AN5418

Application note

How to build a simple USB-PD sink application with STM32CubeMX

Introduction

This application note is a guideline to build a very simple USB power delivery sink example, starting from STM32CubeMX. This document applies to all STM32 MCUs embedding the UCPD (USB Type-C®Power Delivery controller) 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.

The X-NUCLEO-USBPDM1 or X-NUCLEO-SNK1M1 shield associates a TCPP01-M12 protection circuit and provides a USB Type-C® connector. The STM32L5 Nucleo-144, Discovery, and Evaluation boards are USB-PD ready as they embed the TCPP01-M12 chip.

This document details how to build a USBPD sink application with the two shield versions shown in Figure 1 and Figure 2. X-NUCLEO-SNK1M1 has some jumpers that are replaced by solder bridges, and only one additional feature not connected by default jumpers, which is SINK 5V up to 3A without USB Power Delivery.

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

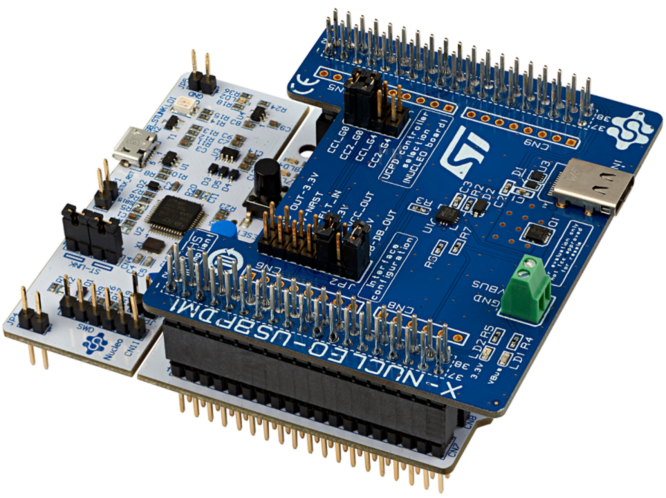
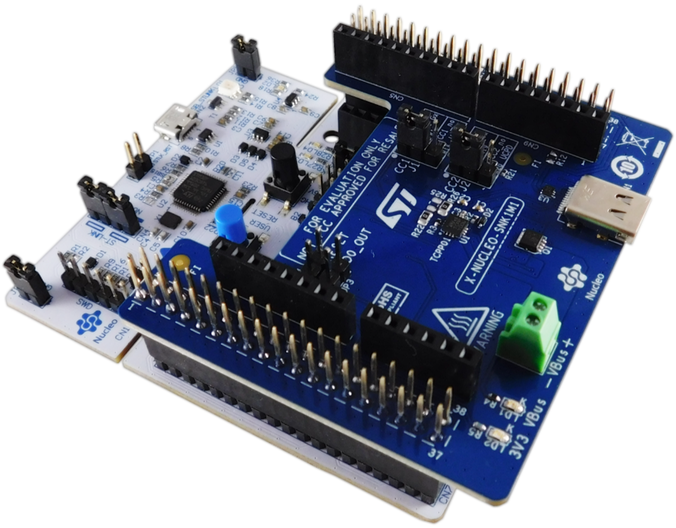


Figure 2. STM32G0 Nucleo-64 board equipped with X-NUCLEO-SNK1M1 shield



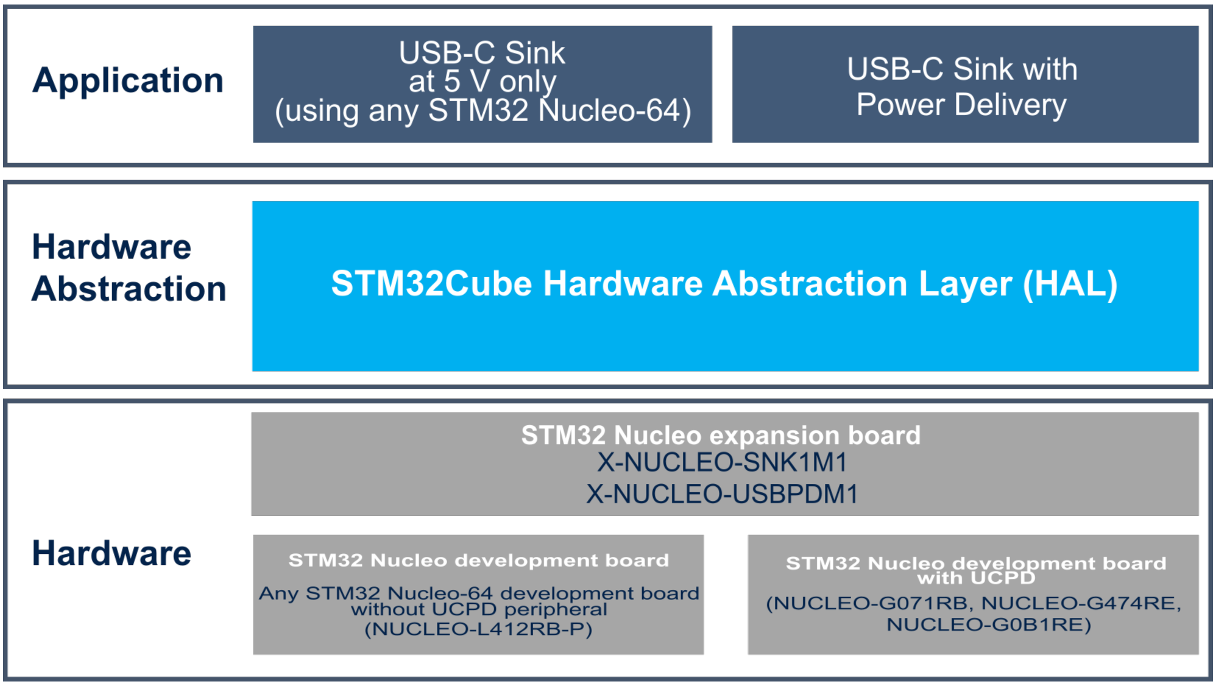
Pictures are not contractual.

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.

Figure 3. **X-CUBE-TCPP block diagram architecture**



Even if this application note targets the creation of a USB-PD application, the TCPP01-M12 shields can also be used for Type-C applications, as described by the architecture shown in Figure 3, in the top left application boxes. The Type-C-only applications are not described here. For source (TCPP02-M18) or DRP (TCPP03-M20) applications, other shields are available and are not addressed in this document either.

2 Acronyms

Table 1. Acronym definitions

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

3 Reference documents

STMicroelectronics ecosystem material:

Name	Title/description
STM32CubeMX	STM32CubeMX: STM32Cube initialization code generator
UM2552	User manual *Managing USB power delivery systems with STM32 microcontrollers* (UM2552)
UM2468	User manual *STM32CubeMonitor-UCPD software tool for USB Type-C™ Power Delivery port management* (UM2468)
DS12900	TCPP01-M12 datasheet *USB-C overvoltage protection for VBUS and CC lines* (DS12900)
AN4871	Application note *USB Type-C protection and filtering* (AN4871)
DB3747	Databrief *STM32CubeMonitor-UCPD software tool for USB Type-C™ Power Delivery port management* (DB3747)
TA0357	Technical article *Overview of USB Type-C and Power Delivery technologies* (TA0357)
AN5225	Application note *USB Type-C Power Delivery using STM32 MCUs and MPUs* (AN5225)
[YouTube_video]	YouTube video STM32G0: Create a USB Power delivery sink application in less than 10 minutes
UM2773	User manual *Getting started with the X-NUCLEO-SNK1M1 USB Type-C™ Power Delivery Sink expansion board based on TCPP01-M12 for STM32 Nucleo* (UM2773)
[USBPD_overview_wiki]	USBPD overview wiki

USB specification documents:

Name	Title/description
[USB2.0 specification]	USB2.0 Universal Serial Bus Revision 2.0 Specification
[USB3.1 specification]	USB3.1 Universal Serial Bus Revision 3.2 Specification
[USB battery charging specification]	USB BC Battery Charging Specification Revision 1.2
[USB Type-C® cable and connector specification]	Universal Serial Bus Type-C Cable and Connector Specification 2.0, August 2019
[USB-PD specification]	Universal Serial Bus Power Delivery Specification, Revision 3.0, Version 2.0, August 28, 2019

4 Getting started

The goal is to configure the UCPD peripheral with a 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 (or the new STM32G0 with added USB support) and STM32G474xx. This peripheral manages power delivery communication over the CC lines. The X-NUCLEO-USBPDM1 or the X-NUCLEO-SNK1M1 shields, embedding the TCPP01-M12 single-chip solution, adds the microcontroller protection on the CC lines, such as ESD, over-voltage, and over-temperature, and it also gives access to a USB Type-C® receptacle.

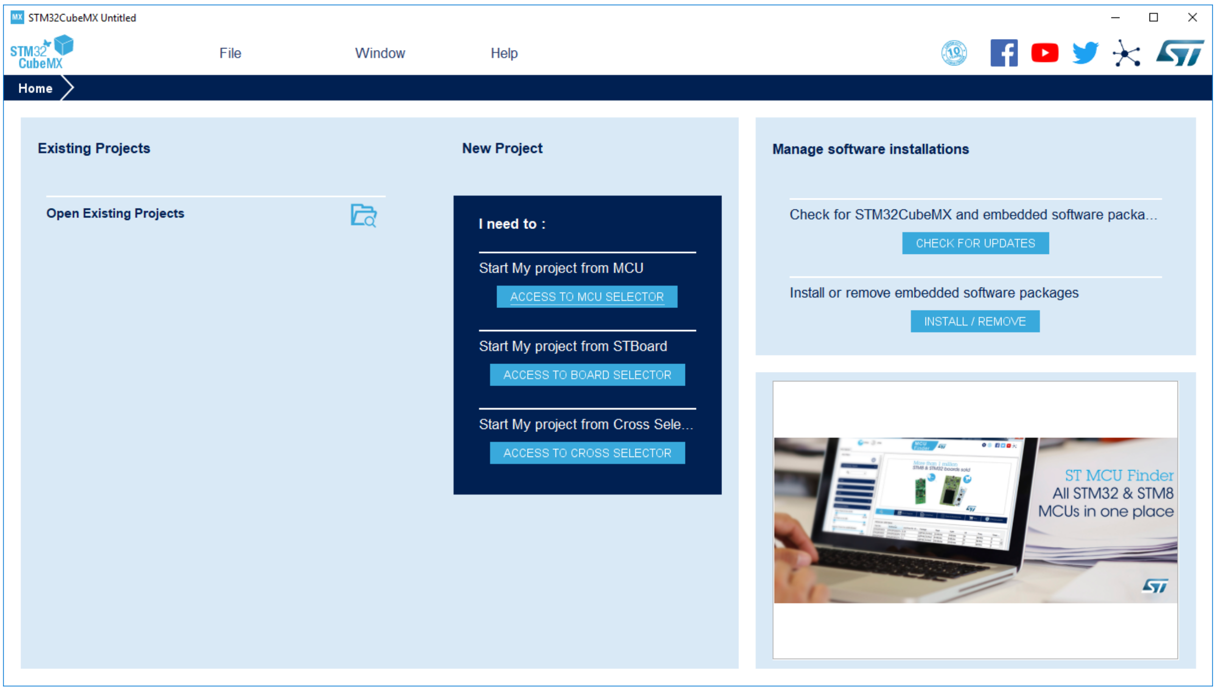
TCPP01-M12 is already embedded in the STM32L5 Nucleo-144, Discovery, and Evaluation boards.

Before starting, check first that both latest versions of STM32CubeMX and STM32CubeG0, STM32CubeG4, or STM32CubeL5 MCU Packages are used.

The sequence described in the following subsections based on the NUCLEO-G071RB STM32G0 Nucleo-64 board with the TCPP01-M12 shield is adaptable to other configurations.

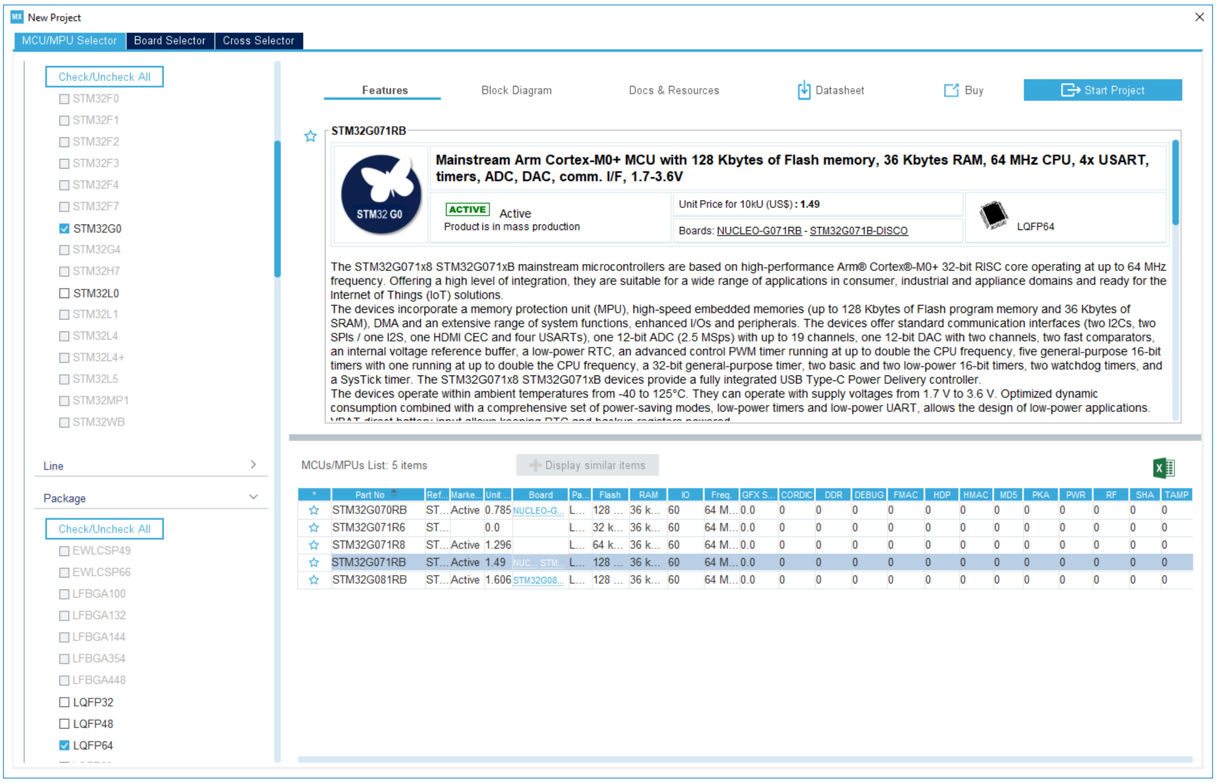
5.1.1 Start STM32CubeMX and select the MCU

Figure 4. **Start STM32CubeMX**



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*.

