XDPP1100 production file format

About this document

Scope and purpose

The Infineon XDPP1100 graphical user interface (GUI) has the function of generating a production file .xcf format, which contains all register data, PMBus command data and firmware (FW) to be stored to one-time programmable memory (OTP). This document describes the format of the production file .xcf.

Intended audience

XDPP1100 users.

Table of contents

About this document **1**
Table of contents **2**
1 Production file format details **3**
1.1 Register configuration 3
1.2 PMBus configuration 5
1.3 FW configuration 7
1.4 FW patched 7
2 Production file creation **9**
2.1 Add device 9
2.1.1 Auto-populate device 9
2.1.2 Add a new device manually 9
2.2 Multi Device Programmer tool 10
2.3 Check OTP partition 10
2.4 Store FW patch 11
2.5 Load design file 12
2.6 Generate production file 13
3 Production file programming **15**
3.1 Storing register configuration to OTP 15
3.2 Storing PMBus configuration to OTP 16
References **19**
Revision history **20**
Disclaimer **21**

1 Production file format details

The production file (.xcf) generated from XDPP1100 GUI includes all the register values and PMBus commands for storing a full configuration via volatile registers to OTP.

Figure 1 shows a snapshot of the production file .xcf generated by the XDPP1100 GUI.



Figure 1 Portion of production file

The production file (.xcf) is categorized into six segments of data: register configuration, loop 0 PMBus configuration, loop 1 PMBus configuration, loop 0 patched PMBus configuration, loop 1 patched PMBus configuration, FW configuration and FW patch data. All numbers in the .xcf file are hex values.

1.1 Register configuration

Figure 2 shows the register configuration segement of the file. The line “Register Configuration: CRC32:” value calculates a checksum of the whole register configuration segment.

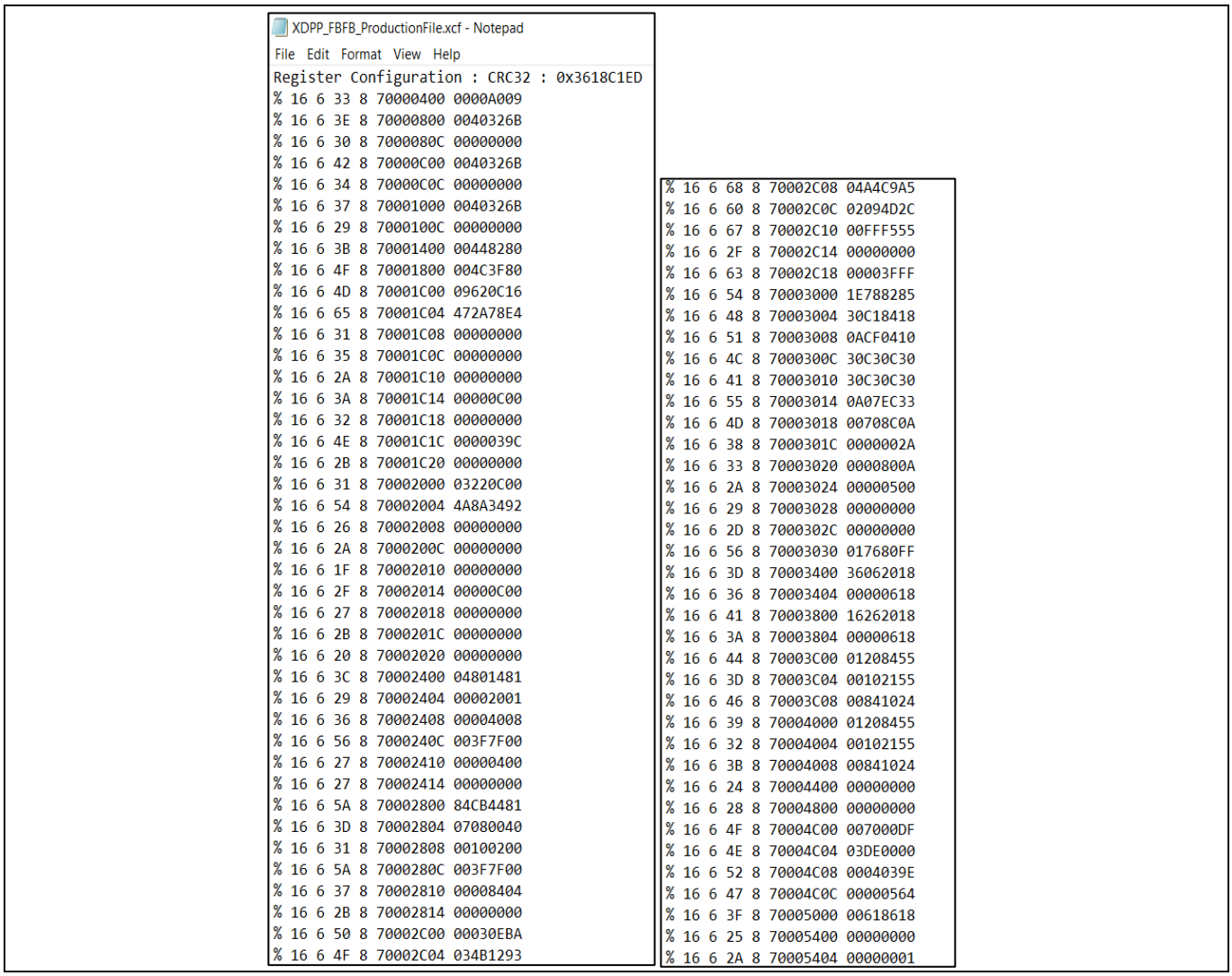


Figure 2 Register configuration of production file

The CRC32 calculation is from the start of the first line of the register configuration to the last register configuration. **Code Listing 1** and **Code Listing 2** show the part of the program code that initializes the CRC32 table and calculates the CRC32 base on each line.

Code Listing 1 CRC32 table initialize

```
Private CRC32Table(255) As UInteger
  Private Sub InitializeCRC32Table()
    For i As UInteger = 0 To CRC32Table.Length - 1
      Dim te As UInteger = i ' table entry
      For j As Integer = 0 To 7
        If (te And 1) = 1 Then te = (te >> 1) Xor &HEDB88320UI Else te >>= 1
      Next
      CRC32Table(i) = te
    Next
  End Sub
```

Code Listing 2 CRC32 calculation for register

```
Public Function CalculateCRC32_mod(ByVal line() As String, ByVal linestart As Integer, ByVal lineend As Integer)
As String
    Dim i, j As Integer
    Dim chrByte As Byte
    InitializeCRC32Table()
    Static crc As UInteger
    crc = UInteger.MaxValue

    For i = linestart To lineend - 1
      For j = line(i).Length - 8 To line(i).Length - 1
        chrByte = CByte(Asc(line(i).Substring(j, 1)))
        crc = (crc >> 8) Xor CRC32Table((crc And &HFF) Xor chrByte)
      Next
    Next
    Return Hex(Not crc).PadLeft(8, "0")
  End Function
```

Table 1 explains the meaning for each line of data.

