
STM32 USB-PD (Power Delivery)
software expansion for STM32Cube

Introduction

This document describes the STM32 USB-PD (Power Delivery) Expansion Package for STM32Cube, referenced as X-CUBE-USB-PD.

The USB Type-C is the newest USB connector ecosystem, it addresses the evolving needs of platform and devices, while retaining the functional benefits of USB.

The USB Power Delivery protocol is embedded in USB Type-C connectors, resulting in easy connection or disconnection of USB cables, and ensuring much more than data transfer. This protocol enables to carry more than the regular 5 V / 1.5 A, with a maximum power supply of 100 W.

X-CUBE-USB-PD is a USB-IF certified Expansion Package and consists of libraries, drivers, sources, APIs and application examples running on STM32F0 Series microcontrollers acting as USB Power Delivery controllers or USB Type-C port managers (TCPMs). Examples are provided to help to develop applications based on USB-PD (Provider, Consumer, DRP, Dual Port and VDM).

The "Core" of the stack is delivered in library format while the "Device" part, in open source format, offers high level of flexibility to match the design considerations.

This Expansion Package supports various hardware implementations covering most of the typical USB Type-C use-cases at optimized cost:

- Legacy solution: STM32F0 devices with discrete Analog Front End PHY
 - Based on Google reference design
 - USB-PD 2.0 only – 1 port
 - Mainly in captive cable configuration, for example USB Type-C to Display Port adapter
 - Hardware development platform P-NUCLEO-USB001
- STM32F0 with STUSB1602 USB Type-C controller
 - For new design or to upgrade existing design based on STM32F0 devices only
 - USB-PD 2.0/3.0+PPS compliant solution
 - Hardware development platform P-NUCLEO-USB002
- Standardized TCPM/TCPC solution for any STM32F
 - Ideal solution to upgrade legacy design based-on any STM32F0/F4 with USB-C
 - Lowest memory footprint and easy porting within the Cortex™-M series
 - USB-PD 2.0/3.0+PPS compliant, multi-port
 - Tested with TCPC controller from On-SEMI FUSB307B: *STM32F072 MCU Type-Port Manager (TCPM) with ON-SEMI FUSB307B Type-C Port Controller (TCPC) evaluation board (DB3623)*



Contents

1	Overview	9
1.1	Acronyms and abbreviations	9
1.2	References	9
2	USB-C PD architecture	10
2.1	Architecture overview	10
2.2	Specific case of TCPM/TCPC architecture	11
2.3	USB-PD layers	12
2.4	Message flow	14
2.5	Data flow	16
3	USB-PD library description	17
3.1	Overview	17
3.2	Features	19
3.3	Library architecture	20
3.4	Hardware related components (for P-NUCLEO-USB001)	21
4	USB-PD library programming guidelines	24
4.1	Description of available configurations	24
4.2	Library initialization	25
4.3	USB-PD core stack library functions	25
4.4	USB-PD core stack	26
4.5	USB-PD device components	26
5	Atomic message sequencing	27
5.1	Acronym	27
5.2	This chapter describes different scenarios:	27
5.3	SRC AMS	28
5.3.1	SRC: Power Negotiation	28
5.3.2	SRC: Receive a request message, accepted by Source	29
5.3.3	SRC: Power Role Swap - initiated by source, accepted by port partner	30
5.3.4	SRC: Power Role Swap – initiated by source, not supported by port partner	30

5.3.5	SRC: Power Role Swap – initiated by port partner, accepted by source	31
5.3.6	SRC: Power Role Swap – initiated by port partner, rejected by source	31
5.3.7	SRC: Hard Reset – initiated by port partner	32
5.3.8	SRC: Hard Reset – initiated by Source	33
5.3.9	SRC: VCONN Support	33
5.4	SNK AMS	34
5.4.1	SNK: Power Negotiation	34
5.4.2	SNK: Send a request message, accepted by port partner	35
5.4.3	SNK: Power Role Swap, initiated by Sink, accepted by port partner	36
5.4.4	SNK: Power Role Swap, initiated by Sink, not supported by port partner	36
5.4.5	SNK: Power Role Swap, initiated by port partner, accepted by Sink	37
5.4.6	SNK: Power Role Swap – initiated by port partner, rejected by Sink	37
5.4.7	SRC: Hard Reset – initiated by sink	38
5.4.8	SRC: Hard Reset – initiated by port partner	39
5.5	Generic AMS	40
5.5.1	Soft Reset TX	40
5.5.2	Soft Reset RX	40
5.5.3	Data Role SWAP TX	41
5.5.4	Data Role SWAP Rx	41
5.5.5	GET extended capa TX	42
5.6	VCONN Swap	42
5.6.1	Initiated VCONN Swap, rejected by port partner	42
5.6.2	VCONN Swap initiated by VCONN source, accepted by port partner	43
5.6.3	VCONN Swap initiated by VCONN sink, accepted by port partner	43
5.6.4	VCONN Swap initiated by port partner, accepted by VCONN sink	44
5.6.5	VCONN Swap initiated by port partner, accepted by VCONN Source	44
5.7	VDM sequences	45
5.7.1	VDM Discovery Identify, requested by DFP	45
5.7.2	VDM Discovery Identity, received by UFP	45
5.7.3	VDM Discovery SVID, requested by DFP	46
5.7.4	VDM Discovery SVID, received by UFP	46
5.7.5	VDM Discovery Modes, requested by DFP	47
5.7.6	VDM Discovery Mode, received by UFP	47
5.7.7	VDM Enter Mode, requested by DFP	48
5.7.8	VDM Enter Mode, received by UFP	48
5.7.9	VDM SPECIFIC, requested by DFP	49

5.7.10	VDM SPECIFIC, received by UFP	49
5.8	TCPM	50
5.8.1	TCPC initialization	50
5.8.2	SNK connection	50
5.8.3	SRC connection	51
5.8.4	TCPC Transmit message	52
5.8.5	TCPC Receive message	53
5.8.6	SRC: HARD RESET Reception	54
6	Examples description	55
6.1	Hardware description	55
6.2	USB-PD provider	56
6.2.1	Example setup	56
6.2.2	Application description	56
6.3	USB-PD provider (with CLI support)	57
6.4	USB-PD provider (with VDM support)	57
6.4.1	USB-PD provider (Low footprint version)	58
6.4.2	Example setup	58
6.4.3	Application description	59
6.5	USB-PD consumer	59
6.5.1	Example setup	59
6.5.2	Application description	60
6.6	USB-PD consumer (with CLI support)	60
6.7	USB-PD consumer (with VDM support)	60
6.8	USB-PD consumer DRP	61
6.8.1	Example setup	61
6.8.2	Application description	62
6.9	USB-PD Dual Port	62
6.9.1	Example setup	63
6.9.2	Application description	63
6.10	USB-PD DRP (TCPM/TCPC architecture)	64
6.10.1	Example setup	64
6.10.2	Application description	65
7	Memory footprint	66

8	Frequently asked questions (FAQs)	69
9	Revision history	70

List of tables

Table 1.	List of acronyms	9
Table 2.	Use of different IPs	21
Table 3.	GPIOs used by Port0	21
Table 4.	GPIOs used by Port1	22
Table 5.	Interrupt priorities	23
Table 6.	USB-PD - Provider memory footprint (in Bytes)	66
Table 7.	USB-PD - Consumer memory footprint (in Bytes)	66
Table 8.	USB-PD - Dual role port memory footprint (in Bytes)	67
Table 9.	USB-PD - Provider (Low Footprint version) memory footprint	67
Table 10.	Document revision history	70

List of figures

Figure 1.	USB power delivery architecture	10
Figure 2.	USB Type-C port manager to USB Type-C port controller interface	11
Figure 3.	ST figure architecture	12
Figure 4.	USB-PD - Messages flow	15
Figure 5.	USB-PD stack architecture	16
Figure 6.	Project files	18
Figure 7.	USB-PD stack architecture (for P-NUCLEO-USB001)	20
Figure 8.	color conventions for following slides	27
Figure 9.	SRC: Power Negotiation	28
Figure 10.	SRC: receive a request message, accepted by Source	29
Figure 11.	SRC: Power Role Swap - initiated by source, accepted by port partner	30
Figure 12.	SRC: Power Role Swap – initiated by source, not supported by port partner	30
Figure 13.	SRC: Power Role Swap – initiated by port partner, accepted by source	31
Figure 14.	SRC: Power Role Swap – initiated by port partner, rejected by source	31
Figure 15.	SRC: Hard Reset – initiated by port partner	32
Figure 16.	SRC: Hard Reset – initiated by Source	33
Figure 17.	SRC: VCONN Support	33
Figure 18.	SNK: Power Negotiation	34
Figure 19.	SNK: Send a request message, accepted by port partner	35
Figure 20.	SNK: Power Role Swap, initiated by Sink, accepted by port partner	36
Figure 21.	SNK: Power Role Swap, initiated by Sink, not supported by port partner	36
Figure 22.	SNK: Power Role Swap, initiated by port partner, accepted by Sink	37
Figure 23.	SNK: Power Role Swap, initiated by port partner, rejected by Sink	37
Figure 24.	SRC: Hard Reset – initiated by sink	38
Figure 25.	SRC: Hard Reset – initiated by port partner	39
Figure 26.	Soft Reset TX	40
Figure 27.	Soft Reset RX	40
Figure 28.	Data Role SWAP TX	41
Figure 29.	Data Role SWAP Rx	41
Figure 30.	GET extended capa TX	42
Figure 31.	Initiated VCONN Swap, rejected by port partner	42
Figure 32.	VCONN Swap initiated by VCONN source, accepted by port partner	43
Figure 33.	VCONN Swap initiated by VCONN sink, accepted by port partner	43
Figure 34.	VCONN Swap initiated by port partner, accepted by VCONN sink	44
Figure 35.	VCONN Swap initiated by port partner, accepted by VCONN Source	44
Figure 36.	VDM Discovery Identify, requested by DFP	45
Figure 37.	VDM Discovery Identity, received by UFP	45
Figure 38.	VDM Discovery SVID, requested by DFP	46
Figure 39.	VDM Discovery SVID, received by UFP	46
Figure 40.	VDM Discovery Modes, requested by DFP	47
Figure 41.	VDM Discovery Mode, received by UFP	47
Figure 42.	VDM Enter Mode, requested by DFP	48
Figure 43.	VDM Enter Mode, received by UFP	48
Figure 44.	VDM SPECIFIC (ex: DP STATUS), requested by DFP	49
Figure 45.	VDM SPECIFIC (ex: DP STATUS), received by UFP	49
Figure 46.	TCPC initialization	50
Figure 47.	SNK connection	50
Figure 48.	SRC connection	51

Figure 49.	TCPC Transmit message	52
Figure 50.	TCPC Receive message	53
Figure 51.	SRC: HARD RESET Reception	54
Figure 52.	STM32F072RB Nucleo with P-NUCLEO-USB001 shield.....	55

1 Overview

This document describes how to use the USB-PD library for regular use and to create a customized application. It covers the following topics to ease the use of the library:

- USB-PD Standard overview
- USB-PD library architecture
- USB-PD Stack usage description

1.1 Acronyms and abbreviations

Table 1. List of acronyms

Term	Meaning
API	Application Programming Interface
CAD	Cable detection module
CLI	Command line interface
DFP	Downstream facing port
DPM	Device policy manager
DRP	Dual role port
FW	Firmware
HW	Hardware
PD	Power delivery
PE	Policy engine
PRL	Protocol layer
TCPC	USB Type-C port controller
TCPCi	USB Type-C port controller interface
TCPM	USB Type-C port manager
UFP	Upstream Facing Port
USB	Universal Serial Bus
VDM	Vendor Defined Messages

1.2 References

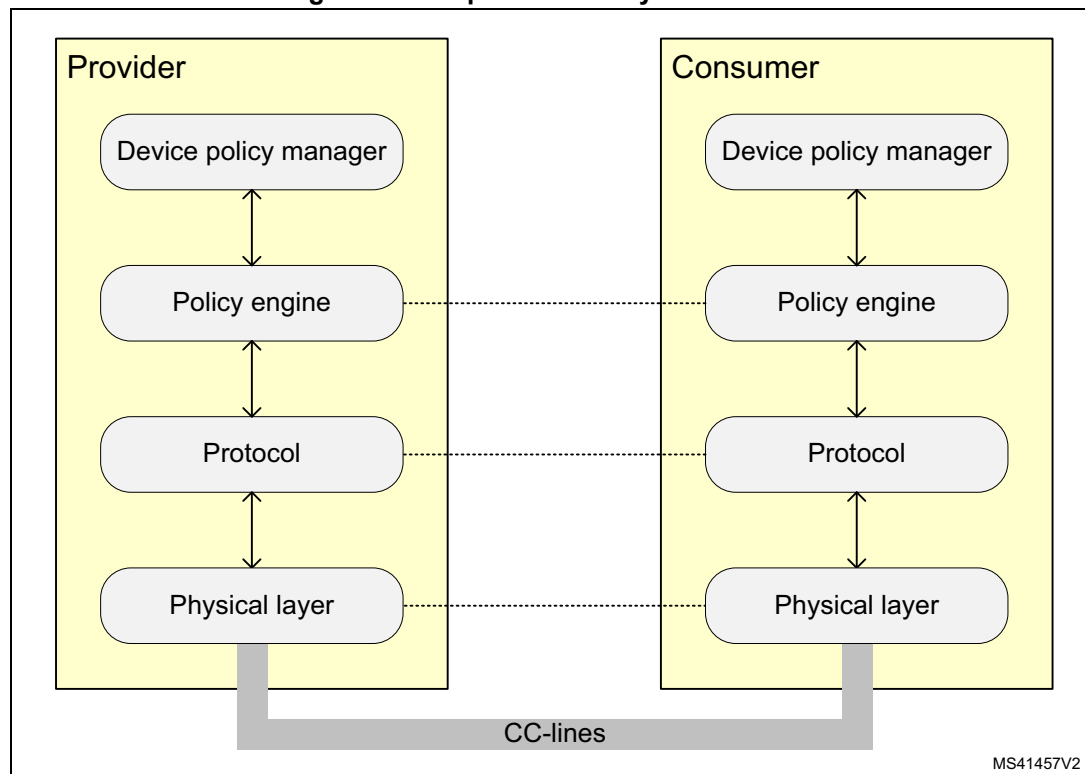
- Universal Serial Bus Power Delivery Specification, Revision 3.0, Version 1.3, January 12, 2017
- Universal Serial Bus Type-C Cable and Connector Specification 1.3, July 14, 2017
- Universal Serial Bus Type-C TM Port Controller Interface Specification, Revision 1.0, Version 1.2, November 2016

2 USB-C PD architecture

2.1 Architecture overview

The USB power delivery specification document defines the communicating layers of a PD device (either provider or consumer) as shown in [Figure 1](#).

Figure 1. USB power delivery architecture



A PD3.0-capable device is assumed to be made up of at least one port, which can

- sink power (a consumer), source power (a provider), or be able to toggle between the two roles (dual role power)
- optionally communicate via USB
- communicate using SOP Packets
- optionally communicate using SOP Prime packets.

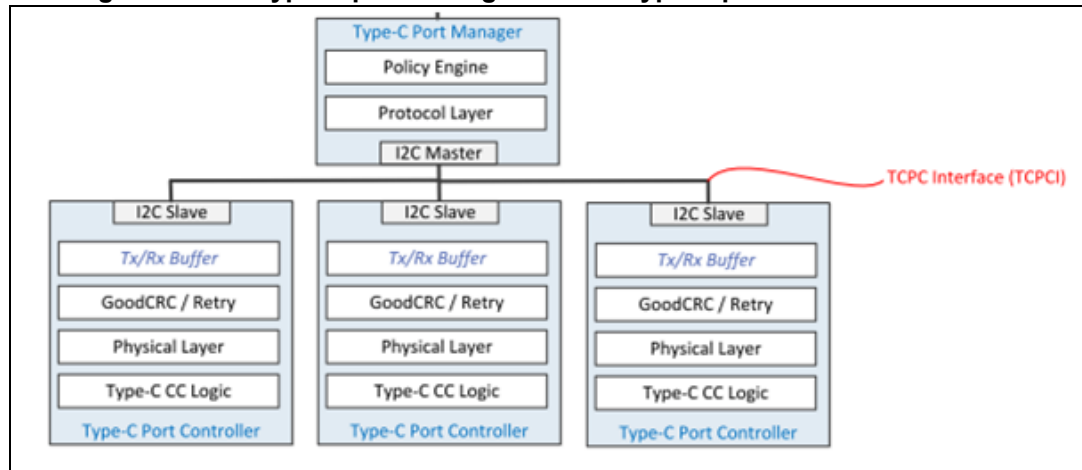
Where USB products support USB Power Delivery protocols an USB DFP is initially a Source and an USB UFP is initially a Sink, although USB-PD enables the Source/Sink and the DFP/UFP roles to be swapped.

Note: There is only one Source port and one Sink port in each PD communication between port partners.

2.2 Specific case of TCPM/TCPC architecture

Implementation of USB-PD function may also be achieved in a TCPM/TCPC architecture, with a standardized interface between USB Type-C port manager (TCPM) and a simple USB Type-C port controller (TCPC) with the goal of easing USB Type-C power Delivery implementations.

Figure 2. USB Type-C port manager to USB Type-C port controller interface



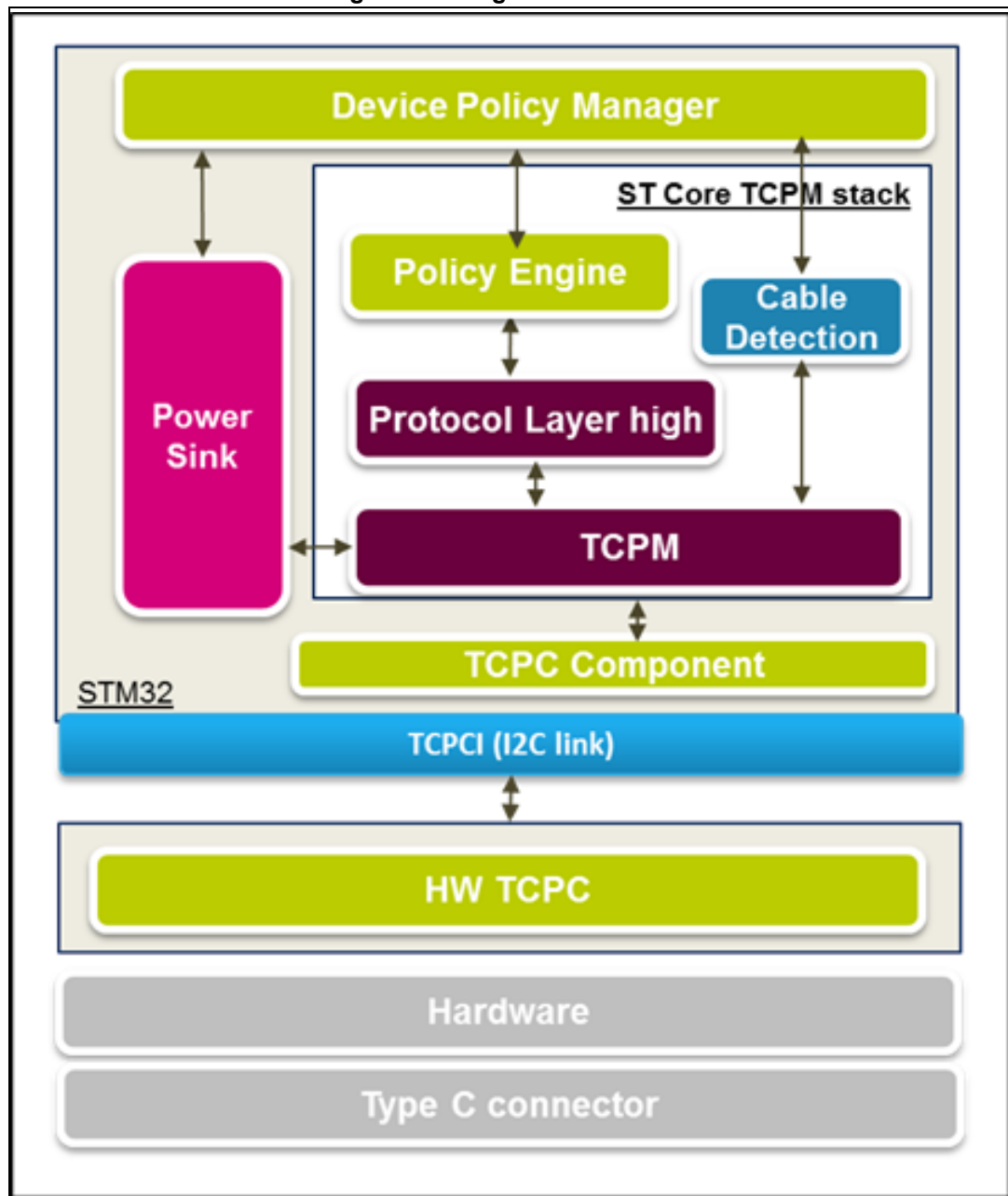
TCPC Interface (TCPCi) is using an I2C link and an alert pin. USB Type-C Port Controller Interface Specification (TCPCi) defines interface between the TCPM and the TCPC(s). In this architecture, one TCPM may be used to drive multiple TCPCs.

Protocol (PRL) functionalities are shared between TCPM and TCPC. TCPC should manage:

- GoodCRC management
- Retransmission

ST Firmware architecture is described in [Figure 3](#):

Figure 3. ST figure architecture



2.3 USB-PD layers

- **Device Policy Manager (DPM)**
The DPM role is to manage the power used by one USB Power Delivery port. It delivers power to a consumer in case of a provider application, or asks for it in case of a consumer application.
The DPM also allows exchanging VDM messages once an Explicit Contract is

established. The Device Policy Manager is the upper layer of the USB-C Power Delivery Stack.

- **Policy Engine (PE) layer**
The Policy engine (PE) role is to drive the message sequences according to the sent message and to its expected response. It allows negotiating power, establishing an Explicit Contract for the power exchange.
The acceptance or the refusal of a request depends on the response of the DPM towards a specific power profile.
The PE also handles the Vendor Defined Messages flow, allowing to discover, enter or exit specified modes according to those supported by both provider and consumer sides.
- **Protocol (PRL) layer**
The PRL layer role is to drive messages construction, transmission and reception independently of their nature. It permits to check the message flow to detect communication errors.
The PRL layer is a wrapping layer between the PE and the PHY.
- **PHY layer**
The PHY layer is responsible for sending and receiving messages across the CC wire. It consists of a transceiver that superimposes a signal on the wire. It is responsible for managing data on the wire and for collision avoidance and detects errors in the messages using a CRC.

Note: For more Information about the USB Power Delivery protocol, refer to the official specification document mentioned in [Section 1.2](#).

