AD1210/ADA1210 User's Manual

Real Time Devices, Inc.

"Accessing the Analog World"®

ISO9001 and AS9100 Certified

Publication No. 1210-9409

AD1210/ADA1210 User's Manual

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

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

9409

Table of Contents

List of Illustrations

INTRODUCTION

The AD1210 and ADA1210 Advanced Industrial Control boards turn your IBM PC/XT/AT or compatible into a high-speed, high-performance data acquisition and control system. Installed within a single short or full size expansion slot in the computer, the 1210 features:

- 16 single-ended analog input channels,
- 12-bit, 5 microsecond analog-to-digital converter with 125 kHz throughput,
- ±5, ±10, or 0 to +10 volt input range,
- Resistor configurable gain,
- Programmable automatic channel scanning,
- Programmable burst mode,
- DMA transfer,
- Trigger in and trigger out for external triggering or cascading boards,
- 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 cascaded for pacer clock),
- Two 12-bit digital-to-analog output channels with dedicated grounds (ADA1210 only),
- ±5, ±10, 0 to +5, or 0 to +10 volt analog output range (ADA1210 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 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. An on-board resistor configurable gain circuit lets you customize the input gain for all channels to any value greater than 1 by adding the appropriate components as described near the end of Chapter 1.

A/D conversions are performed in 5 microseconds, and the maximum throughput rate is 125 kHz. Conversions are controlled through software, by an on-board pacer clock, or by an external trigger brought onto the board through the I/O connector.

The converted data can be transferred through the PC data bus to PC memory in one of two ways: by using the microprocessor or by direct memory access (DMA). The mode of transfer is software-selectable and the DMA channel is chosen by jumper settings on the board. The PC data bus is used to read and/or transfer data to PC memory. In the DMA transfer mode, you can make continuous transfers directly to PC memory without going through the processor.

Digital-to-Analog Conversion (ADA1210 Only)

The digital-to-analog (D/A) circuitry on the ADA1210 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. Both outputs are simultaneously updated by writing to an I/O port. Access through DMA is not available.

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. Two of the timer/counters are cascaded and can be used internally for the pacer clock. The third is available for counting applications, or it can be cascaded to the other two timer/counters.

Digital I/O

The 1210 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 board package:

- AD1210 or ADA1210 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 board 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*VIEW for monitoring and data acquisition, or SIGNAL*MATH for integrated data acquisition and sophisticated digital signal processing and analysis. 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 1210 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 XP50 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 1210 has jumper and switch settings you can change if necessary for your application. The board is factory-configured as listed in the table and shown on the 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.

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.

The final section describes how to install two resistors and a trimpot to set the resistor configurable gain to the value desired for your application. A pad for installing a capacitor is also included in the gain circuitry for creating a low-pass filter.

Factory-Configured Switch and Jumper Settings

Table 1-1 lists the factory settings of the user-configurable jumpers and switch on the 1210 board. 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; selects A/D trigger source; selects GATE 2 source	Jumpers installed on CLK0-XTAL, CLK2-OUT1, PCLK-PCK, TRIG-OUT2, GT2-EG2
P4	Connects one of four software selectable interrupt sources to an interrupt channel; pulls tri-state buffer to ground (G) for multiple interrupt applications	Jumper installed on G (ground for buffer); interrupt channels disabled
P5	Sets the DMA request (DRQ) and DMA acknowledge (DACK) channel	Disabled
P6	Sets the D/A output voltage range for DAC 1	-5 to +5 volts
P7	Sets the D/A output voltage range for DAC 2	-5 to +5 volts
P8	Sets the analog input voltage range and polarity	-5 to +5 volts
P9	Sets the state of the top 4 bits (the bits not used by the 12-bit converter) of the 16-bit A/D output word	Bipolar (must be the same as P8)
S1	Sets the base address	300 hex (768 decimal)

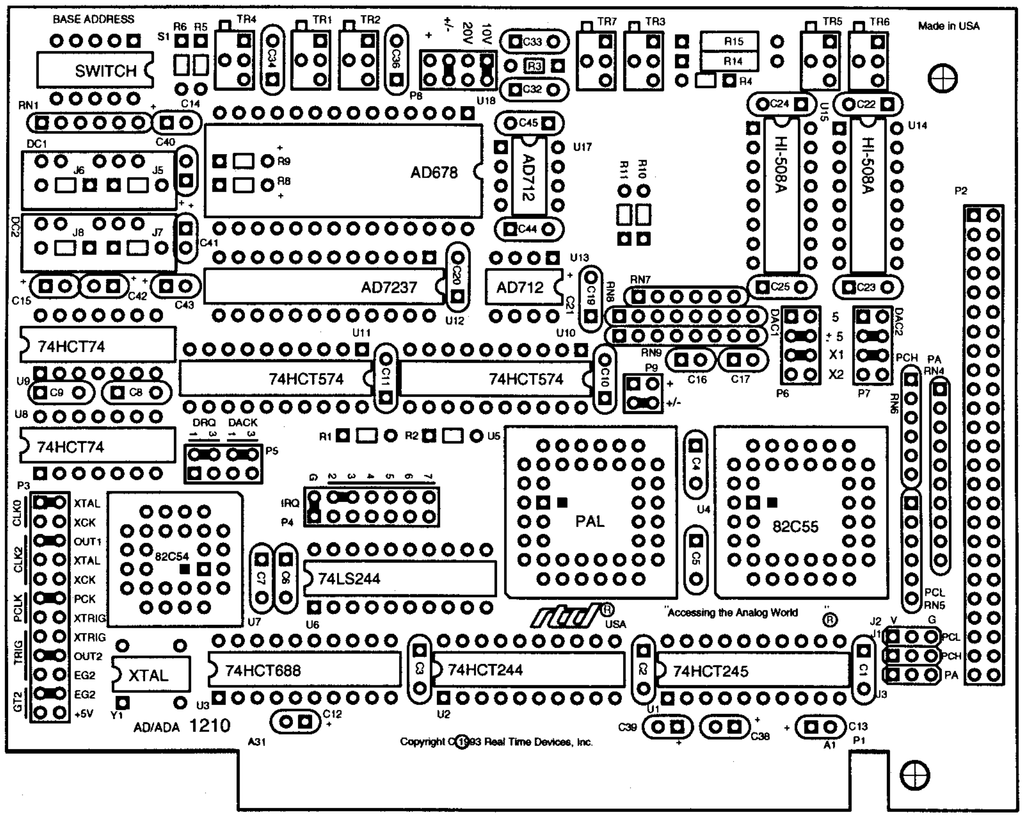


Fig. 1-1 — Board Layout Showing Factory-Configured Settings

P3 — 8254 Timer/Counter Sources (Factory Settings: See Table 1-1 & Figure 1-2)

This header connector, shown in Figure 1-2, lets you select the clock sources for the 8254 timer/counters, TC0 and TC2. TC0 and TC1 are cascaded to form the pacer clock. This header is also used to configure the pacer clock input, trigger input, and GATE 2 sources. Figure 1-3 shows a block diagram of the timer/counter circuitry to help you in making these connections.

The clock source for TC0 and TC1 is selected by placing a jumper on XTAL or XCK in the CLK0 section of the header. XTAL is the on-board 8 MHz clock and XCK is an external clock source you can connect through the external I/O connector (P2-45).

Below the CLK0 pins are three pairs of pins labeled CLK2. These pins are used to select the clock source for TC2. OUT1 connects the output of TC1 (the pacer clock output) to the clock input of TC2. Installing a jumper here cascades all three timer/counters, a feature necessary when using SIGNAL*VIEW or SIGNAL*MATH software for data acquisition and control. XTAL is the on-board 8 MHz clock, and XCK is connected to the same external clock source as CLK0-XCK (P2-45).

The next group of pins on this header, PCLK, lets you use the on-board (internal) pacer clock or an external pacer clock connected through the TRIGGER IN pin on the I/O connector (P2-39) to control A/D conversions. A jumper must be placed on PCK in order to use the internal pacer clock (output from TC1). Or, you can place the jumper across XTRIG and connect any external pacer clock source to P2-39 to trigger the A/D converter.

The TRIG pins select the hardware source used to trigger the burst mode when the external trigger enable bit at BA + 5 is enabled. Bursts can be triggered from one of three hardware sources: XTRIG, an external trigger signal routed onto the board through P2-39; OUT2, the output from timer/counter 2; or EG2, an external gate (EXT GATE 2) signal routed onto the board through P2-46. When the trigger enable bit at BA + 5 is disabled, bursts are triggered through software.

The last group of pins, GT2, select the gate source for timer/counter 2's gate input. This jumper is provided so that you can disconnect the GATE input for the third timer/counter from the EXT GATE 2 pin at the I/O connector and tie the gate high if you are using the EXT GATE 2 pin as a trigger source.

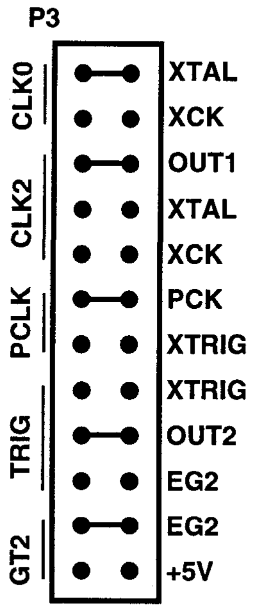


Fig. 1-2 — 8254 Timer/Counter Sources Jumpers, P3

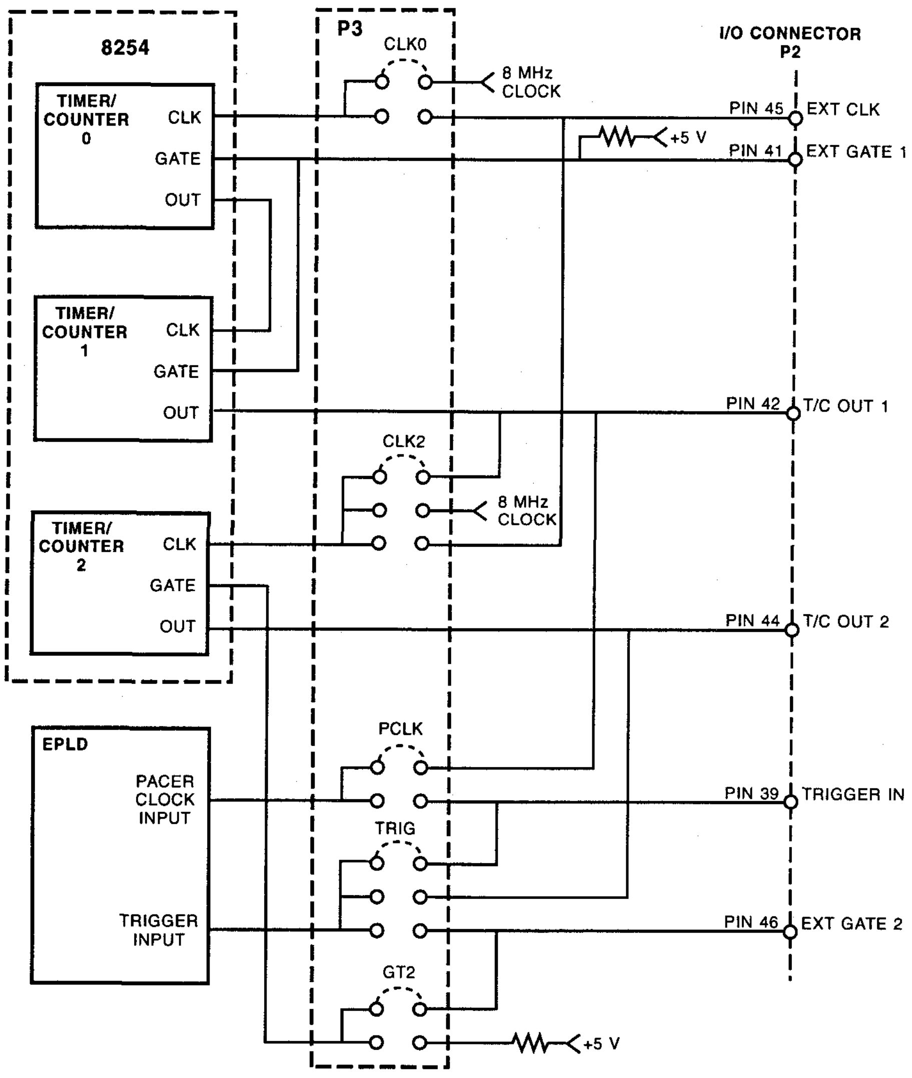


Fig. 1-3 — 8254 Timer/Counter Circuit Diagram

P4 — Interrupt Channel Select(Factory Setting: Jumper G; Interrupt Channel Disabled)

This header connector, shown in Figure 1-4, lets you connect any one of four software selectable 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 1-4a shows the factory setting; Figure 1-4b shows the interrupt source connected to IRQ3.

When jumpered, the leftmost pair of pins on P4, labeled G, connects a pull-down resistor to the output of a high-impedance tri-state driver which carries the interrupt request signal. This pull-down resistor drives the interrupt request line low whenever interrupts are not active. Whenever an interrupt request is made, the tri-state buffer is enabled, forcing the output high and generating an interrupt. You can monitor the interrupt status through bit 2 in the status word (I/O address location BA + 0). After the interrupt has been serviced, the reset command returns the IRQ line low, disabling the tri-state buffer, and pulling the output low again. Figure 1-5 shows this circuit. Because the interrupt request line is driven low only by the pull-down resistor, you can have two or more boards which share the same IRQ channel. You can tell which board issued the interrupt request by monitoring each board's IRQ status bit.

NOTE: When you use multiple boards that share the same interrupt, only one board should have the G jumper installed. The rest should be disconnected. Whenever you operate a single board, the G jumper should be installed.

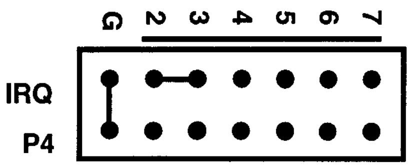


Fig. 1-4a — IRQ Disabled

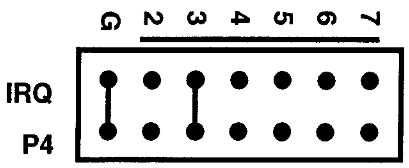


Fig. 1-4b — IRQ3 Selected

Fig. 1-4 — Interrupt Channel Select Jumper, P4

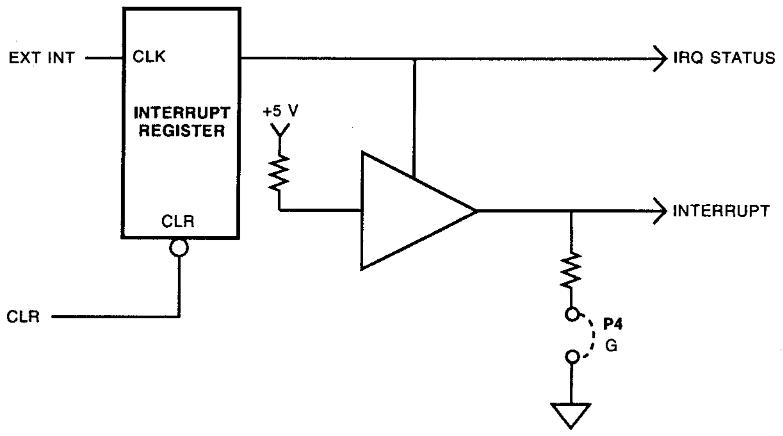


Fig. 1-5 — Pulling Down the Interrupt Request Line

P5 — DMA Request and DMA Acknowledge Channel (Factory Setting: Disabled)

This header connector, shown in Figure 1-6, lets you select channel 1 or 3 for DMA transfers by installing two jumpers, one on the DMA request line and one on the DMA acknowledge line. The DMA request line (DRQ) must be set to the same channel as the DMA acknowledge line (DACK). The factory setting is DMA disabled (jumpers in a stored position). Note that if any other device in your system is already using your selected DMA channel, channel contention will result, causing erratic operation.

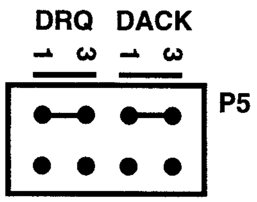


Fig. 1-6 — DMA Request & DMA Acknowledge Channel Jumpers, P5

P6 — DAC 1 Output Voltage Range (Factory Setting: +5 to -5 volts)

This header connector, shown in Figure 1-7, 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 select the range, bipolar (±5) or unipolar (5). The bottom two jumpers select the multiplier, X2 or X1. When a jumper is on the X2 multiplier pins, the range values become ±10 and 10. The table below shows the four possible combinations of jumper settings, and the diagram shows the two bipolar settings. This header does not have to be set the same as P7.

	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

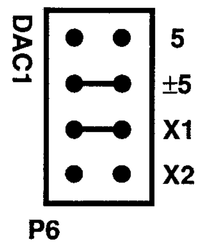


Fig. 1-7 — DAC 1 Output Voltage Range Jumper, P6

P7 — DAC 2 Output Voltage Range (Factory Setting: +5 to -5 volts)

This header connector, shown in Figure 1-8, 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 select the range, bipolar (±5) or unipolar (5). The bottom two jumpers select the multiplier, X2 or X1. When a jumper is on the X2 multiplier pins, the range values become ±10 and 10. The table below shows the four possible combinations of jumper settings, and the diagram shows the two bipolar settings. This header does not have to be set the same as P6.

	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

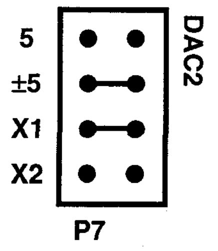


Fig. 1-8 — DAC 2 Output Voltage Range Jumper, P7

P8 — Analog Input Voltage Range and Polarity (Factory Setting: -5 to +5 Volts)

This header connector, shown in Figure 1-9, 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. Note that the jumper on P9 must match the polarity selected on P8 for proper board operation.

CAUTION: When using the 0 to +10 or ±10 volt range, you must have ±12 volts DC supplied to the board for accurate results at the extremes of these ranges. Any supplies less than ±12 volts will degrade performance.

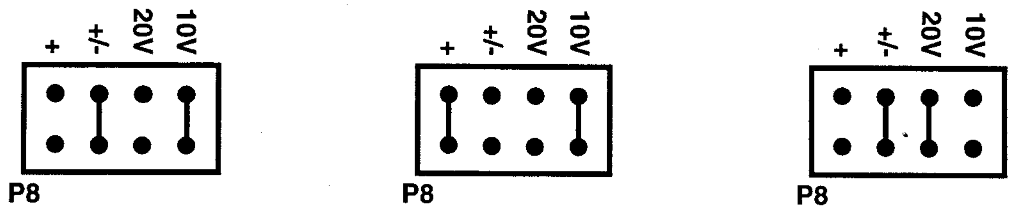


Fig. 1-9a: -5 to +5 volts (Factory Setting)

Fig. 1-9b: 0 to +10 volts

Fig. 1-9c: -10 to +10 volts

Fig. 1-9 —Analog Input Voltage Range and Polarity Jumper, P8

P9 — A/D Data Word Bit State Set (Factory Setting: +/-)

This header connector, shown in Figure 1-10, sets the state of the unused four bits in the 8-bit MSB of the 16-bit A/D data word. This header ensures that these four topmost bits are set at 0 for unipolar conversions and at the same state as the most significant bit of the 12-bit A/D converted data for bipolar conversions. Chapter 4, BA + 1, explains this in more detail. **NOTE: P8 and P9 must be set the same for proper board operation.**

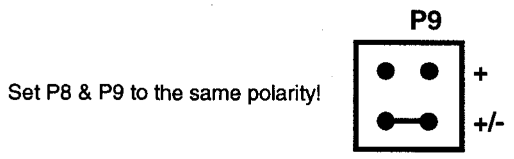


Fig. 1-10 —A/D Data Word Bit State Set Jumper, P9

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 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 1210 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

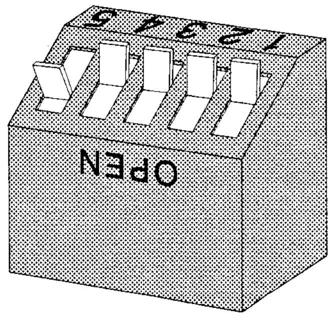


Fig. 1-11 — Base Address Switch, S1

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 1-12 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 1-12 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 1-13 shows Port A lines with pull-ups, Port C Lower with pull-downs, and Port C Upper with no resistors.

