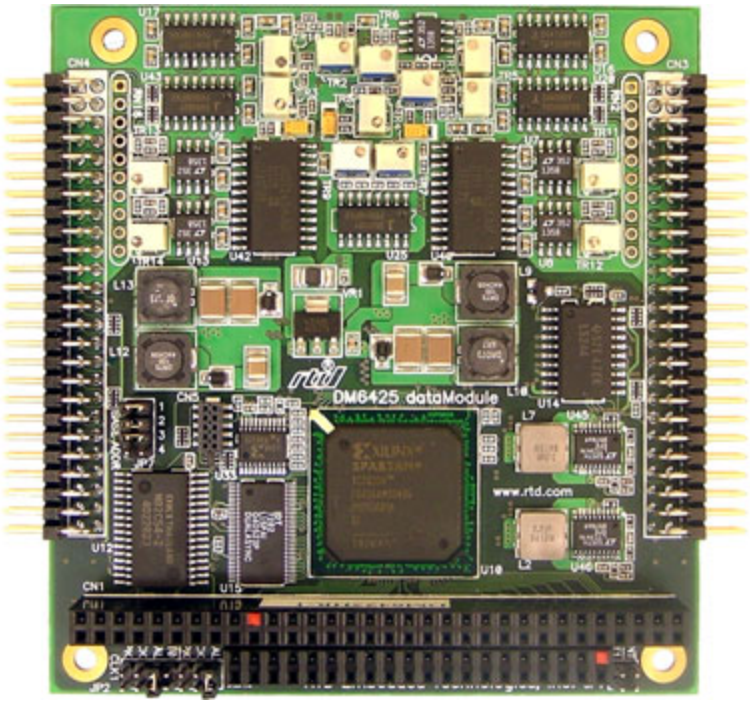
DM6425HR 32-Channel, 500 kHz PC/104 dataModule®



User's Manual

BDM-610010034 Revision D

www.rtd.com

An ISO9001:2000 Company

"Accessing the Analog World"®

DM6425HR 32-Channel, 500 kHz PC/104 dataModule® User's Manual

RTD Document Number: BDM-610010034 Revision D
Copyright © 2008 RTD Embedded Technologies, Inc. All rights reserved.

Trademarks

Advanced Digital I/O, aDIO, a2DIO, Autonomous SmartCal, cpuModule, dspFramework, dspModule, IDAN, HiDAN, HiDANplus, "MIL Value for COTS prices", multiPort, and PC/104EZ are trademarks, and "Accessing the Analog World", dataModule, RTD, and the RTD logo are registered trademarks of RTD Embedded Technologies, Inc. PS/2, PC/XT, PC/AT, and IBM are trademarks of International Business Machines Inc. MS-DOS, Windows, Windows 95, Windows 98, and Windows NT are trademarks of Microsoft Corp. Linux is a registered trademark of Linus Torvalds. M-Systems and DiskOnChip are trademarks or registered trademarks of M-Systems Flash Disk Pioneers, Ltd. PC/104 is a registered trademark of PC/104 Consortium. All other trademarks appearing in this document are the property of their respective owners.

Contents and specifications within this manual are subject to change without notice.

Revision History

Revision	Date	Reason for Change
A	2/23/06	Initial release.
B	3/2/06	Corrected Write Register to show BA + 0x02h, bit 4, reserved; revised bit field description accordingly.
C	11/5/08	Corrected examples in *Special Considerations* on page 111. Simplified formulas in Table 8, *Analog Trigger Threshold Values* — 80
D	3/16/15	Explained that the FPGA is disconnected from the digital I/O lines at initial power up and until FPGA configuration is complete. See: *JS1, JS2, JS3, and JS4—Pull-Up/Pull-Down Resistors on Digital I/O Lines* on page 18. Corrected the IDAN part number to reflect the double high frame.

DM6425 32-Channel, 500 kHz PC/104 dataModule®

www.rtd.com

An ISO9001:2000 Company

"Accessing the Analog World"®

Contact Information

RTD Embedded Technologies, Inc.
103 Innovation Blvd.
State College, PA 16803-0906
USA

Phone: +1-814-234-8087
Fax: +1-814-234-5218

E-mail: sales@rtd.com
techsupport@rtd.com

Internet: http://www.rtd.com

Table of Contents

Chapter 1 Introduction

DM6425HR Ordering Information 3
- DM6425HR Model Options 3
- DM6425HR Accessories 4

Features 5

Major Functions 6

Analog-to-Digital Conversion 7

A/D Conversion Circuitry 7
- Analog Inputs 7
- Channel-Gain Scan Memory 7
- A/D Converter 8
- 1024 Sample Buffer 8
- Data Transfer 8

Digital-to-Analog Conversion 8

8254 Timer/Counters 9

Digital I/O 10

For More Information 11

Chapter 2 DM6425HR Module Settings

Factory-Configured Jumper and Solder Blob Settings 14
- JP1—CN3, Pin 43 Signal Select (Factory Setting: OT1) 16
- JP2—User TC Clock Source Select (Factory Settings—CLK 0: XTAL; CLK 1: OT0) 16
- Base Address Jumper (Factory Setting: 300 hex, 768 decimal) 17
- JS1, JS2, JS3, and JS4—Pull-Up/Pull-Down Resistors on Digital I/O Lines 18

Chapter 3 Installing the dataModule

DM6425HR Connectors 20

Installation Considerations 21

Before You Begin 21

Installing the DM6425HR in a PC/104 Stack 22

External I/O Connections 23
- Connecting the Analog Input Pins 25
  - Single-Ended Mode 25
  - Differential Mode 26
- Connecting the Module for Simultaneous Sampling 28
- Connecting the Analog Outputs 28
- Connecting the Timer/Counters and Digital I/O 28

Chapter 4 I/O Mapping

Defining the I/O Map 30

BA + 0x00h (0): Clear/Program Clear Register (Read/Write) 36
BA + 0x02h (2): Read Status/Program Control Register (Read/Write) 38
BA + 0x04h (4): Read Converted Data/Load Channel-Gain and Digital Data (Read/Write) 41
Load Channel-Gain Latch (BA + 0x02h, bits 1 and 0 = 00) 41
Load Channel-Gain Table in Channel-Gain Scan Memory (BA + 0x02h, bits 1 and 0 = 01) 43
Load Digital Table in Channel-Gain Scan Memory (BA + 0x02h, bits 1 and 0 = 10) 44
BA + 0x06h (6): Start Convert/Program Trigger Modes (Read/Write) 44
BA + 0x08h (8): Program IRQ Source and Channel (Write) 46
BA + 0x0Ah (10): Digital Input FIFO/Program Digital Input FIFO Configuration Register (Read/Write) 47
BA + 0x0Eh (14): A/D Sample Counter Trigger (Read) 47
BA + 0x10h (16): TC Counter 0 (Read/Write) 48
BA + 0x12h (18): TC Counter 1 (Read/Write) 48
BA + 0x14h (20): TC Counter 2 (Read/Write) 48
BA + 0x16h (22): Timer/Counter Control Word (Write Only) 48
BA + 0x18h (24): Digital I/O Port 0, Bit Programmable Port (Read/Write) 49
BA + 0x1Ah (26): Digital I/O Port 1, Byte Programmable Port (Read/Write) 49
BA + 0x1Ch (28): Read/Program Port 0 Direction/Mask/Compare Registers (Read/Write) 49
BA + 0x1Eh (30): Digital IRQ Status/Program Digital Mode (Ports 0 and 1, Read/Write) 51
BA + 0x400h (1024): D/A Analog Setup (Read/Write) 52
BA + 0x402h (1026): Simutaneous Update Register (Write) 52
BA + 0x404h (1028): DAC1 Update Register (Write) 52
BA + 0x406h (1030): DAC2 Update Register (Write) 52
BA + 0x408h (1032): DAC3 Update Register (Write) 53
BA + 0x40Ah (1034): DAC4 Update Register (Write) 53
BA + 0x410h (1040): Analog Threshold Detector Setup (Read/Write) 53
BA + 0x418h (1048): Digital I/O Port 2, Bit Programmable Port (Read/Write) 54
BA + 0x41Ah (1050): Digital I/O Port 3, Byte Programmable Port (Read/Write) 54
BA + 0x41Ch (1052): Read/Program Port 2 Direction/Mask/Compare Registers (Read/Write) 55
BA + 0x41Eh (1054): Digital IRQ Status/Program Digital Mode (Ports 2 and 3, Read/Write) 56
BA + 0x800h/801h (2048/2049): Board ID Register (Read) 57
BA + 0x802h (2050): Board ID Reset (Read) 57

Programming the DM6425HR 58
Clearing and Setting Bits in a Port 58

Chapter 5 A/D Conversions

Before Starting Conversions 62
Initializing the Module 62
Programming Channel, Gain, Input Range, and Type 63
Programming the Channel-Gain Table 64
16-Bit A/D Table 64
Channel Select, Gain Select, Input Range, and Input Type 64
Pause Bit 65
Skip Bit 65
8-Bit Digital Table 66
Setting Up A/D and Digital Tables 66
Using the Channel-Gain Table for A/D Conversions 67
Channel-Gain Table and Throughput Rates 67

A/D Conversion Modes 68
Start A/D Conversions 68
Start/Stop Trigger Select 70
Triggering a Burst Sample 70
Trigger Repeat Function 71
Pacer Clock Source 71
Types of Conversions 71
Single Conversion 71
Multiple Conversions 72
Random Channel Scan 72
Programmable Burst 73
Programmable Multi-Scan 73
Starting an A/D Conversion 74
Monitoring Conversion Status (FIFO Empty Flag or End-of-Convert) 74
Halting Conversions 74
Reading the Converted Data 74
Using the A/D Data Markers 76
Programming the Pacer Clock 76
Selecting 16-bit or 32-bit Pacer Clock 76
Programming Steps 77
Programming the Burst Clock 78
Programming the Sample Counter 78
Using the Sample Counter to Create Large Data Arrays 79
Analog Trigger 80

Chapter 6 Data Transfers Using DMA

Choosing a DMA Channel 84
Allocating a DMA Buffer 85
Calculating the Page and Offset of a Buffer 85
Setting the DMA Page Register 87
The DMA Controller 87
DMA Mask Register 88
DMA Mode Register 88
Programming the DMA Controller 88
Programming the DM6425HR for DMA 89
Monitoring for DMA Done 89
Dual DMA Mode 89
Common DMA Problems 89

Chapter 7 Interrupts

Software Selectable Interrupt Sources 92
Software Selectable Interrupt Channel 92
Advanced Digital Interrupts 92
Event Mode 92
Match Mode 92

Sampling Digital Lines for Change of State 92

Basic Programming For Interrupt Handling 93

What Is an Interrupt? 93
Interrupt Request Lines 93
8259 Programmable Interrupt Controllers 93
Interrupt Mask Registers (IMR) 94
End-of-Interrupt (EOI) Command 94
What Happens When an Interrupt Occurs? 94
Using Interrupts in Your Programs 94
Writing an Interrupt Service Routine (ISR) 94
Saving the Startup Interrupt Mask Register (IMR) and Interrupt Vector 96
Restoring the Startup IMR and Interrupt Vector 97
Common Interrupt Mistakes 97

Chapter 8 D/A Conversions

Chapter 9 Timer/Counters

Chapter 10 Digital I/O

Port 0, Bit Programmable Digital I/O 106

Direction Register: 106
Advanced Digital Interrupts: Mask and Compare Registers 106

Port 1, Port Programmable Digital I/O 106

Port 2, Bit Programmable Digital I/O 106

Direction Register: 106
Advanced Digital Interrupts: Mask and Compare Registers 106

Port 3, Port Programmable Digital I/O 107

Resetting the Digital Circuitry 107

Strobing Data into Port 0 107

High-Speed Digital Input 107

Appendix A Example Programs

Controlling the DM6425HR via Software 110

RTD Provided Software 110
Direct Register Control 110
Special Considerations 111

Appendix B Calibration

Required Equipment 114

A/D Calibration 115

Bipolar Calibration 115
Bipolar Range Adjustment: –5 to +5 V 115
Bipolar Range Adjustment: –10 to +10 V 115
Unipolar Calibration 116
Gain Adjustment 117

D/A Calibration 118

Appendix C Specifications

Appendix D I/O Connector Pin Assignments

Appendix E IDAN™ Dimensions and Pinout

IDAN-DM6425HR-62D 128
IDAN-DM6425HR-62D Dimensions and Connector 128
IDAN-DM6425HR-62D External I/O Connections 129

IDAN-DM6425HR-68D 130
IDAN-DM6425HR-68D Dimensions and Connector 130
IDAN-DM6425HR-68D External I/O Connections 131

Appendix F Limited Warranty

Chapter 1 Introduction

The DM6425HR 32-channel, 12-bit, 500 kHz PC/104 dataModule turns your IBM AT-compatible cpuModule™ or other PC/104 computer into a high-speed, high performance data acquisition and control system.

This manual is intended to help you install your new module and get it running quickly, while also providing enough detail about the module and its functions so that you can enjoy maximum use of its features even in the most complex applications. RTD assumes that you already have an understanding of data acquisition principles and that you can customize the example software or write your own application programs.

This manual is organized as follows:

Chapter 1 ***Introduction***
introduces main features and specifications

Chapter 2 ***DM6425HR Module Settings***
provides information on factory-default and user-configurable jumper settings

Chapter 3 ***Installing the dataModule***
provides information on installing the DM6425HR

Chapter 4 ***I/O Mapping***
describes the I/O map for the DM6425HR

Chapter 5 ***A/D Conversions***
shows how to program the DM6425HR to perform A/D conversions and read the results

Chapter 6 ***Data Transfers Using DMA***
explains how data transfers are accomplished using Direct Memory Access (DMA)

Chapter 7 ***Interrupts***
explains software selectable interrupts, digital interrupts, and basic interrupt programming

Chapter 8 ***D/A Conversions***
explains how to perform D/A conversions on the DM6425HR

Chapter 9 ***Timer/Counters***
explains the two 8254 timer/counter circuits on the DM6425HR

Chapter 10 ***Digital I/O***
explains the bit- and port-programmable digital I/O circuitry on the DM6425HR

Appendix A ***Example Programs***
discusses the example programs included with the DM6425HR

Appendix B ***Calibration***
describes how to calibrate the DM6425HR using the 6425DIAG diagnostic program included in the example software package, and using the trimpots on the dataModule

Appendix C ***Specifications***
provides the typical characteristics of the DM6425HR

Appendix D ***I/O Connector Pin Assignments***
provides the pinout for the DM6425HR I/O connectors

Appendix E ***IDAN™ Dimensions and Pinout***
provides connector pinouts for the DM6425HR installed in an RTD Intelligent Data Acquisition Node (IDAN™) frame

Appendix F ***Limited Warranty***

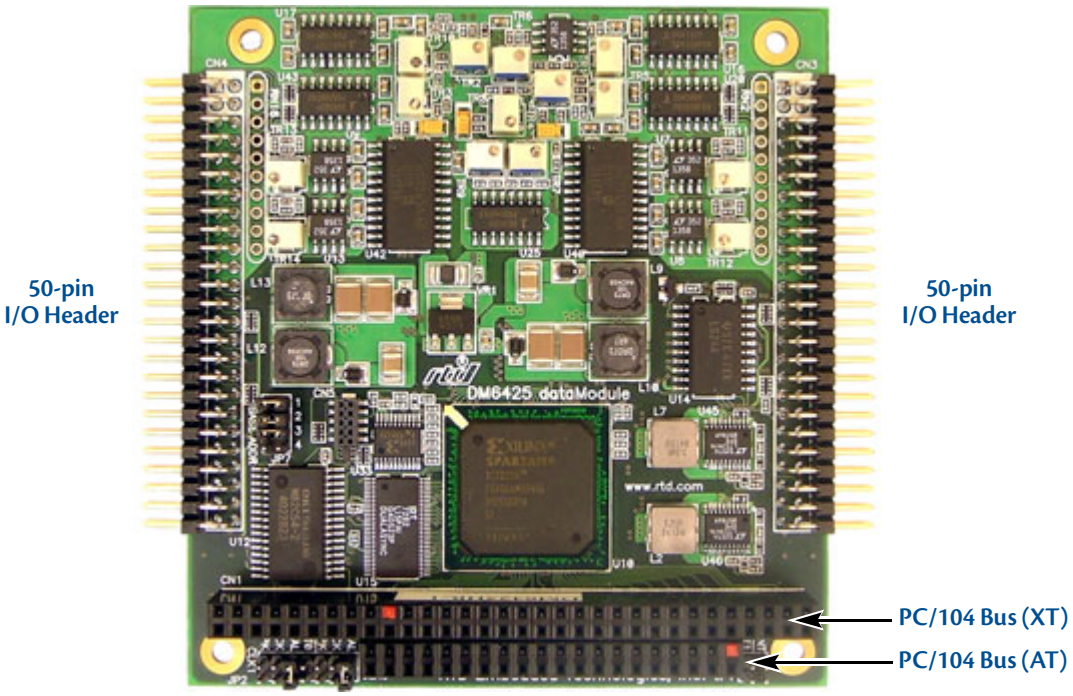


Figure 1 DM6425HR dataModule (top view)

DM6425HR Ordering Information

The DM6425HR 32-channel, 12-bit, 500 kHz PC/104 dataModule is available with both a 1K FIFO buffer and an 8K FIFO buffer. The DM6425HR can also be purchased as part of an Intelligent Data Acquisition Node (IDAN™) building block, which consists of the dataModule and a milled aluminum IDAN frame. The IDAN building block can be used in just about any combination with other IDAN building blocks to create a simple but rugged PC/104 stack. Refer to Appendix E, *IDAN™ Dimensions and Pinout*, for more information. The DM6425HR can also be purchased as part of a custom-built RTD HiDAN™ or HiDANplus High Reliability Intelligent Data Acquisition Node. Contact RTD for more information on its high reliability PC/PCI-104 systems.

DM6425HR Model Options

The DM6425HR model options are shown below. Refer to the RTD website (www.rtd.com) for more detailed ordering information.

Table 1 DM6425HR dataModule Options

Part Number	Description
DM6425HR-1	32-channel, 12-bit, 500 kHz PC/104 dataModule with a 1K FIFO buffer on A/D converter
DM6425HR-8	32-channel, 12-bit, 500 kHz PC/104 dataModule with an 8K FIFO buffer on A/D converter

Each DM6425HR package contains the following items:

- DM6425HR dataModule
- Companion CD containing software and documentation

DM6425HR Accessories

In addition to the items included in your DM6425HR package, several software and hardware accessories are available. Hardware accessories include IDAN frames, power supplies, terminal boards, and other connection systems. Contact RTD or your distributor, or visit the RTD website at www.rtd.com, for more information and for advice on selecting the most appropriate application software, drivers, and hardware accessories to support your control system.

Table 2 Recommended Accessories[1]

Part Number	Description
XT50	3-foot, 50-pin, twisted-pair cable
TB50	50-pin screw terminal board
XB50	50-pin screw terminal extender board
TMX32	Analog input expansion board with thermocouple compensation (can expand a single input channel to 16 differential or 32 single-ended input channels)
OP8/OP16	Opto-isolated, digital input boards (provide 8/16 opto-isolated digital inputs)
MR8/MR16	Mechanical relay output boards (provide 8/16 electro-mechanical relay outputs)
OR16	Mechanical relay and opto-isolated input board (provides 8 electro-mechanical relays outputs and 8 opto-isolated digital inputs)
TS16	J or K thermocouple input expansion board (multiplexes 16 differential thermocouple inputs into one channel)
DM16	PC/104 to ISA bus extender board for testing your module in a conventional desktop computer

1. Refer to the RTD website (www.rtd.com) for more details.

Features

The DM6425HR is a 12-bit, 500 kHz analog I/O dataModule with either 16 differential or 32 single-ended analog input channels, 4 analog output channels, and 4 independent 8-bit digital I/O ports.

Ultra-compact for embedded and portable applications, the DM6425HR complete feature set includes:

- Analog threshold triggering on any one channel
- 16 differential or 32 single-ended analog input channels
- 12-bit, 2 µs analog-to-digital converter with 500 kHz throughput
- Programmable input ranges: ± 5, ± 10, or 0 to +10 V
- Programmable gains of 1, 2, 4, and 8
- 1024 x 24 channel-gain scan memory with skip bit
- Software, pacer clock, and external trigger modes
- Scan, burst, and multi-burst using the channel-gain table
- 16-bit programmable high-speed sample counter
- A/D DMA transfer
- 1024 sample A/D buffer for gap-free, high-speed sampling under Windows™ and DOS
- Pre-, post-, and about-trigger modes
- 3-bit analog input data/trigger marker
- two 8-bit programmable digital I/O lines with Advanced Digital Interrupt modes
- 1024 sample digital input buffer for gap-free, high-speed sampling under Windows™ and DOS
- two byte programmable digital I/O lines
- Six 16-bit timer/counters (two available to user) and onboard 8 MHz clock
- Four 12-bit D/A output channels
- ± 5, 0 to +5, ±10, or 0 to +10 V analog output range
- Programmable interrupt source
- +5 V operation
- Windows™ and Linux® example programs
- DOS example programs with source code
- Diagnostics software

Major Functions

The DM6425HR has four major circuits:

- the A/D
- the D/A
- the timer/counters
- the digital I/O lines

The following sections describe these circuits. Subsequent chapters provide a detailed discussion of module functions. Figure 2 provides a block diagram of the DM6425HR.

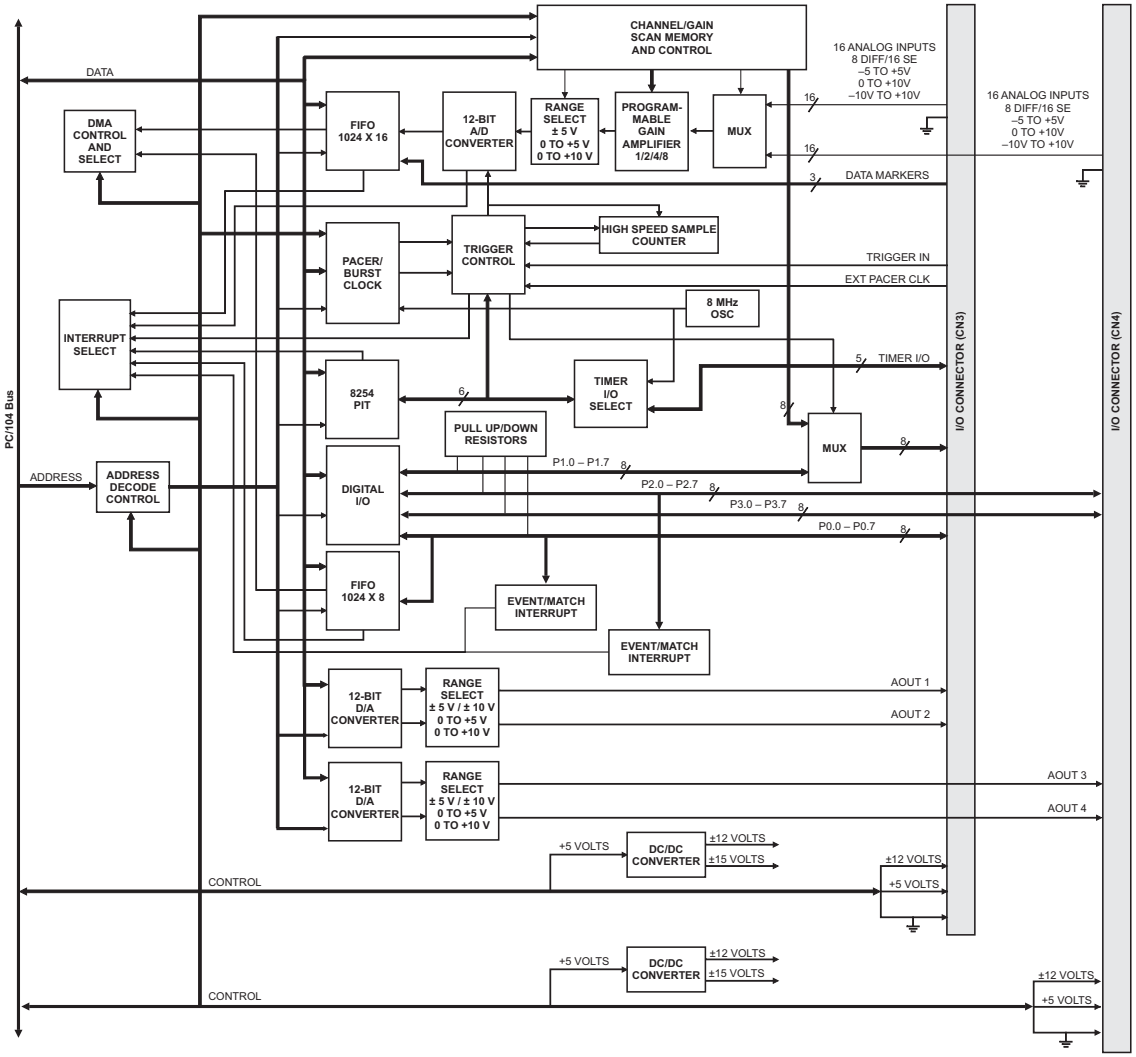


Figure 2 DM6425HR Block Diagram

Analog-to-Digital Conversion

The DM6425HR is software configurable on a channel-by-channel basis for up to 16 differential or 32 single-ended analog inputs. Software programmable unipolar and bipolar input ranges and gains allow easy interfacing to a wide range of sensors. Overvoltage protection to ± 12 V is provided at the inputs. The common mode input voltage for differential operation is ± 10 V.

Analog-to-digital (A/D) conversions are typically performed in 2 µs, and the maximum throughput rate of the board is 500 kHz. Conversions are controlled by software command, by an onboard pacer clock, by using triggers to start and stop sampling, or by using the sample counter to acquire a specified number of samples. Several trigger sources can be used to turn the pacer clock on and off, giving you exceptional flexibility in data acquisition. Scan, burst, and multi-burst modes are supported by using the channel-gain scan memory. A first in, first out (FIFO) sample buffer helps your computer manage the high throughput rate of the A/D converter by acting as an elastic storage bin for the converted data. Even if the computer does not read the data as fast as conversions are performed, conversions can continue until the FIFO is full.

The converted data can be transferred to PC memory in one of three ways. Direct Memory Access (DMA) transfer supports conversion rates of up to 500,000 samples per second. Data also can be transferred using the programmed I/O mode or the interrupt mode. A special interrupt mode using a REP INS (Repeat Input String) instruction supports very high-speed data transfers. By generating an interrupt when the FIFO's half-full flag is set, a REP INS instruction can be executed, transferring data to PC memory and emptying the FIFO buffer at the maximum rate allowed by the data bus.

The mode of transfer and DMA channel are chosen through software. The PC data bus is used to read and/or transfer data to PC memory. In the DMA transfer mode, you can make continuous transfers directly to PC memory without going through the processor.

A/D Conversion Circuitry

The DM6425HR performs analog-to-digital conversions on up to 16 software-selectable analog input channels.

Analog Inputs

The input voltage range is software programmable for –5 to +5 V, –10 to +10 V, or 0 to +10 V. Software-programmable binary gains of 1, 2, 4, and 8 let you amplify lower level signals to more closely match the module's input ranges. Overvoltage protection to ±12 V is provided at the inputs.

Channel-Gain Scan Memory

The channel-gain scan memory lets you sample channels in any order, at high speeds, with a different gain on each channel. This 1024 x 24-bit memory supports complex channel-gain scan sequences, including digital output control. Using the digital output control feature, you can control external input expansion boards such as the TMX32 to expand channel capacity up to 512 channels. When used, these control lines are output on Port 1. When the digital lines are not used for this feature, they are available for other digital control functions.

A skip bit is provided in the channel-gain data word to support different sampling rates on different channels. When this bit is set, the A/D conversion is not written to the FIFO on the selected channel. Chapter 4 and Chapter 5 describe this feature in detail.

A/D Converter

The 12-bit successive approximation A/D converter accurately digitizes dynamic input voltages in 2 µs, for a maximum throughput rate of 500 kHz. The converter IC contains a sample-and-hold amplifier, a 12-bit A/D converter, a 2.5 V reference, a clock, and a digital interface to provide complete A/D conversion on a single chip. Its low power CMOS logic combined with a high-precision, low-noise design provides accurate results.

Conversions are controlled by software command, by pacer clock, by using triggers to start and stop sampling, or by the sample counter to acquire a specified number of samples. An onboard or external pacer clock can be used to control the conversion rate. Chapter 5, *A/D Conversions*, describes conversion modes.

1024 Sample Buffer

A first in, first out (FIFO) 1024 sample buffer helps your computer manage the high throughput rate of the A/D converter by providing an elastic storage bin for the converted data. Even if the computer does not read the data as fast as conversions are performed, conversions will continue until a FIFO full flag is sent to stop the converter.

The sample buffer does not need to be addressed when you are writing to or reading from it; internal addressing makes sure that the data is properly stored and retrieved. All data accumulated in the sample buffer is stored intact until the PC is able to complete the data transfer. Its asynchronous operation means that data can be written to or read from it at any time, at any rate. When a transfer does begin, the data first placed in the FIFO is the first data out.

Data Transfer

The converted data can be transferred to PC memory in one of three ways. Direct Memory Access (DMA) transfer supports conversion rates of up to 500,000 samples per second. Data also can be transferred using the programmed I/O mode or the interrupt mode. A special interrupt mode using a REP INS (Repeat Input String) instruction supports very high-speed data transfers. By generating an interrupt when the FIFO's half full flag is set or when the sample counter counts down, a REP INS instruction can be executed, transferring data to PC memory and emptying the sample buffer at the maximum rate allowed by the data bus.

The PC data bus is used to read and/or transfer data to PC memory. In the DMA transfer mode, you can make continuous transfers directly to PC memory without going through the processor.

The converted data plus a sign bit is stored in the top 13 bits (left-justified) of the 16-bit data word written to the sample buffer. The bottom three bits can be used as a 3-bit data marker as described in Chapter 5.

Digital-to-Analog Conversion

The digital-to-analog (D/A) circuitry features four independent 12-bit analog output channels with software-selectable output ranges of ± 5, 0 to +5, ±10, or 0 to +10 V. The 5 V range has a resolution of 1.22 mV, the 10 V ranges have a resolution of 2.44 mV, and the 20 V range has a resolution of 4.88 mV.

Data is programmed into the D/A converter by writing one 12-bit value. D/A outputs are automatically updated when the data is written. Access through DMA is not available. By setting a bit in a register, all of the D/As can be simultaneously updated, if so desired.

8254 Timer/Counters

Two 8254 programmable interval timers provide six (three each) 16-bit, 8 MHz timer/counters to support a wide range of board operations and user timing and counting functions.

One 8254 is the Clock TC, which is used for board operations. Two of its 16-bit timer/counters, Counter 0 and Counter 1, are cascaded and used internally for the pacer clock. The pacer clock is described in Chapter 5. The third timer/counter in the Clock TC, Counter 2, is the burst clock. Figure 3 shows the Clock TC circuitry.

The other 8254 is the User TC, which has two 16-bit timer/counters (Counter 0 and Counter 1) for user functions and one 16-bit timer/counter (Counter 2) for the sample counter. Figure 4 shows the User TC circuitry.

Each 16-bit timer/counter has two inputs, CLK in and GATE in, and one output, timer/counter OUT. Each can be programmed as binary or BCD down counters by writing the appropriate data to the command word, as described in Chapter 4. The command word also lets you set up the mode of operation. The six programmable modes are:

Mode 0	Event Counter (Interrupt on Terminal Count)
Mode 1	Hardware-Retriggerable One-Shot
Mode 2	Rate Generator
Mode 3	Square Wave Mode
Mode 4	Software-Triggered Strobe
Mode 5	Hardware Triggered Strobe (Retriggerable)

Note *The pacer clock, burst clock, and sample counter should be programmed for Mode 2 operation.*

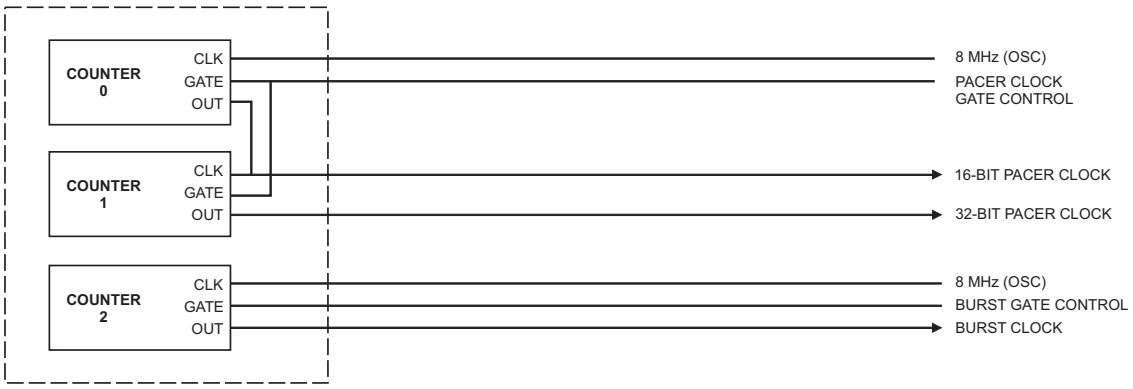


Figure 3 Clock TC Circuit Block Diagram

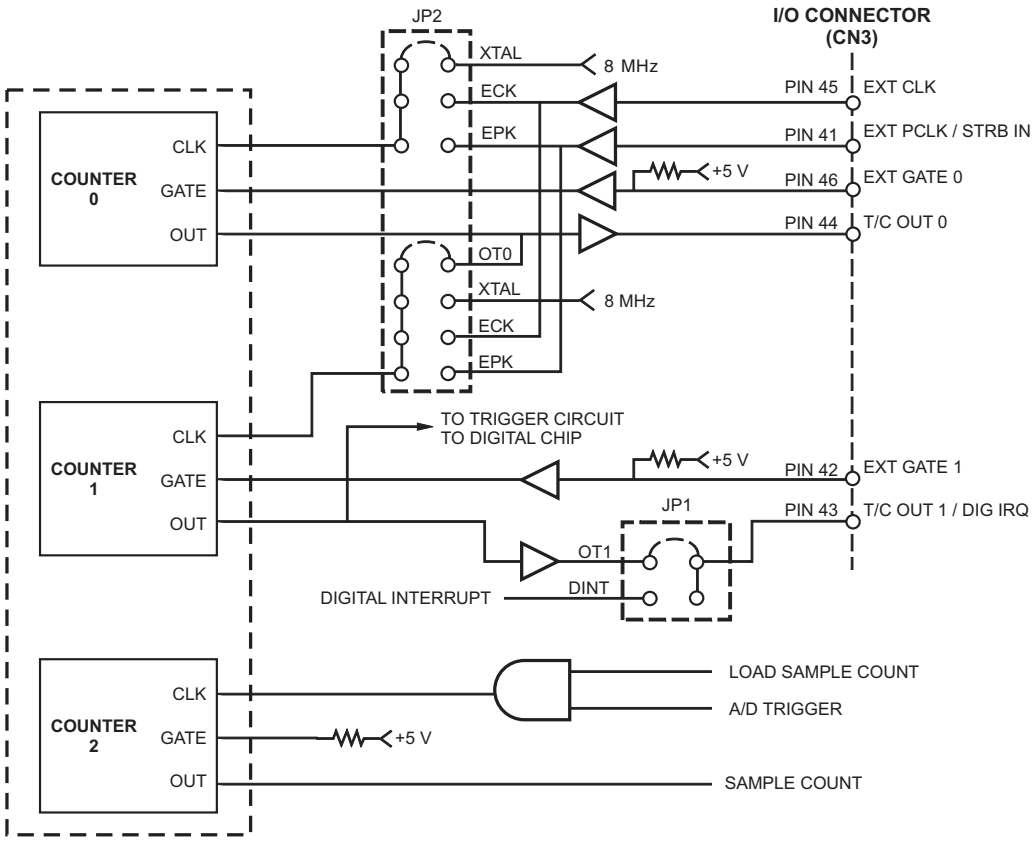


Figure 4 User TC Circuit Block Diagram

Digital I/O

The DM6425HR has 32 buffered TTL/CMOS digital I/O lines, which can be used to transfer data between the computer and external devices. These digital I/O lines are comprised of four independent ports of 8 bits each.

Port 0 and Port 2 provide eight bit programmable lines that can be independently set for input or output. Port 0 and Port 2 support RTD's two Advanced Digital Interrupt modes. An interrupt can be generated when any bit changes value (event interrupt), or when the lines match a programmed value (match interrupt). For either mode, a Mask Register lets you monitor selected lines for interrupt generation. A 1024 sample buffer is also connected to the Port 0 lines. This sample buffer can be used to sample inputs up to a 500 kHz rate.

Port 1 and Port 3 can be programmed as a byte input or output port.

Pull-up or pull-down resistors are provided for all 32 lines. Chapter 2, *DM6425HR Module Settings*, provides instructions for activating these pull-up/pull-down resistors.

Chapter 7 explains digital interrupts, and Chapter 10 details digital I/O operations.

For More Information

This manual and the example programs in the software package included with your dataModule provide enough information to properly use all of the module's features. If you have any problems installing or using this dataModule, contact the RTD Technical Support Department during regular business hours, eastern standard time or eastern daylight time, or send a FAX requesting assistance. When sending a FAX request, please include your company's name and address, your name, your telephone number, and a brief description of the problem. You can also contact RTD via e-mail.

RTD Embedded Technologies, Inc.
103 Innovation Blvd.
State College, PA 16803-0906
USA

Phone: +1-814-234-8087
Fax: +1-814-234-5218

E-mail: techsupport@rtd.com
sales@rtd.com

Internet: http://www.rtd.com

Chapter 2 DM6425HR Module Settings

The DM6425HR has jumper and switch settings that you can change if necessary for your application. The module is factory-configured as listed in Table 3. If you need to change these settings, use these instructions before you stack the module with your computer system.

Note *By placing solder connections at* **JS1**, **JS2**, **JS3**, *and* **JS4** *(located on the bottom of the board), you can configure each set of digital I/O lines to be pulled up or pulled down. This procedure is explained in JS1, JS2, JS3, and JS4—Pull-Up/Pull-Down Resistors on Digital I/O Lines on page 18.*

Factory-Configured Jumper and Solder Blob Settings

Figure 5 shows the DM6425HR top layout and the locations of the factory-set jumpers. Figure 6 shows the DM6425HR bottom layout and the locations of the factory-set solder blobs. Table 3 lists the factory settings of the user-configurable jumpers and solder blobs.

Note *Pay special attention to the base address setting to avoid address contention when you first use the DM6425HR in your system.*

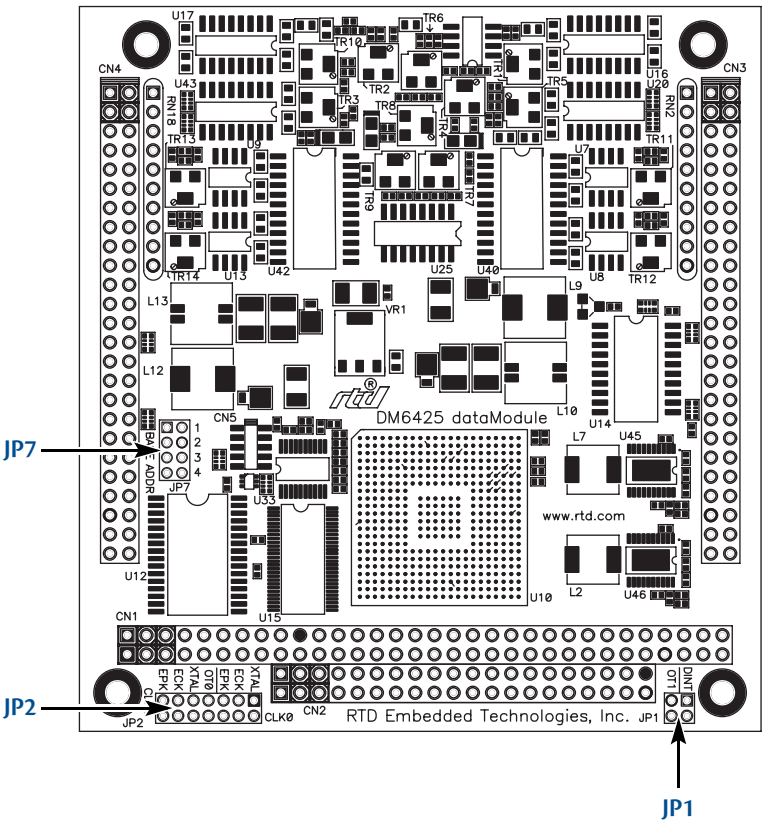


Figure 5 DM6425HR Jumper Settings (top view)

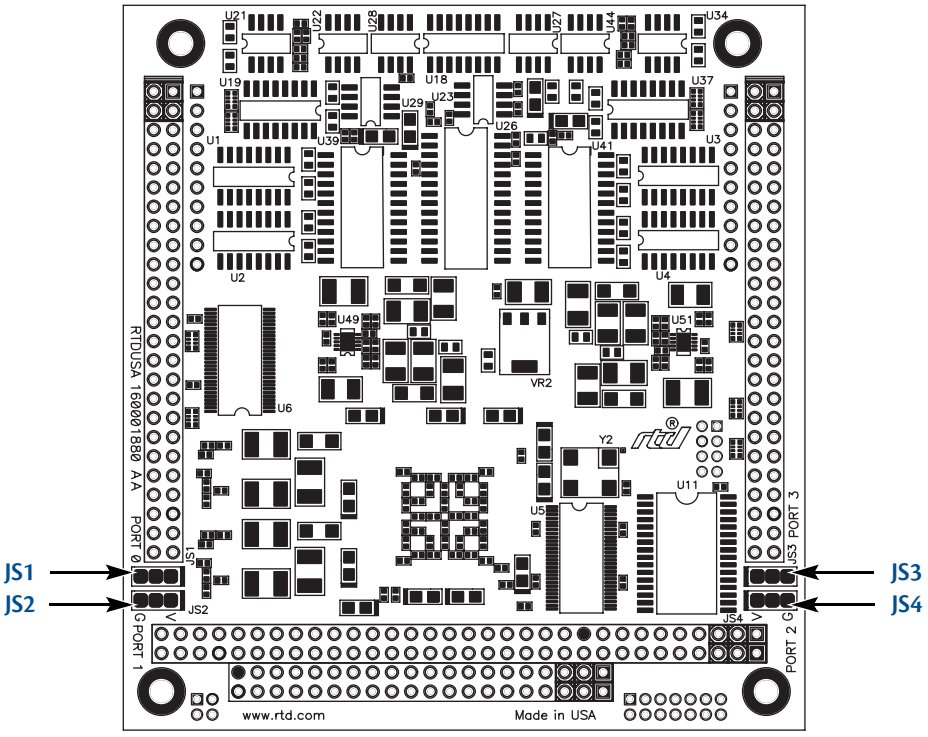


Figure 6 DM6425HR Solder Blobs (bottom view)

Table 3 DM6425HR Factory Jumper and Solder Blob Settings

Jumper/ Solder Blob	Function Controlled	Factory Settings (Jumpers Installed)
JP1	Selects signal available at **CN3**, pin 43	OT1 (User TC, Counter 1)
JP2	Sets clock source for User TC Counters 0 & 1	Clk 0: XTAL; Clk 1: OT0 (timer/counters cascaded)
JP7 (BASE ADDR)	Sets base address	300 hex (768 decimal)
JS1	Activates pull-up/pull-down resistors on Port 0 digital I/O lines	All bits pulled up (solder connections between COM & V)
JS2	Activates pull-up/pull-down resistors on Port 1 digital I/O lines	All bits pulled up (solder connections between COM & V)
JS3	Activates pull-up/pull-down resistors on Port 3 digital I/O lines	All bits pulled up (solder connections between COM & V)
JS4	Activates pull-up/pull-down resistors on Port 2 digital I/O lines	All bits pulled up (solder connections between COM & V)

JP1—CN3, Pin 43 Signal Select (Factory Setting: OT1)

This header connector, shown in Figure 7, lets you select the output signal from the DM6425HR that is present at I/O connector **CN3**, pin 43. You can select the output from the User TC Counter 1 or the Advanced Digital Interrupt from the digital I/O chip. User TC Counter 1 is labeled **OT1** on this header, and the digital interrupt is labeled **DINT**.

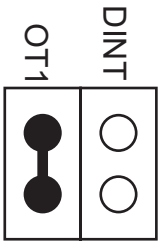


Figure 7 CN3 Pin 43 Signal Select Jumper (JP1)

JP2—User TC Clock Source Select (Factory Settings—CLK 0: XTAL; CLK 1: OT0)

This header connector, shown in Figure 8, lets you select the clock sources for User TC Counters 0 and 1, the 16-bit timer/counters available for user functions. Figure 4 on page 10 shows a block diagram of the User TC circuitry to help you in making these connections.

The three right-most pairs of pins, **XTAL**, **ECK**, or **EPK**, set the clock source for the timer/counter, Counter 0. **XTAL** is the onboard 8 MHz clock; **ECK** is an external clock source that can be connected through I/O connector **CN3**, pin 45; and **EPK** is an external pacer clock that can be connected through I/O connector **CN3**, pin 41.

The four left-most pins, **OT0**, **XTAL**, **ECK**, and **EPK**, set the clock source for timer/counter, Counter 1. **OT0** is the output of Counter 0; **XTAL** is the onboard 8 MHz clock; **ECK** is an external clock source that can be connected through I/O connector **CN3**, pin 45; and **EPK** is an external pacer clock that can be connected through I/O connector **CN3**, pin 41.

Counters 0 and 1 are factory set as a 32-bit cascaded counter clocked by the 8 MHz system clock.

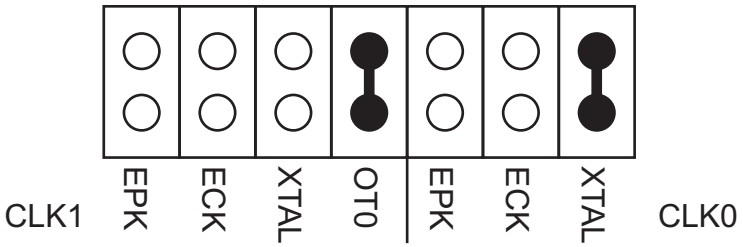


Figure 8 User TC Clock Sources Jumper (JP2)

Base Address Jumper (Factory Setting: 300 hex, 768 decimal)

One of the most common causes of failure when you are first trying your module is address contention. Some of your computer's I/O space is already occupied by internal I/O and other peripherals. When the module attempts to use I/O address locations already used by another device, contention results and the board does not work.

To avoid this problem, the DM6425HR has an easily accessible jumper (**J7**) that lets you select any one of 16 starting addresses in the computer's I/O. If the factory setting of 300 hex (768 decimal) is unsuitable for your system, you can select a different base address simply by setting the jumper to any one of the values listed in Table 4. The table shows the jumper settings and their corresponding hexadecimal and decimal (in parentheses) values. Be sure to verify the order of the numbers on the jumper (1 through 4) before setting them. When you set the base address for your module, record the value on the back cover of this manual. Figure 9 shows the BASE ADDR jumper (**J7**) set for a base address of 300 hex (768 decimal).

Note *In Table 4, 0 = closed or down, corresponding to a logical "0", and 1 = open or up, corresponding to a logical "1". The factory default base address is shaded.*

Table 4 DM6425HR dataModule Base Address Jumper Settings

Base Address Hex/(Decimal)	Jumper Settings 4 3 2 1	Base Address Hex/(Decimal)	Jumper Settings 4 3 2 1
200/(512)	0 0 0 0	300/(768)	1 0 0 0
220/(544)	0 0 0 1	320/(800)	1 0 0 1
240/(576)	0 0 1 0	340/(832)	1 0 1 0
260/(608)	0 0 1 1	360/(864)	1 0 1 1
280/(640)	0 1 0 0	380/(896)	1 1 0 0
2A0/(672)	0 1 0 1	3A0/(928)	1 1 0 1
2C0/(704)	0 1 1 0	3C0/(960)	1 1 1 0
2E0/(736)	0 1 1 1	3E0/(922)	1 1 1 1

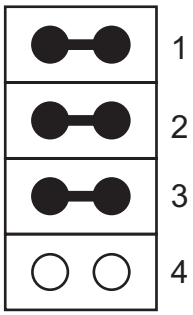


Figure 9 Base Address Jumper (J7)
(shown in the factory default setting)

JS1, JS2, JS3, and JS4—Pull-Up/Pull-Down Resistors on Digital I/O Lines

The DM6425HR has 32 TTL/CMOS compatible digital I/O lines that can be interfaced with external devices. These lines are divided into four groups: Port 0 has eight bit programmable lines, advanced interrupt modes, and bit masking, Port 1 is byte direction programmable, Port 2 is output, and Port 3 is latched input. Resistors are connected to these lines and can be configured as either pull-up or pull-down resistors.

10 kΩ pull-up/pull-down resistors are installed on the DM6425HR, and a solder connection must be made on the bottom of the board to configure their operation. The solder connections are made at **JS1** for Port 0, **JS2** for Port 1, **JS3** for Port 3, and **JS4** for Port 2. The factory default is pull-up for all ports. This is done by placing a solder short between the middle (common) pad and V (+5 V). To configure the resistors as pull-down resistors, remove the existing solder connection and make one between the middle (common) pad and G (ground). To disable the pull-up/pull-down resistor, remove the solder connection.

The FPGA is disconnected from the digital I/O pins at power up until the FPGA configuration is complete which can take several hundred milliseconds. During configuration the pullup/pulldown resistors and any external device connected to the I/O pin will control the state of the I/O lines. Once the FPGA is configured the FPGA will be connected to the I/O pins and all I/Os are set to input.

WARNING *Do not solder all three pads; doing so will damage the board!*

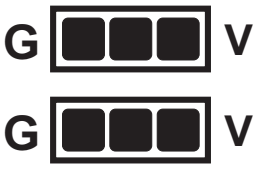


Figure 10 JS1 and JS2 Pull-Up/Pull-Down Resistor Connections

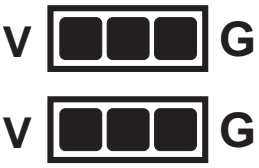


Figure 11 JS3 and JS4 Pull-Up/Pull-Down Resistor Connections

Chapter 3 Installing the dataModule

This chapter provides instructions to install your DM6425HR with an RTD cpuModule™ or other PC/104 based system. After completing the installation, RTD recommends using the diagnostic software included in your DM6425HR package to fully verify that your module is working.

DM6425HR Connectors

Figure 12 shows the connectors of the DM6425HR dataModule.

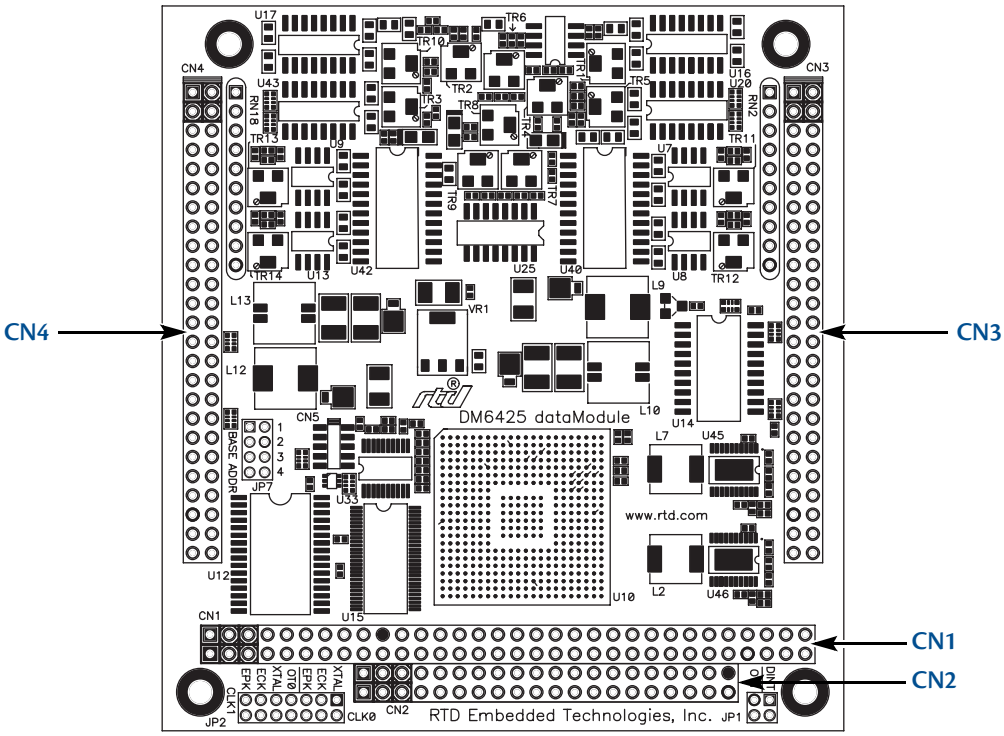


Figure 12 DM6425HR Connector Locations

Note *Pin 1 of each connector is indicated by a square within a white square on both the top and bottom of the board. Pin 1 of the XT and AT connectors match when stacking with other PC/104 modules.*

Table 5 DM6425HR Connectors

Connector	Function	Size
CN1	PC/104 Bus (XT)	64-pin
CN2	PC/104 Bus (AT)	40-pin
CN3	I/O Connector	50-pin
CN4	I/O Connector	50-pin

Installation Considerations

Consider the following points before integrating the DM6425HR dataModule into a PC/104 stack.

1. The DM6425HR is a PC/104 module with an ISA bus (PC XT and AT connectors) and no PCI bus. Therefore, the DM6425HR should not be installed between a PC/104-*Plus* cpuModule and a PC/104-*Plus* peripheral module. Doing so would break the connection of the PC/104-*Plus* bus.
2. RTD recommends building your PC/104 computer by stacking the power supply module and cpuModule on the ends of the stack. These two modules produce heat, and stacking them on the top and bottom maximizes heat dissipation. Placing the cpuModule™ on one end of the stack also minimizes reflections. RTD-built PC/104 systems, such as the Intelligent Data Acquisition Node (IDAN™), normally have a power supply module on the bottom and a cpuModule on the top. Any other combination of PC/104 modules can be integrated into the stack.

Before You Begin

CAUTION *Keep your DM6425HR in the antistatic bag until you are ready to install it to your system!*

Note *Incompatible jumper settings can result in unpredictable operation and erratic response.*

When removing the DM6425HR from the antistatic bag, hold it at the edges and do not touch the components or connectors. Please handle the module in an antistatic environment and use a grounded workbench for testing and handling. Before installing the DM6425HR in your computer, check the jumper settings. Refer to Chapter 2 for factory-default jumper settings and how to change them.

Installing the DM6425HR in a PC/104 Stack

1. Turn off power to your PC/104 system and unplug the cord.
2. Ground yourself with an anti-static strap.
3. Set the Base Address as described in Chapter 2, *DM6425HR Module Settings*.

CAUTION *The DM6425HR should slide into the matching PC/104 connector easily. Do not force the connection. Doing so might bend or break pins.*

4. Line up the pins of the DM6425HR's PC/104 connector with the PC/104 bus of the stack and gently press the module onto the stack.
5. If any modules are to be installed above the DM6425HR, install them.
6. Attach any necessary cables to the PC/104 stack.
7. Reconnect the power cord and apply power to the stack.
8. Boot the system and verify that all of the hardware is working properly.

External I/O Connections

Figure 13 shows the DM6425HR's **CN3** I/O connector pinout. Figure 14 shows the DM6425HR's **CN4** I/O connector pinout. With the dataModule oriented with the PC/104 (ISA) bus in the 6 o' clock position (toward the bottom), Pin 1 is to the left side and is marked with a square pad on the solder (bottom) side of the dataModule. The DM6425HR can be connected to an I/O device using a twisted pair, 50-pin flat cable or discrete wires. Refer to this diagram as you make your I/O connections.

Note *+12 V at pin 47 and –12 V at pin 49 are available only if your computer bus supplies them. (These voltages are not provided by the DM6425HR).*

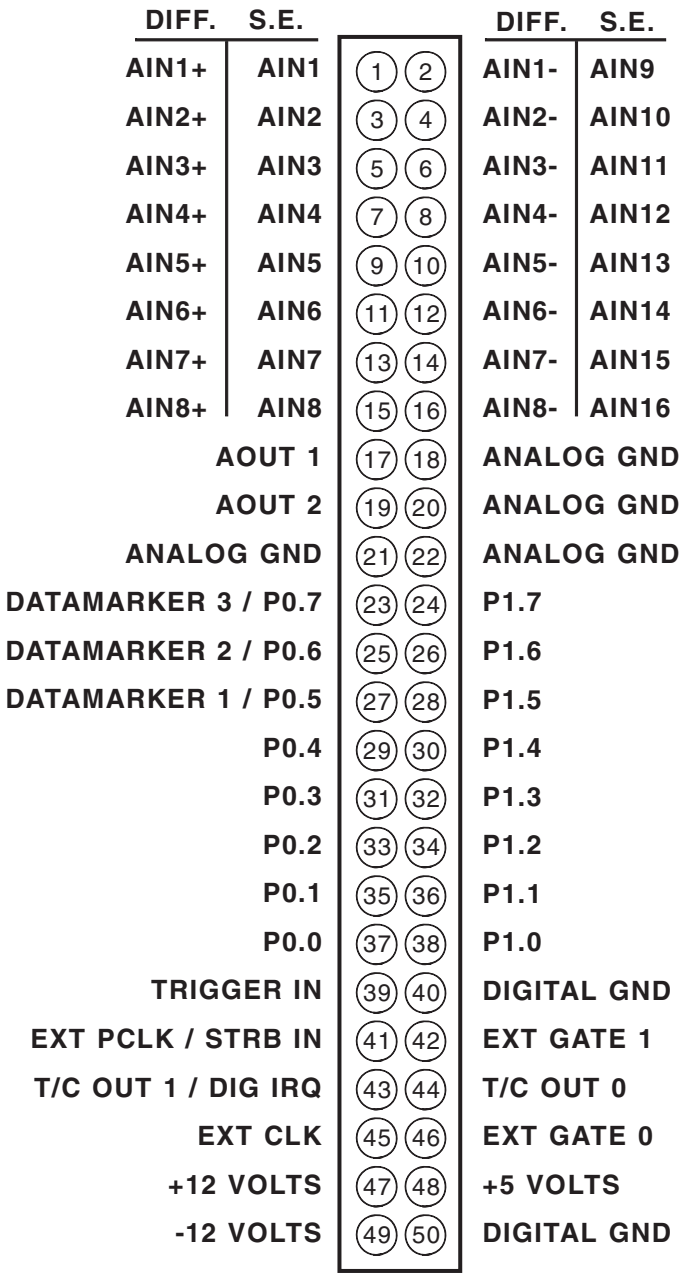


Figure 13 CN3 I/O Connector Pin Assignments

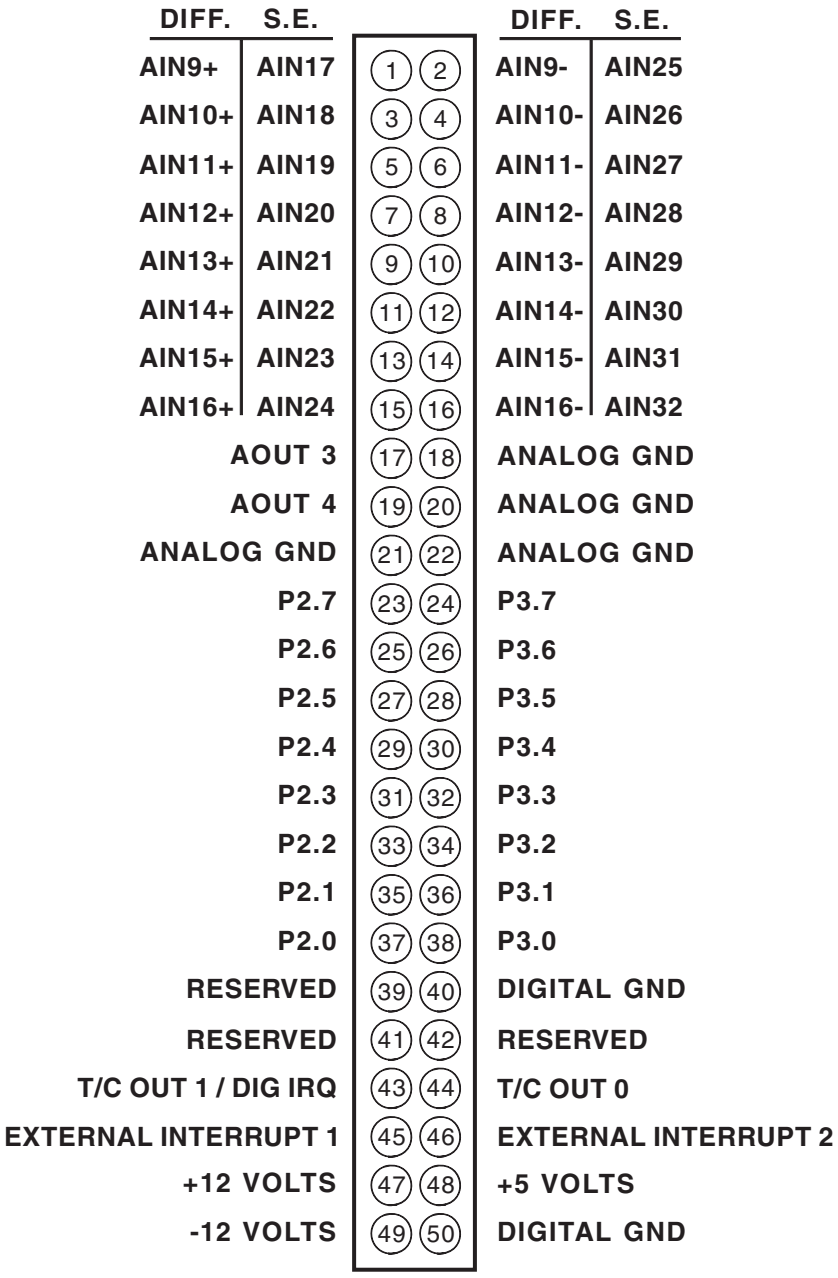


Figure 14 CN4 I/O Connector Pin Assignments

Connecting the Analog Input Pins

The analog inputs on the DM6425HR can be set for single-ended or differential operation. It is good practice to connect all unused channels to ground.

Single-Ended Mode

When operating in single-ended mode, connect the high side of the analog input to one of the analog input channels, AIN1 through AIN32, and connect the low side to an ANALOG GND (pins 18 and 20–22 on **CN3** and **CN4**). Figure 15 shows how these connections are made.

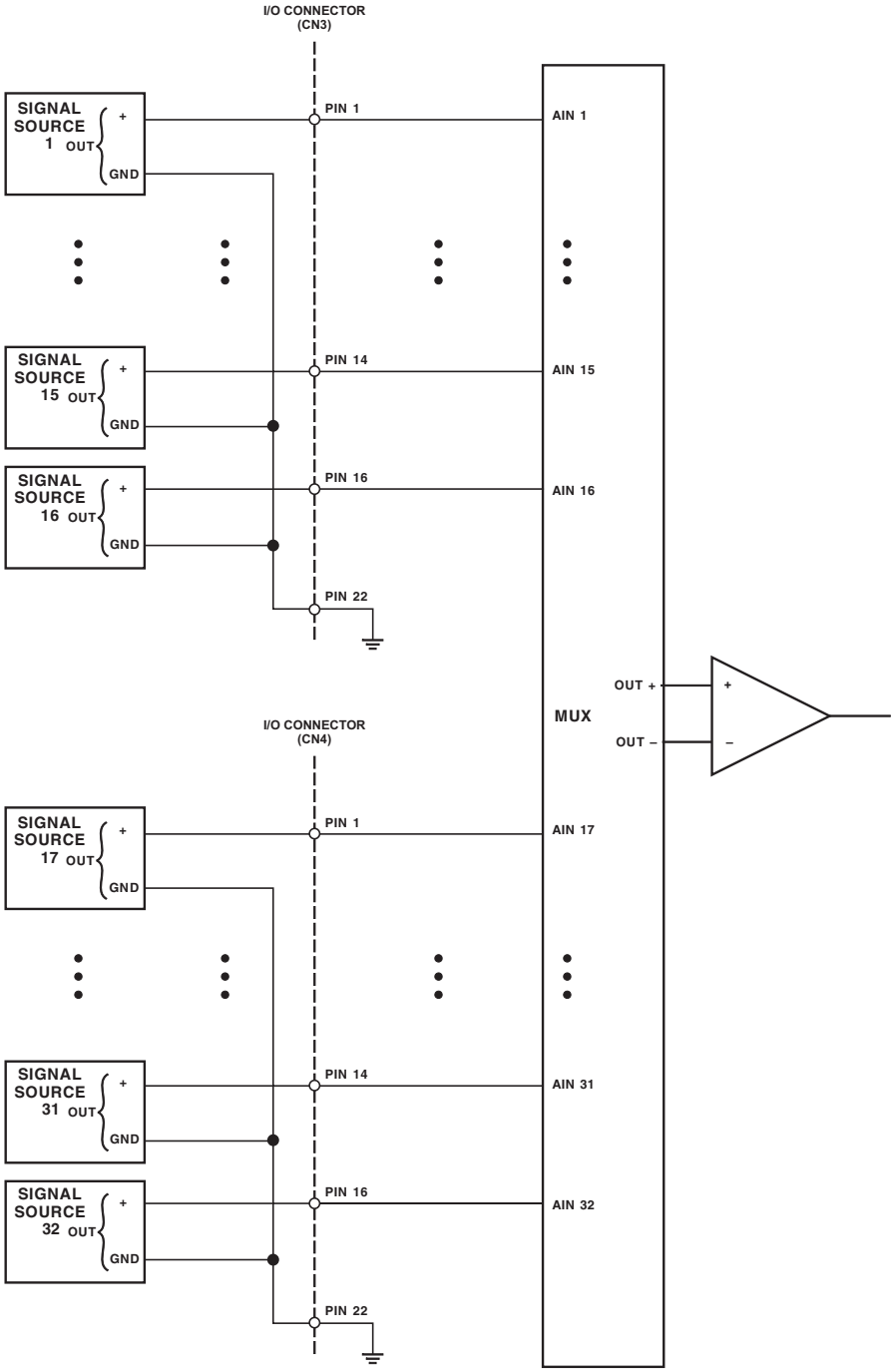


Figure 15 Single-Ended Input Connections

Differential Mode

When operating in differential mode, twisted pair cable is recommended to reduce the effects of magnetic coupling at the inputs. Your signal source may or may not have a separate ground reference. When using the differential mode, you should install a 10 kΩ resistor pack at locations **RN2** and **RN18** on the DM6425HR to provide a reference to ground for signal sources without a separate ground reference.

First, connect the high side of the analog input to the selected analog input channel, AIN1+ through AIN16+, and connect the low side of the input to the corresponding AIN– pin. Then, for signal sources with a separate ground reference, connect the ground from the signal source to an ANALOG GND (pins 18 and 20–22 on **CN3** and **CN4**). Figure 16 shows how these connections are made.

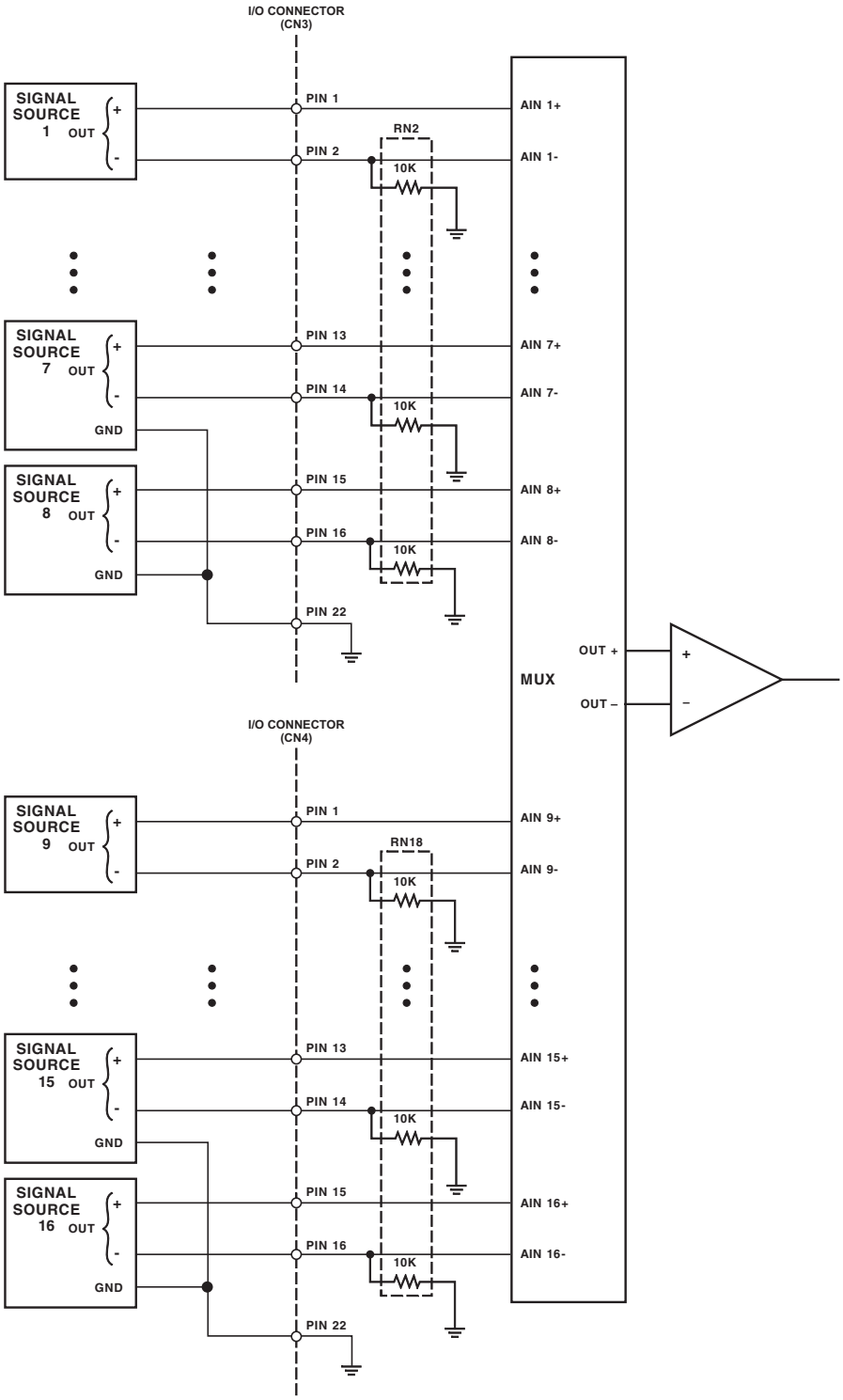


Figure 16 Differential Input Connections

Connecting the Module for Simultaneous Sampling

Multiple modules can be sampled simultaneously by connecting an external trigger source to the TRIGGER IN pin, **CN3** pin 39, and an external pacer clock to the EXT PCLK pin, **CN3** pin 41, of each module. Figure 17 shows how to make these connections.

When applying an external trigger to a module's TRIGGER IN pin, note that the external trigger must be programmed as the start trigger source and the trigger polarity and trigger repeat bits must be configured as desired at the **BA + 0x06h**, Trigger Mode Register. The external trigger pulse duration should be at least 100 ns.

For simultaneous sampling, you must connect the same clock source to each module so that conversions are synchronized. This is accomplished by connecting the same external pacer clock to EXT PCLK, as shown on Figure 17 and selecting the external pacer clock at bit 9 in the Trigger Mode Register programmed at **BA + 0x06h**. The trigger will start the pacer clock, and the pacer clock will simultaneously start conversions on all modules.

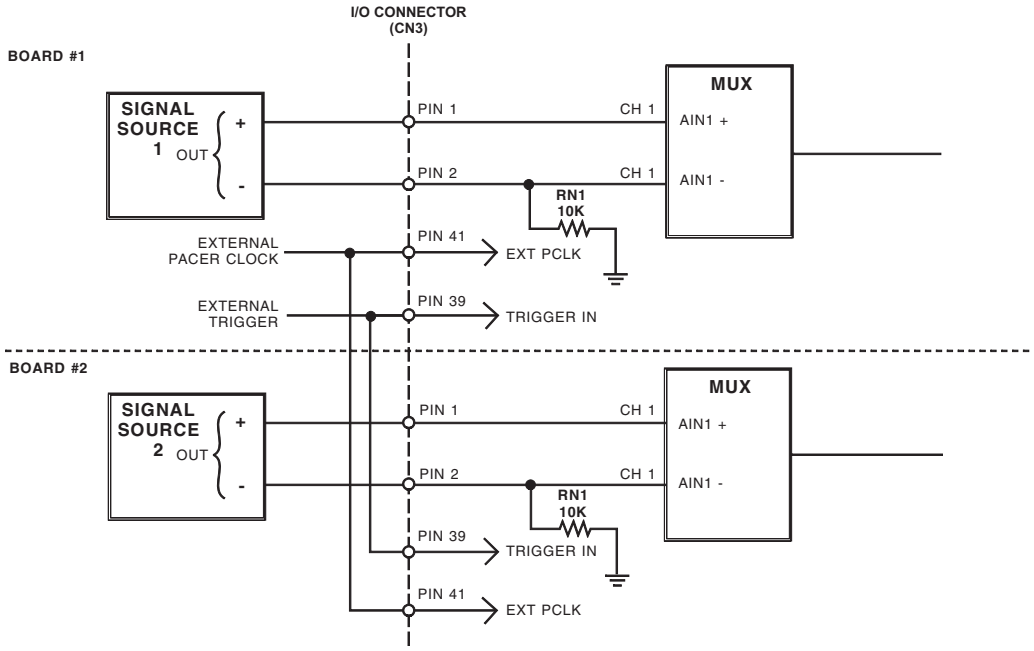


Figure 17 Two Modules Configured for Simultaneous Sampling

Connecting the Analog Outputs

For each of the four D/A outputs, connect the high side of the device receiving the output to the AOUT channel (**CN3**/**CN4**, pin 17, or **CN3**/**CN4**, pin 19) and connect the low side of the device to an ANALOG GND (**CN3**/**CN4**, pin 18, or **CN3**/**CN4**, pin 20).

Connecting the Timer/Counters and Digital I/O

For all of these connections, the high side of an external signal source or destination device is connected to the appropriate signal pin on the I/O connector, and the low side is connected to any DIGITAL GND.

Chapter 4 I/O Mapping

This chapter provides a complete description of the I/O map for the DM6425HR dataModule, general programming information, and how to set and clear bits in a port.

Defining the I/O Map

Tables 6 and 7 provide the DM6425HR read/write registers. As shown, the board occupies 32 consecutive I/O port locations. Because of the 16-bit structure of the AT bus, every other address location is used. RTD's programming structure uses the 16-bit command for reading/writing all locations except for programming the 8254 and digital lines. These require 8-bit read/write operations. The base address (BA) can be selected using the BASE ADDR jumper (**J7**) as described in Chapter 2, *DM6425HR Module Settings*. This switch can be accessed without removing the module from the stack.

The following sections describe the register contents of each address used in the I/O map. BA = base address.

Table 6 DM6425HR Read Registers

BA + Offset hex (dec)	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00h (0)	Reserved				Clear DAC1	Clear DAC2	Clear DAC3	Clear DAC4	Clear IRQ2	Clear IRQ1	Clear Digin FIFO	Reset CGT	Clear CGT	Clear A/D DMA Done Flag	Clear A/D FIFO	Clear Board
0x02h (2)	Reserved	IRQ2 Status	IRQ1 Status	Digin FIFO Full	Digin FIFO Half Full	Digin FIFO Empty	Reserved	About Trig Flag	Pacer Clk Gate Flag	Burst Clk Gate Flag	First DMA Done Flag	A/D DMA Done	EOC Status	A/D Halt Flag	A/D FIFO Full	A/D FIFO Empty
0x04h (4)	Sign Extension	A/D Data												A/D Markers		
0x06h (6)	Start Conversion															
0x0Ah (10)	Reserved								Port 0 Digital Input FIFO							
0x0Eh (14)	Software Trigger for Loading Sample Counter															
0x10h (16)	Base + 0x02h Dependency (bits 6–5)															
	Reserved								Timer 0 LSB/MSB Value							
									Counter 0 LSB/MSB Value							
0x12h (18)	Reserved								Timer 1 LSB/MSB Value							
									Counter 1 LSB/MSB Value							
0x14h (20)	Reserved								Timer 2 LSB/MSB Value							
									Counter 2 LSB/MSB Value							
0x18h (24)	Reserved								Digital I/O Port 0 Value							
0x1Ah (26)	Reserved								Digital I/O Port 1 Value							
0x1Ch (28)	Base + 0x1Eh Dependency (bits 1–0)															
	Reserved								Port 0 Clear Digital IRQ Status Flag							
	Reserved								Port 0 Direction Value							
	Reserved								Port 0 Mask Value							
	Reserved								Port 0 Compare Value							
0x1Eh (30)	Reserved								Port 0 Strobe Status	Port 0 Digital IRQ Status	Port 0 Dig Sample Clk Select	Port 0 Digital IRQ Enable	Port 0 Digital IRQ Mode	Port 1 Direction	Port 0 Register Select	

Table 6 DM6425HR Read Registers (cont'd)

BA + Offset hex (dec)	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x400h (1024)	Reserved							Simult D/A Update	D/A 4 Setup Range/Polarity		D/A 3 Setup Range/Polarity		D/A 2 Setup Range/Polarity		D/A 1 Setup Range/Polarity	
0x410h (1040)		Use Threshold	Value >, ≤ Threshold	Analog Trigger Channel Comparison					Analog Trigger Threshold Value							
0x418h (1048)	Reserved								Digital I/O Port 2 Value							
0x41Ah (1050)	Reserved								Digital I/O Port 3 Value							
0x41Ch (1052)	Base + 0x41Eh Dependency (bits 1–0)															
	Reserved								Port 2 Clear Digital IRQ Status Flag							
	Reserved								Port 2 Direction Value							
	Reserved								Port 2 Mask Value							
	Reserved								Port 2 Compare Value							
0x41Eh (1054)	Reserved								Port 2 Strobe Status	Port 2 Dig IRQ Status	Port 2 Dig Sample Clk Select	Port 2 Dig IRQ Enable	Port 2 Dig IRQ Mode	Port 3 Direction	Port 2 Register Select	
0x800h (2048)	Device ID Vendor ID FPGA Version															
0x802h (2050)	Reset Board ID															

Table 7 DM6425HR Write Registers

Address hex (dec)	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00h (0)	Reserved				Clear DAC1 Enable	Clear DAC2 Enable	Clear DAC3 Enable	Clear DAC4 Enable	Clear IRQ2 Enable	Clear IRQ1 Enable	Clear Digin FIFO Enable	Reset CGT Enable	Clear CGT Enable	Clear A/D DMA Done Enable	Clear A/D FIFO Enable	Clear Board Enable
0x02h (2)	A/D DMA2 Channel Select		A/D DMA1 Channel Select		Reserved	Analog Only Disable	Digital I/O Select for A/D	Pause Enable	Sample Counter Stop Enable	Timer/Counter Select		Reserved	Enable CGT/ Digital Out Table		Load CGT, CG Latch, or Digital Table	
0x04h (4)	Base + 0x02h Dependency (bits 1–0)															
	Reserved					A/D SE/DIFF	Input Range/ Polarity Select		Gain Select			32 Channel Analog Input				
	Reserved			Skip Bit	Pause Bit	A/D SE/DIFF	Input Range/ Polarity Select		Gain Select			32 Channel Analog Input				
	Reserved								Digital Table Output Value							
0x06h (6)	Reserved		Trigger Repeat	Trigger Polarity	Burst Trigger Select		Pacer Clock Int/Ext	Pacer Clock 16-bit/32-bit	Stop Trigger Select			Start Trigger Select			Conversion Select	
0x08h (8)	ISA IRQ Select for IRQ2			IRQ2 Source					ISA IRQ Select for IRQ1			IRQ1 Source				
0x0Ah (10)	Reserved												Digin FIFO Clock Enable	Digin FIFO Clock		
0x10h (16)	Base + 0x02h Dependency (bits 6–5)															
	Reserved								Timer 0 LSB/MSB Value Counter 0 LSB/MSB Value							
0x12h (18)	Reserved								Timer 1 LSB/MSB Value Counter 1 LSB/MSB Value							
0x14h (20)	Reserved								Timer 2 LSB/MSB Value Counter 2 LSB/MSB Value							
0x16h (20)	Reserved								Counter Select for User Timer Counter Select for Counter Timer		Read/Load for User Timer Read/Load for User Counter		Mode Select for User Timer Mode Select for User Counter			BCD/ Binary BCD/ Binary
0x18h (24)	Reserved								Digital I/O Port 0, bit programmable							
0x1Ah (26)	Reserved								Digital I/O Port 1, byte programmable							

Table 7 DM6425HR Write Registers (cont'd)

Address hex (dec)	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1Ch (28)	Base + 0x1Eh Dependency (bits 1–0)															
	Reserved								Port 0/1 Clear Digital I/O Chip Information (value written is a don't care)							
	Reserved								Port 0 Direction (0 = Input)							
	Reserved								Port 0 Mask (1 = Mask)							
	Reserved								Port 0 Compare Value							
0x1Eh (30)	Reserved										Port 0 Dig Sample Clk Select	Port 0 Digital IRQ Enable	Port 0 Digital IRQ Mode	Port 1 Direction	Port 0 Register Select	
0x400h (1024)	Reserved							Simult D/A Update	D/A 4 Setup Range/Polarity		D/A 3 Setup Range/Polarity		D/A 2 Setup Range/Polarity		D/A 1 Setup Range/Polarity	
0x402h (1026)	Reserved				D/A Simultaneous Update											
0x404h (1028)	Reserved				D/A Converter 1 Write											
0x406h (1030)	Reserved				D/A Converter 2 Write											
0x408h (1032)	Reserved				D/A Converter 3 Write											
0x40Ah (1034)	Reserved				D/A Converter 4 Write											
0x410h (1040)		Use Threshold	Value >, ≤ Threshold	Analog Trigger Channel Comparison					Analog Trigger Threshold Value							
0x418h (1048)	Reserved								Digital I/O Port 2, Bit Programmable							
0x41Ah (1050)	Reserved								Digital I/O Port 3, Bit Programmable							

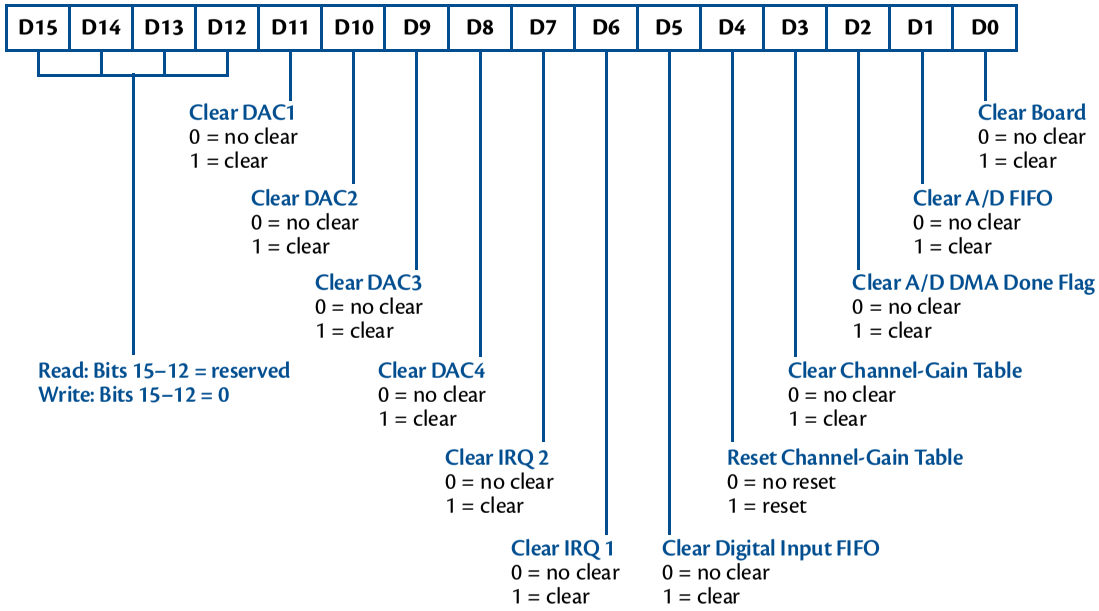
Table 7 DM6425HR Write Registers (cont'd)

Address hex (dec)	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x41Ch (1052)	Base + 0x41Eh Dependency (bits 1–0)															
	Reserved								Port 2/3 Clear Digital I/O Chip Information (value written is a don't care)							
	Reserved								Port 2 Direction (0 = Input)							
	Reserved								Port 2 Mask (1 = Mask)							
	Reserved								Port 2 Compare Value							
0x41Eh (1054)	Reserved										Port 2 Dig Sample Clk Select	Port 2 Digital IRQ Enable	Port 2 Digital IRQ Mode	Port 3 Direction	Port 2 Register Select	

BA + 0x00h (0): Clear/Program Clear Register (Read/Write)

16-bit Operation. A read clears selected circuits on the board, depending on the value programmed at this same address, as described in the following paragraph.

Clear Register:



The value programmed in this register determines which clear, enable, and reset operations are carried out when a read at **BA + 0x00h** is executed. Setting a bit high clears or enables the defined operation. This register's bits are described below:

Bit 0: When high (bit 0 = 1), clears, or resets, the board. Resets the board and initializes the A/D converter.

Bit 1: When high (bit 1 = 1), clears the sample buffer. Empties out all data in the FIFO, sets the FIFO empty flag low (**BA + 0x02h**, bit 0), sets the FIFO full flag high (**BA + 0x02h**, bit 1), and clears the HALT flag (**BA + 0x02h**, bit 2), enabling A/D conversions.

Bit 2: When high (bit 2 = 1), clears the A/D DMA done flag at **BA + 0x02h**, bit 4.

Bit 3: When high (bit 3 = 1), clears the channel-gain table. Erases the data entered into the channel-gain table.

Bit 4: When high (bit 4 = 1), resets the channel-gain table. Resets the channel-gain table's starting point to the beginning of the table.

Bit 5: When high (bit 5 = 1), clears the digital input FIFO. Empties out all data in the FIFO.

Bit 6: When high (bit 6 = 1), clears the interrupt 1 circuitry.
Bit 7: When high (bit 7 = 1), clears the interrupt 2 circuitry.
Bit 8: When high (bit 8 = 1), clears DAC4.
Bit 9: When high (bit 9 = 1), clears DAC3.
Bit 10: When high (bit 10 = 1), clears DAC2.
Bit 11: When high (bit 11 = 1), clears DAC1.

Example: if you want to clear the FIFO and DMA done flag, you would write a 6 to this address to set bits 1 and 2 high, followed by a read to carry out the clear operation.

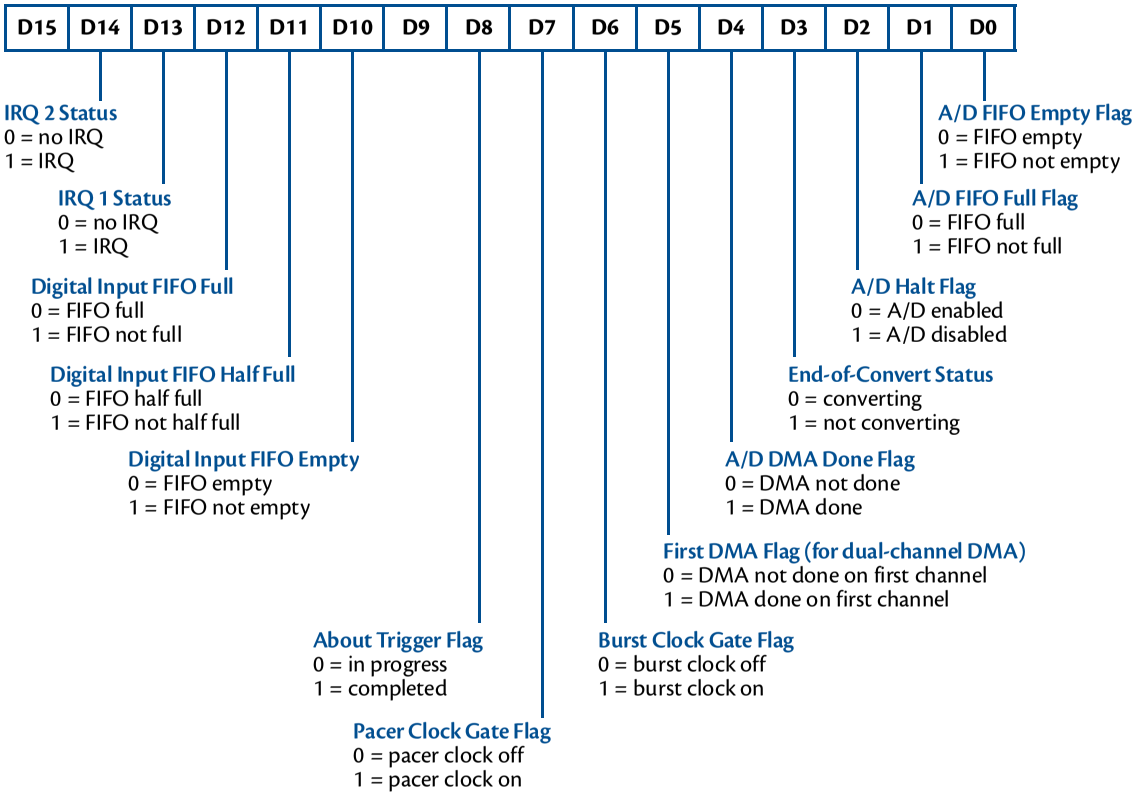
Clear FIFO and DMA Done Flag (value written = 6):

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

BA + 0x02h (2): Read Status/Program Control Register (Read/Write)

16-bit Operation.

Status Register (read): Bits 15 and 9 are reserved.



A read provides the status bits defined below. Starting with bit 0, these status bits show:

Bit 0: Goes high when there is something in the A/D sample buffer (FIFO).

Bit 1: Goes low when the sample buffer is full.

Bit 2: Goes high and halts A/D conversions when the sample buffer is full (this is useful whenever you are emptying the buffer at a slower rate than you are taking data). A clear FIFO written to **BA + 0x00h** (bit 1 set high) clears the sample buffer and this flag.

Bit 3: Shows the status of the A/D converter.

Bit 4: Goes high when an A/D DMA transfer is completed (active in DMA mode only).

Bit 5: Goes high when the DMA transfer for the first channel (set at **BA + 0x02h**, bits 13 and 12) is complete. This flag is used in dual channel DMA mode to signal when the switch is made to the second channel. Dual channel DMA transfer is explained in more detail in Chapter 6, *Data Transfers Using DMA*.

Bit 6: Shows the status of the burst gate (useful when using external triggering).

Bit 7: Shows the status of the pacer clock gate (useful when using external triggering).

Bit 8: Shows the about trigger status. Goes high after the about trigger has occurred.

Bit 10: Goes high when there is something in the Digital Input sample buffer (FIFO).

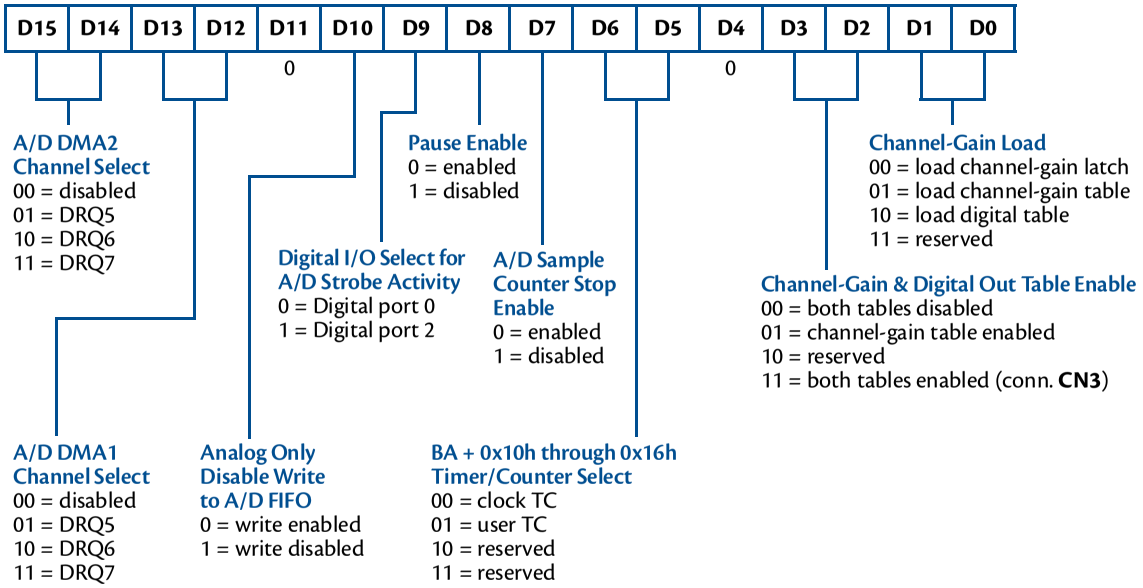
Bit 11: Goes low when the Digital Input sample buffer (FIFO) is half full.

Bit 12: Goes low when the Digital Input sample buffer is full.

Bit 13: Shows the status of IRQ 1.

Bit 14: Shows the status of IRQ 2.

Control Register (write):



A write to **BA + 0x02h** sets up the Control Register shown above. The settings you enter here determine whether the channel-gain data written to **BA + 0x04h** is loaded into the channel-gain latch or into the analog or digital portion of the channel-gain table, which timer/counter you address at **BA + 0x10h** through **BA + 0x16h**, enable/disable the A/D sample counter stop and pause bits, and select the A/D DMA channels. This register sets:

Bits 0 and 1: The setting of these bits determines where the data written at **BA + 0x04h** is stored. When bits 1 and 0 are 00, channel-gain data is loaded into the channel-gain latch. When bits 1 and 0 are 01, channel-gain data is loaded into the Channel-Gain Table of the channel-gain scan memory. When bits 1 and 0 are 10, digital data is loaded into the Digital Table of the channel-gain scan memory.

Bits 2 and 3: These bits are used to enable/disable the Channel-Gain and Digital Tables in the channel-gain scan memory. When bits 3 and 2 are 00, the channel-gain scan memory is disabled and the data written to the channel-gain latch will be used for A/D conversions. When bits 3 and 2 are 01, the Channel-Gain Table in the channel-gain scan memory is activated to be used for A/D conversions. When bits 3 and 2 are 11, both the Channel-Gain and Digital Tables in the channel-gain scan memory are activated to be used for A/D conversions via I/O connector **CN3**. Note that while you can enable, disable, and then re-enable the Channel-Gain table in the middle of taking a set of data, it is not recommended that you do this. One entry in the table is skipped each time the table is disabled and re-enabled unless reset table at **BA + 0x00h** is used to reset the table pointer.

Bits 5 and 6: Selects the 8254 timer/counter to be programmed at **BA + 0x10h** through **BA + 0x16h**. The Clock TC is the pacer clock/burst clock timer; the User TC is the A/D sample counter and the user timer/counters.

Bit 7: When enabled (set to 0), the A/D sample counter counts down once and stops the pacer clock. When disabled (set to 1), the A/D sample counter repeats the countdown until you enable the stop bit (set this bit to 0). Chapter 5 explains how to use this bit for sample counts greater than 65,536 (the size of the 16-bit A/D sample counter).

Bit 8: When enabled (set to 0), the pause bit in the Digital Table in the channel-gain scan memory (**BA + 0x04h**, bit 10) is activated. When disabled, the pause bit setting at **BA + 0x04h** is ignored.

Bit 9: Allows users to create start and stop triggers and burst triggers (see **BA + 0x06h**) using either digital I/O Port 0 or Port 2.

Bit 10: This bit should only be enabled when using the analog trigger. Refer to *Analog Trigger* on page 80.

Bit 12 to 15: These bits are used to set the DRQ channels for A/D DMA transfer. For simple DMA transfers using one channel, select the channel on bits 12 and 13. When using dual channels in the autoinitialized DMA mode (DMA controller autoinitialized so that you can flip-flop transfers, see Chapter 6 for large transfers), you must select different channels for DMA1 and DMA2.

BA + 0x04h (4): Read Converted Data/Load Channel-Gain and Digital Data (Read/Write)

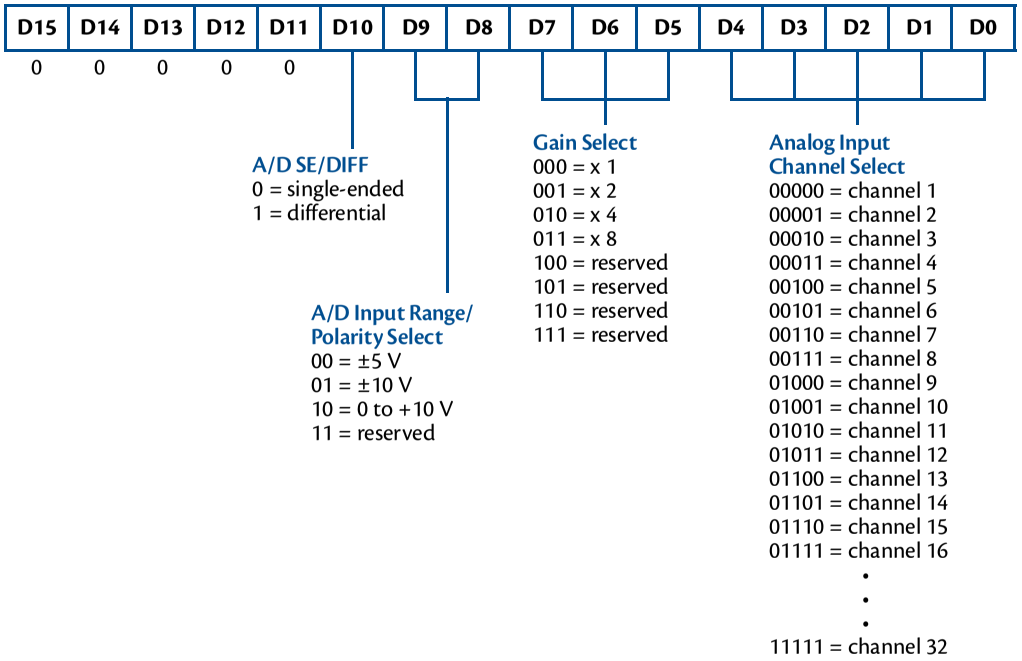
16-Bit Data Word Read from FIFO (16-bit operation): Bit 15 is the sign extension bit.

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	P0.7	P0.6	P0.5
	(MSB)											**(LSB)**	**A/D Data Markers**		

A read provides the 12-bit A/D converted data as shown above. Bit 15 is the sign bit. The next 12 bits provide the 12-bit converted data word. The bottom three bits are the A/D data markers. If you are using the data markers, the information tagged with the A/D data is stored in these three bits. All readings are in two's complement format.

Load Channel-Gain Latch (BA + 0x02h, bits 1 and 0 = 00)

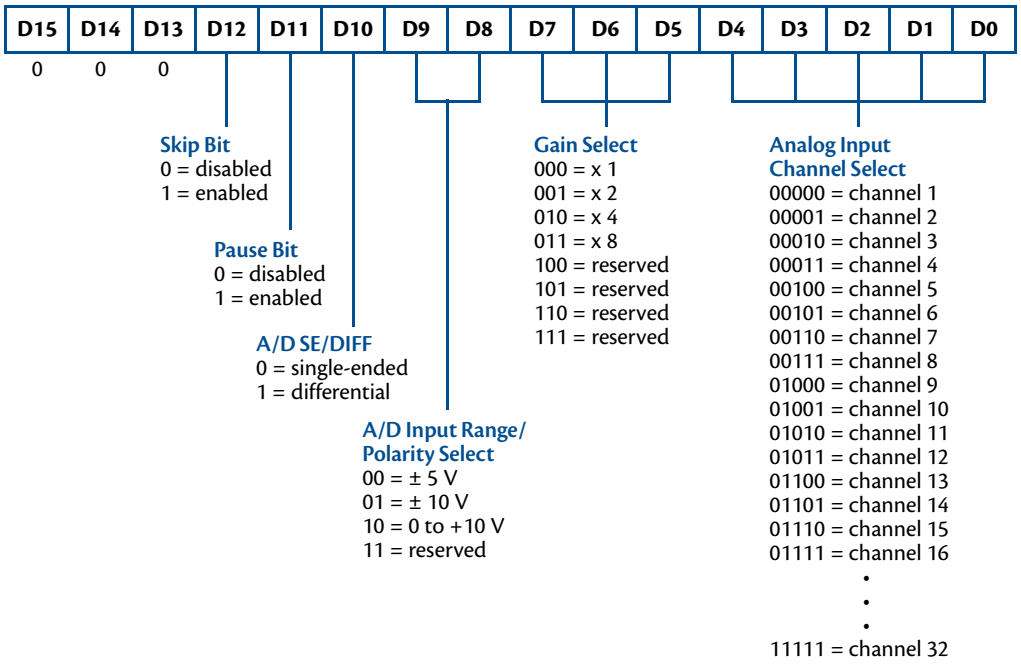
16-bit Operation (write):



To load channel and gain for conversions not using the channel-gain table: First, make sure that bits 1 and 0 at **BA + 0x02h** are set to 00. Then write the desired channel and gain information to **BA + 0x04h**. Bits 8 and 9 select the input range, and bit 10 selects whether the input is single-ended or differential.

Load Channel-Gain Table in Channel-Gain Scan Memory (BA + 0x02h, bits 1 and 0 = 01)

16-Bit Operation (write):



To load the A/D portion of the channel-gain table with channel and gain information: First, set bits 1 and 0 at **BA + 0x02h** to 01 to enable loading of channel and gain data into the A/D portion of the channel-gain table. Then, load the data in the format shown above. Each write fills the next position in the channel-gain table.

Using the pause bit: The pause bit at bit 10 of the channel-gain word is set to 1 if you want to stop at an entry in the table and wait for the next trigger to resume conversions. In burst mode, the pause bit is ignored.

Using the skip bit: The skip bit at bit 11 of the channel-gain word is set to 1 if you want to skip an entry in the table. This feature allows you to sample multiple channels at different rates on each channel. For example, if you want to sample channel 1 once each second and channel 4 once every 3 seconds, you can set the skip bit on channel 4 as shown in Figure 18. With the skip bit set on the four table entries as shown, these entries will be ignored, and no A/D conversion will be performed. This saves memory and eliminates the need to throw away unwanted data.

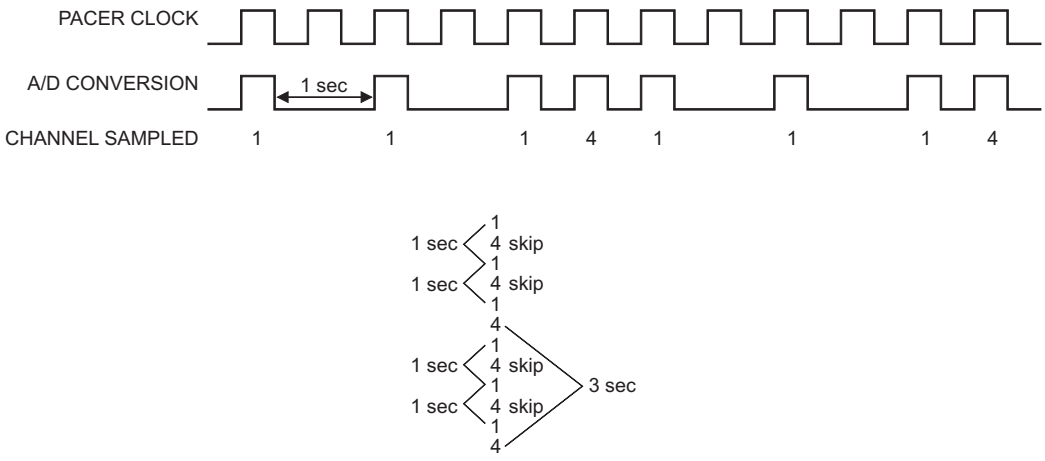
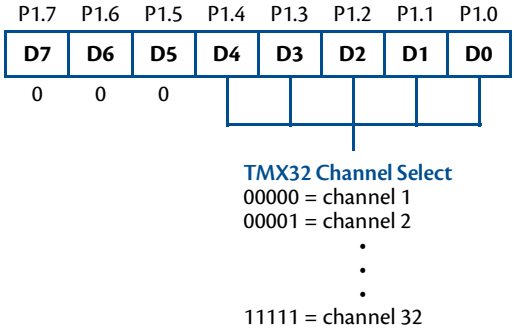


Figure 18 Using the Skip Bit

Load Digital Table in Channel-Gain Scan Memory (BA + 0x02h, bits 1 and 0 = 10)

8-Bit Operation (write):

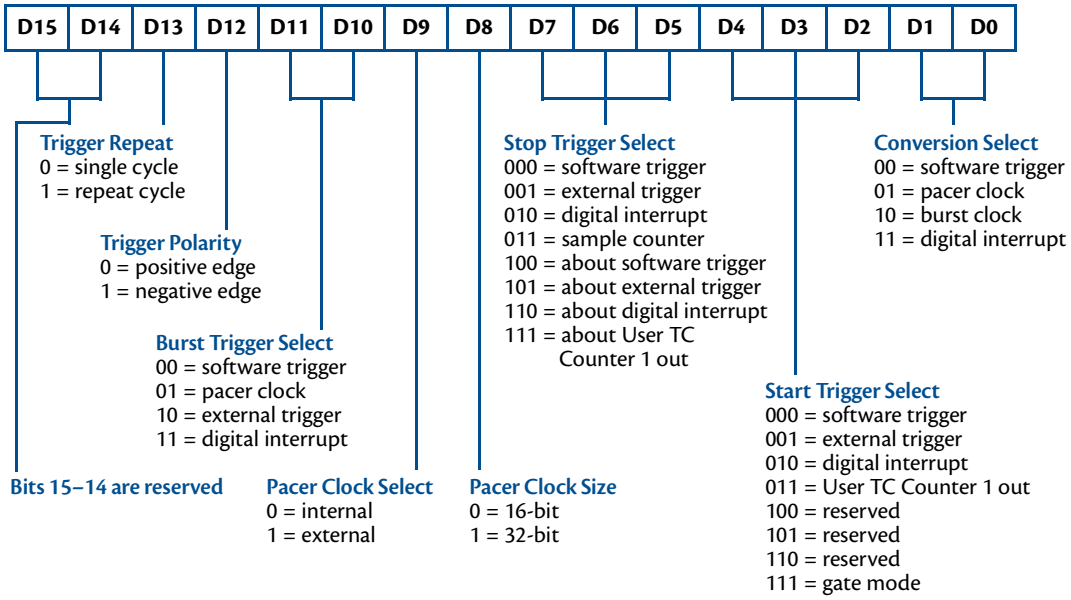


To load the digital portion of the channel-gain table with digital information: The digital portion of the channel-gain table provides 8 bits to control devices such as external expansion boards. For example, if you have connected one of your input channels on the DM6425HR to RTD's TMX32 input expansion board, you can use the bottom 5 bits in this byte to control the TMX32 board channel selection. To load digital information into this portion of the channel-gain table, set bits 1 and 0 at **BA + 0x02h** to 10 to enable loading of the digital portion of the channel-gain table. Then, load the data, setting 0s and 1s as needed by whatever you are controlling. This information will be output on the Port 1 lines when you run through the table. The format shown above is for controlling the TMX32's channel selection (32 single-ended or 16 differential). The first load operation will be in the first entry slot of the table (lining up with the first entry in the A/D table), and each load thereafter fills the next position in the channel-gain table. Note that when you are using the digital table, all 8 bits are used and controlled by the table, regardless of the number of bits you may actually need for your digital control application.

BA + 0x06h (6): Start Convert/Program Trigger Modes (Read/Write)

16-bit Operation. A read at this address issues a Start Convert command (software trigger).

Trigger Mode Register:



This register sets up the trigger mode and the method by which A/D conversions are performed (conversion select bits).

Trigger Mode Register, performing A/D conversions (bits 0 and 1):

00 = Conversions are controlled by reading **BA + 0x06h** (Start Convert).
01 = Conversions are controlled by the internal or an external pacer clock.
10 = Conversions are controlled by the burst clock.
11 = Conversions are controlled by a digital interrupt.

Trigger Mode Register, selecting the start trigger source (bits 2 through 4):

000 = The pacer clock is started by reading **BA + 0x06h** (Start Convert).
001 = The pacer clock is started by an external trigger (TRIGGER IN, **CN3**, pin 39).
010 = The pacer clock is started by a digital interrupt.
011 = The pacer clock is started when the output of User TC Counter 1 reaches 0.
100 = Reserved.
101 = Reserved.
110 = Reserved.
111 = The pacer clock runs as long as the TRIGGER IN line is held high or low, depending on the polarity bit setting at **BA + 0x06h**, bit 12.

Trigger Mode Register, selecting the stop trigger source (bits 5 through 7):

000 = The pacer clock is stopped by reading **BA + 0x06h** (Start Convert).
001 = The pacer clock is stopped by an external trigger (TRIGGER IN, **CN3**, pin 39).
010 = The pacer clock is stopped by a digital interrupt.
011 = The pacer clock is stopped by the sample counter (count reaches 0).

The following four stop trigger sources programmed at these bits provide about triggering, where data is acquired from the time the start trigger is received, and continues for a specified number of samples after the stop trigger is received. The A/D sample counter sets the number of samples taken after the stop trigger is received.

100 = The A/D sample counter takes a specified number of samples after a read at **BA + 0x06h** (Start Convert).
101 = The A/D sample counter takes a specified number of samples after an external trigger is received.
110 = The A/D sample counter takes a specified number of samples after a digital interrupt occurs.
111 = The A/D sample counter takes a specified number of samples after the output of User TC Counter 1 reaches 0.

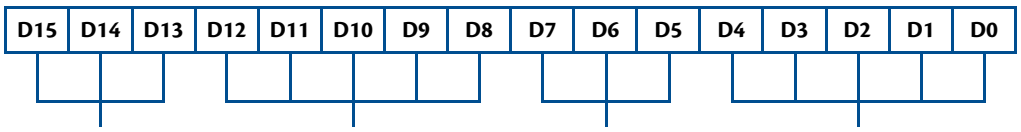
Trigger Mode Register, bits 8 through 13:

Bit 8: Selects a 16-bit or 32-bit onboard pacer clock (Clock TC Counter 0 or 1 output). When a trigger is used to start the pacer clock, there is some delay between the time the trigger occurs and the time the next pacer clock pulse starts an A/D conversion. For a 16-bit clock, this jitter is 125 ns, max. For a 32-bit clock, this jitter depends on the value programmed into the first divider and can be much greater than 125 ns. (See Chapter 5.)
Bit 9: Selects the internal pacer clock, which is the output of Clock TC Counter 0 or 1, or an external pacer clock routed onto the board through **CN3**, pin 41. The max. pacer clock rate supported by the board is 500 kHz.
Bits 10/11: Select the burst mode trigger. Bursts can be triggered through software (Start Convert command), by the pacer clock, by an external trigger, or by a digital interrupt.
Bit 12: Sets the external trigger to occur on the positive-going or negative-going edge of the pulse.
Bit 13: When set to single cycle, a trigger will initiate one conversion cycle and then stop, regardless of whether the trigger line is pulsed more than once; when set to repeat, a new cycle will start each time a trigger is received, and the current cycle has been completed. Triggers received while a cycle is in progress will be ignored.

BA + 0x08h (8): Program IRQ Source and Channel (Write)

16-bit Operation.

Interrupt Register:



IRQ2 Channel Select
000 = disabled
001 = IRQ3
010 = IRQ5
011 = IRQ9
100 = IRQ10
101 = IRQ11
110 = IRQ12
111 = IRQ15

IRQ2 Source Select
00000 = A/D sample counter
00001 = A/D start convert
00010 = A/D end-of-convert
00011 = A/D write FIFO
00100 = A/D FIFO half full
00101 = A/D DMA done
00110 = reset channel-gain table
00111 = pause channel-gain table
01000 = external pacer clock
01001 = external trigger
01010 = digital interrupt
01011 = User TC Counter 0 out
01100 = User TC Counter 0 out inverted
01101 = User TC Counter 1 out
01110 = digital input FIFO half full
01111 = digital input write FIFO
10000 = digital interrupt port2/3
10001 = analog threshold
10010 = external interrupt 1
10011 = external interupt 2

IRQ1 Channel Select
000 = disabled
001 = IRQ3
010 = IRQ5
011 = IRQ9
100 = IRQ10
101 = IRQ11
110 = IRQ12
111 = IRQ15

IRQ1 Source Select
00000 = A/D sample counter
00001 = A/D start convert
00010 = A/D end-of-convert
00011 = A/D write FIFO
00100 = A/D FIFO half full
00101 = A/D DMA done
00110 = reset channel-gain table
00111 = pause channel-gain table
01000 = external pacer clock
01001 = external trigger
01010 = digital interrupt
01011 = User TC Counter 0 out
01100 = User TC Counter 0 out inverted
01101 = User TC Counter 1 out
01110 = digital input FIFO half full
01111 = digital input write FIFO
10000 = digital interrupt port2/3
10001 = analog threshold
10010 = external interrupt 1
10011 = external interupt 2

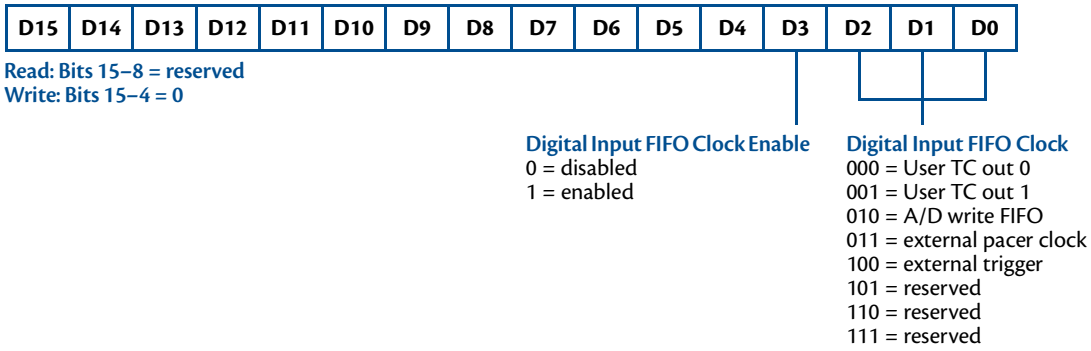
This register programs the software selectable interrupt source and channel. The IRQ circuitry is driven by an open collector device which is turned off when the IRQ channel is set to disable. The IRQ sources are:

A/D sample counter:	an interrupt is generated when the A/D sample counter count reaches 0.
A/D start convert:	an interrupt is generated when a conversion is started.
A/D end-of-convert:	an interrupt is generated when an end-of-convert is issued by the A/D converter.
A/D write FIFO:	an interrupt is generated when data is written into the A/D FIFO.
A/D FIFO half full:	an interrupt is generated when the A/D FIFO is half-full.
A/D DMA done:	an interrupt is generated when the A/D DMA done flag goes high.
Reset channel-gain table:	an interrupt is generated when the channel-gain table resets to the beginning.
Pause channel-gain table:	an interrupt is generated when a pause occurs in the channel-gain table.
External pacer clock:	an interrupt is generated when the external pacer clock line is pulsed.
External trigger:	an interrupt is generated when the external trigger line is pulsed.
Digital interrupt:	an interrupt is generated when an advanced digital interrupt occurs.
User TC Counter 0 out:	an interrupt is generated when user TC Counter 0's count reaches 0.
User TC Counter 0 out inverted:	an interrupt is generated when user TC Counter 0's count reaches 0 (useful for frequency counting).
User TC Counter 1 out:	an interrupt is generated when user TC Counter 1's count reaches 0.
Digital input FIFO half full:	an interrupt is generated when the Digital Input FIFO is half-full.
Digital input write FIFO:	an interrupt is generated when data is written into the Digital Input FIFO.

BA + 0x0Ah (10): Digital Input FIFO/Program Digital Input FIFO Configuration Register (Read/Write)

16-bit Operation. A read provides the contents of the 8-bit Digital Input FIFO connected to Port 0.

Digital Input FIFO Configuration Register:



This register is used to configure the Digital Input FIFO clocks on the DM6425HR as follows:

Bits 0, 1, 2: These bits set the digital input FIFO clock source. Options include the User TC outputs, the write pulse to the A/D FIFO, the external pacer clock (**CN3**, pin 41), and the external trigger (**CN3**, pin 39). The data at the input to the FIFO is latched on the rising edge of the clock.

Bit 3: This bit is used to enable and disable the clock into the FIFO.

BA + 0x0Eh (14): A/D Sample Counter Trigger (Read)

16-bit Operation. A read provides a software trigger so that the A/D sample counter can be loaded with the correct value. This software correction is used as an easy means to compensate for the operating structure of the 8254. Two pulses of the counter are required to actually load the desired count and prepare the counter to count down correctly (this can be looked at as the initialization procedure for the A/D sample counter). A pulse is sent to the A/D sample counter (User TC Counter 2) each time you read this address. Without this correction, the initial count sequence will be off by two pulses. Once the counter is properly loaded and starts, any subsequent countdowns of this count will be accurate. Note that the A/D sample counter must be programmed for Mode 2 operation.

BA + 0x10h (16): TC Counter 0 (Read/Write)

8-bit Operation. A write loads the first counter in one of the two timer/counters on the board with a new 16-bit value in two 8-bit steps, LSB followed by MSB. The counter must be loaded in two 8-bit steps! Counting begins as soon as the count is loaded. The timer/counter being loaded is selected by writing to **BA + 0x02h**, bits 5 and 6. A read shows the count in the counter. Clock Chip: **BA + 0x02h**, bits 5 and 6 = 00. Counter Chip: **BA + 0x02h**, bits 5 and 6 = 01.

BA + 0x12h (18): TC Counter 1 (Read/Write)

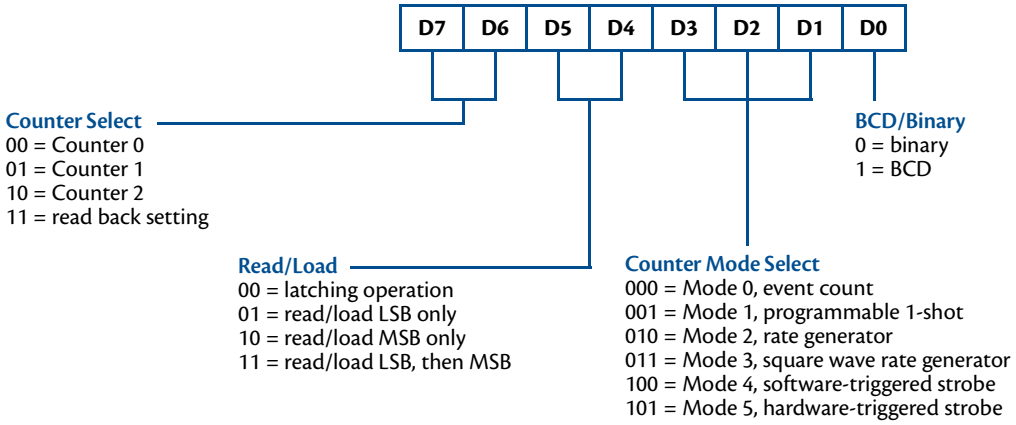
8-bit Operation. A write loads the second counter in one of the two timer/counters on the board with a new 16-bit value in two 8-bit steps, LSB followed by MSB. The counter must be loaded in two 8-bit steps! Counting begins as soon as the count is loaded. The timer/counter being loaded is selected by writing to **BA + 0x02h**, bits 5 and 6. A read shows the count in the counter. Clock Chip: **BA + 0x02h**, bits 5 and 6 = 00. Counter Chip: **BA + 0x02h**, bits 5 and 6 = 01.

BA + 0x14h (20): TC Counter 2 (Read/Write)

8-bit Operation. A write loads the third counter in one of the two timer/counters on the board with a new 16-bit value in two 8-bit steps, LSB followed by MSB. The counter must be loaded in two 8-bit steps! Counting begins as soon as the count is loaded. The timer/counter being loaded is selected by writing to **BA + 0x02h**, bits 5 and 6. A read shows the count in the counter. Clock Chip: **BA + 0x02h**, bits 5 and 6 = 00. Counter Chip: **BA + 0x02h**, bits 5 and 6 = 01.

BA + 0x16h (22): Timer/Counter Control Word (Write Only)

8-bit Operation. Accesses the selected timer/counter's control register to directly control the three 16-bit counters: 0, 1, and 2. Clock Chip: **BA + 0x02h**, bits 5 and 6 = 00. Counter Chip: **BA + 0x02h**, bits 5 and 6 = 01.



BA + 0x18h (24): Digital I/O Port 0, Bit Programmable Port (Read/Write)

8-bit Operation.

Port 0:

D7	D6	D5	D4	D3	D2	D1	D0
P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0

This port transfers the 8-bit Port 0 bit programmable digital input/output data between the module and external devices. The bits are individually programmed as input or output by writing to the Direction Register at **BA + 0x1Ch**. For all bits set as inputs, a read reads the input values and a write is ignored. For all bits set as outputs, a read reads the last value sent out on the line and a write writes the current loaded value out to the line.

Note that when any reset of the digital circuitry is performed (clear chip or computer reset), all digital lines are reset to inputs and their corresponding output registers are cleared.

BA + 0x1Ah (26): Digital I/O Port 1, Byte Programmable Port (Read/Write)

8-bit Operation.

Port 1:

D7	D6	D5	D4	D3	D2	D1	D0
P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0

This port transfers the 8-bit Port 1 digital input or digital output byte between the module and an external device. When Port 1 is set as inputs, a read reads the input values and a write is ignored. When Port 1 is set as outputs, a read reads the last value sent out of the port and a write writes the current loaded value out of the port.

Note that when any reset of the digital circuitry is performed (clear chip or computer reset), all digital lines are reset to inputs and their corresponding output registers are cleared.

BA + 0x1Ch (28): Read/Program Port 0 Direction/Mask/Compare Registers (Read/Write)

8-bit Operation. A read clears the IRQ status flag or provides the contents of one of digital I/O Port 0's three control registers; and a write clears the digital chip or programs one of the three control registers, depending on the setting of bits 0 and 1 at **BA + 0x1Eh**. When bits 1 and 0 at **BA + 0x1Eh** are 00, the read/write operations clear the digital IRQ status flag (read) and the digital chip (write). When these bits are set to any other value, one of the three Port 0 registers is addressed.

Direction Register (BA + 0x1Eh, bits 1 and 0 = 01):

For all bits:
0 = input
1 = output

D7	D6	D5	D4	D3	D2	D1	D0
P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0

This register programs the direction, input or output, of each bit at Port 0.

Mask Register (BA + 0x1Eh, bits 1 and 0 = 10):

For all bits:
0 = bit enabled
1 = bit masked

D7	D6	D5	D4	D3	D2	D1	D0
P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0

In the Advanced Digital Interrupt modes, this register is used to mask out specific bits when monitoring the bit pattern present at Port 0 for interrupt generation. In normal operation where the Advanced Digital Interrupt feature is not being used, any bit which is masked by writing a 1 to that bit will not change state, regardless of the digital data written to Port 0. For example, if you set the state of bit 0 low and then mask this bit, the state will remain low, regardless of what you output at Port 0 (an output of 1 will not change the bit's state until the bit is unmasked).

Compare Register (BA + 0x1Eh, bits 1 and 0 = 11):

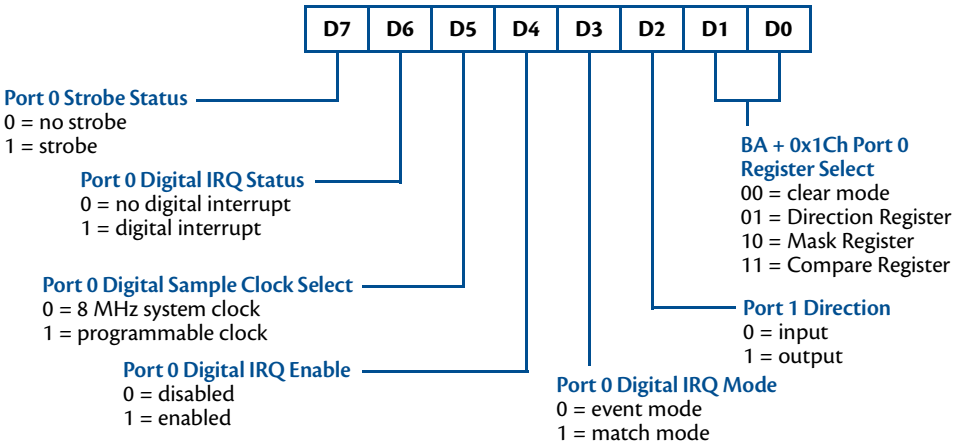
D7	D6	D5	D4	D3	D2	D1	D0
P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0

This register is used for the Advanced Digital Interrupt modes. In the match mode where an interrupt is generated when the Port 0 bits match a loaded value, this register is used to load the bit pattern to be matched at Port 0. Bits can be selectively masked so that they are ignored when making a match. NOTE: Make sure that bit 3 at **BA + 0x1Eh** is set to 1, selecting match mode, BEFORE writing the Compare Register value at this address. In the event mode where an interrupt is generated when any Port 0 bit changes its current state, the value which caused the interrupt is latched at this register and can be read from it. Bits can be selectively masked using the Mask Register so a change of state is ignored on these lines in the event mode.

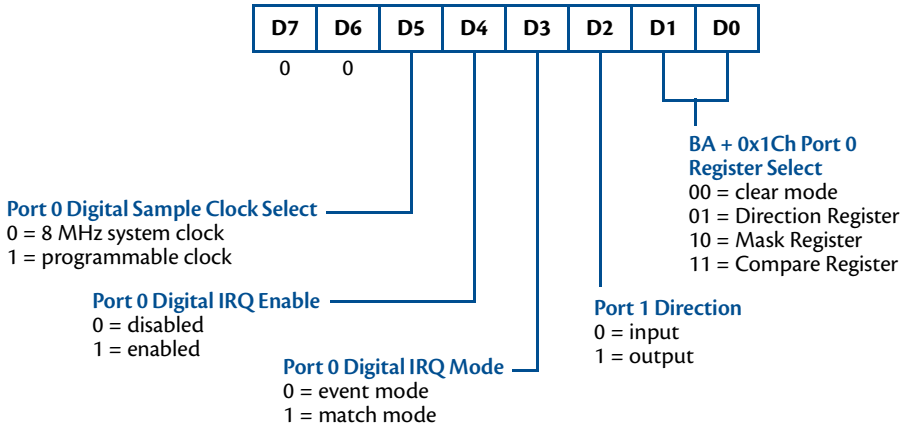
BA + 0x1Eh (30): Digital IRQ Status/Program Digital Mode (Ports 0 and 1, Read/Write)

8-bit Operation. Digital IRQ/Strobe Status (read):

A read shows you whether a digital interrupt has occurred (bit 6), whether a strobe has occurred (bit 7, when using the strobe input as described in Chapter 7), and lets you review the states of bits 0 through 5 in this register. If bit 6 is high, then a digital interrupt has taken place. If bit 7 is high, a strobe has been issued.



Digital Mode Register (write):



Bits 0 and 1: Select the clear mode initiated by a read/write operation at **BA + 0x1Ch** or the Port 0 control register you talk to at **BA + 0x1Ch** (Direction, Mask, or Compare Register).

Bit 2: Sets the direction of the Port 1 digital lines.

Bit 3: Selects the digital interrupt mode: event (any Port 0 bit changes state) or match (Port 0 lines match the value programmed into the Compare Register at **BA + 0x1Ch**).

Bit 4: Disables/enables digital interrupts.

Bit 5: Sets the clock rate at which the digital lines are sampled when in a digital interrupt mode. Available clock sources are the 8 MHz system clock and the output of User TC Counter 1 (16-bit programmable clock). When a digital input line changes state, it must stay at the new state for two edges of the clock pulse (62.5 ns when using the 8 MHz clock) before it is recognized and before an interrupt can be generated. This feature eliminates noise glitches that can cause a false state change on an input line and generate an unwanted interrupt. This feature is detailed in Chapter 7.

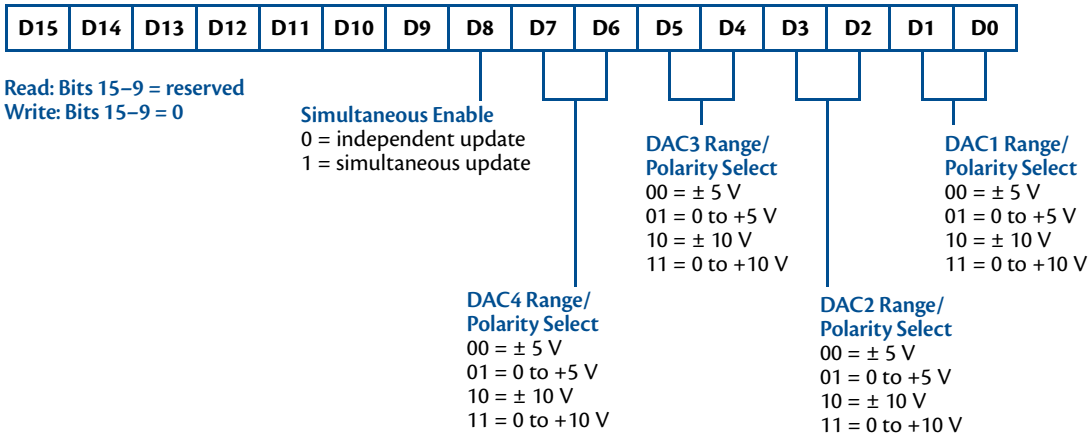
Bit 6: Read only (digital IRQ status).

Bit 7: Read only (strobe status).

BA + 0x400h (1024): D/A Analog Setup (Read/Write)

16-bit Operation.

This register is used to read/write the D/A analog output range. Options are: ± 5 V, 0 to +5 V, ±10 V, and 0 to +10 V.



BA + 0x402h (1026): Simutaneous Update Register (Write)

16-bit Operation. The value written to this register is not important. A write to this register will cause all DACs to be simultaneously updated at their respective analog output ranges when bit 8 of **BA + 0x400h** is set to 1.

BA + 0x404h (1028): DAC1 Update Register (Write)

16-bit Operation. A write to this register will load the value into DAC1 and automatically update the DAC when bit 8 of **BA + 0x400h** is set to 0.

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Bit 16	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1

BA + 0x406h (1030): DAC2 Update Register (Write)

16-bit Operation. A write to this register will load the value into DAC2 and automatically update the DAC when bit 8 of **BA + 0x400h** is set to 0.

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Bit 16	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1

BA + 0x408h (1032): DAC3 Update Register (Write)

16-bit Operation. A write to this register will load the value into DAC3 and automatically update the DAC when bit 8 of **BA + 0x400h** is set to 0.

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Bit 16	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1

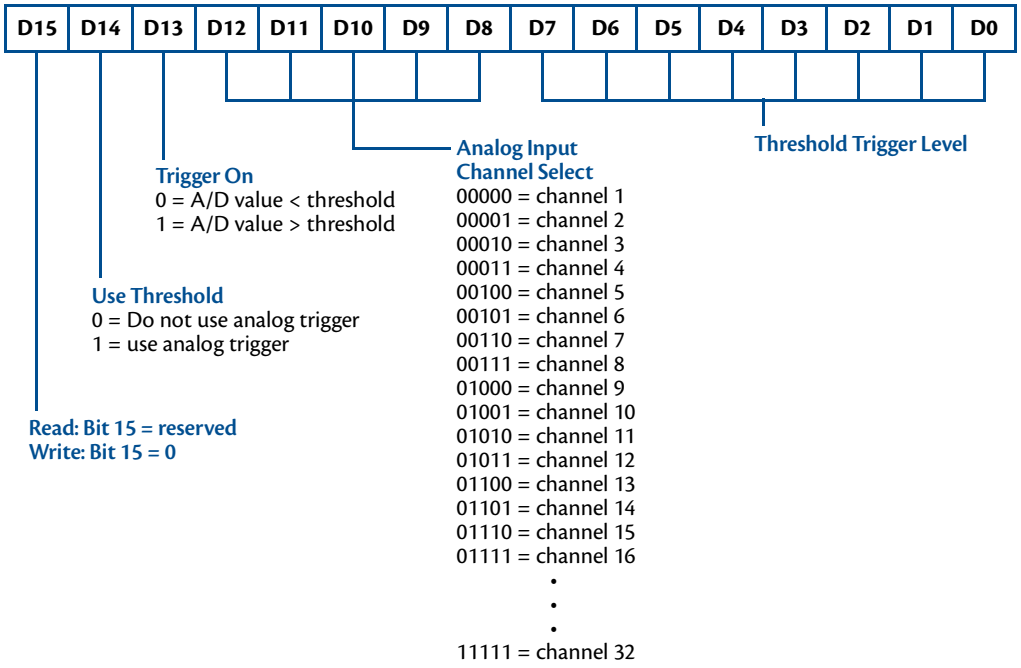
BA + 0x40Ah (1034): DAC4 Update Register (Write)

16-bit Operation. A write to this register will load the value into DAC4 and automatically update the DAC when bit 8 of **BA + 0x400h** is set to 0.

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Bit 16	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1

BA + 0x410h (1040): Analog Threshold Detector Setup (Read/Write)

16-bit Operation. The Threshold Trigger Level value (bits 7 to 0) can be anything between 0x00h and 0xFFh. Refer to *Analog Trigger* on page 80 for further information and corresponding voltage values for each hexadecimal number.



BA + 0x418h (1048): Digital I/O Port 2, Bit Programmable Port (Read/Write)

8-bit Operation.

Port 2:

D7	D6	D5	D4	D3	D2	D1	D0
P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0

This port transfers the 8-bit Port 2 bit programmable digital input/output data between the module and external devices. The bits are individually programmed as input or output by writing to the Direction Register at **BA + 0x41Ch**. For all bits set as inputs, a read reads the input values and a write is ignored. For all bits set as outputs, a read reads the last value sent out on the line and a write writes the current loaded value out to the line.

Note that when any reset of the digital circuitry is performed (clear chip or computer reset), all digital lines are reset to inputs and their corresponding output registers are cleared.

BA + 0x41Ah (1050): Digital I/O Port 3, Byte Programmable Port (Read/Write)

8-bit Operation.

Port 3:

D7	D6	D5	D4	D3	D2	D1	D0
P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0

This port transfers the 8-bit Port 3 digital input or digital output byte between the module and an external device. When Port 3 is set as inputs, a read reads the input values and a write is ignored. When Port 3 is set as outputs, a read reads the last value sent out of the port and a write writes the current loaded value out of the port.

Note that when any reset of the digital circuitry is performed (clear chip or computer reset), all digital lines are reset to inputs and their corresponding output registers are cleared.

BA + 0x41Ch (1052): Read/Program Port 2 Direction/Mask/Compare Registers (Read/Write)

8-bit Operation. A read clears the IRQ status flag or provides the contents of one of digital I/O Port 2's three control registers; and a write clears the digital chip or programs one of the three control registers, depending on the setting of bits 0 and 1 at **BA + 0x41Eh**. When bits 1 and 0 at **BA + 0x41Eh** are 00, the read/write operations clear the digital IRQ status flag (read) and the digital chip (write). When these bits are set to any other value, one of the three Port 2 registers is addressed.

Direction Register (BA + 0x41Eh, bits 1 and 0 = 01):

For all bits:
0 = input
1 = output

D7	D6	D5	D4	D3	D2	D1	D0
P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0

This register programs the direction, input or output, of each bit at Port 2.

Mask Register (BA + 0x41Eh, bits 1 and 0 = 10):

For all bits:
0 = bit enabled
1 = bit masked

D7	D6	D5	D4	D3	D2	D1	D0
P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0

In the Advanced Digital Interrupt modes, this register is used to mask out specific bits when monitoring the bit pattern present at Port 2 for interrupt generation. In normal operation where the Advanced Digital Interrupt feature is not being used, any bit which is masked by writing a 1 to that bit will not change state, regardless of the digital data written to Port 2. For example, if you set the state of bit 0 low and then mask this bit, the state will remain low, regardless of what you output at Port 2 (an output of 1 will not change the bit's state until the bit is unmasked).

Compare Register (BA + 0x41Eh, bits 1 and 0 = 11):

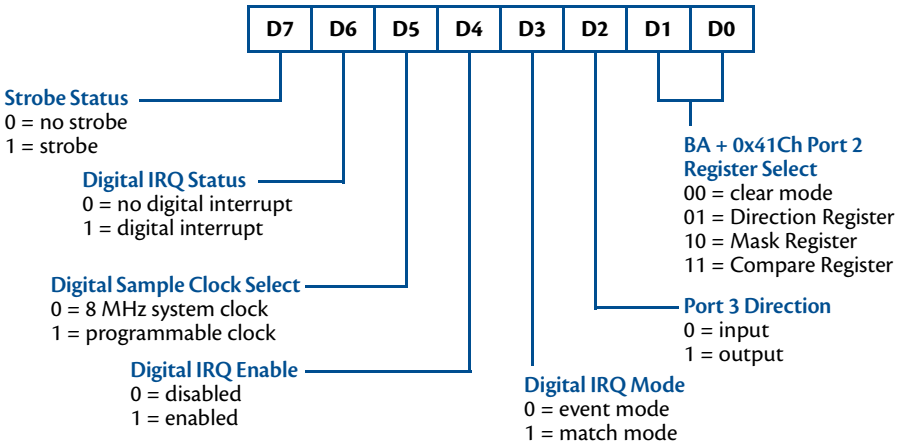
D7	D6	D5	D4	D3	D2	D1	D0
P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0

This register is used for the Advanced Digital Interrupt modes. In the match mode where an interrupt is generated when the Port 2 bits match a loaded value, this register is used to load the bit pattern to be matched at Port 2. Bits can be selectively masked so that they are ignored when making a match. NOTE: Make sure that bit 3 at **BA + 0x41Eh** is set to 1, selecting match mode, BEFORE writing the Compare Register value at this address. In the event mode where an interrupt is generated when any Port 2 bit changes its current state, the value which caused the interrupt is latched at this register and can be read from it. Bits can be selectively masked using the Mask Register so a change of state is ignored on these lines in the event mode.

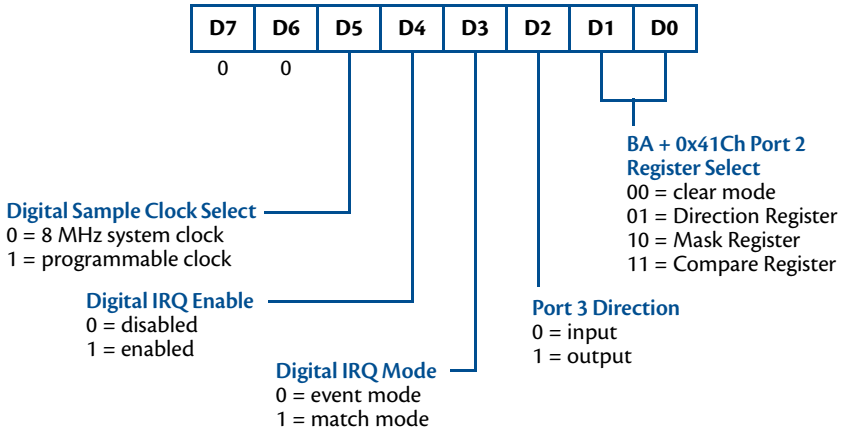
BA + 0x41Eh (1054): Digital IRQ Status/Program Digital Mode (Ports 2 and 3, Read/Write)

8-bit Operation. Digital IRQ/Strobe Status (read):

A read shows you whether a digital interrupt has occurred (bit 6), whether a strobe has occurred (bit 7, when using the strobe input as described in Chapter 7), and lets you review the states of bits 0 through 5 in this register. If bit 6 is high, then a digital interrupt has taken place. If bit 7 is high, a strobe has been issued.



Digital Mode Register (write):



Bits 0 and 1: Select the clear mode initiated by a read/write operation at **BA + 0x41Ch** or the Port 2 control register you talk to at **BA + 0x41Ch** (Direction, Mask, or Compare Register).

Bit 2: Sets the direction of the Port 3 digital lines.

Bit 3: Selects the digital interrupt mode: event (any Port 2 bit changes state) or match (Port 2 lines match the value programmed into the Compare Register at **BA + 0x41Ch**).

Bit 4: Disables/enables digital interrupts.

Bit 5: Sets the clock rate at which the digital lines are sampled when in a digital interrupt mode. Available clock sources are the 8 MHz system clock and the output of User TC Counter 1 (16-bit programmable clock). When a digital input line changes state, it must stay at the new state for two edges of the clock pulse (62.5 ns when using the 8 MHz clock) before it is recognized and before an interrupt can be generated. This feature eliminates noise glitches that can cause a false state change on an input line and generate an unwanted interrupt. This feature is detailed in Chapter 7.

Bit 6: Read only (digital IRQ status).

Bit 7: Read only (strobe status).

BA + 0x800h/801h (2048/2049): Board ID Register (Read)

16-bit Operation. A byte or word accessible read shows the device ID, vendor ID, or FPGA version.

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Bit 16	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1

Device ID: 6425

Vendor ID: 1435

FPGA Revision: xxxx

BA + 0x802h (2050): Board ID Reset (Read)

16-bit Operation. A read resets the board ID register or device ID.

Programming the DM6425HR

This section gives you some general information about programming and the DM6425HR.

The DM6425HR is programmed by writing to and reading from the correct I/O port locations on the board. These I/O ports were defined in the previous section. Because the DM6425HR is AT bus compatible, most operations are done in a 16-bit word format. The 8254 timer/counters must be programmed in 8-bit operations. High-level languages such as Pascal, C, and C++ make it very easy to read/write these ports. The table below shows you how to read from and write to I/O ports in Turbo C and Turbo Pascal.

Language	Read 8 Bits	Write 8 Bits	Read 16 Bits	Write 16 Bits
Turbo C	Data=inportb(Address)	outportb(Address, Data)	Data=inport(Address)	outport(Address, Data)
Turbo Pascal	Data:=Port[Address]	Port[Address]:=Data	Data:=PortW[Address]	PortW[Address]:=Data

In addition to being able to read/write the I/O ports on the DM6425HR, you must be able to perform a variety of operations that you might not normally use in your programming. The table below shows you some of the operators discussed in this section, with an example of how each is used with Pascal and C.

Language	Modulus	Integer Division	AND	OR
C	% a = b % c	/ a = b / c	& a = b & c	| a = b | c
Pascal	MOD a := b MOD c	DIV a := b DIV c	AND a :=b AND c	OR a :=b OR c

Many compilers have functions that can read/write either 8 or 16 bits from/to an I/O port. For example, Turbo Pascal uses **Port** for 8-bit port operations and **PortW** for 16 bits, Turbo C uses **inportb** for an 8-bit read of a port and **inport** for a 16-bit read.

CAUTION *Be sure to use the correct function for 8- and 16-bit operations with the DM6425HR!*

Clearing and Setting Bits in a Port

When you clear or set one or more bits in a port, you must be careful that you do not change the status of the other bits. You can preserve the status of all bits you do not wish to change by proper use of the AND and OR binary operators. Using AND and OR, single or multiple bits can be easily cleared in one operation.

To **clear** a single bit in a port, AND the current value of the port with the value b, where $b = 255 - 2^{bit}$.

Example: Clear bit 5 in a port. Read in the current value of the port, AND it with 223($223 = 255 - 2^5$), and then write the resulting value to the port. In BASIC, this is programmed as:

```
V = INP(PortAddress)
V = V AND 223
OUT PortAddress, V
```

To **set** a single bit in a port, OR the current value of the port with the value b, where $b = 2^{bit}$.

Example: Set bit 3 in a port. Read in the current value of the port, OR it with 8 (8 = 23), and then write the resulting value to the port. In Pascal, this is programmed as:

```
V := Port[PortAddress];
V := V OR 8;
Port[PortAddress] := V;
```

Setting or clearing more than one bit at a time is accomplished just as easily. To **clear** multiple bits in a port, AND the current value of the port with the value b, where b = 255 – (the sum of the values of the bits to be cleared). Note that the bits do not have to be consecutive.

Example: Clear bits 2, 4, and 6 in a port. Read in the current value of the port, AND it with 171 ($171 = 255 - 2^2 - 2^4 - 2^6$), and then write the resulting value to the port. In C, this is programmed as:

```
v = inportb(port_address);
v = v & 171;
outportb(port_address, v);
```

To **set** multiple bits in a port, OR the current value of the port with the value b, where b = the sum of the individual bits to be set. Note that the bits to be set do not have to be consecutive.

Example: Set bits 3, 5, and 7 in a port. Read in the current value of the port, OR it with 168 ($168 = 2^3 + 2^5 + 2^7$), and then write the resulting value back to the port. In assembly language, this is programmed as:

```
mov dx, PortAddress
in al, dx
or al, 168
out dx, al
```

Often, assigning a range of bits is a mixture of setting and clearing operations. You can set or clear each bit individually or use a faster method of first clearing all the bits in the range then setting only those bits that must be set using the method shown above for setting multiple bits in a port. The following example shows how this two-step operation is done.

Example: Assign bits 3, 4, and 5 in a port to 101 (bits 3 and 5 set, bit 4 cleared). First, read in the port and clear bits 3, 4, and 5 by ANDing them with 199. Then set bits 3 and 5 by ORing them with 40, and finally write the resulting value back to the port. In C, this is programmed as:

```
v = inportb(port_address);
v = v & 199;
v = v | 40;
outportb(port_address, v);
```

A final note: Do not be intimidated by the binary operators AND and OR and try to use operators for which you have a better intuition. For instance, if you are tempted to use addition and subtraction to set and clear bits in place of the methods shown above, DON'T! Addition and subtraction may seem logical, but they **will not work** if you try to clear a bit that is already clear or set a bit that is already set. For example, you might think that to set bit 5 of a port, you simply need to read in the port, add 32 (2^5) to that value, and then write the resulting value back to the port. This works fine if bit 5 is not already set. But, what happens when bit 5 is already set? Bits 0 to 4 will be unaffected and we can't say for sure what happens to bits 6 and 7, but we can say for sure that bit 5 ends up cleared instead of being set. A similar problem happens when you use subtraction to clear a bit in place of the method shown above.

Chapter 5 A/D Conversions

This chapter shows you how to program your DM6425HR dataModule to perform A/D conversions and read the results. This chapter also includes instructions on setting up the channel-gain scan memory, the onboard clocks and sample counter, and various conversion and triggering modes.

The following sections walk you through the programming steps for performing A/D conversions, including detailed information about conversion modes and triggering. You can also follow the steps in the example programs included with the module. In this discussion, BA refers to the base address. All values are in decimal unless otherwise specified.

Before Starting Conversions

Initializing the Module

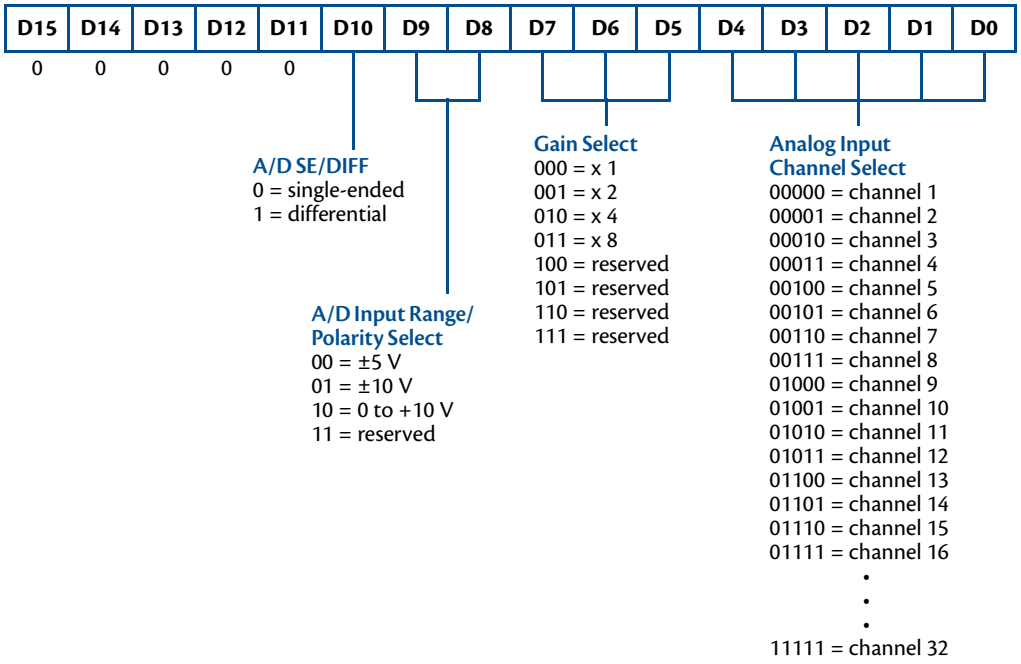
Regardless of the conversion mode you wish to set up, you should always start your program with a module initialization sequence. This sequence should be to:

1. Clear Board command.
2. Clear A/D DMA Done command.
3. Clear Channel Gain Table command.
4. Clear Digital Input FIFO command.
5. Clear Digital I/O chip.
6. Clear A/D FIFO command.

This initialization procedure clears all board registers, resets the DMA done flag to a "0", empties the Channel Gain Table, empties the Digital Input FIFO, resets the digital I/O chip and empties the A/D FIFO. All of these commands are carried out by writing and reading from the registers at **BA + 0x00h**, **BA + 0x1Ch**, **BA + 0x1Eh**, **BA + 0x41Ch**, and **BA + 0x41Eh**. Since you cannot read back the contents of the Control Register (**BA + 0x02h**), Trigger Register (**BA + 0x06h**), IRQ Register (**BA + 0x08h**), or the Digital Input FIFO Initialization Register (**BA + 0x0Ah**), RTD recommends that you store these values in a software variable for each register. These variables should be reset to "0" any time you issue the reset board command.

Programming Channel, Gain, Input Range, and Type

The conversion channel, gain, input range, and input type are programmed at **BA + 0x04h**. To program a conversion channel for direct A/D conversion (not using the channel-gain table), you must first point **BA + 0x04h** to write to the channel/gain latch. This is done by setting bits 0 and 1 to "00" in the Control Register at **BA + 0x02h**. To program the channel, gain, input range, and input type, assign the appropriate values to bits 0 through 10 and write this value to **BA + 0x04h**. The diagram below shows this register.



The program sequence for programming the channel and gain not using the channel-gain scan memory is:

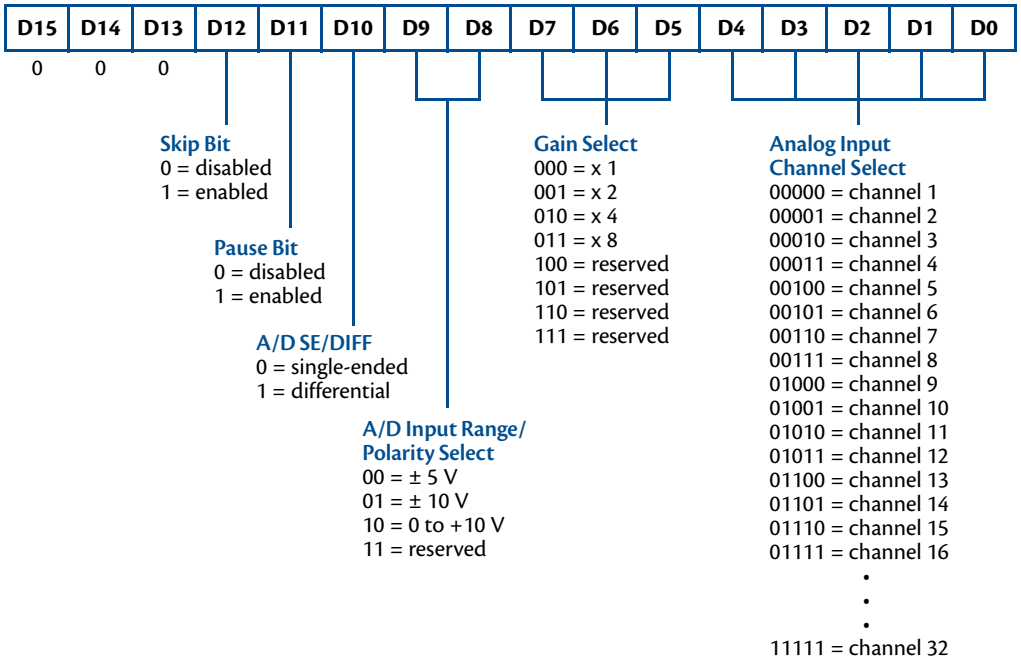
1. Set bits 1 and 0 at **BA + 0x02h** to 00 (this points **BA + 0x04h** to the channel/gain latch).
2. Write the channel and gain data to be loaded to **BA + 0x04h**.

Programming the Channel-Gain Table

The channel-gain scan memory can be programmed with 1024 24-bit entries in tabular format. Sixteen bits contain the A/D channel-gain data, and 8 bits contain digital control data to support complex channel-gain sequences. To load a new channel-gain table, first clear the channel gain table by writing and reading at **BA + 0x00h**. To add entries to an existing table, simply write to the A/D Table (and Digital Table if used) as described in the following paragraphs. Note that writing beyond the end of the table is ignored.

16-Bit A/D Table

The A/D portion of the channel-gain table with the channel, gain, input range, input type, pause and skip bit information is programmed into the channel-gain scan memory using the A/D Table Register at **BA + 0x04h**. This register is defined below. To load channel and gain data into the A/D table, first set bits 1 and 0 at **BA + 0x02h** to 01. This points **BA + 0x04h** to write to the A/D table. Now you can write the 16-bit channel/gain word to **BA + 0x04h**. If you have cleared the existing table, the first word written will be placed in the first entry of the table, the second word will be placed in the second entry, and so on. If you are adding to an existing table, the new data written will be added at the end.



Channel Select, Gain Select, Input Range, and Input Type

The channel number, gain value, input range, and input type are entered in the table using bits 0 through 10. Each of these parameters can be set independently for every entry in the table. This allows you to set up a complex array of sampling sequences mixing channels, gains, input ranges, and input types. Care must be taken in selecting the proper input type. The board is capable of 32 single-ended inputs or 16 differential inputs. You can select combinations of single-ended and differential but each differential channel actually uses 2 single-ended channels. If you select channel 1 to be a differential channel, you must connect your signal to AIN1+ (**CN3**, pin 1) and AIN1- (**CN3**, pin 2). Channel 8 now is not available as a single-ended channel.

Pause Bit

Bit 11 is used as a pause bit. If this bit is set to a "1" and the Pause function is enabled at **BA + 0x02h**, bit 8, the A/D conversions will stop at this entry in the table and resume on the next Start Trigger. This is useful if you have two different sequences loaded in the table. You can enable and disable this bit's function at **BA + 0x02h**, bit 8.

Note *This bit is ignored in the Burst sampling modes.*

Skip Bit

If bit 12 of the data loaded is set to 1, then the skip bit is enabled and this entry in the channel-gain table will be skipped, meaning an A/D conversion will be performed but the data is not written into the FIFO. This feature provides an easy way to sample multiple channels at different rates without saving unwanted data. A simple example illustrates this bit's function.

In this example, we want to sample channel 1 once each second and channel 4 once every three seconds. First, we must program 6 entries into the channel-gain table. The channel 4 entries with the skip bit set will be skipped when A/D conversions are performed. The table will continue to cycle until a stop trigger is received.

Next, we will set the pacer clock to run at 2 Hz (0.5 seconds). This allows us to sample each channel once per second, the maximum sampling rate required by one of the channels (pacer clock rate = number of different channels sampled x fastest sample rate). The first clock pulse starts an A/D conversion according to the parameters set in the first entry of the channel-gain table, and each successive clock pulse incrementally steps through the table entries. As shown in Figure 19, the first clock pulse starts a sample on channel 1. The next pulse looks at the second entry in the channel-gain table and sees that the skip bit is set to 1. No A/D data is stored. The third pulse starts a sample on channel 1 again, the fourth pulse skips the next entry, and the fifth pulse takes our third reading on channel 1. On the sixth pulse, the skip bit is disabled and channel 4 is sampled. Then the sequence starts over again. Samples are not stored when they are not wanted, saving memory and eliminating the need to throw away unwanted data.

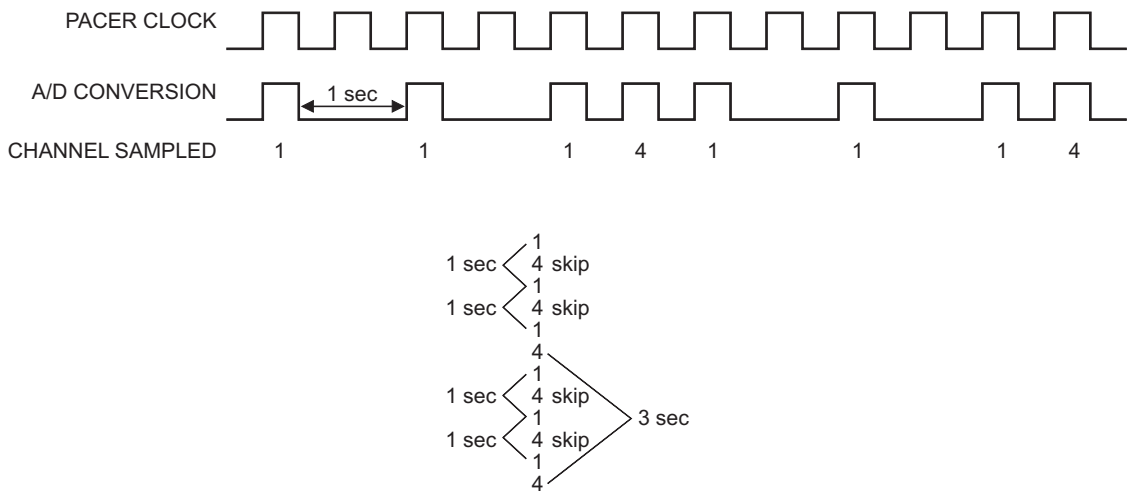


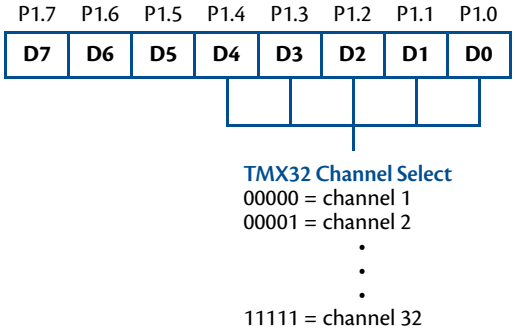
Figure 19 Timing Diagram for Sampling Channels 1 and 4

8-Bit Digital Table

The digital portion of the channel-gain table can be programmed with digital control information using the Digital Table Register at **BA + 0x04h**. To load digital control data into the Digital Table, first set bits 1 and 0 at **BA + 0x02h** to 10. This points **BA + 0x04h** to write to the Digital Table. Now you can write the 8-bit byte to **BA + 0x04h**. If you have cleared the existing table, the first byte written will be placed in the first entry of the table, the second byte will be placed in the second entry, and so on. If you are adding to an existing table, the new data written will be added at the end. The first entry made into the Digital Table lines up with the first entry made into the A/D Table, the second entry made into the Digital Table lines up with the second entry made into the A/D Table, and so on. Make sure that, if you add to an existing table and did not program the Digital Table portion when you made your A/D Table entries previously, you fill those entries with digital data first before entering the desired added data. Since the first digital entry you make always lines up with the first A/D entry made, failure to do this will cause the A/D and digital control data to be misaligned in the table. You cannot turn the digital control lines off for part of a conversion sequence and then turn them on for the remainder of the sequence. Note that the digital data programmed here is sent out on the Port 1 digital I/O lines whenever this portion of the table is enabled.

These lines can be used to control input expansion boards such as the TMX32 analog input expansion board at the same speed as the A/D conversions are performed with no software overhead.

Note *If you only need to use the A/D part of the table, you do not have to program the Digital Table. However if you only want to use the Digital part of the table you must program the A/D part of the table.*



Setting Up A/D and Digital Tables

Let's look at how the channel-gain table is set up for a simple example using both the A/D and Digital Tables. In this example, we have a TMX32 expansion board connected to channel 1 on the DM6425HR. With **BA + 0x02h**, bits 1 and 0 set to 01, load the channel-gain sequence into the A/D Table:

Entry 1	0000 0000 0000 0000	gain = 1, DM6425HR channel = 1
Entry 2	0000 0000 0100 0000	gain = 4, DM6425HR channel = 1
Entry 3	0001 0000 0000 0000	skip sample
Entry 4	0000 0000 0100 0000	gain = 4, DM6425HR channel = 1
Entry 5	0000 0000 0000 0000	gain = 1, DM6425HR channel = 1
Entry 6	0000 0000 0100 0000	gain = 4, DM6425HR channel = 1

With **BA + 0x00h**, bits 1 and 0 set to 10, load the digital data into the Digital Table. The first digital word loaded lines up with the first A/D Table entry, and so on:

Entry 1	0000 0000 0000 0000	gain = 1, DM6425HR channel = 1	0000 0000	TMX32 channel = 1
Entry 2	0000 0000 0100 0000	gain = 4, DM6425HR channel = 1	0000 0011	TMX32 channel = 4
Entry 3	0001 0000 0000 0000	skip sample	0000 0000	TMX32 channel = 1 (skip)
Entry 4	0000 0000 0100 0000	gain = 4, DM6425HR channel = 1	0000 0011	TMX32 channel = 4
Entry 5	0000 0000 0000 0000	gain = 1, DM6425HR channel = 1	0000 0000	TMX32 channel = 1
Entry 6	0000 0000 0100 0000	gain = 4, DM6425HR channel = 1	0000 0011	TMX32 channel = 4

Using the Channel-Gain Table for A/D Conversions

After the channel-gain table is programmed, it must be enabled to be used for A/D conversions. Two bits control this operation. **BA + 0x02h**, bit 2 enables the A/D Table where the channel and gain data are stored. **BA + 0x02h**, bit 3, enables the Digital Table when the digital control data is stored.

Whenever you want to use the channel-gain table, you must set bit 2 at **BA + 0x02h** high to enable the A/D Table. If you are also using the Digital Table, you must enable this portion of the channel-gain table by setting **BA + 0x02h**, bit 3 high. You cannot use the digital portion without enabling the A/D portion of the channel-gain table (bit 3 cannot be set high unless bit 2 is also high). When the Digital Table is enabled, the 8-bit data is sent out on the Port 1 digital I/O lines.

When you are using the channel-gain table to take samples, it is strongly recommended that you do not enable, disable, and then re-enable the table while performing a sequence of conversions. This causes skipping of an entry in the table. In this case you should issue a reset table command at **BA + 0x00h**.

Channel-Gain Table and Throughput Rates

When using the channel-gain table, you should group your entries to maximize the throughput of your module. Low-level input signals and varying gains are likely to drop the throughput rate because low level inputs must drive out high level input residual signals.

To maximize throughput:

- Keep channels configured for a certain range grouped together, even if they are out of sequence.
- Use external signal conditioning if you are performing high-speed scanning of low level signals. This increases throughput and reduces noise.
- If you have room in the channel-gain table, you can make an entry twice to make sure that sufficient settling time has been allowed and an accurate reading has been taken. Set the skip bit for the first entry so that it is ignored.
- For best results, do not use the channel-gain table when measuring steady-state signals. Use the single convert mode to step through the channels.

A/D Conversion Modes

To support a wide range of sampling requirements, the DM6425HR provides several conversion modes with a selection of trigger sources to start and stop a sequence of conversions. Understanding how these modes and sources can be configured to work together is the key to understanding the A/D conversion capabilities of your module.

The commands issued to the Trigger Registers at **BA + 0x06h** set up how the A/D conversions are controlled. The following paragraphs describe the conversion and trigger modes, and Figure 20 shows a block diagram of the A/D conversion select circuitry.

Start A/D Conversions

Bits 0 and 1 of the Trigger Register programmed at **BA + 0x06h** control what method is used to actually perform the A/D conversions. One of four modes can be selected:

- Through software (by reading **BA + 0x06h** to initiate a Start Convert)
- Using a pacer clock
  - internal (Clock TC Counter 0 or 1)
  - external (**CN3**, pin-41)
- Using the burst clock (Clock TC Counter 2)
- Using a digital interrupt generated by the Advanced Digital Interrupt circuit

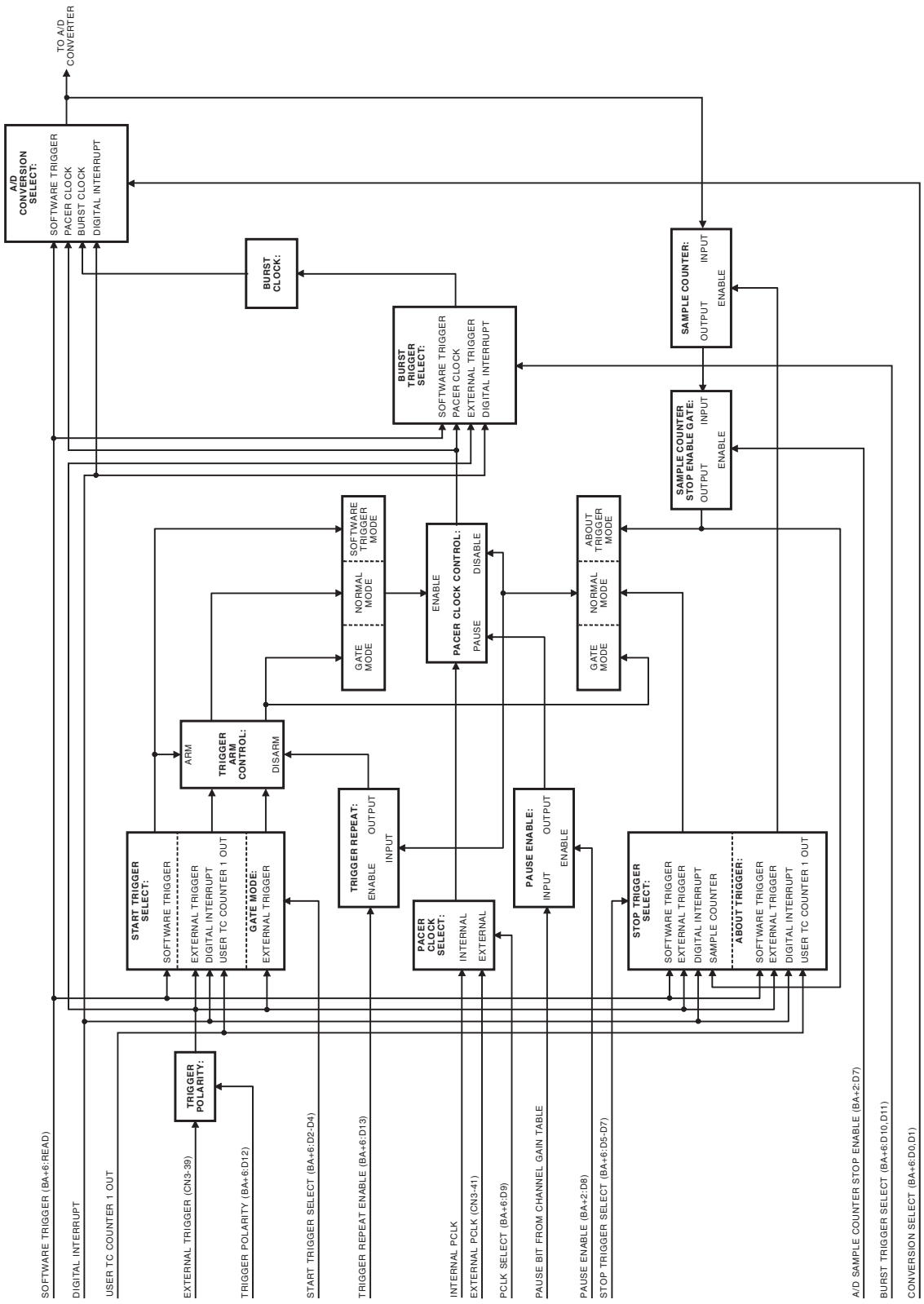


Figure 20 A/D Conversion Select Circuitry

Start/Stop Trigger Select

The start trigger set at bits 2 through 4 and the stop trigger set at bits 5 through 7 of the Trigger Register programmed at **BA + 0x06h** are used to turn the pacer clock (internal or external) on and off. Through these different combinations of start and stop triggers, the DM6425HR supports pre-trigger, post-trigger, and about-trigger modes with various trigger sources.

The five start trigger sources are:

- **Software trigger**. When selected, a read at **BA + 0x06h** will start the pacer clock.
- **External trigger**. When selected, a positive- or negative-going edge (depending on the setting of the trigger polarity, bit 12 in the Trigger Register) on the external TRIGGER IN line (**CN3**, pin 39) will start the pacer clock. The pulse duration should be at least 100 ns.
- **Digital interrupt**. When selected, a digital interrupt will start the pacer clock.
- **User TC Counter 1 output**. When selected, a pulse on the Counter 1 output line (Counter 1's count reaches 0) will start the pacer clock.
- **Gate mode**. When selected, the pacer clock runs when the external TRIGGER IN line (**CN3**, pin 39) is held high. When this line goes low, conversions stop. This trigger mode does not use a stop trigger. If the trigger polarity bit is set for negative, the pacer clock runs when this line is low and stops when it is taken high.

The eight stop trigger sources are listed below. The last four stop trigger sources provide about triggering, where data is acquired from the time the start trigger is received, and continues for a specified number of samples after the stop trigger. The number of samples to acquire after the stop trigger is programmed in the sample counter.

- **Software trigger**. When selected, a read at **BA + 0x06h** will stop the pacer clock.
- **External trigger**. When selected, a positive- or negative-going edge (depending on the setting of the trigger polarity, bit 12 in the Trigger Register) on the external TRIGGER IN line (**CN3**, pin 39) will stop the pacer clock. The pulse duration should be at least 100 ns.
- **Digital interrupt**. When selected, a digital interrupt will stop the pacer clock.
- **Sample counter**. When selected, the pacer clock stops when the sample counter's count reaches 0.
- **About software trigger**. When selected, a software trigger starts the sample counter, and sampling continues until the sample counter's count reaches 0.
- **About external trigger**. When selected, an external trigger starts the sample counter, and sampling continues until the sample counter's count reaches 0.
- **About digital interrupt**. When selected, a digital interrupt starts the sample counter, and sampling continues until the sample counter's count reaches 0.
- **About User TC Counter 1 output**. When selected, a pulse on the Counter 1 output line (Counter 1's count reaches 0) starts the sample counter, and sampling continues until the sample counter's count reaches 0.

Note that the external trigger (TRIGGER IN) can be set to occur on a positive-going edge or a negative-going edge, depending on the setting of bit 12 in the Trigger Register at **BA + 0x06h**.

Triggering a Burst Sample

These triggers, set at Trigger Register bits 10 and 11, **BA + 0x06h**, can trigger bursts:

- Through software (by reading **BA + 0x06h** to initiate a Start Convert)
- Using a pacer clock: internal (Clock TC Counter 0 or 1) or external (**CN3**, pin 41)
- Using an external trigger (**CN3**, pin 39)
- Using the digital interrupt

Trigger Repeat Function

Bit 13 in the Trigger Register at **BA + 0x06h** lets you control the conversion sequence when using a trigger to start the pacer clock. When this bit is low, the first pulse on the trigger line will start the pacer clock. After the stop trigger has ended the conversion cycle, the triggering circuit is disarmed and must be rearmed before another start trigger can be recognized. To rearm this trigger circuit, you must issue a software start convert (read **BA + 0x06h**).

When bit 13 in the Trigger Register, **BA + 0x06h**, is high, the conversion sequence is repeated each time an external trigger is received. Figure 21 shows a timing diagram for this feature.

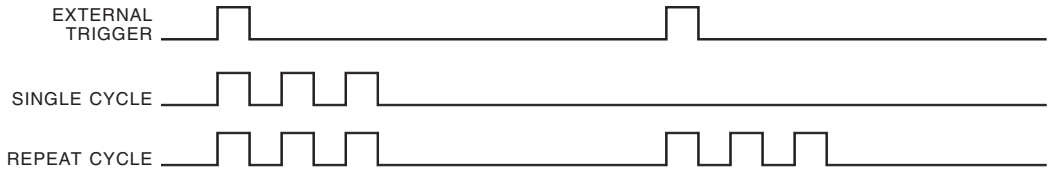


Figure 21 External Trigger Single Cycle vs. Repeat Cycle

Pacer Clock Source

The pacer clock can be generated from an internal source (Clock TC Counter 0 or 1) or an external source (**CN3**, pin 41) by setting bit 9 in the Trigger Register at **BA + 0x06h** as desired.

Types of Conversions

Single Conversion

In this mode, a single specified channel is sampled whenever the Start Convert line is taken high by a read at **BA + 0x06h** (software trigger). The active channel is the one specified in the Channel/Gain Register, bits 0 through 6.

This is the easiest of all conversions. It can be used in a wide variety of applications, such as sample every time a key is pressed on the keyboard, sample with each iteration of a loop, or watch the system clock and sample every five seconds. Figure 22 shows a timing diagram for single conversions.

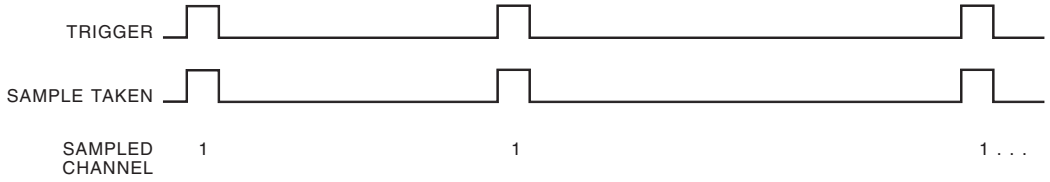


Figure 22 Timing Diagram, Single Conversion

Multiple Conversions

In this mode, conversions are continuously performed at the pacer clock rate. The pacer clock can be internal or external. The maximum rate supported by the module is 500 kHz. The pacer clock can be turned on and off using any of the start and stop triggering modes set up in the Trigger Register at **BA + 0x06h**. If you use the internal pacer clock, you must program it to run at the desired rate.

This mode is ideal for filling arrays, acquiring data for a specified period of time, and taking a specified number of samples. Figure 23 shows a timing diagram for multiple conversions.

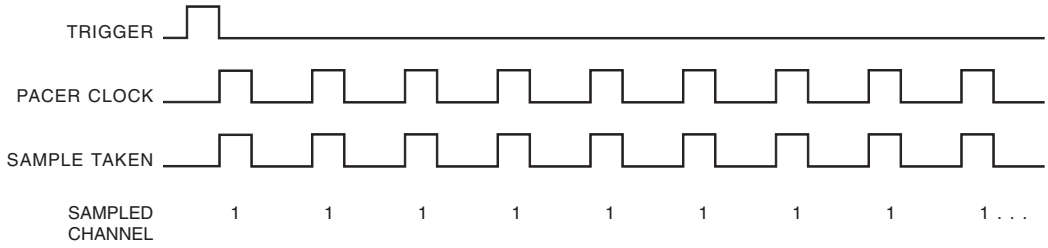


Figure 23 Timing Diagram, Multiple Conversions

Random Channel Scan

In this mode, the channel-gain table is incrementally scanned through, with each pacer clock pulse starting a conversion at the channel and gain specified in the current table entry. Before starting a conversion sequence using the channel-gain table, you need to load the table with the desired data. Then make sure that the channel-gain table is enabled by setting bit 2 at **BA + 0x02h** high. This enables the A/D portion of the channel-gain table. If you are using the Digital Table as well, you must also set bit 3 at **BA + 0x02h** high. Each clock pulse starts a conversion using the current channel-gain data and then increments to the next position in the table. When the last entry is reached, the next pulse starts the table over again. Figure 24 shows a timing diagram for random channel scanning.

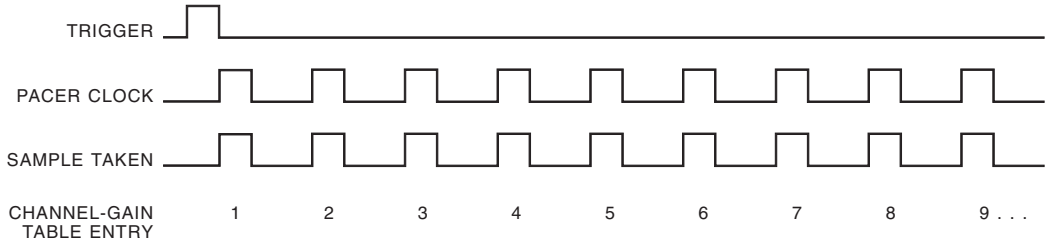


Figure 24 Timing Diagram, Random Channel Scan

Programmable Burst

In this mode, a single trigger initiates a scan of the entire channel-gain table. Before starting a burst of the channel-gain table, you need to load the table with the desired data. Then make sure that the channel-gain table is enabled by setting bit 2 at **BA + 0x02h** high. This enables the A/D portion of the channel-gain table. If you are using the Digital Table as well, you must also set bit 3 at **BA + 0x02h** high.

Burst is used when you want one sample from a specified number of channels for each trigger. Figure 25 shows a timing diagram for burst sampling. As shown, the burst trigger, which is a trigger or pacer clock, triggers the burst and the burst clock initiates each conversion. At high speeds, the burst mode emulates simultaneous sampling of multiple input channels. For time critical simultaneous sampling applications, a simultaneous sample-and-hold board can be used (SS8 eight-channel boards are available from RTD).

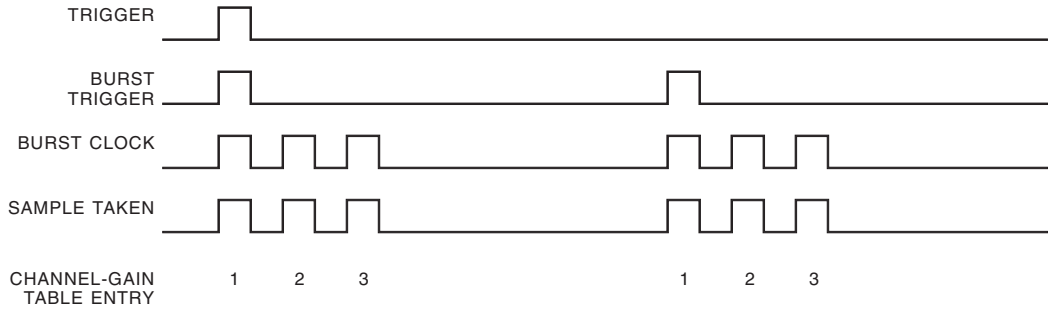


Figure 25 Timing Diagram, Programmable Burst

Programmable Multi-Scan

This mode lets you scan the channel-gain table a specified number of times for each trigger. The total number of samples to be taken is programmed into the sample counter. For example, if you want to take two bursts of a three-entry channel-gain table, as shown in the timing diagram of Figure 26, you would program the sample counter to take six samples. Note that if you do not program the sample counter with a multiple of the number of entries in the channel-gain table, the sample counter's count will not be 0 when the last burst sequence has been completed, which means that the sample counter will not start at the beginning of the countdown the next time you use it unless it has been reprogrammed.

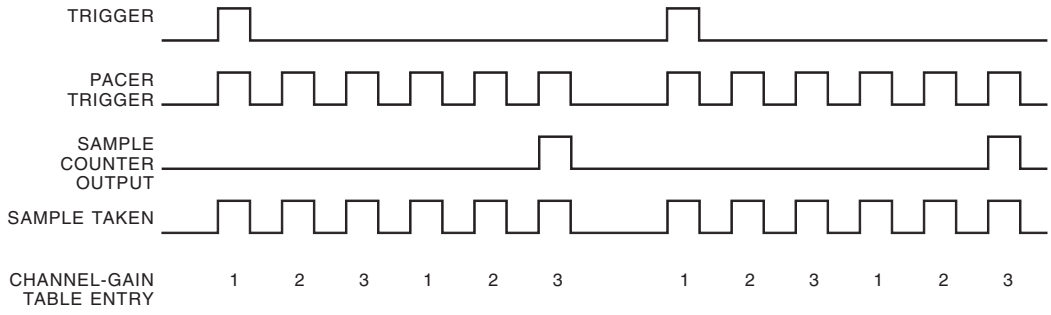


Figure 26 Timing Diagram, Programmable Multi-Scan

As you can see, the DM6425HR is designed to support a wide range of conversion requirements. You can set the clocks, triggers, and channel and gain to a number of configurations to perform simple or very complex acquisition schemes where multiple bursts are taken at timed intervals. Remember that the key to configuring the module for your application is to understand what signals can actually control conversions and what signals serve as triggers. The diagrams and discussions presented in this section and the example programs on the disk should help you to understand how to configure the module.

Starting an A/D Conversion

Depending on your conversion and trigger settings, the software trigger command (read at **BA + 0x06h**) has different functions. In any mode that uses the software trigger, this command will do the appropriate action. For example, if you set the start trigger as software trigger, the read at **BA + 0x06h** will start the pacer clock running. However, in any mode that does not use the software trigger as the trigger, you will still need to do a read at **BA + 0x06h** to arm (enable) the triggering circuitry. An example of this would be, if you set the start trigger as external trigger, a read at **BA + 0x06h** is required to arm the external trigger circuitry. After you have set all the trigger and conversion registers to the proper values, the last command will need to be a software trigger. Any external triggers received before this command will be ignored. It is also a good practice to clear the A/D FIFO just prior to triggering the measurement or arming the trigger. Study the example programs to see this sequence.

Monitoring Conversion Status (FIFO Empty Flag or End-of-Convert)

The A/D conversion status can be monitored through the FIFO empty flag or through the end-of-convert (EOC) bit in the status word read at **BA + 0x02h**. Typically, you will want to monitor the EF flag for a transition from low to high. This tells you that a conversion is complete and data has been placed in the sample buffer. The EOC line is available for monitoring conversion status in special applications.

Halting Conversions

In single convert modes, a single conversion is performed and the module waits for another Start Convert command. In multi-convert modes, conversions are halted by one of two methods: when a stop trigger has been issued to stop the pacer clock, or when the FIFO is full. The halt flag, bit 2 of the status word (**BA + 0x02h**), is set when the sample buffer is full, disabling the A/D converter. Even if you have removed data from the sample buffer since the buffer filled up and the FIFO full flag is no longer set, the halt bit will confirm that at some point in your conversion sequence, the sample buffer filled and conversions were halted. At this point a clear FIFO command must be issued and a software start convert (read at **BA + 0x06h**) to rearm the trigger circuitry.

Reading the Converted Data

Each 12-bit conversion is stored in a 16-bit word in the sample buffer. The buffer can store 1024 samples. This section explains how to read the data stored in the sample buffer.

The sample buffer contains only the converted data and 3-bit data marker (if used) in a 16-bit word. The 12-bit A/D data + sign bit is left justified in a 16-bit word, with the least significant three bits reserved for the data marker. Because of this, the A/D data read must be scaled to obtain a valid A/D reading. The data marker portion should be masked out of the final A/D result. Shifting the word three bits to the right will eliminate the data marker from the data word. If you are using the data marker, then you should preserve these bits someplace in your program.

The output code format is always two's complement. This is true for both bipolar and unipolar signals since the sign bit is added above the 12-bit conversion data. For bipolar conversions, the sign bit will follow the MSB of the 12-bit data. If this bit is a "0", the reading is a positive value. If this bit is a "1", the reading is a negative value. When the input is a unipolar range, the coding is the same except that the sign bit is always a "0" indicating a positive value. The data should always be read from the A/D FIFO as a signed integer.

Voltage values for each bit will vary depending on input range and gain. For example, if the input is set for ±5 V and the gain = 1, the formula for calculating voltage is as follows:

Voltage = [(Input Range/Gain) / 4096] x Conversion Data

Voltage = [(10/1) / 4096] x Conversion Data

Voltage = 2.44 mV x Conversion Data

Remember that when you change the gain, you are increasing the resolution of the bit value but you are decreasing the input range. In the above example if we change the gain to 4, each bit will now be equal to 610 µV but our input range is decreased from 10 V to 2.5 V. The formula would look like this:

Voltage = [(Input Range/Gain) / 4096] x Conversion Data

Voltage = [(10/4) / 4096] x Conversion Data

Voltage = 610 µV x Conversion Data

If we now change the input range to ±10 V and the gain = 1, the formula would be:

Voltage = [(Input Range/Gain) / 4096] x Conversion Data

Voltage = [(20/1) / 4096] x Conversion Data

Voltage = 4.88 mV x Conversion Data

The key digital codes and their input voltage values are given in the following tables. The bit map below shows the configuration of the A/D data. Bit 15 is the sign extension bit.

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	30.7	30.6	30.5
	(MSB)											**(LSB)**	**A/D Data Markers**		

A/D Bipolar Code Table[1]
(±5 V Input Range)

Input Voltages	Sign	Output Code
+4.998 V	0	MSB 0111 1111 1111 LSB
+2.500 V	0	MSB 0100 0000 0000 LSB
0.000 V	0	MSB 0000 0000 0000 LSB
–0.00244 V	1	MSB 1111 1111 1111 LSB
–5.000 V	1	MSB 1000 0000 0000 LSB

1. 1 LSB = 2.44 mV.

A/D Bipolar Code Table[1]
(±10 V Input Range)

Input Voltages	Sign	Output Code
+9.995 V	0	MSB 0111 1111 1111 LSB
+5.000 V	0	MSB 0100 0000 0000 LSB
0.000 V	0	MSB 0000 0000 0000 LSB
–0.00488 V	1	MSB 1111 1111 1111 LSB
–10.000 V	1	MSB 1000 0000 0000 LSB

1. 1 LSB = 2.44 mV.

A/D Bipolar Code Table[1]
(0 to 10 V Input Range)

Input Voltages	Sign	Output Code
+9.99756 V	0	MSB 1111 1111 1111 LSB
+5.000 V	0	MSB 1000 0000 0000 LSB
0.000 V	0	MSB 0000 0000 0000 LSB

1. 1 LSB = 2.44 mV.

Using the A/D Data Markers

For certain applications where you may want to store digital information with the analog data at the same rate the analog data is being acquired, as shown in Figure 27, the bottom three bits of the converted data are available for this feature. For example, you may want to tag the acquired data with a marker so that you know when the data was sampled. Three lines are available at I/O connector **CN3** to send the data marker settings to the sample buffer along with the 12-bit A/D converted data. These lines are P0.5 (**CN3**, pin 27), P0.6 (**CN3**, pin25), and P0.7 (**CN3**, pin 23).

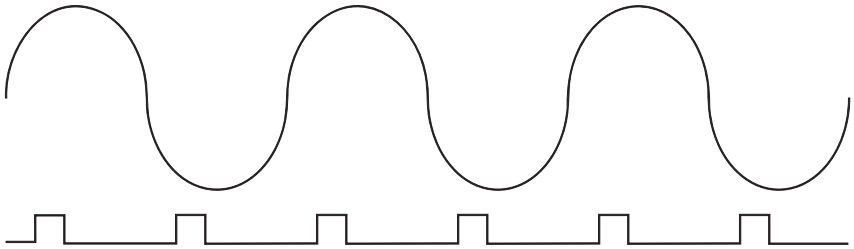


Figure 27 Storing Digital Data with Analog Data at the Acquisition Rate

Programming the Pacer Clock

Two 16-bit timers in the Clock TC, Counters 0 and 1, are cascaded to form a 16-bit or 32-bit on-board pacer clock, shown in Figure 28. When you want to use the pacer clock for continuous A/D conversions, you must select a 16-bit or 32-bit clock configuration and program the clock rate.

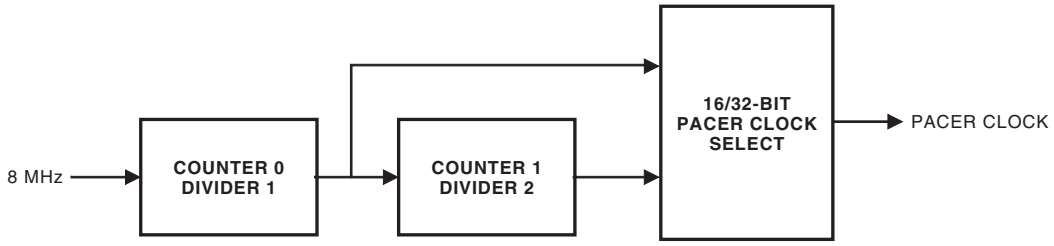


Figure 28 Pacer Clock Block Diagram

Selecting 16-bit or 32-bit Pacer Clock

The size of the pacer clock, 16-bit or 32-bit, is programmed at bit 8 of the Trigger Register at **BA + 0x06h**. When this bit is set to 0, a 16-bit pacer clock is selected. Whenever possible, it is strongly recommended that the 16-bit pacer clock be used to minimize the delay between the time a trigger occurs and the first conversion is initiated by the pacer clock. When using a 16-bit clock, the first conversion will always start within 250 ns of the trigger, and subsequent conversions are synchronized to the pacer clock. The 16-bit clock conversion speeds can be set from 500 kHz down to 123 Hz.

Because the 32-bit pacer clock cascades two 16-bit timers, the uncertainty between the time a trigger occurs and the first conversion is initiated can be significantly greater than for the 16-bit clock. The triggering uncertainty here is based on the value programmed into the first divider and can become unacceptable for certain applications. However, for conversion rates slower than 123 Hz, you must use the 32-bit pacer clock. The 32-bit clock is selected by setting bit 8 in the Trigger Register to 1. When programming the 32-bit clock, you should always program the smallest possible value in Divider 1 in order to minimize the triggering uncertainty.

Programming Steps

The pacer clock is accessed for programming by setting bits 6 and 5 at **BA + 0x02h** to 00. To find the value you must load into the clock to produce the desired rate, you first have to calculate the value of Divider 1 (Clock TC Counter 0) for a 16-bit clock, or the value of Divider 1 and Divider 2 (Clock TC Counter 1) for a 32-bit clock, as shown in Figure 28. The formulas for making this calculation are as follows:

16-bit pacer clock frequency = 8 MHz/(Divider 1)

Divider 1 = 8 MHz/16-bit Pacer Clock Frequency

32-bit pacer clock frequency = 8 MHz/(Divider 1 x Divider 2)

Divider 1 x Divider 2 = 8 MHz/32-bit Pacer Clock Frequency

To set the 16-bit pacer clock frequency at 500 kHz, this equation becomes:

Divider 1 = 8 MHz/500 kHz $\Rightarrow$ 16 = 8 MHz/500 kHz

When Divider 1 is greater than 65,536, you will have to select a 32-bit pacer clock and program the clock rate into Dividers 1 and 2. When programming the 32-bit clock, divide the result by the least common denominator. The least common denominator is the value that is loaded into Divider 1, and the result of the division, the quotient, is loaded into Divider 2. The tables below list some common pacer clock frequencies and the counter settings for a 16-bit and a 32-bit pacer clock. After you calculate the decimal value of each divider, you can convert the result to a hex value if it is easier for you when loading the count into each 16-bit counter.

16-Bit Pacer Clock	Divider 1 decimal/(hex)
500 kHz	16/(0010)
100 kHz	80/(0050)
50 kHz	160/(00A0)
10 kHz	800/(0320)
1 kHz	8000/(1F40)

32-Bit Pacer Clock	Divider 1 decimal/(hex)	Divider 2 decimal/(hex)
100 kHz	2/(0002)	40000/(9C40)
10 kHz	16/(0010)	50000/(C350)

To set up the 16-bit pacer clock:

1. Set pacer clock size to 16 bits (bit 8 of Trigger Register at **BA + 0x06h** = 0).
2. Set **BA + 0x02h**, bits 6 and 5 to 00 to talk to the Clock TC.
3. Program Counter 0 for Mode 2 operation.
4. Load Divider 1 LSB.
5. Load Divider 1 MSB.

To set up the 32-bit pacer clock:

1. Set pacer clock size to 32 bits (bit 8 of Trigger Register at **BA + 0x06h** = 1).
2. Set **BA + 0x02h**, bits 6 and 5 to 00 to talk to the Clock TC.
3. Program Counter 0 for Mode 2 operation.
4. Program Counter 1 for Mode 2 operation.
5. Load Divider 1 LSB.
6. Load Divider 1 MSB.
7. Load Divider 2 LSB.
8. Load Divider 2 MSB.

Depending on your conversion mode, the counters start their countdown and the pacer clock starts running when a trigger occurs.

Programming the Burst Clock

The third 16-bit timer in the Clock TC, Counter 2, is the onboard burst clock. When you want to use the burst clock for performing A/D conversions in the burst mode, you must program the clock rate. To find the value you must load into the clock to produce the desired rate, make the following calculation:

Burst clock frequency = 8 MHz/Counter 2 Divider

To set the burst clock frequency at 100 kHz using the onboard 8 MHz clock source, this equation becomes:

Burst clock frequency = 8 MHz/100 kHz $\Rightarrow$ 80 = 8 MHz/100 kHz

After you determine the value that will result in the desired clock frequency, load it into Counter 2. In this case, decimal 80 (hex 0050) is loaded into the counter.

To set up the burst clock:

1. Set **BA + 0x02h**, bits 6 and 5 to 00 to talk to the Clock TC.
2. Program Counter 2 for Mode 2 operation.
3. Load Divider LSB.
4. Load Divider MSB.

Depending on your conversion mode, the counter start its countdown and the burst clock starts running when a trigger occurs.

Programming the Sample Counter

The sample counter lets you program the DM6425HR to take a certain number of samples and then halt conversions. The number of samples to be taken is loaded into the 16-bit sample counter, User TC Counter 2. Recall that because of the operating structure of the 8254, the count loaded initially is not the count which is counted down during the first cycle. A software correction is used as an easy means to compensate for this. Two pulses of the counter are required to actually load the desired count and prepare the counter to count down correctly (this can be looked at as the initialization procedure for the sample counter). A pulse is sent to the 8254 sample counter each time you read **BA + 0x0Eh**. Without this correction, the initial count sequence will be off by two pulses. Note that once the counter is properly loaded and starts, any subsequent countdowns of this count will be accurate.

After you determine the desired number of samples, load the count into User TC Counter 2.

To set up the sample counter:

1. Set **BA + 0x02h**, bits 6 and 5 to 01 to talk to the User TC.
2. Program Counter 2 for Mode 2 operation.
3. Load Count LSB.
4. Load Count MSB.
5. Pulse line by reading **BA + 0x0Eh** two times so that the loaded count matches the desired count.

Using the Sample Counter to Create Large Data Arrays

The 16-bit sample counter allows you to take up to 65,536 samples before the count reaches 0 and sampling is halted. Suppose, however, you want to take 100,000 samples and stop. The DM6425HR provides a bit in the Control Register at **BA + 0x02h** which allows you to use the sample counter to take more than 65,536 samples in a conversion sequence.

Bit 7 in the Control Register, the sample counter stop enable bit, can be set to 1 to allow the sample counter to continuously cycle through the loaded count until the stop enable bit is set to 0, which then causes the sample counter to stop at the end of the current cycle. Let's look back at our example where we want to take 100,000 readings. First, we must divide 100,000 by a whole number that gives a result of less than 65,536. In our example, we can divide as follows:

Sample Counter Count = 100,000 / 2 = 50,000

To use the sample counter to take 100,000 samples, we will load a value of 50,000 into the counter and cycle the counter two times. After the value is loaded, make sure that bit 7 in the Control Register is set to 1 so that the sample counter will cycle. Then, set up the sample counter so that it generates an interrupt when the count reaches 0. Initialize the sample counter as described in the preceding section and start the conversion sequence. When the sample counter interrupt occurs telling you that the count has reached 0 and the cycle is starting again, set bit 7 in the Control Register to 0 to stop the sample counter after the second cycle is completed. The result: the sample counter runs through the count two times and 100,000 samples are taken. Figure 29 shows a timing diagram for this example.

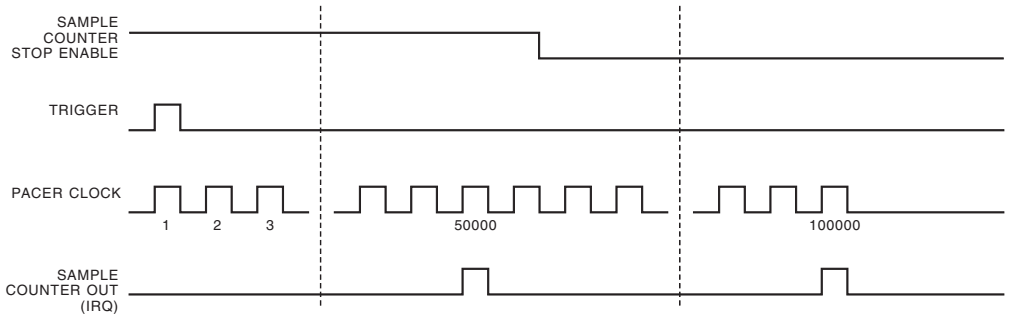


Figure 29 Timing Diagram for Cycling the Sample Counter

Analog Trigger

The DM6425HR has a unique feature that enables it to work like a digital oscilloscope. The board has the ability to trigger when a certain voltage level is reached on any one of the analog inputs.

A register threshold value entered by the user has a built-in hysteretic span for noise, which depends on the front end range. This value is approximately 39 mV for the 0 to +10 V and ± 5 V ranges, and approximately 78 mV for the ± 10 V range. The hysteresis range is split above and below the threshold value. In essence, the threshold value is reduced from twelve bits to eight bits as a filter for environmental noise that might occur.

The user also selects which front end channel to use as a trigger and whether to trigger on a signal that is greater than or less than or equal to the threshold value. Depending on the mode (Channel Gain Latch or Channel Gain Table), the board will then take A/D samples without writing them to the FIFO until the threshold value is reached by the appropriate channel. An onboard one shot is then triggered, which allows samples to be written into the FIFO.

When using this board feature it is important to follow a sequence of driver command functions to avoid potential false triggers. Source code should address the following register sequence:

1. Disable A/D FIFO writes **BA + 0x02h**, bit 10.
2. Set Threshold Value to include trigger channel and >, < bit (BA + 0x410h).
3. Clear A/D FIFO **BA + 0x00h**, bit 2.
4. Start A/D conversions. (This depends on which trigger mechanism is used.)
5. Set Use Threshold bit (**BA + 0x410h**, bit 14).

It is important that the Use Threshold bit (**BA + 0x410h**, bit 14) and the rest of the register (channel, threshold value, and boolean comparison) are written to in a 2-step procedure as described above. This will allow values to be set up within the register before enabling, which in turn will protect against false triggers.

Below is a table showing some of the possible threshold values as related to range and polarity

Table 8 Analog Trigger Threshold Values

Threshold Value (hex)	± 5 V	± 10 V	0 to +10 V
0	0.000	0.000	0.000
1	0.039	0.078	0.039
F	0.586	1.172	0.586
1F	1.211	2.422	1.211
2F	1.836	3.672	1.836
3F	2.461	4.922	2.461
4F	3.086	6.172	3.086
5F	3.711	7.422	3.711
6F	4.336	8.672	4.336
7F	4.961	9.922	4.961
8F	−4.414	−8.828	5.586
9F	−3.789	−7.578	6.211
FF	−0.039	−0.078	9.961

Mathematical formula for calculating threshold voltages:

	± 5 V	± 10 V	0 to +10 V
Positive	(Hex Value) * (10 / (Gain * 256))	(Hex Value) * (20 / (Gain * 256))	(Hex Value) * (10 / (Gain * 256))
Negative	[(Hex Value) * (10 / Gain * 256))] – (9.961 / Gain)	[(Hex Value) * (20 / (Gain * 256))] – (19.922 / Gain)	—

Chapter 6 Data Transfers Using DMA

This chapter explains how data transfers are accomplished using Direct Memory Access (DMA).

DMA transfers data between a peripheral device and PC memory without using the processor as an intermediate. Bypassing the processor in this way allows very fast transfer rates. All PCs contain the necessary hardware components for accomplishing DMA. However, software support for DMA is not included as part of the BIOS or DOS, leaving you with the task of programming the DMA controller yourself. With a little care, such programming can be successfully and efficiently achieved.

The following discussion is based on using the DMA controller to get data from a peripheral device and write it to memory. The opposite can also be done; the DMA controller can read data from memory and pass it to a peripheral device. There are a few minor differences, mostly in programming the DMA controller, but in general the process is the same.

The following steps are required when using DMA:

1. Choose a DMA channel.
2. Allocate a buffer.
3. Calculate the page and offset of the buffer.
4. Set the DMA page register.
5. Program the 8237 DMA controller.
6. Program device generating data (DM6425HR).
7. Enable DMA channel.
8. Wait until DMA is complete.
9. Disable DMA channel.

Each step is detailed in the following sections.

Choosing a DMA Channel

There are a number of DMA channels available on the PC for use by peripheral devices. The DM6425HR can use DMA channel 5, 6, or 7, selected through software. You can arbitrarily choose any of these; in most cases your choice will be fine. Occasionally though, you will have another peripheral device (for example, a tape backup or Bernoulli drive) that also uses the DMA channel you have selected. This will certainly cause erratic results and can be hard to detect. The best approach to pinpoint this problem is to read the documentation for the other peripheral devices in your system and try to determine which DMA channel each uses.

Allocating a DMA Buffer

When using DMA, you must have a location in memory where the 8237 DMA controller will place the 16-bit data words which contain the 12-bit A/D converted data from the DM6425HR. This buffer can be either static or dynamically allocated. The buffer must start on a word boundary (i.e., even numbered address). You should force your compiler to use word alignment for data. Be sure that its location will not change while DMA is in progress. The following code examples show how to allocate buffers for use with DMA.

In Pascal:

```
Var Buffer : Array[1..10000] of Byte;{ static allocation }
```

-or-

```
Var Buffer : ^Byte;                 {dynamic allocation }
. . .
Buffer := GetMem(10000);
```

In C:

```
char Buffer[10000];                 /* static allocation */
```

-or-

```
char *Buffer;                       /* dynamic allocation */
. . .
Buffer = calloc(10000, 0);
```

Calculating the Page and Offset of a Buffer

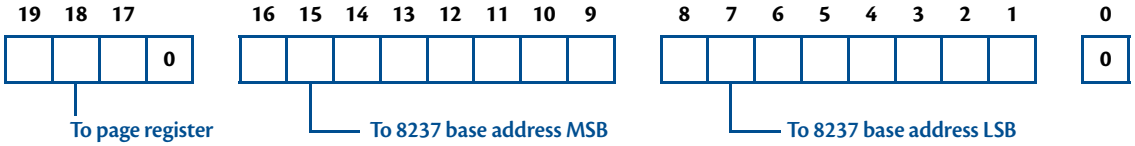
Once you have a buffer into which to place your data, you must inform the 8237 DMA controller of the location of this buffer. This is a little more complex than it sounds because the DMA controller uses a page:offset memory scheme, while you are probably used to thinking about your computer's memory in terms of a segment:offset scheme. Paged memory is simply memory that occupies contiguous, non-overlapping blocks of memory, with each block being 64K (one page) in length. The first page (page 0) starts at the first byte of memory, the second page (page 1) starts at byte 65536, the third page (page 2) at byte 131072, and so on. A computer with 640K of memory has 10 pages of memory.

The DMA controller can write to (or read from) only one page without being reprogrammed. This means that the DMA controller has access to only 64K of memory at a time. If you program it to use page 3, it cannot use any other page until you reprogram it to do so.

When DMA is started, the DMA controller is programmed to place data at a specified offset into a specified page (for example, start writing at word 512 of page 3). Each time a word of data is written by the controller, the offset is automatically incremented so the next word will be placed in the next memory location. The problem for you when programming these values is figuring out what the corresponding page and offset are for your buffer. Most compilers contain macros or functions that allow you to directly determine the segment and offset of a data structure, but not the page and offset. Therefore, you must calculate the page number and offset yourself. Probably the most intuitive way of doing this is to convert the segment:offset address of your buffer to a linear address and then convert that linear address to a page:offset address. The table below shows functions/macros for determining the segment and offset of a buffer.

Language	Segment	Offset
C	FP_SEG s = FP_SEG(&Buffer)	FP_OFF o = FP_OFF(&Buffer)
Pascal	Seg S := Seg(Buffer)	Ofs O := Ofs(Buffer)

Once you have determined the segment and offset, multiply the segment by 16 and add the offset to give you the linear address. (Make sure you store this result as a long integer, or DWORD, or the results will be meaningless.) The linear address is a 20-bit value, with the upper 4 bits representing the page and the lower 16 bits representing the offset into the page. Even though the upper 4 bits are the page, only the upper 3 bits, D17, D18, and D19, are sent to what is called the page register. The remaining bit for the page, D16, is sent to the base address register of the DMA controller along with bits D1 through D15. Since the buffer sits on a word boundary, bit D0 must be zero, and is ignored. The following diagram shows you to which registers the components of the 20-bit linear address are sent.



The following examples show you how to calculate the linear address and break it into components to be sent to the various registers.

In Pascal:

```
Segment := SEG(Buffer);                          { get segment of buffer }
Offset := OFS(Buffer);{ get offset of buffer }
LinearAddress := Segment * 16 + Offset;          { calculate linear address }
PageBits := (LinearAddress DIV 65536) AND $0E;{ determine page corresponding
                                                 to this,linear address and
                                                 clear least significant bit }
OffsetBits := (LinearAddress SHR 2) MOD 65536;{ shift linear address to
                                                ignore D0 then extract bits
                                                D1-D16 }
```

In C:

```
segment = FP_SEG(&Buffer);                   /* get segment of buffer */
offset = FP_OFS(&Buffer);                    /* get offset of buffer */
linear_address = segment * 16 + offset;      /* calculate linear address */
pagebits = (linear_address / 65536) & 0x0E;  /* determine page
                                                corresponding to this linear
                                                address and clear least
                                                significant bit */
offset_bits = (linear_address >> 2) % 65536; /* shift linear address to
                                                ignore D0 then extract bits
                                                D1-D16 */
```

CAUTION! There is one big catch when using page-based addresses. The 8237 DMA controller cannot write properly to a buffer that 'straddles' a page boundary. A buffer straddles a page boundary if one part of the buffer resides in one page of memory while another part resides in the following page. The DMA controller cannot properly write to such a buffer because the DMA controller can only write to one page without reprogramming. When it reaches the end of the current page, it does not start writing to the next page. Instead, it starts writing back at the first byte of the current page. This can be disastrous if the beginning of the page does not correspond to your buffer. More often than not, this location is being used by the code portion of your program or the operating system, and writing data to it will almost always causes erratic behavior and an eventual system crash.

You must check to see if your buffer straddles a page boundary and, if it does, take action to prevent the DMA controller from trying to write to the portion that continues on the next page You can reduce the size of the buffer or try to reposition the buffer. However, this can be difficult when using large static data structures, and often, the only solution is to use dynamically allocated memory.

Setting the DMA Page Register

Oddly enough, you do not inform the DMA controller directly of the page to be used. Instead, you put the page to be used into the DMA page register, with the least significant bit set to zero. The DMA page register is separate from the DMA controller, as shown in the table below.

DMA Channel	Location of Page Register
5	8Bh/(139)
6	89h/(137)
7	8Ah/(138)

The DMA Controller

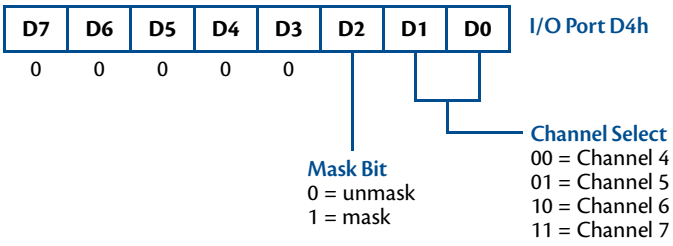
The DMA controller is made up of two complex 8237 chips, one for DMA channels 0–3, and one for channels 4–7, that occupy 32 contiguous bytes of the AT I/O port space starting with port C0h. A complete discussion of how it operates is beyond the scope of this manual; only relevant information is included here. The DMA controller is programmed by writing to the DMA registers in your AT. The table below lists these registers.

DMA Registers	
Base Address Hex/(Decimal)	**Description**
8Bh/(139)	Channel 5 DMA Page Select
C4h/(196)	Channel 5 DMA Base Address
C6h/(198)	Channel 5 DMA Count
89h/(137)	Channel 6 DMA Page Select
C8h/(200)	Channel 6 DMA Base Address
CAh/(202)	Channel 6 DMA Count
8Ah/(138)	Channel 7 DMA Page Select
CCh/(204)	Channel 7 DMA Base Address
CEh/(206)	Channel 7 DMA Count
D4h/(212)	Mask Register
D6h/(214)	Mode Register
D8h/(216)	Byte Pointer Flip-Flop

If you are using DMA channel 5, write your page offset bits to port C4h and the count to C6h; for channel 6, write the offset to C8h and the count to CAh; for channel 7, write the offset to CCh and the count to CEh. The page offset bits are the bits you calculated as shown above. Count indicates the number of samples that you want the DMA controller to transfer. The value that you write to the DMA controller is (number of samples – 1). The mask register and mode register are described next.

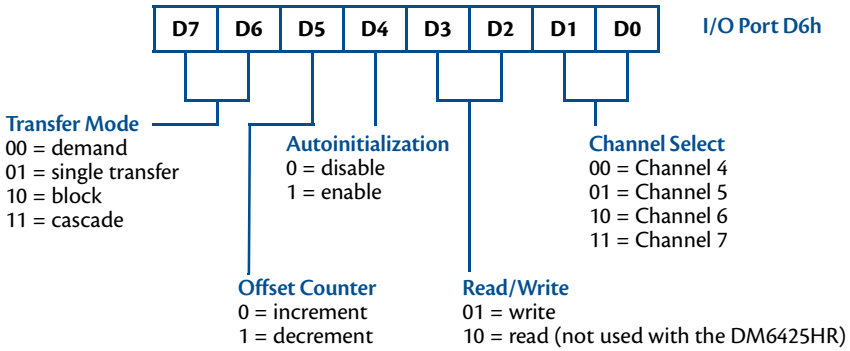
DMA Mask Register

The DMA mask register is used to enable or disable DMA on a specified DMA channel. You should mask (disable) DMA on the DMA channel you will be using while programming the DMA controller. After the DMA controller has been programmed and the DM6425HR has been programmed to sample data, you can enable DMA by clearing the mask bit for the DMA channel you are using. You should manually disable DMA by setting the mask bit before exiting your program or, if for some reason, sampling is halted before the DMA controller has transferred all the data it was programmed to transfer. If you leave DMA enabled and it has not transferred all the data it was programmed to transfer, it will resume transfers the next time data appears at the A/D converter. This can spell disaster if your program has ended and the buffer has been reallocated to another application.



DMA Mode Register

The DMA mode register is used to set parameters for the DMA channel you will be using. The read/write bits are self explanatory; the read mode cannot be used with the DM6425HR. Autoinitialization allows the DMA controller to automatically start over once it has transferred the requested number of words. Decrement means the DMA controller should decrement its offset counter after each transfer; the default is increment. We recommend that you use either the demand or single transfer mode when transferring data. The DM6425HR does not support block mode transfer.



Programming the DMA Controller

To program the DMA controller:

1. Disable DMA on the channel you are using.
2. Write the DMA mode register to choose the DMA parameters.
3. Write the page offset bits (D1–D16) of your buffer.
4. Write the number of samples to transfer.
5. Write the page register.
6. Enable DMA on the channel you are using.

Programming the DM6425HR for DMA

Once you have set up the DMA controller, you must program the DM6425HR for DMA according to the following procedure:

1. Program Conversion and Trigger mode.
2. Program the DMA channel at **BA + 0x02h**.
3. Issue the start trigger.

Monitoring for DMA Done

There are two ways to monitor for DMA done. The easiest is to poll the DMA done bit in the DM6425HR status register (**BA + 0x02h**). While DMA is in progress, the bit is clear (0). When DMA is complete, the bit is set (1). The second way to check is to use the DMA done signal to generate an interrupt. An interrupt can immediately notify your program that DMA is done and any actions can be taken as needed.

Dual DMA Mode

The DM6425HR is capable of running in dual DMA mode. This is useful for acquiring large amounts of data at a high speed. In dual DMA mode, you must allocate two DMA buffers and program two DMA channels as described above. To program the DM6425HR, you must setup the first DMA channel at **BA + 0x02h**, bits 12 and 13 and set up the second DMA channel at **BA + 0x02h**, bits 14 and 15. In this mode, DMA will start and use the first channel and buffer you have set up. When the DMA done for this channel is received, the board will automatically switch to the second channel and buffer. While the board is filling the second buffer, you can empty the first buffer or reprogram the first channel to point to a different buffer. This allows you to stream large quantities of data to memory with very small amounts of software overhead.

Common DMA Problems

- Make sure that your buffer is large enough to hold all of the data you program the DMA controller to transfer.
- Check to be sure that your buffer does not straddle a page boundary.
- Remember that the value for the number of samples for the DMA controller to transfer is equal to (the number of samples – 1).
- If you terminate sampling before the DMA controller has transferred the number of bytes it was programmed for, be sure to disable DMA by setting the mask bit in the mask register.
- If you are in dual DMA mode, be sure to clear the DMA done bit after each DMA cycle is complete.

Chapter 7 Interrupts

This chapter explains software selectable interrupts, digital interrupts, and basic interrupt programming techniques.

The DM6425HR has two completely independent interrupt circuits that can generate interrupts on IRQ channels 3, 5, 9, 10, 11, 12, or 15. By using these two circuits, complex data acquisition systems can be configured.

Software Selectable Interrupt Sources

Each interrupt circuit on the DM6425HR has 16 software selectable interrupt sources that can be programmed in bits 0 through 4 and bits 8 through 12 of the Interrupt Register at **BA + 0x08h**, as described and shown below.

Software Selectable Interrupt Channel

Each interrupt circuit on the DM6425HR has 7 software selectable interrupt channels that can be programmed in bits 5 through 7 and bits 13 through 15 of the Interrupt Register at **BA + 0x08h**. The interrupt output is driven by an open collector device which is turned off when the IRQ channel is set to disable. At power up or reset, this register is set to all zeroes.

Advanced Digital Interrupts

The bit programmable digital I/O circuitry supports two Advanced Digital Interrupt modes, event mode or match mode. These modes are used to monitor input lines for state changes. The mode is selected at **BA + 0x1Eh**, bit 3 and enabled at **BA + 0x1Eh**, bit 4.

Event Mode

When enabled, this mode samples the Port 0 input lines at a specified clock rate (using the 8 MHz system clock or a programmable clock in User TC Counter 1), looking for a change in state in any one of the eight bits. When a change of state occurs, an interrupt is generated and the input pattern is latched into the Compare Register. You can read the contents of this register at **BA + 0x1Ch** to see which bit caused the interrupt to occur. Bits can be masked and their state changes ignored by programming the Mask Register with the mask at **BA + 0x1Ch**.

Match Mode

When enabled, this mode samples the Port 0 input lines at a specified clock rate (using the 8 MHz system clock or a programmable clock in User TC Counter 1) and compares all input states to the value programmed in the Compare Register at **BA + 0x1Ch**. When the states of all of the lines match the value in the Compare Register, an interrupt is generated. Bits can be masked and their states ignored by programming the Mask Register with the mask at **BA + 0x1Ch**.

Sampling Digital Lines for Change of State

In the Advanced Digital Interrupt modes, the digital lines are sampled at a rate set by the 8 MHz system clock or the clock programmed in User TC Counter 1. With each clock pulse, the digital circuitry looks at the state of the next Port 0 bits. To provide noise rejection and prevent erroneous interrupt generation because of noise spikes on the digital lines, a change in the state of any bit must be seen for two edges of a clock pulse to be recognized by the circuit. Figure 30 shows a diagram of this circuit.

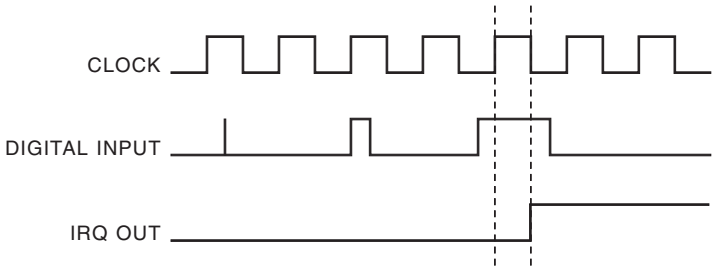


Figure 30 Digital Interrupt Timing Diagram

Basic Programming For Interrupt Handling

What Is an Interrupt?

An interrupt is an event that causes the processor in your computer to temporarily halt its current process and execute another routine. Upon completion of the new routine, control is returned to the original routine at the point where its execution was interrupted.

Interrupts are very handy for dealing with asynchronous events (events that occur at less than regular intervals). Keyboard activity is a good example; your computer cannot predict when you might press a key and it would be a waste of processor time for it to do nothing while waiting for a keystroke to occur. Thus, the interrupt scheme is used and the processor proceeds with other tasks. Then, when a keystroke does occur, the keyboard "interrupts" the processor, and the processor gets the keyboard data, places it in memory, and then returns to what it was doing before it was interrupted. Other common devices that use interrupts are modems, disk drives, and mice.

Your DM6425HR can interrupt the processor when a variety of conditions are met, such as DMA done, timer countdown finished, end-of-convert, and external trigger. By using these interrupts, you can write software that effectively deals with real world events.

Interrupt Request Lines

To allow different peripheral devices to generate interrupts on the same computer, the AT bus has 16 different interrupt request (IRQ) lines. A transition from low to high on one of these lines generates an interrupt request which is handled by one of the AT's two interrupt control chips. One chip handles IRQ0 through IRQ7 and the other chip handles IRQ8 through IRQ15. The controller which handles IRQ8–IRQ15 is chained to the first controller through the IRQ2 line. When an IRQ line is brought high, the interrupt controllers check to see if interrupts are to be acknowledged from that IRQ and, if another interrupt is already in progress, they decide if the new request should supersede the one in progress or if it has to wait until the one in progress is done. This prioritizing allows an interrupt to be interrupted if the second request has a higher priority. The priority level is determined by the number of the IRQ. Because of the configuration of the two controllers, with one chained to the other through IRQ2, the priority scheme is a little unusual. IRQ0 has the highest priority, IRQ1 is second-highest, then priority jumps to IRQ8, IRQ9, IRQ10, IRQ11, IRQ12, IRQ13, IRQ14, and IRQ15, and then following IRQ15, it jumps back to IRQ3, IRQ4, IRQ5, IRQ6, and finally, the lowest priority, IRQ7. This sequence makes sense if you consider that the controller that handles IRQ8–IRQ15 is routed through IRQ2.

8259 Programmable Interrupt Controllers

The chips responsible for handling interrupt requests in the PC are the 8259 Programmable Interrupt Controllers. The 8259 that handles IRQ0–IRQ7 is referred to as 8259A, and the 8259 that handles IRQ8–IRQ15 is referred to as 8259B. To use interrupts, you need to know how to read and set the 8259 interrupt mask registers (IMR) and how to send the end-of-interrupt (EOI) command to the 8259s.

Interrupt Mask Registers (IMR)

Each bit in the interrupt mask register (IMR) contains the mask status of an IRQ line; in 8259A, bit 0 is for IRQ0, bit 1 is for IRQ1, and so on, while in 8259B, bit 0 is for IRQ8, bit 1 is for IRQ9, and so on. If a bit is set (equal to 1), then the corresponding IRQ is masked and it will not generate an interrupt. If a bit is clear (equal to 0), then the corresponding IRQ is unmasked and can generate interrupts. The IMR for IRQ0–IRQ7 is programmed through port 21H, and the IMR for IRQ8–IRQ15 is programmed through port A1H.

IRQ7	IRQ6	IRQ5	IRQ4	IRQ3	IRQ2	IRQ1	IRQ0	I/O Port 21h
IRQ15	IRQ14	IRQ13	IRQ12	IRQ11	IRQ10	IRQ9	IRQ8	I/O Port A1h

For all bits
0 = IRQ unmasked (enabled)
1 = IRQ masked (disabled)

End-of-Interrupt (EOI) Command

After an interrupt service routine is complete, the appropriate 8259 interrupt controller must be notified. When using IRQ0–IRQ7, this is done by writing the value 20H to I/O port 20H only; when using IRQ8–IRQ15, you must write the value 20H to I/O ports 20H and A0H.

What Happens When an Interrupt Occurs?

Understanding the sequence of events when an interrupt is triggered is necessary to properly write software interrupt handlers. When an interrupt request line is driven high by a peripheral device (such as the DM6425HR), the interrupt controllers check to see if interrupts are enabled for that IRQ, and then checks to see if other interrupts are active or requested and determine which interrupt has priority. The interrupt controllers then interrupt the processor. The current code segment (CS), instruction pointer (IP), and flags are pushed on the stack for storage, and a new CS and IP are loaded from a table that exists in the lowest 1024 bytes of memory. This table is referred to as the interrupt vector table and each entry is called an interrupt vector. Once the new CS and IP are loaded from the interrupt vector table, the processor begins executing the code located at CS:IP. When the interrupt routine is completed, the CS, IP, and flags that were pushed on the stack when the interrupt occurred are now popped from the stack and execution resumes from the point where it was interrupted.

Using Interrupts in Your Programs

Adding interrupts to your software is not as difficult as it may seem, and what they add in terms of performance is often worth the effort. Note, however, that although it is not that hard to use interrupts, the smallest mistake will often lead to a system hang that requires a reboot. This can be both frustrating and time-consuming. But, after a few tries, you'll get the bugs worked out and enjoy the benefits of properly executed interrupts. In addition to reading the following paragraphs, study the example programs included on your dataModule program disk for a better understanding of interrupt program development.

Writing an Interrupt Service Routine (ISR)

The first step in adding interrupts to your software is to write the interrupt service routine (ISR). This is the routine that will automatically be executed each time an interrupt request occurs on the specified IRQ. An ISR is different than standard routines that you write. First, on entrance, the processor registers should be pushed onto the stack BEFORE you do anything else. Second, just before exiting your ISR, you must write an end-of-interrupt command to the 8259 controller(s). Since 8259B generates a request on IRQ2 which is handled by 8259A, an EOI must be sent to both 8259A and 8259B for IRQ8–IRQ15. Finally, when exiting the ISR, in addition to popping all the registers you pushed on entrance, you must use the IRET instruction and not a plain RET. The IRET automatically pops the flags, CS, and IP that were pushed when the interrupt was called.

If you find yourself intimidated by these requirements, take heart. Most Pascal and C compilers allow you to identify a procedure (function) as an interrupt type and will automatically add these instructions to your ISR, with one important exception: most compilers do not automatically add the end-of-interrupt command to the procedure; you must do this yourself. Other than this and the few exceptions discussed below, you can write your ISR just like any other routine. It can call other functions and procedures in your program and it can access global data. If you are writing your first ISR, we recommend that you stick to the basics; just something that will convince you that it works, such as incrementing a global variable.

Note *If you are writing an ISR using assembly language, you are responsible for pushing and popping registers and using IRET instead of RET.*

There are a few cautions you must consider when writing your ISR. The most important is, do not use any DOS functions or routines that call DOS functions from within an ISR. DOS is not reentrant; that is, a DOS function cannot call itself. In typical programming, this will not happen because of the way DOS is written. But what about when using interrupts? Then, you could have a situation such as this in your program. If DOS function X is being executed when an interrupt occurs and the interrupt routine makes a call to DOS function X, then function X is essentially being called while it is already active. Such a reentrancy attempt spells disaster because DOS functions are not written to support it. This is a complex concept and you do not need to understand it. Just make sure that you do not call any DOS functions from within your ISR. The one wrinkle is that, unfortunately, it is not obvious which library routines included with your compiler use DOS functions. A rule of thumb is that routines which write to the screen, or check the status of or read the keyboard, and any disk I/O routines use DOS and should be avoided in your ISR.

The same problem of reentrancy exists for many floating point emulators as well, meaning you may have to avoid floating point (real) math in your ISR.

Note that the problem of reentrancy exists, no matter what programming language you are using. Even if you are writing your ISR in assembly language, DOS and many floating point emulators are not reentrant. Of course, there are ways around this problem, such as those which involve checking to see if any DOS functions are currently active when your ISR is called, but such solutions are well beyond the scope of this discussion.

The second major concern when writing your ISR is to make it as short as possible in terms of execution time. Spending long periods of time in your ISR may mean that other important interrupts are being ignored. Also, if you spend too long in your ISR, it may be called again before you have completed handling the first run. This often leads to a hang that requires a reboot.

Your ISR should have this structure:

- Push any processor registers used in your ISR. Most C and Pascal interrupt routines automatically do this for you.
- Put the body of your routine here.
- Issue the EOI command to the 8259 interrupt controller by writing 20H to port 20H and port A0H (if you are using IRQ8–IRQ15).
- Pop all registers pushed on entrance. Most C and Pascal interrupt routines automatically do this for you.

The following C and Pascal examples show what the shell of your ISR should be like:

In C:

```
void interrupt ISR(void)
{
   /* Your code goes here. Do not use any DOS functions! */
   outportb(0x20, 0x20);/* Send EOI command to 8259A (for all IRQs)*/
   outportb(0x20, 0xA0);/* Send EOI command to 8259B (if using IRQ8-15) */
}
```

In Pascal:

```
Procedure ISR; Interrupt;
begin
   { Your code goes here. Do not use any DOS functions! }
   Port[$20] := $20;   { Send EOI command to 8259A (for all IRQs) }
   Port[$A0] := $20;   { Send EOI command to 8259B (if using IRQ8-15) }
end;
```

Saving the Startup Interrupt Mask Register (IMR) and Interrupt Vector

The next step after writing the ISR is to save the startup state of the interrupt mask register and the interrupt vector that you will be using. The IMR for IRQ0–IRQ7 is located at I/O port 21h; the IMR for IRQ8–IRQ15 is located at I/O port A1h. The interrupt vector you will be using is located in the interrupt vector table which is simply an array of 256 four-byte pointers and is located in the first 1024 bytes of memory (Segment = 0, Offset = 0). You can read this value directly, but it is a better practice to use DOS function 35h (get interrupt vector). Most C and Pascal compilers provide a library routine for reading the value of a vector. The vectors for IRQ0–IRQ7 are vectors 8 through 15, where IRQ0 uses vector 8, IRQ1 uses vector 9, and so on. The vectors for IRQ8–IRQ15 are vectors 70h through 77h, where IRQ8 uses vector 70h, IRQ9 uses vector 71h, and so on. Thus, if the DM6425HR will be using IRQ15, you should save the value of interrupt vector 77h.

Before you install your ISR, temporarily mask out the IRQ you will be using. This prevents the IRQ from requesting an interrupt while you are installing and initializing your ISR. To mask the IRQ, read in the current IMR at I/O port 21h for IRQ0–IRQ7, or at I/O port A1h for IRQ8–IRQ15 and set the bit that corresponds to your IRQ (remember, setting a bit disables interrupts on that IRQ while clearing a bit enables them). The IMR on 8259A is arranged so that bit 0 is for IRQ0, bit 1 is for IRQ1, and so on. The IMR on 8259B is arranged so that bit 0 is for IRQ8, bit 1 is for IRQ9, and so on. See the paragraph entitled Interrupt Mask Register (IMR) earlier in this chapter for help in determining your IRQ's bit. After setting the bit, write the new value to I/O port 21h (IRQ0–IRQ7) or I/O port A1h (IRQ8–IRQ15).

With the startup IMR saved and the interrupts on your IRQ temporarily disabled, you can assign the interrupt vector to point to your ISR. Again, you can overwrite the appropriate entry in the vector table with a direct memory write, but this is a bad practice. Instead, use either DOS function 25h (set interrupt vector) or, if your compiler provides it, the library routine for setting an interrupt vector. Remember that vectors 8–15 are for IRQ0–IRQ7 and vectors 70h–77h are for IRQ8–IRQ15.

If you need to program the source of your interrupts, do that next. For example, if you are using the programmable interval timer to generate interrupts, you must program it to run in the proper mode and at the proper rate.

Finally, clear the bit in the IMR for the IRQ you are using. This enables interrupts on the IRQ.

Restoring the Startup IMR and Interrupt Vector

Before exiting your program, you must restore the interrupt mask register and interrupt vectors to the state they were in before your program started. To restore the IMR, write the value that was saved when your program started to I/O port 21h for IRQ0–IRQ7 or I/O port A1h for IRQ8–IRQ15. Restore the interrupt vector that was saved at startup with either DOS function 25h (set interrupt vector), or use the library routine supplied with your compiler. Performing these two steps will guarantee that the interrupt status of your computer is the same after running your program as it was before your program started running.

Common Interrupt Mistakes

- Remember that hardware interrupts are numbered 8 through 15 for IRQ0–IRQ7 and 70h through 77h for IRQ8–IRQ15.
- The most common mistake when writing an ISR is forgetting to issue the EOI command to the appropriate 8259 interrupt controller before exiting the ISR.
- Remember to clear the appropriate IRQ circuit on the DM6425HR at **BA + 0x00h**.

Chapter 8 D/A Conversions

This chapter explains how to perform D/A conversions on the DM6425HR dataModule.

The two D/A converters can be individually programmed to convert 12-bit digital words into a voltage in the range of ± 5, 0 to +5, ±10, or 0 to +10 V. DAC1 is programmed by writing the 12-bit word to **BA + 0x0Ch**. DAC2 is identical, with the 12-bit word written to **BA + 0x0Eh**. The outputs of both DACs are updated independently when you write the 12-bit word. The 12-bit information is right-justified in the 16-bit word.

The following table lists the key digital codes and corresponding output voltages for the D/A converters.

	Ideal Output Voltage (mV)			
D/A Bit Weight	**–5 to +5 V**	**0 to +5 V**	**–10 to +10 V**	**0 to +10 V**
4095 (full-scale)	+4997.56	+4998.78	9995.12	+9997.56
2048	0000.00	+2500.00	0000.00	+5000.00
1024	–2500.00	+1250.00	–5000.00	+2500.00
512	–3750.00	+625.00	–7500.00	+1250.00
256	–4375.00	+312.50	–8750.00	+625.00
128	–4687.50	+156.25	–9375.00	+312.50
64	–4843.75	+78.13	–9687.50	+156.25
32	–4921.88	+39.06	–9843.75	+78.13
16	–4960.94	+19.53	–9921.88	+39.06
8	–4980.47	+9.77	–9960.94	+19.53
4	–4990.23	+4.88	–9980.47	+9.77
2	–4995.12	+2.44	–9990.23	+4.88
1	–4997.56	+1.22	–9995.12	+2.44
0	–5000.00	0.00	–10.00	0.00

Chapter 9 Timer/Counters

This chapter explains the two 8254 timer/counter circuits on the DM6425HR dataModule.

Two 8254 programmable interval timers, Clock TC and User TC, each provide three 16-bit, 8-MHz timers for timing and counting functions such as frequency measurement, event counting, and interrupts. Two of the timers in the Clock TC (**U11**) are cascaded and used for the onboard pacer clock, described in Chapter 5, *A/D Conversions*. The third timer is the burst clock, also discussed in Chapter 5. Figure 31 shows the Clock TC circuitry.

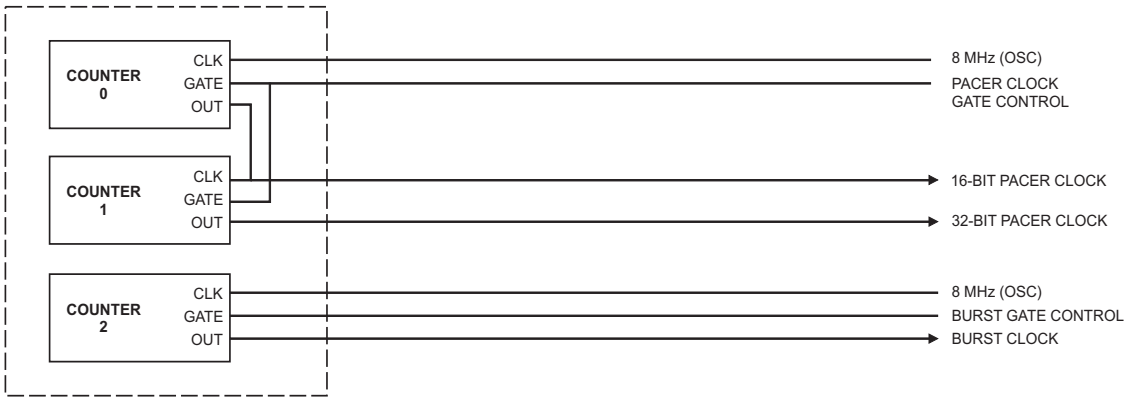


Figure 31 Clock TC Circuitry

Counters 0 and 1 on the User TC (**U12**) are unused and available for your use. The third timer, Counter 2, forms the 16-bit sample counter described in Chapter 5, *A/D Conversions*. Figure 32 shows the User TC circuitry.

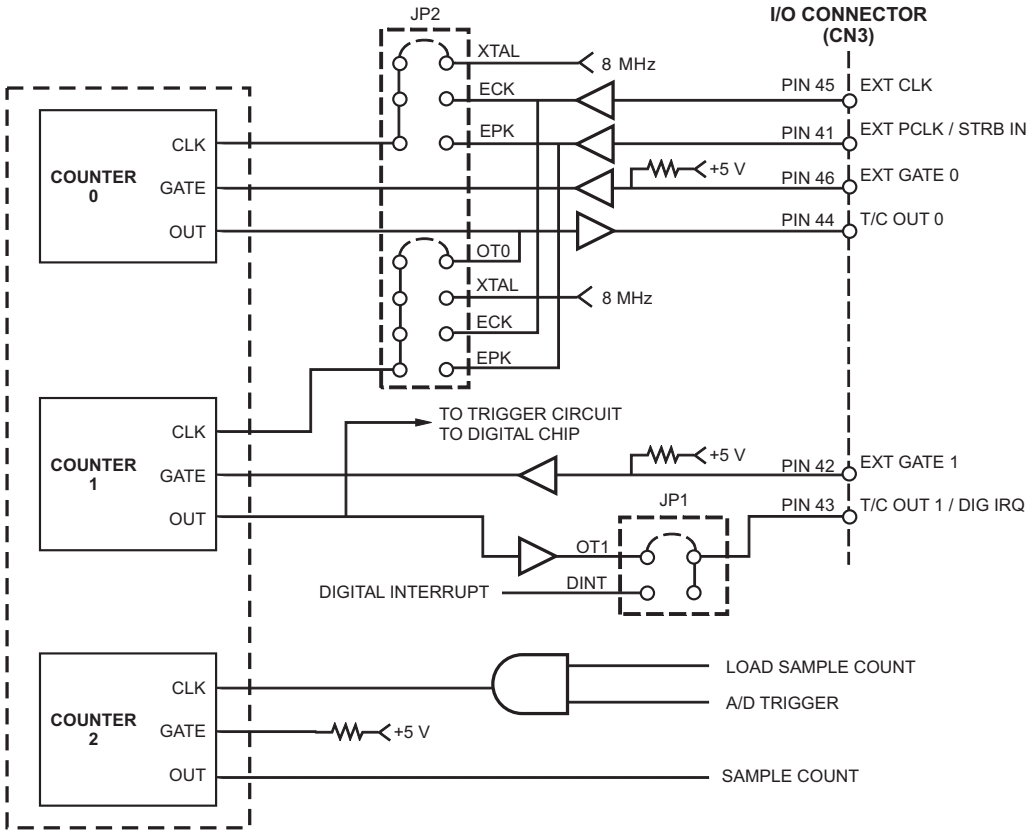


Figure 32 User TC Circuitry

Each timer/counter has two inputs, CLK in and GATE in, and one output, timer/counter OUT. They can be programmed as binary or BCD down counters by writing the appropriate data to the command word, as described in the I/O map discussion in Chapter 4.

The output from User TC Counter 1 is available at the T/C OUT 1 pin (**CN3**, pin 43) on the I/O connector where it can be used for interrupt generation, as an A/D trigger, or for counting functions. The output from User TC Counter 0 is connected to the T/C OUT 0 pin (**CN3**, pin 44) on the I/O connector where it can be used for interrupt generation or for timing functions.

The timers can be programmed to operate in one of six modes, depending on your application. The following paragraphs briefly describe each mode.

Mode 0, Event Counter (Interrupt on Terminal Count). This mode is typically used for event counting. While the timer/counter counts down, the output is low, and when the count is complete, it goes high. The output stays high until a new Mode 0 control word is written to the timer/counter.

Mode 1, Hardware-Retriggerable One-Shot. The output is initially high and goes low on the clock pulse following a trigger to begin the one-shot pulse. The output remains low until the count reaches 0, and then goes high and remains high until the clock pulse after the next trigger.

Mode 2, Rate Generator. This mode functions like a divide-by-N counter and is typically used to generate a real-time clock interrupt. The output is initially high, and when the count decrements to 1, the output goes low for one clock pulse. The output then goes high again, the timer/counter reloads the initial count, and the process is repeated. This sequence continues indefinitely.

Mode 3, Square Wave Mode. Similar to Mode 2 except for the duty cycle output, this mode is typically used for baud rate generation. The output is initially high, and when the count decrements to one-half its initial count, the output goes low for the remainder of the count. The timer/counter reloads and the output goes high again. This process repeats indefinitely.

Mode 4, Software-Triggered Strobe. The output is initially high. When the initial count expires, the output goes low for one clock pulse and then goes high again. Counting is "triggered" by writing the initial count.

Mode 5, Hardware Triggered Strobe (Retriggerable). The output is initially high. Counting is triggered by the rising edge of the gate input. When the initial count has expired, the output goes low for one clock pulse and then goes high again.

Chapter 10 Digital I/O

This chapter explains the bit programmable and port programmable digital I/O circuitry on the DM6425HR dataModule.

The DM6425HR has 32 buffered TTL/CMOS digital I/O lines, which can be used to transfer data between the computer and external devices. These digital I/O lines are comprised of four independent ports of 8 bits each.

Port 0, Bit Programmable Digital I/O

The eight Port 0 digital lines are individually set for input or output by writing to the Port 0 Direction Register at **BA + 0x1Ch**. The input lines are read and the output lines are written at **BA + 0x18h**.

Direction Register:

For all bits:
0 = input
1 = output

D7	D6	D5	D4	D3	D2	D1	D0
P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0

Advanced Digital Interrupts: Mask and Compare Registers

The Port 0 bits support two Advanced Digital Interrupt modes. An interrupt can be generated when the data read at the port matches the value loaded into the Compare Register. This is called a match interrupt. Or, an interrupt can be generated whenever any bit changes state. This is an event interrupt. For either interrupt, bits can be masked by setting the corresponding bit in the Mask Register high. In a digital interrupt mode, this masks out selected bits when monitoring the bit pattern for a match or event. In normal operation where the Advanced Digital Interrupt mode is not activated, the Mask Register can be used to preserve a bit's state, regardless of the digital data written to Port 0.

When using event interrupts, you can determine which bit caused an event interrupt to occur by reading the contents latched into the Compare Register.

Port 1, Port Programmable Digital I/O

The direction of the eight Port 1 digital lines is programmed at **BA + 0x1Eh**, bit 2. These lines are configured as all inputs or all outputs, with their states read and written at **BA + 0x1Ah**.

Port 2, Bit Programmable Digital I/O

The eight Port 2 digital lines are individually set for input or output by writing to the Port 2 Direction Register at **BA + 0x41Ch**. The input lines are read and the output lines are written at **BA + 0x418h**.

Direction Register:

For all bits:
0 = input
1 = output

D7	D6	D5	D4	D3	D2	D1	D0
P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0

Advanced Digital Interrupts: Mask and Compare Registers

The Port 2 bits support two Advanced Digital Interrupt modes. An interrupt can be generated when the data read at the port matches the value loaded into the Compare Register. This is called a match interrupt. Or, an interrupt can be generated whenever any bit changes state. This is an event interrupt. For either interrupt, bits can be masked by setting the corresponding bit in the Mask Register high. In a digital interrupt mode, this masks out selected bits when monitoring the bit pattern for a match or event. In normal operation where the Advanced Digital Interrupt mode is not activated, the Mask Register can be used to preserve a bit's state, regardless of the digital data written to Port 2.

When using event interrupts, you can determine which bit caused an event interrupt to occur by reading the contents latched into the Compare Register.

Port 3, Port Programmable Digital I/O

The direction of the eight Port 3 digital lines is programmed at **BA + 0x41Eh**, bit 2. These lines are configured as all inputs or all outputs, with their states read and written at **BA + 0x41Ah**.

Resetting the Digital Circuitry

When issuing a digital chip clear (Port 0: **BA + 0x1Eh**, bits 1 and 0 = 00 followed by a write to **BA + 0x1Ch**; Port 2: **BA + 0x41Eh**, bits 1 and 0 = 00 followed by a write to **BA + 0x41Ch**), clear board (**BA + 0x00h**), or reset command, all of the digital I/O lines are set up as inputs.

Strobing Data into Port 0

When not in an Advanced Digital Interrupt mode, external data can be strobed into Port 0 by connecting a trigger pulse through the STRB IN pin at **CN3**, pin 41. This data can be read from the Compare Register at **BA + 0x1Ch**.

High-Speed Digital Input

The DM6425HR has a 1024 sample buffer connected to the Port 0 lines, which can be used to collect high-speed digital data. The controls for this FIFO are accessed by the registers at **BA + 0x0Ah**. Data is clocked into the FIFO on the rising edge of the clock which can be set to 5 different sources. The maximum clock rate should not exceed 1 MHz. For high-speed inputs, set up the board to generate an interrupt when the digital input FIFO is half full and use the REP INS (Repeat Input String) command to read the data.

Note *Be sure to disable the input clock at* ***BA + 0x0Ah*** *before you clear the digital input FIFO.*

Appendix A Example Programs

This appendix discusses the example programs included with the DM6425HR dataModule.

Controlling the DM6425HR via Software

RTD Provided Software

The companion CD that shipped with your DM6425HR includes drivers and example programs for various operating systems. Newer versions of the software may also be available from RTD's website.

The drivers provide an easy-to-use programming interface to control the DM6425HR. The example programs demonstrate the DM6425HR's features and how to use the drivers. The companion CD also includes source code for sample programs, which you may use as a template to develop your own software.

Consult the documentation provided with each driver for information on how to install and use it.

Direct Register Control

RTD recommends using the provided drivers and examples for your operating system to interface with the DM6425HR. However, if you are using an unsupported operating system, you may need to access the DM6425HR's registers directly.

Note *Many modern operating systems have protection features that prevent user applications from directly manipulating device registers. Under such operating systems, it will be necessary to develop a driver that controls the DM6425HR on behalf of your application.*

Usually, manipulating the registers of the DM6425HR is done via I/O port read/write functions. Under DOS, this is typically done via the inp() and outp() functions. The following example (written in C) illustrates direct register access under DOS:

```
#define BASE_ADDRESS 0x300
unsigned char register_value;

// Read from digital inputs 1-8 at BA+2
register_value = inp(BASE_ADDRESS + 2);
printf("Value of digital inputs 1-8 is: %x", register_value);

// Write a value to the digital output at BA + 0
// Set the outputs to all '1'
register_value = 0xFF;
outp(BASE_ADDRESS + 0, register_value);
```

Many operating systems have their own equivalent of the inp() and outp() functions. Consult the operating system's documentation for information on how to properly program I/O ports.

Special Considerations

When manipulating the registers of a board, you must be very careful to only affect the bits that need to be altered. Simply writing a value to a register can inadvertently change the state of a board. To manipulate specific bits in a register, you should use the bitwise AND/OR operators.

For example, if you wish to set bit 3 in a register, you should not simply write 0x08 to the register. Doing so will overwrite the value of all other bits in the register. A better approach is to do the following:

1. Read in the register.
2. Use the bitwise OR operator to set bit 3.
3. Write the modified value back to the register.

The above approach is illustrated in the following DOS example (written in C):

```
// Read the register into a variable
register_value = inp(REGISTER_ADDRESS);

// Set bit 3
register_value |= 0x08; // Binary 00001000

// Write the modified value to the register
outp(REGISTER_ADDRESS, register_value);
```

This example sets bit 3 without affecting any of the other bits in the register.

As another example, if you want to clear bit 3 of a register—without affecting the other bits—you can use the bitwise AND operator as shown in the following DOS example (written in C):

```
// Read the register into a variable
register_value = inp(REGISTER_ADDRESS);

// Clear bit 3
register_value &= 0xF7; // Binary 11110111

// Write the modified value back to the register
outp(REGISTER_ADDRESS, register_value);
```

The above example will clear bit 3 without affecting any other bits in the register.

Note *Be very careful to preserve the value of any bits marked as Reserved. The function of a reserved bit may change in future revisions of the DM6425HR. If your software manipulates a reserved bit, it may not be compatible with future board revisions.*

Appendix B Calibration

This appendix describes how to calibrate the DM6425HR using the trimpots on the module. The trimpots calibrate the A/D converter gain and offset.

The D/A converter does not need to be calibrated. The offset and full-scale performance of the module's A/D converter is factory-calibrated. Any time you suspect inaccurate readings, you can check the accuracy of your conversions using the procedure in this appendix, and make adjustments as necessary. Using the 6425DIAG diagnostics program is a convenient way to monitor conversions while you calibrate the module.

Calibration is done with the module installed in your system. Power up the system and let the DM6425HR circuitry stabilize for 15 minutes before calibration.

Required Equipment

The following equipment is required for calibration:

- Precision voltage source: –10 to +10 V
- Digital voltmeter: 5 1/2 digits
- Small screwdriver (for trimpot adjustment)

Figure 33 shows the DM6425HR layout with the trimpots located along the top edge.

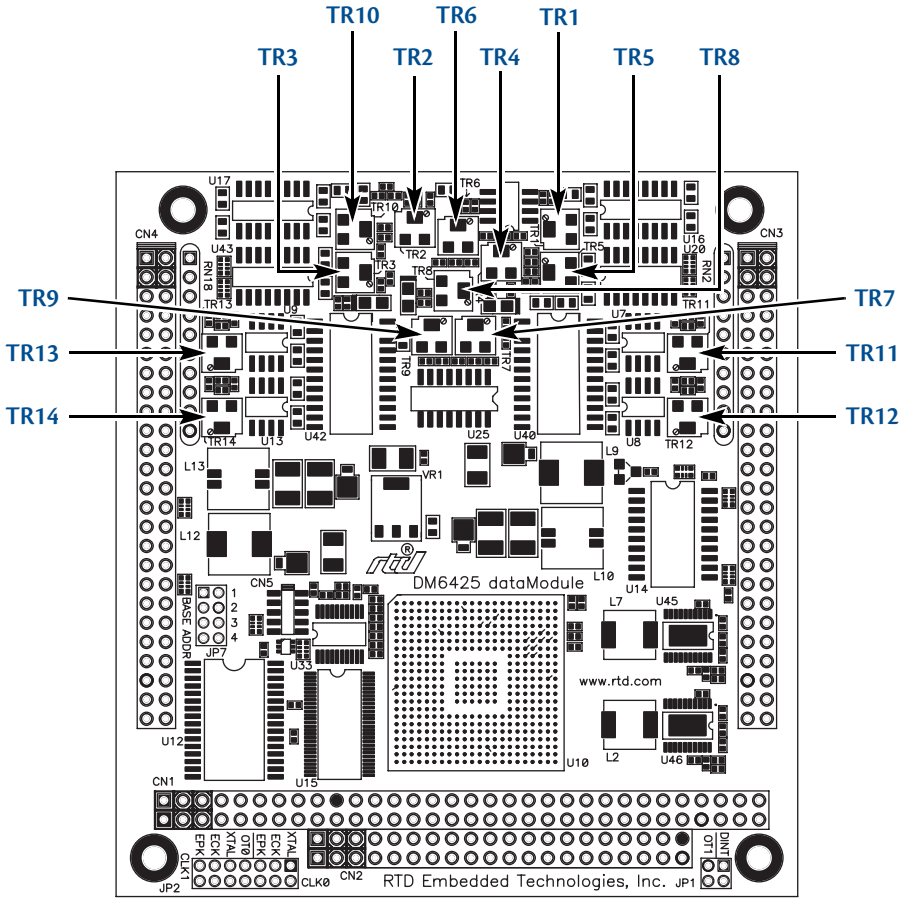


Figure 33 DM6425HR Trimpots

A/D Calibration

Two procedures are used to calibrate the A/D converter for all input voltage ranges. The first procedure calibrates the converter for the bipolar ranges (±5, ±10 V), and the second procedure calibrates the unipolar range (0 to +10 V). Table 9 shows the ideal input voltage for each bit weight for the bipolar ranges, and Table 10 shows the ideal voltage for each bit weight for the unipolar ranges.

Bipolar Calibration

Bipolar Range Adjustment: –5 to +5 V

Two adjustments are made to calibrate the A/D converter for the bipolar range of –5 to +5 V. One is the offset adjustment, and the other is the full scale, or gain, adjustment. Trimpot **TR4** is used to make the offset adjustment, and trimpot **TR5** is used for gain adjustment. Before making these adjustments, make sure that the board is programmed for a range of ±5 V.

Use analog input channel 1 and set it for a gain of 1 while calibrating the board. Connect your precision voltage source to channel 1. Set the voltage source to –1.22070 mV, start a conversion, and read the resulting data. Adjust trimpot **TR4** until the reading flickers between the values listed in the table below. Next, set the voltage to –4.99878 V, and repeat the procedure, this time adjusting **TR5** until the data flickers between the values in the table below.

Data Values for Calibrating Bipolar 10 V Range (–5 to +5 V)		
	Offset (TR4) **Input Voltage = –1.22 mV**	**Converter Gain (TR5)** **Input Voltage = –4.99878 mV**
A/D Converted Data	0000 0000 0000	1000 0000 0000
	1111 1111 1111	1000 0000 0001

Bipolar Range Adjustment: –10 to +10 V

To adjust the bipolar 20 V range (–10 to +10 V), program the board for ± 10 V input range. Then, set the input voltage to +5.0000 V and adjust **TR2** until the output matches the data in the table below.

Data Values for Calibrating Bipolar 20 V Range (–10 to +10 V)	
	TR2 **Input Voltage = +5.0000 V**
A/D Converted Data	0100 0000 0000

Below is a table listing the ideal input voltage for each bit weight for the bipolar ranges.

Table 9 A/D Converter Bit Weights, Bipolar

		Ideal Output Voltage (mV)	
Sign	A/D Bit Weight	−5 to +5 V	−10 to +10 V
1	1111 1111 1111	−2.44	−4.88
1	1000 0000 0000	−5000.00	−10000.00
0	0100 0000 0000	+2500.00	+5000.00
0	0010 0000 0000	+1250.00	+2500.00
0	0001 0000 0000	+625.00	+1250.00
0	0000 1000 0000	+312.50	+625.00
0	0000 0100 0000	+156.25	+312.50
0	0000 0010 0000	+78.13	+156.25
0	0000 0001 0000	+39.06	+78.13
0	0000 0000 1000	+19.53	+39.06
0	0000 0000 0100	+9.77	+19.53
0	0000 0000 0010	+4.88	+9.77
0	0000 0000 0001	+2.44	+4.88
0	0000 0000 0000	0.00	0.00

Unipolar Calibration

One adjustment is made to calibrate the A/D converter for the unipolar range of 0 to +10 V. Trimpot **TR6** is used to make the offset adjustment. This calibration procedure is performed with the module programmed for a 0 to +10 V input range. Before making these adjustments, make sure that the module is programmed properly and has been calibrated for the bipolar ranges.

Use analog input channel 1 and set it for a gain of 1 while calibrating the board. Connect your precision voltage source to channel 1. Set the voltage source to +1.22070 mV, start a conversion, and read the resulting data. Adjust trimpot **TR6** until the data flickers between the values listed in the table below.

Data Values for Calibrating Unipolar 10 V Range (0 to +10 V)	
	Offset (TR6) Input Voltage = +1.22070 mV
A/D Converted Data	0000 0000 0000 0000 0000 0001

Below is a table listing the ideal input voltage for each bit weight for the unipolar range.

Table 10 A/D Converter Bit Weights, Unipolar

		Ideal Output Voltage (mV)
Sign	A/D Bit Weight	0 to +10 V
0	1111 1111 1111	+9997.60
0	1000 0000 0000	+5000.00
0	0100 0000 0000	+2500.00
0	0010 0000 0000	+1250.00
0	0001 0000 0000	+625.00
0	0000 1000 0000	+312.50
0	0000 0100 0000	+156.25
0	0000 0010 0000	+78.13
0	0000 0001 0000	+39.06
0	0000 0000 1000	+19.53
0	0000 0000 0100	+9.77
0	0000 0000 0010	+4.88
0	0000 0000 0001	+2.44
0	0000 0000 0000	0.00

Gain Adjustment

Should you find it necessary to check any of the programmable gain settings, the following table will show the proper trimpot to adjust.

Trimpots for Calibrating Gains	
Gain	Trimpot
x2	TR7
x4	TR8
x8	TR9

D/A Calibration

The D/A circuit requires no calibration. The table below provides, for your reference, a list of the input bits and their corresponding ideal output voltages for each of the three output ranges.

	Ideal Output Voltage (mV)			
D/A Bit Weight	**–5 to +5 V**	**0 to +5 V**	**–10 to +10 V**	**0 to +10 V**
4095 (full-scale)	+4997.56	+4998.78	9995.12	+9997.56
2048	0000.00	+2500.00	0000.00	+5000.00
1024	–2500.00	+1250.00	–5000.00	+2500.00
512	–3750.00	+625.00	–7500.00	+1250.00
256	–4375.00	+312.50	–8750.00	+625.00
128	–4687.50	+156.25	–9375.00	+312.50
64	–4843.75	+78.13	–9687.50	+156.25
32	–4921.88	+39.06	–9843.75	+78.13
16	–4960.94	+19.53	–9921.88	+39.06
8	–4980.47	+9.77	–9960.94	+19.53
4	–4990.23	+4.88	–9980.47	+9.77
2	–4995.12	+2.44	–9990.23	+4.88
1	–4997.56	+1.22	–9995.12	+2.44
0	–5000.00	0.00	10.00	0.00

Appendix C Specifications

This appendix provides the typical characteristics of the DM6425HR dataModule @ 25 °C.

Table 11 DM6425HR Specifications

Characteristic	Specification
Interface	
Jumper-selectablebase address, I/O mapped	
Software programmable interrupts and DMA channel	
Analog Input	
Up to 16 differential or 32 single-ended inputs, software selectable	
Input Impedance, each channel	>10 MΩ
Gains, software-selectable	1, 2, 4, and 8
Gain Error	0.05 dB, typ.; 0.1 dB, max.
Input Ranges, software selectable	±5, ±10, or 0 to +10 V
Overvoltage Protection	±12 VDC
Common Mode Input Voltage	±10 V, max.
Settling Time (gain = 1)	2 µs, max.
A/D Converter	
Type	Successive approximation
Resolution	12 bits (2.44 mV @ 10 V; 4.88 mV @ 20 V)
Linearity	±1 bit, typ.
Conversion Speed	2 µs, typ.
Module Throughput	500 kHz
Channel-Gain Table	
Size	1024 x 16 bits
Pacer Clock & Sample Counter	
Range (using onboard 8 MHz clock)	9 minutes to 2 µs
Sample Counter Maximum Count (1 cycle)	65,536
Digital I/O	
Number of Lines (32)	Port 0: 8 bit programmable lines Port 1: 1 byte programmable line Port 2: 8 bit programmable lines Port 3: 1 byte programmable line
Isource	–12 MA
Isink	24 mA
Sample Buffer	
FIFO Size (A/D)	1024 x 16 bits
FIFO Size (Digital Input Port 0)	1024 x 8 bits
D/A Converter	**AD7245**
Analog Outputs	4 channels
Resolution	12 bits
Output Ranges	± 5, 0 to +5, ±10, or 0 to +10 V
Relative Accuracy	±1 bit, max.
Full-Scale Accuracy	±5 bits, max.
Non-Linearity	±1 bit, max.
Settling Time	5 µs, max.

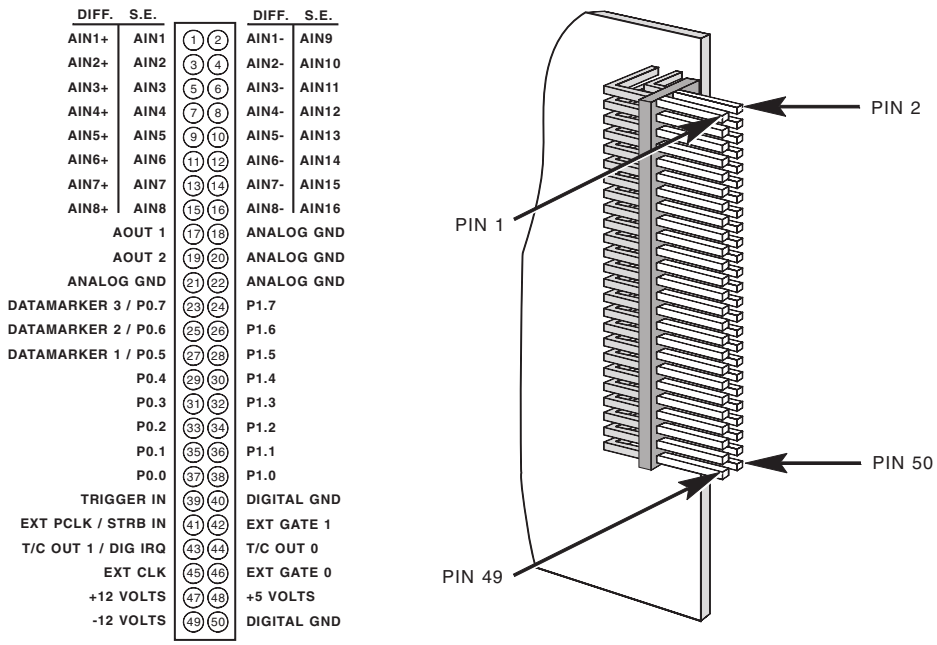
Table 11 DM6425HR Specifications (cont'd)

Characteristic	Specification
Timer/Counters	**CMOS 82C54**
Six 16-bit down counters	
Six programmable operating modes	
Counter Input Source	External clock (8 MHz, max.) or onboard 8 MHz clock
Counter Outputs	Available externally; used as PC interrupts
Counter Gate Source	External gate or always enabled
Miscellaneous Inputs/Outputs (PC bus-sourced)	
±5 V, ±12 V, ground	
Power Requirements	
+5 V, 4.0 W, typ.	
I/O Connectors (CN3/CN4)	
50-pin right angle headers	
Environmental	
Operating Temperature	–40 to +85°C
Storage Temperature	–55 to +125°C
Humidity	0 to 90%, noncondensing
Size	
3.55" L x 3.775" W x 0.6" H (90mm x 96mm x 16mm)	

Appendix D I/O Connector Pin Assignments

This appendix provides the pinouts for the dataModule I/O connectors, **CN3** and **CN4**.

CN3 (Right I/O Connector)



CN4 (Left I/O Connector)

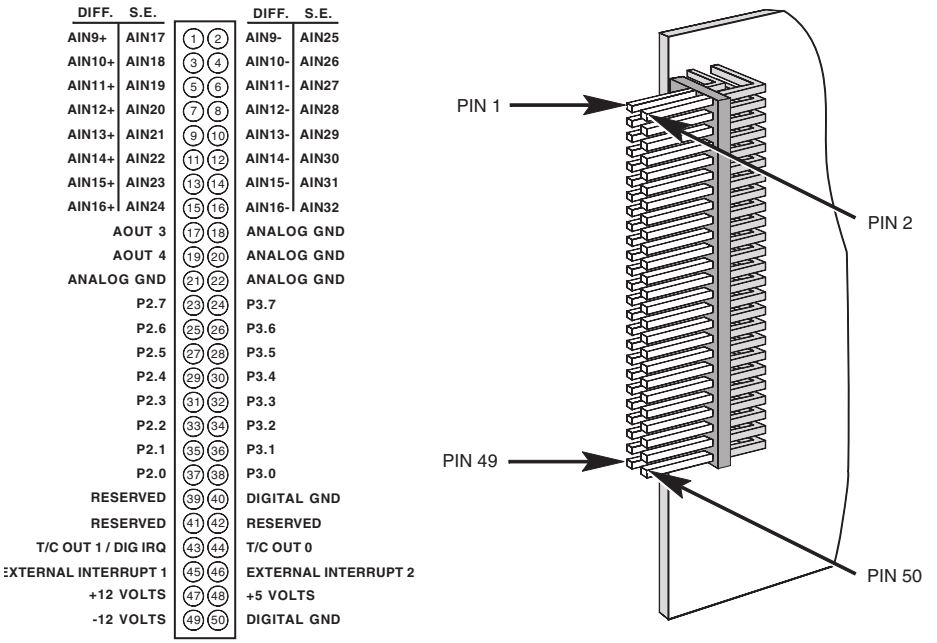


Figure 34 DM6425HR I/O Connectors

Note *+12 V at pin 47 and –12 V at pin 49 are available only if supplied by the computer bus.*

Mating Connector Part Numbers	
Manufacturer	**Part Number**
AMP	1-746094-0
3M	3425-7650

Appendix E IDAN™ Dimensions and Pinout

dataModules, like all other RTD PC/PCI-104 modules, can be packaged in Intelligent Data Acquisition Node (IDAN) frames, which are milled aluminum frames with integrated heat sinks and heat pipes for fanless operation. RTD modules installed in IDAN frames are called building blocks. IDAN building blocks maintain the simple but rugged PC/104 stacking concept. Each RTD module is mounted in its own IDAN frame and all I/O connections are brought to the walls of each frame using standard PC connectors. No connections are made from module to module internal to the system other than through the PC/104 and PC/104-*Plus* bus, enabling quick interchangeability and system expansion without hours of rewiring and board redesign.

The DM6425HR dataModule can also be purchased as part of a custom-built RTD HiDAN™ or HiDAN*plus*™ High Reliability Intelligent Data Acquisition Node. This appendix provides the dimensions and pinouts of the installed in an IDAN frame. Contact RTD for more information on high reliability IDAN, HiDAN, and HiDAN*plus* PC/PCI-104 systems.

IDAN—Adhering to the PC/104 stacking concept, IDAN allows you to build a customized system with any combination of RTD modules.

IDAN Heat Pipes—Advanced heat pipe technology maximizes heat transfer to heat sink fins.

HiDAN*plus*—Integrating the modularity of IDAN with the ruggedization of HiDAN, HiDANplus enables connectors on all system frames, with signals running between frames through a dedicated stack-through raceway.

IDAN-DM6425HR-62D

The IDAN-DM6425HR-62D is a DM6425HR dataModule packaged in an IDAN frame with a 62-pin D subminiature, front panel, I/O connector (female).

IDAN-DM6425HR-62D Dimensions and Connector

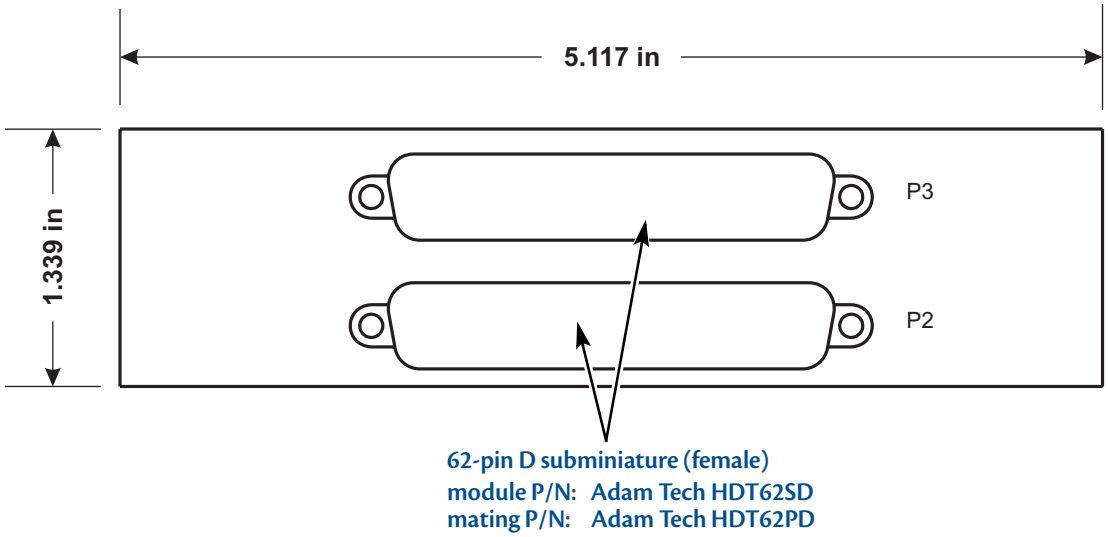


Figure 35 IDAN-DM6425HR-62D Connector

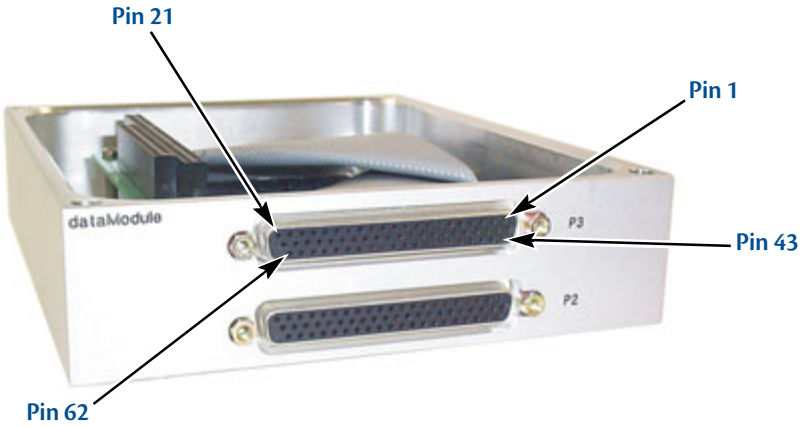


Figure 36 IDAN-DM6425HR-62D Connector Pins

CAUTION *Do not make connections to reserved pins!*

Note *Photographs are not to scale.*

IDAN-DM6425HR-62D External I/O Connections

Table 12 IDAN-DM6425HR-62D Connector

IDAN P2 Pin #			Signal	DM6425 CN3 Pin #	IDAN P3 Pin #			Signal	DM6425 CN4 Pin #
Row 1	Row 2	Row 3			Row 1	Row 2	Row 3		
1	—	—	AIN1+/AIN1	1	1	—	—	AIN9+/AIN17	1
—	22	—	AIN1–/AIN9	2	—	22	—	AIN9–/AIN25	2
—	—	43	AIN2+/AIN2	3	—	—	43	AIN10+/AIN18	3
2	—	—	AIN2–/AIN10	4	2	—	—	AIN10–/AIN26	4
—	23	—	AIN3+/AIN3	5	—	23	—	AIN11+/AIN19	5
—	—	44	AIN3–/AIN11	6	—	—	44	AIN11–/AIN27	6
3	—	—	AIN4+/AIN4	7	3	—	—	AIN12+/AIN20	7
—	24	—	AIN4–/AIN12	8	—	24	—	AIN12–/AIN28	8
—	—	45	AIN5+/AIN5	9	—	—	45	AIN13+/AIN21	9
4	—	—	AIN5–/AIN13	10	4	—	—	AIN13–/AIN29	10
—	25	—	AIN6+/AIN6	11	—	25	—	AIN14+/AIN22	11
—	—	46	AIN6–/AIN14	12	—	—	46	AIN14–/AIN30	12
5	—	—	AIN7+/AIN7	13	5	—	—	AIN15+/AIN23	13
—	26	—	AIN7–/AIN15	14	—	26	—	AIN15–/AIN31	14
—	—	47	AIN8+/AIN8	15	—	—	47	AIN16+/AIN24	15
6	—	—	AIN8–/AIN16	16	6	—	—	AIN16–/AIN32	16
—	27	—	AOUT 1	17	—	27	—	AOUT 3	17
—	—	48	ANALOG GND	18	—	—	48	ANALOG GND	18
7	—	—	AOUT 2	19	7	—	—	AOUT 4	19
—	28	—	ANALOG GND	20	—	28	—	ANALOG GND	20
—	—	49	ANALOG GND	21	—	—	49	ANALOG GND	21
8	—	—	ANALOG GND	22	8	—	—	ANALOG GND	22
—	29	—	P0.7/DATA MARKER 3	23	—	29	—	P2.7	23
—	—	50	P1.7	24	—	—	50	P3.7	24
9	—	—	P0.6/DATA MARKER 2	25	9	—	—	P2.6	25
—	30	—	P1.6	26	—	30	—	P3.6	26
—	—	51	P0.5/DATA MARKER 1	27	—	—	51	P2.5	27
10	—	—	P1.5	28	10	—	—	P3.5	28
—	31	—	P0.4	29	—	31	—	P2.4	29
—	—	52	P1.4	30	—	—	52	P3.4	30
11	—	—	P0.3	31	11	—	—	P2.3	31
—	32	—	P1.3	32	—	32	—	P3.3	32
—	—	53	P0.2	33	—	—	53	P2.2	33
12	—	—	P1.2	34	12	—	—	P3.2	34
—	33	—	P0.1	35	—	33	—	P2.1	35
—	—	54	P1.1	36	—	—	54	P3.1	36
13	—	—	P0.0	37	13	—	—	P2.0	37
—	34	—	P1.0	38	—	34	—	P3.0	38
—	—	55	TRIGGER IN	39	—	—	55	RESERVED	39
14	—	—	DIGITAL GND	40	14	—	—	DIGITAL GND	40
—	35	—	EXT PCLK/STRB IN	41	—	35	—	RESERVED	41
—	—	56	EXT GATE 1	42	—	—	56	RESERVED	42
15	—	—	T/C OUT 1/DIG IRQ	43	15	—	—	RESERVED	43
—	36	—	T/C OUT 0	44	—	36	—	RESERVED	44
—	—	57	EXT CLK	45	—	—	57	RESERVED	45
16	—	—	EXT GATE 0	46	16	—	—	RESERVED	46
—	37	—	+12 VOLTS	47	—	37	—	+12 VOLTS	47
—	—	58	+5 VOLTS	48	—	—	58	+5 VOLTS	48
17	—	—	–12 VOLTS	49	17	—	—	–12 VOLTS	49
—	38	—	DIGITAL GND	50	—	38	—	DIGITAL GND	50

IDAN-DM6425HR-68D

The IDAN-DM6425HR-68D is a DM6425HR dataModule packaged in an IDAN frame with a 68-pin high density D front panel, I/O connector (female).

IDAN-DM6425HR-68D Dimensions and Connector

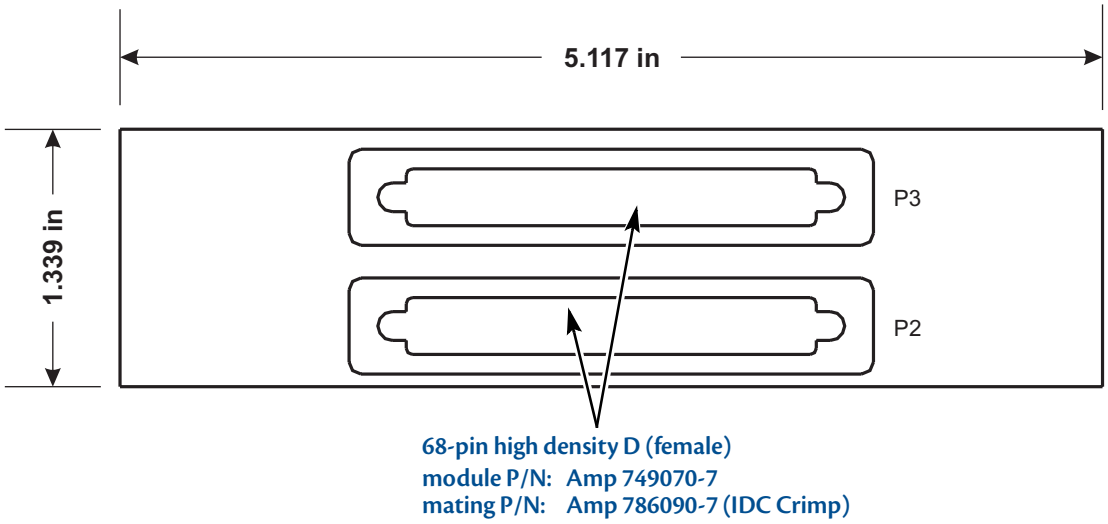


Figure 37 IDAN-DM6425HR-68D Connector

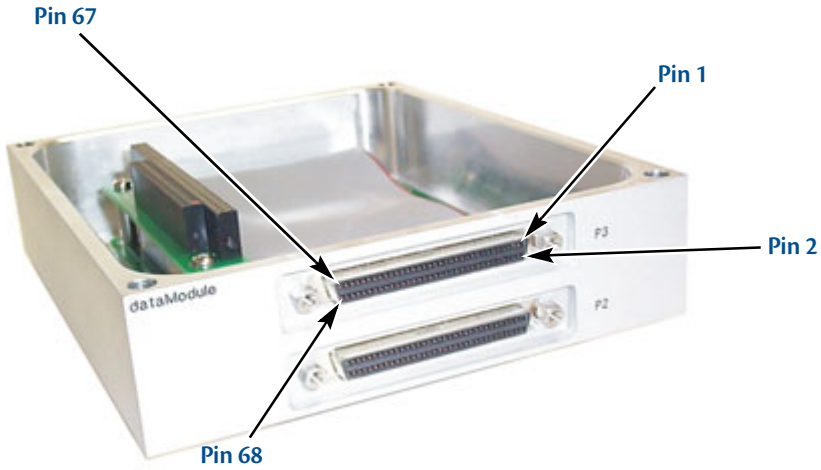


Figure 38 IDAN-DM6425HR-68D Connector Pins

CAUTION *Do not make connections to reserved pins!*

Note *Photographs are not to scale.*

IDAN-DM6425HR-68D External I/O Connections

Table 13 IDAN-DM6425HR-68D Connectors

IDAN P2 Pin #		Signal	DM6425 CN3 Pin #	IDAN P3 Pin #		Signal	DM6425 CN4 Pin #
Row 1	Row 2			Row 1	Row 2		
1	—	AIN1+/AIN1	1	1	—	AIN9+/AIN17	1
—	2	AIN1–/AIN9	2	—	2	AIN9–/AIN25	2
3	—	AIN2+/AIN2	3	3	—	AIN10+/AIN18	3
—	4	AIN2–/AIN10	4	—	4	AIN10–/AIN26	4
5	—	AIN3+/AIN3	5	5	—	AIN11+/AIN19	5
—	6	AIN3–/AIN11	6	—	6	AIN11–/AIN27	6
7	—	AIN4+/AIN4	7	7	—	AIN12+/AIN20	7
—	8	AIN4–/AIN12	8	—	8	AIN12–/AIN28	8
9	—	AIN5+/AIN5	9	9	—	AIN13+/AIN21	9
—	10	AIN5–/AIN13	10	—	10	AIN13–/AIN29	10
11	—	AIN6+/AIN6	11	11	—	AIN14+/AIN22	11
—	12	AIN6–/AIN14	12	—	12	AIN14–/AIN30	12
13	—	AIN7+/AIN7	13	13	—	AIN15+/AIN23	13
—	14	AIN7–/AIN15	14	—	14	AIN15–/AIN31	14
15	—	AIN8+/AIN8	15	15	—	AIN16+/AIN24	15
—	16	AIN8–/AIN16	16	—	16	AIN16–/AIN32	16
17	—	AOUT 1	17	17	—	AOUT 1	17
—	18	ANALOG GND	18	—	18	ANALOG GND	18
19	—	AOUT 2	19	19	—	AOUT 2	19
—	20	ANALOG GND	20	—	20	ANALOG GND	20
21	—	ANALOG GND	21	21	—	ANALOG GND	21
—	22	ANALOG GND	22	—	22	ANALOG GND	22
23	—	P0.7/DATA MARKER 3	23	23	—	P2.7	23
—	24	P1.7	24	—	24	P3.7	24
25	—	P0.6/DATA MARKER 2	25	25	—	P2.6	25
—	26	P1.6	26	—	26	P3.6	26
27	—	P0.5/DATA MARKER 1	27	27	—	P2.5	27
—	28	P1.5	28	—	28	P3.5	28
29	—	P0.4	29	29	—	P2.4	29
—	30	P1.4	30	—	30	P3.4	30
31	—	P0.3	31	31	—	P2.3	31
—	32	P1.3	32	—	32	P3.3	32
33	—	P0.2	33	33	—	P2.2	33
—	34	P1.2	34	—	34	P3.2	34
35	—	P0.1	35	35	—	P2.1	35
—	36	P1.1	36	—	36	P3.1	36
37	—	P0.0	37	37	—	P2.0	37
—	38	P1.0	38	—	38	P3.0	38
39	—	TRIGGER IN	39	39	—	RESERVED	39
—	40	DIGITAL GND	40	—	40	DIGITAL GND	40
41	—	EXT PCLK/STRB IN	41	41	—	RESERVED	41
—	42	EXT GATE 1	42	—	42	RESERVED	42
43	—	T/C OUT 1/DIG IRQ	43	43	—	RESERVED	43
—	44	T/C OUT 0	44	—	44	RESERVED	44
45	—	EXT CLK	45	45	—	RESERVED	45
—	46	EXT GATE 0	46	—	46	RESERVED	46
47	—	+12 VOLTS	47	47	—	+12 VOLTS	47
—	48	+5 VOLTS	48	—	48	+5 VOLTS	48
49	—	–12 VOLTS	49	49	—	–12 VOLTS	49
—	50	DIGITAL GND	50	—	50	DIGITAL GND	50

Appendix F 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 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 Blvd.
State College PA 16803-0906
USA
Website: www.rtd.com