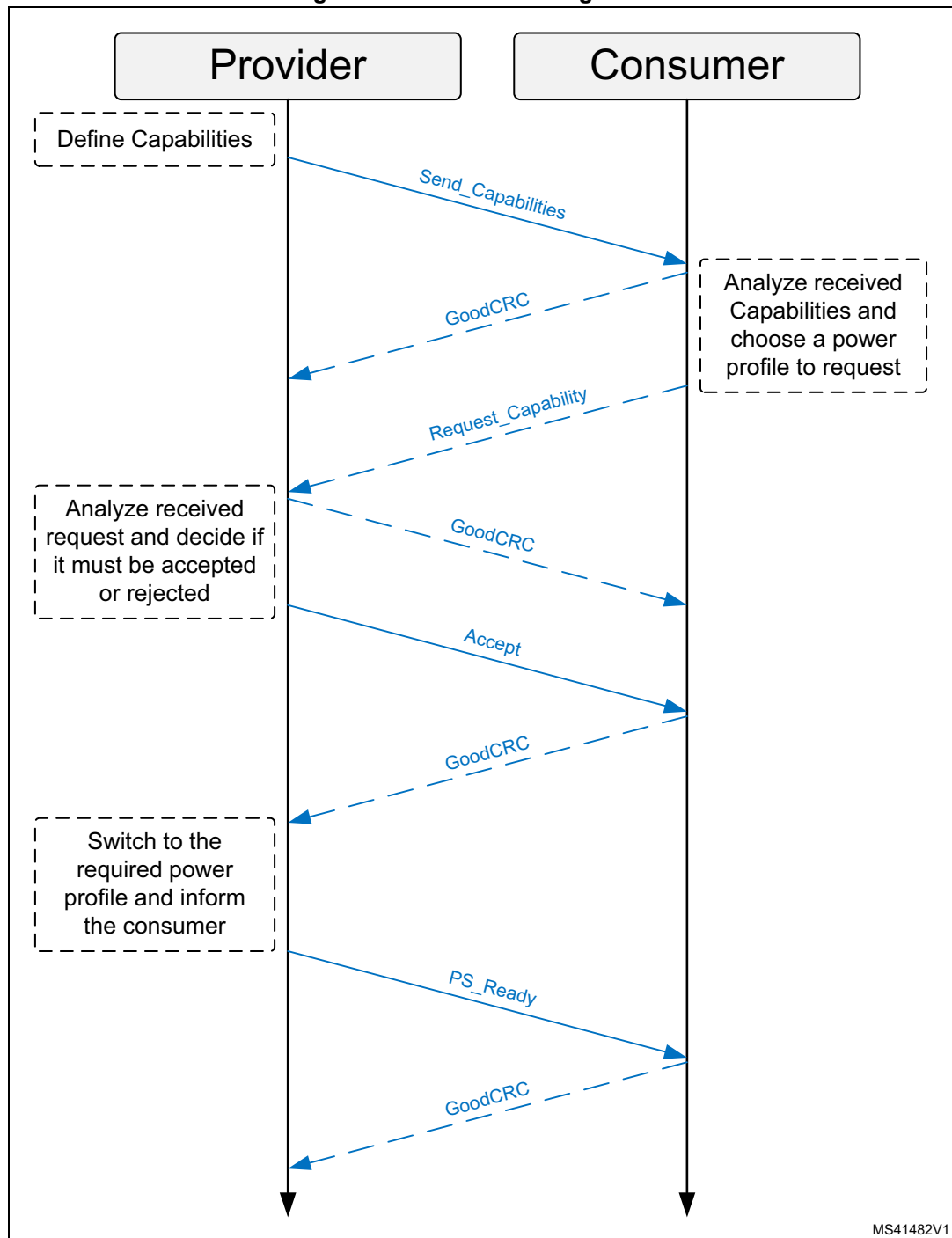
2.4 Message flow

Each message sent from a device is replied by a GoodCRC message. This allows the receiver to acknowledge the sender that its message is correctly received, and treated by upper layers.

The wrong messages received should be ignored. In case of a persistent communication error, a soft reset permits to reset protocol parameters to re-establish the communication. If the error persists, a hard reset of the system is performed.

In normal conditions, negotiating a power contract goes on according to the sequence illustrated in [Figure 4](#).

Figure 4. USB-PD - Messages flow

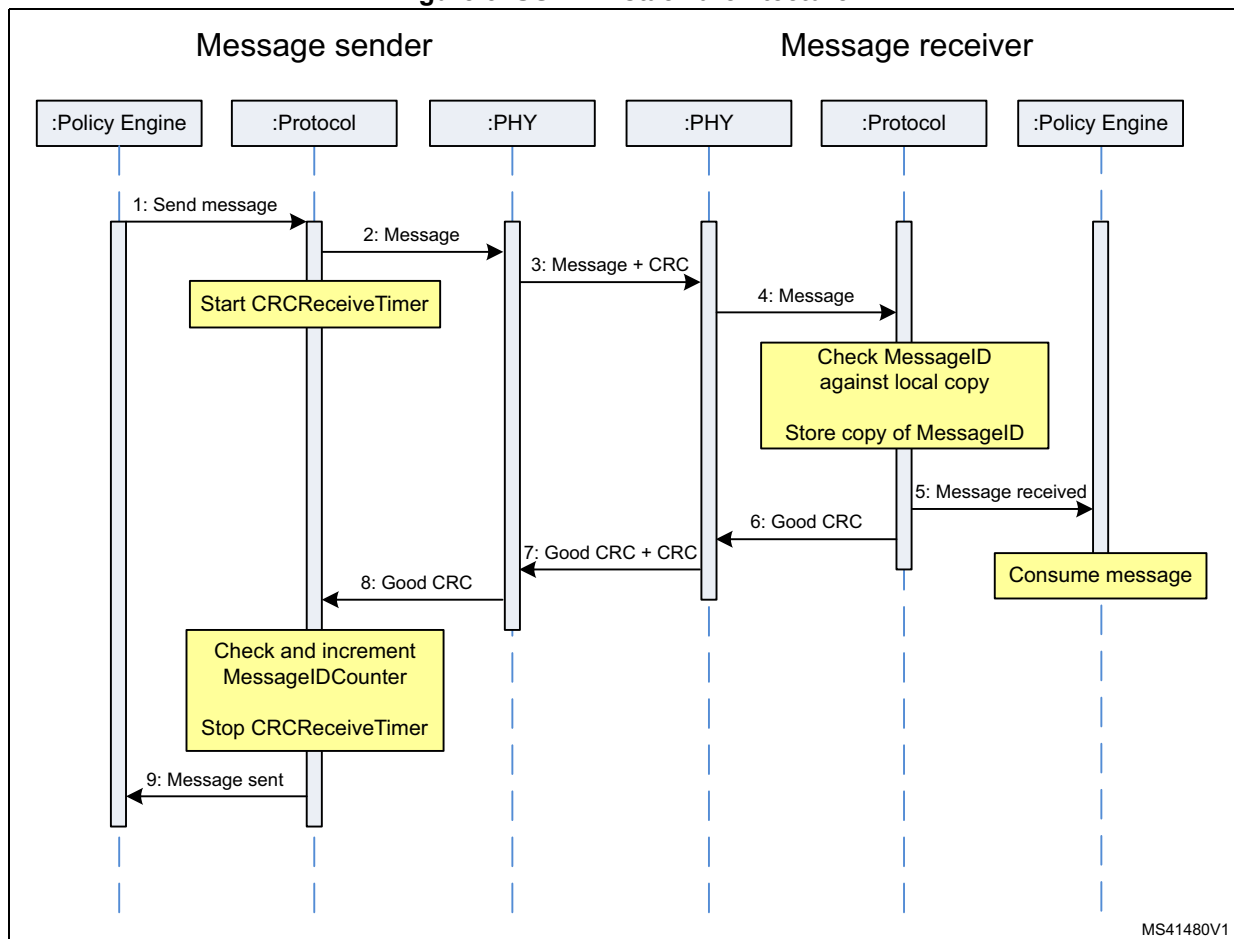


2.5 Data flow

In the normal case, every USB-PD basic message passes from the PE layer down to the PHY layer.

Each received message is replied to with a GoodCRC message to inform the sender of the correct reception (see [Figure 5](#)). BIST messages are an exception, as they must not be passed to the PE layer once received, and are only replied to with a GoodCRC.

Figure 5. USB-PD stack architecture



3 USB-PD library description

3.1 Overview

STMicroelectronics offers an USB-C Power Delivery Library supporting STM32F0xRB microcontrollers, based on an Arm^{®(a)} core.

The library is provided in binary format, comes on top of the STM32Cube HAL driver and offers all the APIs required to develop an USB PD application.

This section describes the USB-PD library middleware module and illustrates how users can develop their own power delivery application using this library.

The USB-PD library is developed following the Universal Serial Bus Power Delivery Specification Revision 2.0, V1.3 (January 12, 2017) and Universal Serial Bus type-C Cable and Connector Specification, Revision 1.3 (July 14, 2017). It has passed successfully the official certification.

The STM32 USB-PD package contains:

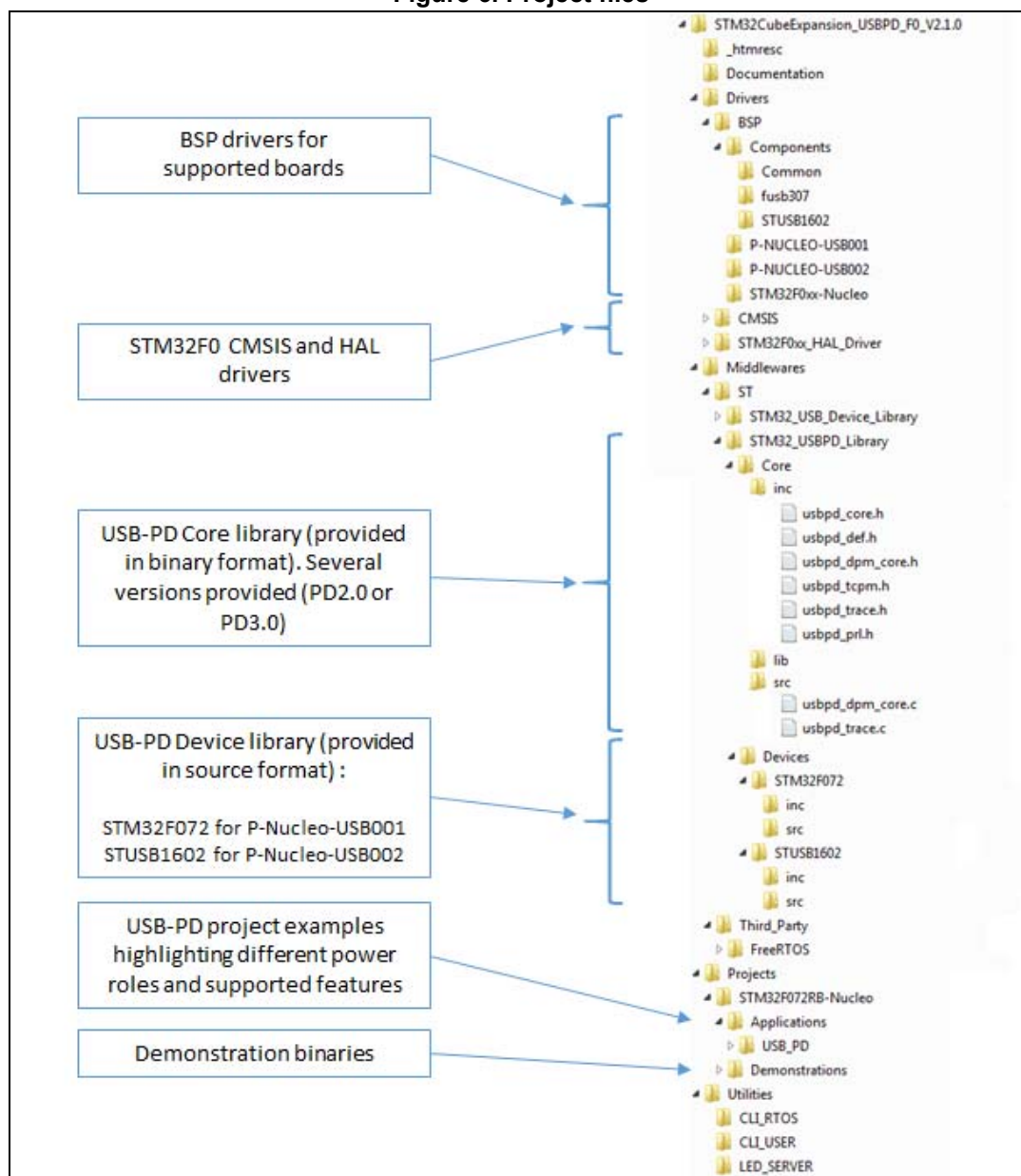
- the USB-PD core stack and the device drivers
- a set of examples that illustrate how to use USB-PD drivers to develop a power delivery application. This set of examples includes some STM32 Power Delivery provider, consumer, Dual Role Power and Dual Port examples.

The use of the VDM (Vendor Defined Messages) feature is also illustrated in both consumer and provider cases.

arm

a. Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

Figure 6. Project files



3.2 Features

The main characteristics of the USB-PD software Expansion Package are listed below:

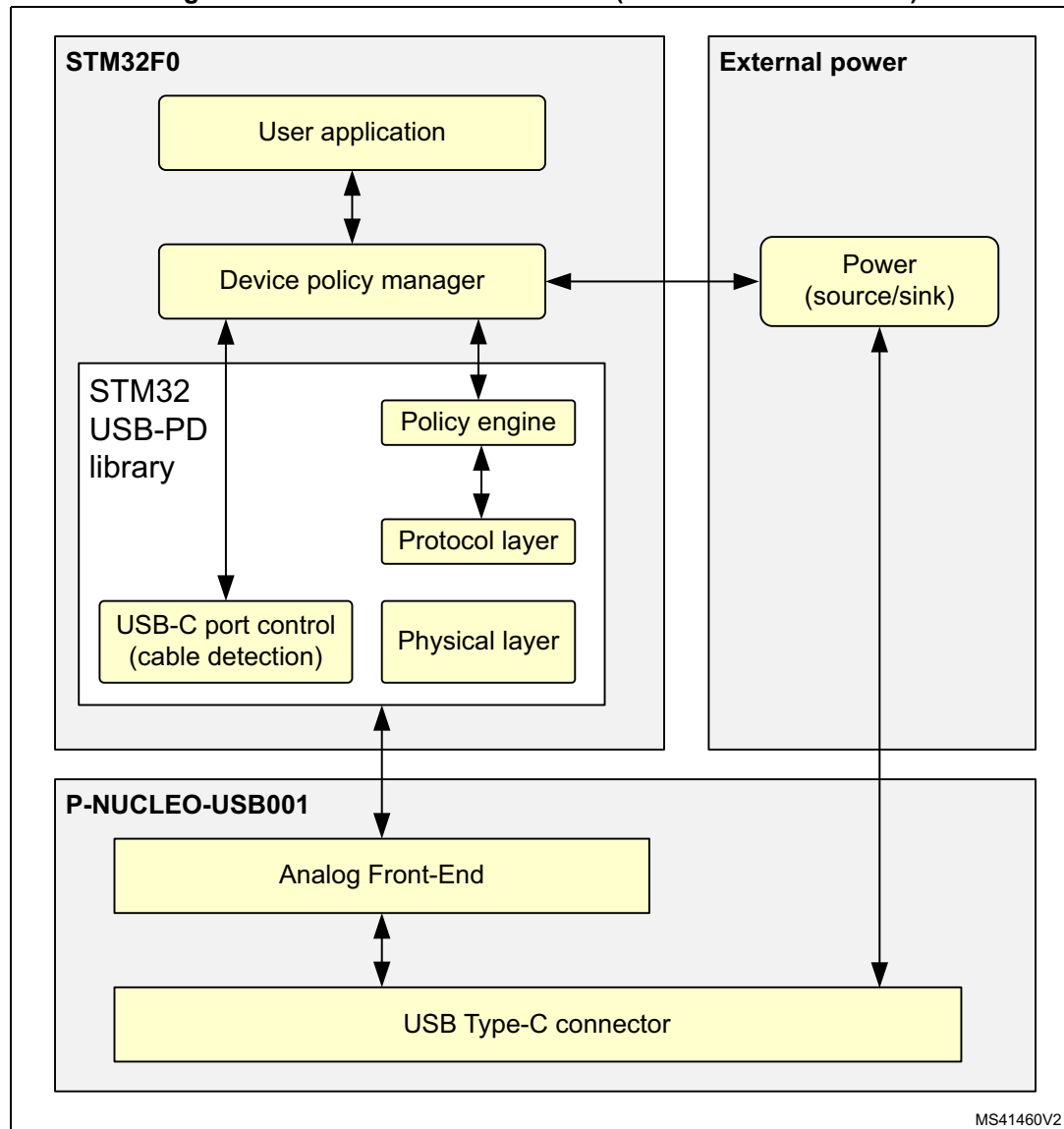
- compliant with USB Type-C 1.3 and USB Power Delivery 3.0 Standards
- Provider, Consumer and Dual Role mode (DRP) support
- Dual Port support
- PD communication supported for the two sides of the cable
- cable detection (of an USB-C cable plug/unplug operation)
- cable orientation (detection of orientation, to allow the user to choose one communication line CC1/CC2)
- USB-PD messages transmission/reception: Communication through CC line and message exchanging, coding/decoding using BMC and 5b4b coding
- drive VCONN switch
- BIST test mode support: BIST test mode to enable testing the platform at runtime
- structured VDM support allowing Alternate modes and extensions implementation
- Support of TCCP/TCCPC architectures

3.3 Library architecture

The USB-PD library uses the USB-C PD specification architecture as a reference for building the library.

A hardware expansion board (P-NUCLEO-USB001 or P-NUCLEO-USB002 shield), containing analog front ends and USB Type-C™ connectors, is used with the Nucleo board to ensure the communication.

Figure 7. USB-PD stack architecture (for P-NUCLEO-USB001)



The USB-PD libraries are provided for different configurations (PD2.0 or PD3.0) and various feature sets to fit different RAM/ROM constraints. USB-PD core stack libraries embed all generic components, independently from the HW:

- Protocol Layer
- Policy Engine Layer
- Cable Detection Module (CAD) part independent from HW.

Such Core library uses services provided by a device component (provided in source code), depending upon the used HW board. With the P-NUCLEO-USB001 or with the P-NUCLEO-USB002 boards the STM32F072 device component (specific) ensures:

- Physical Layer
- Cable Detection Module (CAD) part dependent from HW.

3.4 Hardware related components (for P-NUCLEO-USB001)

The Physical Layer and the Cable Detection are directly related to HW and need the use of a set of IPs (see [Table 2](#)).

Table 2. Use of different IPs

IP	Use
CRC	Good CRC calculation
ADC1	Detection of cable attach/detach
COMP1, COMP2	Data reception
DMA1 (Channels 2, 3, 4 and 5)	Buffering data
SPI1, SPI2	Data transmission
TIM1, TIM3, TIM14, TIM15, TIM16, TIM17	Power Delivery timers calculation Providing a time base for the communication process

When using those IPs, some GPIOs (see tables [3](#) and [4](#)) are reserved for the library and cannot be used for other purposes.

Table 3. GPIOs used by Port0

GPIO	Pin	IP	Use
GPIOA	0	COMP1-ADC1	CC1 data reception
	1	ADC1	CC reference voltage value
	3	ADC1	ADC channel pin for Port0
	5	COMP1-ADC1	CC2 data reception
GPIOB	1	TIM14	Data transmission
	8	-	P-NUCLEO-USB001 shield control
	9	-	
	12	-	
	13	SPI2	Data transmission
	14	SPI2	CC1 data transmission pin

Table 3. GPIOs used by Port0 (continued)

GPIO	Pin	IP	Use
GPIOC	0	ADC1	ADC channel pin for Port0
	2	SPI2	CC2 data transmission pin
	3	-	P-NUCLEO-USB001 shield control
	4	ADC1	ADC channel pin for Port0
	8	-	P-NUCLEO-USB001 shield control
GPIOD	2	-	P-NUCLEO-USB001 shield control

Table 4. GPIOs used by Port1

GPIO	Pin	IP	Use
GPIOA	1	ADC1	CC reference voltage value
	2	COMP2-ADC1	CC1 data reception
	4	COMP2-ADC1	CC2 data reception
	6	SPI1	CC2 data transmission pin
	7	ADC1	ADC channel pin for Port1
	15	-	P-NUCLEO-USB001 shield control
GPIOB	2	-	P-NUCLEO-USB001 shield control
	3	SPI1	Data transmission
	4	SPI1	CC1 data transmission pin
	5	-	P-NUCLEO-USB001 shield control
	15	TIM15	Data transmission
GPIOC	0	ADC1	ADC channel pin for Port1
	5	ADC1	ADC channel pin for Port1
	6	-	P-NUCLEO-USB001 shield control
	7	-	
	14	-	

Note: For more information on P-NUCLEO-USB001 shield and the needed connections with the STM32, refer to user manual UM2050, available on www.st.com.

The USB-C PD library defines and uses the three groups of interrupt priorities listed in [Table 5](#).

Table 5. Interrupt priorities

Priority level	Process	Priority value
Critical	Transmission	0
High	Reception	2
Low/medium	Others	3

Critical and high priority interrupts are used for handling USB PD transmission and reception processes. Consequently, the user application must not define an interrupt with a priority equal or lower than two, to avoid interference with PD/VDM communication. Interrupts with priority values higher than two can be used at any time.

4 USB-PD library programming guidelines

The X-CUBE-USB-PD package includes libraries (USB-PD Core stack provided in binary format and USB-PD Device component provided in source code), with corresponding headers files “.h” describing library APIs.

4.1 Description of available configurations

USB-PD Core stack is delivered as a library. Several configurations are supported in order to address all user use cases. All examples projects provided in the X-CUBE-USB-PD Software package, include use of a specific library, in line with project configuration and capabilities.

Naming convention used for available libraries is described below. It should help user to select the appropriate library suitable for his project. Library name is built as

LibraryType_PDVersion_Capabilities_CMx_IDE.(a/lib), where:

- ***LibraryType*** describes the type of architecture:
 USBPDCORE: Standard USBPD architecture (complete framework) as used on P-Nucleo-USB001 and P-Nucleo-USB002 kits
 USBPDTCPM: TCPM/TCPC architecture (library embeds the TCPM task, in charge of driving TCPC controller)
- ***PDVersion*** describes the PD Specification Revision:
 PD2: Revision 2.0
 PD3: Revision 3.0
- ***Capabilities*** describes the functional capabilities supported by the library (goal of offering some libraries with less capabilities is to allow integration in smaller memory size targets):
 CONFIG_MINSNK: Sink only role supported, no optional feature
 CONFIG_MINSRC: Source only role supported, no optional feature
 CONFIG_1: Sink, Source or DRP role with support of optional features. List of optional features supported for each configuration could be found in .\Middlewares\ST\STM32_USBPD_Library\Core\inc\usbpd_def.h file.
 FULL: Same as CONFIG_1 with addition of VDM support.
- ***CMx*** describes the CMx core class (CM0, CM3, CM4 or CM7)
- ***IDE*** defines the binary convention, and determines target compiler to be used
 wc16: Wchar_t 16 bits (suitable for EWARM v7)
 wc32: Wchar_t 32 bits (suitable for EWARM v8, MDK-ARM v6, SW4STM32)
 Keil: Wchar_t 32 bits (suitable for MDK-ARM v5)

All libraries are generated with <high size> optimization.

When building project, user should also ensure that compilation option flag set in project options, is in line with selected library. For example, when user project is linked with USBPDCORE_PD2_FULL_CM0_wc32.a, user should ensure that USBPDCORE_LIB_PD2_FULL compilation option is properly set in project options. All compilation options to be used in project options could be found in .\Middlewares\ST\STM32_USBPD_Library\Core\inc\usbpd_def.h file.

4.2 Library initialization

To use and initialize the library, follow the steps listed below.