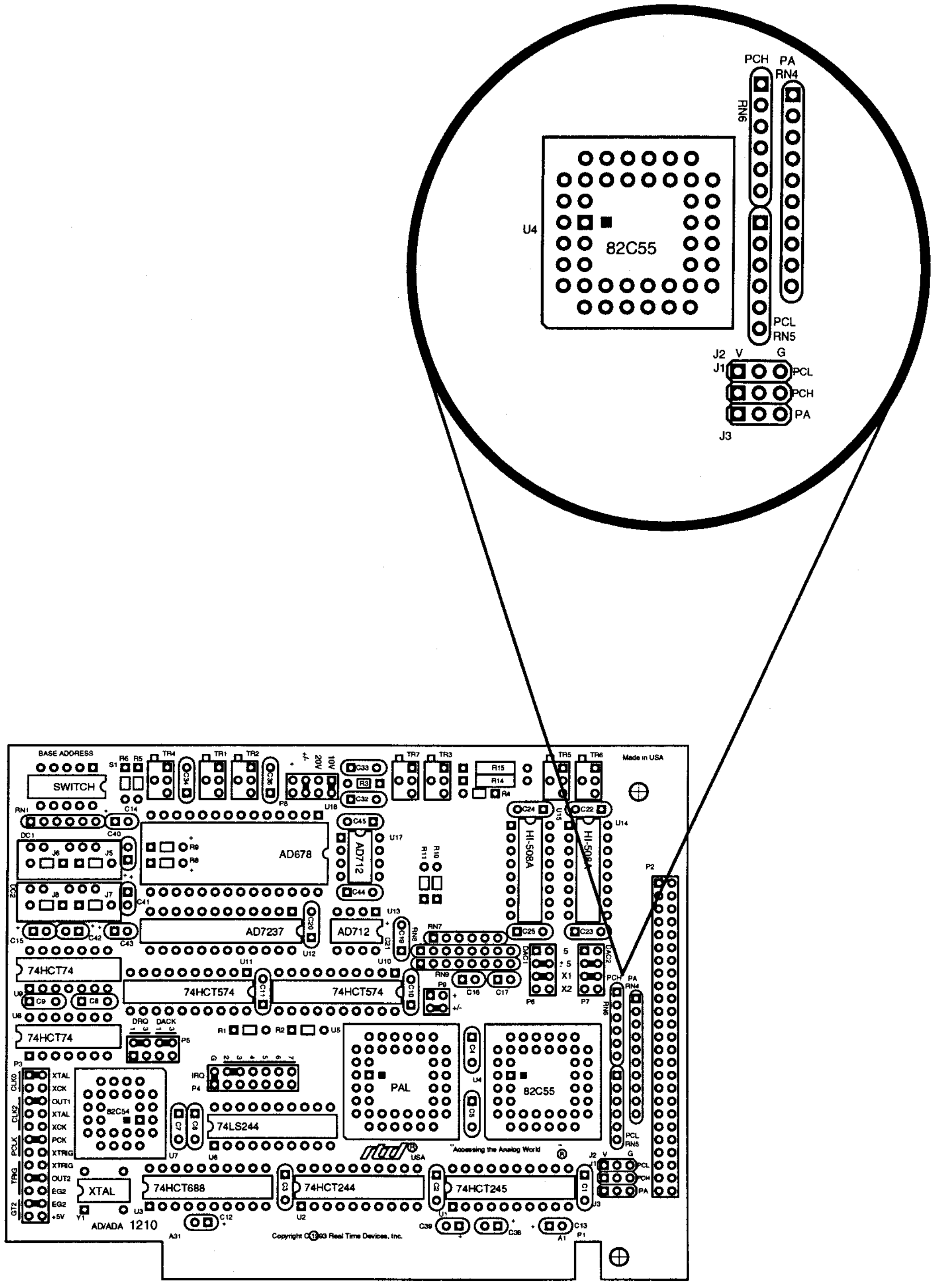


Fig. 1-12 — Pull-up/Pull-down Resistor Circuitry

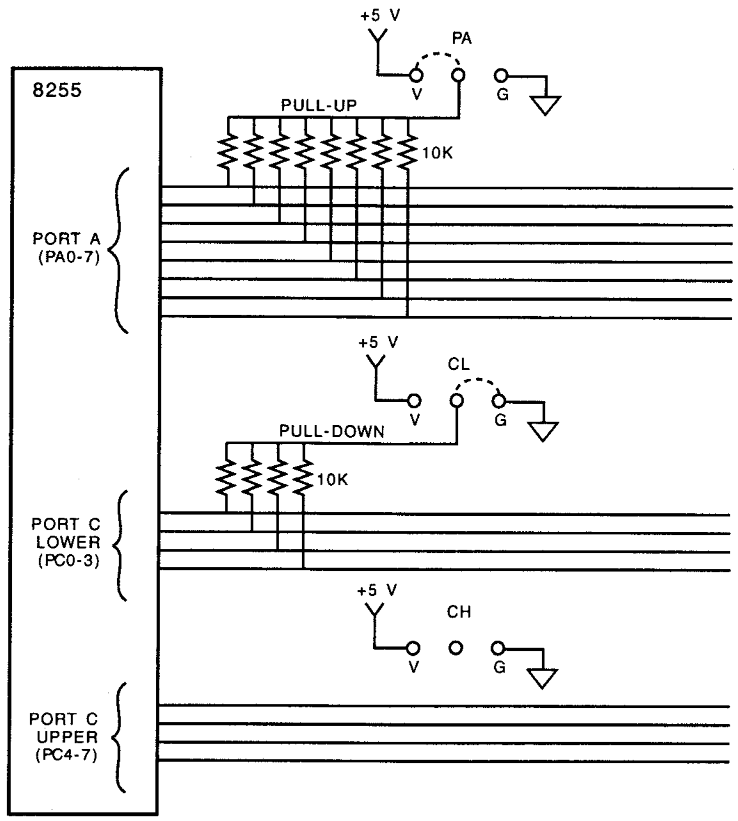
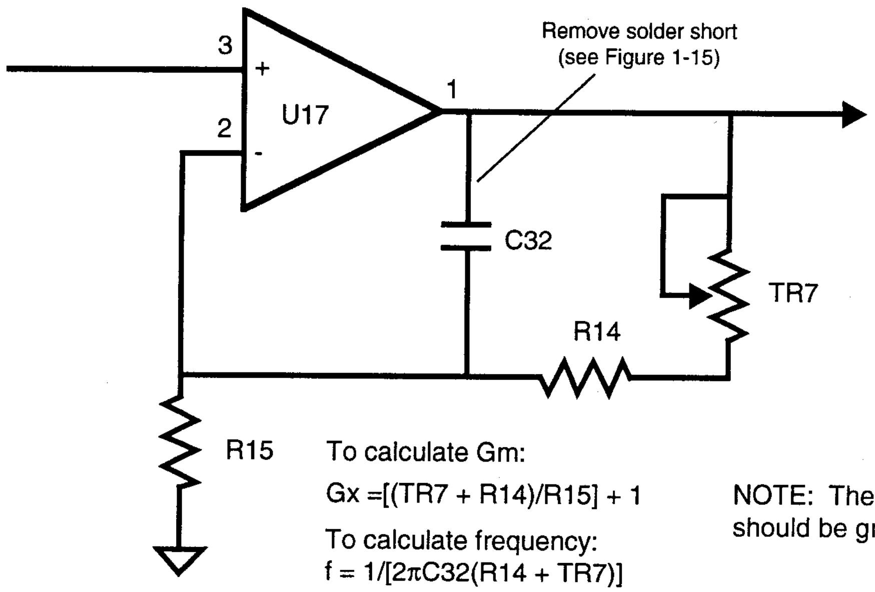


Fig. 1-13 — Adding Pull-ups and Pull-downs to Digital I/O Lines

Gx, Resistor Configurable Gain

The 1210 has a resistor configurable gain circuit, Gx, so that you can easily configure special gain settings for a specific application. Note that when you use this feature, all of the input channels will operate only at your custom gain setting. Gx is derived by adding resistors R14 and R15, trimpot TR7, and capacitor C32, 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 C32 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 C32 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 Gx circuitry is configured.

As shown in Figure 1-14, a solder short must be removed from the board to activate the Gx circuitry. This short is located on the **bottom side** of the board under U17 (AD712 IC). Figure 1-15 shows the location of the solder short.



To calculate Gm:

$Gx = [(TR7 + R14)/R15] + 1$

To calculate frequency:

$f = 1/[2\pi C32(R14 + TR7)]$

NOTE: The value of R15 should be greater than 1 kilohm.

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

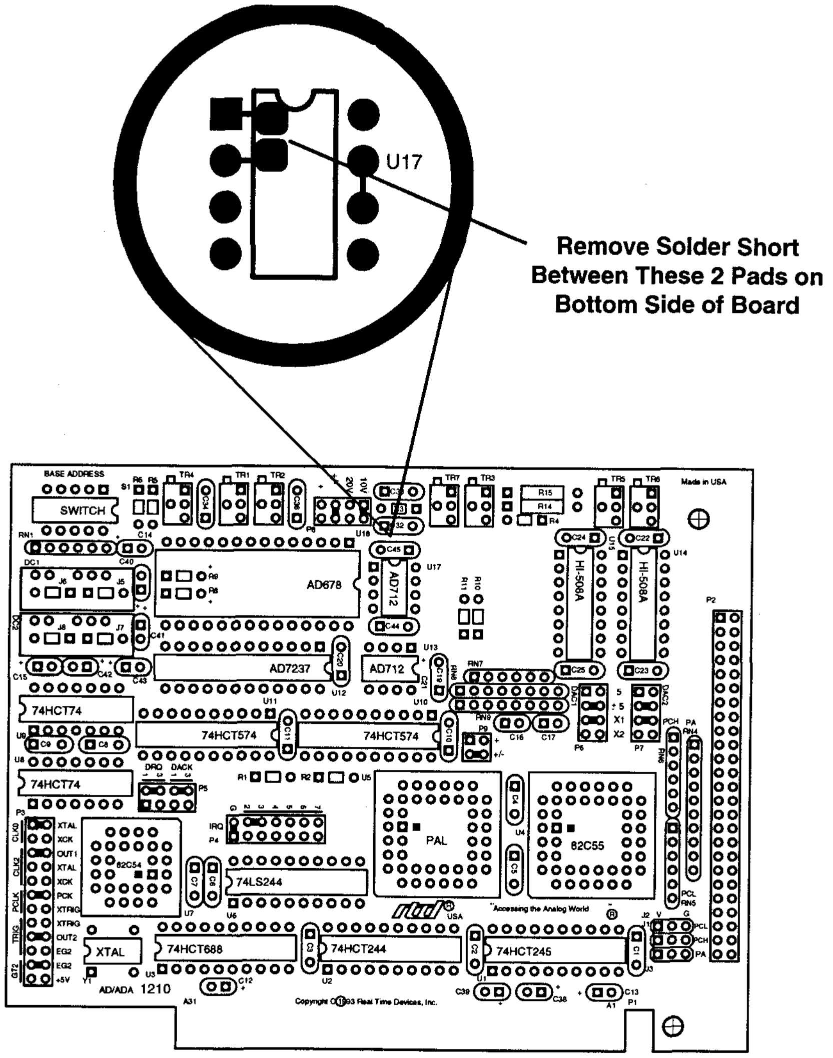


Fig. 1-15 — Diagram for Removal of Solder Short

CHAPTER 2

BOARD INSTALLATION

The 1210 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 1210DIAG 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 1210's P2 I/O connector pinout. Refer to this diagram as you make your I/O connections.

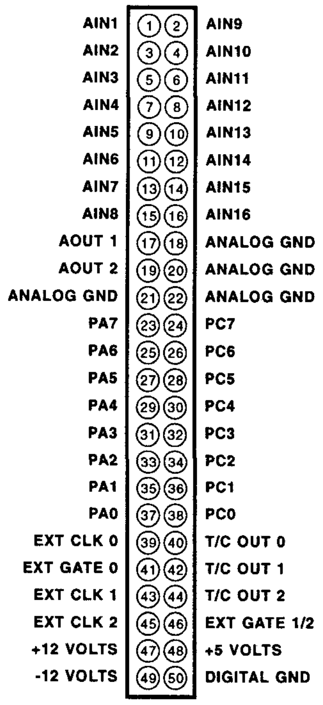


Fig. 2-1 — P2 I/O Connector Pin Assignments

Connecting the Analog Input Pins

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.

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.

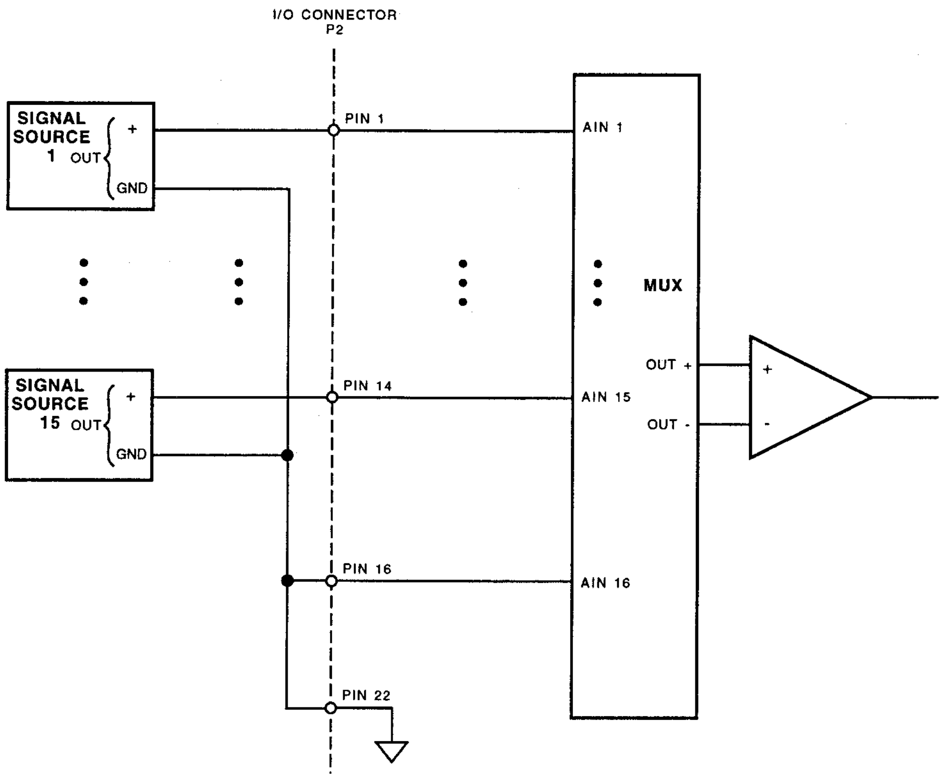


Fig. 2-2 — Connecting the Analog Inputs

Connecting the Trigger In and Trigger Out Pins, Cascading Boards

The board has an external trigger input (P2-39) and output (P2-43) so that conversions can be started based on external events, or so that two or more boards can be cascaded and run synchronously in a "master/slave" configuration. By cascading two (or more) boards as shown in Figure 2-3, they can be triggered to start an A/D conversion at the same time (sampling uncertainty is less than 50 nanoseconds). When you cascade boards, be sure to set each board for a different base address (see Chapter 1), or system contention will result.

NOTE: When cascading boards, the sampling uncertainty is less than 50 nanoseconds. If this level of uncertainty is too great for your application, you can connect the trigger signal to the trigger input of each board. In this configuration, the boards are not cascaded, but rather driven by the same trigger pulse at the same time, and the sampling uncertainty is reduced to less than 5 nanoseconds.

If you apply an external trigger to the board's trigger in pin, note that a jumper should be installed on PCLK-XTRIG on P3 (see Chapter 1). The board is triggered on the positive edge of the pulse and the pulse duration should be at least 100 nanoseconds.

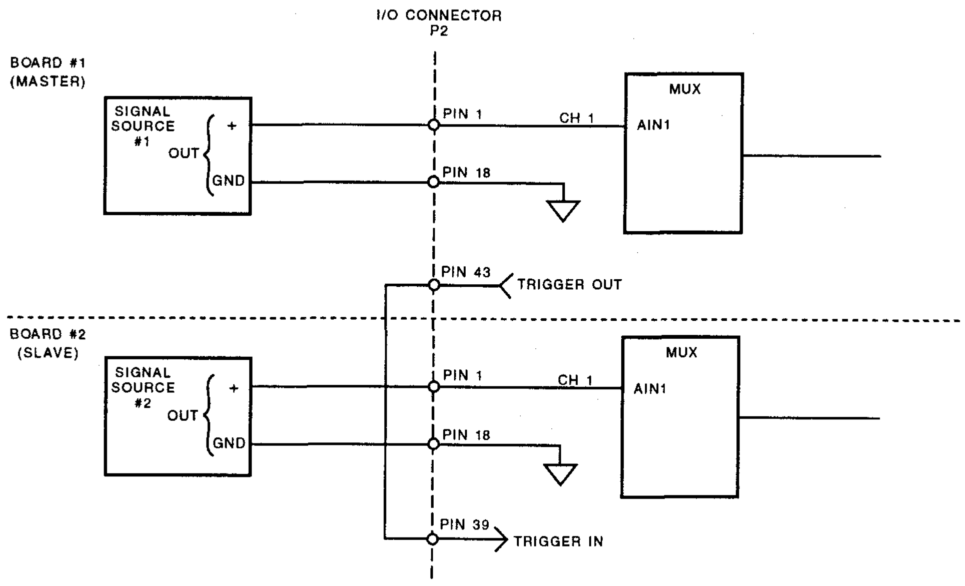


Fig. 2-3 — Cascading Two Boards for Simultaneous Sampling

Connecting the Analog Outputs (ADA1210 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 1210DIAG 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, 1210DIAG, 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 1210 hardware. The major circuits are the A/D, the D/A, the timer/counters, and the digital I/O lines.

The 1210 board has four major circuits, the A/D, the D/A (ADA1210 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.

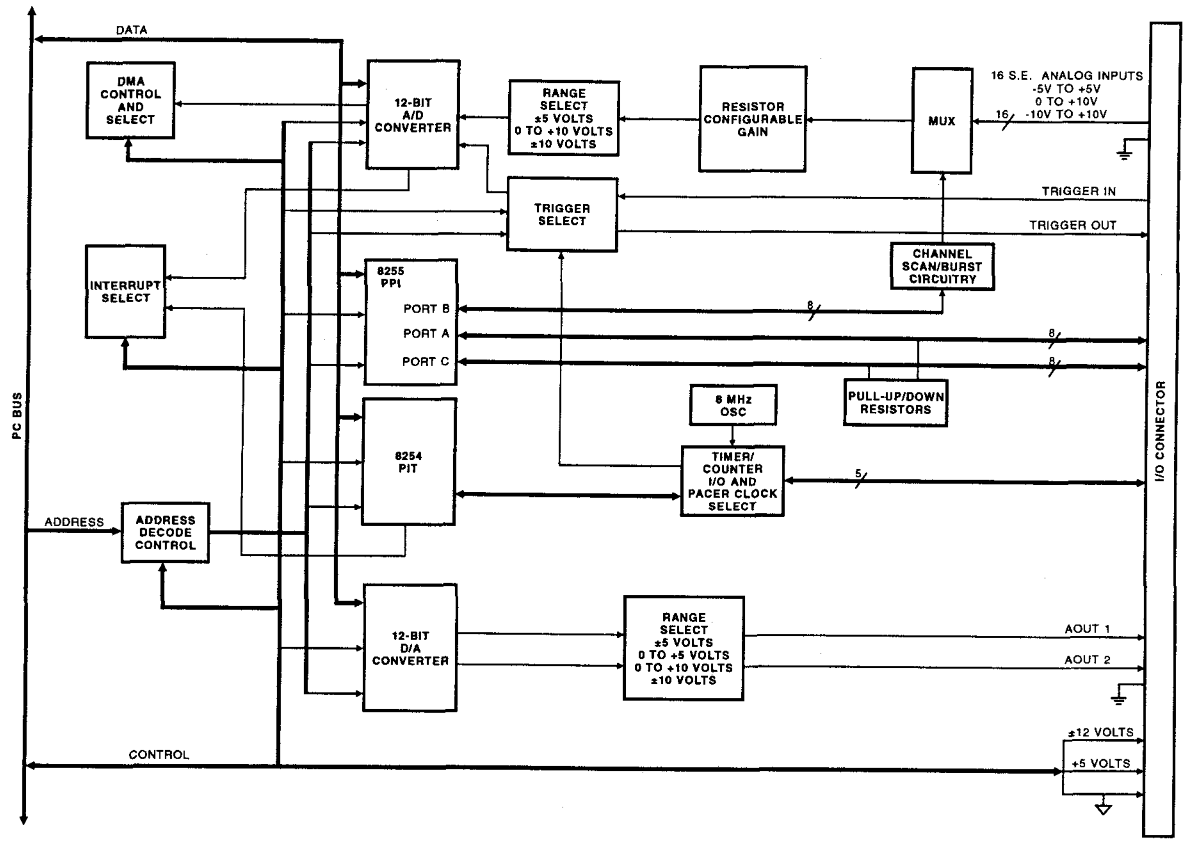


Fig. 3-1 — 1210 Block Diagram

A/D Conversion Circuitry

The 1210 performs analog-to-digital conversions on up to 16 single-ended 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. A resistor configurable gain circuit lets you amplify lower level signals to more closely match the board's input ranges. Overvoltage protection to ±35 volts is provided at the inputs.

