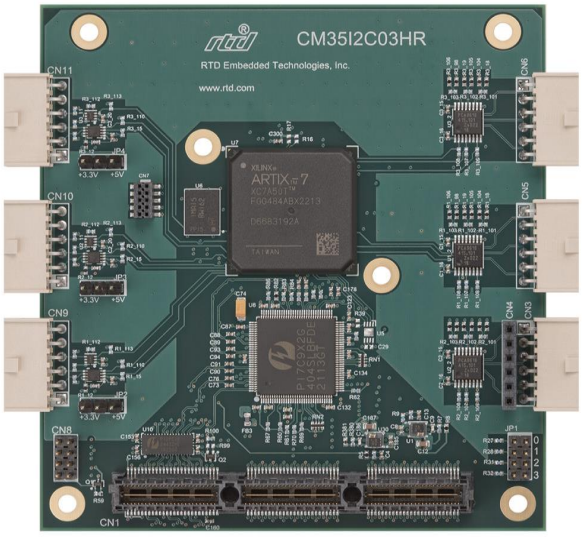
CM35I2C03HR

I2C Communication Module

User's Manual

BDM-610020163 Rev. A



RTD Embedded Technologies, Inc.
AS9100 and ISO 9001 Certified

RTD Embedded Technologies, Inc.
103 Innovation Boulevard
State College, PA 16803 USA
Telephone: 814-234-8087
Fax: 814-234-5218

www.rtd.com

sales@rtd.com
techsupport@rtd.com

PC/104

DESIGNED AND MADE IN THE UNITED STATES

AS9100 - ISO 9001 CERTIFIED

Revision History

Rev A	Initial Release

Advanced Analog I/O, Advanced Digital I/O, aAIO, aDIO, a2DIO, Autonomous SmartCal, "Catch the Express", cpuModule, dspFramework, dspModule, eBuild, expressMate, ExpressPlatform, "MIL Value for COTS prices", multiPort, PlatformBus, and PC/104EZ are trademarks, and "Accessing the Analog World", dataModule, IDAN, HiDAN, HiDANplus, RTD, the RTD logo, and StackNET are registered trademarks of RTD Embedded Technologies, Inc. (formerly Real Time Devices, Inc.). PS/2 is a trademark of International Business Machines Inc. PCI, PCI Express, and PCIe are trademarks of PCI-SIG. PC/104, PC/104-Plus, PCI-104, PCIe/104, PCI/104-Express and 104 are trademarks of the PC/104 Consortium. All other trademarks appearing in this document are the property of their respective owners.

Failure to follow the instructions found in this manual may result in damage to the product described in this manual, or other components of the system. The procedure set forth in this manual shall only be performed by persons qualified to service electronic equipment. Contents and specifications within this manual are given without warranty and are subject to change without notice. RTD Embedded Technologies, Inc. shall not be liable for errors or omissions in this manual, or for any loss, damage, or injury in connection with the use of this manual.

Copyright © 2024 by RTD Embedded Technologies, Inc. All rights reserved.

Table of Contents

1 Introduction **7**

1.1 Product Overview 7

1.2 Board Features 7

1.3 Ordering Information 7

1.4 Contact Information 8
- 1.4.1 Sales Support 8
- 1.4.2 Technical Support 8

2 Specifications **9**

2.1 Operating Conditions 9

2.2 Electrical Characteristics 9

2.3 Functional Characteristics 10

3 Board Connection **11**

3.1 Board Handling Precautions 11

3.2 Physical Characteristics 11

3.3 Connectors and Jumpers 12
- 3.3.1 External I/O Connectors 12
  - CN3/CN5/CN6: Differential I2C Connector 12
  - CN4: Stacking I2C Connector 13
  - CN9/CN10/CN11: I2C Connector 13
- 3.3.2 Bus Connectors 13
  - CN1(Top) & CN2(Bottom): PCIe Connector 13
- 3.3.3 Jumpers 14
  - JP1: User ID Jumper 14
  - JP2/JP3/JP4: Voltage Select Jumper 14

3.4 Steps for Installing 15

4 Functional Description **16**

4.1 Block Diagram 16

4.2 Overview 16
- 4.2.1 I²C Host State Machine 16
  - Building I²C Packets 16
  - I2C Read Process 18
  - I2C Write Process 18
- 4.2.2 NXP PCA9616 Driver information 19
  - NXP PCA9616 Buffer 19
  - Slave Termination Circuit 19
- 4.2.3 Differential Cabling 20

5 Register Address Space **21**
- Register Types 21
- Function Block Mapping 21

5.1 BAR0: General Board Control 22
- 5.1.1 GBC_FMT (Read-Only) 22
- 5.1.2 GBC_REV (Read-Only) 22
- 5.1.3 GBC_BRD_RST (Read/Write) 22
- 5.1.4 GBC_PDP (Read-Only) 22
- 5.1.5 GBC_BUILD (Read-Only) 22
- 5.1.6 GBC_SYS_CLK_FREQ (Read Only) 23
- 5.1.7 GBC_USER_ID (Read Only) 23
- 5.1.8 GBC_IRQ_STATUS (Read/Clear) 23
- 5.1.9 GBC_DIRQ_STATUS (Read/Clear) 23
- 5.1.10 GBC_EOI (Read/Clear) 23
- 5.1.11 FBn_ID (Read-Only) 23

5.1.12 FBn_Offset (Read-Only) 23
5.1.13 FBn_Offset_DMA (Read-Only) 23
5.2 BAR2: Functional Block Standard DMA 24
5.2.1 FB_DMAm_Action (Read/Write) 24
5.2.2 FB_DMAm_Last_Action (Read/Write) 24
5.2.3 FB_DMAm_Setup (Read/Write) 24
5.2.4 FB_DMAm_Stat_Used (Read/Write) 25
5.2.5 FB_DMAm_Stat_Invalid (Read/Write) 25
5.2.6 FB_DMAm_Stat_Overflow (Read/Write) 25
5.2.7 FB_DMAm_Stat_Underflow (Read/Write) 25
5.2.8 FB_DMAm_Stat_Complete (Read/Write) 25
5.2.9 FB_DMAm_Current_Buffer (Read-Only) 25
5.2.10 FB_DMAm_COUNT (Read-Only) 25
5.2.11 FB_DMAm_RD_FIFO_CNT(Read-Only) 25
5.2.12 FB_DMAm_WR_FIFO_CNT(Read-Only) 25
5.2.13 FB_DMAm_ADDRESSn (Read/Write) 25
5.2.14 FB_DMAm_SIZEn (Read/Write) 25
5.2.15 FB_DMAm_CTRLn (Read/Write) 26
5.2.16 FB_DMAm_STATn (Read/Clear) 26
5.3 BAR2: I2C Host Controller 27
5.3.1 Function Block Register Map 27
5.3.2 I2C_BUS_CTRL (Read/Write) 27
5.3.3 I2C_CONFIG (Read/Write) 27
5.3.4 FB_RST (Read/Write) 27
5.3.5 CLK_DIV (Read/Write) 27
5.3.6 INT_ENA (MASKABLE READ/WRITE) 27
5.3.7 INT_STAT (Read/Write) 27
5.3.8 WRITE_FIFO (Read/Write) 28
5.3.9 WRITE_FIFO _COUNT (Read-Only) 28
5.3.10 READ_FIFO (Read-Only) 28
5.3.11 READ_FIFO_COUNT (Read-Only) 28
5.4 BAR2: Flash Programmer (Reserved) 29
5.4.1 SERIAL_NUMBER (Read-Only) 29
6 Troubleshooting 30
7 Additional Information 31
7.1 PC/104 Specifications 31
7.2 PCI and PCI Express Specification 31
8 Limited Warranty 32

Table of Figures

Figure 1: Board Dimensions 11
Figure 2: Board Connections 12
Figure 3: CN3 / CN5 / CN6 Cabled I2C Connector 12
Figure 4: CN4 Stacking I2C Connector 13
Figure 5: CN9 / CN10 / CN11 Cabled I2C Connector 13
Figure 6: Example 104™Stack 15
Figure 7: CM35I2C03HR Block Diagram 16
Figure 8: I2C Read Flow chart 18
Figure 9: I2C Write Flow chart 18
Figure 10: NXP PCA9616 Example Circuit 19
Figure 11: PCA9616 Termination 19

Table of Tables

Table 1: Ordering Options 7
Table 2: Operating Conditions 9
Table 3: Electrical Characteristics 9
Table 4: Functional Characteristics 10
Table 5 CN4 Cabled I2C Connector Pin Assignments 12
Table 6: CN9/CN10/CN11 Cabled I2C Connector Pin Assignments 13
Table 7 JP1- User ID Jumper (default open) 14
Table 8 JP2/JP3/JP4 –Voltage Select Jumper 14
Table 9: Function Block Mapping 21
Table 10: Base Functional Block 22
Table 11: DMA Registers 24
Table 12: I2C Host Controller Functional Block 27
Table 13: Flash Programmer Functional Block 29

1 Introduction

1.1 Product Overview

The CM35I2C03HR is an I2C interface module that provides six independent host interfaces in a PCIe/104 format. Two types of output buffers are provided by this module: one for standard I2C bus and another for differential I2C bus. The differential ports feature an NXP PCA9616 Fast-mode Plus (Fm+) buffer which enables I2C cabling over 3-meters at rates up to 1Mhz. The standard ports feature a TI TCA39306 buffer, which allows voltage translation between the host controller and attached devices. Each standard port has built-in pull-up resistors and selectable 3.3V/5V bus voltage.

