XDPP1100 firmware development tutorial

About this document

Scope and purpose

This document provides examples for customizing firmware (FW) development in patch form, which can be stored in XDPP1100 one-time-programmable memory (OTP) or random-access memory (RAM). It provides step-by-step examples to create new projects.

Intended audience

Field application engineers, application engineers, and firmware engineers.

Table of contents

About this document **1**
Table of contents **2**
1 Introduction to XDPP1100 FW tutorial **4**
2 New project creation **6**
2.1 Create new project 6
2.2 Add a new PMBus MFR command 9
3 PMBus MFR command control GPIO output **13**
3.1 PMBus MFR command for GPIO control 13
3.2 Modify source code 14
3.2.1 PMBus command handler 14
3.2.2 User app 16
3.3 Compile and program to XDPP1100 RAM/OTP 16
3.3.1 Compile source code 16
3.3.2 Connect to evaluation board 17
3.3.3 XDPP1100 GUI operation 17
3.3.4 XDPP1100 GUI patch download 22
3.3.5 Running example 1 patch 24
4 Frequency dithering **28**
4.1 PMBus MFR command for frequency dithering 28
4.2 Modify source code 30
4.2.1 PMBus command handler 31
4.2.2 User data variable and regulation callback function 32
4.2.3 Declare function for frequency dithering 33
4.2.4 Regulation callback function 36
4.2.5 User app 38
4.3 Compile and program to XDPP1100 RAM/OTP 38
4.3.1 Running example 2 patch 38
5 Telemetry interrupt and FW fault **42**
5.1 PMBus MFR command for OVP fault limit 42
5.2 Modify source code 43
5.2.1 Add telemetry interrupt functions and PMBus handler function 43
5.2.2 Add header file initialization 47
5.2.3 User app 48
5.3 Compile and program to XDPP1100 RAM/OTP 48
5.3.1 Running example 4 patch 48
6 PWM kickstart pulse **52**
6.1 Modify source code 52
6.1.1 Add kickstart pulse function 52
6.1.2 Add header file initialization 56
6.1.3 User data variable and regulation callback function 57
6.1.4 Regulation callback function 58
6.1.5 User app 59
6.2 Compile and program to XDPP1100 RAM/OTP 60
6.2.1 Running example 5 patch 60
7 Delay secondary PWM start **62**
7.1 Modify source code 62
7.1.1 Add soft-start SR PWM function 62

Table of contents

7.1.2 Add header file initialization 66
7.1.3 User data variable and regulation callback function 66
7.1.4 Regulation callback function 68
7.1.5 Add initialize function 70
7.1.6 User app 70
7.2 Compile and program to XDPP1100 RAM/OTP 71
7.2.1 Running example 6 patch 71
References 73
Revision history 74

1 Introduction to XDPP1100 FW tutorial

This document provides a step-by-step guide to creating customized FW in the form of a patch program to OTP to add functionality not provided in the default FW integrated into the ROM.

For a guide to setting up and starting to develop customized FW, refer to the **XDPP1100 product page** and the **XDPP1100 firmware user guide**.

Documentation for the default FW integrated into the ROM can be found in the XDPP1100 firmware documentation [3].

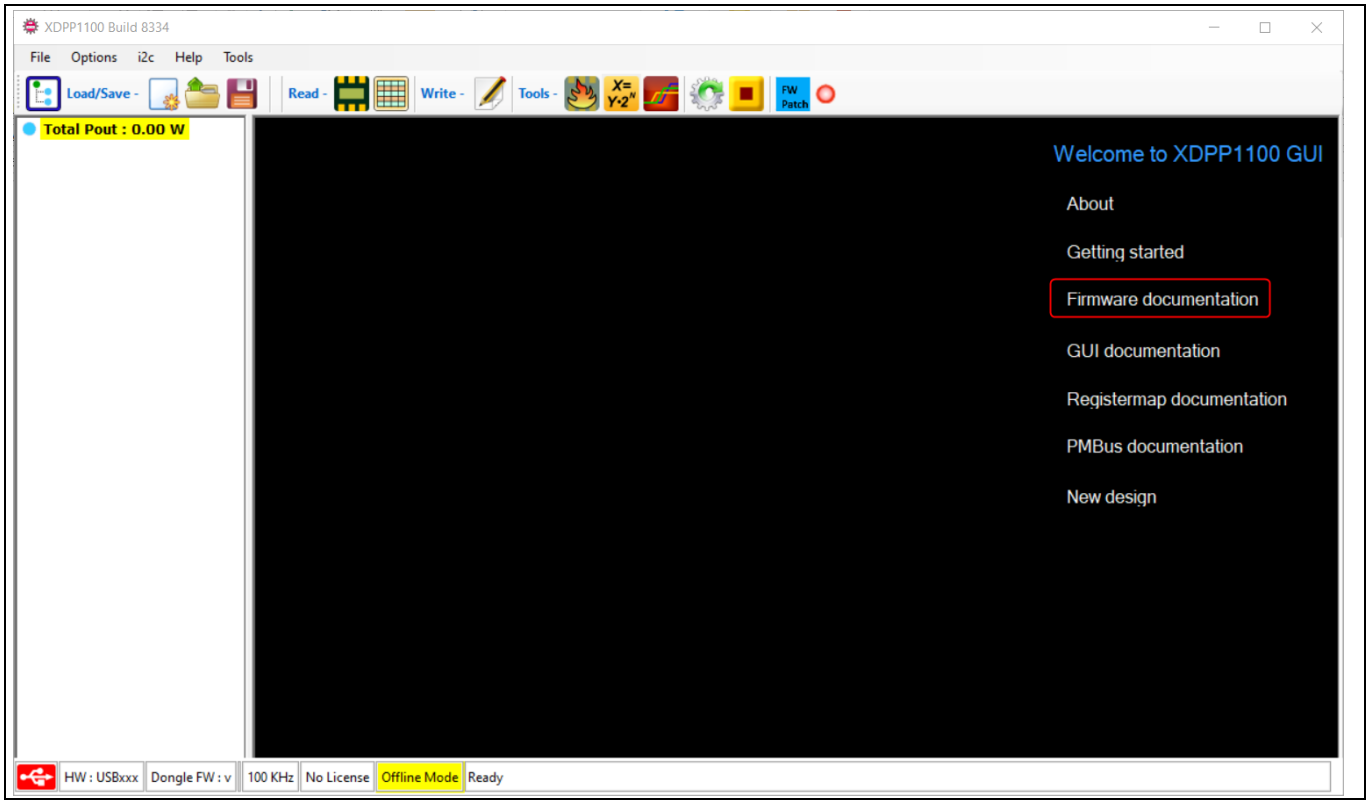


Figure 1 XDPP1100 GUI build 8334

References for these project examples can be found at **Infineon BitBucket DMZ**. You can request access through your local field application engineer (FAE) to create an account and clone the repo.

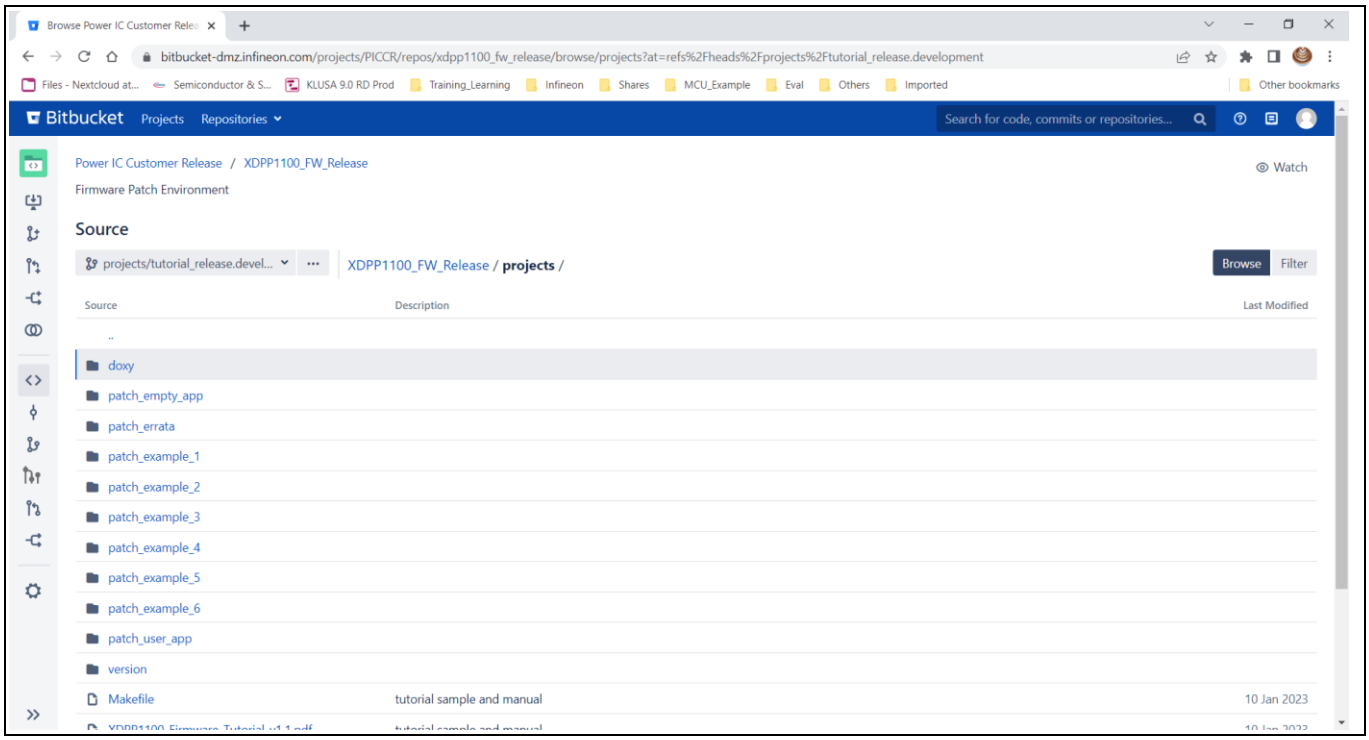


Figure 2 Bitbucket DMZ

2 New project creation

2.1 Create new project

In a newly installed workspace, update the path for Python, Eclipse and Java installation in “user_env.bat”.

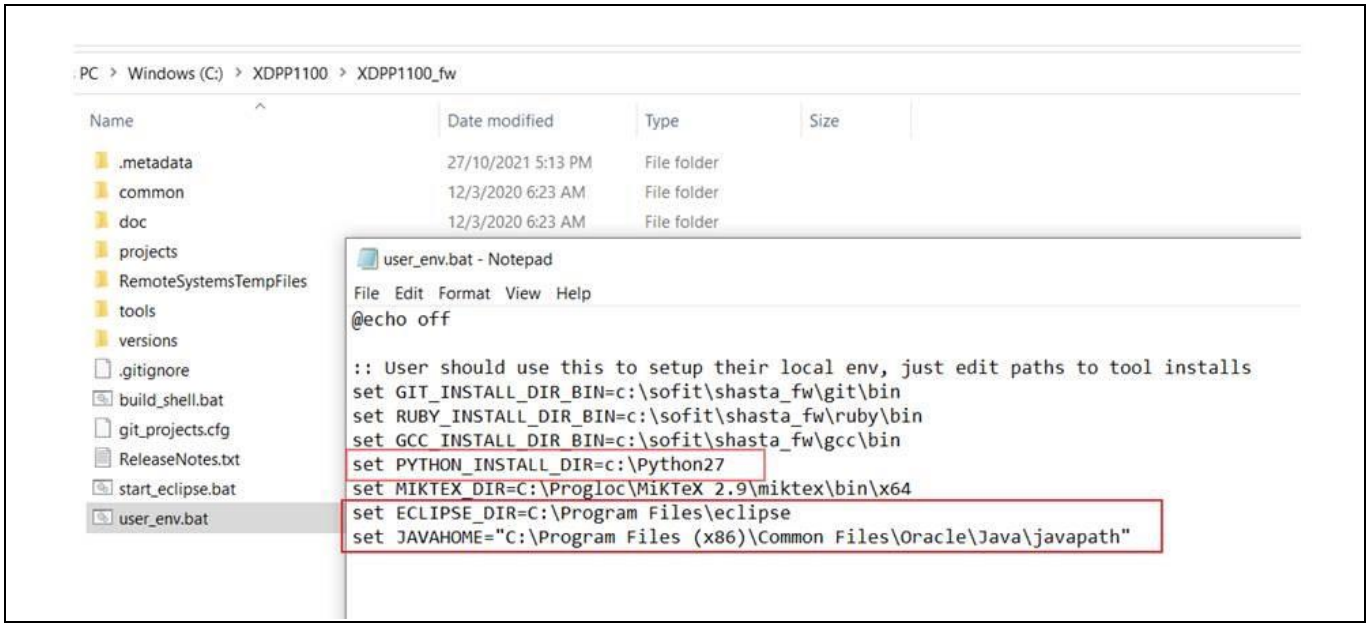


Figure 3 Installation path

Open the XDPP1100 FW patch workspace by clicking on “start_eclipse.bat” in the XDPP1100_fw folder.

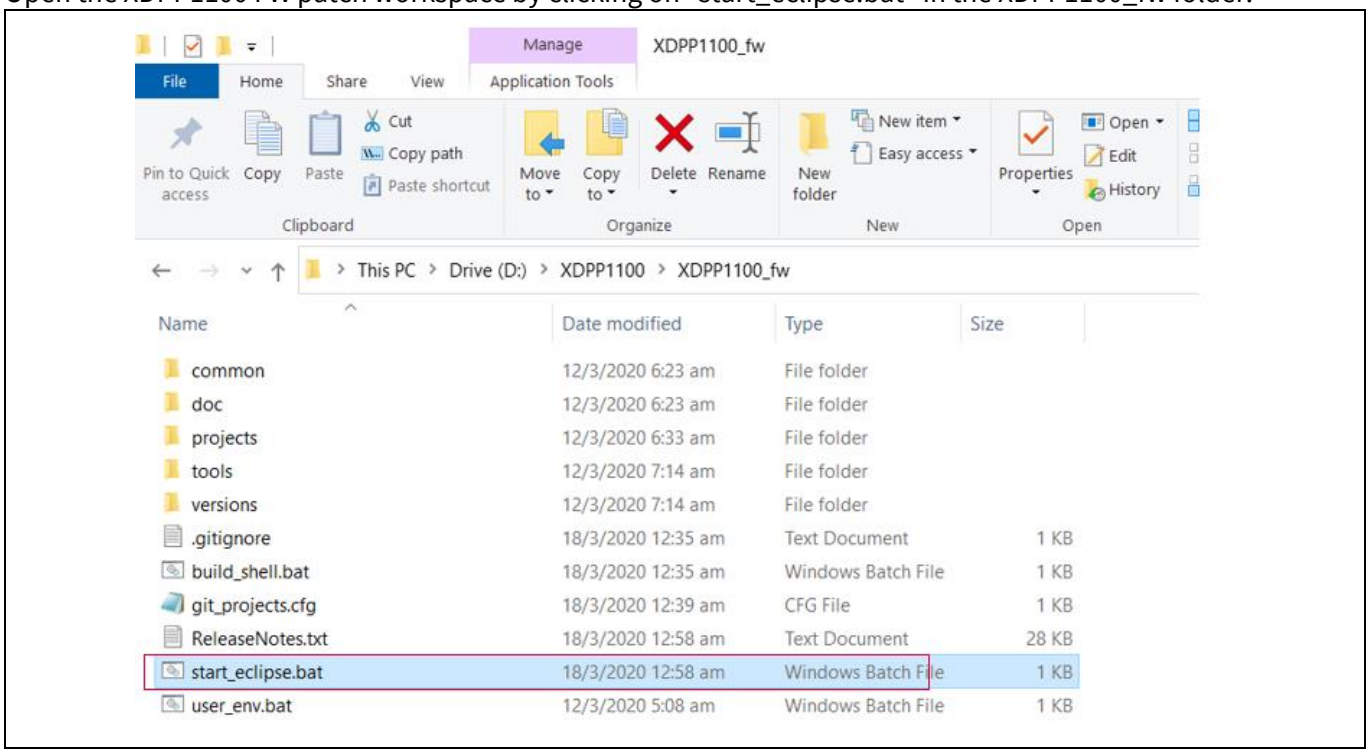


Figure 4 XDPP1100 FW directory

Select a workspace directory. Use the XDPP1100_fw folder as the directory.

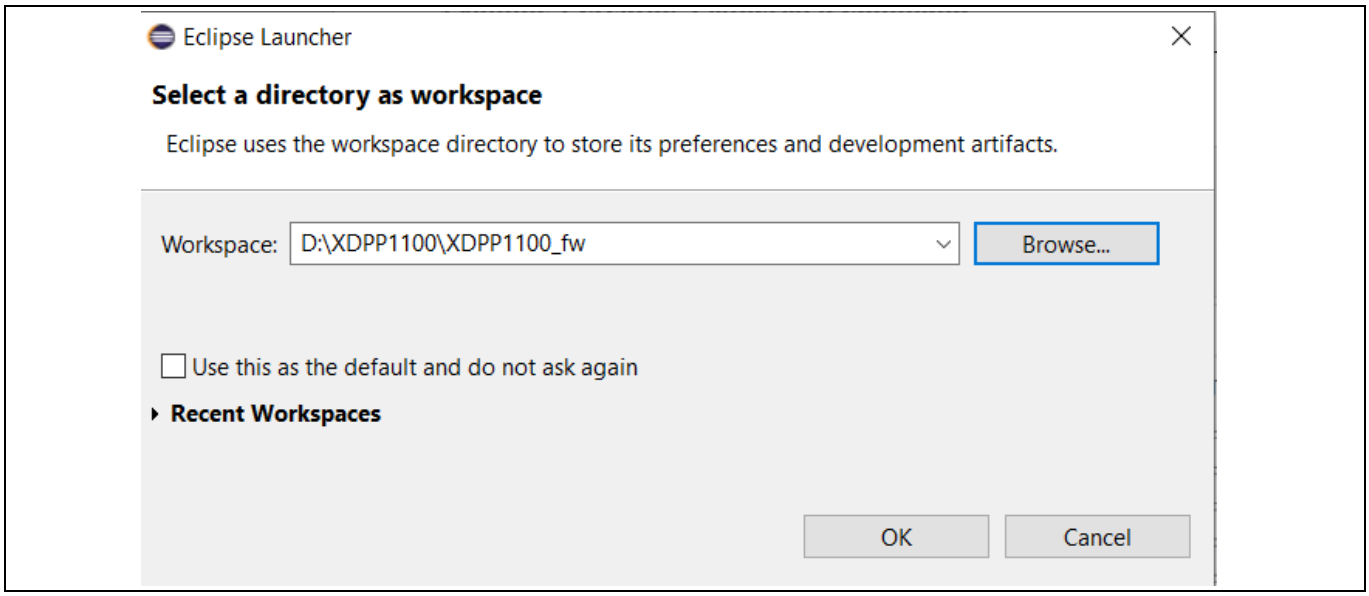


Figure 5 Select the workspace directory window of Eclipse

Import the sample FW patch project provided. Go to File > Import.

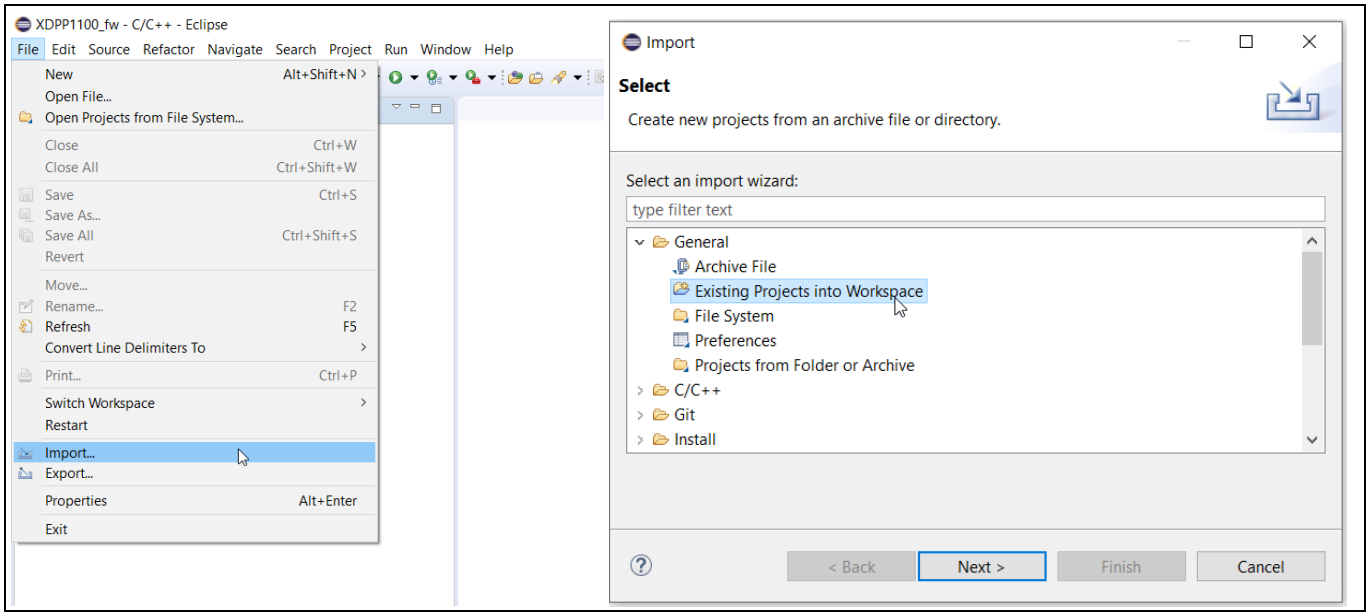


Figure 6 Eclipse environment – import existing project, step 1

Select the patch_empty_app to import.

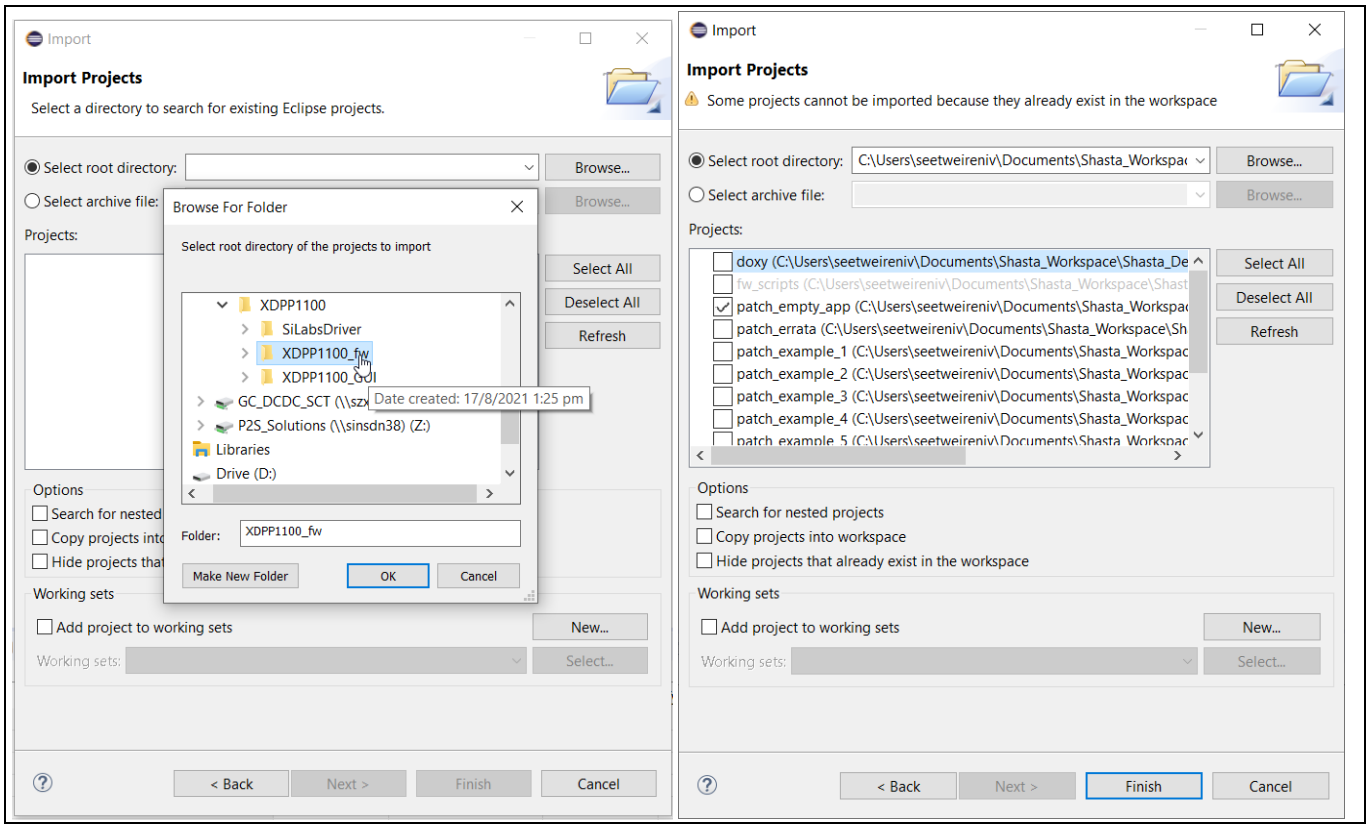


Figure 7 Eclipse environment – import existing project, step 2

Copy patch_clean_version and paste to create a new project patch_example_1.

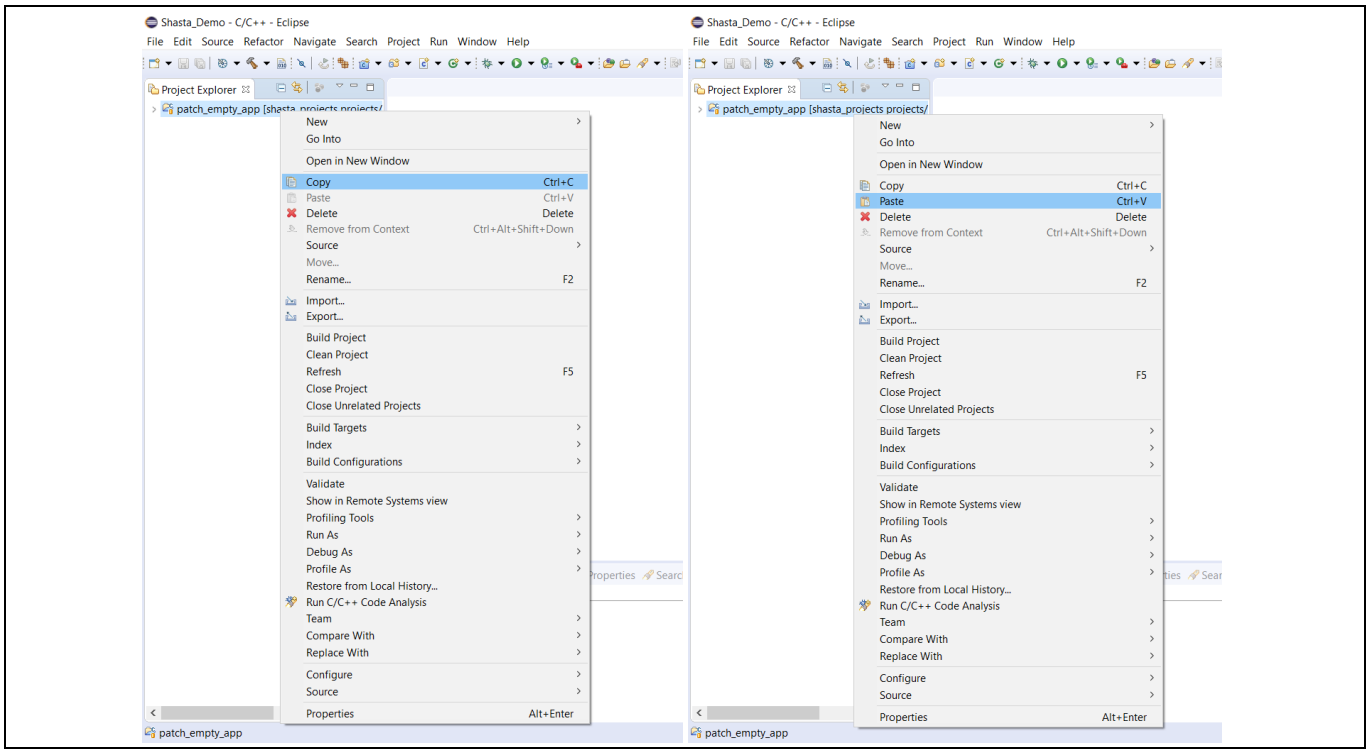


Figure 8 Eclipse environment – create new project from existing project, step 1

Key the project name as patch_example_1 and select location as ..\XDPP1100\XDPP1100_fw\projects\patch_example_1.

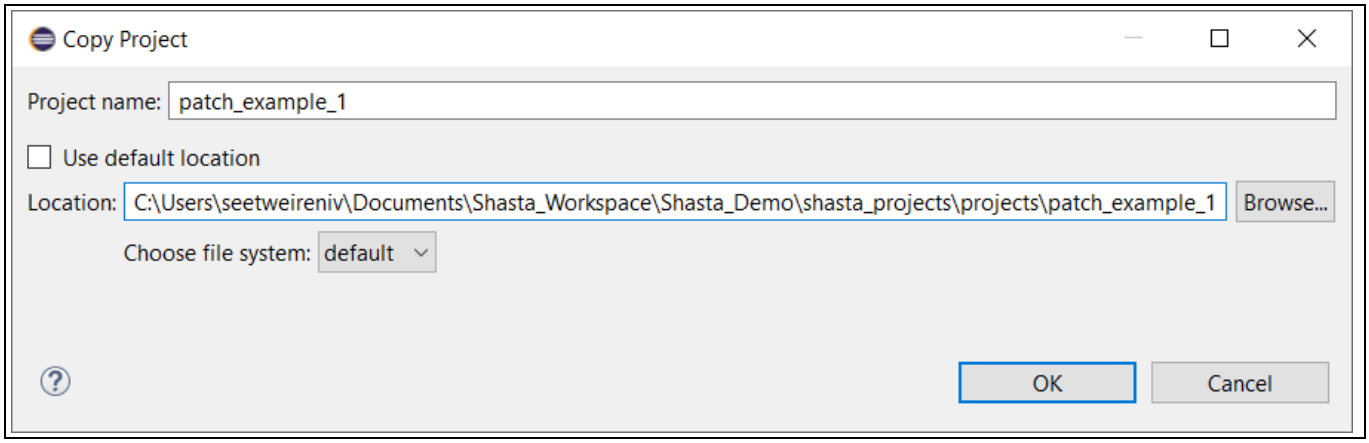


Figure 9 Eclipse environment – import existing project, step 2

patch_example_1 is created in the workspace.

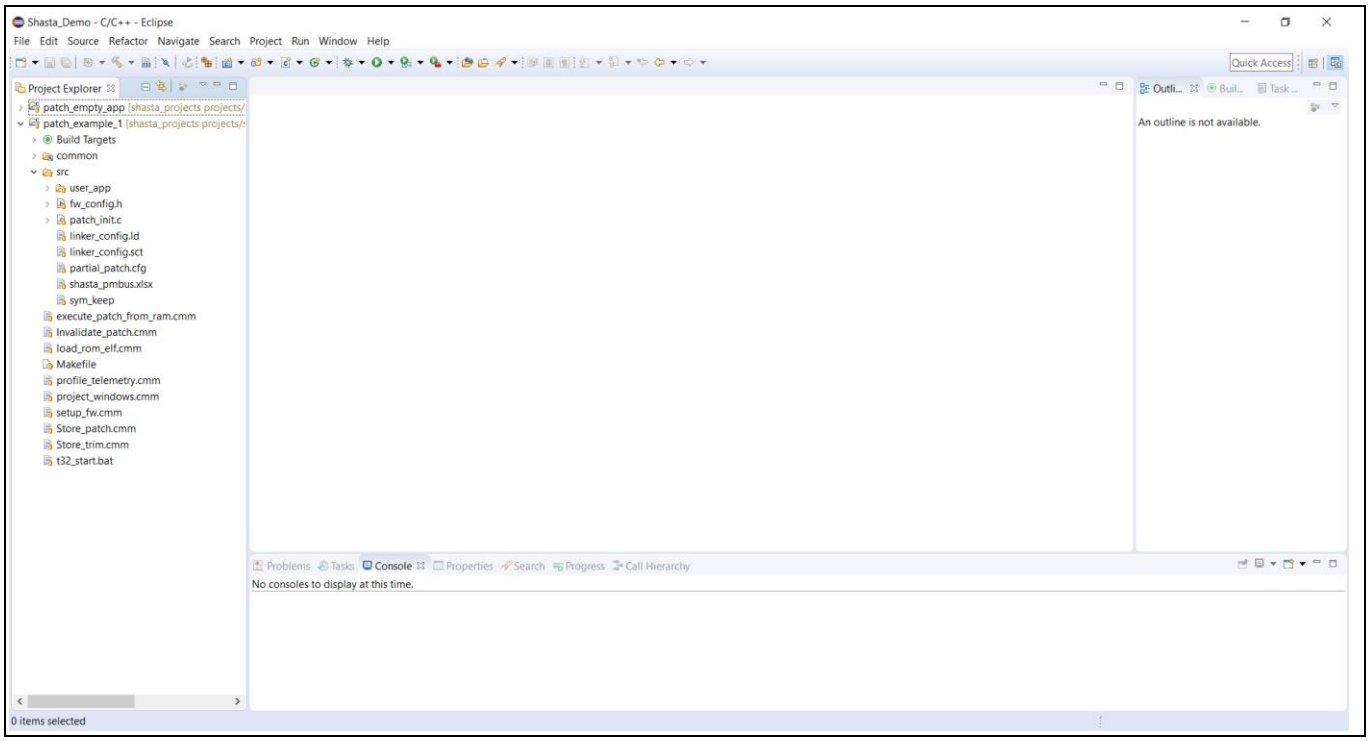


Figure 10 Eclipse environment – import existing project, step 3

2.2 Add a new PMBus MFR command

Open shasta_pmbus.xlsx under the src folder of the project, for example “\XDPP1100\XDPP1100_fw\projects\patch_example_1\src”.

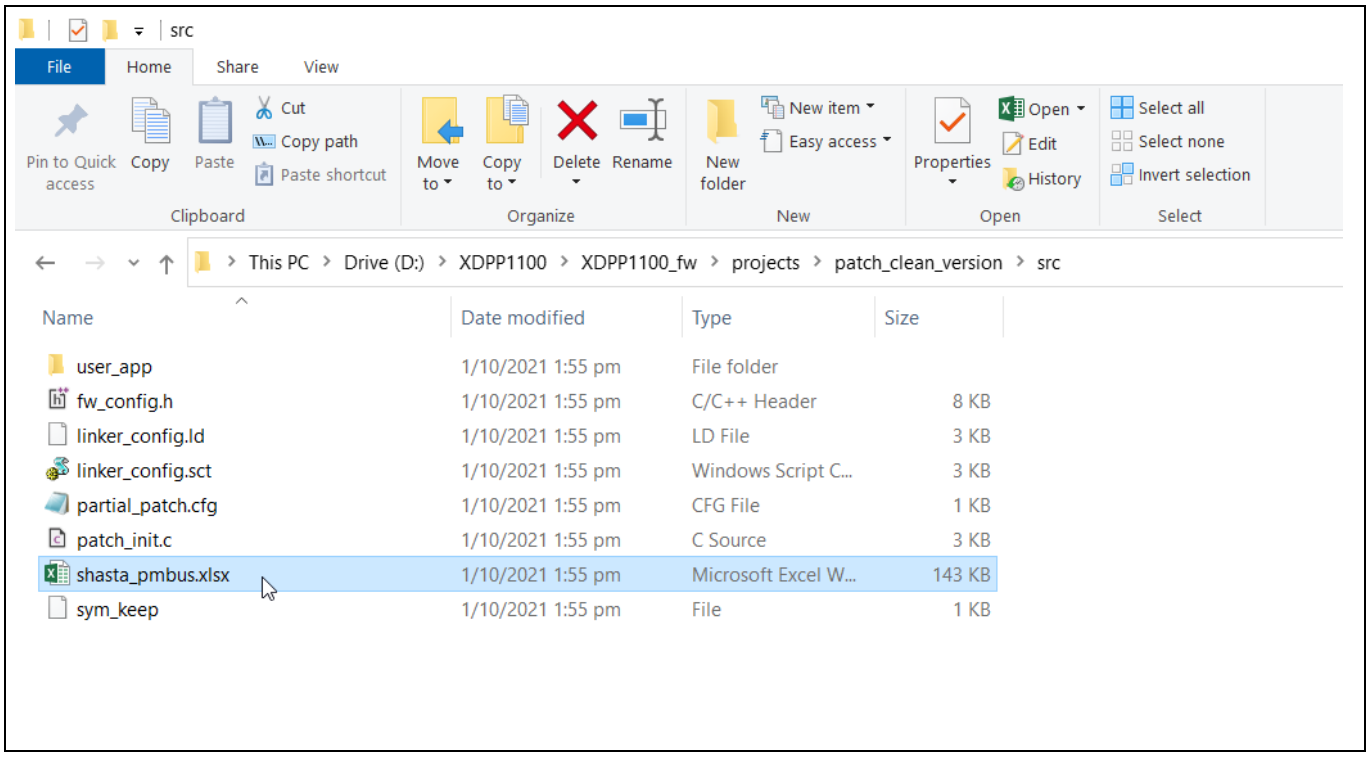


Figure 11 FW patch project directory – open shasta_pmbus.xlsx

Find an available command code to assign to the new MFR commands. In this example command code 0xB2 on the spreadsheet is chosen. Enable the command as shown in **Figure 18**. Set column F (MFR) to y, column G (Loop 0 support) to y and column U (has FW HANDLE) to y. If support for Loop 1 is required expand the column after I to look for the Loop 1 support column and set accordingly.

In this example Loop 1 is not used, therefore it is set to “n”.

Save and close the spreadsheet after modification.

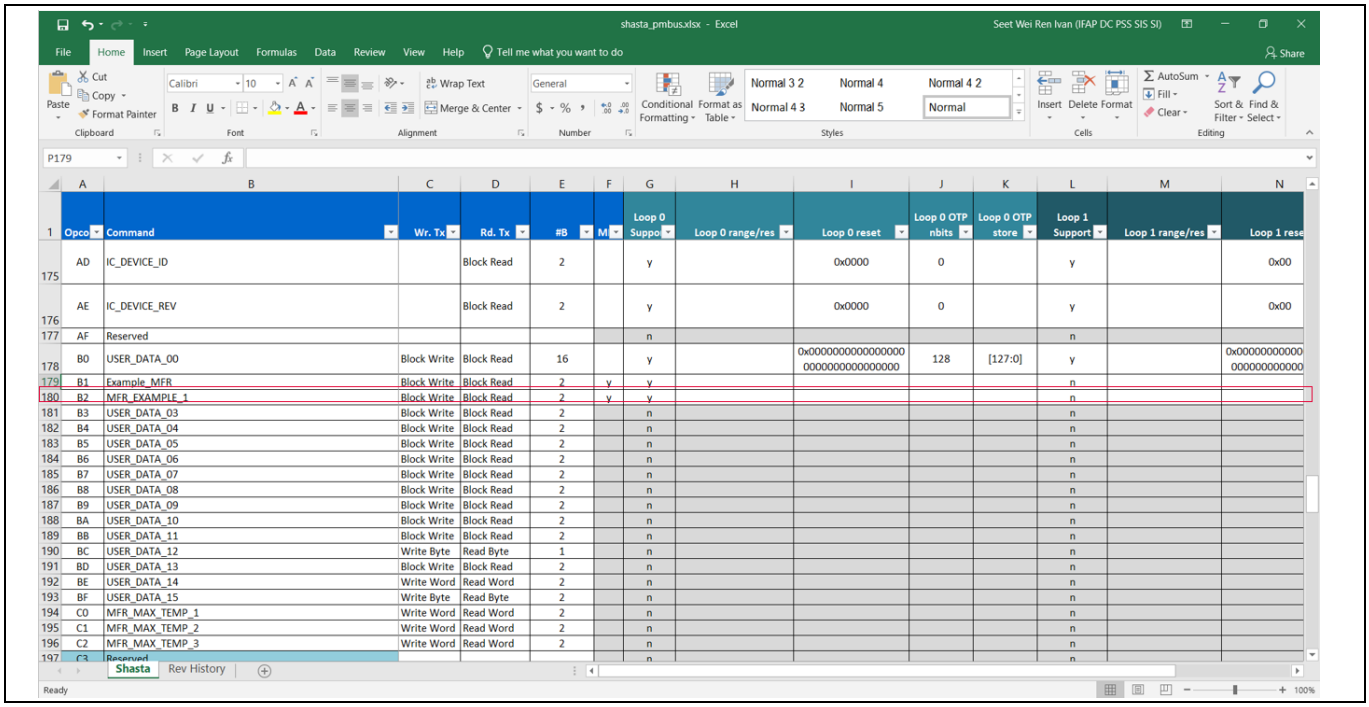


Figure 12 shasta_pmbus.xlsx

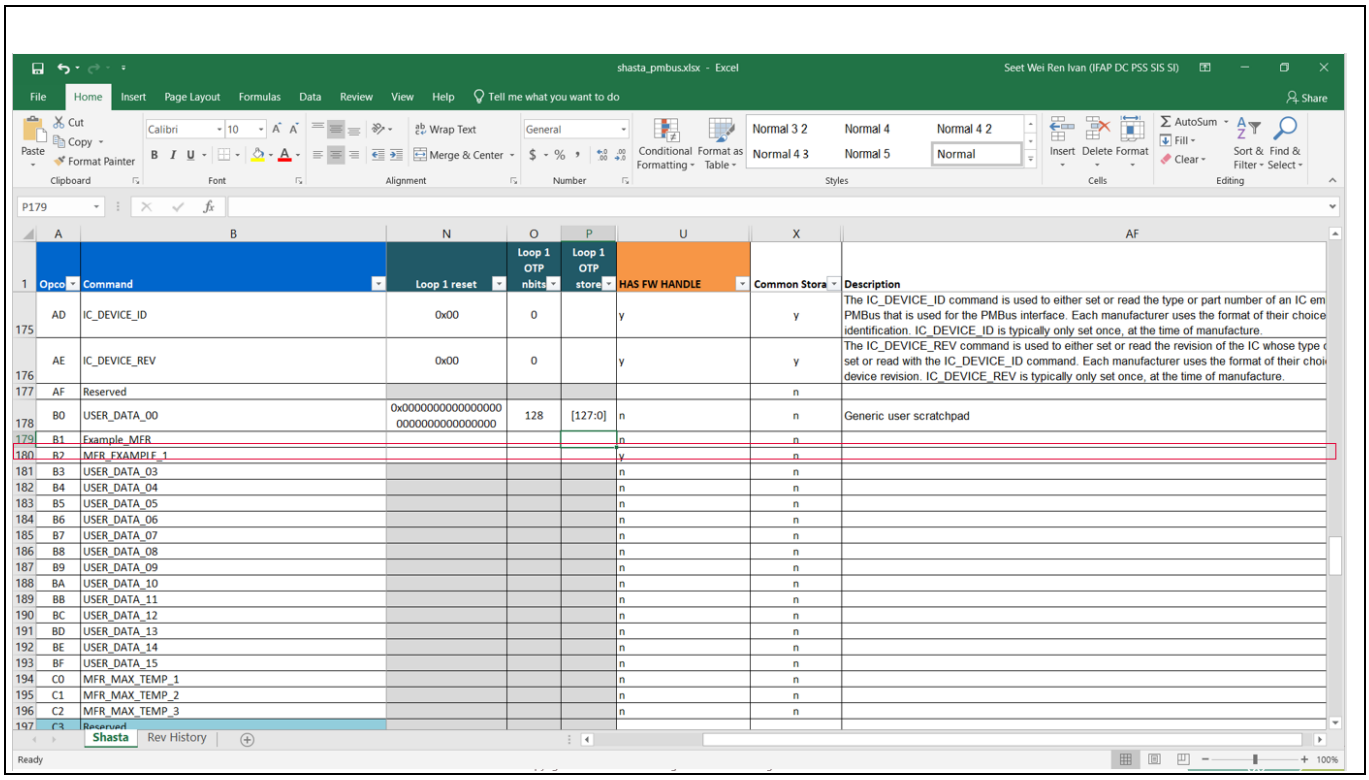


Figure 13 shasta_pmbus.xlsx

In the project workspace click “clean” followed by “pmbus”, to generate the PMBus setting to c code.

