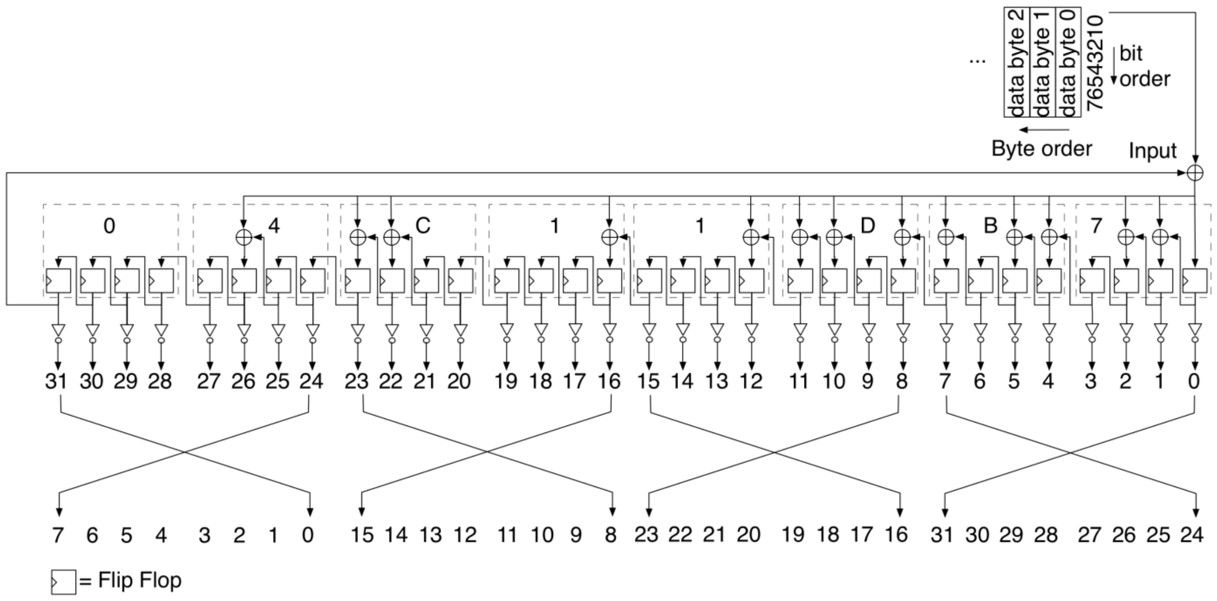


Figure 5-7 CRC 32 generation

Table 5-6 CRC-32 Mapping

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

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

The CRC-32 shall be encoded before transmission.

5.7.3 Error reporting

CRC errors, or errors detected while decoding encoded symbols using the code table, shall be treated the same way; the message shall be discarded and a ***GoodCRC*** message shall not be returned.

5.7.4 Hard Reset

Hard Reset signaling is an ordered set of bytes sent with the purpose to be recognized as an embedded ***Control Message*** to the PHY-layer. The ***Hard Reset*** signaling ordered set is defined as: three ***RST-1*** K-codes followed by one ***RST-2*** K-code.

Table 5-7 Hard Reset ordered set

K-code number	K-code in code table
1	***RST-1***
2	***RST-1***
3	***RST-1***
4	***RST-2***

A Device shall perform a Hard Reset when it receives ***Hard Reset*** signaling. After receiving the ***Hard Reset*** signaling, the device shall reset as described in the Protocol Layer, see Section 6.7.2.

Table 5-8 Validation of Hard Reset

	1st code	2nd code	3rd code	4th code
Valid, reset	Error	***RST-1***	***RST-1***	***RST-2***
Valid, reset	***RST-1***	Error	***RST-1***	***RST-2***
Valid, reset	***RST-1***	***RST-1***	Error	***RST-2***
Valid, reset	***RST-1***	***RST-1***	***RST-1***	Error
Valid (perfect) , reset	***RST-1***	***RST-1***	***RST-1***	***RST-2***
Not Valid, NO reset(example)	***RST-1***	Error	***RST-1***	Error

The procedure for sending ***Hard Reset*** signaling shall be as follows:

- If the PHY is currently sending a message, the message shall be interrupted by sending an ***EOP*** K-code.
- If V_{BUS} is not idle, wait for it to become idle then discard the message.
- Wait ***tBusIdle***
- Send the Preamble followed by the 4 K-codes for ***Hard Reset*** signaling.
- Reset the PHY-layer.

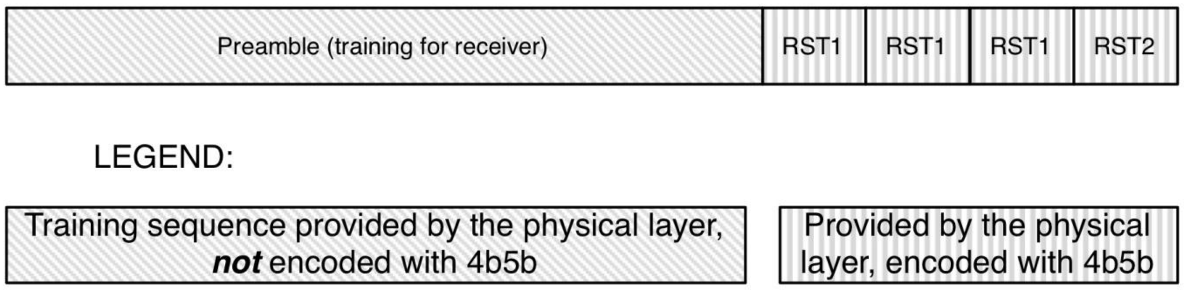

Preamble (training for receiver)
RST1
RST1
RST1
RST2
LEGEND:
Training sequence provided by the physical layer, *not* encoded with 4b5b
Provided by the physical layer, encoded with 4b5b

Figure 5-8 Line format of Hard Reset

5.7.5 Bit Stream

A Bit Stream transmission is defined to allow a Consumer/Provider to detect a Provider/Consumer with a dead battery (see Section 4.1).

The transmitter of the Provider/Consumer that implements Dead Battery Support shall be able to transmit a Bit Stream consisting of alternating “0s” and “1s” which may be viewed as concatenating multiple preambles as shown in Figure 5-9 (note that the last preamble may not contain 64 bits). The minimum duration of this Bit Stream transmission shall be ***tSendBitStream***. The PHY shall continue to transmit the Bit Stream until it receives a stop signal (see Figure 4-1).

The Consumer/Provider’s receiver shall declare a Bit Stream is detected after detecting 128 consecutive bits that match the preamble pattern of alternating “0s” and “1s”.

After the Bit-Stream is detected, the receiver shall indicate that the Bit-Stream has stopped when the squelch has closed (the signal level is below ***vSquelchOperating***) for ***tBitStreamComplete***.

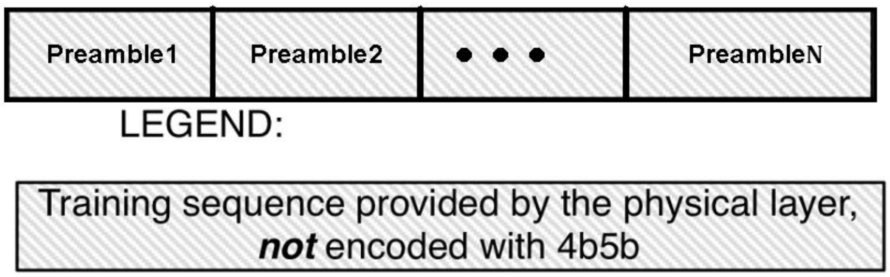

Preamble1
Preamble2
● ● ●
PreambleN
LEGEND:
Training sequence provided by the physical layer, *not* encoded with 4b5b

Figure 5-9 Line Format of Bit Stream

5.8 Collision Avoidance

The PHY shall monitor the channel for data transmission and only initiate transmissions when V_{BUS} is idle. If the squelch is not broken, it shall be considered safe to start a transmission. The squelch shall be checked immediately prior to transmission.

5.9 Common Specifications

This section defines the common receiver and transmitter requirements that shall be followed.

5.9.1 Common Parameters

The electrical requirements specified in Table 5-9 shall apply to both the transmitter and receiver.

Table 5-9 Common Normative Requirements

Parameter	Description	Min	Nom	Max	Units	Comment
fCarrier	FSK carrier frequency	22.4	23.2	24.0	MHz	
fDeviation	FSK frequency deviation	450	500	600	kHz	
fBitRate	Bit rate	270	300	330	Kbps	
cAC-Coupling	AC coupling capacitance	3		10	nF	

Table 5-10 Transceiver Isolation Impedance Normative Requirements

Parameter	Description	Min	Nom	Max	Units	Comment
zIsolation	Impedance Allowed	80			Ω	
fLow	Lowest frequency used in communication			*fCarrier* – 2MHz	MHz	The 2MHz is the space the signal takes including FM tolerance and deviation.
fHigh	Highest frequency used in communication	*fCarrier* + 2MHz			MHz	

One example implementation of a transceiver isolation impedance using the parameters listed in Table 5-10 is a 1µH inductor (see Appendix C).

5.10 Transmitter Specifications

5.10.1 Transmitter Requirements

Table 5-11 Transmitter Requirements

Name	Description	Min	Nom	Max	Units	Comment
vTX	TX voltage injected on VBUS	100	150	200	mVRMS	This is the voltage on V_{BUS} (the link terminated by ***rTX***.).
rTX	The TX output impedance and the cable impedance for test and calculation.	52	62	72	Ω	
pCarrierNoise	Transmit carrier FM noise at >20KHz from carrier			-48	dBc / Hz	Measured by transmitting continuous 1's or 0's.
pBitRate	Difference between minimum and maximum bitrate during the entire packet including preamble.			0.25	%	This shall be considered as a percentage of the average bit rate over the packet.
pCarrierFreq	Difference between minimum and maximum carrier frequency during the entire packet including preamble.			0.1	%	This shall be considered as a percentage of the average carrier frequency over the packet.
pDevFreq	Difference between minimum and maximum deviation frequency during the entire packet including preamble.			1	%	
tBusIdle	Transceiver turn around time, do not transmit during this time after receiving the last bit of a message.	20			µs	This time is available to reconfigure the system for reception after transmitting

5.10.1.1 Carrier Noise

The phase noise measured more than 20kHz away from the carrier shall not exceed ***pCarrierNoise***. The phase noise is measured while the transmitter sends a ***BIST Carrier Mode 2*** signal.

5.10.1.2 Bit-Rate Drift

The change in ***fBitRate*** during a packet shall be less than ***pBitRate*** to ensure that ***fBitRate*** does not drift too much during a packet. The parameter ***pBitRate*** is defined as follows assuming that the transmitted bits are alternating "0s" and "1s". The average bit rate of any group of eight bits shall be within ***pBitRate*** of the average bit rate of the first 64 bits. This may be expressed mathematically as

$$\frac{1}{\bar{T}}\left|\bar{T}-\frac{1}{8}\sum_{i=1}^{8}\hat{T}((k-1)\cdot 8+i)\right| < \textit{pBitRate}$$

where $\hat{T}(i)$ = duration of the i–th bit, $\bar{T}=\frac{1}{64}\sum_{i=1}^{64}\hat{T}(i)$ is the average bit duration of the first 64 bits, where k indexes the groups of 8 bits. In practice, $\bar{T}$ equals ***fBitRate***. For a packet of maximum duration it will be true that $k \le 59$.

The transmitter shall have the same ***pBitRate*** for non-BIST packets and BIST packets.

5.10.1.3 Carrier-Frequency Drift

The change in ***fCarrier*** during a packet shall be less than ***pCarrierFreq*** to ensure that ***fCarrier*** does not drift too much during a packet. The parameter ***pCarrierFreq*** is defined as follows assuming that the transmitted bits are alternating "0s" and "1s". The average frequency of any group of eight bits shall be within ***pCarrierFreq*** of the average frequency of the first 64 bits. This may be expressed mathematically as

$$\frac{1}{\bar{f}_C}\left|\bar{f}_C-\frac{1}{n}\sum_{i=1}^{n}\hat{f}((k-1)\cdot n+i)\right| < \textit{pCarrierFreq}$$

where $\hat{f}(i)$ is the frequency at the i-th sample, where $\bar{f}_C=\frac{1}{N}\sum_{i=1}^{N}\hat{f}(i)$ is the average frequency of the first 64bits, where k indexes the groups of 8 bits, n is the number of samples in 8 bits, and N is the number of samples in 64 bits. In practice, $\bar{f}_C$ equals ***fCarrier***. For a packet maximum it will be true that duration, $k \le 59$.

The transmitter shall have the same ***pCarrierFreq*** for non-BIST packets and BIST packets.

5.10.1.4 Deviation-Frequency Drift

The change in ***fDeviation*** shall be less than ***pDevFreq*** to ensure that ***fDeviation*** does not drift too much during a packet. The parameter ***pDevFreq*** is defined as follows assuing that the transmitter bits are alternating "0s" and "1s". The average deviation frequency during any group of eight bits shall be within ***pDevFreq*** of the average deviation frequency of the first 64 bits. This may be expressed mathematically as

$$\frac{1}{\bar{f}_M}\left|\bar{f}_M-\frac{1}{n}\sum_{i=1}^{n}\left|\hat{f}((k-1)\cdot n+i)-\bar{f}_C\right|\right| < \textit{pDevFreq}$$

where $\hat{f}(i)$ is the frequency at the i-th sample, where $\bar{f}_M=\frac{1}{N}\sum_{i=1}^{N}\left|\hat{f}(i)-\bar{f}_C\right|$ is the average deviation frequency of the first 64 bits, where $\bar{f}_C=\frac{1}{N}\sum_{i=1}^{N}\hat{f}(i)$ is the average frequency of the first 64bits, where k indexes the groups of 8 bits, n is the number of samples in 8 bits, and N is the number of samples in 64 bits. In practice, $\bar{f}_C$ equals ***fCarrier*** and $\bar{f}_M$ equals ***fDeviation***. For a packet of maximum duration it will be true that $k \le 59$.

The transmitter shall have the same ***pDevFreq*** for non-BIST packets and BIST packets.

5.10.2 Transmitter Characteristics

In order to allow for low cost and simple receivers, there is a requirement for the transmitted waveform to have a minimum of edge steepness. The transmitted waveform shall fulfill the eye diagram mask in Figure 5-10.

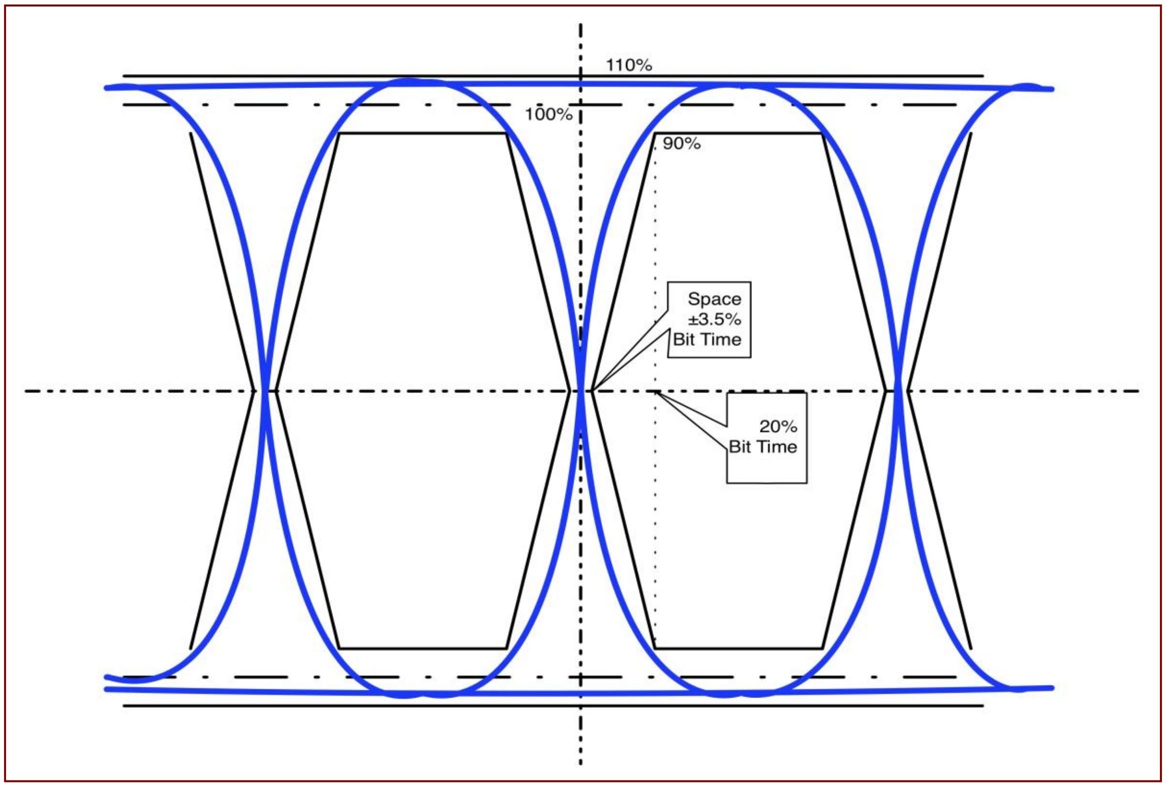


Figure 5-10 Eye diagram of Modulation

In order to manage the noise emitted from the cables, the emitted spectrum shall comply with the following mask in Figure 5-11. Normal rules and regulations for noise emissions shall still be applicable.

Side lobes outside the coverage of Figure 5-11 shall be kept below the level as the figure shows.

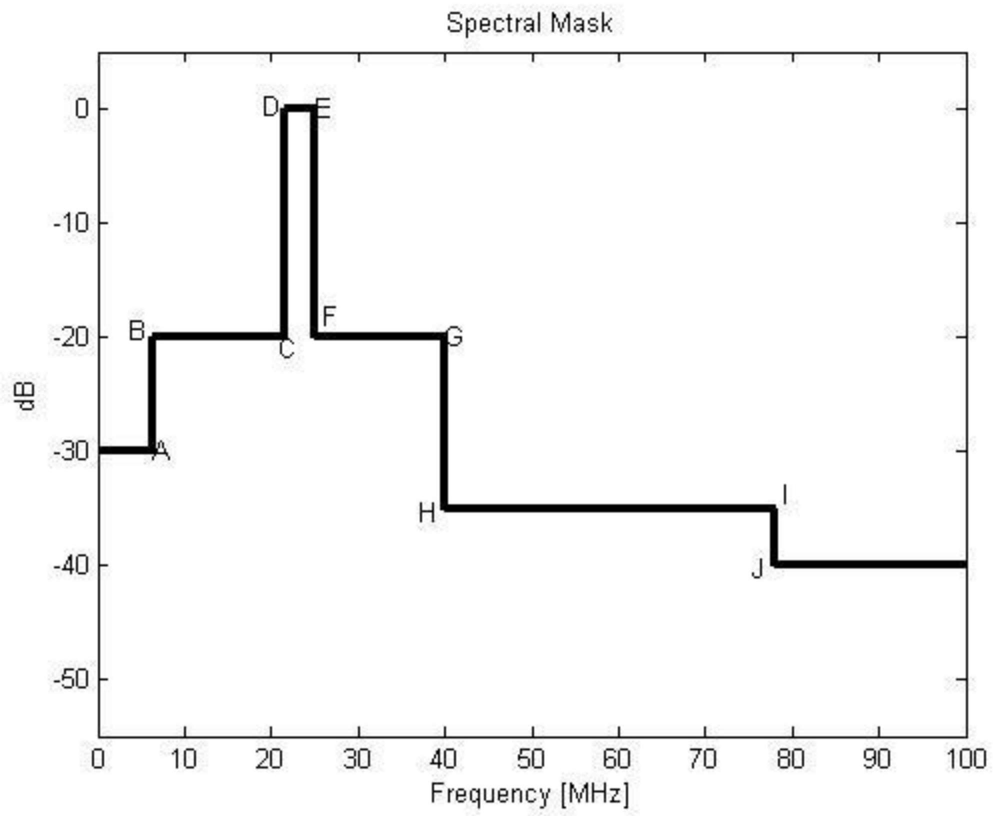


Figure 5-11 Spectrum Mask for Transmitted Signal

The corners in Figure 5-11 are specified in Table 5-12.

Table 5-12 Spectrum Mask Corners

Frequency (MHz)	Attenuation (dB)
<6.4 (A)	-30
6.4 (B) to 21.45 (C)	-20
21.45(D) to 24.89 (E)	0
24.89(F) to 40 (G)	-20
40(H) to 78(I)	-35
>78 (J)	-40

The electrical requirements specified in Table 5-11 shall apply to the transmitter.

5.11 Receiver Specifications

5.11.1 Receiver Electrical Parameters

The electrical requirements specified in Table 5-13 shall apply to the receiver. There are two different squelch modes for the receiver. The Squelch Detection mode is used when the receiver is implementing cable-type detection (see Section 4.4). The Squelch Operating mode is when the receiver is watching for a packet to arrive. These two squelch modes have different required sensitivities. In the Squelch Detection mode, the receiver shall detect signals exceeding ***vSquelchDetecting***. In the Squelch Operating mode the receiver shall meet the requirement ***nBER*** when the signal level exceeds ***vSquelchOperating***. While the receiver is processing a packet, if at any time V_{BUS} becomes idle the receiver shall stop processing the packet and discard it (no ***GoodCRC*** message is returned).

If at any time the signal level is less than ***vSquelchOperating*** then V_{BUS} is declared idle.

The input impedance of the receiver shall be ***zRX*** as measured from V_{BUS} to GND in the band from 19MHz to 27MHz. The high impedance is required for cable-type detection (see Section 4.4).

Table 5-13 Receiver Normative Requirements

Name	Description	Min	Nom	Max	Units	Comment
vSquelchOperating	Squelch detection sensitivity	35		55	mVRMS	
vSquelchDetecting	Squelch detection sensitivity	15	20	25	mVRMS	
nBER	Bit error rate, S/N = 25 dB			10^{-6}		
zRX	Receiver input impedance	10			kΩ	15% tolerance, measured from V_{BUS} to GND in the band from 19 MHz to 27 MHz. Note: The high impedance allows the cable detection to work.
tBitStreamComplete	The Bit Stream has stopped if the squelch has closed for tBitStreamComplete.	1		3	ms	

5.11.2 Receiver Filter Specification

The design of the receiver filter represented by the “Filter Bandpass” in Figure 5-2 is implementation specific, but should take into account the out-of-band power-supply noise (see Sections 7.1.9 and 7.2.7).

5.11.3 Crosstalk in the cables

In order to maintain good communications, the cables shall fulfill the crosstalk requirements in Section 3.6.6.

5.12 Built in Self-Test (BIST)

5.12.1 BIST PRBS Pattern

The generator polynomial for the PRBS-8 pattern shall be $x^8 + x^6 + x^5 + x^4 + 1$.

Figure 5-12 shows an example implementation of the PRBS-generator and checker.

The preloaded pattern shall be "all ones" i.e. all 8 bits in the shift register shall be set to "1". The pattern shall be preloaded when the command to enter test mode is given or received.

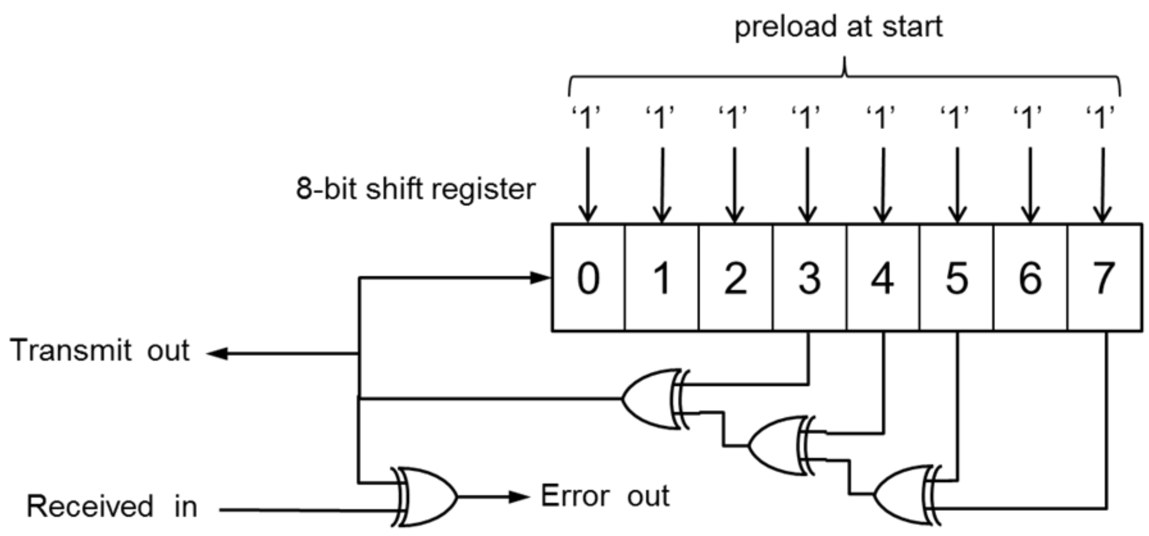


Figure 5-12 Example implementation of the BIST generator and checker

In BIST Transmit or Receiver Test Modes Frames are constructed as shown in Figure 5-13 with a test pattern as defined in Section 5.12.1. Note that the frame does not include an EOP. At least ***nBISTConfidence*** of these frames shall be sent/received without error (see Section 5.12.1.1).

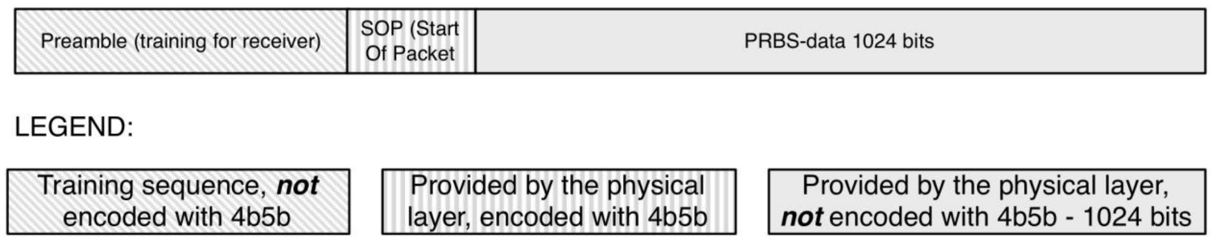

Preamble (training for receiver)
SOP (Start Of Packet
PRBS-data 1024 bits
LEGEND:
Training sequence, not encoded with 4b5b
Provided by the physical layer, encoded with 4b5b
Provided by the physical layer, not encoded with 4b5b - 1024 bits

Figure 5-13 Transmitted test pattern

The PRBS data shall be continued without change in the PRBS generator between frames. If the payloads from all test frames were concatenated the resulting stream shall look like it was generated directly by the BIST generator.

The packet shall have a fixed length and the only other signaling that shall be recognized in the test mode is the ***Hard Reset*** signaling, which shall be used to exit the test mode.

Since the payload length (***nBISTPayload***) and the BIST pattern cycle length are relatively prime, every pattern will eventually appear in every position providing a test of all pattern related weaknesses.

5.12.1.1 Test Pattern Frame Transmission

- The number of bits transferred needed to demonstrate the required ***nBER*** (see Table 5-14) at a 99% confidence level is 4.61x10^6 (see

[Maxim37]). To reach this level of confidence, a minimum of ***nBISTConfidence*** Test Pattern Frames shall be transmitted. To end the test sequence, ***Hard Reset*** signaling shall be sent.

- If errors are detected more bits shall be sent , see

[Maxim37]). The number of frames versus. the number of allowable error is given in Table 5-14.

Table 5-14 Allowable Bit Errors vs. Number of Test Pattern Frames

N (number of allowable errors)	Minimum number of frames required for Confidence level 99%
0	4502
1	6483
2	8209
3	9810
4	11333
5	12802
6	14230
7	15625
8	16995
10	19673
15	26117
20	32328
30	44337

5.12.1.2 Error Counters

The UUT shall maintain a count of errors detected ***BISTErrorCounter*** (see Section 6.6.5). The number of errors shall be compared to the number of errors expected from the number of sent bits and the allowed error rate. Typical testing would take place at each supported voltage and in the presence of an acceptable noise level.

5.12.2 BIST Carrier Purity Test Mode-0

In BIST Carrier Purity Test Mode 0 the Physical Layer shall send out a continuous string of "0"s. This produces a continuous frequency that will allow measurement of ***fCarrier*** - ***fDeviation*** and ***pCarrierNoise.***

5.12.3 BIST Carrier Purity Test Mode-1

In BIST Carrier Purity Test Mode 1 the Physical Layer shall send out a continuous string of "1"s. This produces a continuous frequency that will allow measurement of ***fCarrier*** + ***fDeviation*** and ***pCarrierNoise***.

5.12.4 BIST Eye Pattern Test Mode

In BIST Eye Pattern Test Mode the Physical Layer shall send out a continuous string of bits in accordance with Section 5.12.1. This produces a signal that will allow measurement of the eye pattern and of the spectrum mask.

5.12.5 BIST Carrier Test Mode-2

In ***BIST Carrier Mode 2,*** the Physical Layer shall send out a continuous string of alternating "1"s and “0”s. This enables the measurement of power supply noise and frequency drift.

5.12.6 BIST Parameters

Table 5-15 BIST Parameters

Parameter	Description	Min	Nom	Max	Units	Comment
nBISTConfidence	Number of test frames to transmit in order to reach 99% confidence level	4502			Frames	
nBISTPayload	Number of bits in a BIST frame payload	1024		1024	Bits	

6. Protocol Layer

6.1 Overview

This chapter describes the requirements of the USB Power Delivery Specification's protocol layer including:

- Details of how messages are constructed and used
- Use of timers and timeout values
- Use of message and retry counters
- Reset operation
- Error handling
- State behavior

Refer to Section 2.3 for an overview of the theory of operation of USB Power Delivery.

6.2 Messages

This specification defines two types of messages:

- ***Control Messages*** that are short and used to manage the message flow between Port Partners or to exchange commands that require no additional data. ***Control Messages*** are 16 bits in length.
- ***Data Messages*** that are used to exchange information between a pair of Port Partners. Data Messages range from 48 to 240 bits in length. There are three types of ***Data Messages***:
    - Those used to expose capabilities and negotiate power
    - Those used for the BIST
    - Those that are Vendor Defined

6.2.1 Message Construction

All messages shall be composed of a header and a variable length (including zero) data portion. A message either originates in the Protocol Layer and is passed to the Physical Layer, or it is received by the Physical Layer and is passed to the Protocol Layer.

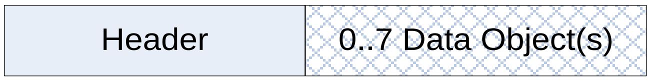


Figure 6-1 High Level Message Structure

6.2.1.1 Message Header

Every message shall start with a 16-bit header, as defined in Figure 6-1. The header contains the basic information used by the Physical Layer to send the message to its Port Partner. The header may be used standalone as a ***Control Message*** when the ***Number of Data Objects*** field is zero, or as the first part of a ***Data Message*** when the ***Number of Data Objects*** field is non-zero.

Table 6-1 Message Header

Bit(s)	Field Name	Notes
15	Reserved	Shall be set to 0
14..12	***Number of Data Objects***	See Section 6.2.1.2
11..9	***MessageID***	See Section 6.2.1.3
8	***Port Role***	See Section 6.2.1.4
7..6	***Specification Revision***	See Section 6.2.1.5
5..4	Reserved	Shall be set to 0
3..0	***Message Type***	See Section 6.2.1.6

6.2.1.2 Number of Data Objects

The 3-bit ***Number of Data Objects*** field shall indicate the number of 32-bit Data Objects that follow the header. When this field is zero the message is a ***Control Message*** and when it is non-zero, the message is a ***Data Message***.

6.2.1.3 MessageID

The 3-bit ***MessageID*** field is the value generated by a rolling counter maintained by the originator of the message. The ***MessageIDCounter*** shall be initialized to zero at power-on as a result of a Soft Reset, or a Hard Reset. The ***MessageIDCounter*** shall be incremented when a message is successfully received as indicated by receipt of a ***GoodCRC*** message.

6.2.1.4 Port Role

The 1-bit ***Port Role*** field shall indicate the current role of the port:

- 0b Sink
- 1b Source

If a port receives a message indicating that its Port Partner is operating in the same role as itself, an error condition exists and it shall issue ***Hard Reset*** signaling. The only exception to this rule is during the Swap Sequence between the first Source sending a ***PS_RDY*** message indicating that its power supply is turned off and the new Source sending a ***PS_RDY*** message to indicate it is now the Source. In this case the ***Port Role*** bit shall not change until the port is ready to start supplying/consuming power (see Figure 8-5).

6.2.1.5 Specification Revision

The 2-bit ***Specification Revision*** field shall indicate the revision of the Power Deliver Specification supported by the Device.

- 00b –Version 1.0
- 01b - 11b – Reserved

6.2.1.6 Message Type

The 4-bit ***Message Type*** field shall indicate the type of message being sent. To fully decode the ***Message Type***, the ***Number of Data Objects*** field is first examined to determine whether the message is a ***Control Message*** or a ***Data Message***. Then the specific ***Message Type*** can be found in Table 6-2 (Control Message) or Table 6-3 (Data Message).

6.3 Control Message

A message is defined as a Control Message when the ***Number of Data Objects*** field in the header is set to 0. The Control Message consists only of a Message Header and a CRC. The Protocol Layer originates the Control Messages (i.e. ***Accept*** message, ***Reject*** message etc.).

The Control Message types are specified in the Header's ***Message Type*** field (bits 3...0) and are summarized in Table 6-2.

Table 6-2 Control Message Types

Bits 3..0	Message Type	Sent by	Description
0000	Reserved	N/A	All values not explicitly defined are reserved and shall not be used.
0001	***GoodCRC***	Source or Sink	See Section 6.3.1.
0010	***GotoMin***	Source only	See Section 6.3.2.
0011	***Accept***	Source or Sink	See Section 6.3.3.

Bits 3..0	Message Type	Sent by	Description
0100	***Reject***	Source or Sink	SeeS 6.3.4.
0101	***Ping***	Source only	See Section 6.3.5.
0110	***PS_RDY***	Source only	See Section 6.3.6.
0111	***Get_Source_Cap***	Source or Sink	See Section 6.3.7.
1000	***Get_Sink_Cap***	Source or Sink	See Section 6.3.8.
1001	***Protocol Error***	Source or Sink	See Section 6.3.9.
1010	***Swap***	Source or Sink	See Section 6.3.10.
1011	***CableVGO***	Sink only	See Section 6.3.11
1100	***Wait***	Source only	See Section 6.3.12.
1101	***Soft Reset***	Source or Sink	See Section 6.3.13.
1110 - 1111	Reserved	N/A	All values not explicitly defined are reserved and shall not be used.

6.3.1 GoodCRC Message

The ***GoodCRC Control Message*** shall be sent by the receiver to acknowledge that the previous message was correctly received (i.e. had a good CRC). The ***GoodCRC*** message shall return the message's ***MessageID*** so the transmitter can determine that the correct message is being acknowledged. The first bit of the ***GoodCRC*** message shall be returned within ***tTransmit*** after receipt of the last bit of the previous message.

BIST does not send the ***GoodCRC*** message during the data stream (see Section 6.4.2.10).

6.3.2 GotoMin Message

The ***GotoMin*** message applies only to those Sinks that have requested power with the GiveBack capable flag set in the Sink Request Data Object.

It is a directive to the Sink Port to reduce its operating power level to the amount specified in the Minimum Operating Current field of its latest Sink Request Data Object.

The GotoMin process is designed to allow the Source to temporarily reallocate power to meet a short term requirement. For example, a Source may reduce a Sink's power consumption for 10-20 seconds to allow another Sink (e.g. an HDD to spin up).

The Source sends this message as a means to harvest power in order to meet a request for power that it cannot otherwise meet. The Device Policy Manager determines which port or ports will receive the message.

The Sink shall respond to a ***GotoMin*** message by reducing its power consumption to less than or equal to the pre-negotiated value (Minimum Operating Current) within ***tSnkNewPower*** time.

The Source sends a ***GotoMin*** message as a shortcut in the power negotiation process since the Source and Sink have already made a contract with respect to the power to be returned. In essence, the Source does not have to advertise its Capabilities and the Sink does not have to make a Request based on them. The Source simply sends the ***GotoMin*** message in place of the ***Accept*** message normally sent during the power negotiation process (see step 19 in Figure 8-4). The power negotiation process then completes from this point in the normal manner with the Sink sending a ***PS_RDY*** message once the power supply transition is complete. The steps of the GotoMin process are fully described in Figure 8-7.

The Source shall return power to the Sink(s) it has 'borrowed' from using the GotoMin mechanism before it can allocate any 'new' power to other devices.

6.3.3 Accept Message

The ***Accept*** message is a valid response to a ***Request*** or a ***Swap*** message.

- It shall be sent by the Source to signal the Sink that the Source is willing to meet the ***Request*** message.
- It shall be sent by the recipient of the ***Swap*** message to signal that it is willing to swap and has begun the ***Swap*** sequence.

The Accept message shall be sent within ***tReceiverResponse*** of the receipt of the last bit of either the ***Request***Message (see Section 6.5.3) or ***Swap*** message.

6.3.4 Reject Message

The ***Reject*** message is a valid response to a ***Request*** or a ***Swap*** message.

- It shall be sent to signal the Sink that the Source is unable to meet the ***Request*** message. This may be due an invalid request or because the Source can no longer provide what it previously advertised.
- It shall be sent by the recipient of a ***Swap*** message to indicate it is unable to do a Swap at this time.

The ***Reject*** message shall be sent within ***tReceiverResponse*** of the receipt of the last bit of either the ***Request*** Message (see Section 6.5.3) or ***Swap*** message.

6.3.5 Ping Message

The ***Ping*** message shall be sent periodically, every ***tSourceActivity***, by the Source to the Sink. It is used to determine the continued presence of the Sink in the ready state when no other power negotiation or other messaging is taking place (see Figure 8-20 in Section 8.3).

Sources shall periodically send this message except when the system is not operating in USB Power Delivery mode (e.g. in standard ***[USB2.0]***, ***[USB3.0]*** and ***[BC1.2]*** modes). ***Ping*** messages are optional when the Source is operating at ***vSafe5V***.

6.3.6 PS_RDY Message

The ***PS_RDY*** message shall be sent by the Source (or by both the Source and Sink during the Swap sequence) to indicate its power supply has reached the desired operating condition. See Section 8.3.2.3 or Section 8.3.2.5 for examples of use.

6.3.7 Get_Source_Cap Message

The ***Get_Source_Cap*** (Get Source Capabilities) message may be sent by a port to request the Source Capabilities and Dual-Role capability of its Port Partner (e.g. dual-role capable). The port shall respond by returning a ***Capabilities*** message (see Section 6.4.1.6.1). If the port is unable to Source power, it shall return a ***Reject*** message.

6.3.8 Get_Sink_Cap Message

The ***Get_Sink_Cap*** (Get Sink Capabilities) message may be sent by a port to request the Sink Capabilities and Dual-Role capability of its Port Partner (e.g. dual-role capable). The port shall respond by returning a ***Capabilities*** message (see Section 6.4.1.6.2). If the port is unable to sink power, it shall return a ***Reject*** message.

6.3.9 Protocol Error Message

The ***Protocol Error*** message shall be sent in response to any command the receiver does not recognize, support or cannot be responded to with the exception of the ***Vendor Defined*** message. It signals a recoverable error condition.

6.3.10 Swap Message

The ***Swap*** message may be sent by either Port Partner. The recipient of the message shall respond by sending an ***Accept*** message or a ***Reject*** message.

- If an ***Accept*** message is sent, the Source and Sink shall exchange operational roles.
- If a ***Reject*** message is sent, the requester is informed that the recipient is unable, or unwilling, to do a Swap and no action shall be taken.

For more information regarding the Swap, refer to Sections 7.3.9 and 7.3.10 in the Power Supply chapter, or Figure 8-5 in the Device Policy chapter.

6.3.11 CableVGO Message

A ***CableVGO*** (V_{BUS} Ground Offset) message shall only be initiated by a Sink. It is sent to its Port Partner to inform it that the IR drop over the ground connection has exceeded the allowable drop. A USB Power Delivery capable Sink shall:

- Monitor V_{GND_DROP} (directly or indirectly) see Section 4.3
- Send this message when V_{GND_DROP} is exceeded

In the case that the Sink wants to retain reliable data communication over ***[USB2.0]*** the Sink shall reduce its power consumption so that V_{GND_DROP} is not exceeded. See also Section 4.3.

6.3.12 Wait Message

The ***Wait*** message is a valid response to a ***Request*** message. It shall be sent by the Source to signal the Sink that the Source is unable to meet the request at this time.

The ***Wait*** message is used by the Source when a Device that has reserved power, requests it. The ***Wait*** message allows the Source time to recover the power it requires to meet the request through the GotoMin process.

It shall be sent within ***tReceiverResponse*** of the receipt of the last bit of the ***Request*** Message (see Section 6.5.3).

The Sink is allowed to repeat the ***Request*** message using the ***SinkRequestTimer*** to ensure that there is ***tSinkRequest*** between requests.

6.3.13 Soft Reset Message

A ***Soft Reset*** message may be initiated by either the Source or Sink to its Port Partner requesting a Soft Reset. The ***Soft Reset*** message shall cause a Soft Reset of the connected Port Pair (see Section 6.7.1). The ***Soft Reset*** message shall not be retried; if it fails a Hard Reset shall be initiated.

A ***Soft Reset*** message is used to recover from Protocol Layer errors; putting the message counters to a known state in order to regain message synchronization. The ***Soft Reset*** message has no effect on the Source or Sink; that is the previously negotiated direction. Voltage and current remain unchanged.

A ***Soft Reset*** message may be sent by either the Source or Sink when there is a message synchronization error; such as an unexpected ***MessageID*** value. If the error is not corrected by the Soft Reset, ***Hard Reset*** signaling shall be issued (see Section 6.7).

6.4 Data Message

A Data Message shall consist of a message header and followed by one or more Data Objects. Data Messages are easily identifiable because the ***Number of Data Objects*** field in the header is a non-zero value.

There are four types of Data Objects:

- Power Data Object (PDO) used to expose a Source port's power capabilities or a Sink's power requirements
- Request Data Object (RDO) used by a Sink port to negotiate a power contract
- BIST Data Object (BDO) used for PHY compliance testing
- Vendor Defined Data Object (VDO) used to convey vendor specific information

The type of Data Object being used in a Data Message is defined by the Message Header's ***Message Type*** field and is summarized in Table 6-3.

Table 6-3 Data Message Types

Bits 3..0	Type	Description
0000	Reserved	All values not explicitly defined are reserved and shall not be used.
0001	***Capabilities***	See Section 6.4.1.
0010	***Request***	See Section 6.4.2.
0011	***BIST***	See Section 6.4.2.10
0100-1110	Reserved	All values not explicitly defined are reserved and shall not be used.
1111	***Vendor Defined***	See Section 6.4.4

6.4.1 Capabilities Message

A Source port shall report its capabilities in a series of 32-bit Power Data Objects (see Section 6.2.1.2) as part of a ***Capabilities*** message (see Figure 6-2). Power Data Objects are used to convey a port's capabilities to Source power including Dual-Role ports presently operating as a Sink.

Each Power Data Object shall describe a specific Source capability such as a battery (e.g. 2.8-4.1V) or a fixed power supply (e.g. 12V) at a maximum allowable current. The ***Number of Data Objects*** field in the header shall define the number of Power Data Objects that follow the Header in a Data Message. All Sources shall minimally offer one Power Data Object that reports ***vSafe5V***.

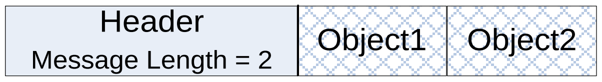


Figure 6-2 Example Capabilities Message with 2 Power Data Objects

In Figure 6-2, the ***Number of Data Objects*** field is 2, so the Source in this case can provide ***vSafe5V*** (required of all Providers) plus one other voltage.

In response to a ***Get_Sink_Cap*** message, a Sink shall report its power requirements by returning one or more 32-bit Power Data Objects in a ***Capabilities*** message. These Power Data Objects provide information on the voltage and current combinations the Sink can consume.

A Sink shall evaluate every ***Capabilities*** message it receives. If its power consumption exceeds the Source's capabilities it shall re-negotiate so as not to exceed the Source's most recently advertised capabilities.

6.4.1.1 Receipt of Unexpected Capabilities Message

When a Provider/Consumer port that is operating as a Source receives an unexpected ***Capabilities*** message (e.g. one not in response to a ***Get_Sink_Cap*** or ***Get_Source_Cap*** message), it shall silently remove ***vSafe5V*** from V_{BUS} and assume the role of a Sink.

Table 6-4 shows the behavior of various combinations of some of the port types.

Table 6-4 Port to Port Behavior

Port 1	Port 2	Description
Provider/Consumer	Provider/Consumer	Either Port 1 or Port 2 will take on the role of Sink depending on which sends its Capabilities first.
Provider/Consumer	Provider	Port 1 will take on the role of Sink
Provider	Provider	Safe operation (see Section 7.1.10).
Provider/Consumer	Legacy host	Safe operation (see Section 7.1.10).
Consumer/Provider	Consumer/Provider	Undefined.
Legacy host	Legacy host	Outside scope of this specification

6.4.1.2 Power Data Object

Power Data Objects (PDO) are identified by the Header's Type field when it is set to 0001b. They shall be used to form ***Capabilities*** messages.

There are four types of Power Data Objects. They contain additional information beyond that encoded in the Header to identify each of the four types of Power Data Objects:

- Fixed Supply is the most commonly used to expose well regulated fixed voltage power supplies.
- Variable Power Supply is used to expose very poorly regulated power supplies.
- Battery is used to expose batteries than can be directly connected to V_{BUS}.

One, or more, of these Power Data Objects shall be used to convey a Source's capabilities, or a Sink's requirements. Lists of Power Data Objects shall be exchanged to allow a Source and Sink to understand the port's power capabilities of each other in order to negotiate a mutually agreeable power contract. They shall also be used to expose additional capabilities that may be utilized; such as in the case of a Swap.

Both Sources and Sinks expose their power capabilities and power requirements by sending a ***Capabilities*** message composed of a number of 32-bit Power Data Objects (see Section 6.4.1.6).

Table 6-5 Power Data Object

Bit(s)	Description	
B31..30	Value	Parameter
	00b	Fixed supply (Vmin = Vmax)
	01b	Battery
	10b	Variable Supply (non-battery)
	11b	Reserved
B29..0	Power Capabilities are described by the following PDOs	

6.4.1.3 Fixed Supply Power Data Object

Table 6-6 describes the Fixed Supply (00b) PDO. See Chapter 7 for the electrical requirements of the power supply.

Since all USB Providers shall support ***vSafe5V***, the required ***vSafe5V*** Fixed Supply Power Supply object is also used to convey additional information that is returned in bits 29 through 20. All other Fixed Supply Power Supply objects shall set bits 29…20 to zero.

Table 6-6 Fixed Supply PDO

Bit(s)	Description
B31..30	Fixed supply
B29	Dual-Role
B28	USB Suspend Supported
B27	Externally Powered
B26	USB Communications Capable
B25..20	Reserved – shall be set to zero.
B19..10	Voltage in 50mV units
B9..0	Maximum Current in 10mA units

6.4.1.3.1 Dual-Role

The Dual-Role bit shall be set when the port is Dual-Role capable. It shall be cleared when the port is simply a Provider or a Consumer.

6.4.1.3.2 USB Suspend Supported

Prior to a PD contract, this flag is undefined and Sinks shall follow the rules for suspend as defined in ***[USB2.0]***, ***[USB3.0]*** or ***[BC1.2]***. After a PD Contract has been negotiated:

1. If the USB Suspend Supported flag is set, then the Sink shall follow the ***[USB2.0]*** or ***[USB3.0]*** rules for suspend and resume but may draw up to ***pPDSuspend***.
2. If the USB Suspend Supported flag is cleared, then the Sink shall ignore the ***[USB2.0]*** or ***[USB3.0]*** rules for suspend and may continue to draw the negotiated power. However, when USB is suspended, the USB device state is also suspended.

Table 6-7 V_{BUS} Power Parameters

Parameter	Description	Min	Nom	Max	Units
pPDSuspend	Power consumption during suspend			25	mW

6.4.1.3.3 Externally Powered

The Externally Powered bit shall only be set when an AC or other bulk power supply is available.

6.4.1.3.4 USB Communications Capable

The USB Communications Capable bit shall only be set for devices capable of communication over the USB data lines (e.g. D+/- or SS Tx/Rx).

6.4.1.4 Variable Supply (non battery)Power Data Object

Table 6‑9 describes a Variable Supply (non battery) (10b) PDO. See Chapter 7 Power Supply for the electrical requirements of the power supply.

The voltage fields shall define the range that output voltage shall fall within. This does not indicate the voltage that will actually be supplied, except it shall fall within that range.

Table 6-8 Variable Supply (non-battery) PDO

Bit(s)	Description
B31..30	Variable Supply (non-battery)
B29..20	Maximum Voltage in 50mV units
B19..10	Minimum Voltage in 50mV units
B9..0	Maximum Current in 10mA units

6.4.1.5 Battery Supply Power Data Object

Table 6-9 describes a Battery (01b) PDO. See Chapter 7 for the electrical requirements of the power supply.

The voltage fields shall represent the battery's voltage range. The battery shall be capable of supplying the Power value over the entire voltage range. Note, only the Battery PDO uses power instead of current.

The Source and Sink shall not negotiate a power level that would allow the current to exceed the maximum current supported by the cable.

The Sink may monitor the battery voltage.

Table 6-9 Battery Supply PDO

Bit(s)	Description
B31..30	Battery
B29..20	Maximum Voltage in 50mV units
B19..10	Minimum Voltage in 50mV units
B9..0	Maximum Allowable Power in 250mW units

6.4.1.6 Use of the Capabilities Message

A ***Capabilities*** message shall have at least one Power Data Object for ***vSafe5V***. The ***Capabilities*** message shall also contain the sending port's information followed by up to 6 additional Power Data Objects. Power Data Objects in a ***Capabilities*** message shall be sent in the following order:

- The ***vSafe5V*** Fixed Supply Object shall always be the first object.
- The remaining Fixed Supply Objects, if present, shall be sent in voltage order; lowest to highest.
- The Battery Supply Objects, if present shall be sent in Minimum Voltage order; lowest to highest.
- The Variable Supply (non battery) Objects, if present, shall be sent in Minimum Voltage order; lowest to highest.

6.4.1.6.1 Use by Sources

Following a HardReset, a power-on event or plug insertion event, a Source port shall send a ***Capabilities*** message every ***tSendSourceCap*** as an advertisement that shall be interpreted by the Sink port on attachment. The Source shall continue sending a minimum of ***nCapsCount Capabilities*** messages:

- Until a successful negotiation has occurred, or
- Until the Sink has returned a Request Data Object with the Capability Mismatch flag set

Additionally, a ***Capabilities*** message shall be sent in each of the following cases:

- Sent by the Source port in order to force re-negotiation or upon a change in its ability to supply power
- Sent in response to a ***Get_Source_Cap*** message
- Sent in response to a ***Get_Sink_Cap*** message

6.4.1.6.2 Use by Sinks

A USB Power Delivery capable Sink, upon detecting ***vSafe5V*** on V_{BUS} and after a ***SinkActivityTimer*** timeout without seeing a ***Capabilities*** message, shall send a HardReset. If the attached Sourceis USB Power Delivery capable, it responds by sending ***Capabilities*** messages thus allowing power negotiations to begin.

6.4.2 Request Message

A ***Request*** message shall be sent by a Sink to request power, typically during the request phase of a power negotiation. The Request Data Object shall be returned by the Sink making a request for power. It shall be sent in response to the most recent ***Capabilities*** message (see Section 8.3.2.3) if the current negotiated contract cannot be met by the ***Capabilities*** message. A ***Request*** message shall return one and only one Sink Request Data Object that shall identify the Power Data Object being requested.

The ***Request*** message includes the requested power level. For example, if the ***Capabilities*** message includes a Fixed Supply PDO that offers 12V @ 1.5A and if the Sink only wants 12V @ 0.5A, it will set the Operating Current field to 50 (e.g. 10mA * 50 = 0.5A). The ***Request*** message requests the highest current the Sink will ever require in the Maximum Operating Current Field (in this example it would be 100 (100 * 10mA = 1.0A)).

The request takes one of two forms depending on the kind of power requested. The Fixed Power Supply Object and Variable Power Supply Object share a common format (see Table 6-10 and Table 6-11). The Battery Power Supply Object uses a different format (see Table 6-12 and Table 6-13).

Table 6-10 Fixed and Variable Request Data Object

Bits	Description
B31	Reserved – shall be set to zero
B30..28	Object position (000b is reserved)
B27	GiveBack flag = 0-
B26	Capability Mismatch
B25	USB Communications Capable
B24..20	Reserved - shall be set to zero.
B19..10	Operating current in 10mA units
B9..0	Maximum Operating Current 10mA units

Table 6-11 Fixed and Variable Request Data Object with GiveBack Support

Bits	Description
B31	Reserved – shall be set to zero
B30..28	Object position (000b is reserved)
B27	GiveBack flag =1
B26	Capability Mismatch
B25	USB Communications Capable
B24..20	Reserved - shall be set to zero.
B19..10	Operating Current in 10mA units
B9..0	Minimum Operating Current 10mA units

Table 6-12 Battery Request Data Object

Bits	Description
B31	Reserved – shall be set to zero
B30..28	Object position (000b is reserved)
B27	GiveBackFlag = 0
B26	Capability Mismatch
B25	USB Communications Capable
B24..20	Reserved - shall be set to zero.
B19..10	Operating Power in 250mW units
B9..0	Maximum Operating Power in 250mW units

Table 6-13 Battery Request Data Object with GiveBack Support

Bits	Description
B31	Reserved – shall be set to zero
B30..28	Object position (000b is reserved)
B27	GiveBackFlag = 1
B26	Capability Mismatch
B25	USB Communications Capable
B24..20	Reserved - shall be set to zero.
B19..10	Operating Power in 250mW units
B9..0	Minimum Operating Power in 250mW units

6.4.2.1 Object Position

The value in the Object Position field shall indicate which object in the ***Capabilities*** message the RDO refers. The value 1 always indicates the 5V Fixed Supply PDO as it is the first object following the ***Capabilities*** message header. The number 2 refers to the next PDO and so forth.

6.4.2.2 GiveBack Flag

The GiveBack flag shall be set to indicate that the Sink will respond to a ***GotoMin*** message by reducing its load to the Minimum Operating Current. It will typically be used by a Device while charging its battery because a short interruption of the charge will have minimal impact on the user and will allow the Source to manage its load better.

6.4.2.3 Capability Mismatch

A Capability Mismatch occurs when the Sink can not satisfy its power requirements from the capabilities offered by the Source. In this case the Capability Mismatch bit shall be set, the Object position field shall be set to 001b, and all other fields in the RDO are undefined (see Section 8.2.5.2).

When a Sink returns a Request Data Object in response to advertised capabilities with this bit set, it indicates that the Sink wants power that the Source cannot provide. This may be due to either a voltage that is not available or the amount of available current.

6.4.2.4 USB Communications Capable

The USB Communications Capable flag shall be set when the Sink has USB data lines and is capable of communicating using either ***[USB2.0]*** or ***[USB3.0]*** protocols. This is used by the Source to determine operation in certain cases such as USB suspend.

6.4.2.5 Operating Current

The Operating current field in the Request Data Object shall be set to the actual amount of current the Sink wants at this time. In conjunction with the Maximum Operating Current field, it provides the Source with additional information that allows it to better manage the distribution of its power. This field shall apply to the Fixed and Variable RDO.

6.4.2.6 Maximum Operating Current

The Maximum Operating Current field in the Request message shall be set to the highest current the Sink will ever require. This gives a Source with a power supply shared amongst multiple ports information about the power it must reserve so it can intelligently distribute power. This field shall apply to the Fixed and Variable RDO.

6.4.2.7 Minimum Operating Current

The Minimum Operating Current field in the Request message shall be set to the lowest current the Sink requires to maintain operation. When combined with the Operating Current, it gives a Source with a power supply shared amongst multiple ports information about how much power it can temporarily get back so it can to intelligently distribute power. This field shall apply to the Fixed and Variable RDO.

6.4.2.8 Operating Power

The Operating current field in the Request Data Object shall be set to the actual amount of power the Sink wants at this time. In conjunction with the Maximum Operating Power field, it provides the Source with additional information that allows it to better manage the distribution of its power. This field shall apply to the Battery RDO.

6.4.2.9 Maximum Operating Power

The Maximum Operating Power field in the Request message shall be set to the highest power the Sink will ever require. This allows a Source with a power supply shared amongst multiple ports to intelligently distribute power. This field shall apply to the Battery RDO.

6.4.2.10 Minimum Operating Power

The Minimum Operating Power field in the Request message shall be set to the lowest current the Sink requires to maintain operation. When combined with the Operating Power, it gives a Source with a power supply shared amongst multiple ports information about how much power it can temporarily get back so it can to intelligently distribute power. This field shall apply to the Battery RDO.

6.4.3 BIST Test Message

The ***BIST*** message is sent to request the port to enter a Physical Layer test mode that performs one of the following functions:

- Enter the receiver mode which does the following:
  - Zero accumulated ***BISTErrorCounter***
  - Process a series of BIST frames and compare the received data to the expected data pattern
  - Count the number of errors received and add to the ***BISTErrorCounter***
- Enter a transmit mode to send BIST frames to the tester
- Return ***BISTErrorCounter*** to the tester

The message format is as follows:

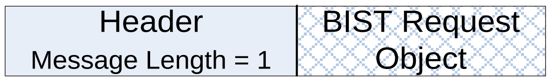


Figure 6-3 BIST Test Message

All ports shall be able to be a Unit Under Test (UUT), and thus shall implement the following operations based on the reception of the appropriate ***BIST*** message Data Object (see Table 6-14):

- Process reception of a ***BIST Receiver Mode***Data Object
- Process reception of a ***BIST Transmit Mode*** Data Object
- Generate a ***Returned BIST Counters*** Data Object response within a ***BIST*** message in response to each received data frame
- Process reception of a ***BIST Carrier Mode 0*** Data Object which shall result in the generation of the appropriate carrier signal
- Process reception of a ***BIST Carrier Mode 1*** Data Object shall result in the generation of the appropriate carrier signal
- Process reception of a ***BIST Carrier Mode 2*** Data Object shall result in the generation of the appropriate carrier signal.
- Process reception of a ***BIST Eye Pattern*** Data Object shall result in the generation of the appropriate carrier signal

It is optional for a port to take on the role of a Tester. All Ports shall support the defined BIST test modes listed in this Section 6.4.3.

It is anticipated that dedicated Testers will exist. Those testers may not be required to implement a local receiver test. However, a Tester shall always be able to complete the operations required when ***BIST*** message with Data Object ***BIST Transmit Mode*** is sent by the Tester.

The usage model of the Phy BIST modes generally assumes that some controlling agent will request a test of its Port Partner. A UUT Port minimally has to process a request to enter test mode and return error counters. A Tester Port shall have a means to place the UUT Port into receiver test mode and retrieve the error counters from the UUT. A Port, that is not part of a Tester, is not expected to be the initiator of a receiver test operation, but is not precluded from doing so.

In Section 8.3.2.9 there is a sequence description of the test sequences that shall be available for compliance testing.

The fields in the BIST Data Object are defined in the Table 6-14.

Table 6-14 BIST Data Object

Bit(s)	**Value**	**Parameter**	Description	Reference
B31..28	0000b	***BIST Receiver Mode***	Requests receiver to enter BIST Receiver Mode	See Section 6.4.3.1
	0001b	***BIST Transmit Mode***	Requests receiver to enter BIST Transmit Mode	See Section 6.4.3.2
	0010b	***Returned BIST Counters***	Returned error counters	See Section 6.4.3.3
	0011b	***BIST Carrier Mode 0***	Requests transmitter to enter BIST Carrier Purity Test Mode-0	See Section 6.4.3.4

Bit(s)	Value	Parameter	Description	Reference
	0100b	***BIST Carrier Mode 1***	Requests transmitter to enter BIST Carrier Purity Test Mode-1	See Section 6.4.3.5
	0101b	***BIST Carrier Mode 2***	Request Transmitter to enter BIST Carrier Test Mode-2	See Section 6.4.3.6
	0110b	***BIST Eye Pattern***	Requests transmitter to enter Eye Pattern Test Mode.	See Section 6.4.3.7
	0111b - 1111b		Reserved	
B27..16		16-bit unsigned integer	When Request Type is ***Returned BIST Counters***, this field shall contain the contents of ***BISTErrorCounter*** otherwise it shall be set to zero.	See Section 6.4.3.3
B15..0			Reserved and shall be set to zero.	

6.4.3.1 BIST Receiver Mode

This operation shall be used to initiate a UUT remote receiver test by sending a ***BIST*** message containing a ***BIST Receiver Mode*** BIST Data object.

On receiving the request, the UUT shall zero its ***BISTErrorCounter*** and both the Tester and the UUT shall preload their PRBS generator with the designated pattern (see Section 5.12.1).

The receiver (UUT) shall acknowledge the ***BIST*** message with a ***GoodCRC*** message.

The test begins after the UUT sends the ***GoodCRC*** message. The Tester sends a test pattern frame and waits for a ***BIST*** message with a Data Object of ***Returned BIST Counters*** from the UUT. After reception of the message, the next test pattern frame is sent.

The test shall be ended by sending ***Hard Reset*** signaling to reset the UUT.

6.4.3.2 BIST Transmit Mode

Loopback mode is not possible so BIST Transmit mode is used to request a UUT transmitter test by sending a ***BIST*** message containing a ***BIST Transmit Mode*** BIST Data object.

Before initiating the request the Tester shall zero its ***BISTErrorCounter*** and preload the PRBS generator with the designated pattern (see Section 5.12.1). On receiving the request the UUT shall preload the PRBS generator with the designated pattern (see Section 5.12.1).

The receiver in the UUT shall acknowledge the ***BIST*** message with a ***GoodCRC*** message. The test begins after the ***GoodCRC*** message. The UUT sends a test pattern frame and waits for a ***BIST*** message with a Data Object of ***Returned BIST Counters*** from the Tester. After reception of the message, the next test pattern frame is sent.

The Tester shall preload its PRBS checker with the designated pattern and start counting errors. After receiving a suitable number of test frames, the Tester shall freeze its error counter. The UUT shall be reset by sending ***Hard Reset*** signaling instead of a ***BIST*** message.

6.4.3.3 Returned BIST Counters

The ***BIST*** message, with a ***Returned BIST Counters*** Data Object, shall contain the error counters obtained during the receiver test.

6.4.3.4 BIST Carrier Purity Test Mode-0

Upon receipt of a ***BIST*** message, with a ***BIST Carrier Mode 0*** Data Object, the UUT shall send out a continuous string of "0"s. This produces a continuous frequency that will allow measurement of ***fCarrier*** - ***fDeviation*** and ***pCarrierNoise***.

The UUT shall be reset by an out of band mechanism to exit this mode (e.g. removing power).

6.4.3.5 BIST Carrier Purity Test Mode-1

Upon receipt of a ***BIST*** message, with a ***BIST Carrier Mode 1*** Data Object, the UUT shall send out a continuous string of "1"s. This produces a continuous frequency that will allow measurement of ***fCarrier*** + ***fDeviation*** and ***pCarrierNoise***.

The UUT shall be reset by an out of band mechanism to exit this mode (e.g. removing power).

6.4.3.6 BIST Carrier Purity Test Mode-2

Upon receipt of a ***BIST*** message, with a ***BIST Carrier Mode 2*** Data Object, the UUT shall send out a continuous string of alternating "1"s and “0”s.

The UUT shall be reset by an out of band mechanism to exit this mode (e.g. removing power).

6.4.3.7 BIST Eye Pattern Test Mode

Upon receipt of a ***BIST*** message, with a ***BIST Eye Pattern*** Data Object, the UUT shall send out a continuous string of bits in accordance with section 5.12.1. This produces a signal that will allow measurement of the eye pattern and of the spectrum mask.

The UUT shall be reset by some out of band mechanism to stop this testing mode (e.g. removing power).

6.4.4 Vendor Defined Message

The ***Vendor Defined*** message (VDM) is provided to allow vendors to exchange information outside of that defined by this specification. If, or how, a vendor uses this mechanism is beyond the scope of this specification.

A ***Vendor Defined*** message shall consist of at least one data object, the First VDM Object, and may contain up to a maximum of six additional VDM Objects.

To ensure vendor uniqueness of ***Vendor Defined*** messages, all ***Vendor Defined*** messages shall contain USB Vendor ID in the First VDM Object.

A port receiving a ***Vendor Defined*** message that it does not recognize shall ignore the message. It shall not send a ***Protocol Error*** message.

The message format shall be as follows:

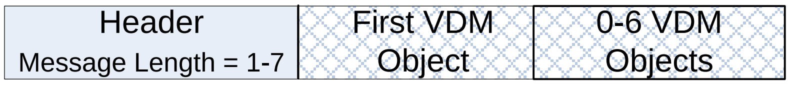


Figure 6-4 Vendor Defined Message

The fields in the First VDM Object shall be as defined in the Table 6-15.

Table 6-15 First VDM Object

Bit(s)	Description	Reference
B31..16	16-bit unsigned integer. USB Vendor ID	
B15..0	Vendor specific	

The fields in the VDM Object are defined in Table 6-16.

Table 6-16 VDM Object

Bit(s)	Description	Reference
B31..0	Vendor specific	

6.5 Timers

All the following timers are defined in terms of bits on the bus regardless of where they are implemented in terms of the logical architecture. This is to ensure a fixed reference for the starting and stopping of timers. It is left to the implementer to ensure that this timing is observed in a real system.

6.5.1 CRCReceiveTimer

The ***CRCReceiveTimer*** shall be used by the sender's Protocol Layer to ensure that a message has not been lost. Failure to receive an acknowledgement of a message (a ***GoodCRC*** message) whether caused by a bad CRC on the receiving end or by a garbled message within ***tReceive*** is detected when the ***CRCReceiveTimer*** expires.

The sender's Protocol Layer response when a ***CRCReceiveTimer*** expires shall be, depending on the message, to either retry ***nRetryCount*** times or to give up and assume communications have been lost.

The ***CRCReceiveTimer*** shall be started when the last bit of the message ***EOP*** has been transmitted on V_{BUS} by the Physical Layer. The ***CRCReceiveTimer*** shall be stopped when the last bit of the ***EOP*** corresponding to the ***GoodCRC*** message has been received by the Physical Layer.

The Protocol Layer receiving a message shall respond with a ***GoodCRC*** message within ***tTransmit*** in order to ensure that the sender's ***CRCReceiveTimer*** does not expire. The ***tTransmit*** shall be measured from when the last bit of the message ***EOP*** has been received by the Physical Layer until the first bit of the preamble of the ***GoodCRC*** message has been transmitted on V_{BUS} by the Physical Layer.

6.5.2 BISTReceiveErrorTimer

The ***BISTReceiveErrorTimer*** shall be used by the sender's Protocol Layer during BIST operation to detect when a frame has been lost and to trigger the transmission of the next test frame. Failure to receive an acknowledgement of a frame (***BIST*** message with a Data Object of ***BISTErrorCounter***) whether caused by a bad CRC on the receiving end or by a garbled message within ***tReceive*** is detected when the ***BISTReceiveErrorTimer*** expires.

The sender's Protocol Layer response when a ***BISTReceiveErrorTimer*** expires shall be to continue operation.

The ***BISTReceiveErrorTimer*** shall be started when the ***nBISTPayload*** bit past the last bit of the ***SOP*** has been transmitted on V_{BUS} by the Physical Layer. The ***BISTReceiveErrorTimer*** shall be stopped when the last bit of the ***EOP*** corresponding to the ***BIST*** message with a Data Object of ***BISTErrorCounter*** has been received by the Physical Layer.

The Protocol Layer receiving a test frame shall respond with a ***BIST*** message with a Data Object of ***BISTErrorCounter*** within ***tTransmit*** in order to ensure that the sender's ***BISTReceiveErrorTimer*** does not expire. The ***tTransmit*** shall be measured from when the last bit of the message ***EOP*** has been received by the Physical Layer until the first bit of the preamble of the ***GoodCRC*** message has been transmitted on V_{BUS} by the Physical Layer.

6.5.3 SenderResponseTimer

The ***SenderResponseTimer*** shall be used by the sender's Protocol Layer to ensure that a message requesting a response (i.e. ***Get_Source_Cap*** message) is responded to within a bounded time of ***tSenderResponse***. Failure to receive the expected response is detected when the ***SenderResponseTimer*** expires.

The Protocol Layer's response when the ***SenderResponseTimer*** expires shall be dependent on the message sent (see Section 8.3).

The ***SenderResponseTimer*** shall be started from the time the last bit of the message ***EOP*** has been transmitted on V_{BUS} by the Physical Layer. The ***SenderResponseTimer*** shall be stopped when the last bit of the expected response message ***EOP*** has been received by the Physical Layer.

The receiver of a message requiring a response shall respond within ***tReceiverResponse*** in order to ensure that the sender's ***SenderResponseTimer*** does not expire.

The ***tReceiverResponse*** time shall be measured from the time the last bit of the message ***EOP*** has been received by the Physical Layer until the first bit of the response message preamble has been transmitted on V_{BUS} by the Physical Layer.

6.5.4 Activity Timers

The Protocol Layer uses activity timers to ensure that there is adequate activity on V_{BUS} to allow the Source to know that a Sink is still present. The activity timer's value and use are specific to the role the Device is playing – either Source or Sink. The values are selected to allow one missed response to a ***Ping*** message without the Source returning its output to the default state.

6.5.4.1 SourceActivityTimer

A PD Source is required to maintain a minimal level of bus traffic. It does so by periodically sending a ***Ping*** message during a connection to a PD Sink, whenever there is no other message traffic.

In order to maintain bus activity the Source shall start its ***SourceActivityTimer*** as described in Section 8.3.3.2. Whenever the timer expires, after ***tSourceActivity***, the Source shall send a ***Ping*** message. It shall then initialize and restart the ***SourceActivityTimer*** ready for the next ***Ping*** message. This ensures that message activity is maintained in cases where the time between normal messages would exceed the Sink's activity timeout. For example power supply transitions may take longer than the activity timeout meaning that a ***Ping*** is sent prior to the ***PS_RDY*** message.

The ***SourceActivityTimer*** shall be reset and restarted when the last bit of the message ***EOP*** has been transmitted on V_{BUS} by the Physical Layer.

A failure to see a ***GoodCRC*** message in response to a ***Ping*** message within ***tReceive*** (after ***nRetryCount*** retries) is indicative of a communications failure and shall cause the Source to send a ***Soft Reset*** message (see Section 6.7.1).

See Section 8.3.3.5 for more details of when ***Ping*** messages shall be transmitted.

6.5.4.2 SinkActivityTimer

The Sink shall support the ***SinkActivityTimer***. Sinks shall observe an absence of a ***Ping***, or other messages within ***tSinkActivity***, as an indication of communications failure and as such shall issue ***Hard Reset*** signaling in order to return to default operation. When a Sink observes an absence of ***Capabilities*** messages ***tSinkActivity*** after V_{BUS} is present, the Sink shall issue ***Hard Reset*** signaling in order to restart the sending of ***Capabilities*** messages by the Source (see Section 6.6.4). Initialization and restarting of the ***SinkActivityTimer*** is described in Section 8.3.3.3.3. Any additional action taken is Device implementation specific. Sinks shall also use the absence of V_{BUS} to return to default operation unless this is part of an ongoing Swap sequence.

The ***SinkActivityTimer*** shall be re-initialized and re-started when the last bit of a ***Ping***, or any other message ***EOP,*** has been received by the Physical Layer.

See Section 8.3.3.3 for more details of when the ***SinkActivityTimer*** shall be run.

6.5.5 Capability Timers

Sources and Sinks use Capability Timers to determine attachment of a PD capable Device. By periodically sending or requesting capabilities it is possible to determine PD Device attachment when a response is received.

6.5.5.1 SourceCapabilityTimer

A Source shall use the ***SourceCapabilityTimer*** to periodically send out a ***Capabilities*** message every ***tSendSourceCap*** prior to a successful negotiation while:

- An A-plug is attached
- The Source is not in an active connection with a PD Sink Port

Whenever there is a ***SourceCapabilityTimer*** timeout the Source shall send a ***Capabilities*** message. It shall then re-initialize and restart the ***SourceCapabilityTimer***. The ***SourceCapabilityTimer*** shall be stopped when the last bit of the ***EOP*** corresponding to the ***GoodCRC*** message has been received by the Physical Layer since a PD connection has been established. At this point the Source shall wait for a ***Request*** message or a response timeout.

See Section 8.3.3.2 more details of when ***Capabilities*** messages shall be transmitted.

6.5.5.2 SinkCapabilityTimer

A Sink may use the ***SinkCapabilityTimer*** in order to periodically issue a ***Get_Source_Cap*** message every ***tGetSourceCap***. Whenever there is a ***SinkCapabilityTimer*** timeout the Sink shall send a ***Capabilities*** message. It shall then re-initialize and restart the ***SinkCapabilityTimer***. The ***SinkCapabilityTimer*** shall be stopped when the last bit of the ***EOP*** corresponding to the ***Capabilities*** message has been received by the Physical Layer since a PD connection has been established. Whenever there is an activity timeout the Sink may resume periodically requesting the Source's capabilities (***Get_Source_Cap*** message).

See Section 8.3.3.2 for more details of when the ***SinkActivityTimer*** may be run.

6.5.6 Sink Request Timer

The ***SinkRequestTimer*** is used to ensure that the time between Sink ***Request*** messages, after a ***Wait*** message has been received from the Host, is a minimum of ***tSinkRequest*** (see Section 6.3.12).

Whenever there is a ***SinkRequestTimer*** timeout the Source may send a ***Request*** message. It shall then re-initialize and restart the ***SinkRequestTimer***.

6.5.7 Power Supply Timers

6.5.7.1 Power Supply Transition Timer

The ***PSTransitionTimer*** is used by the Policy Engine to timeout on a ***PS_RDY*** message. It is started when a request for a new Capability has been accepted and will timeout after ***tPSTransition*** if a ***PS_RDY*** message has not been received. This condition leads to a Hard Reset and a return to default operation. The ***PSTransitionTimer*** relates to the time taken for the Source to transition from one voltage, or current level, to another (see Section 7.2).

The ***PSTransitionTimer*** shall be started when the last bit of the ***Accept*** message ***EOP*** has been received on V_{BUS} by the Physical Layer. The ***PSTransitionTimer*** shall be stopped when the last bit of the ***PS_RDY*** message ***EOP*** has been received by the Physical Layer.

6.5.7.2 Power Supply Off Timer

The ***PSSourceOffTimer*** is used by the Policy Engine in Dual-Role Device that is currently acting as a Sink to timeout on a ***PS_RDY*** message during a Swap sequence. This condition leads to a Hard Reset and return to default operation.

The ***PSSourceOffTimer*** shall be started when:

- The last bit of the ***EOP*** of the ***Swap*** message is received by the Physical Layer from a Dual-Role Device currently acting as a Source
- Or when the last bit of the ***EOP*** of the ***Accept*** message, corresponding to a ***Swap*** message request, has been received by the Physical Layer from the Dual-Role Device currently acting as a Source

The ***PSSourceOffTimer*** shall be stopped when:

- The last bit of the ***EOP*** of the ***PS_RDY*** message is received.

The ***PSSourceOffTimer*** relates to the time taken for the remote Dual-Role Device to stop supplying power (see also Sections 7.3.9 and 7.3.10). The timer shall time out if a ***PS_RDY*** message has not been received from the remote Dual-Role Device within ***tPSSourceOff*** indicating this has occurred.

6.5.7.3 Power Supply On Timer

The ***PSSourceOnTimer*** is used by the Policy Engine in Dual-Role Device that has just stopped supplying power and is waiting to start sinking power to timeout on a ***PS_RDY*** message during a swap. This condition leads to a Hard Reset and return to default operation.

The ***PSSourceOnTimer*** shall be started when:

- The last bit of the ***EOP*** of the ***PS_RDY*** message is transmitted by the Physical Layer

The ***PSSourceOnTimer*** shall be stopped when:

- The last bit of the ***EOP*** of the ***PS_RDY*** message is received by the Physical Layer

The ***PSSourceOnTimer*** relates to the time taken for the remote Dual-Role Device to start supplying power (see also Sections 7.3.9 and 7.3.10) and will time out if a ***PS_RDY*** message indicating this has not been received within ***tPSSourceOn***.

6.5.8 NoResponseTimer

The ***NoResponseTimer*** is used by the Policy Engine in a Source or Sink to determine that its Port Partner is not responding after a Hard Reset. When the ***NoResponseTimer*** times out, the Policy Engine shall issue up to ***nHardResetCount*** additional Hard Resets before determining that the Port Partner is non-responsive to USB Power Delivery messaging.

If the Sink fails to receive a ***Capabilities*** message received within ***tNoResponse*** of:

- The last bit of a ***Hard Reset*** signaling being sent by the PHY layer if the ***Hard Reset*** signaling was initiated by the Sink
- The last bit of a ***Hard Reset*** signaling being received by the PHY layer if the ***Hard Reset*** signaling was initiated by the Source

Then the Sink shall issue additional Hard Resets up to ***nHardResetCount*** times (see Section 6.7.2).

If the Source fails to receive a ***GoodCRC*** message in response to a ***Capabilities*** message within ***tNoResponse*** of:

- The last bit of a ***Hard Reset*** signaling being sent by the PHY layer if the ***Hard Reset*** signaling was initiated by the Sink
- The last bit of a ***Hard Reset*** signaling being received by the PHY layer if the ***Hard Reset*** signaling was initiated by the Source

Then the Source shall issue additional Hard Resets up to ***nHardResetCount*** times (see Section 6.7.2).

For a non-responsive device, the Policy Engine in a Source may either decide to continue sending ***Capabilities*** messages or to go to non-USB Power Delivery operation and cease sending ***Capabilities*** messages.

6.5.9 tBISTMode

tBISTMode is used to define the maximum time that a UUT has to enter a BIST mode when requested by a Tester.

A UUT shall enter the appropriate BIST mode and shall send the last bit of the ***EOP*** of the ***GoodCRC*** message within ***tBISTMode*** of the last bit of the ***EOP*** of the ***BIST*** message used to initiate the test is received by the Physical Layer.

6.5.10 BISTStartTimer

The ***BISTStartTimer*** is used by the Tester to ensure that there is a delay of at least ***tBISTStart*** min before sending the first test frame after requesting test mode. The ***BISTStartTimer*** shall be initialized and run once the ***BIST*** message containing the data object has been sent i.e. a ***GoodCRC*** message has been received. The first test frame shall not be sent until after the ***BISTStartTimer*** has expired.