The CM35I2C03HR provides a basic memory-mapped interface that allows users to send and receive I2C data. Because I2C's slower speed relative to PCIe, the CM35I2C03HR provides zero latency and no delay between back-to-back transactions. RTD provides the user with operating system drivers and example programs that demonstrate the module's capability.

1.2 Board Features

- PC/104 form factor
- PCIe/104 stackable bus structure
  - PCIe universal expansion bus (Type 1 or Type 2)
  - PCIe x1 Interface
- PCIe link repopulation allowing increased PCI Express stack capacity
- Six independent I2C host interfaces
  - Supports 8-Bit Addressing
  - Supports Clock Stretching
  - 4K Send FIFO
  - 4K Send FIFO
  - Configurable I2C bus speeds up to 1Mhz
  - Three differential I2C Buses using NXP PCA9616
    - Support 1Mhz over 3 meters
    - Onboard host termination
  - Three standard I2C Buses using TI TCA39306
    - Selectable 3.3V or 5V bus voltage
    - Onboard 2Kohm pull-up resistors
    - Support for Fast-mode plus/Fast-mode/standard mode clock rates
  - Requires only +5 VDC for operation

1.3 Ordering Information

The CM35I2C03HR is available with the following options:

Table 1: Ordering Options

Part Number	Description
CM35I2C03HR	PCIe/104 I2C Communication Module
CM35I2C03HR-SS	PCIe/104 I2C Communication Module with stack through connector

The Intelligent Data Acquisition Node (IDAN®) building block can be used in just about any combination with other IDAN building blocks to create a simple but rugged 104™ stack. This module can also be incorporated in a custom-built RTD HiDAN® or HiDAN*plus*® High Reliability Intelligent Data Acquisition Node. Contact RTD sales for more information on our high reliability systems.

1.4 Contact Information

1.4.1 Sales Support

For sales inquiries, you can contact RTD Embedded Technologies sales via the following methods:

Phone: 1-814-234-8087 Monday through Friday, 8:00am to 5:00pm (EST).
E-Mail: sales@rtd.com

1.4.2 Technical Support

If you are having problems with your system, please try the steps in the Troubleshooting section of this manual on page 30.

For help with this product, or any other product made by RTD, you can contact RTD Embedded Technologies technical support via the following methods:

Phone: 1-814-234-8087 Monday through Friday, 8:00am to 5:00pm (EST).
E-Mail: techsupport@rtd.com

2 Specifications

2.1 Operating Conditions

Table 2: Operating Conditions

Symbol	Parameter	Test Condition	Min	Max	Unit
V_{cc5}	5V Supply Voltage		4.75	5.25	V
V_{cc3}	3.3V Supply Voltage		n/a	n/a	V
V_{cc12}	12V Supply Voltage		n/a	n/a	V
V_{cc-12}	-12V Supply Voltage		n/a	n/a	V
T_a	Operating Temperature		-40	+85	C
T_s	Storage Temperature		-55	+125	C
RH	Relative Humidity	Non-Condensing	0	90%	%
MTBF	Mean Time Before Failure	Telcordia Issue 2 30°C, Ground benign, controlled		TBD	Hours

2.2 Electrical Characteristics

Table 3: Electrical Characteristics

Symbol	Parameter	Test Condition	Min	Typ	Max	Unit
P	Power Consumption	V_{cc5} = 5.0V			TBD	W
I_{cc}	5V Input Supply Current	Active			TBD	mA
PCIe/104 Bus						
	Differential Output Voltage		0.8		1.2	V
	DC Differential TX Impedance		95.2		116.9	Ω
	Differential Input Voltage		0.175		3.3	V
	DC Differential RX Impedance		92.7		115.8	Ω
	Electrical Idle Detect Threshold		61		173	mV
FM+ I2C (CN9/10/11)						
V_{DD}	I2C bias voltage	Set by Jumper	3.3		5	V
F_{SCL}	Clock Frequency	Limited by bus capacitance			1000	KHz
	Bus capacitance limit	Standard/Fast Modes			400	pF
		Fast Mode Plus			550	pF
V_{IL}	LOW-level Input Voltage		-0.3		0.63	V
V_{IH}	HIGH-level Input Voltage		0.7*V_{DD}		V_{DD}	V
R_{PU}	Onboard Pullup Resistors				2	Kohm
Differential I2C (CN3/4/5/6)						
V_{DD}	Differential I2C bias voltage				5	V
V_{cm}	Common-mode voltage				5	V
F_{SCL}	Clock Frequency	Limited by bus capacitance			1000	KHz
I_{IL}	LOW-level input current		-0.5		0.4	V
V_{IH}	HIGH-level Input Voltage		0.7*V_{DD}		5.5	V
$V_{th(dif)}$	differential receiver threshold voltage	0V≤V_{cm}≤V_{DD}	-200		200	mV
$V_{I(hys)}$	hysteresis of input voltage	receiver; 0V≤V_{cm}≤V_{DD}		30		mV
$V_{o(dif)(p-p)}$	peak-to-peak differential output voltage		-5.0	-1.5	-1.0	V
R_{PU}	Pullup Resistors	pull-up resistor on DSCLM and DSDAM connected to $V_{DD(B)}$ rail		604		ohm
R_{PD}	Pulldown Resistors	pull-down resistor on DSCLP and DSDAP connected to V_{SS} rail		604		ohm

2.3 Functional Characteristics

Table 4: Functional Characteristics

Symbol	Parameter	Value	Unit
GBC_SYS_CLK_FREQ	System Clock Frequency	50	MHz
	I2C Write FIFO Size	4096	D-Words
	I2C Read FIFO Size	4096	D-Words

3 Board Connection

3.1 Board Handling Precautions

To prevent damage due to Electrostatic Discharge (ESD), keep your board in its antistatic bag until you are ready to install it into your system. When removing it from the bag, hold the board at the edges, and do not touch the components or connectors. Handle the board in an antistatic environment, and use a grounded workbench for testing and handling of your hardware.

3.2 Physical Characteristics

STEP model is available upon request; contact RTD Tech Support for more information.

- Weight: Approximately 0.12 lbs. (55 g)
- Dimensions: 3.550 in. L x 3.775 in. W (90.17 mm L x 95.89 mm W)