Figure 5. Select the STM32G0 MCU



5.1.2 UCPD peripheral configuration

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

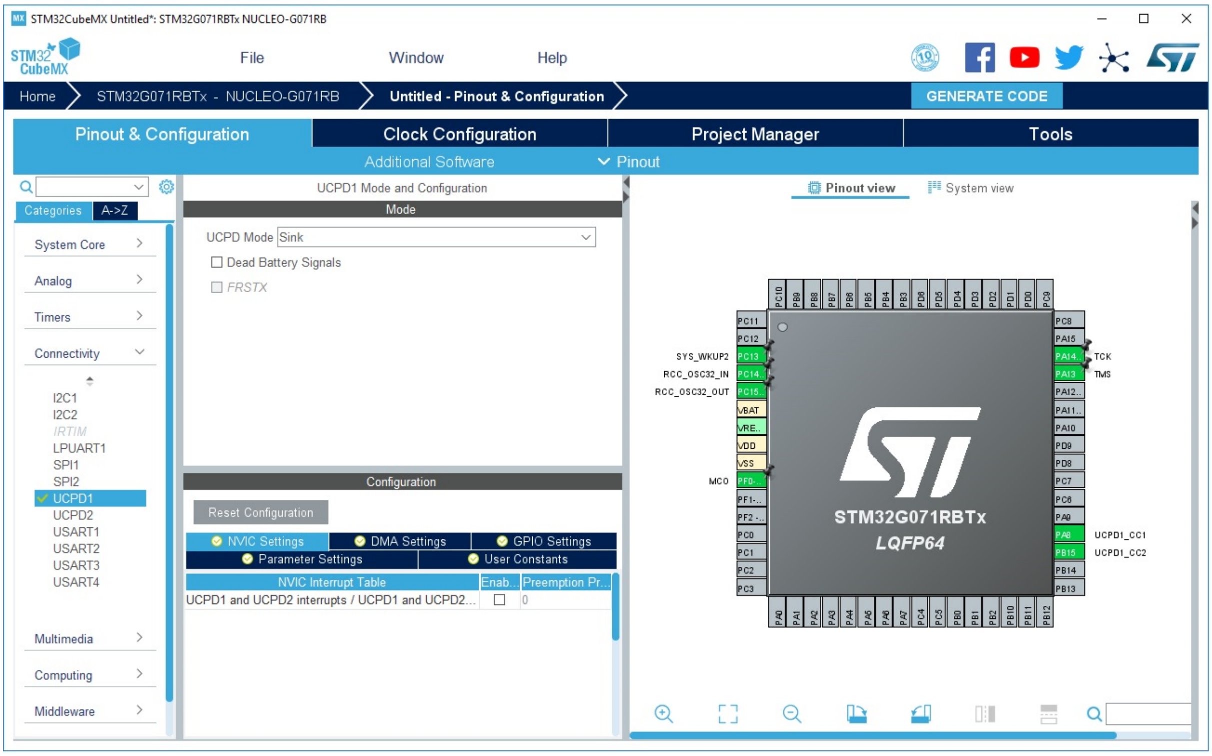
This peripheral manages the power delivery detection and its communication over the CC lines. For more details, refer to *section 35 USB Type-C™ / USB Power Delivery interface (UCPD)* of the reference manual *STM32G0x1 advanced Arm®-based 32-bit MCUs* (RM0444).

Click on *Connectivity*, and select UCPD1. STM32G0 has two instances of the UCPD block. The TCPP01-M12 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-M12 is bypassed, as DB-3.3V jumper is ON, to start with an easy application.

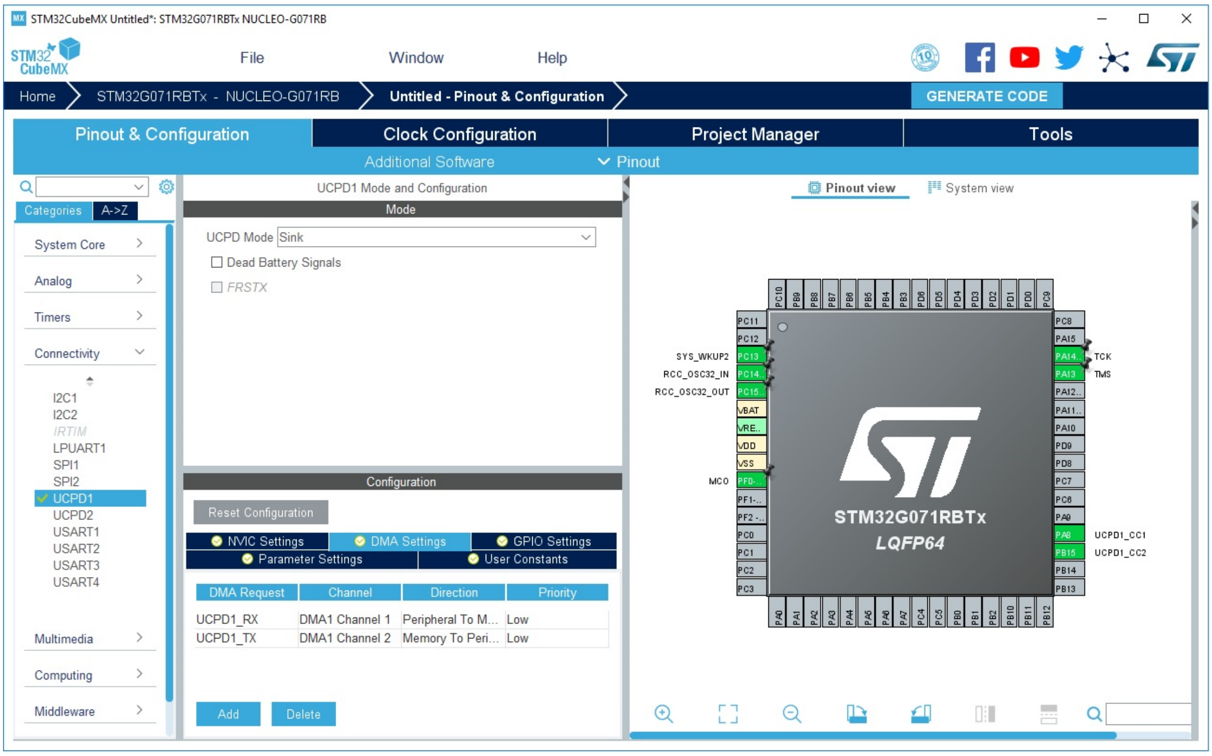
Note: The shield jumper configuration for a dead battery is detailed in *Section 5.7*

Figure 6. **UCPD peripheral basic configuration**



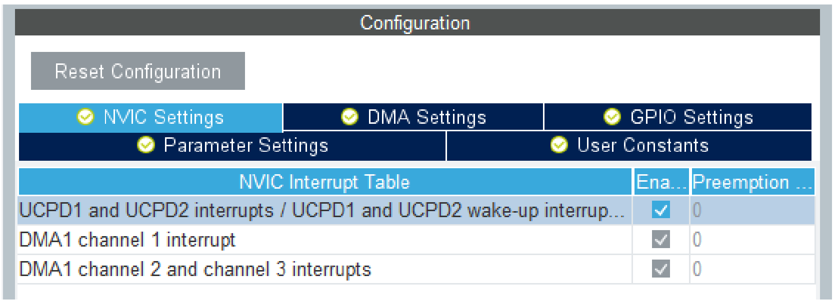
Now the DMA for TX and RX paths must be added, and the interrupts enabled.

Figure 7. **UCPD peripheral DMA configuration**



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 8. 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*:

- Enable *eTaskGetState* among included parameters as shown in Figure 9.
- Set *TOTAL_HEAP_SIZE* to 7000 as shown in Figure 10.

From STM32L5 and further firmware package deliveries, CMSIS_V2 must be selected instead of CMSIS_V1.

Figure 9. FreeRTOS™ parameters

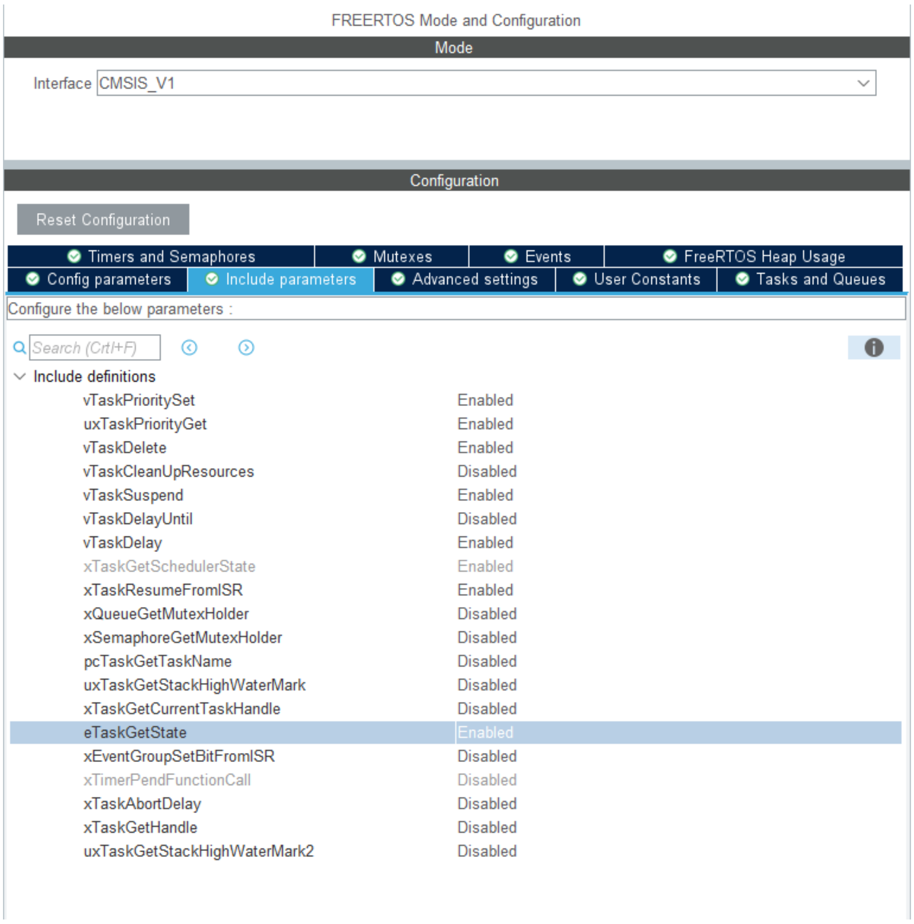
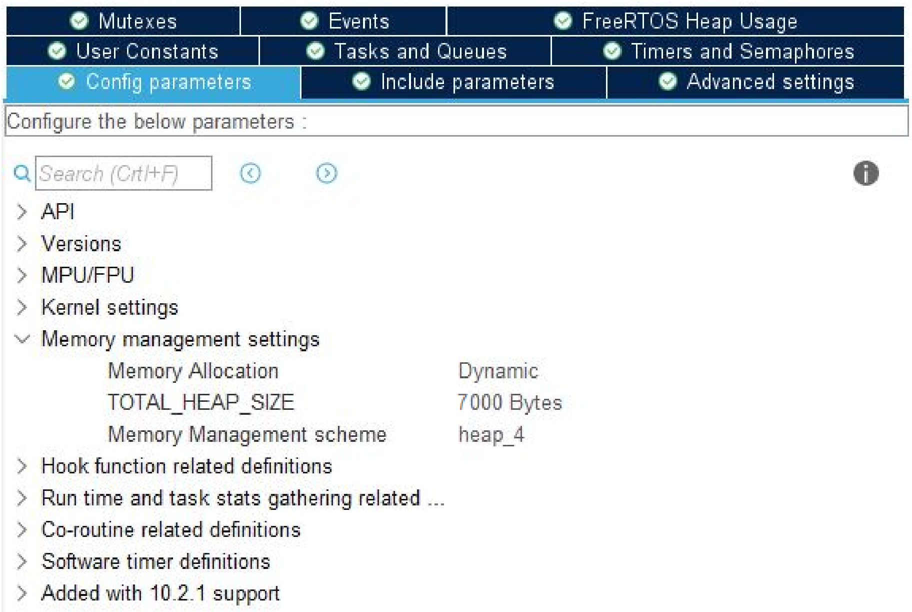


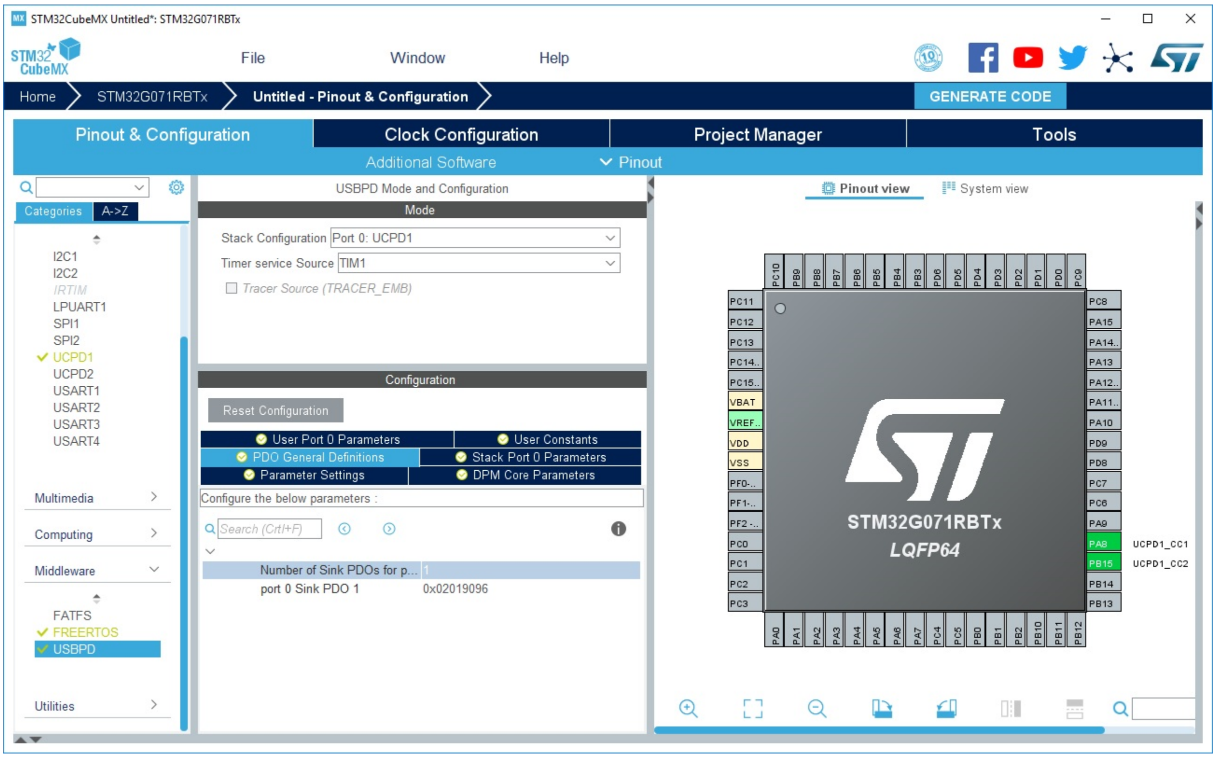
Figure 10. FreeRTOS™ configuration



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 the USB-PD stack role in the overall firmware, refer to UM2552.

