UM2063
User manual

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

Introduction

This document describes the STM32 USB-PD (Power Delivery) software expansion 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/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.

STM32 USB-PD package (X-CUBE-USB-PD) consists of a library (in a binary format) and application examples for STM32F0 devices acting as USB Power Delivery controllers. It will be extended to other STM32 Series.

This package includes examples covering most of applicative use cases, allowing the user to develop applications based on USB-PD (Provider, Consumer, and Dual-Role Port).

The library and the associated examples can be used with the STM32F0 devices and a dedicated P-NUCLEO-USB001 shield containing Analog Front-Ends and USB Type-C™ connectors.

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
- How to use the library for setting an USB-PD application.

Contents

List of tables

List of figures

1 Overview

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
UFP	Upstream Facing Port
USB	Universal Serial Bus
VDM	Vendor Defined Messages

1.2 References

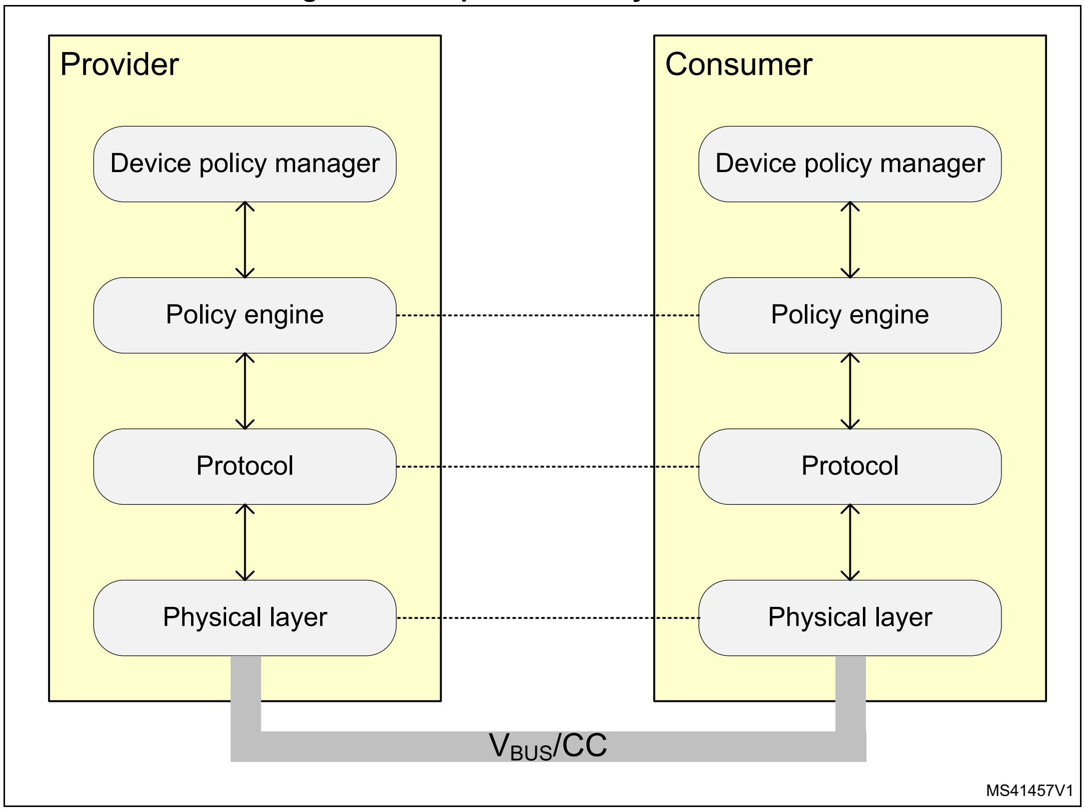
- Universal Serial Bus Power Delivery Specification, Revision 2.0, March 25, 2016
- Universal Serial Bus Type-C Cable and Connector Specification 1.2, March 25, 2016

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 PD-capable device is assumed to be made up of at least one port, which can be:

- Upstream Facing Port (UFPs)
  - Sink power (a consumer).
  - Optionally source power (a consumer/provider).
  - Optionally communicate via USB
  - Communicate using SOP Packets.
- Downstream Facing Port (DFPs):
  - Source power (a provider).
  - Optionally Sink power (a provider/consumer).
  - Optionally communicate via USB.
  - Communicate using SOP Packets
  - Optionally communicates using SOP 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 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 Manger 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.