The ***BISTStartTimer*** shall be started when:

- The last bit of the ***EOP*** of the ***GoodCRC*** message in response to the ***BIST*** message used to initiate the test is received by the Physical Layer

6.5.11 Time Values and Timers

Table 6-17 summarizes the values for the timers listed in this section. Table 6-18 lists the timers.

Table 6-17 Time Values

Parameter	Value (min)	Value (max)	Units
tBISTMode		300	ms
tBISTStart	100		ms
tGetSourceCap	1	2	s
tNoResponse	250	500	ms
tPSSourceOff	200	220	ms

Parameter	Value (min)	Value (max)	Units
tPSSourceOn	200	220	ms
tPSTransition	200	220	ms
tReceive	650	700	µs
tReceiverResponse	12	15	ms
tSenderResponse	24	30	ms
tSendSourceCap	1	2	s
tSinkActivity	120	150	ms
tSinkRequest	100		ms
tSinkTransition		25	ms
tSourceActivity	40	50	ms
tTransmit		75	µs

Table 6-18 Timers

Timer	Parameter	Used By	Section
BISTReceiveErrorTimer	*tReceive*	Protocol	6.5.2
BISTStartTimer	*tBISTStart*	Policy Engine	6.5.10
CRCReceiveTimer	*tReceive*	Protocol	6.5.1
PSSourceOffTimer	*tPSSourceOff*	Policy Engine	6.5.7.2
PSSourceOnTimer	*tPSSourceOn*	Policy Engine	6.5.7.3
PSTransitionTimer	*tPSTransition*	Policy Engine	6.5.7.1
NoResponseTimer	*tNoResponse*	Policy Engine	6.5.8
SenderResponseTimer	*tSenderResponse*	Policy Engine	6.5.3
SinkActivityTimer	*tSinkActivity*	Policy Engine	6.5.4.2
SinkCapabilityTimer	*tGetSourceCap*	Policy Engine	6.5.5.2
SinkRequestTimer	*tSinkRequest*	Policy Engine	6.5.6
SourceActivityTimer	*tSourceActivity*	Policy Engine	6.5.4.1
SourceCapabilityTimer	*tSendSourceCap*	Policy Engine	6.5.5.1

6.6 Counters

6.6.1 MessageID Counter

The ***MessageIDCounter*** is a rolling counter, ranging from 0 to ***nMessageIDCount***, used to detect duplicate messages. This value is used for the ***MessageID*** field in the header of each transmitted message.

Each port shall maintain a copy of the last ***MessageID*** value received from its Port Partner. Devices that support multiple ports, such as Hubs, shall maintain copies of the last ***MessageID*** on a per port basis.

The transmitter shall use the ***MessageID*** in a ***GoodCRC*** message to verify that a particular message was received correctly. The receiver shall use the ***MessageID*** to detect duplicate messages.

6.6.1.1 Transmitter Usage

The Transmitter shall use the ***MessageID*** as follows:

- Upon receiving either ***Hard Reset*** signaling, or a ***Soft Reset*** message, the transmitter shall set its ***MessageIDCounter*** to zero and re-initialize its retry mechanism.
- If a ***GoodCRC*** message with a ***MessageID*** matching the ***MessageIDCounter*** is not received before the ***CRCReceiveTimer*** expires, it shall retry the same packet up to ***nRetryCount*** times using the same ***MessageID***.
- If a ***GoodCRC*** message is received with a ***MessageID*** matching the current ***MessageIDCounter*** before the ***CRCReceiveTimer*** expires, the transmitter shall re-initialize its retry mechanism and increment its ***MessageIDCounter***.
- If the message is aborted by the higher layer protocol, the transmitter shall delete the message from its transmit buffer, re-initialize its retry mechanism and increment its ***MessageIDCounter***.

6.6.1.2 Receiver Usage

The Receiver shall use the ***MessageID*** as follows:

- When the first good packet is received after a reset, the receiver shall store a copy of the received ***MessageID*** value.
- For subsequent messages, if MessageID value in a received message is the same as the stored value, the receiver shall return a ***GoodCRC*** message with that MessageID value and drop the message (this is a retry of an already received message).
- If ***MessageID*** value in the received message is different than the stored value, the receiver shall return a ***GoodCRC*** message with the new ***MessageID*** value, store a copy of the new ***MessageID*** value and process the message.

6.6.2 Retry Counter

The ***RetryCounter*** is used whenever there is a message transmission failure (timeout of ***CRCReceiveTimer***). If the ***nRetryCount*** retry fails, then the link shall be reset using the Soft Reset mechanism.

6.6.3 Hard Reset Counter

The ***HardResetCounter*** is used to retry the Hard Reset whenever there is no response from the remote device (see Section 6.5.8). Once the Hard Reset has been retried ***nHardResetCount*** times then it shall be assumed that the remote device is non-responsive.

6.6.4 Capabilities Counter

The ***CapsCounter*** is used to count the number of ***Capabilities*** messages which have been sent by a Source at power up. Implementation of the ***CapsCounter*** is optional but may be used by any Source which wishes to preserve power by not sending ***Capabilities*** messages after a period of time.

When the Source detects that a Sink is attached then after ***nCapsCount Capabilities*** messages have been sent the Source shall issue ***Hard Reset*** signaling up to ***nHardResetCount*** times (see Section 6.7.2) before it decides that the Sink is non-responsive and it stops sending ***Capabilities*** messages.

A Sink shall use the ***SinkActivityTimer*** to trigger the resending of ***Capabilities*** messages by a USB Power Delivery capable Source which has previously stopped sending ***Capabilities*** messages. Any Sink which is attached and does not detect a ***Capabilities*** message within ***tSinkActivity*** shall issue ***Hard Reset*** signaling in order to reset the Source. Resetting the Source shall also reset the ***CapsCounter*** and restart the sending of ***Capabilities*** messages.

6.6.5 BIST Error Counter

The UUT shall maintain a count of errors detected ***BISTErrorCounter***. The ***BISTErrorCounter*** shall contain the number of bits in error during a BIST Transmit or Receive test.

The ***BISTErrorCounter*** shall be a 16-bit counter that shall be reset to zero upon the BIST test start. The ***BISTErrorCounter*** shall freeze at a value of FFFFh.

6.6.6 Counter Values and Counters

Table 6-20 lists the counters used in this section and Table 6-19 shows the corresponding parameters.

Table 6-19 Counter parameters

Parameter	Value	Section
nMessageIDCount	7	6.6.1
nRetryCount	2	6.6.2
nHardResetCount	2	6.6.3
nCapsCount	50	6.6.4

Table 6-20 Counters

Counter	Max	Section
MessageIDCounter	***nMessageIDCount***	6.6.1
RetryCounter	***nRetryCount***	6.6.2
HardResetCounter	***nHardResetCount***	6.6.3
CapsCounter	***nCapsCount***	6.6.4
BISTErrorCounter	FFFFh	6.6.5

6.7 Reset

Resets are a necessary response to protocol or other error conditions. USB Power Delivery defines two different types of reset; a Soft Reset, that resets protocol, and a Hard Reset which resets both the power supplies and protocol.

6.7.1 Soft Reset

A ***Soft Reset*** message is used to cause a Soft Reset of protocol communication when this has broken down in some way. It shall not have any impact on power supply operation, but shall be used whenever there is a protocol error that it is not possible to handle in another way. The Soft Reset may be triggered by either Port Partner in response to an error.

A Soft Reset shall impact the USB Power Delivery layers in the following ways:

- Physical Layer: Reset not required since the Physical Layer resets on each packet transmission/reception
- Protocol Layer: Reset ***MessageIDCounter***, ***RetryCounter*** and state machines
- Policy Engine: Reset state dependent behavior
- Power Supply: Shall not change

A Soft Reset is performed using a sequence of protocol messages (see Table 8-8). Message numbers shall be set to zero prior to sending the ***Soft Reset***/***Accept*** message since the issue may be with the counters. The sender of a reset message shall reset its transmit and last message received counters, the receiver of the message shall reset its transmit and last message received counters before sending the response. Any failure in the Soft Reset process will trigger a Hard Reset; for example a ***GoodCRC*** message is not received during the Soft Reset process (see Section 6.7.2).

6.7.2 Hard Reset

Hard Resets are signaled by an ordered set as defined in Section 5.7.4. Both the sender and recipient shall cause their Power Supplies to return to their default states (see Sections 7.3.12 to 7.3.17 for details of voltage transitions); this includes returning to default Source/Sink roles if there has been a Swap. In addition their respective Protocol Layers shall be reset as for the Soft Reset. This allows the attached Devices to be in a state where they can re-establish USB PD communication. Hard Reset may be retried up to ***nHardResetCount*** times (see also Sections 6.5.8 and 6.6.3).

6.8 State behavior

6.8.1 Introduction to state diagrams used in Chapter 6

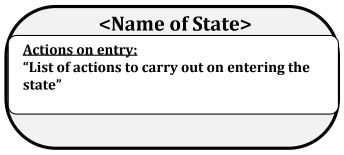


Figure 6-5 Outline of States

Figure 6-5 shows an outline of the states defined in the following sections. At the top there is the name of the state. This is followed by "Actions on entry" a list of actions carried out on entering the state.

Transitions from one state to another are indicated by arrows with the conditions listed on the arrow. Where there are multiple conditions these are connected using either a logical OR "|" or a logical AND "&". The inverse of a condition is shown with a "NOT" in front of the condition.

In some cases there are transitions which can occur from any state to a particular state. These are indicated by an arrow which is unconnected to a state at one end, but with the other end (the point) connected to the final state.

In some state diagrams it is necessary to enter or exit from states in other diagrams. Figure 6-6 indicates how such references are made. The reference is indicated with a hatched box. The box contains the name of the state.

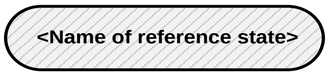


Figure 6-6 References to states

Timers are included in many of the states. Timers are initialized (set to their starting condition) and run (timer is counting) in the particular state it is referenced. As soon as the state is exited then the timer is no longer active. Timeouts of the timers are listed as conditions on state transitions.

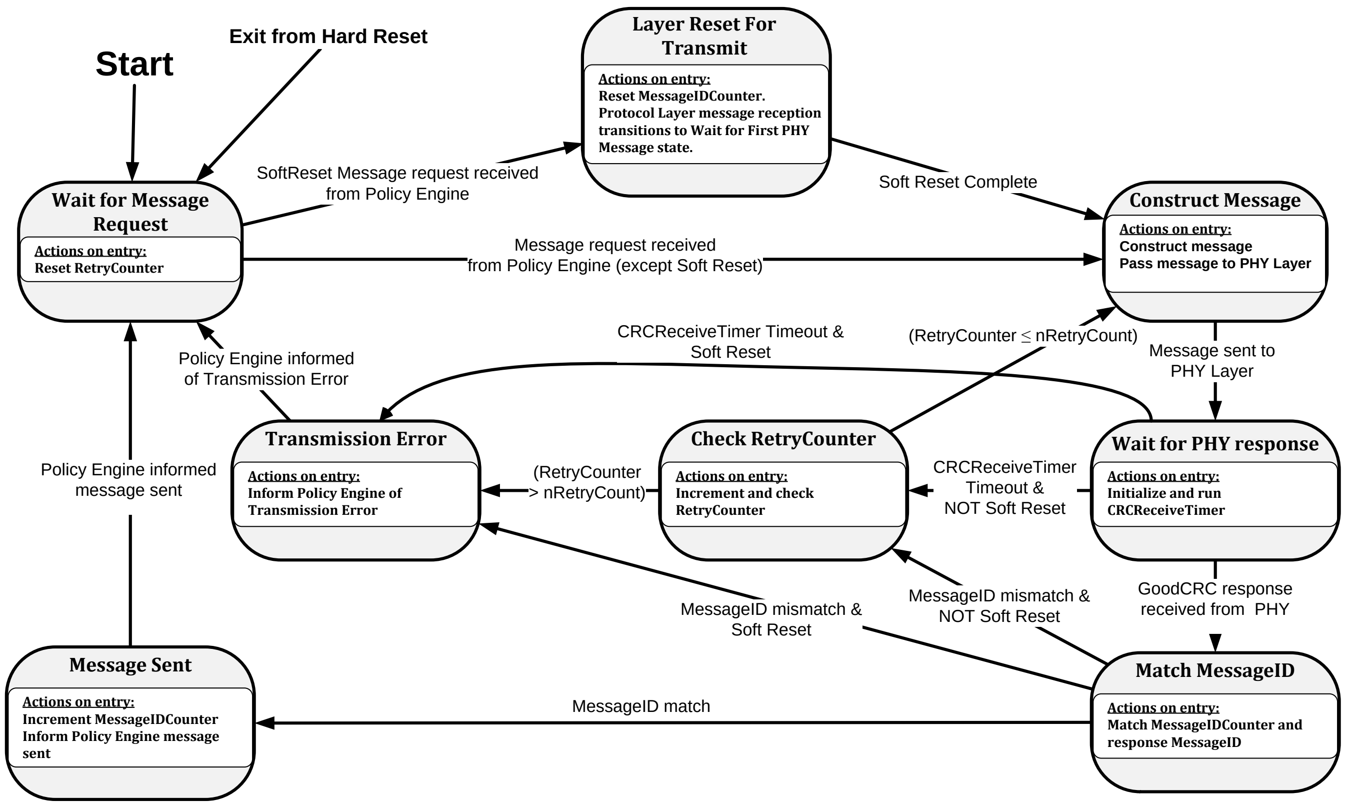
Conditions listed on state transitions will come from one of three sources:

- Messages received from the PHY Layer
- Events triggered within the Protocol Layer e.g. timer timeouts
- Message and related indications passed up to the Policy Engine from the Protocol Layer (message sent, message received etc.)

6.8.2 General Operation

6.8.2.1 Protocol Layer Message Transmission

Figure 6-7 shows the state behavior for the Protocol layer when transmitting a message.



Start
Exit from Hard Reset
Layer Reset For Transmit
Actions on entry:
Reset MessageIDCounter.
Protocol Layer message reception transitions to Wait for First PHY Message state.
SoftReset Message request received from Policy Engine
Soft Reset Complete
Wait for Message Request
Actions on entry:
Reset RetryCounter
Construct Message
Actions on entry:
Construct message
Pass message to PHY Layer
Message request received from Policy Engine (except Soft Reset)
CRCReceiveTimer Timeout & Soft Reset
(RetryCounter ≤ nRetryCount)
Message sent to PHY Layer
Policy Engine informed of Transmission Error
Transmission Error
Actions on entry:
Inform Policy Engine of Transmission Error
Check RetryCounter
Actions on entry:
Increment and check RetryCounter
Wait for PHY response
Actions on entry:
Initialize and run CRCReceiveTimer
Policy Engine informed message sent
(RetryCounter > nRetryCount)
CRCReceiveTimer Timeout & NOT Soft Reset
MessageID mismatch & Soft Reset
MessageID mismatch & NOT Soft Reset
GoodCRC response received from PHY
Message Sent
Actions on entry:
Increment MessageIDCounter
Inform Policy Engine message sent
MessageID match
Match MessageID
Actions on entry:
Match MessageIDCounter and response MessageID

Figure 6-7 Protocol layer message transmission

6.8.2.1.1 Wait for Message Request State

In the ***Wait for Message Request*** state the Protocol Layer waits until Policy Engine directs it to send a message. This shall be the startup state for the Protocol Layer message transmission and also the starting state when exiting from a Hard Reset.

On entry to the ***Wait for Message Request*** state the Protocol Layer shall reset the ***RetryCounter***.

The Protocol Layer shall transition to the ***Construct message*** state when:

- A message request is received from the Policy Engine which is not a ***Soft Reset*** message.

The Protocol Layer shall transition to the ***Layer Reset for Transmit*** state when:

- A message request is received from the Policy Engine which is a ***Soft Reset*** message.

6.8.2.1.2 Layer Reset for Transmit state

On entry to the ***Layer Reset for Transmit*** state the Protocol Layer shall reset the ***MessageIDCounter***. The Protocol Layer shall transition Protocol Layer message reception to the ***Wait for first PHY Message*** state (see Section 6.8.2.2.1) in order to reset the stored ***MessageID***.

The Protocol Layer shall transition to the ***Construct Message*** State when:

- The Soft Reset actions in this state have been completed.

6.8.2.1.3 Construct Message State

On entry to the ***Construct Message*** state the Protocol Layer shall construct the message requested by the Policy Engine, or resend a previously constructed message, and then pass this message to the PHY Layer.

The Protocol Layer shall transition to the ***Wait for PHY Response*** state when:

- The message has been sent to the PHY Layer.

6.8.2.1.4 Wait for PHY Response State

On entry to the ***Wait for PHY Response*** State the Protocol Layer shall initialize and run the ***CRCReceiveTimer***.

The Protocol Layer shall transition to the ***Match MessageID*** state when:

- A ***GoodCRC*** message response is receive from the PHY Layer.

The Protocol Layer shall transition to the ***Check RetryCounter*** state when:

- The ***CRCReceiveTimer*** times out and
- This is not a Soft Reset

The Protocol Layer shall transition to the Transmission Error state when:

- The CRCReceiveTimer times out and
- This is a Soft Reset

6.8.2.1.5 Match MessageID State

On entry to the Match MessageID state the Protocol Layer shall compare the ***MessageIDCounter*** and the ***MessageID*** of the received ***GoodCRC*** message.

The Protocol Layer shall transition to the Message Sent state when:

- The MessageIDCounter and the ***MessageID*** of the received ***GoodCRC*** message match.

The Protocol Layer shall transition to the ***Check RetryCounter*** state when:

- The MessageIDCounter and the ***MessageID*** of the received ***GoodCRC*** message do not match and
- This is not a Soft Reset.

The Protocol Layer shall transition to the ***Transmission Error*** state when:

- The MessageIDCounter and the ***MessageID*** of the received ***GoodCRC*** message do not match and
- This is a Soft Reset

6.8.2.1.6 Message Sent state

On entry to the ***Message Sent*** state the Protocol Layer shall increment the MessageIDCounter, reset the ***RetryCounter*** and inform the Policy Engine that the message has been sent.

The Protocol Layer shall transition to the ***Wait for Message Request*** state when:

- The Policy Engine has been informed that the message has been sent.

6.8.2.1.7 Check RetryCounter State

On entry to the ***Check RetryCounter*** state the Protocol Layer shall increment the value of the ***RetryCounter*** and then check it in order to determine whether it is necessary to retry sending the message.

The Protocol Layer shall transition to the ***Construct Message*** state in order to retry message sending when:

- ***RetryCounter*** ≤ ***nRetryCount***.

The Protocol Layer shall transition to the ***Transmission Error*** state when:

- ***RetryCounter*** > ***nRetryCount***.

6.8.2.1.8 Transmission Error state

On entry to the ***Transmission Error*** state the Protocol Layer shall inform the Policy Engine of the transmission error.

The Protocol Layer shall transition to the ***Wait for Message Request*** state when:

- The Policy Engine has been informed of the transmission error.

6.8.2.2 Protocol Layer message reception

Figure 6-8 shows the state behavior for the Protocol layer when receiving a message.

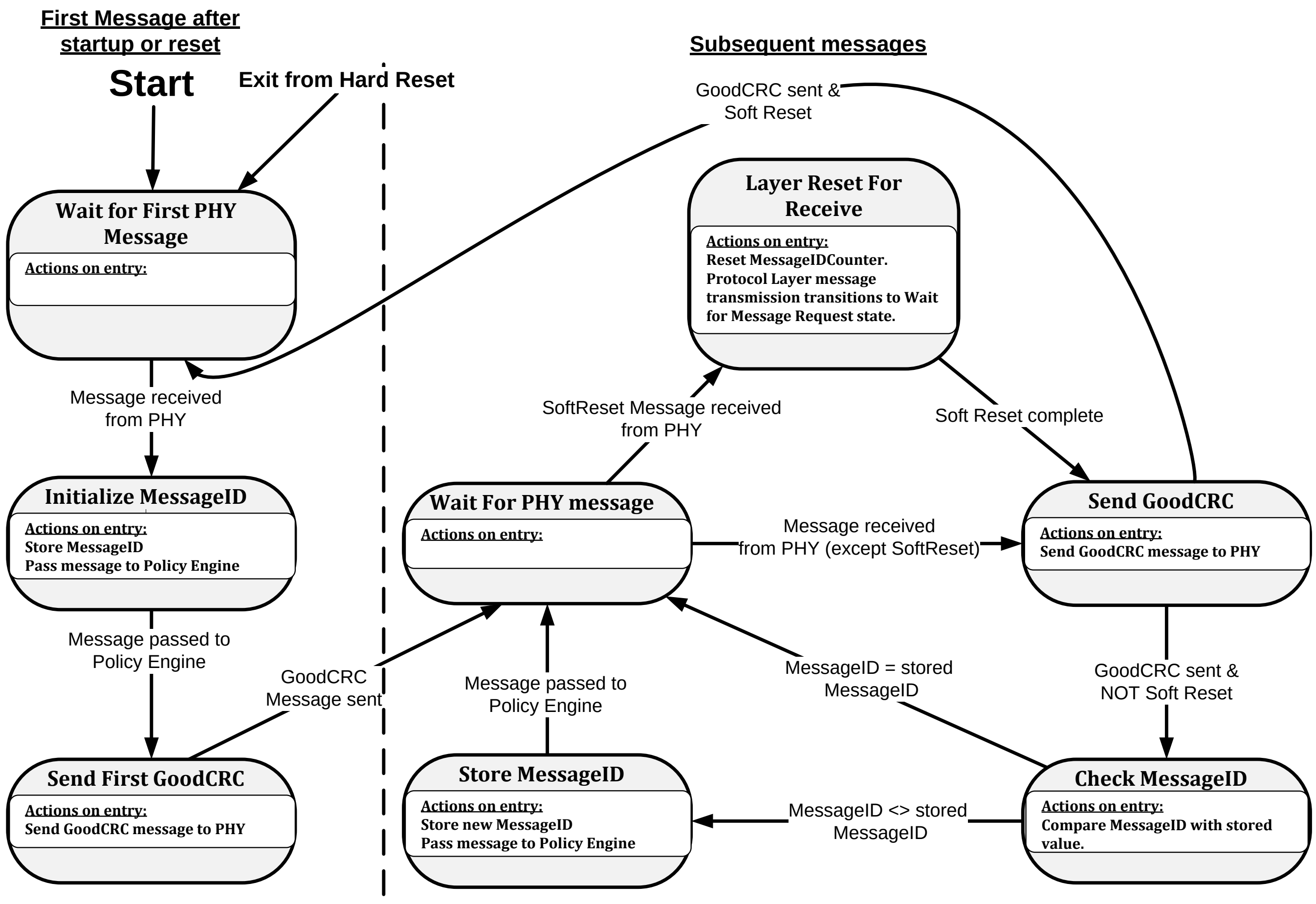

First Message after startup or reset
Start
Exit from Hard Reset
Subsequent messages
GoodCRC sent & Soft Reset
Layer Reset For Receive
Actions on entry:
Reset MessageIDCounter.
Protocol Layer message transmission transitions to Wait for Message Request state.
Wait for First PHY Message
Actions on entry:
Message received from PHY
SoftReset Message received from PHY
Soft Reset complete
Initialize MessageID
Actions on entry:
Store MessageID
Pass message to Policy Engine
Wait For PHY message
Actions on entry:
Message received from PHY (except SoftReset)
Send GoodCRC
Actions on entry:
Send GoodCRC message to PHY
Message passed to Policy Engine
GoodCRC Message sent
Message passed to Policy Engine
MessageID = stored MessageID
GoodCRC sent & NOT Soft Reset
Send First GoodCRC
Actions on entry:
Send GoodCRC message to PHY
Store MessageID
Actions on entry:
Store new MessageID
Pass message to Policy Engine
MessageID <> stored MessageID
Check MessageID
Actions on entry:
Compare MessageID with stored value.

Figure 6-8 Protocol layer message reception

6.8.2.2.1 Wait for First PHY Message state

In the ***Wait for First PHY Message*** state Protocol Layer waits until the PHY Layer passes up a received message. This shall be the startup state for the Protocol Layer message reception and shall also be the state Protocol Layer message reception enters after completion of a Soft Reset or Hard Reset.

The Protocol Layer shall transition to the ***Initialize MessageID*** state when:

- A received message is passed up from the PHY Layer.

6.8.2.2.2 Initialize MessageID state

On entry to the ***Initialize MessageID*** state the Protocol Layer shall store the ***MessageID*** of the incoming message and then pass the message to the Policy Engine.

The Protocol Layer shall transition to the ***Send First GoodCRC*** state when:

- The message has been passed to the Policy Engine.

6.8.2.2.3 Send First GoodCRC state

On entry to the ***Send First GoodCRC*** state the Protocol Layer shall construct ***GoodCRC*** message and request the PHY to transmit it.

The Protocol Layer shall transition to the ***Wait for PHY Message*** state when:

- The ***GoodCRC*** message has been passed to the PHY Layer.

6.8.2.2.4 Wait For PHY Message state

In the ***Wait For PHY Message*** state the Protocol Layer waits until the PHY Layer passes up a received message.

The Protocol Layer shall transition to the ***Send GoodCRC*** state when:

- A message is passed up from the PHY Layer.

The Protocol Layer shall transition to the ***Layer Reset for Receive*** state when:

- A ***Soft Reset*** message is received from the PHY.

6.8.2.2.5 Layer Reset for Receive State

On entry to the ***Layer Reset for Receive*** state the Protocol Layer shall reset the ***MessageIDCounter***. The Protocol Layer shall transition Protocol Layer message transmission to the ***Wait Message Request*** state (see Section 6.8.2.1.1).

The Protocol Layer shall transition to the ***Send GoodCRC*** State when:

- The Soft Reset actions in this state have been completed.

6.8.2.2.6 Send GoodCRC state

On entry to the ***Send GoodCRC*** state the Protocol Layer shall construct a ***GoodCRC*** message and request the PHY to transmit it.

The Protocol Layer shall transition to the ***Check MessageID*** state when:

- The ***GoodCRC*** message has been passed to the PHY Layer and
- A Soft Reset is NOT being performed.

The Protocol Layer shall transition to the ***Wait For First PHY Message*** state when:

- The ***GoodCRC*** message has been passed to the PHY Layer and
- A Soft Reset is being performed.

6.8.2.2.7 Check MessageID state

On entry to the ***Check MessageID*** state the Protocol Layer shall compare the ***MessageID*** of the received message with its stored value.

The Protocol Layer shall transition to the ***Wait For PHY Message*** state when:

- The ***MessageID*** of the received message equals the stored ***MessageID*** value since this is a message retry which shall be discarded.

The Protocol Layer shall transition to the ***Store MessageID*** state when:

- The ***MessageID*** of the received message does not equal the stored ***MessageID*** value since this is a new message.

6.8.2.2.8 Store MessageID

On entry to the ***Store MessageID*** state the Protocol Layer shall replace the stored value of ***MessageID*** with the value of ***MessageID*** in the received message and pass the message up to the Policy Engine.

The Protocol Layer shall transition to the ***Wait for PHY Message*** state when:

- The message has been passed up to the Policy Engine.

6.8.3 Hard Reset operation

Figure 6-9 shows the state behavior for the Protocol layer when receiving a Hard Reset request from the Policy Engine or ***Hard Reset*** signaling from the Physical Layer (see also Section 6.7.2).

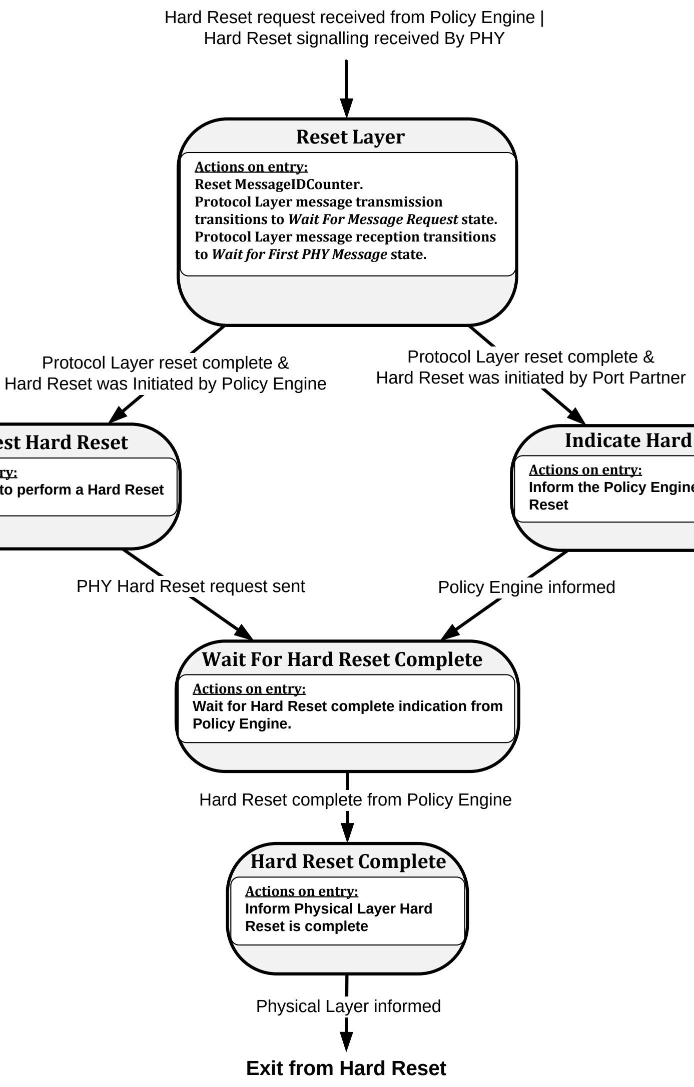

Hard Reset request received from Policy Engine | Hard Reset signalling received By PHY
Reset Layer
Actions on entry:
Reset MessageIDCounter.
Protocol Layer message transmission transitions to Wait For Message Request state.
Protocol Layer message reception transitions to Wait for First PHY Message state.
Protocol Layer reset complete & Hard Reset was Initiated by Policy Engine
Protocol Layer reset complete & Hard Reset was initiated by Port Partner
Request Hard Reset
Actions on entry:
Request PHY to perform a Hard Reset
Indicate Hard Reset
Actions on entry:
Inform the Policy Engine of the Hard Reset
PHY Hard Reset request sent
Policy Engine informed
Wait For Hard Reset Complete
Actions on entry:
Wait for Hard Reset complete indication from Policy Engine.
Hard Reset complete from Policy Engine
Hard Reset Complete
Actions on entry:
Inform Physical Layer Hard Reset is complete
Physical Layer informed
Exit from Hard Reset

Figure 6-9 Hard Reset

6.8.3.1 Reset Layer State

The Reset Layer State defines the mode of operation of both the Protocol Layer Transmit and Receive state machines during a Hard Reset. During this mode no USB Power Delivery protocol messages are sent or received; only ***Hard Reset*** signaling is present after which the communication channel is assumed to have been disabled by the Physical Layer until completion of the Hard Reset.

The Protocol Layer shall enter the ***Reset Layer*** state from any other state when:

- A Hard Reset Request is received from the Policy Engine or
- ***Hard Reset*** signaling is received from the Physical Layer

On Entry to the ***Reset Layer*** state the Protocol Layer shall reset the ***MessageIDCounter***. It shall also reset the states of the Protocol Layer transmit and receive state machines to their starting points. The Protocol Layer transmit state machine shall transition to the ***Wait for Message Request*** state. The Protocol Layer receive state machine shall transition to the ***Wait For First PHY Message*** state.

The Protocol Layer shall transition to the ***Request Hard Reset*** state when:

- The Protocol Layer's reset is complete and
- The Hard Reset request has originated from the Policy Engine.

The Protocol Layer shall transition to the ***Indicate Hard Reset*** state when:

- The Protocol Layer's reset is complete and
- The Hard Reset request has been passed up from the Physical Layer.

6.8.3.2 Request Hard Reset

On entry to the ***Request Hard Reset*** state the Protocol Layer shall request the Physical Layer to send ***Hard Reset*** signaling.

The Protocol Layer shall transition to the ***Wait For Hard Reset Complete*** state when:

- The Physical Layer ***Hard Reset*** signaling has been sent.

6.8.3.3 Indicate Hard Reset

On entry to the Indicate Hard Reset state the Protocol Layer shall indicate to the Policy Engine that Hard Reset signaling has been received.

The Protocol Layer shall transition to the ***Wait for Hard Reset Complete*** state when:

- The Indication to the Policy Engine has been sent.

6.8.3.4 Wait for Hard Reset Complete

In the ***Wait for Hard Reset Complete*** state the Protocol Layer shall wait for the Policy Engine to indicate that the Hard Reset has been completed.

The Protocol Layer shall transition to the ***Hard Reset Complete*** state when:

- A Hard Reset complete indication is received from the Policy Engine.

6.8.3.5 Hard Reset Complete

On entry to the ***Hard Reset Complete*** state the Protocol Layer shall inform the Physical Layer that the Hard Rest is complete.

The Protocol Layer shall exit from the Hard Reset and return to normal operation when:

- The Physical Layer has been informed that the Hard Reset is complete so that it will re-enable the communications channel.

6.8.4 BIST Operation

6.8.4.1 BIST Transmit Test

Figure 6-10 shows the state behavior for the Protocol layer when in BIST Transmitter test mode and transmitting BIST Transmitter Test Frames. The Protocol Layer changes from normal operation for Protocol Message Transmission (see Section 6.8.2.1) to BIST Transmit Test Mode when directed by the Policy Engine.

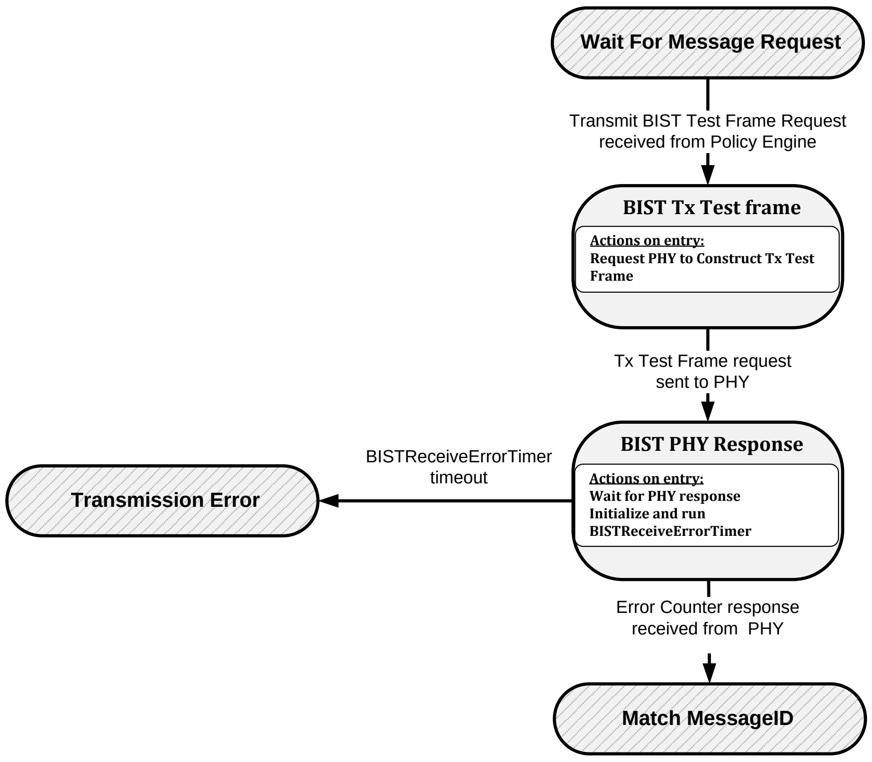

Wait For Message Request
Transmit BIST Test Frame Request received from Policy Engine
BIST Tx Test frame
Actions on entry:
Request PHY to Construct Tx Test Frame
Tx Test Frame request sent to PHY
BIST PHY Response
Actions on entry:
Wait for PHY response
Initialize and run BISTReceiveErrorTimer
BISTReceiveErrorTimer timeout
Transmission Error
Error Counter response received from PHY
Match MessageID

Figure 6-10 BIST Transmitter Test

6.8.4.1.1 BIST Tx Test Frame state

The Protocol Layer shall enter the ***BIST Tx Test Frame*** state from the ***Wait For Message Request*** state (see Section 6.8.2.1.1) when:

- The Policy Engine requests the Protocol Layer to transmit a BIST test frame.

On entry to the ***BIST Tx Test Frame*** state the Protocol Layer shall request the Physical Layer to construct the Test Frame.

The Protocol Layer shall transition to the ***BIST PHY Response*** state when:

- The Test Frame request has been sent to the Physical Layer.

6.8.4.1.2 BIST PHY Response state

In the ***BIST PHY Response*** state the Protocol Layer shall wait for the Physical Layer to provide a response to the Test Frame.

On entry to the ***BIST PHY Response*** state the Protocol Layer shall initialize and run the ***BISTReceiveErrorTimer***.

The Protocol Layer shall transition to the ***Message Sent*** state (see Section 6.8.2.1.5) when:

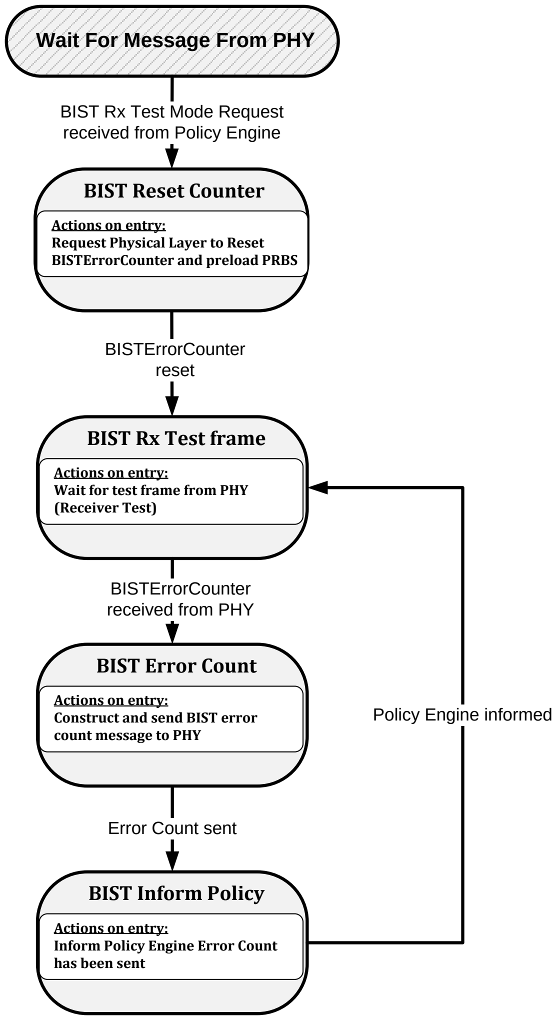
- An error counter response is received from the Physical Layer.

The Protocol Layer shall transition to the ***Transmission Error*** state (see Section 6.8.2.1.8) when:

- The ***BISTReceiveErrorTimer*** times out.

6.8.4.2 BIST Receiver Test

Figure 6-11 shows the state behavior for the Protocol layer when in BIST Receiver test mode and receiving BIST Receiver Test Frames. The Protocol Layer changes from normal operation for Protocol Message Reception (see Section 6.8.2.2) to BIST Receiver Test Mode when directed by the Policy Engine. Exit from this mode of operation is via Hard Reset (see Section 6.8.3).



Wait For Message From PHY
BIST Rx Test Mode Request received from Policy Engine
BIST Reset Counter
Actions on entry:
Request Physical Layer to Reset BISTErrorCounter and preload PRBS
BISTErrorCounter reset
BIST Rx Test frame
Actions on entry:
Wait for test frame from PHY (Receiver Test)
BISTErrorCounter received from PHY
BIST Error Count
Actions on entry:
Construct and send BIST error count message to PHY
Policy Engine informed
Error Count sent
BIST Inform Policy
Actions on entry:
Inform Policy Engine Error Count has been sent

Figure 6-11 BIST Receiver Test

6.8.4.2.1 BIST Reset Counter state

The Protocol Layer shall enter the ***BIST Reset Counter*** state from the ***Wait For Message From PHY*** state when:

A BIST Receiver Test Mode request is received from the Policy Engine.

On entry to the ***BIST Reset Counter*** state the Protocol Layer shall request the Physical Layer to reset the ***BISTErrorCounter*** and preload the PRBS (see Section 5.12.1).

The Protocol Layer shall transition to the ***BIST Rx Test Frame*** state when:

The ***BISTErrorCounter*** has been rest by the Physical Layer.

6.8.4.2.2 BIST Rx Test Frame state

In the ***BIST Rx Test Frame*** State the Protocol Layer shall wait for the Physical Layer to receive the next test frame.

The Protocol Layer shall transition to the ***BIST Error Count*** state when:

The Physical Layer has received the next Test Frame and passes up the current value of the ***BISTErrorCounter***.

6.8.4.2.3 BIST Error Count State

On entry to the ***BIST Error Count*** state the Protocol Layer shall construct a ***BIST*** message with a Data Object of ***Returned BIST Counters*** using the ***BISTErrorCounter*** value returned by the Physical Layer. This ***BIST*** message shall be passed to the Physical Layer for transmission.

The Protocol Layer shall transition to the ***BIST Inform Policy*** state when:

The BIST message has been sent.

6.8.4.2.4 BIST Inform Policy state

On entry to the ***BIST Inform Policy*** state the Protocol Layer shall inform the Policy Engine that the ***BIST*** message containing the ***BISTErrorCounter*** has been sent.

The Protocol Layer shall transition to the ***BIST Rx Test Frame*** state when:

- The Policy Engine has been informed that the ***BIST*** message containing the ***BISTErrorCounter*** has been sent.

7. Power Supply

7.1 Source Requirements

7.1.1 Behavioral Aspects

The Source in the USB Power Delivery system shall exhibit the following behaviors.

- Shall be backward compatible with legacy V_{BUS} ports.
- Shall supply the default ***[USB2.0]*** or ***[USB3.0]*** voltage and current to V_{BUS} when the USB cable is attached.
- Shall supply the default ***[USB2.0]*** or ***[USB3.0]*** voltage and current to V_{BUS} when a USB Power Delivery contract does not exist.
- Shall return ***vSafe0V*** for some time then return to ***vSafe5V*** when a ***Hard Reset*** signaling is received.
- Shall control V_{BUS} voltage transitions as bound by undershoot, overshoot and transition time requirements.

7.1.2 Source Bulk Capacitance

The Source shall have a bulk capacitance located between the output of the power supply and the transceiver isolation impedance as shown in Figure 7-1. The Source bulk capacitance shall not be placed between the transceiver isolation impedance and the USB receptacle. The bulk capacitance consists of C1 and C2 as shown in Figure 7-1. The Ohmic Interconnect may consist of PCB traces for power distribution or power switching devices. The capacitance may be a single capacitor, a capacitor bank or distributed capacitance. If the power supply is shared across multiple ports, bulk capacitance is defined as ***cSrcBulkShared***. If the power supply is dedicated to a single port, the minimum bulk capacitance is defined as ***cSrcBulk***.

The Source bulk capacitance is allowed to change for a newly negotiated power level. The capacitance change shall occur before the Source is ready to operate at the new power level. During a Role Swap, the Default Source shall transition to Swap Standby before operating as the new Sink. Any change in bulk capacitance required to complete the Role Swap shall occur during Swap Standby.

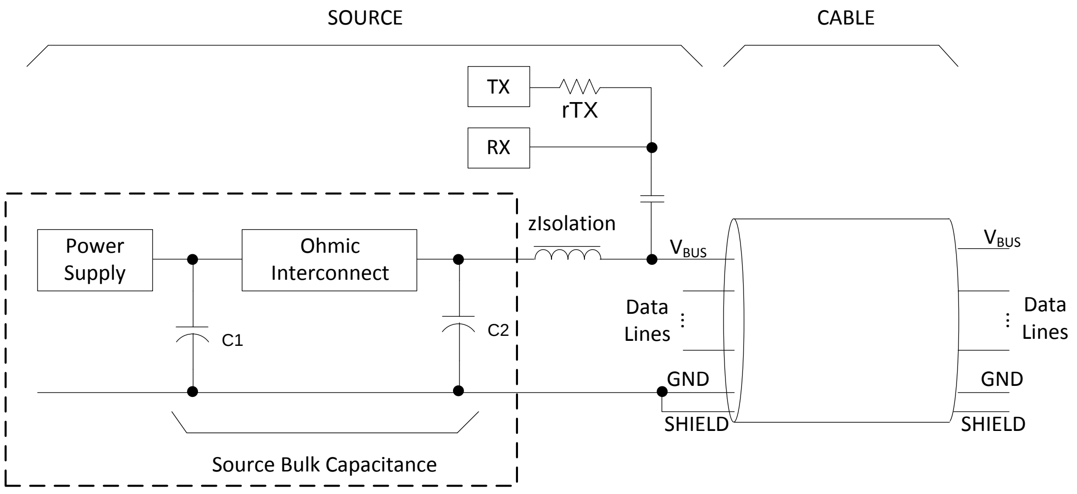


Figure 7-1 Placement of Source Bulk Capacitance

7.1.3 Types of Sources

Consistent with the Power Data Objects discussed in Section 6.4.1.2, the three possible power supply types that are available as Sources in a USB Power Delivery System are:

- Fixed Supply is used to expose well regulated fixed voltage power supplies. Sources shall support at least one fixed power source capable of supplying ***vSafe5V***. The output voltage of a Fixed Supply shall be specified in terms of an absolute tolerance, ***tolSrc,*** relative to the nominal value. Refer to Table 7-22 for the output voltage tolerance specification.

- Variable Power Supply (non-battery) is used to expose very poorly regulated Sources. The output voltage of a Variable Power Supply (non-battery) shall be specified in terms of an absolute maximum output voltage and an absolute minimum output voltage.
- Battery is used to expose batteries than can be connected directly as a Source to V_{BUS}. The output voltage of a Battery shall be expressed in terms of an absolute maximum output voltage and an absolute minimum output voltage.

7.1.4 Positive Voltage Transitions

The Source shall transition V_{BUS} from the old voltage to the higher new voltage in a controlled manner. During the positive transition the Source shall be able to supply the Sink standby power and the transient current to charge the total bulk capacitance on V_{BUS}. The positive transition shall not occur faster than ***tTranFastPos***. The Source output voltage during positive transitions shall settle within the Source output tolerance window ***tolSrc*** by ***tSrcSettlePos***. The Source shall be able to supply the newly negotiated power level at the new voltage by ***tSrcNewPower***. Positive voltage transitions shall remain within the tolerance window specified in Figure 7-2. The lower bound of the transition window is ***tolTranLowerPos*** and the upper bound is ***tolTranUpperPos***. If the old voltage is lower than ***tolTranUpperPos*** then only ***tolTranLowerPos*** shall apply to the positive transition.

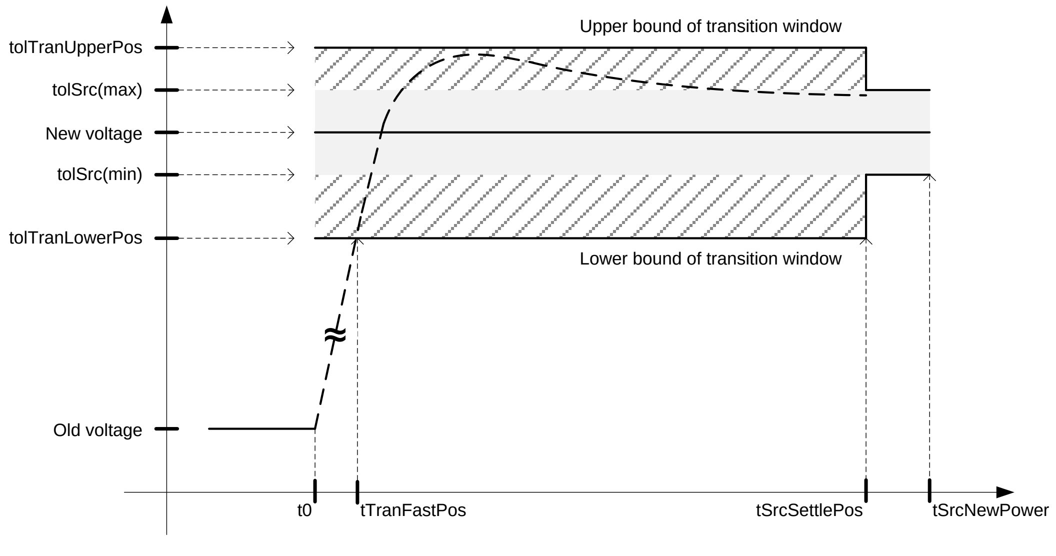


Figure 7-2 Transition Envelope for Positive Voltage Transitions

7.1.5 Negative Voltage Transitions

The Source shall transition V_{BUS} from the old voltage to the lower new voltage in a controlled manner. During the negative transition the Source shall be able to supply the Sink standby current and the transient current to discharge the total bulk capacitance on V_{BUS}. The negative transition times shall not occur faster than ***tTranFastNeg***. The transitioning output voltage shall settle within the Source output tolerance window ***tolSrc*** by ***tSrcSettleNeg***. The Source shall be able to supply the newly negotiated power level at the new voltage by ***tSrcNewPower***. Negative voltage transitions shall remain within the tolerance window specified in Figure 7-3. The lower bound of the transition window is ***tolTranLowerNeg*** and the upper bound is ***tolTranUpperNeg.*** If the old voltage is lower than ***tTranFastNeg*** then only ***tTranFastNeg*** shall apply to the negative transition.

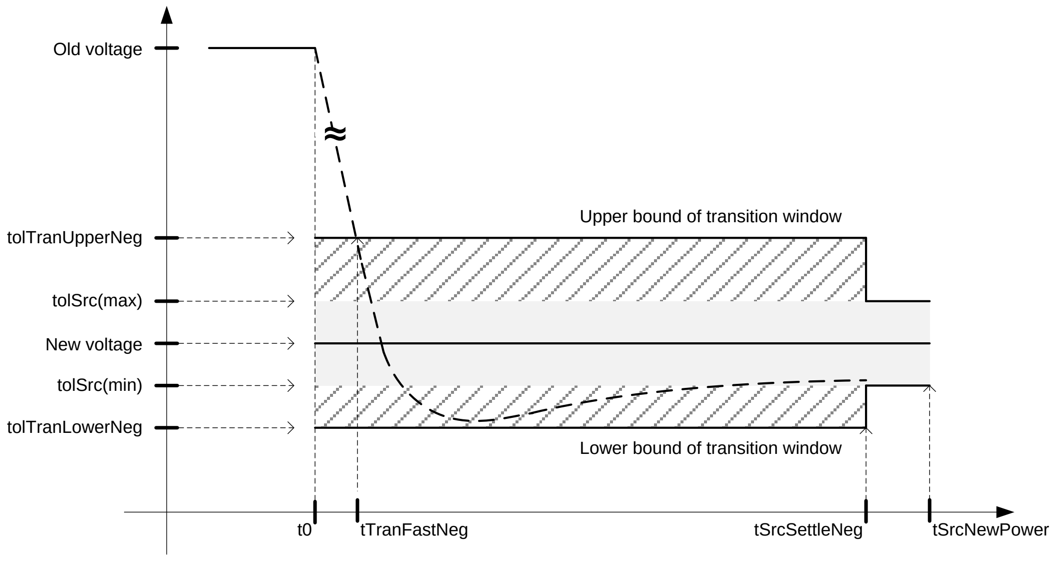


Figure 7-3 Transition Envelope for Negative Voltage Transitions

7.1.6 Response to Hard Resets

Hard Reset signaling indicates a communication failure has occurred and the Source shall revert to a safe operating voltage as specified in Figure 7-4. The safe operating voltage is specified as ***vSafe0V***. The USB connection may reset during a Hard Reset since the V_{BUS} voltage will be less than ***vSafe5V*** for an extended period of time. After establishing the safe voltage condition on V_{BUS}, the power supply shall wait ***tSrcRecover*** before powering V_{BUS} to ***vSafe5V***. Device operation during and after a Hard Reset is defined as follows:

- Self-powered devices should not disconnect from USB during a Hard Reset.
- Self-powered devices operating at more than 5V may not maintain full functionality after a ***Hard Reset***.
- Bus powered devices will disconnect from USB during a Hard Reset due to the loss of their power source.

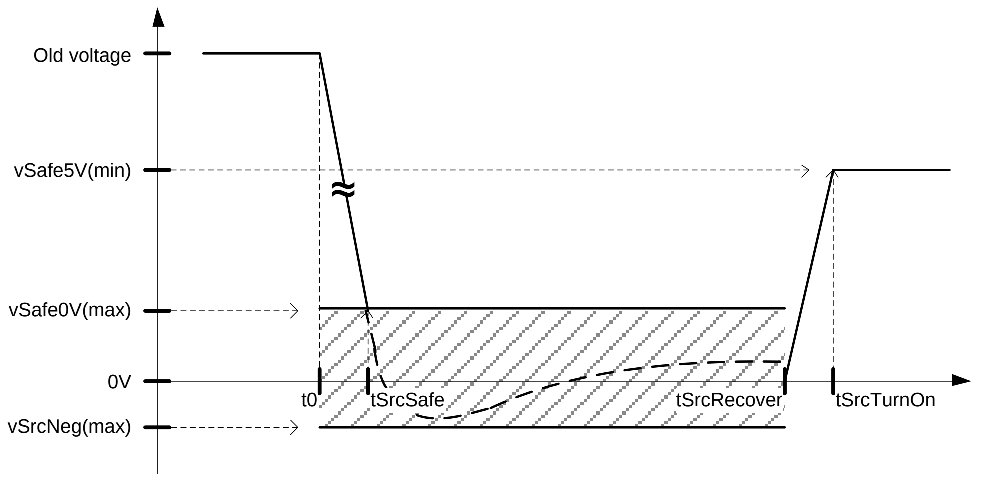


Figure 7-4 Source Response to Hard Reset

7.1.7 Changing the Output Power Capability

Some USB Power Delivery negotiations require the Source to adjust its output power capability without changing the output voltage. The Source shall be able to supply a higher or lower load current within ***tSrcPwrRdy***. This time is much faster than the positive and negative voltage transition times.

7.1.8 Transceiver Isolation Impedance

An isolation impedance as specified in Section 5.2.2 is used to isolate the transceiver from the capacitive loading of the Source. The DC component of the isolation impedance introduces additional voltage loss between the Source power path and the connector. The Source output impedance, the total V_{BUS} bulk capacitance, the transceiver isolation impedance(s), the USB cable and the Sink network creates a complex output isolation impedance that should be taken into consideration when designing the Source. Refer to Appendix C more information on this topic.

7.1.9 Noise Injected on V_{BUS}

The Source shall not interfere with the USB Power Delivery FSK waveform that is transmitted on V_{BUS}. Power stage switching harmonics or poor layout and component selection may result in a significant amount of in-band noise injected on V_{BUS}. The characteristics of the noise will be implementation specific and in some cases an additional filter may be required to meet the in-band signal-to-noise ratio requirement of ***snrSrc***. Refer to Section 5.2.2 and Section 5.10.1.1 for the reference impedance, carrier signal amplitude and transmission bandwidth of the USB Power Delivery FSK waveform. Refer to Appendix C for more information on this topic.

Out-of-band noise (including spurs) can also affect the reception of the FSK signal; therefore the Source shall limit its out-of-band noise below the spectrum shown in Figure 7-5 and the levels indicated in Table 7-1. Figure 7-5 shows the level of allowed noise relative to the level of the allowed in-band noise [for a Source: dbV(minimum value of ***vTX***) – ***snrSrc***]. The limits shown in Figure 7-5 and Table 7-1 do not include any other regulations (such as FCC regulations) that govern signal emissions.

The V_{BUS}-to-GND noise measurement shall be made at the connector with the unit terminated in 50Ω (ac coupled). The analyzer receiver shall be set as follows:

- Resolution bandwidth of 100kHz
- Peak detection mode
- The sweep period or number of sweeps shall be set to detect the maximum noise present.

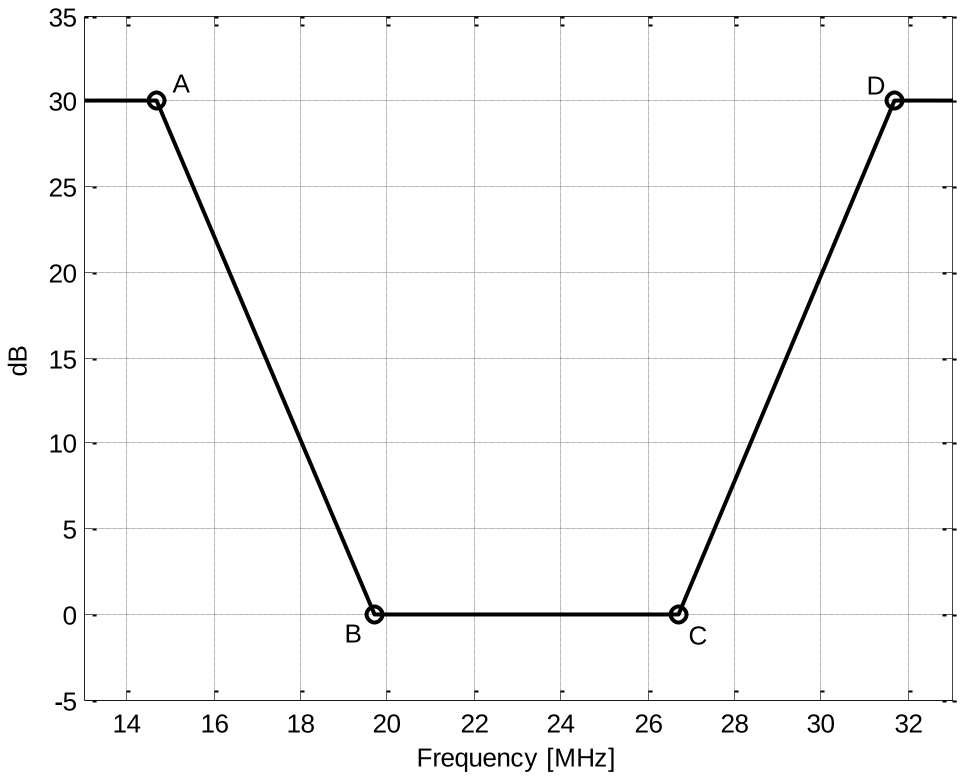


Figure 7-5 Noise spectral mask for Source out-of-band noise

Table 7-1 Noise spectral mask for Source out-of-band noise

Frequency Range [MHz]	Allowed gain relative to in-band noise level limit [dB]
0.01 ≤ Frequency < 14.7 (A)	≤30
14.7 (A) ≤ Frequency < 19.7 (B)	≤ 118.2 – 6*Frequency
19.7 (B) ≤ Frequency < 26.7 (C)	0
26.7 (C) ≤ Frequency < 31.7 (D)	≤ 6*Frequency – 160.2
31.7 (D) ≤ Frequency ≤ 100	≤30

7.1.10 Short Circuit Current Limiting

The Source shall provide short circuit current limiting to protect its port from prolonged exposure to excessive current draw. For the purposes of this standard, excessive current draw is defined as any current that exceeds the output capability of the Power Supply or the contact rating of the receptacle. The maximum steady state operating current of the Source shall be the lesser of these two values. For example, the Source port powered by a 6A power supply connected to two 3A receptacles will provide a maximum continuous operating current of 3A on each receptacle. The short circuit protection mechanism shall be designed to avoid tripping at the maximum continuous operating current of the Source port and shall not interfere with negotiated current levels. The protected port shall recover automatically, by performing a Hard Reset, when the fault is removed and resume operation at ***vSafe5V***. Mechanical or user intervention shall not be required to reset the short circuit protection mechanism. If a Provider is purely a power source, which does not support USB communication, then mechanical intervention may be used.

Safe Operation mandates that Power Delivery Sources shall be tolerant of 5V being present on V_{BUS} when simultaneously applying power to V_{BUS}. This shall not interfere with normal PD communication.

7.1.11 Output Voltage Tolerance

The Source output voltage tolerance, ***tolSrc***, shall be measured at the connector receptacle. The output tolerance includes load regulation, transient response and output ripple. The output voltage tolerance does not include undershoot and overshoot that may occur during voltage transitions. In some systems it may be necessary to design the Source to compensate for the voltage drop between the output stage of the power supply electronics and the receptacle contact. The determination of when compensation may be necessary is left to the discretion of the implementation. Refer to Table 7-22 for the output voltage tolerance specifications that shall be used for the Source.

7.1.12 Charging and Discharging the Bulk Capacitance on V_{BUS}

The Source shall charge and discharge the bulk capacitance on V_{BUS} whenever the Source voltage is negotiated to a different value. The charging or discharging occurs during the voltage transition and shall not interfere with the Source's ability to meet ***tSrcNewPower***.

7.1.13 Swap Standby for Sources

Sources and Sinks of a Dual-Role port shall support Swap Standby. Swap Standby occurs for the Source after the Source power supply has discharged the bulk capacitance on V_{BUS} as part of the Role Swap transition. While in Swap Standby the Source shall not drive V_{BUS}, the Dual-Role port shall be configured as a Sink and the USB connection shall not reset even though ***vSafe5V*** is no longer present on V_{BUS}. The ***PS_RDY*** message associated with the Source being in Swap Standby shall be sent after the V_{BUS} drive is removed and the new Sink power path is ready to consume power. The time for the Source to transition to Swap Standby shall not exceed ***tSrcSwapStdby***. Upon entering Swap Standby the Source has relinquished its role as Source and is ready to become the new Sink. The transition time from Swap Standby to being the new Sink shall be no more than ***tNewSnk***.

7.1.14 Dead Battery Operation

Dead Battery operation is only supported by Provider/Consumers that want to perform an implicit (i.e. non-negotiated) Role Swap when attached to a Consumer/Provider. When a Provider/Consumer port is not able to Source power (or has chosen to not source power) it shall discharge its port voltage to ***vSafe0V*** and be ready to power up the transceiver if ***vSafeDB*** is applied to its port. When operating from ***vSafeDB***, the Provider/Consumer power consumption shall remain within the region labeled PD Active in Figure 7-8. The Provider/Consumer may regain the ability to Source power (or decide to do so) at any time and shall not apply power to its port when ***vSafeDB*** is present. A detailed description of the Provider/Consumers behavior during Dead Battery operation is provided in Sections 4.1 and 7.2.9.

7.2 Sink Requirements

7.2.1 Behavioral Aspects

The Sink in the USB Power Delivery system shall exhibit the following behaviors.

- Shall be backward compatible with legacy V_{BUS} ports.
- Shall draw the default ***[USB2.0]*** or ***[USB3.0]*** V_{BUS} current when the USB cable is attached.
- Shall draw the default ***[USB2.0]*** or ***[USB3.0]*** V_{BUS} current when a USB Power Delivery contract does not exist.
- Shall return to the default ***[USB2.0]*** or ***[USB3.0]*** V_{BUS} when responding to a Hard Reset.
- Shall control V_{BUS} in-rush current when increasing and decreasing current consumption.

7.2.2 Sink Bulk Capacitance

The Sink shall have a bulk capacitance, ***cSnkBulk***, located between the input of the power supply and the transceiver isolation impedance as shown in Figure 7-6. The Sink bulk capacitance shall not be placed between the transceiver isolation impedance and the USB receptacle. The bulk capacitance consists of C3 and C4 as shown in Figure 7-6. The Ohmic Interconnect may consist of PCB traces for power distribution or power switching devices. The capacitance may be a single capacitor, a capacitor bank or distributed capacitance. An upper bound of ***cSnkBulkPd*** shall not be exceeded so that the transient charging, or discharging, of the total bulk capacitance on V_{BUS} can be accounted for during voltage transitions.

The Sink bulk capacitance is allowed to change to support a newly negotiated power level. The capacitance can be changed when the Sink enters Sink Standby or during a voltage transition or when the Sink begins to operate at the new power level. Regardless of when the change occurs, the capacitance change shall occur in such a manner that does not introduce a V_{BUS} transient current greater than ***iCapChange***. During a Role Swap the Default Sink shall transition to Swap Standby before operating as the new Source. Any change in bulk capacitance required to complete the Role Swap shall occur during Swap Standby.

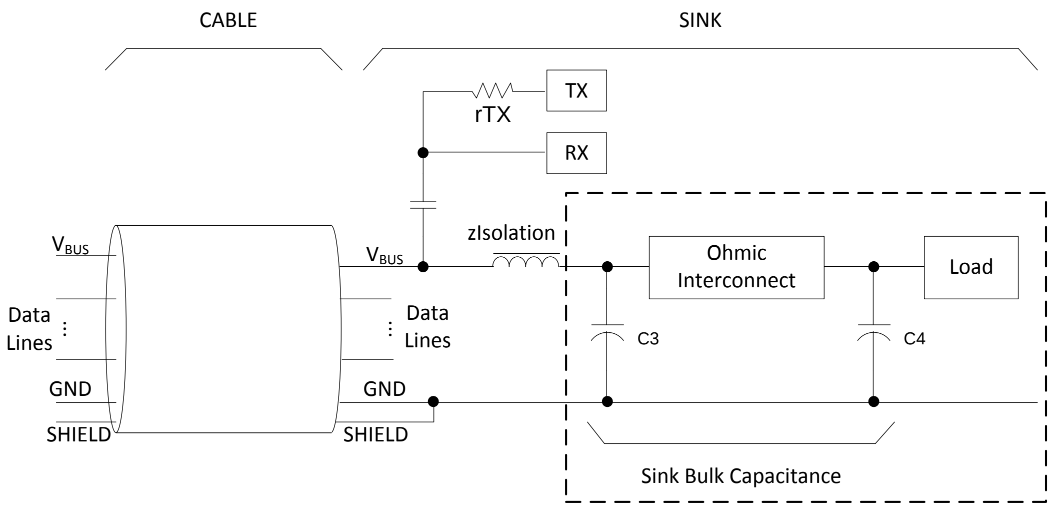


Figure 7-6 Placement of Sink Bulk Capacitance

7.2.3 Sink Standby

The Sink shall transition to Sink Standby before a positive or negative voltage transition of V_{BUS}. The Sink shall reduce its power draw during Sink Standby to ***pSnkStdby*** to allow for the Source to manage the voltage transition as well as supply sufficient operating current to the Sink to maintain PD operation during the transition. The Sink shall complete this transition to Sink Standby within ***tSnkStdby***. The transition when returning to Sink from Sink Standby shall be completed within ***tSnkRtn***. The ***pSnkStdby*** requirement shall only apply if the Sink power draw is higher than this level.

7.2.4 Suspend Power Consumption

When a Source negotiates with a Sink to go to zero current the Sink shall go to the lowest power state, ***pSnkSusp or lower***, where it can still communicate over V_{BUS}. There is no requirement for the Source voltage to be changed during suspend.

7.2.5 In-rush Current Limiting

The Sink shall control the rate of change of current draw when transitioning between negotiated levels, when transitioning to Sink Standby and when transitioning to a newly negotiated current level. The in-rush current rates for increasing and decreasing current shall be defined by ***iInRushPos*** and ***iInRushNeg***.

7.2.6 Transceiver Isolation Impedance

An isolation impedance as specified in Section 5.2.2 is used to isolate the transceiver from the capacitive loading of the Sink. The DC component of the isolation impedance introduces additional voltage loss between the connector and the Sink power path. The Sink input impedance, the total V_{BUS} bulk capacitance, the transceiver isolation impedance(s), the USB cable and the Source creates a complex input isolation impedance that should be taken into consideration when designing the Sink. Refer to Appendix C.1 for more information on this topic.

7.2.7 Noise Reflected on V_{BUS}

The Sink shall not interfere with the USB Power Delivery FSK waveform that is transmitted on V_{BUS}. Power stage switching harmonics or poor layout and component selection may result in a significant amount of in-band noise reflected on V_{BUS}. The characteristics of the noise will be implementation specific and in some cases an additional filter may be required to meet the in-band signal-to-noise ratio requirement of ***snrSnk***. Refer to Section 5.2.2 and Section 5.10.1.1 for the reference impedance, carrier signal amplitude and transmission bandwidth of the USB Power Delivery FSK waveform. Refer to Appendix C.1.1 for more information on this topic.

Out-of-band noise (including spurs) can also affect the reception of the FSK signal; therefore the Sink shall limit its out-of-band noise below the spectrum shown in Figure 7-7 and as detailed in Table 7-2. Figure 7-7 also shows the noise level that shall be allowed relative to the level of the allowed in-band noise [for a Sink: dbV(minimum value of ***vTX***) – ***snrSnk***]. The limits shown in Figure 7-7 and Table 7-2 do not include any other regulations (such as FCC regulations) that govern signal emissions.

The V_{BUS}-to-GND noise measurement shall be made at the connector with the unit terminated in 50Ω (ac coupled). The analyzer receiver shall be set as follows:

- Resolution bandwidth of 100kHz
- Peak detection mode
- The sweep period or number of sweeps shall be set to detect the maximum noise present.

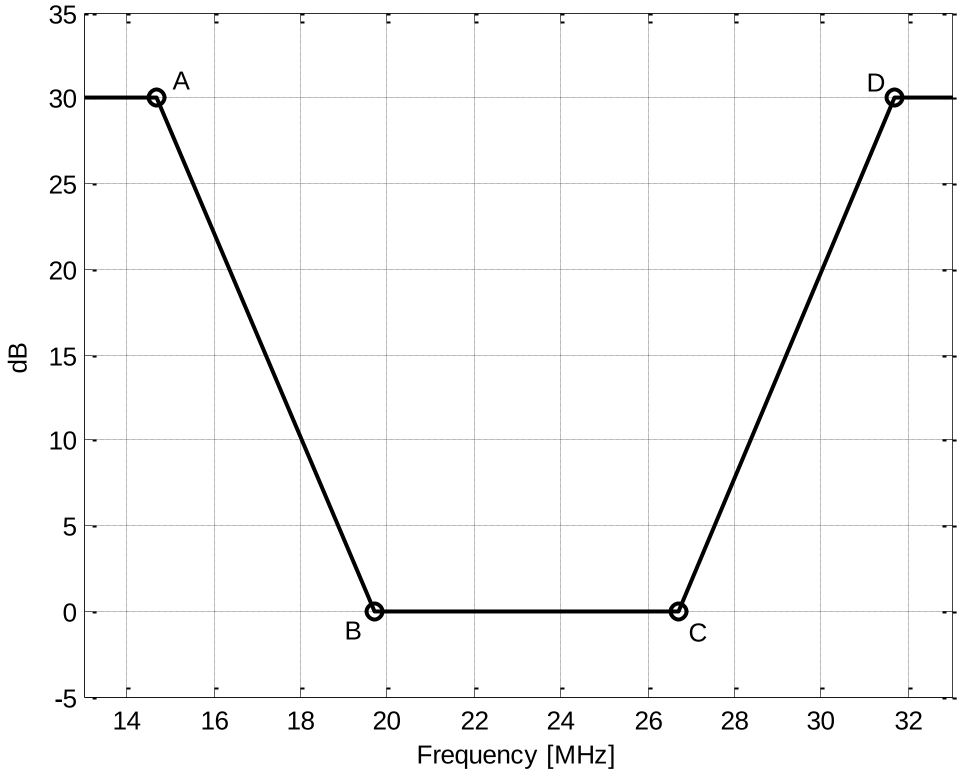


Figure 7-7 Noise spectral mask for Sink out-of-band noise

Table 7-2 Noise spectral mask for Sink out-of-band noise

Frequency Range [MHz]	Allowed gain relative to in-band noise level limit [dB]
0.01 ≤ Frequency < 14.7 (A)	≤30
14.7 (A) ≤ Frequency < 19.7 (B)	≤ 118.2 – 6*Frequency
19.7 (B) ≤ Frequency < 26.7 (C)	0
26.7 (C) ≤ Frequency < 31.7 (D)	≤ 6*Frequency – 160.2
31.7 (D) ≤ Frequency ≤ 100	≤30

7.2.8 Swap Standby for Sinks

The Sink's capability in a Dual-Role port shall support Swap Standby. Swap Standby occurs for the Sink after evaluating the ***Accept*** message from the Source during a Role Swap negotiation. While in Swap Standby the Sink shall not draw power from V_{BUS}, the Dual-Role port shall be configured as a Source and the Sink's USB connection should not be reset even though ***vSafe5V*** is not present on the V_{BUS} conductor. The time for the Sink to transition to SwapStandby shall be no more than ***tSinkTransition***. When in Swap Standby the Sink has relinquished its role as Sink and will prepare to become the new Source. The transition time from Swap Standby to new Source shall be no more than ***tNewSRC***. The Sink current draw shall not exceed ***iSnkSwapStdby*** during Swap Standby.

7.2.9 Dead Battery Operation

Dead Battery operation shall be supported by all Consumer/Providers. A Consumer/Provider shall apply ***vSafeDB*** to its port when it detects its port voltage to be ***vSafe0V*** (see Section 4.1). The load line characteristic of ***vSafeDB*** source shall be as shown in Figure 7-8. A detailed description of the Consumer/Provider's behavior during Dead Battery operation is provided in Section 8.3.3.6.7.

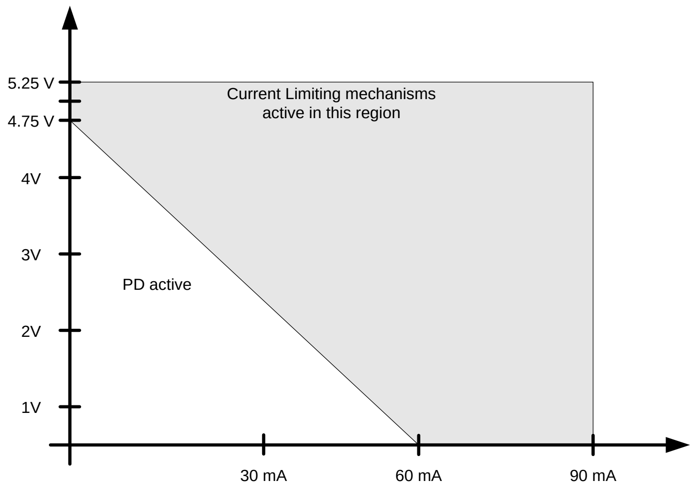


Figure 7-8 vSafeDB Load Line

7.3 Transitions

The following sections illustrate the Power Supply's response to various types of negotiations. The negotiation cases take into consideration for the examples are as follows:

- Higher Power Transitions
  - Increase the current
  - Increase the voltage
  - Increase the voltage and the current
- Relatively Constant Power Transitions
  - Increase the voltage and decrease the current
  - Decrease the voltage and increase the current
- Lower Power Transitions
  - Decrease the current
  - Decrease the voltage
  - Decrease the voltage and the current
- Role Swap Transitions
  - Source requests a Role Swap
  - Sink requests a Role Swap
- Go To Minimum Current Transition
  - Response to ***Hard Reset*** signaling
  - Source issues ***Hard Reset*** signaling
  - Sink issues ***Hard Reset*** signaling
  - New Source issues ***Hard Reset*** signaling and New Sink Receives ***Hard Reset*** signaling.
  - New Source issues ***Hard Reset*** signaling and New Sink Does Not Receive ***Hard Reset*** signaling.
  - New Sink issues ***Hard Reset*** signaling and New Source Receives ***Hard Reset*** signaling.

   - New Sink issues ***Hard Reset*** signaling and New Source Does Not Receive ***Hard Reset*** signaling.
- Dead Battery Operation.

7.3.1 Increasing the Current

The interaction of the System Policy, Device Policy, and Power Supply that shall be followed when increasing the current is shown in Figure 7-9. The sequence that shall be followed is described in Table 7-3. The timing parameters that shall be followed are listed in Table 7-21 and Table 7-22.

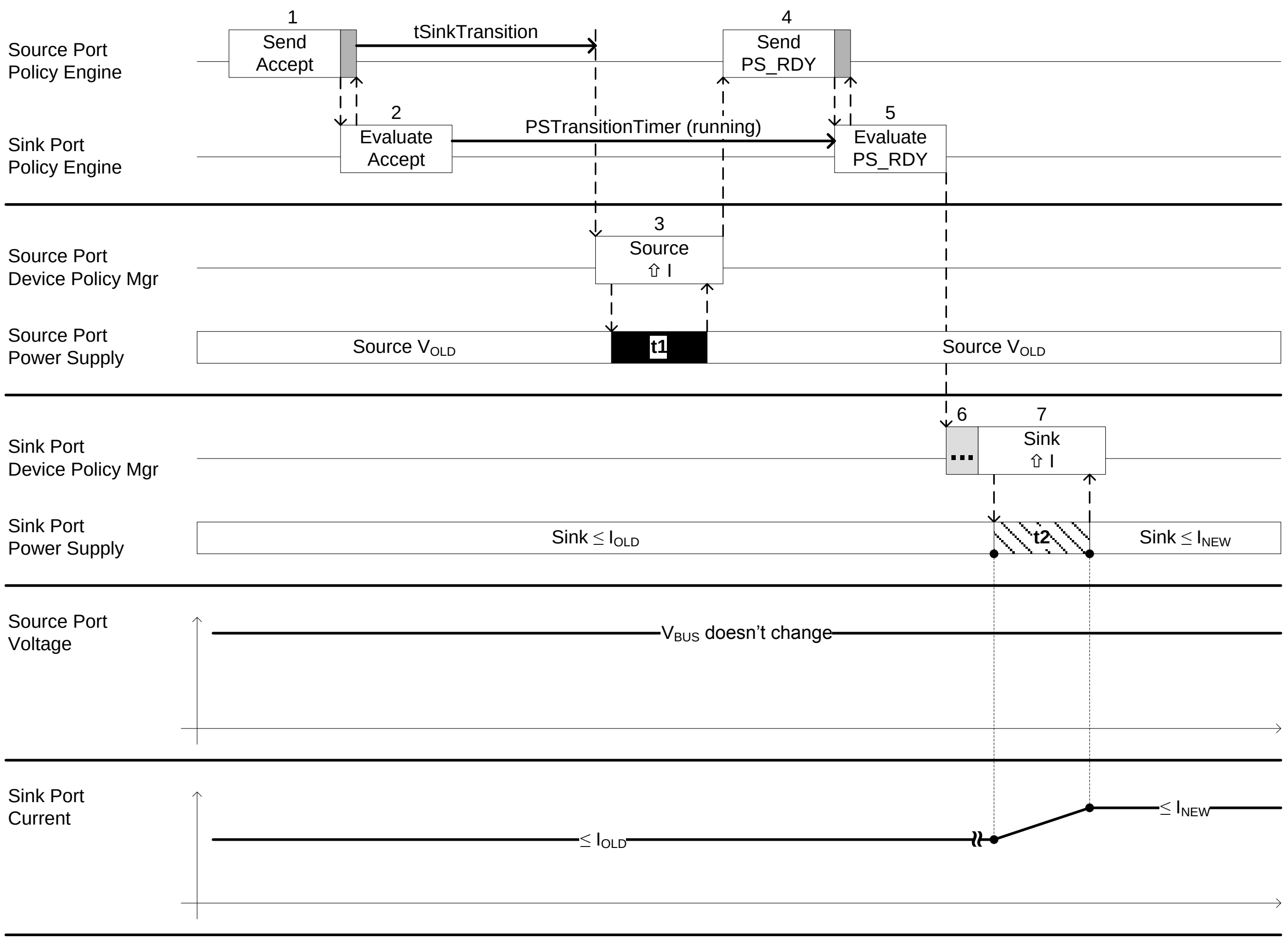


Figure 7-9 Transition Diagram for Increasing the Current

Table 7-3 Sequence Description for Increasing the Current

Step	Source	Sink
1	Policy Engine sends the ***Accept*** message to the Sink. Policy Engine receives the ***GoodCRC*** message from the Sink. The Policy Engine waits ***tSinkTransition*** before telling the Device Policy Manager to instruct the Power Supply to modify its output power.	Policy Engine sends the ***GoodCRC*** message to the Source.
2		Policy Engine evaluates the ***Accept*** message and starts ***PSTransitionTimer***.

