International
IOR Rectifier

APPLICATION NOTE

Application Note AN-1208

Software Optimization of IR3565B and IR36021 for Freescale T-series QorIQ Applications

By Ramesh Balasubramaniam

Table of Contents

	Page
Introduction	2
Controller I2C Address	2
VID Table	3
Optimizing the Core voltage (Loop 1) to meet +/-3% accuracy	3
Optimizing the DDR voltage (Loop 2)	4
Non-Volatile Memory	3
Optimizing the DDR voltage (Loop 2)	4

International Rectifier manufactures pre-configured versions of the IR356B and IR36021 digital voltage regulator controllers that are suitable for Freescale T-series QorIQ processors. This Application Note allows the user to further optimize the IR3565B and IR36021 to the customer application.

Introduction

IR3565B and IR36021 are advanced digital voltage regulator controllers that can be accessed and optimized via I2C registers. The key registers are listed in **table 1**. Communication between the Freescale T-series processor and the IR3565B/IR36021 is achieved using C functions (I2C_read, I2C_write) located in Freescale's U-boot code.

Table 1: I2C Registers

Register Address	Purpose	Comment
12h	I2C base Address	bit[7] - set to 1 to enable I2C communication Bits[6:0] - represents the 7-bit I2C address
17h	Set Boot Voltage Loop 1	Bits[7:0] - sets the boot voltage based on **table 4**
18h	Set Boot Voltage Loop 2	
6Ah	Set Voltage Loop 1	Bits[7:0] - sets the voltage after boot-up based on **table 4**
6Ch	Set Voltage Loop 2	
9Ah	Read Voltage Loop 1	Bits[7:0] - voltage readback where the LSB represents 1/128V
9Bh	Read Voltage Loop 2	
9Ch	Read Current Loop 1	Bits[7:0] - current readback where the LSB represents 2A
9Dh	Read Current Loop 2	Bits[7:0] - current readback where the LSB represents 0.5A
9Eh	Read Temperature 1	Bits[7:0] - temperature readback where the LSB represents 1°C
9Fh	Read Temperature 2	
A1h	Read Fault Status Loop 1	Bit[7] - Over-temperature Bit[6] - Over-current (fast) Bit[5] - Over-voltage Bit[4] - Under-voltage Bit[3] - 12V supply low Bit[2] - 3.3V supply low Bit[1] - Phase Fault Bit[0] - Over-current (slow)
A2h	Read Fault Status Loop 2	
A7h	NVM Counter	Bits[7:4] - undefined. Please ignore. Bits[3:0] - NVM Counter
D0h	NVM Control	Bits[7:4] – NVM Command (e.g. read or write) Bits[3:0] – data
71h	NVM Clock	Bits[7:0] – Turns on/off the clock for NVM programming

Controller I2C Address

The controller's I2C address is a combination of a "base" address + "offset". The base address is stored in a register inside the controller's Non-Volatile memory (NVM). The pre-programmed base addresses of the controllers that are specially pre-configured for Freescale T-series applications are listed in **table 2**. The offset is defined by a resistor at the ADDR_PROT pin (**table 3**). From tables 2 and 3, it can be seen that the I2C address is **38h** if the Freescale Reference Design is followed.

Table 2: Pre-configured Settings

Part	Pre-programmed		
	I2C base address	Boot Voltage Loop 1	Boot Voltage Loop 2
IR3565BFS01MTRP	36h	1.05V	1.5V (DDR3)
IR36021FS01MTRP	36h	1.035V	1.35V (DDR3L)
IR36021FS02MTRP	36h	1.045V	Disabled

Table 3: I2C Address Resistor Offset

Resistor	Offset
0.845kΩ	+0h
1.30kΩ	+1h
1.78kΩ*	+2h
2.32kΩ	+3h
2.87kΩ	+4h
3.48kΩ	+5h
4.12kΩ	+6h
4.75kΩ	+7h

Resistor	Offset
5.49kΩ	+8h
6.19kΩ	+9h
6.98kΩ	+Ah
7.87kΩ	+Bh
8.87kΩ	+Ch
10.00kΩ	+Dh
11.00kΩ	+Eh
12.10kΩ	+Fh

*** Used in T-series Reference Designs**

VID Table

The IR3565B/IR36021 contains an 8-bit Voltage Identifier (VID) table based on 5mV steps (**Table 4**), using the formula, ***Voltage = 0.245V + (VID x 5mV)***.

