Introduction

This application note is a guideline to build a very simple USB power delivery sink example, starting from STM32CubeMX. This document applies to STM32 MCUs embedding the UCPD peripheral.

The principal hardware used in the different screenshots is based on the STM32G0 Series microcontroller with its associated firmware included in the STM32CubeG0 MCU Package, but some notes are added to this document so that the STM32G4 Series microcontroller with its associated firmware in STM32CubeG4 can also be used. The associated firmware of the STM32L5 Series microcontroller is in STM32CubeL5.

To ease connections with USB Type-C™ signals, the STM32G0 and STM32G4 Nucleo-64 boards are equipped with the X-NUCLEO-USBPDM1 shield. The STM32L5 Nucleo-144, Discovery, and Evaluation boards are USB-PD ready as they embed the TCPP01-M12 chip.

Figure 1. STM32G0 Nucleo-64 board equipped with X-NUCLEO-USBPDM1 shield
1 General information

The simple USB-PD sink application runs on STM32G0 Series, STM32G4 Series, and STM32L5 Series 32-bit microcontrollers based on the Arm® Cortex®-M processor.

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

Table 1. Acronym definitions

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMS</td>
<td>Atomic message sequence</td>
</tr>
<tr>
<td>APDO</td>
<td>Augmented power delivery object. Specific PDO to handle PPS</td>
</tr>
<tr>
<td>BSP</td>
<td>Board support package</td>
</tr>
<tr>
<td>CAD</td>
<td>Cable detection module responsible for attaching or detaching detection</td>
</tr>
<tr>
<td>CC</td>
<td>Communication channel</td>
</tr>
<tr>
<td>DPM</td>
<td>Device policy manager. In the power delivery context, this part corresponds to the user application.</td>
</tr>
<tr>
<td>DRP</td>
<td>Dual role power. The ability for a product to either source or sink power.</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical user interface. It applies here to STM32CubeMonitor-UCPD (STM32CubeMonUCPD).</td>
</tr>
<tr>
<td>HAL</td>
<td>Hardware abstraction layer</td>
</tr>
<tr>
<td>HW</td>
<td>Hardware</td>
</tr>
<tr>
<td>LL</td>
<td>Low layer</td>
</tr>
<tr>
<td>OVP</td>
<td>Over-voltage protection</td>
</tr>
<tr>
<td>PDO</td>
<td>Power delivery object</td>
</tr>
<tr>
<td>PPS</td>
<td>Programmable power supply</td>
</tr>
<tr>
<td>PE</td>
<td>Policy engine</td>
</tr>
<tr>
<td>SNK</td>
<td>Power sink. Ability to request for power</td>
</tr>
<tr>
<td>SRC</td>
<td>Power source. Ability to provide power</td>
</tr>
<tr>
<td>UCPD</td>
<td>USB Type-C™ power delivery. A new peripheral in some STM32 series, like STM32G0 Series, STM32G4 Series or STM32L5 Series, which manage power delivery protocol communication with two lines.</td>
</tr>
</tbody>
</table>
3 Reference documents

STMicroelectronics ecosystem material:
- STM32CubeMX: STM32Cube initialization code generator
- User manual Managing USB power delivery systems with STM32 microcontrollers (UM2552)
- User manual STM32CubeMonitor-UCPD software tool for USB Type-C™ Power Delivery port management (UM2468)
- Datasheet TCPP01-M12 USB type-C port protection (DS12900)
- Application note USB Type-C protection and filtering (AN4871)
- Databrief STM32CubeMonitor-UCPD software tool for USB Type-C™ Power Delivery port management (DB3747)
- Technical article Overview of USB Type-C and Power Delivery technologies (TA0357)
- Application note USB Type-C™ Power Delivery using STM32xx Series MCUs and STM32xxx Series MPUs (AN5225)
- Youtube video STM32G0: Create a USB Power delivery sink application in less than 10 minutes
- X-NUCLEO-USBPDM1 user manual (UM2668)
- USBPD overview wiki

USB specification documents:
- USB2.0 Universal Serial Bus Revision 2.0 Specification
- USB3.1 Universal Serial Bus Revision 3.2 Specification
- USB BC Battery Charging Specification Revision 1.2
- Universal Serial Bus Type-C Cable and Connector Specification 2.0, August 2019.
- Universal Serial Bus Power Delivery Specification, Revision 3.0, Version 2.0, August 28, 2019
4 Getting started

The very first goal is to configure the UCPD peripheral with USB-PD stack and to check that a first contract is reached, which means the sink finds a matching power source. So any wall charger or any power delivery certified source is also needed.

To reach this goal, the following steps are necessary:

1. Set up the UCPD peripheral to expose Rd resistors on the CC lines and detect the Rp from the source, using STM32CubeMX.
2. Read the $V_{BUS}$ from the attached source. The initialization part is done by STM32CubeMX, but the measurement start must be added manually in the application.
3. Finally, send a power delivery request message to the source and reach an explicit contract. This can only be done manually, by editing the application files, after they are generated by STM32CubeMX.