Step	Source	Sink
3	After ***tSinkTransition***, the Policy Engine tells the Device Policy Manager to instruct the Power Supply to change its output power capability. The Power Supply shall be ready to operate at the new power level within ***tSrcNewPower*** (t1). The Power Supply informs the Device Policy Manager that it is ready to operate at the new power level. The Power Supply status is passed to the Policy Engine.	
4	The Policy Engine sends the ***PS_RDY*** message to the Sink. Policy Engine receives the ***GoodCRC*** message from the Sink.	Policy Engine sends the ***GoodCRC*** message to the Source.
5		Policy Engine evaluates the ***PS_RDY*** message from the Source and tells the Device Policy to instruct the Power Supply it is okay to operate at the new power level.
6		The Sink may begin operating at the new power level any time after having evaluated the ***PS_RDY*** message. This time duration is indeterminate.
7		The Sink shall not violate the in-rush current specfications ***iInRushPos*** and ***iInRushNeg*** when transitioning to the new power level and shall complete the transition within ***tSnkNewPower*** (t2).

7.3.2 Increasing the Voltage

The interaction of the System Policy, Device Policy, and Power Supply that shall be followed when increasing the voltage is shown in Figure 7-10. The sequence that shall be followed is described in Table 7-4. The timing parameters that shall be followed are listed in Table 7-21, Table 7-22, and Table 7-23.

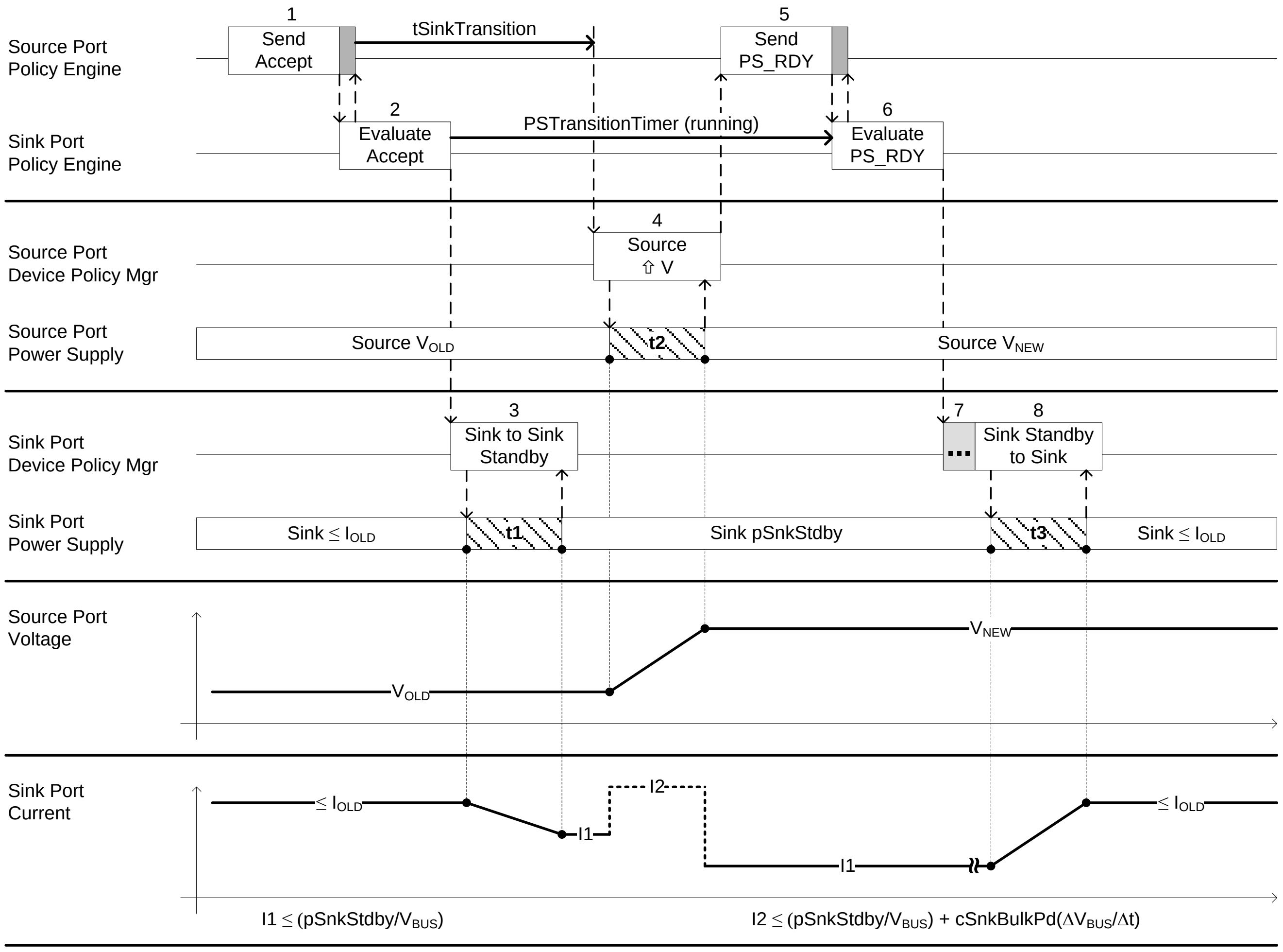


Figure 7-10 Transition Diagram for Increasing the Voltage

Table 7-4 Sequence Description for Increasing the Voltage

Step	Source	Sink
1	Policy Engine sends the ***Accept*** message to the Sink. Policy Engine receives the ***GoodCRC*** message from the Sink. The Policy Engine waits ***tSinkTransition*** before telling the Device Policy Manager to instruct the Power Supply to modify its output power.	Policy Engine sends the ***GoodCRC*** message to the Source.
2		Policy Engine evaluates the ***Accept*** message and starts ***PSTransitionTimer***.
3		Policy Engine tells the Device Policy Manager to instruct the Power Supply to reduce power

Step	Source	Sink
		consumption to **pSnkStdby** within **tSnkStdby** (t1).
4	After ***tSinkTransition***, the Policy Engine tells the Device Policy Manager to instruct the Power Supply to change its output voltage to operate at the new power level. The Power Supply shall be ready to operate at the new power level within ***tSrcRdyPos***. The Power Supply informs the level within ***tSrcRdyPos*** (t2). The Power Supply informs the Device Policy Manager that it is ready to operate at the new power level. The Power Supply status is passed to the Policy Engine.	
5	The Policy Engine sends the ***PS_RDY*** message to the Sink. Policy Engine receives the ***GoodCRC*** message from the Sink.	Policy Engine sends the ***GoodCRC*** message to the Source.
6		Policy Engine evaluates the ***PS_RDY*** message from the Source and tells the Device Policy to instruct the Power Supply it is okay to operate at the new power level.
7		The Sink may begin operating at the new power level any time after having evaluated the ***PS_RDY*** message. This time duration is indeterminate.
8		The Sink shall not violate the in-rush current specfications **iInRushPos** and **iInRushNeg** when transitioning to the new power level and shall complete the transition within **tSnkNewPower** (t3).

7.3.3 Increasing the Voltage and Current

The interaction of the System Policy, Device Policy, and Power Supply that shall be followed when increasing the voltage and current is shown in Figure 7-11. The sequence that shall be followed is described in Table 7-5. The timing parameters that shall be followed are listed in Table 7-21, Table 7-22, and Table 7-23.

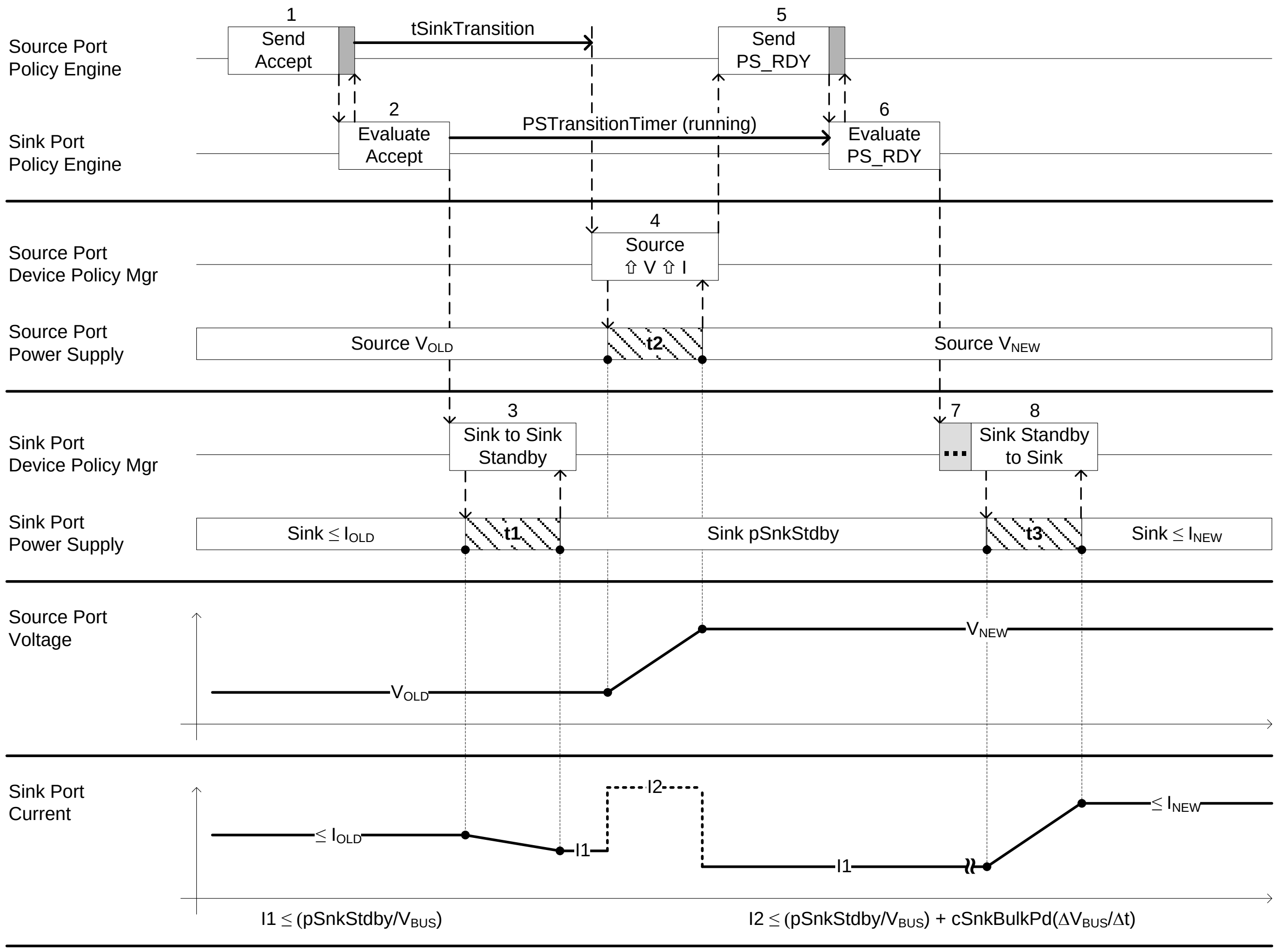


Figure 7-11 Transition Diagram for Increasing the Voltage and Current

Table 7-5 Sequence Diagram for Increasing the Voltage and Current

Step	Source	Sink
1	Policy Engine sends the ***Accept*** message to the Sink Policy Engine receives the ***GoodCRC*** message from the Sink. The Policy Engine waits ***tSinkTransition*** before telling the Device Policy Manager to instruct the Power Supply to modify its output power.	Policy Engine sends the ***GoodCRC*** message to the Source.
2		Policy Engine evaluates the ***Accept*** message and starts ***PSTransitionTimer***.
3		Policy Engine tells the Device Policy Manager to instruct the Power Supply to reduce power

Step	Source	Sink
		consumption to **pSnkStdby** within **tSnkStdby** (t1).
4	After ***tSinkTransition***, the Policy Engine tells the Device Policy Manager to instruct the Power Supply to change its output voltage to operate at the new power level. The Power Supply shall be ready to operate at the new power level within ***tSrcRdyPos*** (t2). The Power Supply informs the Device Policy Manager that it is ready to operate at the new power level. The Power Supply status is passed to the Policy Engine.	
5	The Policy Engine sends the ***PS_RDY*** message to the Sink. Policy Engine receives the ***GoodCRC*** message from the Sink.	Policy Engine sends the ***GoodCRC*** message to the Source.
6		Policy Engine evaluates the ***PS_RDY*** message from the Source and tells the Device Policy to instruct the Power Supply it is okay to operate at the new power level.
7		The Sink may begin operating at the new power level any time after having evaluated the ***PS_RDY*** message. This time duration is indeterminate.
8		The Sink shall not violate the in-rush current specfications **iInRushPos** and **iInRushNeg** when transitioning to the new power level and shall complete the transition within **tSnkNewPower** (t3).

7.3.4 Increasing the Voltage and Decreasing the Current

The interaction of the System Policy, Device Policy, and Power Supply that shall be followed when increasing the voltage and decreasing the current is shown in Figure 7-12. The sequence that shall be followed is described in Table 7-6. The timing parameters that shall be followed are listed in Table 7-21, Table 7-22, and Table 7-23.

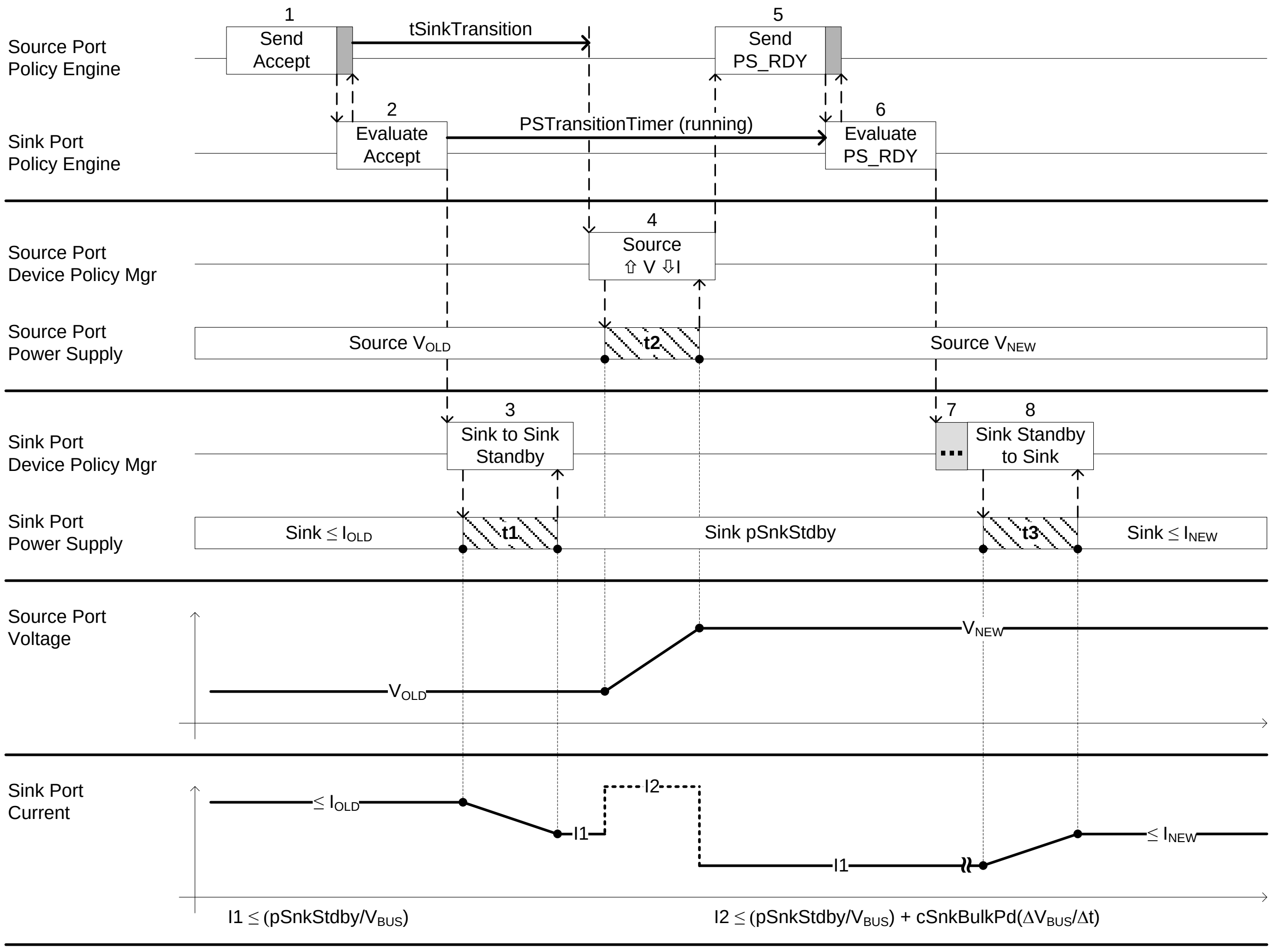


Figure 7-12 Transition Diagram for Increasing the Voltage and Decreasing the Current

Table 7-6 Sequence Description for Increasing the Voltage and Decreasing the Current

Step	Source	Sink
1	Policy Engine sends the ***Accept*** message to the Sink. Policy Engine receives the ***GoodCRC*** message from the Sink. The Policy Engine waits ***tSinkTransition*** before telling the Device Policy Manager to instruct the Power Supply to modify its output power.	Policy Engine sends the ***GoodCRC*** message to the Source.
2		Policy Engine evaluates the ***Accept*** message and starts ***PSTransitionTimer***.
3		Policy Engine tells the Device Policy Manager to instruct the Power Supply to reduce power

Step	Source	Sink
		consumption to **pSnkStdby** within **tSnkStdby** (t1).
4	After ***tSinkTransition***, the Policy Engine tells the Device Policy Manager to instruct the Power Supply to change its output voltage to operate at the new power level. The Power Supply shall be ready to operate at the new power level within ***tSrcRdyPos*** (t2). The Power Supply informs the Device Policy Manager that it is ready to operate at the new power level. The Power Supply status is passed to the Policy Engine.	
5	The Policy Engine sends the ***PS_RDY*** message to the Sink. Policy Engine receives the ***GoodCRC*** message from the Sink.	Policy Engine sends the ***GoodCRC*** message to the Source.
6		Policy Engine evaluates the ***PS_RDY*** message from the Source and tells the Device Policy to instruct the Power Supply it is okay to operate at the new power level.
7		The Sink may begin operating at the new power level any time after having evaluated the ***PS_RDY*** message. This time duration is indeterminate.
8		The Sink shall not violate the in-rush current specfications **iInRushPos** and **iInRushNeg** when transitioning to the new power level and shall complete the transition within **tSnkNewPower** (t3).

7.3.5 Decreasing the Voltage and Increasing the Current

The interaction of the System Policy, Device Policy, and Power Supply that shall be followed when decreasing the voltage and increasing the current is shown in Figure 7-13. The sequence that shall be followed is described in Table 7-7. The timing parameters that shall be followed are listed in Table 7-21, Table 7-22, and Table 7-23.

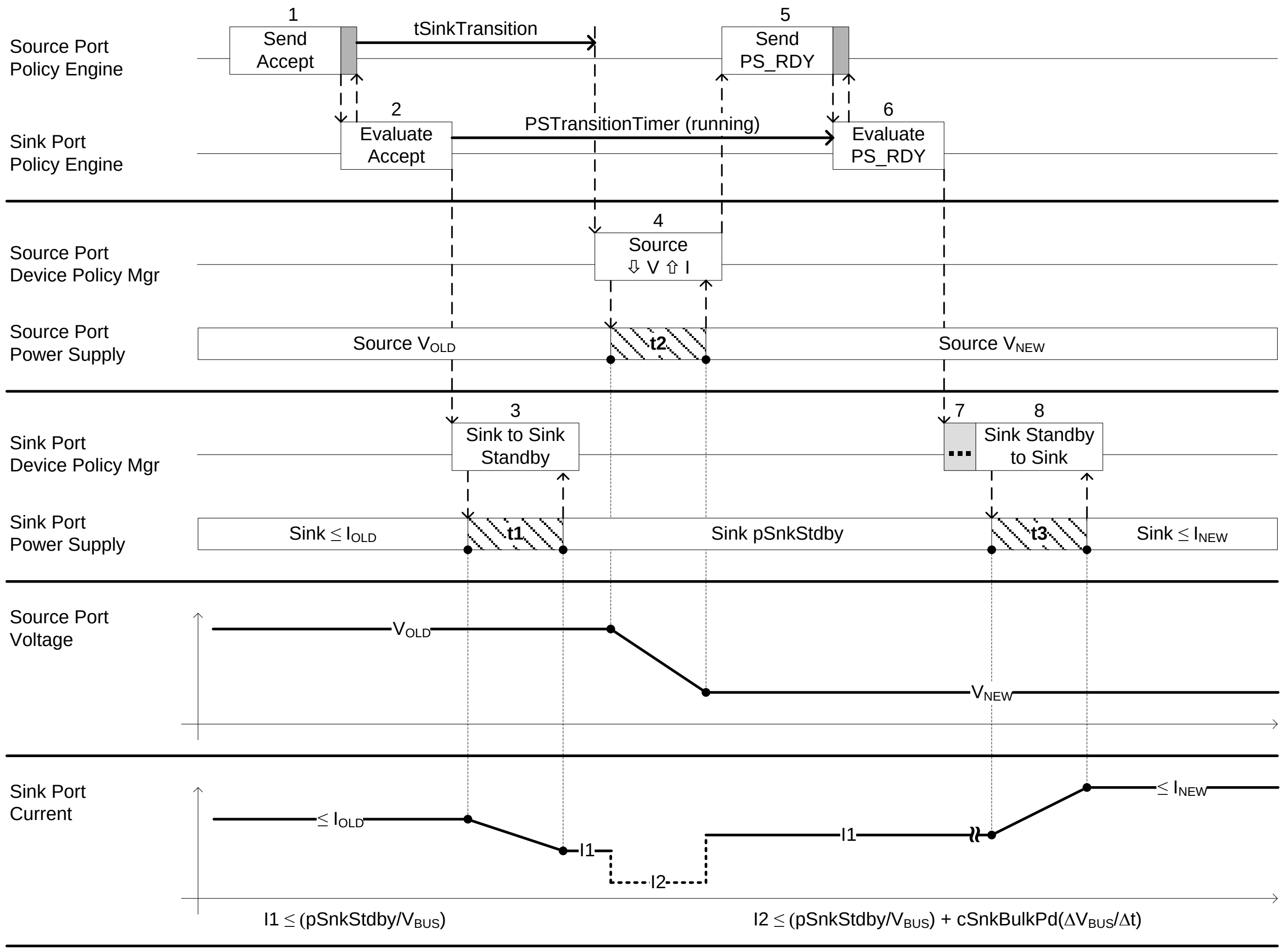


Figure 7-13 Transition Diagram for Decreasing the Voltage and Increasing the Current

Table 7-7 Sequence Description for Decreasing the Voltage and Increasing the Current

Step	Source	Sink
1	Policy Engine sends the ***Accept*** message to the Sink. Policy Engine receives the ***GoodCRC*** message from the Sink. The Policy Engine waits ***tSinkTransition*** before telling the Device Policy Manager to instruct the Power Supply to modify its output power.	Policy Engine sends the ***GoodCRC*** message to the Source.
2		Policy Engine evaluates the ***Accept*** message and starts ***PSTransitionTimer***.
3		Policy Engine tells the Device Policy Manager to instruct the Power Supply to reduce power

Step	Source	Sink
		consumption to **pSnkStdby** within **tSnkStdby** (t1).
4	After ***tSinkTransition***, the Policy Engine tells the Device Policy Manager to instruct the Power Supply to change its output voltage to operate at the new power level. The Power Supply shall be ready to operate at the new power level within ***tSrcRdyPos*** (t2). The Power Supply informs the Device Policy Manager that it is ready to operate at the new power level. The Power Supply status is passed to the Policy Engine.	
5	The Policy Engine sends the ***PS_RDY*** message to the Sink. Policy Engine receives the ***GoodCRC*** message from the Sink.	Policy Engine sends the ***GoodCRC*** message to the Source.
6		Policy Engine evaluates the ***PS_RDY*** message from the Source and tells the Device Policy to instruct the Power Supply it is okay to operate at the new power level.
7		The Sink may begin operating at the new power level any time after having evaluated the ***PS_RDY*** message. This time duration is indeterminate.
8		The Sink shall not violate the in-rush current specfications **iInRushPos** and **iInRushNeg** when transitioning to the new power level and shall complete the transition within **tSnkNewPower** (t3).

7.3.6 Decreasing the Current

The interaction of the System Policy, Device Policy, and Power Supply that shall be followed when decreasing the current is shown in Figure 7-14. The sequence that shall be followed is described in Table 7-8. The timing parameters that shall be followed are listed in Table 7-21, Table 7-22, and Table 7-23.

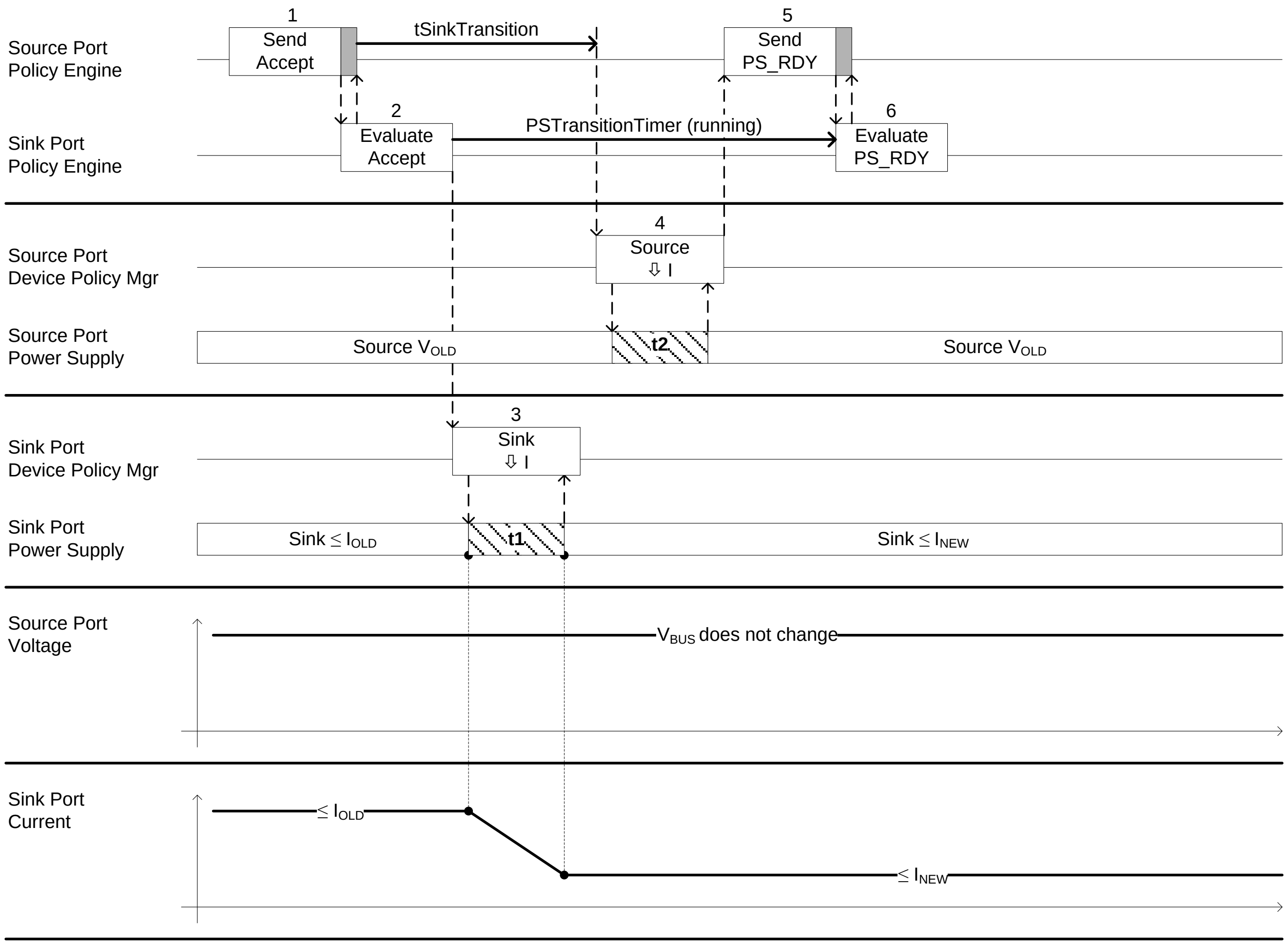


Figure 7-14 Transition Diagram for Decreasing the Current

Table 7-8 Sequence Description for Decreasing the Current

Step	Source	Sink
1	Policy Engine sends the ***Accept*** message to the Sink. Policy Engine receives the ***GoodCRC*** message from the Sink. The Policy Engine waits ***tSinkTransition*** before telling the Device Policy Manager to instruct the Power Supply to modify its output power.	Policy Engine sends the ***GoodCRC*** message to the Source.
2		Policy Engine evaluates the ***Accept*** message and starts ***PSTransitionTimer***. Policy Engine tells the Device Policy Manager to instruct the Power Supply to reduce power consumption.
3		The Sink shall not violate the in-rush current

Step	Source	Sink
		specfications **iInRushPos** and **iInRushNeg** when transitioning to the new power level and shall complete the transition within **tSnkNewPower** (t1).
4	After ***tSinkTransition***, the Policy Engine tells the Device Policy Manager to instruct the Power Supply to change its output power capability. The Power Supply shall be ready to operate at the new power level within ***tSrcNewPower*** (t2). The Power Supply informs the Device Policy Manager that it is ready to operate at the new power level. The Power Supply status is passed to the Policy Engine.	
5	The Policy Engine sends the ***PS_RDY*** message to the Sink. Policy Engine receives the ***GoodCRC*** message from the Sink.	Policy Engine sends the ***GoodCRC*** message to the Source.
6		Policy Engine evaluates the ***PS_RDY*** message from the Source. The Sink is already operating at the new power level so no further action is required.

7.3.7 Decreasing the Voltage

The interaction of the System Policy, Device Policy, and Power Supply that shall be followed when decreasing the voltage is shown in Figure 7-15. The sequence that shall be followed is described in Table 7-9. The timing parameters that shall be followed are listed in Table 7-21, Table 7-22, and Table 7-23.

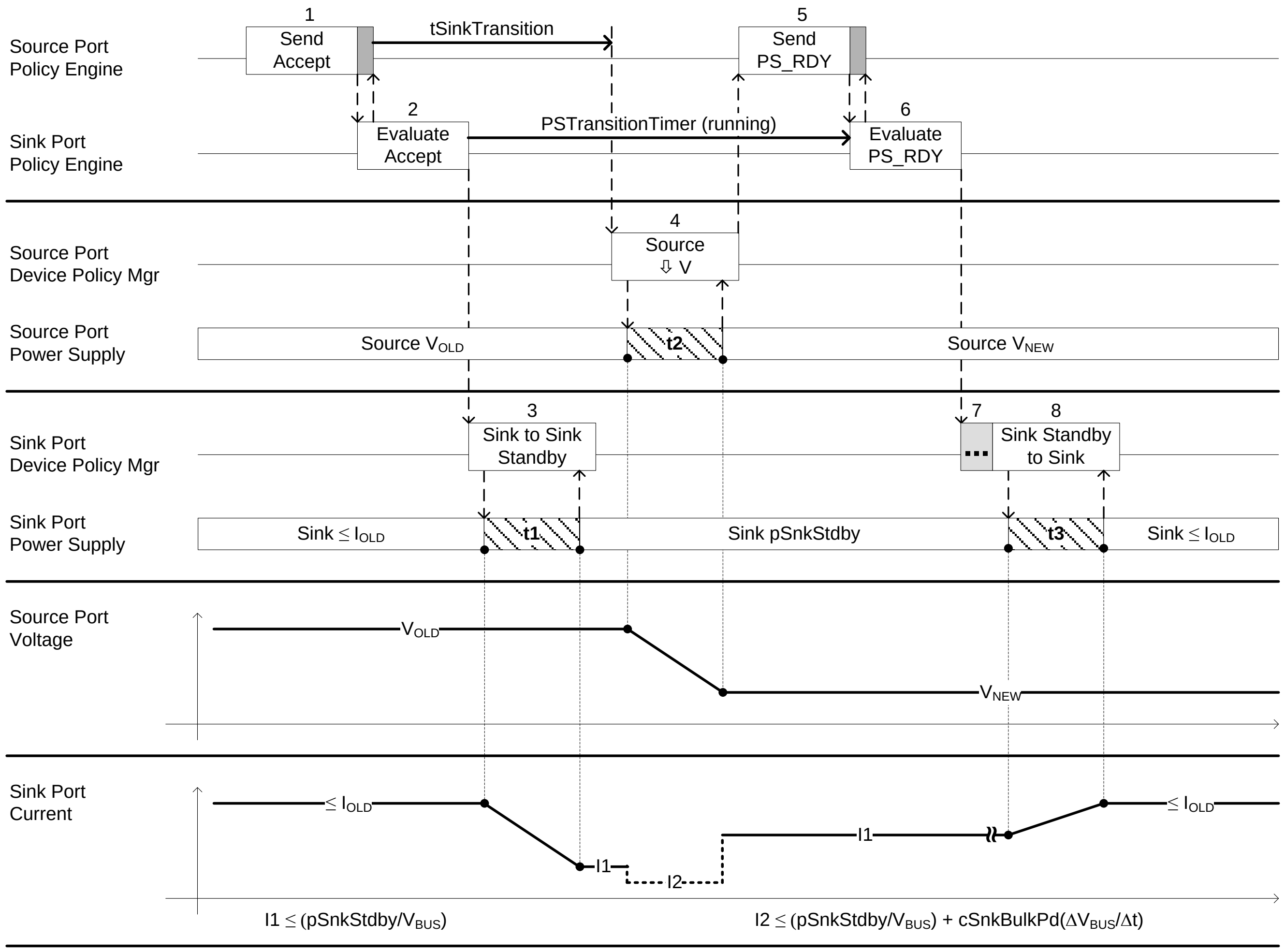


Figure 7-15 Transition Diagram for Decreasing the Voltage

Table 7-9 Sequence Description for Decreasing the Voltage

Step	Source	Sink
1	Policy Engine sends the ***Accept*** message to the Sink. Policy Engine receives the ***GoodCRC*** message from the Sink. The Policy Engine waits ***tSinkTransition*** before telling the Device Policy Manager to instruct the Power Supply to modify its output power.	Policy Engine sends the ***GoodCRC*** message to the Source.
2		Policy Engine evaluates the ***Accept*** message and starts ***PSTransitionTimer***.
3		Policy Engine tells the Device Policy Manager to instruct the Power Supply to reduce power

Step	Source	Sink
		consumption to **pSnkStdby** within **tSnkStdby** (t1).
4	After ***tSinkTransition***, the Policy Engine tells the Device Policy Manager to instruct the Power Supply to change its output voltage to operate at the new power level. The Power Supply shall be ready to operate at the new power level within ***tSrcRdyPos*** (t2). The Power Supply informs the Device Policy Manager that it is ready to operate at the new power level. The Power Supply status is passed to the Policy Engine.	
5	The Policy Engine sends the ***PS_RDY*** message to the Sink. Policy Engine receives the ***GoodCRC*** message from the Sink.	Policy Engine sends the ***GoodCRC*** message to the Source.
6		Policy Engine evaluates the ***PS_RDY*** message from the Source and tells the Device Policy to instruct the Power Supply it is okay to operate at the new power level.
7		The Sink may begin operating at the new power level any time after having evaluated the ***PS_RDY*** message. This time duration is indeterminate.
8		The Sink shall not violate the in-rush current specfications **iInRushPos** and **iInRushNeg** when transitioning to the new power level and shall complete the transition within **tSnkNewPower** (t4).

7.3.8 Decreasing the Voltage and the Current

The interaction of the System Policy, Device Policy, and Power Supply that shall be followed when decreasing the voltage and current is shown in Figure 7-16. The sequence that shall be followed is described in Table 7-10. The timing parameters that shall be followed are listed in Table 7-21, Table 7-22, and Table 7-23.

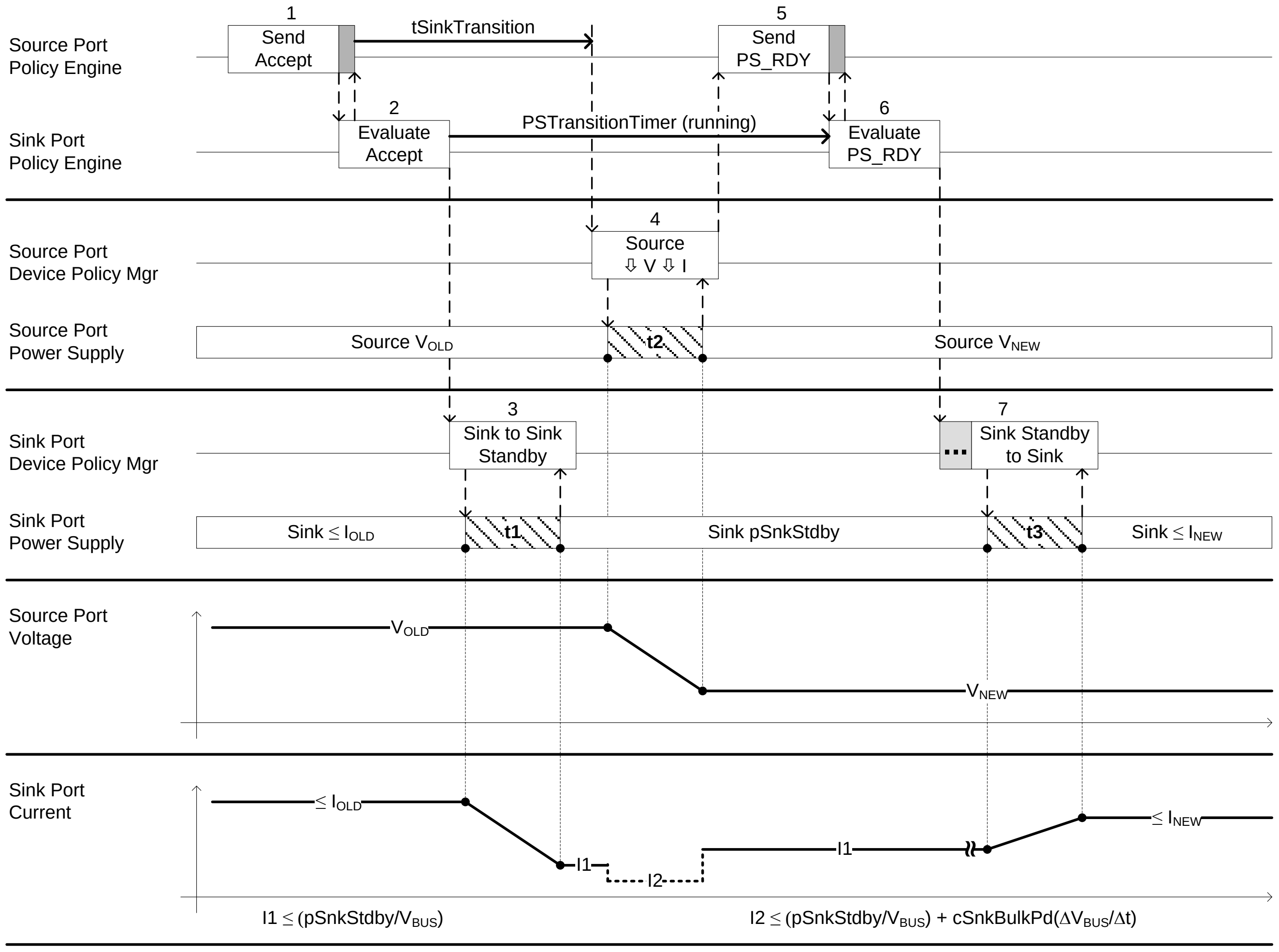


Figure 7-16 Transition Diagram for Decreasing the Voltage and the Current

Table 7-10 Sequence Description for Decreasing the Voltage and the Current

Step	Source	Sink
1	Policy Engine sends the ***Accept*** message to the Sink. Policy Engine receives the ***GoodCRC*** message from the Sink. The Policy Engine waits ***tSinkTransition*** before telling the Device Policy Manager to instruct the Power Supply to modify its output power.	Policy Engine sends the ***GoodCRC*** message to the Source.
2		Policy Engine evaluates the ***Accept*** message and starts ***PSTransitionTimer***.
3		Policy Engine tells the Device Policy Manager to instruct the Power Supply to reduce power

Step	Source	Sink
		consumption to **pSnkStdby** within **tSnkStdby** (t1).
4	After ***tSinkTransition***, the Policy Engine tells the Device Policy Manager to instruct the Power Supply to change its output voltage to operate at the new power level. The Power Supply shall be ready to operate at the new power level within ***tSrcRdyPos*** (t2). The Power Supply informs the Device Policy Manager that it is ready to operate at the new power level. The Power Supply status is passed to the Policy Engine.	
5	The Policy Engine sends the ***PS_RDY*** message to the Sink. Policy Engine receives the ***GoodCRC*** message from the Sink.	Policy Engine sends the ***GoodCRC*** message to the Source.
6		Policy Engine evaluates the ***PS_RDY*** message from the Source and tells the Device Policy to instruct the Power Supply it is okay to operate at the new power level.
7		The Sink may begin operating at the new power level any time after having evaluated the ***PS_RDY*** message. This time duration is indeterminate.
8		The Sink shall not violate the in-rush current specfications **iInRushPos** and **iInRushNeg** when transitioning to the new power level and shall complete the transition within **tSnkNewPower** (t3).

7.3.9 Sink Requested Role Swap

The interaction of the System Policy, Device Policy, and Power Supply that shall be followed during a Sink requested Role Swap is shown in Figure 7-17. The sequence that shall be followed is described in Table 7-11. The timing parameters that shall be followed are listed in Table 7-22. Note in this figure, the Sink has previously sent a ***Swap*** message to the Source.

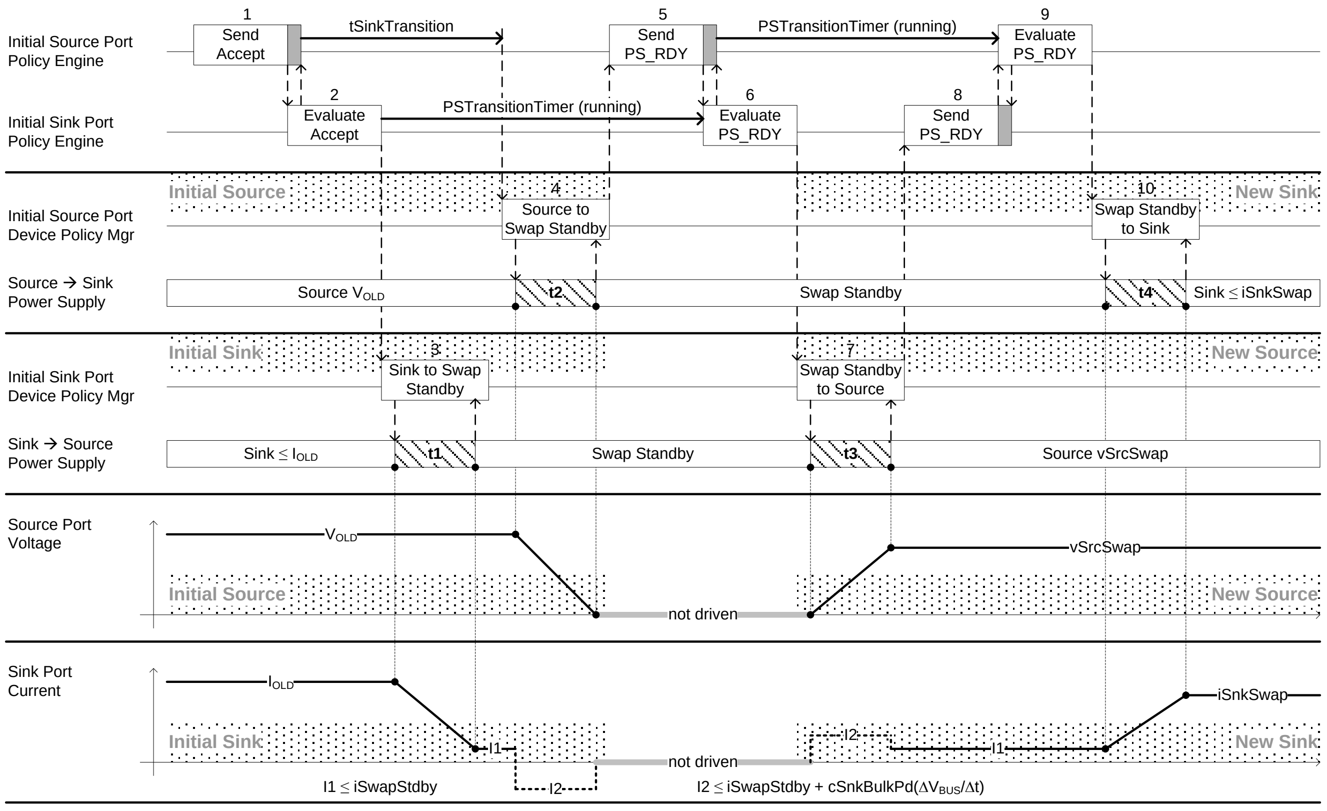


Figure 7-17 Transition Diagram for a Sink Requested Role Swap

Table 7-11 Sequence Description for a Sink Requested Role Swap

Step	Initial Source → New Sink	Initial Sink → New Source
1	Policy Engine sends the ***Accept*** message to the Initial Sink. Policy Engine receives the ***GoodCRC*** message from the Initial Sink. The Policy Engine waits ***tSinkTransition*** before telling the Device Policy Manager to instruct the Power Supply to transition to Swap Standby.	Policy Engine sends the ***GoodCRC*** message to the Initial Source.
2		Policy Engine evaluates the ***Accept*** message and starts ***PSTransitionTimer***.
3		Policy Engine tells the Device Policy Manager to instruct the Power Supply to transition to Swap Standby (t1). When in Sink Standby the Initial Sink shall not draw more than ***iSnkSwapStdby*** (I1) except during the V_{BUS} voltage transition (I2).

Step	Initial Source → New Sink	Initial Sink → New Source
4	After ***tSinkTransition***, the Policy Engine tells the Device Policy Manager to instruct the Power Supply to transition to Swap Standby. The Power Supply shall complete the transition to Swap Standby within ***tSrcSwapStdby*** (t2). The Power Supply informs the Device Policy Manager that it is ready to operate as the new Sink. The Power Supply status is passed to the Policy Engine.	
5	The Power Supply is ready and the Policy Engine sends the ***PS_RDY*** message to the device that will become the new Source. Policy Engine receives the ***GoodCRC*** message from the device that will become the new Source. Upon sending the ***PS_RDY*** message and receiving the ***GoodCRC*** message the Initial Source is ready to be the new Sink.	Policy Engine sends the ***GoodCRC*** message to the new Sink.
6		Upon evaluating the ***PS_RDY*** message the Initial Sink is ready to operate as the new Source. Policy Engine tells the Device Policy to instruct the Power Supply to operate as the new Source.
7		The Power Supply as the new Source transitions from Swap Standby to sourcing default 5V within ***tNewSRC*** (t3). The Power Supply informs the Device Policy Manager that it is operating as the new Source.
8	Policy Engine sends the ***GoodCRC*** message to the new Source.	Device Policy Manager informs the Policy Engine the Power Supply is ready and the Policy Engine sends the ***PS_RDY*** message to the new Sink. Policy Engine receives the ***GoodCRC*** message from the new Sink.
9	Policy Engine evaluates the ***PS_RDY*** message from the new Source and tells the Device Policy Manager to instruct the Power Supply to draw current as the new Sink.	
10	The Power Supply as the new Sink transitions from Swap Standby to drawing iSwap within ***tNewSnk*** (t4). The Power Supply informs the Device Policy Manager that it is operating as the new Sink. At this point subsequent negotiations between the new Source and the new Sink may proceed as normal.	

7.3.10 Source Requested Role Swap

The interaction of the System Policy, Device Policy, and Power Supply that shall be followed during a Source requested Role Swap is shown in Figure 7-18. The sequence that shall be followed is described in Table 7-12. The timing parameters that shall be followed are listed in Table 7-21. Prior to the Sink sending the ***Accept*** message (step 1) the Source has sent a ***Swap*** message to the Sink.

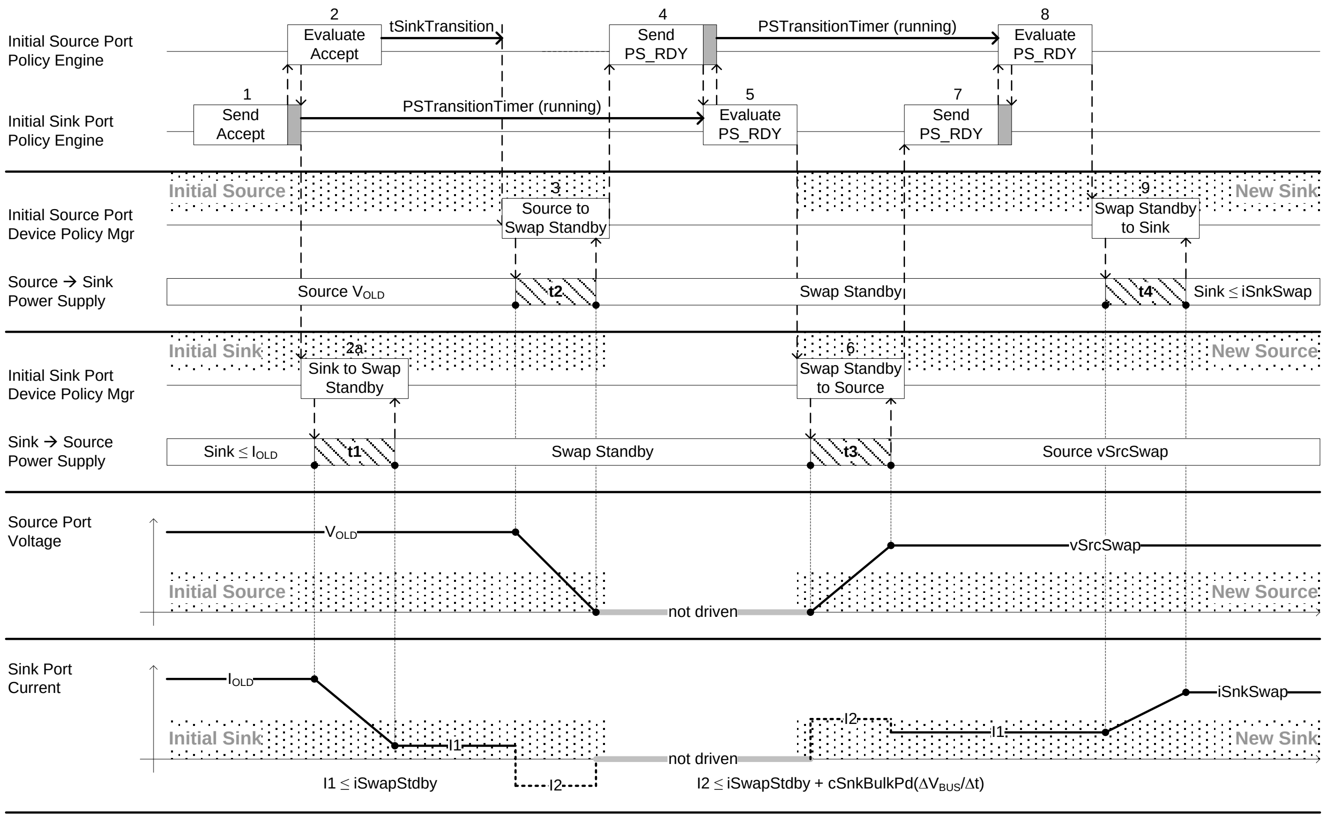


Figure 7-18 Transition Diagram for a Source Requested Role Swap

Table 7-12 Sequence Description for a Source Requested Role Swap

Step	Initial Source → New Sink	Initial Sink → New Source
1	Policy Engine sends the ***GoodCRC*** message to the Initial Sink.	Policy Engine sends the ***Accept*** message to the Initial Source. Policy Engine receives the ***GoodCRC*** message from the Initial Source.
2	Policy Engine evaluates the ***Accept*** message and waits ***tSinkTransition*** before telling the Deive Policy Manager to instruct the Power Supply to transition to Swap Standby.	
2a		The Policy Engine tells the Device Policy Manager to instruct the Power Supply to transition to Swap Standby. The Power supply shall complete the transition to Swap Standby within ***tSnkStdby*** (t1). Policy Engine starts ***PSTransitionTimer***. When in Sink Standby the Initial Sink shall not draw more than ***iSnkSwapStdby*** (I1) except during the V_{BUS}

Step	Initial Source → New Sink	Initial Sink → New Source
		voltage transition (I2).
3	After ***tSinkTransition***, the Policy Engine tells the Device Policy Manager to instruct the Power Supply to transition to Swap Standby. The Power Supply shall complete the transition to Swap Standby within ***tSrcSwapStdby*** (t2). The Power Supply informs the Device Policy Manager that it is ready to operate as the new Sink. The Power Supply status is passed to the Policy Engine.	
4	The Policy Engine sends the ***PS_RDY*** message to the soon to be new Source. Policy Engine receives the ***GoodCRC*** message from the soon to be new Source. At this point the Initial Source is ready to be the new Sink.	Policy Engine sends the ***GoodCRC*** message to the new Sink.
5		Upon evaluating the ***PS_RDY*** message the Initial Sink is ready to operate as the new Source. Policy Engine tells the Device Policy to instruct the Power Supply to operate as the new Source.
6		The Power Supply as the new Source transitions from Swap Standby to sourcing default 5V within ***tNewSRC*** (t3). The Power Supply informs the Device Policy Manager that it is operating as the new Source.
7	Policy Engine sends the ***GoodCRC*** message to the new Source.	Device Policy Manager informs the Policy Engine the Power Supply is ready and the Policy Engine sends the ***PS_RDY*** message to the new Sink. Policy Engine receives the ***GoodCRC*** message from the new Sink.
8	Policy Engine evaluates the ***PS_RDY*** message from the new Source and tells the Device Policy Manager to instruct the Power Supply to draw current as the new Sink.	
9	The Power Supply as the new Sink transitions from Swap Standby to drawing iSnkSwap within ***tNewSnk*** (t4). The Power Supply informs the Device Policy Manager that it is operating as the new Sink. At this point subsequent negotiations between the new Source and the new Sink may proceed as normal	

7.3.11 GotoMin Current Decrease

The interaction of the System Policy, Device Policy, and Power Supply that shall be followed during a GotoMin current decrease is shown in Figure 7-19. The sequence that shall be followed is described in Table 7-13. The timing parameters that shall be followed are listed in Table 7-21 and Table 7-22.

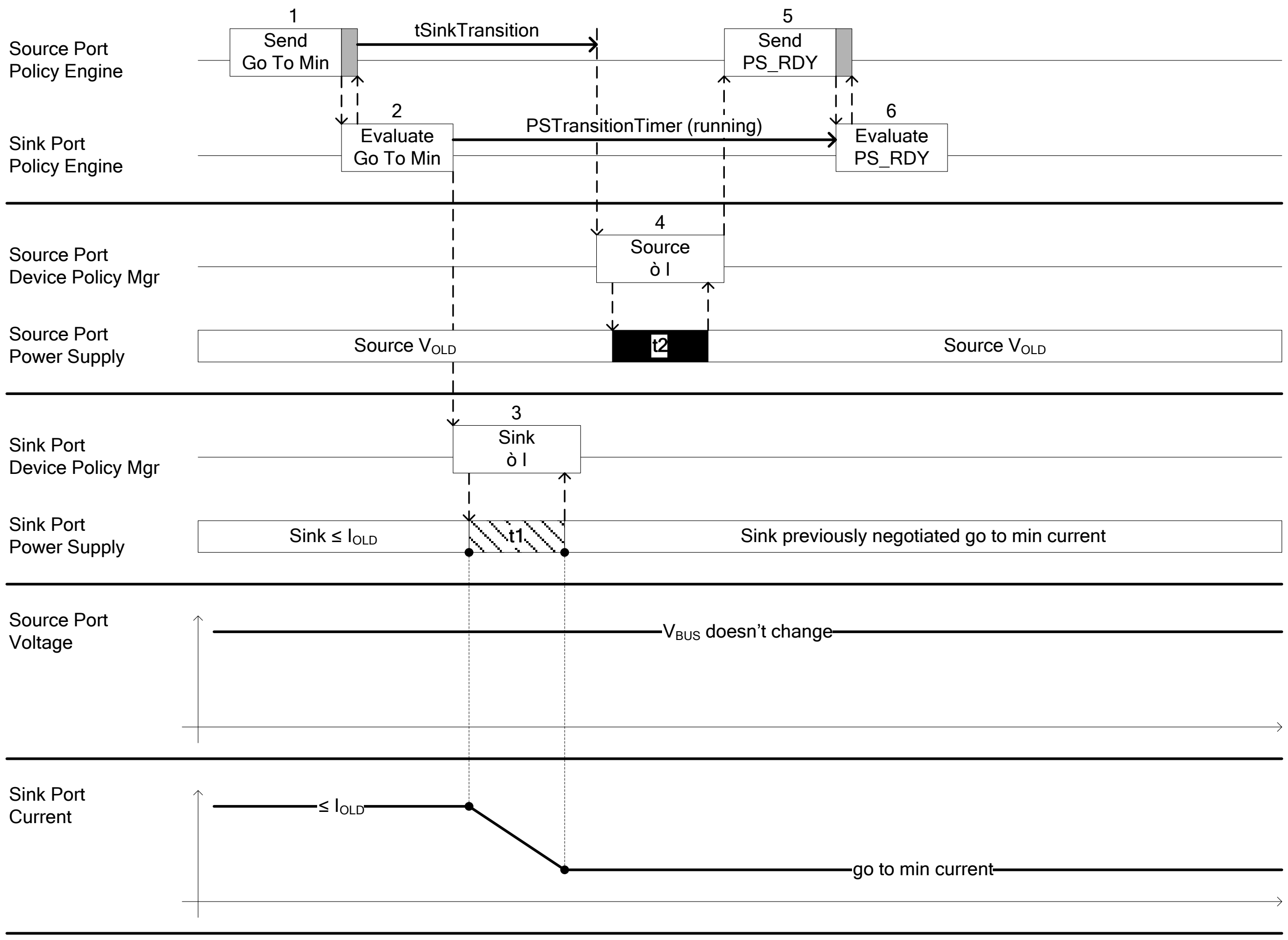


Figure 7-19 Transition Diagram for a GotoMin Current Decrease

Table 7-13 Sequence Description for a GotoMin Current Decrease

Step	Source Port	Sink Port
1	Policy Engine sends the ***GotoMin*** message to the Sink. Policy Engine receives the ***GoodCRC*** message from the Sink. The Policy Engine waits ***tSinkTransition*** before telling the Device Policy Manager to instruct the Power Supply to modify its output power.	Policy Engine sends the ***GoodCRC*** message to the Source.
2		Policy Engine evaluates the ***GotoMin*** message and starts ***PSTransitionTimer***.
3		Policy Engine tells the Device Policy Manager to

Step	Source Port	Sink Port
		instruct the Power Supply to reduce power consumption, within ***tSnkNewPower*** (t1), to the pre-negotiated go to reduced power level.
4	After ***tSinkTransition***, the Policy Engine tells the Device Policy Manager to instruct the Power Supply to change its output power capability. The Power Supply shall be ready to operate at the new power level within ***tSrcNewPower*** (t2). The Power Supply informs the Device Policy Manager that it is ready to operate at the new power level. The Power Supply status is passed to the Policy Engine.	
5	The Policy Engine sends the ***PS_RDY*** message to the Sink. Policy Engine receives the ***GoodCRC*** message from the Sink.	Policy Engine sends the ***GoodCRC*** message to the Source.
6		Policy Engine evaluates the ***PS_RDY*** message from the Source and no further action is required.

7.3.12 Source Initiated Hard Reset

The interaction of the System Policy, Device Policy, and Power Supply that shall be followed during a Source Initiated Hard Reset is shown in Figure 7-20. The sequence that shall be followed is described in Table 7-14. The timing parameters that shall be applied are listed in Table 7-21 and Table 7-22.

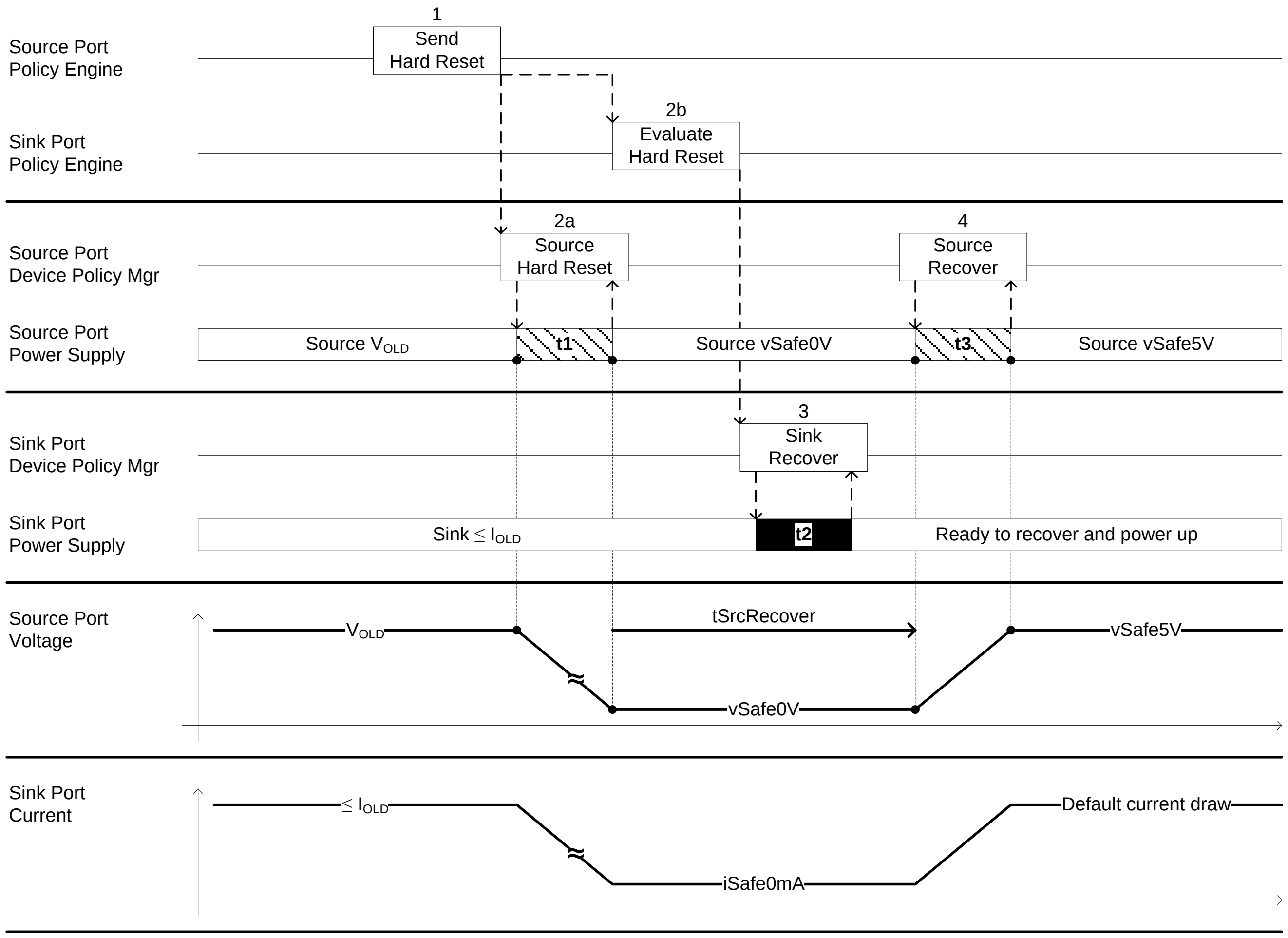


Figure 7-20 Transition Diagram for a Source Initiated Hard Reset

Table 7-14 Sequence Description for a Source Initiated Hard Reset

Step	Source Port	Sink Port
1	Policy Engine sends ***Hard Reset*** signaling to the Sink.	
2a	Policy Engine tells the Device Policy Manager to instruct the Power Supply to perform a Hard Reset. The transition to ***vSafe0V*** shall occur within ***tSrcSafe*** (t1).	The Sink shall not draw more than ***iSafe0mA*** when V_{BUS} is driven to ***vSafe0V***.
2b		Policy Engine is informed of the Hard Reset. Policy Engine tells the Device Policy Manager to instruct the Power Supply to prepare to recover from a Hard Reset.

Step	Source Port	Sink Port
3		The Power Supply shall be ready to resume normal operation within tSnkRecover (t2) and passes an indication to the Device Policy Manager.
4	After ***tSrcRecover*** the Source applies power to V_{BUS} in an attempt to re-establish commnication with the Sink and resume default operation. The transition to ***vSafe5V*** shall occur within ***tSrcTurnOn***.	

7.3.13 Sink Initiated Hard Reset

The interaction of the System Policy, Device Policy, and Power Supply that shall be followed during a Sink Initiated Hard Reset is shown in Figure 7-21. The sequence that shall be followed is described in Table 7-15. The timing parameters that shall be followed are listed in Table 7-21 and Table 7-22.

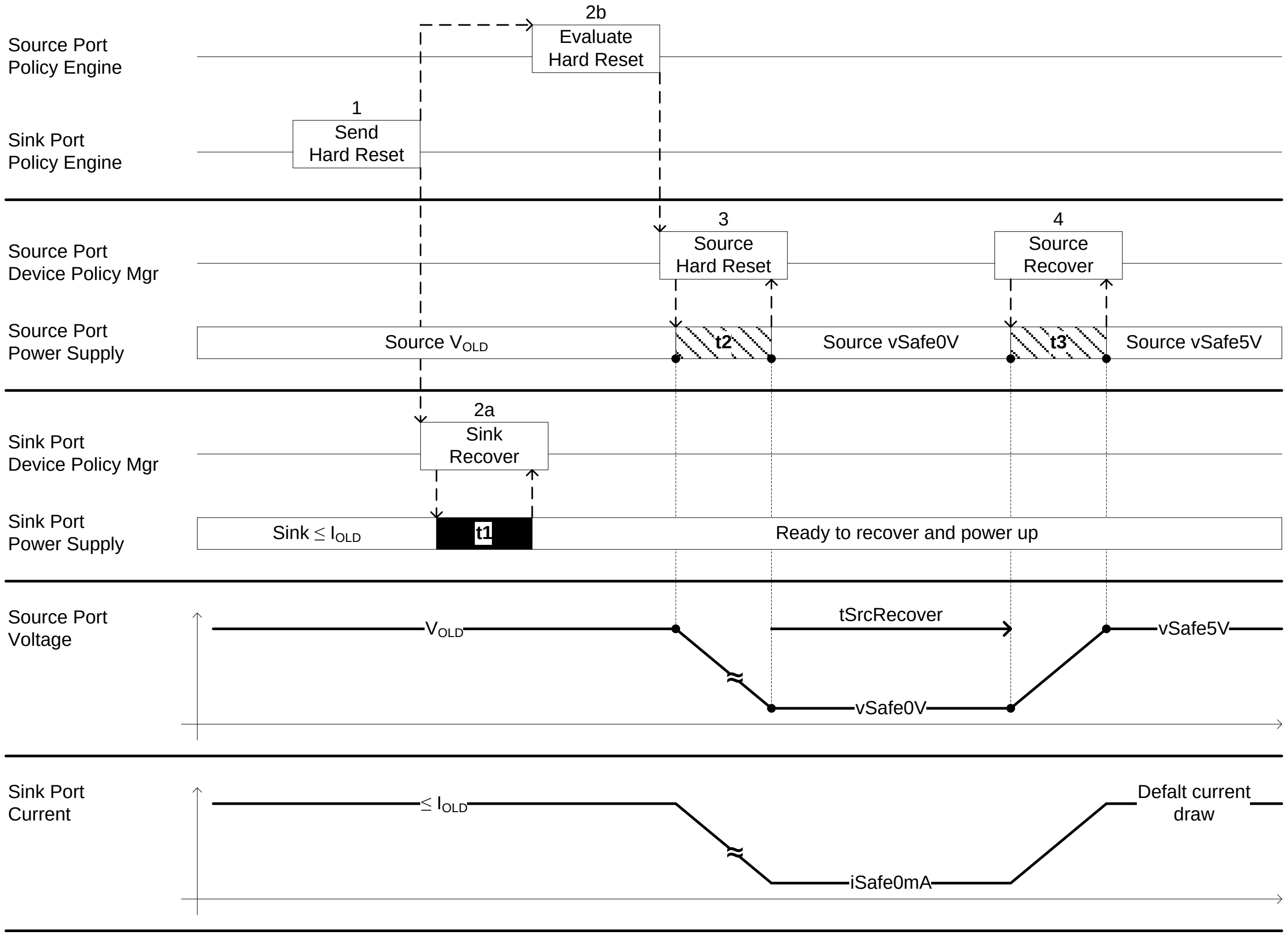


Figure 7-21 Transition Diagram for a Sink Initiated Hard Reset

Table 7-15 Sequence Description for a Sink Initiated Hard Reset

Step	Source Port	Sink Port
1		Policy Engine sends ***Hard Reset*** signaling to the Source
2a		Policy Engine tells the Device Policy Manager to instruct the Power Supply to prepare to recover from a Hard Reset. If action is taken the Sink shall be prepared within ***tSnkHardResetPrepare*** (t1).
2b	Policy Engine is informed of the Hard Reset.	
3	Policy Engine tells the Device Policy Manager to instruct the Power Supply to transition to ***vSafe0V***. The transition shall occur within	The Sink shall not draw more than ***iSafe0mA*** when V_{BUS} is driven to ***vSafe0V***.

Step	Source Port	Sink Port
	tSrcSafe (t2).	
4	After ***tSrcRecover*** the Source applies power to the Sink in an attempt to re-establish commnication and resume default operation. The transition to ***vSafe5V*** shall occur within ***tSrcTurnOn*** (t3).	

7.3.14 New Source Initiates Hard Reset and New Sink Receives Hard Reset Signaling

The interaction of the System Policy, Device Policy, and Power Supply that shall be followed during a Hard Reset initiated by the new Source when the new Sink receives ***Hard Reset*** signaling is shown in Figure 7-22. The sequence that shall be followed is described in Table 7-16. The timing parameters that shall be followed are listed in Table 7-21 and Table 7-22.

After a Hard Reset, a Provider/Consumer shall return to its Source role and a Consumer/Provider shall return to its Sink role.

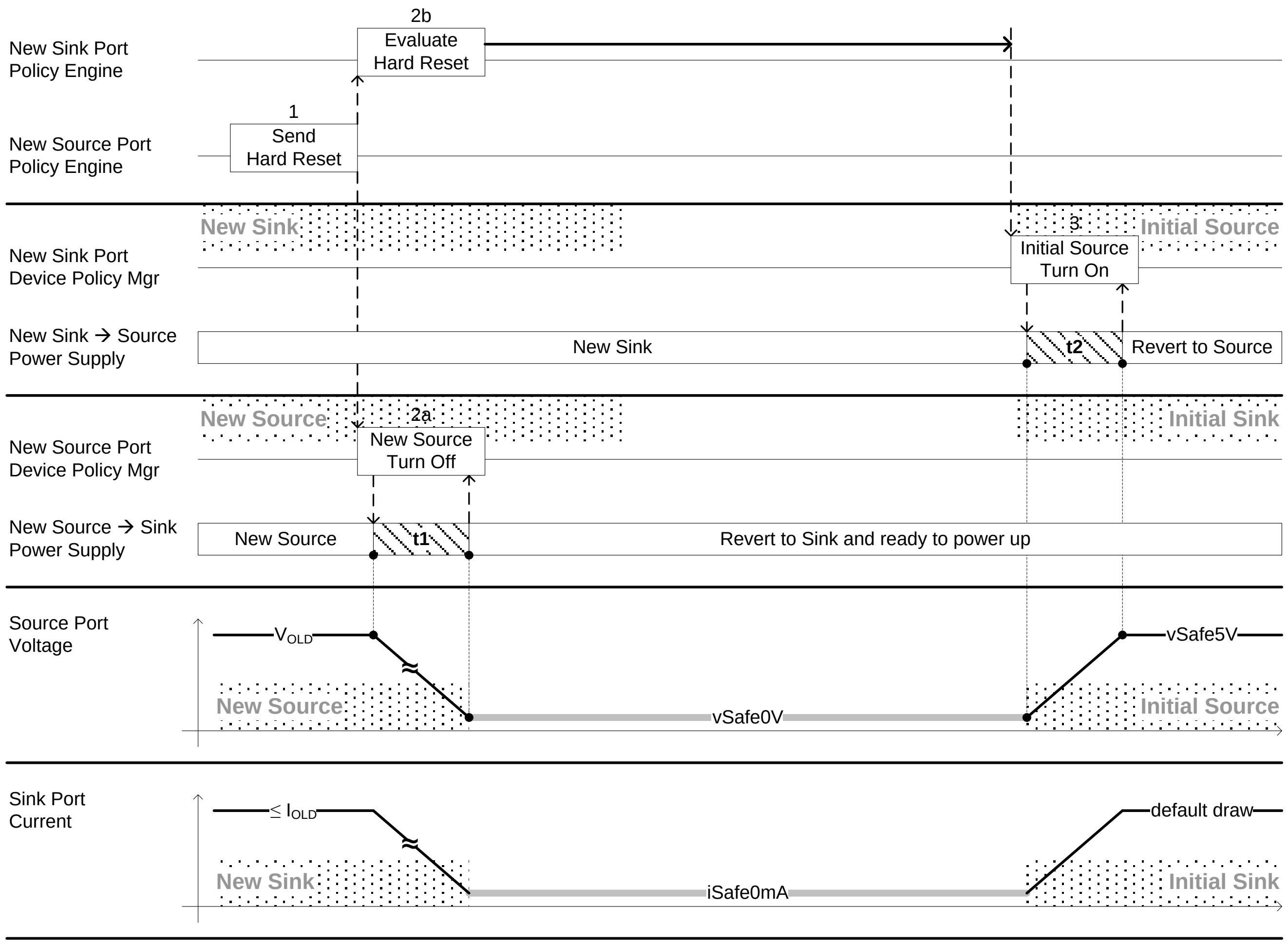


Figure 7-22 Transition Diagram for New Source Initiated Hard Reset and New Sink Receives Hard Reset Signaling

Table 7-16 Sequence Description for New Source Initiated Hard Reset and New Sink Receives Hard Reset Signaling

Step	New Source Port	New Sink Port
1	Policy Engine sends ***Hard Reset*** signaling to the new Sink and tells the Device Policy Manager to turn off the new Source.	
2a	The Device Policy Manager instructs the Power Supply to turn off the Source and revert to Sink operation by ***tNewSrcRevertToSink*** (t1).	
2b	.	Policy Engine receives and evaluates a Hard Reset

Step	New Source Port	New Sink Port
		notification and then tells the Device Policy Manager to turn on the new Source.
3		Source applies ***vSafe5V*** when V_{BUS} is within ***vSafe0V***. The transition to ***vSafe5V*** shall be completed with ***tSrcTurnOn*** (t2).

7.3.15 New Source Initiates Hard Reset and New Sink Does Not Receive Hard Reset Signaling

The interaction of the System Policy, Device Policy, and Power Supply that shall be followed during a Hard Reset initiated by the new Source when the new Sink does not receive the ***Hard Reset*** signaling is shown in Figure 7-23. The sequence that shall be followed is described in Table 7-17. The timing parameters that shall be followed are listed in Table 7-21 and Table 7-22.

After a Hard Reset, a Provider/Consumer shall return to its Source role and a Consumer/Provider shall return to its Sink role.

