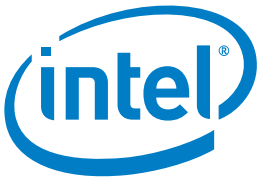


Intel® Enpirion® Power Solutions

EM21xx Family – Programming a New Configuration

Application Note

Contents

List of Figures

List of Tables

1. Introduction

The EM21xx family of PowerSoCs come with default configurations preprogrammed into the device. However, in the event the user wishes to update and customize the configuration for his end application, the EM21xx family supports programming of a completely new configuration using manufacturer specific PMBus commands thereby supporting the option to program in a production environment. The required programming file is created using the Intel Enpirion Digital Configurator GUI.

The reconfiguration is stored in a One-Time-Programmable (OTP) Non Volatile Memory (NVM) integrated into the Digital Controller. The EM21xx PowerSoC's onboard NVM supports up to three complete custom reconfigurations.

Refer to section 5 and also to the respective *EM21xx Data Sheet* regarding PMBus™ commands referenced in this document.

2. Creating the Programming File

In the event a user wants to update the configuration for an EM21xx PowerSoC in a production environment, the first step is to create the desired configuration, then generate the updated reconfiguration and finally, store it as a programming file using the Intel Enpirion Digital Configurator GUI . The contents of this programming file can be downloaded into the device using the procedure detailed below. More details on using the Intel Enpirion Digital Configurator GUI can be found in the EM21xx_GUI_User_Guide application note.

Once the user has finalized the configuration specific to his application, and is ready to generate the programming file, he needs to do the following in order to create the programming file.

