Managing dynamic input power with the STUSB4500 and the STM32F072RB

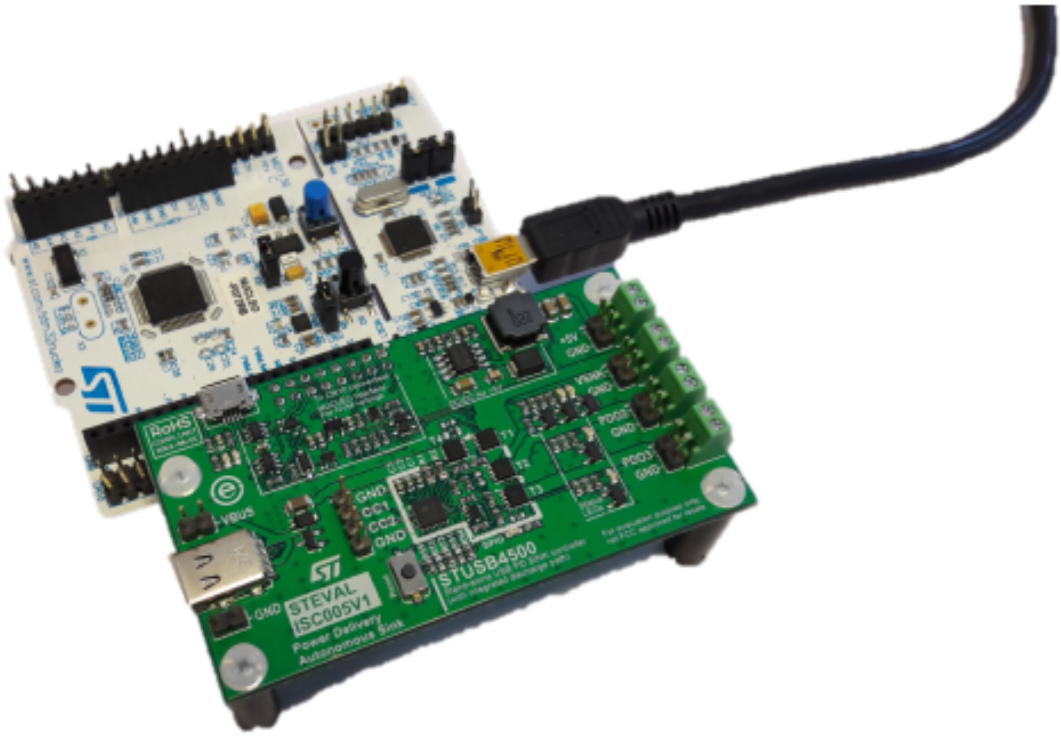
Introduction

This document describes how to manage the dynamic input power by applying software on top of the STUSB4500 from the application processor (in this document, an STM32F072RB). Open source software is available to speed-up end-application SW developments. The code is available as an example only.

Table 1. Minimal configuration

NUCLEO-F072RB	STM32 Nucleo-64 development board with AMR Cortex M0
STEVAL-ISC005V1	STUSB4500 evaluation board
STSW-STUSB003	Software library including the STUSB4500 hardware abstraction layers, drivers and Code example
IAR 8.x	C code compiler

Figure 1. NUCLEO-F072RB+STEVAL_ISC005V1



1 Hardware configuration

The STEVAL_ISC005V1 must be plugged to the NUCLEO-F072RB board according to the schematic below:

Figure 2. NUCLEO-F072RB connections

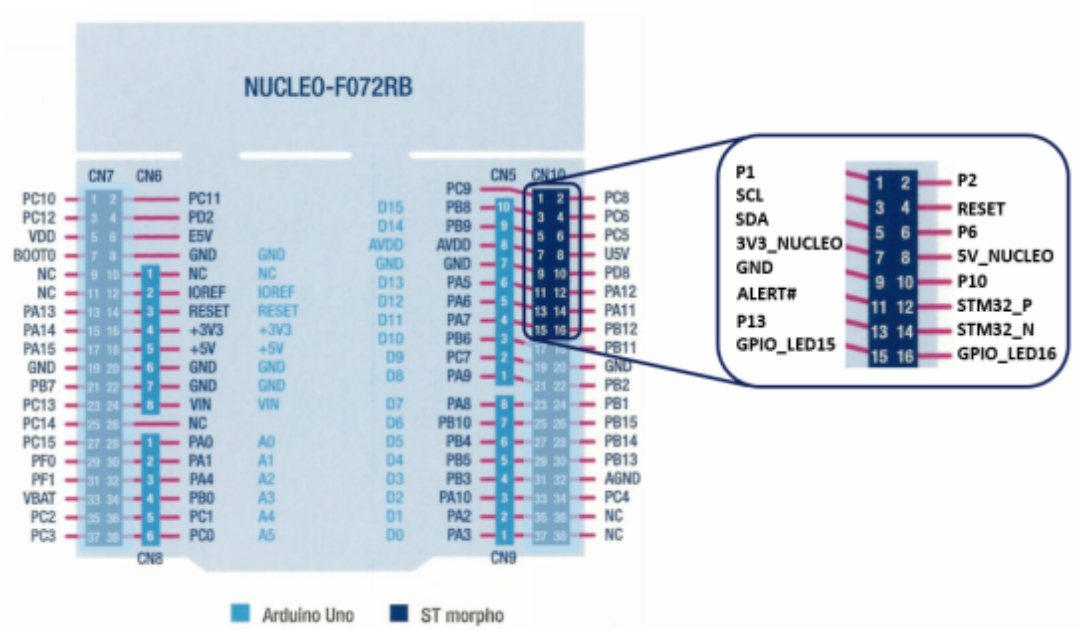
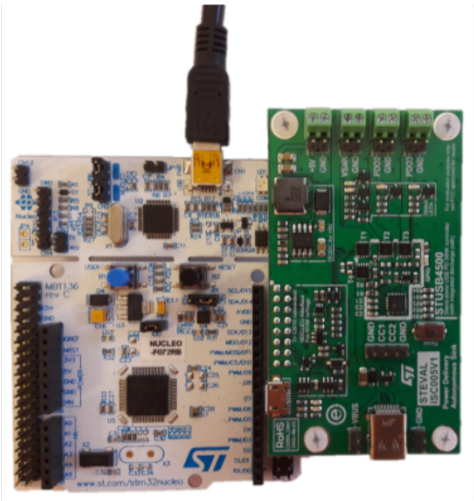


Figure 3. STEVAL_ISC005V1 plugged to the NUCLEO-F072RB



2 Software configuration

2.1 Main files

The library is composed of the following files:

- USB_PD_defines_STUSB-GEN1S.h
- USB_PD_core.h
- USB_PD_core.c
- Main.c

USB_PD_defines_STUSB-GEN1S.h: contains the register definition from the STUSB4500.
USB_PD_core.h: contains main structures to handle the STUSB4500 configuration.
USB_PD_core.c: contains generic functions to build applications.
Main.c: illustrates how to use the functions and build applications.

3 Low level drivers, hardware abstraction functions

Here is the list of available functions developed to show the STUSB4500 software capabilities:

```
void usb_pd_init(uint8_t Port);
void ALARM_MANAGEMENT(uint8_t Port);


void HW_Reset_state(uint8_t Port);
void SW_reset_by_Reg(uint8_t Port);
void Read_SNK_PDO(uint8_t Port);
void Print_SNK_PDO(uint8_t Port);
void Print_PDO_FROM_SRC(uint8_t Port);
void Read_RDO(uint8_t Port);
void Print_RDO(uint8_t Port);
void Update_PDO1 (Port, Current);
void Update_PDO(uint8_t Port, uint8_t PDO_Number, int Voltage, int Current) ;
void Update_Valid_PDO_Number(uint8_t Port, uint8_t Number_PDO);
void Send_Soft_reset_Message(uint8_t Usb_Port);
int  Request_SRC_PDO_NUMBER(uint8_t Usb_Port, uint8_t SRC_PDO_position);
void Print_Type_C_Only_Status(uint8_t Usb_Port);
int Get_Device_STATUS(uint8_t Usb_Port);
```

3.1 usb_pd_init(Port)

This function clears all interrupts and unmasks the useful interrupts.

