ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Find the *RTD Embedded Technologies DM6420* at our website: *Click HERE*

DM6420HR
User's Manual



RTD Embedded Technologies, Inc.
"Accessing the Analog World"®

ISO9001 and AS9100 Certified

BDM-610010006
Rev. A

DM6420
User's Manual



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

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

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

web site
http://www.rtd.com

Revision History

Rev. A New manual naming method

Published by:

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

Copyright 1999, 2002, 2003 by RTD Embedded Technologies, Inc.
All rights reserved
Printed in U.S.A.

The RTD Logo is a registered trademark of RTD Embedded Technologies. cpuModule and utilityModule are trademarks of RTD Embedded Technologies. PhoenixPICO and PheonixPICO BIOS are trademarks of Phoenix Technologies Ltd. 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. PC/104 is a registered trademark of PC/104 Consortium. All other trademarks appearing in this document are the property of their respective owners.

Table of Contents

INTRODUCTION ***i*-1**

Analog-to-Digital Conversion *i*-3
Digital-to-Analog Conversion *i*-4
8254 Timer/Counters *i*-4
Digital I/O *i*-4
What Comes With Your Module *i*-4
Module Accessories *i*-4
Hardware Accessories *i*-4
Using This Manual *i*-5
When You Need Help *i*-5

CHAPTER 1 — MODULE SETTINGS **1-1**

Factory-Configured Switch and Jumper Settings 1-3
JP1— CN3, Pin 43 Signal Select (Factory Setting: OT1) 1-4
JP2 — User TC Clock Source Select (Factory Settings: Counter 0: OSC, Counter 1: OT0) 1-4
JP3 — DAC 1 Output Voltage Range (Factory Setting: +5 to -5 volts) 1-5
JP4 — DAC 2 Output Voltage Range (Factory Setting: +5 to -5 volts) 1-6
S1 — Base Address (Factory Setting: 300 hex (768 decimal)) 1-6
JS1 and JS2, Pull-up/Pull-down Resistors on Digital I/O Lines 1-7

CHAPTER 2 — INSTALLATION **2-1**

Installation 2-3
External I/O Connections 2-3
Connecting the Analog Input Pins 2-4
Connecting the Module for Simultaneous Sampling 2-6
Connecting the Analog Outputs 2-6
Connecting the Timer/Counters and Digital I/O 2-6
Running the 6420DIAG Diagnostics Program 2-6

CHAPTER 3 — HARDWARE DESCRIPTION **3-1**

A/D Conversion Circuitry 3-3
Analog Inputs 3-3
Channel-gain Scan Memory 3-3
A/D Converter 3-4
1024 Sample Buffer 3-4
Data Transfer 3-4
D/A Converters 3-4
Timer/Counters 3-4
Digital I/O 3-6

CHAPTER 4 — I/O MAPPING **4-1**

Defining the I/O Map 4-3
BA + 0: Clear/Program Clear Register (Read/Write) 4-4
BA + 2: Read Status/Program Control Register (Read/Write) 4-5
BA + 4: Read Converted Data/Load Channel-Gain & Digital Data (Read/Write) 4-7
BA + 6: Start Convert/Program Trigger Modes (Read/Write) 4-9
BA + 8: Program IRQ Register (Write) 4-11

BA + 10: Read Digital Input FIFO/Program Digital Input FIFO Configuration (Read/Write) 4-12
BA + 12: Load D/A Converter 1 Data (Write) 4-12
BA + 14: Load A/D Sample Counter/Load D/A Converter 2 Data (Read/Write) 4-13
BA + 16: TC Counter 0 (Read/Write) 4-13
BA + 18: TC Counter 1 (Read/Write) 4-13
BA + 20: TC Counter 2 (Read/Write) 4-13
BA + 22: Timer/Counter Control Word (Write Only) 4-13
BA + 24: Digital I/O Port 0, Bit Programmable Port (Read/Write) 4-14
BA + 26: Digital I/O Port 1, Byte Programmable Port (Read/Write) 4-14
BA + 28: Read/Program Port 0 Direction/Mask/Compare Registers (Read/Write) 4-14
BA + 30: Read Digital IRQ Status/Program Digital Mode (Read/Write) 4-15
Programming the DM6420 4-17
Clearing and Setting Bits in a Port 4-17

CHAPTER 5 — A/D CONVERSIONS 5-1

Before Starting Conversions: Initializing the Module 5-3
Before Starting Conversions: Programming Channel, Gain, Input Range and Type 5-3
Before Starting Conversions: Programming the Channel-Gain Table 5-4
16-Bit A/D Table 5-4
Channel Select, Gain Select, Input Range and Type 5-4
Pause Bit 5-5
Skip Bit 5-5
8-Bit Digital Table 5-6
Setting Up A/D and Digital Tables 5-6
Using the Channel-gain Table for A/D Conversions 5-7
Channel-gain Table and Throughput Rates 5-7
Channel-gain Data Store Enable (BA + 2, bit 4) 5-7
A/D Conversion Modes 5-7
Types of Conversions 5-10
Starting an A/D Conversion 5-12
Monitoring Conversion Status (FIFO Empty Flag or End-of-Convert) 5-12
Halting Conversions 5-12
Reading the Converted Data 5-13
Reading Data with the Channel-gain Data Store Bit Disabled 5-13
Reading Data with the Channel-gain Data Store Bit Enabled 5-14
Using the A/D Data Markers 5-15
Programming the Pacer Clock 5-16
Selecting 16-bit or 32-bit Pacer Clock 5-16
Programming Steps 5-16
Programming the Burst Clock 5-17
Programming the Sample Counter 5-18
Using the Sample Counter to Create Large Data Arrays 5-18

CHAPTER 6 — DATA TRANSFERS USING DMA 6-1

Choosing a DMA Channel 6-3
Allocating a DMA Buffer 6-3
Calculating the Page and Offset of a Buffer 6-4
Setting the DMA Page Register 6-5
The DMA Controller 6-6
DMA Mask Register 6-6
DMA Mode Register 6-7
Programming the DMA Controller 6-7
Programming the DM6420 for DMA 6-7

Monitoring for DMA Done 6-7
Dual DMA Mode 6-7
Common DMA Problems 6-8

CHAPTER 7 — INTERRUPTS 7-1

Software Selectable Interrupt Sources 7-3
Software Selectable Interrupt Channel 7-4
Advanced Digital Interrupts 7-4
Event Mode 7-4
Match Mode 7-4
Sampling Digital Lines for Change of State 7-4
Basic Programming For Interrupt Handling 7-5
What Is an Interrupt? 7-5
Interrupt Request Lines 7-5
8259 Programmable Interrupt Controller 7-5
Interrupt Mask Register (IMR) 7-5
End-of-Interrupt (EOI) Command 7-6
What Exactly Happens When an Interrupt Occurs? 7-6
Using Interrupts in Your Programs 7-6
Writing an Interrupt Service Routine (ISR) 7-6
Saving the Startup Interrupt Mask Register (IMR) and Interrupt Vector 7-7
Restoring the Startup IMR and Interrupt Vector 7-8
Common Interrupt Mistakes 7-8

CHAPTER 8 — D/A CONVERSIONS 8-1

CHAPTER 9 — TIMER/COUNTERS 9-1

CHAPTER 10 — DIGITAL I/O 10-1

Port 0, Bit Programmable Digital I/O 10-3
Advanced Digital Interrupts: Mask and Compare Registers 10-3
Port 1, Port Programmable Digital I/O 10-3
Resetting the Digital Circuitry 10-3
Strobing Data into Port 0 10-3
High Speed Digital Input 10-3

CHAPTER 11 — EXAMPLE PROGRAMS 11-1

C Programs 11-3
Quick Basic Programs 11-3

CHAPTER 12 — CALIBRATION 12-1

Required Equipment 12-3
A/D Calibration 12-4
Bipolar Calibration 12-4
Bipolar Range Adjustments: -5 to +5 Volts 12-4
Bipolar Range Adjustments: -10 to +10 Volts 12-4
Unipolar Calibration 12-5
Gain Adjustment 12-6
D/A Calibration 12-7

APPENDIX A — DM6420 SPECIFICATIONS A-1
APPENDIX B — CN3 CONNECTOR PIN ASSIGNMENTS B-1
APPENDIX C — COMPONENT DATA SHEETS C-1
APPENDIX D — WARRANTY D-1

List of Illustrations

1-1 Module Layout Showing Factory-Configured Settings 1-3
1-2 CN3, Pin 43 Signal Select Jumper, JP1 1-4
1-3 User TC Clock Sources Jumpers, JP2 1-4
1-4 User TC Circuit Diagram 1-5
1-5 DAC 1 Output Voltage Range Jumper, JP3 1-5
1-6 DAC 2 Output Voltage Range Jumper, JP4 1-6
1-7 Base Address Switch, S1 1-7
1-8 Ports 0 and 1 Pull-up/Pull-down Resistor Connections 1-7
2-1 CN3 I/O Connector Pin Assignments 2-4
2-2 Single-Ended Input Connections 2-5
2-3 Differential Input Connections 2-5
2-4 Two Modules Configured for Simultaneous Sampling 2-6
3-1 DM6420 Block Diagram 3-3
3-2 Clock TC Circuit Block Diagram 3-5
3-3 User TC Circuit Block Diagram 3-5
4-1 Using the Skip Bit 4-8
5-1 Setting the Skip Bit 5-5
5-2 Timing Diagram for Sampling Channels 1 and 4 5-5
5-3 A/D Conversion Select Circuitry 5-8
5-4 External Trigger Single Cycle Vs. Repeat Cycle 5-10
5-5 Timing Diagram, Single Conversion 5-10
5-6 Timing Diagram, Multiple Conversions 5-10
5-7 Timing Diagram, Random Channel Scan 5-11
5-8 Timing Diagram, Programmable Burst 5-11
5-9 Timing Diagram, Programmable Multiscan 5-12
5-10 Sample Buffer Circuitry 5-14
5-11 Storing Digital Data with Analog Data at the Acquisition Rate 5-15
5-12 Pacer Clock Block Diagram 5-16
5-13 Timing Diagram for Cycling the Sample Counter 5-19
7-1 Digital Interrupt Timing Diagram 7-4
9-1 Clock TC Circuitry 9-3
9-2 User TC Circuitry 9-3
12-1 Module Layout 12-3

INTRODUCTION

The DM6420 analog I/O dataModule® turns your IBM AT-compatible cpuModule™ or other PC/104 computer into a high-speed, high-performance data acquisition and control system. Ultra-compact for embedded and portable applications, the DM6420 module features:

- 8 differential or 16 single-ended analog input channels,
- 12-bit, 2 microsecond analog-to-digital converter with 500 kHz throughput,
- Programmable input ranges: ±5, ±10, or 0 to +10 volts,
- Programmable gains of 1, 2, 4 & 8,
- 1024 x 24 channel-gain scan memory with skip bit,
- Software, pacer clock and external trigger modes,
- Scan, burst and multiburst 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,
- 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
- 8 port programmable digital I/O lines,
- Six 16-bit timer/counters (two available to user) and on-board 8 MHz clock,
- Two 12-bit digital-to-analog output channels,
- ±5, 0 to +5, or 0 to +10 volt analog output range,
- Programmable interrupt source,
- +5 volt operation,
- Windows™ example programs in Visual Basic and C,
- DOS example programs with source code in BASIC and C,
- Diagnostics software.

The following paragraphs briefly describe the major functions of the module. A detailed discussion of module functions is included in subsequent chapters.

Analog-to-Digital Conversion

The DM6420 is software configurable on a channel-by-channel basis for up to 16 single-ended or 8 differential analog inputs. Software programmable unipolar and bipolar input ranges and gains allow easy interfacing to a wide range of sensors. Overvoltage protection to ±12 volts is provided at the inputs. The common mode input voltage for differential operation is ±10 volts.

A/D conversions are typically performed in 2 microseconds, and the maximum throughput rate of the board is 500 kHz. Conversions are controlled by software command, by an on-board 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 multiburst 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.

Digital-to-Analog Conversion

The digital-to-analog (D/A) circuitry features two independent 12-bit analog output channels with individually jumper-selectable output ranges of -5 to +5 volts, 0 to +5 volts, or 0 to +10 volts. 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.

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. The Clock TC is used for board operations. Two of the 16-bit timer/counters in the Clock TC are cascaded and used internally for the pacer clock. The third timer/counter is used as the burst clock. The User TC has two 16-bit timer/counters for user functions and one 16-bit timer/counter for the sample counter.

Digital I/O

The DM6420 has 16 buffered TTL/CMOS digital I/O lines which are grouped as eight independent, bit programmable lines at Port 0, and an 8-bit programmable port at Port 1. The bit programmable lines 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, masking can be used to monitor selected lines. Pull-up or pull-down resistors are provided for all 16 lines. Instructions for activating these pull-up/pull-down resistors are given at the end of Chapter 1, *Module Settings*. A 1024 sample buffer is also connected to the Port 0 lines. This buffer can be used to sample inputs up to a 1 MHz rate.

What Comes With Your Module

You receive the following items in your module package:

- DM6420 interface module with stackthrough bus header
- Mounting hardware
- Windows™ example programs in Visual Basic and C
- Example programs in BASIC and C with source code & diagnostics software
- User's manual

If any item is missing or damaged, please call Real Time Devices' Customer Service Department at (814) 234-8087. If you require service outside the U.S., contact your local distributor.

Module Accessories

In addition to the items included in your module package, Real Time Devices offers a full line of software and hardware accessories. Call your local distributor or our main office for more information about these accessories and for help in choosing the best items to support your module's application.

Hardware Accessories

Hardware accessories for the DM6420 include the TMX32 analog input expansion board with thermocouple compensation which can expand a single input channel on your module to 16 differential or 32 single-ended input channels, the OP series optoisolated digital input boards, the MR series mechanical relay output boards, the OR16 optoisolated digital input/mechanical relay output board, the USF4 universal sensor interface with sensor excitation, the TS16 thermocouple sensor board, the TB50 terminal board and XB50 prototype/terminal board for easy signal access and prototype development, the DM16 extender board for testing your module in a conventional desktop computer, and XT50 twisted pair wire flat ribbon cable assembly for external interfacing.

Using This Manual

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. We assume that you already have an understanding of data acquisition principles and that you can customize the example software or write your own application programs.

When You Need Help

This manual and the example programs in the software package included with your board provide enough information to properly use all of the module's features. If you have any problems installing or using this dataModule, contact our Technical Support Department, (814) 234-8087, during regular business hours, eastern standard time or eastern daylight time, or send a FAX requesting assistance to (814) 234-5218. 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 us through our E-mail address **techsupport@rtdusa.com**.

CHAPTER 1

MODULE SETTINGS

The DM6420 has jumper and switch settings you can change if necessary for your application. The module is factory-configured as listed in the table and shown on the layout diagram in the beginning of this chapter. Should you need to change these settings, use these easy-to-follow instructions before you stack the module with your computer system.

Also note that by placing solder connections on the bottom of the the board at JS1 and JS2, you can configure each set of digital I/O lines to be pulled up or pulled down. This procedure is explained at the end of this chapter.

Factory-Configured Switch and Jumper Settings

Table 1-1 lists the factory settings of the user-configurable jumpers and switch on the DM6420 module. Figure 1-1 shows the module layout and the locations of the factory-set jumpers. The following paragraphs explain how to change the factory settings. Pay special attention to the setting of S1, the base address switch, to avoid address contention when you first use the module in your system.

Table 1-1 Factory Settings

Switch/ Jumper	Function Controlled	Factory Settings (Jumpers Installed)
JP1	Selects the signal available at CN3, pin 43	OT1 (User TC, Counter 1)
JP2	Sets the clock source for User TC Counters 0 & 1	Clk 0: XTAL; Clk 1: OT0 (timer/counters cascaded)
JP3	Sets the D/A output voltage range for DAC 1	±5 (-5 to +5 volts)
JP4	Sets the D/A output voltage range for DAC 2	±5 (-5 to +5 volts)
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)
S1	Sets the base address	300 hex (768 decimal)

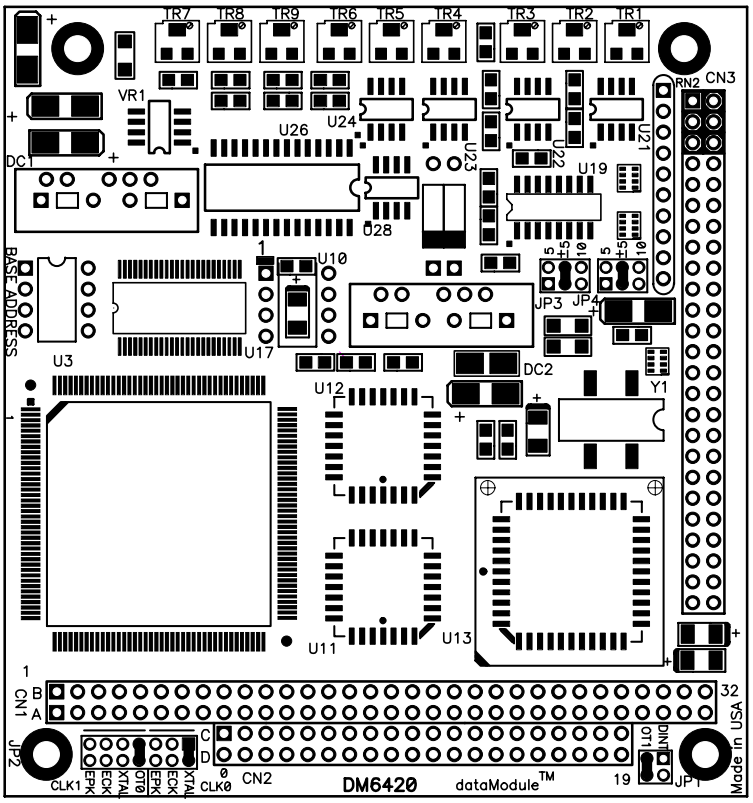


Fig. 1-1 — Module Layout Showing Factory-Configured Settings

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

This header connector, shown in Figure 1-2, lets you select the output signal from the module 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.

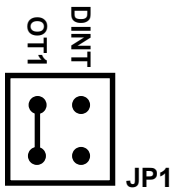


Fig. 1-2 — CN3, Pin 43 Signal Select Jumper, JP1

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

This header connector, shown in Figure 1-3, lets you select the clock sources for User TC Counters 0 and 1, the 16-bit timer/counters available for user functions. Figure 1-4 shows a block diagram of the User TC circuitry to help you in making these connections.

The clock source for Counter 0 is selected by placing a jumper on one of the three rightmost pairs of pins on the header, XTAL, ECK, or EPK. XTAL is the on-board 8 MHz clock; ECK is an external clock source which can be connected through I/O connector CN3, pin 45; and EPK is an external pacer clock which can be connected through I/O connector CN3, pin 41.

The four leftmost 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 on-board 8 MHz clock; ECK is an external clock source which can be connected through I/O connector CN3, pin 45; and EPK is an external pacer clock which 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.