1. Go to the "Control & Program" tab of the GUI and select "Export Model Configuration"

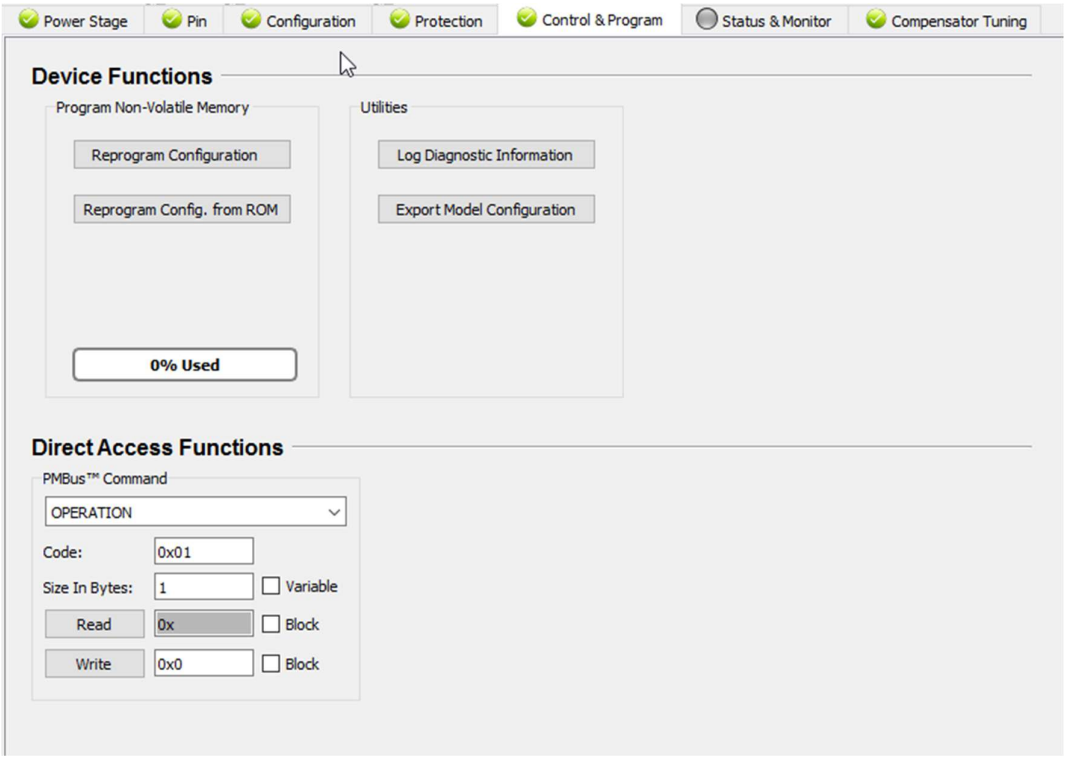


Figure 1: Control and Program Tab

2. The following dialog window appears which allows the user to select appropriate values.

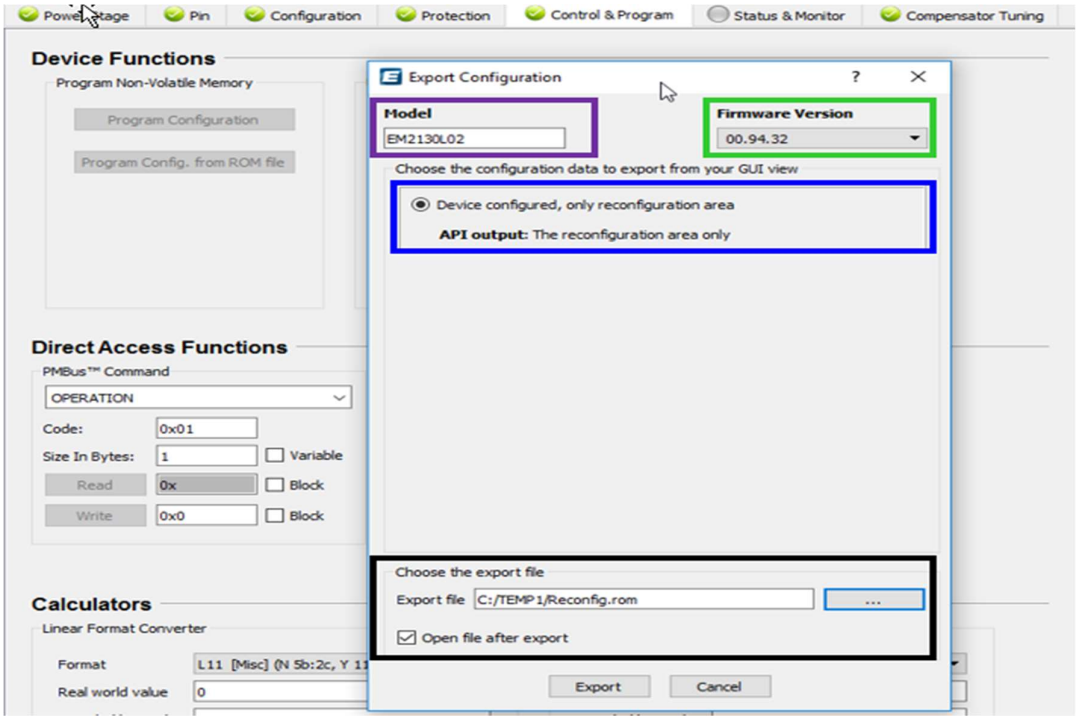


Figure 2: ROM File Dialog Box

- Highlighted in **Purple** – Ensure Correct Model type, e.g. 2130L02, 2130H01 etc.
  - Select appropriate EM21xx Model to the device being programmed. *The Model selected* **MUST** *match that of the device to be programmed.*
- Highlighted in **Green**
  - Select appropriate f/w revision to the device being programmed. *The F/W version selected* **MUST** *match that of the device to be programmed.*
- High-lighted in **Blue**
  - Information only
- Highlighted in **Black**
  - The user may select where the programming file is to be saved. They may also select the option to open the file automatically after exporting.

3. Select "Export" to save the file to a specific location. This file is in the ROM format and when selected, will open in the application specified such as notepad.

3. ROM File Format

The reconfiguration of the device is available in a ROM file format and can be used to program the device in production. The file format of this ROM file is defined as follows.

Comment lines start with // and should be ignored during programming.

The F/W revision with which the programming file is compatible is given in one of the comment lines, as highlighted **in Red Bold** in the below example.

Data lines are defined as follows.

@[OTP ADDRESS] ␣␣ [OTP DATA]

Where OTP_ADDRESS represents the address of the OTP cell and OTP_DATA, its contents. Both values are stored as a 16-bit word in hexadecimal format.

