IRPS5401 programming guide

For IRPS5401 rev 02/03 silicon

About this document

Scope and purpose

The scope of this document covers the functionality of the Non-volatile Memory (NVM) space of the USER and CNFG sections of the IRPS5401 PMIC device. The purpose of the document is to provide the user with the information required to perform the programming steps in their own programming environment. This could be "In-circuit" programming in manufacturing, in-house gang programming device prior to device installation, or in a prototyping lab environment.

Intended audience

Engineering or manufacturing users that require information on how to program the memory space of the IRPS5401 device.

Table of contents

1 Programming overview

There are three "One Time Programmable" (OTP) sections in each device, namely the configuration section, trim section, and user section. The trim section contains the device calibration data which is programmed at the factory. The configuration section (CNFG) stores device specific information and allows 5 writes. The user section (User) consumes most of the OTP memory and can have up to 26 writes.

The IRPS5401 address map is based on 16-bit addresses and each register is 2 bytes wide organized as little endian.

Throughout this document, the small "h" after or "ox" before a number indicates that the number is displayed in "hex" format.

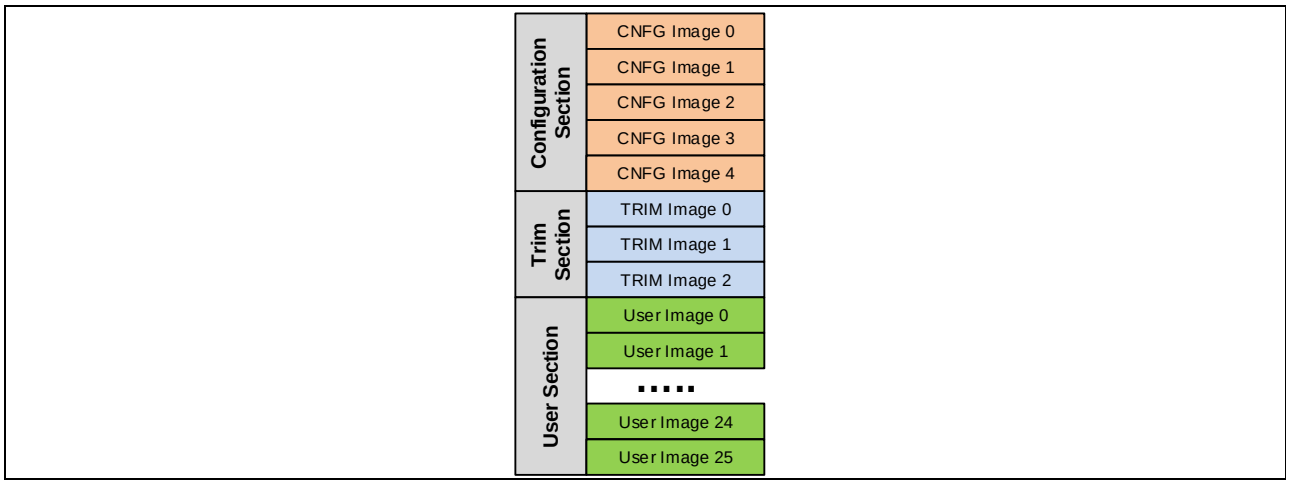

Figure 1 IRPS5401 OTP images

2 Hardware settings and connections

Several IO pins of the IRPS5401 device must be set correctly as shown in Figure 2. Other pins can be left floating.

- I2C clock and data are for communication and programming.
- MTP is an input to enable the I2C test mode address, 0xA, by connecting MTP to 3.3 V prior to device POR.
- 1V8 is the internal 1.8 V LDO output which is derived from the 5.0 V Vcc. A good 1.8 V is an indication that the internal logic is functioning properly.

