

# **AT-MIO E Series**

## **Register-Level Programmer Manual**

*Multifunction I/O Boards for the PC AT*

**April 1995 Edition**

**Part Number 340747B-01**

**© Copyright 1995 National Instruments Corporation.  
All Rights Reserved.**

**National Instruments Corporate Headquarters**

6504 Bridge Point Parkway

Austin, TX 78730-5039

(512) 794-0100

Technical support fax: (800) 328-2203

(512) 794-5678

**Branch Offices:**

Australia (03) 879 9422, Austria (0662) 435986, Belgium 02/757.00.20, Canada (Ontario) (519) 622-9310,

Canada (Québec) (514) 694-8521, Denmark 45 76 26 00, Finland (90) 527 2321, France (1) 48 14 24 24,

Germany 089/741 31 30, Italy 02/48301892, Japan (03) 3788-1921, Mexico 95 800 010 0793,

Netherlands 03480-33466, Norway 32-84 84 00, Singapore 2265886, Spain (91) 640 0085, Sweden 08-730 49 70,

Switzerland 056/20 51 51, Taiwan 02 377 1200, U.K. 0635 523545

## Limited Warranty

The AT-MIO E Series boards are warranted against defects in materials and workmanship for a period of one year from the date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace equipment that proves to be defective during the warranty period. This warranty includes parts and labor.

The media on which you receive National Instruments software are warranted not to fail to execute programming instructions, due to defects in materials and workmanship, for a period of 90 days from date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace software media that do not execute programming instructions if National Instruments receives notice of such defects during the warranty period. National Instruments does not warrant that the operation of the software shall be uninterrupted or error free.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. National Instruments will pay the shipping costs of returning to the owner parts which are covered by warranty.

National Instruments believes that the information in this manual is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, National Instruments reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult National Instruments if errors are suspected. In no event shall National Instruments be liable for any damages arising out of or related to this document or the information contained in it.

EXCEPT AS SPECIFIED HEREIN, NATIONAL INSTRUMENTS MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF NATIONAL INSTRUMENTS SHALL BE LIMITED TO THE AMOUNT THEREFORE PAID BY THE CUSTOMER. NATIONAL INSTRUMENTS WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEREOF. This limitation of the liability of National Instruments will apply regardless of the form of action, whether in contract or tort, including negligence. Any action against National Instruments must be brought within one year after the cause of action accrues. National Instruments shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failures caused by owner's failure to follow the National Instruments installation, operation, or maintenance instructions; owner's modification of the product; owner's abuse, misuse, or negligent acts; and power failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

## Copyright

Under the copyright laws, this publication may not be reproduced or transmitted in any form, electronic or mechanical, including photocopying, recording, storing in an information retrieval system, or translating, in whole or in part, without the prior written consent of National Instruments Corporation.

## Trademarks

LabVIEW®, NI-DAQ®, RTSI®, DAQ-PnP™, DAQ-STC™, and NI-PGIA™ are trademarks of National Instruments Corporation.

Product and company names listed are trademarks or trade names of their respective companies.

## **WARNING REGARDING MEDICAL AND CLINICAL USE OF NATIONAL INSTRUMENTS PRODUCTS**

National Instruments products are not designed with components and testing intended to ensure a level of reliability suitable for use in treatment and diagnosis of humans. Applications of National Instruments products involving medical or clinical treatment can create a potential for accidental injury caused by product failure, or by errors on the part of the user or application designer. Any use or application of National Instruments products for or involving medical or clinical treatment must be performed by properly trained and qualified medical personnel, and all traditional medical safeguards, equipment, and procedures that are appropriate in the particular situation to prevent serious injury or death should always continue to be used when National Instruments products are being used. National Instruments products are NOT intended to be a substitute for any form of established process, procedure, or equipment used to monitor or safeguard human health and safety in medical or clinical treatment.

# Contents

---

<b>About This Manual</b> .....	ix
Organization of This Manual .....	ix
Conventions Used in This Manual.....	x
Related Documentation.....	xi
Customer Communication .....	xi
<b>Chapter 1</b>	
<b>General Description</b> .....	1-1
General Characteristics .....	1-1
<b>Chapter 2</b>	
<b>Theory of Operation</b> .....	2-1
Functional Overview .....	2-1
ISA Bus Channel Interface Circuitry .....	2-4
Analog Input and Timing Circuitry .....	2-6
Analog Input Circuitry .....	2-6
Data Acquisition Timing Circuitry .....	2-8
Single-Read Timing.....	2-8
Data Acquisition Sequence Timing .....	2-9
Multirate Scanning without Using Ghost.....	2-10
Multirate Scanning Using Ghost.....	2-12
Posttrigger and Pretrigger Acquisition.....	2-14
Analog Triggering.....	2-15
Analog Output and Timing Circuitry.....	2-15
Analog Output Circuitry.....	2-16
Analog Output Timing Circuitry.....	2-17
Single-Point Output.....	2-17
Waveform Generation.....	2-18
Digital I/O Circuitry.....	2-19
Timing I/O Circuitry .....	2-19
RTSI Bus Interface Circuitry .....	2-20
<b>Chapter 3</b>	
<b>Register Map and Descriptions</b> .....	3-1
Register Map .....	3-1
Register Sizes .....	3-3
Register Descriptions .....	3-3
Misc Register Group .....	3-3
Serial Command Register .....	3-4
Misc Command Register .....	3-5
Status Register .....	3-6
Analog Input Register Group .....	3-6
ADC FIFO Data Register.....	3-7
Configuration Memory Low Register .....	3-8
Configuration Memory High Register .....	3-10

Analog Output Register Group .....	3-13
AO Configuration Register .....	3-14
DAC FIFO Data Register .....	3-15
DAC0 Direct Data Register .....	3-16
DAC1 Direct Data Register .....	3-17
DMA Control Register Group.....	3-18
Strobes Register .....	3-18
Channel A Mode Register .....	3-19
Channel B Mode Register .....	3-20
Channel C Mode Register.....	3-21
AI AO Select Register.....	3-22
G0 G1 Select Register.....	3-23
DIO Register Group.....	3-24
DAQ-STC Register Group.....	3-24
FIFO Strobe Register Group .....	3-24
Configuration Memory Clear Register .....	3-24
ADC FIFO Clear Register .....	3-24
DAC FIFO Clear Register.....	3-25

## Chapter 4

<b>Programming</b> .....	4-1
Plug and Play Initialization.....	4-1
Windowing Registers .....	4-3
Programming Examples.....	4-3
Digital I/O .....	4-4
Analog Input .....	4-5
Analog Output.....	4-28
General-Purpose Counter/Timer.....	4-41
RTSI Trigger Lines Programming Considerations .....	4-42
Analog Triggering.....	4-42
Interrupt Programming.....	4-44
DMA Programming .....	4-44
Single-Channel versus Dual-Channel DMA.....	4-46

## Chapter 5

<b>Calibration</b> .....	5-1
EEPROM .....	5-1
Calibration DACs.....	5-5
NI-DAQ Calibration Function .....	5-6

## Appendix A

<b>OKI MSM82C55A Data Sheet</b> .....	A-1
---------------------------------------	-----

## Appendix B

<b>Customer Support and Communication</b> .....	B-1
---	-----

<b>Glossary</b> .....	G-1
-----------------------	-----

<b>Index</b> .....	Index-1
--------------------	---------

## Figures

Figure 2-1.	AT-MIO-16E-1, AT-MIO-16E-2, and AT-MIO-64E-3 Block Diagram.....	2-1
Figure 2-2.	AT-MIO-16E-10 and AT-MIO-16DE-10 Block Diagram .....	2-2
Figure 2-3.	AT-MIO-16XE-50 Block Diagram.....	2-3
Figure 2-4.	ISA Bus Interface Circuitry Block Diagram.....	2-4
Figure 2-5.	Analog Input and Data Acquisition Circuitry Block Diagram .....	2-6
Figure 2-6.	ADC Timing .....	2-8
Figure 2-7.	Timing of Scan in Example 1 .....	2-10
Figure 2-8.	Multirate Scanning of Two Channels .....	2-11
Figure 2-9.	Multirate Scanning of Two Channels with 1:x Sampling Rate.....	2-11
Figure 2-10.	Multirate Scanning of Two Channels with 3:1:1 Sampling Rate .....	2-12
Figure 2-11.	Multirate Scanning of Three Channels with 4:2:1 Sampling Rate .....	2-12
Figure 2-12.	Multirate Scanning without Ghost .....	2-13
Figure 2-13.	Occurrences of Conversion on Channel 1 in Example 3 .....	2-13
Figure 2-14.	Successive Scans Using Ghost.....	2-13
Figure 2-15.	Analog Output Circuitry Block Diagram.....	2-16
Figure 2-16.	DAQ-STC Counter Diagram .....	2-19
Figure 2-17.	RTSI Bus Interface Circuitry Block Diagram .....	2-20
Figure 4-1.	Analog Trigger Structure .....	4-43
Figure 4-2.	DMA Structure.....	4-45
Figure 5-1.	EEPROM Read Timing .....	5-2
Figure 5-2.	Calibration DAC Write Timing .....	5-6

## Tables

Table 1-1.	Features of the AT-MIO E Series Boards.....	1-2
Table 2-1.	Analog Input Configuration Memory .....	2-14
Table 3-1.	AT-MIO E Series Register Map .....	3-1
Table 3-2.	AT-MIO E Series Windowed Register Map.....	3-2
Table 3-3.	PGIA Gain Selection.....	3-9
Table 3-4.	Calibration Channel Assignments.....	3-11
Table 3-5.	Differential Channel Assignments.....	3-11
Table 3-6.	Nonreferenced Single-Ended Channel Assignments .....	3-12
Table 3-7.	Referenced Single-Ended Channel Assignments.....	3-12
Table 3-8.	Channel Assignments.....	3-13
Table 5-1.	AT-MIO-16E-1, AT-MIO-16E-2, and AT-MIO-64E-3 EEPROM Map.....	5-2
Table 5-2.	AT-MIO-16E-10 and AT-MIO-16DE-10 EEPROM Map .....	5-3
Table 5-3.	AT-MIO-16XE-50 EEPROM Map.....	5-4

# About This Manual

---

This manual describes the registers and register map of the AT-MIO E Series boards and contains information concerning their register-level programming.

The DAQ-STC, a National Instruments system timing controller ASIC, is the timing engine that drives the AT-MIO E Series boards. Consequently, the timing and programming sections in this manual repeat certain information from, or draw your attention to, sections in the *DAQ-STC Technical Reference Manual*. You must use your register-level programmer manual along with the *DAQ-STC Technical Reference Manual* for a complete understanding of AT-MIO E Series board programming.

Unless otherwise noted, text applies to all boards in the AT-MIO E Series. The AT-MIO E Series boards are:

- AT-MIO-16E-1
- AT-MIO-16E-2
- AT-MIO-64E-3
- AT-MIO-16E-10
- AT-MIO-16DE-10
- AT-MIO-16XE-50

The AT-MIO E Series boards are high-performance multifunction analog, digital, and timing I/O boards for the IBM PC AT series computers. Supported functions include analog input, analog output, digital I/O, and timing I/O.

## Organization of This Manual

The *AT-MIO E Series Register-Level Programmer Manual* is organized as follows:

- Chapter 1, *General Description*, describes the general characteristics of the AT-MIO E Series boards.
- Chapter 2, *Theory of Operation*, contains a functional overview of the AT-MIO E Series board and explains the operation of each functional unit making up the AT-MIO E Series boards.
- Chapter 3, *Register Map and Descriptions*, describes in detail the address and function of each of the AT-MIO E Series control and status registers.
- Chapter 4, *Programming*, contains programming instructions for operating the circuitry on the AT-MIO E Series board.



- Chapter 5, *Calibration*, explains how to calibrate the analog input and output sections of the AT-MIO E Series boards by reading calibration constants from the EEPROM and writing them to the calibration DACs.
- Appendix A, *OKI MSM82C55A Data Sheet*, contains a manufacturer data sheet for the MSM82C55A CMOS programmable peripheral interface (OKI Semiconductor).
- Appendix B, *Customer Support and Communication*, contains forms for you to complete to help you communicate with National Instruments about our products.
- The *Glossary* contains an alphabetical list and description of terms used in this manual, including acronyms, abbreviations, metric prefixes, mnemonics, and symbols.
- The *Index* alphabetically lists topics covered in this manual, including the page where you can find the topic.

## Conventions Used in This Manual

The following conventions are used in this manual:

<b><i>bold italic</i></b>	Bold italic text denotes a note, caution, or warning.
<i>italic</i>	Italic text denotes emphasis, a cross reference, or an introduction to a key concept.
monospace	Lowercase text in this font denotes text or characters that are to be literally input from the keyboard, sections of code, programming examples, and syntax examples. This font is also used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, variables, filenames, and extensions, and for statements and comments taken from program code.
PC	PC refers to the IBM PC AT and compatible computers.
< >	Angle brackets containing numbers separated by an ellipsis represent a range of values associated with a bit, port, or signal name (for example, ACH<0..7> stands for ACH0 through ACH7).

Abbreviations, acronyms, metric prefixes, mnemonics, symbols, and terms are listed in the *Glossary*.

## Related Documentation

You may find the following National Instruments documents helpful for programming interrupts and DMA:

- *Programming Interrupts for Data Acquisition on 80x86-Based Computers, Application Note 010*
- *Programming DMA on PC/XT/AT Computers, Application Note 029*

The following National Instruments manuals contain general information and operating instructions for the AT-MIO E Series boards:

- *AT-MIO E Series User Manual*
- *DAQ-STC Technical Reference Manual*

## Customer Communication

National Instruments wants to receive your comments on our products and manuals. We are interested in the applications you develop with our products, and we want to help if you have problems with them. To make it easy for you to contact us, this manual contains a comment form for you to complete. This form is in Appendix B, *Customer Communication*, at the end of this manual.

# Chapter 1

## General Description

---

This chapter describes the general characteristics of the AT-MIO E Series boards.

### General Characteristics

The AT-MIO E Series boards are completely Plug and Play-compatible multifunction analog, digital, and timing I/O boards for the PC AT and compatible computers. This family of boards features 12-bit and 16-bit ADCs with 16 and 64 analog inputs, 12-bit DACs with voltage outputs, eight and 32 lines of TTL-compatible digital I/O, and two 24-bit counter/timers for timing I/O. Because the AT-MIO E Series boards have no DIP switches, jumpers, or potentiometers, they are easily configured and calibrated using software.

The AT-MIO E Series boards are the first completely switchless and jumperless DAQ boards. This feature is made possible by the National Instruments DAQ-PnP bus interface chip to connect the board to the AT I/O bus. The DAQ-PnP implements the Plug and Play ISA Specification so that the DMA, interrupts, and base I/O addresses are all software configurable. This allows you to easily change the AT-MIO E Series board configuration without having to remove the board from your computer.

The AT-MIO E Series boards use the National Instruments DAQ-STC system timing controller for time-related functions. The DAQ-STC consists of three timing groups that control analog input, analog output, and general-purpose counter/timer functions. These groups include a total of seven 24-bit and three 16-bit counters and a maximum timing resolution of 50 ns.

A common problem with DAQ boards is that you cannot easily synchronize several measurement functions to a common trigger or timing event. The AT-MIO E Series boards have the Real-Time System Integration (RTSI) bus to solve this problem. The RTSI bus consists of our RTSI bus interface and a ribbon cable to route timing and trigger signals between several functions on up to five DAQ boards in your PC.

The AT-MIO E Series boards can interface to an SCXI system so that you can acquire over 3,000 analog signals from thermocouples, RTDs, strain gauges, voltage sources, and current sources. You can also acquire or generate digital signals for communication and control. SCXI is the instrumentation front end for plug-in DAQ boards.

Refer to Table 1-1 for a listing of the various boards in the AT-MIO E Series and their distinguishing features.

Table 1-1. Features of the AT-MIO E Series Boards

Board	Features
AT-MIO-16E-1	16 single-ended or eight differential 12-bit analog inputs, 1,000 kS/s, two 12-bit analog outputs, eight digital I/O, analog and digital triggering
AT-MIO-16E-2	16 single-ended or eight differential 12-bit analog inputs, 500 kS/s, two 12-bit analog outputs, eight digital I/O, analog and digital triggering
AT-MIO-64E-3	64 single-ended or 32 differential 12-bit analog inputs, 333 kS/s, two 12-bit analog outputs, eight digital I/O, analog and digital triggering
AT-MIO-16E-10	16 single-ended or eight differential 12-bit analog inputs, 100 kS/s, two 12-bit analog outputs, eight digital I/O, digital triggering
AT-MIO-16DE-10	16 single-ended or eight differential 12-bit analog inputs, 100 kS/s, two 12-bit analog outputs, 32 digital I/O, digital triggering
AT-MIO-16XE-50	16 single-ended or eight differential 16-bit analog inputs, 20 kS/s, two 12-bit analog outputs, eight digital I/O, digital triggering

Your AT-MIO E Series board is completely software configurable. Refer to your *AT-MIO E Series User Manual* if you have not already installed and configured your board.

# Chapter 2

## Theory of Operation

This chapter contains a functional overview of the AT-MIO E Series boards and explains the operation of each functional unit making up the AT-MIO E Series boards.

### Functional Overview

The block diagram in Figures 2-1 through 2-3 give a functional overview of each AT-MIO E Series board.

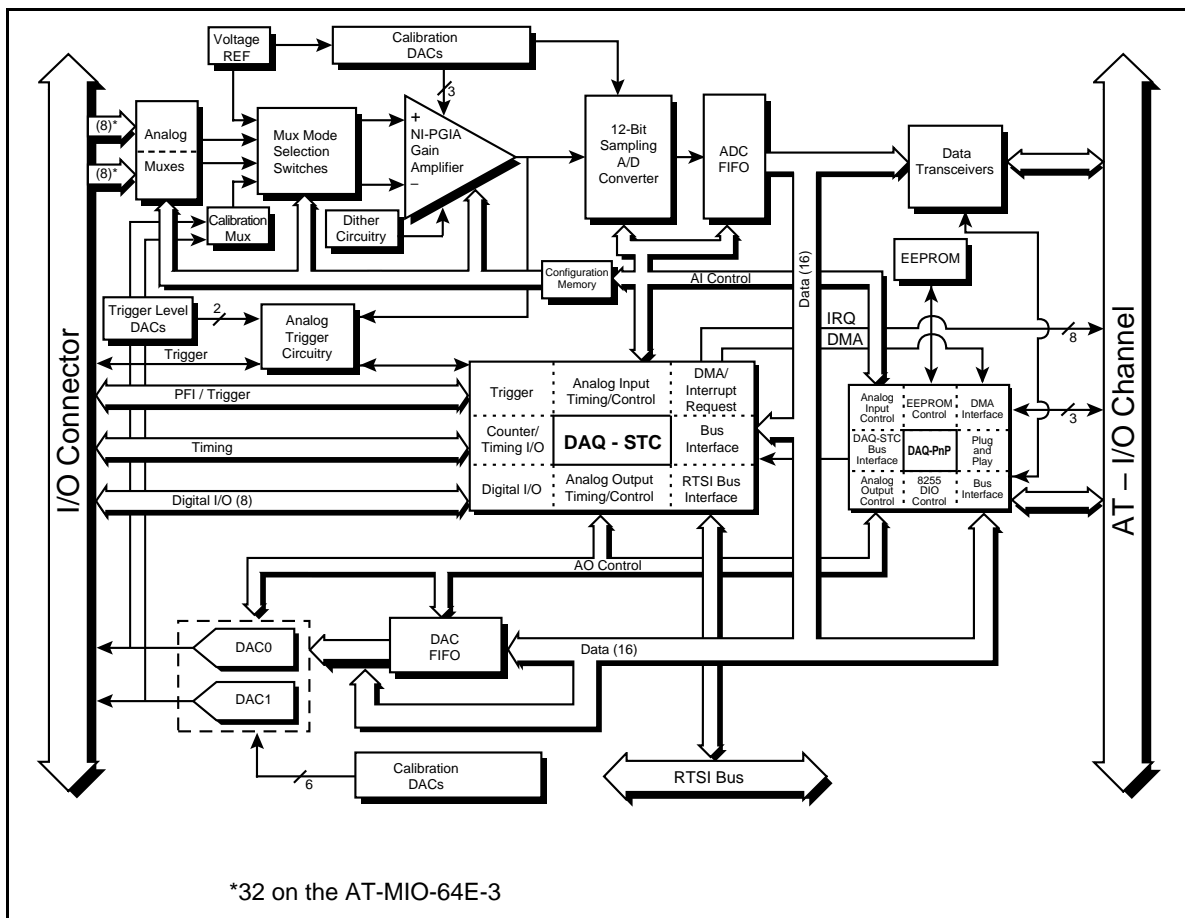


Figure 2-1. AT-MIO-16E-1, AT-MIO-16E-2, and AT-MIO-64E-3 Block Diagram

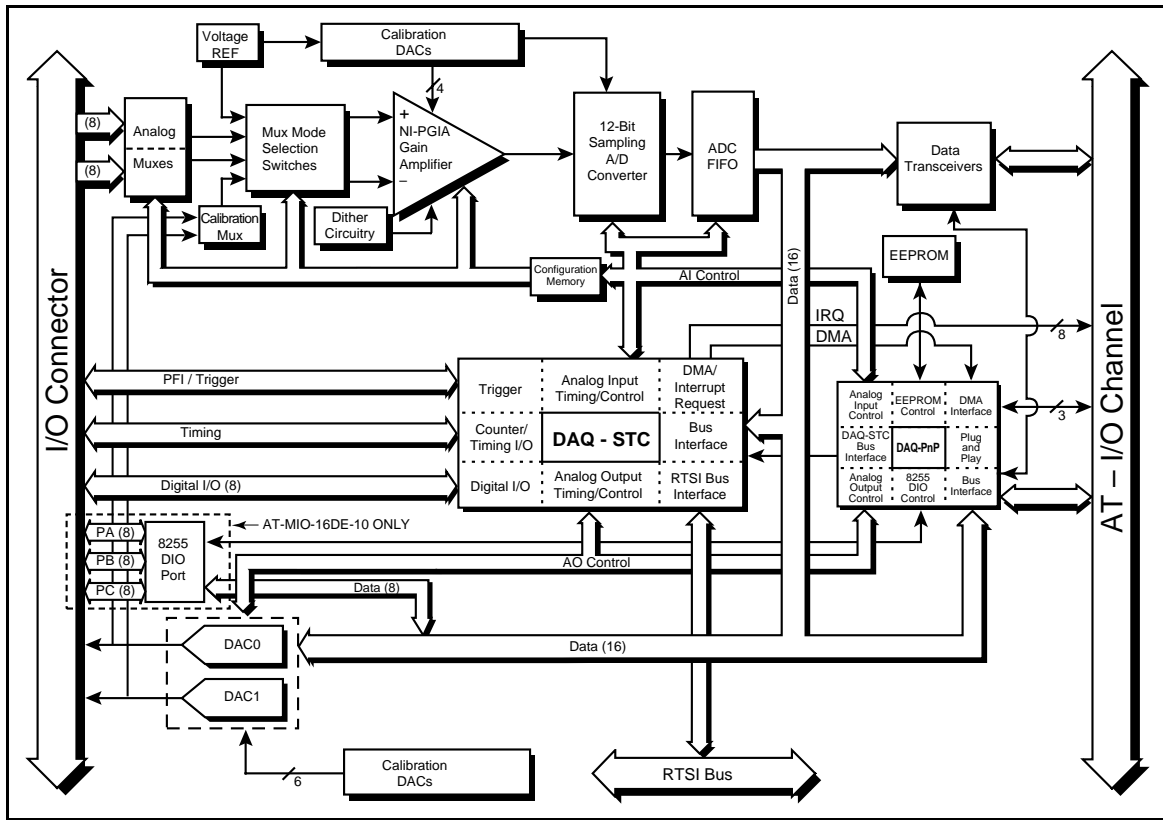


Figure 2-2. AT-MIO-16E-10 and AT-MIO-16DE-10 Block Diagram

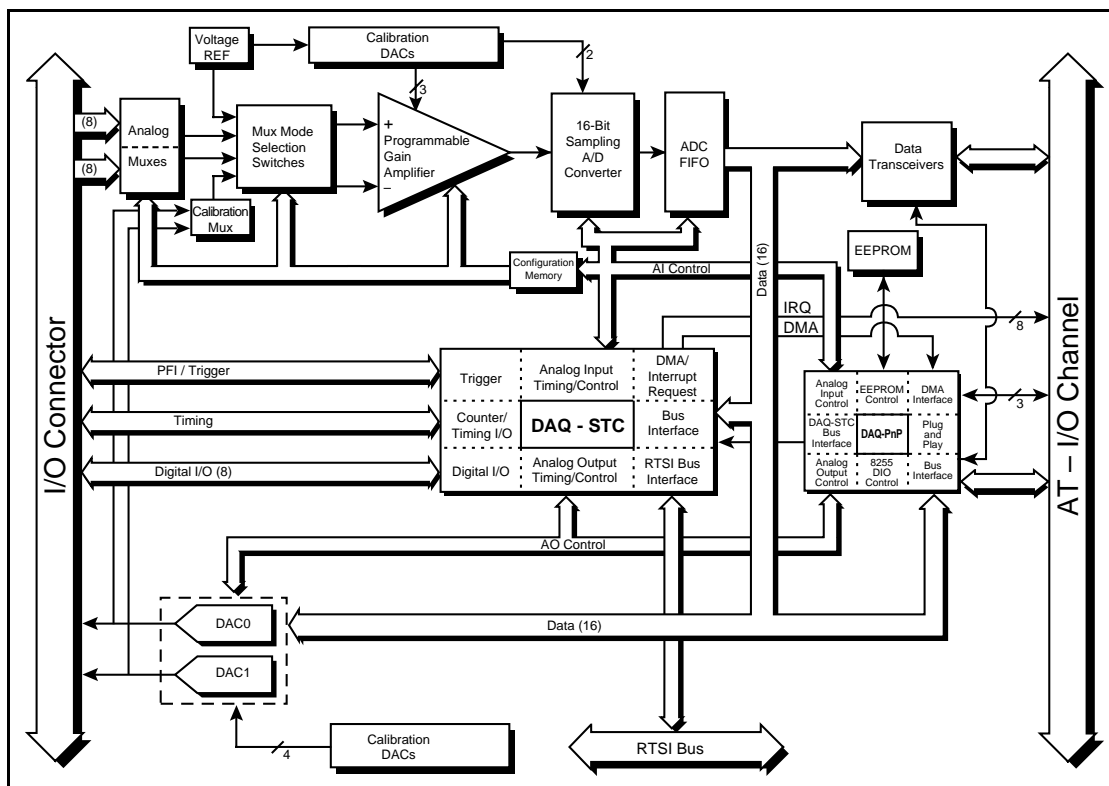


Figure 2-3. AT-MIO-16XE-50 Block Diagram

The following major components make up the AT-MIO E Series boards:

- ISA bus interface circuitry with Plug and Play capability (DAQ-PnP)
- Analog input circuitry
- Analog trigger circuitry
- Analog output circuitry
- Digital I/O circuitry
- Timing I/O circuitry (DAQ-STC)
- RTSI bus interface circuitry

The internal data and control buses interconnect the components. Notice that the DAQ-STC is the timing engine that provides precise timing signals for the analog input and output operations. The timing I/O circuitry information in this manual is skeletal in nature and is sufficient in most cases. For register-level programming information, refer to the *DAQ-STC Technical Reference Manual*.

## ISA Bus Interface Circuitry

The AT-MIO E Series boards are all full-size, 16-bit ISA I/O interface cards. The ISA bus features a 16-bit address bus, a 16-bit data bus, a DMA arbitration bus, interrupt lines, and several control and support signals. Figure 2-4 shows the functional blocks making up the AT-MIO E Series ISA bus interface circuitry.