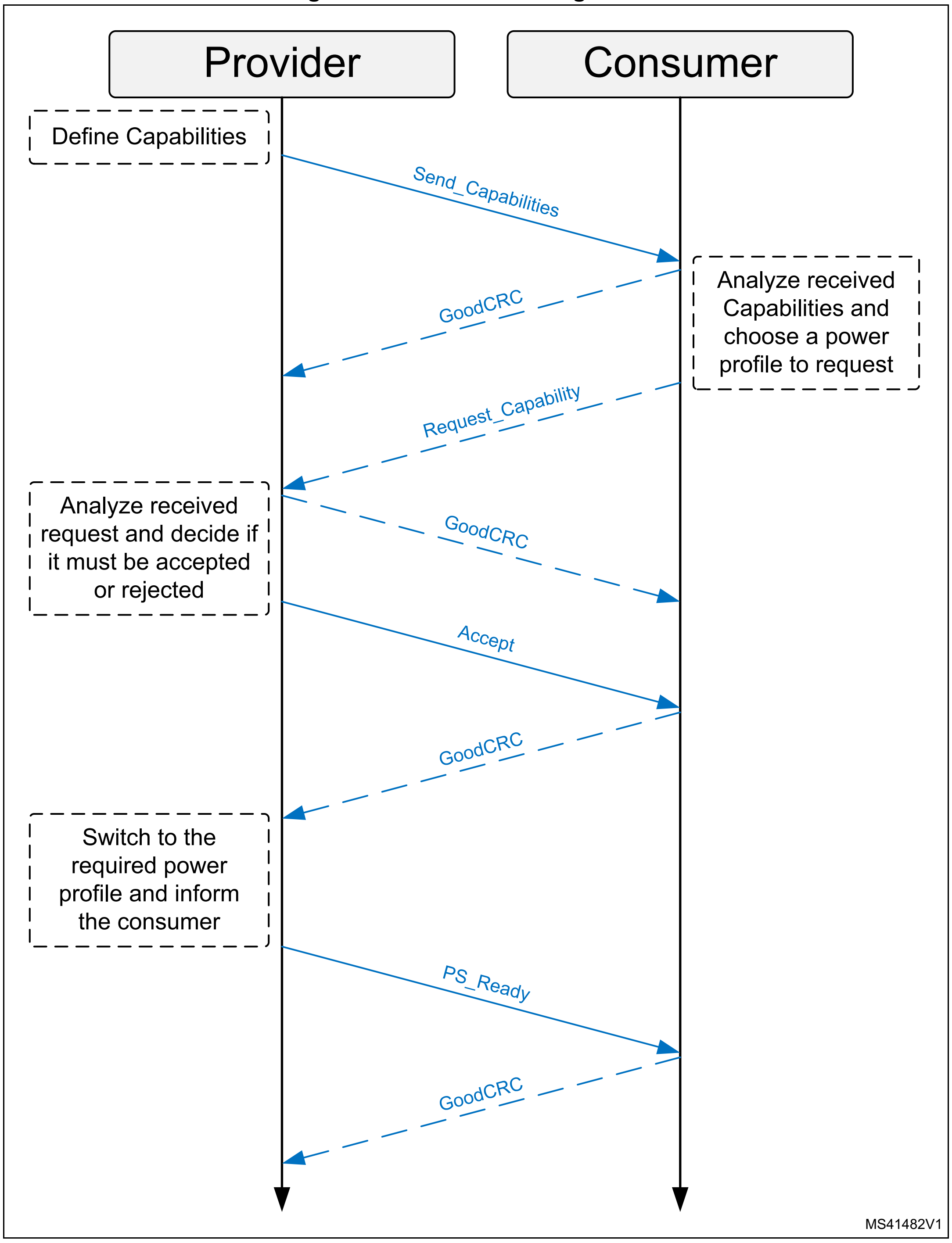
2.3 Message flow

Each message sent from a device should be replied by a GoodCRC message. This permits to the receiver to acknowledge the sender that its message was correctly received, and would be treated by upper layers.

The wrong messages received should be ignored, in the 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 proceeds according to the sequence illustrated in *Figure 2*.