Example of an excerpt from a ROM file for a EM2130L02:

```
//FileName==TEST_ReConfigRomFile1.rom
//FileFormat=1.3
//
// Gui name:          "Enpirion Digital Power Configurator"
// Gui version:       "Version 1.2.0 (Build ****,5)"
// Gui export:        "Device configured, Reconfiguration snapshot export"
// Gui exported ranges:  "[0x0C00-0x0CF5]"
//
// Firmware package:     ""
// Firmware package rev:  "B0"
// Firmware version:     "00.94.32"
// Firmware patch level:  "0"
//
// Manufacturer id:      "INTL"
// Manufacturer model:   "2130L02"
// Manufacturer configId: ""
//
@0C00 4006
@0C01 0009
@0C02 0001
@0C03 00EF
@0C04 0016
@0C05 1CCD
```

4. Programming Configuration Routine

The steps to programming the device are listed below:

1. Optional: Check the current OTP status of the controller using the PMBus™ MFR_STORE_CONFIGS_REMAINING (0xEB) command. (It will consist of the number of remaining complete configurations available)
2. Compare the firmware (FW) version of the part to the FW version information stored in the ROM file to ensure proper operation. The FW version can be read using the MFR_REVISION command (0x9B) (see section 5). These **MUST** match unless otherwise stated.
3. Ensure the Manufacturer Model of the device matches the Manufacturer Model in the ROM file. The Manufacturer Model of the device can be read using the MFR_MODEL command (0x9A) (see section 5). These **MUST** match unless otherwise stated.
4. Prior to starting, read back the STATUS_CML and ensure all bits are clear (especially the CMD_FAULT b [7])
5. Use the MFR_STORE_CONFIG_BEGIN (0xEC) command to signal the commencing of a programming sequence into OTP (the start of a programming of a reconfiguration ROM file into NVM)
6. Use the MFR_STORE_CONFIG_ADDR_WRITE (0xED) command to program the reconfiguration value (identified by address and data in ROM format) into OTP.
   - As an example, to write the data value 0x647A from OTP address 0x100A the necessary I2C byte sequence would be Write: {0xED 0x0A 0x10 0x7A 0x64}. (Single I2C Transaction)
7. Verify the procedure by reading back the written value to the controller using the PMBus™ MFR_STORE_CONFIG_ADDR_READ (0XE9) command.
   - As an example, to read the data value 0x647A from OTP address 0x100A the necessary I2C byte sequence would be Write: {0xED 0x0A 0x10} & Read: {0x7A 0x64}. (Two I2C Transactions)
   - If the value read does not match the value programmed, then an unrecoverable programming error has occurred and the reconfiguration has failed. Provided there is sufficient remaining space the programming routine may be repeated.
8. Continue to repeat step 6 & 7 until the entire reconfiguration is loaded into OTP. The ROM file **MUST** be written sequentially line by line as given in the ROM file. The Read-back **MUST** be also be performed sequentially. after each write
9. Upon completion of loading OTP data into the device the PMBus™ MFR_STORE_CONFIG_END (0xEE) command **MUST** be used to communicate to the device the end of the reconfiguration of OTP routine
10. Finally read back the STATUS_CML and ensure all bits are clear (especially the CMD_FAULT b [7]). *If any bits are not clear an unrecoverable programming error has*

occurred and the reconfiguration has failed. Provided there is sufficient remaining space the programming routine may be repeated.

Optional: If there is any bit set within the STATUS_CML command, then the host is notified by asserting SMBALERT#.

Notes:

- The reconfiguration data is activated in the device only after a power cycle.
- For further reconfigurations, the above procedure should be repeated.
- The comment header of the ROM file contains information about the part for which it has been created.
- *The MFR_STORE_CONFIG_ADDR_READ (0XE9) command will not function after the issuing of the MFR_STORE_CONFIG_END (0xEE) command*

For further detail on each PMBus commands listed above, please refer to the next section.

5. PMBus Command Reference

5.1 STATUS_CML - 7Eh

The STATUS_CML command returns a summary of the communications status information in one data byte and is read only.

