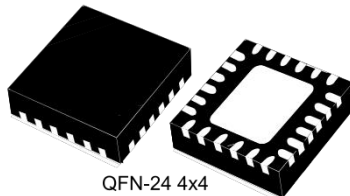


STUSB4500 NVM registers

Description and generic access



Related products

- STUSB4500
Standalone USB PD sink controller with short-to-VBUS protections
- STEVAL-ISC005V1
Evaluation board for the STUSB4500 USB Power Delivery controller
- STREF-SCS001V1
Fast and easy migration from DC barrel to Type-C

Scope

- Summary of STUSB4500 NVM operation
- NVM organization
- I2C access description for NVM Read and Write
- "C" code example
- Full NVM Mapping
- Practical example with STSW-STUSB002 GUI software import and export



Content

1	NVM Principle.....	4
1.1	IC Startup	4
1.2	NVM Organization	4
1.3	NVM map full list.....	5
2	I2C NVM Access	7
2.1	NVM registers	7
2.1.1	Control registers :	7
2.1.2	Data registers :	8
2.2	NVM READ procedure.....	8
2.2.1	Procedure.....	8
2.2.2	Pseudo Code example for full memory read	10
2.3	NVM WRITE procedure	11
2.3.1	Procedure.....	11
2.3.2	Pseudo Code example for full memory Write	13
3	“C” code example	15
3.1	I2C API functions	15
3.2	Header & definition	15
3.1	nvm_flash	16
3.2	EnterReadMode	16
3.3	ReadSector.....	16
3.4	EnterWriteMode.....	17
3.5	WriteSector.....	18
3.6	ExitTestMode.....	18
4	NVM Map	19
4.1	NVM Map customization through STSW-STUSB002 GUI	19
4.1.1	STSW-STUSB002 GUI file format.....	19
4.1.2	STUSB4500 NVM default content (GUI file format).....	19
4.1.1	NVM file interface with STSW-STUSB002 GUI	20
4.2	NVM Map Detailed Content	21
5	Example	29
5.1	Using an Aardvark for I2C & GUI.....	29
5.1.1	Check default configuration.....	29
5.1.2	Read STUSB4500 NVM map.....	30
5.1.3	Extracting the NVM content.....	31
5.1.4	Change parameters with STSW-STUSB002 GUI.....	31
5.1.5	Check modified parameters	33
5.1.6	Insert NVM content into Aardvark Batch file.....	34
5.1.7	Re-Read STUSB4500 NVM map	37
5.1.8	Check new configuration	38

1 NVM Principle

1.1 IC Startup

The STUSB4500 is a USB power delivery controller that addresses sink devices.

It is a full autonomous and auto-run device that implements a proprietary algorithm to allow the negotiation of a power delivery contract with a source without MCU support. PDO profiles and other parameters are configured in an integrated non-volatile memory (NVM).

When the device starts, NVM content is loaded into registers, and parameters are used by the algorithms and state machines.

Device behaviour can be customised by changing some values in NVM. A specific procedure based on I2C accesses allows the read, erase and write of NVM.

Parameters are described in STUSB4500 datasheet, and NVM mapping is described in this document.

Thus, to avoid side effects, it is strongly recommended to use ST provided tools such as [STSW-STUSB002](#) GUI software to modify the parameters.

1.2 NVM Organization

Memory is organized into 5 banks of 64bits

Each bank can be addressed individually.

For any operation, the whole content of the bank (all the 64bits) will be affected.

Memory map : Addresses

Bank	Address							
0	0xC0	0xC1	0xC2	0xC3	0xC4	0xC5	0xC6	0xC7
1	0xC8	0xC9	0xCA	0xCB	0xCC	0xCD	0xCE	0xCF
2	0xD0	0xD1	0xD2	0xD3	0xD4	0xD5	0xD6	0xD7
3	0xD8	0xD9	0xDA	0xDB	0xDC	0xDD	0xDE	0xDF
4	0xE0	0xE1	0xE2	0xE3	0xE4	0xE5	0xE6	0xE7

Memory map : Data

Bank	Data							
0	Data_C0	Data_C1	Data_C2	Data_C3	Data_C4	Data_C5	Data_C6	Data_C7
1	Data_C8	Data_C9	Data_CA	Data_CB	Data_CC	Data_CD	Data_CE	Data_CF
2	Data_D0	Data_D1	Data_D2	Data_D3	Data_D4	Data_D5	Data_D6	Data_D7
3	Data_D8	Data_D9	Data_DA	Data_DB	Data_DC	Data_DD	Data_DE	Data_DF
4	Data_E0	Data_E1	Data_E2	Data_E3	Data_E4	Data_E5	Data_E6	Data_E7

Memory map : Default Values

Bank	Data							
0	0x00	0x00	0xB0	0xAA	0x00	0x45	0x00	0x00
1	0x10	0x40	0x9C	0x1C	0xFF	0x01	0x3C	0xDF
2	0x02	0x40	0x0F	0x00	0x32	0x00	0xFC	0xF1
3	0x00	0x19	0x56	0xAF	0xF5	0x35	0x5F	0x00
4	0x00	0x4B	0x90	0x21	0x43	0x00	0x40	0xFB

1.3 NVM map full list

Bank 0

Addr	Content
0xC0	RESERVED : VENDOR_ID_LOW = 0x00
0xC1	RESERVED : VENDOR_ID_HIGH = 0x00
0xC2	RESERVED : PRODUCT_ID_LOW = 0xB0
0xC3	RESERVED : PRODUCT_ID_HIGH = 0xAA
0xC4	RESERVED : BCD_DEVICE_ID_LOW = 0x00
0xC5	RESERVED : BCD_DEVICE_ID_LOW = 0x45
0xC6	RESERVED : PORT_ROLE_CTRL = 0x00
0xC7	RESERVED : DEVICE_POWER_ROLE_CTRL = 0x00

Bank 1

Addr	Content		
0xC8	RESERVED : 0b00	GPIO_CFG[1:0]	RESERVED : 0x0
0xC9	0	1	VBUS_DCHG_MASK RESERVED : 0b00000
0xCA	DISCHARGE_TIME_TO_0V[3:0]		VBUS_DISCH_TIME_TO_PDO[3:0]
0xCB	RESERVED : 0x1C		
0xCC	RESERVED : 0xFF		
0xCD	RESERVED : 0x01		
0xCE	RESERVED : 0x3C		
0xCF	RESERVED : 0xDF		

Bank 2

Addr	Content
0xD0	RESERVED : 0x02
0xD1	RESERVED : 0x40
0xD2	RESERVED : 0x0F
0xD3	RESERVED : 0x00
0xD4	RESERVED : 0x32
0xD5	RESERVED : 0x00
0xD6	RESERVED : 0xFC
0xD7	RESERVED : 0xF1

Bank 3

Addr	Content			
0xD8	RESERVED : 0x00			
0xD9	RESERVED : 0x19			
0xDA	LUT_SNK_PDO1_I[3:0]	SNK_UNCONS_POWER	DPM_SNK_PDO_NUMB[1:0]	USB_COMM_CAPABLE
0xDB	SNK_HL1[3:0]	SNK_LL1[3:0]		
0xDC	SNK_LL2[3:0]	LUT_SNK_PDO2_I[3:0]		
0xDD	LUT_SNK_PDO3_I[3:0]	SNK_HL2[3:0]		
0xDE	SNK_HL3[3:0]	SNK_LL3[3:0]		
0xDF	RESERVED : SNK_PDO_FILL_0xDF = 0x00			

Bank 4

Addr	Content			
0xE0	SNK_PDO_FLEX1_V[1:0]	RESERVED : 0b000000		
0xE1	SNK_PDO_FLEX1_V[9:2]			
0xE2	SNK_PDO_FLEX2_V[7:0]			
0xE3	SNK_PDO_FLEX_I[5:0]			SNK_PDO_FLEX2_V[9:8]
0xE4	0	POWER_OK_CFG[1:0]	0	SNK_PDO_FLEX_I[9:6]
0xE5	RESERVED : SPARE = 0x00			
0xE6	RESERVED : 0b010	REQ_SRC_CURRENT	RESERVED : 0x0	
0xE7	RESERVED : ALERT_STATUS_1_MASK = 0xFB			

2 I2C NVM Access

NVM access is done through STUSB4500 I2C read and write commands to specific registers.

2.1 NVM registers

2.1.1 Control registers :

Registers list

Addr	Content
0x95	FTP_KEY
0x96	FTP_CTRL_0
0x97	FTP_CTRL_1

FTP_KEY

Bit	7	6	5	4	3	2	1	0
Content	FTP_KEY							

Address: 0x95

Default: 0x00

Description: FTP_KEY register

[7:0]	FTP_KEY: Customer FTP access Key
-------	----------------------------------

FTP_CTRL_0

Bit	7	6	5	4	3	2	1	0
Content	FTP_CUST_PWR	FTP_CUST_RST_N	RESERVED	FTP_CUST_REQ	RESERVED	FTP_CUST_SECT		

Address: 0x96

Default: 0x40

Description: FTP_CTRL_0 register

[7]	FTP_CUST_PWR: Not used
[6]	FTP_CUST_RST_N: NVM macro-cell reset in customer mode (Active Low) 0: Active reset 1: No reset
[5]	RESERVED = 0
[4]	FTP_CUST_REQ: Access request to NVM in customer mode
[3]	RESERVED = 0
[2:1]	FTP_CUST_SECT: 000: Sector 0 accessed 001: Sector 1 accessed 010: Sector 2 accessed 011: Sector 3 accessed 100: Sector 4 accessed others: Not allowed in customer mode (In this case sector 0 accessed)

FTP_CTRL_1

Bit	7	6	5	4	3	2	1	0
Content	FTP_CUST_SER					FTP_CUST_OPCODE		

Address: 0x97

Default: 0x00

Description: FTP_CTRL_1 register

[7:3]	FTP_CUST_SER: NVM sector input in customer mode 0000: (NO_SECTOR) No sector selected xxxx1: Sector 0 selected xxx1x: Sector 1 selected xx1xx: Sector 2 selected x1xxx: Sector 3 selected 1xxxx: Sector 4 selected
[2:0]	FTP_CUST_OPCODE: NVM operation in customer mode 000: Read memory array 001: Shift In Data on Program Load Register 010: Shift In Data on Sector Erase Register 011: Shift Out Data on Program Load Register 100: Shift Out Data on sector Erase Register 101: Erase memory array 110: Program word into EEPROM 111: Soft Program array

2.1.2 Data registers :

NVM Data Read / Write registers

Addr	Content
0x53	NVM data : LSB
0x54	NVM data
0x55	NVM data
0x56	NVM data
0x57	NVM data
0x58	NVM data
0x59	NVM data
0x5A	NVM data : MSB

2.2 NVM READ procedure

2.2.1 Procedure

The following operations shall be done for NVM read:

2.2.1.1 NVM Accessibility

Before any operation, the customer access key must be written in the FTP_KEY register. This write gives the access to the FTP_CTRL_0 and FTP_CTRL_1 registers.

- Unlock NVM by writing password in FTP_KEY register :
 - I2C Write : dev_addr = 0x28, reg_addr = 0x95, data = 0x47

2.2.1.2 NVM Power-up / Reset Sequence

After STUSB4500 start-up sequence, the NVM is powered off.

