

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

About this document	1
Table of contents	1
1 Introduction	3
2 Firmware Architecture	4
2.1 Block Diagram	4
2.2 Custom Firmware Use Cases	5
2.3 Simplified Startup sequences	5
2.4 Regulation State Machines	6
3 Setting Up XDPP1100 Development Environment	8
3.1 STEP 1: Download XDPP1100 Installation Package	8
3.2 STEP 2: Download Softwares and Development Tools	9
3.3 STEP 3: Setup Eclipse User Environment Paths	9
3.4 STEP 4: Start Eclipse	10
4 Creating a New Project	11
4.1 STEP 1: Duplicate patch_user_app project	11
4.2 STEP 2: Rename the folder to your project	11
4.3 STEP 3: Edit “.project” file in the project folder	12
4.4 STEP 4: Go back to Eclipse and import the new project	13
4.5 STEP 5: Right-click and select “Properties”	14
4.6 STEP 6: Set Build Configuration to “Windows”	15
4.7 STEP 7: Expand the new project and Compile	16
5 Implementing “Hello World” Project	17
5.1 STEP 1: Expand the project and go to \src\user_app\user_app.c	17
5.2 STEP 2: Create and Add “hello_world()” function into “user_drv_init()”	18
5.3 STEP 3: Build- Clean to remove /build folder	21
5.4 STEP 4: Compile and Build Patch (ARM_GCC)	22
6 Downloading Patch to OTP	23
6.1 STEP 1: Open XDPP1B GUI	23
6.2 STEP 2: Connect PMBus Dongle and XDPP1B	24
6.3 STEP 3: Add Device	26
6.4 STEP 4: Force I2C/PMBus OK	27

6.5	STEP 5: Access FW Patch Tool.....	28
6.6	STEP 6: Store Trim.....	29
6.7	STEP 7: Load Patch File.....	30
6.8	STEP 8: Store Patch File to OTP	31
6.9	STEP 9: Check for Active Patch	32
6.10	STEP 10: Patch is loaded. Restart XDPP1100.....	33
7	Monitor UART Traffic	34
7.1	STEP 1: Connect XDPP1100 with USB-UART dongle.....	34
7.2	STEP 2: Monitor UART traffic.....	34
7.3	STEP 3: “Hello World” will appear	35
8	Invalidate Patch from OTP.....	36
8.1	STEP 1: Ensure there is an active Patch in the OTP.....	36
8.2	STEP 2: Invalidate the Patch in OTP	37
8.3	STEP 3: Verify that Patch has been invalidated.....	38
8.3.1	Notes.....	38
9	Running Patch from RAM.....	39
10	Implementing Custom PMBus Command in XDPP1100	42
10.1	Adding new PMBus MFR command in the spreadsheet	42
10.1.1	STEP 1: Open PMBus Spreadsheet “shasta_pmbus.xlsx”	42
10.1.2	STEP 2: Select the empty MFR command slot, e.g. 0xED.....	43
10.1.3	STEP 3: Modify the PMBus command accordingly. Save the file.	44
10.2	Generate PMBus codes in the project.....	45
10.2.1	STEP 1: Click on “build/pmbus” to generate PMBus C code	45
10.2.2	STEP 2: Open “pmbus_mfr_autogen.c/h” for generated codes	46
10.2.3	STEP 3: Scroll down at “pmbus_mfr_autogen.c”	47
10.3	Implement PMBus MFR command handler.....	48
10.3.1	STEP 1: Note the PMBus Command Handler name	48
10.3.2	STEP 2: Modify “pmbus_mfr_specific_handler.c”	49
10.3.3	STEP 3: Put in a custom function above the handler.....	50
10.3.4	STEP 4: Call your function in the PMBus MFR handler.....	51
10.3.5	STEP 5: Clean and Build project.	52
10.3.6	STEP 6: Download the Patch	53
10.4	Test the newly created PMBus MFR command	56
10.4.1	STEP 1: Open XDPP1100 GUI and go to “MFR commands” tab	56
10.4.2	STEP 2: Click on “Load PMBus Spreadsheet” and select the correct “shasta_pmbus.xlsx” from the project folder.....	57
10.4.3	STEP 3: If setup is good, 0xED will appear. Set data to FF and write	58
10.5	Quick Guide on PMBus Spreadsheet	59
10.5.1	Quick Guidelines	60
10.5.2	Excel Language Setting.....	61
11	More Resources.....	62
11.1	Firmware Documentation.....	62
11.2	Register Map Documentation	63
11.3	PMBus Documentation	64
12	References	65
	Revision history.....	66

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.

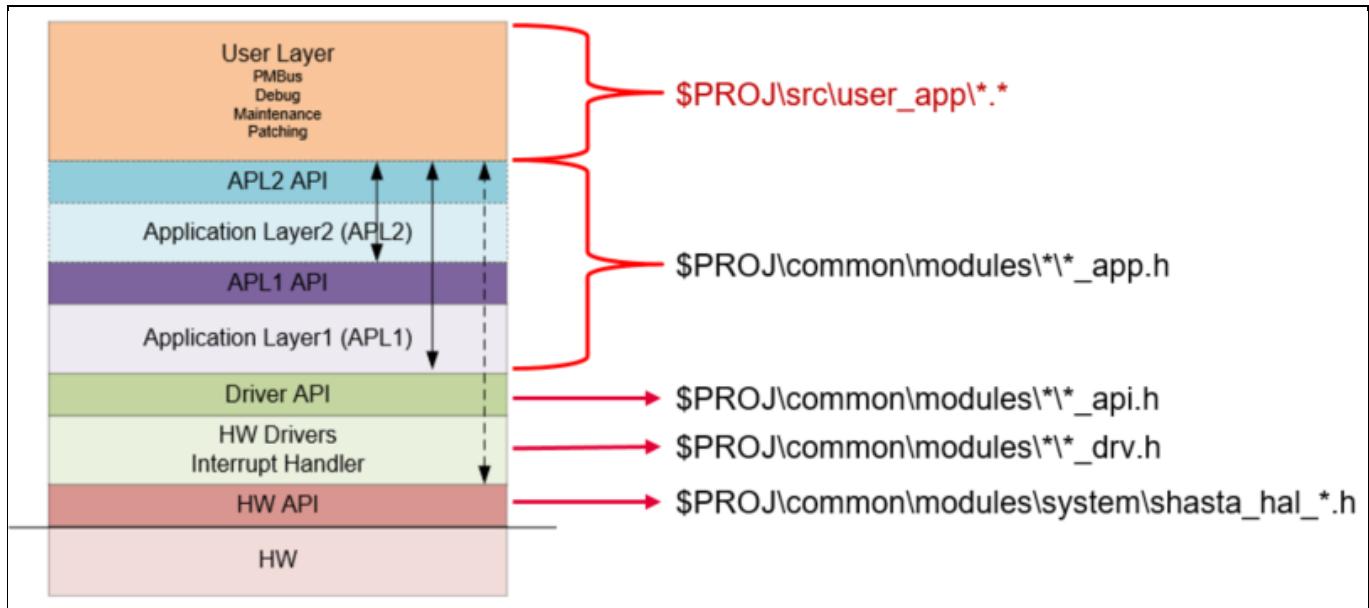


Figure 1 XDPP1100 Firmware Architecture Block Diagram

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()
Peripheral initialization	Add_on_features()
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 analogous 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.

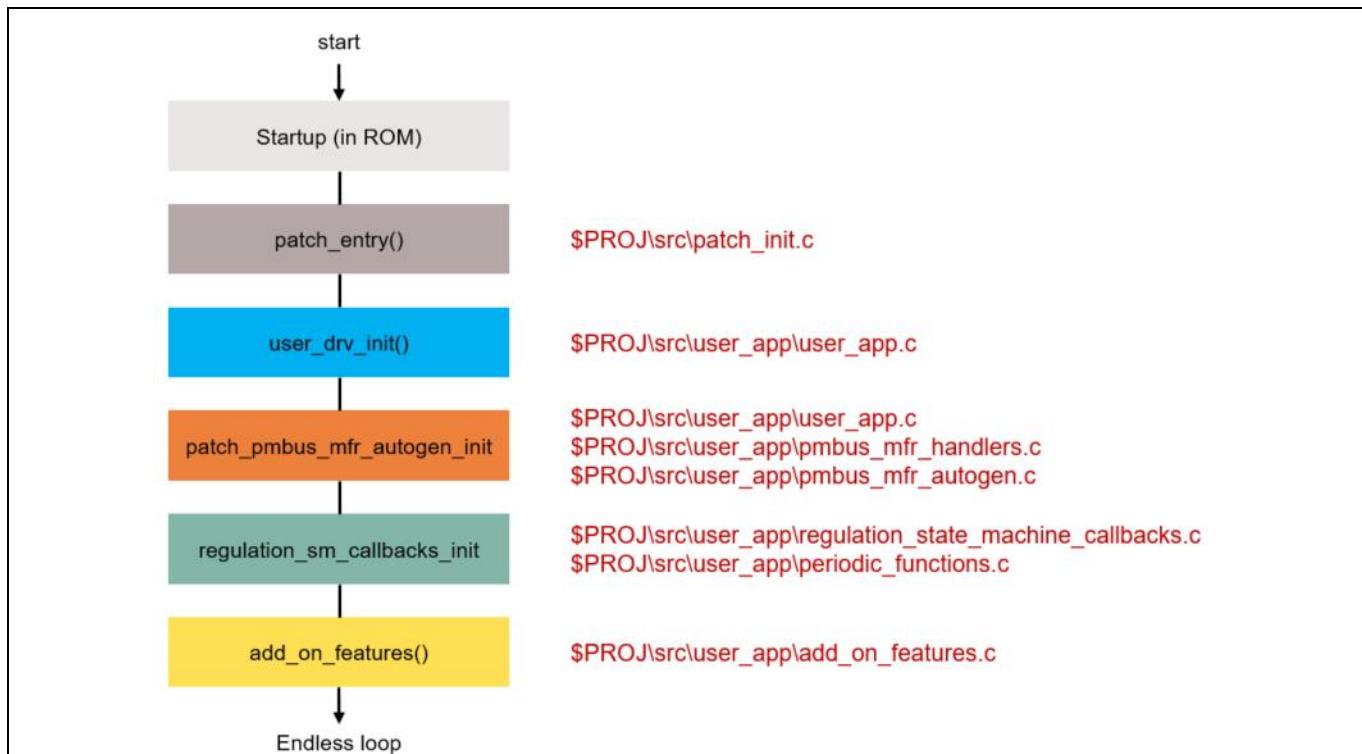


Figure 2 Simplified XDPP1100 Startup Sequence

2.4 Regulation State Machines

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

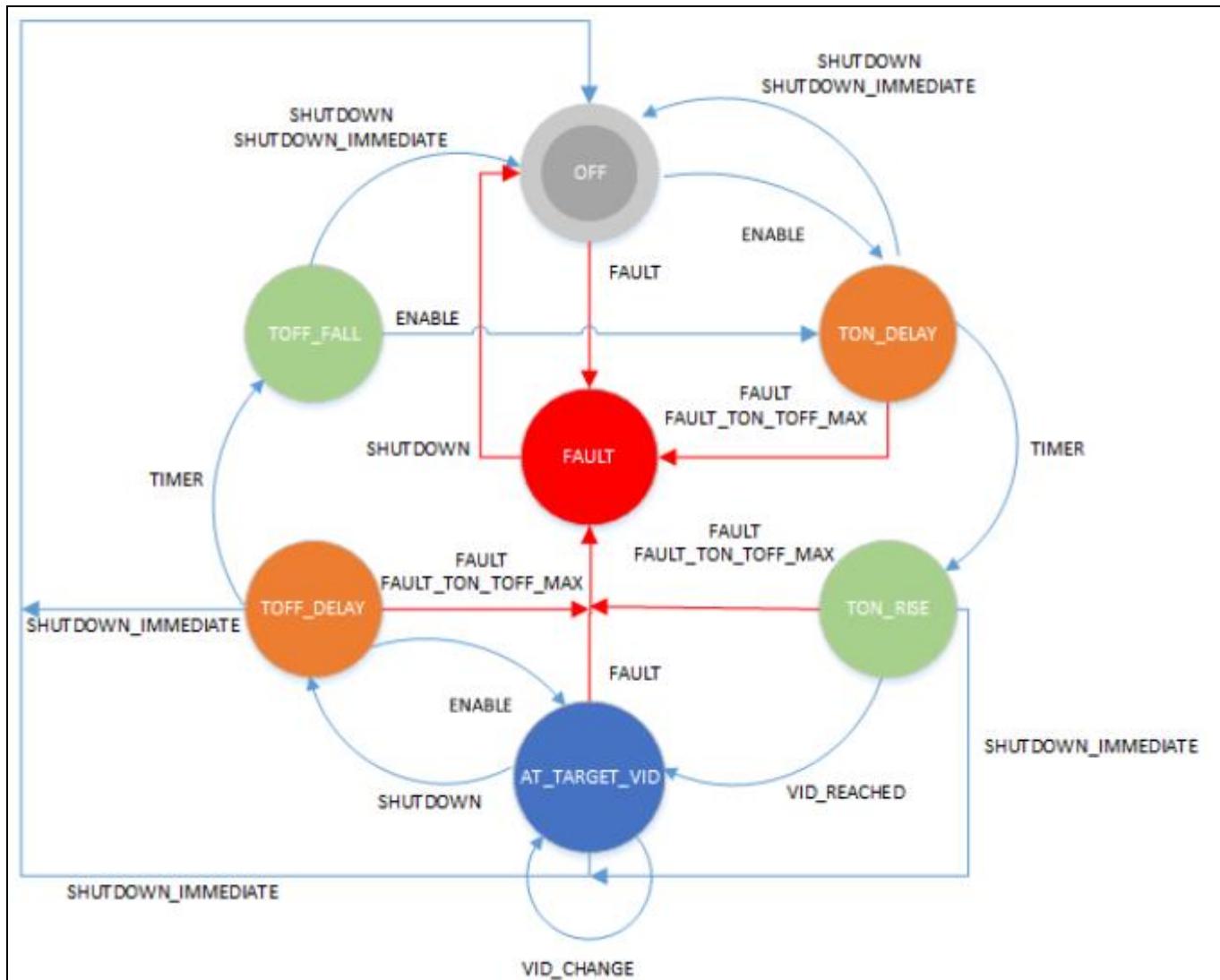
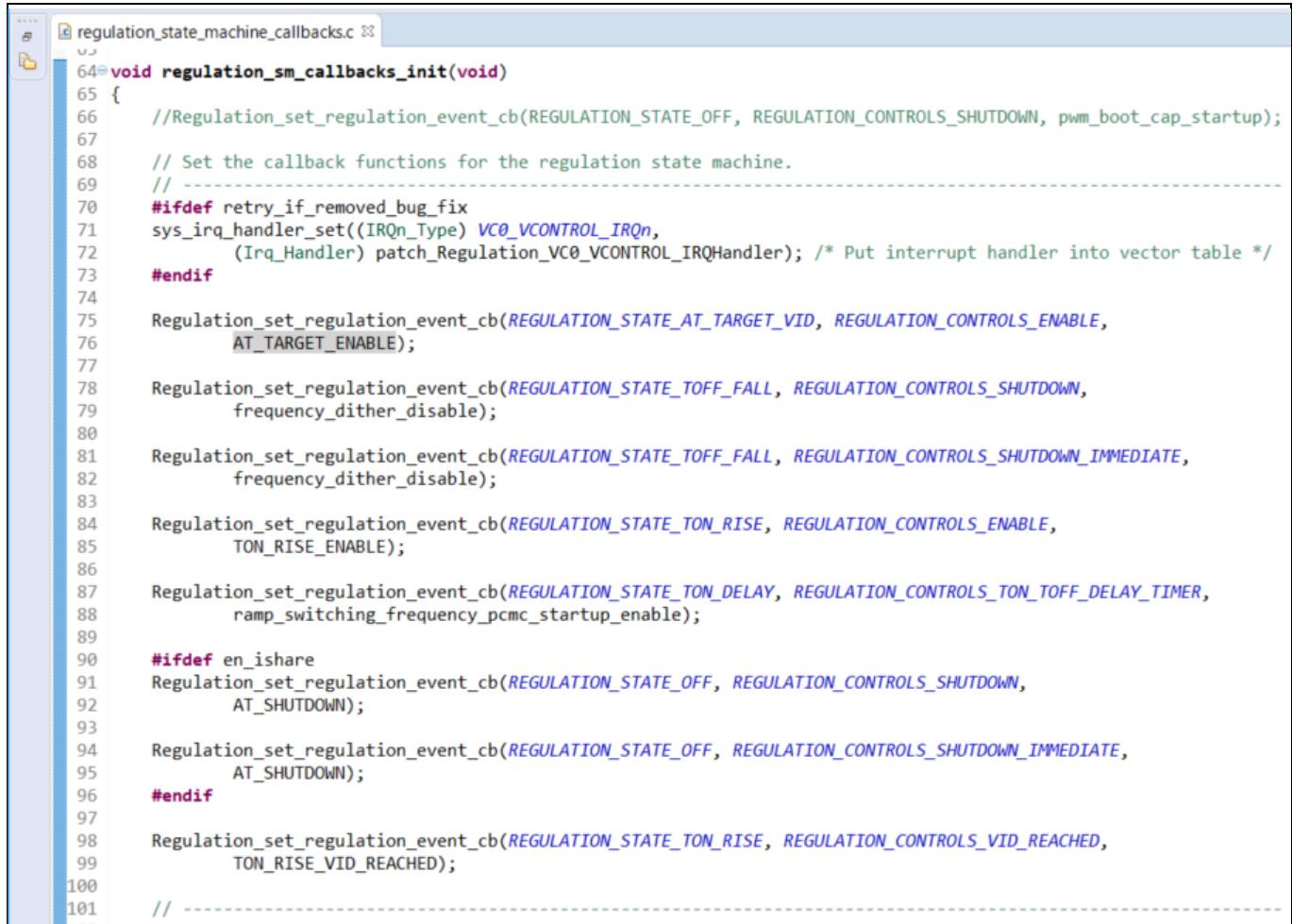


Figure 3 Regulation State Machines Diagram

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) VCO_VCONTROL_IRQn,
72         (Irq_Handler) patch_Regulation_VCO_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

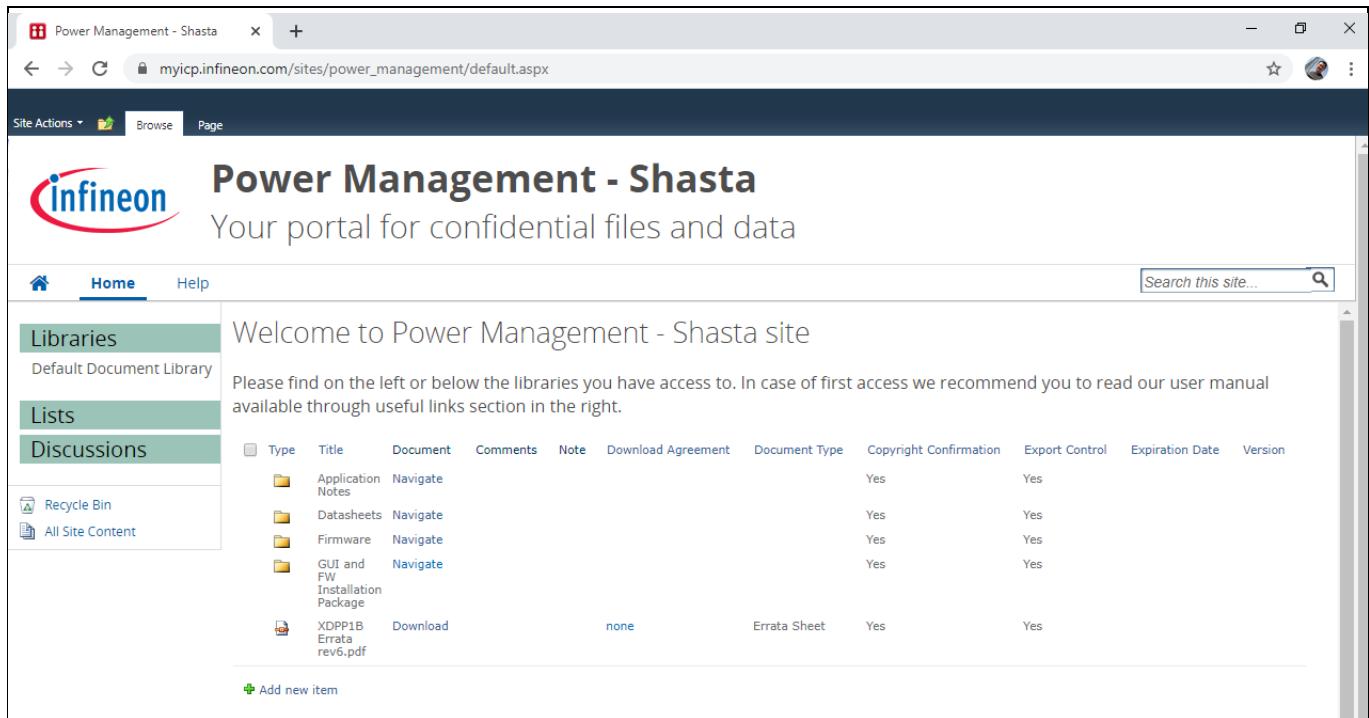
This section describes the XDPP1100 development environment setup and required tools.

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:

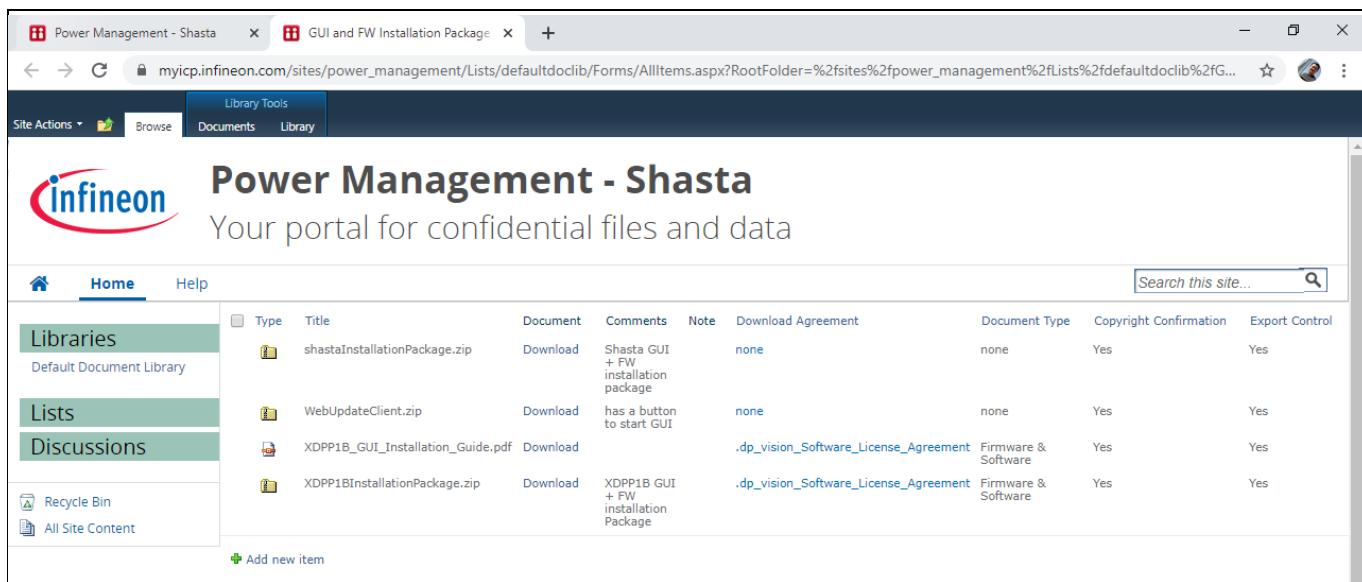


The screenshot shows a SharePoint-based website titled "Power Management - Shasta". The left sidebar contains navigation links for "Home", "Help", "Libraries", "Lists", "Discussions", "Recycle Bin", and "All Site Content". The main content area displays a welcome message and a table listing files in the "GUI and FW Installation Package" library. The table columns are: Type, Title, Document, Comments, Note, Download Agreement, Document Type, Copyright Confirmation, Export Control, Expiration Date, and Version. The listed items are:

Type	Title	Document	Comments	Note	Download Agreement	Document Type	Copyright Confirmation	Export Control	Expiration Date	Version
Application Notes	Navigate						Yes	Yes		
Datasheets	Navigate						Yes	Yes		
Firmware	Navigate						Yes	Yes		
GUI and FW Installation Package	Navigate						Yes	Yes		
XDPP1B Errata rev6.pdf	Download		none			Errata Sheet	Yes	Yes		

Figure 5 MyICP screenshot

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



The screenshot shows a SharePoint library interface titled "Power Management - Shasta". The left navigation bar includes "Home", "Help", "Libraries" (selected), "Lists", "Discussions", "Recycle Bin", and "All Site Content". The main content area displays a table of files:

Type	Title	Document	Comments	Note	Download Agreement	Document Type	Copyright Confirmation	Export Control
shastaInstallationPackage.zip	Download	Shasta GUI + FW installation package	none	none	none	Yes	Yes	
WebUpdateClient.zip	Download	has a button to start GUI	none	none	none	Yes	Yes	
XDPP1B_GUI_Installation_Guide.pdf	Download	.dp_vision_Software_License_Agreement	Firmware & Software	.dp_vision_Software_License_Agreement	Firmware & Software	Yes	Yes	
XDPP1BInstallationPackage.zip	Download	XDPP1B GUI + FW installation Package	.dp_vision_Software_License_Agreement	Firmware & Software	Firmware & Software	Yes	Yes	

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

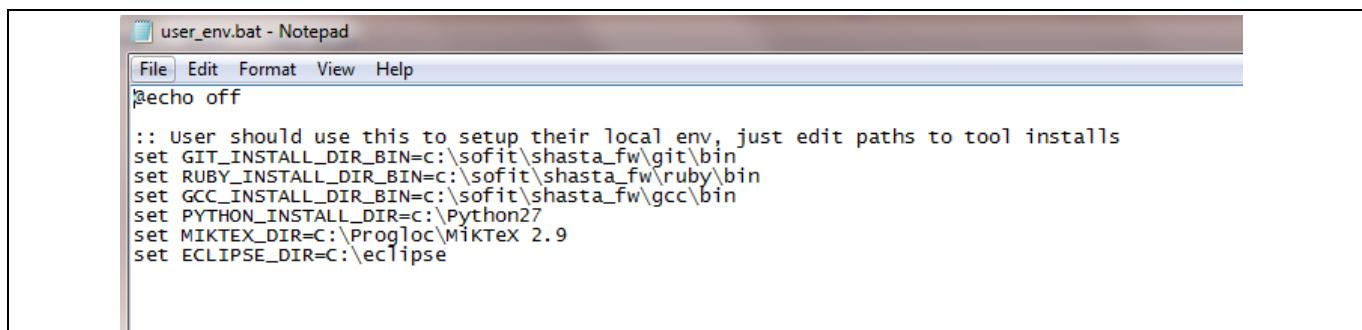
[OPTIONAL] The following tools are optional and not compulsory to get started in firmware development.