For example, for the line of data “% 16 6 33 8 70000400 0000A009”, the description is below.

Table 1 Register configuration

Valid data	Number of characters (hex.)	Type (hex.)	CRC (hex.)	Data length	Register address (hex.)	Data (hex.)
%	16	6	33	8	70000400	000A009

The CRC calculation for **Table 1** is the total of all integer, non-zero numbers.

Example CRC 33 (hex.) = 51 (dec.):

CRC = 1+6+6+8+7+4+10+9 = 51

In the “Data” column of **Table 1** the value “A” in hexadecimal is converted to value “10” in decimal.

1.2 PMBus configuration

The segment for loop 0 PMBus configuration, loop 1 PMBus configuration, loop 0 patched PMBus configuration and loop 1 patched PMBus configuration have the same format and are similar to the register configuration.

Figure 3 shows the loop 0 PMBus configuration segment of the file.

```
Loop 0 PMBus Configuration :  CRC32 : 0xA9B5A51F
% 0B 6 013 2 00 00
% 0B 6 01C 2 01 80
% 0B 6 025 2 02 1F
% 0B 6 014 2 10 00
% 1A 6 01E 10 1B 0000000000000000
% 0B 6 01D 2 20 17
% 0D 6 024 4 21 6400
% 0D 6 01B 4 22 0000
% 0D 6 01C 4 23 0000
% 0D 6 031 4 24 6E00
% 0D 6 01E 4 25 0000
% 0D 6 01F 4 26 0000
% 0D 6 037 4 27 E810
% 0D 6 021 4 28 0000
% 0D 6 03B 4 29 065E
% 0D 6 02E 4 2B 0A00
% 0D 6 03F 4 32 F17C
% 0D 6 02F 4 33 0864
% 0B 6 01D 2 34 03
% 0D 6 03B 4 35 F094
% 0D 6 037 4 36 F080
% 0D 6 026 4 37 0140
% 0D 6 039 4 39 E800
% 0D 6 02A 4 40 7800
% 0B 6 032 2 41 BF
% 0D 6 031 4 42 6E00
% 0D 6 020 4 43 2000
% 0D 6 02C 4 44 1C00
% 0B 6 036 2 45 BF
% 0D 6 02A 4 46 0018
% 0B 6 03C 2 47 FF
% 0D 6 02F 4 48 0C00
% 0D 6 02C 4 4A 0016
% 0D 6 035 4 4B 0780
% 0B 6 041 2 4C FF
% 0D 6 03E 4 4F 007D
% 0B 6 032 2 50 BF
% 0D 6 02C 4 51 005A
% 0D 6 049 4 52 0FEE
% 0D 6 048 4 53 0FEC
% 0B 6 036 2 54 BF
% 0D 6 034 4 55 F130
% 0B 6 038 2 56 BF
% 0D 6 03D 4 57 F128
% 0D 6 047 4 58 F08C
% 0D 6 03C 4 59 F080
% 0B 6 03C 2 5A BF
% 0D 6 044 4 5B F0E0
% 0B 6 03E 2 5C BF
% 0D 6 04C 4 5D F0C8
% 0D 6 030 4 5E 6000
% 0D 6 03C 4 5F 5C00
% 0D 6 03E 4 60 03FF
% 0D 6 035 4 61 F008
% 0D 6 04C 4 62 F3FC
% 0B 6 01C 2 63 00
% 0D 6 021 4 64 0000
% 0D 6 045 4 65 F0C8
% 0D 6 050 4 66 F0FF
% 0D 6 049 4 6A 13FF
% 0D 6 04A 4 6B 13FF
% 22 6 025 18 99 000000000000000000000000
% 32 6 028 28 9A 00000000000000000000000000000000000000000000
% 22 6 027 18 9B 000000000000000000000000
% 22 6 028 18 9C 000000000000000000000000
% 22 6 029 18 9D 000000000000000000000000
% 32 6 02C 28 9E 00000000000000000000000000000000000000000000
% 0D 6 021 4 A0 0000
% 0D 6 022 4 A1 0000
% 0D 6 023 4 A2 0000
% 0D 6 024 4 A3 0000
% 0D 6 025 4 A4 0000
% 0D 6 026 4 A5 0000
% 0D 6 027 4 A6 0000
% 0D 6 028 4 A7 0000
% 0D 6 029 4 A8 0000
% 2A 6 01F 20 B0 00000000000000000000000000000000
% 11 6 04D 8 C4 0F3C00C3
% 26 6 0AB 1C C5 F800F800C800F800F8000000C300
% 34 6 051 2A C6 000000000000000000CDD0000000000000000000000000
% 3C 6 03A 32 C8 0000000000000000000000000000C0000000000000000000000000
% 20 6 03D 16 C9 0000000088000240011001
% 0B 6 03B 2 CA 3F
% 0D 6 02E 4 CB 0000
% 0D 6 049 4 CD B00E
% 0D 6 05D 4 CE B3FF
% 3A 6 13D 30 CF 9C289C28703870387038703870389C289C28703870387038703870387038
% 0D 6 034 4 D1 001E
% 3A 6 029 30 D6 00000000000000000000000000000000000000000000000000000000
% 0B 6 02C 2 DC 00
% 0D 6 029 4 E4 0000
% 0D 6 04E 4 EA B857
% 0B 6 034 2 EB 08
% 15 6 075 C F1 0000D51ED99D
% 2A 6 027 20 F4 00000000000000000000000000000000
% 2A 6 028 20 F5 00000000000000000000000000000000
```

Figure 3 Loop 0 PMBus configuration of production file

The line “Loop 0 Configuration: CRC32:” value calculates a checksum of the whole loop 0 PMBus configuration segment. The CRC32 has the same calculation format, as described in **Code Listing 1** and **Code Listing 2**.

Table 2 explains the meaning for each line of data.

For example, for the line of data “% 22 6 025 18 99 000000000000000000000000”, the description is below.

Table 2 PMBus configuration

Valid data	Number of characters (hex.)	Type	CRC (hex.)	Data length	Command code (hex.)	Data (hex.)
%	22	6	025	18	99	000000000000000000000000

The CRC calculation for **Table 2** is the total of all integer, non-zero numbers.