3.2 ALARM_MANAGEMENT(Port)

This is the device interrupt handler.

3.3 HW_Reset_state(Port)

This function asserts and de-asserts the STUSB4500 hardware reset pin. After the reset, the STUSB4500 behaves according to non-volatile-memory default settings.

3.4 SW_Reset_by_Reg(Port)

This function resets the STUSB4500 Type-C and USB PD state machines. It also clears any ALERT. By initializing Type-C pull-down termination, it forces the electrical USB type-C disconnection (both on source and sink sides).

3.5 Send_Soft_reset_Message(Port)

This function sends an official SOFT reset message to the SOURCE through USB PD protocol.

3.6 Read_SNK_PDO(Port);

This function reads the SINK_PDO registers from the STUSB4500 and loads it in a dedicated structure.

3.7 Print_SNK_PDO(Port);

This function calls the Read_SNK_PDO function and prints the PDO values to the serial interface.

3.8 Print_PDO_FROM_SRC (Port);

This function prints the source capabilities received by the STUSB4500. Source capabilities are automatically stored on the device connection in a dedicated structure.

Figure 4. Print_PDO_FROM_SRC (Port);

```
---- Usb_Port #0: Read PDO from SOURCE ---------
5 Objects => 5 fixed PDO:
 - PDO1 (fixed)     = (5.00V, 3.00A, = 15.0W)
 - PDO2 (fixed)     = (9.00V, 3.00A, = 27.0W)
 - PDO3 (fixed)     = (12.00V, 3.00A, = 36.0W)
 - PDO4 (fixed)     = (15.00V, 3.00A, = 45.0W)
 - PDO5 (fixed)     = (20.00V, 2.25A, = 45.0W)
P(max)=45.0W
```

3.9 Read_RDO (Port);

This function reads the requested data object (RDO) register, in order to access contract PDO number.

3.10 Print_RDO (Port);

This function reads the requested data object (RDO) register, and prints the current contract to the serial interface in case of capability match between the STUSB4500 and the source.

Figure 5. Print_RDO (Port);

```
---- Usb_Port #0: CONNECTION STATUS   ----------
 - CONTRACT              : EXPLICIT
 - Requested PDO #       : 5
 - Voltage requested     : 20.00V
 - PIN                   : CC1
 - Max Current           : 2.25A
 - Operating Current     : 1.00A
 - Capability Mismatch   : 0
 - Give back             : 0
 - USB Com Capable       : 0
 - USB suspend           : 0
```

3.11 Update_PDO1 (Port, current)

This function can be used to overwrite PDO1 in RAM.

Arguments are:

- I²C port number
- Current (in mA) truncated by 10 mA

3.12 Update_PDO(Port, PDO_number, voltage, current)

This function can be used to overwrite PDO2 or PDO3 content in RAM.

Arguments are:

- I²C port number
- PDO index to be updated: 2 (for PDO2) or 3 (for PDO3)
- Voltage (in mV) truncated by 50 mV
- Current (in mA) truncated by 10 mA

3.13 UPDATE_Valid_PDO_Number(Port, Number_PDO)

This function is used to overwrite the number of valid PDO.

Arguments are:

- I²C port number
- active PDO number: from 1 to 3

3.14 Print_Type_C_Only_Status(Port)

This function prints the USB-C information received by the STUSB4500, such as CC pin location and Rp value:

- CC pin information is useful for USB3.x data signals routing (TX/RX)
- Rp value information is useful to set charging current accordingly (when not in explicit contract)

Figure 6. Print_Type_C_Only_Status

3.15 Get_Device_STATUS(Usb_Port)

This function reports the STUSB4500 port status. It is the main entry point for system level decisions (implemented at application level in a system policy manager). The function prints the USB PD information received from the STUSB4500, allowing the application to implement further actions or decisions.