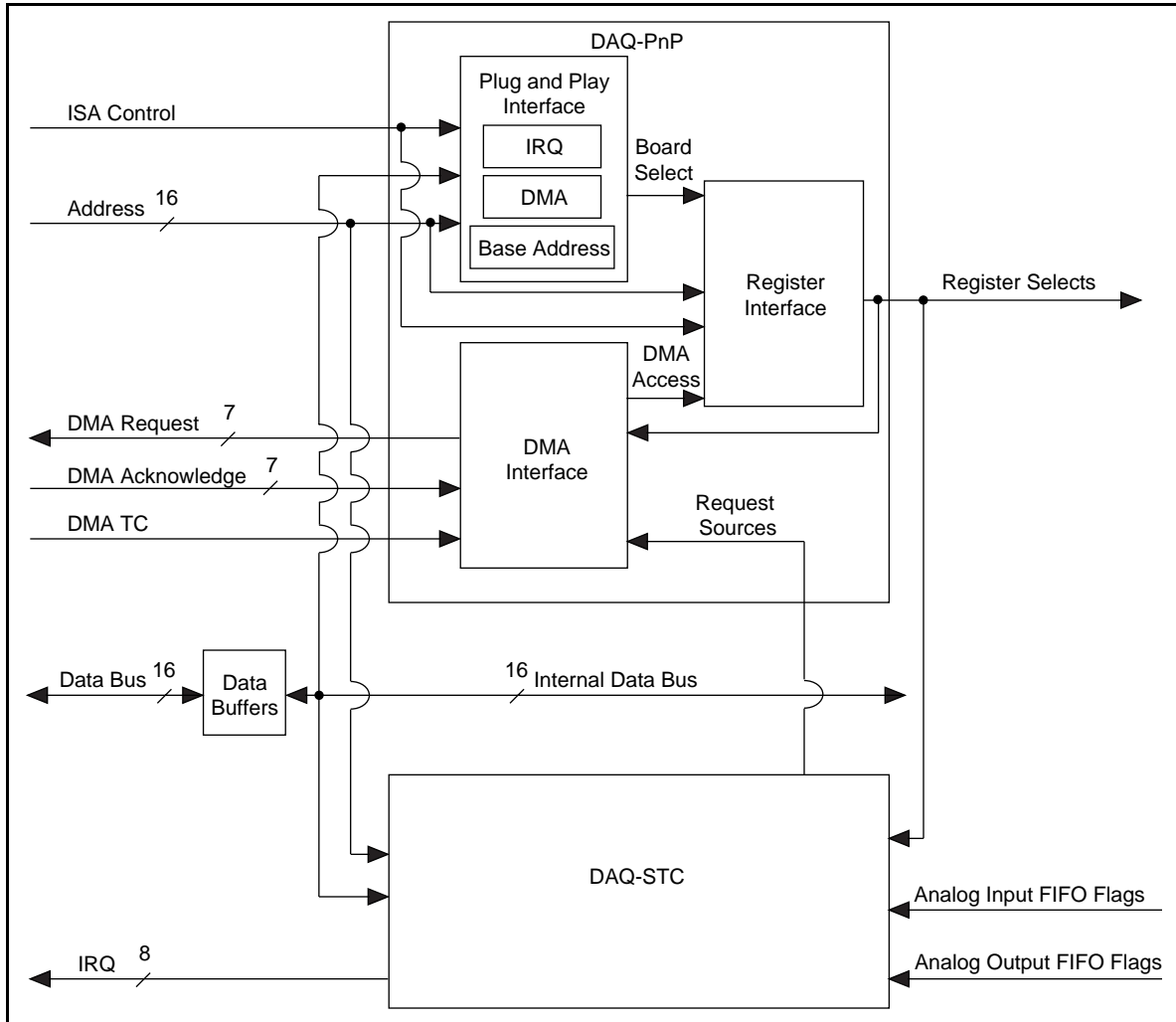


Figure 2-4. ISA Bus Interface Circuitry Block Diagram



The DAQ-PnP provides the main interface to the ISA bus, including support for the Plug and Play ISA Specification. Plug and Play provides a mechanism for assigning bus resources to the card using software only. This eliminates the need for setting jumpers or DIP switches on the board to select the base I/O address, interrupt lines, and DMA channels. In a Plug and Play aware system, the Plug and Play circuitry in the DAQ-PnP can request a base I/O address, up to three DMA channels, and up to two interrupt lines from the configuration manager. In other systems, application software can simply assign particular resources to the card.

The DAQ-PnP stores the resource assignments in the Plug and Play configuration registers. It also performs the base I/O address decoding after the board is activated through software. The DAQ-PnP performs full 16-bit address decoding, where address lines SA<15..5> are used for the comparison. This decoding allows the board to be located on any 32-byte boundary within the 16-bit address space. The remaining address lines SA<4..0> are used to select the onboard registers.

The register interface block within the DAQ-PnP is used to select the target register of a particular read or write cycle. These accesses are controlled by the base address decoding, SA<4..0> and the ISA control lines when the CPU is trying to access the board. In the case of DMA transfers, the access is controlled by the DMA block within the DAQ-PnP and the ISA control lines. The register interface block provides all access to the registers in the DAQ-PnP, DAQ-STC, and other onboard resources such as the FIFOs.

The DMA interface block within the DAQ-PnP provides for high-speed data transfer between the board and the system memory. Up to three DMA channels can be used simultaneously for data transfers with analog input, analog output, and the general-purpose counters. The DMA circuitry can only perform 16-bit transfers—DMA channels 5, 6, and 7 are available in ISA computers, whereas channels 0, 1, 2, 3, 5, 6, and 7 are available in EISA computers. The DMA transfers are initiated by the DAQ-STC, which indicates to the DAQ-PnP when the data transfers should occur. The DAQ-PnP then asserts the appropriate DMA request line and performs the transfer when the DMA acknowledge occurs.

The DAQ-STC can generate interrupts from over 20 sources and can route these interrupts to one or two of eight interrupt lines on the ISA bus interface. The use of two interrupt lines can improve performance in some environments but is usually more difficult to program. The AT-MIO E Series boards connect the eight DAQ-STC interrupt lines to the IRQ3, 4, 5, 7, 10, 11, 12, and 15 lines, respectively.

The data buffers are used to control the direction of the data transfers on the bi-directional data lines based on whether the transfer is a read or write operation. The enabling and direction of the buffers is controlled by the DAQ-PnP.

## Analog Input and Timing Circuitry

The AT-MIO E Series boards have 16 to 64 analog input channels and a timing core within the DAQ-STC that is dedicated to analog input operation. Figure 2-5 shows a general block diagram for the analog input circuitry.

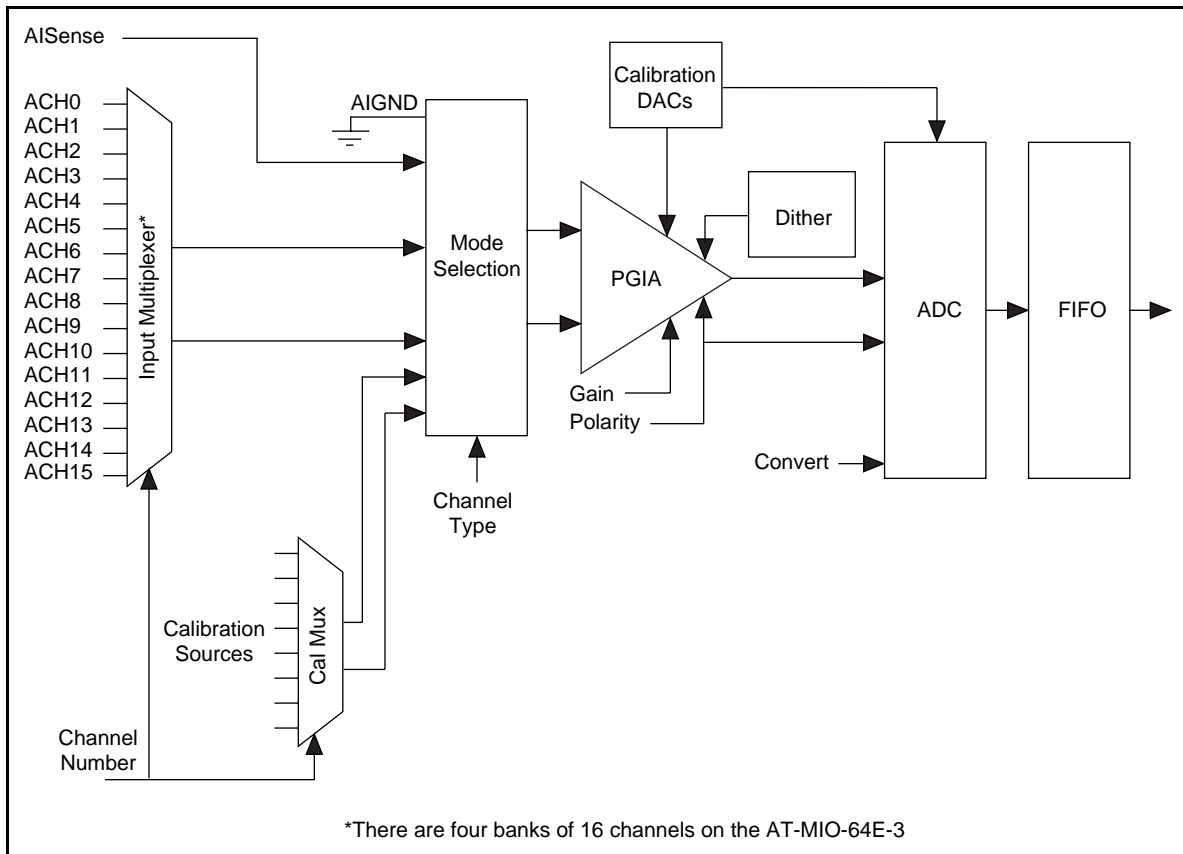


Figure 2-5. Analog Input and Data Acquisition Circuitry Block Diagram

### Analog Input Circuitry

The general model for analog input on the AT-MIO E Series boards includes input multiplexer, multiplexer mode selection switches, a software-programmable gain instrumentation amplifier, calibration hardware, a sampling ADC, a 16-bit wide data FIFO, and a configuration memory.

The configuration memory defines the parameters to use for each conversion. Each entry in the configuration memory includes channel type, channel number, bank, gain, polarity, dither, general trigger, and last channel. The configuration memory is a 512-byte deep FIFO that is initialized prior to the start of the acquisition sequence. It can be incremented after every conversion, allowing the analog input configuration to vary on a per conversion basis. Once the FIFO is empty, the DAQ-STC asserts the FIFO retransmit signal, which restores the FIFO data to its original state.

The channel type field indicates the resource type to be used during the conversion and controls the multiplexer mode selection switches. These resources include calibration channels, analog input channels in differential, referenced single-ended, or nonreferenced single-ended mode, or a ghost channel. The ghost type indicates that a conversion should occur but that the data should not be stored in the data FIFO. This type is useful for multirate scanning, which is described later in this chapter.

The channel number indicates which channel of the specified type will be used during the conversion, while the bank field indicates which bank of 16 channels is active. This bank field is used on boards that have more than 16 channels. These bits control the input multiplexers.

The programmable gain instrumentation amplifier (PGIA) serves two purposes on the AT-MIO E Series boards. The PGIA applies gain to the input signal, amplifying an analog input signal before sampling and conversion to increase measurement resolution and accuracy. This gain is determined by the gain field in the configuration memory. It also provides polarity selection for the input signal, which is also controlled by the configuration memory. In unipolar mode, the input range includes only positive voltages. In bipolar mode, the input signal may also be a negative voltage. The PGIA provides gains of 0.5, 1, 2, 5, 10, 20, 50, and 100, except on the AT-MIO-16XE-50, which supports gains of 1, 2, 10, and 100.

The dither circuitry adds approximately 0.5 LSB rms of white Gaussian noise to the signal to be converted by the ADC. This addition is useful for applications, such as calibration, involving averaging to increase the resolution of the board to more than the resolution of the ADC. In such applications, which are often lower frequency in nature, adding the dither decreases noise modulation and improves differential linearity. Dither should be disabled for high-speed applications not involving averaging because it would only add noise. When taking DC measurements, such as when calibrating the board, you should enable dither and average about 1,000 points to take a single reading. This process removes the effects of quantization, reduces measurement noise, and improves resolution. Notice that dither cannot be disabled on the AT-MIO-16XE-50.

The last channel bit is used to indicate that this is the last conversion in a scan. The DAQ-STC will end the scan on the conversion with this bit set.

The AT-MIO E Series boards use sampling, successive approximation ADCs with 12 or 16 bits of resolution with maximum conversion rates between 50 and 1  $\mu$ s. The converter can resolve its input range into 4,096 different steps for the 12-bit ADC and 65,536 for the 16-bit ADC. The input range of the 12-bit boards is  $\pm 5$  V in bipolar mode and 0 to +10 V in unipolar mode. These modes correspond to ranges of -2,048 to 2,047 in unipolar mode and 0 to 4,095 in bipolar mode. The input range of the 16-bit boards is  $\pm 10$  V in bipolar mode and 0 to +10 V in unipolar mode. These modes correspond to ranges of -32,768 to 32,767 in bipolar mode and 0 to 65,535 in unipolar mode.

The AT-MIO E Series boards include a 16-bit wide FIFO to buffer the analog input data. This buffering will increase the maximum rate that the analog input can sustain during continuous acquisition. The FIFO is 8 kwords deep on the AT-MIO-16E-1, 2 kwords deep on the AT-MIO-16E-2 and AT-MIO-64E-3, and 512 words deep on the others. The DAQ-STC shifts the data into the FIFO from the ADC when the conversion is complete. This buffering allows the ADC to begin a new conversion even though the data has not yet been read from the board.

This buffering also provides more time for the software or DMA to respond and read the analog input data from the board. If the FIFO is full and another conversion completes, an error condition called *FIFO overflow* occurs and the data from that conversion is lost. The FIFO not empty, half-full, and full flags are available to generate interrupts or DMA requests for the data transfer.

Measurement reliability is assured through the onboard calibration circuitry of the board. This circuitry uses an internal, stable 5 V reference that is measured at the factory against a higher accuracy reference; its value is then stored in the EEPROM. With this stored reference value, the board can be recalibrated at any time under any number of different environmental conditions in order to remove errors caused by time and temperature drift. The EEPROM stores calibration constants that can be read and then written to calibration DACs that adjust input offset, output offset, and gain errors associated with the analog input section. When the board leaves the factory, the upper one-fourth of the EEPROM is protected and cannot be overwritten. The lower three-fourths is unprotected and the top fourth of that can be used to store alternate calibration constants for the different conditions under which you use the board. The entire lower half of the EEPROM is used for storing Plug and Play information and should never be written to.

## Data Acquisition Timing Circuitry

This section describes the different methods of acquiring A/D data from a single channel or multiple channels.

From this section through the end of this manual, you are assumed to have a working knowledge of the DAQ-STC features. These features are explained in the *DAQ-STC Technical Reference Manual*. If you have not read the functional description of each DAQ-STC module, you must do so before completing this register-level programmer manual.

### Single-Read Timing

To acquire data from the ADC, initiate a single conversion and read the resulting value from the ADC FIFO buffer after the conversion is complete. You can generate a single conversion in three different ways—apply an active low pulse to the CONVERT\* pin of the I/O connector, generate a falling edge on the sample-interval counter of the DAQ-STC, or strobe the appropriate bit in a register in the AT-MIO E Series register set. Any one of these operations will generate the timing shown in Figure 2-6.

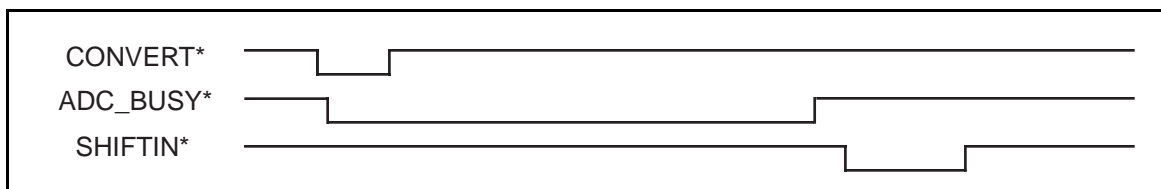


Figure 2-6. ADC Timing

When SHIFTIN\* shifts the ADC value into the ADC FIFO buffer, the AI\_FIFO\_Empty\_St bit in the status register is cleared, which indicates that valid data is available to be read. Single conversion timing of this type is appropriate for reading channel data on an *ad hoc* basis. However, if you need a sequence of conversions, the time interval between successive conversions is not constant because it relies on the software to generate the conversions. For finely timed conversions that require triggering and gating, you must program the boards to automatically generate timed signals that initiate and gate conversions. This is known as a data acquisition sequence.

### Data Acquisition Sequence Timing

The following counters are used for a data acquisition sequence:

- Scan interval (SI)            24 bits
- Sample interval (SI2)        16 bits
- Divide by (DIV)              16 bits
- Scan counter (SC)            24 bits

This section presents a concise summary of only the most important features of your board. For a complete description of all the analog input modes and features of the AT-MIO E Series, refer to the *DAQ-STC Technical Reference Manual*.

The most basic timing signal in the analog input model is the CONVERT\* signal. A group of precisely timed CONVERT\* pulses is a SCAN. The sequence of channels selected in each conversion in a SCAN is programmed in the configuration memory prior to starting the operation. The SI2 counter is a 16-bit counter in the DAQ-STC. This counter determines the interval between CONVERT\* pulses. It can be programmed for a maximum interval of 3.3 ms and a minimum interval of 50 ns. If alternate slow timebases are used, the maximum interval is 0.65 s. Each time the counter reaches terminal count (TC), a CONVERT\* pulse is generated. Alternatively, CONVERT\* pulses could be given externally.

A SCAN sequence is started by the START pulse, which is generated by the terminal count of the SI counter. This counter is a 24-bit counter that determines the time between the start of each SCAN. The minimum duration is 50 ns and the maximum duration is 0.8 s when the internal 20 MHz timebase is used. If the internal 100 kHz timebase is used, the maximum is 167 s. The START pulse triggers the SI2 counter to generate CONVERT\* pulses. With each conversion, the configuration memory advances by one and selects the next set of analog input conditions — channel number, gain, polarity, etc. A STOP pulse ends the SCAN sequence. This STOP could be generated in two ways—either by using the LASTCHANNEL bit in the configuration memory or by programming the 16-bit DIV counter to count the number of conversions per SCAN and using the terminal count of the DIV counter as a STOP pulse.

The SC is a 24-bit counter that counts the number of scans. The data acquisition sequence can be programmed to stop when the terminal count of this counter is reached. Notice that the START and STOP signals could also be supplied externally.

Example 1: To acquire 50 scans, with each scan consisting of one sample on channel 0 at gain 50, one sample on channel 5 at gain 2, and one sample on channel 3 at gain 10, with a SCAN interval of 100  $\mu\text{s}$  and a sample interval of 10  $\mu\text{s}$ , program your configuration memory as follows:

1. channel 0, gain 50
2. channel 5, gain 2
3. channel 3, gain 10, last channel

You should program SI2 for 10  $\mu\text{s}$ , SI for 100  $\mu\text{s}$ , and SC for 50 (50 scans).

Figure 2-7 shows the timing for each scan in example 1.

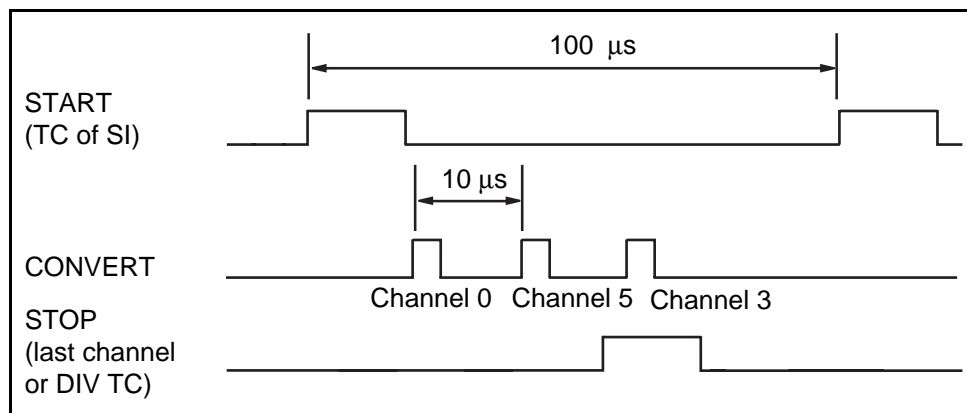


Figure 2-7. Timing of Scan in Example 1

The START pulse starts each scan. The first CONVERT pulse samples channel 0, the second CONVERT pulse samples channel 5, and the third CONVERT pulse samples channel 3. The STOP pulse ends the scan.

Example 1 allows you to sample all three channels at a rate of 10 kS/s per channel (100  $\mu\text{s}$  sample interval period). To achieve different rates for different channels, you must do multirate scanning.

### Multirate Scanning without Using Ghost

Example 2: To sample channel 0 at 10 kS/s and channel 1 at 5 kS/s, both at gain 1 with 50 scans, program the configuration memory as follows:

1. channel 0, gain 1
2. channel 1, gain 1, last channel
3. channel 0, gain 1, last channel

Program SI for 100  $\mu\text{s}$ , SI2 for 10  $\mu\text{s}$ . Figure 2-8 shows the timing sequence for two scans.

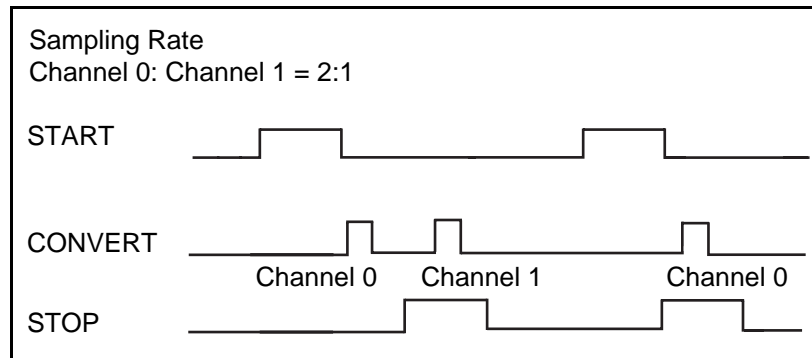


Figure 2-8. Multirate Scanning of Two Channels

This sequence of two scans is repeated 25 times to complete the acquisition. Notice that channel 0 is sampled once every 100  $\mu\text{s}$ . Hence, its sampling rate is 10 kS/s, whereas channel 1 is sampled once every 200  $\mu\text{s}$ . Its rate is 5 kS/s. Similarly, you could implement any 1: $x$  ratio of sampling rates. The effective scan interval of the slower channel will be at  $\frac{1}{x}$  the rate of the faster channel. This implementation requires  $x$  scan sequences in the configuration memory.

Also, you can implement a 1:1: $x$  or 1: $x$ : $m$  ratio for three channels, where  $m$  is a non-negative integer. Figures 2-9, 2-10, and 2-11 show timing sequences for different ratios. In these figures, the numbers above the CONVERT pulses indicate the channels sampled in that conversion.

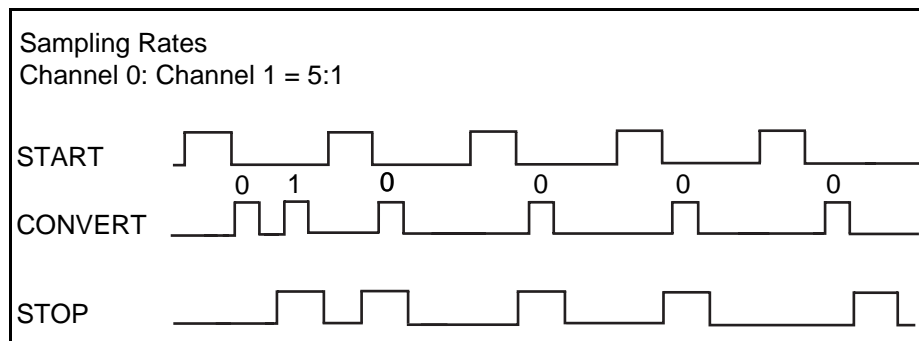


Figure 2-9. Multirate Scanning of Two Channels with 1: $x$  Sampling Rate

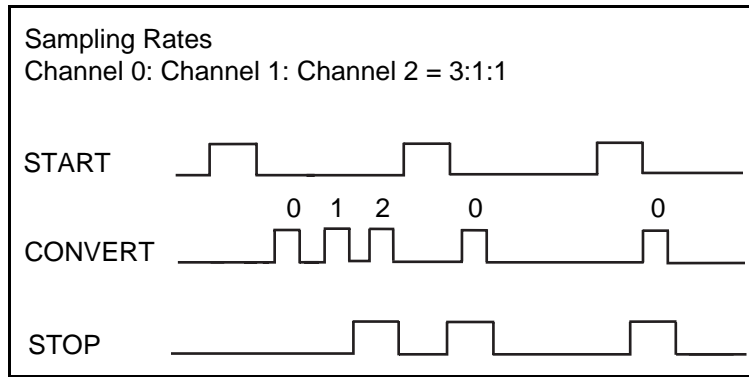


Figure 2-10. Multirate Scanning of Two Channels with 3:1:1 Sampling Rate

Here, channel 0 is sampled three times, whereas channels 1 and 2 are sampled once every three scans.

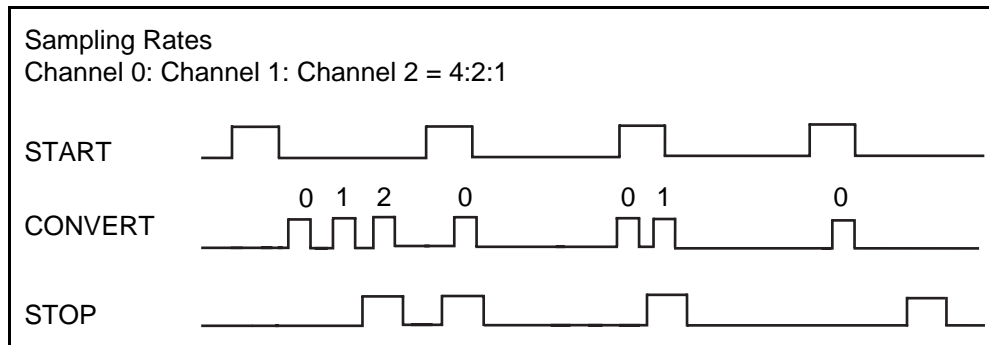


Figure 2-11. Multirate Scanning of Three Channels with 4:2:1 Sampling Rate

### Multirate Scanning Using Ghost

If the ghost option in the configuration memory is set, that conversion occurs but the data is not stored in the analog input FIFO. In other words, a conversion is performed and the data is thrown away. By using this option, multirate scanning with ratios such as  $x:y$  are possible, within the limits imposed by the size of the configuration memory.

Figures 2-12 and 2-13 illustrate the advantages of using the ghost feature. Figure 2-12 shows example 3 timing, and Figure 2-13 shows the same example using ghost.

Example 3: channel 1: channel 0 = 2:3 (without ghost).



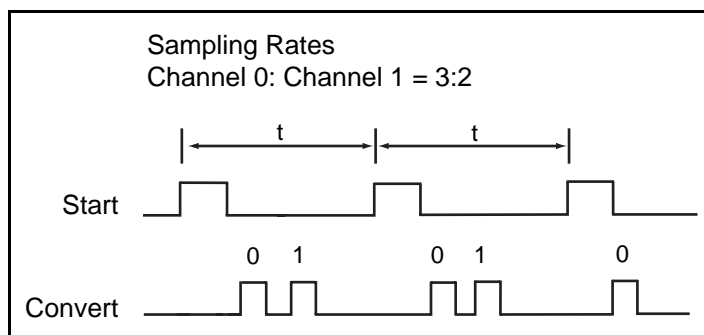


Figure 2-12. Multirate Scanning without Ghost

In example 3, channel 0 is sampled correctly. Although channel 1 is sampled twice, it does not yield a 50% duty cycle. This type of acquisition will result in imprecise rates. Figure 2-13 shows the relative occurrences of convert pulses in Figure 2-12.

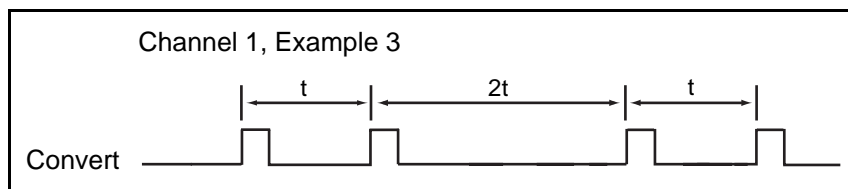


Figure 2-13. Occurrences of Conversion on Channel 1 in Example 3.

To rectify the problem, use ghost as illustrated in Figure 2-14.

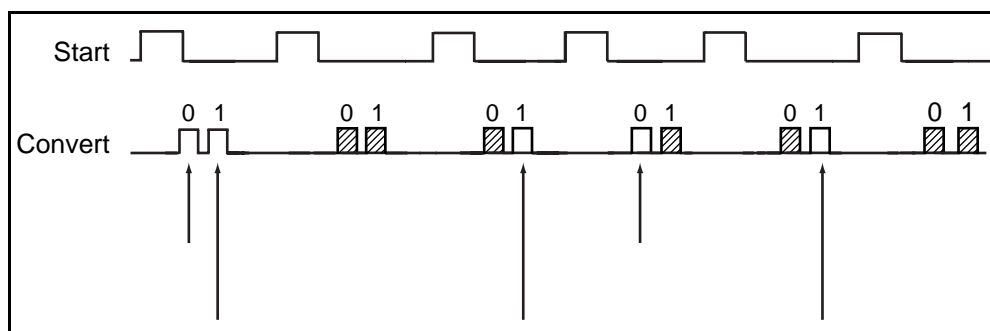


Figure 2-14. Successive Scans Using Ghost

The shaded conversions are ghost conversions. The short arrows indicate channel 0 samples and the long arrows indicate channel 1 samples that are actually stored in the FIFOs.

Table 2-1 shows what the configuration memory would look like.

