XDPP1100 Firmware User Guide

Getting Started in XDPP1100 Custom Firmware Development

About this document

Scope and purpose

This document shows on how to develop custom firmware on XDPP1100 to extend its capabilities beyond the default firmware.

Intended audience

Firmware engineers, Switched Mode Power Supplies (SMPS) engineers

Table of contents

1 Introduction

The XDPP1100 is a digital power supply controller offering superior levels of integration and performance in a single-chip solution. The flexible nature of the IC makes it suitable for a wide variety of power conversion application. Multiple peripherals inside the device have been specifically optimized to enhance the performance of isolated DC-DC applications and reduce the solution component count in the IT and network infrastructure space.

At the core of the XDPP1100 controller are the digital control loop peripherals. Each implements a high-speed digital control loop consisting of a dedicated voltage Analog-to-Digital Converter (ADC), a high resolution current ADC, a PID-based digital compensator, and DPWM outputs with 78.125-ps pulse width resolution. The device also offers 6 channels of 9-bit, 1Msps general purpose ADC, timers, interrupt control, PMBus, and two I2C communications ports.

The device is based on a 32-bit, 100MHz ARM® Cortex™-M0 RISC microcontroller that performs real-time monitoring, configures peripherals, and manages communications. The ARM® microcontroller executes its program out of programmable OTP as well as on-chip RAM and ROM.

The default firmware in XDPP1100 comes with vast list of features such as support for multiple power topologies commonly found in DC/DC server/telecom bricks power conversion, built-in PMBus commands, voltage mode control (VMC), Peak Current Mode Control (PCMC), and many others.

The default firmware made several assumptions on the system with the good intention for minimum programming effort at the system level. However, it is still possible to develop custom firmware for XDPP1100 to fulfill different system requirements as well as for proprietary control algorithms.

2 Firmware Architecture

2.1 Block Diagram

XDPP1100 firmware architecture and the associated files locations are shown in the following diagram.

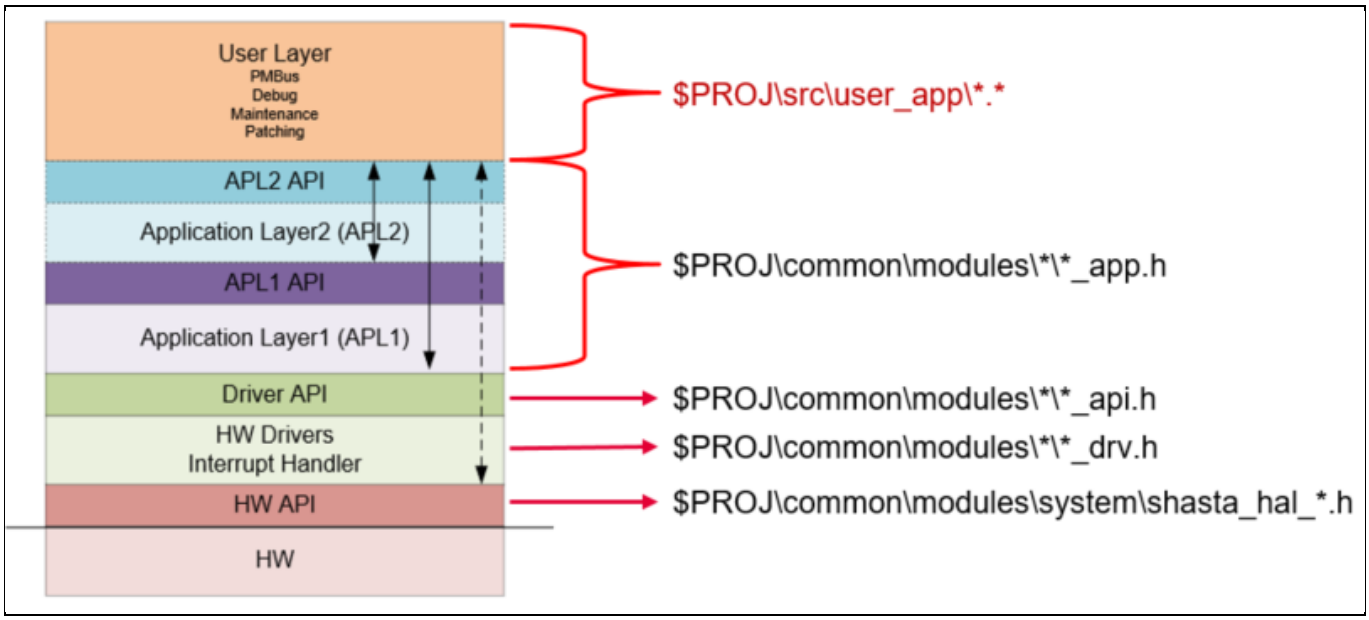


Figure 1 XDPP1100 Firmware Architecture Block Diagram

Each layer can be described as following:

- User Layer:

  This is the top-most layer containing all functions related to user interaction. This level contains functions which can be provided by user and resides in an OTP for example. These functions define how the user experiences the interaction with XDPP1100 via different interfaces like PMBus, SPI and UART. These interfaces will be used for controlling the device in different operational situations like DC/DC power control, configuration, debugging, maintenance or patching.

- Application Layer 2:

  This is a higher-level application layer of the functions defining an applications that are too complex to put them in a single layer. This layer can be thought of as the extension of Application Layer 1 and optional.

- Application Layer 1:

  This is a lower-level application layer of the functions that are defining simple applications.

- HW Layer:

  This is the lowest-level layer defining a direct interaction to the underlying hardware peripherals, including interrupt service routines.

2.2 Custom Firmware Use Cases

Most of custom firmware developments for XDPP1100 can be narrowed down on to several use cases, as shown in the following table. In addition, the relevant function in which such use cases can be implemented are shown as a guide.

Table 1 Custom Firmware Use Cases

Use case	Implementation
One-time-off function initialization	User_drv_init() Add_on_features()
Peripheral initialization	
Other initialization functions	
Periodic functions	Regulation_sm_callbacks_init()
Customized/proprietary control loop algorithms	
Customized/proprietary PMBus MFR commands	Patch_pmbus_mfr_autogen_init()

2.3 Simplified Startup sequences

As XDPP1100 default firmware is integrated into the ROM, it is usually not possible to perform any firmware modification. The default firmware has anticipated this issue by providing a mechanism to jump to the OTP memory for a new firmware, i.e. "**patch_entry()**" function. This function is analoguous to "**main()**" function found in standard microcontrollers.

To simplify the customized firmware development workflow, it is worth to take note on the XDPP1100 startup sequence which are shown in the following diagram. In addition, the relevant file for each flowchart block is indicated.

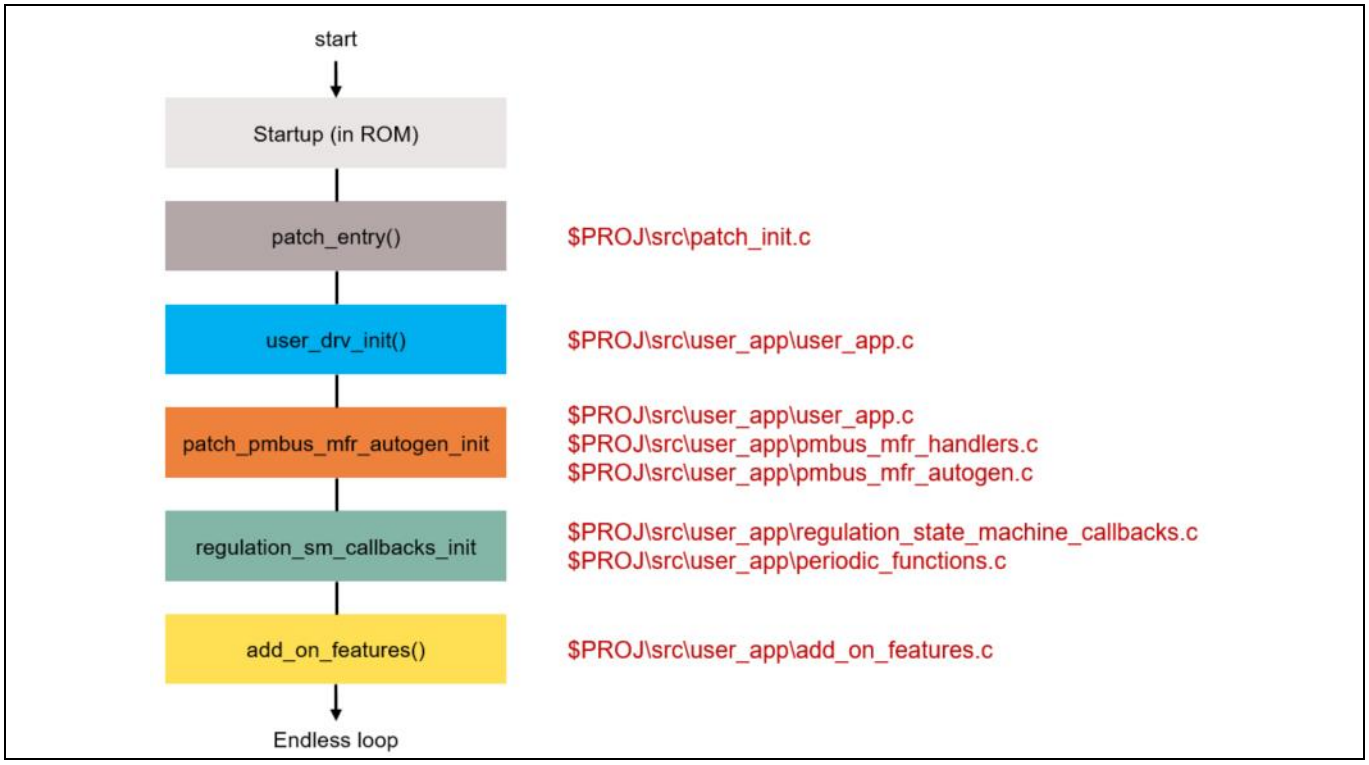


Figure 2 **Simplified XDPP1100 Startup Sequence**

2.4 Regulation State Machines

All the power control and regulation is governed by a firmware state machine as shown in the following diagram.

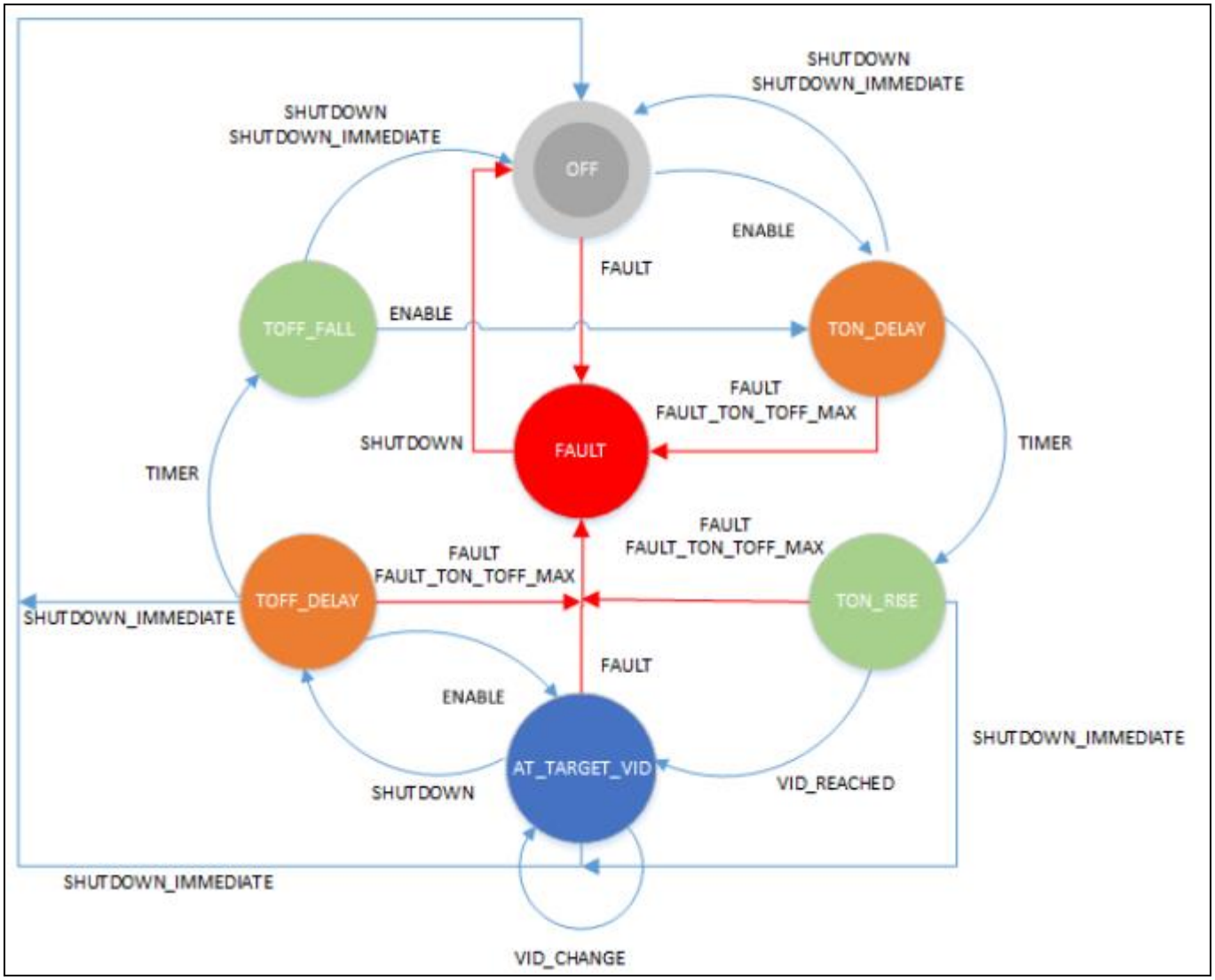


Figure 3 Regulation State Machines Diagram

All the states, control signals and the callback functions can be found in “regulation_sm_callbacks_init()” function in “regulation_state_machine_callbacks.c”. This is the place in which custom/proprietary control loop code can be “registered” in the state machine.

regulation_state_machine_callbacks.c

```
64 void regulation_sm_callbacks_init(void)
65 {
66     //Regulation_set_regulation_event_cb(REGULATION_STATE_OFF, REGULATION_CONTROLS_SHUTDOWN, pwm_boot_cap_startup);
67
68     // Set the callback functions for the regulation state machine.
69     // ------------------------------------------------------------------------------------------------------------
70     #ifdef retry_if_removed_bug_fix
71     sys_irq_handler_set((IRQn_Type) VC0_VCONTROL_IRQn,
72             (Irq_Handler) patch_Regulation_VC0_VCONTROL_IRQHandler); /* Put interrupt handler into vector table */
73     #endif
74
75     Regulation_set_regulation_event_cb(REGULATION_STATE_AT_TARGET_VID, REGULATION_CONTROLS_ENABLE,
76             AT_TARGET_ENABLE);
77
78     Regulation_set_regulation_event_cb(REGULATION_STATE_TOFF_FALL, REGULATION_CONTROLS_SHUTDOWN,
79             frequency_dither_disable);
80
81     Regulation_set_regulation_event_cb(REGULATION_STATE_TOFF_FALL, REGULATION_CONTROLS_SHUTDOWN_IMMEDIATE,
82             frequency_dither_disable);
83
84     Regulation_set_regulation_event_cb(REGULATION_STATE_TON_RISE, REGULATION_CONTROLS_ENABLE,
85             TON_RISE_ENABLE);
86
87     Regulation_set_regulation_event_cb(REGULATION_STATE_TON_DELAY, REGULATION_CONTROLS_TON_TOFF_DELAY_TIMER,
88             ramp_switching_frequency_pcmc_startup_enable);
89
90     #ifdef en_ishare
91     Regulation_set_regulation_event_cb(REGULATION_STATE_OFF, REGULATION_CONTROLS_SHUTDOWN,
92             AT_SHUTDOWN);
93
94     Regulation_set_regulation_event_cb(REGULATION_STATE_OFF, REGULATION_CONTROLS_SHUTDOWN_IMMEDIATE,
95             AT_SHUTDOWN);
96     #endif
97
98     Regulation_set_regulation_event_cb(REGULATION_STATE_TON_RISE, REGULATION_CONTROLS_VID_REACHED,
99             TON_RISE_VID_REACHED);
100
101    // ------------------------------------------------------------------------------------------------------------
```

Figure 4 Regulation State Machine Callbacks screenshot

3 Setting Up XDPP1100 Development Environment

This section describes the XDPP1100 development environment setup and required tools.

3.1 STEP 1: Download XDPP1100 Installation Package.

All XDPP1100 materials can be found in Infineon MyICP. You need to sign-up for access.

Go to https://myicp.infineon.com/sites/power_management/default.aspx.

Once signed in, you will see the following page:

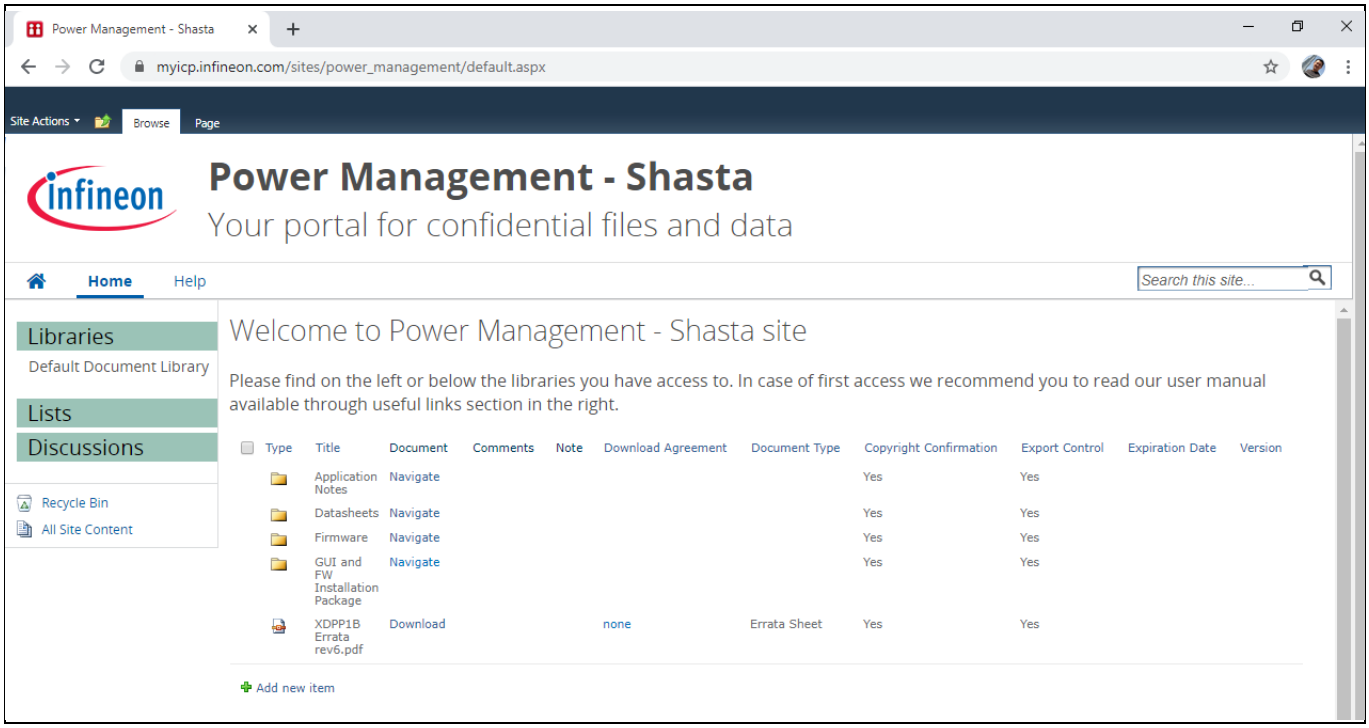
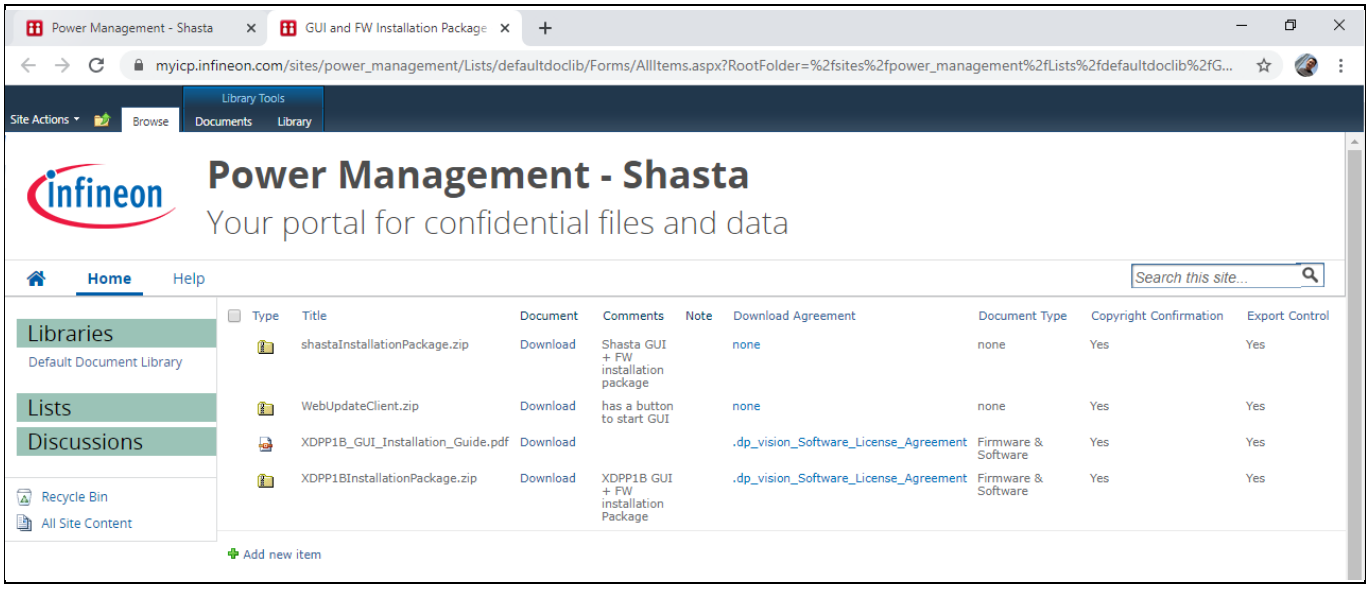


Figure 5 MyICP screenshot

Navigate to “GUI and FW Installation Package”. You will see the following page:



Download the “XDPP1BInstallationPackage.zip” and follow the installation wizard.

3.2 STEP 2: Download Softwares and Development Tools

All the required software and development tools are shown in the following table. Please install them before proceed on to the next steps.

Table 2 Required Software and Tools for XDPP1100 firmware development

Required Software	Download Link
Java	https://www.java.com/en/download/windows_ie.jsp
Python 2.7	https://www.python.org/downloads/release/python-2710/
Eclipse C/C++	http://www.eclipse.org/downloads/packages/eclipse-ide-cc-developers/neon3

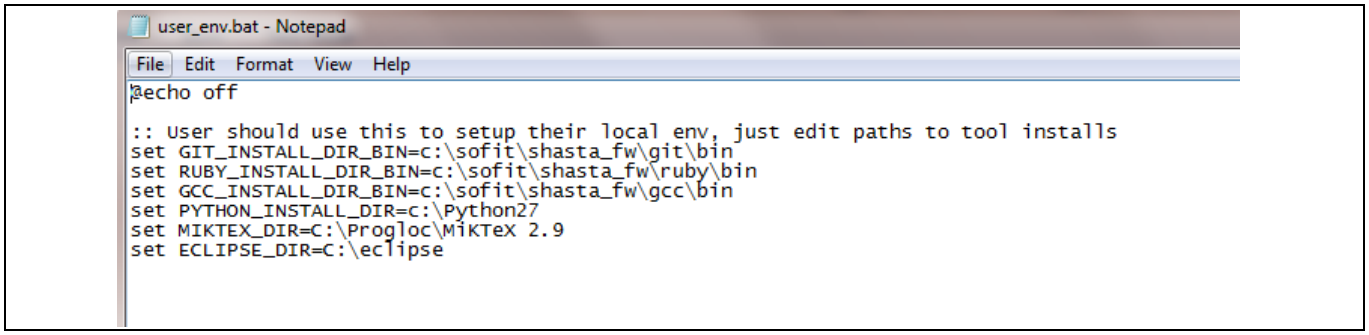
[OPTIONAL] The following tools are optional and not compulsory to get started in firmware development.

Table 3 [OPTIONAL] Required Software and Tools for XDPP1100 firmwware development

Optional Software	Download Link
Keil ARM CC	https://developer.arm.com/products/software-development-tools/license-management
Lauterbach Trace32 debugger	http://www.lauterbach.com/frames.html?microtrace.html

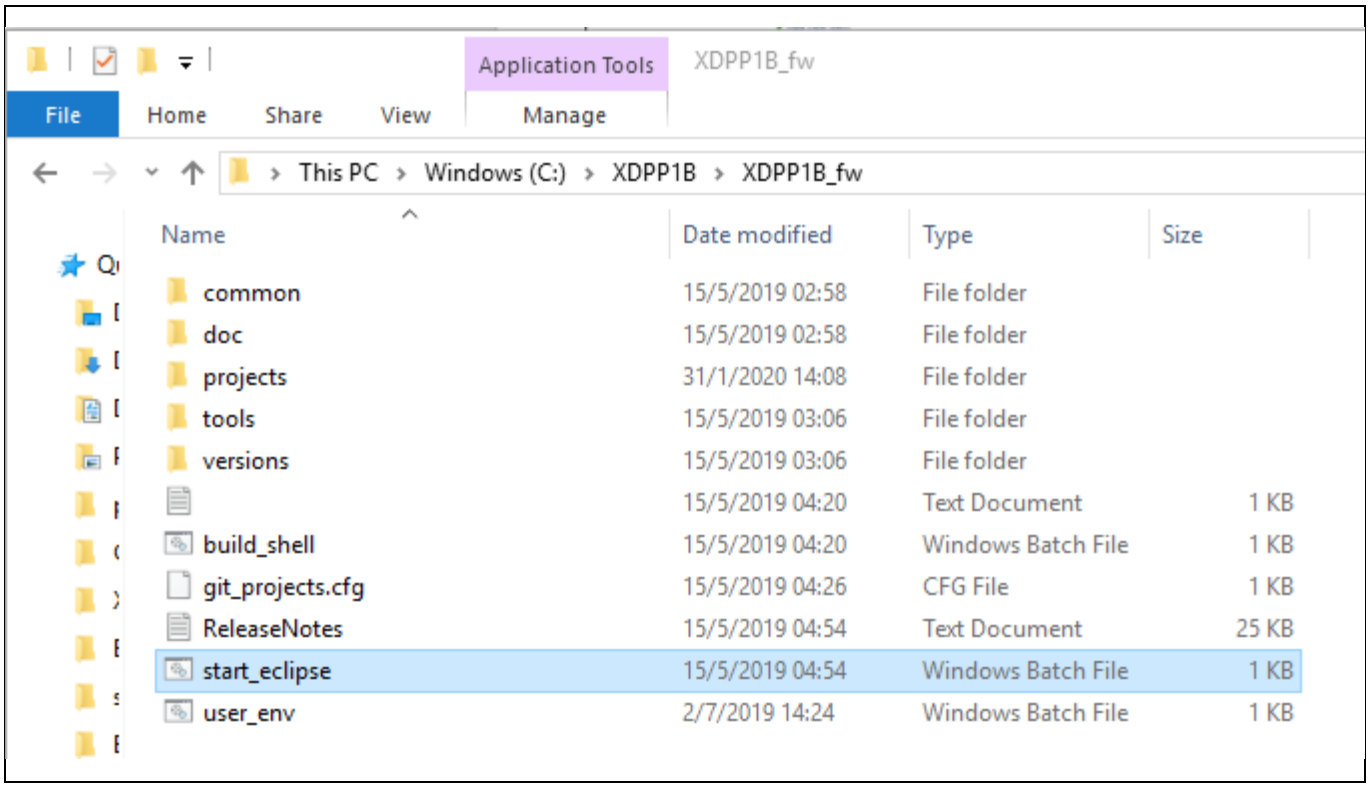
3.3 STEP 3: Setup Eclipse User Environment Paths

Go to C:\XDPP1B\XDPP1B_fw\user_env.bat and make sure the paths are shown like the following figure:



3.4 STEP 4: Start Eclipse

Start Eclipse by double clicking on C:\XDPP1B\XDPP1B_fw\start_eclipse.bat. Give name to your workspace and you are good to work on the firmware.



4 Creating a New Project

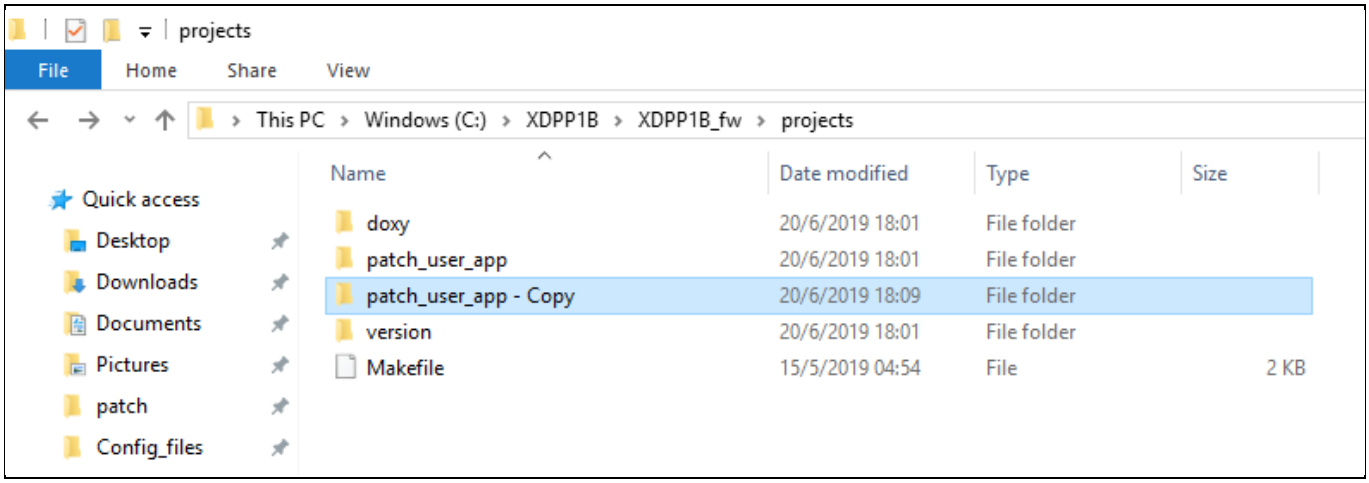
The default firmware of XDPP1100 is hardcoded in ROM. Unlike traditional Flash-based microcontroller, it is not possible to completely erase and reprogram the XDPP1100. For this reason, XDPP1100 comes with OTP “One Time Programmable” memory which allows certain functionalities in XDPP100 to be modified, i.e. “patched”.

Given the above context, XDPP1100 firmware is also known as Firmware Patch. This section shows on how to create a new patch project.

