OptiMOS™ IPOL (R3B & R4B) programming guide

PMBus™ capable: IR38163/IR38263/IR38363

I2C only: IR38165/IR38265/IR38365

Important: Only for parts that have been trimmed at the factory

There are several ways to program (configure) the OptiMOS™ IPOL family of devices. Customers are advised to use methods #1 and #2 only. Method #3 is intended for Programming Station manufacturers.

TABLE 1: PROGRAMMING OPTIONS

	Programming Option	Hardware Required	Benefit	Typical Application	Documentation
#1	**Multi-Device Programmer (in PowIRCenter GUI)**	3-pin connection to IR USB005 Dongle	• Single Push-button via easy graphical interface • Program all IR parts on a board at once	Low volume programming when parts are already soldered onto a board	PowIRCenter User Guide
#2	**PMBus™ command**	Connector to PMBus™	• Re-usable PMBus™ code that can be used for programming and run-time telemetry/updates • Simple programming command for USER section	Boot time on-board programming	This document, Section 1
#3	**Custom I2C code**	Connector to I2C bus	• Customizable for automated programming station using text-based config files for USER & MFR section	High volume programming station prior to assembly	This document, Section 2

For option #1, please refer to the PowIRCenter GUI User Guide.

Although OptiMOS™ IPOL has only one set of bus inputs for (SCL/SDA), it responds to two separate addresses: one is used to accept PMBus™ protocol and the other is used to accept I2C protocol. OptiMOS™ IPOL has default base PMBus™ address of 40h and I2C default base address of 10h. Unless the MSB of register 21h is set, the value of the R_{ADDR} resistor offsets the PMBus™ and I2C base address to form the final addresses for communication. If this MSB is set, then the R_{ADDR} resistor offset is ignored and the device address is the same as the base address.

OptiMOS™ IPOL has a limited programmable user Non Volatile Memory (NVM), also known a Multiple times Programmable (MTP). Parts from the factory are typically programmed 2 times, so customers usually have 7 programming attempts available.

Section 1: Programming USER section by PMBus™ Command

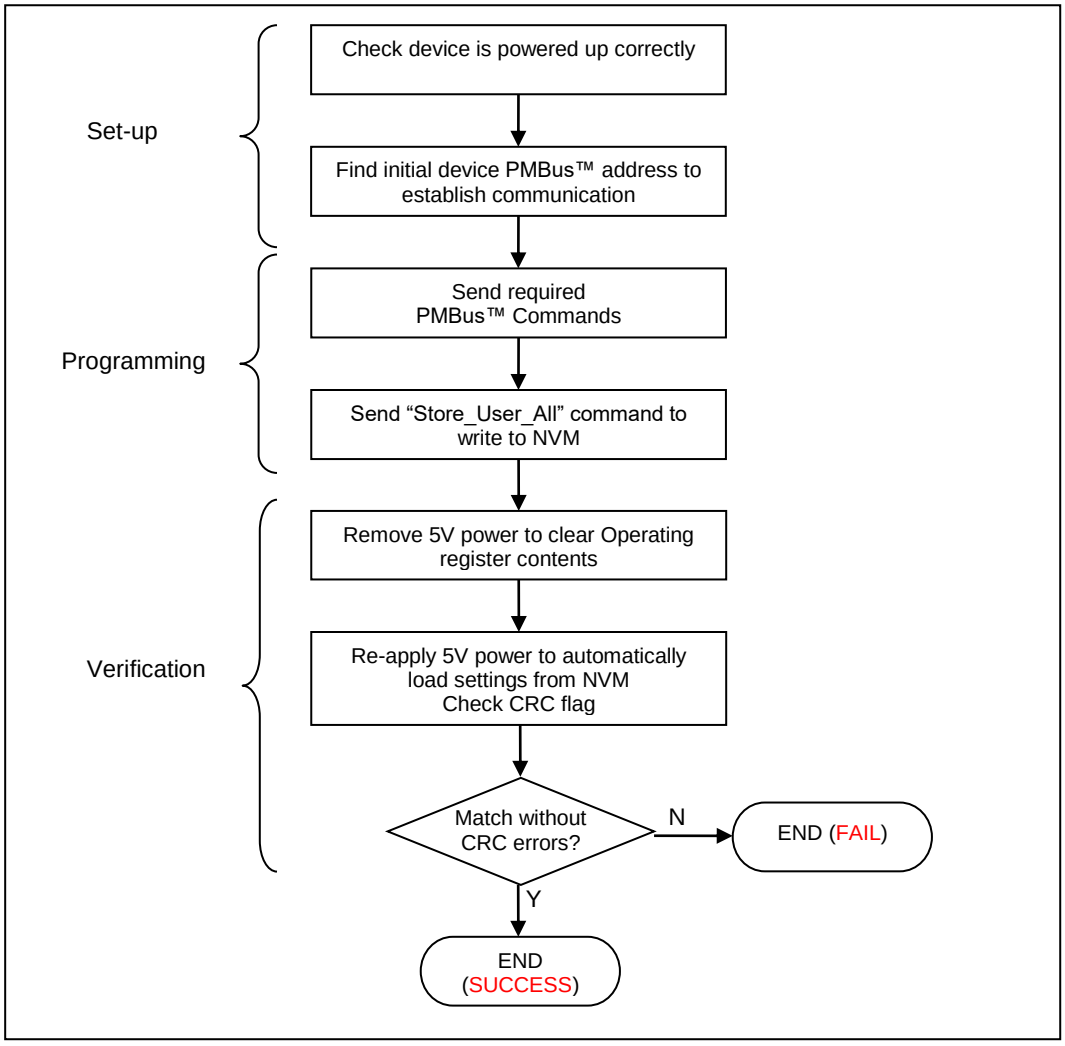
OptiMOS™ IPOL operates primarily by PMBus™ command and this is the recommended method for customers to configure parameters due to its simplicity. Figure 1 shows the PMBus™ programming flow.