Table 4: VID Table

8-bit VID	Voltage
FFh	1.520V
FEh	1.515V
…	…
02h	0.255V
01h	0.250V

Optimizing the Core voltage (Loop 1) to meet +/-3% accuracy

To meet the voltage specifications, the optimum Core voltage value, burned into each Freescale T-series processor individually and located in its FUSE_R register, must be written to the IR3565B/IR36021 **after** the controller has powered up to its boot voltage.. At the time of writing, Freescale plans to implement this routine automatically within the next release of their SDK 1.7.

Alternatively, a simple software routine is easily written in the U-boot code to achieve the optimization:

1. Read the Freescale FUSE_SR register
2. Convert the 5 Freescale VID bits to the IR3565B/IR36021 8-bit VID format
3. Write the desired setpoint voltage to the IR3565B/IR36021 via I2C

Step 1: Read the Freescale FUSE_SR register

Consult Freescale documentation on how to access this register.

Step 2: Convert the 5 Freescale VID bits to the IR3565B/IR36021 8-bit VID format

Freescale's T4240EC_RevG documentation shows the mapping in **table 5**. Note that the Freescale VID table uses as 12.5mV increment whereas the IR3565B/IR36021 use a 5mV increment, thus there will not be an exact match for some VID codes.

Table 5: Freescale T4240EC_RevG VID Table

Freescale Fuse Status Register (DCFG_CCSR_FUSESR),T4240 Datasheet RevG.		Closest IR 8-bit VID Code
Binary value of DA_V / DA_ALT_V	Desired Core voltage (V)	
00000b	1.0500V	A1h (1.050V)
00001b	0.9875V	94h (0.985V)
00010b	0.9750V	92h (0.975V)
00011b	0.9625V	90h (0.965V)
00100b	0.9500V	8Dh (0.950V)
00101b	0.9375V	8A (0.935V)
00110b	0.9250V	88h(0.925V)
00111b	0.9125V	86h (0.915V)
01000	0.9000V	83h (0.900V)
10000	1.0000V	97h (1.000V)
10001	1.0125V	9Ah (1.015V)
10010	1.0250V	9Ch (1.025V)
10011	1.0375V	9Eh (1.035V)
10100	1.0500V	A1h (1.050V)

Step 3: Write the desired setpoint voltage to the IR3565B/IR36021 via I2C

Use Freescale's U-boot I2C_write function (or other method) to send the correct VID code to the IR3565B/IR36021. For example, to set 0.900V as the Core output voltage that is powered by Loop 1 of the IR3565B/IR3621 residing at I2C address 38h:

I2C_write(38h, 6Ah, 83h)
Where 38h = I2C address of the IR3565B/IR36021
6Ah = register address for "Set Voltage Loop 1"
83h = 0.900V

Optimizing the DDR Voltage (Loop 2)

The IR3565B/IR36021 controllers are pre-configured to boot up for specific DDR voltages per **table 2**. If a different DDR voltage is required, simply write the appropriate VID code to the "set voltage Loop 2" register **after** boot-up. e.g. to configure for DDR4 (1.2V), use:

I2C_write(38h, 6Ch, 83h)
Where 38h = I2C address of the IR3565B/IR36021
6Ch = register address for "Set Voltage Loop 2"
BFh = 1.200V

Alternatively, the DDR voltages can be set **before** boot-up using the "Boot Voltage Loop 2" register. The advantage of this method is that the DDR voltage is set correctly at power-up. The disadvantage is that the command must be pre-programmed (burnt into

NVM) or the command must be sent before the IR3565B/IR36021 is enabled. Generally, this can only be achieved using an In-Circuit Test (ICT) system and not through Freescale's U-boot code.

Table 6 lists the VID codes for some of the common DDR voltages.

Table 6: DDR Voltage VID codes

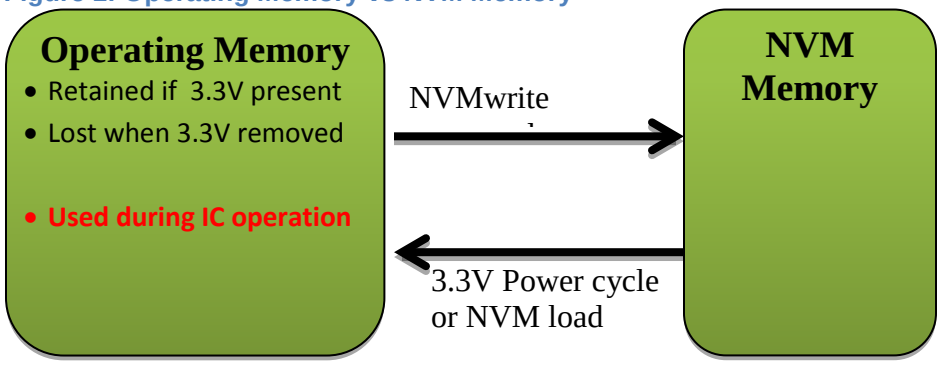
DDR Standard	DDR Voltage	IR 8-bit VID code
DDR3	1.5V	FBh
DDR3L	1.35V	DDh
DDR4	1.2V	BFh

Non-Volatile Memory (NVM)

Any I2C commands written to the IR3565B/IR36021 are retained while the 3.3V supply is maintained and are lost if the 3.3V supply is removed. To make a permanent change, the I2C commands in the (volatile) operating registers must be written to the NVM (**figure 1**). Note that the IR3565B only allows a maximum of 9 writes and typically the pre-configured parts in **table 2** have already used as least 1 write (1 write consists of writing all registers at once). Writing the NVM should follow this procedure:

1. Load the operating registers with the required values
2. Check if there are NVM writes are left
3. Send the command to program the NVM
4. Wait until programming is complete
5. Verify that the correct data is retained in the NVM

Figure 1: Operating Memory vs NVM Memory



Example

The example below shows how to program the Loop 2 Boot Voltage to 1.2V (DDR4) for an IR3565B/IR36021 with I2C address 38h

Step 1: Load the operating registers with the required values

Load the registers with the required values to set up the Loop 2 boot voltage for 1.2V (DDR4).

I2C_write(38h, 6Ch, 83h)

Step 2: Check if there are NVM writes are left

Read the counter value at address A7h[3:0] and use **table 7** to determine if any NVM writes are left and what the next NVM bank to write is.

```
Value = I2C_read(38h, A7h) AND 0Fh          /*mask out the upper 4 bits*/
IF (Value= 15 OR Value<8) THEN  CONTINUE    /*NVM writes left */
                          ELSE STOP         /* No writes left! */
Next_NVM_Bank = value+1
```

Table 7: NVM Counter

A7h[3:0] Value	NVM Writes Remaining	Next NVM Bank
15d	9	0
0d	8	1
1d	7	2
2d	6	3
3d	5	4
4d	4	5
5d	3	6
6d	2	7
7d	1	8
8d – 14d	0	none left

Step 3: Send the command to program the NVM

First, the NVM clock must be turned on by writing data 04h to register 71h. Then, Read and Write operations on the NVM are controlled by the data in register D0h (**table 8**).

Table 8: NVM control

I2C Operation	Register D0h bit Values								
	7	6	5	4	3	2	1	0	Meaning
Write	0	1	0	0	Next NVM Bank				Write ALL Operating Registers data to NVM
Write	0	0	1	0	0	0	0	0	Load NVM to Operating Registers
Read	0	0	0	Don't care					Idle state (operation complete)

```
I2C_write(38h, 71h, 04h)            /* turn on the NVM clock */
Value = 40h + Next_NVM_Bank         /* form the databyte to command a Write to the
Next NVM Bank */
I2C_write (38h, D0h, Value)
```

Step 4: Wait until programming is complete

The NVM typically takes about 30ms to complete writing, thus it is good practice to implement a timeout after about 300ms if the completion signal is not received:

```
Time=0ms
Label Wait_State
   Wait 10ms
   IF Time=300ms THEN FAIL                          /* Programming fail */
   NVM_State = I2C_read(38h, D0) AND E0             /* mask out lower bits to
determine NVM state */
   IF NVM_State= 0 THEN Exit                        /* Programming complete */
                  ELSE Time=Time+10ms;  GOTO Wait_State
```

Step 5: Verify that the correct data is retained in the NVM

Firstly power cycle the 3.3V supply to the IR3565B/IR36021 to clear the operating registers and force an automatic reload of NVM into the operating registers. Then read back the register(s) of interest and compare with the desired value:

```
Turn off 3.3V Supply              /* clears operating memory */
Turn on 3.3V supply               /* loads NVM into the operating memory */
Value = I2C_read(38h, 6Ch)        /* read the Loop 2 Boot Voltage register that was
burnt into NVM */
IF Value= 83h THEN CONTINUE       /* Loop 2 Boot Voltage was correctly
programmed to 1.2V in NVM */
            ELSE FAIL             /*Loop 2 Boot Voltage NVM programming failed */
```