Example CRC 025 (hex.) = 37 (dec.):

CRC = 2+2+6+1+8+9+9 = 37

1.3 FW configuration

Figure 4 contains information to configure the OTP partition, which is the FW trim and the FW configuration.

```
FW Configuration :
Version_16_Size_16: 2005c500 00010008
i2coffset_8_Pmbusoffset_8_Configoffset_8_ConfigXvEnabled_8: 2005c504 00000001
TrimVersion_16_Trimsize_16: 2005c508 0002002A
fwOTPbootdescriptor: 2005c50C 00000003
OTPSection1_16_Section0_16: 2005c510 C0004000
OTPSection3_16_Section2_16: 2005c514 00000000
OTPSection5_16_Section4_16: 2005c518 00000000
OTPSection7_16_Section6_16: 2005c51C 00000000
OTPSection9_16_Section8_16: 2005c520 00000000
OTPSection11_16_Section10_16: 2005c524 00000000
OTPSection13_16_Section12_16: 2005c528 00000000
OTPSection15_16_Section14_16: 2005c52C 00000000
OTPSection16: 2005c530 0
```

Figure 4 FW configuration of production file

Table 3 FW configuration

Type of data	Address	Data
OTPSection1_16_Section0_16	2005c510	C0004000

1.4 FW patched

Figure 5 shows a portion of the data from the FW patch. The first line “Partition number:” indicates the FW patch to be stored, and the partition of the OTP. The data after “Partition number : 1” is the start of the data to be stored to the OTP section for the patch. The data before the line “[End]” is the last data to be stored to the OTP section for the patch.

Production file format details

```
Partition number : 1
00000010 00000000
1C001C00 00000000
E4EF4267 00000000
10024000 00000000
1002401D 00000000
5DD2AD32 00000000
639B36AA 00000000
480CB510 00000000
1A124A0C 00000000
F0012100 00000000
480BF923 00000000
1A124A0B 00000000
F001490B 00000000
480BF8DB 00000000
1A124A0B 00000000
F001490B 00000000
4A0BF8D5 00000000
601A4B0B 00000000
FF84F000 00000000
46C0BD10 00000000
20063800 00000000
20063A38 00000000
20063800 00000000
20063800 00000000
100258DC 00000000
20063A38 00000000
20063A38 00000000
100258DC 00000000
62FE301B 00000000
40000020 00000000
4C07B510 00000000
691B6823 00000000
47982001 00000000
68DB6823 00000000
47982001 00000000
21062200 00000000
F7E12001 00000000
BD10FAF1 00000000
2005EA48 32C6CE4B
4C07B510 [End]
691B6823
```

Figure 5 FW patched production file

2 Production file creation

This chapter is extracted from the **XDPP1100 programming instruction**.

2.1 Add device

The XDPP1100 can be added in the GUI by auto-populating or manually adding the device.

2.1.1 Auto-populate device

Use the auto-populate function to detect the device that is active (with 3.3 V bias). Click the “Auto Populate” icon shown in the red block in **Figure 6**, and a device will be automatically added into the device window.

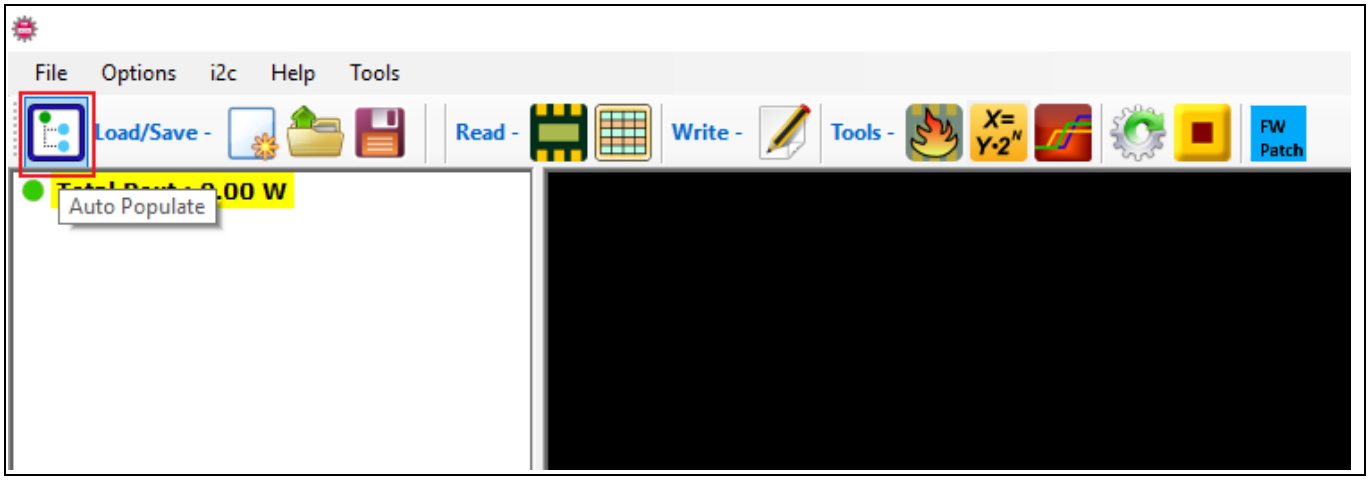


Figure 6 Auto-populate function

2.1.2 Add a new device manually

The user can also manually add a device to the design. Right-click the Device Manager, which is highlighted in yellow in the left window (**Figure 7**), choose “Add Device” and follow the arrow to the right to pick a device to use.

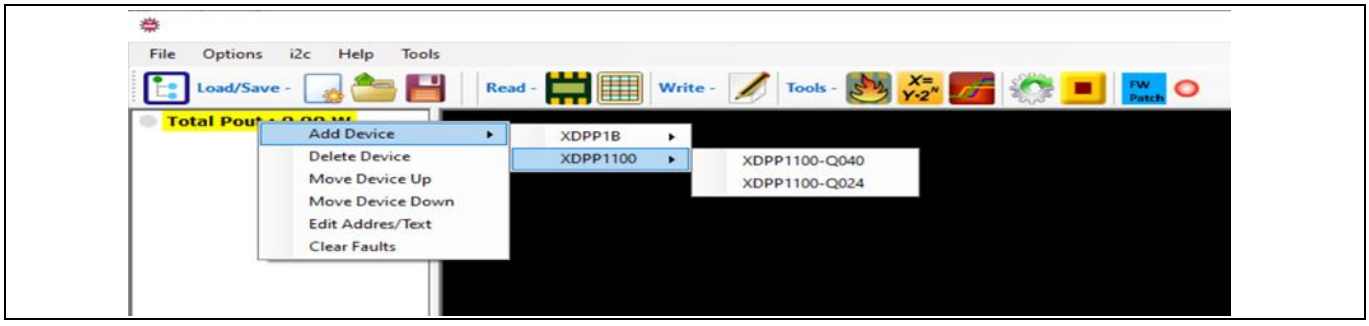


Figure 7 Adding a device manually

If the device is added manually, the default I²C address is 0x10, and the default PMBus address is 0x40, as shown in **Figure 8**. With correct I²C and PMBus addresses, the dot in front of the device will turn blue, indicating the device is ready for I²C communication. If the dot is gray that means the IC is not communicating through I²C; the address may be wrong.