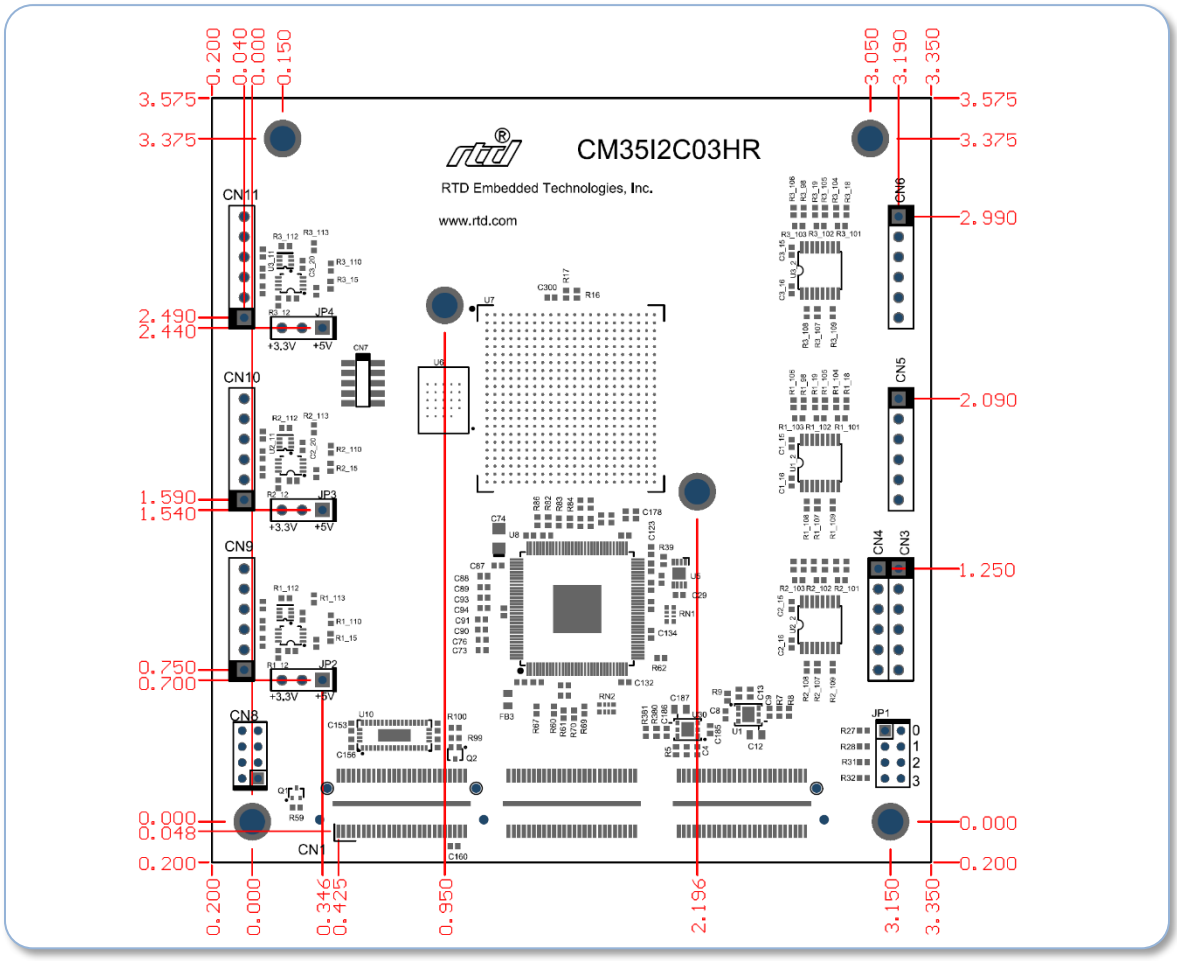


Figure 1: Board Dimensions

3.3 Connectors and Jumpers

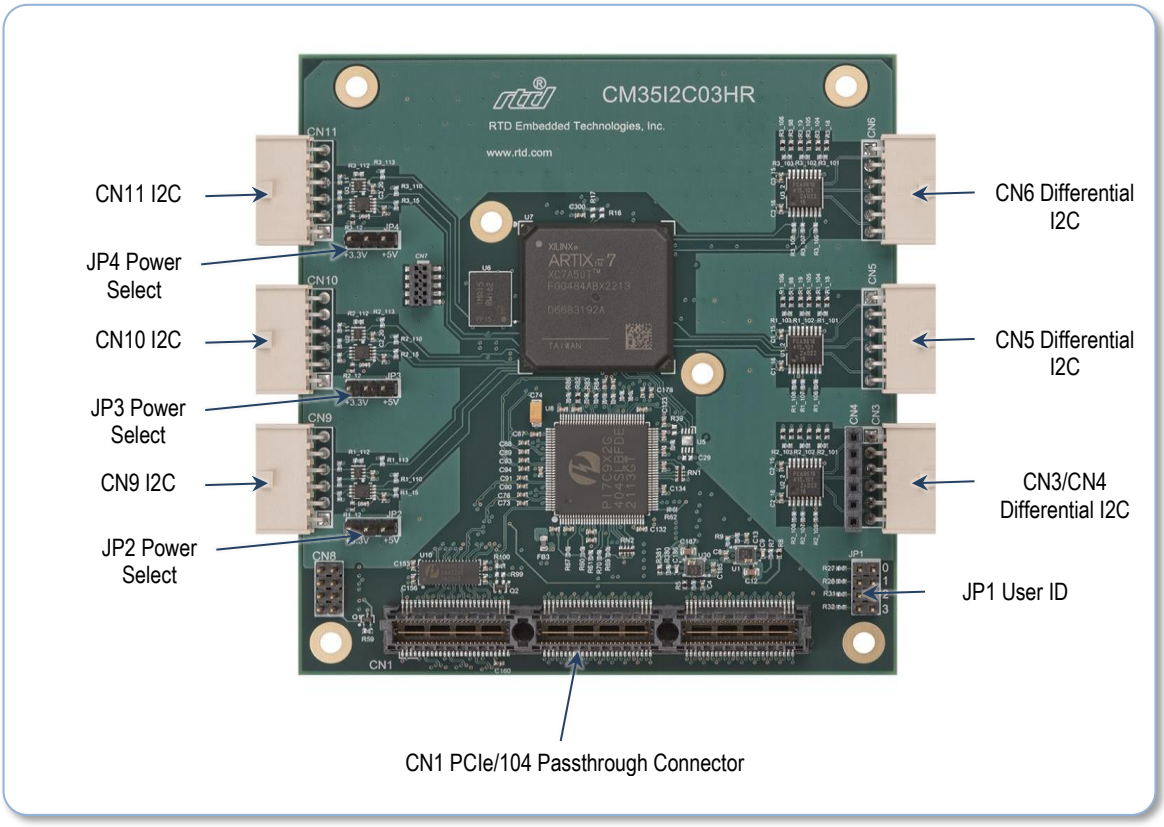


Figure 2: Board Connections

3.3.1 External I/O Connectors

CN3/CN5/CN6: Differential I2C Connector

CN3/CN5/CN6 is a 0.1" 6-pin right angle cabled I2C connector. An example of its' mating connector is the Samtec IPD1-06-S-K. Pin 1 is indicated by a square pad on both the top and bottom of the board.

Figure 3: CN3 / CN5 / CN6 Cabled I2C Connector

1	2	3	4	5	6
I2C_INT_P	I2C_INT_N	I2C_CLK_N	I2C_CLK_P	I2C_DATA_P	I2C_DATA_N

Table 5 CN4 Cabled I2C Connector Pin Assignments

CN4: Stacking I2C Connector

CN4 is a 6-pin vertical 0.1" I2C connector. An example of its' mating connector is the Samtec HTSW-106-05-F-S header. Pin 1 is indicated by a square pad on both the top and bottom of the board. This connector is shares the same I2C interface with CN3.

The CM35I2C03HR-SS version of board uses a stack through connector. This allow the CM35I2C03HR to be stack above or below the DM35MRM128HR.

Warning: Due to the length of pins, the CM35I2C03HR-SS should only be ordered if used with DM35MRM128HR.

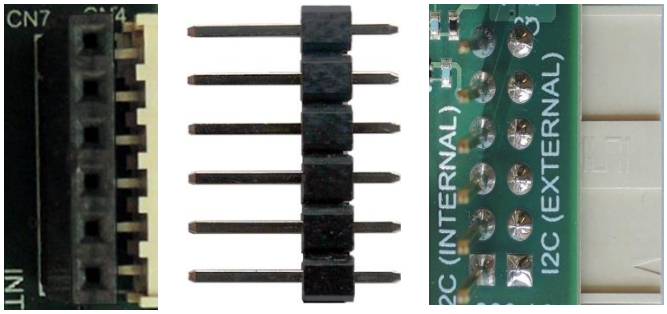


Figure 4: CN4 Stacking I2C Connector

CN9/CN10/CN11: I2C Connector

CN9/CN10/CN11 is a 0.1" 6-pin right angle cabled I2C connector. An example of its' mating connector is the Samtec IPD1-06-S-K. Pin 1 is indicated by a square pad on both the top and bottom of the board.

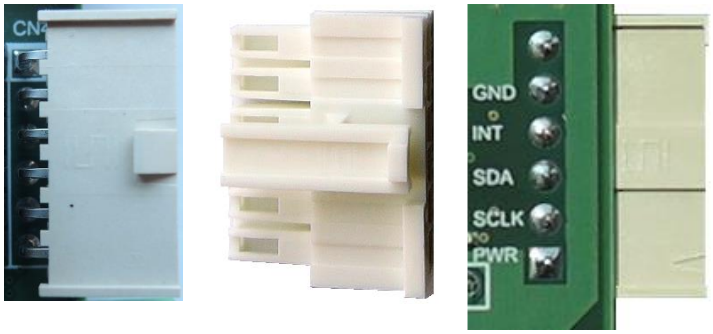


Figure 5: CN9 / CN10 / CN11 Cabled I2C Connector

1	2	3	4	5	6
POWER	I2C_CLOCK	I2C_DATA	I2C_INT	GND	GND

Table 6: CN9/CN10/CN11 Cabled I2C Connector Pin Assignments

3.3.2 BUS CONNECTORS

CN1(Top) & CN2(Bottom): PCIe Connector

The PCIe connector is the connection to the system CPU. The position and pin assignments are compliant with the *PCI/104-Express Specification*. (See PC/104 Specifications on page 31)

The CM35I2C03HR is a “Universal” board and can connect to either a Type 1 or Type 2 PCIe/104 connector.

3.3.3 JUMPERS

JP1: User ID Jumper

The User ID Jumper is a four position, user defined jumper block. The jumpers can be read by the FPGA. An installed jumper results in a logic low, and an open jumper results in a logic high. The function of the ID jumpers is to provide a mechanism for distinguishing between multiple identical Device IDs in PCI space. Since PCI enumeration is based on bus architecture and is not necessarily intuitive in order of operation, the User ID Jumper would be a simple way to be sure which Device ID belongs to which board.

Table 7 JP1- User ID Jumper (default open)

Position	Description
1-2	User ID bit 0
3-4	User ID bit 1
5-6	User ID bit 2
7-8	User ID bit 3

JP2/JP3/JP4: Voltage Select Jumper

This jumper is used to set the voltage level of the I2C Interface. This rail powers the user side of the I2C and onboard pull up resistors. The user can select to power interface from either 3.3V or 5V.