4.1 STEP 1: Duplicate patch_user_app project

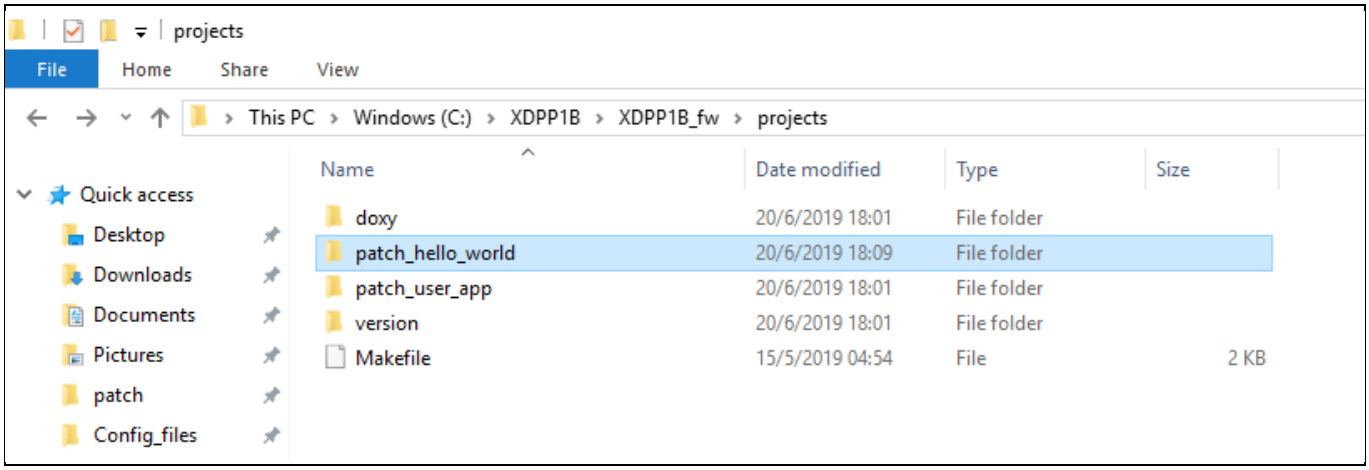
Duplicate the “patch_user_app” folder. Select patch_user_app folder and then use the keyboard shortcut “Ctrl + C” followed by “Ctrl + V”.

You should see “patch_user_app – Copy” folder generated.



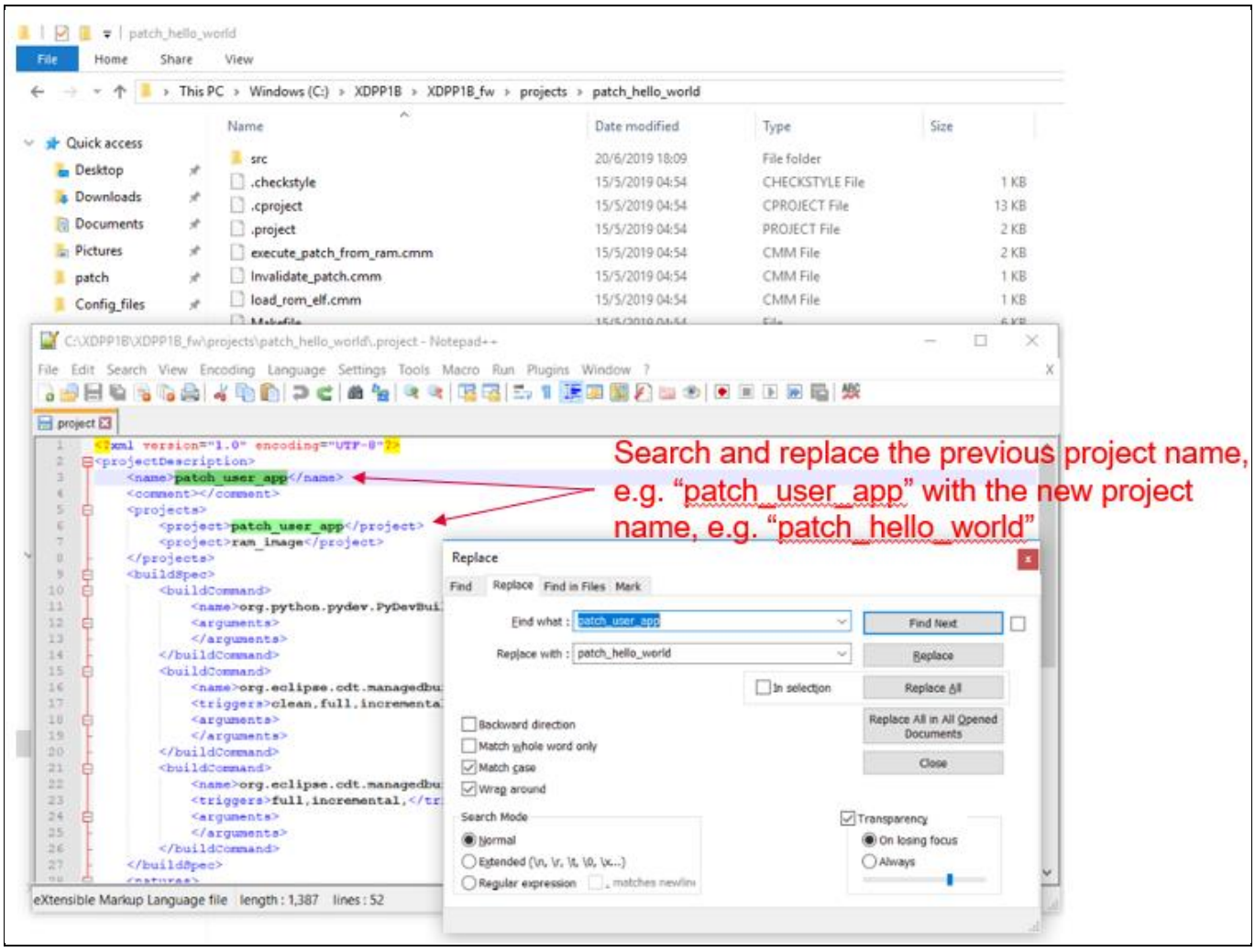
4.2 STEP 2: Rename the folder to your project

Rename the duplicated folder from “patch_user_app – Copy” to your project. In this example, you can rename it as “patch_hello_world”

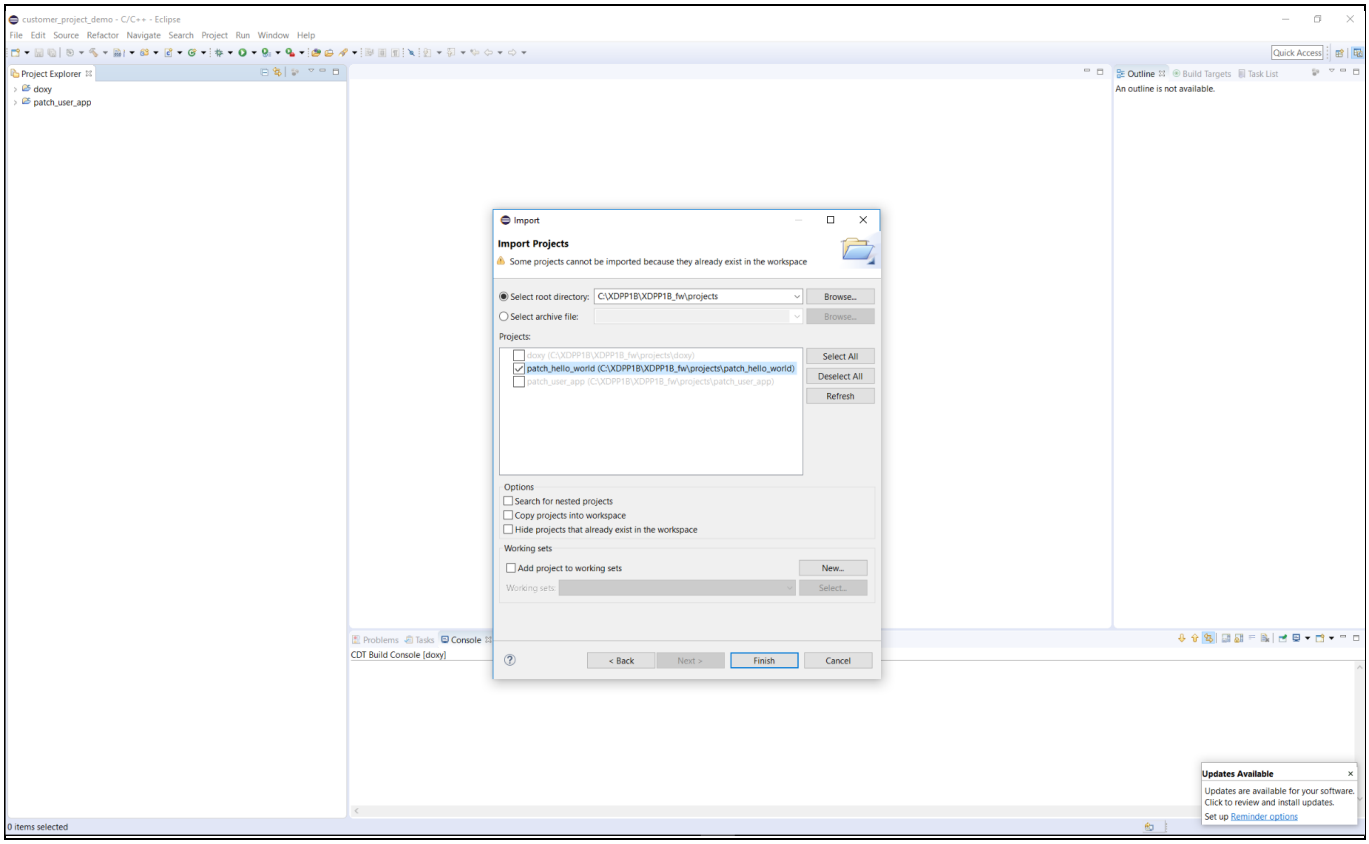


4.3 STEP 3: Edit ".project" file in the project folder

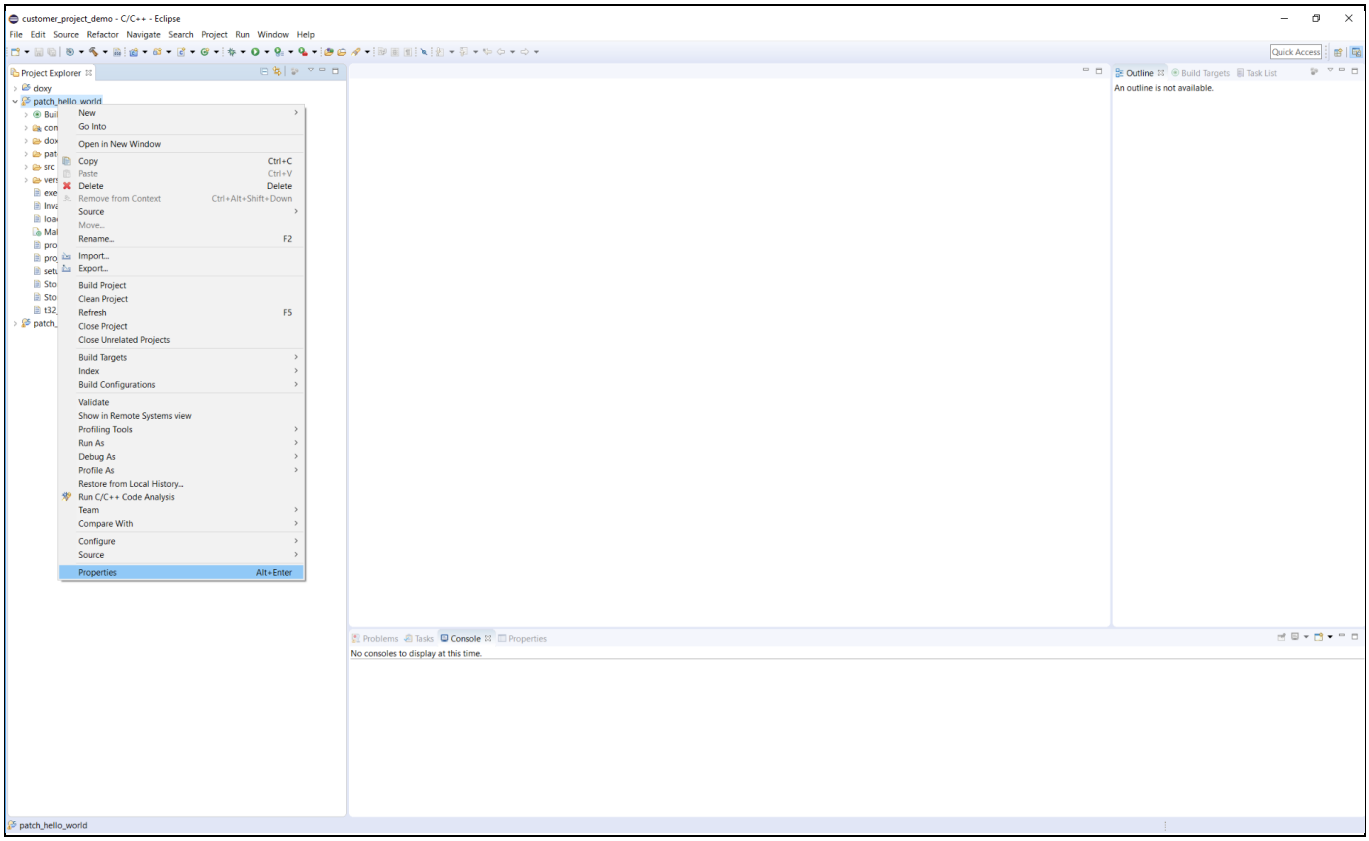
Navigate in to the patch_hello_world folder, look for a file called ".project". Sometimes it could be hidden due to Windows' setting. Use a text editor software to open this file, search and replace the previous project name "patch_user_app" with the new project name "patch_hello_world".



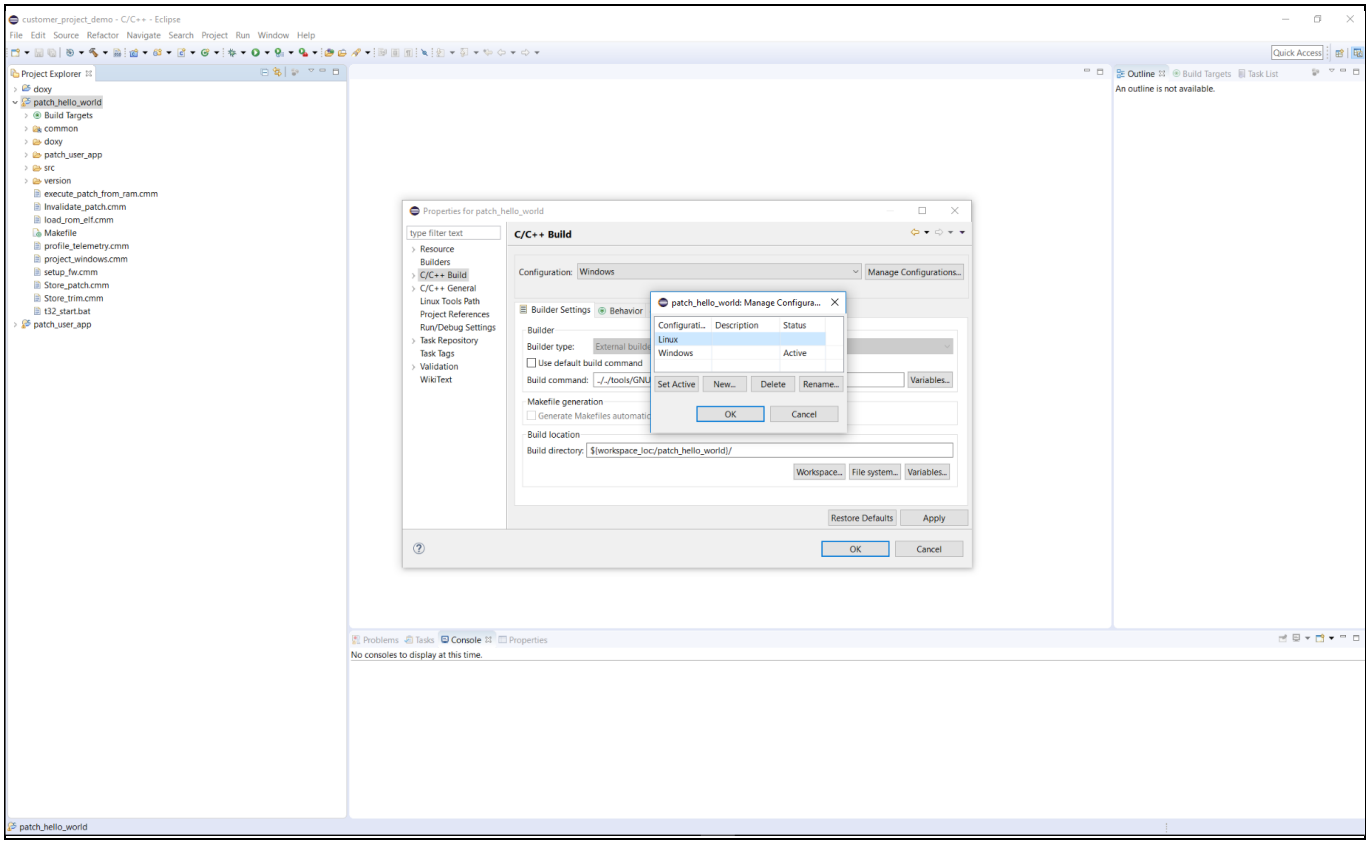
4.4 STEP 4: Go back to Eclipse and import the new project



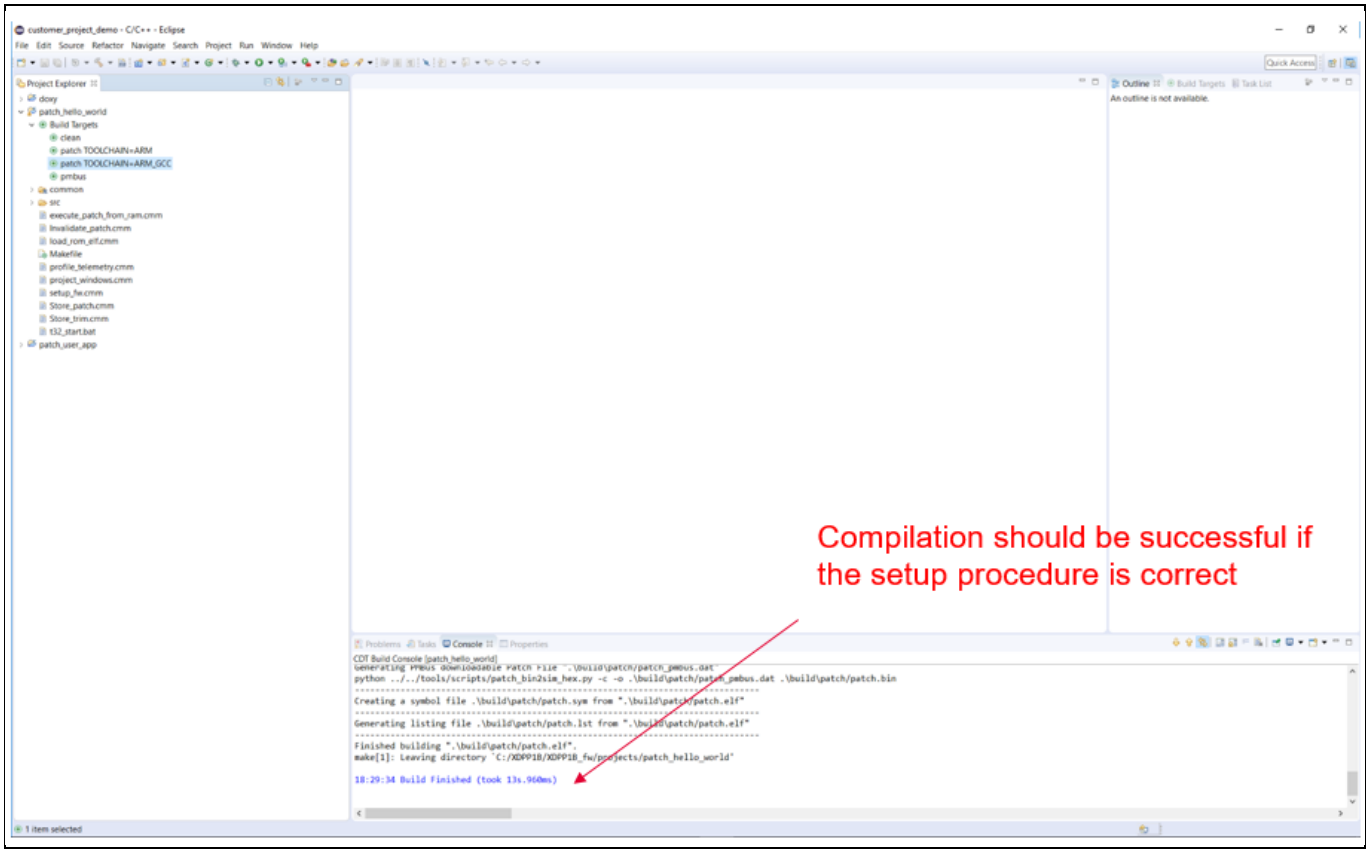
4.5 STEP 5: Right-**click and select "Properties"**



4.6 STEP 6: Set Build Configuration to “Windows”

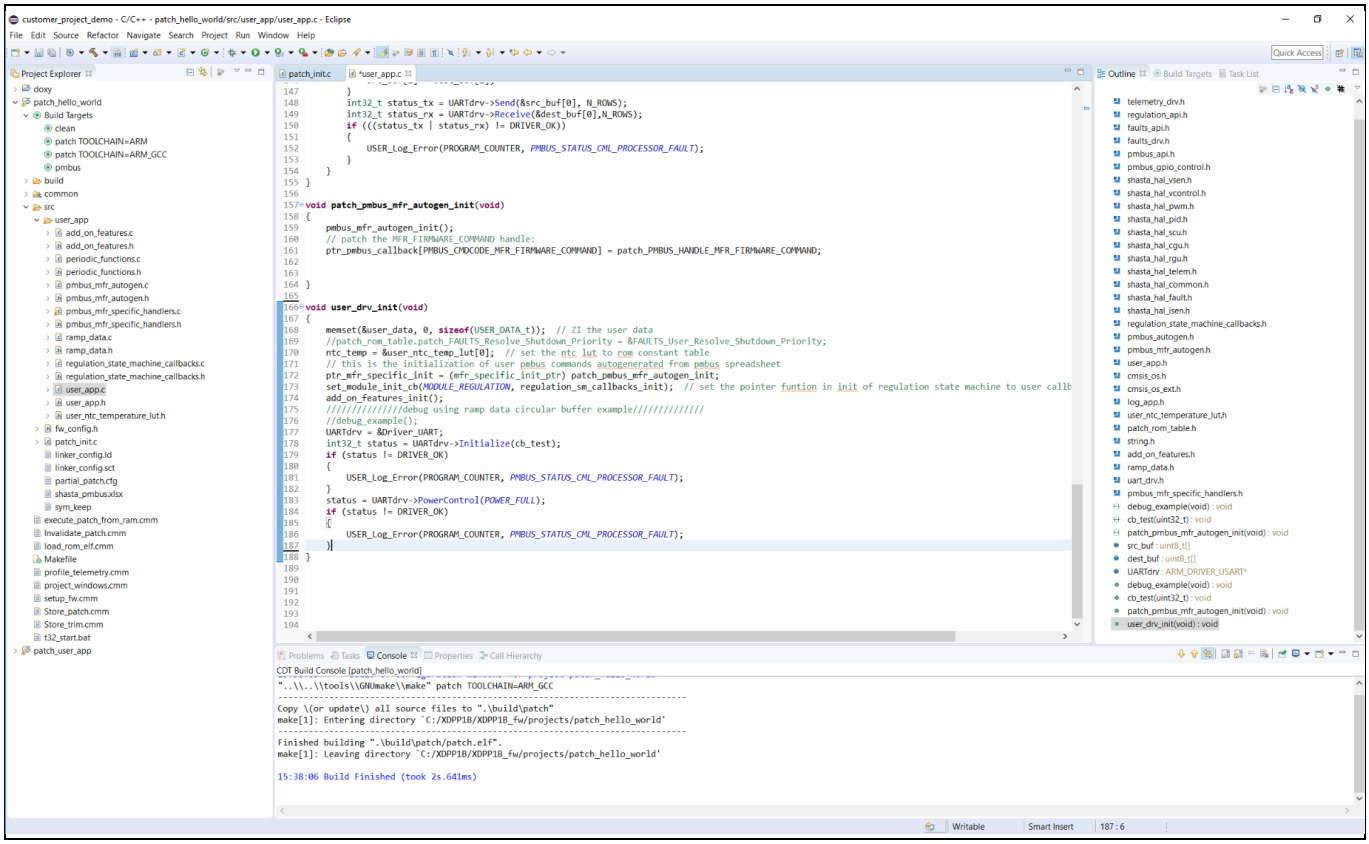


4.7 STEP 7: Expand the new project and Compile

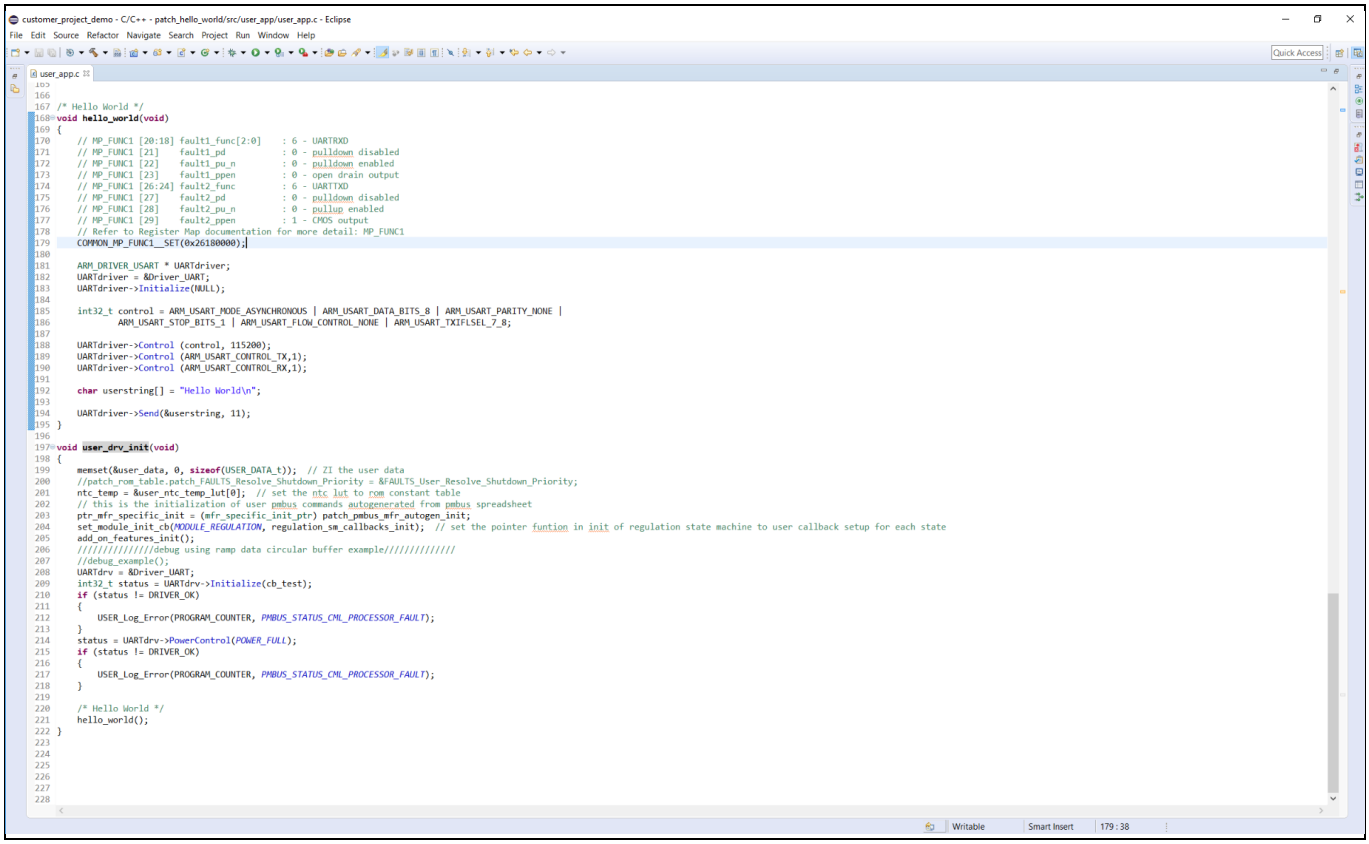


5 Implementing "Hello World" Project

5.1 STEP 1: Expand the project and go to \src\user_app\user_app.c



5.2 STEP 2: Create and Add “hello_world()” function into “user_drv_init()”



Implementing "Hello World" Project

Source code for hello_world() function as following:

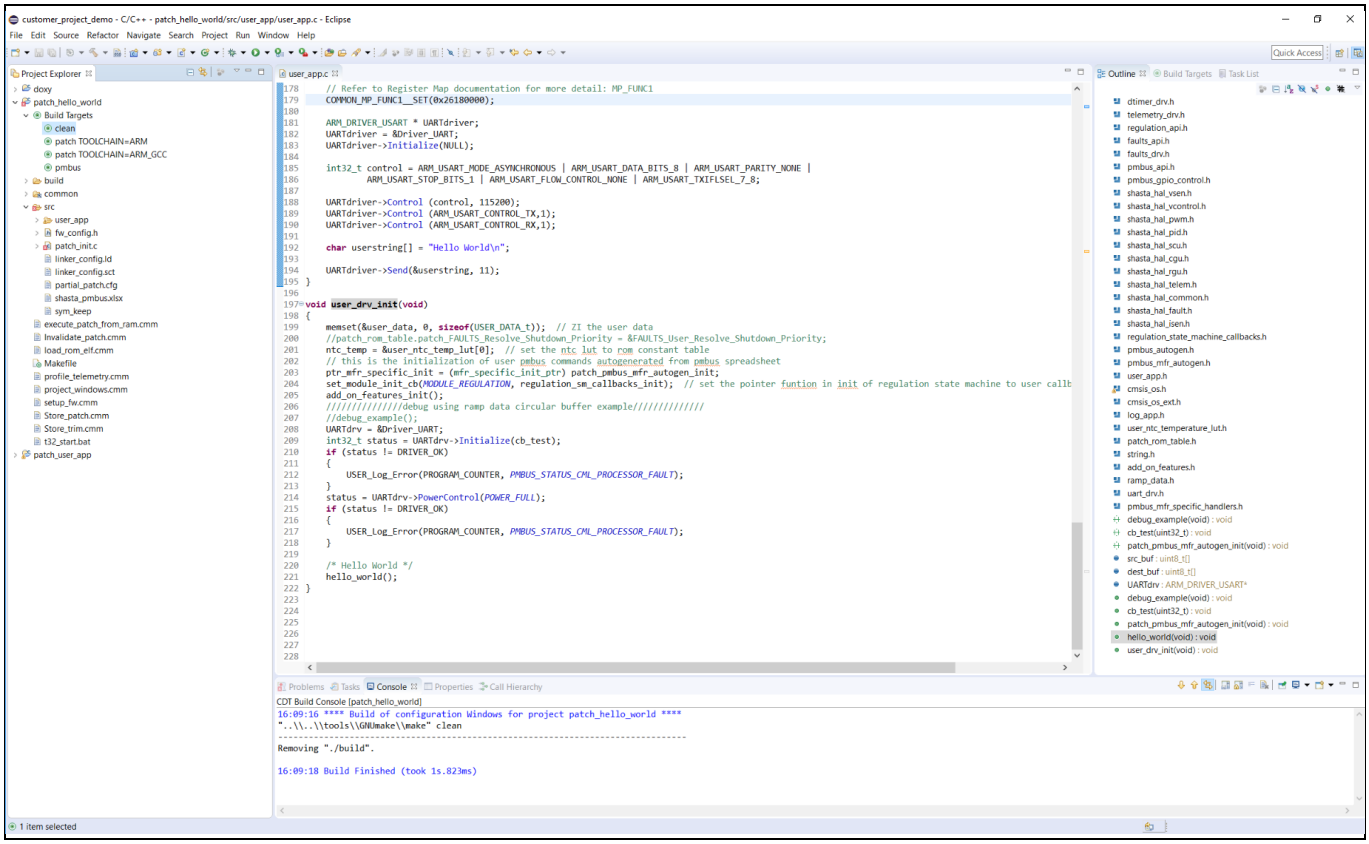
```
/* Hello World */
void hello_world(void)
{
	// MP_FUNC1 [20:18] fault1_func[2:0]   : 6 - UARTRXD
	// MP_FUNC1 [21]    fault1_pd          : 0 - pulldown disabled
	// MP_FUNC1 [22]    fault1_pu_n        : 0 - pulldown enabled
	// MP_FUNC1 [23]    fault1_ppen        : 0 - open drain output
	// MP_FUNC1 [26:24] fault2_func        : 6 - UARTTXD
	// MP_FUNC1 [27]    fault2_pd          : 0 - pulldown disabled
	// MP_FUNC1 [28]    fault2_pu_n        : 0 - pullup enabled
	// MP_FUNC1 [29]    fault2_ppen        : 1 - CMOS output
	// Refer to Register Map documentation for more detail: MP_FUNC1
	COMMON_MP_FUNC1__SET(0x26180000);

	ARM_DRIVER_USART * UARTdriver;
	UARTdriver = &Driver_UART;
	UARTdriver->Initialize(NULL);

	int32_t control = ARM_USART_MODE_ASYNCHRONOUS | ARM_USART_DATA_BITS_8 |
ARM_USART_PARITY_NONE |
		ARM_USART_STOP_BITS_1 | ARM_USART_FLOW_CONTROL_NONE |
ARM_USART_TXIFLSEL_7_8;

	UARTdriver->Control (control, 115200);
	UARTdriver->Control (ARM_USART_CONTROL_TX,1);
	UARTdriver->Control (ARM_USART_CONTROL_RX,1);

	char userstring[] = "Hello World";

	UARTdriver->Send(&userstring, 11);
}
```

Implementing "Hello World" Project

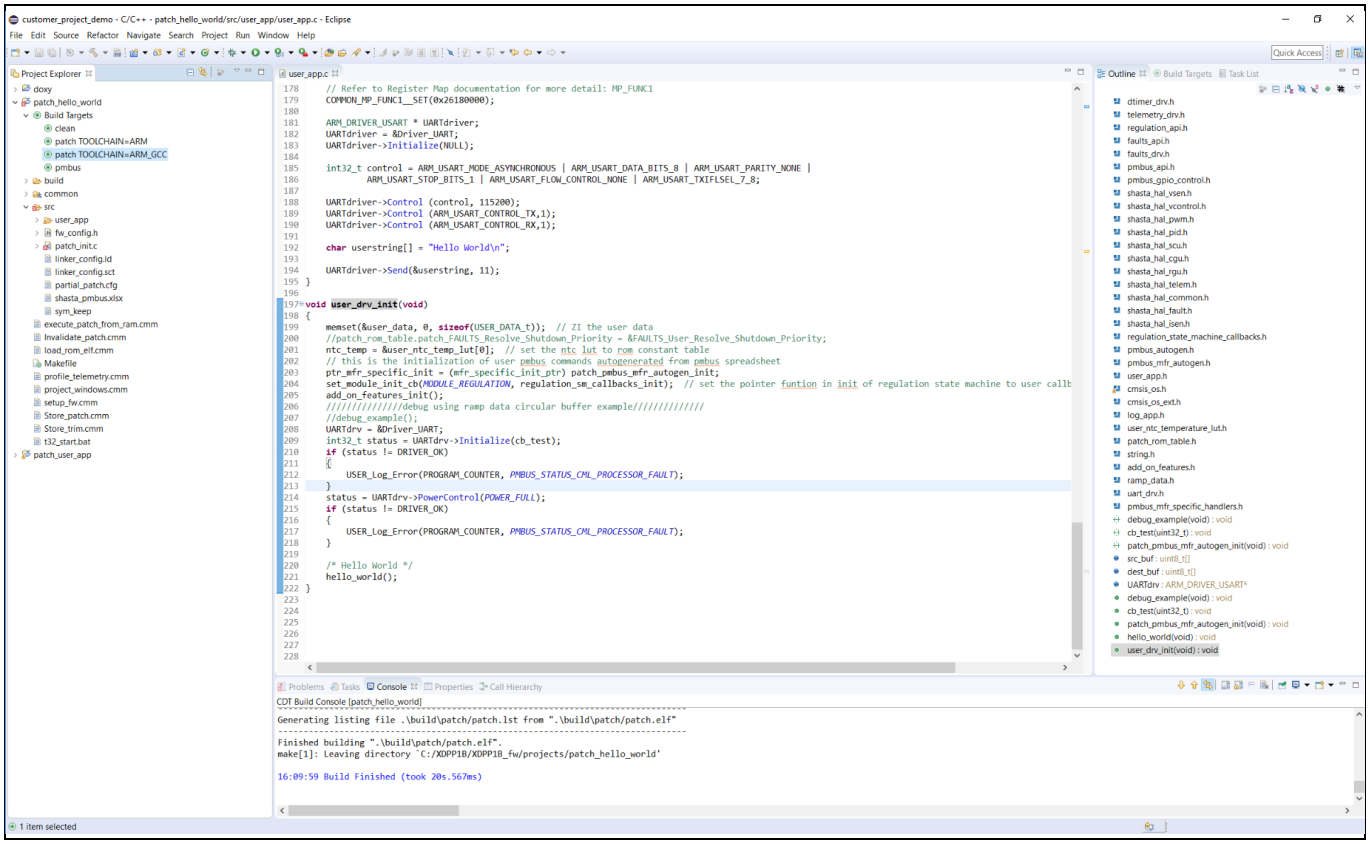
Source code for user_drv_init() has to be modified as following:

```
void user_drv_init(void)
{
	memset(&user_data, 0, sizeof(USER_DATA_t));  // ZI the user data
	//patch_rom_table.patch_FAULTS_Resolve_Shutdown_Priority =
&FAULTS_User_Resolve_Shutdown_Priority;
	ntc_temp = &user_ntc_temp_lut[0];  // set the ntc lut to rom constant table
	// this is the initialization of user pmbus commands autogenerated from pmbus
spreadsheet
	ptr_mfr_specific_init = (mfr_specific_init_ptr) patch_pmbus_mfr_autogen_init;
	set_module_init_cb(MODULE_REGULATION, regulation_sm_callbacks_init);  // set the
pointer funtion in init of regulation state machine to user callback setup for each state
	add_on_features_init();
	///////////////debug using ramp data circular buffer example//////////////
	//debug_example();
	UARTdrv = &Driver_UART;
	int32_t status = UARTdrv->Initialize(cb_test);
	if (status != DRIVER_OK)
	{
		USER_Log_Error(PROGRAM_COUNTER, PMBUS_STATUS_CML_PROCESSOR_FAULT);
	}
	status = UARTdrv->PowerControl(POWER_FULL);
	if (status != DRIVER_OK)
	{
		USER_Log_Error(PROGRAM_COUNTER, PMBUS_STATUS_CML_PROCESSOR_FAULT);
	}

	/* Hello World */
	hello_world();

}
```

5.3 STEP 3: Build- Clean to remove /build folder

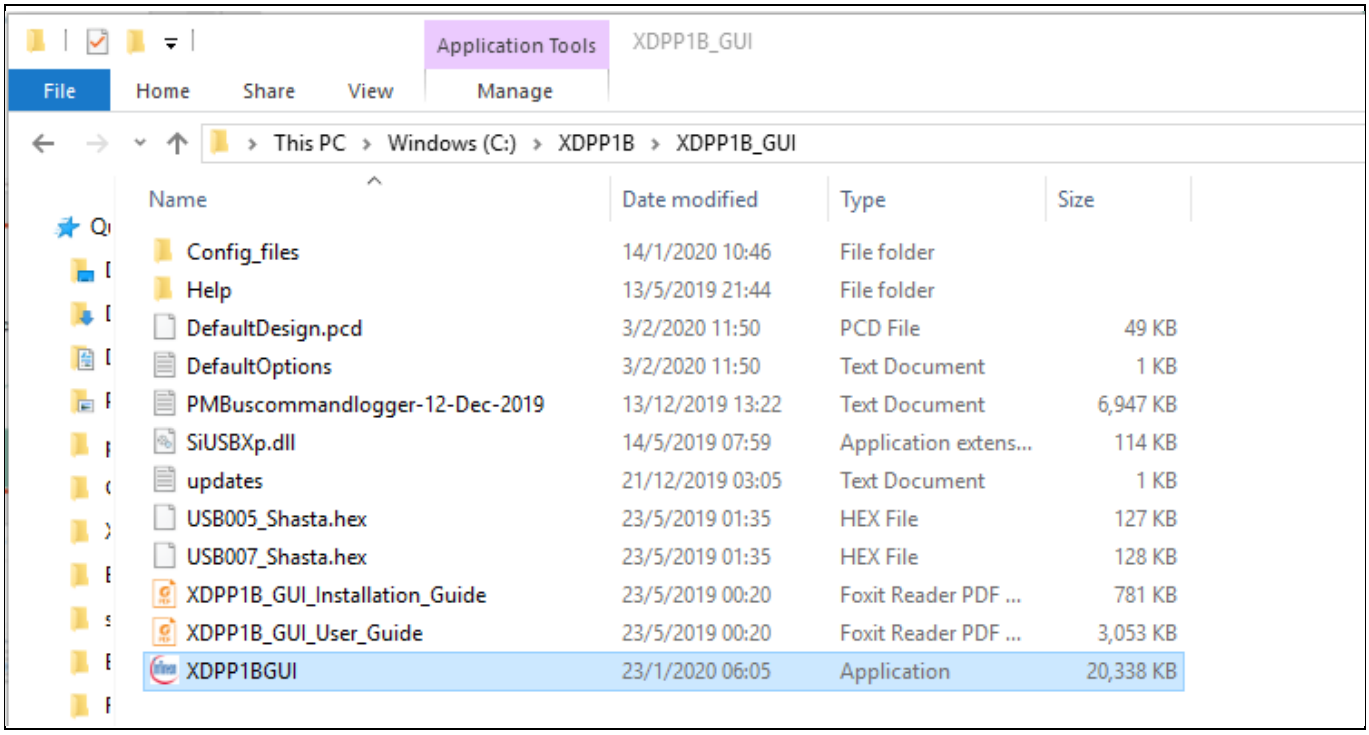


5.4 STEP 4: Compile and Build Patch (ARM_GCC)



6 Downloading Patch to OTP

6.1 STEP 1: Open XDPP1B GUI



6.2 STEP 2: Connect PMBus Dongle and XDPP1B

Photo of dongle:

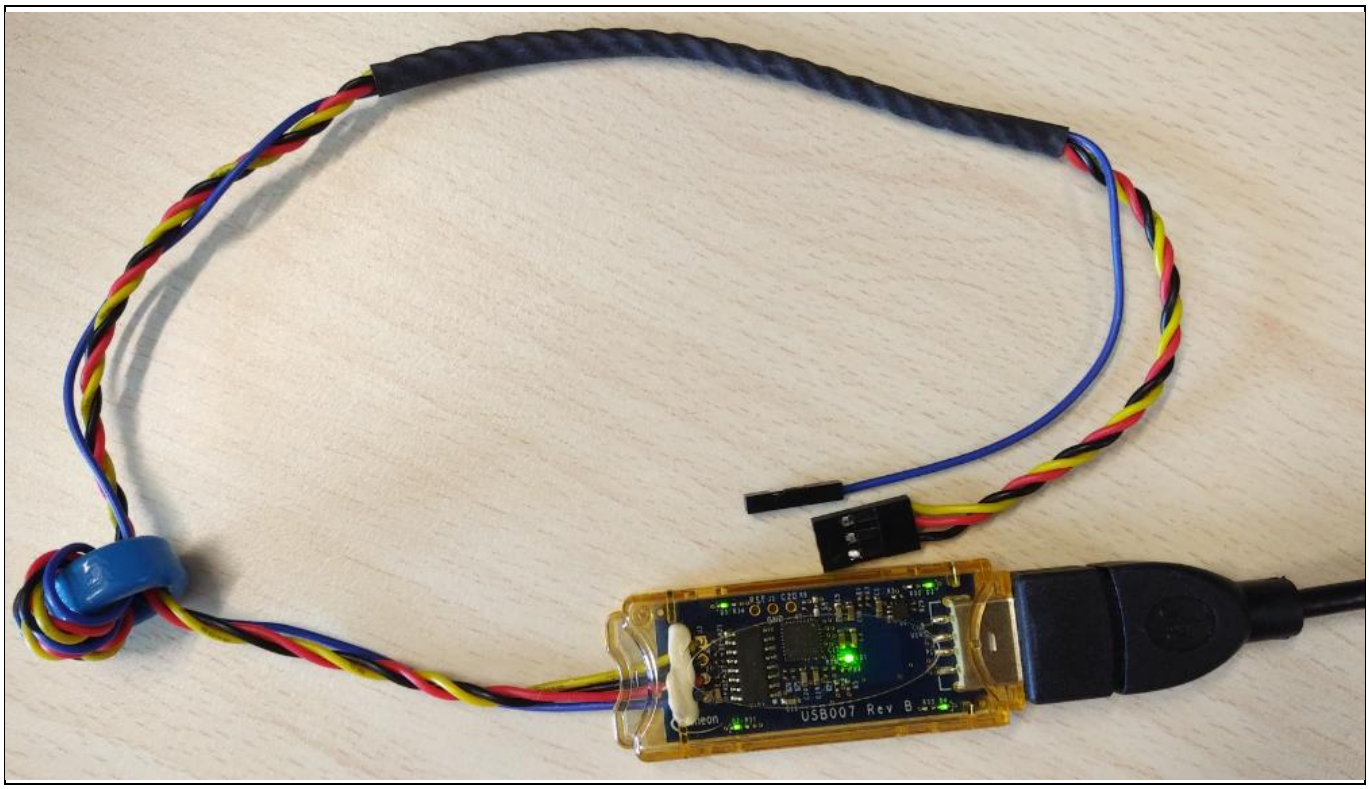
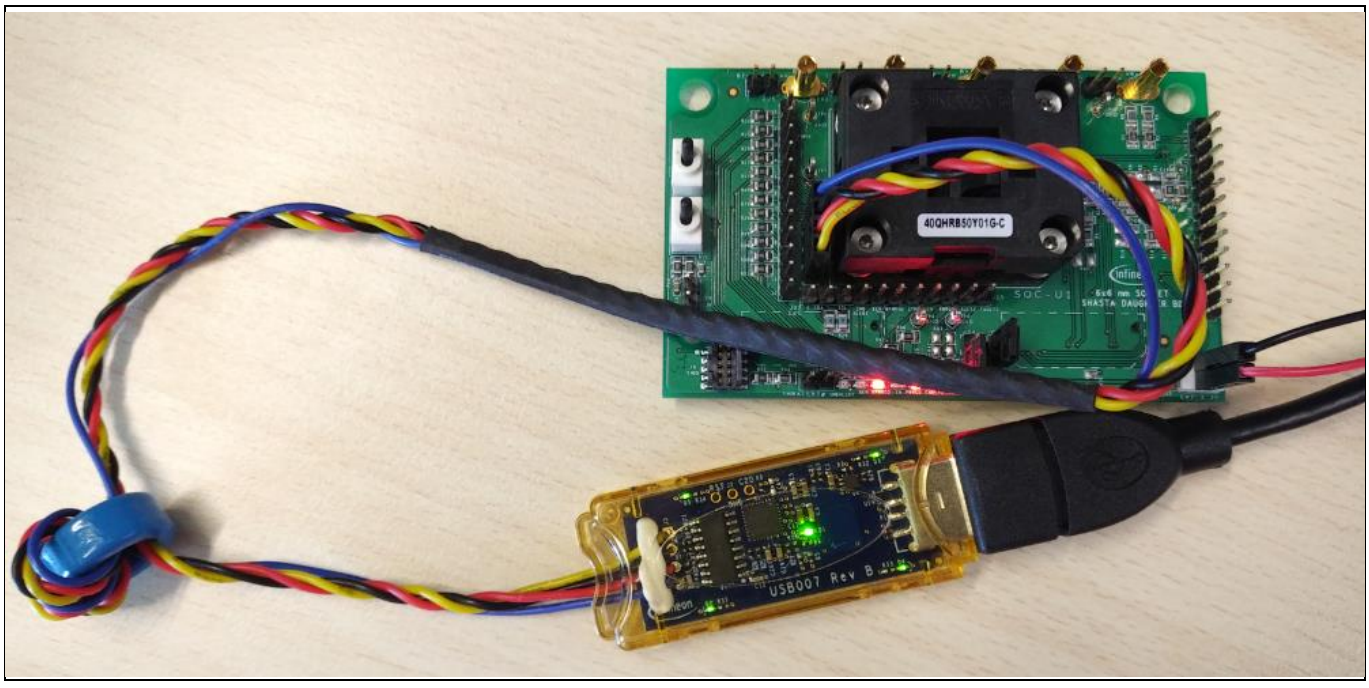
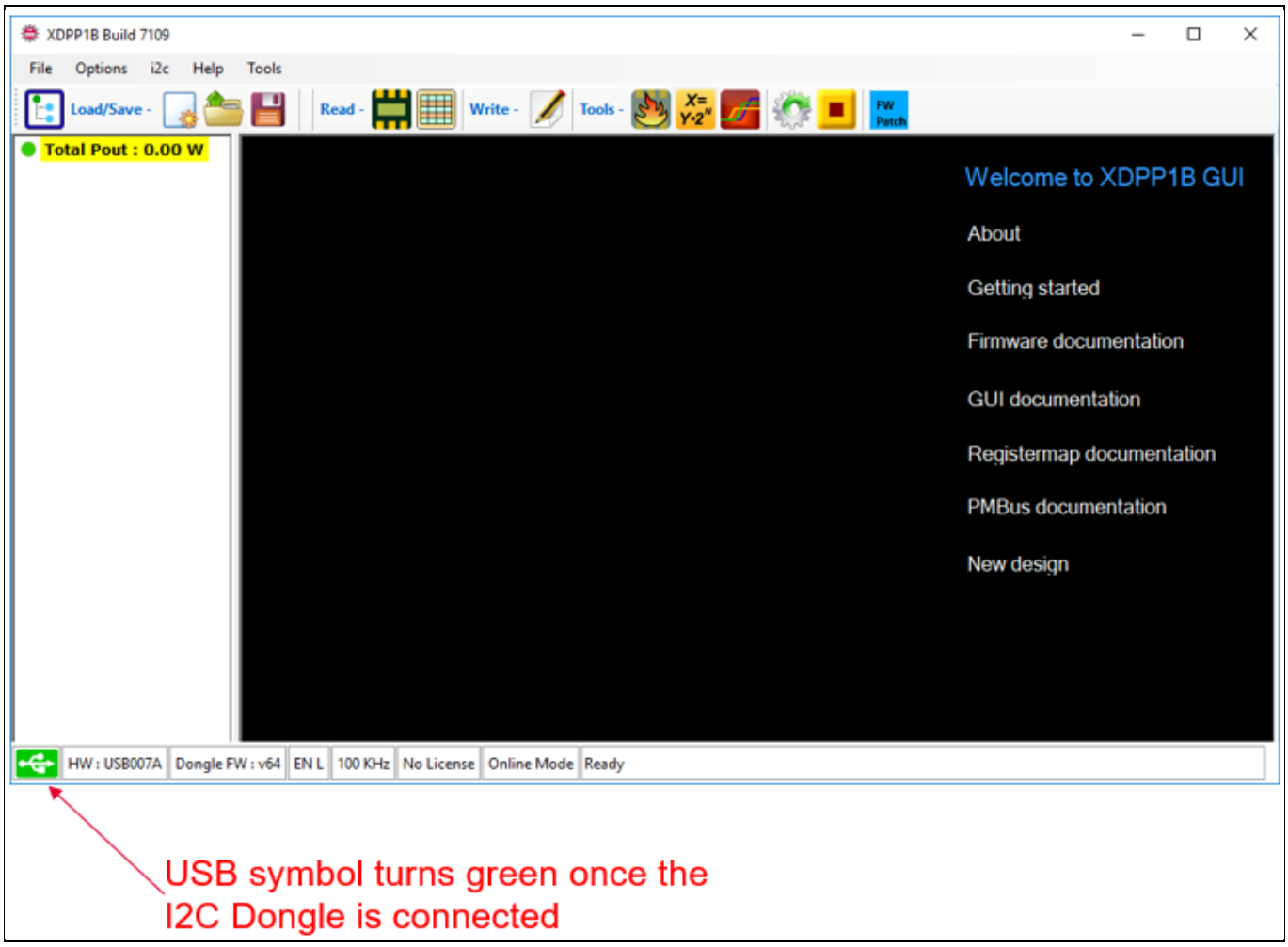
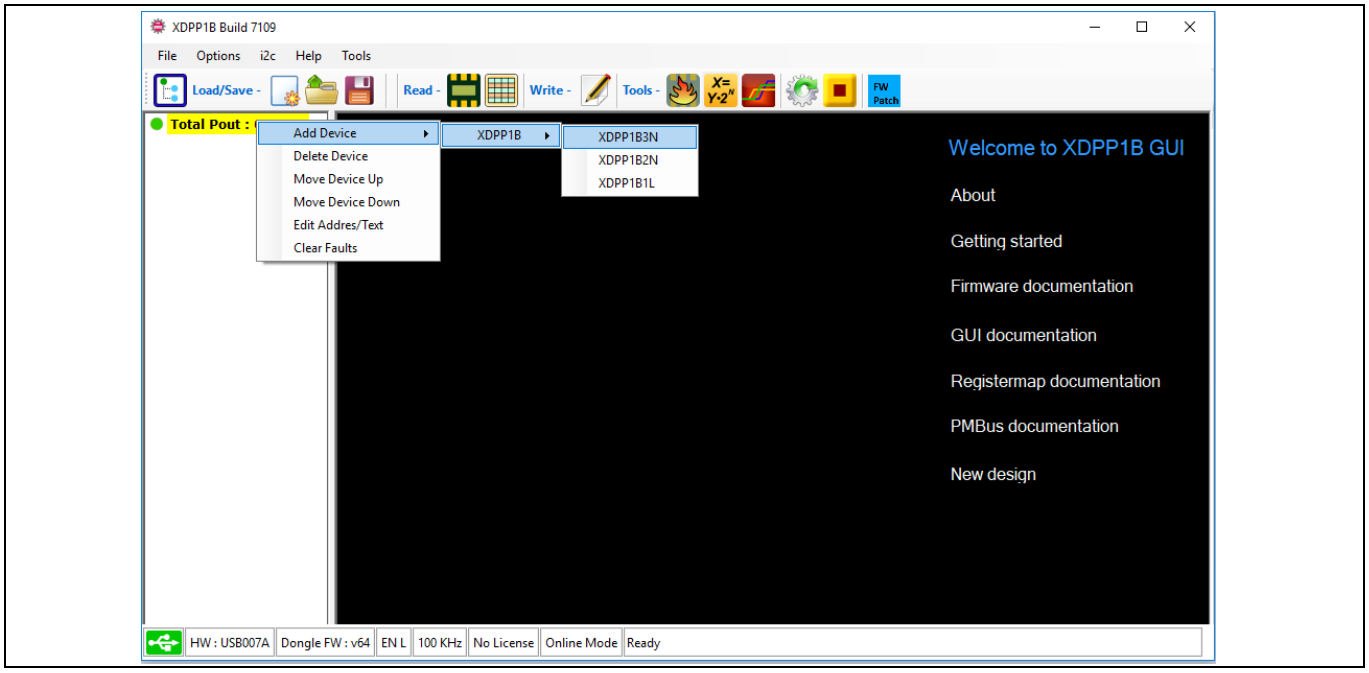


Photo of connection:

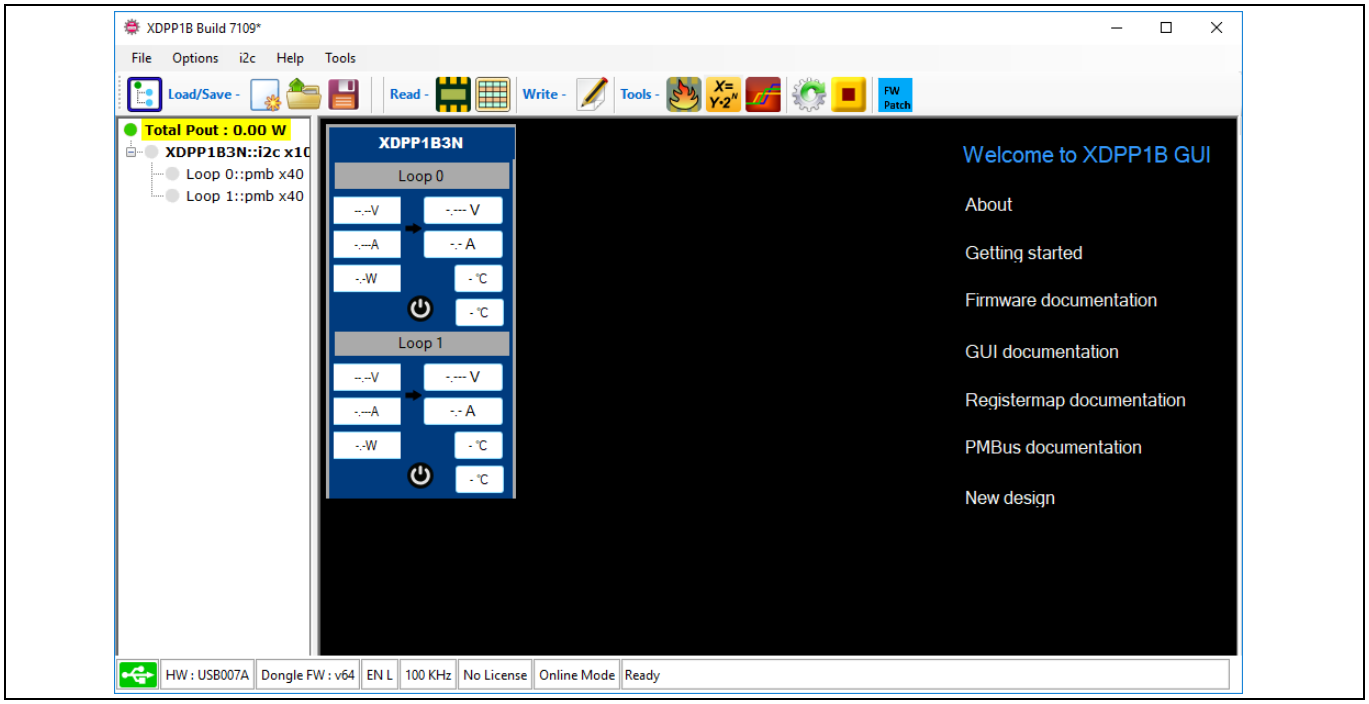



XDPP1B Build 7109
File
Options
i2c
Help
Tools
Load/Save -
Read -
Write -
Tools -
FW Patch
Total Pout : 0.00 W
Welcome to XDPP1B GUI
About
Getting started
Firmware documentation
GUI documentation
Registermap documentation
PMBus documentation
New design
HW : USB007A
Dongle FW : v64
EN L
100 KHz
No License
Online Mode
Ready
USB symbol turns green once the
I2C Dongle is connected