Figure 2. USB-PD - Messages flow

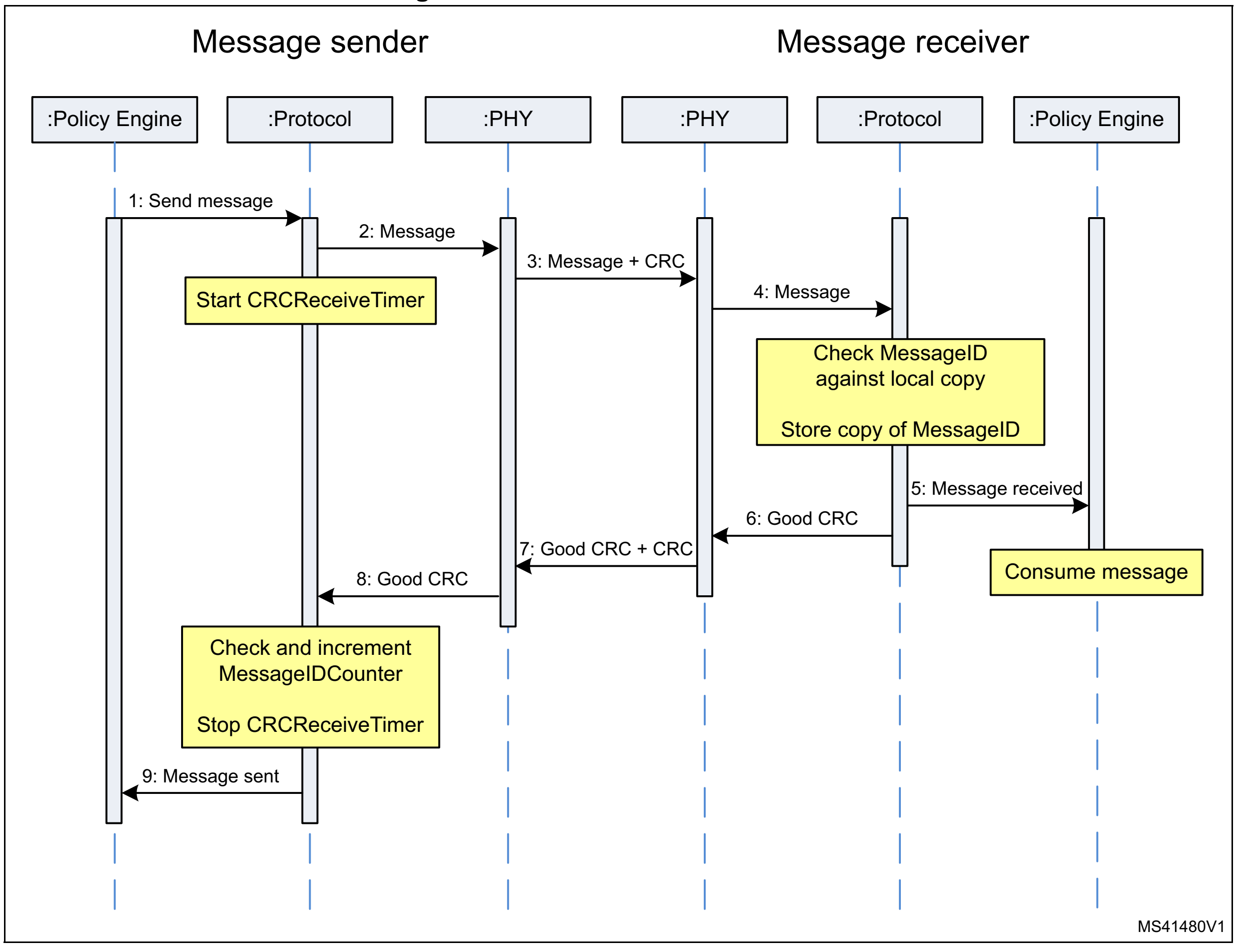

Provider
Consumer
Define Capabilities
Send_Capabilities
GoodCRC
Analyze received Capabilities and choose a power profile to request
Request_Capability
Analyze received request and decide if it must be accepted or rejected
GoodCRC
Accept
GoodCRC
Switch to the required power profile and inform the consumer
PS_Ready
GoodCRC
MS41482V1

2.4 Data flow

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

Each received message should be replied to with a GoodCRC message to inform the sender of the correct reception (see *Figure 3*). BIST messages are an exception, as they should not be passed to the PE layer once received, and should only be replied to with a GoodCRC.