1. Include the STM32 USB-C PD Core Stack library in your project. Select library type according to your tool chain, and according to features required to be supported (PD2.0 or PD3.0, VDM supported or not). Available libraries for the USB-PD Core stack component, are available in the directory `\Middlewares\ST\STM32_USBPD_Library\Core\lib\`. A second library has to be selected for the USB-PD component depending upon the used HW kit (P-Nucleo-USB001 or P-Nucleo-USB002), in the directory `\Middlewares\ST\STM32_USBPD_Library\Devices\`. For P-Nucleo-USB001 specific case, select device source code from `\Middlewares\ST\STM32_USBPD_Library\Devices\STM32F072\` directory.
2. Include the User application files in the application. Some examples of files are available in the provided project examples in Firmware package. In provided examples, User application files are:
 - *main.c*: Program Main file, includes main and System Clock configuration functions. Program main function configures the MCU clock (template implemented in the *systemClock_Config()* function in the *main.c* file of each project example). Then, main calls the *USBPD_HW_IF_GlobalHwInit()* function to initialize the Hardware Interface layer and calls the *USBPD_DPM_Init()* function to initialize the Device Policy Manager layer and all the communication layers (PE, VDM, PRL, PHY).
 - *stm32f0xx_hal_conf.h*: contains STM32F0xx HAL configuration file (such as enabled IPs)
 - *stm32f0xx_hal_msp.c*: contains HW interface MSP functions (referred to by HAL IP Initialization functions)
 - *stm32f0xx_it.c/stm32f0xx_it.h*: provides definitions for all exceptions handlers and peripherals interrupt service routines used in application.
 - *usbpd_dpm_user.c/usbpd_dpm_user.h*: contains USBPD DPM user code, including implementation of functions and callbacks required to fulfill USB Core Stack API.
 - *usbpd_pwr_if.c/usbpd_pwr_if.h*: contains definition of power interface control functions
 - *FreeRTOSConfig.h*: contains Application specific definitions related to FreeRTOS use.
3. In a FreeRTOS configuration, call to *USBPD_DPM_Init()* performs threads, message queues creation and launches the RTOS Kernel (task scheduler).

Note: The USB-C PD library uses FreeRTOS to manage high level layers.

4.3 USB-PD core stack library functions

The DPM layer calls functions from the PE and the VDM layers to trigger PD contract establishment, and to enable the use of Vendor Defined Messages.

For complete description of USBPD core stack API and interface refer to the CHM file provided in the */Documentation* directory of the X-CUBE-USBPD package.

4.4 USB-PD core stack

Policy Engine or VDM layers need to inform the DPM of a notification or of a request. This is enabled by using structures of callbacks.

For a complete description of the USBPD core stack callbacks to be implemented on DPM user side refer to CHM file provided in the */Documentation* directory of the X-CUBE-USBPD package (*STM32F072xB_USBPD_CORE_RELEASE_User_Manual.chm*), in particular to *USBPD_PE_Callbacks Struct Reference* description. For a complete description of the USBPD core TCPM stack callbacks to be implemented on TCPC component side, refer to CHM file provided in the */Documentation* directory of the X-CUBE-USBPD package (*STM32F072xB_USBPD_CORE_TCPM_RELEASE user manual*), in particular to *TCPC_DrvTypeDef Reference* description.

4.5 USB-PD device components

The description of USBPD device components, as provided in X-CUBE-USBPD package, are available in CHM files in the */Documentation* directory of the package *STM32F072xB_USBPD_DEVICES_xxx_User_Manual.chm*.

5 Atomic message sequencing

5.1 Acronym

AMS: Atomic Message Sequence

DPM: Device Policy Manager

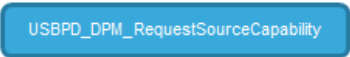


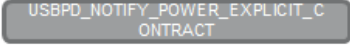
SRC: type C port working as current provider

SNK: type C port working as current consumer

5.2 This chapter describes different scenarios:

- Collision avoidance management (only valid for PD3 specification)
- AMS sequence managed by the ST stack
 - SRC AMS: Power negotiation
 - SNK AMS: Power negotiation
 - Generic AMS: SOFT Reset
- VCONN swap
- VDM sequences
- TCPM sequences

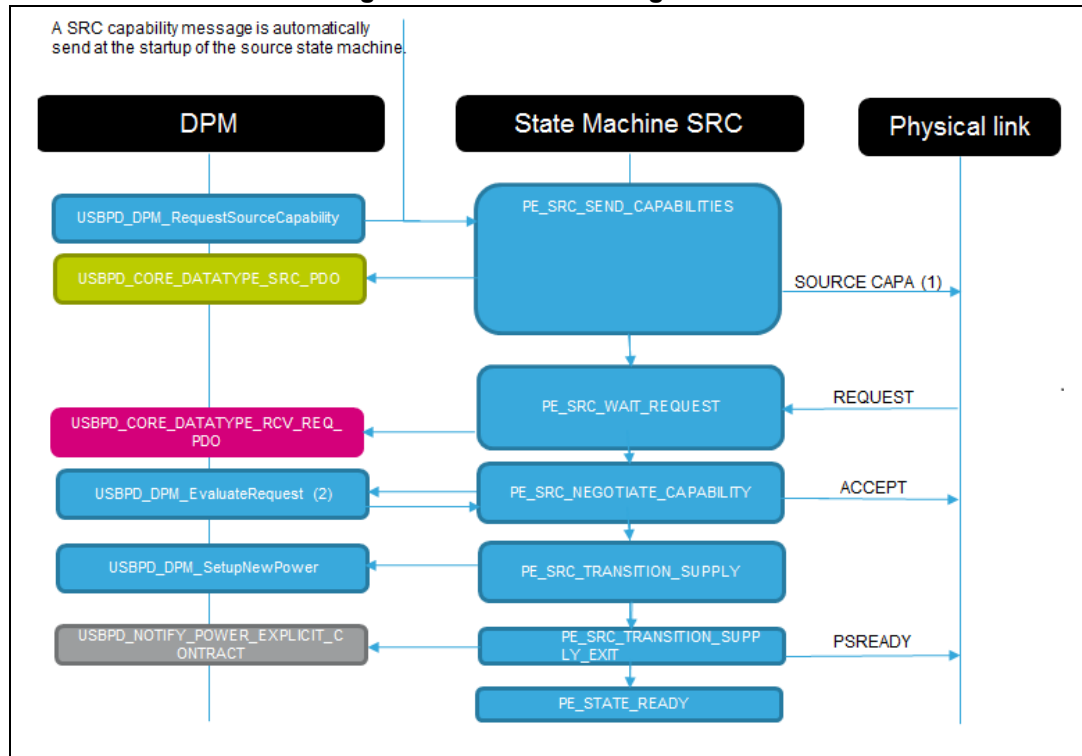
Figure 8. color conventions for following slides

	Func call or state machine state
	Request to get data from DPM (function:USBPD_DPM_GetDataInfo)
	Transfer data information to DPM (function:USBPD_DPM_SetDataInfo)
	Notification to DPM

5.3 SRC AMS

5.3.1 SRC: Power Negotiation

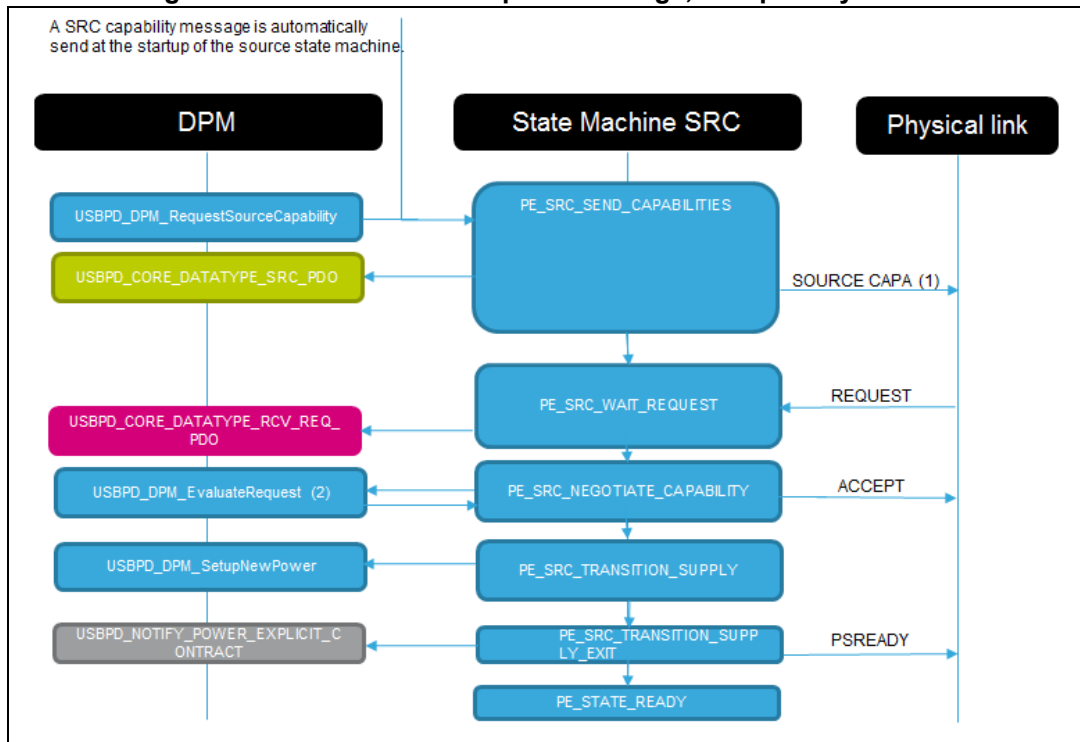
Figure 9. SRC: Power Negotiation



1. Only after having a first PD message acknowledged by a GOODCRC, the messages not acknowledged generate a SOFT RESET.
2. User has the choice to return ACCEPT, REJECT or WAIT. WAIT means SRC is not ready and ask SNK to do a request later.

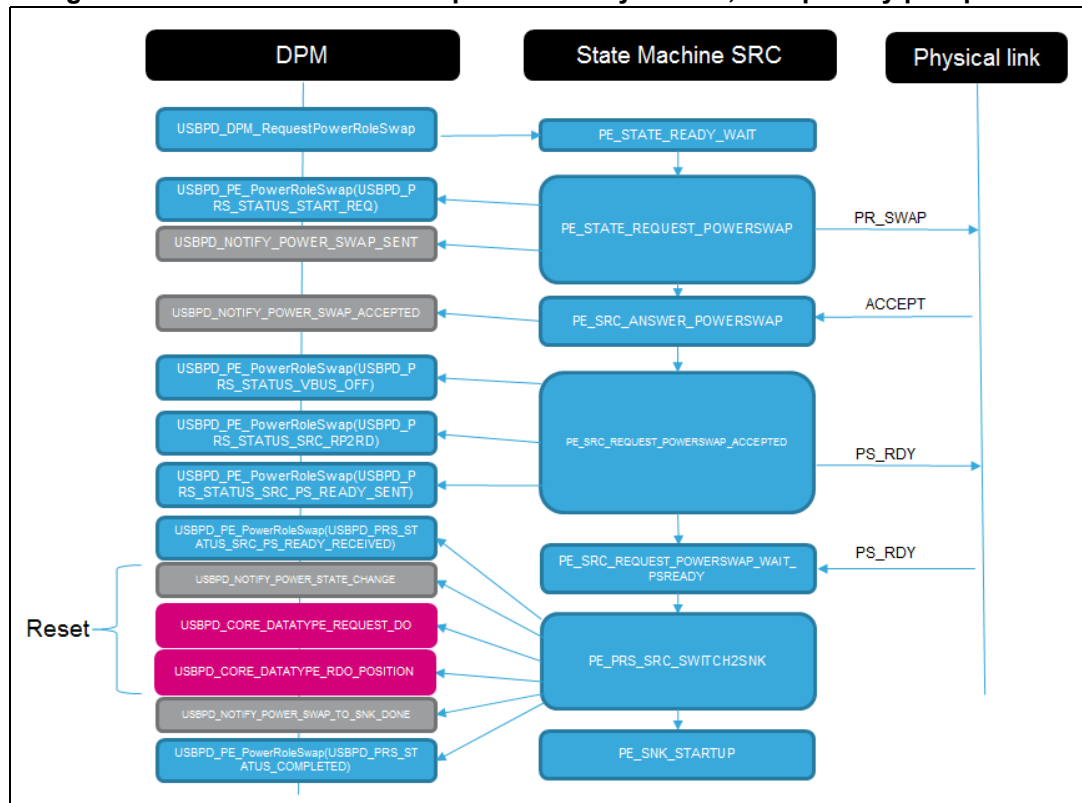
5.3.2 SRC: Receive a request message, accepted by Source

Figure 10. SRC: receive a request message, accepted by Source



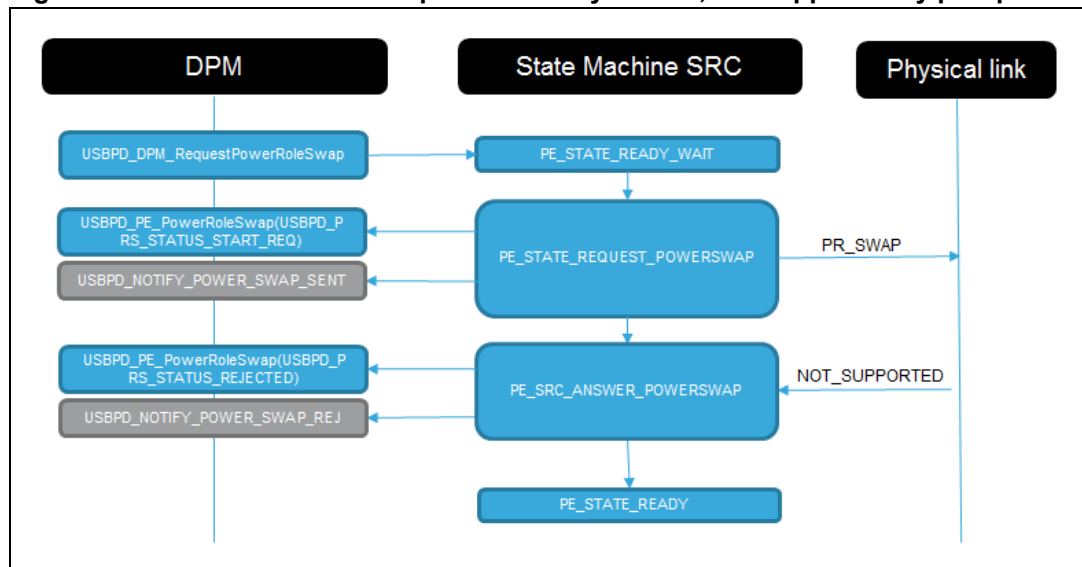
5.3.3 SRC: Power Role Swap - initiated by source, accepted by port partner

Figure 11. SRC: Power Role Swap - initiated by source, accepted by port partner



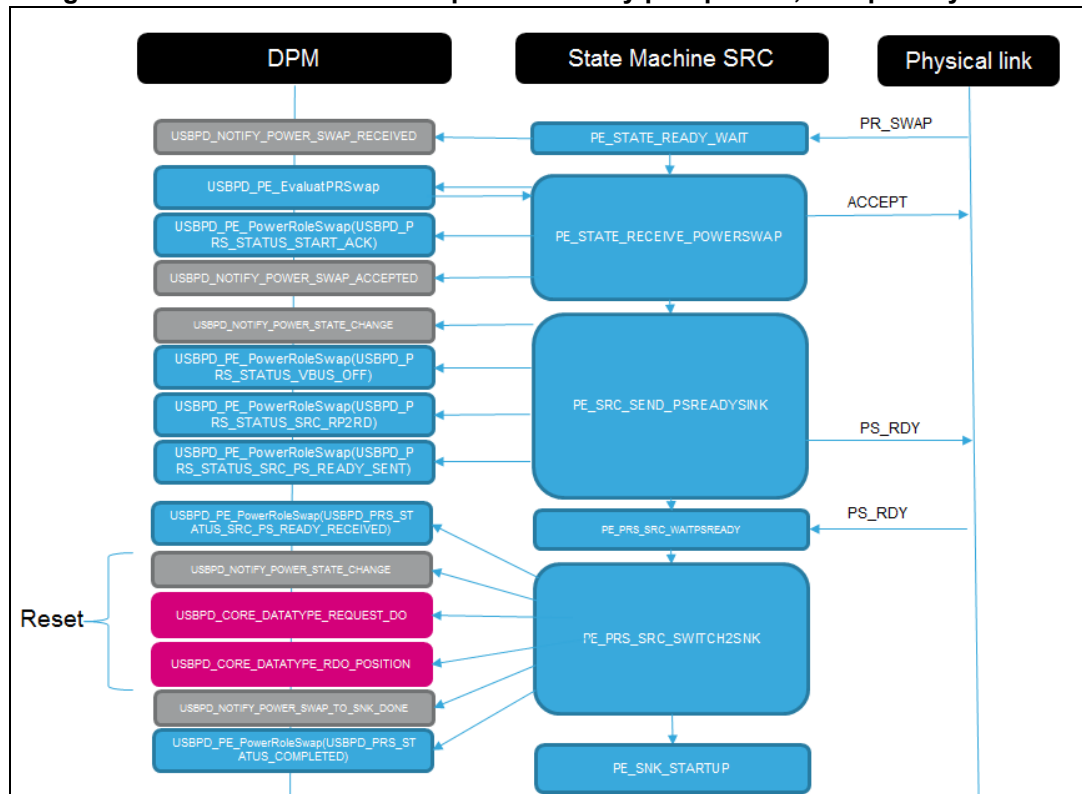
5.3.4 SRC: Power Role Swap – initiated by source, not supported by port partner

Figure 12. SRC: Power Role Swap – initiated by source, not supported by port partner



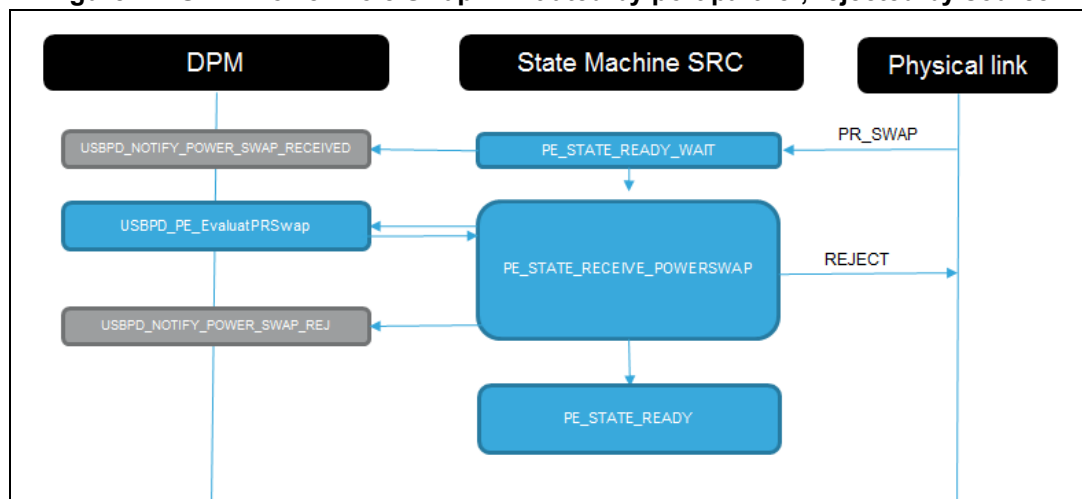
5.3.5 SRC: Power Role Swap – initiated by port partner, accepted by source

Figure 13. SRC: Power Role Swap – initiated by port partner, accepted by source



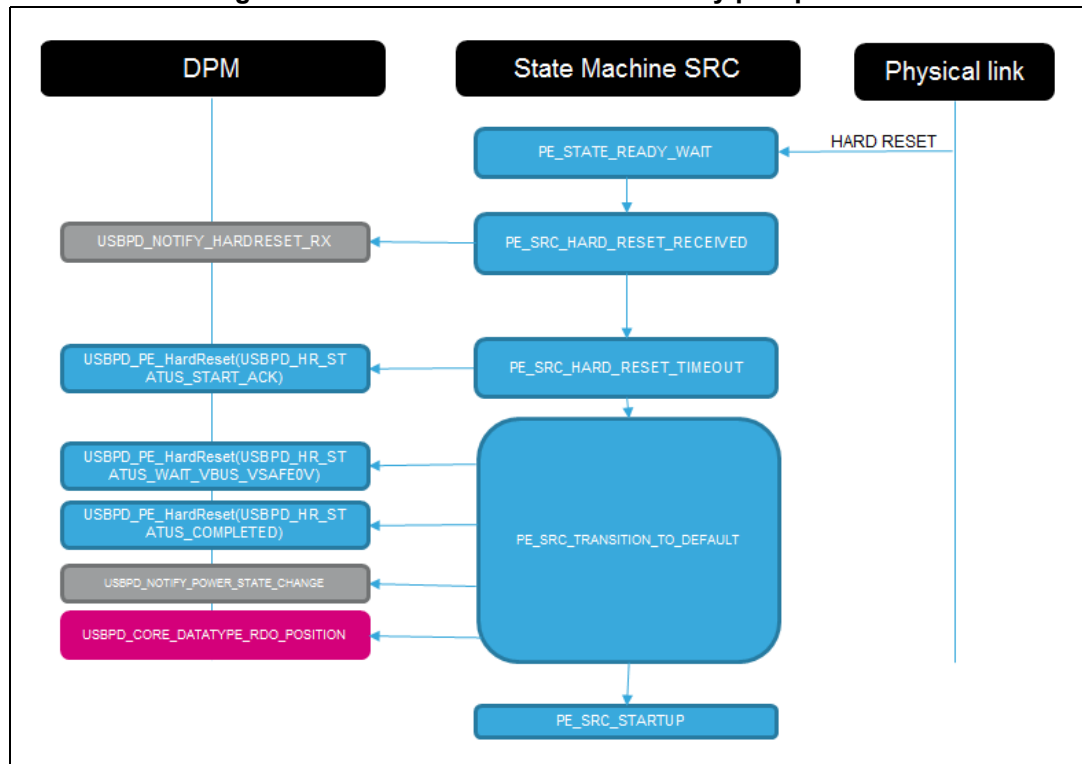
5.3.6 SRC: Power Role Swap – initiated by port partner, rejected by source