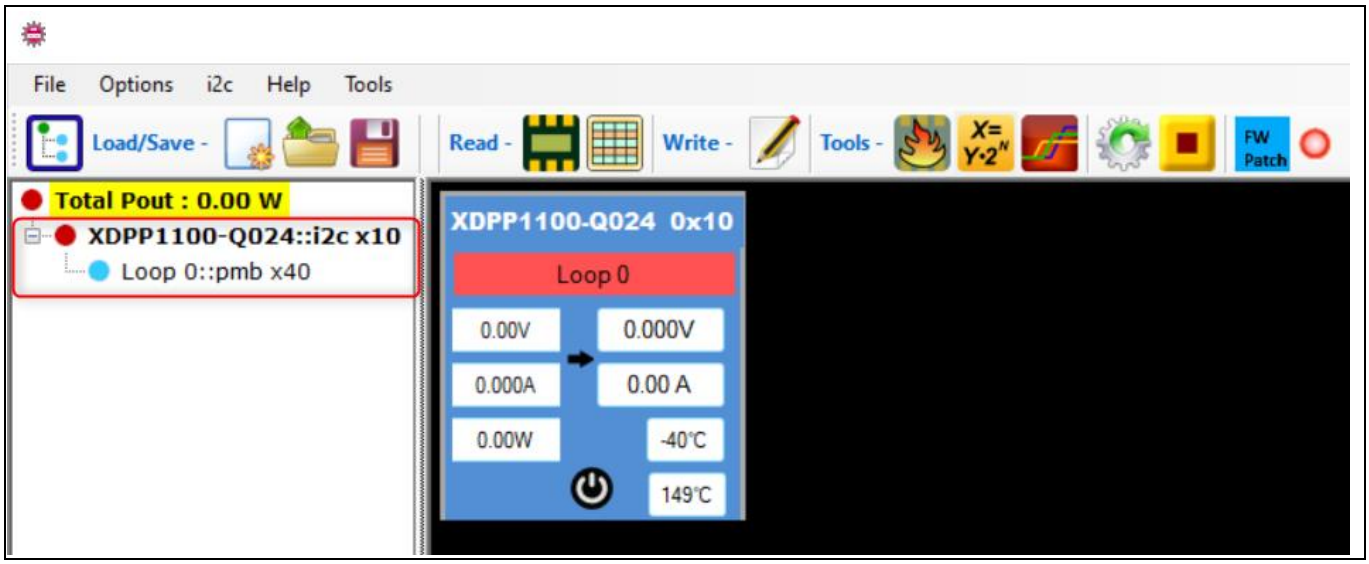


Figure 8 Default I²C and PMBus address

2.2 Multi Device Programmer tool

Use the GUI “Multi Device Programmer” tool to generate the production file .xcf format.

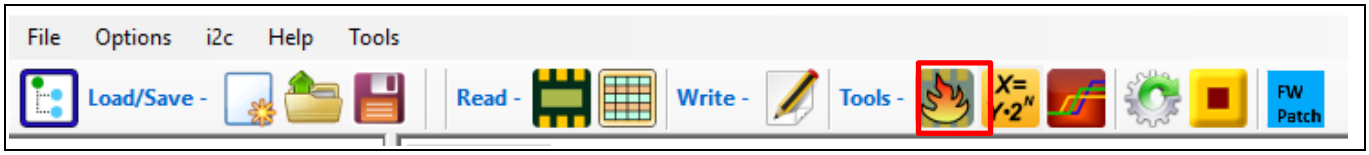


Figure 9 Multi Device Programmer tool

2.3 Check OTP partition

Before storing a patch or configuration, the user should check if the OTP has been properly partitioned. Click the “OTP Data” button (**Figure 10**), and a pop-up window shows all the OTP-related information (**Figure 11**). Verify the OTP partition and size in the “FW Trim” section by clicking the “Read FW Trim” button.

The data partition can be set up to 0x7C00 or 31 kB. The default data partition is 0x4000 or 16 kB. If a different OTP partition is desired, change the partition size and click the “Store Trim” button. The new partition will be saved in the OTP trim structure.