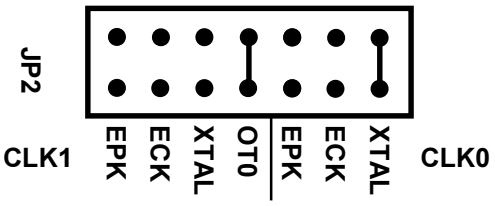


Fig. 1-3 — User TC Clock Sources Jumpers, JP2

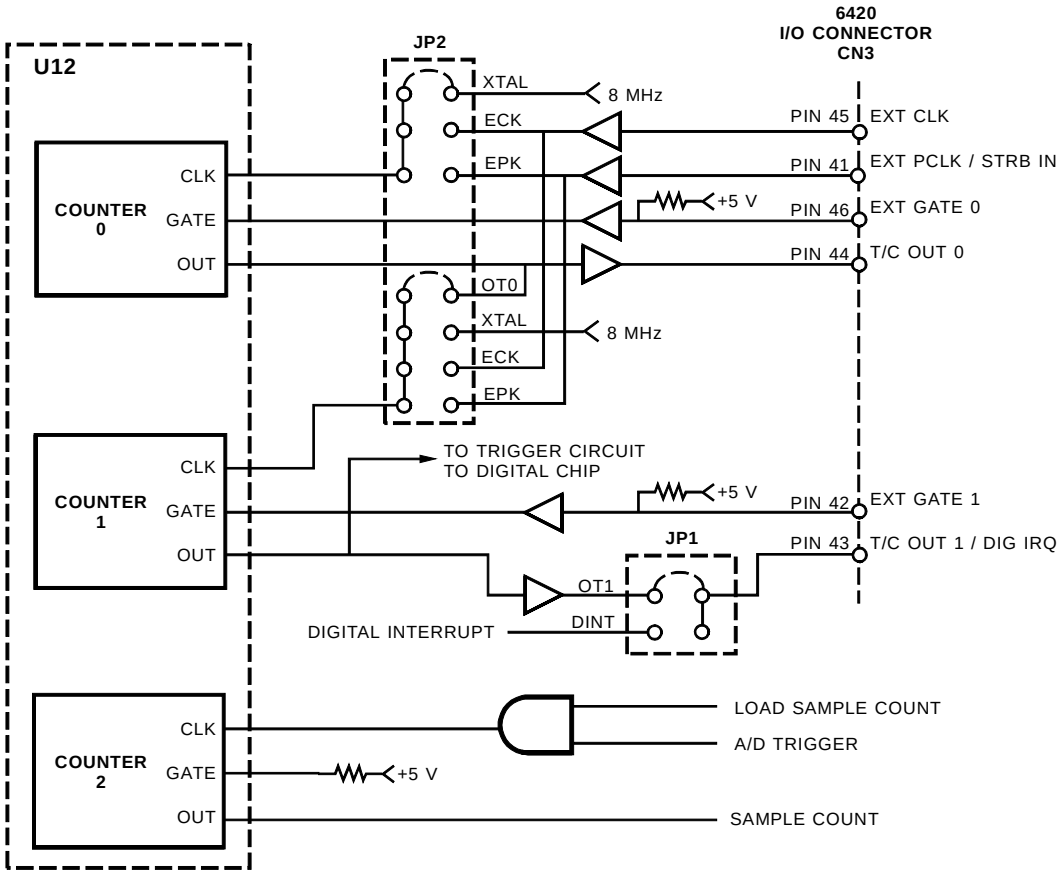


Fig. 1-4 — User TC Circuit Diagram

JP3 — DAC 1 Output Voltage Range (Factory Setting: +5 to -5 volts)

This header connector, shown in Figure 1-5, sets the output voltage range for DAC 1 at 0 to +5, ±5, or 0 to +10 volts (left to right on the header). This header does not have to be set the same as JP4.

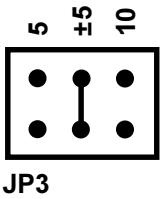


Fig. 1-5 — DAC 1 Output Voltage Range Jumper, JP3

JP4 — DAC 2 Output Voltage Range (Factory Setting: +5 to -5 volts)

This header connector, shown in Figure 1-6, sets the output voltage range for DAC 2 at 0 to +5, ±5, or 0 to +10 volts (left to right on the header). This header does not have to be set the same as JP3.

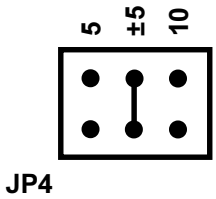


Fig. 1-6 — DAC 2 Output Voltage Range Jumper, JP4

S1 — Base Address (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 DM6420 has an easily accessible DIP switch, S1, which lets you select any one of 16 starting addresses in the computer's I/O. Should the factory setting of 300 hex (768 decimal) be unsuitable for your system, you can select a different base address simply by setting the switches to any one of the values listed in Table 1-2. The table shows the switch settings and their corresponding decimal and hexadecimal (in parentheses) values. Make sure that you verify the order of the switch numbers on the switch (1 through 4) before setting them. When the switches are pulled forward, they are OPEN, or set to logic 1, as labeled on the DIP switch package. When you set the base address for your module, record the value in the table inside the back cover. Figure 1-7 shows the DIP switch set for a base address of 300 hex (768 decimal).

Table 1-2 Base Address Switch Settings, S1

Base Address Decimal / (Hex)	Switch Setting 4 3 2 1	Base Address Decimal / (Hex)	Switch Setting 4 3 2 1
512 / (200)	0 0 0 0	768 / (300)	1 0 0 0
544 / (220)	0 0 0 1	800 / (320)	1 0 0 1
576 / (240)	0 0 1 0	832 / (340)	1 0 1 0
608 / (260)	0 0 1 1	864 / (360)	1 0 1 1
640 / (280)	0 1 0 0	896 / (380)	1 1 0 0
672 / (2A0)	0 1 0 1	928 / (3A0)	1 1 0 1
704 / (2C0)	0 1 1 0	960 / (3C0)	1 1 1 0
736 / (2E0)	0 1 1 1	992 / (3E0)	1 1 1 1

0 = closed, 1 = open

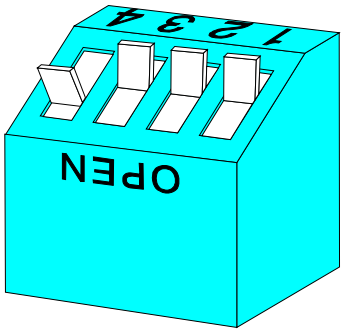


Fig. 1-7 — Base Address Switch, S1

JS1 and JS2, Pull-up/Pull-down Resistors on Digital I/O Lines

The DM6420 has 16 TTL/CMOS compatible digital I/O lines which can be interfaced with external devices. These lines are divided into two groups: Port 0 with eight individual bit programmable lines, and Port 1 with eight port programmable lines. Resistors are connected to these lines and can be configured as either pull-up or pull-down resistors.

10 k ohm pull-up/pull-down resistors are installed on the module, 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 and JS2 for Port 1. The factory default is pull-up for both ports. This is done by placing a solder short between the middle (common) pad and V (+5 volts). 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.

WARNING: Do not install a connection between all three pads as this will damage the board!!

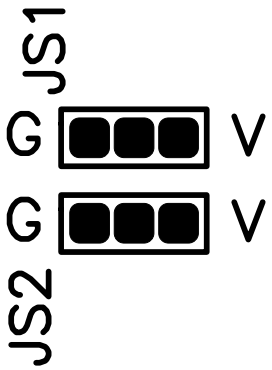


Fig. 1-8 — Ports 0 and 1 Pull-up/Pull-down Resistor Connections

CHAPTER 2

INSTALLATION

The DM6420 is easy to install in your cpuModule™ or other PC/104 based system. This chapter tells you step-by-step how to connect the module.

After you have made all of your connections, you can turn your system on and run the 6420DIAG board diagnostics program included on your example software disk to verify that the module is working.

Installation

Keep the module in its antistatic bag until you are ready to install it in your cpuModule™ or other PC/104 based system. When removing it from the bag, hold the module at the edges and do not touch the components or connectors.

Before installing the module in your system, check the jumper and switch settings. Chapter 1 reviews the factory settings and how to change them. If you need to change any settings, refer to the appropriate instructions in Chapter 1. Note that incompatible jumper settings can result in unpredictable module operation and erratic response.

The DM6420 comes with stackthrough connectors for CN1 and CN2. These stackthrough connectors let you stack another module on top of your DM6420. Pins B10 and C19 are keying pins and will be plugged on the top of the board and removed from the bottom

NOTE: The DM6420 module will only work with an AT cpuModule. Do not try to use it with an XT cpuModule.

To install the module, follow the procedures described in the computer manual and the steps below:

1. Turn OFF the power to your system.
2. Touch a metal rack to discharge any static buildup and then remove the module from its antistatic bag.
3. Select the appropriate standoffs for your application to secure the module when you install it in your system.
4. Holding the module by its edges, orient it so that the bus connector's pin 1 lines up with pin 1 of the expansion connector onto which you are installing the module.
5. After carefully positioning the module so that the pins are lined up and resting on the expansion connector, gently and evenly press down on the module until it is secured on the connector.

   NOTE: Do not force the module onto the connector. If the module does not readily press into place, remove it and try again. Wiggling the module or exerting too much pressure can result in damage to the DM6420 or to the mating module.
6. After the module is installed, connect the cable to I/O connector CN3 on the module. When making this connection, note that there is no keying to guide you in orientation. You must make sure that pin 1 of the cable is connected to pin 1 of CN3 (pin 1 is marked on the module with a small square). For twisted pair cables, pin 1 is the dark brown wire; for standard single wire cables, pin 1 is the red wire.
7. Make sure all connections are secure.

External I/O Connections

Figure 2-1 shows the DM6420's CN3 I/O connector pinout. Refer to this diagram as you make your I/O connections. Note that +12 volts at pin 47 and -12 volts at pin 49 are available only if your computer bus supplies them (these voltages are not provided by the module).

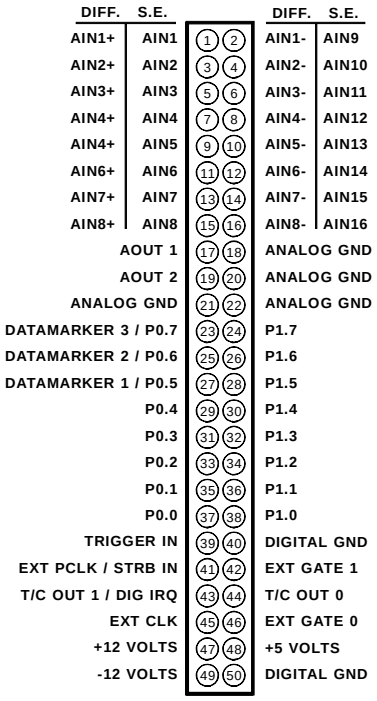


Fig. 2-1 — CN3 I/O Connector Pin Assignments

Connecting the Analog Input Pins

The analog inputs on the module can be set for single-ended or differential operation.

NOTE: It is good practice to connect all unused channels to ground, as shown in the following diagrams.

Single-Ended. When operating in the single-ended mode, connect the high side of the analog input to one of the analog input channels, AIN1 through AIN16, and connect the low side to an ANALOG GND (pins 18 and 20-22 on CN3). Figure 2-2 shows how these connections are made.

Differential. When operating in the 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 kilohm resistor pack at location RN2 on the module 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 AIN8+, 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). Figure 2-3 shows how these connections are made.

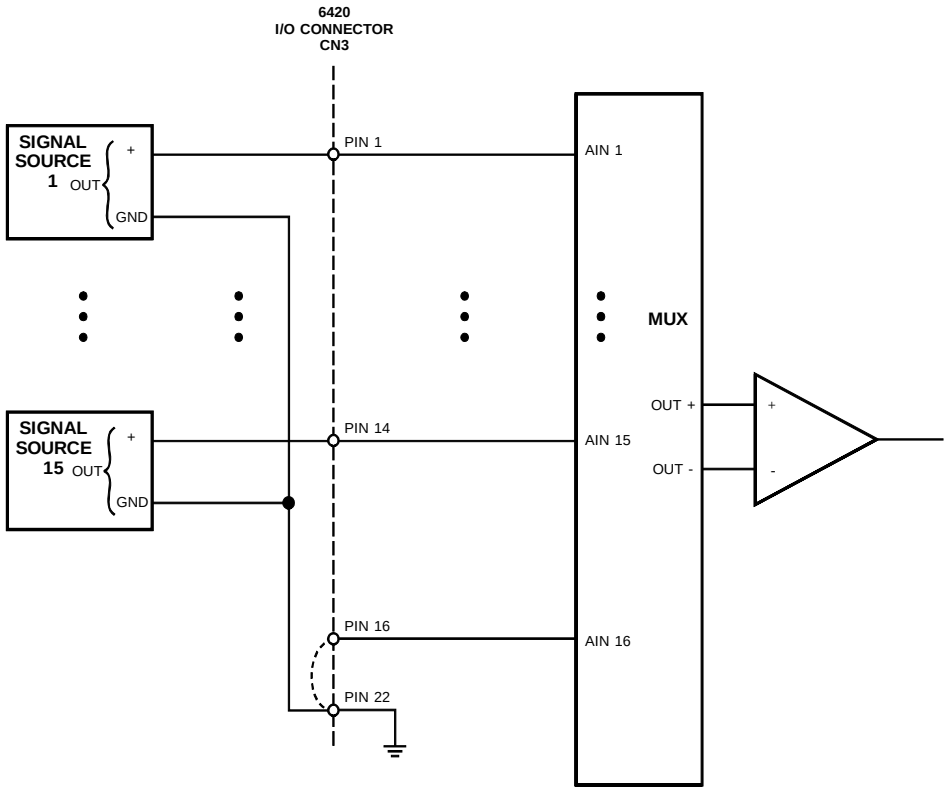


Fig. 2-2 — Single-Ended Input Connections

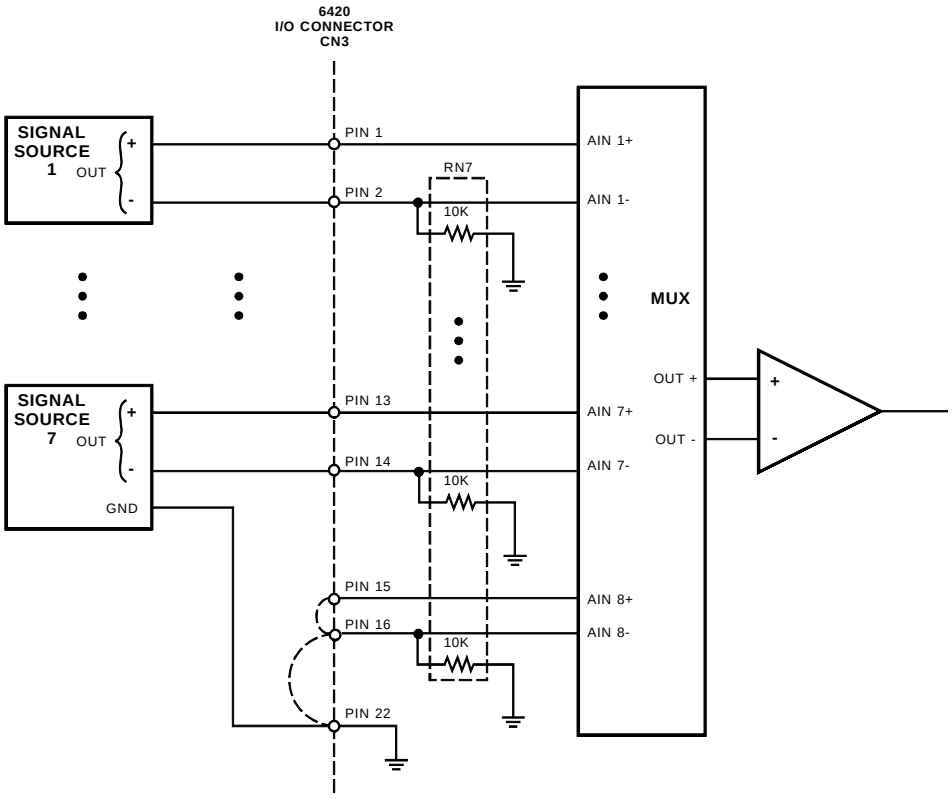


Fig. 2-3 — Differential Input Connections

Connecting the Module for Simultaneous Sampling

Multiple modules can be sampled simultaneously by connecting an external trigger source to the TRIGGER IN pin, CN3-39, and an external pacer clock to the EXT PCLK pin, CN3-41, of each module. Figure 2-4 shows 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 +6, Trigger Mode Register. The external trigger pulse duration should be at least 100 nanoseconds.

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 2-4 and selecting the external pacer clock at bit 9 in the Trigger Mode Register programmed at BA + 6. The trigger will start the pacer clock, and the pacer clock will simultaneously start conversions on all modules.

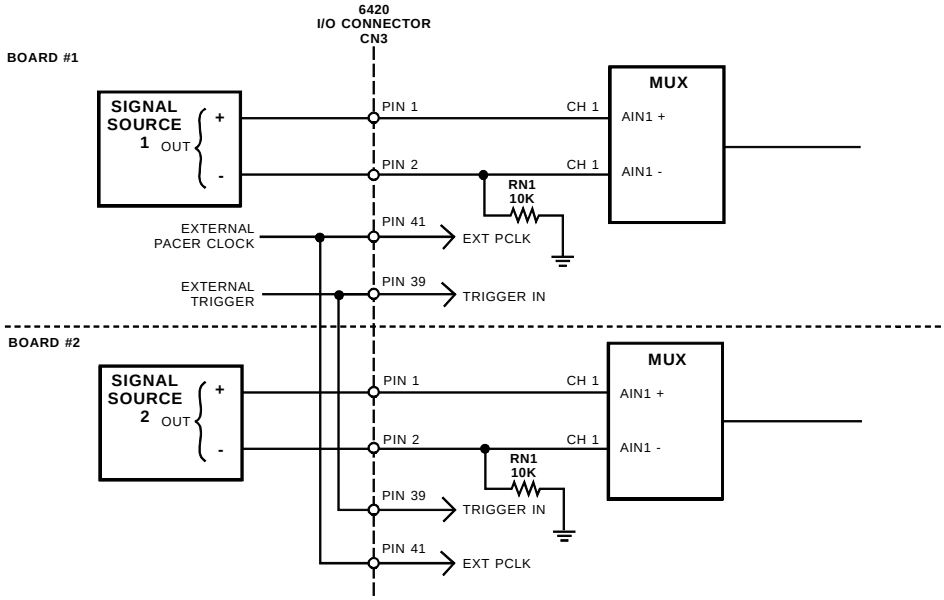


Fig. 2-4 — Two Modules Configured for Simultaneous Sampling

Connecting the Analog Outputs

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

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.

Running the 6420DIAG Diagnostics Program

Now that your module is ready to use, you will want to try it out. An easy-to-use, menu-driven diagnostics program, 6420DIAG, is included with your example software to help you verify your module's operation. You can also use this program to make sure that your current base address setting does not contend with another device.

CHAPTER 3

HARDWARE DESCRIPTION

This chapter describes the features of the DM6420 hardware. The major circuits are the A/D, the D/A, the timer/counters, and the digital I/O lines.