Figure 14. SRC: Power Role Swap – initiated by port partner, rejected by source



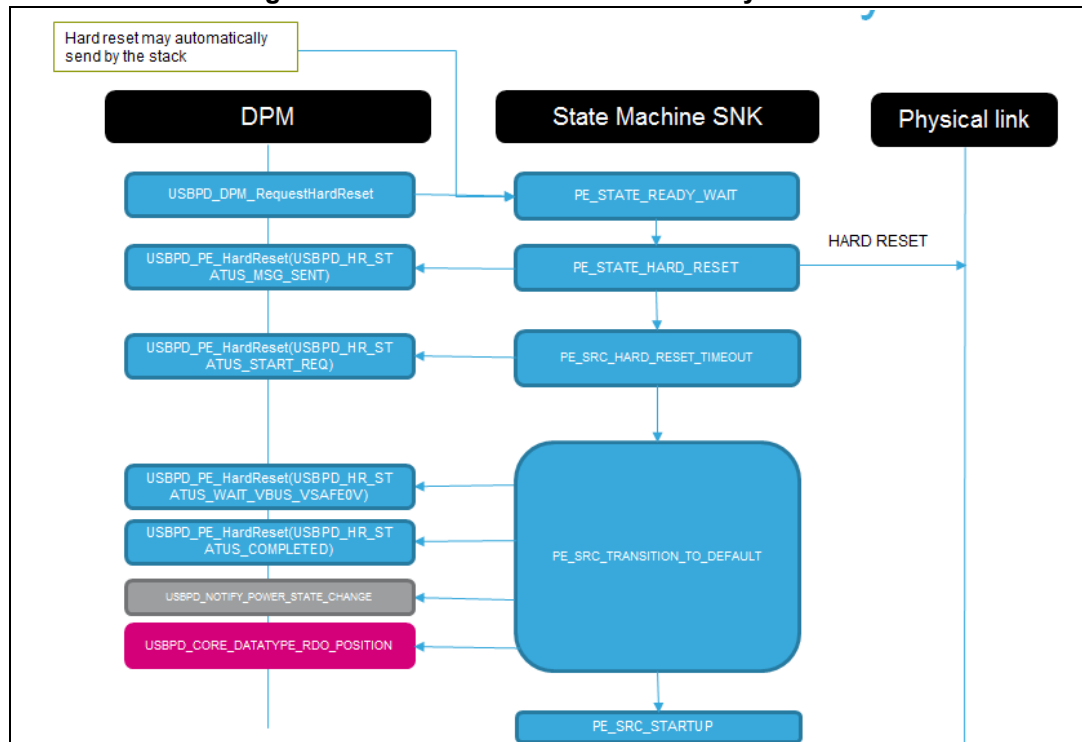
5.3.7 SRC: Hard Reset – initiated by port partner

Figure 15. SRC: Hard Reset – initiated by port partner



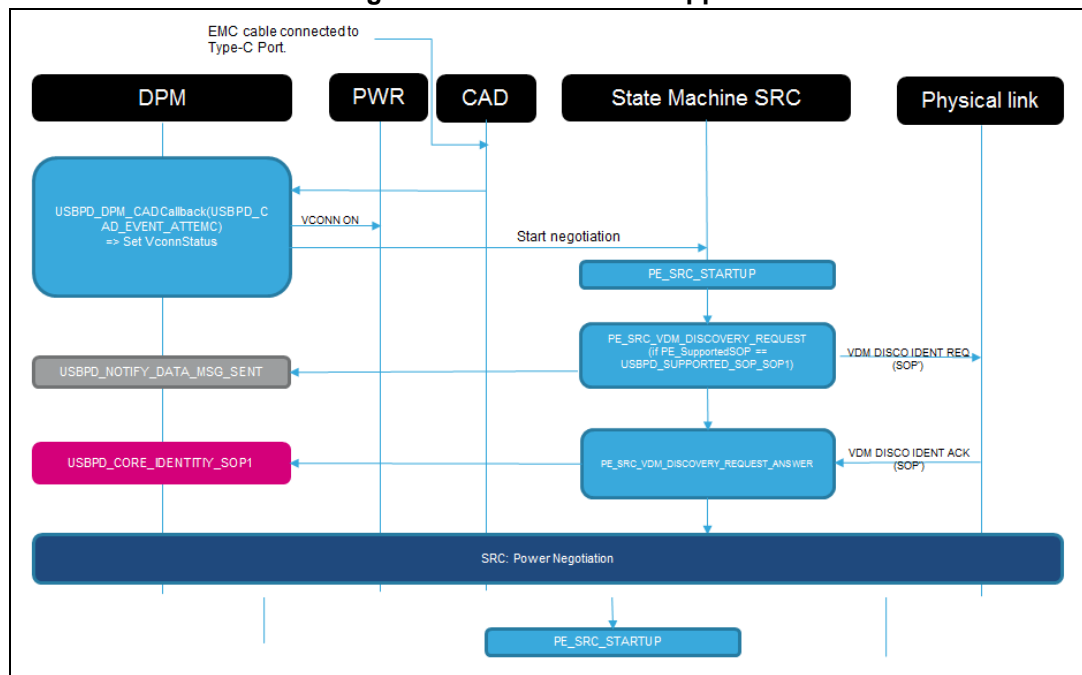
5.3.8 SRC: Hard Reset – initiated by Source

Figure 16. SRC: Hard Reset – initiated by Source



5.3.9 SRC: VCONN Support

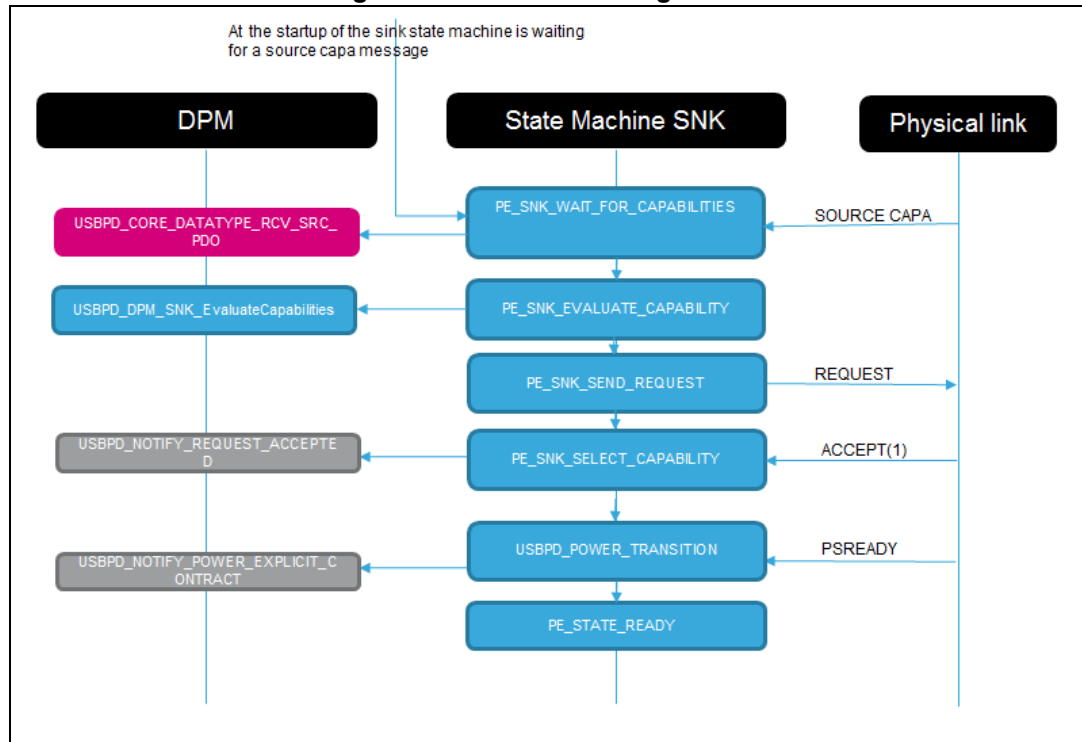
Figure 17. SRC: VCONN Support



5.4 SNK AMS

5.4.1 SNK: Power Negotiation

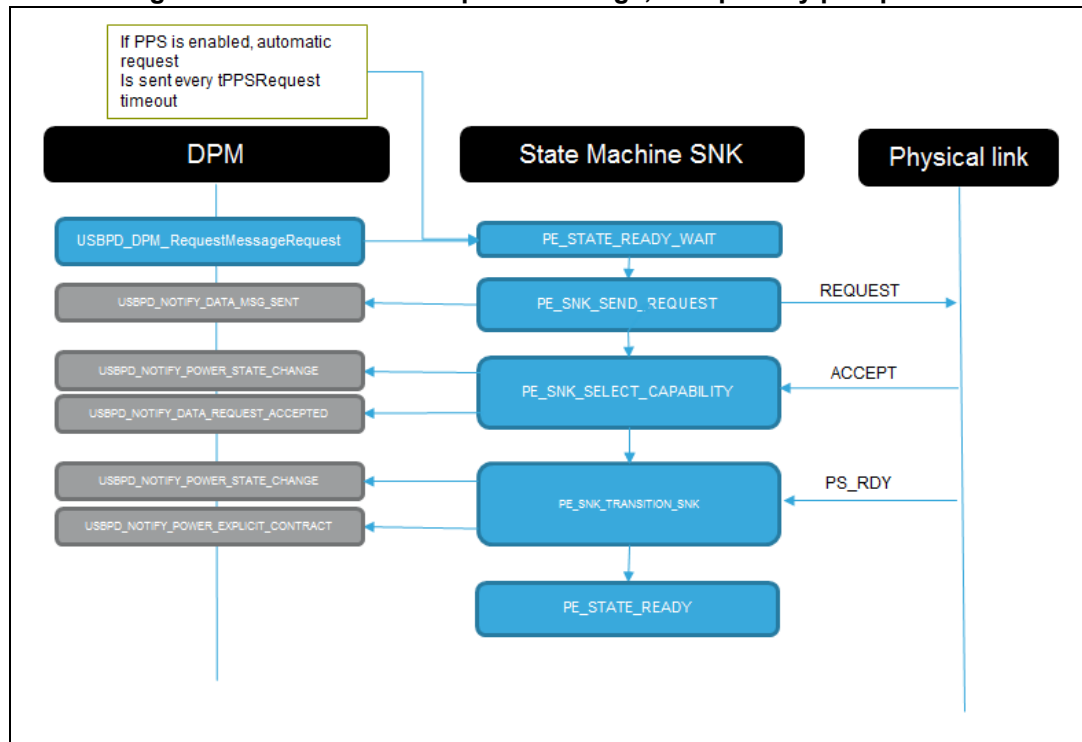
Figure 18. SNK: Power Negotiation



1. If SRC reply WAIT SNK must send a request later, if received REJECT the request has not been accepted by the SNK a new request can be done.

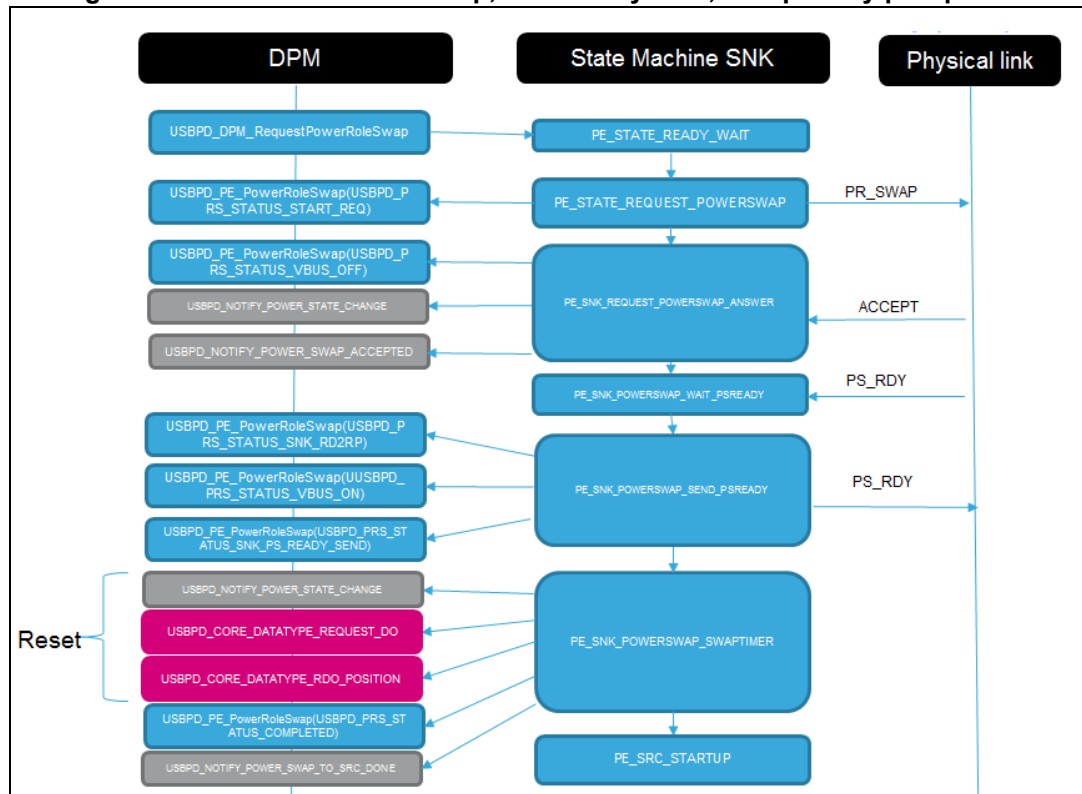
5.4.2 SNK: Send a request message, accepted by port partner

Figure 19. SNK: Send a request message, accepted by port partner



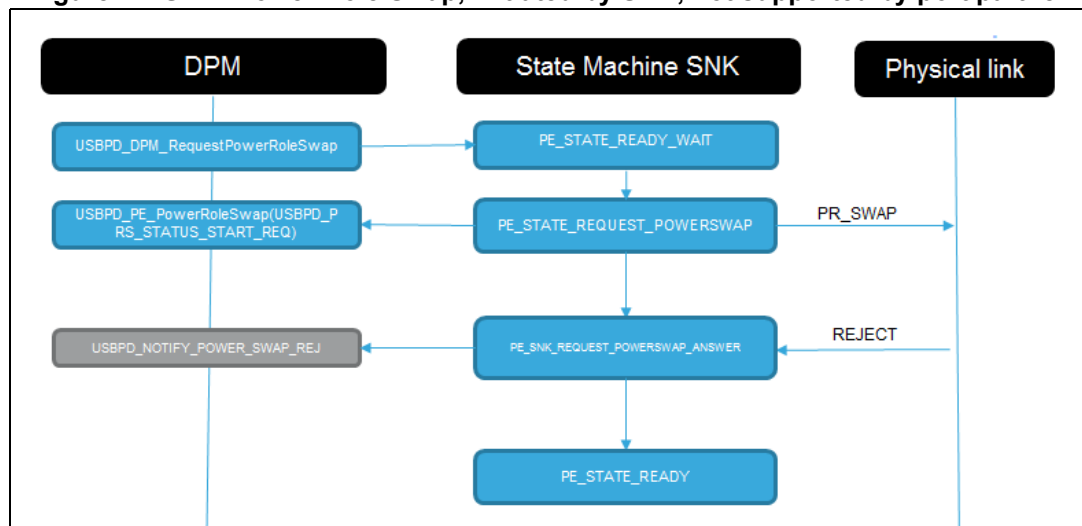
5.4.3 SNK: Power Role Swap, initiated by Sink, accepted by port partner

Figure 20. SNK: Power Role Swap, initiated by Sink, accepted by port partner



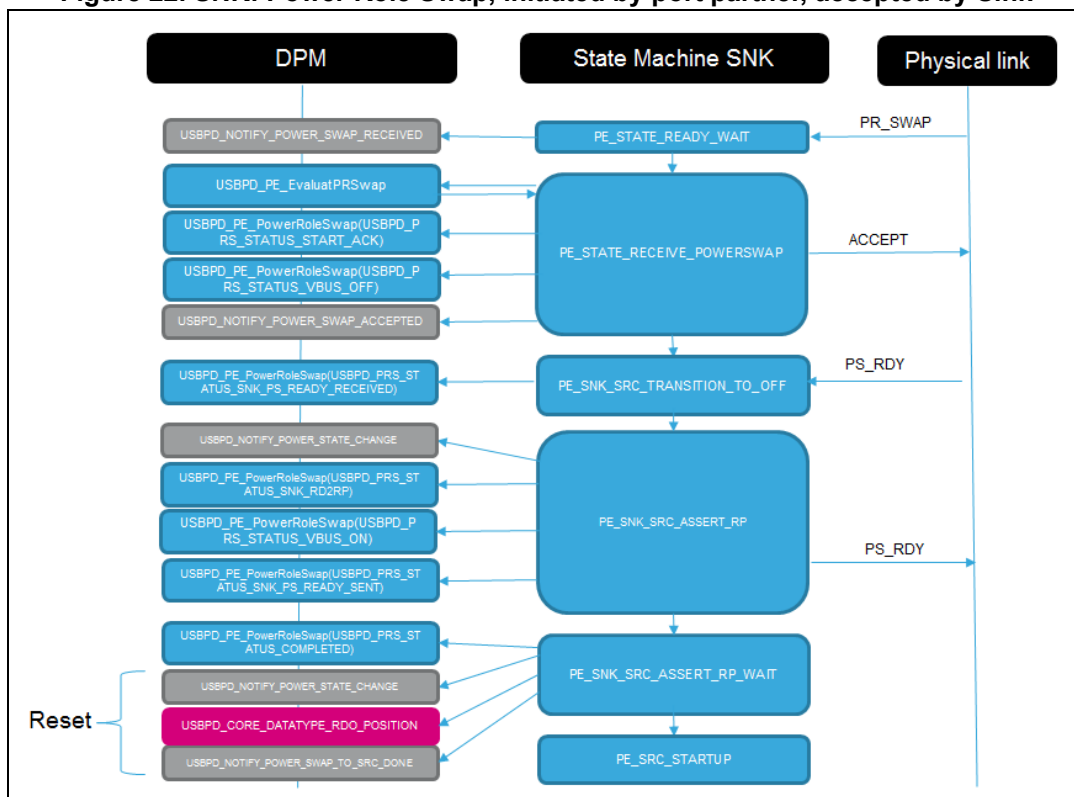
5.4.4 SNK: Power Role Swap, initiated by Sink, not supported by port partner

Figure 21. SNK: Power Role Swap, initiated by Sink, not supported by port partner



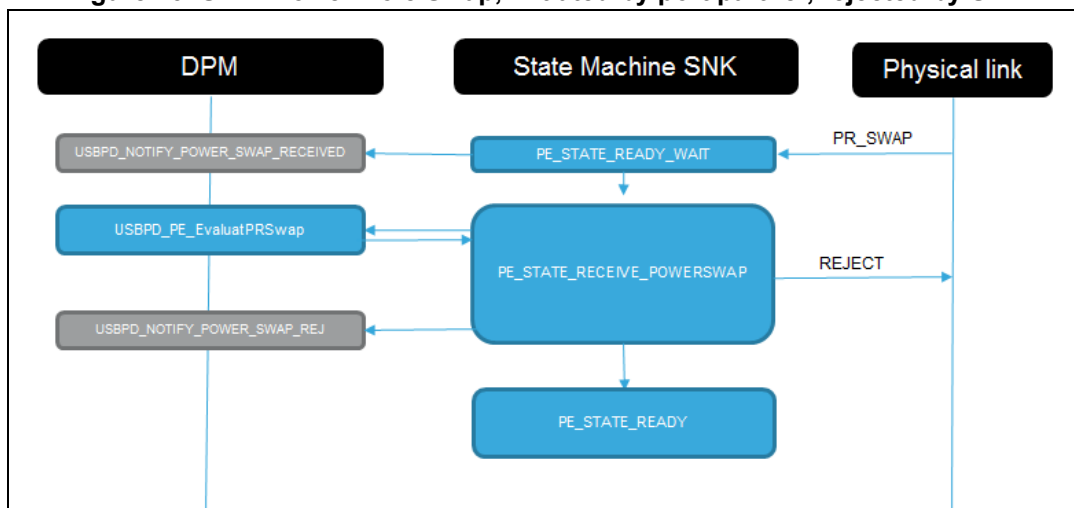
5.4.5 SNK: Power Role Swap, initiated by port partner, accepted by Sink

Figure 22. SNK: Power Role Swap, initiated by port partner, accepted by Sink



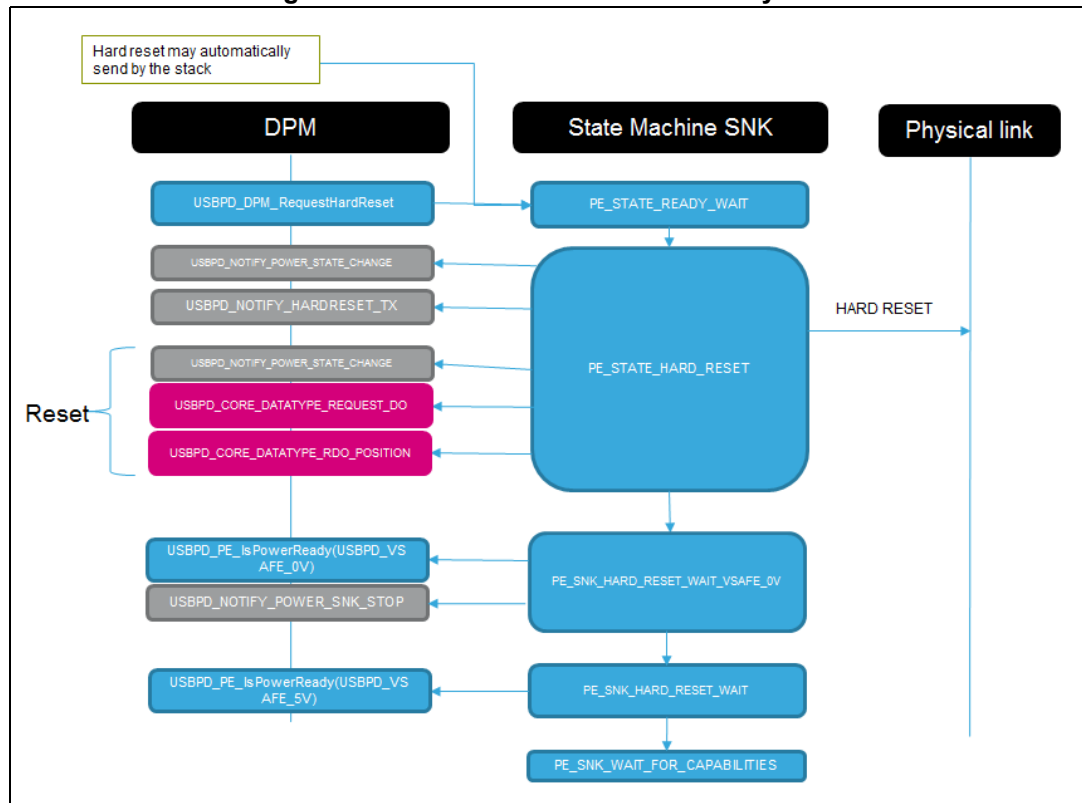
5.4.6 SNK: Power Role Swap – initiated by port partner, rejected by Sink

Figure 23. SNK: Power Role Swap, initiated by port partner, rejected by Sink



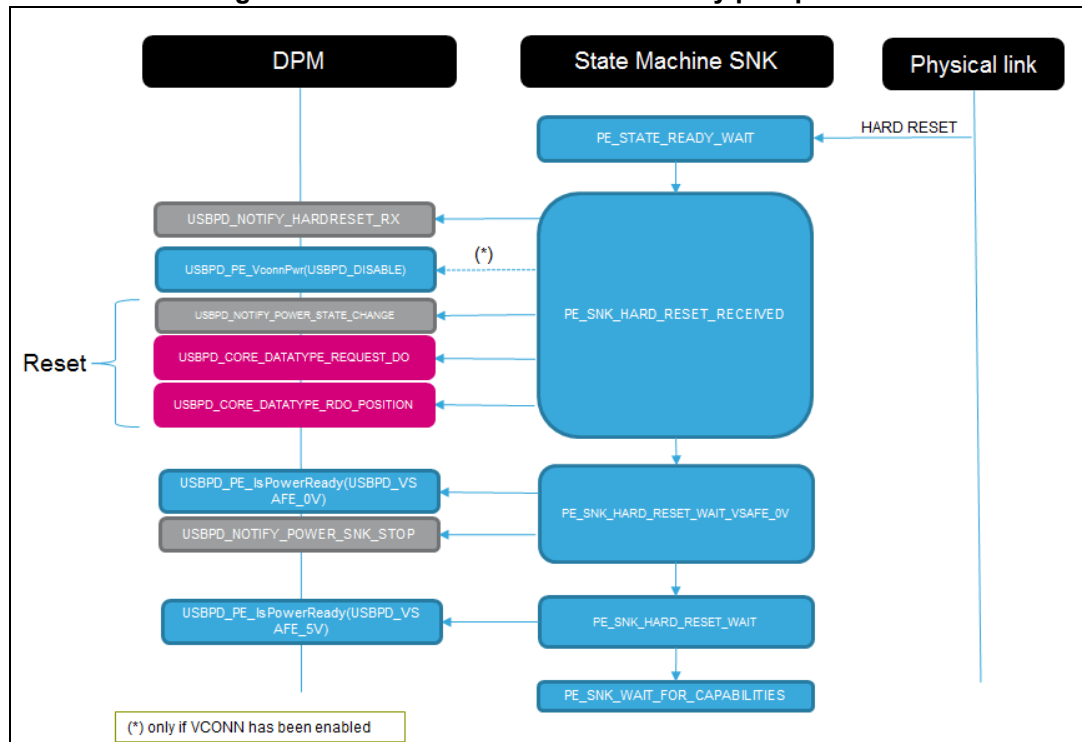
5.4.7 SRC: Hard Reset – initiated by sink