Before any customer operation, the NVM must be powered on and reset pulse must be applied by the following sequence:

- Put NVM internal controller and NVM in operational conditions : write FTP_CUST_RST_N at '1' in FTP_CTRL_0 register :
 - I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x40

- Reset NVM internal controller and NVM : write FTP_CUST_RST_N at '0' in FTP_CTRL_0 register :
 - I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x00
- A temporization upper than 2 us must be observed before the following write.
- Put NVM internal controller and NVM in operational conditions : write FTP_CUST_RST_N at '1' in FTP_CTRL_0 register :
 - I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x40

2.2.1.3 NVM Customer Sector0 Read

- Set Read Sector Opcode (0b000) in FTP_CTRL_1 register :
 - I2C Write : dev_addr = 0x28, reg_addr = 0x97, data = 0x00
- Load Opcode : set FTP_CUST_REQ = '1' and FTP_CUST_SECT = 0b000 in register FTP_CTRL_0 :
 - I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x50
- Wait for command execution : 1ms
- Read NVM Data (8 bytes) at starting address 0x53
 - (Less Significant Byte)
 - Data_C0 : I2C Read : dev_addr = 0x28, reg_addr = 0x53
 - Data_C1 : I2C Read : dev_addr = 0x28, reg_addr = 0x54
 - Data_C2 : I2C Read : dev_addr = 0x28, reg_addr = 0x55
 - Data_C3 : I2C Read : dev_addr = 0x28, reg_addr = 0x56
 - Data_C4 : I2C Read : dev_addr = 0x28, reg_addr = 0x57
 - Data_C5 : I2C Read : dev_addr = 0x28, reg_addr = 0x58
 - Data_C6 : I2C Read : dev_addr = 0x28, reg_addr = 0x59
 - (Most Significant Byte)
 - Data_C7 : I2C Read : dev_addr = 0x28, reg_addr = 0x5A

2.2.1.4 NVM Customer Sector1 Read

- Set Read Sector Opcode (0b000) in FTP_CTRL_1 register :
 - I2C Write : dev_addr = 0x28, reg_addr = 0x97, data = 0x00
- Load Opcode : set FTP_CUST_REQ = '1' and FTP_CUST_SECT = 0b001 in register FTP_CTRL_0 :
 - I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x51
- Wait for command execution : 1ms
- Read NVM Data (8 bytes) at starting address 0x53
 - (Less Significant Byte)
 - Data_C8 : I2C Read : dev_addr = 0x28, reg_addr = 0x53
 - Data_C9 : I2C Read : dev_addr = 0x28, reg_addr = 0x54
 - Data_CA : I2C Read : dev_addr = 0x28, reg_addr = 0x55
 - Data_CB : I2C Read : dev_addr = 0x28, reg_addr = 0x56
 - Data_CC : I2C Read : dev_addr = 0x28, reg_addr = 0x57
 - Data_CD : I2C Read : dev_addr = 0x28, reg_addr = 0x58
 - Data_CE : I2C Read : dev_addr = 0x28, reg_addr = 0x59
 - (Most Significant Byte)
 - Data_CF : I2C Read : dev_addr = 0x28, reg_addr = 0x5A

2.2.1.5 NVM Customer Sector2 Read

- Set Read Sector Opcode (0b000) in FTP_CTRL_1 register :
 - I2C Write : dev_addr = 0x28, reg_addr = 0x97, data = 0x00
- Load Opcode : set FTP_CUST_REQ = '1' and FTP_CUST_SECT = 0b010 in register FTP_CTRL_0 :
 - I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x52
- Wait for command execution : 1ms
- Read NVM Data (8 bytes) at starting address 0x53
 - (Less Significant Byte)
 - Data_D0 : I2C Read : dev_addr = 0x28, reg_addr = 0x53
 - Data_D1 : I2C Read : dev_addr = 0x28, reg_addr = 0x54
 - Data_D2 : I2C Read : dev_addr = 0x28, reg_addr = 0x55
 - Data_D3 : I2C Read : dev_addr = 0x28, reg_addr = 0x56
 - Data_D4 : I2C Read : dev_addr = 0x28, reg_addr = 0x57
 - Data_D5 : I2C Read : dev_addr = 0x28, reg_addr = 0x58
 - Data_D6 : I2C Read : dev_addr = 0x28, reg_addr = 0x59
 - (Most Significant Byte)
 - Data_D7 : I2C Read : dev_addr = 0x28, reg_addr = 0x5A

2.2.1.6 NVM Customer Sector3 Read

- Set Read Sector Opcode (0b000) in FTP_CTRL_1 register :
 - I2C Write : dev_addr = 0x28, reg_addr = 0x97, data = 0x00
- Load Opcode : set FTP_CUST_REQ = '1' and FTP_CUST_SECT = 0b011 in register FTP_CTRL_0 :
- I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x53
- Wait for command execution : 1ms
- Read NVM Data (8 bytes) at starting address 0x53
 - (Less Significant Byte)
Data_D8 : I2C Read : dev_addr = 0x28, reg_addr = 0x53
 - Data_D9 : I2C Read : dev_addr = 0x28, reg_addr = 0x54
 - Data_DA : I2C Read : dev_addr = 0x28, reg_addr = 0x55
 - Data_DB : I2C Read : dev_addr = 0x28, reg_addr = 0x56
 - Data_DC : I2C Read : dev_addr = 0x28, reg_addr = 0x57
 - Data_DD : I2C Read : dev_addr = 0x28, reg_addr = 0x58
 - Data_DE : I2C Read : dev_addr = 0x28, reg_addr = 0x59
 - (Most Significant Byte)
Data_DF : I2C Read : dev_addr = 0x28, reg_addr = 0x5A

2.2.1.7 NVM Customer Sector4 Read

- Set Read Sector Opcode (0b000) in FTP_CTRL_1 register :
 - I2C Write : dev_addr = 0x28, reg_addr = 0x97, data = 0x00
- Load Opcode : set FTP_CUST_REQ = '1' and FTP_CUST_SECT = 0b100 in register FTP_CTRL_0 :
 - I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x54
- Wait for command execution : 1ms
- Read NVM Data (8 bytes) at starting address 0x53
 - (Less Significant Byte)
Data_E0 : I2C Read : dev_addr = 0x28, reg_addr = 0x53
 - Data_E1 : I2C Read : dev_addr = 0x28, reg_addr = 0x54
 - Data_E2 : I2C Read : dev_addr = 0x28, reg_addr = 0x55
 - Data_E3 : I2C Read : dev_addr = 0x28, reg_addr = 0x56
 - Data_E4 : I2C Read : dev_addr = 0x28, reg_addr = 0x57
 - Data_E5 : I2C Read : dev_addr = 0x28, reg_addr = 0x58
 - Data_E6 : I2C Read : dev_addr = 0x28, reg_addr = 0x59
 - (Most Significant Byte)
Data_E7 : I2C Read : dev_addr = 0x28, reg_addr = 0x5A

2.2.1.8 Exit Test mode

- Clear FTP_CTRL registers
 - I2C Write : dev_addr = 0x28, reg_addr = 0x96,
data = 0x40, 0x00
- Clear FTP_KEY register
 - I2C Write : dev_addr = 0x28, reg_addr = 0x95, data = 0x00

2.2.2 Pseudo Code example for full memory read

```

I2C Write : dev_addr = 0x28, reg_addr = 0x95, data = 0x47
I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x40
I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x00
Wait for command execution : 1ms
I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x40
I2C Write : dev_addr = 0x28, reg_addr = 0x97, data = 0x00
I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x50
Wait for command execution : 1ms
[Data_C0..Data_C7] : 8 Bytes I2C Read : dev_addr = 0x28, reg_addr = 0x53
I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x51
Wait for command execution : 1ms
[Data_C8..Data_CF] : 8 Bytes I2C Read : dev_addr = 0x28, reg_addr = 0x53
I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x52
Wait for command execution : 1ms
[Data_D0..Data_D7] : 8 Bytes I2C Read : dev_addr = 0x28, reg_addr = 0x53
I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x53
  
```

```

Wait for command execution : 1ms
[Data_D8..Data_DF] : 8 Bytes I2C Read : dev_addr = 0x28, reg_addr = 0x53
I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x54
Wait for command execution : 1ms
[Data_E0..Data_E7] : 8 Bytes I2C Read : dev_addr = 0x28, reg_addr = 0x53
I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x40, 0x00
I2C Write : dev_addr = 0x28, reg_addr = 0x95, data = 0x00
  
```

2.3 NVM WRITE procedure

2.3.1 Procedure

The following operations shall be done for NVM write :

2.3.1.1 NVM Accessibility

Before any operation, the customer access key must be written in the FTP_KEY register. This write gives the access to the FTP_CTRL_0 and FTP_CTRL_1 registers.

- Unlock NVM by writing password in FTP_KEY register
 - I2C Write : dev_addr = 0x28, reg_addr = 0x95, data = 0x47

2.3.1.2 NVM Power-up Sequence

After STUSB4500 start-up sequence, the NVM is powered off.

Before any customer operation, the NVM must be powered on and reset pulse must be applied by the following sequence:

- Load 0x00 to data register
 - I2C Write : dev_addr = 0x28, reg_addr = 0x53, data = 0x00
- Put NVM internal controller and NVM in operational conditions : write FTP_CUST_PWR and FTP_CUST_RST_N at '1' in FTP_CTRL_0 register :
 - I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x40
- Reset NVM internal controller and NVM : write FTP_CUST_RST_N at '0' in FTP_CTRL_0 register :
 - I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x00
- A temporization upper than 2 us must be observed before the following write.
- Put NVM internal controller and NVM in operational conditions : write FTP_CUST_RST_N at '1' in FTP_CTRL_0 register :
 - I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x40

2.3.1.3 NVM Customer Full Erase

- Set "Shift In Data on Sector Erase Register" Opcode for all sectors : FTP_CUST_SER = 0b11111 and FTP_CUST_OPCODE = 0b010 in register FTP_CTRL_1
 - I2C Write : dev_addr = 0x28, reg_addr = 0x97, data = 0xFA
- Load Opcode : set FTP_CUST_REQ = '1' in register FTP_CTRL_0 :
 - I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x50
- Wait for command execution : 1ms
- Set "Soft Program array" : FTP_CUST_OPCODE = 0b111
 - I2C Write : dev_addr = 0x28, reg_addr = 0x97, data = 0x07
- Load Opcode : set FTP_CUST_REQ = '1' in register FTP_CTRL_0 :
 - I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x50
- Wait for EP (Memory Erase time) : 5ms
- Set "Erase memory array" Opcode : FTP_CUST_OPCODE = 0b101
 - I2C Write : dev_addr = 0x28, reg_addr = 0x97, data = 0x05
- Load Opcode : set FTP_CUST_REQ = '1' in register FTP_CTRL_0 :
 - I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x50
- Wait for EP (Memory Erase time) : 5ms

2.3.1.4 NVM Customer Sector0 Write

- Load NVM Data Sector 0 (8 bytes) at starting address 0x53
 - I2C Write : dev_addr = 0x28, reg_addr = 0x53, data = Data_C0, Data_C1, Data_C2, Data_C3, Data_C4, Data_C5, Data_C6, Data_C7
- Set "Shift In Data on Program Load Register" Opcode : FTP_CUST_OPCODE = 0b001
 - I2C Write : dev_addr = 0x28, reg_addr = 0x97, data = 0x01
- Load Opcode : set FTP_CUST_REQ = '1' in register FTP_CTRL_0 :

- I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x50
- Wait for command execution : 1ms
- Set “Program word into EEPROM” Opcode : FTP_CUST_OPCODE = 0b110
 - I2C Write : dev_addr = 0x28, reg_addr = 0x97, data = 0x06
- Load Opcode with sector value : set FTP_CUST_REQ = ‘1’ and FTP_CUST_SECT = ‘000’ in register FTP_CTRL_0 :
 - I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x50
- Wait for PP (Word Program time) : 2ms

2.3.1.5 NVM Customer Sector1 Write

- Load NVM Data Sector 1 (8 bytes) at starting address 0x53
 - I2C Write : dev_addr = 0x28, reg_addr = 0x53,
data = Data_C8, Data_C9, Data_CA, Data_CB, Data_CC, Data_CD, Data_CE, Data_CF
- Set “Shift In Data on Program Load Register” Opcode : FTP_CUST_OPCODE = 0b001
 - I2C Write : dev_addr = 0x28, reg_addr = 0x97, data = 0x01
- Load Opcode : set FTP_CUST_REQ = ‘1’ in register FTP_CTRL_0 :
 - I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x50
- Wait for command execution : 1ms
- Set “Program word into EEPROM” Opcode : FTP_CUST_OPCODE = 0b110
 - I2C Write : dev_addr = 0x28, reg_addr = 0x97, data = 0x06
- Load Opcode with sector value : set FTP_CUST_REQ = ‘1’ and FTP_CUST_SECT = ‘001’ in register FTP_CTRL_0 :
 - I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x51
- Wait for PP (Word Program time) : 2ms

2.3.1.6 NVM Customer Sector2 Write

- Load NVM Data Sector 2 (8 bytes) at starting address 0x53
 - I2C Write : dev_addr = 0x28, reg_addr = 0x53,
data = Data_D0, Data_D1, Data_D2, Data_D3, Data_D4, Data_D5, Data_D6, Data_D7
- Set “Shift In Data on Program Load Register” Opcode : FTP_CUST_OPCODE = 0b001
 - I2C Write : dev_addr = 0x28, reg_addr = 0x97, data = 0x01
- Load Opcode : set FTP_CUST_REQ = ‘1’ in register FTP_CTRL_0 :
 - I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x50
- Wait for command execution : 1ms
- Set “Program word into EEPROM” Opcode : FTP_CUST_OPCODE = 0b110
 - I2C Write : dev_addr = 0x28, reg_addr = 0x97, data = 0x06
- Load Opcode with sector value : set FTP_CUST_REQ = ‘1’ and FTP_CUST_SECT = ‘010’ in register FTP_CTRL_0 :
 - I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x52
- Wait for PP (Word Program time) : 2ms