Table 2-1. Analog Input Configuration Memory

Channel	Ghost	Last Channel
0	—	—
1	—	last channel
0	ghost	—
1	ghost	last channel
0	ghost	—
1	—	last channel
0	—	—
1	ghost	last channel
0	ghost	—
1	—	last channel
0	ghost	—
1	ghost	last channel

The symbol — indicates that ghost or last channel is absent.

Now both channel 0 and channel 1 are sampled with 50% duty cycle.

## Posttrigger and Pretrigger Acquisition

Whereas a data acquisition operation normally ends when the SC counts down to zero, it can be initiated either through software, by strobing a bit in a control register in the DAQ-STC, or by suitable external triggers.

There are two internal trigger lines—START1 and START2. These appear at the connector on pins called PFI0/Trig1 and PFI1/Trig2, respectively. START1 is used as a trigger line in the posttrigger mode. Since the START1 pulse can be generated through software by strobing a bit, all of the examples discussed so far can be generally categorized as posttrigger acquisition. In the classic posttrigger mode case, the data acquisition circuitry is armed by software but does not start acquiring data until a pulse is given on the START1 line. Then the acquisition starts and ends when the SCAN counter counts down to zero.

In the pretrigger mode, data is acquired before and after the trigger. In this mode, both START1 and START2 lines are used. There are two counts for the SCAN counter—the pretrigger count and the posttrigger count. To begin with, the pretrigger count is loaded into the SC. Acquisition is then started through either software by strobing a bit, or through hardware by externally pulsing the START1 pin of the connector. Acquisition then starts and data is acquired. When the SC counts down to zero, the scans continue and data still gets acquired. During this time, the board waits for a START2 pulse but the SC does not count. Also, the SC gets loaded with the posttrigger count. When a START2 pulse is received, the SC once again starts counting. When it reaches zero, the operation ends. Refer to the *Acquisition Level Timing and Control* section of Chapter 2 in the *DAQ-STC Technical Reference Manual* for timing examples of pretrigger and posttrigger modes.

Variations:

- Posttrigger and pretrigger modes can be retriggerable. This means that after one posttrigger or pretrigger operation is over, the SC gets reloaded and the board sits waiting for an additional START1. This can continue indefinitely and can be disabled through software.
- A special mode in the DAQ-STC allows continuous software-initiated acquisition to continue indefinitely. In this mode, the SC gets reloaded each time it counts down to zero. The acquisition can be stopped by disarming the SC. When the SC counts down to zero, acquisition stops.

## Analog Triggering

The AT-MIO-16E-1, AT-MIO-16E-2, and AT-MIO-64E-3 have an analog trigger in addition to the digital triggers. To use analog triggering to start an acquisition sequence, select either the PFI0/Trig1 input on the I/O connector or one of the analog input pins. An analog input pin allows you to apply gain to the external signal for more flexible triggering conditions. An INT/EXTTRIG bit in the Misc. Command Register selects the source. PFI0/Trig1 pin has an input voltage range of  $\pm 10$  V.

An 8-bit serial DAC sets each of the high and low thresholds. These thresholds are within  $\pm$  full scale. The selected input is compared against each of these thresholds by a comparator. The outputs of the comparators are connected to the DAQ-STC analog trigger inputs 0 and 1. Within the DAQ-STC, these signals can be routed to any of the internal timing signals.

You can trigger on either a positive or negative slope or on absolute values.

Refer to the *DAQ-STC Technical Reference Manual* and the *NI-DAQ Function Reference Manual* for more information on analog triggering. For a detailed description of these modes, and timing diagrams, and for a description of other modes not discussed here, refer to the *DAQ-STC Technical Reference Manual*.

## Analog Output and Timing Circuitry

The AT-MIO E Series boards have two analog output channels and a timing core within the DAQ-STC that is dedicated to analog output operation. Figure 2-15 shows a general block diagram for the analog output circuitry.

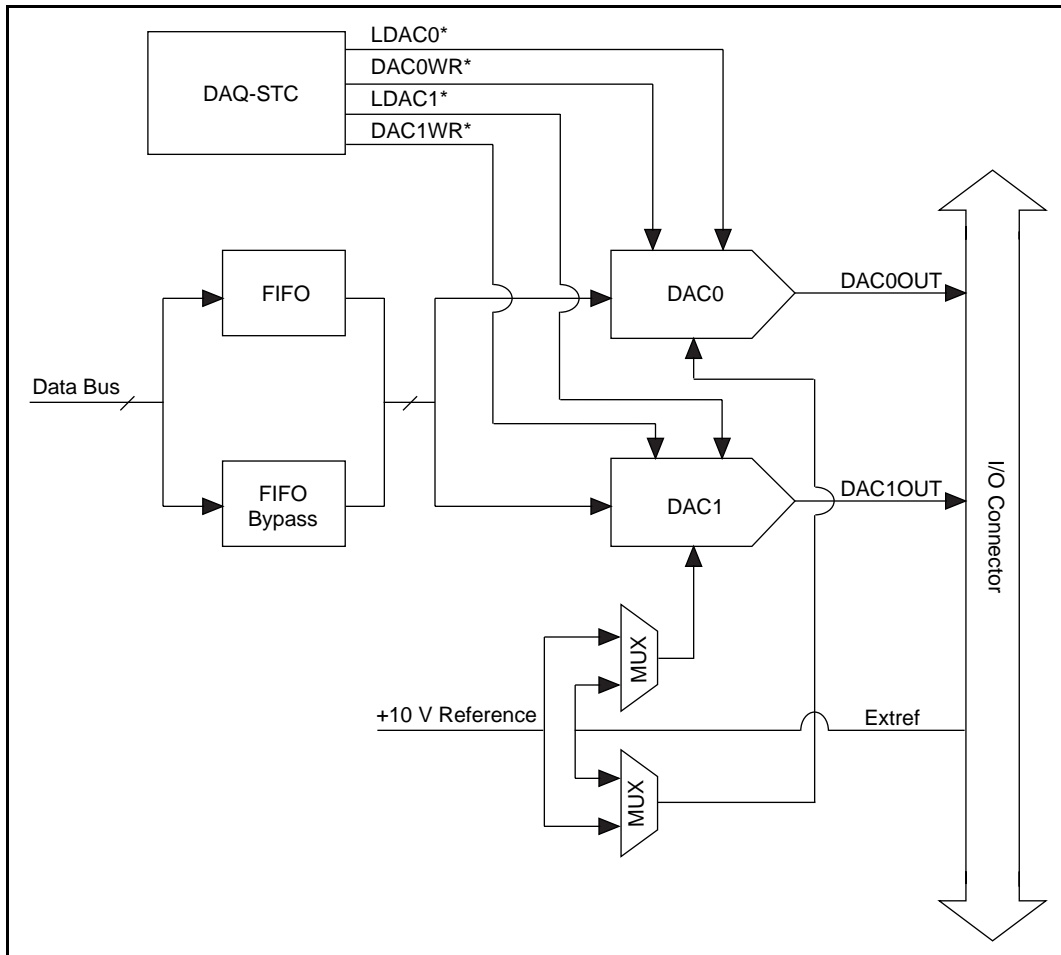


Figure 2-15. Analog Output Circuitry Block Diagram

## Analog Output Circuitry

The general model for analog output on the AT-MIO E Series boards includes two channels of double-buffered analog output with programmable polarity, reference source, and reglitching circuit, as well as a FIFO to buffer the data. However, not all of the AT-MIO E Series boards contain every one of these features.

Each analog output channel contains a 12-bit DAC, an amplification stage, and an onboard voltage reference. The output voltage will be proportional to the voltage reference ( $V_{ref}$ ) multiplied by the digital code loaded into the DAC. Note that the output will be set to 0 V on power up. The polarity, reference source, and reglitching circuit are all configured in the AO Configuration Register.

The voltage reference source for each DAC is selectable from the onboard reference or a voltage supplied at the EXTREF pin on the I/O connector, except for the AT-MIO-16XE-50, which only supports the onboard reference. The onboard reference is fixed at +10 V. The external reference can be either a DC or an AC signal. If you apply an AC reference, the analog output channel acts as a signal attenuator and the AC signal appears at the output attenuated by the digital code.

For unipolar output the voltage is simply attenuated. Four quadrant multiplication occurs in bipolar output, where the signal will not only be attenuated but also inverted for negative digital codes.

The DAC output can be configured to produce either a unipolar or bipolar output range, except for the AT-MIO-16XE-50, which supports only bipolar output. A unipolar output has an output range of 0 to  $+V_{ref} - 1 \text{ LSB}$  V. A bipolar output has an output voltage range of  $-V_{ref}$  to  $(+V_{ref} - 1 \text{ LSB})$  V. For unipolar output, the data written to the DAC is interpreted in straight binary format. For bipolar output, the data is interpreted as two's complement format. One LSB is the voltage increment corresponding to an LSB change in the digital code word. For unipolar output,  $1 \text{ LSB} = (V_{ref})/4,096$ . For bipolar output,  $1 \text{ LSB} = (V_{ref})/2,048$ .

Using the onboard 10 V reference will produce an output voltage range of 0 to 9.9976 V in steps of 2.44 mV for unipolar output and an output voltage range of -10 V to +9.9951 V in steps of 4.88 mV for bipolar operation.

In normal operation, a DAC output will glitch whenever it is updated with a new value. The glitch energy differs from code to code and appears as distortion in the frequency spectrum. Each analog output of the AT-MIO-16E-1, AT-MIO-16E-2, and AT-MIO-64E-3 contains a reglitch circuit that generates uniform glitch energy at every code rather than large glitches at the major code transitions. This uniform glitch energy appears as a multiple of the update rate in the frequency spectrum. Notice that this reglitch circuit does not eliminate the glitches; it only makes them more uniform in size.

The AT-MIO-16E-1, AT-MIO-16E-2, and AT-MIO-64E-3 include 2 kword-deep FIFOs to buffer the analog output data. This buffering will increase the maximum rate that the analog output can sustain for waveform generation. It can also be used to store a complete waveform which can be output repetitively without any further data transfer to the FIFO.

## Analog Output Timing Circuitry

This section describes the different methods of setting the analog output voltage, including single-point updating and waveform generation. The DAQ-STC and DAQ-PnP provide the timing signals necessary to write to the DACs and update them.

The DACs are double buffered and can be individually configured for immediate update or timed update mode. In immediate update mode, the double buffering of the DAC is disabled and the value written to the DAC will appear immediately on the output after the write completes. In the timed update mode, the double buffering is enabled. When double buffering is enabled, writing the digital value loads that value into the buffer stage of the DAC. When an update signal is received, the analog output of the DAC changes.

### Single-Point Output

The data values can be written directly to the DACs under software control without the use of the timing engine provided in the DAQ-STC. This is typically useful for setting the analog outputs to DC levels, where precise timing of the output change is not important. Writing directly to the DACs is accomplished by writing the desired value to the DAC<0..1> Direct Data Register.

This action will store the value in the first buffer of the DAC. If the DAQ-STC is programmed for immediate updating mode, the value will also be applied to the analog output. If the DAQ-STC is programmed for timed updating mode, the appropriate update signal, LDAC0\* or LDAC1\*, must be asserted by writing to the appropriate DAQ-STC register.

## Waveform Generation

The timing engine in the DAQ-STC can be used for waveform generation, where the update signals connected to the DACs can be generated at precise intervals. In this model, the DAQ-STC will generate the update signals and then transfer the data values for the next update to the first buffer of the DACs.

The update interval counter (UI) is 24 bits wide and generates the update pulses. These update pulses can be routed to the DACs. The update counter (UC) is 24 bits wide and counts the number of updates. The value loaded into this counter defines a buffer. The buffer counter (BC) determines how many times a buffer should be repeated. These counters define an analog output sequence.

The DAQ-STC will transfer the data values for the next update after each update pulse. This data transfer will be from the analog output FIFO. Some of the boards have large 2 kword FIFOs; others are zero-deep virtual FIFOs. The large FIFOs are true FIFOs, where data can be written to the FIFOs so long as they are not full and can be read from when they are not empty. In this case, the DAQ-STC will transfer the data from the FIFO to the DACs when the FIFO is not empty. If the FIFO is empty, the DAQ-STC will delay the transfer until more data is written into the FIFO.

The zero-deep virtual FIFOs are not true FIFOs, but they do provide a software compatible method for waveform generation. When the DAQ-STC generates an update signal and is ready to transfer data, the DAQ-PnP will clear the FIFO full flag. This FIFO full flag can be used by interrupts or DMA to transfer data to the virtual FIFO. The virtual FIFO will not actually buffer the data, instead it will transfer it directly to the first buffer of the destination DAC. After the data has been transferred, the DAQ-PnP will set the FIFO full flag, indicating that no more data is required. Note that the half-full and empty flags do not provide useful information for data transfer and should not be used.

The large FIFOs can also be used to generate repetitive waveforms at very high speeds and without using any bus bandwidth. The FIFOs can be loaded with the desired waveform and the DAQ-STC can be programmed to retransmit the same FIFO data to the DACs over and over. When the FIFO is empty, the retransmit signal is asserted, which restores that FIFO data to its original state.

When both DACs are used in waveform generation, the data in the FIFO is interleaved; in other words, every alternate sample belongs to one DAC.

## Digital I/O Circuitry

The AT-MIO E Series boards have eight digital I/O lines. Each of the eight digital lines can be individually programmed to be input or output, if used in parallel. For serial data transfer, DIO4 is used as the serial data in pin, and DIO0 is used as the serial data out pin.

You can use handshaking with the EXTSTROBE\* pin to do either parallel or serial data transfer. Refer to the *DAQ-STC Technical Reference Manual* for more details on the DIO features.

The external strobe signal EXTSTROBE\* is a general-purpose strobe signal. Software can set the output of the EXTSTROBE\* pin to either a high or low state. EXTSTROBE\* is not necessarily part of the digital I/O circuitry but is included here because you can use it to latch digital output from the AT-MIO E Series into an external device.

## Timing I/O Circuitry

The DAQ-STC has two 24-bit general purpose counter/timers. The counters are numbered 0 and 1 and are diagrammed as shown in Figure 2-16.

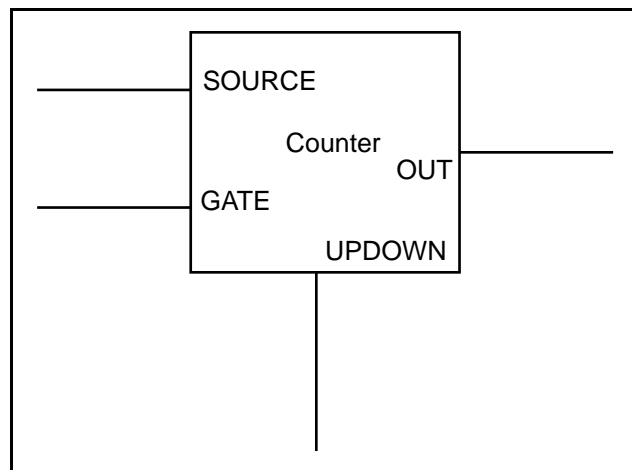


Figure 2-16. DAQ-STC Counter Diagram

The counter counts the transitions, or edges, on the SOURCE line when the GATE is enabled and generates timing signals at its OUT output pin. The UPDOWN pin determines the direction of counting. Active polarities of these pins are software selectable in the DAQ-STC. Notice that on the AT-MIO E Series only the SOURCE, GATE, and OUT pins are brought out to the I/O connector. The UPDOWN pin for counter 0 is internally connected to DIO6 and to DIO7 for counter 1. To drive the UPDOWN pin from the connector, you must tri-state the corresponding DIO line.

The SOURCE of these counters can be selected to be one of the 10 PFI lines or the seven RTSI lines, or the internal 20 MHz or 100 kHz timebases, or the TC of the other counter. With the last

option, the two counters can be concatenated. Similarly, the GATE can also be selected from a variety of different sources. The sources for both of these signals are described in the *DAQ-STC Technical Reference Manual*.

## RTSI Bus Interface Circuitry

The AT-MIO E Series is interfaced to the National Instruments RTSI bus. The RTSI bus has seven trigger lines and a system clock line. You can wire all National Instruments AT-MIO E Series boards with RTSI bus connectors inside the PC AT and share these signals. Figure 2-17 shows a block diagram of the RTSI bus interface circuitry.

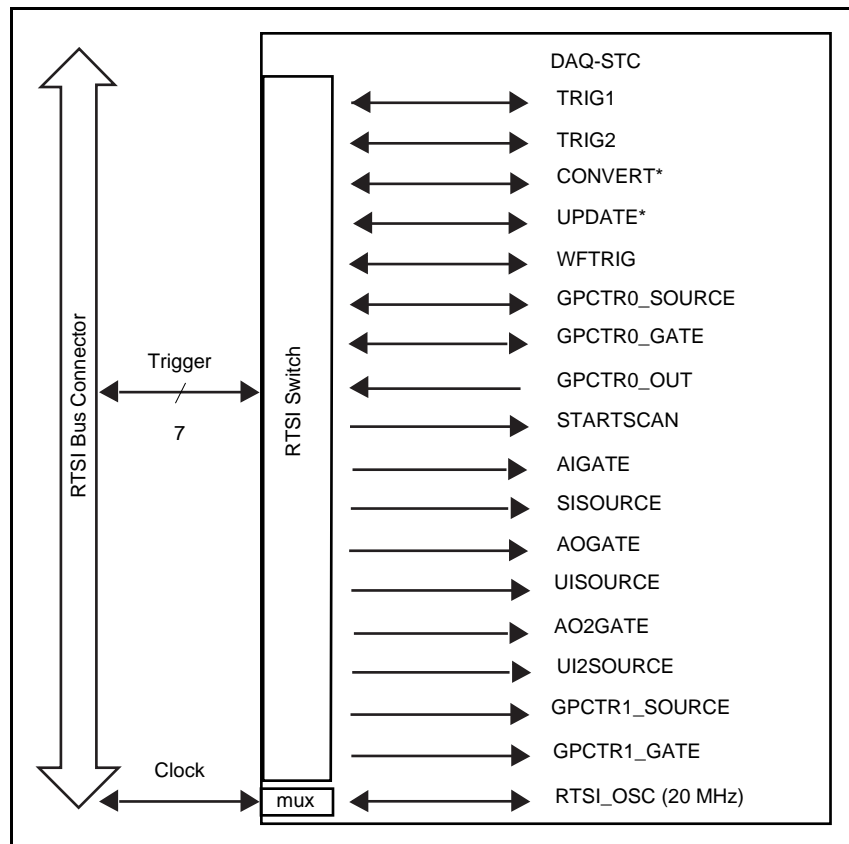


Figure 2-17. RTSI Bus Interface Circuitry Block Diagram

The RTSI functionality is provided in the DAQ-STC chip. This RTSI Trigger Module consists of seven 12-to-1 multiplexers that drive seven RTSI Trigger bus signals and four 8-to-1 multiplexers that drive the four RTSI Board signals.

Any of the seven RTSI Trigger bus signals can be driven by eight internally generated timing signals and the four RTSI Board signals. Similarly, the four RTSI Board signals can be driven by any of the RTSI Trigger bus signals. Of the four RTSI board signals, only one is used. By programming certain bits in the DAQ-STC, you can drive the CONVERT pulse onto RTSI\_BRD0 and then onto any of the TRIGGER lines.



# Chapter 3

## Register Map and Descriptions

---

This chapter describes in detail the address and function of each of the AT-MIO E Series control and status registers.

If you plan to use a programming software package such as NI-DAQ for DOS/Windows or Lab Windows with your AT-MIO E Series board, you need not read this chapter. However, you will gain added insight into your AT-MIO E Series board by reading this chapter.

### Register Map

Table 3-1 shows the register map for the AT-MIO E Series boards and gives the register name, the register offset address, the type of the register (read-only, write-only, or read-and-write), and the size of the register in bits. The actual register address is obtained by adding the appropriate register offset to the I/O base address of the AT-MIO E Series board.

Registers are grouped in the table by function. Each register group is introduced in the order shown in Table 3-1, then described in detail, including a bit-by-bit description.

The DAQ-STC has 180 different registers. The more frequently used registers have been given lower offset addresses and are shown in Table 3-1 as the DAQ-STC Register Group. The advantage of having lower offset addresses is that they can be accessed directly. The remaining registers in the DAQ-STC can be accessed in the windowed mode. In this mode, the address of the register is first written to the Window Address Register. The data to be written is then written to the Window Data Register. Using windowed mode has the advantage of reducing the address space of the board from 180 to 32 bytes. However, this reduced address space is at the cost of two write operations to write data to or two read operations to read data from a register.

Table 3-1. AT-MIO E Series Register Map

Register Name	Offset Address		Type	Size
	Hex	Decimal		
Misc Register Group				
Serial Command	0D	13	Write-only	8-bit
Misc Command	0F	15	Write-only	8-bit
Status	01	1	Read-only	8-bit
Analog Input Register Group				
ADC FIFO Data Register	1C	28	Read-only	16-bit
Configuration Memory Low	10	16	Write-only	16-bit
Configuration Memory High	12	18	Write-only	16-bit

(continues)

Table 3-1. AT-MIO E Series Register Map (Continued)

<b>Analog Output Register Group</b>				
AO Configuration	16	22	Write-only	16-bit
DAC FIFO Data	1E	30	Write-only	16-bit
DAC0 Direct Data	18	24	Write-only	16-bit
DAC1 Direct Data	1A	26	Write-only	16-bit
<b>DMA Control Register Group</b>				
Strobes	01	1	Write-only	8-bit
Channel A Mode	03	3	Write-only	8-bit
Channel B Mode	05	5	Write-only	8-bit
Channel C Mode	07	7	Write-only	8-bit
AI AO Select	09	9	Write-only	8-bit
G0 G1 Select	0B	11	Write-only	8-bit
<b>DIO Register Group</b>				
Port A	19	25	Read-and-write	8-bit
Port B	1B	27	Read-and-write	8-bit
Port C	1D	29	Read-and-write	8-bit
Configuration	1F	31	Write-only	8-bit
<b>DAQ-STC Register Group</b>				
Window Address	0	0	Read-and-write	16-bit
Window Data	2	2	Read-and-write	16-bit
Interrupt A Acknowledge	4	4	Write	16-bit
Interrupt B Acknowledge	6	6	Write	16-bit
AI Command 2	8	8	Write	16-bit
AO Command 2	A	10	Write	16-bit
G0 Command	C	12	Write	16-bit
G1 Command	E	16	Write	16-bit
AI Status 1	2	2	Read	16-bit
AO Status 1	4	4	Read	16-bit
G Status	6	6	Read	16-bit
AI Status 2	8	8	Read	16-bit
AO Status 2	A	10	Read	16-bit
DIO Parallel Input	C	12	Read	16-bit

Table 3-2. AT-MIO E Series Windowed Register Map

Register Name	Word Offset		Type	Size
	Hex	Decimal		
<b>FIFO Strobe Register Group</b>				
Configuration Memory Clear	52	82	Write-only	16-bit
ADC FIFO Clear	53	83	Write-only	16-bit
DAC FIFO Clear	54	84	Write-only	16-bit

## Register Sizes

Two different transfer sizes for read-and-write operations are available on the PC—byte (8-bit) and word (16-bit). Table 3-1 shows the size of each AT-MIO E Series register. For example, reading the ADC FIFO Data Register requires a 16-bit (word) read operation at the selected address, whereas writing to the Misc Command Register requires an 8-bit (byte) write operation at the selected address. For proper board operation you must adhere to these register size accesses. Avoid performing a byte access on a word location or performing a word access on a byte location; these are invalid operations. Pay particular attention to the register sizes; they are very important.

## Register Descriptions

This section discusses each of the AT-MIO E Series registers in the order shown in Table 3-1. Each register group is introduced, followed by a detailed bit description. The individual register description gives the address, type, word size, and bit map of the register, followed by a description of each bit.

The register bit map shows a diagram of the register with the MSB shown on the left (bit 15 for a 16-bit register, bit 7 for an 8-bit register), and the LSB shown on the right (bit 0). A square represents each bit and contains the bit name. An asterisk (\*) after the bit name indicates that the bit is inverted (negative logic).

In many of the registers, several bits are labeled with an X, indicating *don't care* bits. When a register is read, these bits may appear set or cleared but should be ignored because they have no significance.

## Misc Register Group

The three registers making up the Misc Register Group include two command registers that control the serial DACs, EEPROM, and analog trigger source, and one status register that includes EEPROM and DMA information.

Bit descriptions of the three registers making up the Misc Register Group are given on the following pages.

## Serial Command Register

The Serial Command Register contains five bits that control AT-MIO E Series serial EEPROM and DACs. The contents of this register are cleared upon power up and after a reset condition.

Address: Base address + 0D (hex)

Type: Write-only

Word Size: 8-bit

Bit Map:

7	6	5	4	3	2	1	0
Reserved	Reserved	Reserved	SerDacLd1	SerDacLd0	EEPromCS	SerData	SerClk

Bit	Name	Description
7–5	Reserved	Reserved—Always write 0 to these bits.
4	SerDacLd1	Serial DAC Load1—This bit is used to load the second set of serial DACs with the serial data previously shifted into the DACs ( <i>AT-MIO-16XE-50 only</i> ).
3	SerDacLd0	Serial DAC Load0—This bit is used to load the first set of serial DACs with the serial data previously shifted into the DACs.
2	EEPromCS	EEPROM Chip Select—This bit controls the chip select of the onboard EEPROM used to store calibration constants. When EEPROMCS is set, the chip select signal to the EEPROM is enabled.
1	SerData	Serial Data—This bit is the data for the onboard serial devices—the calibration EEPROM and the serial DACs. This bit should be set to the desired value prior to the active write to the SerClk bit.
0	SerClk	Serial Clock—This bit is the clock input to the two onboard serial devices. In order to write to these devices, this bit should be set first to 0 and then to 1. The data in the SerData bit will be written to the devices on the low-to-high transition of the serial clock.

### Misc Command Register

The Misc Command Register contains one bit that controls the AT-MIO E Series analog trigger source. The contents of this register are cleared upon power up and after a reset condition.

Address: Base address + 0F (hex)

Type: Write-only

Word Size: 8-bit

Bit Map:

7	6	5	4	3	2	1	0
Int/Ext Trig	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved

Bit	Name	Description
7	Int/Ext Trig	Internal/External Analog Trigger—This bit controls the analog trigger source. If this bit is set, the output of the PGIA2 is selected as the trigger source. If this bit is cleared, the TRIG1 signal from the I/O connector is selected as the trigger source.
6–0	Reserved	Reserved—Always write 0 to these bits.

## Status Register

The Status Register is used to indicate the status of the DMATCs and the calibration EEPROM output.

Address: Base address + 01 (hex)

Type: Read-only

Word Size: 8-bit

Bit Map:

7	6	5	4	3	2	1	0
Reserved	Reserved	Reserved	Reserved	DMATCC	DMATCB	DMATCA	PROMOUT

Bit	Name	Description
7–4	Reserved	Reserved—Always write 0 to these bits.
3	DMATCC	DMA Terminal Count C—This bit indicates the status of the DMA process on the selected DMA Channel C. When all transfers have been completed, DMATCC will be set. This bit is cleared by the DmaTcCClr bit in the Strokes Register.
2	DMATCB	DMA Terminal Count B—This bit indicates the status of the DMA process on the selected DMA Channel B. When all transfers have been completed, DMATCB will be set. This bit is cleared by the DmaTcBClr bit in the Strokes Register.
1	DMATCA	DMA Terminal Count A—This bit indicates the status of the DMA process on the selected DMA Channel A. When all transfers have been completed, DMATCA will be set. This bit is cleared by the DmaTcAClr bit in the Strokes Register.
0	PROMOUT	EEPROM Output Data—This bit reflects the serial output data of the serial EEPROM.

## Analog Input Register Group

The three registers making up the Analog Input Register Group control the analog input circuitry and can be used to read the ADC FIFO contents. Reading the ADC FIFO Data Register location transfers data from the AT-MIO E Series ADC data FIFO to the PC. Writing to the Configuration Memory Low and Configuration Memory High Register locations sets up channel configuration information for the analog input section. This information is necessary for single conversions as well as for single- and multiple-channel data acquisition sequences.

## ADC FIFO Data Register

The ADC FIFO Data Register returns the oldest ADC conversion value stored in the ADC FIFO. Reading the ADC FIFO removes that value and leaves space for another ADC conversion value to be stored. Values are shifted into the ADC FIFO whenever an ADC conversion is complete unless the GHOST bit is set in that entry of the Configuration Memory.

The ADC FIFO is emptied when all values it contains are read. The empty, half-full, and full flags from the ADC data FIFO are available in a status register in the DAQ-STC. These flags indicate when the FIFO is empty, half-full, or full, respectively. Whenever the FIFO is not empty, the stored data can be read from the ADC FIFO Data register.