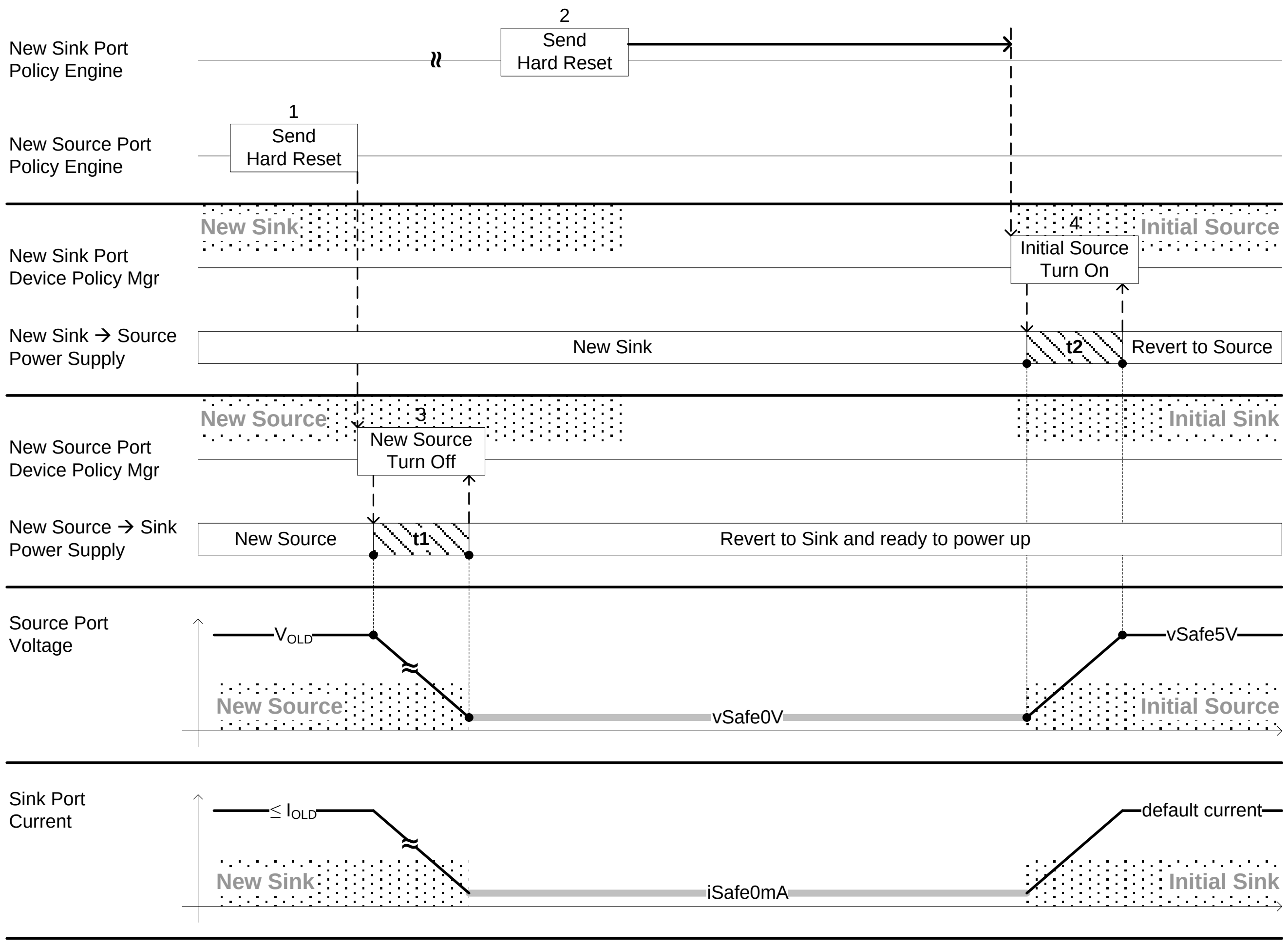


Figure 7-23 Transition Diagram for New Source Initiated Hard Reset and New Sink Does Not Receive Hard Reset Signaling

Table 7-17 Sequence Description for New Source Initiated Hard Reset and New Sink Does Not Receive Hard Reset Signaling

Step	New Source Port	New Sink Port
1	Policy Engine attempts to send the ***Hard Reset*** signaling to the new Sink and tells the Device Policy Manager to Turn off the new Source.	
2		The new Sink has not received a message from the new Source and decides to initiate a Hard Reset then tells the Device Policy Manager to turn on the new

Step	New Source Port	New Sink Port
		Source.
3	The Device Policy Manager instructs the Power Supply to turn off the Source and revert to Sink operation by ***tNewSrcRevertToSink*** (t1).	
4		Source applies ***vSafe5V*** when V_{BUS} is within ***vSafe0V***. The transition to ***vSafe5V*** shall be completed with ***tSrcTurnOn***(t2).

7.3.16 New Sink Initiates Hard Reset and New Source Receives Hard Reset Signaling

The interaction of the System Policy, Device Policy, and Power Supply that shall be followed during a Hard Reset initiated by the new Sink when the new Source receives the ***Hard Reset*** signaling is shown in Figure 7-24. The sequence that shall be followed is described in Table 7-18. The timing parameters that shall be followed are listed in Table 7-21 and Table 7-22.

After a Hard Reset a Provider/Consumer shall return to its Source role and a Consumer/Provider shall return to its Sink role.

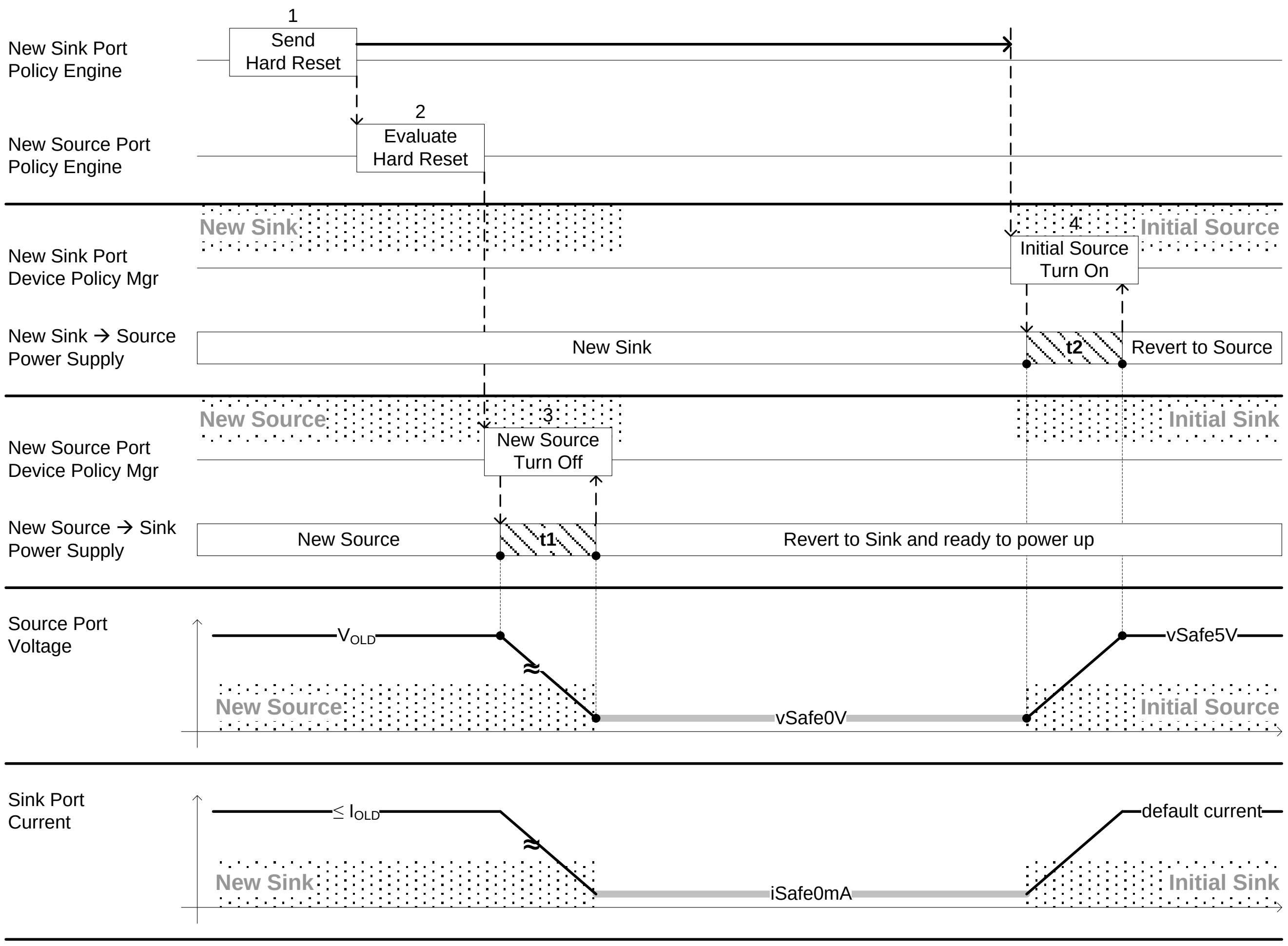


Figure 7-24 Transition Diagram for New Sink Initiated Hard Reset and New Source Receives Hard Reset Signaling

Table 7-18 Sequence Description for New Sink Initiated Hard Reset and New Source Receives Hard Reset Signaling

Step	New Source Port	New Sink Port
1		Policy Engine sends the ***Hard Reset*** signaling to the new Source and waits ***tSwapRecover*** before telling the Device Policy Manger to turn off the Sink and revert to Source operation.
2	Policy Engine evaluates the Hard Reset notification and tells the Device Policy Manager to turn off the new Source.	
3	The Device Policy Manager instructs the	

Step	New Source Port	New Sink Port
	Power Supply to turn off the Source and revert to Sink operation by ***tNewSrcRevertToSink*** (t1).	
4		Source applies ***vSafe5V*** when V_{BUS} is within ***vSafe0V***. The transition to ***vSafe5V*** shall be completed with ***tSrcTurnOn*** (t2).

7.3.17 New Sink Initiates Hard Reset and New Source Does Not Receive Hard Reset Signaling

The interaction of the System Policy, Device Policy, and Power Supply that shall be followed during a Hard Reset initiated by the new Sink when the new Source does not receive the ***Hard Reset*** signaling is shown in Figure 7-25. The sequence that shall be followed is described in Table 7-19. The timing parameters that shall be followed are listed in Table 7-21 and Table 7-22.

After a Hard Reset, a Provider/Consumer shall return to its Source role and a Consumer/Provider shall return to its Sink role.

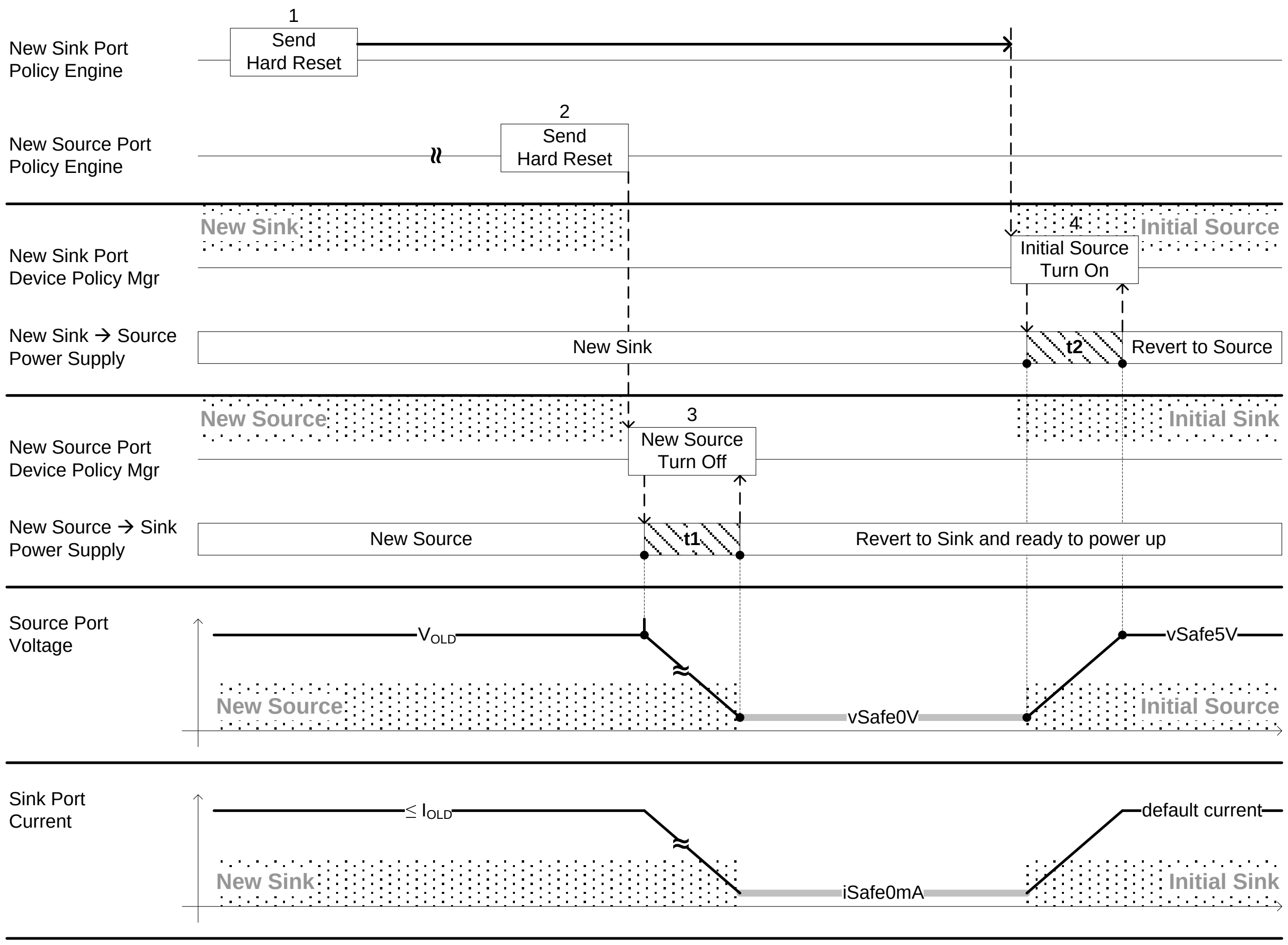


Figure 7-25 Transition Diagram for New Sink Initiated Hard Reset and New Source Does Not Receive Hard Reset Signaling

Table 7-19 Sequence Description for New Sink Initiated Hard Reset and New Source Does Not Receive Hard Reset Signaling

Step	New Source Port	New Sink Port
1		Policy Engine attempts to send the ***Hard Reset*** signaling to the new Source and waits ***tSwapRecover*** before telling the Device Policy Manger to turn off the Sink and revert to Source operation.

Step	New Source Port	New Sink Port
2	The new Source has not received a message from the new Sink and decides to initiate a Hard Reset and tells the Device Policy Manager to Turn off the new Source.	
3	The Device Policy Manager instructs the Power Supply to turn off the Source and revert to Sink operation by ***tNewSrcRevertToSink*** (t1).	
4		Source applies ***vSafe5V*** when V_{BUS} is within ***vSafe0V***. The transition to ***vSafe5V*** shall be completed with ***tSrcTurnOn*** (t2).

7.3.18 Dead Battery Operation

The interaction of the System Policy, Device Policy, and Power Supply that shall be followed during Dead Battery operation is shown in Figure 7-26. The sequence that shall be followed is described in Table 7-20. The timing parameters that shall be followed are listed in Table 4-3, Table 7-21 and Table 7-22. The Initial Source Port is not applying power to V_{BUS} at the beginning of this sequence.

After a Hard Reset, a Provider/Consumer shall return to its Source role and a Consumer/Provider shall return to its Sink role.

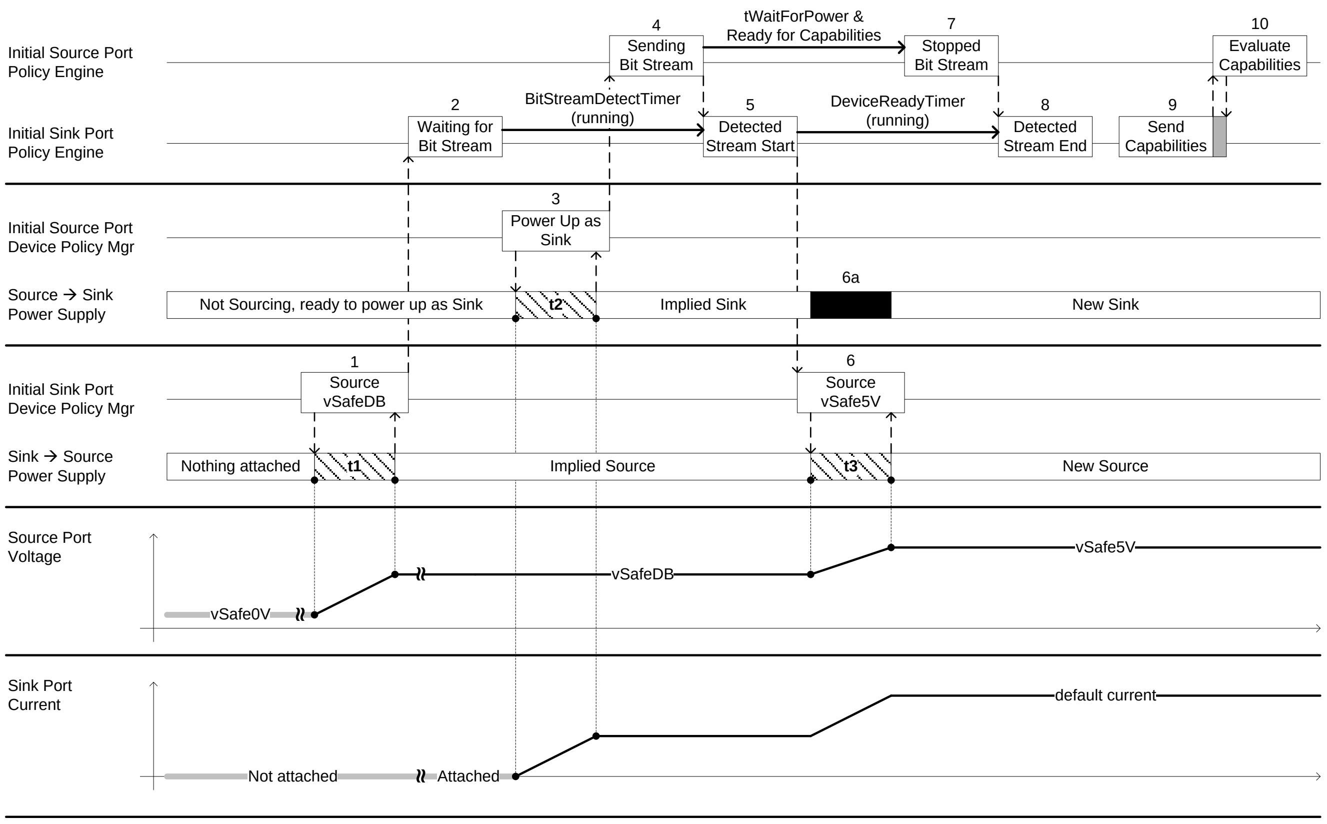


Figure 7-26 Transition Diagram for Dead Battery Operation

Table 7-20 Sequence Description for Dead Battery Operation

Step	Initial Source Port	Initial Sink Port
1		The Initial Sink detects V_{BUS} is within ***vSafe0V*** and performs an implied Role Swap to apply ***vSafeDB*** to V_{BUS}. The turn on time for ***vSafeDB*** is ***tTurnOnSafeDB*** (t1).
2		The implied Source begins to wait for the bit stream after having applied ***vSafeDB*** to V_{BUS}. If an attach does not occur before ***BitStreamDetectTimer*** times out, then the implied Source removes ***vSafeDB*** and discharges V_{BUS} to ***vSafe0V***. The dead battery start up routine for the Source will loop this behavior until an attach occurs. The loop is not illustrated in this diagram.

Step	Initial Source Port	Initial Sink Port
3	At some point in time an unpowered Source waiting to power up as the implied Sink is attached to an implied Source. The turn on time of the implied Sink is ***tTurnOnImpliedSink*** (t2).	
4	If the implied Sink is unpowered it sends the Bit Stream after having been powered up, by ***vSafeDB*** from the implied Source. The bit stream will stop being sent after the new Sink receives power and is ready for ***Capabilities***.	
5		The ***DeviceReadyTimer*** is started and this diagram assumes the bit stream will end befre the ***DeviceReadyTimer*** has timed out.
6		The implied Sink will apply ***vSafe5V*** to V_{BUS} after the start of the bit stream has been detected. The transition time to change from ***vSafeDB*** to ***vSafe5V*** is ***tSafeDBtoSafe5V*** (t3).
6a	The implied Sink will transition to new Sink operation as the V_{BUS} voltage changes from ***vSafeDB*** to ***vSafe5V***.	
7	The new Source stops sending the bit stream after the ***DBDetectTimer*** times out and the Policy Engine is ready to accept a ***Capabilities*** message.	
8		The new Source detects the end of the bit stream.
9		The Policy Engine of the new Source sends a ***Capabilities*** message to the new Sink.
10	The Policy Engine of the new Sink evaluates the ***Capabilities*** message from the new Source and responds as normal for PD negotation.	

7.4 Electrical Parameters

7.4.1 Source Electrical Parameters

The Source Electrical Parameters that shall be followed are specified in Table 7-21 and Table 4-3.

Table 7-21 Source Electrical Parameters

Reference	Parameter	Description	MIN	TYP	MAX	UNITS
Section 7.1.2	***cSrcBulk***	Source bulk capacitance when a port is powered from a dedicated supply.	10		300	µF
Section	***cSrcBulkShared***	Source bulk capacitance when a	120		300	µF

Reference	Parameter	Description	MIN	TYP	MAX	UNITS
7.1.2		port is powered from a shared supply.				
Section 7.1.9	***snrSrc***	Source's output Signal-to-noise ratio	31			dB
Figure 7-17, Figure 7-18	***tNewSnk***	Maximum time allowed for an initial Source in Swap Standby to transition new Sink operation.			15	ms
Table 7-11, Table 7-12 Table 7-17	***tNewSrcRevertToSink***	Hard Reset during Role Swap timing parameter.			90	ms
Figure 7-2, Figure 7-3	***tolSrc***	Measured at Source receptacle for Fixed and Programmable supplies only. See note 1.	0.95 x NV[1]		1.05 x NV[1]	V
Figure 7-2 Section 7.1.4	***tolTranLowerPos***	Lower bound of positive transition window			0.8 x NV	V
Figure 7-2 Section 7.1.4	***tolTranUpperPos***	Upper bound of positive transition window			1.1 x NV	V
Table 7-20	***tSafeDBtoSafe5V***	Transition time from ***vSafeDB*** to ***vSafe5V***.			15	ms
Figure 7-2, Figure 7-3	***tSrcNewPower***	Time from positive/negative transition start (t0) to Source ready at new power level			90	ms
Section 7.1.7	***tSrcPwrRdy***	Time alloted for the Source to change its output power capability			15	ms
Table 7-4, Table 7-5, Table 7-6, Table 7-7, Table 7-9, Table 7-10	***tSrcRdyPos***	The Power Supply shall be ready to operate at the new power level within this time.			90	ms
Section 7.1.6	***tSrcRecover***	Time allotted for the Source to recover	0.5			s
Section 7.1.6	***tSrcSafe***	Response time to be at ***vSafe5V*** or ***vSafe0V***			90	ms

Reference	Parameter	Description	MIN	TYP	MAX	UNITS
Table 7-11, Table 7-12	***tSrcSwapStdby***				90	ms
Section 7.1.6	***tSrcTurnOn***	Transition time from ***vSafe0V*** to ***vSafe5V***			90	ms
Table 7-20	***tTurnOnSafeDB***	Time to turn on ***vSafeDB*** source			15	ms
Section 7.1.6	***vSrcNeg***	Most negative voltage allowed during transition			-0.3	V
Note 1: NV stands for Normal Voltage Note 2: The parameter tolSrc does not apply to Batteries nor Variable Power Supplies. These supply types are exposed during negotiation as a absolute min and max voltage levels.						

7.4.2 Sink Electrical Parameters

The Sink Electrical Parameters are specified in this section.

Table 7-22 Sink Electrical Parameters

Reference	Parameter	Description	MIN	TYP	MAX	UNITS
Section 7.2.2	***cSnkBulk***	Sink bulk capacitance on V_{BUS} at attach	1		10	µF
Section 7.2.2	***cSnkBulkPd***	Bulk capacitance on V_{BUS} a Sink is allowed after a sucessful negotiation	1		100	µF
Section 7.2.5	***iInRushPos***	Current draw change rate when decreasing current			-100	mA/µs
Section 7.2.5	***iInRushNeg***	Current draw change rate when increasing current			100	mA/µs
Section 7.2.2	***iCapChange***	Transient current allowed to flow when the Sink changes its bulk capacitance			10	mA
Figure 7-20 Figure 7-21	***iSafe0mA***	Maximum current a Sink is allowed to draw when V_{BUS} is driven to ***vSafe0V***.			1.0	mA
Section 7.2.8	***iSnkSwapStdby***	Maximum current a Sink can draw during Swap Standby. Ideally			2.5	mA

Reference	Parameter	Description	MIN	TYP	MAX	UNIT S
		this current is very near to 0 mA largely influenced by port leakage current.				
Section 7.2.3	***pSnkStdby***	Power consumption during Sink Standby			150	mW
Section 7.2.4	***pSnkSusp***	Suspend power consumption			25	mW
Section 7.2.7	***snrSnk***	Signal-to-noise ratio of Sink's noise reflected on V_{BUS}.	31			dB
Table 7-12 Table 7-18	***tNewSRC***	Maximum time allowed for an initial Source in Swap Standby to transition new Sink operation.			15	ms
Figure 7-22, Figure 7-23, Figure 7-24, Figure 7-25	***tNewSnkRevertToSrc***	Hard Reset during Role Swap timing parameter			90	ms
	tSnkHardResetPrepare	Time alloted for the Sink power electronics to prepare for a Hard Reset.			15	ms
Section 7.2.3	***tSnkStdby***	Time to transition to Sink Standby from Sink			15	ms
Section 7.2.3	***tSnkRtn***	Time to transition to Sink from Sink Standby			15	ms
Section 7.2.5	***tSnkNewPower***	Maximum transition time between power levels			10	ms
Table 7-14	***tSnkRecover***	Time for the Sink to resume default operation.			150	ms

Reference	Parameter	Description	MIN	TYP	MAX	UNITS
Figure 7-3	***tSrcSettleNeg***	Settling time from negative transition start (t0) to when transitioning voltage is within the specified tolerance of the final voltage			80	ms
Figure 7-2	***tSrcSettlePos***	Settling time from positive tranistion start (t0) to when transitioning voltage is within the specified tolerance of the final voltage			80	ms
Table 7-18, Table 7-19	***tSwapRecover***	Provides settling time after a hard reset before powering back up	0.5			s
Table 7-20	***tTurnOnImpliedSink***	Turn on time of implied Sink during Dead Battery operation			15	ms
Figure 7-2	***tTranFastPos***	Transition to ***tolTranLowerPos*** can not occur faster than this time relative to t0 and is a function of the connector contact rating	Eq.1 Eq.2			ms
Figure 7-3	***tTranFastNeg***	Transition to ***tolTranUpperNeg*** or ***tolTranLowerNeg*** can not occur faster than this time relative to t0 and is a function of the connector contact rating	Eq.1 Eq.2			ms
Eq.1: (3.0A contact rating) = 133e-3 * [OV - (NV * 1.2)] assuming the maximum total bulk capacitance on V_{BUS} during PD Eq.2: (5.0A contact rating) = 80e-3 * [OV - (NV * 1.2)] assuming the maximum total bulk capacitance on V_{BUS} during PD						

7.4.3 Common Electrical Parameters

Electrical Parameters that are common to both the Source and the Sink are specified in this section.

Table 7-23 Common Source/Sink Electrical Parameters

Reference	Parameter	Description	MIN	TYP	MAX	UNITS
Figure 7-3	***tolTranLowerNeg***	Lower bound of negative transition window			0.9 x NV	V
Figure 7-3	***tolTranUpperNeg***	Upper bound of negative transition window			1.2 x NV	V
Section 7.1.6	***vSafe5V***	Safe operating voltage at 5V	4.75		5.25	V
Section 7.1.6	***vSafe0V***	Safe operating voltage at "zero volts"	0		0.5	V
Section 7.1.14 Section 7.2.9 Table 7-20	***vSafeDB***	Safe operating voltage for Dual-Role ports operating as DB Source.		Refer to Load Line in Figure 7-8.		V
Notes: NV - New Voltage OV - Original Voltage						

8. Device Policy

8.1 Overview

This section describes the Device Policy and Policy Engine that implements it. For an overview of the architecture and how the Device Policy Manager fits into this architecture, please see Section 2.4.

8.2 Device Policy Manager

The Device Policy Manager is responsible for managing the power used by one or more USB Power Delivery ports. In order to have sufficient knowledge to complete this task it needs relevant information about the device it resides in. Firstly it has a priori knowledge of the device including the capabilities of the power supply and the receptacles on each port since these will for example have specific current ratings. It also has to know information from the cable detection module regarding cable insertion, type and rating of cable etc. It also has to have information from the Power Supply about changes in its capabilities as well as being able to request power supply changes. With all of this information the Device Policy Manager is able to provide up to date information regarding the capabilities available to a specific port and to manage the power resources within the device.

When working out the capabilities for a given Source port the Device Policy Manager will take into account firstly the current rating of the port's receptacle and whether the inserted cable is PD or non-PD rated and if so what is the capability of the plug. This will set an upper bound for the capabilities which may be offered. After this the Device Policy Manager will consider the available Power Supply resources since this will bound which voltages and currents may be offered. Finally, the Device Policy Manager will consider what power is currently allocated to other ports, which power is in reserve and any other amendments to Policy from the System Policy Manager. The Device Policy Manager will offer a set of Capabilities within the bounds detailed above.

When selecting a capability for a given Sink port the Device Policy Manager will first look at the capabilities offered by the Source. The Device Policy Manager will take into account the current rating of the port's receptacle and whether the inserted cable is PD or non-PD rated and if so what is the capability of the plug. This will set an upper bound for the capabilities which may be requested. The Device Policy Manager will also consider which capabilities are required by the Sink in order to operate. If an appropriate match for Voltage and Current can be found within the limits of the receptacle and cable then this will be requested from the Source. If an appropriate match cannot be found then a request for default voltage and current will be made, along with an indication of a capability mismatch.

For Dual-Role ports the Device Policy Manager manages the functionality of both a Source and a Sink. In addition it is able to manage the role swap process between the two. In terms of power management this could mean that a port which is initially consuming power as a Sink is able to become a power resource as a Source. Conversely, attached Sources may request that power be provided to them.

The functionality within the Device Policy Manager (and to a certain extent the Policy Engine) is scalable depending on the complexity of the device, including the number of different power supply capabilities and the number of different features supported for example System Policy Manager interface or Capability Mismatch, and the number of ports being managed. Within these parameters it is possible to implement devices from very simple power supplies to more complex power supplies or devices such as USB hubs or Hard Drives. Within multiport devices it is also permitted to have a combination of USB Power Delivery and non-USB Power Delivery ports which should all be managed by the Device Policy Manager.

As noted in Section 2.4 the logical architecture used in the PD specification will vary depending on the implementation. This means that different implementations of the Device Policy Manager may be relative small or large depending on the complexity of the device, as indicated above. It is also possible to allocate different responsibilities between the Policy Engine and the Device Policy Manager, which will lead to different types of architectures and interfaces.

The Device Policy Manager shall be responsible for the following:

- Maintaining the Local Policy for the Device including evaluating and responding to capabilities related requests from the Policy Engine for a given port.
- Control of the Source/Sink in the Device.
- Control of the cable detection module for each port.

- Interface to the Policy Engine for a given port.

The Device Policy Manager shall be responsible for the following optional features when implemented:

- Communications with the System Policy over USB.
- Monitoring and balancing power requirements across multiple ports.
- Monitoring of batteries and AC and bulk power supplies.

8.2.1 Capabilities

The Device Policy Manager in a Provider shall know the Power supplies available in the Device and their capabilities. In addition it shall be aware of any other PD Sources of power such as batteries and AC inputs. The available power sources and existing demands on the Device shall be taken into account when presenting capabilities to a Sink.

The Device Policy Manager in a Consumer shall know the requirements of the Sink and use this to evaluate the capabilities offered by a Source. It shall be aware of its own power sources e.g. Batteries or AC supplies where these have a bearing on its operation as a Sink.

The Device Policy Manager in a Provider/Consumer or Consumer/Provider shall combine the above capabilities and shall also be able to present the dual-role nature of the Device to an attached PD Device.

8.2.2 System Policy

A given PD device may have no USB capability, or PD may have been added to a USB device in such a way that PD is not integrated with USB. In these two cases there shall be no requirement for the Device Policy Manager to interact with the USB interface of the device. The following requirements shall only apply to PD devices that expose PD functionality over USB.

The Device Policy Manager shall communicate over USB with the System Policy Manager according to the requirements detailed in Chapter 8.3.3. Whenever requested the Device Policy Manager shall implement a Local Policy according to that requested by the System Policy Manager. For example the System Policy Manager may request that a battery powered Device temporarily stops charging so that there is sufficient power for a HDD to spin up.

The System Policy Manager may operate in an “Intrusive” POLICY_MODE where supported by a PD device. This means that each and every request, which can be handled by the System Policy Manager, received by the Device Policy Manager is passed up to the USB Host for a response. In practice this means that there is a Device Policy Manager running on the USB Host and communicating via the USB bus to the PD Module. This enables software based implementations of the Device Policy Manager but has serious implications in terms of response times which have to be taken into consideration.

Note: that due to timing constraints, a PD device shall be able to respond autonomously to all time-critical PD related requests.

8.2.3 Control of Source/Sink

The Device Policy Manager for a Provider shall manage the power supply for each PD Source Port and shall know at any given time what the negotiated power is. It shall request transitions of the supply and inform the Policy Engine whenever a transition completes.

The Device Policy Manager for a Consumer shall manage the Sink for each PD Sink Port and shall know at any given time what the negotiated power is.

The Device Policy Manager for a Provider/Consumer or Consumer/Provider shall manage the transition between Source/Sink roles for each PD dual-role port and shall know at any given time what operational role the port is in.

The Device Policy Manager for a Consumer/Provider shall be able to detect and handle cases where back-powering is required due to a dead battery on the Provider/Consumer side. It shall determine the absence of V_{BUS} and the Provider/Consumer’s ability to be back-powered from the Cable Detection module. It shall then initiate Provider role and instruct the Power Supply to start providing default output power. Refer to Section 4.1.

The Device Policy Manager for a Provider/Consumer may be able to detect and handle cases where back-powering is possible due to a dead battery. Where supported it shall determine the presence of V_{BUS} from the Cable Detection module. It shall then initiate Consumer role and instruct the Power Supply to start sinking default input power.

8.2.4 Cable Detection

8.2.4.1 Device Policy Manager in a Provider

The Device Policy Manager in the Provider shall control the Cable Detection module and shall be able to use the Cable Detection module to determine the attachment of a USB Power Delivery or non-USB Power Delivery cable for a given port.

This information and the type of receptacle on the local device shall be used to determine the capabilities of the port and attached cabling.

8.2.4.2 Device Policy Manager in a Consumer

The Device Policy Manager in a Consumer shall control the Cable Detection module and shall be able to use the Cable Detection module to determine the following for a given port:

- Attachment of a USB Power Delivery or non-USB Power Delivery cable.
- Cable IR drop.

The type of cabling, receptacle and Cable IR drop together shall be used together to determine the capabilities of the port and attached cabling.

8.2.4.3 Device Policy Manager in a Consumer/Provider

The Device Policy Manager in a Consumer/Provider inherits characteristics of Consumers and Providers and shall control the Cable Detection module in order to support the Dead Battery back-powering case to determine the following for a given port:

- Attachment of a USB Power Delivery Provider/Consumer which supports Dead Battery back-powering
- Presence of V_{BUS}.

8.2.4.4 Device Policy Manager in a Provider/Consumer

The Device Policy Manager in a Provider/Consumer inherits characteristics of Consumers and Providers and may control the Cable Detection module in order to support the Dead Battery back-powering case to determine the following for a given port:

- Presence of V_{BUS}.

8.2.5 Managing Power Requirements

The Device Policy Manager in a Provider shall be aware of the power requirements of all Devices connected to its Source Ports. This includes being aware of any reserve power that may be required by Devices in the future and ensuring that power is shared optimally amongst attached PD Devices. This is a key function of the Device Policy Manager, whose implementation is critical to ensuring that all PD Devices get the power they require in a timely fashion in order to facilitate smooth operation. This is balanced by the fact that the Device Policy Manager is responsible for managing the sources of power that are, by definition, finite.

The Consumer's Device Policy Manager shall ensure that it takes no more power than is required to perform its functions and gives back unneeded power whenever possible (in such cases the Provider shall maintain a reserve to ensure future operation is possible).

8.2.5.1 Managing the Power Reserve

There may be some products where a Device has certain functionality at one power level and a greater functionality at another, for example a Printer/Scanner that operates only as a printer with one power level and as a scanner if it can get more power. Visibility of the linkage between power and functionality will only be apparent at the USB Host; however the Device Policy Manager provides the mechanisms to manage the power requirements of such Devices.

Devices with the GiveBack flag cleared report Operating Current and Maximum Operating Current (see Section 6.4.2). For many Devices the Operating Current and the Maximum Operating Current will be the same. Devices with highly variable loads, such as Hard Disk Drives, may use Maximum Operating Current.

Devices with the GiveBack flag set report Operating Current and Minimum Operating Current (see Section 6.4.2). For many Devices the Operating Current and the Minimum Operating Current will be the same. Devices that charge their own batteries may use the Minimum Operating Current and GiveBack flag.

For example in the first case, a mobile device may require 500mA to operate, but would like an additional 1000mA to charge its battery. The mobile device would then request 500mA in the Minimum Operating Current field and 1500mA in the Operating Current field. Additionally, it would set the GiveBack flag (see Section 6.4.2.2) indicating to the Provider that it could temporarily recover the 1000mA to meet a transitory request.

In the second case, a Hard Disk Drive (HDD) may require 2A to spin-up, but only 1A to operate. At startup the HDD would request Maximum Operating Current of 2A and an Operating Current of 2A. After the drive is spun-up and ready to operate it would make another request of 1A for its Operating Current and 2A for its Maximum Operating Current. Over time, its inactivity timers may expire and the HDD will go to a lower power state. When the HDD is next accessed, it has to spin-up again. So it will request an Operating Current of 2A and a Maximum Operating Current of 2A. The Provider may have the extra power available immediately and can immediately honor the request. If the power is not available, the Provider may have to harvest power, for example use the ***GotoMin*** message to get back some power before honoring the HDD's request. In such a case, the HDD would be told to wait using a ***Wait*** message. The HDD continues to Request additional power until the request is finally granted.

It shall be the Device Policy Manager's responsibility to allocate power and maintain a power Reserve so as not to over-subscribe its available power resource. A Device with multiple ports such as a Hub shall always be able to meet the incremental demands of the port requiring the highest incremental power from its Reserve.

The ***GotoMin*** message is designed to allow the Provider to reclaim power from one port to support a Consumer on another port that temporarily requires additional power to perform some short term operation. In the example above, the mobile device that is being charged reduces its charge rate to allow a Device Policy Manager to meet a request from an HDD for start-up current required to spin-up its platters.

A Consumer requesting power shall take into account its operational requirements when advertising its ability to temporarily return power. For example, a mobile device with a dead battery that is being used to make a call should make a request that retains sufficient power to continue the call. When the Consumer's requirements change, it shall re-negotiate its power to reflect the changed requirements.

8.2.5.2 Power Capability Mismatch

A capability mismatch occurs when a Consumer cannot obtain required power from a Provider and the Consumer requires such capabilities to operate. Different actions are taken by the Device Policy Manager and the System Policy Manager in this case.

8.2.5.2.1 Local device handling of mismatch

The Consumer's Device Policy Manager shall cause a message to be displayed to the end user that a power capability mismatch has occurred. Examples of such feedback can include:

- For a simple Device an LED may be used to indicate the failure. For example, during connection the LED could be solid amber. If the connection is successful the LED could change to green. If the connection fails it could be red or alternately blink amber.
- A more sophisticated Device with a user interface, e.g., a mobile device or monitor, should provide notification through the user interface on the Device.

The Provider's Device Policy Manager may cause a message to be displayed to the user of the power capability mismatch.

8.2.5.2.2 Device Policy Manager Communication with System Policy

In a USB Power Delivery aware system with an active System Policy manager (see Section 8.2.2), the Device Policy Manager shall notify the System Policy Manager of the mismatch. This information shall be passed back to the System Policy Manager using the mechanisms described in Chapter 8.3.3. The System Policy Manager should ensure that the user is informed of the condition. When another port in the system could satisfy the Consumer's power requirements the user should be directed to move the Device to the alternate port.

In order to identify a more suitable Source Port for the Consumer the System Policy Manager shall communicate with the Device Policy Manager in order to determine the Consumer's requirements. The Device Policy Manager shall use a ***Get_Sink_Cap*** message (see Section 6.3.8) to discover which Power levels can be utilized by the Consumer.

8.2.6 Monitor Batteries, AC and Bulk supplies

The Device Policy Manager in a Provider may monitor the status of any variable sources of power that could have an impact on its capabilities as a Source such as Batteries, AC and Bulk supplies. When monitored, and a USB interface is supported, the battery state shall be reported to the System Policy Manager using the USB interface.

The Device Policy Manager in a Consumer may monitor the status of any variable sources of power that could have an impact on its power requirements as a Sink such as Batteries, AC and Bulk supplies. When monitored, and a USB interface is supported, the battery state shall be reported to the System Policy Manager using the USB interface.

8.2.7 Interface to the Policy Engine

The Device Policy Manager shall maintain an interface to the Policy Engine for each port in the Device.

8.2.7.1 Device Policy Manager in a Provider

The Device Policy Manager in a Provider shall also provide the following functions to the Policy Engine:

- Inform the Policy Engine of changes in cable/Device attachment status for a given cable.
- Inform the Policy Engine whenever the Source capabilities available for a port change.
- Evaluate requests from an attached Consumer and provide responses to the Policy Engine.
- Respond to requests for Power Supply transitions from the Policy Engine.
- Indication to Policy Engine when Power Supply transitions are complete.
- Maintain reserve capability for Devices operating on a port at less than maximum power.

8.2.7.2 Device Policy Manager in a Consumer

The Device Policy Manager in a Consumer shall also provide the following functions to the Policy Engine:

- Inform the Policy Engine of changes in cable/Device attachment status as well as the IR drop for a given cable.
- Inform the Policy Engine whenever the power requirements for a port change.
- Evaluate Source capabilities and provide suitable responses:
    - Request from offered capabilities
    - Indicate whether additional power is required
- Respond to requests for Sink transitions from the Policy Engine.

8.2.7.3 Device Policy Manager in a Provider/Consumer

The Device Policy Manager in a Provider/Consumer shall provide the following functions to the Policy Engine:

- Provider Device Policy Manager
- Consumer Device Policy Manager
- Interface for the Policy Engine to request Power Supply transitions from Source to Sink and vice versa.
- Indications to Policy Engine during Power Swap transitions.

8.2.7.4 Device Policy Manager in a Provider/Consumer dead battery handling

The Device Policy Manager in a Provider/Consumer with a battery should also provide:

- Detection and handling of back powering in the case of a dead battery (see Section8.2.7.4).

In this scenario a Provider/Consumer shall:

- Detect that V_{BUS} is present and that its battery is dead
- Switch to Consumer role without using a ***Swap*** message

- Use V_{BUS} to power the USB Power Delivery communications

8.2.7.5 Device Policy Manager in a Consumer/Provider

The Device Policy Manager in a Consumer/Provider shall provide the following functions to the Policy Engine:

- Consumer Device Policy Manager
- Provider Device Policy Manager
- Interface for the Policy Engine to request Power Supply transitions from Sink to Source and vice versa.
- Indications to Policy Engine during Power Swap transitions.

8.2.7.6 Device Policy Manager in a Consumer/Provider dead battery handling

The Device Policy Manager in a Consumer/Provider shall also provide:

- Detection and handling of back powering in the case of a dead battery (see Section 4.1).

In this scenario a Consumer/Provider shall:

- Detect that a Provider/Consumer with a dead battery capable of being back powered is attached
- Check that V_{BUS} is not present.
- Temporarily output ***vSafeDB*** on V_{BUS} to provide power to the Provider/Consumer so that it can send a Bit Stream.
- Detect Bit-Stream sent by Provider/Consumer.
- Switch to the Provider role without using a Swap messageand ouput ***vSafe5V*** on V_{BUS} that will power the Provider/Consumer's PD communications.

8.3 Policy Engine

8.3.1 Introduction

There is one Policy Engine per port that interacts with the Device Policy Manager in order to implement the present Local Policy for that particular port. This section includes:

- Message sequences for various operations
- A state diagram of the Policy Engine for a Source Port
- A state diagram of the Policy Engine for a Sink Port
- A state diagrams of the Policy Engine for Dual-Role Ports
- State diagrams for handling Soft Reset
- State diagrams for handling ***Ping*** messages
- State diagrams for the Provider/Consumer and Consumer/Provider dead-battery/back-powering case
- State diagrams for Hard Reset
- State diagrams for BIST

8.3.2 Message Sequence Diagrams

The Device Policy Engine drives the message sequences and responses based on both the expected message sequences and the present Local Policy. This section contains sequence diagrams that highlight some of the more interesting transactions. It is by no means a complete summary of all possible combinations, but is illustrative in nature.

8.3.2.1 Basic Message Exchange

Figure 8-1 Basic Message Exchange (Successful) below illustrates how a message is sent. Note that the sender may be either a Source or Sink while the receiver may be either a Sink or Source. The basic message sequence is the same. It starts when the Message Sender's Protocol Layer at the behest of its Policy Engine forms a message that it passes to the Physical Layer.

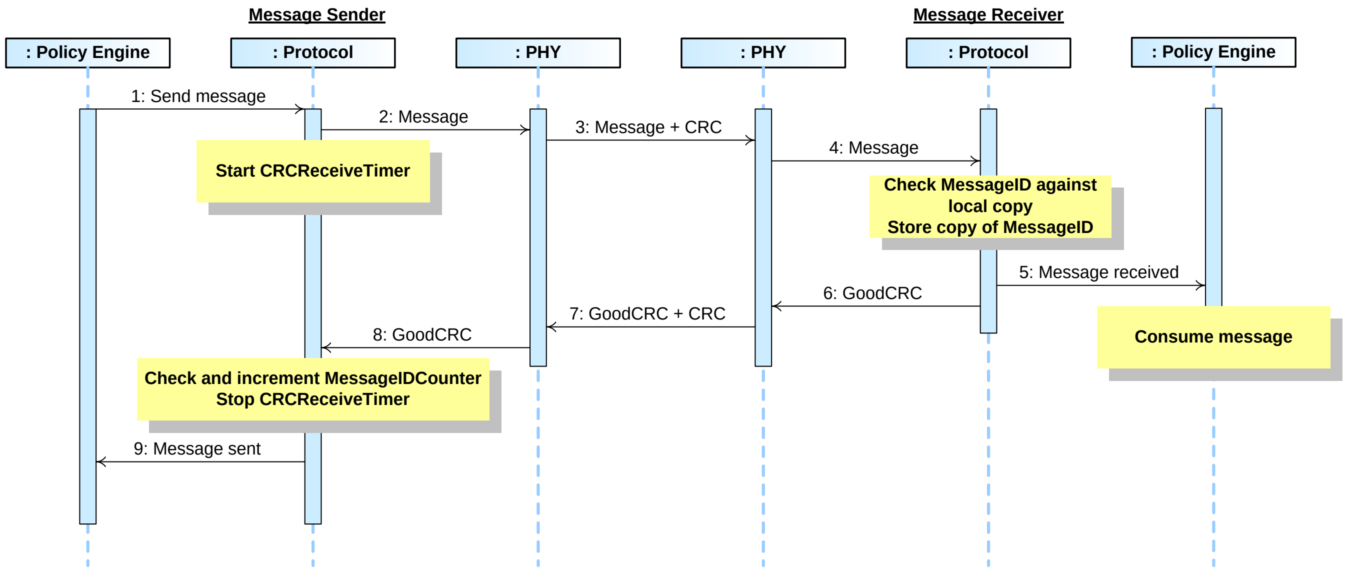


Figure 8-1 Basic Message Exchange (Successful)

Table 8-1 Basic Message Flow

Step	Message Sender	Message Receiver
1	Policy Engine directs Protocol Layer to send a message.	
2	Protocol Layer creates the message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
3	Physical Layer appends a CRC and sends the message.	Physical Layer receives the message and checks the CRC to verify the message.
4		Physical Layer removes the CRC and forwards the message to the Protocol Layer.
5		Protocol Layer checks the ***MessageID*** in the incoming message is different from the previously stored value and then stores a copy of the new value. Protocol Layer forwards the received message information to the Policy Engine that consumes it.
6		Protocol Layer generates a ***GoodCRC*** message and passes it to the Physical Layer.
7	Physical Layer receives the message and checks the CRC to verify the message.	Physical Layer appends CRC and sends the ***GoodCRC*** message.
8	Physical Layer removes the CRC and forwards the ***GoodCRC*** message to the Protocol Layer. Protocol Layer checks and increments the ***MessageIDCounter*** and stops	

Step	Message Sender	Message Receiver
	CRCReceiveTimer.	
9	Protocol Layer informs the Policy Engine that the message was successfully sent.	

8.3.2.2 Errors in Basic Message flow

There are various points during the message flow where failures in communication or other issues can occur. Figure 8-2 is an annotated version of Figure 8-1 indicating at which points issues may occur.

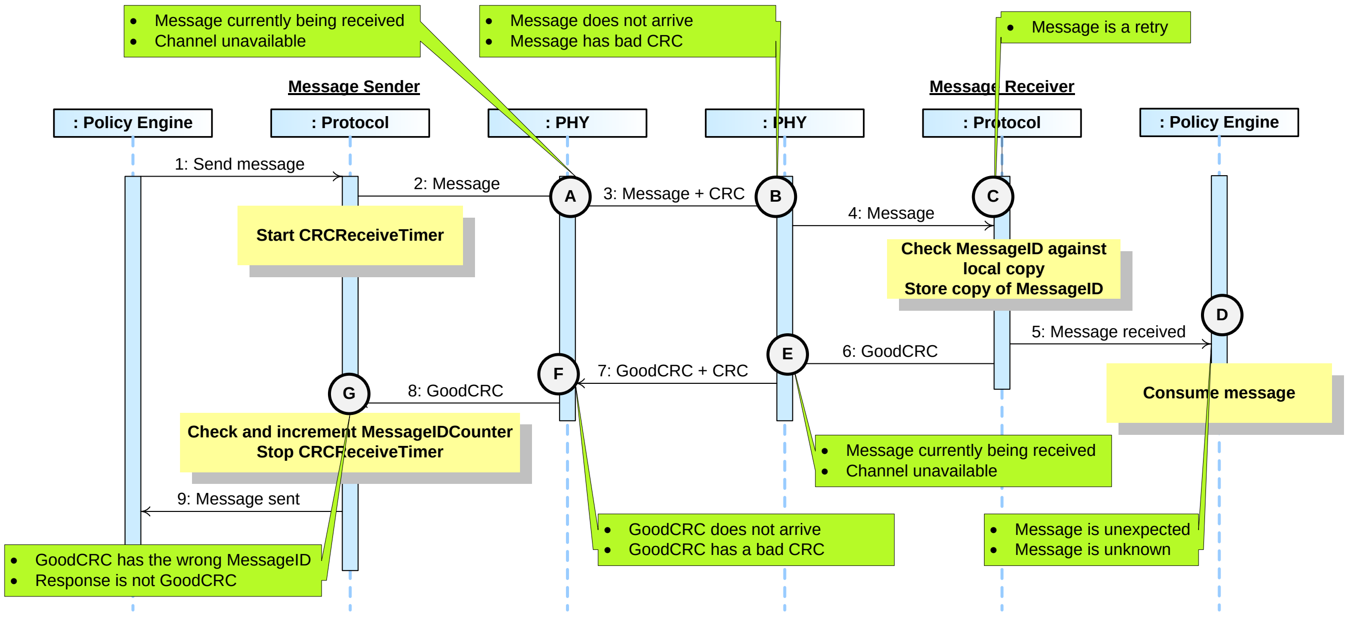


Figure 8-2 Basic Message flow indicating possible errors

Table 8-2 Potential issues in Basic Message Flow

Point	Possible issues
A	1. There is an incoming message on the channel meaning that the Physical Layer is unable to send. In this case the outgoing message shall be removed from the queue and the incoming message processed. 2. Due to some sort of noise on the line it is not possible to transmit. In this case the outgoing message shall be removed from the queue. Retransmission is via the Protocol Layer's normal mechanism.
B	1. Message does not arrive at the Physical Layer due to noise on the channel. 2. Message arrives but has been corrupted and has a bad CRC. There is no message to passed up to the Protocol Layer on the receiver which means a ***GoodCRC*** message is not sent. This leads to a ***CRCReceiveTimer*** timeout in the Message Sender.
C	1. ***MessageID*** of received message matches stored ***MessageID*** so this is a retry. Message is not passed up to the Policy Engine.
D	1. Policy Engine receives a known message that it was not expecting. 2. Policy Engine receives an unknown (unrecognised) message. These cases are errors in the protocol which leads to the generation of a ***Protocol Error*** message.

Point	Possible issues
E	Same as point A but at the Message Receiver side.
F	1. ***GoodCRC*** message response does not arrive Message Sender side due to the noise on the channel. 2. ***GoodCRC*** message response arrives but has a bad CRC. A ***GoodCRC*** message is not received by the Message Sender's Protocol Layer. This leads to a ***CRCReceiveTimer*** timeout in the Message Sender.
G	1. ***GoodCRC*** message is received but does contain the same ***MessageID*** as the transmitted message. 2. A message is received but it is not a ***GoodCRC*** message (similar case to that of an unexpected or unknown message but this time detected in the Protocol Layer). Both of these issues indicate errors in receiving an expected ***GoodCRC*** message which will lead to a ***CRCReceiveTimer*** timeout in the Protocol Layer and a subsequent retry.

Figure 8-3 illustrates one of these cases; the basic message flow with a retry due to a bad CRC at the Message Receiver. It starts when the Message Sender's Protocol Layer at the behest of its Policy Engine forms a message that it passes to the Physical Layer. The Protocol Layer is responsible for retries on a "3 strikes and you are out" basis (***nRetryCount***).

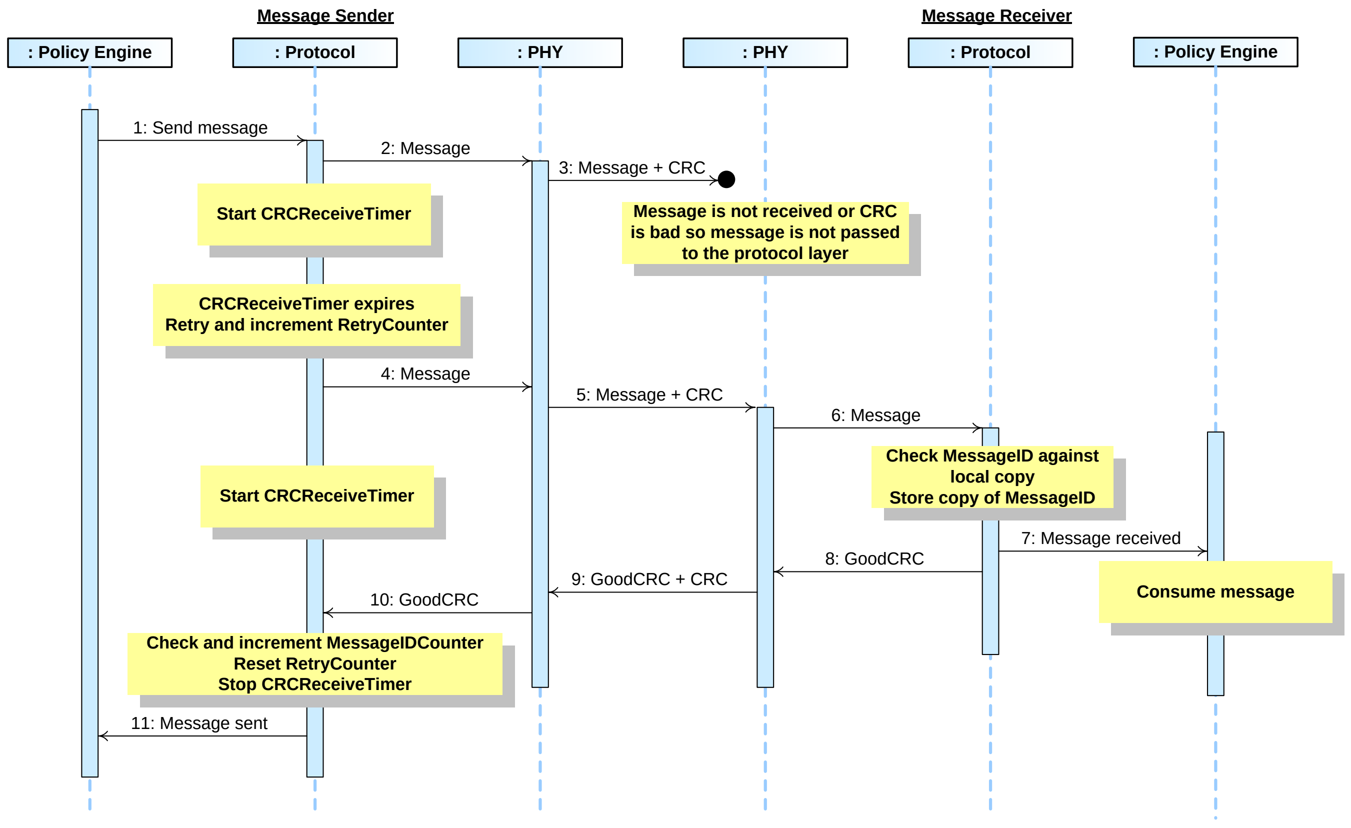


Figure 8-3 Basic Message Flow with Bad CRC followed by a Retry

Table 8-3 Basic Message Flow with CRC failure

Step	Message Sender	Message Receiver
1	Policy Engine directs Protocol Layer to send a message.	
2	Protocol Layer creates the message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
3	Physical Layer appends a CRC and sends the message.	Physical Layer receives no message or a message with an incorrect CRC. Nothing is passed to Protocol Layer.
4	Since no response is received, the ***CRCReceiveTimer*** will expire and trigger the first retry by the Protocol Layer. The ***RetryCounter*** is incremented. Protocol Layer passes the message to the Physical Layer. Starts ***CRCReceiveTimer***.	
5	Physical Layer appends a CRC and sends the message.	Physical Layer receives the message and checks the CRC to verify the message.
6		Physical Layer removes the CRC and forwards the message to the Protocol Layer.
7		Protocol Layer checks the ***MessageID*** in the incoming message is different from the previously stored value and then stores a copy of the new value. Protocol Layer forwards the received message information to the Policy Engine that consumes it.
8		Protocol Layer generates a ***GoodCRC*** message and passes it to the Physical Layer.
9	Physical Layer receives the message and checks the CRC to verify the message.	Physical Layer appends CRC and sends the ***GoodCRC*** message.
10	Physical Layer removes the CRC and forwards the ***GoodCRC*** message to the Protocol Layer.	
11	Protocol Layer verifies the ***MessageID***, stops ***CRCReceiveTimer*** and resets the ***RetryCounter***. Protocol Layer informs the Policy Engine that the message was successfully sent.	

8.3.2.3 Power Negotiation

Figure 8-4 illustrates an example of a successful message flow during Power Negotiation. The negotiation goes through 5 distinct phases:

- The Source sends out its power capabilities in a ***Capabilities*** message.
- The Sink evaluates these capabilities and in the request phase selects one power level by sending a ***Request*** message.
- The Source evaluates the request and accepts the request with an ***Accept*** message.
- The Source transitions to the new power level and then informs the Sink by sending a ***PS_RDY*** message.
- The Sink starts using the new power level.

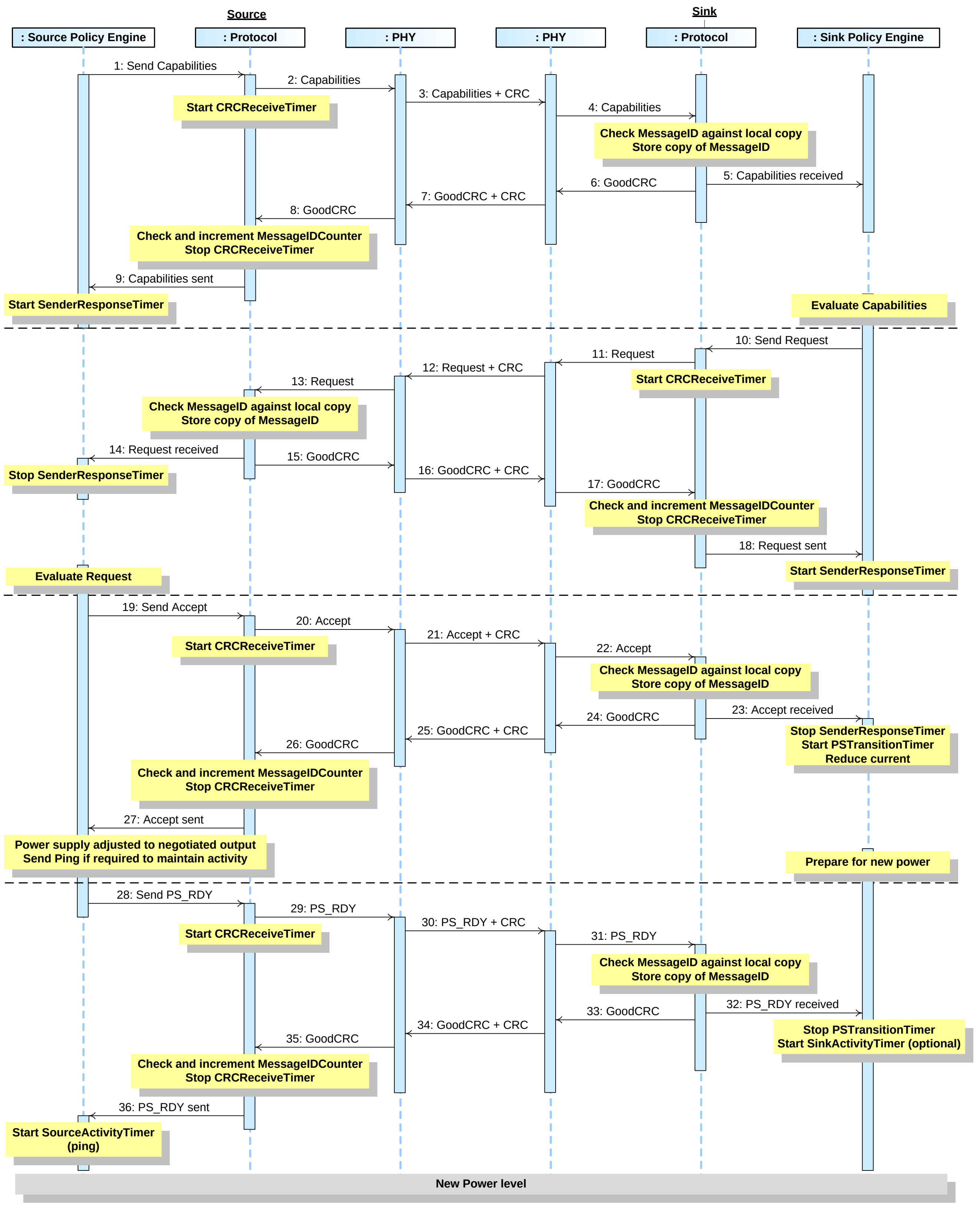


Figure 8-4 Successful Power Negotiation

Table 8-4 below provides a detailed explanation of what happens at each labeled step in Figure 8-4 above.

Table 8-4 Steps for a successful Power Negotiation

Step	Source	Sink
1	Policy Engine directs the Protocol Layer to send a ***Capabilities*** message that represents the power supply's present capabilities.	
2	Protocol Layer creates the message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
3	Physical Layer appends CRC and sends the ***Capabilities*** message.	Physical Layer receives the ***Capabilities*** message and checks the CRC to verify the message.
4		Physical Layer removes the CRC and forwards the ***Capabilities*** message to the Protocol Layer.
5		Protocol Layer checks the ***MessageID*** in the incoming message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received ***Capabilities*** message information to the Policy Engine that consumes it.
6		Protocol Layer generates a ***GoodCRC*** message and passes it Physical Layer.
7	Physical Layer receives the ***GoodCRC*** message and checks the CRC to verify the message.	Physical Layer appends CRC and sends the ***GoodCRC*** message.
8	Physical Layer removes the CRC and forwards the ***GoodCRC*** message to the Protocol Layer.	
9	Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Capabilities*** message was successfully sent. Policy Engine starts ***SenderResponseTimer***.	
10		Policy Engine evaluates the ***Capabilities*** message sent by the Source and selects which power it would like. It tells the Protocol Layer to form the data (e.g. Power Data Object) that represents its Request into a message.
11		Protocol Layer creates the ***Request*** message and passes to Physical Layer. Starts ***CRCReceiveTimer***.
12	Physical Layer receives the ***Request*** message and compares the CRC it calculated with the one sent to verify the message.	Physical Layer appends a CRC and sends the ***Request*** message.
13	Physical Layer removes the CRC and forwards the ***Request*** message to the Protocol Layer.	
14	Protocol Layer checks the ***MessageID*** in the incoming message is different from the previously stored value and then stores a copy	

Step	Source	Sink
	of the new value. The Protocol Layer passes the Request information to the Policy Engine. Policy Engine stops ***SenderResponseTimer***.	
15	The Protocol Layer generates a ***GoodCRC*** message and passes it to its Physical Layer.	
16	Physical Layer appends CRC and sends the message.	Physical Layer receives the message and compares the CRC it calculated with the one sent to verify the message.
17		Physical Layer forwards the ***GoodCRC*** message to the Protocol Layer.
18		The protocol Layer verifies and increments the ***MessageIDCounter***. It informs the Policy Engine that the ***Request*** message was successfully sent. The Protocol Layer stops the ***CRCReceiveTimer***. The Policy Engine starts ***SenderResponseTimer***.
19	Policy Engine evaluates the ***Request*** message sent by the Sink and decides if it can meet the request. It tells the Protocol Layer to form an ***Accept*** message.	
20	The Protocol Layer forms the ***Accept*** message that is passed to the Physical Layer and starts the ***CRCReceiveTimer***.	
21	Physical Layer appends CRC and sends the ***Accept*** message.	Physical Layer receives the message and compares the CRC it calculated with the one sent to verify the message.
22		Physical Layer forwards the ***Accept*** message to the Protocol Layer.
23		Protocol Layer checks the ***MessageID*** in the incoming message is different from the previously stored value and then stores a copy of the new value. Protocol Layer informs the Policy Engine that an ***Accept*** message has been received. The Policy Engine stops ***SenderResponseTimer***, starts the ***PSTransitionTimer*** and reduces its current draw. The Device Policy Manager prepares the Power supply for transition to the new power level.
24		The Protocol Layer generates a ***GoodCRC*** message and passes it to its Physical Layer.
25	Physical Layer receives the message and compares the CRC it calculated with the one sent to verify the message.	Physical Layer appends CRC and sends the message.
26	Physical Layer forwards the ***GoodCRC*** message to the Protocol Layer. The Protocol Layer	

Step	Source	Sink
	verifies and increments the ***MessageIDCounter*** and stops the ***CRCReceiveTimer***.	
27	The Protocol Layer informs the Policy Engine that an ***Accept*** message was successfully sent.	
	Power Supply Adjusts its Output to the Negotiated Value	
28	The Device Policy Manager informs the Policy Engine that the power supply has settled at the new operating condition and tells the Protocol Layer to send a ***PS_RDY*** message. If the time taken to settle exceeds ***SourceActivityTimer tSourceActivity*** then a ***Ping*** message will be sent.	
29	The Protocol Layer forms the ***PS_RDY*** message and starts the ***CRCReceiveTimer***.	
30	Physical Layer appends CRC and sends the ***PS_RDY*** message.	Physical Layer receives the message and compares the CRC it calculated with the one sent to verify the message.
31		Physical Layer forwards the message to the Protocol Layer.
32		Protocol Layer checks the ***MessageID*** in the incoming message is different from the previously stored value and then stores a copy of the new value. Protocol Layer informs the Policy Engine that a RS_RDY has been received. The Policy Engine stops the ***PSTransitionTimer*** and optionally starts the ***SinkActivityTimer*** to monitor for ***Ping*** message timeout.
33		The Protocol Layer generates a ***GoodCRC*** message and passes it to its Physical Layer.
34	Physical Layer receives the message and compares the CRC it calculated with the one sent to verify the message.	Physical Layer appends CRC and sends the message.
35	Physical Layer forwards the ***GoodCRC*** message to the Protocol Layer. The Protocol Layer verifies and increments the ***MessageIDCounter***. Stops the ***CRCReceiveTimer***.	Physical Layer. Starts the CRCTimer and ResponseTimer.
36	The Protocol Layer informs the Policy Engine that the ***PS_RDY*** message was successfully sent. The Policy Engine starts the ***SourceActivityTimer*** in order to start pinging.	

8.3.2.4 Source Initiated Swap without subsequent Power Negotiation

This is an example of a successful Swap operation initiated by the Source. It does not include any subsequent Power Negotiation such as would be required for higher voltage or current which likely will follow (see previous section for the details of a Power Negotiation).

There are four distinct phases to the power swap negotiation:

- A ***Swap*** message is sent.
- An ***Accept*** message in response to the ***Swap*** message.
- The original Source sets its power output to ***vSafe0V*** and sends a ***PS_RDY*** message when it gets there.
- The new Source then sets its power output to ***vSafe5V*** and sends a ***PS_RDY*** message when it is ready to supply power.

Figure 8-5 shows the messages as they flow across the bus and within the Devices to accomplish the Swap sequence.

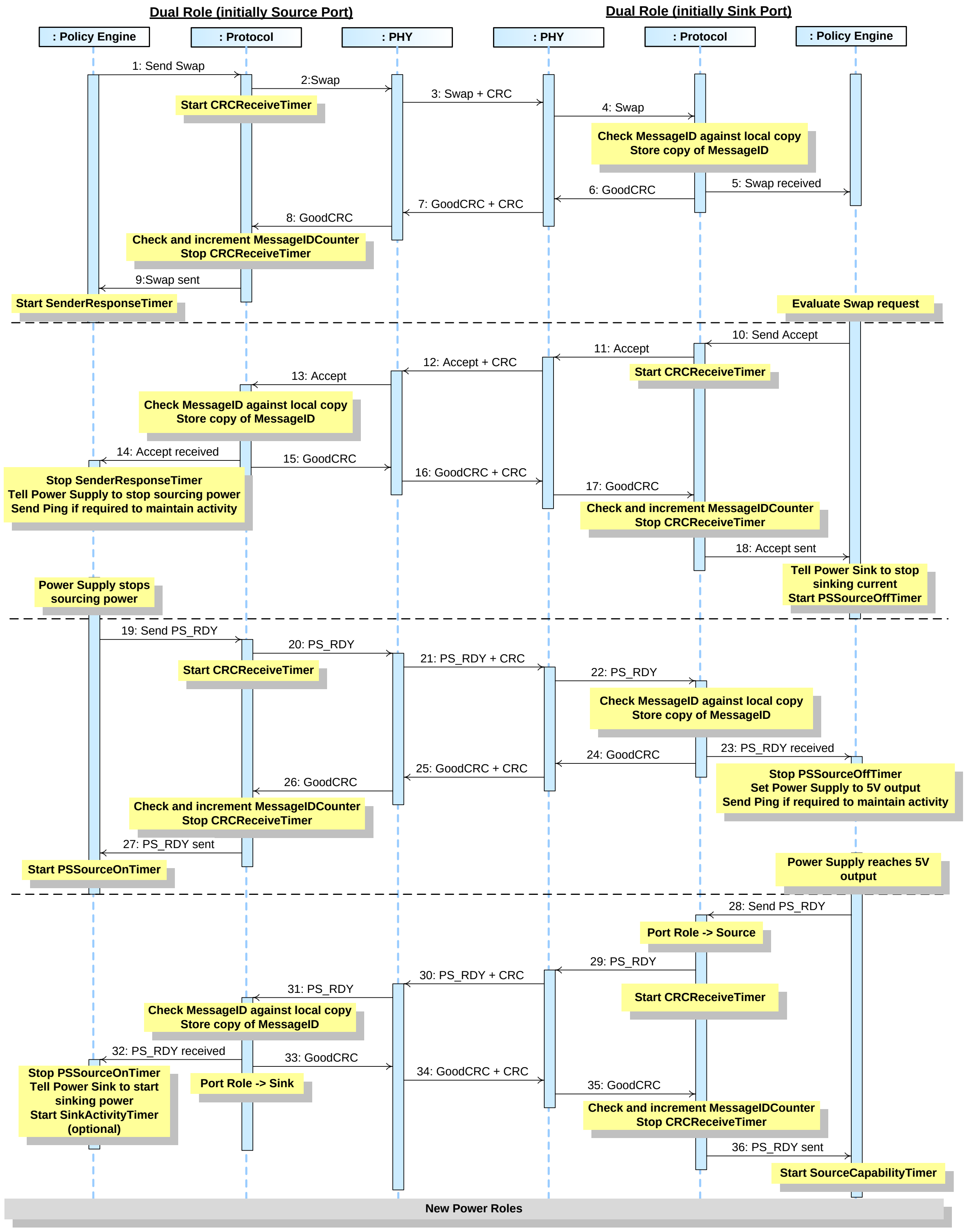


Figure 8-5 Successful Swap Sequence Initiated by the Source

Table 8-5 below provides a detailed explanation of what happens at each labeled step in Figure 8-5 above.

Table 8-5 Steps for a Successful Source Initiated Swap Sequence

Step	Dual-Role (initially Source Port)	Dual-Role (initially Sink Port)
1	Policy Engine directs the Protocol Layer to send a ***Swap*** message.	
2	Protocol Layer creates the message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
3	Physical Layer appends CRC and sends the ***Swap*** message.	Physical Layer receives the ***Swap*** message and checks the CRC to verify the message.
4		Physical Layer removes the CRC and forwards the ***Swap*** message to the Protocol Layer.
5		Protocol Layer checks the ***MessageID*** in the incoming message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received ***Swap*** message information to the Policy Engine that consumes it.
6		Protocol Layer generates a ***GoodCRC*** message and passes it Physical Layer.
7	Physical Layer receives the ***GoodCRC*** message and checks the CRC to verify the message.	Physical Layer appends CRC and sends the ***GoodCRC*** message.
8	Physical Layer removes the CRC and forwards the ***GoodCRC*** message to the Protocol Layer.	
9	Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Swap*** message was successfully sent. Policy Engine starts ***SenderResponseTimer***.	
10		Policy Engine evaluates the ***Swap*** message sent by the Source and decides that it is able and willing to do the Role Swap. It tells the Protocol Layer to form an ***Accept*** message.
11		Protocol Layer creates the message and passes to Physical Layer. Starts ***CRCReceiveTimer***.
12	Physical Layer receives the message and compares the CRC it calculated with the one sent to verify the message.	Physical Layer appends a CRC and sends the message over V_{BUS}.
13	Protocol Layer checks the ***MessageID*** in the incoming message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received ***Swap*** message information to the Policy Engine that	

Step	Dual-Role (initially Source Port)	Dual-Role (initially Sink Port)
	consumes it.	
14	The Policy Engine requests its Power Supply to stop supplying power. Stops ***SenderResponseTimer***. If the time taken to stop supplying power exceeds ***tSourceActivity*** then a ***Ping*** message will be sent.	
15	Protocol Layer generates a ***GoodCRC*** message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the ***GoodCRC*** message over V_{BUS}.	Physical Layer receives ***GoodCRC*** message and compares the CRC it calculated with the one sent to verify the message.
17		Physical Layer removes the CRC and forwards the ***GoodCRC*** message to the Protocol Layer.
18		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Accept*** message was successfully sent. The Policy Engine tells the Power supply to stop sinking current and starts ***PSSourceOffTimer***.
19	The Policy Engine determines its power supply is no longer supplying V_{BUS}, it directs the Protocol Layer to generate a ***PS_RDY*** message to tell its Port Partner that it can begin to Source V_{BUS}.	
20	Protocol Layer creates the message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
21	Physical Layer appends CRC and sends the ***PS_RDY*** message.	Physical Layer receives the ***PS_RDY*** message and checks the CRC to verify the message.
22		Physical Layer removes the CRC and forwards the ***PS_RDY*** message to the Protocol Layer.
23		Protocol Layer checks the ***MessageID*** in the incoming message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received ***PS_RDY*** message information to the Policy Engine that consumes it. The Policy Engine stops the ***PSSourceOffTimer*** and starts switching the power supply to ***vSafe5V*** Source operation. If the time taken to start supplying power exceeds ***tSourceActivity*** then a ***Ping*** message will be sent.
24		Protocol Layer generates a ***GoodCRC*** message and passes it Physical Layer.

Step	Dual-Role (initially Source Port)	Dual-Role (initially Sink Port)
25	Physical Layer receives the ***GoodCRC*** message and checks the CRC to verify the message.	Physical Layer appends CRC and sends the ***GoodCRC*** message.
26	Physical Layer removes the CRC and forwards the ***GoodCRC*** message to the Protocol Layer.	
27	Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***PS_RDY*** message was successfully sent. Policy Engine starts ***PSSourceOnTimer***.	
28		Policy Engine, when its power supply is ready to supply power, tells the Protocol Layer to form a ***PS_RDY*** message. The ***Port Role*** bit used in this and subsequent message headers is now set to "Source".
29		Protocol Layer creates the message and passes to Physical Layer. Starts ***CRCReceiveTimer***.
30	Physical Layer receives the message and compares the CRC it calculated with the one sent to verify the message.	Physical Layer appends a CRC and sends the message over V_{BUS}.
31	Physical Layer removes the CRC and forwards the ***GoodCRC*** message to the Protocol Layer.	
32	Protocol Layer checks the ***MessageID*** in the incoming message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received ***PS_RDY*** message information to the Policy Engine that consumes it. The Policy Engine stops the ***PSSourceOnTimer***, informs the power supply that it can start consuming power and optionally starts the ***SinkActivityTimer*** to check for ***Ping*** messages.	
33	Protocol Layer generates a ***GoodCRC*** message and passes it Physical Layer. Subsequent messages will be sent with the ***Port Role*** bit in the header set to "Sink".	
34	Physical Layer appends a CRC and sends the ***GoodCRC*** message over V_{BUS}.	Physical Layer receives ***GoodCRC*** message and compares the CRC it calculated with the one sent to verify the message.
35		Physical Layer removes the CRC and forwards the ***GoodCRC*** message to the Protocol Layer.

Step	Dual-Role (initially Source Port)	Dual-Role (initially Sink Port)
36		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***PS_RDY*** message was successfully sent. The Policy Engine starts the ***SourceCapabilityTimer*** in order to generate ***Capabilities*** messages.
	The Swap is complete, the roles have been reversed and the ports are free to negotiate for more power.	

8.3.2.5 Sink Initiated Swap without subsequent Power Negotiation

This is an example of a successful Swap operation initiated by the Sink. It does not include any subsequent Power Negotiation such as would be required for higher voltage or current which likely will follow (see previous section for the details of a Power Negotiation).

There are four distinct phases to the power swap negotiation:

1) A ***Swap*** message is sent.
2) An ***Accept*** message in response to the ***Swap*** message.
3) The original Source sets its power output to ***vSafe0V*** and sends a ***PS_RDY*** message when it gets there.
4) The new Source then sets its power output to ***vSafe5V*** and sends a ***PS_RDY*** message when it is ready to supply power.

Figure 8-6 shows the messages as they flow across the bus and within the Devices to accomplish the Role Swap.