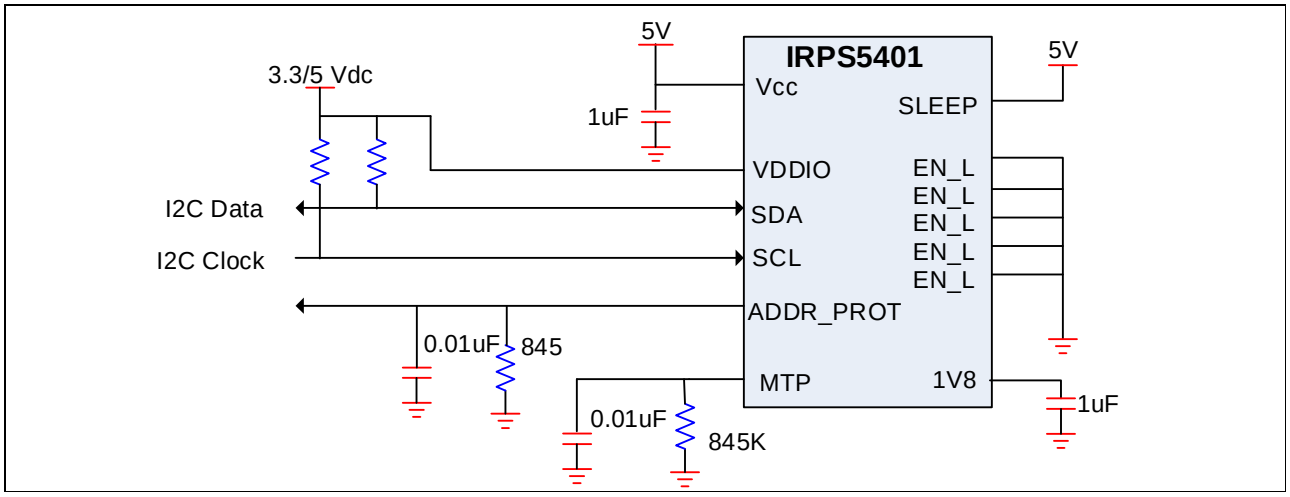

Figure 2 IRPS5401 critical hardware settings and connections

3 Programming registers

3.1 Test mode I2C address

The I2C address of a device is determined by the content of register 0x20[14:8] during normal operation. However, a device can be operated in test mode by applying 3.3 V to the MTP pin prior to applying 5 V to the Vcc pin. In test mode, the I2C address is fixed at 0Ah (7-bit) no matter what value is stored in the I2C register. In this guide, the test mode will be used for all register read and write operations.

3.2 Silicon version register

The silicon version value are stored in register 0xFD[7:0] of page 0. To use the programming algorithms described in this document, the silicon register value must be 2 or higher.

3.3 Register access – I2C read/write command

There are three I2C protocols as shown in Figure 3 that are used; Write Byte, Read Byte, and Write Word. Byte commands are used for single register byte read and write operations. **Write word command is only used to send the 16-bit programming command to REG 88h.**

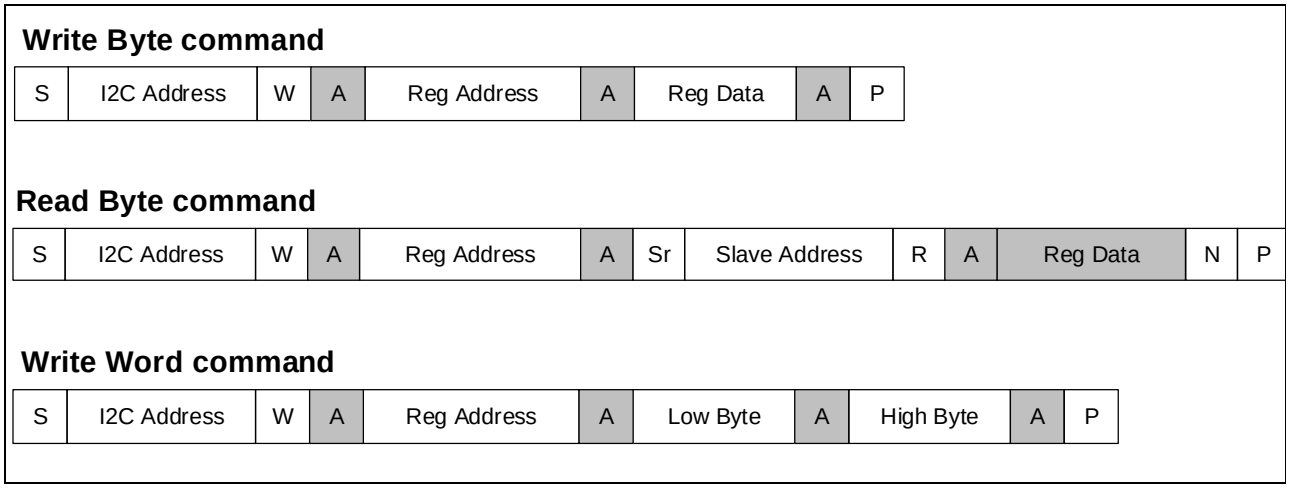

Figure 3 Read/write commands

Each device has 6144 registers – address 0 to 6143, they are divided into 24 pages - page number 0 to 23 (00h to 17h). So each page has 256 registers. The page number register is located at address FFh. For example, to write or read register at address 515