Figure 24. SRC: Hard Reset – initiated by sink



5.4.8 SRC: Hard Reset – initiated by port partner

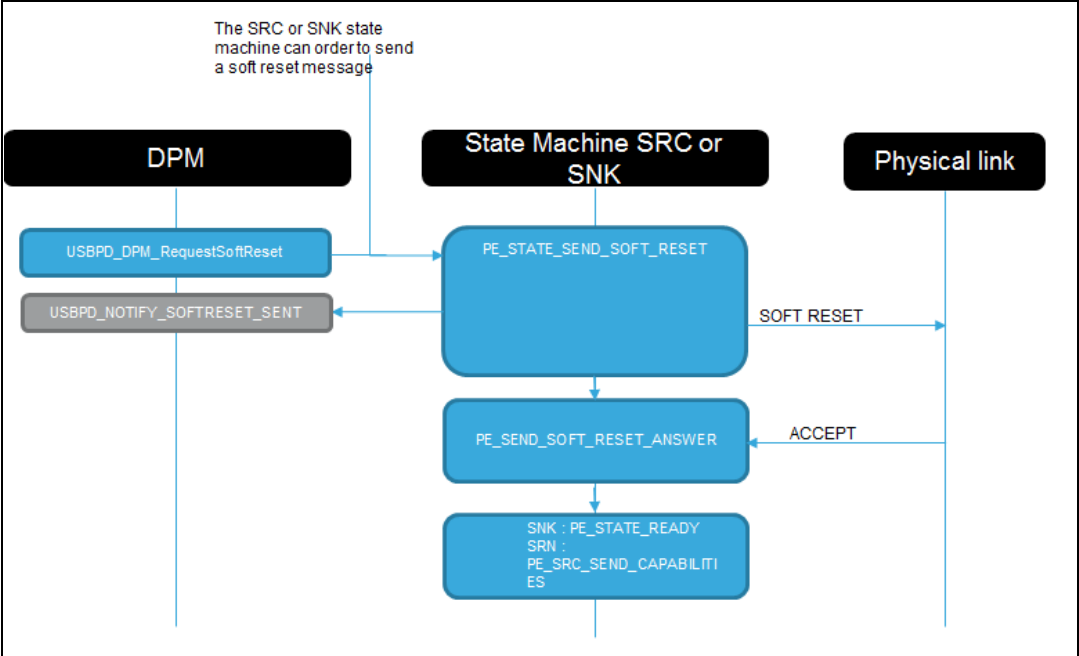
Figure 25. SRC: Hard Reset – initiated by port partner



5.5 Generic AMS

5.5.1 Soft Reset TX

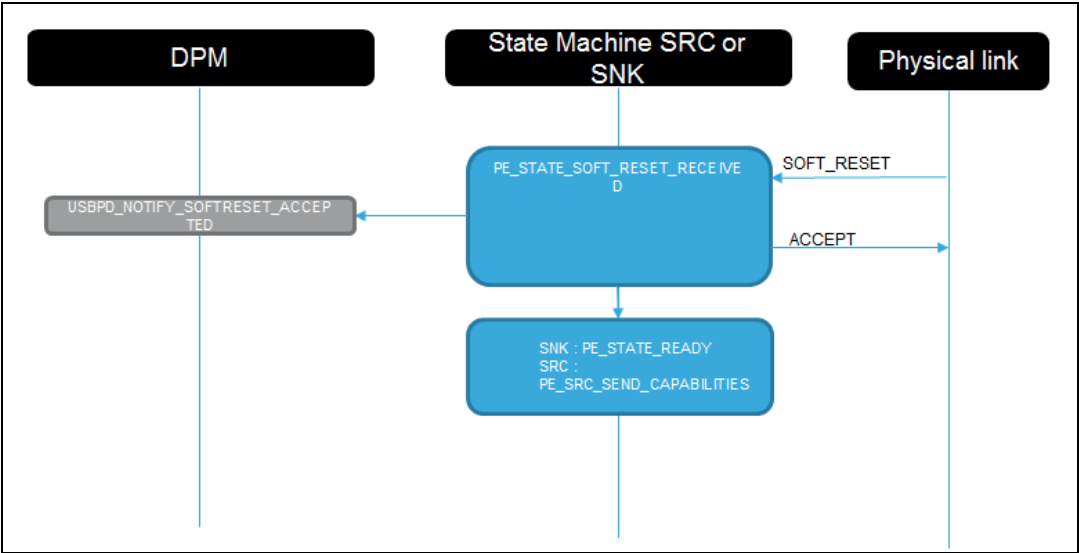
Figure 26. Soft Reset TX



1. In case a `SOFT_RESET` message not received an `ACCEPT` answer, a `HARD_RESET` message is sent.

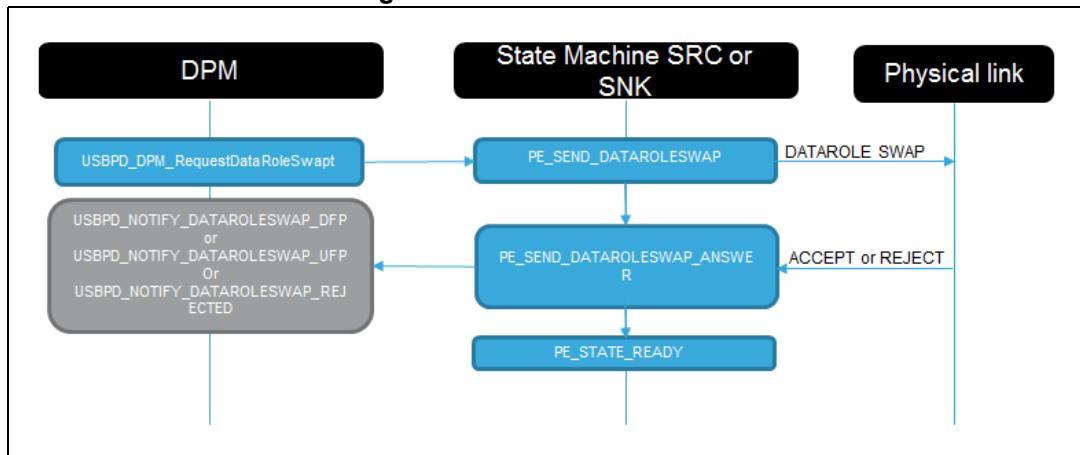
5.5.2 Soft Reset RX

Figure 27. Soft Reset RX



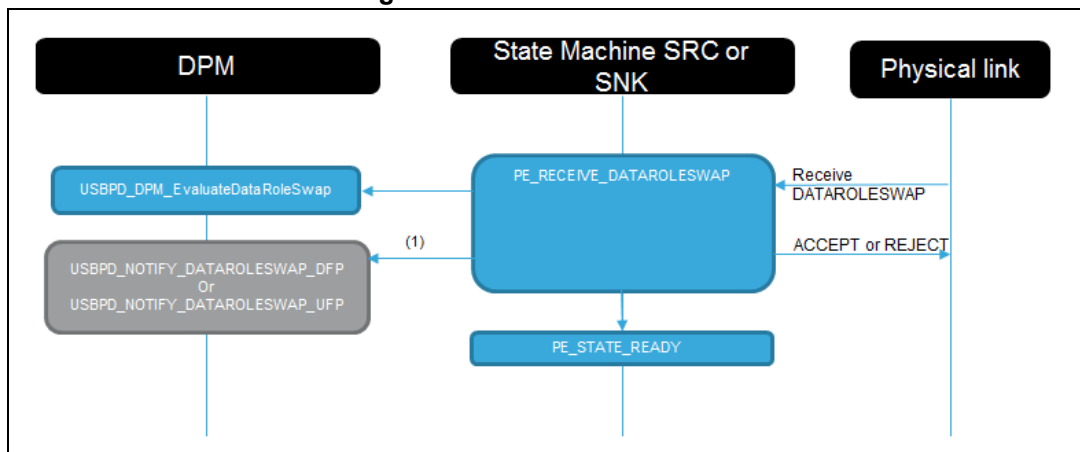
5.5.3 Data Role SWAP TX

Figure 28. Data Role SWAP TX



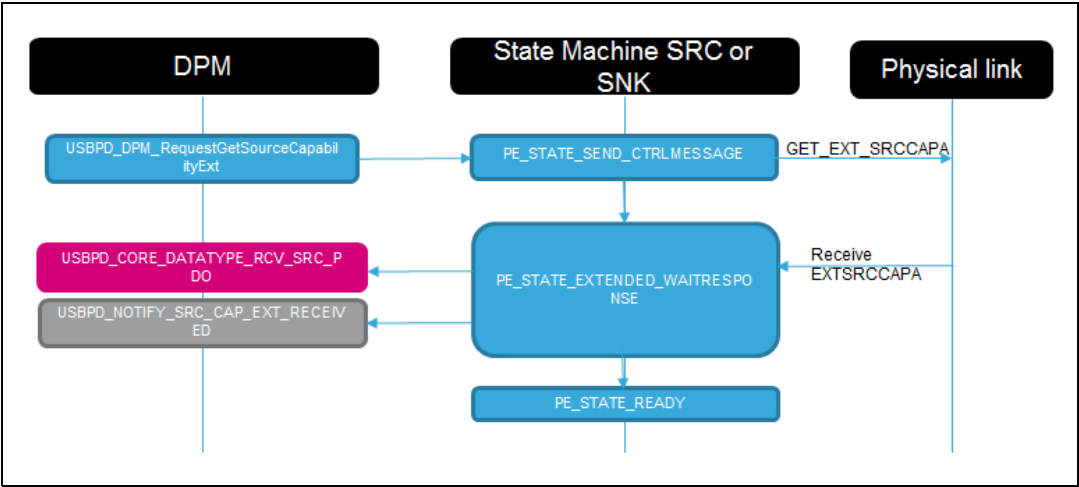
5.5.4 Data Role SWAP Rx

Figure 29. Data Role SWAP Rx



5.5.5 GET extended capa TX

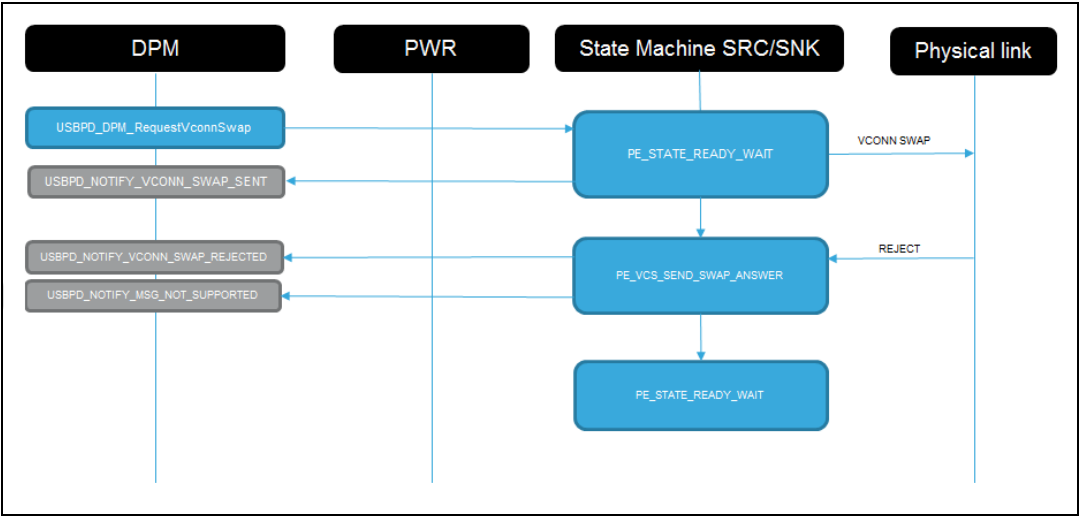
Figure 30. GET extended capa TX



5.6 VCONN Swap

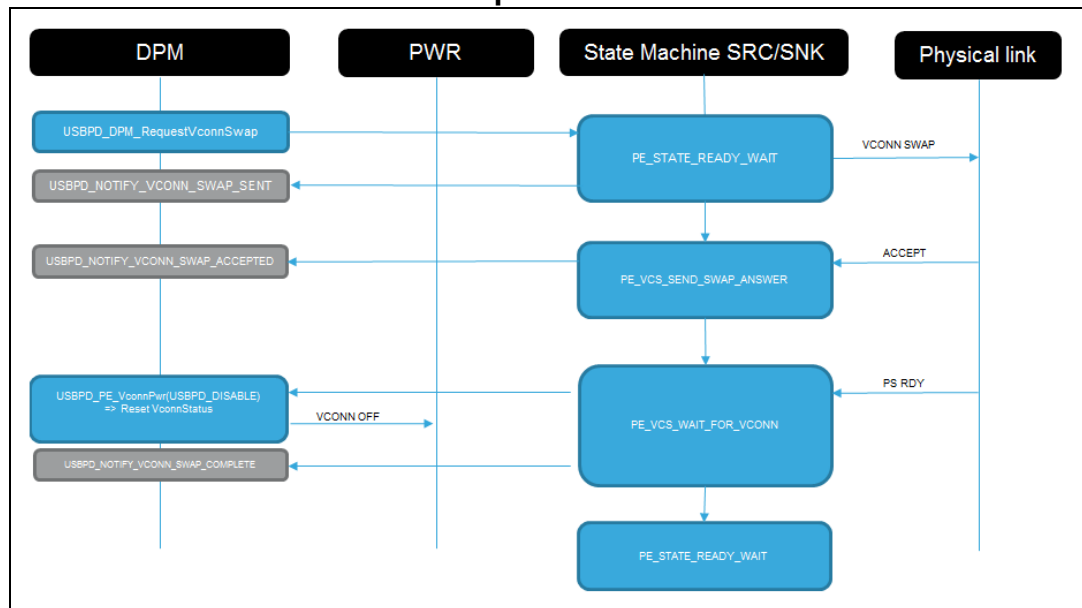
5.6.1 Initiated VCONN Swap, rejected by port partner

Figure 31. Initiated VCONN Swap, rejected by port partner



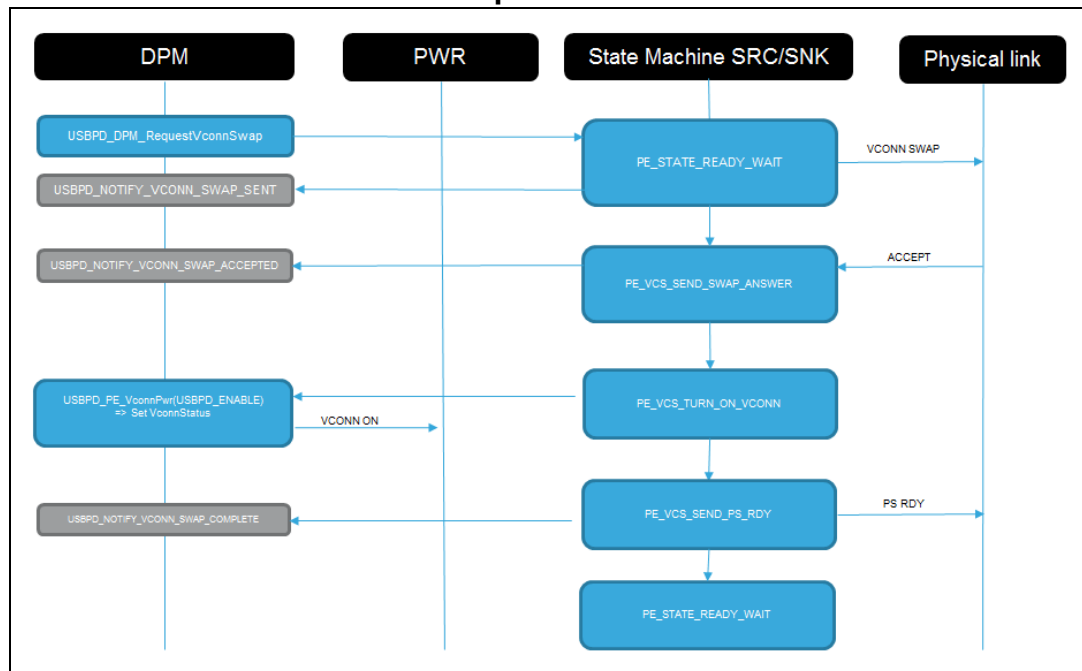
5.6.2 VCONN Swap initiated by VCONN source, accepted by port partner

Figure 32. VCONN Swap initiated by VCONN source, accepted by port partner



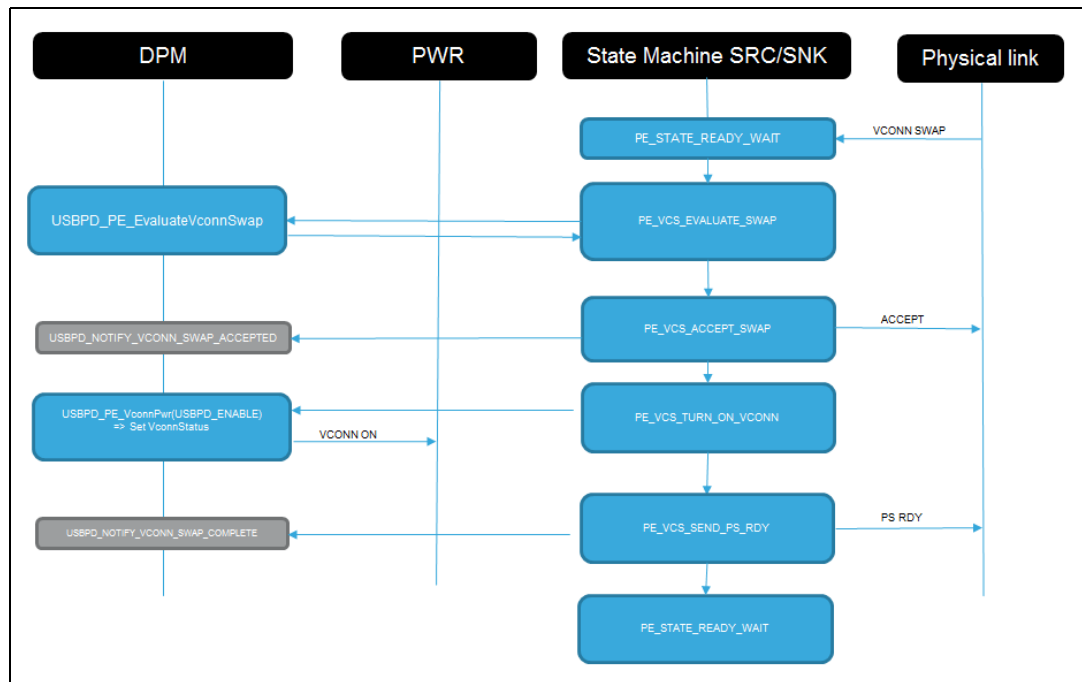
5.6.3 VCONN Swap initiated by VCONN sink, accepted by port partner

Figure 33. VCONN Swap initiated by VCONN sink, accepted by port partner



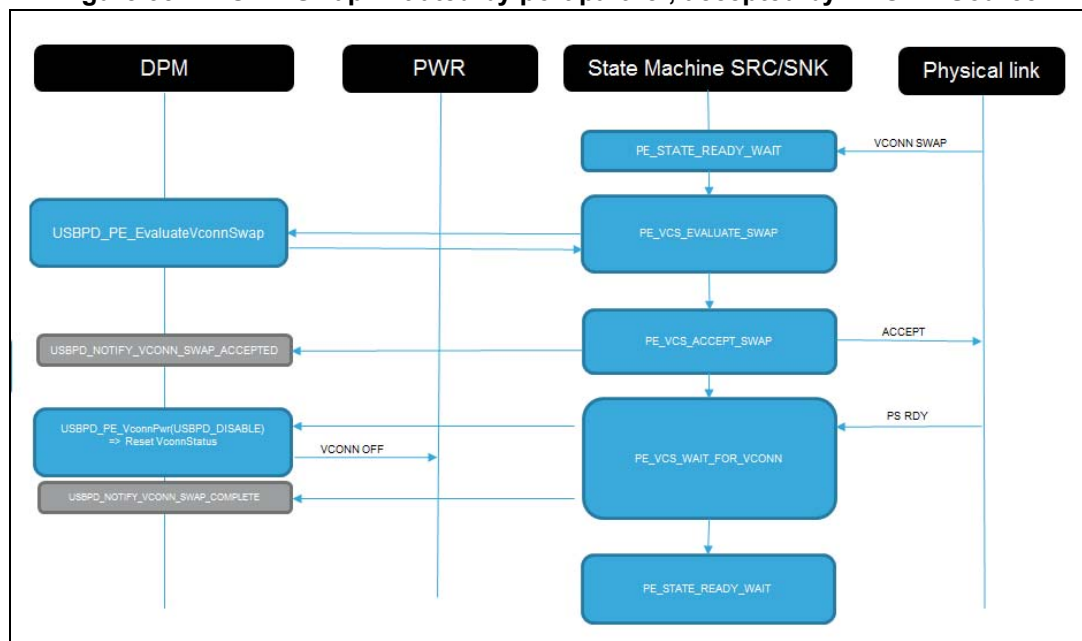
5.6.4 VCONN Swap initiated by port partner, accepted by VCONN sink