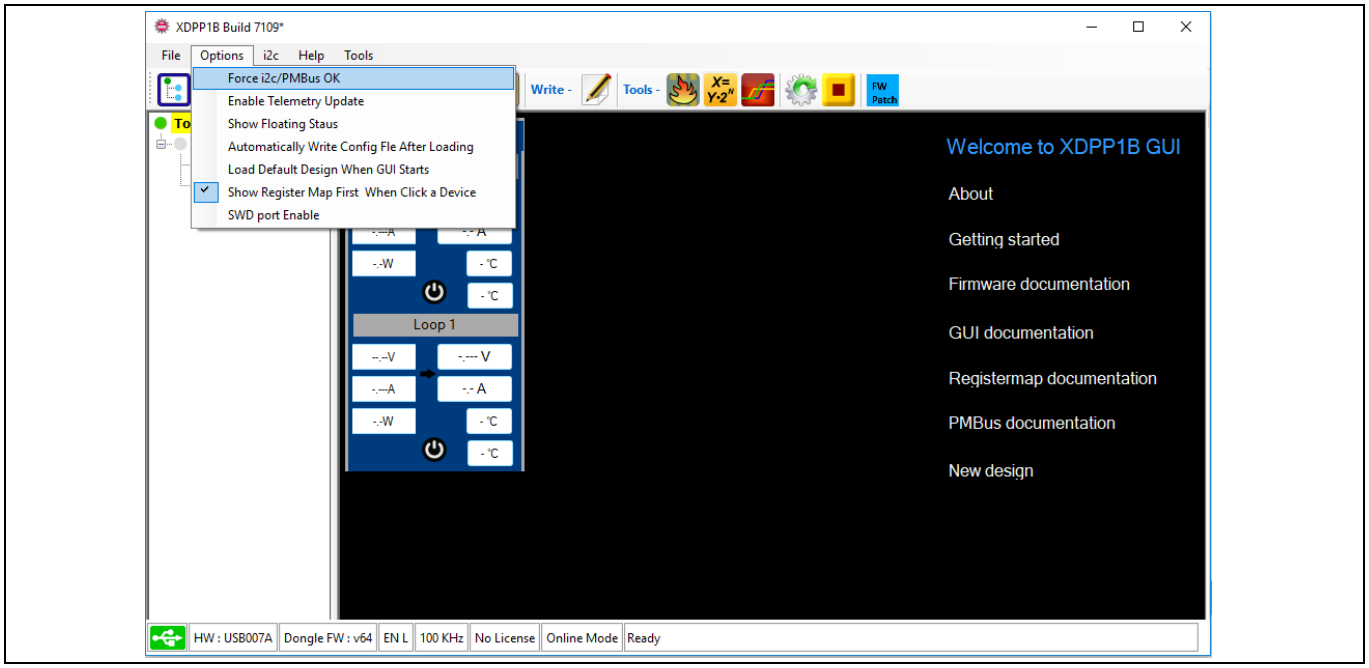
6.3 STEP 3: Add Device



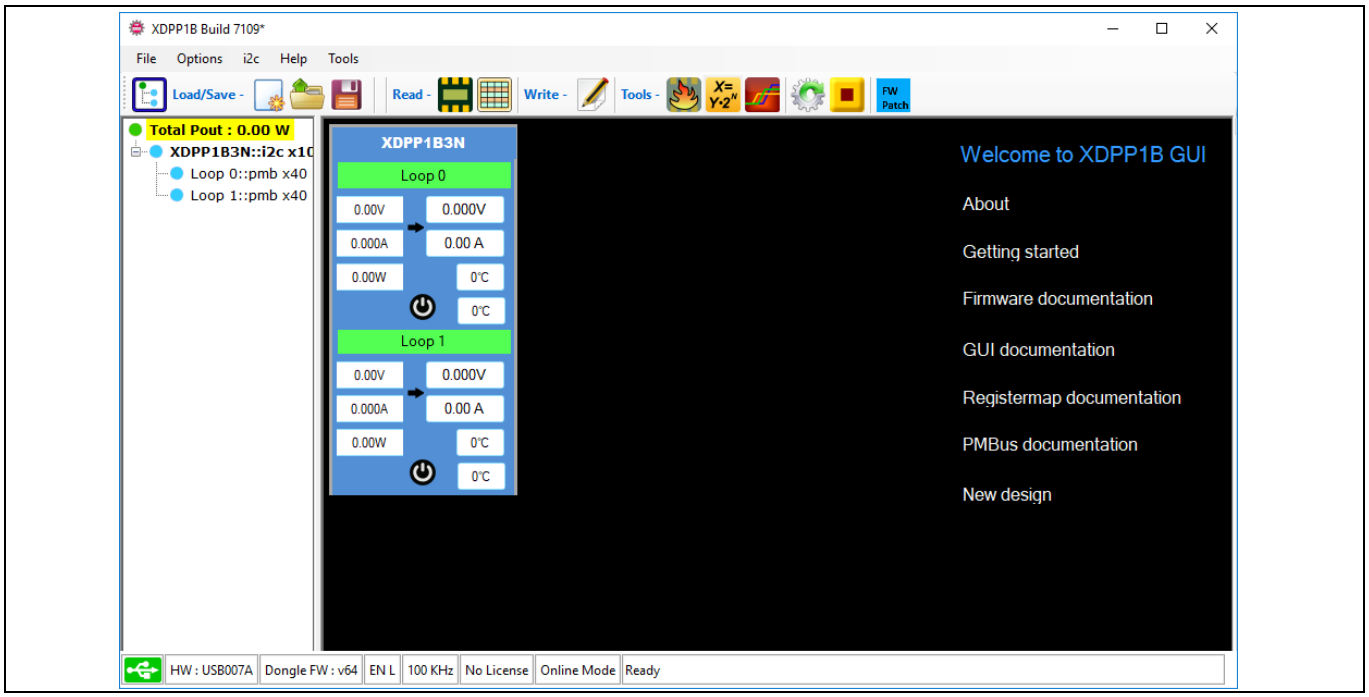
After add device:



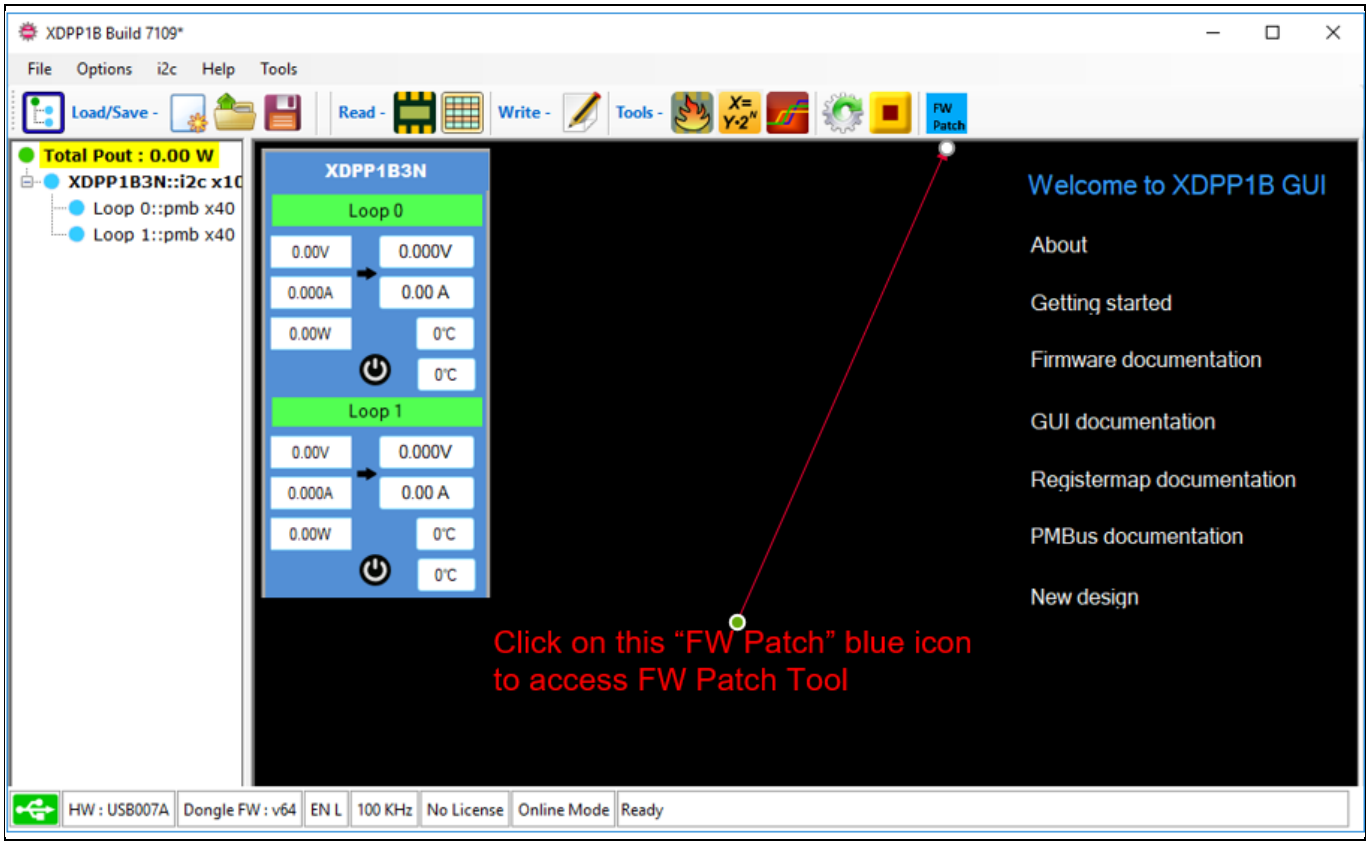
6.4 STEP 4: Force I2C/PMBus OK

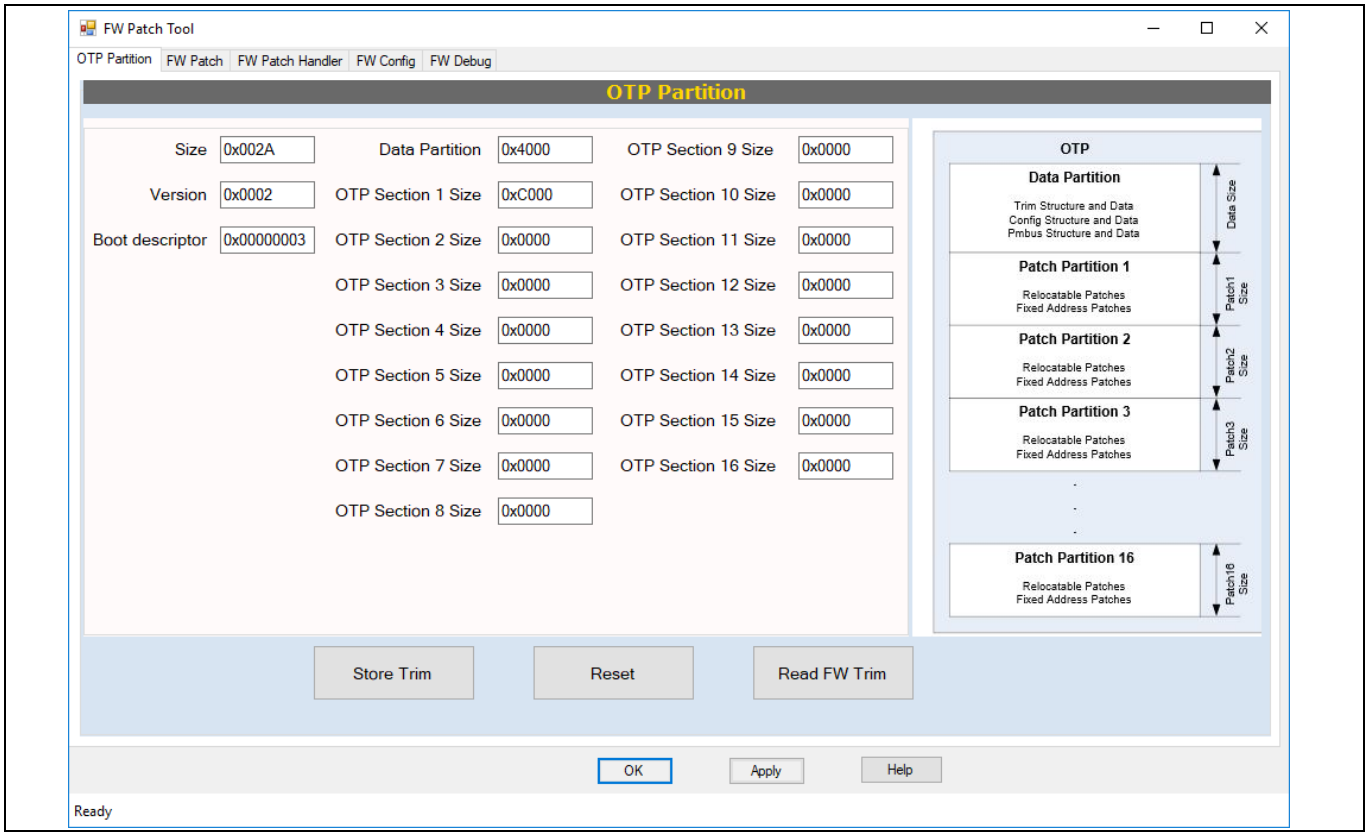


After Force I2C/PMBus:

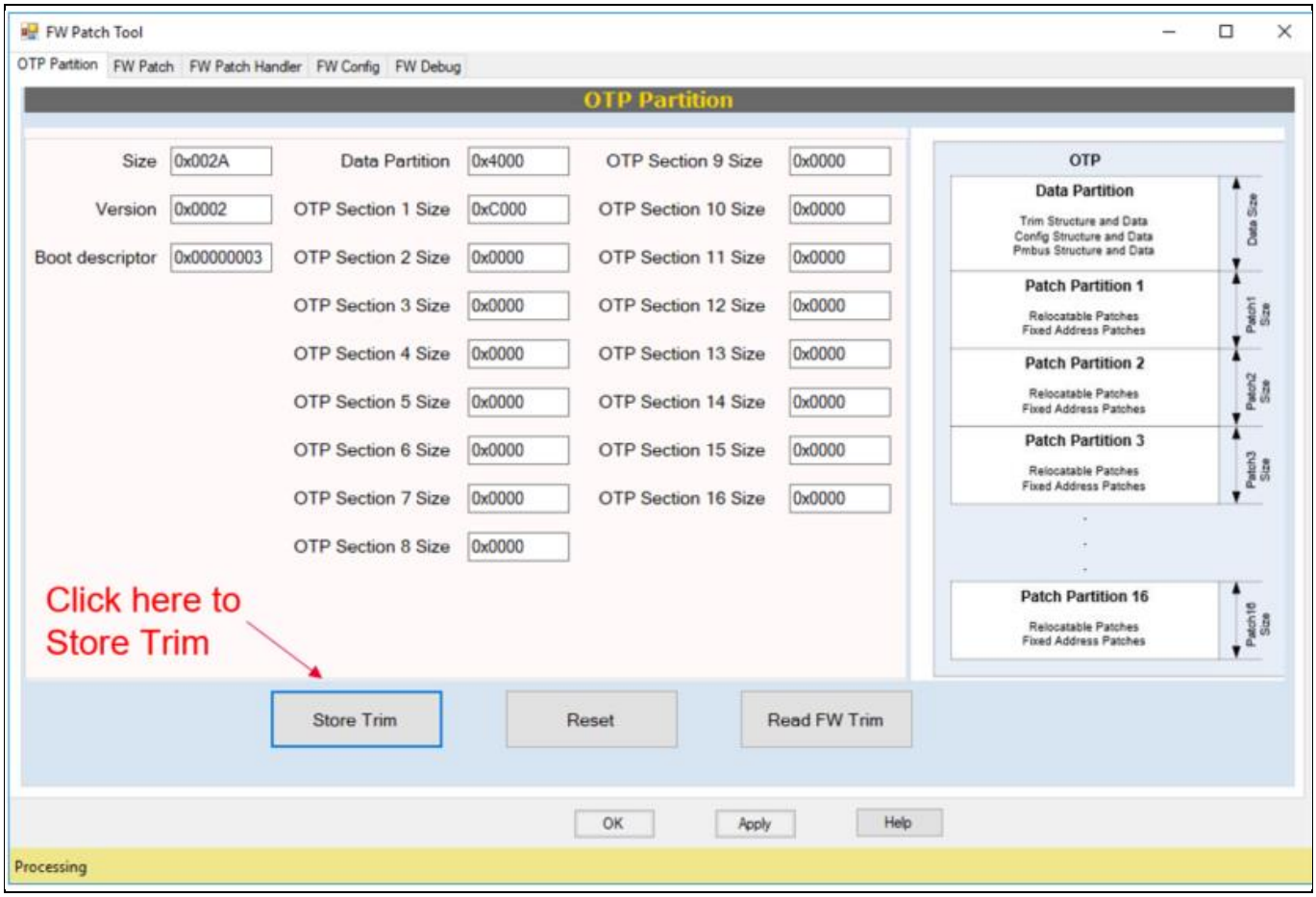


6.5 STEP 5: Access FW Patch Tool

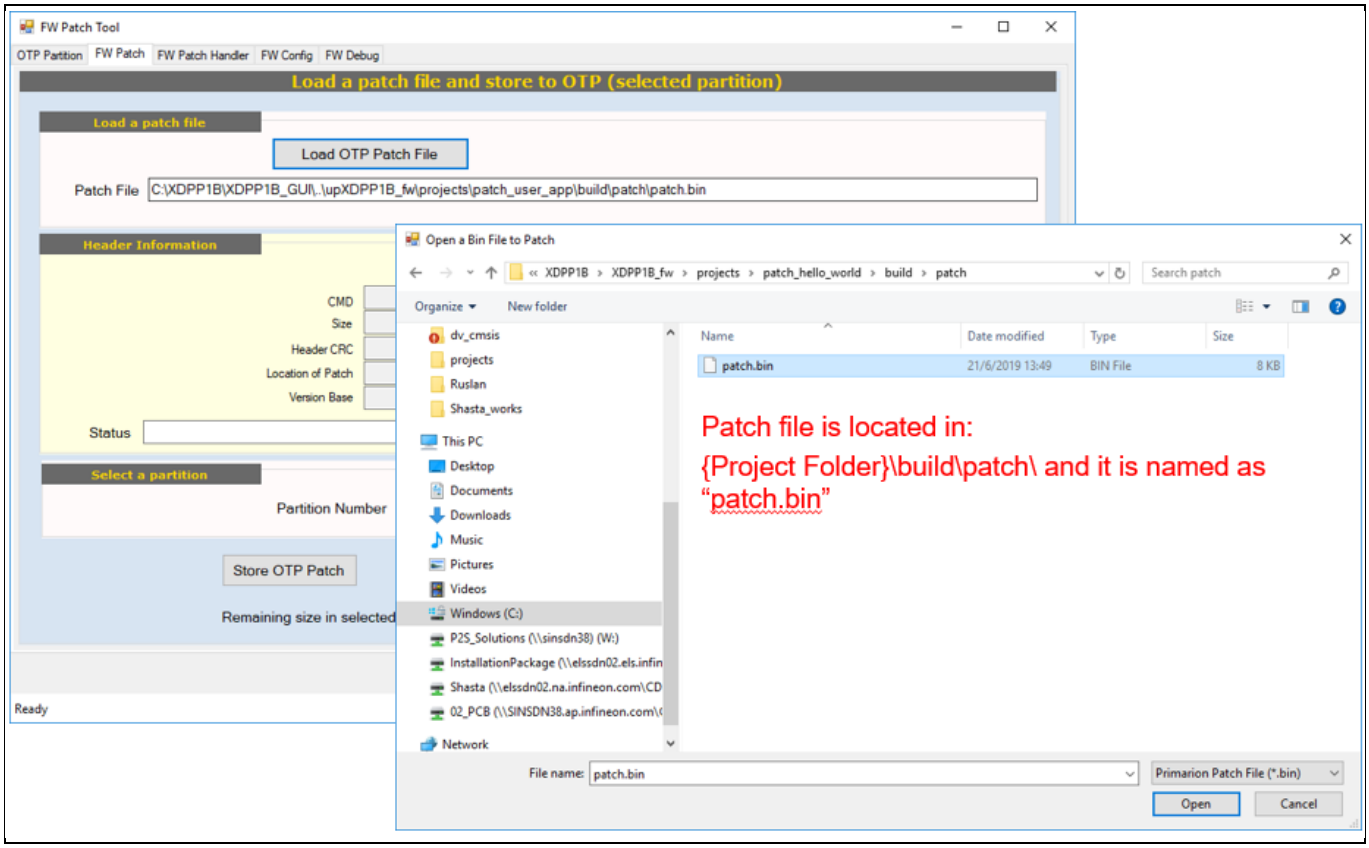




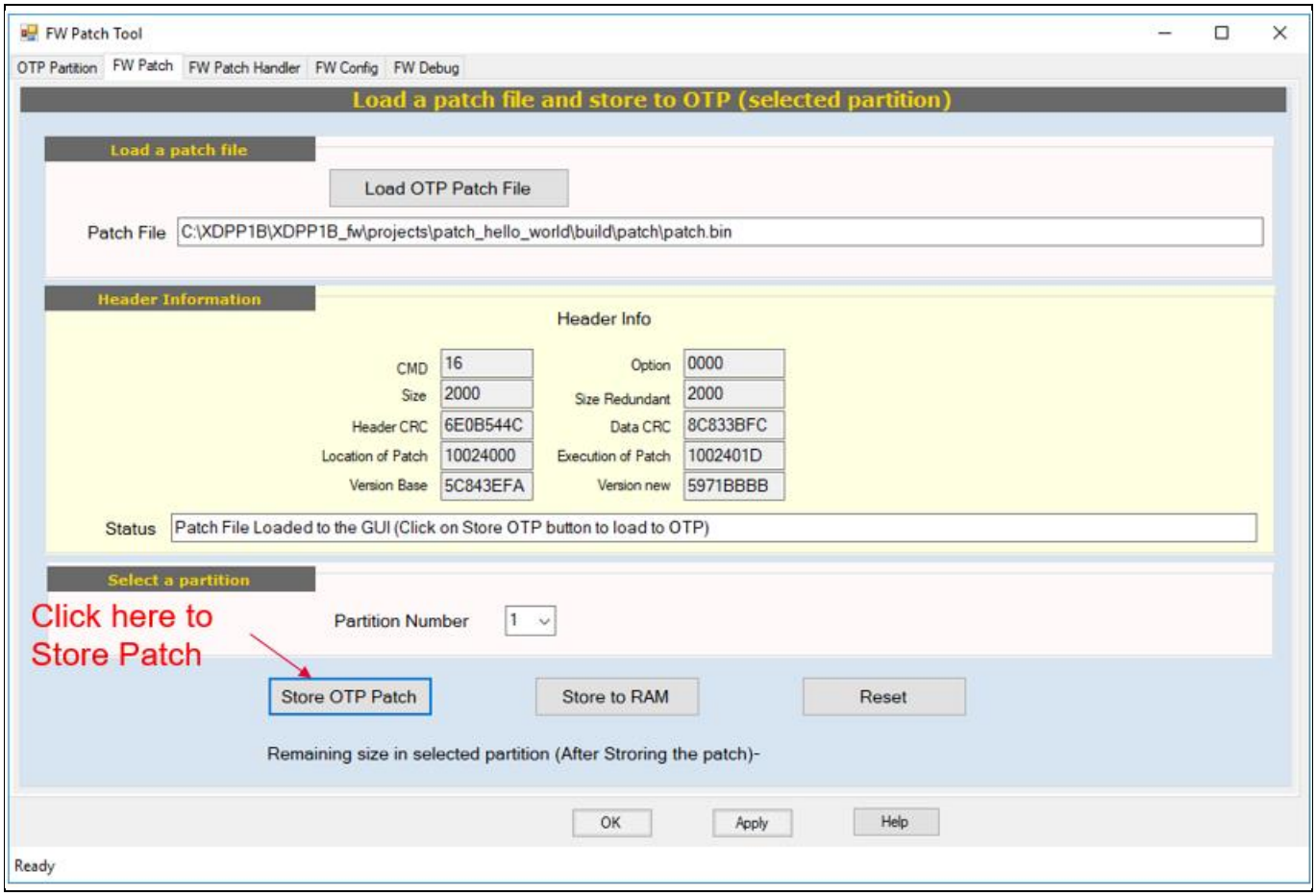
6.6 STEP 6: Store Trim



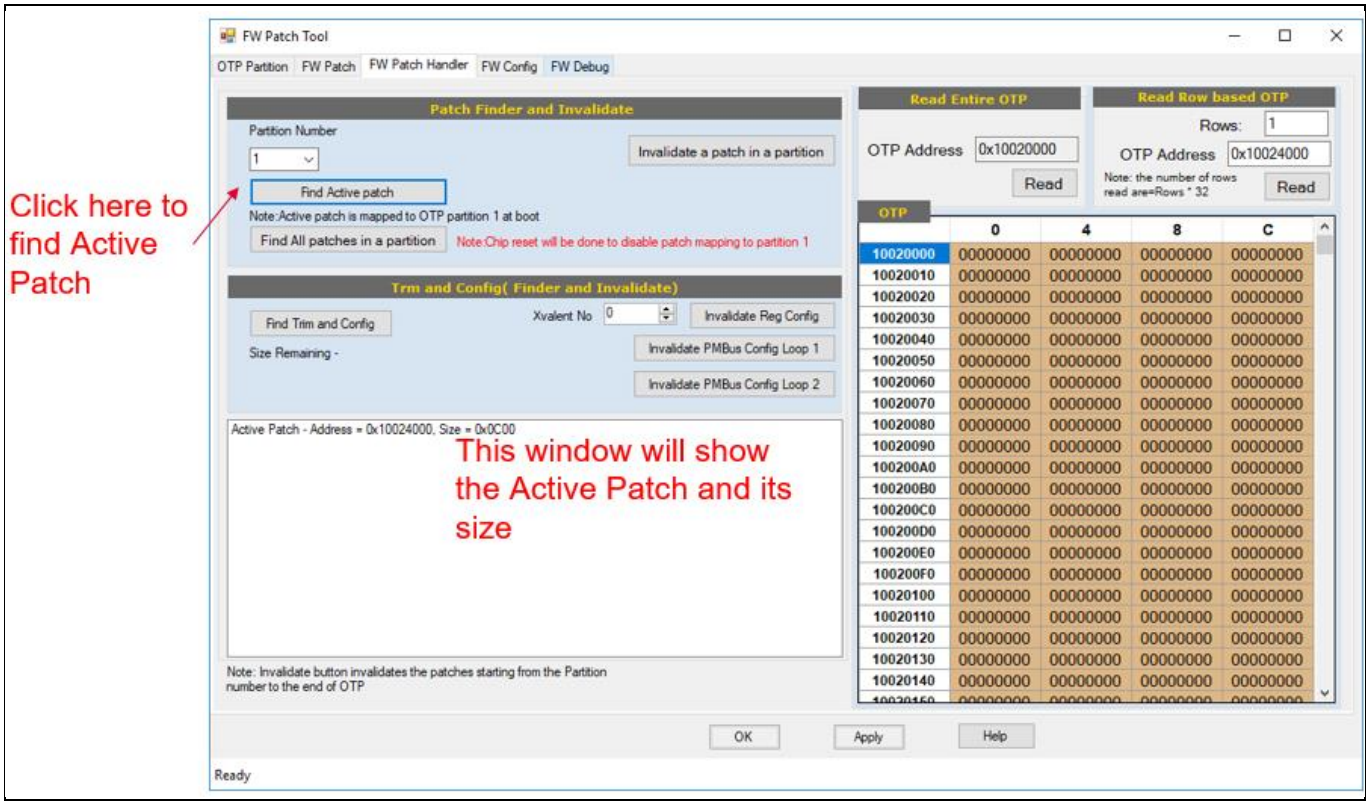
6.7 STEP 7: Load Patch File



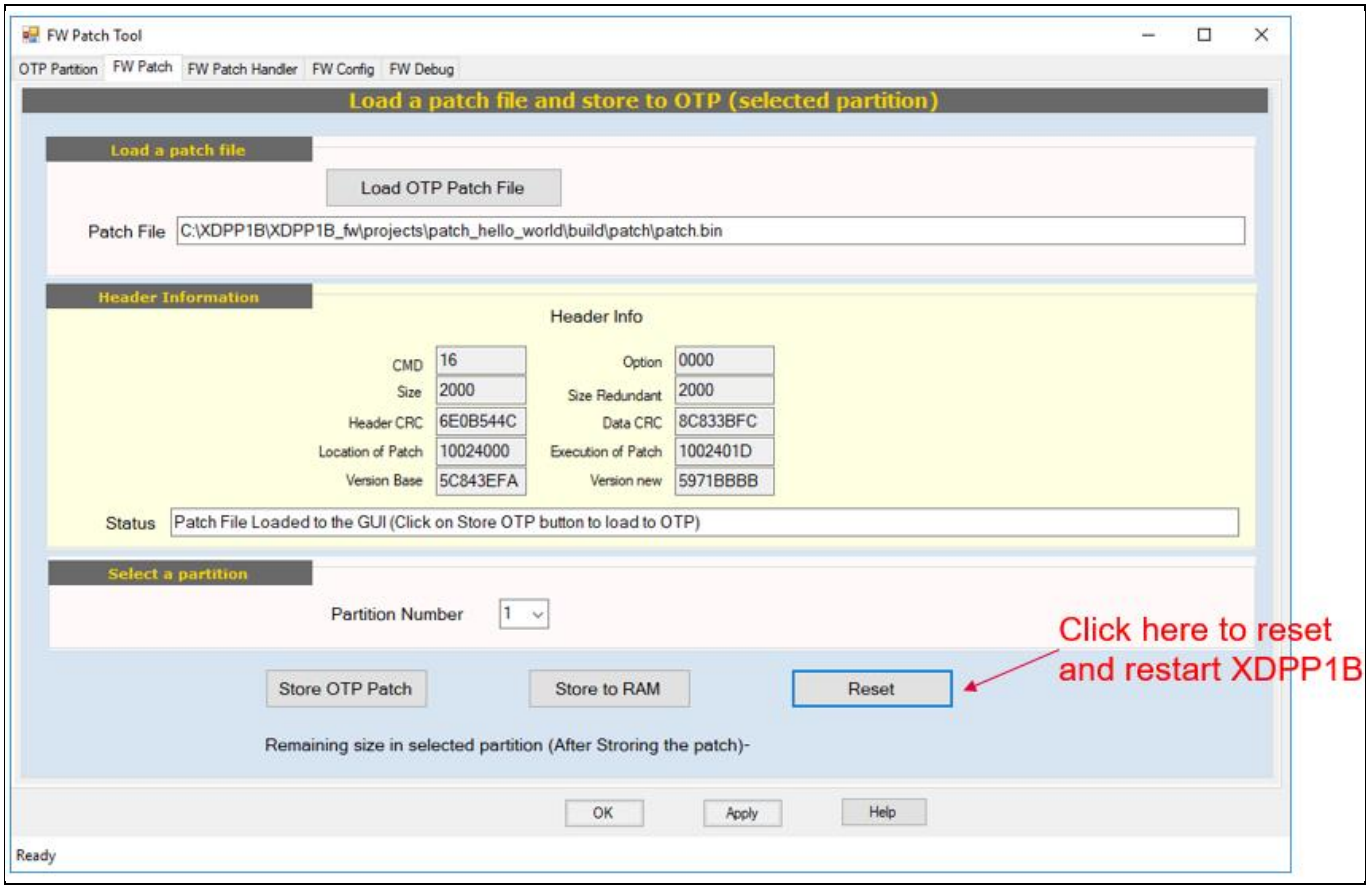
6.8 STEP 8: Store Patch File to OTP



6.9 STEP 9: Check for Active Patch

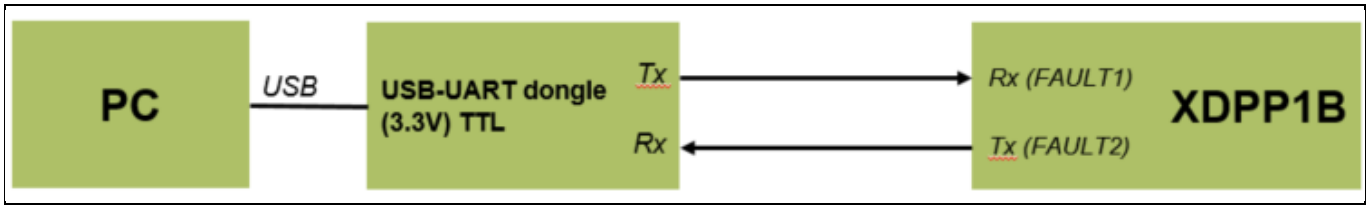


6.10 STEP 10: Patch is loaded. Restart XDPP1100.

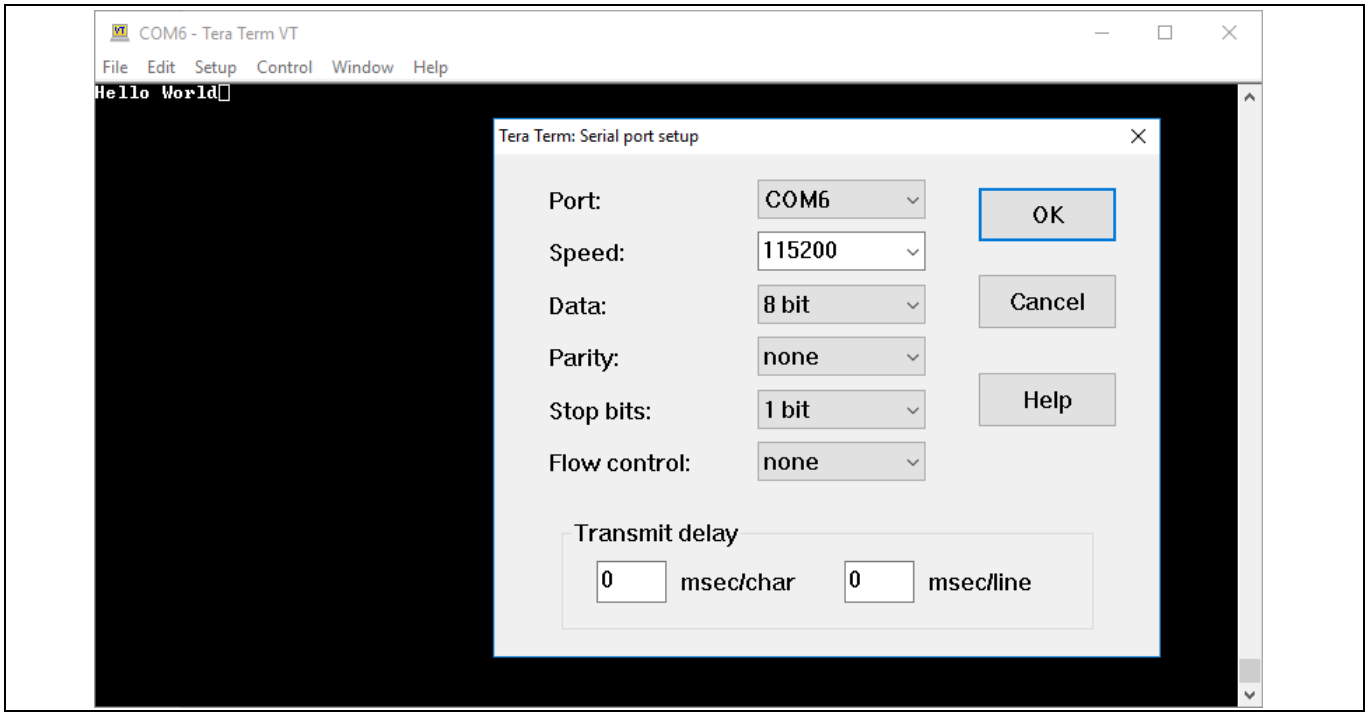


7 Monitor UART Traffic

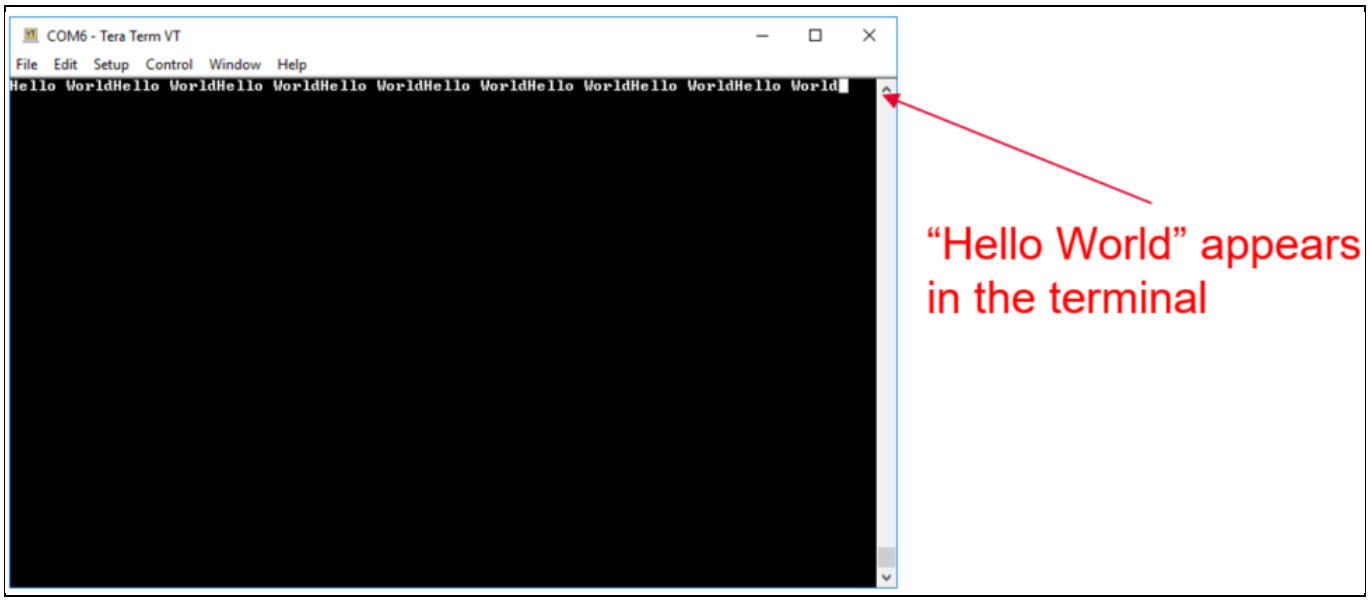
7.1 STEP 1: Connect XDPP1100 with USB-UART dongle