Table 3 [OPTIONAL] Required Software and Tools for XDPP1100 firmware 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

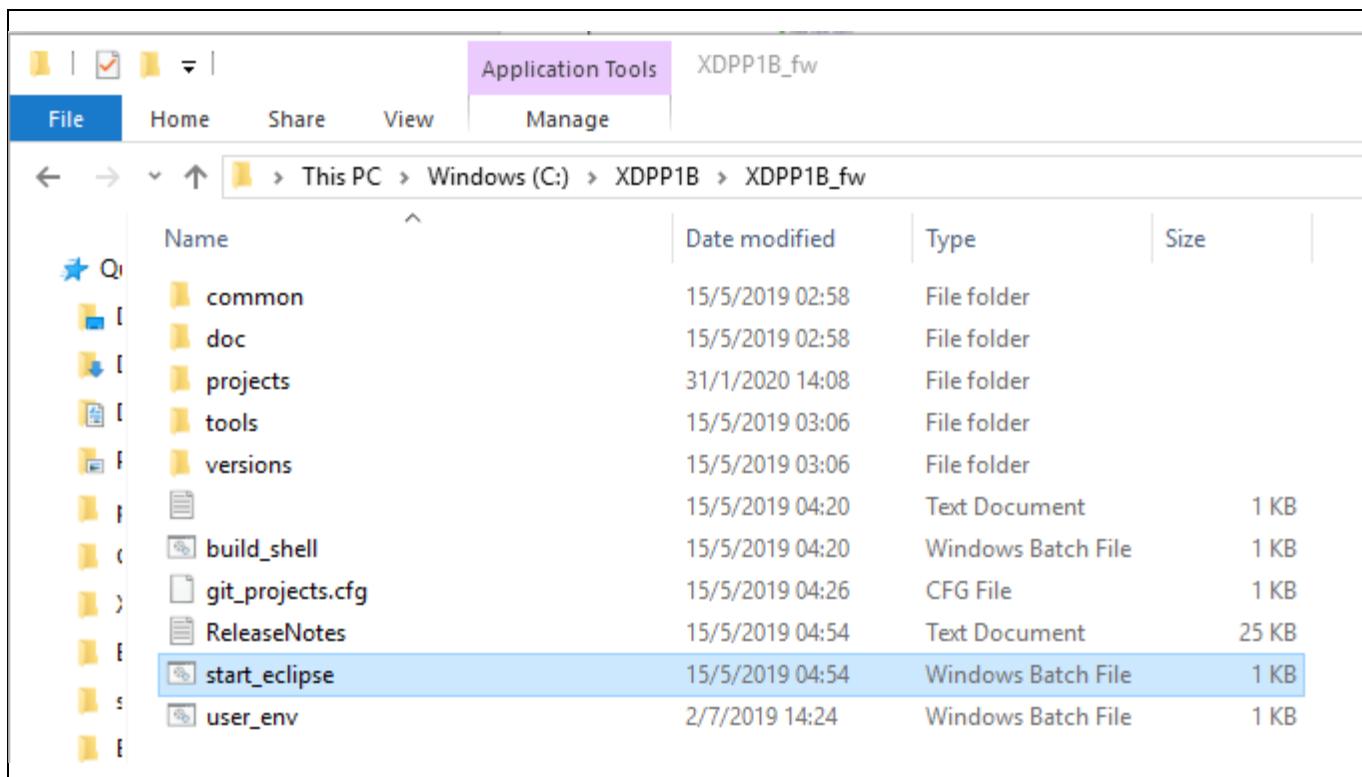
Go to C:\XDPP1B\XDPP1B_fw\user_env.bat and make sure the paths are shown like the following figure:



```
user_env.bat - Notepad
File Edit Format View Help
@echo off
:: User should use this to setup their local env, just edit paths to tool installs
set GIT_INSTALL_DIR_BIN=c:\sofit\shasta_fw\git\bin
set RUBY_INSTALL_DIR_BIN=c:\sofit\shasta_fw\ruby\bin
set GCC_INSTALL_DIR_BIN=c:\sofit\shasta_fw\gcc\bin
set PYTHON_INSTALL_DIR=c:\Python27
set MIKTEX_DIR=C:\Proglc\MiKTeX 2.9
set ECLIPSE_DIR=C:\eclipse
```

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

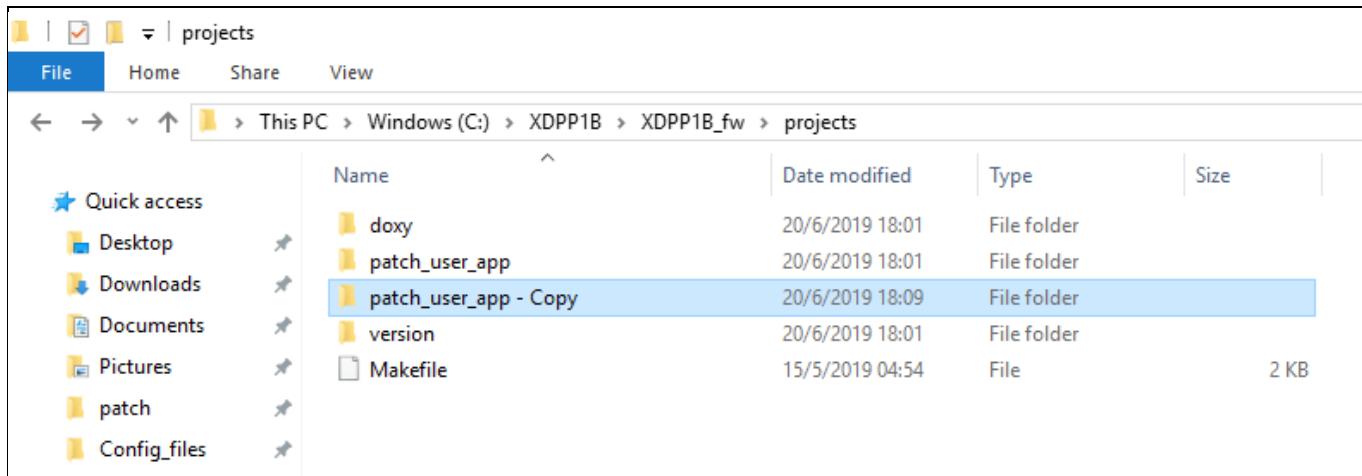
The default firmware of XDPP1100 is hardcoded in ROM. Unlike traditional Flash-based microcontroller, it is not possible to completely erase and reprogram the XDPP1100. For this reason, XDPP1100 comes with OTP “One Time Programmable” memory which allows certain functionalities in XDPP100 to be modified,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

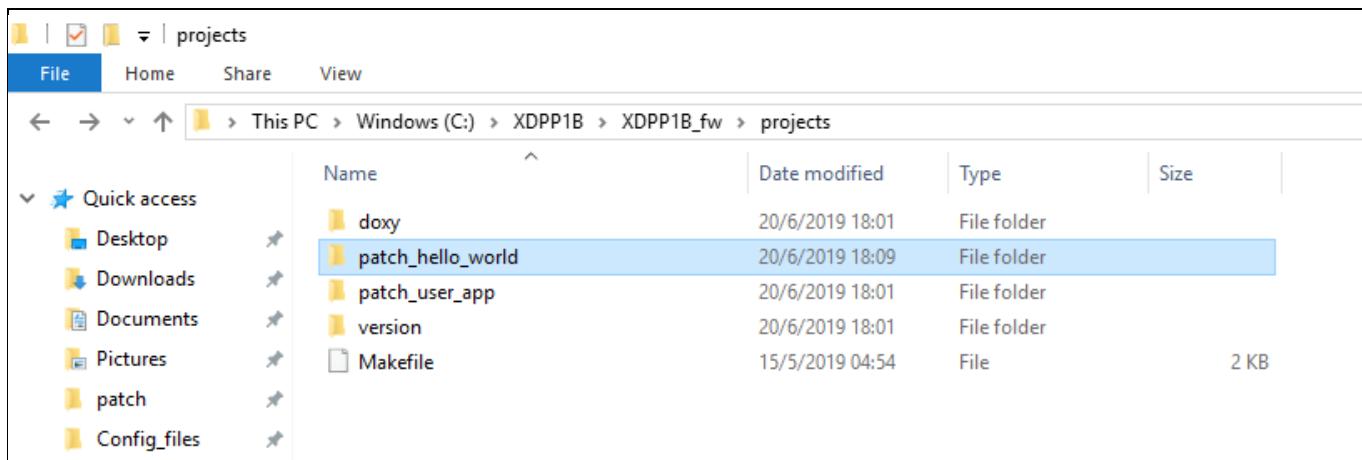
Duplicate the “patch_user_app” folder. Select patch_user_app folder and then use the keyboard shortcut “Ctrl + C” followed by “Ctrl + V”.

You should see “patch_user_app – Copy” folder generated.



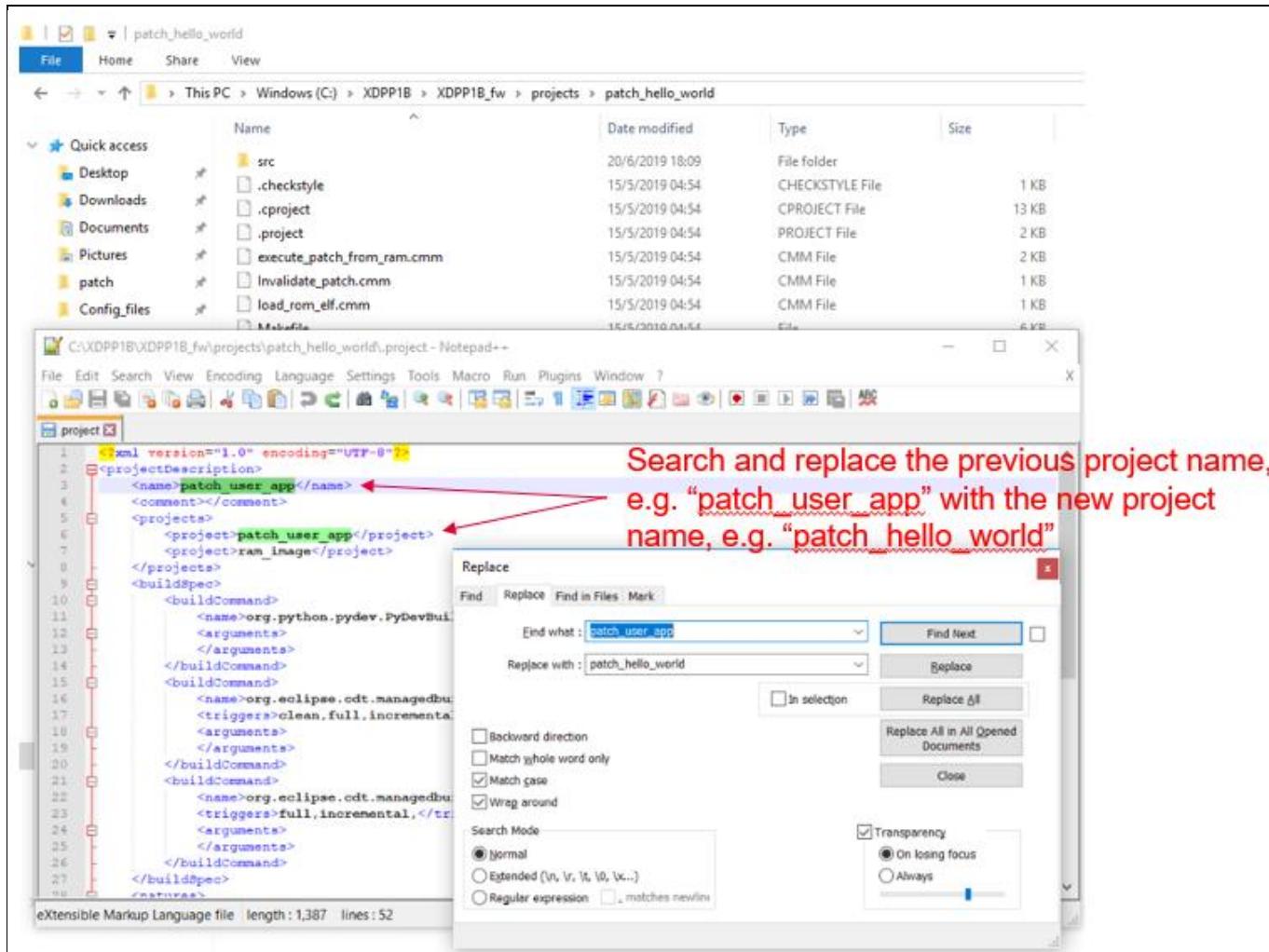
4.2 STEP 2: Rename the folder to your project

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

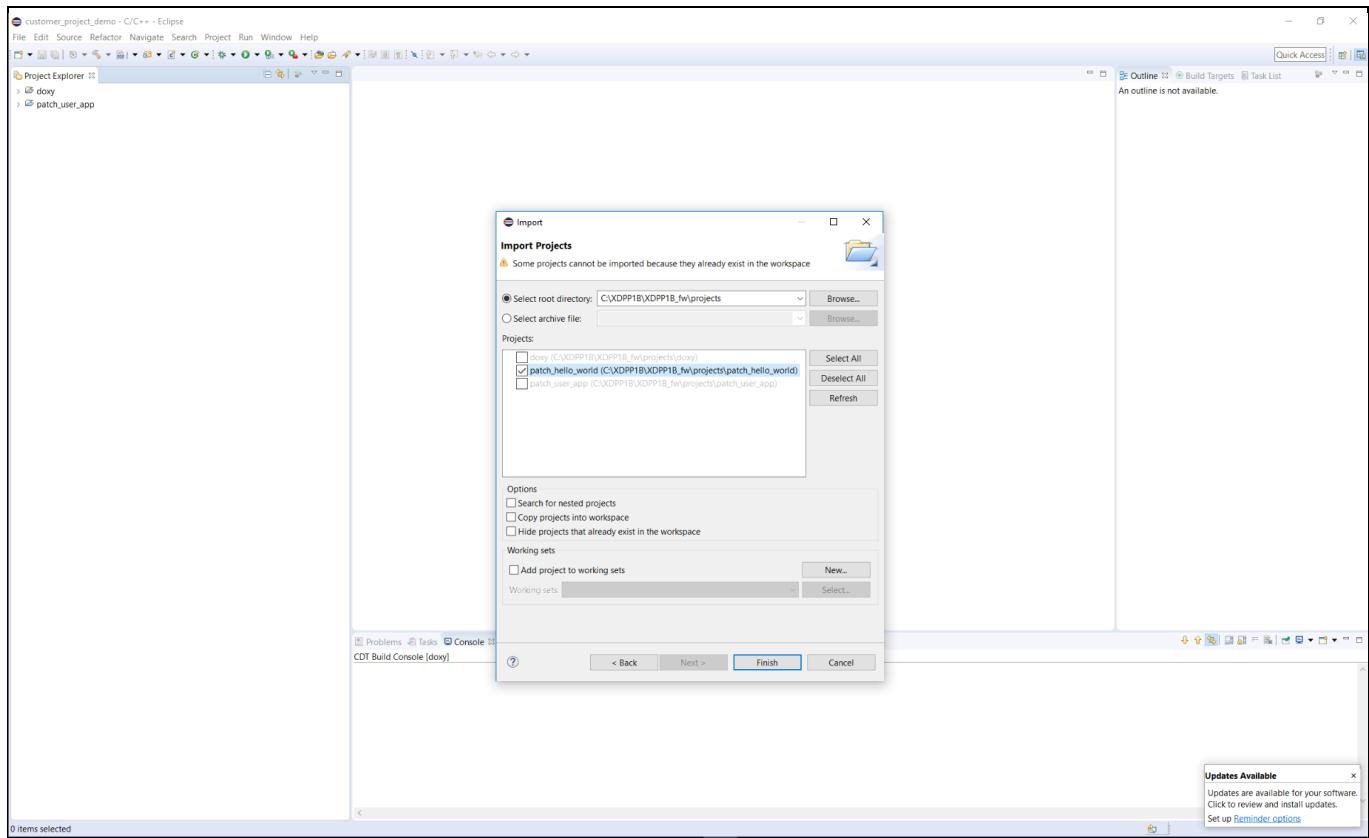


4.3 STEP 3: Edit “.project” file in the project folder

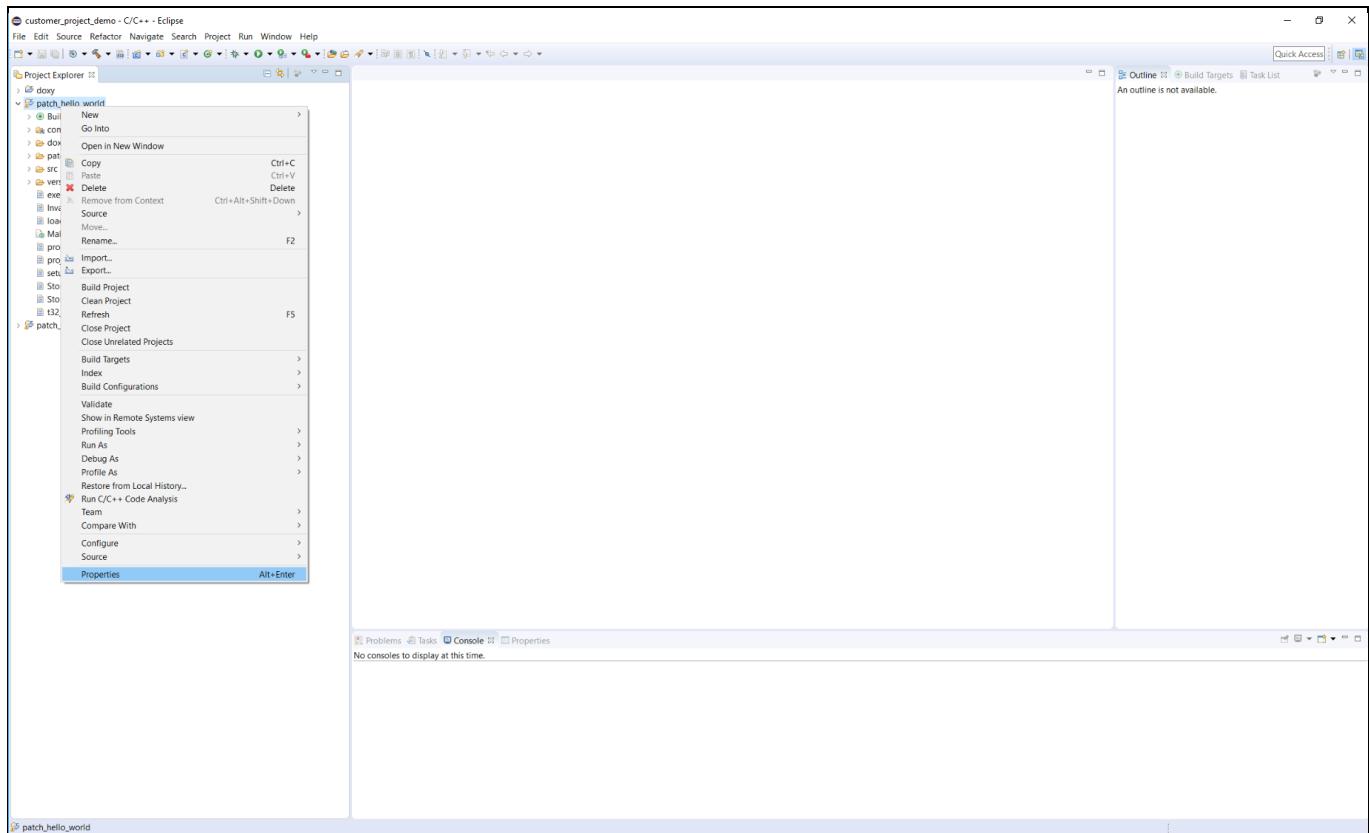
Navigate in to the patch_hello_world folder, look for a file called “.project”. Sometimes it could be hidden due to Windows’ setting. Use a text editor software to open this file, search and replace the previous project name “patch_user_app” with the new project name “patch_hello_world”.



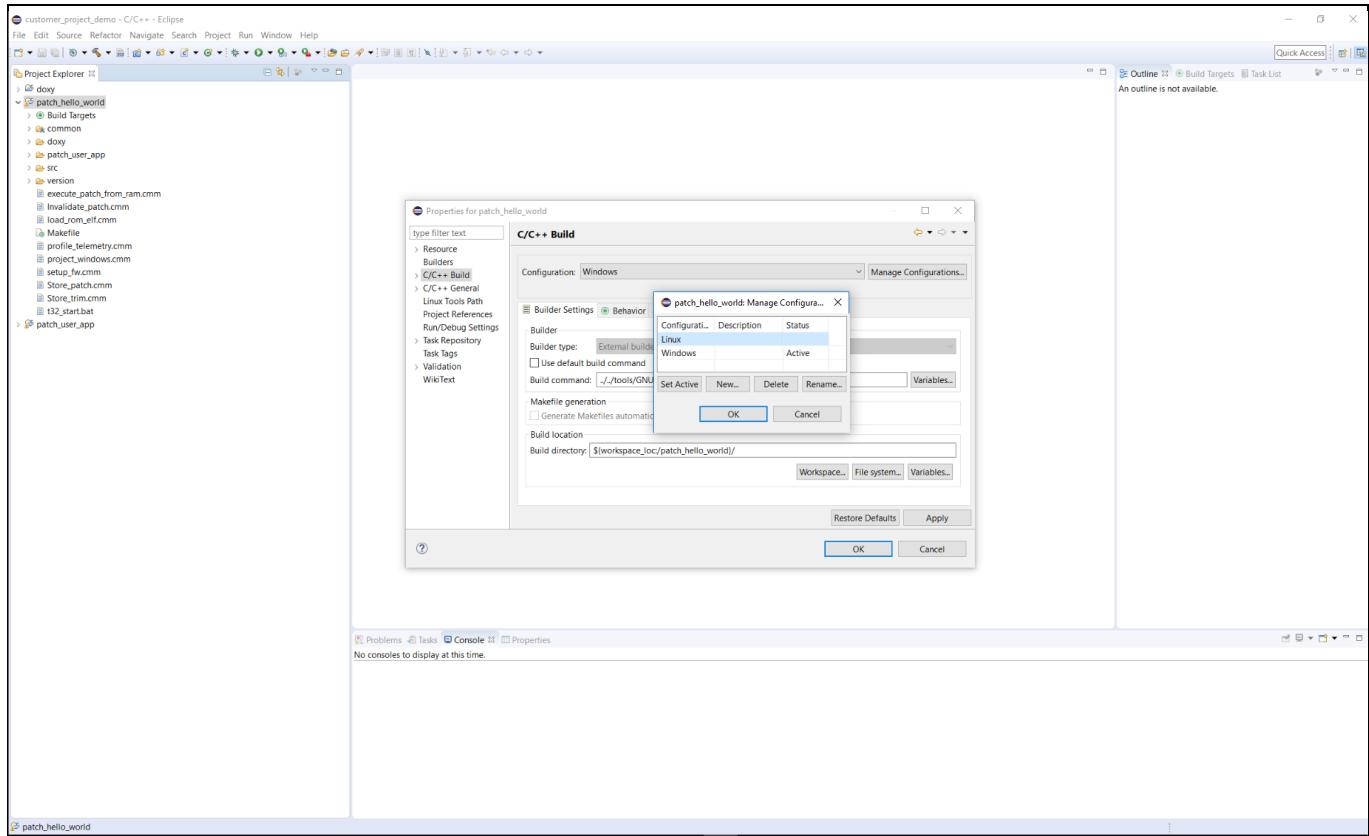
4.4 STEP 4: Go back to Eclipse and import the new project



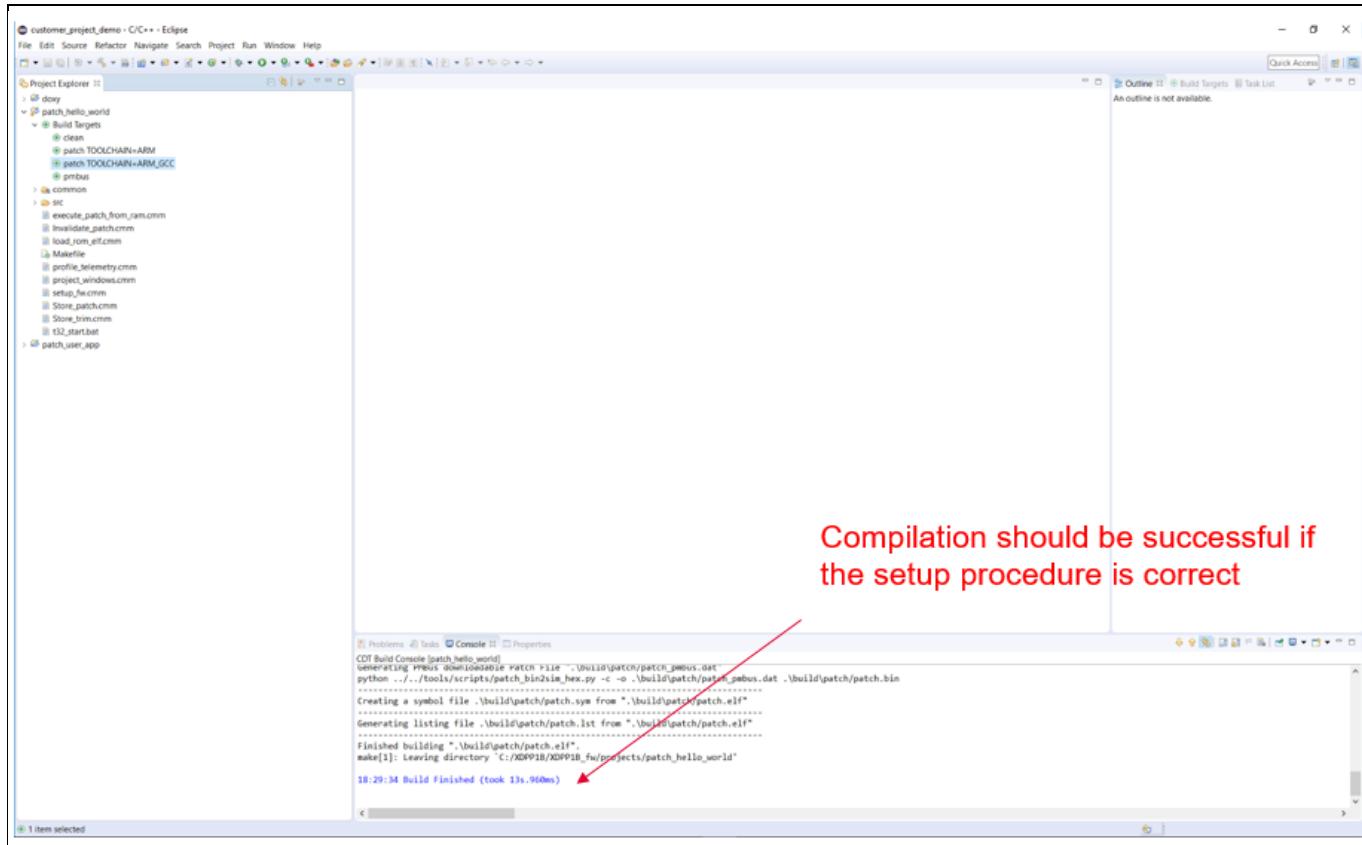
4.5 STEP 5: Right-click and select “Properties”



4.6 STEP 6: Set Build Configuration to “Windows”



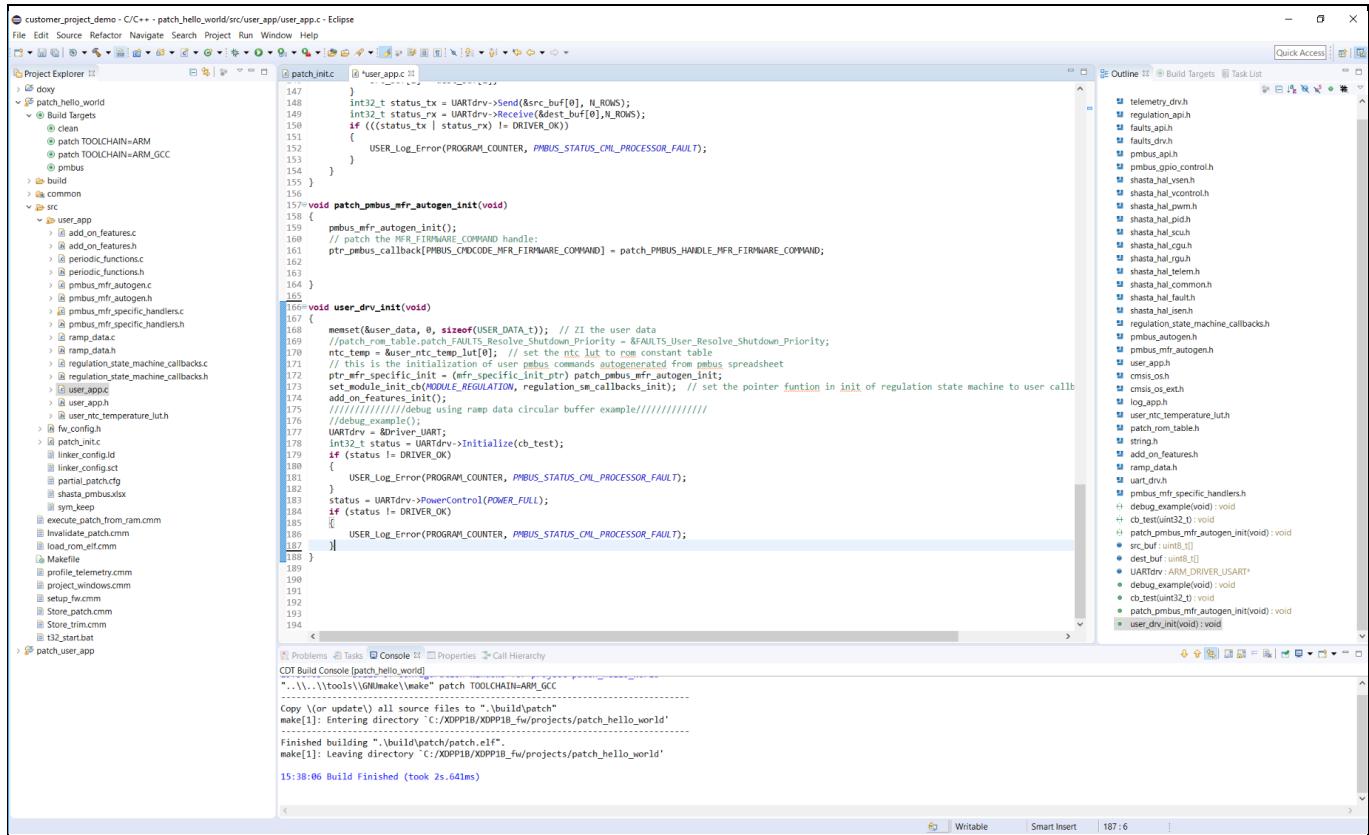
4.7 STEP 7: Expand the new project and Compile



Implementing “Hello World” Project

5 Implementing “Hello World” Project

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

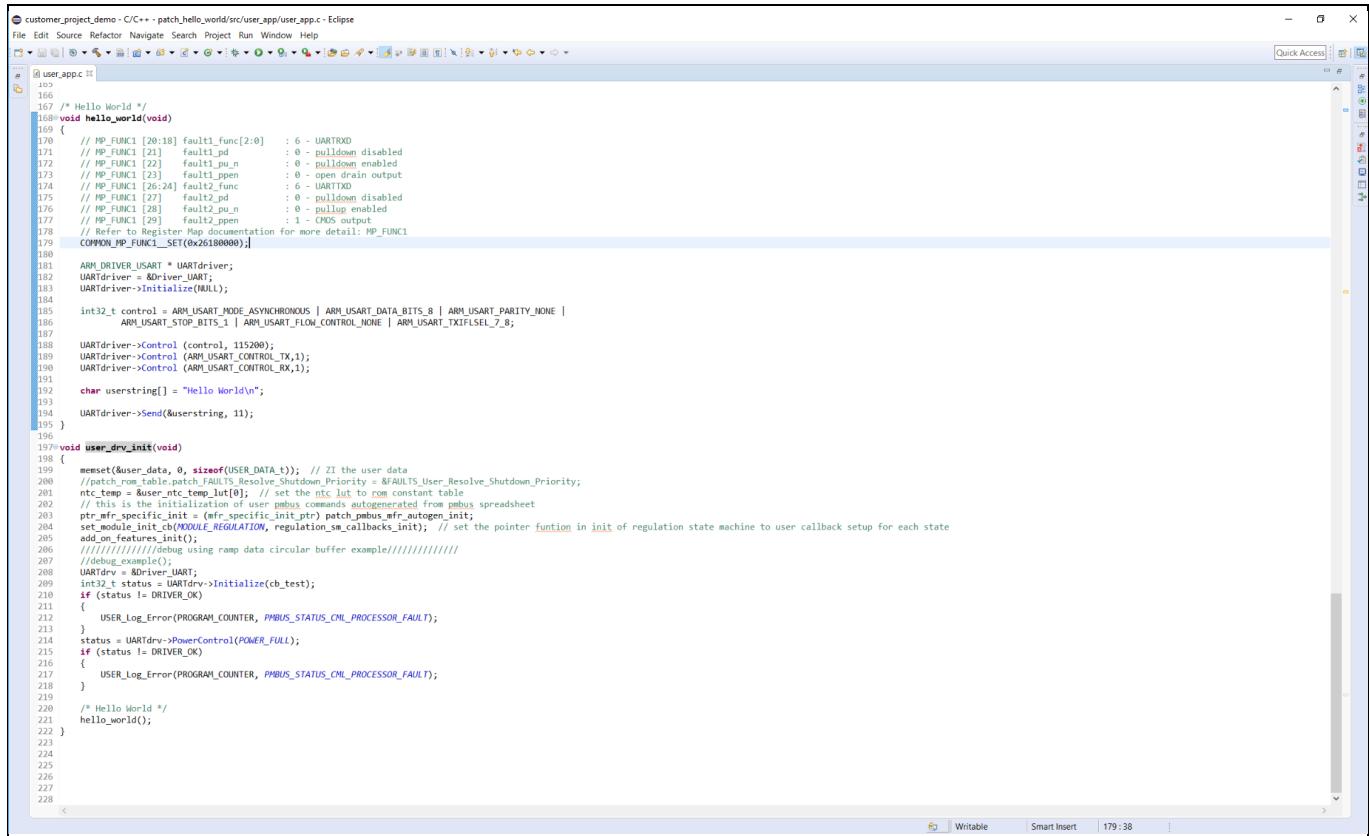


The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure under "patch_hello_world". The "src" folder contains "user_app", which further contains "user_app.c". Other files like "patch.h", "main.c", etc., are also listed.
- Code Editor:** Displays the content of "user_app.c". The code includes functions like "patch_initc()", "patch_pmbus_mfr_autogen_init()", and "user_drv_init()". It uses comments to explain the purpose of each section, such as "/// patch rom table example using rom data circular buffer example///////////".
- Outline View:** Shows a list of header files included in the project, such as "telemetry_drv.h", "regulation_api.h", "faults_apiph", etc.
- Console:** Shows the build log output, indicating a successful build with the message "15:38:06 Build Finished (took 2s.641ms)".