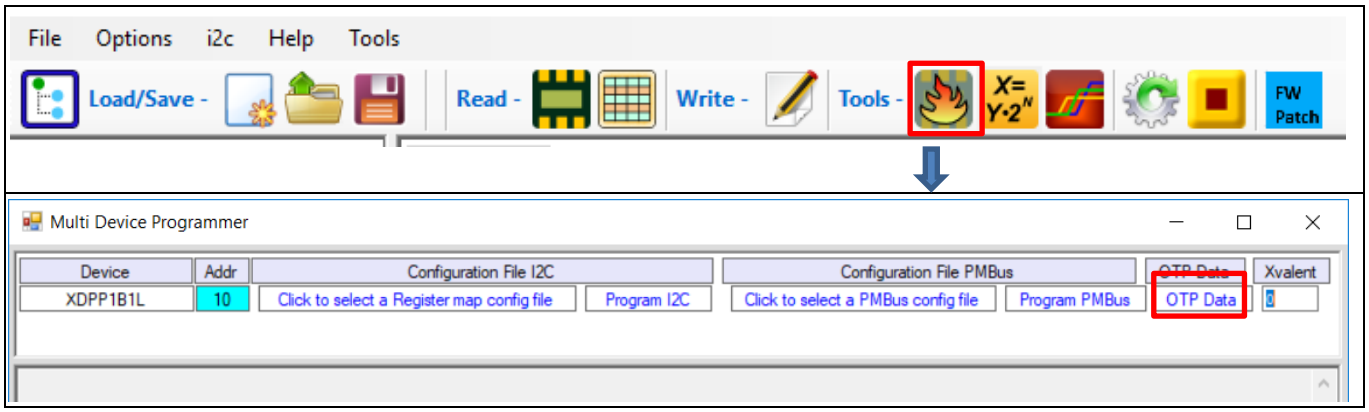


Figure 10 Multi Device Programmer tool – OTP data

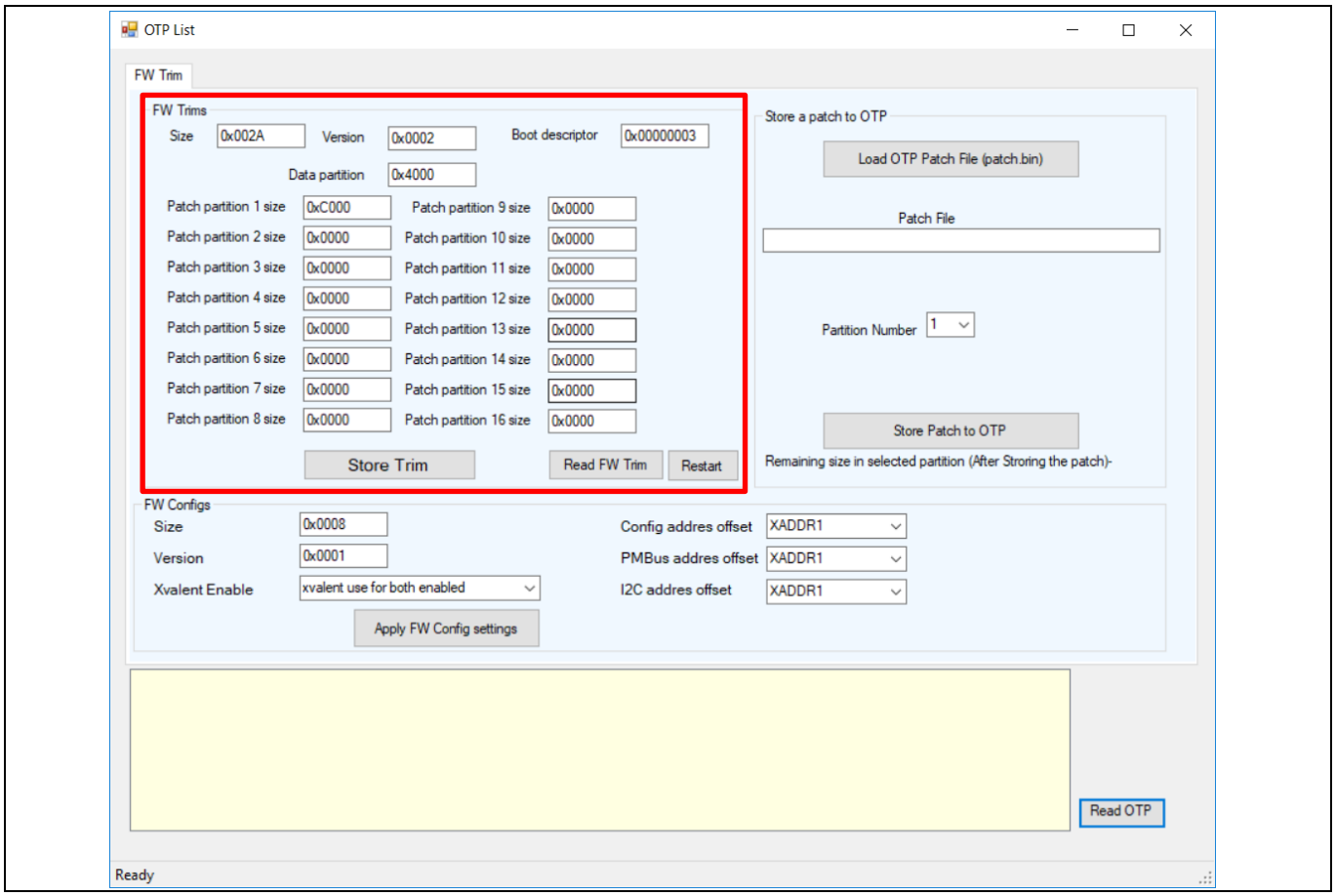


Figure 11 Multi Device Programmer tool – OTP data, FW trims

The number of FW patch rewrites depends on the size of the patch code. The old patch must be invalidated before storing a new patch to the same partition. The facility to invalidate the existing patch is provided in the GUI “FW patch” tool.

2.4 Store FW patch

If a FW patch applies, use the following steps to store the patch to OTP.

- Click on the button “Load OTP Patch File (patch.bin)” (**Figure 12**) to browse for the patch.bin file; select the file and click “OK”.

- Select the patch partition (from 1 to 15). By default, the patch partition is 1.
- Click on the button “Store Patch to OTP”.
- Optional steps:
  - To verify if a patch is stored in OTP or not, check the data present in the OTP area from the starting address to the size of the patch file. The last word of the patch file is data CRC.
  - The starting address of the patch partition is 0x1002400, if the data partition size is 16 kB = 0x4000.

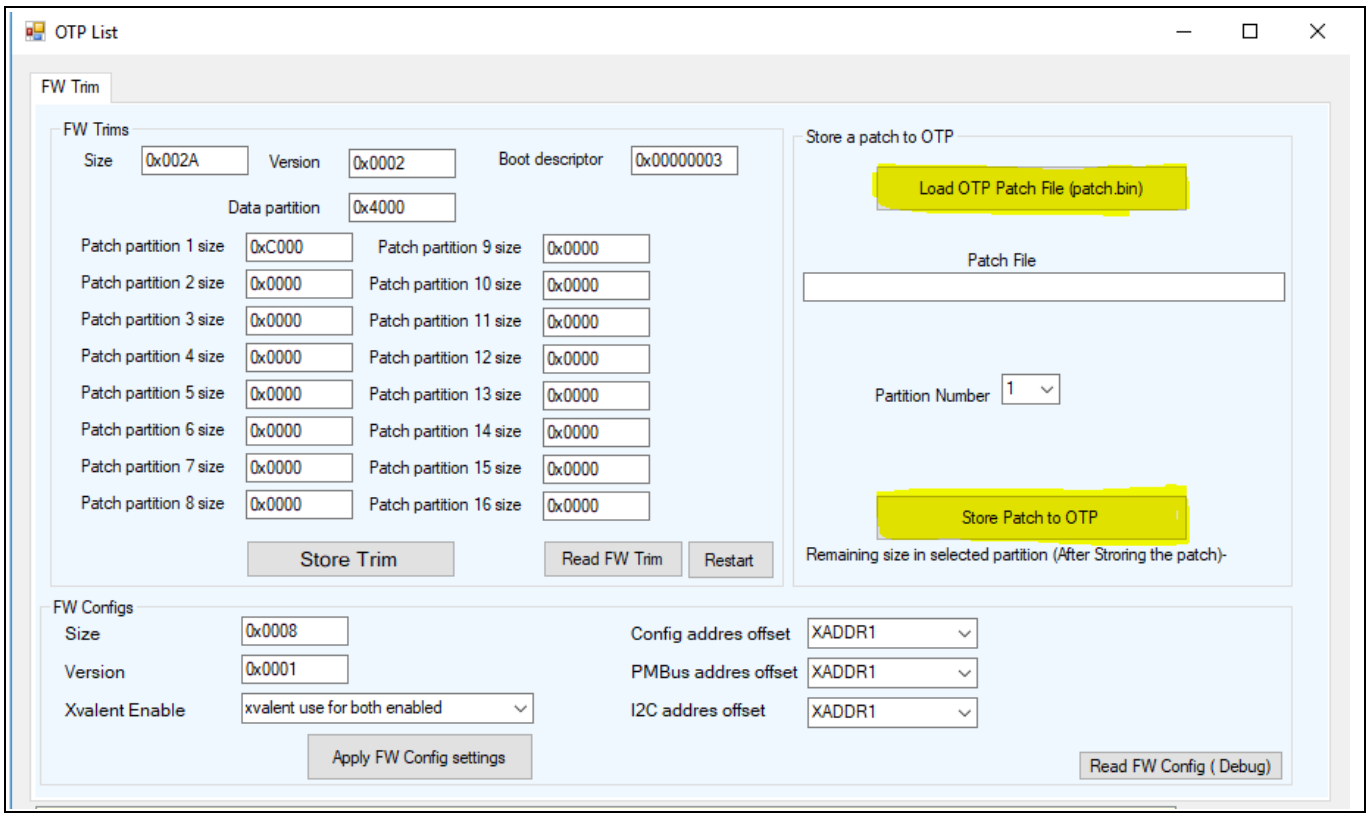


Figure 12 Multi Device Programmer tool – OTP data, store patch to OTP

Note: Size, version, and boot descriptor are constants – changing these values is not recommended.