The values returned by reading the ADC Data Register are available in two different binary formats: straight binary, which generates only positive numbers, or two's complement binary, which generates both positive and negative numbers. The binary format used is determined by the mode in which the ADC is configured. Following is the bit pattern returned for either format:

Address: Base address + 1C (hex)

Type: Read-only

Word Size: 16-bit

Bit Map:

15	14	13	12	11	10	9	8
D15	D14	D13	D12	D11	D10	D9	D8
7	6	5	4	3	2	1	0
D7	D6	D5	D4	D3	D2	D1	D0

Bit	Name	Description
15–0	D<15..0>	Data bits 15 through 0—These bits are the result of the ADC conversion. The boards with a 12-bit ADC will return values ranging from 0 to 4,095 decimal (0x0000 to 0x0FFF) when the ADC is in unipolar mode, and -2,048 to 2,047 decimal (0xF800 to 0x07FF) when the ADC is in bipolar mode. The boards with a 16-bit ADC will return values ranging from 0 to 65,535 decimal (0x0000 to 0xFFFF) when the ADC is in unipolar mode and 32,768 to 32,767 decimal (0x8000 to 0x7FFF) when the ADC is in bipolar mode. The mode is controlled by the Unip/Bip bit in the Configuration Memory Low Register.

## Configuration Memory Low Register

The Configuration Memory Low Register works with the Configuration Memory High Register to control the input channel selection multiplexers, gain, range, and mode settings. The values written to these registers are placed into the Configuration Memory, which is sequenced through during an acquisition. This register contains seven of these bits. The contents of the Configuration Memory are emptied by a control register in the DAQ-STC.

Address: Base address + 10 (hex)

Type: Write-only

Word Size: 16-bit

Bit Map:

15	14	13	12	11	10	9	8
LastChan	Reserved	Reserved	GenTrig	Reserved	Reserved	DitherEn	Unip/Bip
7	6	5	4	3	2	1	0
Reserved	Reserved	Reserved	Reserved	Reserved	Gain2	Gain1	Gain0

Bit	Name	Description
15	LastChan	Last Channel—This bit should be set in the last entry of the scan sequence loaded into the channel configuration memory. More than one occurrence of the LastChan bit is possible in the configuration memory list for the interval-scanning mode. For example, there can be multiple scan sequences in one memory list.
14–13, 11–10, 7–3	Reserved	Reserved—Always write 0 to these bits.
12	GenTrig	General Trigger—This bit synchronizes actions in the DAQ-STC with the scan list. When this bit is set, an active low trigger pulse is sent into the RTSI_BRD0 input of the DAQ-STC during the CONVERT signal. This trigger is used at the application level and can perform such actions as time stamping and starting waveform generation.
9	DitherEn	Dither Enable—This bit controls the dither circuitry feeding the analog input. If this bit is set, approximately $\pm 0.5$ LSB of white Gaussian noise is added to the input signal. This bit is reserved on the AT-MIO-16XE-50 and should be set to 1. Dither cannot be disabled on the AT-MIO-16XE-50.



Bit	Name	Description (Continued)
8	Unip/Bip	Channel Unipolar/Bipolar—This bit configures the ADC for unipolar or bipolar mode. When Unip/Bip is set, the ADC is configured for unipolar operation and values read from the ADC Data Register are in straight binary format. When Unip/Bip is clear, the ADC is configured for bipolar operation and values. The data values are two's complement and automatically sign extended.
2-0	Gain<2..0>	Channel Gain Select 2 through 0—These three bits control the gain settings of the input PGIA for the selected analog channel. The following gains can be selected on the AT-MIO E Series:

Table 3-3. PGIA Gain Selection

Gain<2..0>	Actual Gain
000	0.5*
001	1
010	2
011	5*
100	10
101	20*
110	50*
111	100

\*Not supported on the AT-MIO-16XE-50

### Configuration Memory High Register

The Configuration Memory High Register works with the Configuration Memory Low Register to control the input channel selection multiplexers, gain, range, and mode settings. This register contains nine of these bits. The contents of this register are cleared by a control register in the DAQ-STC.

Address: Base address + 12 (hex)

Type: Write-only

Word Size: 16-bit

Bit Map:

15	14	13	12	11	10	9	8
Reserved	ChanType2	ChanType1	ChanType0	Reserved	Reserved	Reserved	Reserved
7	6	5	4	3	2	1	0
Reserved	Reserved	Bank1	Bank0	Chan3	Chan2	Chan1	Chan0

<b>Bit</b>	<b>Name</b>	<b>Description</b>
------------	-------------	--------------------

15, 11–6	Reserved	Reserved—Always write 0 to these bits.
----------	----------	--

14–12	ChanType<2..0>	Channel Type 2 through 0—These bits indicate which type of resource is active for the current entry in the scan list. The following table lists the valid channel types.
-------	----------------	--

Type<2..0>	Resource
000	Calibration
001	Differential
010	NRSE
011	RSE
100	X
101	X
110	X
111	Ghost

5–4	Bank<1..0>	Bank Select 1 through 0—These bits indicate which bank of channels is active for the current resource in the scan list. All AT-MIO-16 or higher boards segment the channels into banks of 16. Not every resource uses all of the bank bits. For example, the AT-MIO-64E-3 uses four banks of channels for the DIFF, NRSE, and RSE resources, but only one bank for CAL.
-----	------------	---

Bit	Name	Description (Continued)
3–0	Chan<3..0>	Channel Select 3 through 0—These bits indicate which channel is active for the current resource in the scan list. Not every resource uses all 16 channels in a bank. Channel assignments for all AT-MIO-16 or higher boards follow.

Table 3-4. Calibration Channel Assignments

Type<2..0> = CAL			
Chan<3..0>	PGIA(+)	PGIA(-)	Purpose
0000	AOGND*	AOGND*	ADC Offset
0001	AOGND	AIGND	Ground Differential
0010	DAC0OUT	AOGND	DAC 0 Offset/Linearity
0011	DAC1OUT	AOGND	DAC 1 Offset/Linearity
0100	REF5V*	REF5V*	ADC Offset
0001	REF5V	AI GND	ADC Gain
0110	DAC0OUT	REF5V	DAC 0 Gain
0111	DAC1OUT	REF5V	DAC 1 Gain
1xxx	Reserved	Reserved	—

\* AIGND on the AT-MIO-16XE-50

Table 3-5. Differential Channel Assignments

Type<2..0> = DIFF		
Chan<3..0>	PGIA(+)	PGIA(-)
0000	ACh0	ACh8
0001	ACh1	ACh9
0010	ACh2	ACh10
0011	ACh3	ACh11
0100	ACh4	ACh12
0001	ACh5	ACh13
0110	ACh6	ACh14
0111	ACh7	ACh15
1xxx	Reserved	Reserved

Table 3-6. Nonreferenced Single-Ended Channel Assignments

<b>Type&lt;2..0&gt; = NRSE</b>		
<b>Chan&lt;3..0&gt;</b>	<b>PGIA(+)</b>	<b>PGIA(-)</b>
0000	ACh0	AI Sense
0001	ACh1	AI Sense
0010	ACh2	AI Sense
0011	ACh3	AI Sense
0100	ACh4	AI Sense
0101	ACh5	AI Sense
0110	ACh6	AI Sense
0111	ACh7	AI Sense
1000	ACh8	AI Sense
1001	ACh9	AI Sense
1010	ACh10	AI Sense
1011	ACh11	AI Sense
1100	ACh12	AI Sense
1001	ACh13	AI Sense
1110	ACh14	AI Sense
1111	ACh15	AI Sense

Table 3-7. Referenced Single-Ended Channel Assignments

<b>Type&lt;2..0&gt; = RSE</b>		
<b>Chan&lt;3..0&gt;</b>	<b>PGIA(+)</b>	<b>PGIA(-)</b>
0000	ACh0	AI Ground
0001	ACh1	AI Ground
0010	ACh2	AI Ground
0011	ACh3	AI Ground
0100	ACh4	AI Ground
0101	ACh5	AI Ground
0110	ACh6	AI Ground
0111	ACh7	AI Ground
1000	ACh8	AI Ground
1001	ACh9	AI Ground
1010	ACh10	AI Ground
1011	ACh11	AI Ground
1100	ACh12	AI Ground
1001	ACh13	AI Ground
1110	ACh14	AI Ground
1111	ACh15	AI Ground

Table 3-8. Channel Assignments

<b>Type&lt;2..0&gt; = GHOST</b>	
<b>Chan&lt;3..0&gt;</b>	<b>Purpose</b>
xxxx	Used to preserve timing for multirate sampling. No acquisition is performed for this scan entry.

## Analog Output Register Group

The four registers making up the Analog Output Register Group access the two analog output channels, the analog output FIFO, and the analog output configuration. Data can be transferred to the DACs in one of two ways. Data can be directly sent to the DACs from the host computer, or buffered from the host by the DAC data FIFO.

## AO Configuration Register

The AO Configuration Register contains 5 bits that control the AT-MIO E Series analog output configuration. The contents of this register are cleared upon power up and after a reset condition.

Address: Base address + 16 (hex)

Type: Write-only

Word Size: 16-bit

Bit Map:

15	14	13	12	11	10	9	8
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	DACSel
7	6	5	4	3	2	1	0
Reserved	Reserved	Reserved	Reserved	GroundRef	ExtRef	ReGlitch	BipDac

Bit	Name	Description
15–9, 7–4	Reserved	Reserved—Always write 0 to these bits.
8	DACSel	DAC Select—This bit indicates which DAC is the destination for the configuration bits in this register. DAC0 will be selected when this bit is cleared, and DAC1 will be selected when set.
3	GroundRef	Ground Reference—This bit connects the reference for both DACs to ground when this bit is set. This is useful for calibration of the DAC linearity. This bit is not currently implemented as a separate selection for the two DACs. Therefore, its state is determined by the last value written to this bit field. This bit is reserved on the AT-MIO-16XE-50 and should be set to 0.
2	ExtRef	External Reference for DAC—This bit controls the reference selection for the selected DAC. If this bit is set, the reference used for the DAC is the external reference voltage from the I/O connector. If this bit is cleared, the internal +10 V <sub>ref</sub> is used for the DAC reference. This bit is reserved on the AT-MIO-16XE-50 and should be set to 0.
1	ReGlitch	Reglitch DAC—When set, this bit configures the selected DAC to have a more uniform glitch when changing the output at the expense of a higher average glitch energy. This bit is reserved on the AT-MIO-16E-10, AT-MIO-16DE-10, and AT-MIO-16XE-50 and should be set to 0.

0	BipDac	Bipolar DAC—This bit configures the voltage range of the selected DAC. If this bit is set, then the DAC is configured for bipolar operation of $-V_{ref}$ to $+V_{ref}$ . In this mode, data written to the DAC is interpreted in two's complement format. If this bit is cleared, then the DAC is configured for unipolar operation of 0 V to $+V_{ref}$ . In this mode, data written to the DAC is interpreted in straight binary format. This bit is reserved on the AT-MIO-16XE-50 and should be set to 1. The AT-MIO-16XE-50 DACs are always configured in bipolar mode.
---	--------	---

### DAC FIFO Data Register

The DAC FIFO Data Register is used to load the desired data into the DAC data FIFO.

The DAC FIFO is 2 kwords deep on the AT-MIO-16E-1, AT-MIO-16E-2, and AT-MIO-64E-3. The DAC FIFO is 0 words deep on the AT-MIO-16E-10, AT-MIO-16DE-10, and AT-MIO-16XE-50. The empty, half-full, and full flags from the 2 kword DAC data FIFO are available in a status register in the DAQ-STC. Only the full flag is available on the boards with the 0 word deep DAC data virtual FIFO. These flags indicate when the FIFO is empty, half-full, or full, respectively. Whenever the FIFO is not full the host is free to write additional data.

Address: Base address + 1E (hex)

Type: Write-only

Word Size: 16-bit

Bit Map:

15	14	13	12	11	10	9	8
Reserved	Reserved	Reserved	Reserved	D11	D10	D9	D8
7	6	5	4	3	2	1	0
D7	D6	D5	D4	D3	D2	D1	D0

Bit	Name	Description
15–12	Reserved	Reserved—Always write 0 to these bits.
11–0	D<11..0>	Data bits 11 through 0—The 12-bit data to be written to the DAC data FIFO. This data is interpreted in straight binary form when DAC1 is configured for unipolar operation. When DAC1 is configured for bipolar operation, the data is interpreted in two's complement form.

## DAC0 Direct Data Register

The DAC0 Direct Data Register loads the desired data directly into DAC0, without using the DAC data FIFO.

Address: Base address + 18 (hex)

Type: Write-only

Word Size: 16-bit

Bit Map:

15	14	13	12	11	10	9	8
Reserved	Reserved	Reserved	Reserved	D11	D10	D9	D8
7	6	5	4	3	2	1	0
D7	D6	D5	D4	D3	D2	D1	D0

Bit	Name	Description
15–12	Reserved	Reserved—Always write 0 to these bits.
11–0	D<11:0>	Data bits 11 through 0—The 12-bit data to be written to DAC0, itself. This data is interpreted in straight binary form when DAC0 is configured for unipolar operation. When DAC0 is configured for bipolar operation, the data is interpreted in two's complement form.



### DAC1 Direct Data Register

The DAC1 Direct Data Register loads the desired data directly into DAC1, without using the DAC data FIFO.

Address: Base address + 1A (hex)

Type: Write-only

Word Size: 16-bit

Bit Map:

15	14	13	12	11	10	9	8
Reserved	Reserved	Reserved	Reserved	D11	D10	D9	D8
7	6	5	4	3	2	1	0
D7	D6	D5	D4	D3	D2	D1	D0

Bit	Name	Description
15–12	Reserved	Reserved—Always write 0 to these bits.
11–0	D<11:0>	Data bits 11 through 0—The 12-bit data to be written to DAC1 itself. This data is interpreted in straight binary form when DAC1 is configured for unipolar operation. When DAC1 is configured for bipolar operation, the data is interpreted in two's complement form.

## DMA Control Register Group

The six registers making up the DMA Control Register Group configure the AT-MIO E Series boards DMA interface. The AI AO Select and G0 G1 Select Registers select single or dual DMA modes for the analog input, analog output, and general purpose counter timer resources. The Channel A Mode, Channel B Mode, and Channel C Mode Registers contain the physical DMA channel selections, the transfer type, and channel and interrupt enables. The Strobes Register contains the DMA TC Acknowledge bits.

The AT-MIO E Series boards support three logical channels, which can be chosen from any of the available 16-bit physical channels. The ISA bus supports three 16-bit channels, while there are seven 16-bit DMA channels available on the EISA bus.

### Strobes Register

The Strobes Register contains 3 bits that clear the DMA terminal count status bits. These bits are cleared by hardware after each write. The contents of this register are cleared upon power up and after a reset condition.

Address: Base address + 01 (hex)

Type: Write-only

Word Size: 8-bit

Bit Map:

7	6	5	4	3	2	1	0
Reserved	Reserved	Reserved	Reserved	Reserved	DmaTcCClr	DmaTcBClr	DmaTcAClr

Bit	Name	Description
7–3	Reserved	Reserved—Always write 0 to these bits.
2	DmaTcCClr	DMA Terminal Count C Clear—Clears the DMA TC status bit for logical channel C. This action also removes the DMA TC interrupt if caused by logical channel C.
1	DmaTcBClr	DMA Terminal Count B Clear—Clears the DMA TC status bit for logical channel B. This action also removes the DMA TC interrupt if caused by logical channel B.
0	DmaTcAClr	DMA Terminal Count A Clear—Clears the DMA TC status bit for logical channel A. This action also removes the DMA TC interrupt if caused by logical channel A.

## Channel A Mode Register

The Channel A Mode Register contains 8 bits that control the logical DMA channel A. The contents of this register are cleared upon power up and after a reset condition.

Address: Base address + 03 (hex)

Type: Write-only

Word Size: 8-bit

Bit Map:

7	6	5	4	3	2	1	0
ChanEnable	TCIntEnable	Transfer2	Transfer1	Transfer0	Channel2	Channel1	Channel0

Bit	Name	Description
7	ChanEnable	DMA Channel A Enable—This bit enables DMA channel A to process DMA request/acknowledge cycles. If this bit is set, any requests on channel A are translated to the physical DMA interface. If this bit is cleared, any logical requests are ignored. All other setup for logical channel A should be performed prior to setting this bit. This bit must be cleared between acquisition sequences to properly initialize the circuitry.
6	TCIntEnable	DMATCA Interrupt Enable—This bit enables an interrupt to be generated whenever a DMATCA occurs. The interrupts are enabled when this bit is set and disabled when cleared.
5–3	Transfer<2..0>	Transfer Type 2 through 0—These three bits determine the size and type of the DMA transfer. If Transfer<2..0> is set to 0, DMA requests are generated for single transfers. If Transfer<2..0> is set to 1, the DMA request is asserted for up to eight transfers without re arbitration.
2–0	Channel<2..0>	Physical Channel Select 2 through 0—These three bits select which physical channel is associated with logical channel A. The AT-MIO E Series can use only 16-bit DMA channels, so these bits must be set appropriately.

## Channel B Mode Register

The Channel B Mode Register contains 8 bits that control the logical DMA channel B. The contents of this register are cleared upon power up and after a reset condition.

Address: Base address + 05 (hex)

Type: Write-only

Word Size: 8-bit

Bit Map:

7	6	5	4	3	2	1	0
ChanEnable	TCIntEnable	Transfer2	Transfer1	Transfer0	Channel2	Channel1	Channel0

Bit	Name	Description
7	ChanEnable	DMA Channel B Enable—This bit enables DMA channel B to process DMA request/acknowledge cycles. If this bit is set, any requests on channel B are translated to the physical DMA interface. If this bit is cleared, any logical requests are ignored. All other setup for logical channel B should be performed prior to setting this bit. This bit must be cleared between acquisition sequences to properly initialize the circuitry.
6	TCIntEnable	DMATCB Interrupt Enable—This bit enables an interrupt to be generated whenever a DMATCB occurs. The interrupts are enabled when this bit is set and disabled when cleared.
5–3	Transfer<2..0>	Transfer Type 2 through 0—These three bits determine the size and type of the DMA transfer. If Transfer<2..0> is set to 0, DMA requests are generated for single transfers. If Transfer<2..0> is set to 1, the DMA request is asserted for up to eight transfers without re arbitration.
2–0	Channel<2..0>	Physical Channel Select 2 through 0—These three bits select which physical channel is associated with logical channel B. The AT-MIO E Series can use only 16-bit DMA channels, so these bits must be set appropriately.

## Channel C Mode Register

The Channel C Mode Register contains 8 bits that control the logical DMA channel C. The contents of this register are cleared upon power up and after a reset condition.

Address: Base address + 07 (hex)

Type: Write-only

Word Size: 8-bit

Bit Map:

7	6	5	4	3	2	1	0
ChanEnable	TCIntEnable	Transfer2	Transfer1	Transfer0	Channel2	Channel1	Channel0

Bit	Name	Description
7	ChanEnable	DMA Channel C Enable—This bit enables DMA channel C to process DMA request/acknowledge cycles. If this bit is set, any requests on channel C are translated to the physical DMA interface. If this bit is cleared, any logical requests are ignored. All other setup for logical channel C should be performed prior to setting this bit. This bit must be cleared between acquisition sequences to properly initialize the circuitry.
6	TCIntEnable	DMATCC Interrupt Enable—This bit enables an interrupt to be generated whenever a DMATCC occurs. The interrupts are enabled when this bit is set and disabled when cleared.
5–3	Transfer<2..0>	Transfer Type 2 through 0—These three bits determine the size and type of the DMA transfer. If Transfer<2..0> is set to 0, DMA requests are generated for single transfers. If Transfer<2..0> is set to 1, the DMA request is asserted for up to eight transfers without re arbitration.
2–0	Channel<2..0>	Physical Channel Select 2 through 0—These three bits select which physical channel is associated with logical channel C. The AT-MIO E Series can use only 16-bit DMA channels, so these bits must be set appropriately.

## AI AO Select Register

The AI AO Select Register contains 6 bits that control the logical DMA selection for the analog input and analog output resources. The contents of this register are cleared upon power up and after a reset condition.

Address: Base address + 09 (hex)

Type: Write-only

Word Size: 8-bit

Bit Map:

7	6	5	4	3	2	1	0
Reserved	Output C	Output B	Output A	Reserved	Input C	Input B	Input A

Bit	Name	Description
7, 3	Reserved	Reserved—Always write 0 to this bit.
6–4	Output <C..A>	Analog Output Logical Channel C through A—These three bits select the logical channels to be used by the analog output. This resource supports both single and dual channel DMA; therefore, either one or two of the three bits must be set for proper operation. These bits must be set prior to enabling the logical channels.
2–0	Input <C..A>	Analog Input Logical Channel C through A—These three bits select the logical channels to be used by the analog input. The AT-MIO-16E-1 supports both single-channel and dual-channel DMA; therefore, either one or two of the three bits must be set for proper operation. All other AT-MIO E Series boards support only single channel DMA; therefore, only one of the three bits must be set for proper operation. These bits must be set prior to enabling the logical channels.

## G0 G1 Select Register

The G0 G1 Select Register contains 6 bits that control the logical DMA selection for the two general purpose counter timer resources. The contents of this register are cleared upon power up and after a reset condition.

Address: Base address + 0B (hex)

Type: Write-only

Word Size: 8-bit

Bit Map:

7	6	5	4	3	2	1	0
Reserved	GPCT1 C	GPCT1 B	GPCT1 A	Reserved	GPCT0 C	GPCT0 B	GPCT0 A

Bit	Name	Description
7, 3	Reserved	Reserved—Always write 0 to this bit.
6–4	GPCT1 <C..A>	General Purpose Counter Timer 1 Logical Channel C through A— These three bits select the logical channels that the GPCT1 uses. This resource supports both single and dual-channel DMA; therefore, either one or two of the three bits must be set for proper operation. The circuitry initializes so that the first channel to transfer data is the first alphabetically. These bits must be set prior to enabling the logical channels.
2–0	GPCT0 <C..A>	General Purpose Counter Timer 0 Logical Channel C through A— These three bits select the logical channels that the GPCT1 uses. This resource supports both single and dual-channel DMA; therefore, either one or two of the three bits must be set for proper operation. The circuitry initializes so that the first channel to transfer data is the first alphabetically. These bits must be set prior to enabling the logical channels.

## DIO Register Group

The four registers making up the DIO Register Group configure and control the 8255 digital I/O integrated circuit used on the AT-MIO-16DE-10. These registers are described in the 8255 data sheet, included in Appendix A. Note that the port C interrupt lines, PC0 and PC3, are ORed together on the board and connected to the DAQ-STC group A pass-through interrupt.

## DAQ-STC Register Group

The registers making up the DAQ-STC Register Group configure and control the DAQ-STC system timing controller ASIC. These registers are described in the *DAQ-STC Technical Reference Manual*.

## FIFO Strobe Register Group

The three registers making up the FIFO Strobe Register Group are used to clear the three FIFOs on the AT-MIO E Series.

### Configuration Memory Clear Register

Accessing the Configuration Memory Clear Register clears all information in the channel configuration memory and resets the write pointer to the first location in the memory.

Window Address:	52 (hex)
Type:	Write-only
Word Size:	16-bit
Bit Map:	Not applicable; no bits used

### ADC FIFO Clear Register

Accessing the ADC FIFO Clear Register clears all information in the ADC data FIFO.

Window Address:	53 (hex)
Type:	Write-only
Word Size:	16-bit
Bit Map:	Not applicable; no bits used



**DAC FIFO Clear Register**

Accessing the DAC FIFO Clear Register clears all information in the DAC data FIFO.

Window Address: 54 (hex)

Type: Write-only

Word Size: 16-bit

Bit Map: Not applicable; no bits used

# Chapter 4

## Programming

---

This chapter contains programming instructions for operating the circuitry on the AT-MIO E Series boards.

Programming the AT-MIO E Series boards involves writing to and reading from registers on the board. Many of these registers belong to the DAQ-STC, and you will find a listing of these registers in the *DAQ-STC Technical Reference Manual*. For a list of board discrete registers, not on the DAQ-STC, see Chapter 3, *Register Map and Descriptions*, of this manual.

The DAQ-STC has a set of Board Environment Registers that have to be initialized once at the beginning of configuration. These registers have to be initialized based on the properties of the hardware surrounding the DAQ-STC.

The programming steps for analog input, output, digital I/O, and timing I/O are explained in the *DAQ-STC Technical Reference Manual*. These operations are explained in terms of bitfields. A bitfield is defined as a group of contiguous bits that jointly perform a function. Using bitfields has the advantage of establishing an efficient mapping technique to the underlying hardware. Also, the programming style becomes very modular, and a functional description of every programming step is easily understood.

Because the *DAQ-STC Technical Reference Manual* has detailed, step-by-step programming instructions, this register-level programmer manual gives a series of common programming examples with references to the *DAQ-STC Technical Reference Manual*. The program for each example is presented as it is in the *DAQ-STC Technical Reference Manual*; that is, in terms of functions, with bitfields to be written for each function. To implement the function, you will need to perform a write to or a read from the specified registers. The complete examples are provided on the *AT-MIO E Series Register-Level Programmer Manual* Companion Disk.

## Plug and Play Initialization

The AT-MIO E Series boards are fully compatible with the Intel Plug and Play Specification version 1.0. The Plug and Play system arbitrates and assigns resources through software, freeing the user from manually setting switches and jumpers. The bus-related resources must be configured before attempting to execute a register level program. Both the bus-related and data acquisition-related Plug and Play configurations are described in the *AT-MIO E Series User Manual*.

If your system has a Plug and Play configuration manager which assigns available resources at startup, you must query the configuration manager to determine the resources assigned to each board. If no configuration manager is installed, then a configuration program is necessary to assign resources to the board. The *AT-MIO E Series Register-Level Programmer Manual* Companion Disk contains a function that a user can execute to configure the base address of the

AT-MIO E Series board. It is very important to note that the given Plug and Play initialization only applies for one board. If more than one board needs to be configured, the Plug and Play specification explains the procedure.

The software implementation of the full Plug and Play protocol is beyond the scope of this manual, and is discussed in the ISA Plug and Play Specification version 1.0 available from Intel. This section discusses how to assign a base I/O address to an AT-MIO E Series board. All access to the Plug and Play registers is performed through three I/O address locations. The Address Port is at location 0x279, and is used to indicate which configuration register is the target of the next data transfer. The Write Data Port is at location 0xA79, and is used to write to the configuration register indicated by the Address Port contents. The Read Data Port is moveable in the range of 0x203 to 0x3FF, and is used to read the configuration register indicated by the Address Port contents. The location of the Read Data Port is set by writing to certain configuration registers.

Access to the Plug and Play configuration registers is controlled by an internal state machine. The state machine consists of four states: Wait4Key, Sleep, Isolation, and Configuration. The board powers up in the Wait4Key state, where the board is inactive and not yet configured. A sequence of specific accesses to the Address Port will move the board into the Sleep state. This long sequence of 34 writes was chosen to decrease the chances that a poorly written program that writes randomly into I/O space does not accidentally access the Plug and Play configuration registers. The function `Write_PNP_Initiation_Key` performs this series of writes.

The board will now be in the Sleep state where it is waiting to wake up. If the board has been previously configured, you may wish to reset all of the Plug and Play configuration registers with the function `Reset_PNP_Registers`. Note that resetting the registers will return the state machine to the Wait4Key state. You will need to write the Initiation Key to the board once more to return to the Sleep state.

The board can now be moved into the Isolation state. The Isolation state is used to identify a single Plug and Play board from all of the boards in the system. Since this code will only support a single Plug and Play board in the system, no action is actually performed during the Isolation state. The `Set_PNP_Isolation_State` will move the board from the Sleep to the Isolation state.

Since no actions need to be performed in the Isolation state, the board can now move into the Configuration state. The board's resource assignments can only be made from the Configuration state, where we want to set the base I/O address. `Set_PNP_Configuration_State` will move the board from the Isolation to the Configuration state.

The base I/O address for the board can now be assigned. The E Series boards use full 16 bit decoding, so both the high and low bytes of the base I/O address must be assigned. This assignment is accomplished by running `Assign_PNP_Base_Address`.

So far there has been no feedback to the program to see if everything is OK. The base address registers can be read to verify that they were written correctly. First, the Read Data Port must be assigned to a free location. The Read Data Port can be located at any location between 0x203 and 0x3FF where the lowest two bits of the address are set, for example: 0x203, 0x207, 0x20B, ..., 0x3FB, 0x3FF. You must write the value of the address right shifted by two to the configuration register, as shown in the function `Assign_PNP_Data_Port_Address`.

You can now read the contents of the base I/O address registers with the function `Verify_PNP_Base_Address`. These values should match those that were written earlier in the program.

The board now needs to be activated for the assigned base I/O address to take effect. Once activated, the board will return to the `Wait4Key` state. This activation is accomplished with `Activate_PNP_Board`.

The registers for the E Series board can now be accessed by their offset from the base I/O address that was assigned above. It is important to note that the code given above must be repeated every time that the computer is rebooted or powered off. The hardware stores the assigned base I/O address in an onboard register, but not in any form of nonvolatile memory. The E Series board will "forget" what its base I/O address is after a reboot.