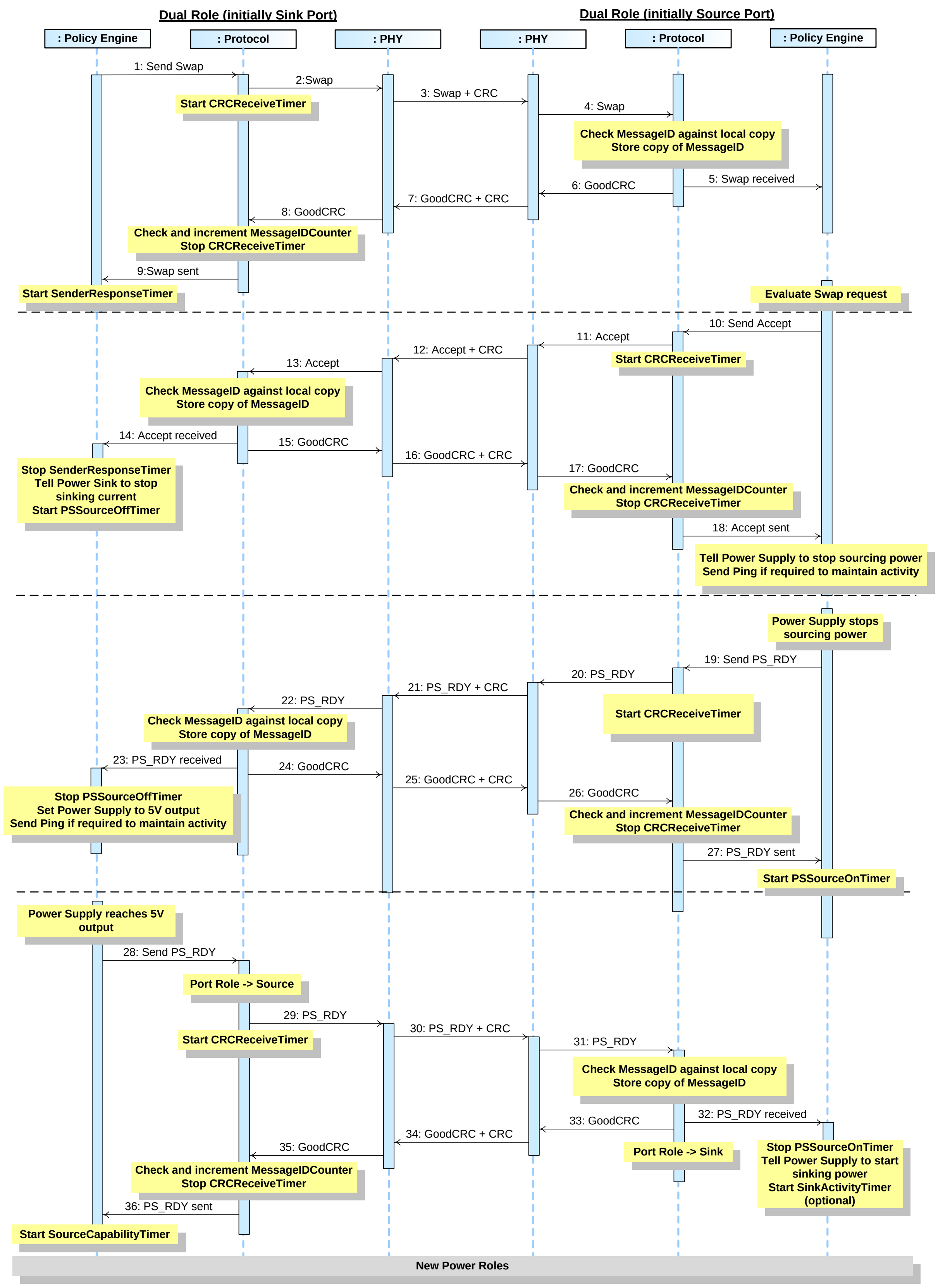


Figure 8-6 Successful Swap Sequence Initiated by the Sink

Table 8-6 Steps for a Successful Sink Initiated Swap Sequence

Step	Dual-Role (initially Sink Port)	Dual-Role (initially Source Port)
1	Policy Engine directs the Protocol Layer to send a ***Swap*** message.	
2	Protocol Layer creates the message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
3	Physical Layer appends CRC and sends the ***Swap*** message.	Physical Layer receives the ***Swap*** message and checks the CRC to verify the message.
4		Physical Layer removes the CRC and forwards the ***Swap*** message to the Protocol Layer.
5		Protocol Layer checks the ***MessageID*** in the incoming message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received ***Swap*** message information to the Policy Engine that consumes it.
6		Protocol Layer generates a ***GoodCRC*** message and passes it Physical Layer.
7	Physical Layer receives the ***GoodCRC*** message and checks the CRC to verify the message.	Physical Layer appends CRC and sends the ***GoodCRC*** message.
8	Physical Layer removes the CRC and forwards the ***GoodCRC*** message to the Protocol Layer.	
9	Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Swap*** message was successfully sent. Policy Engine starts ***SenderResponseTimer***.	
10		Policy Engine evaluates the ***Swap*** message sent by the Sink and decides that it is able and willing to do the Role Swap. It tells the Protocol Layer to form an ***Accept*** message.
11		Protocol Layer creates the message and passes to Physical Layer. Starts ***CRCReceiveTimer***.
12	Physical Layer receives the message and compares the CRC it calculated with the one sent to verify the message.	Physical Layer appends a CRC and sends the message over V_{BUS}.
13	Protocol Layer checks the ***MessageID*** in the incoming message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received ***Swap*** message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the ***SenderResponseTimer***, tells the Power supply	

Step	Dual-Role (initially Sink Port)	Dual-Role (initially Source Port)
	to stop sinking current and starts ***PSSourceOffTimer***.	
15	Protocol Layer generates a ***GoodCRC*** message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the ***GoodCRC*** message over V_{BUS}.	Physical Layer receives ***GoodCRC*** message and compares the CRC it calculated with the one sent to verify the message.
17		Physical Layer removes the CRC and forwards the ***GoodCRC*** message to the Protocol Layer.
18		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Accept*** message was successfully sent. The Policy Engine tells the Power supply to stop supplying power. If the time taken to stop supplying power exceeds ***tSourceActivity*** then a ***Ping*** message will be sent.
19		The Policy Engine determines its power supply is no longer supplying V_{BUS}, it directs the Protocol Layer to generate a ***PS_RDY*** message to tell its Port Partner that it can begin to source V_{BUS}.
20		Protocol Layer creates the message and passes to Physical Layer. Starts ***CRCReceiveTimer***.
21	Physical Layer receives the ***PS_RDY*** message and checks the CRC to verify the message.	Physical Layer appends CRC and sends the ***PS_RDY*** message.
22	Physical Layer removes the CRC and forwards the ***PS_RDY*** message to the Protocol Layer.	
23	Protocol Layer checks the ***MessageID*** in the incoming message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received ***PS_RDY*** message information to the Policy Engine that consumes it. The Policy Engine stops the ***PSSourceOffTimer*** and starts switching the power supply to ***vSafe5V*** Source operation. If the time taken to start supplying power exceeds ***tSourceActivity*** then a ***Ping*** message will be sent.	
24	Protocol Layer generates a ***GoodCRC*** message and passes it Physical Layer.	
25	Physical Layer appends CRC and sends the ***GoodCRC*** message.	Physical Layer receives the ***GoodCRC*** message and checks the CRC to verify the message.
26		Physical Layer removes the CRC and forwards

Step	Dual-Role (initially Sink Port)	Dual-Role (initially Source Port)
		the ***GoodCRC*** message to the Protocol Layer.
27		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***PS_RDY*** message was successfully sent. Policy Engine starts ***PSSourceOnTimer***.
28	Policy Engine, when its power supply is ready to supply power, tells the Protocol Layer to form a ***PS_RDY*** message. The ***Port Role*** bit used in this and subsequent message headers is now set to "Source".	
29	Protocol Layer creates the message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
30	Physical Layer appends a CRC and sends the message over V_{BUS}.	Physical Layer receives the message and compares the CRC it calculated with the one sent to verify the message.
31		Physical Layer removes the CRC and forwards the ***GoodCRC*** message to the Protocol Layer.
32		Protocol Layer checks the ***MessageID*** in the incoming message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received ***PS_RDY*** message information to the Policy Engine that consumes it. The Policy Engine stops the ***PSSourceOnTimer***, informs the power supply that it can start consuming power and optionally starts the ***SinkActivityTimer*** to check for ***Ping*** messages.
33		Protocol Layer generates a ***GoodCRC*** message and passes it Physical Layer. Subsequent messages will be sent with the ***Port Role*** bit in the header set to "Sink".
34	Physical Layer receives ***GoodCRC*** message and compares the CRC it calculated with the one sent to verify the message.	Physical Layer appends a CRC and sends the ***GoodCRC*** message over V_{BUS}.
35	Physical Layer removes the CRC and forwards the ***GoodCRC*** to the Protocol Layer.	
36	Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***PS_RDY*** message was successfully sent. The Policy Engine starts the ***SourceCapabilityTimer*** in order to generate ***Capabilities*** messages.	
	The Role Swap is complete, the roles have been reversed and the ports are free to negotiate for more power.	

8.3.2.6 Reclaiming Power with GotoMin message

This is an example of a GotoMin operation. Figure 8-7 shows the messages as they flow across the bus and within the Devices to accomplish the GotoMin. The following figure illustrates the case where the Sink receives the request.

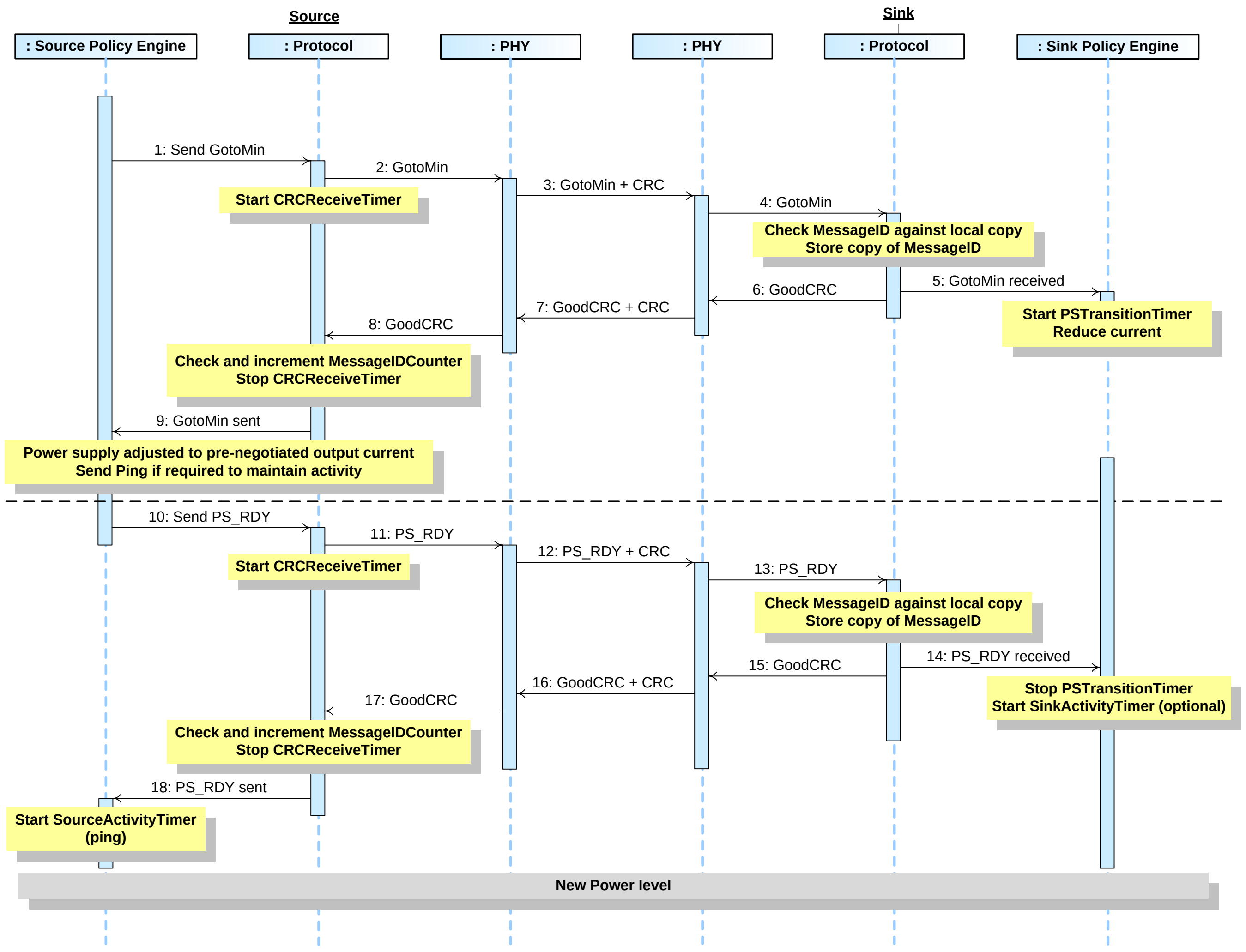


Figure 8-7 Successful GotoMin operation

The table below provides a detailed explanation of what happens at each labeled step in Figure 8-7 above.

Table 8-7 Steps for a GotoMin Negotiation

Step	Source	Sink
1	Policy Engine tells the Protocol Layer to form a ***GotoMin*** message.	
2	The Protocol Layer forms the ***Accept*** message that is passed to the Physical Layer and starts the ***CRCReceiveTimer***.	
3	Physical Layer appends CRC and sends the ***GotoMin*** message.	Physical Layer receives the message and compares the CRC it calculated with the one sent

Step	Source	Sink
		to verify the message.
4		Physical Layer forwards the ***GotoMin*** message to the Protocol Layer.
5		Protocol Layer checks the ***MessageID*** in the incoming message is different from the previously stored value and then stores a copy of the new value. Protocol Layer informs the Policy Engine that a ***GotoMin*** message has been received. The Policy starts the ***PSTransitionTimer*** and reduces its current draw. The Policy Engine prepares the Power supply for transition to the new power level.
6		The Protocol Layer generates a ***GoodCRC*** message and passes it to its Physical Layer.
7	Physical Layer receives the message and compares the CRC it calculated with the one sent to verify the message.	Physical Layer appends CRC and sends the message.
8	Physical Layer forwards the ***GoodCRC*** message to the Protocol Layer. The Protocol Layer verifies and increments the ***MessageIDCounter*** and stops the ***CRCReceiveTimer***.	
9	The Protocol Layer informs the Policy Engine that an ***Accept*** message was successfully sent.	
Power Supply Adjusts its Output to the Negotiated Value		
10	Policy Engine sees the power supply has settled at the new operating condition and tells the Protocol Layer to send a ***PS_RDY*** message. If the time taken to settle exceeds ***tSourceActivity*** then a ***Ping*** message will be sent.	
11	The Protocol Layer forms the ***PS_RDY*** message and starts the ***CRCReceiveTimer***.	
12	Physical Layer appends CRC and sends the ***PS_RDY*** message.	Physical Layer receives the message and compares the CRC it calculated with the one sent to verify the message.
13		Physical Layer forwards the message to the Protocol Layer.
14		Protocol Layer checks the ***MessageID*** in the incoming message is different from the previously stored value and then stores a copy of the new value. Protocol Layer informs the Policy Engine that a ***PS_RDY*** message has been received. The Policy Engine stops the ***PSTransitionTimer*** and optionally starts the ***SinkActivityTimer*** to monitor for ***Ping*** message timeout.

Step	Source	Sink
15		The Protocol Layer generates a ***GoodCRC*** message and passes it to its Physical Layer.
16	Physical Layer receives the message and compares the CRC it calculated with the one sent to verify the message.	Physical Layer appends CRC and sends the message.
17	Physical Layer forwards the ***GoodCRC*** message to the Protocol Layer. The Protocol Layer verifies and increments the ***MessageIDCounter*** and stops the ***CRCReceiveTimer***.	
18	The Protocol Layer informs the Policy Engine that the ***PS_RDY*** message was successfully sent. The Policy Engine starts the ***SourceActivityTimer*** in order to start pinging.	

8.3.2.7 Soft Reset

This is an example of a Soft Reset operation. Figure 8-8 shows the messages as they flow across the bus and within the Devices to accomplish the Soft Reset.

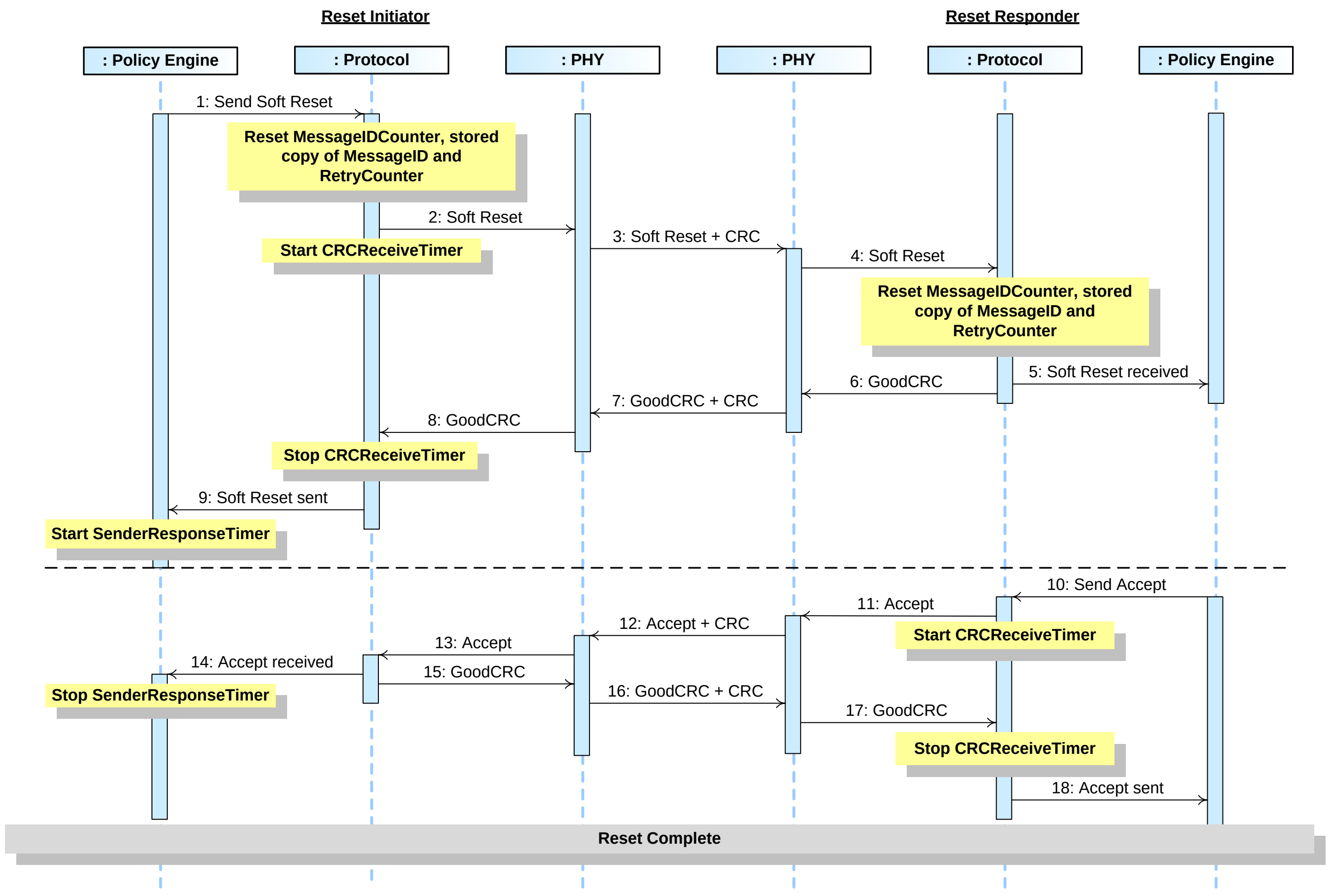


Figure 8-8 Soft Reset

Table 8-8 below provides a detailed explanation of what happens at each labeled step in Figure 8-8 above.

Table 8-8 Steps for a Soft Reset

Step	Reset Initiator	Reset Responder
1	The Policy Engine directs the Protocol Layer to generate a ***Soft Reset*** message to request a Soft Reset.	
2	Protocol Layer resets ***MessageIDCounter***, stored copy of ***MessageID*** and ***RetryCounter***. Protocol Layer creates the message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
3	Physical Layer appends CRC and sends the ***Soft Reset*** message over V_{BUS}.	Physical Layer receives the ***Soft Reset*** message and compares the CRC it calculated with the one sent to verify the message.
4		Physical Layer removes the CRC and forwards the ***Soft Reset*** message to the Protocol Layer.
5		Protocol Layer does not check the ***MessageID*** in the incoming message and resets ***MessageIDCounter***, stored copy of ***MessageID*** and ***RetryCounter***. The Protocol Layer forwards the received ***Soft Reset*** message information to the Policy Engine that consumes it.
6		Protocol Layer generates a ***GoodCRC*** message and passes it Physical Layer.
7	Physical Layer receives the ***GoodCRC*** and checks the CRC to verify the message.	Physical Layer appends CRC and sends the ***GoodCRC*** message.
8	Physical Layer removes the CRC and forwards the ***GoodCRC*** message to the Protocol Layer.	
9	Protocol Layer does not verify or increment the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Soft Reset*** message was successfully sent. Policy Engine starts ***SenderResponseTimer***.	
10		Policy Engine tells the Protocol Layer to form an ***Accept*** message.
11		Protocol Layer creates the message and passes to Physical Layer. Starts ***CRCReceiveTimer***.
12	Physical Layer receives the message and compares the CRC it calculated with the one sent to verify the message.	Physical Layer appends a CRC and sends the message over V_{BUS}.
13	Protocol Layer does not check or store the ***MessageID*** of the incoming message.	

Step	Reset Initiator	Reset Responder
14	The Protocol Layer forwards the received ***Accept*** message information to the Policy Engine that consumes it.	
15	Protocol Layer generates a ***GoodCRC*** message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the ***GoodCRC*** message over V_{BUS}.	Physical Layer receives ***GoodCRC*** message and compares the CRC it calculated with the one sent to verify the message.
17		Physical Layer removes the CRC and forwards the ***GoodCRC*** message to the Protocol Layer.
18		Protocol Layer does not verify or increment the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Accept*** message was successfully sent.
	The reset is complete and protocol communication can restart.	

8.3.2.8 Hard Reset

The following sections describe the steps required for a USB Power Delivery Hard Reset. The Hard Reset returns the operation of the USB Power Delivery to default role and operating voltage/current. During the Hard ResetUSB Power Delivery communication between the Port Partners shall be disabled.

Note: Hard Resetin this case is applied to the USB Power Delivery capability of an individual port on which the Hard Reset is requested. A side effect of the is that it might reset other functions on the port such as USB.

This is an example of a Hard Reset operation when initiated by a Source. Figure 8-9 shows the messages as they flow across the bus and within the Devices to accomplish the Hard Reset.

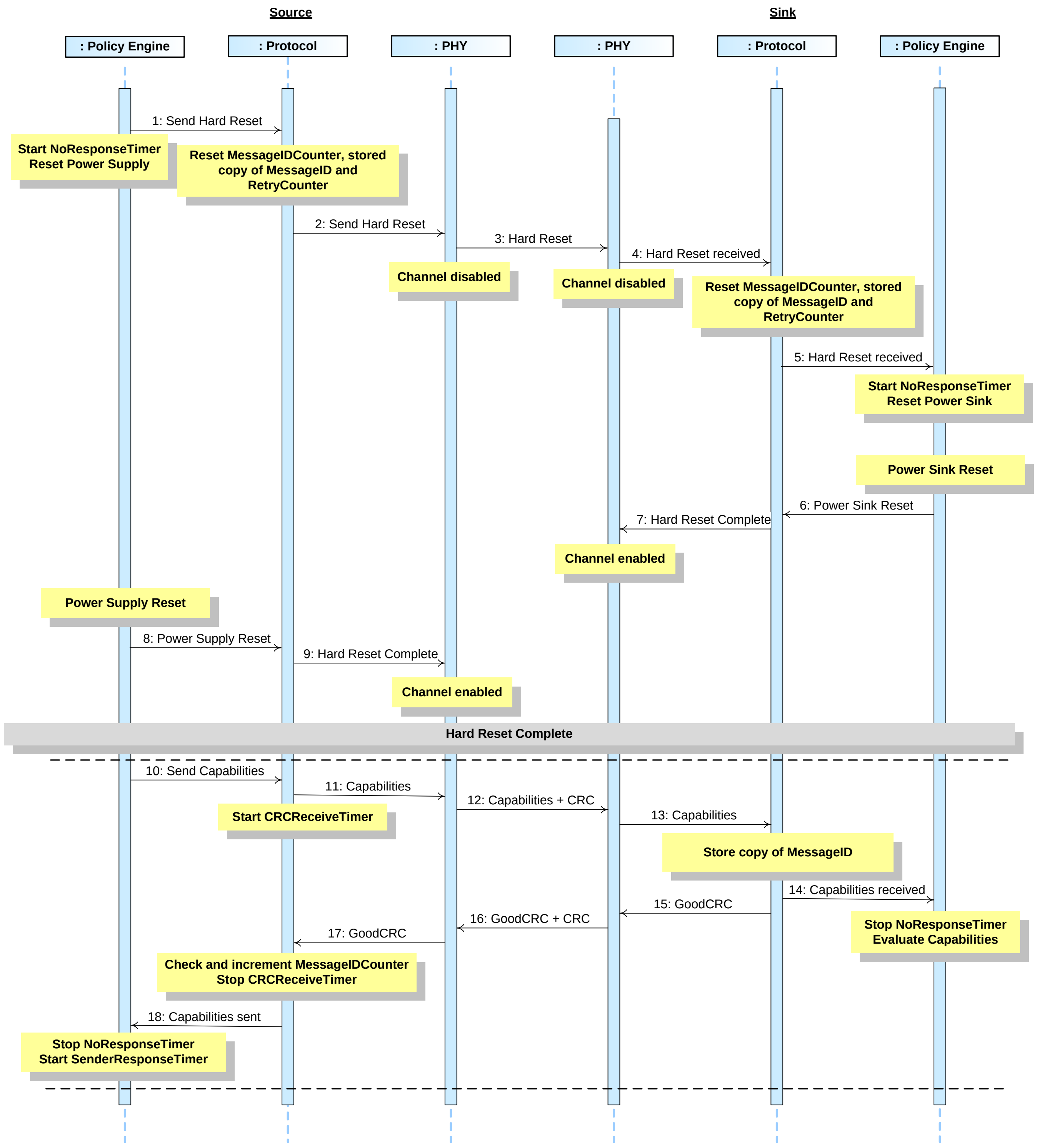


Figure 8-9 Source initiated Hard Reset

Table 8-9 Steps for Source initiated Hard Reset

Step	Source	Sink
1	The Policy Engine directs the Protocol Layer to generate ***Hard Reset*** signaling. The Policy Engine starts the ***NoResponseTimer*** and directs the Power Supply to reset to default operation.	
2	Protocol Layer resets ***MessageIDCounter***, stored copy of ***MessageID*** and ***RetryCounter***. Protocol Layer requests the Physical Layer send ***Hard Reset*** signaling.	
3	Physical Layer sends ***Hard Reset*** signaling over V_{BUS} and then disables the PHY communications channel for transmission and reception.	Physical Layer receives the ***Hard Reset*** signaling and disables the PHY communications channel for transmission and reception.
4		Physical Layer informs the Protocol Layer of the Hard Reset. Protocol Layer resets ***MessageIDCounter***, stored copy of ***MessageID*** and ***RetryCounter***.
5		The Protocol Layer informs the Policy Engine of the Hard Reset. The Policy Engine starts the ***NoResponseTimer*** and directs the Sink to reset to default operation.
6		The Sink returns to default operation. The Policy Engine informs the Protocol Layer that the Sink has been reset.
7		The Protocol Layer informs the PHY layer that the Hard Reset is complete. The PHY layer enables the PHY communications channel for transmission and reception.
8	The Power Supply is reset to default operation. The Policy Engine informs the Protocol Layer that the Power Supply has been reset.	
9	The Protocol Layer informs the PHY layer that the Hard Reset is complete. The PHY layer enables the PHY communications channel for transmission and reception.	
	The reset is complete and protocol communication can restart.	
10	Policy Engine directs the Protocol Layer to send a ***Capabilities*** message that represents the power supply's present capabilities.	
11	Protocol Layer creates the message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
12	Physical Layer appends CRC and sends the ***Capabilities*** message.	Physical Layer receives the ***Capabilities*** message and checks the CRC to verify the message.

Step	Source	Sink
13		Physical Layer removes the CRC and forwards the ***Capabilities*** message to the Protocol Layer.
14		Protocol Layer stores the ***MessageID*** of the incoming message. The Protocol Layer forwards the received ***Capabilities*** message information to the Policy Engine that consumes it. The Policy Engine stops the ***NoResponseTimer***.
15		Protocol Layer generates a ***GoodCRC*** message and passes it Physical Layer.
16	Physical Layer receives the ***GoodCRC*** message and checks the CRC to verify the message.	Physical Layer appends CRC and sends the ***GoodCRC*** message.
17	Physical Layer removes the CRC and forwards the ***GoodCRC*** message to the Protocol Layer.	
18	Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Capabilities*** message was successfully sent. Policy Engine stops the ***NoResponseTimer*** and starts the ***SenderResponseTimer***.	
	USB Power Delivery communication is re-established.	

8.3.2.8.1 Sink Initiated Hard Reset

This is an example of a Hard Reset operation when initiated by a Sink. Figure 8-10 shows the messages as they flow across the bus and within the Devices to accomplish the Hard Reset.

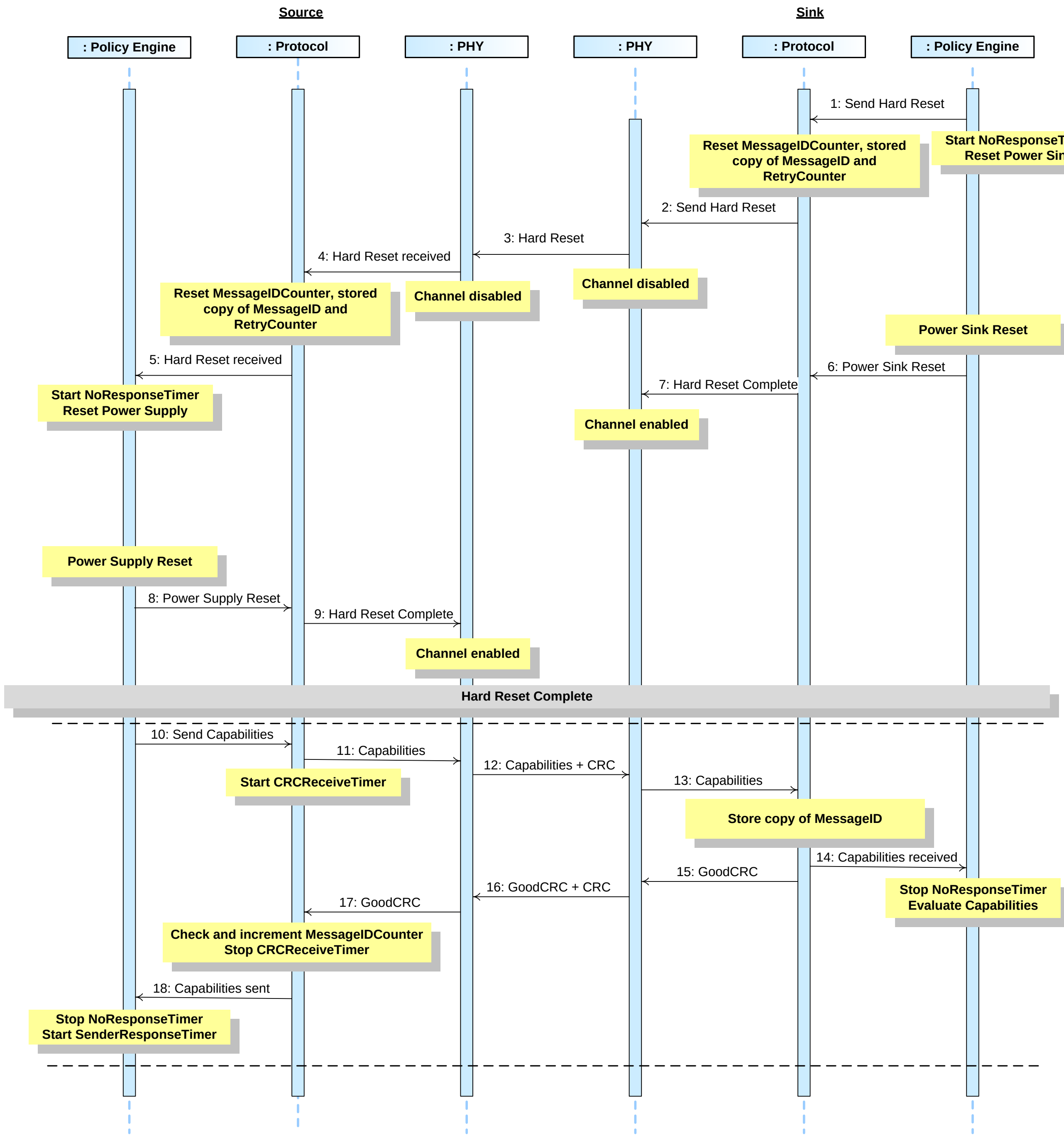


Figure 8-10 Sink Initiated Hard Reset

Table 8-10 Steps for Sink initiated Hard Reset

Step	Source	Sink
1		The Policy Engine directs the Protocol Layer to generate ***Hard Reset*** signaling. The Policy Engine starts the ***NoResponseTimer*** and directs the Sink to reset to default operation.
2		Protocol Layer resets ***MessageIDCounter***, stored copy of ***MessageID*** and ***RetryCounter***. Protocol Layer requests the Physical Layer send ***Hard Reset*** signaling.
3	Physical Layer receives the ***Hard Reset*** signaling and disables the PHY communications channel for transmission and reception.	Physical Layer sends the ***Hard Reset*** signaling over V_{BUS} and then disables the PHY communications channel for transmission and reception.
4	Physical Layer informs the Protocol Layer of the Hard Reset. Protocol Layer resets ***MessageIDCounter***, stored copy of ***MessageID*** and ***RetryCounter***.	
5	The Protocol Layer Informs the Policy Engine of the Hard Reset. The Policy Engine starts the ***NoResponseTimer*** and requests the Power Supply to reset to default operation.	
6		The Sink returns to default operation. The Policy Engine informs the Protocol Layer that the Sink has been reset.
7		The Protocol Layer informs the PHY layer that the Hard Reset is complete. The PHY layer enables the PHY communications channel for transmission and reception.
8	The Power Supply is reset to default operation. The Policy Engine informs the Protocol Layer that the Power Supply has been reset.	
9	The Protocol Layer informs the PHY layer that the Hard Reset is complete. The PHY layer enables the PHY communications channel for transmission and reception.	
	The reset is complete and protocol communication can restart.	
10	Policy Engine directs the Protocol Layer to send a ***Capabilities*** message that represents the power supply's present capabilities.	
11	Protocol Layer creates the message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
12	Physical Layer appends CRC and sends the ***Capabilities*** message.	Physical Layer receives the ***Capabilities*** message and checks the CRC to verify the

Step	Source	Sink
		message.
13		Physical Layer removes the CRC and forwards the ***Capabilities*** message to the Protocol Layer.
14		Protocol Layer stores the ***MessageID*** of the incoming message. The Protocol Layer forwards the received ***Capabilities*** message information to the Policy Engine that consumes it. The Policy Engine stops the ***NoResponseTimer***.
15		Protocol Layer generates a ***GoodCRC*** message and passes it Physical Layer.
16	Physical Layer receives the ***GoodCRC*** message and checks the CRC to verify the message.	Physical Layer appends CRC and sends the ***GoodCRC*** message.
17	Physical Layer removes the CRC and forwards the ***GoodCRC*** message to the Protocol Layer.	
18	Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Capabilities*** message was successfully sent. Policy Engine stops the ***NoResponseTimer*** and starts the ***SenderResponseTimer***.	
	USB Power Delivery communication is re-established.	

8.3.2.8.2 Source Initiated Hard Reset – Sink Long Reset

This is an example of a Hard Reset operation when initiated by a Source. In this example the Sink is slow responding to the reset causing the Source to send multiple ***Capabilities*** messages before it receives a ***GoodCRC*** message response. Figure 8-11 shows the messages as they flow across the bus and within the Devices to accomplish the Hard Reset.

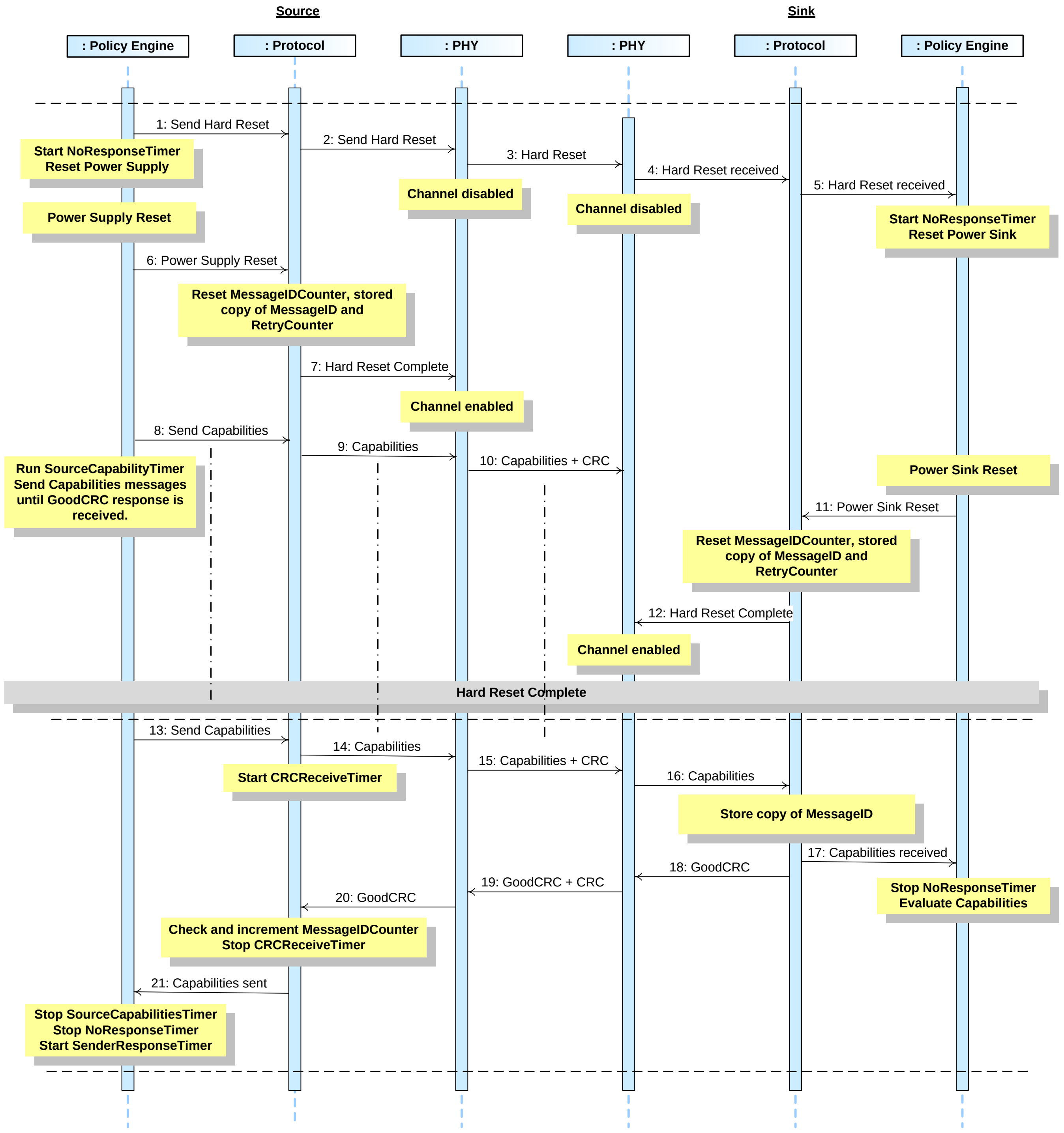


Figure 8-11 Source initiated reset - Sink long reset

Table 8-11 Steps for Source initiated Hard Reset – Sink long reset

Step	Source	Sink
1	The Policy Engine directs the Protocol Layer to generate ***Hard Reset*** signaling. The Policy Engine starts the ***NoResponseTimer*** and directs the Power Supply to reset to default operation.	
2	Protocol Layer resets ***MessageIDCounter***, stored copy of ***MessageID*** and ***RetryCounter***. Protocol Layer requests the Physical Layer send ***Hard Reset*** signaling.	
3	Physical Layer sends the ***Hard Reset*** signaling over V_{BUS} and then disables the PHY communications channel for transmission and reception.	Physical Layer receives the ***Hard Reset*** signaling and disables the PHY communications channel for transmission and reception.
4		Physical Layer informs the Protocol Layer of the Hard Reset. Protocol Layer resets ***MessageIDCounter***, stored copy of ***MessageID*** and ***RetryCounter***.
5		The Protocol Layer Informs the Policy Engine of the Hard Reset. The Policy Engine starts the ***NoResponseTimer*** and requests the Sink to reset to default operation.
6	The Power Supply is reset to default operation. The Policy Engine informs the Protocol Layer that the Power Supply has been reset.	
7	The Protocol Layer informs the PHY layer that the Hard Reset is complete. The PHY layer enables the PHY communications channel for transmission and reception.	
	The reset is complete and protocol communication can restart.	
8	Policy Engine directs the Protocol Layer to send a ***Capabilities*** message that represents the power supply's present capabilities. Starts the ***SourceCapabilityTimer***. The ***SourceCapabilityTimer*** times out one or more times until a ***GoodCRC*** message response is received.	
9	Protocol Layer creates the message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
10	Physical Layer appends CRC and sends the ***Capabilities*** message.	
11		The Sink returns to default operation. The Policy Engine informs the Protocol Layer that

Step	Source	Sink
		the Sink has been reset.
12		The Protocol Layer informs the PHY layer that the Hard Reset is complete. The PHY layer enables the PHY communications channel for transmission and reception.
13	Policy Engine directs the Protocol Layer to send a ***Capabilities*** message that represents the power supply's present capabilities. Starts the ***SourceCapabilityTimer***.	
14	Protocol Layer creates the message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
15	Physical Layer appends CRC and sends the ***Capabilities*** message.	Physical Layer receives the ***Capabilities*** message and checks the CRC to verify the message.
16		Physical Layer removes the CRC and forwards the ***Capabilities*** message to the Protocol Layer.
17		Protocol Layer stores the ***MessageID*** of the incoming message. The Protocol Layer forwards the received ***Capabilities*** message information to the Policy Engine that consumes it. The Policy Engine stops the ***NoResponseTimer***.
18		Protocol Layer generates a ***GoodCRC*** message and passes it Physical Layer.
19	Physical Layer receives the ***GoodCRC*** message and checks the CRC to verify the message.	Physical Layer appends CRC and sends the ***GoodCRC*** message.
20	Physical Layer removes the CRC and forwards the ***GoodCRC*** message to the Protocol Layer.	
21	Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Capabilities*** message was successfully sent. Policy Engine stops the ***SourceCapabilityTimer***, stops the ***NoResponseTimer*** and starts the ***SenderResponseTimer***.	
	USB Power Delivery communication is re-established.	

8.3.2.8.3 Source Initiated Hard Reset (role swapped)

This is an example of a Hard Reset operation when initiated by a Dual-Role device which is currently swapped and acting as a Source. In this example both Dual-Role devices return to their original roles after the Hard Reset. Figure 8-12 shows the messages as they flow across the bus and within the Devices to accomplish the Hard Reset.

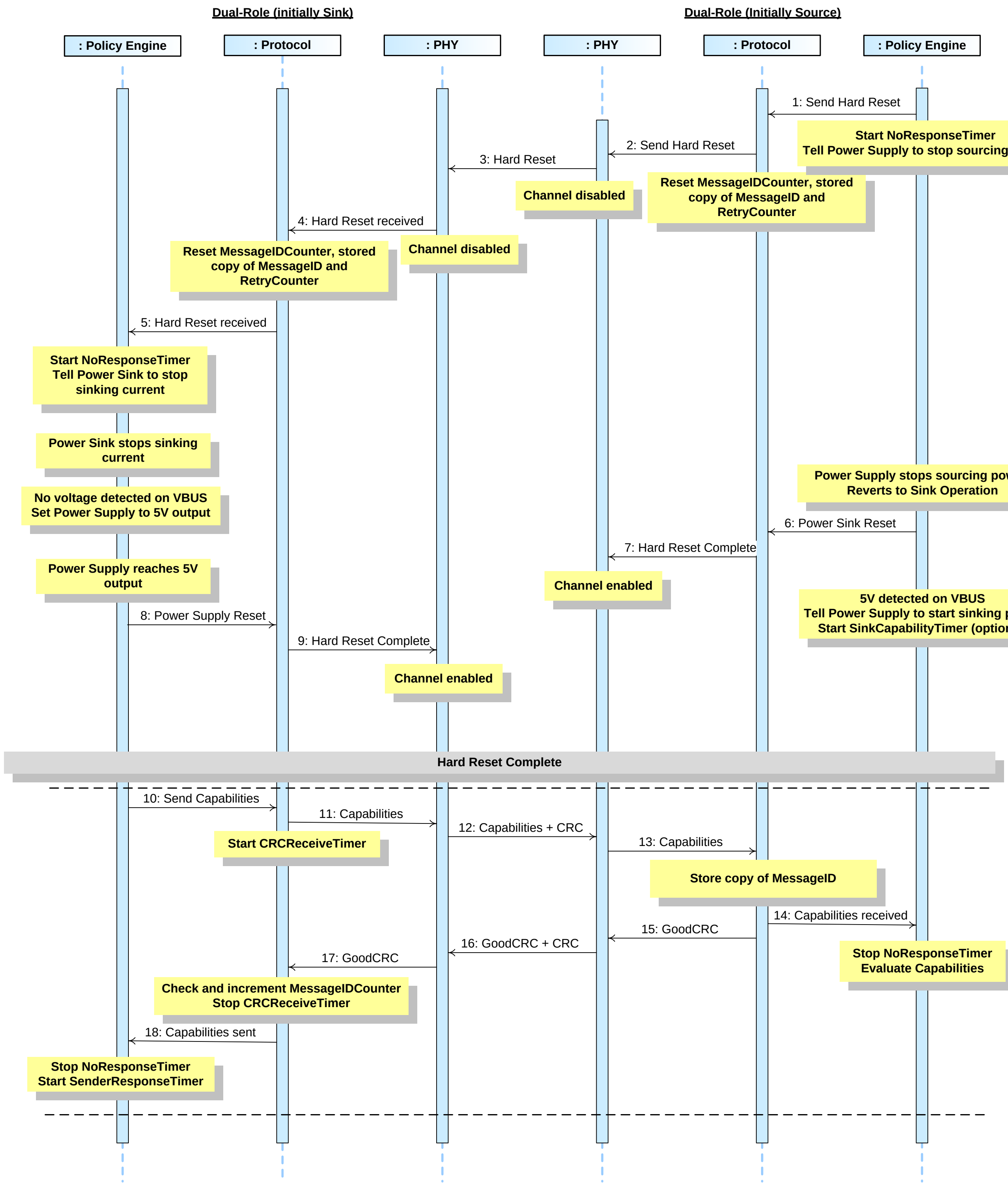


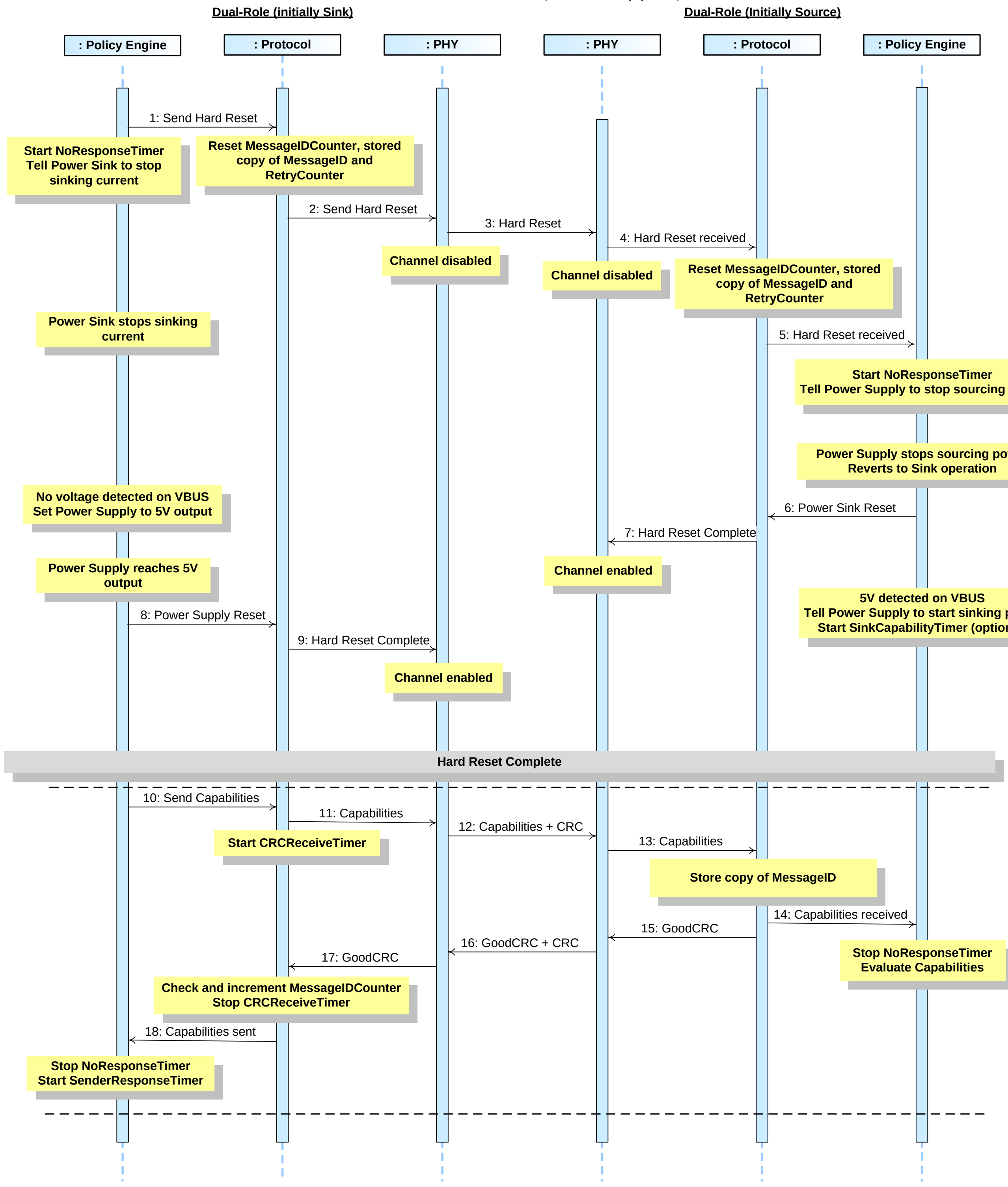
Figure 8-12 Source initiated Hard Reset (role swapped)

Table 8-12 Steps for Source initiated Hard Reset (role swapped)

Step	Dual-role (InitiallySink)	Dual-role (InitiallySource)
1		The Policy Engine directs the Protocol Layer to generate ***Hard Reset*** signaling. The Policy Engine starts the ***NoResponseTimer*** and directs the Power Supply to stop sourcing power.
2		Protocol Layer resets ***MessageIDCounter***, stored copy of ***MessageID*** and ***RetryCounter***. Protocol Layer requests the Physical Layer send ***Hard Reset*** signaling.
3	Physical Layer receives the ***Hard Reset*** signaling and disables the PHY communications channel for transmission and reception.	Physical Layer sends the ***Hard Reset*** signaling over V_{BUS} and then disables the PHY communications channel for transmission and reception.
4	Physical Layer informs the Protocol Layer of the Hard Reset. Protocol Layer resets ***MessageIDCounter***, stored copy of ***MessageID*** and ***RetryCounter***.	
5	The Protocol Layer Informs the Policy Engine of the Hard Reset. The Policy Engine starts the ***NoResponseTimer*** and directs the Sink to stop sinking power.	
6		The Power Supply stops sourcing power on V_{BUS} and reverts to Sink operation. The Policy Engine informs the Protocol Layer that the Sink has been reset.
7		The Protocol Layer informs the PHY layer that the Hard Reset is complete. The PHY layer enables the PHY communications channel for transmission and reception. When ***vSafe5V*** is detected on V_{BUS} the Policy Engine directs the Sink to start Sinking power and optionally starts the ***SinkCapabilityTimer***.
8	The Sink stops sinking power. When there is no voltage detected on V_{BUS} the Policy Engine requests the Source to go to ***vSafe5V*** output. When the Source reaches ***vSafe5V*** the Policy Engine informs the Protocol Layer that the Power Supply has been reset.	
9	The Protocol Layer informs the PHY layer that the Hard Reset is complete. The PHY layer enables the PHY communications channel for transmission and reception.	
	The reset is complete and protocol communication can restart.	

Step	Dual-role (InitiallySink)	Dual-role (InitiallySource)
10	Policy Engine directs the Protocol Layer to send a ***Capabilities*** message that represents the power supply's present capabilities.	
11	Protocol Layer creates the message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
12	Physical Layer appends CRC and sends the ***Capabilities*** message.	Physical Layer receives the ***Capabilities*** message and checks the CRC to verify the message.
13		Physical Layer removes the CRC and forwards the ***Capabilities*** message to the Protocol Layer.
14		Protocol Layer stores the ***MessageID*** of the incoming message. The Protocol Layer forwards the received ***Capabilities*** message information to the Policy Engine that consumes it. The Policy Engine stops the ***NoResponseTimer***.
15		Protocol Layer generates a ***GoodCRC*** message and passes it Physical Layer.
16	Physical Layer receives the ***GoodCRC*** message and checks the CRC to verify the message.	Physical Layer appends CRC and sends the ***GoodCRC*** message.
17	Physical Layer removes the CRC and forwards the ***GoodCRC*** message to the Protocol Layer.	
18	Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Capabilities*** message was successfully sent. Policy Engine stops the ***NoResponseTimer*** and starts the ***SenderResponseTimer***.	
	USB Power Delivery communication is re-established.	

8.3.2.8.4 Sink Initiated Hard Reset (role swapped)



Dual-Role (initially Sink)
Dual-Role (Initially Source)
: Policy Engine
: Protocol
: PHY
: PHY
: Protocol
: Policy Engine
1: Send Hard Reset
Start NoResponseTimer Tell Power Sink to stop sinking current
Reset MessageIDCounter, stored copy of MessageID and RetryCounter
2: Send Hard Reset
3: Hard Reset
4: Hard Reset received
Channel disabled
Channel disabled
Reset MessageIDCounter, stored copy of MessageID and RetryCounter
Power Sink stops sinking current
5: Hard Reset received
Start NoResponseTimer Tell Power Supply to stop sourcing power
Power Supply stops sourcing power Reverts to Sink operation
No voltage detected on VBUS Set Power Supply to 5V output
6: Power Sink Reset
7: Hard Reset Complete
Power Supply reaches 5V output
Channel enabled
5V detected on VBUS Tell Power Supply to start sinking power Start SinkCapabilityTimer (optional)
8: Power Supply Reset
9: Hard Reset Complete
Channel enabled
Hard Reset Complete
10: Send Capabilities
11: Capabilities
12: Capabilities + CRC
13: Capabilities
Start CRCReceiveTimer
Store copy of MessageID
14: Capabilities received
15: GoodCRC
16: GoodCRC + CRC
17: GoodCRC
Stop NoResponseTimer Evaluate Capabilities
Check and increment MessageIDCounter Stop CRCReceiveTimer
18: Capabilities sent
Stop NoResponseTimer Start SenderResponseTimer

Figure 8-13 Sink Initiated Hard Reset (role swapped)

Table 8-13 Steps for Sink initiated Hard Reset (role swapped)

Step	Dual-role (InitiallySink)	Dual-role (InitiallySource)
1	The Policy Engine directs the Protocol Layer to generate ***Hard Reset*** signaling. The Policy Engine starts the ***NoResponseTimer*** and directs the Sink to stop sinking power.	
2	Protocol Layer resets ***MessageIDCounter***, stored copy of ***MessageID*** and ***RetryCounter***. Protocol Layer requests the Physical Layer send ***Hard Reset*** signaling.	
3	Physical Layer sends the ***Hard Reset*** signaling over V_{BUS} and then disables the PHY communications channel for transmission and reception.	Physical Layer receives the ***Hard Reset*** signaling and disables the PHY communications channel for transmission and reception.
4		Physical Layer informs the Protocol Layer of the Hard Reset. Protocol Layer resets ***MessageIDCounter***, stored copy of ***MessageID*** and ***RetryCounter***.
5		The Protocol Layer Informs the Policy Engine of the Hard Reset. The Policy Engine starts the ***NoResponseTimer*** and directs the Source to stop sourcing power.
6		The Power Supply stops sourcing power on V_{BUS} and reverts to Sink operation. The Policy Engine informs the Protocol Layer that the Sink has been reset.
7		When ***vSafe5V*** is detected on V_{BUS} the Policy Engine directs the Sink to start Sinking power and optionally starts the ***SinkCapabilityTimer***. The Protocol Layer informs the PHY layer that the Hard Reset is complete. The PHY layer enables the PHY communications channel for transmission and reception.
8	The Sink stops sinking power. When there is no voltage detected on V_{BUS} the Policy Engine requests the Source to go to ***vSafe5V*** output. When the Source reaches ***vSafe5V*** the Policy Engine informs the Protocol Layer that the Power Supply has been reset.	
9	The Protocol Layer informs the PHY layer that the Hard Reset is complete. The PHY layer enables the PHY communications channel for transmission and reception.	
	The reset is complete and protocol communication can restart.	

Step	Dual-role (InitiallySink)	Dual-role (InitiallySource)
10	Policy Engine directs the Protocol Layer to send a ***Capabilities*** message that represents the power supply's present capabilities.	
11	Protocol Layer creates the message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
12	Physical Layer appends CRC and sends the ***Capabilities*** message.	Physical Layer receives the ***Capabilities*** message and checks the CRC to verify the message.
13		Physical Layer removes the CRC and forwards the ***Capabilities*** message to the Protocol Layer.
14		Protocol Layer stores the ***MessageID*** of the incoming message. The Protocol Layer forwards the received ***Capabilities*** message information to the Policy Engine that consumes it. The Policy Engine stops the ***NoResponseTimer***.
15		Protocol Layer generates a ***GoodCRC*** message and passes it Physical Layer.
16	Physical Layer receives the ***GoodCRC*** message and checks the CRC to verify the message.	Physical Layer appends CRC and sends the ***GoodCRC*** message.
17	Physical Layer removes the CRC and forwards the ***GoodCRC*** message to the Protocol Layer.	
18	Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***Capabilities*** message was successfully sent. Policy Engine stops the ***NoResponseTimer*** and starts the ***SenderResponseTimer***.	
	USB Power Delivery communication is re-established.	

8.3.2.9 Built in Self-Test (BIST)

8.3.2.9.1 BIST Receiver Mode

This is an example of a BIST Receiver Mode test between a Tester and a Unit Under Test (UUT).

Figure 8-14shows the messages as they flow across the bus and within the Devices. This test verifies that the UUT can receive data with a performance according this specification.

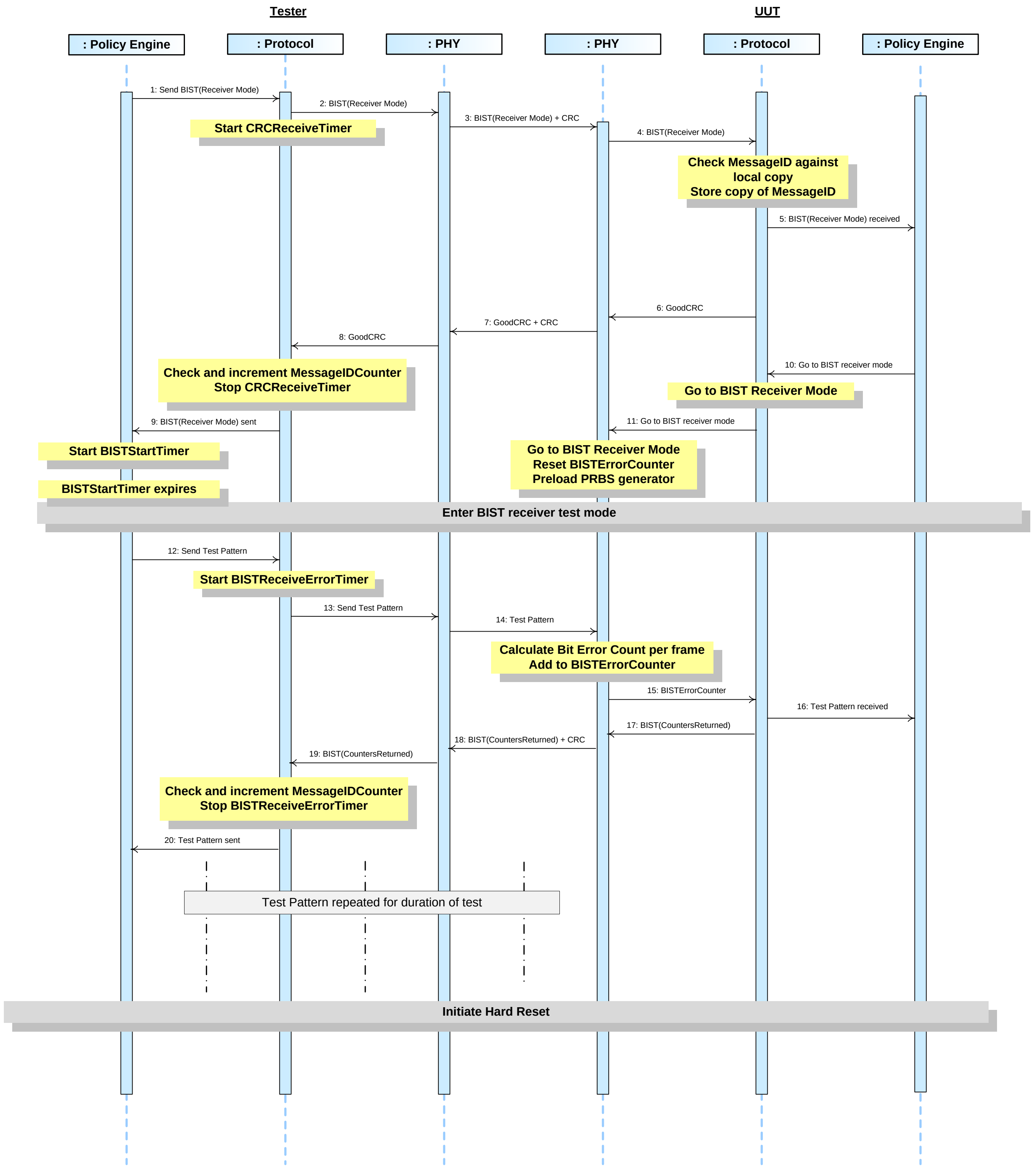

Tester
UUT
: Policy Engine
: Protocol
: PHY
: PHY
: Protocol
: Policy Engine
1: Send BIST(Receiver Mode)
2: BIST(Receiver Mode)
Start CRCReceiveTimer
3: BIST(Receiver Mode) + CRC
4: BIST(Receiver Mode)
Check MessageID against local copy
Store copy of MessageID
5: BIST(Receiver Mode) received
6: GoodCRC
7: GoodCRC + CRC
8: GoodCRC
Check and increment MessageIDCounter
Stop CRCReceiveTimer
10: Go to BIST receiver mode
Go to BIST Receiver Mode
9: BIST(Receiver Mode) sent
11: Go to BIST receiver mode
Start BISTStartTimer
Go to BIST Receiver Mode
Reset BISTErrorCounter
Preload PRBS generator
BISTStartTimer expires
Enter BIST receiver test mode
12: Send Test Pattern
Start BISTReceiveErrorTimer
13: Send Test Pattern
14: Test Pattern
Calculate Bit Error Count per frame
Add to BISTErrorCounter
15: BISTErrorCounter
16: Test Pattern received
17: BIST(CountersReturned)
18: BIST(CountersReturned) + CRC
19: BIST(CountersReturned)
Check and increment MessageIDCounter
Stop BISTReceiveErrorTimer
20: Test Pattern sent
Test Pattern repeated for duration of test
Initiate Hard Reset

Figure 8-14 BIST Receiver Mode Test

Table 8-14 Steps for BIST Receiver Mode Test

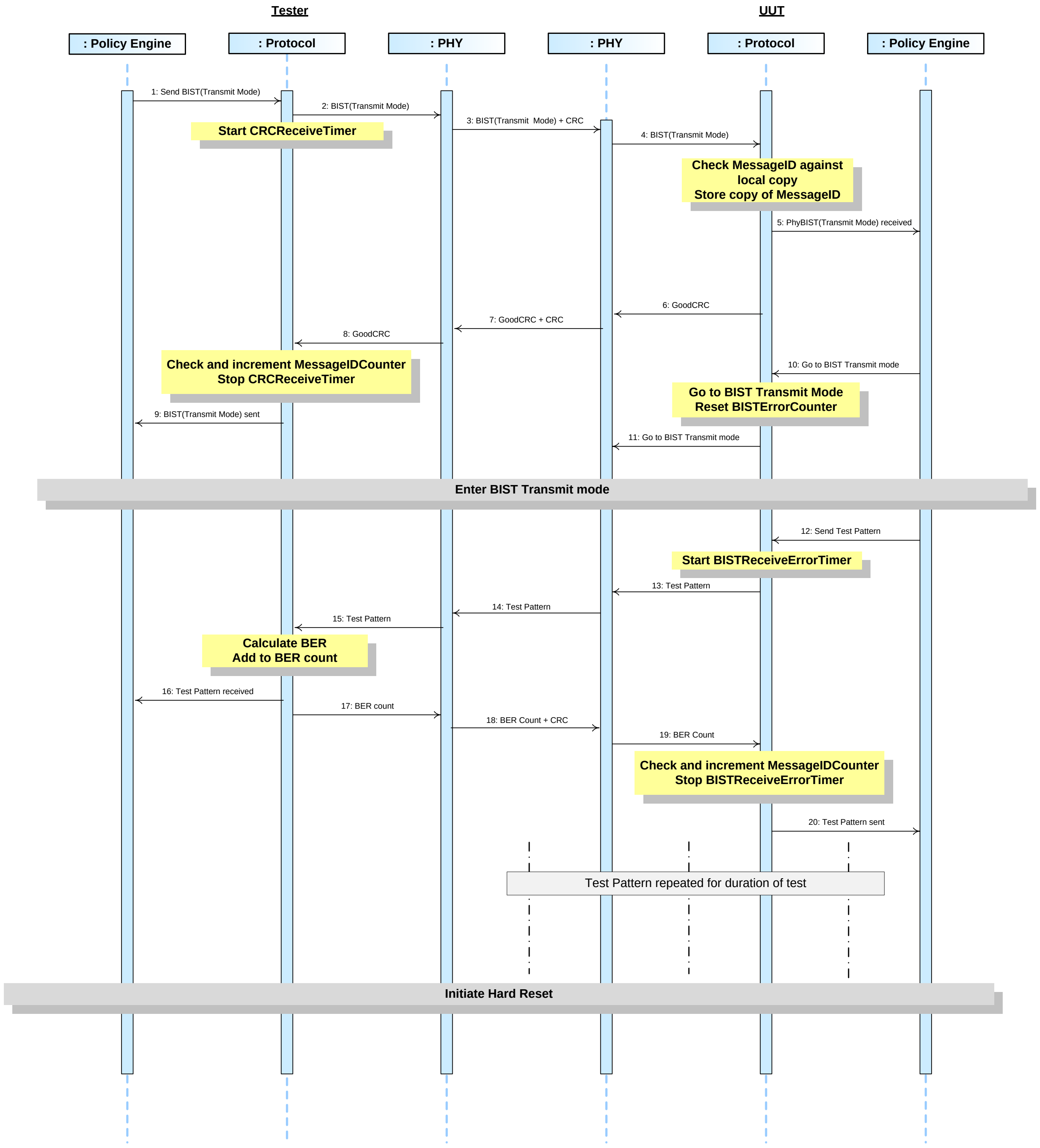
Step	Tester	UUT
1	The Policy Engine directs the Protocol Layer to generate a ***BIST*** message with Data Object ***BIST Receiver Mode*** to put the UUT into BIST receiver mode.	
2	Protocol Layer creates the message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
3	Physical Layer appends CRC and sends the ***BIST*** message ove V_{BUS}.	Physical Layer receives the ***BIST*** message and compares the CRC it calculated with the one sent to verify the message.
4		Physical Layer removes the CRC and forwards the ***BIST*** message to the Protocol Layer.
5		Protocol Layer checks the ***MessageID*** in the incoming message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received ***BIST*** message information to the Policy Engine that consumes it.
6		Protocol Layer generates a ***GoodCRC*** message and passes it Physical Layer.
7	Physical Layer receives the ***GoodCRC*** and checks the CRC to verify the message.	Physical Layer appends CRC and sends the ***GoodCRC*** message.
8	Physical Layer removes the CRC and forwards the ***GoodCRC*** message to the Protocol Layer.	
9	Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***BIST*** message was successfully sent. The Policy Engine initializes and runs the ***BISTStartTimer***.	
10		Policy Engine tells Protocol Layer to go into Receiver Test Mode. The Protocol Layer goes to BIST receiver mode.
11		Protocol Layer tells Physical Layer to go into Receiver Test Mode. The Physical Layer goes to BIST receiver mode, resets the ***BISTErrorCounter*** and preloads the PRBS generator.
	UUT enters BIST Receiver Test Mode	
12	After the ***BISTStartTimer*** has expired the Policy Engine directs the Protocol Layer to generate a packet containing the next Test	

Step	Tester	UUT
	Pattern.	
13	Protocol Layer creates the message and passes to Physical Layer. Starts ***BISTReceiveErrorTimer***.	
14	Physical Layer does not append a CRC and sends the Test Pattern packet over V_{BUS}.	Physical Layer receives the Test Pattern packet. The Physical Layer calculates the Bit Error Count for the frame and adds this to the ***BISTErrorCounter***.
15		Physical Layer forwards the ***BISTErrorCounter*** value to the Protocol Layer.
16		The Protocol Layer informs the Policy Engine that a Test Frame has been received.
17		Protocol Layer generates a ***BIST*** message with Data Object of ***Returned BIST Counters*** and passes it Physical Layer.
18	Physical Layer receives ***BIST*** message and checks the CRC to verify the message.	Physical Layer appends CRC and sends the BER Count message.
19	Physical Layer removes the CRC and forwards the ***BIST*** message to the Protocol Layer.	
20	Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***BISTReceiveErrorTimer***. Protocol Layer informs the Policy Engine that the Test Pattern packet was successfully sent.	
	Steps 10 to 18 are repeated until ***Hard Reset*** signaling is generated at which point the UUT returns to normal operation.	

8.3.2.9.2 BIST Transmit mode

This is an example of a BIST Transmitter Mode test between a Tester and a UUT.

Figure 8-15 shows the messages as they flow across the bus and within the Devices. This test verifies that the UUT transmitter can transmit with sufficient quality (see Section 6.4.3.2).



Tester
UUT
: Policy Engine
: Protocol
: PHY
: PHY
: Protocol
: Policy Engine
1: Send BIST(Transmit Mode)
2: BIST(Transmit Mode)
Start CRCReceiveTimer
3: BIST(Transmit Mode) + CRC
4: BIST(Transmit Mode)
Check MessageID against local copy Store copy of MessageID
5: PhyBIST(Transmit Mode) received
6: GoodCRC
7: GoodCRC + CRC
8: GoodCRC
Check and increment MessageIDCounter Stop CRCReceiveTimer
10: Go to BIST Transmit mode
Go to BIST Transmit Mode Reset BISTErrorCounter
9: BIST(Transmit Mode) sent
11: Go to BIST Transmit mode
Enter BIST Transmit mode
12: Send Test Pattern
Start BISTReceiveErrorTimer
13: Test Pattern
14: Test Pattern
15: Test Pattern
Calculate BER Add to BER count
16: Test Pattern received
17: BER count
18: BER Count + CRC
19: BER Count
Check and increment MessageIDCounter Stop BISTReceiveErrorTimer
20: Test Pattern sent
Test Pattern repeated for duration of test
Initiate Hard Reset

Figure 8-15 BIST Transmit Mode Test

Table 8-15 Steps for BIST Transmitter Mode Test

Step	Tester	UUT
1	The Policy Engine directs the Protocol Layer to generate a ***BIST*** message, with Data Object ***BIST Transmit Mode***, to put the UUT into BIST Transmit Mode.	
2	Protocol Layer creates the message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
3	Physical Layer appends CRC and sends the ***BIST*** message over V_{BUS}.	Physical Layer receives the ***BIST*** message and compares the CRC it calculated with the one sent to verify the message.
4		Physical Layer removes the CRC and forwards the ***BIST*** message to the Protocol Layer.
5		Protocol Layer checks the ***MessageID*** in the incoming message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received ***BIST*** message information to the Policy Engine that consumes it.
6		Protocol Layer generates a ***GoodCRC*** message and passes it Physical Layer.
7	Physical Layer receives the ***GoodCRC*** and checks the CRC to verify the message.	Physical Layer appends CRC and sends the ***GoodCRC*** message.
8	Physical Layer removes the CRC and forwards the ***GoodCRC*** message to the Protocol Layer.	
9	Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***BIST*** message was successfully sent.	
10		Policy Engine tells Protocol Layer to go into Transmit Test Mode. The Policy Engine goes to BIST Transmit mode and resets the ***BISTErrorCounter***.
11		Protocol Layer tells Physical Layer to go into Transmit Test Mode.
	UUT enters BIST Transmitter Test Mode	
12		The Policy Engine directs the Protocol Layer to generate a packet containing the next Test Pattern.
13		Protocol Layer creates the message and passes to Physical Layer. Starts ***BISTReceiveErrorTimer***.
14	Physical Layer receives the Test Pattern packet.	Physical Layer does not append a CRC and sends the Test Pattern packet over V_{BUS}.

Step	Tester	UUT
15	Physical Layer forwards the Test Pattern packet to the Protocol Layer.	
16	The Protocol Layer forwards the received Test Pattern packet information to the Policy Engine that consumes it.	
17	Protocol Layer generates a BER Count message and passes it Physical Layer.	
18	Physical Layer appends CRC and sends the BER Count message.	Physical Layer receives the BER Count message and checks the CRC to verify the message.
19		Physical Layer removes the CRC and forwards the BER Count message to the Protocol Layer.
20		Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***BISTReceiveErrorTimer***. Protocol Layer informs the Policy Engine that the Test Pattern packet was successfully sent.
	Steps 10 to 18 are repeated until ***Hard Reset*** signaling is generated at which point the UUT returns to normal operation.	

8.3.2.9.3 BIST Test Patterns

In addition to the BIST transmit and receive test frames there are various test patterns which the UUT can be made to send continuously in order to perform measurements on the transmission spectrum, eye pattern etc. (see Section 5.12). The following is an example of a ***BIST Eye Pattern*** test between a Tester and a UUT but the sequence applies equally to other continuous test patterns. When the UUT is connected to the Tester the sequence below is executed.

Figure 8-16 shows the messages as they flow across the bus and within the Devices. This test verifies the eye pattern and the spectrum of the transmitted signal.

1) Connection is established and stable.
2) Tester sends a ***BIST*** message with a ***BIST Eye Pattern*** Data Object.
3) UUT answers with a ***GoodCRC*** message.
4) UUT starts sending the test pattern.
5) Operator does the measurements.
6) Operator restarts the UUT. Note: that the method of operator restart for the UUT is outside the scope of this specification. This is assumed to be some mechanism whereby the operator restores the UUT to normal PD operation.

Refer to Sections 6.4.3.4 through 6.4.3.7.