2.5 Load design file

The design file should be loaded into RAM before storing in OTP. If this is not yet done, close the Multi Device Programmer and open the design file; click the “Write All (I^2C and PMBus)” shortcut button to write all registers and PMBus commands to RAM.

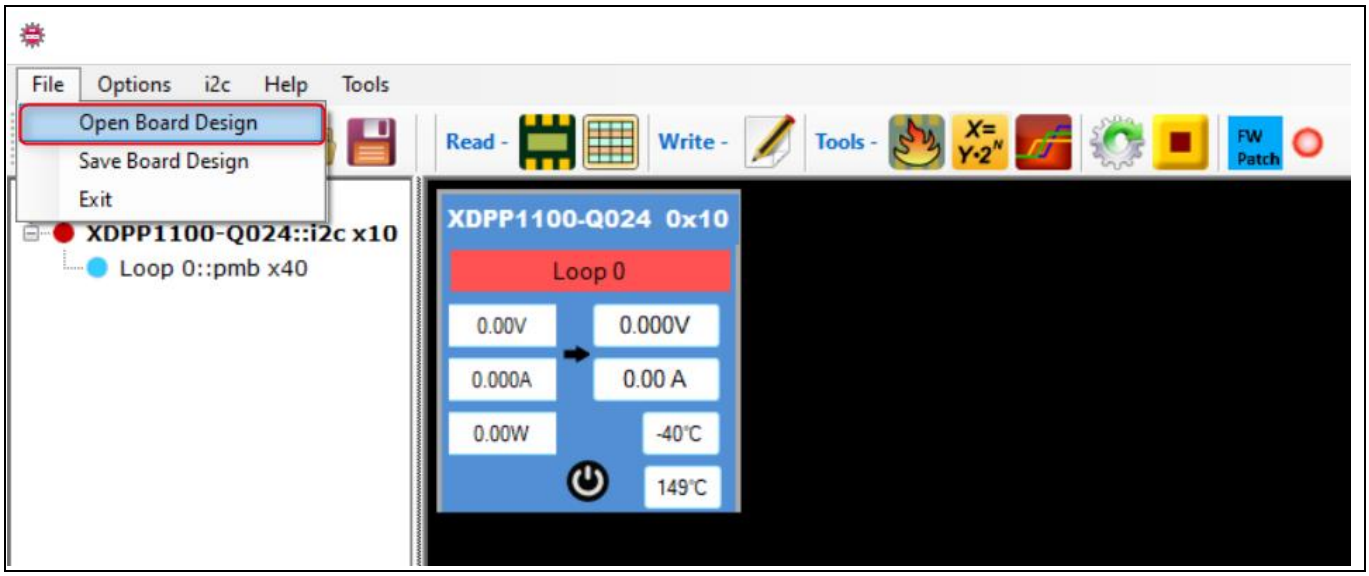
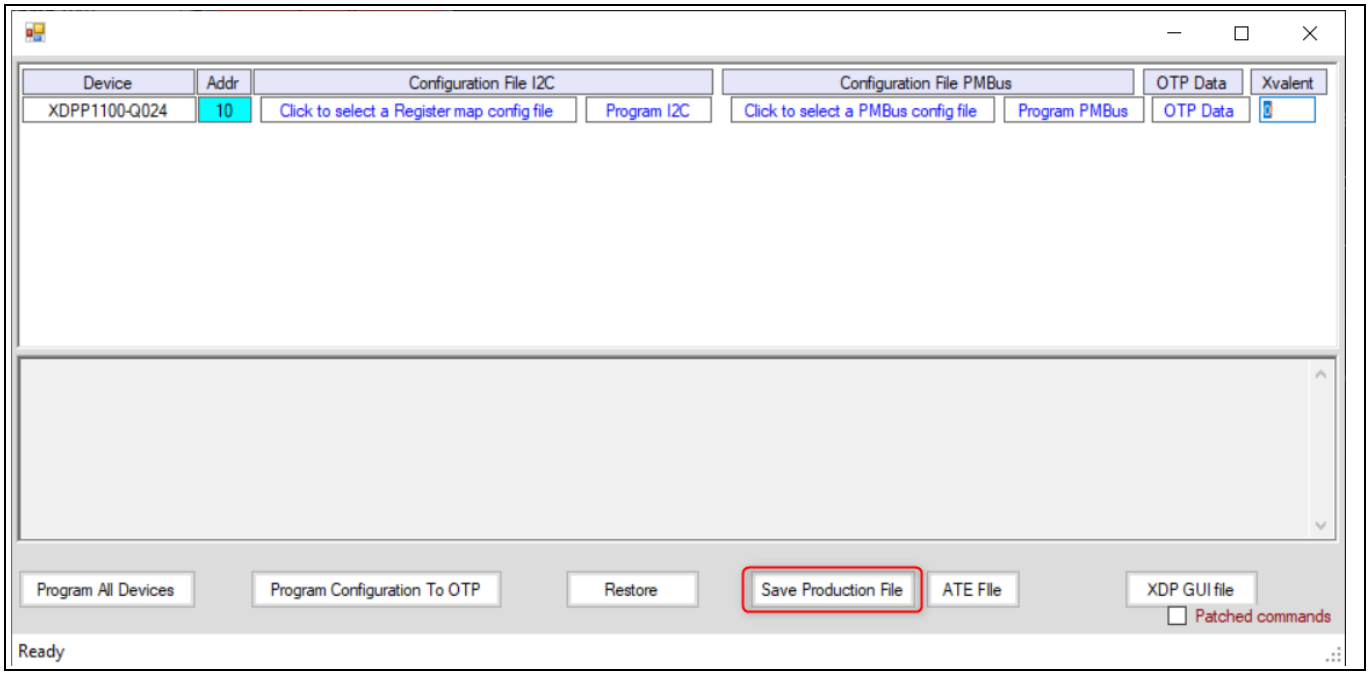


Figure 13 Open Board Design

2.6 Generate production file

Open the Multi Device Programmer, and click “Save Production File”.