Optional steps are described in this document, to help the user to debug:

4. Addition of a trace utility that uses the ST-LINK Virtual COM port to get some debug information from the board
5. Addition of an embedded tool to communicate with STM32CubeMonitor-UCPD, a Java application (GUI) to help to build the application
5 STM32CubeMX step-by-step sequence

5.1 Mandatory parts

The below steps can be done because of the presence of the UCPD peripheral in some STM32 devices, such as STM32G071xx and STM32G431xx. This peripheral manages power delivery communication over the CC lines. The X-NUCLEO-USBPDM1 shield, embedding the TCPP01-M12 single-chip solution, adds the microcontroller protection on the CC lines, such as over-voltage and over-temperature, and it also gives access to a USB Type-C™ receptacle.

Note that TCPP01-M12 is already embedded in the STM32L5 Nucleo-144, Discovery, and Evaluation boards. Before starting, check first that both latest versions of STM32CubeMX and STM32CubeG0 or STM32CubeG4 MCU Packages are used.

5.1.1 Start STM32CubeMX and select the MCU

Create a new project File/New Project or click on ACCESS TO MCU SELECTOR, and check STM32G0 and LQFP64 package, to filter available MCUs, double click on STM32G071RB.
5.1.2 UCPD peripheral configuration

Now it is possible to activate the UCPD peripheral inside the microcontroller.

This peripheral manages the power delivery detection and its communication over the CC lines. For more details, refer to section 35 USB Type-C™ / USB Power Delivery interface (UCPD) of STM32G071 reference manual (RM0444).

Click on Connectivity, and select UCPD1. STM32G0 has two instances of the UCPD block. The TCPP01 shield is wired to UCPD instance 1. The use of instance 2 is possible, not plugging the shield onto the Nucleo board, but with flying wires.

This demonstration runs a sink application. So for UCPD mode, the user selects Sink. Untick the Dead Battery Signals to avoid using the internal dead-battery management of the UCPD peripheral, because, in the demonstration, the ST-LINK is powering the kit (Nucleo and shield) and the dead-battery management of the TCPP01 is bypassed, as DB-3.3V jumper is ON, to start with an easy application.
Now the DMA for TX and RX paths must be added, and the interrupts enabled.
Our device requires DMA initialization for the PD communication through the UCPD peripheral, so the user must configure a DMA channel for TX and RX. For example, set RX on DMA1 Channel 1, and TX on DMA1 Channel 2.

Figure 6. UCPD peripheral IT activation

Two DMA handlers are enabled but they are not directly used by the firmware. All the UCPD treatments are done through UCPD handlers.

5.1.3 FreeRTOS™ configuration

In the middleware category, enable FreeRTOS™. Select CMSIS_V1 and set TOTAL_HEAP_SIZE to 5000. From STM32L5 and further firmware package deliveries, CMSIS_V2 must be selected instead of CMSIS_V1.

Heap size here is for a start. It must be tuned later when optimizing the final application.

Figure 7. FreeRTOS configuration

5.1.4 USB-PD middleware configuration

In the middleware category, select USB-PD, then Port 0: UCPD1 in the drop-down list inside USBPD Mode and Configuration. For more details on USB-PD stack role in the overall firmware, refer to the user manual Managing USB power delivery systems with STM32 microcontrollers (UM2552).
Figure 8. USB-PD middleware configuration

Check that port 0 Sink PDO1 is set to 0x02019096, even if it is not important, because it is unused in the current simple example. 0x02019096 means 5000 mV, 1500 mA, dual-role data, without Fast Role Swap. For further information, see table 6-14 in document Universal Serial Bus Power Delivery Specification, Revision 3.0, Version 2.0, August 28, 2019, replicated below.

Figure 9. Specification detail (table 6-14 in Universal Serial Bus Power Delivery Specification)

<table>
<thead>
<tr>
<th>Bit(s)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>B31...30</td>
<td>Fixed supply</td>
</tr>
<tr>
<td>B29</td>
<td>Dual-Role Power</td>
</tr>
<tr>
<td>B28</td>
<td>Higher Capability</td>
</tr>
<tr>
<td>B27</td>
<td>Unconstrained Power</td>
</tr>
<tr>
<td>B26</td>
<td>USB Communications Capable</td>
</tr>
<tr>
<td>B25</td>
<td>Dual-Role Data</td>
</tr>
<tr>
<td>B24...23</td>
<td>Fast Role Swap required USB Type-C Current</td>
</tr>
</tbody>
</table>

Value  Description
00b     Fast Swap not supported (default)
01b     Default USB Power
10b     1.5A @ 5V
11b     3.0A @ 5V

B22...20 Reserved – Shall be set to zero.
B19...10 Voltage in 50mV units
B9...0  Operational Current in 10mA units
Figure 10 gives an explanation of the applied value 0x02019096.