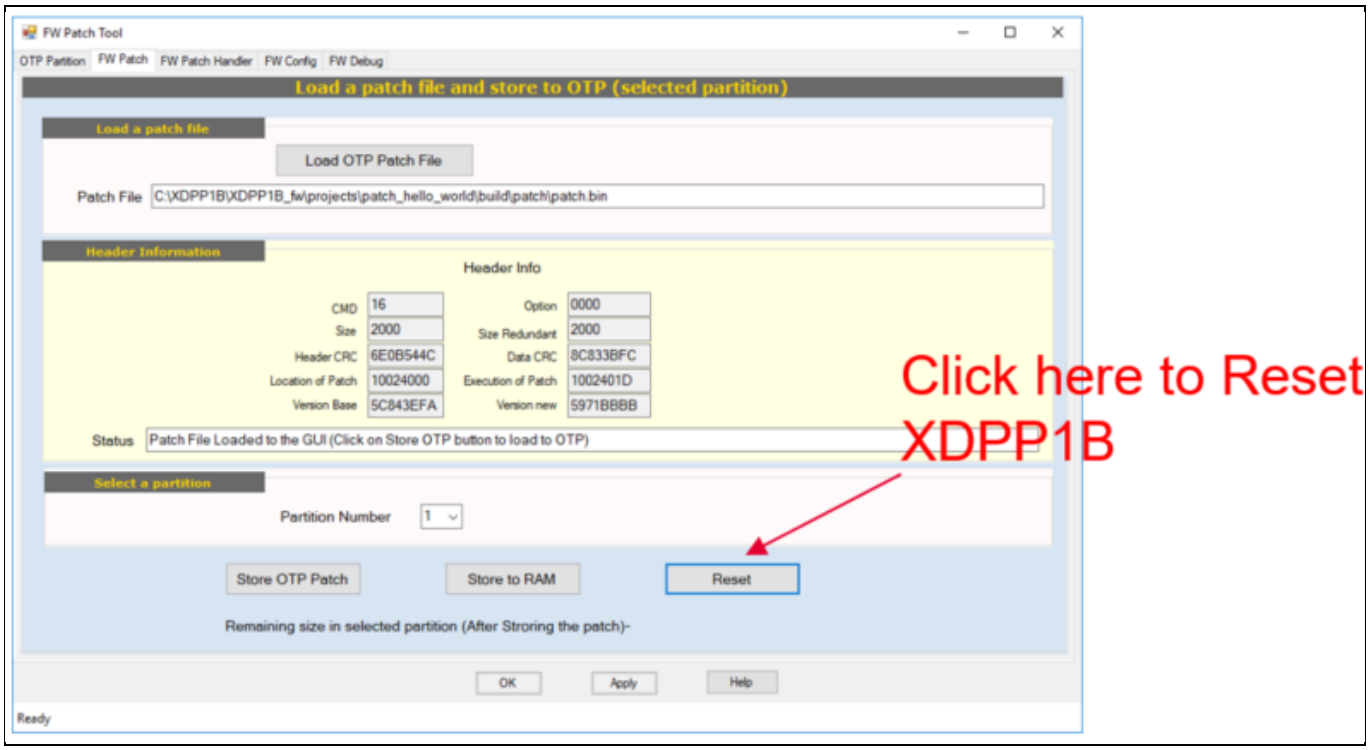


7.2 STEP 2: Monitor UART traffic



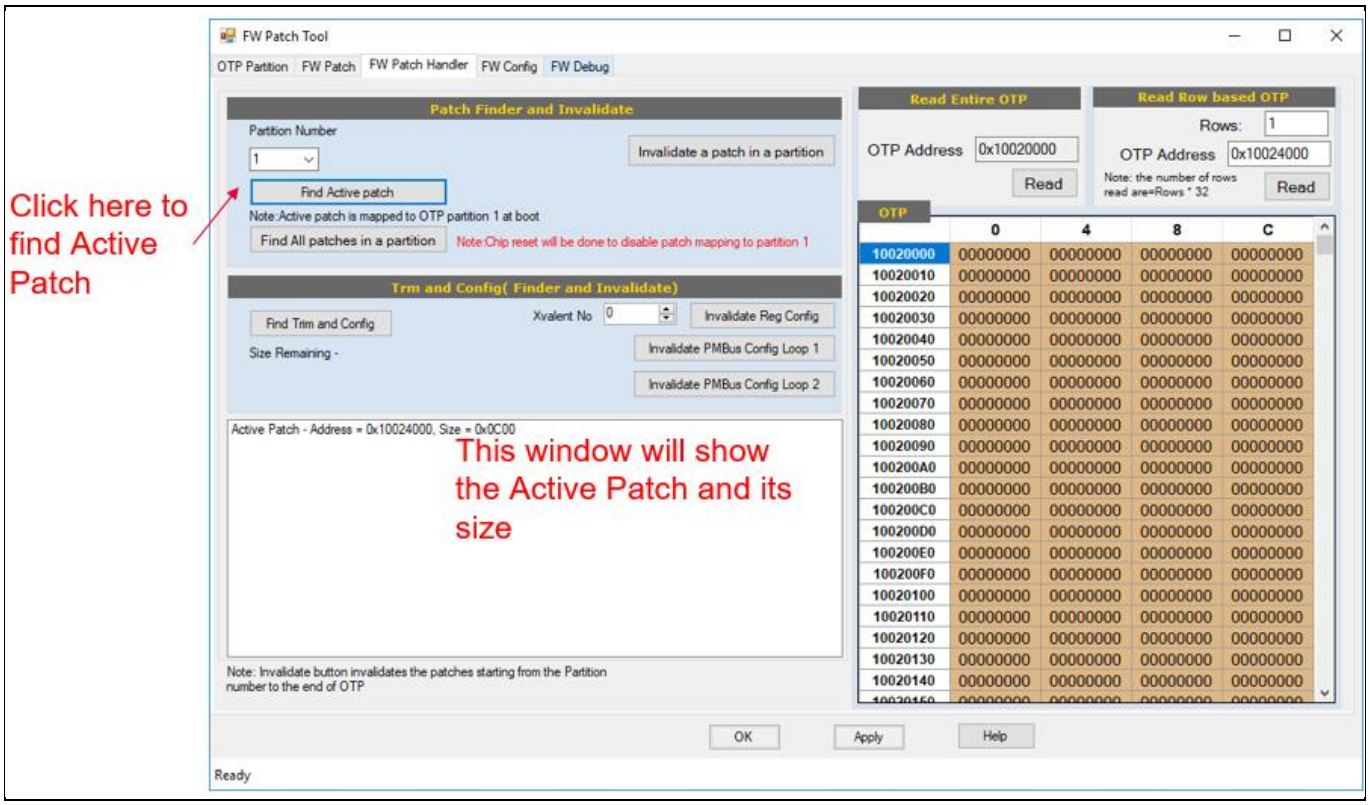
7.3 STEP 3: “Hello World” will appear



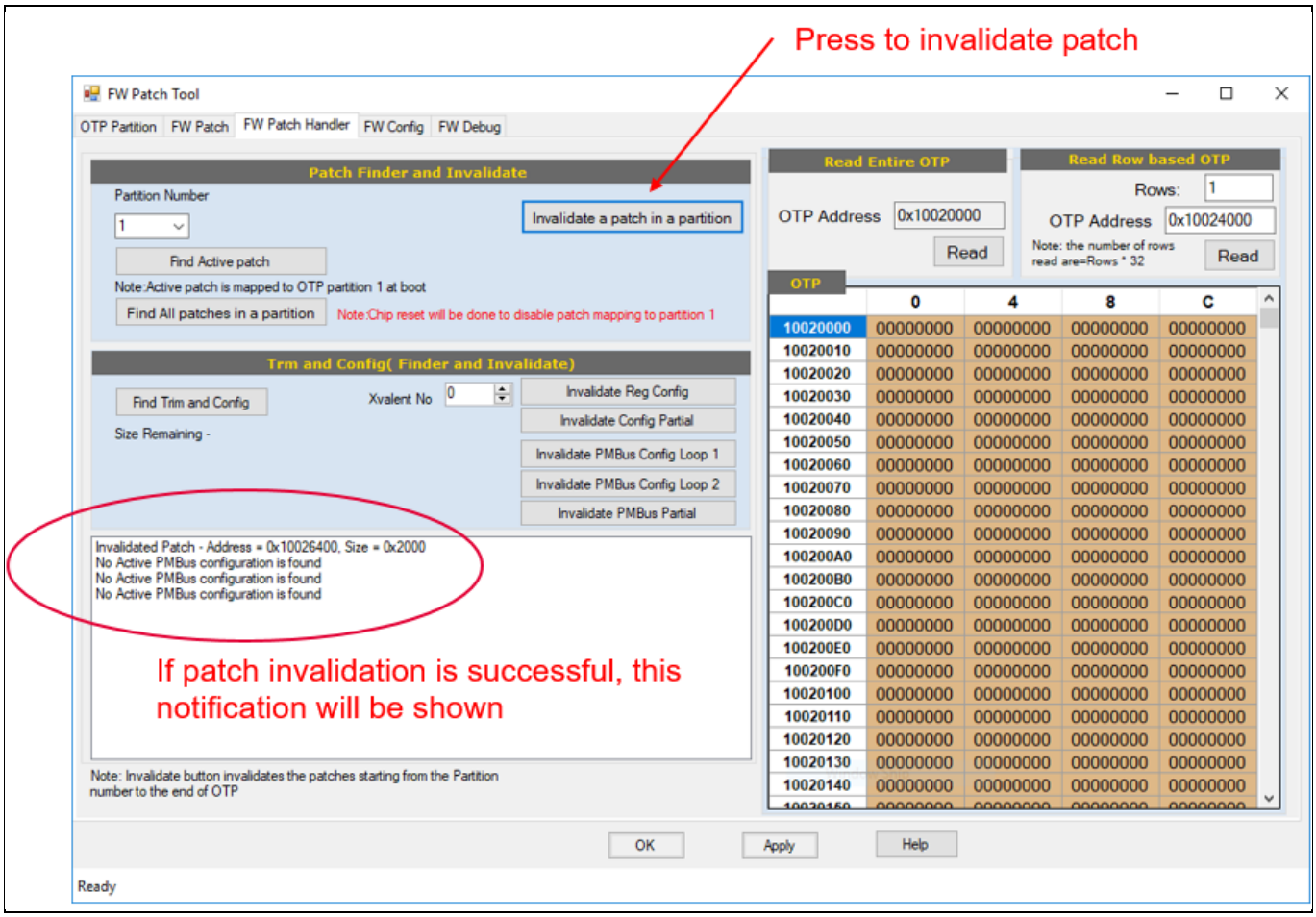


8 Invalidating Patch from OTP

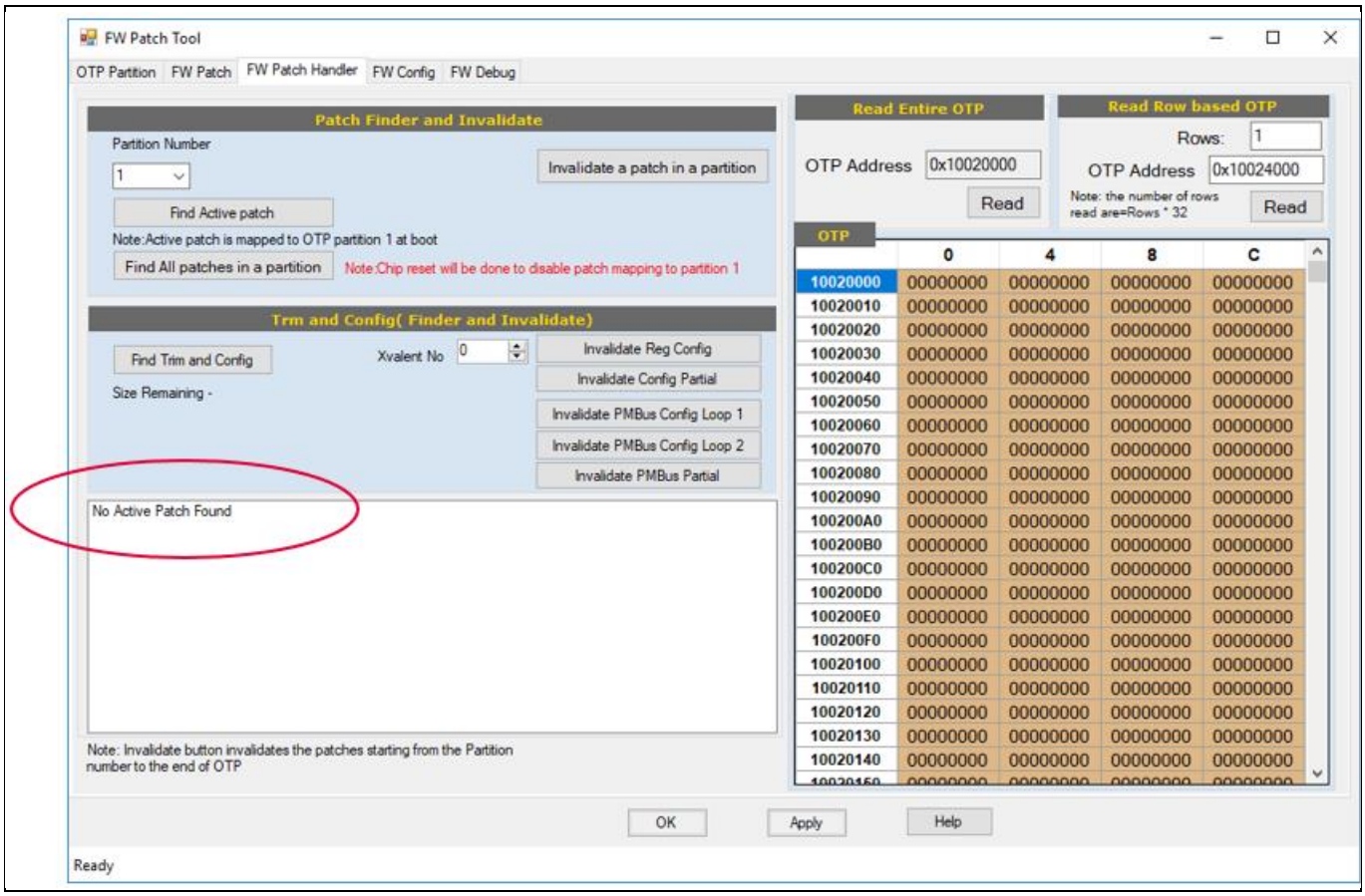
8.1 STEP 1: Ensure there is an active Patch in the OTP



8.2 STEP 2: Invalidate the Patch in OTP



8.3 STEP 3: Verify that Patch has been invalidated



8.3.1 Notes

When patch is invalidated from OTP, it will also invalidate any existing PMBus configuration.

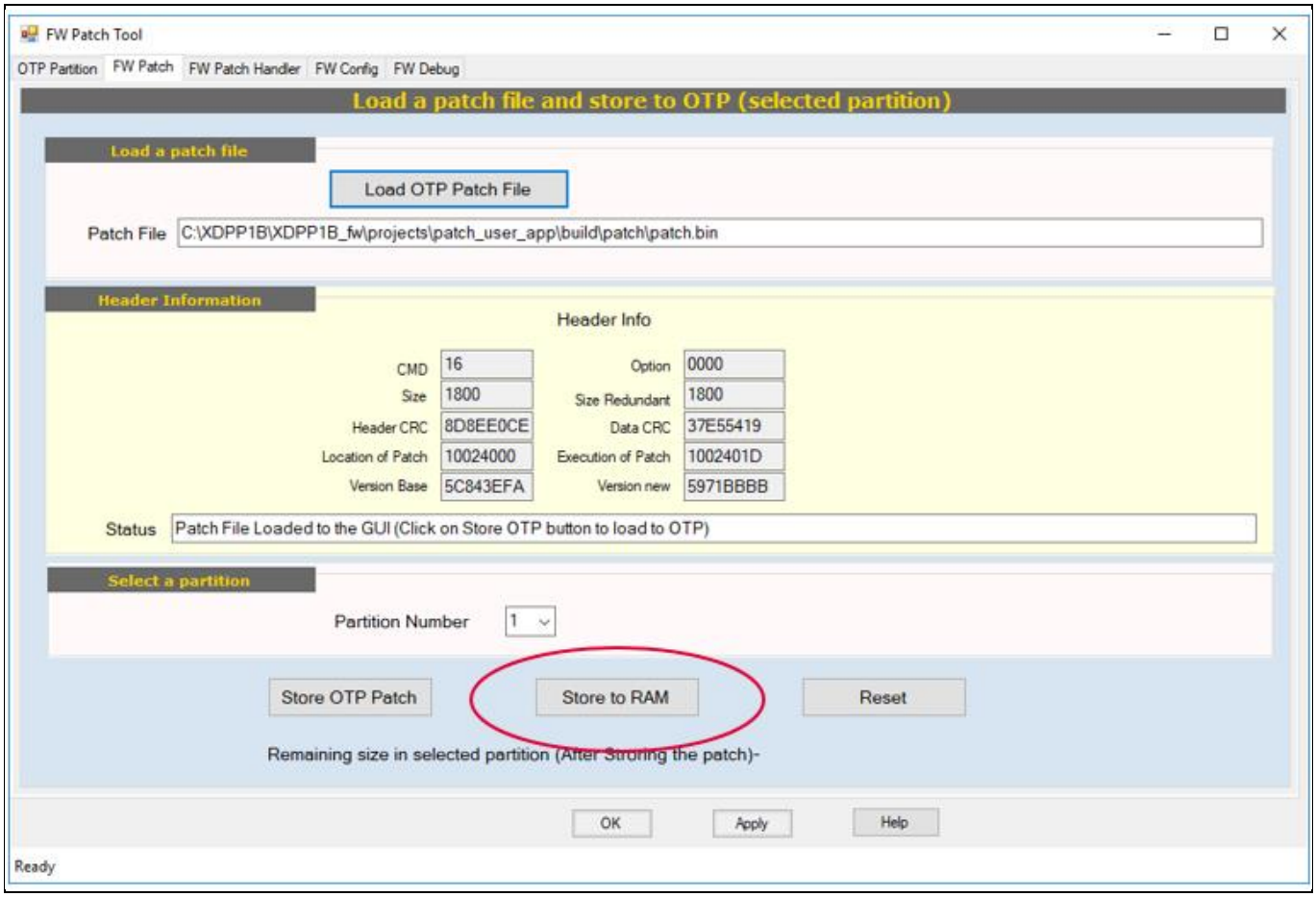
When PMBus configuration is programmed to the OTP, it is programmed at the same section with active patch. When the patch is invalidated, both FW patch section as well as PMBus configuration section are invalidated. Therefore, user needs to redownload the PMBus configuration again after an active patch has been invalidated.

9 Running Patch from RAM

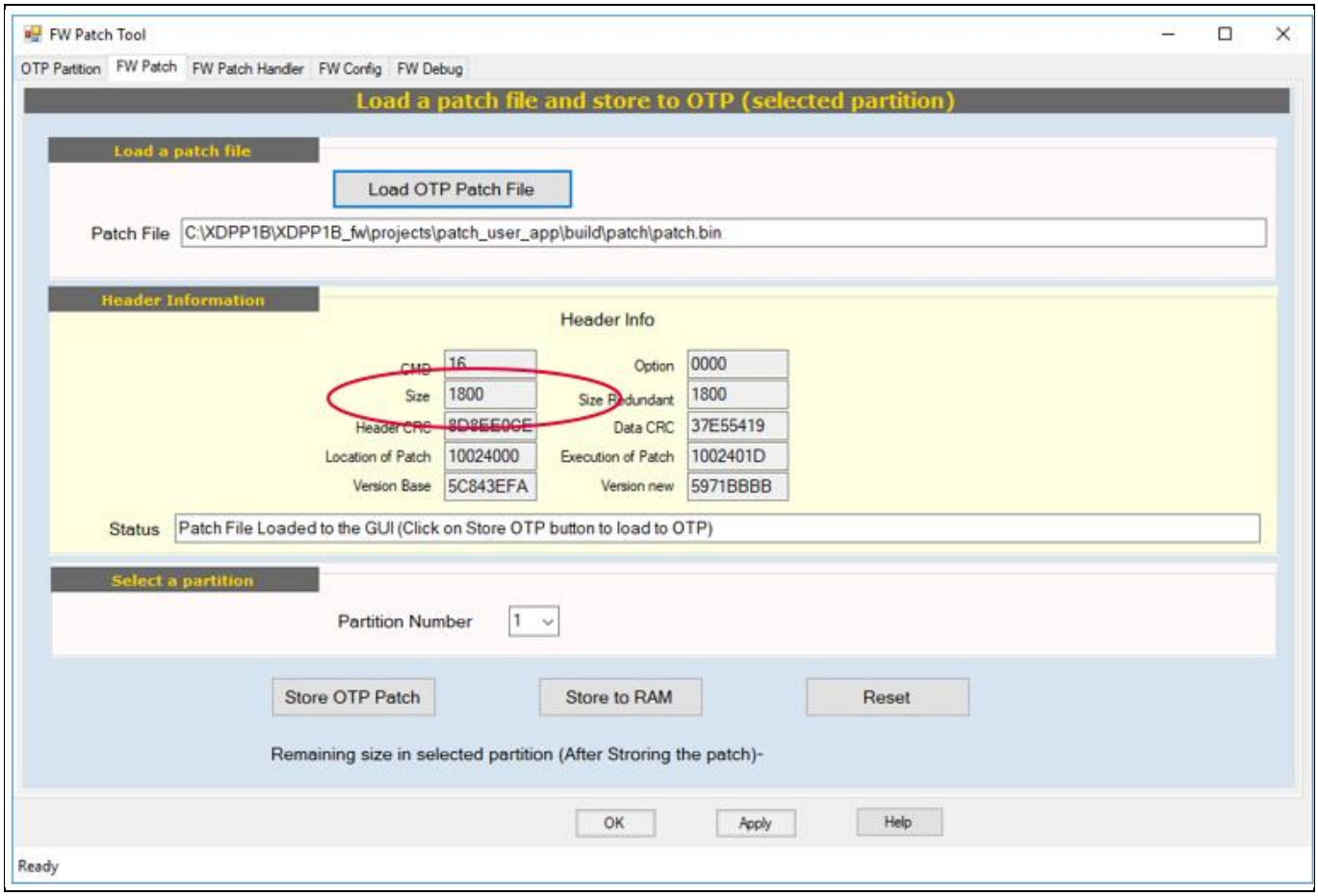
In order to run patch in RAM, user has to modify user_drv_init() at user_app.c with the following codes as shown in the following figure:

```
void user_drv_init(void)
{
    memset(&user_data, 0, sizeof(USER_DATA_t));  // ZI the user data
    //patch_rom_table.patch_FAULTS_Resolve_Shutdown_Priority = &FAULTS_User_Resolve_Shutdown_Priority;
    ntc_temp = &user_ntc_temp_lut[0];  // set the ntc lut to rom constant table
    // this is the initialization of user pmbus commands autogenerated from pmbus spreadsheet
#ifdef UART_DEBUG
    ptr_mfr_specific_init = (mfr_specific_init_ptr) patch_pmbus_mfr_autogen_init;
#else
    ptr_mfr_specific_init = (mfr_specific_init_ptr) pmbus_mfr_autogen_init;
#endif
    set_module_init_cb(MODULE_REGULATION, regulation_sm_callbacks_init);  // set the pointer funtion in
    /*lint -e522 */
    add_on_features_init();
    ///////////////debug using ramp data circular buffer example//////////////
    //

    regulation_sm_callbacks_init();
    pmbus_mfr_autogen_init();

    // create a user thread to execute the fan_pid function periodically
    //tid_USER_Thread = osExtThreadCreate(osThread(USER_Thread), NULL);
    //THROW_IF((tid_USER_Thread == NULL), OUT_OF_RESOURCE); /* osErrorNoMemory */
}
```

Once the source codes are compiled, user can then download the patch to RAM as shown in the following figure:


FW Patch Tool
OTP Partition
FW Patch
FW Patch Handler
FW Config
FW Debug
Load a patch file and store to OTP (selected partition)
Load a patch file
Load OTP Patch File
Patch File
C:\XDPP1B\XDPP1B_fw\projects\patch_user_app\build\patch\patch.bin
Header Information
Header Info
CMD
16
Option
0000
Size
1800
Size Redundant
1800
Header CRC
8D8EE0CE
Data CRC
37E55419
Location of Patch
10024000
Execution of Patch
1002401D
Version Base
5C843EFA
Version new
5971BBBB
Status
Patch File Loaded to the GUI (Click on Store OTP button to load to OTP)
Select a partition
Partition Number
1
Store OTP Patch
Store to RAM
Reset
Remaining size in selected partition (After Storing the patch)-
OK
Apply
Help
Ready

Note: The maximum patch size can be downloaded into RAM is 8k bytes. The patch size can be seen in FW Patch Tool, as shown in the following figure. The size is 1800 is the patch size expressed in Hexadecimal, i.e. ~6k Byte. For 8k Byte size, the Size will display 2000. If the patch size exceeds 8k Byte, user can test features in RAM by disabling certain features in their patch.