2.3.1.7 NVM Customer Sector3 Write

- Load NVM Data Sector 3 (8 bytes) at starting address 0x53
 - I2C Write : dev_addr = 0x28, reg_addr = 0x53,
data = Data_D8, Data_D9, Data_DA, Data_DB, Data_DC, Data_DD, Data_DE, Data_DF
- Set “Shift In Data on Program Load Register” Opcode : FTP_CUST_OPCODE = 0b001
 - I2C Write : dev_addr = 0x28, reg_addr = 0x97, data = 0x01
- Load Opcode : set FTP_CUST_REQ = ‘1’ in register FTP_CTRL_0 :
 - I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x50
- Wait for command execution : 1ms
- Set “Program word into EEPROM” Opcode : FTP_CUST_OPCODE = 0b110
 - I2C Write : dev_addr = 0x28, reg_addr = 0x97, data = 0x06
- Load Opcode with sector value : set FTP_CUST_REQ = ‘1’ and FTP_CUST_SECT = ‘011’ in register FTP_CTRL_0 :
 - I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x53
- Wait for PP (Word Program time) : 2ms

2.3.1.8 NVM Customer Sector4 Write

- Load NVM Data Sector 4 (8 bytes) at starting address 0x53

- I2C Write : dev_addr = 0x28, reg_addr = 0x53,
data = Data_E0, Data_E1, Data_E2, Data_E3, Data_E4, Data_E5, Data_E6, Data_E7
- Set “Shift In Data on Program Load Register” Opcode : FTP_CUST_OPCODE = 0b001
 - I2C Write : dev_addr = 0x28, reg_addr = 0x97, data = 0x01
- Load Opcode : set FTP_CUST_REQ = ‘1’ in register FTP_CTRL_0 :
 - I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x50
- Wait for command execution : 1ms
- Set “Program word into EEPROM” Opcode : FTP_CUST_OPCODE = 0b110
 - I2C Write : dev_addr = 0x28, reg_addr = 0x97, data = 0x06
- Load Opcode with sector value : set FTP_CUST_REQ = ‘1’ and FTP_CUST_SECT = ‘100’ in register FTP_CTRL_0 :
 - I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x54
- Wait for PP (Word Program time) : 2ms

2.3.1.9 Exit Test mode

- Clear FTP_CTRL registers
 - I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x40, 0x00
- Clear FTP_KEY register
 - I2C Write : dev_addr = 0x28, reg_addr = 0x95, data = 0x00

2.3.2 Pseudo Code example for full memory Write

```

I2C Write : dev_addr = 0x28, reg_addr = 0x95, data = 0x47
I2C Write : dev_addr = 0x28, reg_addr = 0x53, data = 0x00
I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x40
I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x00
Wait for command execution : 1ms
I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x40
I2C Write : dev_addr = 0x28, reg_addr = 0x97, data = 0xFA
I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x50
Wait for command execution : 1ms
I2C Write : dev_addr = 0x28, reg_addr = 0x97, data = 0x07
I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x50
Wait for EP (Memory Erase time) : 5ms
I2C Write : dev_addr = 0x28, reg_addr = 0x97, data = 0x05
I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x50
Wait for EP (Memory Erase time) : 5ms
I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = [Data_C0..Data_C7]
Wait for command execution : 1ms
I2C Write : dev_addr = 0x28, reg_addr = 0x97, data = 0x01
I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x50
Wait for command execution : 1ms
I2C Write : dev_addr = 0x28, reg_addr = 0x97, data = 0x06
I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x50
Wait for PP (Word Program time) : 2ms
I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = [Data_C8..Data_CF]
Wait for command execution : 1ms
I2C Write : dev_addr = 0x28, reg_addr = 0x97, data = 0x01
I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x50
Wait for command execution : 1ms
I2C Write : dev_addr = 0x28, reg_addr = 0x97, data = 0x06
I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x51
Wait for PP (Word Program time) : 2ms
I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = [Data_D0..Data_D7]
Wait for command execution : 1ms
I2C Write : dev_addr = 0x28, reg_addr = 0x97, data = 0x01
I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x50
Wait for command execution : 1ms
I2C Write : dev_addr = 0x28, reg_addr = 0x97, data = 0x06
I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x52
  
```

```
Wait for PP (Word Program time) : 2ms
I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = [Data_D8..Data_DF]
Wait for command execution : 1ms
I2C Write : dev_addr = 0x28, reg_addr = 0x97, data = 0x01
I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x50
Wait for command execution : 1ms
I2C Write : dev_addr = 0x28, reg_addr = 0x97, data = 0x06
I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x53
Wait for PP (Word Program time) : 2ms
I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = [Data_E0..Data_E7]
Wait for command execution : 1ms
I2C Write : dev_addr = 0x28, reg_addr = 0x97, data = 0x01
I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x50
Wait for command execution : 1ms
I2C Write : dev_addr = 0x28, reg_addr = 0x97, data = 0x06
I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x54
Wait for PP (Word Program time) : 2ms
I2C Write : dev_addr = 0x28, reg_addr = 0x96, data = 0x40, 0x00
I2C Write : dev_addr = 0x28, reg_addr = 0x95, data = 0x00
```

3 “C” code example

3.1 I2C API functions

```

HAL_StatusTypeDef I2C_Read_USB_PD(uint8_t Port, uint16_t Address ,uint8_t *DataR ,uint16_t Length);
/* I2C Read for STUSB4500
   Device address : DevADDR defined by STUSB4500 IC, ADDR0 and ADDR1 pins
   (default : 0x28 in 7 bits format)
   uint16_t Address : Register address RegADDR
   uint8_t *DataR : data pointer
   uint16_t Length : number of data to read */

HAL_StatusTypeDef I2C_Write_USB_PD(uint8_t Port, uint16_t Address ,uint8_t *DataW ,uint16_t Length);
/* I2C Write for STUSB4500
   Device address : DevADDR defined by STUSB4500 IC, ADDR0 and ADDR1 pins
   (default : 0x28 in 7 bits format)
   uint16_t Address : Register address RegADDR
   uint8_t *DataW : data pointer
   uint16_t Length : number of data to write */

```

3.2 Header & definition

```

/*NVM Flasher Registers Definition */
#define FTP_CUST_PASSWORD_REG    0x95
#define FTP_CUST_PASSWORD       0x47
#define FTP_CTRL_0              0x96
#define FTP_CUST_PWR            0x80
#define FTP_CUST_RST_N          0x40
#define FTP_CUST_REQ            0x10
#define FTP_CUST_SECT           0x07
#define FTP_CTRL_1              0x97
#define FTP_CUST_SER            0xF8
#define FTP_CUST_OPCODE         0x07
#define RW_BUFFER                0x53

/*"000" then No Operation
"001" then Read
"010" and FTP_ADR[2:0]="000" then Shift-In Write Bit Data (0x20-0x28). (to be done before
Programming)
"010" and FTP_ADR[2:0]="001" then Shift-In Erase Sector Data (0x20). (to be done before Erasing)
"011" and FTP_ADR[2:0]="000" then Shift-Out Read Bit Data (0x20-0x28). (to be done after Reading)
"011" and FTP_ADR[2:0]="001" then Shift-Out Erase Sector Data (0x20). (to be done after Erasing)
"100" then Verify (to be done after Programming)
"101" then Erase
"110" then Program
"111" then Soft Programming (to be done after Erasing)*/
#define READ                    0x00
#define WRITE_PL                0x01
#define WRITE_SER               0x02
#define READ_PL                 0x03
#define READ_SER                0x04
#define ERASE_SECTOR            0x05
#define PROG_SECTOR             0x06
#define SOFT_PROG_SECTOR        0x07
#define SECTOR_0                0x01
#define SECTOR_1                0x02
#define SECTOR_2                0x04
#define SECTOR_3                0x08
#define SECTOR_4                0x10

```

3.1 nvm_flash

```
uint8_t nvm_flash(uint8_t Port)
{
    if (CUST_EnterWriteMode(0, SECTOR_0 | SECTOR_1 | SECTOR_2 | SECTOR_3 | SECTOR_4 ) != 0 ) return 1;

    if (CUST_WriteSector(0,0,Sector0) != 0 ) return 1;
    if (CUST_WriteSector(0,1,Sector1) != 0 ) return 1;
    if (CUST_WriteSector(0,2,Sector2) != 0 ) return 1;
    if (CUST_WriteSector(0,3,Sector3) != 0 ) return 1;
    if (CUST_WriteSector(0,4,Sector4) != 0 ) return 1;

    if (CUST_ExitTestMode(0) != 0 ) return 1;

    return 0;
}
```

3.2 EnterReadMode

```
uint8_t CUST_EnterReadMode(uint8_t Port)
{
    unsigned char Buffer[10];

    /* Set Password*/
    Buffer[0]=FTP_CUST_PASSWORD;
    if ( I2C_Write_USB_PD(Port,FTP_CUST_PASSWORD_REG,Buffer,1) != HAL_OK ) return 1;

    /* Set RST_N bit */
    Buffer[0]= FTP_CUST_RST_N ;
    if ( I2C_Write_USB_PD(Port,FTP_CTRL_0,Buffer,1) != HAL_OK ) return 1;

    /* Reset NVM */
    Buffer[0]=0;
    if ( I2C_Write_USB_PD(Port,FTP_CTRL_0,Buffer,1) != HAL_OK ) return 1;

    /* Set RST_N bit */
    Buffer[0]= FTP_CUST_RST_N ;
    if ( I2C_Write_USB_PD(Port,FTP_CTRL_0,Buffer,1) != HAL_OK ) return 1;

    return 0 ;
}
```

3.3 ReadSector

```
uint8_t CUST_ReadSector(uint8_t Port,char SectorNum, unsigned char *SectorData)
{
    unsigned char Buffer[10];

    /* Set Read Sectors Opcode */
    Buffer[0]= (READ & FTP_CUST_OPCODE);
    if ( I2C_Write_USB_PD(Port,FTP_CTRL_1,Buffer,1) != HAL_OK ) return 1;

    /* Load Read Sectors Opcode */
    Buffer[0]= (SectorNum & FTP_CUST_SECT) | FTP_CUST_RST_N | FTP_CUST_REQ;
    if ( I2C_Write_USB_PD(Port,FTP_CTRL_0,Buffer,1) != HAL_OK ) return 1;

    /* Wait for execution */
    do
    {
        if ( I2C_Read_USB_PD(Port,FTP_CTRL_0,Buffer,1) != HAL_OK ) return 1;
    }
    while(Buffer[0] & FTP_CUST_REQ);

    /* Sectors Data are available in RW-BUFFER @ 0x53 */
    I2C_Read_USB_PD(Port,RW_BUFFER,&SectorData[0],8);
}
```


3.4 EnterWriteMode

```

uint8_t CUST_EnterWriteMode(uint8_t Port,unsigned char ErasedSector)
{
    unsigned char Buffer[10];

    /* Set Password*/
    Buffer[0]=FTP_CUST_PASSWORD;
    if ( I2C_Write_USB_PD(Port,FTP_CUST_PASSWORD_REG,Buffer,1) != HAL_OK ) return 1;

    /* this register must be NULL for Partial Erase feature */
    Buffer[0]= 0;
    if ( I2C_Write_USB_PD(Port,RW_BUFFER,Buffer,1) != HAL_OK ) return 1;

    /* Set RST_N bit */
    Buffer[0]= FTP_CUST_RST_N ;
    if ( I2C_Write_USB_PD(Port,FTP_CTRL_0,Buffer,1) != HAL_OK ) return 1;

    /* Reset NVM */
    Buffer[0]=0;
    if ( I2C_Write_USB_PD(Port,FTP_CTRL_0,Buffer,1) != HAL_OK ) return 1;

    /* Set RST_N bit */
    Buffer[0]= FTP_CUST_RST_N;
    if ( I2C_Write_USB_PD(Port,FTP_CTRL_0,Buffer,1) != HAL_OK ) return 1;

    /* Load 0xF1 to FTP_CUST_SER to erase all sectors of FTP and Set Write SER Opcode */
    Buffer[0]=((ErasedSector << 3) & FTP_CUST_SER) | ( WRITE_SER & FTP_CUST_OPCODE);
    if ( I2C_Write_USB_PD(Port,FTP_CTRL_1,Buffer,1) != HAL_OK ) return 1;

    /* Load Write SER Opcode */
    Buffer[0]= FTP_CUST_RST_N | FTP_CUST_REQ ;
    if ( I2C_Write_USB_PD(Port,FTP_CTRL_0,Buffer,1) != HAL_OK ) return 1;

    /* Wait for execution */do
    {
        if ( I2C_Read_USB_PD(Port,FTP_CTRL_0,Buffer,1) != HAL_OK ) return 1;
    }
    while(Buffer[0] & FTP_CUST_REQ);

    /* Set Soft Prog Opcode */
    Buffer[0]= SOFT_PROG_SECTOR & FTP_CUST_OPCODE ;
    if ( I2C_Write_USB_PD(Port,FTP_CTRL_1,Buffer,1) != HAL_OK ) return 1;

    /* Load Soft Prog Opcode */
    Buffer[0]= FTP_CUST_RST_N | FTP_CUST_REQ ;
    if ( I2C_Write_USB_PD(Port,FTP_CTRL_0,Buffer,1) != HAL_OK ) return 1;

    /* Wait for execution */
    do
    {
        if ( I2C_Read_USB_PD(Port,FTP_CTRL_0,Buffer,1) != HAL_OK ) return 1;
    }
    while(Buffer[0] & FTP_CUST_REQ);

    /* Set Erase Sectors Opcode */
    Buffer[0]= ERASE_SECTOR & FTP_CUST_OPCODE ;
    if ( I2C_Write_USB_PD(Port,FTP_CTRL_1,Buffer,1) != HAL_OK ) return 1;

    /* Load Erase Sectors Opcode */
    Buffer[0]= FTP_CUST_RST_N | FTP_CUST_REQ ;
    if ( I2C_Write_USB_PD(Port,FTP_CTRL_0,Buffer,1) != HAL_OK ) return 1;

    /* Wait for execution */
    do
    {
        if ( I2C_Read_USB_PD(Port,FTP_CTRL_0,Buffer,1) != HAL_OK ) return 1;
    }
}