CAUTION: When using the 0 to +10 or ±10 volt range, you must have ±12 volts DC supplied to the board for accurate results at the extremes of these ranges. Any supplies less than ±12 volts will degrade performance.

A/D Converter

The AD678 12-bit successive approximation A/D converter accurately digitizes dynamic input voltages in 5 microseconds, for a maximum throughput rate of 200 kHz for the converter alone. The AD678 contains a sample-and-hold amplifier, a 12-bit A/D converter, a 5-volt reference, a clock, and a digital interface to provide a complete A/D conversion function on a single chip. Its low-power CMOS logic combined with a high-precision, low-noise design give you accurate results.

Conversions are initiated through software (internally triggered) or by using an external trigger brought onto the board through the I/O connector. An on-board or external pacer clock can be used to control the conversion rate. Conversion modes are described in Chapter 4, *Board Operation and Programming*.

Data Transfer

The converted data can be transferred through the PC data bus to PC memory in one of two ways: by using the microprocessor or by direct memory access (DMA). Data bus transfers take more processor time to execute. They use polling and interrupts to determine when data has been acquired and is ready for transfer. DMA places data directly into the PC's memory, one byte at a time, with minimal use of processor time. DMA transfers are managed by the DMA controller as a background function of the PC, letting you operate at higher throughput rates. The maximum throughput rate of the board is 125 kHz.

D/A Converters (ADA1210 Only)

Two independent 12-bit analog output channels are included on the ADA1210. 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. Two of the timer/counters, TC0 and TC1, are cascaded so that they can be used for the pacer clock. The pacer clock is described in Chapter 4. You can use the remaining timer/counter, TC2, for counting applications, or cascade it to TC0 and TC1 for timing applications. Figure 3-2 shows the timer/counter circuitry.

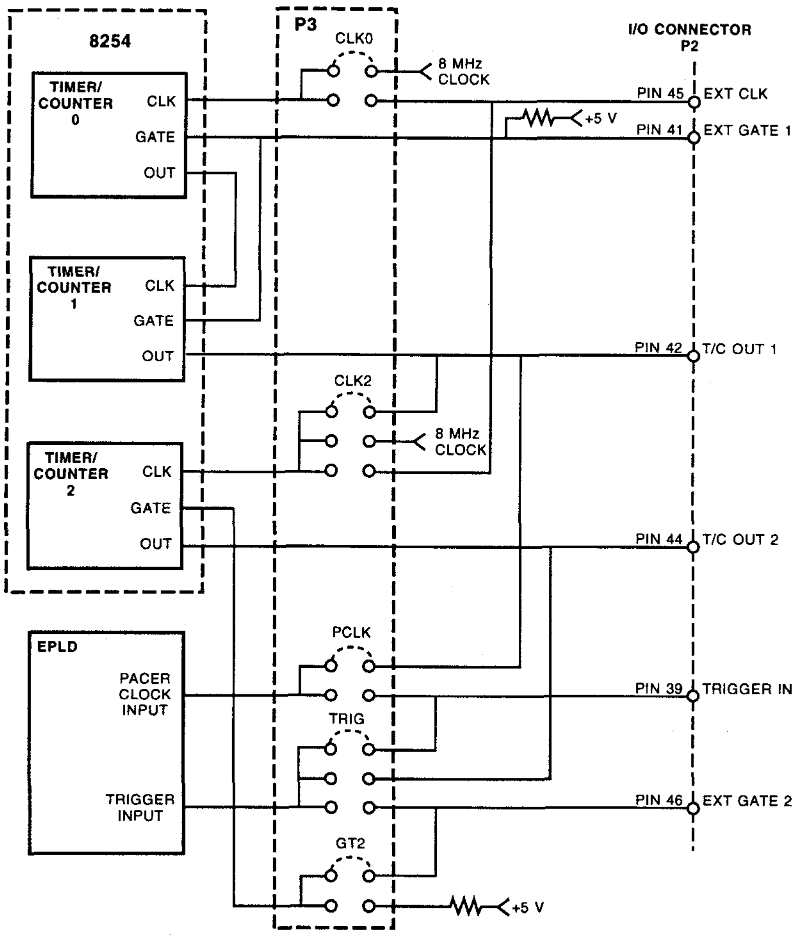


Fig. 3-2 — 8254 Timer/Counter Circuit Block Diagram

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.

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 handshaking 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.

CHAPTER 4

BOARD OPERATION AND PROGRAMMING

This chapter shows you how to program and use your 1210 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 1210 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 computer. 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 — AD1210/ADA1210 I/O Map

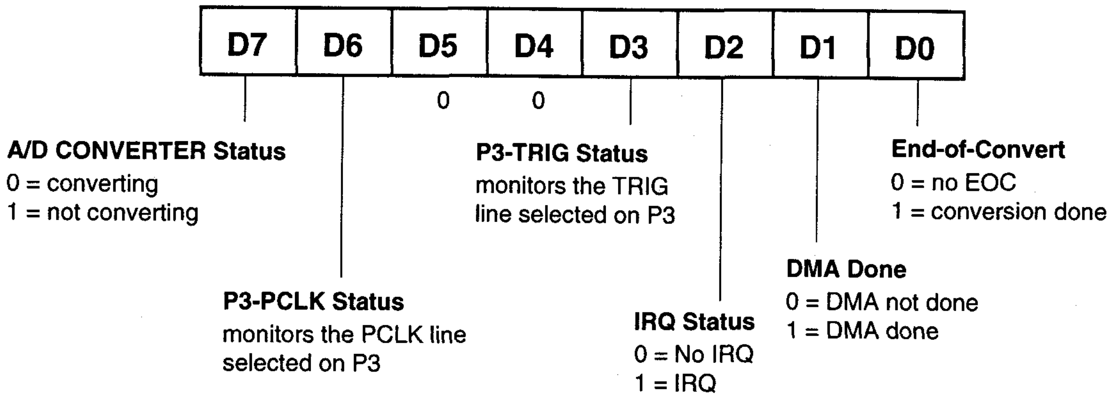
Register Description	Read Function	Write Function	Address * (Decimal)
Read Status/Start Convert	Read status word	Start A/D conversion	BA + 0
Read Data/Update DACs	Read converted data, LSB first, then MSB	Simultaneously update DAC 1 and DAC 2 (ADA1210 only)	BA + 1
Reset	Reserved	Resets board so that it is ready to start A/D conversions	BA + 2
Scan/Burst	Read current settings	Programs number of scan/burst channels, enables scan/burst; selects IRQ source	BA + 3
8255 PPI Port A	Read Port A digital input lines	Program Port A digital output lines	BA + 4
8255 PPI Port B (Channel/ Board Functions Select)	Read Port B bits	Program channel; external trigger enable, IRQ enable	BA + 5
8255 PPI Port C	Read Port C digital input lines	Program Port C digital output lines	BA + 6
8255 PPI Control Word	Reserved	Program PPI configuration	BA + 7
8254 Timer/Counter 0 (Used for pacer clock)	Read count value	Load count register	BA + 8
8254 Timer/Counter 1 (Used for pacer clock)	Read count value	Load count register	BA + 9
8254 Timer/Counter 2 (Available for external use)	Read count value	Load count register	BA + 10
8254 Timer/Counter Control Word	Reserved	Program counter mode	BA + 11
D/A Converter 1 LSB	Read A/D converted data LSB	Program DAC1 LSB (ADA1210 only)	BA + 12
D/A Converter 1 MSB	Read A/D converted data MSB	Program DAC1 MSB (ADA1210 only)	BA + 13
Clear IRQ/ D/A Converter 2 LSB	Clear IRQ status	Program DAC2 LSB (ADA1210 only)	BA + 14
Clear DMA Done/ D/A Converter 2 MSB	Clear DMA done flag	Program DAC2 MSB (ADA1210 only)	BA + 15

* BA = Base Address

BA + 0: Read Status/Start Convert (Read/Write)

A read provides the six status bits defined below. The end-of-convert bit goes high when a conversion is complete and does not go low until the data is read, useful information when using external triggering to start conversions. The DMA done bit goes high when you are in the DMA mode and the DMA transfer is complete. The IRQ status bit goes high when an interrupt has occurred and stays high until a clear IRQ command is sent (BA + 14). D3 shows the status of the burst trigger source jumpered at P3-TRIG. D6 shows the status of the PCLK line jumpered at P3. Unlike the EOC status at bit 0, the A/D converter status, D7, goes low when a conversion starts and then goes high as soon as the conversion is completed. When the input has been sampled and a conversion is in progress, this line goes low. At this time, the analog input channel can be changed, allowing maximum throughput for scanning channels through software.

A write starts an A/D conversion (data written is irrelevant).



BA + 1: Read A/D Data/Update DAC Outputs (Read/Write)

Two successive reads provide the LSB first, followed by the MSB, for each A/D conversion, as defined below. When jumpers on P8 and P9 are set for bipolar conversions, the data word's four most significant bits match the most significant bit of the A/D converted data (bit 11). This is necessary to provide the correct twos complement representation of the converted data. When P8 and P9 are set for unipolar conversions, these top four bits are 0.

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
LSB	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MSB, Bipolar	Bit 11	Bit 11	Bit 11	Bit 11	Bit 11	Bit 10	Bit 9	Bit 8
MSB, Unipolar	0	0	0	0	Bit 11	Bit 10	Bit 9	Bit 8

BA + 2: Reset (Write Only)

Resets internal registers so that the board is ready to start conversions. The data written is irrelevant; the act of writing to this address clears the board. A reset command sets the internal byte pointer to read the LSB on the next read, resets the DRQ and IRQ registers, clears the EPLD scan/burst circuitry, and resets (clears) the DMA done bit, BA + 0, bit 1.

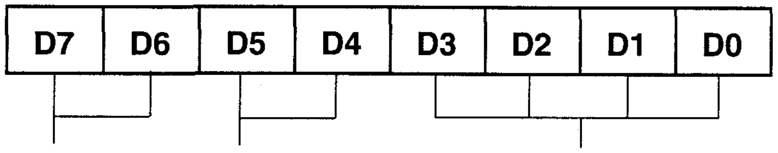
BA + 3: Scan/Burst (Read/Write)

Bits D0 through D3 program the number of consecutive analog input channels to be scanned or bursted. The channel scan or burst begins with the channel selected at BA + 5.

Bits D4 and D5 program one of four IRQ sources available for generating interrupts. The IRQ channel is set by the jumper on P4. The IRQ is enabled at bit 7, BA + 5.

Bits D6 and D7 enable the scan or burst mode. Each time you start a new scan or burst, you should first reset the board by writing to BA + 2 or by programming these bits to disable scan/burst, and then follow that step by enabling the scan or burst mode. This ensures that the EPLD scan/burst counter circuitry is cleared.

A read tells you the bit settings.



Scan/Burst Enable

00	= disabled
01	= reserved
10	= scan enabled
11	= burst enabled

IRQ Source Select

00	= A/D start convert
01	= DMA done
10	= trigger (P3-TRIG)
11	= pacer clock (P3-PCLK)

Number of Channels Scanned

0000	= 1 channel	1000	= 9 channels
0001	= 2 channels	1001	= 10 channels
0010	= 3 channels	1010	= 11 channels
0011	= 4 channels	1011	= 12 channels
0100	= 5 channels	1100	= 13 channels
0101	= 6 channels	1101	= 14 channels
0110	= 7 channels	1110	= 15 channels
0111	= 8 channels	1111	= 16 channels

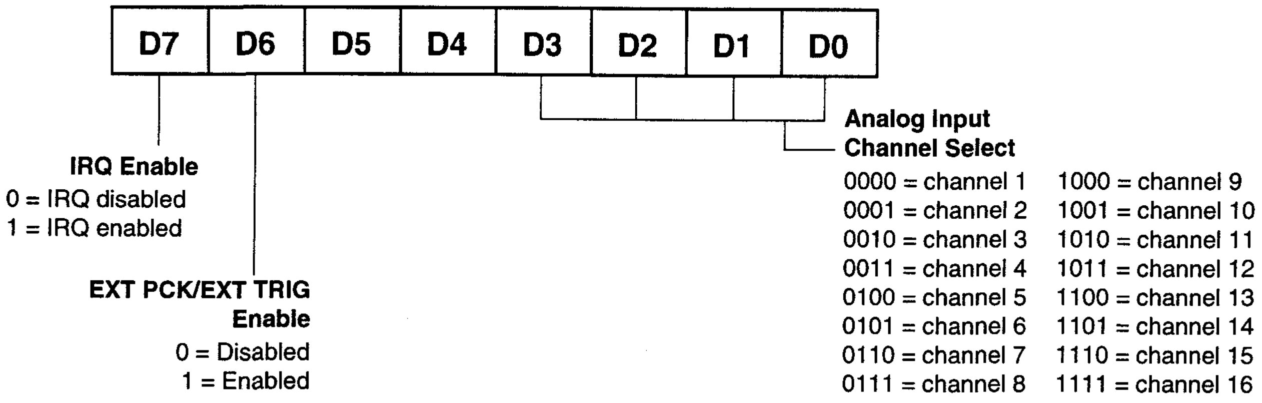
BA + 4: PPI Port A — Digital I/O (Read/Write)

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 + 5: PPI Port B — Channel/Board Functions Select (Read/Write)

A write programs the analog input channel and enables the IRQ and external trigger. Note that, because some of the Port B bits are built into the EPLD, writing to the 8255 control word does not automatically set the Port B bits to zero as it does in a typical 8255 configuration. Therefore, you must write a zero to Port B to ensure all bits are zero whenever you desire to reset this port.

Reading this register shows you the current bit settings.

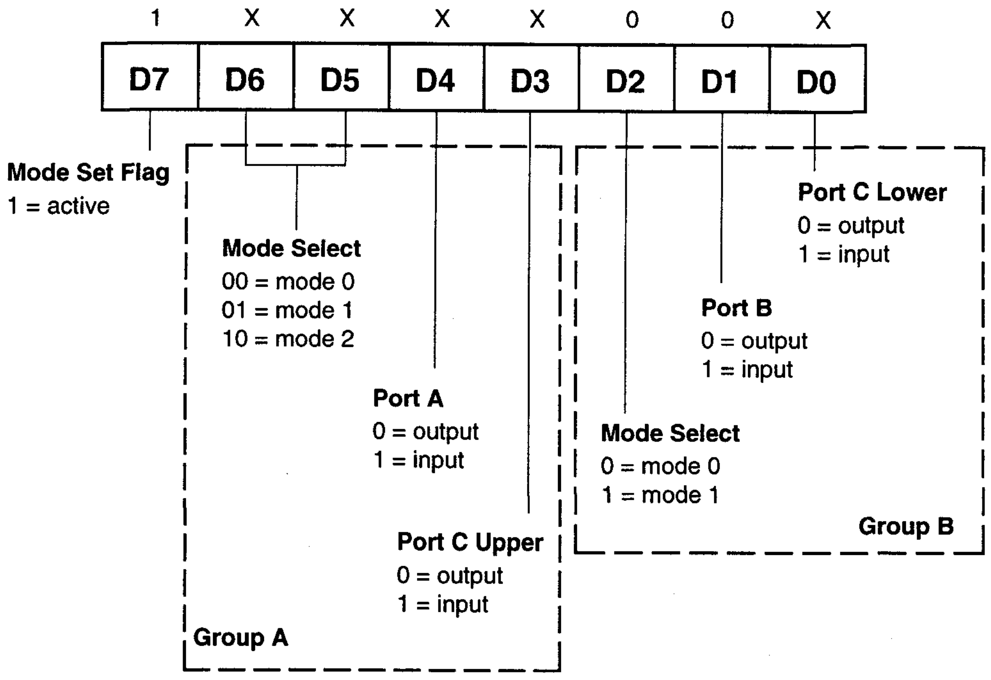


BA + 6: PPI Port C — Digital I/O (Read/Write)

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 + 7: 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).

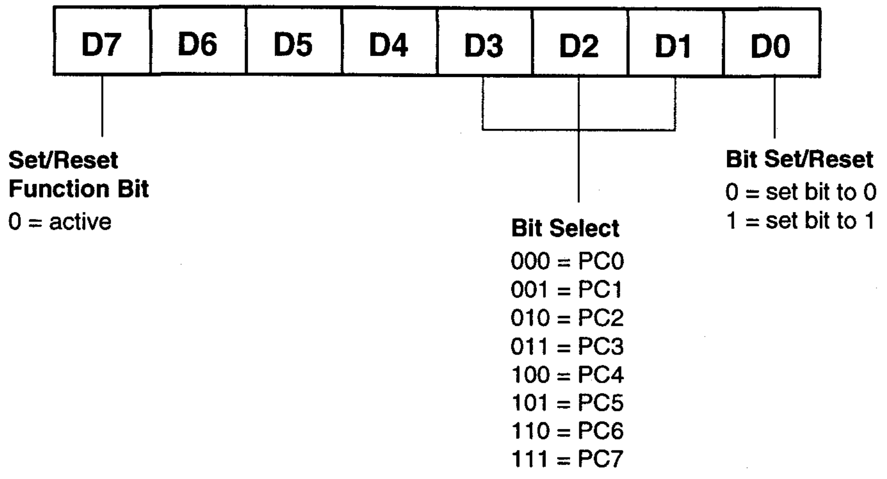


The table below shows the control words for the 16 possible Mode 0 Port I/O combinations. The control words which set Port B as an input cannot be used on the 1210.

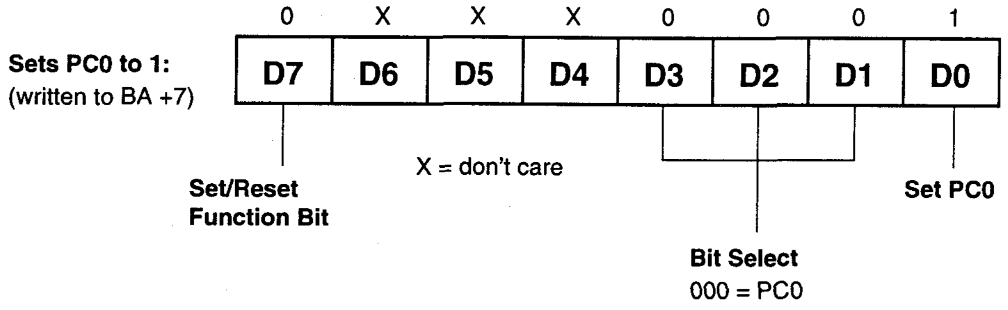
8255 Port I/O Flow Direction and Control Words, Mode 0

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 + 8: 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. This counter is cascaded with TC1 to form the 32-bit on-board pacer clock.

BA + 9: 8254 Timer/Counter 1 (Read/Write)

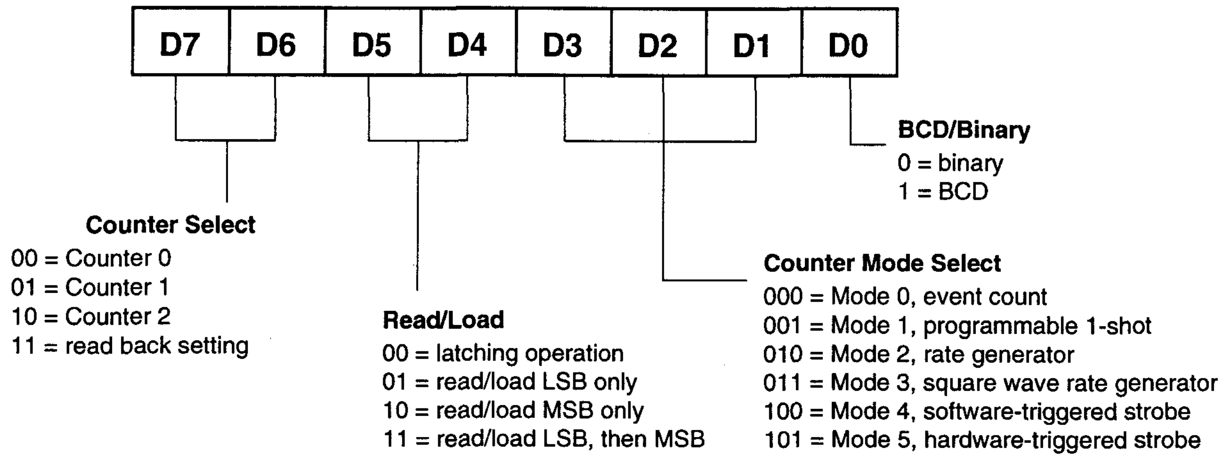
A read shows the count in the counter, and a write loads the counter with a new value. Counting begins as soon as the count is loaded. This counter is cascaded with TC0 to form the 32-bit on-board pacer clock.

BA + 10: 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. This counter can be cascaded to TC0 and TC1 or it can be used independently.

BA + 11: 8254 Control Word (Write Only)

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


D7 D6 D5 D4 D3 D2 D1 D0
BCD/Binary
0 = binary
1 = BCD
Counter Select
00 = Counter 0
01 = Counter 1
10 = Counter 2
11 = read back setting
Read/Load
00 = latching operation
01 = read/load LSB only
10 = read/load MSB only
11 = read/load LSB, then MSB
Counter Mode Select
000 = Mode 0, event count
001 = Mode 1, programmable 1-shot
010 = Mode 2, rate generator
011 = Mode 3, square wave rate generator
100 = Mode 4, software-triggered strobe
101 = Mode 5, hardware-triggered strobe

BA + 12: A/D Data LSB/D/A Converter 1 LSB (Read/Write)

A read provides the A/D converter LSB, the same data which can be read at BA + 1. This option is included with your board to take advantage of the XT computer's ability to perform a word read with a single command. When using the word read command at an even numbered address, the XT will read the byte at the first address and then automatically increment and read the byte at the next address. In this case, a word read at BA + 12 will provide the LSB, followed by the MSB (at BA + 13).

A write programs the DAC1 LSB (eight bits).

BA + 13: A/D Data MSB/D/A Converter 1 MSB (Read/Write)

A read provides the A/D converter MSB, the same data which can be read at BA + 1.

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

BA + 14: Clear IRQ Status/D/A Converter 2 LSB (Read/Write)

A read clears the IRQ status bit at bit 2, BA + 0.

A write programs the DAC2 LSB (eight bits).

BA + 15: Clear DMA Done Flag/D/A Converter 2 MSB (Read/Write)

A read clears the DMA done flag at bit 1, BA + 0.

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

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

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

Programming the AD1210/ADA1210

This section gives you some general information about programming and the board, and then walks you through the major 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 1210 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 board, 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	\ (backslash) 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 1210!**

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.

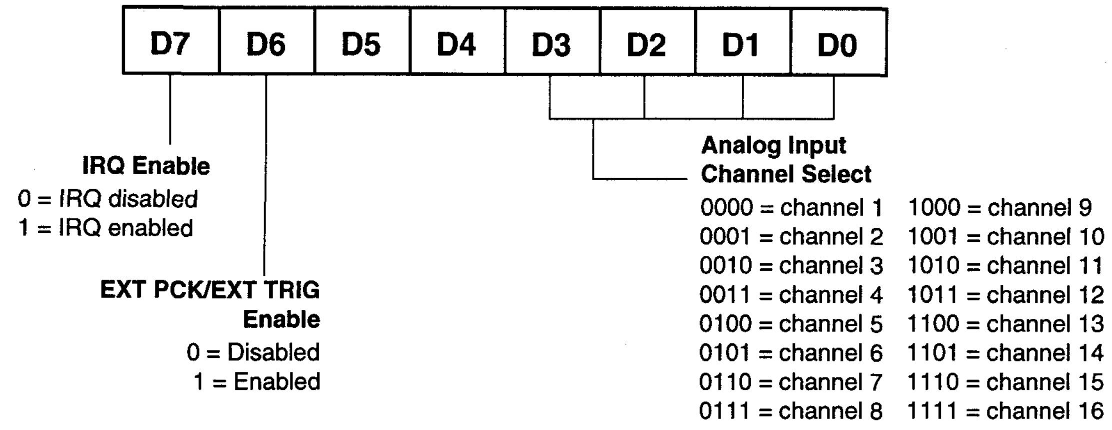
Now that you know how to clear and set bits, we are ready to look at the programming steps for the board functions.

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 diagrams 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, programmable IRQ, and external trigger and DMA enable. Port B is programmed at I/O address location BA + 5:



0000	channel 1	1000	channel 9
0001	channel 2	1001	channel 10
0010	channel 3	1010	channel 11
0011	channel 4	1011	channel 12
0100	channel 5	1100	channel 13
0101	channel 6	1101	channel 14
0110	channel 7	1110	channel 15
0111	channel 8	1111	channel 16

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 + 7 (X = don't care):

1	X	X	X	X	0	0	X
D7	D6	D5	D4	D3	D2	D1	D0

• Clearing the Board

It is good practice to start your program by resetting the board. You can do this by writing to the RESET port at BA + 2. The actual value you write to this port is irrelevant. After resetting the board following power-up, you must take an A/D reading and throw it away to make sure the converter is initialized and contains no unwanted data.

• Selecting a Channel

To select a conversion channel or the starting channel for a scan or burst, you must assign values to bits 0 through 3 in the PPI Port B port at BA + 5. The table below shows you how to determine the bit settings.

x	x	x	x	CH3	CH2	CH1	CH0	BA + 5

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

• **Enabling and Disabling the External Trigger**

The external trigger enable bit at BA + 5 enables the A/D trigger source. When this bit is enabled in the scan mode, A/D conversions are controlled by the on-board or external pacer clock, depending on the setting of the PCLK jumper at P3. When disabled, A/D conversions are triggered from software (Start Convert command), for all modes except burst. When this bit is disabled in the burst mode, bursts are triggered by software, and each conversion in the burst is triggered by the on-board or external pacer clock, again depending on the setting of the PCLK jumper at P3. When enabled in the burst mode, bursts are triggered by an external trigger, the output of timer/counter 2, or a signal brought onto the board through the EXT GATE 2 pin on P2, depending on the setting of the TRIG jumper at P3.

• **Enabling and Disabling Interrupts**

Any time you use interrupts, this bit at port BA + 5 must be set high to enable the IRQ circuitry.

• **Types of Conversions**

The board can perform single and multiple conversions, channel scanning, and bursts. Figure 4-1 shows the basic timing diagram for a conversion.

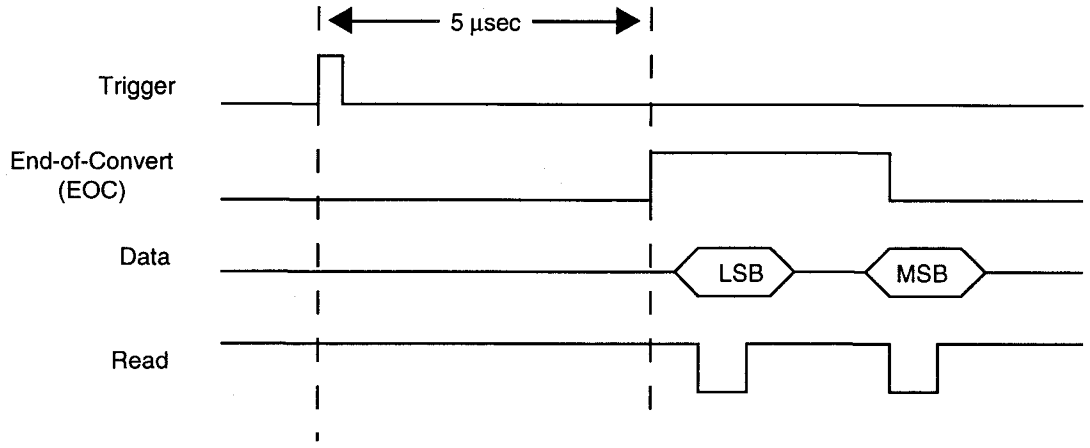


Fig. 4-1 — A/D Conversion Timing Diagram, All Modes

Single Conversion. In this mode, a single specified channel is sampled whenever a value is written to the START CONVERT port, BA + 0 (software trigger). The active channel is the one specified in the CHANNEL SELECT port.

This is the easiest of all conversions. It can be used in a wide variety of applications, such as sample every time a key is pressed on the keyboard, sample with each iteration of a loop, or watch the system clock and sample every five seconds. Figure 4-2 shows a timing diagram for single conversions. See the SOFTTRIG sample program in C and Pascal and the SINGLE program in BASIC on the example programs disk included with your board.

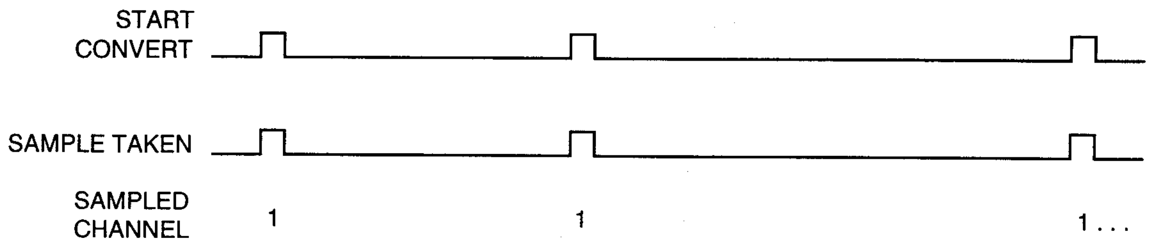


Fig. 4-2 — Timing Diagram, Single Conversion

Multiple Conversions. In this mode, conversions are continuously performed at the pacer clock rate. The pacer clock can be internal or external, depending on the setting of the PCLK jumper on P3. The maximum rate supported by the board is 125 kHz. If you use the internal pacer clock, you must program it to run at the desired rate.

This mode is ideal for filling arrays, acquiring data for a specified period of time, and taking a specified number of samples. Figure 4-3 shows a timing diagram for multiple conversions. See the MULTI sample programs in C and Pascal on the example programs disk included with your board.

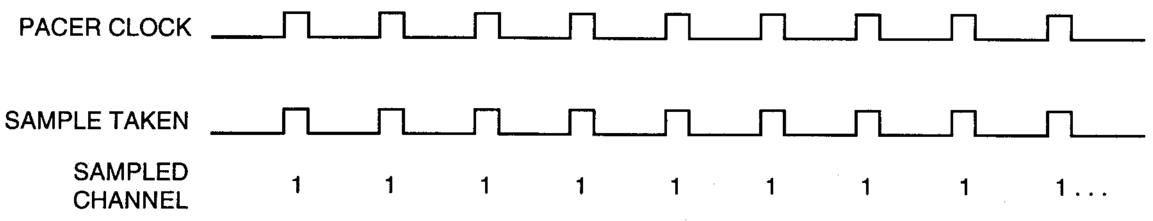


Fig. 4-3 — Timing Diagram, Multiple Conversions

Automatic Channel Scan. In this mode, the channel sampled is automatically incremented after each conversion is complete. The channel at which the scan starts is specified in the channel select bits of BA + 5. The number of channels to be scanned is specified at BA + 3, bits D0 through D3. Scanning continues until stopped. For example, by programming channel 3 at BA + 5 and four channels at BA + 3, the scan will be 3, 4, 5, 6, 3, 4, 5, 6, 3, ... until stopped. Figure 4-4 shows a diagram of this mode.

When using channel scan, you must set bits D6 and D7 at BA + 3 for the scan mode. Conversions can be performed through software or using the pacer clock. Use automatic channel scan when you want to continuously sample a sequence of channels. Since the channel counter is automatically incremented after each conversion, this mode is faster and easier than using the single conversion mode and setting the channel for each conversion from software. See the SCAN program in BASIC on the example programs disk included with your board.

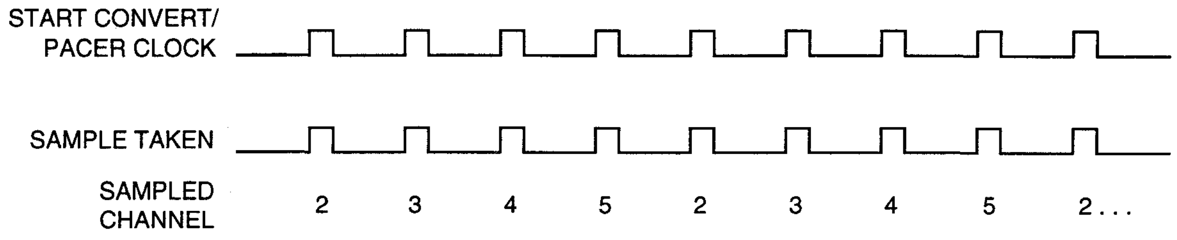


Fig. 4-4 — Timing Diagram, Channel Scanning

Programmable Burst. In this mode, a sequence of channels (from 2 to 16 channels) is scanned a single time at the burst clock rate each time a burst trigger is applied The starting channel is the channel specified at BA + 5. The number of channels to be scanned is specified at BA + 3. When using burst, you must set bits D6 and D7 at BA + 3 for the burst mode. A burst can be triggered using the Start Convert command at BA + 0 or by the hardware source (external trigger, timer/counter out 2, or external gate 2) set at TRIG on P3. If the external trigger enable bit, D6 at BA + 5, is disabled, the burst will be software triggered. If this bit is enabled, the burst will be hardware triggered. The pacer clock (internal or external) sets the time between each sample in the burst.

Burst is used when you want one sample from a specified number of channels for each trigger. Figure 4-5 shows a timing diagram for burst sampling. Often, the burst mode can be used for near-simultaneous sampling from multiple input channels. For critical simultaneous sampling applications, a simultaneous sample-and-hold board can be used (SS4 four-channel and SS8 eight-channel boards are available from Real Time Devices).

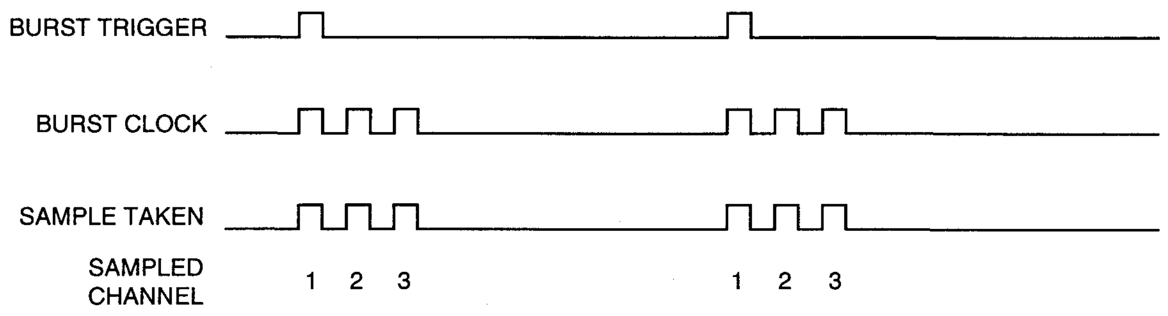


Fig. 4-5 — Timing Diagram, Burst

• Starting an A/D Conversion

Software triggered single conversions are started by writing to the START CONVERT port at BA + 0. The value you write is irrelevant. For single conversions, you must write to this port to initiate *every* conversion. Multiple conversions are triggered by the pacer clock or by a software or hardware trigger (burst mode). They are started by the first pulse present after the trigger has been enabled.

• Monitoring Conversion Status (DMA Done or End-of-Convert)

The A/D conversion status can be monitored through the DMA done flag or through the end-of-convert (EOC) bit in the STATUS port at BA + 0. When doing DMA transfers, you will want to monitor the DMA done flag for a transition from low to high. This tells you when the DMA transfer is complete and data has been placed in the PC's memory. The EOC line is available for monitoring conversion status when performing single conversions not using DMA transfer. When the EOC goes from low to high, the A/D converter has completed its conversion and the data is ready to read. The EOC line stays high following a conversion until the data has been read. Then the line goes back to low until the next conversion is complete.

• Reading the Converted Data

Two successive reads of port BA + 1 provide the LSB and MSB of the 12-bit A/D conversion in the format defined in the I/O map section at the beginning of this chapter. The LSB must always be read first, followed by the MSB. The converted data can also be read at BA + 12 (LSB) and BA + 13 (MSB). This allows you to use a read word command on an XT computer to automatically read both ports using one command.

The output code and the resolution of the conversion vary, depending on the input voltage range selected. Bipolar conversions are in twos complement form, and unipolar conversions are straight binary. When a bipolar value is read, you must first convert the result to straight binary and then calculate the voltage. The conversion formula is simple: for values greater than 2047, you must subtract 4096 from the value to get the sign of the voltage. For example, if your output is 2048, you subtract 4096: 2048 - 4096 = -2048. This result corresponds to -5 volts or -10 volts, depending on your binary range. For values of 2047 or less, you simply convert the result. The key digital codes and their input voltage values are given for each range in the three tables which follow.

A/D Bipolar Code Table (±5V; twos complement)

Input Voltage	Output Code
+4.998 volts	MSB 0111 1111 1111 LSB
+2.500 volts	0100 0000 0000
0 volts	0000 0000 0000
-.00244 volts	1111 1111 1111
-5.000 volts	1000 0000 0000
1 LSB = 2.44 millivolts	

A/D Bipolar Code Table (±10V; twos complement)

Input Voltage	Output Code
+9.995 volts	MSB 0111 1111 1111 LSB
+5.000 volts	0100 0000 0000
0 volts	0000 0000 0000
-.00488 volts	1111 1111 1111
-10.000 volts	1000 0000 0000
1 LSB = 4.88 millivolts	

A/D Unipolar Code Table (0 to +10V; straight binary)

Input Voltage	Output Code
+9.99756 volts	MSB 1111 1111 1111 LSB
+5.00000 volts	1000 0000 0000
0 volts	0000 0000 0000
1 LSB = 2.44 millivolts	

• Programming the Pacer Clock

Two of the three 16-bit timer/counters in the 8254 programmable interval timer are cascaded to form the on-board pacer clock, shown in Figure 4-6. When you want to use the pacer clock for continuous A/D conversions, you must program the clock rate. To find the value you must load into the clock to produce the desired rate, you first have to calculate the value of Divider 1 (Timer/Counter 0) and Divider 2 (Timer/Counter 1) shown in the diagram. The formulas for making this calculation are as follows:

Pacer clock frequency = Clock Source Frequency/(Divider 1 x Divider 2)
Divider 1 x Divider 2 = Clock Source Frequency/Pacer Clock Frequency

To set the pacer clock frequency at 100 kHz using the on-board 8 MHz clock source, this equation becomes:

Divider 1 x Divider 2 = 8 MHz/100 kHz ---> 80 = 8 MHz/100 kHz

After you determine the value of Divider 1 x Divider 2, you then divide the result by the least common denominator. The least common denominator is the value that is loaded into Divider 1, and the result of the division, the quotient, is loaded into Divider 2. In our example above, the least common denominator is 2, so Divider 1 equals 2, and Divider 2 equals 80/2, or 40. The table below lists some common pacer clock frequencies and the counter settings (using the on-board 8 MHz clock source).

After you calculate the decimal value of each divider, you can convert the result to a hex value if it is easier for you when loading the count into the 16-bit counter.

To set up the pacer clock on the board, follow these steps:

1. Select a clock source (the 8 MHz on-board clock or an external clock source).
2. Program Timer/Counter 0 for Mode 2 operation.
3. Program Timer/Counter 1 for Mode 2 operation.
4. Load Divider 1 LSB.
5. Load Divider 1 MSB.
6. Load Divider 2 LSB.
7. Load Divider 2 MSB.

The pacer clock starts running as soon as the last divider is loaded. A/D conversions can be started and stopped by enabling and disabling the external trigger.

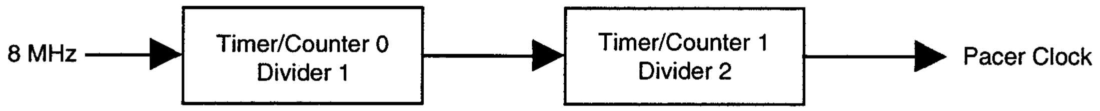


Fig. 4-6 — Pacer Clock Block Diagram

Pacer Clock	Divider 1 decimal / (hex)	Divider 2 decimal / (hex)
125 kHz	2 / (0002)	32 / (0020)
100 kHz	2 / (0002)	40 / (0028)
50 kHz	2 / (0002)	80 / (0050)
10 kHz	2 / (0002)	400 / (0190)
1 kHz	2 / (0002)	4000 / (0FA0)
100 Hz	2 / (0002)	40000 / (9C40)

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 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 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.

IRQ7	**IRQ6**	**IRQ5**	**IRQ4**	**IRQ3**	**IRQ2**	**IRQ1**	**IRQ0**	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 1210), 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 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 clear the interrupt status of the board and 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.
- Clear the interrupt bit on the board by writing any value to BA + 2.
- 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(BaseAddress + 2, 0);     /* Clear 1210 interrupt */
   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[BaseAddress + 2] := 0;      { Clear 1210 interrupt }
   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 1210 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.
- Two of the most common mistakes when writing an ISR are forgetting to clear the interrupt status of the board and forgetting to issue the EOI command to the 8259 interrupt controller before exiting the ISR.

Data Transfers Using DMA

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

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

The following steps are required when using DMA:

1. Choose a DMA channel.
2. Allocate a buffer.
3. Calculate the page and offset of the buffer.
4. Set the DMA page register.
5. Program the DMA controller.
6. Program device generating data (AD1210/ADA1210).
7. Wait until DMA is complete.
8. Disable DMA.

Each step is detailed in the following paragraphs.

• Choosing a DMA Channel

There are a number of DMA channels available on the PC for use by peripheral devices. The 1210 can use either DMA channel 1 or DMA channel 3. The factory setting is DMA disabled. You can arbitrarily choose one or the other by setting the jumpers on P5 as described in Chapter 1; in most cases either choice is fine. Occasionally though, you will have another peripheral device (for example, a tape backup or Bernoulli drive) that also uses the DMA channel you have selected. This will certainly cause erratic results and can be hard to detect. The best approach to pinpoint this problem is to read the documentation for the other peripheral devices in your system and try to determine which DMA channel each uses.