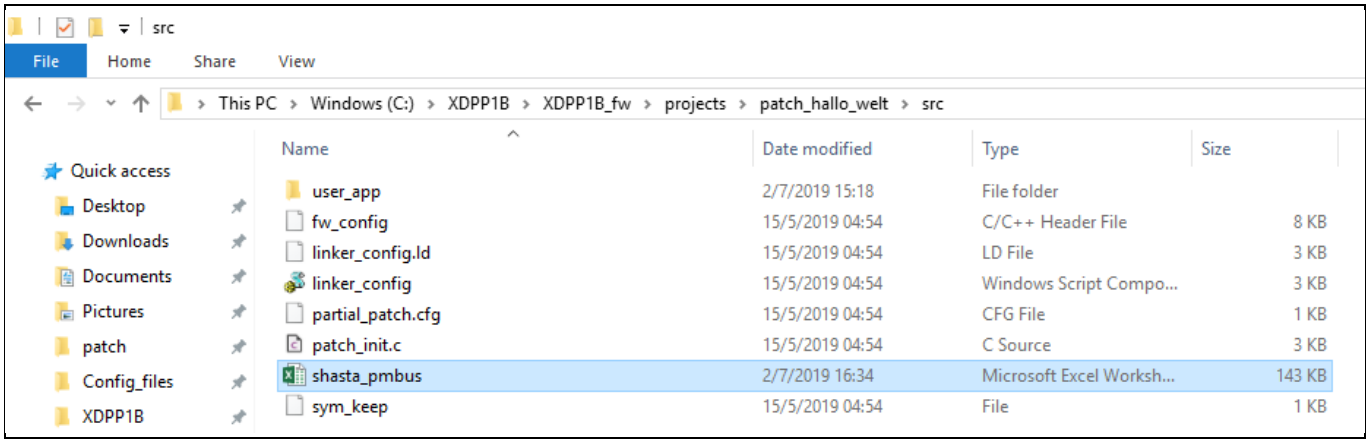


10 Implementing Custom PMBus Command in XDPP1100

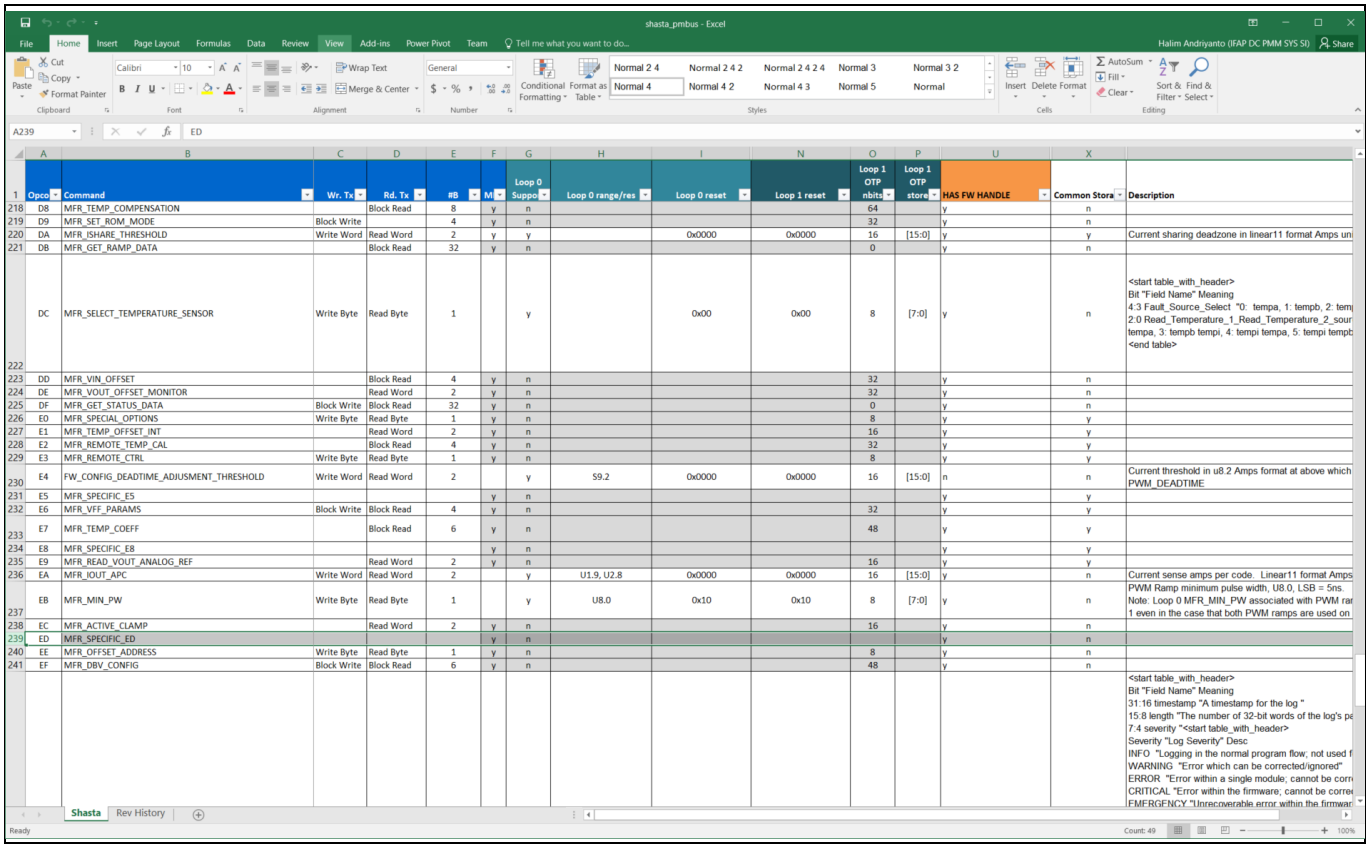
This section demonstrate on how to implement a custom PMBus command in XDPP1100. Please create a new project called “patch_hallo_welt”. The steps on creating a new project can be found in the previous section.

10.1 Adding new PMBus MFR command in the spreadsheet

10.1.1 **STEP 1: Open PMBus Spreadsheet “shasta_pmbus.xlsx”**

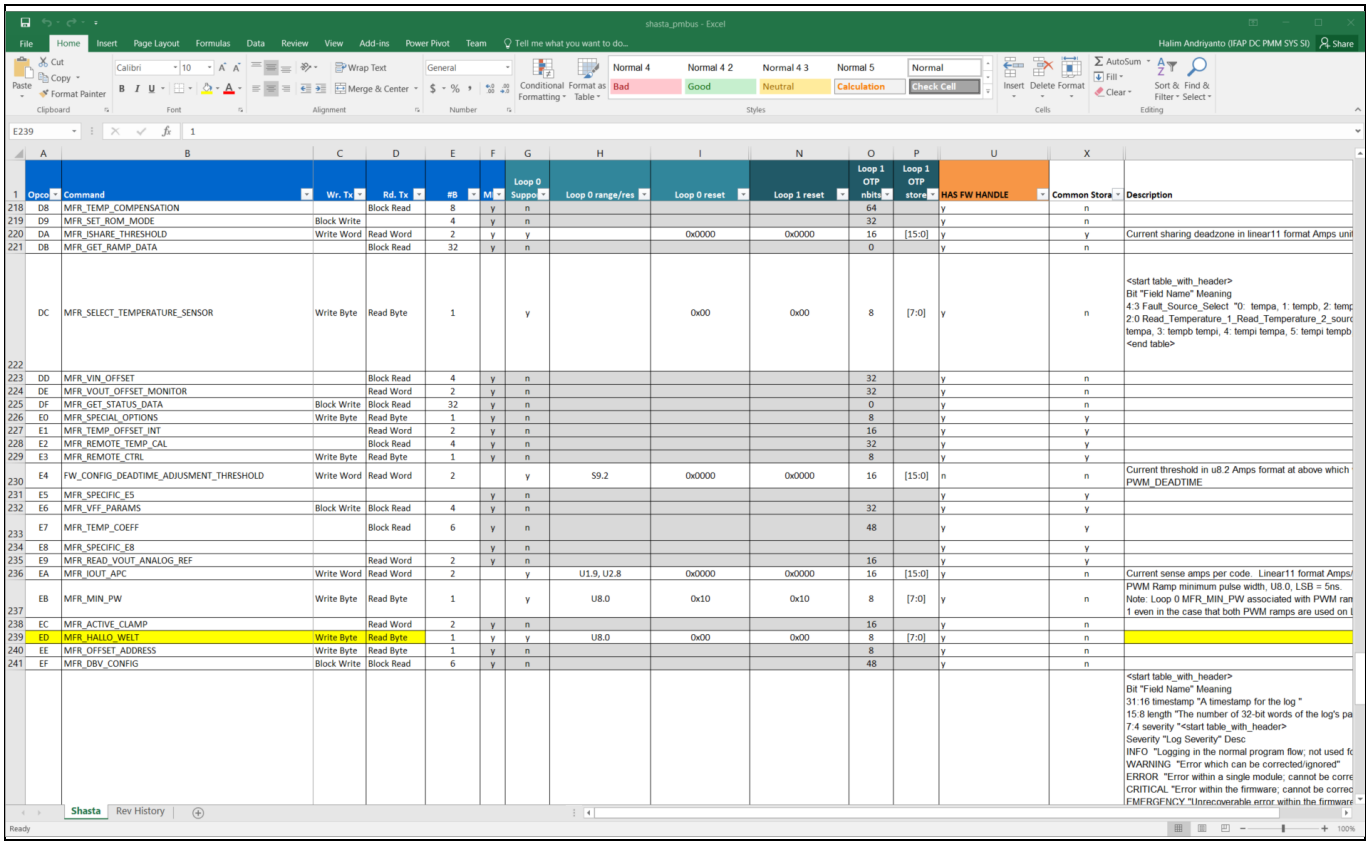


10.1.2 STEP 2: Select the empty MFR command slot, e.g. 0xED



	A	B	C	D	E	F	G	H	I	N	O	P	U	X	
1	Opco	Command	Wr. Tx	Rd. Tx	#B	M	Loop 0 Suppo	Loop 0 range/res	Loop 0 reset	Loop 1 reset	Loop 1 OTP nbits	Loop 1 OTP store	HAS FW HANDLE	Common Stora	Description
218	D8	MFR_TEMP_COMPENSATION		Block Read	8	y	n				64		y	n	
219	D9	MFR_SET_ROM_MODE	Block Write		4	y	n				32		y	n	
220	DA	MFR_ISHARE_THRESHOLD	Write Word	Read Word	2	y	y		0x0000	0x0000	16	[15:0]	y	y	Current sharing deadzone in linear11 format Amps un
221	DB	MFR_GET_RAMP_DATA		Block Read	32	y	n				0		y	n	
222	DC	MFR_SELECT_TEMPERATURE_SENSOR	Write Byte	Read Byte	1		y		0x00	0x00	8	[7:0]	y	n	<start table_with_header> Bit "Field Name" Meaning 4:3 Fault_Source_Select "0: tempa, 1: tempb, 2: tem 2:0 Read_Temperature_1_Read_Temperature_2_sour tempa, 3: tempb tempi, 4: tempi tempa, 5: tempi tempb <end table>
223	DD	MFR_VIN_OFFSET		Block Read	4	y	n				32		y	n	
224	DE	MFR_VOUT_OFFSET_MONITOR		Read Word	2	y	n				32		y	n	
225	DF	MFR_GET_STATUS_DATA	Block Write	Block Read	32	y	n				0		y	n	
226	E0	MFR_SPECIAL_OPTIONS	Write Byte	Read Byte	1	y	n				8		y	y	
227	E1	MFR_TEMP_OFFSET_INT		Read Word	2	y	n				16		y	y	
228	E2	MFR_REMOTE_TEMP_CAL		Block Read	4	y	n				32		y	y	
229	E3	MFR_REMOTE_CTRL	Write Byte	Read Byte	1	y	n				8		y	y	
230	E4	FW_CONFIG_DEADTIME_ADJUSMENT_THRESHOLD	Write Word	Read Word	2		y	S9.2	0x0000	0x0000	16	[15:0]	n	n	Current threshold in u8.2 Amps format at above which PWM_DEADTIME
231	E5	MFR_SPECIFIC_E5				y	n						y	y	
232	E6	MFR_VFF_PARAMS	Block Write	Block Read	4	y	n				32		y	y	
233	E7	MFR_TEMP_COEFF		Block Read	6	y	n				48		y	y	
234	E8	MFR_SPECIFIC_E8				y	n						y	y	
235	E9	MFR_READ_VOUT_ANALOG_REF		Read Word	2	y	n				16		y	y	
236	EA	MFR_IOUT_APC	Write Word	Read Word	2		y	U1.9, U2.8	0x0000	0x0000	16	[15:0]	y	n	Current sense amps per code. Linear11 format Amps
237	EB	MFR_MIN_PW	Write Byte	Read Byte	1		y	U8.0	0x10	0x10	8	[7:0]	y	n	PWM Ramp minimum pulse width, U8.0, LSB = 5ns. Note: Loop 0 MFR_MIN_PW associated with PWM ra 1 even in the case that both PWM ramps are used on
238	EC	MFR_ACTIVE_CLAMP		Read Word	2	y	n				16		y	n	
239	ED	MFR_SPECIFIC_ED				y	n						y	n	
240	EE	MFR_OFFSET_ADDRESS	Write Byte	Read Byte	1	y	n				8		y	n	
241	EF	MFR_DBV_CONFIG	Block Write	Block Read	6	y	n				48		y	n	
															<start table_with_header> Bit "Field Name" Meaning 31:16 timestamp "A timestamp for the log " 15:8 length "The number of 32-bit words of the log's pa 7:4 severity "<start table_with_header> Severity "Log Severity" Desc INFO "Logging in the normal program flow; not used f WARNING "Error which can be corrected/ignored" ERROR "Error within a single module; cannot be corr CRITICAL "Error within the firmware; cannot be corre EMERGENCY "Unrecoverable error within the firmwar

10.1.3 STEP 3: Modify the PMBus command accordingly. Save the file.

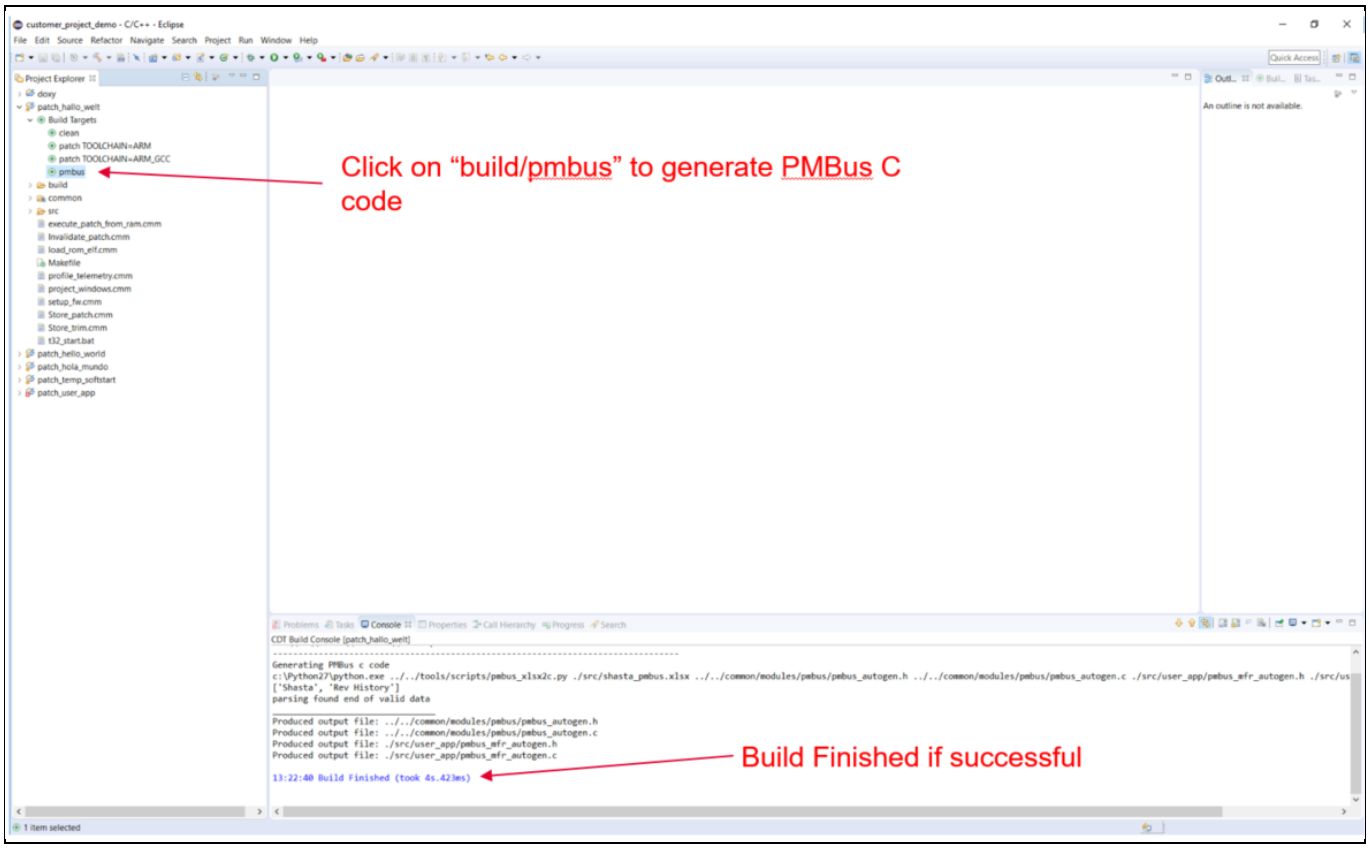


	A	B	C	D	E	F	G	H	I	N	O	P	U	X	
1	Opco	Command	Wr. Tx	Rd. Tx	#B	M	Loop 0 Suppo	Loop 0 range/res	Loop 0 reset	Loop 1 reset	Loop 1 OTP nbits	Loop 1 OTP store	HAS FW HANDLE	Common Stora	Description
218	D8	MFR_TEMP_COMPENSATION		Block Read	8	y	n				64		y	n	
219	D9	MFR_SET_ROM_MODE	Block Write		4	y	n				32		y	n	
220	DA	MFR_ISHARE_THRESHOLD	Write Word	Read Word	2	y	y		0x0000	0x0000	16	[15:0]	y	y	Current sharing deadzone in linear11 format Amps unit
221	DB	MFR_GET_RAMP_DATA		Block Read	32	y	n				0		y	n	
222	DC	MFR_SELECT_TEMPERATURE_SENSOR	Write Byte	Read Byte	1		y		0x00	0x00	8	[7:0]	y	n	<start table_with_header> Bit "Field Name" Meaning 4:3 Fault_Source_Select "0: tempa, 1: tempb, 2: temp 2:0 Read_Temperature_1_Read_Temperature_2_sourc tempa, 3: tempb tempi, 4: tempi tempa, 5: tempi tempb <end table>
223	DD	MFR_VIN_OFFSET		Block Read	4	y	n				32		y	n	
224	DE	MFR_VOUT_OFFSET_MONITOR		Read Word	2	y	n				32		y	n	
225	DF	MFR_GET_STATUS_DATA	Block Write	Block Read	32	y	n				0		y	n	
226	E0	MFR_SPECIAL_OPTIONS	Write Byte	Read Byte	1	y	n				8		y	y	
227	E1	MFR_TEMP_OFFSET_INT		Read Word	2	y	n				16		y	y	
228	E2	MFR_REMOTE_TEMP_CAL		Block Read	4	y	n				32		y	y	
229	E3	MFR_REMOTE_CTRL	Write Byte	Read Byte	1	y	n				8		y	y	
230	E4	FW_CONFIG_DEADTIME_ADJUSMENT_THRESHOLD	Write Word	Read Word	2		y	S9.2	0x0000	0x0000	16	[15:0]	n	n	Current threshold in u8.2 Amps format at above which PWM_DEADTIME
231	E5	MFR_SPECIFIC_E5				y	n						y	y	
232	E6	MFR_VFF_PARAMS	Block Write	Block Read	4	y	n				32		y	y	
233	E7	MFR_TEMP_COEFF		Block Read	6	y	n				48		y	y	
234	E8	MFR_SPECIFIC_E8				y	n						y	y	
235	E9	MFR_READ_VOUT_ANALOG_REF		Read Word	2	y	n				16		y	y	
236	EA	MFR_IOUT_APC	Write Word	Read Word	2		y	U1.9, U2.8	0x0000	0x0000	16	[15:0]	y	n	Current sense amps per code. Linear11 format Amps/
237	EB	MFR_MIN_PW	Write Byte	Read Byte	1		y	U8.0	0x10	0x10	8	[7:0]	y	n	PWM Ramp minimum pulse width, U8.0, LSB = 5ns. Note: Loop 0 MFR_MIN_PW associated with PWM ran 1 even in the case that both PWM ramps are used on t
238	EC	MFR_ACTIVE_CLAMP		Read Word	2	y	n				16		y	n	
239	ED	MFR_HALLO_WELT	Write Byte	Read Byte	1	y	y	U8.0	0x00	0x00	8	[7:0]	y	n	
240	EE	MFR_OFFSET_ADDRESS	Write Byte	Read Byte	1	y	n				8		y	n	
241	EF	MFR_DBV_CONFIG	Block Write	Block Read	6	y	n				48		y	n	
															<start table_with_header> Bit "Field Name" Meaning 31:16 timestamp "A timestamp for the log " 15:8 length "The number of 32-bit words of the log's pa 7:4 severity "<start table_with_header> Severity "Log Severity" Desc INFO "Logging in the normal program flow; not used fo WARNING "Error which can be corrected/ignored" ERROR "Error within a single module; cannot be corre CRITICAL "Error within the firmware; cannot be correc EMERGENCY "Unrecoverable error within the firmware

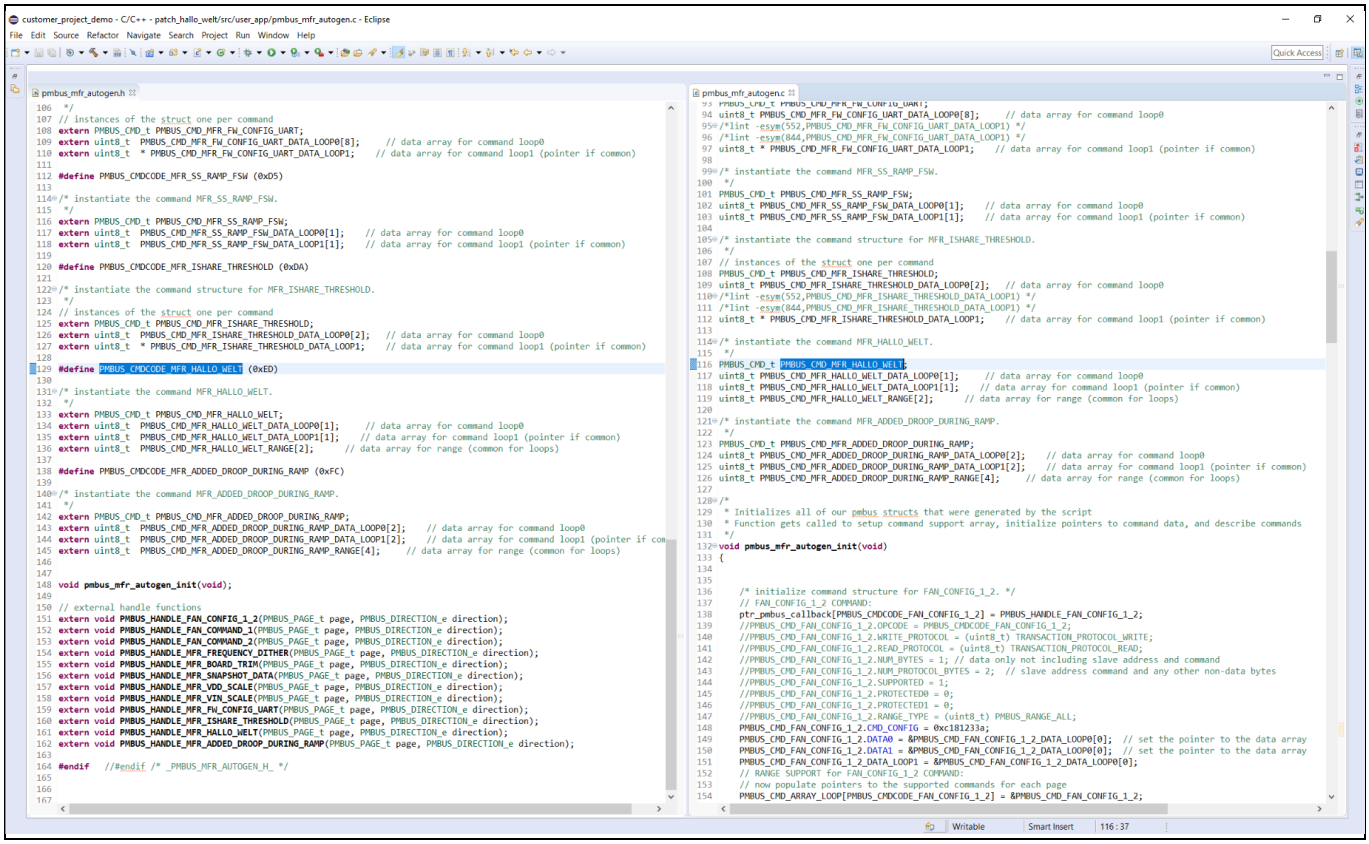
10.2 Generate PMBus codes in the project

If not working, check python path

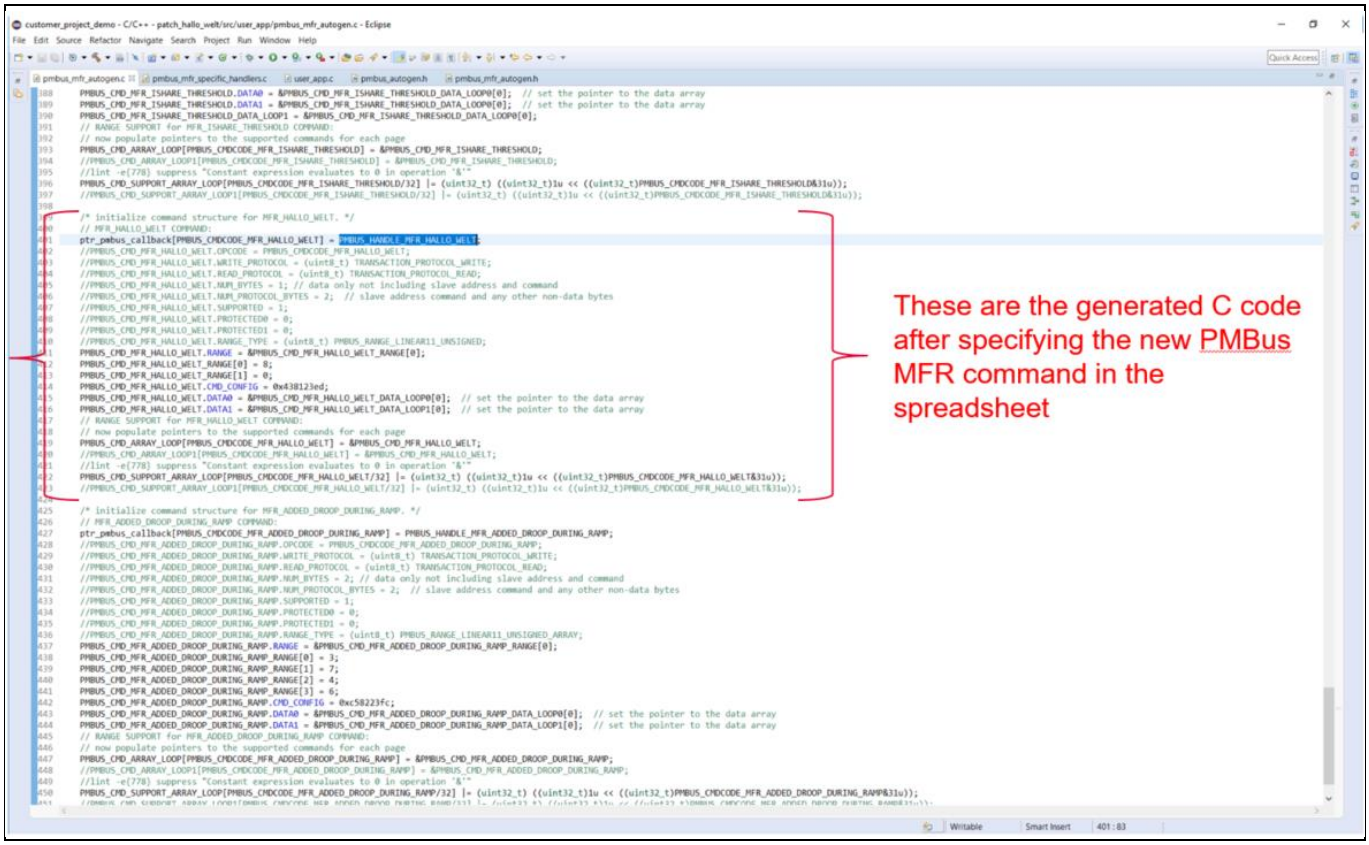
10.2.1 **STEP 1: Click on “build/pmbus” to generate PMBus C code**