Figure 3. USB-PD stack architecture



3 USB-PD library description

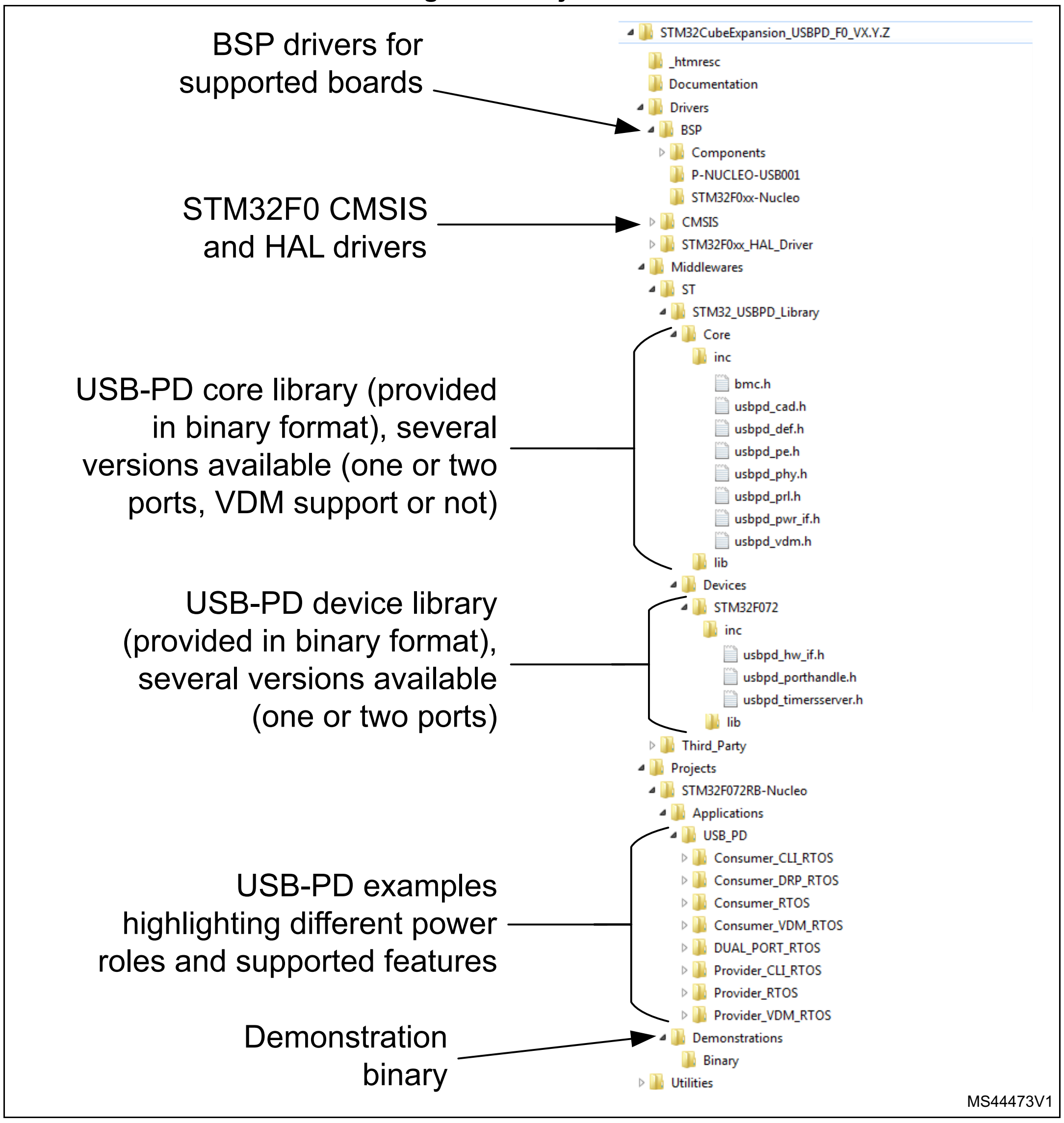
3.1 Overview

STMicroelectronics offers an USB-C Power Delivery Library supporting STM32F0xRB microcontrollers. 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, V1.2 (March 25, 2016) and Universal Serial Bus type-C Cable and Connector Specification, Revision 1.2 (March 25, 2016). It has passed successfully the official certification.

Figure 4. Project files



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.

3.2 Features

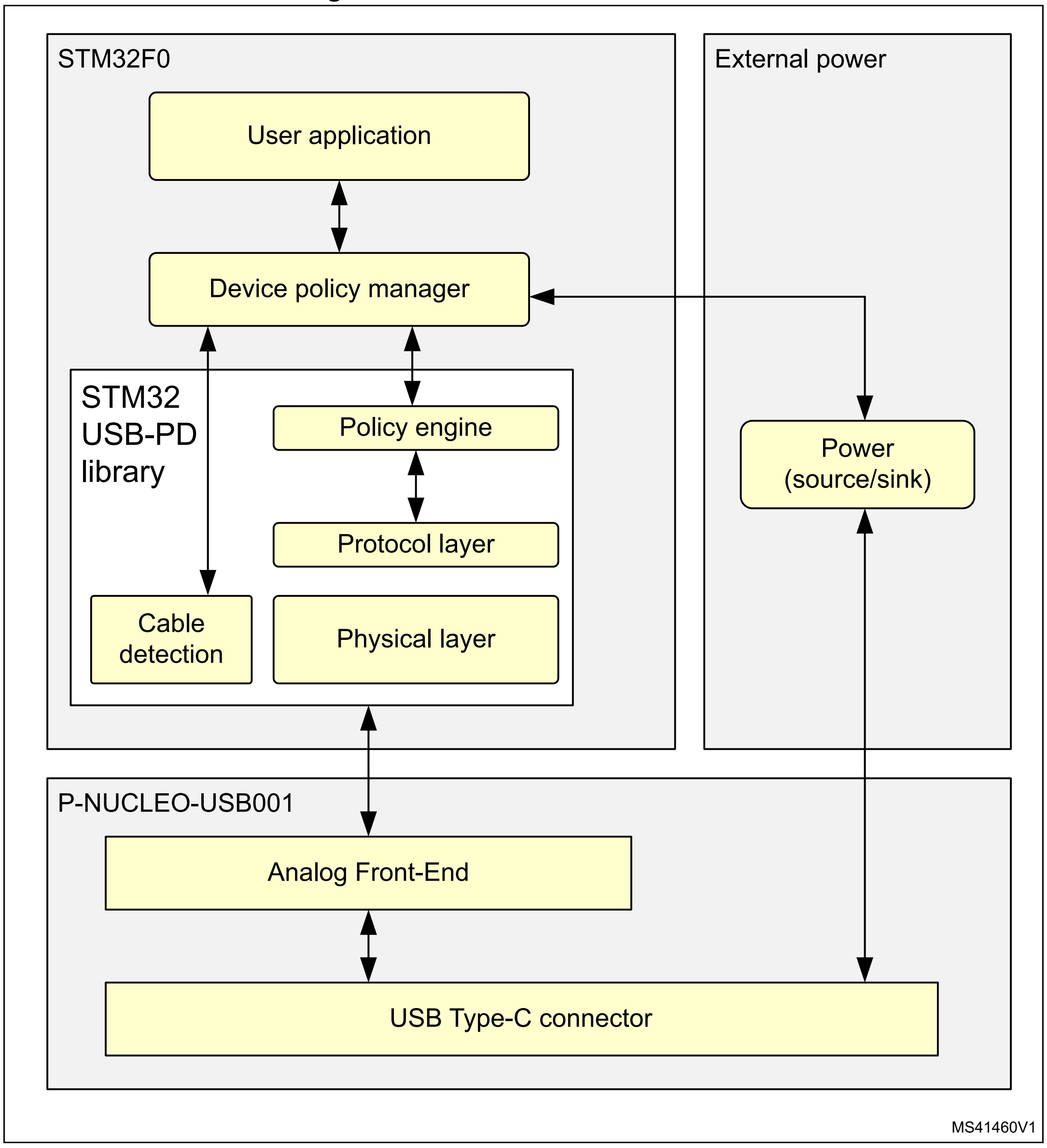
The main characteristics of the USB-PD software expansion package are listed below:

- compliant with USB Type-C 1.2 and USB Power Delivery 2.0 Standards
- provider, consumer and Dual Role mode (DRP) support
- Dual Port support
- PD communication supported for the two sides of the cable
- cable detection (detection of an USB-C cable plug/unplug)
- 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

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 shield), containing analog front ends and USB-C connectors is used with the Nucleo board to ensure the communication.