Note, standard PMBus™ commands are used to set up parameters and write the NVM. Optionally, to access special functions such as the NVM count or the CRC flag, requires the use of manufacturer specific PMBus™ commands using the Process Call format:

MFR_READ_REG: use PMBus™ command D0h as shown in Figure 6

MFR_WRITE_REG use PMBus™ command D1h as shown in Figure 5

Figure 1: PMBus™ Programming Flow



Programming Procedure via PMBus™ (applies only to PMBus™ capable parts)

Use the PMBus™ read/write protocols in Figure 5 and Figure 6 to read and write registers. Refer to the "*PMBus™ commandset (UN7005)*" for a detailed list of valid commands and formats.

Set-up & Check

1. Check Vcc is 5V.
2. Check P1V8 pin voltage is 1.8V +/- 5%.

Determine PMBus™ address

3. Find out what is the PMBus™ address of the device (if not known).
   a) Send an innocuous PMBus™ command such as CAPABILITY from address 08h to 0Bh and 0Dh to 77h until the device ACKs the command.

b) Using the address obtained in step 3a), send the command IC_DEVICE_ID and check if the data is 63h, 64h, 65h, 66h, 67h or 68h (IR38163/165/263/265/363/365 respectively). If yes, the device is a OptiMOS™ IPOL device. Record this PMBus™ address and go to step 4. If not, it is either the I2C address of the OptiMOS™ IPOL device or another device on the bus, thus continue sending next address in step 3a).

Send required PMBus™ commands

4. Send any required PMBus™ commands (e.g. VOUT_COMMAND, FREQUENCY_SWITCH …)

Write New Configuration into MTP Memory

5. Send the STORE_USER_ALL command

Verification

6. Remove the 5V power to clear the operating memory.
7. Re-apply 5V power. The device will initialize and transfer its NVM contents into the operating memory
8. Read the CRC flags by sending MFR_READ_REG with Register-Address=97h (refer to Figure 6)
9. Check the CRC error flag 97h[2:0] = 000 (successful) – see Table 10.