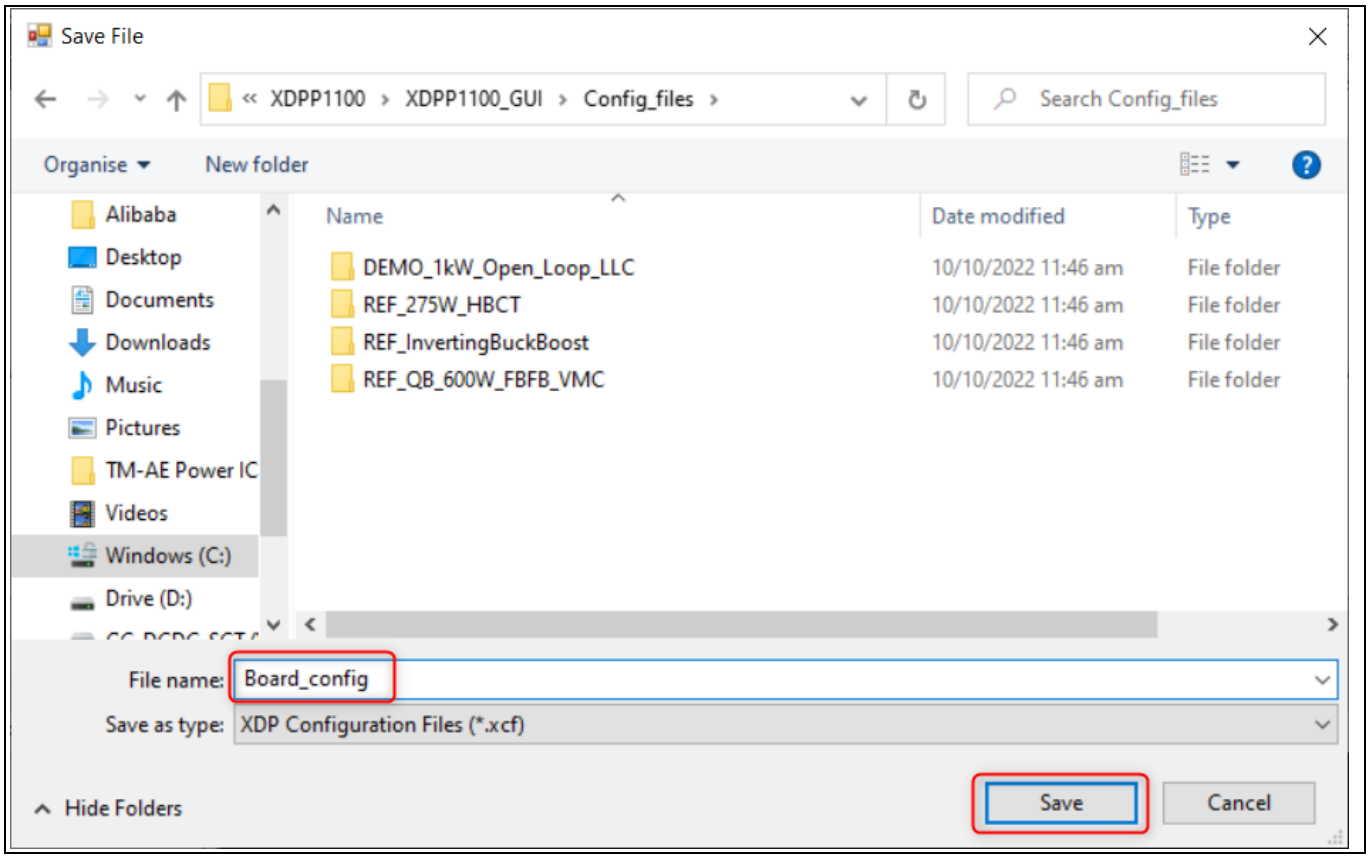


Figure 14 Open board design

3 Production file programming

The details of using production file .xcf format to program are found in the **XDPP1100 programming instruction**.