Figure 5. USB-PD stack architecture



The USB-PD libraries cover two main parts:

- USB-PD core stack (generic components, independent from HW):
  - Protocol Layer
  - Policy Engine Layer
  - Device Policy Manager
  - Application Layer
- STM32F072 components (specific):
  - Physical Layer
  - Cable Detection Module (CAD)

3.4 Hardware related components

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 shouldn't 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 and Device component) in binary format, with corresponding headers files ".h" describing library APIs.

4.1 Library initialization

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

1. Include the STM32 USB-C PD libraries in your project. Selection libraries type according to your toolchain, and according to features required to be supported (1 or two ports, VDM supported or not). One library should be selected for the USB-PD Core stack component, in the directory *\Middlewares\ST\STM32_USBPD_Library\Core\lib\*.
   A second library should be selected for the USB-PD component dependent from the used device, in the directory *\Middlewares\ST\STM32_USBPD_Library\Devices\*.
   For STM32F072 specific case, please select device library from *.\Middlewares\ST\STM32_USBPD_Library\Devices\STM32F072\lib\*.
2. Include the DPM files in the application. Some examples of DPM files are available in the provided project examples in Firmware package, as indicated in *Table 6*.
3. Configure the MCU clock, a template is implemented and can be used in the *systemClock_Config()* function in the main.c file of each project example.
4. Call the *USBPD_HW_IF_GlobalHwInit()* function to Initialize the Hardware Interface layer.
5. Call the *USBPD_DPM_Init()* function to initialize the Device Policy Manager layer and all the communication layers (PE, VDM, PRL, PHY).
6. Launch the RTOS Kernel (task scheduler).

Table 6. DPM files

Application	File location in *.\Projects\STM32F072RB-Nucleo\Applications\USB_PD*
Provider	*Provider_RTOS\Inc\usbpd_dpm.h*
	Provider_RTOS\Src\usbpd_dpm.c
Provider with Command Line Interface	*Provider_CLI_RTOS\Inc\usbpd_dpm.h*
	Provider_CLI_RTOS\Src\usbpd_dpm.c
Provider with VDM support	*Provider_VDM_RTOS\Inc\usbpd_dpm.h*
	Provider_VDM_RTOS\Src\usbpd_dpm.c
Consumer	*Consumer_RTOS\Inc\usbpd_dpm.h*
	Consumer_RTOS\Src\usbpd_dpm.c
Consumer with Command Line Interface	*Consumer_CLI_RTOS\Inc\usbpd_dpm.h*
	Consumer_CLI_RTOS\Src\usbpd_dpm.c
Consumer with VDM support	*Provider_VDM_RTOS\Inc\usbpd_dpm.h*
	Provider_VDM_RTOS\Src\usbpd_dpm.c
Dual port with Command Line Interface	*DUAL_PORT_RTOS\Inc\usbpd_dpm.h*
	DUAL_PORT_RTOS\Src\usbpd_dpm.c

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

4.2 USB-PD 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.

Table 7. USB-PD user functions

Function	Description
`void USBPD_CAD_PortEnable (uint8_t PortNum, USBPD_CAD_activation State);`	Enables or disables the CAD port.
`USBPD_CAD_StatusTypeDef USBPD_CAD_Init(uint8_t PortNum, USBPD_CAD_Callbacks CallbackFunctions);`	Initializes the CAD module for a specified port.
`void USBPD_CAD_AssertRd(uint8_t PortNum);`	Sets the port as Sink.
`void USBPD_CAD_AssertRp(uint8_t PortNum);`	Sets the port as Source.
`USBPD_StatusTypeDef USBPD_PE_Init(uint8_t PortNum, USBPD_PortPowerRole_TypeDef role, USBPD_PE_Callbacks pecallbacks);`	Initializes the Policy Engine layer for a port with a specified role.
`USBPD_StatusTypeDef USBPD_PE_DeInit(uint8_t PortNum);`	De-initialize the Policy Engine layer.
`void USBPD_PE_SRCProcess(uint8_t PortNum);`	Policy Engine processing function for Source application.
`void USBPD_PE_SNKProcess(uint8_t PortNum);`	Policy Engine processing function for Sink application.
`void USBPD_PE_DRPProcess(uint8_t PortNum);`	Policy Engine processing function for Dual Role application.
`USBPD_StatusTypeDef USBPD_PE_RequestNewPowerProfile(uint8_t PortNum, uint8_t PDOIndex);`	Called by DPM to evaluate received Capabilities Message from Source port and prepare Request message.
`USBPD_StatusTypeDef USBPD_PE_RequestPowerRoleSwap(uint8_t PortNum);`	Called by DPM to request PE to perform a Power Role Swap.
`USBPD_StatusTypeDef USBPD_PE_GetReceivedPowerProfile(uint8_t PortNum, uint32_t *pPDO, uint32_t *pNbPDO);`	Called by DPM to retrieve received power profiles from source port partner.