10.2.2 STEP 2: Open “pmbus_mfr_autogen.c/h” for generated codes

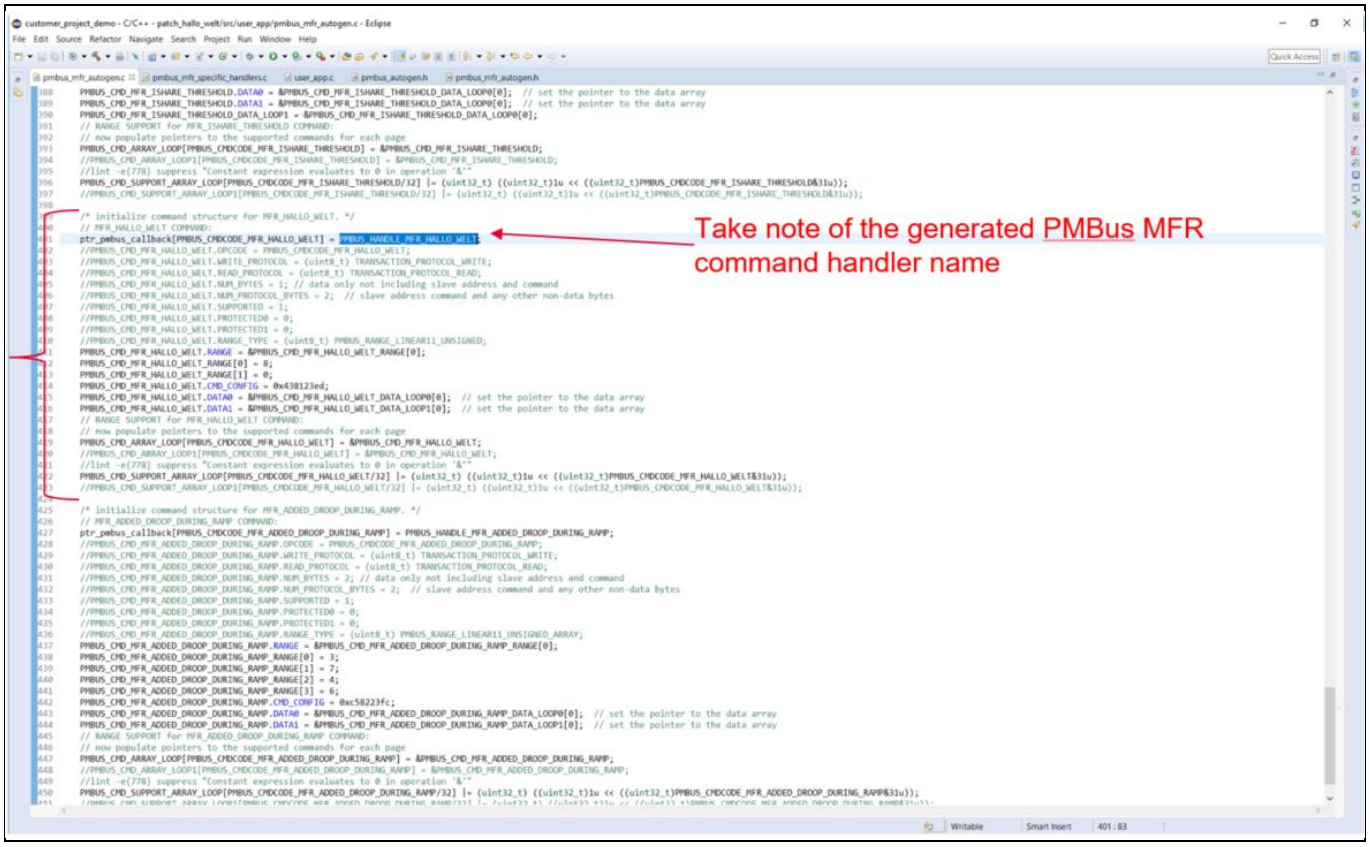


10.2.3 STEP 3: Scroll down at “pmbus_mfr_autogen.c”

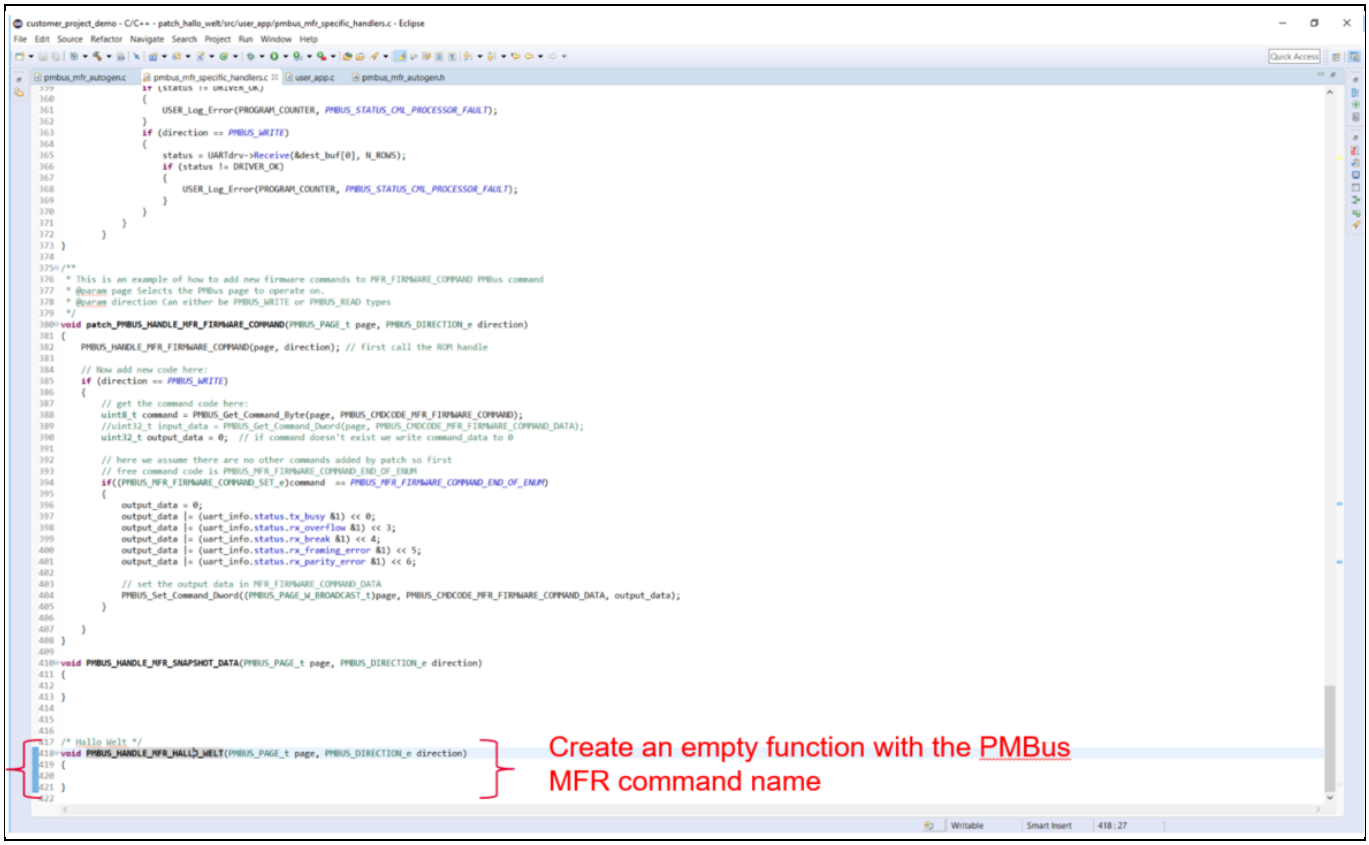


10.3 Implement PMBus MFR command handler

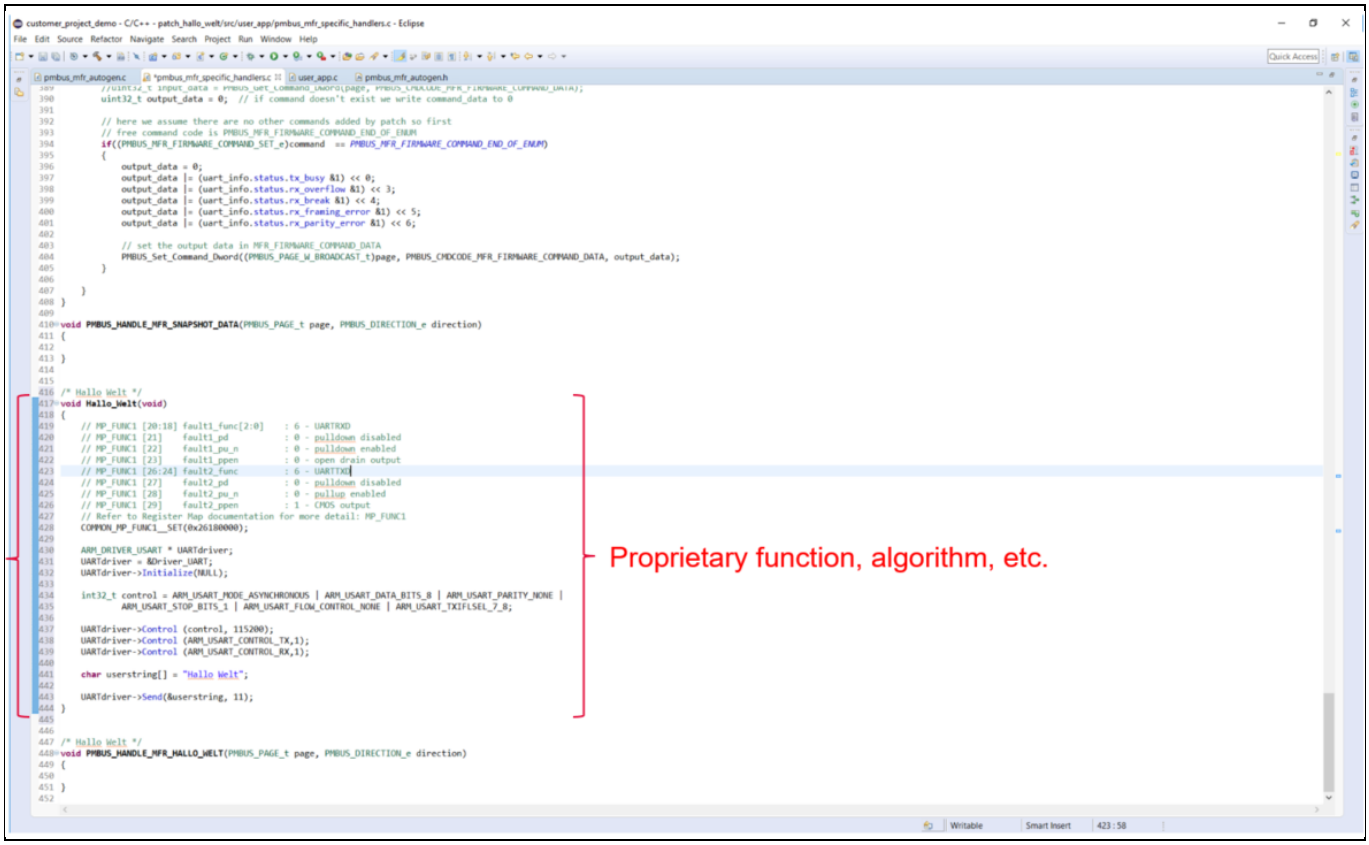
10.3.1 STEP 1: Note the PMBus Command Handler name



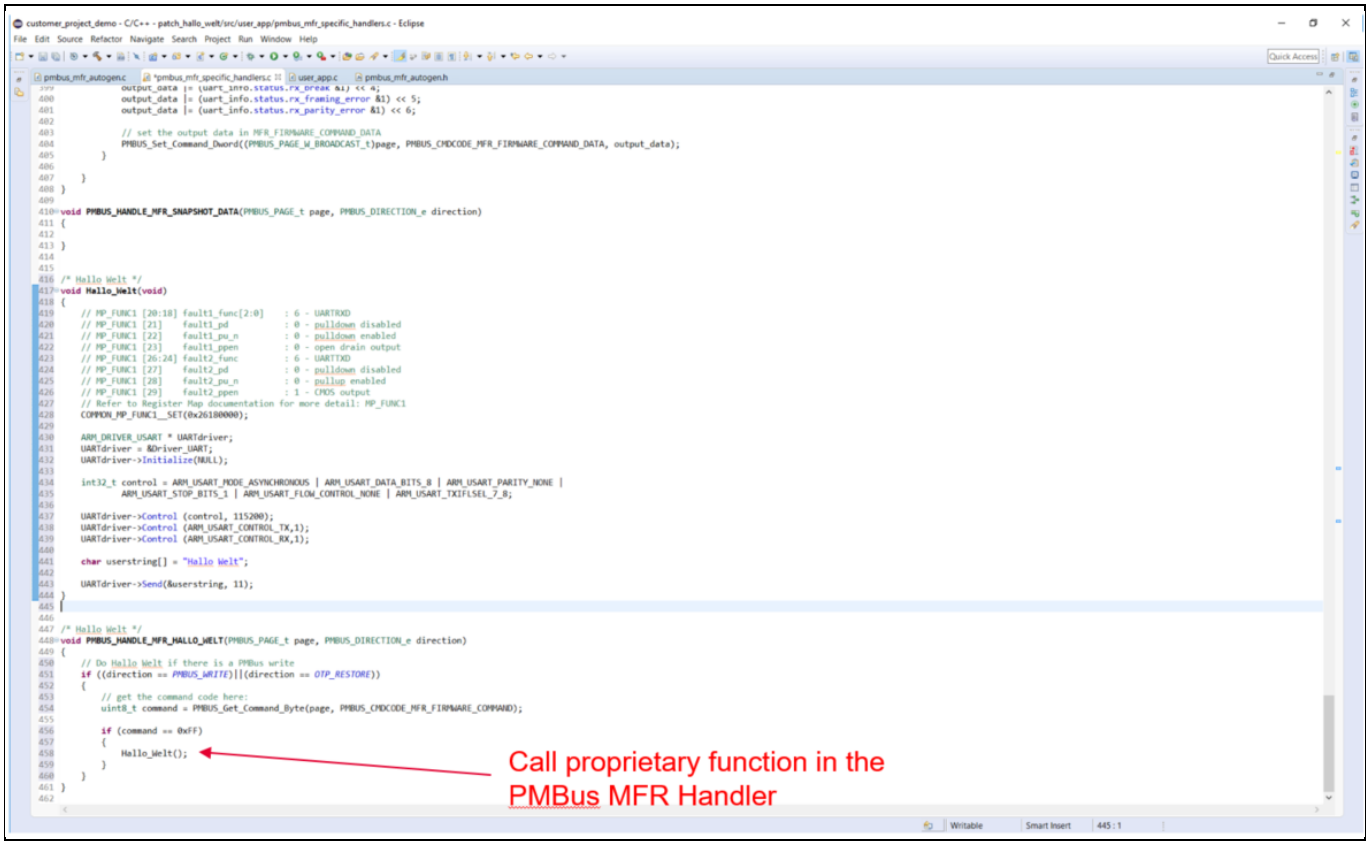
10.3.2 STEP 2: Modify “pmbus_mfr_specific_handler.c”



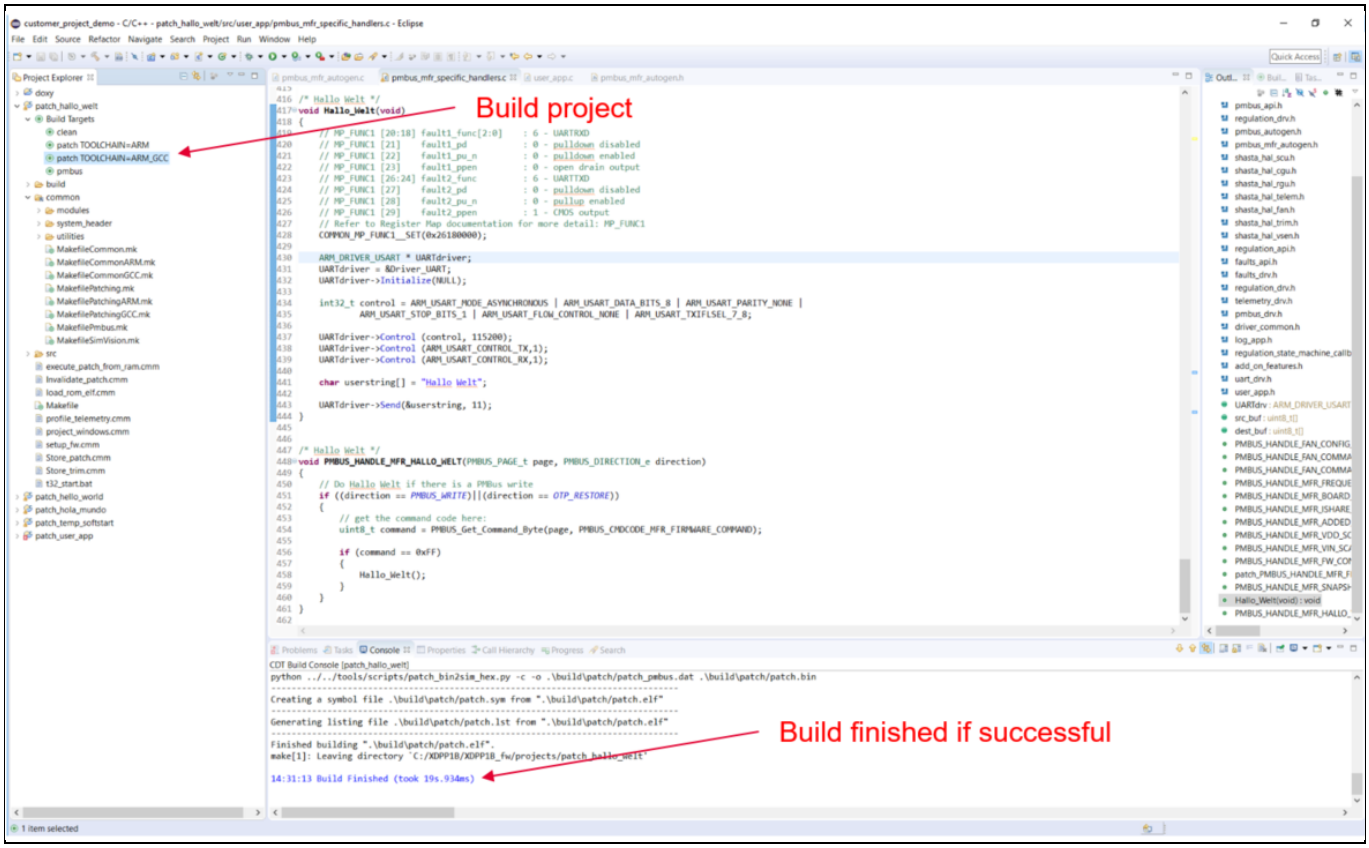
10.3.3 STEP 3: Put in a custom function above the handler



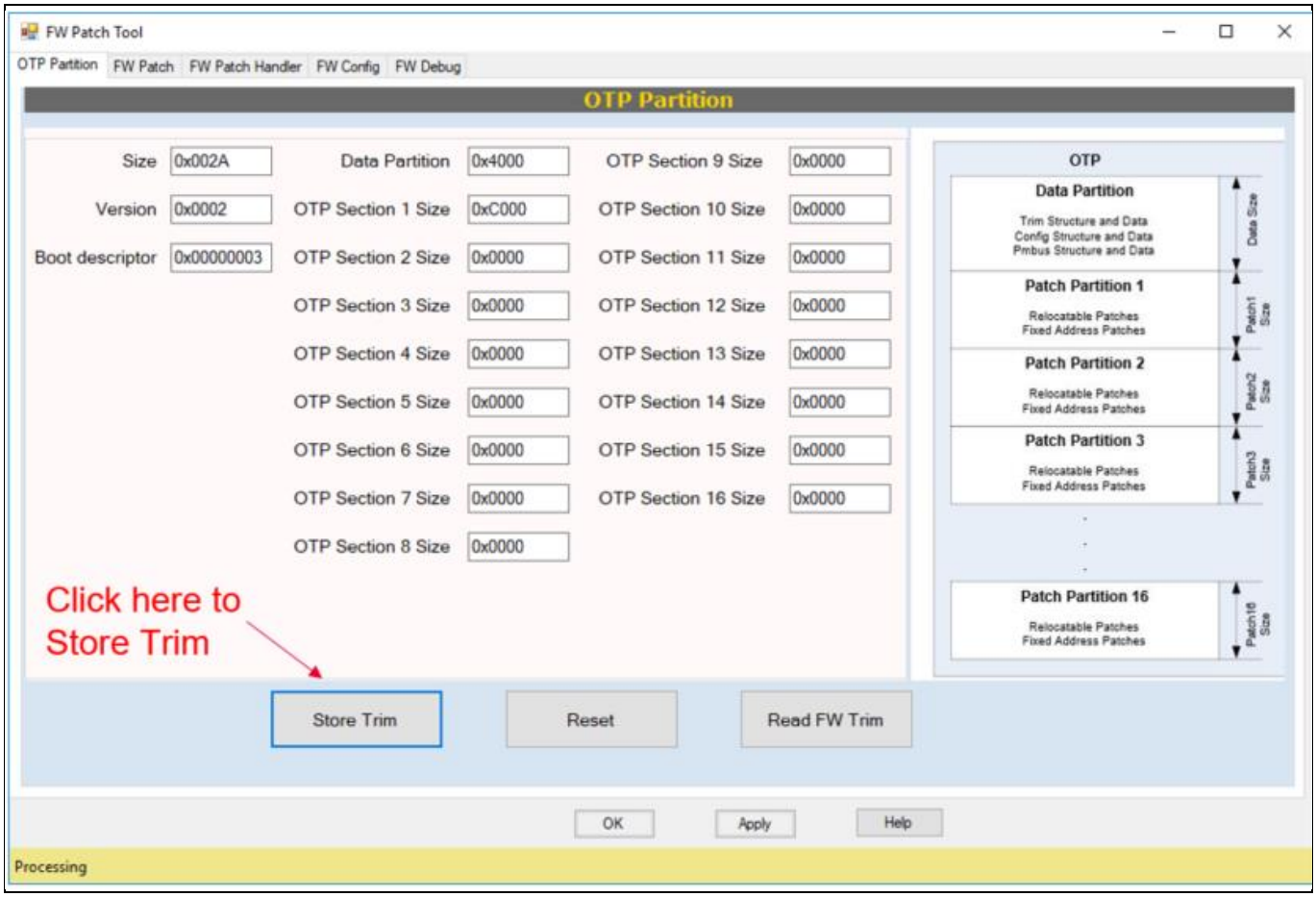
10.3.4 STEP 4: Call your function in the PMBus MFR handler

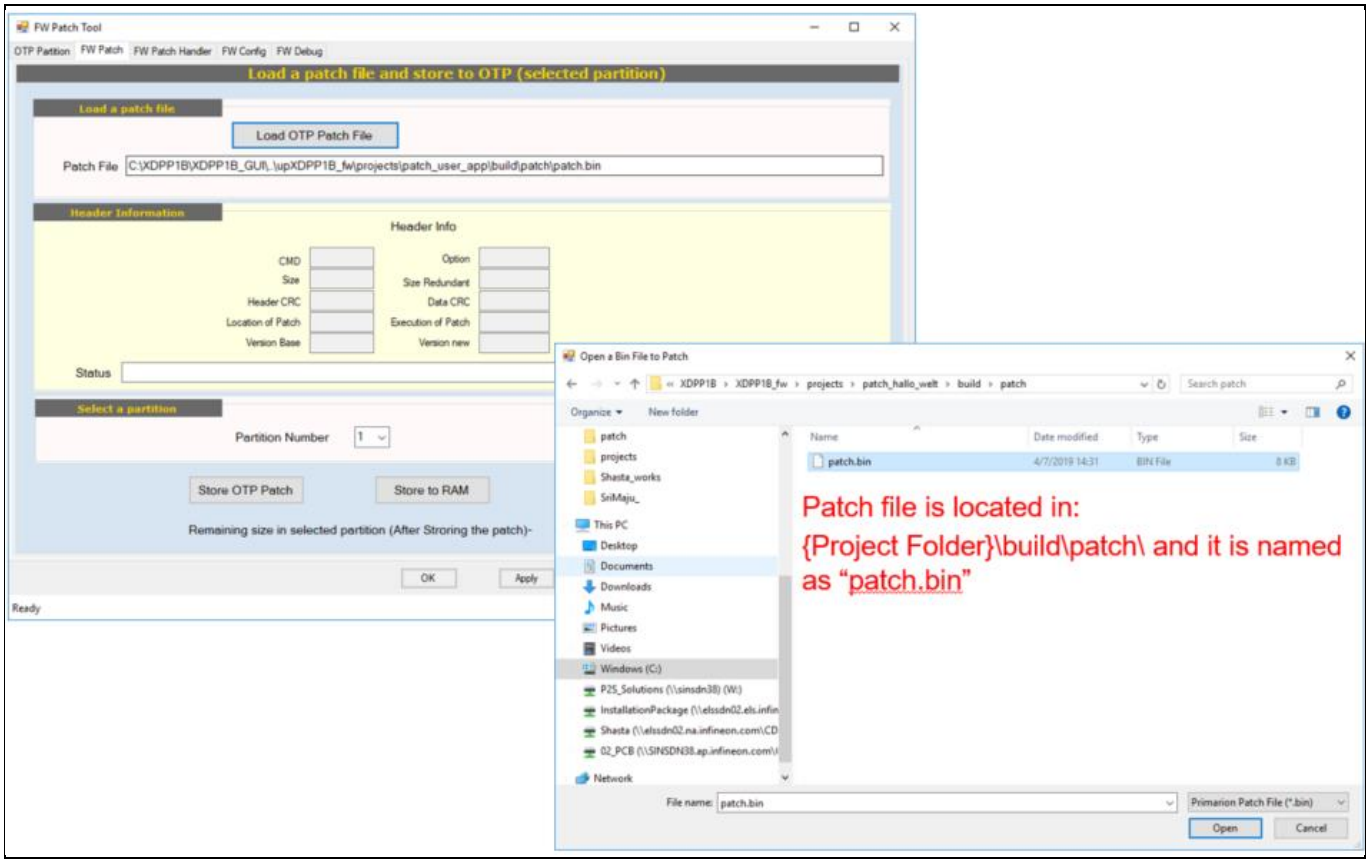


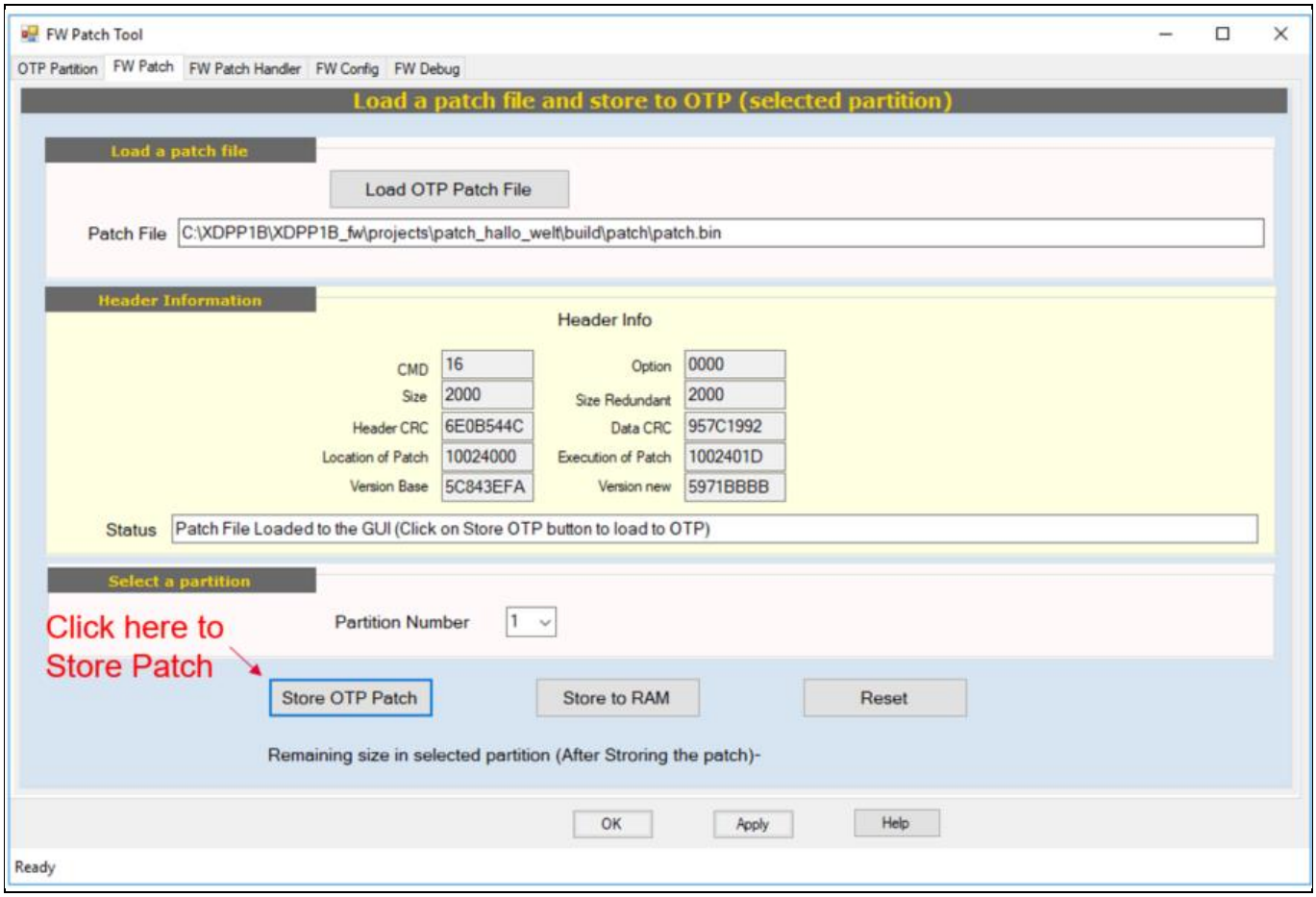
10.3.5 STEP 5: Clean and Build project.