```

```

    while(Buffer[0] & FTP_CUST_REQ);
    return 0;
}

```

3.5 WriteSector

```

uint8_t CUST_WriteSector(uint8_t Port, char SectorNum, unsigned char *SectorData)
{
    unsigned char Buffer[10];

    /* Write Sectors Data in RW-BUFFER @ 0x53 */
    I2C_Write_USB_PD(Port, RW_BUFFER, SectorData, 8);

    /*Set Write to PL Opcode*/
    Buffer[0]= (WRITE_PL & FTP_CUST_OPCODE);
    if ( I2C_Write_USB_PD(Port, FTP_CTRL_1, Buffer, 1) != HAL_OK ) return 1;

    /* Load Write to PL Sectors Opcode */
    Buffer[0]= FTP_CUST_RST_N | FTP_CUST_REQ;
    if ( I2C_Write_USB_PD(Port, FTP_CTRL_0, Buffer, 1) != HAL_OK ) return 1;

    /* Wait for execution */
    do
    {
        if ( I2C_Read_USB_PD(Port, FTP_CTRL_0, Buffer, 1) != HAL_OK ) return 1;
    }
    while(Buffer[0] & FTP_CUST_REQ) ;

    /*Set Prog Sectors Opcode*/
    Buffer[0]= (PROG_SECTOR & FTP_CUST_OPCODE);
    if ( I2C_Write_USB_PD(Port, FTP_CTRL_1, Buffer, 1) != HAL_OK ) return 1;

    /* Load Prog Sectors Opcode */
    Buffer[0]=(SectorNum & FTP_CUST_SECT) | FTP_CUST_RST_N | FTP_CUST_REQ;
    if ( I2C_Write_USB_PD(Port, FTP_CTRL_0, Buffer, 1) != HAL_OK ) return 1;

    /* Wait for execution */
    do
    {
        if ( I2C_Read_USB_PD(Port, FTP_CTRL_0, Buffer, 1) != HAL_OK ) return 1;
    }
    while(Buffer[0] & FTP_CUST_REQ) ;
    return 0;
}

```

3.6 ExitTestMode

```

uint8_t CUST_ExitTestMode(uint8_t Port)
{
    unsigned char Buffer[10];

    /* clear registers */
    Buffer[0]= FTP_CUST_RST_N; Buffer[1]=0x00;
    if ( I2C_Write_USB_PD(Port, FTP_CTRL_0, Buffer, 2) != HAL_OK ) return 1;

    /* Clear Password */
    Buffer[0]=0x00;
    if ( I2C_Write_USB_PD(Port, FTP_CUST_PASSWORD_REG, Buffer, 1) != HAL_OK ) return 1;

    return 0 ;
}

```

4 NVM Map

4.1 NVM Map customization through STSW-STUSB002 GUI

Thanks to [STSW-STUSB002](#) GUI software, parameters can be modified to configure STUSB4500 without studying the full map.

GUI can read and write STUSB4500 device through NUCLEO-F072RB interface board for evaluation, configuration validation and quick prototyping.

GUI can also interpret and generate text format files of NVM map.

4.1.1 STSW-STUSB002 GUI file format

```

0xC0: →Data_C0→Data_C1→Data_C2→Data_C3→Data_C4→Data_C5→Data_C6→Data_C7→CR LF
0xC8: →Data_C8→Data_C9→Data_CA→Data_CB→Data_CC→Data_CD→Data_CE→Data_CF→CR LF
0xD0: →Data_C0→Data_C1→Data_C2→Data_C3→Data_C4→Data_C5→Data_C6→Data_C7→CR LF
0xD8: →Data_C8→Data_C9→Data_CA→Data_CB→Data_CC→Data_CD→Data_CE→Data_CF→CR LF
0xE0: →Data_E0→Data_E1→Data_E2→Data_E3→Data_E4→Data_E5→Data_E6→Data_E7→CR LF
    
```

Symbols note :

→ : ASCII code 0x09 TAB (horizontal tab)

CR : ASCII code 0x0D CR (carriage return)

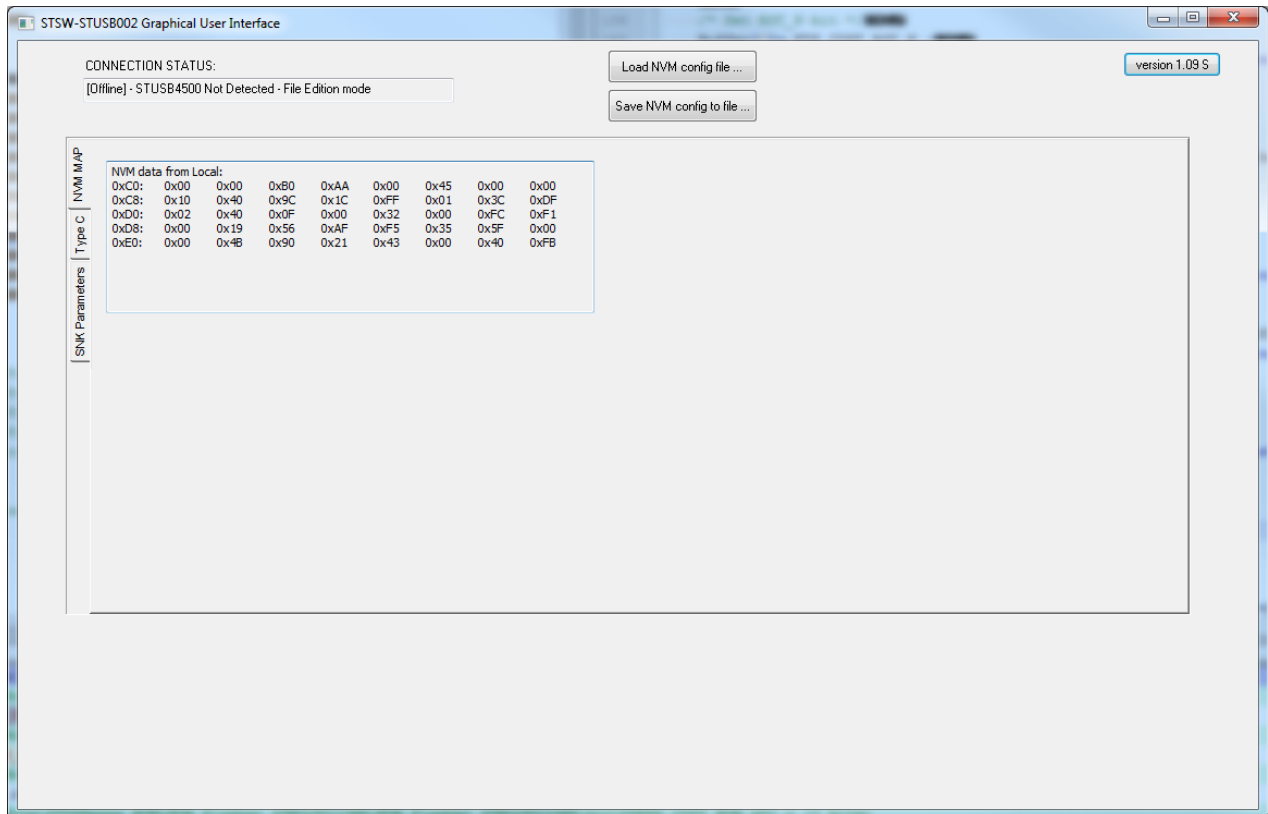
LF : ASCII code 0x0A LF (NL line feed, new line)

4.1.2 STUSB4500 NVM default content (GUI file format)

```

0xC0: 0x00 0x00 0xB0 0xAA 0x00 0x45 0x00 0x00
0xC8: 0x10 0x40 0x9C 0x1C 0xFF 0x01 0x3C 0xDF
0xD0: 0x02 0x40 0x0F 0x00 0x32 0x00 0xFC 0xF1
0xD8: 0x00 0x19 0x56 0xAF 0xF5 0x35 0x5F 0x00
0xE0: 0x00 0x4B 0x90 0x21 0x43 0x00 0x40 0xFB
    
```

4.1.1 NVM file interface with STSW-STUSB002 GUI



NVM mapping, read from I2C command and in the right file format, can be loaded into STSW-STUSB002 GUI with the “Load NVM config file ...” button.

Load NVM config file ...

Then the parameters can be displayed and modified in the other tabs (“SNK Parameters”, “TypeC”)

The “NVM MAP” tab reflects the modified parameters, and then the configuration can be exported with the “Save NVM config to file ...” button.

Save NVM config to file ...

The exported file contains the NVM Map that can be flashed with I2C commands previously described.

4.2 NVM Map Detailed Content

VENDOR_ID_LOW

Bit	7	6	5	4	3	2	1	0
Content	RESERVED : 0x00							

Address: 0xC0

Default: 0x00

Description: VENDOR_ID_LOW

[7:0]	RESERVED : VENDOR_ID_LOW = 0x00
-------	---------------------------------

VENDOR_ID_HIGH

Bit	7	6	5	4	3	2	1	0
Content	RESERVED : 0x00							

Address: 0xC1

Default: 0x00

Description: VENDOR_ID_HIGH

[7:0]	RESERVED : VENDOR_ID_HIGH = 0x00
-------	----------------------------------

PRODUCT_ID_LOW

Bit	7	6	5	4	3	2	1	0
Content	RESERVED : 0xB0							

Address: 0xC2

Default: 0xB0

Description: PRODUCT_ID_LOW

[7:0]	RESERVED : PRODUCT_ID_LOW = 0xB0
-------	----------------------------------

PRODUCT_ID_HIGH

Bit	7	6	5	4	3	2	1	0
Content	RESERVED : 0xAA							

Address: 0xC3

Default: 0xAA

Description: PRODUCT_ID_HIGH

[7:0]	RESERVED : PRODUCT_ID_HIGH = 0xAA
-------	-----------------------------------

BCD_DEVICE_ID_LOW

Bit	7	6	5	4	3	2	1	0
Content	RESERVED : 0x00							

Address: 0xC4

Default: 0x00

Description: BCD_DEVICE_ID_LOW

[7:0]	RESERVED : BCD_DEVICE_ID_LOW = 0x00
-------	-------------------------------------

BCD_DEVICE_ID_HIGH

Bit	7	6	5	4	3	2	1	0
Content	RESERVED : 0x45							

Address: 0xC5

Default: 0x45

Description: BCD_DEVICE_ID_HIGH

[7:0]	RESERVED : BCD_DEVICE_ID_LOW = 0x45
-------	-------------------------------------

PORT_ROLE_CTRL

Bit	7	6	5	4	3	2	1	0
Content	RESERVED : 0x00							

Address: 0xC6

Default: 0x00

Description: PORT_ROLE_CTRL

[7:0]	RESERVED : PORT_ROLE_CTRL = 0x00
-------	----------------------------------

DEVICE_POWER_ROLE_CTRL

Bit	7	6	5	4	3	2	1	0
Content	RESERVED : 0x00							

Address: 0xC7

Default: 0x00

Description: DEVICE_POWER_ROLE_CTRL

[7:0]	RESERVED : DEVICE_POWER_ROLE_CTRL = 0x00
-------	--

GPIO_CTRL

Bit	7	6	5	4	3	2	1	0
Content	0	0	GPIO_CFG[1:0]		0	0	0	0

Address: 0xC8

Default: 0x10

Description: GPIO_CTRL

[7:6]	RESERVED : 0b00
[5:4]	GPIO_CFG[1:0]
[3:0]	RESERVED : 0x0

ANALOG_CTRL

Bit	7	6	5	4	3	2	1	0
Content	0	1	VBUS_DCHG_MASK	0	0	0	0	0

Address: 0xC9

Default: 0x40

Description: ANALOG_CTRL

[7]	RESERVED : 0b0
[6]	RESERVED : 0b1
[5]	VBUS_DCHG_MASK
[4:0]	RESERVED : 0b00000

DISCHARGE_TIME_CTRL

Bit	7	6	5	4	3	2	1	0
Content	DISCHARGE_TIME_TO_0V[3:0]				VBUS_DISCH_TIME_TO_PDO[3:0]			