The DM6420 has four major circuits, the A/D, the D/A, the timer/counters, and the digital I/O lines. Figure 3-1 shows the block diagram of the module. This chapter describes the hardware which makes up the major circuits.

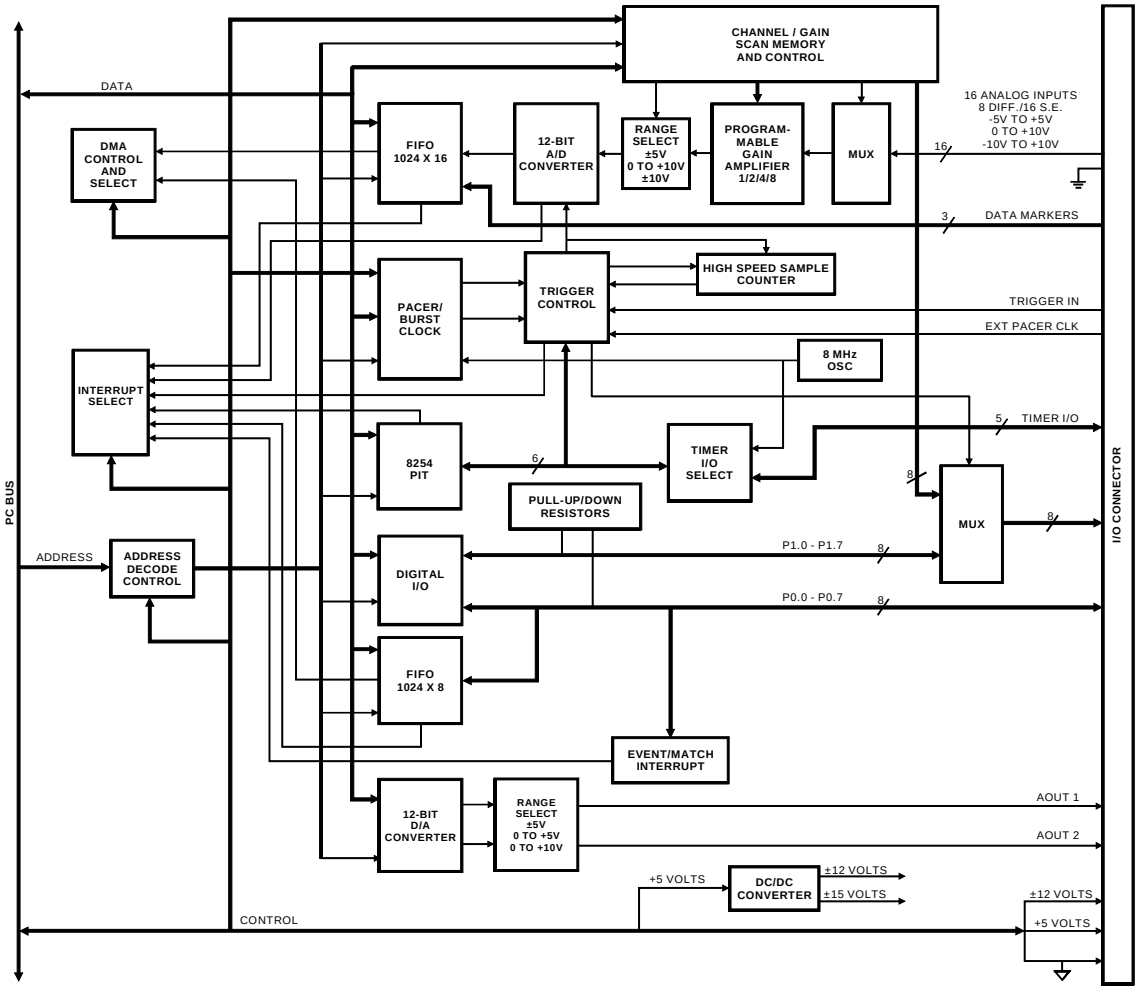


Fig. 3-1 — DM6420 Block Diagram

A/D Conversion Circuitry

The DM6420 performs analog-to-digital conversions on up to 16 software-selectable analog input channels. The following paragraphs describe the A/D circuitry.

Analog Inputs

The input voltage range is software programmable for -5 to +5 volts, -10 to +10 volts, or 0 to +10 volts. 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 volts 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, no A/D conversion is performed on the selected channel. Chapters 4 and 5 detail this feature.

A/D Converter

The 12-bit successive approximation A/D converter accurately digitizes dynamic input voltages in 2 microseconds, for a maximum throughput rate of 500 kHz. The converter IC contains a sample-and-hold amplifier, a 12-bit A/D converter, a 2.4-volt reference, a clock, and a digital interface to provide a complete A/D conversion function on a single chip. Its low power CMOS logic combined with a high precision, low noise design give you 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 on-board or external pacer clock can be used to control the conversion rate. Conversion modes are described in Chapter 5, *A/D Conversions*.

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.

D/A Converters

Two independent 12-bit analog output channels are included on the DM6420. The analog outputs are generated by two 12-bit D/A converters with independent jumper-selectable output ranges of ±5, 0 to +5, or 0 to +10 volts. The 10-volt ranges have a resolution of 2.44 millivolts, and the 5-volt range has a resolution of 1.22 millivolts.

Timer/Counters

Two 8254 programmable interval timers provide six 16-bit, 8-MHz timer/counters to support a wide range of timing and counting functions. The 8254 at U11 is the Clock TC. Two of its 16-bit timer/counters, Counter 0 and Counter 1, are cascaded and reserved 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 3-2 shows the Clock TC circuitry.

The 8254 at U12 is the User TC. On the User TC, Counters 0 and 1 are available to the user. Counter 2 is the sample counter. Figure 3-3 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)

These modes are detailed in the 8254 Data Sheet, reprinted from Intel in Appendix C. The pacer clock, burst clock and sample counter should be programmed for mode 2 operation.

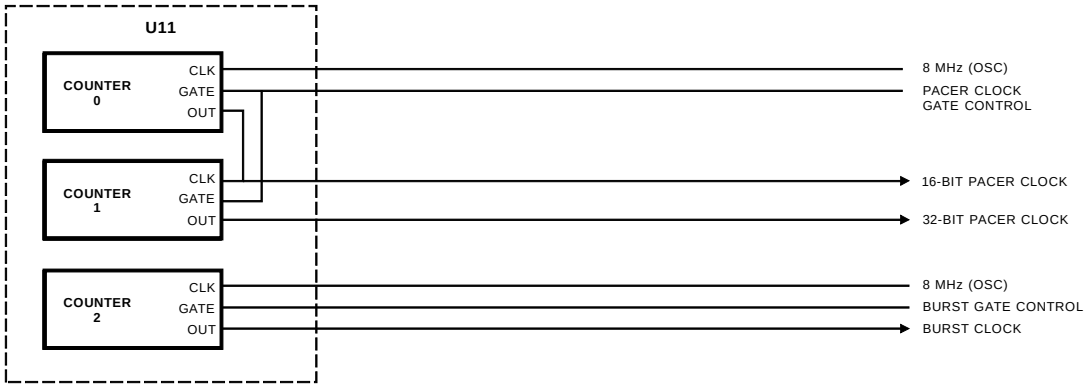


Fig. 3-2 — Clock TC Circuit Block Diagram

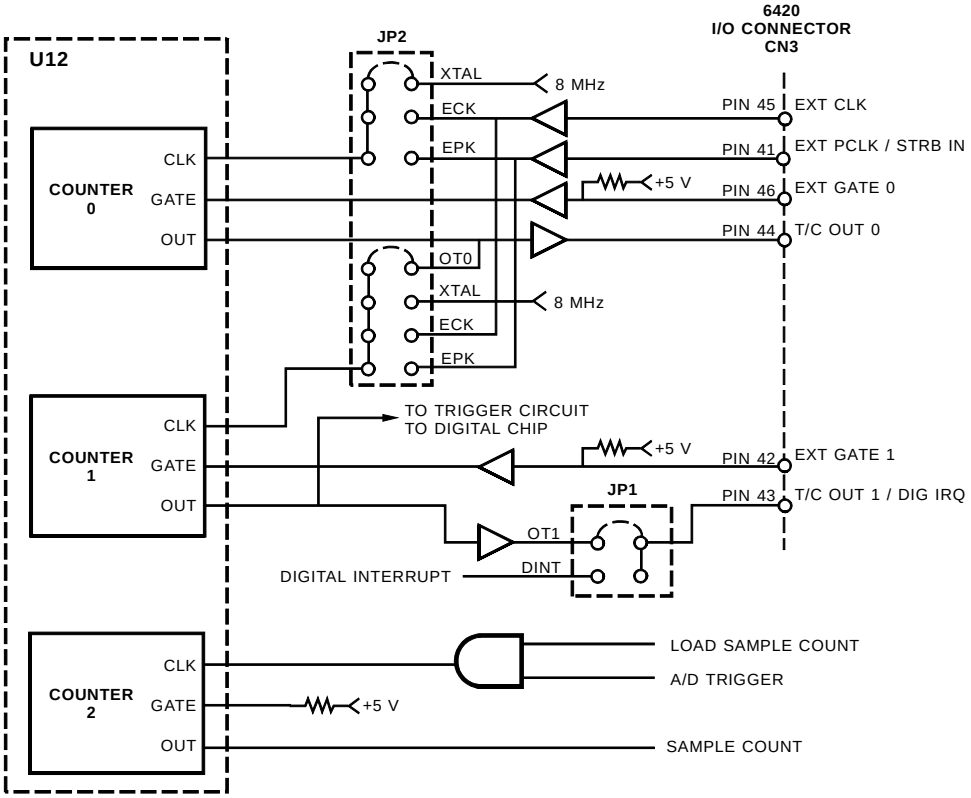


Fig. 3-3 — User TC Circuit Block Diagram

Digital I/O

The 16 digital I/O lines can be used to transfer data between the computer and external devices. Eight lines are bit programmable and eight lines are byte, or port, programmable.

Port 0 provides eight bit programmable lines which can be independently set for input or output. Port 0 supports RTD's two Advanced Digital Interrupt modes. An interrupt can be generated when the lines match a programmed value or when any bit changes its current state. A Mask Register lets you monitor selected lines for interrupt generation. A 1024 sample buffer is also connected to Port 0 to provide buffering for high speed digital inputs.

Port 1 can be programmed as an 8-bit input or output port.

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

CHAPTER 4

I/O MAPPING

This chapter provides a complete description of the I/O map for the DM6420, general programming information, and how to set and clear bits in a port.

Defining the I/O Map

The I/O map for the DM6420 is shown in Table 4-1 below. 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. Our 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 (designated as BA) can be selected using DIP switch S1, located on the edge of the board as described in Chapter 1, *Board 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.

Table 4-1 DM6420 I/O Map

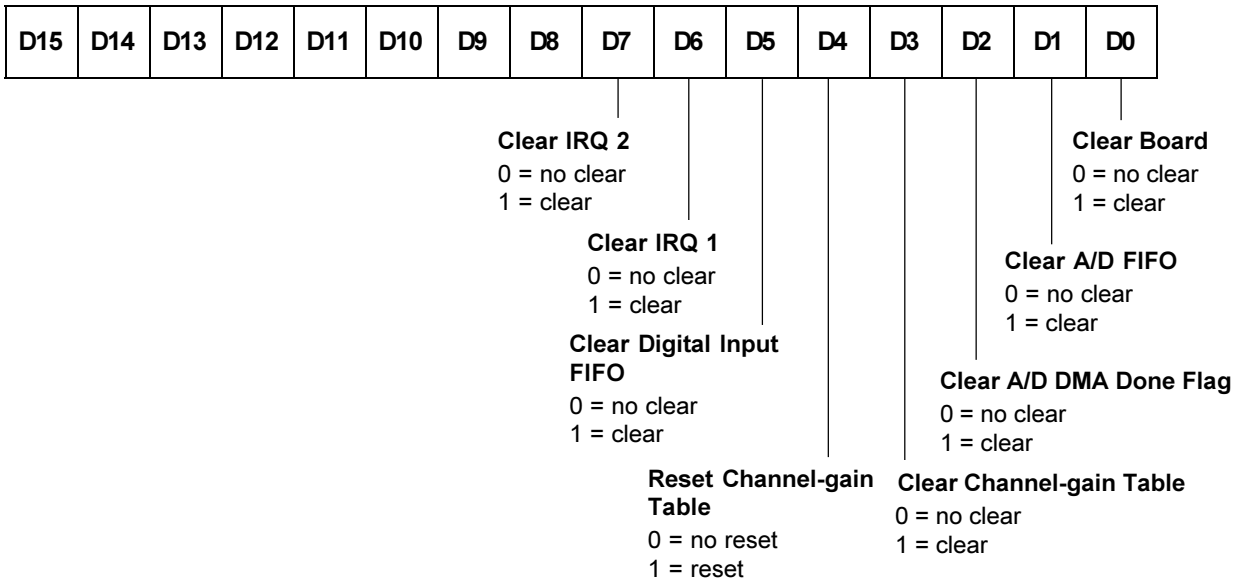
Register Description	Read Function	Write Function	Address * (Decimal)
Clear / Clear Mask Register	Clears board circuits programmed by a write to this address	Sets the board circuits to be cleared	BA + 0
Read Board Status / Set Control Register	Read board status word	Program 6420 control register	BA + 2
Read Converted Data / Load Channel-Gain Data	Read 12-bit converted data plus sign and data markers	Load channel & gain; Load A/D & digital data into channel-gain table	BA + 4
Start Convert / Set Trigger Modes	Software start convert	Program trigger modes	BA + 6
Set IRQ Source & Channel	Reserved	Program interrupt register	BA + 8
High Speed Digital Input Buffer	Read high speed digital input sample buffer	Initialize high speed digital input buffer	BA + 10
D/A Converter 1	Reserved	Program D/A converter 1	BA + 12
Initialize A/D Sample Counter / D/A Converter 2	Provides software trigger to load A/D sample counter	Program D/A converter 2	BA + 14
8254 Clock TC Counter 0 & User TC Counter 0	Read value in Clock or User TC Counter 0 (dependent on BA + 2)	Load count in Clock or User TC Counter 0 (dependent on BA + 2)	BA + 16
8254 Clock TC Counter 1 & User TC Counter 1	Read value in Clock or User TC Counter 1 (dependent on BA + 2)	Load count in Clock or User TC Counter 1 (dependent on BA + 2)	BA + 18
8254 Clock TC Counter 2 & User TC Counter 2	Read value in Clock or User TC Counter 2 (dependent on BA + 2)	Load count in Clock or User TC Counter 2 (dependent on BA + 2)	BA + 20
8254 Clock TC & User TC Control Word	Reserved	Program counter mode for Clock or User TC (dependent on BA + 2)	BA + 22
Digital I/O Port 0 (Bit Programmable)	Read Port 0 digital input lines	Program Port 0 digital output lines	BA + 24
Digital I/O Port 1 (Port Programmable)	Read Port 1 digital input lines	Program Port 1 digital output lines	BA + 26
Port 0 Clear/ Direction/Mask/Compare	Clear digital IRQ status flag/read Port 0 direction, mask or compare register (dependent on BA + 30)	Clear digital chip/program Port 0 direction, mask or compare register (dependent on BA + 30)	BA + 28
Read Digital I/O Status/ Set Digital Control Register	Read digital status word	Program digital control register & digital interrupt enable	BA + 30

* BA = Base Address

BA + 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 + 0 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 + 2, bit 0), sets the FIFO full flag high (BA + 2, bit 1), and clears the HALT flag (BA + 2, bit 2), enabling A/D conversions.
Bit 2 – When high (bit 2 = 1), clears the A/D DMA done flag at BA + 2, 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.

For 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.

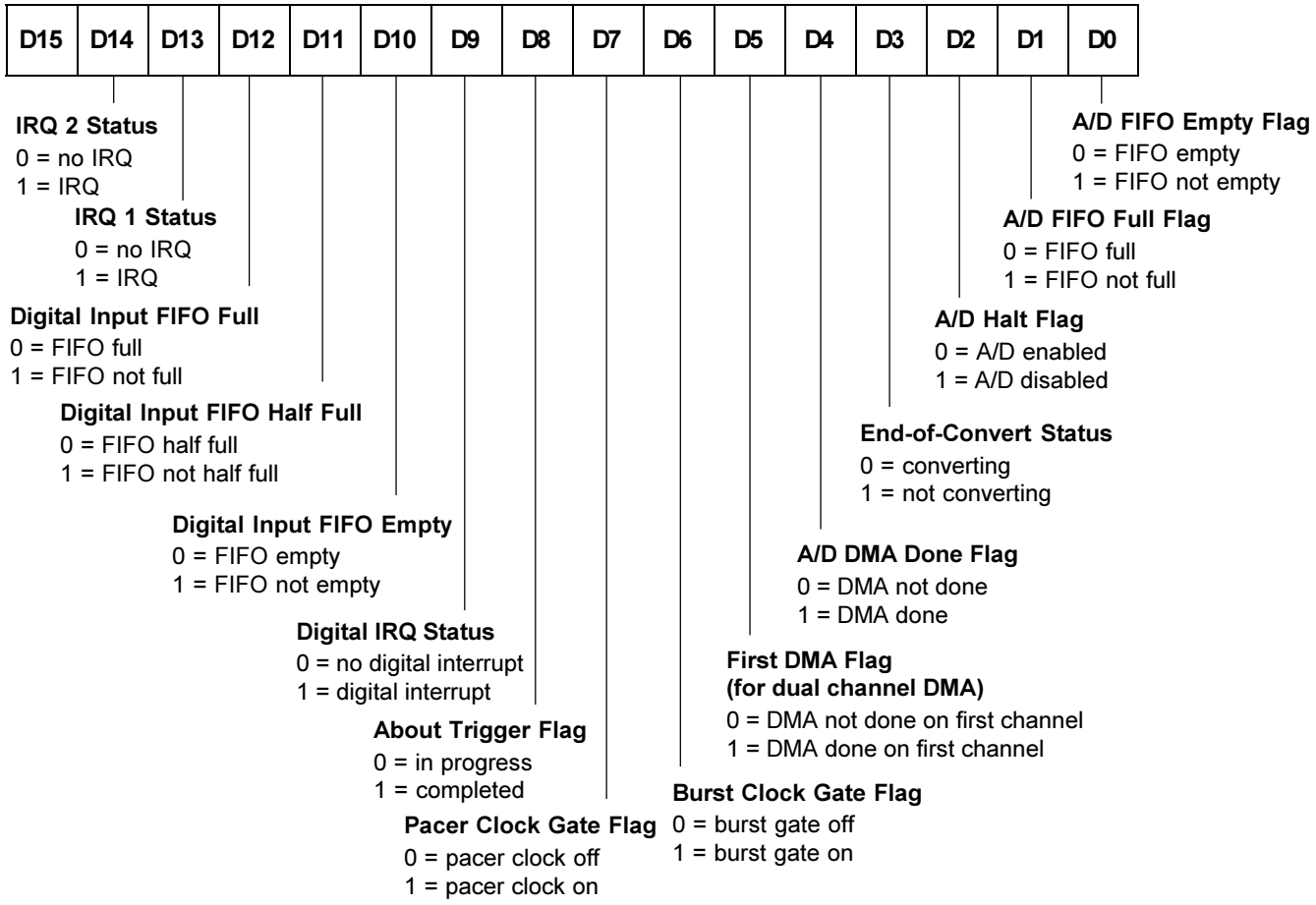
Clear FIFO and DMA Done Flag (value written = 6):

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

16-bit operation.