Figure 11. 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, refer to Table 6-14 in [USB-PD specification], replicated in Figure 12.

Figure 12. Specification detail (table 6-14 in Universal Serial Bus Power Delivery Specification)

Table 6-14 Fixed Supply PDO - Sink

Bit(s)	Description
B31…30	Fixed supply
B29	Dual-Role Power
B28	Higher Capability
B27	Unconstrained Power
B26	USB Communications Capable
B25	Dual-Role Data
B24…23	Fast Role Swap required USB Type-C Current (see also ***[USB Type-C 1.3]***): <table><tr><th>Value</th><th>Description</th></tr><tr><td>00b</td><td>Fast Swap not supported (default)</td></tr><tr><td>01b</td><td>Default USB Power</td></tr><tr><td>10b</td><td>1.5A @ 5V</td></tr><tr><td>11b</td><td>3.0A @ 5V</td></tr></table>
B22…20	***Reserved – Shall*** be set to zero.
B19…10	Voltage in 50mV units
B9…0	Operational Current in 10mA units

Figure 13 explains the applied value `0x02019096`.

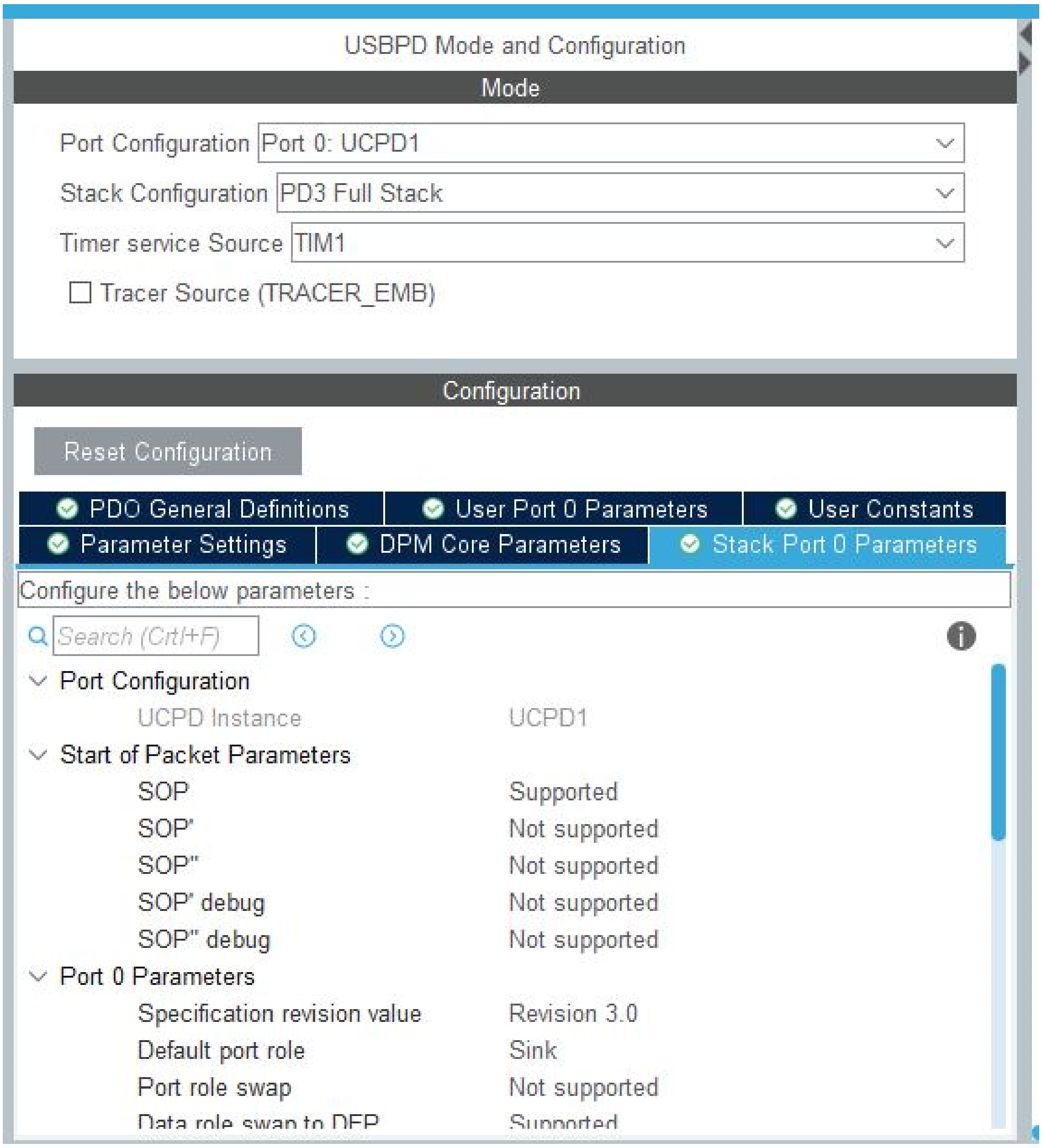
Figure 13. Detailed PDO decoding

0x02019096 = **00 0 0 0 0 1 00 000 0001100100** 0010010110b

Bit(s)	Description	Applied value	The definitions can be found in STM32 CubeG0 Firmware\Projects\STM32G081B-EVAL\Demonstrations \DemoUCPD\Inc\usbpd_pdo_defs.h
[31,30]	**Fixed supply**	**00b**	**USBPD_PDO_TYPE_FIXED**
[29]	Dual-role power	0b	USBPD_PDO_SNK_FIXED_DRP_NOT_SUPPORTED
[28]	**Higher capability**	**0b**	**USBPD_PDO_SNK_FIXED_HIGHERCAPAB_NOT_SUPPORTED**
[27]	Unconstrained power	0b	USBPD_PDO_SNK_FIXED_EXT_POWER_NOT_AVAILABLE
[26]	**USB communications capable**	**0b**	**USBPD_PDO_SNK_FIXED_USBCOMM_NOT_SUPPORTED**
[25]	Dual-role data	1b	USBPD_PDO_SNK_FIXED_DRD_SUPPORTED
[24,23]	<table><tr><th>Value</th><th>Description</th></tr><tr><td>00b</td><td>Fast swap not supported (default)</td></tr><tr><td>01b</td><td>Default USB power</td></tr><tr><td>10b</td><td>1.5 A at 5 V</td></tr><tr><td>11b</td><td>3.0 A at 5 V</td></tr></table>	**00b**	**USBPD_PDO_SNK_FIXED_FRS_NOT_SUPPORTED**
[22-20]	Reserved – Must be set to zero.	000b	
[19-10]	**Voltage in 50 mV units**	**0001100100b = 0x64**	**0x64*50 = 5000 mV**
[9-0]	Operational current in 10 mA units	0010010110b = 0x96	0x96*10 = 1500 mA

For more details on the PDO definition, look at the POWER_IF section in UM2552.

Figure 14. Detailed stack configuration

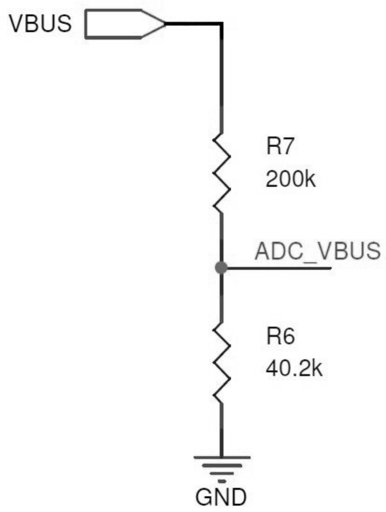


In the *Stack Port0 Parameters* tab, the user selects what he wants to support, such as the PD3.0 specification revision, among other parameters. These parameters are all power delivery settings. There is no need to change them for a first application. For more information, refer to [USB-PD specification].

5.1.5 ADC configuration for V_{BUS} reading

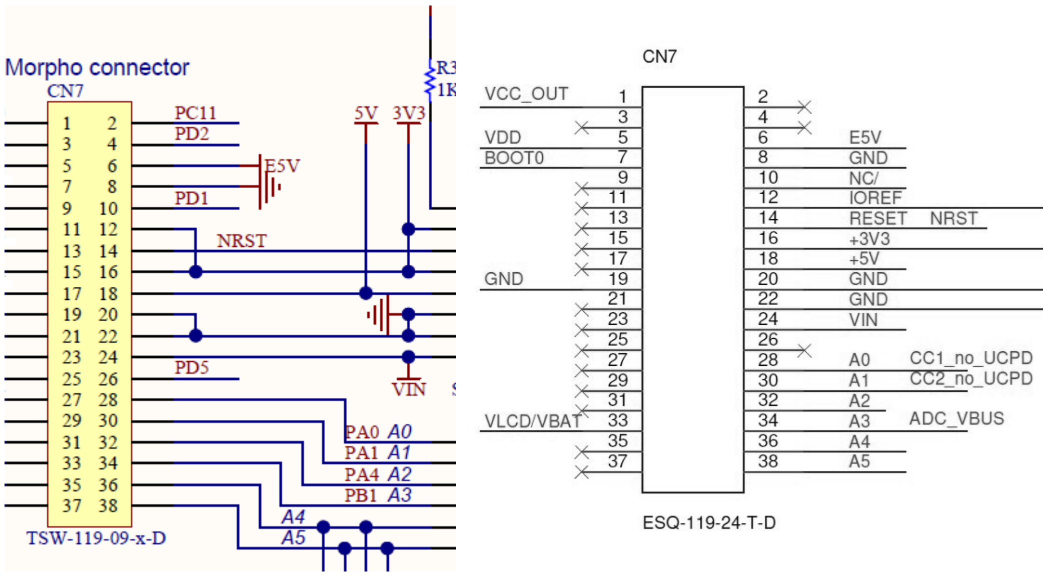
V_{BUS} detection is mandatory to respect the Type-C state machines. For this, we use an ADC connected to a voltage divider bridge, to remain in the GPIO STM32 voltage range.

Figure 15. TCPP01-M12 shield voltage divider



Looking at the X-NUCLEO-SNK1M1 shield schematics in the *Schematic Pack* under *CAD Resources*, V_{BUS} is on the CN7 ST morpho connector pin 34, corresponding to PB1, as shown in Figure 16.

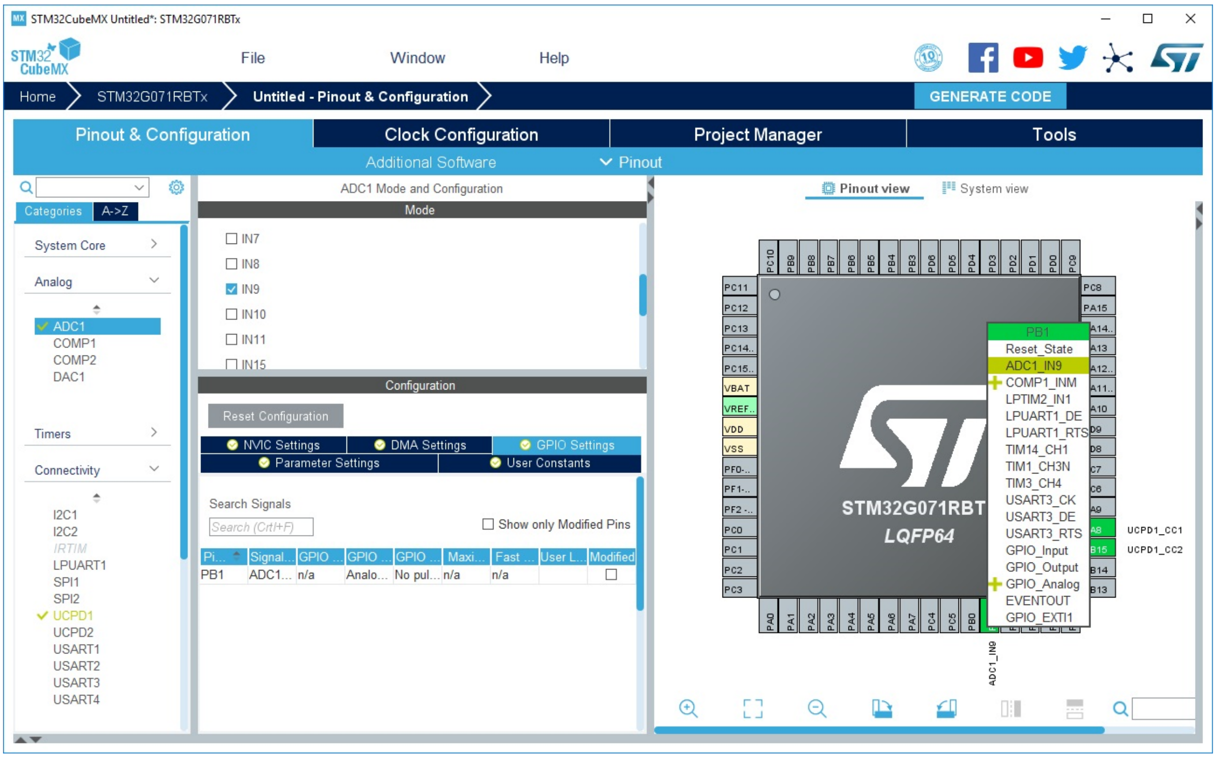
Figure 16. STM32G0 Nucleo-64 board (left) and TCPP01-M12 shield (right) schematics



In the analog category, select ADC1.

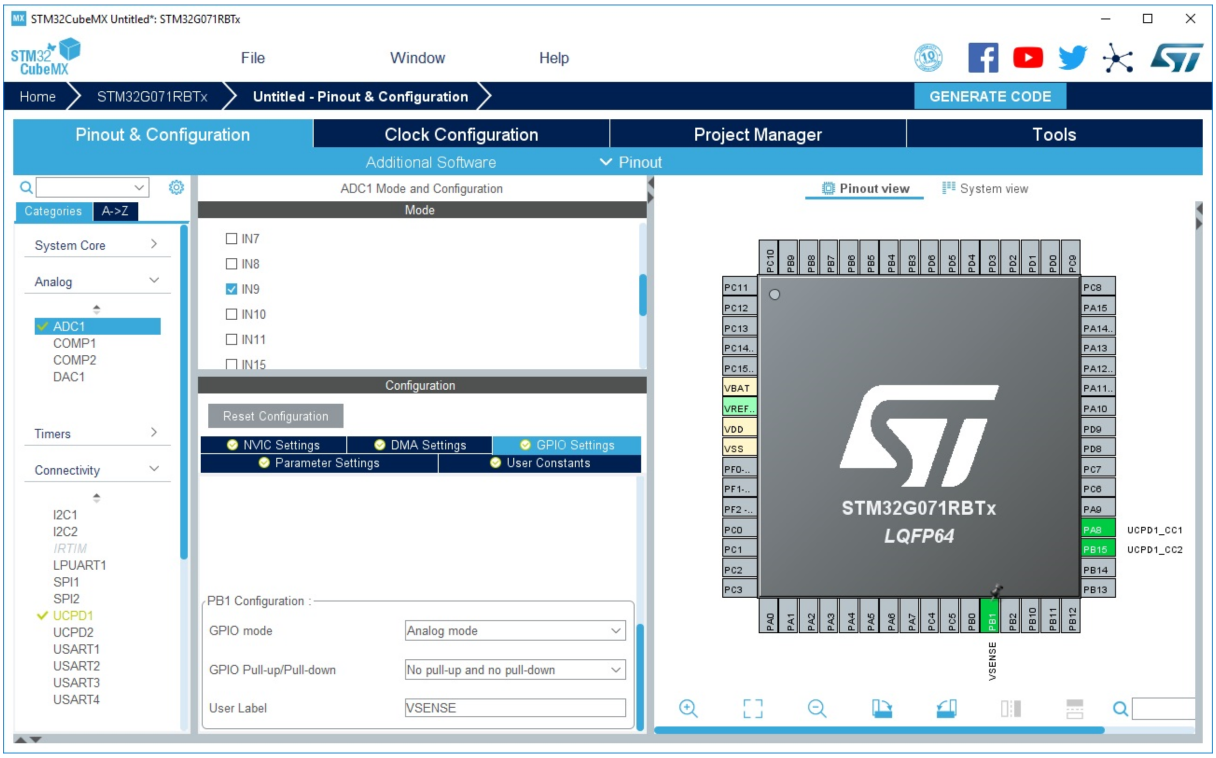
Clicking on the right side in STM32CubeMX, PB1 is connected to the ADC1_IN9 alternate function input. So select it in the mode part of the STM32CubeMX window, or select the ADC1_IN9 in the pinout view.

Figure 17. ADC configuration



Then in the *GPIO settings* tab, add the *User Label* VSENSE for this signal:

Figure 18. ADC GPIO settings

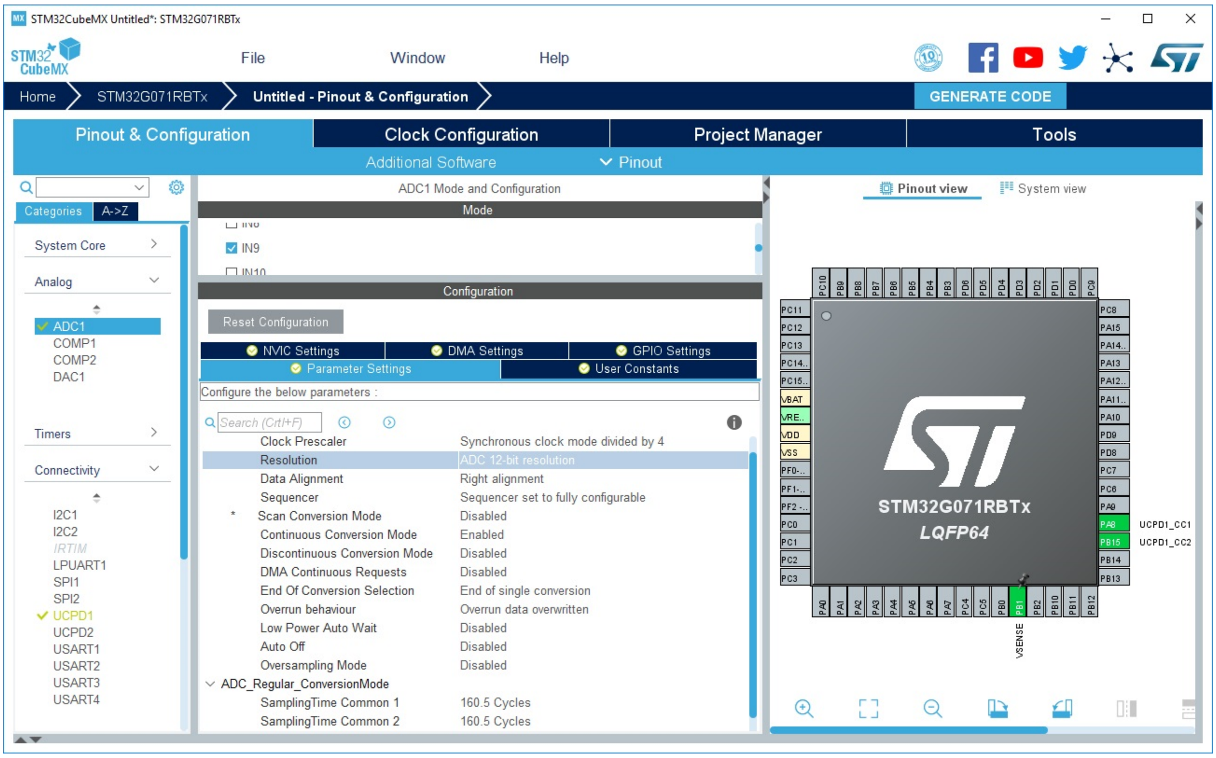


Select simple and basic settings for the ADC1:

- Clock Prescaler: Synchronous mode divided by 4
- Continuous conversion mode: Enabled
- Overrun behavior: Overrun data overwritten
- Sampling time: 160.5 Cycles

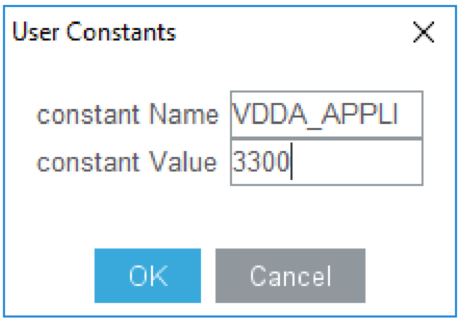
The sampling time must be adjusted with the impedance linked to the measure. In the case of the X-NUCLEO-USBPDM1 TCPP01-M12 shield, there are higher than 10 KΩ resistors, therefore a high number of cycles is preferred.

Figure 19. 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 20. ADC user constant



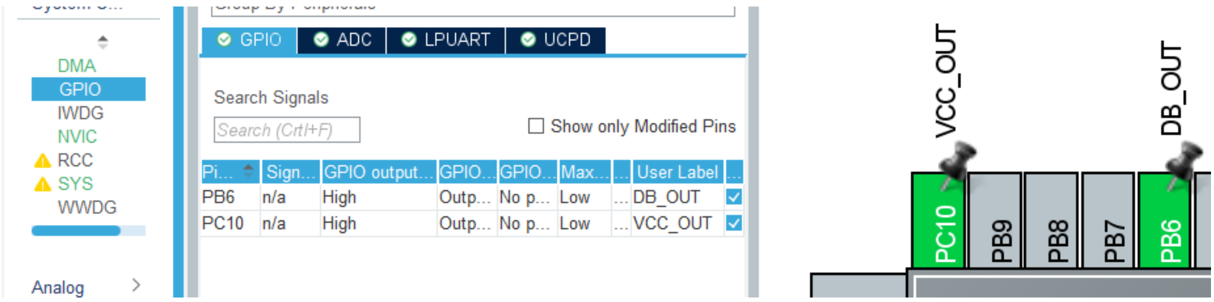
Note: On the STM32G4 Nucleo-64 board, the ADC IN15 signal must be used instead of IN9, because of the ADC mapping difference with STM32G0, and the rank sampling time can be set to 247.5 Cycles.

5.1.6 Additional GPIO settings

For the X-NUCLEO-SNK1M1 shield, two additional GPIO settings are needed (not in X-NUCLEO-USBPDM1 because the settings are forced by jumpers), as shown in Figure 21.

1. PB6 (DB_OUT for dead battery disabling) GPIO output to HIGH
2. PC10 (VCC_OUT pin to power on the TCPP01-M12) GPIO output to HIGH

Figure 21. Modify TCPP01-M12 controls

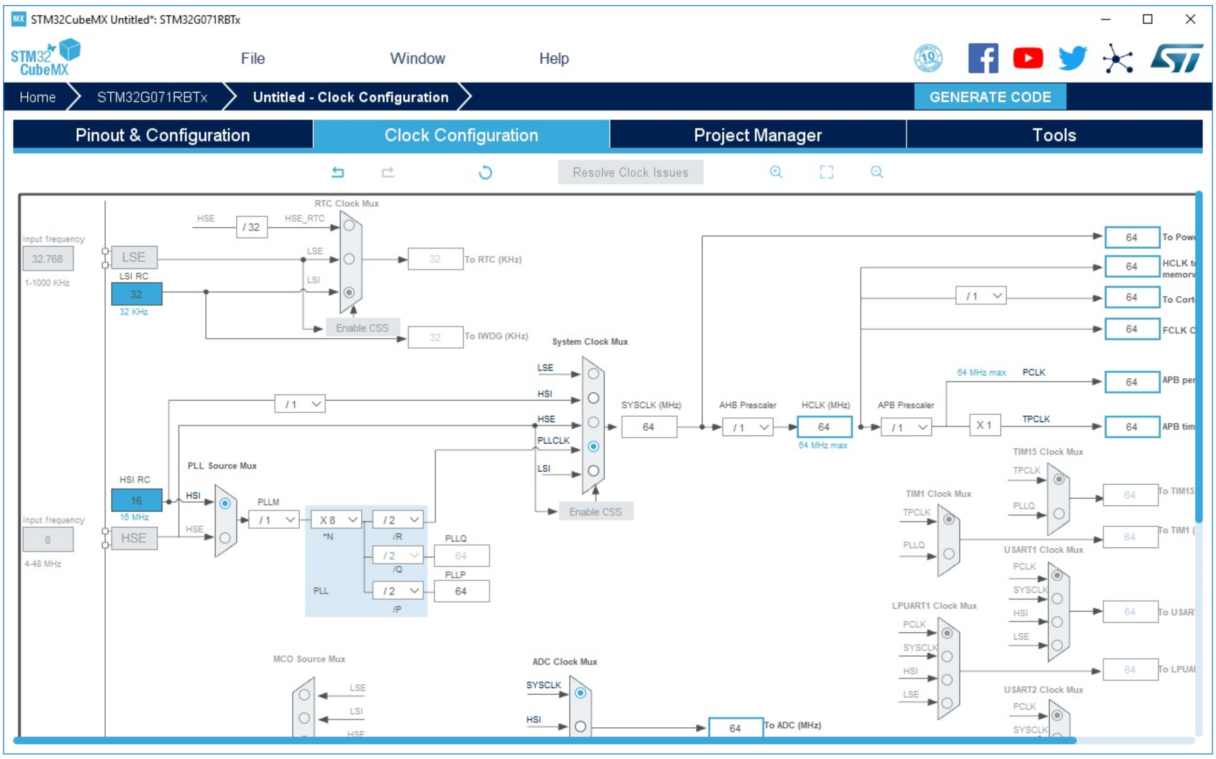


For a real application, these GPIO settings must be performed after the UCPD initialization.

5.1.7 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 22. Clock configuration



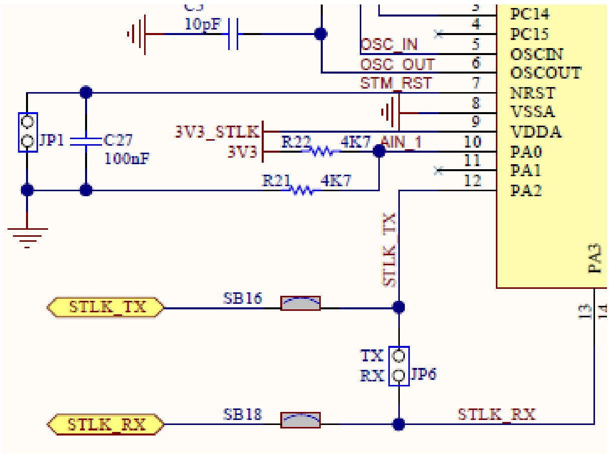
The mandatory settings for the simple USB-PD sink application are finished. The following part is highly recommended for debugging.

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.

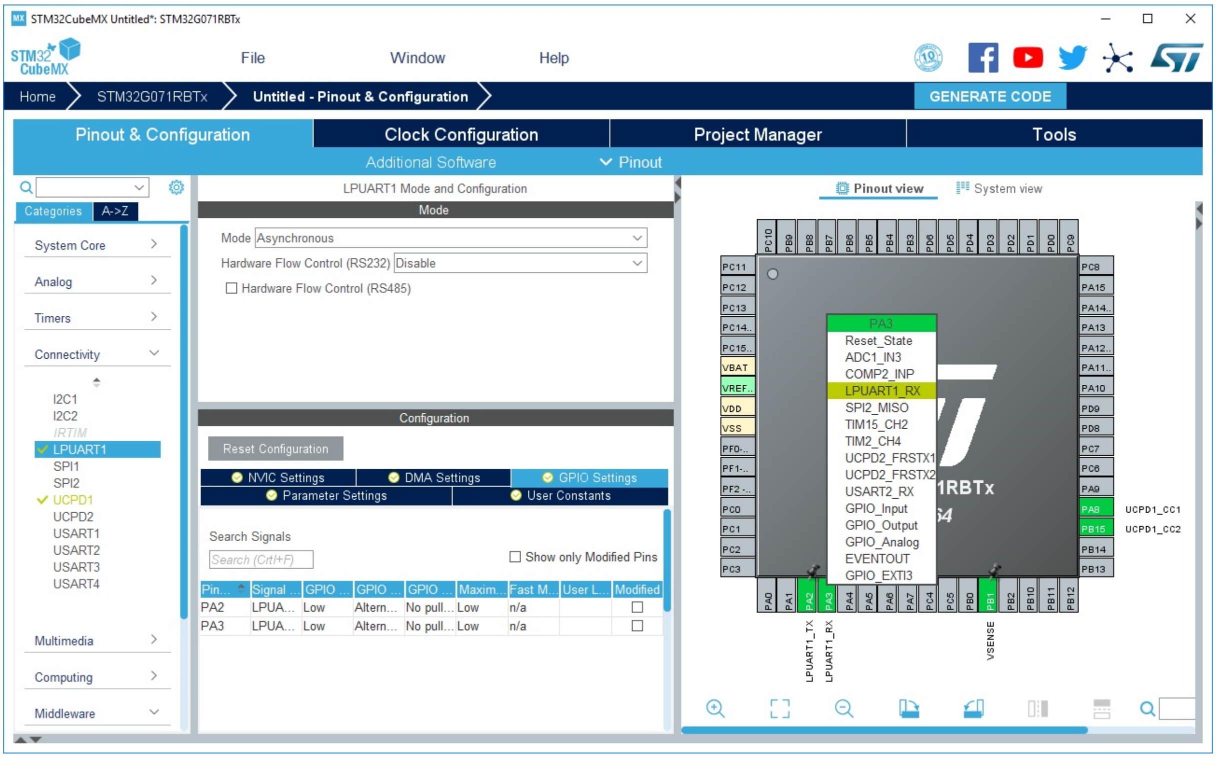
Figure 23. STM32G0 Nucleo-64 board STLK connection



This interface is activated in STM32CubeMX, in asynchronous mode.

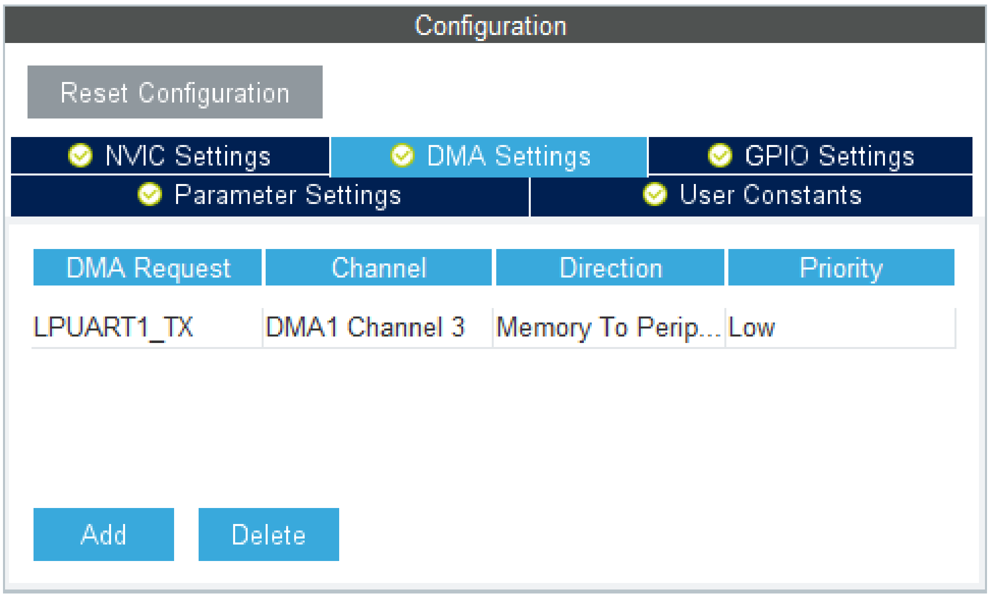
Important:
The default STM32CubeMX pins used by LPUART1 must be changed to match the STM32G0 Nucleo-64 hardware: PA2 for TX, and PA3 for RX.

Figure 24. LPUART1 activation and GPIO pin (TX and RX) update



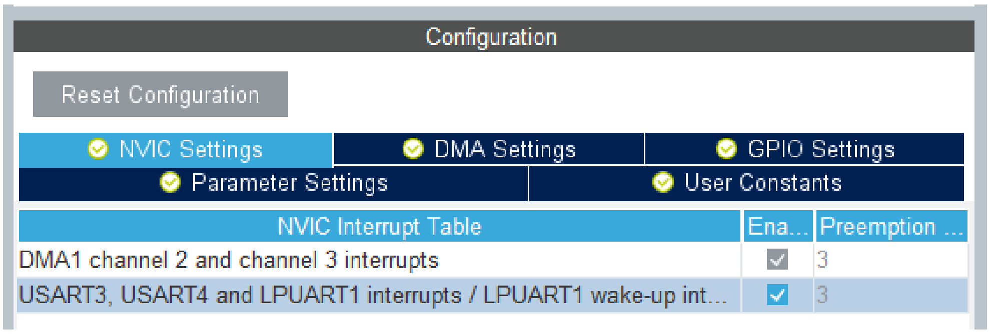
Then the DMA requests are activated for the TX path only: DMA1 channel 3.

Figure 25. DMA activation for LPUART



And the interrupt is activated:

Figure 26. DMA activation for LPUART1



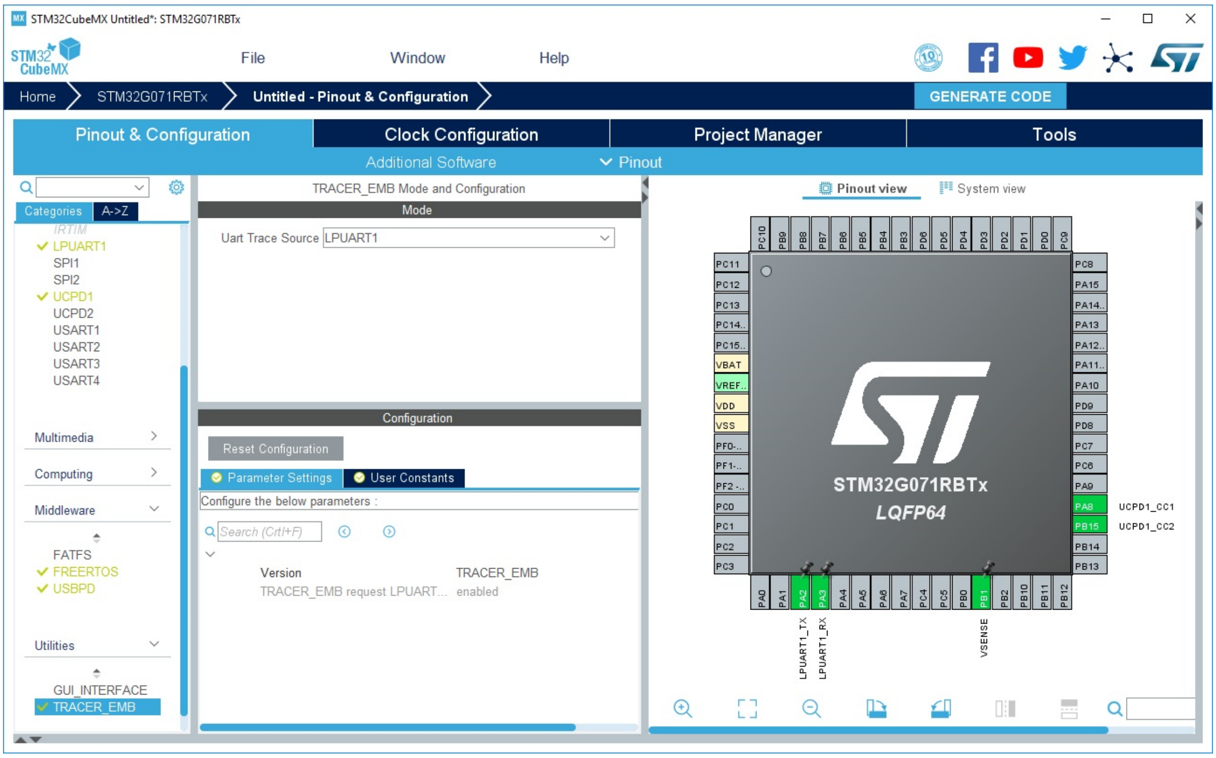
Note: If the STM32G4 Nucleo-64 board is used, USART1 must be used.

Note: The default UART configuration is used. The debug trace runs at 921600 bauds.

5.2.2 Activation of embedded tracer for debug

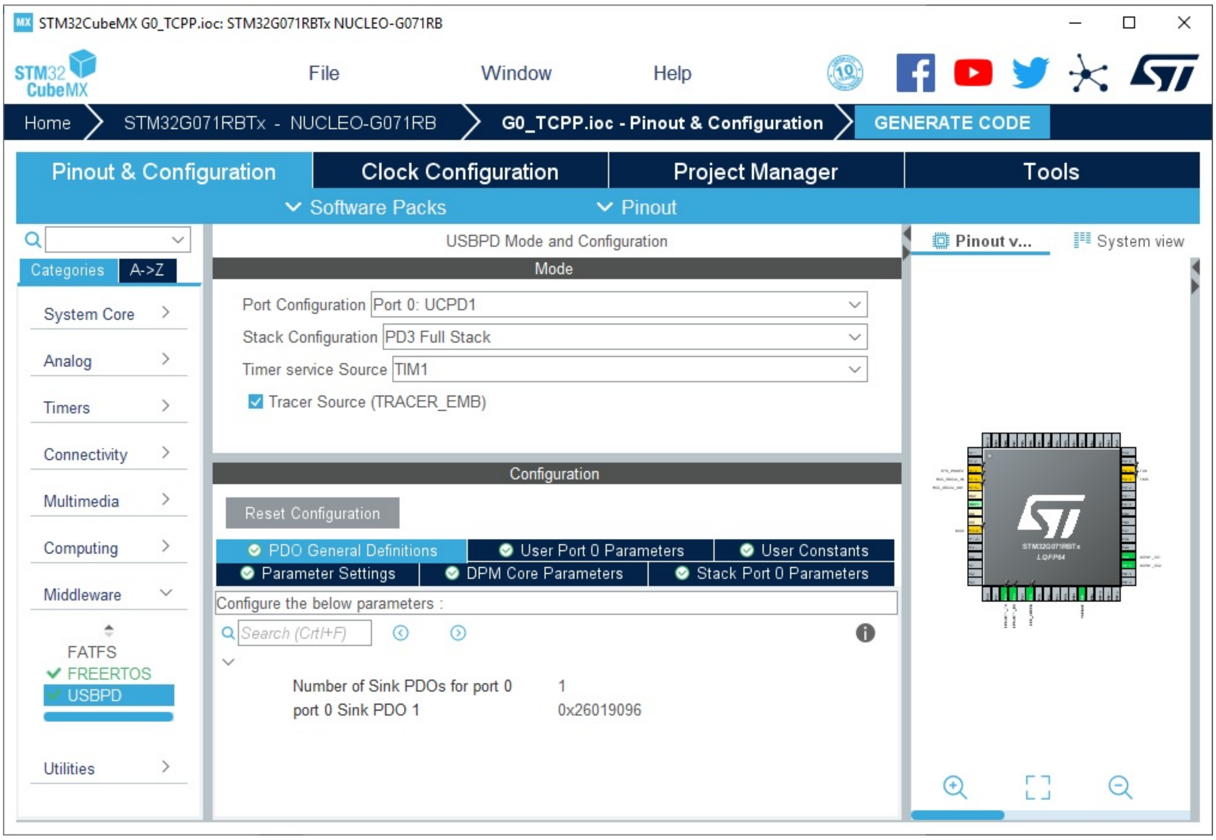
This is done in the utility category: *TRACER_EMB* is selected followed by the *LPUART1* in the UART trace source mode.

Figure 27. Activation of TRACER_EMB



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

Figure 28. Selection of USB-PD middleware TRACER_EMB source

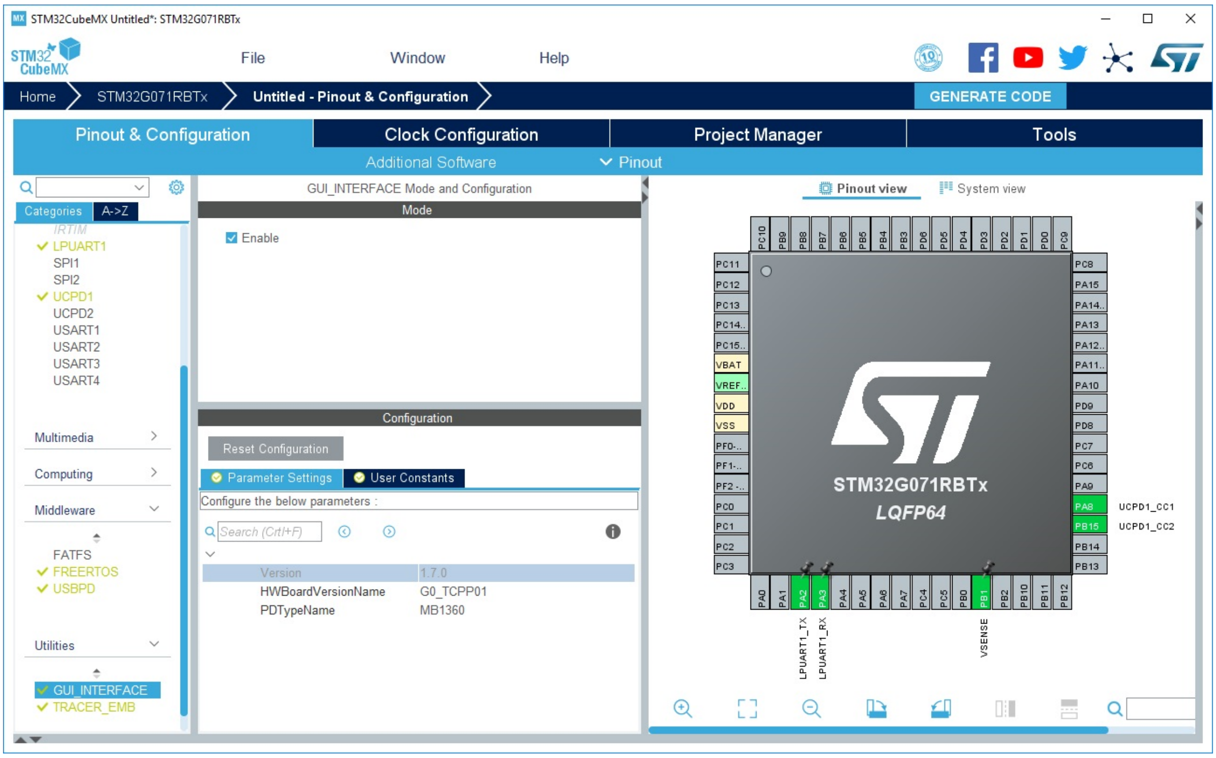


The firmware interactive stack responder can be activated if interaction with the USB-PD stack is needed, using the UCPD monitor tool STM32CubeMonUCPD.

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 29. Activation of GUI_INTERFACE



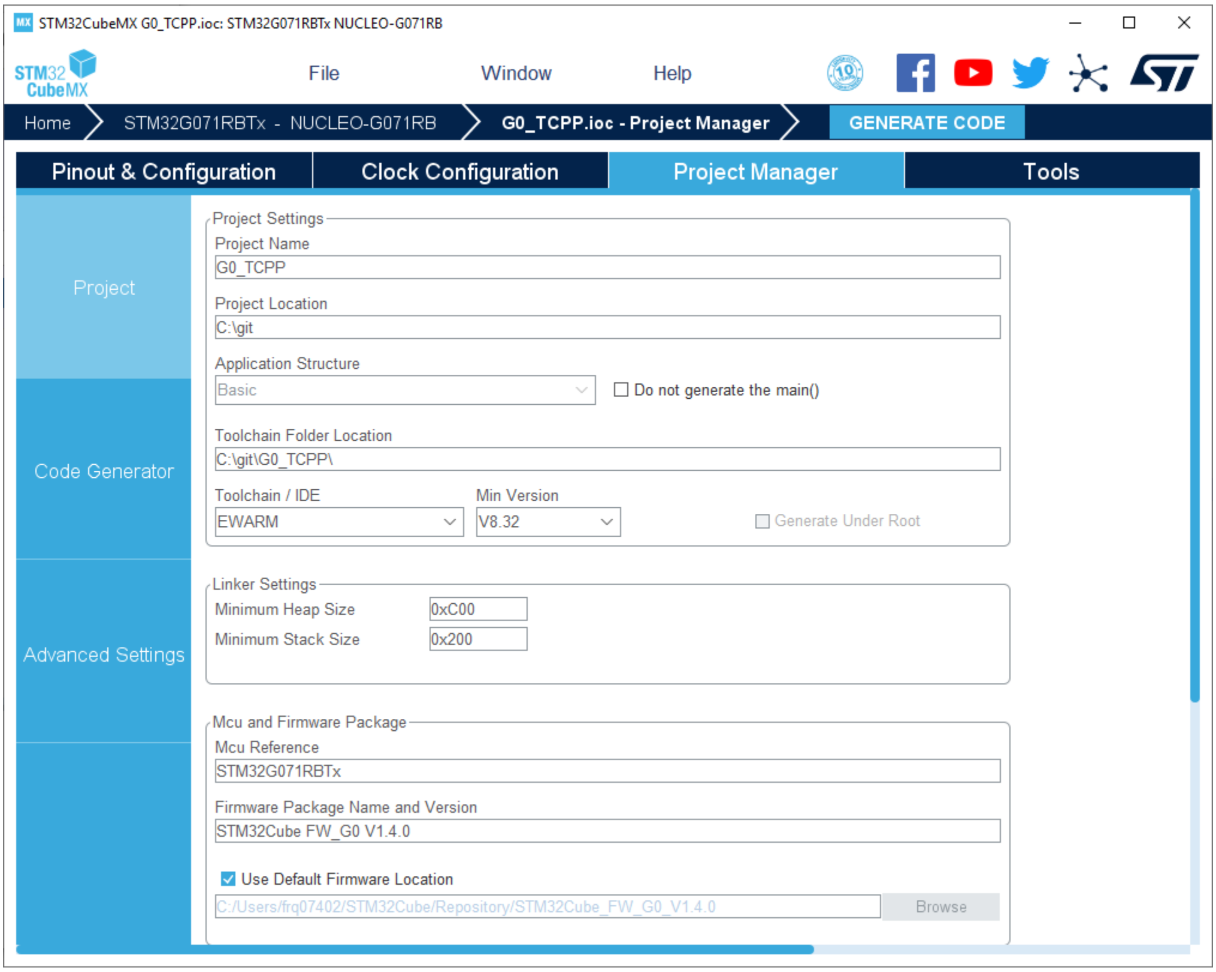
5.3 Update and save project configuration

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.

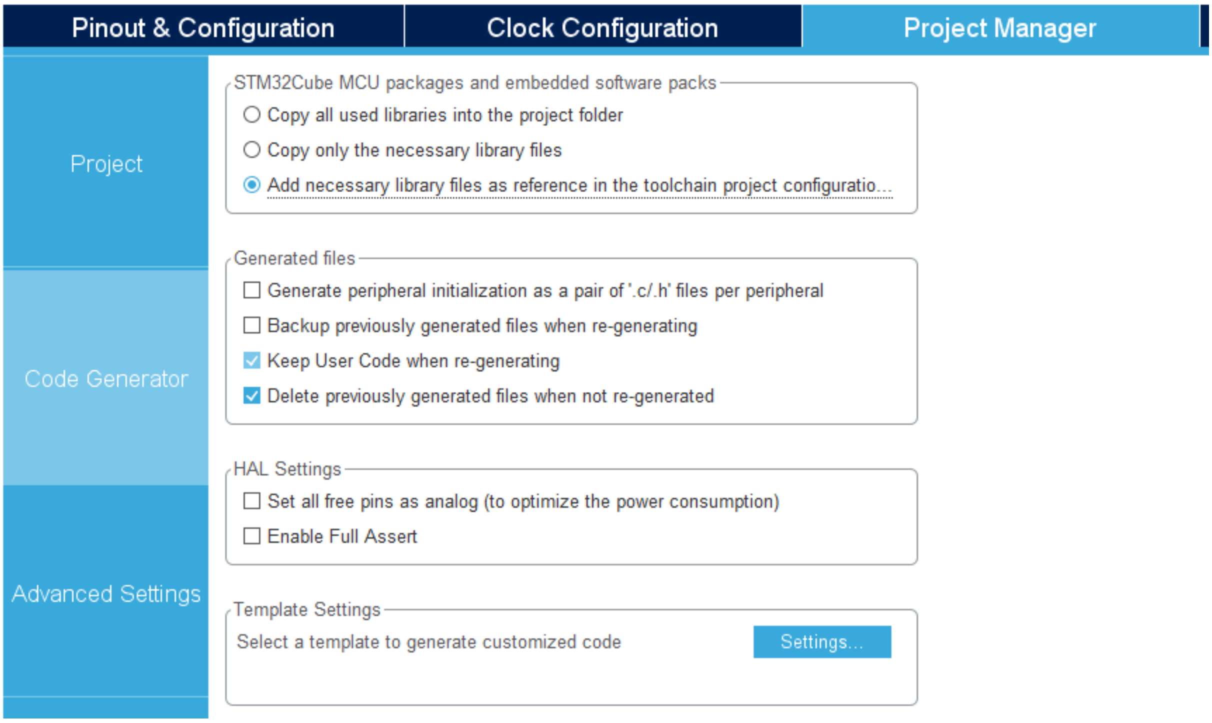
Configure the minimum stack size to `0xC00`. This is the first version, which can be tuned later, depending on the application needs.

Figure 30. Project manager settings



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

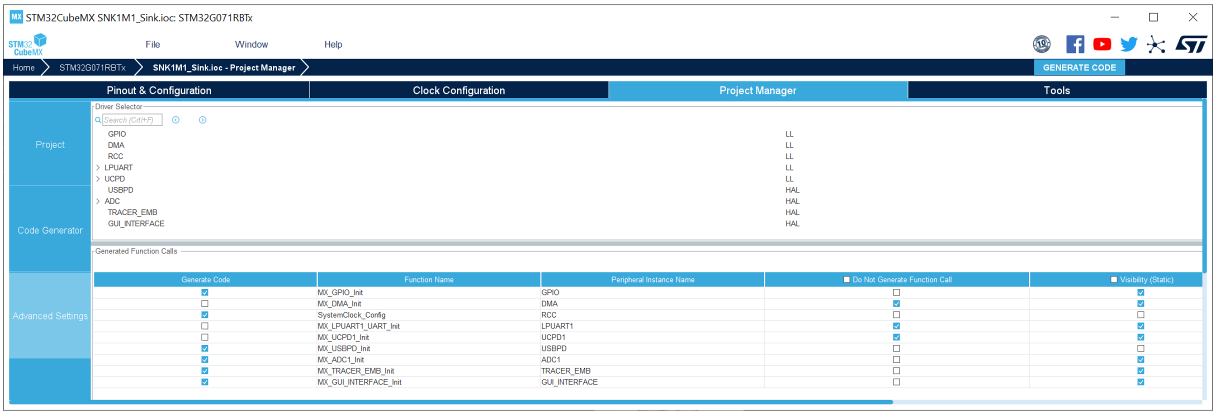
Figure 31. Code generator settings



Click on *Advanced Settings*

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

Figure 32. Project advanced setting



Work must be saved: menu file / save

5.4 Code generation

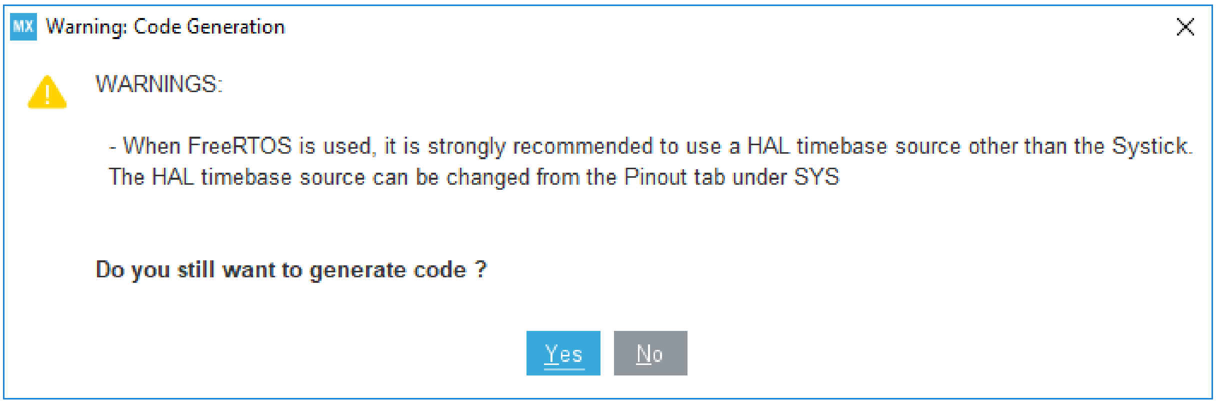
Click on generate code.

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.

Figure 33. Generation warning

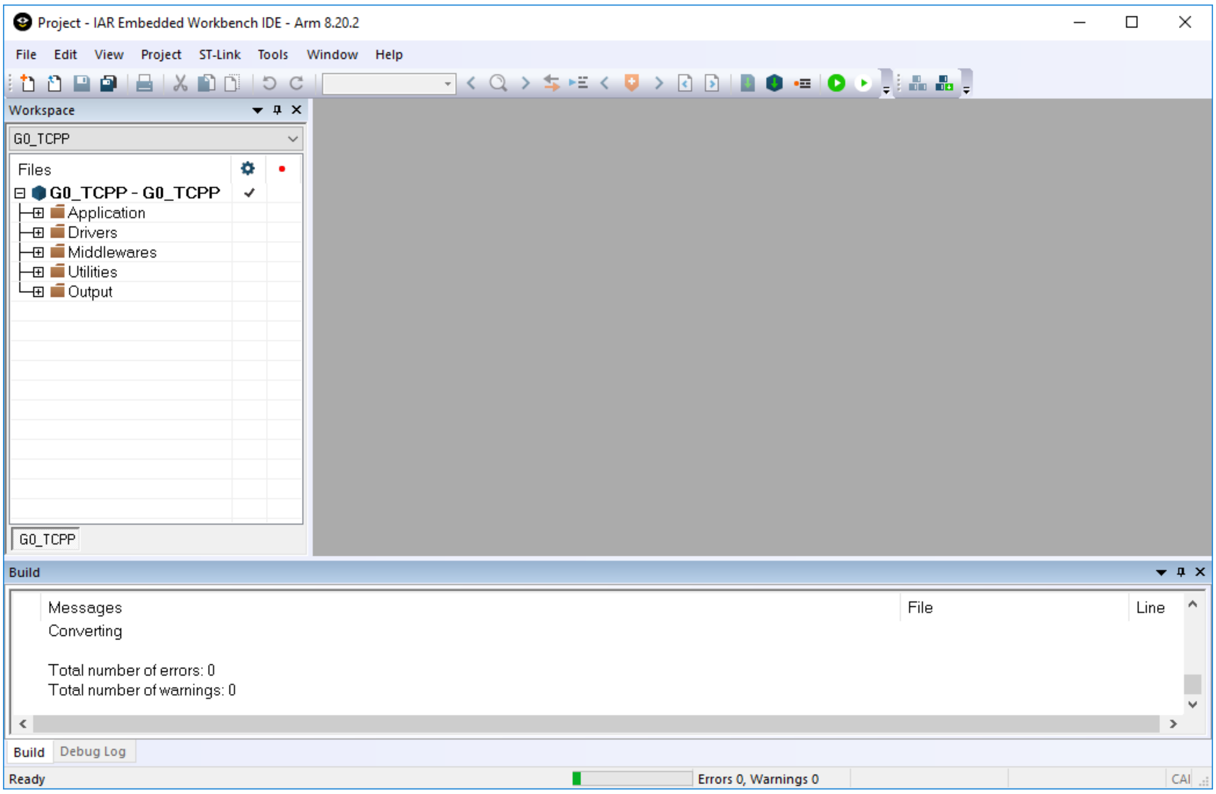


Then it is recommended to initialize Git to experiment with some code and be able to roll back to previous versions, as in classic software development.

5.5 Compilation of generated application

The compilation must be performed without error or warning.

Figure 34. **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.
- 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 V_{BUS} divider bridge.
- Complete *USBPD_DPM_SNK_EvaluateCapabilities* to answer one source capability message.
- TCPP01-M12 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 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 between `USER CODE ADC1_Init 2` tags:

```
…
 /* 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:

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

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

5.6.2 Modification in usbpd_dpm_user.c

To avoid a hard fault if the distant device asks for sink capabilities, some code must be added inside the *USBPD_DPM_GetDataInfo* function.

In the case, before the default add :

```
case USBPD_CORE_DATATYPE_SNK_PDO: /*!< Handling of port Sink PDO, requested by get sink
capa*/
  USBPD_PWR_IF_GetPortPDOs(PortNum, DataId, Ptr, Size);
  *Size *= 4;
  break;
```

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 are not yet managed by STM32CubeMX.

In the user code for *USBPD_DPM_SNK_EvaluateCapabilities* replace

```
DPM_USER_DEBUG_TRACE(PortNum, "ADVICE: update USBPD_DPM_SNK_EvaluateCapabilities");
```

with the following text:

```
…
/* 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 */
…
```

Note: The `ADVICE` keyword is used to indicate to the user that he may need to add his code to get a functional application.

5.6.3 Modification in usbpd_pwr_user.c

It is important to add this part to correctly read V_{BUS} provided by the ADC. The stack needs to know the V_{BUS} level all along the cable presence to determine the action to take. In the case of SINK, the detachment is done when V_{BUS} is below vSafe0V.

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

  /* Check if instance is valid       */
  int32_t ret = BSP_ERROR_NONE;

  if ((Instance >= USBPD_PWR_INSTANCES_NBR) || (NULL == pVoltage))
  {
    ret = BSP_ERROR_WRONG_PARAM;
    *pVoltage = 0;
  }
  else
  {
    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 (40.2K/200K) for VSENSE) */
    val *= 597;
    val /= 100;
    *pVoltage = val;
  }
  return ret;

/* USER CODE END BSP_USBPD_PWR_VBUSGetVoltage */
…
```

The calculation of the `val` variable depends on the voltage divider shown in Figure 15. On the X-NUCLEO-USBPDM1 shield, `Value` is multiplied by 5.97 (Divider R6/R7 40.2 kΩ/200 kΩ) for VSENSE.

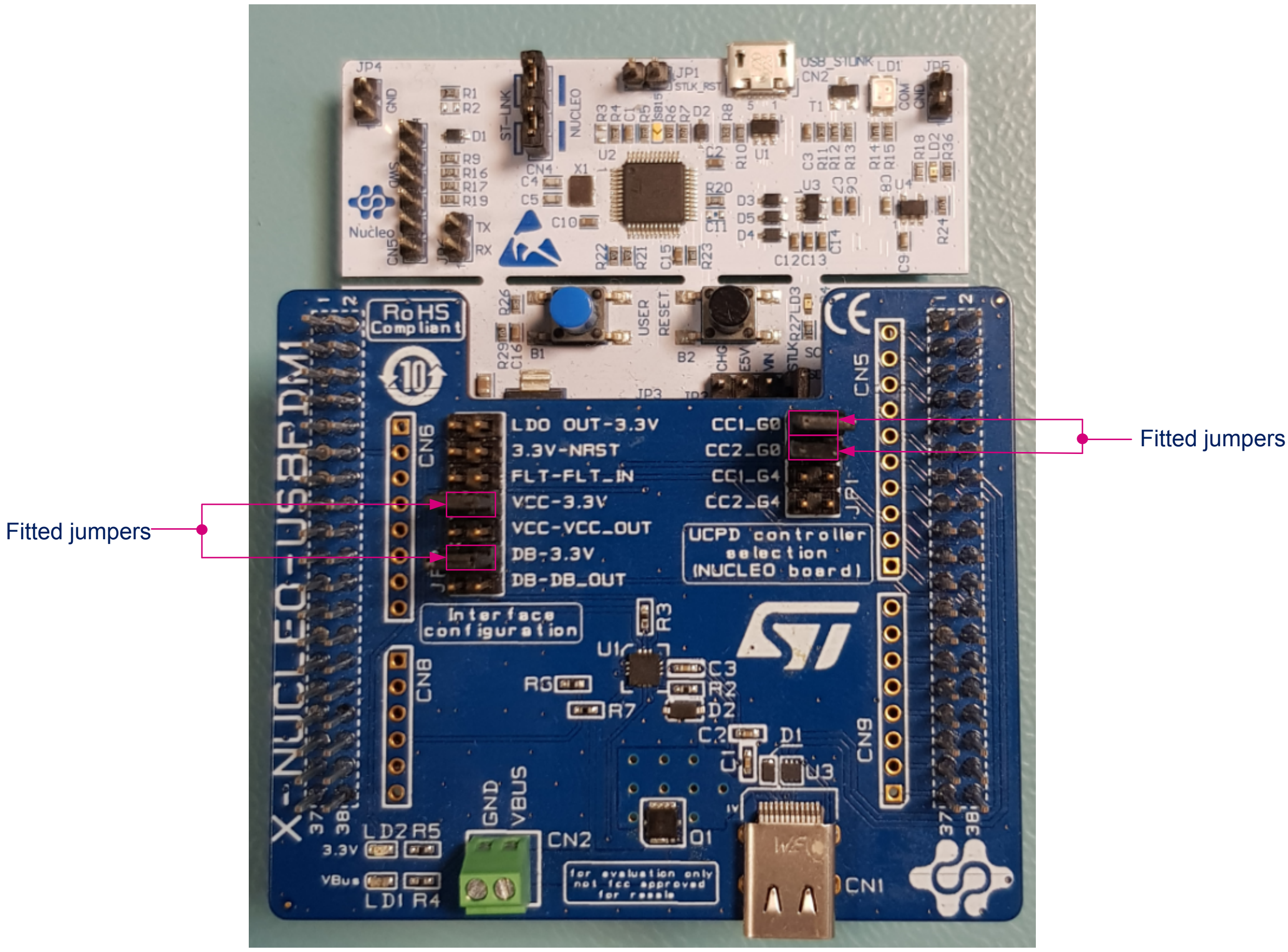
Note: In the X-CUBE-TCPP project, for the `Projects\NUCLEO-G071RB\Applications\USB_PD\USBPDM1_Sink_PPS` application, the `.extsettings` file is used to exercise the BSP shield (available in `Drivers/BSP/X-NUCLEO-USBPDM1` directory). Doing so, the weak functions in the generated code for the power parts are overloaded by the BSP files, and there is no need to manually modify the files.

5.7 Check jumpers

This is the last jumper setting check before the first power delivery contract.

5.7.1 X-NUCLEO-USBPDM1 jumpers

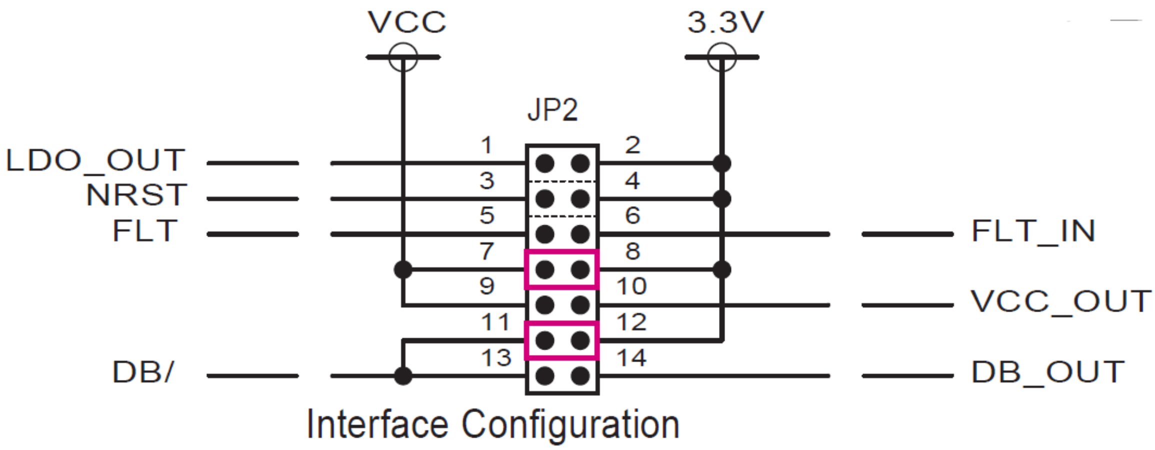
Figure 35. **X-NUCLEO-USBPDM1 shield picture**



Verify that the two JP1 jumpers located on the right to select the STM32G0 and STM32G4 configurations are inserted.

Then select the pins that are controlled by the MCU, using the left JP2 jumpers:

Figure 36. TCPP01-M12 jumper settings for X-NUCLEO-USBPDM1

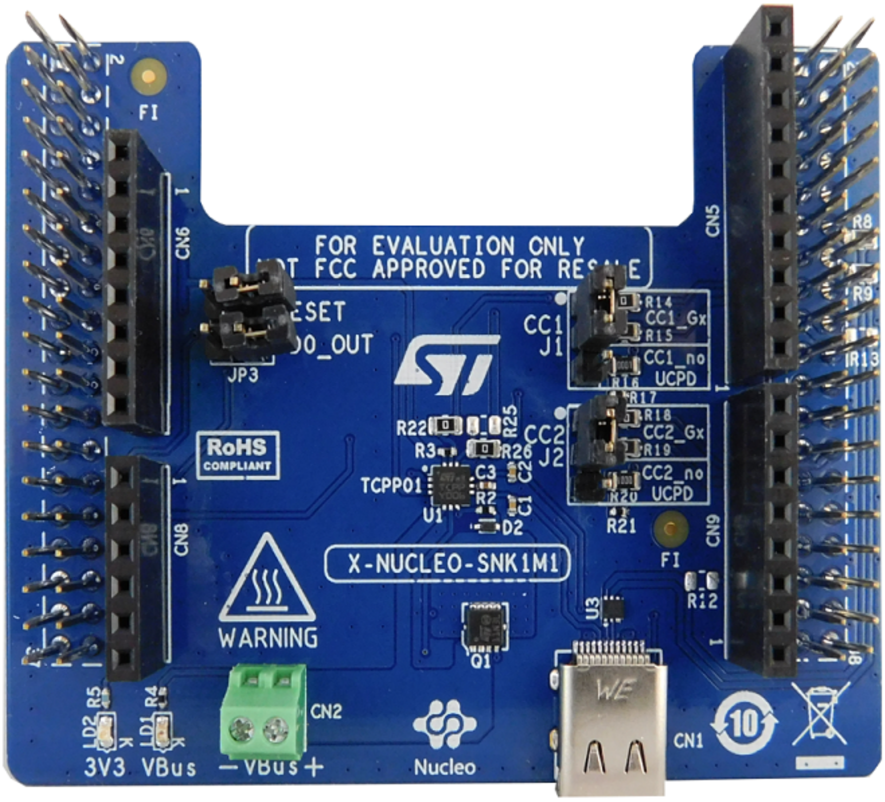


- Fault detection and hard reset are not managed in this demonstration.
- The power consumption is also not optimized. This is the reason why the TCPP01-M12 VCC is set to the fixed 3.3 V, instead of taking an MCU GPIO, so the JP2 jumper VCC-3.3V [7-8] is ON
- In the first step demonstration, the dead battery from the TCPP01-M12 is not used, so the JP2 jumper DB-3.3V [11-12] is ON.

5.7.2 X-NUCLEO-SNK1M1 jumpers

The JP3 jumpers are needed to select the STM32 power supply, from V_{BUS} or ST-LINK. For now, both jumpers need to be left open to allow download and debugging from ST-LINK. Refer to Figure 37.

Figure 37. X-NUCLEO-SNK1M1 top view

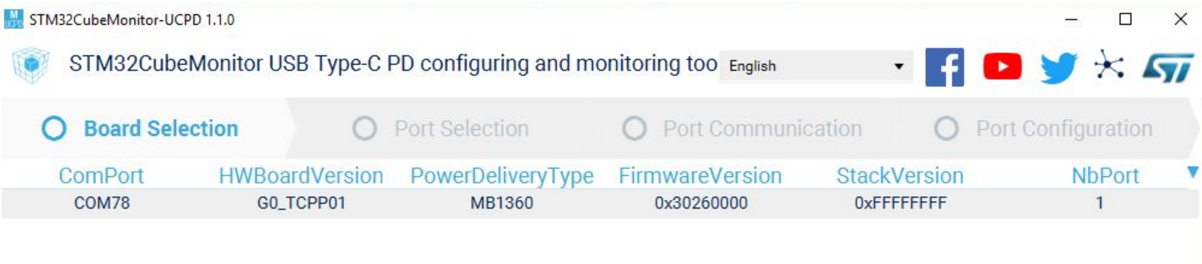


6 Establish the first explicit contract

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").

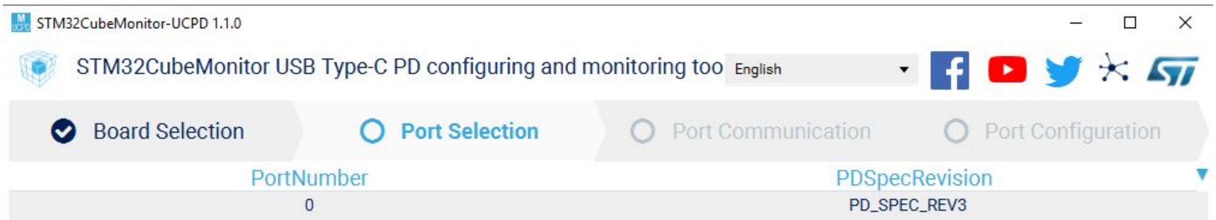
Figure 38. **Board selection**



Note: The ComPort may be different. It depends on the number of boards installed on the computer.

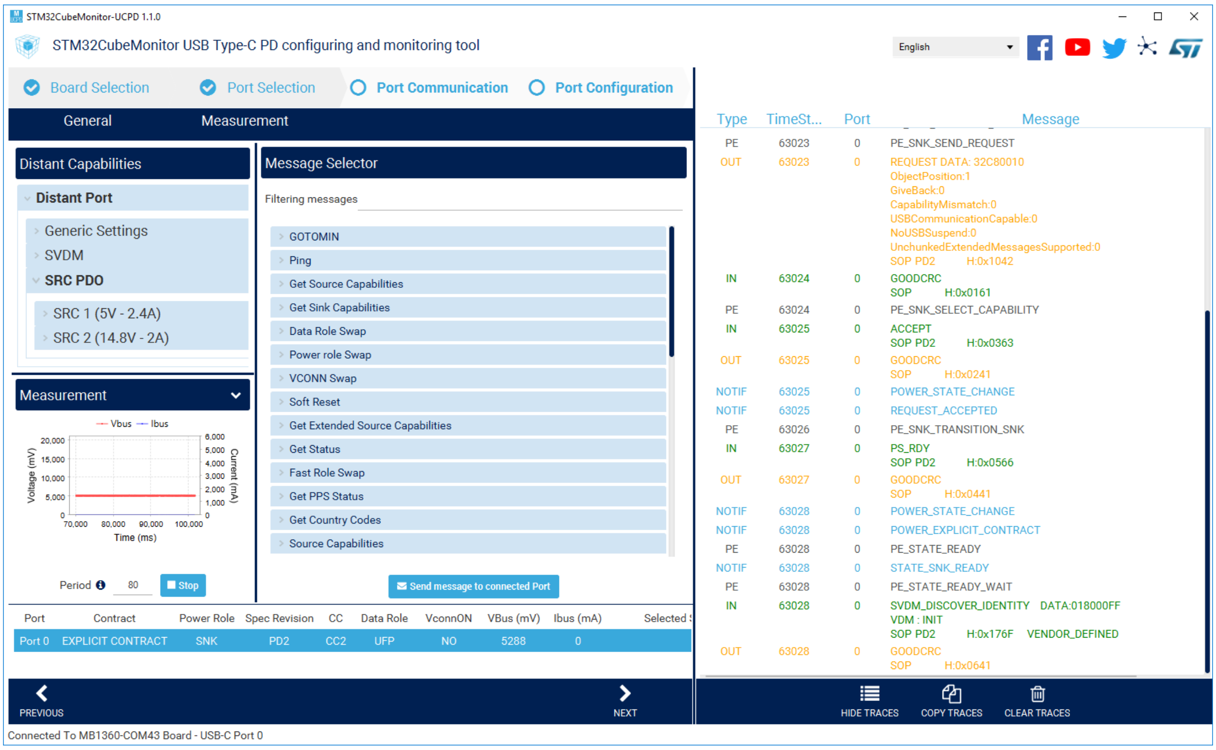
Then double click on the desired UCPD port, here Port 0, or select it and click "NEXT".

Figure 39. **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 into the USB Type-C® receptacle of the X-NUCLEO-USBPDM1 shield. The screen may look like Figure 40:

Figure 40. Explicit contract visible in UCPD monitor



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

Figure 40 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.

Note that this trace is very helpful for debugging and application development.

6.1 How to debug a bit deeper

6.1.1 livewatch variable setting

For more information in trace about CAD state machine, refer to Figure 41, where CAD appears in column *Type*.

Figure 41. Example of CAD debug information visible in the trace

Type	TimeSt...	Port	Message
CAD	906209	0	USBPD_CAD_STATE_SWITCH_TO_SRC
EVENT	906209	0	EVENT_DETACHED
NOTIF	906209	0	POWER_STATE_CHANGE
DEBUG	906209	0	HELP: update USBPD_DPM_SetDataInfo:7
DEBUG	906209	0	HELP: update USBPD_DPM_SetDataInfo:2
PE	906209	0	PE_SNK_STARTUP
CAD	906209	0	USBPD_CAD_STATE_DETACHED
DEBUG	914101	0	HELP: Update BSP_PWR_VBUSInit
CAD	914101	0	USBPD_CAD_STATE_ATTACHED_WAIT
CAD	914316	0	USBPD_CAD_STATE_ATTACHED0

Add *livewatch* on CAD_HW_Handles. This variable can be used to check the Type-C attachment or detachment.

Figure 42. cstate=1: detached

CAD_HW_Handles	<array>
[0]	<struct>
cc	1
CurrentHWconditi...	HW_Attachment
CAD_tDebounce...	0
CAD_tDebounce...	0
CAD_ErrorRecov...	0
CAD_ResistorUp...	0
cstate	USBPD_CAD_STATE_ATTACHED
CAD_Accessory_...	0
CAD_Accessory_...	0
reserved	0
pstate	USBPD_CAD_STATE_ATTACHED
reserved2	0
CAD_tDebounce...	65256
CAD_PtrStateMa...	CAD_StateMachine_SNK (0x8006B99)

If the CC lines level changes are invisible, check that the TCPP is powered on and 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 usbpd_cad_hw_if.c there must be:*

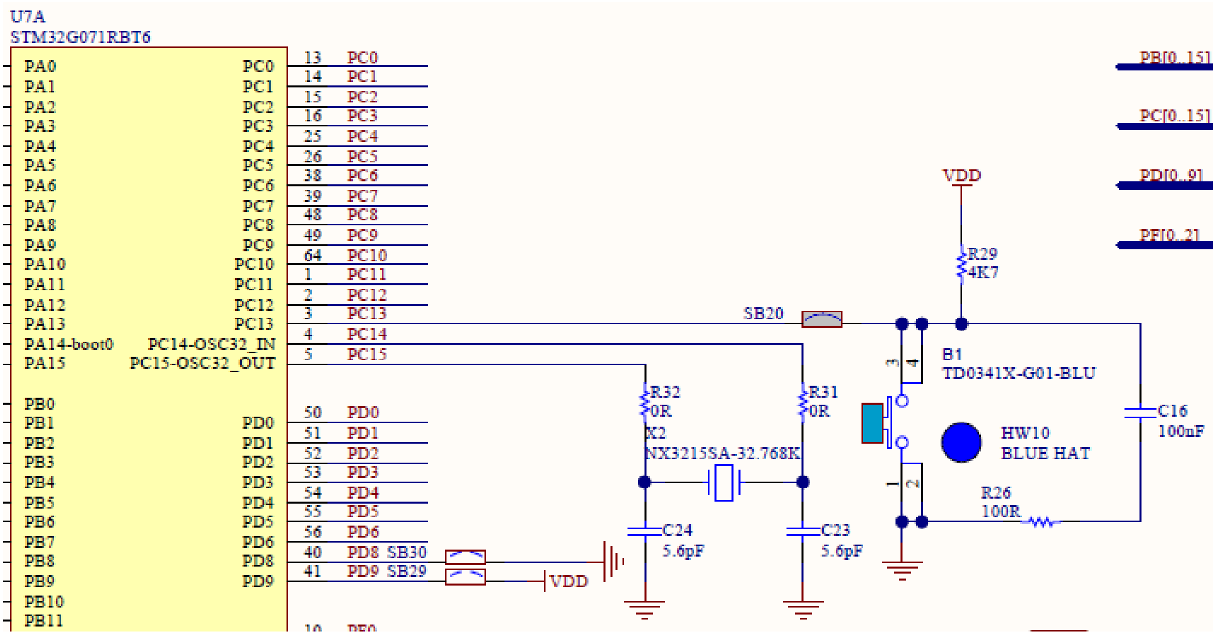
```
LL_GPIO_SetPinMode(GPIOB, LL_GPIO_PIN_4, LL_GPIO_MODE_ANALOG);
```

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.

6.1.2 User button

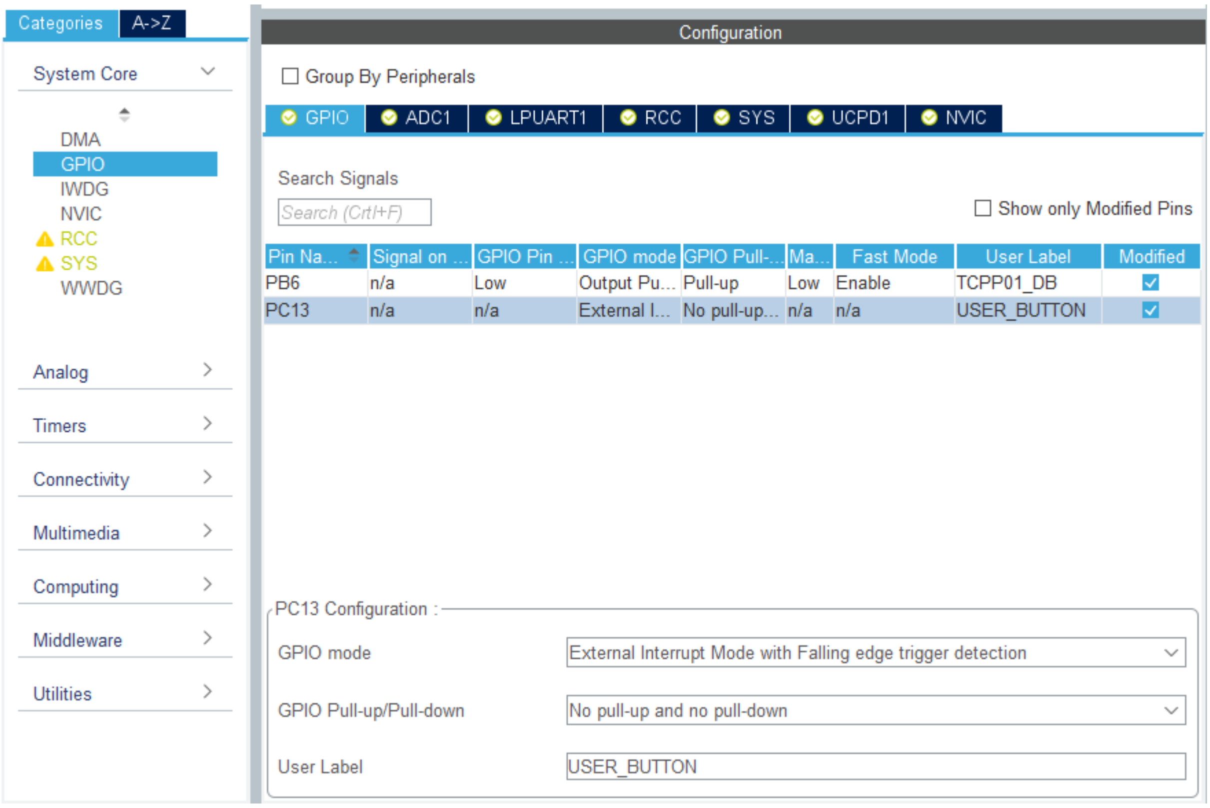
For further debugging, the V_{BUS} measured value can be printed in the trace, using the user button:

Figure 43. User button on STM32G0 Nucleo-64 board schematics



Add the button from STM32CubeMX as described in Figure 44.

Figure 44. Add the user button in STM32CubeMX



Add in src/main.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:

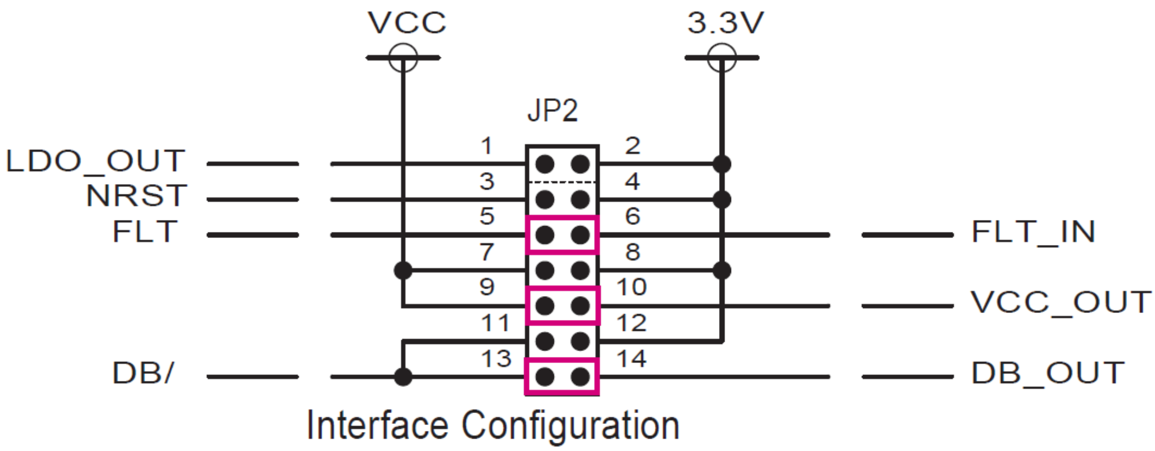
```
/**
  * @brief  This function handles the external line 4_15 interrupt request.
  * @retval None
  */
void EXTI4_15_IRQHandler(void)
{
  HAL_GPIO_EXTI_IRQHandler(USER_BUTTON_PIN);
}
```

7 Next steps

X-NUCLEO-USBPDM1 can be better controlled by driving its dead battery pin and its power like it is described in Section 5.1.6 for X-NUCLEO-SNK1M1.

If these pins need to be controlled by the application, in case the X-NUCLEO-USBPDM1 shield is used, the jumper positions must be adapted. The potential faults must also be read by setting the JP2 jumper in [5-6] position.

Figure 45. X-NUCLEO-USBPDM1 shield jumpers position when application manages FLT, DB, and the TCPP01-M12 VCC



The user application also needs to be done to react upon the TCPP01-M12 fault detection (Over-temperature, over-voltage).

8 Conclusion

This demonstration is only the first step to a power delivery application.

This quickly developed application is not optimized from the low-power point of view.

The USB PD application performed here is the bare minimum. No software code 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

Date	Version	Changes
9-Jan-2020	1	Initial release
30-Apr-2020	2	Added specific technical data throughout the document
27-Nov-2020	3	Updated: Figures: • *Figure 7. FreeRTOS™ configuration* • *Figure 11. Detailed stack configuration* • *Figure 24. Selection of USB-PD middleware TRACER_EMB source* • *Figure 26. Project manager settings* • *Figure 34. Port selection* Code blocks in: • *Section 5.6.2 Modification in stm32g0xx_it.c* • *Section 5.6.3 Modification in usbpd_dpm_user.c* • *Section 5.6.4 Modification in usbpd_pwr_user.c*
3-May-2021	4	Added: • X-NUCLEO-SNK1M1 shield • Links in *Introduction* • *Figure 2. STM32G0 Nucleo-64 board equipped with X-NUCLEO-SNK1M1 shield* • *Figure 3. X-CUBE-TCPP block diagram architecture* • *Figure 20. Modify TCPP01-M12 controls* • TCPP in *Acronym definitions* • *Section 5.1.6 Additional GPIO settings* • *Section 5.7.2 X-NUCLEO-SNK1M1 jumpers* Updated: • UM2668 replaced by UM2773 and *YouTube video* link in *Section 3 Reference documents* • *Figure 14. TCPP01-M12 shield voltage divider* • *Figure 15. STM32G0 Nucleo-64 board (left) and TCPP01-M12 shield (right) schematics* • *Figure 31. Project advanced setting* • `usbpd_cad_hw_if.c` code and *Figure 41* in *Section 6.1.1 livewatch variable setting* • *Section 7* Removed: • Former 5.6.2 section on *Modification in stm32g0xx_it.c*
5-Oct-2021	5	Added: • Figure 9 showing *eTaskGetState* new setting Updated: • *TOTAL_HEAP_SIZE* value • `usbpd_dpm_user.c` and `usbpd_pwr_user.c` files in Modification in usbpd_dpm_user.c and Modification in usbpd_pwr_user.c chapters

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 11
5.1.5 ADC configuration for V_{BUS} reading 14
5.1.6 Additional GPIO settings 17
5.1.7 Clock check 17
5.2 Additional recommended optional debugging 18
5.2.1 UART configuration for debug 18
5.2.2 Activation of embedded tracer for debug 20
5.2.3 Activation of UCPD monitor firmware responder 22
5.3 Update and save project configuration 23
5.4 Code generation 25
5.5 Compilation of generated application 26
5.6 Complete USB-PD application. 26
5.6.1 Modification in main.c 27
5.6.2 Modification in usbpd_dpm_user.c 27
5.6.3 Modification in usbpd_pwr_user.c 28
5.7 Check jumpers. 29
5.7.1 X-NUCLEO-USBPDM1 jumpers 29
5.7.2 X-NUCLEO-SNK1M1 jumpers 31
6 **Establish the first explicit contract** 32
6.1 How to debug a bit deeper 34
6.1.1 livewatch variable setting 34
6.1.2 User button 35
7 **Next steps** 37
8 **Conclusion** 38
Revision history 39
List of tables 42

List of figures 43

List of tables

Table 1. Acronym definitions . . . 3
Table 2. Document revision history . . . 39

List of figures

Figure 1. STM32G0 Nucleo-64 board equipped with X-NUCLEO-USBPDM1 shield 1
Figure 2. STM32G0 Nucleo-64 board equipped with X-NUCLEO-SNK1M1 shield 1
Figure 3. X-CUBE-TCPP block diagram architecture 2
Figure 4. Start STM32CubeMX 6
Figure 5. Select the STM32G0 MCU 7
Figure 6. UCPD peripheral basic configuration 8
Figure 7. UCPD peripheral DMA configuration 8
Figure 8. UCPD peripheral IT activation 9
Figure 9. FreeRTOS™ parameters 10
Figure 10. FreeRTOS™ configuration 10
Figure 11. USB-PD middleware configuration 11
Figure 12. Specification detail (table 6-14 in Universal Serial Bus Power Delivery Specification) 12
Figure 13. Detailed PDO decoding 12
Figure 14. Detailed stack configuration 13
Figure 15. TCPP01-M12 shield voltage divider 14
Figure 16. STM32G0 Nucleo-64 board (left) and TCPP01-M12 shield (right) schematics 14
Figure 17. ADC configuration 15
Figure 18. ADC GPIO settings 15
Figure 19. ADC parameters settings 16
Figure 20. ADC user constant 16
Figure 21. Modify TCPP01-M12 controls 17
Figure 22. Clock configuration 17
Figure 23. STM32G0 Nucleo-64 board STLK connection 18
Figure 24. LPUART1 activation and GPIO pin (TX and RX) update 18
Figure 25. DMA activation for LPUART 19
Figure 26. DMA activation for LPUART1 19
Figure 27. Activation of TRACER_EMB 20
Figure 28. Selection of USB-PD middleware TRACER_EMB source 21
Figure 29. Activation of GUI_INTERFACE 22
Figure 30. Project manager settings 23
Figure 31. Code generator settings 24
Figure 32. Project advanced setting 24
Figure 33. Generation warning 25
Figure 34. First compilation 26
Figure 35. X-NUCLEO-USBPDM1 shield picture 29
Figure 36. TCPP01-M12 jumper settings for X-NUCLEO-USBPDM1 30
Figure 37. X-NUCLEO-SNK1M1 top view 31
Figure 38. Board selection 32
Figure 39. Port selection 32
Figure 40. Explicit contract visible in UCPD monitor 33
Figure 41. Example of CAD debug information visible in the trace 34
Figure 42. cstate=1: detached 34
Figure 43. User button on STM32G0 Nucleo-64 board schematics 35
Figure 44. Add the user button in STM32CubeMX 35
Figure 45. X-NUCLEO-USBPDM1 shield jumpers position when application manages FLT, DB, and the TCPP01-M12 VCC 37