Address: 0xCA

Default: 0x9C

Description: DISCHARGE_TIME_CTRL

[7:4]	DISCHARGE_TIME_TO_0V[3:0]
[3:0]	VBUS_DISCH_TIME_TO_PDO[3:0]

RESERVED_0xCB

Bit	7	6	5	4	3	2	1	0
Content	RESERVED : 0x1C							

Address: 0xCB

Default: 0x1C

Description: RESERVED

[7:0]	RESERVED : 0x1C
-------	-----------------

RESERVED_0xCC

Bit	7	6	5	4	3	2	1	0
Content	RESERVED : 0xF0							

Address: 0xCC

Default: 0xF0

Description: RESERVED

[7:0]	RESERVED : 0xFF
-------	-----------------

RESERVED_0xCD

Bit	7	6	5	4	3	2	1	0
Content	RESERVED : 0x01							

Address: 0xCD

Default: 0x01

Description: RESERVED

[7:0]	RESERVED : 0x01
-------	-----------------

RESERVED_0xCE

Bit	7	6	5	4	3	2	1	0
Content	RESERVED : 0x00							

Address: 0xCE

Default: 0x00

Description: RESERVED

[7:0]	RESERVED : 0x3C
-------	-----------------

RESERVED_0xCF

Bit	7	6	5	4	3	2	1	0
Content	RESERVED : 0xDF							

Address: 0xCF

Default: 0xDF

Description: RESERVED

[7:0]	RESERVED : 0xDF
-------	-----------------

RESERVED_0xD0

Bit	7	6	5	4	3	2	1	0
Content	RESERVED : 0x02							

Address: 0xD0

Default: 0x02

Description: RESERVED

[7:0]	RESERVED : 0x02
-------	-----------------

RESERVED_0xD1

Bit	7	6	5	4	3	2	1	0
Content	RESERVED : 0x40							

Address: 0xD1

Default: 0x40

Description: RESERVED

[7:0]	RESERVED : 0x40
-------	-----------------

RESERVED_0xD2

Bit	7	6	5	4	3	2	1	0
Content	RESERVED : 0x0F							

Address: 0xD2

Default: 0x0F

Description: RESERVED

[7:0]	RESERVED : 0x0F
-------	-----------------

RESERVED_0xD3

Bit	7	6	5	4	3	2	1	0
Content	RESERVED : 0x00							

Address: 0xD3

Default: 0x00

Description: RESERVED

[7:0]	RESERVED : 0x00
-------	-----------------

RESERVED_0xD4

Bit	7	6	5	4	3	2	1	0
Content	RESERVED : 0x32							

Address: 0xD4

Default: 0x32

Description: RESERVED

[7:0]	RESERVED : 0x32
-------	-----------------

RESERVED_0xD5

Bit	7	6	5	4	3	2	1	0
Content	RESERVED : 0x00							

Address: 0xD5

Default: 0x00

Description: RESERVED

[7:0]	RESERVED : 0x00
-------	-----------------

RESERVED_0xD6

Bit	7	6	5	4	3	2	1	0
Content	RESERVED : 0xFC							

Address: 0xD6

Default: 0xFC

Description: RESERVED

[7:0]	RESERVED : 0xFC
-------	-----------------

RESERVED_0xD7

Bit	7	6	5	4	3	2	1	0
Content	RESERVED : 0xF1							

Address: 0xD7

Default: 0xF1

Description: RESERVED

[7:0]	RESERVED : 0xF1
-------	-----------------

RESERVED_0xD8

Bit	7	6	5	4	3	2	1	0
Content	RESERVED : 0x00							

Address: 0xD8

Default: 0x00

Description: RESERVED

[7:0]	RESERVED : 0x00
-------	-----------------

RESERVED_0xD9

Bit	7	6	5	4	3	2	1	0
Content	RESERVED : 0x19							

Address: 0xD9

Default: 0x19

Description: RESERVED

[7:0]	RESERVED : 0x19
-------	-----------------

SNK_PDO_FILL_0xDA

Bit	7	6	5	4	3	2	1	0
Content	LUT_SNK_PDO1_I[3:0]				SNK_UNCONS_POWER	DPM_SNK_PDO_NUMB[1:0]		USB_COMM_CAPABLE

Address: 0xDA

Default: 0x56

Description: SNK_PDO_FILL, part 1 of 11

[7:4]	LUT_SNK_PDO1_I[3:0]
[3]	SNK_UNCONS_POWER
[2:1]	DPM_SNK_PDO_NUMB[1:0]
[0]	USB_COMM_CAPABLE

SNK_PDO_FILL_0xDB

Bit	7	6	5	4	3	2	1	0
Content	SNK_HL1[3:0]				SNK_LL1[3:0]			

Address: 0xDB

Default: 0xAF

Description: SNK_PDO_FILL, part 2 of 11

[7:4]	SNK_HL1[3:0]
[3:0]	SNK_LL1[3:0]

SNK_PDO_FILL_0xDC

Bit	7	6	5	4	3	2	1	0
Content	SNK_LL2[3:0]				LUT_SNK_PDO2_I[3:0]			

Address: 0xDC

Default: 0xF5

Description: SNK_PDO_FILL, part 3 of 11

[7:4]	SNK_LL2[3:0]
[3:0]	LUT_SNK_PDO2_I[3:0]

SNK_PDO_FILL_0xDD

Bit	7	6	5	4	3	2	1	0
Content	LUT_SNK_PDO3_I[3:0]				SNK_HL2[3:0]			

Address: 0xDD

Default: 0x35

Description: SNK_PDO_FILL, part 4 of 11

[7:4]	LUT_SNK_PDO3_I[3:0]
[3:0]	SNK_HL2[3:0]

SNK_PDO_FILL_0xDE

Bit	7	6	5	4	3	2	1	0
Content	SNK_HL3[3:0]				SNK_LL3[3:0]			

Address: 0xDE

Default: 0x5F

Description: SNK_PDO_FILL, part 5 of 11

[7:4]	SNK_HL3[3:0]
[3:0]	SNK_LL3[3:0]

SNK_PDO_FILL_0xDF

Bit	7	6	5	4	3	2	1	0
Content	RESERVED : 0x00							

Address: 0xDF

Default: 0x00

Description: SNK_PDO_FILL, part 6 of 11

[7:0]	RESERVED : SNK_PDO_FILL_0xDF = 0x00
-------	-------------------------------------

SNK_PDO_FILL_0xE0

Bit	7	6	5	4	3	2	1	0
Content	SNK_PDO_FLEX1_V[1:0]		RESERVED : 0b000000					

Address: 0xE0

Default: 0x00

Description: SNK_PDO_FILL, part 7 of 11

[7:6]	SNK_PDO_FLEX1_V[1:0]
[5:0]	RESERVED : 0b000000

SNK_PDO_FILL_0xE1

Bit	7	6	5	4	3	2	1	0
Content	SNK_PDO_FLEX1_V[9:2]							

Address: 0xE1

Default: 0x4B

Description: SNK_PDO_FILL, part 8 of 11

[7:0]	SNK_PDO_FLEX1_V[9:2]
-------	----------------------

SNK_PDO_FILL_0xE2

Bit	7	6	5	4	3	2	1	0
Content	SNK_PDO_FLEX2_V[7:0]							

Address: 0xE2

Default: 0x90

Description: SNK_PDO_FILL, part 9 of 11

[7:0]	SNK_PDO_FLEX2_V[7:0]
-------	----------------------

SNK_PDO_FILL_0xE3

Bit	7	6	5	4	3	2	1	0
Content	SNK_PDO_FLEX_I[5:0]						SNK_PDO_FLEX2_V[9:8]	

Address: 0xE3

Default: 0x21

Description: SNK_PDO_FILL, part 10 of 11

[7:2]	SNK_PDO_FLEX_I[5:0]
[1:0]	SNK_PDO_FLEX2_V[9:8]

SNK_PDO_FILL_0xE4

Bit	7	6	5	4	3	2	1	0
Content	0	POWER_OK_CFG[1:0]		0	SNK_PDO_FLEX_I[9:6]			

Address: 0xE4

Default: 0x43

Description: SNK_PDO_FILL, part 11 of 11

[7]	RESERVED : 0b0
[6:5]	POWER_OK_CFG[1:0]
[4]	RESERVED : 0b0
[3:0]	SNK_PDO_FLEX_I[9:6]

SPARE

Bit	7	6	5	4	3	2	1	0
Content	RESERVED : 0x00							

Address: 0xE5

Default: 0x00

Description: SPARE

[7:0]	RESERVED : SPARE = 0x00
-------	-------------------------

VBUS_CTRL

Bit	7	6	5	4	3	2	1	0
Content	RESERVED : 0b010			REQ_SRC_CURRENT	RESERVED : 0x0			

Address: 0xE6

Default: 0x40

Description: VBUS_CTRL

[7:5]	RESERVED : 0b010
[4]	REQ_SRC_CURRENT
[3:0]	RESERVED : 0x0

ALERT_STATUS_1_MASK

Bit	7	6	5	4	3	2	1	0
Content	RESERVED : 0xFB							

Address: 0xE7

Default: 0xFB

Description: ALERT_STATUS_1_MASK

[7:0]	RESERVED : ALERT_STATUS_1_MASK = 0xFB
-------	---------------------------------------

5 Example

5.1 Using an Aardvark for I2C & GUI

In this example, the following procedure will be described:

- Check default configuration
- Read STUSB4500 NVM map using Aardvark tool & Batch file
- Extract the NVM content
- Change parameters with STSW-STUSB002 GUI
 - Load NVM configuration
 - Modify parameters
 - Save NVM configuration
- Check modified parameters
- Insert NVM content into Aardvark Batch file
- Write STUSB4500 NVM
- Re-Read STUSB4500 NVM map
- Check new configuration

5.1.1 Check default configuration

Connect a TypeC / Power delivery SRC to STUSB4500 and check SNK profiles.

For example, with STUSB4710A Evaluation board (STEVAL-ISC004V1) and associated Software

The screenshot shows the 'Monitoring STUSB4710A Autorun Source' window with the following data:

PDO	Voltage	Current
PDO 1	5V	3.00A
PDO 2	9.00V	3.00A
PDO 3	12.00V	3.00A
PDO 4	15.00V	3.00A
PDO 5	20.00V	2.25A

PDO	Type	V	I	SNK	High Cap
PDO 1	Fixed	V = 5.00V	I = 1.50A	SNK	High Cap: YES
PDO 2	Fixed	V = 15.00V	I = 1.50A	SNK	High Cap: NO
PDO 3	Fixed	V = 20.00V	I = 1.00A	SNK	High Cap: NO

STUSB4710A_-_STUSB4500_capas_Default

5.1.2 Read STUSB4500 NVM map

5.1.2.1 Read batch

```

<aardvark>
  <configure i2c="1"/>
  <i2c_bitrate khz="400"/>
  <i2c_write addr="0x28" radix="16"> 95 47 </i2c_write>
  <i2c_write addr="0x28" radix="16"> 96 40 </i2c_write>
  <i2c_write addr="0x28" radix="16"> 96 00 </i2c_write>
  <sleep ms="1"/>
  <i2c_write addr="0x28" radix="16"> 96 40 </i2c_write>
  <i2c_write addr="0x28" radix="16"> 97 00 </i2c_write>
  <i2c_write addr="0x28" radix="16"> 96 50 </i2c_write>
  <sleep ms="1"/>
  <i2c_write addr="0x28" radix="16"> 53 </i2c_write>
  <i2c_read addr="0x28" count="8"/>
  <i2c_write addr="0x28" radix="16"> 96 51 </i2c_write>
  <sleep ms="1"/>
  <i2c_write addr="0x28" radix="16"> 53 </i2c_write>
  <i2c_read addr="0x28" count="8"/>
  <i2c_write addr="0x28" radix="16"> 96 52 </i2c_write>
  <sleep ms="1"/>
  <i2c_write addr="0x28" radix="16"> 53 </i2c_write>
  <i2c_read addr="0x28" count="8"/>
  <i2c_write addr="0x28" radix="16"> 96 53 </i2c_write>
  <sleep ms="1"/>
  <i2c_write addr="0x28" radix="16"> 53 </i2c_write>
  <i2c_read addr="0x28" count="8"/>
  <i2c_write addr="0x28" radix="16"> 96 54 </i2c_write>
  <sleep ms="1"/>
  <i2c_write addr="0x28" radix="16"> 53 </i2c_write>
  <i2c_read addr="0x28" count="8"/>
  <i2c_write addr="0x28" radix="16"> 96 40 00 </i2c_write>
  <i2c_write addr="0x28" radix="16"> 95 00 </i2c_write>
</aardvark>