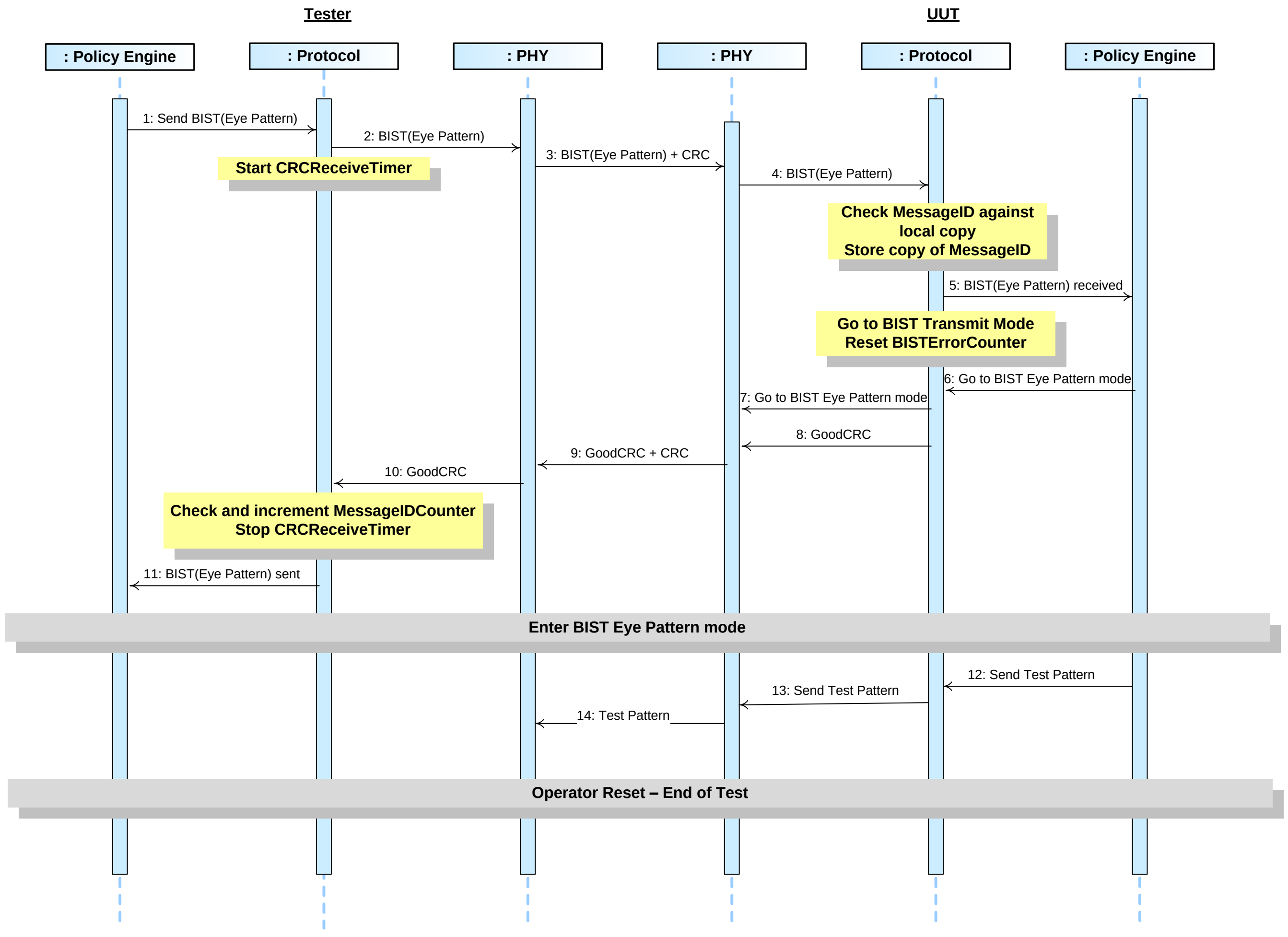


Figure 8-16 BIST Eye Pattern Test

Table 8-16 Steps for BIST Eye Pattern Test

Step	Tester	UUT
1	The Policy Engine directs the Protocol Layer to generate a ***BIST*** message, with Data Object ***BIST Eye Pattern***, to put the UUT into BIST Carrier Purity Mode 1.	
2	Protocol Layer creates the message and passes to Physical Layer. Starts ***CRCReceiveTimer***.	
3	Physical Layer appends CRC and sends the ***BIST*** message ove V_{BUS}.	Physical Layer receives the ***BIST*** message and compares the CRC it calculated with the one sent to verify the message.
4		Physical Layer removes the CRC and forwards the ***BIST*** message to the Protocol Layer.
5		Protocol Layer checks the ***MessageID*** in the incoming message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received ***BIST*** message information to the Policy Engine that consumes it.
6		Policy Engine tells Protocol Layer to go into BIST Eye Pattern Test Mode. The Policy Engine goes to BIST Eye Pattern Test Mode.
7		Protocol Layer tells Physical Layer to go into BIST Eye Pattern Test Mode.
8		Protocol Layer generates a ***GoodCRC*** message and passes it Physical Layer.
9	Physical Layer receives the ***GoodCRC*** and checks the CRC to verify the message.	Physical Layer appends CRC and sends the ***GoodCRC*** message.
10	Physical Layer removes the CRC and forwards the ***GoodCRC*** message to the Protocol Layer.	
11	Protocol Layer verifies and increments the ***MessageIDCounter*** and stops ***CRCReceiveTimer***. Protocol Layer informs the Policy Engine that the ***BIST*** message was successfully sent.	
	UUT enters BIST Eye Pattern Test Mode	
12		The Policy Engine directs the Protocol Layer to start generation of the Test Pattern.
13		Protocol Layer directs the PHY Layer to generate the test pattern.
14	Physical Layer receives the Test Pattern stream.	Physical Layer generates a continuous test pattern stream over V_{BUS}.
	Operator reset of UUT terminates the test	

8.3.3 State Diagrams

8.3.3.1 Introduction to state diagrams used in Chapter 8

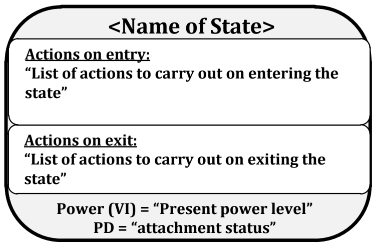

<Name of State>
Actions on entry:
"List of actions to carry out on entering the state"
Actions on exit:
"List of actions to carry out on exiting the state"
Power (VI) = "Present power level"
PD = "attachment status"

Figure 8-17 Outline of States

Figure 8-17 shows an outline of the states defined in the following sections. At the top there is the name of the state. This is followed by "Actions on entry" a list of actions carried out on entering the state. If there are also "Actions on exit" a list of actions carried out on exiting the state then these are listed as well; otherwise this box is omitted from the state. At the bottom the status of PD is listed:

- "Power" which indicates the present output power for a Source Port or input power for a Sink Port.
- "PD" which indicates the present attachment status either "attached", "unattached", or "unknown".

Transitions from one state to another are indicated by arrows with the conditions listed on the arrow. Where there are multiple conditions these are connected using either a logical OR "|" or a logical AND "&".

In some cases there are transitions which can occur from any state to a particular state. These are indicated by an arrow which is unconnected to a state at one end, but with the other end (the point) connected to the final state.

In some state diagrams it is necessary to enter or exit from states in other diagrams (e.g. Source Port or Sink Port state diagrams). Figure 8-18 indicates how such references are made. The reference is indicated with a hatched box. The box contains the name of the state and whether the state is in the Source or Sink Port state diagrams. It has also been necessary to indicate conditional entry to either Source Port or Sink Port state diagrams. This is achieved by the use of a bulleted list indicating the pre-conditions (see example in Figure 8-19).

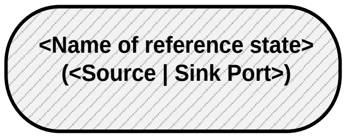


Figure 8-18 References to states

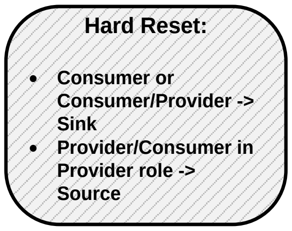

Hard Reset:
Consumer or Consumer/Provider -> Sink
Provider/Consumer in Provider role -> Source

Figure 8-19 Example of state reference with conditions

Timers are included in many of the states. Timers are initialized (set to their starting condition) and run (timer is counting) in the particular state it is referenced. As soon as the state is exited then the timer is no longer active.

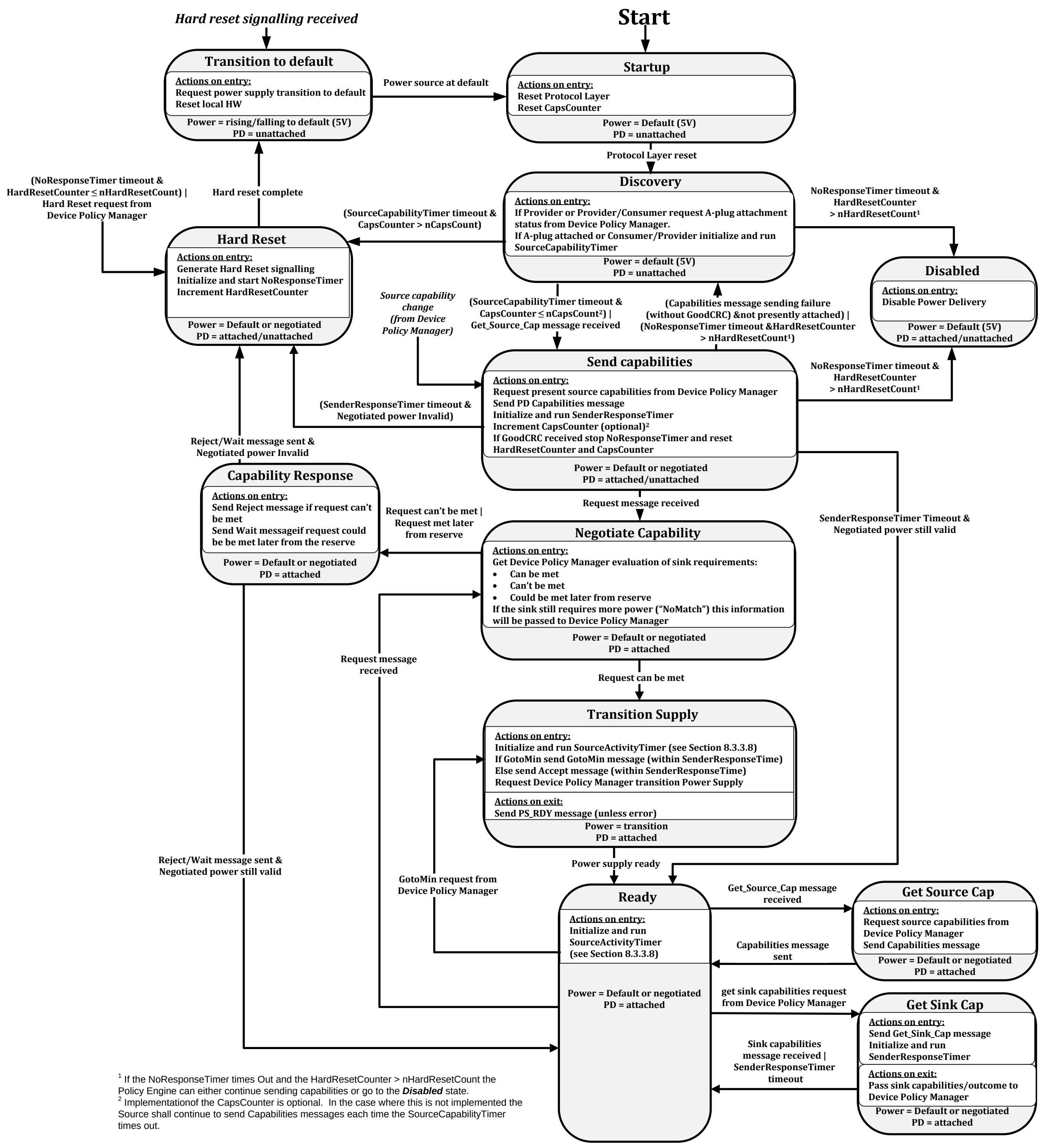
Where the timers continue to run outside of the state (such as the ***NoResponseTimer***), this is called out in the text. Timeouts of the timers are listed as conditions on state transitions.

Conditions listed on state transitions will come from one of three sources:

- Events triggered within the Policy Engine e.g. timer timeouts.
- Message and related indications passed up to the Policy Engine from the Protocol Layer (message sent, message received etc.)
- Information and requests coming from the Device Policy manager relating either to Local Policy, or to other modules which the Device Policy Manager controls such as Power Supply and Cable Detection.

8.3.3.2 Policy Engine Source Port State Diagram

Figure 8-20 below shows the state diagram that shall be used by the Policy Engine in a Source Port. The following sections describe operation in each of the states.


Hard reset signalling received
Start
Transition to default
Actions on entry:
Request power supply transition to default
Reset local HW
Power = rising/falling to default (5V)
PD = unattached
Power source at default
Startup
Actions on entry:
Reset Protocol Layer
Reset CapsCounter
Power = Default (5V)
PD = unattached
Protocol Layer reset
Discovery
Actions on entry:
If Provider or Provider/Consumer request A-plug attachment status from Device Policy Manager.
If A-plug attached or Consumer/Provider initialize and run SourceCapabilityTimer
Power = default (5V)
PD = unattached
NoResponseTimer timeout & HardResetCounter > nHardResetCount¹
Disabled
Actions on entry:
Disable Power Delivery
Power = Default (5V)
PD = attached/unattached
(NoResponseTimer timeout & HardResetCounter ≤ nHardResetCount) | Hard Reset request from Device Policy Manager
Hard reset complete
(SourceCapabilityTimer timeout & CapsCounter > nCapsCount)
Hard Reset
Actions on entry:
Generate Hard Reset signalling
Initialize and start NoResponseTimer
Increment HardResetCounter
Power = Default or negotiated
PD = attached/unattached
Source capability change (from Device Policy Manager)
(SourceCapabilityTimer timeout & CapsCounter ≤ nCapsCount²) | Get_Source_Cap message received
(Capabilities message sending failure (without GoodCRC) &not presently attached) | (NoResponseTimer timeout &HardResetCounter > nHardResetCount¹)
Send capabilities
Actions on entry:
Request present source capabilities from Device Policy Manager
Send PD Capabilities message
Initialize and run SenderResponseTimer
Increment CapsCounter (optional)²
If GoodCRC received stop NoResponseTimer and reset HardResetCounter and CapsCounter
Power = Default or negotiated
PD = attached/unattached
NoResponseTimer timeout & HardResetCounter > nHardResetCount¹
(SenderResponseTimer timeout & Negotiated power Invalid)
Reject/Wait message sent & Negotiated power Invalid
Capability Response
Actions on entry:
Send Reject message if request can't be met
Send Wait messageif request could be be met later from the reserve
Power = Default or negotiated
PD = attached
Request message received
Request can't be met | Request met later from reserve
SenderResponseTimer Timeout & Negotiated power still valid
Negotiate Capability
Actions on entry:
Get Device Policy Manager evaluation of sink requirements:
Can be met
Can't be met
Could be met later from reserve
If the sink still requires more power ("NoMatch") this information will be passed to Device Policy Manager
Power = Default or negotiated
PD = attached
Request message received
Request can be met
Transition Supply
Actions on entry:
Initialize and run SourceActivityTimer (see Section 8.3.3.8)
If GotoMin send GotoMin message (within SenderResponseTime)
Else send Accept message (within SenderResponseTime)
Request Device Policy Manager transition Power Supply
Actions on exit:
Send PS_RDY message (unless error)
Power = transition
PD = attached
Reject/Wait message sent & Negotiated power still valid
GotoMin request from Device Policy Manager
Power supply ready
Ready
Actions on entry:
Initialize and run SourceActivityTimer (see Section 8.3.3.8)
Power = Default or negotiated
PD = attached
Get_Source_Cap message received
Get Source Cap
Actions on entry:
Request source capabilities from Device Policy Manager
Send Capabilities message
Power = Default or negotiated
PD = attached
Capabilities message sent
get sink capabilities request from Device Policy Manager
Get Sink Cap
Actions on entry:
Send Get_Sink_Cap message
Initialize and run SenderResponseTimer
Actions on exit:
Pass sink capabilities/outcome to Device Policy Manager
Power = Default or negotiated
PD = attached
Sink capabilities message received | SenderResponseTimer timeout

[1] If the NoResponseTimer times Out and the HardResetCounter > nHardResetCount the Policy Engine can either continue sending capabilities or go to the ***Disabled*** state.
[2] Implementationof the CapsCounter is optional. In the case where this is not implemented the Source shall continue to send Capabilities messages each time the SourceCapabilityTimer times out.

Figure 8-20 Source Port Policy Engine state diagram

8.3.3.2.1 Startup

Startup shall be the starting state for a Source Policy Engine either on power up or after a Hard Reset. On entry to this state the Policy Engine shall reset the Protocol Layer and reset the ***CapsCounter***.

Once the reset has completed the Policy Engine shall transition to the ***Discovery*** state.

8.3.3.2.2 Discovery

On entry to the ***Discovery*** state the Policy Engine shall perform the following:

- If this is a Policy Engine in a Provider or a Provider/Consumer request A-plug attachment status from the Device Policy manager
- Initialize and run the ***SourceCapabilityTimer*** in order to trigger sending a ***Capabilities*** message if one of the following is true:
    - This is a Provider or Provider/Consumer and an A-plug is attached.
    - This is a Consumer/Provider.

Note: Providers or Provider/Consumers without an A-plug attached will remain in the Discovery state, without sending any ***Capabilities*** messages until an A-plug is attached.

The Policy Engine shall transition to the ***Send Capabilities*** state when:

- The ***SourceCapabilityTimer*** times out and ***CapsCounter*** ≤ ***nCapsCount***.
- Or a ***Get_Source_Cap*** message is received from an attached Sink Port.

The Policy Engine shall transition to the ***Hard Reset*** state when:

- The ***SourceCapabilityTimer*** times out and ***CapsCounter*** > ***nCapsCount***.

The Policy Engine may optionally go to the ***Disabled*** state when:

- The ***NoResponseTimer*** times out and the ***HardResetCounter*** > ***nHardResetCount***. Note in this case the attached device is assumed to be unresponsive. The Policy Engine operates as if the device is unattached until such time as a detach/reattach is detected.

8.3.3.2.3 Send Capabilities

A capabilities change event from the Device Policy Manager while in any state shall trigger the Policy Engine to enter the ***Send Capabilities*** state. There are also transitions from the ***Capability Response***, ***Soft Reset*** and ***Protocol Error*** states.

On entry to the ***Send Capabilities*** state the Policy Engine shall request the present port capabilities from the Device Policy Manager. The Policy Engine shall then request the Protocol Layer to send a ***Capabilities*** message containing these capabilities and shall initialize and run the ***SenderResponseTimer*** and increment the ***CapsCounter*** (if implemented). If a ***GoodCRC*** message is received then the Policy Engine shall stop the ***NoResponseTimer*** and reset the ***HardResetCounter*** to zero. Note that the ***HardResetCounter*** shall only be reset to zero in this state and at power up; its value shall be maintained during a Hard Reset.

The Policy Engine shall transition to the ***Negotiate Capability*** state when:

- A ***Request*** message is received from an attached Sink Port.

The Policy Engine shall transition to the ***Discovery*** state when:

- The Protocol Layer indicates that the message has not been sent and we are presently unattached. This is part of the Capabilities sending process whereby successful message sending indicates attachment to a PD Sink Port.

The Policy Engine shall transition to the ***Ready*** state when:

- The ***SenderResponseTimer*** times out and the current power being used is still within the Source's present capabilities. No response from the Sink indicates it is content to continue to use the present power.

The Policy Engine shall transition to the ***Hard Reset*** state when:

- The ***SenderResponseTimer*** times out and the present power being used is outside the Source's present capabilities. In this case a transition back to default power is required.

When the ***NoResponseTimer*** times out and the ***HardResetCounter*** > ***nHardResetCount*** the Policy Engine shall do one of the following:

- Transition to the ***Discovery*** State.
- Transition to the ***Disabled*** state.

Note that in either case the attached device is assumed to be unresponsive. The Policy Engine should operate as if the device is unattached until such time as a detach/reattach is detected.

8.3.3.2.4 Negotiate Capability

On entry to the ***Negotiate Capability*** state the Policy Engine shall ask the Device Policy Manager to evaluate the Request from the attached Sink. The response from the Device Policy Manager shall be one of the following:

- The Request can be met.
- The Request cannot be met
- The Request could be met later from the reserve.

The Policy Engine shall transition to the ***Transition Supply*** state when:

- The Request can be met.

The Policy Engine shall transition to the ***Capability Response*** state when:

- The Request cannot be met.
- Or the Request can be met later from the reserve.

8.3.3.2.5 Transition Supply

The Policy Engine shall be in the ***Transition Supply*** state while the Power Supply is transitioning from one power to another.

On entry to the ***Transition Supply*** state, the Policy Engine shall initialize and run the ***SourceActivityTimer*** (see Section 8.3.3.5 for details of ***Ping*** messaging for Source Ports), request the Protocol Layer to either send a ***GotoMin*** message (if this was requested by the Device Policy Manager) or otherwise an ***Accept*** message and inform the Device Policy Manager that it shall transition the Power Supply to the Requested value of power. Note: that if the Power Supply is currently operating at the requested power no change will be necessary.

On exit from the ***Transition Supply*** state the Policy Engine shall request the Protocol Layer to send a ***PS_RDY*** message.

The Policy Engine shall transition to the ***Ready*** state when:

- The Device Policy Manager informs the Policy Engine that the Power Supply is ready.

8.3.3.2.6 Ready

In the ***Ready*** state the PD Source shall operating at a stable power with no ongoing negotiation. It shall respond to requests from the Sink, events from the Device Policy Manager and shall send out ***Ping*** messages to maintain the PD link.

On entry to the ***Ready*** state the Policy Engine shall initialize and run the ***SourceActivityTimer*** (see Section 8.3.3.5 for details of ***Ping*** messaging for Source Ports).

The Policy Engine shall transition to the ***Transition supply*** state when:

- A GoToMin request is received from the Device Policy Manager for the attached Device to go to minimum power.

The Policy Engine shall transition to the ***Get Source Cap*** state when:

- A ***Get_Source_Cap*** message is received from the Sink.

The Policy Engine shall transition to the ***Get Sink Cap*** state when:

- The Device Policy Manager asks for the Sink's capabilities.

8.3.3.2.7 Disabled

In the ***Disabled*** state the PD Source supplies default power and is unresponsive to USB Power Delivery messaging. In order for the PD Source to transition to USB Power Delivery operation a Hard Reset or some other form of reset is required.

8.3.3.2.8 Capability Response

The Policy Engine shall enter the ***Capability Response*** state if there is a Request received from the Sink that cannot be met based on the present capabilities.

On entry to the ***Capability Response*** state the Policy Engine shall request the Protocol Layer to send one of the following:

- ***Reject*** message – if the request cannot be met
- ***Wait*** message – if the request could be met later from the reserve.

The Policy Engine shall transition to the ***Ready*** state when:

- The ***Reject*** message or ***Wait*** message has been sent and the present value of power is still valid.

The Policy Engine shall transition to the ***Hard Reset*** state when:

- The ***Reject*** message or ***Wait*** message has been sent and the present value of power is not valid (i.e. the Sink had to request a new value so instead we will return to default operation).

8.3.3.2.9 Hard Reset

The Protocol Engine shall transition from any state to the ***Hard Reset*** state when

- The ***NoResponseTimer*** times out and the ***HardResetCounter*** ≤ ***nHardResetCount*** or
- The Device Policy Manager requests a Hard Reset.

On entry to the ***Hard Reset*** state the Policy Engine shall request the generation of ***Hard Reset*** signaling by the PHY, initialize and run the ***NoResponseTimer*** and increment the ***HardResetCounter***. Note that the ***NoResponseTimer*** shall continue to run in every state until it is stopped or times out.

The Policy Engine shall transition to the ***Transition to default*** state when:

- The Hard Reset is complete.

8.3.3.2.10 Transition to default

The Policy Engine shall transition from any state to the ***Transition to default*** state when:

- ***Hard Reset*** signaling is detected.

On entry to the ***Transition to default*** state the Policy Engine shall indicate to the Device Policy Manager that the Power Supply shall transition to default (5V). The Policy Engine shall then request a reset of the local hardware.

The Policy Engine shall transition to the ***Startup*** state when:

- The Device Policy Manager indicates that the Power Supply has reached the default level.

8.3.3.2.11 Get Source Cap

On entry to the ***Get Source Cap*** state the Policy Engine shall request the Device Policy Manager for the current system capabilities. The Policy Engine shall then request the Protocol Layer to send a ***Capabilities*** message containing these capabilities.

The Policy Engine shall transition to the ***Ready*** state when:

- The ***Capabilities*** message has been successfully sent.

8.3.3.2.12 Get Sink Cap

In this state the Policy Engine, due to a request from the Device Policy Manager, shall request the Sink's capabilities from the attached Sink.

On entry to the ***Get Sink Cap*** state the Policy Engine shall request the Protocol Layer to send a ***Get_Sink_Cap*** message in order to retrieve the Sink's capabilities. The Policy Engine shall then start the ***SenderResponseTimer***.

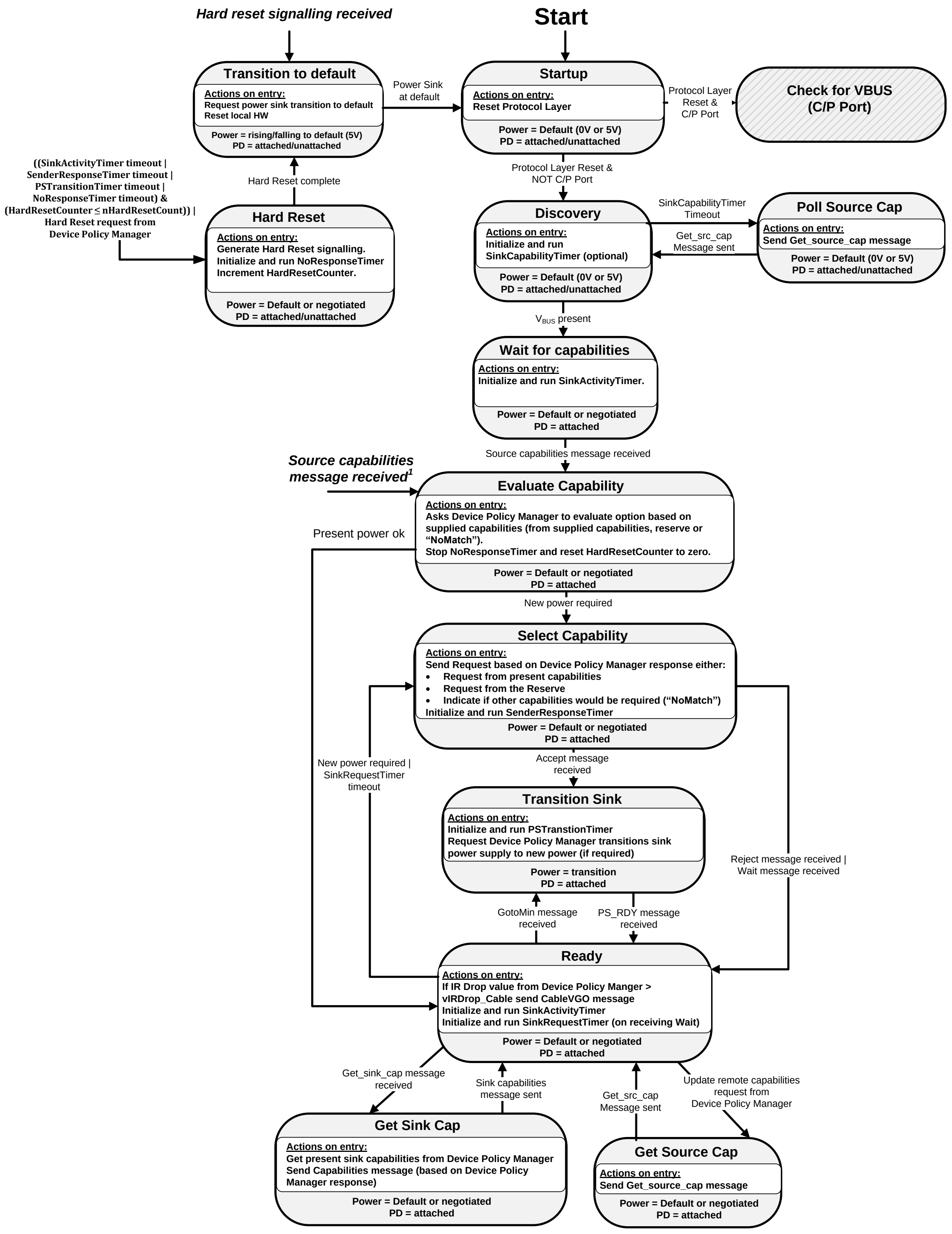
On exit from the ***Get Sink Cap*** state the Policy Engine shall inform the Device Policy Manager of the outcome (capabilities, response timeout or reject).

The Policy Engine shall transitions to the ***Ready*** state when:

- A Sink ***Capabilities*** message is received.
- Or ***SenderResponseTimer*** times out.

8.3.3.3 Policy Engine Sink Port State Diagram

Figure 8-21 below shows the state diagram that shall be followed for the Policy Engine in a Source Port. The following sections describe operation in each of the states.



Hard reset signalling received
Start
Transition to default
Actions on entry:
Request power sink transition to default
Reset local HW
Power = rising/falling to default (5V)
PD = attached/unattached
Power Sink at default
Startup
Actions on entry:
Reset Protocol Layer
Power = Default (0V or 5V)
PD = attached/unattached
Protocol Layer Reset & C/P Port
Check for VBUS (C/P Port)
({SinkActivityTimer timeout | SenderResponseTimer timeout | PSTransitionTimer timeout | NoResponseTimer timeout} & (HardResetCounter ≤ nHardResetCount)) | Hard Reset request from Device Policy Manager
Hard Reset complete
Hard Reset
Actions on entry:
Generate Hard Reset signalling.
Initialize and run NoResponseTimer
Increment HardResetCounter.
Power = Default or negotiated
PD = attached/unattached
Protocol Layer Reset & NOT C/P Port
Discovery
Actions on entry:
Initialize and run SinkCapabilityTimer (optional)
Power = Default (0V or 5V)
PD = attached/unattached
SinkCapabilityTimer Timeout
Get_src_cap Message sent
Poll Source Cap
Actions on entry:
Send Get_source_cap message
Power = Default (0V or 5V)
PD = attached/unattached
V_BUS present
Wait for capabilities
Actions on entry:
Initialize and run SinkActivityTimer.
Power = Default or negotiated
PD = attached
Source capabilities message received
Source capabilities message received^1
Evaluate Capability
Actions on entry:
Asks Device Policy Manager to evaluate option based on supplied capabilities (from supplied capabilities, reserve or “NoMatch”).
Stop NoResponseTimer and reset HardResetCounter to zero.
Power = Default or negotiated
PD = attached
Present power ok
New power required
Select Capability
Actions on entry:
Send Request based on Device Policy Manager response either:
• Request from present capabilities
• Request from the Reserve
• Indicate if other capabilities would be required (“NoMatch”)
Initialize and run SenderResponseTimer
Power = Default or negotiated
PD = attached
New power required | SinkRequestTimer timeout
Accept message received
Transition Sink
Actions on entry:
Initialize and run PSTranstionTimer
Request Device Policy Manager transitions sink power supply to new power (if required)
Power = transition
PD = attached
Reject message received | Wait message received
GotoMin message received
PS_RDY message received
Ready
Actions on entry:
If IR Drop value from Device Policy Manger > vIRDrop_Cable send CableVGO message
Initialize and run SinkActivityTimer
Initialize and run SinkRequestTimer (on receiving Wait)
Power = Default or negotiated
PD = attached
Get_sink_cap message received
Sink capabilities message sent
Get_src_cap Message sent
Update remote capabilities request from Device Policy Manager
Get Sink Cap
Actions on entry:
Get present sink capabilities from Device Policy Manager
Send Capabilities message (based on Device Policy Manager response)
Power = Default or negotiated
PD = attached
Get Source Cap
Actions on entry:
Send Get_source_cap message
Power = Default or negotiated
PD = attached

Note 1: Source capabilities messages will be ignored in the ***Hard Reset*** and ***Transition to Default*** states and during a Soft Reset.

Figure 8-21 Sink Port state diagram

8.3.3.3.1 Startup

Startup shall be the starting state for a Sink Policy Engine either on power up or after a Hard Reset. On entry to this state the Policy Engine shall reset the Protocol Layer.

Once the reset process completes, the Policy Engine shall transition to the ***Discovery*** state for a Consumer only and to the ***Check for VBUS*** state for a Consumer/Provider (see Section 8.3.3.6.7).

8.3.3.3.2 Discovery

On entry to the ***Discovery*** state the Policy Engine shall perform the following:

- Initialize and run the ***SinkCapabilityTimer*** in order to trigger the polling of Source capabilities using ***Get_Source_Cap*** message.

The Policy Engine shall transition to the ***Wait for Capabilities*** state when:

- The Device Policy Manager indicates that V_{BUS} has been detected.

8.3.3.3.3 Wait for Capabilities

On entry to the ***Wait for capabilities*** state the Policy Engine shall initialize and start the ***SinkActivityTimer***.

The Policy Engine shall transition to the ***Evaluate Capability*** state when:

- Source capabilities are received.

When the ***SinkActivityTimer*** times out, the Policy Engine will perform a Hard Reset.

8.3.3.3.4 Evaluate Capability

Whenever the Policy Engine receives a Source ***Capabilities*** message (except in the ***Hard Reset*** and ***Transition to Default*** states or during a Soft Reset) it shall transition to the ***Evaluate Capability*** state. At this point the Sink knows that it is attached to and communicating with a PD capable Source.

On entry to the ***Evaluate Capability*** state the Policy Engine shall request the Device Policy Manager to evaluate the supplied Source capabilities based on Local Policy and provide one of the following responses:

- Sink is content to operate at the present power level.
- Selection from the present offered capabilities.
- Selection from Reserve.
- Power mismatch; offered power does not meet the device's requirements.

The Policy Engine shall transition to the ***Select Capability*** state when:

- A new power level is required.

The Policy Engine shall transition to the ***Ready*** state when:

- The Sink is content to operate at the present power level.

8.3.3.3.5 Select Capability

On entry to the ***Select Capability*** state the Policy Engine shall request the Protocol Layer to send a response message, based on the evaluation from the Device Policy Manager. The message shall be one of the following:

- A Request from the offered Source Capabilities.
- A Request from the Source's Reserve for this Sink.
- A Request from the offered Source Capabilities with an indication that another power level would be preferred.

The Policy Engine shall initialize and run the ***SenderResponseTimer***.

The Policy Engine shall transition to the ***Transition Sink*** state when:

- An ***Accept*** message is received from the Source.

The Policy Engine shall transition to the ***Ready*** state when:

- A ***Reject*** message is received from the Source.
- A ***Wait*** message is received from the Source.

8.3.3.3.6 Transition Sink

On entry to the ***Transition Sink*** state the Policy Engine shall initialize and run the ***PSTransitionTimer*** (timeout will lead to a Hard Reset see Section 8.3.3.3.8 and shall then request the Device Policy Manager to transition the Sink's Power Supply to the new power level. Note that if there is no power level change the Device Policy Manager should not affect any change to the Power Supply.

The Policy Engine shall transition to the ***Ready*** state when:

- A ***PS_RDY*** message is received from the Source.

8.3.3.3.7 Ready

In the ***Ready*** state the PD Sink shall be operating at a stable power level with no ongoing negotiation. It shall respond to requests from the Source, events from the Device Policy Manager and may monitor for ***Ping*** messages to maintain the PD link.

On entry to the ***Ready*** state the Policy Engine shall do the following:

- If IR drop value given by Device Policy Manager exceeds ***vIRDrop_Cable***, then request the Protocol Layer to send a ***CableVGO*** message

On entry to the ***Ready*** state as the result of a wait the Policy Engine should do the following:

- Initialize and run the ***SinkRequestTimer***.

On entry to the ***Ready*** state as the result of a wait the Policy Engine shall do the following:

- Initialize and run the ***SinkActivityTimer***.

The Policy Engine shall transition to the ***Select Capability*** state when:

- A new power level is requested by the Device Policy Manager.
- A ***SinkRequestTimer*** timeout occurs.

The Policy Engine shall transition to the ***Transition Sink*** state when:

- A ***GotoMin*** message is received.

The Policy Engine shall transition to the ***Hard Reset*** state when:

- A ***SinkActivityTimer*** timeout occurs.

The Policy Engine shall transition back to the ***Ready*** state when:

- A ***Ping*** message is received. Note this should not cause the ***SinkRequestTimer*** to be reinitialized.

The Policy Engine shall transition to the ***Get Sink Cap*** state when:

- When a ***Get_Sink_Cap*** message is received from the Protocol Layer.

The Policy Engine shall transition to the ***Get Source Cap*** state when:

- When the Device Policy Manager requests an update of the remote Source's capabilities.

8.3.3.3.8 Hard Reset

The Policy Engine shall transition to the ***Hard Reset*** state from any state when:

- ((**SinkActivityTimer** timeout |
- **SenderResponseTimer** timeout |
- **PSTransitionTimer** timeout |
- **NoResponseTimer** timeout) &
- (**HardResetCounter** ≤ **nHardResetCount**)) |

- Hard Reset request from Device Policy Manager

Note: if the ***NoResponseTimer*** times out and the ***HardResetCounter*** is greater than ***nHardResetCount*** the Sink shall assume that the Source is non-responsive.

Note: The ***nHardResetCount*** is reset on a power cycle or detach.

On entry to the ***Hard Reset*** state the Policy Engine shall request the generation of ***Hard Reset*** signaling by the PHY and increment the ***HardResetCounter***. Note that the ***NoResponseTimer*** shall continue to run in every state until it is stopped or times out.

The Policy Engine shall transition to the ***Transition to default*** state when:

- The Hard Reset is complete.

8.3.3.3.9 Transition to default

The Policy Engine shall transition from any state to ***Transition to default*** state when:

- ***Hard Reset*** signaling is detected.

When ***Hard Reset*** signaling is detected or message sending in the Protocol Layer fails after retries (i.e. a ***GoodCRC*** message is not received) then the Policy Engine shall transition from any state to ***Transition to default***. This state can also be entered from the ***Hard Reset*** state.

On entry to the ***Transition to default*** state the Policy Engine shall indicate to the Device Policy Manager that the Sink shall transition to default. The Policy Engine shall then request a reset of the local hardware.

The Policy Engine shall transition to the ***Startup*** state when:

- The Device Policy Manager indicates that the Sink has reached the default level.

8.3.3.3.10 Get Sink Cap

On entry to the ***Get Sink Cap*** state the Policy Engine shall request the Device Policy Manager for the current system capabilities. The Policy Engine shall then request the Protocol Layer to send a Sink ***Capabilities*** message containing these capabilities.

The Policy Engine shall transition to the ***Ready*** state when:

- The ***Capabilities*** message has been successfully sent.

8.3.3.3.11 Get Source Cap

On entry to the ***Get Source Cap*** state the Policy Engine shall request the Protocol Layer to send a ***Get_Source_Cap*** message in order to retrieve the Source capabilities.

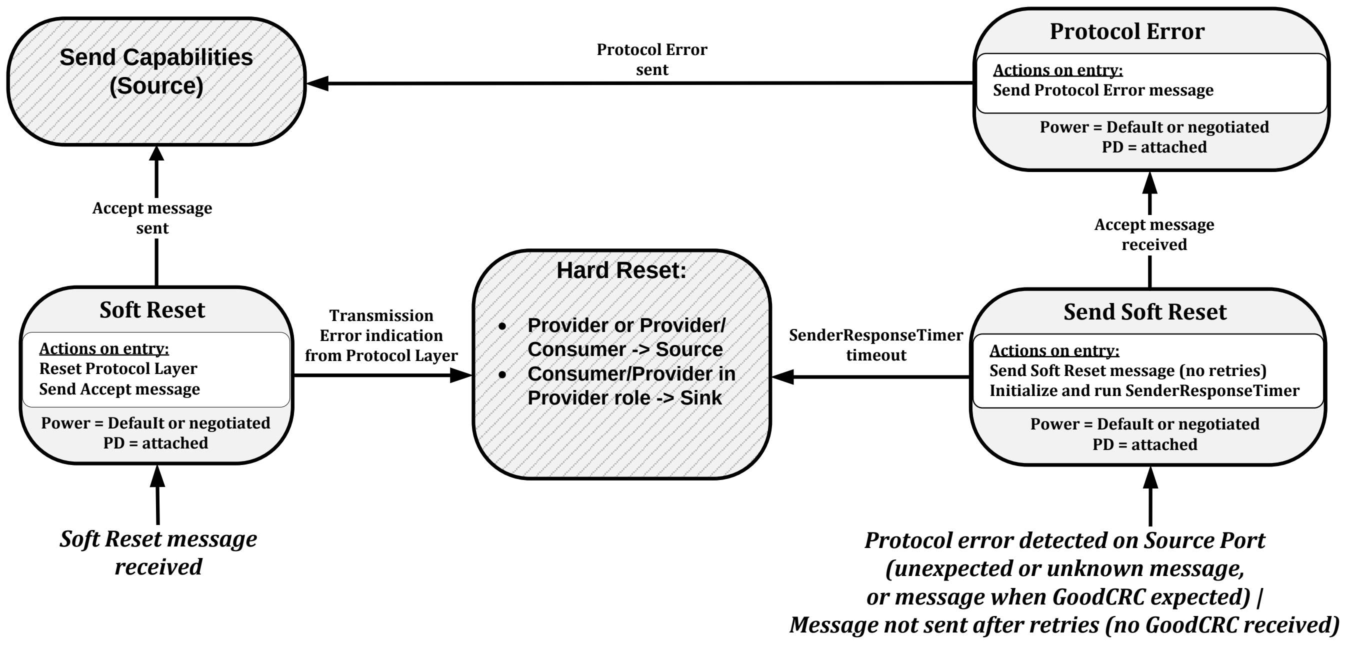
The Policy Engine shall transition to the ***Ready*** state when:

- The ***Get_Source_Cap*** message has been sent.

8.3.3.4 Soft Reset State Diagrams

8.3.3.4.1 Source Port Soft Reset State Diagram

Figure 8-22 below shows the state diagram that shall be followed for the Policy Engine in a Source Port when performing a Soft Reset. The following sections describe operation in each of the states.



Send Capabilities (Source)
Protocol Error sent
Protocol Error
Actions on entry:
Send Protocol Error message
Power = Default or negotiated
PD = attached
Accept message sent
Accept message received
Soft Reset
Actions on entry:
Reset Protocol Layer
Send Accept message
Power = Default or negotiated
PD = attached
Transmission Error indication from Protocol Layer
Hard Reset:
Provider or Provider/ Consumer -> Source
Consumer/Provider in Provider role -> Sink
SenderResponseTimer timeout
Send Soft Reset
Actions on entry:
Send Soft Reset message (no retries)
Initialize and run SenderResponseTimer
Power = Default or negotiated
PD = attached
Soft Reset message received
Protocol error detected on Source Port (unexpected or unknown message, or message when GoodCRC expected) | Message not sent after retries (no GoodCRC received)

Figure 8-22 Source Port Protocol Soft Reset Diagram

8.3.3.4.1.1 Send Soft Reset

The ***Send Soft Reset*** state shall be entered from any state when a Protocol Error is detected by the Protocol Layer or when a message has not been sent after retries.

On entry to the ***Send Soft Reset*** state the Policy Engine shall request the Protocol Layer to perform a Soft Reset and then shall send a ***Soft Reset*** message to the Sink.

The Policy Engine shall transition to the ***Protocol Error*** state when:

- An ***Accept*** message has been received.

The Policy Engine shall transition to the ***Hard Reset*** state when:

- A ***SenderResponseTimer*** timeout occurs.

The decision as to whether to go to ***Hard Reset (Source)*** or ***Hard Reset (Sink)*** shall depend on the type of device:

- The Source port in a Provider or Provider/Consumer shall go to ***Hard Reset (Source)***.
- The Source port in a Consumer/Provider shall go to ***Hard Reset (Sink)*** i.e. revert to default operation as Sink Port.

8.3.3.4.1.2 Protocol Error

On entry to the ***Protocol Error*** state the Policy Engine shall request the Protocol Layer to send a ***Protocol Error*** message indicating that a protocol error has occurred.

The Policy Engine shall transition to the ***Send Capabilities (Source)*** state when:

- The ***Protocol Error*** message has been sent.

8.3.3.4.1.3 Soft Reset

The ***Soft Reset*** state shall be entered from any state when a Reset message is received from the Protocol Layer.

On entry to the ***Soft Reset*** state the Policy Engine shall reset the Protocol Layer and shall then request the Protocol Layer to send an ***Accept*** message.

The Policy Engine shall transition to the ***Send Capabilities*** state for a Source Port (see Section 8.3.3.2.3) when:

- The *Accept* message has been sent.

The Policy Engine shall transition to the ***Hard Reset*** state when:

- The Protocol Layer indicates that a transmission error has occured.

The decision as to whether to go to ***Hard Reset (Source)*** or ***Hard Reset (Sink)*** shall depend on the type of device:

- The Source port in a Provider or Provider/Consumer shall go to ***Hard Reset (Source)***.
- The Source port in a Consumer/Provider shall go to ***Hard Reset (Sink)*** i.e. revert to default operation as Sink Port.

8.3.3.4.2 Sink Port Protocol Soft Reset State Diagram

Figure 8-23 below shows the state diagram that shall be followed for the Policy Engine in a Sink Port when performing a Soft Reset. The following sections describe operation in each of the states.

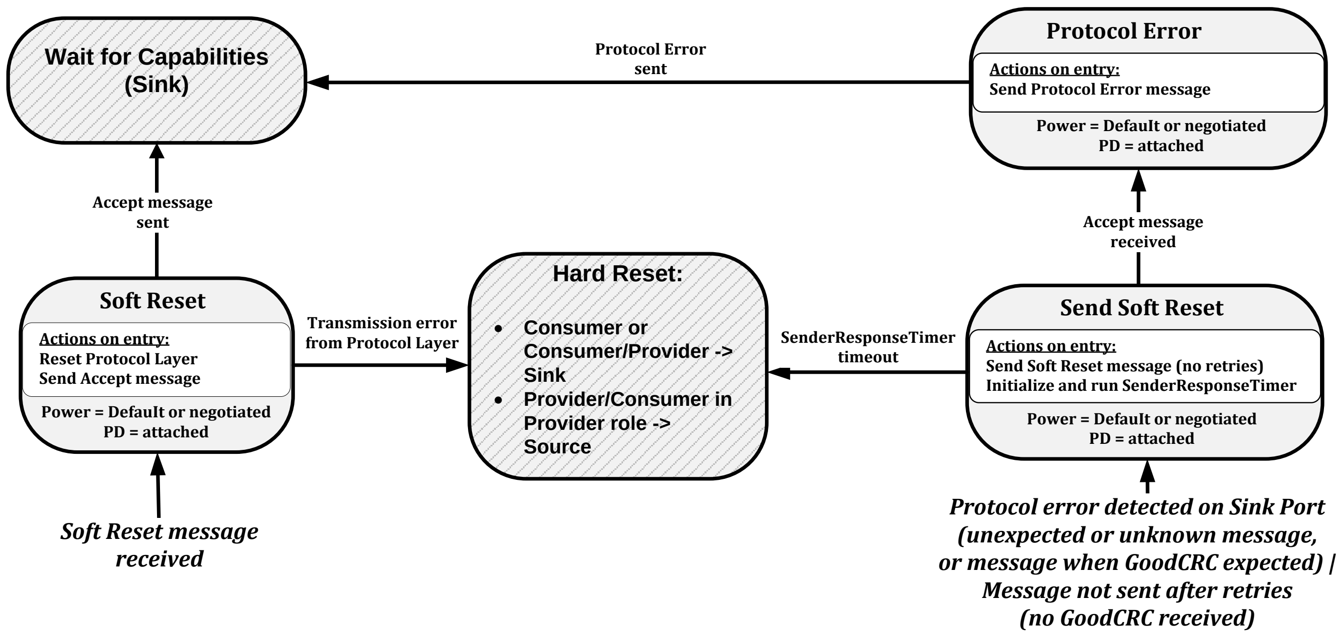

Wait for Capabilities (Sink)
Protocol Error sent
Protocol Error
Actions on entry:
Send Protocol Error message
Power = Default or negotiated
PD = attached
Accept message sent
Accept message received
Soft Reset
Actions on entry:
Reset Protocol Layer
Send Accept message
Power = Default or negotiated
PD = attached
Transmission error from Protocol Layer
Hard Reset:
Consumer or Consumer/Provider -> Sink
Provider/Consumer in Provider role -> Source
SenderResponseTimer timeout
Send Soft Reset
Actions on entry:
Send Soft Reset message (no retries)
Initialize and run SenderResponseTimer
Power = Default or negotiated
PD = attached
Soft Reset message received
Protocol error detected on Sink Port (unexpected or unknown message, or message when GoodCRC expected) | Message not sent after retries (no GoodCRC received)

Figure 8-23 Sink Port Protocol Soft Reset Diagram

8.3.3.4.2.1 Send Soft Reset

The ***Send Soft Reset*** state shall be entered from any state when a Protocol Error is detected by the Protocol Layer or when a message has not been sent after retries.

On entry to the ***Send Soft Reset*** state the Policy Engine shall request the Protocol Layer to perform a Soft Reset and then shall send a *Soft Reset* message to the Sink.

The Policy Engine shall transition to the ***Protocol Error*** state when:

- An *Accept* message has been received.

The Policy Engine shall transition to the ***Hard Reset*** state when:

- A *SenderResponseTimer* timeout occurs.

The decision as to whether to go to ***Hard Reset (Source)*** or ***Hard Reset (Sink)*** shall depend on the type of device:

- The Sink port in a Consumer or Consumer/Provider shall go to ***Hard Reset (Sink)***.
- The Sink port in a Provider/Consumer shall go to ***Hard Reset (Source)*** i.e. revert to default operation as Source Port.

8.3.3.4.2.2 Protocol Error

On entry to the ***Protocol Error*** state the Policy Engine shall request the Protocol Layer to send a ***Protocol Error*** message indicating that a protocol error has occurred.

The Policy Engine shall transition to the ***Wait for Capabilities (Source)*** state when:

- The ***Protocol Error*** message has been sent.

8.3.3.4.2.3 Soft Reset

The ***Soft Reset*** state shall be entered from any state when a Reset message is received from the Protocol Layer.

On entry to the ***Soft Reset*** state the Policy Engine shall reset the Protocol Layer and shall then request the Protocol Layer to send an ***Accept*** message.

The Policy Engine shall transition to the ***Wait for Capabilities*** state when:

- The ***Accept*** message has been sent.

The Policy Engine shall transition to the ***Hard Reset*** state when:

- The Protocol Layer indicates that a transmission error has occured.

The decision as to whether to go to ***Hard Reset (Source)*** or ***Hard Reset (Sink)*** shall depend on the type of device:

- The Sink port in a Consumer or Consumer/Provider shall go to ***Hard Reset (Sink)***.
- The Sink port in a Provider/Consumer shall go to ***Hard Reset (Source)*** i.e. revert to default operation as Source Port.

8.3.3.5 Source Port Ping State Diagram

Figure 8-24 shows the state diagram that shall be followed for a ***Ping*** message from a Source Port. Note: Pings are optional under certain operating conditions (see Section 6.3.5).

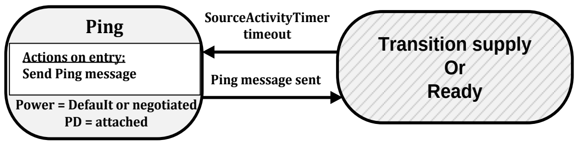

Ping
Actions on entry:
Send Ping message
Power = Default or negotiated
PD = attached
SourceActivityTimer timeout
Ping message sent
Transition supply Or Ready

Figure 8-24 Source Port Ping State Diagram

8.3.3.5.1 Ping

On entry to the ***Ping*** state (from the ***Transition Supply*** or ***Ready*** states) the Policy Engine shall request the Protocol Layer to send a ***Ping*** message.

The Policy Engine shall transition back to the previous state (***Transition Supply*** or ***Ready)*** state (see Figure 8-20) when:

- The ***Ping*** message has been successfully sent.

On re-entry to the Transition Supply or Ready states the Policy Engine shall not perform any of the "Actions on Entry" except for initializing and running the SourceActivityTimer.

8.3.3.6 Dual-Role Port State Diagrams

Dual-Role Ports that combine Source and Sink capabilities shall comprise Source and Sink Policy Engine state machines. In addition they shall have the capability to Swap roles from the ***Ready*** state and shall return to default operation on a Hard Reset or non-Protocol Error. The following sections describe how the Source and Sink state diagrams shall be combined in the case of Hard Reset, the role swap process and the operation in the case of a dead battery.

8.3.3.6.1 Dual-Role (initially Source Port) Ping State Diagram

Figure 8 25 shows the state diagram that shall be followed for a Dual-Role Port which is initially a Source Port. Note: Pings are optional under certain operating conditions (see Section 6.3.5).

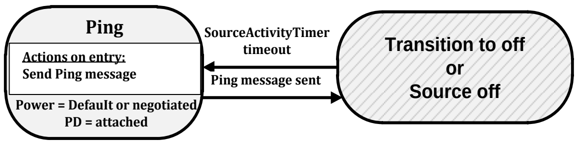

Ping
Actions on entry:
Send Ping message
Power = Default or negotiated
PD = attached
SourceActivityTimer timeout
Ping message sent
Transition to off or Source off

Figure 8-25 Dual-Role (initially Source Port) Ping State Diagram

8.3.3.6.1.1 Ping

On entry to the Ping state from the ***Transition to off*** or ***Source Off*** state, during a Role Swap by a Dual-Role port which is initially a Source Port, the Policy Engine shall request the Protocol Layer to send a Ping message.

The Policy Engine shall transition back to either the ***Transition to off*** or ***Source Off*** state (see Figure 8-20) when:

- The ***Ping*** message has been successfully sent.

8.3.3.6.2 Dual-Role (initially Sink Port) Ping State Diagram

Figure 8-26 shows the state diagram that shall be followed for a Dual-Role Port which is initially a Sink Port. Note: Pings are optional under certain operating conditions (see Section 6.3.5).

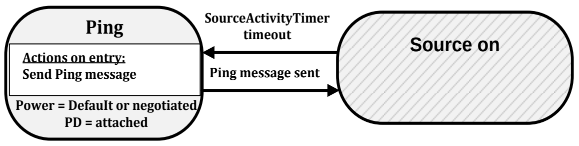


Figure 8-26 Dual-Role (initially Sink Port) Ping State Diagram

8.3.3.6.2.1 Ping

On entry to the Ping state from the ***Source on*** state the Policy Engine shall request the Protocol Layer to send a ***Ping*** message.

The Policy Engine shall transition back to the ***Source on*** state (see Figure 8-21) when:

- The ***Ping*** message has been successfully sent.

8.3.3.6 .3 Hard Reset of Policy Engine in a Provider/Consumer in Sink Role

Figure 8-27 shows the state transitions that shall be followed in the case where a Provider/Consumer with a port operating in Sink Role is required to perform a Hard Reset.

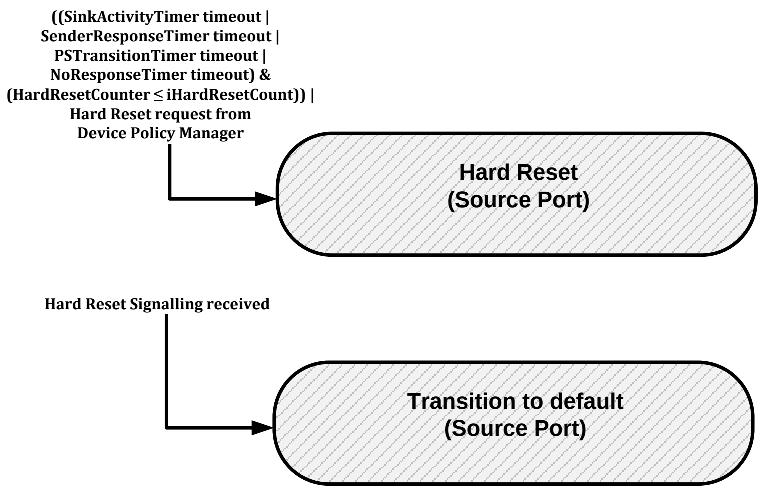

((SinkActivityTimer timeout | SenderResponseTimer timeout | PSTransitionTimer timeout | NoResponseTimer timeout) & (HardResetCounter ≤ iHardResetCount)) | Hard Reset request from Device Policy Manager
Hard Reset (Source Port)
Hard Reset Signalling received
Transition to default (Source Port)

Figure 8-27 State Diagram for Hard Reset of P/C in Sink Role

8.3.3.6.3.1 Hard Reset

The Policy Engine shall transition to the ***Hard Reset*** state for a Source Port from any state when:

- ((**SinkActivityTimer** timeout |
- **SenderResponseTimer** timeout |
- **PSTransitionTimer** timeout |
- **NoResponseTimer** timeout) &
- (**HardResetCounter** ≤ **nHardResetCount**)) |
- Hard Reset request from Device Policy Manager

8.3.3.6.3.2 Transition to default

The Policy Engine shall transition from any state to the ***Transition to default*** state for a Source Port when:

- ***Hard Reset*** signaling is detected.

8.3.3.6.4 Hard Reset of Policy Engine in a Consumer/Provider in Source Role

Figure 8-28 shows the state transitions that shall be followed in the case where a Consumer/Provider with a port operating in Source Role is required to perform a Hard Reset.

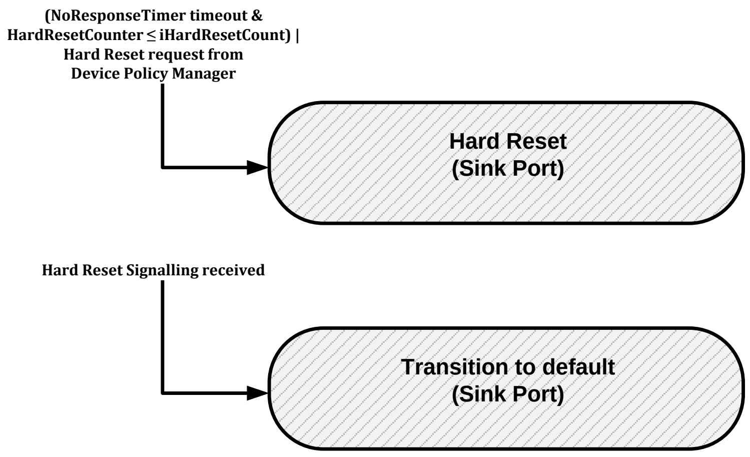


Figure 8-28 State Diagram for the Hard Reset of a C/P in Source Role

8.3.3.6.4.1 Hard Reset

The Protocol Engine shall transition from any state to the ***Hard Reset*** state for a Sink Port when

- The ***NoResponseTimer*** times out and the ***HardResetCounter*** ≤ ***nHardResetCount*** or
- The Device Policy Manager requests a Hard Reset.

8.3.3.6.4.2 Transition to default

The Policy Engine shall transition from any state to the ***Transition to default*** state for a Sink Port when:

- ***Hard Reset*** signaling is detected.

8.3.3.6.5 Policy Engine in Source to Sink Swap State Diagram

Figure 8-29 shows the additional states and transitions that shall be followed to swap from Source to Sink roles and the changes that shall be followed for error and Hard Reset handling.

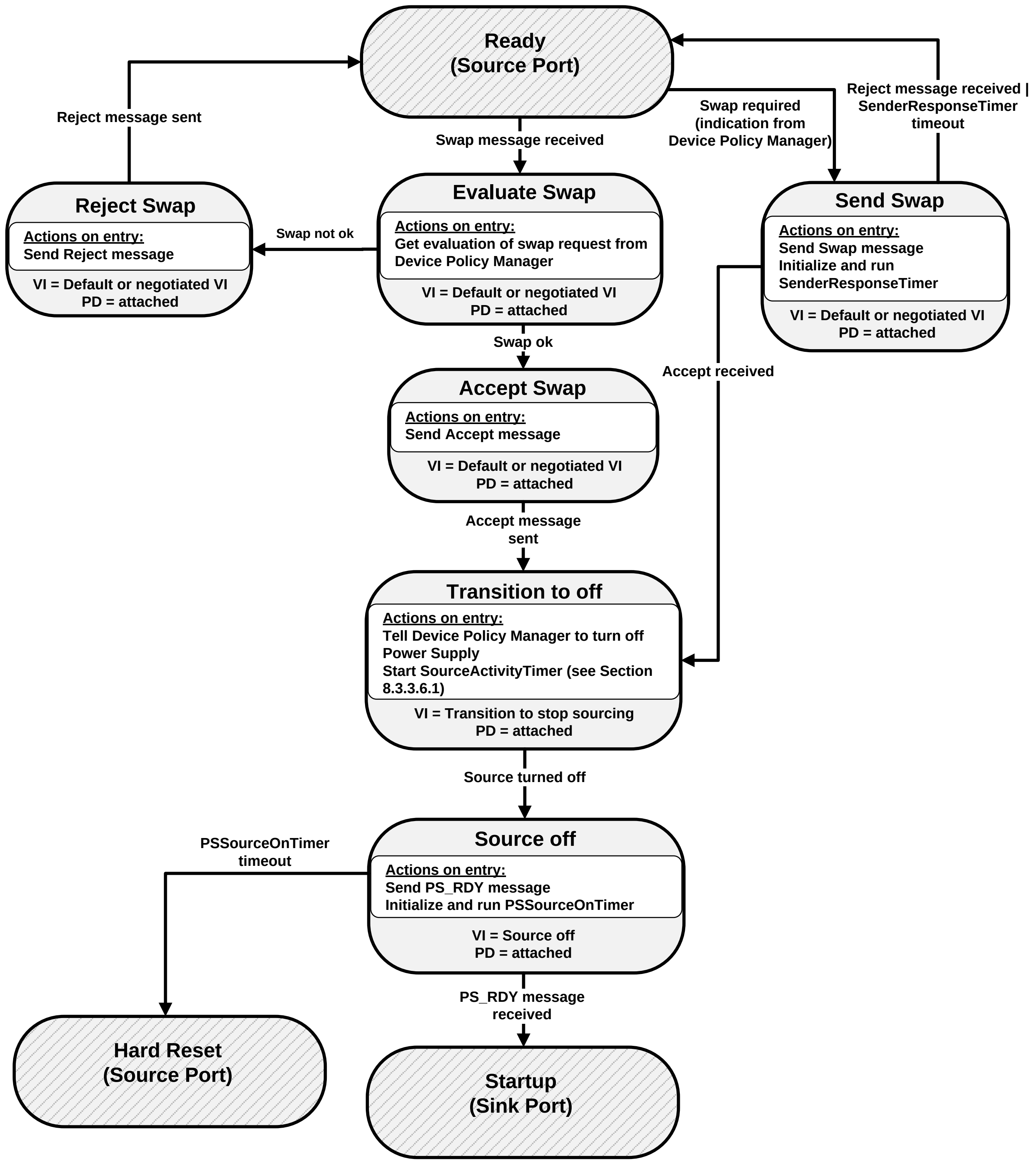


Figure 8-29: Dual-Role Port in a Provider/Consumer State Diagram

8.3.3.6.5.1 Ready

The Role Swap process shall start only from the ***Ready*** state where power is stable.

The Policy Engine shall transition to the ***Evaluate Swap*** state when:

- A ***Swap*** message is received.

The Policy Engine shall transition to the ***Send Swap*** state when:

- The Device Policy Manager indicates that a Role Swap is required.

8.3.3.6.5.2 Evaluate Swap

On entry to the ***Evaluate Swap*** state the Policy Engine shall ask the Device Policy Manager whether a Swap can be made.

The Policy Engine shall transition to the ***Accept Swap*** state when:

- The Device Policy Manager indicates that a Role Swap is ok.

The Policy Engine shall transition to the ***Reject Swap*** state when:

- The Device Policy Manager indicates that a Role Swap is not ok.

8.3.3.6.5.3 Accept Swap

On entry to the ***Accept Swap*** state the Policy Engine shall request the Protocol Layer to send an ***Accept*** message.

The Policy Engine shall transition to the ***Transition to off*** state when:

- The ***Accept*** message has been sent.

8.3.3.6.5.4 Transition to off

On entry to the ***Transition to off*** state the Policy Engine shall request the Device Policy Manager to turn off the Source and shall initialize and run the ***SourceActivityTimer*** (see Section 8.3.3.6.1.1 for use of ***Ping*** messaging for Dual-Role Ports which are initially Source Ports).

The Policy Engine shall transition to the ***Source off*** state when:

- The Device Policy Manager indicates that the Source has been turned off.

8.3.3.6.5.5 Source off

On entry to the ***Source off*** state the Policy Engine shall request the Protocol Layer to send a ***PS_RDY*** message and shall start the ***PSSourceOnTimer***.

On exit from the Source off state the Policy Engine shall stop the ***PSSourceOnTimer***.

The Policy Engine shall transition to the ***Transition to default*** state for a Sink Port when:

- A ***PS_RDY*** message is received indicating that the remote Source is now supplying power.

The Policy Engine shall transition to the ***Hard Reset*** state for a Source Port when:

- The ***PSSourceOnTimer*** times out.

8.3.3.6.5.6 Send Swap

On entry to the ***Send Swap*** state the Policy Engine shall request the Protocol Layer to send a ***Swap*** message and shall start the ***SenderResponseTimer***.

On exit from the ***Send Swap*** state the Policy Engine shall stop the ***SenderResponseTimer***.

The Policy Engine shall transition to the ***Ready*** state for a Source Port when:

- A ***Reject*** message is received.
- Or the ***SenderResponseTimer*** times out.

The Policy Engine shall transition to the ***Transition to off*** state when:

- An ***Accept*** message is received.

8.3.3.6.5.7 Reject Swap

On entry to the ***Reject Swap*** state the Policy Engine shall request the Protocol Layer to send a ***Reject*** message.

The Policy Engine shall transition to the ***Ready*** state for a Source Port when:

- The ***Reject*** message has been sent.

8.3.3.6.6 Policy Engine in Sink to Source swap State Diagram

Dual-Role Ports that combine Sink and Source capabilities shall comprise Sink and Source Policy Engine state machines. In addition they shall have the capability to do a Role Swap from the ***Ready*** state and shall return to default operation on a Hard Reset or non-Protocol error. Figure 8-30 shows the additional states and transitions that shall be followed to swap from Sink to Source roles and the changes that shall be followed for error and Hard Reset handling.

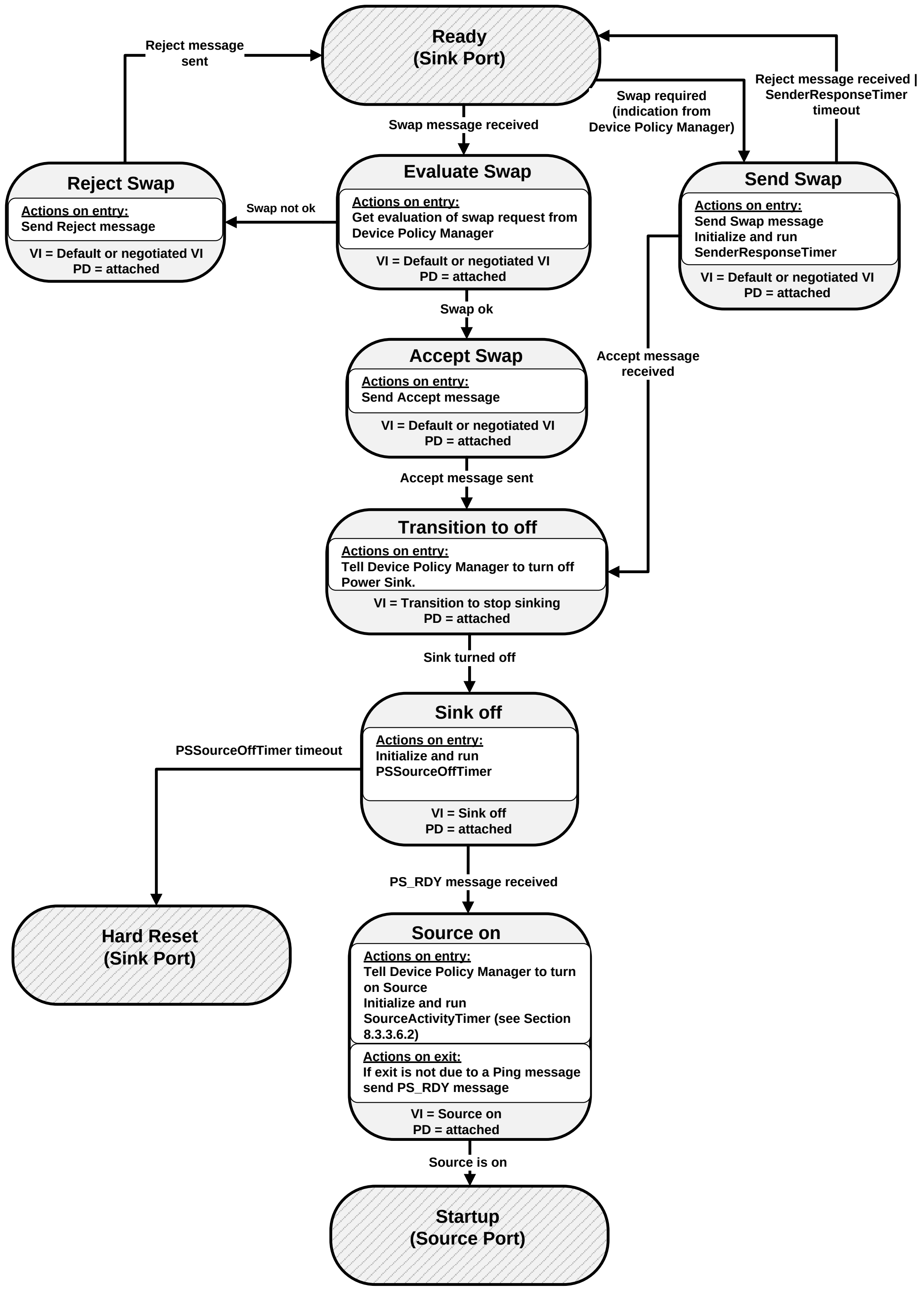


Figure 8-30: Dual-role port in a Consumer/Provider state diagram

8.3.3.6.6.1 Ready

The Swap process shall start only from the ***Ready*** state where power is stable.

The Policy Engine shall transition to the ***Evaluate Swap*** state when:

- A ***Swap*** message is received.

The Policy Engine shall transition to the ***Send Swap*** state when:

- The Device Policy Manager indicates that a Role Swap is required.

8.3.3.6.6.2 Evaluate Swap

On entry to the ***Evaluate Swap*** state the Policy Engine shall ask the Device Policy Manager whether a Role Swap can be made.

The Policy Engine shall transition to the ***Accept Swap*** state when:

- The Device Policy Manager indicates that a Role Swap is ok.

The Policy Engine shall transition to the ***Reject Swap*** state when:

- The Device Policy Manager indicates that a Role Swap is not ok.

8.3.3.6.6.3 Accept Swap

On entry to the ***Accept Swap*** state the Policy Engine shall request the Protocol Layer to send an ***Accept*** message.

The Policy Engine shall transition to the ***Transition to off*** state when:

- The ***Accept*** message has been sent.

8.3.3.6.6.4 Transition to off

On entry to the ***Transition to off*** state the Policy Engine shall request the Device Policy Manager to turn off the Sink.

The Policy Engine shall transition to the ***Sink off*** state when:

- The Device Policy Manager indicates that the Sink has been turned off.

8.3.3.6.6.5 Sink off

On entry to the ***Sink off*** state the Policy Engine shall initialize and run the ***PSSourceOffTimer***.

The Policy Engine shall transition to the ***Hard Reset*** state for a Sink Port when:

- The ***PSSourceOffTimer*** times out.

8.3.3.6.6.6 Source on

On entry to the ***Source on*** state the Policy Engine shall request the Device Policy Manager to turn on the Source and shall initialize and run the ***SourceActivityTimer*** (see Section 8.3.3.6.2.1 for details of ***Ping*** messaging for Dual-Role ports which are initially Sink Ports).

On exit from the ***Source on*** state (except if the exit is due to a ***SourceActivityTimer*** timeout) the Policy Engine shall send a ***PS_RDY*** message.

The Policy Engine shall transition to the ***Startup*** state for a Source Port when:

- The Source Port has been turned on.

8.3.3.6.6.7 Send Swap

On entry to the ***Send Swap*** state the Policy Engine shall request the Protocol Layer to send a ***Swap*** message and shall initialize and run the ***SenderResponseTimer***.

The Policy Engine shall transition to the ***Ready*** state for a Sink Port when:

- A ***Reject*** message is received.
- Or the ***SenderResponseTimer*** times out.

The Policy Engine shall transition to the ***Transition to off*** state when:

- An ***Accept*** message is received.

8.3.3.6.6.8 Reject Swap

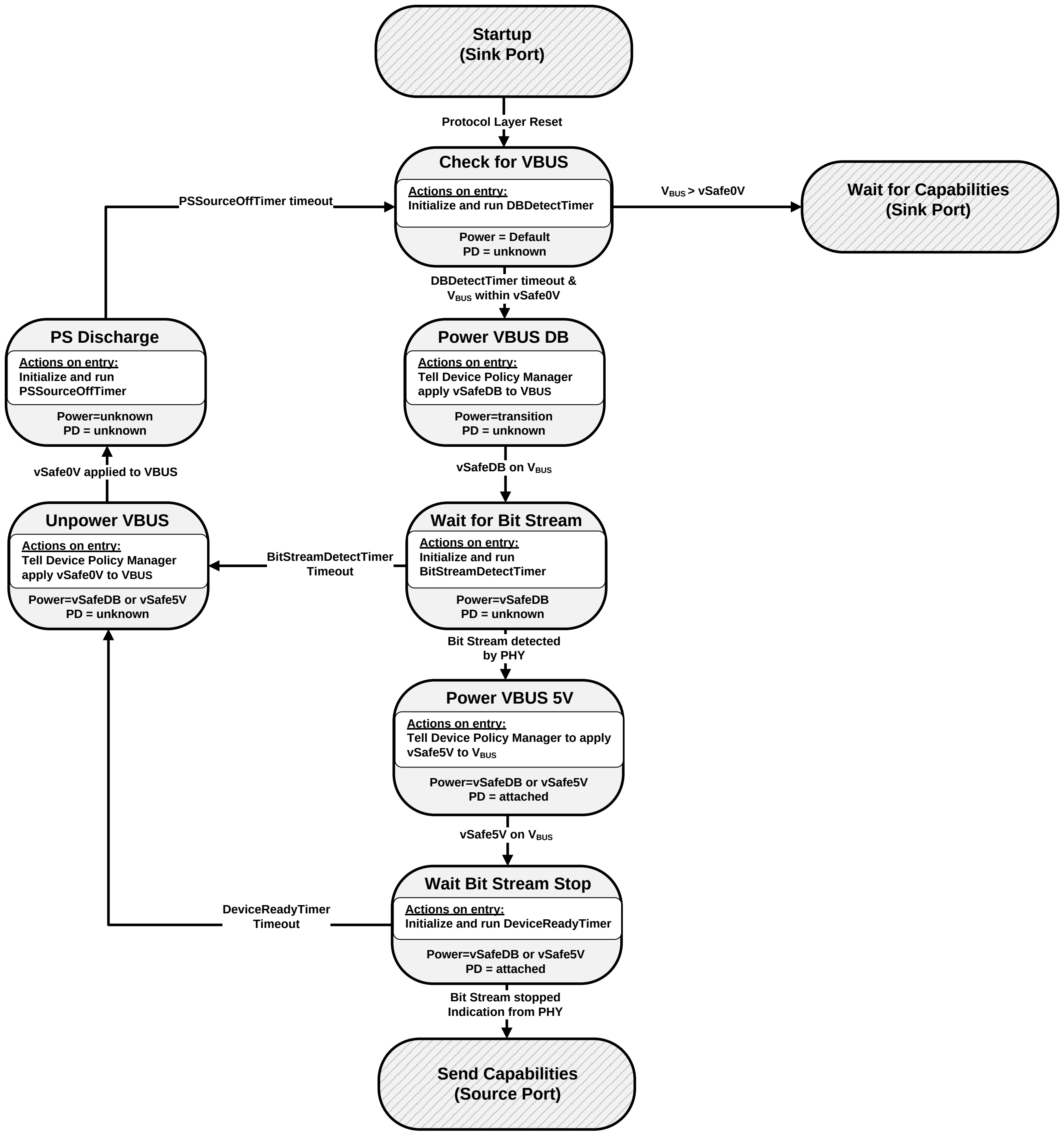
On entry to the ***Reject Swap*** state the Policy Engine shall request the Protocol Layer to send a ***Reject*** message.

The Policy Engine shall transition to the ***Ready*** state for a Sink Port when:

- The ***Reject*** message has been sent.

8.3.3.6.7 Consumer/Provider Dead Battery/Power Loss State Diagram

Figure 8-31 shows the additional state behavior that shall be followed for a Consumer/Provider to handle dead battery detection. After the Consumer/Provider Policy Engine has transitioned to the ***Send Capabilities*** state for a Source Port, its subsequent state operation shall conform to that of a Consumer/Provider which has completed a Role Swap (see Section 8.3.3.6.6). The Consumer/Provider has effectively undergone a Role Swap without the requirement of protocol negotiation.



Startup (Sink Port)
Protocol Layer Reset
Check for VBUS
Actions on entry: Initialize and run DBDetectTimer
Power = Default
PD = unknown
V_BUS > vSafe0V
Wait for Capabilities (Sink Port)
PSSourceOffTimer timeout
DBDetectTimer timeout & V_BUS within vSafe0V
PS Discharge
Actions on entry: Initialize and run PSSourceOffTimer
Power=unknown
PD = unknown
Power VBUS DB
Actions on entry: Tell Device Policy Manager apply vSafeDB to VBUS
Power=transition
PD = unknown
vSafe0V applied to VBUS
vSafeDB on V_BUS
Unpower VBUS
Actions on entry: Tell Device Policy Manager apply vSafe0V to VBUS
Power=vSafeDB or vSafe5V
PD = unknown
BitStreamDetectTimer Timeout
Wait for Bit Stream
Actions on entry: Initialize and run BitStreamDetectTimer
Power=vSafeDB
PD = unknown
Bit Stream detected by PHY
Power VBUS 5V
Actions on entry: Tell Device Policy Manager to apply vSafe5V to V_BUS
Power=vSafeDB or vSafe5V
PD = attached
vSafe5V on V_BUS
Wait Bit Stream Stop
Actions on entry: Initialize and run DeviceReadyTimer
Power=vSafeDB or vSafe5V
PD = attached
DeviceReadyTimer Timeout
Bit Stream stopped Indication from PHY
Send Capabilities (Source Port)

Figure 8-31 Consumer/Provider Dead Battery/Power Loss State Diagram

8.3.3.6.7.1 Check for V_{BUS}

The Policy Engine for a Consumer/Provider shall initially start in the ***Startup*** state for a Sink Port. Once the Protocol Layer has been reset it shall transition to the ***Check for*** V_{BUS} state.

On entry to the ***Check for*** V_{BUS} state the Policy Engine shall initialize and run the ***DBDetectTimer***.

The Policy Engine shall transition to the ***Wait For Capabilities*** state when:

- V_{BUS} is greater than ***vSafe0V***.

The Policy Engine shall transition to the ***Power*** V_{BUS} state when:

- The ***DBDetectTimer*** has timed out and
- V_{BUS} is within ***vSafe0V***.

8.3.3.6.7.2 Power V_{BUS} DB

On entry to the ***Power*** V_{BUS} ***DB*** state the Policy Engine shall tell the Device Policy Manager to apply ***vSafeDB*** to V_{BUS}.

The Policy Engine shall transition to the ***Wait For Bit Stream*** state when:

- ***vSafeDB*** is on V_{BUS}.

8.3.3.6.7.3 Wait For Bit Stream

On entry to the Wait For Bit Stream state the Policy Engine shall initialize and run the ***BitStreamDetectTimer***.

The Policy Engine shall transition to the *Power* V_{BUS} *5V* state when:

- The PHY Layer indicates that Bit Stream signaling has been received.

The Policy Engine shall transition to the ***Unpower*** V_{BUS} state when:

- The ***BitStreamDetectTimer*** times out.

8.3.3.6.7.4 Power V_{BUS} 5V

On entry to the ***Power*** V_{BUS} ***5V*** state the Policy Engine shall request the Device Policy Manager to apply ***vSafe5V*** to V_{BUS}.

The Policy Engine shall transition to the ***Wait Bit Stream Stop*** state when:

- ***vSafe5V*** is present on V_{BUS}.

8.3.3.6.7.5 Wait Bit Stream Stop

On entry to the Wait Bit Stream Stop state the Policy Engine shall initialize and run the ***DeviceReadyTimer***.

The Policy Engine shall transition to the Send Capabilities state for a Source Port when:

- An indication is received from the PHY that the Bit Stream has stopped.

The Policy Engine shall transition to the ***Unpower*** V_{BUS} state when:

- The ***DeviceReadyTimer*** times out.

8.3.3.6.7.6 Unpower V_{BUS}

On entry to the ***Unpower*** V_{BUS} state the Policy Engine shall tell the Device Policy Manager to apply ***vSafe0V*** to V_{BUS}.

The Policy Engine shall transition to the ***PS Discharge*** state when:

- ***vSafe0V*** has been applied to V_{BUS}.

Note: the intention of applying ***vSafe0V*** is that the Consumer/Provider will utilize the same mechanism to unpower V_{BUS} as it uses to remove V_{BUS} power during a role swap.