Figure 34. VCONN Swap initiated by port partner, accepted by VCONN sink



5.6.5 VCONN Swap initiated by port partner, accepted by VCONN Source

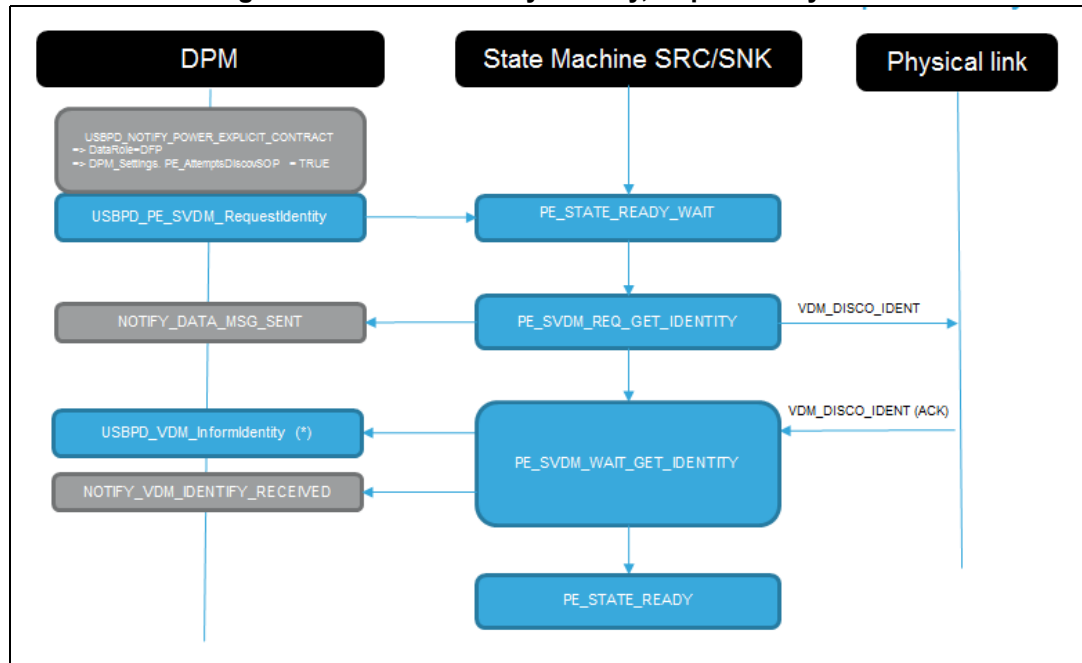
Figure 35. VCONN Swap initiated by port partner, accepted by VCONN Source



5.7 VDM sequences

5.7.1 VDM Discovery Identify, requested by DFP

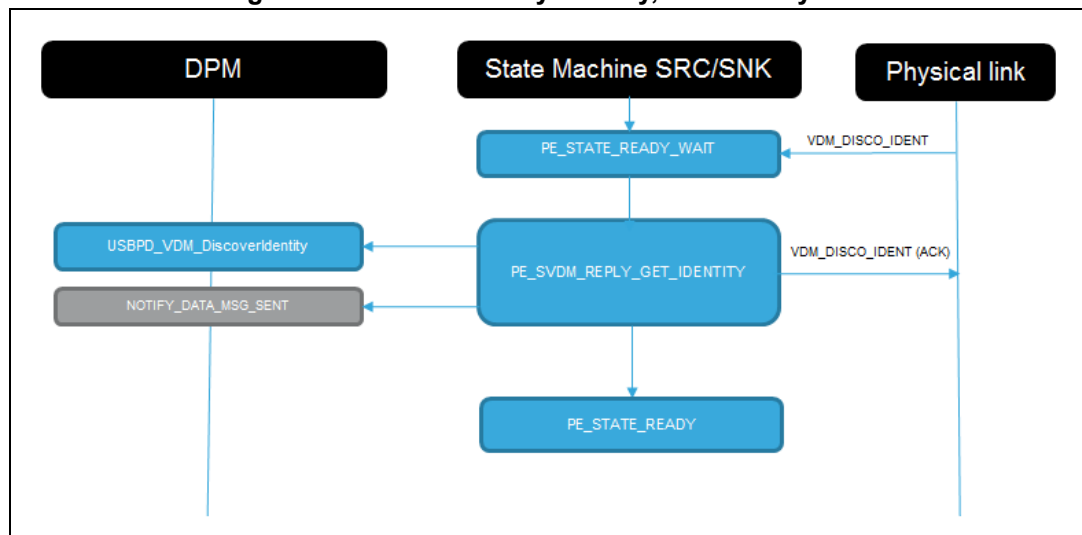
Figure 36. VDM Discovery Identify, requested by DFP



(*) In VDM callback, VDM_DISCO_SVID is posted in the USB-PD stack

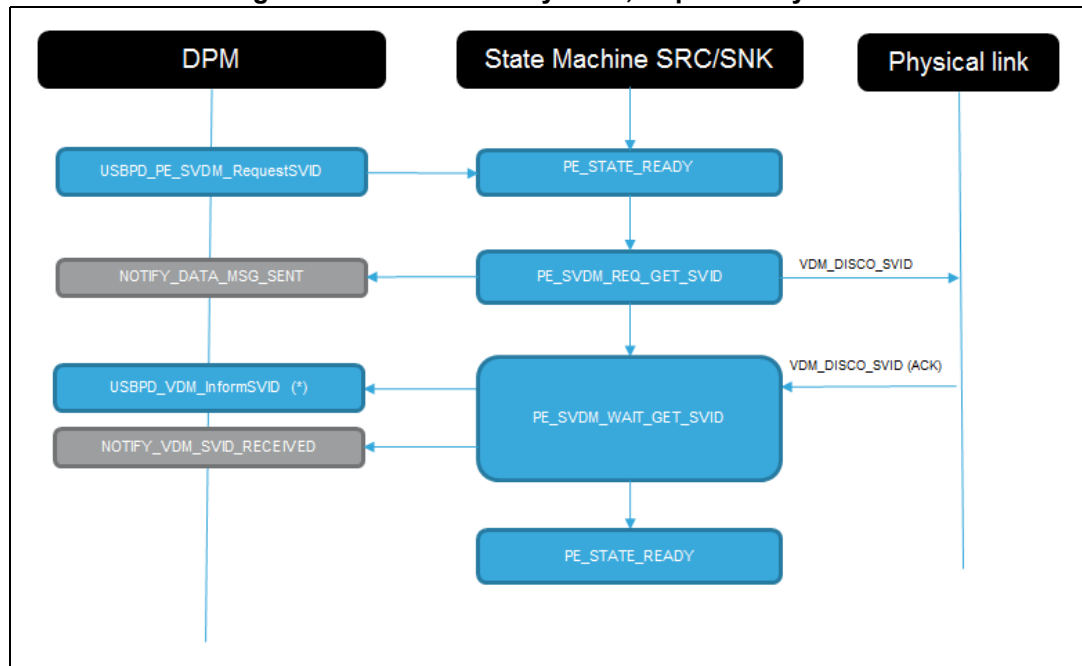
5.7.2 VDM Discovery Identity, received by UFP

Figure 37. VDM Discovery Identity, received by UFP



5.7.3 VDM Discovery SVID, requested by DFP

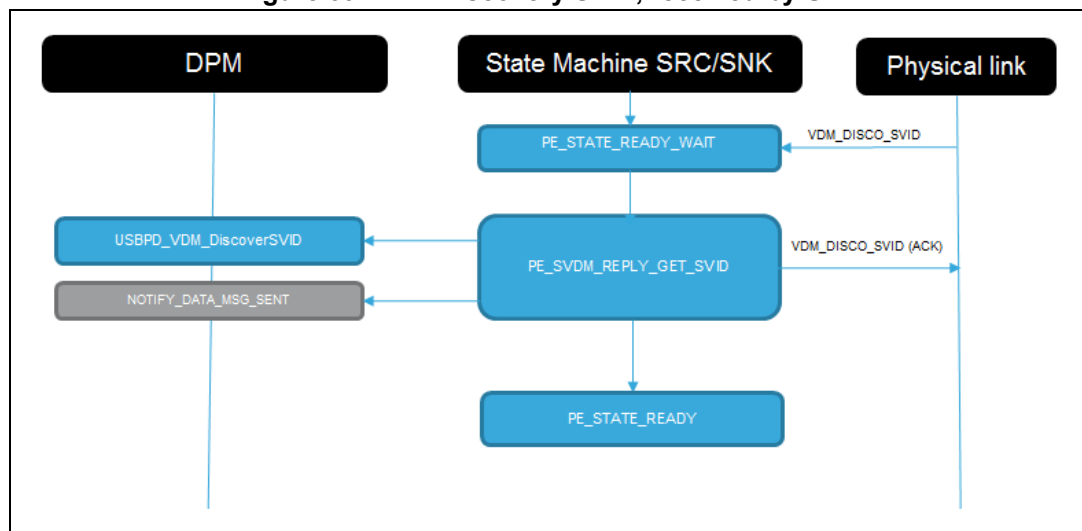
Figure 38. VDM Discovery SVID, requested by DFP



(*) In VDM callback, `VDM_DISCO_MODE` is posted in the USB-PD stack

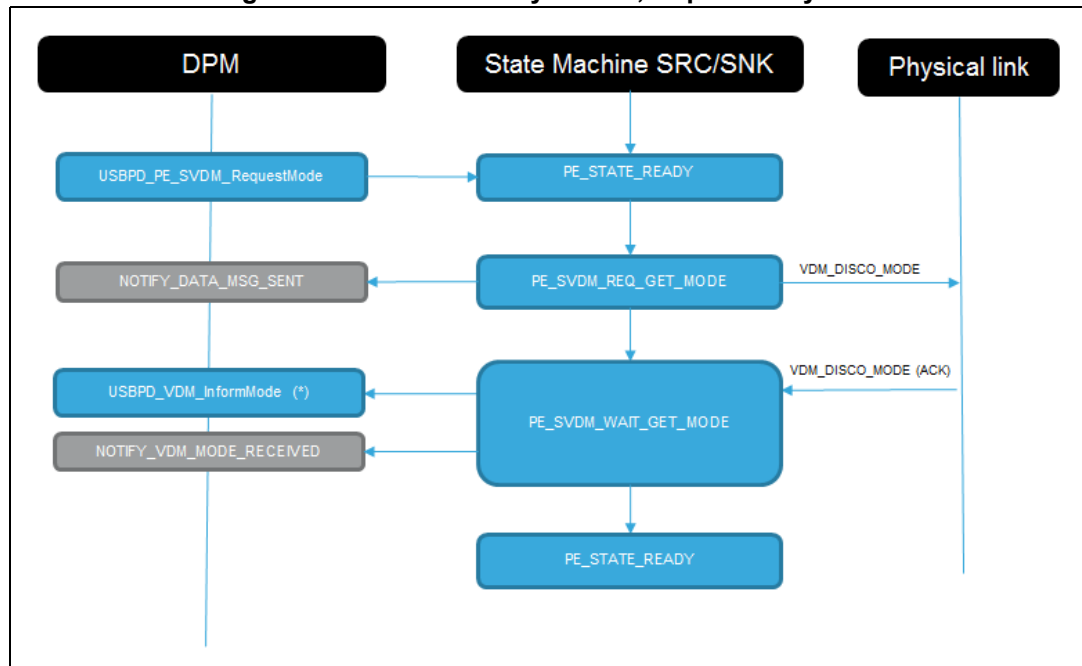
5.7.4 VDM Discovery SVID, received by UFP

Figure 39. VDM Discovery SVID, received by UFP



5.7.5 VDM Discovery Modes, requested by DFP

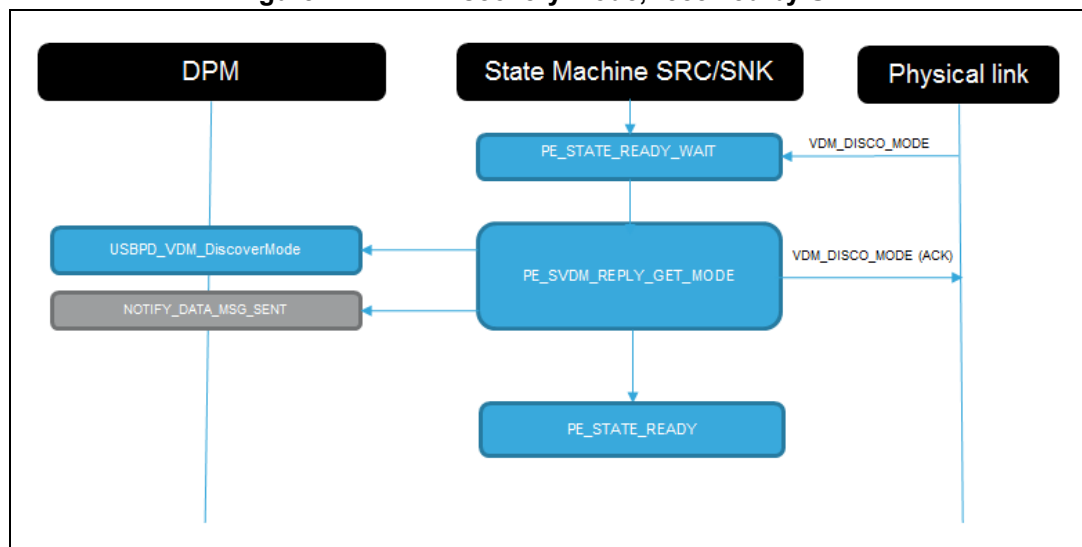
Figure 40. VDM Discovery Modes, requested by DFP



(*) In VDM callback, VDM_ENTER_MODE is posted in the USB-PD stack

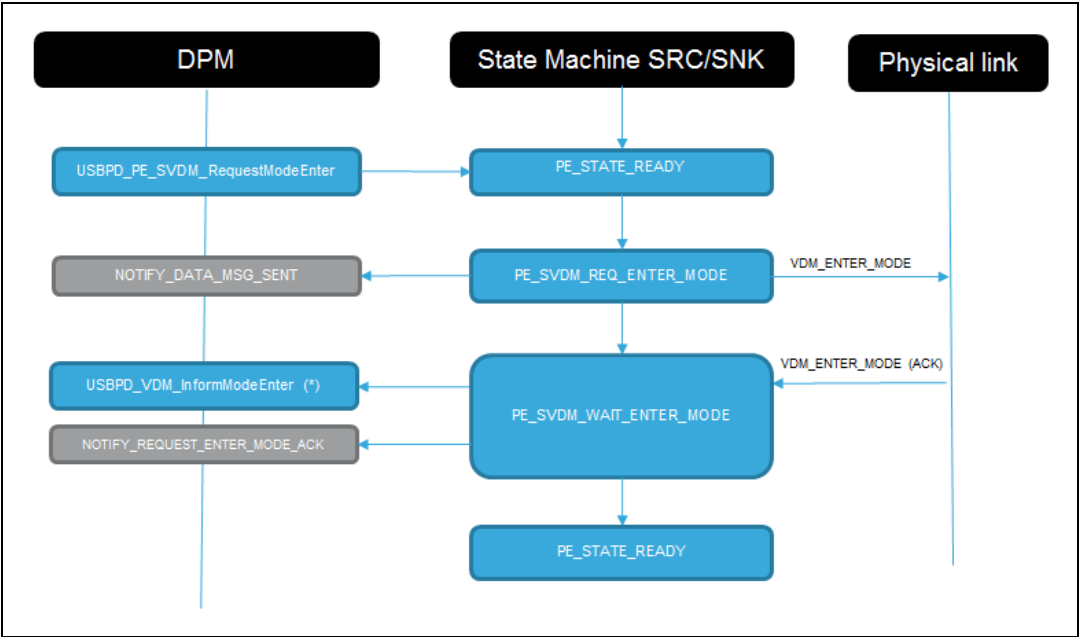
5.7.6 VDM Discovery Mode, received by UFP

Figure 41. VDM Discovery Mode, received by UFP



5.7.7 VDM Enter Mode, requested by DFP

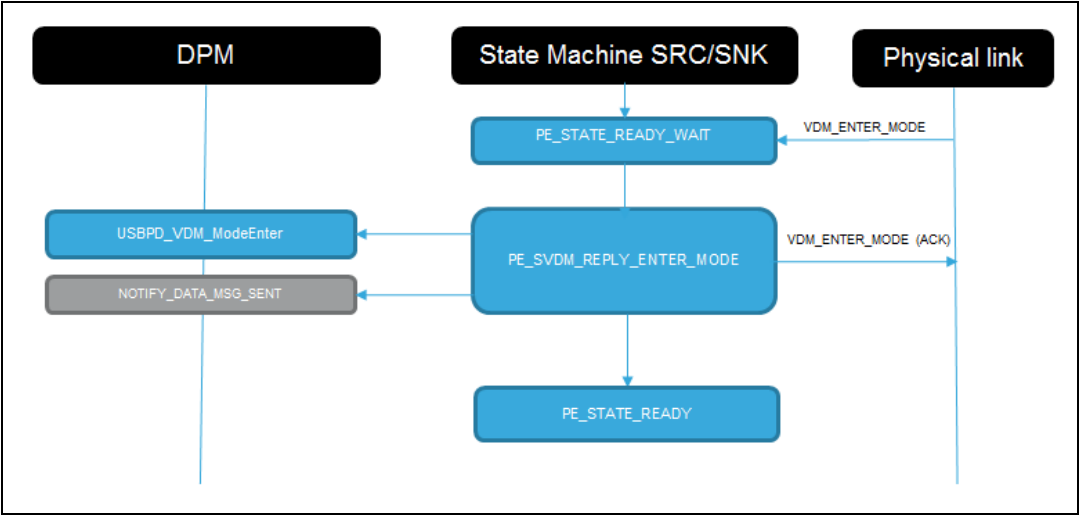
Figure 42. VDM Enter Mode, requested by DFP



(*) In VDM callback, DP STATUS may be posted in the USB-PD stack

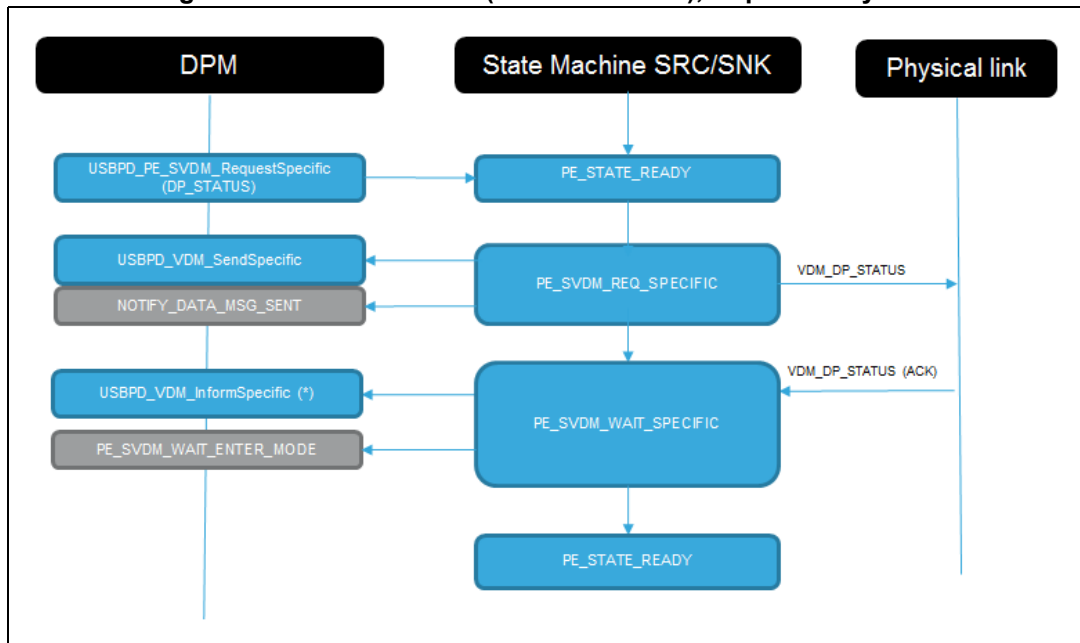
5.7.8 VDM Enter Mode, received by UFP

Figure 43. VDM Enter Mode, received by UFP



5.7.9 VDM SPECIFIC, requested by DFP

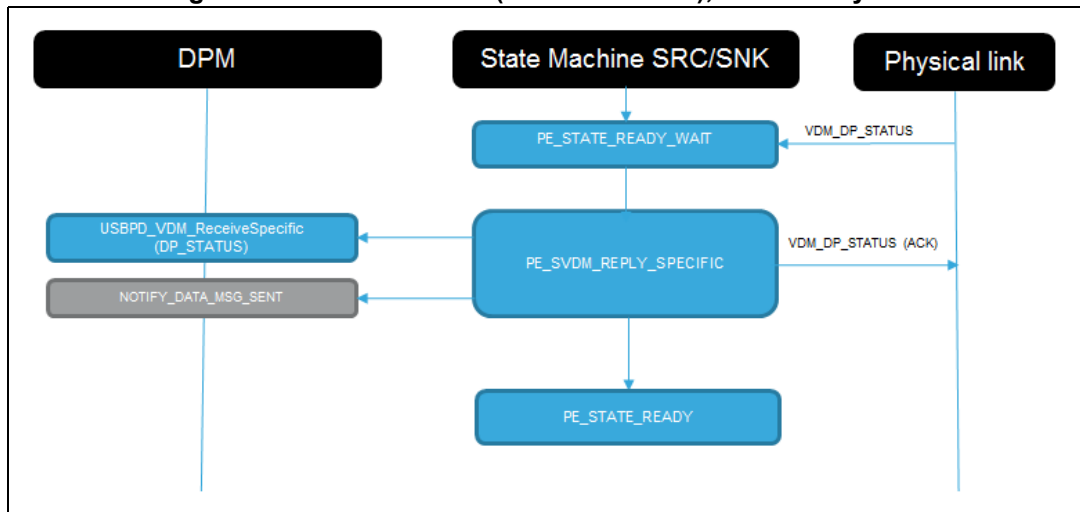
Figure 44. VDM SPECIFIC (ex: DP STATUS), requested by DFP



1. In VDM callback, DP CONFIG may be posted in the USB-PD stack

5.7.10 VDM SPECIFIC, received by UFP

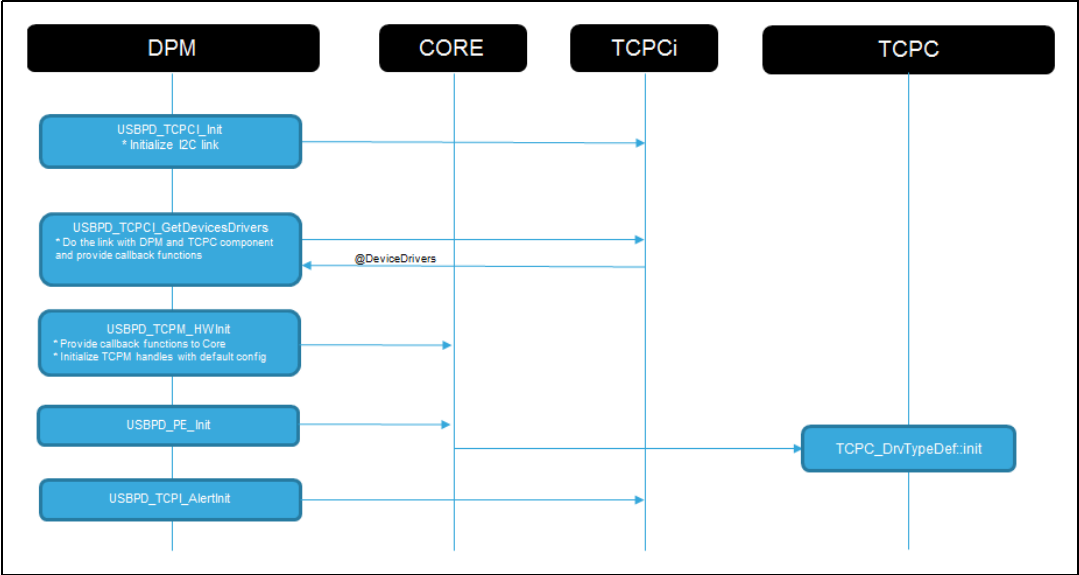
Figure 45. VDM SPECIFIC (ex: DP STATUS), received by UFP