**Figure 10. Detailed PDO decoding**

<table>
<thead>
<tr>
<th>Bit(s)</th>
<th>Description</th>
<th>Applied value</th>
<th>Value Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[31:20]</td>
<td>Fixed supply</td>
<td>00b</td>
<td>USBPD(PDO)_TYPE_FIXED</td>
</tr>
<tr>
<td>[29]</td>
<td>Dual-role power</td>
<td>0b</td>
<td>USBPD(PDO)_SNK_FIXED_DRP_NOT_SUPPORTED</td>
</tr>
<tr>
<td>[28]</td>
<td>Higher capability</td>
<td>0b</td>
<td>USBPD(PDO)_SNK_FIXED_HIGHERCAPABILITY_NOT_SUPPORTED</td>
</tr>
<tr>
<td>[27]</td>
<td>Unconstrained power</td>
<td>0b</td>
<td>USBPD(PDO)_SNK_FIXED_EXT_POWER_NOT_AVAILABLE</td>
</tr>
<tr>
<td>[26]</td>
<td>USB communications capable</td>
<td>0b</td>
<td>USBPD(PDO)_SNK_FIXED_USB_COMM_NOT_SUPPORTED</td>
</tr>
<tr>
<td>[25]</td>
<td>Dual-role data</td>
<td>1b</td>
<td>USBPD(PDO)_SNK_FIXED_ORD_SUPPORTED</td>
</tr>
<tr>
<td>[24:23]</td>
<td>Value Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>00b Fast swap not supported (default)</td>
<td>00b</td>
<td>USBPD(PDO)_SNK_FIXED_FRS_NOT_SUPPORTED</td>
</tr>
<tr>
<td></td>
<td>01b Default USB power</td>
<td>10b</td>
<td>1.5 A at 5 V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11b</td>
<td>3.0 A at 5 V</td>
</tr>
<tr>
<td>[19:10]</td>
<td>Voltage in 50 mV units</td>
<td>0001100100b = 0x64</td>
<td>0x64*50 = 5000 mV</td>
</tr>
<tr>
<td>[8:6]</td>
<td>Operational current in 16 mA units</td>
<td>0010010110b = 0x60</td>
<td>0x60*10 = 1500 mA</td>
</tr>
</tbody>
</table>

For more details on the PDO definition, have a look at the POWER_IF section in the user manual *Managing USB power delivery systems with STM32 microcontrollers* (UM2552).

**Figure 11. Detailed stack configuration**
In the Stack Port0 Parameters tab, the user selects what he wants to support, such as the PD3.0 specification revision, among other parameters. These parameters are all power delivery settings. There is no need to change them for a first application. For more information, have a look inside the Power Delivery specification (Universal Serial Bus Power Delivery Specification, Revision 3.0, Version 2.0, August 28, 2019).

5.1.5 ADC configuration for VBUS reading

VBUS detection is mandatory to respect the Type-C state machines. For this, we use an ADC connected to a voltage divider bridge, to remain in the GPIO STM32 voltage range.

Figure 12. TCPP01 shield voltage divider

Looking at the X-NUCLEO-USBPDM1 shield schematics, VBUS is on the ST morpho connector pin 34, corresponding to PB1

Figure 13. STM32G0 Nucleo-64 board (left) and TCPP01 shield (right) schematics

In the analog category, select ADC1.

Clicking on the right side in STM32CubeMX, PB1 is connected to the ADC1_IN9 alternate function input. So select it in the mode part of the STM32CubeMX window, or select the ADC1_IN9 in the pinout view.
Then in the **GPIO settings** tab, add the **User Label** VSENSE for this signal:
Select simple and basic settings for the ADC1:
- Clock Prescaler: Synchronous mode divided by 4
- Continuous conversion mode: Enabled
- Overrun behavior: Overrun data overwritten
- Sampling time: 160.5 Cycles

The sampling time must be adjusted with the impedance linked to the measure. In the case of the X-NUCLEO-USBPDM1 TCPP01 shield, there are higher than 10 KΩ resistors, therefore a high number of cycles is preferred.

**Figure 16. ADC parameters settings**

Last edition for ADC: A user constant VDDA_APPLI with 3300 value is created, representing the ADC voltage reference of 3.3 V. This variable is called by the generated code.

**Figure 17. ADC user constant**

**Note:** On the STM32G4 Nucleo-64 board, the ADC IN15 signal must be used instead of IN9, because of ADC mapping difference with STM32G0, and the rank sampling time can be set to 247.5 Cycles.
5.1.6 **Clock check**

PLLCLK can be chosen as an input clock of the system clock multiplexor, to produce SYSCLK and HCLK set to 16 MHz minimum. There is no limit for maximum. It can be 170 MHz for STM32G4. HSI is used to clock the UCPD peripheral, so it must be enabled.

![Figure 18. Clock configuration](image)

The mandatory settings for the simple USB-PD sink application are finished. The following part is highly recommended for debugging.
5.2 Additional recommended optional debugging

5.2.1 UART configuration for debug

On the STM32G0 Nucleo-64 board, the Virtual COM port connected to the ST-LINK is the LPUART1.

This interface is activated in STM32CubeMX, in asynchronous mode.

Important:
The default STM32CubeMX pins used by LPUART1 must be changed to match the STM32G0 Nucleo-64 hardware: PA2 for TX, and PA3 for RX.
Then the DMA requests are activated for the TX path only: DMA1 channel 3.