8.3.3.6.7.7 PS Discharge

On entry to the PS Discharge state the Policy Engine shall initialize and run the ***PSSourceOffTimer***.

The Policy Engine shall transition to the ***Check for V_{BUS}*** state when:

- The ***PSSourceOffTimer*** times out.

Note: the ***PSSourceOffTimer*** is used to ensure that the Consumer/Provider is not powering V_{BUS} when it proceeds to check the voltage on V_{BUS}. This assumes that the discharge of V_{BUS} follows the same process as when removing power during a role swap.

8.3.3.6.8 Provider/Consumer Dead Battery/Power Loss State Diagram

Figure 8-32 shows the additional state behavior that shall be followed for a Provider/Consumer to handle dead battery detection. The Provider/Consumer is assumed to startup in a state where it is either powered off or is unable to power its port (e.g. due to a dead battery). If the Provider/Consumer is powered on and has sufficient power to power its port it should start up as a Source Port.

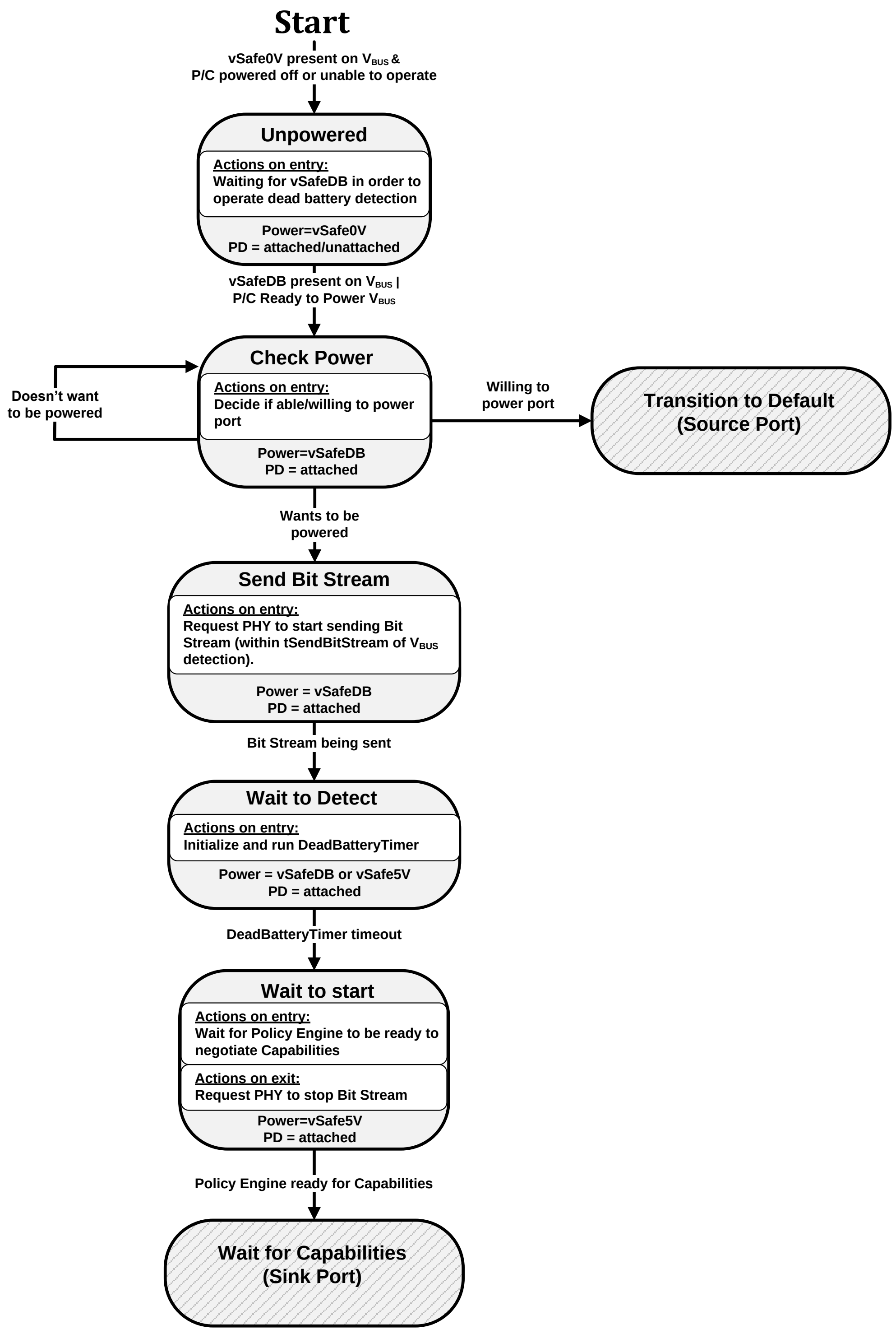

Start
vSafe0V present on V_{BUS} & P/C powered off or unable to operate
Unpowered
Actions on entry:
Waiting for vSafeDB in order to operate dead battery detection
Power=vSafe0V
PD = attached/unattached
vSafeDB present on V_{BUS} | P/C Ready to Power V_{BUS}
Check Power
Actions on entry:
Decide if able/willing to power port
Power=vSafeDB
PD = attached
Doesn't want to be powered
Willing to power port
Transition to Default (Source Port)
Wants to be powered
Send Bit Stream
Actions on entry:
Request PHY to start sending Bit Stream (within tSendBitStream of V_{BUS} detection).
Power = vSafeDB
PD = attached
Bit Stream being sent
Wait to Detect
Actions on entry:
Initialize and run DeadBatteryTimer
Power = vSafeDB or vSafe5V
PD = attached
DeadBatteryTimer timeout
Wait to start
Actions on entry:
Wait for Policy Engine to be ready to negotiate Capabilities
Actions on exit:
Request PHY to stop Bit Stream
Power=vSafe5V
PD = attached
Policy Engine ready for Capabilities
Wait for Capabilities (Sink Port)

Figure 8-32 Provider/Consumer Dead Battery/Power Loss State Diagram

8.3.3.6.8.1 Unpowered

The ***Unpowered*** state is the startup state for a Provider/Consumer at power up when either there is no power to the Provider/Consumer (in this case there may be no physical "state" as such) or when the Provider/Consumer has some sort of power supply but is inactive.

The ***Unpowered*** state shall be entered from any state when:

- V_{BUS} is within ***vSafe0V*** and
- The Provider/Consumer is powered off or has insufficient power to operate.

On entry to the ***Unpowered*** state the Policy Engine shall wait for ***vSafeDB*** to appear on V_{BUS} in order to start the dead battery detection process.

The Policy Engine shall transition to the ***Check Power*** state when:

- ***vSafeDB*** is present on V_{BUS},.or
- The Provider/Consumer is ready to power V_{BUS}

8.3.3.6.8.2 Check Power

On entry to the ***Check Power*** state the Policy Engine shall decide whether it is able and willing to supply power to the port.

The Policy Engine shall transition to the ***Transition to Default*** state for a Source port when:

- It is willing to power V_{BUS}.

The Policy Engine shall transition to the ***Send Preamble*** state when:

- It wants to be powered by the Consumer/Provider.

The Policy Engine shall stay in the ***Check Power*** state when:

- The Provider/Consumer does not want to either power the port or be powered.

8.3.3.6.8.3 Send Bit Stream

On entry to the ***Send Bit Stream*** state the Policy Engine shall request the PHY Layer to start sending the Bit Stream (see Section 4.2).

The Policy Engine shall transition to the ***Wait to Detect*** state when:

- The Bit Stream is being sent.

8.3.3.6.8.4 Wait to Detect

On entry to the ***Wait to Detect*** state the Policy Engine shall initialize and run the DeadBatteryTimer to allow the Consumer/Provider time to detect the Bit Stream and apply power to V_{BUS}.

The Policy Engine shall transition to the ***Wait to Start*** state when:

- The DeadBatteryTimer times out.

8.3.3.6.8.5 Wait to Start

On entry to the ***Wait to Start*** state the Policy Engine shall wait until it is ready to negotiate Capabilities.

On exit from the ***Wait to Start*** state the Policy Engine shall request the PHY to stop sending the Bit Stream.

The Policy Engine shall transition to the ***Wait for Capabilities*** state for a Sink Port when:

- The Policy Engine is ready to negotiate Capabilities.

8.3.3.7 BIST Receive Mode State Diagram

Figure 8-33 shows the state behavior that shall be followed by a UUT, which can be either a Source or Sink, when operating in BIST Receive Mode. Transitions to ***BIST Receive Mode*** state shall be from the ***Ready*** state for either Source or Sink operation.

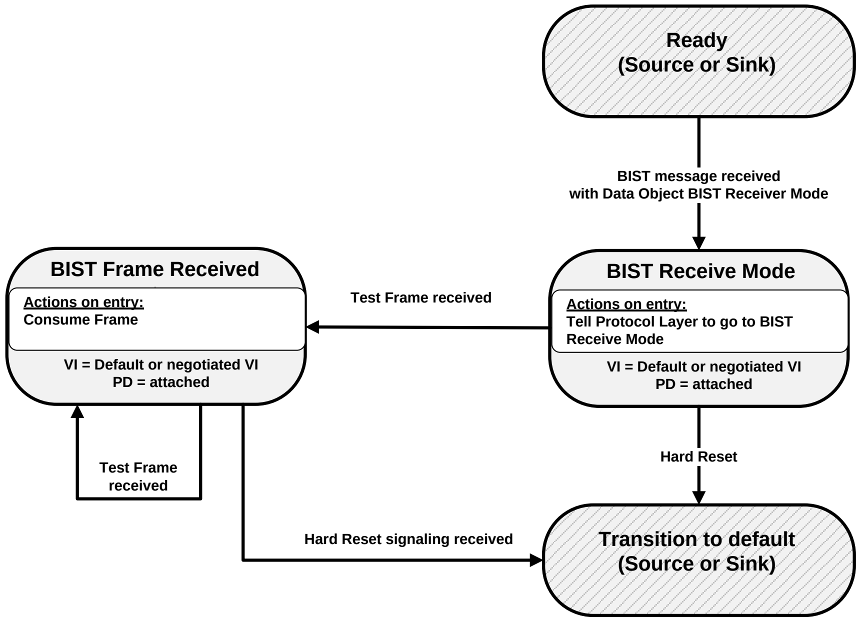

Ready (Source or Sink)
BIST message received with Data Object BIST Receiver Mode
BIST Frame Received
Actions on entry: Consume Frame
VI = Default or negotiated VI PD = attached
Test Frame received
BIST Receive Mode
Actions on entry: Tell Protocol Layer to go to BIST Receive Mode
VI = Default or negotiated VI PD = attached
Test Frame received
Hard Reset
Hard Reset signaling received
Transition to default (Source or Sink)

Figure 8-33 BIST Receive Mode State Diagram

8.3.3.7.1 BIST Receive Mode

The Source or Sink shall enter the ***BIST Receive Mode*** state from the ***Ready*** state when a ***BIST*** message is received with a ***BIST Receiver Mode*** data object.

On entry to the ***BIST Receive Mode*** state the Policy Engine shall tell the Protocol Layer to go to BIST Receive Mode.

The Policy Engine shall transition to the ***BIST Frame Received*** state when:

- A test frame is received.

The Policy Engine shall transition to the ***Transition to default*** state for Source or Sink (as appropriate) when:

- ***Hard Reset*** signaling is received.

8.3.3.7.2 BIST Frame Received

On entry to the ***BIST Frame Received*** state the Policy Engine shall consume a BIST Transmit Test Frame if one has been received.

The Policy Engine shall transition back to the ***BIST Send Frame*** state when:

- The BIST Receive Test Frame has been received.

The Policy Engine shall transition to the ***Transition to default*** state for Source or Sink (as appropriate) when:

- ***Hard Reset*** signaling is received.

8.3.3.8 BIST Transmit Mode State Diagram

Figure 8-34 shows the state behavior that shall be followed by a UUT that can be either a Source or Sink, when operating in BIST Transmit Mode. Transitions to ***BIST Transmit Mode*** state shall be from the ***Ready*** state for either Source or Sink operation.

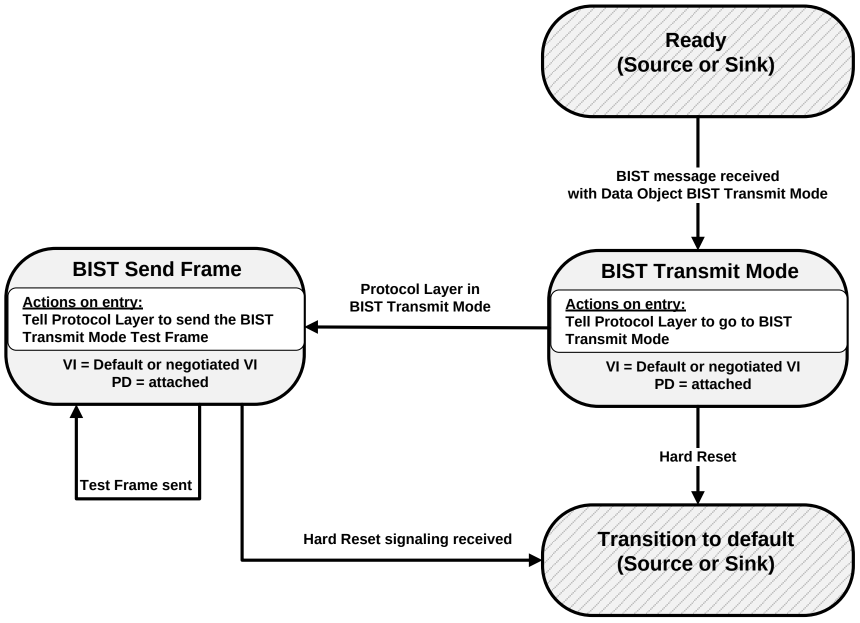


Figure 8-34 BIST Transmit Mode State Diagram

8.3.3.8.1 BIST Transmit Mode

The Source or Sink shall enter the ***BIST Transmit Mode*** state from the ***Ready*** state when a ***BIST*** message is received with a ***BIST Transmit Mode*** data object.

On entry to the ***BIST Transmit Mode*** state the Policy Engine shall tell the Protocol Layer to go to BIST Transmit Mode.

The Policy Engine shall transition to the ***BIST Send Frame*** state when:

- The Protocol Layer is in BIST Transmit Mode.

The Policy Engine shall transition to the ***Transition to default*** state for Source or Sink (as appropriate) when:

- ***Hard Reset*** signaling is received.

8.3.3.8.2 BIST Send Frame

On entry to the ***BIST Send Frame*** state the Policy Engine shall tell the Protocol Layer to send the next BIST Transmit Test Frame in the PRBS sequence.

The Policy Engine shall transition back to the ***BIST Send Frame*** state when:

- The BIST Transmit Test Frame has been sent.

The Policy Engine shall transition to the ***Transition to default*** state for Source or Sink (as appropriate) when:

- ***Hard Reset*** signaling is received.

8.3.3.9 BIST Carrier Purity and Eye Pattern State Diagram

Figure 8-35 shows the state behavior that shall be followed by a UUT, which can be either a Source or Sink, when operating in BIST Eye Pattern, BIST Carrier Mode 0, BIST Carrier Mode 1 or BIST Carrier Mode 2. Transitions to any of the Test Mode states shall be from the ***Ready*** state for either Source or Sink operation.

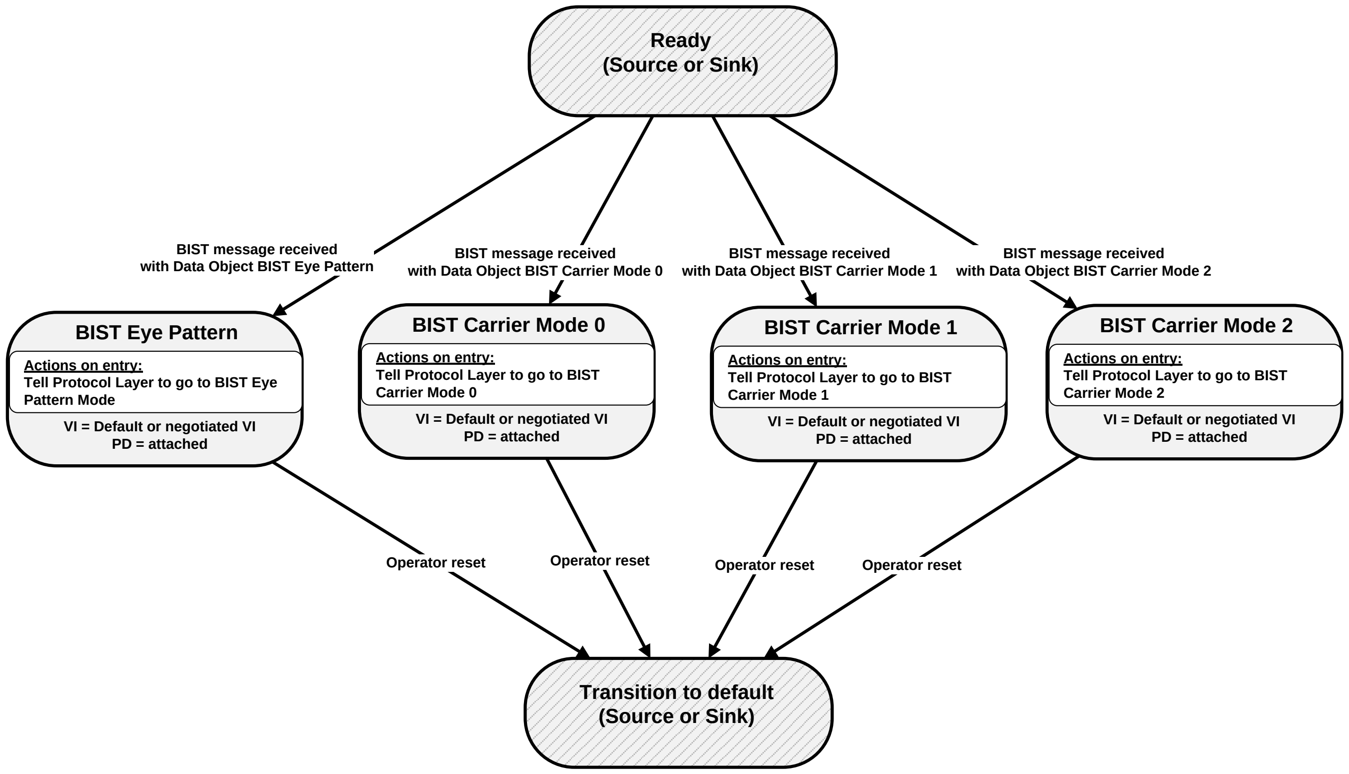

Ready (Source or Sink)
BIST message received with Data Object BIST Eye Pattern
BIST message received with Data Object BIST Carrier Mode 0
BIST message received with Data Object BIST Carrier Mode 1
BIST message received with Data Object BIST Carrier Mode 2
BIST Eye Pattern
Actions on entry:
Tell Protocol Layer to go to BIST Eye Pattern Mode
VI = Default or negotiated VI
PD = attached
BIST Carrier Mode 0
Actions on entry:
Tell Protocol Layer to go to BIST Carrier Mode 0
VI = Default or negotiated VI
PD = attached
BIST Carrier Mode 1
Actions on entry:
Tell Protocol Layer to go to BIST Carrier Mode 1
VI = Default or negotiated VI
PD = attached
BIST Carrier Mode 2
Actions on entry:
Tell Protocol Layer to go to BIST Carrier Mode 2
VI = Default or negotiated VI
PD = attached
Operator reset
Operator reset
Operator reset
Operator reset
Transition to default (Source or Sink)

Figure 8-35 BIST Carrier Purity and Eye Pattern State Diagram

8.3.3.9.1 BIST Eye Pattern

The Source or Sink shall enter the ***BIST Eye Pattern Mode*** state from the ***Ready*** state when a ***BIST*** message is received with a ***BIST Eye Pattern*** data object.

On entry to the ***BIST Eye Pattern*** state the Policy Engine shall tell the Protocol Layer to go to BIST Eye Pattern Test Mode.

The Policy Engine shall transition to the ***Transition to default*** state for Source or Sink (as appropriate) when:

- There is an operator reset of the UUT.

8.3.3.9.2 BIST Carrier Mode 0

The Source or Sink shall enter the ***BIST Carrier Mode 0*** state from the ***Ready*** state when a ***BIST*** message is received with a ***BIST Carrier Mode 0*** data object.

On entry to the ***BIST Carrier Mode 0*** state the Policy Engine shall tell the Protocol Layer to go to BIST Carrier Purity Mode 0.

The Policy Engine shall transition to the ***Transition to default*** state for Source or Sink (as appropriate) when:

- There is an operator reset of the UUT.

8.3.3.9.3 BIST Carrier Mode 1

The Source or Sink shall enter the ***BIST Carrier Mode 1*** state from the ***Ready*** state when a ***BIST*** message is received with a ***BIST Carrier Mode 1*** data object.

On entry to the ***BIST Carrier Mode 1*** state the Policy Engine shall tell the Protocol Layer to go to BIST Carrier Purity Mode 1.

The Policy Engine shall transition to the ***Transition to default*** state for Source or Sink (as appropriate) when:

- There is an operator reset of the UUT.

8.3.3.9.4 BIST Carrier Mode 2

The Source or Sink shall enter the ***BIST Carrier Mode 2*** state from the ***Ready*** state when a ***BIST*** message is received with a ***BIST Carrier Mode 2*** data object.

On entry to the ***BIST Carrier Mode 2*** state the Policy Engine shall tell the Protocol Layer to go to BIST Carrier Purity Mode 2.

The Policy Engine shall transition to the ***Transition to default*** state for Source or Sink (as appropriate) when:

- There is an operator reset of the UUT.

9. System Policy

9.1 Overview

This chapter describes the requirements of the USB Power Delivery Specification's System Policy and Status mechanisms for devices with data connections (e.g. D+/D- and or SSTx+/- and SSRx+/-). The Policies themselves are not described here; these are left to the implementers of the relevant products and systems to define. The aim of these mechanisms is to support a System USB Power Delivery Policy Manager (SPM) which:

- Provides feedback of system power status to the end user as and when appropriate
- Provides co-ordination of system power resources based on a system wide policy when enabled
- Exposes relevant policy settings to the end user
- Allows a device to report its capabilities in relation to the power it draws

All PD Capable USB (PDUSB) Devices shall report themselves as self-powered devices (over USB) when plugged into a PD capable port even if they are entirely powered from $V_{BUS.}$ However, there are some differences between PD and ***[USB2.0]*** / ***[USB3.0]***, for example, the presence of V_{BUS} alone does not mean that the device (Consumer) moves from the USB Attached state to the USB Powered state. Similarly the removal of V_{BUS} alone does not move the device (Consumer) from any of the USB states to the Attached state. See Section 9.1.1 for details.

PDUSB Devices shall follow the PD requirements when it comes to suspend (see Section 6.4.1.3.2), configured, and operational power. The PD requirements when the device is configured, or operational are defined in this section (see Table 9-3). Note that the power requirements reported in the PD Consumer Port descriptor of the device shall override the power draw reported in the *bMaxPower* field in the configuration descriptor. A PDUSB Device shall report zero in the *bMaxPower* field after successfully negotiating a mutually agreeable power contract and shall disconnect and re-enumerate when it switches operation back to operating in standard ***[USB2.0]***, ***[USB3.0]*** or ***[BC1.2]***. When operating in ***[USB2.0]***, ***[USB3.0]*** or ***[BC1.2]*** mode it shall report its power draw via the *bMaxPower* field.

As shown in Figure 9-1, each Provider and Consumer will have their own Local Policies which operate between directly connected ports. An example of a typical PD system is shown in Figure 9-1. This example consists of a Provider, Consumer/Providers and Consumers connected together in a tree topology. Between directly connected devices there is both a flow of Power and also Communication consisting of both Status and Control information.

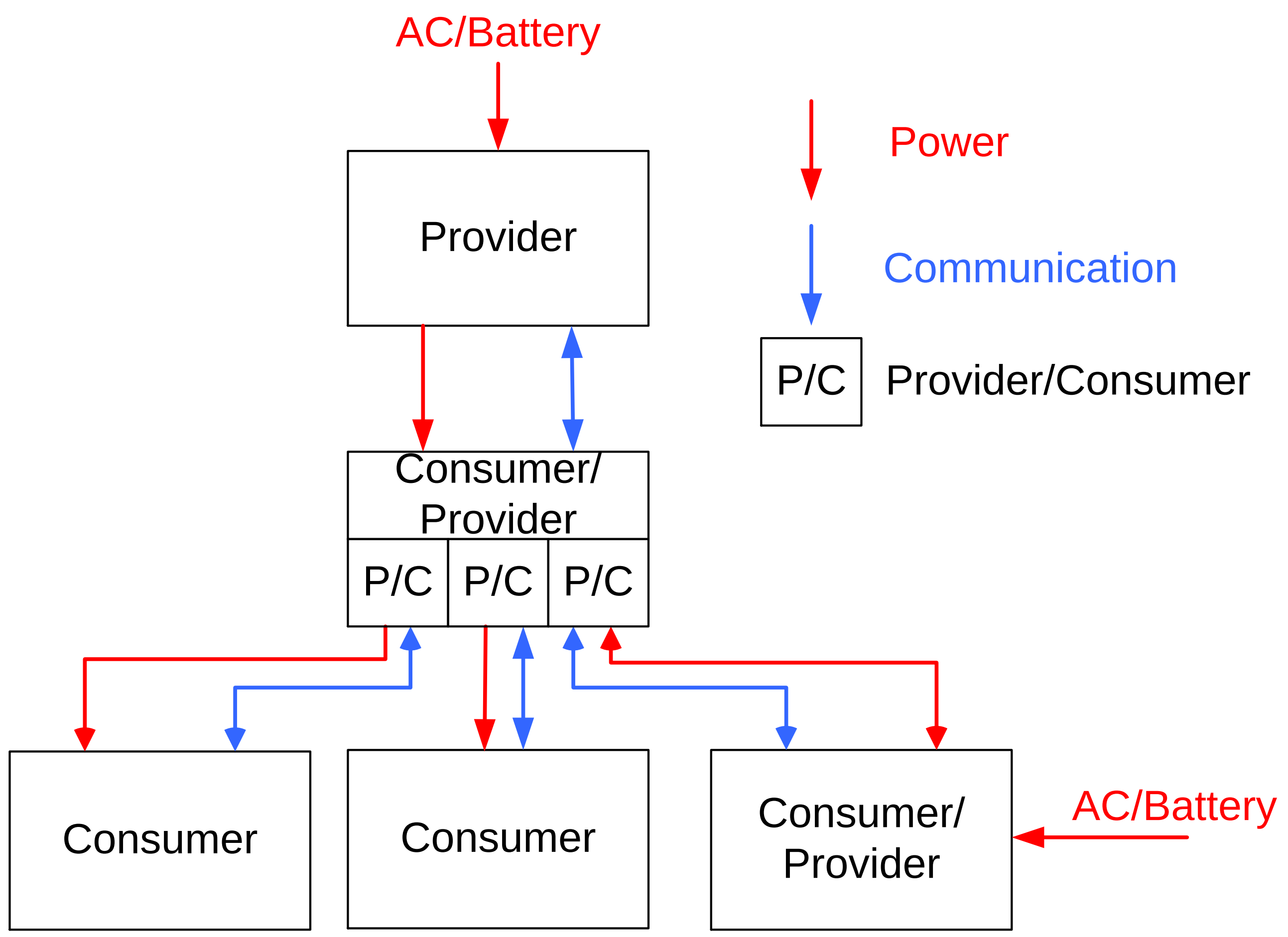


Figure 9-1 Example PD Topology

Figure 9-2 shows how this same topology can be mapped to USB. In a USB based system, policy is managed by the host and communication of system level policy information is via standard USB data line communication. This is a separate mechanism to the USB Power Delivery V_{BUS} protocol which is used to manage Local Policy. When USB data line communication is used, status information and control requests are passed directly between the System Policy Manager (SPM) on the host and the Provider or Consumer.

Status information comes from a Provider or Consumer to the SPM so it can better manage the resources on the host and provide feedback to the end user. Control information comes from the SPM to the Provider or Consumer allowing the SPM to manage resources at a system level and expose relevant settings to the end user.

Real systems will be a mixture of devices which in terms of power management support may have implemented PD, Battery Charging, Standard USB 2.0/3.0 or they may even just be non-compliant Power Sucking Devices. The level of communication of system status to the SPM will therefore not necessarily be comprehensive. The aim of the System Policies and Status mechanisms is to provide a mechanism whereby each connected entity in the system provides as much information as possible on the status of itself and, in the case of hubs, its downstream ports. This will enable the PM to build up as comprehensive a picture of the system as possible.

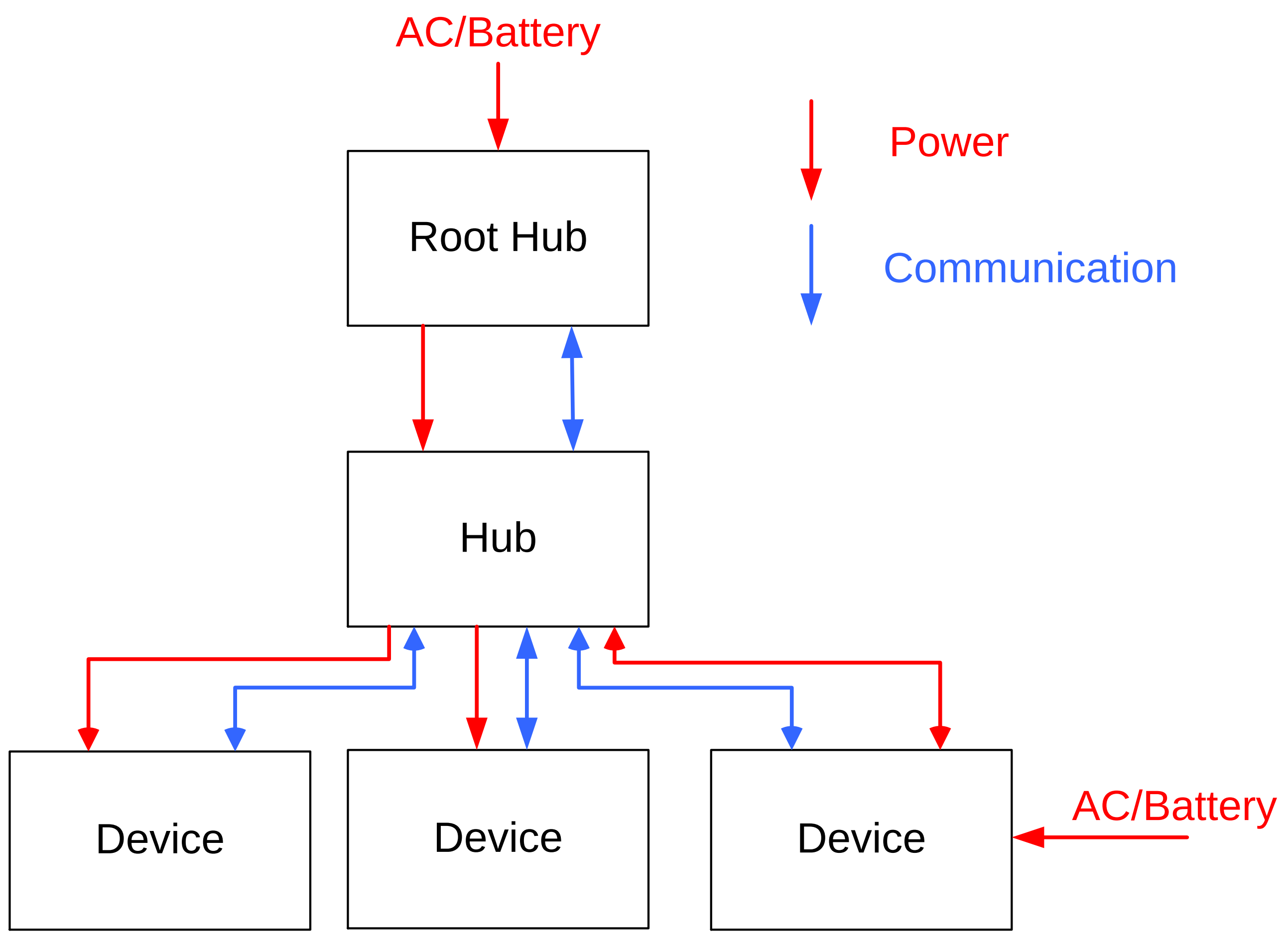


Figure 9-2 Mapping of PD Topology to USB

Information which will be communicated to the SPM is as follows:

- Versions of PD and BC supported
- Capabilities of each Provider/Consumer including a per port summary for Providers
- Current operational state of each Port e.g. Standard, BC, PD and negotiated power level
- Status of AC or Battery Power for each PDUSB Device in the system

The SPM can negotiate with Providers or Consumers in the system in order to request a different Local Policy, or to request the amount of power to be delivered by the Provider to the Consumer. Any change in Local Policy could trigger a renegotiation of the USB Power Delivery contract, using USB Power Delivery protocols, between a directly connected Provider and Consumer. A change in how much power is to be delivered will, for example, cause a renegotiation.

All PDUSB Devices with data lines shall return all relevant descriptors mentioned in this chapter. PDUSB Hubs shall also support a PD notification capability as defined in this chapter. In addition, a PDUSB Hubs shall support the capability to return the PD specific capabilities of a device that is attached to any of its downstream ports. Additionally, a PDUSB Hubs may reject System Policy manager requests, but should make a reasonable effort to comply if it is able to do so. Note that in a USB Hub, the notifications are sent over the USB Hub notification endpoint.

9.1.1 Mapping to USB Device States

As mentioned in Section 9.1 a PDUSB Device reports itself as a self-powered device. However, it shall determine whether or not it is in the USB Attached or USB Powered states as described in the following state machines. All other USB states of the PDUSB Device shall be as described in Chapter 9 of *[USB2.0]* and *[USB3.0]*.

Figure 9-3 shows how a PDUSB Device determines when to transition from the USB Attached to the USB Powered state.

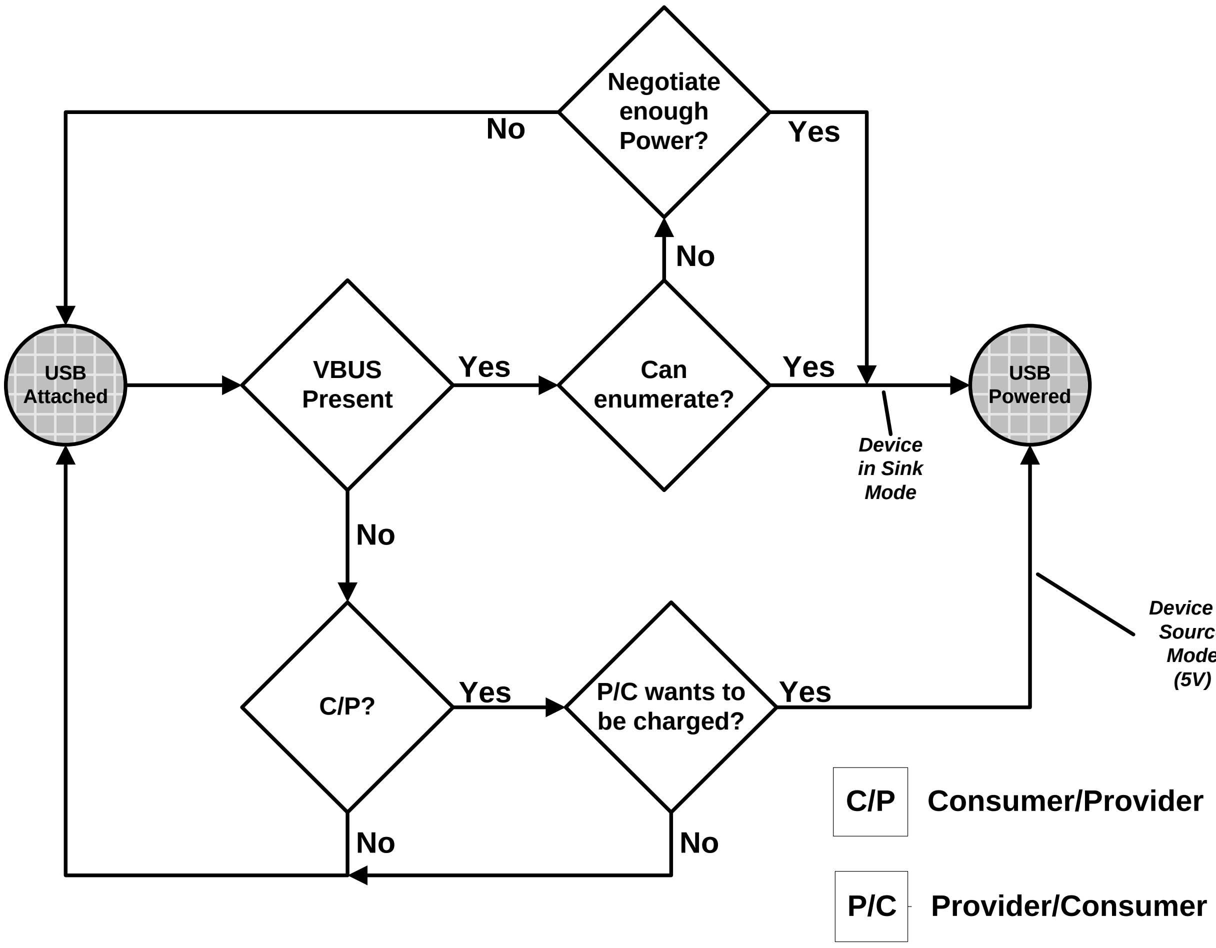


Figure 9-3 USB Attached to USB Powered State Transition

Figure 9-4 shows how a PDUSB Device determines when to transition from the USB Powered state to the USB Attached state when the device is a Consumer. A PDUSB Device determines that is swapping roles as described in Sections 8.3.2.4 and 8.3.2.5.

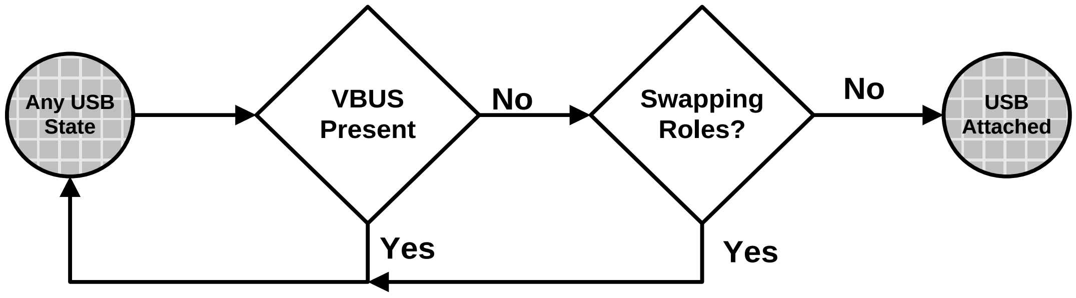


Figure 9-4 Any USB State to USB Attached State Transition (When operating as a Consumer)

Figure 9-5 shows how a PDUSB Device determines when to transition from the USB Powered state to the USB Attached state when the device is a Provider.

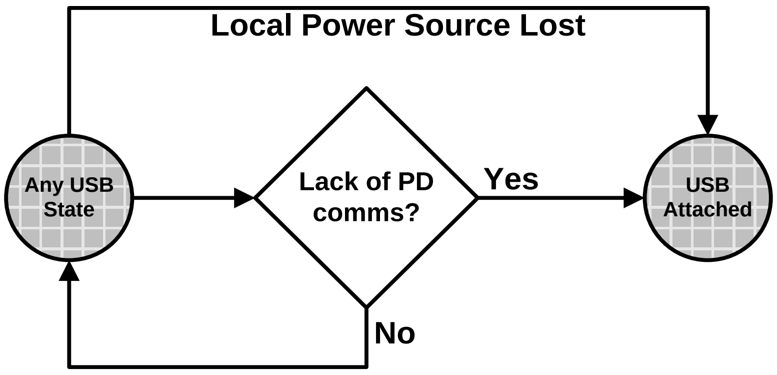


Figure 9-5 Any USB State to USB Attached State Transition (When operating as a Provider)

9.1.2 PD Software Stack

Figure 9-6 gives an example of the software stack on a PD aware OS. In this stack we are using the example of a system with an xHCI based controller. The USB Power Delivery hardware may or may not be a part of the xHC. The software that controls the USB Power Delivery hardware is called the System Policy Manager (SPM) in this specification.

It is expected that the SPM will expose an interface that can be used by an updated hub driver and updated client drivers that wish to interact with PDUSB Hubs and Peripheral Devices respectively.

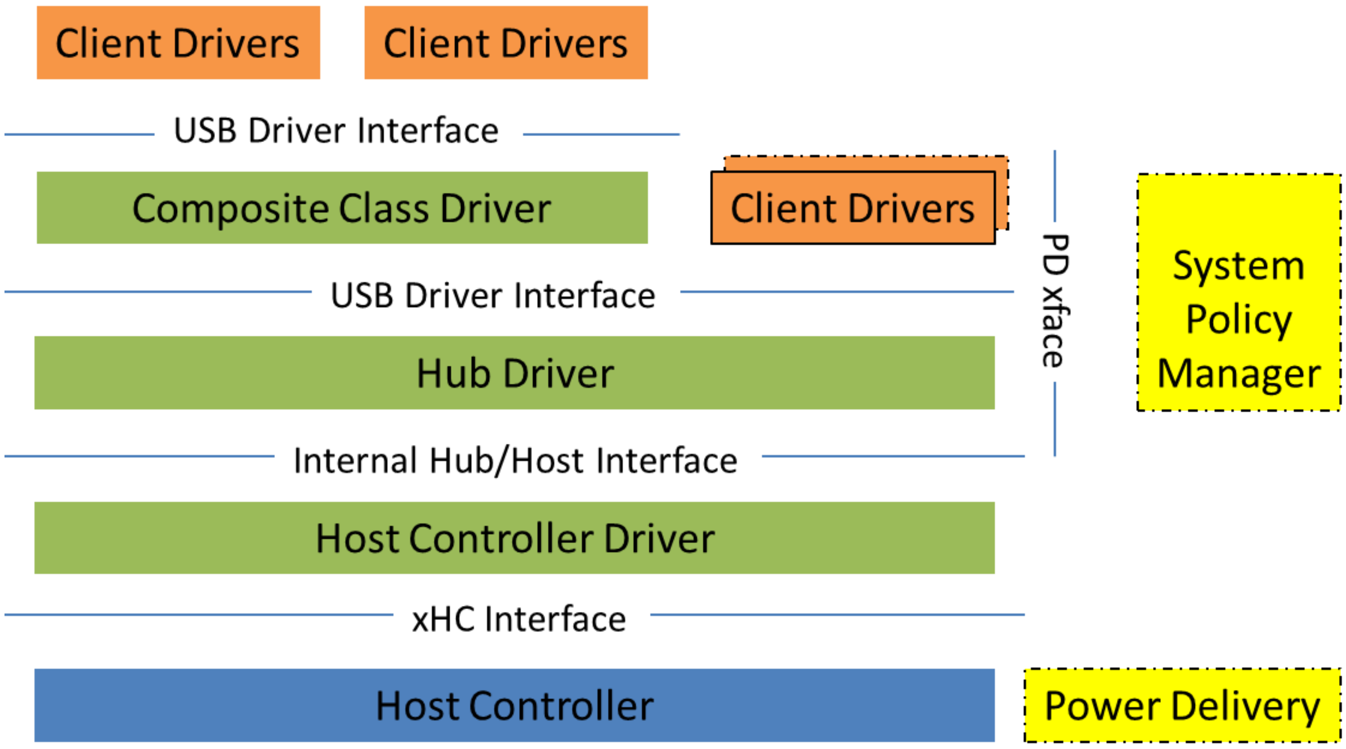


Figure 9-6 Software stack on a PD aware OS

9.1.3 PDUSB Device Enumeration

As described earlier, a PDUSB Device acts as a self-powered device with some caveats with respect to how it transitions from the USB Attached state to USB Powered state. Figure 9-7 gives a high level overview of the enumeration steps involved due to this change. A PDUSB Device will first (Step1) interact with the Power Delivery hardware and the Local Policy manager to determine whether or not it can get sufficient power to enumerate/operate. The SPM will be notified (Step 2) of the result of this negotiation between the Power Delivery hardware and the PDUSB Device. It may request changes to the Local Policy manager if it deems it necessary. After successfully negotiating a mutually agreeable power contract the device will signal a connect to the xHC. The standard USB enumeration process (Steps 3, 4 and 5) is then followed to load the appropriate driver for the function(s) that the PDUSB Device exposes.

It is the responsibility of the SPM to ensure that the total amount of power supplied to a PDUSB Device is sufficient to run as many of the functions exposed by the device as software needs to operate at the same time.

Note that the interfaces between the SPM and USB Power Delivery blocks in Figure 9-6 are outside scope of this specification.

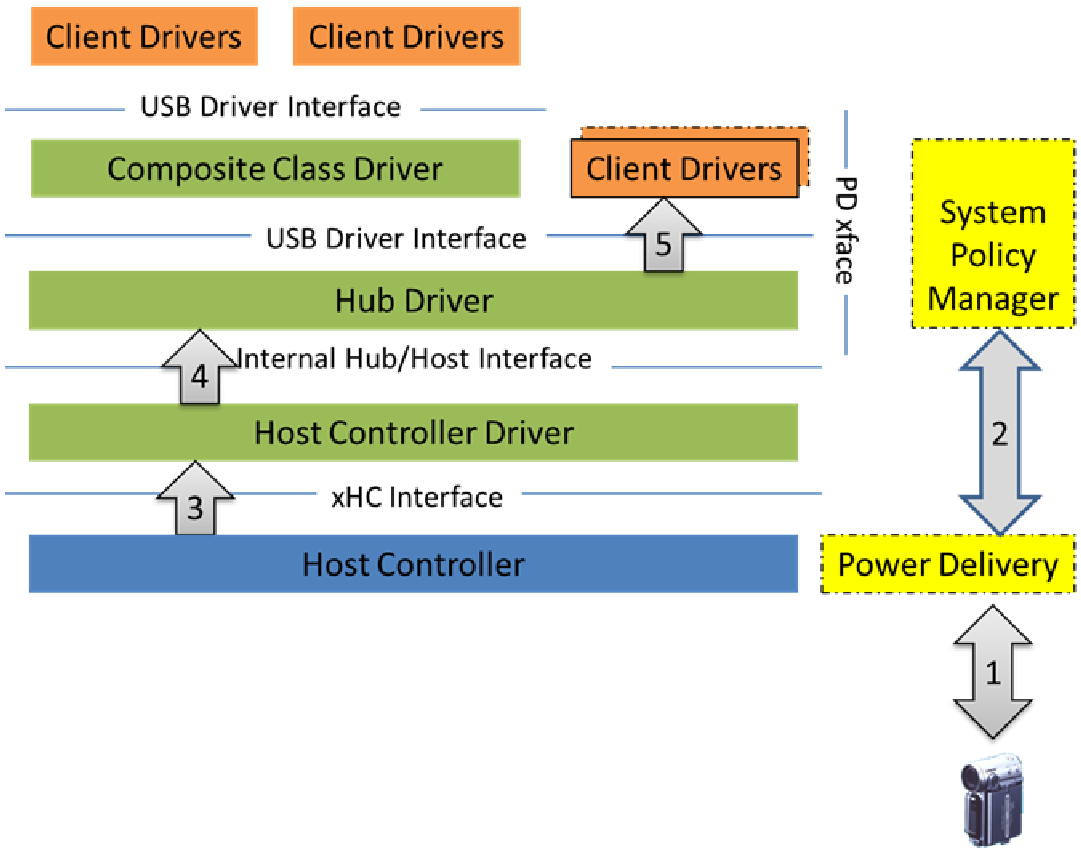


Figure 9-7 Enumeration of a PDUSB Device

If a PDUSB Device cannot perform its intended function with the amount of power that it can get from the port it is connected to then the host system should display a message (on a PD aware OS) about the failure to provide sufficient power to the device. In addition the device shall follow the requirements listed in Section 8.2.5.2.1.

9.2 PD Class Specific Descriptors

A PDUSB Device shall return all relevant descriptors mentioned in this section.

The following four capability descriptors shall be returned as part of the device's Binary Object Store (BOS) descriptor set. As with any Device Capability within the BOS descriptor it cannot be queried by itself and shall only be returned when software requests for the BOS descriptor set.

9.2.1 USB Power Delivery Capability Descriptor

Table 9-1 USB Power Delivery Capability Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of descriptor
1	*bDescriptorType*	1	Constant	DEVICE CAPABILITY Descriptor type
2	*bDevCapabilityType*	1	Constant	Capability type: POWER_DELIVERY_CAPABILITY
3	*bReserved*	1	Reserved	Shall be set to zero.
4	*bmAttributes*	4	Bitmap	Bitmap encoding of supported device level features. A value of one in a bit location indicates a feature is supported; a value of zero indicates it is not supported. Encodings are:

Offset	Field	Size	Value	Description
				<u>Bit</u> <u>Description</u> 0 Reserved. Shall be set to zero. 1 Battery Charging. This bit shall be set to one to indicate this device supports the Battery Charging Specification as per the value reported in the *bcdBCVersion* field. 2 USB Power Delivery. This bit shall be set to one to indicate this device supports the USB Power Delivery Specification as per the value reported in the *bcdPDVersion* field. 3 Provider. This bit shall be set to one to indicate this device is capable of providing power. This field is only valid if Bit 2 is set to one. 4 Consumer. This bit shall be set to one to indicate that this device is a consumer of power. This field is only valid if Bit 2 is set to one. 7:5 Reserved. Shall be set to zero. 15:8 bmPowerSource. At least one of the following bits 8, 9 and 14 shall be set to indicate which power sources are supported. [Bit | Description] [8 | AC or other Bulk Supply] [9 | Battery] [10 | Other] [13:11 | NumBatteries. This field shall only be valid when the Battery field is set to one and shall be used to report the number of batteries in the device.] [14 | Uses V_{BUS}] [15 | Reserved and shall be set to zero.] 31:16 Reserved and shall be set to zero.
8	*bmProviderPorts*	2	Bitmap	The bit corresponding to the port shall be set to one to indicate that this device is capable of providing power on that port. (Either BC 1.2 or PD) Bit zero refers to the upstream port of the device. Bits one through fifteen are only valid for hubs and refers to the downstream ports of the hub.

Offset	Field	Size	Value	Description
10	*bmConsumerPorts*	2	Bitmap	The bit corresponding to the port shall be set to one to indicate that this device is capable of consuming power on that port. (Either BC 1.2 or PD). Bit zero refers to the upstream port of the device. Bits one through fifteen are only valid for hubs and refers to the downstream ports of the hub.
12	*bcdBCVersion*	2	BCD	Battery Charging Specification Release Number in Binary-Coded Decimal (i.e., 2.10 is 210H). This field shall only be valid if the device indicates that it supports BC in the *bmAttributes* field.
14	*bcdPDVersion*	2	BCD	USB Power Delivery Specification Release Number in Binary-Coded Decimal. This field shall only be valid if the device indicates that it supports PD in the *bmAttributes* field.

9.2.2 Battery Info Capability Descriptor

A PDUSB Device shall support this capability descriptor if it reported that one of its power sources was a Battery in the *bmPowerSource* field in its Power Deliver Capability Descriptor. It shall return one Battery Info Descriptor per battery it supports.

Table 9-2 Battery Info Capability Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of descriptor
1	*bDescriptorType*	1	Constant	DEVICE CAPABILITY Descriptor type
2	*bDevCapabilityType*	1	Constant	Capability type: BATTERY_INFO_CAPABILITY
3	*iBattery*	1	Index	Index of string descriptor shall contain the user friendly name for this battery.
4	*iSerial*	1	Index	Index of string descriptor shall contain the Serial Number String for this battery.
5	*iManufacturer*	1	Index	Index of string descriptor shall contain the name of the Manufacturer for this battery.
6	*bBatteryId*	1	Number	Value shall be used to uniquely identify this battery in status messages.
7	*bReserved*	1	Number	Reserved and shall be set to zero.
8	*dwChargedThreshold*	4	mWh	Shall contain the Battery Charge value above which this battery is considered to be fully charged but not necessarily "topped off."
12	*dwWeakThreshold*	4	mWh	Shall contain the minimum charge level of this battery such that above this threshold, a device can be assured of being able to power up successfully (see Battery Charging 1.2).
16	*dwBatteryDesignCapacity*	4	mWh	Shall contain the design capacity of the battery.
20	*dwBatteryLastFullchargeCapacity*	4	mWh	Shall contain the maximum capacity of the battery when fully charged.

9.2.3 PD Consumer Port Capability Descriptor

A PDUSB Device shall support this capability descriptor if it reported that one, or more, of its ports are a Consumer port, as described in the *bmConsumerPorts* field in its Power Deliver Capability Descriptor. It shall return one PD

Consumer Port Capability descriptor per port that is a Consumer. The order of the descriptors shall be port order number (low to high).

A PDUSB Peripheral Device shall have at most one Consumer port. A PDUSB Hub may have a maximum of sixteen Consumer ports.

Table 9-3 PD Consumer Port Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of descriptor
1	*bDescriptorType*	1	Constant	DEVICE CAPABILITY Descriptor type
2	*bDevCapabilityType*	1	Constant	Capability type: PD_CONSUMER_PORT_CAPABILITY
3	*bReserved*	1	Number	Reserved and shall be set to zero.
4	*bmCapabilities*	2	Bitmap	Capability: This field shall indicate the specification the Consumer port will operate under. Bit | Description 0 | Battery Charging (BC) 1 | USB Power Delivery (PD) 15:2 | Reserved and shall be set to zero.
6	*wMinVoltage*	2	Number	Shall contain the minimum voltage that this Consumer is capable of operating at.
8	*wMaxVoltage*	2	Number	Shall contain the maximum voltage that this Consumer is capable of operating at.
10	*wReserved*	2	Number	Reserved and shall be set to zero.
12	*dwMaxOperatingPower*	4	Number	Shall contain the maximum power in 10mW units this Consumer can draw when it is in a steady state operating mode.
16	*dwMaxPeakPower*	4	Number	Shall contain the maximum power in 10mW units this Consumer can draw for a short duration of time (*dwMaxPeakPowerTime*) before it falls back into a steady state.
20	*dwMaxPeakPowerTime*	4	Number	Shall contain the time in 100ms units that this Consumer can draw peak power. A device shall set this field to 0xFFFF if this value is unknown.

9.2.4 PD Provider Port Capability Descriptor

A PDUSB Device shall support this capability descriptor if it reported that one or more of its ports is a Provider port, as described in the *bmProviderPorts* field in its Power Deliver Capability Descriptor. It shall return one PD Provider Port Capability descriptor per port that is a Provider. The order of the descriptors will be port order number (low to high).

A PDUSB Peripheral Device shall have at most one Provider port. A PDUSB Hub can have a maximum of sixteen Provider ports.

Table 9-4 PD Provider Port Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of descriptor
1	*bDescriptorType*	1	Constant	DEVICE CAPABILITY Descriptor type
2	*bDevCapabilityType*	1	Constant	Capability type: PD_PROVIDER_PORT_CAPABILITY
3	*bReserved*	1	Number	Reserved and shall be set to zero.
4	*bmCapabilities*	2	Bitmap	Bits 0 and 1 shall be set if the corresponding capability is supported: Bit / Description; 0 / Battery Charging (BC); 1 / USB Power Delivery (PD); 15:2 / Reserved. Shall be set to zero.
6	*bNumOfPDObjects*	1	Number	Shall indicate the number of Power Data Objects.
7	*bReserved*	1	Number	Reserved and shall be set to zero.
8	*wPowerDataObject1*	4	Bitmap	Shall contain the first Power Data Object supported by this Provider port. See Section 6.4.1 for details of the Power Data Objects.
…	…	…	…	…
N+4	*wPowerDataObjectN*	4	Bitmap	Shall contain the 2nd and subsequent Power Data Objects supported by this Provider port. See Section 6.4.1 for details of the Power Data Objects.

9.3 PD Class Specific Requests and Events

A PDUSB Hub that is compliant to this specification shall support the PD specific requests and events detailed in this section irrespective of whether the PDUSB Hub is a Power Provider, a Power Consumer, or both.

A PDUSB Device that is compliant to this specification shall support the battery related requests if it has a battery.

The events shall be sent to the system software in response to changes on the PDUSB Hub ports that are not a direct result of a request from system software. These notifications shall be sent over the Interrupt endpoint of a PDUSB Hub.

Note that a PDUSB Hub shall only send these notifications, or respond to PD requests (except for GetBatteryStatus); if it has been made aware that the SPM is active on the host as detailed in Section 9.4.4.

9.3.1 Class-specific Requests

The PD class defines requests to which PDUSB Devices shall respond as outlined in Table 9-5. All requests in Table 9-5 shall be implemented by PDUSB devices with data lines and PDUSB hubs.

Table 9-5 PD Class Requests

Request	bmRequestType	bRequest	wValue	wIndex	wLength	Data
ClearPortPDFeature[1]	00100011B	CLEAR_FEATURE	Feature Selector	Port Number	Two	Clear Change Mask
GetBatteryStatus[2]	10100000B	GET_BATTERY_STATUS	Zero	Battery ID	Eight	Battery Status

Request	bmRequestType	bRequest	wValue	wIndex	wLength	Data
GetPartnerPDO¹	10100011B	GET_PARTNER_PDO	Zero or One	Port Number	Variable	Power Data Objects
GetPDStatus¹	10100011B	GET_STATUS	Zero	Port Number	Six	PD Status
SetPDFeature¹	00100000B	SET_FEATURE	Feature Selector	Feature Specific	Zero	None
SetPortPDFeature¹	00100011B	SET_FEATURE	Feature Selector	Port Number	Zero	None
SetPortPDOs¹	00100011B	SET_PDO	Zero	Port Number	Variable	Provider Power Data Objects

¹Requests valid for PDUSB Hubs only.

²Requests valid for PDUSB Hubs and PDUSB Peripheral Devices.

Table 9-6 gives the bRequest values for commands that are not listed in the hub/device framework chapters of ***[USB2.0]***, ***[USB3.0]***.

Table 9-6 PD Class Request Codes

bRequest	Value
GET_PARTNER_PDO	20
GET_BATTERY_STATUS	21
SET_PDO	22

Table 9-7 gives the valid feature selectors for the PD class. Refer to Sections 9.4.1 and 9.4.4 for a description of the features.

Table 9-7 PD Class Feature Selectors

Feature Selector	Recipient	Value
BATTERY_WAKE_MASK	Device	40
OS_IS_PD_AWARE	Device	41
POLICY_MODE	Device	42
SWAP	Port	43
GOTO_MIN	Port	44
RETURN_POWER	Port	45
ACCEPT_PD_REQUEST	Port	46
REJECT_PD_REQUEST	Port	47
PORT_PD_RESET	Port	48
C_PORT_PD_CHANGE	Port	49

9.3.2 PDUSB Hub Event Reporting

PDUSB Hubs shall report events back to the system via the "Port N change detected" bit in the PDUSB Hub Status Change bitmap returned via the PDUSB Hub notification endpoint. 'N' refers to the port number of the port on which a PD change has occurred.

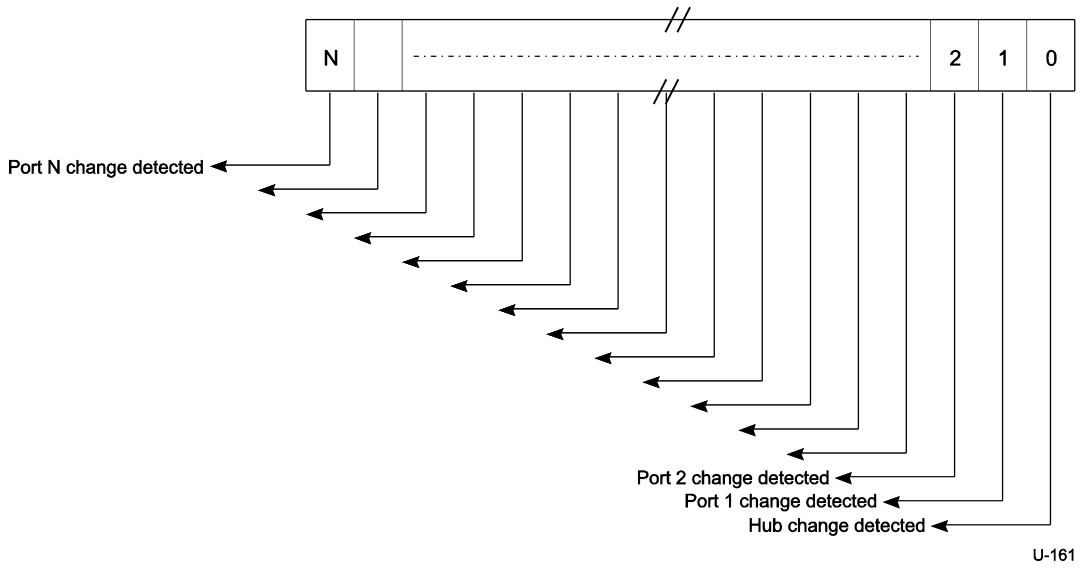


Figure 9-8 Hub Status Change

In order to indicate that a PD change occurred, a PDUSB Hub compliant to the PD class shall set Bit 15 "C_PORT_PD_CHANGE" in the Port Change field when queried using the standard Get Port Status command.

A PDUSB Hub shall send these notifications for all downstream ports irrespective of whether it is a Consumer or Provider on that port. For example, if a PDUSB Peripheral Device is a Provider on one of the downstream ports and it starts a negotiation to change the amount of power it will deliver – the PDUSB Hub sends a Port Change notification to the SPM. However, if the PDUSB Hub loses all power (e.g. the power Provider is unplugged) then it is sufficient if the PDUSB Hub disconnects from the downstream port it is attached to without sending any notifications.

9.4 PDUSB Hub and PDUSB Peripheral Device Requests

9.4.1 Clear Port PD Feature (PDUSB Hub)

This request resets a value reported in the PDUSB Hub status. This request is only valid for hubs.

bmRequestType	bRequest	WValue	wIndex	wLength	Data
10100011B	CLEAR_FEATURE	Feature Selector	Port Number	Two	Clear Change Mask

The port number shall be a valid port number for that PDUSB Hub, greater than zero. The port field is located in bits 7..0 of the *wIndex* field.

Clearing a feature shall disable that feature; refer to Table 9-7 for the feature selector definitions that shall apply to the port as a recipient. If the feature selector is associated with a status change, the Clear Change Mask shall define the status changes that are being acknowledged by the SPM. Only when all the status changes in PD Status Change field (see Table 9-10) are acknowledged shall the hub clear the C_PORT_PD_CHANGE bit in the port's Port Change field. The Clear Change Mask is defined in Table 9-8.

Table 9-8 Clear Change Mask

Bit	Change Acknowledged when bit is set
0	Reserved
1	External supply.

Bit	Change Acknowledged when bit is set
2	Port operation mode
3	Supported Provider Capabilities
4	Negotiated power level
5	New power direction
6	PD Reset Complete
7	Insertion Detect
8	PD Capable Cable
9	Request Rejected
10	CableVGO
11-14	Reserved
15	Error

It shall be a Request Error if *wValue* is not a feature selector listed in Table 9-7, or if *wIndex* specifies a port that does not exist, or if *wLength* is not as specified above.

If the PDUSB Hub is not configured, the PDUSB Hub's response to this request is undefined.

9.4.2 Get Battery Status (PDUSB Hub/Peripheral Device)

This request returns the current status of the battery in a PDUSB Hub/Peripheral Device.

bmRequestType	**bRequest**	**wValue**	**wIndex**	**wLength**	**Data**
10100000B	BATTERY_STATUS	Zero	Battery ID	Eight	Battery Status

The PDUSB Hub/Peripheral Device shall return the Battery Status of the Battery identified by the value of *wIndex* field.

Every PDUSB Device that has a battery shall return its Battery Status when queried with this request. For Providers or Consumers with multiple batteries, the status of each battery shall be reported per battery.

Table 9-9 Battery Status Structure

Offset	Field	Size	Value	Description
0	*bBatteryAttributes*	1	Number	Shall indicate whether a battery is installed and whether this is charging or discharging. Value / Description: 0 – There is no battery 1 – The battery is charging 2 – The battery is discharging 3 – The battery is neither discharging nor charging 255-4 – Reserved and shall not be used
1	*bBatterySOC*	1	Number	Shall indicate the Battery State of Charge given as percentage value from Battery Remaining Capacity.
2	*bBatteryStatus*	1	Number	If a battery is present shall indicate the present status of the battery.

Offset	Field	Size	Value	Description
				<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>No error</td></tr><tr><td>1</td><td>Battery required and not present</td></tr><tr><td>2</td><td>Battery non-chargeable/wrong chemistry</td></tr><tr><td>3</td><td>Over-temp shutdown</td></tr><tr><td>4</td><td>Over-voltage shutdown</td></tr><tr><td>5</td><td>Over-current shutdown</td></tr><tr><td>6</td><td>Fatigued battery</td></tr><tr><td>7</td><td>Unspecified error</td></tr><tr><td>255-8</td><td>Reserved and shall not be used</td></tr></table>
3	*bRemoteWakeCapStatus*	1	Bitmap	If the device supports remote wake, then the device shall support Battery Remote wake events. The default value for the Remote wake events shall be turned off (set to zero) and can be enable/disabled by the host as required. See Section 9.4.5<table><tr><th>Bit</th><th>Description</th></tr><tr><td>0</td><td>Battery present event</td></tr><tr><td>1</td><td>Charging flow</td></tr><tr><td>2</td><td>Battery error</td></tr><tr><td>7:3</td><td>Reserved and shall be set to zero</td></tr></table>
4	*wRemainingOperatingTime*	2	Number	Shall contain the operating time (in minutes) until the Weak Battery threshold is reached, based on Present Battery Strength and the device's present operational power needs. Note: this value shall exclude any additional power received from charging. A battery that is not capable of returning this information shall return a value of 0xFFFF.
6	*wRemainingChargeTime*	2	Number	Shall contain the remaining time (in minutes) until the Charged Battery threshold is reached based on Present Battery Strength, charging power and the device's present operational power needs. Value shall only be valid if the Charging Flow is "Charging". A battery that is not capable of returning this information shall return a value of 0xFFFF.

If *wValue* or *wLength* are not as specified above, then the behavior of the PDUSB Device is not specified.
If *wIndex* refers to a Battery that does not exist, then the PDUSB Device shall respond with a Request Error.
If the PDUSB Device is not configured, the PDUSB Hub's response to this request is undefined.

9.4.3 Get Port Partner PD Objects (PDUSB Hub)

This request returns the Provider/Consumer PD Objects of the PDUSB Device attached to the downstream port on a PDUSB Hub.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
10100011B	GET_PARTNER_PDO	Zero or one	Port Number	Variable	Power Data Objects

The *wIndex* field shall indicate the port number to which a PDUSB Device is attached. The port number shall be a valid port number for that PDUSB Hub, greater than zero.

When this request is received by the PDUSB Hub it shall return the Consumer PDOs (if wValue is set to zero) or Provider PDOs (if wValue is set to one) for the PDUSB Device connected on the downstream port indicated by the *wIndex* field. The PDUSB Hub shall use PD mechanisms (see Section 6.3.8) to retrieve the Power Data Objects from the PDUSB Device.

The *wLength* field shall specify the number of bytes to return. The maximum length of this field is a function of the maximum number of PDOs a PDUSB device can return as defined in Section 6.2.1.2. If the device on the Partner Port does not support PD, then the PDUSB Hub shall respond with a Request Error.

If the PDUSB Hub is not configured, the PDUSB Hub's response to this request is undefined.

9.4.4 Get PD Status (PDUSB Hub)

This request returns the PD Status of the specified port on a PDUSB Hub.

bmRequestType	**bRequest**	**wValue**	**wIndex**	**wLength**	**Data**
10100011B	GET_ STATUS	Zero	Port Number	Eight	PD Status

The *wIndex* field shall indicate the PD capable port that is being queried. The port number shall be a valid port number for that PDUSB Hub, greater than zero.

If *wValue* or *wLength* are not as specified above, then the behavior of the PDUSB Hub is not specified.

If a Port is specified that does not exist, then the PDUSB Hub shall respond with a Request Error. The PD status changes on the port are indicated in the PD Status Change field. The SPM can acknowledge the Port Change notifications via the Clear Port PD Feature (See Section9.4.1).

If the PDUSB Hub is not configured, the PDUSB Hub's response to this request is undefined.

A GetPDStatus() request to a PDUSB Hub shall return the information as shown in Table 9-10.

Table 9-10 PD Status

Bit	Description
0:15	**PD Status Change**: A one in the bit position shall indicate the types of status changes that have occurred on the Port.

Bit	Change Indicated
0	Reserved and shall not be used.
1	External supply. When set, the **Supply** field shall indicate the current status of the supply.
2	Port operation mode. When set the **Port Operation Mode** field shall indicated the current operational mode of the port.
3	Supported Provider Capabilities. When set, the SPM shall use the Get PD descriptor to get further details on this notification.
4	Negotiated power level. When set, the **Request Data Object** field shall indicated the newly negotiated power level if the PDUSB Hub was operating in Non-Intruisive mode. If the PDUSB Hub is working in intrusive mode then, when set, the **Request Data Object** field shall indicate the power level that the port partner wants to operate at.
5	New power direction. When set, this field shall indicate that the power direction has changed if the PDUSB Hub was operating in Non-Intrusive mode. If the PDUSB Hub is

Bit	Description
	working in intrusive mode then, when set, this field shall indicate the port partner wants to perform a power swap.
6	PD Reset Complete. When set, this field shall indicate that a PD Hard Reset has completed. When this bit is set, then the other bits in the **PD Status Change** field shall report their change status as if the PDUSB Hub was working in Non-Intrusive mode.
7	Insertion Detect. When set, the **Insertion Detect** field shall indicate the current status of whether a cable is plugged into this port.
8	PD Capable Cable. When set, the **PD Capable Cable** field shall indicate the current status of whether a PD Capable Cable is plugged into this port.
9	Request Rejected. This bit shall be only valid if the PDUSB Hub is operating in Non-Intrusive mode. When set, this field shall indicate one of two conditions: • the PDUSB Hub received a request to change the power level (Negotiated power level (Bit 4) bit in this field shall be set) that was rejected or • the PDUSB Hub received a request to role swap (New power direction bit (Bit 5) in this field shall be set) that was rejected.
10	CableVGO. When set, this field shall indicate that the IR drop over the ground connection exceeded the allowable drop.
11-14	Reserved and shall be set to zero.
15	Error. When set, this field shall indicate that an unknown error has occurred on the port.

16 — **Supply**: This field shall indicate the type of supply that is attached to this port.

Value	Meaning
0	No Supply
1	Supply Present

17:19 — **Port Operation Mode**: The field shall indicate the current mode of operation of the port.

Value	Meaning
0	No Consumer
1	Legacy
2	BC
3	PD
4-7	Reserved

20 — **Insertion Detect**: The field shall indicate the current status of whether a cable is plugged into this port.

Value	Meaning
0	No cable detected
1	Cable detected

21 — **PD Capable Cable**: The field shall indicate the current status of whether a PD Capable cable is plugged into this port.

Value	Meaning
0	PD Capable Cable not detected
1	PD Capable Cable detected

Bit	Description
22:31	Reserved. Shall be set to zero.

Bit	Description
32:63	**Request Data Object**: This field shall return the currently negotiated power level if the hub was operating in Non-Intrusive mode. If the PDUSB Hub is working in intrusive mode then this field shall indicate the power level that the port partner wants to operate at. See Table 6-10 for additional information on the contents of this data structure.

9.4.5 Set PD Feature (PDUSB Hub/Peripheral Device)

This request sets the value requested in the PDUSB Hub/Peripheral Device.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00100000B	SET_ FEATURE	Feature Selector	Feature Specific	Zero	None

Setting a feature enables that feature or starts a process associated with that feature; see Table 9-7 for the feature selector definitions. Features that may be set with this request are:

- BATTERY_WAKE_MASK
- OS_IS_PD_AWARE (Only valid for PDUSB Hubs)
- POLICY_MODE (Only valid for PDUSB Hubs)

When the feature selector is set to BATTERY_WAKE_MASK, then the *wIndex* field is structured as shown in the following table.

Table 9-11 Battery Wake Mask

Bit	Description
0	**Battery Present**: When this bit is set then the PDUSB Device shall generate a wake event if it detects that a battery has been attached.
1	**Charging Flow**: When this bit is set then the PDUSB Device shall generate a wake event if it detects that a battery switched from charging to discharging or vice versa.
2	**Battery Error**: When this bit is set then the PDUSB Device shall generate a wake event if the battery has detected an error condition.
15:3	Reserved and shall not be used.

The SPM may Enable or Disable the wake events associated with one or more of the above events by using this feature. This feature shall be the only feature that may be set by the SPM for a PDUSB Peripeheral Device. The other features that may be set by this request are only valid for PDUSB Hubs.

When the feature selector is set to OS_IS_PD_AWARE, then if the *wIndex* field is set to one it informs the PDUSB Hub that the SPM is active on the host. If the *wIndex* field is set to zero then it informs the PDUSB Hub that the SPM is not active on the host. The default state for this feature shall be zero. The PDUSB Hub shall respond to all PD requests (except GetBatteryStatus) or send any PD notifications only after the OS_IS_PD_AWARE feature is set.

When the feature selector is set to POLICY_MODE, the *wIndex* field shall be set to one of the values in Table 9-12.

Table 9-12 Policy Mode Encoding

Value	Description
00H	Local: Status Reporting only (Default)
01H	Intrusive: All requests are sent to the SPM

Value	Description
02H	Hybrid: Send only requests that cannot be accommodated to the SPM
03H-FFFFH	Reserved and shall not be used

If the Policy Mode is set to Intrusive, then the PDUSB Hub shall be set to operate in Intrusive mode. If the Policy Mode is set to Local or Hybrid, the hub shall be set to operate in Non-Intrusive mode.

When the Policy Mode is set to Intrusive then the PDUSB Hub shall pass any requests that need policy decisions to be made to the SPM and use PD mechanisms to delay the completion of PD Requests (e.g. using the Wait Message) over PD. When the Policy Mode is set to Hybrid then only requests which cannot be accommodated by the PDUSB Hub shall be sent to the SPM.

Some requests are only valid when the PDUSB Hub is operating in Intrusive mode. The requests that are valid in Intrusive mode are indicated as such in the description for those requests.In addition notifications that operate differently in Intrusive mode are also called out in the description of those notifications.

It shall be a Request Error if *wValue* is not a feature selector listed in Table 9-7 or if *wIndex* is not set to a value as specified above.

If the PDUSB Hub is not configured, the PDUSB Hub's response to this request is undefined.

9.4.6 Set Port PD Feature (PDUSB Hub)

This request sets the feature for a port in a PDUSB Hub. This request shall only be valid when the PDUSB Hub is operating in Intrusive mode. If the PDUSB Hub is not operating in Intrusive mode, the PDUSB Hub's response to this request is undefined. If the PDUSB Hub cannot satisfy a request then it shall respond with a Request Error to the SPM.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00100011B	SET_ FEATURE	Feature Selector	Port Number	Zero	None

The *wIndex* field shall indicate the PD capable port that is being queried. The port number shall be a valid port number for that hub, greater than zero.

Setting a feature enables that feature or starts a process associated with that feature; see Table 9-7 for the feature selector definitions. Features that may be set with this request are:

- SWAP
- GOTO_MIN
- RETURN_POWER
- ACCEPT_PD_REQUEST
- REJECT_PD_REQUEST
- PORT_PD_RESET

When the feature selector is set to SWAP then the PDUSB Hub shall intiate a Role Swap with the port partner on the specified port. See Section 7.3.9 and Section 7.3.10 for details on Swap message.

When the feature selector is set to GOTO_MIN and the downstream port is operating as a Provider, then the PDUSB Hub shall send the ***GotoMin*** request to the port partner on the specified port. See Section 6.3.2 for details on ***GotoMin*** request.

When the feature selector is set to RETURN_POWER and the downstream port is operating as a Provider, then the PDUSB Hub shall return the power that it borrowed from the PD Device on that port.

When the feature selector is set to ACCEPT_PD_REQUEST then the PDUSB Hub shall accept the request that was previously made by the port partner on this downstream port.

When the feature selector is set to REJECT_PD_REQUEST then the PDUSB Hub shall reject the request that was previously made by the port partner on this downstream port.

When the feature selector is set to PORT_PD_RESET, the port shall be sent a Hard Reset. Once the Hard Reset is complete, as described in Section 8.3.2.8, the hub shall set the C_PORT_PD_CHANGE bit in the port change field.

It shall be a Request Error if *wValue* is not a feature selector listed in Table 9-7, or if *wIndex* specifies a port that does not exist or if the port is operating as a Consumer, or if *wLength* is not as specified above.

If the PDUSB Hub is not configured, the PDUSB Hub's response to this request is undefined.

9.4.7 Set Port PDOs (PDUSB Hub)

This request sets the PDOs that a port in a PDUSB Hub shall send to its Port Partner if the port on the PDUSB Hub is operating as a Provider. This request shall only be valid when the PDUSB Hub is operating in Intrusive mode. If the PDUSB Hub is not operating in Intrusive mode, the PDUSB Hub's response to this request is undefined.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00100011B	SET_PDO	Zero	Port Number	Variable	Provider Power Data Objects

The *wIndex* field shall indicate the PD capable port that is being queried. The port number shall be a valid port number for that hub, greater than zero.

The *wLength* field shall be a multiple of 4 and shall include all the PDOs (see Section 6.4.1) that the port may send to its port partner when sending its PD capabilities. The port shall send the PDOs in the order that they are included in this request. It is the responsibility of the PDUSB Hub to verifiy that the contents of the PDOs sent by the SPM are valid for this port. A PDUSB Hub shall respond with a Request Error if the PDOs sent by the SPM are not valid for the port specified.

It shall be a Request Error if *wValue* is not set to zero or if *wIndex* specifies a port that does not exist, or if *wLength* is not as specified above.

If the port on the PDUSB Hub is not operating as a Provider, the PDUSB Hub's response to this request is undefined.

If the PDUSB Hub is not configured, the PDUSB Hub's response to this request is undefined.

A Power Profiles

Power Profiles are optional normative. They define a standardized set of voltages at several current ranges that are offered by USB Power Delivery Sources. The profiles are defined for Sources only.

The Power Profiles are optional but are intended to provide a finite set of power levels to:

- Limit the number of voltages and current combinations a Source has to supply
- Provide a well-defined set of voltage and current combinations from which a Sink can choose
- Provide a selection of power ranging from 10W to 100W in approximately 2x steps
- Limit the number of valid combinations

A.1 Profile Definitions

There are the following profiles based on Fixed Supply Objects:

- Profile 0 reserved
- Profile 1 capable of supplying 5V @ 2.0A
- Profile 2 ports are capable of supplying 5V@ 2.0A, 12V @ 1.5A.
- Profile 3 ports are capable of supplying 5V @ 2.0A, 12V @ 3A.
- Profile 4 ports are capable of supplying 5V @ 2.0A, 12V and 20V at 3A respectively.
- Profile 5 ports are capable of supplying 5V @ 2.0A, 12V and 20V at 5A respectively.