Implementing “Hello World” Project

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



```

customer_project_demo - C/C++ - patch_hello_world/src/user_app/user_app.c - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
user_app.c
165
166 /* Hello World */
167 void hello_world(void)
168 {
169     // MP_FUNC1 [20:18] fault1_func[2:0] : 0 - pulldown disabled
170     // MP_FUNC1 [21]   fault1_pd      : 0 - pulldown enabled
171     // MP_FUNC1 [22]   fault1_pu_n    : 0 - pullup enabled
172     // MP_FUNC1 [23]   fault1_ppen   : 0 - open drain output
173     // MP_FUNC1 [26:24] fault2_func[2:0] : 0 - USART_RXD
174     // MP_FUNC1 [27]   fault2_pd      : 0 - pulldown disabled
175     // MP_FUNC1 [28]   fault2_pu_n    : 0 - pullup enabled
176     // MP_FUNC1 [29]   fault2_ppen   : 1 - CMOS output
177     // Refer to Register Map documentation for more detail: MP_FUNC1
178     COMMON_MP_FUNC1_SET(0x26180000);
179
180     ARM_DRIVER_USART * UARTdriver;
181     UARTdriver = &Driver_USART;
182     UARTdriver->Initialize(NULL);
183
184     int32_t control = ARM_USART_MODE_ASYNCHRONOUS | ARM_USART_DATA_BITS_8 | ARM_USART_PARITY_NONE |
185                     ARM_USART_STOP_BITS_1 | ARM_USART_FLOW_CONTROL_NONE | ARM_USART_TXIFSEL_7_8;
186
187     UARTdriver->Control(control, 115200);
188     UARTdriver->Control(ARM_USART_CONTROL_TX,1);
189     UARTdriver->Control(ARM_USART_CONTROL_RX,1);
190
191     char userstring[] = "Hello World\n";
192
193     UARTdriver->Send(userstring, 11);
194 }
195
196 void user_drv_init(void)
197 {
198     memset(&user_data, 0, sizeof(USER_DATA_t)); // Zi the user data
199     //patch_rom_table.patchFAULTSResolve_ShutdownPriority(); //Patch ROM table to resolve Shutdown Priority
200     int new_Router_Index[10]; // set new int list to non constant table
201     // this is the initialization of user commandis autogenerated from previous spreadsheet
202     ptr_mfr_specific_init = (mfr_specific_init_ptr)patch_pmbus_mfr_autogen_init;
203     set_module_init_cb(MODULE_REGULATION, regulation_sm_callbacks_init); // set the pointer function in init of regulation state machine to user callback setup for each state
204     add_on_features_init();
205     ////////////////////using ramp data circular buffer example///////////
206     //patch_pmbus_mfr();
207     //UARTdrv = &Driver_USART;
208     UARTdrv = &Driver_USART;
209     int32_t status = UARTdrv->Initialize(cb_test);
210     if (status != DRIVER_OK)
211     {
212         USER_Log_Error(PROGRAM_COUNTER, PMBUS_STATUS_CHL_PROCESSOR_FAULT);
213     }
214     status = UARTdrv->PowerControl(POWER_FULL);
215     if (status != DRIVER_OK)
216     {
217         USER_Log_Error(PROGRAM_COUNTER, PMBUS_STATUS_CHL_PROCESSOR_FAULT);
218     }
219
220     /* Hello World */
221     hello_world();
222 }
223
224
225
226
227
228

```

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_USART;
    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:

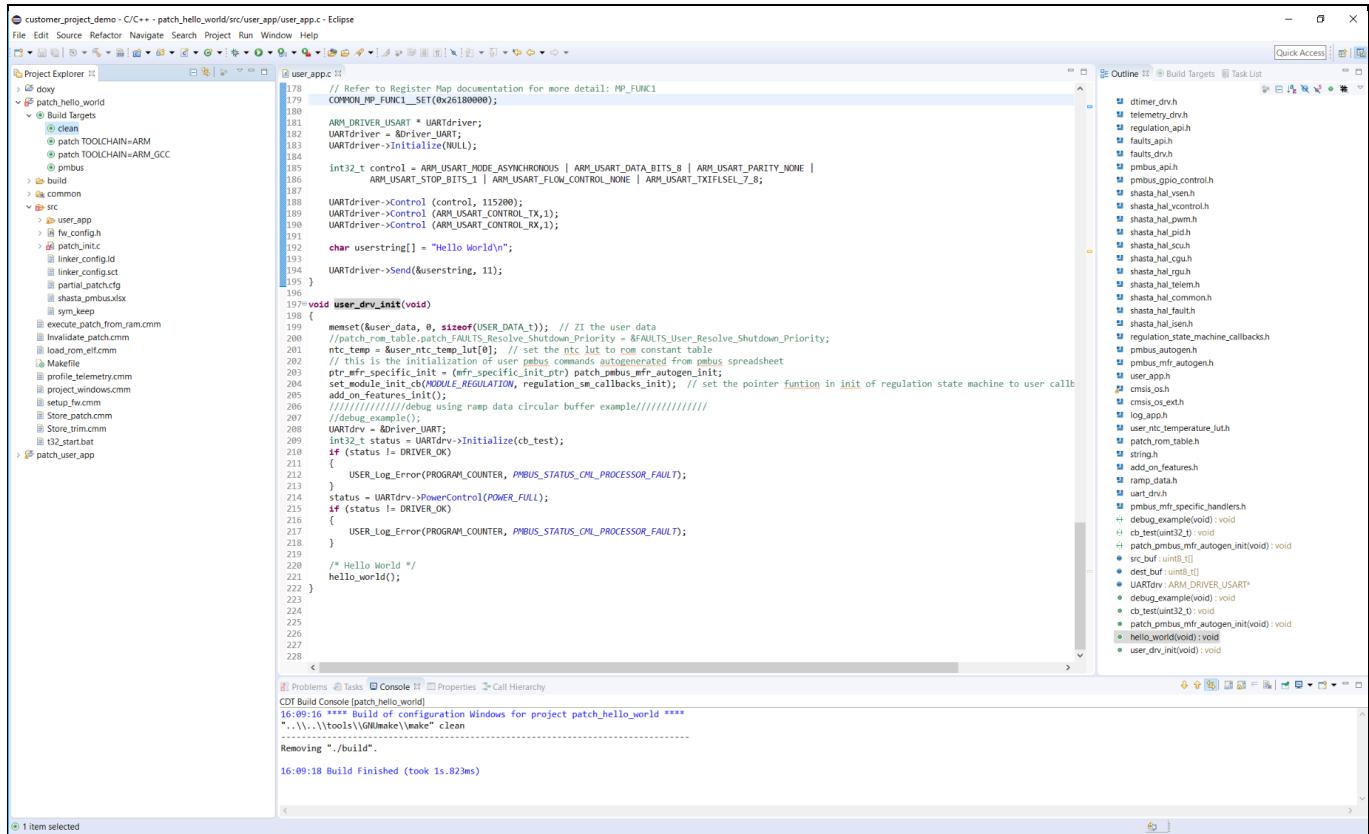
```
void user_drv_init(void)
{
    memset(&user_data, 0, sizeof(USER_DATA_t)); // ZI the user data
    //patch_rom_table.patch_FAULTS_Resolve_Shutdown_Priority =
&FAULTS_User_Resolve_Shutdown_Priority;
    ntc_temp = &user_ntc_temp_lut[0]; // set the ntc lut to rom constant table
    // this is the initialization 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 function 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();

}
```

Implementing “Hello World” Project

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



```

customer_project_demo - C/C++ - patch_hello_world/src/user_app/user_app.c - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer [ ] user_app [ ] Refer to Register Map documentation for more detail: MP_FUNC1
Patch Hello_World Build Targets Outline [ ] Build Targets Task List
src [ ] Clean [ ] dimmer_drv.h
patch_hello_world [ ] Build Targets [ ] telemetry_drv.h
[ ] Build Targets [ ] driver_ntc.h
[ ] Build Targets [ ] fauth_apiph.h
[ ] Build Targets [ ] fauth_drv.h
[ ] Build Targets [ ] pmbus_apiph.h
[ ] Build Targets [ ] pmbus_gpio_control.h
[ ] Build Targets [ ] shasta_hal_vsen.h
[ ] Build Targets [ ] shasta_hal_vcontrol.h
[ ] Build Targets [ ] shasta_hal_pwm.h
[ ] Build Targets [ ] shasta_hal_pdh.h
[ ] Build Targets [ ] shasta_hal_scub.h
[ ] Build Targets [ ] shasta_hal_cgph.h
[ ] Build Targets [ ] shasta_hal_gph.h
[ ] Build Targets [ ] shasta_hal_telem.h
[ ] Build Targets [ ] shasta_hal_cooling.h
[ ] Build Targets [ ] shasta_hal_barth.h
[ ] Build Targets [ ] shasta_hal_sen.h
[ ] Build Targets [ ] regulation_state_machine_callbacks.h
[ ] Build Targets [ ] pmbus_autogen.h
[ ] Build Targets [ ] pmbus_mfr_autogen.h
[ ] Build Targets [ ] user_app.h
[ ] Build Targets [ ] cmsis_os.h
[ ] Build Targets [ ] cmsis_os_exth.h
[ ] Build Targets [ ] log_app.h
[ ] Build Targets [ ] user_ntc_temperature_lut.h
[ ] Build Targets [ ] patch_rom_table.h
[ ] Build Targets [ ] string.h
[ ] Build Targets [ ] add_on_features.h
[ ] Build Targets [ ] ramp_data.h
[ ] Build Targets [ ] pmbs_mfr_specific_handlers.h
[ ] Build Targets [ ] debug_example(void) void
[ ] Build Targets [ ] ch_tetrim32_t void
[ ] Build Targets [ ] patch_pmbus_mfr_autogen_init(void) void
[ ] Build Targets [ ] src_buf uint8_t
[ ] Build Targets [ ] dest_buf uint8_t
[ ] Build Targets [ ] UARTdrv_>ARM_DRIVER_USART*
[ ] Build Targets [ ] debug_example(void) void
[ ] Build Targets [ ] cb_testunit32_t void
[ ] Build Targets [ ] patch_pmbus_mfr_autogen_init(void) void
[ ] Build Targets [ ] hello_world(void) void
[ ] Build Targets [ ] user_drv_init(void) void

customer_project_demo - C/C++ - patch_hello_world/src/user_app/user_app.c - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer [ ] user_app [ ] Refer to Register Map documentation for more detail: MP_FUNC1
Patch Hello_World Build Targets Outline [ ] Build Targets Task List
src [ ] Clean [ ] dimmer_drv.h
patch_hello_world [ ] Build Targets [ ] telemetry_drv.h
[ ] Build Targets [ ] driver_ntc.h
[ ] Build Targets [ ] fauth_apiph.h
[ ] Build Targets [ ] fauth_drv.h
[ ] Build Targets [ ] pmbus_apiph.h
[ ] Build Targets [ ] pmbus_gpio_control.h
[ ] Build Targets [ ] shasta_hal_vsen.h
[ ] Build Targets [ ] shasta_hal_vcontrol.h
[ ] Build Targets [ ] shasta_hal_pwm.h
[ ] Build Targets [ ] shasta_hal_pdh.h
[ ] Build Targets [ ] shasta_hal_scub.h
[ ] Build Targets [ ] shasta_hal_cgph.h
[ ] Build Targets [ ] shasta_hal_gph.h
[ ] Build Targets [ ] shasta_hal_telem.h
[ ] Build Targets [ ] shasta_hal_cooling.h
[ ] Build Targets [ ] shasta_hal_barth.h
[ ] Build Targets [ ] shasta_hal_sen.h
[ ] Build Targets [ ] regulation_state_machine_callbacks.h
[ ] Build Targets [ ] pmbus_autogen.h
[ ] Build Targets [ ] pmbus_mfr_autogen.h
[ ] Build Targets [ ] user_app.h
[ ] Build Targets [ ] cmsis_os.h
[ ] Build Targets [ ] cmsis_os_exth.h
[ ] Build Targets [ ] log_app.h
[ ] Build Targets [ ] user_ntc_temperature_lut.h
[ ] Build Targets [ ] patch_rom_table.h
[ ] Build Targets [ ] string.h
[ ] Build Targets [ ] add_on_features.h
[ ] Build Targets [ ] ramp_data.h
[ ] Build Targets [ ] pmbs_mfr_specific_handlers.h
[ ] Build Targets [ ] debug_example(void) void
[ ] Build Targets [ ] ch_tetrim32_t void
[ ] Build Targets [ ] patch_pmbus_mfr_autogen_init(void) void
[ ] Build Targets [ ] src_buf uint8_t
[ ] Build Targets [ ] dest_buf uint8_t
[ ] Build Targets [ ] UARTdrv_>ARM_DRIVER_USART*
[ ] Build Targets [ ] debug_example(void) void
[ ] Build Targets [ ] cb_testunit32_t void
[ ] Build Targets [ ] patch_pmbus_mfr_autogen_init(void) void
[ ] Build Targets [ ] hello_world(void) void
[ ] Build Targets [ ] user_drv_init(void) void

Problems Tasks Console Properties Call Hierarchy
Build Console [patch_hello_world]
16:09:16 *** Build of configuration Windows for project patch_hello_world ***
..\..\..\tools\GNULinux\make clean
-----
Removing "./build".
16:09:18 Build Finished (took 1s.823ms)

1 item selected

```

Implementing “Hello World” Project

5.4

STEP 4: Compile and Build Patch (ARM_GCC)

```

customer_project_demo - C/C++ - patch_hello_world/src/user_app/user_app.c - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer Build Targets Task List
patch_hello_world
  Build Targets
    Clean
      Clean TOOLCHAIN=ARM
      Clean TOOLCHAIN=ARM_GCC
      Clean pmbus
      Clean common
      Clean user_app
      Clean fw_config.h
      Clean patch_ntc
      Clean linker_config_id
      Clean linker_config.sct
      Clean partial_patchcfg
      Clean shasta_pmbusxx
      Clean sys_keen
      Clean execute_patch_from_ram.cmm
      Clean Invalidate_patch.cmm
      Clean load_rom_efcm.cmm
      Clean Makefile
      Clean profile_telemetry.cmm
      Clean project_windows.cmm
      Clean setup_fwm.cmm
      Clean Store_patch.cmm
      Clean Store_trim.cmm
      Clean t32_start.bat
      Clean patch_user_app
  src
    user_app
      fw_config.h
      patch_ntc.h
      linker_config_id
      linker_config.sct
      partial_patchcfg
      shasta_pmbusxx
      sys_keen
      execute_patch_from_ram.cmm
      Invalidate_patch.cmm
      load_rom_efcm.cmm
      Makefile
      profile_telemetry.cmm
      project_windows.cmm
      setup_fwm.cmm
      Store_patch.cmm
      Store_trim.cmm
      t32_start.bat
      patch_user_app
  build
  doc
  patch_hello_world
    Build Targets
      Clean
        Clean TOOLCHAIN=ARM
        Clean TOOLCHAIN=ARM_GCC
        Clean pmbus
        Clean common
        Clean user_app
        Clean fw_config.h
        Clean patch_ntc
        Clean linker_config_id
        Clean linker_config.sct
        Clean partial_patchcfg
        Clean shasta_pmbusxx
        Clean sys_keen
        Clean execute_patch_from_ram.cmm
        Clean Invalidate_patch.cmm
        Clean load_rom_efcm.cmm
        Clean Makefile
        Clean profile_telemetry.cmm
        Clean project_windows.cmm
        Clean setup_fwm.cmm
        Clean Store_patch.cmm
        Clean Store_trim.cmm
        Clean t32_start.bat
        Clean patch_user_app
    src
      user_app
        fw_config.h
        patch_ntc.h
        linker_config_id
        linker_config.sct
        partial_patchcfg
        shasta_pmbusxx
        sys_keen
        execute_patch_from_ram.cmm
        Invalidate_patch.cmm
        load_rom_efcm.cmm
        Makefile
        profile_telemetry.cmm
        project_windows.cmm
        setup_fwm.cmm
        Store_patch.cmm
        Store_trim.cmm
        t32_start.bat
        patch_user_app
    build
    doc
  outline
  Build Targets Task List
  dimmer_drv.h
  telemetry_drv.h
  sensor_api.h
  faults.h
  fault_drv.h
  pmbus_api.h
  pmbus_gpio_control.h
  shasta_hal_vsen.h
  shasta_hal_vcontrol.h
  shasta_hal_pwm.h
  shasta_hal_pid.h
  shasta_hal_scub.h
  shasta_hal_cgub.h
  shasta_hal_gpiub.h
  shasta_hal_telem.h
  shasta_hal_cooling.h
  shasta_hal_barth.h
  shasta_hal_sen.h
  regulation_state_machine_callbacks.h
  pmbus_autogen.h
  pmbus_mfr_autogen.h
  user_app.h
  cmsis_os.h
  cmsis_os_exth.h
  log_app.h
  user_ntc_temperature_lut.h
  patch_rom_table.h
  string.h
  add_on_features.h
  ramp_data.h
  pmbusDrv.h
  pmbus_mfr_specific_handlers.h
  debug_example(void) void
  ch_testim32(j) void
  patch_pmbus_mfr_autogen_init(void) void
  src_buf uint8_t]
  dest_buf uint8_t]
  UARTdrv<ARM_DRIVER_USART>
  debug_example(void) void
  cb_testim32(j) void
  patch_pmbus_mfr_autogen_init(void) void
  hello_world(void) void
  user_drv_init(void) void
  
```

Problems Tasks Console Properties Call Hierarchy

CMake Build Console [patch_hello_world]

Generating listing file ..\build\patch\patch.lst from ".\build\patch\patch.eif"

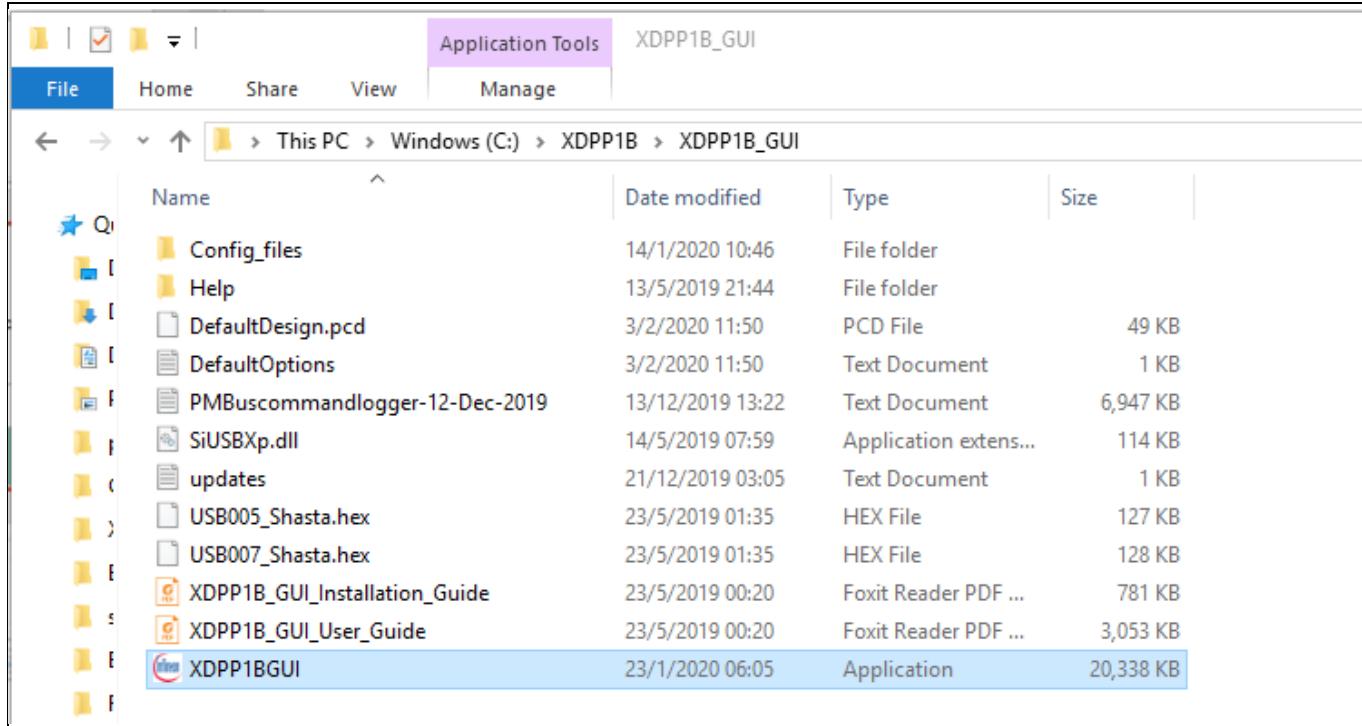
Finished building ".\build\patch\patch.eif".

make[1]: Leaving directory 'C:/XOPP1B/XDPP10_fw/projects/patch_hello_world'

16:09:59 Build Finished (took 20s.567ms)

6 Downloading Patch to OTP

6.1 STEP 1: Open XDPP1B GUI



6.2 STEP 2: Connect PMBus Dongle and XDPP1B

Photo of dongle:

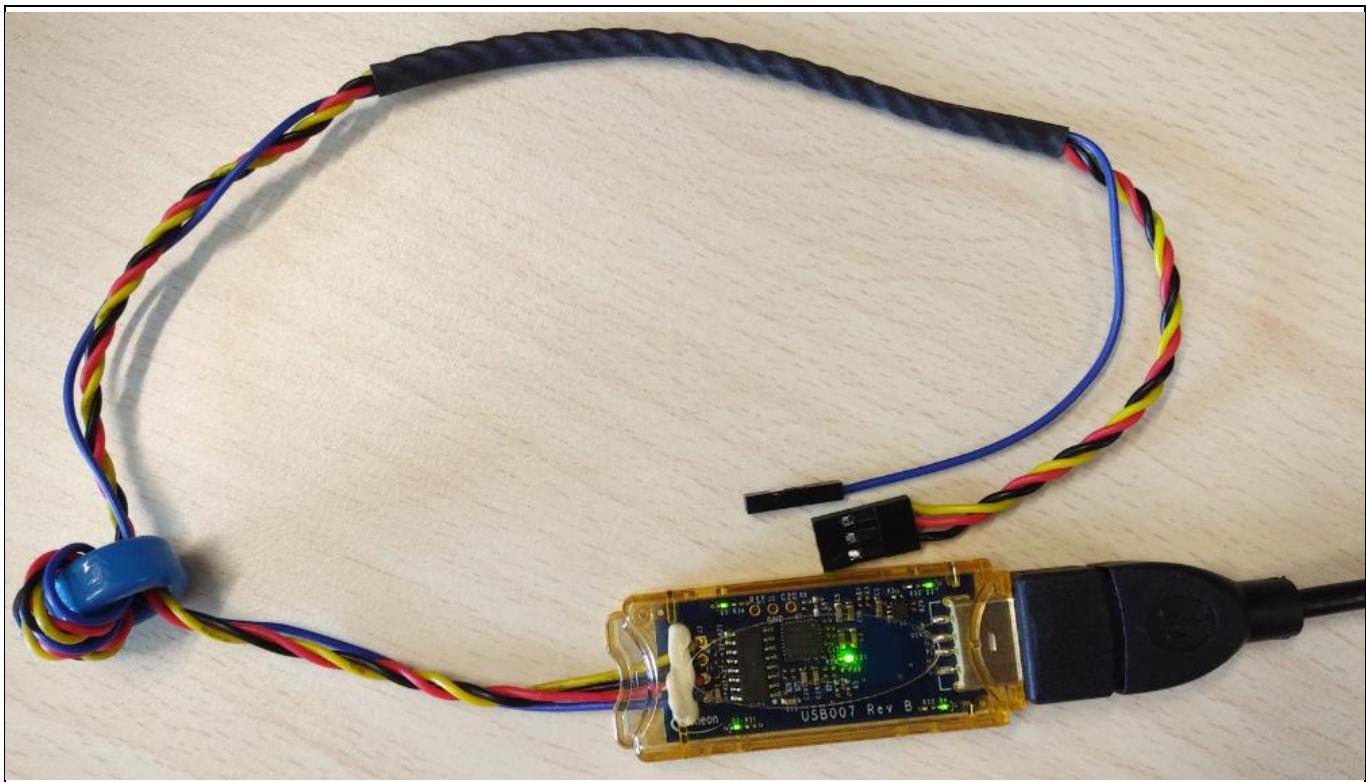
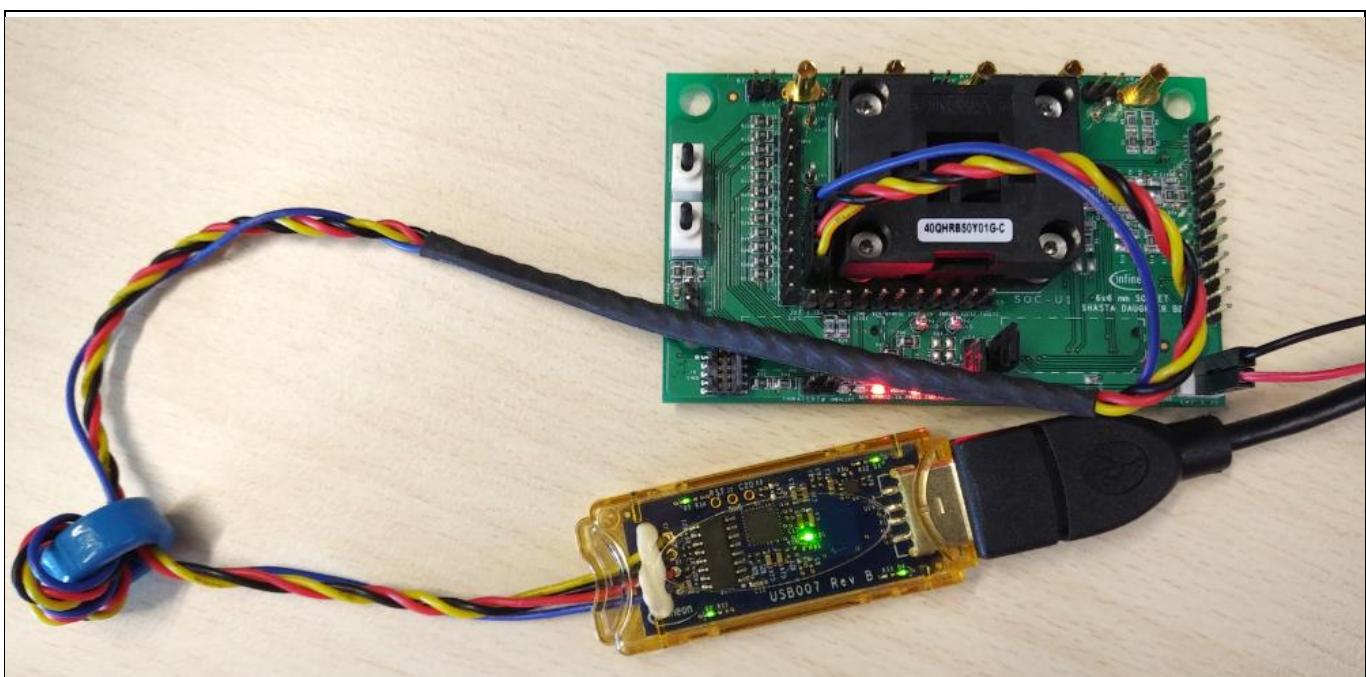
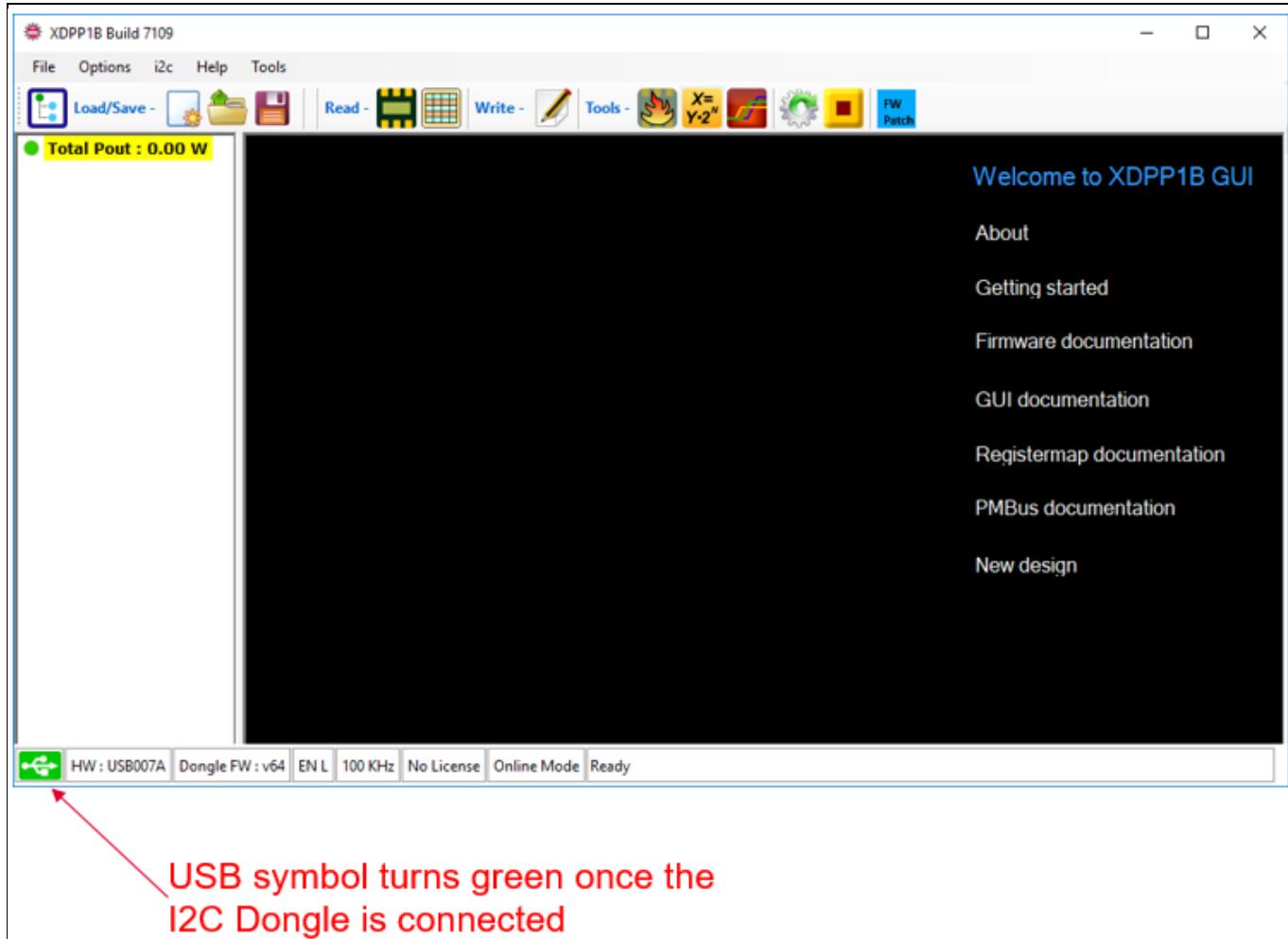
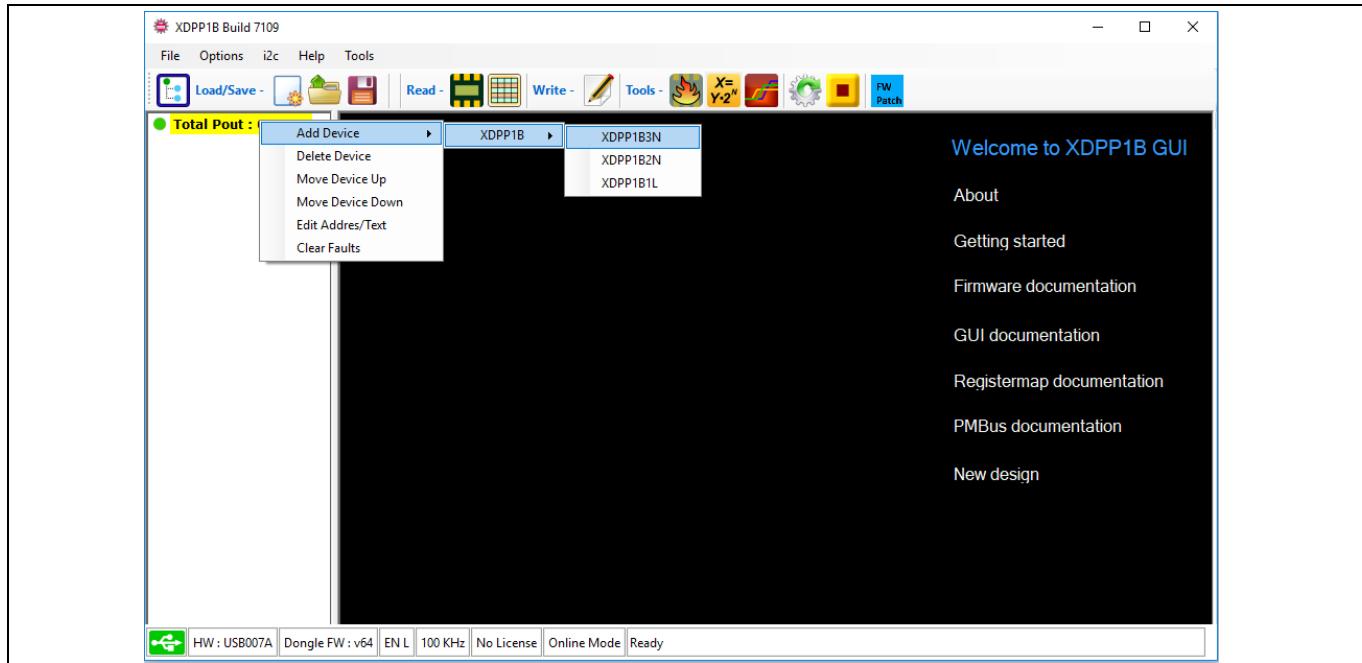