Section 2: Programming by I2C Custom Code

MTP Register Space

There are two sections in the MTP register space that may be accessed by the user, namely, the user section, and if required, the manufacturer section. Each section can be programmed a finite number of times as shown in table 2. Registers are provided to indicate the number of MTP programming times that are left for each section. Devices are always trimmed at the factory and should **never** be changed by users. *Throughout this document, the small "h" after a number indicates that the number is displayed in "hex" format.*

TABLE 2: MTP SECTION ADDRESS RANGES

Section	Register Address Range	Max # of Programming Attempts
User	20h – 87h	9
Manufacturer (MFR)	8Ah – 8Fh	3

- **Factory defaults part will already have 2 user MTP banks written, so the user will have a max of 7 programming attempts remaining**

TABLE 3: MTP PROGRAMMING TIMES LEFT REGISTER (PTR)

Section	Register Address
Manufacturer (MFR)	98h[2:0]
User	99h[3:0]

TABLE 4: PTR VALUE DEFINITIONS FOR MANUFACTURER SECTION

98h[2:0]	Remaining Programming Times	Set Next Programming Pointer to
7	3	0
0	2	1
1	1	2
2 - 6	0	none left

TABLE 5: PTR VALUE DEFINITIONS FOR USER SECTION

99h[3:0]	Remaining Programming Times	Set Next Programming Pointer to
15	9	0
0	8	1
1	7	2
2	6	3
3	5	4
4	4	5
5	3	6
6	2	7
7[1]	1	8
8 - 14	0	none left

Note1: Due to the device bug, the user section pointer at register 0x99[3:0] remains at 7 after the last image 8 is programmed.

MTP Programming commands

The MTP read/write Command register is located at address B0h, which also serves as a "Return" register for the results of the command. After the write command is issued, the command register should be polled periodically until B0[7:5] is changed to 000 ("IDLE" state - see Table 8) *OR* the worst case programming time is exceeded.

TABLE 6: USER WRITE COMMAND STRUCTURE (REGISTER B0H)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit2	Bit1	Bit0
0	1	0	0	Next Programming Pointer 0 - 8			

TABLE 7: MANUFACTURER WRITE COMMAND STRUCTURE (REGISTER B0H)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit2	Bit1	Bit0
0	1	0	1	1	Next Programming Pointer 0 - 3		

TABLE 8: "RETURN" IDLE STATE STRUCTURE (REGISTER B0H)

Bit[7:5]	Bit 4	Bit 3	Bit 2	Bit1	Bit 0
000 = operation complete XXX = operation still processing	Bit[7:5] from MTP write/load command			0=success 1=fail	0

e.g. returns 08h for if user/mfr write is successful, returns 0Ah if user/mfr write fails, returns 04h if load command is successful.

TABLE 9: LOAD MTP TO OPERATING REGISTER COMMAND STRUCTURE (REGISTER B0H)