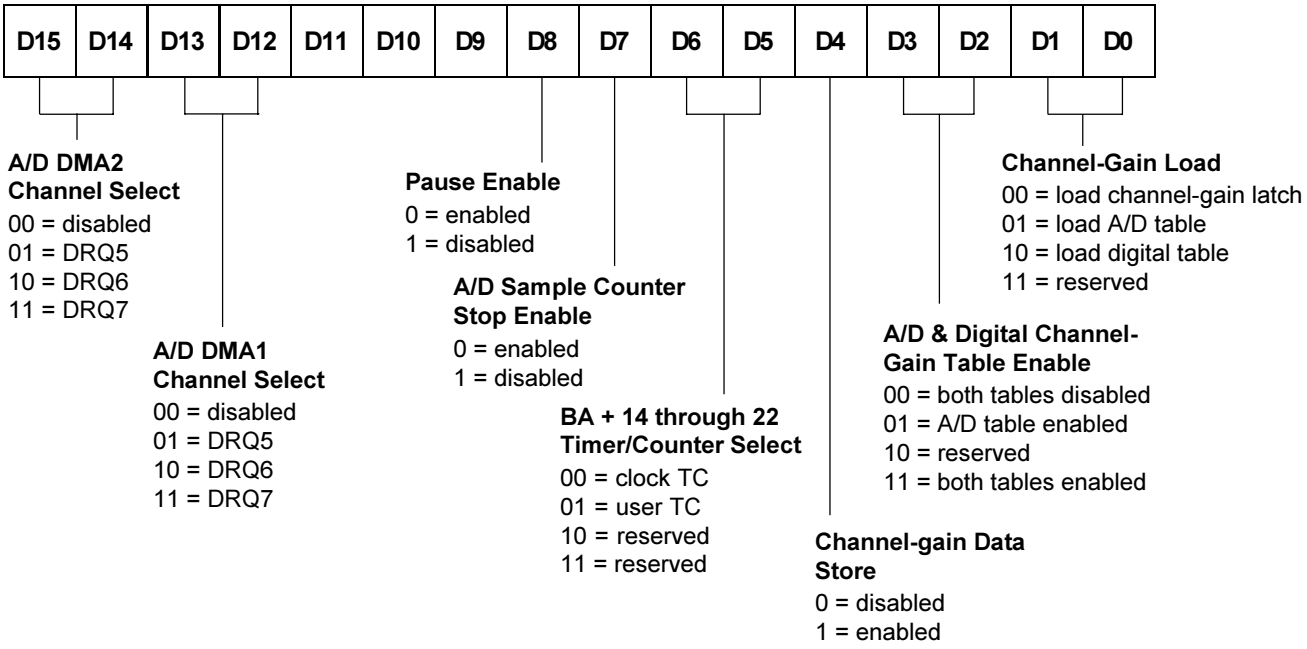
Status Register:



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 + 0 (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 + 2, 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, *DMA Transfers*.
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 9 – Shows when an Advanced Digital Mode interrupt has occurred. In this manual, the term digital interrupt specifically refers to an interrupt generated by the bit programmable digital I/O Port 0 Advanced Digital Interrupt circuitry.
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:



A write to BA + 2 sets up the Control Register shown above. The settings you enter here determine whether the channel-gain data written to BA + 4 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 + 16 through BA + 22, 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 + 4 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 A/D 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 A/D 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 A/D 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 A/D and Digital Tables in the channel-gain scan memory are activated to be used for A/D conversions. 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 + 0 is used to reset the table pointer.

Bit 4 – When enabled, the 16-bit channel-gain table entry for each conversion is stored in the sample buffer along with the converted data. The order of storage in the buffer is channel-gain table data, followed by the converted data.

Bits 5 and 6 – Selects the 8254 timer/counter to be programmed at BA + 16 through BA + 22. 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 A/D table in the channel-gain scan memory (BA + 4, bit 10) is activated. When disabled, the pause bit setting at BA + 4 is ignored.

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 + 4: Read Converted Data/Load Channel-Gain & Digital Data (Read/Write)

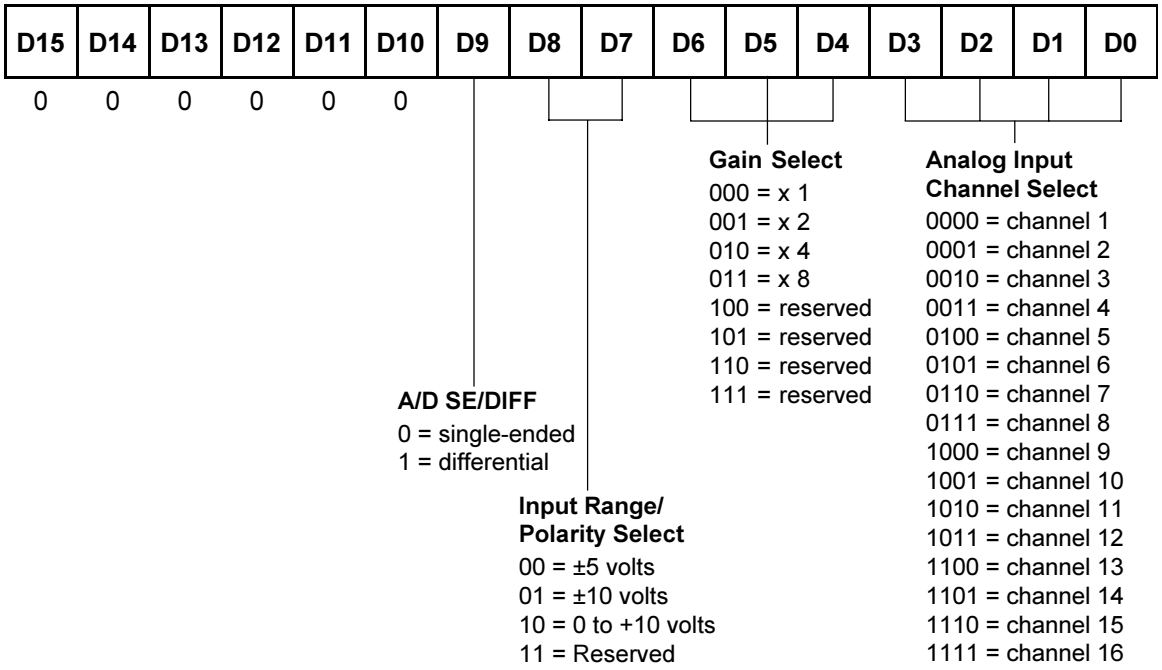
16-bit Data Word Read from FIFO (16-bit operation):

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
6igQ	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 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.

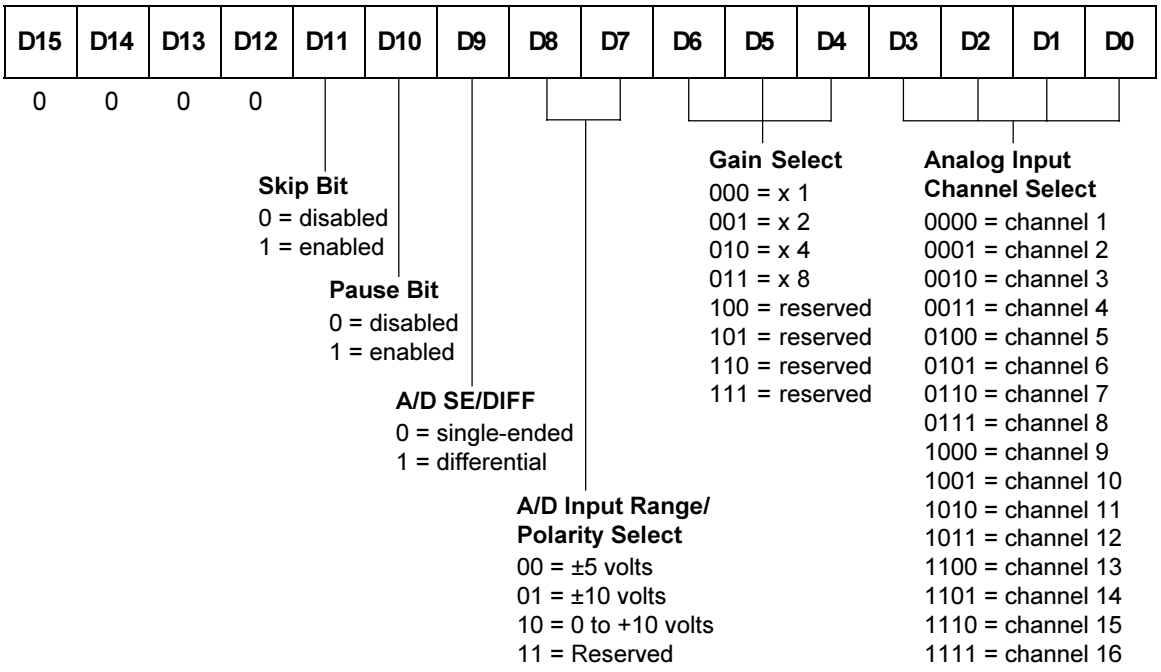
If the channel-gain data store bit at BA + 2, bit 4 is enabled, the first read at this address provides the 16-bit channel-gain table entry, and is followed by a second read to provide the converted data. Chapter 5 details how readings are taken when using the channel-gain data store feature.

Load Channel-Gain Latch (BA + 2, bits 1 and 0 = 00) (16-bit operation):



To load channel and gain for conversions not using the channel-gain table: First, make sure that bits 1 and 0 at BA + 2 are set to 00. Then write the desired channel and gain information to BA + 4. Bits 7 and 8 select the input range, and bit 9 selects whether the input is single-ended or differential.

Load A/D Table in Channel-Gain Scan Memory (BA + 2, bits 1 and 0 = 01) (16-bit operation):



To load the A/D portion of the channel-gain table with channel and gain information: First, set bits 1 and 0 at BA + 2 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 4-1. 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.

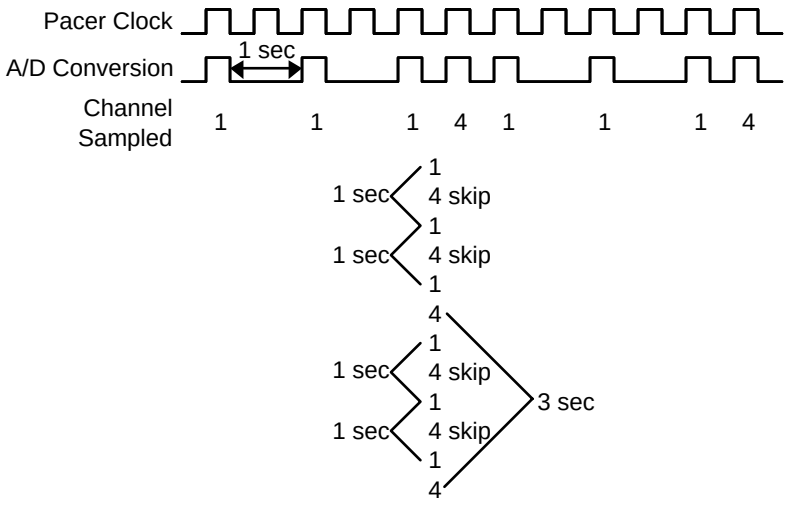
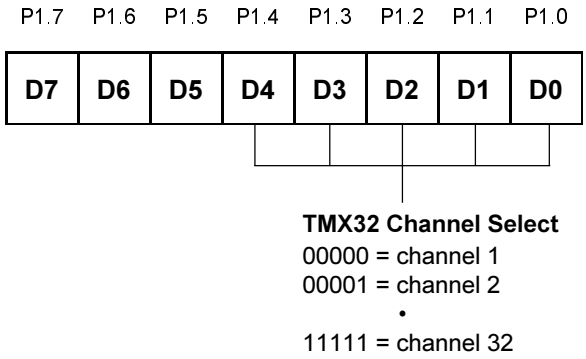


Fig. 4-1 — Using the Skip Bit

Load Digital Table in Channel-Gain Scan Memory (BA + 2, bits 1 and 0 = 10) (8-bit operation):

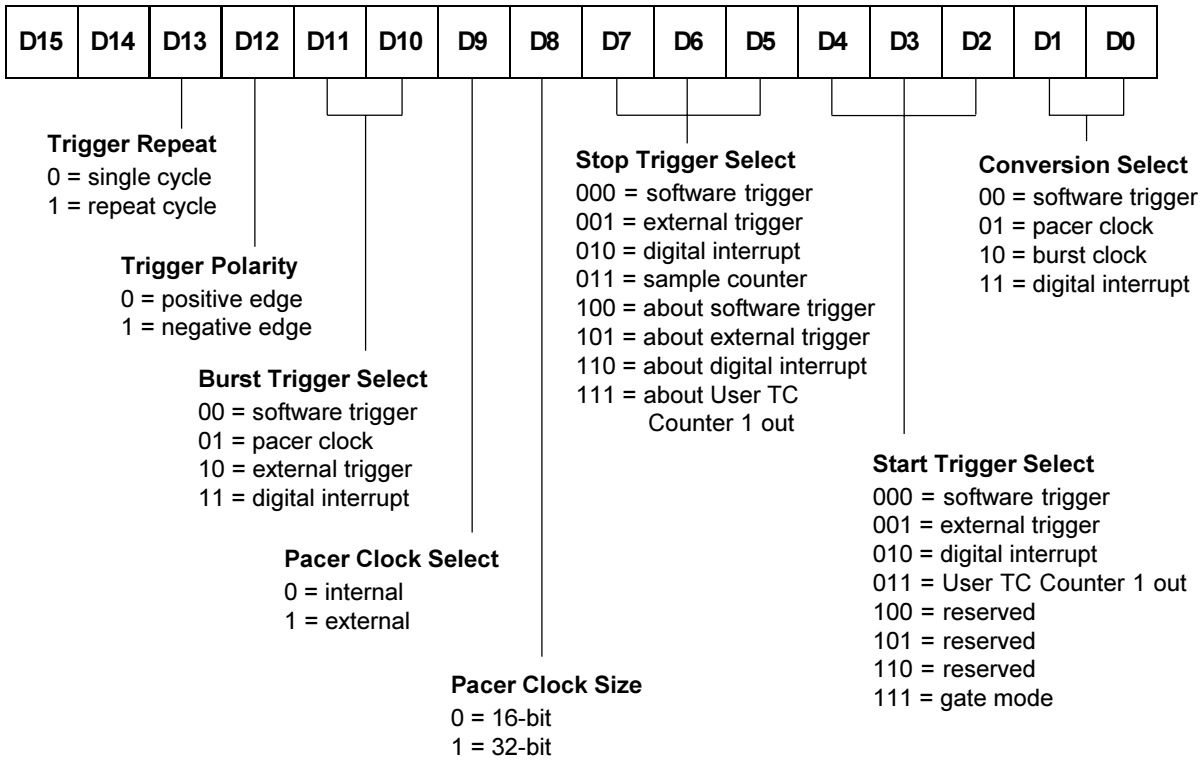


To load the digital portion of the channel-gain table with digital information: The digital portion of the channel-gain table provides 8 bits to control devices such as external expansion boards. For example, if you have connected one of your input channels on the DM6420 to RTD's TMX32 input expansion 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 + 2 to 10 to enable loading of the digital portion of the channel-gain table. Then, load the data, setting 0's and 1's 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 + 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 method by which A/D conversions are performed (conversion select bits) and the trigger mode.

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

00 = conversions are controlled by reading BA + 6 (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 + 6 (Start Convert).
001 = the pacer clock is started by an external trigger (TRIGGER IN, CN3-39).
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 + 6, bit 12.

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

000 = the pacer clock is stopped by reading BA + 6 (Start Convert).
001 = the pacer clock is stopped by an external trigger (TRIGGER IN, CN3-39).
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 number of samples taken after the stop trigger is received is set by the A/D sample counter.

100 = the A/D sample counter takes a specified number of samples after a read at BA + 6 (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 on-board 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 nanoseconds maximum. However, a 32-bit clock's jitter is dependent on the value programmed into the first divider and can be much greater than 125 nanoseconds. (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 CN3-41. The maximum pacer clock rate supported by the board is 500 kHz.

Bits 10 and 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.

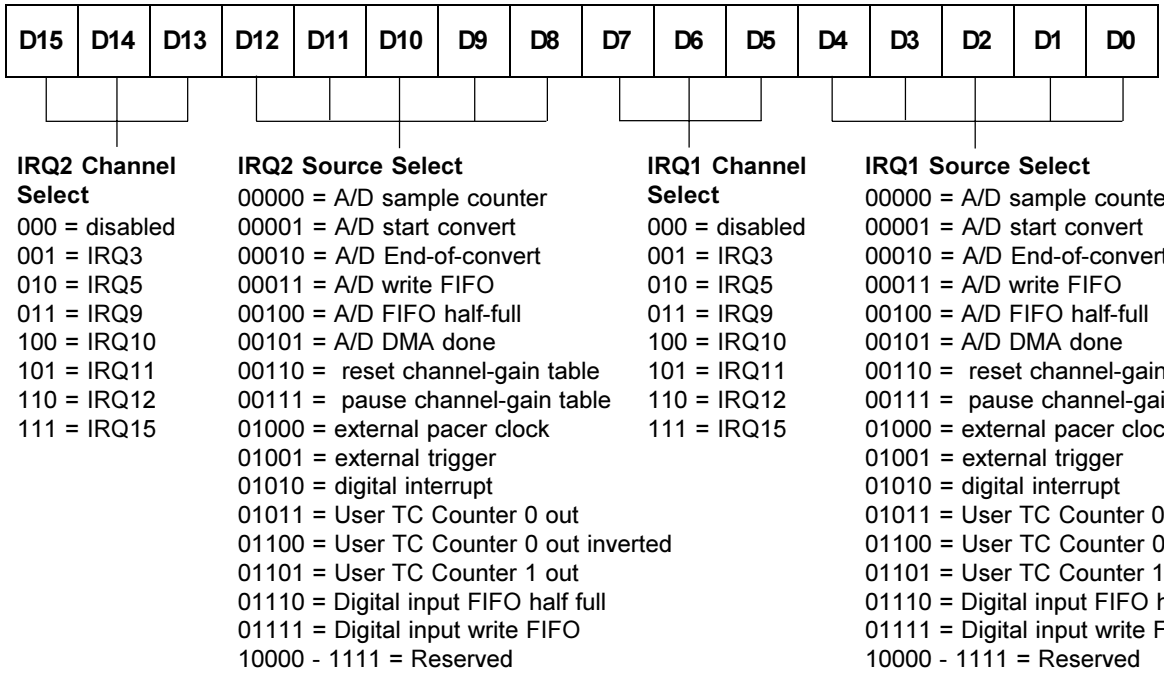
Bit 12 – Sets the external trigger to occur on the positive-going or negative-going edge of the pulse.