```

5.1.2.1 Default Read results

Export Time: 2019-05-07 16:18:01

Port 0

Adapter HW_Version: 3.00 FW_Version: 3.51

```

"Time","Module","Read/Write","Master/Slave","Features","Bitrate","Address","Length","Data"
"2019-05-07 16:17:59.644","","","","","","","","","Configure: I2C=1 SPI=1 GPIO=0"
"2019-05-07 16:17:59.644","","","","","","","","","Power Control Disabled"
"2019-05-07 16:17:59.644","I2C","","","","","","","I2C Pullups Disabled"
"2019-05-07 16:17:59.648","I2C","","","","","","","I2C Bitrate Set to: 400"
"2019-05-07 16:17:59.651","I2C","W","M","----","400","0x28","2","95 47"
"2019-05-07 16:17:59.653","I2C","W","M","----","400","0x28","2","96 40"
"2019-05-07 16:17:59.655","I2C","W","M","----","400","0x28","2","96 00"
"2019-05-07 16:17:59.657","I2C","W","M","----","400","0x28","2","96 40"
"2019-05-07 16:17:59.661","I2C","W","M","----","400","0x28","2","97 00"
"2019-05-07 16:17:59.664","I2C","W","M","----","400","0x28","2","96 50"
"2019-05-07 16:17:59.674","I2C","W","M","----","400","0x28","1","53"
"2019-05-07 16:17:59.684","I2C","R","M","----","400","0x28","8","00 00 B0 AA 00 45 00 00"
"2019-05-07 16:17:59.688","I2C","W","M","----","400","0x28","2","96 51"
"2019-05-07 16:17:59.706","I2C","W","M","----","400","0x28","1","53"
"2019-05-07 16:17:59.710","I2C","R","M","----","400","0x28","8","10 40 9C 1C FF 01 3C DF"
"2019-05-07 16:17:59.713","I2C","W","M","----","400","0x28","2","96 52"
"2019-05-07 16:17:59.726","I2C","W","M","----","400","0x28","1","53"
"2019-05-07 16:17:59.730","I2C","R","M","----","400","0x28","8","02 40 0F 00 32 00 FC F1"
"2019-05-07 16:17:59.734","I2C","W","M","----","400","0x28","2","96 53"
"2019-05-07 16:17:59.752","I2C","W","M","----","400","0x28","1","53"
"2019-05-07 16:17:59.756","I2C","R","M","----","400","0x28","8","00 19 56 AF F5 35 5F 00"
"2019-05-07 16:17:59.760","I2C","W","M","----","400","0x28","2","96 54"

```

```
"2019-05-07 16:17:59.769","I2C","W","M","----","400","0x28","1","53"
"2019-05-07 16:17:59.776","I2C","R","M","----","400","0x28","8","00 4B 90 21 43 00 40 FB"
"2019-05-07 16:17:59.779","I2C","W","M","----","400","0x28","3","96 40 00"
"2019-05-07 16:17:59.782","I2C","W","M","----","400","0x28","2","95 00"
```

5.1.3 Extracting the NVM content

5.1.3.1 Read results

```
"2019-05-07 16:17:59.684","I2C","R","M","----","400","0x28","8","00 00 B0 AA 00 45 00 00"
"2019-05-07 16:17:59.710","I2C","R","M","----","400","0x28","8","10 40 9C 1C FF 01 3C DF"
"2019-05-07 16:17:59.730","I2C","R","M","----","400","0x28","8","02 40 0F 00 32 00 FC F1"
"2019-05-07 16:17:59.756","I2C","R","M","----","400","0x28","8","00 19 56 AF F5 35 5F 00"
"2019-05-07 16:17:59.776","I2C","R","M","----","400","0x28","8","00 4B 90 21 43 00 40 FB"
```

5.1.3.2 Extract data

```
00 00 B0 AA 00 45 00 00
10 40 9C 1C FF 01 3C DF
02 40 0F 00 32 00 FC F1
00 19 56 AF F5 35 5F 00
00 4B 90 21 43 00 40 FB
```

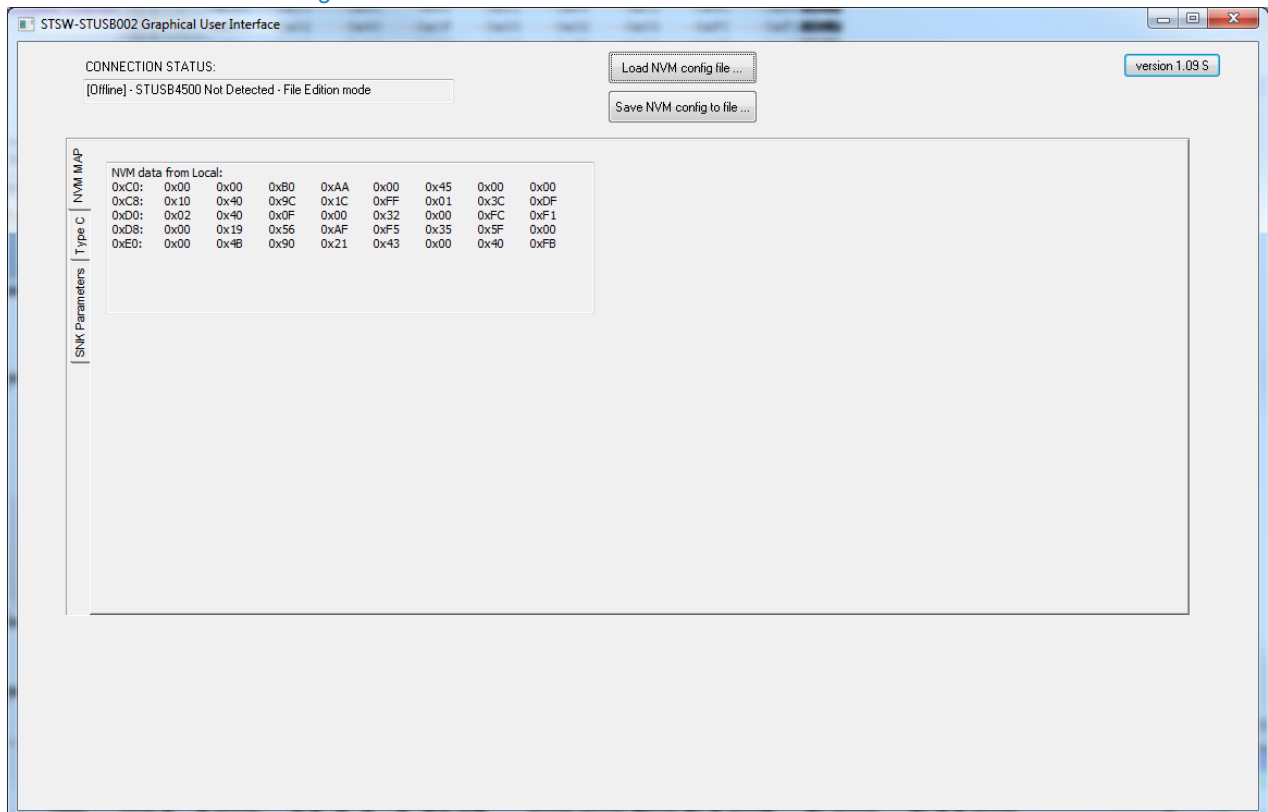
5.1.3.2.1 Convert to GUI File format

```
0xC0: 0x00 0x00 0xB0 0xAA 0x00 0x45 0x00 0x00
0xC8: 0x10 0x40 0x9C 0x1C 0xFF 0x01 0x3C 0xDF
0xD0: 0x02 0x40 0x0F 0x00 0x32 0x00 0xFC 0xF1
0xD8: 0x00 0x19 0x56 0xAF 0xF5 0x35 0x5F 0x00
0xE0: 0x00 0x4B 0x90 0x21 0x43 0x00 0x40 0xFB
```

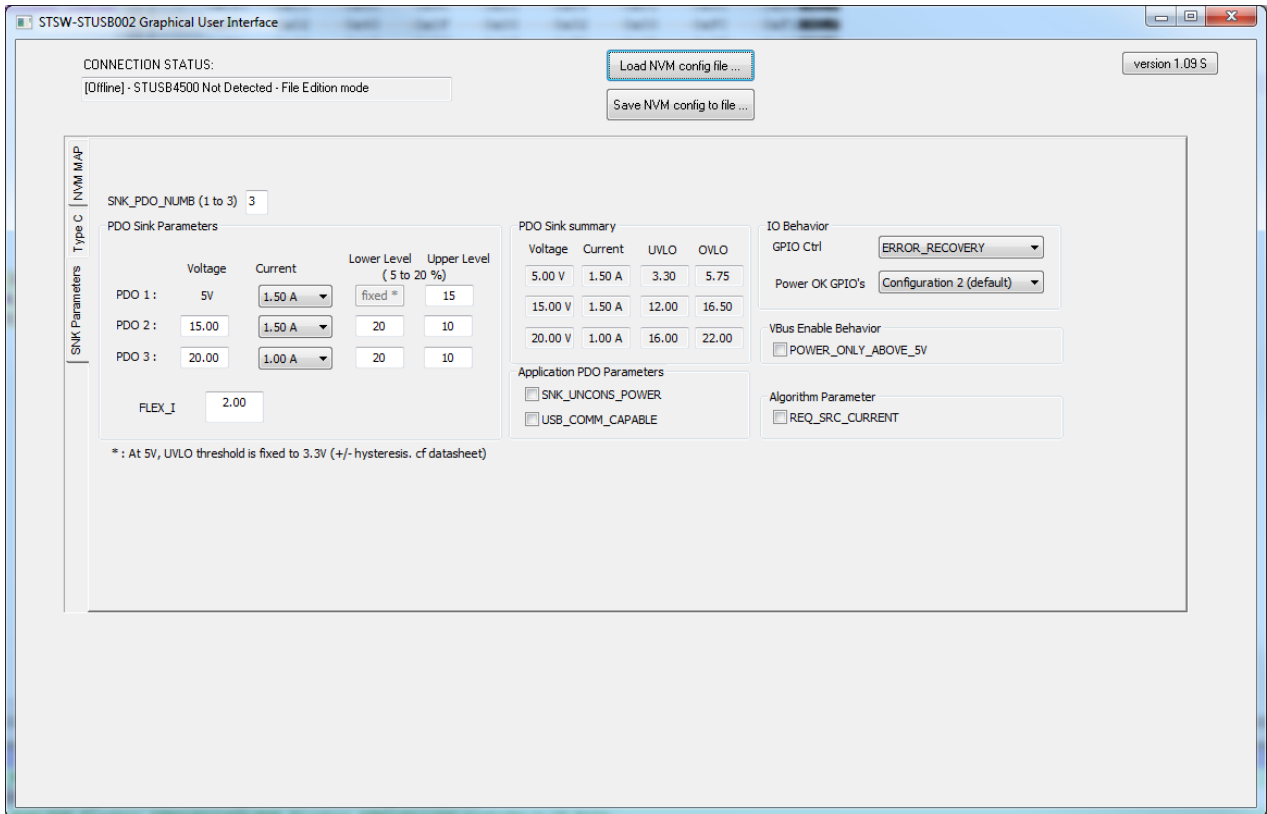
Save this configuration to a file, for example STUSB4500_Default.txt