Power Profiles are defined to overlap such that a Device that requires a Profile 2 Source will operate equally well when connected to a Profile 2 or any higher Profile Source.

Sources may have additional capabilities. For example a Source might advertise 5V @ 2.0A, 12V and 15V at 1.5A respectively. It is a Profile 2 Source because it meets the Profile 2 requirements to supply 5V @ 2.0A and 12V @ 1.5A. The fact that it can also supply 15V will have no effect on a Device that wants a Profile 2 source.

Profile 5 Sources that are capable of 100W operation are subject to various worldwide safety standards. In order to meet the most common safety standards, the continuous output power cannot exceed 100W and the continuous output current cannot exceed 5A. The industry is well versed in meeting the safety requirements for such power sources (e.g. Wall Warts). Refer to Figure A-1 for an interpretation of the safety requirements imposed by IEC/UL60950 Table 2B.

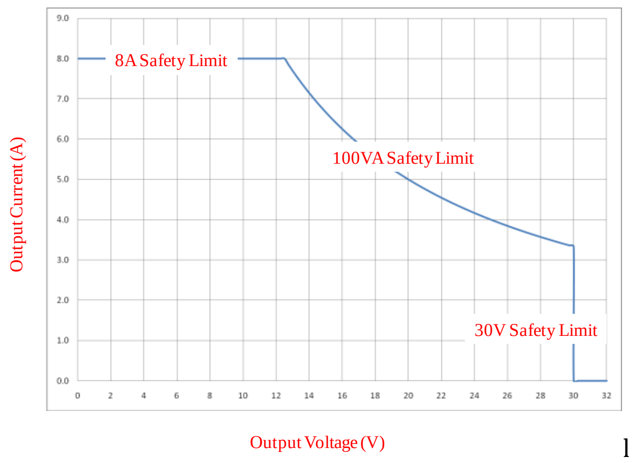


Figure A-1 Interpretation of UL60950

A.2 Voltage Selection Rationale

Voltages used by the profiles were not picked randomly; this section describes the rationale behind the choices.

5V is the USB default that must be supported to provide interoperability with existing Devices. However, higher voltages are needed to provide more power through USB connectors because of their current carrying capability.

12V was selected because it is very common in PCs and many other systems. The current limitation of the Micro USB Connector family is 3A for the enhanced PD version. The use of 12V with 3A provides sufficient power to charge tablets in the 20-30W range that use the Micro USB Connector.

20V was selected because larger systems, such as notebooks, often have 4 lithium cells in series and require charging voltages in the 18-20V range. A sampling of systems showed that chargers for these systems were typically 19.5-20V. Typical systems had chargers that supplied between 60W and 100W with the exception of a few very high-end performance systems that were well over 100W. The 5A current limit of the PD enhanced Standard-A and Standard-B connectors, with 20V allows up to 100W to be delivered to charge this class of Devices.

A.3 Relevance of Profiles to Sink Ports

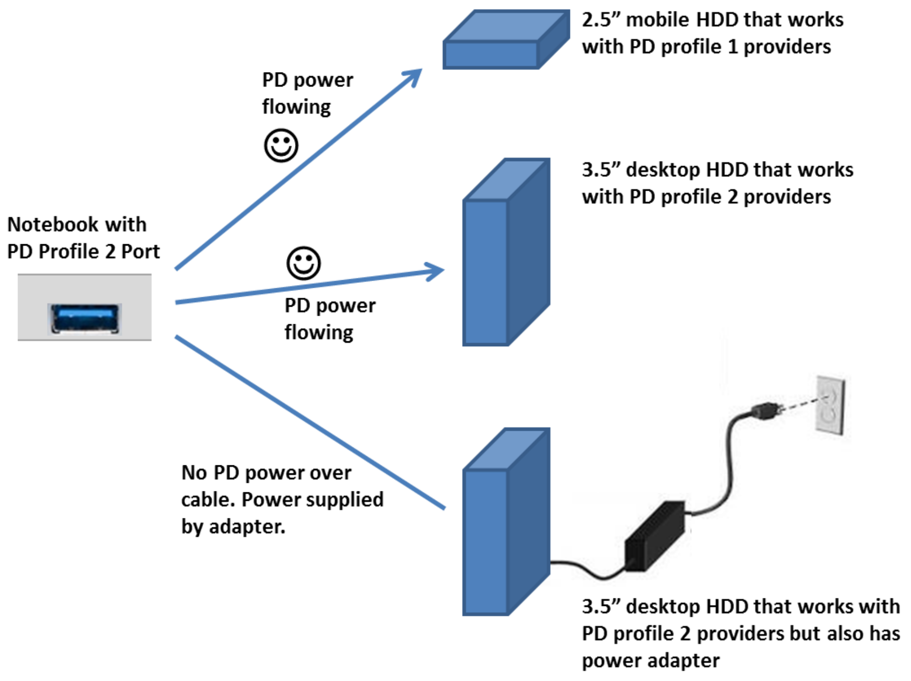
Sinks do not have Power Profiles per se. A Sink port only need to be compatible with the voltage and current offered in one or more Power Profiles. This can lead to some interesting cases where dual-role ports are involved (see Appendix A.4 for examples). A dual- role port may offer one Profile when operating as a Source and request power from another Profile when operating as a Sink. For example, a tablet might offer Profile 1 (5V @ 2.0A) when operating as a Source, but request power (12V @ 3A) supplied by a Profile 3 Source when operating as a Sink. Another example is a notebook computer with a Profile 2 Source port it also uses as a Sink for charging which requires it be supplied with 20V @ 5A from Profile 5 Source.

A.4 System Level Examples

The following examples are provided to help illustrate some typical system use cases. Examples using a notebook and HDDs have been selected but the principles apply equally to other types of USB Power Delivery devices (e.g. printers, scanners, tablets, phones, mp3 players etc.).

A.4.1 Notebook and HDD Examples

In the following two examples, the user has varying experiences attaching his hard disk-drive to a PD port on his notebook computer.



2.5" mobile HDD that works with PD profile 1 providers
PD power flowing
☺
3.5" desktop HDD that works with PD profile 2 providers
Notebook with PD Profile 2 Port
☺
PD power flowing
No PD power over cable. Power supplied by adapter.
3.5" desktop HDD that works with PD profile 2 providers but also has power adapter

Figure A-2 Notebook is Capable of Meeting User's Requirements

The notebook shown in Figure A-2 is able to supply up to PD profile 2. The larger HDD is able to successfully operate without a separate power adapter since it gets adequate power from the PD port. In the case of the HDD with separate power adapter plugged in, the HDD should not draw power from the notebook's port.

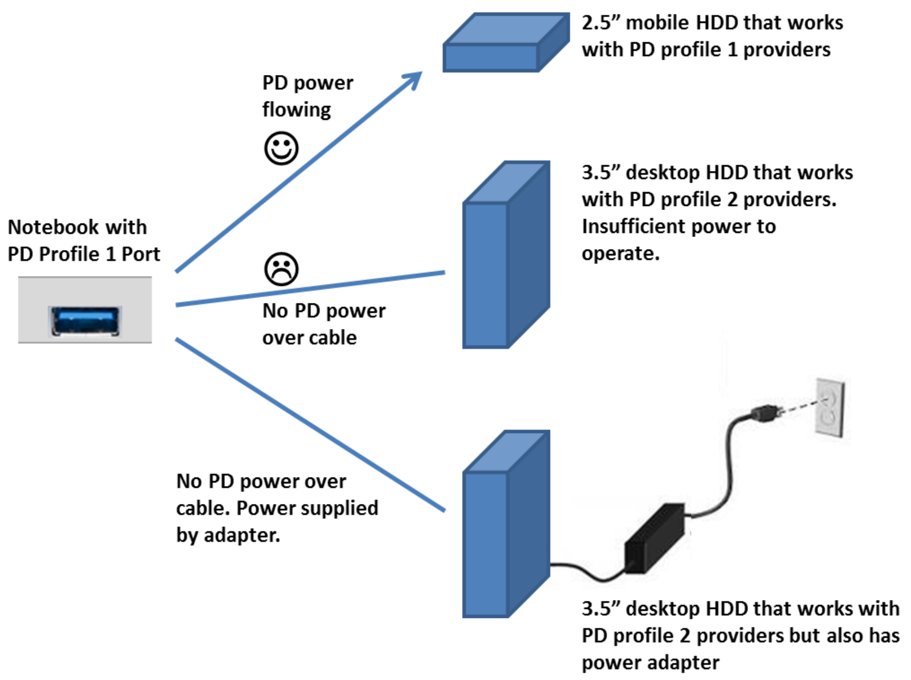

2.5” mobile HDD that works with PD profile 1 providers
PD power flowing
☺
3.5” desktop HDD that works with PD profile 2 providers. Insufficient power to operate.
Notebook with PD Profile 1 Port
☹
No PD power over cable
No PD power over cable. Power supplied by adapter.
3.5” desktop HDD that works with PD profile 2 providers but also has power adapter

Figure A-3 Notebook is not Capable of Meeting User’s Requirements

Note that in the case shown in Figure A-3, the experience for the user is not unlike their experience today. USB HDDs that attempt to rely only on USB bus power only will often fail because not all notebooks on the market are capable of supplying adequate current.

A.4.2 Display with Hub Example

The following example is for a display dock where the integrated hub also provides power and at least one of the hub ports is of high enough power capacity to power the host platform.

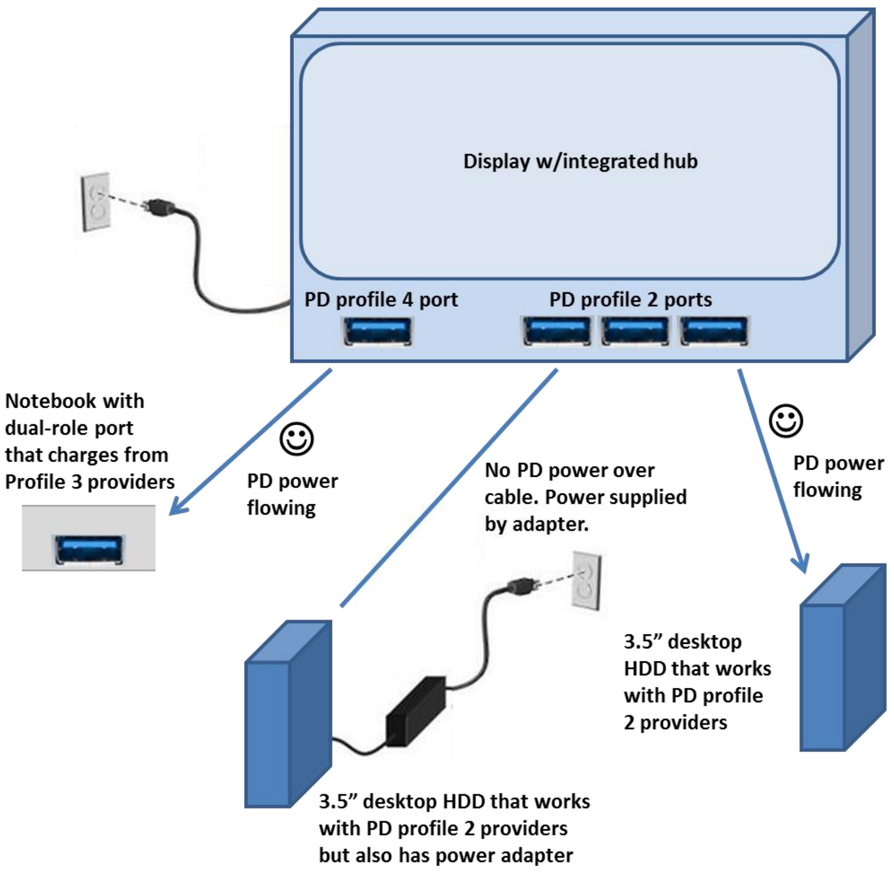

Display w/integrated hub
PD profile 4 port
PD profile 2 ports
Notebook with dual-role port that charges from Profile 3 providers
☺
PD power flowing
No PD power over cable. Power supplied by adapter.
☺
PD power flowing
3.5" desktop HDD that works with PD profile 2 providers
3.5" desktop HDD that works with PD profile 2 providers but also has power adapter

Figure A-4 Display with Integrated Hub Capable of Meeting User's Requirements

B CRC calculation

B.1 C code example

```
//
// USB PD CRC Demo Code.
//

#include <stdio.h>

int crc;


//---------------------------------------------------------------------------

void crcBits(int x, int len) {

  const int poly = 0x04C11DB6;  //spec 04C1 1DB7h
  int newbit,newword,rl_crc;

  for(int i=0;i<len;i++) {

    newbit = ((crc>>31) ^ ((x>>i)&1)) & 1;
    if(newbit) newword=poly; else newword=0;
    rl_crc = (crc<<1) | newbit;
    crc = rl_crc ^ newword;
    printf("%2d newbit=%d, x>>i=0x%x, crc=0x%x\n",i,newbit,(x>>i),crc);
  }
}

int crcWrap(int c){

  int ret = 0;
  int j, bit;

  c = ~c;
  printf("~crc=0x%x\n",c);

  for(int i=0;i<32;i++) {
    j = 31-i;

    bit = (c>>i) & 1;
    ret |= bit<<j;

  }
  return ret;

}

//---------------------------------------------------------------------------

int main(){

  int txCrc=0,rxCrc=0,residue=0,data;
  printf("using packet data 0x%x\n", data=0x0101);

  crc = 0xffffffff;
  crcBits(data,16);
  txCrc = crcWrap(crc);

  printf("crc=0x%x, txCrc=0x%x\n",crc,txCrc);



  printf("received packet after decode= 0x%x, 0x%x\n",data,txCrc);

  crc = 0xffffffff;
  crcBits(data,16);
  rxCrc = crcWrap(crc);

  printf("Crc of the received packet data is (of course) =0x%x\n",rxCrc);

  printf("continue by running the transmit crc through the crc\n");
  crcBits(rxCrc,32);

  printf("Now the crc residue is 0x%x\n",crc);

  printf("should be 0xc704dd7b\n");

}
```

B.2 Table showing the full calculation over one message

Function	Nibble	Symbol	Bits	CRC register transmitter	CRC register receiver	bit nr.
Preamble			0	FFFFFFFF	FFFFFFFF	1
			1	FFFFFFFF	FFFFFFFF	2
			0	FFFFFFFF	FFFFFFFF	3
			1	FFFFFFFF	FFFFFFFF	4
			0	FFFFFFFF	FFFFFFFF	5
			1	FFFFFFFF	FFFFFFFF	6
			0	FFFFFFFF	FFFFFFFF	7
			1	FFFFFFFF	FFFFFFFF	8
			0	FFFFFFFF	FFFFFFFF	9
			1	FFFFFFFF	FFFFFFFF	10
			0	FFFFFFFF	FFFFFFFF	11
			1	FFFFFFFF	FFFFFFFF	12
			0	FFFFFFFF	FFFFFFFF	13
			1	FFFFFFFF	FFFFFFFF	14
			0	FFFFFFFF	FFFFFFFF	15
			1	FFFFFFFF	FFFFFFFF	16
			0	FFFFFFFF	FFFFFFFF	17
			1	FFFFFFFF	FFFFFFFF	18
			0	FFFFFFFF	FFFFFFFF	19
			1	FFFFFFFF	FFFFFFFF	20
			0	FFFFFFFF	FFFFFFFF	21
			1	FFFFFFFF	FFFFFFFF	22
			0	FFFFFFFF	FFFFFFFF	23
			1	FFFFFFFF	FFFFFFFF	24
			0	FFFFFFFF	FFFFFFFF	25
			1	FFFFFFFF	FFFFFFFF	26
			0	FFFFFFFF	FFFFFFFF	27
			1	FFFFFFFF	FFFFFFFF	28
			0	FFFFFFFF	FFFFFFFF	29
			1	FFFFFFFF	FFFFFFFF	30
			0	FFFFFFFF	FFFFFFFF	31
			1	FFFFFFFF	FFFFFFFF	32
			0	FFFFFFFF	FFFFFFFF	33
			1	FFFFFFFF	FFFFFFFF	34
			0	FFFFFFFF	FFFFFFFF	35
			1	FFFFFFFF	FFFFFFFF	36
			0	FFFFFFFF	FFFFFFFF	37
			1	FFFFFFFF	FFFFFFFF	38
			0	FFFFFFFF	FFFFFFFF	39
			1	FFFFFFFF	FFFFFFFF	40
			0	FFFFFFFF	FFFFFFFF	41
			1	FFFFFFFF	FFFFFFFF	42
			0	FFFFFFFF	FFFFFFFF	43
			1	FFFFFFFF	FFFFFFFF	44
			0	FFFFFFFF	FFFFFFFF	45
			1	FFFFFFFF	FFFFFFFF	46
			0	FFFFFFFF	FFFFFFFF	47
			1	FFFFFFFF	FFFFFFFF	48
			0	FFFFFFFF	FFFFFFFF	49
			1	FFFFFFFF	FFFFFFFF	50
			0	FFFFFFFF	FFFFFFFF	51
			1	FFFFFFFF	FFFFFFFF	52
			0	FFFFFFFF	FFFFFFFF	53
			1	FFFFFFFF	FFFFFFFF	54
			0	FFFFFFFF	FFFFFFFF	55
			1	FFFFFFFF	FFFFFFFF	56
			0	FFFFFFFF	FFFFFFFF	57
			1	FFFFFFFF	FFFFFFFF	58
			0	FFFFFFFF	FFFFFFFF	59
			1	FFFFFFFF	FFFFFFFF	60
			0	FFFFFFFF	FFFFFFFF	61
			1	FFFFFFFF	FFFFFFFF	62
			0	FFFFFFFF	FFFFFFFF	63
			1	FFFFFFFF	FFFFFFFF	64
SOP		Sync1 (#18)	0	FFFFFFFF	FFFFFFFF	65
			0	FFFFFFFF	FFFFFFFF	66
			0	FFFFFFFF	FFFFFFFF	67
			1	FFFFFFFF	FFFFFFFF	68
			1	FFFFFFFF	FFFFFFFF	69
		Sync1 (#18)	0	FFFFFFFF	FFFFFFFF	70
			0	FFFFFFFF	FFFFFFFF	71
			0	FFFFFFFF	FFFFFFFF	72
			1	FFFFFFFF	FFFFFFFF	73
			1	FFFFFFFF	FFFFFFFF	74
		Sync1 (#18)	0	FFFFFFFF	FFFFFFFF	75
			0	FFFFFFFF	FFFFFFFF	76
			0	FFFFFFFF	FFFFFFFF	77
			1	FFFFFFFF	FFFFFFFF	78
			1	FFFFFFFF	FFFFFFFF	79
		Sync2 (#11)	1	FFFFFFFF	FFFFFFFF	80
			0	FFFFFFFF	FFFFFFFF	81
			0	FFFFFFFF	FFFFFFFF	82
			0	FFFFFFFF	FFFFFFFF	83
			1	FFFFFFFF	FFFFFFFF	84

Function	Nibble	Symbol	Bits	CRC register transmitter	CRC register receiver	bit nr.
GoodCRC Header #0101	#1	#09	1	FFFFFFFF	FFFFFFFF	85
			0	FFFFFFFE	FFFFFFFF	86
			0	FB3EE24B	FFFFFFFF	87
			1	F2BCD921	FFFFFFFF	88
			0	E1B8AFF5	FFFFFFFF	89
	#0	#1E	0	E1B8AFF5	FFFFFFFF	90
			1	C7B0425D	FFFFFFFE	91
			1	8BA1990D	FB3EE24B	92
			1	13822FAD	F2BCD921	93
			1	27045F5A	E1B8AFF5	94
	#1	#09	1	27045F5A	E1B8AFF5	95
			0	4AC9A303	C7B0425D	96
			0	95934606	8BA1990D	97
			1	2FE791BB	13822FAD	98
			0	5FCF2376	27045F5A	99
	#0	#1E	0	5FCF2376	27045F5A	100
			1	BF9E46EC	4AC9A303	101
			1	7BFD906F	95934606	102
			1	F7FB20DE	2FE791BB	103
			1	EB375C0B	5FCF2376	104
CRC-32 = swapped and inverted EB375C0B = 2FC51328	#8	#12	0	EB375C0B	5FCF2376	105
			1	EB375C0B	BF9E46EC	106
			0	EB375C0B	7BFD906F	107
			0	EB375C0B	F7FB20DE	108
			1	EB375C0B	EB375C0B	109
	#2	#14	0	EB375C0B	EB375C0B	110
			0	EB375C0B	D2AFA5A1	111
			1	EB375C0B	A19E56F5	112
			0	EB375C0B	47FDB05D	113
			1	EB375C0B	8B3A7D0D	114
	#3	#15	1	EB375C0B	8B3A7D0D	115
			0	EB375C0B	12B5E7AD	116
			1	EB375C0B	21AAD2ED	117
			0	EB375C0B	4355A5DA	118
			1	EB375C0B	86AB4BB4	119
	#1	#09	1	EB375C0B	86AB4BB4	120
			0	EB375C0B	0D569768	121
			0	EB375C0B	1E6C3367	122
			1	EB375C0B	3CD866CE	123
			0	EB375C0B	79B0CD9C	124
	#5	#0B	1	EB375C0B	79B0CD9C	125
			1	EB375C0B	F7A0868F	126
			0	EB375C0B	EB8010A9	127
			1	EB375C0B	D3C13CE5	128
			0	EB375C0B	A343647D	129
	#C	#1A	0	EB375C0B	A343647D	130
			1	EB375C0B	4686C8FA	131
			0	EB375C0B	8D0D91F4	132
			1	EB375C0B	1A1B23E8	133
			1	EB375C0B	343647D0	134
	#F	#1D	1	EB375C0B	343647D0	135
			0	EB375C0B	686C8FA0	136
			1	EB375C0B	D0D91F40	137
			1	EB375C0B	A1B23E80	138
			1	EB375C0B	43647D00	139
	#2	#14	0	EB375C0B	43647D00	140
			0	EB375C0B	8209E7B7	141
			1	EB375C0B	0413CF6E	142
			0	EB375C0B	0CE6836B	143
			1	EB375C0B	1D0C1B61	144
EOP		#0D	1	EB375C0B	1D0C1B61	145
			0	EB375C0B	3A1836C2	146
			1	EB375C0B	70F17033	147
			1	EB375C0B	E1E2E066	148
			0	EB375C0B	C704DD7B	149

▨	Note: CRC transmitter is calculated over data bytes only, in casu marked nibbles, and calculation results are available one (bit-) clock later
▨	Note: CRC receiver is calculated over data bytes and received CRC bytes, in casu marked nibbles, and calculation results are available five (bit-) clocks later
🟩	Fixed residual

C Power Implementation Considerations

This section has been added as an informative guide for implementers of the PD specification. Expansion of USB Power Delivery to higher voltage and power levels with its AC coupled communication over V_{BUS} requires new design considerations to the power delivery system. Many of these may not have been as important in legacy USB systems, but should be considered when implementing the PD specification. Component labels within this Appendix are specific to the Appendix unless otherwise stated.

C.1 Managing Isolation Impedance

The isolation impedance outlined in the PD Specification Section 5.2.2 is used to block specific frequency bands and pass others in order to provide a communication path on the V_{BUS} conductor. This impedance will likely be inductive in nature (for example a 1µH inductor see Table 5-10). The following section describes potential problems that may arise due to this impedance. Measurement techniques to validate conducted noise will be discussed.

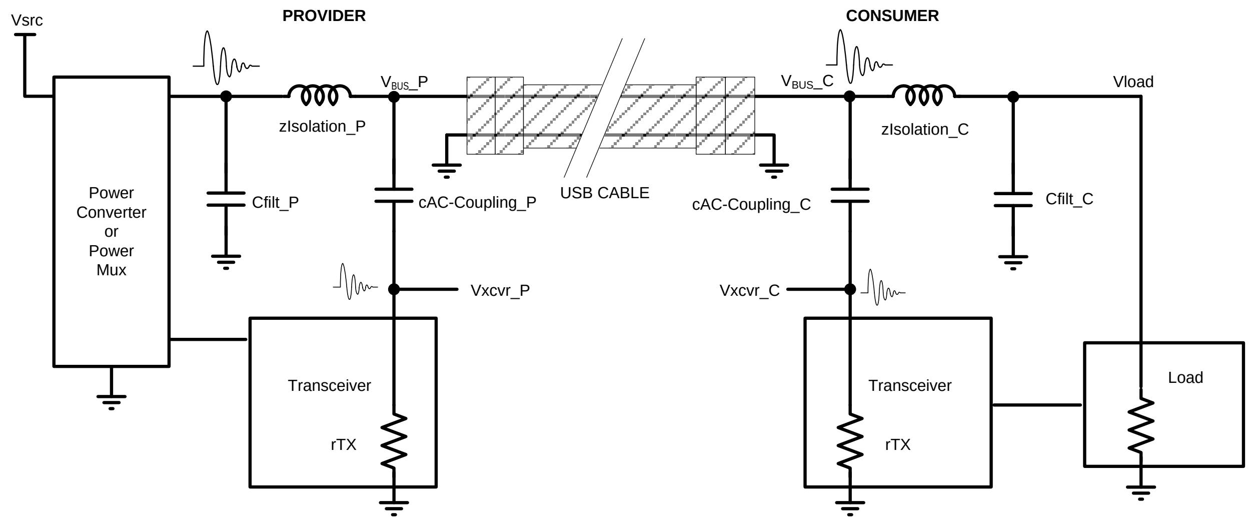


Figure C-1 Typical System Electrical Model

C.1.1 In-band fCarrier Spurious Noise

The output stage of a switch mode power converter can typically produce high frequency noise. This noise is differentiated from fundamental switching ripple. The noise tends to be a high frequency damped sinusoid occurring regularly at the converter power switch transitions. It is likely that some imlementations will produce noise that coincides with the fCarrier frequency band of the transceivers.

Figure C-2 shows parasitic elements within a basic synchronous buck power stage that can generate high frequency conducted noise which may interfere with the PD Transceiver. A basic synchronous buck stage is shown that is comprised of two power MOSFETs, HS_FET and LS_FET, an inductor, L_out, and output filter capacitor, C_filt. Also shown are the parasitic elements that create damped sinusoid noise that may be in contention with fCarrier. L_HSSRC, L_LSDRV, and CDS_LSFET are typically responsible for creating resonant noise on the VSW node. Other topologies may have different parasitic elements to consider. This waveform can couple to the output through C_ind and or circuit board parasitics and impose a voltage across R_Cfilr_ESR. This section is meant to help implementors identify and resolve noise problems in the design process before submitting PD designs to compliance testing.

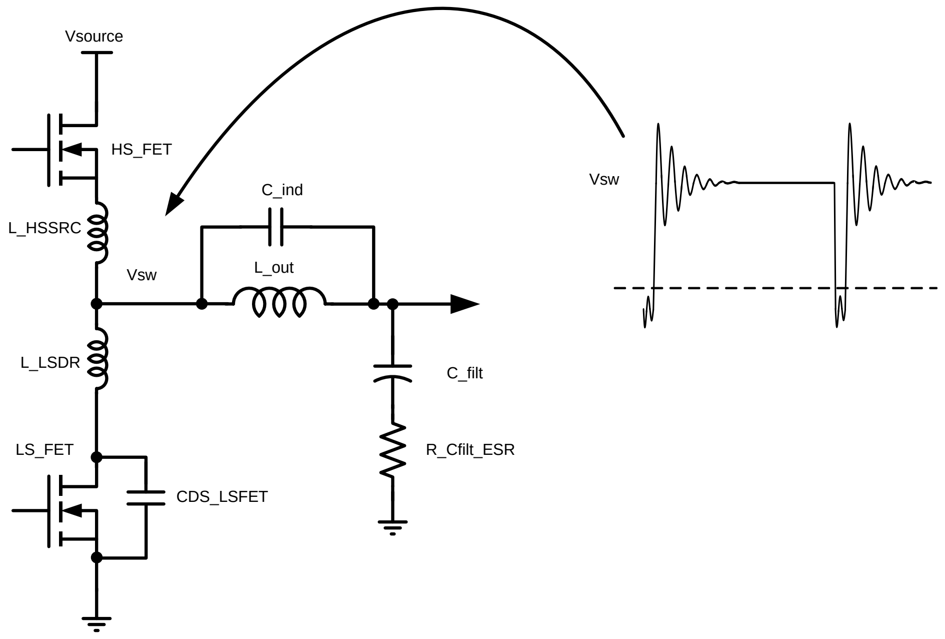


Figure C-2 Typical Synchronous Buck Power Stage with ParasiticsSpurious Noise Test Setup and Calibration

Measurement of in-band spurious noise from the power converter involves setting up measurinment equipment with the correct scales and impedances. The process involves calibrating a known carrier signal in a test setup containing the correct source and load impedances with the power converter connected, but not operational. Once the test setup is calibrated, the operational noise can be evaluated. In order to simplify this test and to match industry standard communication test equipment, 50 Ohm terminations and dBm scaled measurements will be used.

Measurement of in-band spurious noise will require the use of:

- An RF signal generator capable of providing a ***vTX*** nominal (150mVrms), ***fCarrier*** nominal(23MHz) carrier sine wave into a 50 ohm load.
- An oscilloscope with at least 200MHz Bandwidth.
- A spectrum analyzer with better than -80dBm measurement capability and 2MHz/div scale capability at 23MHz center frequency setting.
- A low noise environment with less than -60dBm of RF noise at 13 to 33MHz.

Figure C-3 shows the test setup which includes both the Provider and Consumer to be connected using PD USB cables. During calibration the power converter and load impedances should be in place with the power converter turned off.

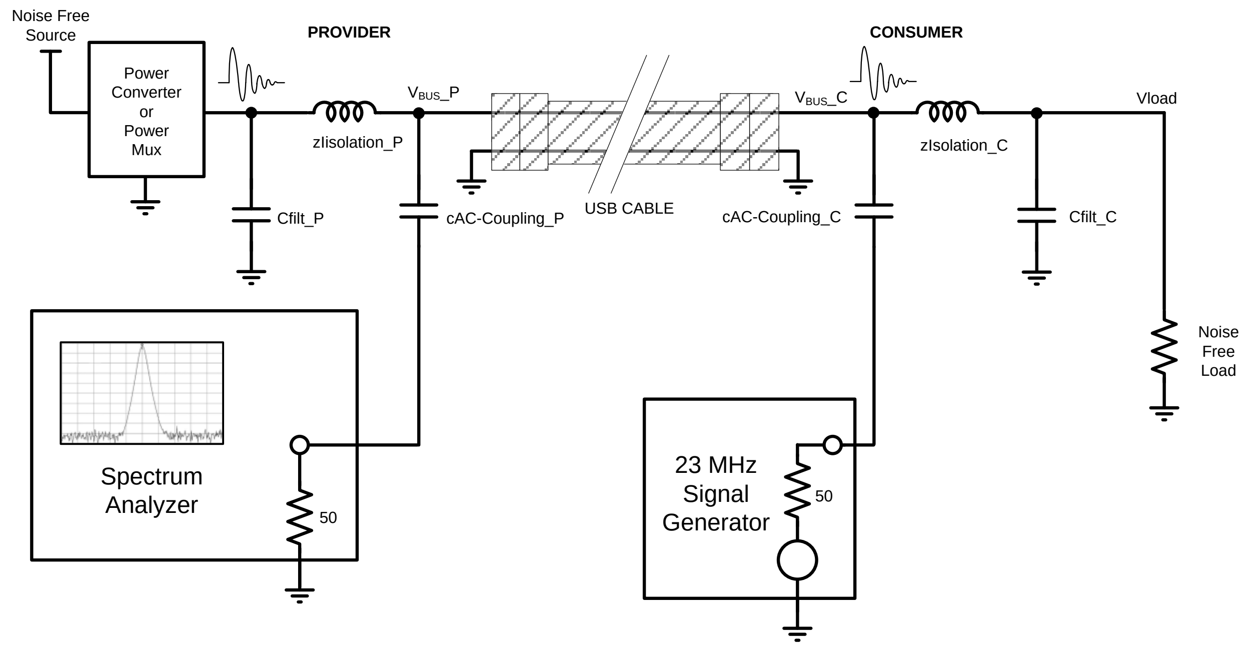


Figure C-3 Spurious Noise Measurement Test Setup

1) Using appropriately matched RF cabling configure the test setup as shown in Figure C-3 set the termination for both the signal generator and the spectrum analyzer to 50 ohms.
2) Set the signal generator to ((***vTX*** minimum x 1.414) x 2) Peak to Peak (~282mV) at ***fCarrier*** nominal (23MHz). Confirm the level using an oscilloscope.
3) Set the spectrum analyzer to ***fCarrier*** nominal (23MHz) Center Frequency, fCarrier nominal ± 10MHz span and set the Resolution Bandwidth (RBW) to 10 KHz.
4) Confirm that the measured level on the Spectrum Analyzer is approximately - 7 to -13 dBm (dBMilliwatts on 50 Ohms). Record the Baseline level (dBM Baseline).
5) Confirm that no spurious noise levels are present above -60dBm. If the noise floor is higher, it may be necessary to test in an environment with EMI shielding from radio noise sources.
6) Turn on the power converter and turn off the ***fCarrier*** nominal signal from the signal generator.
7) Confirm that the noise floor has not increased beyond (dBm_Baseline – ***snrSrc***). Reference Figure 7-5 for in-band noise. The (dBm_Baseline – ***snrSrc***) represents the 0dB level in Figure 7-5. Confirm that all measured noise falls below the curve.

The same measurement can be made from the perspective of the Sink using a quiet source. In this case the frequency generator is placed on the Provider side and the signal generator is placed on the Consumer side. The levels would be compared to Figure 7-7 in this case.

* Noise levels should be validated across expected line and load conditions including expected combinations of USB Cable types and lengths.

C.2 Connector Detach Transients

The presence of inductive elements ***zIsolation***_P and ***zIsolation***_C will cause transient voltages to be presented to the transceiver inputs. This section describes this behavior and some protection methods that can be considered

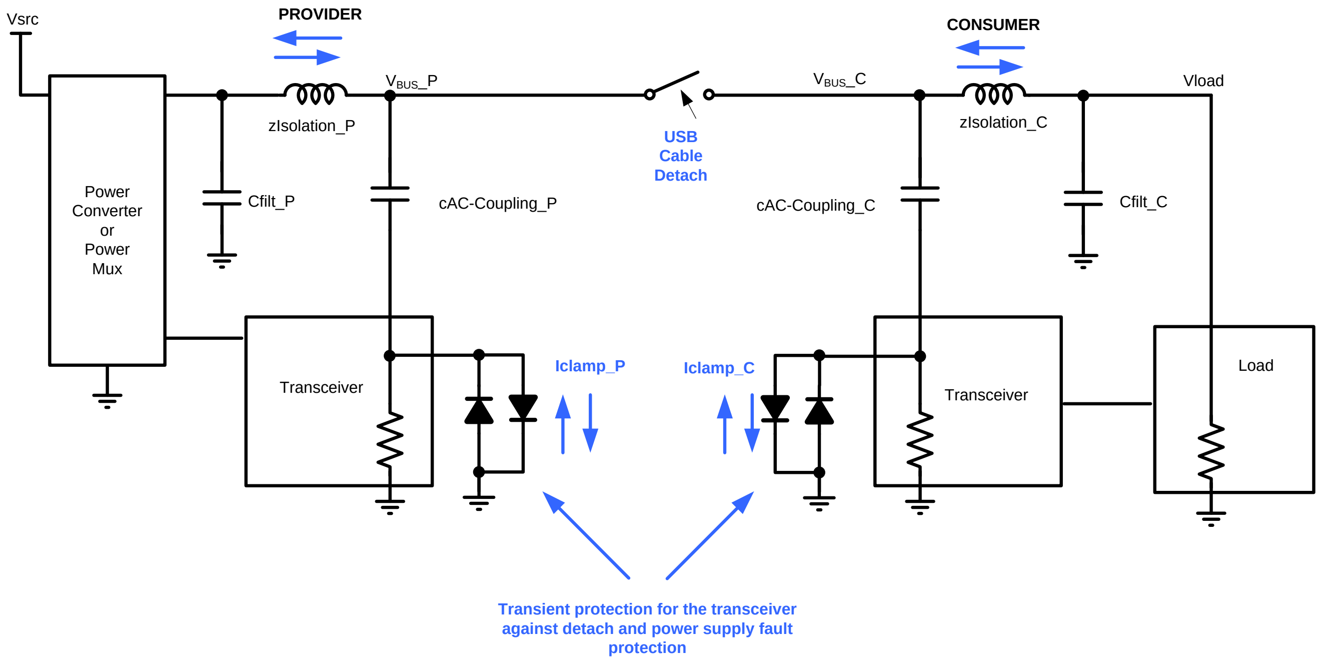


Figure C-4 Current Transients when Cable/Load Removed

As shown in Figure C-4, equal current is flowing in both isolation elements (***zIsolation***_P and ***zIsolation***_C) within the Provider and Consumer just prior to detach. Inductive elements resist changes in current and will force voltage transients on V_{BUS}_P and V_{BUs}_C terminals just following detach. These high dv/dt rate voltages will AC couple through the coupling capacitors ***cAC-Coupling***_P and ***cAC-Coupling***_C. A positive going voltage transient will be presented on the Provider side and a negative going voltage transient will be presented on the Provider side. Clamping elements should be included or the transceiver should be capable of absorbing the energy of the attach and detach events to prevent damage to the transceiver.

Figure C-5 Isolation Inductor Energy versus Load

Figure C-5 shows the total energy that could be delivered during a detach event with the example 1µH inductor where:

$$Energy = \frac{LI^2}{2}$$

An external or integrated clamp should be implemented to absorb this energy and limit the applied voltage at the transceiver input.

C.3 Closed Loop Stability Effects

Addition of the isolation inductor as well as cable inductance will affect the small signal power stage response of a switch mode power converter PD implementation.

C.3.1 Basic Power Stage Small Signal AC Model

Figure C-6 shows a simplified diagram of the small signal power stage AC response including the parasitic elements that should be considered. Power stage response refers to the voltage regulation system around which a feedback loop will be applied. This model does not include modulator gain or dc gain in the power stage as it is only meant to display exemplary parasitic poles and zeros in the typical system.

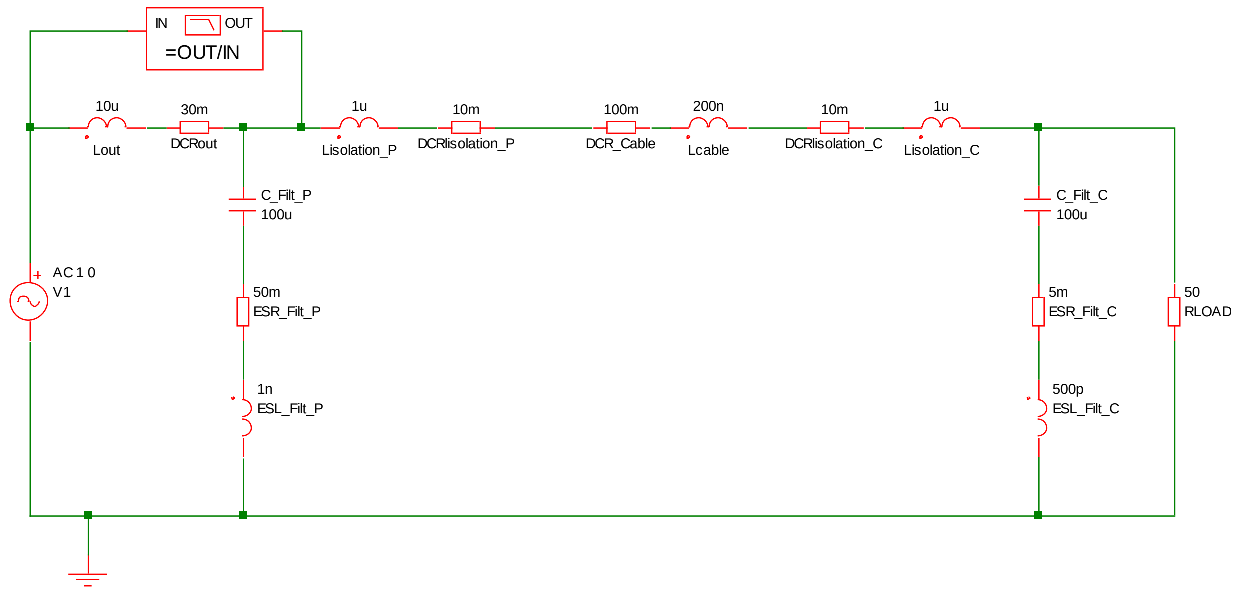


Figure C-6 Simplified Small Signal AC Model

The dominant pole of the power converter is set by Lout and C_Filt_P assuming the value of Lout is larger than (***zIsolation***_P + Lcable + ***zIsolation***_C). Parasitic ESR (Equivalent Series Resistance) and ESL (Equivalent Series Inductance) form high frequency zeros in the power stage response gain and phase. Phase distortion is introduced into the power stage response from (***zIsolation***_P + Lcable + ***zIsolation***_C) interacting with C_Filt_C. Secondary high frequency poles and zeros can degrade phase margin in the feedback loop of the power converter. These effects need to be modeled and accounted for when designing the feedback loop for stability and transient performance. Figure C-7 shows the resulting phase and gain impacts of the parasitics using the model in Figure C-6.

These traces compare the power stage phase and gain with and without isolation inductance. In this case, a phase boost is introduced which is not necessarily bad for compensation; however it is important for the designer to be aware of these effects to maintain predictable stability and transient response.

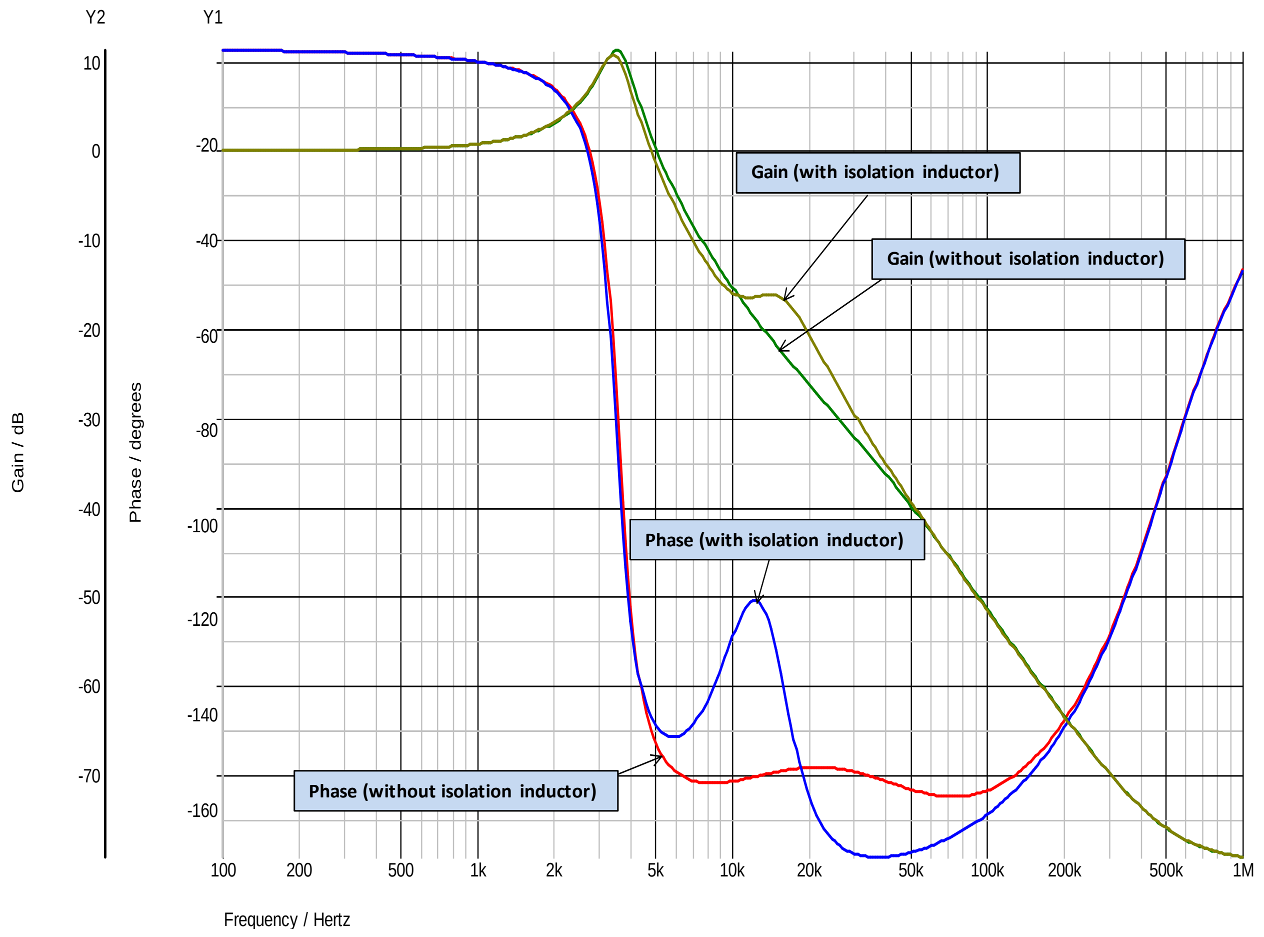


Figure C-7 Power Stage Phase And Gain with and without Isolation Inductors

It is important to note that worst case conditions tend to occur at:

- Light load (high Q Condition)
- Load capacitance highly tuned to series isolation inductance (high Q Condition)
- Main power converter LC pole (Lout, C_Filt_P) is selected too close to the parasitic LC ((***zIsolation***_P+ ***zIsolation***_C), C_Filt_C)

It is also important to note that selection of a C_Filt_C capacitor network that has parasitic ESR large enough to de-tune the dominant pole will help reduce the negative phase effects.

C.3.2 Feedback Past Isolation Inductor

It may be desirable to compensate the voltage loop past the Provider isolation inductor (***zIsolation***_P in order to eliminate the IR voltage drop due to the DC Resistance of the inductor. This further complicates the power stage response curve by placing the inductors reactance in series with the feedback path. Figure C-7 shows the same simple model with feedback measured past the isolation inductor.

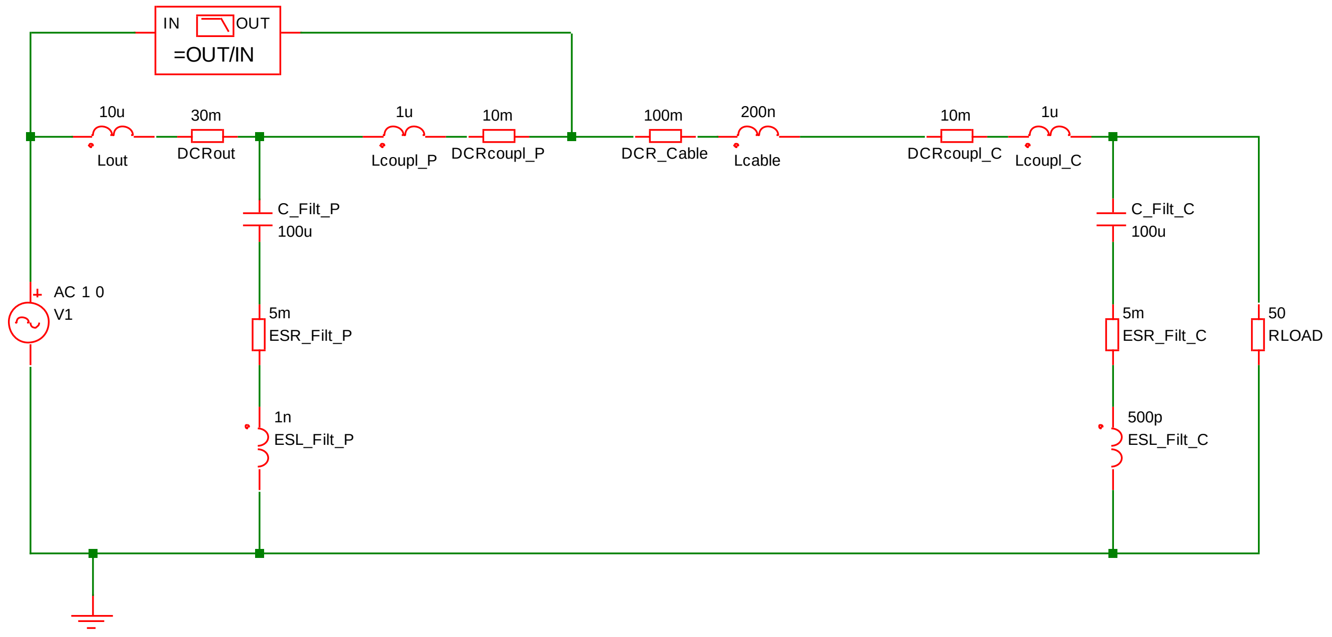


Figure C-8 Simplified Small Signal AC Model (Feedback before and after Inductor zIsolation_P)

Figure C-8 compares power stage response with compensation taken before and after the ***zIsolation***_P Provider isolation inductor. As can be seen in in Figure C-9, the phase and gain impacts are larger. A pole is created by the ***zIsolation***_P inductor that forces a phase reduction beyond 180 degrees which will have to be accounted for in the compensation network.

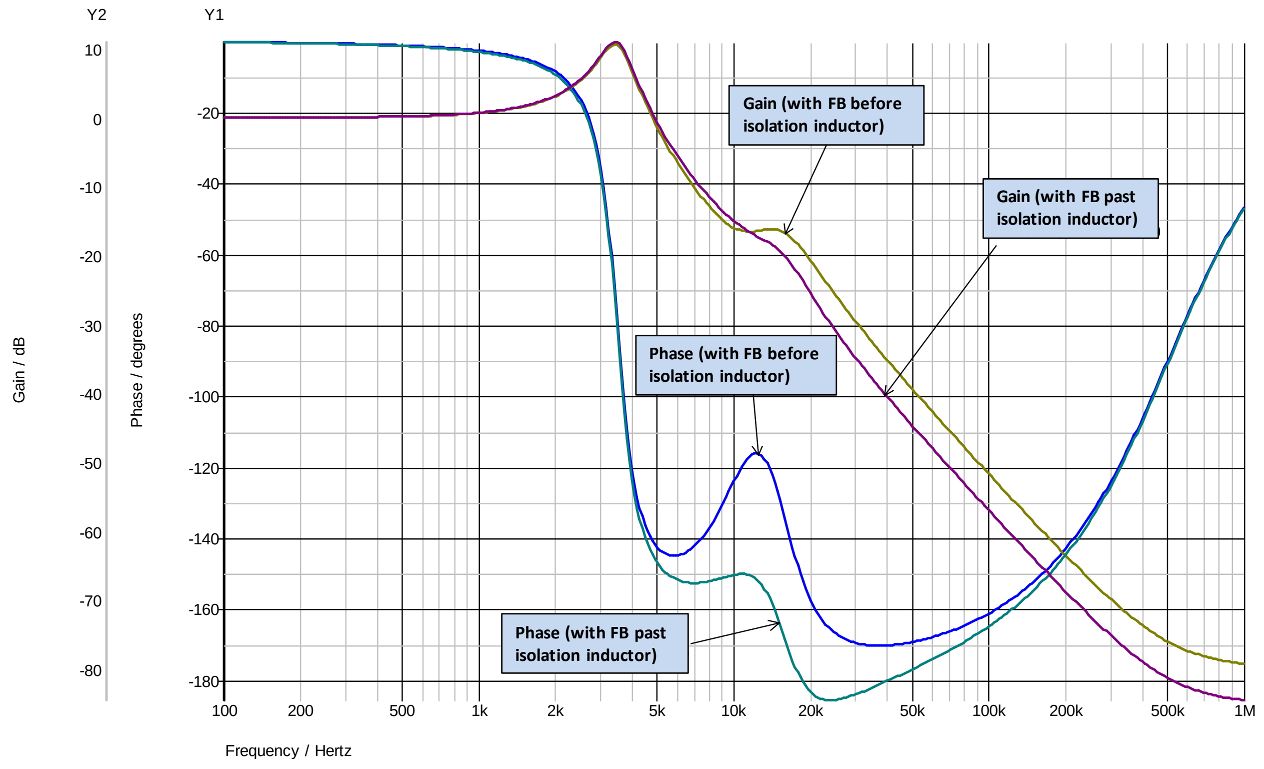
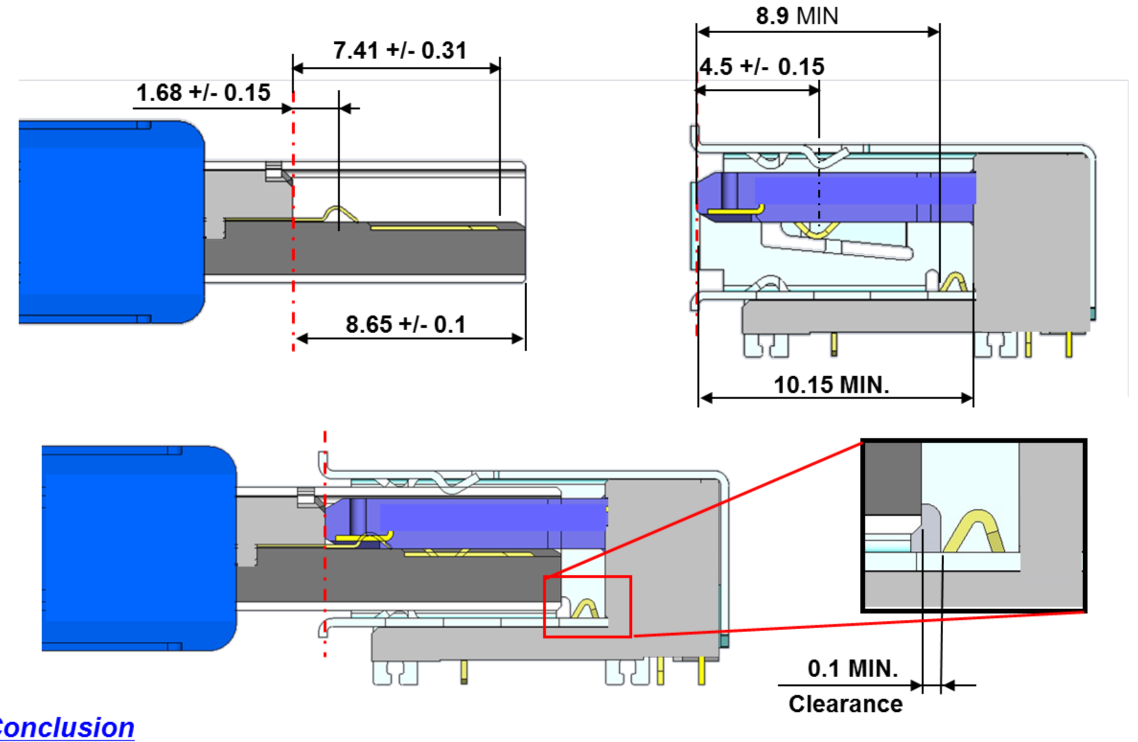


Figure C-9 Simplified Small Signal AC Model (Feedback before and after Inductor zIsolation_P)

D Standard-A Mating Illustrations

The following illustrations show the mating configurations between PD and non-PD Standard-A connectors. All dimensions are in mm.

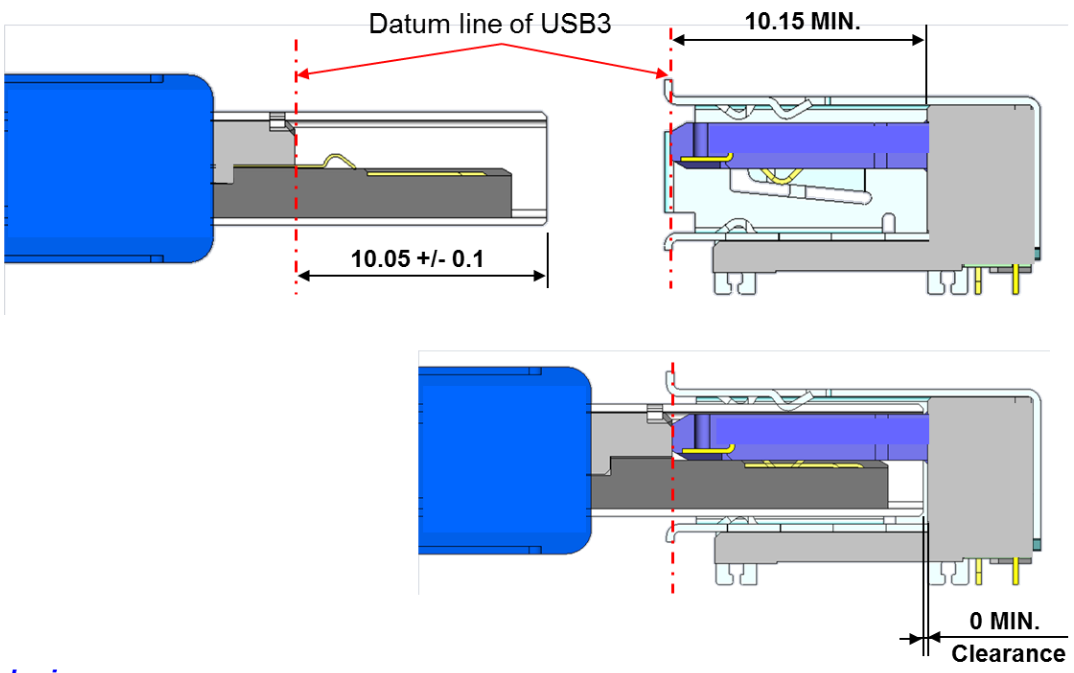


Conclusion

1) Plug does not contact to detect pins, then is identified as legacy USB3 plug.

2) The minimum clearance would be 0.1mm in the worst case which includes angled insertion case.

Figure D-1 USB 3.0 Standard-A Plug with USB 2.0 PD or 3.0 PD Standard-A Receptacle

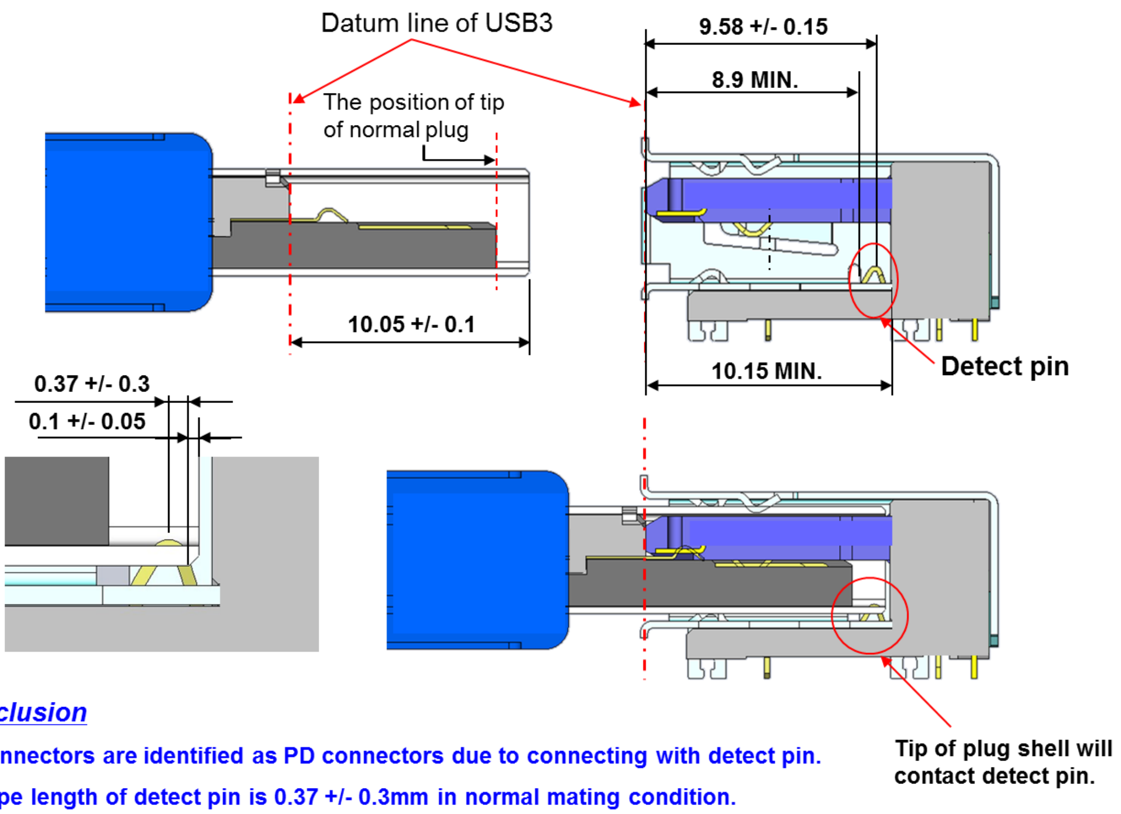


Conclusion

1) Legacy receptacle does not have detect pins, so PD plug will work as normal USB3 plug.

2) Wipe length will be same as legacy USB combination since there would be clearance (0 MIN.) between receptacle and plug after completing mating.

Figure D-2 USB 3.0 PD Standard-A Plug with USB 2.0 or 3.0 Standard-A Receptacle

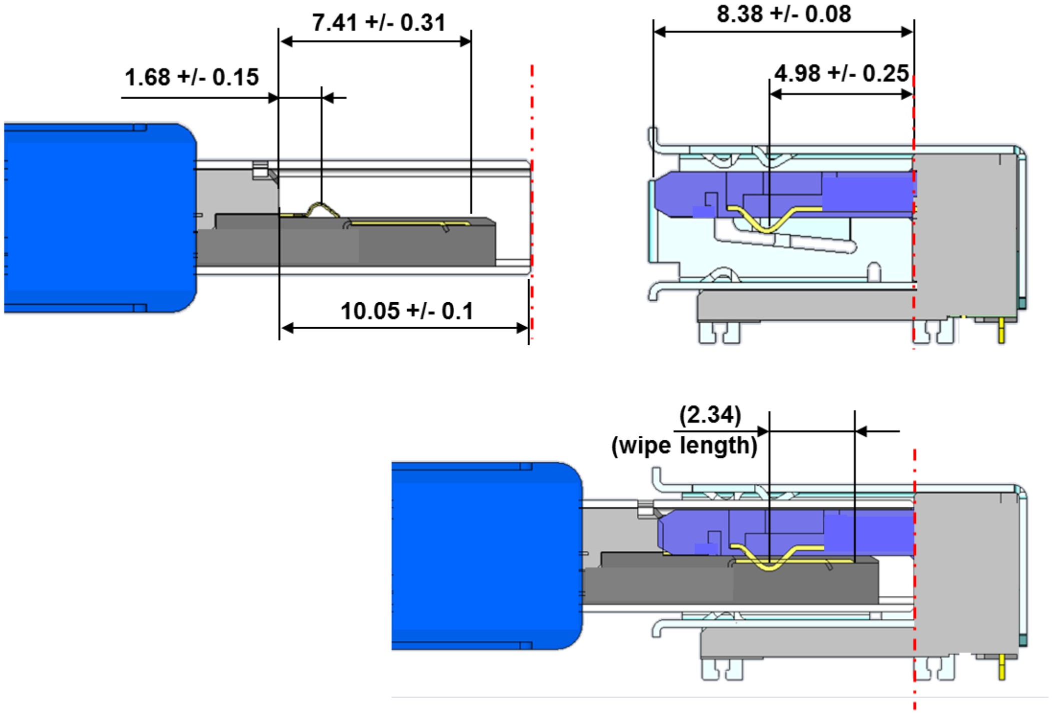


Conclusion

1) Connectors are identified as PD connectors due to connecting with detect pin.

2) Wipe length of detect pin is 0.37 +/- 0.3mm in normal mating condition.

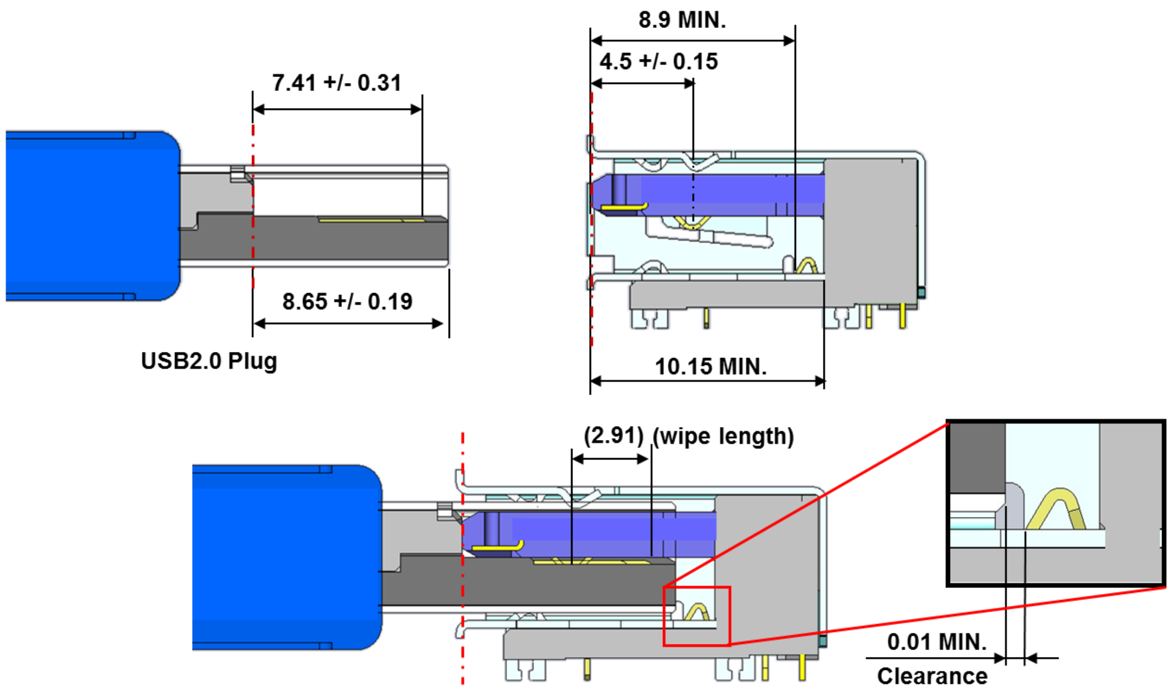
Figure D-3 USB 2.0 PD or 3.0 PD Standard-A plug with USB 2.0 PD or 3.0 PD Standard-A receptacle



Conclusion

USB2 legacy receptacle does not have detect pins, so PD plug will work as normal USB2 plug.

Figure D-4 USB 2.0 PD or 3.0 PD Standard-A Plug with USB 2.0 Standard-A Receptacle

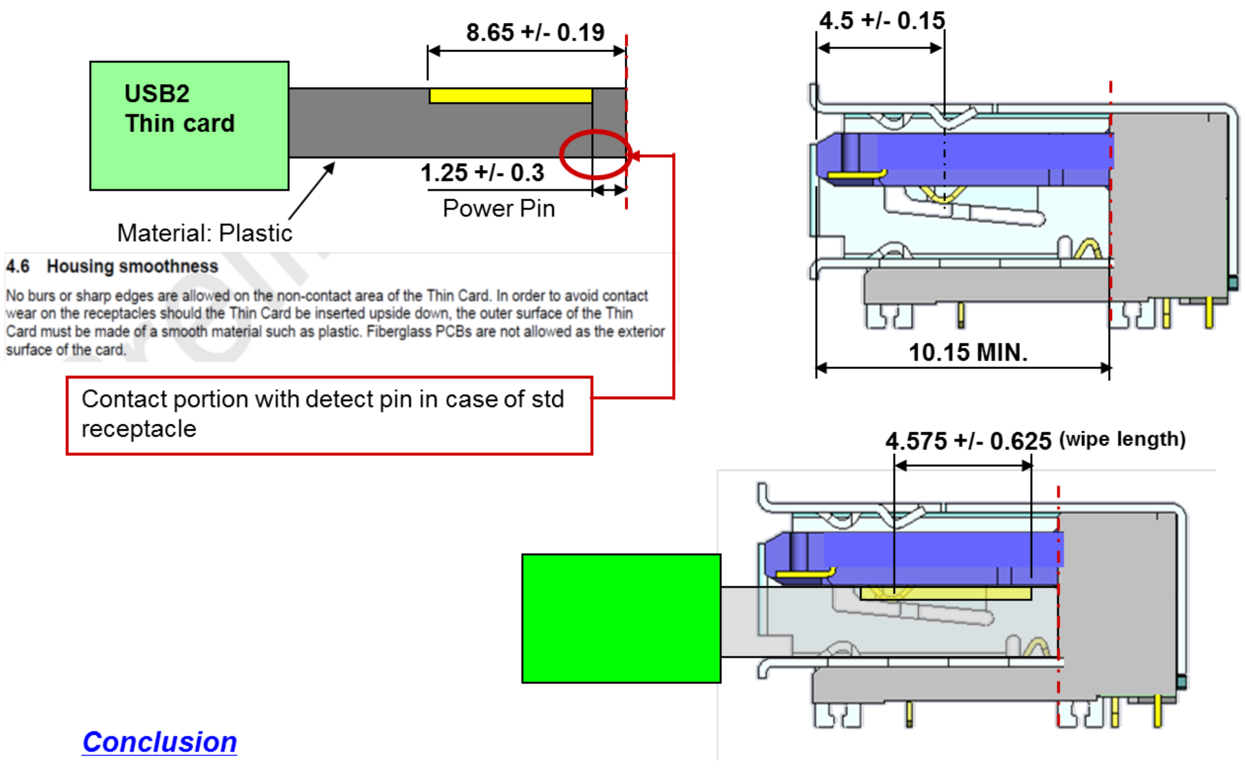


Conclusion

1) USB2 legacy plug does not contact to detect pins, so plug is identified as legacy plug then perform as USB2.

2) There would be 0.33mm clearance. The minimum clearance would be 0.01mm in the worst case which includes angled insertion case.

Figure D-5 USB 2.0 PD or 3.0 PD Standard-A Plug with USB 3.0 Standard-A Receptacle


8.65 +/- 0.19
USB2
Thin card
1.25 +/- 0.3
Power Pin
Material: Plastic
Contact portion with detect pin in case of std receptacle
4.5 +/- 0.15
10.15 MIN.
4.575 +/- 0.625 (wipe length)

4.6 Housing smoothness

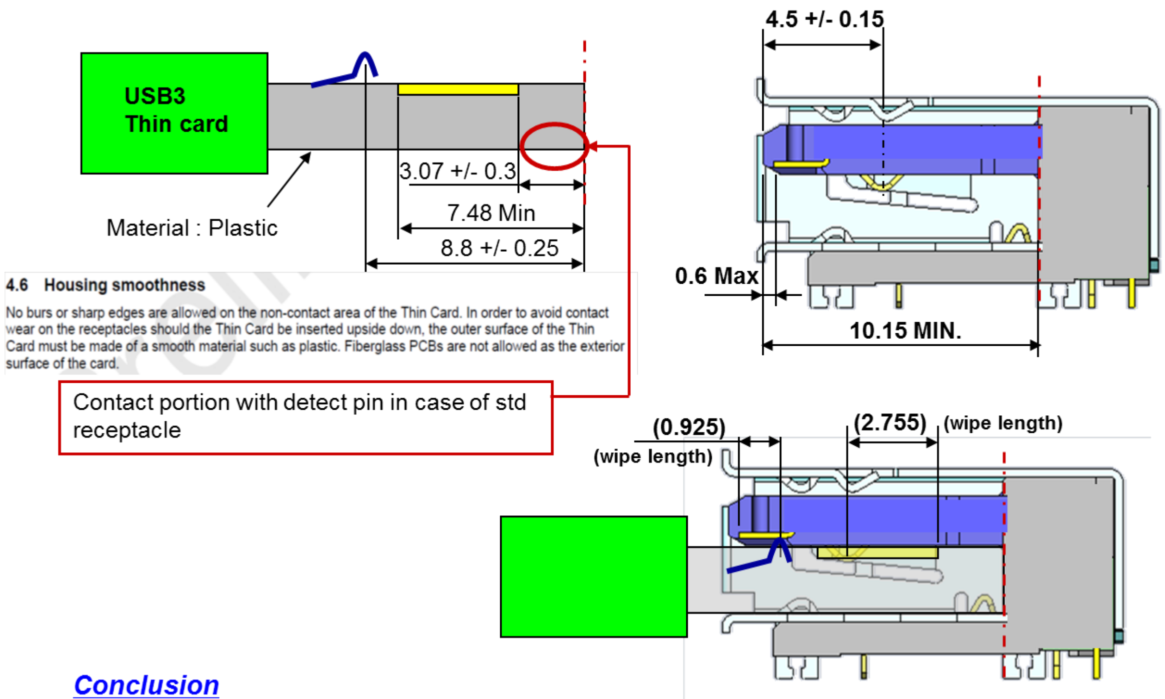
No burs or sharp edges are allowed on the non-contact area of the Thin Card. In order to avoid contact wear on the receptacles should the Thin Card be inserted upside down, the outer surface of the Thin Card must be made of a smooth material such as plastic. Fiberglass PCBs are not allowed as the exterior surface of the card.

Conclusion

1) USB2 thin card would be identified as thin card due to discontinuity with detect pin.

2) Detect pin would be displaced in case of standard receptacle. However it will not make electric connection as long as contact portion of thin card is insulator.

Figure D-6 USB 2.0 Thin Card with USB 2.0 PD or 3.0 PD Standard-A Receptacle



4.6 Housing smoothness

No burs or sharp edges are allowed on the non-contact area of the Thin Card. In order to avoid contact wear on the receptacles should the Thin Card be inserted upside down, the outer surface of the Thin Card must be made of a smooth material such as plastic. Fiberglass PCBs are not allowed as the exterior surface of the card.

Conclusion

1) USB3 thin card would be identified as thin card due to discontinuity with detect pin.

2) Detect pin would be displaced in case of standard receptacle. However it will not make electric connection as long as contact portion of thin card is insulator.

Figure D-7 USB 3.0 Thin Card with USB 2.0 PD or 3.0 PD Standard-A Receptacle

E Physical Layer Informative Material

This section contains informative material about the Physical Layer.

E.1 Squelch Budget

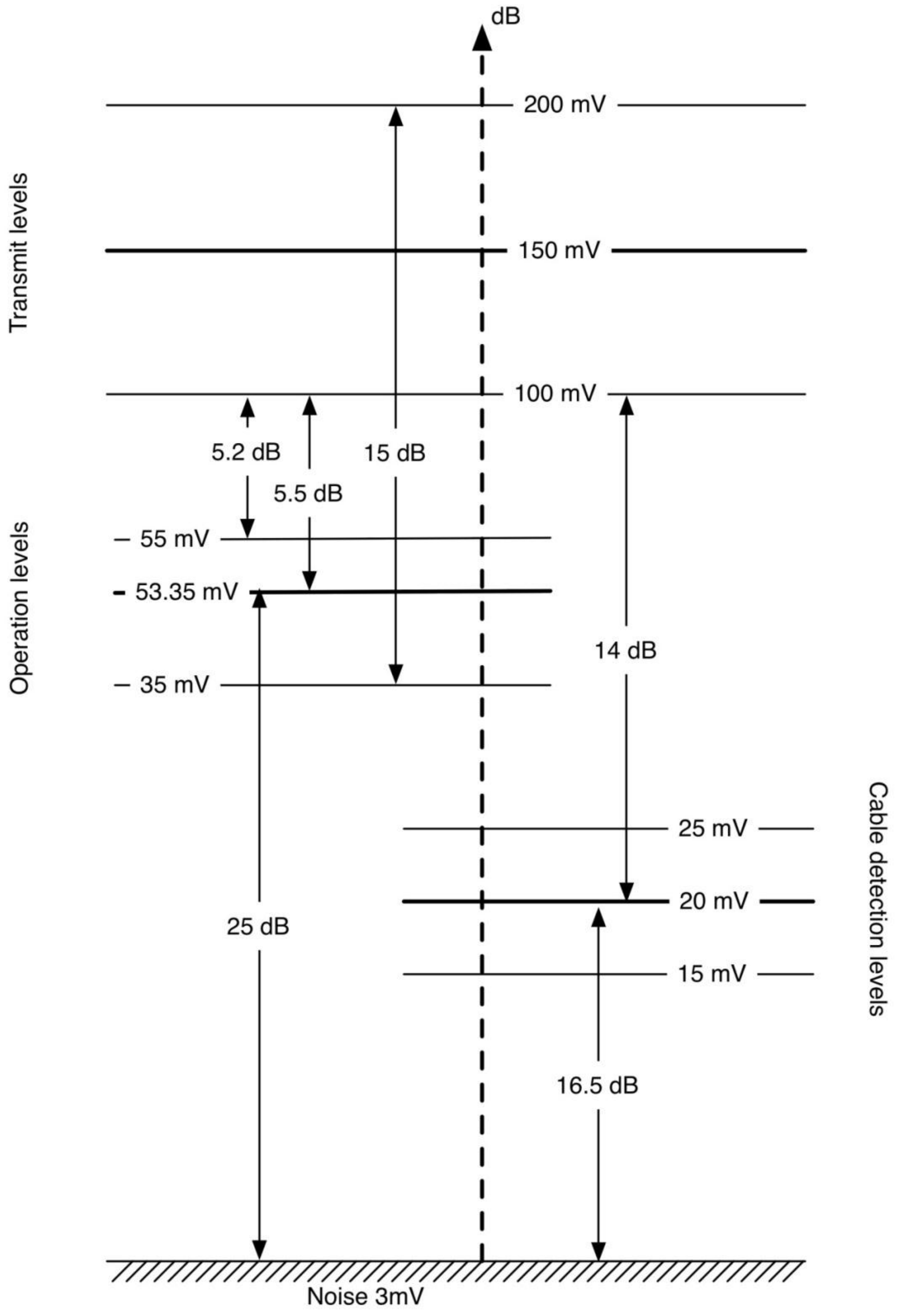


Figure E-1 Squelch Budget

Figure E-1 shows the signal levels in mV(rms) needed to illustrate the squelch budget for normal operation (***vSquelchOperating***) and cable-type detection operation (***vSquelchDetecting***). The transmission level of the signal is between 100mV and 200mV, or at least 26.5dB above a 3mV noise floor. During normal operation the nominal SNR is 25dB, packets with a signal level below 55mV may be discarded, and packets with a signal level below 35mV are discarded. This allows for at least 5.2dB of attenuation in the channel and at most 15dB. During plug-type detection the nominal SNR is 16.5dB, a signal level below 25mV may be discarded, and a signal level below 15mV must be discarded. This allows for at least 12dB of attenuation in the channel and at most 22.5dB.