Photo of connection:

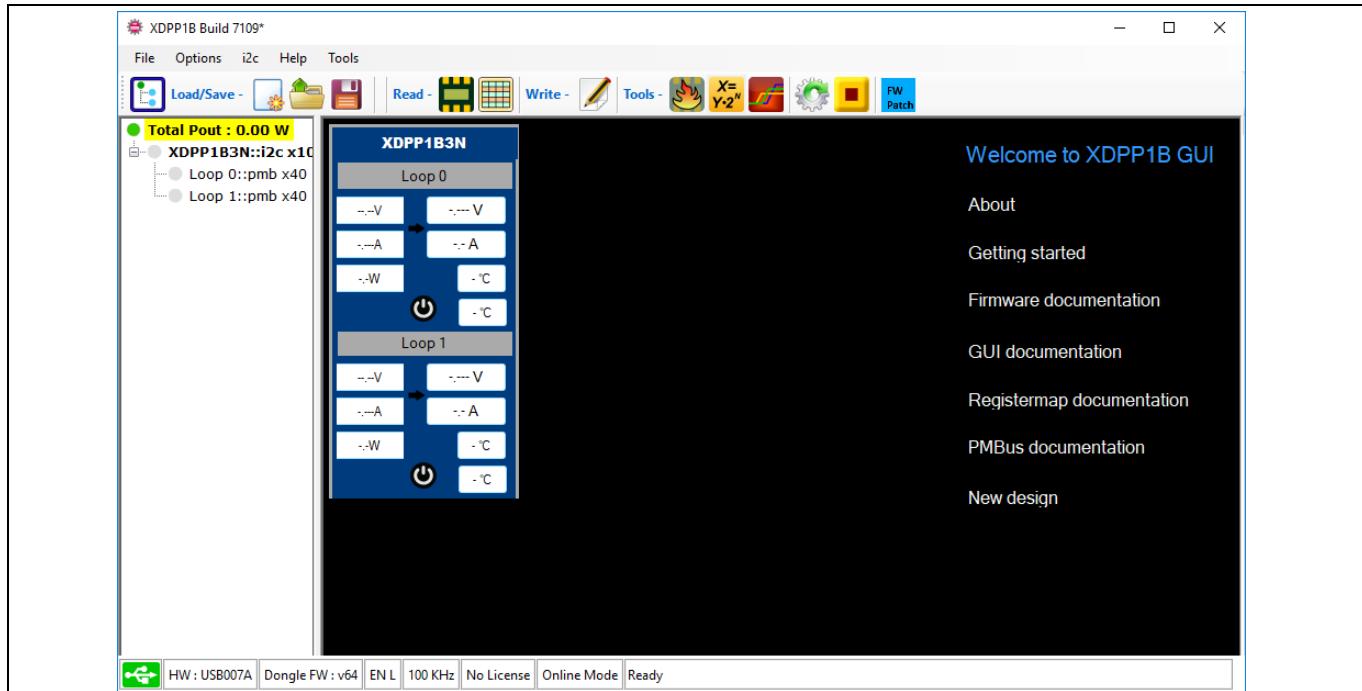




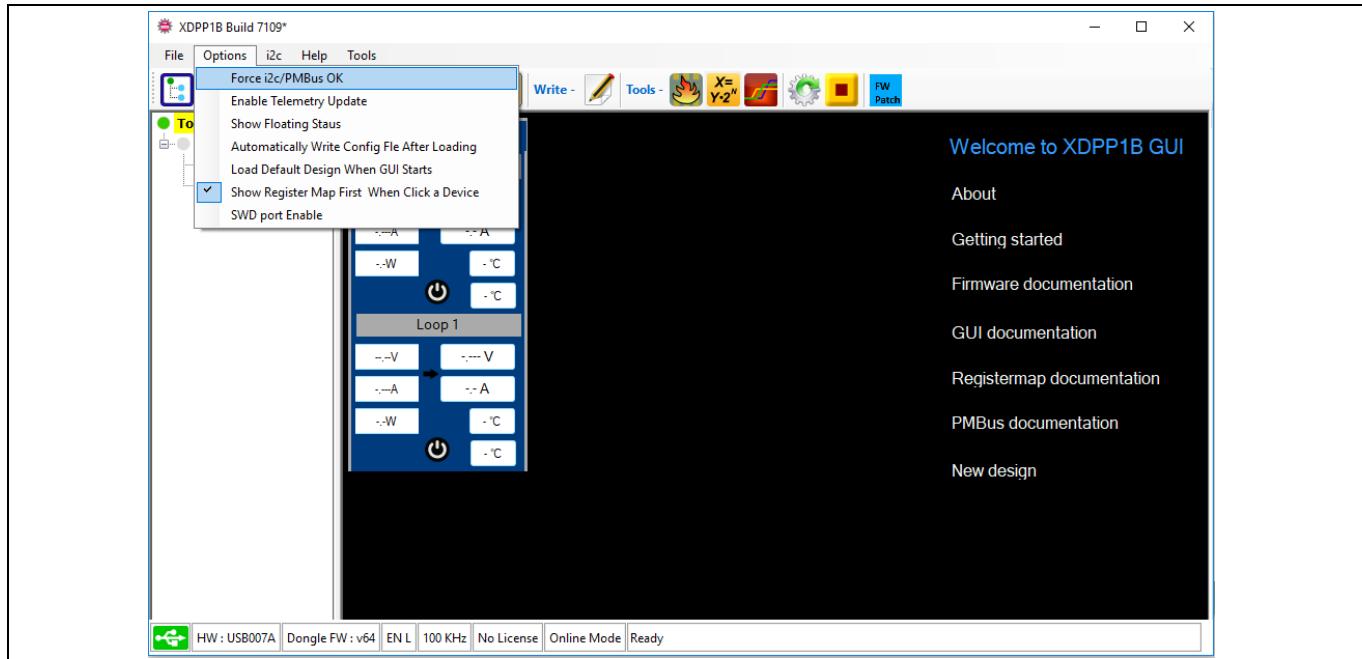
6.3 STEP 3: Add Device



After add device:



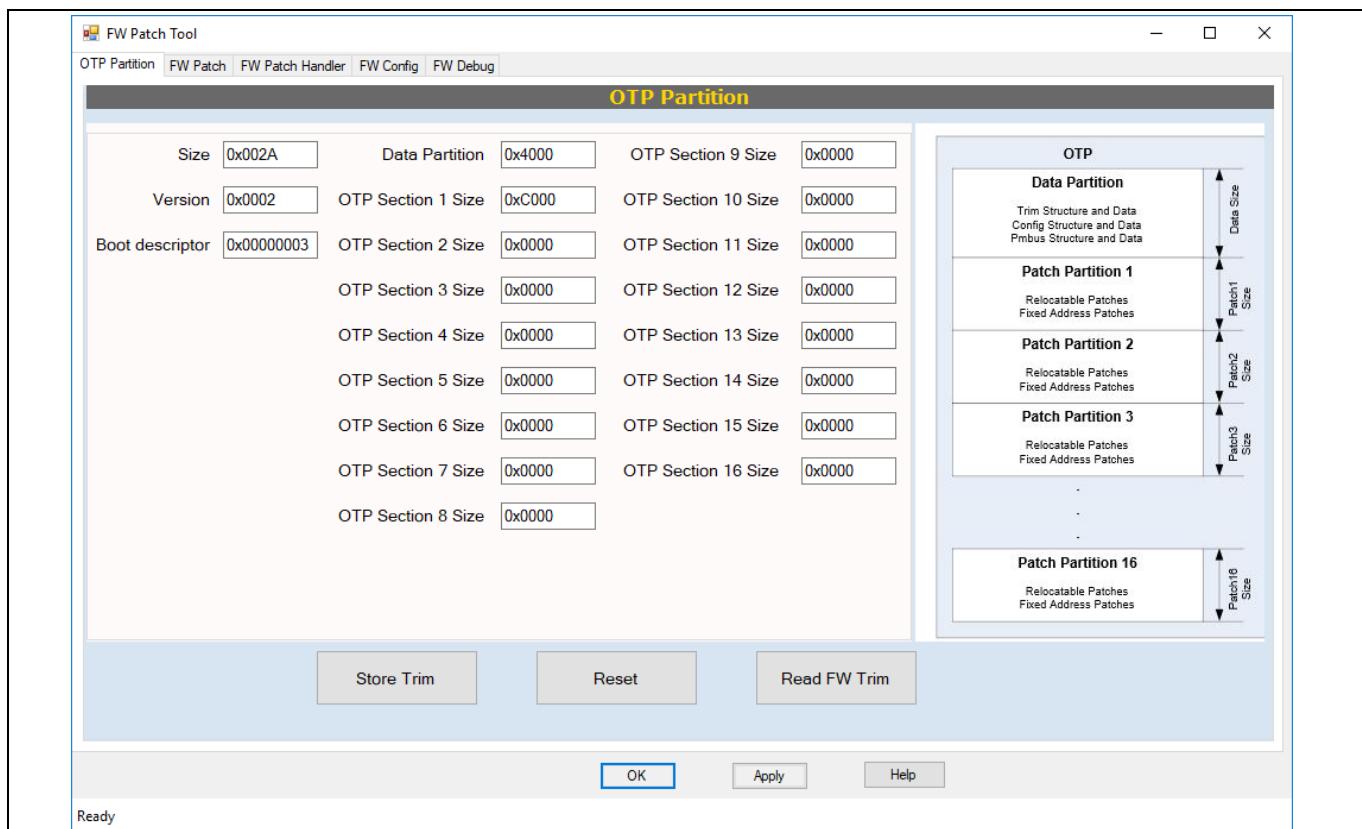
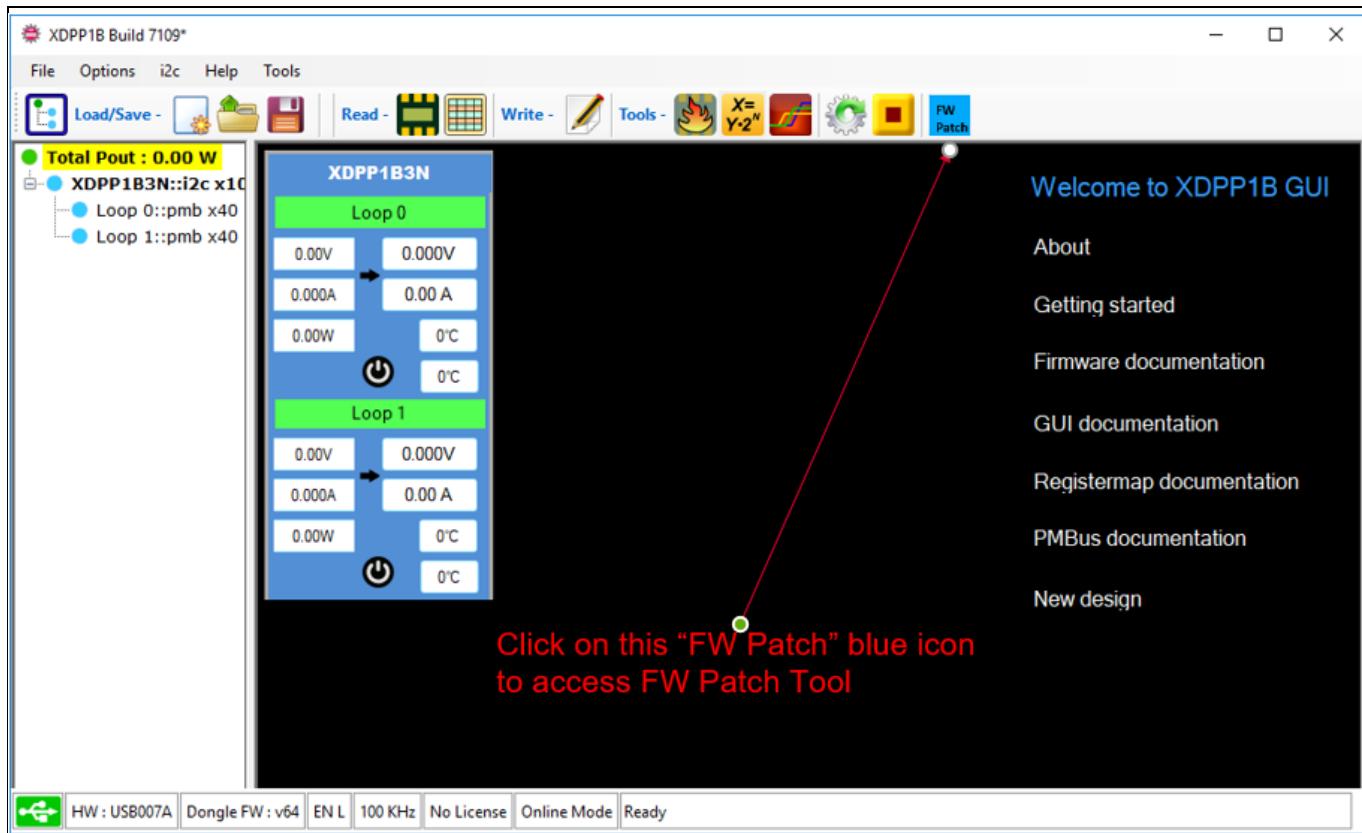
6.4 STEP 4: Force I2C/PMBus OK



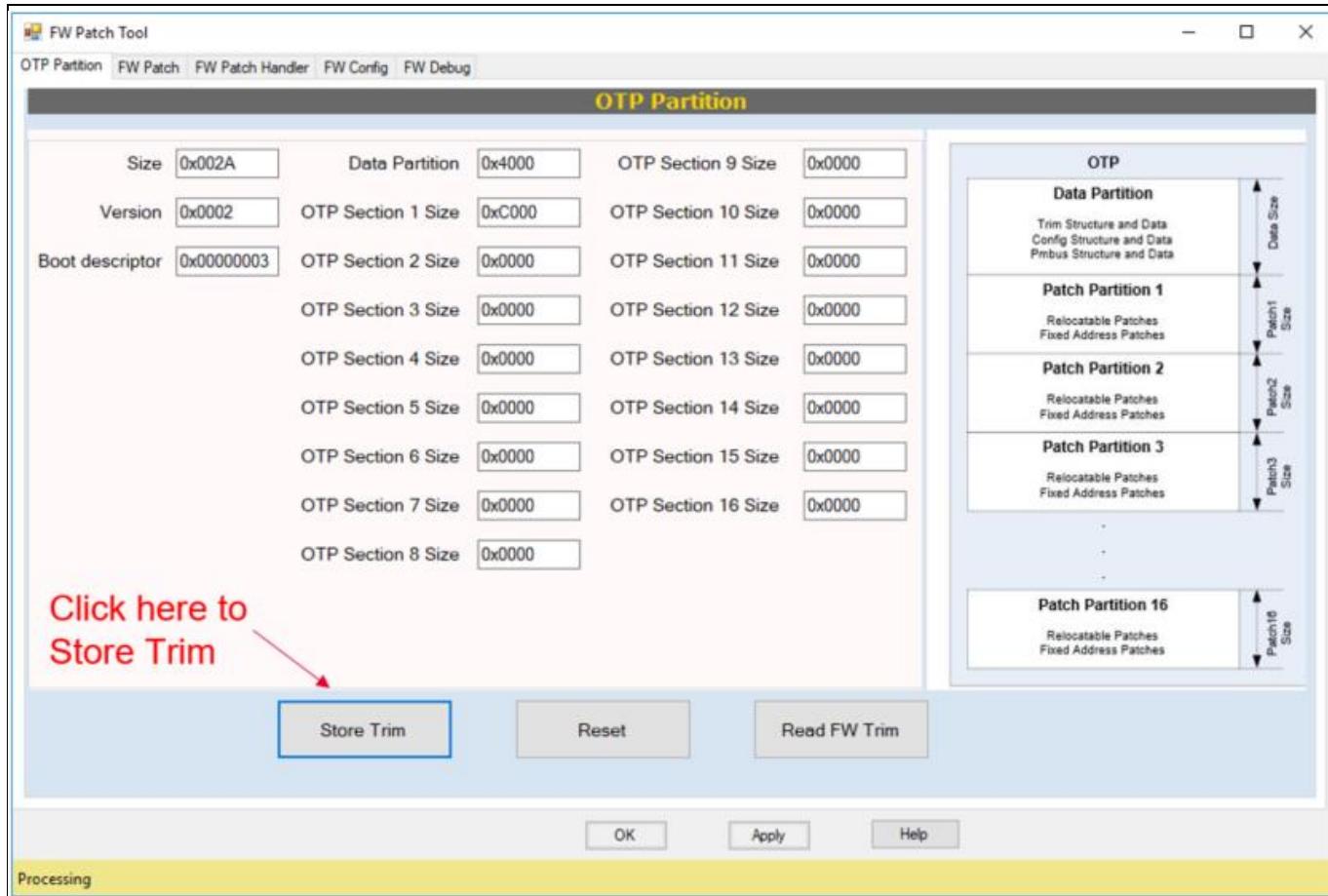
After Force I2C/PMBus:



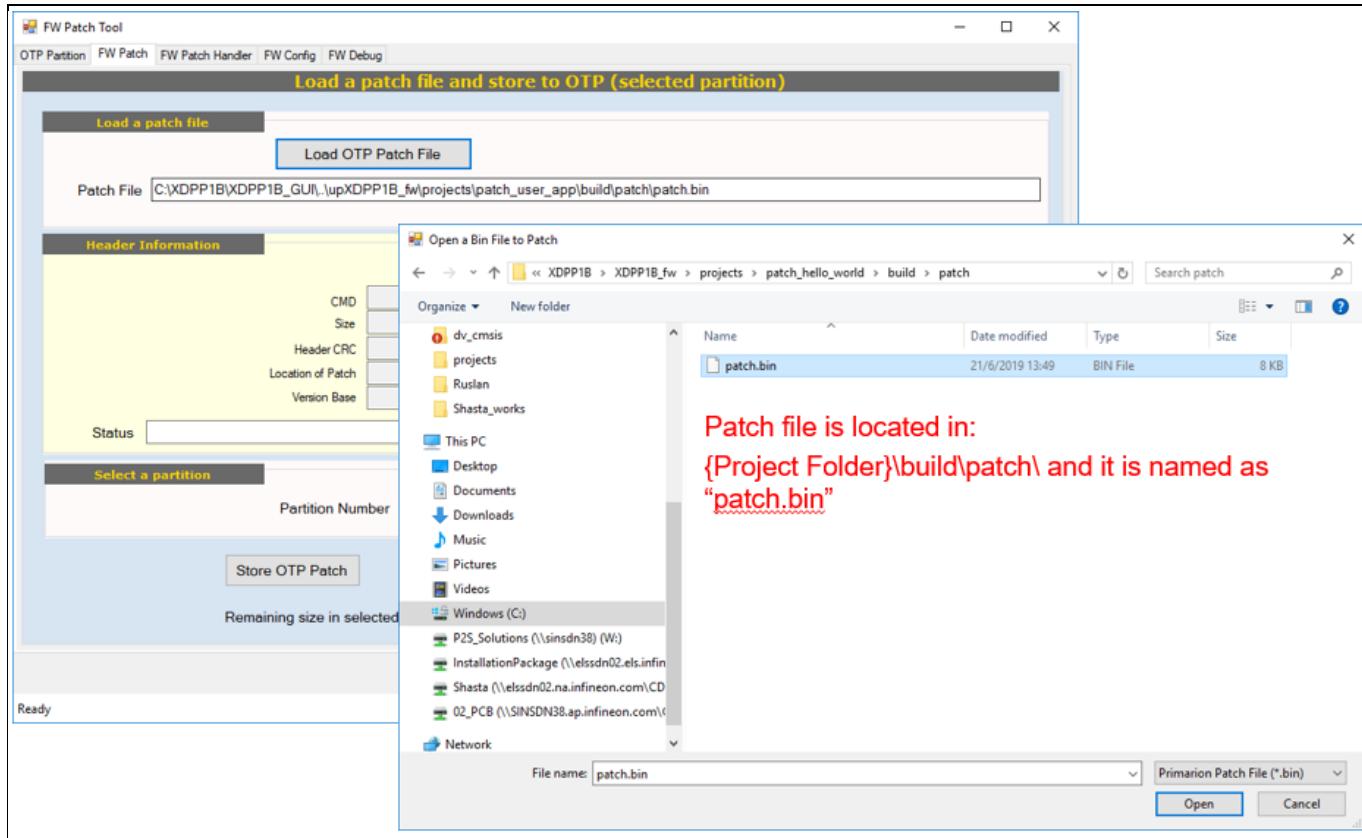
6.5 STEP 5: Access FW Patch Tool



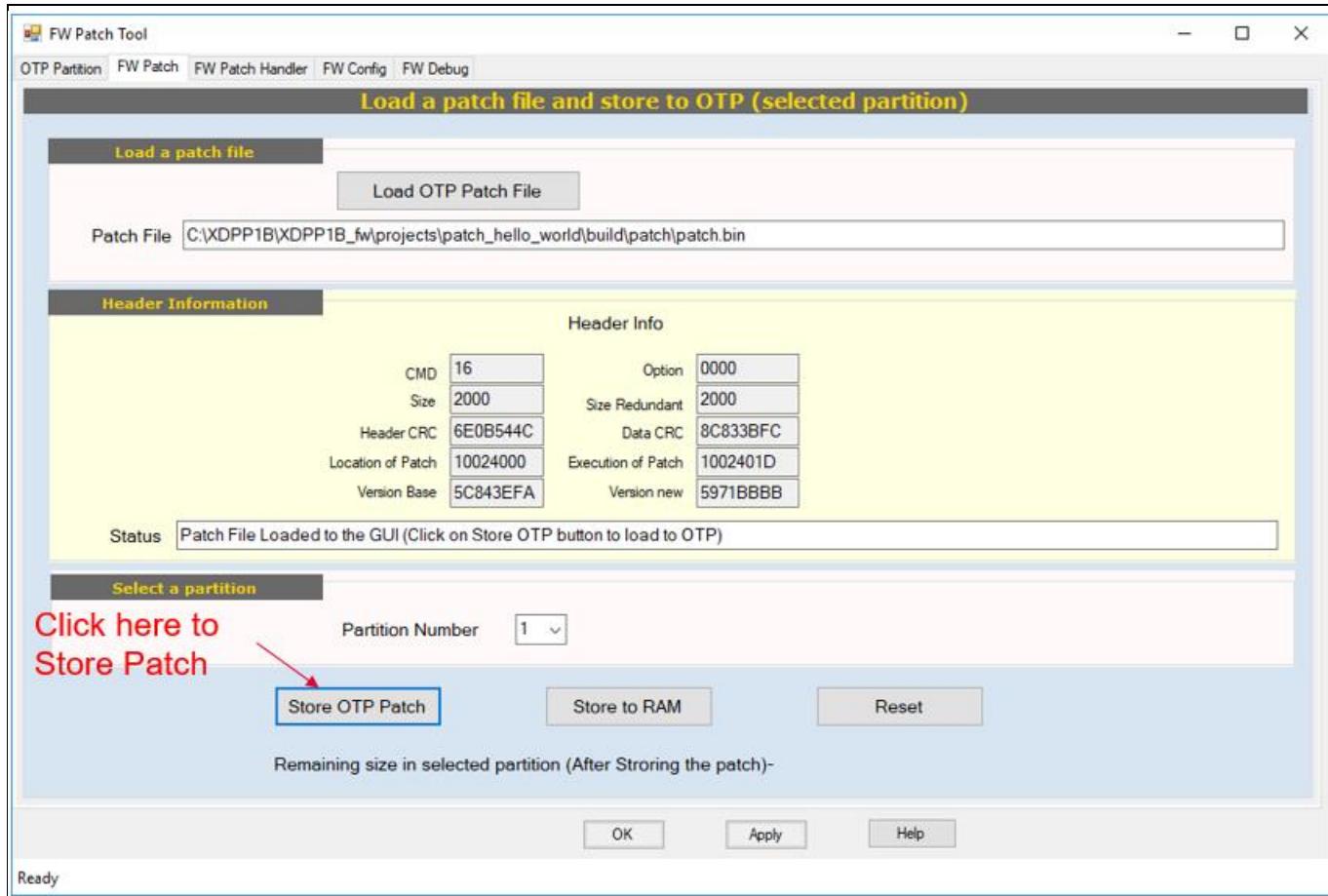
6.6 STEP 6: Store Trim



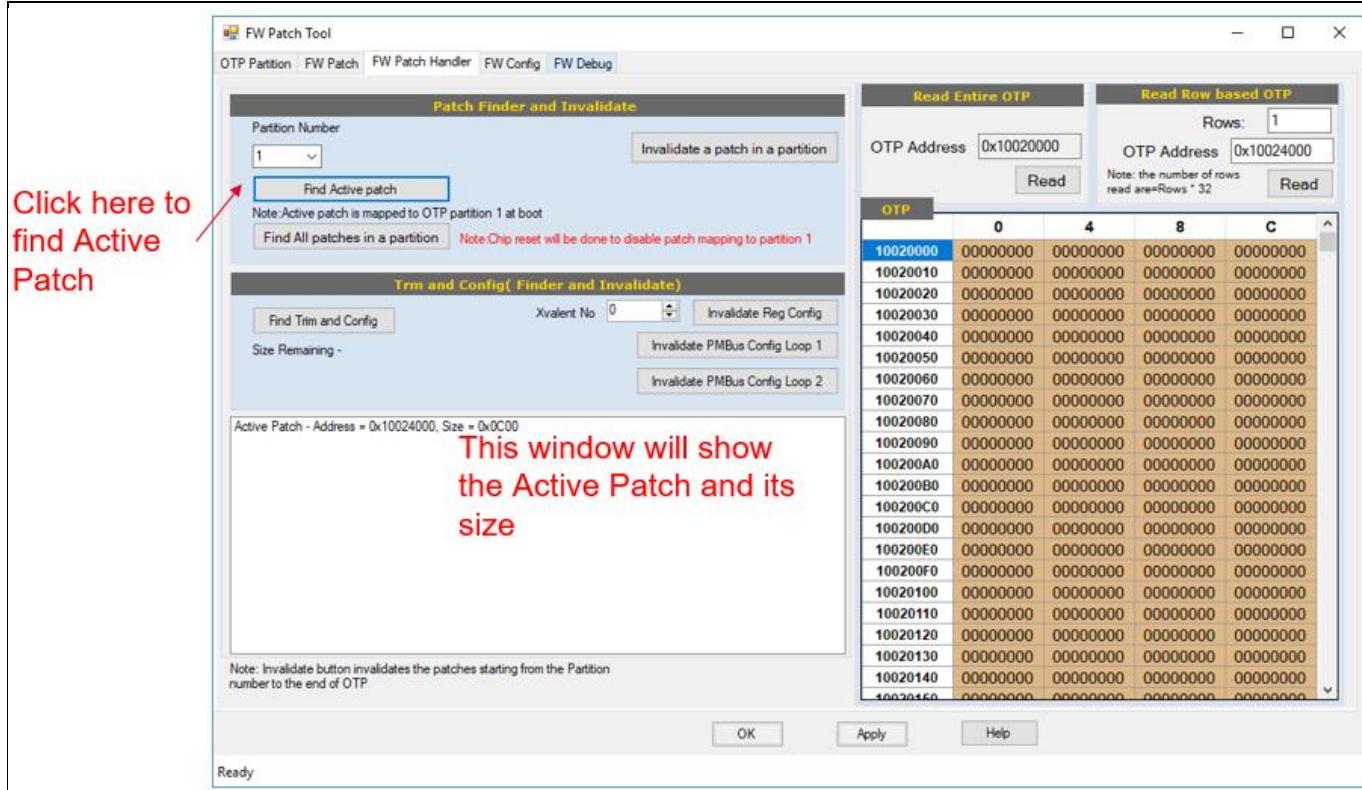
6.7 STEP 7: Load Patch File



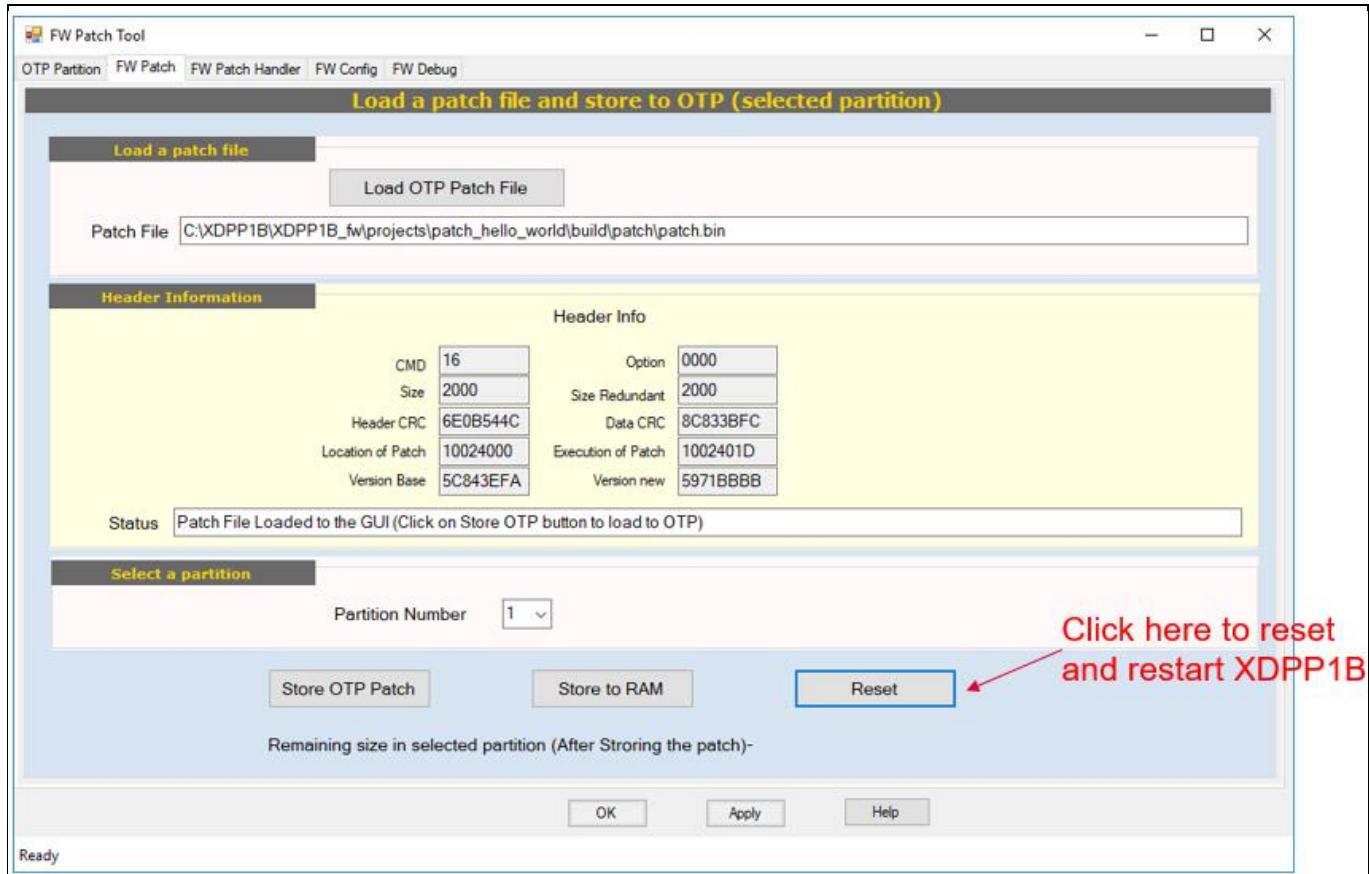
6.8 STEP 8: Store Patch File to OTP



6.9 STEP 9: Check for Active Patch

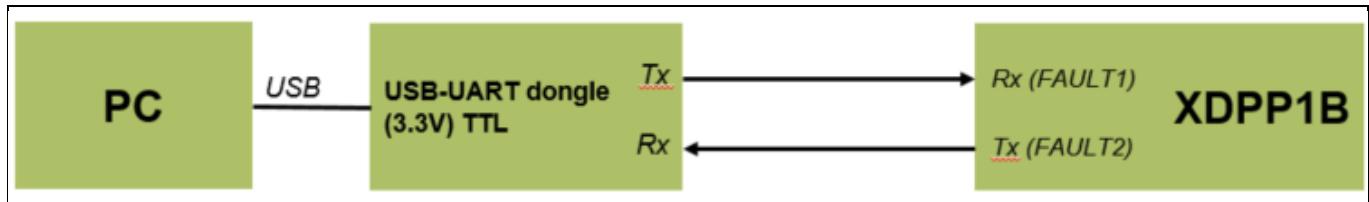


6.10 STEP 10: Patch is loaded. Restart XDPP1100.

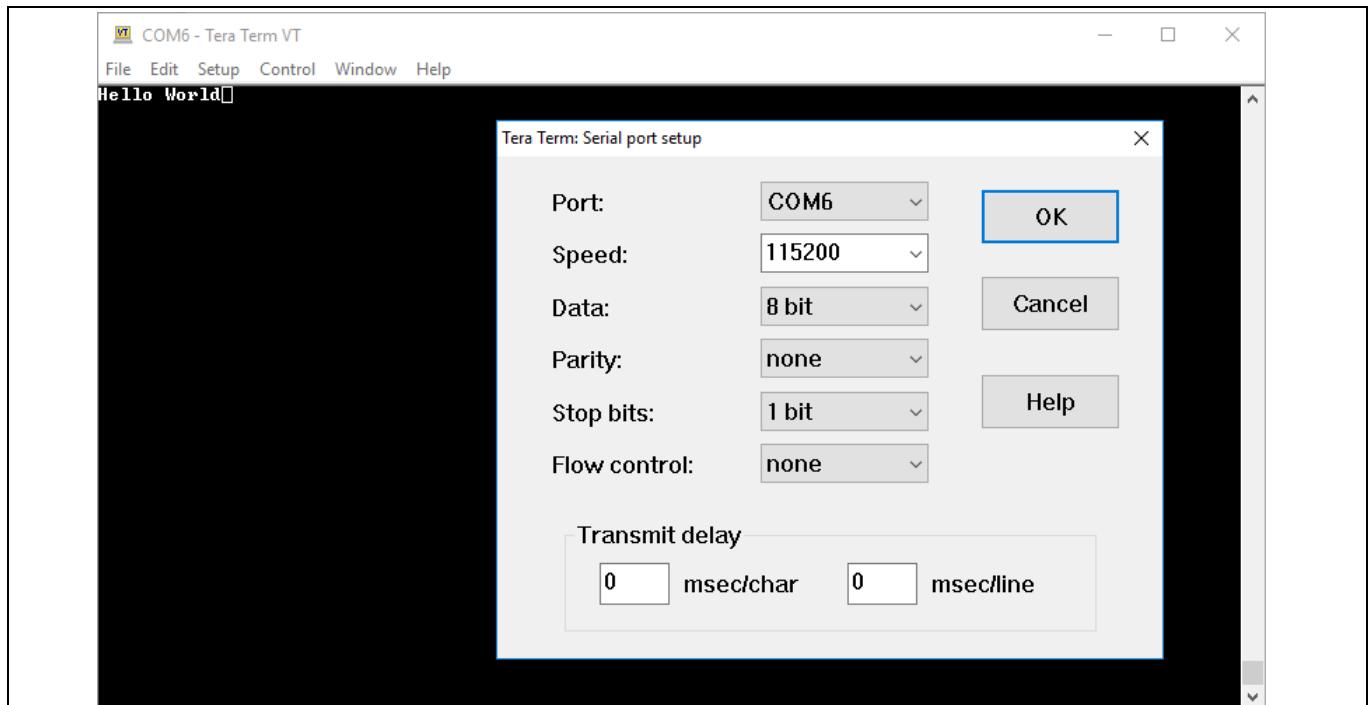


7 Monitor UART Traffic

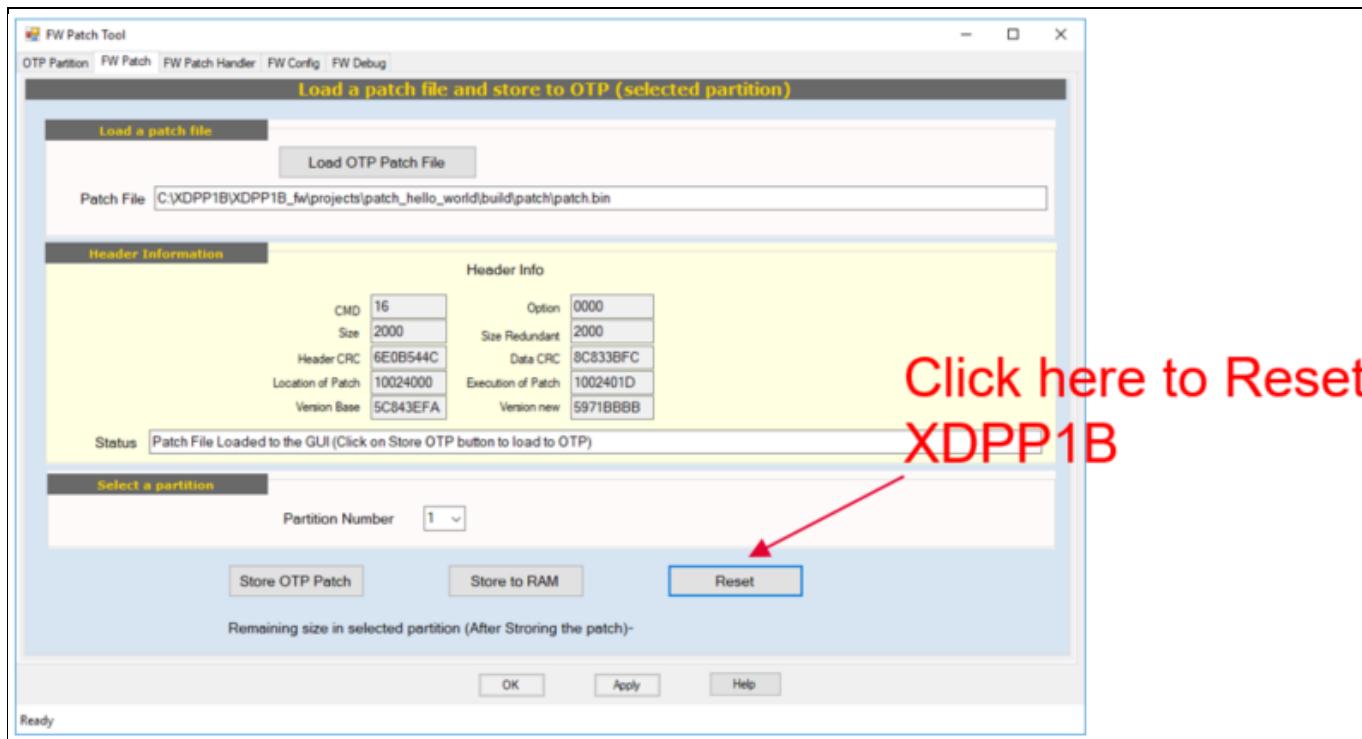
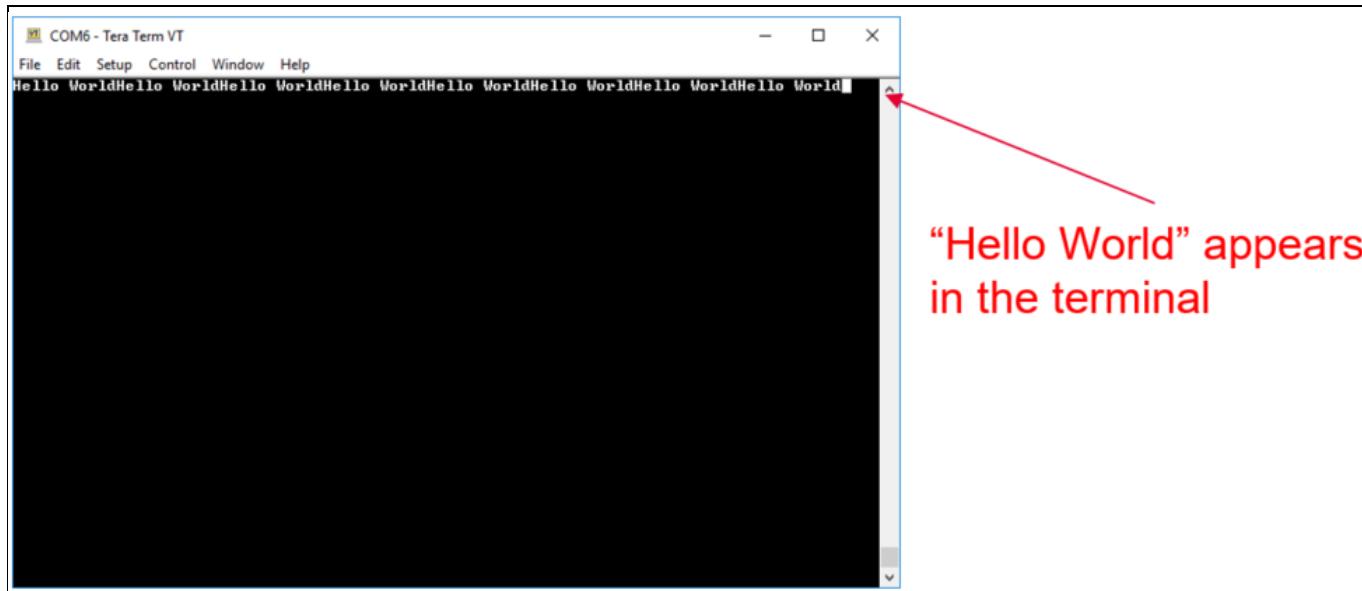
7.1 STEP 1: Connect XDPP1100 with USB-UART dongle



7.2 STEP 2: Monitor UART traffic

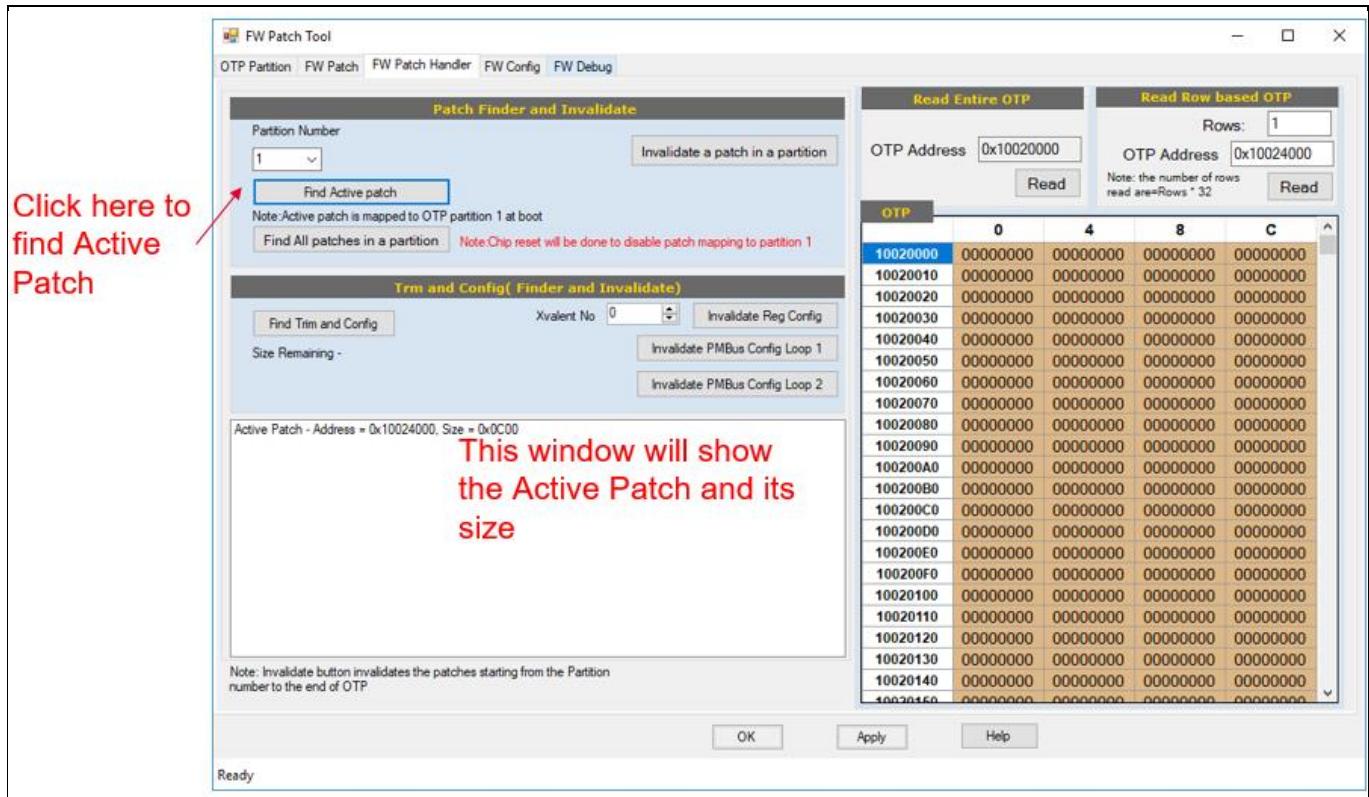


7.3

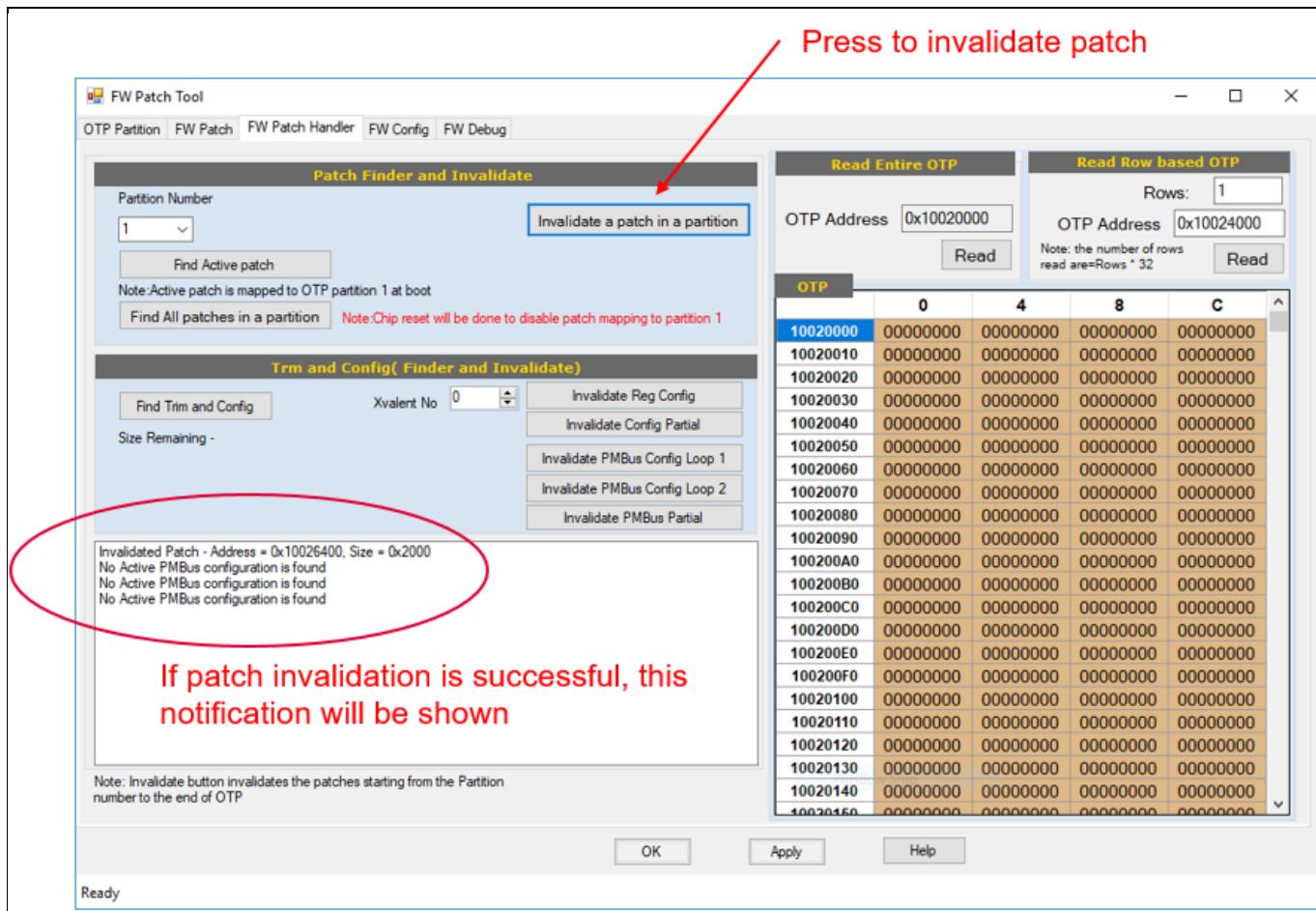
STEP 3: “Hello World” will appear

8 Invalidating Patch from OTP

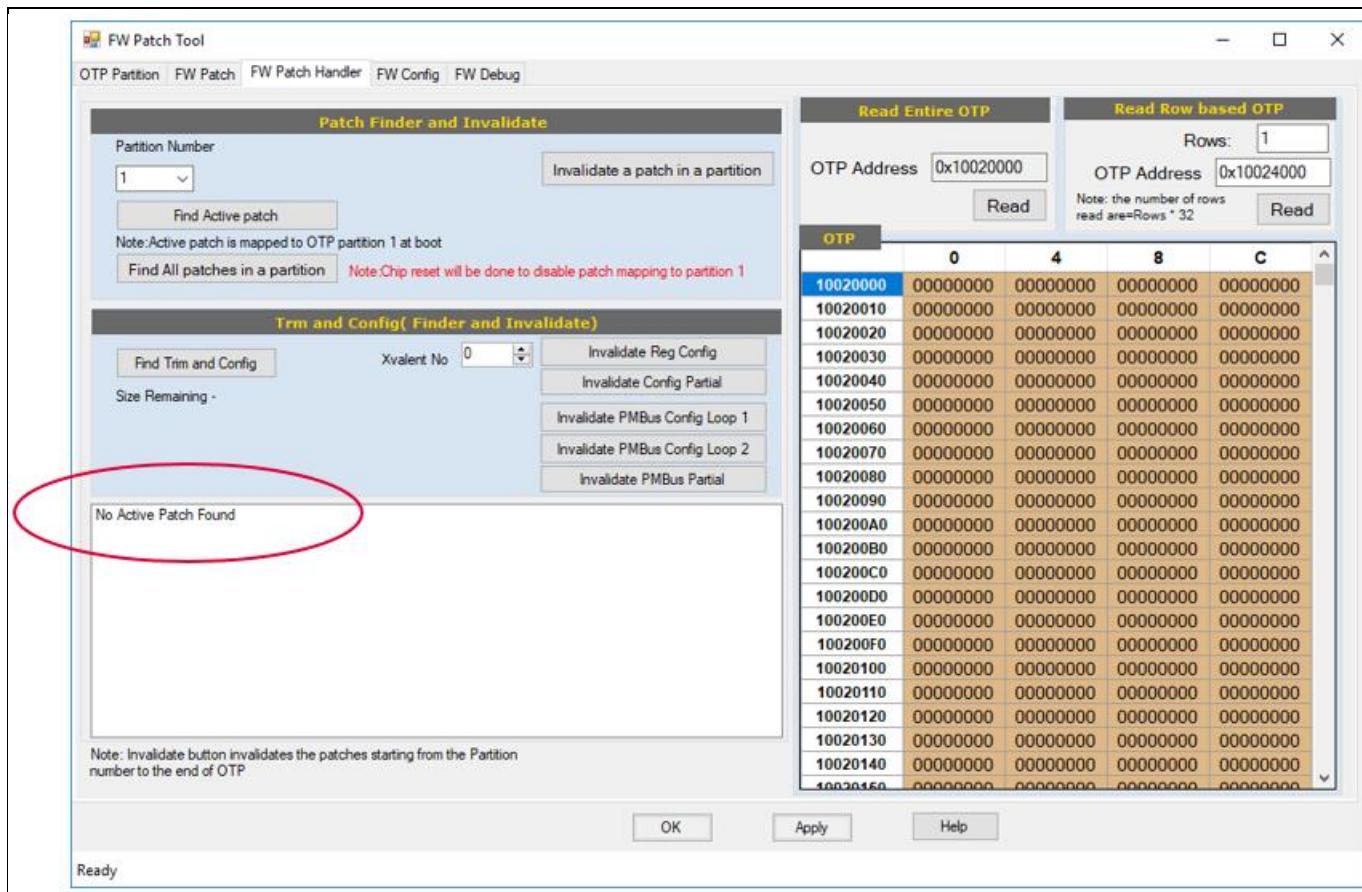
8.1 STEP 1: Ensure there is an active Patch in the OTP