5.1.4 Change parameters with STSW-STUSB002 GUI

5.1.4.1 Load NVM configuration



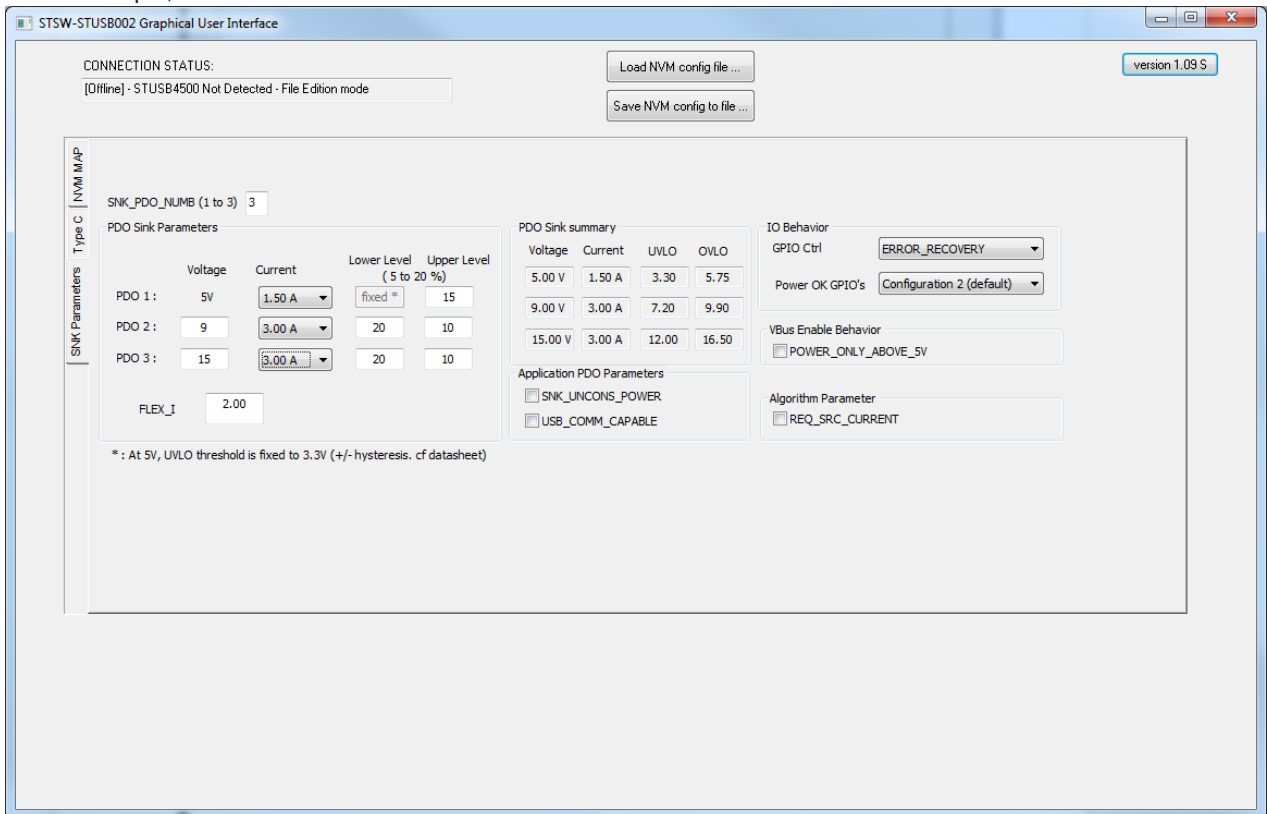
STSW-STUSB002_-_STUSB4500_NVM_Map_Load_After_Aardvark_Read



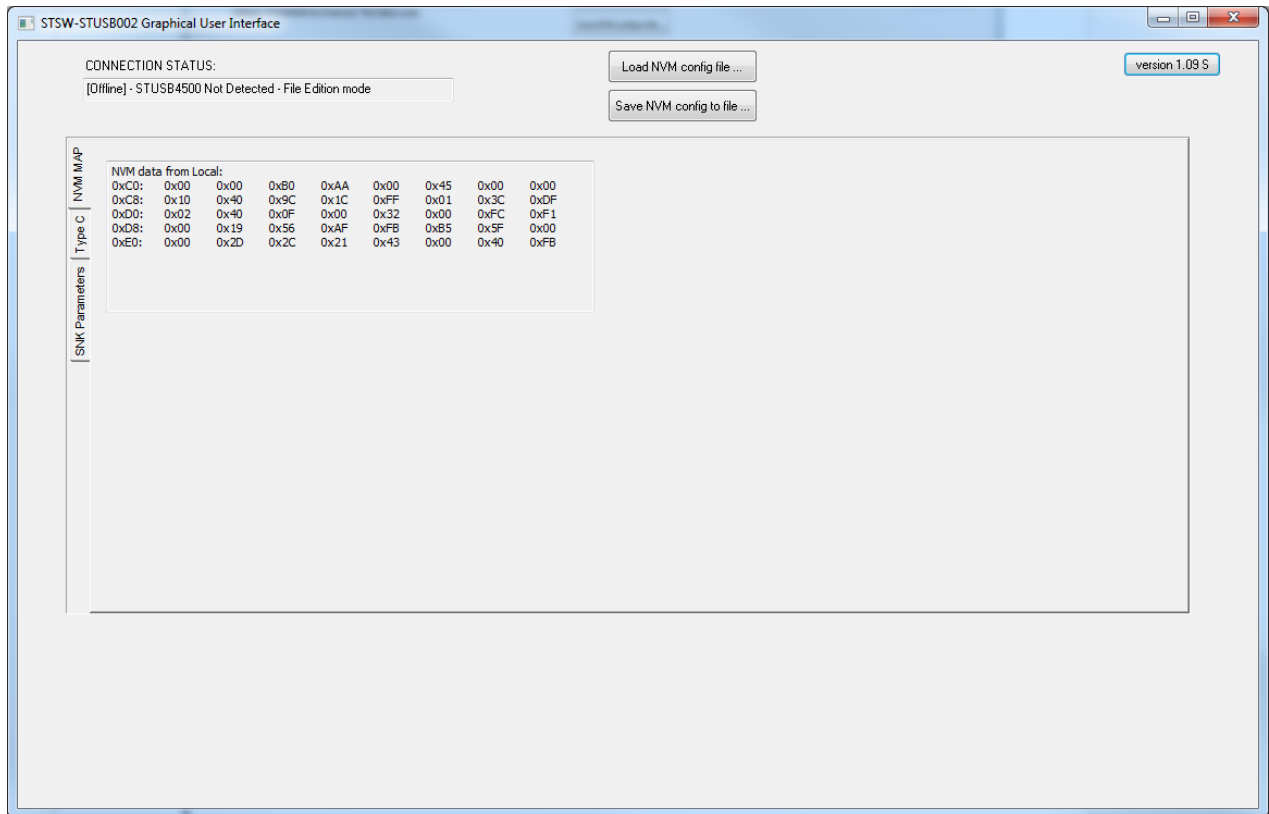
STSW-STUSB002 - _STUSB4500_SNK_Load_After_Aardvark_Read

5.1.4.2 Modify parameters

In this example, we set PDO2 to 9V/3A and PDO3 to 15V/3A



STSW-STUSB002 - _STUSB4500_SNK_Modification



STSW-STUSB002 - _STUSB4500_NVM_Map_Modification

5.1.4.3 Save NVM configuration

With the “Save NVM config to file ...” button, export the modified NVM configuration to a file, for example STUSB4500_5V-1A5_9V-3A_15V-3A.txt

```

0xC0:  0x00  0x00  0xB0  0xAA  0x00  0x45  0x00  0x00
0xC8:  0x10  0x40  0x9C  0x1C  0xF0  0x01  0x00  0xDF
0xD0:  0x02  0x40  0x0F  0x00  0x32  0x00  0xFC  0xF1
0xD8:  0x00  0x19  0x56  0xAF  0xFB  0xB5  0x5F  0x00
0xE0:  0x00  0x2D  0x2C  0x21  0x43  0x00  0x40  0xFB
    
```

5.1.5 Check modified parameters

STUSB4500_Default.txt

```

0xC0:  0x00  0x00  0xB0  0xAA  0x00  0x45  0x00  0x00
0xC8:  0x10  0x40  0x9C  0x1C  0xFF  0x01  0x3C  0xDF
0xD0:  0x02  0x40  0x0F  0x00  0x32  0x00  0xFC  0xF1
0xD8:  0x00  0x19  0x56  0xAF  0xF5  0x35  0x5F  0x00
0xE0:  0x00  0x4B  0x90  0x21  0x43  0x00  0x40  0xFB
    
```

STUSB4500_5V-1A5_9V-3A_15V-3A.txt

```

0xC0:  0x00  0x00  0xB0  0xAA  0x00  0x45  0x00  0x00
0xC8:  0x10  0x40  0x9C  0x1C  0xF0  0x01  0x00  0xDF
0xD0:  0x02  0x40  0x0F  0x00  0x32  0x00  0xFC  0xF1
0xD8:  0x00  0x19  0x56  0xAF  0xFB  0xB5  0x5F  0x00
0xE0:  0x00  0x2D  0x2C  0x21  0x43  0x00  0x40  0xFB
    
```

Différences :

- Address 0xCC
 - STUSB4500_Default : 0xFF
 - STUSB4500_5V-1A5_9V-3A_15V-3A : 0xF0
 - Not taken into account : GUI side effect
- Address 0xCE
 - STUSB4500_Default : 0x3C
 - STUSB4500_5V-1A5_9V-3A_15V-3A : 0x00
 - Not taken into account : GUI side effect
- Address 0xDC 0xDD
 - STUSB4500_Default : 0xF5 0x35
 - STUSB4500_5V-1A5_9V-3A_15V-3A : 0xFB 0xB5
 - This corresponds to PDO currents modification
 - STUSB4500_Default : LUT_SNK_PDO2_I = 0b0101 =
 - STUSB4500_Default : LUT_SNK_PDO3_I = 0b0011 =
 - STUSB4500_5V-1A5_9V-3A_15V-3A : LUT_SNK_PDO2_I = 0b1011 =
 - STUSB4500_5V-1A5_9V-3A_15V-3A : LUT_SNK_PDO3_I = 0b1011 =
- Address 0xE0 to 0xE4
 - STUSB4500_Default : 0x00 0x4B 0x90 0x21 0x43
 - STUSB4500_5V-1A5_9V-3A_15V-3A : 0x00 0x2D 0x2C 0x21 0x43
 - This corresponds to PDO voltages modification
 - STUSB4500_Default :
 - SNK_PDO_FLEX1_V[1:0] = 0b00
 - SNK_PDO_FLEX1_V[9:2] = 0x4B = 0b01001011
 - SNK_PDO_FLEX1_V = 0b0100101100 = 300 = 15V
 - STUSB4500_Default :
 - SNK_PDO_FLEX2_V[7:0] = 0x90 = 0b10010000
 - SNK_PDO_FLEX2_V[9:8] = 0b01
 - SNK_PDO_FLEX2_V = 0b0110010000 = 400 = 20V
 - STUSB4500_5V-1A5_9V-3A_15V-3A :
 - SNK_PDO_FLEX1_V[1:0] = 0b00
 - SNK_PDO_FLEX1_V[9:2] = 0x2D = 0b00101101
 - SNK_PDO_FLEX1_V = 0b0010110100 = 180 = 9V
 - STUSB4500_5V-1A5_9V-3A_15V-3A:
 - SNK_PDO_FLEX2_V[7:0] = 0x2C = 0b00101100
 - SNK_PDO_FLEX2_V[9:8] = 0b01
 - SNK_PDO_FLEX2_V = 0b0100101100 = 300 = 15V

5.1.6 Insert NVM content into Aardvark Batch file

5.1.6.1 Extract data

```

00 00 B0 AA 00 45 00 00
10 40 9C 1C F0 01 00 DF
02 40 0F 00 32 00 FC F1
00 19 56 AF FB B5 5F 00
00 2D 2C 21 43 00 40 FB
  