Bit[7:5]	Bit 4	Bit 3	Bit 2	Bit1	Bit 0
001	00000				

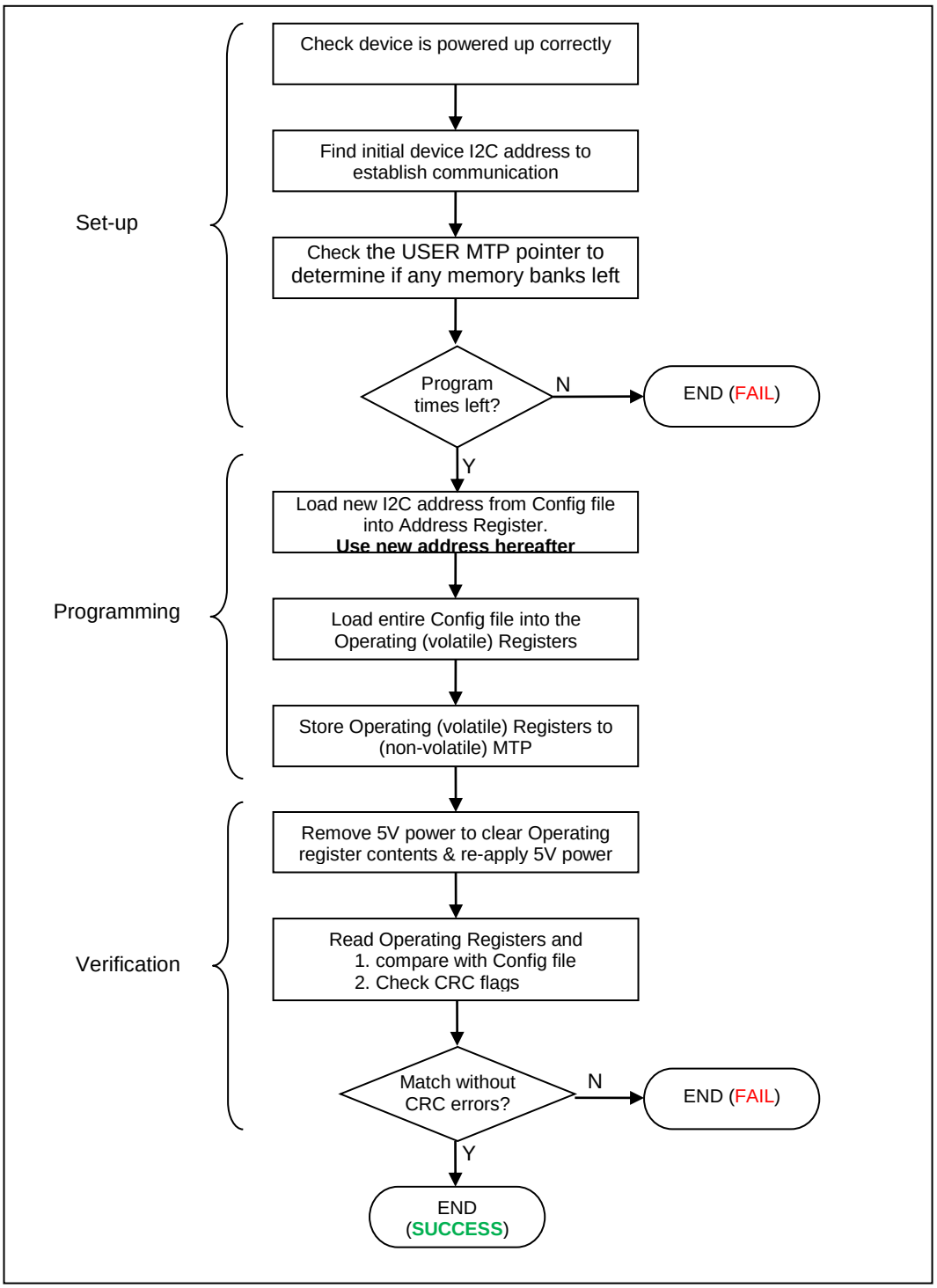
TABLE 10: CRC ERROR FLAGS (REGISTER 97H)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit1	Bit 0
xxxxx					Trim CRC Flag 0 = no error 1 = error	User CRC Flag 0 = no error 1 = error	MFR CRC Flag 0 = no error 1 = error

Programming Flow

Figure 2 provides an overview of the programming flow. A verification of the “store” should be done at the end.

Figure 2: Programming Flow



Programming Procedure via I2C

Use the I2C read/write protocols in Figure 4 to read and write registers.

Set-up & Check

1. Power on Vcc and Vin to 5V.
2. Check P1V8 pin voltage is 1.8V +/- 5%.

Determine current I2C address & Programming times left

3. Find out what is the i2c address of the device.
   a) Send 7-bit i2c address from 08h to 0Bh and 0Dh to 77h until the device ACKs the address.
   b) Read register 16h (IC_DEVICE_ID) and check if the data is 63h, 64h, 65h, 66h, 67h or 68h (IR38163/165/263/265/363/365 respectively). If yes, the device is an OptiMOS™ IPOL device. Record this i2c address and go to step 4. If not, continue sending next address in step a).
4. To determine if there are any programming times left, use the i2c address to read MTP pointer register 99h (user section).
   - *Section_ptr=reg 99[3:0]h*
   - *If section_ptr>=8, quit*
   - *else if section_ptr= 15, new_section_ptr=0*
   - *else new_section_ptr=section_ptr+1*

Determine new I2C & PMBus™ addresses and Write new Configuration data into Working Register

5. Read the value of register 0x9B[3:0] which contains the detected address offset and save this for later use.
6. For PMBus™ capable devices, Read register 20h (PMBus™ address) in the Config file and write the config file data to register 20h only – note this may change the PMBus™ address
7. Read register 21h (i2c address) in the Config file and write the config file data to register 21h only – note this may change the i2c address!
8. Record the values sent to registers 20h and 21h which determine the new PMBus™ and i2C base addresses.
9. Since the new configuration may change the i2c register data at 21h, re-calculate the new i2c address as 21[6:0]h (base) + 9B[3:0]h (offset) (offset should be added only if MSB of register 21h is 0) and, for PMBus™ capable devices, the new PMBus™ address as 20[6:0]h (base) + 9B[3:0]h (offset) (offset should be added only if MSB of register 21h is 0)
10. Start using the new i2c and PMBus™ address for further read/write communications.
11. Read the entire config file and Write all USER section config file data to the device's operating register.