## Windowing Registers

Since the DAQ-STC contains a large number of registers (180), eight address lines would be required to decode the addresses in direct address mode. In addition to the DAQ-STC registers, the AT-MIO E Series boards have other discrete registers. This enormous I/O address space makes it impossible to use other peripheral cards with their own registers.

The windowing scheme allows a smaller address space requirement for the DAQ-STC at the expense of requiring more accesses to perform the same task. To write to a register in Windowed mode, write the address offset to the `Window_Address_Register`. Write the bit pattern (data) to the `Window_Data_Write_Register`. Similarly, to read from a register in Windowed mode, write the address offset to the `Window_Address_Register`; read from the `Window_Data_Read_Register`.

## Programming Examples

The programs presented in this chapter are broken into three sections: Digital I/O, Analog Input, and Analog Output. Each example provides the pseudo-code for the main program. The examples follow the procedure presented in detail in the *DAQ-STC Technical Reference Manual*, and further explanation of the functions can be found in this manual. Each program is also provided in its entirety on the *AT-MIO E Series Register Level Programmer Manual Companion Disk*.

The Companion Disk also contains the support files necessary to run the examples. The support files include `PNPINIT.c`, `ESERRLP.h`, and `ESERFNCT.c` which should all be included with the project for each example. `PNPINIT.c` includes the `Setup_PNP_Board` function which assigns the base I/O address as discussed in the Plug and Play Initialization section of this manual; `ESERRLP.h` declares the external function prototypes and the register addresses; `ESERFNCT.c` contains functions which write to and read from Windowed and board discrete registers: `DAQ_STC_Windowed_Mode_Write`, `DAQ_STC_Windowed_Mode_Read`, `Board_Write`, and `Board_Read`.

Before beginning register-level programming of the analog input and analog output modules, you should test the Windowed addressing scheme. To test the Windowed addressing scheme, use a simple example to operate on the DIO lines. When this example works, try to write code for the other modules.

As mentioned in Chapter 2 of the *DAQ-STC Technical Reference Manual*, several write-only registers on the DAQ-STC contain bitfields that control a number of functionally independent parts of the chip. To update bitfields, you must set or clear bits without changing the status of the remaining bits in the register. Since you cannot read these registers to determine which bits are set, you should maintain a software copy of the write-only registers.

**Note:** *For simplicity that these examples do not include software copies of the registers. If you wish to write your own examples, or modify these examples, we strongly recommend adding software copies of the write-only registers. Please refer to Chapter 2, Register and Bitfield Programming Considerations, in the DAQ-STC Technical Reference Manual for further information.*

## Digital I/O

Chapter 7 of the *DAQ-STC Technical Reference Manual* describes the DIO module of the DAQ-STC and illustrates an example (C language) in Windowed mode to toggle the DIO lines. Example 1 verifies that the Windowed addressing scheme works. Example 2 illustrates digital I/O.

### Example 1

This example illustrates the use of Windowed registers by toggling the digital lines.

First configure all the digital lines as outputs. Write 0x0 through 0xFF to the DIO output register and make sure the digital lines toggle. (This example is also presented in Chapter 7 of the *DAQ-STC Technical Reference Manual*.)

1. Set up the Plug and Play resources. Use the function `Setup_PNP_Board` provided on the Companion Disk.
2. Configure all the digital lines as outputs.

```
DIO_Control_Register = 0xFF;
```

3. Output the digital patterns.

```
for (i=0;i<=255;i++)  
{  
    DIO_Output_Register = i;  
}
```

## Example 2

This example shows how to perform digital I/O.

Configure digital lines 0, 2, 4, and 6 as outputs and the remaining lines as inputs. Wrap back the output lines to the input lines. Write patterns 0b0000 and 0b1111 to the output lines. Read back the input pins to verify the data. Also check some intermediate patterns as well.

1. Set up the Plug and Play resources. Use the function `Setup_PNP_Board` provided on the Companion Disk.
2. Configure lines 0, 2, 4, and 6 as outputs and 1, 3, 5, and 7 as inputs.

```
DIO_Control_Register = 0x55;
```

3. Write the digital pattern.

```
DIO_Output_Register = 0x00;
```

4. Read the digital pattern.

```
Pattern = DIO_Parallel_Input_Register;
```

5. Repeat Steps 3 and 4 for subsequent patterns.

## Analog Input

See Chapter 2 in the *DAQ-STC Technical Reference Manual* for information on programming analog input and the relevant registers. This section also describes the programming sequences for the various functions and organizes these functions for a particular operation. Individual bitfield descriptions are also given.

Programming the AT-MIO E Series boards for analog input can be divided into writing to and reading from two main register groups: discrete board registers, and DAQ-STC registers. The following functions configure the board by calling the `Board_Read` and `Board_Write` functions:

```
Setup_PNP_Board;
Configure_Board;
```

Analog input DAQ-STC programming consists of the following functions which call the `DAQ_STC_Windowed_Mode_Read` and `DAQ_STC_Windowed_Mode_Write` functions:

```
AI_Initialize_Configuration_Memory_Output,
MSC_Clock_Configure,
Clear_FIFO,
AI_Reset_All,
AI_Board_Personalize,
FIFO_Request,
```

```

AI_Hardware_Gating,
AI_Trigger_Signals,
Number_of_Scans,
AI_Scan_Start,
AI_End_of_Scan,
Convert_Signal,
AI_Interrupt_Enable,
AI_Arming,
AI_Start_The_Acquisition,

```

As the analog input examples increase in complexity, more of these functions will be necessary. The functions will be presented in the applicable examples; subsequent examples address only the specific differences from Example 1. This manual provides the structure and pseudo-code for each example. The *AT-MIO E Series Register Level Programmer Manual Companion Disk* contains the complete programs. The following pseudo-code examples and the programs on the Companion Disk follow the flowchart structure presented in the *DAQ-STC Technical Reference Manual*.

### Example 1

This example acquires one sample from channel 0.

Connect a voltage source to ACH0 on the I/O connector. Channel 0 should be configured for bipolar RSE with no dithering. No external multiplexers are present. Sample analog input channel 0 at a gain of 1 and read the unscaled result. Compare the unscaled value to the input voltage.

The first two steps set up the E-Series board, and the subsequent steps configure the DAQ-STC.

1. Set up the Plug and Play resources. Use the function `Setup_PNP_Board` provided on the Companion Disk.
2. The second step configures the analog channel for the given settings.

The function `Configure_Board` clears the configuration memory, clears the ADC FIFO, and then sets channel 0 to the given settings. (Clearing the configuration memory and ADC FIFO require windowed mode writes as well as board writes.)

```

Write_Strobe_0_Register
    Write strobe 0 = 1;

```

```

Write_Strobe_1_Register
    Write strobe 1 = 1;

```

```

Configuraton_Memory_High_Register
    Channel number = 0;
    Channel type = 3;

```

```

Configuration_Memory_Low_Register
    Last channel = 1;

```

```
Gain = 1;
Polarity = 0;
Dither enable = 0;
```

3. The programming of the DAQ-STC begins with the clock configuration.

The function `MSC_Clock_Configure` selects the timebase for the DAQ-STC.

```
Clock_and_FOUT_Register
Slow internal timebase = 1;
Divide timebase by two = 1;
Clock to board = 1;
Board divide by two = 1;
```

4. Call the function `Clear_FIFO` to clear the ADC FIFO.

```
Write_Strobe_1_Register
Write strobe 1 = 1;
```

5. The function `AI_Reset_All` will stop any other activities in progress and start the configuration process.

```
Joint_Reset_Register
AI reset = 1;
AI configuration start = 1;
```

```
Interrupt_A_Ack_Register = 0x3F80;
```

```
AI_Mode_1_Register
Reserved one = 1;
AI start stop = 1;
```

```
Joint_Reset_Register
AI configuration end = 1;
```

6. `AI_Board_Personalize` sets the DAQ-STC for the AT-MIO E Series board.

```
Joint_Reset_Register
AI configuration start = 1;
```

```
Clock_and_FOUT_Register
Output divide by two = 1;
```

```
Set the AI_Personal_Register = 0xA4A0;
```

```
Set the AI_Output_Control Register = 0x032E;
```

```
Joint_Reset_Register
AI configuration end = 1;
```



7. Call the function `AI_Initialize_Configuration_Memory_Output` to output one pulse and access the first value in the configuration FIFO.

```
AI_Command_1_Register
  Convert pulse = 1;
```

8. The function `AI_Board_Environmentalize` configures the board for any external multiplexers.

```
Joint_Reset_Register
  AI configuration start = 1;
```

```
AI_Mode_2_Register
  External mux present = 0;
```

```
Joint_Reset_Register
  AI configuration end = 1;
```

9. Call `AI_Trigger_Signals` to set the triggering options.

```
Joint_Reset_Register
  AI configuration start = 1;
```

```
AI_Mode_1_Register
  Trigger once = 1;
```

```
AI_Trigger_Select_Register
  Start edge = 1;
  Start sync = 1;
```

```
Joint_Reset_Register
  AI configuration end = 1;
```

10. The function `AI_Scan_Start` selects the scan start event.

```
Joint_Reset_Register
  AI configuration start = 1;
```

```
AI_Start_Stop_Select_Register
  Start edge = 1;
  Start sync = 1;
```

```
Joint_Reset_Register
  AI configuration end = 1;
```

11. Call the function `AI_End_of_Scan` to select the end of scan event.

```
Joint_Reset_Register
  AI configuration start = 1;
```

```
AI_Start_Stop_Select_Register
  Stop select = 19;
  Stop sync = 1;
```

```
Joint_Reset_Register
  AI configuration end = 1;
```

12. Same as Step 4.
13. Now start the acquisition with `AI_Start_The_Acquisition`.

```
AI_Command_1_Register
  Convert Pulse = 1;
```

14. Poll the AI FIFO not empty flag in the `AI_Status_1_Register` until not empty and read the ADC FIFO data in the `ADC_FIFO_Data_Register`.

```
Do
{
  If (AI FIFO not empty) then
    read FIFO data;
} while (FIFO not read)
```

## Example 2

Example 2 illustrates the manner in which to program the STC for scanning.

Acquire 5 scans at a scan interval of 1ms. The scan list contains channels 5, 4, 1, and 0, respectively. The channels are configured as RSE at a gain of 1. Within each scan, the sample interval should be 100 $\mu$ s. Dithering should remain off during the operation. No external multiplexers are used. Use polled input to acquire the data.

1. Perform Analog Input Example 1 Step 1.
2. Perform Analog Input Example 1 Step 2 for each channel in the scan list. Only channel 0 has Last channel set to 1.
3. Perform Analog Input Example 1 Steps 3-9.
4. Call the function `Number_of_Scans` to load the number of scans.

```
Joint_Reset_Register
  AI configuration start = 1;
```

AI\_SC\_Load\_A\_Registers (24 bits)  
Number of posttrigger scans - 1 = 4;

AI\_Command\_1\_Register  
AI SC Load = 1;

Joint\_Reset\_Register  
AI configuration end = 1;

5. The function `AI_Scan_Start` selects the scan start event.

Joint\_Reset\_Register  
AI configuration start = 1;

AI\_Start\_Stop\_Select\_Register  
Start edge = 1;  
Start sync = 1;

AI\_SI\_Load\_A\_Registers (24 bits)  
AI SI special ticks -1 = 1;

AI\_Command\_1\_Register  
AI SI load = 1;

AI\_SI\_Load\_A\_Registers (24 bits)  
AI SI ordinary ticks - 1 = 19999;

Joint\_Reset\_Register  
AI configuration end = 1;

6. Perform Analog Input Example 1 Step 11.

7. `Convert_Signal` selects the convert signal for the acquisition.

Joint\_Reset\_Register  
AI configuration start = 1;

AI\_SI2\_Load\_A\_Register  
AI SI2 special ticks - 1 = 1999;

AI\_SI2\_Load\_B\_Register  
AI SI2 ordinary ticks -1 = 1999;

AI\_Mode\_2\_Register  
AI SI2 reload mode = 1;

AI\_Command\_1\_Register  
AI SI2 load = 1;

AI\_Mode\_2\_Register

```
AI SI2 initial load source = 1;
```

```
Joint_Reset_Register
  AI configuration end = 1;
```

8. Perform Analog Input Example 1 Step 4.
9. Call `AI_Arming` to arm the analog input counter.

```
AI_Command_1_Register
  AI SC arm = 1;
  AI SI arm = 1;
  AI SI2 arm = 1;
  AI DIV arm = 1;
```

10. The function `AI_Start_The_Acquisition` starts the acquisition process.

```
AI_Command_2_Register
  AI START1 Pulse = 1;
```

11. Poll the AI FIFO not empty flag in the `AI_Status_1_Register` until not empty and read the ADC FIFO data in the `ADC_FIFO_Data_Register`.

```
Do
{
  If (AI FIFO not empty) then
    read FIFO data;
} while (20 samples have not been read)
```

### Example 3

Example 3 performs the same acquisition as Example 2, but with interrupts.

Acquire 5 scans at a scan interval of 1ms. The scan list contains channels 5, 4, 1, and 0 respectively. The channels are configured as RSE at a gain of 1. Within each scan, the sample interval should be 100 $\mu$ s. Dithering should remain off during the operation. No external multiplexers are used. Use interrupts to acquire the data.

Only minor modifications to Analog Input Example 2 are needed for this example. Instead of installing the interrupt service routine (ISR) as an interrupt, this example emulates the operation of the interrupt by polling the status register in the DAQ-STC. When the status register indicates an interrupt, the main loop transfers control to the ISR. To use the example ISR as an actual interrupt, you need to learn how to install software interrupts on your system. Generally, the procedure is as follows:

- A. Determine the software interrupt number corresponding to the IRQ line you are using.
- B. Use the `getvect()` and `setvect()` functions to replace the default interrupt handler with your ISR. You should disable interrupts during this step.
- C. Reset the interrupt controller hardware.
- D. Perform your analog input.
- E. After the analog input operation completes, you should re-install the default interrupt handler.
  1. Perform Analog Input Example 1 Step 1.
  2. Perform Analog Input Example 1 Step 2 for each channel in the scan list. Only channel 0 has Last channel set to 1.
  3. Perform Analog Input Example 1 Steps 3-9.
  4. Call the function `Number_of_Scans` to load the number of scans.

Joint\_Reset\_Register

AI configuration start = 1;

AI\_SC\_Load\_A\_Registers (24 bits)

Number of posttrigger scans - 1 = 4;

AI\_Command\_1\_Register

AI SC Load = 1;

Joint\_Reset\_Register

AI configuration end = 1;

5. The function `AI_Scan_Start` selects the scan start event.

Joint\_Reset\_Register

AI configuration start = 1;

AI\_Start\_Stop\_Select\_Register

Start edge = 1;

Start sync = 1;

AI\_SI\_Load\_A\_Registers (24 bits)

AI SI special ticks -1 = 1;

AI\_Command\_1\_Register

AI SI load = 1;

AI\_SI\_Load\_A\_Registers (24 bits)

AI SI ordinary ticks - 1 = 19999;

```
Joint_Reset_Register
  AI configuration end = 1;
```

6. Perform Analog Input Example 1 Step 11.
7. Convert\_Signal selects the convert signal for the acquisition.

```
Joint_Reset_Register
  AI configuration start = 1;
```

```
AI_SI2_Load_A_Register
  AI SI2 special ticks - 1 = 1999;
```

```
AI_SI2_Load_B_Register
  AI SI2 ordinary ticks -1 = 1999;
```

```
AI_Mode_2_Register
  AI SI2 reload mode = 1;
```

```
AI_Command_1_Register
  AI SI2 load = 1;
```

```
AI_Mode_2_Register
  AI SI2 initial load source = 1;
```

```
Joint_Reset_Register
  AI configuration end = 1;
```

8. Perform Analog Input Example 1 Step 4.
9. The function AI\_Interrupt\_Enable enables interrupts for the acquisition.

```
Interrupt_A_Enable_Register
  AI FIFO interrupt enable = 1;
  AI error interrupt enable = 1;
```

```
Interrupt_Control_Register
  MSC IRQ pin = 4;
  MSC IRQ enable = 1;
```

10. Call AI\_Arming to arm the analog input counter.

```
AI_Command_1_Register
  AI SC arm = 1;
  AI SI arm = 1;
  AI SI2 arm = 1;
  AI DIV arm = 1;
```

11. Install the interrupt service routine to handle the interrupt.

```
Interrupt_Service_Routine()
    read FIFO data;
    increment sample counter;
```

12. The function `AI_Start_The_Acquisition` starts the acquisition process.

```
AI_Command_2_Register
    AI START1 Pulse = 1;
```

13. Poll the AI FIFO not empty flag in the `AI_Status_1_Register` until not empty and call the ISR.

```
Do
{
    If (AI FIFO not empty) then
        call Interrupt_Service_Routine;
} while (20 samples have not been read)
```

#### Example 4

Example 4 performs the same acquisition as Example 2, but with DMA.

Acquire 5 scans at a scan interval of 1ms. The scan list contains channels 5, 4, 1, and 0, respectively. The channels are configured as RSE at a gain of 1. Within each scan, the sample interval should be 100 $\mu$ s. Dithering should remain off during the operation. No external multiplexers are used. Use DMA to acquire the data.

In this example the programming of the DMA controller is left to the user.

1. Perform Analog Input Example 1 Step 1.
2. Perform Analog Input Example 1 Step 2 for each channel in the scan list. Only channel 0 has Last channel set to 1.
3. Perform Analog Input Example 1 Steps 3-9.
4. Call the function `Number_of_Scans` to load the number of scans.

```
Joint_Reset_Register
    AI configuration start = 1;
```

```
AI_SC_Load_A_Registers (24 bits)
    Number of posttrigger scans - 1 = 4;
```

```
AI_Command_1_Register
    AI SC Load = 1;
```

```
Joint_Reset_Register
  AI configuration end = 1;
```

5. The function `AI_Scan_Start` selects the scan start event.

```
Joint_Reset_Register
  AI configuration start = 1;
```

```
AI_Start_Stop_Select_Register
  Start edge = 1;
  Start sync = 1;
```

```
AI_SI_Load_A_Registers (24 bits)
  AI SI special ticks -1 = 1;
```

```
AI_Command_1_Register
  AI SI load = 1;
```

```
AI_SI_Load_A_Registers (24 bits)
  AI SI ordinary ticks - 1 = 19999;
```

```
Joint_Reset_Register
  AI configuration end = 1;
```

6. Perform Analog Input Example 1 Step 11.
7. `Convert_Signal` selects the convert signal for the acquisition.

```
Joint_Reset_Register
  AI configuration start = 1;
```

```
AI_SI2_Load_A_Register
  AI SI2 special ticks - 1 = 1999;
```

```
AI_SI2_Load_B_Register
  AI SI2 ordinary ticks -1 = 1999;
```

```
AI_Mode_2_Register
  AI SI2 reload mode = 1;
```

```
AI_Command_1_Register
  AI SI2 load = 1;
```

```
AI_Mode_2_Register
  AI SI2 initial load source = 1;
```

```
Joint_Reset_Register
  AI configuration end = 1;
```

8. Perform Analog Input Example 1 Step 4.



## 9. Program\_DMA\_Controller( ).

Strobes\_Register

DMA A channel enable = 1;

AI\_AO\_Select\_Register

DMA input channel A, B, C = 1;

Channel\_A\_Mode\_Register

DMA A channel select = 5;

DMA A transfer type = 1;

DMA A TC interrupt enable = 1;

Channel\_A\_Mode\_Register

DMA A channel enable = 1;

Interrupt\_B\_Enable\_Register

MSC pass thru interrupt enable = 1;

## 10. The function AI\_Interrupt\_Enable enables interrupts for the acquisition.

Interrupt\_A\_Enable\_Register

AI SC TC interrupt enable = 1;

AI error interrupt enable = 1;

Interrupt\_Control\_Register

MSC IRQ pin = 4;

MSC IRQ enable = 1;

MSC IRQ B pin = 4;

MSC IRQ B enable = 1;

## 11. Arm\_DMA\_Controller( ).

## 12. Call AI\_Arming to arm the analog input counter.

AI\_Command\_1\_Register

AI SC arm = 1;

AI SI arm = 1;

AI SI2 arm = 1;

AI DIV arm = 1;

## 13. The function AI\_Start\_The\_Acquisition starts the acquisition process.

AI\_Command\_2\_Register

AI START1 Pulse = 1;

14. If an interrupt is generated, a terminal count was detected - reprogram the DMA controller.

```
Interrupt_Service_Routine()
    Program_DMA_Controller();
```

### Example 5

Example 5 performs the same acquisition as Example 2, but with the start trigger and scan start pulses applied externally.

Acquire 5 scans. The scan list contains channels 5, 4, 1, and 0, respectively, each at a gain of 1 and in RSE mode. The acquisition should begin on a start trigger applied to PFI0. The sample rate should be set to 100 $\mu$ s, and the start pulses should be connected to PFI1 to trigger each scan. Use polled input to read the AI FIFO data.

1. Perform Analog Input Example 1 Step 1.
2. Perform Analog Input Example 1 Step 2 for each channel in the scan list. Only channel 0 has Last channel set to 1.
3. Perform Analog Input Example 1 Steps 3-8.
4. Call `AI_Trigger_Signals` to set the triggering options.

```
Joint_Reset_Register
    AI configuration start = 1;
```

```
AI_Mode_1_Register
    AI trigger once = 1;
```

```
AI_Trigger_Select_Register = 0x8061;
```

```
Joint_Reset_Register
    AI configuration end = 1;
```

5. Call the function `Number_of_Scans` to load the number of scans.

```
Joint_Reset_Register
    AI configuration start = 1;
```

```
AI_SC_Load_A_Registers (24 bits)
    Number of posttrigger scans - 1 = 4;
```

```
AI_Command_1_Register
    AI SC Load = 1;
```

```
Joint_Reset_Register
    AI configuration end = 1;
```

6. The function `AI_Scan_Start` selects the scan start event.

```
Joint_Reset_Register  
    AI configuration start = 1;
```

```
AI_START_STOP_Select_Register = 0x0062;
```

```
Joint_Reset_Register  
    AI configuration end = 1;
```

7. Perform Analog Input Example 1 Step 11.
8. `Convert_Signal` selects the convert signal for the acquisition.

```
Joint_Reset_Register  
    AI configuration start = 1;
```

```
AI_Mode_3_Register  
    AI SI2 source = 1;
```

```
AI_SI2_Load_A_Register  
    AI SI2 special ticks - 1 = 1999;
```

```
AI_SI2_Load_B_Register  
    AI SI2 ordinary ticks -1 = 1999;
```

```
AI_Mode_2_Register  
    AI SI2 reload mode = 1;
```

```
AI_Command_1_Register  
    AI SI2 load = 1;
```

```
AI_Mode_2_Register  
    AI SI2 initial load source = 1;
```

```
Joint_Reset_Register  
    AI configuration end = 1;
```

9. Perform Analog Input Example 1 Step 4.
10. Call `AI_Arming` to arm the analog input counter.

```
AI_Command_1_Register  
    AI SC arm = 1;  
    AI SI arm = 0;  
    AI SI2 arm = 1;  
    AI DIV arm = 1;
```

11. Poll the AI FIFO not empty flag in the AI\_Status\_1\_Register until not empty and read the ADC FIFO data in the ADC\_FIFO\_Data\_Register.

```

Do
{
    If (AI FIFO not empty) then
        read FIFO data;
} while (20 samples have not been read)

```

### Example 6

Example 6 performs 20 scans as in Example 2. Additionally, an external start and stop trigger control the acquisition.

Acquire 20 scans. The scan list contains channels 5, 4, 1, and 0, respectively, each at a gain of 1 and in RSE mode. The acquisition should begin on a start trigger applied to PF10. The sample rate should be set to 100 $\mu$ s. The stop trigger should be connected to PF11. Acquire 10 scans after the stop trigger, leaving 10 scans before the stop trigger. Use polled input to read the AI FIFO data.

1. Perform Analog Input Example 1 Step 1.
2. Perform Analog Input Example 1 Step 2 for each channel in the scan list. Only channel 0 has Last channel set to 1.
3. Perform Analog Input Example 1 Steps 3-8.
4. Call AI\_Trigger\_Signals to set the triggering options.

```

Joint_Reset_Register
    AI configuration start = 1;

```

```

AI_Mode_1_Register
    AI trigger once = 1;

```

```

AI_Trigger_Select_Register = 0xB161;

```

```

Joint_Reset_Register
    AI configuration end = 1;

```

5. Call the function Number\_of\_Scans to load the number of scans.

```

Joint_Reset_Register
    AI configuration start = 1;

```

```

AI_Mode_2_Register
    AI pretrigger = 1;

```

```

AI_SC_Load_B_Registers (24 bits)

```

Number of pretrigger scans - 1 = 9;

AI\_Mode\_2\_Register

AI SC initial load source = 1;

AI SC reload mode = 1;

AI\_SC\_Load\_A\_Registers (24 bits)

Number of posttrigger scans - 1 = 9;

AI\_Command\_1\_Register

AI SC load = 1;

Joint\_Reset\_Register

AI configuration end = 1;

6. The function `AI_Scan_Start` selects the scan start event.

Joint\_Reset\_Register

AI configuration start = 1;

AI\_START\_STOP\_Select\_Register = 0x0060;

AI\_SI\_Load\_A\_Registers (24 bits)

AI SI special ticks - 1 = 1;

AI\_Command\_1\_Register

AI SI load = 1;

AI\_SI\_Load\_A\_Registers (24 bits)

AI SI ordinary ticks - 1 = 19999;

Joint\_Reset\_Register

AI configuration end = 1;

7. Perform Analog Input Example 1 Step 11.
8. `Convert_Signal` selects the convert signal for the acquisition.

Joint\_Reset\_Register

AI configuration start = 1;

AI\_SI2\_Load\_A\_Register

AI SI2 special ticks - 1 = 1999;

AI\_SI2\_Load\_B\_Register

AI SI2 ordinary ticks -1 = 1999;

```

AI_Mode_2_Register
  AI SI2 reload mode = 1;

AI_Command_1_Register
  AI SI2 load = 1;

AI_Mode_2_Register
  AI SI2 initial load source = 1;

Joint_Reset_Register
  AI configuration end = 1;

```

9. Perform Analog Input Example 1 Step 4.
10. Call `AI_Arming` to arm the analog input counter.

```

AI_Command_1_Register
  AI SC arm = 1;
  AI SI arm = 1;
  AI SI2 arm = 1;
  AI DIV arm = 1;

```

11. Poll the AI FIFO not empty flag in the `AI_Status_1_Register` until not empty and read the ADC FIFO data in the `ADC_FIFO_Data_Register`.

```

Do
{
  If (AI FIFO not empty) then
    read FIFO data;
    if (count = 80)
      reset count to 0;
} while (SC_TC flag in the AI_Status_1_Register is not set or the AIFIFO not empty)

```

### Example 7

Example 7 performs the same scanning as Example 2, but as a single wire acquisition.

Acquire 5 scans. The scan list contains channels 5, 4, 1, and 0, respectively, each at a gain of 1 and in RSE mode. The `START` and `CONVERT` signals should be applied to `PFIO`. Use polled input to read the AI FIFO data.

1. Perform Analog Input Example 1 Step 1.
2. Perform Analog Input Example 1 Step 2 for each channel in the scan list. Only channel 0 has Last channel set to 1.
3. Perform Analog Input Example 1 Steps 3-9.
4. Call the function `Number_of_Scans` to load the number of scans.

```
Joint_Reset_Register
  AI configuration start = 1;
```

```
AI_SC_Load_A_Registers (24 bits)
  Number of posttrigger scans - 1 = 4;
```

```
AI_Command_1_Register
  AI SC Load = 1;
```

```
Joint_Reset_Register
  AI configuration end = 1;
```

5. The function `AI_Scan_Start` selects the scan start event.

```
Joint_Reset_Register
  AI configuration start = 1;
```

```
AI_START_STOP_Select_Register = 0x0021;
```

```
Joint_Reset_Register
  AI configuration end = 1;
```

6. Perform Analog Input Example 1 Step 11.
7. `Convert_Signal` selects the convert signal for the acquisition.

```
Joint_Reset_Register
  AI configuration start = 1;
```

```
AI_Mode_2_Register
  AI SC gate enable = 1;
  AI START STOP gate enable = 1;
```

```
AI_Mode_1_Register
  AI CONVERT source select = 1;
  AI CONVERT source polarity = 1;
```

```
Joint_Reset_Register
  AI configuration end = 1;
```

8. Perform Analog Input Example 1 Step 4.
9. Call `AI_Arming` to arm the analog input counter.

```
AI_Command_1_Register
  AI SC arm = 1;
  AI SI arm = 0;
  AI SI2 arm = 0;
  AI DIV arm = 1;
```

11. The function `AI_Start_The_Acquisition` starts the acquisition process.

```
AI_Command_2_Register
  AI START1 pulse = 1;
```

12. Poll the AI FIFO not empty flag in the `AI_Status_1_Register` until not empty and read the ADC FIFO data in the `ADC_FIFO_Data_Register`.

```
Do
{
  If (AI FIFO not empty) then
    read FIFO data;
} while (20 samples have not been read)
```

### Example 8

This example samples one channel on an AMUX-64T.

Use the factory default settings on the AMUX-64T, internal power, single-board configuration, no temperature setting, and the shield unconnected. Sample channel 3 on the AMUX-64T to acquire 100 samples at a sample interval of 10 $\mu$ s. Connect a voltage source to channel 3, and compare the unscaled results to the applied voltage. Read the samples using polled input.

1. Perform Analog Input Example 1 Step 1.
2. Perform Analog Input Example 1 Step 2 for channel 3.
3. Perform Analog Input Example 1 Steps 3-6.
4. Call the function `AI_Initialize_Configuration_Memory_Output` to output one pulse and access the first value in the configuration FIFO. This function also configures the DIO circuitry for the AMUX-64T.

```
AI_Command_1_Register
  AI convert pulse = 1;
```

```
DIO_Control_Register = 0x0003;
DIO_Output_Register = 0x0003;
DIO_Control_Register = 0x0803;
DIO_Control_Register = 0x0003;
```

5. Perform Analog Input Example 1 Steps 8-9.



6. Call the function `Number_of_Scans` to load the number of scans.

```
Joint_Reset_Register
```

```
    AI configuration start = 1;
```

```
AI_SC_Load_A_Registers (24 bits)
```

```
    Number of posttrigger scans - 1 = 99;
```

```
AI_Command_1_Register
```

```
    AI SC Load = 1;
```

```
Joint_Reset_Register
```

```
    AI configuration end = 1;
```

7. The function `AI_Scan_Start` selects the scan start event.

```
Joint_Reset_Register
```

```
    AI configuration start = 1;
```

```
AI_START_STOP_Select_Register = 0x0060;
```

```
AI_SI_Load_A_Registers (24 bits)
```

```
    AI SI special ticks - 1 = 1
```

```
AI_Command_1_Register
```

```
    AI SI load = 1;
```

```
AI_SI_Load_A_Registers (24 bits)
```

```
    AI AI ordinary ticks - 1 = 199;
```

```
Joint_Reset_Register
```

```
    AI configuration end = 1;
```

8. Perform Analog Input Example 1 Step 11.
9. `Convert_Signal` selects the convert signal for the acquisition.

```
Joint_Reset_Register
```

```
    AI configuration start = 1;
```

```
AI_SI2_Load_A_Register
```

```
    AI SI2 special ticks - 1 = 1;
```

```
AI_SI2_Load_B_Register
```

```
    AI SI2 ordinary ticks -1 = 1;
```

```
AI_Mode_2_Register
```

```
    AI SI2 reload mode = 1;
```

```
AI_Command_1_Register
```

```
AI SI2 load = 1;
```

```
AI_Mode_2_Register
  AI SI2 initial load source = 1;
```

```
Joint_Reset_Register
  AI configuration end = 1;
```

10. Perform Analog Input Example 1 Step 4.

11. Call `AI_Arming` to arm the analog input counter.

```
AI_Command_1_Register
  AI SC arm = 1;
  AI SI arm = 1;
  AI SI2 arm = 1;
  AI DIV arm = 1;
```

12. Now start the acquisition with `AI_Start_The_Acquisition`.

```
AI_Command_2_Register
  AI START1 pulse = 1;
```

13. Poll the AI FIFO not empty flag in the `AI_Status_1_Register` until not empty and read the ADC FIFO data in the `ADC_FIFO_Data_Register`.

```
Do
{
  If (AI FIFO not empty) then
    read FIFO data;
} while (100 samples have not been read)
```

### Example 9

This example scans 8 channels on an AMUX-64T.

Use the factory default settings on the AMUX-64T, internal power, single-board configuration, no temperature setting, and the shield unconnected. Scan channels 0 through 7 on the AMUX-64T. Acquire 10 scans at a scan interval of 200 $\mu$ s and a sample interval of 20 $\mu$ s. Connect a voltage source to channels 0-3 and ground channels 4-7. Compare the unscaled results to the applied voltage. Read the samples using polled input.

1. Perform Analog Input Example 1 Step 1.
2. Perform Analog Input Example 1 Step 2 for channels 0 and 1. Only channel 1 has Last channel set to 1.
3. Perform Analog Input Example 1 Steps 3-6.

4. Call the function `AI_Initialize_Configuration_Memory_Output` to output one pulse and access the first value in the configuration FIFO. This function also configures the DIO circuitry for the AMUX-64T.

```
AI_Command_1_Register
  AI convert one pulse = 1;
```

```
AI_Mode_2_Register
  AI external mux present = 1;
```

```
DIO_Control_Register = 0x0800;
DIO_Control_Register = 0x0000;
DIO_Output_Register = 0x0000;
DIO_Control_Register = 0x0003;
DIO_Control_Register = 0x0803;
DIO_Control_Register = 0x0003;
```

5. The function `AI_Board_Environmentalize` configures the board for any external multiplexers.

```
Joint_Reset_Register
  AI configuration start = 1;
```

```
AI_Mode_2_Register
  External mux present = 1;
```

```
AI_Output_Control_Register
  AI external mux clock select = 3;
```

```
AI_DIV_Load_A_Register
  AI number of channels ratio - 1 = 3;
```

```
AI_Command_1_Register
  AI DIV load = 1;
```

```
Joint_Reset_Register
  AI configuration end = 1;
```

6. Perform Analog Input Example 1 Step 9.
7. Call the function `Number_of_Scans` to load the number of scans.

```
Joint_Reset_Register
  AI configuration start = 1;
```

```
AI_SC_Load_A_Registers (24 bits)
  Number of posttrigger scans - 1 = 99;
```

```
AI_Command_1_Register
  AI SC Load = 1;
```

```
Joint_Reset_Register
  AI configuration end = 1;
```

8. The function `AI_Scan_Start` selects the scan start event.

```
Joint_Reset_Register
  AI configuration start = 1;
```

```
AI_START_STOP_Select_Register = 0x0060;
```

```
AI_SI_Load_A_Registers (24 bits)
  AI SI special ticks - 1 = 1
```

```
AI_Command_1_Register
  AI SI load = 1;
```

```
AI_SI_Load_A_Registers (24 bits)
  AI AI ordinary ticks - 1 = 3999;
```

```
Joint_Reset_Register
  AI configuration end = 1;
```

9. Perform Analog Input Example 1 Step 11.

10. `Convert_Signal` selects the convert signal for the acquisition.

```
Joint_Reset_Register
  AI configuration start = 1;
```

```
AI_SI2_Load_A_Register
  AI SI2 special ticks - 1 = 399;
```

```
AI_SI2_Load_B_Register
  AI SI2 ordinary ticks -1 = 399;
```

```
AI_Mode_2_Register
  AI SI2 reload mode = 1;
```

```
AI_Command_1_Register
  AI SI2 load = 1;
```

```
AI_Mode_2_Register
  AI SI2 initial load source = 1;
```

```
Joint_Reset_Register
  AI configuration end = 1;
```

11. Perform Analog Input Example 1 Step 4.
12. Call `AI_Arming` to arm the analog input counter.

```
AI_Command_1_Register
  AI SC arm = 1;
  AI SI arm = 1;
  AI SI2 arm = 1;
  AI DIV arm = 1;
```

13. Now start the acquisition with `AI_Start_The_Acquisition`.

```
AI_Command_2_Register
  AI START1 pulse = 1;
```

14. Poll the AI FIFO not empty flag in the `AI_Status_1_Register` until not empty and read the ADC FIFO data in the `ADC_FIFO_Data_Register`.

```
Do
{
  If (AI FIFO not empty) then
    read FIFO data;
} while (80 samples have not been read)
```

## Analog Output

Chapter 3 of the *DAQ-STC Technical Reference Manual* contains all the information on the analog output timing/control module of the DAQ-STC, with specific programming steps in the *Programming Information* section. The instructions are arranged in a sequence of functions, which are used in the examples below to program the DAQ-STC.

Analog output DAQ-STC programming consists of the following functions:

```
AO_Reset_All;
MSC_Clock_Configure;
AO_Board_Personalize;
AO_Triggering;
AO_Counting;
AO_Updating;
AO_Channels;
AO_LDAC_Source_And_Update_Mode;
AO_Errors_To_Stop_On;
AO_FIFO;
AO_Arming;
AO_Start_The_Acquisition;
```

As the analog output examples increase in complexity, more of these functions will be necessary. The functions will be presented in the applicable examples; subsequent examples address only

the specific differences from Example 1. This manual provides the structure and pseudo-code for each example. The *AT-MIO E Series Register Level Programmer Manual Companion Disk* contains the complete programs. The following pseudo-code examples and the programs on the Companion Disk follow the flowchart structure presented in the *DAQ-STC Technical Reference Manual*.

### Example 1

This is an example of CPU write to the DAC.

Write 5 volts to DAC 1 in bipolar mode. Use internal reference and disable reglitching. The data FIFO is not used. Use the immediate update mode to update the output on DAC1.

1. Set up the Plug and Play resources. Use the function `Setup_PNP_Board` provided on the Companion Disk.
2. Configure the board by setting the following board level bits.

AO\_Configuration\_Register

DACSel<3:0> = 1;

BipDac = 1;

ExtRef = 0;

ReGlitch = 0;

GroundRef = 0;

3. Call `AO_Reset_All` to reset the DAQ-STC.

Joint\_Reset\_Register

AO configuration start = 1;

AO\_Command\_1\_Register

AO disarm = 1;

Interrupt\_B\_Enable\_Register = 0x0000;

AO\_Personal\_Register

AO BC source select = 1;

Interrupt\_B\_Ack\_Register = 0x3F98;

Joint\_Reset\_Register

AO configuration start = 0;

AO configuration end = 1;

4. Call `AO_Board_Personalize` to configure the DAQ-STC.

Joint\_Reset\_Register

AO\_Configuration\_Start = 1;

```
AO_Personal_Register = 0x1430;
```

```
Clock_and_FOUT_Register = 0x1B20;
```

```
AO_Output_Control_Register = 0x0000;
```

```
AO_START_Select_Register = 0x 0000;
```

```
Joint_Reset_Register
```

```
  AO configuration start = 0;
```

```
  AO configuration end = 1;
```

5. Call `AO_LDAC_Source_And_Update_Mode` to set the update mode to immediate update mode.

```
Joint_Reset_Register
```

```
  AO configuration start = 1;
```

```
AO_Command_1_Register
```

```
  AO LDAC0 source select = 0;
```

```
  AO DAC0 update mode = 0;
```

```
  AO LDAC1 source select = 0;
```

```
  AO DAC1 update mode = 0;
```

```
Joint_Reset_Register
```

```
  AO configuration start = 0;
```

```
  AO configuration end = 0;
```

6. Write to the DAC1 Data Register. In bipolar mode, +10V corresponds to 2,047 (0x07FF) and -10V corresponds to -2,048 (0xF800). Hence, +5V corresponds to 1,024 (0x0400). Writing this to the DAC1 data register results in +5V appearing at the OUT1 line.

## Example 2

This example generates a waveform using polled writes to the data FIFO.

Initialize the buffer with 3000 points. Use polled writes to write each point to the data FIFO. Updates occur every 2 milliseconds. Output the buffer 5 times. Confirm operation with an oscilloscope.

1. Perform Analog Output Example 1 Step 1.
2. Load an array with the voltages of the waveform to be output. In bipolar mode, +10V corresponds to 2,047 (0x07FF) and -10V corresponds to -2,048 (0xF800). To convert the voltage you want to output to the binary value, use the following formula:

$$\text{binary value} = (\text{voltage}/10) * 2048$$

- Configure the board by setting the following board level bits.

AO\_Configuration\_Register:

```
DACSel<3:0> = 1;
BipDac = 1;
ExtRef = 0;
ReGlitch = 0;
GroundRef = 0;
```

- Reset the data FIFO.
- Pre-load the data FIFO with the voltages from the array.

```
while (FIFO is not full and there is more data to write)
{
    AO_DAC_FIFO_Data = data;
}
```

- Call `MSC_Clock_Configure` to program the timebase options.

```
Clock_and_FOUT_Register = 0x1B00;
```

- Call `AO_Reset_All` to program the timebase options.

```
Joint_Reset_Register
    AO configuration start = 1;
```

```
AO_Command_1_Register
    AO disarm = 1;
```

```
Interrupt_B_Enable_Register = 0x0000;
```

```
AO_Personal_Register
    AO BC source select = 1;
```

```
Interrupt_B_Ack_Register = 0x3F98;
```

```
Joint_Reset_Register
    AO configuration start = 0;
    AO configuration end = 1;
```

- Call `AO_Board_Personalize` to configure the DAQ-STC for the MIO board. Use the following bitfield settings:

```
Joint_Reset_Register
    AO_Configuration_Start = 1;
```

```
AO_Personal_Register = 0x1430;
```

```
Clock_and_FOUT_Register = 0x1B20;
```



```
AO_Output_Control_Register = 0x0000;
```

```
AO_START_Select_Register = 0x 0000;
```

```
Joint_Reset_Register
```

```
    AO configuration start = 0;
```

```
    AO configuration end = 1;
```

9. Call `AO_Triggering` to program the trigger signal. Configure the DAQ-STC to trigger once and use a software START1 trigger.

```
Joint_Reset_Register
```

```
    AO configuration start = 1;
```

```
AO_Mode_1_Register
```

```
    AO trigger once = 1;
```

```
AO_Trigger_Select_Register
```

```
    AO START1 select = 0;
```

```
    AO START1 polarity = 0;
```

```
    AO START1 edge = 1;
```

```
    AO START1 sync = 1;
```

```
AO_Mode_3_Register
```

```
    AO trigger length = 0;
```

```
Joint_Reset_Register
```

```
    AO configuration start = 0;
```

```
    AO configuration end = 1;
```

10. Call `AO_Counting` to program the buffer size and the number of buffers. Configure the DAQ-STC for non-continuous operation (AO will stop on BC\_TC). Load the BC counter with 4 (output the buffer 5 times). Load the UC counter with 3000 (the first buffer contains 3000 points). Write 2999 to UC Load Register A (each subsequent buffer contains 3000 points).

```
Joint_Reset_Register
```

```
    AO configuration start = 1;
```

```
AO_Mode_1_Register
```

```
    AO continuous = 0;
```

```
AO_Mode_2_Register
```

```
    AO BC initial load source = 0;
```

```
AO_BC_Load_A_Registers (24 bits)
```

```
    Number of buffers -1 = 4;
```

AO\_Command\_1\_Register  
AO BC load = 1;

AO\_Mode\_2\_Register  
AO UC initial load source = 0;

AO\_UC\_Load\_A\_Registers (24 bits)  
Points per buffer = 3000;

AO\_Command\_1\_Register  
AO UC load = 1;

AO\_UC\_Load\_A\_Registers (24 bits)  
Points per buffer - 1 = 2999;

Joint\_Reset\_Register  
AO configuration start = 0;  
AO configuration end = 1;

11. Call `AO_Updating` to program the update interval. Use the internal UPDATE mode. Set the UI source to `AO_IN_TIMEBASE1`. Load the UI counter with 1 (minimum delay from the `START1` to the first UPDATE). Write `0x9C40` to UI Load Register A (2 millisecond update interval).

Joint\_Reset\_Register  
AO configuration start = 1;

AO\_Command\_2\_Register  
AO BC gate enable = 0;

AO\_Mode\_1\_Register  
AO UPDATE source select = 0;  
AO UPDATE source polarity = 0;

AO\_Mode\_1\_Register  
AO UI source select = 0;  
AO UI source polarity = 0;

AO\_Mode\_2\_Register  
AO UI initial load source = 0;  
AO UI reload mode = 0;

AO\_UI\_Load\_A\_Registers (24 bits)  
AO UI special ticks - 1 = 1;

AO\_Command\_1\_Register  
AO UI load = 1;

AO\_UI\_Load\_A\_Registers (24 bits)  
AO UI ordinary ticks - 1 = `9C40`;

```
Joint_Reset_Register
  AO configuration start = 0;
  AO configuration end = 1;
```

12. Call `AO_Channels` to select single channel output.

```
Joint_Reset_Register
  AO configuration start = 1;
```

```
AO_Mode_1_Register
  AO multiple channels = 0;
```

```
AO_Output_Control_Register
  AO number of channels = 1;
```

```
Joint_Reset_Register
  AO configuration start = 0;
  AO configuration end = 1;
```

13. Call `AO_LDAC_Source_And_Update_Mode` to configure LDAC1 to output UPDATE in the timed update mode.

```
Joint_Reset_Register
  AO configuration start = 0;
```

```
AO_Command_1_Register
  AO LDAC0 source select = 0;
  AO DAC0 update mode = 1;
  AO LDAC1 source select = 0;
  AO DAC1 update mode = 1;
```

```
Joint_Reset_Register
  AO configuration start = 0;
  AO configuration end = 1;
```

14. Call `AO_Errors_To_Stop_On` to configure the analog output to stop on overrun error.

```
Joint_Reset_Register
  AO configuration start = 1;
```

```
AO_Mode_3_Register = 0x0020;
```

```
Joint_Reset_Register
  AO configuration start = 0;
  AO configuration end = 1;
```

15. Call `AO_FIFO` to disable the FIFO retransmit.

```

Joint_Reset_Register
    AO configuration start = 0;

AO_Mode_2_Register
    AO FIFO retransmit enable = 0;

Joint_Reset_Register
    AO configuration start = 0;
    AO configuration end = 1;

```

16. Call `Kick_Start_FIFO` to initialize the virtual FIFO boards.

```

if (VirtualFIFO)
    AO_DAC_FIFO_Data = 0;

```

17. Call `AO_Arming` to arm the counters and preload the DAC with the first analog output value.

```

AO_Mode_3_Register
    AO not an UPDATE = 1;

AO_Mode_3_Register
    AO not an UPDATE = 0;

If (!VirtualFIFO)
    Wait until DACs have been preloaded.

AO_Command_1_Register = 0x0554;

```

18. Call `AO_Start_The_Acquisition` to pulse the software START1 trigger.

```

AO_Command_2_Register
    AO START1 pulse = 1;

```

19. Poll the `AO_FIFO_Full_St` bit and write data from the array into the data FIFO whenever the data FIFO is not full.

```

while (there is more data to write)
{
    while (FIFO is not full)
    {
        AO_DAC_FIFO_Data = data;
    }
}

```

**Example 3**

This example generates a waveform using local buffer mode. This example does not support those boards with virtual analog output FIFOs: AT-MIO-16E-10, AT-MIO-16DE-10, and AT-MIO-16XE-50.

Initialize the data FIFO with a 100 point buffer. Output the buffer 50 times. The update interval is 100 microseconds. Confirm operation with an oscilloscope.

1. Perform Analog Output Example 2 Steps 1-9.
2. Call `AO_Counting` to program the buffer size and the number of buffers. Configure the DAQ-STC for non-continuous operation (AO will stop on BC\_TC). Load the BC counter with 49 (output the buffer 50 times). Load the UC counter with 100 (the first buffer contains 100 points). Write 99 to UC Load Register A (each subsequent buffer contains 100 points).

Joint\_Reset\_Register

AO configuration start = 1;

AO\_Mode\_1\_Register

AO continuous = 0;

AO\_Mode\_2\_Register

AO BC initial load source = 0;

AO\_BC\_Load\_A\_Registers (24 bits)

Number of buffers - 1 = 49;

AO\_Command\_1\_Register

AO BC load = 1;

AO\_Mode\_2\_Register

AO UC initial load source = 0;

AO\_UC\_Load\_A\_Registers (24 bits)

Points per buffer = 100;

AO\_Command\_1\_Register

AO UC load = 1;

AO\_UC\_Load\_A\_Registers (24 bits)

Points per buffer - 1 = 99;

Joint\_Reset\_Register

AO configuration start = 0;

AO configuration end = 1;

3. Call `AO_Updating` to program the update interval. Use the internal UPDATE mode. Set the UI source to `AO_IN_TIMEBASE1`. Load the UI counter with 1 (minimum delay from

the START1 to the first UPDATE). Write 1999 to UI Load Register A (100 microsecond update interval).

Joint\_Reset\_Register

AO configuration start = 1;

AO\_Command\_2\_Register

AO BC gate enable = 0;

AO\_Mode\_1\_Register

AO UPDATE source select = 0;

AO UPDATE source polarity = 0;

AO\_Mode\_1\_Register

AO UI source select = 0;

AO UI source polarity = 0;

AO\_Mode\_2\_Register

AO UI initial load source = 0;

AO UI reload mode = 0;

AO\_UI\_Load\_A\_Registers (24 bits)

AO UI special ticks -1 = 1;

AO\_Command\_1\_Register

AO UI load = 1;

AO\_UI\_Load\_A\_Registers (24 bits)

AO UI ordinary ticks - 1 = 1999;

Joint\_Reset\_Register

AO configuration start = 0;

AO configuration end = 1;

4. Perform Analog Output Example 2 Steps 12-14.

5. Call AO\_FIFO to enable the FIFO retransmit.

Joint\_Reset\_Register

AO configuration start = 0;

AO\_Mode\_2\_Register

AO FIFO retransmit enable = 0;

Joint\_Reset\_Register

AO configuration start = 0;

AO configuration end = 1;

6. Call `AO_Arming` to arm the counters and preload the DAC with the first analog output value.

```
AO_Mode_3_Register  
  AO not an UPDATE = 1;
```

```
AO_Mode_3_Register  
  AO not an UPDATE = 0;
```

*Wait until DACs have been updated.*

```
AO_Command_1_Register = 0x0554;
```

7. Call `AO_Start_The_Acquisition` to pulse the software START1 trigger.

```
AO_Command_2_Register  
  AO START1 pulse = 1;
```

#### Example 4

This example generates a waveform using local buffer mode with external UPDATE and external trigger. This example does not support those boards with a virtual FIFO due to the use of the local buffer mode, but the programming of the external UPDATE and START1 applies to all boards.

Initialize the data FIFO with a 100 point buffer. Output the buffer 50 times. The update interval is determined externally. Confirm operation with an oscilloscope.

1. Perform Analog Output Example 2 Steps 1-8.
2. Call `AO_Triggering` to program the trigger signal. Configure the DAQ-STC to trigger once. Set the START1 select to PFI6.

```
Joint_Reset_Register  
  AO configuration start = 1;
```

```
AO_Mode_1_Register  
  AO trigger once = 1;
```

```
AO_Trigger_Select_Register  
  AO START1 select = 7;  
  AO START1 polarity = 0;  
  AO START1 edge = 1;  
  AO START1 sync = 1;
```

```
AO_Mode_3_Register  
  AO trigger length = 0;
```

**Joint\_Reset\_Register**

AO configuration start = 0;  
 AO configuration end = 1;

3. Call `AO_Counting` to program the buffer size and the number of buffers. Configure the DAQ-STC for non-continuous operation (AO will stop on `BC_TC`). Load the BC counter with 49 (output the buffer 50 times). Load the UC counter with 100 (the first buffer contains 100 points). Write 99 to UC Load Register A (each subsequent buffer contains 100 points).

**Joint\_Reset\_Register**

AO configuration start = 1;

**AO\_Mode\_1\_Register**

AO continuous = 0;

**AO\_Mode\_2\_Register**

AO BC initial load source = 0;

**AO\_BC\_Load\_A\_Registers (24 bits)**

Number of buffers - 1 = 49;

**AO\_Command\_1\_Register**

AO BC load = 0;

**AO\_Mode\_2\_Register**

AO UC initial load source = 0;

**AO\_UC\_Load\_A\_Registers (24 bits)**

Points per buffer = 100;

**AO\_Command\_1\_Register**

AO UC load = 1;

**AO\_UC\_Load\_A\_Registers (24 bits)**

Points per buffer - 1 = 99;

**Joint\_Reset\_Register**

AO configuration start = 0;  
 AO configuration end = 1;

4. Call `AO_Updating` to program the update interval.

Use the external `UPDATE` mode.  
 Set the `UPDATE` source to `PF15`.

5. Perform Analog Output Example 3 Steps 4-7.



**Example 5**

This example generates a waveform using interrupts to write data to the data FIFO.

Initialize the buffer with 3000 points. Use polled writes to write each point to the data FIFO. Updates occur every 2 milliseconds. Output the buffer 5 times. Confirm operation with an oscilloscope.

Instead of installing the interrupt service routine (ISR) as an interrupt, this example emulates the operation of the interrupt by polling the status register in the DAQ-STC. When the status register indicates an interrupt, the main loop transfers control to the ISR. To use the example ISR as an actual interrupt, you need to learn how to install software interrupts on your system. Generally, the procedure is as follows:

- A. Determine the software interrupt number corresponding to the IRQ line you are using.
- B. Use the `getvect()` and `setvect()` functions to replace the default interrupt handler with your ISR. You should disable interrupts during this step.
- C. Reset the interrupt controller hardware.
- D. Perform your analog output.
- E. After the analog output operation completes, you should re-install the default interrupt handler.
  1. Perform Analog Output Example 2 Steps 1-14.
  2. Call `AO_FIFO` to disable the FIFO retransmit and set the FIFO mode.

```
Joint_Reset_Register
  AO configuration start = 1;
```

```
AO_Mode_2_Register
  AO FIFO Mode = 1;
  AO FIFO retransmit enable = 0;
```

```
Joint_Reset_Register
  AO configuration start = 0;
  AO configuration end = 1;
```

3. Call `Kick_Start_FIFO` to initialize the virtual FIFO boards.

```
if (VirtualFIFO)
  AO_DAC_FIFO_Data = 0;
```

4. Call `AO_Arming` to arm the counters and preload the DAC with the first analog output value.

```
AO_Mode_3_Register
```

```
AO not an UPDATE = 1;
```

```
AO_Mode_3_Register
AO not an UPDATE = 0;
```

```
if (!VirtualFIFO)
    Wait until DACs have been preloaded.
```

```
AO_Command_1_Register = 0x554;
```

5. Program the DAQ-STC to generate interrupts on the FIFO condition.

```
Interrupt_B_Enable_Register
AO FIFO interrupt enable = 1;
```

```
Interrupt_Control_Register
Interrupt B output select = IRQ number;
Interrupt B enable = 1;
```

6. Install the interrupt service routine to handle the interrupt.

```
service_interrupt()
Do
{
    If (AO FIFO not full) {
        AO_DAC_FIFO_Data = data;
        increment data index counter;
    }
} while (AO FIFO not full && total point have not been written)
```

7. Call `AO_Start_The_Acquisition` to pulse the software START1 trigger.

```
AO_Command_2_Register
AO START1 pulse = 1;
```

8. Poll the AO FIFO half full flag in the `AO_Status_1_Register` until half full and call the ISR.

```
If (AO FIFO half full) then
    call service_interrupt;
```

## General-Purpose Counter/Timer

There are two 24-bit up/down counters available for performing event counting, pulse-width measurement, pulse and pulse train generation, etc. Associated with each counter are load and save registers. You can use interrupts with these counters to do buffered measurements. Each time the counter generates an interrupt, a new set of values can be loaded into the counters.

You can select the input signals to the counters from any of the 17 (10 PFI + 7 RTSI) external timing I/O pins. Signals can also be output on certain dedicated lines. For example, the pin name PFI8/CTRSRC0 means that the clock source input for Counter 0 can come from any of the PFI/RTSI lines but can be output only on the PFI8 line. All the information necessary for programming the general-purpose counters/timers is provided in Chapter 4 of the *DAQ-STC Technical Reference Manual*.

Because of the large number of modes in which the counters can be programmed, no specific examples are discussed here. You should follow the step-by-step building blocks discussed in the *Programming Information* section of Chapter 4 in the *DAQ-STC Technical Reference Manual* to write your own program.

## RTSI Trigger Lines Programming Considerations

For detailed descriptions of RTSI and RTSI programming, see Chapter 6, *RTSI Trigger*, of the *DAQ-STC Technical Reference Manual*. Also, see the same section of the DAQ-STC manual for a description of the signal lines referred to in this chapter.

There are seven 12-1 muxes that drive the seven RTSI lines. Any of the RTSI lines can be driven with any of eight internally generated timing signals or with any of the four RTSI board signals. Similarly, there are four 8-1 muxes that can drive the four RTSI board signals with any of the seven RTSI trigger signals and the AISTART or AISTOP signal.

See the DAQ-STC manual for information on programming the RTSI interface. The function `MSC_RTSI_Pin_Configure` is very straightforward and easy to use.

Notice that, of the four RTSI board signals, only `RTSI_BRD0` is connected to `GENTRIG0`. The remaining three are unused. The `GENTRIG0` signal is a logical AND of the `CONVERT` pulse and the `DOTRIG0` bit in the configuration memory. When the `DOTRIG0` is set, the `GENTRIG` line is driven by the `CONVERT` signal. This can be driven out to the RTSI lines and used by some other board for synchronization purposes.

## Analog Triggering

The AT-MIO-16E-1, AT-MIO-16E-2, and AT-MIO-64E-3 contain true analog triggering hardware, which provides fast slope and level detection, as well as window detection. The mode selection circuitry is in the DAQ-STC, while the analog comparison is performed by onboard circuitry.

Refer to the *Analog Trigger* section of Chapter 10, *Miscellaneous Functions*, in the *DAQ-STC Technical Reference Manual* for information about the analog trigger functionality of the DAQ-STC. The various modes, Low Window, High Window, Middle Window, high and low hysteresis are discussed. The signals connected to the High Value and Low Value are analog signals generated by `CALDAC11` and `CALDAC12`. These are the remaining two CALDACs on the MB88341 chip. The first 10 CALDACs were used up by the calibration circuitry. For more information about the MB88341, see Chapter 5, *Calibration*, later in this manual.

One of two sources may be used as the trigger source, the PFI0/TRIG1 input on the I/O connector or the analog inputs passing through the PGIA2. The PGIA2 allows you to apply gain to an external signal for more flexible triggering conditions. The PFI0/TRIG1 pin has an input voltage range of  $\pm 10$  volts, which corresponds to the PGIA2 path at a gain of 1. The Analog\_Trigger\_Drive bit in the Analog\_Trigger\_Etc\_Register controls which input is used. The PGIA2 output is selected when this bit is cleared, and the PFI0/TRIG1 input is selected when this bit is set. When the PFI0/TRIG1 input is being used for an analog signal, it must be disconnected from the DAQ-STC PFI input. You can do this by clearing the Drive\_Trigger bit in the DAQ-STC. When the Drive\_Trigger bit is set, the PFI0/TRIG1 pin is connected to PFI0 on the DAQ-STC.

The high and low thresholds are set by an 8-bit serial DAC to be within  $\pm$  full scale. Writing a 0x00 to the DAC sets it for + full scale, while writing 0xFF sets it to - full scale. The selected input is compared against each of these thresholds by a comparator. The outputs of each comparator goes high when the input voltage is greater than the threshold. The outputs of these two comparators are connected to the DAQ-STC analog trigger inputs 0 and 1. The DAQ-STC circuitry uses these digital signals to generate the different slope and level detection modes. See the *DAQ-STC Technical Reference Manual* for their appropriate use. Figure 4-1 shows the analog trigger structure.

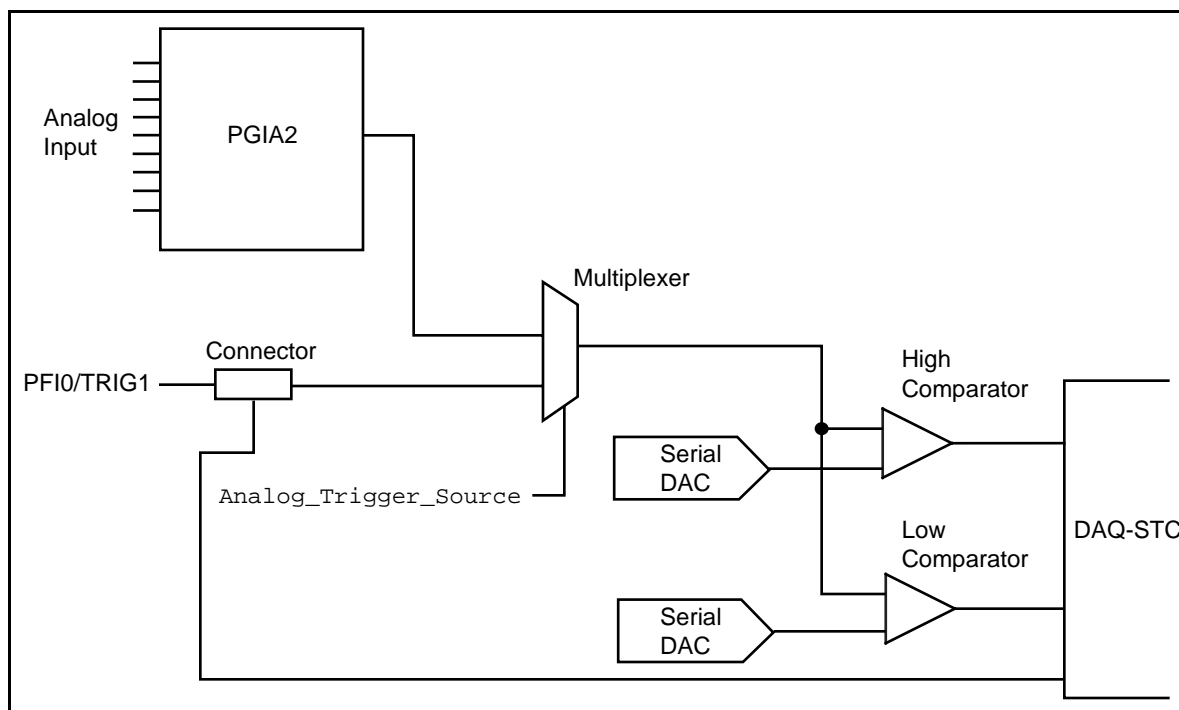


Figure 4-1. Analog Trigger Structure

To set the low and high analog thresholds, CALDACs 11 and 12 must be written with appropriate values. The protocol for writing to these DACs is the same as that for all the CALDACs mentioned in Chapter 5, *Calibration*, in this manual.

The function `Analog_Trigger_Control` enables analog triggering to set a mode of operation. When analog trigger is enabled, the analog trigger signal takes over the PFI0/TRIG1 slot, and this pin can no longer be used as input. For more information about this function, see the *Programming Analog Trigger* section of Chapter 2 in the *DAQ-STC Technical Reference Manual*.

## Interrupt Programming

Chapter 8, *Interrupt Control*, in the *DAQ-STC Technical Reference Manual*, discusses the interrupt programming aspect of the AT-MIO E Series boards.

There are two groups—Interrupt Group A and Interrupt Group B. Group A handles the analog input interrupts, general-purpose Counter 0 interrupts, and one pass-through interrupt. The Group A pass-through interrupt is connected to the DIO interrupt on the AT-MIO-16DE-10, which is the ORed output of the 8255 Port C pins PC0 and PC3. Group B handles the analog output interrupts, general purpose counter 1 interrupts, and one pass-through interrupt. The Group B pass-through interrupt is connected to the DMA TC interrupt, which will be asserted whenever the individual DMA TC interrupts are enabled and the TC occurs.

The functions `MSC_IRQ_Configure` and `MSC_Pass_Through_Polarity` allows you to select one of the IRQ 0-7 lines for Group A and one for Group B and allows the enabling of pass-through interrupts respectively. The *Interrupt Control* section also describes two interrupt programs, one for Group A and one for Group B, which are skeletons of the actual interrupt service routines. These programs do not address the programming of the interrupt controller.

## DMA Programming

You can program your AT-MIO E Series board so that the analog input, analog output, or general purpose counter/timers can generate DMA requests under appropriate circumstances. There are three logical DMA channels—A, B, and C. There are three registers—Channel A Mode Register (address 0x03), Channel B Mode Register (address 0x05), and Channel C Mode Register (address 0x07)—corresponding to these logical channels. These registers have bits to enable the logical channels, enable interrupts on terminal count, define the transfer type and finally define which physical DMA channel is associated with the logical channel.

Each logical channel, in turn, can service either analog input, analog output, or the general-purpose counter/timers. You must program the AO AI Select Register (address 0x09) and the GO GI Select Register (address 0x0B) to assign particular logical channels to either AI, AO, or GPCTs. Figure 4-2 shows the three-stage DMA structure.

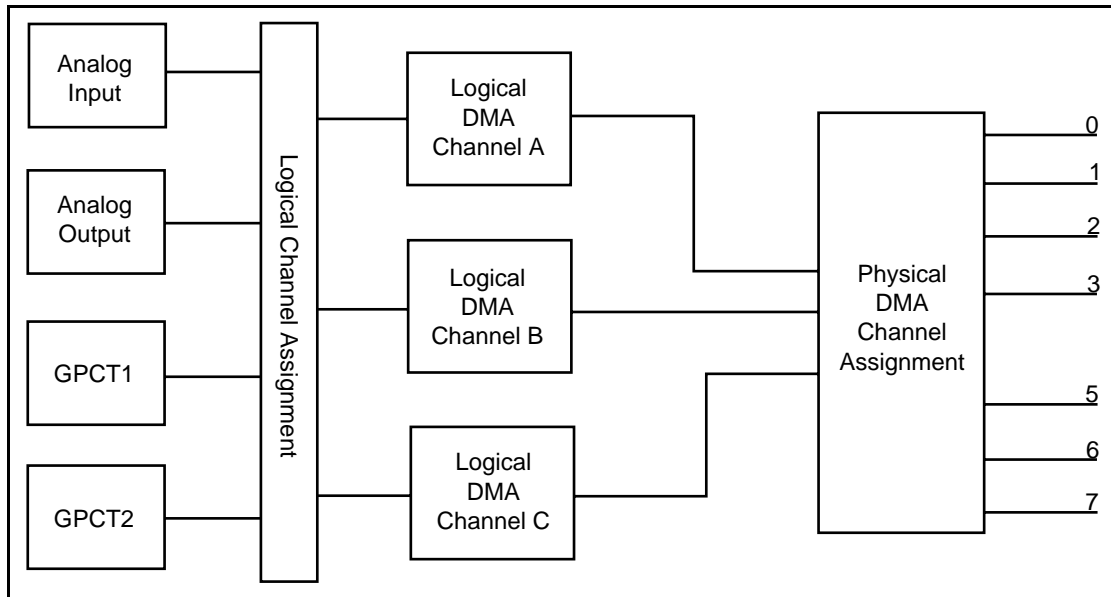


Figure 4-2. DMA Structure

Make sure that the same logical channel is not assigned to more than one resource. Also, the same physical channel number should not be assigned to more than one logical channel.

DMA requests are generated when all of the above mentioned initializations are done and when the source is programmed appropriately. For analog input, the AIFREQ signal from the DAQ-STC is used as the DMA source. For more information, see Chapter 2 of the *DAQ-STC Technical Reference Manual*. By using the `FIFO_Request_Selection` function, you can generate DMA requests based on any of the following conditions:

- FIFO not empty
- FIFO half-full
- FIFO full
- FIFO half-full until FIFO is empty

For analog output, the AOFREQ signal from the DAQ-STC is used as the DMA source. By using the `AO_FIFO` function you can generate DMA requests based on any of the following conditions:

- FIFO empty
- FIFO less than half-full
- FIFO not full
- Assert on FIFO half-full and deassert on FIFO full

For general-purpose counter/timers, an interrupt is produced in buffered modes, such as the buffered event counting, the buffered period measurement, and so on. The secondary bank of interrupts in the DAQ-STC is used for generating DMA requests for the general-purpose counter/timers.

## Single Channel Versus Dual Channel DMA

If single channel DMA is used, the DMA controller uses a single channel to transfer data. The DMA controller writes to or reads from a buffer in memory. The maximum number of bytes that can be transferred vary from platform to platform, with 64 kbytes in PCs, 128 kbytes in ISA to 16 MB in EISA. Regardless of the size, after the maximum bytes have been transferred, the DMA controller issues a TC which generates an interrupt, if the board is programmed for DMA. Usually, the interrupt latency is long enough so that, by the time you reprogram your DMA controller, some data is lost. This affects sustained throughput.

In dual channel DMA, two buffers are allocated, one for each DMA channel. Initially, one channel transfers data. When it issues a TC, hardware circuitry in the AT-MIO E Series board automatically switches to the other channel, which is previously programmed. In the meantime, the first channel is reprogrammed and kept ready so that, when the second channel issues TC, the AT-MIO E Series board switches back to the first channel. When one buffer is being serviced, the CPU can access the other buffer. This way, seamless data transfer is possible.

Dual-channel DMA can be used in analog output, or for the general purpose counter/timers. Only the AT-MIO-16E-1 supports dual-channel DMA for analog input. Notice that in analog output when a single DMA channel is serving both DACs, the buffer must contain interleaved data, meaning that every other word is written to a DAC. In dual channel DMA, there is one buffer for each DMA channel and within each buffer data is interleaved.

Also notice that whenever a TC interrupt is generated, the corresponding DMATCCLR bit in the Strobes Register (address 0x01) must be strobed. This clears the TC status bits for that logical DMA channel.

# Chapter 5

## Calibration

---

This chapter explains how to calibrate the analog input and output sections of the AT-MIO E Series boards by reading calibration constants from the EEPROM and writing them to the calibration DACs. This chapter also explains how to generate the calibration constants using NI-DAQ.

All AT-MIO E Series boards are factory calibrated before shipment, and the resulting calibration constants are stored in the EEPROM. Because the calibration DACs have no memory capability, they do not retain calibration information when the computer is turned off. Therefore, they must be reloaded every time the computer is turned on, and the most straightforward method is to copy these values from the EEPROM. In addition to the factory calibration constants, new calibration constants can be generated in the field through the use of the NI-DAQ calibration function call. Generating new constants results in a more accurate calibration for the actual environment in which the board is used.

### EEPROM

The EEPROM is used to store all non-volatile information about the board, including the factory and user calibration constants. The AT-MIO E Series boards use a XICOR X25040 EEPROM, which is 512 by 8 bits in size and has a serial interface. The signals used to interface to the EEPROM are clock, data in, data out, and chip select.

The Serial Command Register has three bits—SerClk (bit 0), SerData (bit 1), and EEPROMCs (bit 2)—that are connected to the EEPROM clock, data in, and chip select pins, respectively. PROMOut (bit 0) of the Status Register is connected to the EEPROM data out pin.

The format for reading from the EEPROM is very straightforward. The basic read cycle consists of shifting an 8-bit instruction and 9-bit address into the EEPROM, then shifting an 8-bit data out of the EEPROM. The timing diagram for the read cycle is shown in Figure 5-1.

**Note:** *Review the timing diagram and specifications very carefully before attempting to write code. Do not attempt to write to the EEPROM. If the factory area of the EEPROM (the upper 128 bytes) or the Plug and Play area (the lower 256 bytes) is lost, the board can be rendered inoperable. In this situation, you will have to send the board back to National Instruments to be reprogrammed. National Instruments is NOT liable for such mistakes, and you will have to bear the full expense of the RMA.*



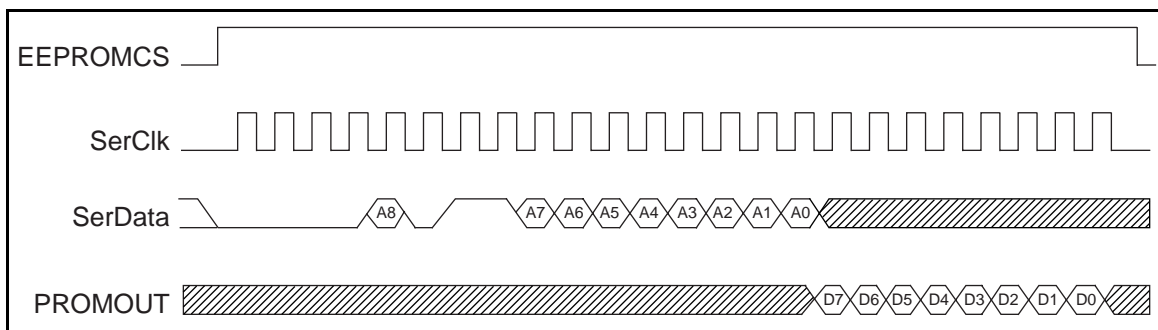


Figure 5-1. EEPROM Read Timing

Notice that, because the CalDACs and the EEPROM share the same clock and data in lines, it might seem that writing to the CalDACs could result in accidental writes to the EEPROM. However, this is not true. A write cycle to the EEPROM needs the chip select bit asserted. While writing to the CalDACs, make sure that this bit is cleared. Clearing this bit will ensure that no writes to the EEPROM will occur. It might also seem as though an access to the EEPROM could result in an access to the CalDACs, but this is also not true. The CalDACs will be updated only when the LdCalDAC<1..0> bit is pulsed.

Tables 5-1, 5-2, and 5-3 show a selected portion of the EEPROM map for each of the boards. The lower 256 bytes of the EEPROM contain valuable information pertaining to Plug and Play. The upper 256 bytes are divided into two sections of 128 locations each. The uppermost 128 locations contain factory data. The lower section of 128 locations contains the user calibration constants. There are five user calibration constants sections that use a format identical to the factory calibration section. These user areas start at location 371 in the EEPROM.

Table 5-1. AT-MIO-16E-1, AT-MIO-16E-2, AT-MIO-64E-3 EEPROM Map

Address	Data	Description
511	Board Code*	NI-DAQ Board Code
510	Revision	Revision
509	Sub-revision	Sub-revision
508	Year	Year of last factory calibration
507	Month	Month of last factory calibration
506	Day	Day of last factory calibration
426	0	Factory Reference MSB
425	0	Factory Reference LSB
424	0	Factory CALDAC 4 value (AI) (8-bit)
423	0	Factory CALDAC 1 value (AI) (8-bit)

(continues)

Table 5-1. AT-MIO-16E-1, AT-MIO-16E-2, AT-MIO-64E-3 EEPROM Map (Continued)

Address	Data	Description
422	0	Factory CALDAC 3 value (AI) (8-bit)
421	0	Factory CALDAC 2 value (AI) (8-bit)
420	0	Factory CALDAC 5 bipolar value (AO) (8-bit)
419	0	Factory CALDAC 7 bipolar value (AO) (8-bit)
418	0	Factory CALDAC 6 bipolar value (AO) (8-bit)
417	0	Factory CALDAC 8 bipolar value (AO) (8-bit)
416	0	Factory CALDAC 10 bipolar value (AO) (8-bit)
415	0	Factory CALDAC 9 bipolar value (AO) (8-bit)
414	0	Factory CALDAC 5 unipolar value (AO) (8-bit)
413	0	Factory CALDAC 7 unipolar value (AO) (8-bit)
412	0	Factory CALDAC 6 unipolar value (AO) (8-bit)
411	0	Factory CALDAC 8 unipolar value (AO) (8-bit)
410	0	Factory CALDAC 10 unipolar value (AO) (8-bit)
409	0	Factory CALDAC 9 unipolar value (AO) (8-bit)
371	0	Start of the 5 user calibration sections

\* Board Codes:

44—AT-MIO-16E-1

25—AT-MIO-16E-2

38—AT-MIO-64E-3

Table 5-2. AT-MIO-16E-10 and AT-MIO-16DE-10 EEPROM Map

Address	Data	Description
511	Board Code*	NI-DAQ Board Code
510	Revision	Revision
509	Sub-revision	Sub-revision
508	Year	Year of last factory calibration
507	Month	Month of last factory calibration
506	Day	Day of last factory calibration
424	0	Factory Reference MSB
423	0	Factory Reference LSB
422	0	Factory CALDAC 4 value (AI) (8-bit)
421	0	Factory CALDAC 11 value (AI) (8-bit)

(continues)

Table 5-2. AT-MIO-16E-10 and AT-MIO-16DE-10 EEPROM Map (Continued)

Address	Data	Description
420	0	Factory CALDAC 1 value (AI) (8-bit)
419	0	Factory CALDAC 3 value (AI) (8-bit)
418	0	Factory CALDAC 2 value (AI) (8-bit)
417	0	Factory CALDAC 5 bipolar value (AO) (8-bit)
416	0	Factory CALDAC 7 bipolar value (AO) (8-bit)
415	0	Factory CALDAC 6 bipolar value (AO) (8-bit)
414	0	Factory CALDAC 8 bipolar value (AO) (8-bit)
413	0	Factory CALDAC 10 bipolar value (AO) (8-bit)
412	0	Factory CALDAC 9 bipolar value (AO) (8-bit)
411	0	Factory CALDAC 5 unipolar value (AO) (8-bit)
410	0	Factory CALDAC 7 unipolar value (AO) (8-bit)
409	0	Factory CALDAC 6 unipolar value (AO) (8-bit)
408	0	Factory CALDAC 8 unipolar value (AO) (8-bit)
407	0	Factory CALDAC 10 unipolar value (AO) (8-bit)
406	0	Factory CALDAC 9 unipolar value (AO) (8-bit)
371	0	Start of the 5 user calibration sections

- \* Board Codes:  
 36—AT-MIO-16E-10  
 37—AT-MIO-16DE-10

Table 5-3. AT-MIO-16XE-50 EEPROM Map

Address	Data	Description
511	39	NI-DAQ Board Code
510	Revision	Revision
509	Sub-revision	Sub-revision
508	Year	Year of last factory calibration
507	Month	Month of last factory calibration
506	Day	Day of last factory calibration
438	0	Factory Reference MSB
437	0	Factory Reference LSB
436	0	Factory CALDAC 0 value MSB (AI)–bipolar (12-bit)

(continues)

Table 5-3. AT-MIO-16XE-50 EEPROM Map (Continued)

Address	Data	Description
435	0	Factory CALDAC 0 value LSB (AI)–bipolar (12-bit)
434	0	Factory CALDAC 2 value (AI)– bipolar (8-bit)
433	0	Factory CALDAC 0 value (AI)–bipolar (8-bit)
432	0	Factory CALDAC 1 value (AI)– bipolar (8-bit)
431	0	Factory CALDAC 8 value MSB (AI)–unipolar (8-bit)
430	0	Factory CALDAC 8 value LSB (AI)–unipolar (8-bit)
429	0	Factory CALDAC 2 value (AI)–unipolar (8-bit)
428	0	Factory CALDAC 0 value (AI)–unipolar (8-bit)
427	0	Factory CALDAC 1 value (AI)–unipolar (8-bit)
426	0	Factory CALDAC 6 value (AO) (8-bit)
425	0	Factory CALDAC 4 value (AO) (8-bit)
424	0	Factory CALDAC 7 value (AO) (8-bit)
423	0	Factory CALDAC 5 value (AO) (8-bit)
371	0	Start of the 5 user calibration sections

## Calibration DACs

The calibration DACs are used to adjust the analog signal paths for errors such as offset and gain. The 12-bit boards use an MB88341 serial DAC for calibration, which contains twelve 8-bit DACs. Notice that the last two DACs are used for the analog trigger circuitry on the AT-MIO-16E-1, AT-MIO-16E-2, and the AT-MIO-64E-3. The AT-MIO-16XE-50 uses a DAC8800 and DAC8043 serial DAC for calibration. The DAC8800 contains eight 8-bit DACs and the DAC8043 contains one 12-bit DAC.

The Serial Command Register has four bits—SerClk (bit 0), SerData (bit 1), SerDacLd0 (bit 3), and SerDacLd1 (bit 4)—that are connected to the serial DAC clock, data in, load for the 8-bit DACs, and load for the 12-bit DAC pins, respectively.

The format for writing to all of the serial DACs is similar to the EEPROM. The basic write cycle consists of shifting an address/data pair into the DAC, then pulsing the appropriate SerDacLd pin. The timing diagram for the write cycle for each DAC is shown in Figure 5-2 (a, b, c).

**Note:** *Review the timing diagram and specifications very carefully before attempting to write code.*

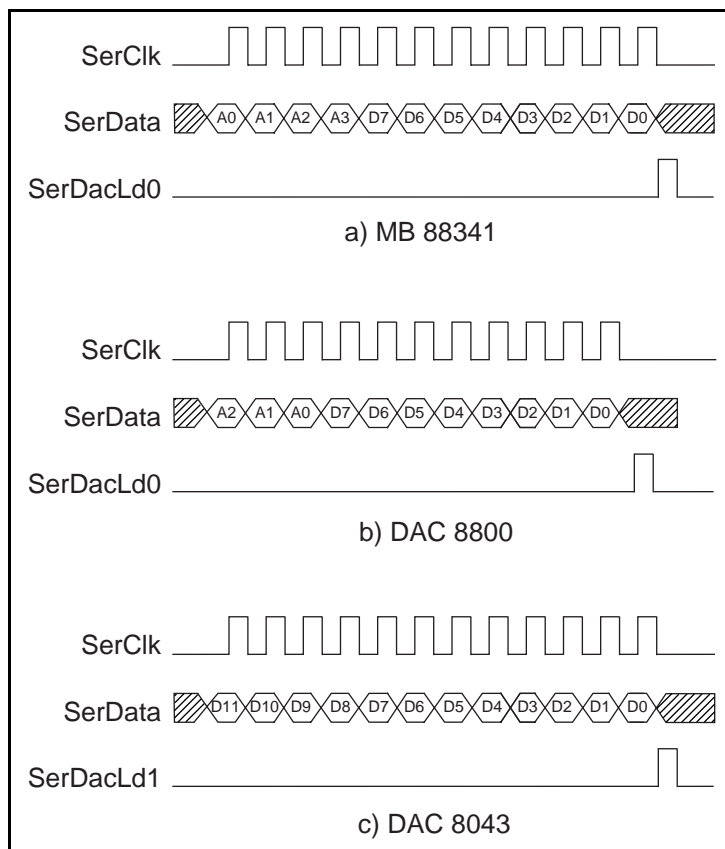


Figure 5-2. Calibration DAC Write Timing

## NI-DAQ Calibration Function

The NI-DAQ function called `Calibrate_E_Series` can calibrate the analog input, analog output, and internal reference on the AT-MIO E Series boards. Due to the complexity of the actual calibration algorithm, `Calibrate_E_Series` should be used to calibrate each section and store the results in the EEPROM. You can write a separate application using `Calibrate_E_Series`, which is run only whenever the board needs new calibration constants. Writing such an application allows the normal application to simply copy the calibration constants from the EEPROM and write them to the calibration DACs upon board initialization.

# Appendix A

## OKI MSM82C55A Data Sheet\*

---

This appendix contains a manufacturer data sheet for the MSM82C55A CMOS programmable peripheral interface (OKI Semiconductor). This interface is used on the AT-MIO E Series boards.

\* Copyright © OKI Semiconductor, 1993. Reprinted with permission of copyright owner.  
All rights reserved.  
OKI Semiconductor. *Microprocessor Data Book 1993.*

# Appendix B

## Customer Communication

---

For your convenience, this appendix contains a form you can use to comment on the product documentation.

National Instruments provides comprehensive technical assistance around the world. In the U.S. and Canada, applications engineers are available Monday through Friday from 8:00 a.m. to 6:00 p.m. (central time). In other countries, contact the nearest branch office. You may fax questions to us at any time.

### Corporate Headquarters

(512) 795-8248

Technical support fax: (800) 328-2203  
(512) 794-5678

<b>Branch Offices</b>	<b>Phone Number</b>	<b>Fax Number</b>
Australia	(03) 879 9422	(03) 879 9179
Austria	(0662) 435986	(0662) 437010-19
Belgium	02/757.00.20	02/757.03.11
Denmark	45 76 26 00	45 76 71 11
Finland	(90) 527 2321	(90) 502 2930
France	(1) 48 14 24 24	(1) 48 14 24 14
Germany	089/741 31 30	089/714 60 35
Hong Kong	02 2637 5019	02 2686 8505
Italy	02/48301892	02/48301915
Japan	(03) 3788-1921	(03) 3788-1923
Korea	02 596-7456	02 596-7455
Mexico	05 202 2544	05 202 2544
Netherlands	03480-33466	03480-30673
Norway	32-848400	32-848600
Singapore	2265886	2265887
Spain	(1) 640 0085	(1) 640 0533
Sweden	08-730 49 70	08-730 43 70
Switzerland	056/20 51 51	056/20 51 55
Taiwan	62 377 1200	62 737 4644
U.K.	1635 523545	1635 523154

# Documentation Comment Form

---

National Instruments encourages you to comment on the documentation supplied with our products. This information helps us provide quality products to meet your needs.

Title: **AT-MIO E Series Register-Level Programmer Manual**

Edition Date: **May 1995**

Part Number: **340747B-01**

Please comment on the completeness, clarity, and organization of the manual.

---

---

---

---

---

---

---

---

If you find errors in the manual, please record the page numbers and describe the errors.

---

---

---

---

---

---

---

---

Thank you for your help.

Name \_\_\_\_\_

Title \_\_\_\_\_

Company \_\_\_\_\_

Address \_\_\_\_\_

Phone ( \_\_\_\_\_ ) \_\_\_\_\_

Mail to: Technical Publications  
National Instruments Corporation  
6504 Bridge Point Parkway, MS 53-02  
Austin, TX 78730-5039

Fax to: Technical Publications  
National Instruments Corporation  
MS 53-02  
(512) 794-5678



# Glossary

---

Prefix	Meaning	Value
p-	pico-	10 <sup>-12</sup>
n-	nano-	10 <sup>-9</sup>
μ-	micro-	10 <sup>-6</sup>
m-	milli-	10 <sup>-3</sup>
k-	kilo-	10 <sup>3</sup>
M-	mega-	10 <sup>6</sup>
G-	giga-	10 <sup>9</sup>

## Symbols

\* inverted bit (negative logic) if after a bit name  
 Ω ohms

## A

A amperes  
 AC alternating current  
 A/D analog-to-digital  
 ADC A/D converter  
 AIGND analog input ground signal  
 AOGND analog output ground signal  
 ASIC application-specific integrated circuit

## B

Bank bank select bit  
 BC buffer counter  
 BipDac bipolar DAC bit

## C

CALDAC calibration DAC  
 Chan channel select bit  
 Channel physical channel select bit  
 ChanType channel type bit  
 ChanEnable DMA channel enable bit  
 CONVERT convert signal

## D

D data bit  
 D/A digital-to-analog  
 DAC D/A converter

DAC0OUT	analog channel 0 output signal
DAC1OUT	analog channel 1 output signal
DACSe	DAC select bit
DAQ	data acquisition
DC	direct current
DitherEn	dither enable bit
DIV	divide by signal
DMA	direct memory access
DMATCA	DMA terminal count A bit
DmaTcAClr	DMA terminal count A clear bit
DmaTcBClr	DMA terminal count B clear bit
DMATCB	DMA terminal count B bit
DMATCC	DMA terminal count C bit
DmaTcCClr	DMA terminal count C clear bit

## E

EEPROM	electrically erasable programmable read-only memory
EEPromCS	EEPROM chip select bit
EXTREF	external reference signal
ExtRef	external reference for DAC bit
EXTTRIG	external trigger signal
EXTSTROBE*	external strobe signal

## F

FIFO	first-in-first-out
------	--------------------

## G

Gain	channel gain select bit
GenTrig	general trigger bit
ghost	a conversion that is performed but the data is thrown away
GPCT1	general-purpose counter timer 1 bit
GPCT0	general-purpose counter timer 0 bit
GroundRef	ground reference bit

## H

hex	hexadecimal
Hz	hertz

## I

Input	analog input bit
Int/Ext Trig	internal/external analog trigger
I/O	input/output
IRQ	interrupt request signal
ISA	Industry Standard Architecture

## L

LASTCHANNEL	last channel bit
LSB	least significant bit

**M**

m	meters
MB	megabytes of memory
MSB	most significant bit
MUX	multiplexer

**N**

NRSE	nonreferenced single-ended input
------	----------------------------------

**O**

op-amp	operational amplifiers
Output	analog output bit

**P**

PFI0/Trig1	PFI 0/Trigger 1 signal
PFI1/Trig2	PFI 1/Trigger 2 signal
PGIA	Programmable Gain Instrumentation Amplifier
ppm	parts per million
PRETRIG	pretrigger signal
PROMOUT	EEPROM output data bit

**R**

ReGlitch	reglitch DAC bit
RTD	resistance-temperature detector
RTSI	Real-Time System Integration bus
RTSI_BRDO	RTSI board .

**S**

S	samples
s	second
SC	scan counter
SerClk	serial clock bit
SerDacLd	serial DAC load bit
SerData	serial data bit
SHIFTIN	shift in signal
SI	scan interval counter
SI2	sample interval
START	start signal
STOP	stop signal

**T**

TC	terminal count
TCIntEnable	DMATC interrupt enable bit
Transfer	transfer type bit
TTL	transistor-transistor logic

*Glossary*

**U**

UC	update counter
UI	update interval
UI2	update interval 2
Unip/Bip	channel unipolar/bipolar bit

**V**

V	volts
V <sub>ref</sub>	input voltage reference

**X**

X	don't care bits
---	-----------------

# Index

---

## A

- Activate\_PNP\_Board function, 4-3
  - ADC FIFO Clear Register
    - description, 3-24
    - register map, 3-2
  - ADC FIFO Data Register
    - description, 3-7
    - register map, 3-1
  - ADCs
    - single-read timing, 2-8 to 2-9
      - illustration, 2-8
    - theory of operation, 2-7
  - AI AO Select Register
    - description, 3-22
    - register map, 3-2
  - AI\_Arming function
    - AMUX-64T examples, 4-25, 4-28
    - STC programming examples, 4-11
      - single wire acquisition example, 4-22
    - with external start and stop trigger, 4-21
    - with external start trigger and scan start pulses, 4-18
    - with interrupts, 4-13
  - AI\_Board\_Environmentalize function, 4-7, 4-26
  - AI\_Board\_Personalize function, 4-7
  - AI\_End\_of\_Scan function, 4-9
  - AI\_FIFO\_Empty\_St bit, 2-9
  - AIFREQ signal, 4-45
  - AI\_Initialize\_Configuration\_Memory\_Output function, 4-8, 4-26
  - AI\_Interrupt\_Enable function, 4-13, 4-16
  - AI\_Reset\_All function, 4-7
  - AI\_Scan\_Start function
    - AMUX-64T examples
      - sampling one channel, 4-24
      - scanning eight channels, 4-27
    - sampling from channel 0, 4-8
    - STC scanning examples, 4-10
      - single wire acquisition, 4-22
      - with DMA, 4-15
    - with external start and stop trigger control, 4-20
    - with external start trigger and scan start, 4-18
    - with interrupts, 4-12
  - AI\_Start\_The\_Acquisition function
    - AMUX-64T examples
      - sampling one channel, 4-25
      - scanning eight channels, 4-28
    - STC scanning examples, 4-11
      - with DMA, 4-16
      - with interrupts, 4-14
      - with single wire acquisition, 4-23
  - AI\_Trigger\_Signals function, 4-7, 4-17, 4-19
  - AMUX-64T programming examples
    - sampling one channel, 4-23 to 4-25
    - scanning eight channels, 4-25 to 4-28
  - analog input circuitry, 2-6 to 2-8
    - block diagram, 2-6
  - analog input programming examples, 4-5 to 4-28
    - acquiring 20 scans, 4-19 to 4-21
    - acquiring one sample from channel 0, 4-6 to 4-9
  - AMUX-64T
    - sampling one channel, 4-23 to 4-25
    - scanning eight channels, 4-25 to 4-28
  - functions for programming, 4-5 to 4-6
  - single wire acquisition, 4-21 to 4-23
  - STC scanning, 4-9 to 4-11
    - with DMA, 4-14 to 4-17
    - with external start trigger and scan start pulses, 4-18 to 4-19
    - with interrupts, 4-11 to 4-14
- Analog Input Register Group
  - ADC FIFO Data Register, 3-7
  - Configuration Memory High Register, 3-10 to 3-13
  - Configuration Memory Low Register, 3-8 to 3-9
  - overview, 3-6
  - register map, 3-1
- analog output circuitry, 2-16 to 2-17
  - block diagram, 2-16

analog output programming examples,  
4-28 to 4-40

- CPU write to DAC, 4-29 to 4-30
- functions for programming, 4-28
- waveform generation
  - using interrupts, 4-40 to 4-41
  - using local buffer mode, 4-36 to 4-38
  - with external UPDATE and external trigger, 4-38 to 4-39
  - with polled writes, 4-30 to 4-35

Analog Output Register Group

- AO Configuration Register, 3-14 to 3-15
- DAC FIFO Data Register, 3-15
- DAC0 Direct Data Register, 3-16
- DAC1 Direct Data Register, 3-17
- overview, 3-13
- register map, 3-2

analog output timing circuitry, 2-17 to 2-18

- single-point output, 2-17 to 2-18
- waveform generation, 2-18

analog triggering

- programming considerations,  
4-42 to 4-44
  - trigger structure (illustration), 4-43
  - theory of operation, 2-15

Analog\_Trigger\_Control function, 4-44

AO Configuration Register

- description, 3-14 to 3-15
- register map, 3-2

AO\_Arming function, 4-35, 4-38, 4-40

AO\_Board\_Personalize function, 4-29, 4-31

AO\_Channels function, 4-34

AO\_Counting function, 4-32, 4-36, 4-39

AO\_Errors\_To\_Stop\_On function, 4-34

AO\_FIFO function, 4-35, 4-37, 4-40

AO\_FREQ signal, 4-45

AO\_LDAC\_Source\_And\_Update\_Mode  
function, 4-30, 4-34

AO\_Reset\_All function, 4-29, 4-31

AO\_Start\_The\_Acquisition function,  
4-35, 4-41

AO\_Triggering function, 4-32, 4-38

AO\_Updating function, 4-33, 4-36, 4-39

ARM\_DMA\_Controller() function, 4-16

Assign\_PNP\_Base\_Address function, 4-2

Assign\_PNP\_Data\_Port\_Address  
function, 4-2

AT-MIO E series boards. *See also* theory  
of operation.

block diagrams

- AT-MIO-16E-1, AT-MIO-16E-2,  
and AT-MO-64E-3, 2-1
- AT-MIO-16E-10,  
AT-MIO-16DE-10, 2-2
- AT-MIO-16XE-50, 2-3

characteristics, 1-1

features (table), 1-2

## B

Bank<1..0> bits, 3-10

BipDac bit, 3-14 to 3-15

bipolar output, 2-17

bitfields, 4-1

bits

- AI\_FIFO\_Empty\_St, 2-9
- Bank<1..0>, 3-10
- BipDac, 3-14 to 3-15
- Chan<3..0>, 3-11 to 3-13
- ChanEnable, 3-19, 3-20, 3-21
- Channel<2..0>, 3-19, 3-20, 3-21
- ChanType<2..0>, 3-10
- D<11..0>, 3-15, 3-16, 3-17
- D<15..0>, 3-7
- DACSel, 3-14
- DitherEn, 3-8
- DMATCA, 3-6
- DmaTcAClr, 3-18
- DMATCB, 3-6
- DmaTcBClr, 3-18
- DMATCC, 3-6
- DmaTcCClr, 3-18
- DOTRIG0, 4-42
- ECIntEnable, 3-19
- EEPromCS, 3-4, 5-1
- ExtRef, 3-14
- Gain<2..0>, 3-9
- GenTrig, 3-8
- GPCT0<C..A>, 3-23
- GPCT1<C..A>, 3-23
- GroundRef, 3-14
- Input<C..A>, 3-22
- Int/Ext Trig, 3-5
- LastChan, 3-8
- LASTCHANNEL, 2-9
- Output<C..A>, 3-22
- PROMOUT, 3-6

- ReGlitch, 3-14
  - SerClk, 3-4, 5-1, 5-5
  - SerDacLd0, 3-4, 5-5
  - SerDacLd1, 3-4, 5-5
  - SerData, 3-4, 5-5
  - TCIntEnable, 3-19, 3-20, 3-21
  - Transfer<2..0>, 3-19, 3-20, 3-21
  - Unip/Bip, 3-9
  - Board Environment Registers, 4-1
  - Board\_Read function, 4-3
  - Board\_Write function, 4-3
- C**
- CALDACs. *See* calibration DACs.
  - Calibrate\_E\_Series function, 5-6
  - calibration
    - circuitry, 2-8
    - EEPROM
      - AT-MIO-16E-1, AT-MIO-16E-2,  
and AT-MIO-64E-3 map,  
5-2 to 5-3
      - AT-MIO-16E-10 and  
AT-MIO-16DE-10 map, 5-3 to 5-4
      - AT-MIO-16XE-50 map, 5-4 to 5-5
      - calibration constant storage, 2-8, 5-1
      - reading from, 5-1 to 5-2
      - writing to accidentally (note), 5-1
    - NI-DAQ calibration function, 5-6
  - calibration DACs
    - analog triggering programming  
considerations, 4-42 to 4-43
    - purpose and use, 5-5
    - write timing diagram, 5-6
    - writing to, 5-5
  - Chan<3..0> bits
    - calibration channel assignments  
(table), 3-11
    - channel assignments (table), 3-13
    - description, 3-11
    - differential channel assignments  
(table), 3-11
    - nonreferenced single-ended channel  
assignments (table), 3-12
    - referenced single-ended channel  
assignments (table), 3-12
  - ChanEnable bit, 3-20
    - Channel A Mode Register, 3-19
    - Channel B Mode Register, 3-20
    - Channel C Mode Register, 3-21
  - Channel A Mode Register
    - description, 3-19
    - register map, 3-2
  - Channel B Mode Register, 3-20
    - description, 3-20
    - register map, 3-2
  - Channel C Mode Register
    - description, 3-21
    - register map, 3-2
  - Channel<2..0> bits
    - Channel A Mode Register, 3-19
    - Channel B Mode Register, 3-20
    - Channel C Mode Register, 3-21
  - channels
    - calibration channel assignments  
(table), 3-11
    - channel assignments (table), 3-13
    - differential channel assignments  
(table), 3-11
    - nonreferenced single-ended channel  
assignments (table), 3-12
    - referenced single-ended channel  
assignments (table), 3-12
    - valid channel types (table), 3-10
  - ChanType<2..0> bit, 3-10
  - Clear\_FIFO function, 4-7
  - configuration memory
    - definition, 2-6
    - multirate scanning without ghost  
(table), 2-14
  - Configuration Memory Clear Register
    - description, 3-24
    - register map, 3-2
  - Configuration Memory High Register
    - description, 3-10 to 3-13
    - register map, 3-1
  - Configuration Memory Low Register,  
3-8 to 3-9
    - description, 3-8 to 3-9
    - register map, 3-1
  - Configure\_Board function, 4-6
  - CONVERT\* signal
    - data acquisition sequence timing, 2-9
    - initiating conversions, 2-8
    - RTSI trigger lines programming  
considerations, 4-42
  - Convert\_Signal function

- AMUX-64T examples
    - sampling one channel, 4-24
    - scanning eight channels, 4-27
  - STC scanning examples, 4-10
    - single wire acquisition, 4-22
    - with DMA, 4-15
    - with external start and stop trigger, 4-20
    - with external start trigger and scan start, 4-18
    - with interrupts, 4-13
  - counter/timer, programming, 4-41 to 4-42
  - customer communication, *xi*, B-1
- D**
- D<11..0> bits
    - DAC FIFO Data Register, 3-15
    - DAC0 Direct Data Register, 3-16
    - DAC1 Direct Data Register, 3-17
  - D<15..0> bits, 3-7
  - DAC, analog output circuitry, 2-16 to 2-17
  - DAC FIFO Clear Register
    - description, 3-25
    - register map, 3-2
  - DAC FIFO Data Register
    - description, 3-15
    - register map, 3-2
  - DAC0 Direct Data Register, 3-16
    - description, 3-16
    - register map, 3-2
  - DAC1 Direct Data Register
    - description, 3-17
    - register map, 3-2
  - DACSel bit, 3-14
  - DAQ-PnP support for Plug and Play specification, 2-5
  - DAQ-STC programming examples. *See* programming examples.
  - DAQ-STC Register Group
    - Board Environment Registers, 4-1
    - overview, 3-24
    - register map, 3-2
  - DAQ-STC system timing controller, *x*
    - counter diagram, 2-19
    - programming. *See* programming examples.
  - DAQ\_STC\_Windowed\_Mode\_Read function, 4-3
  - DAQ\_STC\_Windowed\_Mode\_Write function, 4-3
  - data acquisition timing circuitry, 2-8 to 2-14
    - ADC timing (illustration), 2-8
    - block diagram, 2-6
    - data acquisition sequence timing, 2-9 to 2-14
    - multirate scanning with ghost, 2-12 to 2-14
      - analog input configuration memory (table), 2-14
      - illustration, 2-13
      - occurrences of conversion on channel 1 (illustration), 2-13
      - successive scans (illustration), 2-13
    - multirate scanning without ghost, 2-10 to 2-12
      - scanning three channels with 4:2:1 sampling rate (illustration), 2-12
      - scanning two channels (illustration), 2-11
        - 1:x sampling rate, 2-11
        - 3:1:1 sampling rate, 2-12
      - single-read timing, 2-8 to 2-9
      - timing of scan (illustration), 2-10
  - differential channel assignments (table), 3-11
  - digital I/O circuitry, 2-19
  - digital I/O programming examples
    - performing digital I/O, 4-5
    - windowed registers, 4-4
  - DIO Register Group
    - description, 3-24
    - register map, 3-2
  - dither circuitry, 2-7
  - DitherEn bit, 3-8
  - DIV counter, 2-9
  - DMA Control Register Group
    - AI AO Select Register, 3-22
    - Channel A Mode Register, 3-19
    - Channel B Mode Register, 3-20
    - Channel C Mode Register, 3-21
    - G0 G1 Select Register, 3-23
    - overview, 3-18
    - register map, 3-2
    - Strobes Register, 3-18
  - DMA programming, 4-44 to 4-45



- analog input examples, 4-14 to 4-17
  - STC scanning with DMA, 4-14 to 4-17
- generating DMA requests, 4-45
- single channel vs. dual channel DMA, 4-46
- structure (illustration), 4-45
- DMATCA bit, 3-6
- DmaTcAClr bit, 3-18
- DMATCB bit, 3-6
- DmaTcBClr bit, 3-18
- DMATCC bit, 3-6
- DmaTcCClr bit, 3-18
- documentation
  - about this manual, *ix*
  - conventions used in manual, *x*
  - organization of manual, *ix-x*
  - related documentation, *xi*
- DOTRIG0 bit, 4-42

## E

### EEPROM

- AT-MIO-16E-1, AT-MIO-16E-2, and AT-MIO-64E-3 map, 5-2 to 5-3
- AT-MIO-16E-10 and AT-MIO-16DE-10 map, 5-3 to 5-4
- AT-MIO-16XE-50 map, 5-4 to 5-5
- calibration constant storage, 2-8, 5-1
- reading from, 5-1 to 5-2
- writing to accidentally (note), 5-1
- EEPromCS bit, 3-4, 5-1
- ESERFNCT.c example file, 4-3
- ESERRLP.h example file, 4-3
- ExtRef bit, 3-14
- EXTSTROBE\* signal, digital I/O circuitry, 2-19

## F

- fax technical support, B-1
- FIFO

- analog output, 2-17
- configuration memory, 2-6
- overflow, 2-8
- theory of operation, 2-7 to 2-8
- waveform generation, 2-18

- FIFO Strobe Register Group
  - ADC FIFO Clear Register, 3-24
  - Configuration Memory Clear Register, 3-24
  - DAC FIFO Clear Register, 3-25
  - register map, 3-2
- FIFO\_Request\_Selection function, 4-45
- files for example programs, 4-3

## G

- G0 G1 Select Register
  - description, 3-23
  - register map, 3-2
- Gain<2..0> bits, 3-9
- GATE signal, timing I/O circuitry, 2-19 to 2-20
- general purpose counter/timer, programming, 4-41 to 4-42
- GenTrig bit, 3-8
- GENTRIG0 signal, 4-42
- getvect() function, 4-12
- ghost channel
  - definition, 2-7
  - multirate scanning
    - with ghost, 2-12 to 2-14
    - without ghost, 2-10 to 2-12
- glitching, 2-17
- GPCT0<C..A> bits, 3-23
- GPCT1<C..A> bits, 3-23
- GroundRef bit, 3-14

## I

- initialization of Plug and Play, 4-1 to 4-3
- Input<C..A> bits, 3-22
- Int/Ext Trig bit, 3-5
- interrupt programming, 4-44
- Interrupt\_Service\_Routine function, 4-14
- ISA bus interface circuitry, 2-4 to 2-5

## K

- Kick\_Start\_FIFO function, 4-35, 4-40

**L**

LastChan bit, 3-8  
 LASTCHANNEL bit, 2-9

**M**

manual. *See* documentation.  
 Misc Command Register  
   description, 3-5  
   register map, 3-1  
 Misc Register Group  
   Misc Command Register, 3-5  
   overview, 3-3  
   register map, 3-1  
   Serial Command Register, 3-4  
   Status Register, 3-6  
 MSC\_Clock\_Configure function, 4-7, 4-31  
 MSC\_IRQ\_Configure function, 4-44  
 MSC\_Pass\_Through\_Polarity function, 4-44  
 MSC\_RTSI\_Pin\_Configure function, 4-42  
 MSM82C55A CMOS programmable  
   peripheral interface, A-1 to A-16  
 multirate scanning with ghost, 2-12 to 2-14  
   analog input configuration memory  
     (table), 2-14  
   illustration, 2-13  
   occurrences of conversion on channel 1  
     (illustration), 2-13  
   successive scans (illustration), 2-13  
 multirate scanning without ghost,  
 2-10 to 2-12  
   scanning three channels with 4:2:1  
     sampling rate (illustration), 2-12  
   scanning two channels  
     (illustration), 2-11  
     1:x sampling rate, 2-11  
     3:1:1 sampling rate, 2-12

**N**

nonreferenced single-ended channel  
   assignments (table), 3-12  
 Number\_of\_Scans function  
   AMUX-64T examples

  sampling one channel, 4-24  
   scanning eight channels, 4-26  
 STC scanning examples, 4-9  
   single wire acquisition, 4-22  
   with DMA, 4-14  
   with external start and stop  
     trigger, 4-19  
   with external start trigger and scan  
     start, 4-17

**O**

operation of AT-MIO E series boards.  
   *See* theory of operation.  
 OUT signal, timing I/O circuitry, 2-19  
 Output<C..A> bits, 3-22

**P**

PGIA (programmable gain instrumentation  
 amplifier)  
   analog trigger programming  
     considerations, 4-43  
     gain selection with Gain<2..0> bits, 3-9  
     theory of operation, 2-7  
 Plug and Play specification  
   DAQ-PnP support, 2-5  
   programming considerations, 4-1 to 4-3  
 PNPINIT.c example file, 4-3  
 posttrigger acquisition, 2-14 to 2-15  
 pretrigger acquisition, 2-14 to 2-15  
 Program\_DMA\_Controller() function,  
 4-16, 4-17  
 programmable gain instrumentation  
 amplifier (PGIA). *See* PGIA  
 (programmable gain  
 instrumentation amplifier).  
 programming  
   analog triggering, 4-42 to 4-44  
   DMA programming, 4-44 to 4-45  
     single channel vs. dual channel, 4-46  
   examples. *See* programming examples.  
   general-purpose counter/timer,  
     4-41 to 4-42  
   interrupt programming, 4-44  
   Plug and Play initialization, 4-1 to 4-3

- RTSI trigger lines considerations, 4-42
- windowing registers, 4-3
- programming examples
  - analog input, 4-5 to 4-28
    - acquiring 20 scans, 4-19 to 4-21
    - acquiring one sample from channel 0, 4-6 to 4-9
    - functions for programming, 4-5 to 4-6
    - sampling one channel on AMUX-64T, 4-23 to 4-25
    - scanning eight channels on AMUX-64T, 4-25 to 4-28
    - single wire acquisition, 4-21 to 4-23
    - STC scanning, 4-9 to 4-11
      - with DMA, 4-14 to 4-17
      - with external start trigger and scan start pulses, 4-18 to 4-19
      - with interrupts, 4-11 to 4-14
  - analog output
    - CPU write to DAC, 4-29 to 4-30
    - functions for programming, 4-28
    - waveform generation
      - using interrupts, 4-40 to 4-41
      - using local buffer mode, 4-36 to 4-38
      - with external UPDATE and external trigger, 4-38 to 4-39
      - with polled writes, 4-30 to 4-35
  - digital I/O, 4-4 to 4-5
  - files on companion disk, 4-3
- PROMOUT bit, 3-6, 5-1

## R

- referenced single-ended channel assignments (table), 3-12
- registers
  - Analog Input Register Group
    - ADC FIFO Data Register, 3-7
    - Configuration Memory High Register, 3-10 to 3-13
    - Configuration Memory Low Register, 3-8 to 3-9
    - overview, 3-6
  - Analog Output Register Group
    - AO Configuration Register, 3-14 to 3-15

- DAC FIFO Data Register, 3-15
- DAC0 Direct Data Register, 3-16
- DAC1 Direct Data Register, 3-17
  - overview, 3-13
- DAQ-STC Register Group, 3-24, 4-1
- DIO Register Group, 3-24
- DMA Control Register Group
  - AI AO Select Register, 3-22
  - Channel A Mode Register, 3-19
  - Channel B Mode Register, 3-20
  - Channel C Mode Register, 3-21
  - G0 G1 Select Register, 3-23
  - overview, 3-18
  - Strobes Register, 3-18
- FIFO Strobe Register Group
  - ADC FIFO Clear Register, 3-24
  - Configuration Memory Clear Register, 3-24
  - DAC FIFO Clear Register, 3-25
- Misc Register Group
  - Misc Command Register, 3-5
  - overview, 3-3
  - Serial Command Register, 3-4
  - Status Register, 3-6
- register maps, 3-1 to 3-2
- sizes, 3-3
- windowed registers
  - programming considerations, 4-3
  - register map, 3-2
- ReGlitch bit, 3-14
- reglitch circuitry, 2-17
- Reset\_PNP\_Registers function, 4-2
- RTSI bus interface circuitry, 2-20
  - block diagram, 2-20
- RTSI trigger lines, programming considerations, 4-12
- RTSI\_BRD0 signal, 4-42

## S

- sample interval (SI2), 2-9
- scan counter (SC), 2-9 to 2-10
- scan interval (SI), 2-9
- SCAN sequence
  - definition, 2-9
  - starting, 2-9
- scanning, multirate. *See* multirate scanning.
- SerClk bit

- connection to EEPROM clock, 5-1
- connection to serial DAC clock, 5-5
- description, 3-4
- SerDacLd0 bit
  - connection to serial DAC clock, 5-5
  - description, 3-4
- SerDacLd1 bit
  - connection to serial DAC clock, 5-5
  - description, 3-4
- SerData bit
  - connection to serial DAC clock, 5-5
  - description, 3-4
- Serial Command Register
  - description, 3-4
  - register map, 3-1
- service\_interrupt() routine, 4-41
- Set\_PNP\_Configuration\_State function, 4-2
- Set\_PNP\_Isolation\_State function, 4-2
- Setup\_PNP\_Board function, 4-3, 4-4
- setvect() function, 4-12
- SHIFTIN\* signal, ADC timing, 2-9
- single channel vs. dual channel DMA, 4-46
- single-point output, analog output timing circuitry, 2-17 to 2-18
- single-read timing, data acquisition timing circuitry, 2-8 to 2-9
- SOURCE signal, timing I/O circuitry, 2-19
- START signal, 2-9
- START1 signal, 2-14
- START2 signal, 2-14
- Status Register
  - description, 3-6
  - register map, 3-1
- STOP signal, 2-9
- Strobes Register
  - description, 3-18
  - register map, 3-2

## T

- TCIntEnable bit
  - Channel A Mode Register, 3-19
  - Channel B Mode Register, 3-20
  - Channel C Mode Register, 3-21
- technical support, B-1
- theory of operation
  - analog input circuitry, 2-6 to 2-8
  - block diagram, 2-6

- analog output circuitry, 2-16 to 2-17
  - block diagram, 2-16
- analog output timing circuitry, 2-17 to 2-18
  - single-point output, 2-17 to 2-18
  - waveform generation, 2-18
- analog triggering, 2-15
- block diagrams
  - AT-MIO-16E-1, AT-MIO-16E-2, and AT-MO-64E-3, 2-1
  - AT-MIO-16E-10,
  - AT-MIO-16DE-10, 2-2
  - AT-MIO-16XE-50, 2-3
- components of AT-MIO E series boards, 2-3
- data acquisition timing circuitry, 2-8 to 2-14
  - ADC timing (illustration), 2-8
  - block diagram, 2-6
  - data acquisition sequence timing, 2-9 to 2-14
  - multirate scanning with ghost, 2-12 to 2-14
  - multirate scanning without ghost, 2-10 to 2-12
  - single-read timing, 2-8 to 2-9
  - timing of scan (illustration), 2-10
- digital I/O circuitry, 2-19
- functional overview, 2-1 to 2-3
- ISA bus interface circuitry, 2-4 to 2-5
- posttrigger and pretrigger acquisition, 2-14 to 2-15
- RTSI bus interface circuitry, 2-20
  - block diagram, 2-20
- timing I/O circuitry, 2-19 to 2-20
  - DAQ-STC counter diagram, 2-19
- timing circuitry
  - analog output, 2-17 to 2-18
  - data acquisition, 2-8 to 2-14
    - ADC timing (illustration), 2-8
    - block diagram, 2-6
    - data acquisition sequence timing, 2-9 to 2-14
    - multirate scanning with ghost, 2-12 to 2-14
    - multirate scanning without ghost, 2-10 to 2-12
    - single-read timing, 2-8 to 2-9

- timing of scan (illustration), 2-10
- timing I/O circuitry, 2-19 to 2-20
  - DAQ-STC counter diagram, 2-19
- Transfer<2..0> bits
  - Channel A Mode Register, 3-19
  - Channel B Mode Register, 3-20
  - Channel C Mode Register, 3-21
- trigger lines, RTSI, programming
  - considerations, 4-42
- triggering
  - analog triggering
    - programming considerations, 4-42 to 4-44
    - theory of operation, 2-15
  - posttrigger and pretrigger acquisition, 2-14 to 2-15

## U

- Unip/Bip bit, 3-9
- unipolar output, 2-17
- update interval counter (UI), 2-18
- UPDOWN signal, timing I/O circuitry, 2-19

## V

- Verify\_PNP\_Base\_Address
  - function, 4-3

## W

- waveform generation, 2-18
- waveform generation programming
  - examples
    - using interrupts, 4-40 to 4-41
    - using local buffer mode, 4-36 to 4-38
    - with external UPDATE and external trigger, 4-38 to 4-39
    - with polled writes, 4-30 to 4-35
- windowed registers
  - digital I/O programming example, 4-4
  - programming considerations, 4-3
  - register map, 3-2
- Write\_PNP\_Initiation\_Key function, 4-2