Reported device states are:

- Device not connected
- Device not connected but in attached wait
- Device connected at 5 V (USB-C)
- Device connected at 5 V (USB PD)
- Device connected but PDO from source are not available
- Device connected but no match found (versus the STUSB4500 SINK PDO)
- Device connected - match found - request ongoing
- Device connected - matching found - power OK
- Hard reset message received

3.16 Request_SRC_PDO_NUMBER(Usb_Port, SRC_PDO_position);

This function copies the SRC_PDO corresponding to the position set in parameter into STUSB4500 PDO2.This allows the STUSB4500 to negotiate with the SOURCE on the given PDO index, whatever its voltage node.

Arguments are:

- I²C port number
- position from the SOURCE PDO (in the SOURCE capability list received) to negotiate with

4 Application example

4.1 Negotiate_5V(Port);

The sample function reconfigures the PDO number to only one, so by default PDO1. In this manner, the STUSB4500 negotiates 5 V back with the source.

4.2 Find_Matching_SRC_PDO(Port, Min_Power, Min_V, Max_V);

This function can be used to check if one of the source PDO (the sink is connected to) is compatible with the sink application. In such case, a PDO from the sink can be defined dynamically with compatible parameters in order to renegotiate at the compatible voltage node. In practice, the function scans the source PDO (received on connection). If one of the source PDO falls within the range of the function arguments, ie. within a voltage range (min. and max.) and with enough current at the given voltage, then it redefines the SINK_PDO3 with the identified and matching source PDO. This allows the STUSB4500 to best match the source capabilities.

4.3 Device_Manager(Usb_Port);

This function is an example of system policy manager. It relies on **Get_Device_STATUS()** function to interact with the STUSB4500 and can be used as a entry point to implement system policy or to connect with other application functions.

5 Demo

5.1 STUSB4500_PDO_rolling_DEMO.bin

This demo sequentially changes PDO2 and PDO3 to 9 V and 12 V respectively, then 15 V and 20 V respectively. The number of valid PDO is successfully set to 2 and 3. It results a demo in which the STUSB4500 negotiates from 5 V to 9 V then 12 V then 15 V then 20 V, then back to 15 V then 12 V, then 9 V etc.. etc..

This demo has been implemented by using a combination of a timer calling the previously defined functions.

5.2 STUSB4500_SRC_scan_DEMO.bin

This demo scans the SOURCE PDO at the connection and negotiates each individual profile sequentially every 5 seconds using the Request_SRC_PDO_NUMBER function.

By pressing the blue push button from the NUCLEO board, different functions are activated:

- Press #1: STUSB4500 forces 5V profile negotiation, using Negotiate_5V function
- Press #2: STUSB4500 negotiates with the highest SOURCE PDO between 14 V and 20 V as long as it is 15 W or more
- Press #3: the STUSB4500_PDO_rolling_DEMO is called
- Press #4: Hardware reset is called, forcing the STUSB4500 to detach, loosing VBUS, advertise itself as a SINK, negotiate again in order to power the NUCLEO board and re-start software sequence

Revision history

Table 2. **Document revision history**

Date	Version	Changes
05-Sep-2018	1	Initial release.
20-Jun-2019	2	Updated Section 3 Low level drivers, hardware abstraction functions, Section 3.8 Print_PDO_FROM_SRC (Port);, Section 3.10 Print_RDO (Port);. Added Section 3.5 Send_Soft_reset_Message(Port), Section 3.14 Print_Type_C_Only_Status(Port) and Section 3.16 Request_SRC_PDO_NUMBER(Usb_Port, SRC_PDO_position);.
14-Dec-2020	3	Updated Section 3 Low level drivers, hardware abstraction functions. Add Section 3.11 Update_PDO1 (Port, current), Section 3.15 Get_Device_STATUS(Usb_Port), Section 4.3 Device_Manager(Usb_Port);.

Contents