Page number = 515 \ 256 = 2

Register address = 515 mod 256 = 3

1. Write 2 to register FFh
2. Write or read data at register 3

3.4 Programming writes left register

Programming writes left information is stored in the read only register 0x56[4:0] for CNFG and 0x58[15:0] + 0x5A[15:0] for USER. Each bit indicates programming availability of the user image. 0 means the image is blank and available for programming. 1 means the image has been programmed.

Table 1

Section	Register address
CNFG	0x56[4:0]
USER	0x58[15:0] + 0x5A[15:0]

Examples:

1. If 0x56[4:0] = 00111, the first three images (0, 1, 2) of the CNFG section have been programmed. The next programming pointer should be set to 3.
2. If 0x58[15:0] + 0x5A[15:0] = 00000000 00000000 00000000 00011111, the first 5 images (0, 1, 2, 3, 4) of the USER section have been programmed. The next programming pointer should be set to 5.

3.5 Programming command register 88h

Programming command is made of 16 bits as shown in Figure 4. There are three fields that need to be determined by the user – image number [13:8], one of the section bits [7:4], and opcode [3:0]. Other bits must be set to '0'.

The NVM_COMMAND register			
field	**bits**	**encoding**	**description**
Done	15	1	Command has completed
		0	Ready (or command is in progress). Set this bit to zero when issuing a new command.
Error	14	1	An error occurred during a previous NVM operation. See the other NVM status registers for additional information
		0	The previous operation completed successfully
Image	13:8		image number to be accessed (0x3F = read the most recent image or program the next available image)
Section	7	rsv	This bit field specifies the NVM section(s) to be accessed. Examples: to read just the TRIM section, set this field to 'b0010. To read the CNFG, TRIM and USER sections set this field to 'b0111.
	6	USER	
	5	TRIM	
	4	CNFG	
Opcode	3:0	15 - 3	*reserved*
		2	PROGRAM: write the values from the REGMAPs into the NVM
		1	READ: restore the values from NVM to the REGMAPs.
		0	IDLE: nop

Figure 4 NVM command register

Examples:

1. If 0x56[4:0] = 00111, the next CNFG programming command to register 88h should be 0312h.
2. If 0x58[15:0] + 0x5A[15:0] = 00000000 00000000 00000000 00011111, the next USER programming command should be 0542h.

4 Single image programming flowchart

The three-column configuration file - see Appendix A - is used to program the single image. The flow of programming is outlined in Figure 5.

