ACCES I/O PRODUCTS, INC.

10623 Roselle Street, San Diego, CA 9212 • (858) 550-9559 • Fax (858) 550-7322
contactus@accesio.com • www.accesio.com

MODEL 104-AIO16-16E

USER MANUAL

FILE: M104-AIO16-16E.A1r

Notice

The information in this document is provided for reference only. ACCES does not assume any liability arising out of the application or use of the information or products described herein. This document may contain or reference information and products protected by copyrights or patents and does not convey any license under the patent rights of ACCES, nor the rights of others.

IBM PC, PC/XT, and PC/AT are registered trademarks of the International Business Machines Corporation.

WARNING!!

ALWAYS CONNECT AND DISCONNECT YOUR FIELD CABLING WITH THE COMPUTER POWER OFF. ALWAYS TURN COMPUTER POWER OFF BEFORE INSTALLING A CARD. CONNECTING AND DISCONNECTING CABLES, OR INSTALLING CARDS INTO A SYSTEM WITH THE COMPUTER OR FIELD POWER ON MAY CAUSE DAMAGE TO THE I/O CARD AND WILL VOID ALL WARRANTIES, IMPLIED OR EXPRESSED.

Warranty

Prior to shipment, ACCES equipment is thoroughly inspected and tested to applicable specifications. However, should equipment failure occur, ACCES assures its customers that prompt service and support will be available. Any equipment originally manufactured by ACCES which is found to be defective will be repaired or replaced subject to the following considerations.

Terms and Conditions

If a unit is suspected of failure, contact ACCES' Customer Service department. Be prepared to give the unit model number, serial number, and a description of the failure symptom(s). We may suggest some simple tests to confirm the failure. We will assign a Return Material Authorization (RMA) number which must appear on the outer label of the return package. All units/components should be properly packed for handling and returned with freight prepaid to the ACCES designated Service Center, and will be returned to the customer's/user's site freight prepaid and invoiced.

Coverage

First Three Years: Returned unit/part will be repaired and/or replaced at ACCES option with no charge for labor or parts not excluded by warranty. Warranty commences with equipment shipment.

Following Years: Throughout your equipment's lifetime, ACCES stands ready to provide on-site or in-plant service at reasonable rates similar to those of other manufacturers in the industry.

Equipment Not Manufactured by ACCES

Equipment provided but not manufactured by ACCES is warranted and will be repaired according to the terms and conditions of the respective equipment manufacturer's warranty.

General

Under this Warranty, liability of ACCES is limited to replacing, repairing or issuing credit (at ACCES discretion) for any products which are proved to be defective during the warranty period. In no case is ACCES liable for consequential or special damage arriving from use or misuse of our product. The customer is responsible for all charges caused by modifications or additions to ACCES equipment not approved in writing by ACCES or, if in ACCES opinion the equipment has been subjected to abnormal use. "Abnormal use" for purposes of this warranty is defined as any use to which the equipment is exposed other than that use specified or intended as evidenced by purchase or sales representation. Other than the above, no other warranty, expressed or implied, shall apply to any and all such equipment furnished or sold by ACCES.

Table of Contents

Chapter 1: Introduction

This board is a high-speed, 16-bit resolution Analog to Digital (A/D) multifunction board featuring an excellent price/performance value. It is very useful for precision PC/104-based data acquisition, control, or signal analysis in applications such as standalone environmental test stations, compact production test equipment, portable testers, avionics and others. In addition to direct data transfers, the board's ability to trigger the A/D in real time assures synchronized sampling that is unaffected by other computer operations. This is an essential requirement for signal, vibration and transient analysis where high data rates must be sustained for short periods of time. This high-speed sampling rate is supported by a FIFO for reducing processor overhead and is filtered for an extremely quiet front end. Sixteen parallel bits of digital I/O and two 12-bit D/A outputs allow for a complete, high-performance data acquisition solution.

Analog In

16 (Single-Ended) or 8 (Differential) 16-Bit Analog Inputs are provided. Any single channel can be acquired at 250,000 samples / second, and any range of channels can be acquired at 250,000 samples / second divided by the number of input channels your taking data on. The Analog Inputs feature hardware / software selectable ranges of 0-1V, 0-2V, 0-4V, 0-5V, 0-10V, ±0.5V, ±1V, ±2V, ±2.5V, ±5V, and ±10V. All the data from the A/D is placed in a 2KSample FIFO for the fastest possible data transfer. Conversions can be started using software commands, an internal burst mode, or with an external trigger via a pin at the connector.

Analog Out

Two 12-Bit Digital-to-Analog converter (DAC) outputs are provided. Output ranges of 0-5V and 0-10V are field selectable with jumpers, per channel.

Digital I/O

16 Buffered Digital Input/Outputs are provided. All 16 lines may be programmed as inputs or outputs or 8 lines may be inputs while the other 8 are outputs. All lines are pulled-up via resistors to 5 Volts.

Counter/Timer

The board uses an 82C54 Programmable Interval Timer (3 sixteen bit counters). The counters are generally used to time various aspects of the A/D conversion process. If you are using non-counter timed A/D modes, Counter 2 becomes available for frequency generation. Consult Chapter 4, Chapter 6, and Appendix B for more information on the A/D, programming, and the counter itself.

Calibration

This board features digitally controlled potentiometers which are used to adjust the gain and offset of the A/D function and the gain of the DAC function. This allows both the analog inputs and outputs to be calibrated from software in the field.

In order to obtain accurate data, the proper values must be loaded into the digital potentiometers each time the board is powered. If no constants are loaded, the potentiometers will power-on to the center of their ranges, which will result in a non- or poorly-calibrated situation. The calibration values are stored in an on-board EEPROM allowing simple re-use of calibration data from one boot to the next.

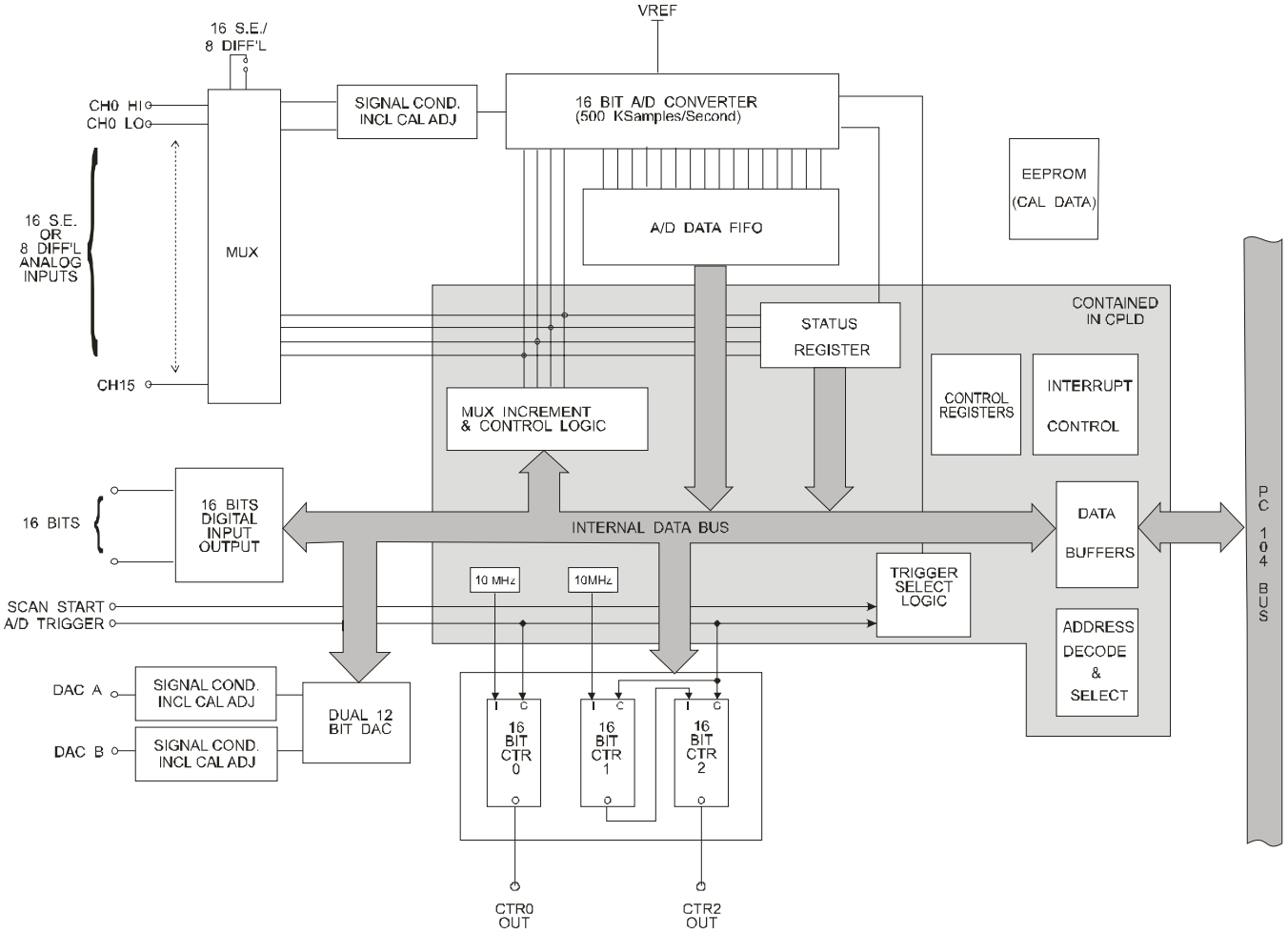


Figure 1-1: Block Diagram

Chapter 2: Installation

ENSURE YOU HAVE APPROPRIATE POWER IN YOUR PC/104 STACK TO OPERATE THIS DATA ACQUISITION BOARD! It requires +5V *and* +/-12V to operate properly unless you purchased the board with a DC/DC converter. The paper label on the board will indicate DC if a DC/DC converter is installed. The DC requires only +5V power at about 250mA.

A printed Quick-Start Guide (QSG) is packed with the board for your convenience. If you've already performed the steps from the QSG, you may find this chapter to be redundant and may skip forward to begin developing your application.

The software provided with this PC/104 Board is on CD and must be installed onto your hard disk prior to use. To do this, perform the following steps as appropriate for your operating system. Substitute the appropriate drive letter for your CD-ROM where you see d: in the examples below.

CD Installation

The following instructions assume the CD-ROM drive is drive "D". Please substitute the appropriate drive letter for your system as necessary.

DOS

1. Place the CD into your CD-ROM drive.
2. Type D : Enter to change the active drive to the CD-ROM drive.
3. Type I N S T A L L Enter to run the install program.
4. Follow the on-screen prompts to install the software for this board.

WINDOWS

1. Place the CD into your CD-ROM drive.
2. The system should automatically run the install program. If the install program does not run promptly, click START | RUN and type D : I N S T A L L, click OK or press Enter.
3. Follow the on-screen prompts to install the software for this board.

LINUX

1. Please refer to linux.htm on the CD-ROM for information on installing under linux.

Installing the Hardware

Before installing the board, carefully read Chapter 3 and Chapter 4 of this manual and configure the board according to your requirements. The SETUP Program can be used to assist in configuring jumpers on the board. Be especially careful with Address Selection. If the addresses of two installed functions overlap, you will experience unpredictable computer behavior. To help avoid this problem, refer to the FINDBASE.EXE program installed from the CD. The setup program does not set the options on the board, these must be set by jumpers.

To Install the Board

1. Install jumpers for selected options and base address according to your application requirements, as mentioned above.
2. Remove power from the PC/104 stack.
3. Assemble standoff hardware for stacking and securing the boards.
4. Carefully plug the board onto the PC/104 connector on the CPU or onto the stack, ensuring proper alignment of the pins before completely seating the connectors together.
5. Install I/O cables onto the board's I/O connectors and proceed to secure the stack together or repeat steps 3-5 until all boards are installed using the selected mounting hardware.
6. Check that all connections in your PC/104 stack are correct and secure then power up the system.
7. Run one of the provided sample programs appropriate for your operating system that was installed from the CD to test and validate your installation.

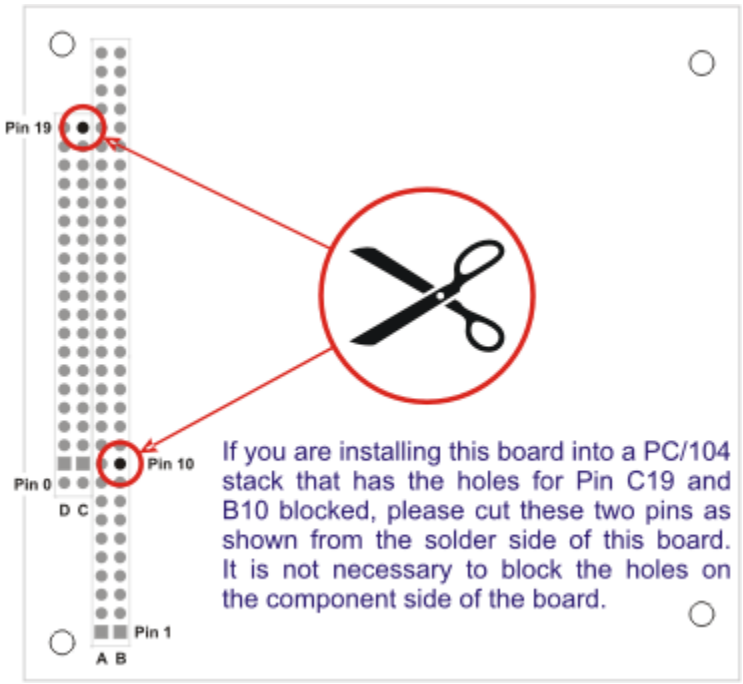


Figure 2-1: PC/104 Key Information

Chapter 3: Option Selection

Jumpers are available on the board to setup the following selections:

- Base address
- IRQ level
- DAC output voltage ranges
- A/D input mode (single ended or differential)
- Input Voltage Range

Please refer to the Setup Program on the Software Master CD for details of selecting the appropriate options for your application.

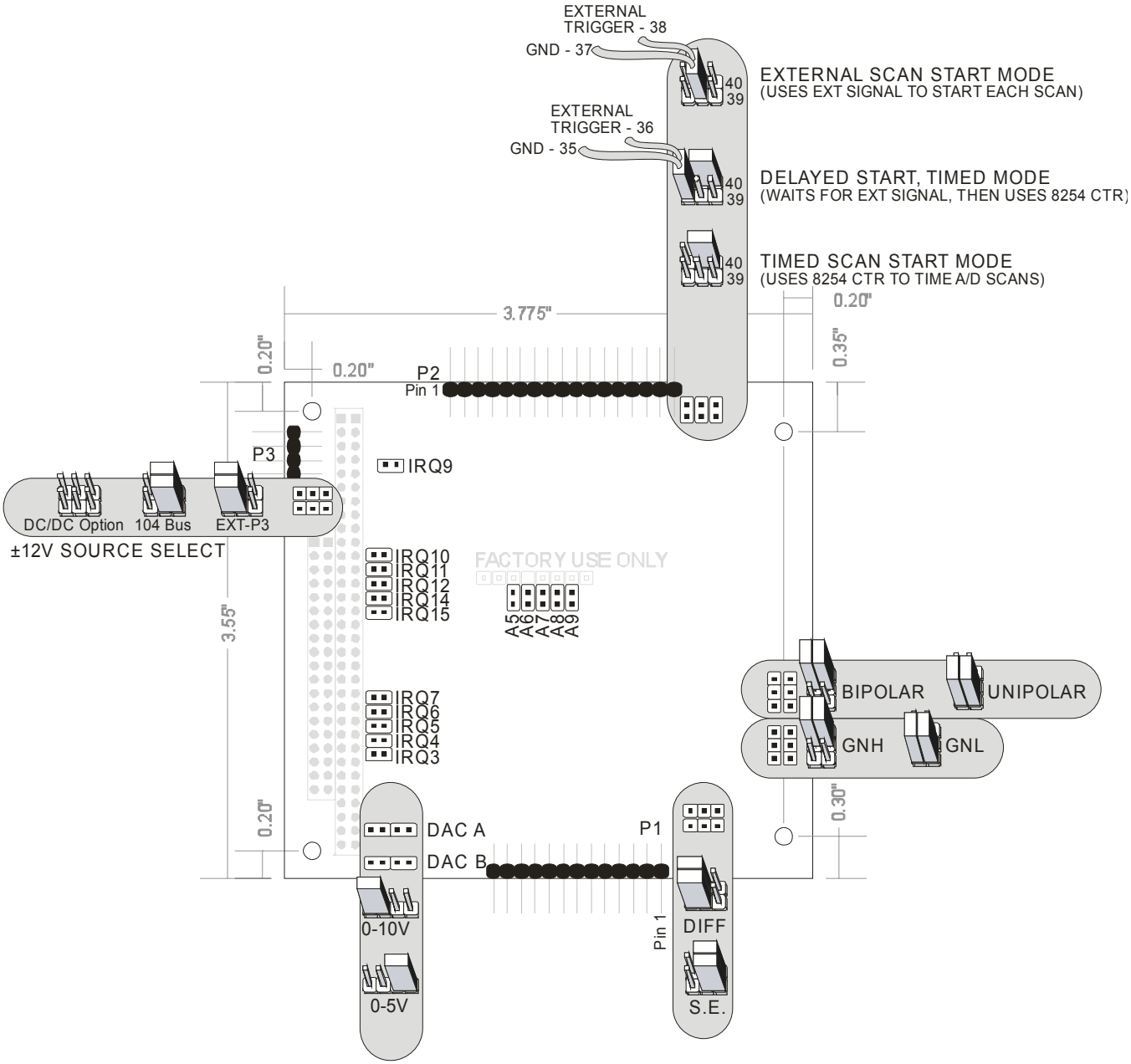


Figure 3-1: Option Selection Map

Address Selection

The board's Base Address is set by jumpers labeled "BOARD ADDR." The jumpers are marked /A5 through /A9 and /A5 is the least significant bit of the address. The base addresses can be selected anywhere within the I/O address range 000-3E0 provided that they do not overlap with other functions. The FINDBASE software utility provided on the CD with your board will help you select a base address that does not conflict with other assignments.

This board requires a block of 32 addresses (20 hex). In order to configure the desired address, the hexadecimal address must be converted to a binary representation.

For example, as illustrated below, switch selection corresponds to hex **2C0 (or binary 10 110x xxxx).** The "xxxxx" represents address lines A4 through A0 used on the board to select individual registers as described in Chapter 6: Programming of the manual.

Hex Representation	2		C		
Conversion Factors	2	1	8	4	2
Binary Representation	1	0	1	1	0
Jumper Installed	NO	YES	NO	NO	YES
Jumper Label	A9	A8	A7	A6	A5

Table 3-1: Address Selection and Hex Conversions

Review the Address Selection Table carefully before selecting the board address. If you have doubts concerning available addresses in your particular computer, use the FINDBASE utility provided to determine available addresses.

HEX RANGE	USAGE
000-00F	8237 DMA Controller 1
020-021	8259 Interrupt
040-043	8253 Timer
060-06F	8042 Keyboard Controller
070-07F	CMOS RAM, NMI Mask Reg, RT Clock
080-09F	DMA Page Register
0A0-0BF	8259 Slave Interrupt Controller
0C0-0DF	8237 DMA Controller 2
0F0-0F1	Math Coprocessor
0F8-0FF	Math Coprocessor
170-177	Fixed Disk Controller 2
1F0-1F8	Fixed Disk Controller 1
200-207	Game Port
238-23B	Bus Mouse
23C-23F	Alt. Bus Mouse
278-27F	Parallel Printer
2B0-2BF	EGA
2C0-2CF	EGA
2D0-2DF	EGA
2E0-2E7	GPIB (AT)
2E8-2EF	Serial Port
2F8-2FF	Serial Port
300-30F	reserved
310-31F	reserved
320-32F	Hard Disk (XT)
370-377	Floppy Controller 2
378-37F	Parallel Printer
380-38F	SDLC
3A0-3AF	SDLC
3B0-3BB	MDA
3BC-3BF	Parallel Printer
3C0-3CF	VGA EGA
3D0-3DF	CGA
3E8-3EF	Serial Port
3F0-3F7	Floppy Controller 1
3F8-3FF	Serial Port

Table 3-2: Hex Ranges for Standard Computers

IRQ Configuration

This board can be software enabled to generate an IRQ when the A/D FIFO becomes half-full. This IRQ is one method of taking A/D data off the board quickly. See Chapter 4 for more information on the A/D, this IRQ, and taking data off the board.

The selection of the Interrupt (IRQ) to be used is made by selecting one of the IRQ jumpers on the board.

These jumpers are located adjacent to the 104 connector, in three groups. The location of these jumpers is shown in the Option Selection map, as well as in the Setup Program provided with the board.

Place a jumper on the posts corresponding to the IRQ you wish to select. If you do not intend on using the A/D Data FIFO Half Full IRQ, do not install a jumper on any IRQ pins.

DAC Configuration

This board provides two analog outputs. Each analog output is adjusted by output calibration circuitry including a digital potentiometer. The DAC and digital calibration potentiometers are serial devices, with data being entered bit by bit. Although the details of writing bit by bit are described in Chapter 6, the DAC data will be loaded using a software subroutine.

In order to obtain correct analog outputs, the following operations must be performed:

1) The desired output ranges must be selected by means of jumper selections on the board.

   There are two jumpers, labeled DA5 and DB5, respectively. To select the range for channel A, a jumper needs to be installed at DA5. To select the 5 Volt range, the jumper should be in the position away from the 104 connector. To select the 10 Volt range, the jumper should be in the position nearer the 104 connector. To select the range for channel B, a jumper needs to be similarly installed at DB5. The Setup Program and Option Selection Map provide convenient references for setting these options.

2) The correct calibration constants must be loaded into the digital potentiometers that adjust the output circuitry. (This operation is described in Appendix C: Calibration).

A/D Configuration

In order to use the A/D properly, the following operations must be completed:

1) The range of the A/D must be set by selecting jumpers on the board.
2) The mode (single ended or differential) of the A/D input must be selected by jumpers on the board.
3) The appropriate calibration constants should be loaded into the digital calibration potentiometers used with the A/D.

Jumpers		Prog Gain = 0	Prog Gain = 1	Prog Gain = 2	Prog Gain = 3
GNH	Unipolar	0 - 10V	0 - 5V	0 - 2V	0 - 1V
	Bipolar	± 5V	± 2.5V	± 1V	± 0.5
GNL	Unipolar	invalid	0 - 10V	0 - 4V	0 - 2V
	Bipolar	± 10V	± 5V	± 2V	± 1V

Table 3-3: Gain/Range Selection

Range/Mode Selection

Carefully plan which range to select on the board by examining **Table 3-1**. Only the ranges shown in a single row are available simultaneously (via software control). For example, if you want to use both ±5V and ±2V inputs, you must select GNL/Bipolar range; if you're using only ±5V inputs you could select either GNH/Bipolar or GNL/Bipolar (although GNH/Bipolar would be simpler from a software perspective, as the software programmable gain selection defaults to "0", the correct value for this range.)

1) The maximum span of the A/D circuitry is selected by choosing the GNH (high gain) or GNL (low gain) jumpers, located at the right edge of the board. Note that there are two jumpers that must be installed in parallel, one beside the other.

Another pair of jumpers must be installed to determine whether the ranges are unipolar (e.g. 0-10V) or bipolar (+/-5V). These jumpers are located above the gain jumpers at the right edge of the board. The pair must be installed in parallel, one beside the other.

2) The Diff/SE mode of the multiplexer and the A/D circuitry is selected by installing two jumpers.

Two jumpers must be installed to select the mode of the multiplexer (8 channels of differential inputs or 16 channels of single ended inputs). This pair of jumpers is located above and to the right of P1. Both jumpers must be installed, one below the other, in either the "16 CH SE" or "8 CH BAL" positions.

The Single-Ended Mode requires two connections per channel: A ground wire and a signal wire. The differential Mode requires two signal connections: the input to the A/D is the difference in potential between the two signal wires, rather than the amplitude of either or both of them. In Differential mode pickup interference will usually move both signal wires in the same direction, leaving the difference in potential unchanged: this is referred to as common mode rejection (CMR). Differential CMR improves signal to noise characteristics but reduces system input channel count. For best results a ground wire should be supplied to limit the system's Common Mode.

The maximum common mode an input can reject in this manner is defined in the Specifications. If no external ground wire is available and two signal wires are "floating", then two pull-down resistors at the inputs of the PGA onboard will keep the common mode within ±10V.

The gain of the Programmable Gain Amplifier may also be set through software. If not programmed, the default gain is x1.

The board has a capability of a stored different gain for a different channel for A/D inputs 0 through 7.

There are two addresses used in setting up the gain-channel storage: (+04) and (+05). The (+04) address is used for setting up the four lower channels, 0 through 3. The (+05) address is used for setting up the four upper channels, 4 through 7.

NOTE: The second jumper referred to above is a “tattletale” jumper, which informs the Digital circuitry the setting of the analog jumper.

NOTE: The channel-gain programming of channels 0 through 7 is repeated for channels 8 through F.

Loading Calibration Constants

The gain and offset of the signal conditioning circuitry are adjusted by means of digital potentiometers. If constants are not loaded, the potentiometers will be set to the center of their ranges, by default. Therefore, for maximum accuracy, appropriate settings should be entered into them each time the board is powered.

Consult Appendix C: Calibration for information on how to determine and load appropriate settings.

Chapter 4: Analog/Digital Converter Operation

The A/D circuitry on this board forms a very powerful and flexible framework on which you can build a broad array of data acquisition applications. The rest of this chapter is broken down into several sections. The first section is an **Overview**, which will introduce you to the most common modes of using the A/D. This will be followed by a **Breakdown** of the steps recommended to use each of the four modes. Next, a **Step-By-Step** walkthrough is provided. Steps are numbered consistently between the three sections so you can easily flip between them to build an understanding of the process. Following these discussions of the A/D operational modes is some explanation of the various methods of reading the A/D off the board.

Before using any Analog function of the board, make sure you've configured the jumpers as described in Chapter 3. Also, make sure the proper calibration constants have been loaded into the calibration potentiometers using the procedures discussed in Appendix C. Also, Pins 33 through 40 provide the method of connecting triggers and scan-start signals, and proper configuration of these pins is important to success in using each mode. See the detailed mode descriptions for more information.

Overview: A/D Operational Modes

Five modes of A/D operation are available: Software, Burst Mode, Delayed, Triggered, and Timed.

Software: In this mode conversions are started by software command, and are neither synchronized nor repeated. Write to base + 0 takes one channel entry from the Base + 2 and if more than one channel is selected, increments the channel. Primarily used for test and development.

Burst Mode: Takes data at a very fast rate[1] on a single channel - started and stopped by software command.

Delayed: Takes data at a fast rate[2] "oversample" number of times for each channel selected from the base + 2 register, but doesn't start taking data until a single rising edge of the External input pin.

Timed: Takes one scan of data "oversample" number each time the output of Counter 2 goes low. Software setup and started.

External Scan Start: Takes an "oversample" scan of data each time the External Scan Signal goes low. Scan needs to be complete before another scan is started.

[1] 250,000 Samples / Second max

[2]: There is a small inter-channel delay that reduces the max speed when acquiring data from more than one channel.

Breakdown: A/D Operational Modes

The following describes the four modes of operation in more detail:

1. Software:
    - 1.1 Set Channel Scan Limits. Write to Base + 2
    - 1.2 Loop
        - 1.3 Set Gain Code[1]. Write to Base + 4
        - 1.4 Start Conversion by writing to Base + 0
        - 1.5 Wait for FIFO not empty. Read Base + 8, Bit 7 low
        - 1.6 Read Data – Word Read Base + 0
    - 1.7 Until done

2. Burst Mode:
    - 2.1 Set Channel Scan limits - write to Base + 2
    - 2.2 Set Gain Code – Use Word Write to Base + 4
    - 2.3 Start Burst – Use Byte Write 1 to Base + 3
    - 2.4 READ DATA FAST[2]
    - 2.5 Disable Burst – Use Byte Write 0 to Base +3

3. Delayed:
    - 3.1 Set Channel Scan Limits – Write to Base + 2
    - 3.2 Set Gain Code Use Word Write to Base + 4
    - 3.3 Configure Counters 1 and 2 – Base + 14 - 17
    - 3.4 Enable Delay mode – Use Byte Write 10 to Base + 1A
    - 3.5 Configure Oversampling (if desired) (defaults to x1) Byte Write to Base + 1F
    - 3.6 Data started by External Trigger – See Figure 3-1 for connection
    - 3.7 READ DATA FAST[2]
    - 3.8 Stop Data acquisition – Byte Write 20 to Base + 1A

4. Timed:
    - 4.1 Set Channel Scan Limits – Write to Base + 2
    - 4.2 Set Gain Code – Use Word Write to Base + 4
    - 4.3 Configure Counters 1 and 2 – Base + 14 - 17
    - 4.4 Configure Oversambling (if desired) (defaults to x1) Byte Write to Base + 1F
    - 4.5 Enable TIMED Mode – Byte Write 10 to Base + 1A – and Byte Write 40 to Base + 1E
    - 4.6 READ DATA FAST[2]
    - 4.7 Disable Timed Mode – Byte Write 20 to Base + 1A and Byte Write 0 to Base + 1E

5. External Scan Start:
    - 5.1 Connect External Scan Start signal to pins 37 and 38. See Chapter 3. (Power OFF)
    - 5.2 Set Disable Code 1 Write 20 to + 1A and Write 0 to + 1E
    - 5.3 Set Channel (Write to + 02) Start Channel and Stop Channel
    - 5.4 Set Overscan (Write to + 1F) (if desired)
    - 5.5 Configure Counters 1 and 2 – Base + 14 – 17
    - 5.6 Set Gain Code (Write Word + 4 & + 5)
    - 5.7 Set Enable code – Write 10 to +1A
    - 5.8 External Start Signal should be started
    - 5.9 READ DATA FAST[2]
    - 5.10 Set disable code – Byte Write 20 to Base + 1A to stop process

[1]: Software start conversion mode is the most efficient way to achieve different gains/channel. In other modes, timing the gain changes involves obscure operations related to Base + A Read.

[2]: Several methods exist to read the data quickly from the board. Consult the section READ DATA FAST, below.

Step-By-Step: A/D Operational Modes

Now let's describe these modes in detail. All Base + xx offsets are in hexadecimal. Please refer to Chapter 6: Programming for more information on these Base + xx registers.

1. Software Step-by-Step

Step 1.1 Set Channel Scan Limits. Base + 2 contains the Start and End Channel Scan Limit register. The top nybble is the End Channel, the bottom nybble is the Start Channel. In this mode the A/D will take its first conversion on the Start channel. The channel number then increments until it has acquired the End Channel, and starts over at the Start Channel. The current channel can be read at Base + A bits 0 - 2. Any time you write to this register the current channel is set to the Start Channel.

Step 1.2 Loop. In step 1.7 you'll come back to this step.

Step 1.3 Set Gain Code. This optional step allows you to configure a different gain code per channel, allowing for different input ranges on each channel of the board. If all of your channels are being acquired at the same gain, you should swap steps 1.2 and 1.3, so this only needs to be written once. To set the gain code, write to Base + 4 using a WORD WRITE. Refer to **Table 3-1** in Chapter 3 for a quick range/gain reference. Also see Chapter 6.

Step 1.4 Start Conversion. Writing any value to Base + 0 will start a conversion on the current channel. When the conversion is completed the data will be moved into the A/D Data FIFO, and the EMPTY bit in Base + 8 (bit 7 low) will indicate the presence of data in the FIFO.

Step 1.5 Wait for FIFO not empty. By checking the EMPTY bit in Base + 8, determine when the FIFO has received the A/D data. While you are waiting your program can perform other operations, such as DAC outputs, without disturbing the A/D.

Step 1.6 Read Data. Read a 16-bit value from Base + 0 to retrieve the results of the conversion from the A/D Data FIFO. Once you have the data you can store it on disk, display on screen, or whatever else your system requires.

Step 1.7 Until Done. Repeat from 1.2 until you don't need any more data.

2. Burst Mode Step-By-Step

Step 2.1 Set Channel. Burst mode only takes data from one channel. Write the channel you want to take data from to Base + 2. Only the bottom nybble is significant.

Step 2.2 Set Gain. All of the data taken will be at the same gain code / range. Specify the range you want by writing to Base + 4. Refer to **Table 3-1** in Chapter 3 for a quick range/gain reference.

Step 2.3 Start Burst. Write "1" to Base + 3 to start Burst Mode. As soon as the byte is written data will be taken at the 250KHz rate. As each conversion completes the data will be moved into the A/D Data FIFO. If any data is in the FIFO the EMPTY bit (bit 7 LOW for NOT EMPTY) will so indicate, and if the FIFO reaches half-full, the DFH bit (bit 5 = 1 for HALF FULL or MORE) will so indicate when reading from Base + 8.

Step 2.4 READ DATA FAST. The maximum rate, 500KHz, is fast. If you fail to take the data out of the FIFO fast enough, it will fill. If the FIFO reaches full, the burst will pause until some data has been removed from the FIFO to make room for more. Various methods for reading the data are explained in the section READ DATA FAST, below.

Step 2.5 Disable Burst. Whenever you have enough data for your needs, write "0" to Base + 3 to stop the A/D conversions.

3. Delayed Step-By-Step

Step 3.1 Set Channel. Please see Step 1.1, above, for information.

Step 3.2 Set Gain Code. Please see Step 2.2, above, for information.

Step 3.3 Configure Counters 1 and 2, Base + 14 - 17

Step 3.4 Enable Delayed Mode. Use Byte Write 10 to Base +1A.

Step 3.5 Configure Oversampling (if desired) (defaults to x1). Byte Write to Base +1F. in this mode, x1, x2, x8, or x16 samples per channel can be acquired before changing to the next selected channel.

Step 3.6 Data started by External Trigger – see Figure 3.1 for connection information.

Step 3.7 READ DATA FAST. See Step 2.5 for more information.

Step 3.8 Stop Data Acquisition – Byte Write 10 to Base + 1A – and Byte Write 40 to Base +1E.

4. Timed Step-By-Step

Step 4.1 Set Channel. Please see Step 1.1, above, for information.

Step 4.2 Set Gain Code. Please see Step 2.2, above, for information.

Step 4.3 Configure Counters 1+2 using periodic timed scans using the output of Counter 2.

When using periodically timed scans via the output of Counter 1+2, also configure these counters in Mode 2, with a load value appropriate to the timing you want. The input frequency to Counter 1 is 10MHz, the input to Counter 2 is the output of Counter 1. Therefore, the equation to determine the needed load value for the "32-bit" Counter 1+2 is 10,000,000/RateInHertz. For example, if you want scans to start every 15 milliseconds, your equation is 10,000,000 / (1 / (0.015)) = 10,000,000/ 66.67 = 150,000. This load value must then be allocated to the two 16-bit counters which make up the 32-bit Counter 1+2. If the counters accepted fractional load values, the simplest method would be to take the square-root of 150,000. Instead, find two integer factors of 150,000 each smaller than 65,535 and larger than 1. 3 and 50,000 work fine; load Counter 1 with 3 and Counter 2 with 50,000, and your scan rate is set at 66.67Hz.

See Appendix B for more information on loading the counters.

Step 4.4 Configure Oversampling (if desired) (defaults to x1).

Step 4.5 Enable TIMED Mode. Write "40" to Base + 1E and Write "10" to Base + 1A. Setting these two bits will cause the board to initiate one scan of A/D data each timeout occurring on Counter 2. It is important that the Start trigger does not occur during a scan. When using the counter you must have a long enough timout period loaded in Counters 1+2 that the board has time to complete "Oversample" number of conversions on as many channels as you have selected at Base Counter 1+2. Once step 4.5 is performed and as soon as the counters have timed out, a scan will start.

Step 4.6 READ DATA FAST.

5. External Scan Start

Step 5.1 Connect External Scan Start Signal to pin 37 (GND) and pin 38 (HOT) – see Chapter 3. (NOT Running).

Step 5.2 Set Disable Code 1 Write 20 to Base + 1A AND Write 0 to Base +1E.

Step 5.3 Set Channel(s) Write to Base +2 for Starting Channel and Ending Channel.

Step 5.4 Configure Oversample Mode if Desired. Write 03 to Base + 1F for 2x, Write 00 for 8x, Write 02 for 16x. Writing 01 to Base +1D selects NO Oversampling.

Step 5.5 Configure Counters 1 and 2 – Base +14 - 17

Step 5.6 Set Gain Code (Write Word to Base +4 – Base +5).
Example: Binary 11100100 HEX E4 produces a gain of x1 for Channel 0, a gain of x2 for Channel 1, a gain of x5 for Channel 2, and a gain of x10 for Channel 3. Writing HEX E4E4 extends the gain pattern from Channel 0 through Channel 7. The gain settings for Channels 0 through 7 are random access. The gain settings for Channels 8 through F are the settings for Channels 0 through 7.

Step 5.7 Set enable code – Write 10 to +1A

Step 5.8 External Start Signal may now be started

Step 5.9 READ DATA FAST

Step 5.10 Stop Data Acquisition – Byte Write 20 to Base +1A – and Write 0 to Base +1E

READ DATA FAST

The board contains a 4096-byte A/D Data FIFO configured as a 2048-sample FIFO. The FIFO provides two status bits useful when taking data from the board. First, a Data FIFO Half (DFH) bit is available. This bit indicates the FIFO is now half full. In addition this bit, if enabled, will generate an IRQ. Second, a EMPTY bit is provided which indicates the FIFO contains no data. Conversely, if there is any data in the FIFO the EMPTY bit will not be true.

Using combinations of these two bits and the DFH IRQ allows several fast and efficient methods of taking the data from the FIFO.

Perhaps the simplest method of taking data is to read one sample at a time using a 16-bit read of Base + 0. In this mode, you determine when to read the sample by polling the EMPTY bit. When the FIFO is *not* empty, you read one sample. This method is simple to program, but not very resource efficient; reading the status register and the FIFO for every sample can result in a very busy computer. Despite the drawbacks, this method is often the best when data rates are very slow, or the only goal is to quickly test operation of the board..

Write value 10 to Base + 0C to Enable IRQ operation. Write value 00 to Base + 0C to Disable IRQ Operation

The fastest method is to INSW 1024 samples from Base + 0 every time the DFH bit indicates the FIFO is half-full. Barely slower is to enable the IRQ to generate an IRQ each time DFH is true, and have an ISR respond to the IRQ by using INSW to read 1024 samples.

NOTE: Since this is an 8-bit PC/104 bus board, and since the A/D Data is 16-bits wide, two sequential reads are necessary to retrieve the 16-bits of the converted Data. However, by assuming the data can be read as a 16-bit value and issuing a word read instead of two byte reads the bus controller on the computer will optimize the operation and reduce the overhead associated with reading the data.

Chapter 5: Digital Input/Output

The use of the Digital Input/Output function is completely controlled by application software.

There are two independent Digital I/O ports. Each port consists of 8 bits. The external connections are via connector P2 and are listed in Chapter 7.

Port 0 is located at Base Address + 10 and reads/controls pins DIO0 through DIO7
Port 1 is located at Base Address + 11 and reads/controls pins DIO8 through DIO15.

Both ports default to input (read) mode. Each remains in that mode until written to. When written to, *that* port will switch to and remain in output (write) mode.

Both ports may be reset to the input mode by writing anything to Base Address + 19.

Please refer to the appropriate section of Chapter 6 for more information.

Chapter 6: Programming

The following table shows the register definition map. All offsets are in hexadecimal.

Offset	Write Function	Read Function
0	Start A/D Conversion	Read A/D Data FIFO
1	A/D Data FIFO Reset	Read A/D Data FIFO
2	A/D Channel Scan Control	
3	A/D Burst Mode Control	
4	A/D Software Gain Select: Low Channels 0 through 3	
5	A/D Software Gain Select: High Channels 4 through 7 (Note: use Word Write for 4 and 5)	
6		
7		Reset Software A/D Gain to ONE
8		Jumper Configuration / Status
9	DAC Outputs	Internal Status
A	EEPROM Data Write	EEPROM Read / A/D Channel Read
B	Calibration Data Write	
C	Enable IRQ or Disable !RQ	
D		
E		
F		
10	Digital Outputs 0 - 7	Digital I/O 0 - 7
11	Digital Outputs 8 - 15	Digital I/O 8 - 15
12		
13		
14	Program Counter 0 (NOT USED)	
15	Program Counter 1	Read Counter 1
16	Program Counter 2	Read Counter 2
17	Set-up Counter Control Register	n/a
18		
19	Reset Digital I/O to Input Mode	
1A	Start- Stop Triggered Mode and setup External Trigger Start	
1B		
1C		
1D		Master Reset
1E	Prestart Setup for Timed, Delayed and Scan Start Modes	
1F	Configure OVERSAMPLING	

Table 6-1: Register Definition Map

Addresses left blank in the table above are reserved for factory use, and should not be accessed by user software programs.

Base Address + 0 - Start A/D Conversion / Read A/D Data FIFO

Writing any value to this address causes the A/D to make one data acquisition and load it into the FIFO. This is known as a "Software Start Conversion" command. The channel acquired is the currently selected one in the A/D Channel Scan Control register. Please refer to the discussion in Base + 2 for information on selecting the channels.

Reading two bytes consecutively from this address will retrieve the LSByte and MSByte of the data from the FIFO. It is very important to avoid getting out of sync with the data, so it is recommended you use 16-bit (word) reads instead of consecutive byte reads.

Reading a (word) value from this base address will automatically grab one conversion's data from the FIFO (both bytes). It is useful to always use 16-bit reads from this address to ensure the integrity of FIFO data.

The data is returned in 16 binary coded bits, where the hexadecimal value FFFF corresponds to maximum input voltage, and hexadecimal 0000 corresponds to minimum input voltage, based on the jumper and software selected A/D input range. Refer to Chapter 3 for information on range selection, and Base + 4 for more information on software selectable gains. Please note in bipolar ranges the value 0000 corresponds to maximum-negative-voltage.

For example, if the board is configured for a ±2V range, FFFF is 2V, 0000 is -2V, and 8000 is 0V. Consult the sample code on the provided CD for examples of algorithms to convert the hexadecimal return value to voltage for any arbitrary range.

Base Address + 1 - A/D Data FIFO Reset

Writing any value to this address resets the FIFO. This will empty the FIFO of any data that remains. It is a good idea to reset the data FIFO before you start any data acquisition process to ensure you're in good sync with the data that will be acquired. To make sure the FIFO reset will leave the FIFO empty, make sure the A/D isn't currently acquiring data before issuing this command. Chapter 4 and the descriptions of Base +3, Base +1C, Base +1D, Base +1E have more information on various ways acquisition could be happening.

Base Address + 2 - A/D Channel Scan Control (write only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EMA3*	EMA2	EMA1	EMA0	SMA3*	SMA2	SMA1	SMA0

This register controls the channel scan limits for the A/D input multiplexer. Specify a channel number from 0 to 15 (0-F) in each nybble for the start and end limit of the auto-incrementing mux channel number.

The "S" entries refer to the starting address for the scan and the "E" entries refer to the ending address for the scan.

The scan is performed from the starting to the ending addresses, inclusively. No channels are omitted.

Write same address in both nybbles to dwell on one channel.

When the board is operated in the Differential mode, (E/S)MA3 is ignored.
When the board is operated in the Single Ended mode, (E/S)MA3 is the most significant bit of the input address.

Notes:
The selection of Single ended or Differential inputs must be made by properly installing jumpers on the board. Please refer to Chapter 3, or the Setup program, for details.
Since the MUX addresses do not advance during the burst mode, the ending address (EMA3-EMA0) is ignored, and all conversions occur on the channel specified in SMA3-SMA0.

Base Address + 3 - A/D Burst Mode Control
This register enables burst mode.

Write 01 to start, 00 to stop. Set desired channel using Base + 2 above.

Burst Mode operation acquires data on **one** previously selected channel at the maximum speed of the A/D, approximately 4 microseconds per conversion (250KHz). This data is stored in the FIFO. Conversions pause when the FIFO is full. Conversions resume when a FIFO read command removes data from the FIFO (Base+0), thereby removing the FIFO full state. Alternatively, operation can be stopped short of a full FIFO if a Burst Mode Off command is sent (write 00 here).

Usually you'll be using the FIFO Half-full IRQ or polling its status bit to time when to take data out of the FIFO, ensuring it never reaches full. See Base Address + 8 for more information.

Base Address + 4 - Software Programmable Gain Select 0-7 (Word Write)

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Ch7 and 15 Gain Code		Ch6 and 14 Gain Code		Ch5 and 13 Gain Code		Ch4 and 12 Gain Code		Ch3 and 11 Gain Code		Ch2 and 10 Gain Code		Ch1 and 9 Gain Code		Ch0 and 8 Gain Code	

The full scale range of the board depends on the settings of the Bipolar/Unipolar and GNH/GNL jumpers, as described in Chapter 3.

A gain code of "0" provides an amplifier gain of x1, "1" provides a gain of x2, "2" provides a gain of x5, while "3" provides a gain of x10. To quickly set all channels to the same gain, multiply the gain code by 5555 hex, then write the result to Base + 4.

Please refer to **Table 3-1** in Chapter 3 of this manual for a detailed breakdown of the effect of these bits. Each time the A/D changes to a new channel, the A/D gain will change according to the gain code configured here.

Base Address + 8 - Jumper Configuration / Status (Read Only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EMPTY	DFF	DFH	DA5V	DB5V	GNH	BIPOLAR	SE

Bit 0 = 0 8 AD Channels enabled - = 1 16 Channels enabled

Bit 1 = 0 Bipolar Operation - = 1 Unipolar Operation

Bit 2 = 0 Full Gain - = 1 Gain divided by 2

Bit 3 = 0 Digital Bit 0 = 0 to +10V - = 1 = 0 to +5V

Bit 4 = 0 Digital Bit 1 = 0 to +10V - = 1 = 0 to +5V

Bit 5 = 0 FIFO is less than half full - = 1 FIFO is half full

Bit 6 = 0 FIF0 is not full - = 1 FIFO is full

Bit 7 = 0 FIFO is not empty - = 1 FIFO is empty

Base Address + 9 - DAC Outputs (Write)

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Write	Data	X	X	X	X	X	Load	SClock

DAC data loading (Write)

The Software Master CD contains sample programs demonstrating the use of the DAC in a variety of languages, including a "driverlet" which encapsulates the complexities of the process. Using this "driverlet" is as simple as passing the DAC number and voltage you want to a function. It is highly recommended that you use the provided source code as a basis for your own programs.

This chip is a dual 12 bit DAC with a serial interface. In order to load a value into the DAC and have it converted to an analog output, 18 successive writes must be made to base address + 9. These writes are used to create the signals needed to communicate to the DAC.

The DACs are jumper configured as 0-5 or 0-10V outputs. Writing a code of 0 results in 0 Volts, writing FFF results in either 5V or 10V as per range selection.

As an example to load 7FF (midrange) into DAC0, the following writes are required. (The value 7FF is 011111111111 in binary - a value of either 2.5V or 5.0V, depending on range selected.)

Write	Value	Description
1	1xxxxx01=81	Start Bit. Always write 81 as the 1st byte
2	**0**xxxxx11=**0**3	DAC selection. Write 03 for DAC0, 83 for DAC1
3	**1**xxxxx11=**8**3	DAC selection. Write 83 for DAC0, 03 for DAC1
4	0xxxxx11=03	Spacer Bit - Always write 03 as the 4th byte
5	**0**xxxxx11=**0**3	MSB of data. Bit 7 should be 1 or 0 based on the D11 bit of data to be output. Always set D1 and D0 to "1"
6	**1**xxxxx11=**8**3	Bit 7 should be bit D10 of data. Always set D1 and D0 to "1"
7	**1**xxxxx11=**8**3	Bit 7 should be bit D9 of data. Always set D1 and D0 to "1"
8	**1**xxxxx11=**8**3	Bit 7 should be bit D8 of data. Always set D1 and D0 to "1"
9	**1**xxxxx11=**8**3	Bit 7 should be bit D7 of data. Always set D1 and D0 to "1"
10	**1**xxxxx11=**8**3	Bit 7 should be bit D6 of data. Always set D1 and D0 to "1"
11	**1**xxxxx11=**8**3	Bit 7 should be bit D5 of data. Always set D1 and D0 to "1"
12	**1**xxxxx11=**8**3	Bit 7 should be bit D4 of data. Always set D1 and D0 to "1"
13	**1**xxxxx11=**8**3	Bit 7 should be bit D3 of data. Always set D1 and D0 to "1"
14	**1**xxxxx11=**8**3	Bit 7 should be bit D2 of data. Always set D1 and D0 to "1"
15	**1**xxxxx11=**8**3	Bit 7 should be bit D1 of data. Always set D1 and D0 to "1"
16	**1**xxxxx11=**8**3	LSB of data. Bit 7 should be bit D0 of data. Always set D1 and D0 to "1"
17	0xxxxx00=00	Update DAC. Always write 00 as the 17th byte
18	0xxxxx10=02	End. Always write 02 as the 18th byte.

As you can see from the table, the only writes that can vary from one output to the next are the DAC channel number in bytes 2 and 3, and the data in bytes 5 through 16. The only valid values for these bytes are 03 or 83. Byte 1 must always be 81; Byte 4 must always be 03; Byte 17 must always be 00; and Byte 18 must always be 02. For ease of reference the bits which you can change are typeset in **bold**.

The behavior of the DAC is undefined if these guidelines are not followed.

Base +9 Internal Status (Read)

Bits 0 – 3 Not Used

Bit 4 IRQ On = 1 IRQ Off = 0

Bit 5 Low Gain Bit of selected (in use) Channel gain

Bit 6 High Gain Bit of selected (in use) Channel gain

Bit 7 Not used

Internal Status Register (Base + 9 Read) provides internal diagnostic data, not needed for operational use.

Base Address + A - EEPROM Data Write and Read (bit 7)

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit1	Bit 0
Write	Data	X	X	X	X	X	X	SClock
Read	Data	X	X	X	X	X	X	X

The EEPROM is intended to hold calibration data for the A/D Gain and Offset correction, and the Gain Correction data for both DACs. Calibration is only needed for GNL/GNH and BIP/UNI jumper selected ranges, so a total of 4 A/D Input calibration data pairs are used. Consult the provided calibration program or sample code for information on the locations in the EEPROM used to store the calibration data.

Although the EEPROM is intended to contain calibration data, it is unlikely your program will need to keep calibration data for the ranges you are not going to use. In this case you can use those locations in the EEPROM for your own purposes.

The Software Master CD contains sample programs demonstrating the use of the EEPROM in a variety of languages, including a “driverlet” which encapsulates the complexities of the process. Using this “driverlet” is as simple as passing the address and data in the EEPROM you wish to write, or the address from which to read, to our functions. It is highly recommended that you use the provided source code as a basis for your own programs.

Writing to the EEPROM (Base + A)

In order to write to the EEPROM, a start bit must be transmitted, then the Write opcode (2 bits, 01) followed by the address location of the data to be loaded into the EEPROM (6 bits, MSB first), followed by the data (16bits, MSB first). Then the transmission is ended with 0. Therefore, to write a value of aa55 to location 5, you would perform the following 26 writes:

Write	Value	Description
1	1xxxxxx1=81	Start Bit. Always write 81 as the 1st byte
2	0xxxxxx1=01	opcode bit 1, always write 01 as 2nd byte for EEPROM writing
3	1xxxxxx1=81	opcode bit 0, always write 81 as 3rd byte for EEPROM writing
4	**0**xxxxxx1=**0**1	MSB of address. Bit 7 should be 1 or 0 based on the D5 bit of address to be written. Always set D0 to "1"
5	**0**xxxxxx1=**0**1	Bit 7 should be bit D4 of address. Always set D0 to "1"
6	**0**xxxxxx1=**0**1	Bit 7 should be bit D3 of address. Always set D0 to "1"
7	**1**xxxxxx1=**8**1	Bit 7 should be bit D2 of address. Always set D0 to "1"
8	**0**xxxxxx1=**0**1	Bit 7 should be bit D1 of address. Always set D0 to "1"
9	**1**xxxxxx1=**8**1	LSB of address. Bit 7 should be 1 or 0 based on the D0 bit of address to be written. Always set D0 to "1"
10	**1**xxxxxx1=**8**1	MSB of data. Bit 7 should be 1 or 0 based on the D15 bit of address to be written. Always set D0 to "1"
11	**0**xxxxxx1=**0**1	Bit 7 should be bit D14 of data. Alwways set D0 to "1"
12	**1**xxxxxx1=**8**1	Bit 7 should be bit D13 of data. Always set D0 to "1"
13	**0**xxxxxx1=**0**1	Bit 7 should be bit D12 of data. Always set D0 to "1"
14	**1**xxxxxx1=**8**1	Bit 7 should be bit D11 of data. Always set D0 to "1"
15	**0**xxxxxx1=**0**1	Bit 7 should be bit D10 of data. Always set D0 to "1"
16	**1**xxxxxx1=**8**1	Bit 7 should be bit D9 of data. Always set D0 to "1"
17	**0**xxxxxx1=**0**1	Bit 7 should be bit D8 of data. Always set D0 to "1"
18	**0**xxxxxx1=**0**1	Bit 7 should be bit D7 of data. Always set D0 to "1"
19	**1**xxxxxx1=**8**1	Bit 7 should be bit D6 of data. Always set D0 to "1"
20	**0**xxxxxx1=**0**1	Bit 7 should be bit D5 of data. Always set D0 to "1"
21	**1**xxxxxx1=**8**1	Bit 7 should be bit D4 of data. Always set D0 to "1"
22	**0**xxxxxx1=**0**1	Bit 7 should be bit D3 of data. Always set D0 to "1"
23	**1**xxxxxx1=**8**1	Bit 7 should be bit D2 of data. Always set D0 to "1"
24	**0**xxxxxx1=**0**1	Bit 7 should be bit D1 of data. Always set D0 to "1"
25	**1**xxxxxx1=**8**1	LSB of data. Bit 7 should be 1 or 0 based on the D0 bit of data written. Always set D0 to "1"
26	0xxxxxx0=00	End. Always write 00 as the 26th byte

For ease of reference the bits which can change are typeset in **bold**.

Please note, it is not possible to write to the EEPROM until an EEPROM WRITE ENABLE (EWREN) sequence has been written to Base + A. The EWREN sequence consists of the following bytes, in order: 81, 01, 01, 81, 81, 01, 01, 01, 01, 00.

Once the EWREN sequence has been written it is possible to write to the EEPROM as desired. If you wish to subsequently disable writes to the EEPROM, a disable sequence of bytes may be written to Base + A as follows: 81, 01, 01, 01, 01, 01, 01, 01, 01, 00.

Reading from the EEPROM (Base + A)

Similarly, reading a word takes 10 writes (start bit, read opcode (2 bits -1,0) and the address (6 bits, MSB first)), followed by 16 reads to acquire the data from the eeprom, followed by one write to terminate communication with the eeprom. Therefore, to read from address 4, the following reads and writes are performed:

Write	Value	Description
1	1xxxxxx1=81	Start Bit. Always write 81 as the 1st byte.
2	1xxxxxx1=81	Opcode Bit 1. Always write 81 as the 2nd byte for reading
3	0xxxxxx1=01	Opcode Bit 0. Always write 01 as the 3rd byte for reading
4	**0**xxxxxx1=**0**1	MSB of address. Bit 7 should be 1 or 0 based on the D5 bit of address in the EEPROM to be Read. Always set D0 to "1"
5	**0**xxxxxx1=**0**1	Bit 7 should be bit D4 of address. Always set D0 to "1"
6	**0**xxxxxx1=**0**1	Bit 7 should be bit D3 of address. Always set D0 to "1"
7	**1**xxxxxx1=**8**1	Bit 7 should be bit D2 of address. Always set D0 to "1"
8	**0**xxxxxx1=**0**1	Bit 7 should be bit D1 of address. Always set D0 to "1"
9	**0**xxxxxx1=**0**1	LSB of address. Bit 7 should be 1 or 0 based on the D0 bit of address to be read. Always set D0 to "1"
Read 1		Bit D7 of this Read returns the Most Significant Bit (D15) of the 16-bit data stored at the address specified in writes 4 through 9.
Read 2		D7 contains bit D14 from the word at the specified address
Read 3		D7 contains bit D13 from the word at the specified address
Read 4		D7 contains bit D12 from the word at the specified address
Read 5		D7 contains bit D11 from the word at the specified address
Read 6		D7 contains bit D10 from the word at the specified address
Read 7		D7 contains bit D9 from the word at the specified address
Read 8		D7 contains bit D8 from the word at the specified address
Read 9		D7 contains bit D7 from the word at the specified address
Read 10		D7 contains bit D6 from the word at the specified address
Read 11		D7 contains bit D5 from the word at the specified address
Read 12		D7 contains bit D4 from the word at the specified address
Read 13		D7 contains bit D3 from the word at the specified address
Read 14		D7 contains bit D2 from the word at the specified address
Read 15		D7 contains bit D1 from the word at the specified address
Read 16		D7 contains bit D0 from the word at the specified address
Write	0xxxxxx0=00	End. Always Write 00 as the last step

For ease of reference the bits which can change are typeset in **bold**.

Base Address + A - A/D Channel Read (Bits 3 - 0)

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Read	X	X	X	X	MA3*	MA2	MA1	MA0

Bits 3 through 0 readback the currently in-use channel on the board. In auto-incrementing mux modes this data can be used to confirm proper operation of the device.

MA3* In Single Ended Mode, this is the most significant bit of the A/D Channel number
In Differential Mode, this bit is ignored
MA2, MA1, MA0 - Current Channel Number

Base Address + B - Calibration Data Write

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Data	X	X	X	X	X	X	SClock

The board contains 4 digital potentiometers used to calibrate the device. The four calibration corrections provided are: A/D Offset (00), A/D Gain (01), Gain for DAC0 (10), and DAC1 (11). These four corrections are internally addressed 0, 1, 2, and 3. Each calibration correction consists of a value from 0 - 255 (8 bits) corresponding to the internal value of the digital potentiometer.

A/D Offset corresponds to the “B” in a Y=mX+B equation. A/D Gain is the “m” term of the same equation. DAC Gain represents “m” in the equation Y=mX. The nature of the DAC circuitry eliminates the need to provide offset (B) calibration.

The value you load into these calibration potentiometers you normally read from the EEPROM and write here during program initialization, and only needs to be written once per power-on cycle.

The Software Master CD contains sample programs demonstrating the use of the calibration potentiometers in a variety of languages, including a "driverlet" which encapsulates the complexities of the process. Using this "driverlet" is as simple as passing the M and B correction values (for A/D calibration) or the DAC number and M correction values (for D/A calibration) to our functions. It is highly recommended that you use the provided source code as a basis for your own programs.

To load one of the calibration correction values you must write the address of the correction you wish to load, the 8-bit value you wish to load, and an End byte.

Similar to the operation of the DAC and EEPROM, the calibration correction values are loaded serially into the digital potentiometers in the board. The steps are outlined in the table below.

Therefore, to load a value of 4F into the A/D Gain potentiometer, the following writes are performed:

Write	Value	Description
1	**0**xxxxxx1=**0**1	These two bits are the address of the potentiometer to write. 00 is A/D offset, 01 is gain, 10 is DAC0 gain, 11 is DAC1 gain. Write 1 is the MSB, Write 2 is the LSB
2	**1**xxxxxx1=**8**1	
3	**0**xxxxxx1=**0**1	MSB of data. Bit D7 of Write 3 should be the D7 bit of the calibration correction value you are loading.
4	**1**xxxxxx1=**8**1	D7 should be bit D6 of the calibration value
5	**0**xxxxxx1=**0**1	D7 should be bit D5 of the calibration value
6	**0**xxxxxx1=**0**1	D7 should be bit D4 of the calibration value
7	**1**xxxxxx1=**8**1	D7 should be bit D3 of the calibration value
8	**1**xxxxxx1=**8**1	D7 should be bit D2 of the calibration value
9	**1**xxxxxx1=**8**1	D7 should be bit D1 of the calibration value
10	**1**xxxxxx1=**8**1	D7 should be bit D0 of the calibration value
11	0xxxxxx1=00	End. Always Write 00 as the last step

For ease of reference the bits which can change are typeset in **bold**.

Base Address + C - Interrupt Enable (Write Only)
The selection of the IRQ used to transmit the interrupt is made by jumper selection on the board. This selection is described in Chapter 3.

If enabled, an IRQ occurs when the data FIFO reaches half full, giving you time to take the data out of the FIFO before it reaches full (and subsequently pauses the A/D conversions).

Writing 10 hex enables IRQs. Writing 00 disables IRQs.

Base Address + 10 - Digital I/O Bits 0 - 7

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Read / Write	DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1	DIO0

Reading from Base + 10 will return the digital data available on pins DIO0-7.

Writing a value to Base + 10 will configure the bits as outputs, and output the value to the pins.

Once the bits have been configured as outputs, subsequent reads return the most recently written value, not the state of the input pins.

You may reset all DIOs (DIO0-15) to the read function by writing to Base + 19.

Base Address + 11 - Digital I/O Bits 8 - 15

	Bit 7	Bit 6	Bit5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Read / Write	DIO15	DIO14	DIO13	DIO12	DIO11	DIO10	DIO9	DIO8

Reading from Base + 11 will return the digital data available on pins DIO8-15.

Writing a value to Base + 11 will configure the bits as outputs, and output the value to the pins.

Once the bits have been configured as outputs, subsequent reads return the most recently written value, not the state of the input pins.

You can reset all DIOs (DIO0-15) to the read function by writing to Base + 19.

Base Address + 14 through Base Address + 17 - Counter Programming
For detailed information on programming the 8254 Counter Timer device, please refer to Appendix B.

When using TIMED acquisition, it is important that the load value in Counter 1 and 2 be sufficient to give the A/D circuit time to finish acquiring all the channels specified in Base + 2. I.e., if you are acquiring 8 channels of data, you need at least (8*2.5 Sec=20) 20 microseconds between Counter 1 and 2 timeouts. Please refer to Appendix B for information on programming the 8254 Counter Timer chip.

Data acquisition is continuous. When the FIFO is filled, data storage pauses. It resumes when data is read from the FIFO, thereby changing its FULL state.

It is expected that, when operating in this mode, the application program will be reading data from the FIFO as it is being written by the A/D, preventing the FIFO from filling during data acquisition. This is normally accomplished by reading data equal to half of the FIFO capacity whenever the FIFO half full flag is used to generate an interrupt. See READ DATA FAST in Chapter 4 for more information on various ways to get data from the board.

NOTE: Program Counters 1 and 2 before using Base +1E and 1A to start Data Acquisition.

Base Address + 19 - Reset Digital I/O to Input Mode (Write)
Any write to this address results in both Digital I/O Blocks being put into the input (read) mode.

Refer to Base + 10 and Base + 11 for more information.

Base Address + 1A - Write

Writing 10 to this address enables ONE bit of TWO needed to start a Counter controlled Data Acquisition.

Writing 40 to address 1E enables the OTHER bit needed to start a counter controlled Data Acquisition.

Always program Counters 1 and 2 before enabling Either of these bits.

Base + 1A can only be Set or Reset by Software.

Base Address + 1E - Enable Data Acquisition (Write Only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
X	Enable	X	X	X	X	X	X

After 10 has been written to base + 1A (See above), Write 40 to base +1E to Start TIMED Data Acquisition.

NOTE: Counters 1 and 2 should be Programmed BEFORE enabling Bits 1A and 1E.

NOTE: An external trigger may start a Counter controlled data acquisition, enabling Base +1E.

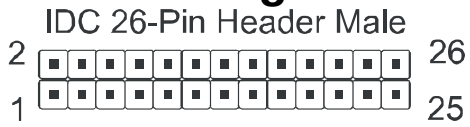
Base Address + 1F – Configure Oversampling (Write)

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Write	X	X	X	X	X	X	D1	D0

The Board is capable of performing multiple data acquisitions of a single input very quickly. The board uses this ability to perform a unique mode of operation, Oversampling. Oversampling is a technique wherein the board will convert the current channel more than one time per start-conversion signal. Configurations of 2X, 8X, and 16X Oversamples are available. Write 03 for 2X, Write 00 for 8X, Write 02 for 16X, or Write 01 for the normal one conversion per start-conversion signal.

Oversampling has no effect on Software Mode, nor on Burst Mode operation.

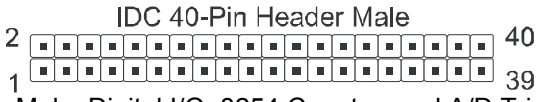
Chapter 7: Connector Pin Assignments



Connector P1, 26-pin IDC header male, Analog Inputs (MUX), DAC Outputs

16-Channel Single-Ended			
Pin	Function	Pin	Function
1	A/D Ch 0 Input	2	A/D Ch 8 Input
3	Ground	4	A/D Ch 9 Input
5	A/D Ch1 Input	6	Ground
7	A/D Ch 2 Input	8	A/D Ch 10 Input
9	Ground	10	A/D Ch 11 Input
11	A/D Ch 3 Input	12	Ground
13	A/D Ch 4 Input	14	A/D Ch 12 Input
15	Ground	16	A/D Ch 13 Input
17	A/D Ch 5 Input	18	Ground
19	A/D Ch 6 Input	20	A/D Ch 14 Input
21	Ground	22	A/D Ch 15 Input
23	A/D Ch 7 Input	24	Ground
25	DAC 0 Output	26	DAC 1 Output

8-Channel Differential			
Pin	Function	Pin	Function
1	A/D Ch 0+ Input	2	A/D Ch 0- Input
3	Ground	4	A/D Ch 1- Input
5	A/D Ch1+ Input	6	Ground
7	A/D Ch 2+ Input	8	A/D Ch 2- Input
9	Ground	10	A/D Ch 3- Input
11	A/D Ch 3+ Input	12	Ground
13	A/D Ch 4+ Input	14	A/D Ch 4- Input
15	Ground	16	A/D Ch 5- Input
17	A/D Ch 5+ Input	18	Ground
19	A/D Ch 6+ Input	20	A/D Ch 6- Input
21	Ground	22	A/D Ch 7- Input
23	A/D Ch 7+ Input	24	Ground
25	DAC 0 Output	26	DAC 1 Output



Connector P2, 40-pin IDC Header Male, Digital I/O, 8254 Counter, and A/D Trigger Input

Pin	Function	Pin	Function
1	DIO 0	2	Ground
3	DIO 1	4	Ground
5	DIO 2	6	Ground
7	DIO 3	8	Ground
9	DIO 4	10	Ground
11	DIO 5	12	Ground
13	DIO 6	14	Ground
15	DIO 7	16	Ground
17	DIO 8	18	Ground
19	DIO 9	20	Ground
21	DIO 10	22	Ground
23	DIO 11	24	Ground
25	DIO 12	26	Ground
27	DIO 13	28	Ground
29	DIO 14	30	Ground
31	DIO 15	32	Ground
33	Reserved	34	Ground
35	Ground	36	ScanEnable
37	Ground	38	ScanStart
39	Reserved	40	Ctr 2 Out

The connections for the Digital Input/Output Signals are provided on the odd numbered pins, from 1 to 31.

The Digital I/O are configured in blocks of 8, as described elsewhere in this manual.

The ScanEnable input (pin 36) may be used to provide an external trigger source for the A/D function. If this option is used, the board must be configured appropriately, as described in Chapter 3.

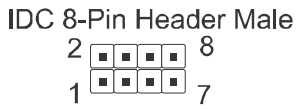
The other connections involve the counter.

Counters 1 and 2 are concatenated internal to the board.

For normal counter driven A/D timing operation, please connect Ctr2 Out (pin 40) to pin 38. (Timed Scan Mode)

See Chapter 4 and 6 for more information on A/D Timing Modes.

See Appendix B for details on programming the counter circuit.



Connector P3, External Power, 8-pin Male IDC Header

Pin	Signal	Pin	Signal
1	Ground	2	Ground
3	Ground	4	+12V
5	Ground	6	-12V
7	Ground	8	Ground

The board can take ±12V power from the PC/104 bus, an external source, or an optional DC/DC converter. If the DC/DC converter is installed on the board, the two jumpers next to P3 should not be installed, and their posts may be missing. If no DC/DC converter is present, the jumpers should be installed in their right positions to take ±12V power from the PC/104 bus, or the left positions (toward the edge of the board) to take ±12V power from an external source, provided via P3.

Appendix A: Technical Specifications

A/D Inputs

Sampling rate:	250KHz (in Burst mode, single channel) 250KHz (in Scan mode, 1 to 16 channels)
A/D FIFO:	2048 16-bit wide samples (Requires two 8 bit reads) FIFO upgrades of 4K, 8K, 16K, 32K samples available
Accuracy:	Automatic Calibration. Offset and Gain Calibration Values stored in EEPROM individually for each range
Ranges:	Software Programmable Gain Channel by Channel during Scan
Selections of:	Unipolar/Bipolar, Single Ended/Differential by jumper ±0.5V, ±1V, ±2V, ±2.5V, ±5V, ±10V 0-1V, 0-2V, 0-4V, 0-5V, 0-10V
Number of Channels:	16 single-ended or 8 differential
Resolution:	16-bit, successive approximation
Input Impedance:	1 Megohm
Overvoltage protection:	±40V
Common Mode Rejection:	75-84 dB depending on instrumentation amplifier gain
Integral Nonlinearity:	±4 LSB typical
Conversion Start Modes:	Software Command Single conversion on selected channel Single channel Burst Continuous Scan of pre-selected Sequential channels External Hardware Trigger Enable Continuous Scan of pre-selected Sequential channels
Oversampling Modes:	2 samples, 8 samples, 16 samples (software selectable)

Counter/Timers

Type	82C54
A/D Pacer clock	16 or 32-bit
Clock Frequency	10MHz

D/A Analog Outputs

Number of Channels:	Two
Resolution:	12-bit
Ranges:	0-5V, 0-10V via jumper selection
Safety Feature:	Automatically set to zero on power up
Accuracy:	Automatic Gain Calibration. Gain Values stored in EEPROM individually for each range and channel
Nonlinearity:	±0.2 LSB typical
Update Rate:	10µs (5KHz)
Settling time:	8µs
Output current:	5mA source

Digital I/O

Number of I/O		16
Programmable as:		Inputs or Outputs in groups of 8
Inputs		
Logic low:		0.0V min, 0.8V max
Logic high:		2.0V min, 5.0V max
current:		±1µA max
Outputs		
Logic low:		0.0V min, 0.55V max
Logic high:		2.4V min, 5.0V max
current:	Logic low:	24mA max sink
	Logic high:	24mA max source
Update rate:		up to 1 MHz
Pull Up Resistor		10K each I/O line

General

Power required:	+5VDC - 100mA typical ±12VDC - 50mA typical each With DC/DC Converter, +5V@ 250mA typical
Operating Temperature:	0 to +70 C
Storage Temperature:	-50 to +120 C
Humidity:	5% to 90% RH, non-condensing
Size and Bus Type:	PC/104 compliant, 8-Bit PC/104 with 16-Bit pass-through connectors and extended IRQ's
I/O Connectors:	26 and 40 Pin right-angle header with 0.1" spacing
Cable Accessories:	Cable Part Numbers C104-26F-X and C104-40F-X (X = length in inches)
Screw Terminal Boards:	Part Numbers STB-26 and STB-40, also 104-STA-16E

Appendix B: 82C54 Counter Timer Operation

The board contains one type 8254 programmable counter/timer. The 8254 is a flexible but powerful device that consists of three independent, 16-bit, presettable down counters. Each counter can be programmed to any count between 2 and 65,535 in binary format, depending on the mode chosen.

On the board these two counters are designated Counter 1 and Counter 2.

Counters 1 and 2 are concatenated by the board to form a single 32-bit counter. The input of the counter 1 is fixed at 10MHz. Counter 2's output is available at connector P2, Pin 40. Counters 1 and 2's gates are enabled via software at Base + 1E.

OPERATIONAL MODES

The 8254 modes of operation are described in the following paragraphs to familiarize you with the versatility and power of this device. For those interested in more detailed information, a full description of the 8254 programmable interval timer can be found in the Intel (or equivalent manufacturers) data sheets. The following conventions apply for use in describing operation of the 8254:

Clock:	A positive pulse into the counter's clock input.
Trigger:	A rising edge input to the counter's gate input.
Counter Loading:	Programming of a binary count into the counter.

Mode 0: Pulse on Terminal Count

After the counter is loaded, the output is set low and will remain low until the counter decrements to zero. The output then goes high and remains high until a new count is loaded into the counter. A trigger enables the counter to start decrementing.

Mode 1: Retriggerable One-Shot

The output goes low on the clock pulse following a trigger to begin the one-shot pulse and goes high when the counter reaches zero. Additional triggers result in reloading the count and starting the cycle over. If a trigger occurs before the counter decrements to zero, a new count is loaded. Thus, this forms a re-triggerable one-shot. In mode 1, a low output pulse is provided with a period equal to the counter count-down time.

Mode 2: Rate Generator

This mode provides a divide-by-N capability where N is the count loaded into the counter. When triggered, the counter output goes low for one clock period after N counts, reloads the initial count, and the cycle starts over. This mode is periodic, the same sequence is repeated indefinitely until the gate input is brought low.

Mode 3: Square Wave Generator

This mode operates periodically like mode 2. The output is high for half of the count and low for the other half. If the count is even, then the output is a symmetrical square wave. If the count is odd, then the output is high for (N+1)/2 counts and low for (N-1)/2 counts. Periodic triggering or frequency synthesis are two possible applications for this mode. Note that in this mode, to achieve the square wave, the counter decrements by two for the total loaded count, then reloads and decrements by two for the second part of the wave form.

Mode 4: Software Triggered Strobe

This mode sets the output high and, when the count is loaded, the counter begins to count down. When the counter reaches zero, the output will go low for one input period. The counter must be reloaded to repeat the cycle. A low gate input will inhibit the counter. This mode can be used to provide a delayed software trigger for initiating A/D conversions.

Mode 5: Hardware Triggered Strobe

In this mode, the counter will start counting after the rising edge of the trigger input and will go low for one clock period when the terminal count is reached. The counter is retriggerable. The output will not go low until the full count after the rising edge of the trigger.

PROGRAMMING

On this board the 8254 counters occupy the following addresses (hex):

Base Address + 14: Read/Write Counter 0 (Unused)
Base Address + 15: Read/Write Counter 1
Base Address + 16: Read/Write Counter 2
Base Address + 17: Write to Counter Control register

The counters are programmed by writing a control byte into a counter control register. The control byte specifies the counter to be programmed, the counter mode, the type of read/write operation, and the modulus. The control byte format is as follows:

B7	B6	B5	B4	B3	B2	B1	B0
SC1	SC0	RW1	RW0	M2	M1	M0	BCD

SC0-SC1: These bits select the counter that the control byte is destined for.

SC1	SC0	Function
0	0	Program Counter0**
0	1	Program Counter 1
1	0	Program Counter 2
1	1	Read/Write Cmd.*

* See section on READING AND LOADING THE COUNTERS.
** Not used.

RW0-RW1: These bits select the read/write mode of the selected counter.

RW1	RW0	Counter Read/Write Function
0	0	Counter Latch Command
0	1	Read/Write LS Byte
1	0	Read/Write MS Byte
1	1	Read/Write LS Byte, then MS Byte

M0-M2: These bits set the operational mode of the selected counter.

MODE	M2	M1	M0
0	0	0	0
1	0	0	1
2	X	1	0
3	X	1	1
4	1	0	0
5	1	0	1

BCD: Set the selected counter to count in binary (BCD = 0) or BCD (BCD = 1).

READING AND LOADING THE COUNTERS

If you attempt to read the counters on the fly when there is a high input frequency, you will most likely get erroneous data. This is partly caused by carries rippling through the counter during the read operation. Also, the low and high bytes are read sequentially rather than simultaneously and, thus, it is possible that carries will be propagated from the low to the high byte during the read cycle.

To circumvent these problems, you can perform a counter-latch operation in advance of the read cycle. To do this, load the RW1 and RW2 bits with zeroes. This instantly latches the count of the selected counter (selected via the SC1 and SC0 bits) in a 16-bit hold register. (An alternative method of latching counter(s) which has an additional advantage of operating simultaneously on several counters is by use of a readback command to be discussed later.) A subsequent read operation on the selected counter returns the held value. Latching is the best way to read a counter on the fly without disturbing the counting process. You can only rely on directly read counter data if the counting process is suspended while reading, by bringing the gate low, or by halting the input pulses.

For each counter you must specify in advance the type of read or write operation that you intend to perform. You have a choice of loading/reading (a) the high byte of the count, or (b) the low byte of the count, or (c) the low byte followed by the high byte. This last is of the most general use and is selected for each counter by setting the RW1 and RW0 bits to ones. Of course, subsequent read/load operations must be performed in pairs in this sequence or the sequencing flip-flop in the 8254 chip will get out of step.

The readback command byte format is:

B7	B6	B5	B4	B3	B2	B1	B0
1	1	CNT	STA	C2	C1	C0	0

CNT: When is 0, latches the counters selected by bits C0-C2.
STA: When is 0, returns the status byte of counters selected by C0-C2.
C0, C1, C2: When high, select a particular counter for readback. C0 selects Counter 0, C1 selects Counter 1, and C2 selects Counter 2.

You can perform two types of operations with the readback command. When CNT=0, the counters selected by C0 through C2 are latched simultaneously. When STA=0, the counter status byte is read when the counter I/O location is accessed. The counter status byte provides information about the current output state of the selected counter and its configuration. The status byte returned if STA=0 is:

B7	B6	B5	B4	B3	B2	B1	B0
OUT	NC	RW1	RW2	M2	M1	M0	BCD

OUT: Current state of counter output pin.
NC: Null count. This indicates when the last count loaded into the counter register has actually been loaded into the counter itself. The exact time of load depends on the configuration selected. Until the count is loaded into the counter itself, it cannot be read.
RW1, RW0: Read/Write command.
M2, M1, M0: Counter mode.
BCD: BCD = 0 is binary mode, otherwise counter is in BCD mode.

If both STA and CNT bits in the readback command byte are set low and the RW1 and RW0 bits have both been previously set high in the counter control register (thus selecting two-byte reads), then reading a selected counter address location will yield:

1st Read:	Status byte
2nd Read:	Low byte of latched data
3rd Read:	High byte of latched data

After any latching operation of a counter, the contents of its hold register must be read before any subsequent latches of that counter will have any effect. If a status latch command is issued before the hold register is read, then the first read will read the status, not the latched value.

Appendix C: Calibration

This board features digitally controlled potentiometers which are used to adjust the gain and offset of the A/D function and the gain of the DAC function.

This allows both the analog inputs and outputs to be calibrated from software in the field.

In order to obtain accurate data, the proper values must be loaded into the digital potentiometers each time the board is powered. If no constants are loaded, the potentiometers will power-on to the center of their ranges, which will result in a non- or poorly-calibrated situation.

When the board ships from the factory it already has a valid set of calibration constants preloaded into the EEPROM for your immediate use.

The various calibration steps are all wrapped up in a calibration program for your use. This program runs in DOS (and compatible environments only) and is written in Borland C/C++ 3.1 with source code provided. You will need a DVM and a calibrated voltage source to run the program.

The following steps are only necessary if you are writing your own calibration program, e.g. for a new operating system.

If you are unable to run the provided calibration program, it is recommended you examine its source code for details on performing the calibration in your own code.

The rest of the chapter is broken down into several sections. First is an **overview**, a kind of "executive summary" describing the two major steps involved in calibrating the board. Following this is a **breakdown** of the 5 calibration steps for the DAC. This breakdown is an "engineering summary" providing enough detail that someone very familiar with the board could proceed. Last an in-depth **step-by-step** walkthrough is provided. Steps are numbered consistently between the overview, breakdown, and step-by-step sections of this section so you can easily flip between them to build an understanding of the process.

Overview: Calibrating without using the provided calibration program

Two parts exist to the calibration process, and understanding these two parts is key to the entire procedure.

Step 1. Determine the calibration constants.
Step 2. Write the calibration constants into the calibration potentiometers.

Step 1 only needs to be performed if the board's data begins to appear out-of-calibration. A good rule of thumb is to determine new calibration constants every six to twelve months of regular use. If your environment undergoes frequent environmental changes, more frequent calibration may be indicated. **When the board ships from the factory it already has a valid set of calibration constants preloaded into the EEPROM for your immediate use**.

Step 2, writing the calibration constants into the digital calibration potentiometers, must occur each time the board is powered-up (or reset). Typically, each time your program executes you'll write these values, even if the program was run before.

Many devices using digital potentiometers require the software to load the calibration coefficients from a file-on-disk, matching the file to the board based on a manually entered serial number, or some similarly complex method. This board instead contains EEPROM to store the calibration constants. This makes it very simple: the board remembers its own constants, there's no need for a file on disk, or serial number lookup databases, etc.

The details are different between A/D calibration and DAC calibration, and are discussed separately, below.

Breakdown: Calibrating the DACs

DAC calibration is very simple, and provides a clear introduction to several of the concepts used in calibrating the A/D. The process is designed to evaluate the differences between what you've asked the DAC to output and what it really produces. This difference is the amount the board is out of calibration. By adjusting digital potentiometers in the DAC circuit, you reduce this calibration error by successive approximation until it reaches zero. When it is zero, and the board is calibrated, you store the amount of adjustment for later use.

First, let's expand Step 1 mentioned above into its component sub-steps for the DAC:

Step 1. **Determine the Calibration Constants for the DAC**

- 1.1 Output a value corresponding to a known voltage[1] on a DAC
- 1.2 Measure the output value of the DAC with a DVM
- 1.3 Adjust the value in the digital calibration potentiometer for that DAC until the voltage read by the DVM matches the known value being output.
- 1.4 Store the value from the digital calibration potentiometer into a spot[2] in the EEPROM for use on the next and subsequent board initializations.
- 1.5 Repeat steps for the other DAC.

Step 2. **Write the Calibration Constants into the Calibration Potentiometers**

For details on Step 2, please refer to section "Step 2" near the end of this appendix.

[1]: The value corresponding to a known voltage to output depends on the range of the DAC as selected by jumpers on the board. Software can determine the current jumper configured range using the status register at Base + 8 (Chapter 6).

[2]: The correct spot in EEPROM varies with the DAC number being calibrated, and the currently selected range (see note 1). Please note, you could use any location in the EEPROM you want, as long as you always use the same location. We recommend you use the same locations as our provided Calibration program, drivers, and samples, as shown in **Table C-1**, below.

Step-By-Step: Calibrating the DACs

Now lets describe these steps in detail.

1.1 Output a known value to the DAC. You should determine the maximum range of the DAC using Base + 8 (see Chapter 6). Pick a value approximately 5% lower than this maximum. By using a value lower than the maximum you avoid calibrating off-the-end of the range. Alternately, choose the voltage most likely to be output by your application in your own use. For example, if you're going to be using primarily voltages near 6.2 Volts, you may want to calibrate it at that voltage. Just substitute whatever voltage you choose in the math that follows.

Output this value to the DAC using the process described at Base + 9.

For example, if your DAC is jumpered for the 0-10 Volt output range, a value of 9.5 Volts (95% of 10 Volts) would be a good choice.

Convert this voltage to a 12-bit digital count value (Counts = (Volts / MaxVolts) * MaxCounts, so: Counts = (9.5 / 10) * (2^12 - 1) = 0.95 * 4095 = 3890.25. Only a 12-bit integer numbers can be written to the DAC, so we'll use 3890 as our count value. This equates to F32 hex, which we'll write to the DAC as described at Base + 9. Its important that you are precise when calibrating, so don't forget about that 0.25 count we threw away. If we run the equation backwards to determine what voltage F32 counts equates to, we don't get 9.5 volts. Let's run the math: Volts = (Counts / MaxCounts) * MaxVolts, so Volts = (F32 / FFF) * 10 = 0.9499 * 10 = 9.499 Volts. Not quite 9.5 volts.

1.2 Measure the output value of the DAC with a DVM. Now that we know precisely what output voltage to expect, connect a DVM to the DAC output pin on P1 (pin 25 or pin 26 for DAC 0 or DAC 1, respectively. Use Pin 24 for ground.) The DVM should be set for DC voltage measurement. Once connected the DVM

should read 9.499 Volts, but may not be exactly correct. Remember, if you're using a different "known output value", you should see a number near it, not near 9.499.

1.3 Adjust the value in the digital calibration potentiometer for the DAC you're calibrating until the reading on the DVM shows your known output value. You adjust this potentiometer's value using the process described in Base + B in Chapter 6. Probably the best way to quickly find the correct value is to initialize the potentiometer with half its maximum value (use 80hex), then increment or decrement the value until the DVM reads accurately.

1.4 Once you've determined the value that correctly adjusts the DAC output to match the known output voltage, store it in the EEPROM for later use. Doing so allows you to re-write the value into the calibration potentiometer on the next board initialization without resorting to storing the values in a reference database or calibration configuration files on a hard disk or floppies, etc. The location into which you store the value varies based on the DAC number you're calibrating and the range you have selected on the board via jumpers. Consult **Table C-1** for a list of the locations the provided Calibration software, samples, and drivers use.

1.5 Repeat these steps for the other DAC. Doing so means switching pins that you're reading on the DVM, changing the channel select bits in the writes to Base + 9, and changing the location you're writing the correct value into, as described in the above steps.

When you've finished these steps the DACs are calibrated. The next time the board is reset, only Step 2 needs to be performed. In brief, this involves reading the value out of the correct EEPROM location and writing it to the calibration potentiometer. The details of Step 2 are described near the end of this appendix, below.

Breakdown: Calibrating the A/D

A/D Calibration, while fundamentally different than the DAC calibration process described above, is also very similar. In the A/D calibration you are determining the amount of calibration error, adjusting the digital potentiometers until the error is eliminated, and storing the adjustment for later use. Unlike the DAC, there are two digital potentiometers for the A/D (offset adjust and gain adjust). The same two steps apply:

Step 1. Determine the calibration constants.
Step 2. Write the calibration constants into the calibration potentiometers.

First, let's expand Step 1 into its component sub-steps for the A/D:

Step 1. **Determine the Calibration Constants for the A/D**

1.1 Offset Adjust

1.1.1 Apply a known low voltage[1] to the A/D input.
1.1.2 Acquire the voltage using the A/D converter.
1.1.3 Adjust the value in the digital calibration potentiometer for the A/D until the voltage read by the A/D matches the input voltage.
1.1.4 Store the value from the digital calibration potentiometer into a spot[2] in the EEPROM for use on the next and subsequent board initializations.

1.2 Gain Adjust

1.2.1 Apply a known voltage[3] to the A/D Input.
1.2.2 Acquire the voltage using the A/D converter.
1.2.3 Adjust the value in the digital calibration potentiometer for the A/D until the voltage read by the A/D matches the input voltage.
1.2.4 Store the value from the digital calibration potentiometer into a spot[2] in the EEPROM for use on the next and subsequent board initializations.

Step 2. **Write the Calibration Constants into the Calibration Potentiometers**
For details on step 2, please refer to the section "Step 2" near the end of this appendix.

Note 1: The known voltage to use varies with the jumper selected A/D input range. For best results apply a voltage within 5% of the minimum scale (most negative) voltage for your selected range..

Note 2: The correct spot in EEPROM varies with the A/D range and single-ended/differential selection being calibrated. We recommend you use the same locations as our provided Calibration program, drivers, and samples, as shown in **Table C-1**, below.

Note 3: The known voltage to use varies with the jumper selected A/D input range. For best results apply a voltage within 5% of the full scale voltage for your selected range.

Step-By-Step: Calibrating the A/D

Step 1. **Determine the Calibration Constants for the A/D**

1.1 Offset Adjust

1.1.1 Apply a known low voltage to the A/D input. For best results, all channels should have this voltage, but any single channel would also work. The voltage should be 5% above the minimum scale (most negative) voltage for your selected range. For example, in the ±10V range, this would be -9V, and in the 0-10V range, this would be 0.5V.

1.1.2 Acquire the voltage using the A/D converter. The simplest way is using the Software Mode. Software mode is described in Chapter 4. In essence it consists of five steps: Set Channel, Set Gain, Start Conversion, Wait for End of Conversion, Read Data. For calibration purposes the Channel should be whichever channel has the known voltage applied. The Channel Scan Limits register at Base +2 should contain only that one channel. Read the data using the EMPTY bit to indicate when data is available. Software gain at Base +4 should be configured for whichever range you'll actually be using, or the Gain x1 setting. The software gain amplifier is a laser-trimmed part and does not need separate calibration per setting. For optimum results data should be heavily averaged to eliminate any noise from the computations.

1.1.3 Adjust the value in the digital calibration potentiometer for the A/D until the voltage read by the A/D is correct. Probably the best way to quickly find the correct value is to initialize the potentiometer with half its maximum value (use 80 hex), then increment or decrement the value until the A/D reads accurately.

1.1.4 Store the value from the digital calibration potentiometer into a spot in the EEPROM for use on the next and subsequent board initializations. The correct spot in EEPROM varies with the A/D range and single-ended/differential selection being calibrated. We recommend you use the same locations as our provided Calibration program, drivers, and samples, as shown in **Table C-1**, below. For example, if you are calibrating the Offset of the 0-10 Volts Single-Ended setting of the board, we recommend you write the calibration Potentiometer value into the "5" location of the EEPROM. Write the value using the procedure outlined in the description of Base +A in Chapter 6.

1.2 Gain Adjust

1.2.1 Apply a known voltage to the A/D Input. When adjusting the gain, a known voltage 5% below the maximum input of the A/D converter will result in a good reading across the entire range. Alternately, calibrating the A/D to a voltage near the actual application voltage you'll be expecting in your system results in perfect readings in your specific system. In the vast majority of cases, either calibration method will result in correct data. Continuing our example from above, when calibrating the 0-10V range a voltage near 10 Volts is recommended: we'll use 9.5 Volts as our Known Voltage. Using a calibrated voltage source apply your known voltage to the inputs of at least one A/D channel. The other channels should not have signals connected, or should be grounded. To connect a known voltage to channel 0, connect the voltage to P2 Pin 1, and use P2 Pin 3 as the reference ground.

1.2.2 Acquire the voltage using the A/D converter. See step 1.1.2

1.2.3 Adjust the value in the digital calibration potentiometer for the A/D until the voltage read by the A/D matches the input voltage. Many methods of determining values for the digital calibration potentiometer exist. One possible method: Set the Pot to "0" and take a reading. If the result is off-scale (reading 0000 or FFFF counts) set the Pot to "FF" and take another reading. One of these two readings will not be "railed" off-scale. Increment or decrement the Pot load value until the data read from the A/D reads the Known Voltage. To convert from counts to voltage use the following equation: Volts=Span*Counts/MaxCounts-Offset. MaxCounts on a 16-bit A/D is 65536, and Span and Offset vary with the selected range. In any unipolar range, Offset is zero, and the Span is equal to the maximum voltage. In any bipolar range the Span is equal to the maximum input voltage minus the minimum input voltage, and Offset is half of this value. For example, ±5Volt range has a Span of 10V and an Offset of 5V. If the A/D reads FFE9 counts on a 0-10V range, the voltage being indicated is 10*FAE9/FFFF-0, or 9.801 Volts.

1.2.4 Store the value from the digital calibration potentiometer into a spot in the EEPROM for use on the next and subsequent board initializations. The correct spot in EEPROM varies with the A/D range and single-ended/differential selection being calibrated. We recommend you use the same locations as our provided Calibration program, drivers, and samples, as shown in **Table C-1**, below. For example, if you are calibrating the Gain of the 0-10 Volts Single-Ended setting of the board, we recommend you write the calibration Potentiometer value into the "D" location of the EEPROM. Write the value using the procedure outlined in the description of Base +A in Chapter 6.

Step 2. **Write the Calibration Constants into the Calibration Potentiometers**

Step 2 applies to both DAC and A/D Calibration. Step 1 involved applying or measuring voltages, connecting pins and sources etc. Step 1 only needs to be performed if the data from the A/D appears to be out-of-calibration, or approximately every 6-12 months depending on environmental and usage considerations. Step 2 however needs to be performed every time the board is reset. In Step 2 you merely perform a read from the EEPROM and write the value to the digital calibration potentiometers, for each of the digital pots.

2.1 Determine the location in the EEPROM for the calibration constants. The register at Base +8 (Chapter 6) indicates the currently jumper selected range for both the A/D and the DAC. Read this register. Software should not assume the user has left the range setting where it was.

2.2 Read the correction constants from the EEPROM. Correlate the jumper settings as read with the **Table C-1** and read the EEPROM entries for the A/D Offset, A/D Gain, and the DAC 0 and DAC 1 corrections. Base +A in Chapter 6 describes the process of reading from the EEPROM.

2.3 Write each calibration constant to the correct Digital Calibration Potentiometer. Refer to Chapter 6 Base +B for more information on writing to the Digital Potentiometers.

EEPROM Location	Calibration Value Stored
0	unused
1	unused
2	Offset “B” ±10V Differential
3	Offset “B” ±10V Single-ended
4	Offset “B” 0-10V Differential
5	Offset “B” 0-10V Single-ended
6	Offset “B” ±5V Differential
7	Offset “B” ±5V Single-ended
8	unused
9	unused
A	Scale “M” ±10V Differential
B	Scale “M” ±10V Single-ended
C	Scale “M” 0-10V Differential
D	Scale “M” 0-10V Single-ended
E	Scale “M” ±5V Differential
F	Scale “M” ±5V Single-ended
10	DAC 0, 0-10V Range
11	DAC 0, 0-5V Range
12	DAC 1, 0-10V Range
13	DAC 1, 0-5V Range
14-3F	unused

Table C-1: Factory EEPROM Calibration Locations

Remember that all of these steps are already encapsulated in a “C” language DOS compatible Calibration program that ships free with the board. For the fastest way to write your own calibration program, consider referring to the source code of the Calibration program.

Customer Comments

If you experience any problems with this manual or just want to give us some feedback, please email us at: ***manuals@accesio.com***. Please detail any errors you find and include your mailing address so that we can send you any manual updates.

ACCES I/O PRODUCTS, INC.
10623 Roselle Street, San Diego CA 92121-1506
Tel. (858) 550-9559 FAX (858) 550-7322
www.accesio.com