Write New Configuration into MTP Memory

12. Enable the MTP programming clock by setting register 8Eh to 01h.
13. Write a user section programming command based on Table 5 i.e set register B0h = (*40h + new_section_ptr*)
14. Wait 50ms.
15. Read register B0h and check if register B0h = 08h. If yes, the programming is successful; if not, time out if >200ms has elapsed otherwise go back to step 14. After 200ms, the programming is considered a failure.

Verification

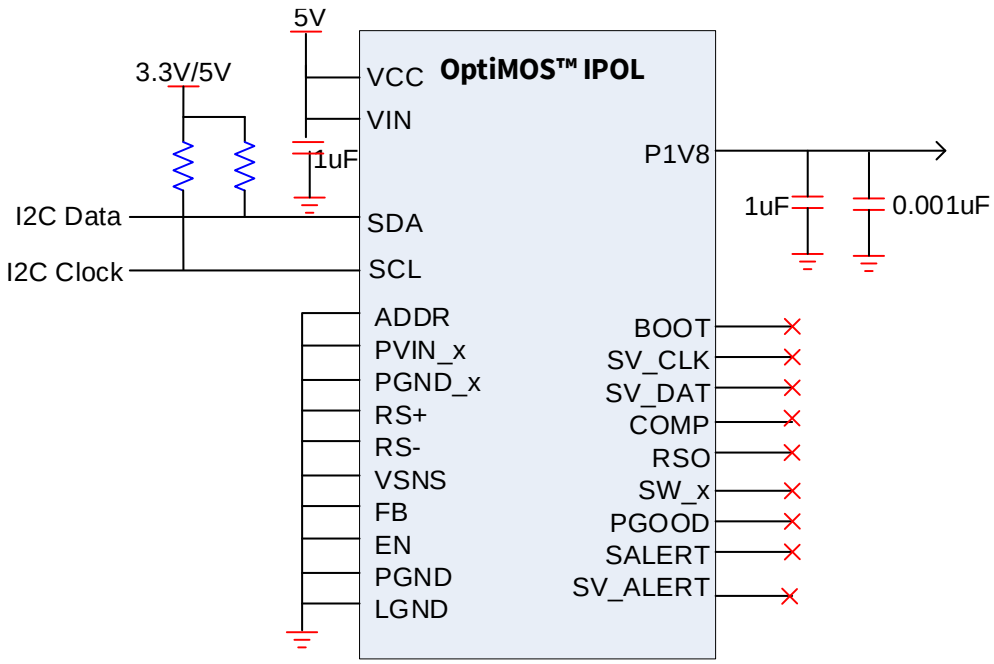
16. Remove the 5V power to clear the operating memory.
17. Re-apply 5V power. The device will initialize and transfer its NVM contents into the operating memory
18. For the User section (reg 20h-87h), read each register and compare contents with the Config file
19. If all register contents match and CRC error flag 0x97[2:0] = 000 – see Table 10. Then the programming is successful.

Note:

1. The quoted i2c address is 7-bit and becomes 8-bit by appending the R/W bit the LSB
2. Only the User section should be programmed. The Trim and MFR section should not be changed by customers.

I/O Terminations for a Programming Station

Figure 3: I/O Terminations



P1V8 = cap to GND (will self-bias to 1.8V)

Vin/Vcc = 5V. A better option is to separate Vin and Vcc and apply 12Vonly to Vin. Vcc (with 1uF cap to GND) will self-bias to ~5V.