Table 7. USB-PD user functions (continued)

Function	Description
`USBPD_StatusTypeDef USBPD_PE_IsCableConnected(uint8_t PortNum, uint8_t IsConnected);`	Called by DPM to set the cable status connected or disconnected.
`USBPD_StatusTypeDef USBPD_VDM_Init(uint8_t PortNum, USBPD_VDM_Callbacks cbs);`	Initializes the VDM layer.
`USBPD_StatusTypeDef USBPD_PE_SVDM_RequestIdentity(uint8_t PortNum, USBPD_SOPType_TypeDef SOPType);`	Called by DPM to request PE to perform a VDM identity request.
`USBPD_StatusTypeDef USBPD_PE_SVDM_RequestModeExit(uint8_t PortNum, USBPD_SOPType_TypeDef SOPType);`	Called by DPM to request PE to perform a VDM mode exit.
`USBPD_StatusTypeDef USBPD_PE_SVDM_RequestAttention(uint8_t PortNum);`	Called by DPM to request PE to perform a VDM Attention.

4.3 USB-C PD library callbacks

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.

Table 8. USB-C PD callbacks

Function	Description
`void (*USBPD_CAD_CallbackEvent)(uint8_t PortNum, USBPD_CAD_STATE State, CCxPin_TypeDef Cc)`	Called by the CAD to inform of a cable attach/detach, and the CC line.
`void (*USBPD_PE_RequestSetupNewPower)(uint8_t PortNum);`	Requests the DPM to setup the new power level.
`uint32_t (*USBPD_PE_HardReset)(uint8_t PortNum);`	Requests the DPM to perform an Hard Reset.
`USBPD_StatusTypeDef (*USBPD_PE_EvaluatPRSwap)(uint8_t PortNum);`	Gets evaluation of PR swap request from DPM.
`void (*USBPD_PE_TurnOffPower)(uint8_t PortNum, USBPD_PortPowerRole_TypeDef Role);`	Requests the DPM to turn Off power supply.
`void (*USBPD_PE_TurnOnPower)(uint8_t PortNum, USBPD_PortPowerRole_TypeDef Role);`	Requests the DPM to turn on the power supply.
`void (*USBPD_PE_AssertRd)(uint8_t PortNum);`	Requests the DPM to assert Rd.

Table 8. USB-C PD callbacks (continued)

Function	Description
`void (*USBPD_PE_AssertRp)(uint8_t PortNum);`	Requests the DPM to assert Rp.
`void (*USBPD_PE_ExplicitContractDone)(uint8_t PortNum);`	Informs DPM that an Explicit contract has been established.
`void (*USBPD_PE_GetDataInfo)(uint8_t PortNum, USBPD_CORE_DataInfoType_TypeDef DataId, uint32_t *Ptr, uint32_t *Size);`	Allows PE to retrieve information from DPM/PWR_IF.
`void (*USBPD_PE_SetDataInfo)(uint8_t PortNum, USBPD_CORE_DataInfoType_TypeDef DataId, uint32_t *Ptr, uint32_t Size);`	Allows PE to update information in DPM/PWR_IF.
`USBPD_StatusTypeDef (*USBPD_PE_EvaluateRequest)(uint8_t PortNum);`	Allows PE to request evaluation in DPM of a Request received from Sink.
`USBPD_StatusTypeDef (*USBPD_PE_EvaluateCapabilities)(uint8_t PortNum);`	Allows PE to request evaluation in DPM of Source Capabilities received from Source.
`USBPD_StatusTypeDef (*USBPD_VDM_DiscoverIdentity)(uint8_t Port, USBPD_DiscoveryIdentity_TypeDef **pIdentity);`	VDM Discovery identity callback (answers to Discover Identity message).
`USBPD_StatusTypeDef (*USBPD_VDM_DiscoverSVIDs)(uint8_t Port, USBPD_SVIDInfo_TypeDef **p_SVID_Info);`	VDM Discover SVID callback (retrieves SVID supported by device for answer to Discovery mode).
`USBPD_StatusTypeDef (*USBPD_VDM_DiscoverModes)(uint8_t Port, uint16_t SVID, USBPD_ModeInfo_TypeDef **p_ModeInfo);`	VDM Discover Mode callback (reports all the modes supported by SVID).
`USBPD_StatusTypeDef (*USBPD_VDM_ModeEnter)(uint8_t Port, uint16_t SVID, uint32_t ModeIndex);`	Reports to DPM the result of Mode Entry.
`USBPD_StatusTypeDef (*USBPD_VDM_ModeExit)(uint8_t Port, uint16_t SVID, uint32_t ModeIndex);`	Reports to DPM the result of Mode Exit.
`USBPD_StatusTypeDef (*USBPD_VDM_InformIdentity)(uint8_t Port, USBPD_SOPType_TypeDef SOPType, USBPD_DiscoveryIdentity_TypeDef *Identity);`	Informs identity callback (Identity information received in Discovery identity answer).
`USBPD_StatusTypeDef (*USBPD_VDM_InformSVID)(uint8_t Port, USBPD_SOPType_TypeDef SOPType, uint16_t SVID);`	Informs SVID callback.