3.1 Storing register configuration to OTP

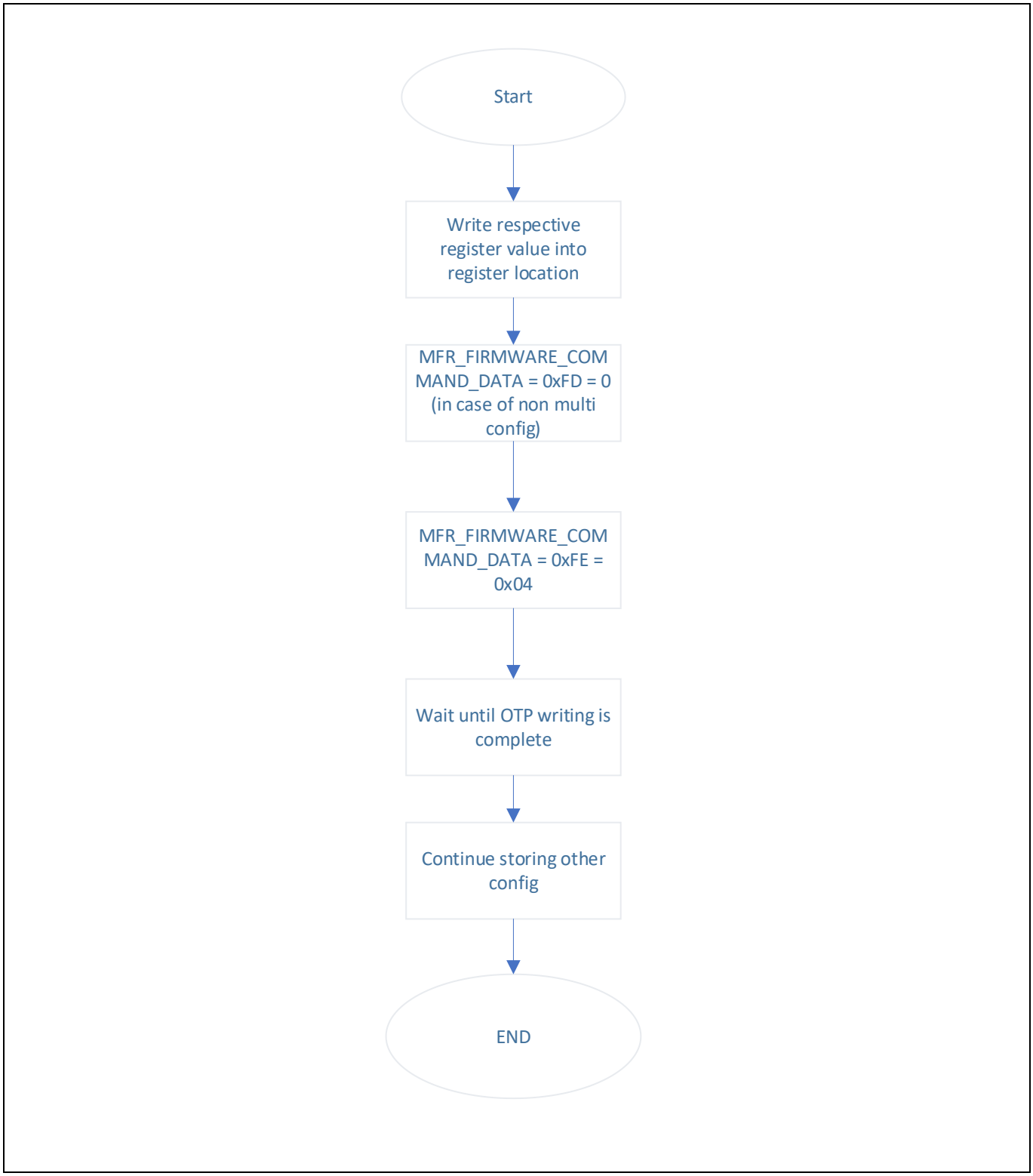


Figure 15 Programming Flow Chart

Code Listing 3

```
// 1. Write the register configuration to RAM
     // Write register values into the respective address by following the I2C protocol
     For register = start to end
          i2c Write 4 Bytes (register.address, register.data)
     Next
     // Follow the below steps to write the register configuration from RAM to OTP

// 2. Write MFR_FIRMWARE_COMMAND_DATA with the following data
     writeData(0) = 0  // Xvalent number (usually 0 if multi-config is not used)
     writeData(1) = 0
     writeData(2) = 0
     writeData(3) = 0
     PMBus Block Write (MFR_FIRMWARE_COMMAND_DATA, writeData)

// 3. Optional – Only needed if you want to change the settings
//     Write to register address fw_config_xv_enable (0x2005C504) the following data byes
     writeData(0) = 1  // XValent and address offsets enabled (0 – disabled, 1 – enabled)
     writeData(1) = 0  // Source of Config XValent number (0 = XADDR1, 1 = XADDR2)
     writeData(2) = 0  // Source of PMBus address offset (0 = XADDR1, 1 = XADDR2)
     writeData(3) = 0  // Source of I2C address offset (0 = XADDR1, 1 = XADDR2)
     i2c Write 4 Bytes (0x2005C504, writeData)


// 4. Write MFR_FIRMWARE_COMMAND with the data byte 0x04 to store register configuration to OTP
     PMBus Write Byte (MFR_FIRMWARE_COMMAND, 0x04)

// 5. Wait until writing to OTP is completed
     data(0) = 1
     i = 0
     While data(0) <> 0  // As long as data(0) is not 0, writing to OTP is ongoing
          If i >= 10 Then
               Exit While  // Exit if writing was not successful after 1s – do some fault handling
          End If
          i++
          Wait(100ms)
          PMBus Write Byte (MFR_FIRMWARE_COMMAND, 0x05)  // 0x05 = Get Configurator Status
          PMBus Block Read (MFR_FIRMWARE_COMMAND_DATA, data)
     End While
```

3.2 Storing PMBus configuration to OTP

There are two PMBus pages (0 and 1) for the two control loops of the device, which have to be stored to OTP separately. For the XDPP1100-Q040, both loops are used by the device, and therefore the configuration of both loops has to be written as PMBus User Section. For the XDPP1100-Q024, the device only uses loop 0. When programming a new device, loop 0 has to be written as PMBus User Section and loop 1 as PMBus Default Section. When changing some PMBus commands of a device which has been programmed before, you will usually only change loop 0 commands and afterward store the changed loop 0 User Section.

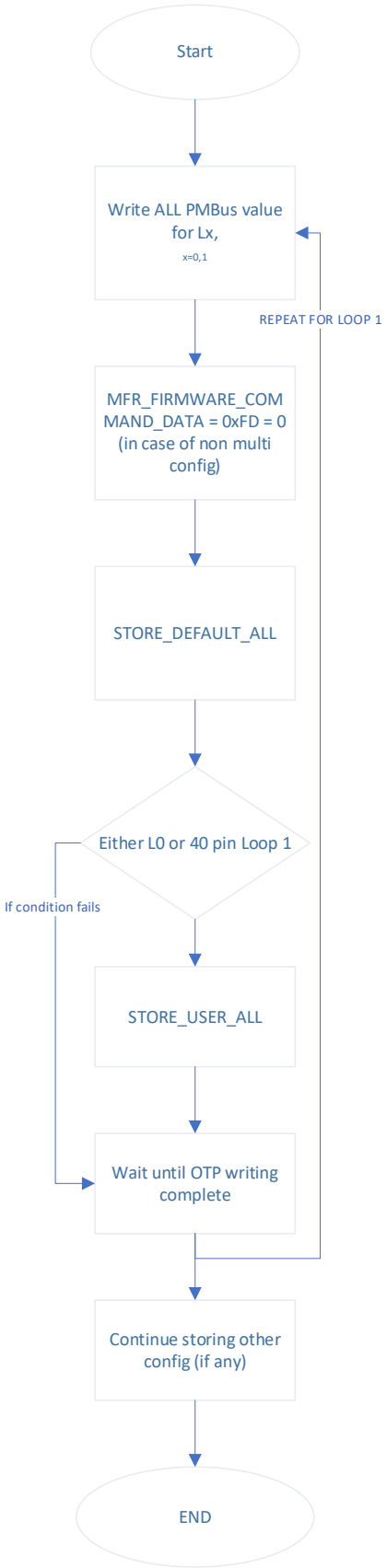


Figure 16 Programming Flow Chart

Code Listing 4

```
// There are two PMBus pages (0 and 1) for the two control loops of the device
     For loop = 0 to 1
// 1. Write the PMBus configuration (PMBus commands) to RAM
          // Write page command to select the loop
          // Write all PMBus commands for the selected loop
          PMBus Write Byte (PAGE, loop)  // Select loop 0 or 1
          For command code = start to end
               Write Pmbus command (command code, data bytes)
          Next
          // Follow the below steps to write the PMBus configuration from RAM to OTP

// 2. Write MFR_FIRMWARE_COMMAND_DATA with the following data
          writeData(0) = 0  // Xvalent number (usually 0 if multi-config is not used)
          writeData(1) = 0
          writeData(2) = 0
          writeData(3) = 0
          PMBus Block Write (MFR_FIRMWARE_COMMAND_DATA, writeData)

// 3. Write PMBus commands STORE_DEFAULT_ALL and STORE_USER_ALL to store PMBus
//     configurations to OTP
          // Store loop 0 and loop 1 Default sections – can be omitted when reprogramming
          PMBus Send Byte (STORE_DEFAULT_ALL)
          // Store loop 0 User for both device types – Store loop 1 User only for XDPP1100-Q040
          If ( (loop = 0) OR (Device = XDPP1100-Q040) ) Then
               PMBus Send Byte (STORE_USER_ALL)
          End If

// 4. Wait until writing to OTP is completed
          data(0) = 1
          i = 0
          While data(0) <> 0  // As long as data(0) is not 0, writing to OTP is ongoing
               If i >= 10 Then
                    Exit While  // Exit if writing was not successful after 1 s – do some fault handling
               End If
               i++
               Wait(100ms)
               PMBus Write Byte (MFR_FIRMWARE_COMMAND, 0x05)  // 0x05 = get configurator status
               PMBus Block Read (MFR_FIRMWARE_COMMAND_DATA, data)
          End While
     Next
```

References

[1] Infineon Technologies AG: *XDPP1100 programming instruction*; **Available online**

Revision history

Document revision	Date	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_162820

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.

Please note that this product is not qualified according to the AEC Q100 or AEC Q101 documents of the Automotive Electronics Council.

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.