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 + 8: Program IRQ Source and Channel (Write)

16-bit operation.

Interrupt Register:



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 described below:

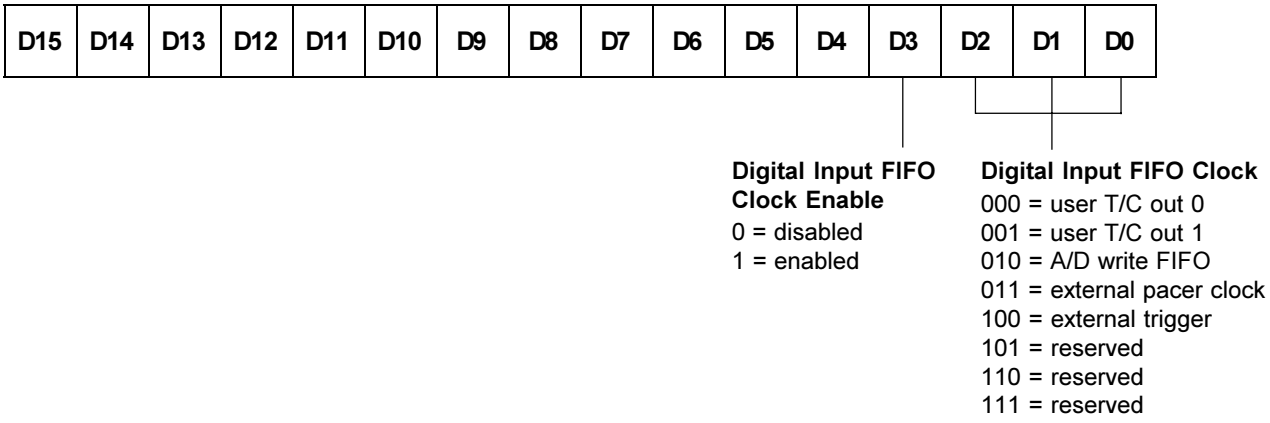
A/D sample counter - an interrupt is generated when the A/D sample counter count reaches 0.
A/D start convert - an interrupt is generated when a conversion is started.
A/D End-of-convert - an interrupt is generated when an end-of-convert is issued by the A/D converter.
A/D write FIFO - an interrupt is 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 + 10: Read 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 DM6420 as follows:

Bits 0, 1, 2 – These bits set the digital input FIFO clock source. Options include the outputs from the user T/C, the write pulse to the A/D FIFO, The external pacer clock (CN3-41) and the external trigger (CN3-39). 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 + 12: Load D/A Converter 1 Data (Write)

16-bit operation.

DAC1 Output:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	Bit 12 (MSB)	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1 (LSB)

A write programs the DAC1 12-bit output in the format shown above. Output coding is straight binary for both uni-polar and bi-polar ranges. The act of writing to this port automatically updates the D/A output.

BA + 14: Load A/D Sample Counter/Load D/A Converter 2 Data (Read/Write)

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.

DAC2 Output:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
				(MSB)											(LSB)

A write programs the DAC1 12-bit output in the format shown above. Output coding is straight binary for both uni-polar and bi-polar ranges. The act of writing to this port automatically updates the D/A output.

BA + 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 + 2, bits 5 and 6. A read shows the count in the counter.

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

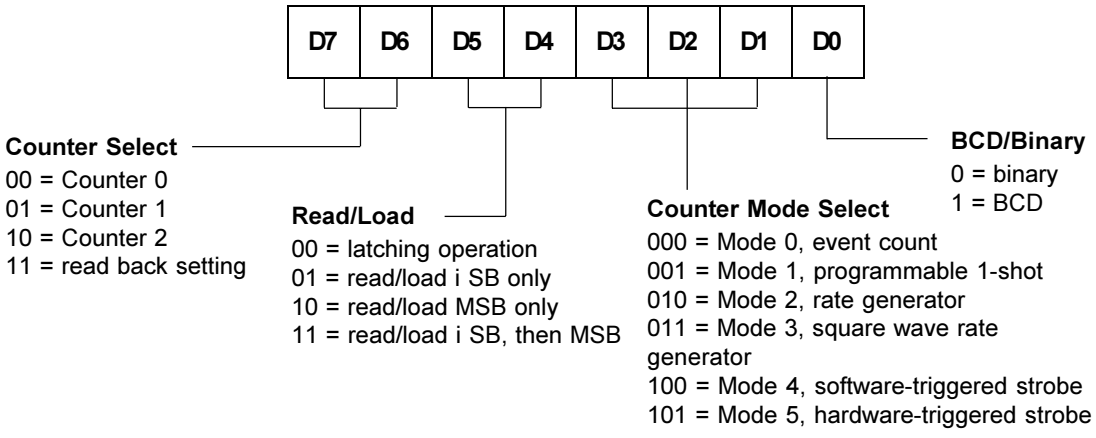
8-bit Operation. A write loads the second counter in one of the two timer/counters on the board with a new 16-bit value in two 8-bit steps, LSB followed by MSB. The counter must be loaded in two 8-bit steps! Counting begins as soon as the count is loaded. The timer/counter being loaded is selected by writing to BA + 2, bits 5 and 6. A read shows the count in the counter.

BA + 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 + 2, bits 5 and 6. A read shows the count in the counter.

BA + 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.



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

8-bit operation.

Port 0:

D7	D6	D5	D4	D3	D2	D1	D0
30.7	30.6	30.5	30.4	30.3	30.2	30.1	30.0

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 + 28. 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 + 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	P1.1

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 + 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 + 30. When bits 1 and 0 at BA + 30 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 + 30, 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 + 30, 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

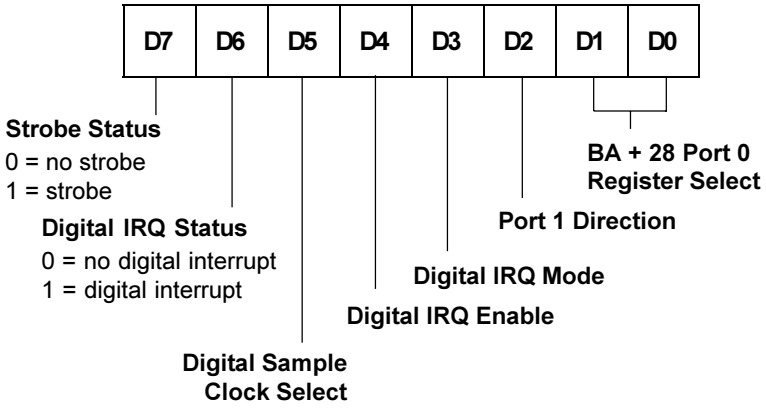
In the Advanced Digital Interrupt modes, this register is used to mask out specific bits when monitoring the bit pattern present at Port 0 for interrupt generation. In normal operation where the Advanced Digital Interrupt feature is not being used, any bit which is masked by writing a 1 to that bit will 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 un-masked).

Compare Register (BA + 30, bits 1 and 0 = 11):
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 + 30 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.

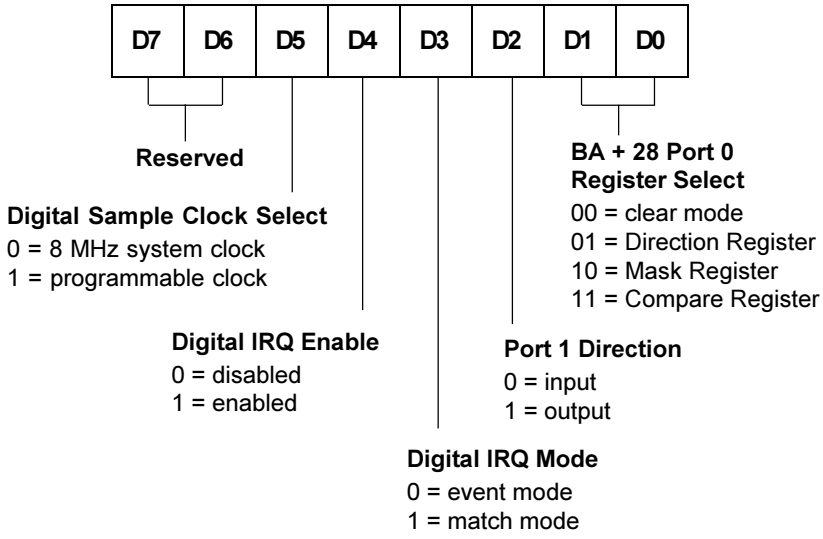
BA + 30: Read Digital IRQ Status/Program Digital Mode (Read/Write)

8-bit operation.

Digital IRQ/Strobe Status:
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:



Bits 0 and 1 – Select the clear mode initiated by a read/write operation at BA + 28 or the Port 0 control register you talk to at BA + 28 (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 + 28).

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 nanoseconds 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 – Reserved.

Programming the DM6420

This section gives you some general information about programming and the DM6420 board.

The DM6420 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 DM6420 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	Write16 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 DM6420, 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. **Be sure to use the correct function for 8- and 16-bit operations with the 6420!**

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 ($8 = 2^3$), 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: Don't 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 DM6420 to perform A/D conversions and read the results. Included in this discussion are instructions on setting up the channel-gain scan memory, the on-board clocks and sample counter, and various conversion and triggering modes.

The following paragraphs walk you through the programming steps for performing A/D conversions. Detailed information about the conversion modes and triggering is presented in this section. You can follow these 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 include:

Clear Board command.

Clear A/D DMA Done command

Clear Channel Gain Table command.

Clear Digital Input FIFO command.

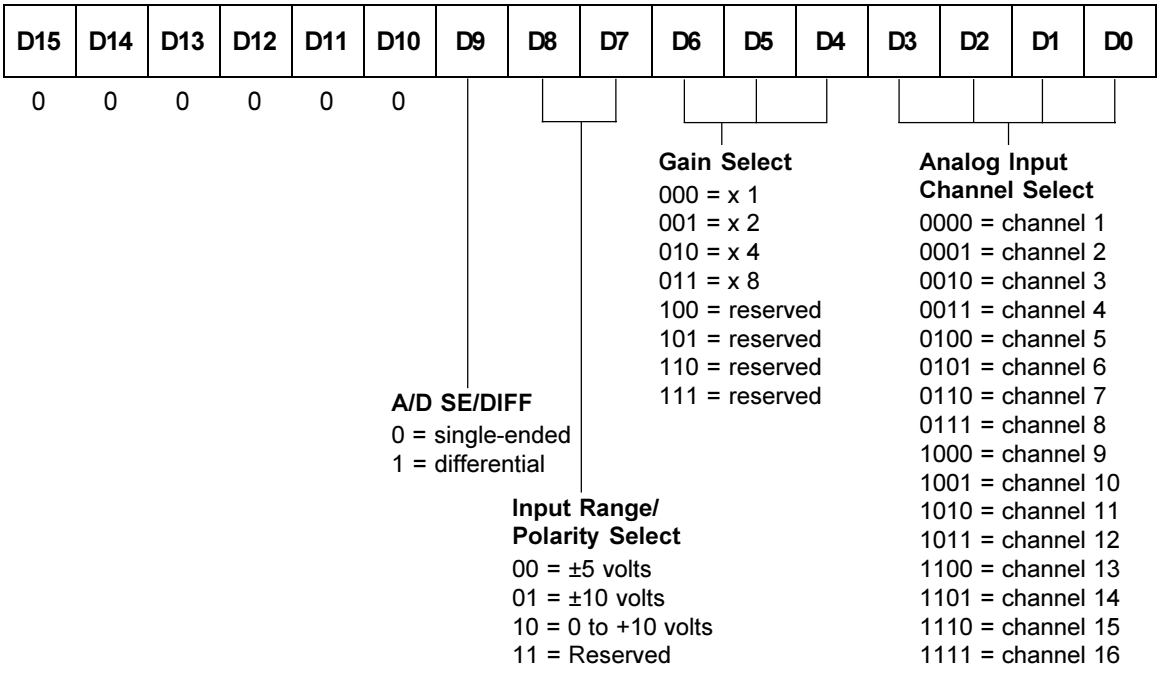
Clear Digital I/O chip.

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 + 0, BA + 28 and BA + 30. Since you cannot read back the contents of the Control Register (BA + 2), Trigger Register (BA + 6), IRQ Register (BA + 8) or the Digital Input FIFO Initilization Register (BA + 10) we recommend 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.

Before Starting Conversions: Programming Channel, Gain, Input Range and Type

The conversion channel, gain, input range and input type are programmed at BA + 4. To program a conversion channel for direct A/D conversion (not using the channel-gain table), you must first point BA + 4 to write to the channel/gain latch. This is done by setting bits D0 and D1 to "00" in the Control Register at BA + 2. To program the channel, gain, input range and input type, assign the appropriate values to bits 0 through 9 and write this value to BA + 4. The diagram below shows this register.



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

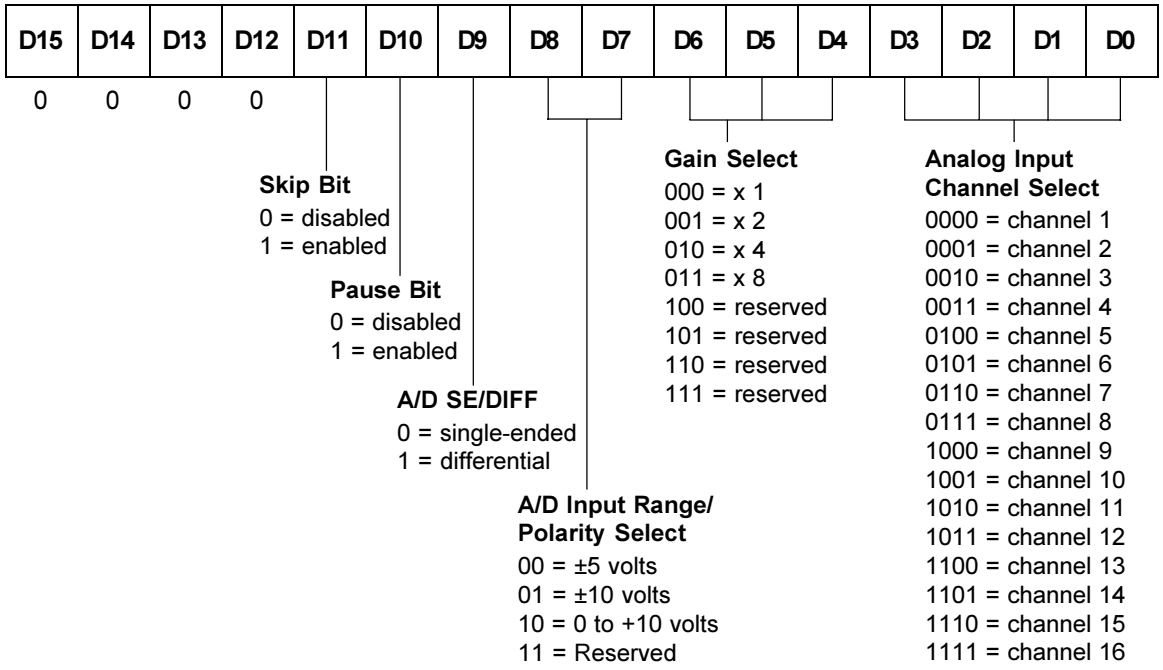
1. Set bits 1 and 0 at BA + 2 to 00 (this points BA + 4 to the channel/gain latch).
2. Write the channel and gain data to be loaded to BA + 4.

Before Starting Conversions: 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 + 0. 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 + 4. This register is defined below. To load channel and gain data into the A/D table, first set bits 1 and 0 at BA + 2 to 01. This points BA + 4 to write to the A/D table. Now you can write the 16-bit channel/gain word to BA + 4. 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 9. 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 16 single-ended inputs or 8 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+ (CN3-1) and AIN1- (CN3-2). Channel 8 now is not available as a single-ended channel.

Pause bit

Bit 10 is used as a pause bit. If this bit is set to a "1"and the Pause function is enabled at BA + 2, 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 2 different sequences loaded in the table. You can enable and disable this bit's function at BA + 2, bit 8.

NOTE: This bit is ignored in the Burst sampling modes.

Skip bit

If bit 11 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 5-1 and Figure 5-2, 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.

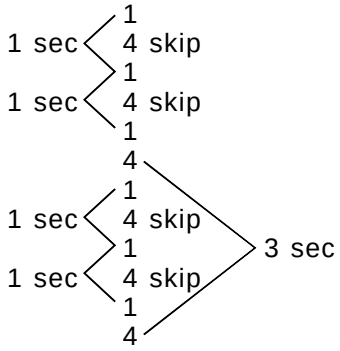


Fig. 5-1 — Setting the Skip Bit

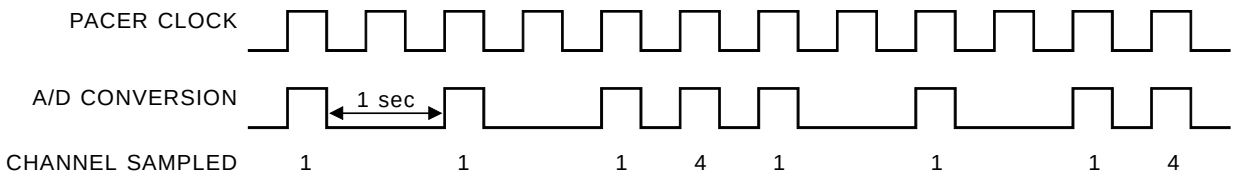


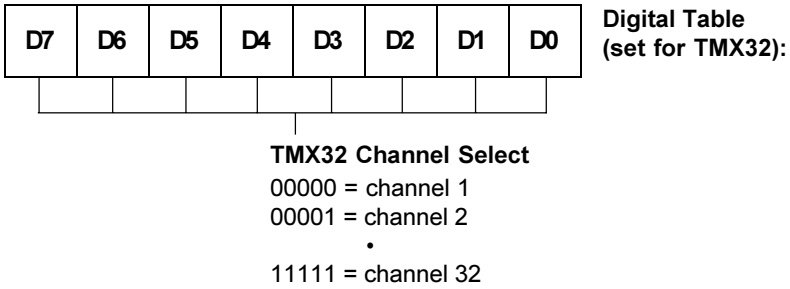
Fig. 5-2 — Timing Diagram for Sampling Channels 1 and 4

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 + 4. To load digital control data into the Digital table, first set bits 1 and 0 at BA + 2 to 10. This points BA + 4 to write to the Digital table. Now you can write the 8-bit byte to BA + 4. 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 DM6420. With BA + 2, 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, DM6420 channel = 1
Entry 2	0000 0000 0010 0000	gain = 4, DM6420 channel = 1
Entry 3	0000 1000 0000 0000	skip sample
Entry 4	0000 0000 0010 0000	gain = 4, DM6420 channel = 1
Entry 5	0000 0000 0000 0000	gain = 1, DM6420 channel = 1
Entry 6	0000 0000 0010 0000	gain = 4, DM6420 channel = 1

With BA + 0, 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, DM6420 channel = 1	0000 0000 TMX32 channel = 1
Entry 2	0000 0000 0010 0000	gain = 4, DM6420 channel = 1	0000 0011 TMX32 channel = 4
Entry 3	0000 1000 0000 0000	skip sample	0000 0000 TMX32 channel = 1 (skip)
Entry 4	0000 0000 0010 0000	gain = 4, DM6420 channel = 1	0000 0011 TMX32 channel = 4
Entry 5	0000 0000 0000 0000	gain = 1, DM6420 channel = 1	0000 0000 TMX32 channel = 1
Entry 6	0000 0000 0010 0000	gain = 4, DM6420 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 in order to be used for A/D conversions. Two bits control this operation. BA + 2, bit 2 enables the A/D Table where the channel and gain data are stored. BA + 2, 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 + 2 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 + 2, 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 + 0.