Table 1: STATUS_CML Command Return Summary

STATUS_CML (read only)		
Bits	**Name**	**Description**
[0]	Not supported	Not supported
[1]	SMBUS_FAULT	SMBus timeout or a format error have occurred
[2]	RESERVED	PMBus reserved
[3]	Not supported	Not supported
[4]	Not supported	Not supported
[5]	PEC_FAULT	A packet error check fault has occurred
[6]	Not supported	Not supported
[7]	CMD_FAULT	An invalid or an unsupported command has been received

5.2 MFR_STORE_CONFIG_ADDR_READ - E9h

The MFR_STORE_CONFIG_ADDR_READ PMBus command can be used to verify the configuration value programmed via the MFR_STORE_CONFIG_ADDR_WRITE PMBus command. *This command only functions within the programming routine (i.e. between the Start (ECh) and end (EEh) commands).* The command is implemented as a PMBus process call, taking the address as an input (host word write) and providing the value as an output (host word read).

This command must be used in conjunction with the MFR_STORE_CONFIG_ADDR_WRITE command. *It must be issued after the MFR_STORE_CONFIG_ADDR_WRITE command that programmed the configuration value to be verified.*

Table 2: MFR_STORE_CONFIG_ADDR_READ Structure

MFR_STORE_CONFIG_ADDR_READ (write)		
Bits	**Name**	**Description**
[15:0]	ADDRESS	Configuration address
MFR_STORE_CONFIG_ADDR_READ (read)		
Bits	**Name**	**Description**
[15:0]	DATA	Configuration value

As an example, to read the data value 0x647A from OTP address 0x100A the necessary I2C byte sequence would be Write: {0xED 0x0A 0x10} + Read: {0x7A 0x64}. *If the value read does not match the value programmed, then an unrecoverable programming error has occurred and the reconfiguration has failed.*

5.3 MFR_STORE_CONFIGS_REMAINING – EBh

The MFR_STORE_CONFIGS_REMAINING PMBus™ command returns the number of remaining times that a full configuration can be stored via the MFR_STORE_CONFIG_BEGIN, MFR_STORE_CONFIG_ADDR_WRITE, MFR_STORE_CONFIG_END commands. Each time a full configuration is stored, a portion of the available space in OTP is permanently used up.

If the device isn't already configured, i.e. has no initial configuration, then this command will return a negative number to indicate that the device has no configuration. In this case the magnitude of the return value indicates the number of times a configuration could be subsequently programmed once the device is configured.

Table 3: MFR_STORE_CONFIGS_REMAINING Command Structure

MFR_STORE_CONFIG_REMAINING (read only)		
Bits	**Name**	**Description**
[15:0]	CONFIGS_COUNT	The number of remaining times a full configuration can be successfully stored, in linear format. (A negative result indicates the device has not yet been configured OTP)

5.4 MFR_STORE_CONFIG_BEGIN - ECh

The MFR_STORE_CONFIG_BEGIN PMBus™ command is used to commence programming of configuration to OTP.

This command must be used in conjunction with the MFR_STORE_CONFIG_ADDR_WRITE and MFR_STORE_CONFIG_END commands. It must be issued prior to sending any MFR_STORE_CONFIG_ADDR_WRITE commands.

Table 4: MFR_STORE_CONFIG_BEGIN Command Structure

MFR_STORE_CONFIG_BEGIN (write only)		
Bits	**Name**	**Description**
No data		

5.5 MFR_STORE_CONFIG_ADDR_WRITE - EDh

This command must be used in conjunction with the MFR_STORE_CONFIG_BEGIN and MFR_STORE_CONFIG_END commands. It must be sent only after issuing a MFR_STORE_CONFIG_BEGIN command and prior to issuing a MFR_STORE_CONFIG_END command.

Table 5: MFR_STORE_CONFIG_ADDR_WRITE Command Structure

MFR_ STORE_CONFIG_ADDR_WRITE (write)		
Bits	**Name**	**Description**
[15:0]	ADDRESS	Configuration address
[31:16]	DATA	Configuration value

As an example, to read the data value 0x647A from OTP address 0x100A the necessary I2C byte sequence would be Write: {0xED 0x0A 0x10 0x7A 0x64}.

Example: Program Reconfiguration Section and Compensator #3

MFR_STORE_CONFIG_BEGIN

MFR_STORE_CONFIG_ADDR_WRITE (ADDRESS=0x0C00, DATA=0x....)