Position	Description
1-2	5V
2-3	3.3V (Default)

Table 8 JP2/JP3/JP4 –Voltage Select Jumper

3.4 Steps for Installing

1. Always work at an ESD protected workstation and wear a grounded wrist-strap.
2. Turn off power to the PC/104 system or stack.
3. Select and install stand-offs to properly position the module on the stack.
4. Remove the module from its anti-static bag.
5. Check that pins of the bus connector are properly positioned.
6. Check the stacking order; make sure all of the busses used by the peripheral cards are connected to the cpuModule.
7. Hold the module by its edges and orient it so the bus connector pins line up with the matching connector on the stack.
8. Gently and evenly press the module onto the PC/104 stack.
9. If any boards are to be stacked above this module, install them.
10. Attach any necessary cables to the PC/104 stack.
11. Re-connect the power cord and apply power to the stack.
12. Boot the system and verify that all of the hardware is working properly.

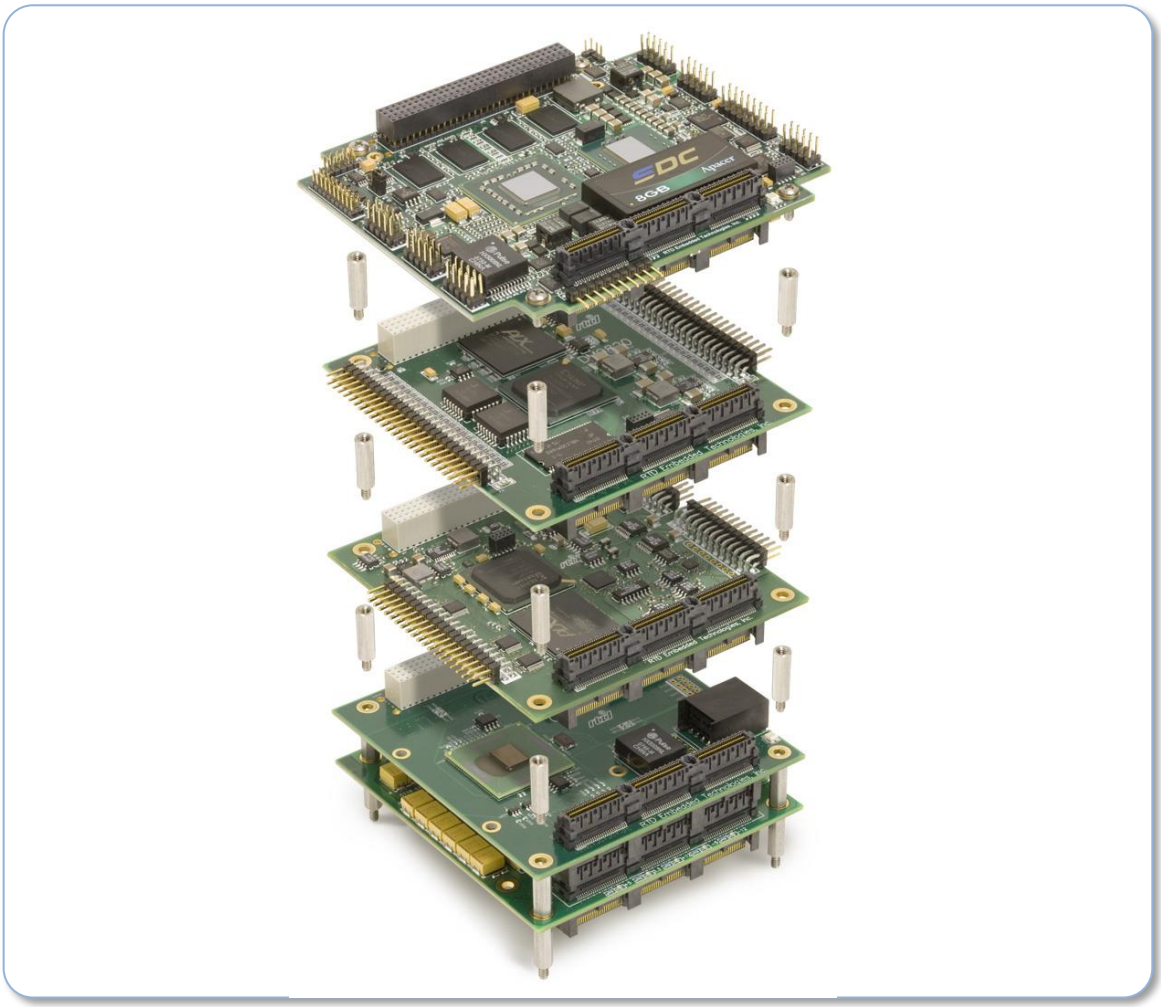


Figure 6: Example 104™ Stack

4 Functional Description

4.1 Block Diagram

The Figure below shows the functional block diagram of the CM35I2C03HR. The various parts of the block diagram are discussed in the following sections.

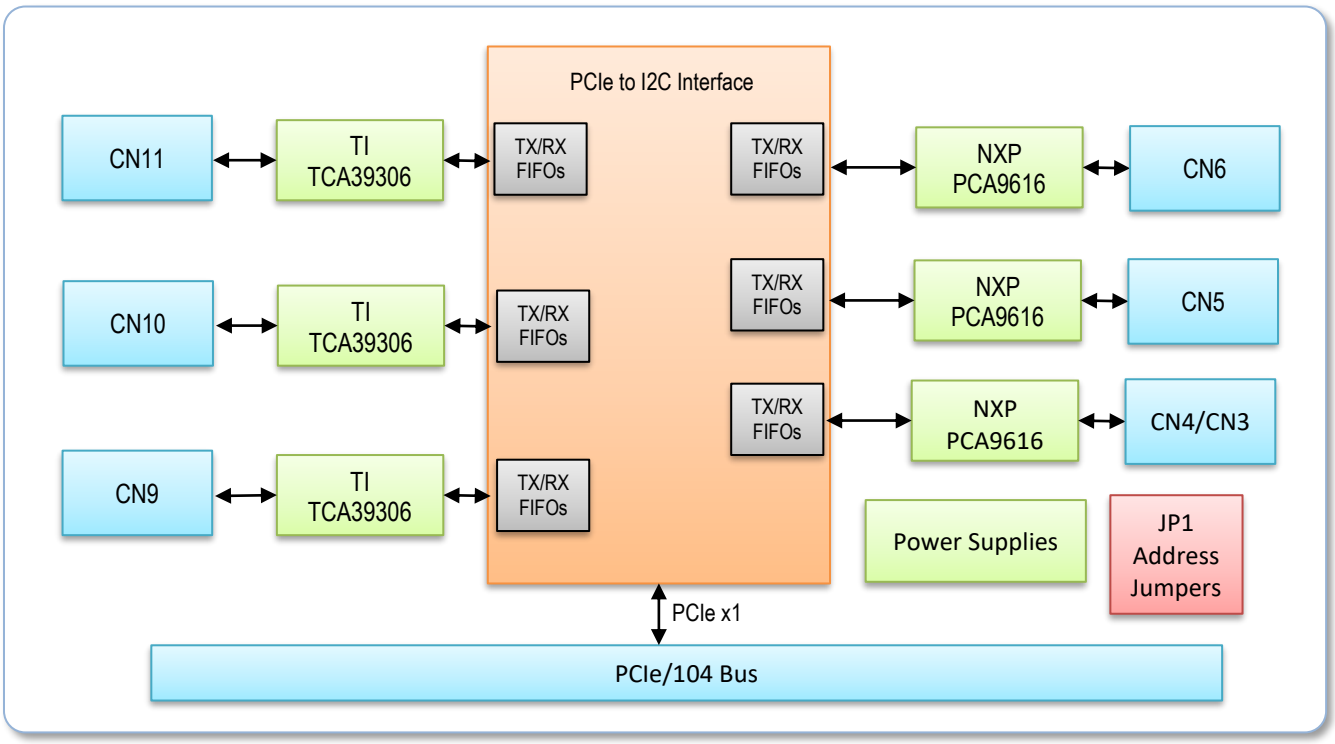


Figure 7: CM35I2C03HR Block Diagram

4.2 Overview

The CM35I2C03HR provides PCIe memory mapped interface that can control up to six independently I²C host interface. Each interface provide has an adjustable clock, TX FIFO, and RX FIFO. The FIFOs are used to send and receive data from I²C devices. The I²C interface handles the state machine logic need to send and receive I²C data.

4.2.1 I²C Host State Machine

This section will show the steps in I²C host state machine. The state machine automates the I²C transaction, so the user only needs to build a packet of bytes that need to be sent to I²C devices and set the GO bit.

Building I²C Packets

The CM35I2C03HR module using a memory mapped register to write I²C commands/data into a FIFO that will be sent to across the I²C bus to the devices. Below provides basic examples of how to place data into the FIFO.