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.

Channel-gain Data Store Enable (BA + 2, bit 4)

When this bit is set to 1, a 16-bit channel-gain table entry is stored in the sample buffer with the converted data. This feature tags each 12-bit conversion with its channel-gain identifier.

Each channel-gain tag is stored in a 16-bit word in the sample buffer. For each conversion, the tag is sent to the sample buffer, followed by the converted data. When the channel-gain store feature is enabled, the sample buffer's capacity is reduced to 512 samples.

The channel-gain table data stored in the sample buffer is read as explained later in this chapter.

A/D Conversion Modes

To support a wide range of sampling requirements, the DM6420 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 + 6 set up how the A/D conversions are controlled. The following paragraphs describe the conversion and trigger modes, and Figure 5-3 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 + 6 control what method is used to actually perform the A/D conversions. One of four modes can be selected:

- Through software (by reading BA + 6 to initiate a Start Convert)
- Using a pacer clock (internal (Clock TC Counter 0 or 1) or external (CN3-41))
- Using the burst clock (Clock TC Counter 2)
- Using a digital interrupt generated by the Advanced Digital Interrupt circuit

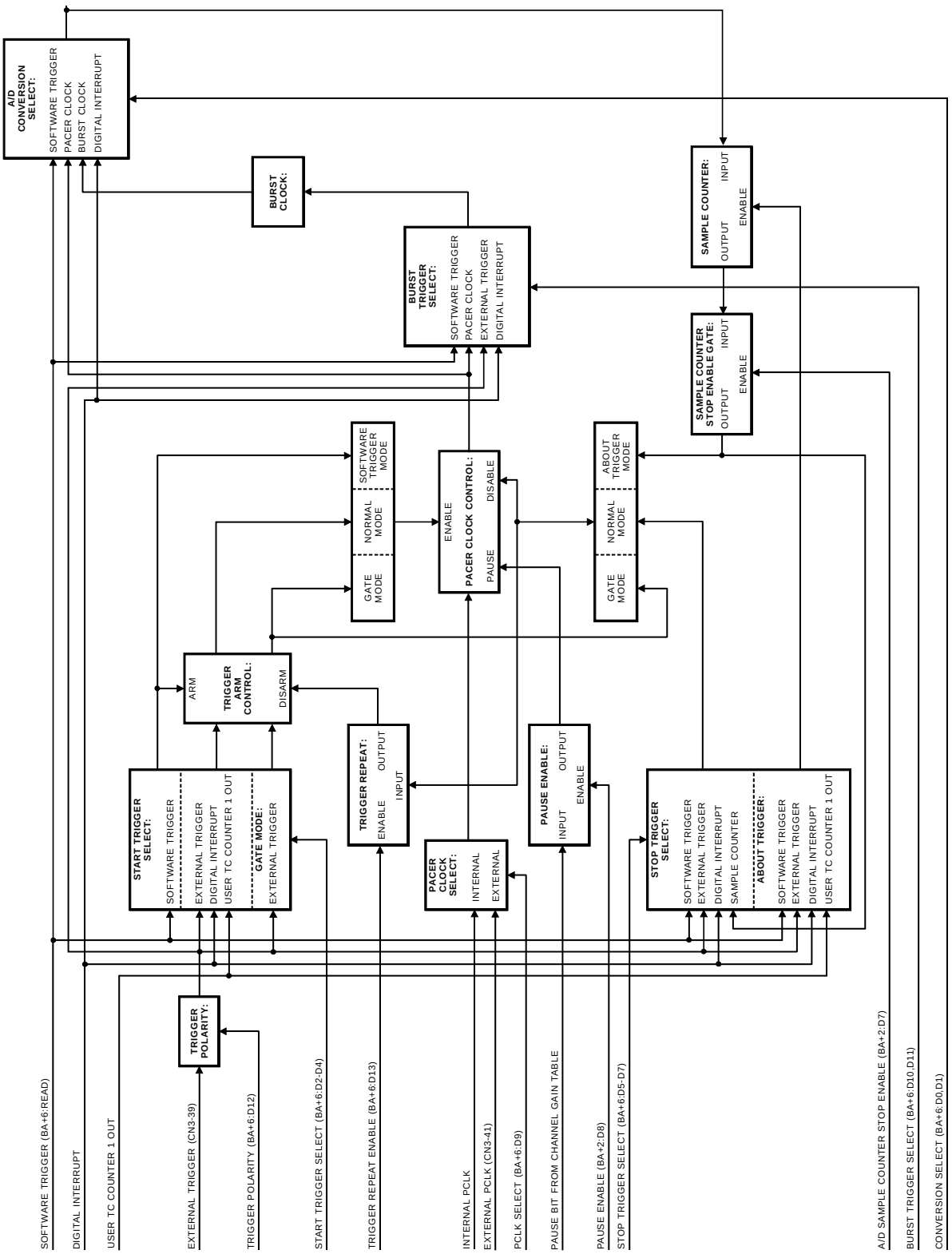


Fig. 5-3 — A/D Conversion Select Circuitry

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 + 6 are used to turn the pacer clock (internal or external) on and off. Through these different combinations of start and stop triggers, the DM6420 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 + 6 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, CN3-39, will start the pacer clock. The pulse duration should be at least 100 nanoseconds.
- 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, CN3-39, 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:

- Software trigger. When selected, a read at BA + 6 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, CN3-39, will stop the pacer clock. The pulse duration should be at least 100 nanoseconds.
- 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.

The next 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.

- 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 + 6.

Triggering a Burst Sample. These triggers, set at Trigger Register bits 10 and 11, BA + 6, can trigger bursts:

- Through software (by reading BA + 6 to initiate a Start Convert)
- Using a pacer clock (internal (Clock TC Counter 0 or 1) or external (CN3-41))
- Using an external trigger (CN3-39)
- Using the digital interrupt

Trigger Repeat Function. Bit 13 in the Trigger Register at BA + 6 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 + 6).

When bit 13 in the Trigger Register, BA + 6, is high, the conversion sequence is repeated each time an external trigger is received. Figure 5-4 shows a timing diagram for this feature.

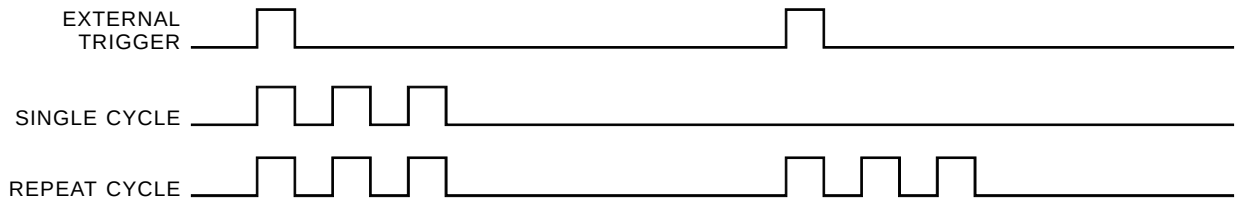


Fig. 5-4 — External Trigger Single Cycle Vs. Repeat Cycle

Pacer Clock Source. The pacer clock can be generated from an internal source (Clock TC Counter 0 or 1) or an external source (CN3-41) by setting bit 9 in the Trigger Register at BA + 6 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 + 6 (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 5-5 shows a timing diagram for single conversions.

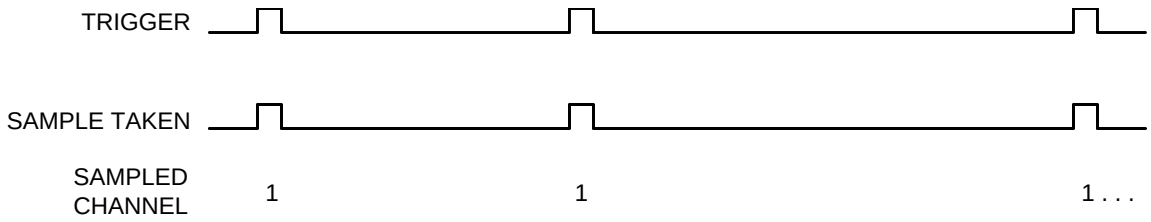


Fig. 5-5 — Timing Diagram, Single Conversion

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 + 6. 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 5-6 shows a timing diagram for multiple conversions.

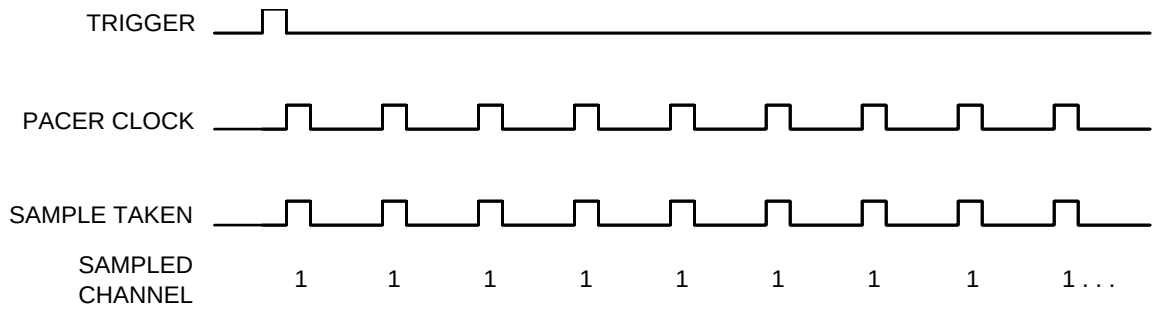


Fig. 5-6 — Timing Diagram, Multiple Conversions

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 + 2 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 + 2 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 5-7 shows a timing diagram for random channel scanning.

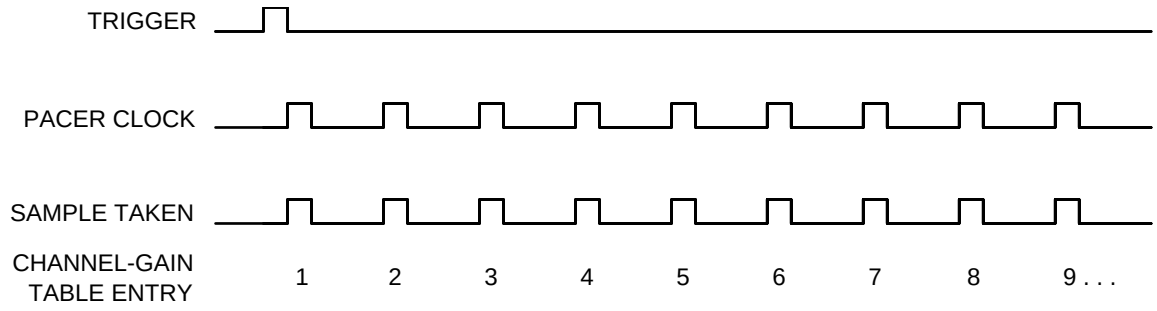


Fig. 5-7 — Timing Diagram, Random Channel Scan

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 + 2 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 + 2 high.

Burst is used when you want one sample from a specified number of channels for each trigger. Figure 5-8 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 Real Time Devices).

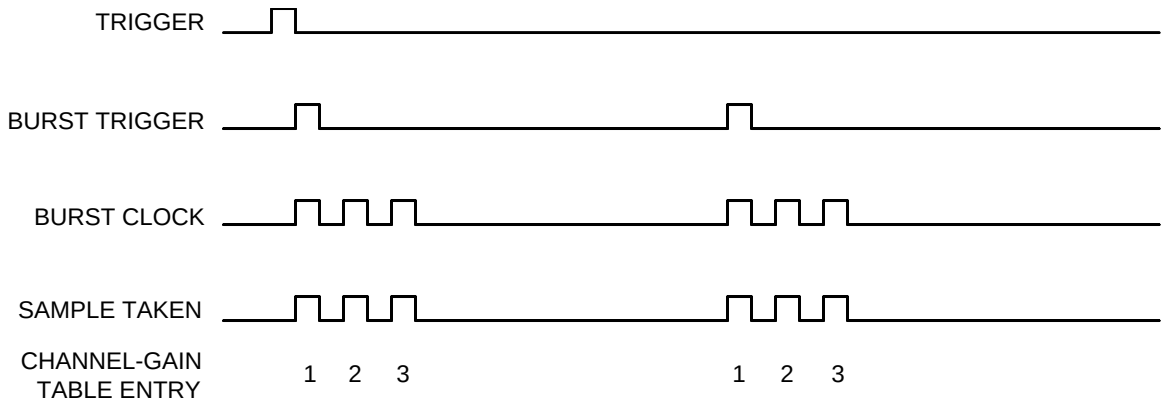


Fig. 5-8 — Timing Diagram, Programmable Burst

Programmable Multiscan. 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 5-9 below, 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.

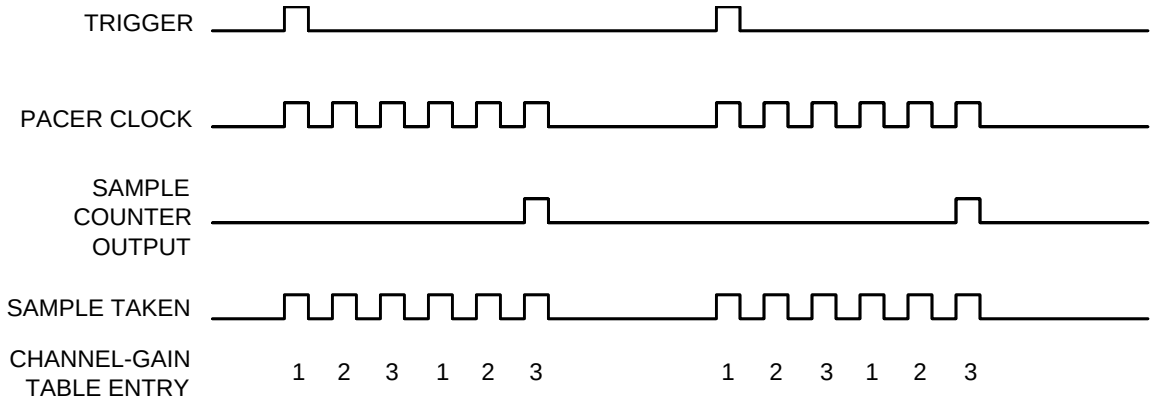


Fig. 5-9 — Timing Diagram, Programmable Multiscan

As you can see, the DM6420 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 + 6) 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 + 6 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 + 6 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 + 6 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 + 2. 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 + 2), is set when the sample buffer is full, disabling the A/D converter. Even if you've 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 + 6) 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. If you want to tag each conversion with its channel-gain table identifier, the channel-gain tag is stored in a 16-bit word in the sample buffer. This section explains how to read the data stored in the sample buffer.

Reading Data with the Channel-gain Data Store Bit Disabled

When the channel-gain data store bit is disabled, 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 volts 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 uV but our input range is decreased from 10 volts to 2.5 volts. The formula would look like this:

Voltage = ((input range / Gain) / 4096) x Conversion Data

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

Voltage = 610 uV x Conversion Data

If we now change the input range to ±10 volts 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.

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
6igQ	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 (±5 volt input range)		
Input Voltage	**Sign**	**Output Code**
+4.998 volts	0	msb 0111 1111 1111 lsb
+2.500 volts	0	msb 0100 0000 0000 lsb
0 volts	0	msb 0000 0000 0000 lsb
-.00244 volts	1	msb 1111 1111 1111 lsb
-5.000 volts	1	msb 1000 0000 0000 lsb
1 lsb = 2.44 millivolts		

A/D Bipolar Code Table (±10 volt input range)		
Input Voltage	**Sign**	**Output Code**
+9.995 volts	0	msb 0111 1111 1111 lsb
+5.000 volts	0	msb 0100 0000 0000 lsb
0 volts	0	msb 0000 0000 0000 lsb
-.00488 volts	1	msb 1111 1111 1111 lsb
-10.000 volts	1	msb 1000 0000 0000 lsb
1 lsb = 4.88 millivolts		

A/D Unipolar Code Table (0 to +10 volt input range)		
Input Voltage	**Sign**	**Output Code**
+9.99756 volts	0	msb 1111 1111 1111 lsb
+5.000 volts	0	msb 1000 0000 0000 lsb
0 volts	0	msb 0000 0000 0000 lsb
1 lsb = 2.44 millivolts		

Reading Data with the Channel-gain Data Store Bit Enabled

When the channel-gain data store bit is enabled, the sample buffer contains two 16-bit words for each 12-bit conversion: the 16-bit channel-gain data word followed by the 16-bit converted data plus data markers word. Figure 5-10 shows how these words are sent to the sample buffer. Below is the format of the 16-bit channel-gain data word.