Table 8. USB-C PD callbacks (continued)

Function	Description
`USBPD_StatusTypeDef (*USBPD_VDM_InformMode)(uint8_t Port, USBPD_SOPType_TypeDef SOPType, USBPD_ModeInfo_TypeDef *ModesInfo);`	Informs Mode callback.
`USBPD_StatusTypeDef (*USBPD_VDM_InformModeEnter)(uint8_t Port, uint16_t SVID, uint32_t ModeIndex);`	Informs Mode enter callback.
`USBPD_StatusTypeDef (*USBPD_VDM_InformModeExit)(uint8_t Port, uint16_t SVID, uint32_t ModeIndex);`	Informs Mode exit callback.
`uint16_t (*USBPD_VDM_GetSVID)(uint8_t Port);`	Gets SVID for Discovery Mode callback.
`uint32_t (*USBPD_VDM_GetModeIndex)(uint8_t Port, uint16_t SVID);`	Gets index of VDO for SVID (Enter mode) callback.
`uint16_t (*USBPD_VDM_GetSVID)(uint8_t Port);`	Gets SVID for Discovery Mode callback.
`USBPD_StatusTypeDef (*USBPD_VDM_Attention)(uint8_t Port, USBPD_AttentionInfo_TypeDef **pAttentionInfo);`	Reports to VDM that an attention command is processed.
`uint32_t (*USBPD_VDM_HardReset)(uint8_t Port);`	Requests DPM to perform a VDM Hard Reset.

5 Examples description

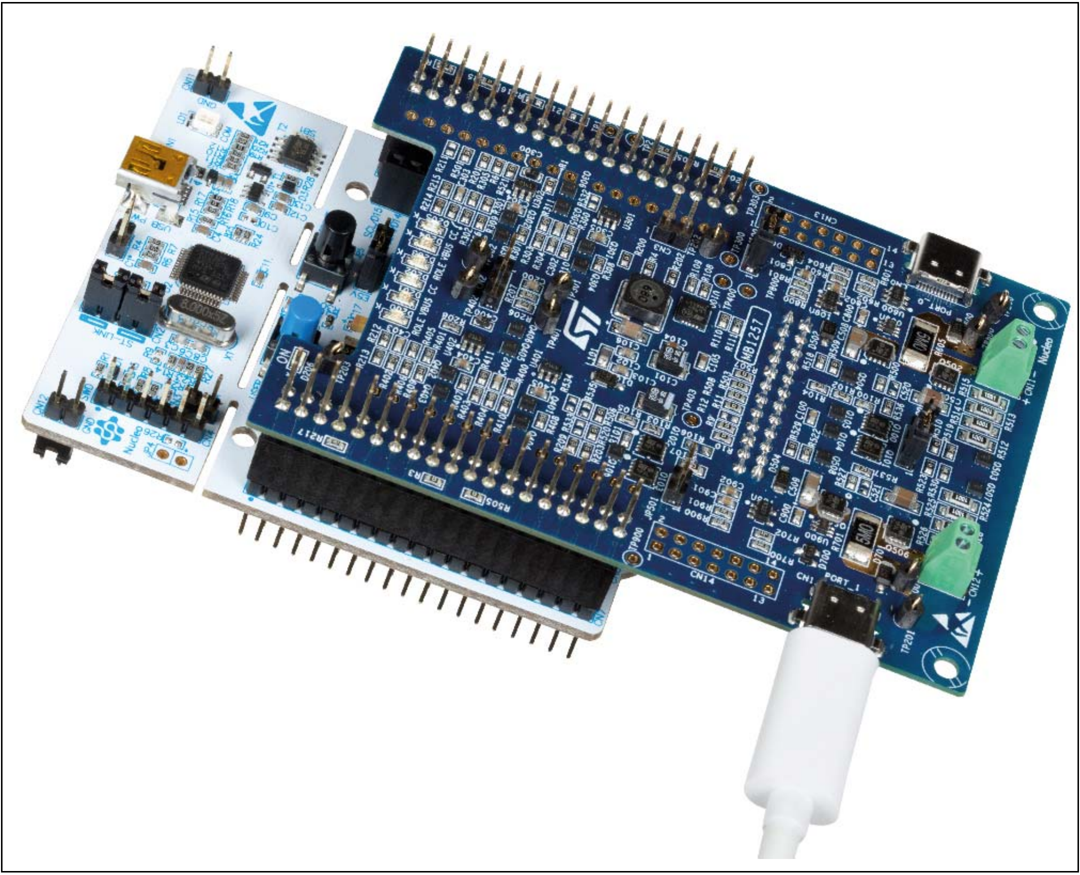
5.1 Hardware description

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

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

Figure 6. STM32FO72RB Nucleo with P-Nucleo-USB001 shield



For all the provided examples, the user has to

1. open its preferred toolchain
2. rebuild all files and load the image into 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).

5.2 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.

5.2.1 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.