Standard write I²C device

1. Write the new_packet_flag/device address/R_W=0 to the WRITE_FIFO (Example: 0x180)
2. Write the device register that needs to be access to the WRITE_FIFO (Example: 0x00)
3. Write the first byte that will be sent to I²C device to the WRITE_FIFO (Example: 0x55)
4. Continue to write additional bytes to be sent to the I²C device to the WRITE_FIFO (Example: 0xAA)
5. Set the B[0] of I2C_BUS_CTRL to start the I²C transaction.
6. Read B[0] of I2C_BUS_CTRL to verify I²C transaction as completed.

7. Read INT_STAT to verify no errors have occurred

Back-to-Back Writes

1. Write the new_packet_flag/device address/R_W=0 to the WRITE_FIFO (Example: 0x180)
2. Write the device register that needs to be access to the WRITE_FIFO (Example: 0x00)
3. Write the first byte that will be sent to I²C device to the WRITE_FIFO (Example: 0x55)
4. Continue to write additional bytes to be sent to the I²C device to the WRITE_FIFO (Example: 0xAA)
5. Write the new_packet_flag/device address/R_W=0 to the WRITE_FIFO (Example: 0x182)
6. Write the device register that needs to be access to the WRITE_FIFO (Example: 0x45)
7. Write the first byte that will be sent to I²C device to the WRITE_FIFO (Example: 0x55)
8. Continue to write additional bytes to be sent to the I²C device to the WRITE_FIFO (Example: 0xAA)
9. Set the B[0] of I2C_BUS_CTRL to start the I²C transaction.
10. Read B[0] of I2C_BUS_CTRL to verify I²C transaction as completed.
11. Read INT_STAT to verify no errors have occurred

Standard read I²C device

1. Write the new_packet_flag/device address/R_W=0 to the WRITE_FIFO (Example: 0x180)
2. Write the device register that needs to be access to the WRITE_FIFO (Example: 0x00)
3. Set the B[0] of I2C_BUS_CTRL to start the I²C transaction.
4. Read B[0] of I2C_BUS_CTRL to verify I²C transaction as completed.
5. Read INT_STAT to verify no errors have occurred
6. Write the new_packet_flag/device address/R_W=1 to the WRITE_FIFO (Example: 0x181)
7. Write number bytes to be read from the I²C device to the WRITE_FIFO (Example: 0x03)
8. Set the B[0] of I2C_BUS_CTRL to start the I²C transaction.
9. Read B[0] of I2C_BUS_CTRL to verify I²C transaction as completed.
10. Read INT_STAT to verify no errors have occurred
11. Read I²C data from READ_FIFO

Read I²C device without STOP

1. Write the new_packet_flag/device address/R_W=0 to the WRITE_FIFO (Example: 0x180)
2. Write the device register that needs to be access to the WRITE_FIFO (Example: 0x00)
3. Write the new_packet_flag/device address/R_W=1 to the WRITE_FIFO (Example: 0x181)
4. Write number bytes to be read from the I²C device to the WRITE_FIFO (Example: 0x03)
5. Set the B[0] of I2C_BUS_CTRL to start the I²C transaction.
6. Read B[0] of I2C_BUS_CTRL to verify I²C transaction as completed.
7. Read INT_STAT to verify no errors have occurred
8. Read I²C data from READ_FIFO

I2C Read Process

Figure 8 below is a flowchart to that demonstrates how the CM35I2C03HR handles the I²C read command. The CM35I2C03HR will handle the ACK/NACK handshaking. Once the user set the GO/BUSY bit, this bit will be held low until the bus returns to idle state. Once GO/BUSY is high, the user should check to see if any status bits have been set to indicate an error has occurred. If there are no errors, than the READ FIFO will hold data captured from the I²C device.

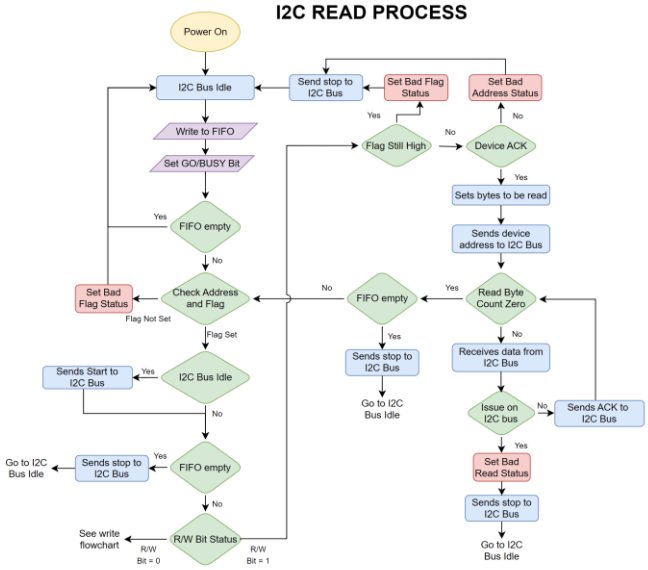


Figure 8: I2C Read Flow chart

I2C Write Process

Figure 9 below is a flowchart to that demonstrates how the CM35I2C03HR handles the I²C write command. The CM35I2C03HR will handle the ACK/NACK handshaking. Once the user set the GO/BUSY bit, this bit will be held low until the bus returns to idle state. Once GO/BUSY is high, the user should check to see if any status bits have been set to indicate an error has occurred.

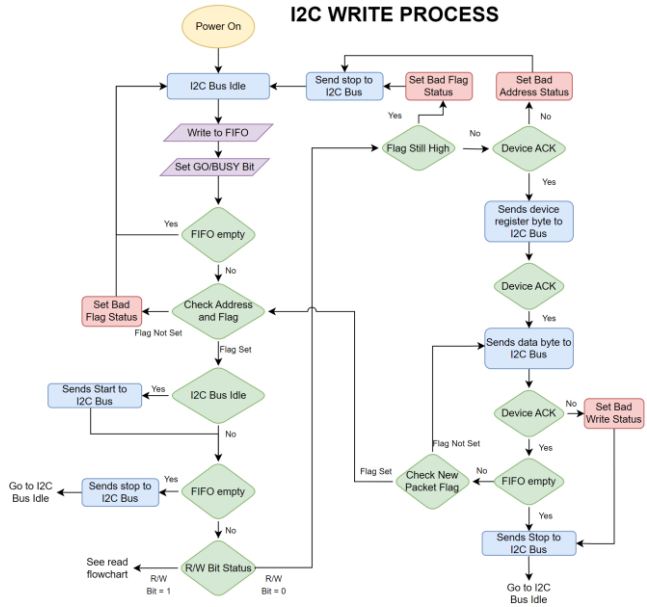


Figure 9: I2C Write Flow chart

4.2.2 NXP PCA9616 DRIVER INFORMATION

The CM35I2C03HR the uses the NXP PCA9616 differential I2C-bus buffer to drive the I²C bus. The PCA9616 ports can be used to be achieve I²C communication up to 3m at 1Mhz clock rates.

NXP PCA9616 Buffer

Any I2C device connected to the CM35I2C03HR differential ports requires a PCA9616. Figure 10 is a reference example circuit for the NXP PCA9616. The single ended I²C (pins 2, 4, 7) should be connected to the I²C slave device, as the CM35I2C03HR is the host. The differential I²C (pins 10-15) will should be wired to a CM35I2C03HR differential port. See section CN3/CN5/CN6: Differential I2C Connector on page 12. The CM35I2C03HR has $V_{DD(B)}$ set to 5V, the slave side requires the same drive voltage level.

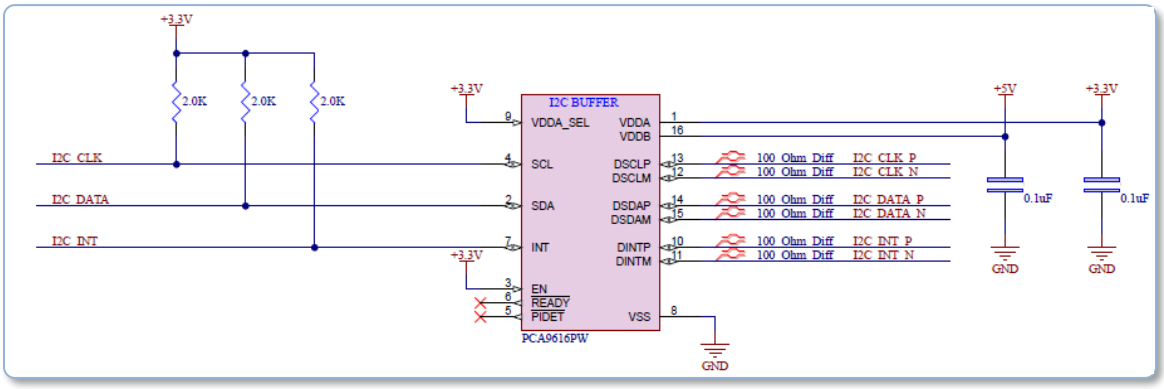


Figure 10: NXP PCA9616 Example Circuit

Slave Termination Circuit

Typical single-ended I²C is not suited for long distance due to ringing and reflections. The use of termination and differential signaling minimizes this issue. The CM35I2C03HR provides 5V to the $V_{DD(B)}$ of the PCA9616 for output drive. The termination pull resistors should be connected to 5V and GND, as shown in Figure 11. When designing a slave controller, you must provide termination, as it will be an end point.

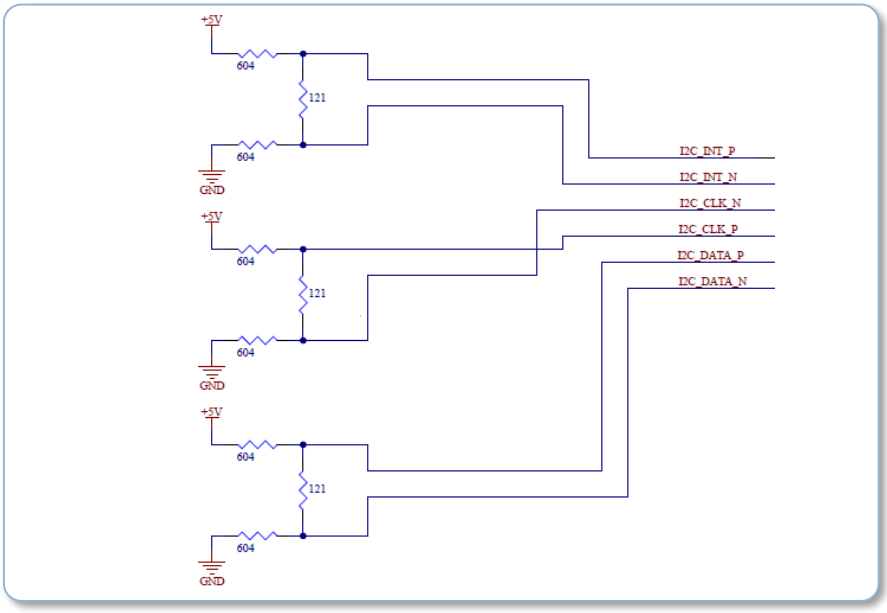


Figure 11: PCA9616 Termination

4.2.3 DIFFERENTIAL CABLING

The differential I²C interface is designed for 100Ω impedance transmission lines. Using the termination circuit shown above, RTD suggest using CAT5, CAT6, or CAT7 data cable for cabling between host and slave. CAT5, CAT6 and CAT7 provide 4 twisted pairs, the 4 pairs can be used for 3 signals (INT, CLK, DATA) and power.

5 Register Address Space

This product was built as a modular design, which allows each board function to have its own Functional Block (FB). Each functional block is designed to work independently. For this reason, we provide individual DMA channels, interrupts, clocks, and FIFOs to each functional block.

The registers are described by their PCIe Base Address Register (BAR), which is defined in the PCI configuration space for this board. The configuration space is generally handled by the operating system. For more information on how to use the configuration space, consult the *PCI Local Bus Specification, Revision 3.0* from the PCI-SIG.

Register Types

There are several different types of registers that are referred to in this section. A description of each type is below.

- Read/Write Registers: The value that is written to this register can also be read back.
- Maskable Registers: This is a 32 bit register that consists of 16-bit data field in the upper word and a 16-bit mask value in the lower word. For each bit in the data field, it is only written to the register if the corresponding bit in the mask field is '1'.
- Sticky Registers: This is a status read register. When a bit in this register has a value of '1', a '1' needs written to that bit to reset the register to '0'. This is typically used for interrupt status registers.
- Read Only: This register can only be read.

NOTE: Writing to Read-Only registers may have unexpected results.

Function Block Mapping

Below is the Function Block slot mapping for the CM35I2C03HR.

Table 9: Function Block Mapping

Slot	ID	Description	Connector
FB0	0x00003300	I2C Host Controller	CN3/CN4
FB1	0x00003300	I2C Host Controller	CN5
FB2	0x00003300	I2C Host Controller	CN6
FB3	0x00003300	I2C Host Controller	CN9
FB4	0x00003300	I2C Host Controller	CN10
FB5	0x00003300	I2C Host Controller	CN11
FB6	0x0000F002	Flash Programmer	

5.1 BAR0: General Board Control

This section contains the board-level interrupt control, as well as a description of all the Functional Blocks in BAR2. Most of the registers in this section are Read-Only. Although a full implementation requires BAR0 to be 0x400 in length, most implementations will have a shorter length.

Table 10: Base Functional Block

Offset	0x03	0x02	0x01	0x00
0x00	GBC_BRD_RST	GBC_EOI	GBC_REV	GBC_FMT
0x04	GBC_PDP			
0x08	GBC_BUILD			
0x0C	*Reserved*	GBC_USER_ID	GBC_SYS_CLK_FREQ	
0x10	GBC_IRQ_STATUS			
0x14				
0x18	GBC_DIRQ_STATUS			
0x1C				
0x20	FB0_ID			
0x24	FB0_OFFSET			
0x28	FB0_OFFSET_DMA			
0x2C	reserved			
0x30	FB1_ID			
0x34	FB1_OFFSET			
0x38	FB1_OFFSET_DMA			
0x3C	reserved			
…				
0x20+0x10*n	FBn_ID			
0x24+0x10*n	FBn_OFFSET			
0x28+0x10*n	FBn_OFFSET_DMA			
0x2C+0x10*n	reserved			
…				
0x3F0	FB61_ID			
0x3F4	FB61_OFFSET			
0x3F8	FB61_OFFSET_DMA			
0x3FC	reserved			

5.1.1 GBC_FMT (READ-ONLY)

This register contains the format ID that is used in this board.

- B[7:4] Major Format – Changes to this value indicate changes that are not backward compatible.
  - 0 – Initial Major Format
- B[3:0] Minor Format – Changes to this value indicate additional features that are backward compatible.
  - 1 – Initial Minor Format
  - 2 – Addition of FB_DMAm_CTRLn[Pause] flag and Early DMA Buffer Termination.

5.1.2 GBC_REV (READ-ONLY)

This register contains the PDP revision for this board. A=1, B=2, etc.

5.1.3 GBC_BRD_RST (READ/WRITE)

Write a value of 0xAA to this register to reset the board.

5.1.4 GBC_PDP (READ-ONLY)

This register contains the PDP number for this board in decimal.

5.1.5 GBC_BUILD (READ-ONLY)

Unique 32-bit build number for every FPGA build.

5.1.6 GBC_SYS_CLK_FREQ (READ ONLY)

The measured frequency of the system clock. Units are 10 kHz, i.e. (Frequency in Hertz) = (GBC_SYS_CLK_FREQ * 10 kHz). This is measured once after any reset, including GBC_BRD_RST. This value is not available (will read 0) until 100us after reset.

5.1.7 GBC_USER_ID (READ ONLY)

This value is set by jumpers on the board, and is used to identify individual board when there are multiple in the same system.

5.1.8 GBC_IRQ_STATUS (READ/CLEAR)

This is a 64-bit interrupt status register for non-DMA interrupts. Each bit in this register corresponds to one of the Function Blocks; bit 0 corresponds to FB0 (whose ID and OFFSET are at 0x020), etc. Bits 60 through 63 are reserved. This is a "sticky" register, and the driver clears it by writing a '1' to the appropriate bit. The interrupts signaled by this registered are generally handled by a User Mode Delayed Procedure Call. Any status information required to service the interrupt must be latched in the Function Block. This typically means that there will also be a sticky status register in the Function Block.

The Interrupt Service Routine must be able to clear the interrupt without accessing any registers in the Function Block. This means that if a persistent interrupt condition occurs (i.e. FIFO Empty) the GBC_IRQ_STATUS register will initially be set. The ISR will then clear the bit in GBC_IRQ_STATUS and launch the DPC. In this state the interrupt condition still exists, but the status bit is cleared (because we are servicing it). Depending on implementation, that Function Block may not be able to issue another interrupt while in this state. In this state the full status of the board cannot be determined just from the GBC_IRQ_STATUS register. The Function Block should also maintain a status register that will indicate if the interrupt condition still exists, which will be cleared by the DPC. By doing an End-Of-Interrupt (see Section 5.1.10 GBC_EOI (Read/Clear) below) any "hidden" interrupts will be generated again.

5.1.9 GBC_DIRQ_STATUS (READ/CLEAR)

This is a 64-bit interrupt status register for DMA interrupts. Each bit in this register corresponds to one of the Function Blocks; bit 0 corresponds to FB0 (whose ID and OFFSET are at 0x020), etc. Bits 60 through 63 are reserved. This is a "sticky" register, and the driver clears it by writing a '1' to the appropriate bit. The interrupts signaled by this registered are generally handled by the Kernel Mode ISR. This interrupt is only for the Function Block Standard DMA in Section **5.2** on page 24.

5.1.10 GBC_EOI (READ/CLEAR)

This register is used to acknowledge an interrupt. It is used to safeguard against missing an interrupt. At the end of the ISR, write a 0x01 to this register. If there is another interrupt pending in the status registers, the interrupt line is toggled (Legacy Mode), or another interrupt is sent (MSI Mode).