```

5.1.6.1 Write batch

```

<aardvark>
  <configure i2c="1"/>
  <i2c_bitrate khz="400"/>
  <i2c_write addr="0x28" radix="16"> 95 47 </i2c_write>
  <i2c_write addr="0x28" radix="16"> 53 00 </i2c_write>
  <i2c_write addr="0x28" radix="16"> 96 40 </i2c_write>
  <i2c_write addr="0x28" radix="16"> 96 00 </i2c_write>
  <sleep ms="1"/>
  <i2c_write addr="0x28" radix="16"> 96 40 </i2c_write>
  <i2c_write addr="0x28" radix="16"> 97 FA </i2c_write>
  <i2c_write addr="0x28" radix="16"> 96 50 </i2c_write>
  <sleep ms="1"/>
  <i2c_write addr="0x28" radix="16"> 97 07 </i2c_write>
  <i2c_write addr="0x28" radix="16"> 96 50 </i2c_write>
  <sleep ms="5"/>
  <i2c_write addr="0x28" radix="16"> 97 05 </i2c_write>
  <i2c_write addr="0x28" radix="16"> 96 50 </i2c_write>
  <sleep ms="5"/>
  <i2c_write addr="0x28" radix="16"> 53 00 00 B0 AA 00 45 00 00 </i2c_write>
  <sleep ms="1"/>
  <i2c_write addr="0x28" radix="16"> 97 01 </i2c_write>
  <i2c_write addr="0x28" radix="16"> 96 50 </i2c_write>
  <sleep ms="1"/>
  <i2c_write addr="0x28" radix="16"> 97 06 </i2c_write>
  <i2c_write addr="0x28" radix="16"> 96 50 </i2c_write>
  <sleep ms="2"/>
  <i2c_write addr="0x28" radix="16"> 53 10 40 9C 1C F0 01 00 DF </i2c_write>
  <sleep ms="1"/>
  <i2c_write addr="0x28" radix="16"> 97 01 </i2c_write>
  <i2c_write addr="0x28" radix="16"> 96 50 </i2c_write>
  <sleep ms="1"/>
  <i2c_write addr="0x28" radix="16"> 97 06 </i2c_write>
  <i2c_write addr="0x28" radix="16"> 96 51 </i2c_write>
  <sleep ms="2"/>
  <i2c_write addr="0x28" radix="16"> 53 02 40 0F 00 32 00 FC F1 </i2c_write>
  <sleep ms="1"/>
  <i2c_write addr="0x28" radix="16"> 97 01 </i2c_write>
  <i2c_write addr="0x28" radix="16"> 96 50 </i2c_write>
  <sleep ms="1"/>
  <i2c_write addr="0x28" radix="16"> 97 06 </i2c_write>
  <i2c_write addr="0x28" radix="16"> 96 52 </i2c_write>
  <sleep ms="2"/>
  <i2c_write addr="0x28" radix="16"> 53 00 19 56 AF FB B5 5F 00 </i2c_write>
  <sleep ms="1"/>
  <i2c_write addr="0x28" radix="16"> 97 01 </i2c_write>
  <i2c_write addr="0x28" radix="16"> 96 50 </i2c_write>
  <sleep ms="1"/>
  <i2c_write addr="0x28" radix="16"> 97 06 </i2c_write>
  <i2c_write addr="0x28" radix="16"> 96 53 </i2c_write>
  <sleep ms="2"/>
  <i2c_write addr="0x28" radix="16"> 53 00 2D 2C 21 43 00 40 FB </i2c_write>
  <sleep ms="1"/>
  <i2c_write addr="0x28" radix="16"> 97 01 </i2c_write>
  <i2c_write addr="0x28" radix="16"> 96 50 </i2c_write>
  <sleep ms="1"/>
  <i2c_write addr="0x28" radix="16"> 97 06 </i2c_write>
  <i2c_write addr="0x28" radix="16"> 96 54 </i2c_write>
  <sleep ms="2"/>
  <i2c_write addr="0x28" radix="16"> 96 40 00 </i2c_write>
  <i2c_write addr="0x28" radix="16"> 95 47 </i2c_write>
</aardvark>

```

5.1.6.1 Write results

Export Time: 2019-05-07 17:03:06

Port 0

Adapter HW_Version: 3.00 FW_Version: 3.51

```
"Time","Module","Read/Write","Master/Slave","Features","Bitrate","Address","Length","Data"
"2019-05-07 17:03:04.957","","","","","","","","","Configure: I2C=1 SPI=1 GPIO=0"
"2019-05-07 17:03:04.957","","","","","","","","","Power Control Disabled"
"2019-05-07 17:03:04.959","I2C","","","","","","","","I2C Pullups Disabled"
"2019-05-07 17:03:04.963","I2C","","","","","","","","I2C Bitrate Set to: 400"
"2019-05-07 17:03:04.966","I2C","W","M","----","400","0x28","2","95 47"
"2019-05-07 17:03:04.967","I2C","W","M","----","400","0x28","2","53 00"
"2019-05-07 17:03:04.970","I2C","W","M","----","400","0x28","2","96 40"
"2019-05-07 17:03:04.973","I2C","W","M","----","400","0x28","2","96 00"
"2019-05-07 17:03:04.994","I2C","W","M","----","400","0x28","2","96 40"
"2019-05-07 17:03:05.000","I2C","W","M","----","400","0x28","2","97 FA"
"2019-05-07 17:03:05.002","I2C","W","M","----","400","0x28","2","96 50"
"2019-05-07 17:03:05.020","I2C","W","M","----","400","0x28","2","97 07"
"2019-05-07 17:03:05.029","I2C","W","M","----","400","0x28","2","96 50"
"2019-05-07 17:03:05.055","I2C","W","M","----","400","0x28","2","97 05"
"2019-05-07 17:03:05.059","I2C","W","M","----","400","0x28","2","96 50"
"2019-05-07 17:03:05.068","I2C","W","M","----","400","0x28","9","53 00 00 B0 AA 00 45 00 00"
"2019-05-07 17:03:05.082","I2C","W","M","----","400","0x28","2","97 01"
"2019-05-07 17:03:05.085","I2C","W","M","----","400","0x28","2","96 50"
"2019-05-07 17:03:05.098","I2C","W","M","----","400","0x28","2","97 06"
"2019-05-07 17:03:05.105","I2C","W","M","----","400","0x28","2","96 50"
"2019-05-07 17:03:05.117","I2C","W","M","----","400","0x28","9","53 10 40 9C 1C F0 01 00 DF"
"2019-05-07 17:03:05.131","I2C","W","M","----","400","0x28","2","97 01"
"2019-05-07 17:03:05.135","I2C","W","M","----","400","0x28","2","96 50"
"2019-05-07 17:03:05.145","I2C","W","M","----","400","0x28","2","97 06"
"2019-05-07 17:03:05.150","I2C","W","M","----","400","0x28","2","96 51"
"2019-05-07 17:03:05.161","I2C","W","M","----","400","0x28","9","53 02 40 0F 00 32 00 FC F1"
"2019-05-07 17:03:05.176","I2C","W","M","----","400","0x28","2","97 01"
"2019-05-07 17:03:05.180","I2C","W","M","----","400","0x28","2","96 50"
"2019-05-07 17:03:05.194","I2C","W","M","----","400","0x28","2","97 06"
"2019-05-07 17:03:05.198","I2C","W","M","----","400","0x28","2","96 52"
"2019-05-07 17:03:05.207","I2C","W","M","----","400","0x28","9","53 00 19 56 AF FB B5 5F 00"
"2019-05-07 17:03:05.223","I2C","W","M","----","400","0x28","2","97 01"
"2019-05-07 17:03:05.230","I2C","W","M","----","400","0x28","2","96 50"
"2019-05-07 17:03:05.240","I2C","W","M","----","400","0x28","2","97 06"
"2019-05-07 17:03:05.246","I2C","W","M","----","400","0x28","2","96 53"
"2019-05-07 17:03:05.253","I2C","W","M","----","400","0x28","9","53 00 2D 2C 21 43 00 40 FB"
"2019-05-07 17:03:05.273","I2C","W","M","----","400","0x28","2","97 01"
"2019-05-07 17:03:05.278","I2C","W","M","----","400","0x28","2","96 50"
"2019-05-07 17:03:05.286","I2C","W","M","----","400","0x28","2","97 06"
"2019-05-07 17:03:05.290","I2C","W","M","----","400","0x28","2","96 54"
"2019-05-07 17:03:05.305","I2C","W","M","----","400","0x28","3","96 40 00"
"2019-05-07 17:03:05.311","I2C","W","M","----","400","0x28","2","95 47"
```

5.1.7 Re-Read STUSB4500 NVM map

Run the same read batch as first step.

5.1.7.1 Read results

Export Time: 2019-05-09 11:43:42

Port 0

Adapter HW_Version: 3.00 FW_Version: 3.51

```

"Time","Module","Read/Write","Master/Slave","Features","Bitrate","Address","Length","Data"
"2019-05-09 11:43:40.588","","","","","","","","","Configure: I2C=1 SPI=1 GPIO=0"
"2019-05-09 11:43:40.588","","","","","","","","","Power Control Disabled"
"2019-05-09 11:43:40.591","I2C","","","","","","","","I2C Pullups Disabled"
"2019-05-09 11:43:40.595","I2C","","","","","","","","I2C Bitrate Set to: 400"
"2019-05-09 11:43:40.598","I2C","W","M","----","400","0x28","2","95 47"
"2019-05-09 11:43:40.599","I2C","W","M","----","400","0x28","2","96 40"
"2019-05-09 11:43:40.601","I2C","W","M","----","400","0x28","2","96 00"
"2019-05-09 11:43:40.621","I2C","W","M","----","400","0x28","2","96 40"
"2019-05-09 11:43:40.624","I2C","W","M","----","400","0x28","2","97 00"
"2019-05-09 11:43:40.625","I2C","W","M","----","400","0x28","2","96 50"
"2019-05-09 11:43:40.634","I2C","W","M","----","400","0x28","1","53"
"2019-05-09 11:43:40.644","I2C","R","M","----","400","0x28","8","00 00 B0 AA 00 45 00 00"
"2019-05-09 11:43:40.648","I2C","W","M","----","400","0x28","2","96 51"
"2019-05-09 11:43:40.671","I2C","W","M","----","400","0x28","1","53"
"2019-05-09 11:43:40.674","I2C","R","M","----","400","0x28","8","10 40 9C 1C F0 01 00 DF"
"2019-05-09 11:43:40.677","I2C","W","M","----","400","0x28","2","96 52"
"2019-05-09 11:43:40.694","I2C","W","M","----","400","0x28","1","53"
"2019-05-09 11:43:40.700","I2C","R","M","----","400","0x28","8","02 40 0F 00 32 00 FC F1"
"2019-05-09 11:43:40.703","I2C","W","M","----","400","0x28","2","96 53"
"2019-05-09 11:43:40.714","I2C","W","M","----","400","0x28","1","53"
"2019-05-09 11:43:40.720","I2C","R","M","----","400","0x28","8","00 19 56 AF FB B5 5F 00"
"2019-05-09 11:43:40.723","I2C","W","M","----","400","0x28","2","96 54"
"2019-05-09 11:43:40.746","I2C","W","M","----","400","0x28","1","53"
"2019-05-09 11:43:40.750","I2C","R","M","----","400","0x28","8","00 2D 2C 21 43 00 40 FB"
"2019-05-09 11:43:40.753","I2C","W","M","----","400","0x28","3","96 40 00"
"2019-05-09 11:43:40.756","I2C","W","M","----","400","0x28","2","95 00"

```

5.1.7.1 Read results comparison

```

"2019-05-07 16:17:59.684","I2C","R","M","----","400","0x28","8","00 00 B0 AA 00 45 00 00"
"2019-05-07 16:17:59.710","I2C","R","M","----","400","0x28","8","10 40 9C 1C FF 01 3C DF"
"2019-05-07 16:17:59.730","I2C","R","M","----","400","0x28","8","02 40 0F 00 32 00 FC F1"
"2019-05-07 16:17:59.756","I2C","R","M","----","400","0x28","8","00 19 56 AF F5 35 5F 00"
"2019-05-07 16:17:59.776","I2C","R","M","----","400","0x28","8","00 4B 90 21 43 00 40 FB"

"2019-05-09 11:43:40.644","I2C","R","M","----","400","0x28","8","00 00 B0 AA 00 45 00 00"
"2019-05-09 11:43:40.674","I2C","R","M","----","400","0x28","8","10 40 9C 1C F0 01 00 DF"
"2019-05-09 11:43:40.700","I2C","R","M","----","400","0x28","8","02 40 0F 00 32 00 FC F1"
"2019-05-09 11:43:40.720","I2C","R","M","----","400","0x28","8","00 19 56 AF FB B5 5F 00"
"2019-05-09 11:43:40.750","I2C","R","M","----","400","0x28","8","00 2D 2C 21 43 00 40 FB"

```

5.1.8 Check new configuration

Monitoring STUSB4710A Autorun Source

CC Connection

Sink attached

Operating as a source

VCONN is not supplied on CC pin

CC Operation

CC2 is attached

Source supplies 3.0 A USB Type-C current

Monitoring

VBUS is within valid voltage range

VBUS is above VSafe0V threshold

VBUS is above UWLO threshold

Hardware Fault Status

No hardware fault detected

Source Capabilities

	Voltage	Current
PDO 1:	5V	3.00A
PDO 2:	9.00V	3.00A
PDO 3:	12.00V	3.00A
PDO 4:	15.00V	3.00A
PDO 5:	20.00V	2.25A

Attached device capabilities

PDO	Type	V	I	SNK	High Cap
PDO 1:	Fixed	V = 5.00V	I = 1.50A	SNK	High Cap: YES
PDO 2:	Fixed	V = 9.00V	I = 3.00A	SNK	High Cap: NO
PDO 3:	Fixed	V = 15.00V	I = 3.00A	SNK	High Cap: NO

Contract

Explicit contract: PDO 4

STUSB4710A_-_STUSB4500_capas_After_Write

IMPORTANT NOTICE – PLEASE READ CAREFULLY

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

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

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

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product. ST and the ST logo are trademarks of ST. All other product or service names are the property of their respective owners.

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

© 2018 STMicroelectronics – All rights reserved