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

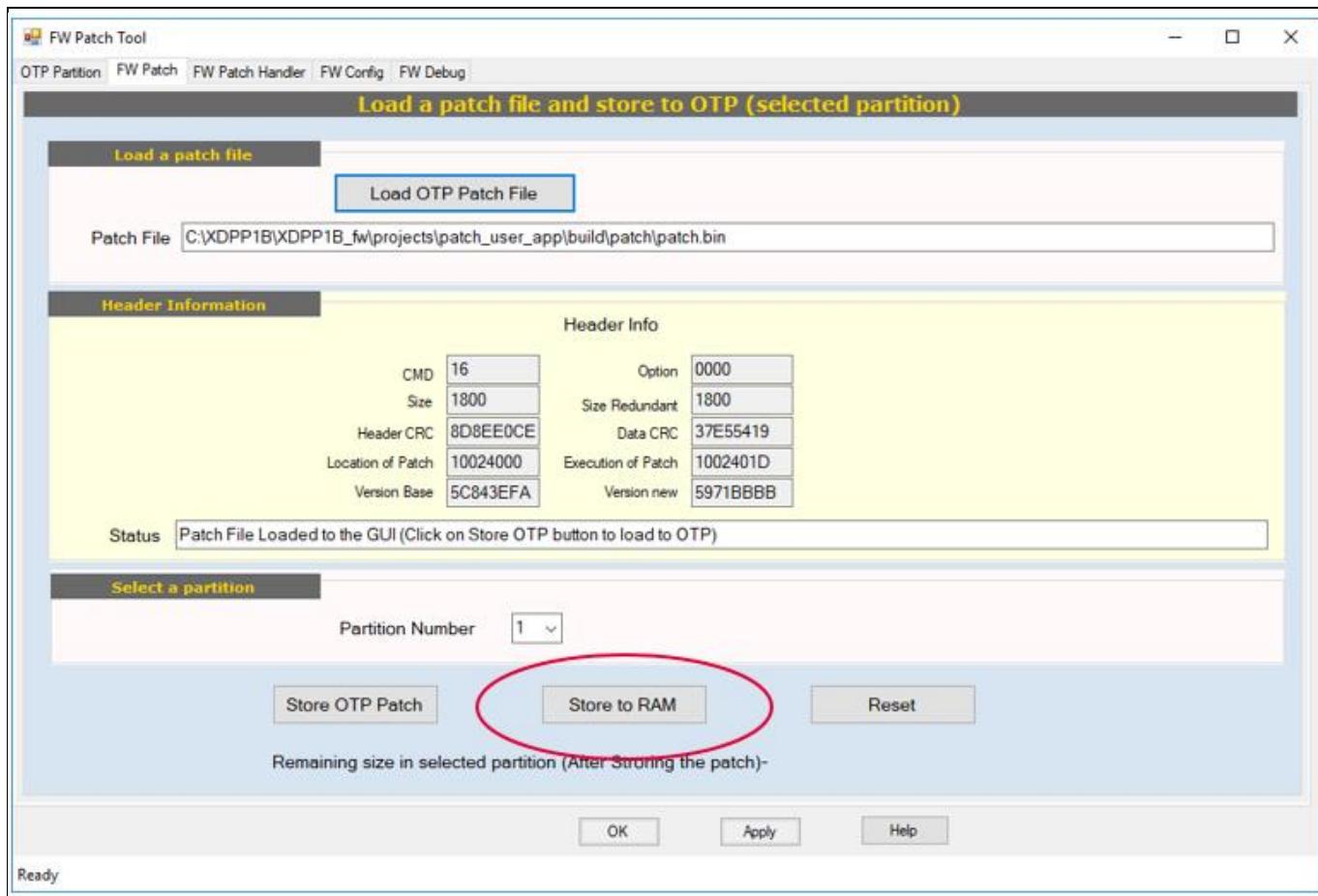
In order to run patch in RAM, user has to modify user_drv_init() at user_app.c with the following codes as shown in the following figure:

```
\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
#ifndef 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 function in
    /*lint -e522 */
    add_on_features_init();
    /////////////////////debug using ramp data circular buffer example/////////////////
    //

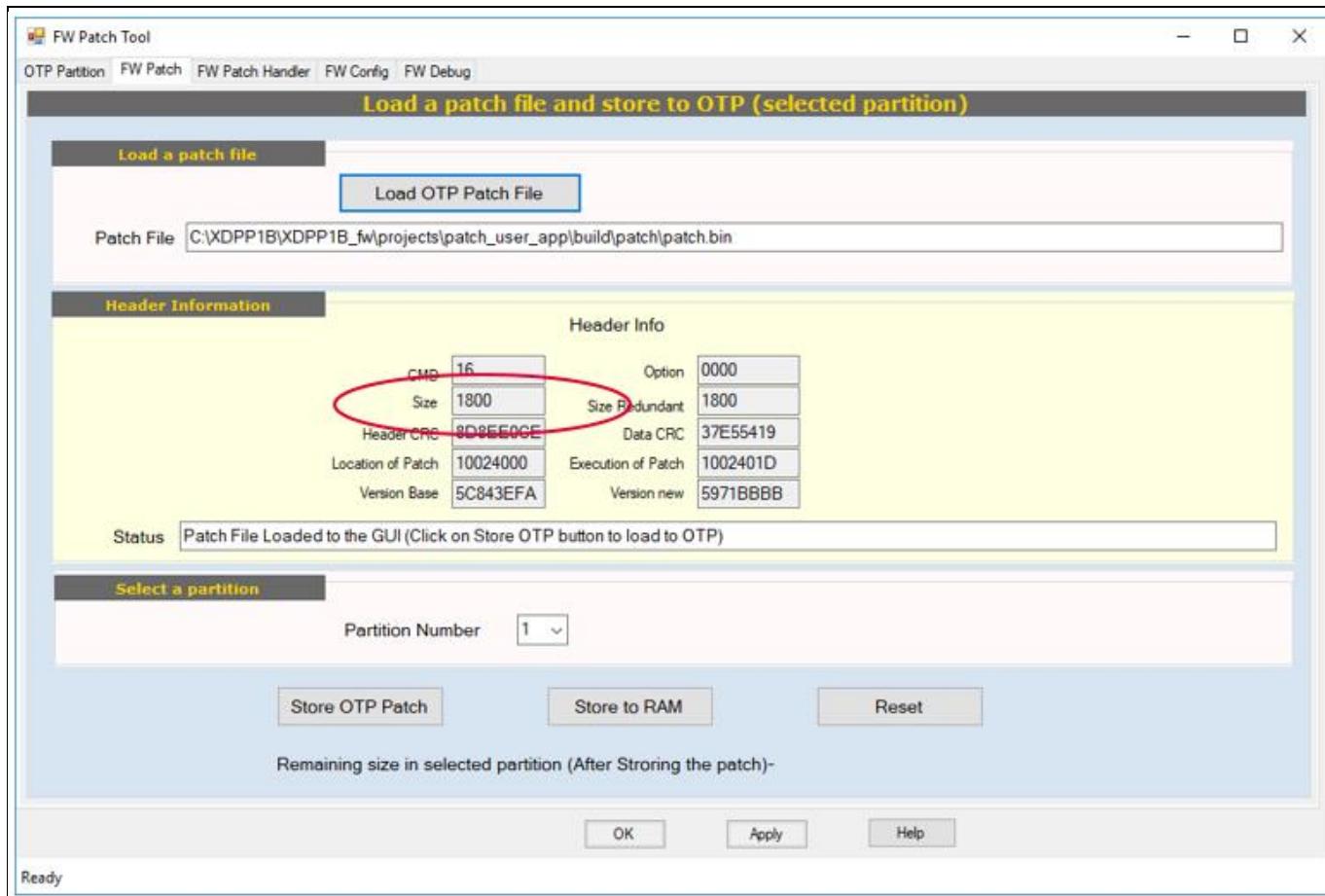
    regulation_sm_callbacks_init();
    pmbus_mfr_autogen_init();

    // 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:



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.

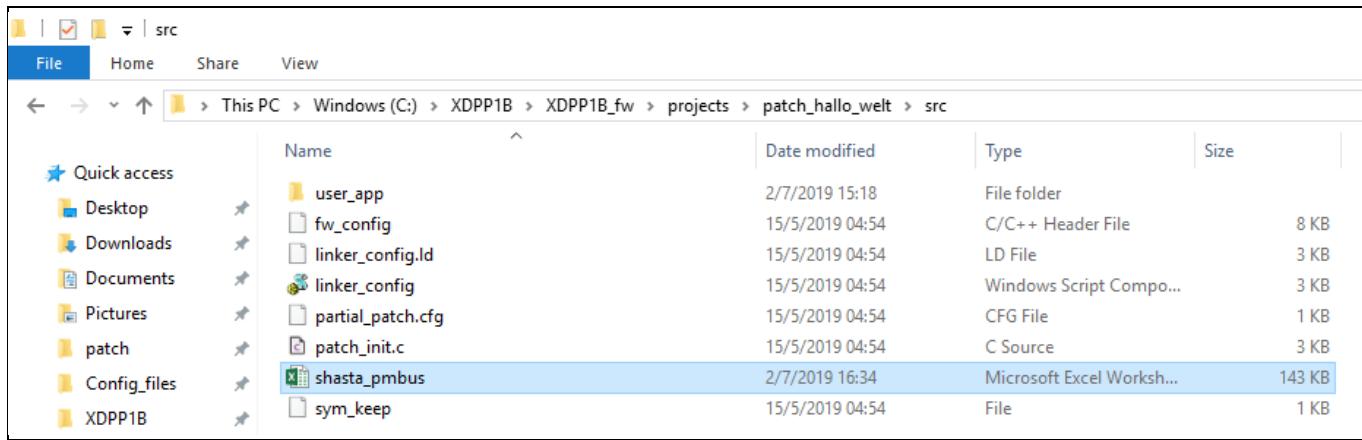


10 Implementing Custom PMBus Command in XDPP1100

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”**



Name	Date modified	Type	Size
user_app	2/7/2019 15:18	File folder	
fw_config	15/5/2019 04:54	C/C++ Header File	8 KB
linker_config.ld	15/5/2019 04:54	LD File	3 KB
linker_config	15/5/2019 04:54	Windows Script Compo...	3 KB
partial_patch.cfg	15/5/2019 04:54	CFG File	1 KB
patch_init.c	15/5/2019 04:54	C Source	3 KB
shasta_pmbus	2/7/2019 16:34	Microsoft Excel Worksh...	143 KB
sym_keep	15/5/2019 04:54	File	1 KB

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

shasta.pmbus - Excel

Halim Andriyanto (IFAP DC PMM SYS SI) Share

Line	OpCode	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			
																O	P	U
218	D8	MFR_TEMP_COMPENSATION	Block Write		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 tempi <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	ED	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	MFR_CONFIG_DEADTIME_ADJUSTMENT_THRESHOLD	Write Word	Read Word	2		y	\$9.2	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_VFP_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_APPC	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 rati 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
8:1 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 tempi
<end table>

<start_table_with_header>
Bit "Field Name" Meaning
15:8 length "The number of 32-bit words of the log's pe
15:8 length "The number of 32-bit words of the log's pe
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 corr
FMRGNCY "Unrecoverable error within the firmwar

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

shasta.pmbus - Excel

Halim Andriyanto (IFAP DC PMM SYS SI) Sort & Find & Filter Select

Line	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 stor	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
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
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	ED	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	EA	FW_CONFIG_DEADTIME_ADJUSTMENT_THRESHOLD	Write Word	Read Word	2		y	S9.2	0x0000	0x0000	16	[15:0]	n			Current threshold in uB 2 Amps format at above which PWM_DEADTIME
231	ES	MFR_SPECIFIC_E5				y	n						y			y
232	EE	MFR_VF_PPARAMS	Block Write	Block Read	4	y	n				32	y				y
233	EY	MFR_TEMP_COEFF			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			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			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 I
238	EC	MFR_ACTIVE_CLAMP	Read Word		2	y	n				16	y				n
239	ED	MFR_HALO_WELT	Write Byte	Read Byte	1	y	y	U8.0	0x00	0x00	8	[7:0]	y			
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

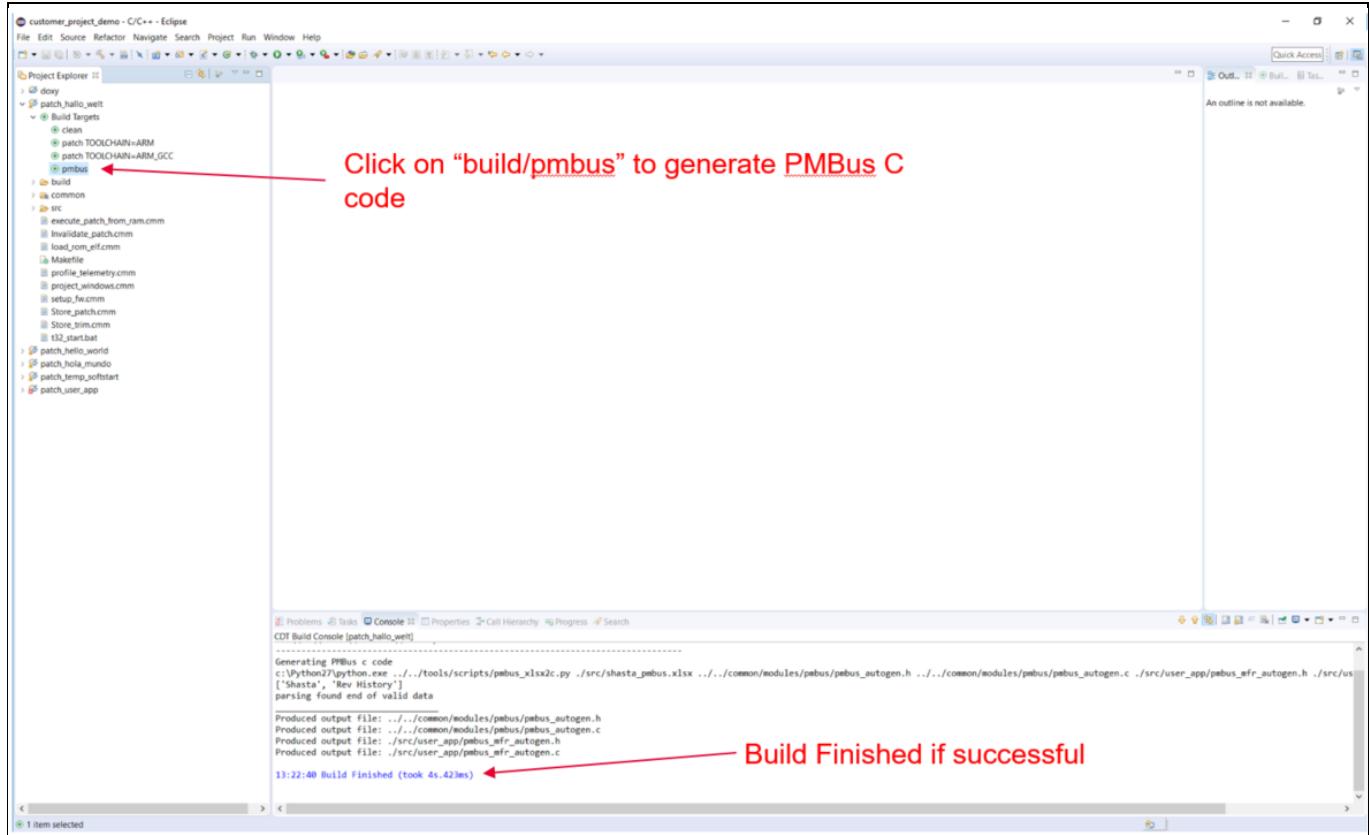
<start table_with_header>
Bit "Field Name" Meaning
4:3 Fault_Source,Select "0: tempa, 1: tempb, 2: tempc, 2:0 Read_Temperature, 1: Read_Temperature_2_source tempa, 3: tempb tempi, 4: tempi tempa, 5: tempi tempb
<end table>

<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 payload"
7:4 severity "<start table_with_header>
Severity "Log Severity" Desc
INFO "Logging in the normal program flow, not used for WARNING "Error which can be corrected/ignored" ERROR "Error within a single module, cannot be corrected" CRITICAL "Error within the firmware, cannot be recovered" FIRMWARE_FNCYC "Unrecoverable error within the firmware</table>">

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

customer_project_demo - C/C++ - patch_hallo_welt/src/user_app/pmbus_mfr_autogen.c - Eclipse

```

customer_project_demo - C/C++ - patch_hallo_welt/src/user_app/pmbus_mfr_autogen.c - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access

pmbus_mfr_autogen.h
106 /*
107 // instances of the struct one per command
108 extern PMBUS_CMD_t PMBUS_CMD_MFR_FW_CONFIG_UART;
109 extern uint8_t * PMBUS_CMD_MFR_FW_CONFIG_UART_DATA_LOOP0[8]; // data array for command loop0
110 extern uint8_t * PMBUS_CMD_MFR_FW_CONFIG_UART_DATA_LOOP1; // data array for command loop1 (pointer if common)
111
112 #define PMBUS_CMDCODE_MFR_SS_RAMP_FSW (0x05)
113
114 /* instantiate the command MFR_SS_RAMP_FSW.
115 */
116 extern PMBUS_CMD_t PMBUS_CMD_MFR_SS_RAMP_FSW;
117 extern uint8_t * PMBUS_CMD_MFR_SS_RAMP_FSW_DATA_LOOP0[1]; // data array for command loop0
118 extern uint8_t * PMBUS_CMD_MFR_SS_RAMP_FSW_DATA_LOOP1[1]; // data array for command loop1 (pointer if common)
119
120 #define PMBUS_CMDCODE_MFR_ISHARE_THRESHOLD (0xD)
121
122 /* instantiate the command structure for MFR_ISHARE_THRESHOLD.
123 */
124 instances of the struct one per command
125 extern PMBUS_CMD_t PMBUS_CMD_MFR_ISHARE_THRESHOLD;
126 extern uint8_t * PMBUS_CMD_MFR_ISHARE_THRESHOLD_DATA_LOOP0[2]; // data array for command loop0
127 extern uint8_t * PMBUS_CMD_MFR_ISHARE_THRESHOLD_DATA_LOOP1; // data array for command loop1 (pointer if common)
128
129 #define PMBUS_CMDCODE_MFR_HALLO_WELT (0xED)
130
131 /* instantiate the command MFR_HALLO_WELT.
132 */
133 extern PMBUS_CMD_t PMBUS_CMD_MFR_HALLO_WELT;
134 extern uint8_t * PMBUS_CMD_MFR_HALLO_WELT_DATA_LOOP0[1]; // data array for command loop0
135 extern uint8_t * PMBUS_CMD_MFR_HALLO_WELT_DATA_LOOP1[1]; // data array for command loop1 (pointer if common)
136 extern uint8_t * PMBUS_CMD_MFR_HALLO_WELT_RANGE[2]; // data array for range (common for loops)
137
138 #define PMBUS_CMDCODE_MFR_ADDED_DROP_DURING_RAMP (0xFc)
139
140 /* instantiate the command MFR_ADDED_DROP_DURING_RAMP.
141 */
142 extern PMBUS_CMD_t PMBUS_CMD_MFR_ADDED_DROP_DURING_RAMP;
143 extern uint8_t * PMBUS_CMD_MFR_ADDED_DROP_DURING_RAMP_DATA_LOOP0[2]; // data array for command loop0
144 extern uint8_t * PMBUS_CMD_MFR_ADDED_DROP_DURING_RAMP_DATA_LOOP1[2]; // data array for command loop1 (pointer if common)
145 extern uint8_t * PMBUS_CMD_MFR_ADDED_DROP_DURING_RAMP_RANGE[4]; // data array for range (common for loops)
146
147
148 void pmbus_mfr_autogen_init(void);
149
150 // external handle functions
151 extern void PMBUS_HANDLE_FAN_CONFIG_1_2(PMBUS_PAGE_t page, PMBUS_DIRECTION_e direction);
152 extern void PMBUS_HANDLE_FAN_COMMAND_1(PMBUS_PAGE_t page, PMBUS_DIRECTION_e direction);
153 extern void PMBUS_HANDLE_FAN_COMMAND_2(PMBUS_PAGE_t page, PMBUS_DIRECTION_e direction);
154 extern void PMBUS_HANDLE_MFR_FREQUENCY_DITHER(PMBUS_PAGE_t page, PMBUS_DIRECTION_e direction);
155 extern void PMBUS_HANDLE_MFR_BOARD_TRIM(PMBUS_PAGE_t page, PMBUS_DIRECTION_e direction);
156 extern void PMBUS_HANDLE_MFR_VIN_SCALE(PMBUS_PAGE_t page, PMBUS_DIRECTION_e direction);
157 extern void PMBUS_HANDLE_MFR_VDD_SCALE(PMBUS_PAGE_t page, PMBUS_DIRECTION_e direction);
158 extern void PMBUS_HANDLE_MFR_VIN_SCALE(PMBUS_PAGE_t page, PMBUS_DIRECTION_e direction);
159 extern void PMBUS_HANDLE_MFR_FW_CONFIG_UART(PMBUS_PAGE_t page, PMBUS_DIRECTION_e direction);
160 extern void PMBUS_HANDLE_MFR_ISHARE_THRESHOLD(PMBUS_PAGE_t page, PMBUS_DIRECTION_e direction);
161 extern void PMBUS_HANDLE_MFR_HALLO_WELT(PMBUS_PAGE_t page, PMBUS_DIRECTION_e direction);
162 extern void PMBUS_HANDLE_MFR_ADDED_DROP_DURING_RAMP(PMBUS_PAGE_t page, PMBUS_DIRECTION_e direction);
163
164 #endif //endif /* _PMBUS_MFR_AUTOPEN_H_ */
165
166
167

```

pmbus_mfr_autogen.c

```

93 PMBUS_CMD_MFR_SS_RAMP_FSW_LTC_PFL_TW_LIGHT_FL_UART;
94 uint8_t * PMBUS_CMD_MFR_SS_RAMP_FSW_CONFIG_UART_DATA_LOOP0[8]; // data array for command loop0
95 /*list -esym(552,PMBUS_CMD_MFR_FW_CONFIG_UART_DATA_LOOP0) */
96 /*list -esym(844,PMBUS_CMD_MFR_FW_CONFIG_UART_DATA_LOOP1) */
97 uint8_t * PMBUS_CMD_MFR_FW_CONFIG_UART_DATA_LOOP1; // data array for command loop1 (pointer if common)
98
99 /* instantiate the command MFR_SS_RAMP_FSW.
100 */
101 PMBUS_CMD_t PMBUS_CMD_MFR_SS_RAMP_FSW;
102 uint8_t * PMBUS_CMD_MFR_SS_RAMP_FSW_DATA_LOOP0[1]; // data array for command loop0
103 uint8_t * PMBUS_CMD_MFR_SS_RAMP_FSW_DATA_LOOP1[1]; // data array for command loop1 (pointer if common)
104
105 /* instantiate the command structure for MFR_ISHARE_THRESHOLD.
106 */
107 /* instances of the struct one per command
108 PMBUS_CMD_t PMBUS_CMD_MFR_ISHARE_THRESHOLD;
109 uint8_t * PMBUS_CMD_MFR_ISHARE_THRESHOLD_DATA_LOOP0[1]; // data array for command loop0
110 /*list -esym(552,PMBUS_CMD_MFR_ISHARE_THRESHOLD_DATA_LOOP0) */
111 /*list -esym(844,PMBUS_CMD_MFR_ISHARE_THRESHOLD_DATA_LOOP1) */
112 uint8_t * PMBUS_CMD_MFR_ISHARE_THRESHOLD_DATA_LOOP1; // data array for command loop1 (pointer if common)
113
114 /* instantiate the command MFR_HALLO_WELT.
115 */
116 PMBUS_CMD_t PMBUS_CMD_MFR_HALLO_WELT;
117 uint8_t * PMBUS_CMD_MFR_HALLO_WELT_DATA_LOOP0[1]; // data array for command loop0
118 uint8_t * PMBUS_CMD_MFR_HALLO_WELT_DATA_LOOP1[1]; // data array for command loop1 (pointer if common)
119 uint8_t * PMBUS_CMD_MFR_HALLO_WELT_RANGE[2]; // data array for range (common for loops)
120
121 /* instantiate the command MFR_ADDED_DROP_DURING_RAMP.
122 */
123 PMBUS_CMD_t PMBUS_CMD_MFR_ADDED_DROP_DURING_RAMP;
124 uint8_t * PMBUS_CMD_MFR_ADDED_DROP_DURING_RAMP_DATA_LOOP0[2]; // data array for command loop0
125 uint8_t * PMBUS_CMD_MFR_ADDED_DROP_DURING_RAMP_DATA_LOOP1[2]; // data array for command loop1 (pointer if common)
126 uint8_t * PMBUS_CMD_MFR_ADDED_DROP_DURING_RAMP_RANGE[4]; // data array for range (common for loops)
127
128 /*
129 * Initializes all of our pmbus structs that were generated by the script
130 * Function gets called to setup command support array, initialize pointers to command data, and describe commands
131 */
132 void pmbus_mfr_autogen_init(void)
133 {
134
135     /* initialize command structure for FAN_CONFIG_1_2. */
136     // FAN_CONFIG_1_2 COMMAND
137     ptr_pmbus_callback(PMBUS_CMDCODE_FAN_CONFIG_1_2) = PMBUS_HANDLE_FAN_CONFIG_1_2;
138     //PMBUS_CMD_FAN_CONFIG_1_2.OPCODE = PMBUS_CMDCODE_FAN_CONFIG_1_2;
139     //PMBUS_CMD_FAN_CONFIG_1_2.WRITE_PROTOCOL = (uint8_t) TRANSACTION_PROTOCOL_WRITE;
140     //PMBUS_CMD_FAN_CONFIG_1_2.READ_PROTOCOL = (uint8_t) TRANSACTION_PROTOCOL_READ;
141     //PMBUS_CMD_FAN_CONFIG_1_2.NUMPBYS = 1; // data only not including slave address and command
142     //PMBUS_CMD_FAN_CONFIG_1_2.NUMPBUDBYTES = 2; // slave address command and any other non-data bytes
143     //PMBUS_CMD_FAN_CONFIG_1_2.PROTECTEDED = 0;
144     //PMBUS_CMD_FAN_CONFIG_1_2.PROTECTEDED = 0;
145     //PMBUS_CMD_FAN_CONFIG_1_2.NUMPBUDBYTES = 0;
146     //PMBUS_CMD_FAN_CONFIG_1_2.RANGE_TYPE = 0;
147     PMBUS_CMD_FAN_CONFIG_1_2.RANGE_TYPE = 0;
148     PMBUS_CMD_FAN_CONFIG_1_2.CMD_CONFIG = 0x181231;
149     PMBUS_CMD_FAN_CONFIG_1_2.DATA0 = &PMBUS_CMD_FAN_CONFIG_1_2_DATA_LOOP0[0]; // set the pointer to the data array
150     PMBUS_CMD_FAN_CONFIG_1_2.DATA1 = &PMBUS_CMD_FAN_CONFIG_1_2_DATA_LOOP0[0]; // set the pointer to the data array
151     PMBUS_CMD_FAN_CONFIG_1_2.DATA2 = &PMBUS_CMD_FAN_CONFIG_1_2_DATA_LOOP0[0];
152     PMBUS_CMD_FAN_CONFIG_1_2.NUMPBUDBYTES = 0;
153     // now populate pointers to the supported commands for each page
154     PMBUS_CMD_ARRAY_LOOP[PMBUS_CMDCODE_FAN_CONFIG_1_2] = &PMBUS_CMD_FAN_CONFIG_1_2;

```



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

customer_project_demo - C/C++ - patch_hallo_welt/src/user_app/pmbus_mfr autogenerated - Eclipse

File Edit Source Refactor Search Project Run Window Help

Quick Access

```
/* pmbus_mfr autogenerated */  
/* pmbus_mfr_specific_handlers.c */  
/* user_app.c */  
/* pmbus_mfr autogenerated */  
  
188 PMBUS_CMD_MFR_ISHARE_THRESHOLD_DATA[0] = &PMBUS_CMD_MFR_ISHARE_THRESHOLD_DATA[0]; // set the pointer to the data array  
189 PMBUS_CMD_MFR_ISHARE_THRESHOLD_DATA[1] = &PMBUS_CMD_MFR_ISHARE_THRESHOLD_DATA[0]; // set the pointer to the data array  
190 PMBUS_CMD_MFR_ISHARE_THRESHOLD_DATA[2] = &PMBUS_CMD_MFR_ISHARE_THRESHOLD_DATA[0]; // set the pointer to the data array  
191 // RAM Support for MFR_ISHARE_THRESHOLD_COMPARE  
192 // now populate pointers to the supported commands for each page  
193 PMBUS_CMD_ARRAY_LOOP(PMBUS_CMDCODE_MFR_ISHARE_THRESHOLD) = &PMBUS_CMD_MFR_ISHARE_THRESHOLD;  
194 //PMBUS_CMD_ARRAY_LOOP(PMBUS_CMDCODE_MFR_ISHARE_THRESHOLD) = &PMBUS_CMD_MFR_ISHARE_THRESHOLD;  
195 //lint (-770) suppress "Constant expression evaluates to 0 in operation"  
196 //PMBUS_CMD_SUPPORT_ARRAY_LOOP(PMBUS_CMDCODE_MFR_ISHARE_THRESHOLD)[0] |= (uint32_t){(uint32_t)1lu << ((uint32_t)PMBUS_CMDCODE_MFR_ISHARE_THRESHOLD&31)};  
197 //PMBUS_CMD_SUPPORT_ARRAY_LOOP(PMBUS_CMDCODE_MFR_ISHARE_THRESHOLD)[1] |= (uint32_t){(uint32_t)1lu << ((uint32_t)PMBUS_CMDCODE_MFR_ISHARE_THRESHOLD&31)};  
198  
199 /* initialize command structure for MFR_HALLO_WELT */  
200 //MFR_HALLO_WELT.CMD_CODE = PMBUS_CMDCODE_MFR_HALLO_WELT; //CMD_HANDLE_PMR_HALLO_WELT;  
201 //PMBUS_CMD_MFR_HALLO_WELT.CODE = PMBUS_CMDCODE_MFR_HALLO_WELT;  
202 //PMBUS_CMD_MFR_HALLO_WELT.WRITE_PROTOCOL = (uint8_t)_TRANSACTION_PROTOCOL_WRITE;  
203 //PMBUS_CMD_MFR_HALLO_WELT.READ_PROTOCOL = (uint8_t)_TRANSACTION_PROTOCOL_READ;  
204 //PMBUS_CMD_MFR_HALLO_WELT.HALU_BYTES = 1; // data only not including slave address and command  
205 //PMBUS_CMD_MFR_HALLO_WELT.HALU_PROTOCOL_BYTES = 2; // slave address command and any other non-data bytes  
206 //PMBUS_CMD_MFR_HALLO_WELT.SUPPORTED = 1;  
207 //PMBUS_CMD_MFR_HALLO_WELT.PROTECTED = 0;  
208 //PMBUS_CMD_MFR_HALLO_WELT.PROTECTED = 0;  
209 //PMBUS_CMD_MFR_HALLO_WELT.RANGE_TYPE = (uint8_t) PMBUS_RANGE_LINEAR1_UNSIGNED;  
210 PMBUS_CMD_MFR_HALLO_WELT.RANGE[0] = 0x00000000; //PMBUS_CMD_MFR_HALLO_WELT_RANGE[0];  
211 PMBUS_CMD_MFR_HALLO_WELT.RANGE[1] = 0;  
212 //PMBUS_CMD_MFR_HALLO_WELT.CH0_CONFIG = 0x0418123d;  
213 PMBUS_CMD_MFR_HALLO_WELT.DATA[0] = &PMBUS_CMD_MFR_HALLO_WELT_DATA[0]; // set the pointer to the data array  
214 PMBUS_CMD_MFR_HALLO_WELT.DATA[1] = &PMBUS_CMD_MFR_HALLO_WELT_DATA[0]; // set the pointer to the data array  
215 // now populate pointers to the supported commands for each page  
216 PMBUS_CMD_ARRAY_LOOP(PMBUS_CMDCODE_MFR_HALLO_WELT) = &PMBUS_CMD_MFR_HALLO_WELT;  
217 //PMBUS_CMD_ARRAY_LOOP(PMBUS_CMDCODE_MFR_HALLO_WELT) = &PMBUS_CMD_MFR_HALLO_WELT;  
218 //lint (-770) suppress "Constant expression evaluates to 0 in operation"  
219 //PMBUS_CMD_SUPPORT_ARRAY_LOOP(PMBUS_CMDCODE_MFR_HALLO_WELT)[0] |= (uint32_t){(uint32_t)1lu << ((uint32_t)PMBUS_CMDCODE_MFR_HALLO_WELT&31)};  
220 //PMBUS_CMD_SUPPORT_ARRAY_LOOP(PMBUS_CMDCODE_MFR_HALLO_WELT)[1] |= (uint32_t){(uint32_t)1lu << ((uint32_t)PMBUS_CMDCODE_MFR_HALLO_WELT&31)};  
221  
222 /* initialize command structure for MFR_ADDED_DROP_DURING_RAMP */  
223 //MFR_ADDED_DROP_DURING_RAMP.COMMAND = PMBUS_MFR_ADDED_DROP_DURING_RAMP; //PMBS_MFR_HANDLE_PMR_ADDDROP_DURING_RAMP;  
224 //PMBUS_CMD_MFR_ADDED_DROP_DURING_RAMP.CODE = PMBUS_CMDCODE_PMR_ADDDROP_DURING_RAMP;  
225 //PMBUS_CMD_MFR_ADDED_DROP_DURING_RAMP.WRITE_PROTOCOL = (uint8_t)_TRANSACTION_PROTOCOL_WRITE;  
226 //PMBUS_CMD_MFR_ADDED_DROP_DURING_RAMP.READ_PROTOCOL = (uint8_t)_TRANSACTION_PROTOCOL_READ;  
227 //PMBUS_CMD_MFR_ADDED_DROP_DURING_RAMP.HALU_BYTES = 2; // data only not including slave address and command  
228 //PMBUS_CMD_MFR_ADDED_DROP_DURING_RAMP.HALU_PROTOCOL_BYTES = 2; // slave address command and any other non-data bytes  
229 //PMBUS_CMD_MFR_ADDED_DROP_DURING_RAMP.SUPPORTED = 1;  
230 //PMBUS_CMD_MFR_ADDED_DROP_DURING_RAMP.PROTECTED = 0;  
231 //PMBUS_CMD_MFR_ADDED_DROP_DURING_RAMP.PROTECTED = 0;  
232 //PMBUS_CMD_MFR_ADDED_DROP_DURING_RAMP.RANGE_TYPE = (uint8_t) PMBUS_RANGE_LINEAR1_UNSIGNED_ARRAY;  
233 PMBUS_CMD_MFR_ADDED_DROP_DURING_RAMP.RANGE[0] = 0;  
234 PMBUS_CMD_MFR_ADDED_DROP_DURING_RAMP.RANGE[1] = 0;  
235 PMBUS_CMD_MFR_ADDED_DROP_DURING_RAMP.RANGE[2] = 0;  
236 PMBUS_CMD_MFR_ADDED_DROP_DURING_RAMP.RANGE[3] = 0;  
237 //PMBUS_CMD_MFR_ADDED_DROP_DURING_RAMP.CH0_CONFIG = 0x58223Fc;  
238 PMBUS_CMD_MFR_ADDED_DROP_DURING_RAMP.DATA[0] = &PMBUS_CMD_MFR_ADDED_DROP_DURING_RAMP_DATA[0]; // set the pointer to the data array  
239 PMBUS_CMD_MFR_ADDED_DROP_DURING_RAMP.DATA[1] = &PMBUS_CMD_MFR_ADDED_DROP_DURING_RAMP_DATA[0]; // set the pointer to the data array  
240 // now populate pointers to the supported commands for each page  
241 PMBUS_CMD_ARRAY_LOOP(PMBUS_CMDCODE_MFR_ADDED_DROP_DURING_RAMP) = &PMBUS_CMD_MFR_ADDED_DROP_DURING_RAMP;  
242 //PMBUS_CMD_ARRAY_LOOP(PMBUS_CMDCODE_MFR_ADDED_DROP_DURING_RAMP) = &PMBUS_CMD_MFR_ADDED_DROP_DURING_RAMP;  
243 //lint (-770) suppress "Constant expression evaluates to 0 in operation"  
244 //PMBUS_CMD_SUPPORT_ARRAY_LOOP(PMBUS_CMDCODE_MFR_ADDED_DROP_DURING_RAMP)[0] |= (uint32_t){(uint32_t)1lu << ((uint32_t)PMBUS_CMDCODE_MFR_ADDED_DROP_DURING_RAMP&31)};  
245 //PMBUS_CMD_SUPPORT_ARRAY_LOOP(PMBUS_CMDCODE_MFR_ADDED_DROP_DURING_RAMP)[1] |= (uint32_t){(uint32_t)1lu << ((uint32_t)PMBUS_CMDCODE_MFR_ADDED_DROP_DURING_RAMP&31)};
```

These are the generated C code after specifying the new PMBus MFR command in the spreadsheet

These are the generated C code after specifying the new PMBus MFR command in the spreadsheet

10.3 Implement PMBus MFR command handler

10.3.1 STEP 1: Note the PMBus Command Handler name

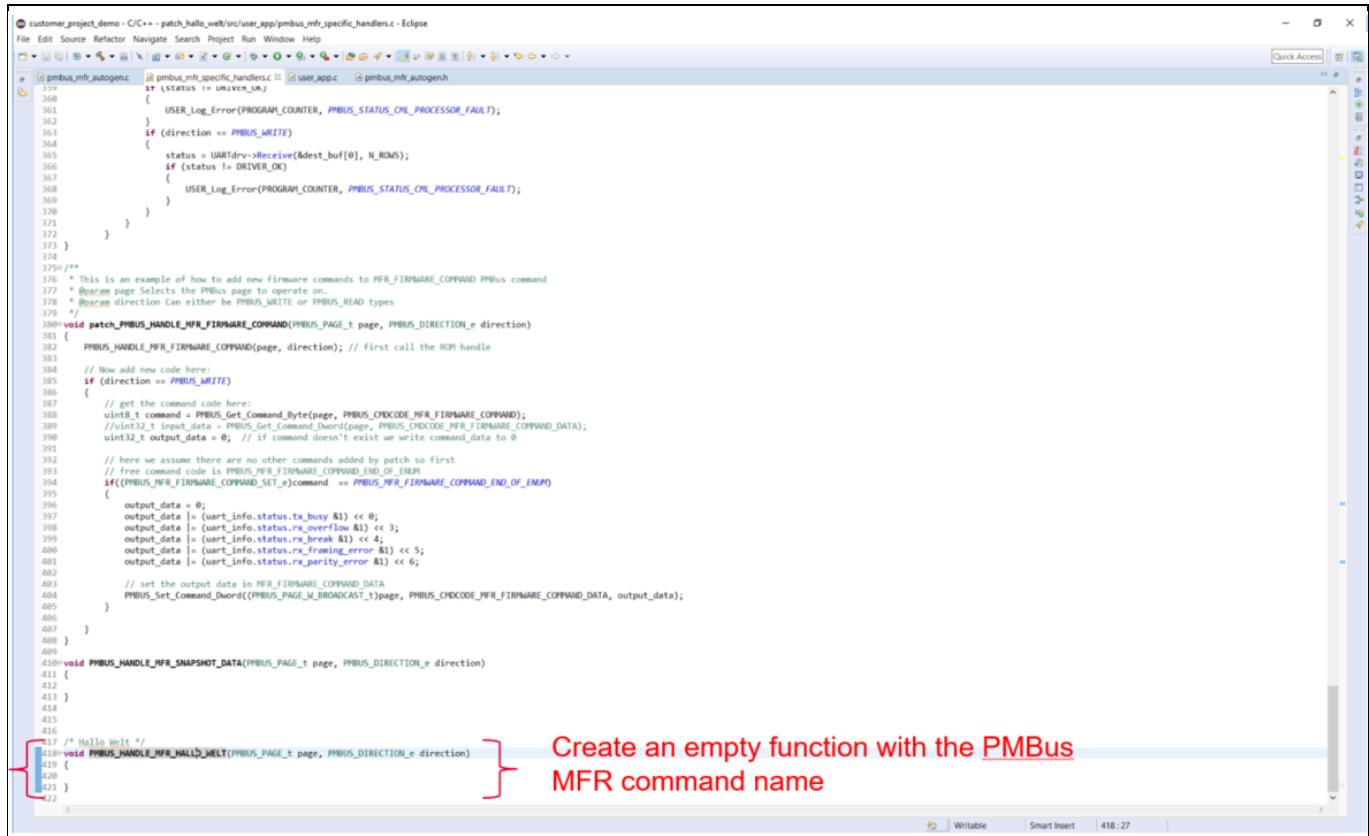
Take note of the generated PMBus MFR command handler name