10.3.6 STEP 6: Download the Patch

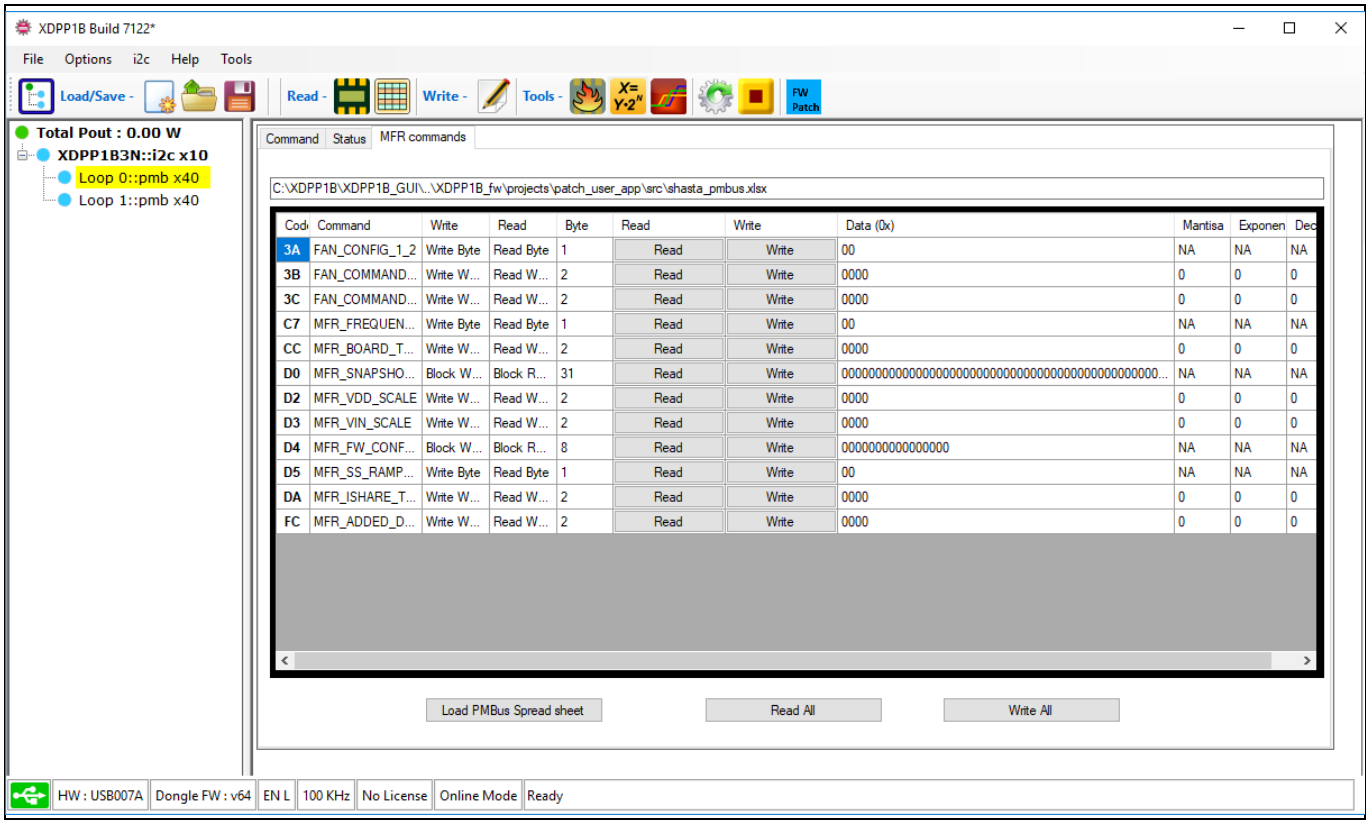



FW Patch Tool
OTP Partition
FW Patch
FW Patch Handler
FW Config
FW Debug
Load a patch file and store to OTP (selected partition)
Load a patch file
Load OTP Patch File
Patch File
C:\XDPP1B\XDPP1B_GUI\..\upXDPP1B_fw\projects\patch_user_app\build\patch\patch.bin
Header Information
Header Info
CMD
Size
Header CRC
Location of Patch
Version Base
Option
Size Redundant
Data CRC
Execution of Patch
Version new
Status
Select a partition
Partition Number
Store OTP Patch
Store to RAM
Remaining size in selected partition (After Stroring the patch)-
OK
Apply
Ready
Open a Bin File to Patch
XDPP1B
XDPP1B_fw
projects
patch_hallo_welt
build
patch
Search patch
Organize
New folder
patch
projects
Shasta_works
SriMaju_
This PC
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows (C:)
P25_Solutions (\\sinsdn38) (W:)
Network
Name
Date modified
Type
Size
patch.bin
4/7/2019 14:31
BIN File
8 KB
Patch file is located in:
{Project Folder}\build\patch\ and it is named as “patch.bin”
File name:
patch.bin
Primarion Patch File (*.bin)
Open
Cancel


FW Patch Tool
OTP Partition
FW Patch
FW Patch Handler
FW Config
FW Debug
Load a patch file and store to OTP (selected partition)
Load a patch file
Load OTP Patch File
Patch File
C:\XDPP1B\XDPP1B_fw\projects\patch_hallo_welt\build\patch\patch.bin
Header Information
Header Info
CMD
16
Option
0000
Size
2000
Size Redundant
2000
Header CRC
6E0B544C
Data CRC
957C1992
Location of Patch
10024000
Execution of Patch
1002401D
Version Base
5C843EFA
Version new
5971BBBB
Status
Patch File Loaded to the GUI (Click on Store OTP button to load to OTP)
Select a partition
Click here to Store Patch
Partition Number
1
Store OTP Patch
Store to RAM
Reset
Remaining size in selected partition (After Stroring the patch)-
OK
Apply
Help
Ready

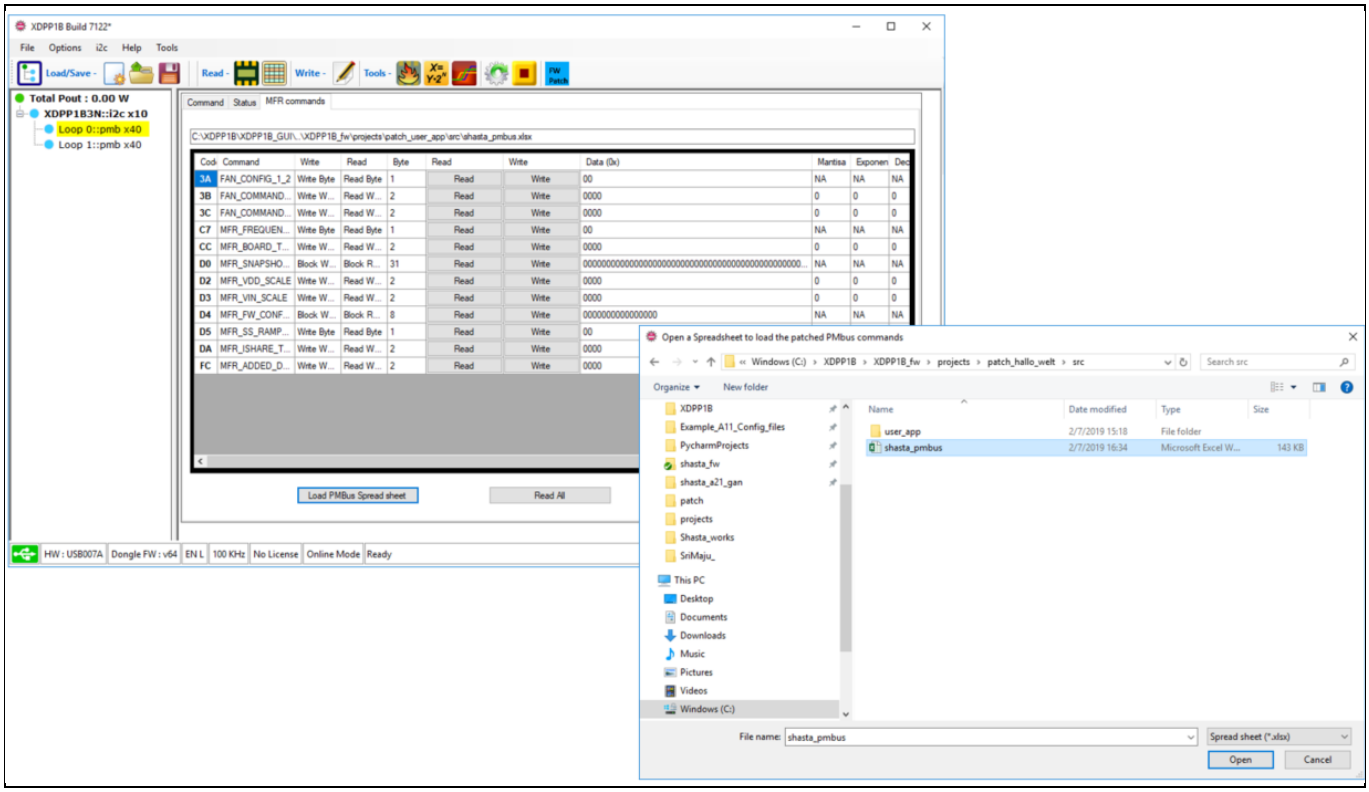
10.4 Test the newly created PMBus MFR command

10.4.1 STEP 1: **Open XDPP1100 GUI and go to “MFR commands” tab**

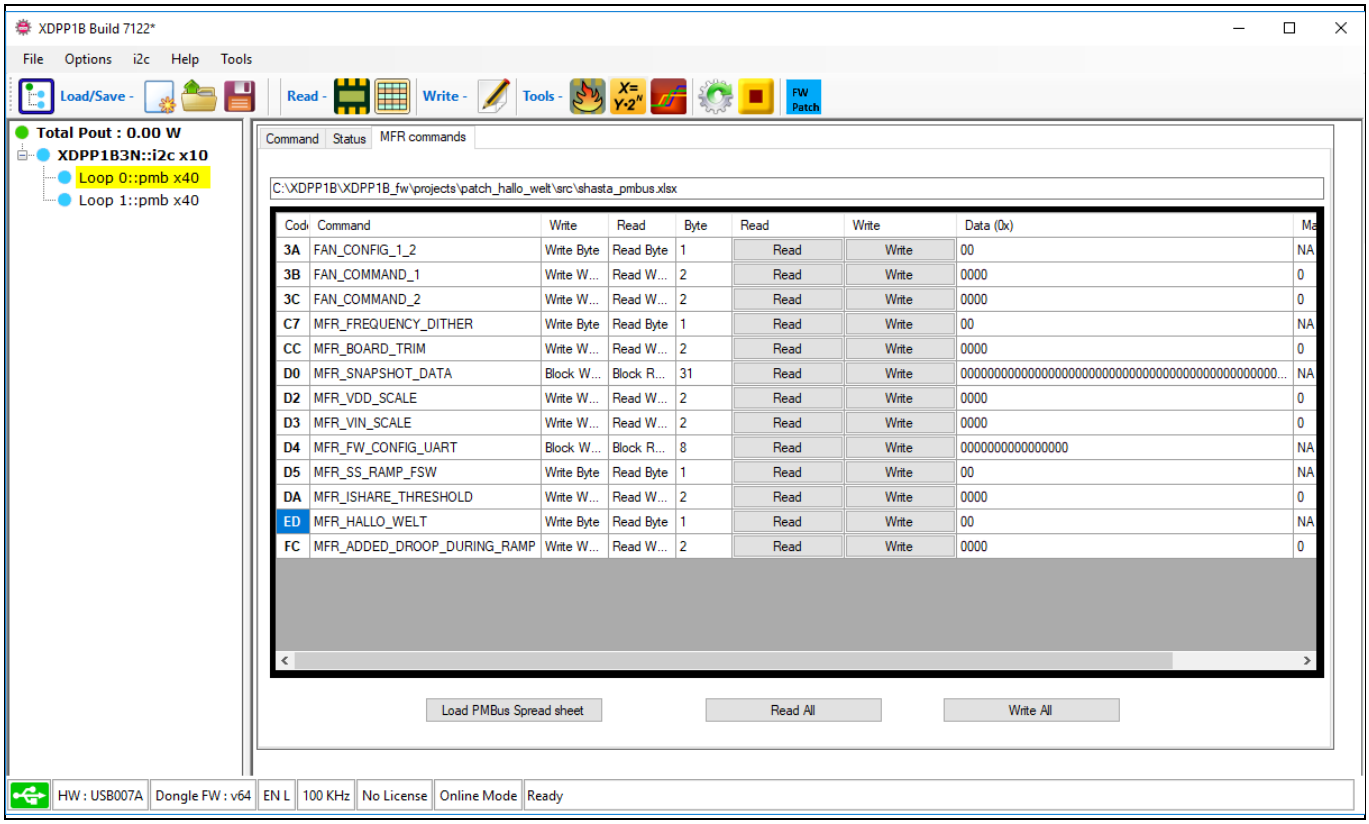


Cod	Command	Write	Read	Byte	Read	Write	Data (0x)	Mantisa	Exponen	Dec
3A	FAN_CONFIG_1_2	Write Byte	Read Byte	1	Read	Write	00	NA	NA	NA
3B	FAN_COMMAND...	Write W...	Read W...	2	Read	Write	0000	0	0	0
3C	FAN_COMMAND...	Write W...	Read W...	2	Read	Write	0000	0	0	0
C7	MFR_FREQUEN...	Write Byte	Read Byte	1	Read	Write	00	NA	NA	NA
CC	MFR_BOARD_T...	Write W...	Read W...	2	Read	Write	0000	0	0	0
D0	MFR_SNAPSHO...	Block W...	Block R...	31	Read	Write	00000000000000000000000000000000000000000000000000...	NA	NA	NA
D2	MFR_VDD_SCALE	Write W...	Read W...	2	Read	Write	0000	0	0	0
D3	MFR_VIN_SCALE	Write W...	Read W...	2	Read	Write	0000	0	0	0
D4	MFR_FW_CONF...	Block W...	Block R...	8	Read	Write	0000000000000000	NA	NA	NA
D5	MFR_SS_RAMP...	Write Byte	Read Byte	1	Read	Write	00	NA	NA	NA
DA	MFR_ISHARE_T...	Write W...	Read W...	2	Read	Write	0000	0	0	0
FC	MFR_ADDED_D...	Write W...	Read W...	2	Read	Write	0000	0	0	0

10.4.2 STEP 2: Click on “Load PMBus Spreadsheet” and select the correct “shasta_pmbus.xlsx” from the project folder



10.4.3 STEP 3: If setup is good, 0xED will appear. Set data to FF and write



Cod	Command	Write	Read	Byte	Read	Write	Data (0x)	Ma
3A	FAN_CONFIG_1_2	Write Byte	Read Byte	1	Read	Write	00	NA
3B	FAN_COMMAND_1	Write W...	Read W...	2	Read	Write	0000	0
3C	FAN_COMMAND_2	Write W...	Read W...	2	Read	Write	0000	0
C7	MFR_FREQUENCY_DITHER	Write Byte	Read Byte	1	Read	Write	00	NA
CC	MFR_BOARD_TRIM	Write W...	Read W...	2	Read	Write	0000	0
D0	MFR_SNAPSHOT_DATA	Block W...	Block R...	31	Read	Write	00000000000000000000000000000000000000000000000000...	NA
D2	MFR_VDD_SCALE	Write W...	Read W...	2	Read	Write	0000	0
D3	MFR_VIN_SCALE	Write W...	Read W...	2	Read	Write	0000	0
D4	MFR_FW_CONFIG_UART	Block W...	Block R...	8	Read	Write	0000000000000000	NA
D5	MFR_SS_RAMP_FSW	Write Byte	Read Byte	1	Read	Write	00	NA
DA	MFR_ISHARE_THRESHOLD	Write W...	Read W...	2	Read	Write	0000	0
ED	MFR_HALLO_WELT	Write Byte	Read Byte	1	Read	Write	00	NA
FC	MFR_ADDED_DROOP_DURING_RAMP	Write W...	Read W...	2	Read	Write	0000	0

10.5 Quick Guide on PMBus Spreadsheet

As shown in the previous section, adding PMBus MFR commands in XDPP1100 is done by modifying the PMBus spreadsheet given in each firmware patch project using Microsoft Excel. Only some columns that need to be modified. The following table describes the columns in the spreadsheet.

Table 4 PMBus Spreadsheet

Column Name	Description
Opcode	[DO NOT MODIFY] PMBus command code. This is fixed, ranging from 0 to 255.
Command	Put prefix “MFR_” when keying in the new PMBus MFR command name.
Wr. Tx	Put “Write” to allow GUI to write. Leave blank if this command is just for reading (e.g. Telemetry-kind of functionality). Determine the data size: Byte, Word, Block.
Rd. Tx	Put “Read” to allow GUI to read. Usually paired with “Write”. Determine the data size: Byte, Word, Block.
#B	PMBus data size, expressed in a unit of Byte. Therefore, If Byte: 1 If Word: 2 (equivalent to 2 Byte) If Block: n-number of Bytes.
MFR	To indicate that this command is an MFR command. Put ‘y’
Loop 0 Support	To indicate this PMBus command acts on firmware features or hardware modules that belong to Loop 0 (usually control loop-related peripherals such as PID, TELEM, FAULT, VCONTROL). Set to ‘y’ to enable Loop 0 support. Set to ‘n’ to disable Loop 0 support.
Loop 0 range/res	To indicate the range/resolution of the PMBus MFR commands. Fill in the Q-number format if it is telemetry type of MFR command Leave blank if it is register/configuration/setting type of MFR command
Loop 0 reset	Reset value. Put 0 as default. If byte: 0x00 If word: 0x0000 If block: n-number of 0x00 where n is the block size in Bytes.
Loop 0 OTP nbits	Number of Bits (not to be confused with Bytes) of the data to be stored to the OTP. In general it should be 8 times the #B column. If #B column is 1: 8 If #B column is 2: 16 If #B column is 6: 48
Loop 0 OTP store	To indicate the bit numbers. If “Loop 0 OTP nbits” is 8: [7:0] If “Loop 0 OTP nbits” is 16: [15:0] If “Loop 0 OTP nbits” is 48: [47:0]
Loop 1 Support	

Column Name	Description
Loop 1 range/res	Same description as per Loop 0 shown above. Use only if there is a need to support Loop 1.
Loop 1 reset	
Loop 1 OTP nbits	
Loop 1 OTP store	
Reg store	[DO NOT MODIFY] Not used. Leave it as it is.
Reg nbits	
Cmd nBytes	
Notes	
HAS FW HANDLE	To indicate whether the PMBus has firmware handler (callback function). Put 'y' to indicate there is a need for firmware handler. A function declaration with default name will be generated automatically. Put 'n' if there is no need for firmware handler. *It is also possible to put in the function name immediately but ensure there is no duplicate naming with any other commands.* *Users are encouraged to stick to 'y' with default generated function name to minimize error.*
Coverage Loop 0	[DO NOT MODIFY]
Coverage Loop 1	
Common Storage	Used when this PMBus command acts on firmware features or on hardware modules that do not belong to either Loop 0 or Loop 1, e.g. PWM, COMMON, ANALOG). Setting 'y' to this column will force the Loop 1 generated codes to point to Loop 0.
Description	To describe the PMBus MFR command for documentation purpose.

10.5.1 Quick Guidelines

In general, the following guidelines must be observed when implementing new MFR commands:

- To use 0xE5 (MFR_SPECIFIC_E5), 0xE8 (MFR_SPECIFIC_E8) and 0xED (MFR_SPECIFIC_ED) if there are only 3 new MFR commands needed.
- Use 0xC0 (MFR_MAX_TEMP_1), 0xC1 (MFR_MAX_TEMP_2), 0xC2 (MFR_MAX_TEMP_3) can be used if the above have been used.
- Any other commands marked as MFR can be replaced but proceed with caution to ensure there is no critical functions being replaced.
- Do not replace any "RESERVED" Commands

10.5.2 Excel Language Setting

Please note that for full compatibility, the Microsoft Excel language has to be set as “English (United States)” by clicking File > Options > Language. Select “English (United States)” and click on “Set as Default”.

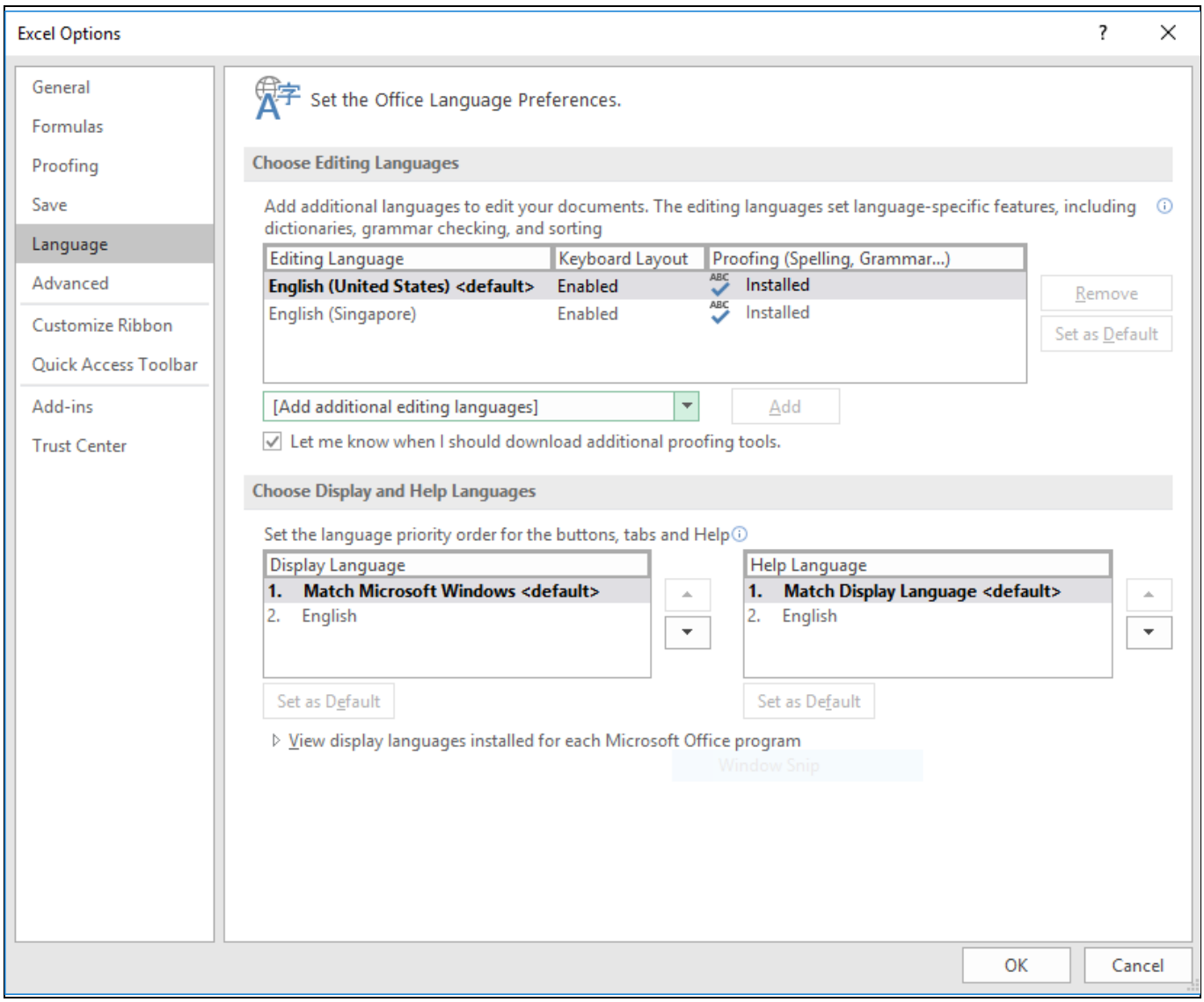


Figure 6 Excel Language Options

11 More Resources

More programming resources can be accessed easily from GUI main interface:

11.1 Firmware Documentation

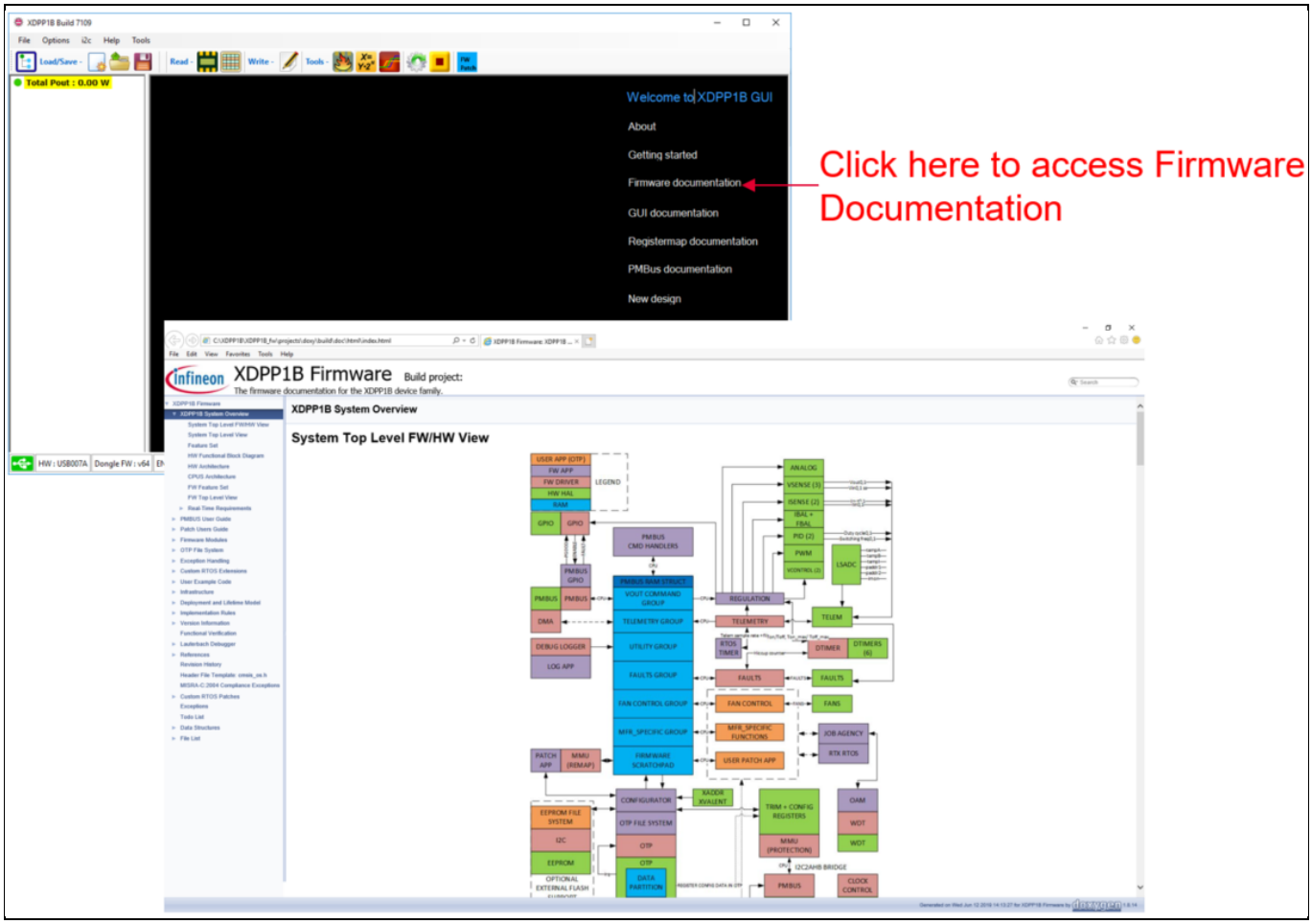


Figure 7 Firmware Documentation

11.2 Register Map Documentation

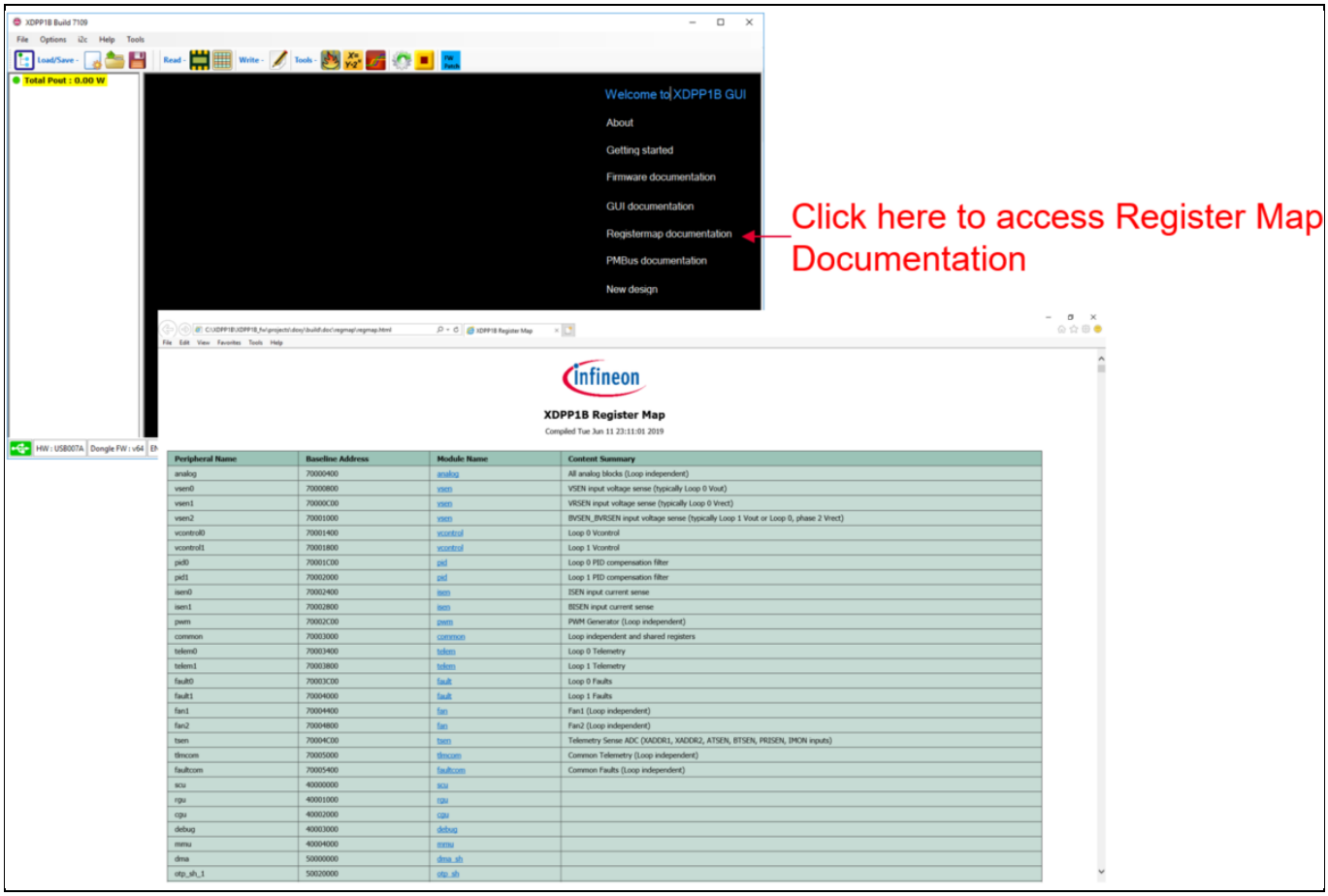


Figure 8 Register Map Documentation

11.3 PMBus Documentation

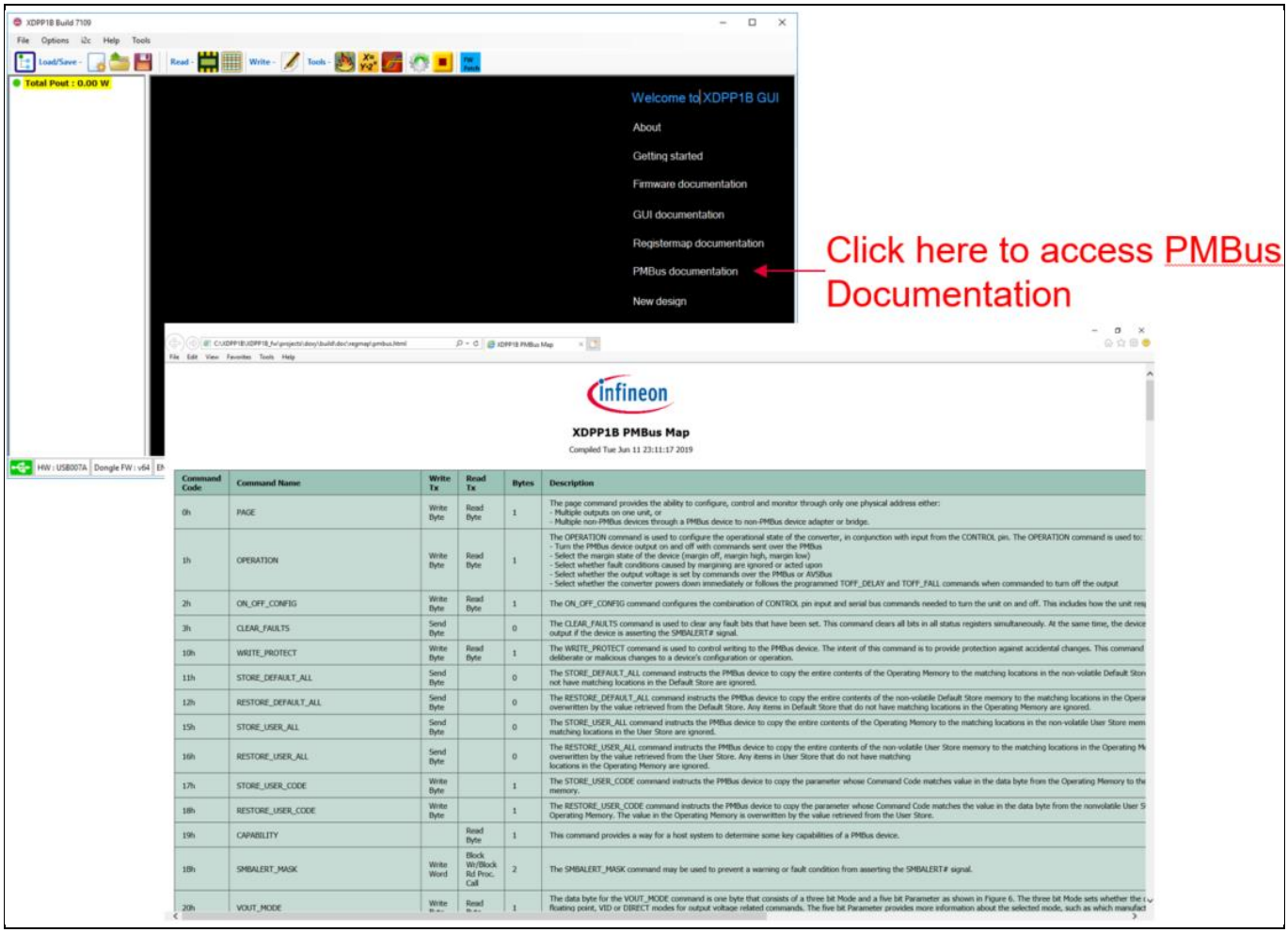


Figure 9 PMBus Documentation

12 References

http://www.pmbus.org/Home

http://www.superkits.net/whitepapers/Fixed%20Point%20Representation%20&%20Fractional%20Math.pdf

https://www2.keil.com/mdk5/cmsis/rtx

http://eguruchela.com/math/calculator/twos-complement

Revision history

Document version	Date of release	Description of changes

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2020-01-13

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Email: erratum@infineon.com

Document reference
XDPP1100 FW DEV

IMPORTANT NOTICE

The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology delivery terms and conditions and prices please contact your nearest Infineon Technologies office (www.infineon.com).

WARNINGS

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.