5.8 TCPM

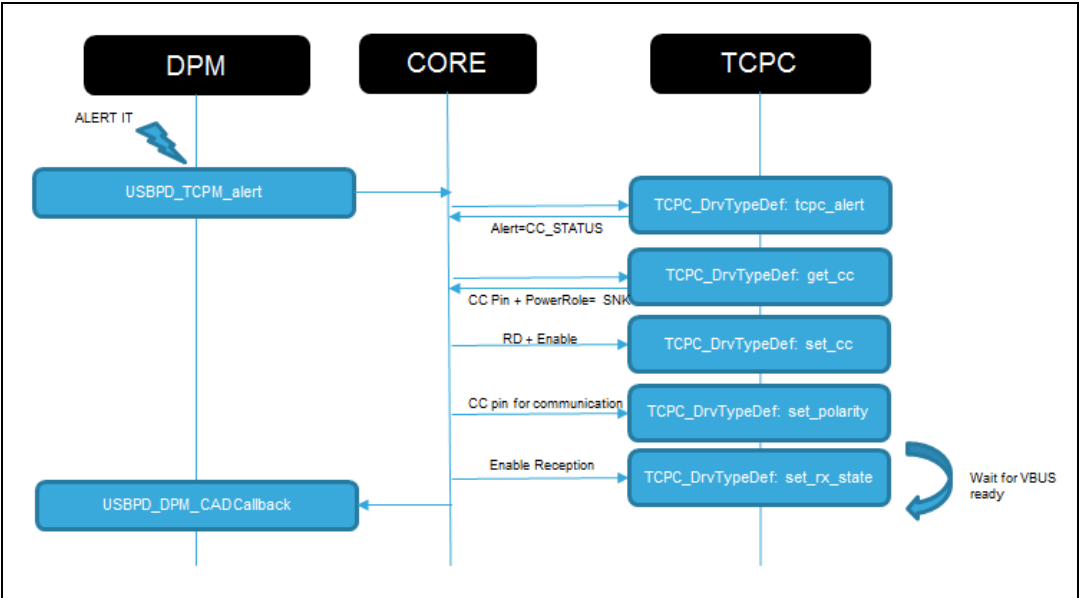
5.8.1 TCPC initialization

Figure 46. TCPC initialization



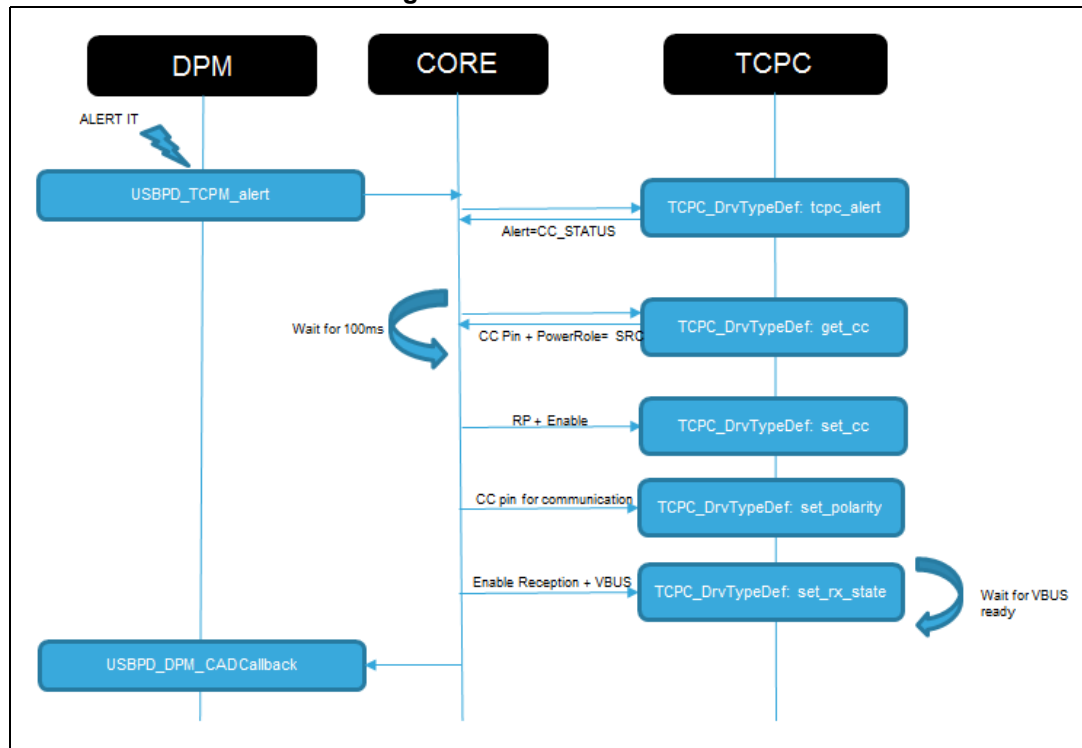
5.8.2 SNK connection

Figure 47. SNK connection



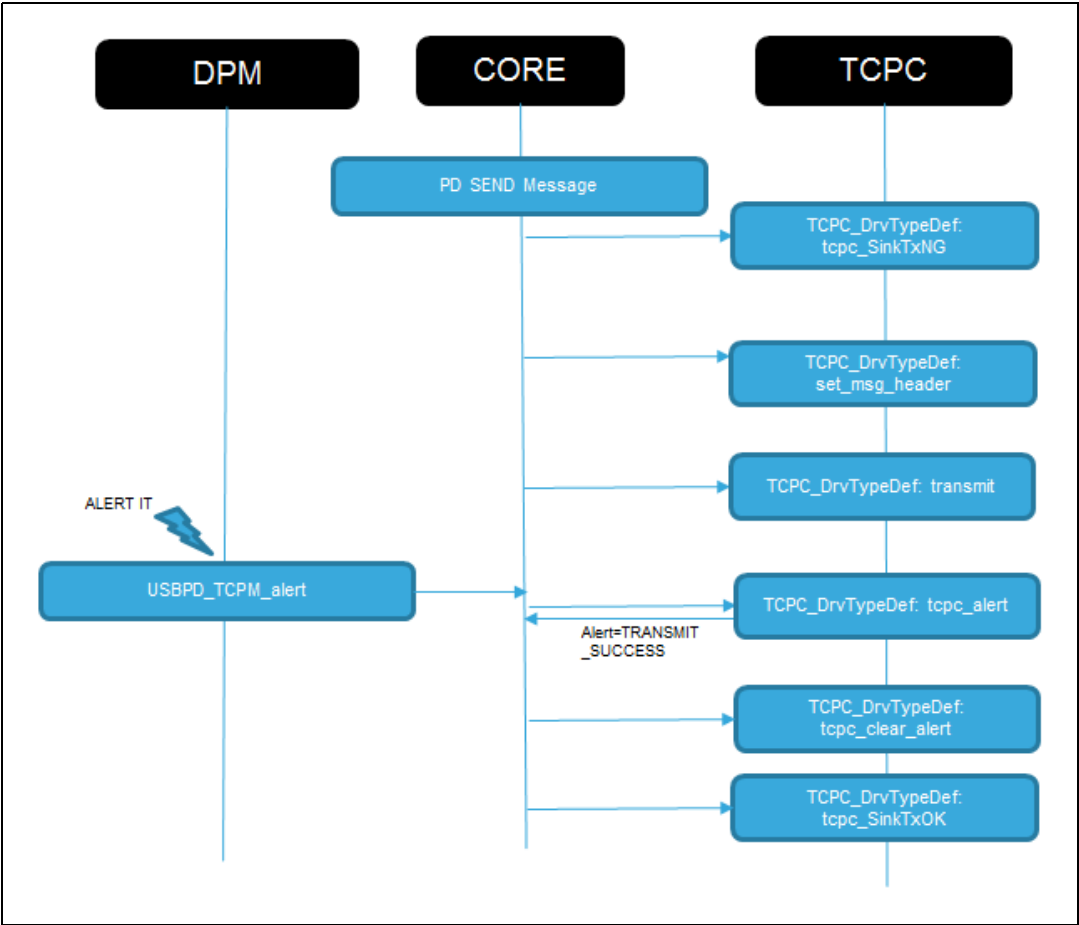
5.8.3 SRC connection

Figure 48. SRC connection



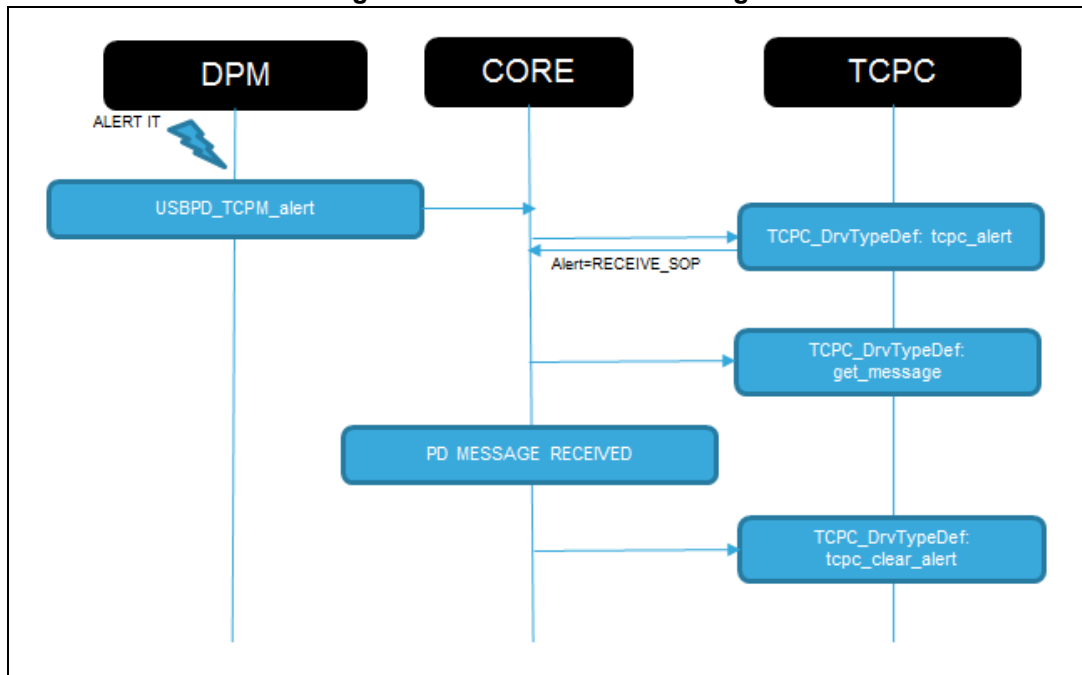
5.8.4 TCPC Transmit message

Figure 49. TCPC Transmit message



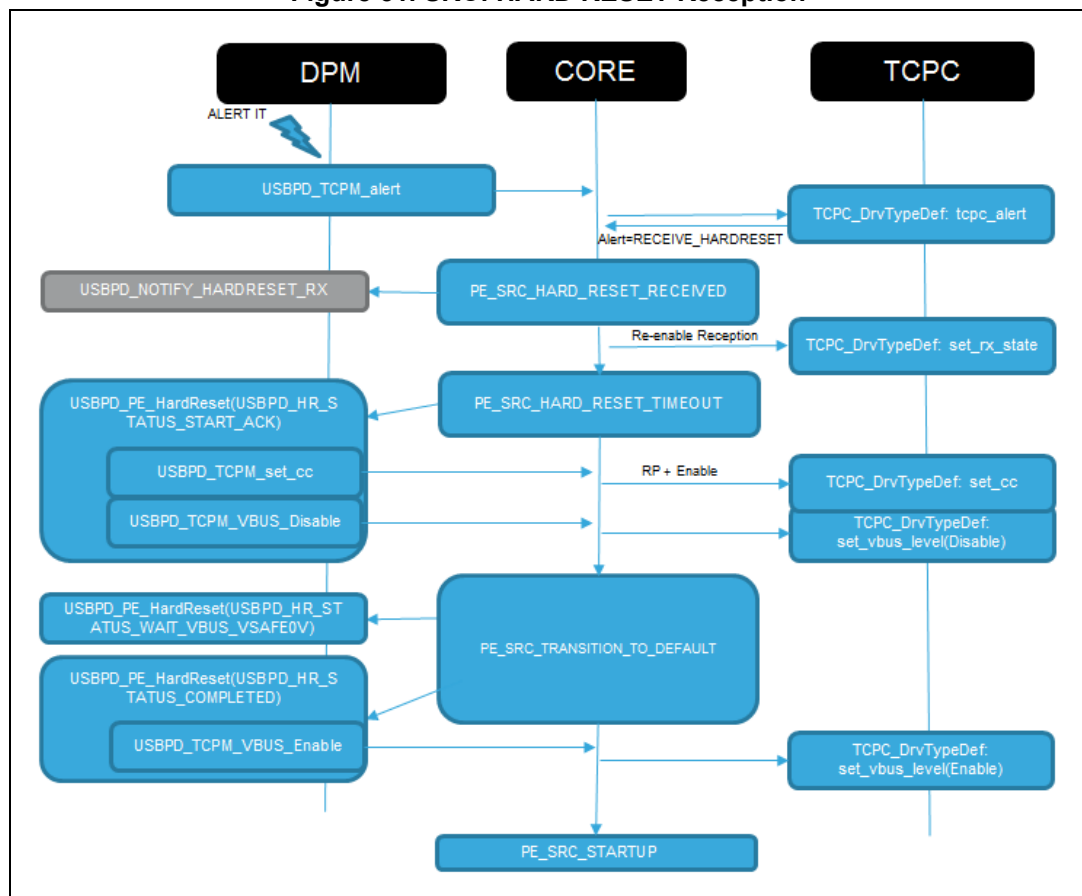
5.8.5 TCPC Receive message

Figure 50. TCPC Receive message



5.8.6 SRC: HARD RESET Reception

Figure 51. SRC: HARD RESET Reception



6 Examples description

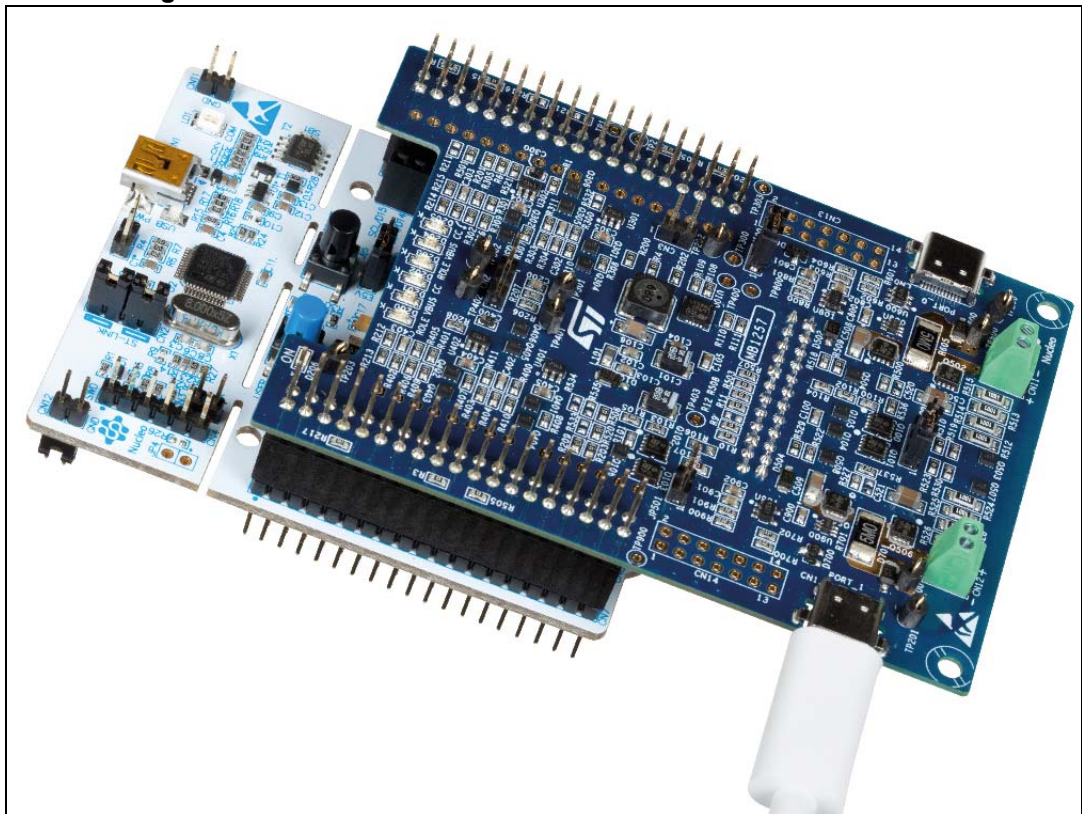
6.1 Hardware description

The P-NUCLEO-USB001 shield is needed as an expansion board for the STM32F072RB Nucleo in order to use the library. The P-NUCLEO-USB001 shield is an expansion board with two USB Type-C connectors for management of two ports, with the following features:

- two dual role ports
- dedicated power connector to interface with external power supply board providing different profiles (up to 20 V and 5 A) and V_{CONN}
- on-board power management, able to provide internal needed voltages from V_{BUS}
- six debug LEDs
- USB 2.0 interface capability available on one port
- compatible with STM32 Nucleo boards
- equipped with ST Morpho connectors

The P-NUCLEO-USB001 shield must be connected to the CN7 and CN10 connectors of the Nucleo board.

Figure 52. STM32F072RB Nucleo with P-NUCLEO-USB001 shield



For all the provided examples, the user has to

1. open its preferred tool chain
2. rebuild all files and load the image into the target memory
3. run the application.

The USB-PD applications can be used with any USB-C device that is PD-capable, according to its specific power role (provider, consumer, DRP).

6.2 USB-PD provider

This project implements an USB-PD provider-only application suitable for STM32 Nucleo pack for USB Type-C and Power Delivery (STM32F072 Nucleo board and USB-C PD expansion board MB1257), based on the use of USB-PD libraries delivered in the X-CUBE-USB-PD Expansion Package.

6.2.1 Example setup

The USB-PD provider application can be found under
Projects\STM32F072RB-Nucleo\Applications\USB_PD\Provider_RTOS

The provider role can be managed with two different supply options, corresponding to two configuration settings.

1. The provider is supplied by the on-board STM32F072RB-Nucleo RevC voltage regulator, by means of an USB Type-A to Mini-B cable plugged to the CN1 connector and then to a PC.
 - a) On STM32F072RB-Nucleo RevC board, verify that jumper JP1 is open, JP5 (PWR) closed on U5V (fitting the pins 1-2), and JP6 (IDD) closed.
 - b) On MB1257 expansion board, close the jumper related to the provider port (J500 for PORT_0 and JP501 for PORT_1).

This setting allows to manage the V_{BUS} on the selected port, starting from the STM32F072RB-Nucleo RevC USB PWR voltage (CN1 connector).

2. The provider is equipped with an external board by power connector CN4:
 - a) On the STM32F072RB-Nucleo RevC board, the following jumper settings must be guaranteed: JP1 closed, JP5 (PWR) closed on E5V (fitting the pins 2-3), and JP6 (IDD) closed.
 - b) On the P-NUCLEO-USB001 expansion board, the jumpers JP100, J500 and JP501 must be left open.

This setting configuration allows the external power board to supply the whole system, and, particularly for the USB PD application, to offer a voltage level for the V_{BUS} of the port.

The application can be tested with a connection (through an USB-C Power Delivery cable connected to CN0 in the X-NUCLEO shield) to an USB-C Power Delivery consumer device.

A second board loaded with the Consumer_RTOS application can be used for acting as a connected consumer device.

6.2.2 Application description

This application provides an example for managing the Port0 as a provider-only port.

When the application starts, connecting an USB-C Power Delivery consumer device (sink mode) triggers the power negotiation:

- At start, Role LED (LED D203) blinks, indicating Port role (blinking only once indicates provider role).
- User has to plug the USB-C cable on the dedicated connector. When attached, CC LEDs (D205) blinks once if connected to CC1, twice if connected to CC2.
- Blue LED (D203) blinks once, to show that the device behaves as a provider.
- The STM32 MCU behaves as a provider (source mode), it exchanges Power profiles with the connected device and waits for Power Request message from the attached consumer.
- When attached, and before Explicit Contract is established, V_{BUS} LED (D204) blinks.
- If the requested power can be met, the STM32 MCU sends the Accept message followed by PS_RDY message.
- Once the Explicit Contract is established, V_{BUS} LED (D204) is on to indicate that the Power Contract has been established.

6.3 USB-PD provider (with CLI support)

This application is similar to USB-PD provider application, so the description provided in [Section 6.2](#) applies also to this project, with the difference that support of the Command Line interface (CLI) is enabled.

This application embeds a Command Line Interface (CLI) feature, which allows user to get status of the Power Delivery application running on Port0 and to act on it through a serial communication link.

Refer to UM2051 “Getting started with the STM32 Nucleo pack for USB Type-C™ and Power Delivery”, available on www.st.com, for more details on:

- boards configuration for CLI use
- HyperTerminal configuration (on PC side)
- available CLI command lines and parameters.

In this application, only one port (Port0) is handled and is then accessed by CLI commands.

The USB-PD provider (with CLI support) application can be found under Projects\STM32F072RB-Nucleo\Applications\USB_PD\Provider_CLI_RTOS

6.4 USB-PD provider (with VDM support)

This application is similar to USB-PD provider application, so the description provided in [Section 6.2](#) applies also to this project, with the difference that support of Vendor Defined Messages is enabled.

In addition to managing the Port0 as a provider-only port, this application provides an example of VDM messages use to enable information exchanges between devices, outside of USB-PD related messages. Provider application has been implemented to generate SVDM messages. VDM callbacks are provided as examples.

These callbacks allow the user to simulate an entry in DP mode (all the LEDs toggle when an enter mode is accepted).

To test this application for Power Delivery part, the Consumer_RTOS application can be used as a consumer, on a second board.

Vendor Defined Messages can be tested when connected to a device corresponding to Port0 capabilities (complete definition of the example is available in the provided VIF file) or using an USB PD conformance tester supporting VDM feature.

The USB-PD provider supporting Vendor Defined Messages application can be found under *Projects\STM32F072RB-Nucleo\Applications\USB_PD\Provider_VDM_RTOS*.

6.4.1 USB-PD provider (Low footprint version)

This project implements an USB-PD provider-only application suitable for STM32 Nucleo pack for USB Type-C and Power Delivery (STM32F072 Nucleo board and USB-C PD expansion board MB1257), based on the use of USB-PD libraries delivered in the X-CUBE-USB-PD expansion package, and highlighting the small Ram/Rom consumption of the STMicroelectronics USBPD solution. As target of this example is to reach small Ram/Rom footprint, LED management has been disabled in this project. FreeRTOS is also not used anymore in this configuration.

6.4.2 Example setup

The USB-PD provider application can be found under *Projects\STM32F072RB-Nucleo\Applications\USB_PD\Provider_RTOS*

The provider role can be managed with two different supply options, corresponding to two configuration settings.

1. The provider is supplied by the on-board STM32F072RB-Nucleo RevC voltage regulator, by means of an USB Type-A to Mini-B cable plugged to the CN1 connector and then to a PC.
 - a) On STM32F072RB-Nucleo RevC board, verify that jumper JP1 is open, JP5 (PWR) closed on U5V (fitting the pins 1-2), and JP6 (IDD) closed.
 - b) On MB1257 expansion board, close the jumper related to the provider port (J500 for PORT_0 and JP501 for PORT_1).

This setting allows to manage the VBUS on the selected port, starting from the STM32F072RB-Nucleo RevC USB PWR voltage (CN1 connector).

2. The provider is equipped with an external board by power connector CN4:
 - a) On the STM32F072RB-Nucleo RevC board, the following jumper settings must be guaranteed: JP1 closed, JP5 (PWR) closed on E5V (fitting the pins 2-3), and JP6 (IDD) closed.
 - b) On the P-NUCLEO-USB001 expansion board, the jumpers JP100, J500 and JP501 must be left open.