```

customer_project_demo - C/C++ - patch_hallo_welt/src/user_app/pmbus_mfr_autogen.c - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
customer_project_demo.c [ ] pmbus_mfr_specific_handlers.c [ ] user_applc.c [ ] pmbus_adugen.h [ ] pmbus_mfr.adugen.h
customer_project_demo.c [ ] pmbus_mfr_specific_handlers.c [ ] user_applc.c [ ] pmbus_adugen.h [ ] pmbus_mfr.adugen.h
168 PMBUS_CMD_MFR_ISHARE_THRESHOLD.DATARR = &PMBUS_CMD_MFR_ISHARE_THRESHOLD.DATA_LOOP[0]; // set the pointer to the data array
169 PMBUS_CMD_MFR_ISHARE_THRESHOLD.DATARR = &PMBUS_CMD_MFR_ISHARE_THRESHOLD.DATA_LOOP[0]; // set the pointer to the data array
170 PMBUS_CMD_MFR_ISHARE_THRESHOLD.DATARR = &PMBUS_CMD_MFR_ISHARE_THRESHOLD.DATA_LOOP[0]; // set the pointer to the data array
171 // RANGE SUPPORT for MFR_ISHARE_THRESHOLD:
172 // now populate pointers to the supported commands for each page
173 PMBUS_CMD_ARRAY_LOOP(PMBUS_CMDCODE_MFR_ISHARE_THRESHOLD) = &PMBUS_CMD_MFR_ISHARE_THRESHOLD;
174 //lint -e778 suppress "Constant expression evaluates to 0 in operation '&'"
175 //lint -e778 suppress "Constant expression evaluates to 0 in operation '&'"
176 //PMBUS_CMD_SUPPORT_ARRAY_LOOP(PMBUS_CMDCODE_MFR_ISHARE_THRESHOLD/32) |= (uint32_t){(uint32_t)1u << ((uint32_t)PMBUS_CMDCODE_MFR_ISHARE_THRESHOLD&31u)};
177 //PMBUS_CMD_SUPPORT_ARRAY_LOOP(PMBUS_CMDCODE_MFR_ISHARE_THRESHOLD/32) |= (uint32_t){(uint32_t)1u << ((uint32_t)PMBUS_CMDCODE_MFR_ISHARE_THRESHOLD&31u)};
178
179 // Initialize command structure for MFR_HALLO_WELT:
180 // MFR_HALLO_WELT.COPPROD:
181 ptr_pbus_callback[PMBUS_CMDCODE_MFR_HALLO_WELT] = PMBUS_HANDLE_MFR_HALLO_WELT;
182 //PMBUS_CMD_MFR_HALLO_WELT.CODE = PMBUS_CMDCODE_MFR_HALLO_WELT;
183 //PMBUS_CMD_MFR_HALLO_WELT.WRITE_PROTOCOL = (uint8_t) TRANSACTION_PROTOCOL_WRITE;
184 //PMBUS_CMD_MFR_HALLO_WELT.READ_PROTOCOL = (uint8_t) TRANSACTION_PROTOCOL_READ;
185 //PMBUS_CMD_MFR_HALLO_WELT.MPL_BYTES = 1; // data only not including slave address and command
186 //PMBUS_CMD_MFR_HALLO_WELT.MPU_PROTOCOL_BYTES = 2; // slave address command and any other non-data bytes
187 //PMBUS_CMD_MFR_HALLO_WELT.SUPPORTED = 1;
188 //PMBUS_CMD_MFR_HALLO_WELT.PROTECTED = 0;
189 //PMBUS_CMD_MFR_HALLO_WELT.RANGE_TYPE = 0;
190 //PMBUS_CMD_MFR_HALLO_WELT.RANGE_TYPE = (uint8_t) PMBUS_RANGE_LINEAR11_UNSIGNED;
191 PMBUS_CMD_MFR_HALLO_WELT.RANGE = &PMBUS_CMD_MFR_HALLO_WELT_RANGE[0];
192 PMBUS_CMD_MFR_HALLO_WELT_RANGE[0] = 8;
193 PMBUS_CMD_MFR_HALLO_WELT_RANGE[1] = 0;
194 PMBUS_CMD_MFR_HALLO_WELT_RANGE[2] = 0x43B123d;
195 PMBUS_CMD_MFR_HALLO_WELT.DATARR = &PMBUS_CMD_MFR_HALLO_WELT_DATA_LOOP[0]; // set the pointer to the data array
196 PMBUS_CMD_MFR_HALLO_WELT.DATARR = &PMBUS_CMD_MFR_HALLO_WELT_DATA_LOOP[0]; // set the pointer to the data array
197 // RANGE SUPPORT for MFR_HALLO_WELT COMMAND:
198 // now populate pointers to the supported commands for each page
199 PMBUS_CMD_ARRAY_LOOP(PMBUS_CMDCODE_MFR_HALLO_WELT) = &PMBUS_CMD_MFR_HALLO_WELT;
200 //lint -e778 suppress "Constant expression evaluates to 0 in operation '&'"
201 //lint -e778 suppress "Constant expression evaluates to 0 in operation '&'"
202 //PMBUS_CMD_SUPPORT_ARRAY_LOOP(PMBUS_CMDCODE_MFR_HALLO_WELT/32) |= (uint32_t){(uint32_t)1u << ((uint32_t)PMBUS_CMDCODE_MFR_HALLO_WELT&31u)};
203 //PMBUS_CMD_SUPPORT_ARRAY_LOOP(PMBUS_CMDCODE_MFR_HALLO_WELT/32) |= (uint32_t){(uint32_t)1u << ((uint32_t)PMBUS_CMDCODE_MFR_HALLO_WELT&31u)};
204
205 // Initialize command structure for MFR_ADDED_DROOP_DURING_RAMP:
206 // MFR_ADDED_DROOP_DURING_RAMP.COMMAND:
207 ptr_pbus_callback[PMBUS_CMDCODE_MFR_ADDED_DROOP_DURING_RAMP] = PMBUS_HANDLE_MFR_ADDED_DROOP_DURING_RAMP;
208 //PMBUS_CMD_MFR_ADDED_DROOP_DURING_RAMP.CODE = PMBUS_CMDCODE_MFR_ADDED_DROOP_DURING_RAMP;
209 //PMBUS_CMD_MFR_ADDED_DROOP_DURING_RAMP.WRITE_PROTOCOL = (uint8_t) TRANSACTION_PROTOCOL_WRITE;
210 //PMBUS_CMD_MFR_ADDED_DROOP_DURING_RAMP.READ_PROTOCOL = (uint8_t) TRANSACTION_PROTOCOL_READ;
211 //PMBUS_CMD_MFR_ADDED_DROOP_DURING_RAMP.MPL_BYTES = 2; // data only not including slave address and command
212 //PMBUS_CMD_MFR_ADDED_DROOP_DURING_RAMP.MPU_PROTOCOL_BYTES = 2; // slave address command and any other non-data bytes
213 //PMBUS_CMD_MFR_ADDED_DROOP_DURING_RAMP.SUPPORTED = 1;
214 //PMBUS_CMD_MFR_ADDED_DROOP_DURING_RAMP.PROTECTED = 0;
215 //PMBUS_CMD_MFR_ADDED_DROOP_DURING_RAMP.RANGE_TYPE = 0;
216 //PMBUS_CMD_MFR_ADDED_DROOP_DURING_RAMP.RANGE_TYPE = (uint8_t) PMBUS_RANGE_LINEAR11_UNSIGNED;
217 PMBUS_CMD_MFR_ADDED_DROOP_DURING_RAMP.RANGE = &PMBUS_CMD_MFR_ADDED_DROOP_DURING_RAMP_RANGE[0];
218 PMBUS_CMD_MFR_ADDED_DROOP_DURING_RAMP_RANGE[0] = 3;
219 PMBUS_CMD_MFR_ADDED_DROOP_DURING_RAMP_RANGE[1] = 75;
220 PMBUS_CMD_MFR_ADDED_DROOP_DURING_RAMP_RANGE[2] = 0;
221 PMBUS_CMD_MFR_ADDED_DROOP_DURING_RAMP_RANGE[3] = 61;
222 PMBUS_CMD_MFR_ADDED_DROOP_DURING_RAMP.CODE = 0xc58223fc;
223 PMBUS_CMD_MFR_ADDED_DROOP_DURING_RAMP.DATARR = &PMBUS_CMD_MFR_ADDED_DROOP_DURING_RAMP_DATA_LOOP[0]; // set the pointer to the data array
224 PMBUS_CMD_MFR_ADDED_DROOP_DURING_RAMP.DATARR = &PMBUS_CMD_MFR_ADDED_DROOP_DURING_RAMP_DATA_LOOP[0]; // set the pointer to the data array
225 // now populate pointers to the supported commands for each page
226 PMBUS_CMD_ARRAY_LOOP(PMBUS_CMDCODE_MFR_ADDED_DROOP_DURING_RAMP) = &PMBUS_CMD_MFR_ADDED_DROOP_DURING_RAMP;
227 //PMBUS_CMD_ARRAY_LOOP(PMBUS_CMDCODE_MFR_ADDED_DROOP_DURING_RAMP) = &PMBUS_CMD_MFR_ADDED_DROOP_DURING_RAMP;
228 //lint -e778 suppress "Constant expression evaluates to 0 in operation '&'"
229 //lint -e778 suppress "Constant expression evaluates to 0 in operation '&'"
230 //PMBUS_CMD_SUPPORT_ARRAY_LOOP(PMBUS_CMDCODE_MFR_ADDED_DROOP_DURING_RAMP/32) |= (uint32_t){(uint32_t)1u << ((uint32_t)PMBUS_CMDCODE_MFR_ADDED_DROOP_DURING_RAMP&31u)};
231 //PMBUS_CMD_SUPPORT_ARRAY_LOOP(PMBUS_CMDCODE_MFR_ADDED_DROOP_DURING_RAMP/32) |= (uint32_t){(uint32_t)1u << ((uint32_t)PMBUS_CMDCODE_MFR_ADDED_DROOP_DURING_RAMP&31u)};
232

```

10.3.2 STEP 2: Modify “pmbus_mfr_specific_handler.c”



```

customer_project_demo - C/C++ - patch_hallo_welt/src/user_app/pmbus_mfr_specific_handlers.c - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
pmbus_mfr_autogen.c pmbus_mfr_specific_handlers.c user_app.c pmbus_mfr_autogen.h
359     if ((status != UNLIVEN_USB))
360     {
361         USER_Log_Error(PROGRAM_COUNTER, PMBUS_STATUS_CHI_PROCESSOR_FAULT);
362     }
363     if (direction == PMBUS_WRITE)
364     {
365         status = UARTdrv->Receive(&dest_buf[0], N_ROWS);
366         if (status != DRIVER_OK)
367         {
368             USER_Log_Error(PROGRAM_COUNTER, PMBUS_STATUS_CHI_PROCESSOR_FAULT);
369         }
370     }
371 }
372 }
373 }
374 */
375 /* This is an example of how to add new firmware commands to MFR_FIRMWARE_COMMAND PMBus command
376 * @param page Selects the PMbus page to operate on.
377 * @param direction Can either be PMBUS_WRITE or PMBUS_READ types
378 */
379 void patch_PMBUS_HANDLE_MFR_FIRMWARE_COMMAND(PMBUS_PAGE_t page, PMBUS_DIRECTION_e direction)
380 {
381     PMBUS_HANDLE_MFR_FIRMWARE_COMMAND(page, direction); // first call the ROM handle
382
383     // Now add new code here:
384     if (direction == PMBUS_WRITE)
385     {
386         // get the command code here:
387         uint8_t command = PMBUS_Get_Command_Byt
388         //uint32_t input_data = PMBUS_Get_Command_Dword(page, PMBUS_CMDCODE_MFR_FIRMWARE_COMMAND);
389         uint32_t output_data = 0; // If command doesn't exist we leave command_data to 0
390
391         // here we assume there are no other commands added by patch so first
392         // free command code is PMBUS_MFR_FIRMWARE_COMMAND_END_OF_ENUM
393         if (PMBUS_MFR_FIRMWARE_COMMAND_Set_e)command == PMBUS_MFR_FIRMWARE_COMMAND_END_OF_ENUM)
394         {
395             output_data = 0;
396             output_data |= (uart_info.status.tx_busy &1) << 0;
397             output_data |= (uart_info.status.rx_overflow &1) << 3;
398             output_data |= (uart_info.status.rx_break &1) << 4;
399             output_data |= (uart_info.status.rx_framing_error &1) << 5;
400             output_data |= (uart_info.status.rx_parity_error &1) << 6;
401
402             // set the output data in MFR_FIRMWARE_COMMAND_DATA
403             PMBUS_Set_Command_Dword((PMBUS_PAGE_t)_BROADCAST_t)page, PMBUS_CMDCODE_MFR_FIRMWARE_COMMAND_DATA, output_data);
404
405         }
406     }
407 }
408 }
409
410 void PMBUS_HANDLE_MFR_SNAPSHOT_DATA(PMBUS_PAGE_t page, PMBUS_DIRECTION_e direction)
411 {
412
413 }
414
415
416 /* Hallo Welt */
417 void PMBUS_HANDLE_MFR_HALLS_WELT(PMBUS_PAGE_t page, PMBUS_DIRECTION_e direction)
418 {
419
420 }
421 }
422

```

Create an empty function with the PMBus MFR command name



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

```
customer_project_demo - C/C++ - patch_hello_welt/src/user_app/pmbus_mfr_specific_handlers.c - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access

pmbus_mfr_autogen.c | pmbus_mfr_specific_handlers.c | user_app.c | pmbus_mfr_autogen.h
190
191     uint32_t output_data = 0; // if command doesn't exist we write command_data to 0
192
193     // here we assume there are no other commands added by patch so first
194     // free command code is PMBUS_MFR_FIRMWARE_COMMAND_END_OF_ENUM
195     if((PMBUS_MFR_COMMAND_SET_e)command == PMBUS_MFR_FIRMWARE_COMMAND_END_OF_ENUM)
196     {
197         output_data = 0;
198         output_data |= (uart_info.status.tx_busy &1) << 0;
199         output_data |= (uart_info.status.rx_overflow &1) << 3;
200         output_data |= (uart_info.status.tx_break &1) << 4;
201         output_data |= (uart_info.status.tx_error &1) << 5;
202         output_data |= (uart_info.status.rx_parity_error &1) << 6;
203
204         // set the output data in MFR_FIRMWARE_COMMAND_DATA
205         PMBUS_Set_Command_Dword((PMBUS_PAGE_M_BROADCAST_t)page, PMBUS_CMDCODE_MFR_FIRMWARE_COMMAND_DATA, output_data);
206     }
207 }
208
209 void PMBUS_HANDLE_MFR_SNAPSHOT_DATA(PMBUS_PAGE_t page, PMBUS_DIRECTION_e direction)
210 {
211 }
212
213 /* Hello Welt */
214 void Hello_Welt(void)
215 {
216     // MP_FUNC1 [20:18] fault1_func[2:0] : 6 - UART0RD
217     // MP_FUNC1 [21]   fault1_pd      : 0 - pulldown disabled
218     // MP_FUNC1 [22]   fault1_pd_n    : 0 - pullup enabled
219     // MP_FUNC1 [23]   fault1_en      : 0 - open drain output
220     // MP_FUNC1 [26:24] fault12_func   : 6 - UART1#E
221     // MP_FUNC1 [27]   fault12_pd      : 0 - pulldown disabled
222     // MP_FUNC1 [28]   fault12_pd_n    : 0 - pullup enabled
223     // MP_FUNC1 [29]   fault12_en      : 1 - CROS output
224     // Refer to Register Map documentation for more details: MP_FUNC1
225     COMMON_MP_FUNC1_SET(0x261B0000);
226
227     ARM_DRIVER_USART * USARTdriver;
228     USARTdriver = ARM_DRIVER_USART;
229     USARTdriver->Initialize(NULL);
230
231     int32_t control = ARM_USART_MODE_ASYNCHRONOUS | ARM_USART_DATA_BITS_8 | ARM_USART_PARITY_NONE |
232                     ARM_USART_STOP_BITS_1 | ARM_USART_FLOW_CONTROL_NONE | ARM_USART_TXIFLSEL_2_B;
233
234     USARTdriver->Control(control, 115200);
235     USARTdriver->Control(ARM_USART_CONTROL_TX_1);
236     USARTdriver->Control(ARM_USART_CONTROL_RX_1);
237
238     char userstring[] = "Hello Welt";
239
240     USARTdriver->Send(userstring, 11);
241
242 }
243
244 /* Hello Welt */
245 void PMBUS_HANDLE_MFR_HELLO_WELT(PMBUS_PAGE_t page, PMBUS_DIRECTION_e direction)
246 {
247 }
248
249 }
```

Proprietary function, algorithm, etc.

- Proprietary function, algorithm, etc.

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

customer_project_demo - C/C++ - patch_hallo_welt/src/user_app/pmbus_mfr_specific_handlers.c - Eclipse

```

1  pmbus_mfr_autogen.c  pmbus_mfr_specific_handlers.c  user_app.c  pmbus_mfr_autogen.h
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
2  pmbus_mfr_autogen.c  pmbus_mfr_specific_handlers.c  user_app.c  pmbus_mfr_autogen.h
3  409  output_data |= uart_info.status_rx_break << 4;
410  output_data |= uart_info.status_rx_framing_error << 5;
411  output_data |= uart_info.status_rx_parity_error << 6;
412
413  // set the output data in MFR_FIRMWARE_COMMAND_DATA
414  PMBUS_Set_Command_Dword((PMBUS_PAGE_t)page, PMBUS_CMDCODE_MFR_FIRMWARE_COMMAND_DATA, output_data);
415
416 }
417
418 /* Hello Welt */
419 void Hallo_Welt(void)
420 {
421     // MP_FUNC1 [20:18] fault1_func[2:0] : 6 - UARTTXD
422     // MP_FUNC1 [19] fault1_pd : 0 - pulldown disabled
423     // MP_FUNC1 [22] fault1_pe_n : 0 - pulldown enabled
424     // MP_FUNC1 [23] fault1_popen : 0 - open drain output
425     // MP_FUNC1 [26:24] fault2_func : 6 - UARTRXD
426     // MP_FUNC1 [27] fault2_pd : 0 - pulldown disabled
427     // MP_FUNC1 [28] fault2_pe_n : 0 - pullup enabled
428     // MP_FUNC1 [29] fault2_popen : 1 - OOS output
429     // Refer to Register Map documentation for more detail: MP_FUNC1
430     COMMON_MP_FUNC1_SET(0x26100000);
431
432     API_DRIVER_USART *UARTdriver;
433     UARTdriver = API_DRIVER_USART;
434     UARTdriver->Initialize(MAIL);
435
436     int32_t control = ARM_USART_MODE_ASYNCHRONOUS | ARM_USART_DATA_BITS_8 | ARM_USART_PARITY_NONE |
437                     ARM_USART_STOP_BITS_1 | ARM_USART_FLOW_CONTROL_NONE | ARM_USART_TXIFLSEL_7_8;
438
439     UARTdriver->Control(control, 115200);
440     UARTdriver->Control(ARM_USART_CONTROL_TX, 1);
441     UARTdriver->Control(ARM_USART_CONTROL_RX, 1);
442
443     char userstring[] = "Hallo Welt";
444     UARTdriver->Send(userstring, 11);
445 }
446
447 /* Hello Welt */
448 void PMBUS_HANDLE_MFR_HELLO_WELT(PMBUS_PAGE_t page, PMBUS_DIRECTION_e direction)
449 {
450     // Do Hello Welt if there is a PMbus write
451     #if ((direction == PMBUS_WRITE) || (direction == OTP_RESTORE))
452     {
453         // get the command code here:
454         uint8_t command = PMBUS_Get_Command_Byt
455
456         if (command == 0xFF)
457         {
458             Hallo_Welt();
459         }
460     }
461 }
462

```

Call proprietary function in the
PMBus MFR Handler

10.3.5 STEP 5: Clean and Build project.

Build project

```

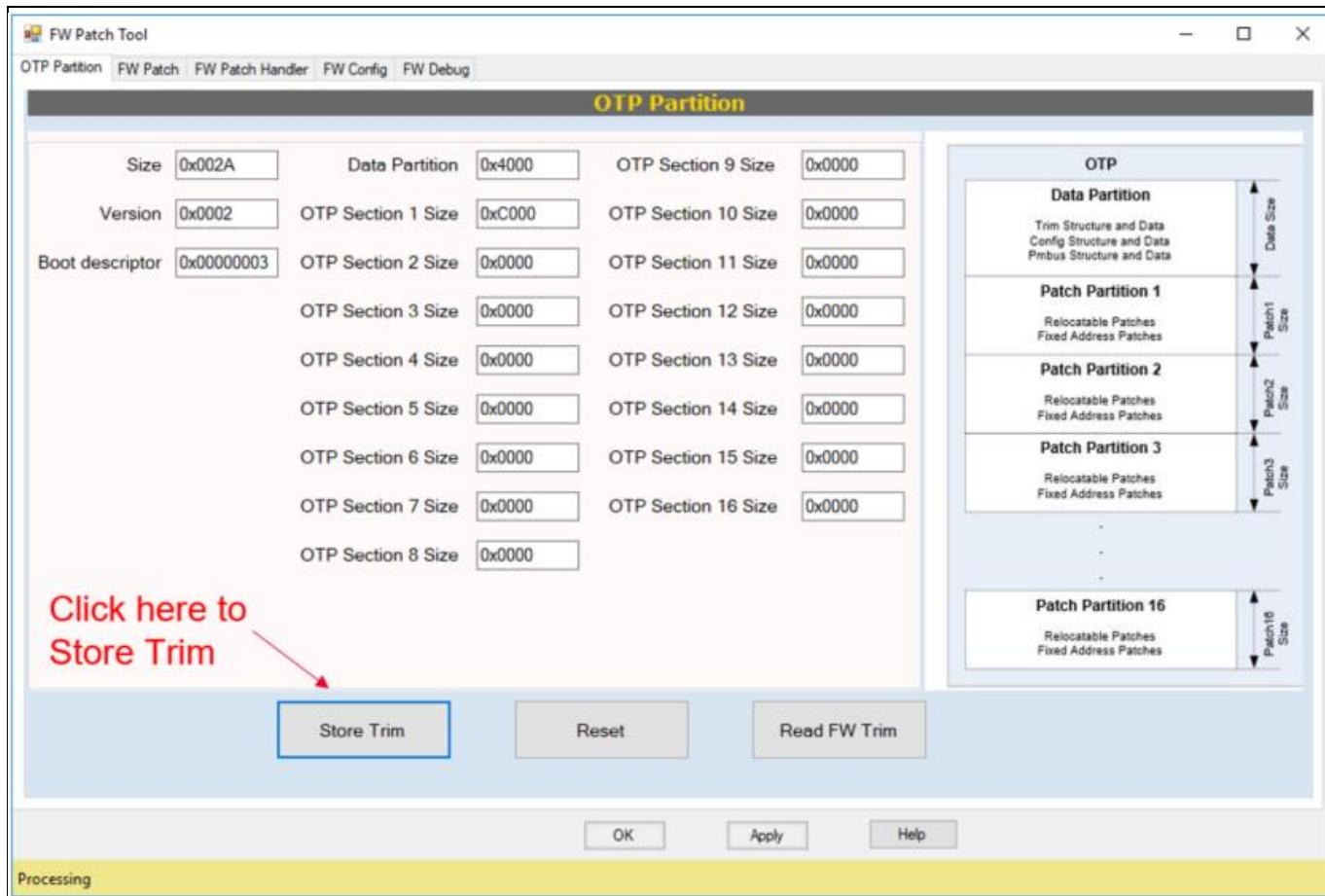
customer_project_demo - C/C++ - patch_hallo_welt/src/user/app/pmbus_mfr_specific_handlers.c - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer  Build Targets  Out...  Quick Access
patch_hallo_welt  pmbus_mfr_autogen.h  pmbus_mfr_specific_handlers.c  user_app.c  pmbus_mfr_autogen.h
  Build Targets
    clear
    patch_TOOLCHAIN=ARM
    patch_TOOLCHAIN=ARM_GCC
  pmbus
    docx
    patch_hallo_welt
    Build Targets
      clear
      patch_hallo_welt
      patch_TOOLCHAIN=ARM
      patch_TOOLCHAIN=ARM_GCC
    common
      modules
        system_header
        utilities
          MakefileCommon.mk
          MakefileCommonARM.mk
          MakefileCommonGCC.mk
          MakefilePatching.mk
          MakefilePatchingARM.mk
          MakefilePatchingGCC.mk
          MakefilePmbus.mk
          MakefileSimVision.mk
    src
      bootcode_patch_from_firmware
      Invitations.patch.cmm
      load_rom_eif.cmm
      Makefile
      profile.telemetry.cmm
      project.windows.cmm
      setup_jw.cmm
      Store_patch.cmm
      Store_trim.cmm
      t32_start.bat
    patch_hello_world
    patch_hola_mundo
    patch_temp_softstart
    patch_user_app
  pmbus
    pmbus_apl.h
    regulation_drv.h
    pmbus_autogen.h
    pmbus_mfr_autogen.h
    shasta_hat_socah
    shasta_hat_cqah
    shasta_hat_telem
    shasta_hat_fsn.h
    shasta_hat_trim.h
    shasta_hat_vseen.h
    regulation_apl.h
    faults_apl.h
    faults_drv.h
    regulation_drv.h
    telemetry_drv.h
    pmbusDrv.h
    driver_common.h
    logApp.h
    regulation_state_machine.calib
    regulation_state_machine
    uartDrv.h
    userApp.h
    UARTDrv_ARM_DRIVER_USART
    src_buf : uint8_t []
    dest_buf : uint8_t []
  PMBUS_HANDLE_MFR_CONFIG
  PMBUS_HANDLE_FAN_COMM
  PMBUS_HANDLE_FAN_CONFIG
  PMBUS_HANDLE_MFR_FREQUE
  PMBUS_HANDLE_MFR_BOARD
  PMBUS_HANDLE_MFR_SHARE
  PMBUS_HANDLE_MFR_EDID
  PMBUS_HANDLE_MFR_VDD_S5
  PMBUS_HANDLE_MFR_VIN_S5
  PMBUS_HANDLE_MFR_FW_COF
  patch_PMBUS_HANDLE_MFR_FW
  PMBUS_HANDLE_MFR_SNAPS
  Hallo_Welt(void)
  PMBUS_HANDLE_MFR_HALO...

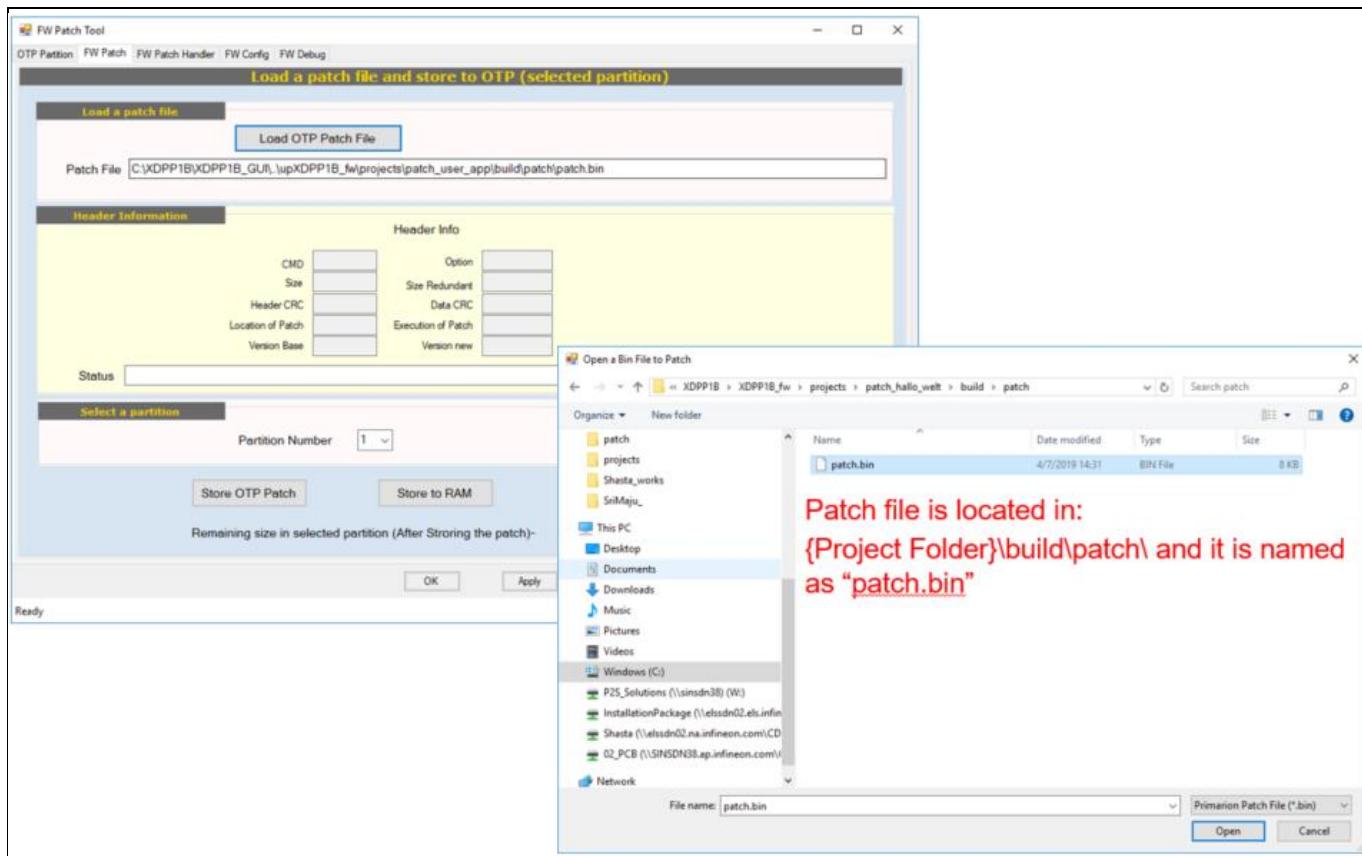
```

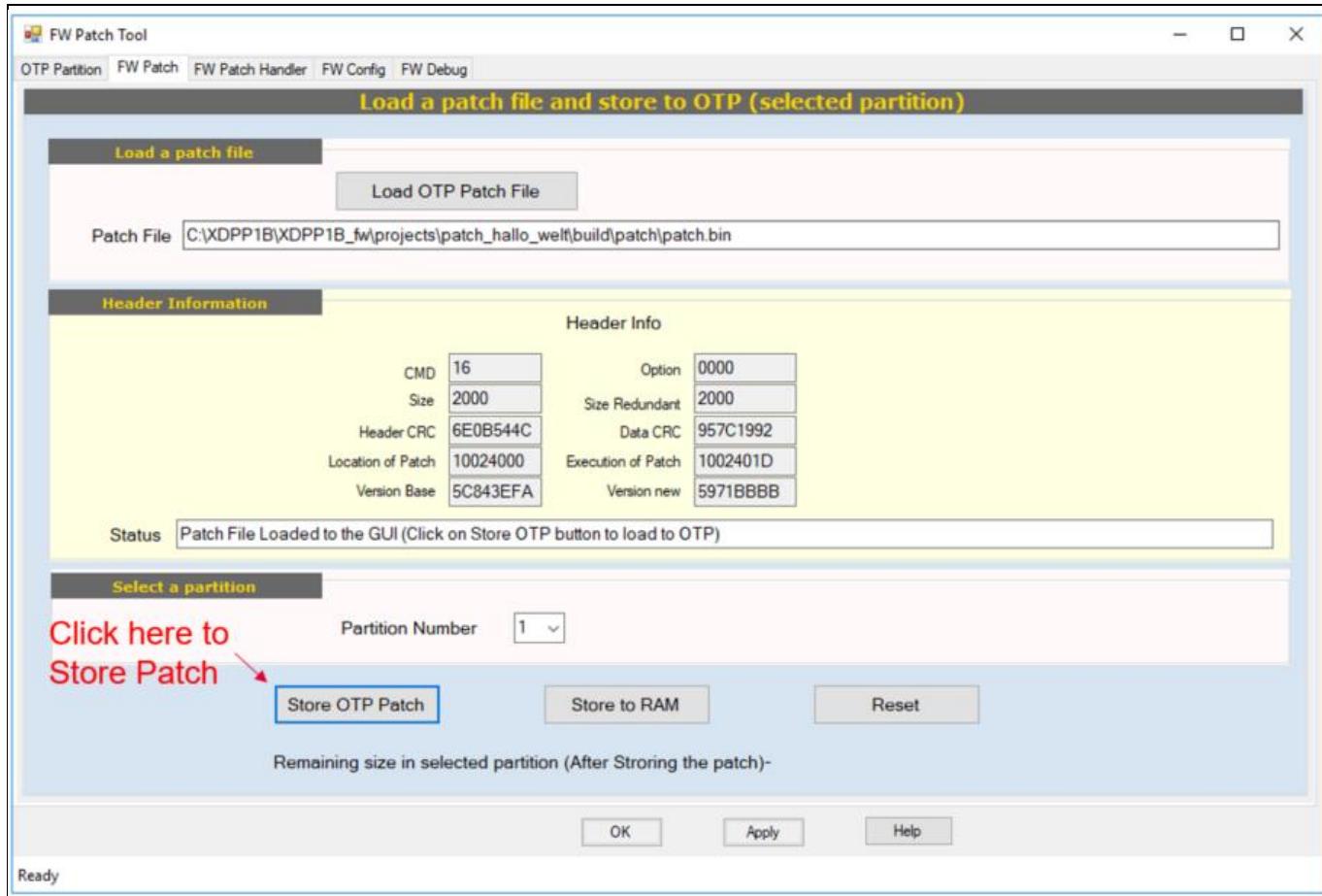
Build finished if successful

14:31:13 Build Finished (took 19s.934ms)

10.3.6 STEP 6: Download the Patch

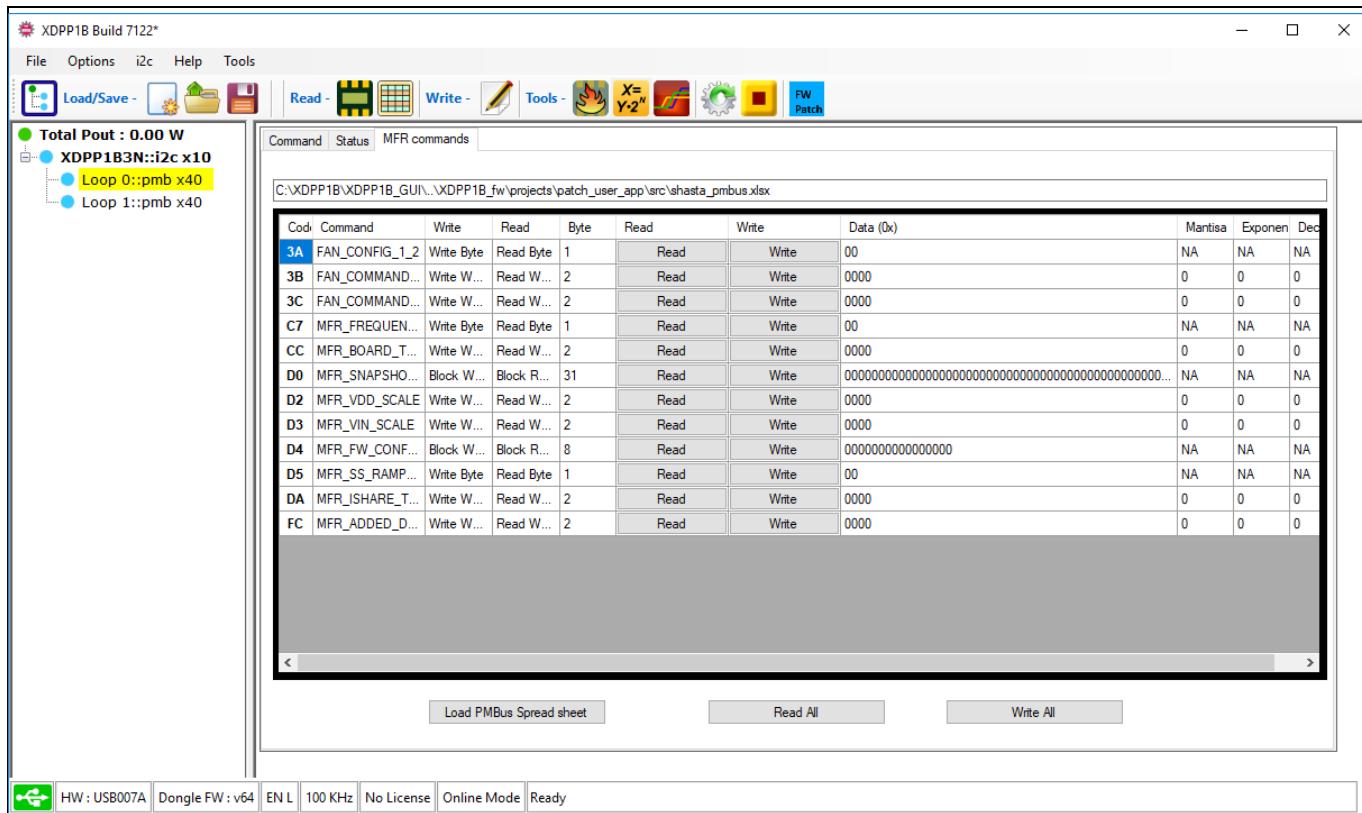




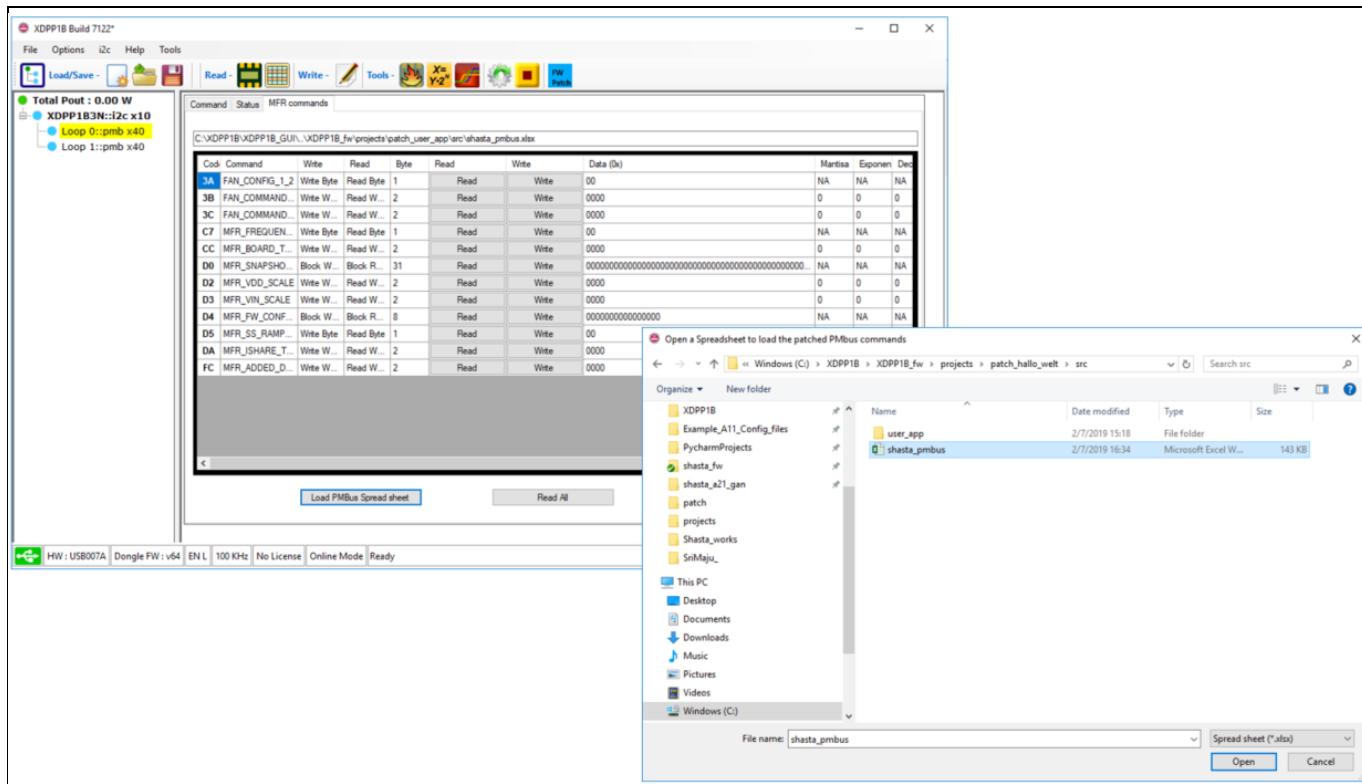


10.4 Test the newly created PMBus MFR command

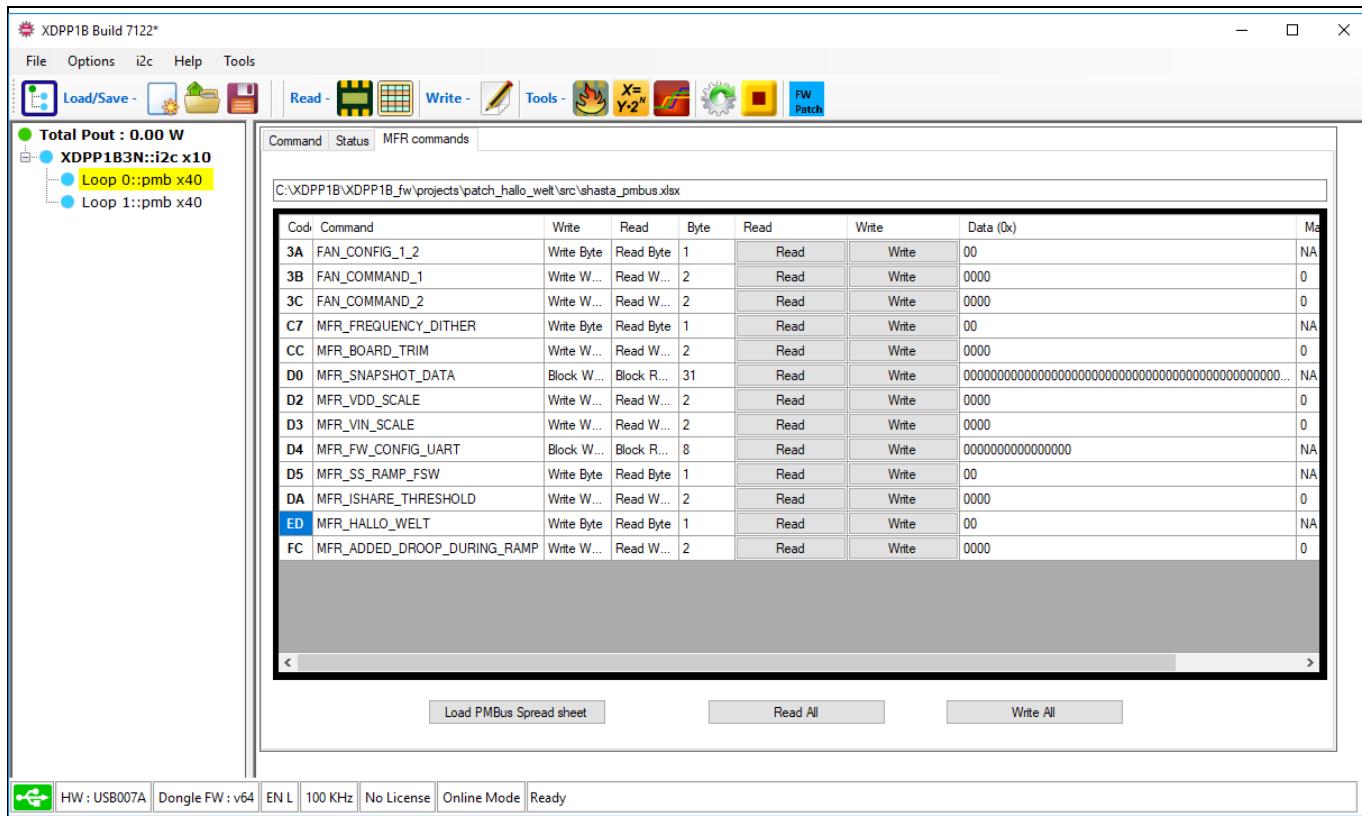
10.4.1 STEP 1: Open XDPP1100 GUI and go to “MFR commands” tab



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



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	<p>To indicate whether the PMBus has firmware handler (callback function).</p> <p>Put ‘y’ to indicate there is a need for firmware handler. A function declaration with default name will be generated automatically.</p> <p>Put ‘n’ if there is no need for firmware handler.</p> <p><i>It is also possible to put in the function name immediately but ensure there is no duplicate naming with any other commands.</i></p> <p><i>Users are encouraged to stick to ‘y’ with default generated function name to minimize error.</i></p>
Coverage Loop 0	[DO NOT MODIFY]
Coverage Loop 1	
Common Storage	<p>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).</p> <p>Setting ‘y’ to this column will force the Loop 1 generated codes to point to Loop 0.</p>
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”.

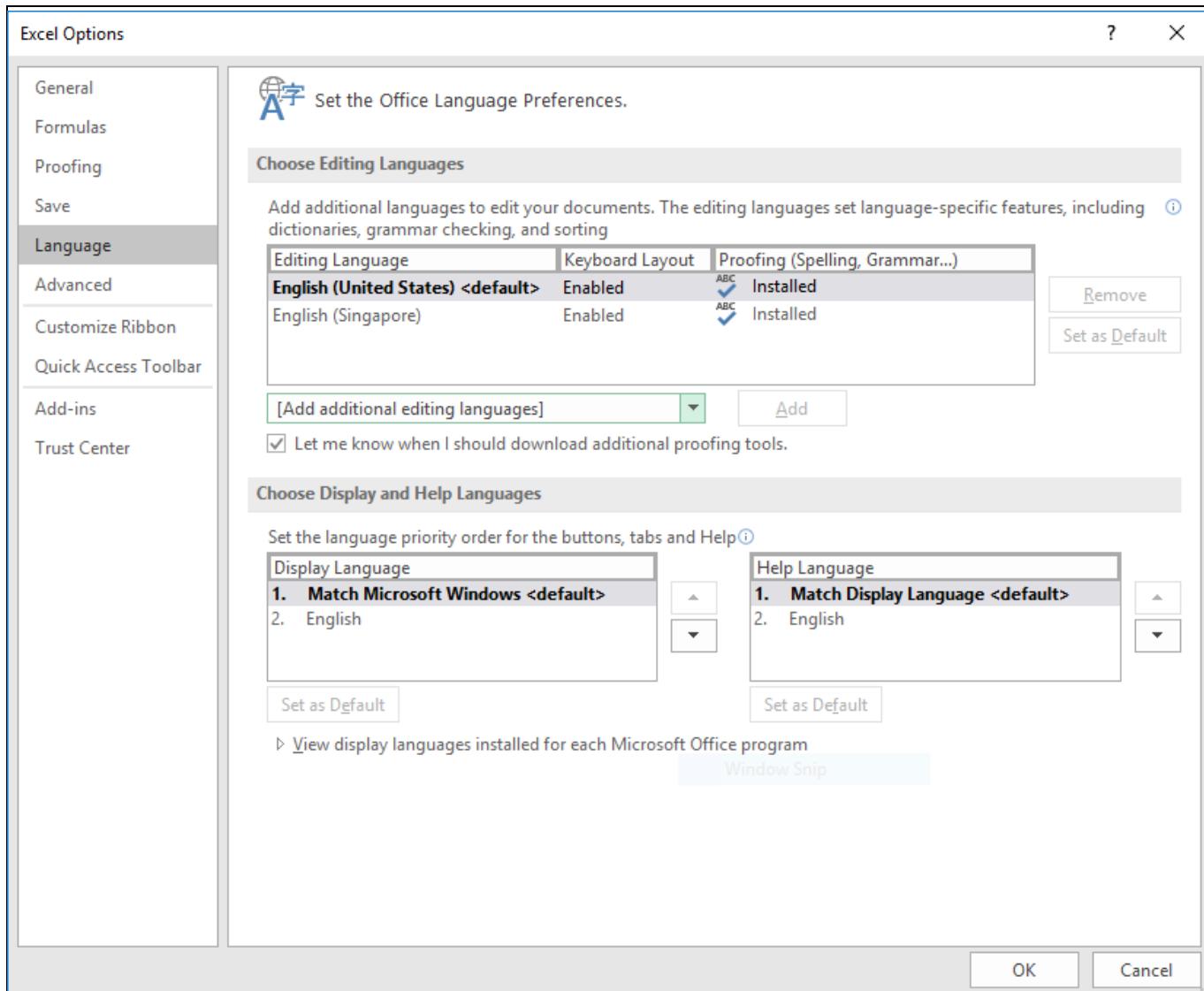


Figure 6 Excel Language Options

11 More Resources

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

11.1 Firmware Documentation

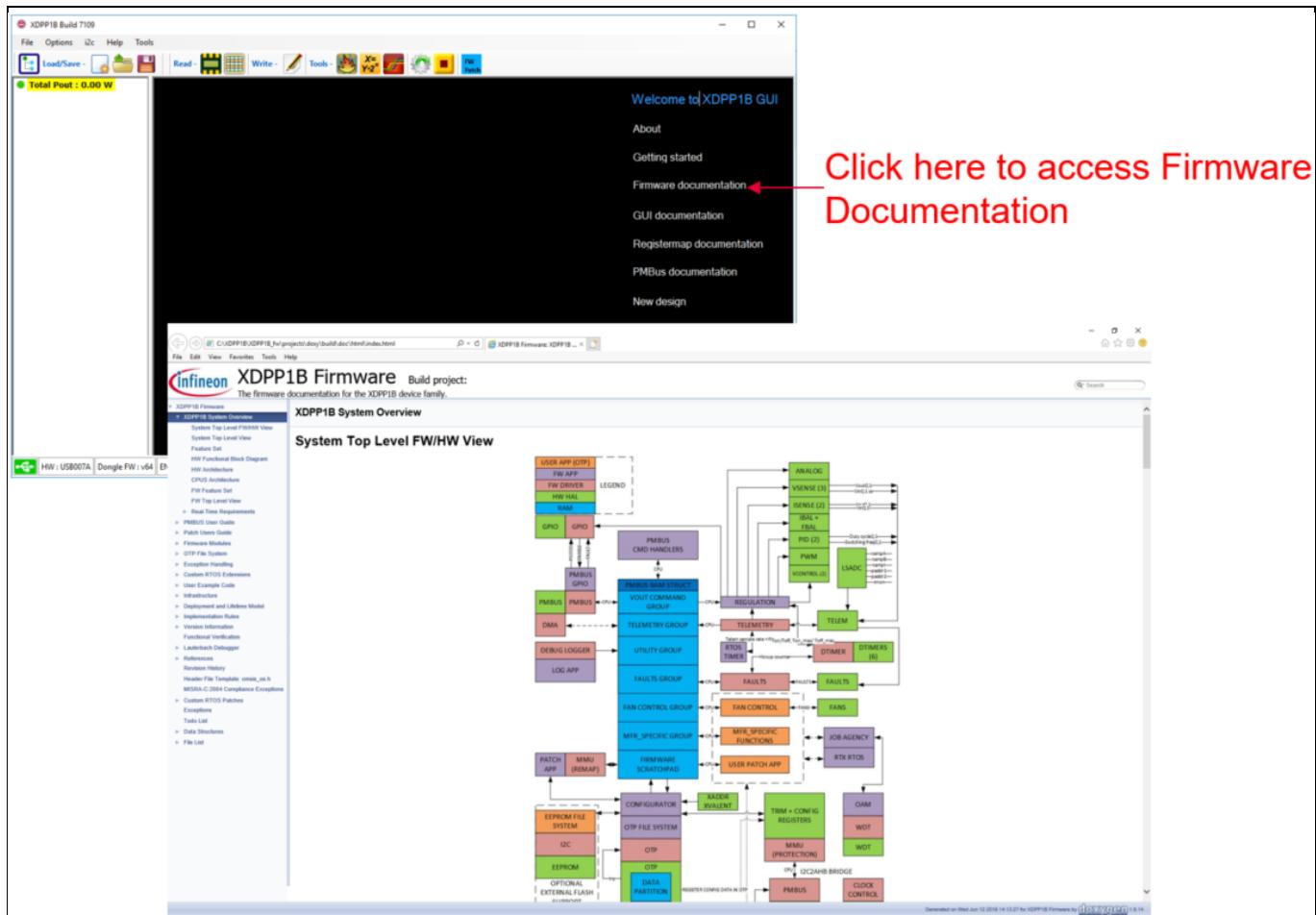


Figure 7 Firmware Documentation

11.2 Register Map Documentation

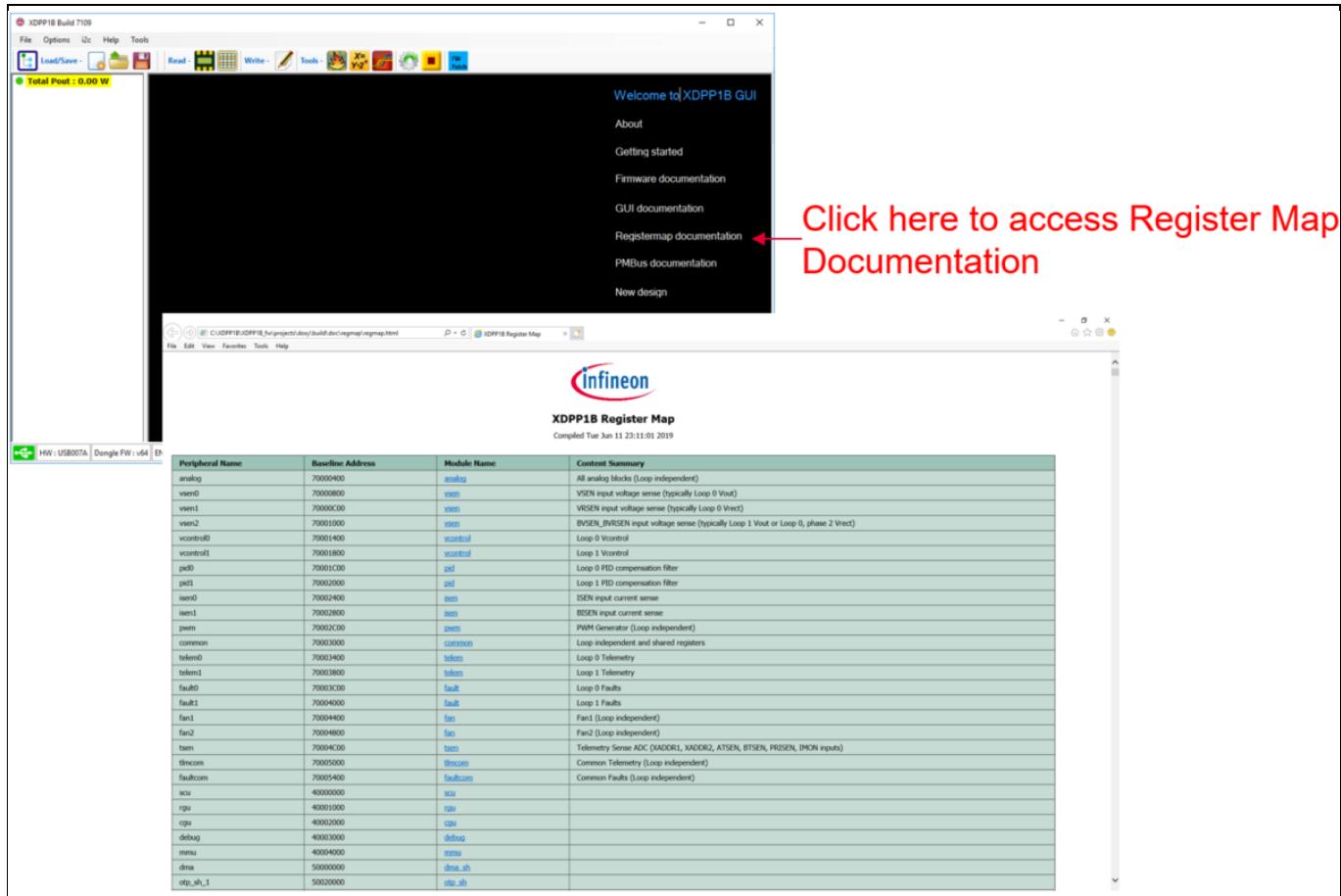


Figure 8 Register Map Documentation

11.3 PMBus Documentation

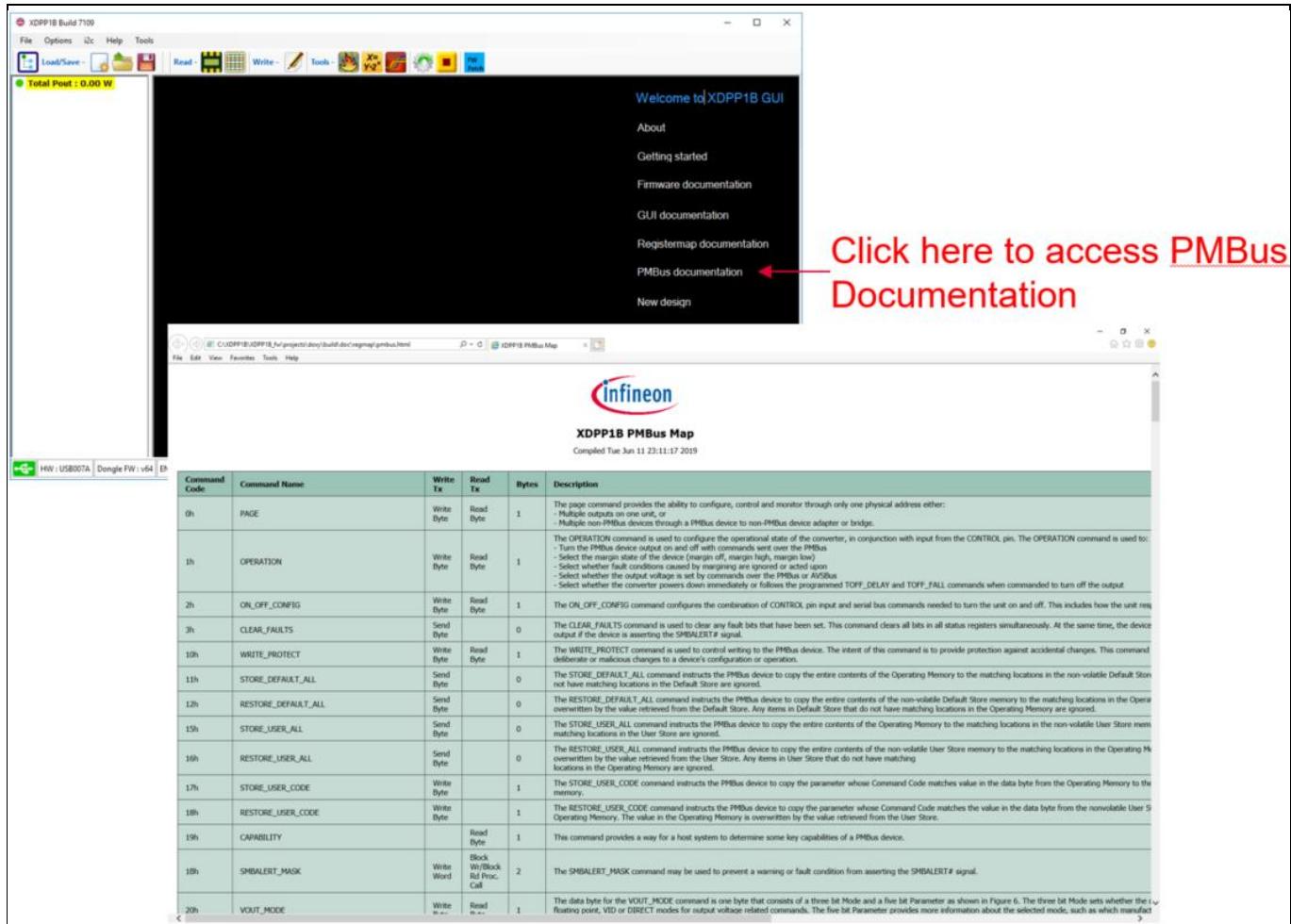


Figure 9 PMBus Documentation

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>

restricted

XDPP1100 Firmware User Guide

Getting Started in XDPP1100 Custom Firmware Development

Revision history



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

© 2020 Infineon Technologies AG.
All Rights Reserved.

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.