5.1.11 FBN_ID (READ-ONLY)

This is a 32-bit value that identifies the type of Function Block in slot 'n'. Any changes to a Function Block result in a new Block ID. The mappings for the Functional Block ID can be found on page 21. The ID consists of three fields:

- B[31:24] Type Revision – This is used to indicate the addition of features to the function block. Any checks in the software should be of the form (Type Revision > NN) so that the driver doesn't fail with a newer FPGA. Any changes to the FPGA that would break existing software must change the Type, not the Type Revision.
- B[23:16] Subtype – A Subtype is a variant of the Type with one of the following differences. This allows the driver and library to use the same functions, but have different enumerations.
- B[15:0] Type – The Type of the Function Block is used to define the register map. Any time that the positions of registers change, or registers are added or removed require a new Type number. This is loosely separated into
    - B[15:8] Family – i.e. ADC, DAC, Digital I/O, etc.
    - B[7:0] Member

5.1.12 FBN_OFFSET (READ-ONLY)

This is the offset from the beginning of the Functional Block section (typically the offset in BAR2) that this Functional Block resides in.

5.1.13 FBN_OFFSET_DMA (READ-ONLY)

This is the offset from the beginning of the Functional Block section (typically the offset in BAR2) that the Functional Block DMA Registers resides in.

5.2 BAR2: Functional Block Standard DMA

This section describes a standard DMA engine implementation that may be used by the Functional Blocks. Any Functional Block that uses DMA shall use this implementation so that the kernel mode driver may be used to service it, resulting in better performance.

Table 11: DMA Registers

Offset	0x03	0x02	0x01	0x00
D + 0x00	FB_DMAm_Stat_Underflow	FB_DMAm_Stat_Overflow	FB_DMAm_Setup	FB_DMAm_Action
D + 0x04	FB_DMAm_Current_Buffer	FB_DMAm_Count		
D + 0x08	FB_DMAm_RD_FIFO_CNT		FB_DMAm_WR_FIFO_CNT	
D + 0x0C	FB_DMAm_Last_Action	FB_DMAm_Stat_Complete	FB_DMAm_Stat_Invalid	FB_DMAm_Stat_Used
D + 0x10	FB_DMAm_CTRL0	FB_DMAm_STAT0	Reserved	
D + 0x14	Reserved	FB_DMAm_SIZE0		
D + 0x18	FB_DMAm_ADDRESS0			
D + 0x1C				
D + 0x20	FB_DMAm_CTRL1	FB_DMAm_STAT1	Reserved	
D + 0x24	Reserved	FB_DMAm_SIZE1		
D + 0x28	FB_DMAm_ADDRESS1			
D + 0x2C				
…				
D + 0x10 + (0x10 * n)	FB_DMAm_CTRLn	FB_DMAm_STATn	Reserved	
D + 0x14 + (0x10 * n)	Reserved	FB_DMAm_SIZEn		
D + 0x18 + (0x10 * n)	FB_DMAm_ADDRESSn			
D + 0x1C + (0x10 * n)				

5.2.1 FB_DMAm_Action (Read/Write)

After writing to the Action register, the user should poll the Last_Action register (below) until it reads the same value. This shows that the action has been performed by the DMA state machine. This is especially important when entering and exiting the Clear state.

- 0x00 = Clear: Clear the Current Buffer field, the internal offset counters, and the FIFO. DMA is stopped.
- 0x01 = Go: Starts DMA
- 0x02 = Pause: DMA transfers are stopped, but all internal registers maintain their state. During PAUSE you will still receive Stat_Underflow and Stat_Overflow interrupts. After PAUSE, you may transition to GO or CLEAR.
- 0x03 = Halt: Buffer has been filled that has the HALT bit set, or has attempted to use a buffer with the Valid bit cleared. After HALT, you must transition to CLEAR.

5.2.2 FB_DMAm_Last_Action (Read/Write)

The DMA Engine writes the value of FB_DMAm_Action to this register after it has completed this channel. This indicates to the user that the last command has been processed. It specifically aids the transition to Clear. When transitioning to Clear, the user should wait until FB_DMAm_Last_Action indicates that the Clear has been processed before allowing any other Action changes.

The user may also write a value to this register and then poll the register to see when the value changes. This method can be used to detect when the DMA engine services the channel without an Action change.

5.2.3 FB_DMAm_Setup (Read/Write)

- B0:IntEna: Set to '1' to enable the DMA engine to generate interrupts on completion of a buffer.
- B1:ErrIntEna: Set to '1' to enable the DMA engine to generate interrupts on error.
- B2:Direction (RO or R/W): Set to '1' to transfer from the FPGA to the PCI bus. Clear to '0' to transfer from the PCI bus to the FPGA.
- B3:IgnoreUsed: Set to '1' to prevent an error condition when accessing a buffer with the Used bit set. When "IgnoreUsed" is set, the hardware will still set the Used_Desc bit (below) if the next buffer has the Used bit set. However, it will not generate an interrupt, and will not PAUSE the DMA engine. Examples are continuous output from a DAC, or very large Pre-trigger buffering using system memory.

5.2.4 FB_DMAm_Stat_Used (Read/Write)

This register is used to determine the source of a DMA interrupt. The bits are cleared by writing 0x00 to the byte. Stat_Used will be set regardless of having ErrIntEna set to '1', and regardless of the state of the IgnoreUsed bit (above).

- B0: Used_Desc (R/C). Set to '1' by the DMA engine if it attempting to use a descriptor with the Used bit set.

5.2.5 FB_DMAm_Stat_Invalid (Read/Write)

This register is used to determine the source of a DMA interrupt. The bits are cleared by writing 0x00 to the byte. Stat_Invalid will be set regardless of having ErrIntEna set to '1'.

- B0: Invalid_Desc (R/C). Set to '1' by the DMA engine if it attempting to use a descriptor with the Valid bit cleared.

5.2.6 FB_DMAm_Stat_Overflow (Read/Write)

This register is used to determine the source of a DMA interrupt. The bits are cleared by writing 0x00 to the byte. Stat_Overflow will be set regardless of having ErrIntEna set to '1'.

- B0: Overflow (R/C). Set to '1' by the DMA engine if an overflow occurred on the FIFO.

5.2.7 FB_DMAm_Stat_Underflow (Read/Write)

This register is used to determine the source of a DMA interrupt. The bits are cleared by writing 0x00 to the byte. Stat_Underflow will be set regardless of having ErrIntEna set to '1'.

- B0: Underflow (R/C). Set to '1' by the DMA engine if an underflow occurred on the FIFO.

5.2.8 FB_DMAm_Stat_Complete (Read/Write)

This register is used to determine the source of a DMA interrupt. The bits are cleared by writing 0x00 to the byte.

- B0: Buffer_Complete (R/C). Set to '1' by the DMA engine when a buffer is filled that has the Interrupt bit set.

5.2.9 FB_DMAm_Current_Buffer (Read-Only)

This is the ID for the buffer that will be used for the next access. The user may use this to track the progress of the DMA activity.

5.2.10 FB_DMAm_COUNT (Read-Only)

This is the offset in the DMA buffer for the next access. The user may use this to track the progress of the DMA activity.

5.2.11 FB_DMAm_RD_FIFO_CNT(Read-Only)

- B[9:0] This is the amount of data available in the read FIFO in bytes. Software can use this to determine when the FIFO is empty. A value of 0x3FC indicates that there are 1020 or more bytes available.
- B15: RD_EMPTY- '1' indicates that the read FIFO is empty

5.2.12 FB_DMAm_WR_FIFO_CNT(Read-Only)

- B[9:0] This is the amount of space available in the write FIFO in bytes. Software can use this to determine when the FIFO is empty. A value of 0x3FC indicated that there are 1020 or more bytes available.
- B15: WR_FULL- '1' indicates that the read FIFO is full

5.2.13 FB_DMAm_ADDRESSn (Read/Write)

This is the 64-bit PCI address for DMA Channel m, buffer n. It must be double-word aligned (i.e. b[1:0] are reserved).

5.2.14 FB_DMAm_SIZEn (Read/Write)

This is the size in bytes of the buffer for DMA Channel m, buffer n. It must be an integer number of double-words (i.e. b[1:0] are reserved). The actual size is FB_DMAm_SIZEn + 4 Bytes. The maximum buffer size is 16MB.

5.2.15 FB_DMAM_CTRLN (READ/WRITE)

- B0: Valid: Sets this bit to '1' to indicate that this contains valid information. The DMA engine will set the error bit and halt if it is ready to use this descriptor and it is not valid.
- B1: Halt: Sets this bit to '1' to halt the DMA engine after this buffer is full.
- B2: Loop: Sets this bit to '1' to start back at descriptor 0 after this buffer is full. This has a higher priority than the HALT bit.
- B3: Interrupt: Sets this bit to '1' to generate an interrupt after this buffer is full.
- B4: Pause: Sets this bit to '1' to pause the DMA engine after this buffer is full.

If the last buffer is reached, and the HALT and LOOP bits are both '0', the DMA engine will loop.