5.2.2 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) will blink, indicating Port role (blinking only once indicates provider role).
- User should plug the USB-C cable on the dedicated connector. When attached, CC LEDs (D205) will blink once if connected to CC1, twice if connected to CC2.
- Blue LED (D203) will be blinking one time, 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) will be blinking.
- If the requested power can be met, the STM32 MCU will send 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.

5.3 USB-PD provider (with CLI support)

This application is similar to USB-PD provider application, so the description provided in *Section 5.2* 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

5.4 USB-PD provider (with VDM support)

This application is similar to USB-PD provider application, so the description provided in *Section 5.2* 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 will 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 underProjects\STM32F072RB-Nucleo\Applications\USB_PD\Provider_VDM_RTOS

5.5 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.

5.5.1 Example setup

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

5.5.2 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) will be blinking indicating Port role (blink twice indicating consumer Role).
- User should plug the USB-C cable on the dedicated connector.
- When attached, CC LEDs (D205) will blink once if connected to CC1, twice if connected to CC2.
- Blue LED (D203) will be blinking 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) will be blinking.
- The STM32 will send 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) will be ON to indicate that the Power Contract has been established.

5.6 USB-PD consumer (with CLI support)

This application is similar to USB-PD consumer application, so the description provided in *Section 5.5* 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

5.7 USB-PD consumer (with VDM support)

This application is similar to USB-PD consumer application, so the description provided in *Section 5.5* 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 should 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

5.8 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 X-CUBE-USB-PD expansion package.

5.8.1 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 will take, 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 will allow 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.

5.8.2 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 will act 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 will act 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) will be blinking indicating Port role (blink three times indicates DRP).
- User should 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 will send 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 will send 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) will be blinking.
- When attached, CC LEDs (D205) will blink once if connected to CC1, twice if connected to CC2.
- Role (Blue) LED (D203) will be blinking twice each time if the device behaves as a consumer, or will be blinking once each time, if the device behaves as a provider.

5.9 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.

5.9.1 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 will take, 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 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, 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 will allow 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.

5.9.2 Application description

This application provides an example for managing both Port0 and Port1 simultaneously. Both ports are DRP, and behave independently. User should 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) will be blinking indicating the port role (blink three times indicates DRP Role).
- User should 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 will then send 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 will send 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) will be blinking.
- When attached, CC LEDs (LED D205 for Port0 and LED D202 for Port1) will blink once if connected to CC1, twice if connected to CC2.
- Role (Blue) LED (LED D203 for Port0 and LED D200 for Port1) will be blinking twice each time if the device the behaves as a consumer, or will be blinking once each time if the device behaves as a provider.

This application also embeds a Command Line Interface (CLI) feature, which allows 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 Memory footprint

The values in tables 9 to 11 are calculated according to the following configuration:

- Compiler: IAR Embedded Workbench® for ARM®, Version 7.50.1
- Optimization: high speed
- MCU: STM32F072RB
- Expansion board: P-NUCLEO-USB001 shield

Table 9. USB-PD - Provider memory footprint

Project	Provider (RTOS-based)		Description
	Flash memory (bytes)	RAM (bytes)	
USB-PD library	15014	2329	Memory needed for the USB-PD library.
Application layer	18489	3863	User application, HAL drivers, utilities, FreeRTOS, LEDs management, and others.
Total	33053	6192	Total memory.

Table 10. USB-PD - Consumer memory footprint

Project	Consumer (RTOS-based)		Description
	Flash memory (bytes)	RAM (bytes)	
USB-PD library	14815	2389	Memory needed for the USB-PD library.
Application layer	18529	3811	User application, HAL drivers, utilities, FreeRTOS, LEDs management, and others.
Total	33344	6200	Total memory.

Table 11. USB-PD - Dual role port memory footprint

Project	DRP (RTOS-based)		Description
	Flash memory (bytes)	RAM (bytes)	
USB-PD library	16028	2389	Memory needed for the USB-PD library.
Application layer	18768	3811	User application, HAL drivers, utilities, FreeRTOS, LEDs management, and others.
Total	34796	6200	Total memory.

Note: DRP values are valid for both provider DRP and consumer DRP.

7 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 Attachement and Detachement) 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 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).

8 Revision history

Table 12. Document revision history

Date	Revision	Changes
08-Jun-2016	1	Initial release.
23-Jan-2017	2	Updated *Section 1.2: References*, *Section 3.1: Overview*, *Section 3.2: Features*, *Section 3.3: Library architecture*, *Section 3.4: Hardware related components*, *Section 4: USB-PD library programming guidelines*, *Section 4.1: Library initialization*, *Section 4.2: USB-PD library functions*, *Section 4.3: USB-C PD library callbacks*, *Section 5.1: Hardware description*, *Section 5.2: USB-PD provider*, *Section 5.5: USB-PD consumer*, *Section 5.8: USB-PD consumer DRP* and *Section 7: Frequently asked questions (FAQs)*. Added *Section 5.3: USB-PD provider (with CLI support)*, *Section 5.4: USB-PD provider (with VDM support)*, *Section 5.6: USB-PD consumer (with CLI support)*, *Section 5.7: USB-PD consumer (with VDM support)* and *Section 5.9: USB-PD Dual Port*. 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*, *Table 10: USB-PD - Consumer memory footprint* and *Table 11: USB-PD - Dual role port memory footprint*. Updated *Figure 4: Project files*.