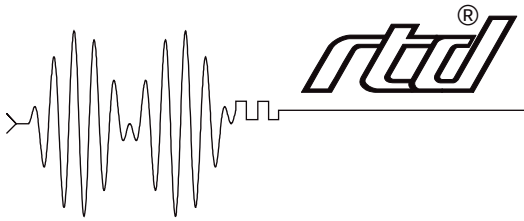
AD2110/ADA2110
User's Manual



Real Time Devices USA, Inc.

"Accessing the Analog World"®

Publication No. 2110-9/1/99

AD2110/ADA2110
User's Manual

REAL TIME DEVICES USA, INC.
Post Office Box 906
State College, Pennsylvania 16804 USA
Phone: (814) 234-8087
FAX: (814) 234-5218

Published by
Real Time Devices USA, Inc.
P.O. Box 906
State College, PA 16804 USA

Printed in U.S.A.

Table of Contents

List of Illustrations

INTRODUCTION

The AD2110 and ADA2110 Low Cost Industrial Control boards turn your IBM PC/XT/AT or compatible into a high-performance data acquisition and control system. Installed within a single short or full size expansion slot in the computer, each 2110 series board features:

- 16 single-ended or 8 differential analog input channels,
- 12-bit, 20 microsecond analog-to-digital converter with 40 kHz throughput,
- ±5, ±10, or 0 to +10 volt input range,
- Programmable gains of 1, 2, 4 & 8 with on-board gain multiplier,
- 16 TTL/CMOS 8255-based digital I/O lines which can be configured with pull-up or pull-down resistors,
- Three 16-bit timer/counters,
- Two 12-bit digital-to-analog output channels (ADA2110 only),
- ±5, ±10, 0 to +5, or 0 to +10 volt analog output range (ADA2110 only),
- Turbo Pascal, Turbo C, and BASIC source code; diagnostics program.

The following paragraphs briefly describe the major functions of the board. A more detailed discussion of board functions is included in Chapter 3, *Hardware Operation*, and Chapter 4, *Board Operation and Programming*. The board setup is described in Chapter 1, *Board Settings*.

Analog-to-Digital Conversion

The analog-to-digital (A/D) circuitry receives up to 16 single-ended or 8 differential analog inputs and converts these inputs into 12-bit digital data words which can then be read and/or transferred to PC memory.

The analog input voltage range is jumper-selectable for bipolar ranges of -5 to +5 volts or -10 to +10 volts, or a unipolar range of 0 to +10 volts. The board is factory set for -5 to +5 volts. Overvoltage protection to ±35 volts is provided at the inputs. The high-performance A/D converter supports fast-settling, software programmable gains of 1, 2, 4, and 8 with on-board gain multiplier circuitry so that you can customize the input gain.

A/D conversions are performed by an industry standard successive approximation converter. The converter and high-speed sample-and-hold amplifier before it combine to provide a maximum throughput rate of 40 kHz.

The converted data is read and/or transferred to PC memory, one byte at a time, through the PC data bus.

Digital-to-Analog Conversion (ADA2110 Only)

The digital-to-analog (D/A) circuitry on the ADA2110 features two independent 12-bit analog output channels with individually jumper-selectable output ranges of -5 to +5 volts, -10 to +10 volts, 0 to +5 volts, or 0 to +10 volts. Data is programmed into the D/A converter by two write operations. The outputs of both channels are simultaneously updated by a single write operation.

8254 Timer/Counter

An 8254 programmable interval timer contains three 16-bit, 8-MHz timer/counters to support a wide range of timing and counting functions.

Digital I/O

The 2110 has 16 TTL/CMOS-compatible digital I/O lines which can be directly interfaced with external devices or signals to sense switch closures, trigger digital events, or activate solid-state relays. These lines are provided by the on-board 8255 programmable peripheral interface chip. Pads for installing and activating pull-up or pull-down resistors are included on the board. Installation procedures are given near the end of Chapter 1, *Board Settings.*

What Comes With Your Board

You receive the following items in your 2110 package:

- AD2110 or ADA2110 interface board
- Software and diagnostics diskette with Turbo Pascal, Turbo C, and BASIC source code
- User's manual

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

Board Accessories

In addition to the items included in your 2110 package, Real Time Devices offers a full line of software and hardware accessories. Call your local distributor or our main office for more information about these accessories and for help in choosing the best items to support your board's application.

Application Software and Drivers

Our custom application software packages provide excellent data acquisition and analysis support. Use SIGNAL*MATH for integrated data acquisition and sophisticated digital signal processing and analysis, or SIGNAL*VIEW for monitoring and data acquisition. rtdLinx and rtdLinx/NB drivers provide full-featured high level interfaces between the board and custom or third party software, including Labtech Notebook, Notebook/XE, and LT/Control. rtdLinx source code is available for a one-time fee.

Hardware Accessories

Hardware accessories for the 2110 include the MX32 analog input expansion board which can expand a single input channel on your board to 16 differential or 32 single-ended input channels, MR series mechanical relay output boards, OP series optoisolated digital input boards, the OR16 mechanical relay/optoisolated digital I/O board, the TS16 thermocouple sensor board, the TB50 terminal board and XB50 prototype/terminal board for prototype development and easy signal access, EX-XT and EX-AT extender boards for simplified testing and debugging of prototype circuitry, and the XT50 twisted pair flat ribbon cable assembly for external interfacing.

Using This Manual

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

When You Need Help

This manual and the example programs in the software package included with your board provide enough information to properly use all of the board's features. If you have any problems installing or using this board, contact our Technical Support Department, (814) 234-8087, during regular business hours, eastern standard time or eastern daylight time, or send a FAX requesting assistance to (814) 234-5218. When sending a FAX request, please include your company's name and address, your name, your telephone number, and a brief description of the problem.

CHAPTER 1

BOARD SETTINGS

The AD2110 and ADA2110 boards have jumper and switch settings you can change if necessary for your application. The 2110 is factory-configured as listed in the table and shown on a diagram in the beginning of this chapter. Should you need to change these settings, use these easy-to-follow instructions before you install the board in your computer.

Note that by installing resistor packs at three locations around the 8255 PPI and soldering jumpers in the associated pads, you can configure the 16 available digital I/O lines to be pulled up or pulled down. This procedure is explained near the end of this chapter.

Also note that by installing components at R3, R4, TR4, and C14, you can add your own resistor configurable gain. The gain circuitry is described at the end of this chapter.

Factory-Configured Switch and Jumper Settings

Table 1-1 lists the factory settings of the user-configurable jumpers and switch on the AD2110 and ADA2110 boards. Figure 1-1 shows the board layout and the locations of the factory-set jumpers. The following paragraphs explain how to change the factory settings. Pay special attention to the setting of S1, the base address switch, to avoid address contention when you first use your board in your system.

Table 1-1: Factory Settings

Switch/ Jumper	Function Controlled	Factory Settings (Jumpers Installed)
P3	Sets the clock sources for the 8254 timer/counters (TC0-TC2)	Jumpers installed on CLK0-OSC, CLK1-OT0, CLK2-OT1 (timers cascaded)
P4	Sets the D/A output voltage range for DAC 1	Jumpers installed on ±5, X1 (+5 to -5 volts)
P5	Sets the D/A output voltage range for DAC 2	Jumpers installed on ±5, X1 (+5 to -5 volts)
P6	Selects single-ended or differential analog input type	Single-ended (jumpers installed on three S pins)
P7	Sets the analog input voltage range and polarity	Jumpers installed on 10V, +/- (-5 to +5 volts)
P8	Selects the gate source for TC1 and TC2	GT1-+5V, GT2-+5V
P9	Connects one of five interrupt sources to an interrupt channel	Jumper installed on OUT2, interrupt channel disabled
P10	Connects the EOC signal to an interrupt channel	Disabled
S1	Sets the base address	300 hex (768 decimal)

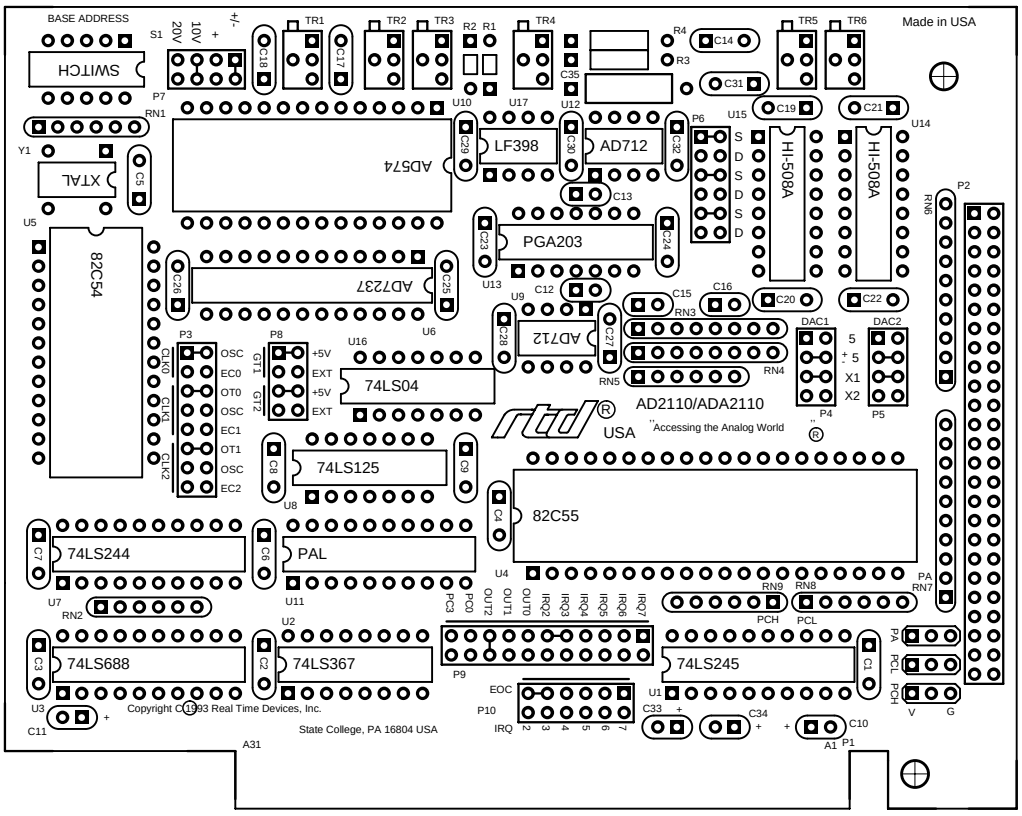


Fig. 1-1 — Board Layout Showing Factory-Configured Settings

P3 — 8254 Timer/Counter Clock Sources (Factory Settings: CLK0-OSC, CLK1-OT0, CLK2-OT1)

This header connector, shown in Figure 1-2, lets you select the clock sources for the 8254 timer/counters, TC0, TC1, and TC2. The factory setting cascades all three timer/counters, with the clock source for TC0 being the on-board 8 MHz oscillator, the output of TC0 providing the clock for TC1, and the output of TC1 providing the clock for TC2. You can connect any or all of the sources to an external clock input through the P2 I/O connector, or you can set TC1 and TC2 to be clocked by the 8 MHz oscillator. Figure 1-3 shows a block diagram of the timer/counter circuitry to help you with these connections.

NOTE: When installing jumpers on this header, make sure that only one jumper is installed in each group of two or three CLK pins.

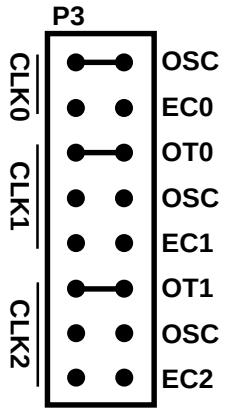


Fig. 1-2 — 8254 Timer/Counter Clock Source Jumpers, P3

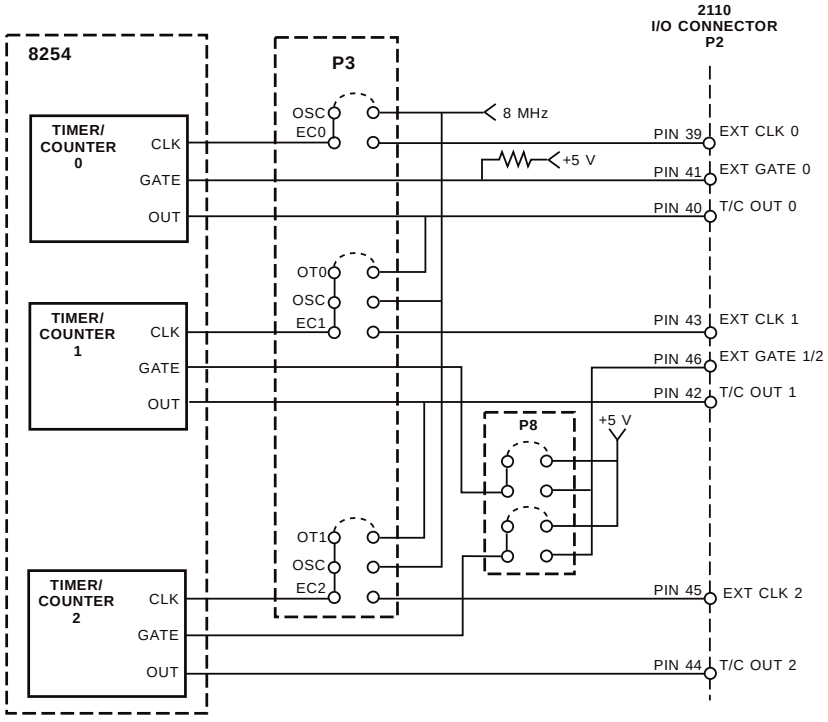


Fig. 1-3 — 8254 Timer/Counter Circuit Block Diagram

P4 — DAC 1 Output Voltage Range (Factory Setting: +5 to -5 volts)

This header connector, shown in Figure 1-4, sets the output voltage range for DAC 1 at 0 to +5, ±5, 0 to +10, or ±10 volts. Two jumpers must be installed, one to select the range and one to select the multiplier. The top two jumpers set the range, bipolar (±5) or unipolar (5). The bottom two jumpers set the multiplier, X2 or X1. When a jumper is on X2, the range values become ±10 and 10. The table below shows the four possible combinations of jumper settings, and the diagram shows the factory setting. This header does not have to be set the same as P5.

	Jumpers (Top to Bottom)			
Voltage Range	**5**	**±5**	**X1**	**X2**
-5 to +5 volts	OFF	ON	ON	OFF
0 to +5 volts	ON	OFF	ON	OFF
-10 to +10 volts	OFF	ON	OFF	ON
0 to +10 volts	ON	OFF	OFF	ON

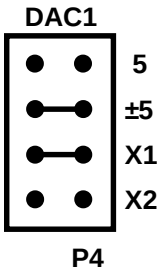


Fig. 1-4 — DAC 1 Output Voltage Range Jumper, P4

P5 — DAC 2 Output Voltage Range (Factory Setting: +5 to -5 volts)

This header connector, shown in Figure 1-5, sets the output voltage range for DAC 2 at 0 to +5, ±5, 0 to +10, or ±10 volts. Two jumpers must be installed, one to select the range and one to select the multiplier. The top two jumpers set the range, bipolar (±5) or unipolar (5). The bottom two jumpers set the multiplier, X2 or X1. When a jumper is on X2, the range values become ±10 and 10. The table below shows the four possible combinations of jumper settings, and the diagram shows thefactory setting. This header does not have to be set the same as P4.

	Jumpers (Top to Bottom)			
Voltage Range	**5**	**±5**	**X1**	**X2**
-5 to +5 volts	OFF	ON	ON	OFF
0 to +5 volts	ON	OFF	ON	OFF
-10 to +10 volts	OFF	ON	OFF	ON
0 to +10 volts	ON	OFF	OFF	ON

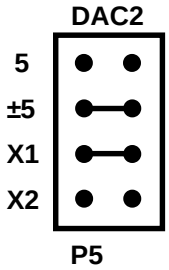


Fig. 1-5 — DAC 2 Output Voltage Range Jumper, P5

P6 — Single-Ended/Differential Analog Inputs (Factory Setting: Single-Ended)

This header connector, shown in Figure 1-6, configures the board for 8 differential or 16 single-ended analog input channels. When operating in the single-ended mode, three jumpers must be installed across the S pins. When operating in the differential mode, three jumpers must be installed across the D pins. DO NOT install jumpers across both S and D pins at the same time!

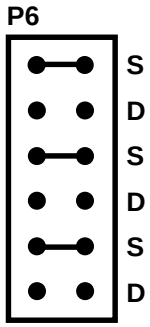


Fig. 1-6 — Single-Ended/Differential Analog Input Signal Type Jumpers, P6

P7 — Analog Input Voltage Range and Polarity (Factory Setting: -5 to +5 Volts)

This header connector, shown in Figure 1-7, sets the analog input voltage range and polarity. Two jumpers are installed to select one of three input ranges, as shown in the diagram: ±5, ±10, and 0 to +10 volts.

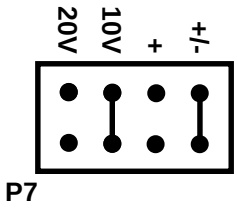


Fig. 1-7a: -5 to +5 volts (Factory Setting)

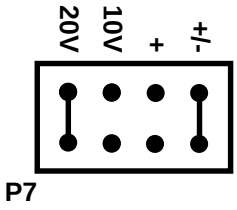


Fig. 1-7b: -10 to +10 volts

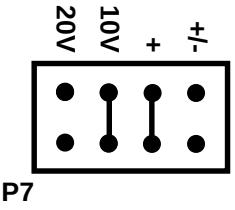


Fig. 1-7c: 0 to +10 volts

Fig. 1-7 —Analog Input Voltage Range and Polarity Jumper, P7

P8 — 8254 Timer/Counter TC1 & TC2 Gate Sources (Factory Setting: GT1-+5V, GT2-+5V)

This header connector, shown in Figure 1-8, lets you select the gate sources for the 8254 timer/counters, TC1 and TC2. Each gate can be independently connected to +5 volts (tied high), or to the external gate 1/2 signal at I/O connector P2, pin 46.

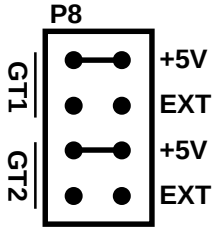


Fig. 1-8 — 8254 Timer/Counter TC1 & TC2 Gate Source Jumpers, P8

P9 — Interrupt Source and Channel (Factory Setting: Jumper on OUT2; Interrupt Channels Disabled)

This header connector, shown in Figure 1-9, lets you connect any one of five interrupt sources to any of six interrupt channels, IRQ2 (highest priority channel) through IRQ7 (lowest priority channel). To activate a channel, you must install a jumper vertically across the desired IRQ channel. Figure 1-9a shows the factory setting; Figure 1-9b shows interrupt source OUT2 connected to IRQ3.

On the left side of the header, you can select any one of five signal sources to generate an interrupt. An interrupt source is chosen by placing a jumper across the desired pair of pins. The interrupt sources available are timer/ counter outputs OUT0, OUT1, and OUT2, and the 8255 PPI's PC0 (INTRB) and PC3 (INTRA) signals. Note that only ONE interrupt source on this header can be activated at a time. If you are also using the EOC interrupt on P10, make sure that you select different IRQ channels on each header.

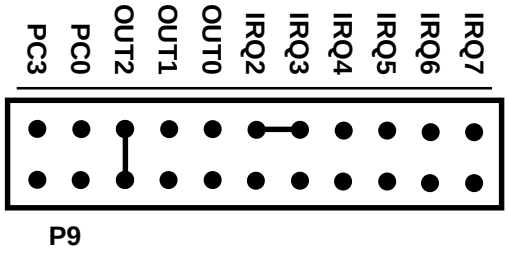


Fig. 1-9a — Factory Setting (IRQ disabled)

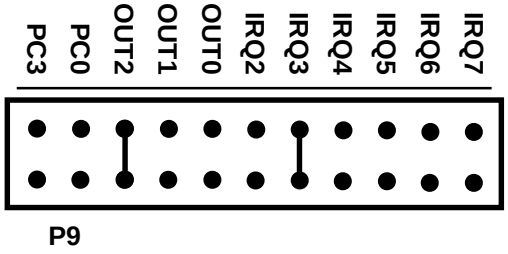


Fig. 1-9b — OUT2 connected to IRQ3

Fig. 1-9 — Interrupt Source and Channel Select Jumper, P9

P10 — EOC Interrupt Channel Select (Factory Setting: Disabled)

This header connector, shown in Figure 1-10, lets you connect the end-of-convert signal from the A/D converter to an interrupt channel, IRQ2 (highest priority channel) through IRQ7 (lowest priority channel). To activate this interrupt, you must install a jumper vertically across the desired IRQ channel. Figure 1-10a shows the factory setting; Figure 1-10b shows the EOC interrupt source connected to IRQ4.

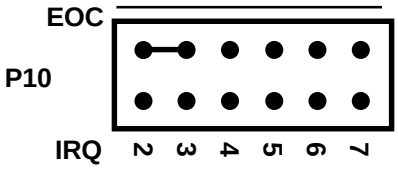


Fig. 1-10a:
Factory Setting

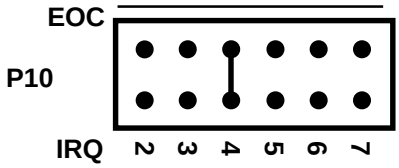


Fig. 1-10b: EOC
connected to IRQ4

Fig. 1-10 — EOC Interrupt Channel Jumper, P10

S1 — Base Address (Factory Setting: 300 hex (768 decimal))

One of the most common causes of failure when you are first trying your board is address contention. Some of your computer's I/O space is already occupied by internal I/O and other peripherals. When the 2110 board attempts to use I/O address locations already used by another device, contention results and the board does not work.

To avoid this problem, the 2110 has an easily accessible five-position DIP switch, S1, which lets you select any one of 32 starting addresses in the computer's I/O. Should the factory setting of 300 hex (768 decimal) be unsuitable for your system, you can select a different base address simply by setting the switches to any one of the values listed in Table 1-2. The table shows the switch settings and their corresponding decimal and hexadecimal (in parentheses) values. Make sure that you verify the order of the switch numbers on the switch (1 through 5) before setting them. When the switches are pulled forward, they are OPEN, or set to logic 1, as labeled on the DIP switch package. When you set the base address for your board, record the value in the table inside the back cover. Figure 1-11 shows the DIP switch set for a base address of 300 hex (768 decimal).

Table 1-2: Base Address Switch Settings, S1

Base Address Decimal / (Hex)	Switch Setting 5 4 3 2 1	Base Address Decimal / (Hex)	Switch Setting 5 4 3 2 1
512 / (200)	0 0 0 0 0	768 / (300)	1 0 0 0 0
528 / (210)	0 0 0 0 1	784 / (310)	1 0 0 0 1
544 / (220)	0 0 0 1 0	800 / (320)	1 0 0 1 0
560 / (230)	0 0 0 1 1	816 / (330)	1 0 0 1 1
576 / (240)	0 0 1 0 0	832 / (340)	1 0 1 0 0
592 / (250)	0 0 1 0 1	848 / (350)	1 0 1 0 1
608 / (260)	0 0 1 1 0	864 / (360)	1 0 1 1 0
624 / (270)	0 0 1 1 1	880 / (370)	1 0 1 1 1
640 / (280)	0 1 0 0 0	896 / (380)	1 1 0 0 0
656 / (290)	0 1 0 0 1	912 / (390)	1 1 0 0 1
672 / (2A0)	0 1 0 1 0	928 / (3A0)	1 1 0 1 0
688 / (2B0)	0 1 0 1 1	944 / (3B0)	1 1 0 1 1
704 / (2C0)	0 1 1 0 0	960 / (3C0)	1 1 1 0 0
720 / (2D0)	0 1 1 0 1	976 / (3D0)	1 1 1 0 1
736 / (2E0)	0 1 1 1 0	992 / (3E0)	1 1 1 1 0
752 / (2F0)	0 1 1 1 1	1008 / (3F0)	1 1 1 1 1

0 = closed, 1 = open

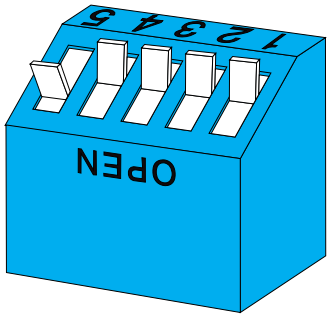


Fig. 1-11 — Base Address Switch, S1

Pull-up/Pull-down Resistors on Digital I/O Lines

The 8255 programmable peripheral interface provides 16 TTL/CMOS compatible digital I/O lines which can be interfaced with external devices. These lines are divided into three groups: eight Port A lines, four Port C Lower lines, and four Port C Upper lines. (The eight lines of Port B are used for internal board functions.) You can install and connect pull-up or pull-down resistors for any or all of these three groups of lines. You may want to pull lines up for connection to switches. This will pull the line high when the switch is disconnected. Or, you may want to pull lines down for connection to relays which control turning motors on and off. These motors turn on when the digital lines controlling them are high. The Port A lines of the 8255 automatically power up as inputs, which can float high during the few moments before the board is first initialized. This can cause the external devices connected to these lines to operate erratically. By pulling these lines down, when the data acquisition system is first turned on, the motors will not switch on before the 8255 is initialized.

To use the pull-up/pull-down feature, you must first install resistor packs in any or all of the three locations near the 8255, labeled PA, PCL, and PCH. PA takes a 10-pin pack, and PCL and PCH take 6-pin packs. Figure 1-13 shows a blowup of the PA, PCL, and PCH resistor pack locations.

After the resistor packs are installed, you must connect them into the circuit as pull-ups or pull-downs. Locate the three-hole pads on the board below the resistor packs. They are labeled G (for ground) on one end and V (for +5V) on the other end. The middle hole is common. PA is for Port A, PCL is for Port C Lower, and PCH is for Port C Upper. Figure 1-13 shows these pads. To operate as pull-ups, solder a jumper wire between the common pin (middle pin of the three) and the V pin. For pull-downs, solder a jumper wire between the common pin (middle pin) and the G pin. Figure 1-12 shows Port A lines with pull-ups, Port C Lower with pull-downs, and Port C Upper with no resistors.

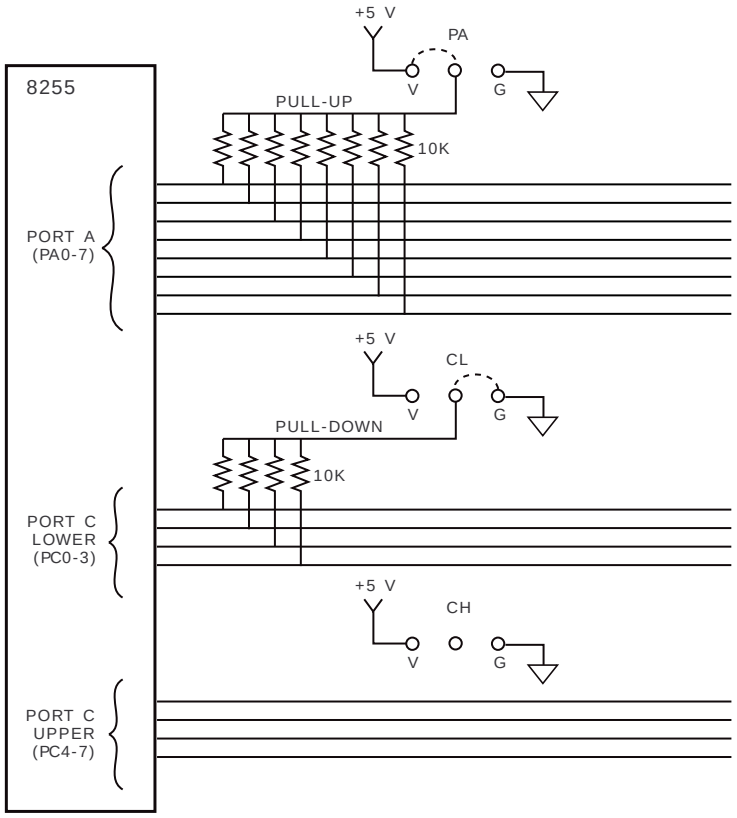


Fig. 1-12 — Adding Pull-ups and Pull-downs to Digital I/O Lines

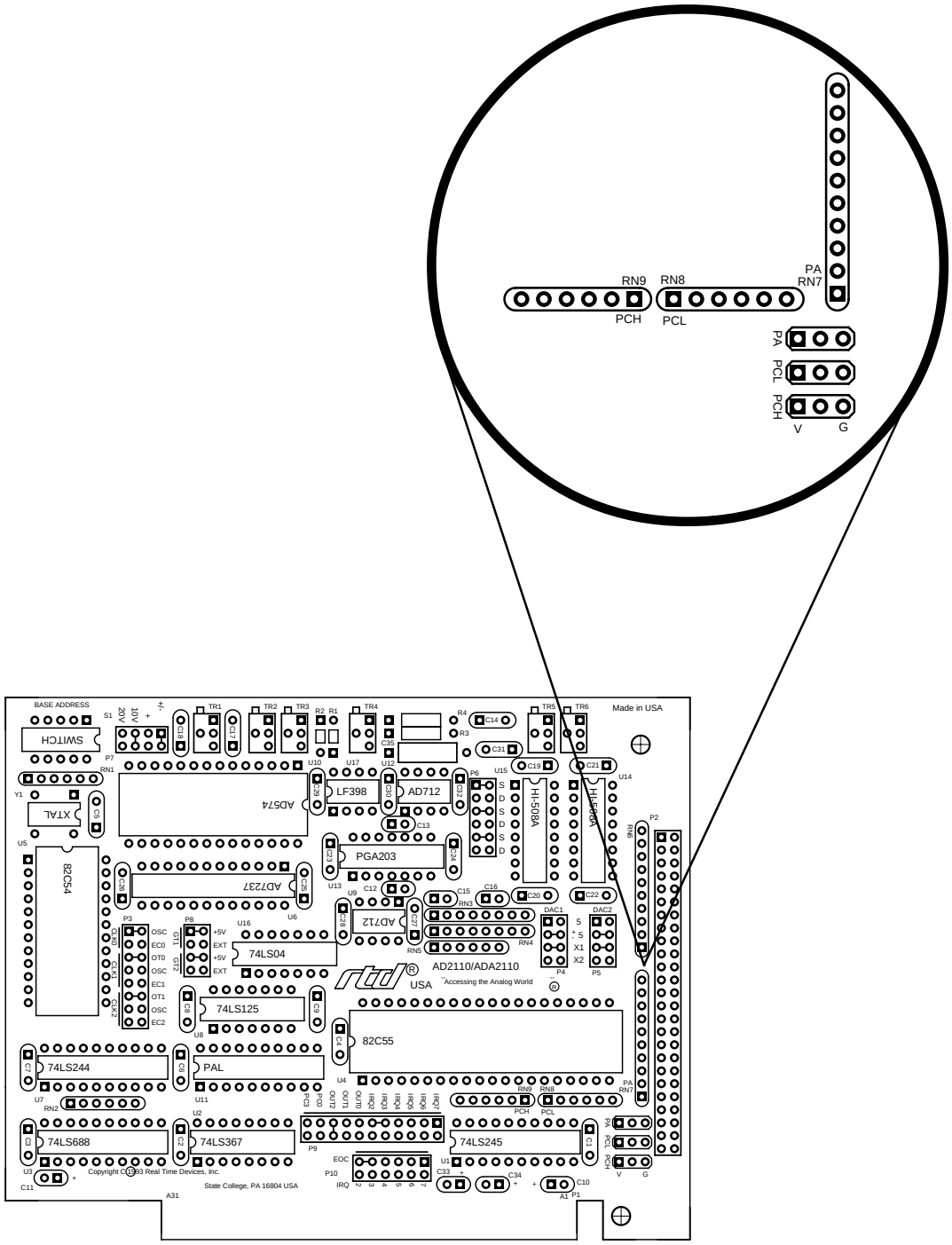


Fig. 1-13 — Pull-up/Pull-down Resistor Circuitry

Gm, Gain Multiplier Circuitry

The 2110 has software programmable binary gains of 1, 2, 4, and 8. A gain multiplier circuit, Gm, is provided so that you can easily configure special gain settings for a specific application. Note that when you use this feature and set up the board for a gain of other than 1, all of the input channels will operate only at your custom gain settings. In other words, if you install circuitry which gives you a gain multiplier of 10, then the four programmable gains available are 10, 20, 40, and 80.

Gm is derived by adding resistors R3 and R4, trimpot TR4, and capacitor C14, all located in the upper center and right areas of the board. The resistors and trimpot combine to set the gain, as shown in the formula in Figure 1-14. Capacitor C14 is provided so that you can add low-pass filtering in the gain circuit. If your input signal is a slowly changing one and you do not need to measure it at a higher rate, you may want to add a capacitor at C14 in order to reduce the input frequency range and in turn reduce the noise on your input signal. The formula for setting the frequency is given in the diagram. Figure 1-14 shows how the Gm circuitry is configured.

As shown in Figure 1-14, a solder short must be removed from the board to activate the Gm circuitry. This short is located on the **bottom side** of the board under U12 (AD712 IC). Figure 1-15 shows the location of the solder short.

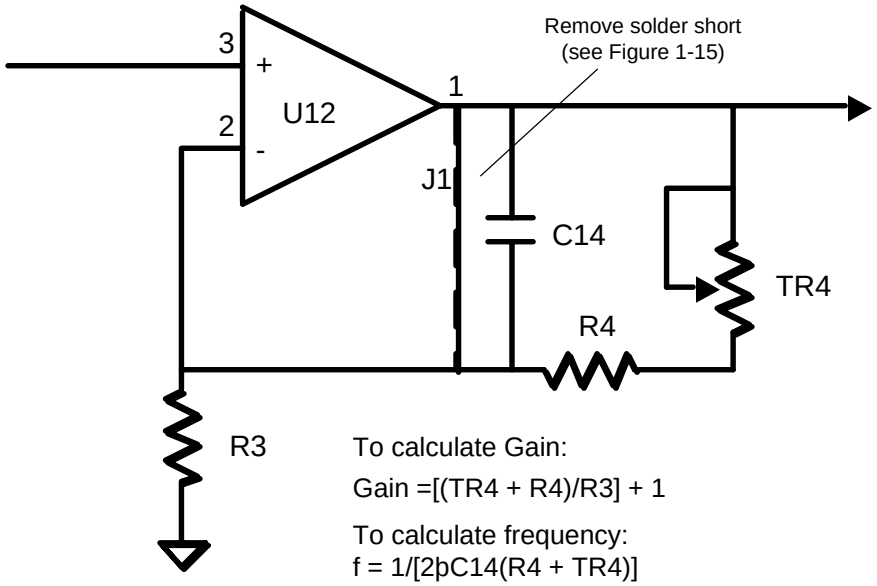


Fig. 1-14 — Gain Circuitry and Formulas for Calculating Gain and f

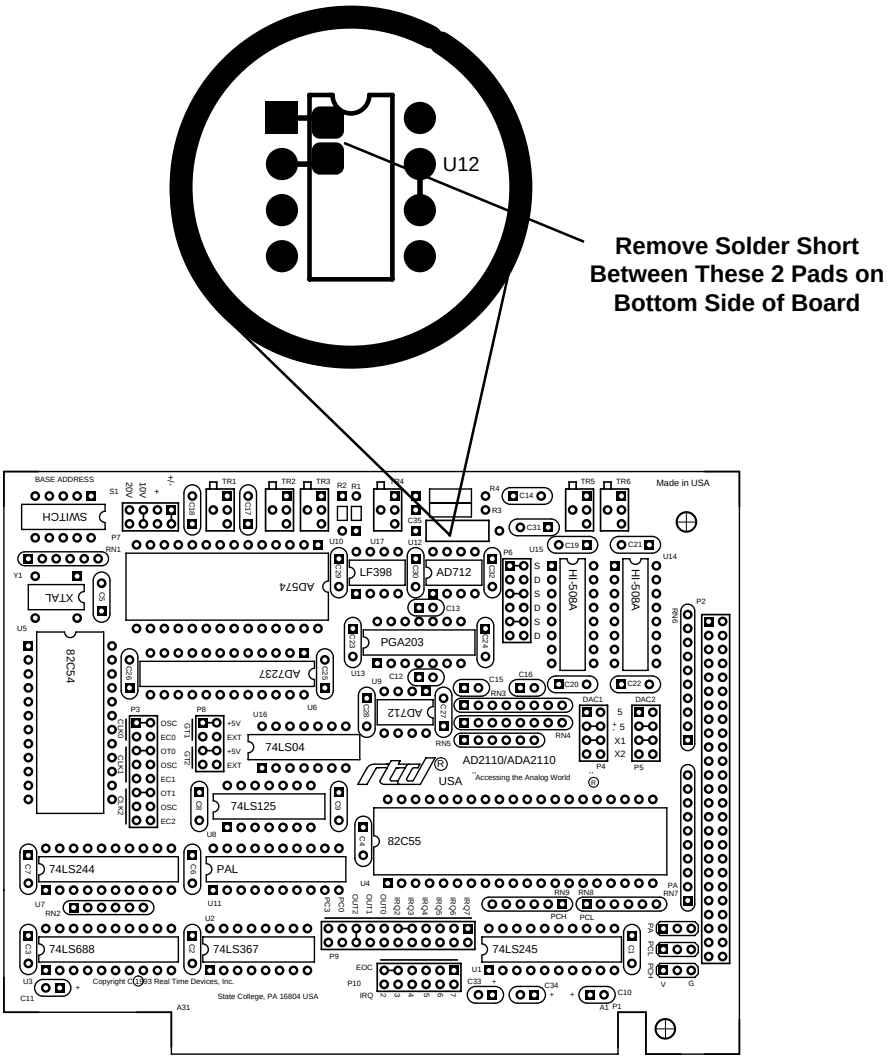


Fig. 1-15 — Diagram for Removal of Solder Short

CHAPTER 2

BOARD INSTALLATION

The 2110 is easy to install in your IBM PC/XT/AT or compatible computer. It can be placed in any slot, short or full-size. This chapter tells you step-by-step how to install and connect the board.

After you have installed the board and made all of your connections, you can turn your system on and run the 2110DIAG board diagnostics program included on your example software disk to verify that your board is working.

Board Installation

Keep the board in its antistatic bag until you are ready to install it in your computer. When removing it from the bag, hold the board at the edges and do not touch the components or connectors.

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

To install the board:

1. Turn OFF the power to your computer.
2. Remove the top cover of the computer housing (refer to your owner's manual if you do not already know how to do this).
3. Select any unused short or full-size expansion slot and remove the slot bracket.
4. Touch the metal housing of the computer to discharge any static buildup and then remove the board from its antistatic bag.
5. Holding the board by its edges, orient it so that its card edge (bus) connector lines up with the expansion slot connector in the bottom of the selected expansion slot.
6. After carefully positioning the board in the expansion slot so that the card edge connector is resting on the computer's bus connector, gently and evenly press down on the board until it is secured in the slot.

   NOTE: Do not force the board into the slot. If the board does not slide into place, remove it and try again. Wiggling the board or exerting too much pressure can result in damage to the board or to the computer.
7. After the board is installed, secure the slot bracket back into place and put the cover back on your computer. The board is now ready to be connected via the external I/O connector at the rear panel of your computer.

External I/O Connections

Figure 2-1 shows the 2110's P2 I/O connector pinout. Refer to this diagram as you make your I/O connections.

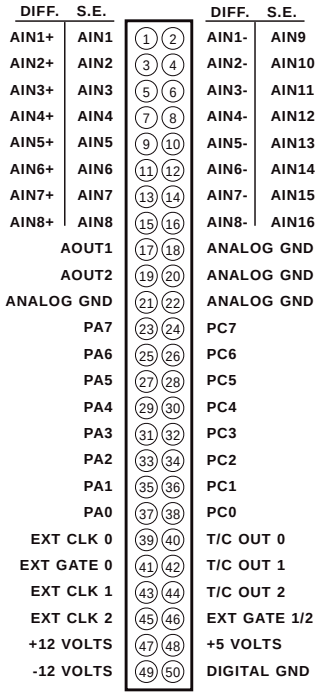


Fig. 2-1 — P2 I/O Connector Pin Assignments

Connecting the Analog Input Pins

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

NOTE: It is good practice to connect all unused channels to ground, as shown in the following diagrams. Failure to do so may affect the accuracy of your results.

Single-Ended. When operating in the single-ended mode, connect the high side of the analog input to one of the analog input channels, AIN1 through AIN16, and connect the low side to an ANALOG GND (pins 18 and 20-22 on P2). Figure 2-2 shows how these connections are made.

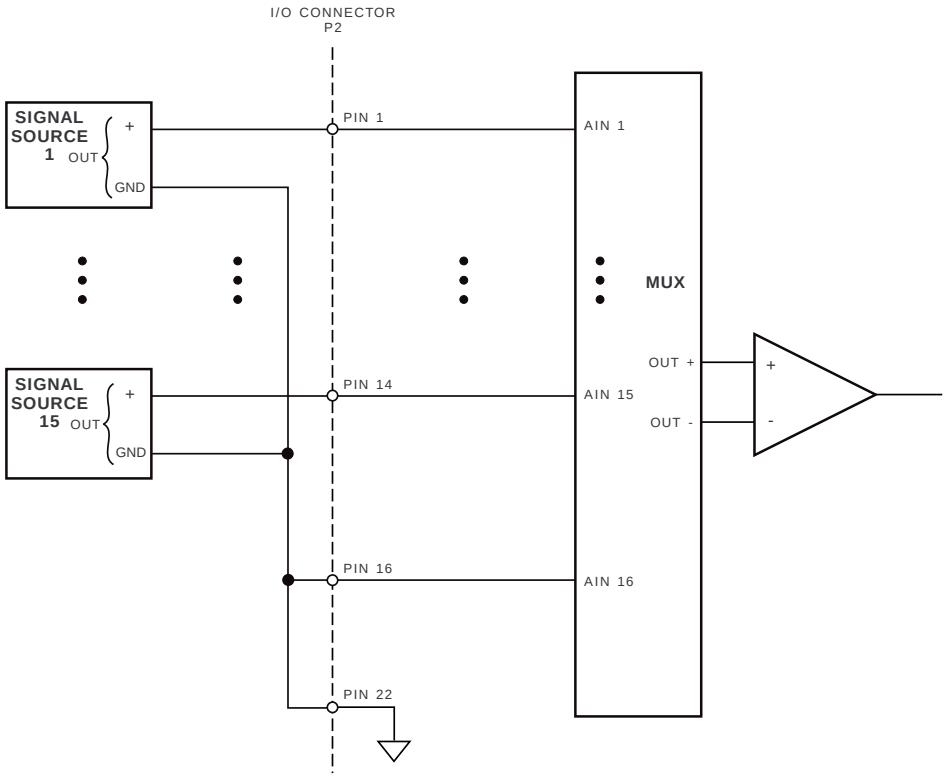


Fig. 2-2 — Single-Ended Input Connections

Differential. When operating in the differential mode, twisted pair cable is recommended to reduce the effects of magnetic coupling at the inputs. Your signal source may or may not have a separate ground reference. When using the differential mode, you should install a resistor pack at location RN6 on the board to provide a reference to ground for signal sources without a separate ground reference.

First, connect the high side of the analog input to the selected analog input channel, AIN1+ through AIN8+, and connect the low side of the input to the corresponding AIN- pin. Then, for signal sources with a separate ground reference, connect the ground from the signal source to an ANALOG GND (pins 18 and 20-22 on P2). Figure 2-3 shows how these connections are made.

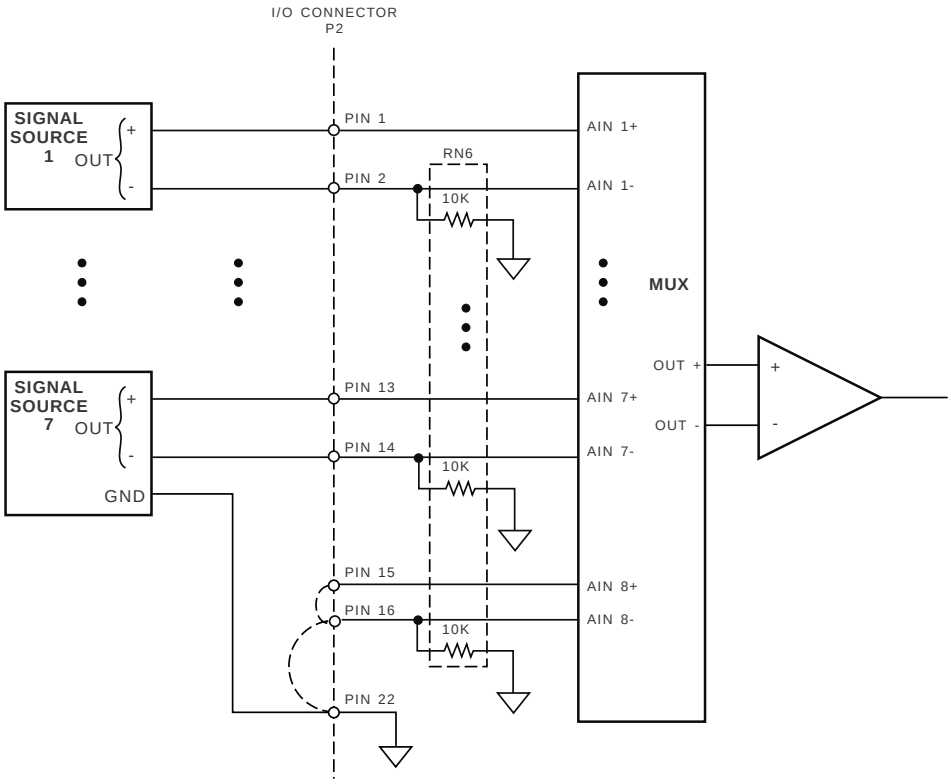


Fig. 2-3 — Differential Input Connections

Connecting the Analog Outputs (ADA 2110 Only)

For each of the two D/A outputs, connect the high side of the device receiving the output to the AOUT channel (P2-17 or P2-19) and connect the low side of the device to an ANALOG GND (P2-18 or P2-20).

Connecting the Timer/Counters and Digital I/O

For all of these connections, the high side of an external signal source or destination device is connected to the appropriate signal pin on the P2 I/O connector and the low side is connected to any DIGITAL GND.

Running the 2110DIAG Diagnostics Program

Now that your board is ready to use, you will want to try it out. An easy-to-use, menu-driven diagnostics program, 2110DIAG, is included with your example software to help you verify your board's operation. You can also use this program to make sure that your current base address setting does not contend with another device.

CHAPTER 3

HARDWARE DESCRIPTION

This chapter describes the features of the 2110 hardware. The major circuits are the A/D, the D/A, the timer/counters, and the digital I/O lines. This chapter also describes the hardware-selectable interrupts.

The 2110 board has four major circuits, the A/D, the D/A (ADA2110 only), the timer/counters, and the digital I/O lines. Figure 3-1 shows the block diagram of the board. This chapter describes the hardware which makes up the major circuits and hardware-selectable interrupts.

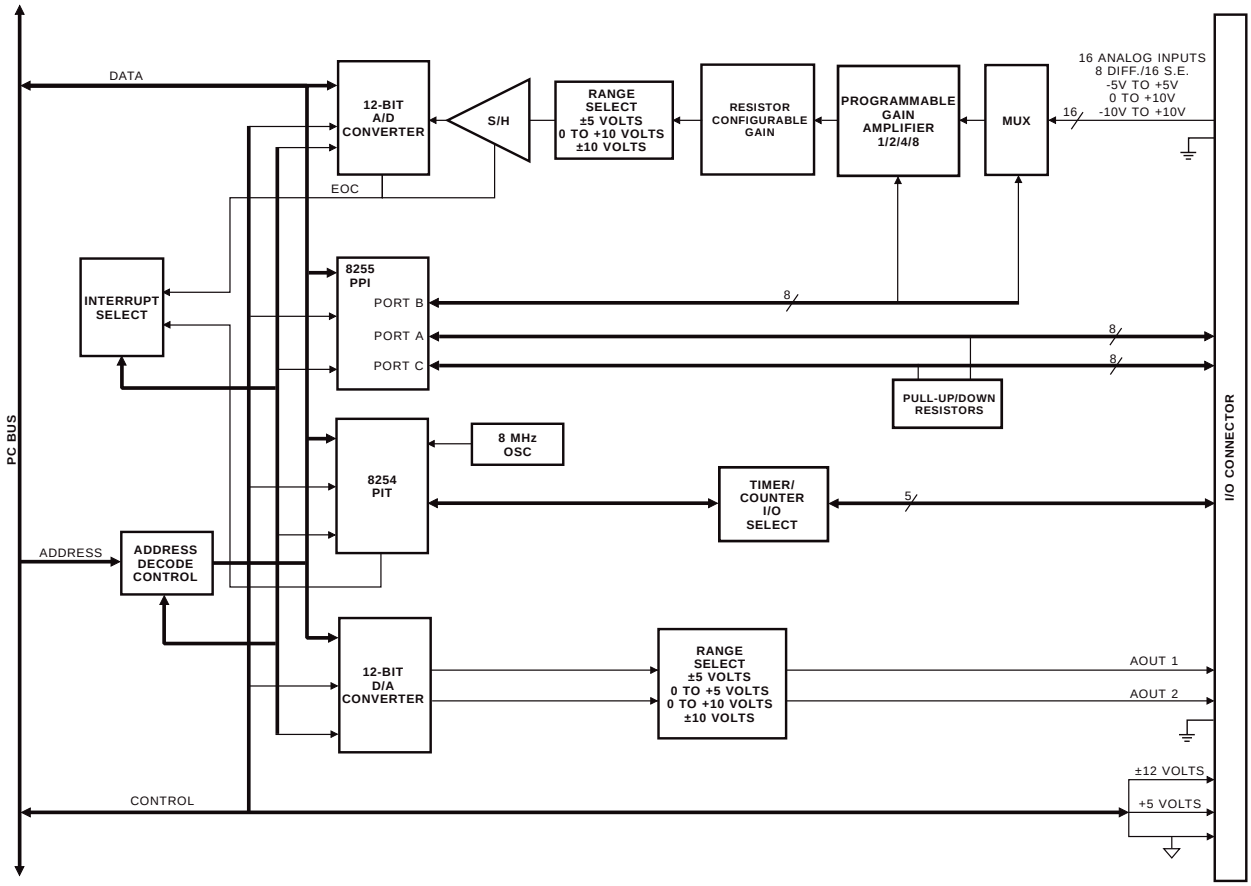


Fig. 3-1 — AD2110/ADA2110 Block Diagram

A/D Conversion Circuitry

The2110 performs analog-to-digital conversions on up to 16 single-ended or 8 differential software-selectable analog input channels. The following paragraphs describe the A/D circuitry.

Analog Inputs

The input voltage range is jumper-selectable for -5 to +5 volts, -10 to +10 volts, or 0 to +10 volts. Software-programmable binary gains of 1, 2, 4, and 8 let you amplify lower level signals to more closely match the board's input ranges. These gains can be customized for even greater input control by adding a gain multiplying resistor circuit as described in Chapter 1. Overvoltage protection to ±35 volts is provided at the inputs.

A/D Converter

The 12-bit A/D converter , when combined with the typical acquisition time of the sample-and-hold circuitry, provides a throughput rate of 40,000 samples per second. The A/D output is a 12-bit data word. Note that 8-bit conversions can be performed when speed is more critical than resolution. Eight-bit conversions increase the throughput rate to about 45 kHz.

D/A Converters (ADA2110 Only)

Two independent 12-bit analog output channels are included on the ADA2110. The analog outputs are generated by two 12-bit D/A converters with independent jumper-selectable output ranges of ±5, ±10, 0 to +5, or 0 to +10 volts. The ±10 volt range has a resolution of 4.88 millivolts, the ±5 and 0 to +10 volt ranges have a resolution of 2.44 millivolts, and the 0 to +5 volt range has a resolution of 1.22 millivolts.

Timer/Counters

An 8254 programmable interval timer provides three 16-bit, 8 MHz timer/counters to support a wide range of timing and counting functions. These timer/counters can be cascaded or used individually for many applications. Figure 3-2 shows the timer/counter circuit block diagram.

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

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

These modes are detailed in the 8254 Data Sheet, reprinted from Intel in Appendix C.

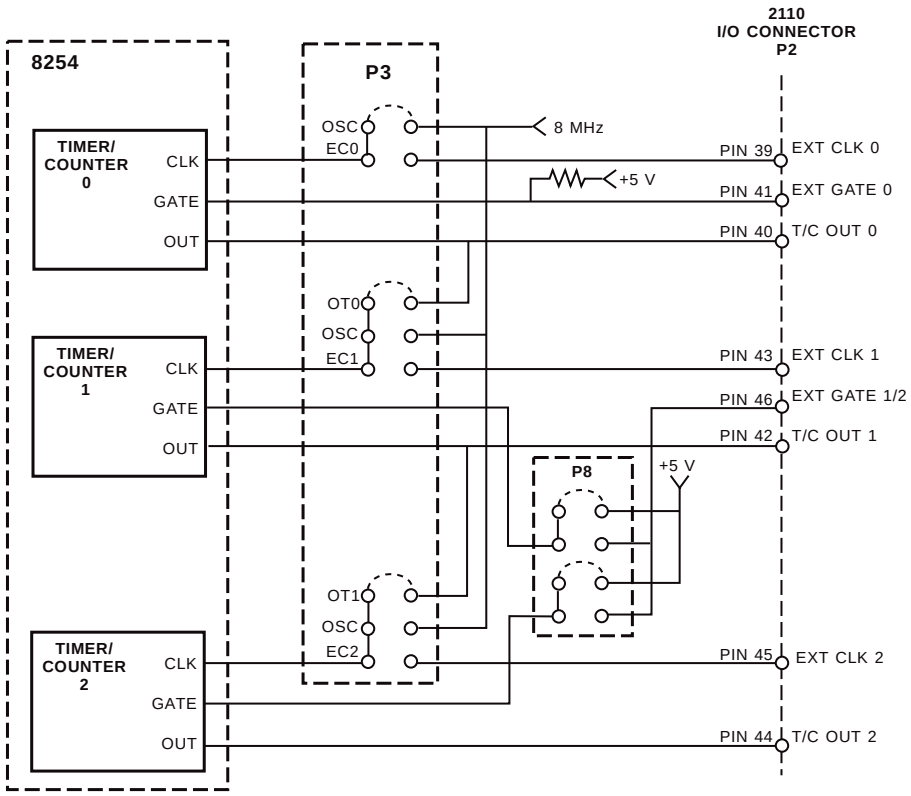


Fig. 3-2 — 8254 Timer/Counter Circuit Block Diagram

Digital I/O, Programmable Peripheral Interface

The programmable peripheral interface (PPI) is used for digital I/O functions. This high-performance TTL/CMOS compatible chip has 24 digital I/O lines divided into two groups of 12 lines each:

Group A — Port A (8 lines) and Port C Upper (4 lines);
Group B — Port B (8 lines) and Port C Lower (4 lines).

Port A and Port C are available at the external I/O connector, P2. Port B is dedicated to on-board functions and is not available for your use. You can use the 16 lines of Ports A and C in one of these three PPI operating modes:

Mode 0 — Basic input/output. Lets you use simple input and output operation for a port. Data is written to or read from the specified port.

Mode 1 — Strobed input/output. Lets you transfer I/O data from Port A in conjunction with strobes or hand-shaking signals.

Mode 2 — Strobed bidirectional input/output. Lets you communicate bidirectionally with an external device through Port A. Handshaking is similar to Mode 1.

These modes are detailed in the 8255 Data Sheet, reprinted from Intel in Appendix C.

Interrupts

The 2110 has five jumper-selectable interrupt sources on P9: timer/counter OUT0, OUT1, and OUT2, and PPI PC0 (INTRB) and PC3 (INTRA). The end-of-convert signal is available as an interrupt on P10 and can be used to interrupt the computer when an A/D conversion is completed. Chapter 1 tells you how to set the jumpers on the interrupt header connectors and Chapter 4 describes how to program interrupts.

CHAPTER 4

BOARD OPERATION AND PROGRAMMING

This chapter shows you how to program and use your 2110 board. It provides a complete description of the I/O map, a detailed description of programming operations and operating modes, and flow diagrams to aid you in programming. The example programs included on the disk in your board package are listed at the end of this chapter. These programs, written in Turbo C, Turbo Pascal, and BASIC, include source code to simplify your applications programming.

Defining the I/O Map

The I/O map for the 2110 is shown in Table 4-1 below. As shown, the board occupies 16 consecutive I/O port locations. The base address (designated as BA) can be selected using DIP switch S1 as described in Chapter 1, *Board Settings*. This switch can be accessed without removing the board from the connector. S1 is factory set at 300 hex (768 decimal). The following sections describe the register contents of each address used in the I/O map.

Table 4-1: AD2110/ADA2110 I/O Map

Register Description	Read Function	Write Function	Address * (Decimal)
8255 PPI Port A	Read Port A digital input lines	Program Port A digital output lines	BA + 0
8255 PPI Port B (Channel/Gain Select)	Read Port B bits	Program channel and gain	BA + 1
8255 PPI Port C	Read Port C digital input lines	Program Port C digital output lines	BA + 2
8255 PPI Control Word	Reserved	Program PPI configuration	BA + 3
8254 Timer/Counter 0	Read count value	Load count register	BA + 4
8254 Timer/Counter 1	Read count value	Load count register	BA + 5
8254 Timer/Counter 2	Read count value	Load count register	BA + 6
8254 Timer/Counter Control Word	Reserved	Program counter mode	BA + 7
Start 12-bit Conversion/ Read MSB	Read A/D converted data, MSB	Start 12-bit A/D conversion	BA + 8
Start 8-bit Conversion/ Read LSB	Read A/D converted data, LSB	Start 8-bit A/D conversion	BA + 9
Read Status/Update DACs	Read status word	Simultaneously update DAC1 & DAC2 (ADA2110 only)	BA + 10
Reserved	Reserved	Reserved	BA + 11
D/A Converter 1 LSB (ADA2110 only)	Reserved	Program DAC1 LSB	BA + 12
D/A Converter 1 MSB (ADA2110 only)	Reserved	Program DAC1 MSB	BA + 13
D/A Converter 2 LSB (ADA2110 only)	Reserved	Program DAC2 LSB	BA + 14
D/A Converter 2 MSB (ADA2110 only)	Reserved	Program DAC2 MSB	BA + 15
* BA = Base Address			

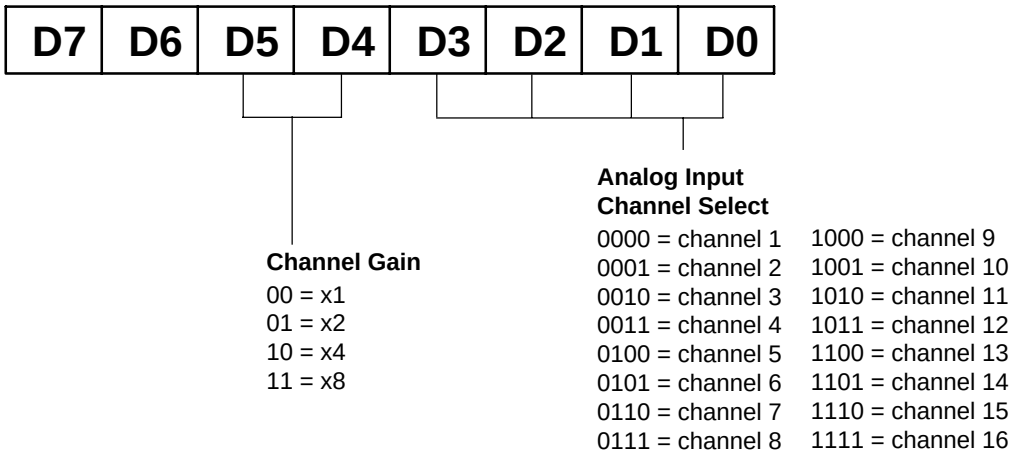
BA + 0: PPI Port A — Digital I/O (Read/Write)

Transfers the 8-bit Port A digital input and digital output data between the board and an external device. A read transfers data from the external device, through P2, and into PPI Port A; a write transfers the written data from Port A through P2 to an external device.

BA + 1: PPI Port B — Channel/Gain Select (Read/Write)

Programs the analog input channel and gain.

Reading this register shows you the current settings.

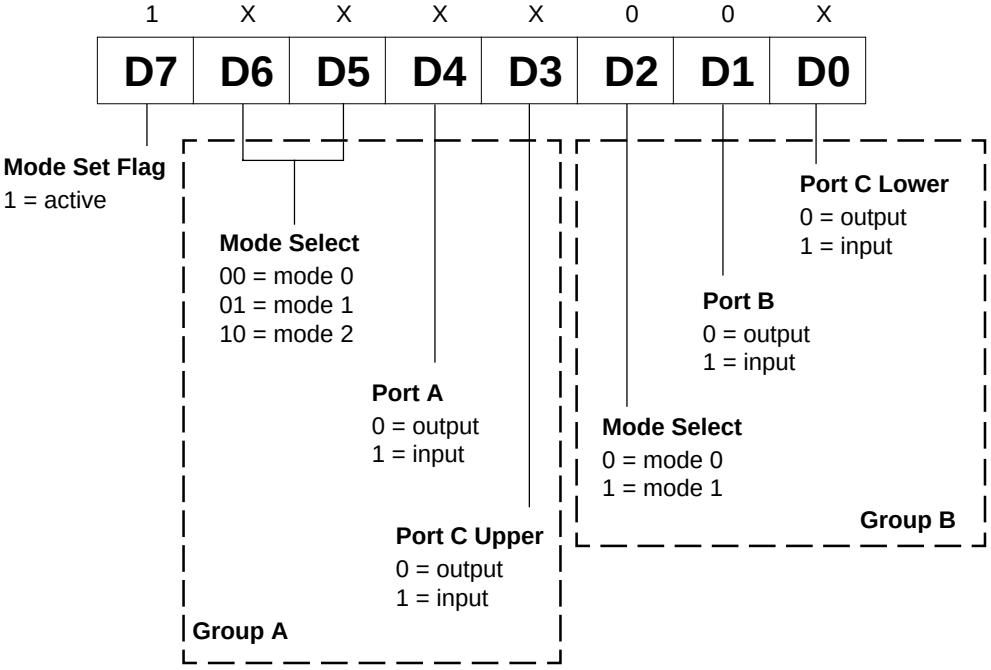


BA + 2: PPI Port C — Digital I/O (Read/Write)

Transfers the two 4-bit Port C digital input and digital output data groups (Port C Upper and Port C Lower) between the board and an external device. A read transfers data from the external device, through P2, and into PPI Port C; a write transfers the written data from Port C through P2 to an external device.

BA + 3: 8255 PPI Control Word (Write Only)

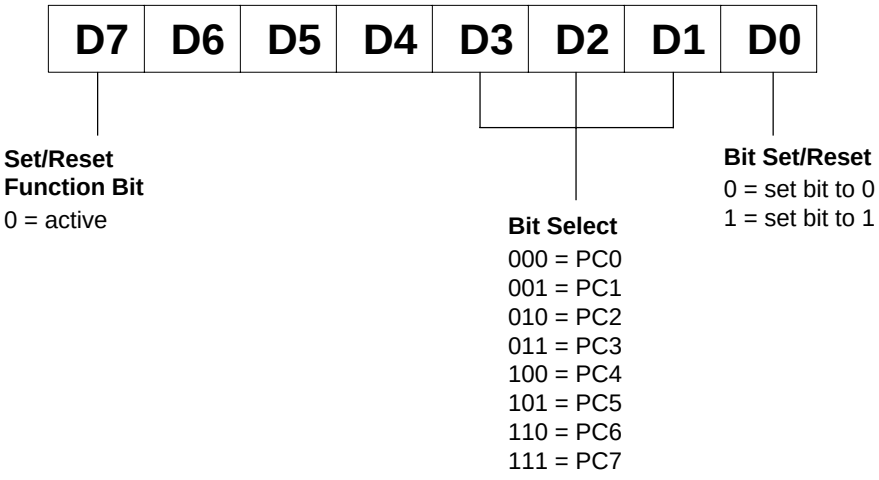
When bit 7 of this word is set to 1, a write programs the PPI configuration. The PPI must be programmed so that Port B is a Mode 0 output port, as shown below (X = don't care).



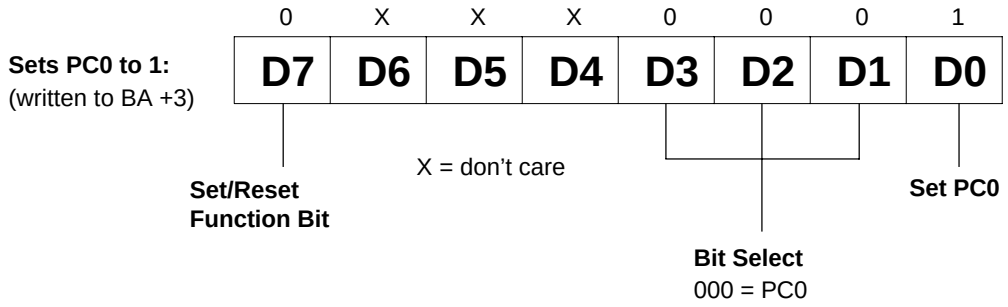
The table below shows the control words for the 16 possible Mode 0 Port I/O combinations.

8255 Port I/O Flow Direction and Control Words, Mode 0						
Group A		Group B		Control Word		
Port A	Port C Upper	Port B	Port C Lower	Binary	Decimal	Hex
Output	Output	Output	Output	1 0 0 0 0 0 0 0	128	80
Output	Output	Output	Input	1 0 0 0 0 0 0 1	129	81
Output	Output	Input	Output	1 0 0 0 0 0 1 0	130	82
Output	Output	Input	Input	1 0 0 0 0 0 1 1	131	83
Output	Input	Output	Output	1 0 0 0 1 0 0 0	136	88
Output	Input	Output	Input	1 0 0 0 1 0 0 1	137	89
Output	Input	Input	Output	1 0 0 0 1 0 1 0	138	8A
Output	Input	Input	Input	1 0 0 0 1 0 1 1	139	8B
Input	Output	Output	Output	1 0 0 1 0 0 0 0	144	90
Input	Output	Output	Input	1 0 0 1 0 0 0 1	145	91
Input	Output	Input	Output	1 0 0 1 0 0 1 0	146	92
Input	Output	Input	Input	1 0 0 1 0 0 1 1	147	93
Input	Input	Output	Output	1 0 0 1 1 0 0 0	152	98
Input	Input	Output	Input	1 0 0 1 1 0 0 1	153	99
Input	Input	Input	Output	1 0 0 1 1 0 1 0	154	9A
Input	Input	Input	Input	1 0 0 1 1 0 1 1	155	9B

When bit 7 of the PPI control word is set to 0, a write can be used to individually program the Port C lines.



For example, if you want to set Port C bit 0 to 1, you would set up the control word so that bit 7 is 0; bits 1, 2, and 3 are 0 (this selects PC0); and bit 0 is 1 (this sets PC0 to 1). The control word is set up like this:



BA + 4: 8254 Timer/Counter 0 (Read/Write)

A read shows the count in the counter, and a write loads the counter with a new value. Counting begins as soon as the count is loaded.

BA + 5: 8254 Timer/Counter 1 (Read/Write)

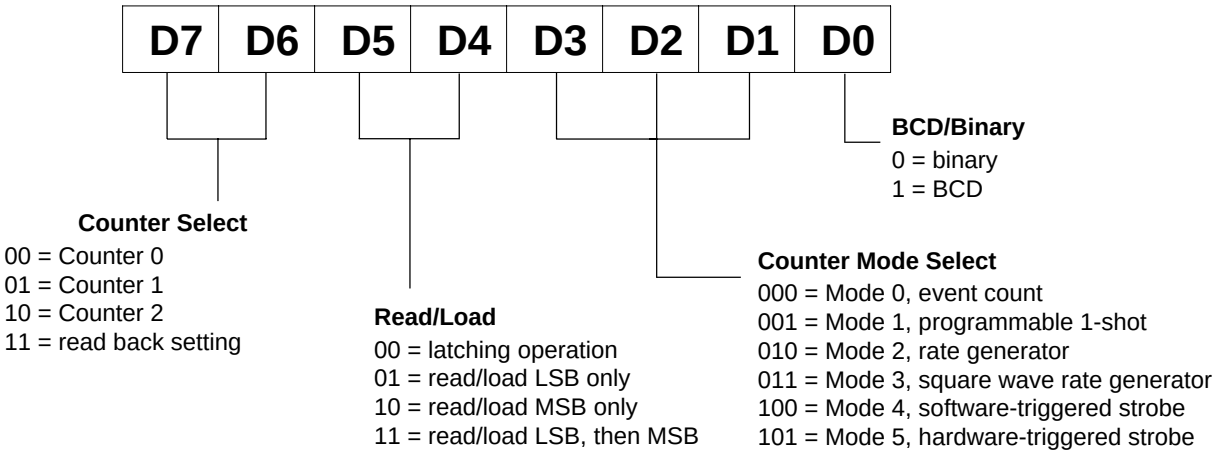
A read shows the count in the counter, and a write loads the counter with a new value. Counting begins as soon as the count is loaded.

BA + 6: 8254 Timer/Counter 2 (Read/Write)

A read shows the count in the counter, and a write loads the counter with a new value. Counting begins as soon as the count is loaded.

BA + 7: 8254 Control Word (Write Only)

Accesses the 8254 control register to directly control the three timer/counters.



BA + 8: Start 12-Bit Conversion/Read MSB Data (Read/Write)

Writing to this address starts a 12-bit A/D conversion (the data written is irrelevant). A read provides the MSB (8 most significant bits) of the A/D conversion, as defined below. The converted data is left-justified. When you are performing 8-bit conversions, only the MSB must be read.

MSB	D7	D6	D5	D4	D3	D2	D1	D0
12-Bit:	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4
8-Bit:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

BA + 9: Start 8-Bit Conversion/Read LSB Data (Read/Write)

Writing to this address starts an 8-bit A/D conversion (the data written is irrelevant). A read provides the LSB (4 least significant bits) of the A/D conversion, as defined below. The converted data is left-justified.

	D7	D6	D5	D4	D3	D2	D1	D0
LSB	Bit 3	Bit 2	Bit 1	Bit 0	X	X	X	X

BA + 10: Read Status/Update DAC Outputs (Read/Write)

A read provides the status bit defined below. The end-of-convert bit goes high when a conversion is complete.

A write simultaneously starts a D/A conversion in both DACs (data written is irrelevant). If the data written to either channel has not been updated since the last conversion, the output of the corresponding DAC will not change.

D7	D6	D5	D4	D3	D2	D1	D0

End-of-Convert
0 = converting
1 = not converting

BA + 11: Reserved

BA + 12: D/A Converter 1 LSB: ADA2110 (Write Only)

Programs the DAC1 LSB (eight bits).

BA + 13: D/A Converter 1 MSB: ADA2110 (Write Only)

Programs the DAC1 MSB (four bits) into D0 through D3; D4 through D7 are irrelevant.

BA + 14: D/A Converter 2 LSB: ADA2110 (Write Only)

Programs the DAC2 LSB (eight bits).

BA + 15: D/A Converter 2 MSB: ADA2110 (Write Only)

Programs the DAC2 MSB (four bits) into D0 through D3; D4 through D7 are irrelevant.

	D7	D6	D5	D4	D3	D2	D1	D0
DAC LSB	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

	D7	D6	D5	D4	D3	D2	D1	D0
DAC MSB	X	X	X	X	Bit 11	Bit 10	Bit 9	Bit 8

Programming the AD2110/ADA2110

This section gives you some general information about programming and the 2110 board, and then walks you through the major 2110 programming functions. These descriptions will help you as you use the example programs included with the board and the programming flow diagrams at the end of this chapter. All of the program descriptions in this section use decimal values unless otherwise specified.

The 2110 is programmed by writing to and reading from the correct I/O port locations on the board. These I/O ports were defined in the previous section. Most high-level languages such as BASIC, Pascal, C, and C++, and of course assembly language, make it very easy to read/write these ports. The table below shows you how to read from and write to I/O ports using some popular programming languages.

Language	Read	Write
BASIC	Data=INP(Address)	OUT Address,Data
Turbo C	Data=inportb(Address)	outportb(Address,Data)
Turbo Pascal	Data:=Port[Address]	Port[Address]:=Data

Assembly	mov dx,Address
	in al,dx
		mov dx,Address
		mov al,Data
		out dx,al

In addition to being able to read/write the I/O ports on the 2110, you must be able to perform a variety of operations that you might not normally use in your programming. The table below shows you some of the operators discussed in this section, with an example of how each is used with Pascal, C, and BASIC. Note that the modulus operator is used to retrieve the least significant byte (LSB) of a two-byte word, and the integer division operator is used to retrieve the most significant byte (MSB).

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

Many compilers have functions that can read/write either 8 or 16 bits from/to an I/O port. For example, Turbo Pascal uses **Port** for 8-bit port operations and **PortW** for 16 bits, Turbo C uses **inportb** for an 8-bit read of a port and **inport** for a 16-bit read. **Be sure to use only 8-bit operations with the 2110!**

Clearing and Setting Bits in a Port

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

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

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

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

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

Example: Set bit 3 in a port. Read in the current value of the port, OR it with 8 (8 = 2^3), and then write the resulting value to the port. In Pascal, this is programmed as:

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

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

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

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

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

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

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

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

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

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

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

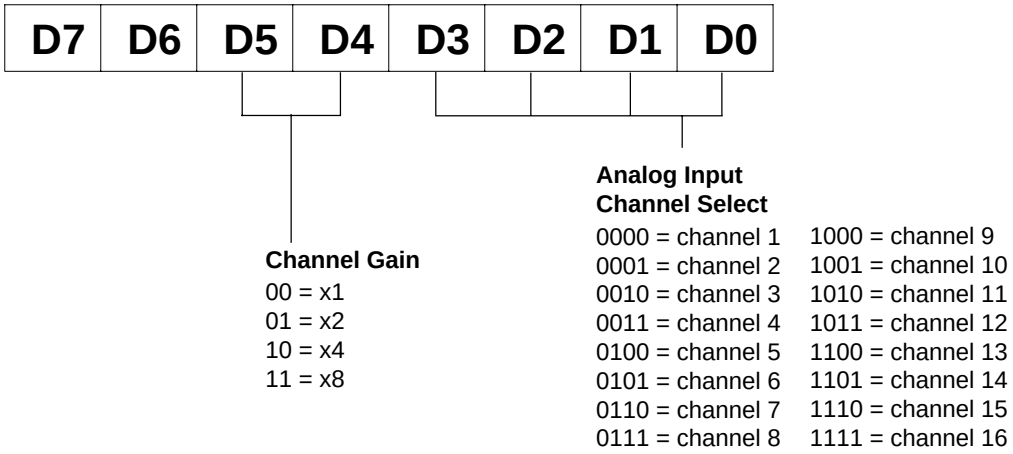
Now that you know how to clear and set bits, we are ready to look at the programming steps for the 2110 board functions.

A/D Conversions

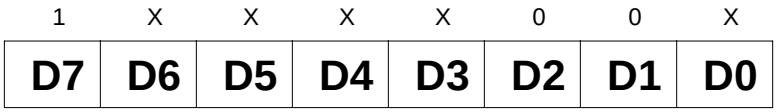
The following paragraphs walk you through the programming steps for performing A/D conversions. Detailed information about the conversion modes is presented in this section. You can follow these steps on the flow diagram at the end of this chapter and in our example programs included with the board. In this discussion, BA refers to the base address.

• Initializing the 8255 PPI

The eight Port B lines of the 8255 PPI control the channel selection & IRQ enable. Port B is programmed at I/O address location BA + 1:



To use Port B for these control functions, the 8255 must be initialized so that Port B is set up as a Mode 0 output port. This is done by writing this data to the PPI control word at I/O address BA + 3 (X = don't care):



• Selecting a Channel

To select a conversion channel, you must assign values to bits 0 through 3 in the PPI Port B port at BA + 1. The table below shows you how to determine the bit settings. Channels 9-16 are available in single-ended operation only. Note that if you do not want to change the gain setting, also programmed through BA + 1, you must preserve it when you set the channel.

X	X	X	X	CH3	CH2	CH1	CH0

BA + 1

Channel	CH3	CH2	CH1	CH0	Channel	CH3	CH2	CH1	CH0
1	0	0	0	0	9	1	0	0	0
2	0	0	0	1	10	1	0	0	1
3	0	0	1	0	11	1	0	1	0
4	0	0	1	1	12	1	0	1	1
5	0	1	0	0	13	1	1	0	0
6	0	1	0	1	14	1	1	0	1
7	0	1	1	0	15	1	1	1	0
8	0	1	1	1	16	1	1	1	1

• Setting the Gain

You may choose among the various levels of programmable gain by setting bits 4 and 5 in the PPI Port B port, BA + 1. The table below shows you how to determine the bit settings for the gain you need. Note that if you do not want to change the channel setting, also programmed through BA + 1, you must preserve it when you set the gain.

x	x	G1	G0	x	x	x	x

BA + 1

Gain	G1	G0
1	0	0
2	0	1
4	1	0
8	1	1

• Starting an A/D Conversion

A/D conversions are started by writing to the appropriate I/O port. For 12-bit conversions, Port BA + 8 is used. For 8-bit conversions, Port BA + 9 is used. A START CONVERT command must be written for each A/D conversion. Figure 4-1 shows the timing diagram for A/D conversions.

• Monitoring Conversion Status

The A/D conversion status can be monitored through the end-of-convert (EOC) signal. This signal, the inverse of the STATUS signal output by the A/D converter, is low when a conversion is in progress and goes high when the conversion is completed. This low-to-high transition can be used to generate an interrupt on P10.

• Reading the Converted Data

The general algorithm for taking an A/D reading is:

1. Start a 12-bit conversion by writing to BA + 8:

```
out base_address+8,0
```

(Note that the value you send is not important. The act of writing to this I/O location is the key to starting a conversion.)

2. Delay at least 20 microseconds or monitor end-of-convert for a transition, or use an interrupt.
3. Read the least significant byte of the converted data from BA + 9:

```
lsb% = inp(base_address% +9)
```

4. Read the most significant byte of the converted data from BA + 8:

```
msb% = inp(base_address% +8)
```

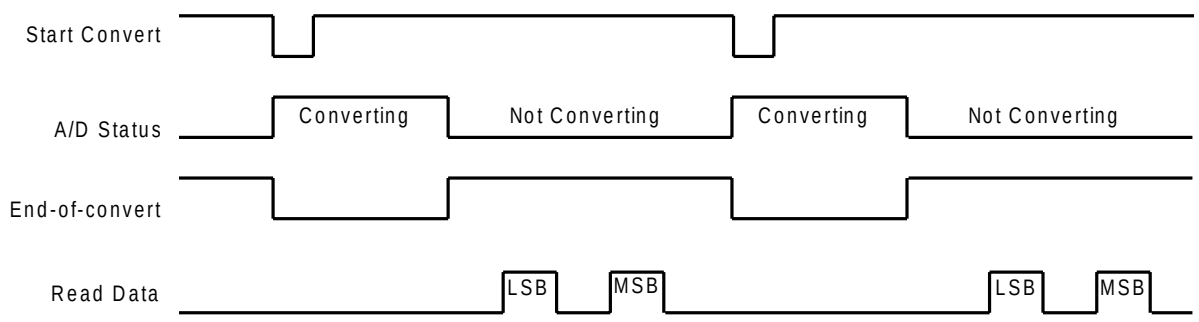


Fig. 4-1 — A/D Conversion Timing Diagram

5. Combine them into the 12-bit result by shifting the LSB four bits to the right. The MSB must also be weighted correctly:

```
result% = (msb% * 16) + (lsb%/16)
```

For a 12-bit conversion, the A/D data read is left justified in a 16-bit word, with the least significant four bits equal to zero. Because of this, the two bytes of A/D data read must be scaled to obtain a valid A/D reading. For example, for a voltage range of ±5 volts, once the reading is calculated, it can be correlated to a voltage value by subtracting 2048 to scale it and then multiplying by 2.4414 millivolts.

For example, if the A/D reading is 1024, the analog input voltage is calculated as follows:

```
(1024 - 2048) bits * 2.4414 mV/bit = -2.49999 volts.
```

Note that 8-bit A/D conversions can also be performed by writing to I/O location BA + 9 to start a conversion. While an 8-bit conversion has a lower resolution, it is performed much more rapidly.

The key digital codes and their input voltage values are given for 12-bit and 8-bit conversions in the following two tables.

12-Bit A/D Code Table

Input Voltage Range			Output Code
0 to +10 Volts	**-10 to +10 Volts**	**-5 to +5 Volts**	
+9.9976 volts	+9.9951 volts	+4.9976 volts	MSB 1111 1111 1111 LSB
+7.500 volts	+5.000 volts	+2.500 volts	1100 0000 0000
+5.000 volts	0 volts	0 volts	1000 0000 0000
+2.500 volts	-5.000 volts	-2.500 volts	0100 0000 0000
0 volts	-10.000 volts	-5.000 volts	0000 0000 0000
For 0 to +10 & ±5 volts, 1 LSB = 2.44 millivolts; for ±10 volts, 1 LSB = 4.88 millivolts.			

8-Bit A/D Code Table

Input Voltage Range			Output Code
0 to +10 Volts	**-10 to +10 Volts**	**-5 to +5 Volts**	
+9.9609 volts	+9.9219 volts	+4.9609 volts	MSB 1111 1111 LSB
+7.500 volts	+5.000 volts	+2.500 volts	1100 0000
+5.000 volts	0 volts	0 volts	1000 0000
+2.500 volts	-5.000 volts	-2.500 volts	0100 0000
0 volts	-10.000 volts	-5.000 volts	0000 0000
For 0 to +10 & ±5 volts, 1 LSB = 39.063 millivolts; for ±10 volts, 1 LSB = 78.126 millivolts.			

Interrupts

• What Is an Interrupt?

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

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

Your 2110 board can interrupt the processor when a variety of conditions are met. By using these interrupts, you can write software that effectively deals with real world events.

• Interrupt Request Lines

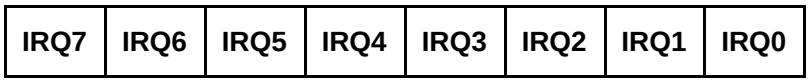
To allow different peripheral devices to generate interrupts on the same computer, the PC bus has eight different interrupt request (IRQ) lines. A transition from low to high on one of these lines generates an interrupt request which is handled by the PC's interrupt controller. The interrupt controller checks to see if interrupts are to be acknowledged from that IRQ and, if another interrupt is already in progress, it decides if the new request should supersede the one in progress or if it has to wait until the one in progress is done. This prioritizing allows an interrupt to be interrupted if the second request has a higher priority. The priority level is based on the number of the IRQ; IRQ0 has the highest priority, IRQ1 is second-highest, and so on through IRQ7, which has the lowest. Many of the IRQs are used by the standard system resources. IRQ0 is used by the system timer, IRQ1 is used by the keyboard, IRQ3 by COM2, IRQ4 by COM1, and IRQ6 by the disk drives. Therefore, it is important for you to know which IRQ lines are available in your system for use by the 2110 board.

• 8259 Programmable Interrupt Controller

The chip responsible for handling interrupt requests in the PC is the 8259 Programmable Interrupt Controller. To use interrupts, you need to know how to read and set the 8259's interrupt mask register (IMR) and how to send the end-of-interrupt (EOI) command to the 8259.

• Interrupt Mask Register (IMR)

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



I/O Port 21H

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

• End-of-Interrupt (EOI) Command

After an interrupt service routine is complete, the 8259 interrupt controller must be notified. This is done by writing the value 20H to I/O port 20H.

• What Exactly Happens When an Interrupt Occurs?

Understanding the sequence of events when an interrupt is triggered is necessary to properly write software interrupt handlers. When an interrupt request line is driven high by a peripheral device (such as the 2110), the

interrupt controller checks to see if interrupts are enabled for that IRQ, and then checks to see if other interrupts are active or requested and determines which interrupt has priority. The interrupt controller then interrupts the processor. The current code segment (CS), instruction pointer (IP), and flags are pushed on the stack for storage, and a new CS and IP are loaded from a table that exists in the lowest 1024 bytes of memory. This table is referred to as the interrupt vector table and each entry is called an interrupt vector. Once the new CS and IP are loaded from the interrupt vector table, the processor begins executing the code located at CS:IP. When the interrupt routine is completed, the CS, IP, and flags that were pushed on the stack when the interrupt occurred are now popped from the stack and execution resumes from the point where it was interrupted.

• Using Interrupts in Your Programs

Adding interrupts to your software is not as difficult as it may seem, and what they add in terms of performance is often worth the effort. Note, however, that although it is not that hard to use interrupts, the smallest mistake will often lead to a system hang that requires a reboot. This can be both frustrating and time-consuming. But, after a few tries, you'll get the bugs worked out and enjoy the benefits of properly executed interrupts. In addition to reading the following paragraphs, study the INTRPTS source code included on your 2110 program disk for a better understanding of interrupt program development.

• Writing an Interrupt Service Routine (ISR)

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

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

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

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

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

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

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

Your ISR should have this structure:

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

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

In C:

```
void interrupt ISR(void)
{
   /* Your code goes here. Do not use any DOS functions! */
   outportb(0x20, 0x20);            /* Send EOI command to 8259 */
}
```

In Pascal:

```
Procedure ISR; Interrupt;
begin
   { Your code goes here. Do not use any DOS functions! }
   Port[$20] := $20;                { Send EOI command to 8259 }
end;
```

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

The next step after writing the ISR is to save the startup state of the interrupt mask register and the interrupt vector that you will be using. The IMR is located at I/O port 21H. The interrupt vector you will be using is located in the interrupt vector table which is simply an array of 256-bit (4-byte) pointers and is located in the first 1024 bytes of memory (Segment = 0, Offset = 0). You can read this value directly, but it is a better practice to use DOS function 35H (get interrupt vector). Most C and Pascal compilers provide a library routine for reading the value of a vector. The vectors for the hardware interrupts are vectors 8 through 15, where IRQ0 uses vector 8, IRQ1 uses vector 9, and so on. Thus, if the 2110 will be using IRQ3, you should save the value of interrupt vector 11.

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

With the startup IMR saved and the interrupts on your IRQ temporarily disabled, you can assign the interrupt vector to point to your ISR. Again, you can overwrite the appropriate entry in the vector table with a direct memory write, but this is a bad practice. Instead, use either DOS function 25H (set interrupt vector) or, if your compiler provides it, the library routine for setting an interrupt vector. Remember that vector 8 is for IRQ0, vector 9 is for IRQ1, and so on.

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

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

• Restoring the Startup IMR and Interrupt Vector

Before exiting your program, you must restore the interrupt mask register and interrupt vectors to the state they were in when your program started. To restore the IMR, write the value that was saved when your program started to I/O port 21H. Restore the interrupt vector that was saved at startup with either DOS function 35H (get interrupt vector), or use the library routine supplied with your compiler. Performing these two steps will guarantee that the interrupt status of your computer is the same after running your program as it was before your program started running.

• Common Interrupt Mistakes

- Remember that hardware interrupts are numbered 8 through 15, even though the corresponding IRQs are numbered 0 through 7.
- The most common mistake when writing an ISR is forgetting to issue the EOI command to the 8259 interrupt controller before exiting the ISR.

D/A Conversions (ADA2110 Only)

The two D/A converters can be individually programmed to convert 12-bit digital words into a voltage in the range of ±5, ±10, 0 to +5, or 0 to +10 volts. DAC 1 is programmed by writing the LSB to BA + 12 and the MSB to BA + 13. DAC 2 is programmed by writing the LSB to BA + 14 and the MSB to BA + 15. The following tables list the key digital codes and corresponding output voltages for the D/A converters.

D/A Converter Unipolar Calibration Table

	Ideal Output Voltage (in millivolts)	
D/A Bit Weight	0 to +5 V	0 to +10 V
4095 (Max. Output)	4998.8	9997.6
2048	2500.0	5000.0
1024	1250.0	2500.0
512	625.00	1250.0
256	312.50	625.00
128	156.250	312.50
64	78.125	156.250
32	39.063	78.125
16	19.5313	39.063
8	9.7656	19.5313
4	4.8828	9.7656
2	2.4414	4.8828
1	1.2207	2.4414
0	0.0000	0.0000

D/A Converter Bipolar Calibration Table		
	Ideal Output Voltage (in millivolts)	
D/A Bit Weight	±5 V	±10 V
4095 (Max. Output)	+4997.6	+9995.1
2048	0.0	0.0
1024	-2500.0	-5000.0
512	-3750.0	-7500.0
256	-4375.0	-8750.0
128	-4687.5	-9375.0
64	-4843.8	-9687.5
32	-4921.9	-9843.8
16	-4960.9	-9921.9
8	-4980.5	-9960.9
4	-4990.2	-9980.5
2	-4995.1	-9990.2
1	-4997.6	-9995.1
0	-5000.0	-10000.0

Timer/Counters

An 8254 programmable interval timer provides three 16-bit, 8-MHz timer/counters for timing and counting functions such as frequency measurement, event counting, and interrupts. All three timer/counters are cascaded at the factory. Figure 4-2 shows the timer/counter circuitry.

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

One of two clock sources, the on-board 8-MHz crystal or an external clock can be selected as the clock input to each timer/counter. In addition, the timer/counters can be cascaded by connecting TC0's output to TC1's clock input and TC1's output to TC2's clock input. The diagram shows how these clock sources are connected to the timer/counters.

An external gate source can be connected to each timer/counter through the I/O connector and P8 (TC1 and TC2's gate input jumper, P8). When TC0's gate is disconnected, an on-board pull-up resistor automatically pulls the gate high, enabling the timer/counter. TC1 and TC2 can be jumpered to +5 volts at P8.

The output from each timer/counter is available at the I/O connector, where it can be used for interrupt generation or for counting functions.

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

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

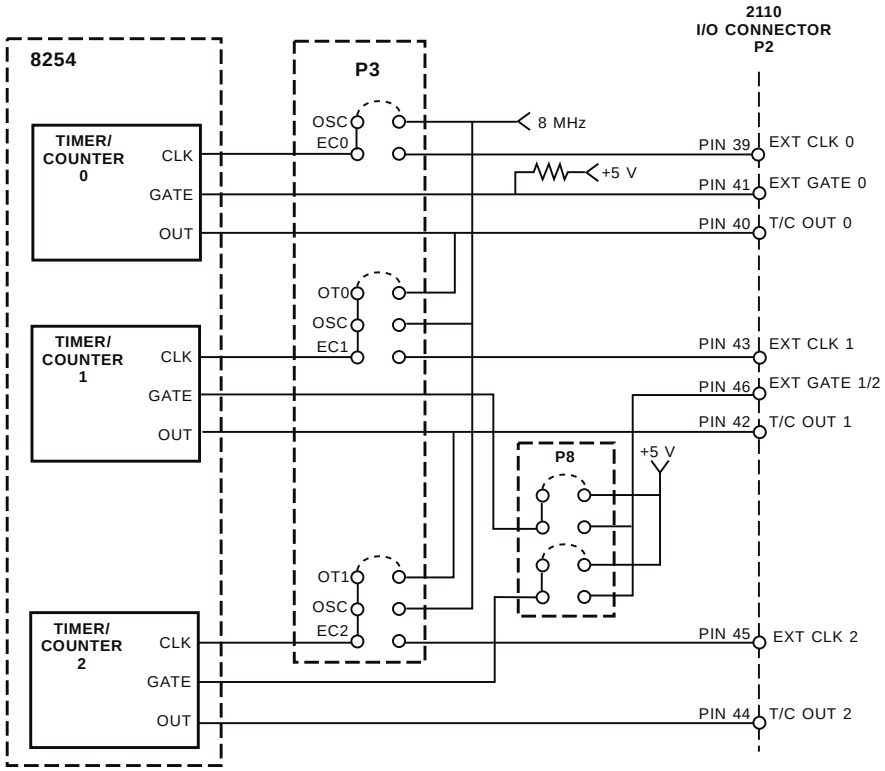


Fig. 4-2 — 8254 Programmable Interval Timer Circuit Block Diagram

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

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

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

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

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

Digital I/O

The 16 8255 PPI-based digital I/O lines can be used to transfer data between the computer and external devices. The digital input lines of Ports A and C can have pull-up or pull-down resistors installed, as described in Chapter 1.

Example Programs and Flow Diagrams

Included with the 2110 is a set of example programs that demonstrate the use of many of the board's features. These examples are in written in C, Pascal, and BASIC. Also included is an easy-to-use menu-driven diagnostics program, 2110DIAG, which is especially helpful when you are first checking out your board after installation and when calibrating the board (Chapter 5).

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

C and Pascal Programs

These programs are source code files so that you can easily develop your own custom software for your 2110. In the C directory, 2110.H and 2110.INC contain all the functions needed to implement the main C programs. H defines the addresses and INC contains the routines called by the main programs. In the Pascal directory, 2110.PNC contains all of the procedures needed to implement the main Pascal programs.

Analog-to-Digital:

SOFTTRIG	Demonstrates how to use a trigger for acquiring data.

Timer/Counters:

TIMER	A short program demonstrating how to program the 8254 for use as a timer.

Digital I/O:

DIGITAL	Simple program that shows how to read and write the digital I/O lines.

Digital-to-Analog:

DAC	Shows how to use the DACs. Uses A/D channel 1 to monitor the output of DAC1.
WAVES	A more complex program that shows how to use the 8254 timer and the DACs as a waveform generator.

Interrupts:

INTRPTS	Shows the bare essentials required for using interrupts.
INTSTR	A complete program showing interrupt-based streaming to disk.

BASIC Programs

These programs are source code files so that you can easily develop your own custom software for your 2110.

Analog-to-Digital:

SINGLE	Demonstrates how to perform single conversions.
SCAN	Demonstrates how to change channels while acquiring data.

Timer/Counters:

TIMER	A short program demonstrating how to program the 8254 for use as a timer.

Digital I/O:

DIGITAL	Simple program that shows how to read and write the digital I/O lines.

Digital-to-Analog:

DASCAN	Demonstrates D/A conversion.

Flow Diagrams

The following paragraphs provide a description and flow diagram for the 2110's A/D and D/A conversion functions. These diagrams will help you to build your own custom application programs.

• Single Convert Flow Diagram (Figure 4-3)

This flow diagram shows you the steps for taking a single sample on a selected channel. A sample is taken each time you send the Start Convert command. All of the samples will be taken on the same channel and until you change the value in the PPI Port B register (BA + 1). Changing this value before each Start Convert command is issued lets you take the next reading from a different channel.

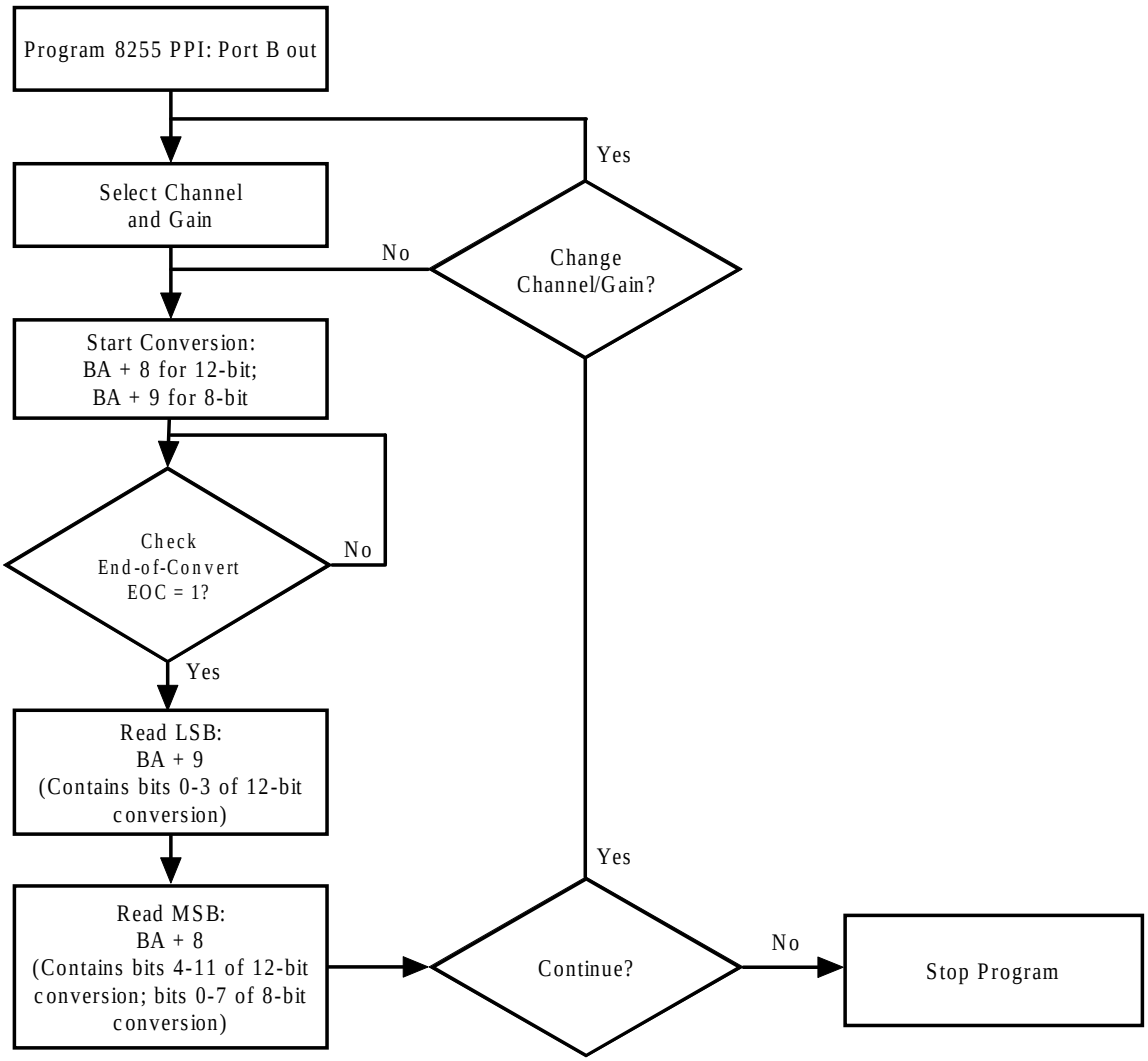


Fig. 4-3 — Single Conversion Flow Diagram

• D/A Conversion Flow Diagram (Figure 4-4)

This flow diagram shows you how to generate a voltage output through the D/A converter (ADA2110 only). The outputs of both converters are updated each time an update command is issued by writing to BA + 10.

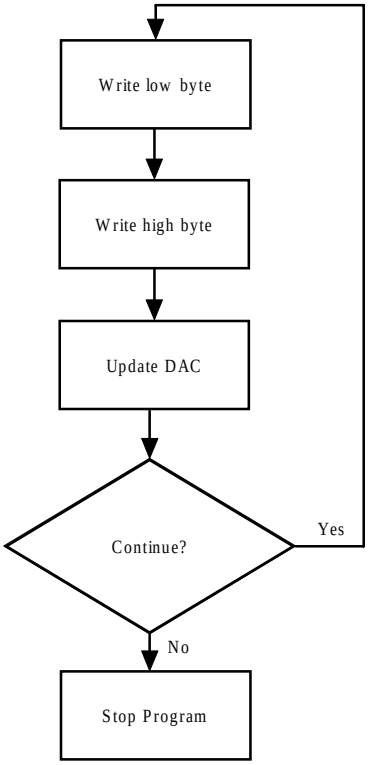


Fig. 4-4 — D/A Conversion Flow Diagram

CHAPTER 5

CALIBRATION

This chapter tells you how to calibrate the 2110 using the 2110DIAG calibration program included in the example software package and five trimpots on the board. These trimpots calibrate the A/D converter gain and offset and the D/A X2 multiplier output.

This chapter tells you how to calibrate the A/D converter gain and offset and the D/A converter X2 multiplier (ADA2110 only). All A/D and D/A ranges are factory-calibrated before shipping. Any time you suspect inaccurate readings, you can check the accuracy of your conversions using the procedure below, and make adjustments as necessary. Using the 2110DIAG diagnostics program is a convenient way to monitor conversions while you calibrate the board.

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

Required Equipment

The following equipment is required for calibration:

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

While not required, the 2110DIAG diagnostics program (included with example software) is helpful when performing calibrations. Figure 5-1 shows the board layout with the trimpots located along the top edge of the board.

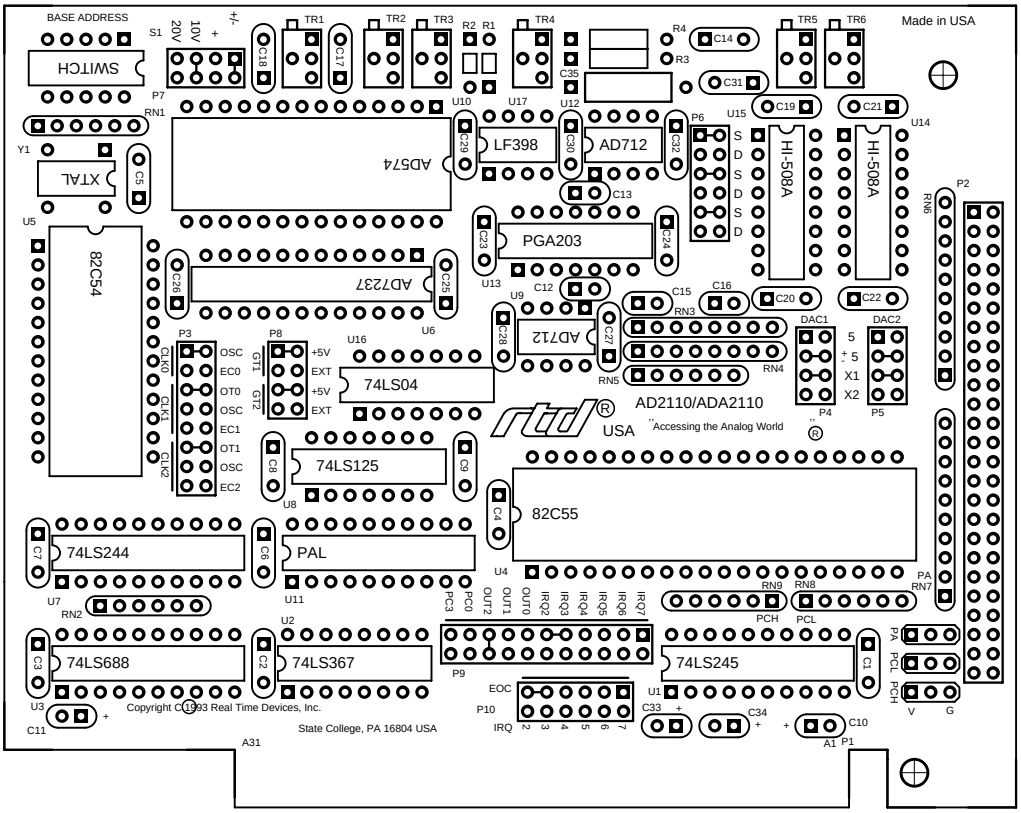


Fig. 5-1 — Board Layout

A/D Calibration

Two procedures are used to calibrate the A/D converter for all input voltage ranges. The first procedure calibrates the converter for the unipolar range (0 to +10 volts), and the second procedure calibrates the bipolar ranges (±5, ±10 volts). Table 5-1 shows the ideal input voltage for each bit weight for all three input ranges.

Unipolar Calibration

Two adjustments are made to calibrate the A/D converter for the unipolar range of 0 to +10 volts. One is the offset adjustment, and the other is the full scale, or gain, adjustment. Trimpot TR3 is used to make the offset adjustment, and trimpot TR1 is used for gain adjustment. This calibration procedure is performed with the board set up for a 0 to +10 volt input range. Before making these adjustments, make sure that the jumpers on P7 are set for this range.

Use analog input channel 1 and set it for a gain of 1 while calibrating the board. Connect your precision voltage source to channel 1. Set the voltage source to +1.22070 millivolts, start a conversion, and read the resulting data. Adjust trimpot TR3 until it flickers between the values listed in the table at the top of the next page. Next, set the voltage to +9.49829 volts, and repeat the procedure, this time adjusting TR1 until the data flickers between the values in the table. Note that the value used to adjust the full scale voltage is not the ideal full scale value for a 0 to +10 volt input range. This value is used because it is the maximum value at which the A/D converter is guaranteed to be linear, and ensures accurate calibration results.

Table 5-1: A/D Converter Bit Weights for All Input Ranges

A/D Bit Weight	Ideal Input Voltage (millivolts)		
	-5 to +5 Volts	-10 to +10 Volts	0 to +10 Volts
4095 (full-scale)	+4997.6	+9995.1	+9997.6
2048	0000.0	0000.0	+5000.0
1024	-2500.0	-5000.0	+2500.0
512	-3750.0	-7500.0	+1250.0
256	-4375.0	-8750.0	+625.00
128	-4687.5	-9375.0	+312.50
64	-4843.8	-9687.5	+156.250
32	-4921.9	-9843.8	+78.125
16	-4960.9	-9921.9	+39.063
8	-4980.5	-9960.9	+19.5313
4	-4990.2	-9980.5	+9.7656
2	-4995.1	-9990.2	+4.8828
1	-4997.6	-9995.1	+2.4414
0	-5000.0	-10000.0	+0.0000

Data Values for Calibrating Unipolar Range (0 to +10 volts)		
	Offset (TR3) **Input Voltage = +1.22070 mV**	**Converter Gain (TR2)** **Input Voltage = +9.49829 V**
A/D Converted Data	0000 0000 0000 0000 0000 0001	1111 0011 0010 1111 0011 0011

Bipolar Calibration

Two adjustments are made to calibrate the A/D converter for the bipolar ranges of ±5 and ±10 volts. One is the offset adjustment, and the other is the full scale, or gain, adjustment. Trimpot TR2 is used to make the offset adjustment, and trimpot TR1 is used for gain adjustment. These adjustments are made with the board set for a range of -5 to +5 volts. Before making these adjustments, make sure that the jumpers on P7 are set for this range.

Use analog input channel 1 and set it for a gain of 1 while calibrating the board. Connect your precision voltage source to channel 1. Set the voltage source to -4.99878 volts, start a conversion, and read the resulting data. Adjust trimpot TR2 until it flickers between the values listed in the table below. Next, set the voltage to +4.99634 volts, and repeat the procedure, this time adjusting TR1 until the data flickers between the values in the table.

Data Values for Calibrating Bipolar Ranges (Using -5 to +5 volts)		
	Offset (TR2) **Input Voltage = -4.99878V**	**Converter Gain (TR1)** **Input Voltage = +4.99634V**
A/D Converted Data	0000 0000 0000 0000 0000 0001	1111 1111 1110 1111 1111 1111

D/A Calibration (ADA2110)

The D/A converter requires no calibration for the X1 ranges (0 to +5 and ±5 volts). The following paragraph describes the calibration procedure for the X2 multiplier ranges.

To calibrate for X2 (0 to +10 or ±10 volts), set the DAC output voltage range to 0 to +10 volts (jumpers on X2 and 5 on P4, AOUT1, or P5, AOUT2). Then, program the corresponding D/A converter (DAC1 or DAC2) with the digital value 2048. The ideal DAC output for 2048 at X2 (0 to +10 volt range) is 5.0000 volts. Adjust TR5 for AOUT1 and TR6 for AOUT2 until 5.0000 volts is read at the output. Table 5-2 lists the ideal output voltages per bit weight for unipolar ranges and Table 5-3 lists the ideal output voltages for bipolar ranges.

Table 5-2: D/A Converter Unipolar Calibration Table		
	Ideal Output Voltage (in millivolts)	
D/A Bit Weight	0 to +5 V	0 to +10 V
4095 (Max. Output)	4998.8	9997.6
2048	2500.0	5000.0
1024	1250.0	2500.0
512	625.00	1250.0
256	312.50	625.00
128	156.250	312.50
64	78.125	156.250
32	39.063	78.125
16	19.5313	39.063
8	9.7656	19.5313
4	4.8828	9.7656
2	2.4414	4.8828
1	1.2207	2.4414
0	0.0000	0.0000

Table 5-3: D/A Converter Bipolar Calibration Table		
	Ideal Output Voltage (in millivolts)	
D/A Bit Weight	±5 V	±10 V
4095 (Max. Output)	+4997.6	+9995.1
2048	0.0	0.0
1024	-2500.0	-5000.0
512	-3750.0	-7500.0
256	-4375.0	-8750.0
128	-4687.5	-9375.0
64	-4843.8	-9687.5
32	-4921.9	-9843.8
16	-4960.9	-9921.9
8	-4980.5	-9960.9
4	-4990.2	-9980.5
2	-4995.1	-9990.2
1	-4997.6	-9995.1
0	-5000.0	-10000.0

APPENDIX A

2110 SPECIFICATIONS

AD2110/ADA2110 Characteristics Typical @ 25° C

Interface

IBM PC/XT/AT compatible
Switch-selectable base address, I/O mapped
Jumper-selectable interrupts

Analog Input

16 single-ended or 8 differential inputs
Input impedance, each channel ..... >10 megohms
Gain ..... 1, 2, 4, 8 plus Gm gain multiplier
Input ranges ..... ±5, ±10, or 0 to +10 volts
Guaranteed linearity across input ranges ..... ±5, ±9.5, and 0 to +9.5 volts
Overvoltage protection ..... ±35 Vdc
Settling time (gain = 1) ..... 5 µsec, max

A/D Converter AD574

Type ..... Successive approximation
Resolution ..... 12 bits (2.44 mV @ 10V; 4.88 mV @ 20V)
Linearity ..... ±1 bit, typ
Conversion speed ..... 20 µsec, typ
Sample-and-hold acquisition time ..... 5 µsec, typ
Throughput ..... 40 kHz

Digital I/O ..... CMOS 82C55

Number of lines 16
Logic compatibility ..... TTL/CMOS
(Configurable with optional I/O pull-up/pull-down resistors)
High-level output voltage ..... 4.2V, min
Low-level output voltage ..... 0.45V, max
High-level input voltage ..... 2.2V, min; 5.5V, max
Low-level input voltage ..... -0.3V, min; 0.8V, max
Input load current ..... ±10 µA
Input capacitance,
C(IN)@F=1MHz ..... 10 pF
Output capacitance,
C(OUT)<@F=1MHz ..... 20 pF

D/A Converter (ADA2110 Only) AD7237

Analog outputs ..... 2 channels
Resolution ..... 12 bits
Output ranges ..... 0 to +5, ±5, or 0 to +10 volts
Guaranteed linearity across output ranges ..... 0 to +5, ±5, and 0 to +9.2 volts
Relative accuracy ..... ±1 bit, max
Full-scale accuracy ..... ±5 bits, max
Non-linearity ..... ±1 bit, max
Settling time ..... 10 µsec, max

Timer/Counters ..... CMOS 82C54

Three 16-bit down counters (2 cascaded, 1 independent)
6 programmable operating modes
Counter input source ..... External clock (8 MHz, max) or on-board 8-MHz clock
Counter outputs ..... Available externally; used as PC interrupts
Counter gate source ..... External gate or always enabled

Miscellaneous Inputs/Outputs (PC bus-sourced)

±5 volts, ±12 volts, ground

Current Requirements

140 mA @ +5 volts; 32 mA @ +12 volts; 30 mA @ –12 volts

Connector

50-pin right angle shrouded header

Size

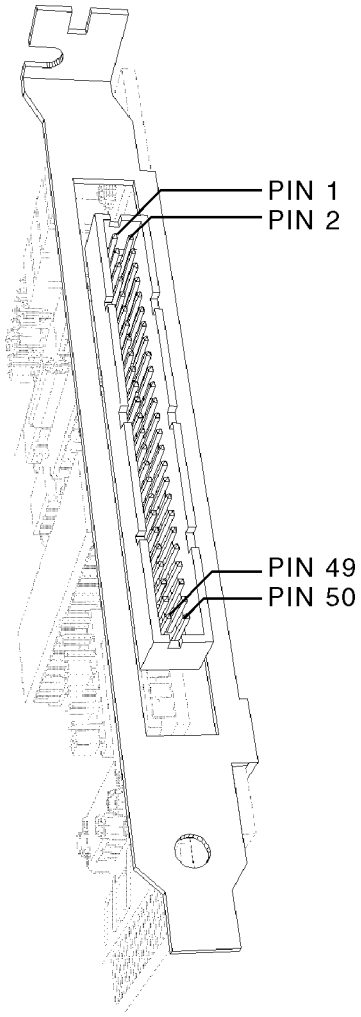
Short slot — 3.875"H x 5.25"W (99mm x 134mm)

APPENDIX B

P2 CONNECTOR PIN ASSIGNMENTS

P2 Connector:

DIFF.	S.E.			DIFF.	S.E.
AIN1+	AIN1	1	2	AIN1-	AIN9
AIN2+	AIN2	3	4	AIN2-	AIN10
AIN3+	AIN3	5	6	AIN3-	AIN11
AIN4+	AIN4	7	8	AIN4-	AIN12
AIN5+	AIN5	9	10	AIN5-	AIN13
AIN6+	AIN6	11	12	AIN6-	AIN14
AIN7+	AIN7	13	14	AIN7-	AIN15
AIN8+	AIN8	15	16	AIN8-	AIN16
	AOUT1	17	18	ANALOG GND	
	AOUT2	19	20	ANALOG GND	
	ANALOG GND	21	22	ANALOG GND	
	PA7	23	24	PC7	
	PA6	25	26	PC6	
	PA5	27	28	PC5	
	PA4	29	30	PC4	
	PA3	31	32	PC3	
	PA2	33	34	PC2	
	PA1	35	36	PC1	
	PA0	37	38	PC0	
	EXT CLK 0	39	40	T/C OUT 0	
	EXT GATE 0	41	42	T/C OUT 1	
	EXT CLK 1	43	44	T/C OUT 2	
	EXT CLK 2	45	46	EXT GATE 1/2	
	+12 VOLTS	47	48	+5 VOLTS	
	-12 VOLTS	49	50	DIGITAL GND	



P2 Mating Connector Part Numbers	
Manufacturer	**Part Number**
AMP	1-746094-0
3M	3425-7650

APPENDIX C

COMPONENT DATA SHEETS

Intel 82C54 Programmable Interval Timer
Data Sheet Reprint

Intel 82C55A Programmable Peripheral Interface
Data Sheet Reprint

APPENDIX D

WARRANTY

LIMITED WARRANTY

Real Time Devices, Inc. warrants the hardware and software products it manufactures and produces to be free from defects in materials and workmanship for one year following the date of shipment from REAL TIME DEVICES. This warranty is limited to the original purchaser of product and is not transferable.

During the one year warranty period, REAL TIME DEVICES will repair or replace, at its option, any defective products or parts at no additional charge, provided that the product is returned, shipping prepaid, to REAL TIME DEVICES. All replaced parts and products become the property of REAL TIME DEVICES. **Before returning any product for repair, customers are required to contact the factory for an RMA number.**

THIS LIMITED WARRANTY DOES NOT EXTEND TO ANY PRODUCTS WHICH HAVE BEEN DAMAGED AS A RESULT OF ACCIDENT, MISUSE, ABUSE (such as: use of incorrect input voltages, improper or insufficient ventilation, failure to follow the operating instructions that are provided by REAL TIME DEVICES, “acts of God” or other contingencies beyond the control of REAL TIME DEVICES), OR AS A RESULT OF SERVICE OR MODIFICATION BY ANYONE OTHER THAN REAL TIME DEVICES. EXCEPT AS EXPRESSLY SET FORTH ABOVE, NO OTHER WARRANTIES ARE EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, AND REAL TIME DEVICES EXPRESSLY DISCLAIMS ALL WARRANTIES NOT STATED HEREIN. ALL IMPLIED WARRANTIES, INCLUDING IMPLIED WARRANTIES FOR MECHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED TO THE DURATION OF THIS WARRANTY. IN THE EVENT THE PRODUCT IS NOT FREE FROM DEFECTS AS WARRANTED ABOVE, THE PURCHASER’S SOLE REMEDY SHALL BE REPAIR OR REPLACEMENT AS PROVIDED ABOVE. UNDER NO CIRCUMSTANCES WILL REAL TIME DEVICES BE LIABLE TO THE PURCHASER OR ANY USER FOR ANY DAMAGES, INCLUDING ANY INCIDENTAL OR CONSEQUENTIAL DAMAGES, EXPENSES, LOST PROFITS, LOST SAVINGS, OR OTHER DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PRODUCT.

SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES FOR CONSUMER PRODUCTS, AND SOME STATES DO NOT ALLOW LIMITATIONS ON HOW LONG AN IMPLIED WARRANTY LASTS, SO THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU.

THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS, AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM STATE TO STATE.

AD2110/ADA2110 User Settings	
Base I/O Address:	
(hex)	(decimal)
P9 IRQ Source:	**P9 IRQ Channel:**
P10 EOC:	**P10 IRQ Channel:**