**Figure 21. DMA activation for LPUART**

<table>
<thead>
<tr>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset Configuration</td>
</tr>
<tr>
<td>NVIC Settings</td>
</tr>
<tr>
<td>DMA Settings</td>
</tr>
<tr>
<td>GPIO Settings</td>
</tr>
<tr>
<td>Parameter Settings</td>
</tr>
<tr>
<td>User Constants</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DMA Request</th>
<th>Channel</th>
<th>Direction</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPUART1_TX</td>
<td>DMA1 Channel 3</td>
<td>Memory To Perip...</td>
<td>Low</td>
</tr>
</tbody>
</table>

And the interrupt is activated:

**Figure 22. DMA activation for LPUART1**

<table>
<thead>
<tr>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset Configuration</td>
</tr>
<tr>
<td>NVIC Settings</td>
</tr>
<tr>
<td>DMA Settings</td>
</tr>
<tr>
<td>GPIO Settings</td>
</tr>
<tr>
<td>Parameter Settings</td>
</tr>
<tr>
<td>User Constants</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NVIC Interrupt Table</th>
<th>Ena.</th>
<th>Preemption</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMA1 channel 2 and channel 3 interrupts</td>
<td>✓</td>
<td>3</td>
</tr>
<tr>
<td>USART3, USART4 and LPUART1 interrupts / LPUART1 wake-up int.</td>
<td>✓</td>
<td>3</td>
</tr>
</tbody>
</table>

**Note:** If the STM32G4 Nucleo-64 board is used, USART1 must be used.
5.2.2 Activation of embedded tracer for debug

This is done in the utility category: TRACER_EMB is selected followed by the LPUART1 in the UART trace source mode.

Figure 23. Activation of TRACER_EMB
Back to the USB-PD middleware configuration, the trace evacuation is activated: check the tracer source for TRACER_EMB.

**Figure 24. Selection of USB-PD middleware TRACER_EMB source**

The firmware interactive stack responder can be activated if interaction with the USB-PD stack is needed, using the UCPD monitor tool STM32CubeMonUCPD.
5.2.3 Activation of UCPD monitor firmware responder

The monitor can simply be activated in the utility category: GUI_INTERFACE. Then enter free text to describe the board.

**Figure 25. Activation of GUI_INTERFACE**
5.3 Update and save project configuration

Once the configuration is finished, few parameters must be saved in the project manager tab before saving the project.

Under the project manager tab, select a name for the project. For the project directory, avoid using One drive, if STM32CubeMX is not in One drive too.

Configure the minimum heap and stack size to 0x400. This is the first version, which can be tuned later, depending on the application needs.

**Figure 26. Project manager settings**

In the Code Generator tab, STMicroelectronics recommends checking the Add necessary library files as reference tab.
Click on **Advanced Settings**

LPUART is selected as LL to save a bit of memory heap size.

Work must be saved: menu file / save
5.4 Code generation

Click on generate code.
A warning appears, informing that a proper HAL timebase is not defined.
It is safer to use a dedicated timer as a HAL timebase source.

Note: This becomes the recommended standard way of working in the forthcoming firmware package deliveries, especially when using CMSIS OS V2, which defines Systick as FreeRTOS™ timebase. For this demonstration, the below warning can be ignored by clicking Yes.

Then it is recommended to initialize Git to experiment some code and be able to roll back to previous versions, as in classic software development.
5.5 Compilation of generated application

The compilation must be performed without error or warning.

**Figure 30. First compilation**

In this project, different folders can be found:

- The **Application/User** folder contains the source files that we need to edit to enrich the application.
- The **Drivers** folder contains the HAL drivers for the STM32.
- The **Middleware** folder contains the source files and the libraries for FreeRTOS and USB-PD.
- The **Utilities** folder contains the GUI (UCPD monitor) and tracer embedded source files part.
- The **Output** folder contains the compilation result files.

5.6 Complete USB-PD application

Now that the peripherals are initialized by STM32CubeMX, some minimum level of the application needs to be added:

- ADC needs to be calibrated, and conversion needs to start.
- DPM timer must be enabled to run the USB-PD stack.
- Fill the handlers for the interrupts to wake up the UCPD peripheral.
- Fill `BSP_USBPD_PWR_VBUSGetVoltage` function with the right coefficient depending on the VBUS divider bridge.
- Complete `USBPD_DPM_SNK_EvaluateCapabilities` to answer to one source capability message.
- TCP01 dead battery pin needs to be disabled, GPIO driven HIGH, to see the source Rp, or the jumper has to be set on the shield.
5.6.1 Modification in src/main.c

In this file, the ADC must start after its calibration, using HAL. The ADC is needed to read $V_{BUS}$.

Code to be added:

```c
-  /* USER CODE BEGIN ADC1_Init 2 */
    HAL_ADCEx_Calibration_Start(&hadc1);
    HAL_ADC_Start(&hadc1);
    /* USER CODE END ADC1_Init 2 */
-  

Note: For STM32G4, ADC calibration API is different, the calibration line must be replaced by:

```c
HAL_ADCEx_Calibration_Start(&hadc1, sConfig.SingleDiff);
```

Note: This simple example is not optimized from power point of view, as the ADC is always running.

5.6.2 Modification in src/stm32g0xx_it.c

- The `USBPD_DPM_TimerCounter` function is used to manage the stack timers. It must be called in the function which is responsible of the HAL timebase management. (SysTick in this case, or the corresponding Timer function for example)
- TRACE module needs a DMA and an IRQ handler. So the corresponding code must be added in the right channel/IRQ place
- The UCPD handler must be added to be able to react upon coming interrupts, like cable detection or message reception.

```c
-  /* USER CODE BEGIN SysTick_IRQn 0 */
    USBPD_DPM_TimerCounter();
    #if defined( _GUI_INTERFACE)
        GUI_TimerCounter();        /* needed with GUI_Interface */
    #endif
    /* USER CODE END SysTick_IRQn 0 */
-  
-  /* USER CODE BEGIN DMA1_Channel2_3_IRQn 0 */
    #if defined(_TRACE)
        TRACER_EMB_IRQHandlerDMA();
    #endif
    /* USER CODE END DMA1_Channel2_3_IRQn 0 */
-  
-  /* USER CODE BEGIN USART3_4_LPUART1_IRQn 0 */
    #if defined(_TRACE)
        TRACER_EMB_IRQHandlerUSART();
    #endif
    /* USER CODE END USART3_4_LPUART1_IRQn 0 */
-  
-  /* USER CODE BEGIN UCPD1_2_IRQn 0 */
    USBPD_PORT0_IRQHandler();
    /* USER CODE END UCPD1_2_IRQn 0 */
-  
```
5.6.3 Modification in src/usbpd_dpm_user.c

The `USBPD_DPM_SNK_EvaluateCapabilities` function needs to be added to establish the first contract. It is a very basic example that requests the first default 5V PDO. This must be modified to match with real SINK PDOs, which is not yet managed by STM32CubeMX.

The part `_GUI_INTERFACE` compilation switch is needed if the user wants to interact with the UCPD monitor.

```c
/* USER CODE BEGIN USBPD_DPM_SNK_EvaluateCapabilities */
USBPD_SNKRDO_TypeDef rdo;
/* Initialize RDO */
rdo.d32 = 0;
/* Prepare the requested pdo */
rdo.FixedVariableRDO.ObjectPosition = 1;
rdo.FixedVariableRDO.OperatingCurrentIn10mAunits = 50;
rdo.FixedVariableRDO.MaxOperatingCurrent10mAunits = 50;
rdo.FixedVariableRDO.CapabilityMismatch = 0;
*PtrPowerObjectType = USBPD_CORE_PDO_TYPE_FIXED;
*PtrRequestData = rdo.d32;
/* USER CODE END USBPD_DPM_SNK_EvaluateCapabilities */
```

5.6.4 Modification in src/usbpd_pwr_user.c

It is important to add this part in order to correctly read VBUS provided by the ADC. The stack needs to know the VBUS level all along the cable presence to determine the action to take. In case of SINK, the detach is done when VBUS is below vSafe0V.

```c
/* USER CODE BEGIN include */
#include "main.h"
/* USER CODE END include */

/* USER CODE BEGIN BSP_PWR_VBUSGetVoltage */
uint32_t val;
val = __LL_ADC_CALC_DATA_TO_VOLTAGE( VDDA_APPLI, 
    LL_ADC_REG_ReadConversionData12(ADC1), 
    LL_ADC_RESOLUTION_12B); /* mV */
/* X-NUCLEO-USBPDM board is used */
/* Value is multiplied by 5.97 (Divider R6/R7 40.2 kΩ/200 kΩ) for VSENSE */
val *= 597;
val /= 100;
return val;
/* USER CODE END BSP_PWR_VBUSGetVoltage */
```

The calculation of val variable depends on the voltage divider shown in Figure 12. On the X-NUCLEO-USBPDM1 shield, Value is multiplied by 5.97 (Divider R6/R7 40.2 kΩ/200 kΩ) for VSENSE. For STM32G4, because of BSP v2.0, there is a slight change in this function interface. Instead of the line return val above, there must be:

```c
*pVoltage = val;
return BSP_ERROR_NONE;
```

Note: There is a known issue with STM32G4 in src/usbpd.c: GUI_Init must be called after the USBPD_DPM_InitCore. This is fixed in STM32G4 firmware 1.2.0 and further.
5.7 Check jumpers

Figure 31. TCPP01 shield picture

Verify that the two JP1 jumpers located on the right to select the STM32G0 and STM32G4 configuration are inserted.
Then select the pins that are controlled by the MCU, using the left JP2 jumpers:

**Figure 32. TCPP01 jumper settings**

- Fault detection and hard reset are not managed in this demonstration.
- The power consumption is also not optimized. This is the reason why the TCPP01 VCC is set to the fixed 3.3 V, instead of taking an MCU GPIO, so the JP2 jumper VCC-3.3V [7-8] is ON.
- In the first step demonstration, the dead battery from the TCPP01 is not used, so the JP2 jumper DB-3.3V [11-12] is ON.

This concludes the settings.
Now is the time to check the first contract.
Establish the first explicit contract

Compile the application, flash the board, start the STM32G0 program, keep the USB cable plugged, as the Virtual COM port is mandatory, and launch the UCPD monitor. The user's board must appear in the list when clicking "Refresh list of connected boards", so double click on the corresponding line (or click "NEXT").

Figure 33. Board selection

Note: The ComPort may be different. It depends on the number of boards installed on the computer. Then double click on the desired UCPD port, here Port 0, or select it and click "NEXT".

Figure 34. Port selection
Click on the "TRACES" button in the bottom right corner to get protocol traces. Then it is possible to plug a power delivery source cable to the Type-C™ receptacle of the X-NUCLEO-USBPDM1 shield. The screen may look like Figure 35:

**Figure 35. Explicit contract visible in UCPD monitor**

![Figure 35](image)

Note: The SRC PDO part may look different. It depends on the capabilities of the power source.

Figure 35 shows the communication between the STM32G0 and the power delivery source on the right panel. It is possible to verify the correct sequence to reach an explicit contract:
1. The capabilities are sent by the source (IN green message).
2. The request is sent by the STM32G0 (OUT orange message).
3. The ACCEPT and the PS_RDY are sent by the source (IN green message).

For more details on how to use this tool, refer to UM2468.

And for more details on the protocol, refer to UM2552.
### 6.1 How to debug a bit deeper

- The `__DEBUG_CAD` compilation switch can be set to have more information in trace about CAD state machine. Refer to Figure 36, where CAD appears in column `Type`.

#### Figure 36. Example of CAD debug information visible in the trace

<table>
<thead>
<tr>
<th>Type</th>
<th>TimeStamp</th>
<th>Port</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAD</td>
<td>906209</td>
<td>0</td>
<td>USBPD_CAD_STATE_SWITCH_TO_SRC</td>
</tr>
<tr>
<td>EVENT</td>
<td>906209</td>
<td>0</td>
<td>EVENT_DETACHED</td>
</tr>
<tr>
<td>NOTIF</td>
<td>906209</td>
<td>0</td>
<td>POWER_STATE_CHANGE</td>
</tr>
<tr>
<td>DEBUG</td>
<td>906209</td>
<td>0</td>
<td>HELP: update USBPD_DPM_SetDataInfo:7</td>
</tr>
<tr>
<td>DEBUG</td>
<td>906209</td>
<td>0</td>
<td>HELP: update USBPD_DPM_SetDataInfo:2</td>
</tr>
<tr>
<td>PE</td>
<td>906209</td>
<td>0</td>
<td>PE_SNIC_STARTUP</td>
</tr>
<tr>
<td>CAD</td>
<td>906409</td>
<td>0</td>
<td>USBPD_CAD_STATE_DETACHED</td>
</tr>
<tr>
<td>DEBUG</td>
<td>914101</td>
<td>0</td>
<td>HELP: Update BSP_PWR_VBUSInit</td>
</tr>
<tr>
<td>CAD</td>
<td>914101</td>
<td>0</td>
<td>USBPD_CAD_STATE_ATTACHED_WAIT</td>
</tr>
<tr>
<td>CAD</td>
<td>914316</td>
<td>0</td>
<td>USBPD_CAD_STATE_ATTACHED0</td>
</tr>
</tbody>
</table>
• Add *livewatch* on CAD_HW_Handles. This variable can be used to check the Type-C attachment or detachment.

See usbpdcadhw_if.c:

```c
#define USBPD_CAD_STATE_RESET                     0u  /*!< USBPD CAD State Reset */
#define USBPD_CAD_STATE_DETACHED                  1u  /*!< USBPD CAD State No cable detected */
#define USBPD_CAD_STATE_ATTACHED_WAIT             2u  /*!< USBPD CAD State Port partner detected */
#define USBPD_CAD_STATE_ATTACHED                  3u  /*!< USBPD CAD State Port partner attached */
```

**Figure 37. cstate=1: detached**

| CAD_HW_Handles | <array> | 0x200025A0 | CAD_HW_H...
|----------------|---------|-------------|-----------
| [0]            | struct  | 0x200025A0 | uint32_t |
| cc             | 0       | 0x200025A0 | uint32_t |
| SNK_Source_Current_Adv | 0       | 0x200025A0 | uint32_t |
| Current_Hwcondition | 0       | 0x200025A0 | uint32_t |
| CAD_IdDebounce_flag | 0       | 0x200025A0 | uint32_t |
| CAD_ErrorRecoveryflag | 0       | 0x200025A0 | uint32_t |
| reserved       | 0       | 0x200025A0 | uint32_t |
| cstate         | 1       | 0x200025A0 | uint32_t |
| pstate         | 1       | 0x200025A0 | uint32_t |
| CAD_IdDebounce_start | 0       | 0x200025A4 | uint32_t |

If the CC lines level changes are invisible, check that the TCPP is powered on, and that the active low _DB_ pin is not set at 3.3 V. It may come from the JP2 jumper or some GPIO settings, like pull up resistors.

Note: In the current STM32CubeMX for STM32G4, there is an issue with the default GPIO mode for CC2. In usbpdcadhw_if.c there must be:

```c
LL_GPIO_SetPinMode(GPIOB, LL_GPIO_PIN_4, LL_GPIO_MODE_ANALOG);
```

In STM32G4 versions before firmware 1.2.0, the correct compilation switch is not set. An easy way to correct this issue is to activate the compilation switch MB1367.
For further debugging, the $V_{BUS}$ measured value can be printed in the trace, using the user button:

**Figure 39. User button on STM32G0 Nucleo-64 board schematics**

Add the button from STM32CubeMX as described in Figure 40.

**Figure 40. Add the user button in STM32CubeMX**
Add in src/main.c:

```c
/**
 * @brief EXTI line detection callbacks
 * @param GPIO_Pin Specifies the pins connected EXTI line
 * @retval None
 */
void HAL_GPIO_EXTI_Falling_Callback(uint16_t GPIO_Pin)
{
    if (GPIO_Pin == USER_BUTTON_PIN) /* Will display in trace the VBUS value when user button is pressed */
    {
        char _str[10];
        BSP_PWR_VBUSGetVoltage(0);
        sprintf(_str, "VBUS:%d", BSP_PWR_VBUSGetVoltage(0));
        USBPD_TRACE_Add(USBPD_TRACE_DEBUG, 0, 0, (uint8_t*)_str, strlen(_str));
    }
}
```

And the corresponding interrupt in src/stm32g0xx_it.c:

```c
/**
 * @brief This function handles external line 4_15 interrupt request.
 * @retval None
 */
void EXTI4_15_IRQHandler(void)
{
    HAL_GPIO_EXTI_IRQHandler(USER_BUTTON_PIN);
}
```
TCPP01 can be better controlled by driving its dead battery pin and its power.

**Figure 41. Modify TCPP01 controls**

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Signal on Pin</th>
<th>GPIO Pin</th>
<th>GPIO Mode</th>
<th>GPIO Pull-up</th>
<th>Maximum Power Mode</th>
<th>Active Mode</th>
<th>User Label</th>
<th>Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>PB6</td>
<td>n/a</td>
<td>High</td>
<td>Output Pulled Up</td>
<td>Pull-up</td>
<td>Low</td>
<td>Disable</td>
<td>DB_</td>
<td></td>
</tr>
<tr>
<td>PC10</td>
<td>n/a</td>
<td>High</td>
<td>Output Pulled Up</td>
<td>Pull-up</td>
<td>Low</td>
<td>n/a</td>
<td>VCC_OUT_</td>
<td></td>
</tr>
</tbody>
</table>

If these pins need to be controlled by the application, the jumper positions must be adapted. The potential faults must also be read by setting the JP2 jumper in [5-6] position.

**Figure 42. TCPP01 shield jumpers position when application manages FLT, DB, and the TCPP01 VCC**

The user application also needs to be done to react upon the TCPP01 fault detection (Over-temperature, over-voltage).
This demonstration is only the first step to a power delivery application. It is not optimized from the low power point of view.

The USB PD application performed here is the minimum. No intelligence is added to select the power level by looking at the proposal sent by the source. The mandatory hard reset management is also missing.

To continue further, various demonstrations are available on STM32G0, STM32G4, and STM32L5. Check the Projects directory in the firmware package available in each serie, on the ST website. For instance, UCPD demonstration on EVAL_G0 is available under the folder .\Projects\STM32G081B-EVAL\Demonstrations\DemoUCPD.
## Revision history

Table 2. Document revision history

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>9-Jan-2020</td>
<td>1</td>
<td>Initial release</td>
</tr>
<tr>
<td>30-Apr-2020</td>
<td>2</td>
<td>Added specific technical data throughout the document</td>
</tr>
</tbody>
</table>
## Contents

1. **General information** ............................................................... 2
2. **Acronyms** ......................................................................... 3
3. **Reference documents** ............................................................. 4
4. **Getting started** .................................................................... 5
5. **STM32CubeMX step-by-step sequence** ............................................. 6
   5.1 Mandatory parts .................................................................. 6
   5.1.1 Start STM32CubeMX and select the MCU ............................. 6
   5.1.2 UCPD peripheral configuration ........................................... 7
   5.1.3 FreeRTOS™ configuration ................................................... 9
   5.1.4 USB-PD middleware configuration ....................................... 9
   5.1.5 ADC configuration for V\_BUS reading ............................... 12
   5.1.6 Clock check ................................................................... 15
   5.2 Additional recommended optional debugging .......................... 16
   5.2.1 UART configuration for debug ............................................ 16
   5.2.2 Activation of embedded tracer for debug ............................ 18
   5.2.3 Activation of UCPD monitor firmware responder .................. 20
   5.3 Update and save project configuration ................................... 21
   5.4 Code generation .................................................................. 23
   5.5 Compilation of generated application .................................... 24
   5.6 Complete USB-PD application ............................................. 24
   5.6.1 Modification in src/main.c ............................................... 25
   5.6.2 Modification in src/stm32g0xx_it.c .................................... 25
   5.6.3 Modification in src/usbpd_dpm_user.c ................................ 26
   5.6.4 Modification in src/usbpd_pwr_user.c ................................. 26
   5.7 Check jumpers .................................................................. 27
6. **Establish the first explicit contract** ........................................ 29
   6.1 How to debug a bit deeper .................................................. 31
7. **Next steps** ......................................................................... 35
8. **Conclusion** ........................................................................ 36
Revision history ................................................................. 37
Contents ............................................................................. 38
List of tables ....................................................................... 40
List of figures ...................................................................... 41
List of tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>Acronym definitions</td>
<td>3</td>
</tr>
<tr>
<td>Table 2</td>
<td>Document revision history</td>
<td>37</td>
</tr>
</tbody>
</table>
### List of figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>STM32G0 Nucleo-64 board equipped with X-NUCLEO-USBPDM1 shield</td>
<td>1</td>
</tr>
<tr>
<td>Figure 2</td>
<td>Start STM32CubeMX</td>
<td>6</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Select the STM32G0 MCU</td>
<td>7</td>
</tr>
<tr>
<td>Figure 4</td>
<td>UCPD peripheral basic configuration</td>
<td>8</td>
</tr>
<tr>
<td>Figure 5</td>
<td>UCPD peripheral DMA configuration</td>
<td>8</td>
</tr>
<tr>
<td>Figure 6</td>
<td>UCPD peripheral IT activation</td>
<td>9</td>
</tr>
<tr>
<td>Figure 7</td>
<td>FreeRTOS configuration</td>
<td>9</td>
</tr>
<tr>
<td>Figure 8</td>
<td>USB-PD middleware configuration</td>
<td>10</td>
</tr>
<tr>
<td>Figure 9</td>
<td>Specification detail (table 6-14 in Universal Serial Bus Power Delivery Specification)</td>
<td>10</td>
</tr>
<tr>
<td>Figure 10</td>
<td>Detailed PDO decoding</td>
<td>11</td>
</tr>
<tr>
<td>Figure 11</td>
<td>Detailed stack configuration</td>
<td>11</td>
</tr>
<tr>
<td>Figure 12</td>
<td>TCPP01 shield voltage divider</td>
<td>12</td>
</tr>
<tr>
<td>Figure 13</td>
<td>STM32G0 Nucleo-64 board (left) and TCPP01 shield (right) schematics</td>
<td>12</td>
</tr>
<tr>
<td>Figure 14</td>
<td>ADC configuration</td>
<td>13</td>
</tr>
<tr>
<td>Figure 15</td>
<td>ADC GPIO settings</td>
<td>13</td>
</tr>
<tr>
<td>Figure 16</td>
<td>ADC parameters settings</td>
<td>14</td>
</tr>
<tr>
<td>Figure 17</td>
<td>ADC user constant</td>
<td>14</td>
</tr>
<tr>
<td>Figure 18</td>
<td>Clock configuration</td>
<td>15</td>
</tr>
<tr>
<td>Figure 19</td>
<td>STM32G0 Nucleo-64 board STLK connection</td>
<td>16</td>
</tr>
<tr>
<td>Figure 20</td>
<td>LPUART1 activation and GPIO pin (TX and RX) update</td>
<td>16</td>
</tr>
<tr>
<td>Figure 21</td>
<td>DMA activation for LPUART</td>
<td>17</td>
</tr>
<tr>
<td>Figure 22</td>
<td>DMA activation for LPUART1</td>
<td>17</td>
</tr>
<tr>
<td>Figure 23</td>
<td>Activation of TRACER_EMB</td>
<td>18</td>
</tr>
<tr>
<td>Figure 24</td>
<td>Selection of USB-PD middleware TRACER_EMB source</td>
<td>19</td>
</tr>
<tr>
<td>Figure 25</td>
<td>Activation of GUI_INTERFACE</td>
<td>20</td>
</tr>
<tr>
<td>Figure 26</td>
<td>Project manager settings</td>
<td>21</td>
</tr>
<tr>
<td>Figure 27</td>
<td>Code generator settings</td>
<td>22</td>
</tr>
<tr>
<td>Figure 28</td>
<td>Project advanced setting</td>
<td>22</td>
</tr>
<tr>
<td>Figure 29</td>
<td>Generation warning</td>
<td>23</td>
</tr>
<tr>
<td>Figure 30</td>
<td>First compilation</td>
<td>24</td>
</tr>
<tr>
<td>Figure 31</td>
<td>TCPP01 shield picture</td>
<td>27</td>
</tr>
<tr>
<td>Figure 32</td>
<td>TCPP01 jumper settings</td>
<td>28</td>
</tr>
<tr>
<td>Figure 33</td>
<td>Board selection</td>
<td>29</td>
</tr>
<tr>
<td>Figure 34</td>
<td>Port selection</td>
<td>29</td>
</tr>
<tr>
<td>Figure 35</td>
<td>Explicit contract visible in UCPD monitor</td>
<td>30</td>
</tr>
<tr>
<td>Figure 36</td>
<td>Example of CAD debug information visible in the trace</td>
<td>31</td>
</tr>
<tr>
<td>Figure 37</td>
<td>cstate=1: detached</td>
<td>32</td>
</tr>
<tr>
<td>Figure 38</td>
<td>cstate=3: attached</td>
<td>32</td>
</tr>
<tr>
<td>Figure 39</td>
<td>User button on STM32G0 Nucleo-64 board schematics</td>
<td>33</td>
</tr>
<tr>
<td>Figure 40</td>
<td>Add the user button in STM32CubeMX</td>
<td>33</td>
</tr>
<tr>
<td>Figure 41</td>
<td>Modify TCPP01 controls</td>
<td>35</td>
</tr>
<tr>
<td>Figure 42</td>
<td>TCPP01 shield jumpers position when application manages FLT, DB, and the TCPP01 VCC</td>
<td>35</td>
</tr>
</tbody>
</table>