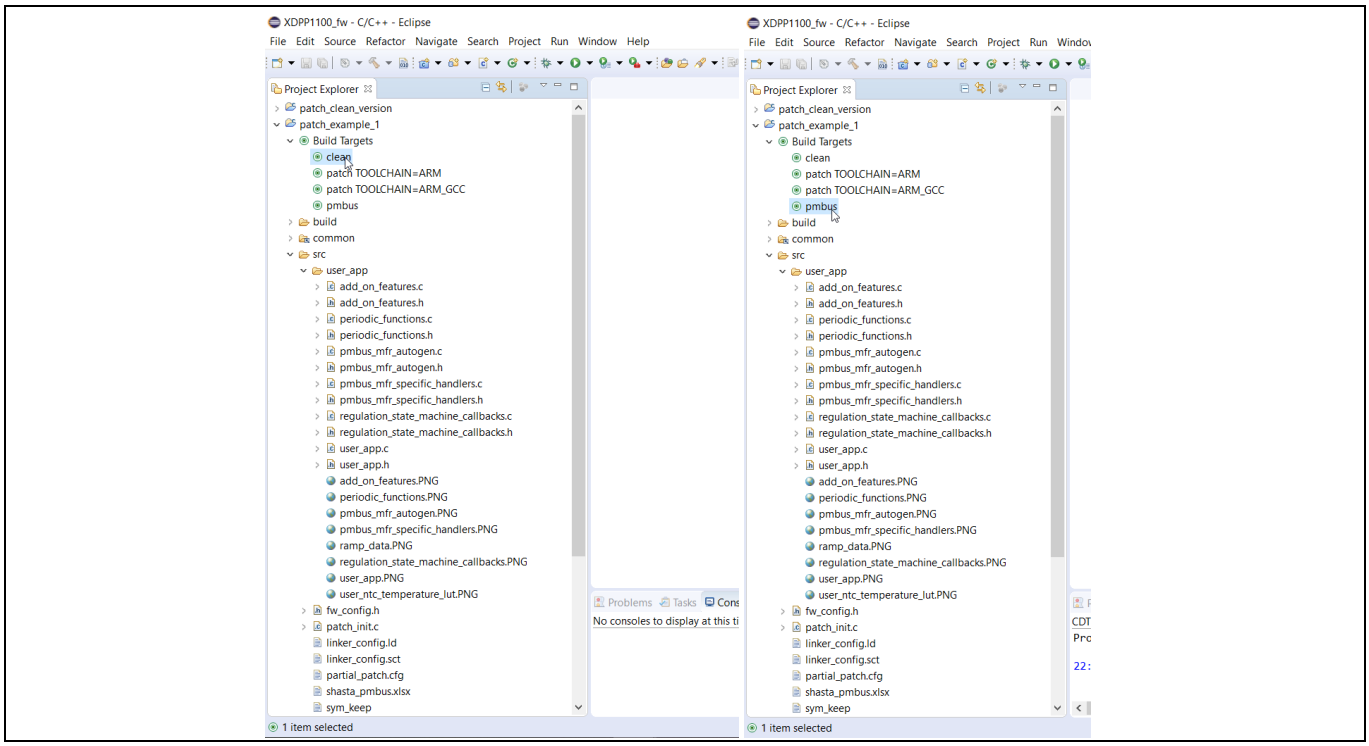


Figure 14 Eclipse environment – generate new MFR commands set in shasta_pmbus.xlsx

The PMBus MFR commands code will be generated and can be found in pmbus_mfr_autogen.c and pmbus_mfr_autogen.h.

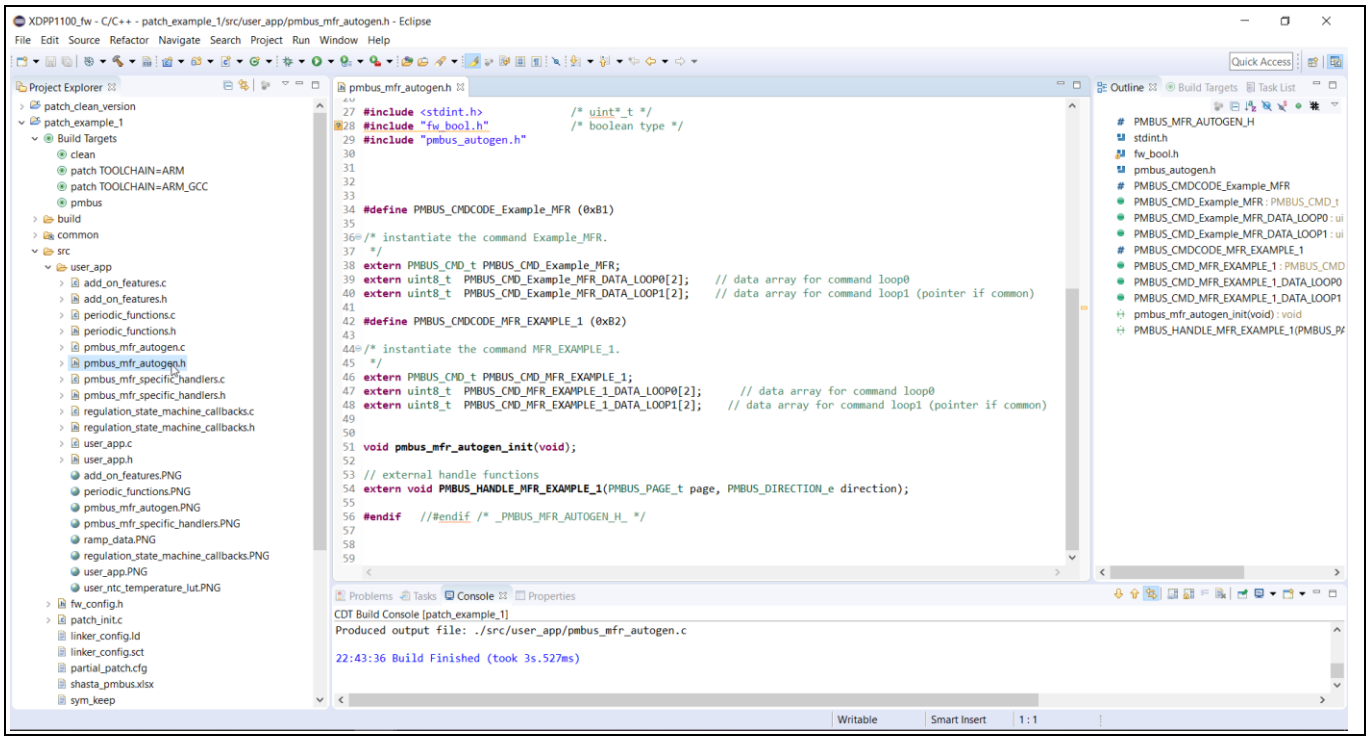


Figure 15 pmbus_mfr_autogen.h

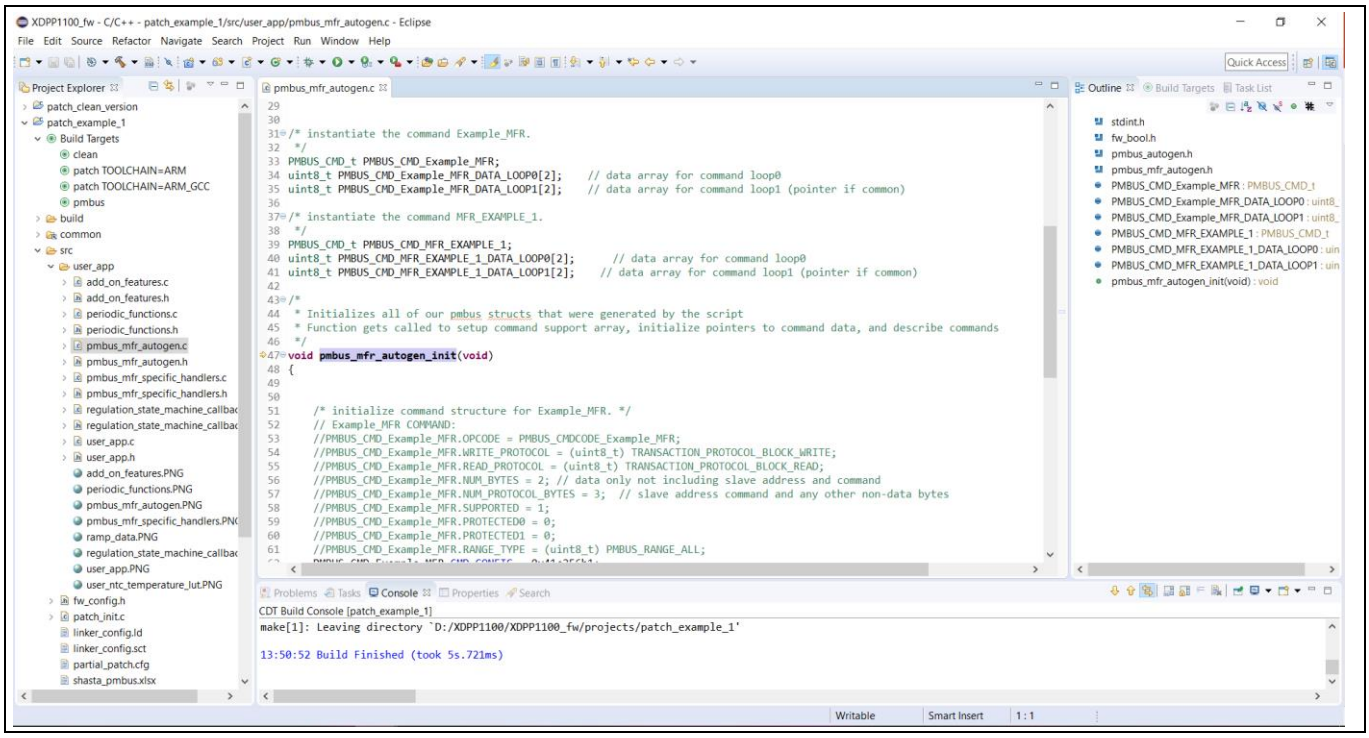


Figure 16 pmbus_mfr_autogen.c

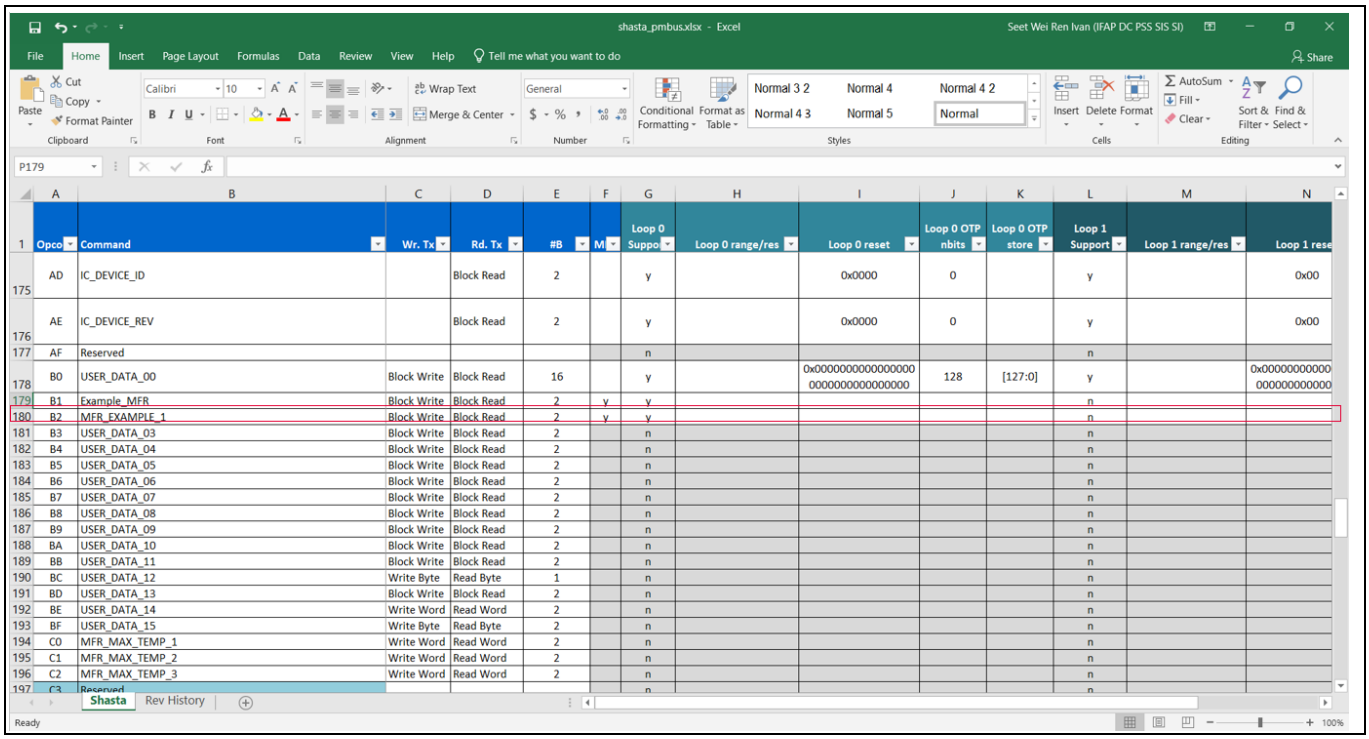
3 PMBus MFR command control GPIO output

In this example a PMBus MFR command will be created and the PWM 3 pin set as GPIO function setting output high or low.

To set up a new FW patch project patch_example_2, refer to **Section 2.1** Create new project.

3.1 PMBus MFR command for GPIO control

To create a new PMBus MFR command 0xB2, refer to **Section 2.2** Add a new PMBus MFR command.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Opco	Command	Wr. Tx	Rd. Tx	#B	M	Loop 0 Suppo	Loop 0 range/res	Loop 0 reset	Loop 0 OTP nbits	Loop 0 OTP store	Loop 1 Support	Loop 1 range/res	Loop 1 rese
175	AD	IC_DEVICE_ID		Block Read	2		y		0x0000	0		y		0x00
176	AE	IC_DEVICE_REV		Block Read	2		y		0x0000	0		y		0x00
177	AF	Reserved					n					n		
178	B0	USER_DATA_00	Block Write	Block Read	16		y		0x00000000000000000000000000000000	128	[127:0]	y		0x00000000000000000000000000000000
179	B1	Example_MFR	Block Write	Block Read	2	y	y					n		
180	B2	MFR_EXAMPLE_1	Block Write	Block Read	2	y	y					n		
181	B3	USER_DATA_03	Block Write	Block Read	2		n					n		
182	B4	USER_DATA_04	Block Write	Block Read	2		n					n		
183	B5	USER_DATA_05	Block Write	Block Read	2		n					n		
184	B6	USER_DATA_06	Block Write	Block Read	2		n					n		
185	B7	USER_DATA_07	Block Write	Block Read	2		n					n		
186	B8	USER_DATA_08	Block Write	Block Read	2		n					n		
187	B9	USER_DATA_09	Block Write	Block Read	2		n					n		
188	BA	USER_DATA_10	Block Write	Block Read	2		n					n		
189	BB	USER_DATA_11	Block Write	Block Read	2		n					n		
190	BC	USER_DATA_12	Write Byte	Read Byte	1		n					n		
191	BD	USER_DATA_13	Block Write	Block Read	2		n					n		
192	BE	USER_DATA_14	Write Word	Read Word	2		n					n		
193	BF	USER_DATA_15	Write Byte	Read Byte	2		n					n		
194	C0	MFR_MAX_TEMP_1	Write Word	Read Word	2		n					n		
195	C1	MFR_MAX_TEMP_2	Write Word	Read Word	2		n					n		
196	C2	MFR_MAX_TEMP_3	Write Word	Read Word	2		n					n		
197	C3	Reserved					n					n		

Figure 17 shasta_pmbus.xlsx

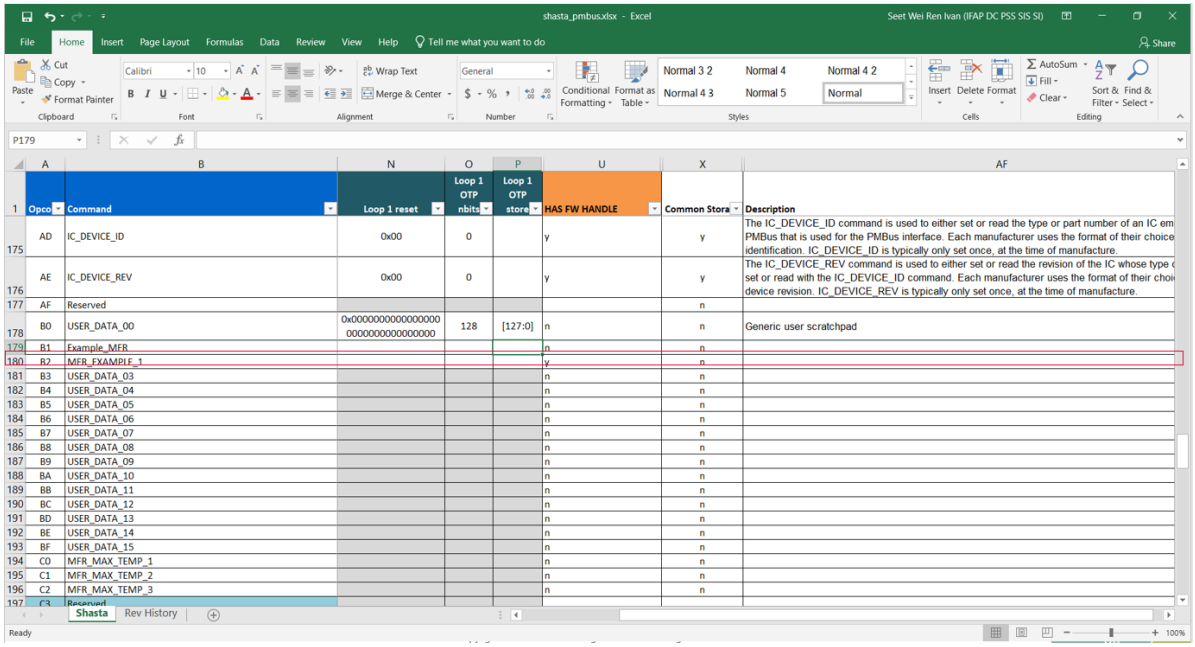


Figure 18 shasta_pmbus.xlsx

3.2 Modify source code

This section explains source changes to enable PMBus MFR command 0xB1 to enable the SYNC pin as a GPIO output and set it as output high or low.

Table 1 Additions/Modifications to create a custom V_{IN} telemetry IRQ

Filename	Function name
pmbus_mfr_specific.c	void PMBUS_HANDLE_MFR_EXAMPLE_1(PMBUS_PAGE_t page, PMBUS_DIRECTION_e direction)
user_app.c	void user_drv_init(void)

3.2.1 PMBus command handler

Open the the pmbus_mfr_specific_handlers.c and insert the PMBus handler “void PMBUS_HANDLER_MFR_EXAMPE_1()” found in pmbus_mfr_autogen.h. Add the source code according to Code Listing 1.

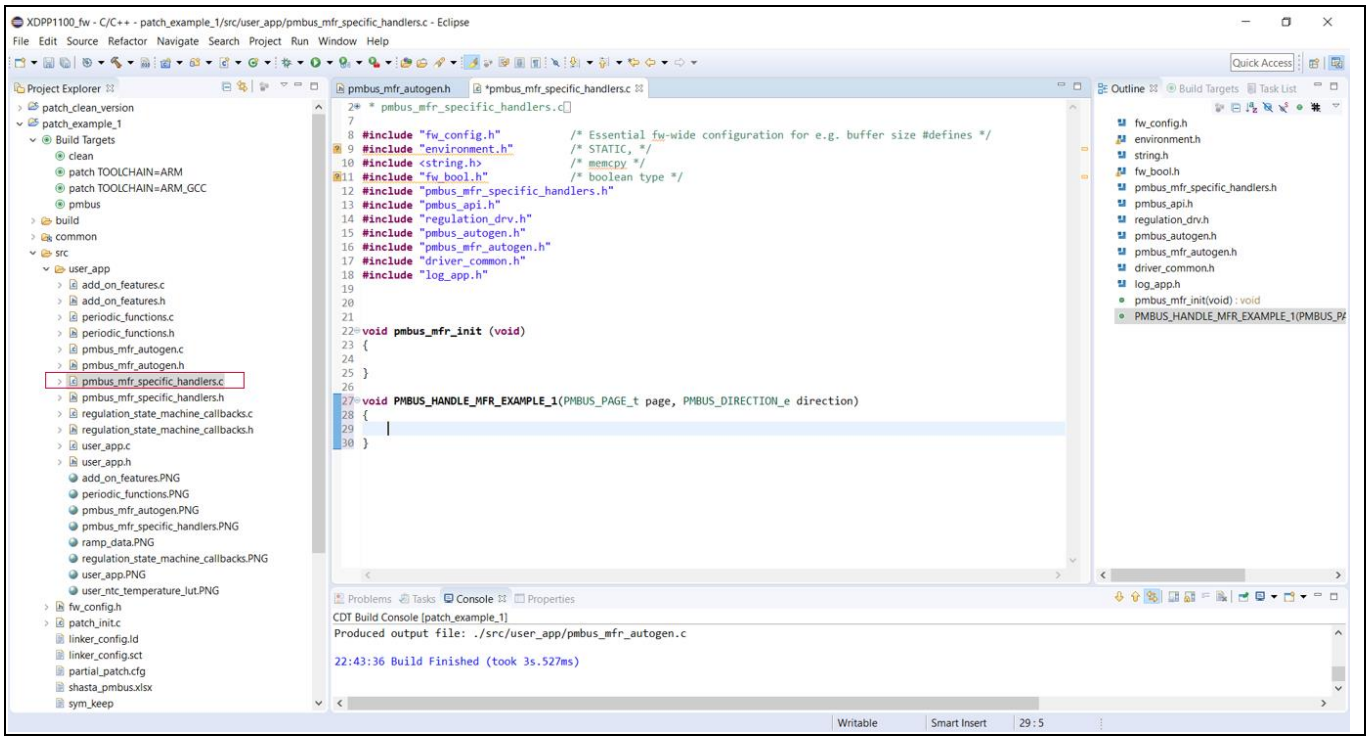


Figure 19 pmbus_mfr_specific_handlers.c

Code Listing 1 pmbus_mfr_specific_handlers.c

```
#include "gpio_drv.h"
#include "shasta_hal_common.h"
void PMBUS_HANDLE_MFR_EXAMPLE_1(PMBUS_PAGE_t page, PMBUS_DIRECTION_e
direction)
{
  uint8_t sync_bit_index = 7; //bit7
  uint8_t pin_setting = PMBUS_Get_Command_Dword(page,
  PMBUS_CMDCODE_MFR_EXAMPLE_1);
  if ((direction == PMBUS_WRITE))
  {
    COMMON_MP_FUNC2__SYNC_FUNC__SET(1); //Set SYNC Pin to GPIO function
    COMMON_MP_FUNC2__SYNC_PD__SET(0); //pulldown disabled
    COMMON_MP_FUNC2__SYNC_PU_N__SET(1);//pullup disabled
    COMMON_MP_FUNC2__SYNC_PPEN__SET(1);//CMOS output

    GpioBit_SetCnfg (GPIO_0, sync_bit_index); // select pin to configure
    GpioBit_SetDir (GPIO_0, sync_bit_index, DIRECTION_OUTPUT);
    // pin as output
    if (pin_setting == 1)
    {
      GpioBit_DatWr(GPIO_0, (uint8_t)sync_bit_index, 0xff);
    }
    else if(pin_setting == 0)
    {
      GpioBit_DatWr(GPIO_0, (uint8_t)sync_bit_index, 0x00);
    }
  }
}
```

3.2.2 User app

Open the the user_app.c and insert source code according to **Code Listing 2** in void user_drv_init(void) to run the FW patch in RAM.

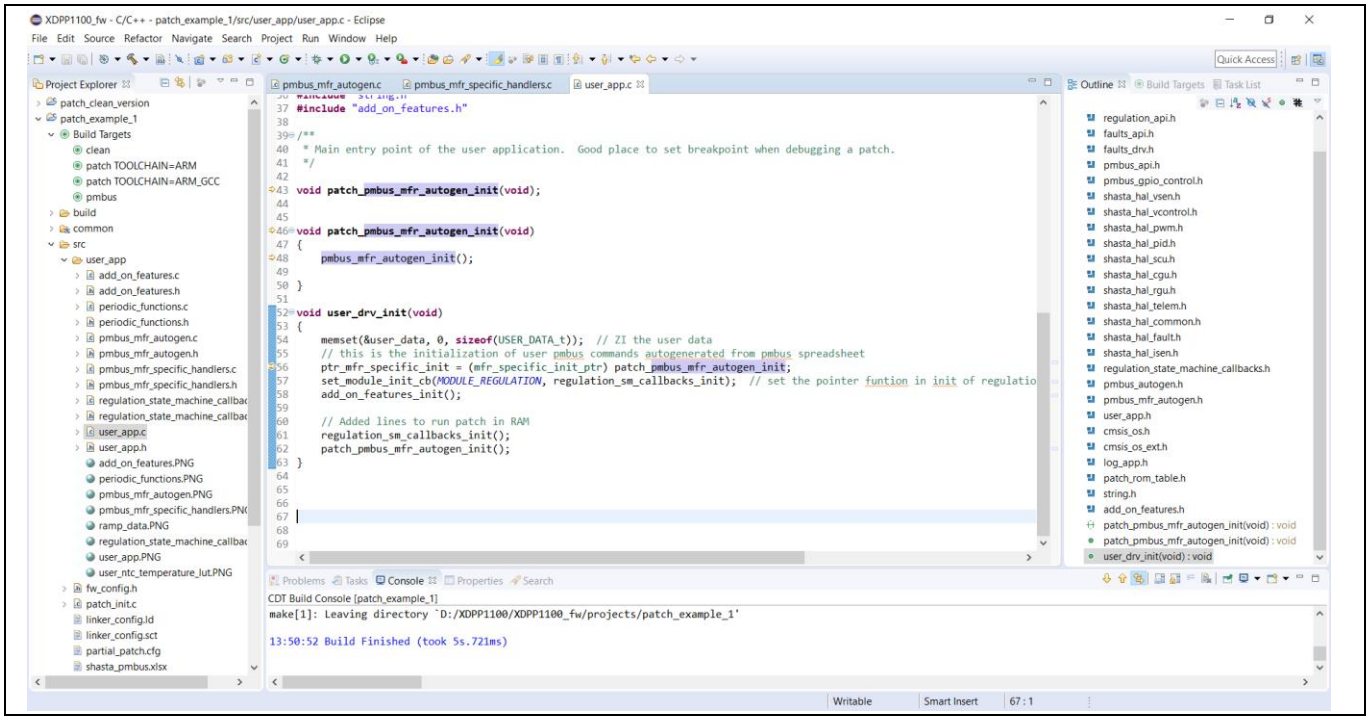


Figure 20 user_app.c

Code Listing 2 user_app.c

```
void user_drv_init(void)
{
  memset(&user_data, 0, sizeof(USER_DATA_t));  // ZI the user data
  // 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();

  // Added lines to run patch in RAM
    regulation_sm_callbacks_init();
    patch_pmbus_mfr_autogen_init();
}
```

3.3 Compile and program to XDPP1100 RAM/OTP

3.3.1 Compile source code

After editing the source as explained in **Section 3.2**, compile the code. In the project workspace click “clean” followed by “patch TOOLCHAIN=ARM_GCC” to compile the FW patch code.

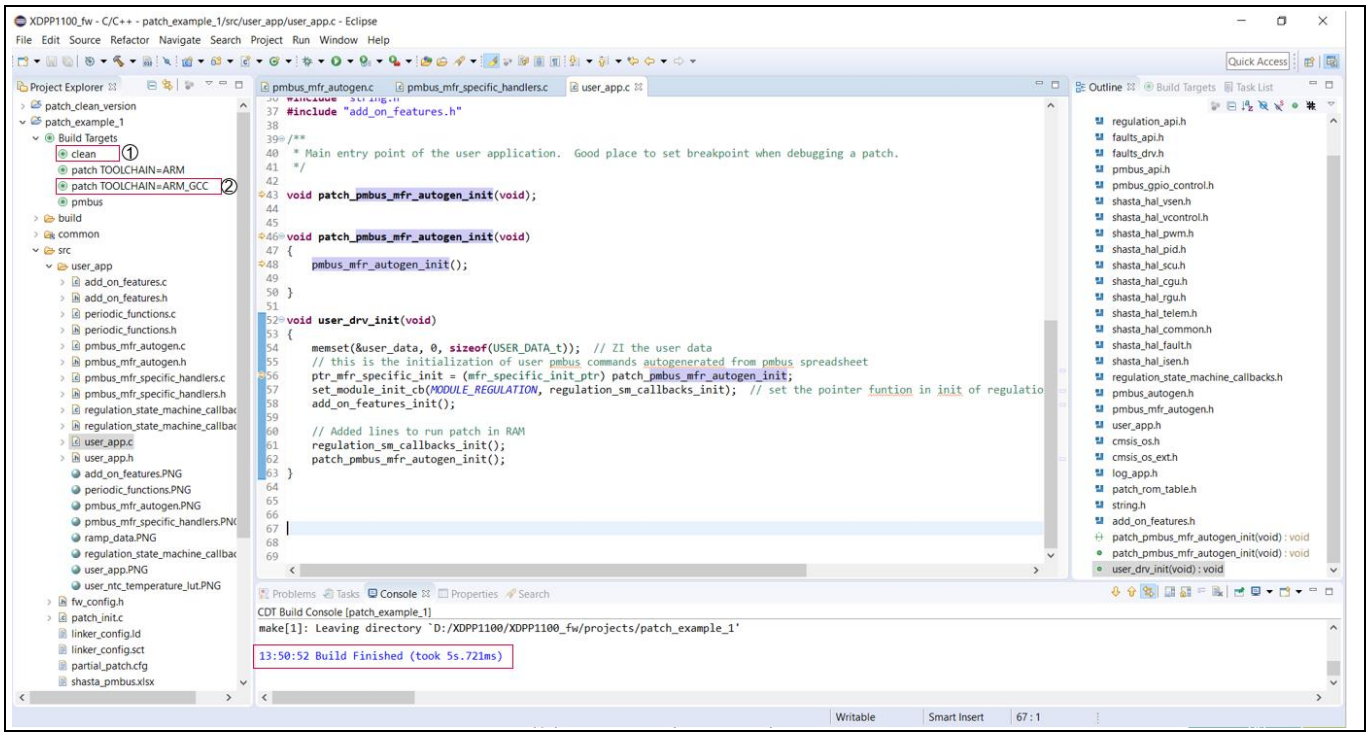


Figure 21 Eclipse environment – compile source code

3.3.2 Connect to evaluation board

Connect the XDPP1100 daughter board/demo board I²C port to the USB700A dongle to connect to the PC.

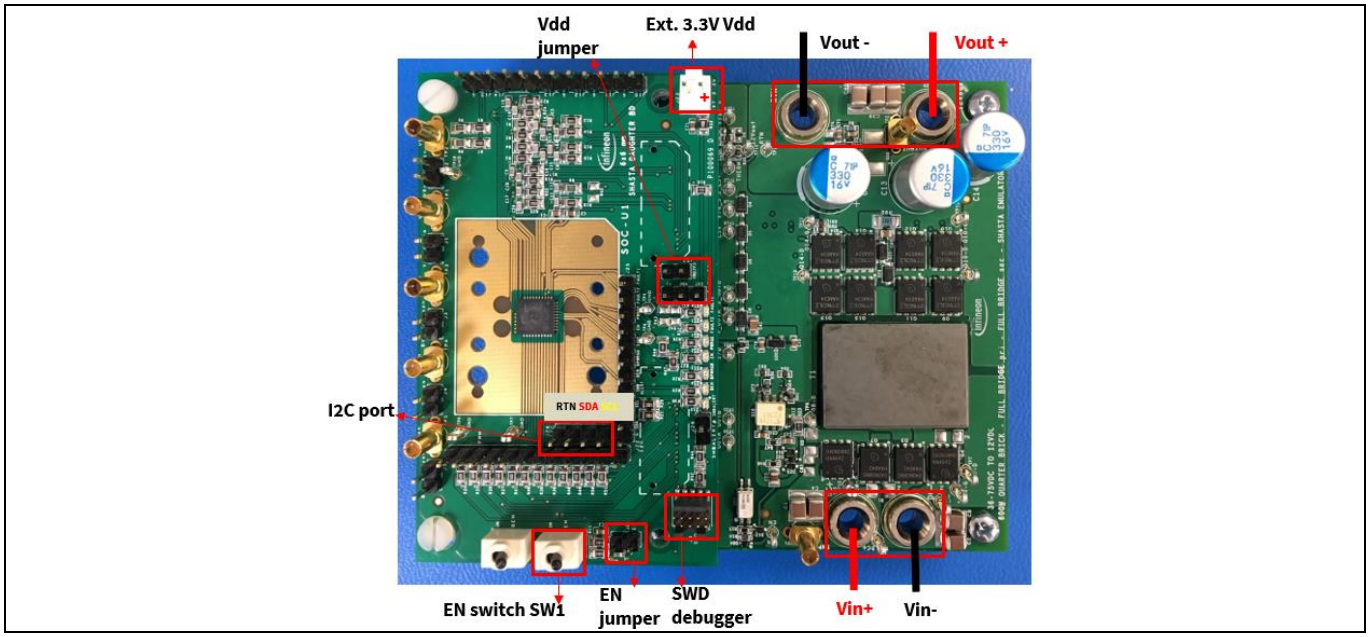


Figure 22 XDPP1100 evaluation board

3.3.3 XDPP1100 GUI operation

Open the XDPP1100 graphical user interface (GUI) in the XDPP1100_GUI folder.

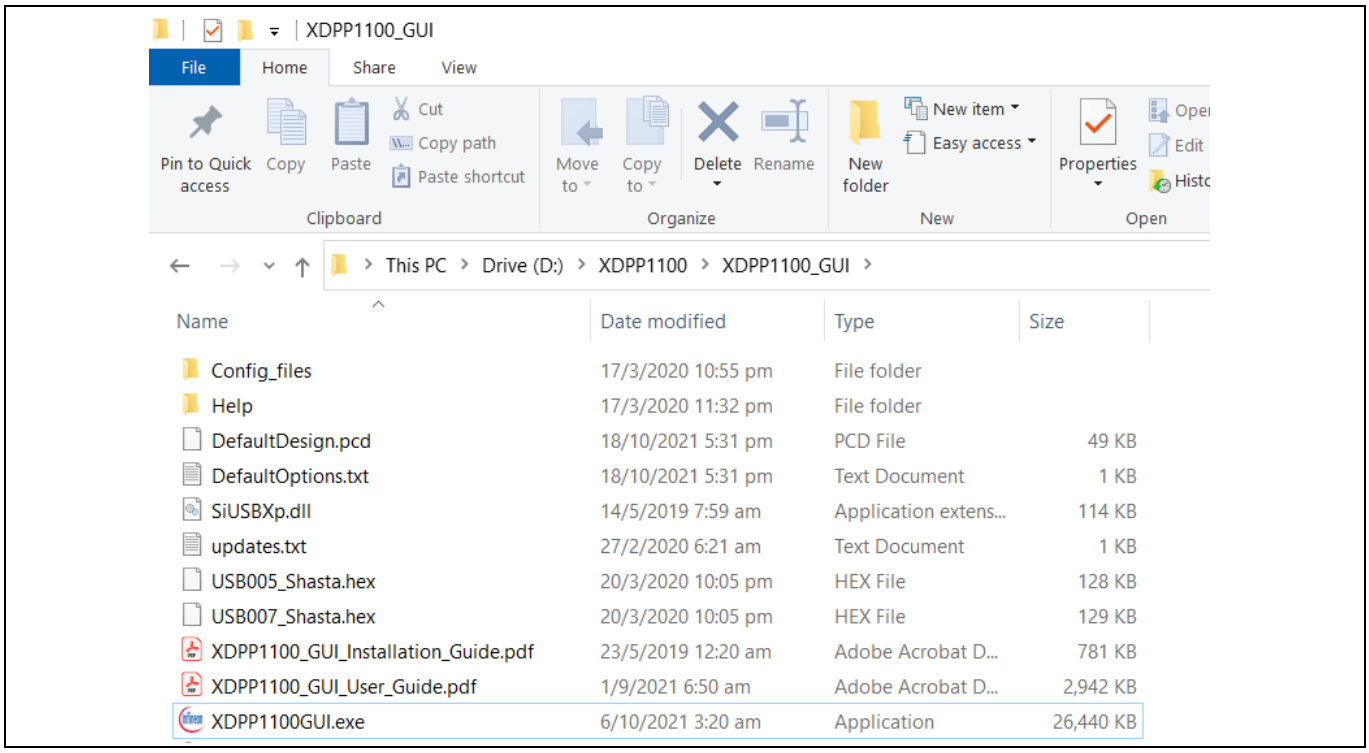


Figure 23 XDPP1100 GUI directory

Open an existing design file or create a new design according to your requirements.

In this example an existing design file for 24 pins is used.

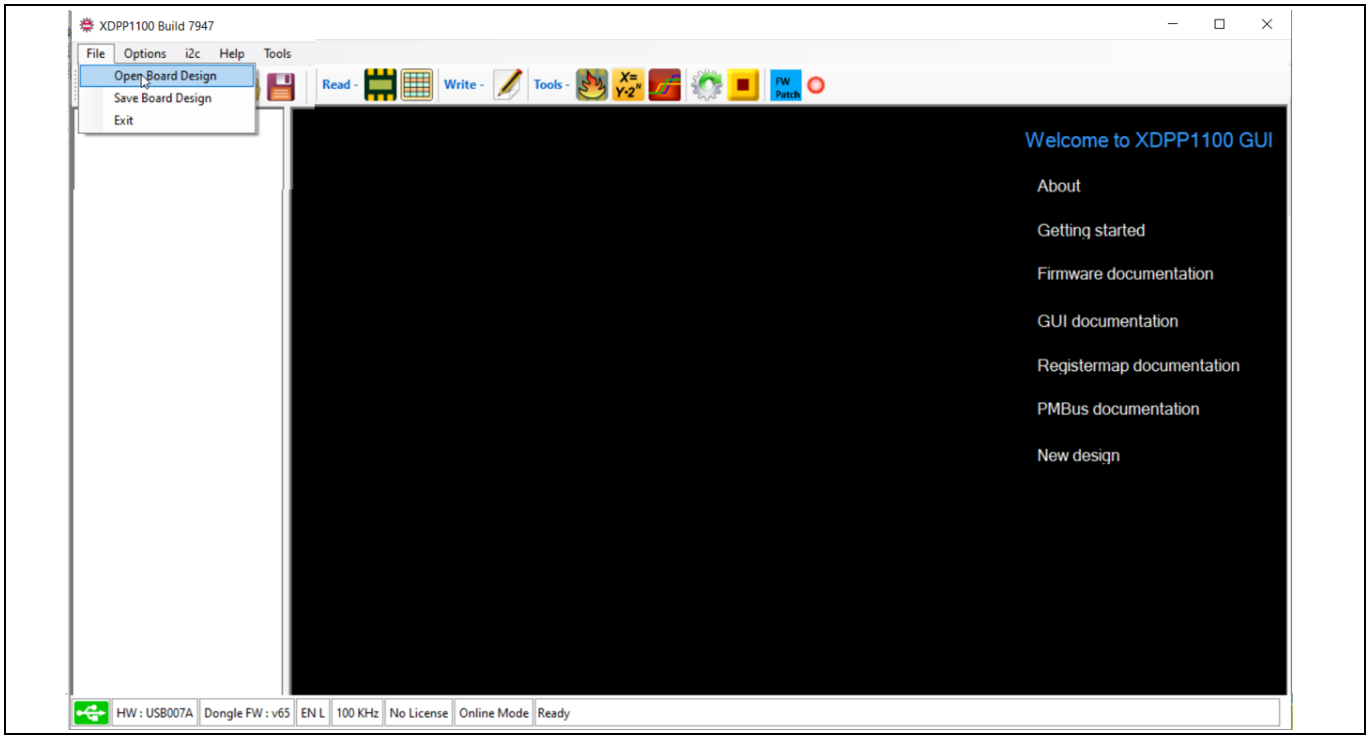


Figure 24 XDPP1100 GUI

Select “EVAL_XDPP1100_600W_VMC_4x_config.pcd” in the Config_files folder.

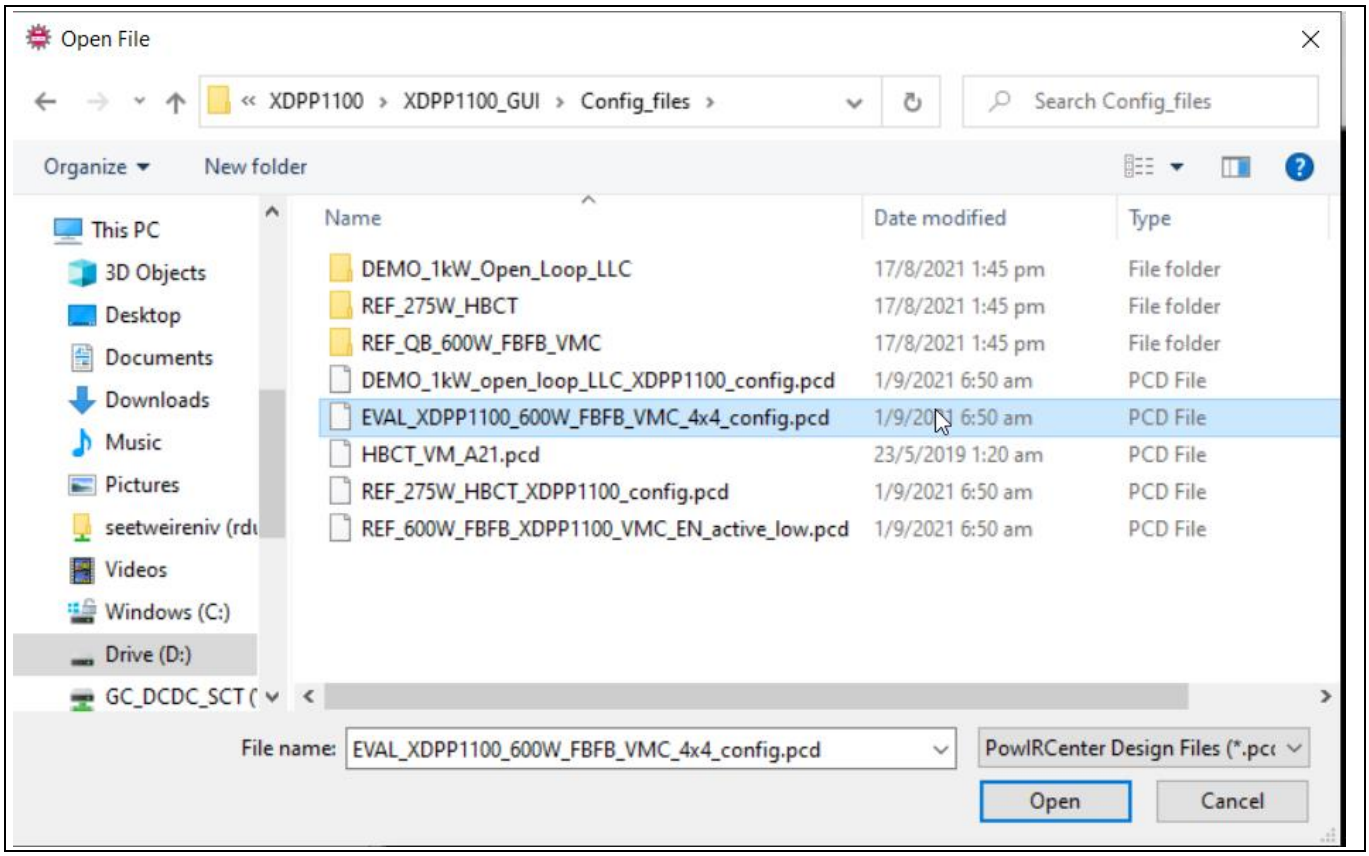


Figure 25 XDPP1100 GUI open config file

Click “Write to Device 0x10” when prompted by the GUI and select “Enable Telemetry Update” and “Show Floating Status”.

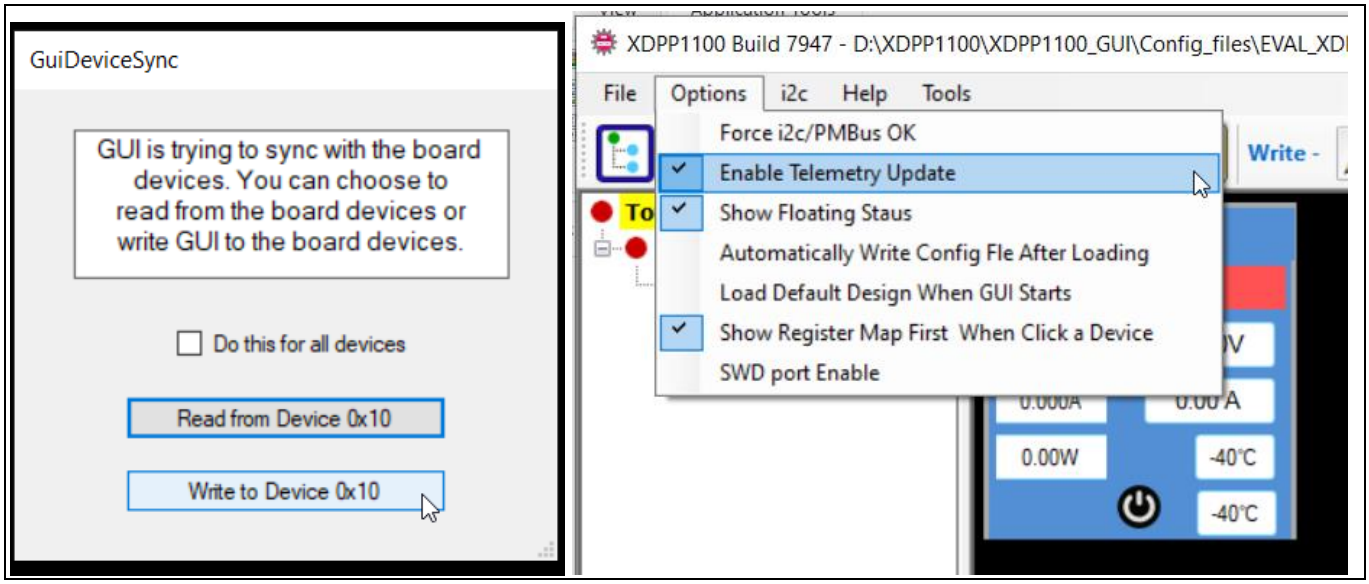


Figure 26 XDPP1100 GUI – GuiDeviceSync window and option selection

When using only the daughter board without the power stage board and using the EVAL_XDPP1100_600W_VMC_4x_config.pcd, follow the steps below to get the board running.

The steps shown from **Figure 27** to **Figure 30** can be ignored if using full power stage with 48 V V_{IN} supply.

Find command 0xDC and set the temperature as shown in **Figure 27**.

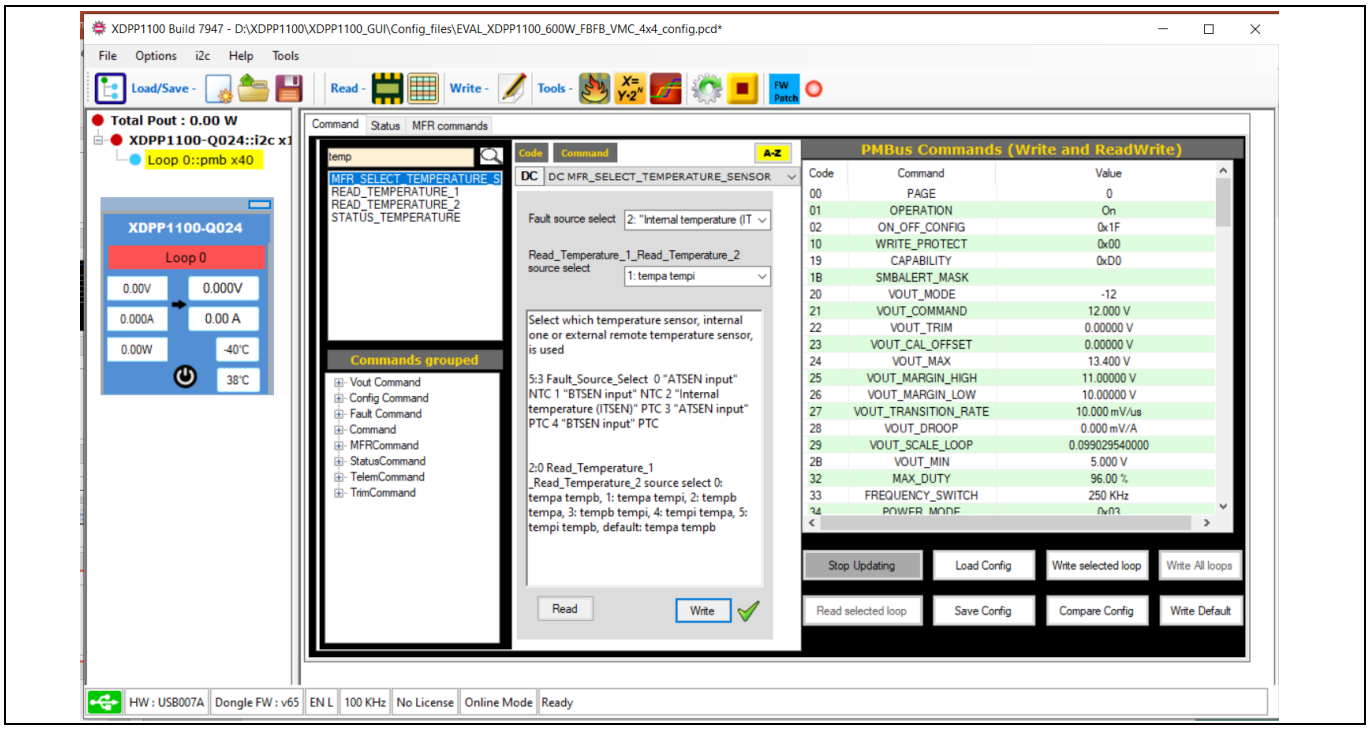


Figure 27 XDPP1100 GUI – PMBus commands settings

Go to Design Tools, and select “5. Basic Configuration” to configure the V_{IN} telemetry to set tlm0_vin_src_sel to tlm0_vin_force or TS ADC V_{IN}. In **Figure 28** TS ADC V_{IN} is used and the vin_pwl_slope value is changed to achieve 48 V as the V_{IN} value; the user can check the value based on the telemetry update window.

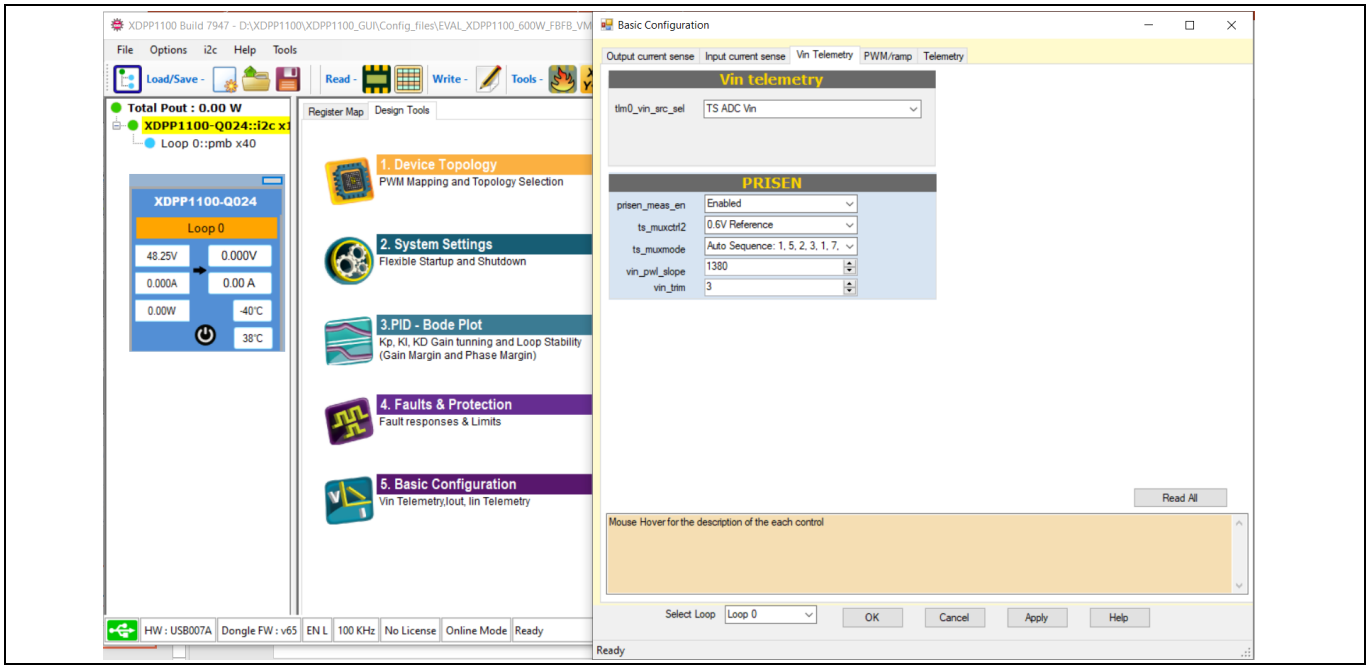


Figure 28 XDPP1100 GUI – design tool settings

Click “EN L/EN H“ to start the board regulation. Check the V_{OUT} value in the telemetry window to see whether the V_{OUT} is below VOUT_COMMAND. If not, modify the PMBus command 0x29 value to achieve a low V_{OUT} reading.

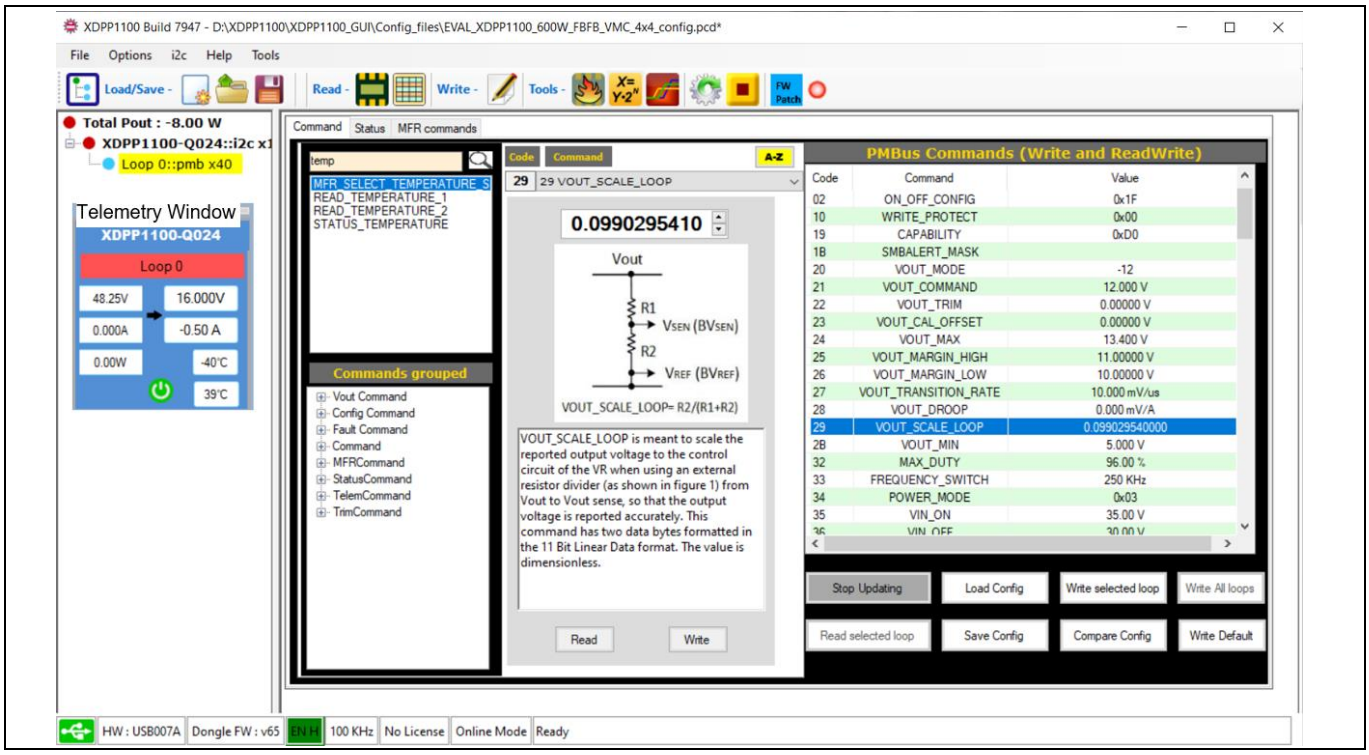


Figure 29 XDPP1100 GUI – PMBus commands settings

Once the value reaches within the VOUT_COMMAND range, click “EN H/EN L” to stop regulating.

These steps ensure you can run the board without a power stage to test the pulse-width modulation (PWM) signal, regulation state machine, etc.

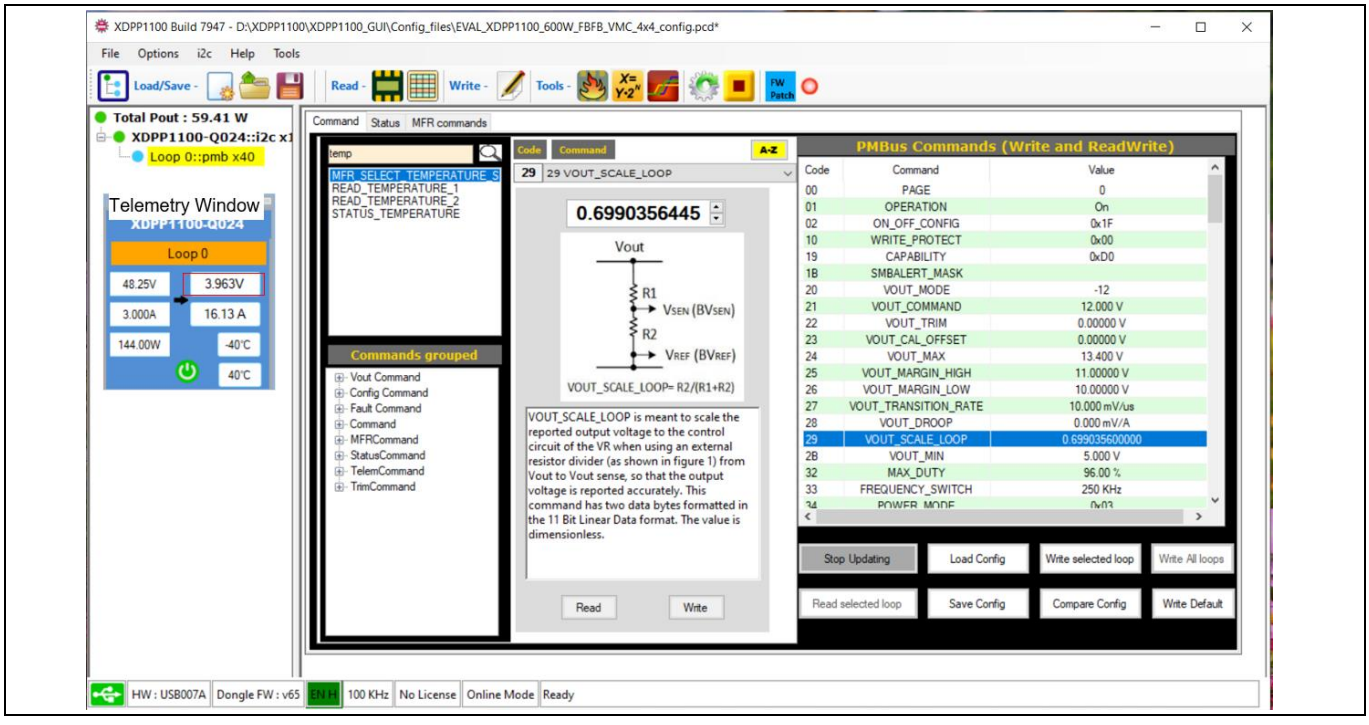


Figure 30 XDPP1100 GUI – PMBus commands settings

3.3.4 XDPP1100 GUI patch download

Select the “FW Patch” button to open the window for FW Patch download.

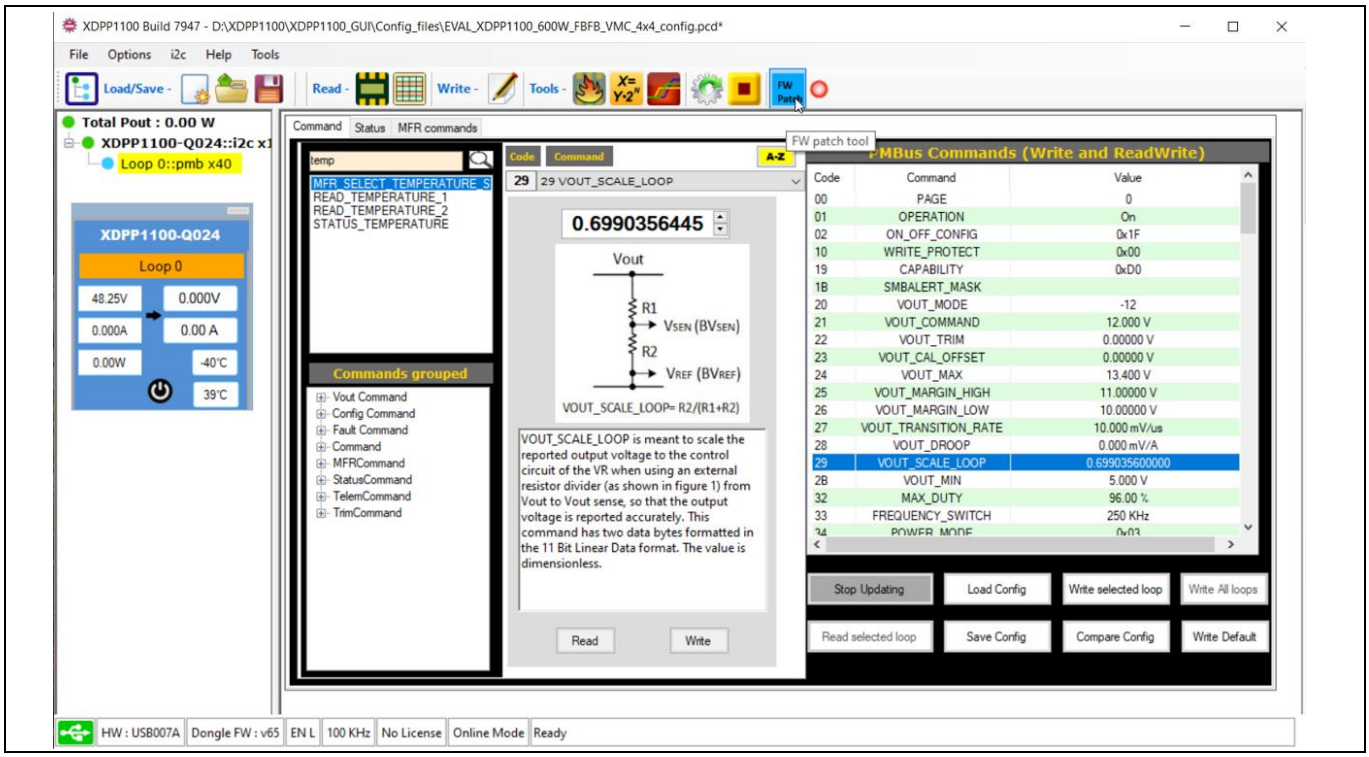


Figure 31 XDPP1100 GUI – select FW Patch button

Click the “FW Patch” tab and click the “Load OTP Patch File” to select the bin file to download.

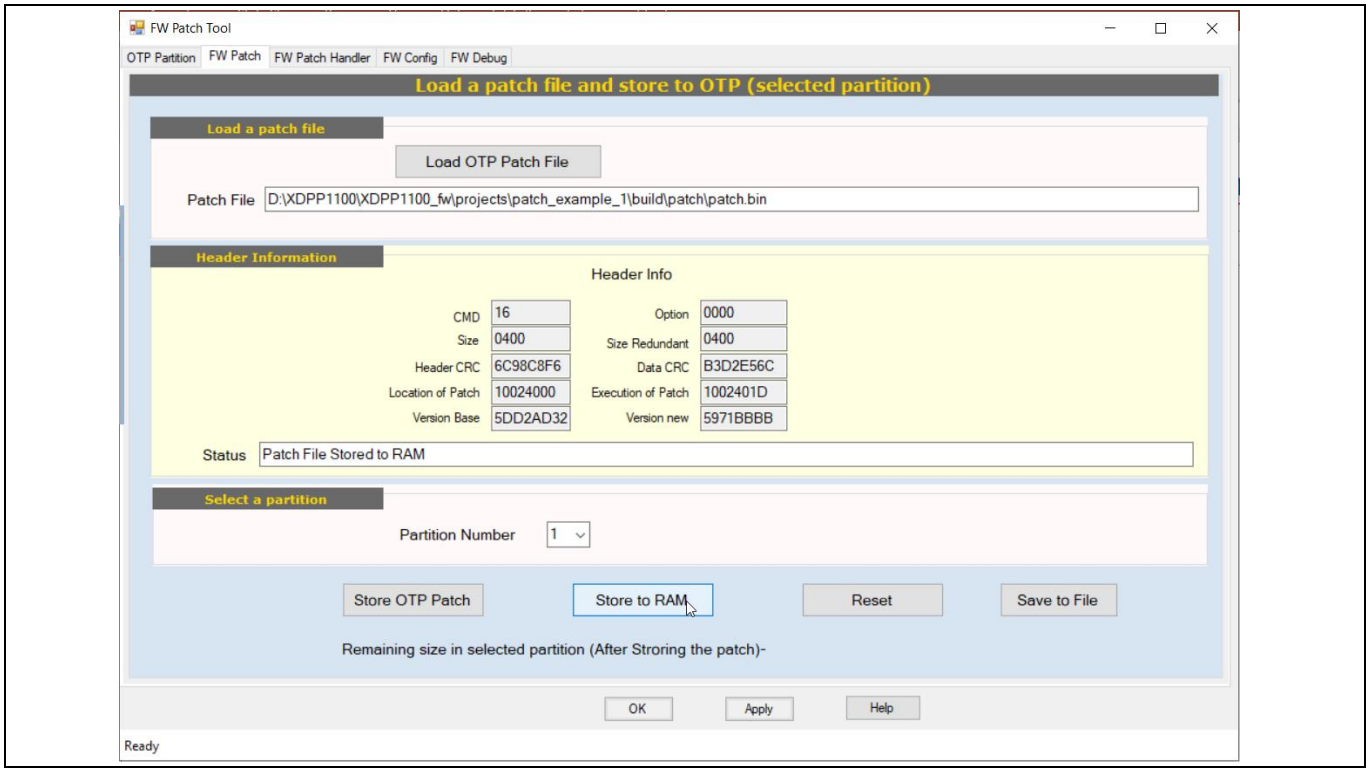


Figure 32 XDPP1100 GUI – FW Patch tool window

Nagivate to the folder “\patch_example_1\build\patch” and select “patch.bin”.

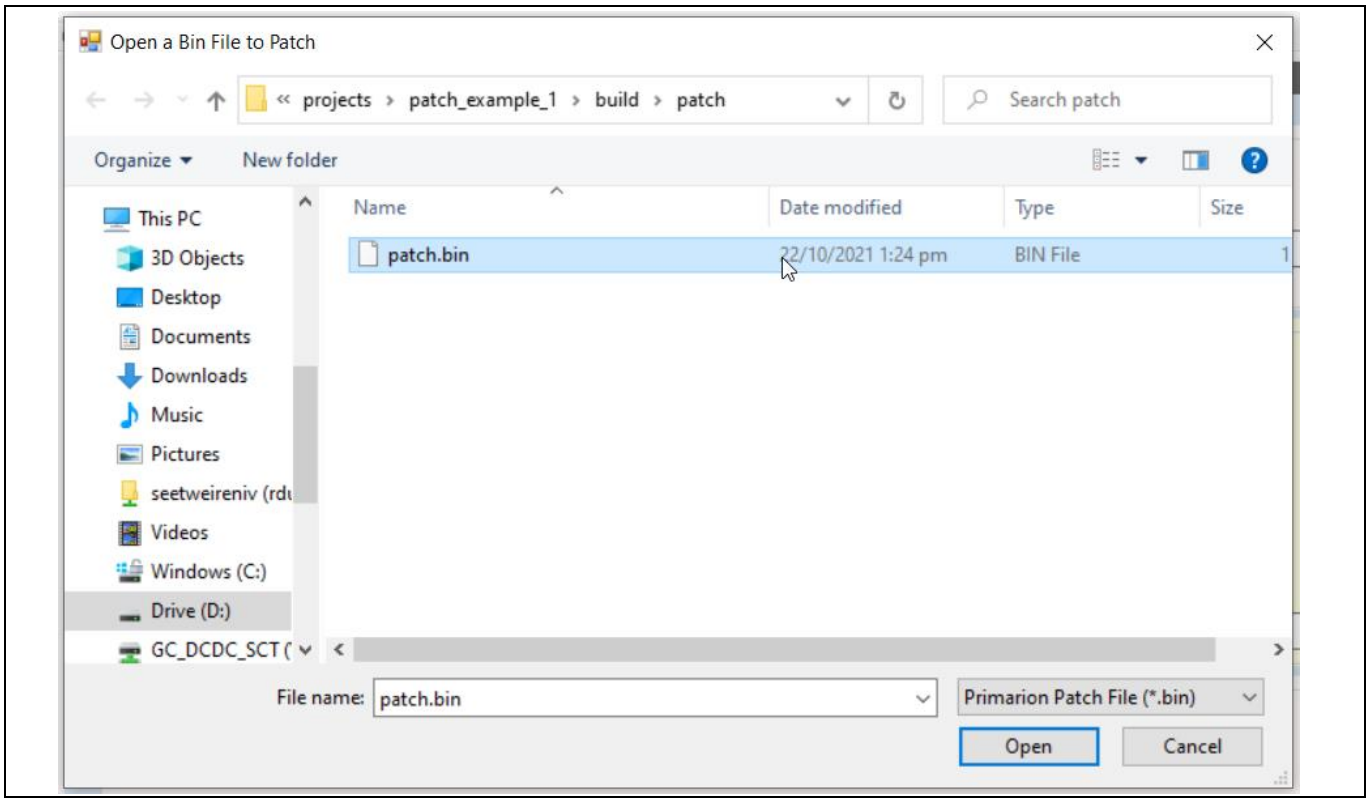


Figure 33 XDPP1100 GUI – FW Patch tool, select patch.bin

In this example the patch is stored to RAM, therefore select “Store to RAM”. Once done, the word “Ready” will display at bottom left.

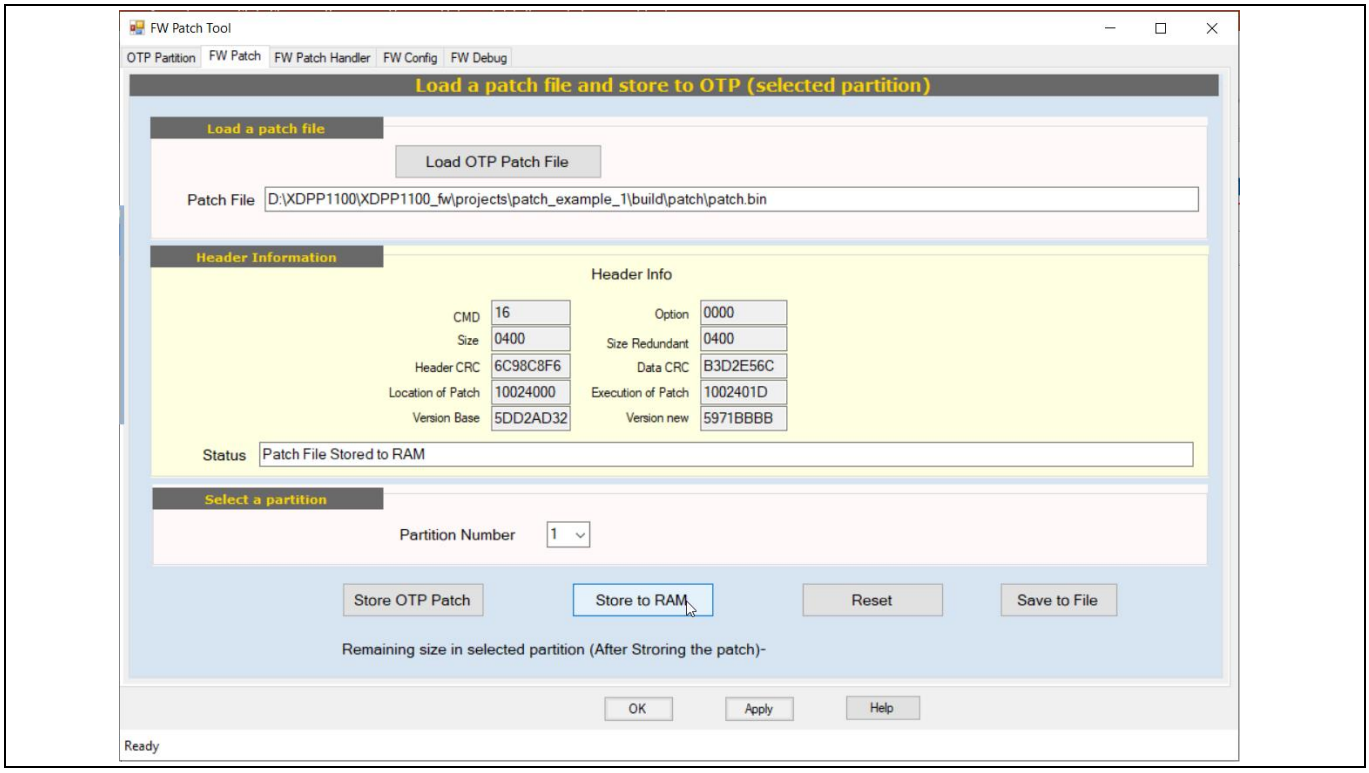


Figure 34 XDPP1100 GUI – FW Patch tool window, select “Store to RAM”

Click the “FW Patch Handler” tab and click “Find Active patch” to verify successful download.

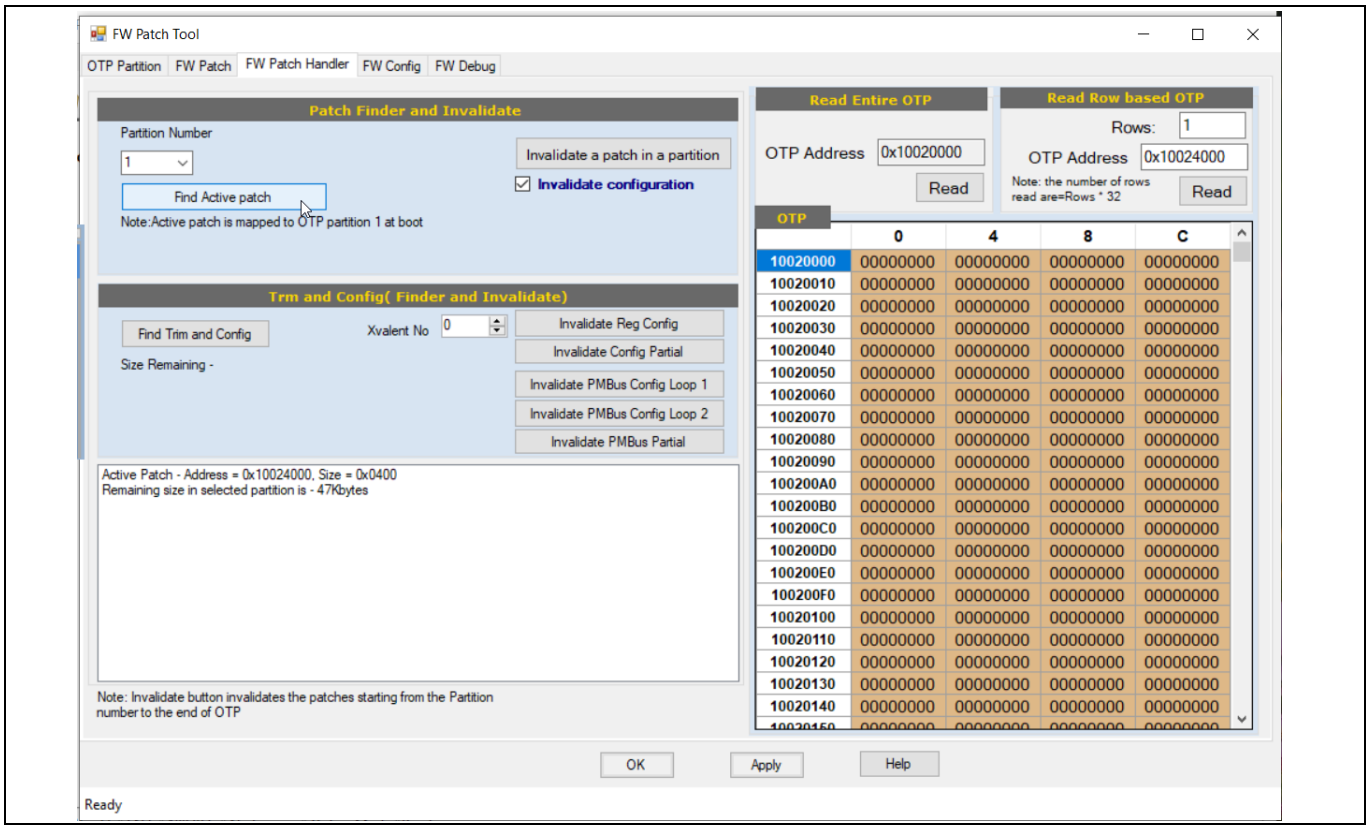


Figure 35 XDPP1100 GUI – FW Patch tool, check active patch

3.3.5 Running example 1 patch

Click the “MFR commands” tab and click the “Load PMBus Spread sheet” button to load the spreadsheet with modified MFR commands.

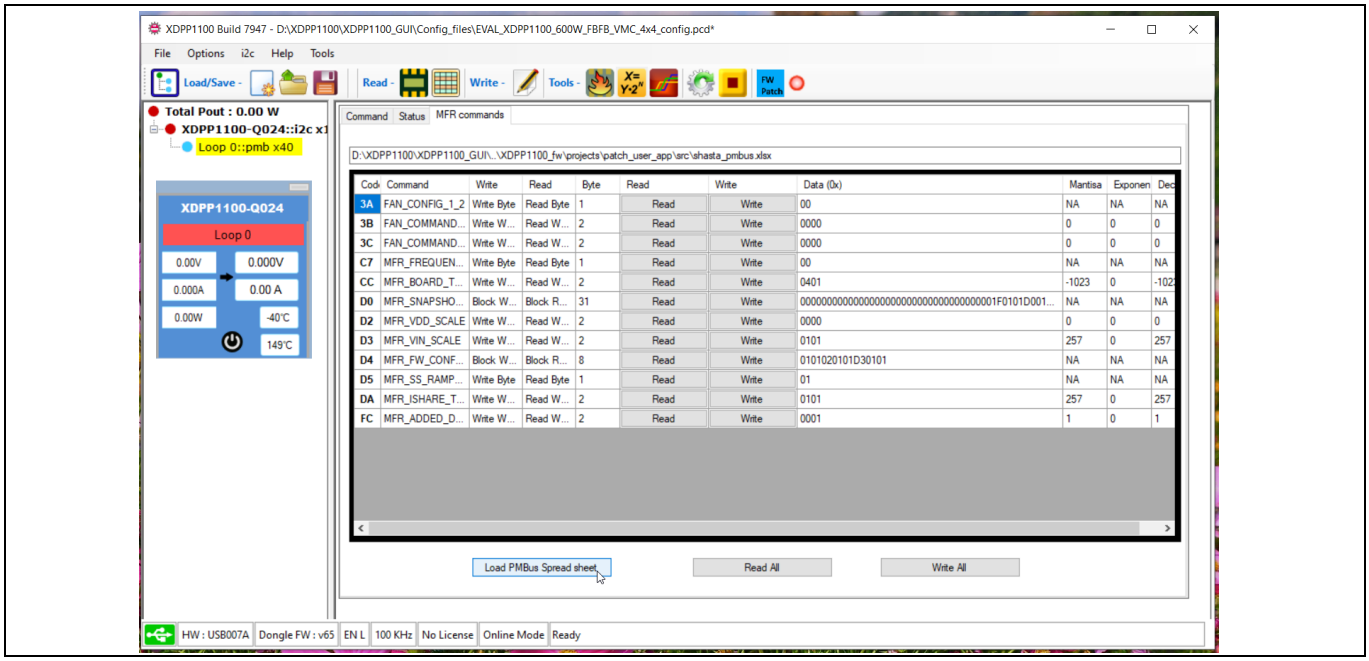


Figure 36 XDPP1100 GUI – MFR commands tab

Select shasta_pmbus.xlsx in \project\patch_example_1\src.

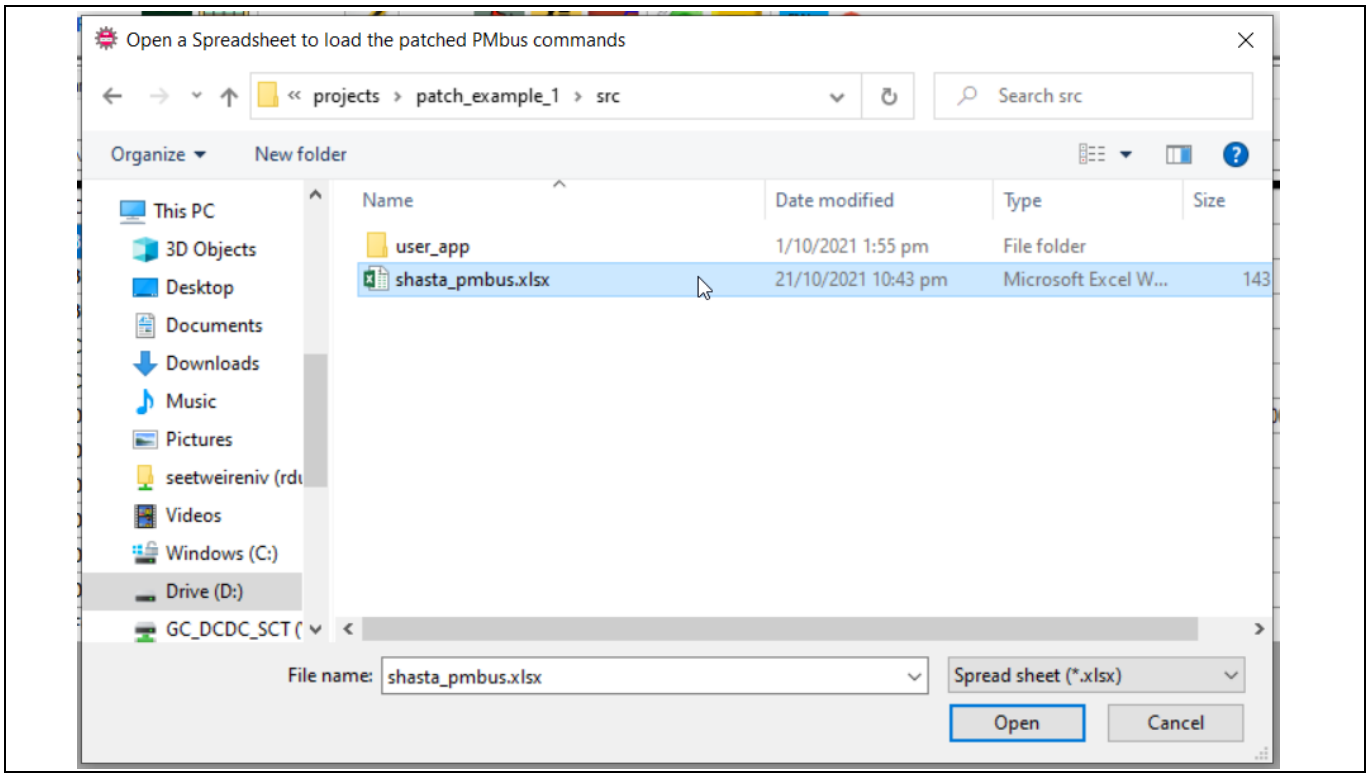


Figure 37 XDPP1100 GUI – open latest MFR commands spreadsheet

The spreadsheet is loaded as shown in **Figure 38**. Click the “Write” button in the top panel to write all register map values and PMBus commands values into RAM.

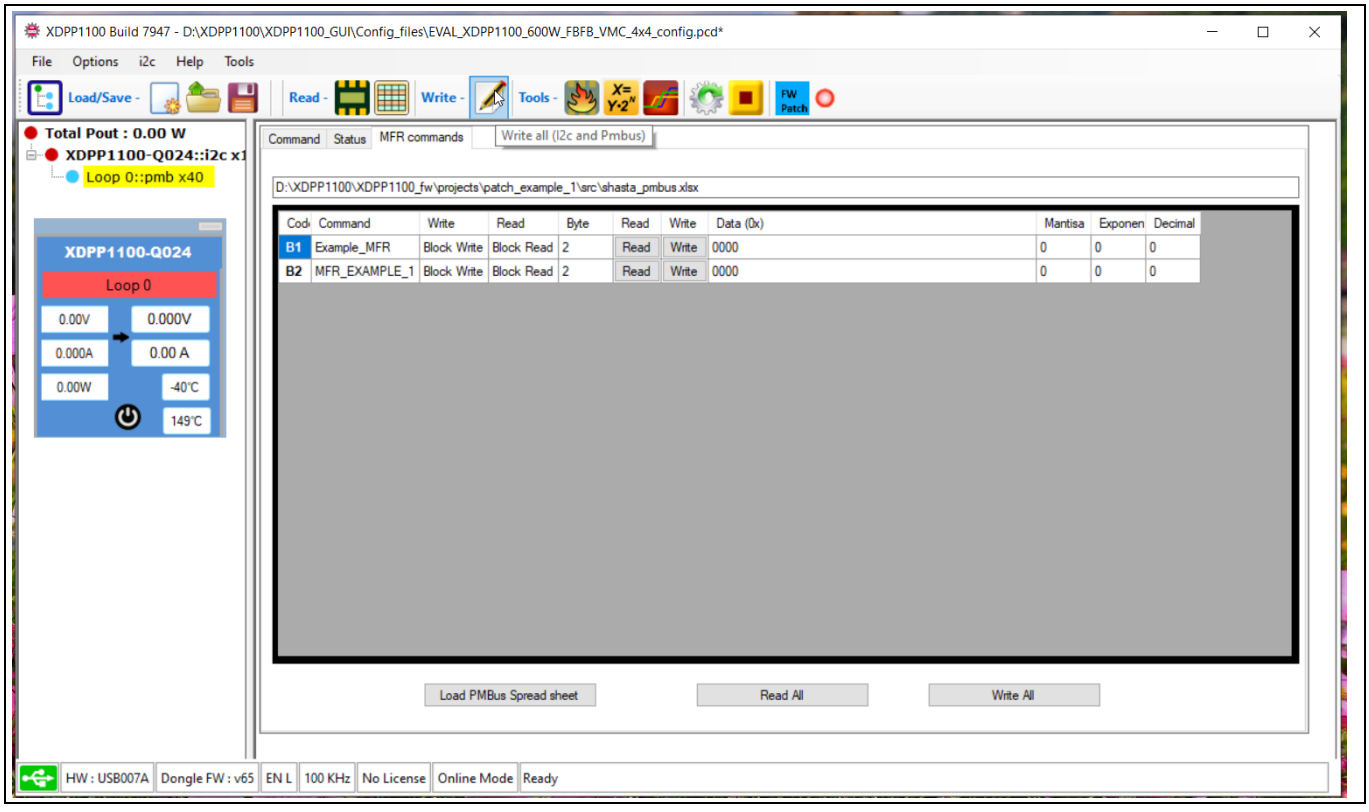


Figure 38 XDPP1100 GUI – MFR commands tab with latest spreadsheet

Right-click “XDPP1100-Q024::i2c” in the left panel and press “Clear Fault” to clear all faults after writing to RAM.

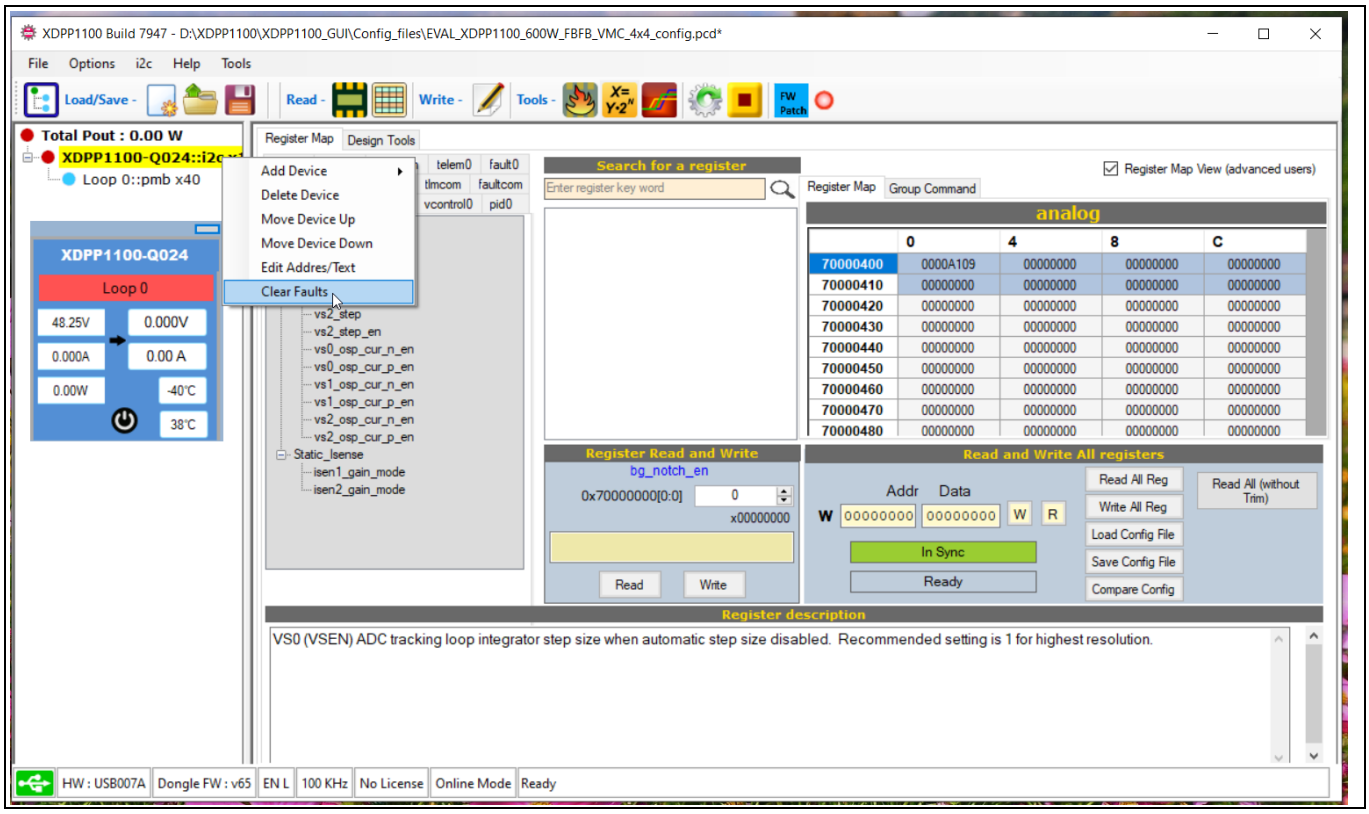


Figure 39 XDPP1100 GUI – clear faults

Proceed to the MFR commands and change the value of 0xB2 accordingly, then click “Write”. The SYNC pin output will change according to the setting of 0xB2. Click “EN H/EN L” to start the PWM regulating.

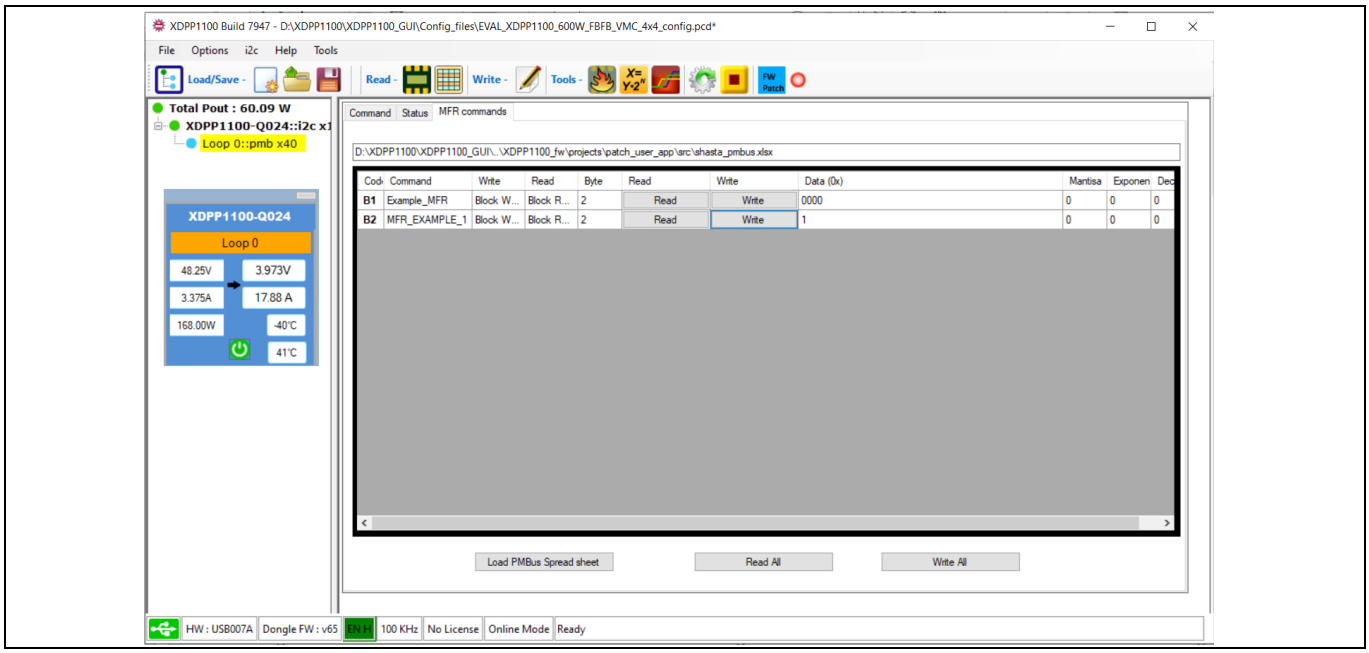


Figure 40 XDPP1100 GUI – running

Figure 39 shows PWM 1 and PWM 2 regulating while the SYNC pin is set to low base on PMBus MFR command 0xB2 set to 0.

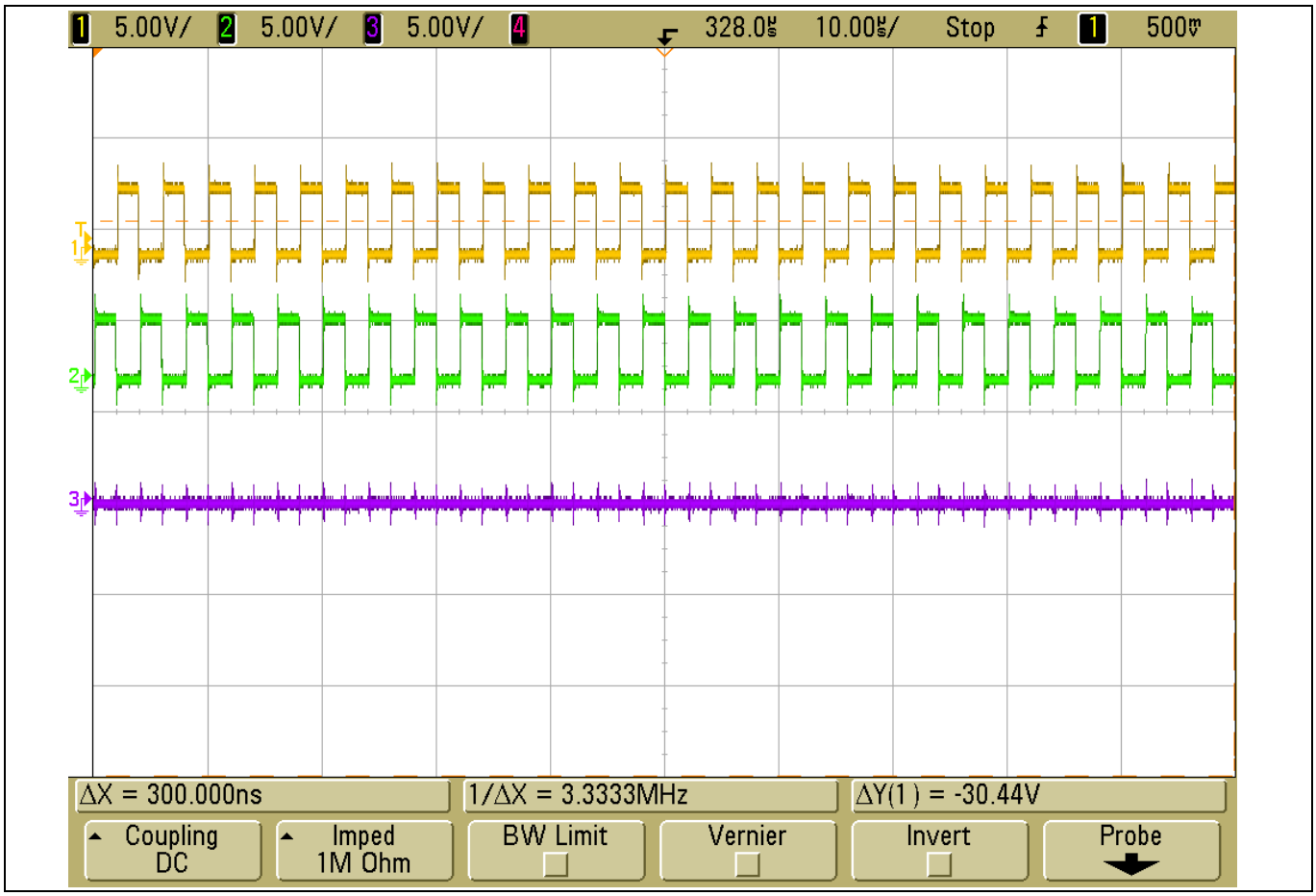


Figure 41 XDPP1100 evaluation measured output signal (PWM 1 pin, PWM 2 pin and SYNC pin)

4 Frequency dithering

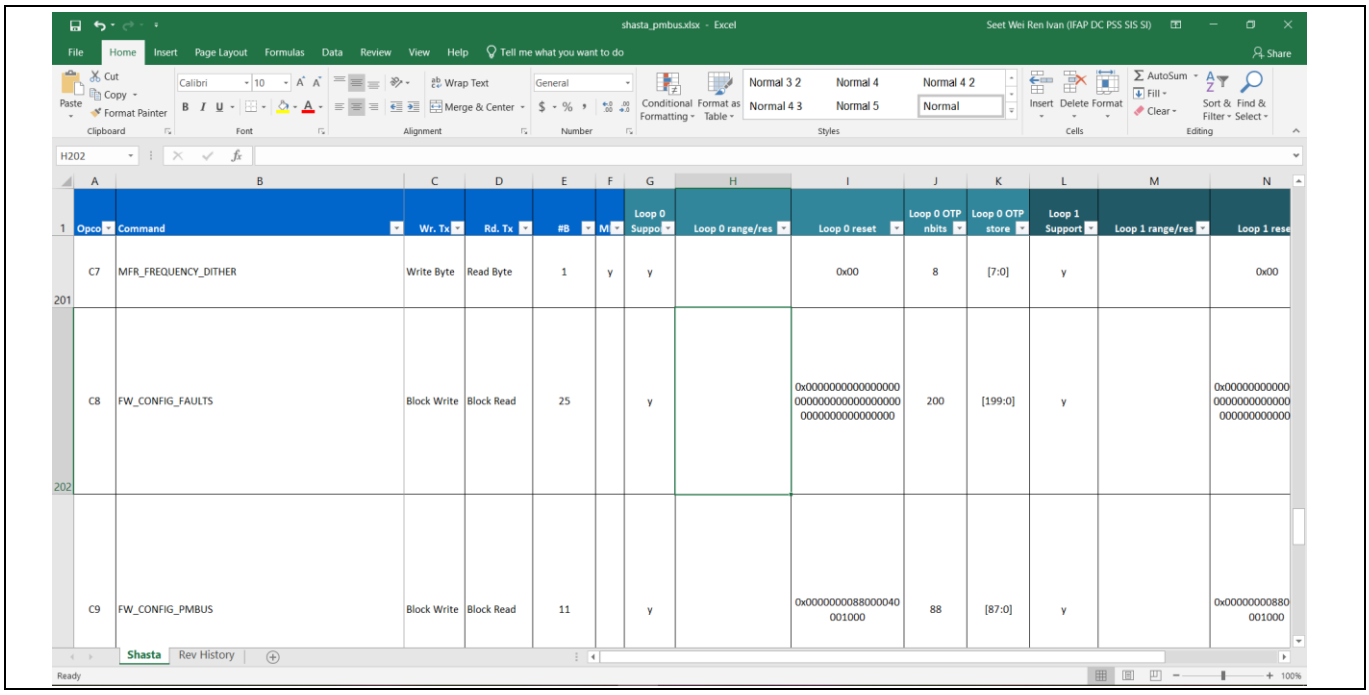
This example is taken from **XDP™ digital power XDPP1100 firmware examples [1]**.

Adding some dithering in the switching frequency helps to randomize quantization errors, which in turn gives a perceived smoother regulation performance. This example codes shows how to add frequency dithering in the PWM.

To set up a new FW patch project patch_example_2, refer to **Section 2.1** Create new project.

4.1 PMBus MFR command for frequency dithering

To create a new PMBus MFR command 0xC7, refer to **Section 2.2** Add a new PMBus MFR command.



	Opco	Command	Wr. Tx	Rd. Tx	#B	M	Loop 0 Suppo	Loop 0 range/res	Loop 0 reset	Loop 0 OTP nbits	Loop 0 OTP store	Loop 1 Support	Loop 1 range/res	Loop 1 rese
201	C7	MFR_FREQUENCY_DITHER	Write Byte	Read Byte	1	y	y		0x00	8	[7:0]	y		0x00
202	C8	FW_CONFIG_FAULTS	Block Write	Block Read	25		y		0x00000000000000000000000000000000000000000000000000000000000000	200	[199:0]	y		0x00000000000000000000000000000000000000000000000000000000000000
	C9	FW_CONFIG_PMBUS	Block Write	Block Read	11		y		0x0000000088000040001000	88	[87:0]	y		0x0000000088000040001000

Figure 42 shasta_pmbus.xlsx

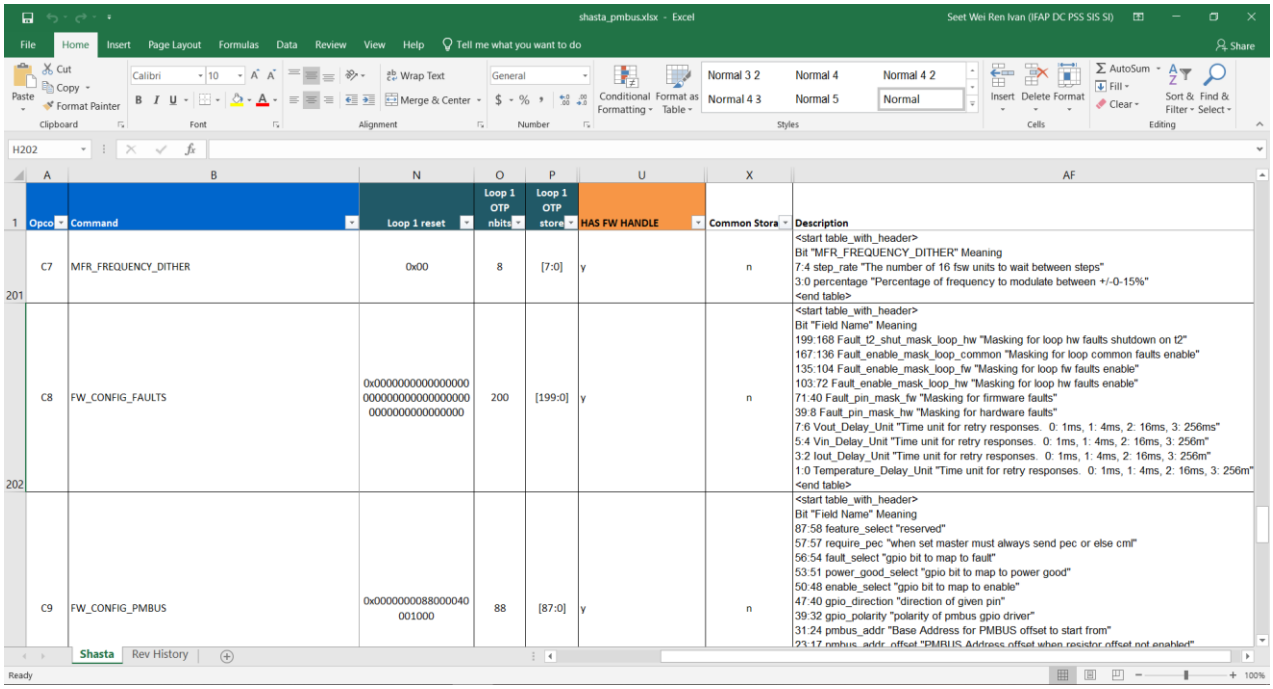


Figure 43 shasta_pmbus.xlsx

The configuration consists of 8-bit data where:

- [7:4] dithering update rate. The number of 16 f_{sw} units to wait between steps.
- [3:0] percentage of frequency to be dithered. The resolution is the percentage number, ±15 percent maximum.

When MFR_FREQUENCY_DITHER is set to a non-zero value, frequency dithering is enabled. The switching frequency will sweep within the percentage defined by bit 3:0, with 20 ns resolution per step, and the duration of each step is defined by bit 7:4, (step_rate +1) *16 * t_{sw}.

For example, writing 25h to the MFR_FREQUENCY_DITHER command will set frequency dithering to +/-5 percent. With 300 kHz base switching frequency, the switching frequency range is 285 kHz to 315 kHz. Keep in mind that the frequency step is calculated based on 20 ns resolution. It takes 32 steps to sweep from F_{min} to F_{max}. Each step will run (2+1) * 16 * 3.32 µs = 159.36 µs. The modulation frequency is 1/(32*2*159.36 µs) = 98 Hz.

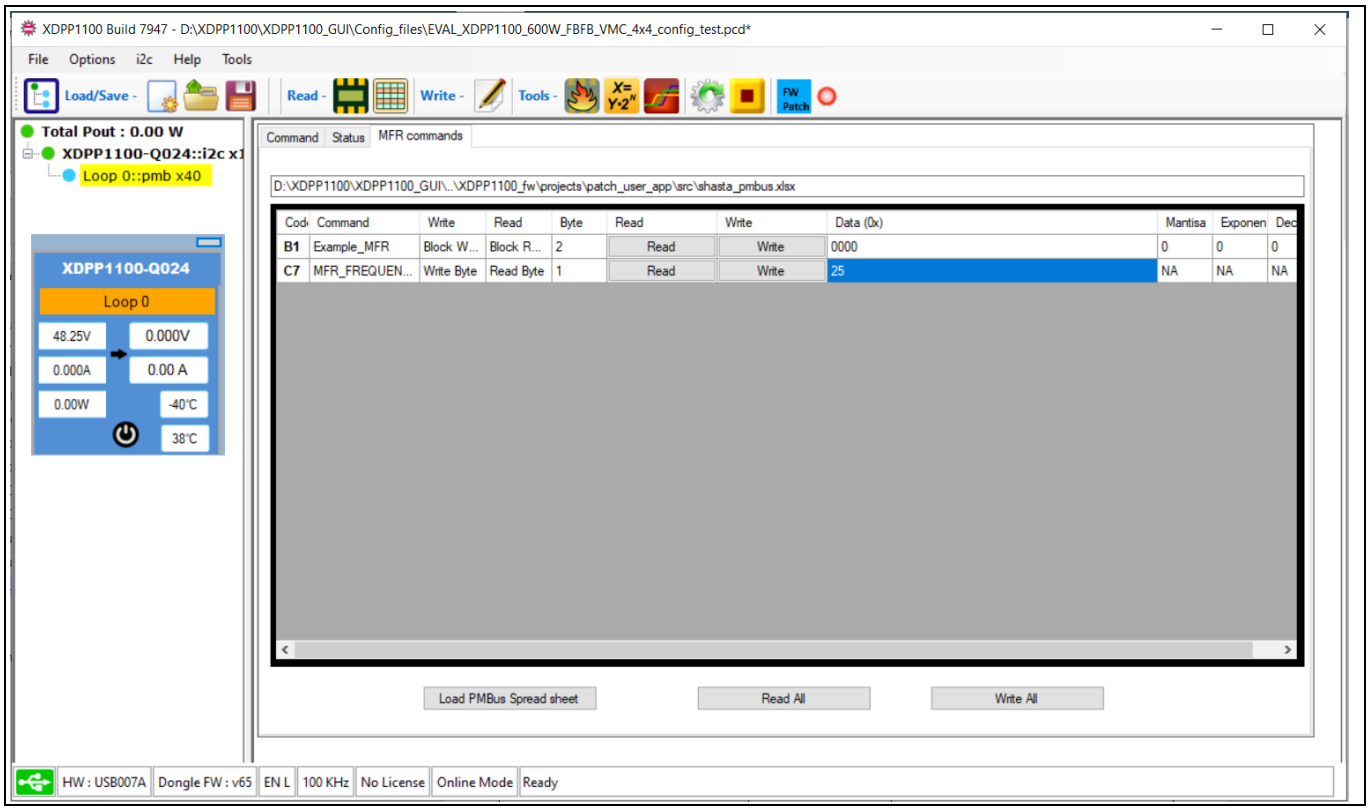


Figure 44 PMBus MFR 0xC7 for frequency dithering

4.2 Modify source code

This section explains the source changes required to enable this function.

Table 2 Additions/Modifications for frequency dithering

Filename	Function name
periodic_functions.c/h	Frequency_dither_enable() Frequency_dither_disable() Frequency_dither_irq_callback()
pmbus_mfr_specific_handlers.c/h	PMBUS_HANDLE_MFR_FREQUENCY_DITHER ()
regulation_state_machine_callbacks.c	Regulation_set_regulation_event_cb(REGULATION_STATE_TOFF_FALL, REGULATION_CONTROLS_SHUTDOWN, AT_TOFF_FALL_SHUTDOWN); Regulation_set_regulation_event_cb(REGULATION_STATE_TOFF_FALL, REGULATION_CONTROLS_SHUTDOWN_IMMEDIATE, AT_TOFF_FALL_SHUTDOWN); Regulation_set_fsw_irq_event_cb(fsw_irq_idx_3, frequency_dither_irq_callback); AT_TARGET_ENABLE() AT_TOFF_FALL_SHUTDOWN()
regulation_state_machine_callbacks.h	USER_DATA{} AT_TARGET_ENABLE() AT_TOFF_FALL_SHUTDOWN()

Filename	Function name
user_app.c	void user_drv_init(void)

4.2.1 PMBus command handler

Open the the pmbus_mfr_specific_handlers.c and insert the PMBus handler “void PMBUS_HANDLER_MFR_FREQUENCY_DITHER()” found in pmbus_mfr_autogen.h. Add the source code accordingly.

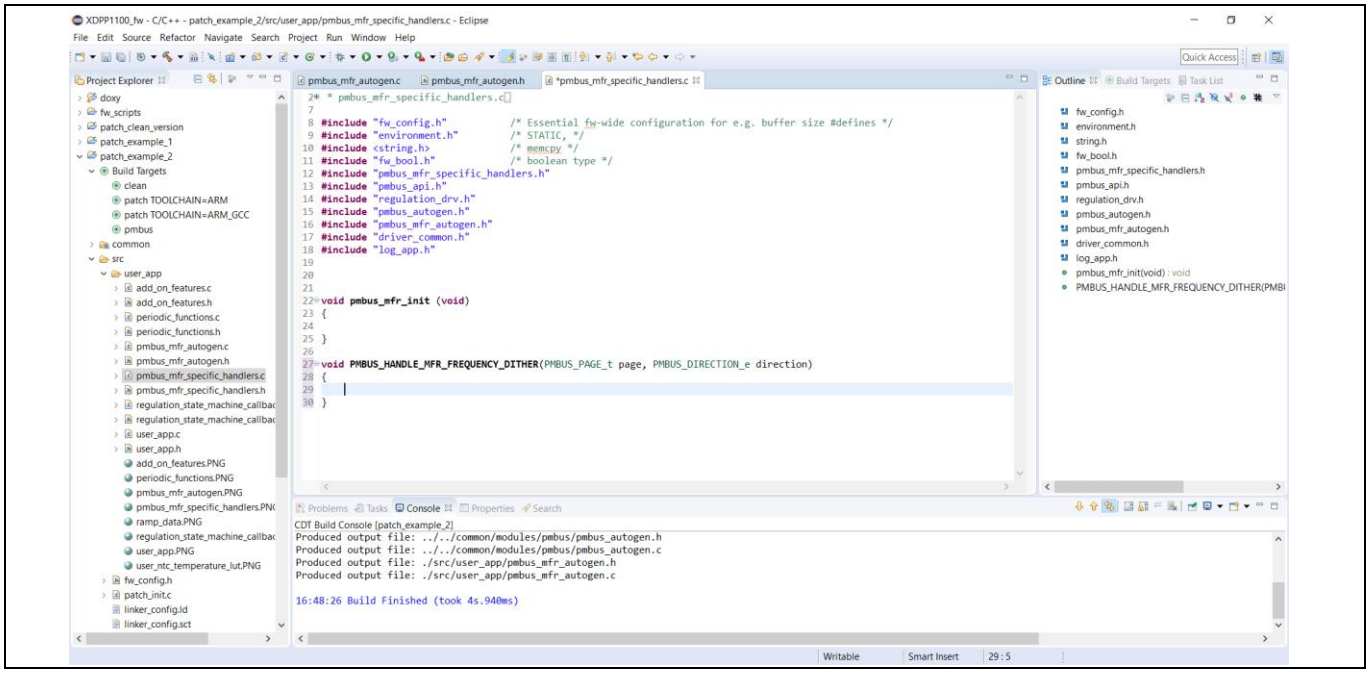


Figure 45 pmbus_mfr_specific_handlers.c

Code Listing 3 pmbus_mfr_specific_handlers.c

```
#include "regulation_state_machine_callbacks.h"
#include "regulation_api.h"
void PMBUS_HANDLE_MFR_FREQUENCY_DITHER(PMBUS_PAGE_t page, PMBUS_DIRECTION_e
direction)
{
if ((direction == PMBUS_WRITE)||(direction == OTP_RESTORE))
{
uint32_t data, percentage, freq_khz, period, delta;
int32_t exponent;
// update the user control data used by irq handle for feature
user_data.control[page] = PMBUS_Get_Command_Byte(page,
PMBUS_CMDCODE_MFR_FREQUENCY_DITHER);
percentage = user_data.control[page]&0xf;
percentage = (uint32_t)SHIFT_EXPONENT((int32_t)percentage, (-PMBUS_Q_EXP));  //
move to PMBUS_Q_EXP format
percentage = percentage/100;  // convert from percentage to fractional number in
PMBUS_Q_EXP format

// grab the current frequency set point and calculate the min and max values
// of the triangle wave
data = PMBUS_Get_Command_Word(page, PMBUS_CMDCODE_FREQUENCY_SWITCH);
freq_khz = (uint32_t)LINEAR11_TO_MANTISSA((uint16_t)data);
exponent = LINEAR11_TO_EXPONENT((uint16_t)data);
```

Code Listing 3 pmbus_mfr_specific_handlers.c

```
    freq_khz = (uint16_t)SHIFT_EXPONENT((int32_t)freq_khz, (exponent));

    // convert frequency to period in ns for midpoint
    period = 1000000/(freq_khz*SWITCHING_PERIOD_LSB_NS);
    if (period > 0x1ff) period = 0x1ff;
    delta = (uint32_t)SHIFT_EXPONENT((int32_t)(period*percentage), (PMBUS_Q_EXP));

    user_data.period_max[page] = period + delta;
    if (user_data.period_max[page] > 0x1fff) user_data.period_max[page] = 0x1ff;
    user_data.period_min[page] = period - delta;
            }
    }
```

4.2.2 User data variable and regulation callback function

Declare variable use in the user data structure and function use in regulation_state_machine_callback.h.

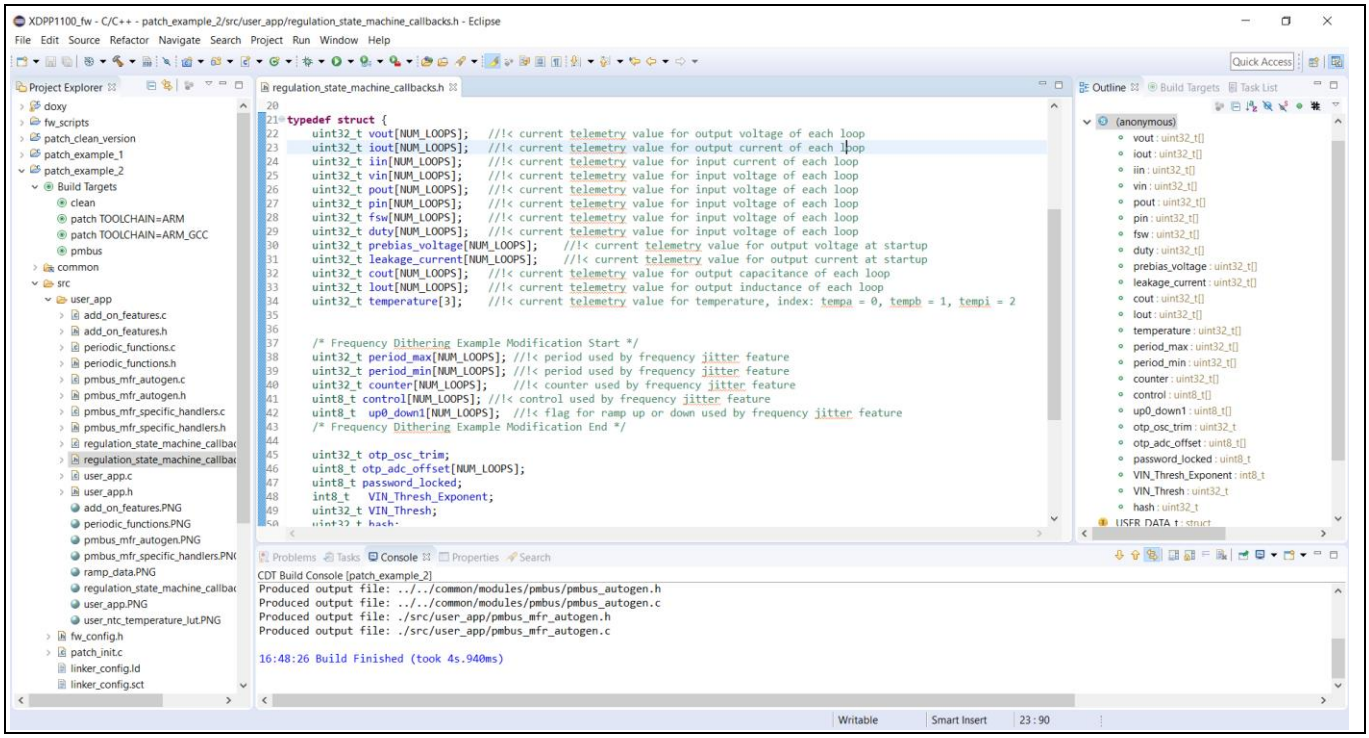


Figure 46 regulation_state_machine_callback.h

Code Listing 4 regulation_state_machine_callback.h

```
    /**
     * Local queue for priority reordering
     */

    typedef struct {
    uint32_t vout[NUM_LOOPS]; //!< current telemetry value for output voltage of each
    loop
    uint32_t iout[NUM_LOOPS]; //!< current telemetry value for output current of each
    loop
    uint32_t iin[NUM_LOOPS];  //!< current telemetry value for input current of each
    loop
```

Code Listing 4 regulation_state_machine_callback.h

```
    uint32_t vin[NUM_LOOPS];  //!< current telemetry value for input voltage of each
    loop
    uint32_t pout[NUM_LOOPS]; //!< current telemetry value for input voltage of each
    loop
    uint32_t pin[NUM_LOOPS];  //!< current telemetry value for input voltage of each
    loop
    uint32_t fsw[NUM_LOOPS];  //!< current telemetry value for input voltage of each
    loop
    uint32_t duty[NUM_LOOPS]; //!< current telemetry value for input voltage of each
    loop
    uint32_t prebias_voltage[NUM_LOOPS];    //!< current telemetry value for output
    voltage at startup
    uint32_t leakage_current[NUM_LOOPS];    //!< current telemetry value for output
    current at startup
    uint32_t cout[NUM_LOOPS]; //!< current telemetry value for output capacitance of
    each loop
    uint32_t lout[NUM_LOOPS]; //!< current telemetry value for output inductance of
    each loop
    uint32_t temperature[3];         //!< current telemetry value for temperature,
    index: tempa = 0, tempb = 1, tempi = 2


    /* Frequency Dithering Example Modification Start */
    uint32_t period_max[NUM_LOOPS];  //!< period used by frequency jitter feature
    uint32_t period_min[NUM_LOOPS];  //!< period used by frequency jitter feature
    uint32_t counter[NUM_LOOPS];     //!< counter used by frequency jitter feature
    uint8_t control[NUM_LOOPS];      //!< control used by frequency jitter feature
    uint8_t  up0_down1[NUM_LOOPS];   //!< flag for ramp up or down used by frequency
    jitter feature
    /* Frequency Dithering Example Modification End */

    uint32_t otp_osc_trim;
    uint8_t otp_adc_offset[NUM_LOOPS];
    uint8_t password_locked;
    int8_t   VIN_Thresh_Exponent;
    uint32_t VIN_Thresh;
    uint32_t hash;



    } USER_DATA_t;

    void AT_TARGET_ENABLE(PMBUS_PAGE_t loop);
    void AT_TOFF_FALL_SHUTDOWN(PMBUS_PAGE_t loop);
```

4.2.3 Declare function for frequency dithering

Declare function required to activate the frequency dithering functions.

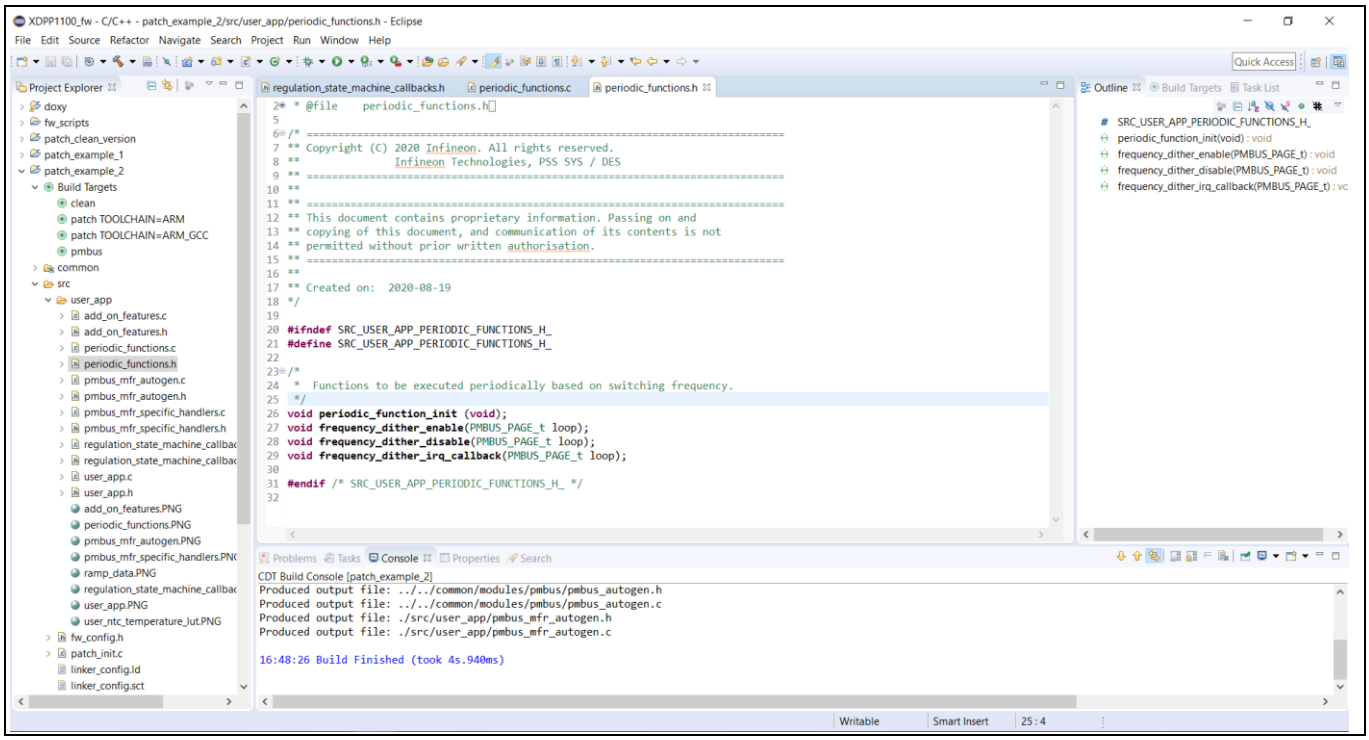


Figure 47 periodic_functions.h

Code Listing 5 periodic_functions.h

```
void frequency_dither_enable(PMBUS_PAGE_t loop);
void frequency_dither_disable(PMBUS_PAGE_t loop);
void frequency_dither_irq_callback(PMBUS_PAGE_t loop);
```

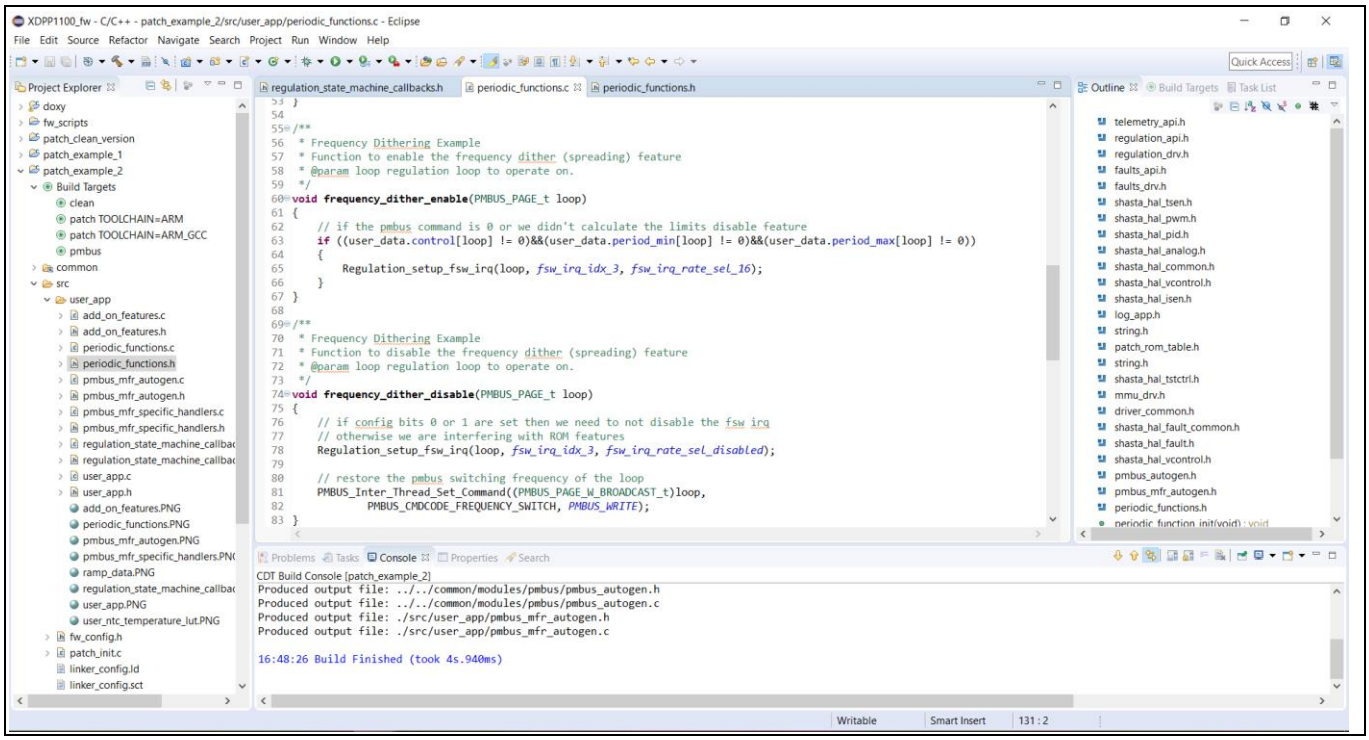


Figure 48 periodic_functions.c

Code Listing 6 periodic_function.c

```
#include "regulation_state_machine_callbacks.h"
/**
 * Frequency Dithering Example
 * Function to enable the frequency dither (spreading) feature
 * @param loop regulation loop to operate on.
 */
void frequency_dither_enable(PMBUS_PAGE_t loop)
{
      // if the pmbus command is 0 or we didn't calculate the limits disable
feature
      if ((user_data.control[loop] != 0)&&(user_data.period_min[loop] !=
0)&&(user_data.period_max[loop] != 0))
      {
                  Regulation_setup_fsw_irq(loop, fsw_irq_idx_3,
fsw_irq_rate_sel_16);
      }
}

/**
 * Frequency Dithering Example
 * Function to disable the frequency dither (spreading) feature
 * @param loop regulation loop to operate on.
 */
void frequency_dither_disable(PMBUS_PAGE_t loop)
{
      // if config bits 0 or 1 are set then we need to not disable the fsw irq
      // otherwise we are interfering with ROM features
      Regulation_setup_fsw_irq(loop, fsw_irq_idx_3, fsw_irq_rate_sel_disabled);

      // restore the pmbus switching frequency of the loop
      PMBUS_Inter_Thread_Set_Command((PMBUS_PAGE_W_BROADCAST_t)loop,
                        PMBUS_CMDCODE_FREQUENCY_SWITCH, PMBUS_WRITE);
}

/**
 * Frequency Dithering Example
 * Interrupt service callback for FSW_IRQ3, to handle frequency dithering
 * This is used for current limited startup for cycle by cycle adjustment of
limit
 * @param loop regulation loop to operate on.
 */
void frequency_dither_irq_callback(PMBUS_PAGE_t loop)
{
      uint32_t num_period_units;
      uint32_t current_period;

      // [7:4] of control is number of periods to step at where period is fsw/64
      num_period_units = user_data.control[loop] >> 4;

      // first we modulo divide the counter by the unit count to see if we are
      // a multiple of the base unit of 64 periods or not
      // we use num_period_units+1 to ensure that we can't divide by zero
      // figure out if we are a multiple of the num_period_units+1 value
           // otherwise we do nothing increment count and exit isr
      if ((user_data.counter[loop] % (num_period_units+1)) == 0)
      {
                  // now grab the current period we are working from
                  if (loop == PMBUS_PAGE_0)
                        current_period = PWM_RAMP0_TSWITCH0__TSWITCH0__GET();
```

Code Listing 6 periodic_function.c

```
                        else
                                current_period = PWM_RAMP1_TSWITCH1__TSWITCH1__GET();

                        // figure out which direction to go
                        if ((user_data.up0_down1[loop] == 0) && (current_period >=
user_data.period_max[loop]))
                                user_data.up0_down1[loop] = 1;  // change to downward
direction
                        else if ((user_data.up0_down1[loop] == 1) && (current_period
<= user_data.period_min[loop]))
                                user_data.up0_down1[loop] = 0;

                        // now do the actual step up or down
                        if (user_data.up0_down1[loop]==0)
                                current_period++;  // 20ns step size multiplied by
number of steps in pmbus command
                        else
                                current_period--;

                        // write the register in hal:
                        if (loop == PMBUS_PAGE_0)
                                PWM_RAMP0_TSWITCH0__TSWITCH0__SET(current_period);
                        else
                                PWM_RAMP1_TSWITCH1__TSWITCH1__SET(current_period);
        }
        user_data.counter[loop]++;  // keep track of interrupt count
}
```

4.2.4 Regulation callback function

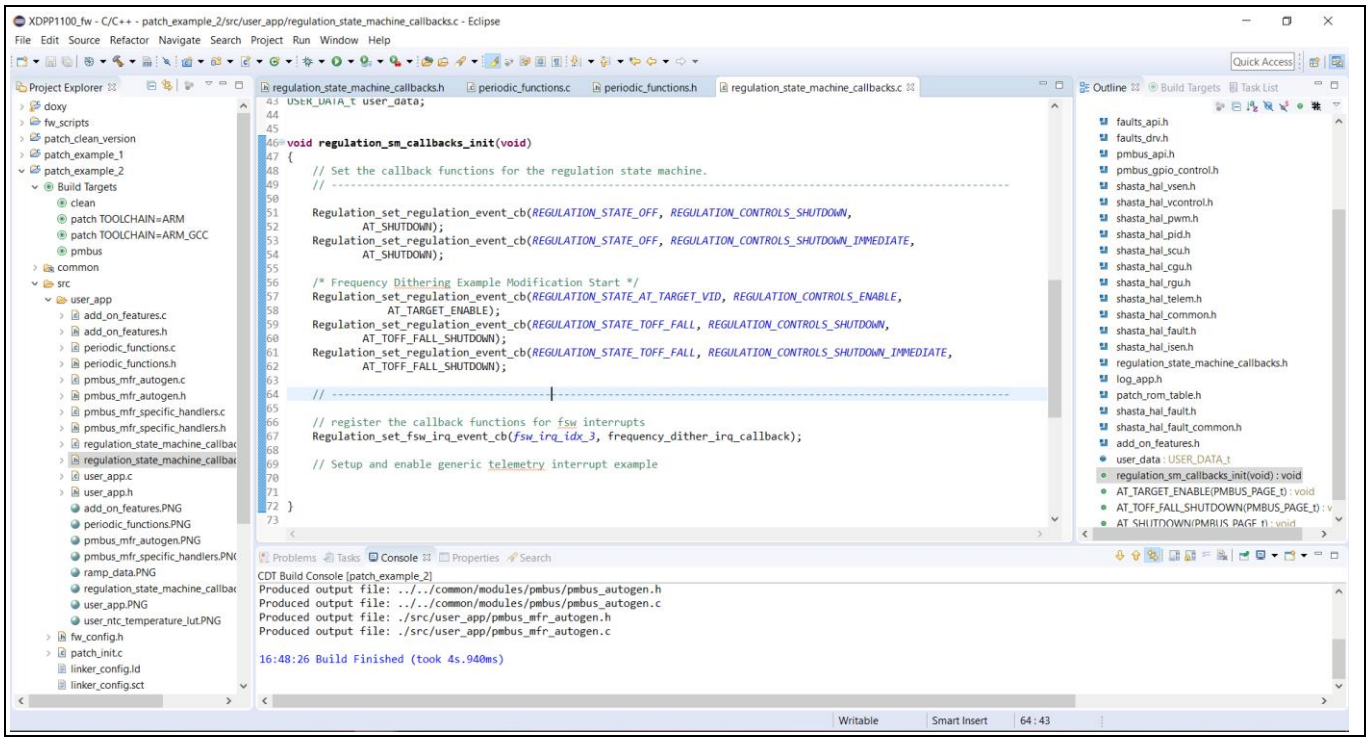


Figure 49 regulation_state_machine.c

Code Listing 7 regulation_state_machine.c

```
#include "periodic_functions.h"
void regulation_sm_callbacks_init(void)
{
      // Set the callback functions for the regulation state machine.
      // ---------------------------------------------------------------------------
---------------------------------

      Regulation_set_regulation_event_cb(REGULATION_STATE_OFF,
REGULATION_CONTROLS_SHUTDOWN,
                          AT_SHUTDOWN);
      Regulation_set_regulation_event_cb(REGULATION_STATE_OFF,
REGULATION_CONTROLS_SHUTDOWN_IMMEDIATE,
                          AT_SHUTDOWN);

      /* Frequency Dithering Example Modification Start */
      Regulation_set_regulation_event_cb(REGULATION_STATE_AT_TARGET_VID,
REGULATION_CONTROLS_ENABLE,
                                 AT_TARGET_ENABLE);
      Regulation_set_regulation_event_cb(REGULATION_STATE_TOFF_FALL,
REGULATION_CONTROLS_SHUTDOWN,
                          AT_TOFF_FALL_SHUTDOWN);
      Regulation_set_regulation_event_cb(REGULATION_STATE_TOFF_FALL,
REGULATION_CONTROLS_SHUTDOWN_IMMEDIATE,
                          AT_TOFF_FALL_SHUTDOWN);

      // ---------------------------------------------------------------------------
---------------------------------

      // register the callback functions for fsw interrupts
      Regulation_set_fsw_irq_event_cb(fsw_irq_idx_3, frequency_dither_irq_callback);

      // Setup and enable generic telemetry interrupt example


}

void AT_TARGET_ENABLE(PMBUS_PAGE_t loop)
{
      /* Frequency Dithering Example Modification Start */
      frequency_dither_enable(loop);
      /* Frequency Dithering Example Modification End */
}

void AT_TOFF_FALL_SHUTDOWN(PMBUS_PAGE_t loop)
{
      /* Frequency Dithering Example Modification Start */
      frequency_dither_disable(loop);
      /* Frequency Dithering Example Modification End */
}
void AT_SHUTDOWN(PMBUS_PAGE_t loop)
{
     VCONTROL_VC_VRAMP3__VC_VCONTROL_DELTA__SET((vcontrol_idx_t) loop, 0);
      frequency_dither_disable(loop);
}
```

4.2.5 User app

Open the the user_app.c and insert source code accordingly in void user_drv_init(void) to run the FW patch in RAM.

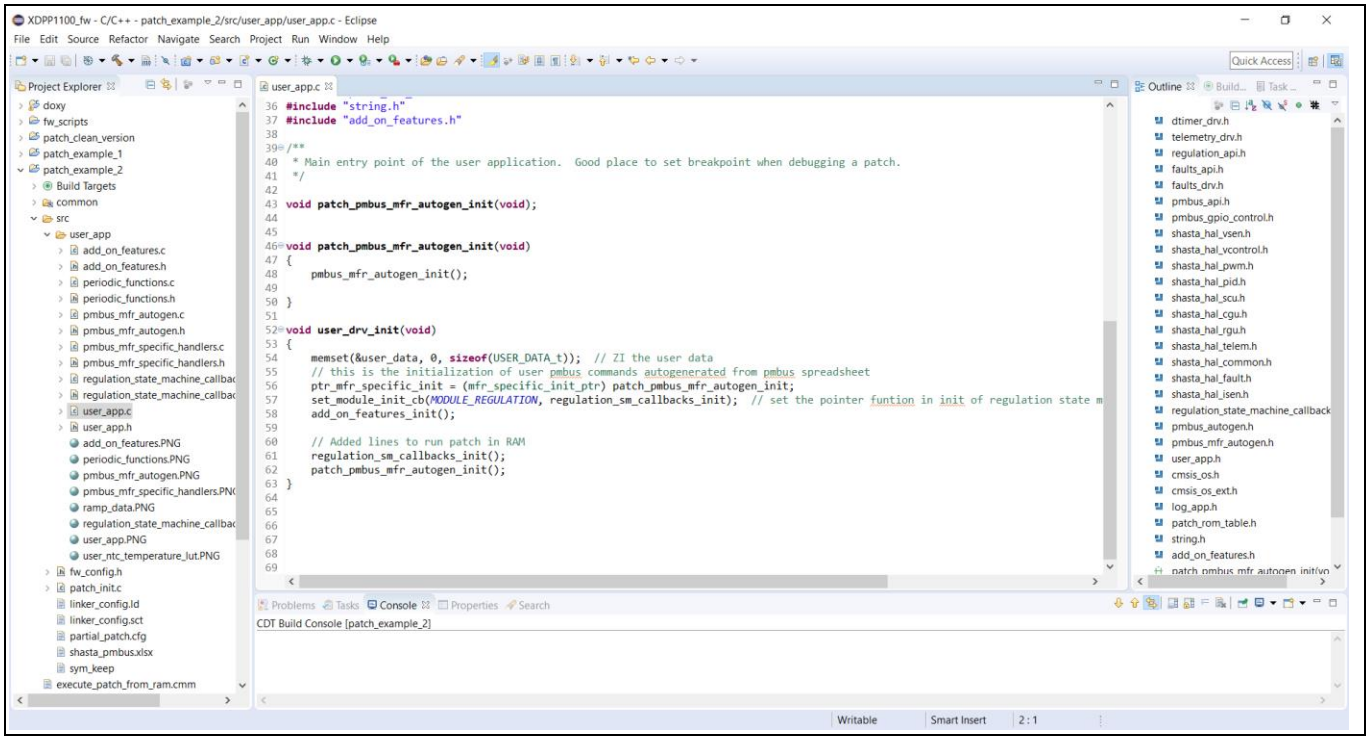


Figure 50 user_app.c

Code Listing 8 user_app.c

```
void user_drv_init(void)
{
  memset(&user_data, 0, sizeof(USER_DATA_t));  // ZI the user data
  // 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();

  // Added lines to run patch in RAM
   regulation_sm_callbacks_init();
   patch_pmbus_mfr_autogen_init();
}
```

4.3 Compile and program to XDPP1100 RAM/OTP

To compile and program the FW patch, refer to **Section 3.3** Compile and program to XDPP1100 RAM/OTP.

4.3.1 Running example 2 patch

Click the “MFR commands” tab and click the “Load PMBus Spread sheet” button to load the spreadsheet with modified MFR commands.

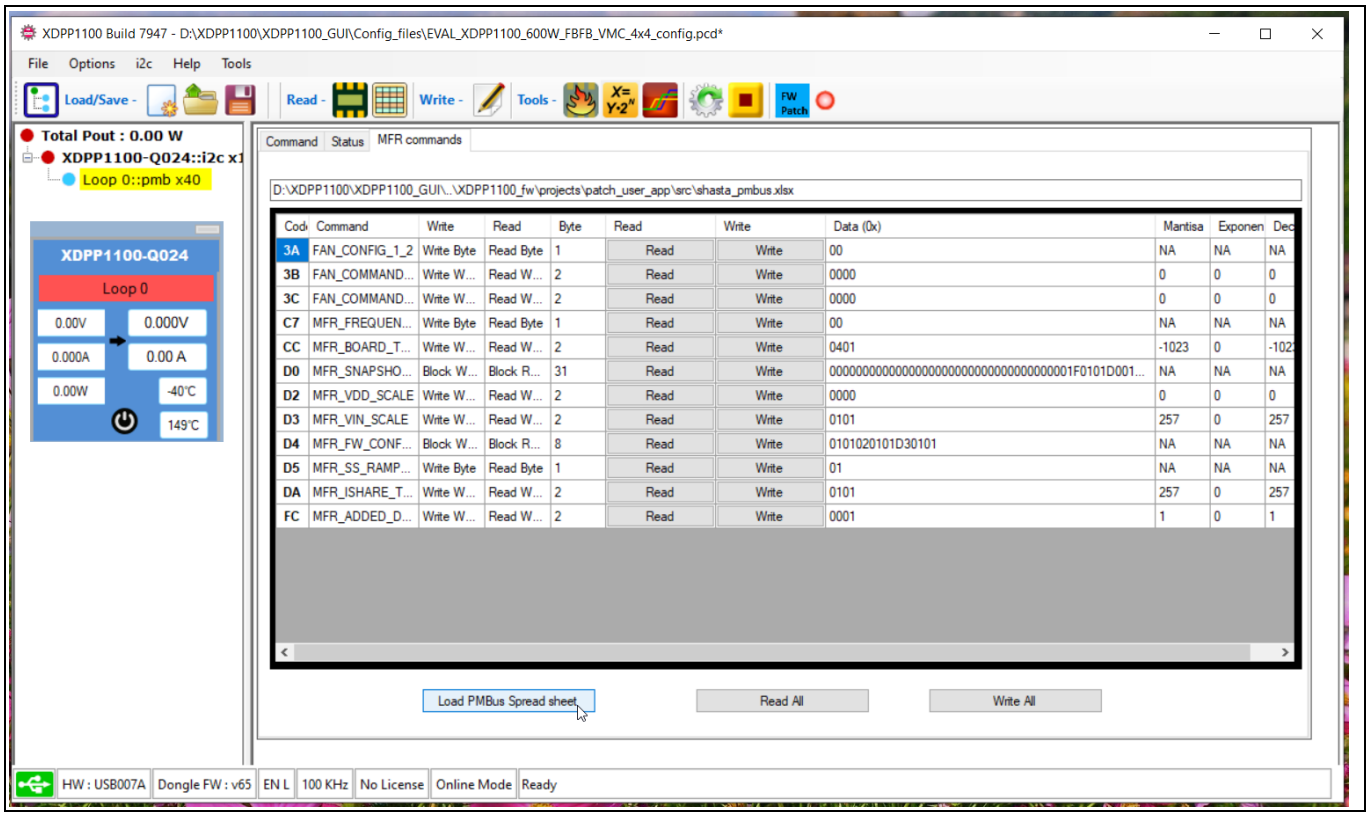


Figure 51 XDPP1100 GUI – MFR commands tab

Select the shasta_pmbus.xlsx in \project\patch_example_1\src.

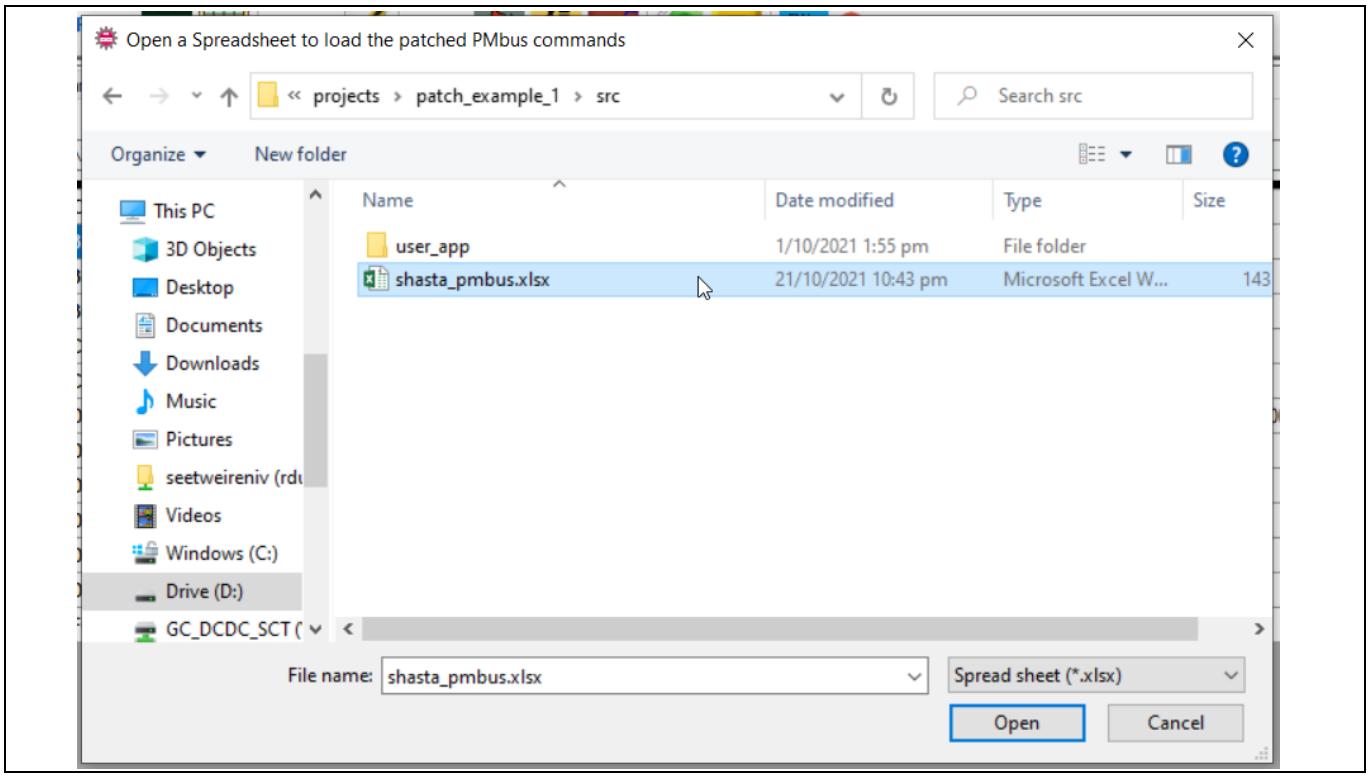


Figure 52 XDPP1100 GUI – open latest MFR commands spreadsheet

The spreadsheet is loaded as shown in **Figure 49**. Click the “Write” button in the top panel to write all register map values and PMBus commands values into RAM.

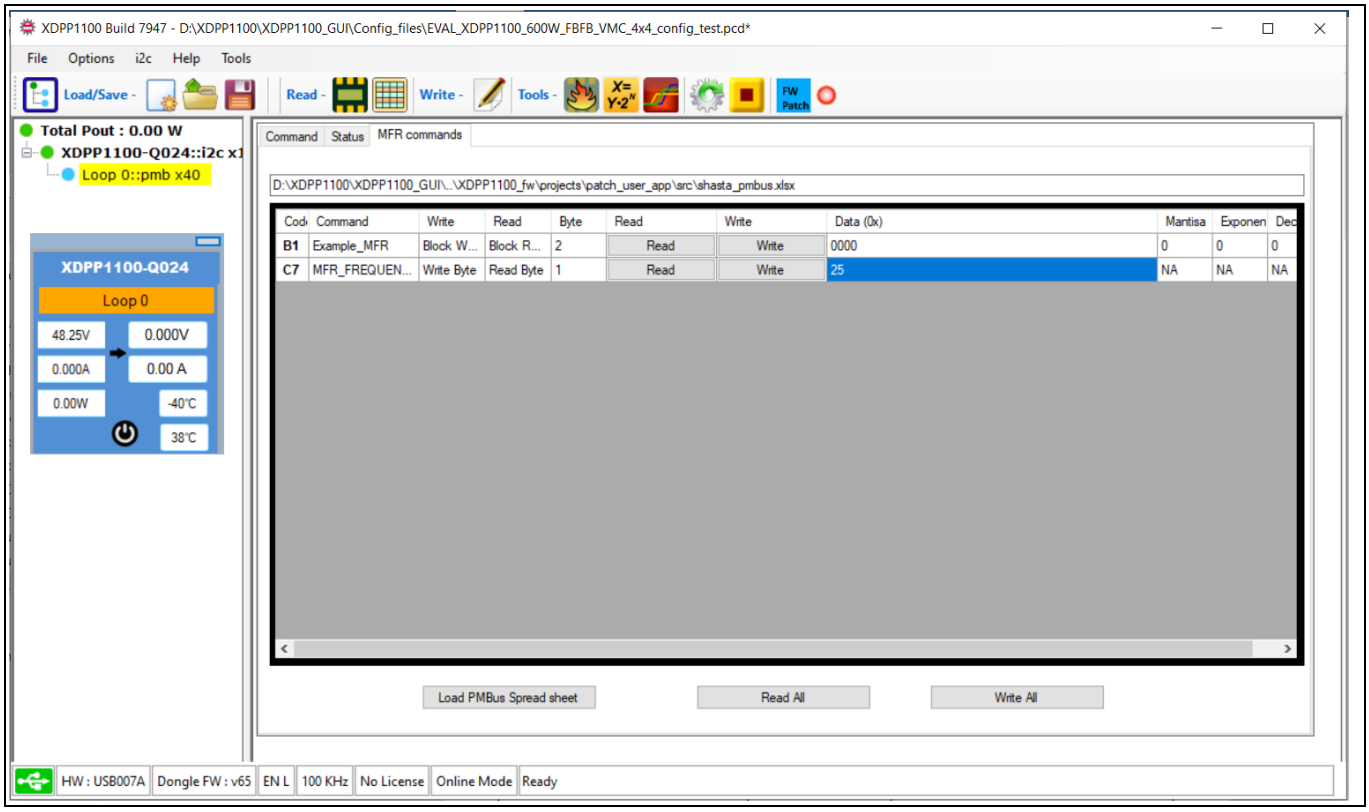


Figure 53 XDPP1100 GUI – MFR commands tab with latest spreadsheet

Right-click “XDPP1100-Q024::i2c” in the left panel and press “Clear Fault” to clear all faults after writing to RAM.

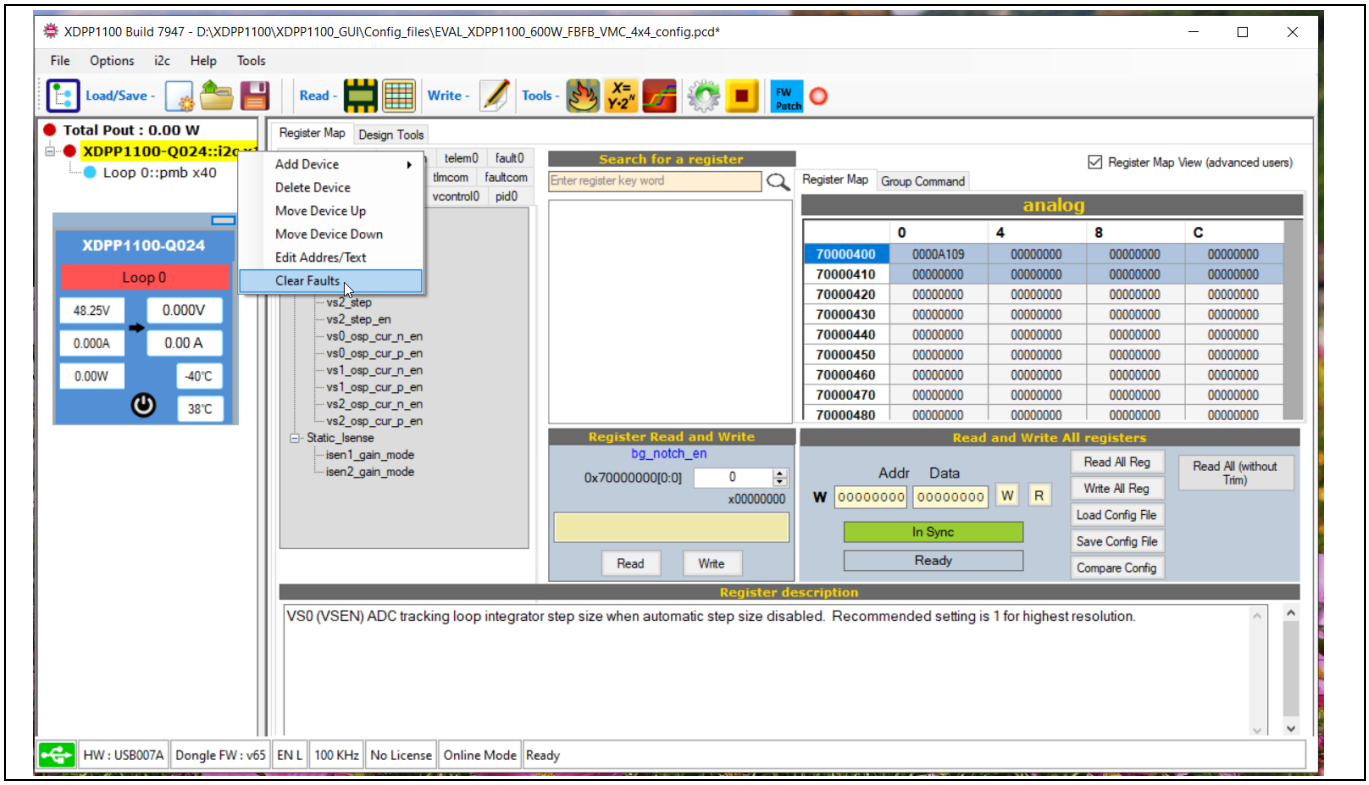


Figure 54 XDPP1100 GUI – clear faults

Frequency dithering

Proceed to the MFR commands and change the value of 0xC7 accordingly, and click “Write”. Click “EN H/EN L” to start the PWM regulating.

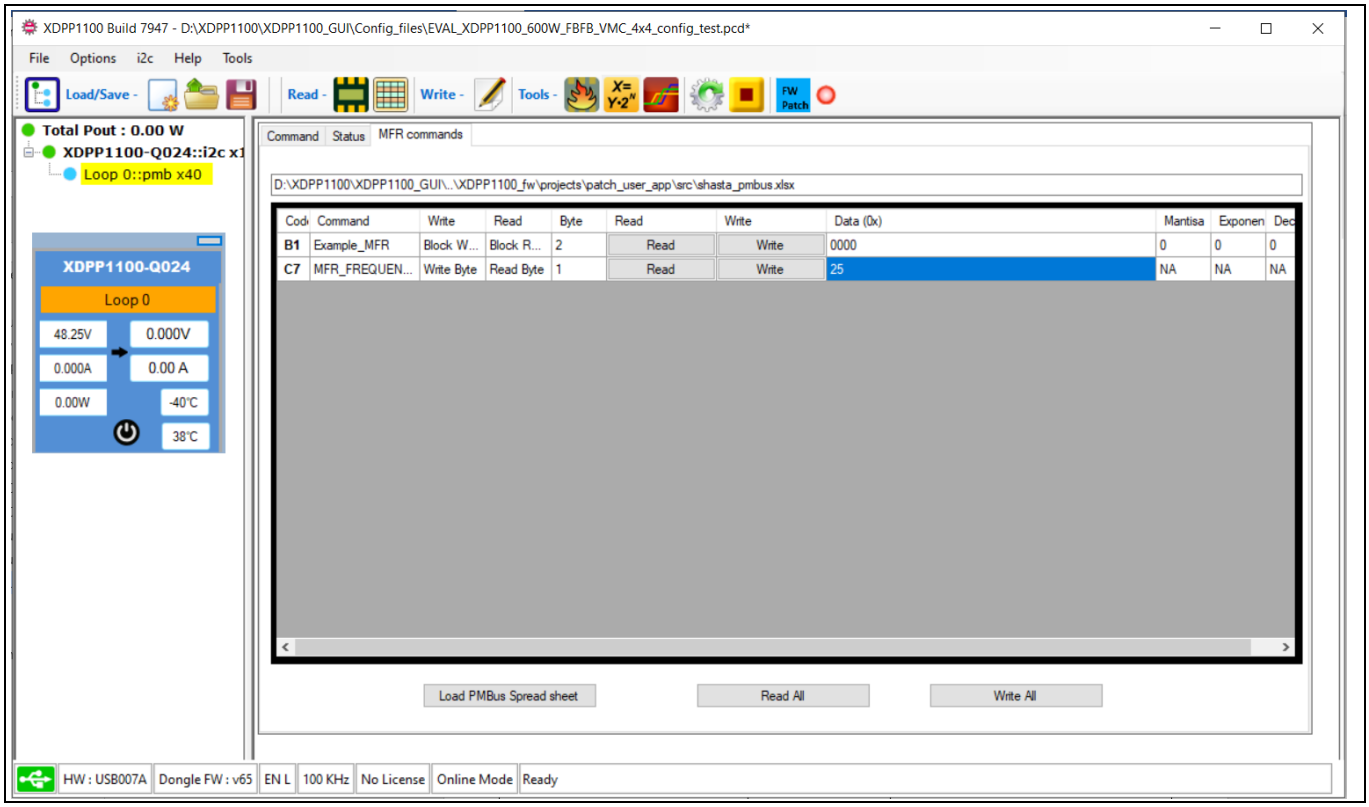


Figure 55 XDPP1100 GUI – running

5 Telemetry interrupt and FW fault

This example is taken from **XDP™ digital power XDPP1100 firmware examples [1]**.

This example shows how to use the telemetry IRQ to detect the output voltage above or below the set limit. It will trigger a FW fault and report PMBus status when above the set limit.

To set up a new FW patch project patch_example_4, refer to **Section 2.1** Create new project.

5.1 PMBus MFR command for OVP fault limit

To create a new PMBus MFR command 0xB1 and 0xB2, refer to **Section 2.2** Add a new PMBus MFR command.

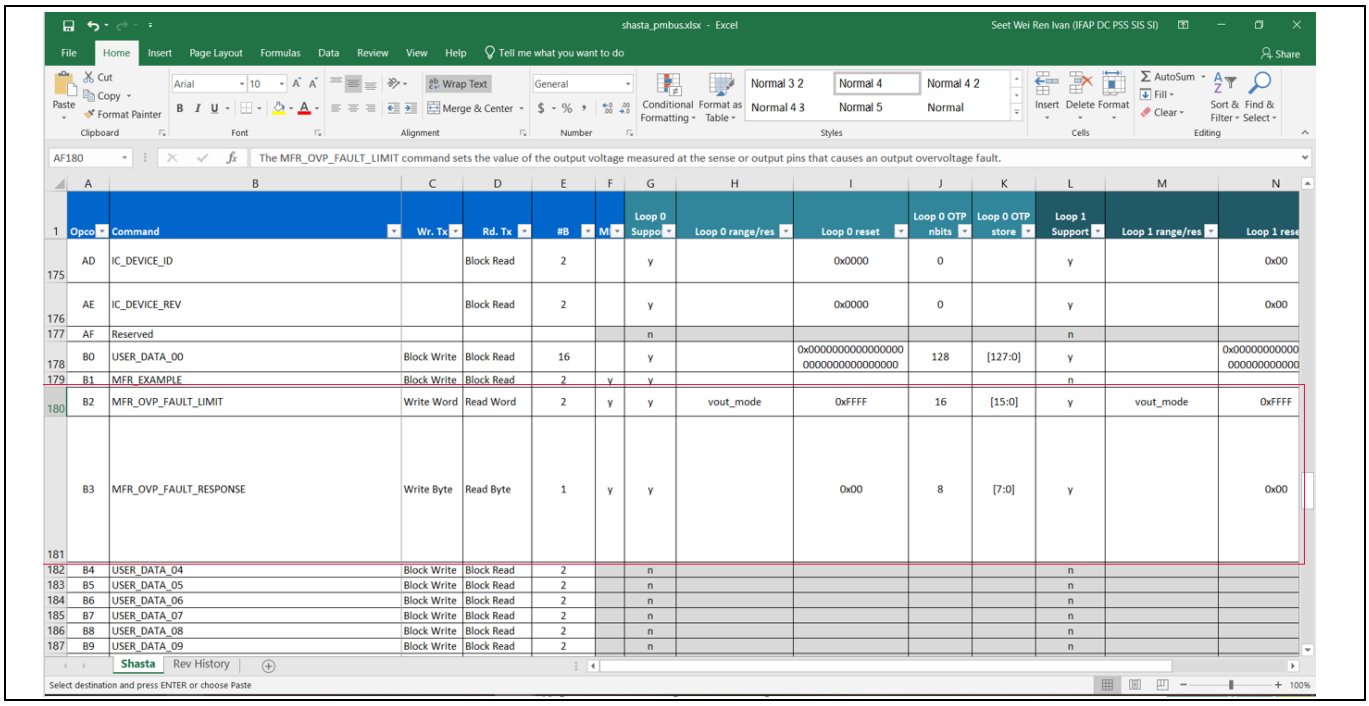
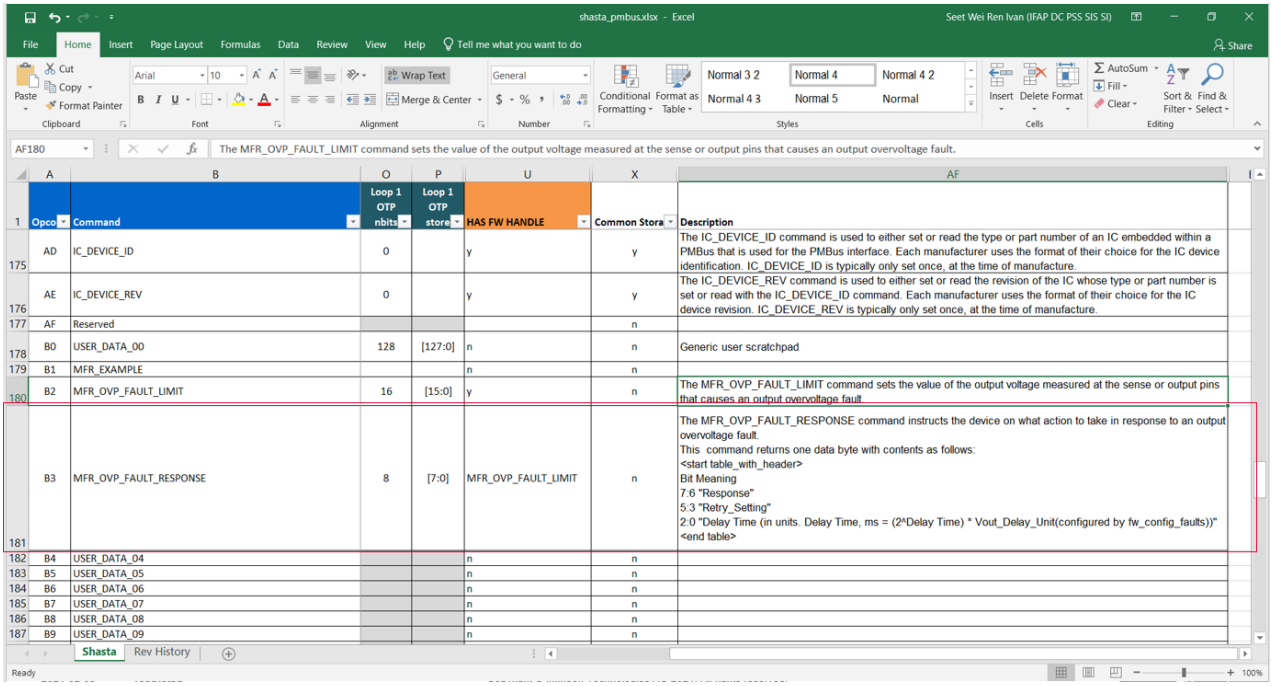


Figure 56 shasta_pmbus.xlsx



shasta_pmbus.xlsx - Excel

	A	B	O	P	U	X	AF
1	Opco	Command	Loop 1 OTP nbits	Loop 1 OTP store	HAS FW HANDLE	Common Stora	Description
175	AD	IC_DEVICE_ID	0		y	y	The IC_DEVICE_ID command is used to either set or read the type or part number of an IC embedded within a PMBus that is used for the PMBus interface. Each manufacturer uses the format of their choice for the IC device identification. IC_DEVICE_ID is typically only set once, at the time of manufacture.
176	AE	IC_DEVICE_REV	0		y	y	The IC_DEVICE_REV command is used to either set or read the revision of the IC whose type or part number is set or read with the IC_DEVICE_ID command. Each manufacturer uses the format of their choice for the IC device revision. IC_DEVICE_REV is typically only set once, at the time of manufacture.
177	AF	Reserved				n	
178	B0	USER_DATA_00	128	[127:0]	n	n	Generic user scratchpad
179	B1	MFR_EXAMPLE			n	n	
180	B2	MFR_OVP_FAULT_LIMIT	16	[15:0]	y	n	The MFR_OVP_FAULT_LIMIT command sets the value of the output voltage measured at the sense or output pins that causes an output overvoltage fault.
181	B3	MFR_OVP_FAULT_RESPONSE	8	[7:0]	MFR_OVP_FAULT_LIMIT	n	The MFR_OVP_FAULT_RESPONSE command instructs the device on what action to take in response to an output overvoltage fault. This command returns one data byte with contents as follows: <start table_with_header> Bit Meaning 7:6 "Response" 5:3 "Retry_Setting" 2:0 "Delay Time (in units. Delay Time, ms = (2^Delay Time) * Vout_Delay_Unit(configured by fw_config_faults))" <end table>
182	B4	USER_DATA_04			n	n	
183	B5	USER_DATA_05			n	n	
184	B6	USER_DATA_06			n	n	
185	B7	USER_DATA_07			n	n	
186	B8	USER_DATA_08			n	n	
187	B9	USER_DATA_09			n	n	

Figure 57 shasta_pmbus.xlsx

5.2 Modify source code

Table 3 Additions/Modifications to create a custom V_{IN} telemetry IRQ

Filename	Function name
ovp_fault.c/h	void Telemetry_IRQ_VOUT_MFR_OVP_FAULT_HANDLE (Telemetry_IRQ_TYPE_t Telemetry_type); void PMBUS_HANDLE_MFR_OVP_FAULT_LIMIT (PMBUS_PAGE_t page, PMBUS_DIRECTION_e direction)
user_app.c	void user_drv_init(void)

5.2.1 Add telemetry interrupt functions and PMBus handler function

Create a new source file and header file name ovp_fault.c and ovp_fault.h.

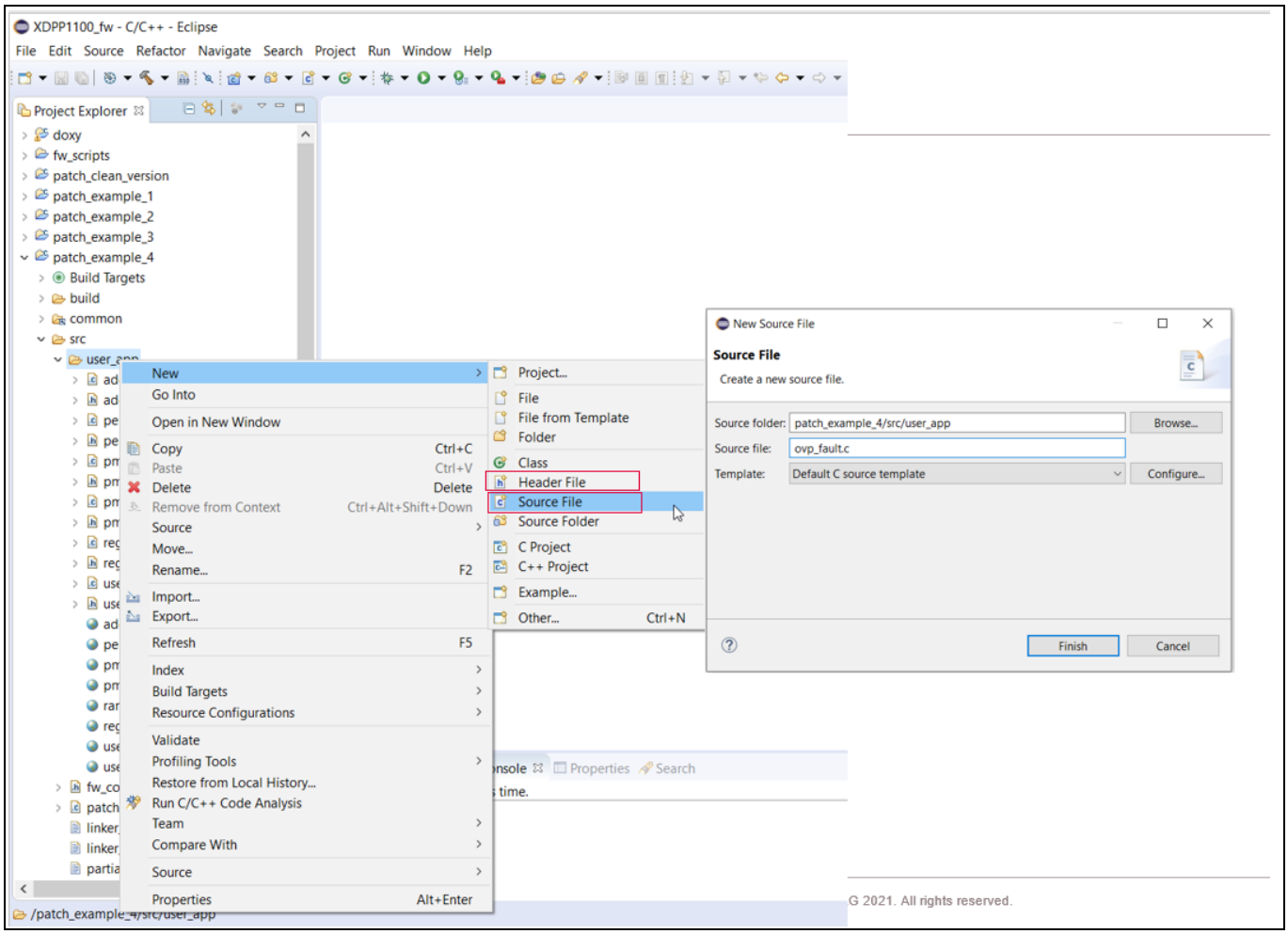


Figure 58 Create source file ovp_fault.c

Open ovp_fault.c and insert the necessary function to initialize telemetry interrupt. Add the source code accordingly.

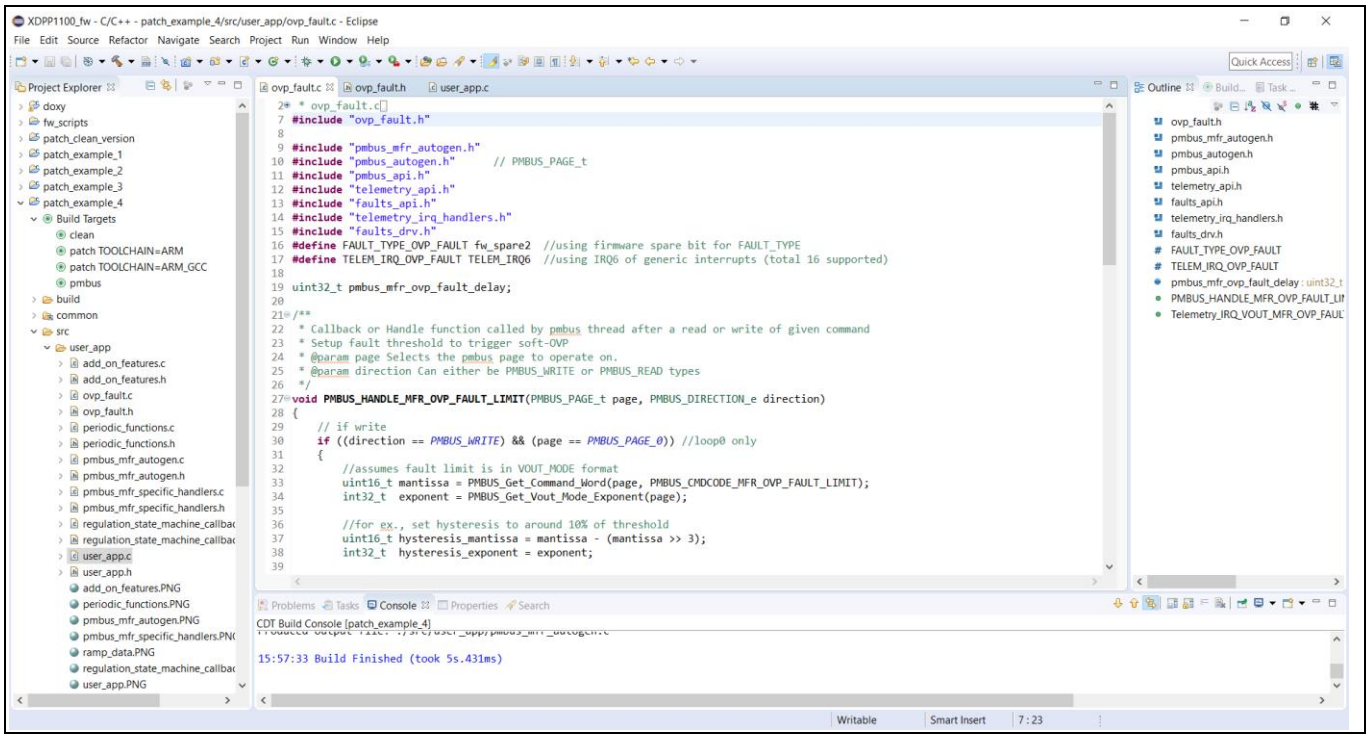


Figure 59 ovp_fault.c

Code Listing 9 ovp_fault.c

```
#include "ovp_fault.h"

#include "pmbus_mfr_autogen.h"
#include "pmbus_autogen.h"       // PMBUS_PAGE_t
#include "pmbus_api.h"
#include "telemetry_api.h"
#include "faults_api.h"
#include "telemetry_irq_handlers.h"
#include "faults_drv.h"
#define FAULT_TYPE_OVP_FAULT fw_spare2  //using firmware spare bit for
FAULT_TYPE
#define TELEM_IRQ_OVP_FAULT TELEM_IRQ6  //using IRQ6 of generic interrupts
(total 16 supported)

uint32_t pmbus_mfr_ovp_fault_delay;

/**
 * Callback or Handle function called by pmbus thread after a read or write of
given command
 * Setup fault threshold to trigger soft-OVP
 * @param page Selects the pmbus page to operate on.
 * @param direction Can either be PMBUS_WRITE or PMBUS_READ types
 */
void PMBUS_HANDLE_MFR_OVP_FAULT_LIMIT(PMBUS_PAGE_t page, PMBUS_DIRECTION_e
direction)
{
      // if write
      if ((direction == PMBUS_WRITE) && (page == PMBUS_PAGE_0)) //loop0 only
      {
        //assumes fault limit is in VOUT_MODE format
        uint16_t mantissa = PMBUS_Get_Command_Word(page,
PMBUS_CMDCODE_MFR_OVP_FAULT_LIMIT);
```

Code Listing 9 ovp_fault.c

```
            int32_t  exponent = PMBUS_Get_Vout_Mode_Exponent(page);

            //for ex., set hysteresis to around 10% of threshold
            uint16_t hysteresis_mantissa = mantissa - (mantissa >> 3);
            int32_t  hysteresis_exponent = exponent;


            Telemetry_Set_Irq_Threshold(TELEM_IRQ_OVP_FAULT, (int32_t)mantissa,
(int8_t)exponent);
            Telemetry_Set_Irq_Hysteresis_Threshold(TELEM_IRQ_OVP_FAULT,
(int32_t)hysteresis_mantissa, (int8_t)hysteresis_exponent);
            Telemetry_Setup_Irq(page, VOUT_0, TELEM_IRQ_OVP_FAULT,
Telemetry_IRQ_VOUT_MFR_OVP_FAULT_HANDLE);

            uint8_t fault_response_cmd_byte = PMBUS_Get_Command_Byte(page,
PMBUS_CMDCODE_MFR_OVP_FAULT_RESPONSE);

            //SETUP response and reporting on STATUS MFR SPECIFIC bit 1
            FAULTS_SETUP_FAULT(page, FAULT_TYPE_OVP_FAULT, fault_response_cmd_byte,
FAULTS_CONFIG_VOUT_DELAY_UNIT, VOLTAGE_RESPONSE_TYPE,
(uint8_t)PMBUS_CMDCODE_STATUS_MFR_SPECIFIC,
(uint8_t)PMBUS_STATUS_MFR_SPECIFIC_1, FAULT_TYPE_OVP_FAULT);

            // enable telemetry interrupt
            Telemetry_Enable_IRQ_Type(TELEM_IRQ_OVP_FAULT);
        }
}

/**
 * This function is Telem event callback for soft-OVP.
 * @param Telemetry_type
 */
void Telemetry_IRQ_VOUT_MFR_OVP_FAULT_HANDLE(Telemetry_IRQ_TYPE_t
Telemetry_type)
{
        _Telemetry_s * Telemetry_str =
Telemetry_GET_ARRAY_POINTER(Telemetry_type);
        if(Telemetry_str->polarity == 1)
        {
            //VOUT above the limit
            _FAULT_DESC_s * fault;
            //overwrite the delay to shutdown
            fault = FAULTS_GET_ARRAY_POINTER(PMBUS_PAGE_0, FAULT_TYPE_OVP_FAULT);
            if (fault == NULL)
            {
                PMBUS_Log_Error(FAULTS, PROGRAM_COUNTER,
PMBUS_STATUS_CML_PROCESSOR_FAULT);
                return;
            }

            pmbus_mfr_ovp_fault_delay = fault->delay_time_ms; //save the value of
delay from pmbus command

            fault->delay_time_ms = 10; //set the delay to shutdown

            Set_Firmware_Fault(PMBUS_PAGE_0, FAULT_TYPE_OVP_FAULT);
        }
        else
        {
```

Code Listing 9 ovp_fault.c

```
            //VOUT below hysteresis level

            // set the delay to retry
            _FAULT_DESC_s * fault;
            fault = FAULTS_GET_ARRAY_POINTER(PMBUS_PAGE_0, FAULT_TYPE_OVP_FAULT);
            if (fault == NULL)
            {
                PMBUS_Log_Error(FAULTS, PROGRAM_COUNTER,
    PMBUS_STATUS_CML_PROCESSOR_FAULT);
                return;
            }

            //set the delay to retry
            fault->delay_time_ms = pmbus_mfr_ovp_fault_delay; //

            Clear_Firmware_Fault(PMBUS_PAGE_0, FAULT_TYPE_OVP_FAULT);
        }
    }
```

5.2.2 Add header file initialization

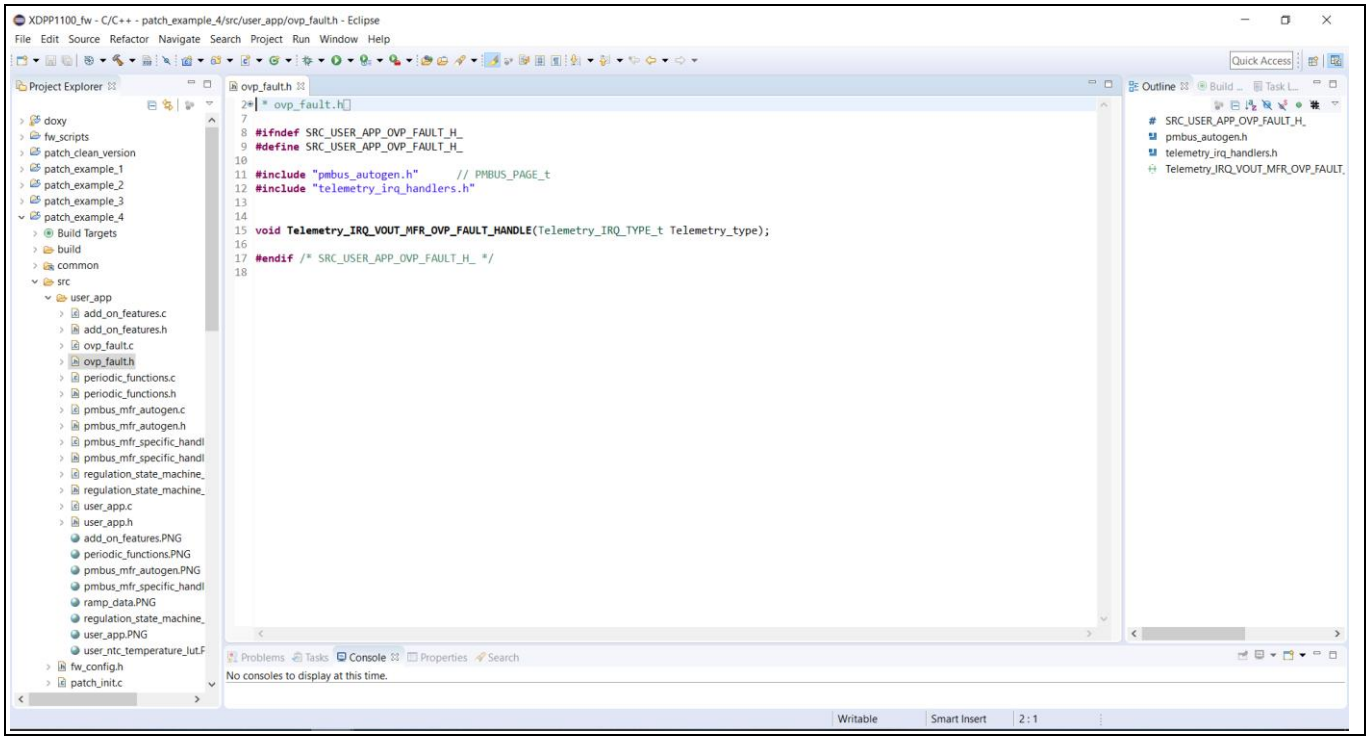


Figure 60 ovp_fault.h

Code Listing 10 ovp_fault.h

```
    #include "pmbus_autogen.h"      // PMBUS_PAGE_t
    #include "telemetry_irq_handlers.h"


    void Telemetry_IRQ_VOUT_MFR_OVP_FAULT_HANDLE(Telemetry_IRQ_TYPE_t
    Telemetry_type);
```

5.2.3 User app

Open the the user_app.c and insert source code accordingly to **Code Listing 2** in void user_drv_init(void) to run the FW patch in RAM.

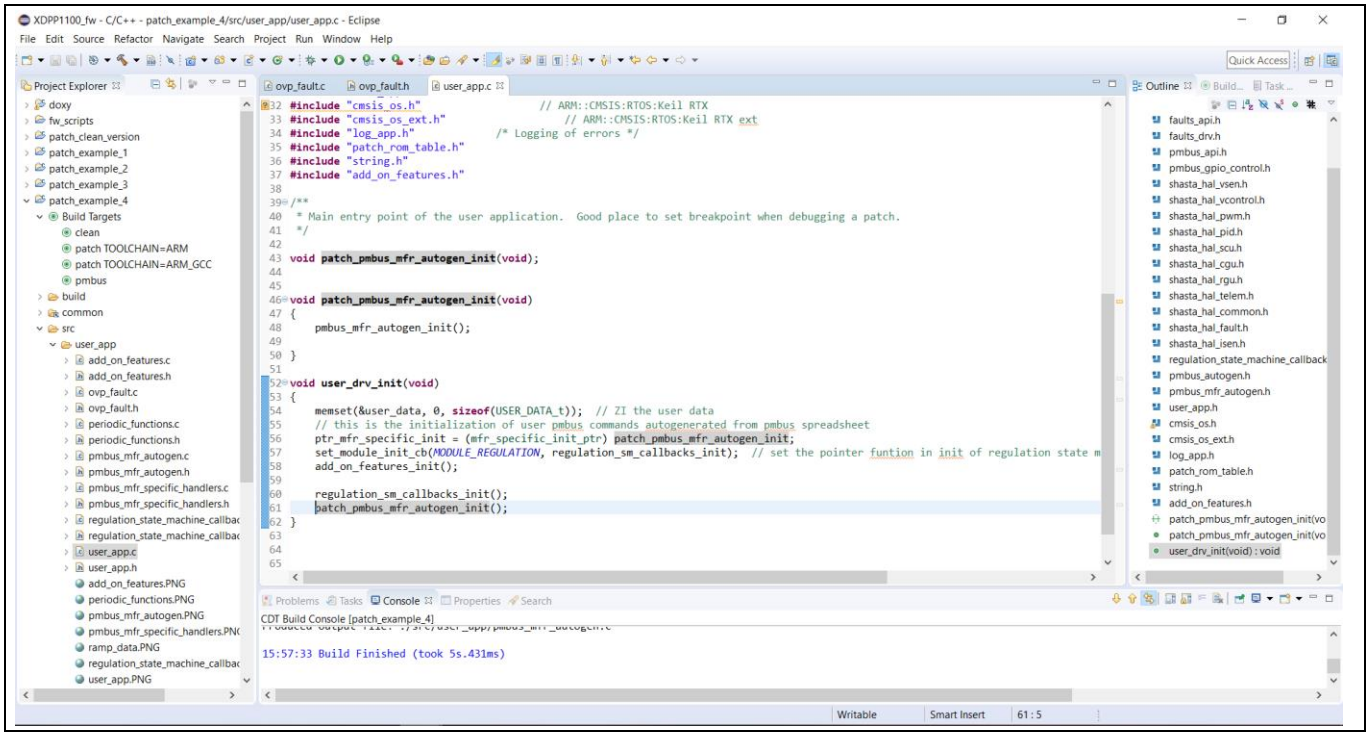


Figure 61 user_app.c

Code Listing 11 user_app.c

```
void user_drv_init(void)
{
  memset(&user_data, 0, sizeof(USER_DATA_t));  // ZI the user data
  // 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();

  // Added lines to run patch in RAM
   regulation_sm_callbacks_init();
   patch_pmbus_mfr_autogen_init();
}
```

5.3 Compile and program to XDPP1100 RAM/OTP

To compile and program the FW patch refer to **Section 3.3** Compile and program to XDPP1100 RAM/OTP.

5.3.1 Running example 4 patch

Click the “MFR commands” tab and click the “Load PMBus Spread sheet” button to load the spreadsheet with modified MFR commands.

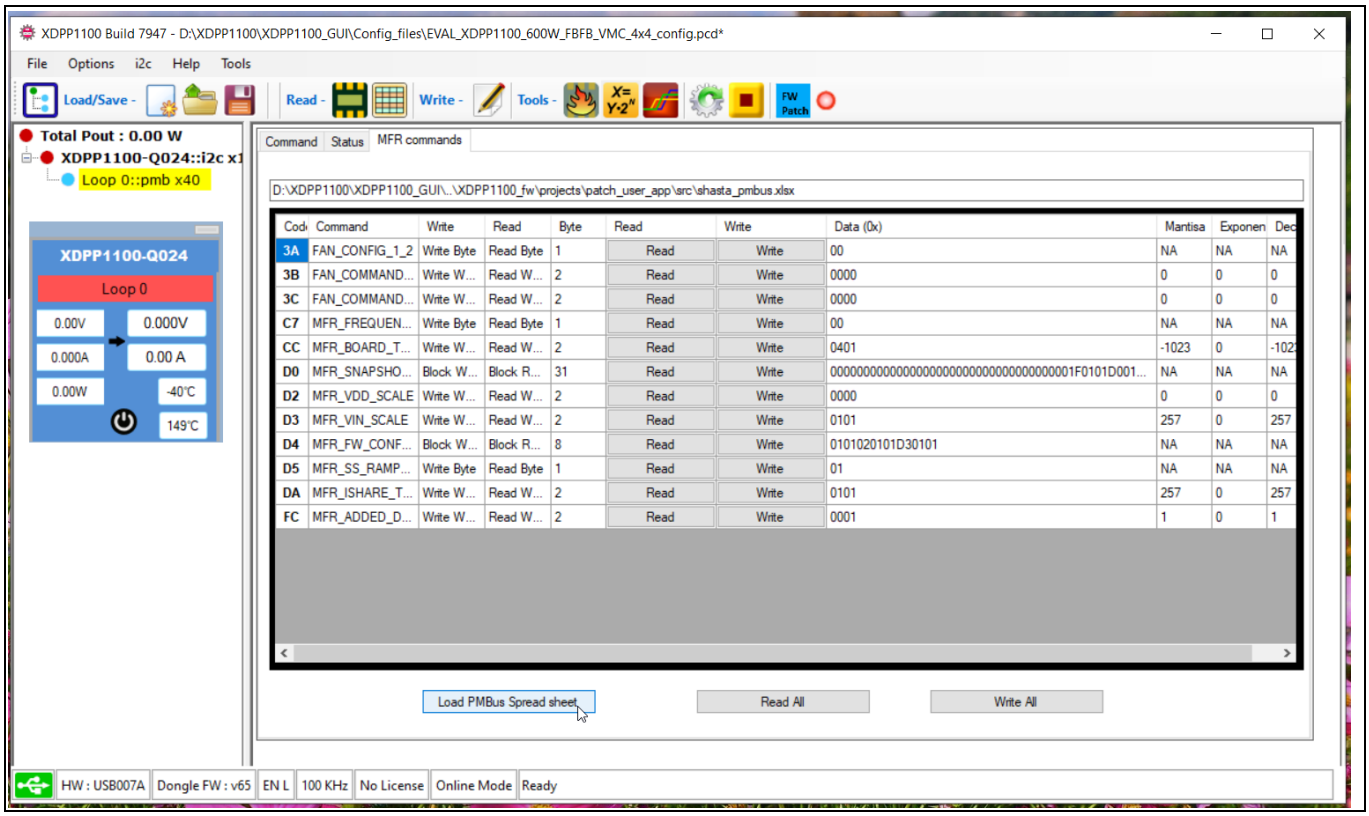


Figure 62 XDPP1100 GUI – MFR commands tab

Select shasta_pmbus.xlsx in \project\patch_example_4\src.

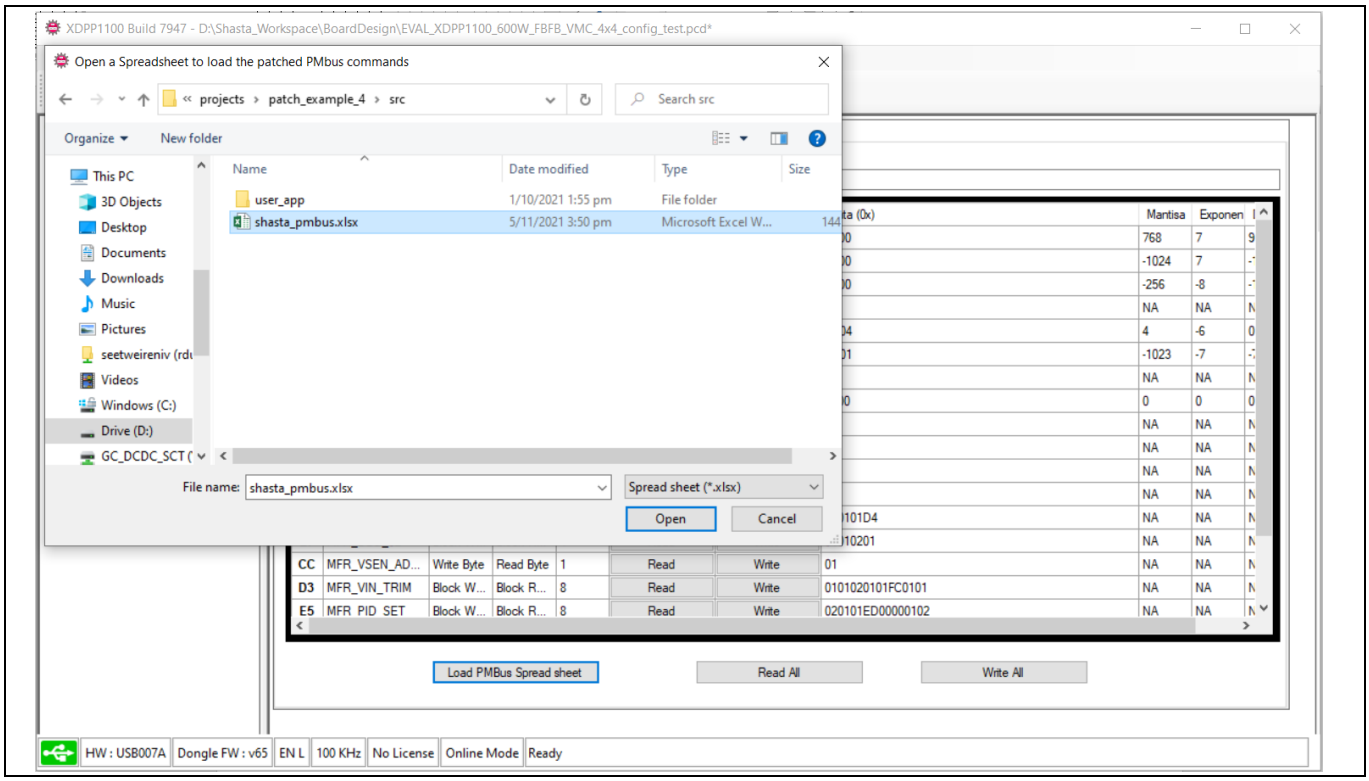


Figure 63 XDPP1100 GUI – open latest MFR commands spreadsheet

The spreadsheet is loaded as shown in **Figure 59**. Click the “Write” button in the top panel to write all register map values and PMBus commands values into RAM. Write the setting for 0xB2 and 0xB3. 0xB2 is set to 5000 (5 V at V_{OUT} mode -12) and 0xB3 to B8 (disable retry continuously).

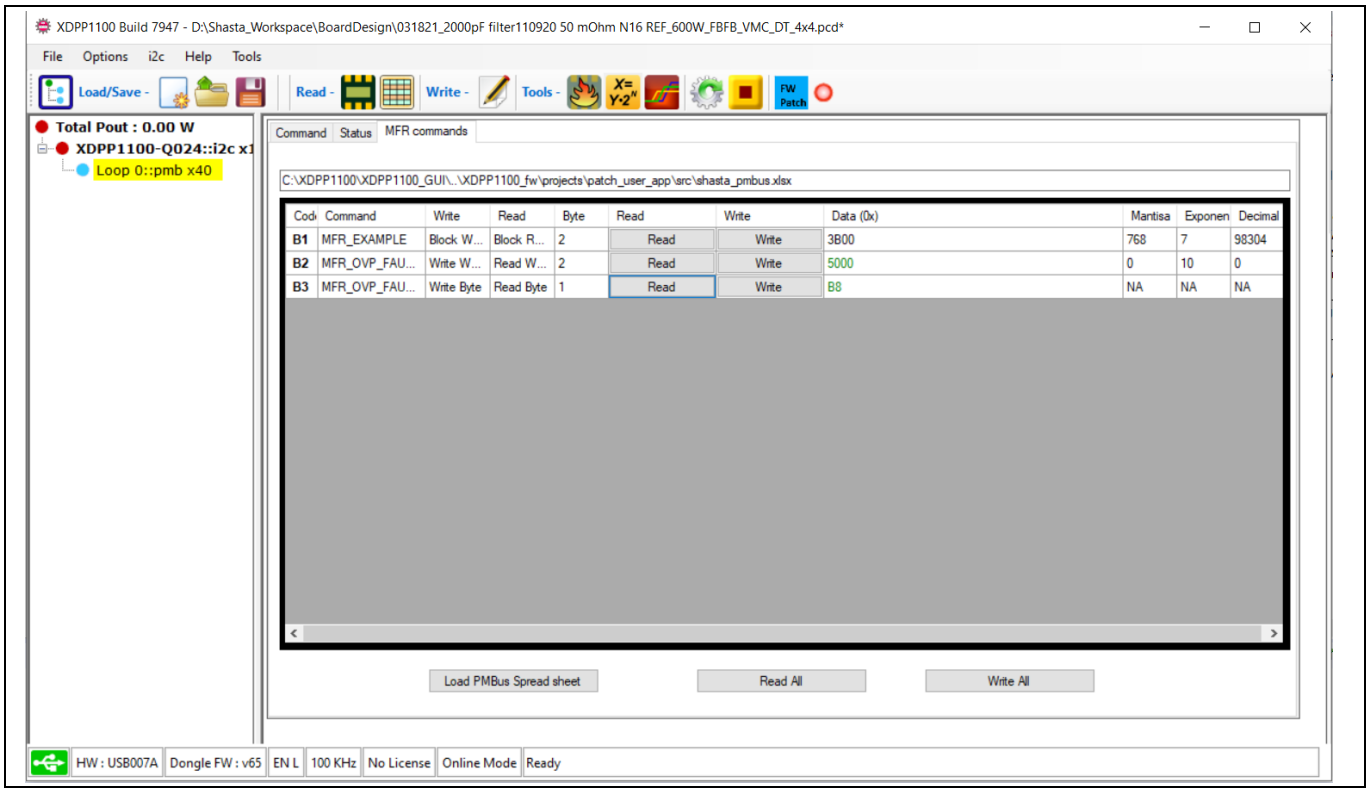


Figure 64 XDPP1100 GUI – MFR commands tab with latest spreadsheet

Right-click “XDPP1100-Q024::i2c” in the left panel and press “Clear Fault” to clear all faults after writing to RAM.

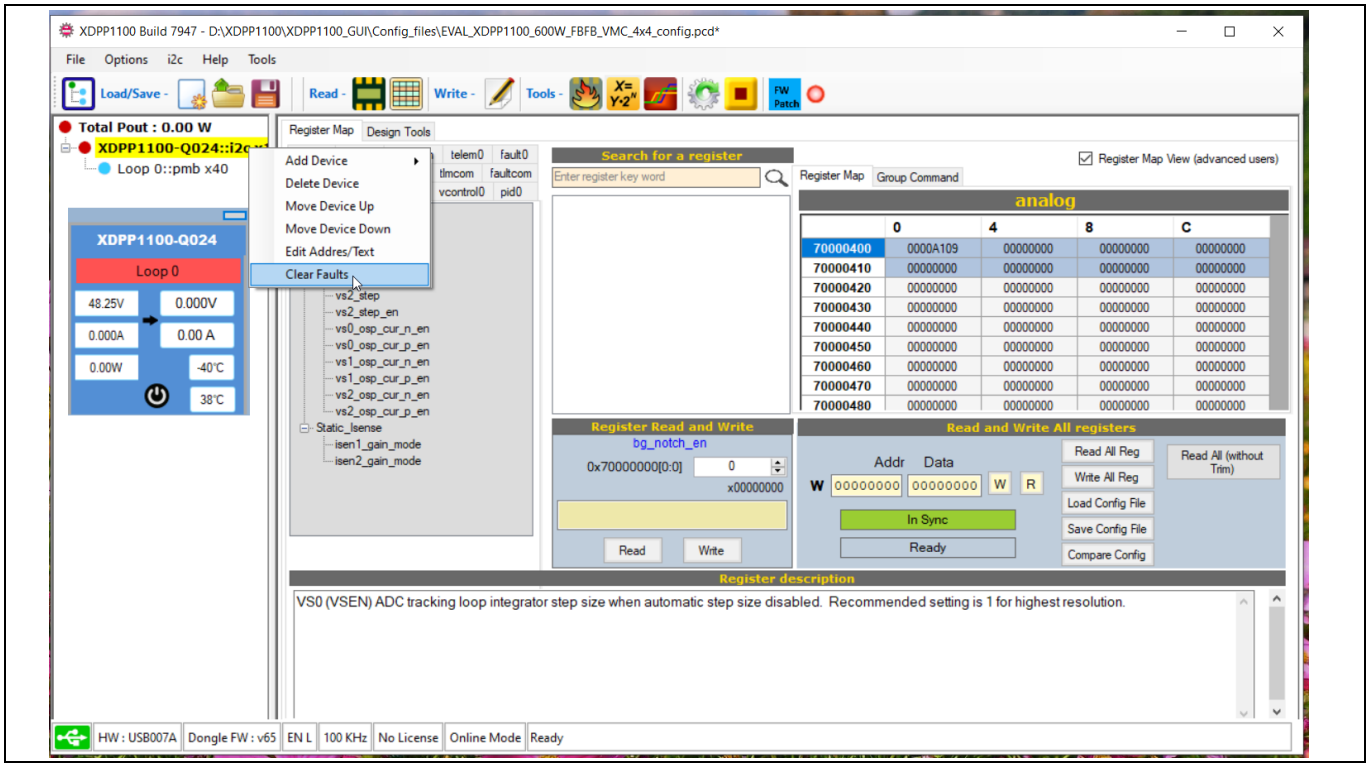


Figure 65 XDPP1100 GUI – clear faults

Click “EN H/EN L” to start the PWM regulating. Change the V_{OUT} ADC to trigger the fault.

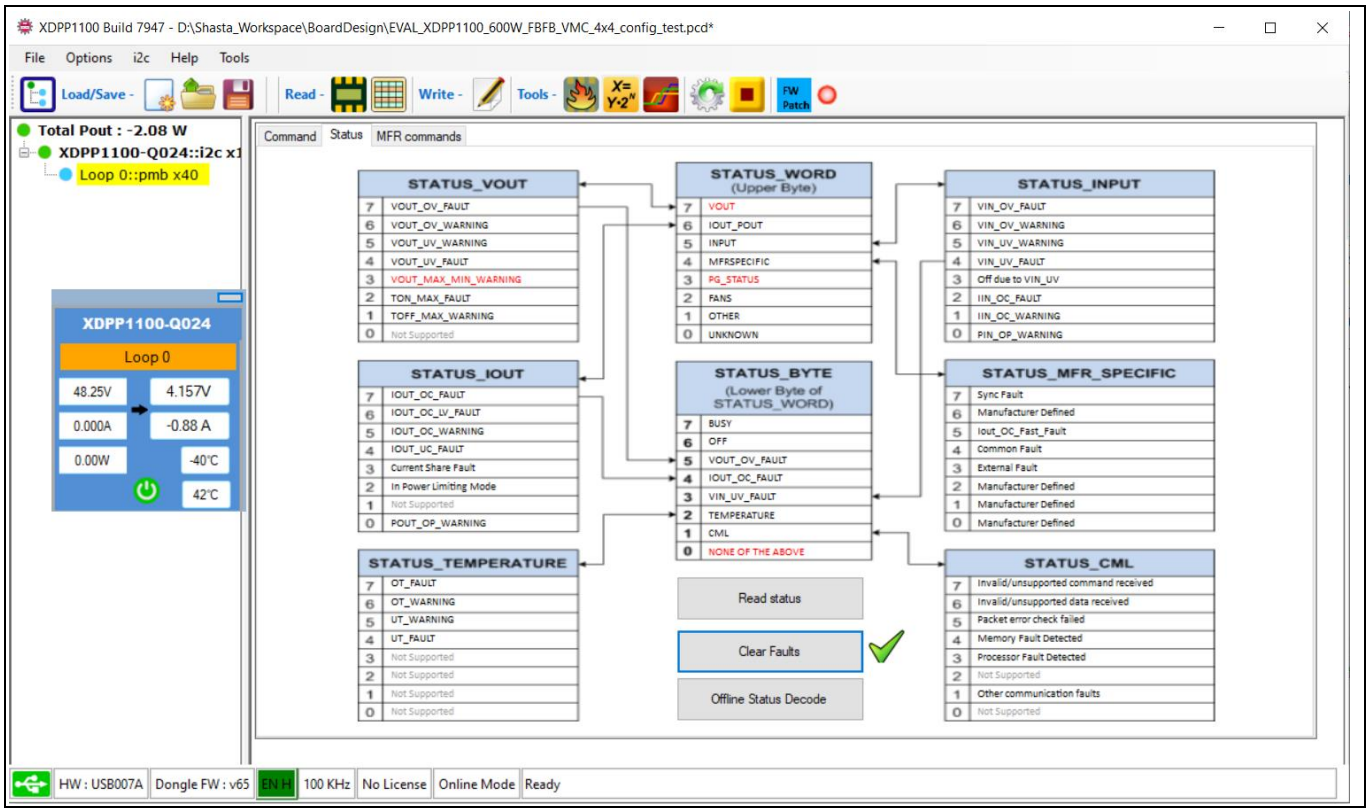


Figure 66 XDPP1100 GUI – running

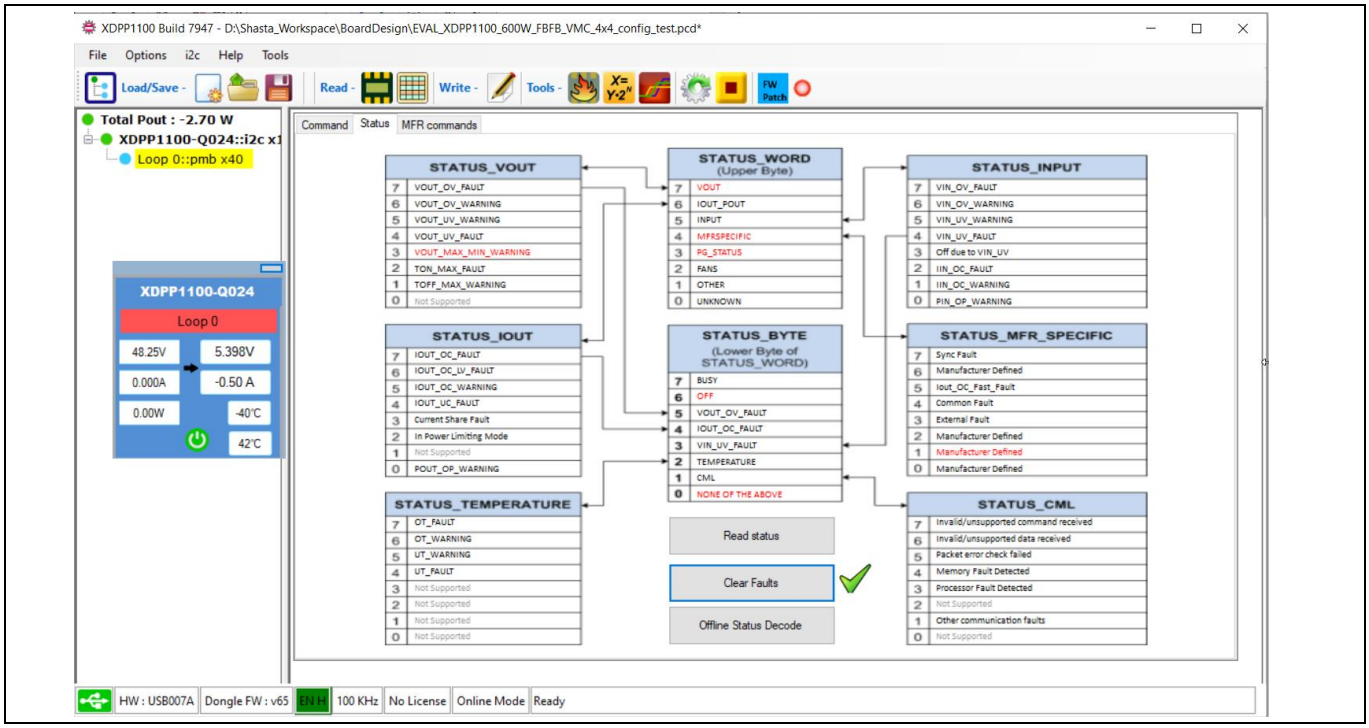


Figure 67 XDPP1100 GUI – running

6 PWM kickstart pulse

This example is taken from **XDP™ digital power XDPP1100 firmware examples [1]**.

This example shows how to output PWM signal before TON_RISE state with forced duty cycle.

To set up a new FW patch project patch_example_5, refer to **Section 2.1** Create new project.

6.1 Modify source code

Table 4 Additions/Modifications to create a custom V_{IN} telemetry IRQ

Filename	Function name
kickstart_pulse.c/h	void force_ramp_dutycycle (PMBUS_PAGE_t loop, uint32_t duty); void pulse_kickstart (PMBUS_PAGE_t loop, uint16_t pulse_width_ns); void release_ramp_dutycycle (PMBUS_PAGE_t loop); void RAMP0_T2_IRQHandler2(void);
regulation_state_machine_callbacks.c	void regulation_sm_callbacks_init(void) void TON_DELAY_END(PMBUS_PAGE_t loop) void AT_SHUTDOWN(PMBUS_PAGE_t loop)
regulation_state_machine_callbacks.h	USER_DATA_t; void regulation_sm_callbacks_init(void) void TON_DELAY_END(PMBUS_PAGE_t loop) void AT_SHUTDOWN(PMBUS_PAGE_t loop)
user_app.c	void user_drv_init(void)

6.1.1 Add kickstart pulse function

Create new source file and header file kickstart_pulse.c and kickstart_pulse.h.

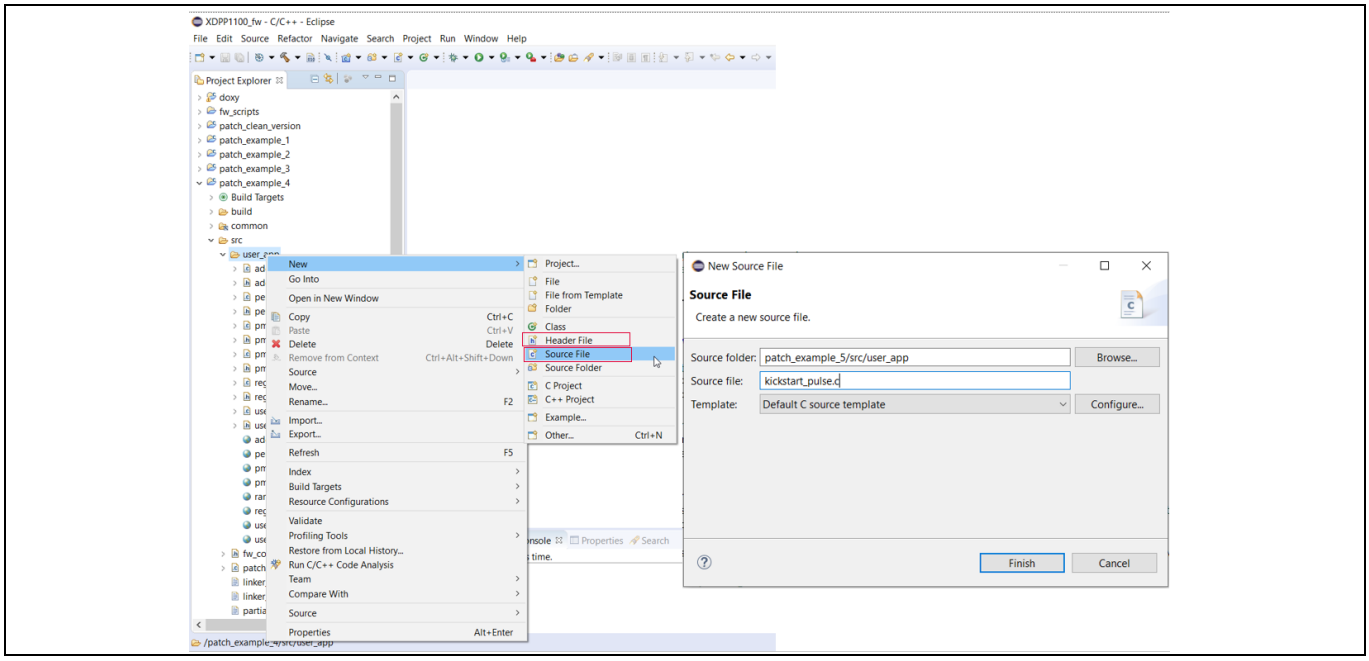


Figure 68 Create source file kickstart_pulse.c

PWM kickstart pulse

Open kickstart_pulse.c and insert the necessary function to initialize telemetry interrupt. Add the source code accordingly.

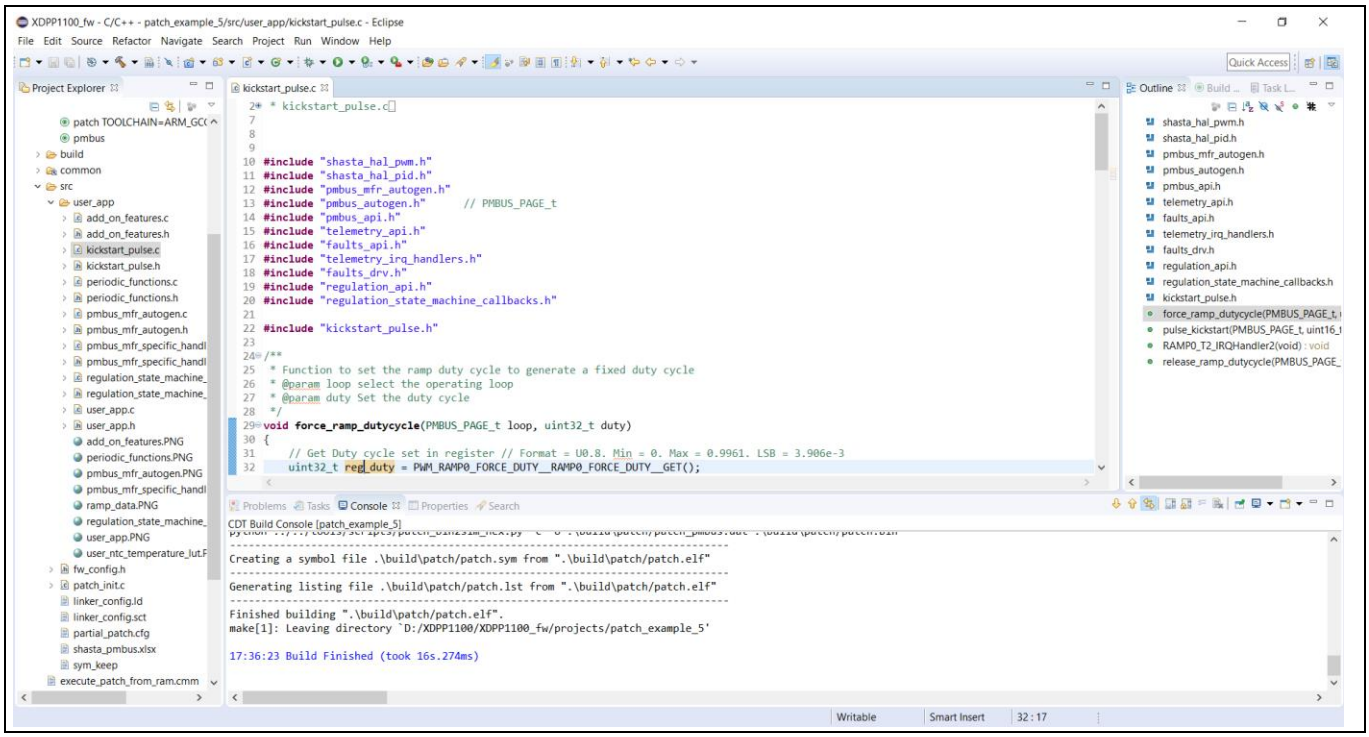


Figure 69 kickstart_pulse.c

Code Listing 12 kickstart_pulse.c

```
#include "shasta_hal_pwm.h"
#include "shasta_hal_pid.h"
#include "pmbus_mfr_autogen.h"
#include "pmbus_autogen.h"       // PMBUS_PAGE_t
#include "pmbus_api.h"
#include "telemetry_api.h"
#include "faults_api.h"
#include "telemetry_irq_handlers.h"
#include "faults_drv.h"
#include "regulation_api.h"
#include "regulation_state_machine_callbacks.h"

#include "kickstart_pulse.h"

/**
 * Function to set the ramp duty cycle to generate a fixed duty cycle
 * @param loop select the operating loop
 * @param duty Set the duty cycle
 */
void force_ramp_dutycycle(PMBUS_PAGE_t loop, uint32_t duty)
{
    // Get Duty cycle set in register // Format = U0.8. Min = 0. Max = 0.9961.
LSB = 3.906e-3
    uint32_t reg_duty = PWM_RAMP0_FORCE_DUTY__RAMP0_FORCE_DUTY__GET();

    if (reg_duty !=0) // register duty cycle not set use duty set in software
    {
        duty = reg_duty;
```

Code Listing 12 kickstart_pulse.c

```
    }

    // Set the force duty cycle for pwm
    if(loop == PMBUS_PAGE_0){
        PWM_RAMP0_FORCE_DUTY__RAMP0_FORCE_DUTY__SET(duty); //U0.8 //
FORCE_DUTY
    }
    else{
        PWM_RAMP1_FORCE_DUTY__RAMP1_FORCE_DUTY__SET(duty); //U0.8 //
FORCE_DUTY
    }

    // Enable FORCE DUTY
    if(loop == PMBUS_PAGE_0){
        PWM_RAMP0_FORCE_DUTY__RAMP0_FORCE_DUTY_EN__SET(1);
    }
    else{
        PWM_RAMP1_FORCE_DUTY__RAMP1_FORCE_DUTY_EN__SET(1);
    }

    for(uint16_t i = 0; i < 2; i++); // For-loop delay before disabling forced
duty cycle
}

/**
 * Function to initialize and start the kickstart pulse
 * @param loop select the operating loop
 * @param pulse_width_ns set the duty cycle in pulse width
 */
void pulse_kickstart(PMBUS_PAGE_t loop, uint16_t pulse_width_ns)
{
    // Get current common register setting
    uint32_t reg1 = COMMON_EN_REG1__GET();
    // Get user configuration from PMBus
    uint16_t freq_khz = PMBUS_Get_Command_Word(loop,
PMBUS_CMDCODE_FREQUENCY_SWITCH);

    // Convert LINEAR11 to mantissa and exponent
    int32_t freq_khz_man = (uint16_t)LINEAR11_TO_MANTISSA(freq_khz);
    int32_t freq_khz_exp = LINEAR11_TO_EXPONENT(freq_khz);
    freq_khz = (uint16_t)SHIFT_EXPONENT(freq_khz_man, freq_khz_exp);

    // Set Switching Frequency for pulse kickstart. API can be found on
regulation_api.h
    Regulation_Switching_Frequency_Set(loop, freq_khz);

    // calculate period_in_ns
    uint32_t period_ns = 1000000/freq_khz;

    // Initialize forced_duty to zero
    uint32_t forced_duty = 0;

    // translate desired_pulse_ns/period_in_ns to duty_cycle_percentage and
calculate forced_duty in hex
    forced_duty = (pulse_width_ns*0xFF)/period_ns;

    // enable the pwm module for loop 0.
    reg1 |= COMMON_EN_REG1__EN_PWM0__Msk;
    COMMON_EN_REG1__SET(reg1);
```

Code Listing 12 kickstart_pulse.c

```
    user_data.ramp0_count = 0; // counter for ramp interrupt

    Regulation_PWM_Control(loop, REGULATION_SRFET_OFF);  // primary side PWM's
on, SR's off

    // Pulse kickstart by forcing duty cycle
    force_ramp_dutycycle(loop, forced_duty);

    //Enable T2 IRQ
    PWM_RAMP_IRQ__RAMP0_T2_IRQ_SEL__SET(1); //enable T2
    PWM_RAMP_IRQ__RAMP0_IRQ_RATE__SET(0);    //Set IRQ rate to every Tswitch
    sys_irq_handler_set((IRQn_Type) RAMP0_T2_IRQn,
                (Irq_Handler) RAMP0_T2_IRQHandler2); /* Put interrupt handler
into vector table */
    /* Cortex specific signals for enabling IRQs */
    NVIC_ClearPendingIRQ(RAMP0_T2_IRQn);
    NVIC_EnableIRQ(RAMP0_T2_IRQn);

}

/**
 * lint -esym(714, tlm_irq_enable) -esym(759, tlm_irq_enable) -esym(765,
tlm_irq_enable)
 * This function is the handler function of T2 to set the T2 value
accordingly to pid duty cycle
 */
void RAMP0_T2_IRQHandler2(void)
{

    user_data.ramp0_count++;
    if (user_data.ramp0_count == 6)
    {
        user_data.ramp0_count = 0;
        // OFF all pwm signal
        PWM_PWM_ON__PWM_ON__SET(0x00);

        // disable forced duty cycle
        release_ramp_dutycycle(PMBUS_PAGE_0);

        PWM_RAMP_IRQ__RAMP0_T2_IRQ_SEL__SET(0); //disable T2 interrupt
        NVIC_ClearPendingIRQ(RAMP0_T2_IRQn);
        /* Cortex specific signals for disabling IRQs */
        NVIC_DisableIRQ(RAMP0_T2_IRQn);
        return;
    }
    NVIC_ClearPendingIRQ(RAMP0_T2_IRQn);

}

/**
 * Function to release force PID duty cycle
 * @param loop select the operating loop
 */
void release_ramp_dutycycle(PMBUS_PAGE_t loop)
{
    // Get user configuration from PMBus
    uint16_t freq_khz = PMBUS_Get_Command_Word(loop,
PMBUS_CMDCODE_FREQUENCY_SWITCH);
```

Code Listing 12 kickstart_pulse.c

```
        // Convert LINEAR11 to mantissa and exponent
        int32_t freq_khz_man = (uint16_t)LINEAR11_TO_MANTISSA(freq_khz);
        int32_t freq_khz_exp = LINEAR11_TO_EXPONENT(freq_khz);
        freq_khz = (uint16_t)SHIFT_EXPONENT(freq_khz_man, freq_khz_exp);

        // Update Switching frequency with the user configuration from PMBus
        Regulation_Switching_Frequency_Set(loop, freq_khz);

        // Set the force duty cycle for pwm to 0
        if(loop == PMBUS_PAGE_0){
            PWM_RAMP0_FORCE_DUTY__RAMP0_FORCE_DUTY__SET(0);     //U0.8 //
    FORCE_DUTY
        }
        else{
            PWM_RAMP1_FORCE_DUTY__RAMP1_FORCE_DUTY__SET(0);     //U0.8 //
    FORCE_DUTY
        }

        // Disable FORCE DUTY
        if(loop == PMBUS_PAGE_0){
            PWM_RAMP0_FORCE_DUTY__RAMP0_FORCE_DUTY_EN__SET(0);
        }
        else{
            PWM_RAMP1_FORCE_DUTY__RAMP1_FORCE_DUTY_EN__SET(0);
        }

    }
```

6.1.2 Add header file initialization

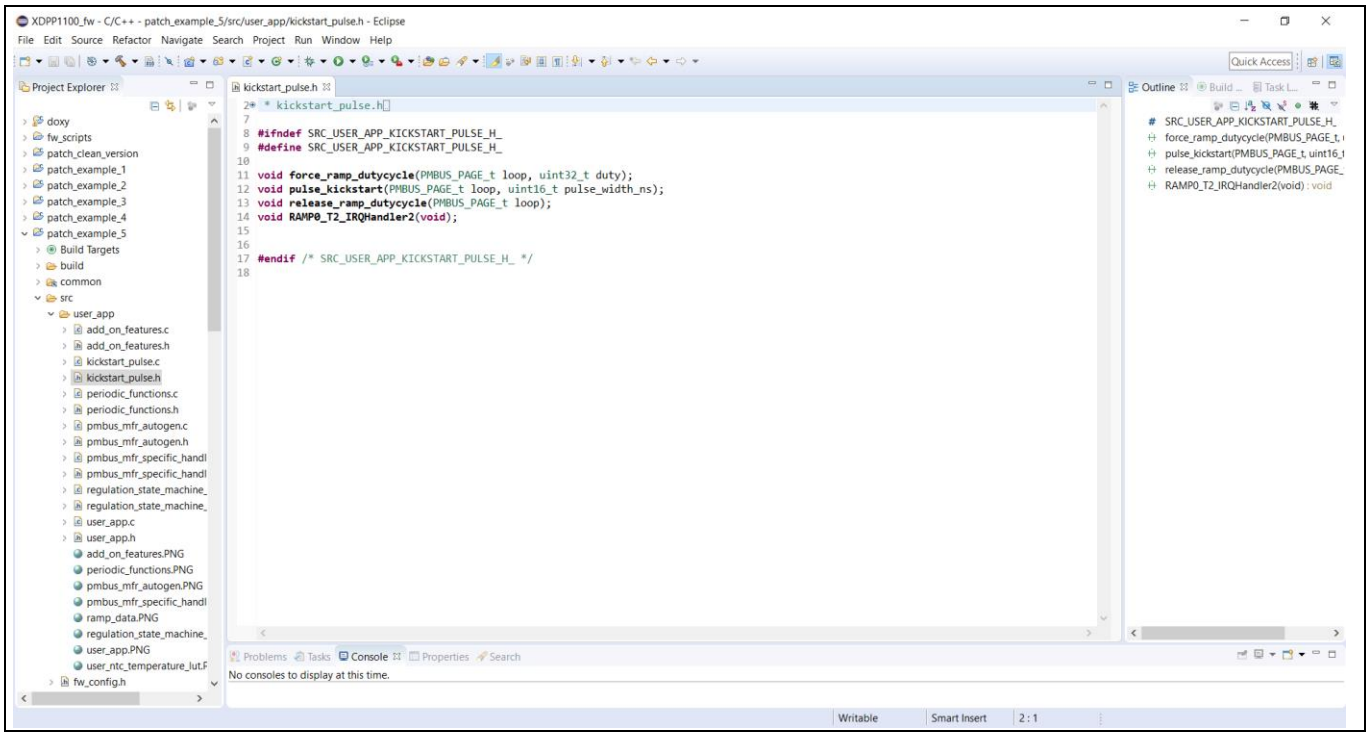


Figure 70 kickstart_pulse.h

Code Listing 13 kickstart_pulse.h

```
void force_ramp_dutycycle(PMBUS_PAGE_t loop, uint32_t duty);
void pulse_kickstart(PMBUS_PAGE_t loop, uint16_t pulse_width_ns);
void release_ramp_dutycycle(PMBUS_PAGE_t loop);
void RAMP0_T2_IRQHandler2(void);
```

6.1.3 User data variable and regulation callback function

Declare variable use in the user data structure and function use in regulation_state_machine_callback.c.

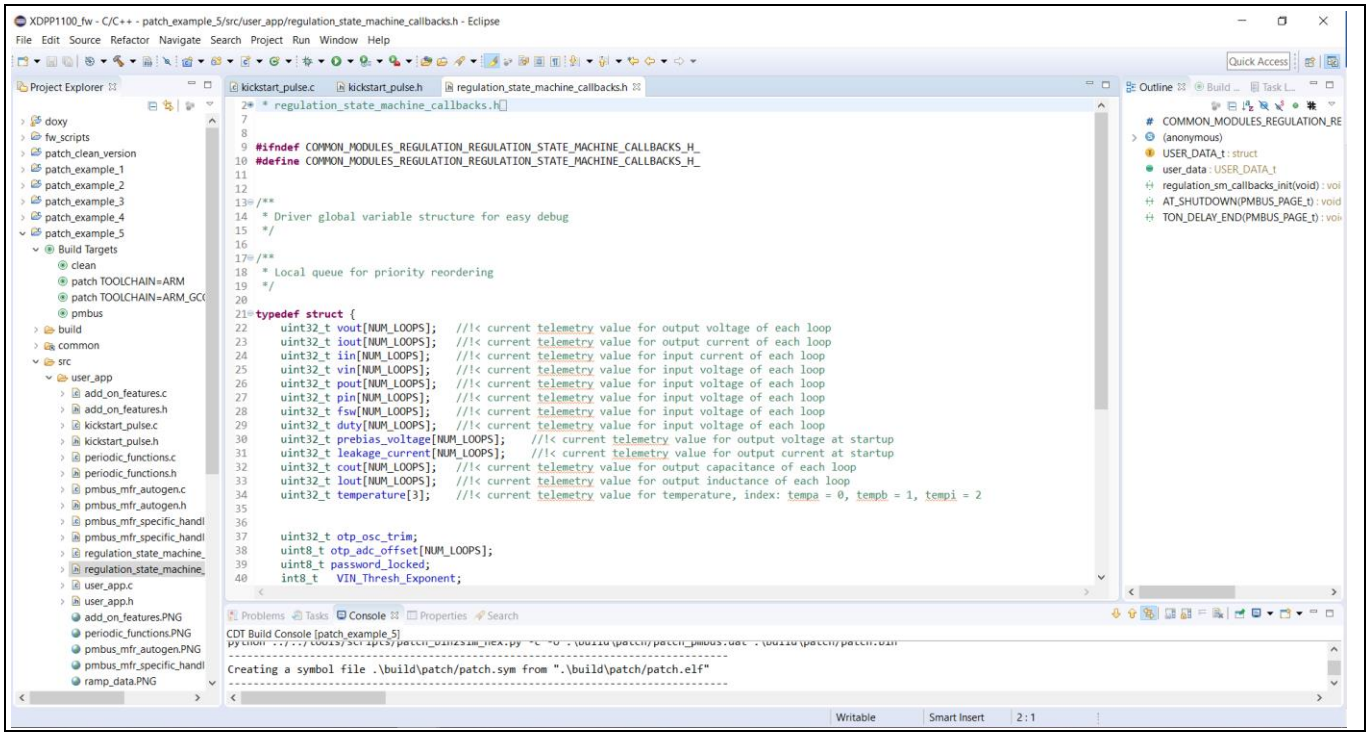


Figure 71 regulation_state_machine_callback.h

Code Listing 14 regulation_state_machine_callback.h

```
typedef struct {
    uint32_t vout[NUM_LOOPS]; //!< current telemetry value for output voltage
of each loop
    uint32_t iout[NUM_LOOPS]; //!< current telemetry value for output current
of each loop
    uint32_t iin[NUM_LOOPS];  //!< current telemetry value for input current
of each loop
    uint32_t vin[NUM_LOOPS];  //!< current telemetry value for input voltage
of each loop
    uint32_t pout[NUM_LOOPS]; //!< current telemetry value for input voltage
of each loop
    uint32_t pin[NUM_LOOPS];  //!< current telemetry value for input voltage
of each loop
    uint32_t fsw[NUM_LOOPS];  //!< current telemetry value for input voltage
of each loop
    uint32_t duty[NUM_LOOPS]; //!< current telemetry value for input voltage
of each loop
```

Code Listing 14 regulation_state_machine_callback.h

```
    uint32_t prebias_voltage[NUM_LOOPS];     //!< current telemetry value for
output voltage at startup
    uint32_t leakage_current[NUM_LOOPS];     //!< current telemetry value for
output current at startup
    uint32_t cout[NUM_LOOPS]; //!< current telemetry value for output
capacitance of each loop
    uint32_t lout[NUM_LOOPS]; //!< current telemetry value for output
inductance of each loop
    uint32_t temperature[3];  //!< current telemetry value for temperature,
index: tempa = 0, tempb = 1, tempi = 2


    uint32_t otp_osc_trim;
    uint8_t otp_adc_offset[NUM_LOOPS];
    uint8_t password_locked;
    int8_t   VIN_Thresh_Exponent;
    uint32_t VIN_Thresh;
    uint32_t hash;

    // Kickstart Pulse
    uint8_t ramp0_count;



} USER_DATA_t;

extern USER_DATA_t user_data;

void regulation_sm_callbacks_init(void);
void AT_SHUTDOWN(PMBUS_PAGE_t loop);
void TON_DELAY_END(PMBUS_PAGE_t loop);
```

6.1.4 Regulation callback function

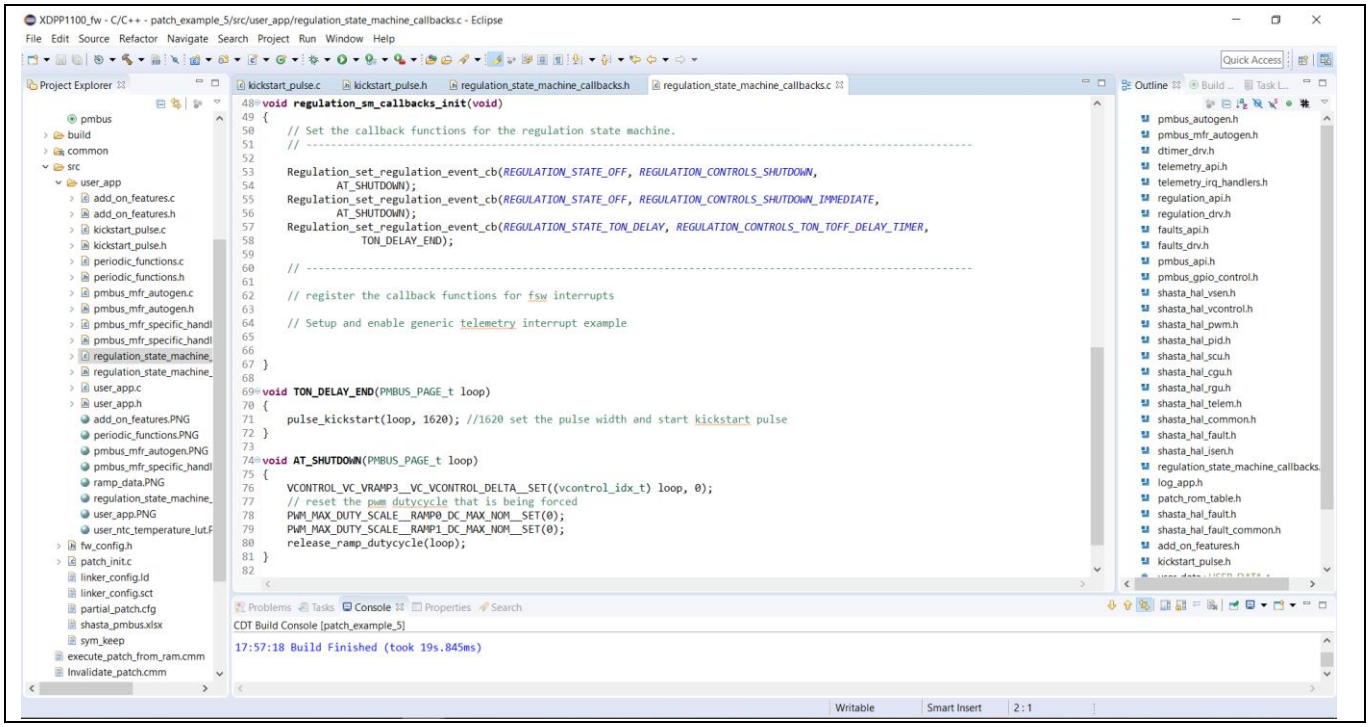


Figure 72 regulation_state_machine.c

Code Listing 15 regulation_state_machine.c

```
#include "kickstart_pulse.h"

USER_DATA_t user_data;
void regulation_sm_callbacks_init(void)
{
    // Set the callback functions for the regulation state machine.
    // ------------------------------------------------------------------------
---------------------------------

    Regulation_set_regulation_event_cb(REGULATION_STATE_OFF,
REGULATION_CONTROLS_SHUTDOWN,
           AT_SHUTDOWN);
    Regulation_set_regulation_event_cb(REGULATION_STATE_OFF,
REGULATION_CONTROLS_SHUTDOWN_IMMEDIATE,
           AT_SHUTDOWN);
    Regulation_set_regulation_event_cb(REGULATION_STATE_TON_DELAY,
REGULATION_CONTROLS_TON_TOFF_DELAY_TIMER,
                        TON_DELAY_END);

    // ------------------------------------------------------------------------
---------------------------------

    // register the callback functions for fsw interrupts

    // Setup and enable generic telemetry interrupt example


}

void TON_DELAY_END(PMBUS_PAGE_t loop)
{
    pulse_kickstart(loop, 1620); //1620 set the pulse width and start kickstart
pulse
}

void AT_SHUTDOWN(PMBUS_PAGE_t loop)
{
    VCONTROL_VC_VRAMP3__VC_VCONTROL_DELTA__SET((vcontrol_idx_t) loop, 0);
    // reset the pwm dutycycle that is being forced
    PWM_MAX_DUTY_SCALE__RAMP0_DC_MAX_NOM__SET(0);
    PWM_MAX_DUTY_SCALE__RAMP1_DC_MAX_NOM__SET(0);
    release_ramp_dutycycle(loop);
}
```

6.1.5 User app

Open the the user_app.c and insert source code according to **Code Listing 2** in void user_drv_init(void) to run the FW patch in RAM.

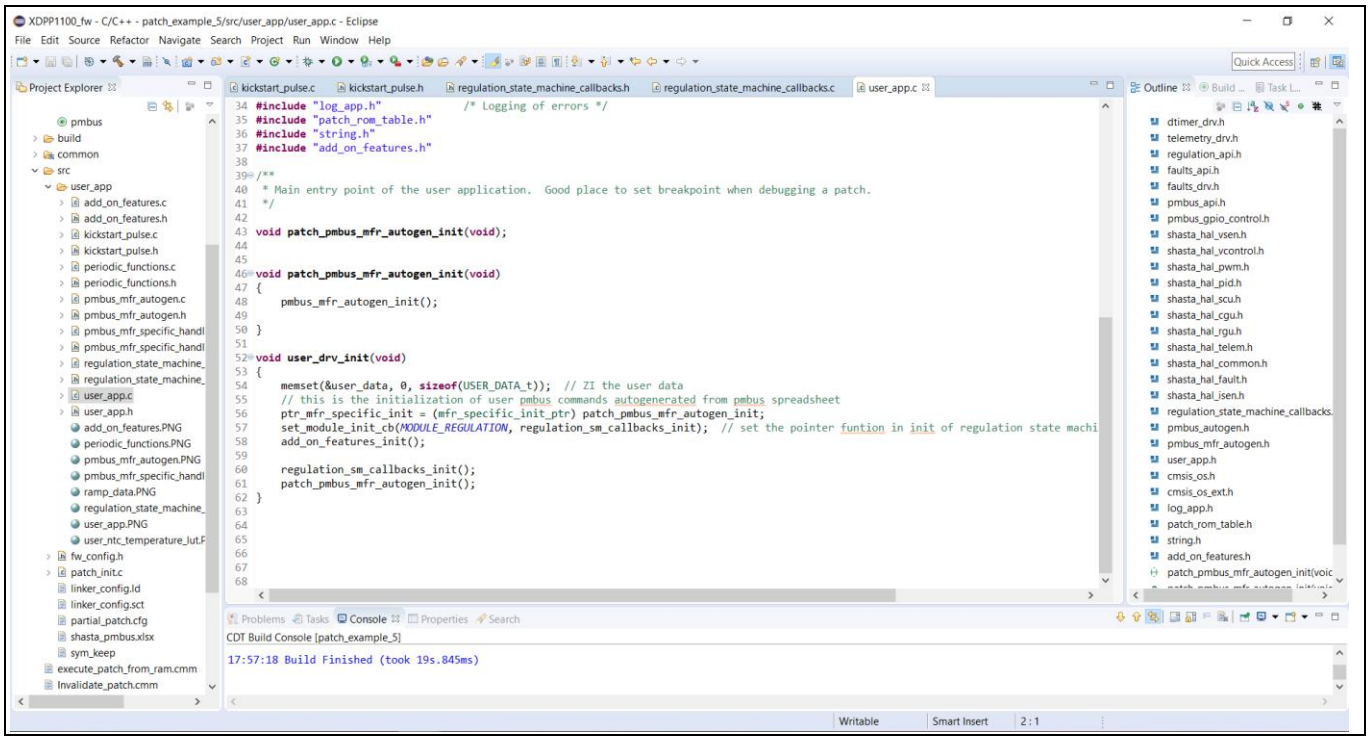


Figure 73 user_app.c

Code Listing 16 user_app.c

```
void user_drv_init(void)
{
  memset(&user_data, 0, sizeof(USER_DATA_t));  // ZI the user data
  // 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();

  // Added lines to run patch in RAM
   regulation_sm_callbacks_init();
   patch_pmbus_mfr_autogen_init();
}
```

6.2 Compile and program to XDPP1100 RAM/OTP

To compile and program the FW patch refer to **Section 3.3** Compile and program to XDPP1100 RAM/OTP.

6.2.1 Running example 5 patch

Click “EN H/EN L” to start the PWM regulating. Use an oscilloscope to confirm initial kickstart pulse upon enable pin go high.

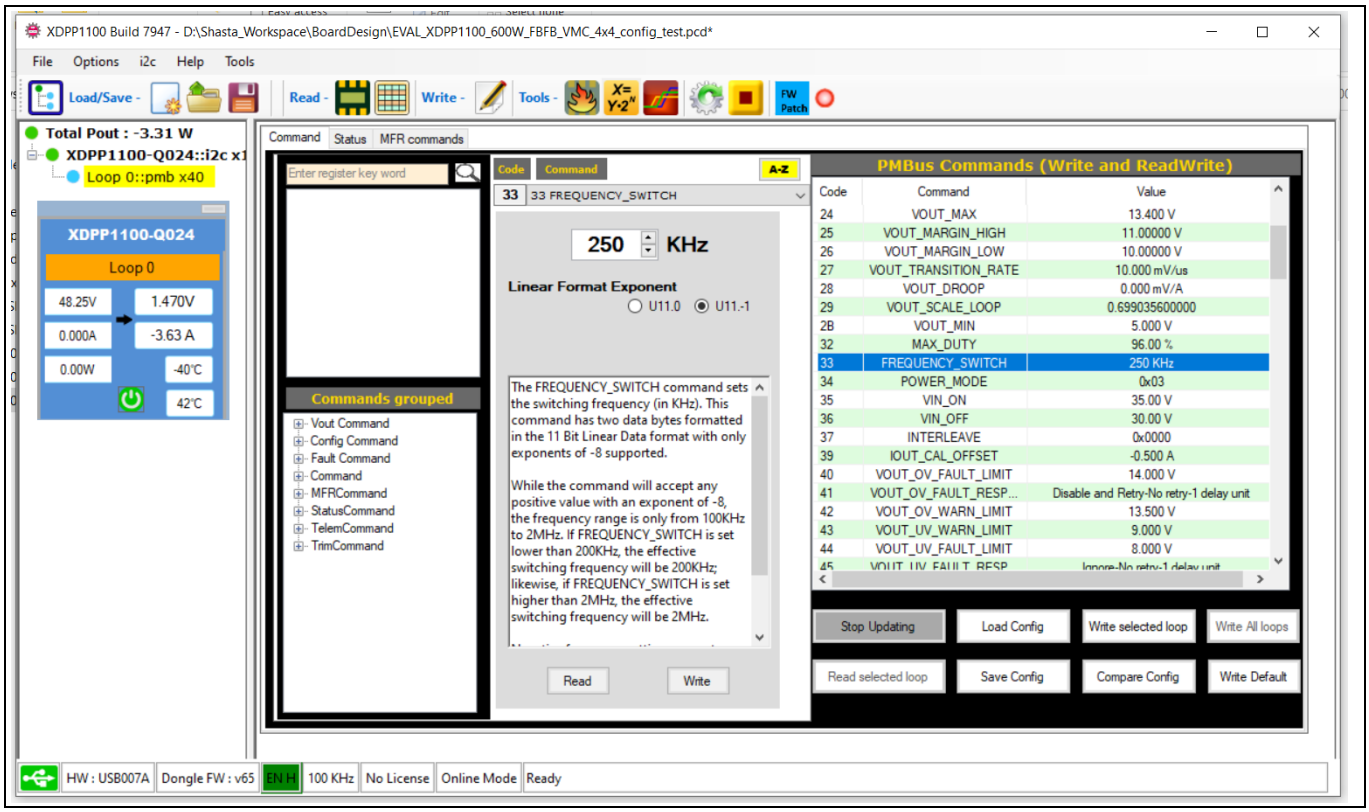


Figure 74 XDPP1100 GUI

7 Delay secondary PWM start

This example shows the use of D_{TIMER}, frequency switch interrupt, regulation callback and using RAMP 1 in a loop to control the PWM pin assigned to the secondary side.

To set up a new FW patch project patch_example_6, refer to **Section 2.1** Create new project.

7.1 Modify source code

Table 5 Additions/Modifications to create a custom V_{IN} telemetry IRQ

Filename	Function name

softstart_srpwm.c	STATIC DTIMER_CONFIG_t dtimer4_cfg;
	STATIC DRIVER_DTIMER *drv4_info;
	void DTIMER4_Sr_PWM_Timer_IRQHandler(void);
	void SR_PWM_Timer_Activation (void);
	void Sr_PWM_Timer_Stop(PMBUS_PAGE_t loop);
	void sr_pwm_setting(uint8_t select);
	void RAMP1_Setting(void);
	void sr_pwm_fsw_irq_enable(PMBUS_PAGE_t loop);
	void sr_pwm_fsw_irq_callback (PMBUS_PAGE_t loop);

softstart_srpwm.h	void SR_PWM_Timer_Activation (void);
	void Sr_PWM_Timer_Stop(PMBUS_PAGE_t loop);
	void sr_pwm_setting(uint8_t select);
	void RAMP1_Setting(void);
	void sr_pwm_fsw_irq_enable(PMBUS_PAGE_t loop);
	void sr_pwm_fsw_irq_callback (PMBUS_PAGE_t loop);

regulation_state_machine_callbacks.c	void regulation_sm_callbacks_init(void) void TON_Rise(PMBUS_PAGE_t loop) void AT_SHUTDOWN(PMBUS_PAGE_t loop)
regulation_state_machine_callbacks.h	USER_DATA_t; void regulation_sm_callbacks_init(void) void TON_RISE(PMBUS_PAGE_t loop) void AT_SHUTDOWN(PMBUS_PAGE_t loop)
user_app.c	void user_drv_init(void)

7.1.1 Add soft-start SR PWM function

Create new source file and header file softstart_srpwm.c and softstart_srpwm.h.

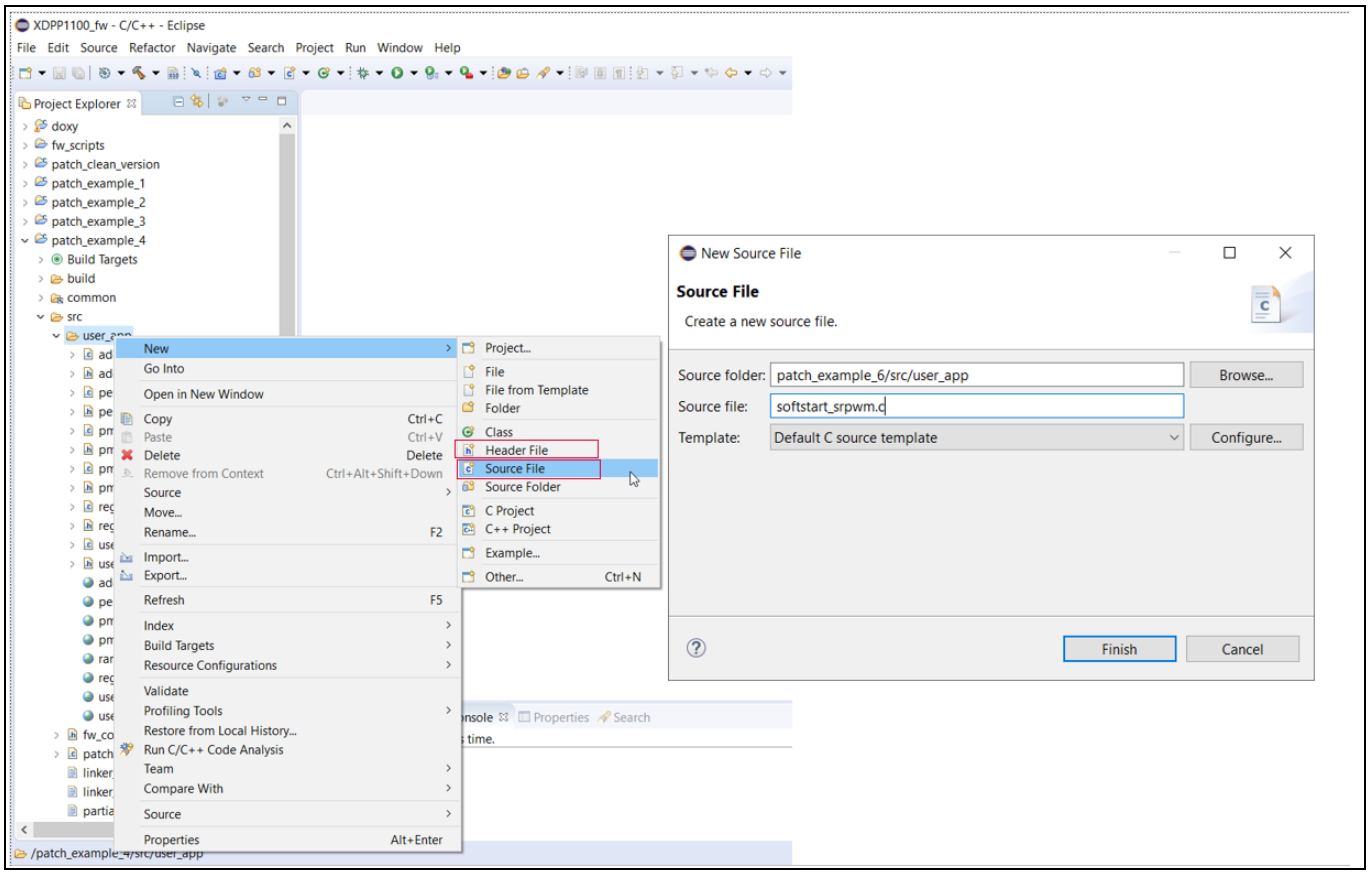


Figure 75 Create source file softstart_srpwm.c

Open softstart_srpwm.c and insert the necessary function to initialize D_{TIMER} and RAMP setting. Add the source code accordingly.

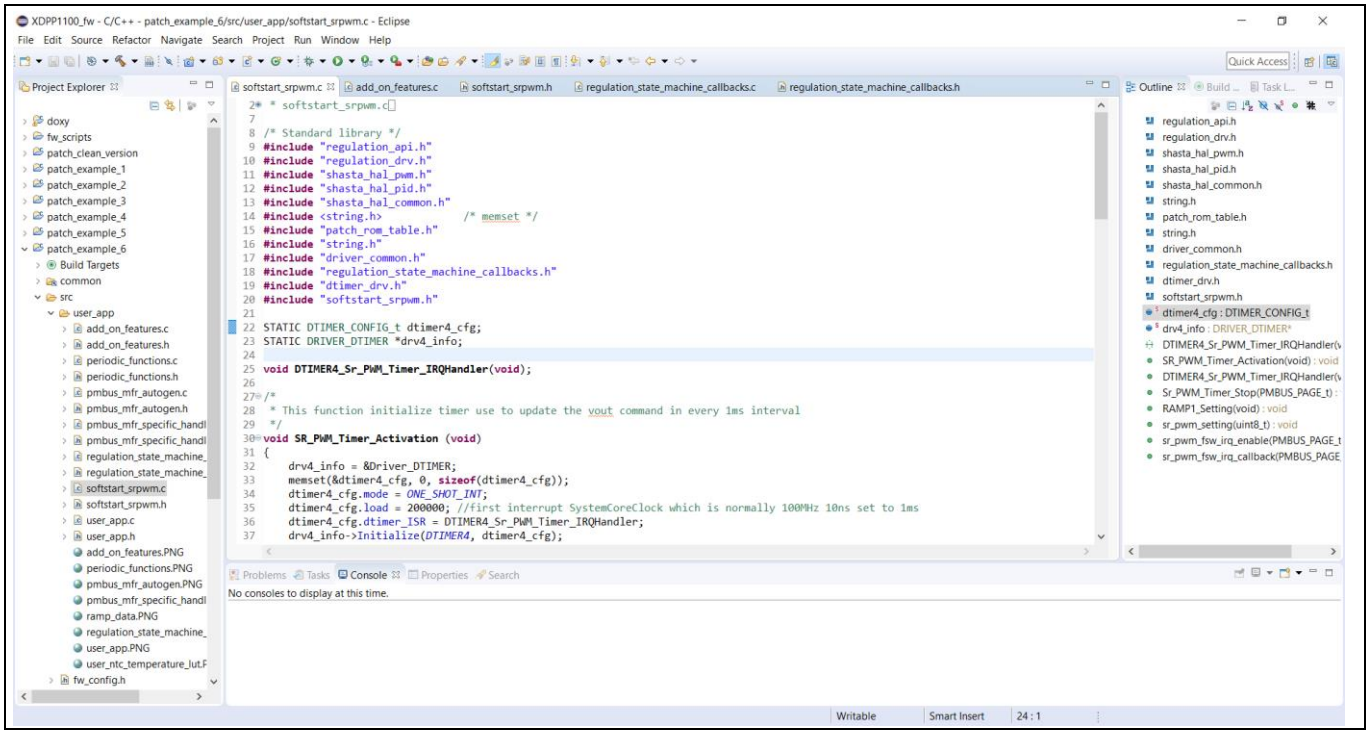


Figure 76 softstart_srpwm.c

Code Listing 17 softstart_srpwm.c

```
#include "regulation_api.h"
#include "regulation_drv.h"
#include "shasta_hal_pwm.h"
#include "shasta_hal_pid.h"
#include "shasta_hal_common.h"
#include <string.h>              /* memset */
#include "patch_rom_table.h"
#include "string.h"
#include "driver_common.h"
#include "regulation_state_machine_callbacks.h"
#include "dtimer_drv.h"
#include "softstart_srpwm.h"

STATIC DTIMER_CONFIG_t dtimer4_cfg;
STATIC DRIVER_DTIMER *drv4_info;

void DTIMER4_Sr_PWM_Timer_IRQHandler(void);

/*
 * This function initialize timer use to update the vout command in every 1ms
interval
 */
void SR_PWM_Timer_Activation (void)
{
    drv4_info = &Driver_DTIMER;
    memset(&dtimer4_cfg, 0, sizeof(dtimer4_cfg));
    dtimer4_cfg.mode = ONE_SHOT_INT;
    dtimer4_cfg.load = 200000; //first interrupt SystemCoreClock which is
normally 100MHz 10ns set to 1ms
    dtimer4_cfg.dtimer_ISR = DTIMER4_Sr_PWM_Timer_IRQHandler;
    drv4_info->Initialize(DTIMER4, dtimer4_cfg);
}

/**
 * Interrupt service routines for DTIMER4 use to update vout command
*/
void DTIMER4_Sr_PWM_Timer_IRQHandler(void)
{
    sr_pwm_setting(1);
    Regulation_PWM_Control(PMBUS_PAGE_0, REGULATION_SRFET_ON);  // SRFET's off
    sr_pwm_fsw_irq_enable(PMBUS_PAGE_0);
     drv4_info->ClearIRQ(DTIMER4);
}

/**
 * Stop DTIMER4
*/
void Sr_PWM_Timer_Stop(PMBUS_PAGE_t loop)
{
    if (drv4_info != NULL)
    {
        drv4_info->ClearIRQ(DTIMER4);
        drv4_info->Uninitialize(DTIMER4);
    }
}

void RAMP1_Setting(void)
{
    PWM_RAMP1_TSWITCH1__TSWITCH1__SET(PWM_RAMP0_TSWITCH0__TSWITCH0__GET());
```

Code Listing 17 softstart_srpwm.c

```
        PWM_RAMP_CONFIG__RAMP1_SYNC_SEL__SET(1);
        PWM_RAMP1_PHASE__RAMP1_PHASE__SET(0);
        PWM_RAMP_CONFIG__RAMP1_HALF_MODE__SET(1);
        PWM_RAMP_CONFIG__RAMP1_PID_SEL__SET(0);
        PWM_RAMP_CONFIG__RAMP1_M_FLAVOR__SET(2);
        COMMON_EN_REG2__EN_RAMP1__SET(1);
        PWM_RAMP1_PW_MINMAX__RAMP1_PW_MIN__SET(0); //U8.0, 5ns
    }

    void sr_pwm_setting(uint8_t select)
    {
        if(select == 1)
        {
           PWM_RAMP1_TSWITCH1__TSWITCH1__SET(PWM_RAMP0_TSWITCH0__TSWITCH0__GET());

            PWM_RAMP1_FORCE_T2__RAMP1_FORCE_T2_EN__SET(1);
            PWM_RAMP1_FORCE_T2__RAMP1_FORCE_T2__SET(0);
            PWM_PWM_TOPO_2__PWM5_FALL_SEL__SET(3);
            PWM_PWM_TOPO_2__PWM5_RISE_SEL__SET(2);
            PWM_PWM_TOPO_2__PWM6_FALL_SEL__SET(5);
            PWM_PWM_TOPO_2__PWM6_RISE_SEL__SET(4);
            PWM_PWM_MAP__PWM5_LOOP_MAP__SET(3);
            PWM_PWM_MAP__PWM6_LOOP_MAP__SET(3);
        }
        else if(select ==2)
        {
            PWM_PWM_MAP__PWM5_LOOP_MAP__SET(1);
            PWM_PWM_MAP__PWM6_LOOP_MAP__SET(1);
            PWM_PWM_TOPO_2__PWM5_FALL_SEL__SET(4);
            PWM_PWM_TOPO_2__PWM5_RISE_SEL__SET(5);
            PWM_PWM_TOPO_2__PWM6_FALL_SEL__SET(2);
            PWM_PWM_TOPO_2__PWM6_RISE_SEL__SET(3);

        }
    }

    /*Enable fsw for vin sel */
    void sr_pwm_fsw_irq_enable(PMBUS_PAGE_t loop)
    {
        user_data.RAMP1_Switch_Count = 0;
        Regulation_setup_fsw_irq(PMBUS_PAGE_0, fsw_irq_idx_2, fsw_irq_rate_sel_2);
    // enable fsw irq and set the rate
    }

    /*
     * FSW IRQ callback.
     *
     */
    void sr_pwm_fsw_irq_callback (PMBUS_PAGE_t loop)
    {
        user_data.RAMP1_Switch_Count++; // 1 4cycle
        if (user_data.RAMP1_Switch_Count  ==1) // count number of fsw irq cycle
            PWM_RAMP1_FORCE_T2__RAMP1_FORCE_T2__SET(20); // 20 x 5 = 100ns

        if (user_data.RAMP1_Switch_Count  ==5) // count number of fsw irq cycle
    8cycle ++
        {
            PWM_RAMP1_FORCE_T2__RAMP1_FORCE_T2__SET(60); // 60 x5 = 300
        }
```

Code Listing 17 softstart_srpwm.c

```
        else if(user_data.RAMP1_Switch_Count  >=12) // 4 cycle // if 60==60
        {
            user_data.RAMP1_Switch_Count = 0;
            sr_pwm_setting(2);
            Regulation_setup_fsw_irq(PMBUS_PAGE_0, fsw_irq_idx_2,
    fsw_irq_rate_sel_disabled); // enable fsw irq and set the rate
        }
    }
```

7.1.2 Add header file initialization

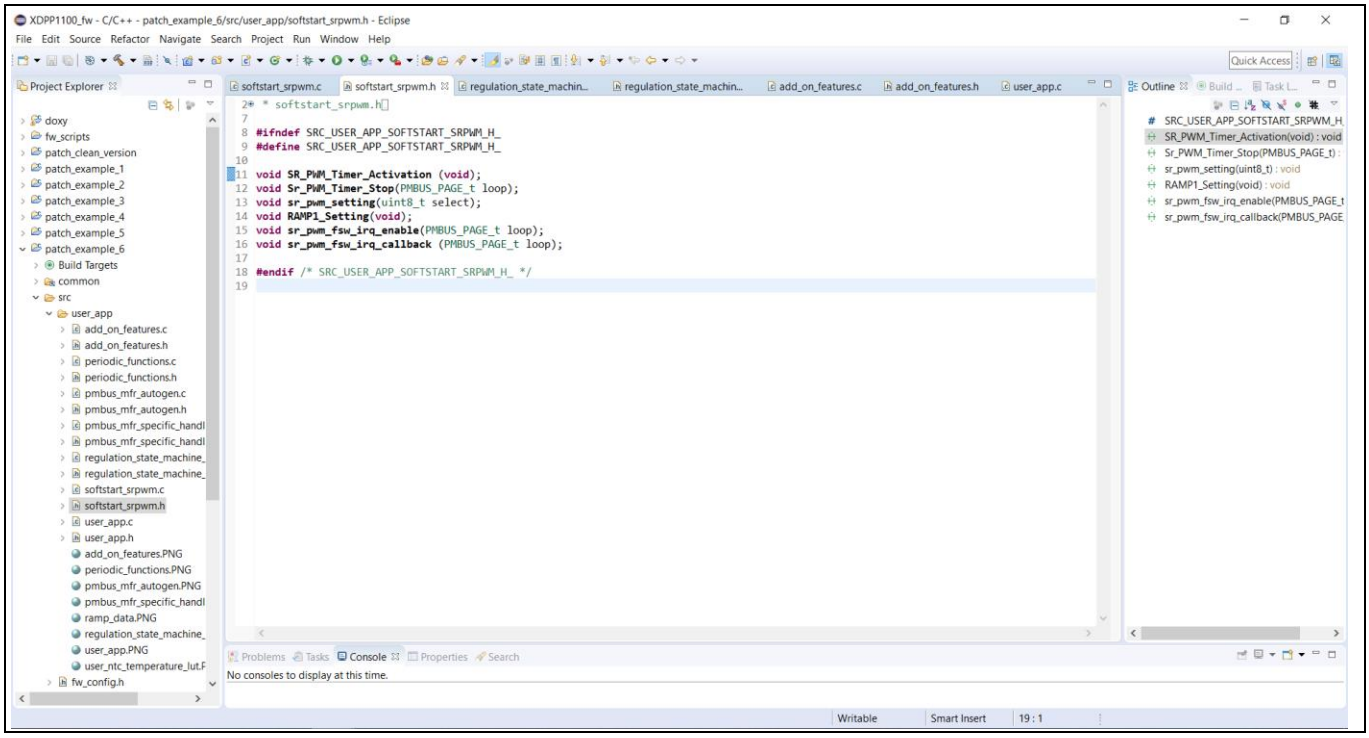


Figure 77 softstart_srpwm.h

Code Listing 18 softstart_srpwm.h

```
    void SR_PWM_Timer_Activation (void);
    void Sr_PWM_Timer_Stop(PMBUS_PAGE_t loop);
    void sr_pwm_setting(uint8_t select);
    void RAMP1_Setting(void);
    void sr_pwm_fsw_irq_enable(PMBUS_PAGE_t loop);
    void sr_pwm_fsw_irq_callback (PMBUS_PAGE_t loop);
```

7.1.3 User data variable and regulation callback function

Declare variable use in the user data structure and function use in regulation_state_machine_callback.c.

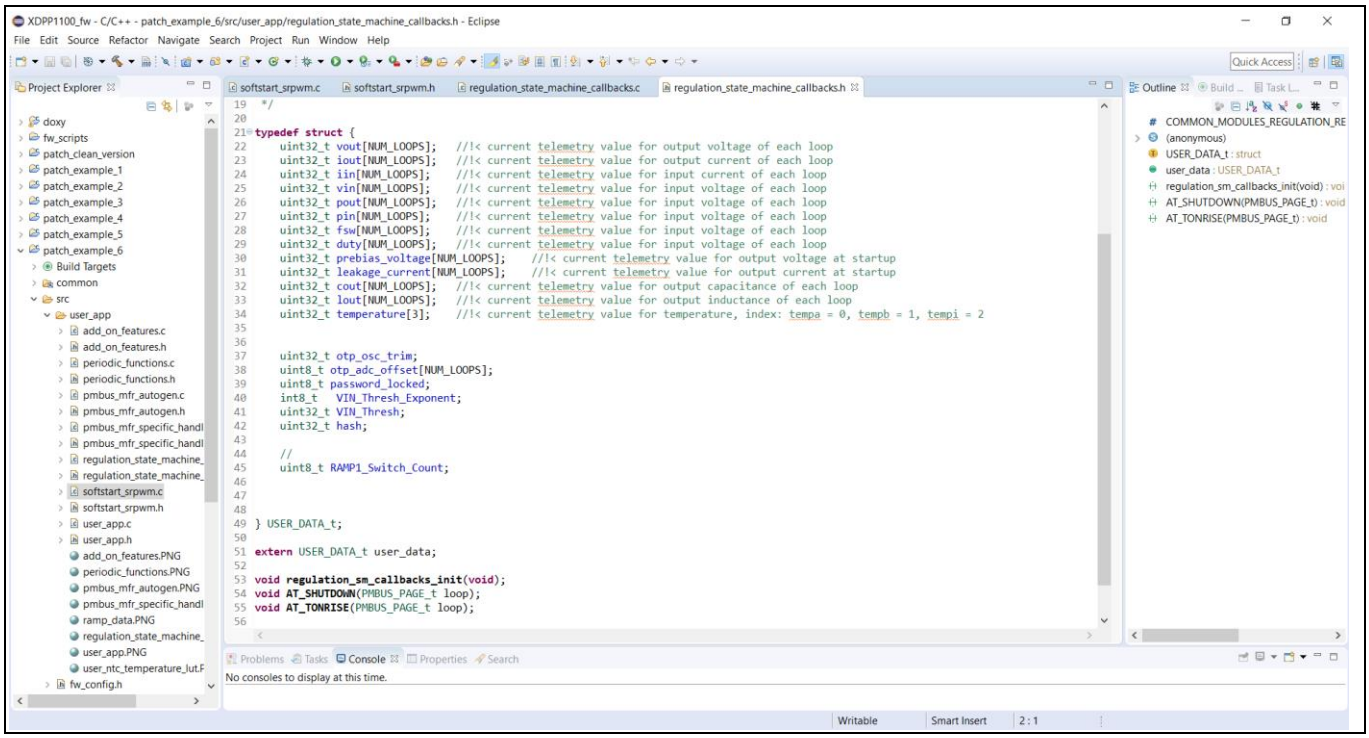


Figure 78 regulation_state_machine_callback.h

Code Listing 19 regulation_state_machine_callback.h

```
typedef struct {
        uint32_t vout[NUM_LOOPS];    //!< current telemetry value for output
voltage of each loop
        uint32_t iout[NUM_LOOPS];    //!< current telemetry value for output
current of each loop
        uint32_t iin[NUM_LOOPS];     //!< current telemetry value for input
current of each loop
        uint32_t vin[NUM_LOOPS];     //!< current telemetry value for input
voltage of each loop
        uint32_t pout[NUM_LOOPS];    //!< current telemetry value for input
voltage of each loop
        uint32_t pin[NUM_LOOPS];     //!< current telemetry value for input
voltage of each loop
        uint32_t fsw[NUM_LOOPS];     //!< current telemetry value for input
voltage of each loop
        uint32_t duty[NUM_LOOPS];    //!< current telemetry value for input
voltage of each loop
        uint32_t prebias_voltage[NUM_LOOPS];       //!< current telemetry
value for output voltage at startup
        uint32_t leakage_current[NUM_LOOPS];       //!< current telemetry
value for output current at startup
        uint32_t cout[NUM_LOOPS];    //!< current telemetry value for output
capacitance of each loop
        uint32_t lout[NUM_LOOPS];    //!< current telemetry value for output
inductance of each loop
        uint32_t temperature[3];     //!< current telemetry value for
temperature, index: tempa = 0, tempb = 1, tempi = 2


        uint32_t otp_osc_trim;
        uint8_t otp_adc_offset[NUM_LOOPS];
        uint8_t password_locked;
```

Code Listing 19 regulation_state_machine_callback.h

```
            int8_t   VIN_Thresh_Exponent;
            uint32_t VIN_Thresh;
            uint32_t hash;

            //
            uint8_t RAMP1_Switch_Count;



    } USER_DATA_t;

    extern USER_DATA_t user_data;

    void regulation_sm_callbacks_init(void);
    void AT_SHUTDOWN(PMBUS_PAGE_t loop);
    void AT_TONRISE(PMBUS_PAGE_t loop);
```

7.1.4 Regulation callback function

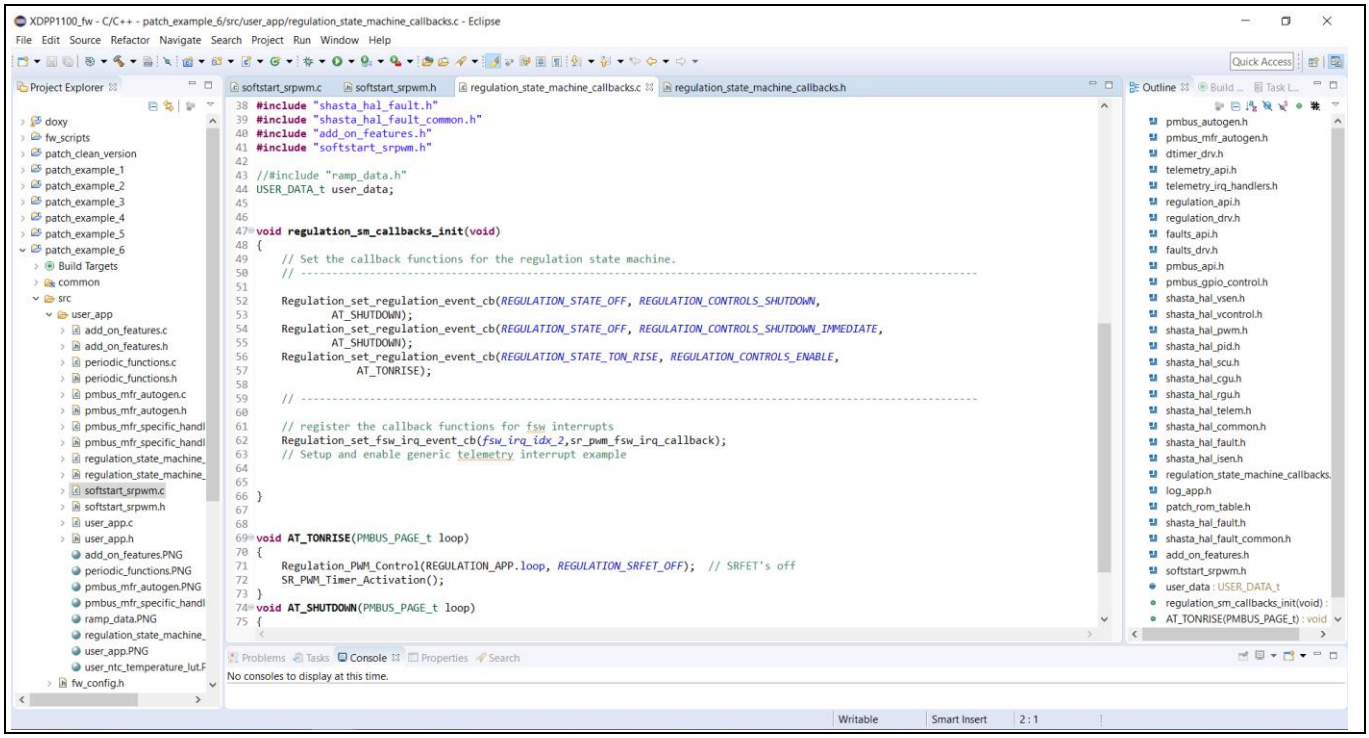


Figure 79 regulation_state_machine.c

Code Listing 20 regulation_state_machine.c

```
    #include "softstart_srpwm.h"

    //#include "ramp_data.h"
    USER_DATA_t user_data;


    void regulation_sm_callbacks_init(void)
    {
        // Set the callback functions for the regulation state machine.
```

Code Listing 20 regulation_state_machine.c

```
    // ------------------------------------------------------------------------
------------------------------------

    Regulation_set_regulation_event_cb(REGULATION_STATE_OFF,
REGULATION_CONTROLS_SHUTDOWN,
                    AT_SHUTDOWN);
    Regulation_set_regulation_event_cb(REGULATION_STATE_OFF,
REGULATION_CONTROLS_SHUTDOWN_IMMEDIATE,
                    AT_SHUTDOWN);
    Regulation_set_regulation_event_cb(REGULATION_STATE_TON_RISE,
REGULATION_CONTROLS_ENABLE, AT_TONRISE);

    // ------------------------------------------------------------------------
------------------------------------

    // register the callback functions for fsw interrupts
    Regulation_set_fsw_irq_event_cb(fsw_irq_idx_2,sr_pwm_fsw_irq_callback);
    // Setup and enable generic telemetry interrupt example


}


void AT_TONRISE(PMBUS_PAGE_t loop)
{
    Regulation_PWM_Control(REGULATION_APP.loop, REGULATION_SRFET_OFF);  //
SRFET's off
    SR_PWM_Timer_Activation();
}
void AT_SHUTDOWN(PMBUS_PAGE_t loop)
{
    VCONTROL_VC_VRAMP3__VC_VCONTROL_DELTA__SET((vcontrol_idx_t) loop, 0);
}
```

7.1.5 Add initialize function

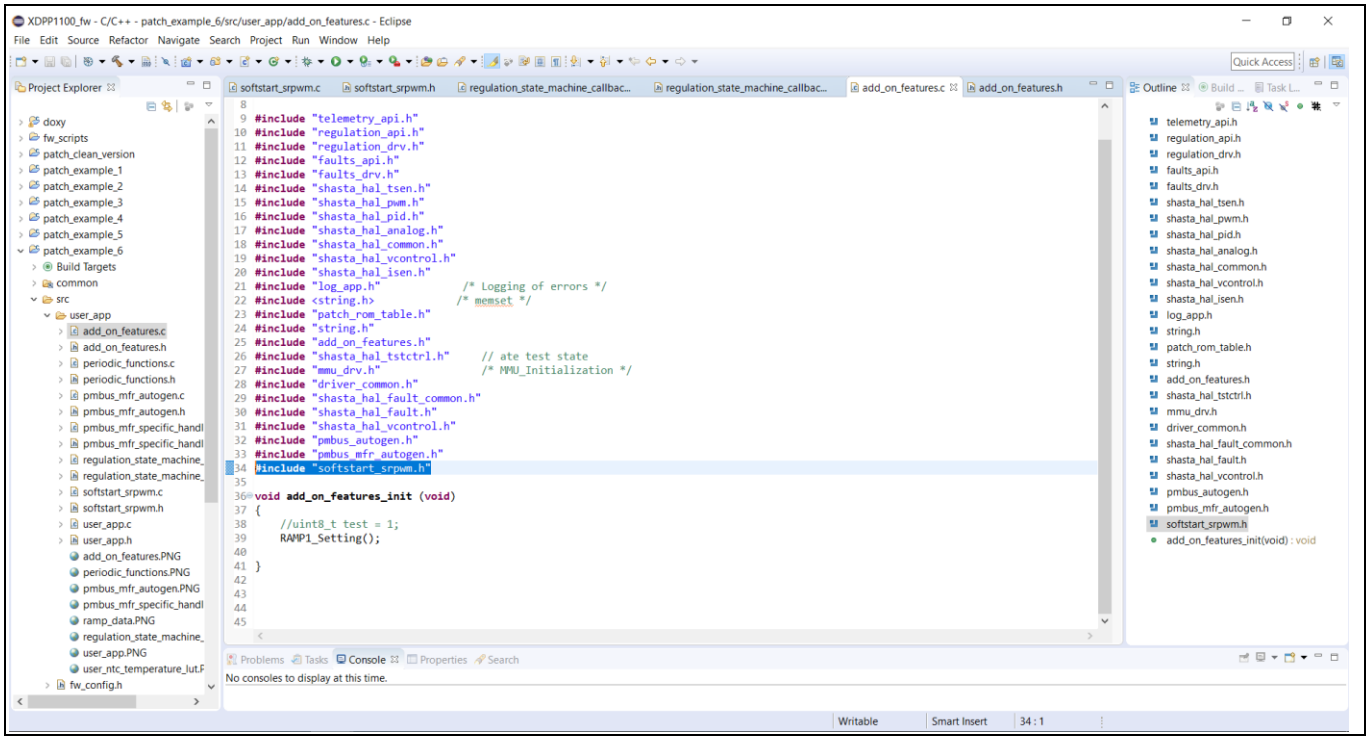


Figure 80 add_on_features.c

Code Listing 21 add_on_features.c

```
#include "softstart_srpwm.h"
void add_on_features_init (void)
{
    //uint8_t test = 1;
    RAMP1_Setting();

}
```

7.1.6 User app

Open the the user_app.c and insert source code according to **Code Listing 2** in void user_drv_init(void) to run the FW patch in RAM.

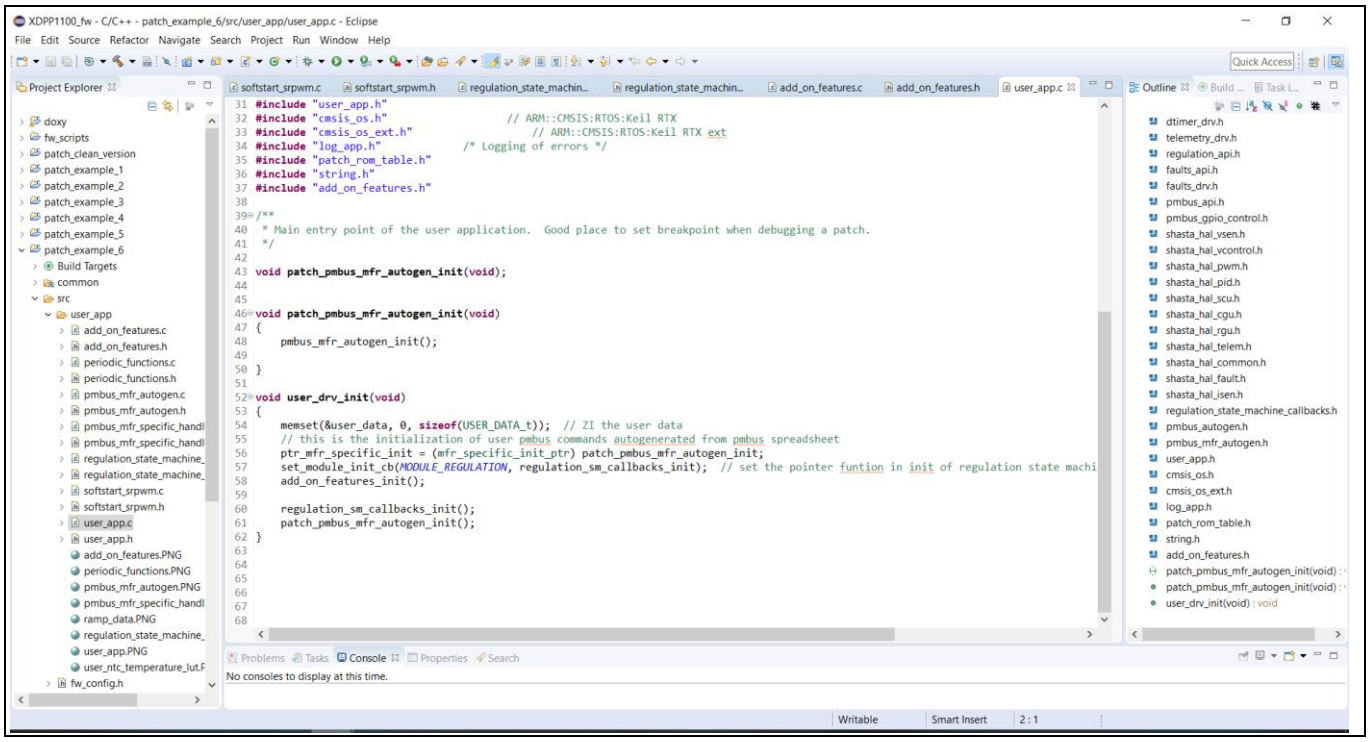


Figure 81 user_app.c

Code Listing 22 user_app.c

```
void user_drv_init(void)
{
  memset(&user_data, 0, sizeof(USER_DATA_t));  // ZI the user data
  // 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();

  // Added lines to run patch in RAM
   regulation_sm_callbacks_init();
   patch_pmbus_mfr_autogen_init();
}
```

7.2 Compile and program to XDPP1100 RAM/OTP

To compile and program the FW patch refer to **Section 3.3** Compile and program to XDPP1100 RAM/OTP.

7.2.1 Running example 6 patch

Click “EN H/EN L” to start the PWM regulating. Use an oscilloscope to confirm initial kickstart pulse upon enable pin go high.

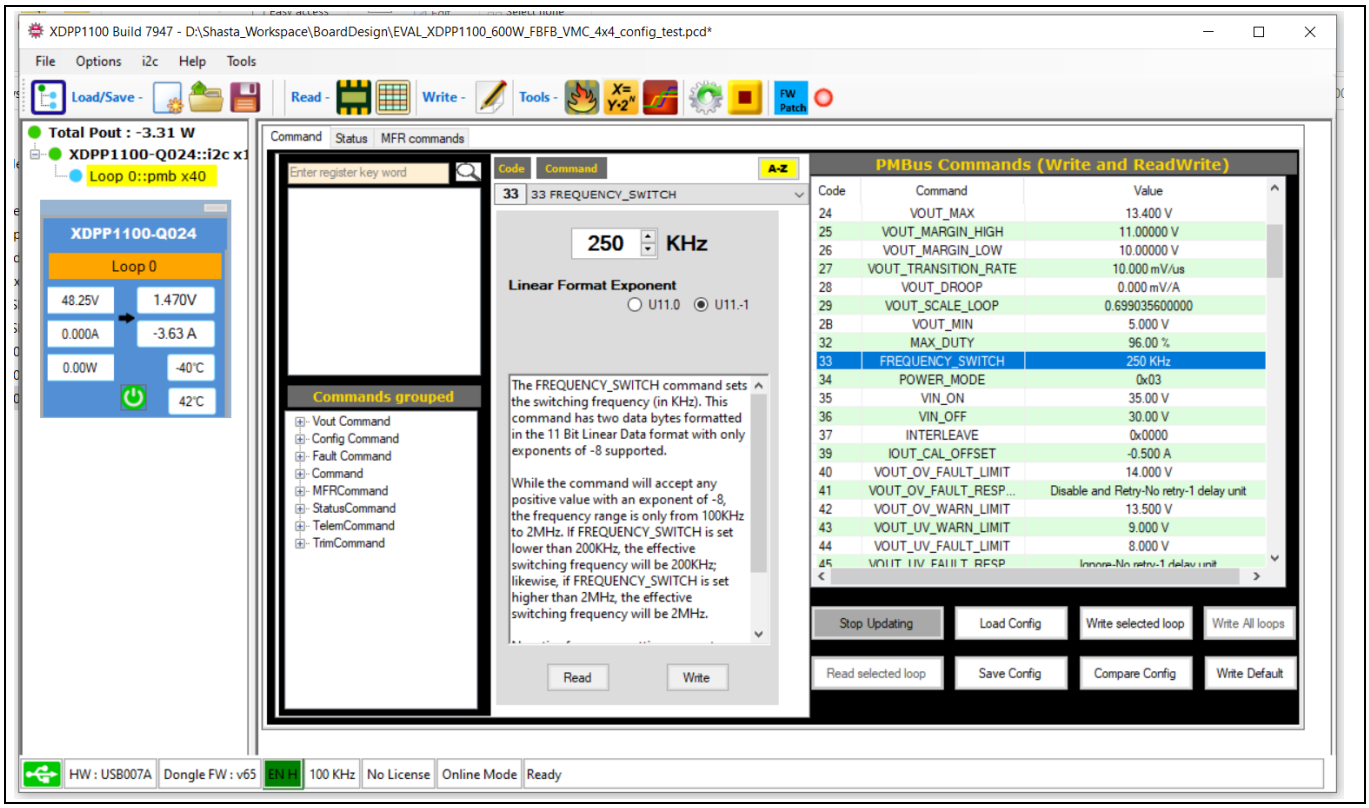


Figure 82 XDPP1100 GUI

References

[1] Infineon Technologies AG: *XDP™ digital power XDPP1100 Product Page*; **Available online**

[2] Infineon Technologies AG: *XDP™ digital power XDPP1100 firmware user guide*; **Available online**

[3] Infineon Technologies AG: *XDPP1100 firmware documentation (from XDPP11000 GUI)*

[4] Infineon Technologies AG: *XDP™ digital power controller XDPP1100 number format and conversion*; **Available online**

[5] Infineon Technologies AG: *XDP™ digital power controllers XDPP1100 programming instruction*; **Available online**

[6] Infineon Technologies AG: *XDP™ digital power XDPP1100 technical reference manual*; **Available online**

[7] Infineon Technologies AG: *XDP™ digital power controllers XDPP1100 PMBus commands*; **Available online**

Revision history

Document version	Date of release	Description of changes
V 1.0	2023-03-28	Initial release

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

Edition 2023-03-28

Published by

Infineon Technologies AG

81726 Munich, Germany

© 2023 Infineon Technologies AG. All Rights Reserved.

Do you have a question about this document?

Email: erratum@infineon.com

Document reference

UM_2302_PL88_2302_163121

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.