This setting configuration allows the external power board to supply the whole system, and, particularly for the USB PD application, to offer a voltage level for the VBUS of the port.

The application can be tested with a connection (through an USB-C Power Delivery cable connected to CN0 in the X-NUCLEO shield) to an USB-C Power Delivery consumer device.

A second board loaded with the Consumer_RTOS application can be used for acting as a connected consumer device.

6.4.3 Application description

This application provides an example for managing the Port 0 as a Provider Only port. When the application starts, connecting an USB-C Power Delivery consumer device (Sink mode) should trigger the power negotiation. In order to reduce RAM/ROM consumption, LED management used in other examples for indicating current role, CC orientation, and explicit contract establishment, has been removed:

- User should plug the USB-C cable on the dedicated connector.
- When attached, the STM32 MCU behaves as a Provider (Source mode), it exchanges Power profiles with the connected device and waits for Power Request message from the attached consumer
- If the requested power can be met, the STM32 MCU shall send the Accept message followed by PS_RDY message

6.5 USB-PD consumer

This project implements an USB-PD consumer-only application suitable for STM32 Nucleo pack for USB Type-C and Power Delivery (STM32F072 Nucleo board and USB-C PD expansion board MB1257), based on use of USB-PD libraries delivered in X-CUBE-USB-PD Expansion Package.

6.5.1 Example setup

The USB-PD consumer application can be found under
Projects\STM32F072RB-Nucleo\Applications\USB_PD\Consumer_RTOS.

The system can manage two supply options for the consumer configuration. The first one is supplied by NUCLEO-F072RB, while the second implements a specific feature of the USB PD solutions (i.e. when a consumer is supplied by the provider by mean of its V_{BUS}).

Both configurations correspond to two different settings:

- If the consumer is supplied by the NUCLEO-F072RB voltage regulator, the system setting is the following one:
 - On NUCLEO-F072RB board, verify that the jumper JP1 is open, JP5 (PWR) closed on U5V (fitting the pins 1-2), and JP6 (IDD) closed.
 - On P-NUCLEO-USB001 expansion board, open the jumpers JP100, J500, JP501.
- If the consumer is supplied by the V_{BUS} delivered by the provider attached by the USB Type-C cable, the system setting is the following one:
 - On the NUCLEO-F072RB board, the jumper JP1 must be closed, JP5 (PWR) closed on E5V (fitting the pins 2-3), and JP6 (IDD) closed.
 - On P-NUCLEO-USB001 expansion board, while the jumpers J500, JP501 are opened, the jumper JP100 must be set according to the port chosen for supplying the system (fit 2-3 for PORT_0 or 1-2 for PORT_1).

The application can be tested with a connection (through an USB-C Power Delivery cable connected to CN0 in the X-NUCLEO shield) to an USB-C Power Delivery consumer device.

A second board loaded with the Provider_RTOS application can be used for acting as this connected provider device.

6.5.2 Application description

When the application starts, connecting an USB-C Power Delivery provider device (source mode) triggers the power negotiation:

- At start, Role LED (LED D203) blinks indicating Port role (blinking twice indicates consumer Role).
- User has to plug the USB-C cable on the dedicated connector.
- When attached, CC LEDs (D205) blinks once if connected to CC1, twice if connected to CC2.
- Blue LED (D203) blinks twice each time, to show that the device behaves as a consumer.
- The STM32 MCU behaves as a consumer (sink mode), it waits for Power Capabilities message from the attached provider. When a Source Capabilities message is received, the STM32 starts the evaluation of the received capabilities and checks if one of the received power objects can meet its power requirement.
- While communicating, V_{BUS} LED (D204) blinks.
- The STM32 sends a message to request the new power level from the offered Source Capabilities.
- Once the Explicit Contract is established (PS_Ready message received), V_{BUS} LED (D204) is ON to indicate that the Power Contract has been established.

6.6 USB-PD consumer (with CLI support)

This application is similar to USB-PD consumer application, so the description provided in [Section 6.5](#) applies also to this project, with the difference that support of Command Line interface (CLI) is enabled.

This application embeds a Command Line Interface (CLI) feature, which allows user to get status of the Power Delivery application running on Port0 and to act on it through a serial communication link.

Refer to UM2051 “Getting started with the STM32 Nucleo pack for USB Type-C™ and Power Delivery”, available on www.st.com, for more details on:

- boards configuration for CLI use
- HyperTerminal configuration (on PC side)
- available CLI command lines and parameters.

In this application, only one port (Port0) is handled and then accessed by CLI commands.

The USB-PD consumer (with CLI support) application can be found under *Projects\STM32F072RB-Nucleo\Applications\USB_PD\Consumer_CLI_RTOS*.

6.7 USB-PD consumer (with VDM support)

This application is similar to USB-PD consumer application, so the description provided in [Section 6.5](#) applies also to this project, with the difference that support of Vendor Defined Messages is enabled in this project.

In addition to managing the Port0 as a consumer-only port, this application provides an example of VDM messages use to allow information exchanges between devices, outside of

USB-PD related messages. Consumer application has been implemented to answer to SVDM messages. VDM callbacks are provided as example.

These callbacks allow the user to simulate an entry in DP mode (all the LEDs toggle when an enter mode is accepted).

To test this application for Power Delivery part, the Provider_RTOS application can be used as a provider (Source), on a second board.

Vendor Defined Messages can be tested when connected a device corresponding to Port0 capabilities (complete definition of example is present in provided VIF file), or using an USB PD conformance tester supporting VDM feature.

The USB-PD consumer supporting Vendor Defined Messages application can be found under *Projects\STM32F072RB-Nucleo\Applications\USB_PD\Consumer_VDM_RTOS*.

6.8 USB-PD consumer DRP

This project implements an USB-PD Dual Role Port (DRP) application suitable for STM32 Nucleo pack for USB Type-C and Power Delivery (STM32F072 Nucleo board and USB-C PD expansion board MB1257), based on use of USB-PD libraries delivered in the X-CUBE-USB-PD Expansion Package.

6.8.1 Example setup

The USB-PD Dual Role Port (DRP) application can be found under *Projects\STM32F072RB-Nucleo\Applications\USB_PD\Consumer_DRP_RTOS*.

The system can manage two supply options for the Port0, depending upon Role that Port is taking, after being connected to another device (either provider or consumer).

- System is supplied by the on-board STM32F072RB-Nucleo RevC voltage regulator, by mean of an USB Type-A to Mini-B cable plugged to the CN1 connector and then to a PC.
 - On STM32F072RB-Nucleo RevC board, verify that the jumper JP1 is open, JP5 (PWR) closed on U5V (fitting the pins 1-2), and JP6 (IDD) closed.
 - On MB1257 expansion board, close the jumper related to the Provider Port (J500 for PORT_0).

This setting enables to manage the V_{BUS} on the selected port, starting from the STM32F072RB-Nucleo RevC USB PWR voltage (CN1 connector).

- System is equipped with an external board by power connector CN4:
 - On the STM32F072RB-Nucleo RevC board, the following jumper settings must be guaranteed: JP1 closed, JP5 (PWR) closed on E5V (fitting the pins 2-3), and JP6 (IDD) closed.
 - On P-NUCLEO-USB001 expansion board, while the jumpers J500, JP501 are opened, the jumper JP100 must be set according to the port chosen for supplying the system (fit 2-3 for PORT_0 or 1-2 for PORT_1).

In provider case, this setting configuration allows the external power board to supply the entire system and, particularly for the USB PD application, to offer a voltage level for the V_{BUS} of the port.

In consumer case, consumer is supplied by mean of the V_{BUS} delivered by the provider attached by the USB Type-C cable.

The application can be tested with a connection (through an USB-C Power Delivery cable connected to CN0 in the X-NUCLEO shield) to an USB-C Power Delivery device, behaving as a consumer, a provider or as a DRP device.

A second board loaded with the corresponding application can be used.

6.8.2 Application description

This application provides an example for managing the Port0 as dual-role, behaving either as a provider or as consumer, according to the connected device. While connected, Power Role swap is also supported:

- When connected to an USB-C provider-only device (source mode), the power role swap between the two boards is not possible, DRP board acts automatically as a sink.
- When connected to an USB-C consumer-only device (sink mode), the power role swap between the two boards is not possible, DRP board acts automatically as a source.
- When connected to an USB-C with DRP, the power role swap is done each time user button is pressed.
- When the application starts, Port0 has the capability to operate either as Source or as Sink. When connecting an USB-PD device (source or sink) on Port0, the application is able to detect the type of connected device, and adopt a corresponding suitable role, in order to trigger the power negotiation:
- At start, Role LED (LED D203) blinks indicating Port role (blinking three times indicates DRP).
- User has to plug the USB-C cable on the dedicated connector.
- When STM32 MCU behaves as a consumer (sink mode), i.e. when connected to a source device, it waits for Power Capabilities message from the attached provider. When a Source Capabilities message is received, the STM32 starts the evaluation of the received capabilities and checks if one of the received power objects can meet its power requirement. The STM32 sends the Request message to request the new power level from the offered Source Capabilities. Once the Explicit Contract is established (PS_Ready message received), V_{BUS} LED (D204) is on to indicate that the Power Contract has been established.
- When STM32 MCU behaves as a provider (source mode), i.e. when connected to a Sink device, it exchanges Power profiles with the connected device and waits for Power Request message from the attached consumer. If the requested power can be met, the STM32 MCU sends the Accept message followed by PS_RDY message. Once the Explicit Contract is established, V_{BUS} LED (D204) is on to indicate that the Power Contract has been established.
- When attached, and before Explicit Contract is established, V_{BUS} LED (D204) blinks.
- When attached, CC LEDs (D205) blinks once if connected to CC1, twice if connected to CC2.
- Role (Blue) LED (D203) blinks twice each time if the device behaves as a consumer, or blinks once each time, if the device behaves as a provider.

6.9 USB-PD Dual Port

This project implements an USB-PD Dual Role Port application suitable for STM32 Nucleo pack for USB Type-C and Power Delivery (STM32F072 Nucleo board and USB-C PD

expansion board MB1257), based on use of USB-PD libraries delivered in X-CUBE-USB-PD Expansion Package.

6.9.1 Example setup

The USB-PD Dual Port application can be found under Projects\STM32F072RB-Nucleo\Applications\USB_PD\DUAL_PORT_RTOS

The system can manage two supply options for the Port0 and Port1, depending upon the role that the ports is taking, after being connected to another device (either provider or consumer).

- System is supplied by the on-board STM32F072RB-Nucleo RevC voltage regulator, by mean of an USB Type-A to Mini-B cable plugged to the CN1 connector and then to a PC
 - on STM32F072RB-Nucleo RevC board, verify that the jumper JP1 is open, JP5 (PWR) closed on U5V (fitting the pins 1-2), and JP6 (IDD) closed.
 - on MB1257 expansion board, close the jumper related to the provider Port (J500 for PORT_0, J501 for PORT_1).

This setting allows to manage the V_{BUS} on the selected port, starting from the STM32F072RB-Nucleo RevC USB PWR voltage (CN1 connector).

- System is equipped with an external board by power connector CN4
 - on the STM32F072RB-Nucleo RevC board, the following jumper settings must be guaranteed: JP1 closed, JP5 (PWR) closed on E5V (fitting the pins 2-3), and JP6 (IDD) closed.
 - on P-NUCLEO-USB001 expansion board, while the jumpers J500 and JP501 are opened, the jumper JP100 must be set according to the port chosen for supplying the system (fit 2-3 for PORT_0 or 1-2 for PORT_1).

In provider case, this setting configuration allows the external power board to supply the entire system and, particularly for the USB PD application, to offer a voltage level for the V_{BUS} of the port.

In consumer case, consumer is supplied by mean of the VBUS delivered by the provider attached by the USB Type-C cable.

The application can be tested with a connection (through an USB-C Power Delivery cable connected to CN0 in the X-NUCLEO shield) to an USB-C Power Delivery device, behaving as a consumer, a provider or as a DRP device.

A second board loaded with the corresponding application can be used.

6.9.2 Application description

This application provides an example for managing both Port0 and Port1 simultaneously. Both ports are DRP, and behave independently. User has to plug the USB-C cable on the dedicated connector (Port0: CN0 or Port1: CN1).

When the application starts, each port has the capability of operating either as a Source or as a Sink. When connecting an USB-C Power Delivery device (source or sink) on a given

port, the application is able to detect type of connected device, and adopt corresponding suitable role, in order to trigger the power negotiation:

- At start, Role LED (LED D203 for Port0 and LED D200 for Port1) blinks, indicating the port role (three times blink indicates DRP Role).
- User has to plug the USB-C cable on the dedicated connector (CN0 for Port0 and CN1 for Port1).
- When the port behaves as a consumer (sink mode), i.e. when connected to a Source device, it waits for Power Capabilities message from the attached provider.
- When a Source Capabilities message is received, the STM32 starts the evaluation of the received capabilities and checks if one of the received power objects can meet its power requirement. The STM32 then sends the Request message to request the new power level from the offered Source Capabilities. Once the Explicit Contract is established (PS_Ready message received), V_{BUS} LED (LED D204 for Port0 and LED D201 for Port1) is on to indicate that the Power Contract has been established.
- When the port behaves as a provider (source mode), i.e. when connected to a Sink device, it exchanges Power profiles with the connected device and waits for Power Request message from the attached consumer. If the requested power can be met, the STM32 MCU sends the Accept message followed by PS_RDY message. Once the Explicit Contract is established, V_{BUS} LED (LED D204 for Port0 and LED D201 for Port1) is on to indicate that the Power Contract has been established.
- When attached, and before Explicit Contract is established, V_{BUS} LED (LED D204 for Port0 and LED D201 for Port1) blinks.
- When attached, CC LEDs (LED D205 for Port0 and LED D202 for Port1) blink once if connected to CC1, twice if connected to CC2.
- Role (Blue) LED (LED D203 for Port0 and LED D200 for Port1) blinks twice each time if the device behaves as a consumer, or blinks once each time if the device behaves as a provider.

This application also embeds a Command Line Interface (CLI) feature, which allows the user to get the status of Power Delivery application running on Port0 and Port1 and to interact with the application through a serial communication.

Refer to UM2051 “Getting started with the STM32 Nucleo pack for USB Type-C™ and Power Delivery”, available on www.st.com, for more details on

- boards configuration for CLI use
- HyperTerminal configuration (on PC side)
- available CLI command lines and parameters.

In this application, both ports (Port0 and Port1) can be accessed by CLI commands.

6.10 USB-PD DRP (TCPM/TCPC architecture)

This project implements an USB-PD DRP application, suitable for OnSemiconductor FUSB307 evaluation board, based on the use of USB-PD libraries delivered in the X-CUBE-USB-PD expansion package, and highlighting the TCPM/TCPC architecture.

6.10.1 Example setup

The USB-PD TCPM/TCPC designed for FUSB307 evaluation board application can be found under Projects\STM32F072RB-Nucleo\Applications\USB_PD\EVAL_FUSB307_DRP

This application provides an example of USB Power Delivery implementation based on TCPM/TCPC architecture. Application and USBPD Core Stack are located on TCPM side, and are running on STM32F0xx Nucleo device on OnSemiconductor FUSB307 board. Application is driving TCPC controller through I2C link.

6.10.2 Application description

When the application starts, USB-PD has capability of operating as either a Source or Sink.

When connecting to an USB-PD device (source or sink), application should be able to detect type of connected device, and adopt corresponding suitable role, in order to trigger the power negotiation:

- User should plug the USB-C cable on the dedicated connector.
- When STM32 MCU side behaves as a Consumer (Sink mode), i.e. when connected to a Source device, it waits for Power Capabilities message from the attached provider. When a Source Capabilities message is received, the STM32 starts the evaluation of the received capabilities and check if one of the received power objects can meet its power requirement. The STM32 shall send the Request message to request the new power level from the offered Source Capabilities. Once the PS_RDY message is received, Explicit Contract is established.
- When STM32 MCU side behaves as a Provider (Source mode), i.e. when connected to a Sink device, it exchanges Power profiles with the connected device and waits for Power Request message from the attached consumer. If the requested power can be met, the STM32 MCU shall send the Accept message followed by PS_RDY message. Explicit Contract is then considered as established.

7 Memory footprint

The values in [Table 6](#) to [Table 8](#) are calculated according to the following configuration:

- Compiler: IAR Embedded Workbench® for Arm®, Version 8.20.2
- Optimization: high size
- MCU: STM32F072RB
- Expansion board: P-NUCLEO-USB001 shield

Table 6. USB-PD - Provider memory footprint (in Bytes)

Project	Provider (RTOS-based)		Description
	Flash memory (bytes)	RAM Memory (bytes)	
User application	3154	212	Memory needed for the User application code
USBPD core stack	11006	44	Memory needed for the USB-PD core stack library
USBPD device driver	12172	1593	Memory needed for the USB-PD device driver
HAL/LL, BSP, Drivers	2408	8	HAL/ LL BSP and Nucleo drivers services
FreeRTOS	3828	4528	FreeRTOS allocation and services
Linker created	918	2828	Memory space requirements added at link stage
Total	33486	9213	Total memory.

Table 7. USB-PD - Consumer memory footprint (in Bytes)

Project	Provider (RTOS-based)		Description
	Flash memory (bytes)	RAM Memory (bytes)	
User application	3540	212	Memory needed for the User application code
USBPD core stack	10510	44	Memory needed for the USB-PD core stack library
USBPD device driver	12168	1593	Memory needed for the USB-PD device driver
HAL/LL, BSP, Drivers	2408	8	HAL/ LL BSP and Nucleo drivers services
FreeRTOS	3828	4528	FreeRTOS allocation and services

Table 7. USB-PD - Consumer memory footprint (in Bytes) (continued)

Project	Provider (RTOS-based)		Description
	Flash memory (bytes)	RAM Memory (bytes)	
Linker created	1470	2828	Memory space requirements added at link stage
Total	33924	9213	Total memory.

Table 8. USB-PD - Dual role port memory footprint (in Bytes)

Project	Consumer DRP (RTOS-based)		Description
	Flash memory (bytes)	RAM Memory (bytes)	
User application	4024	220	Memory needed for the User application code
USBPD core stack	16234	44	Memory needed for the USB-PD core stack library
USBPD device driver	12172	1593	Memory needed for the USB-PD device driver
HAL/LL, BSP, Drivers	2534	12	HAL/ LL BSP and Nucleo drivers services
FreeRTOS	3828	4528	FreeRTOS allocation and services
Linker created	1470	2828	Memory space requirements added at link stage
Total	40262	9225	Total memory.

Note: When enabling “--basic_heap” option in IAR™ Linker Extra Options, it is possible to spare some RAM space allocated to heap and dynamic allocations management.

Table 9. USB-PD - Provider (Low Footprint version) memory footprint

Project	Provider Low Footprint (Non RTOS)		Description
	Flash Memory (bytes)	RAM Memory (bytes)	
User application	1970	116	Memory needed for User application code
USBPD Core stack	7070	32	Memory needed for the USB-PD Core Stack library
USBPD Device driver	12036	1521	Memory needed for the USB-PD Device Driver

Table 9. USB-PD - Provider (Low Footprint version) memory footprint (continued)

Project	Provider Low Footprint (Non RTOS)		Description
	Flash Memory (bytes)	RAM Memory (bytes)	
HAL/LL, BSP, Drivers	2432	8	HAL/ LL BSP and Nucleo drivers services
FreeRTOS	-	-	FreeRTOS allocation and services
Linker Created	758	2828	Memory space requirements added at link stage
Total	24266	4505	Total memory

8 Frequently asked questions (FAQs)

How can I get the STM32 USB-PD library?

The library is provided for free download in a binary format, from www.st.com.

Does the library support USB data communication?

The library is responsible only of PD communication, however, the first port can carry USB data. It is possible to add the STM32 USB library to allow USB communication through the first port.

I want to use only the USB-C feature (cable detachment attachment and cable orientation). Is this possible?

Yes, this is possible since the CAD (Cable attachment and detachment) module and the PD communication are driven by two separated processes. You can call only the CAD process to ensure cable detection.

Does the X-CUBE-USB-PD Expansion Package work on platforms different from STM32F0?

The core stack is device independent, but only STM32F0 platform is supported on the device part in this delivery. Package will be enhanced to support other STM32 microcontrollers in the future.

Using a provider, how can I power a consumer port partner that needs voltage values higher than 5 V?

For providing higher voltage values (up to 20 V), an external power board must be connected to P-NUCLEO-USB001 shield through connector CN4 (refer to UM2050 for more details).

9 Revision history

Table 10. Document revision history

Date	Revision	Changes
08-Jun-2016	1	Initial release.
23-Jan-2017	2	<p>Updated Section 1.2: References, Section 3.1: Overview, Section 3.2: Features, Section 3.3: Library architecture, Section 3.4: Hardware related components (for P-NUCLEO-USB001), Section 4: USB-PD library programming guidelines, Section 4.2: Library initialization, Section 4.3: USB-PD core stack library functions, Section 4.4: USB-PD core stack, Section 6.1: Hardware description, Section 6.2: USB-PD provider, Section 6.5: USB-PD consumer, Section 6.8: USB-PD consumer DRP and Section 8: Frequently asked questions (FAQs).</p> <p>Added Section 6.3: USB-PD provider (with CLI support), Section 6.4: USB-PD provider (with VDM support), Section 6.6: USB-PD consumer (with CLI support), Section 6.7: USB-PD consumer (with VDM support) and Section 6.9: USB-PD Dual Port.</p> <p>Updated Table 1: List of acronyms, Table 2: Use of different IPs, Table 3: GPIOs used by Port0, Table 4: GPIOs used by Port1, Table 6: DPM files, Table 7: USB-PD user functions, Table 8: USB-C PD callbacks, Table 9: USB-PD - Provider memory footprint (in Bytes), Table 10: USB-PD - Consumer memory footprint and Table 11: USB-PD - Dual role port memory footprint.</p> <p>Updated Figure 6: Project files.</p>
09-May-2018	3	<p>Updated Introduction, Section 1.2: References, Section 2.1: Architecture overview, Section 3.1: Overview, Section 3.2: Features, Section 3.3: Library architecture, Section 3.4: Hardware related components (for P-NUCLEO-USB001), Section 4: USB-PD library programming guidelines, Section 4.2: Library initialization, Section 4.4: USB-PD core stack, Section 6.5.1: Example setup, Section 6.9.1: Example setup, Section 7: Memory footprint and Section 8: Frequently asked questions (FAQs).</p> <p>Added Section 4.5: USB-PD device components.</p> <p>Updated Figure 1: USB power delivery architecture, Figure 6: Project files and Figure 7: USB-PD stack architecture (for P-NUCLEO-USB001).</p> <p>Updated Table 6: USB-PD - Provider memory footprint (in Bytes), Table 7: USB-PD - Consumer memory footprint (in Bytes) and Table 8: USB-PD - Dual role port memory footprint (in Bytes).</p> <p>Removed former Table 6: DPM files, Table 7: USB-PD user functions and Table 8: USB-C PD callbacks.</p>
30-May-2018	4	<p>Updated Introduction</p> <p>Added Chapter 2.2: Specific case of TCPM/TCPC architecture, Chapter 4.1: Description of available configurations, Chapter 5: Atomic message sequencing and Chapter 6.10: USB-PD DRP (TCPM/TCPC architecture)</p>

IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST's terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers' products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2018 STMicroelectronics – All rights reserved