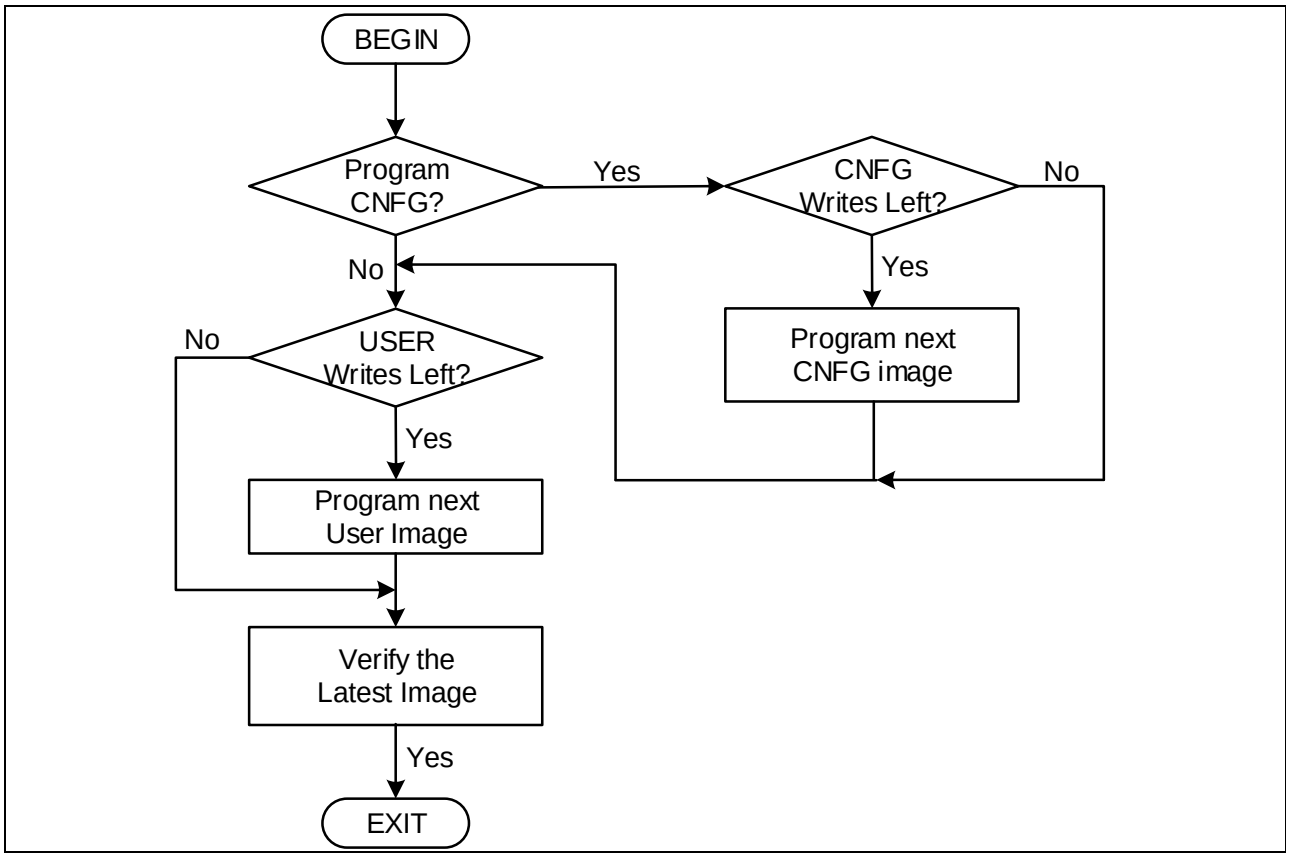

Figure 5 Single image flowchart

5 Mutiple image programming flowchart

To program a multiple image, the multiple image configuration file (mic) - see Appendix B – must be used for the programming. The flow of programming is outlined in Figure 5.