I2C Read/Write Protocol

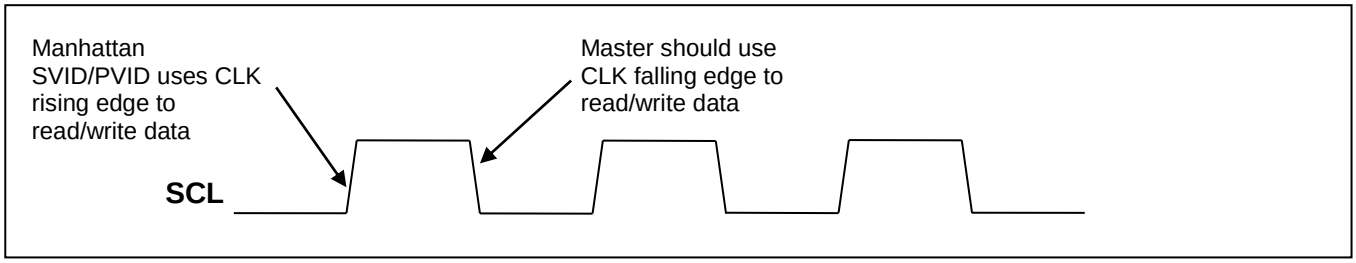
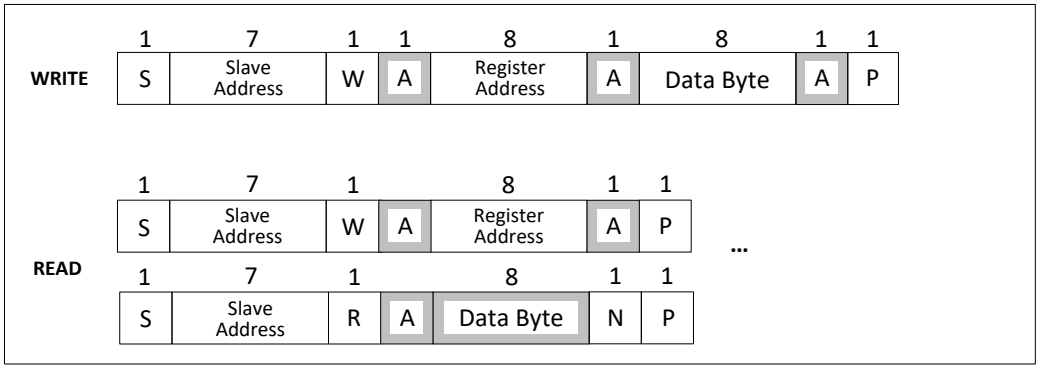


Figure 4: I2C Protocols to read or write a register



S: Start Condition
A: Acknowledge (0')
N: Not Acknowledge (1')
Sr: Repeated Start Condition
P: Stop Condition
R: Read (1')
W: Write (0')
PEC: Packet Error Checking
*: Present if PEC is enabled
: Master to Slave
: Slave to Master

PMBus™ Commands to Read/Write registers

Figure 5: PMBus™ Protocol to Write 1 byte to a register

*PEC is optional

Figure 6: PMBus™ Protocol to Read 2 bytes from a register

*PEC is optional

The first byte returned is the contents of the specified register address. The 2nd byte returned is the contents of register address + 1.

Appendix A: The Configuration File

The program that determines IR controller operation is called a 'configuration file' (also called 'Config File').
It is a three-column, space delimited text document as shown below.

```
//PROJECT NAME:POWERCENTER
20 C2 FF
21 12 FF
22 80 FF
23 1F FF
24 01 FF
……
```

The first column contains the register address, the second column contains the register data, and the third column contains the register mask. Each line of the file is provided in ascending order of the register address. Data is read line by line. For example, row 1 indicates that data-value C2h is targeted for register address 20h with a mask of FFh. As seen from this example, the register address and data fields are self-explanatory. However, the register mask column requires some explanation.

If the mask bit is zero, the corresponding register bit should not be verified after programming. If the mask bit is one, the corresponding bit must match the value from the config file after programming.

Published by
Infineon Technologies AG
81726 München, Germany

IMPORTANT NOTICE

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics ("Beschaffenheitsgarantie"). With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, 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.

In addition, any information given in this document is subject to customer's compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer's products and any use of the product of Infineon Technologies in customer's applications.

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.

Revision History

Version	ECN#	Reason for Change	Date	Modified by:	Approved by:
1.4		Updated for R3b & R4b silicon, removing the Programming Scale Factor section.	6/19/2019	William Gatune	Shihchin Lee
1.3		Add note on user section pointer is incorrect after the last image is programmed	7/12/2017	Shihchin Lee	D. Williams
1.2		Add Programming Scale Factor section	1/11/2017	Shihchin Lee	
1.0		Initial Release	3/30/2016	Shihchin Lee	