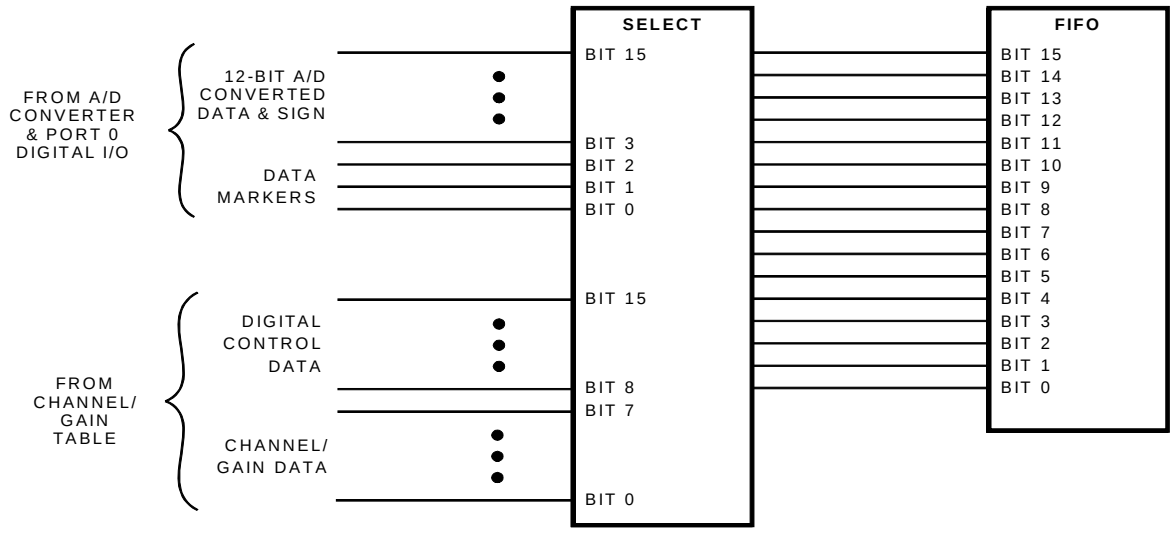
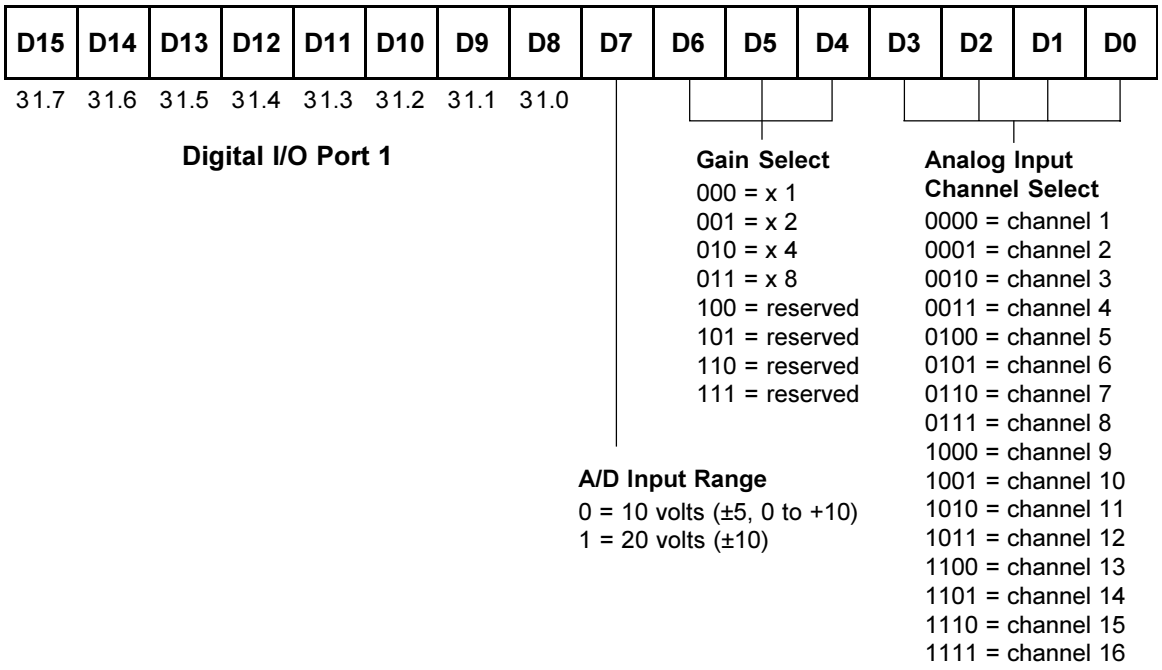


Fig. 5-10 — Sample Buffer Circuitry

The bottom 8 bits contain the channel, gain and input range information. Note that the input range only uses one bit to specify if you are using a 10 volt range (±5 volt or 0 to +10 volt) or if you are using a 20 volt range (±10 volt). The upper 8 bits contain the Digital I/O Port 1 lines. This information is useful if you are using the Digital part of the channel/gain table. If you are not using the Digital part of the table, these bits will contain whatever the bit pattern at Port 1 is.



Remember that when you have the channel-gain data store enabled, each sample in the FIFO will consist of two 16-bit words. The first word will contain the channel-gain information shown above and the second 16-bit word will contain the A/D data.

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 5-11, 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 (CN3-27), P0.6 (CN3-25) and P0.7 (CN3-23).

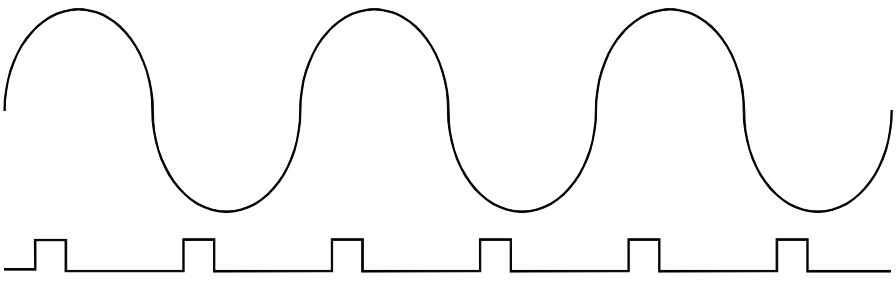


Fig. 5-11 — Storing Digital Data with Analog Data at the Acquisition Rate

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 5-12. 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.

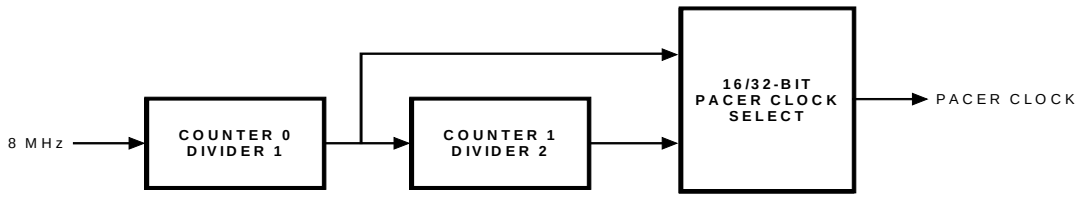


Fig. 5-12 — Pacer Clock Block Diagram

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 + 6. 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 nanoseconds 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 + 2 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 5-12. 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 ---> 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 Hz	2 / (0002)	40000 / (9C40)
10 Hz	16 / (0010)	50000 / (C350)

To set up the 16-bit pacer clock, follow these steps:

1. Set pacer clock size to 16 bits (bit 8 of Trigger Register at BA + 6 = 0).
2. Set BA + 2, 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, follow these steps:

1. Set pacer clock size to 32 bits (bit 8 of Trigger Register at BA + 6 = 1).
2. Set BA + 2, 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 on-board 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 on-board 8 MHz clock source, this equation becomes:

Burst clock frequency = 8 MHz/100 kHz ---> 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, follow these steps:

1. Set BA + 2, 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 DM6420 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 + 14. 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, follow these steps:

2. Set BA + 2, 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 + 14 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 DM6420 provides a bit in the Control Register at BA + 2 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 5-13 shows a timing diagram for this example.

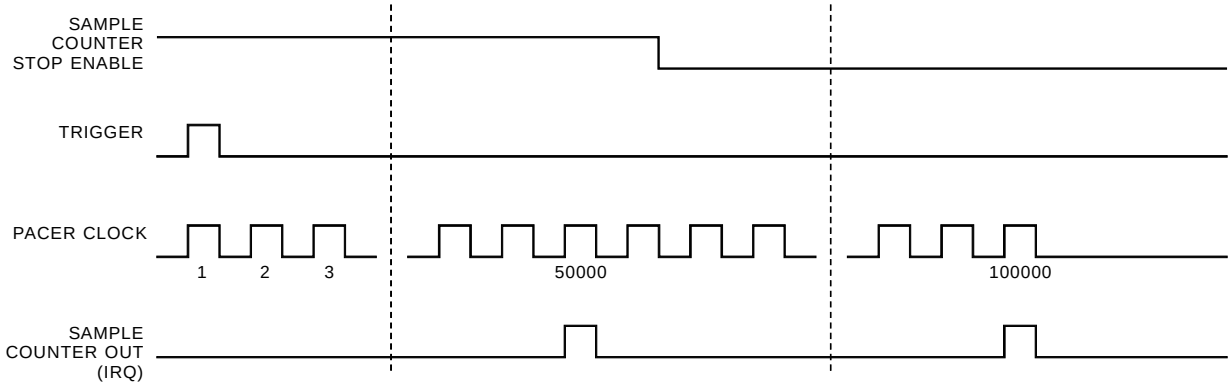


Fig. 5-13 — Timing Diagram for Cycling the Sample Counter

CHAPTER 6

DATA TRANSFERS USING DMA

This chapter explains how data transfers are accomplished using DMA.

Direct Memory Access (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 (DM6420).
7. Enable DMA channel.
8. Wait until DMA is complete.
9. Disable DMA channel.

Each step is detailed in the following paragraphs.

• Choosing a DMA Channel

There are a number of DMA channels available on the PC for use by peripheral devices. The DM6420 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 DM6420 board. 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