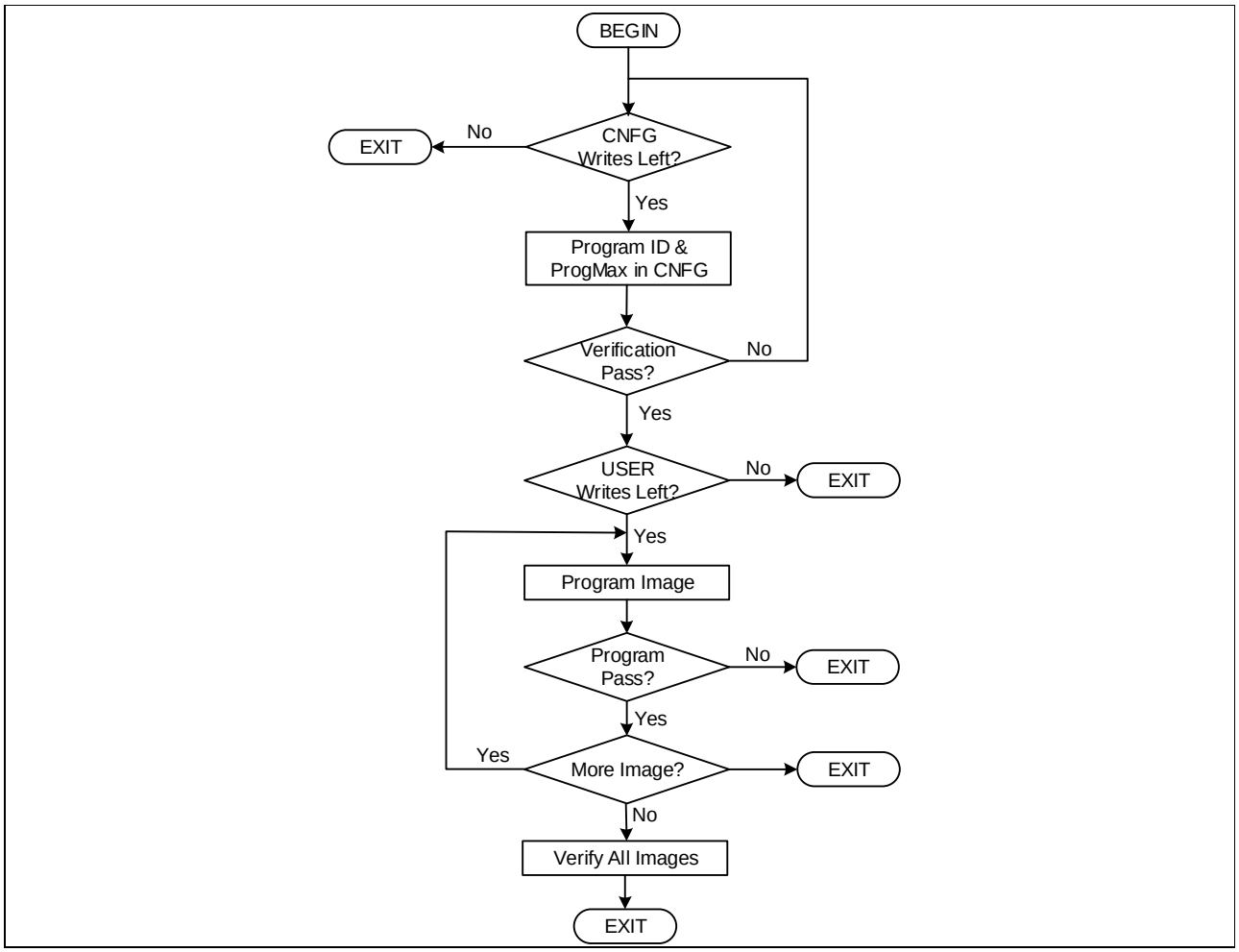

Figure 6 Multi image flowchart

6 Programming section details

6.1 CNFG section

1. Set MTP pin to 3.3 V. The I2C address (7-bit) is fixed at 0x0A.
2. Apply 5 V to Vcc pin and delay 10 ms.
3. Check 1.8 V +/- 5% is present at 1V8 pin. If 1.8 V is not detected, the I2C communication will not work.
4. Write 0x00 to register 0x86.
5. Read register bit 0x6C[1].
    - If it's 0, write 0x5A to register 0x8A and 0xA5 to register 0x8B
    - If it's 1, do nothing.
6. Write 0 to register 0x0B
7. Read register 0x56[4:0] to determine next CNFG write pointer.
8. Write CNFG data to register 0 and 1
9. Write programming command 0xPP12 to register 88h using I2C write word command, where PP can be 00, 01, 02, 03, or 04 depending on which image will be programmed in. Refer to section 6 for next programming pointer details.
10. Check register 0x89[7] repeatedly for programming progress. 0 = in progress. 1 = done. Or use a timer for 250 ms.

6.2 USER section

1. Set MTP pin to 3.3 V. The I2C address (7-bit) is fixed at 0x0A.
2. Apply 5 V to Vcc pin and delay 10 ms.
3. Check 1.8 V +/- 5% is present at 1V8 pin. If 1.8 V is not detected, the I2C communication will not work.
4. Write 0x00 to register 0x86
5. Read register bit 0x6C[1].
    - If it's 0, write 0x5A to register 0x8A and 0xA5 to register 0x8B
    - If it's 1, do nothing.
6. Write 0 to register 0x0B
7. Write configuration data to:
    - Page 0h : 0x20 – 0x3B (register addresses 0x20 – 0x3B)
    - Page 4h : 0x20 – 0x2B (register addresses 0x420 – 0x42B)
    - Page 6h : 0x00 – 0xFF (register addresses 0x600 – 0x6FF)
    - Page 7h : 0x00 – 0xFF (register addresses 0x700 – 0x7FF)
    - Page 8h : 0x20 – 0x2B (register addresses 0x820 – 0x82B)