MFR_STORE_CONFIG_ADDR_WRITE (ADDRESS=0x0C01, DATA=0x....)

...

MFR_STORE_CONFIG_ADDR_WRITE (ADDRESS=0x0C33, DATA=0x....)

MFR_STORE_CONFIG_ADDR_WRITE (ADDRESS=0x0C34, DATA=0x....)

MFR_STORE_CONFIG_ADDR_WRITE (ADDRESS=0x11B8, DATA=0x....)

MFR_STORE_CONFIG_ADDR_WRITE (ADDRESS=0x11B9, DATA=0x....)

...

MFR_STORE_CONFIG_ADDR_WRITE (ADDRESS=0x11F2, DATA=0x....)

MFR_STORE_CONFIG_ADDR_WRITE (ADDRESS=0x11F3, DATA=0x....)

MFR_STORE_CONFIG_END

The MFR_STORE_CONFIG_ADDR_WRITE PMBus™ command is used to program a configuration value (identified by address and data) to OTP. A sequence of these commands can be used to efficiently program an entire configuration to OTP.

The address/data pair corresponds directly to one line in a configuration ROM file. *Addresses must be programmed in strict ascending order, as they appear in a configuration ROM file.*

Table 6: Section of NVM Configuration and Reconfiguration Address Range

Start Address	End Address	Description	Description
0x0C00	0x0FF9	Reconfiguration Section	Configuration Section

Notes:

- A normal Reconfiguration file will only contain data within the address 0x0C00 to 0x0FF9.
- For all address outside those listed in Table 6 above, any attempt to write to these locations will fail.
- Addresses in the range 0x0C00 – 0x0FF9 refer to the reconfiguration section. Any writes to these addresses will be remapped into the available space remaining in the reconfiguration section. If there is insufficient space remaining the command will fail.
- For all address in Table 6, if the user attempts to write to an already programed OTP location, the attempt will fail.

5.6 MFR_STORE_CONFIG_END - EEh

The MFR_STORE_CONFIG_END PMBus™ command is used to complete programming of configuration to OTP.

This command must be used in conjunction with the MFR_STORE_CONFIG_BEGIN and MFR_STORE_CONFIG_ADDR_WRITE commands. It **MUST** be issued after sending all required MFR_STORE_CONFIG_ADDR_WRITE commands.

Table 7: MFR_STORE_CONFIG_END Command Structure

MFR_STORE_CONFIG_END (Write Only)		
Bits	Name	Description
No data		

5.7 MFR_MODEL – 9Ah

The MFR_MODEL command retrieves text (ISO/IEC 8859-1 [A05]) characters that contain the manufacturer's model number, as configured via the 48-bit MFR_MODEL OTP cell.

Table 8: MFR_MODEL Command Structure

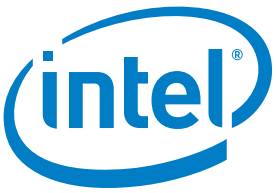
MFR_REVISION (Read)	
Bits	**Description**
[7:0]	"**2**" = 0x32
[15:8]	"**1**" = 0x31
[23:16]	"**3**" = 0x33
[31:24]	"**0**" = 0x30
[39:32]	"**L**" = 0x4C / "**H**" = 0x48
[47:40]	"**0**" = 0x30
[55:48]	"**2**" = 0x32 / "**1**" = 0x31
[63:56]	"" = 0x00

5.8 MFR_REVISION 9Bh

The MFR_REVISION command retrieves text (ISO/IEC 8859-1 [A05]) characters that contain the manufacturer's revision number. This is the firmware revision number, e.g. "00.94.32".

Table 9: MFR_REVISION Command Structure

MFR_REVISION (Read)	
Bits	**Description**
[7:0]	“0” = 0x30
[15:8]	“0” = 0x30
[23:16]	“.” = 0x2E
[31:24]	“9” = 0x39
[39:32]	“4” = 0x34
[47:40]	“.” = 0x2E
[55:48]	“3” = 0x33
[63:56]	“2” = 0x32



6. Revision History

Revision Number	Description	Revision Date
001	Initial release.	March 2017
002	Updated related to new F/W revision	May 2017