If the last buffer is reached, and the HALT and LOOP bits are both '1', the DMA engine will halt and the Current_Buffer will be set to 0.

5.2.16 FB_DMAM_STATN (READ/CLEAR)

- B0: Used (R/C): DMA engine sets to '1' to indicate that it has completely used this descriptor. The user must clear this bit when it is ready to be used again. The DMA engine will set the error bit and PAUSE if it is ready to use this descriptor and the Used bit is set, unless the IgnoreUsed bit is set. The bits are cleared by writing 0x00 to the byte.
- B1: Terminated (R/C): DMA engine sets to '1' to indicate that this buffer was terminated early at the request of the associated Function Block.

5.3 BAR2: I2C Host Controller

This function block provides control register for the I2C Host. This function block provides interrupts for events on I2C bus. However there is are no DMA channels for the I2C Host.

5.3.1 Function Block Register Map

Table 12: I2C Host Controller Functional Block

	Offset	0x03	0x02	0x01	0x00
Header	FB + 0x00	FB_ID			
	FB + 0x04	FB_DMA_BUFFERS	FB_DMA_CHANNELS	Reserved	*Reserved*
	FB + 0x08	FB_RST	Reserved	I2C_CONFIG	I2C_BUS_CTRL
	FB + 0x0C	CLK_DIV			
	FB + 0x10	INT_ENA			
	FB + 0x14	INT_STAT		Reserved	
	FB + 0x18	WRITE_FIFO			
	FB + 0x1C	WRITE_FIFO _COUNT			
	FB + 0x20	READ_FIFO			
	FB + 0x24	READ_FIFO_COUNT			

5.3.2 I2C_BUS_CTRL (Read/Write)

This provides control to the I2C bus.

B[0]: GO/BUSY – When the value is 1, the I2C bus is idle. The user can write a '1' to the register to start the I2C to transaction. Once the I2C transaction has started, this bit will go low '0'. Once the I2C transaction is completed this will return to '1'

B[1]: BUS_RESET – Write '1' to issue a stuck bus reset. Once the transaction has started, this bit will go low '0'. Once the bus reset is complete this will return to '1'. This command should be issued if the I2C data line is held low by a device. It will have the I2C controller issue out clocks to try and release the bus.

5.3.3 I2C_CONFIG (Read/Write)

This provides configuration to the Front End for this DAC Channel, to allow adjustment of gains, ranges.

B[0]: Packet Read – When set to 1. If an I2C read operation occurs, the device address will be captured in the read FIFO with bit 8 set to 1. This will be used to separate read operation if back-to-back reads are queued up in the write FIFO.

5.3.4 FB_RST (Read/Write)

Write a value of 0xAA to this register to reset the function block to power defaults.

5.3.5 CLK_DIV (Read/Write)

Divider for the I2C clock. I2C Clock Frequency = (System Clock) / (1 + CLK_DIV). Power on default sets the CLK_DIV = 0x7C. (400 KHz)

5.3.6 INT_ENA (Maskable Read/Write)

Each bit corresponds to an interrupt source. A value of '1' enables the source, and a value of '0' disables it. See below for a description of the sources.

5.3.7 INT_STAT (Read/Write)

Each bit corresponds to an interrupt source. Reading a value of '1' indicates that an event has occurred. Reading a value of '0' indicates that the event has not occurred. Writing a '1' will clear that bit. If bits 3:0 are set high the I2C bus halt after complete current packet.

B[0]: Bad Flag – Indicates there was malformed I2C packet sent.

B[1]: Bad Address – Indicates no device ACK to address sent.

B[2]: Bad Read – Indicates there was an issue with readback.

B[3]: Bad Write – Indicates there was an issue with write

B[4]: FIFO Complete – Indicates the packets in write FIFO has completed

B[5]: Interrupt Line – Indicates that I2C interrupt input is low. User needs to clear I2C device causing interrupt before clearing bit.

5.3.8 WRITE_FIFO (READ/WRITE)

This register is used to write bytes into the I2C write FIFO. This data is sent once the GO/BUSY bit is toggled. Each register writes advances to the next entry. The I2C state machine will pull data out of the FIFO as needed.

This register is cleared, when performing a FB_RST.

If an error occurs during a transaction, the I2C FIFO will step through to next Device Address Flag is found and return the bus to idle.

B[7:0] – I2C Data

B[8] – 1 = I2C Device Address, 0 = I2C Dat

5.3.9 WRITE_FIFO _COUNT (READ-ONLY)

This register shows the current number of commands/data that are in the write FIFO.

5.3.10 READ_FIFO (READ-ONLY)

This register contains the bytes that have captured during I2C read process. Each register access advances to the next sample.

B[7:0] – I2C Data

If Packet Read = 1:

B[8] – 1 = I2C Device Address, 0 = I2C Data

5.3.11 READ_FIFO_COUNT (READ-ONLY)

This register shows the current number bytes of data that are in the read FIFO.

5.4 BAR2: Flash Programmer (Reserved)

This function block is used to write new firmware to the CM35I2C03HR. This allows RTD to provide updates to CM35I2C03HR without the need to returning the board back to the factory.

Table 13: Flash Programmer Functional Block

	Offset	0x03	0x02	0x01	0x00
Header	FB + 0x00	FB_ID			
	FB + 0x04	*Reserved*			
	FB + 0x08	Reserved			
	FB + 0x0C	Reserved			
	FB + 0x10	Reserved			
	FB + 0x14	Reserved			
	FB + 0x18	SERIAL_NUMBER			

5.4.1 SERIAL_NUMBER (READ-ONLY)

This register provides the serial number of the CM35I2C03HR product. This allows users a way to reteive the serial number without have access to the board.

6 Troubleshooting

If you are having problems with your system, please try the following initial steps:

- **Simplify the System** – Remove modules one at a time from your system to see if there is a specific module that is causing a problem. Perform you troubleshooting with the least number of modules in the system possible.

- **Swap Components** – Try replacing parts in the system one at a time with similar parts to determine if a part is faulty or if a type of part is configured incorrectly.

If problems persist, or you have questions about configuring this product, contact RTD Embedded Technologies via the following methods:

Phone: +1-814-234-8087
E-Mail: techsupport@rtd.com

Be sure to check the RTD web site (http://www.rtd.com) frequently for product updates, including newer versions of the board manual and application software.

7 Additional Information

7.1 PC/104 Specifications

A copy of the latest PC/104 specifications can be found on the webpage for the PC/104 Embedded Consortium:

www.pc104.org

7.2 PCI and PCI Express Specification

A copy of the latest PCI and PCI Express specifications can be found on the webpage for the PCI Special Interest Group:

www.pcisig.com

8 Limited Warranty

RTD Embedded Technologies, Inc. warrants the hardware and software products it manufactures and produces to be free from defects in materials and workmanship for one year following the date of shipment from RTD Embedded Technologies, Inc. This warranty is limited to the original purchaser of product and is not transferable.

During the one year warranty period, RTD Embedded Technologies will repair or replace, at its option, any defective products or parts at no additional charge, provided that the product is returned, shipping prepaid, to RTD Embedded Technologies. All replaced parts and products become the property of RTD Embedded Technologies. Before returning any product for repair, customers are required to contact the factory for a Return Material Authorization (RMA) number.

This limited warranty does not extend to any products which have been damaged as a result of accident, misuse, abuse (such as: use of incorrect input voltages, improper or insufficient ventilation, failure to follow the operating instructions that are provided by RTD Embedded Technologies, “acts of God” or other contingencies beyond the control of RTD Embedded Technologies), or as a result of service or modification by anyone other than RTD Embedded Technologies. Except as expressly set forth above, no other warranties are expressed or implied, including, but not limited to, any implied warranties of merchantability and fitness for a particular purpose, and RTD Embedded Technologies expressly disclaims all warranties not stated herein. All implied warranties, including implied warranties for merchantability and fitness for a particular purpose, are limited to the duration of this warranty. In the event the product is not free from defects as warranted above, the purchaser's sole remedy shall be repair or replacement as provided above. Under no circumstances will RTD Embedded Technologies be liable to the purchaser or any user for any damages, including any incidental or consequential damages, expenses, lost profits, lost savings, or other damages arising out of the use or inability to use the product.

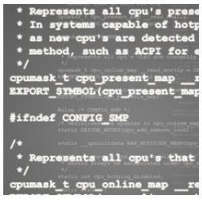
Some states do not allow the exclusion or limitation of incidental or consequential damages for consumer products, and some states do not allow limitations on how long an implied warranty lasts, so the above limitations or exclusions may not apply to you.

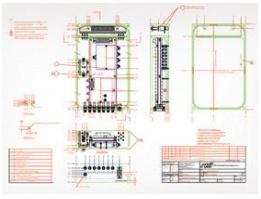
This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

RTD Embedded Technologies, Inc.
103 Innovation Boulevard
State College, PA 16803 USA
Telephone: 814-234-8087
Fax: 814-234-5218

www.rtd.com

sales@rtd.com
techsupport@rtd.com





Copyright 2024 by RTD Embedded Technologies, Inc. All rights reserved.