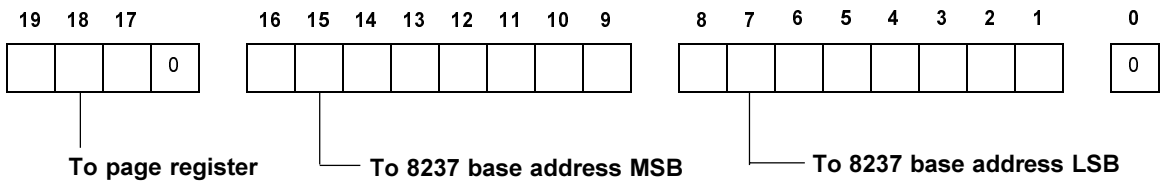
Once you have a buffer into which to place your data, you must inform the 8237 DMA controller of the location of this buffer. This is a little more complex than it sounds because the DMA controller uses a **page**:offset memory scheme, while you are probably used to thinking about your computer's memory 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've 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 */
```

Beware! 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	8B/(139)
6	89/(137)
7	8A/(138)

• The DMA Controller

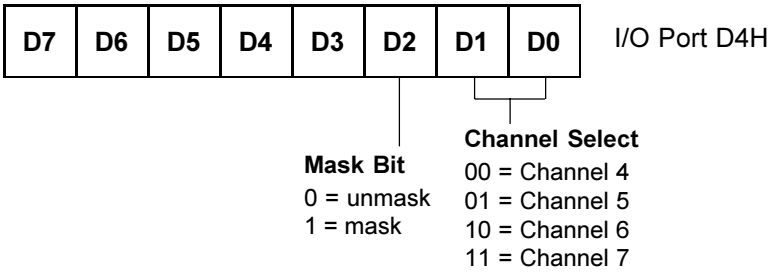
The DMA controller is made up of two complex 8237 chips, one for DMA channels 0-3, and one for channels 4-7, that occupy 32 contiguous bytes of the AT I/O port space starting with port C0H. A complete discussion of how it operates is beyond the scope of this manual; only relevant information is included here. The DMA controller is programmed by writing to the DMA registers in your AT. The table below lists these registers.

' 0 $ 5 HJLVWHUV	
$ GGUHVVHV/(GHFLP DO)	/ RFDWLRQ RI 3DJH5 HJLVWHU
8B/(139)	Channel 5 DMA Page Select
C4/(196)	Channel 5 DMA Base Address
C6/(198)	Channel 5 DMA Count
89/(137)	Channel 6 DMA Page Select
C8/(200)	Channel 6 DMA Base Address
CA/(202)	Channel 6 DMA Count
8A/(138)	Channel 7 DMA Page Select
CC/(204)	Channel 7 DMA Base Address
CE/(206)	Channel 7 DMA Count
D4/(212)	Mask Register
D6/(214)	Mode Register
D8/(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 below.

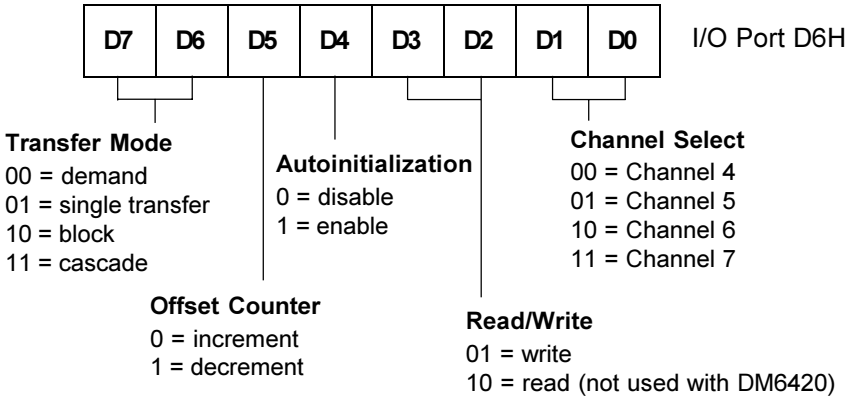
• DMA Mask Register

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 DM6420 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 DM6420. 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. Block mode transfer is not supported by this board.



• Programming the DMA Controller

To program the DMA controller, follow these steps:

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 DM6420 for DMA

Once you have set up the DMA controller, you must program the DM6420 for DMA. The following steps list this procedure:

1. Program Conversion and Trigger mode.
2. Program the DMA channel at BA + 2.
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 DM6420 status register (BA +2). 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 DM6420 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 DM6420, you must setup the first DMA channel at BA + 2, bits 12 and 13 and set up the second DMA channel at BA + 2, 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

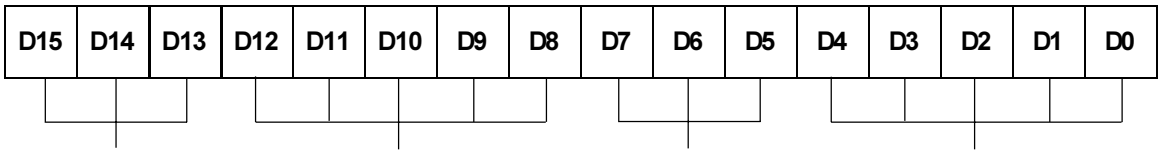
INTERRUPTS

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

The DM6420 has two completely independent interrupt circuits which 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 DM6420 has 16 software selectable interrupt sources which can be programmed in bits 0 through 4 and bits 8 through 12 of the Interrupt Register at BA + 8, as described and shown below.



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 - 1111 = Reserved

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 - 1111 = Reserved

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 generated by the Digital I/O chip occurs (see the section below labeled Advanced Digital Interrupts).
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.

Software Selectable Interrupt Channel

Each interrupt circuit on the DM6420 has 7 software selectable interrupt channels which can be programmed in bits 5 through 7 and bits 13 through 15 of the Interrupt Register at BA + 8. 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 zero's.

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 + 30, bit 3 and enabled at BA + 30, 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 + 28 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 + 28.

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 + 28. 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 + 28.

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 7-1 shows a diagram of this circuit.

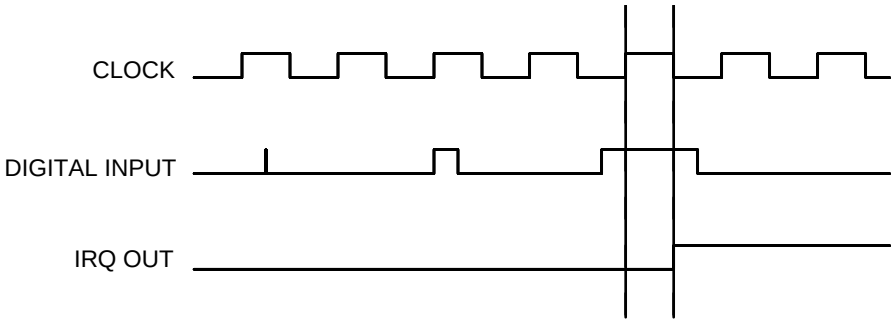


Fig. 7-1 — Digital Interrupt Timing Diagram

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 DM6420 board 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

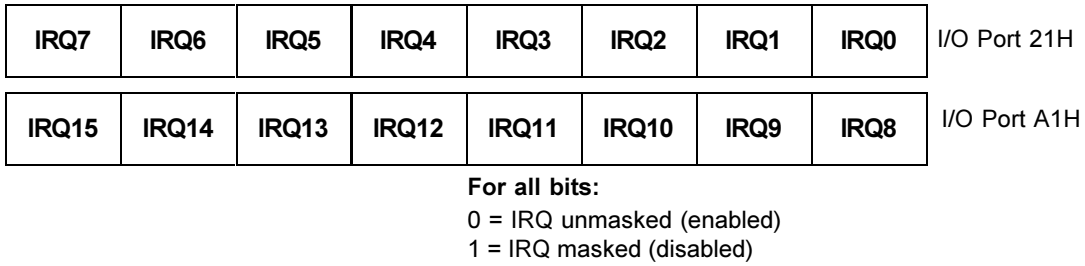
To allow different peripheral devices to generate interrupts on the same computer, the AT bus has 16 different interrupt request (IRQ) lines. A transition from low to high on one of these lines generates an interrupt request which is handled by one of the AT's two interrupt control chips. One chip handles 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.



• 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 Exactly 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 DM6420), 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 DM6420 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 DM6420 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 DM6420 at BA + 0.

CHAPTER 8

D/A CONVERSIONS

This chapter explains how to perform D/A conversions on the DM6420.

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, or 0 to +10 volts. DAC 1 is programmed by writing the 12-bit word to BA + 12. DAC 2 is identical, with the 12-bit word written to BA + 14. 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 (millivolts)		
D/A Bit Weight	**-5 to +5 Volts**	**0 to +5 Volts**	**0 to +10 Volts**
4095 (full-scale)	+4997.56	+4998.78	+9997.56
2048	0000.00	+2500.00	+5000.00
1024	-2500.00	+1250.00	+2500.00
512	-3750.00	+625.00	+1250.00
256	-4375.00	+312.50	+625.00
128	-4687.50	+156.25	+312.50
64	-4843.75	+78.13	+156.25
32	-4921.88	+39.06	+78.13
16	-4960.94	+19.53	+39.06
8	-4980.47	+9.77	+19.53
4	-4990.23	+4.88	+9.77
2	-4995.12	+2.44	+4.88
1	-4997.56	+1.22	+2.44
0	-5000.00	0.00	0.00

CHAPTER 9

TIMER/COUNTERS

This chapter explains the two 8254 timer/counter circuits on the DM6420.

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 on-board pacer clock, described in Chapter 5. The third timer is the burst clock, also discussed in Chapter 5. Figure 9-1 shows the Clock TC circuitry.

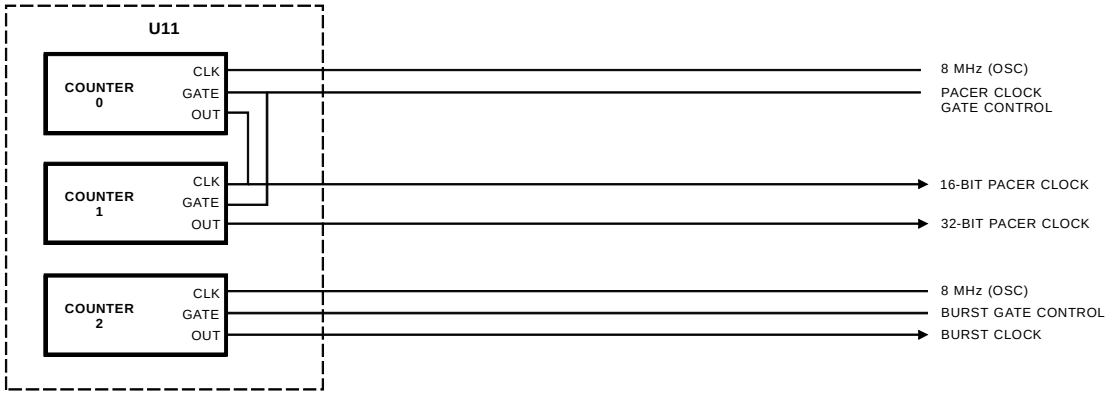


Fig. 9-1 — Clock TC Circuitry

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. Figure 9-2 shows the User TC circuitry.

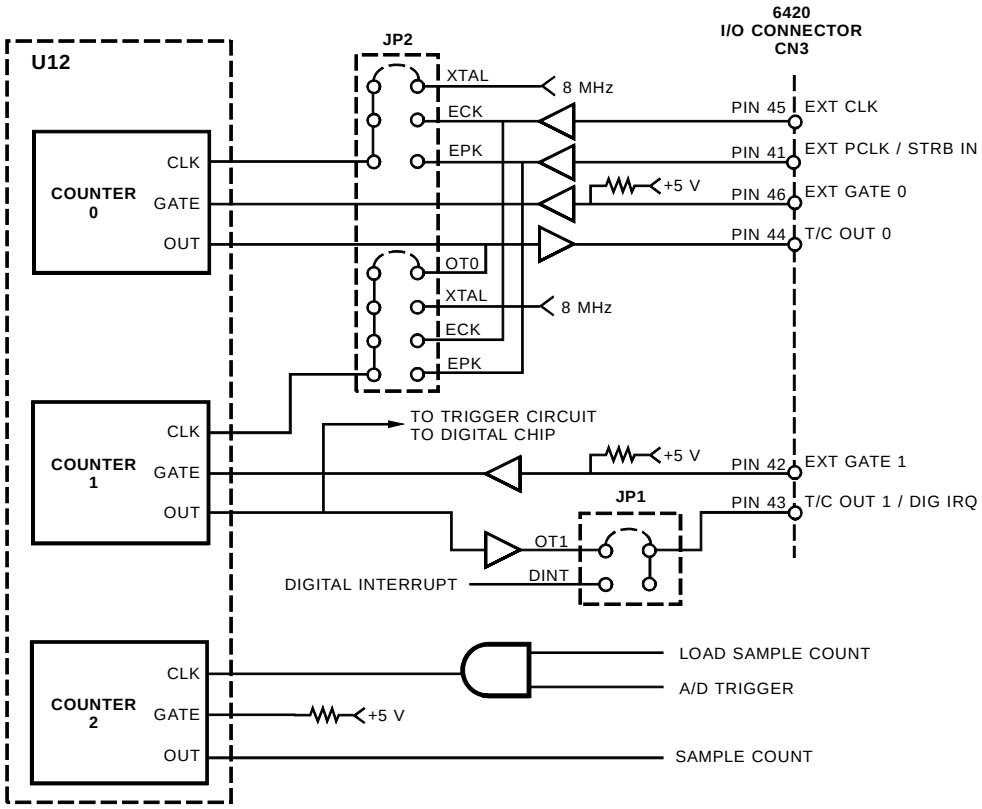


Fig. 9-2 — User TC Circuitry

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 (CN3-43) 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 (CN3-44) 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 DM6420.

The DM6420 has 16 buffered TTL/CMOS digital I/O lines available for digital control applications. These lines are grouped in two 8-bit ports. The eight bits in Port 0 can be independently programmed as input or output. Port 1 can be programmed as an 8-bit input or output port.

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 + 28. The input lines are read and the output lines are written at BA + 24.

Direction Register:

For all bits:
0 = input
1 = output

D7	D6	D5	D4	D3	D2	D1	D0
30.7	30.6	30.5	30.4	30.3	30.2	30.1	30.0

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 + 30, bit 2. These lines are configured as all inputs or all outputs, with their states read and written at BA + 26.

Resetting the Digital Circuitry

When a digital chip clear (BA + 30, bits 1 and 0 = 00 followed by a write to BA + 28), clear board (BA + 0), or reset command is issued, 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 CN3-41. This data can be read from the Compare Register at BA + 28.

High Speed Digital Input

The DM6420 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 + 10. 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 + 10 before you clear the digital input fifo.

CHAPTER 11

EXAMPLE PROGRAMS

This chapter discusses the example programs included with the DM6420.

Included with the DM6420 is a set of example programs that demonstrate the use of many of the module's features. These examples are in written in C and BASIC. Also included is an easy-to-use menu-driven diagnostics program, 6420DIAG, which is especially helpful when you are first checking out your module after installation and when calibrating the module (Chapter 12).

Before using the software included with your module, make a backup copy of the disk. You may make as many backups as you need.

C Programs

These programs are source code files so that you can easily develop your own custom software for your DM6420. All of the programs use the files, DRVR6420.C, DIO5812.C and PCUTILS.C. These files contain all of the routines for setting up the board and acquiring data.

DRVR6420.C contains all the functions needed to control the A/D converter, the D/A converter and the Timer/Counters. These functions are used to set up the conversion type; set the trigger sources; program the pacer, burst and user clocks; read the A/D data and program the D/A outputs.

DIO5812.C contains all the functions needed to control the digital I/O chip. This chip is the same one used on the Real Time Devices' DM5812 module providing two 8-bit ports. Port 0 can have its lines set as input or output on a bit by bit basis. This allows maximum flexibility when connecting your signals. Port 1 is set to be input or output as a group. In addition, Port 0 supports RTD's two Advanced Digital Interrupt modes. An interrupt can be generated when the lines match a programmed value or when any bit changes its current state. A Mask Register lets you monitor selected lines for interrupt generation.

PCUTILS.C contain functions to help program the CPU for interrupts and DMA.

Quick Basic Programs

These programs are source code files so that you can easily develop your own custom software for your DM6420. All of the programs rely on the DRVR6420.LIB and the DRVR6420.QLB library files. These library files contain all of functions needed to interface to the DM6420. Make sure the proper library is loaded when starting Quick Basic by typing QB/L DRVR6420. These libraries were created using Borland C 3.1 and were generated from the files DRVR6420.C and DIO5812.C. Should you need to recompile the libraries, contact the factory for details on this procedure.

CHAPTER 12

CALIBRATION

This chapter tells you how to calibrate the DM6420 using the 6420DIAG diagnostic program included in the example software package and the trimpots on the module. These trimpots calibrate the A/D converter gain and offset.

This chapter tells you how to 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 below, and make adjusts as necessary. Using the 6420DIAG diagnostics program is a convenient way to monitor conversions while you calibrate the module.

Calibration is done with the module installed in your system. You can access the trimpots at the edge of the module. Power up the system and let the board circuitry stabilize for 15 minutes before you start calibrating.

Required Equipment

The following equipment is required for calibration:

- Precision Voltage Source: -10 to +10 volts
- Digital Voltmeter: 5-1/2 digits
- Small Screwdriver (for trimpot adjustment)

While not required, the 6420DIAG diagnostics program (included with example software) is helpful when performing calibrations. Figure 12-1 shows the module layout with the trimpots located along the top edge.

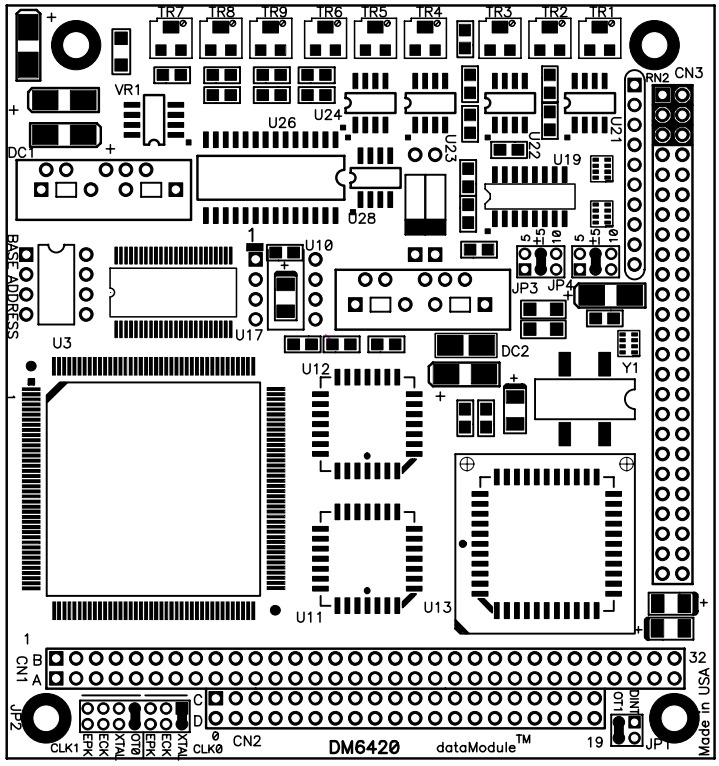


Fig. 12-1 — Module Layout

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 volts), and the second procedure calibrates the unipolar range (0 to +10 volts). Table 12-1 shows the ideal input voltage for each bit weight for the bipolar ranges, and Table 12-2 shows the ideal voltage for each bit weight for the unipolar range.

Bipolar Calibration

• Bipolar Range Adjustments: -5 to +5 Volts

Two adjustments are made to calibrate the A/D converter for the bipolar range of -5 to +5 volts. 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 volts.

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 millivolts, 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 volts, and repeat the procedure, this time adjusting TR5 until the data flickers between the values in the table.

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

• Bipolar Range Adjustments: -10 to +10 Volts

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

Data Value for Calibrating Bipolar 20 Volt Range (-10 to +10 volts)	
	TR2 Input Voltage = +5.0000V
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 12-1 A/D Converter Bit Weights, Bipolar			
		Ideal Input Voltage (millivolts)	
SIGN	A/D Bit Weight	-5 to +5 Volts	-10 to +10 Volts
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 volts. Trimpot TR6 is used to make the offset adjustment. This calibration procedure is performed with the module programmed for a 0 to +10 volt 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 millivolts, 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 Volt Range (0 to +10 volts)	
	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 12-2 A/D Converter Bit Weights, Unipolar		
		Ideal Input Voltage (millivolts)
SIGN	A/D Bit Weight	0 to +10 Volts
0	1111 1111 1111	+9997.6
0	1000 0000 0000	+5000.0
0	0100 0000 0000	+2500.0
0	0010 0000 0000	+1250.0
0	0001 0000 0000	+625.00
0	0000 1000 0000	+312.50
0	0000 0100 0000	+156.25
0	0000 0010 0000	+78.125
0	0000 0001 0000	+39.063
0	0000 0000 1000	+19.531
0	0000 0000 0100	+9.7656
0	0000 0000 0010	+4.8828
0	0000 0000 0001	+2.4414
0	0000 0000 0000	0

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 (millivolts)		
D/A Bit Weight	-5 to +5 Volts	0 to +5 Volts	0 to +10 Volts
4095 (full-scale)	+4997.56	+4998.78	+9997.56
2048	0000.00	+2500.00	+5000.00
1024	-2500.00	+1250.00	+2500.00
512	-3750.00	+625.00	+1250.00
256	-4375.00	+312.50	+625.00
128	-4687.50	+156.25	+312.50
64	-4843.75	+78.13	+156.25
32	-4921.88	+39.06	+78.13
16	-4960.94	+19.53	+39.06
8	-4980.47	+9.77	+19.53
4	-4990.23	+4.88	+9.77
2	-4995.12	+2.44	+4.88
1	-4997.56	+1.22	+2.44
0	-5000.00	0.00	0.00

APPENDIX A

DM6420HR SPECIFICATIONS

DM6420HR Characteristics Typical @ 2R° C

Interface

Switch-selectable base address, I/O mapped
Software programmable interrupts & DMA channel

Analog Input

Up to 8 differential or 16 single-ended inputs, software selectable
Input impedance, each channel ..... >10 megohms
Gains, software-selectable ..... 1, 2, 4, & 8
Gain error ..... .05B , typ; 0.1B , max
Input ranges, software selectable ..... ±5, ±10, or 0 to +10 volts
Overvoltage protection ..... ±12 Vdc
Common mode input voltage ..... ±10 volts, max
Settling time (gain = 1) ..... 2 µsec, max

A/D Converter

Type ..... Successive approximation
Resolution ..... 12 bits (2.44 mV @ 10V; 4.88 mV @ 20V)
i inearity ..... ±1 bit, typ
Conversion speed ..... 2 µsec, typ
Module throughput ..... 500 kHz

Channel-gain Table

Size ..... 1024 x 16 bits

Pacer Clock & Sample Counter

Range (using on-board 8 MHz clock) ..... 9 minutes to 2 µsec
Sample counter maximum count (1 cycle) 65,536

Digital I/O

Number of lines ..... 8 bit programmable & 8 port programmable
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 AD7237

Analog outputs ..... 2 channels
Resolution ..... 12 bits
Output ranges ..... 0 to +5, ±5, or 0 to +10 volts
Relative accuracy ..... ±1 bit, max
Full-scale accuracy ..... ±5 bits, max
Non-linearity ..... ±1 bit, max
Settling time ..... 5 µsec, max

Timer/Counters ..... CMOS 82C54

Six 16-bit down counters
6 programmable operating modes
Counter input source ..... External clock (8 MHz, max) or on-board 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 volts, ±12 volts, ground

Power Requirements

DM6420: +5 volts, 2.5t typ.

CN3 Connector

50-pin right angle header

Environmental

Operating temperature ..... -40 to +85°C
Storage temperature ..... -55 to +125°C
Humidity ..... 0 to 90B non-condensing

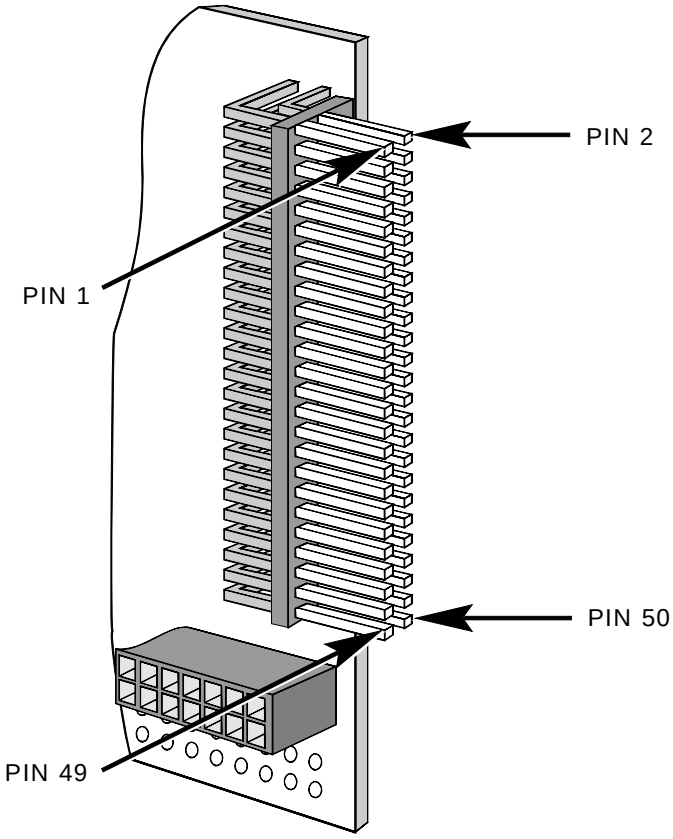
Size

3.55"i x 3.775"t x 0.6"H (90mm x 96mm x 16mm)

APPENDIX B

CN3 CONNECTOR PIN ASSIGNMENTS

DIFF.	S.E.			DIFF.	S.E.
AIN1+	AIN1	1	2	AIN1-	AIN9
AIN2+	AIN2	3	4	AIN2-	AIN10
AIN3+	AIN3	5	6	AIN3-	AIN11
AIN4+	AIN4	7	8	AIN4-	AIN12
AIN4+	AIN5	9	10	AIN5-	AIN13
AIN6+	AIN6	11	12	AIN6-	AIN14
AIN7+	AIN7	13	14	AIN7-	AIN15
AIN8+	AIN8	15	16	AIN8-	AIN16
	AOUT 1	17	18	ANALOG GND	
	AOUT 2	19	20	ANALOG GND	
	ANALOG GND	21	22	ANALOG GND	
	DATAMARKER 3 / P0.7	23	24	P1.7	
	DATAMARKER 2 / P0.6	25	26	P1.6	
	DATAMARKER 1 / P0.5	27	28	P1.5	
	P0.4	29	30	P1.4	
	P0.3	31	32	P1.3	
	P0.2	33	34	P1.2	
	P0.1	35	36	P1.1	
	P0.0	37	38	P1.0	
	TRIGGER IN	39	40	DIGITAL GND	
	EXT PCLK / STRB IN	41	42	EXT GATE 1	
	T/C OUT 1 / DIG IRQ	43	44	T/C OUT 0	
	EXT CLK	45	46	EXT GATE 0	
	+12 VOLTS	47	48	+5 VOLTS	
	-12 VOLTS	49	50	DIGITAL GND	



NOTE:
On the DM6420, +12 volts at pin 47 and -12 volts at pin 49 are available only if supplied by the computer bus.

CN3 Mating Connector Part Numbers	
Manufacturer	**Part Number**
AMP	1-746094-0
3M	3425-7650

APPENDIX C

COMPONENT DATA SHEETS

Intel 82C54 Programmable Interval Timer
Data Sheet Reprint

APPENDIX D WARRANTY AND RETURN POLICY

Return Policy

If you wish to return a product to the factory for service, please follow this procedure:

Read the Limited Warranty to familiarize yourself with our warranty policy.

Contact the factory for a Return Merchandise Authorization (RMA) number.

Please have the following available:

- Complete board name
- Board serial number
- A detailed description of the board's behavior

List the name of a contact person, familiar with technical details of the problem or situation, **along with their phone and fax numbers, address, and e-mail address** (if available).

List your shipping address!!

Indicate the shipping method you would like used to return the product to you.
We will not ship by next-day service without your pre-approval.

Carefully package the product, using proper anti-static packaging.

Write the RMA number in large (1") letters on the outside of the package.

Return the package to:

RTD Embedded Technologies, Inc.

103 Innovation Blvd.

State College PA 16803-0906
USA

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 an RMA number.

THIS LIMITED WARRANTY DOES NOT EXTEND TO ANY PRODUCTS WHICH HAVE BEEN DAMAGED AS A RESULT OF ACCIDENT, MISUSE, ABUSE (such as: use of incorrect input voltages, improper or insufficient ventilation, failure to follow the operating instructions that are provided by RTD Embedded Technologies, “acts of God” or other contingencies beyond the control of RTD Embedded Technologies), OR AS A RESULT OF SERVICE OR MODIFICATION BY ANYONE OTHER THAN RTD Embedded Technologies. EXCEPT AS EXPRESSLY SET FORTH ABOVE, NO OTHER WARRANTIES ARE EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, AND RTD Embedded Technologies EXPRESSLY DISCLAIMS ALL WARRANTIES NOT STATED HEREIN. ALL IMPLIED WARRANTIES, INCLUDING IMPLIED WARRANTIES FOR MECHANTABILITY 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
Our website: www.rtd.com

DM6420 User Settings	
Base I/O Address:	
(hex)	(decimal)
IRQ Channel:	
DMA Channel:	

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP