DM34116HR

1-25MHz A/D Digitizer with 25Mhz D/A Outputs

User's Manual

BDM-610010057 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

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, expressMate, ExpressPlatform, HiDANplus, "MIL Value for COTS prices", multiPort, PlatformBus, and PC/104EZ are trademarks, and "Accessing the Analog World", dataModule, IDAN, HiDAN, RTD, and the RTD logo 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 Embedded 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 © 2019 by RTD Embedded Technologies, Inc. All rights reserved.

Table of Contents

1 Introduction **10**

1.1 Product Overview 10

1.2 Board Features 10

1.3 Ordering Information 11

1.4 Contact Information 11
1.4.1 Sales Support 11
1.4.2 Technical Support 11

2 Specifications **12**

2.1 Operating Conditions 12

2.2 Electrical Characteristics 12

2.1 Functional Characteristics 13
2.1.1 Analog Input FFT plots 13

3 Board Connection **15**

3.1 Board Handling Precautions 15

3.2 Physical Characteristics 15

3.3 Connectors and Jumpers 16
3.3.1 Bus Connectors 16
3.3.2 External I/O Connectors 17
3.3.3 Jumpers 18
3.3.4 LEDs 18

3.4 Steps for Installing 19

4 IDAN Connections **20**

4.1 Module Handling Precautions 20

4.2 Physical Characteristics 20

4.3 Connectors 21
4.3.1 External I/O Connectors 21
4.3.2 Bus Connectors 22

4.4 Steps for Installing 23

5 Functional Description **24**

5.1 Block Diagram 24

5.2 FPGA with DMA Engine 24

5.3 Programmable Clock 25

5.4 SyncBus 25
5.4.1 Clock Generator Sync 25

5.5 Capture Window 26

5.6 Advanced Digital I/O 26
5.6.1 Quad Pulse Width Modulator 27
5.6.2 Incremental Encoder 27
5.6.3 External Clocking 28

5.7 Temperature Sensor 29

5.8 Analog input 29
5.8.1 Initializing the ADC Converter 30
5.8.2 Simplified block diagram of analog input 30
5.8.3 Input connection examples 31

5.9 Analog Output 32
5.9.1 Initializing the DAC Converter 32

5.9.2 Simplified block diagram of analog output 33

6 Register Address Space 34

6.1 BAR0: General Board Control ... 35
6.1.1 GBC_FMT (Read-Only) 35
6.1.2 GBC_REV (Read-Only) 35
6.1.3 GBC_BRD_RST (Read/Write) 35
6.1.4 GBC_PDP (Read-Only) 35
6.1.5 GBC_BUILD (Read-Only) 35
6.1.6 GBC_SYS_CLK_FREQ (Read Only) 36
6.1.7 GBC_USER_ID (Read Only) 36
6.1.8 GBC_IRQ_STATUS (Read/Clear) 36
6.1.9 GBC_DIRQ_STATUS (Read/Clear) 36
6.1.10 GBC_EOI (Read/Clear) 36
6.1.11 FBn_ID (Read-Only) 36
6.1.12 FBn_Offset (Read-Only) 36
6.1.13 FBn_Offset_DMA (Read-Only) 36

6.2 BAR2: Functional Block Standard DMA ... 37
6.2.1 FB_DMAm_Action (Read/Write) 37
6.2.2 FB_DMAm_Last_Action (Read/Write) 37
6.2.3 FB_DMAm_Setup (Read/Write) 37
6.2.4 FB_DMAm_Stat_Used (Read/Write) 38
6.2.5 FB_DMAm_Stat_Invalid (Read/Write) 38
6.2.6 FB_DMAm_Stat_Overflow (Read/Write) 38
6.2.7 FB_DMAm_Stat_Underflow (Read/Write) 38
6.2.8 FB_DMAm_Stat_Complete (Read/Write) 38
6.2.9 FB_DMAm_Current_Buffer (Read-Only) 38
6.2.10 FB_DMAm_COUNT (Read-Only) 38
6.2.11 FB_DMAm_RD_FIFO_CNT(Read-Only) 38
6.2.12 FB_DMAm_WR_FIFO_CNT(Read-Only) 38
6.2.13 FB_DMAm_ADDRESSn (Read/Write) 38
6.2.14 FB_DMAm_SIZEn (Read/Write) 38
6.2.15 FB_DMAm_CTRLn (Read/Write) 39
6.2.16 FB_DMAm_STATn (Read/Clear) 39

6.3 BAR2: Functional Block Header Registers ... 39
6.3.1 FB_ID (Read-Only) 39
6.3.2 FB_DMA_CHANNELS (Read -Only) 39
6.3.3 FB_DMA_BUFFERS (Read-Only) 39
6.3.4 FB_CONTROL_SIZE (Read -Only) 39
6.3.5 FB_CHANNEL_SIZE (Read-Only) 39

6.4 BAR2: Analog to Digital Converter (ADC) ... 40
6.4.1 Function Block Register Map 40
6.4.2 Mode_Status (Read/Write, Read-Only) 40
6.4.3 START_TRIG (Read/Write) 41
6.4.4 STOP_TRIG (Read/Write) 41
6.4.5 CLK_DIV(Read/Write) 41
6.4.6 CLK_DELAY(Read/Write) 42
6.4.7 CLK_READY 42
6.4.8 PRE_TRIGGER_CAPTURE (Read/Write) 42
6.4.9 POST_STOP_CAPTURE (Read/Write) 42
6.4.10 SAMPLE_CNT (Read Only) 42
6.4.11 INT_ENA (Maskable Read/Write) 42
6.4.12 INT_STAT (Read/Clear) 42
6.4.13 CLK_BUSn 43
6.4.14 AD_CONFIG (Maskable Read/Write) 43
6.4.15 ADC_TRIM (Read/Write) 43
6.4.16 ADC_TRIM_BUSY (Read Only) 43
6.4.17 FRONT_END_CONFIG (Maskable Read/Write) 43
6.4.18 FIFO_DATA_CNT (Read) 43
6.4.19 MAX_FIFO_SIZE (Read) 44
6.4.20 FILTER (Read/Write) 44
6.4.21 INT_STAT(Read/Clear) 44
6.4.22 INT_ENA (Read/Write) 45
6.4.23 THRESH_LOW (Read/Write) 45

6.4.24 THRESH_HIGH (Read/Write) 45
6.4.25 LAST_SAMPLE (Read-Only) 45
6.4.26 CH_FIFO_ACCESS (Read/Write) 45

6.5 BAR2: Digital to Analog Converter (DAC) 46
6.5.1 Function Block Register Map 46
6.5.2 Mode_Status (Read/Write, Read-Only) 46
6.5.3 START_TRIG (Read/Write) 47
6.5.4 STOP_TRIG (Read/Write) 47
6.5.5 CLK_DIV(Read/Write) 47
6.5.6 CLK_DELAY(Read/Write) 48
6.5.7 CLK_READY 48
6.5.8 POST_STOP_CONVERSIONS (Read/Write) 48
6.5.9 CONVERSION_CNT (Read Only) 48
6.5.10 INT_ENA (Maskable Read/Write) 48
6.5.11 INT_STAT (Read/Clear) 48
6.5.12 CLK_BUSn 49
6.5.13 DA_CONFIG (Maskable Read/Write) 49
6.5.14 DAC_TRIM (Read/Write) 49
6.5.15 DAC_TRIM_BUSY (Read Only) 49
6.5.16 DAC_OFFSET (Read/Write) 49
6.5.17 DAC_OFFSET_BUSY (Read Only) 49
6.5.18 FRONT_END_CONFIG (Maskable Read/Write) 49
6.5.19 FIFO_DATA_CNT (Read) 50
6.5.20 MAX_FIFO_SIZE (Read) 50
6.5.21 INT_STAT (Read/Clear) 50
6.5.22 INT_ENA (Read/Write) 50
6.5.23 LAST_CONVERSION (Read/Write) 50
6.5.24 FIFO_ACCESS (Read/Write) 50

6.6 BAR2: Advanced Digital I/O 51
6.6.1 Function Block Register Map 51
6.6.2 Mode_Status (Read/Write, Read-Only) 51
6.6.3 CLK_SRC (Read/Write) 52
6.6.4 START_TRIG (Read/Write) 52
6.6.5 STOP_TRIG (Read/Write) 52
6.6.6 CLK_DIV (Read/Write) 53
6.6.7 CLK_DIV_CNTR (Read Only) 53
6.6.8 PRE_START_COUNT (Read/Write) 53
6.6.9 POST_STOP_COUNT (Read/Write) 53
6.6.10 SAMPLE_CNT (Read Only) 53
6.6.11 INT_ENA (Maskable Read/Write) 53
6.6.12 INT_STAT (Read/Clear) 53
6.6.13 CLK_BUSn 53
6.6.14 ADV_INT_MODE (Read/Write) 54
6.6.15 ADV_INT_MASK (Read/Write) 54
6.6.16 ADV_INT_COMP (Read/Write) 54
6.6.17 ADV_INT_CAPT (Read/Write) 54
6.6.18 MODE_CONFIG (MASKABLE READ/WRITE) 55
6.6.19 PERIPH_SEL_BIT_n (Read/Write) 55
6.6.20 CHn_FIFO_DATA_CNT (Read) 55
6.6.21 CHn_MAX_FIFO_SIZE (Read) 55
6.6.22 CH0_DATA (DIG_IN) (Read Only) 56
6.6.23 CH1_DATA (DIG_OUT) (Read/Write) 56
6.6.24 CH2_DATA (DIG_DIR) (Read/Write) 56
6.6.25 CHn_FIFO_ACCESS (Read/Write) 56

6.1 BAR2: PWM 57
6.1.1 Function Block Register Map 57
6.1.2 Mode_Status (Read/Write, Read-Only) 57
6.1.3 CLK_SRC (Read/Write) 58
6.1.4 START_TRIG (Read/Write) 58
6.1.5 STOP_TRIG (Read/Write) 58
6.1.6 CLK_DIV (Read/Write) 58
6.1.7 CLK_DIV_CNTR (Read Only) 58
6.1.8 PERIOD_COUNT (Read Only) 59
6.1.9 INT_ENA (Maskable Read/Write) 59

6.1.10 INT_STAT (Read/Clear) 59
6.1.11 CLK_BUSn 59
6.1.12 CHn_FIFO_DATA_CNT (Read) 59
6.1.13 CHn_MAX_FIFO_SIZE (Read) 59
6.1.14 CHn_PWM_WIDTH (Read/Write) 60
6.1.15 CHn_FIFO_ACCESS (Read/Write) 60

6.2 BAR2: Incremental Encoder 61
6.2.1 Function Block Register Map 61
6.2.2 Mode_Status (Read/Write, Read-Only) 61
6.2.3 CLK_SRC (Read/Write) 62
6.2.4 START_TRIG (Read/Write) 62
6.2.5 STOP_TRIG (Read/Write) 62
6.2.6 CLK_DIV (Read/Write) 62
6.2.7 CLK_DIV_CNTR (Read Only) 62
6.2.8 PRE_START_COUNT (Read/Write) 63
6.2.9 POST_STOP_COUNT (Read/Write) 63
6.2.10 SAMPLE_CNT (Read Only) 63
6.2.11 INT_ENA (Maskable Read/Write) 63
6.2.12 INT_STAT (Read/Clear) 63
6.2.13 CLK_BUSn 63
6.2.14 MODE_CONFIG (MASKABLE READ/WRITE) 64
6.2.15 FIFO_DATA_CNT (Read) 64
6.2.16 MAX_FIFO_SIZE (Read) 64
6.2.17 THRESH_LOW (Read/Write) 64
6.2.18 THRESH_HIGH (Read/Write) 64
6.2.19 INCENC_VALUE (Read/Write) 64
6.2.20 FIFO_ACCESS (Read) 64

6.3 BAR2: External Clocking 65
6.3.1 Function Block Register Map 65
6.3.2 EXT_CLK_IN (Read-Only) 65
6.3.3 EXT_CLK_GATE_IN (Read Only) 65
6.3.4 EXT_CLK_DIR (Read/Write) 65
6.3.5 EXT_CLK_EDGE (Read/Write) 66
6.3.6 EXT_CLKn_PW (Read/Write) 66
6.3.7 EXT_CLKn_CFG (Read/Write) 66

6.4 BAR2: Temperature Sensor 67
6.4.1 Function Block Register Map 67
6.4.2 TEMPERATURE_VAL (Read Only) 67

6.5 BAR2: Clock Generator 67
6.5.1 Function Block Register Map 67
6.5.2 CLK_SEL (Read/Write) 67
6.5.3 PLL_STATUS (Read Only) 68
6.5.4 CLK_SYNC (READ/CLEAR) 68

6.6 BAR2: Capture Window 69
6.6.1 Function Block Register Map 69
6.6.2 Mode_Status (Read/Write, Read-Only) 69
6.6.3 CLK_SRC (Read/Write) 70
6.6.4 START_TRIG (Read/Write) 70
6.6.5 CLK_DIV (Read/Write) 70
6.6.6 CLK_DIV_CNTR (Read Only) 70
6.6.7 SAMPLE_CNT (Read Only) 70
6.6.8 INT_ENA (Maskable Read/Write) 70
6.6.9 INT_STAT (Read/Clear) 71
6.6.10 CLK_BUSn 71
6.6.11 CONFIG (Maskable Read/Write) 72
6.6.12 DELAY_COUNTER_n (Read/Write) 72
6.6.13 CAPTURE_COUNTER_n (Read/Write) 72

6.7 BAR2: SyncBus 73
6.7.1 Function Block Register Map 73
6.7.2 CLK_SEL (Read) 73
6.7.3 PLL_LOCKED (Read Only) 73
6.7.4 TERM_ENABLE (Read/Write) 73

6.7.5 ENABLE (Read/Write) 73
6.7.6 SB_ n_CLK_SRC (Read/Write) 74
6.7.7 DIRECTION (Read/Write) 74
6.7.8 CLK_BUSn 74
6.7.9 PLL_SYNC (Read/Write) 74
6.7.10 SYNCBUS_MODE (Read/Write) 75
6.7.11 DIG_IN (Read-Only) 75
6.7.12 DIG_OUT (Read/Write) 75

6.8 BAR2: Programmable Clock 76
6.8.1 Function Block Register Map 76
6.8.2 Mode_Status (Read/Write, Read-Only) 76
6.8.3 CLK_SRC (Read/Write) 77
6.8.4 START_TRIG (Read/Write) 77
6.8.5 STOP_TRIG (Read/Write) 77
6.8.6 CLK_DIV (Read/Write) 77
6.8.7 CLK_DIV_CNTR (Read Only) 77
6.8.8 COUNT (Read Only) 78
6.8.9 INT_ENA (Maskable Read/Write) 78
6.8.10 INT_STAT (Read/Clear) 78
6.8.11 CLK_BUSn 78
6.8.12 MODE_CONFIG (MASKABLE READ/WRITE) 78
6.8.13 CHn_FIFO_DATA_CNT (Read) 78
6.8.14 CHn_MAX_FIFO_SIZE (Read) 79
6.8.15 INTERVAL_CLK_DIV (Read/Write) 79
6.8.16 LAST_INTERVAL_COUNT (Read-Only) 79
6.8.17 CHn_FIFO_ACCESS (Read/Write) 79

7 Troubleshooting 80

8 Additional Information 81

8.1 PC/104 Specifications 81

8.2 PCI and PCI Express Specification 81

9 Limited Warranty 82

Table of Figures

Figure 1: Gain of 1 FFT 13
Figure 2: Gain of 6 FFT 14
Figure 3: Board Dimensions 15
Figure 4: Board Connections 16
Figure 5: Example 104™Stack 19
Figure 6: IDAN Dimensions 20
Figure 7: Example IDAN System 23
Figure 8: DM34116HR Block Diagram 24
Figure 9: Incremental Encoder Signals 28
Figure 10: Analog Front End 30
Figure 11:High Impedance Bipolar Example 31
Figure 12:High Impedance Unipolar Example 32
Figure 13: Output Circuit 33
Figure 14: Filter Response with each ORDER Value 44

Table of Tables

Table 1: Ordering Options 11
Table 2: Operating Conditions 12
Table 3: Electrical Characteristics 12
Table 4: Functional Characteristics 13
Table 5: CN3 Digital I/O Pin-out 17
Table 6: CN5 Pin-out 18
Table 7: JP1 User ID Jumper 18
Table 8: IDAN- DM34116HR 37-Pin High Density "D" Connector 21
Table 9: Incremental Encoder Inputs 28
Table 10: ADC Full-Scale Settings 31
Table 11: Function Block Mapping 34
Table 12: Base Functional Block 35
Table 13: DMA Registers 37
Table 14: Functional Block Header 39
Table 15: High Speed A/D Functional Block 40
Table 16: High Speed D/A Functional Block 46
Table 17: Advance DIO Functional Block 51
Table 18:PWM Functional Block 57
Table 19: Incremental Encoder Functional Block 61
Table 20:External Clocking Functional Block 65
Table 21: Temperature Sensor Functional Block 67
Table 22: Clock Generator Functional Block 67
Table 23: Capture Window Functional Block 69
Table 24: SyncBus Functional Block 73
Table 25: Programmable Clock Functional Block 76

1 Introduction

1.1 Product Overview

The DM34116HR dataModule is a rugged high-speed data acquisition (DAQ) module in the PCIe/104 format which boasts two 16-bit 25 MHz A/D converters and two 16-bit 25Mhz D/A converters.

This module provides 2 single-ended analog input channels with software-selectable input ranges and input impedances. It also providing 2 single-ended analog output channels with a ± 2.5V range and a fixed 50 ohm output impedance. Each channel has a dedicated converter, permitting both independent or simultaneous conversions. Additionally, each channel also has a dedicated DMA channel, which ensures the ability for the host controller or a DSP to continuously collect data from all four channels across the PCIe x4 interfaces.

The DM34116HR also features 32-bit advanced DIO with peripheral output capabilities including a quad PWM and external clocking. The SyncBus permits multiple DAQ cards to be synchronized within the system.

When used with the RTD's SPM34CP high-performance DSP, data can be transferred directly to the DSP's memory using ta CPU's PCIe connection.

1.2 Board Features

- PCI Express Bus:
  - PCIe/104 Type 2 Board
  - PCIe x4 interface (Gen 2 20 GT/s)
  - In-band interrupts and messages
  - Message Signaled Interrupt (MSI) support
- Analog inputs:
  - 2 high-speed inputs with independent sampling, or simultaneous sampling
  - 25 MSPS maximum input sampling rate
  - 16-bit resolution
  - Single-ended inputs
  - Programmable 50 ohm or high impedance input
  - Programmable input full-scale ranges in bipolar mode: ±1.667V, ±1.25V, ±0.833V, ±0.625V, ±0.4167V
    - Only available in high impedance mode: ±5V, ±3.3V
  - Programmable input full-scale ranges in Unipolar mode: 0-3.3V, 0-2.5V, 0-1.67V, 0-1.25V, 0-0.8333V
    - Only available in high impedance mode: 0-5V
  - Sampling modes and triggers are configurable independently
  - Threshold detection can generate an interrupt, or be used as a start or stop trigger
  - 2 MMCX connectors
- Analog Outputs:
  - 2 high-speed inputs with independent, or simultaneous operation
  - 1-25Mhz DAC update rates
  - 16-bit resolution
  - Single-ended outputs
  - 50 ohm output impedance
  - ±2.5V Output range with adjustable offset
  - 2 MMCX connectors
- SyncBus
  - Allows synchronization of the sampling clock on multiple boards
  - Allows synchronization of start and stop triggers on multiple boards
  - Software selectable Master/Slave
  - Software selectable termination
- Selectable 10Mhz Reference
  - Onboard
  - SyncBus
  - External via MMCX connector
- Digital I/O
  - 32 channels

    - Bit programmable direction
    - 5V Tolerant TTL Signaling level
    - Advance Interrupt
        - Interrupt on Match, Change or Strobe
    - Peripheral Selectable Outputs
        - PWM
        - External Clocking
    - Parallel Bus Mode
    - Max output current 10 mA
- Incremental Encoders
    - One Incremental Encoder channels
    - Single-ended or Pseudo-differential inputs
    - Variable frequency input filtering
    - 32-bit resolution
    - FIFO with DMA channel available for position sampling
- Pulse Width Modulators
    - One Quad PWM outputs
    - Single-ended or Differential outputs
    - 32-bit resolution
    - FIFO with DMA channel
- External Clocking
    - Provides 6 external clocks that can be configured as inputs or outputs
    - Can be used as external trigger
    - External gate provided for each clock input
- On-board temperature monitor

1.3 Ordering Information

The DM34116HR is available with the following options:

Table 1: Ordering Options

Part Number	Description
DM34116HR	PCIe/104 1–25 MHz A/D Converter
IDAN-DM34116HR	PCIe/104 1–25 MHz A/D Converter in modular IDAN frame

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 HiDANplus® 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 57.

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	Typical	Max	Unit
P	Power Consumption[i]	V_{cc5} = 5.0V		TBD		W
I_{cc5}	5V Input Supply Current	Active		TBD		A
		PCIe/104 Bus				
	Differential Output Voltage		0.8		1.2	V
	DC Differential TX Impedance		80		120	Ω
	Differential Input Voltage		0.175		1.2	V
	DC Differential RX Impedance		80		120	Ω
	Electrical Idle Detect Threshold		65		175	mV
		Analog to Digital Converter				
	Linear Input Voltage	IN+ or IN -	-5		5	V
FSR	Full-scale Input Voltage	V_{IN} = (IN+ - IN-) G = PGA Gain	$\frac{-2.5}{G}$		$\frac{2.49996}{G}$	V
	Resolution				16	Bits
	Data Rate				25	MSPS
	Pipeline Delay			7		Cycles
	Input Impedance	High Impedance Mode 50 Ω Impedance Mode		$1x10^6$ 50		Ω
	ENOB			12.48		Bits
	SNR			73.82		dB
	SINAD			73.32		dB
	THD			-83.17		dB
	SFDR			84.69		dB
	Noise Free Bits			TBD		Bits
G	Gains				0.5, 0.75, 1, 1.5, 2, 3, 4, 6	
		Digital to Analog Converter				
FSR	DAC Full-scale Output Voltage		2.5		2.49996	V
	Offset Full-scale Output Voltage		2.5		2.49996	V
	Max Output Voltage	V = DAC + Offset	-4.5		+4.5	V
	Resolution				16	Bits
	Update Rate		1		25	MSPS
	Output Impedance			50		Ω
		External Reference				
	Input Impedance			50		Ω
	Input Voltage		0.2	0.8	2	V_{P-P}
	Input Frequency			10		MHz

Table 3: Electrical Characteristics

Symbol	Parameter	Test Condition	Min	Typical	Max	Unit
Digital I/O (LVTTL)						
V_{IL}	Input High Voltage		2.0		3.3	V
V_{IL}	Input Low Voltage		-0.5		0.8	V
V_{OL}	Output Low Voltage	I_O =-12mA	0		0.4	V
V_{OH}	Output High Voltage	I_O = -12mA	2.4		3.3	V
	5V Output	CN3			100	mA
SyncBus (LVDS)						
	Differential Input Voltage				2.4	V
	Input Voltage Threshold		-0.05		0.05	V
	Differential Output Voltage	I_O = -4µA	0.480		0.650	V
	Common Mode Output Voltage	I_O =-20µA	0.3		2.1	V

2.1 Functional Characteristics

Table 4: Functional Characteristics

Symbol	Parameter	Value	Unit
GBC_SYS_CLK_FREQ	System Clock Frequency	100	MHz
	Analog to Digital FIFO Size	1023	D-Words
	Analog to Digital FIFO Size	1023	D-Words
	Advanced DIO FIFO Size	511	D-Words
	PWM FIFO Size	511	D-Words
	Incremental Encoder FIFO Size	511	D-Words
	Programmable Clock FIFO Size	511	D-Words

2.1.1 Analog Input FFT plots

In Figure 1, a coherent 250kHz sine wave signal was attached to input Channel 0-3 in the +/-2.5V, 50Ω mode. The FFT was generated using 500000 samples.

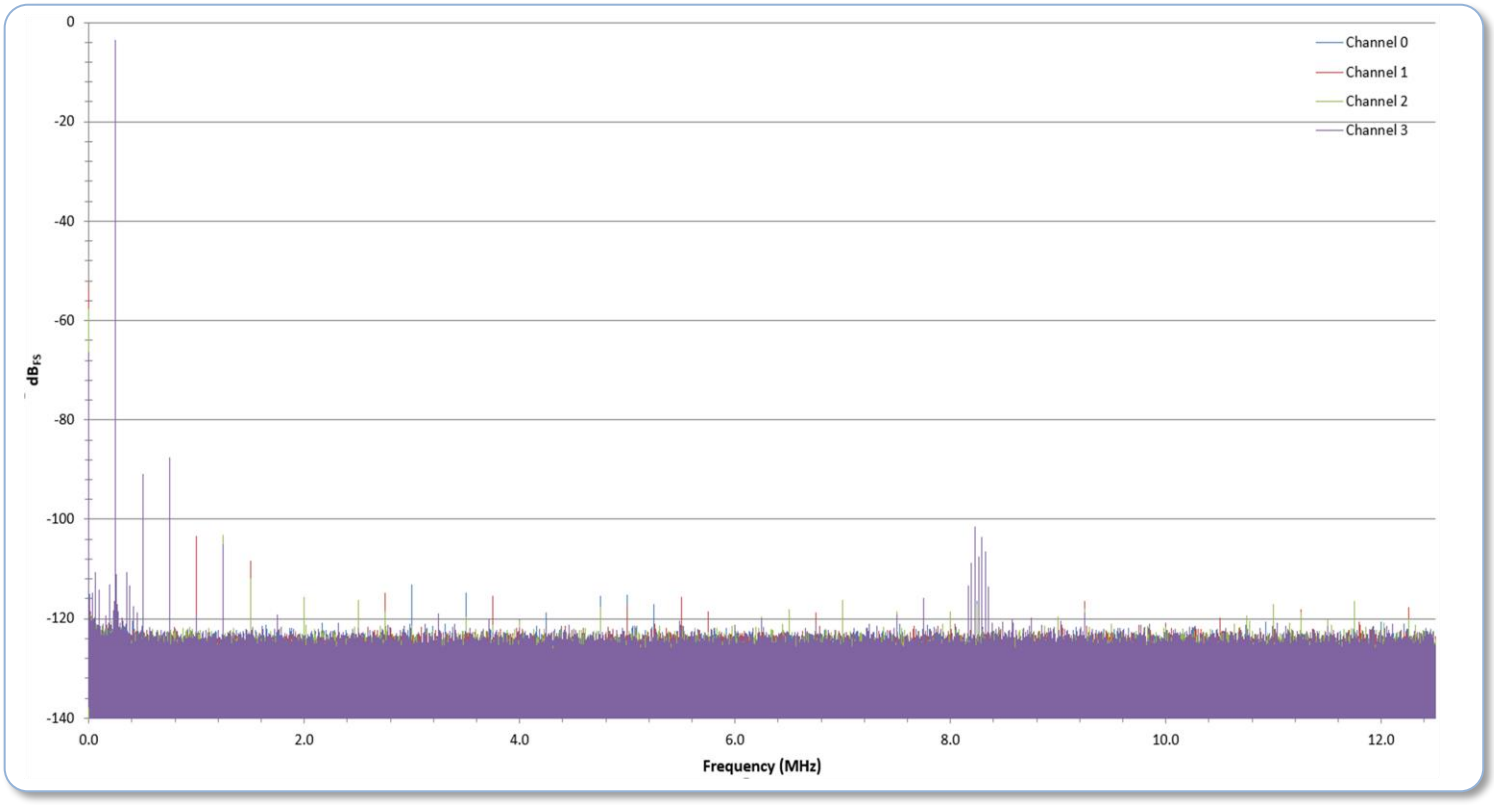


Figure 1: Gain of 1 FFT

In Figure 2, a coherent 250kHz sine wave signal was attached to input Channel 0-3 in the +/-0.4167V, 50Ω mode. The FFT was generated using 500000 samples.

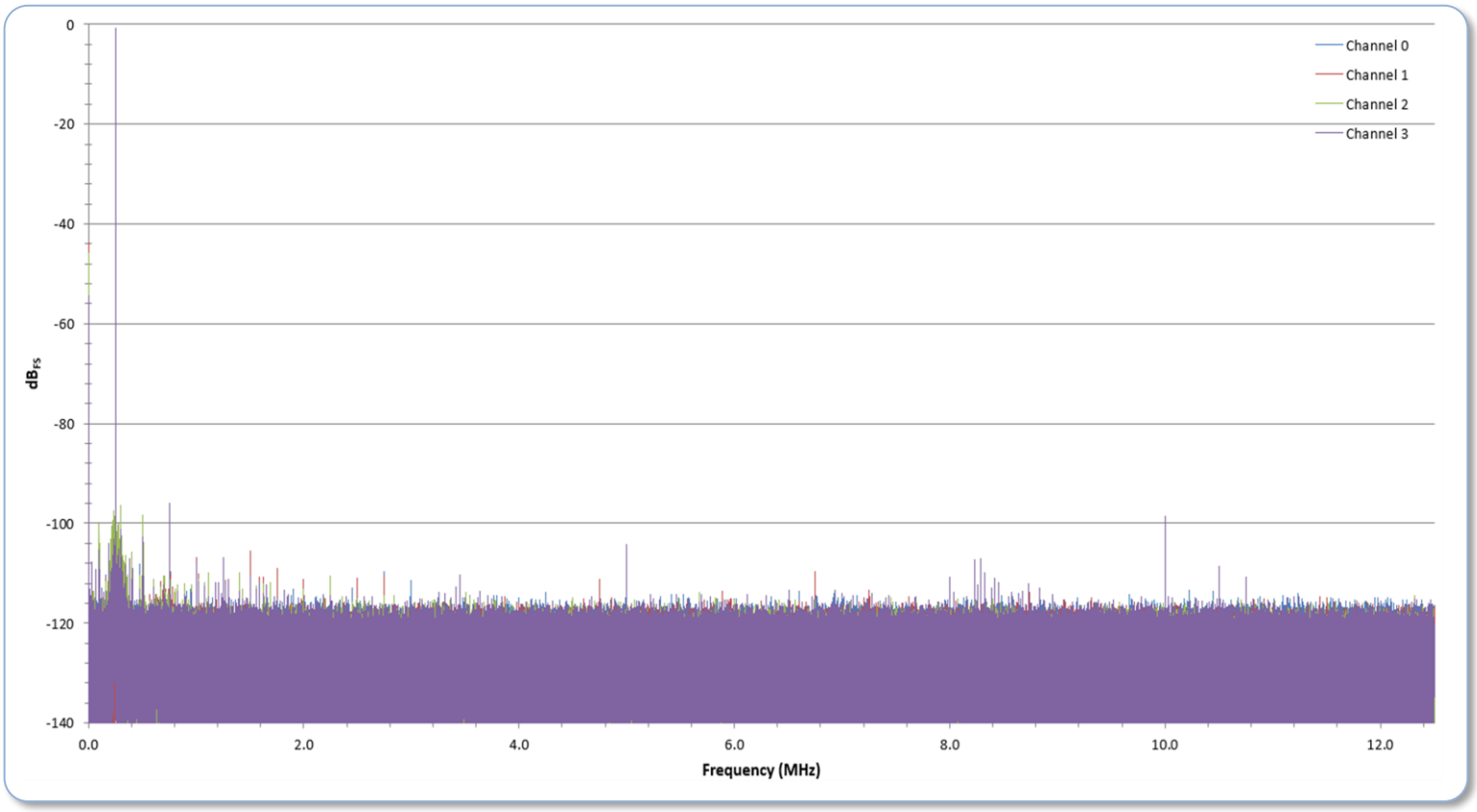


Figure 2: Gain of 6 FFT

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 99.8 g (0.22 lbs) with heatsink
- Dimensions: 90.17 mm L x 95.89 mm W (3.550 in L x 3.775 in W)

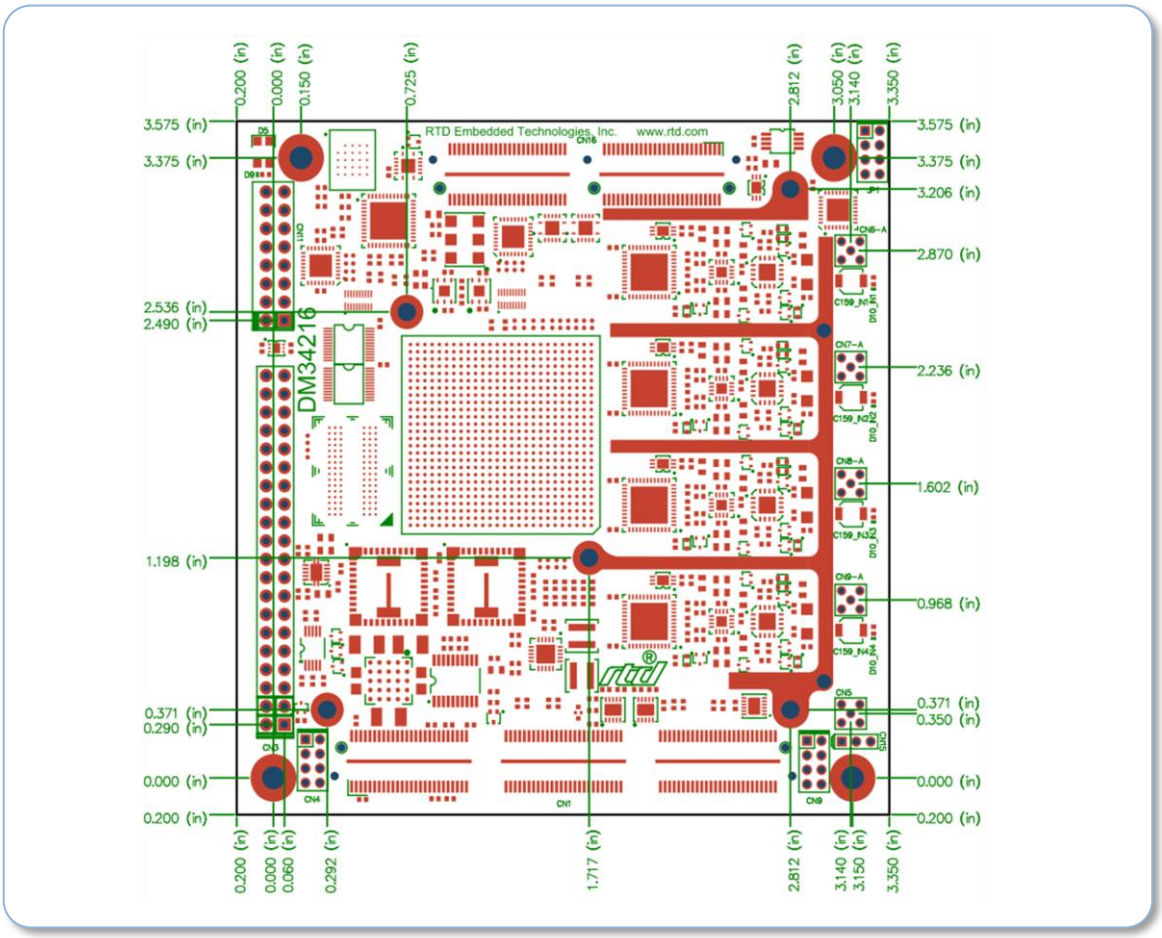


Figure 3: Board Dimensions

3.3 Connectors and Jumpers

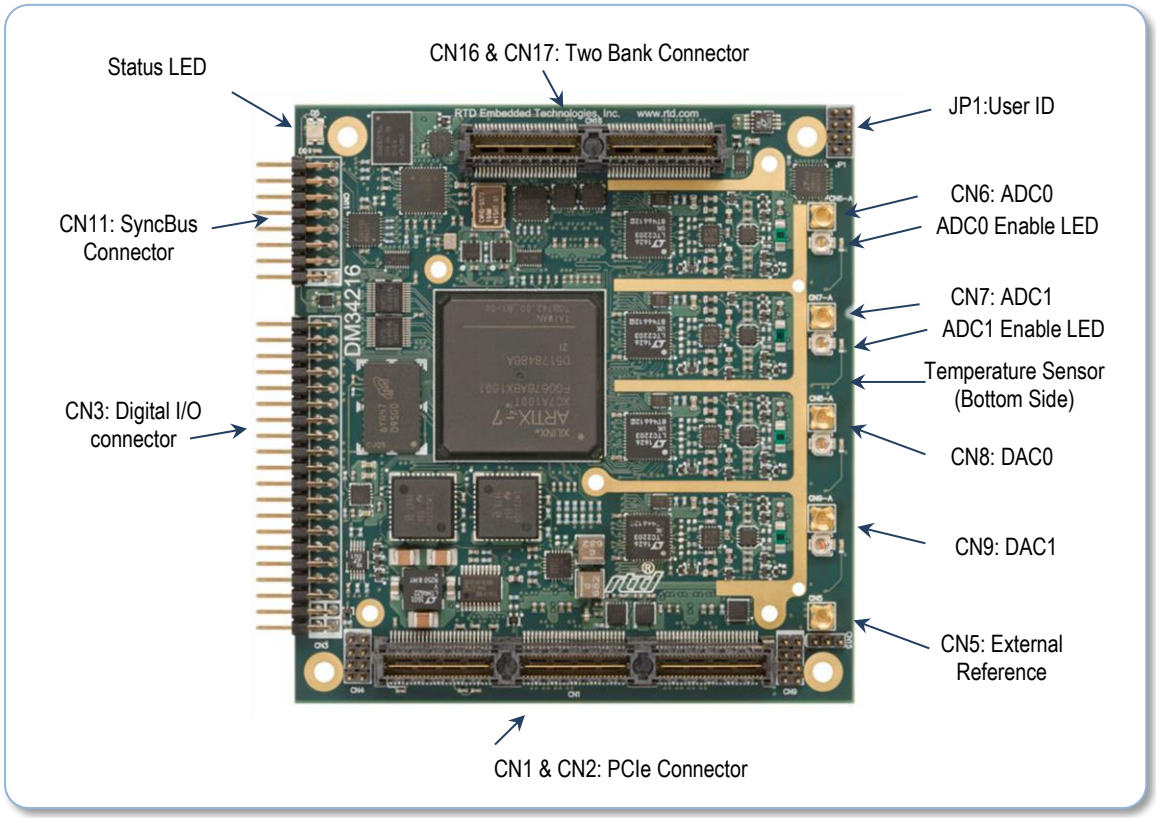


Figure 4: Board Connections

3.3.1 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 58)

The DM34116HR is a Type 2 board, and can only connect to either Type 2 PCIe/104 connector.

CN16(Top) & CN17(Bottom): Two Bank Connector

The Two Bank connector is a high-speed board to board interconnect. Currently the only feature the two bank provides is an independent SyncBus connection. This SyncBus can be used between other RTD products that have a Two Bank connector.

3.3.2 EXTERNAL I/O CONNECTORS

CN3: Digital I/O Connector

The Digital I/O Connector is a 2 x 20, 0.1" spacing right-angle connector. The pin assignments are shown in Table 5 below. A typical mating connector is a FCI 65043-017LF. Pin 1 is indicated by a square solder pad.

Pin 39	Pin 37	Pin 35	Pin 33	Pin 31	Pin 29	Pin 27	Pin 25	Pin 23	Pin 21	Pin 19	Pin 17	Pin 15	Pin 13	Pin 11	Pin 9	Pin 7	Pin 5	Pin 3	**Pin 1**
Pin 40	Pin 38	Pin 36	Pin 34	Pin 32	Pin 30	Pin 28	Pin 26	Pin 24	Pin 22	Pin 20	Pin 18	Pin 16	Pin 14	Pin 12	Pin 10	Pin 8	Pin 6	Pin 4	Pin 2

Table 5: CN3 Digital I/O Pin-out

Alternate Functions		DIO	PIN #		DIO	Alternate Functions	
EXT_CLK_GATE2	PWM_N (0)	DIO0.1	2	1	DIO0.0	PWM_P (0)	EXT_CLK_2
EXT_CLK_GATE3	PWM_N (2)	DIO0.3	4	3	DIO0.2	PWM_P (1)	EXT_CLK_3
EXT_CLK_GATE4	PWM_N (3)	DIO0.5	6	5	DIO0.4	PWM_P (2)	EXT_CLK_4
EXT_CLK_GATE5	PWM_N (4)	DIO0.7	8	7	DIO0.6	PWM_P (3)	EXT_CLK_5
EXT_CLK_GATE6	----------	DIO0.9	10	9	DIO0.8	----------	EXT_CLK_6
EXT_CLK_GATE7	----------	DIO0.11	12	11	DIO0.10	----------	EXT_CLK_7
	INC_ENC_A-	DIO0.13	14	13	DIO0.12	INC_ENC_A+	
	INC_ENC_B-	DIO0.15	16	15	DIO0.14	INC_ENC_B+	
		GND	18	17	GND		
	INC_ENC_ Index-	DIO0.17	20	19	DIO0.16	INC_ENC_ Index+	
		DIO0.19	22	21	DIO0.18		
		DIO0.21	24	23	DIO0.20		
		DIO0.23	26	25	DIO0.22		
		DIO0.25	28	27	DIO0.24		
		DIO0.27	30	29	DIO0.26		
		DIO0.29	32	31	DIO0.28		
		DIO0.31	34	33	DIO0.30		
		GND	36	35	GND		
		+5V	38	37	+5V		
		GND	40	39	GND		

CN5: External Reference Input Connector

The external reference input connector is a MMCX connector. The center pin is the external reference input signal. The outer shell of the connector is connected to ground. A typical mating connector is AMP RF 908-43300.

The external reference connector is used to provide a 10MHz reference clock, allowing the user to synchronize the module with its working environment.

CN6/CN7: Analog Input Connector

The analog input connector is a MMCX connector. The center pin is the analog input signal. The outer shell of the connector is connected to ground. A typical mating connector is AMP RF 908-43300.

CN8/CN9: Analog Output Connector

The analog output connector is a MMCX connector. The center pin is the analog output signal. The outer shell of the connector is connected to ground. A typical mating connector is AMP RF 908-43300.

CN11: SyncBus Connector

The SyncBus Connector is a 2 x 8, 0.1" spacing right-angle connector. The pin assignments are shown in Table 6 below. A typical mating connector is a FCI 65043-029LF. Pin 1 is indicated by a square solder pad.

Pin 15	Pin 13	Pin 11	Pin 9	Pin 7	Pin 5	Pin 3	**Pin 1**

Pin 16	Pin 14	Pin 12	Pin 10	Pin 8	Pin 6	Pin 4	Pin 2

Table 6: CN5 Pin-out

SYNC0-	2	1	SYNC0+
GND	4	3	GND
SYNC1-	6	5	SYNC1+
GND	8	7	GND
SYNC2--	10	9	SYNC2+
GND	12	11	GND
SYNC3-	14	13	SYNC3+
GND	16	15	GND

Other Connectors

CN4 and CN9 are for Factory Use only

3.3.3 JUMPERS

JP1: User ID Jumper

The User ID Jumper is a four position, user defined jumper block. It can be used to uniquely identify multiple boards in the system. The jumper settings can be read from the USER_ID register. An installed jumper results in a logic high, and an open jumper results in a logic low.

Table 7: JP1 User ID Jumper

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

3.3.4 LEDS

Status LED

This LED provides a visible indicator of the clock generator status.

Green

A green LED means that the clock generator has a PLL lock with the on-board reference.

Cyan

A cyan LED means that the clock generator has a PLL lock with one of the external references.

Red

A red LED means that there was an issue the clock generator. Try performing a global board reset (GBC_BRD_RST). If this doesn't fix the issue, the module will need to be power cycled.

ADCn Enable LED

This LED is green when a ADC channel is enabled.

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.

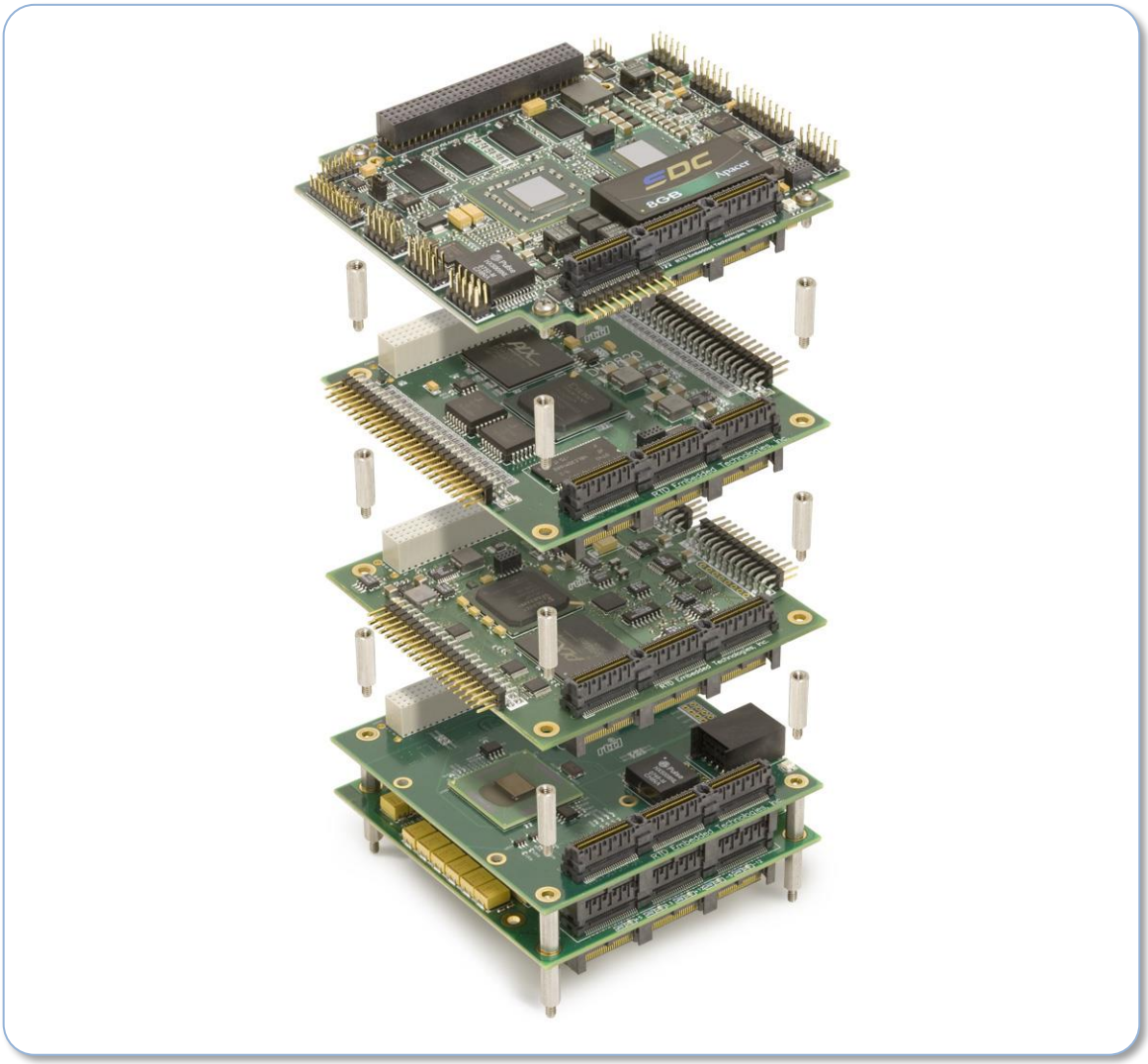


Figure 5: Example 104™ Stack

4 IDAN Connections

4.1 Module Handling Precautions

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

4.2 Physical Characteristics

- Weight: Approximately TBD Kg (TBD lbs.)
- Dimensions: 152 mm L x 130 mm W x 24.5 mm H (5.98 in L x 5.12 in W x 0.97 in H)

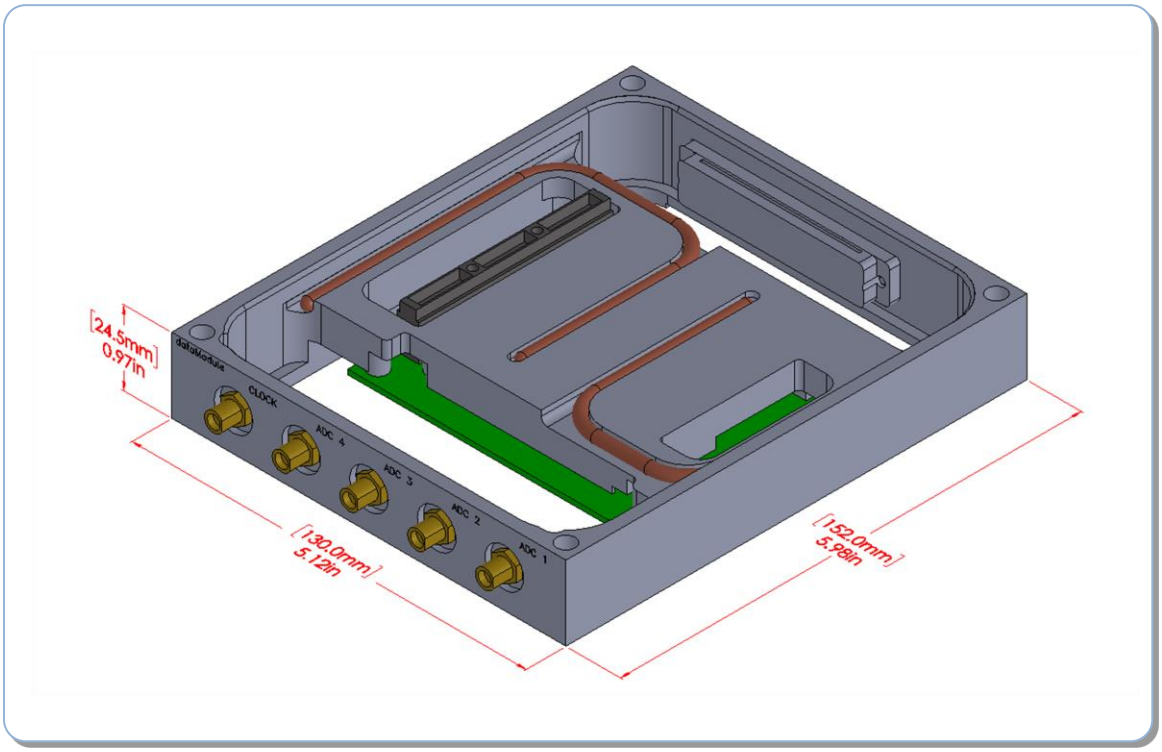
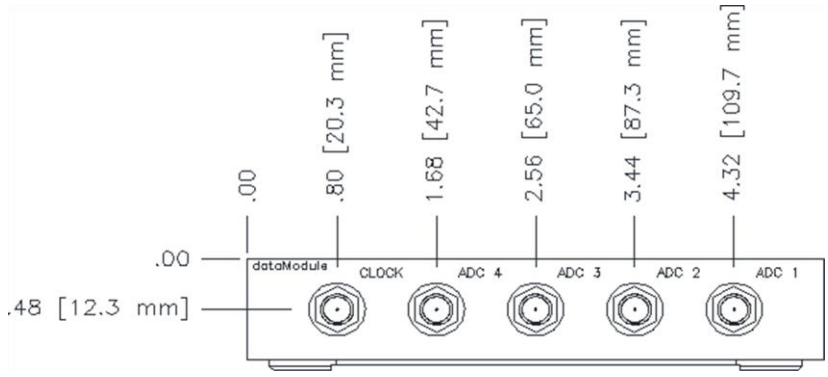


Figure 6: IDAN Dimensions

4.3 Connectors

4.3.1 External I/O Connectors

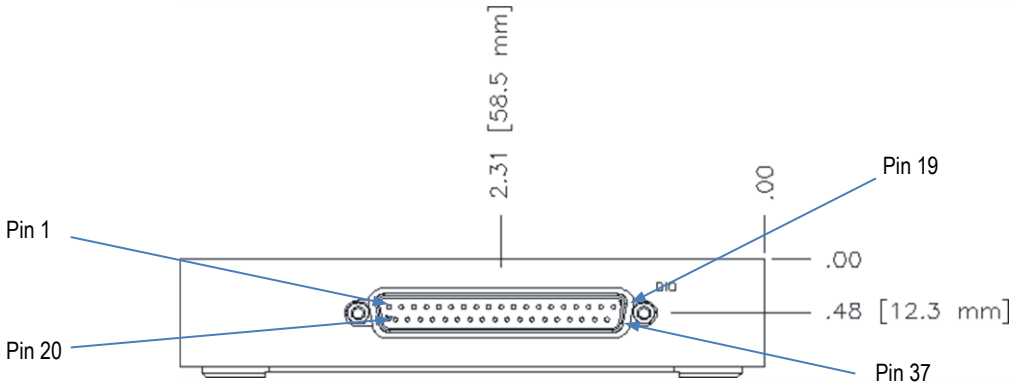
Analog Input/Output Connectors – SMA Female Connector



Connector Part #: SAMTEC RF174-03RP1-01BJ1-0114

Example Mating Connector: Amphenol 901-9877-RFX

Digital I/O Connector - 37-pin High Density "D" Female Connector



Connector Part #: AMP 1658610-1

Example Mating Connector: TYCO 1658608-1

Table 8: IDAN- DM34116HR 37-Pin High Density "D" Connector

IDAN Pin#	Signal	DM34116HR Pin #	
1	DIO0.0	CN4	1
2	DIO0.2	CN4	3
3	DIO0.4	CN4	5
4	DIO0.6	CN4	7
5	DIO0.8	CN4	9
6	DIO0.10	CN4	11
7	DIO0.12	CN4	13
8	DIO0.14	CN4	15
9	GND	CN4	17
10	DIO0.16	CN4	19
11	DIO0.18	CN4	21

Table 8: IDAN- DM34116HR 37-Pin High Density "D" Connector

IDAN Pin#	Signal	DM34116HR Pin #	
12	DIO0.20	CN4	23
13	DIO0.22	CN4	25
14	DIO0.24	CN4	27
15	DIO0.26	CN4	29
16	DIO0.28	CN4	31
17	DIO0.30	CN4	33
18	GND	CN4	35
19	+5V	CN4	37
20	DIO0.1	CN4	2
21	DIO0.3	CN4	4
22	DIO0.5	CN4	6
23	DIO0.7	CN4	8
24	DIO0.9	CN4	10
25	DIO0.11	CN4	12
26	DIO0.13	CN4	14
27	DIO0.15	CN4	16
28	GND	CN4	18
29	DIO0.17	CN4	20
30	DIO0.19	CN4	22
31	DIO0.21	CN4	24
32	DIO0.23	CN4	26
33	DIO0.25	CN4	28
34	DIO0.27	CN4	30
35	DIO0.29	CN4	32
36	DIO0.31	CN4	34
37	GND	CN4	36

4.3.2 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 58)

The DM34116HR is a Type 2 board, and can only connect to either Type 2 PCIe/104 connector.

4.4 Steps for Installing

1. Always work at an ESD protected workstation and wear a grounded wrist-strap.
2. Turn off power to the IDAN system.
3. Remove the module from its anti-static bag.
4. Check that pins of the bus connector are properly positioned.
5. Check the stacking order; make sure all of the busses used by the peripheral cards are connected to the cpuModule.
6. Hold the module by its edges and orient it so the bus connector pins line up with the matching connector on the stack.
7. Gently and evenly press the module onto the IDAN system.
8. If any boards are to be stacked above this module, install them.
9. Finish assembling the IDAN stack by installing screws of an appropriate length.
10. Attach any necessary cables to the IDAN system.
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.

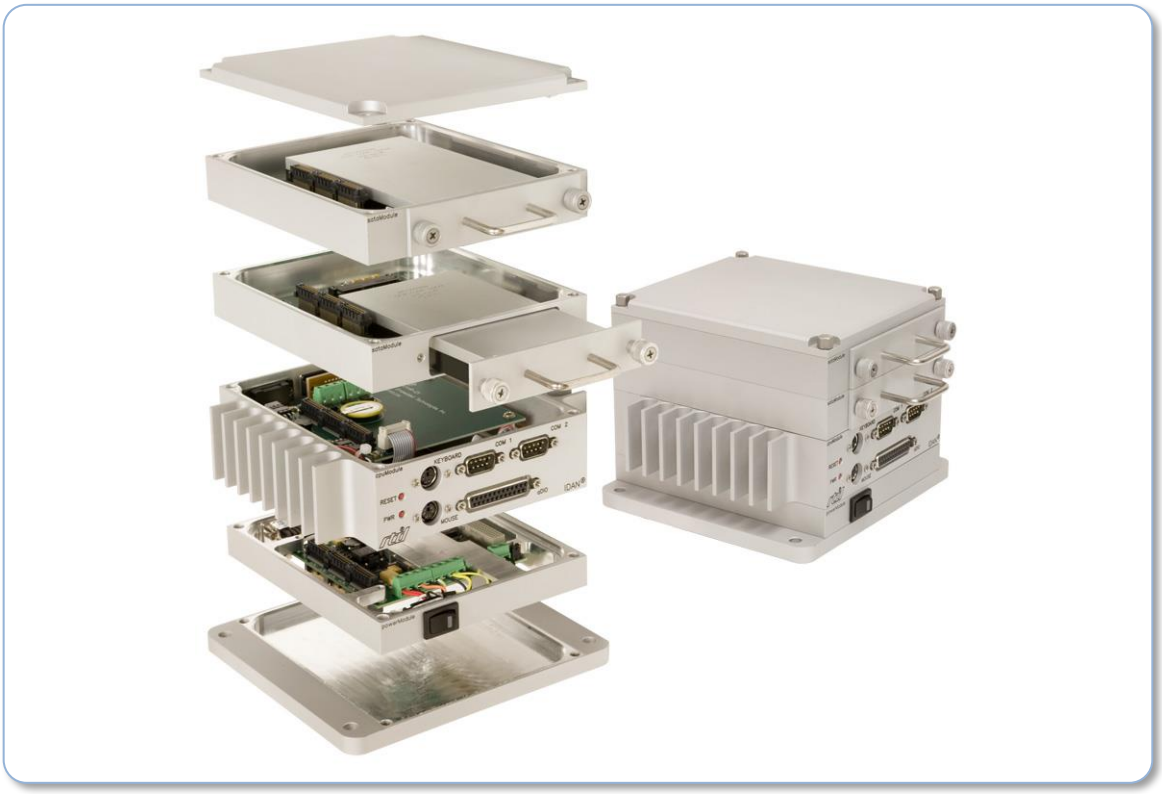


Figure 7: Example IDAN System

5 Functional Description

5.1 Block Diagram

The Figure below shows the functional block diagram of the DM34116HR. The various parts of the block diagram are discussed in the following sections.

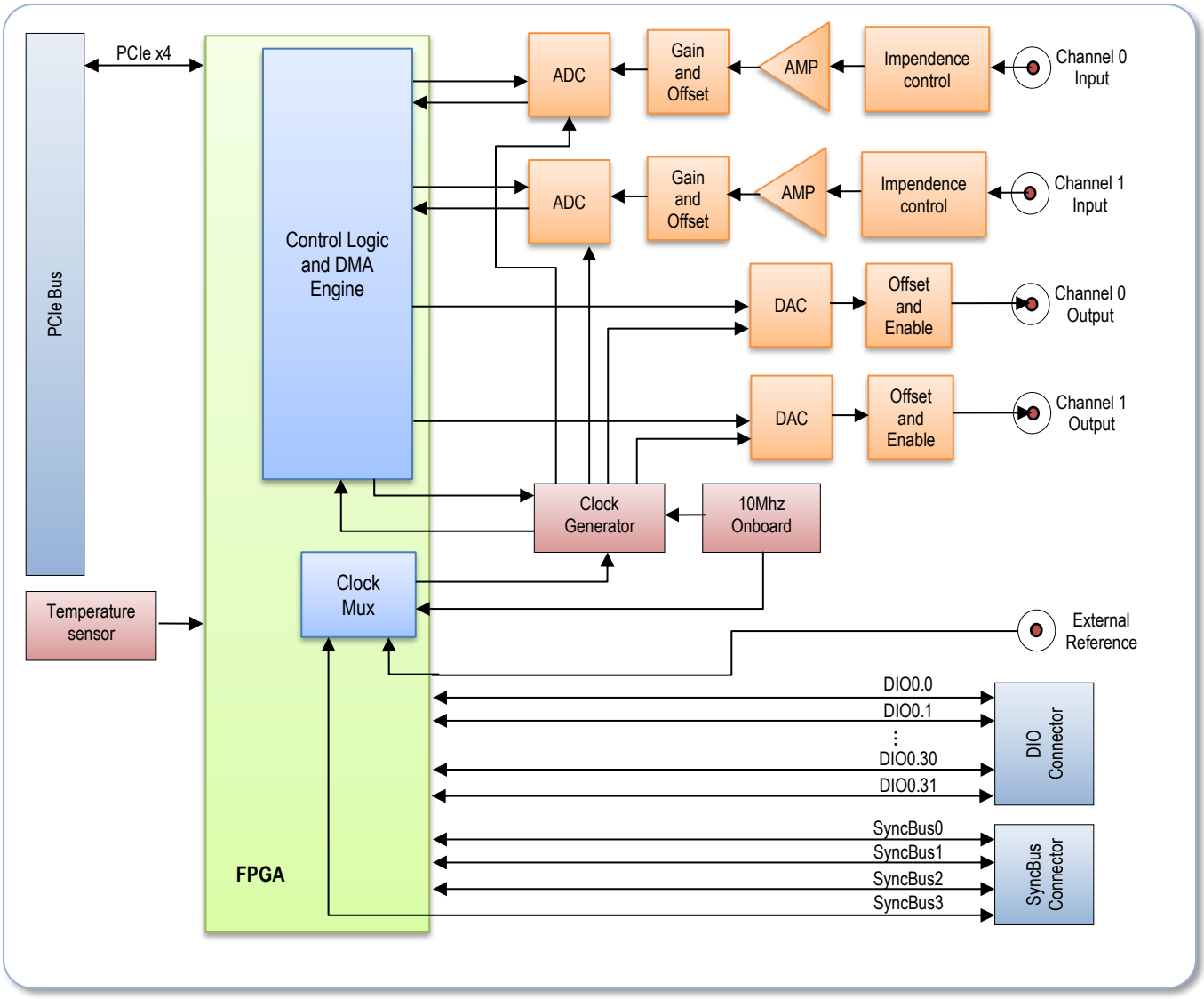


Figure 8: DM34116HR Block Diagram

5.2 FPGA with DMA Engine

The DM34116HR features a FPGA with a built in PCI Express interface and DMA engine. The FPGA controls all communication between the bus and the control logic on the board. The FPGA also features small FIFOs for use with DMA, which is needed for continuous data transfer. Each ADC is provided with its own FIFO and DMA channel, allowing them to transfer data independent of one another.

Each DMA channel can be programmed to transfer data from FPGA to PCIe bus or from the PCIe bus to the FPGA. Each DMA channel also features a 64-bit PCI addressing and a 16MB maximum buffer for memory accessing.

5.3 Programmable Clock

The DM34116HR features 2 programmable clock functional block.

The programmable clocks use a master clock and divide it down by an integer. The result of this clock can then be placed on the CLK_BUS and be used as a clock source or used as a stop or start trigger for any other functional blocks.

The programmable clock functional block also features an interval counter and event counter register. These can be used to capture how many events are occur on a particular CLK_BUSn and how many events occur at a particular interval of the system clock.

Dividing a clock using the Programmable Clock

The following is a list of typical steps used to divide a clock with the programmable clock and assign it to clock bus.

1. Set the Programmable Clock function block to the Uninitialized state (**Mode_Status**)
2. Set the start and stop triggers (**START_TRIG**, **STOP_TRIG**)
3. Set the pacer clock source and frequency (**CLK_DIV**, **CLK_SRC**)
4. Assign Count to source one of the clock bus (**CLK_BUSn**)
5. Set the Programmable Clock function block to the Reset state (**Mode_Status**)
6. Start the Programmable Clock function block (**Mode_Status**)

5.4 SyncBus

This module features an LVDS SyncBus. The SyncBus provides the user with three event-based triggering lines and one 10MHz based master clock. All four are software configurable for input/output. To use any of the SyncBus lines the module must be configured to use the SyncBus master clock.

SyncBus0 – SyncBus2 lines on CN11 are the user event triggering. When these lines are assigned as output, they can output any of the clocks find in **SB_n_CLK_SRC** section on page 74. When lines are assigned as an input, they can be assigned as a **CLK_BUSn** register on page 74, which can be then used to trigger events.

SyncBus3 on CN11 is the master clock. This clock is used to synchronize multiple boards featuring the LVDS SyncBus. To synchronize multiple boards, you must set one board as an output and the rest as inputs. The board that is set as an output, the master will drive a 10MHz clock on SyncBus3 to synchronize all the boards set as inputs, or slaves. Once boards are synchronized the **PLL_LOCKED** register will be set to '1' indicating it is now stable and ready to use.

NOTE: Only enable one output per SyncBus line when synchronizing multiple boards. Enabling multiple outputs on a SyncBus line will result in unexpected behavior.

5.4.1 Clock Generator Sync

The DM34116HR provides a method to synchronize the clock generator on multiple DM34116HR modules using the syncbus. This allows the ADC on multiple modules to be synchronize within one **System Clock** cycle. This process should be performed once all ADC frequencies have been set.

NOTE: If you change any frequency or lose PLL_LOCKED on any module, once the modules have been synchronized, you will have to resynchronize the system.

Synchronizing Multiple DM34116HR

The following is a list of typical steps needed to perform the synchronization process.

1. Connect SyncBus0 between all modules.
2. Connect SyncBus3 between all modules.

3. Set the ADC's sampling clock across all modules (**CLK_DIV**)
4. Set the reference clock on the master module (**CLK_SEL**)
    - Note: This should not be set to the SyncBus of the connector in use
5. Set the reference clock on the slave modules as the SyncBus of the connector in use. (**CLK_SEL**)
6. Set the direction of SyncBus0 and SyncBus3 as outputs on the master module (**DIRECTION**)
7. Set the direction of SyncBus0 and SyncBus3 as inputs on all the slave modules (**DIRECTION**)
8. Set the SYNC_Enable bit on all the modules (**PLL_SYNC**)
9. Set the Go/Busy bit on the master module (**PLL_SYNC**)
10. Check the Go/Busy bit on all modules, to tell when the sync has finished. (**PLL_SYNC**)

5.5 Capture Window

The DM34116HR features a configurable capture window function block. The function block provides four capture and delay registers to setup the capture/non-capture of window. This gives the user the ability to setup unique pattern for data capture. For example, a user could use the capture window for a pulse repetition interval signal.

Capturing ADC data with Capture Window

The following is a list of typical steps needed to setup the capture window function block to capture

1. Set the capture and delay counter registers (**CAPTURE_COUNTER_n** , **DELAY_COUNTER_n**)
2. Set the capture window configuration (**CONFIG**)
3. Set the ADC input mode and enable the channel (**FRONT_END_CONFIG**)
4. Set the capture window function block to the Uninitialized state (**Mode_Status**)
5. Set the ADC function block to the Uninitialized state (**Mode_Status**)
6. Set the capture window's start trigger (**START_TRIG**)
7. Assign Capture Window to source one of the clock bus (**CLK_BUSn**)
8. Set the ADC start and stop trigger to the corresponding clock bus line of capture window (**START_TRIG**, **STOP_TRIG**)
9. Set the capture window's pacer clock source and frequency (**CLK_SRC**, **CLK_DIV**)
10. Set the ADC's pacer clock frequency (**CLK_DIV**)
11. Set the capture window to the Reset state (**Mode_Status**)
12. Set the ADC to the Reset state (**Mode_Status**)
13. Start the ADC function block (**Mode_Status**)
14. Start the capture window function block (**Mode_Status**)

5.6 Advanced Digital I/O

The DM34116HR provides 32-bits of bi-directional digital I/O. Each bit is programmable as either an input or output. On power cycle, all digital I/O lines are programmed as inputs, meaning that the digital I/O line's initial state is undetermined. If the digital I/O lines must power up to a known state, an external 10 kΩ resistor must be added to pull the line high or low. The advanced Digital I/O also supports DMA, advanced interrupts, and peripheral select.

DMA Channel

The DM34116HR provides a DMA channel for each digital input, digital out and direction. Each channel has 511 sample FIFO for DMA. Each sample is packed into 32-bits.

Advanced Interrupts

The DM34116HR has an advanced interrupt block that can generate an interrupt on a match, event or strobe. The interrupts are across all 32-digital input of the function block. The bits can be individually selected.

When an interrupt is generated, the data on all the ports is latched into the Capture registers. Bits are tested regardless of if a pin is an input or output.

A Match interrupt is generated when all un-masked bits in the Compare register match the input value of the port. An Event interrupt is generated when any un-masked input port bit changes. A strobe interrupt is generated every event of the **CLK_SRC** for the function block.

Peripheral Select

The DM34116HR has multipurpose output pins. The outputs can be configured as standard outputs, PWM, or external clocks. The outputs are configurable by bit to maximum flexible of all the features.

Please refer to PERIPH_SEL_BIT_n section on page 55 on how to configure the register for a particular output mode.

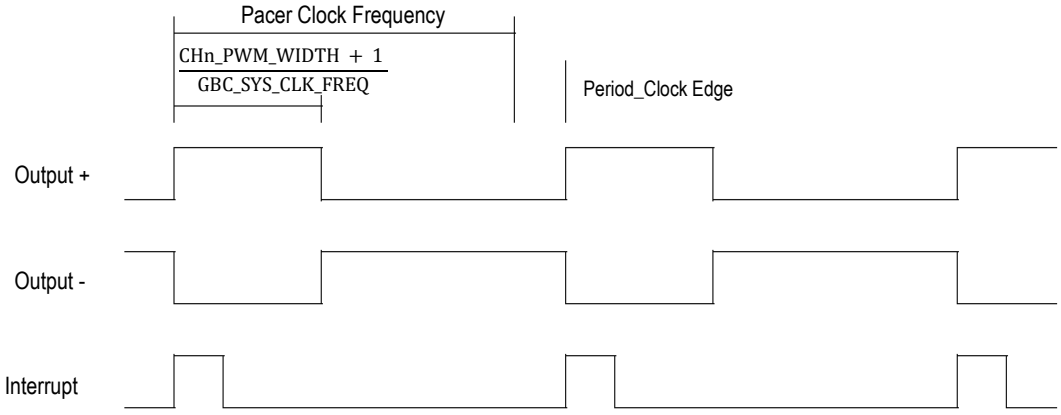
5.6.1 QUAD PULSE WIDTH MODULATOR

The DM34116HR features a quad PWM output functional block with DMA. The PWM is available with the output pin peripherals are configured as PWM.

All the PWM outputs have the same period. The pulse width of each of the four outputs is individually adjustable.

An interrupt is generated at the beginning of every period.

The width register is checked at the beginning of every period. If the width register is modified in the middle of a period, the output will not be affected until the next period.



Initializing the PWM

The following is a list of typical steps needed to initialize the PWM outputs.

1. Set the corresponding pins as input in the Advanced_DIO function block (**CH2_DATA**)
2. Set the PWM function block to the Uninitialized state (**Mode_Status**)
3. Setup the DMA for the PWM channel
4. Setup the channel's width (**CHn_PWM_WIDTH**)
5. Set the start and stop triggers (**START_TRIG**, **STOP_TRIG**)
6. Set the pacer clock (**CLK_DIV**)
7. Set the PWM function block to the Reset state (**Mode_Status**)
8. Start the DMA
9. Start the PWM function block (**Mode_Status**)

5.6.2 INCREMENTAL ENCODER

The DM34116HR features a 32-bit incremental encoder functional block with DMA.

An Incremental Encoder is used to detect the relative position of a shaft or linear actuator. A typical implementation is a slotted wheel with two optical sensors positioned such that when one sensor is positioned over a slot, the other is positioned between slots. The output of the optical sensors is shown in Figure 9, with one sensor named "A," and the other named "B." At every edge of the "A" or "B" input, the counter either increments or decrements. The direction can be interpreted from the state of the signals, i.e. which signal leads.

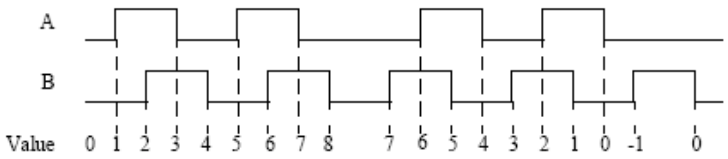


Figure 9: Incremental Encoder Signals

The encoders include a "Phase Filter" that prevents the counter from counting on certain transitions. This allows the encoders to count pulses, and other specialized applications.

Encoder inputs can be configured as single ended or pseudo-differential. In pseudo-differential mode, the "+" and "-" inputs must be the inverse of each other in order for the encoder to see a change.

Separate interrupts are generated for positive and negative rollover. Positive rollover occurs when the counter is at its maximum value and receives a signal to count up. Negative rollover occurs when the counter is at 0 and receives a signal to count down. Because separate interrupts are generated, the counter can be easily expanded in software.

The Incremental Encoder inputs are show in the table below.

Table 9: Incremental Encoder Inputs

Pin	CN3 Pin #
A+	13
A-	14
B+	15
B-	16
Index +	19
Index -	20

Initializing the Incremental Encoder

The following is a list of typical steps needed to initialize the incremental encoder inputs.

1. Set the corresponding pins as input in the Advanced_DIO function block (**CH2_DATA**)
2. Set the corresponding pins' peripheral select to 0x00 (PERIPH_SEL_BIT_n)
3. Set the Incremental Encoder function block to the Uninitialized state (**Mode_Status**)
4. Setup the DMA for the Incremental Encoder channel
5. Pre-load with a known position (**INCENC_VALUE**)
6. Set the mode/configuration (**MODE_CONFIG**)
7. Set the start and stop triggers (**START_TRIG, STOP_TRIG**)
8. Set the pacer clock (**CLK_SRC**)
9. Set the Pre and/or Post Capture counters (**PRE_START_COUNT**, POST_**STOP_COUNT**)
10. Set the Incremental Encoder function block to the Reset state (**Mode_Status**)
11. Start the DMA
12. Start the Incremental Encoder function block (**Mode_Status**)

5.6.3 External Clocking

The DM34116HR features an external clocking function block. This feature allows the user to input a clock to drive a CLK_BUSn signal or output a CLK_BUSn signal. The CLK_SRCn are used to drive the CLK_BUSn signal which are part of the most FPGA function blocks. The CLK_BUSn signals are used as either sample clocks for function blocks or triggers for starting and stopping them.

There are 6 available CLK_BUSn, each is associated with a pin on CN3. Each can be configured to be either an input or an output. As an input to a CLK_BUSn, the max input clock frequency is ½ system clock frequency. This value can be found in **GBC_SYS_CLK_FREQ** on page 36. As an output CLK_BUSn will generated on the associated pin. By default, this signal will be a pulse that is high for 10ns when the CLK_BUSn signal goes high. The width of this pulse can be increased using EXT_CLKn_PW on page 66.

Inputting an External Signal to the Clock Bus

The following is a list of typical steps needed to perform to input an external signal to the clock bus.

1. Connect the external signal to one of the external clocking pins EXT_CLK_2- EXT_CLK_7
2. Connect the gated signal to corresponding EXT_CLK_GATE_N, if gated signal is being used.
3. Set the corresponding pin as input in the Advanced_DIO function block (**CH2_DATA**)
4. Set the corresponding pin's peripheral select to 0x00 (PERIPH_SEL_BIT_n)
5. Set the corresponding clock as input in the External Clocking function block (**EXT_CLK_DIR**)
6. Set the corresponding clock's edge detect (**EXT_CLK_EDGE**)
7. Enable the external clock (**EXT_CLKn_CFG**)

Outputting a Signal from the Clock Bus

The following is a list of typical steps needed to perform to output an external signal from the clock bus.

1. Connect to one of the external clocking pins EXT_CLK_2- EXT_CLK_7
2. Set the corresponding pin as output in the Advanced_DIO function block (**CH2_DATA**)
3. Set the corresponding pin's peripheral select to 0x02 (PERIPH_SEL_BIT_n)
4. Set the corresponding clock as output in the External Clocking function block (**EXT_CLK_DIR**)
5. Adjust the pulse width of the EXT_CLKn if needed (**EXT_CLKn_PW**)
6. Enable the external clock (**EXT_CLKn_CFG**)

5.7 Temperature Sensor

This module provides an on-board temperature sensor. This sensor provides the module's ambient temperature. Refer to **Board Connections** figure on page 16 for temperature sensor location.

The temperature sensor provides a signed 12-bit value that capable of measuring temperatures within ±2°C. The temperature sensor is read continuously, approximately every 1 to 2 seconds. This value is then automatically updated in the register. Refer to **TEMPERATURE_VAL** section on page 67 for more register information.

5.8 Analog input

The DM34116HR has 2 independent 16-bit Pipeline ADC converters, to provide high speed and high digital resolution of the analog input. The ADC converter has a max throughput of 25MHz per channel. Since the ADC is a pipeline type the corresponding digital code will be delayed by 8 cycles from when taken by the converter. This module provides +6.5V/ -4.5V overvoltage protection to the analog input with the attenuator is disabled.

The DM34116HR has a programmable input. This provides the user the ability to select input impedance, full-scale input range, and unipolar/bipolar input.

Each ADC converter supports a 1023 sample FIFO for DMA. Each sample packed into a 32-bit word.

Full-Scale Input Range

The DM34116HR has a programmable gain input per channel. This gain can be programed for 0.5, 0.75, 1, 1.5, 2, 3, 4, 6 to achieve input ranges ±5V, ±3.3V, ±2.5V, ±1.667V, ±1.25V, ±0.833V, ±0.625V, ±0.416V.

Refer to Table 10 on page 31 for full-scale ranges for both unipolar and bipolar modes.

Bipolar/Unipolar Mode

In bipolar mode the data is collected in two's complement format. In this format the MSB of the data is the sign bit. If the MSB is "0" the output data is positive value. If the MSB is "1" the output code is negative. The 16-bits of data is sign-extended into a 32-bit word before transferring to the FIFO.

In unipolar mode there is no sign bit, since all values are positive. When the output code is all "0" the input voltage is 0 volts. When the output code is all "1" the input voltage is 5 volts when the gain is set to 1.

Voltage values for each bit will vary depending on input range and gain. For example, if the input is set for ±5V, the formula for calculating voltage is as follows:

$$Voltage = \frac{Input\ Range}{2^{16}} \times Conversion\ Data$$

$$Voltage = \frac{10}{2^{16}} \times Conversion\ Data$$

$$Voltage = 152.587\mu V \times Conversion\ Data$$

5.8.1 INITIALIZING THE ADC CONVERTER

The following is a list of the typical steps needed to initialize the ADC converter and begin sampling:

1. Set the ADC to the Uninitialized state (Mode_Status)
2. Setup the DMA for the channel
3. Set the input mode (FRONT_END_CONFIG)
4. Set the start and stop triggers (START_TRIG, STOP_TRIG)
5. Set the sample rate (CLK_DIV)
6. Set the Pre and/or Post Capture counters (PRE_TRIGGER_CAPTURE, POST_STOP_CAPTURE)
7. Set the ADC to the Reset state (Mode_Status)
8. Start the DMA
9. Start the ADC (Mode_Status)

5.8.2 SIMPLIFIED BLOCK DIAGRAM OF ANALOG INPUT

The following figure shows the front end circuit for the DM34116HR. It also shows the names of the FPGA registers in bold and the different ways the front-end can be configured for different modes of operation.

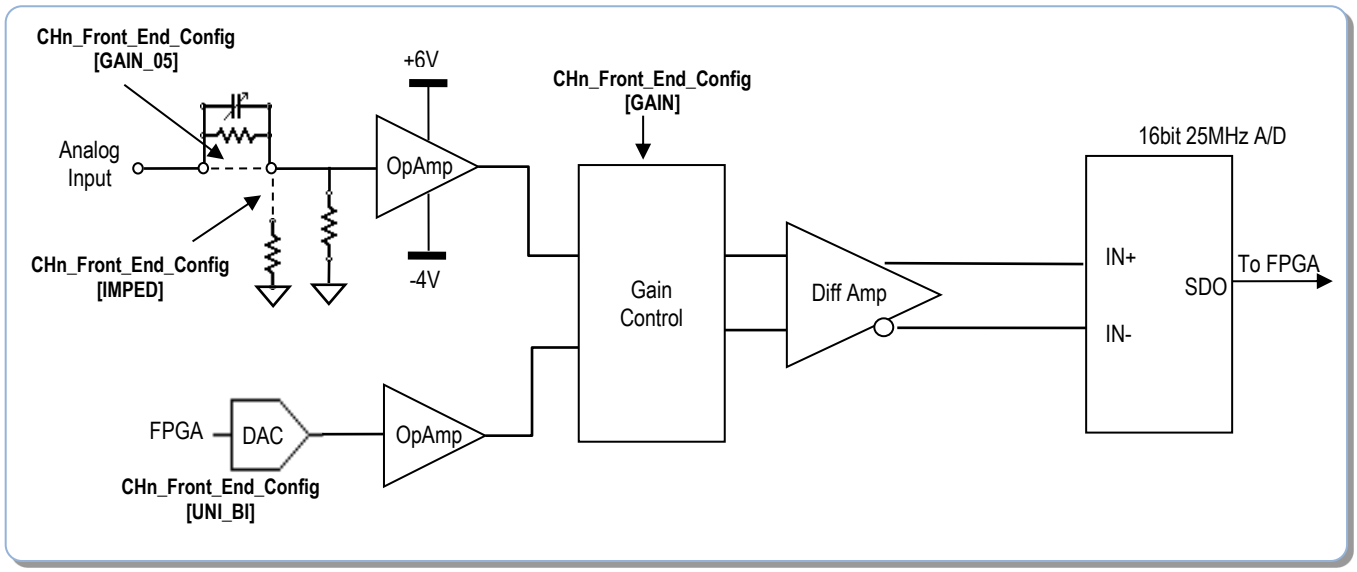


Figure 10: Analog Front End

Table 10: ADC Full-Scale Settings

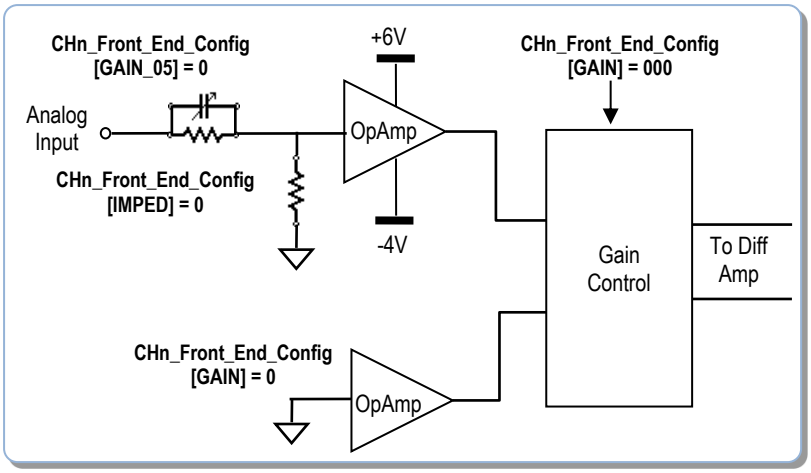
CHn_Front_End_Config [GAIN]	CHn_Front_End_Config [GAIN_05]	CHn_Front_End_Config [IMPED]	Signal Path Gain	Unipolar Mode	Bipolar Mode
000	0	0	0.5	n/a	±5V
001	0	0	0.75	n/a	±3.3V
100	0	0	1	0-5V	±2.5V
101	0	0	1.5	0-3.3V	±1.667V
001	1	0/1	1.5	0-3.3V	±1.667V
100	1	0/1	2	0-2.5V	±1.25V
101	1	0/1	3	0-1.67V	±0.833V
110	1	0/1	4	0-1.25V	±0.625V
111	1	0/1	6	0-833V	±0.4167V

Refer to section 6.4.17 for more information about FPGA register control

5.8.3 INPUT CONNECTION EXAMPLES

5.4.3.1. High Impedance, 5V Bipolar Input

The following figure shows an example how to set up the FPGA registers for a high impedance, bipolar signal with a gain of 0.5. Also provided are graphs showing the difference between the input vs. output to the differential amp.



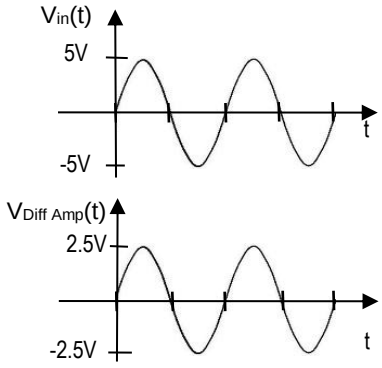


Figure 11:High Impedance Bipolar Example

5.4.3.3. High Impedance, Unipolar, 0-3.3V Input

The following figure shows an example how to set up the FPGA registers for a single-ended, unipolar signal with a gain of 1. Also provided are graphs showing the difference between the input vs. output to the differential amp.

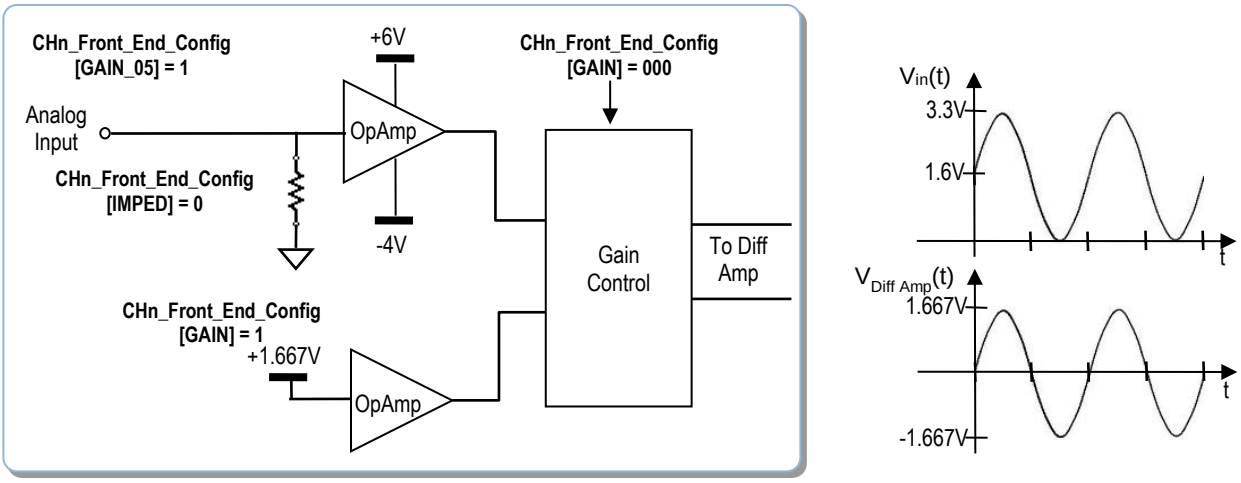


Figure 12:High Impedance Unipolar Example

5.9 Analog Output

The DM34116HR has 2 independent 16-bit DAC converters, to provide high speed analog output. The DAC converter has a max update rate of 25MHz per channel. The output op amp driver provides the ability to drive a 50-ohm system over the full voltage range.

DAC Output Range

The DM34116HR DAC output has range of ±2.5V. However, the module features an additional DAC that allows you to adjust the offset of high speed DAC output. The min/max voltage out of the output buffer is ±4.5V.

5.9.1 Initializing the DAC Converter

The following is a list of the typical steps needed to initialize the DAC converter and begin sampling:

1. Set the ADC to the Uninitialized state (Mode_Status)
2. Setup the DMA for the channel
3. Set the start and stop triggers (START_TRIG, STOP_TRIG)
4. Set the sample rate (CLK_DIV)
5. Set the Post Conversion counters (POST_STOP_CONVERSIONS)
6. Set the DAC to the Reset state (Mode_Status)
7. Enable the DAC output (FRONT_END_CONFIG)
8. Start the DMA
9. Start the DAC (Mode_Status)

5.9.2 SIMPLIFIED BLOCK DIAGRAM OF ANALOG OUTPUT

The following figure shows the output circuit for the DM34116HR. It also shows the names of the FPGA registers in bold and how the output can be configured.

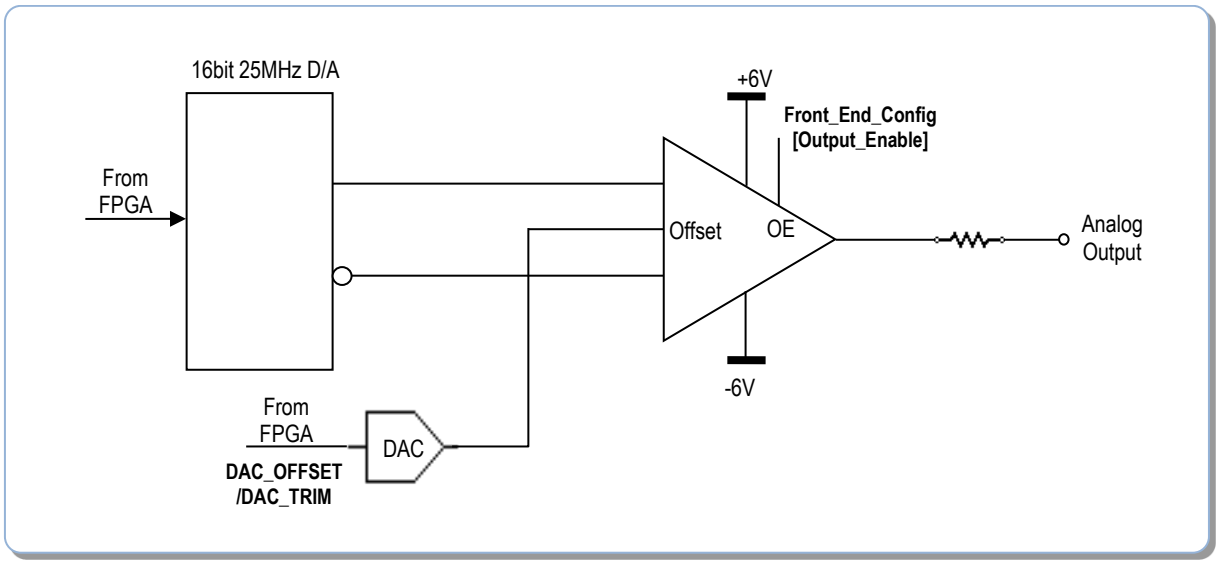


Figure 13: Output Circuit

DAC Bit Weight	Ideal Output Voltage (mV)
0111 1111 1111 1111	2499.9237
0000 0000 0000 0001	0.0763
0000 0000 0000 0000	0
1111 1111 1111 1111	-0.0763
1000 0000 0000 0000	-2500.00

Refer to section 6.5.18 for more information about FPGA register control

6 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 DM34116HR.

Table 11: Function Block Mapping

Slot	ID	Description	Connector
FB0	0x01011001	Analog to Digital Converter	CN6
FB1	0x01011001	Analog to Digital Converter	CN7
FB2	0x01011001	Analog to Digital Converter	CN8
FB3	0x01011001	Analog to Digital Converter	CN9
FB4	0x01003010	Advanced DIO	CN3
FB5	0x01003100	PWM	CN3
FB6	0x01003200	Incremental Encoder	CN3
FB7	0x00000003	External Clocking	CN3
FB8	0x0000F001	Temperature Sensor	N/A
FB9	0x0000F003	Clock Generator	N/A
FB10	0x00000005	Capture Window	N/A
FB11	0x00010001	SyncBus	CN11
FB12	0x00010001	SyncBus	CN16
FB13	0x00000004	Programmable Clock	N/A
FB14	0x00000004	Programmable Clock	N/A
FB15	0x0000F002	Reserved	N/A

6.1 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 12: 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			

6.1.1 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.

6.1.2 GBC_REV (READ-ONLY)

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

6.1.3 GBC_BRD_RST (READ/WRITE)

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

6.1.4 GBC_PDP (READ-ONLY)

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

6.1.5 GBC_BUILD (READ-ONLY)

Unique 32-bit build number for every FPGA build.

6.1.6 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.

6.1.7 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.

6.1.8 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 6.1.10 GBC_EOI (Read/Clear) below) any "hidden" interrupts will be generated again.

6.1.9 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 **6.2** on page 37.

6.1.10 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).

6.1.11 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 34. 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

6.1.12 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.

6.1.13 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.

6.2 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 13: 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)				

6.2.1 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.

6.2.2 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.

6.2.3 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.

6.2.4 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.

6.2.5 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.

6.2.6 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.

6.2.7 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.

6.2.8 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.

6.2.9 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.

6.2.10 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.

6.2.11 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

6.2.12 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

6.2.13 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).

6.2.14 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.

6.2.15 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.

6.2.16 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.

6.3 BAR2: Functional Block Header Registers

Each Functional Block has a standard header as follows.

Table 14: Functional Block Header

Offset	0x03	0x02	0x01	0x00
FB + 0x00	FB_ID			
FB + 0x04	FB_DMA_BUFFERS	FB_DMA_CHANNELS	FB_CONTROL_SIZE	FB_CHANNEL_SIZE

6.3.1 FB_ID (READ-ONLY)

This is a repeat of the ID from the General Board Control section. This can be used to validate that the correct Function Block is being accessed.

6.3.2 FB_DMA_CHANNELS (READ -ONLY)

This register contains the number of DMA Channels in this Function Block.

6.3.3 FB_DMA_BUFFERS (READ-ONLY)

This register contains the number of Buffer Descriptors in each DMA Channel.

6.3.4 FB_CONTROL_SIZE (READ -ONLY)

This register contains the total number of DWORD registers in the control section of the function block. This should be used to calculate the offset for each register in a channel.

NOTE: This register is not implemented on every function block. You must check each function block's register map to see if this register is defined.

6.3.5 FB_CHANNEL_SIZE (READ-ONLY)

This register contains the total number of DWORD registers in each channel of the function block. This should be used to calculate the offset for each register in a channel.

NOTE: This register is not implemented on every function block. You must check each function block's register map to see if this register is defined.

6.4 BAR2: Analog to Digital Converter (ADC)

This Function Block is for the Analog to Digital converter. The DM34116HR has two ADC Function blocks, one for each of the ADC inputs. The ADC0 input is associated with the lower-numbered ADC Function Block, and ADC1 is associated with the higher-numbered ADC Function Block. Each Function Block has its own clock, triggers, and DMA channel.

6.4.1 Function Block Register Map

Table 15: High Speed A/D Functional Block

	Offset	0x03	0x02	0x01	0x00
Header	FB + 0x00	FB_ID			
	FB + 0x04	FB_DMA_BUFFERS	FB_DMA_CHANNELS	FB_CONTROL_SIZE	FB_CANNEL_SIZE
ADC Control	FB + 0x08	STOP_TRIG	START_TRIG	*Reserved*	MODE_STATUS
	FB + 0x0C	CLK_READY	*Reserved*	CLK_DELAY	CLK_DIV
	FB + 0x10	*Reserved*			
	FB + 0x14	PRE_TRIGGER_CAPTURE (limited by FIFO size)			
	FB + 0x18	POST_STOP_CAPTURE			
	FB + 0x1C	SAMPLE_CNT			
	FB + 0x20	INT_ENA (Sample, Start, Stop, Threshold, Pacer Tick, etc)			
	FB + 0x24	INT_STAT		*Reserved*	
	FB + 0x28	CLK_BUS3	CLK_BUS2	*Reserved*	*Reserved*
	FB + 0x2C	CLK_BUS7	CLK_BUS6	CLK_BUS5	CLK_BUS4
	FB + 0x30	AD_CONFIG (Maskable register – 16-bit)			
	FB + 0x34	ADC_TRIM_BUSY		ADC_TRIM	
ADC Channel	FB + 0x08 + (FB_CONTROL_SIZE * 0x04)	FRONT_END_CONFIG (Maskable register – 16-bit)			
	FB + 0x0C + (FB_CONTROL_SIZE * 0x04)	FIFO_DATA_CNT			
	FB + 0x10 + (FB_CONTROL_SIZE * 0x04)	MAX_FIFO_SIZE			
	FB + 0x14 + (FB_CONTROL_SIZE * 0x04)	INT_ENA	INT_STAT	FILTER	*Reserved*
	FB + 0x18 + (FB_CONTROL_SIZE * 0x04)	THRESH_LOW			
	FB + 0x1C + (FB_CONTROL_SIZE * 0x04)	THRESH_HIGH			
	FB + 0x20 + (FB_CONTROL_SIZE * 0x04)	LAST_SAMPLE			
	FB + 0x24 + (FB_CONTROL_SIZE * 0x04)	FIFO_ACCESS			

6.4.2 Mode_Status (Read/Write, Read-Only)

Selects the current mode of operation and indicates its triggering status.

B[3:0]: Mode

- 0x04: Uninitialized. This is the power-on state. No converter initialization has taken place. Sampling is stopped, and all counters are reset, and the triggering state machine is reset. Transition to any of the other Modes will start converter initialization (sampling will not start until initialization is complete).
- 0x00: Reset. Sampling is stopped. All counters are reset, and the triggering state machine is reset.
- 0x01: Paused. Sampling is stopped, but the counters and triggering state machine maintain their state.
- 0x02: Go, Single-Shot. After filling the buffer with the Post-Stop samples, capturing stops. The Mode must be set back to RESET in order to capture more samples.
- 0x03: Go, Re-arm. After filling the buffer with the Post-Stop samples and the FIFO is empty, the triggering state machine is restarted, i.e. FIFO is filled with Pre-Start samples and waits for a start trigger.

B[7:4]: Status

- 0x08: Uninitialized – The status when in the “Uninitialized” mode and the converter requires initialization.
- 0x09: Initializing
- 0x00: Stopped – The status when in the “Reset” mode, or in the “Uninitialized” mode and the converter does not require initialization.
- 0x01: Filling Pre-Trigger buffer
- 0x02: Waiting for start trigger
- 0x03: Sampling/Waiting for stop trigger
- 0x04: Filling Post-Stop buffer
- 0x05: Wait to re-arm – Waiting until local FIFO is empty, so the pre-trigger buffer can be filled.
- 0x07: Done capturing

6.4.3 START_TRIG (READ/WRITE)

Selects the start trigger from the clock bus. The ADC will start capturing after the start trigger, unless PRE_TRIGGER_CAPTURE is non-zero in which case the ADC will start capturing immediately.

Below is the list of clock sources and the register value needed to select the source.

B[7:0]:

- 0x00: System clock/immediate
- 0x01: Never
- 0x02: CLK_BUS2
- 0x03: CLK_BUS3
- 0x04: CLK_BUS4
- 0x05: CLK_BUS5
- 0x06: CLK_BUS6
- 0x07: CLK_BUS7
- 0x08: Threshold – ADC has exceeded the High or Low threshold.
- 0x09: Threshold Inverted– The ADC is within the High and Low threshold.
- 0x0A: CLK_BUS2 Inverted
- 0x0B: CLK_BUS3 Inverted
- 0x0C: CLK_BUS4 Inverted
- 0x0D: CLK_BUS5 Inverted
- 0x0E: CLK_BUS6 Inverted
- 0x0F: CLK_BUS7 Inverted

6.4.4 STOP_TRIG (READ/WRITE)

Selects the stop trigger from the clock bus.

Refer to START_TRIG section above, or list of valid values.

6.4.5 CLK_DIV(READ/WRITE)

Divider for the sample clock.

$$Sample\ Clock\ Frequency\ = \frac{200\ MHz}{(2\ \times\ CLK_DIV)}$$

Maximum clock rate is 25Mhz, CLK_DIV = 0x04

Minimum clock rate is 1Mhz, CLK_DIV = 0x64

Disabling clock, CLK_DIV = 0x00

6.4.6 CLK_DELAY(READ/WRITE)

This register is used to set a delay relative to the ADC's distribution clock. This delay will only be noticeable affect the ADC output clocks have been synchronized by using the **6.5.4** register of the Clock Generator function block.

$$Sample\ Clock\ Delay\ =\ 150ps\ \times CLK_DELAY$$

Maximum clock delay is 2250ps, CLK_DELAY = 0xF

Minimum clock delay is 0ps, CLK_DIV = 0x0

6.4.7 CLK_READY

This register is used to tell the status of the ADC clock. When changing the CLK_DIV or CLK_DELAY register, checking this register will indicate if the latest values have been set.

B7: Clock status 1 – Clock is ready; the latest values have been loaded0 – Clock is not ready, still setting the latest values

6.4.8 PRE_TRIGGER_CAPTURE (READ/WRITE)

Number of samples to collect before the Start Trigger. The length is limited by the FIFO size – writing a value larger than the FIFO size will have indeterminate results.

6.4.9 POST_STOP_CAPTURE (READ/WRITE)

Number of samples to collect after the Stop Trigger.

6.4.10 SAMPLE_CNT (READ ONLY)

Total number of samples collected. This does not increment in while in the "Waiting For Start Trigger" state. It also continues counting after a Re-Arm.

6.4.11 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.

6.4.12 INT_STAT (READ/CLEAR)

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.

B0: Sample – A sample has been taken.

B1: Threshold – The ADC has exceeded the High or Low threshold. Check the THRESH_STAT register.

B2: Pre-Start Buffer Filled

B3: Start Trigger

B4: Stop Trigger

B5: Post-Stop Buffer Filled

B6: Sampling has completed, and the FIFO is empty (all data transferred to host)

B7: Pacer – The pacer clock has ticked.

6.4.13 CLK_BUSN

Selects a source to drive onto Clock Bus N. That clock bus can then be used by a different function block as a clock source or trigger.

A function block can drive multiple different Clock Buses. However, a Clock Bus N should not be driven by more than one function block at the same time or the clock signal will be undefined.

B[7:0]:

- 0x00: Disables Clock Source
- 0x80: Sample – A sample has been taken.
- 0x81: Threshold –The ADC has exceeded the High or Low threshold. Check the THRESH_STAT register.
- 0x82: Pre-Start Buffer Filled
- 0x83: Start Trigger
- 0x84: Stop Trigger
- 0x85: Post-Stop Buffer Filled
- 0x86: Sampling has completed and the FIFO is empty (all data transferred to host)
- 0x87: Pacer – The pacer clock has ticked.

6.4.14 AD_CONFIG (MASKABLE READ/WRITE)

Not used.

6.4.15 ADC_TRIM (READ/WRITE)

This register is used to zero out the offset of the ADC input. The value is stored as a two's compliment format. Currently there is no way to store this value in the onboard, so this register defaults to 0 at power on.

$$Trim\ Voltage\ = \frac{ADC_TRIM \times\ 5}{4096}$$

6.4.16 ADC_TRIM_BUSY (READ ONLY)

This register is used to tell the status of the ADC trim control. When changing the ADC_TRIM register, checking this register will indicate if the latest values have been set.

B7: Trim status 1 – Ready; the latest values have been set 0 – Busy, still setting the latest values

6.4.17 FRONT_END_CONFIG (MASKABLE READ/WRITE)

Refer to section **5.8** for more information about the front-end circuit.

This register provides configuration to the Front End for the ADC.

B[8]: PACKED_DATA	0 = Data is unpacked	1 = Captured samples are packed (2 samples per 32-bit Sample)
B[7]: UNI_BI	0 = Bipolar Mode	1 = Unipolar Mode
B[6]: DITH	0 = Disable ADC dither	1 = Enables ADC dither
B[5]: IMPED	0 = High Impedance Input	1 = 50Ω Impedance Input
B[4]: GAIN_05	0 = gain *0.5 selected,	1 = gain *1 selected

Note: Gain_05 must be 1 when in 50Ω Impedance Mode

B[3:1]: GAIN

- 000: Gain of 1
- 001: Gain of 1.5
- 100: Gain of 2
- 101: Gain of 3
- 110: Gain of 4
- 111: Gain of 6

B[0]:Shutdown 0 = ADC is enabled 1 = ADC is disabled

6.4.18 FIFO_DATA_CNT (READ)

This register shows the current sample count that is available in the ADC's FIFO.

6.4.19 MAX_FIFO_SIZE (READ)

This register shows the max number of samples that the ADC's FIFO can hold.

6.4.20 FILTER (READ/WRITE)

The programmable digital filter provides a single pole Infinite Impulse Response (IIR) filter on the ADC. This a unity-gain filter. The filtered data has a value of:

$$D_n = \frac{D_{n-1} \times (2^{ORDER} - 1) + NewSample}{2^{ORDER}}$$

The response of the filter is shown in the Figure 14 below. The Table below shows the -3dB cutoff for each of the filter settings. Both the figure and the table are relative to the sample rate (f_s).

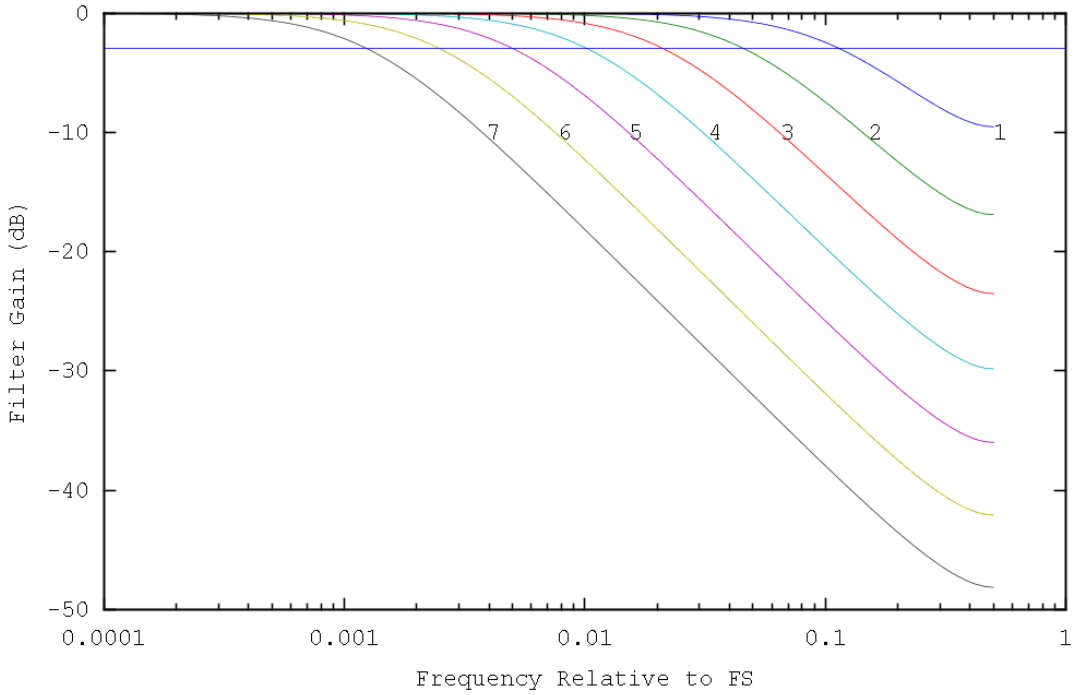


Figure 14: Filter Response with each ORDER Value

ORDER	-3 dB Cutoff
0	n/a
1	0.114791 * f_s
2	0.045995 * f_s
3	0.021236 * f_s
4	0.010255 * f_s
5	0.005042 * f_s
6	0.002501 * f_s
7	0.001246 * f_s

6.4.21 INT_STAT(READ/CLEAR)

This is the status register for the ADC's interrupts. 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.

B0: '1' = Low Threshold has been crossed
B1: '1' = High Threshold has been crossed

6.4.22 INT_ENA (READ/WRITE)

This is the interrupts enable for the threshold detection. Bit defines are above. An interrupt is generated (if not already generated) each time a sample is taken and the value is above the high threshold or below the low threshold.

6.4.23 THRESH_LOW (READ/WRITE)

Signed 32-bit value indicating the low threshold. If the input signal drops below this value, an interrupt or clock can be generated until the signal goes above this value. The 3 least significant bits are ignored from the actual threshold value.

NOTE: The threshold value should not exceed the ADC range. If the threshold value exceeds the ADC range unexpected results will occur.

6.4.24 THRESH_HIGH (READ/WRITE)

Signed 32-bit value indicating the high threshold. If the input signal goes above this value, an interrupt or clock can be generated until the signal goes below this value. The 3 least significant bits are ignored from the actual threshold value.

NOTE: The threshold value should not exceed the ADC range. If the threshold value exceeds the ADC range unexpected results will occur.

6.4.25 LAST_SAMPLE (READ-ONLY)

The last sample read from the ADC Converter, after filtering. This is the same value that is written to the DMA FIFO.

- When PACKED_DATA = 0, one sample stored in this register
- When PACKED_DATA = 1, two samples stored in this register
  - B[15:0]: First Sample
  - B[31:16]: Second Sample

6.4.26 CH_FIFO_ACCESS (READ/WRITE)

This register provides direct access to the DMA FIFO. It can be used to access the data without the use of the DMA engine. The DMA engine for this channel must be set to “Pause.” Each register access advances to the next sample.

6.5 BAR2: Digital to Analog Converter (DAC)

This Function Block is for the Digital to Analog converter. The DM34116HR has two DAC Function blocks, one for each of the DAC outputs. The DAC0 input is associated with the lower-numbered DAC Function Block, and DAC1 is associated with the higher-numbered DAC Function Block. Each Function Block has its own clock, triggers, and DMA channel.

6.5.1 FUNCTION BLOCK REGISTER MAP

Table 16: High Speed D/A Functional Block

	Offset	0x03	0x02	0x01	0x00
Header	FB + 0x00	FB_ID			
	FB + 0x04	FB_DMA_BUFFERS	FB_DMA_CHANNELS	FB_CONTROL_SIZE	FB_CANNEL_SIZE
DAC Control	FB + 0x08	STOP_TRIG	START_TRIG	*Reserved*	MODE_STATUS
	FB + 0x0C	CLK_READY	*Reserved*	CLK_DELAY	CLK_DIV
	FB + 0x10	*Reserved*			
	FB + 0x14	*Reserved*			
	FB + 0x18	POST_STOP_CONVERSION			
	FB + 0x1C	SAMPLE_CNT			
	FB + 0x20	INT_ENA (Sample, Start, Stop, Threshold, etc)			
	FB + 0x24	INT_STAT		*Reserved*	
	FB + 0x28	CLK_BUS3	CLK_BUS2	*Reserved*	*Reserved*
	FB + 0x2C	CLK_BUS7	CLK_BUS6	CLK_BUS5	CLK_BUS4
	FB + 0x30	DA_CONFIG (Maskable register – 16-bit)			
	FB + 0x34	DAC_TRIM_BUSY		DAC_TRIM	
	FB + 0x38	DAC_OFFSET_BUSY		DAC_OFFSET	
DAC Channel	FB + 0x08 + (FB_CONTROL_SIZE * 0x04)	FRONT_END_CONFIG (Maskable register – 16-bit)			
	FB + 0x0C + (FB_CONTROL_SIZE * 0x04)	FIFO_DATA_CNT			
	FB + 0x10 + (FB_CONTROL_SIZE * 0x04)	MAX_FIFO_SIZE			
	FB + 0x14 + (FB_CONTROL_SIZE * 0x04)	INT_ENA	INT_STAT	*Reserved*	*Reserved*
	FB + 0x18 + (FB_CONTROL_SIZE * 0x04)	*Reserved (Checksum of DAC Data)*			
	FB + 0x1C + (FB_CONTROL_SIZE * 0x04)	LAST_CONVERSION			
	FB + 0x20 + (FB_CONTROL_SIZE * 0x04)	FIFO_ACCESS			

6.5.2 MODE_STATUS (READ/WRITE, READ-ONLY)

Selects the current mode of operation and indicates its triggering status.

B[3:0]: Mode

- 0x04: Uninitialized. This is the power-on state. No converter initialization has taken place. Sampling is stopped, and all counters are reset, and the triggering state machine is reset. Transition to any of the other Modes will start converter initialization (sampling will not start until initialization is complete).
- 0x00: Reset. Sampling is stopped. All counters are reset, and the triggering state machine is reset.
- 0x01: Paused. Sampling is stopped, but the counters and triggering state machine maintain their state.
- 0x02: Go, Single-Shot. After converting the Post-Stop number of values, converting stops. The Mode must be set back to RESET in order to convert more values.
- 0x03: Go, Re-arm. After converting the Post-Stop number of values, the triggering state machine is restarted. DAC data is resumed from that last value sent.

B[7:4]: Status

- 0x08: Uninitialized – The status when in the “Uninitialized” mode and the converter requires initialization.
- 0x09: Initializing
- 0x00: Stopped – The status when in the “Reset” mode, or in the “Uninitialized” mode and the converter does not require initialization.
- 0x01: Filling Pre-Trigger buffer
- 0x02: Waiting for start trigger
- 0x03: Sampling/Waiting for stop trigger
- 0x04: Filling Post-Stop buffer
- 0x05: Wait to re-arm – Waiting until local FIFO is empty, so the pre-trigger buffer can be filled.
- 0x07: Done capturing

6.5.3 START_TRIG (READ/WRITE)

Selects the start trigger from the clock bus. The DAC will start capturing after the start trigger, unless PRE_TRIGGER_CAPTURE is non-zero in which case the DAC will start capturing immediately.

Below is the list of clock sources and the register value needed to select the source.

B[7:0]:

- 0x00: System clock/immediate
- 0x01: Never
- 0x02: CLK_BUS2
- 0x03: CLK_BUS3
- 0x04: CLK_BUS4
- 0x05: CLK_BUS5
- 0x06: CLK_BUS6
- 0x07: CLK_BUS7
- 0x08: Threshold – DAC has exceeded the High or Low threshold.
- 0x09: Threshold Inverted– The DAC is within the High and Low threshold.
- 0x0A: CLK_BUS2 Inverted
- 0x0B: CLK_BUS3 Inverted
- 0x0C: CLK_BUS4 Inverted
- 0x0D: CLK_BUS5 Inverted
- 0x0E: CLK_BUS6 Inverted
- 0x0F: CLK_BUS7 Inverted

6.5.4 STOP_TRIG (READ/WRITE)

Selects the stop trigger from the clock bus.

Refer to START_TRIG section above, for a list of valid values.

6.5.5 CLK_DIV(READ/WRITE)

Divider for the conversion clock.

$$Sample\ Clock\ Frequency\ = \frac{200\ MHz}{(2\ \times\ CLK_DIV)}$$

Maximum clock rate is 25Mhz, CLK_DIV = 0x04

Minimum clock rate is 1Mhz, CLK_DIV = 0x64

Disabling clock, CLK_DIV = 0x00

6.5.6 CLK_DELAY(READ/WRITE)

This register is used to set a delay relative to the DAC's distribution clock. This delay will only be noticeable affect the DAC output clocks have been synchronized by using the 6.5.4 register of the Clock Generator function block.

$$Sample\ Clock\ Delay\ =\ 150ps\ \times CLK_DELAY$$

Maximum clock delay is 2250ps, CLK_DELAY = 0xF

Minimum clock delay is 0ps, CLK_DIV = 0x0

6.5.7 CLK_READY

This register is used to tell the status of the DAC clock. When changing the CLK_DIV or CLK_DELAY register, checking this register will indicate if the latest values have been set.

B7: Clock status 1 – Clock is ready; the latest values have been loaded0 – Clock is not ready, still setting the latest values

6.5.8 POST_STOP_CONVERSIONS (READ/WRITE)

Number of conversions to send after the Stop Trigger.

6.5.9 CONVERSION_CNT (READ ONLY)

Total number of conversions collected. This does not increment in while in the "Waiting For Start Trigger" state. It also continues counting after a Re-Arm.

6.5.10 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.

6.5.11 INT_STAT (READ/CLEAR)

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.

B0: Conversion – A value has been sent

B1: Channel Marker – One of the channels has an enabled marker.

B2: Reserved

B3: Start Trigger

B4: Stop Trigger

B5: Post-Stop Conversions Completed

B6: Reserved

B7: Reserved

6.5.12 CLK_BUSN

Selects a source to drive onto Clock Bus N. That clock bus can then be used by a different function block as a clock source or trigger.

A function block can drive multiple different Clock Buses. However, a Clock Bus N should not be driven by more than one function block at the same time or the clock signal will be undefined.

B[7:0]:

- 0x00: Disables Clock Source
- 0x80: Conversion – A value has been sent.
- 0x81: Channel Marker – One of the channels has an enabled marker.
- 0x82: Reserved
- 0x83: Start Trigger
- 0x84: Stop Trigger
- 0x85: Post-Stop Conversions Completed
- 0x86: Reserved
- 0x87: Reserved

6.5.13 DA_CONFIG (MASKABLE READ/WRITE)

Not used.

6.5.14 DAC_TRIM (READ/WRITE)

This register is used to zero out the offset of the DAC output. The value is stored as a two's compliment format. Currently there is no way to store this value in the onboard, so this register defaults to 0 at power on.

$$Trim\ Voltage\ = \frac{DAC_TRIM \times\ 5}{4096}$$

6.5.15 DAC_TRIM_BUSY (READ ONLY)

This register is used to tell the status of the DAC trim control. When changing the DAC_TRIM register, checking this register will indicate if the latest values have been set.

B7: Trim status 1 – Ready; the latest values have been set 0 – Busy, still setting the latest values

6.5.16 DAC_OFFSET (READ/WRITE)

This register is used to offset the DAC output. The value is stored as a two's compliment format. This register defaults to 0 at power on.

$$Offset\ Voltage\ = \frac{DAC_Offset \times\ 5}{4096}$$

6.5.17 DAC_OFFSET_BUSY (READ ONLY)

This register is used to tell the status of the DAC trim control. When changing the DAC_OFFSET register, checking this register will indicate if the latest values have been set.

B7: Offset status 1 – Ready; the latest values have been set 0 – Busy, still setting the latest values

6.5.18 FRONT_END_CONFIG (MASKABLE READ/WRITE)

Refer to section **5.9** for more information about the front-end circuit.

This register provides configuration to the Front End for the DAC.

B[0]: Output_Enable 0 = Disable 1 = Enable
B[1]: Reserved

6.5.19 FIFO_DATA_CNT (READ)

This register shows the current conversion count that is available in the DAC's FIFO.

6.5.20 MAX_FIFO_SIZE (READ)

This register shows the max number of conversions that the DAC's FIFO can hold.

6.5.21 INT_STAT (READ/CLEAR)

This is the status register for the channel interrupts. 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.

The upper eight bits of the DAC value can be used for Markers. These Markers can be used to generate an interrupt when a certain part of the waveform is sent to the DAC. This allows an automated indication to the application software as to the state of the data being sent to the DAC. Marker bit 7 corresponds to bit 31 of the DAC data, and Marker bit 0 corresponds to bit 24 of the DAC data.

B0: Marker 0
B1: Marker 1
B2: Marker 2
B3: Marker 3
B4: Marker 4
B5: Marker 5
B6: Marker 6
B7: Marker 7

6.5.22 INT_ENA (READ/WRITE)

These are interrupts enables for the Data Markers. Bit defines are above.

6.5.23 LAST_CONVERSION (READ/WRITE)

The last value sent to the DAC Converter.

B[31:24]: DAC Markers
B[15:0]: DAC Data

If the current Mode is "Reset" or the associated DMA engine is set to "Clear", a write to this register will immediately update the DAC Converter.

6.5.24 FIFO_ACCESS (READ/WRITE)

This register provides direct access to the DMA FIFO. It can be used to access the data without the use of the DMA engine. The DMA engine for this channel must be set to "Pause." Each register access advances to the next conversion.

6.6 BAR2: Advanced Digital I/O

This function block is for 32 bi-directional digital I/O.

6.6.1 Function Block Register Map

Table 17: Advance DIO Functional Block

	Offset	0x03	0x02	0x01	0x00
Header	FB + 0x00	FB_ID			
	FB + 0x04	FB_DMA_BUFFERS	FB_DMA_CHANNELS	FB_CONTROL_SIZE	FB_CANNEL_SIZE
Digital I/O Control	FB + 0x08	STOP_TRIG	START_TRIG	CLK_SRC	MODE_STATUS
	FB + 0x0C	CLK_DIV			
	FB + 0x10	CLK_DIV_CNTR			
	FB + 0x14	PRE_TRIGGER_CAPTURE (limited by FIFO size)			
	FB + 0x18	POST_STOP_CAPTURE			
	FB + 0x1C	SAMPLE_CNT			
	FB + 0x20	INT_ENA (Sample, Start, Stop, Threshold, Pacer Tick, etc)			
	FB + 0x24	INT_STAT		*Reserved*	
	FB + 0x28	CLK_BUS3	CLK_BUS2	*Reserved*	*Reserved*
	FB + 0x2C	CLK_BUS7	CLK_BUS6	CLK_BUS5	CLK_BUS4
	FB + 0x30	*Reserved*			ADV_INT_MODE
	FB + 0x34	ADV_INT_MASK			
	FB + 0x38	ADV_INT_COMP			
	FB + 0x3C	ADV_INT_CAPT			
	FB + 0x40	MODE_CONFIG (Maskable register – 16-bit)			
	FB + 0x44	PERIPH_SEL_BIT_3	PERIPH_SEL_BIT_2	PERIPH_SEL_BIT_1	PERIPH_SEL_BIT_0
	FB + 0x48	PERIPH_SEL_BIT_7	PERIPH_SEL_BIT_6	PERIPH_SEL_BIT_5	PERIPH_SEL_BIT_4
	FB + 0x4C	PERIPH_SEL_BIT_11	PERIPH_SEL_BIT_10	PERIPH_SEL_BIT_9	PERIPH_SEL_BIT_8
	FB + 0x50	PERIPH_SEL_BIT_15	PERIPH_SEL_BIT_14	PERIPH_SEL_BIT_13	PERIPH_SEL_BIT_12
	FB + 0x44	PERIPH_SEL_BIT_19	PERIPH_SEL_BIT_18	PERIPH_SEL_BIT_17	PERIPH_SEL_BIT_16
	FB + 0x48	PERIPH_SEL_BIT_23	PERIPH_SEL_BIT_22	PERIPH_SEL_BIT_21	PERIPH_SEL_BIT_20
	FB + 0x4C	PERIPH_SEL_BIT_27	PERIPH_SEL_BIT_26	PERIPH_SEL_BIT_25	PERIPH_SEL_BIT_24
	FB + 0x50	PERIPH_SEL_BIT_31	PERIPH_SEL_BIT_30	PERIPH_SEL_BIT_29	PERIPH_SEL_BIT_28
Digital I/O Channel	FB + 0x18 + (FB_CONTROL_SIZE * 0x04) + (FB_CHANNEL_SIZE * 0x04 * n)	CHn_FIFO_DATA_CNT			
	FB + 0x1C + (FB_CONTROL_SIZE * 0x04) + (FB_CHANNEL_SIZE * 0x04 * n)	CHn_MAX_FIFO_SIZE			
	FB + 0x20 + (FB_CONTROL_SIZE * 0x04) + (FB_CHANNEL_SIZE * 0x04 * n)	CHn_DATA (DIG_IN/DIG_OUT/DIG_DIR)			
	FB + 0x24 + (FB_CONTROL_SIZE * 0x04) + (FB_CHANNEL_SIZE * 0x04 * n)	CHn_FIFO_ACCESS			

Note: n in the offset stands for channel number. Channel 0 is Digital In, Channel 1 is Digital Out, Channel 2 is Pin Direction

6.6.2 Mode_Status (Read/Write, Read-Only)

Selects the current mode of operation and indicates its triggering status.

B[3:0]: Mode

- 0x04: Uninitialized. This is the power-on state. No converter initialization has taken place. Sampling is stopped, and all counters are reset and the triggering state machine is reset.

Transition to any of the other Modes will start converter initialization (sampling will not start until initialization is complete).

- 0x00: Reset. Sampling is stopped. All counters are reset and the triggering state machine is reset.
- 0x01: Paused. Sampling is stopped, but the counters and triggering state machine maintain their state.
- 0x02: Go, Single-Shot. After filling the buffer with the Post-Stop samples, capturing stops. The Mode must be set back to RESET in order to capture more samples.
- 0x03: Go, Re-arm. After filling the buffer with the Post-Stop samples and the FIFO is empty, the triggering state machine is restarted, i.e. FIFO is filled with Pre-Start samples and waits for a start trigger.

B[7:4]: Status

- 0x08: Uninitialized – The status when in the "Uninitialized" mode and the converter requires initialization.
- 0x09: Initializing
- 0x00: Stopped – The status when in the "Reset" mode, or in the "Uninitialized" mode and the converter does not require initialization.
- 0x01: Filling Pre-Trigger buffer
- 0x02: Waiting for start trigger
- 0x03: Sampling/Waiting for stop trigger
- 0x04: Filling Post-Stop buffer
- 0x05: Wait to re-arm – Waiting until local FIFO is empty so the pre-trigger buffer can be filled.
- 0x07: Done capturing

6.6.3 CLK_SRC (READ/WRITE)

Selects the source for the clock of this function block.

- 0x00: System clock/immediate
- 0x01: Never
- 0x02: CLK_BUS2
- 0x03: CLK_BUS3
- 0x04: CLK_BUS4
- 0x05: CLK_BUS5
- 0x06: CLK_BUS6
- 0x07: CLK_BUS7
- 0x08: Advanced Interrupt
- 0x09: Reserved
- 0x0A: CLK_BUS2 Inverted
- 0x0B: CLK_BUS3 Inverted
- 0x0C: CLK_BUS4 Inverted
- 0x0D: CLK_BUS5 Inverted
- 0x0E: CLK_BUS6 Inverted
- 0x0F: CLK_BUS7 Inverted

6.6.4 START_TRIG (READ/WRITE)

Selects the start trigger from the clock bus. CLK_DIV will start counting after the start trigger, unless PRE_START_COUNT is non-zero in which case CLK_DIV will start counting immediately.

Refer to CLK_SRC section above, for list of valid values.

6.6.5 STOP_TRIG (READ/WRITE)

Selects the stop trigger from the clock bus.

Refer to CLK_SRC section above, for list of valid values.

6.6.6 CLK_DIV (READ/WRITE)

Divider for the pacer clock. Pacer Clock Frequency = (Clk_Src_Frequency) / (1 + CLK_DIV). If synchronizing with the pacer clock from another Function Block (by using one of the CLK_BUS signals), this is typically set to 0.

6.6.7 CLK_DIV_CNTR (READ ONLY)

The current value of the Clock Divide Counter. This counter starts at a value of CLK_DIV and counts down. When it reaches zero, a sample is taken. This is useful when using a slow sample clock.

6.6.8 PRE_START_COUNT (READ/WRITE)

Number of samples to collect before the Start Trigger. The length is limited by the FIFO size – writing a value larger than the FIFO size will have indeterminate results.

6.6.9 POST_STOP_COUNT (READ/WRITE)

Number of samples to collect after the Stop Trigger.

6.6.10 SAMPLE_CNT (READ ONLY)

Total number of samples collected. This does not increment in while in the "Waiting For Start Trigger" state. It also continues counting after a Re-Arm.

6.6.11 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.

6.6.12 INT_STAT (READ/CLEAR)

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.

B0: Sample – A sample has been taken.

B1: Advanced Interrupt – Interrupt generated from advanced interrupt mode.

B2: Pre-Start Buffer Filled

B3: Start Trigger

B4: Stop Trigger

B5: Post-Stop Buffer Filled

B6: Sampling has completed, and the FIFO is empty (all data transferred to host)

B7: Pacer – The pacer clock has ticked.

B8: Reserved

B9: Reserved

B10: Strobe – Positive edge detected (Must have strobe enable,MODE_CONFIG(Strobe) = '1')

6.6.13 CLK_BUSN

Selects a source to drive onto Clock Bus N. That clock bus can then be used by a different function block as a clock source or trigger.

A function block can drive multiple different Clock Buses. However, a Clock Bus N should not be driven by more than one function block at the same time or the clock signal will be undefined.

B[7:0]:

- 0x00: Disabled (Default)
- 0x80: Sample – A sample has been taken.
- 0x81: Advanced Interrupt – Interrupt generated from advanced interrupt mode.
- 0x82: Pre-Start Buffer Filled
- 0x83: Start Trigger
- 0x84: Stop Trigger
- 0x85: Post-Stop Buffer Filled
- 0x86: Sampling has completed, and the FIFO is empty (all data transferred to host)
- 0x87: Pacer – The pacer clock has ticked.
- 0x88: Reserved
- 0x89: Reserved
- 0x8A: Strobe – Positive edge detected (Must have strobe enable,MODE_CONFIG(Strobe) = '1')

6.6.14 ADV_INT_MODE (READ/WRITE)

Set the current mode for the advance interrupts.

B[1:0]: Mode

- 0x0: Disabled. This is the power-on state. In this mode no advanced interrupts will occur.
- 0x1: Match. A Match interrupt is generated when all un-masked bits in the Compare register match the input value of the port. This is when the following expression is true:

  ((DIG_IN xor ADV_INT_COMP) and not ADV_INT_MASK) = '0'
- 0x2: Event Mode. An Event interrupt is generated when any un-masked input port bit changes. This is when the following expression is:

  ((DIG_IN xor ADV_INT_ CAPT) and not ADV_INT_MASK) = '1'

  The ADV_INT_CAPT register is updated at every advanced interrupt or event.
- 0x3: Strobe. A Strobe interrupt is generated every rising edge of the CLK_SRC for the function block:

  The ADV_INT_CAPT register is updated at every advanced interrupt

6.6.15 ADV_INT_MASK (READ/WRITE)

This register determines if a bit is checked for the advanced interrupts.

0 = Bit is used for match/event 1 = Bit is ignored

6.6.16 ADV_INT_COMP (READ/WRITE)

The compare register is used for the Match interrupt. When all selected bits in this register match all selected bits on the DIG_IN register, an interrupt is generated.

6.6.17 ADV_INT_CAPT (READ/WRITE)

The Capture register latches the input ports when an interrupt is generated. All values are latched, regardless of the ADV_INT_MASK register, or DIO_DIRECTION.

This register can be written to when ADV_INT_MODE is set to Disabled.

6.6.18 MODE_CONFIG (MASKABLE READ/WRITE)

This register provides configuration for Digital Input Function Block.

B[2]: Strobe: Configures the most significant bit of the DIN function block as a strobe input
- '0' = Normal input
- '1' = Strobe input

B[1:0]: Reserved

6.6.19 PERIPH_SEL_BIT_N (READ/WRITE)

This register selects the peripheral for the outputs pin of this function block.

NOTE: Only set the PERIPH_SEL_BIT when use the pin as an output. Unexpected results will happen the PERIPH_SEL_BIT is changed on an input.

Bit	CN3 Pin Number	PERIPH_SEL_BIT_n		
		0x00	0x01	0x02
31	34	DIG_OUT (31)		
30	33	DIG_OUT (30)		
29	32	DIG_OUT (29)		
28	31	DIG_OUT (28)		
27	30	DIG_OUT (27)		
26	29	DIG_OUT (26)		
25	28	DIG_OUT (25)		
24	27	DIG_OUT (24)		
23	26	DIG_OUT (23)		
22	25	DIG_OUT (22)		
21	24	DIG_OUT (21)		
20	23	DIG_OUT (20)		
19	22	DIG_OUT (19)		
18	21	DIG_OUT (18)		
17	20	DIG_OUT (17)		
16	19	DIG_OUT (16)		
15	16	DIG_OUT (15)		
14	15	DIG_OUT (14)		
13	15	DIG_OUT (13)		
12	13	DIG_OUT (12)		
11	12	DIG_OUT (11)		EXT_CLK_GATE7
10	11	DIG_OUT (10)		EXT_CLK_7
9	10	DIG_OUT (9)		EXT_CLK_GATE6
8	9	DIG_OUT (8)		EXT_CLK_6
7	8	DIG_OUT (7)	PWM0_N (3)	EXT_CLK_GATE5
6	7	DIG_OUT (6)	PWM0_P (3)	EXT_CLK_5
5	6	DIG_OUT (5)	PWM0_N (2)	EXT_CLK_GATE4
4	5	DIG_OUT (4)	PWM0_P (2)	EXT_CLK_4
3	4	DIG_OUT (3)	PWM0_N (1)	EXT_CLK_GATE3
2	3	DIG_OUT (2)	PWM0_P (1)	EXT_CLK_3
1	2	DIG_OUT (1)	PWM0_N (0)	EXT_CLK_GATE2
0	1	DIG_OUT (0)	PWM0_P (0)	EXT_CLK_2

6.6.20 CHN_FIFO_DATA_CNT (READ)

This register shows the current sample count in Digital I/O channel FIFO.

6.6.21 CHN_MAX_FIFO_SIZE (READ)

This register shows the max number of samples that the Digital I/O channel FIFO can hold.

6.6.22 CH0_DATA (DIG_IN) (READ ONLY)

This register provides the current value on the Digital In lines. The bits in the register correspond with the Digital In pins as follows:

Bit	CN3	Signal Name
31	34	DIO0.31
30	33	DIO0.30
29	32	DIO0.29
28	31	DIO0.28
27	30	DIO0.27
26	29	DIO0.26
25	28	DIO0.25
24	27	DIO0.24
23	26	DIO0.23
22	25	DIO0.22
21	24	DIO0.21
20	23	DIO0.20
19	22	DIO0.19
18	21	DIO0.18
17	20	DIO0.17
16	19	DIO0.16

Bit	CN3	Signal Name
15	16	DIO0.15
14	15	DIO0.14
13	14	DIO0.13
12	13	DIO0.12
11	12	DIO0.11
10	11	DIO0.10
9	10	DIO0.9
8	9	DIO0.8
7	8	DIO0.7
6	7	DIO0.6
5	6	DIO0.5
4	5	DIO0.4
3	4	DIO0.3
2	3	DIO0.2
1	2	DIO0.1
0	1	DIO0.0

This is the same value that is written into the DMA INPUT FIFO.

6.6.23 CH1_DATA (DIG_OUT) (READ/WRITE)

The last value sent to the Digital Output. For each bit, a '0' causes the output to be low, and a value '1' causes the output to be high. Bit assignments are the same as DIG_IN.

This register defaults to all 0 at power on.

If the current Mode is "Reset" or the associated DMA engine is set to "Clear", a write to this register will immediately update the Digital I/O Output. Bit assignments are the same as above.

6.6.24 CH2_DATA (DIG_DIR) (READ/WRITE)

The last value sent to the Digital I/O Direction.

If the current Mode is "Reset" or the associated DMA engine is set to "Clear", a write to this register will immediately update the Digital I/O Direction. Bit assignments are the same as above.

Selects the direction of the I/O bit. 0=input, 1=output.

All pins default to inputs at power-up.

6.6.25 CHN_FIFO_ACCESS (READ/WRITE)

This register provides direct access to the DMA FIFO. It can be used to access the data without the use of the DMA engine. The DMA engine for this channel must be set to "Pause." Each register access advances to the next sample.

6.1 BAR2: PWM

This function block provides interface to the PWM function block. This function block features 4 PWM channels.

6.1.1 FUNCTION BLOCK REGISTER MAP

Table 18:PWM Functional Block

	Offset	0x03	0x02	0x01	0x00
Header	FB + 0x00	FB_ID			
	FB + 0x04	FB_DMA_BUFFERS	FB_DMA_CHANNELS	FB_CONTROL_SIZE	FB_CHANNEL_SIZE
PWM Control	FB + 0x08	STOP_TRIG	START_TRIG	CLK_SRC	MODE_STATUS
	FB + 0x0C	CLK_DIV			
	FB + 0x10	CLK_DIV_CNTR			
	FB + 0x14	Reserved			
	FB + 0x18	Reserved			
	FB + 0x1C	PERIOD_COUNT			
	FB + 0x20	INT_ENA (Conversion, Start, Stop, Error, Channel)			
	FB + 0x24	INT_STAT		*Reserved*	
	FB + 0x28	CLK_BUS3	CLK_BUS2	*Reserved*	*Reserved*
	FB + 0x2C	CLK_BUS7	CLK_BUS6	CLK_BUS5	CLK_BUS4
Channel n	FB + 0x08 + (FB_CONTROL_SIZE * 0x04) + (FB_CHANNEL_SIZE * 0x04 * n)	CHn_FIFO_DATA_CNT			
	FB + 0x0C + (FB_CONTROL_SIZE * 0x04) + (FB_CHANNEL_SIZE * 0x04 * n)	CHn_MAX_FIFO_SIZE			
	FB + 0x10 + (FB_CONTROL_SIZE * 0x04) + (FB_CHANNEL_SIZE * 0x04 * n)	CHn_PWM_WIDTH			
	FB + 0x14 + (FB_CONTROL_SIZE * 0x04) + (FB_CHANNEL_SIZE * 0x04 * n)	CHn_FIFO_ACCESS			

Note: n in the offset stands for the channel number.

6.1.2 MODE_STATUS (READ/WRITE, READ-ONLY)

Selects the current mode of operation and indicates its triggering status.

B[3:0]: Mode

- 0x04: Uninitialized. This is the power-on state. No converter initialization has taken place. Sampling is stopped, and all counters are reset and the triggering state machine is reset. Transition to any of the other Modes will start converter initialization (sampling will not start until initialization is complete).
- 0x00: Reset. Counting is stopped. All counters are reset and the triggering state machine is reset.
- 0x01: Paused. Counting is stopped, but the counters and triggering state machine maintain their state.
- 0x02: Go, Single-Shot. After filling the buffer with the Post-Stop samples, capturing stops. The Mode must be set back to RESET in order to capture more samples.
- 0x03: Go, Re-arm. After filling the buffer with the Post-Stop samples and the FIFO is empty, the triggering state machine is restarted, i.e. FIFO is filled with Pre-Start samples and waits for a start trigger.

B[7:4]: Status

- 0x08: Uninitialized – The status when in the “Uninitialized” mode and the converter requires initialization.
- 0x09: Initializing
- 0x00: Stopped – The status when in the “Reset” mode, or in the “Uninitialized” mode and the converter does not require initialization.
- 0x01: Reserved
- 0x02: Waiting for start trigger
- 0x03: Running/Waiting for stop trigger
- 0x04: Reserved
- 0x05: Reserved
- 0x07: Done capturing

6.1.3 CLK_SRC (READ/WRITE)

Selects the source for the clock of this function block.

0x00: System clock/immediate
0x01: Never
0x02: CLK_BUS2
0x03: CLK_BUS3
0x04: CLK_BUS4
0x05: CLK_BUS5
0x06: CLK_BUS6
0x07: CLK_BUS7
0x08: Reserved
0x09: Reserved
0x0A: CLK_BUS2 Inverted
0x0B: CLK_BUS3 Inverted
0x0C: CLK_BUS4 Inverted
0x0D: CLK_BUS5 Inverted
0x0E: CLK_BUS6 Inverted
0x0F: CLK_BUS7 Inverted

6.1.4 START_TRIG (READ/WRITE)

Selects the start trigger from the clock bus. CLK_DIV will start counting after the start trigger, unless PRE_START_COUNT is non-zero in which case CLK_DIV will start counting immediately.

Refer to CLK_SRC section above, for list of valid values.

6.1.5 STOP_TRIG (READ/WRITE)

Selects the stop trigger from the clock bus.

Refer to CLK_SRC section above, for list of valid values.

6.1.6 CLK_DIV (READ/WRITE)

Divider for the pacer clock. Pacer Clock Frequency = (Clk_Src_Frequency) / (1 + CLK_DIV). If synchronizing with the pacer clock from another Function Block (by using one of the CLK_BUS signals), this is typically set to 0.

6.1.7 CLK_DIV_CNTR (READ ONLY)

The current value of the Clock Divide Counter. This counter starts at a value of CLK_DIV, and counts down. When it reaches zero, a sample is taken. This is useful when using a slow sample clock.

6.1.8 PERIOD_COUNT (READ ONLY)

Total number of periods executed. This does not increment in while in the "Waiting For Start Trigger" state. It also continues counting after a Re-Arm.

6.1.9 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.

6.1.10 INT_STAT (READ/CLEAR)

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.

B0: Period – A period occurred.

B1: Reserved

B2: Reserved

B3: Start Trigger

B4: Stop Trigger

B5: Post-Stop Buffer Empty

B6: Reserved

B7: Pacer – The pacer clock has ticked.

6.1.11 CLK_BUSN

Selects a source to drive onto Clock Bus N. That clock bus can then be used by a different function block as a clock source or trigger.

A function block can drive multiple different Clock Buses. However, a Clock Bus N should not be driven by more than one function block at the same time or the clock signal will be undefined.

B[7:0]:

- 0x00: Disabled (Default)
- 0x80: Period – A period occurred.
- 0x81: Reserved
- 0x82: Reserved
- 0x83: Start Trigger
- 0x84: Stop Trigger
- 0x85: Post-Stop Buffer Empty
- 0x86: Reserved
- 0x87: Pacer – The pacer clock has ticked.

6.1.12 CHN_FIFO_DATA_CNT (READ)

This register shows the current sample count that is available in the interval counter FIFO.

6.1.13 CHN_MAX_FIFO_SIZE (READ)

This register shows the max number of samples that the interval counter FIFO can hold.

6.1.14 CHN_PWM_WIDTH (READ/WRITE)

Sets the width of output of the pulse width modulator. Pulse Width = (1 + CHn_PWM_WIDTH) / (System Clock). The width is defined as the time that the non-inverted output is high, and the inverted output is low.

The width register is checked at the beginning of every period. If the width register is modified in the middle of a period, the output will not be affected until the next period.

Note: that with Pacer Clock Frequency sets the maximum value for this register, a 100% duty cycle is not possible. This will result in the non-inverted output to be low, and the inverted output to be high.

The function block Mode needs to be set to: Go, Single-Shot or Go, Re-arm for the PWM to work. When Status is "Stopped", non-inverted output is low, and the inverted output is high.

6.1.15 CHN_FIFO_ACCESS (READ/WRITE)

This register provides direct access to the DMA FIFO. It can be used to access the data without the use of the DMA engine. The DMA engine for this channel must be set to "Pause." Each register access advances to the next sample.

6.2 BAR2: Incremental Encoder

This function block provides interface to the Incremental Encoder function block.

6.2.1 Function Block Register Map

Table 19: Incremental Encoder Functional Block

	Offset	0x03	0x02	0x01	0x00
Header	FB + 0x00	FB_ID			
	FB + 0x04	FB_DMA_BUFFERS	FB_DMA_CHANNELS	FB_CONTROL_SIZE	*Reserved*
Incremental Encoder Control	FB + 0x08	STOP_TRIG	START_TRIG	CLK_SRC	MODE_STATUS
	FB + 0x0C	CLK_DIV			
	FB + 0x10	CLK_DIV_CNTR			
	FB + 0x14	PRE_START_COUNT (limited by FIFO size)			
	FB + 0x18	POST_STOP_COUNT			
	FB + 0x1C	SAMPLE_CNT			
	FB + 0x20	INT_ENA (Sample, Start, Stop, Threshold, Pacer Tick, etc)			
	FB + 0x24	INT_STAT		*Reserved*	
	FB + 0x28	CLK_BUS3	CLK_BUS2	*Reserved*	*Reserved*
	FB + 0x2C	CLK_BUS7	CLK_BUS6	CLK_BUS5	CLK_BUS4
Incremental Encoder Channel	FB + 0x08 + (FB_CONTROL_SIZE * 0x04)	MODE_CONFIG (Maskable register – 16-bit)			
	FB + 0x0C + (FB_CONTROL_SIZE * 0x04)	FIFO_DATA_CNT			
	FB + 0x10 + (FB_CONTROL_SIZE * 0x04)	MAX_FIFO_SIZE			
	FB + 0x14 + (FB_CONTROL_SIZE * 0x04)	THRESH_LOW			
	FB + 0x1C + (FB_CONTROL_SIZE * 0x04)	THRESH_HIGH			
	FB + 0x20 + (FB_CONTROL_SIZE * 0x04)	INCENC_VALUE			
	FB + 0x24 + (FB_CONTROL_SIZE * 0x04)	FIFO_ACCESS			

6.2.2 MODE_STATUS (Read/Write, Read-Only)

Selects the current mode of operation and indicates its triggering status.

B[3:0]: Mode

- 0x04: Uninitialized. This is the power-on state. No converter initialization has taken place. Sampling is stopped, and all counters are reset and the triggering state machine is reset. Transition to any of the other Modes will start converter initialization (sampling will not start until initialization is complete).
- 0x00: Reset. Counting is stopped. All counters are reset and the triggering state machine is reset.
- 0x01: Paused. Counting is stopped, but the counters and triggering state machine maintain their state.
- 0x02: Go, Single-Shot. After filling the buffer with the Post-Stop samples, capturing stops. The Mode must be set back to RESET in order to capture more samples.
- 0x03: Go, Re-arm. After filling the buffer with the Post-Stop samples and the FIFO is empty, the triggering state machine is restarted, i.e. FIFO is filled with Pre-Start samples and waits for a start trigger.

B[7:4]: Status

- 0x08: Uninitialized – The status when in the “Uninitialized” mode and the converter requires initialization.
- 0x09: Initializing
- 0x00: Stopped – The status when in the “Reset” mode, or in the “Uninitialized” mode and the converter does not require initialization.
- 0x01: Reserved
- 0x02: Waiting for start trigger
- 0x03: Running/Waiting for stop trigger
- 0x04: Reserved
- 0x05: Reserved
- 0x07: Done capturing

6.2.3 CLK_SRC (READ/WRITE)

Selects the source for the clock of this function block.

0x00: System clock/immediate
0x01: Never
0x02: CLK_BUS2
0x03: CLK_BUS3
0x04: CLK_BUS4
0x05: CLK_BUS5
0x06: CLK_BUS6
0x07: CLK_BUS7
0x08: Reserved
0x09: Reserved
0x0A: CLK_BUS2 Inverted
0x0B: CLK_BUS3 Inverted
0x0C: CLK_BUS4 Inverted
0x0D: CLK_BUS5 Inverted
0x0E: CLK_BUS6 Inverted
0x0F: CLK_BUS7 Inverted

6.2.4 START_TRIG (READ/WRITE)

Selects the start trigger from the clock bus. CLK_DIV will start counting after the start trigger, unless PRE_START_COUNT is non-zero in which case CLK_DIV will start counting immediately.

Refer to CLK_SRC section above, for list of valid values.

6.2.5 STOP_TRIG (READ/WRITE)

Selects the stop trigger from the clock bus.

Refer to CLK_SRC section above, for list of valid values.

6.2.6 CLK_DIV (READ/WRITE)

Divider for the pacer clock. Pacer Clock Frequency = (Clk_Src_Frequency) / (1 + CLK_DIV). If synchronizing with the pacer clock from another Function Block (by using one of the CLK_BUS signals), this is typically set to 0.

6.2.7 CLK_DIV_CNTR (READ ONLY)

The current value of the Clock Divide Counter. This counter starts at a value of CLK_DIV, and counts down. When it reaches zero, a sample is taken. This is useful when using a slow sample clock.

6.2.8 PRE_START_COUNT (READ/WRITE)

Number of samples to collect before the Start Trigger. The length is limited by the FIFO size – writing a value larger than the FIFO size will have indeterminate results.

6.2.9 POST_STOP_COUNT (READ/WRITE)

Number of samples to collect after the Stop Trigger.

6.2.10 SAMPLE_CNT (READ ONLY)

Total number of samples collected. This does not increment in while in the "Waiting For Start Trigger" state. It also continues counting after a Re-Arm.

6.2.11 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.

6.2.12 INT_STAT (READ/CLEAR)

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.

B0: Sample – A sample has been taken.

B1: Reserved

B2: Pre-Start Buffer Filled

B3: Start Trigger

B4: Stop Trigger

B5: Post-Stop Buffer Filled

B6: Sampling has completed and the FIFO is empty (all data transferred to host)

B7: Pacer – The pacer clock has ticked

B8: Positive rollover - Indicates channel has transitioned from 0xFFFFFFFF to 0x00000000.

B9: Negative rollover - Indicates channel has transitioned from 0x00000000 to 0xFFFFFFFF.

B10: High threshold crossed.

B11: Low threshold crossed.

6.2.13 CLK_BUSN

Selects a source to drive onto Clock Bus N. That clock bus can then be used by a different function block as a clock source or trigger.

A function block can drive multiple different Clock Buses. However, a Clock Bus N should not be driven by more than one function block at the same time or the clock signal will be undefined.

B[7:0]:

- 0x00: Disabled (Default)
- 0x80: Sample – A sample has been taken.
- 0x81: Reserved
- 0x82: Pre-Start Buffer Filled
- 0x83: Start Trigger
- 0x84: Stop Trigger
- 0x85: Post-Stop Buffer Filled
- 0x86: Sampling has completed and the FIFO is empty (all data transferred to host)
- 0x87: Pacer – The pacer clock has ticked.
- 0x88: Positive rollover - Indicates channel has transitioned from 0xFFFFFFFF to 0x00000000.
- 0x89: Negative rollover - Indicates channel has transitioned from 0x00000000 to 0xFFFFFFFF.
- 0x8A: High threshold crossed.
- 0x8B: Low threshold crossed.

6.2.14 MODE_CONFIG (MASKABLE READ/WRITE)

This register provides configuration for Incremental Encoder.

B[12]: Differential: Selects single-ended or differential mode
- '0' = Single Ended. Only "+" inputs are used
- '1' = Pseudo-Differential

B[11]: Reserved

B[10]: Index: When enabled, a high input on the Index input clears the counter
- '0' = Index Inputs Disabled
- '1' = Index Input Enabled

B[9]: Hold: When enabled, the encoder continues counting in the background, but the INCENC_VALUE register remains constant.
- '0' = INCENC_VALUE registers are not held
- '1' = INCENC_VALUE registers are held

B[8]: Enable: Enable for this incremental encoder
- '0' = Encoder is disabled
- '1' = Encoder is enabled

B[7:0]: Phase Filter: Selects if a particular state transition will cause the encoder counter to change.

For each bit:
- '0' = Transition will change counter
- '1' = Transition will not change counter.

The bit assignments for the transitions are:

Bit	Previous State [B:A]	Current State [B:A]	Direction
7	00	10	Down
6	10	11	Down
5	11	01	Down
4	01	00	Down
3	10	00	Up
2	11	10	Up
1	01	11	Up
0	00	01	Up

6.2.15 FIFO_DATA_CNT (READ)

This register shows the current sample count that is available in the interval counter FIFO.

6.2.16 MAX_FIFO_SIZE (READ)

This register shows the max number of samples that the interval counter FIFO can hold.

6.2.17 THRESH_LOW (READ/WRITE)

Unsigned 32-bit value indicating the low threshold. If the count value goes below the threshold value, an interrupt will be generated. The interrupt will only be generated once the threshold value is crossed.

6.2.18 THRESH_HIGH (READ/WRITE)

Unsigned 32-bit value indicating the high threshold. If the count value goes above the threshold value, an interrupt will be generated. The interrupt will only be generated once the threshold value is crossed.

6.2.19 INCENC_VALUE (READ/WRITE)

The unsigned 32-bit current value of the incremental encoder channel. This is the same value that is written to the DMA FIFO.

This register does not update when MODE_CONFIG[HOLD] = 1.

This register can only be written to when MODE_CONFIG[Enable] = 0. This allows the counter to be pre-loaded with a known position value.

6.2.20 FIFO_ACCESS (READ)

This register provides direct access to the DMA FIFO. It can be used to access the data without the use of the DMA engine. The DMA engine for this channel must be set to "Pause." Each register access advances to the next sample.

If the MODE_CONFIG[HOLD] = 1, the last value before HOLD was enable will be written to the FIFO on each clock pulse, until HOLD is disabled.

6.3 BAR2: External Clocking

This function block provides interface to the Incremental Encoder function block.

6.3.1 Function Block Register Map

Table 20:External Clocking Functional Block

	Offset	0x03	0x02	0x01	0x00
Header	FB + 0x00	FB_ID			
	FB + 0x04	FB_DMA_BUFFERS	FB_DMA_CHANNELS	*Reserved*	*Reserved*
	FB + 0x08	EXT_CLK_EDGE	EXT_CLK_DIR	EXT_CLK_GATE_IN	EXT_CLK_IN
	FB + 0x0C	EXT_CLK2_PW			
	FB + 0x10	EXT_CLK3_PW			
	FB + 0x14	EXT_CLK4_PW			
	FB + 0x18	EXT_CLK5_PW			
	FB + 0x1C	EXT_CLK6_PW			
	FB + 0x20	EXT_CLK7_PW			
	FB + 0x24	EXT_CLK3_CFG	EXT_CLK2_CFG	*Reserved*	*Reserved*
	FB + 0x28	EXT_CLK7_CFG	EXT_CLK6_CFG	EXT_CLK5_CFG	EXT_CLK4_CFG

6.3.2 EXT_CLK_IN (Read-Only)

This register provides the current value on the External Clocking lines. The bits in the register correspond with the External Clocking pins as follows:

Bit	CN3 Pin #	CLK_BUS	Signal
5	11	7	EXT_CLK_7
4	9	6	EXT_CLK_6
3	7	5	EXT_CLK_5
2	5	4	EXT_CLK_4
1	3	3	EXT_CLK_3
0	1	2	EXT_CLK_2

6.3.3 EXT_CLK_GATE_IN (Read Only)

This register provides the current value on the External Clocking Gates lines. External Clocking Gates can only be used when inputting an external clock. The bits in the register correspond with the External Clocking Gates pins as follows:

Bit	CN3 Pin #	CLK_BUS	Signal
5	12	7	EXT_CLK_GATE7
4	10	6	EXT_CLK_GATE6
3	8	5	EXT_CLK_GATE5
2	6	4	EXT_CLK_GATE4
1	4	3	EXT_CLK_GATE3
0	2	2	EXT_CLK_GATE2

6.3.4 EXT_CLK_DIR (Read/Write)

Selects the direction of the External Clocking bits. 0=input, 1=output.

All pins default to inputs at power-up.

6.3.5 EXT_CLK_EDGE (READ/WRITE)

Selects which edge detect to trigger on. This is a bit settable register.

0= Rising Edge Detect, 1= Fall Edge Detect.

6.3.6 EXT_CLKN_PW (READ/WRITE)

This register is used to increase the pulse width of the clock when it is set as an output.

$$Pulse\ Width\ =\ (1\ +\ EXT_CLKn_PW)\ /\ (System\ Clock)$$

NOTE: If EXT_CLKn_PW is set to be wider than the EXT_CLKn_CFG Clock Frequency the signal will just stay high.

6.3.7 EXT_CLKN_CFG (READ/WRITE)

Selects clocking method.

B[7:0]:

- 0x00: Disables External Clocking.
- 0x80: Not Gated: CLK_BUSn will be inputted/outputted independent of the EXT_CLK_GATEn corresponding gate value.
- 0x81: Clock Gated (High): CLK_BUSn will be inputted when the EXT_CLK_GATEn corresponding gate value is high, this doesn't affect when outputting a clock.
- 0x82: Clock Gated (Low): CLK_BUSn will be inputted when the EXT_CLK_GATEn corresponding gate value is low, this doesn't affect when outputting a clock.

6.4 BAR2: Temperature Sensor

This function block provides readout of the onboard temperature sensor. There are not any interrupts or DMA.

6.4.1 Function Block Register Map

Table 21: Temperature Sensor Functional Block

	Offset	0x03	0x02	0x01	0x00
Header	FB + 0x00	FB_ID			
	FB + 0x04	FB_DMA_BUFFERS	FB_DMA_CHANNELS	Reserved	*Reserved*
	FB + 0x08	*Reserved*		TEMPERATURE_VAL	

6.4.2 TEMPERATURE_VAL (Read Only)

The current value on the temperature sensor as a signed 16-bit integer. B[2:0] of this register always read '0'. See the equation and table below for converting register value to temperature.

$$Temperature = Register\ Value * 0.0078125\ C$$

Temperature (ºC)	Register Value
150	0x4B00
125	0x3E80
25	0x0C80
0.0625	0x0008
0	0x0000
-0.0625	0xFFF8
-25	0xF380
-55	0xE480

6.5 BAR2: Clock Generator

This function block provides interface to the Clock Generator function block.

6.5.1 Function Block Register Map

Table 22: Clock Generator Functional Block

	Offset	0x03	0x02	0x01	0x00
Header	FB + 0x00	FB_ID			
	FB + 0x04	FB_DMA_BUFFERS	FB_DMA_CHANNELS	Reserved	*Reserved*
	FB + 0x08	*Reserved*	*Reserved*	PLL_STATUS	CLK_SEL
	FB + 0x0C	*Reserved*			CLK_SYNC

6.5.2 CLK_SEL (Read/Write)

Selects the 10MHz reference clock that is used to generate the system and ADC clocks.

B[3:0]:

- 0x00: Onboard Reference Clock
- 0x01: External Reference Clock (CN5)
- 0x02: SyncBus Reference Clock, (SyncBus3 of CN11)
- 0x03: Reserved

6.5.3 PLL_STATUS (READ ONLY)

This register indicates the status of the PLL of clock generator.

B0: Lock Detect – The clock generator's PLL are locked and clock is being outputted, this should always be '1'.
0 = PLL is not locked 1 = PLL is locked
B1: Loss of Signal 0 (LOS0) - Indicates the loss of 10MHz reference when not using the onboard reference
0 = Clock is not present 1 = Clock is present
B2: Loss of Signal 1 (LOS1) - Indicates the loss of the onboard 10MHz reference, this should always be '1'.
0 = Clock is not present 1 = Clock is present

6.5.4 CLK_SYNC (READ/CLEAR)

Used to synchronize the system and active ADC clock to the reference clock. Once, the ADC clock are set to the needed frequency, writing to this register will synchronize the clocks outputs to the reference clock.

B0: for reads, 0 = Busy, 1 = Synchronization is ready. Write 0x01 to start the synchronization process.

6.6 BAR2: Capture Window

The capture window function block provides the ability to generate an arbitrary waveform that then can be place on the clock bus. Each window has a delay counter register and a capture counter register. The delay counter register determines how many clock cycles to delay sampling for and the capture counter register determines the how many clock cycles to capture data for.

6.6.1 FUNCTION BLOCK REGISTER MAP

Table 23: Capture Window Functional Block

	Offset	0x03	0x02	0x01	0x00
Header	FB + 0x00	FB_ID			
	FB + 0x04	FB_DMA_BUFFERS	FB_DMA_CHANNELS	FB_CONTROL_SIZE	Reserved
Control	FB + 0x08	Reserved	START_TRIG	CLK_SRC	MODE_STATUS
	FB + 0x0C	CLK_DIV			
	FB + 0x10	CLK_DIV_CNTR			
	FB + 0x14	Reserved			
	FB + 0x18	Reserved			
	FB + 0x1C	SAMPLE_CNT			
	FB + 0x20	INT_ENA (Conversion, Start, Stop, Error, Channel)			
	FB + 0x24	INT_STAT		Reserved	
	FB + 0x28	CLK_BUS3	CLK_BUS2	Reserved	Reserved
	FB + 0x2C	CLK_BUS7	CLK_BUS6	CLK_BUS5	CLK_BUS4
	FB + 0x30	CONFIG (Maskable register – 16-bit)			
	FB + 0x08 + (FB_CONTROL_SIZE * 0x04)	DELAY_COUNTER_0			
	FB + 0x0C + (FB_CONTROL_SIZE * 0x04)	CAPTURE_COUNTER_0			
	FB + 0x10 + (FB_CONTROL_SIZE * 0x04)	DELAY_COUNTER_1			
	FB + 0x14 + (FB_CONTROL_SIZE * 0x04)	CAPTURE_COUNTER_1			
	FB + 0x18 + (FB_CONTROL_SIZE * 0x04)	DELAY_COUNTER_2			
	FB + 0x1C + (FB_CONTROL_SIZE * 0x04)	CAPTURE_COUNTER_2			
	FB + 0x20 + (FB_CONTROL_SIZE * 0x04)	DELAY_COUNTER_3			
	FB + 0x24 + (FB_CONTROL_SIZE * 0x04)	CAPTURE_COUNTER_3			

6.6.2 MODE_STATUS (READ/WRITE, READ-ONLY)

Selects the current mode of operation and indicates its triggering status.

B[3:0]: Mode

- 0x04: Uninitialized. This is the power-on state. No converter initialization has taken place. Sampling is stopped, and all counters are reset and the triggering state machine is reset. Transition to any of the other Modes will start converter initialization (sampling will not start until initialization is complete).
- 0x00: Reset. Sampling is stopped. All counters are reset and the triggering state machine is reset.
- 0x01: Paused. Sampling is stopped, but the counters and triggering state machine maintain their state.
- 0x02: Go, Single-Shot. After filling the buffer with the Post-Stop samples, capturing stops. The Mode must be set back to RESET in order to capture more samples.
- 0x03: Go, Re-arm. After filling the buffer with the Post-Stop samples and the FIFO is empty, the triggering state machine is restarted, i.e. FIFO is filled with Pre-Start samples and waits for a start trigger.

B[7:4]: Status

- 0x08: Uninitialized – The status when in the “Uninitialized” mode and the converter requires initialization.
- 0x09: Initializing
- 0x00: Stopped – The status when in the “Reset” mode, or in the “Uninitialized” mode and the converter does not require initialization.
- 0x01: Filling Pre-Trigger buffer
- 0x02: Waiting for start trigger
- 0x03: Sampling/Waiting for stop trigger
- 0x04: Filling Post-Stop buffer
- 0x05: Wait to re-arm – Waiting until local FIFO is empty so the pre-trigger buffer can be filled.
- 0x07: Done capturing

6.6.3 CLK_SRC (READ/WRITE)

Selects the source for the clock of this function block.

- 0x00: System clock/immediate
- 0x01: Never
- 0x02: CLK_BUS2
- 0x03: CLK_BUS3
- 0x04: CLK_BUS4
- 0x05: CLK_BUS5
- 0x06: CLK_BUS6
- 0x07: CLK_BUS7
- 0x08: Reserved
- 0x09: Reserved
- 0x0A: CLK_BUS2 Inverted
- 0x0B: CLK_BUS3 Inverted
- 0x0C: CLK_BUS4 Inverted
- 0x0D: CLK_BUS5 Inverted
- 0x0E: CLK_BUS6 Inverted
- 0x0F: CLK_BUS7 Inverted

6.6.4 START_TRIG (READ/WRITE)

Selects the start trigger from the clock bus. CLK_DIV will start counting after the start trigger.

Refer to CLK_SRC section above, for list of valid values.

6.6.5 CLK_DIV (READ/WRITE)

Divider for the pacer clock. Pacer Clock Frequency = (Clk_Src_Frequency) / (1 + CLK_DIV). If synchronizing with the pacer clock from another Function Block (by using one of the CLK_BUS signals), this is typically set to 0.

6.6.6 CLK_DIV_CNTR (READ ONLY)

The current value of the Clock Divide Counter. This counter starts at a value of CLK_DIV, and counts down. When it reaches zero, a sample is taken. This is useful when using a slow sample clock.

6.6.7 SAMPLE_CNT (READ ONLY)

Total number of samples collected. This does not increment in while in the “Waiting For Start Trigger” state. It also continues counting after a Re-Arm.

6.6.8 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.

6.6.9 INT_STAT (READ/CLEAR)

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.

B0: Capture Window – Indicates state of window, high when during capture and low when during delay

B1: Reserved

B2: Reserved

B3: Start Trigger

B4: Reserved

B5: Reserved

B6: Reserved

B7: Reserved

6.6.10 CLK_BUSN

Selects a source to drive onto Clock Bus N. That clock bus can then be used by a different function block as a clock source or trigger.

A function block can drive multiple different Clock Buses. However, a Clock Bus N should not be driven by more than one function block at the same time or the clock signal will be undefined.

B[7:0]:

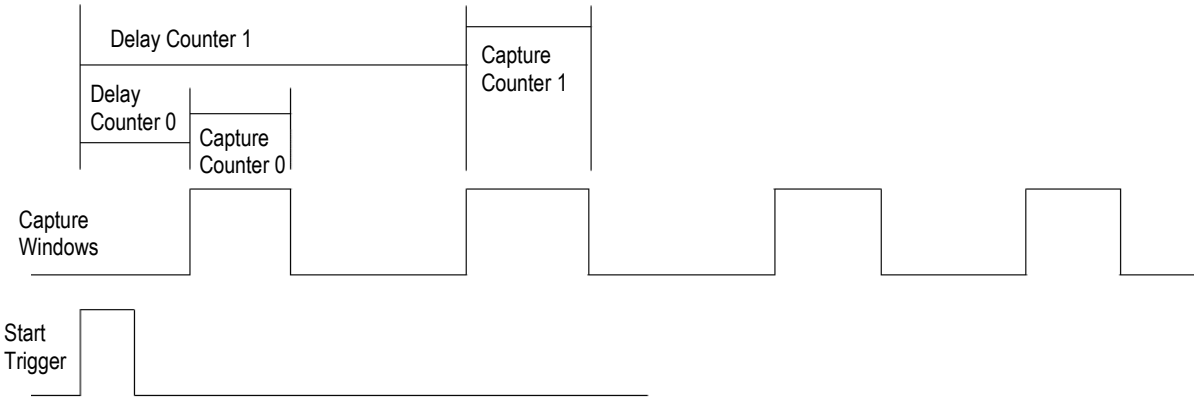
- 0x00: Disabled (Default)
- 0x80: Capture Window – Indicates state of window, high when during capture and low when during delay
- 0x81: Reserved
- 0x82: Reserved
- 0x83: Start Trigger
- 0x84: Reserved
- 0x85: Reserved
- 0x86: Reserved

6.6.11 CONFIG (MASKABLE READ/WRITE)

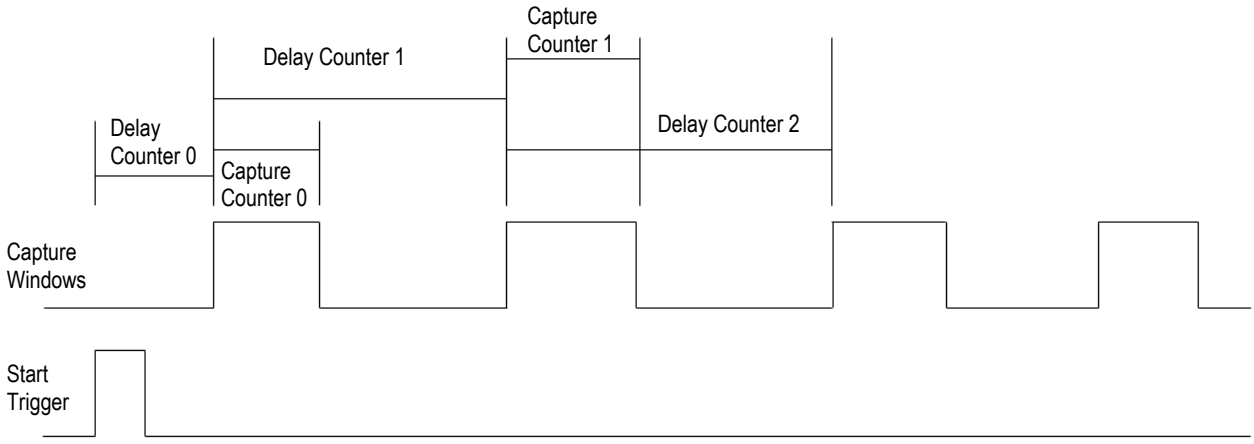
This register provides configuration for the Capture Window.

B[1:0]: Mode

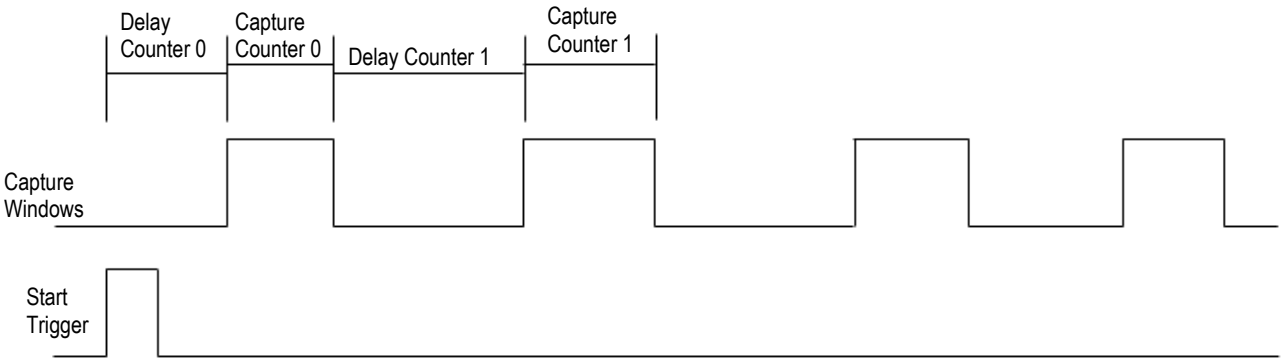
- 00:



- 01:



- 10:



6.6.12 DELAY_COUNTER_N (READ/WRITE)

Sets the width of delay counter n. Delay Counter = (DELAY_COUNTER_n) / (Pacer Clock).

6.6.13 CAPTURE_COUNTER_N (READ/WRITE)

Sets the width of capture counter n. Capture Counter = (CAPTURE_COUNTER_n) / (Pacer Clock).

6.7 BAR2: SyncBus

This function block provides an interface to the SyncBus. It is used clock synchronization between multiple boards.

6.7.1 Function Block Register Map

Table 24: SyncBus Functional Block

	Offset	0x03	0x02	0x01	0x00
Header	FB + 0x00	FB_ID			
	FB + 0x04	FB_DMA_BUFFERS	FB_DMA_CHANNELS	*Reserved*	*Reserved*
SyncBus Control	FB + 0x08	ENABLE	TERM_ENABLE	PLL_LOCKED	CLK_SEL
	FB + 0x0C	DIRECTION	SB_2_CLK_SRC	SB_1_ CLK_SRC	SB_0_ CLK_SRC
	FB + 0x10	CLK_BUS3	CLK_BUS2	*Reserved*	*Reserved*
	FB + 0x14	CLK_BUS7	CLK_BUS6	CLK_BUS5	CLK_BUS4
	FB + 0x18	DIG_OUT	DIG_IN	SYNCBUS_MODE	PLL_SYNC

6.7.2 CLK_SEL (Read)

Select if the system clock is using the local clock or the SyncBus clock. If the SyncBus PLL loses its lock, this register will revert back to the local clock.

B0: Select main clock input, 0 = local clock, 1 = SyncBus clock.

6.7.3 PLL_LOCKED (Read Only)

This register indicates when the SyncBus PLL is locked, which means that the input SyncBus clock is valid, and may be used.

B0: SyncBus PLL locked, 0 = not locked, 1 = locked.

6.7.4 TERM_ENABLE (Read/Write)

Selects if termination is enable on the SyncBus. When using the SyncBus the first and last module in the SyncBus needs to have termination enabled.

B0: Enable SyncBus termination, 0 = disabled, 1 = enabled.

6.7.5 ENABLE (Read/Write)

Select if the SyncBus driver is enable. When the SyncBus driver is disabled, the transceiver is powered down and SyncBus will not receive or transmit. This bit must be enabled in order to use the SyncBus.

B0: Enable SyncBus driver, 0 = disabled (low power), 1 = enabled.

6.7.6 SB_N_CLK_SRC (READ/WRITE)

Select source for SYNCBUS_N line output.

Below is the list of clock sources and the register value needed to select the source.

B[7:0]:

- 0x00: System clock/immediate
- 0x01: Never
- 0x02: CLK_BUS2
- 0x03: CLK_BUS3
- 0x04: CLK_BUS4
- 0x05: CLK_BUS5
- 0x06: CLK_BUS6
- 0x07: CLK_BUS7
- 0x08: N/A
- 0x09: N/A
- 0x0A: CLK_BUS2 Inverted
- 0x0B: CLK_BUS3 Inverted
- 0x0C: CLK_BUS4 Inverted
- 0x0D: CLK_BUS5 Inverted
- 0x0E: CLK_BUS6 Inverted
- 0x0F: CLK_BUS7 Inverted

6.7.7 DIRECTION (READ/WRITE)

Selects the direction of SYNCBUS_N.

NOTE: When synchronizing multiple boards, enabling multiple outputs on a SyncBus line may result in expected errors.

B0: Selects the direction SYNCBUS_0 line, 0 =Input, 1 = Output.

B1: Selects the direction SYNCBUS_1 line, 0 =Input, 1 = Output.

B2: Selects the direction SYNCBUS_2 line, 0 =Input, 1 = Output.

B3: Selects the direction SYNCBUS_3 (CLK) line, 0 =Input, 1 = Output.

6.7.8 CLK_BUSN

Selects the source to drive onto Clock Bus signal N. Values are:

B[7:0] Clock Source Select

- 0x00: Disable Clock Source
- 0x80: CLK_EVENT_SYNCBUS_0
- 0x81: CLK_EVENT_SYNCBUS_1
- 0x82: CLK_EVENT_SYNCBUS_2

6.7.9 PLL_SYNC (READ/WRITE)

This register is used to issue a PLL sync signal across multiple DM34116HR. When enabling the pll sync mode, SYNCBUS_0 is repurposed to send/receive the pll sync signal. Refer to section xxx for more information.

B0: SYNC_Enable, 0 = disabled (Normal SyncBus Operation), 1 = Allows Clock Synchronization

B3: Go/Busy(Sticky Register) When this bit is high (‘1’) it is ready to synchronize the boards. Writing a ‘1’ to this bit will start the process. While synchronization is in process this bit will be low (‘0’), once the bit returns

to ‘1’ the boards have been synchronized. This process should only be issued by the module that is driving SYNCBUS_0

6.7.10 SYNCBUS_MODE (READ/WRITE)

The SyncBus may be configured to be used as either SyncBus connections or GPIO. To set the mode for each signal pair, set the corresponding bit for that to ‘0’ for SyncBus mode or ‘1’ for GPIO mode.

- B[7:4]: *Reserved*
- B[3]: 0 = SYNC3p/n for SyncBus mode, 1 = GPIO mode.
- B[2]: 0 = SYNC2p/n for SyncBus mode, 1 = GPIO mode.
- B[1]: 0 = SYNC1p/n for SyncBus mode, 1 = GPIO mode.
- B[0]: 0 = SYNC0p/n for SyncBus mode, 1 = GPIO mode.

6.7.11 DIG_IN (READ-ONLY)

When the SyncBus is configured for GPIO mode (with the SYNCBUS_MODE register), the contents of the given DIG_IN register bit will display the data read on the SyncBus pair which corresponds to that bit.

- B[7:4]: *Reserved*
- B[3]: Data read from SYNC3p/n when the SyncBus is configured for GPIO mode.
- B[2]: Data read from SYNC2p/n when the SyncBus is configured for GPIO mode.
- B[1]: Data read from SYNC1p/n when the SyncBus is configured for GPIO mode.
- B[0]: Data read from SYNC0p/n when the SyncBus is configured for GPIO mode.

6.7.12 DIG_OUT (READ/WRITE)

When the SyncBus is configured for GPIO mode (with the SYNCBUS_MODE register), and the SyncBus pair of interest is enabled to have data driven on it (with the ENABLE register), the contents of the given DIG_OUT register bit will then be driven to the SyncBus pair which corresponds to that bit.

- B[7:4]: *Reserved*
- B[3]: Data be to driven on SYNC3p/n when the SyncBus is configured for GPIO mode and SYNC_OUT_EN is set.
- B[2]: Data be to driven on SYNC2p/n when the SyncBus is configured for GPIO mode and SYNC_OUT_EN is set.
- B[1]: Data be to driven on SYNC1p/n when the SyncBus is configured for GPIO mode and SYNC_OUT_EN is set.
- B[0]: Data be to driven on SYNC0p/n when the SyncBus is configured for GPIO mode and SYNC_OUT_EN is set.

6.8 BAR2: Programmable Clock

This function block provides an interface to the Programmable Clock.

6.8.1 Function Block Register Map

Table 25: Programmable Clock Functional Block

	Offset	0x03	0x02	0x01	0x00
Header	FB + 0x00	FB_ID			
	FB + 0x04	FB_DMA_BUFFERS	FB_DMA_CHANNELS	FB_CONTROL_SIZE	Reserved
Control	FB + 0x08	Reserved	START_TRIG	CLK_SRC	MODE_STATUS
	FB + 0x0C	CLK_DIV			
	FB + 0x10	CLK_DIV_CNTR			
	FB + 0x14	Reserved			
	FB + 0x18	Reserved			
	FB + 0x1C	SAMPLE_CNT			
	FB + 0x20	INT_ENA (Conversion, Start, Stop, Error, Channel)			
	FB + 0x24	INT_STAT		Reserved	
	FB + 0x28	CLK_BUS3	CLK_BUS2	Reserved	Reserved
	FB + 0x2C	CLK_BUS7	CLK_BUS6	CLK_BUS5	CLK_BUS4
	FB + 0x30	CONFIG (Maskable register – 16-bit)			
	FB + 0x08 + (FB_CONTROL_SIZE * 0x04)	FIFO_DATA_CNT			
	FB + 0x0C + (FB_CONTROL_SIZE * 0x04)	MAX_FIFO_SIZE			
	FB + 0x10 + (FB_CONTROL_SIZE * 0x04)	INTERVAL_CLK_DIV			
	FB + 0x14 + (FB_CONTROL_SIZE * 0x04)	LAST_INTERVAL_COUNT			
	FB + 0x1C + (FB_CONTROL_SIZE * 0x04)	FIFO_ACCESS			

6.8.2 Mode_Status (Read/Write, Read-Only)

Selects the current mode of operation and indicates its triggering status.

B[3:0]: Mode

- 0x04: Uninitialized. This is the power-on state. No converter initialization has taken place. Sampling is stopped, and all counters are reset and the triggering state machine is reset. Transition to any of the other Modes will start converter initialization (sampling will not start until initialization is complete).
- 0x00: Reset. Counting is stopped. All counters are reset and the triggering state machine is reset.
- 0x01: Paused. Counting is stopped, but the counters and triggering state machine maintain their state.
- 0x02: Go, Single-Shot. After filling the buffer with the Post-Stop samples, capturing stops. The Mode must be set back to RESET in order to capture more samples.
- 0x03: Go, Re-arm. After filling the buffer with the Post-Stop samples and the FIFO is empty, the triggering state machine is restarted, i.e. FIFO is filled with Pre-Start samples and waits for a start trigger.

B[7:4]: Status

- 0x08: Uninitialized – The status when in the “Uninitialized” mode and the converter requires initialization.
- 0x09: Initializing
- 0x00: Stopped – The status when in the “Reset” mode, or in the “Uninitialized” mode and the converter does not require initialization.
- 0x01: Reserved
- 0x02: Waiting for start trigger
- 0x03: Running/Waiting for stop trigger
- 0x04: Reserved
- 0x05: Reserved
- 0x07: Done capturing

6.8.3 CLK_SRC (READ/WRITE)

Selects the source for the clock of this function block.

0x00: System clock/immediate
0x01: Never
0x02: CLK_BUS2
0x03: CLK_BUS3
0x04: CLK_BUS4
0x05: CLK_BUS5
0x06: CLK_BUS6
0x07: CLK_BUS7
0x08: Reserved
0x09: Reserved
0x0A: CLK_BUS2 Inverted
0x0B: CLK_BUS3 Inverted
0x0C: CLK_BUS4 Inverted
0x0D: CLK_BUS5 Inverted
0x0E: CLK_BUS6 Inverted
0x0F: CLK_BUS7 Inverted

6.8.4 START_TRIG (READ/WRITE)

Selects the start trigger from the clock bus. CLK_DIV will start counting after the start trigger, unless PRE_START_COUNT is non-zero in which case CLK_DIV will start counting immediately.

Refer to CLK_SRC section above, for list of valid values.

6.8.5 STOP_TRIG (READ/WRITE)

Selects the stop trigger from the clock bus.

Refer to CLK_SRC section above, for list of valid values.

6.8.6 CLK_DIV (READ/WRITE)

Divider for the pacer clock. Pacer Clock Frequency = (Clk_Src_Frequency) / (1 + CLK_DIV). If synchronizing with the pacer clock from another Function Block (by using one of the CLK_BUS signals), this is typically set to 0.

6.8.7 CLK_DIV_CNTR (READ ONLY)

The current value of the Clock Divide Counter. This counter starts at a value of CLK_DIV, and counts down. When it reaches zero, a sample is taken. This is useful when using a slow sample clock.

6.8.8 COUNT (READ ONLY)

Total number of events that have occurred. This does not increment in while in the “Waiting For Start Trigger” state. It also continues counting after a Re-Arm.

When MODE_CONFIG(1) = ‘1’ this register will reset this register to 0 every interval clock pulse.

6.8.9 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.

6.8.10 INT_STAT (READ/CLEAR)

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.

B0: Count – The count has changed.

B1: Interval Counter – Interval count has been reached

B2: Reserved

B3: Start Trigger

B4: Stop Trigger

B5: Reserved

B6: Reserved

B7: Reserved

6.8.11 CLK_BUSN

Selects a source to drive onto Clock Bus N. That clock bus can then be used by a different function block as a clock source or trigger.

A function block can drive multiple different Clock Buses. However, a Clock Bus N should not be driven by more than one function block at the same time or the clock signal will be undefined.

B[7:0]:

- 0x00: Disabled (Default)
- 0x80: Count – The count has changed.
- 0x81: Interval Counter – Interval count has been reached.
- 0x82: Reserved
- 0x83: Start Trigger
- 0x84: Stop Trigger
- 0x85: Reserved
- 0x86: Reserved
- 0x87: Reserved

6.8.12 MODE_CONFIG (MASKABLE READ/WRITE)

This register provides configuration for Programmable Clock function block.

B[1]: Interval Count Reset: When enabled this causes the Count register to reset to 0 every interval clock pulse.

- ’0’ = Interval Count Reset is disabled
- ’1’ = Interval Count Reset is enabled

B[0]: Interval Counter Enable: Used to enable/disable the interval counter.

- ’0’ = Interval Counter is disabled
- ’1’ = Interval Counter is enabled

6.8.13 CHN_FIFO_DATA_CNT (READ)

This register shows the current sample count that is available in the interval counter FIFO.

6.8.14 CHN_MAX_FIFO_SIZE (READ)

This register shows the max number of samples that the interval counter FIFO can hold.

6.8.15 INTERVAL_CLK_DIV (READ/WRITE)

Divider for the interval clock. Interval Clock Frequency = (System Clock) / (1 + INTERVAL_CLK_DIV). When the interval counter is enabled, every interval clock pulse will update the LAST_INTERVAL_COUNT and generate an interval counter interrupt.

6.8.16 LAST_INTERVAL_COUNT (READ-ONLY)

The last count value taken from the COUNT register. This is the same value that is written to the DMA FIFO.

6.8.17 CHN_FIFO_ACCESS (READ/WRITE)

This register provides direct access to the DMA FIFO. It can be used to access the data without the use of the DMA engine. The DMA engine for this channel must be set to "Pause." Each register access advances to the next sample.

7 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.

8 Additional Information

8.1 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

8.2 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

9 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.

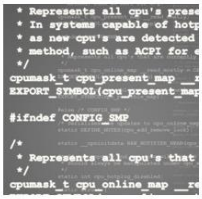
Some states do not allow the exclusion or limitation of incidental or consequential damages for consumer products, and some states do not allow limitations on how long an implied warranty lasts, so the above limitations or exclusions may not apply to you.

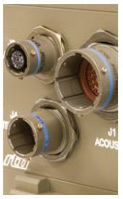
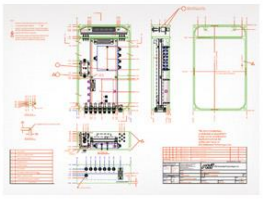
This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

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 2019 by RTD Embedded Technologies, Inc. All rights reserved.