- Page Ah : 0x00 – 0xFF (register addresses 0xA00 – 0xAFF)
- Page Bh : 0x00 – 0xFF (register addresses 0xB00 – 0xBFF)
- Page Ch : 0x20 – 0x2B (register addresses 0xC20 – 0xC2B)
- Page Eh : 0x00 – 0xFF (register addresses 0xE00 – 0xEFF)
- Page Fh : 0x00 – 0xFF (register addresses 0xF00 – 0xFFF)
- Page 10h : 0x20 – 0x2B (register addresses 0x1020 – 0x102B)
- Page 12h : 0x00 – 0xFF (register addresses 0x1200 – 0x12FF)
- Page 13h : 0x00 – 0xFF (register addresses 0x1300 – 0x13FF)
- Page 14h : 0x20 – 0x21 (register addresses 0x1420 – 0x1421)
- Page 16h : 0x00 – 0xFF (register addresses 0x1600 – 0x16FF)
- Page 17h : 0x00 – 0xFF (register addresses 0x1700 – 0x17FF)

8. Write programming command 0xPP42 to register 88h using I2C write word command, where PP can be 00h – 19h (0 - 25) depending on which image will be programmed in. Refer to section 6 for next programming pointer details.
9. Check register 0x89[7] repeatedly for programming progress. 0 = in progress. 1 = done. Or use a timer for 250 ms.
10. If the programming progress bit 0x89[7] is set, then check register 0x89[6] for programming status. 0 = succeeded. 1 = failed

6.3 Verification of the CNFG and USER section

1. Set MTP pin to 3.3 V. The I2C address (7-bit) is fixed at 0x0A.
2. Apply 5 V to Vcc pin and delay 10 ms.
3. Check 1.8 V +/- 5% is present at 1V8 pin. If 1.8 V is not detected, the I2C communication will not work.
4. Send read command 0xPP41 to register 88h, where PP can be 00h – 13h (0 - 19) depending on which image will be read.
5. Check register 0x89[7] repeatedly for command progress. 0 = in progress. 1 = done. Or use a timer for 250 ms.
6. Read register 0x52[6:4]. 0x52[6] = User; 0x52[5] = Trim; 0x52[4] = CNFG; If any bit is 1, that means there are CRC errors in that section. That section has to be reprogrammed.
7. Read all register from 0x0000 to 0x17FF.
8. Always verify common registers. But only verify switcher registers if they are not disabled based on register 0x38[4:0]. Switcher D can also be disabled by register bit 0x23[4]. See details in section 12.
9. Compare read data with configuration data using associated masks. Note: The following registers should be ignored due to an NVM restore bug - 0x16F9, 0x16FB, 0x16FD, 0x17B0, 0x17BC.

Note: *The latest CNFG OTP data will be loaded into the registers on power up. Therefore, there is no need to use an OTP read command for the CNFG registers.*

7 Output disable register

Register 0x38[4:0] of page 0 is used to disable an output when its bit value is set to 1.

Table 2

Register Bit	Switcher
0x38[4]	LDO
0x38[3]	D
0x38[2]	C
0x38[1]	B
0x38[0]	A

Register bit 0x23[4] of page 0 is used to combine switchers C and D as a single output. The switcher D is disabled when this bit is set to 1.

8 Appendix A

A three-column configuration file is used for single image programming. An example file format is shown in Figure 7.

1. User comments area. Any text can be added above CRC32 line. Each line must begin with //.
2. File headers area.
3. Configuration data in hex values.
    - Column 1 is the address of the register
    - Column 2 is the data of the register
    - Column 3 is the mask for each bit in a register. If the mask bit is 0 then the corresponding data bit shall be ignored during verification after programming. For instance, if the mask is FB which is equivalent to 1111 1011 in binary, then bit 2 value can be ignored after programming.
4. CRC32 is calculated and covered from the line below CRC32 value to the end of the file.

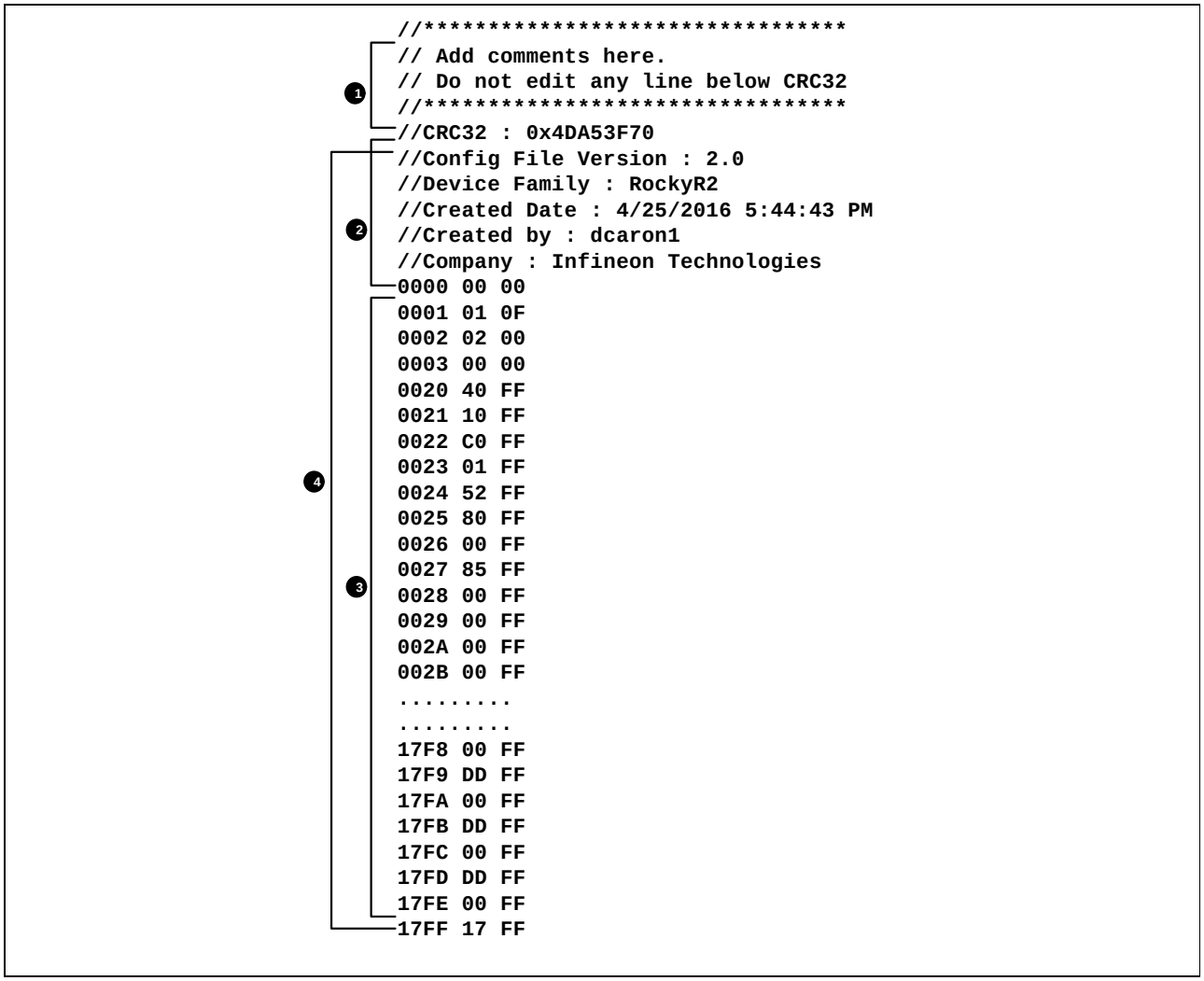

Figure 7 Single image config file

9 Appendix B

For multiple image programming, a multi-image configuration file (.mic) shall be used. A four-image mic file example is shown in Figure 8.

1. User comments area. Any text can be added above CRC32 line. Each line must begin with //.
2. File headers area.
3. CNFG section data. Four bytes. From address 0 to 1
4. Masks in hex value for verification purpose. There are 384 rows and 16 bytes per row. A bit 0 means the corresponding register data doesn't need to be verified after programming. A bit 1 means verification is required.
5. Images of individual configuration file. There are 384 rows and 16 bytes per row representing the complete device register data 6144 bytes.
6. CRC32 is calculated and covered from the line below CRC32 value to the end of the file.

```
//*********************************
// Add comments here.
// Do not edit any line below CRC32
//*********************************
//CRC32 : 0x0C6DE0FA
//MIC File Version : 2.0
//Device Family : RockyR2
//Created Date : 4/24/2016 7:25:16 PM
//Created by : dcaron1
//Company : Infineon Technologies
//Image Count : 4
[CNFG]
00 04 55
[Mask]
F0 0F F3 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
.........
.........
00 00 00 00 00 00 FF 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 FF FF
[Image 00]
Config File : RockyR2 i2c x20 pmb x50 gpr x1111.txt
00 00 00 00 00 00 00 00 85 04 80 80 20 00 00 00
00 00 00 00 00 00 00 00 5A 36 00 00 00 00 FF FF
.........
.........
FA 00 00 DD 00 DD 08 00 00 DD 00 DD 02 00 02 00
51 00 00 00 00 00 00 00 00 DD 00 DD 00 DD 00 0B
[Image 01]
Config File : RockyR2 i2c x20 pmb x50 gpr x1212.txt
00 00 00 00 00 00 00 00 85 04 80 80 20 00 00 00
00 00 00 00 00 00 00 00 5A 36 00 00 00 00 FF FF
.........
.........
FA 00 00 DD 00 DD 08 00 00 DD 00 DD 02 00 02 00
51 00 00 00 00 00 00 00 00 DD 00 DD 00 DD 00 0B
[Image 02]
Config File : RockyR2 i2c x20 pmb x50 gpr x1818.txt
00 00 00 00 00 00 00 00 85 04 80 80 20 00 00 00
00 00 00 00 00 00 00 00 5A 36 00 00 00 00 FF FF
.........
.........
FA 00 00 DD 00 DD 08 00 00 DD 00 DD 02 00 02 00
51 00 00 00 00 00 00 00 00 DD 00 DD 00 DD 00 0B
[Image 03]
Config File : RockyR2 i2c x20 pmb x50 gpr x2020.txt
00 00 00 00 00 00 00 00 85 04 80 80 20 00 00 00
00 00 00 00 00 00 00 00 5A 36 00 00 00 00 FF FF
.........
.........
FA 00 00 DD 00 DD 08 00 00 DD 00 DD 02 00 02 00
51 00 00 00 00 00 00 00 00 DD 00 DD 00 DD 00 0B
[End]
```

1 2 3 4 5 6

Figure 8 Multi image config file (MIC)

10 Appendix C

The configuration file integrity is protected by a CRC32 value. Below is the Visual Basic sample codes of CRC32 calculation used in the configuration file.

```
Private CRC32Table(255) As Integer

    Private Sub InitializeCRC32Table()
        ' This is the official polynomial used by CRC32 in PKZip.
        ' Often the polynomial is shown reversed (04C11DB7).
        Dim dwPolynomial As Integer
        dwPolynomial = &HEDB88320
        Dim i, j As Integer
        Dim dwCrc As Integer

        For i = 0 To 255
            dwCrc = i
            For j = 8 To 1 Step -1
                If (dwCrc And 1) Then
                    dwCrc = ((dwCrc And &HFFFFFFFE) \ 2) And &H7FFFFFFF
                    dwCrc = dwCrc Xor dwPolynomial
                Else
                    dwCrc = ((dwCrc And &HFFFFFFFE) \ 2) And &H7FFFFFFF
                End If
            Next j
            CRC32Table(i) = dwCrc
        Next i

    End Sub

Public Function CalculateCRC32(ByVal line() As String, ByVal lineCount As Integer) As
Integer

Dim crc32Result As Integer = &HFFFFFFFF
Dim iLookup As Integer
Dim i, j As Integer
Dim chrByte As Byte

InitializeCRC32Table()
        For i = 0 To lineCount - 1
            For j = 0 To line(i).Length - 1
                chrByte = CByte(Asc(line(i).Substring(j, 1)))
                iLookup = (crc32Result And &HFF) Xor chrByte
                ' shift right 8 bit; VB style
                crc32Result = ((crc32Result And &HFFFFFF00) \ &H100) And &HFFFFFF
                crc32Result = crc32Result Xor CRC32Table(iLookup)
            Next
        Next
        Return (Not crc32Result)


End Function
```

Revision history

Document version	Date of release	Description of changes
1.6	2/1/2018	Update section 6.1 step 5 Update section 6.2 step 5 and step 7
1.5	10/19/2017	Add a note in step 9, section 6.3.
1.4	6/16/2017	Update section 6.3 and 7 about combined switcher C and D setting during verification.
1.3	6/14/2017	Change maximum image number from 20 to 26 in USER section
1.2	7/5/2016	Add step 4, 5, 6 in programming CNFG and USERsections
1.1	6/27/2016	1. Add section 3.2 of silicon version register 2. Delete step to check 0x89[6] after OTP write or read command in section 6. 3. Add output disable register of section 7 4. Change verification process in section 6.3
1.0	4/25/2016	Initial release

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

Edition 2018-02-01

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Email: erratum@infineon.com

Document reference
TB0035

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.