• Allocating a DMA Buffer

When using DMA, you must have a location in memory where the DMA controller will place data from the board. This buffer can be either static or dynamically allocated. Just be sure that its location will not change while DMA is in progress. The following code examples show how to allocate buffers for use with DMA.

In Pascal:

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

-or-

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

In C:

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

-or-

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

In BASIC:

```
DIM BUFFER%(5000)
```

• Calculating the Page and Offset of a Buffer

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

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

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

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

Once you've determined the segment and offset, multiply the segment by 16 and add the offset to give you the linear address. (Make sure you store this result in a long integer, or DWORD, or the results will be meaningless.) The page number is the quotient of the division of the linear address by 65536 and the offset into the page is the remainder of that division. Below are some programming examples for Pascal, C, and BASIC.

In Pascal:

```
Segment := SEG(Buffer);                    { get segment of buffer }
Offset := OFS(Buffer);                     { get offset of buffer }
Linear Address := Segment * 16 + Offset;   { calculate a linear address }
Page := LinearAddress DIV 65536;           { determine page corresponding to this linear
                                             address }
PageOffset := LinearAddress MOD 65536;     { determine offset into the page }
```

In C:

```
segment = FP_SEG(&Buffer);                 /* get segment of buffer */
offset = FP_OFS(&Buffer);                  /* get offset of buffer */
linear_address = segment * 16 + offset;    /* calculate a linear address */
page = linear_address / 65536;             /* determine page corresponding to this linear
                                              address */
page_offset = linear_address % 65536;      /* determine offset into the page */
```

In BASIC:

```
S = VARSEG(BUFFER)
O = VARPTR(BUFFER)
LA= S * 16 + O
PAGE = INT(LA / 65536)
POFF = LA - (PAGE * 65536)
```

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

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

• Setting the DMA Page Register

Oddly enough, you do not inform the DMA controller directly of the page to be used. Instead, you put the page to be used into the DMA page register which is separate from the DMA controller, as shown in the table below. The location of this register depends on the DMA channel being used.

DMA Channel	Location of Page Register
1	83/(131)
3	82/(130)

• The DMA Controller

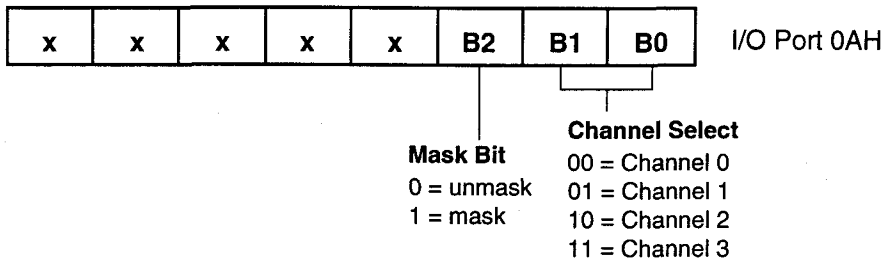
The DMA controller is a complex chip that occupies the first 16 bytes of the PC's I/O port space. A complete discussion on how it operates is beyond the scope of this manual; only relevant information is included here. The DMA controller is programmed by writing to the DMA registers in your PC. The table below lists these registers. Note that when you write 16-bit values to any of these registers (such as to the Count registers), you must write the LSB first, followed by the MSB.

If you are using DMA channel 1, write your page offset and count to ports 02H and 03H; if you are using channel 3, write your page offset and count to ports 06H and 07H. The page offset is simply the offset that you calculated for your buffer (see discussion above). Count indicates the number of bytes that you want the DMA controller to transfer. Remember that each digitized sample from the 1210 consists of 2 bytes, so the count that you write to the DMA controller should be equal to (the number of samples x 2) - 1. The single mask register and mode register are described below. The clear byte pointer sets an internal flip-flop on the DMA controller that keeps track of whether the LSB or MSB will be sent next to registers that accept both LSB and MSB. Ordinarily, you never **need** to write to this port, but it is a good habit to do so before programming the DMA controller. Writing any value to this port clears the flip-flop.

Address hex/(decimal)	Register Description
02/(02)	Channel 1 Page Offset (write 2 bytes, LSB first)
03/(03)	Channel 1 Count (write 2 bytes, LSB first)
06/(06)	Channel 3 Page Offset (write 2 bytes, LSB first)
07/(07)	Channel 3 Count (write 2 bytes, LSB first)
0A/(10)	Single Mask Register
0B/(11)	Mode Register (write only)
0C/(12)	Clear Byte Pointer Flip-Flop (write only)

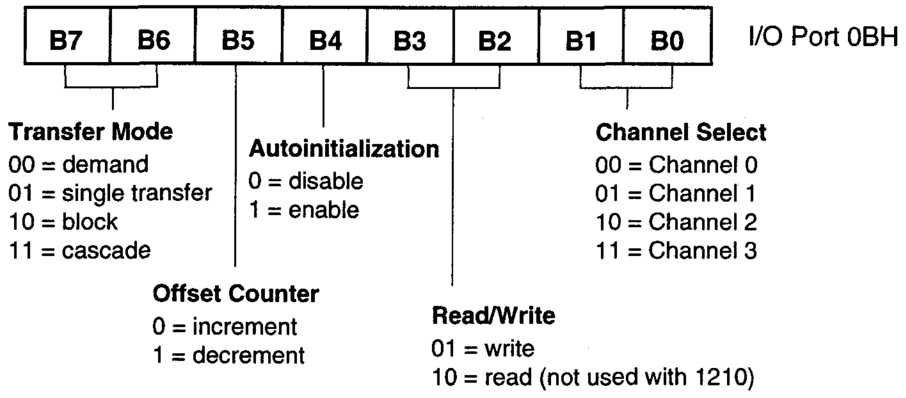
• DMA Single Mask Register

The DMA single mask register is used to enable or disable DMA on a specified DMA channel. You should mask (disable) DMA on the DMA channel you will be using while programming the DMA controller. After the DMA controller has been programmed and the 1210 has been programmed to sample data, you can enable DMA by clearing the mask bit for the DMA channel you are using. You should manually disable DMA by setting the mask bit before exiting your program or, if for some reason, sampling is halted before the DMA controller has transferred all the data it was programmed to transfer. If you leave DMA enabled and it has not transferred all the data it was programmed to transfer, it will resume transfers the next time data appears at the A/D converter. This can spell disaster if your program has ended and the buffer has be reallocated to another application.



• DMA Mode Register

The DMA mode register is used to set parameters for the DMA channel you will be using. The read/write bits are self explanatory; the read mode cannot be used with the 1210. Autoinitialization allows the DMA controller to automatically start over once it has transferred the requested number of bytes. Decrement means the DMA controller should decrement its offset counter after each transfer; the default is increment. You can use either the demand or single transfer mode when transferring data. The demand mode transfers data to the PC on demand for fastest transfer rate. The single transfer mode forces the DMA controller to relinquish every other cycle so that the processor can take care of other tasks.



• Programming the DMA Controller

To program the DMA controller, follow these steps:

1. Clear the byte pointer flip-flop.
2. Disable DMA on the channel you are using.
3. Write the DMA mode register to choose the DMA parameters.
4. Write the LSB of the page offset of your buffer.
5. Write the MSB of the page offset of your buffer.
6. Write the LSB of the number of bytes to transfer.
7. Write the MSB of the number of bytes to transfer.
8. Enable DMA on the channel you are using.

• Programming the Board for DMA

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

1. Set up the 8255 PPI for Port B output.
2. Set up the timer/counters for the desired transfer rate.
3. Enable DMA and external trigger.
4. Monitor DMA done bit.

NOTE: If the DMA is set up in the single transfer mode, each DMA transfer takes two read cycles to complete. Therefore, in single transfer, you can run the board at speeds up to about 100 kHz so the DMA transfer rate can keep up with the board's conversion rate. The demand mode supports even higher transfer rates. However, rates faster than 125 kHz, even in the demand mode, may give unreliable results.

• Monitoring for DMA Done

There are two ways to monitor for DMA done. The easiest is to poll the DMA done bit in the 1210 status register (BA +0). While DMA is in progress, the bit is clear (0). When DMA is complete, the bit is set (1). The second way to check is to use the DMA done signal to generate an interrupt. An interrupt can immediately notify your program that DMA is done and any actions can be taken as needed. Both methods are demonstrated in the sample C and Pascal programs, the polling method in the program named DMA and the interrupt method in DMASTRM.

• Common DMA Problems

- Make sure that your buffer is large enough to hold all of the data you program the DMA controller to transfer.
- Check to be sure that your buffer does not straddle a page boundary.
- Remember that the number of bytes for the DMA controller to transfer is equal to twice the number of samples. This is because each sample is two bytes in size.
- If you terminate sampling before the DMA controller has transferred the number of bytes it was programmed for, be sure to disable DMA by setting the mask bit in the single mask register.
- Make sure that the board is not running too fast for DMA transfers.

D/A Conversions (ADA1210 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. DAC1 is programmed by writing the 12-bit digital data word to BA + 8. DAC2 is identical, with the data word written to BA + 10. 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. Two of the timer/counters are cascaded and can be used for the pacer clock. The remaining timer/counter is available for your use. Figure 4-7 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 the external clock (P2-45) can be selected as the clock input to TC0 or TC2. The diagram shows how these clock sources are connected to the timer/counters.

Two gate sources are available at the I/O connector (P2-41 and P2-46). When a gate is disconnected, an on-board pull-up resistor automatically pulls the gate high, enabling the timer/counter.

The output from timer/counter 1 is available at the T/C OUT 1 pin (P2-42) and timer/counter 2's output is available at T/C 2 OUT (P2-44), where they can be used for interrupt generation, as an A/D trigger, 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.

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.

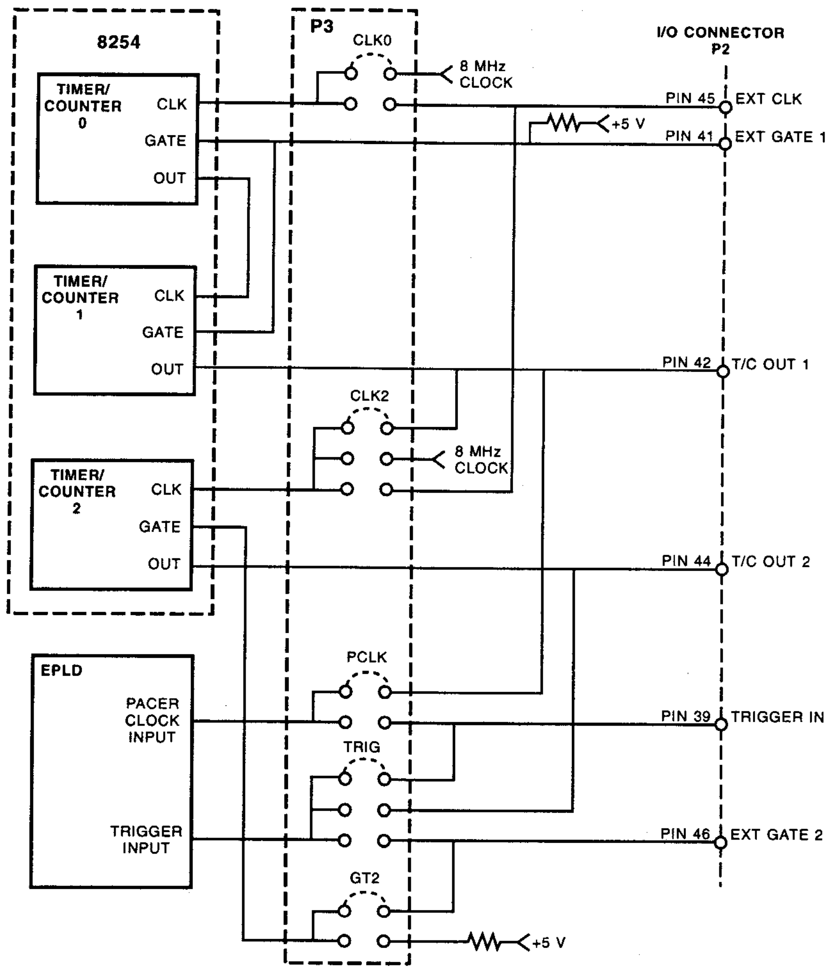


Fig. 4-7 — 8254 Programmable Interval Timer Circuit Block Diagram

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 can have pull-up or pull-down resistors installed, as described in Chapter 1.

Example Programs and Flow Diagrams

Included with the board is a set of example programs that demonstrate the use of many of the board's features. These examples are written in C, Pascal, and BASIC. Also included is an easy-to-use menu-driven diagnostics program, 1210DIAG, 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 board. In the C directory, 1210.H and 1210.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, 1210.PNC contains all of the procedures needed to implement the main Pascal programs.

Analog-to-Digital:

SOFTTRIG	Demonstrates how to use the software trigger mode for acquiring data.
EXTTRIG	Similar to SOFTTRIG except that an external trigger is used.
MULTI	Shows how to fill an array with data using a hardware trigger.

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.
INTSTRM	A complete program showing interrupt-based streaming to disk.

DMA:

DMA	Demonstrates how to use DMA to transfer acquired data to a memory buffer. Buffer can be written to disk and viewed with the included VIEWDAT program.
DMASTRM	Demonstrates how to use DMA for disk streaming. Very high continuous acquisition rates can be obtained.

BASIC Programs

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

Analog-to-Digital:

SINGLE	Demonstrates how to use the single convert, internal trigger mode for acquiring data.
EXTTRIG	Demonstrates how to use the external trigger to acquire data.
SCAN	Demonstrates how to scan channels to acquire 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.

DMA:

DMA	Shows how to take samples and transfer them to PC memory using DMA.

Flow Diagrams

The following paragraphs provide descriptions and flow diagrams for some of the 1210's A/D and D/A conversion functions. These diagrams will help you to build your own custom applications programs.

• **Single Convert Flow Diagram (Figure 4-8)**

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 at the same gain 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 at a different gain.

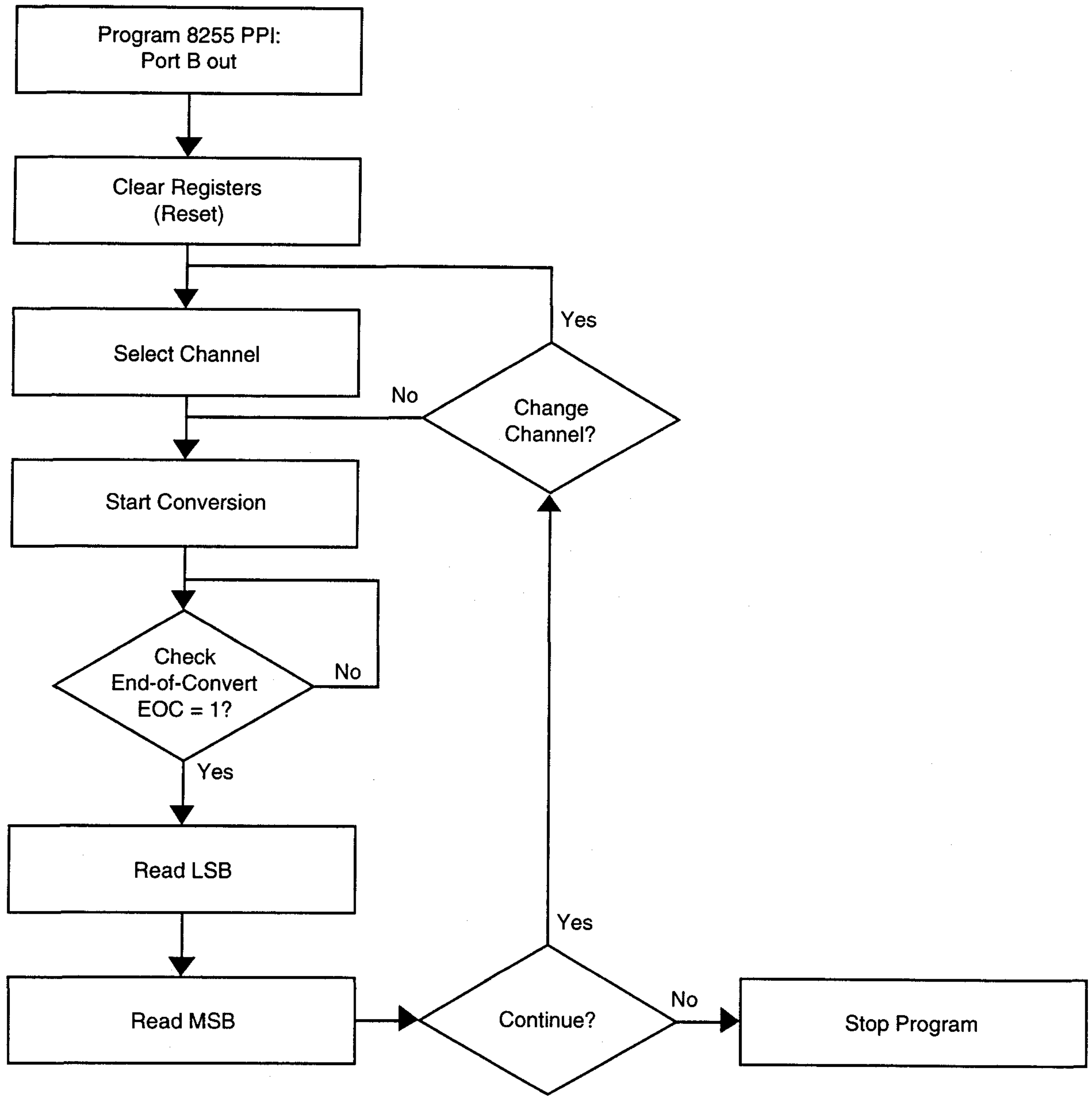


Fig. 4-8 — Single Conversion Flow Diagram

• DMA Flow Diagram (Figure 4-9)

This flow diagram shows you how to take samples and transfer the data directly into the computer's memory. You can use DMA channel 1 or 3 to transfer data to the computer's memory. The pacer clock can be used to set the sampling interval.

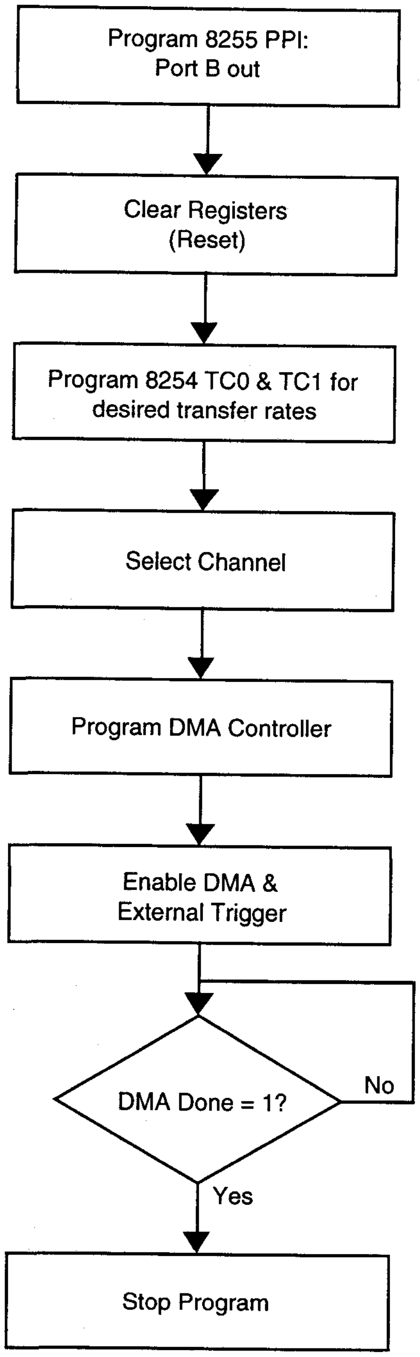


Fig. 4-9 — DMA Flow Diagram

• **Scan Flow Diagram (Figure 4-10)**

This flow diagram shows you how to take samples from a sequence of channels without selecting the channel each time a conversion is started. The scan mode is enabled and the number of channels sampled is set by the data written to BA + 3, and the starting channel is set by the data written to BA + 5.

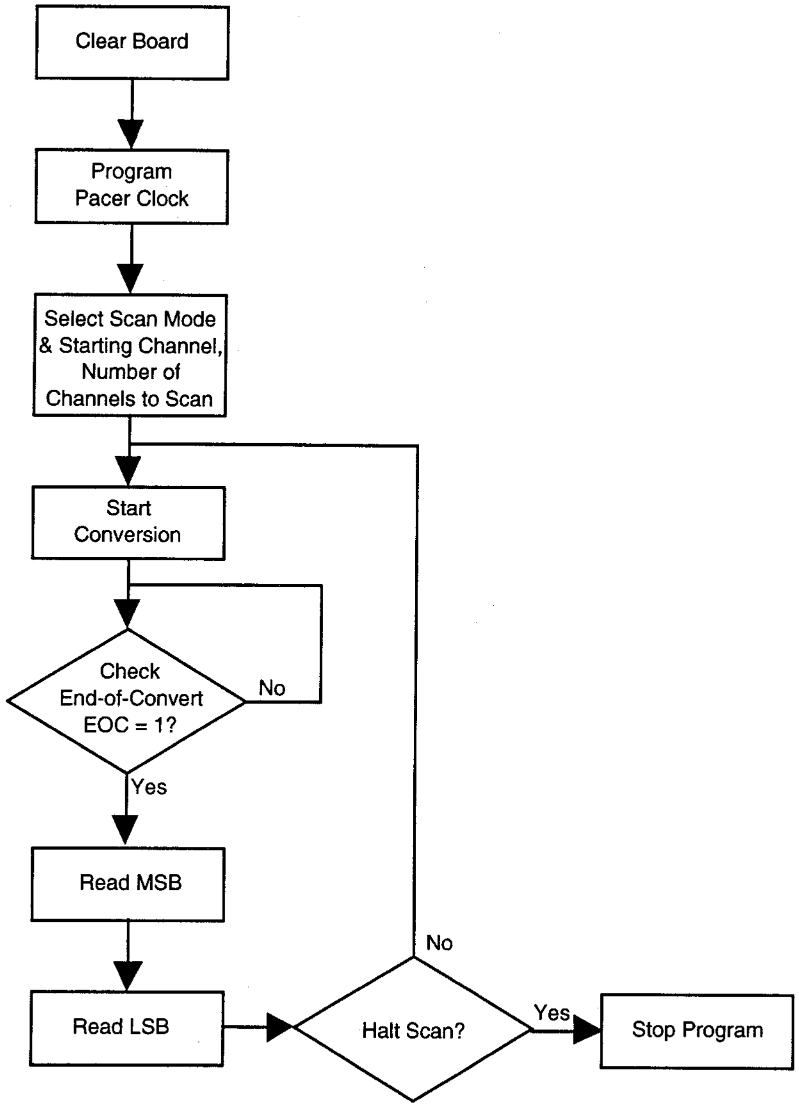


Fig. 4-10 — Scan Flow Diagram

• **Interrupts Flow Diagram (Figure 4-11)**

This flow diagram shows you how to program an interrupt routine for your board. The diagram parallels the interrupts discussion included earlier in this chapter. You can use this diagram in conjunction with the detailed text in this chapter to develop an interrupt program for your board.

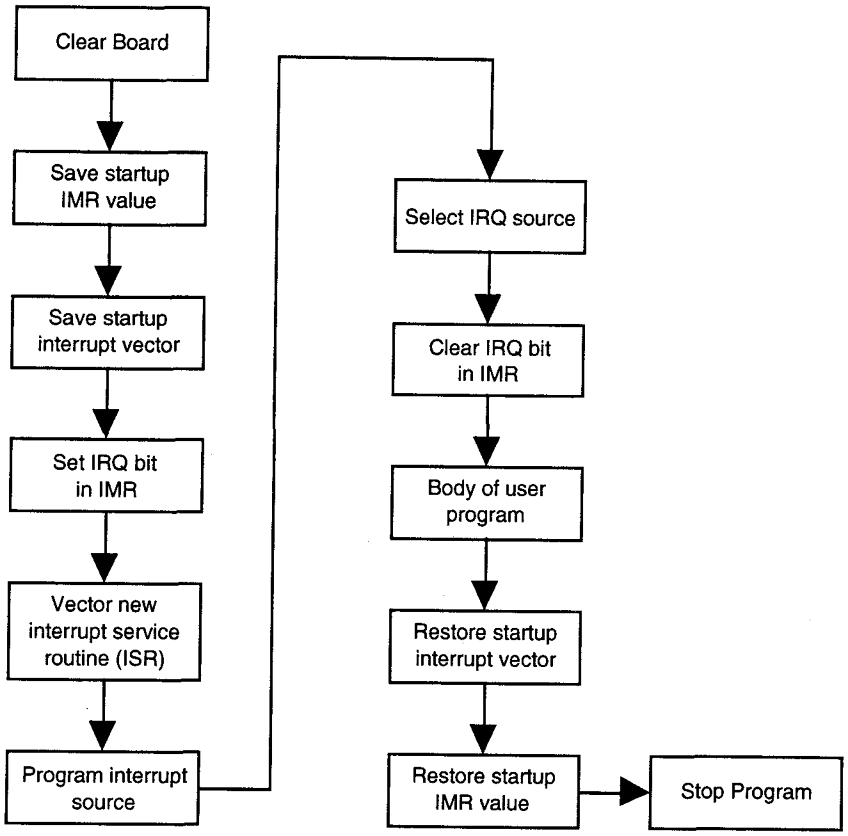


Fig. 4-11 — Interrupts Flow Diagram

• **D/A Conversion Flow Diagram (Figure 4-12)**

This flow diagram shows you how to generate a voltage output through the D/A converter (ADA1210 only). A conversion is initiated each time the high byte (MSB) is written to the D/A converter.

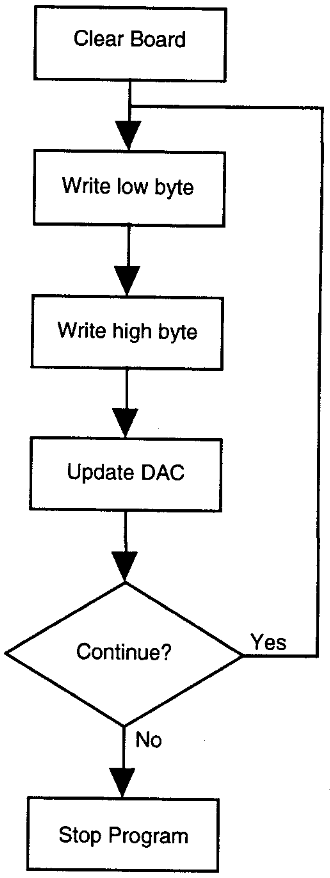


Fig. 4-12 — D/A Conversion Flow Diagram

CHAPTER 5

CALIBRATION

This chapter tells you how to calibrate the AD1210/ADA1210 using the 1210DIAG calibration program included in the example software package and six 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 (ADA1210 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 1210DIAG 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 1210DIAG 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.

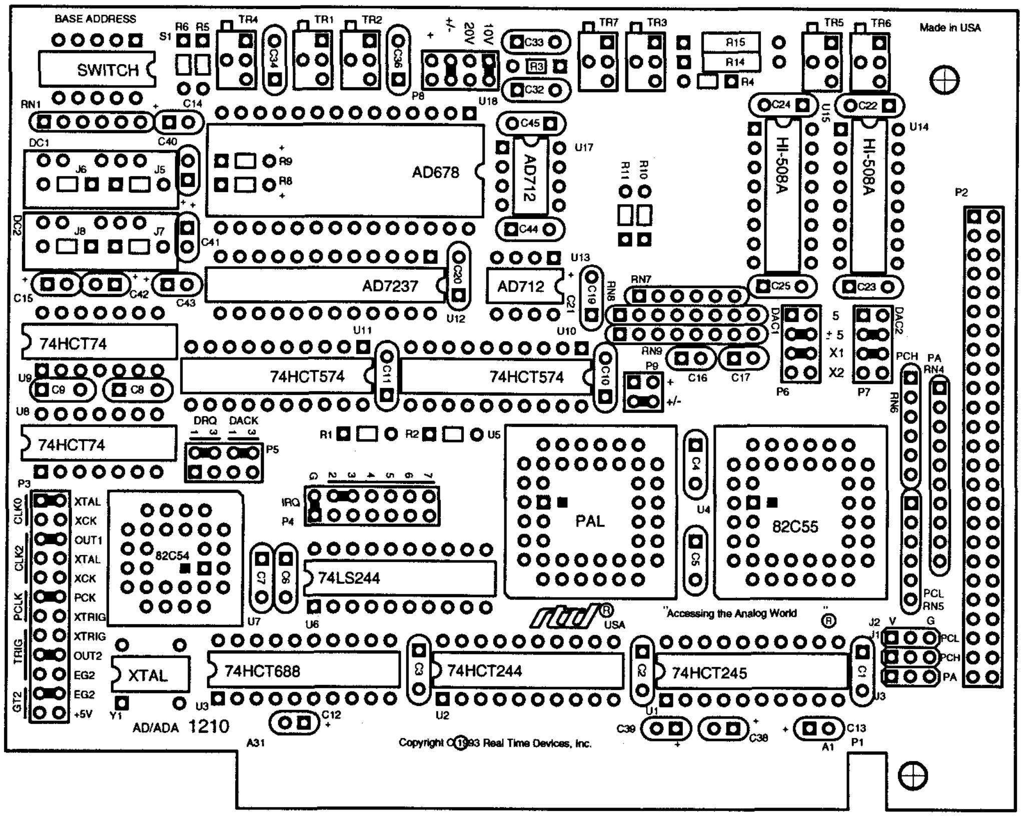


Fig. 5-1 — Board Layout

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 the unipolar, straight binary range, and Table 5-2 shows the ideal voltage for each bit weight for the bipolar, twos complement 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 TR4 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 P8 are set for 10V and +, and the jumper on P9 is set for +.

Use analog input channel 1 and set it for a gain of 1 while calibrating the board. Connect your precision voltage source to channel 1. Set the voltage source to +1.22070 millivolts, start a conversion, and read the resulting data. Adjust trimpot TR4 until 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 on the 1210 board, and ensures accurate calibration results.

CAUTION: When using the 0 to +10 volt range, you must have ±12 volts DC supplied to the board for accurate results at the extremes of these ranges. Any supplies less than ±12 volts will degrade performance.

Table 5-1 — A/D Converter Bit Weights, Unipolar, Straight Binary

A/D Bit Weight	Ideal Input Voltage (millivolts)
	0 to +10 Volts
1111 1111 1111	+9997.6
1000 0000 0000	+5000.0
0100 0000 0000	+2500.0
0010 0000 0000	+1250.0
0001 0000 0000	+625.00
0000 1000 0000	+312.50
0000 0100 0000	+156.250
0000 0010 0000	+78.125
0000 0001 0000	+39.063
0000 0000 1000	+19.5313
0000 0000 0100	+9.7656
0000 0000 0010	+4.8828
0000 0000 0001	+2.4414
0000 0000 0000	+0.0000

Data Values for Calibrating Unipolar 10 Volt Range (0 to +10 volts)		
	Offset (TR4) Input Voltage = +1.22070 mV	Converter Gain (TR1) Input Voltage = +9.49829 V
A/D Converted Data	0000 0000 0000 0000 0000 0001	1111 0011 0010 1111 0011 0011

Bipolar Calibration

• Bipolar Range Adjustments: -5 to +5 Volts

Two adjustments are made to calibrate the A/D converter for the bipolar range of -5 to +5 volts. One is the offset adjustment, and the other is the full scale, or gain, adjustment. Trimpot TR2 is used to make the offset adjustment, and trimpot TR1 is used for gain adjustment. Before making these adjustments, make sure that the jumpers on P8 are set for 10V and +/-, and the jumper on P9 is set for +/-.

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 10 Volt Range (-5 to +5 volts)		
	Offset (TR2) Input Voltage = -4.99878V	Converter Gain (TR1) Input Voltage = +4.99634V
A/D Converted Data	1000 0000 0000 1000 0000 0001	0111 1111 1110 0111 1111 1111

Table 5-2 — A/D Converter Bit Weights, Bipolar, Twos Complement

	Ideal Input Voltage (millivolts)	
A/D Bit Weight	-5 to +5 Volts	-10 to +10 Volts
1111 1111 1111	-2.44	-4.88
1000 0000 0000	-5000.00	-10000.00
0100 0000 0000	+2500.00	+5000.00
0010 0000 0000	+1250.00	+2500.00
0001 0000 0000	+625.00	+1250.00
0000 1000 0000	+312.50	+625.00
0000 0100 0000	+156.25	+312.50
0000 0010 0000	+78.13	+156.25
0000 0001 0000	+39.06	+78.13
0000 0000 1000	+19.53	+39.06
0000 0000 0100	+9.77	+19.53
0000 0000 0010	+4.88	+9.77
0000 0000 0001	+2.44	+4.88
0000 0000 0000	0.00	0.00

• **Bipolar Range Adjustments: -10 to +10 Volts**

To adjust the bipolar 20-volt range (-10 to +10 volts), change the range jumper on P8 so that it is installed across the 20V pins. Leave the other jumper on P8 at +/- and make sure the P9 jumper is on +/-. Then, set the input voltage to +5.0000 volts and adjust TR3 until the output matches the data in the table below.

CAUTION: When using the ±10 volt range, you must have ±12 volts DC supplied to the board for accurate results at the extremes of these ranges. Any supplies less than ±12 volts will degrade performance.

Data Value for Calibrating Bipolar 20 Volt Range (-10 to +10 volts)	
	TR3 **Input Voltage = +5.0000V**
A/D Converted Data	0100 0000 0000

D/A Calibration (ADA1210)

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 P6, AOUT1, or P7, 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 5-3 lists the ideal output voltages per bit weight for unipolar ranges and Table 5-4 lists the ideal output voltages for bipolar ranges.

Table 5-3 — 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 5-4 — 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

AD1210/ADA1210 SPECIFICATIONS

AD1210/ADA1210 Characteristics Typical @ 25° C

Interface

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

Analog Input

16 single-ended inputs

Input impedance, each channel	>10 megohms
Gain	resistor configurable
Input ranges	±5, ±10, or 0 to +10 volts
Guaranteed linearity across input ranges	2210: ±5, ±9.5, and 0 to +9.5 volts 2310: ±5, ±10, and 0 to +10 volts
Overvoltage protection	±35 Vdc
Common mode input voltage	±10 volts, max
Settling time (gain = 1)	5 µsec, max

CAUTION: When using the 0 to +10 or ±10 volt range, you must have ±12 volts DC supplied to the board for accurate results at the extremes of these ranges. Any supplies less than ±12 volts will degrade performance.

A/D Converter AD678

Type	Successive approximation
Resolution	12 bits (2.44 mV @ 10V; 4.88 mV @ 20V)
Linearity	±1 LSB, typ
Conversion speed	5 µsec, typ
Throughput	125 kHz

Pacer Clock

Range (using on-board 8 MHz clock)	9 minutes to 5 µsec

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 (ADA1210 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 LSB, max
Full-scale accuracy	±5 LSB, max
Non-linearity	±1 LSB, 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 box header

Environmental

Operating temperature	0 to +70°C
Storage temperature	-40 to +85°C
Humidity	0 to 90% non-condensing

Size

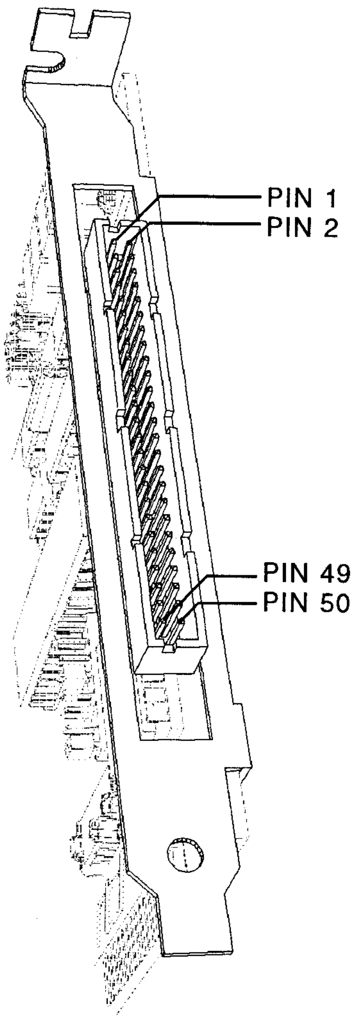
Short slot — 3.875"H x 5.25"W (99mm x 134mm)

APPENDIX B

P2 CONNECTOR PIN ASSIGNMENTS

P2 Connector:

AIN1	1	2	AIN9
AIN2	3	4	AIN10
AIN3	5	6	AIN11
AIN4	7	8	AIN12
AIN5	9	10	AIN13
AIN6	11	12	AIN14
AIN7	13	14	AIN15
AIN8	15	16	AIN16
AOUT 1	17	18	ANALOG GND
AOUT 2	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®

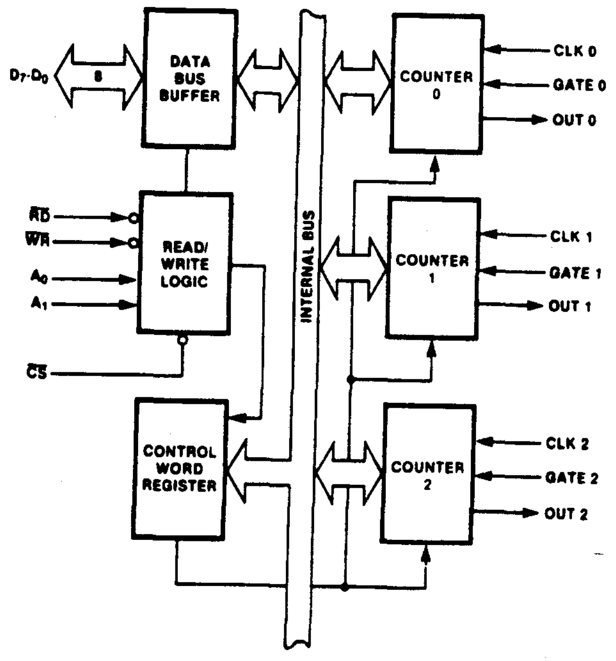
82C54
CHMOS PROGRAMMABLE INTERVAL TIMER

- Compatible with all Intel and most other microprocessors
- High Speed, "Zero Wait State" Operation with 8 MHz 8086/88 and 80186/188
- Handles Inputs from DC to 8 MHz
  — 10 MHz for 82C54-2
- Available in EXPRESS
  — Standard Temperature Range
  — Extended Temperature Range
- Three independent 16-bit counters
- Low Power CHMOS
  — I_{CC} = 10 mA @ 8 MHz Count frequency
- Completely TTL Compatible
- Six Programmable Counter Modes
- Binary or BCD counting
- Status Read Back Command
- Available in 24-Pin DIP and 28-Pin PLCC

The Intel 82C54 is a high-performance, CHMOS version of the industry standard 8254 counter/timer which is designed to solve the timing control problems common in microcomputer system design. It provides three independent 16-bit counters, each capable of handling clock inputs up to 10 MHz. All modes are software programmable. The 82C54 is pin compatible with the HMOS 8254, and is a superset of the 8253.

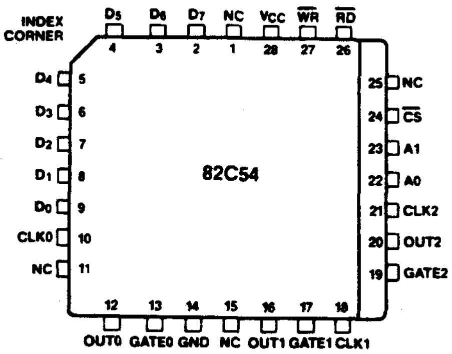
Six programmable timer modes allow the 82C54 to be used as an event counter, elapsed time indicator, programmable one-shot, and in many other applications.

The 82C54 is fabricated on Intel's advanced CHMOS III technology which provides low power consumption with performance equal to or greater than the equivalent HMOS product. The 82C54 is available in 24-pin DIP and 28-pin plastic leaded chip carrier (PLCC) packages.



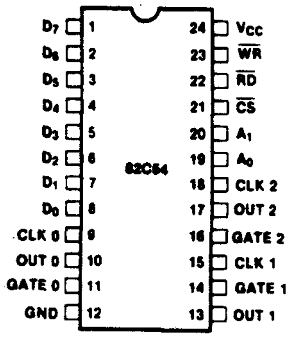
231244-1

Figure 1. 82C54 Block Diagram



231244-3

PLASTIC LEADED CHIP CARRIER



231244-2

Diagrams are for pin reference only.
Package sizes are not to scale.

Figure 2. 82C54 Pinout

September 1989
Order Number: 231244-005

Table 1. Pin Description

Symbol	Pin Number		Type	Function
	DIP	PLCC		
D_7-D_0	1-8	2-9	I/O	Data: Bidirectional tri-state data bus lines, connected to system data bus.
CLK 0	9	10	I	Clock 0: Clock input of Counter 0.
OUT 0	10	12	O	Output 0: Output of Counter 0.
GATE 0	11	13	I	Gate 0: Gate input of Counter 0.
GND	12	14		Ground: Power supply connection.
OUT 1	13	16	O	Out 1: Output of Counter 1.
GATE 1	14	17	I	Gate 1: Gate input of Counter 1.
CLK 1	15	18	I	Clock 1: Clock input of Counter 1.
GATE 2	16	19	I	Gate 2: Gate input of Counter 2.
OUT 2	17	20	O	Out 2: Output of Counter 2.
CLK 2	18	21	I	Clock 2: Clock input of Counter 2.
A_1, A_0	20-19	23-22	I	Address: Used to select one of the three Counters or the Control Word Register for read or write operations. Normally connected to the system address bus. A_1 / A_0 / Selects 0 / 0 / Counter 0 0 / 1 / Counter 1 1 / 0 / Counter 2 1 / 1 / Control Word Register
$\overline{CS}$	21	24	I	Chip Select: A low on this input enables the 82C54 to respond to $\overline{RD}$ and $\overline{WR}$ signals. $\overline{RD}$ and $\overline{WR}$ are ignored otherwise.
$\overline{RD}$	22	26	I	Read Control: This input is low during CPU read operations.
$\overline{WR}$	23	27	I	Write Control: This input is low during CPU write operations.
V_{CC}	24	28		Power: +5V power supply connection.
NC		1, 11, 15, 25		No Connect

FUNCTIONAL DESCRIPTION

General

The 82C54 is a programmable interval timer/counter designed for use with Intel microcomputer systems. It is a general purpose, multi-timing element that can be treated as an array of I/O ports in the system software.

The 82C54 solves one of the most common problems in any microcomputer system, the generation of accurate time delays under software control. Instead of setting up timing loops in software, the programmer configures the 82C54 to match his requirements and programs one of the counters for the desired delay. After the desired delay, the 82C54 will interrupt the CPU. Software overhead is minimal and variable length delays can easily be accommodated.

Some of the other counter/timer functions common to microcomputers which can be implemented with the 82C54 are:

- Real time clock
- Even counter
- Digital one-shot
- Programmable rate generator
- Square wave generator
- Binary rate multiplier
- Complex waveform generator
- Complex motor controller

Block Diagram

DATA BUS BUFFER

This 3-state, bi-directional, 8-bit buffer is used to interface the 82C54 to the system bus (see Figure 3).

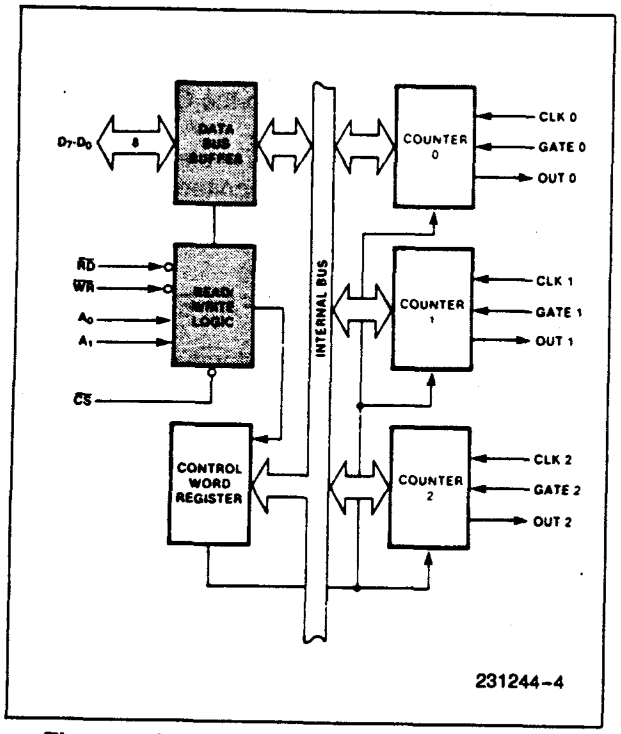


Figure 3. Block Diagram Showing Data Bus Buffer and Read/Write Logic Functions

READ/WRITE LOGIC

The Read/Write Logic accepts inputs from the system bus and generates control signals for the other functional blocks of the 82C54. A_1 and A_0 select one of the three counters or the Control Word Register to be read from/written into. A "low" on the $\overline{RD}$ input tells the 82C54 that the CPU is reading one of the counters. A "low" on the $\overline{WR}$ input tells the 82C54 that the CPU is writing either a Control Word or an initial count. Both $\overline{RD}$ and $\overline{WR}$ are qualified by $\overline{CS}$; $\overline{RD}$ and $\overline{WR}$ are ignored unless the 82C54 has been selected by holding $\overline{CS}$ low.

CONTROL WORD REGISTER

The Control Word Register (see Figure 4) is selected by the Read/Write Logic when $A_1, A_0 = 11$. If the CPU then does a write operation to the 82C54, the data is stored in the Control Word Register and is interpreted as a Control Word used to define the operation of the Counters.

The Control Word Register can only be written to; status information is available with the Read-Back Command.

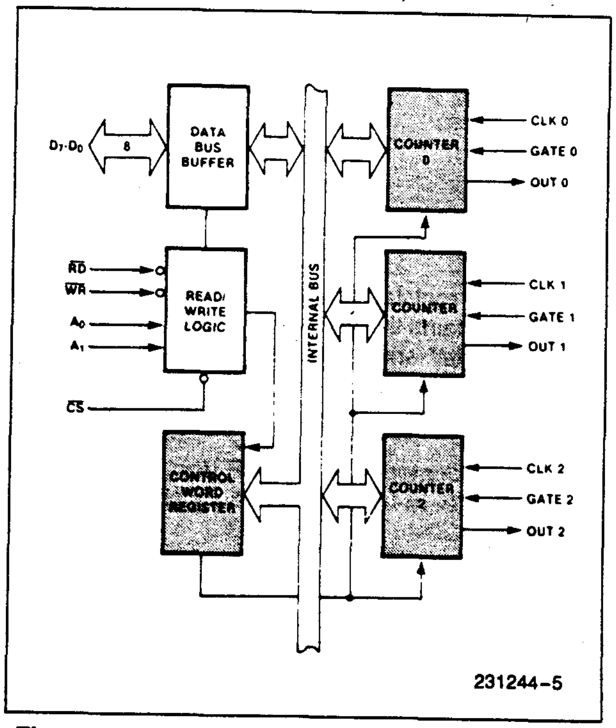


Figure 4. Block Diagram Showing Control Word Register and Counter Functions

COUNTER 0, COUNTER 1, COUNTER 2

These three functional blocks are identical in operation, so only a single Counter will be described. The internal block diagram of a single counter is shown in Figure 5.

The Counters are fully independent. Each Counter may operate in a different Mode.

The Control Word Register is shown in the figure; it is not part of the Counter itself, but its contents determine how the Counter operates.

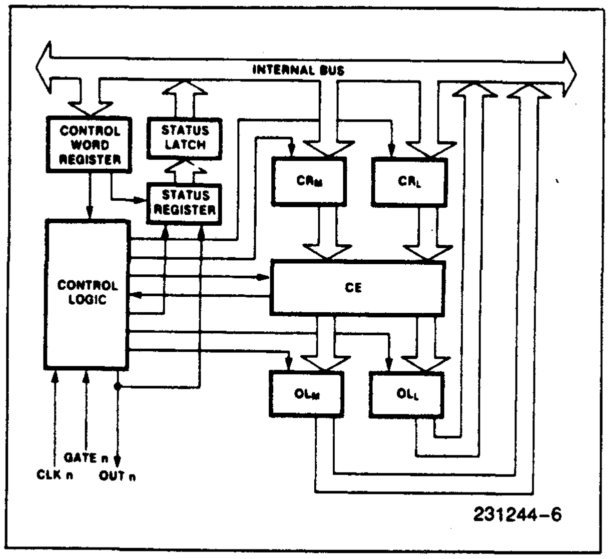


Figure 5. Internal Block Diagram of a Counter

The status register, shown in the Figure, when latched, contains the current contents of the Control Word Register and status of the output and null count flag. (See detailed explanation of the Read-Back command.)

The actual counter is labelled CE (for "Counting Element"). It is a 16-bit presettable synchronous down counter.

OL_M and OL_L are two 8-bit latches. OL stands for "Output Latch"; the subscripts M and L stand for "Most significant byte" and "Least significant byte" *respectively*. Both are normally referred to as one unit and called just OL. These latches normally "follow" the CE, but if a suitable Counter Latch Command is sent to the 82C54, the latches "latch" the present count until read by the CPU and then return to "following" the CE. One latch at a time is enabled by the counter's Control Logic to drive the internal bus. This is how the 16-bit Counter communicates over the 8-bit internal bus. Note that the CE itself cannot be read; whenever you read the count, it is the OL that is being read.

Similarly, there are two 8-bit registers called CR_M and CR_L (for "Count Register"). Both are normally referred to as one unit and called just CR. When a new count is written to the Counter, the count is stored in the CR and later transferred to the CE. The Control Logic allows one register at a time to be loaded from the internal bus. Both bytes are transferred to the CE simultaneously. CR_M and CR_L are cleared when the Counter is programmed. In this way, if the Counter has been programmed for one byte counts (either most significant byte only or least significant byte only) the other byte will be zero. Note that the CE cannot be written into; whenever a count is written, it is written into the CR.

The Control Logic is also shown in the diagram. CLK n, GATE n, and OUT n are all connected to the outside world through the Control Logic.

82C54 SYSTEM INTERFACE

The 82C54 is treated by the systems software as an array of peripheral I/O ports; three are counters and the fourth is a control register for MODE programming.

Basically, the select inputs A_0, A_1 connect to the A_0, A_1 address bus signals of the CPU. The $\overline{CS}$ can be derived directly from the address bus using a linear select method. Or it can be connected to the output of a decoder, such as an Intel 8205 for larger systems.

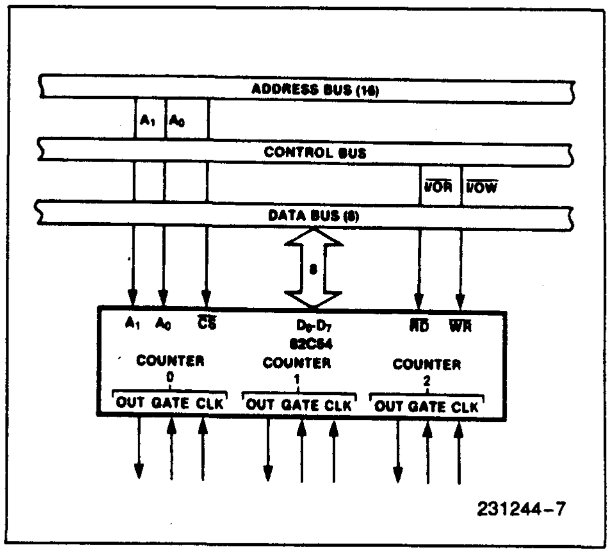


Figure 6. 82C54 System Interface

OPERATIONAL DESCRIPTION

General

After power-up, the state of the 82C54 is undefined. The Mode, count value, and output of all Counters are undefined.

How each Counter operates is determined when it is programmed. Each Counter must be programmed before it can be used. Unused counters need not be programmed.

Programming the 82C54

Counters are programmed by writing a Control Word and then an initial count. The control word format is shown in Figure 7.

All Control Words are written into the Control Word Register, which is selected when $A_1, A_0 = 11$. The Control Word itself specifies which Counter is being programmed.

By contrast, initial counts are written into the Counters, not the Control Word Register. The A_1, A_0 inputs are used to select the Counter to be written into. The format of the initial count is determined by the Control Word used.

Control Word Format

$A_1, A_0 = 11$ $\overline{CS} = 0$ $\overline{RD} = 1$ $\overline{WR} = 0$

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC1	SC0	RW1	RW0	M2	M1	M0	BCD

SC — Select Counter:

SC1	SC0	
0	0	Select Counter 0
0	1	Select Counter 1
1	0	Select Counter 2
1	1	Read-Back Command (See Read Operations)

RW — Read/Write:

RW1	RW0	
0	0	Counter Latch Command (see Read Operations)
0	1	Read/Write least significant byte only.
1	0	Read/Write most significant byte only.
1	1	Read/Write least significant byte first, then most significant byte.

M — MODE:

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

BCD:

0	Binary Counter 16-bits
1	Binary Coded Decimal (BCD) Counter (4 Decades)

NOTE: Don't care bits (X) should be 0 to insure compatibility with future Intel products.

Figure 7. Control Word Format

Write Operations

The programming procedure for the 82C54 is very flexible. Only two conventions need to be remembered:

1) For each Counter, the Control Word must be written before the initial count is written.
2) The initial count must follow the count format specified in the Control Word (least significant byte only, most significant byte only, or least significant byte and then most significant byte).

Since the Control Word Register and the three Counters have separate addresses (selected by the A_1, A_0 inputs), and each Control Word specifies the Counter it applies to (SC0, SC1 bits), no special instruction sequence is required. Any programming sequence that follows the conventions above is acceptable.

A new initial count may be written to a Counter at any time without affecting the Counter's programmed Mode in any way. Counting will be affected as described in the Mode definitions. The new count must follow the programmed count format.

If a Counter is programmed to read/write two-byte counts, the following precaution applies: A program must not transfer control between writing the first and second byte to another routine which also writes into that same Counter. Otherwise, the Counter will be loaded with an incorrect count.

		A_1	A_0
Control Word —	Counter 0	1	1
LSB of count —	Counter 0	0	0
MSB of count —	Counter 0	0	0
Control Word —	Counter 1	1	1
LSB of count —	Counter 1	0	1
MSB of count —	Counter 1	0	1
Control Word —	Counter 2	1	1
LSB of count —	Counter 2	1	0
MSB of count —	Counter 2	1	0

		A_1	A_0
Control Word —	Counter 2	1	1
Control Word —	Counter 1	1	1
Control Word —	Counter 0	1	1
LSB of count —	Counter 2	1	0
MSB of count —	Counter 2	1	0
LSB of count —	Counter 1	0	1
MSB of count —	Counter 1	0	1
LSB of count —	Counter 0	0	0
MSB of count —	Counter 0	0	0

		A_1	A_0
Control Word —	Counter 0	1	1
Counter Word —	Counter 1	1	1
Control Word —	Counter 2	1	1
LSB of count —	Counter 2	1	0
LSB of count —	Counter 1	0	1
LSB of count —	Counter 0	0	0
MSB of count —	Counter 0	0	0
MSB of count —	Counter 1	0	1
MSB of count —	Counter 2	1	0

		A_1	A_0
Control Word —	Counter 1	1	1
Control Word —	Counter 0	1	1
LSB of count —	Counter 1	0	1
Control Word —	Counter 2	1	1
LSB of count —	Counter 0	0	0
MSB of count —	Counter 1	0	1
LSB of count —	Counter 2	1	0
MSB of count —	Counter 0	0	0
MSB of count —	Counter 2	1	0

NOTE:
In all four examples, all counters are programmed to read/write two-byte counts.
These are only four of many possible programming sequences.

Figure 8. A Few Possible Programming Sequences

Read Operations

It is often desirable to read the value of a Counter without disturbing the count in progress. This is easily done in the 82C54.

There are three possible methods for reading the counters: a simple read operation, the Counter Latch Command, and the Read-Back Command. Each is explained below. The first method is to perform a simple read operation. To read the Counter, which is selected with the A1, A0 inputs, the CLK input of the selected Counter must be inhibited by using either the GATE input or external logic. Otherwise, the count may be in the process of changing when it is read, giving an undefined result.

COUNTER LATCH COMMAND

The second method uses the "Counter Latch Command". Like a Control Word, this command is written to the Control Word Register, which is selected when $A_1, A_0 = 11$. Also like a Control Word, the SC0, SC1 bits select one of the three Counters, but two other bits, D5 and D4, distinguish this command from a Control Word.

$A_1, A_0 = 11$; $\overline{CS} = 0$; $\overline{RD} = 1$; $\overline{WR} = 0$

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC1	SC0	0	0	X	X	X	X

SC1, SC0 - specify counter to be latched

SC1	SC0	Counter
0	0	0
0	1	1
1	0	2
1	1	Read-Back Command

D5,D4 - 00 designates Counter Latch Command

X - don't care

NOTE:
Don't care bits (X) should be 0 to insure compatibility with future Intel products.

Figure 9. Counter Latching Command Format

The selected Counter's output latch (OL) latches the count at the time the Counter Latch Command is received. This count is held in the latch until it is read by the CPU (or until the Counter is reprogrammed). The count is then unlatched automatically and the OL returns to "following" the counting element (CE). This allows reading the contents of the Counters "on the fly" without affecting counting in progress. Multiple Counter Latch Commands may be used to latch more than one Counter. Each latched Counter's OL holds its count until it is read. Counter Latch Commands do not affect the programmed Mode of the Counter in any way.

If a Counter is latched and then, some time later, latched again before the count is read, the second Counter Latch Command is ignored. The count read will be the count at the time the first Counter Latch Command was issued.

With either method, the count must be read according to the programmed format; specifically, if the Counter is programmed for two byte counts, two bytes must be read. The two bytes do not have to be read one right after the other; read or write or programming operations of other Counters may be inserted between them.

Another feature of the 82C54 is that reads and writes of the same Counter may be interleaved; for example, if the Counter is programmed for two byte counts, the following sequence is valid.

1. Read least significant byte.
2. Write new least significant byte.
3. Read most significant byte.
4. Write new most significant byte.

If a Counter is programmed to read/write two-byte counts, the following precaution applies; A program must not transfer control between reading the first and second byte to another routine which also reads from that same Counter. Otherwise, an incorrect count will be read.

READ-BACK COMMAND

The third method uses the Read-Back command. This command allows the user to check the count value, programmed Mode, and current state of the OUT pin and Null Count flag of the selected counter(s).

The command is written into the Control Word Register and has the format shown in Figure 10. The command applies to the counters selected by setting their corresponding bits D3,D2,D1 = 1.

A0, A1 = 11 $\overline{CS} = 0$ $\overline{RD} = 1$ $\overline{WR} = 0$

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	1	$\overline{\text{COUNT}}$	$\overline{\text{STATUS}}$	CNT 2	CNT 1	CNT 0	0

D_5: 0 = Latch count of selected counter(s)
D_4: 0 = Latch status of selected counter(s)
D_3: 1 = Select counter 2
D_2: 1 = Select counter 1
D_1: 1 = Select counter 0
D_0: Reserved for future expansion; must be 0

Figure 10. Read-Back Command Format

The read-back command may be used to latch multiple counter output latches (OL) by setting the $\overline{\text{COUNT}}$ bit D5=0 and selecting the desired counter(s). This single command is functionally equivalent to several counter latch commands, one for each counter latched. Each counter's latched count is held until it is read (or the counter is reprogrammed). That counter is automatically unlatched when read, but other counters remain latched until they are read. If multiple count read-back commands are issued to the same counter without reading the count, all but the first are ignored; i.e., the count which will be read is the count at the time the first read-back command was issued.

The read-back command may also be used to latch status information of selected counter(s) by setting $\overline{\text{STATUS}}$ bit D4=0. Status must be latched to be read; status of a counter is accessed by a read from that counter.

The counter status format is shown in Figure 11. Bits D5 through D0 contain the counter's programmed Mode exactly as written in the last Mode Control Word. OUTPUT bit D7 contains the current state of the OUT pin. This allows the user to monitor the counter's output via software, possibly eliminating some hardware from a system.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
OUTPUT	NULL COUNT	RW1	RW0	M2	M1	M0	BCD

D_7	1 =	Out Pin is 1
	0 =	Out Pin is 0
D_6	1 =	Null count
	0 =	Count available for reading
D_5-D_0		Counter Programmed Mode (See Figure 7)

Figure 11. Status Byte

NULL COUNT bit D6 indicates when the last count written to the counter register (CR) has been loaded into the counting element (CE). The exact time this happens depends on the Mode of the counter and is described in the Mode Definitions, but until the count is loaded into the counting element (CE), it can't be read from the counter. If the count is latched or read before this time, the count value will not reflect the new count just written. The operation of Null Count is shown in Figure 12.

THIS ACTION:	CAUSES:
A. Write to the control word register:[1]	Null count = 1
B. Write to the count register (CR);[2]	Null count = 1
C. New count is loaded into CE (CR → CE);	Null count = 0

[1] Only the counter specified by the control word will have its null count set to 1. Null count bits of other counters are unaffected.

[2] If the counter is programmed for two-byte counts (least significant byte then most significant byte) null count goes to 1 when the second byte is written.

Figure 12. Null Count Operation

If multiple status latch operations of the counter(s) are performed without reading the status, all but the first are ignored; i.e., the status that will be read is the status of the counter at the time the first status read-back command was issued.

Both count and status of the selected counter(s) may be latched simultaneously by setting both $\overline{\text{COUNT}}$ and $\overline{\text{STATUS}}$ bits D5,D4=0. This is functionally the same as issuing two separate read-back commands at once, and the above discussions apply here also. Specifically, if multiple count and/or status read-back commands are issued to the same counter(s) without any intervening reads, all but the first are ignored. This is illustrated in Figure 13.

If both count and status of a counter are latched, the first read operation of that counter will return latched status, regardless of which was latched first. The next one or two reads (depending on whether the counter is programmed for one or two type counts) return latched count. Subsequent reads return unlatched count.

Command								Description	Results
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0		
1	1	0	0	0	0	1	0	Read back count and status of Counter 0	Count and status latched for Counter 0
1	1	1	0	0	1	0	0	Read back status of Counter 1	Status latched for Counter 1
1	1	1	0	1	1	0	0	Read back status of Counters 2, 1	Status latched for Counter 2, but not Counter 1
1	1	0	1	1	0	0	0	Read back count of Counter 2	Count latched for Counter 2
1	1	0	0	0	1	0	0	Read back count and status of Counter 1	Count latched for Counter 1, but not status
1	1	1	0	0	0	1	0	Read back status of Counter 1	Command ignored, status already latched for Counter 1

Figure 13. Read-Back Command Example

$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	A_1	A_0	
0	1	0	0	0	Write into Counter 0
0	1	0	0	1	Write into Counter 1
0	1	0	1	0	Write into Counter 2
0	1	0	1	1	Write Control Word
0	0	1	0	0	Read from Counter 0
0	0	1	0	1	Read from Counter 1
0	0	1	1	0	Read from Counter 2
0	0	1	1	1	No-Operation (3-State)
1	X	X	X	X	No-Operation (3-State)
0	1	1	X	X	No-Operation (3-State)

Figure 14. Read/Write Operations Summary

Mode Definitions

The following are defined for use in describing the operation of the 82C54.

CLK PULSE: a rising edge, then a falling edge, in that order, of a Counter's CLK input.

TRIGGER: a rising edge of a Counter's GATE input.

COUNTER LOADING: the transfer of a count from the CR to the CE (refer to the "Functional Description")

MODE 0: INTERRUPT ON TERMINAL COUNT

Mode 0 is typically used for event counting. After the Control Word is written, OUT is initially low, and will remain low until the Counter reaches zero. OUT then goes high and remains high until a new count or a new Mode 0 Control Word is written into the Counter.

GATE = 1 enables counting; GATE = 0 disables counting. GATE has no effect on OUT.

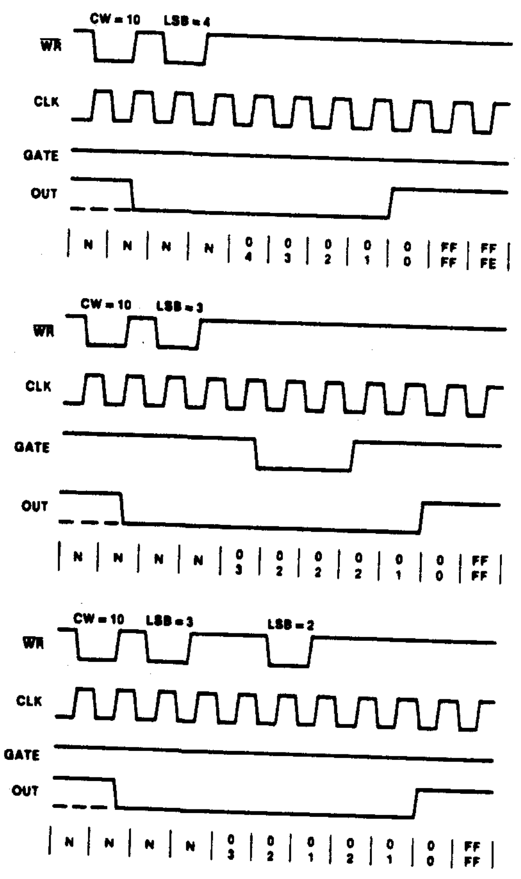
After the Control Word and initial count are written to a Counter, the initial count will be loaded on the next CLK pulse. This CLK pulse does not decrement the count, so for an initial count of N, OUT does not go high until N + 1 CLK pulses after the initial count is written.

If a new count is written to the Counter, it will be loaded on the next CLK pulse and counting will continue from the new count. If a two-byte count is written, the following happens:

1) Writing the first byte disables counting. OUT is set low immediately (no clock pulse required).
2) Writing the second byte allows the new count to be loaded on the next CLK pulse.

This allows the counting sequence to be synchronized by software. Again, OUT does not go high until N + 1 CLK pulses after the new count of N is written.

If an initial count is written while GATE = 0, it will still be loaded on the next CLK pulse. When GATE goes high, OUT will go high N CLK pulses later; no CLK pulse is needed to load the Counter as this has already been done.



231244-8

NOTE:
The Following Conventions Apply To All Mode Timing Diagrams:
1. Counters are programmed for binary (not BCD) counting and for Reading/Writing least significant byte (LSB) only.
2. The counter is always selected ($\overline{CS}$ always low).
3. CW stands for "Control Word"; CW = 10 means a control word of 10, hex is written to the counter.
4. LSB stands for "Least Significant Byte" of count.
5. Numbers below diagrams are count values.
The lower number is the least significant byte.
The upper number is the most significant byte. Since the counter is programmed to Read/Write LSB only, the most significant byte cannot be read.
N stands for an undefined count.
Vertical lines show transitions between count values.

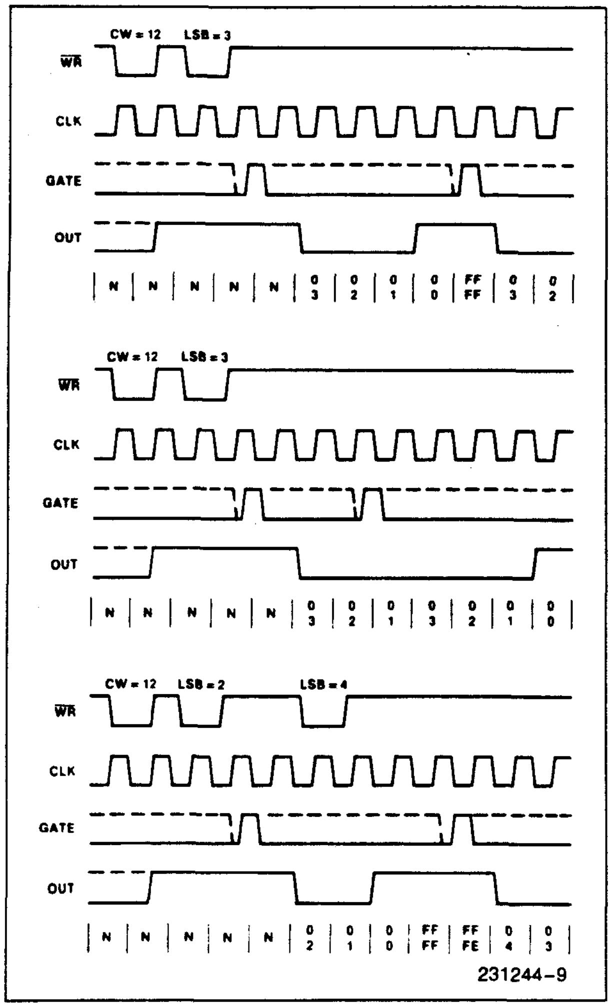
Figure 15. Mode 0

MODE 1: HARDWARE RETRIGGERABLE ONE-SHOT

OUT will be initially high. OUT will go low on the CLK pulse following a trigger to begin the one-shot pulse, and will remain low until the Counter reaches zero. OUT will then go high and remain high until the CLK pulse after the next trigger.

After writing the Control Word and initial count, the Counter is armed. A trigger results in loading the Counter and setting OUT low on the next CLK pulse, thus starting the one-shot pulse. An initial count of N will result in a one-shot pulse N CLK cycles in duration. The one-shot is retriggerable, hence OUT will remain low for N CLK pulses after any trigger. The one-shot pulse can be repeated without rewriting the same count into the counter. GATE has no effect on OUT.

If a new count is written to the Counter during a one-shot pulse, the current one-shot is not affected unless the Counter is retriggered. In that case, the Counter is loaded with the new count and the one-shot pulse continues until the new count expires.



CW=12 LSB=3
WR
CLK
GATE
OUT
N N N N N 0 3 0 2 0 1 0 0 FF FF 0 3 0 2
CW=12 LSB=3
WR
CLK
GATE
OUT
N N N N N 0 3 0 2 0 1 0 3 0 2 0 1 0 0
CW=12 LSB=2 LSB=4
WR
CLK
GATE
OUT
N N N N N 0 2 0 1 0 0 FF FF FF FE 0 4 0 3
231244-9

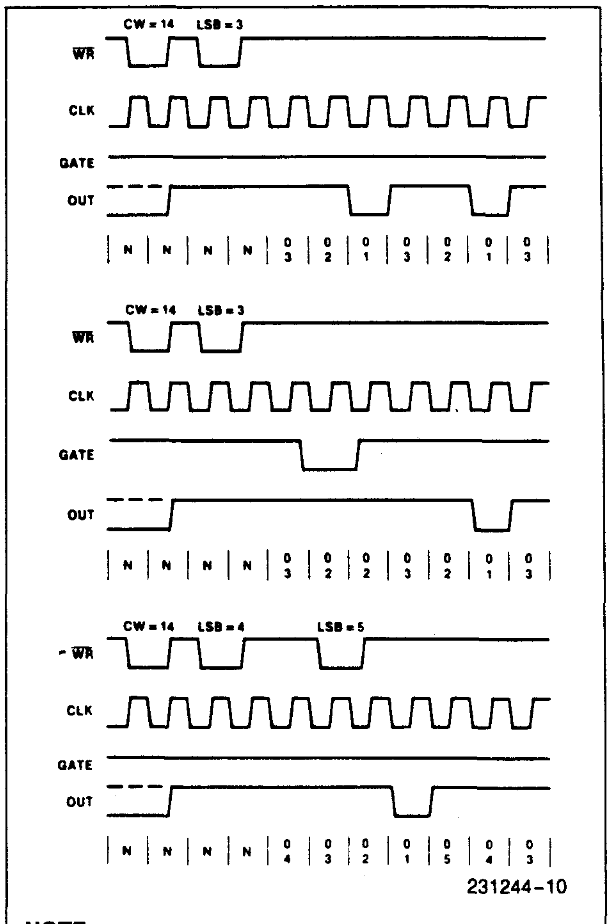
Figure 16. Mode 1

MODE 2: RATE GENERATOR

This Mode functions like a divide-by-N counter. It is typicially used to generate a Real Time Clock interrupt. OUT will initially be high. When the initial count has decremented to 1, OUT goes low for one CLK pulse. OUT then goes high again, the Counter reloads the initial count and the process is repeated. Mode 2 is periodic; the same sequence is repeated indefinitely. For an initial count of N, the sequence repeats every N CLK cycles.

GATE = 1 enables counting; GATE = 0 disables counting. If GATE goes low during an output pulse, OUT is set high immediately. A trigger reloads the Counter with the initial count on the next CLK pulse; OUT goes low N CLK pulses after the trigger. Thus the GATE input can be used to synchronize the Counter.

After writing a Control Word and initial count, the Counter will be loaded on the next CLK pulse. OUT goes low N CLK Pulses after the initial count is written. This allows the Counter to be synchronized by software also.



CW=14 LSB=3
WR
CLK
GATE
OUT
N N N N 0 3 0 2 0 1 0 3 0 2 0 1 0 3
CW=14 LSB=3
WR
CLK
GATE
OUT
N N N N 0 3 0 2 0 2 0 3 0 2 0 1 0 3
CW=14 LSB=4 LSB=5
WR
CLK
GATE
OUT
N N N N 0 4 0 3 0 2 0 1 0 5 0 4 0 3
231244-10

NOTE:
A GATE transition should not occur one clock prior to terminal count.

Figure 17. Mode 2

Writing a new count while counting does not affect the current counting sequence. If a trigger is received after writing a new count but before the end of the current period, the Counter will be loaded with the new count on the next CLK pulse and counting will continue from the new count. Otherwise, the new count will be loaded at the end of the current counting cycle. In mode 2, a COUNT of 1 is illegal.

MODE 3: SQUARE WAVE MODE

Mode 3 is typically used for Baud rate generation. Mode 3 is similar to Mode 2 except for the duty cycle of OUT. OUT will initially be high. When half the initial count has expired, OUT goes low for the remainder of the count. Mode 3 is periodic; the sequence above is repeated indefinitely. An initial count of N results in a square wave with a period of N CLK cycles.

GATE = 1 *enables counting;* GATE = 0 *disables* counting. If GATE goes low while OUT is low, OUT is set high immediately; no CLK pulse is required. A trigger reloads the Counter with the initial count on the next CLK pulse. Thus the GATE input can be used to synchronize the Counter.

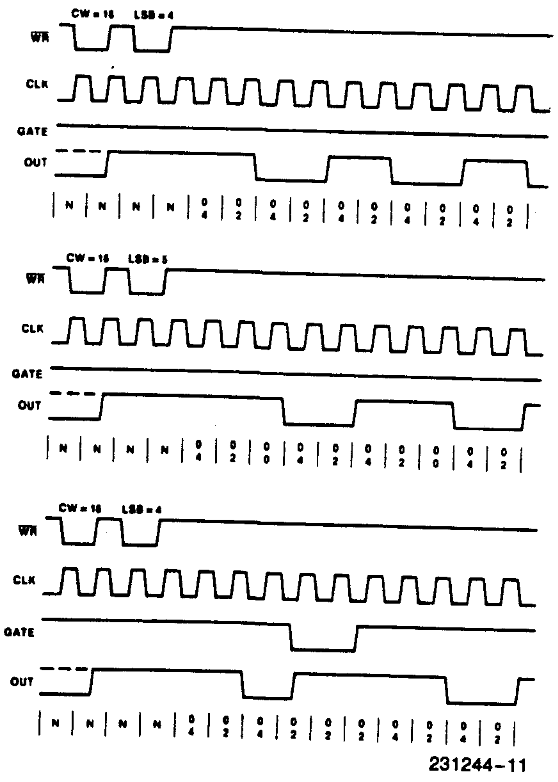
After writing a Control Word and initial count, the Counter will be loaded on the next CLK pulse. This allows the Counter to be synchronized by software also.

Writing a new count while counting does not affect the current counting sequence. If a trigger is received after writing a new count but before the end of the current half-cycle of the square wave, the Counter will be loaded with the new count on the next CLK pulse and counting will continue from the new count. Otherwise, the new count will be loaded at the end of the current half-cycle.

Mode 3 is implemented as follows:

Even counts: OUT is initially high. The initial count is loaded on one CLK pulse and then is decremented by two on succeeding CLK pulses. When the count expires OUT changes value and the Counter is reloaded with the initial count. The above process is repeated indefinitely.

Odd counts: OUT is initially high. The initial count minus one (an even number) is loaded on one CLK pulse and then is decremented by two on succeeding CLK pulses. One CLK pulse *after* the count expires, OUT goes low and the Counter is reloaded with the initial count minus one. Succeeding CLK pulses decrement the count by two. When the count expires, OUT goes high again and the Counter is reloaded with the initial count minus one. The above process is repeated indefinitely. So for odd counts, OUT will be high for (N + 1)/2 counts and low for (N − 1)/2 counts.



NOTE:
A GATE transition should not occur one clock prior to terminal count.

Figure 18. Mode 3

MODE 4: SOFTWARE TRIGGERED STROBE

OUT will be initially high. When the initial count expires, OUT will go low for one CLK pulse and then go high again. The counting sequence is "triggered" by writing the initial count.

GATE = 1 enables counting; GATE = 0 disables counting. GATE has no effect on OUT.

After writing a Control Word and initial count, the Counter will be loaded on the next CLK pulse. This CLK pulse does not decrement the count, so for an initial count of N, OUT does not strobe low until N + 1 CLK pulses after the initial count is written.

If a new count is written during counting, it will be loaded on the next CLK pulse and counting will continue from the new count. If a two-byte count is written, the following happens:

1) Writing the first byte has no effect on counting.
2) Writing the second byte allows the new count to be loaded on the next CLK pulse.

This allows the sequence to be "retriggered" by software. OUT strobes low N+1 CLK pulses after the new count of N is written.

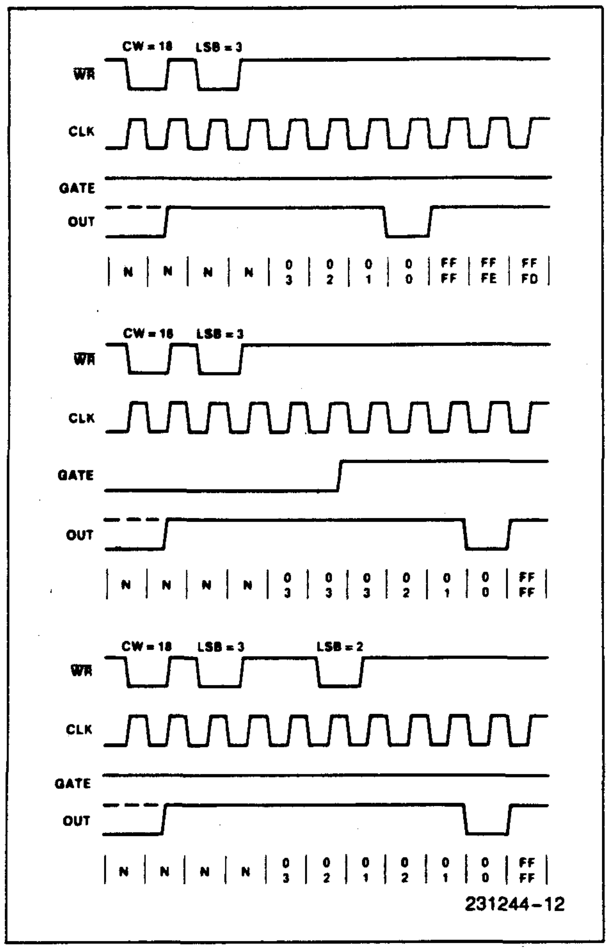


Figure 19. Mode 4

MODE 5: HARDWARE TRIGGERED STROBE (RETRIGGERABLE)

OUT will initially be high. Counting is triggered by a rising edge of GATE. When the initial count has expired, OUT will go low for one CLK pulse and then go high again.

After writing the Control Word and initial count, the counter will not be loaded until the CLK pulse after a trigger. This CLK pulse does not decrement the count, so for an initial count of N, OUT does not strobe low until N+1 CLK pulses after a trigger.

A trigger results in the Counter being loaded with the initial count on the next CLK pulse. The counting sequence is retriggerable. OUT will not strobe low for N + 1 CLK pulses after any trigger. GATE has no effect on OUT.

If a new count is written during counting, the current counting sequence will not be affected. If a trigger occurs after the new count is written but before the current count expires, the Counter will be loaded with the new count on the next CLK pulse and counting will continue from there.

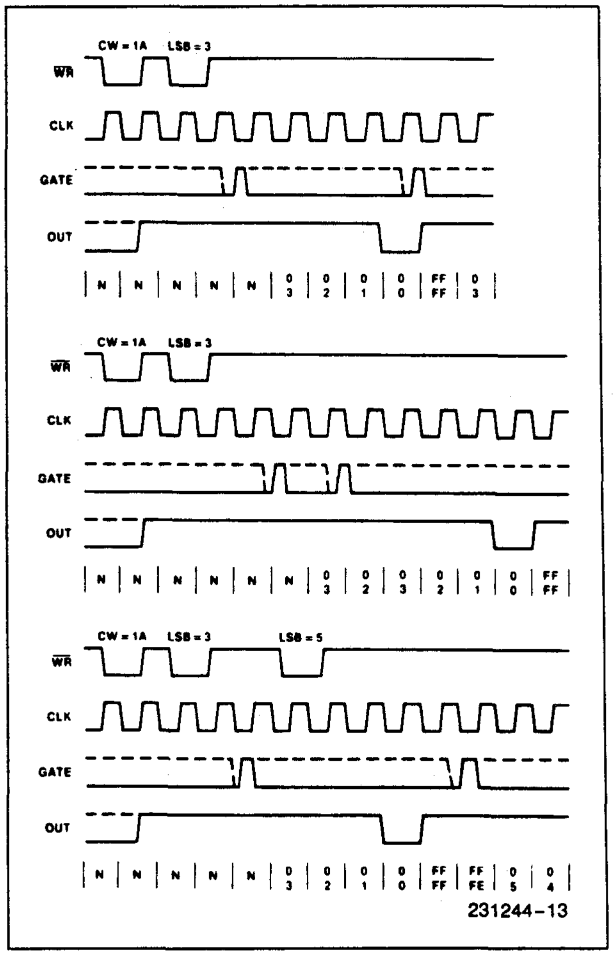


Figure 20. Mode 5

Signal Status Modes	Low Or Going Low	Rising	High
0	Disables counting	—	Enables counting
1	—	1) Initiates counting 2) Resets output after next clock	—
2	1) Disables counting 2) Sets output immediately high	Initiates counting	Enables counting
3	1) Disables counting 2) Sets output immediately high	Initiates counting	Enables counting
4	Disables counting	—	Enables counting
5	—	Initiates counting	—

Figure 21. Gate Pin Operations Summary

MODE	MIN COUNT	MAX COUNT
0	1	0
1	1	0
2	2	0
3	2	0
4	1	0

NOTE:
0 is equivalent to 2^{16} for binary counting and 10^4 for BCD counting

Figure 22. Minimum and Maximum Initial Counts

Operation Common to All Modes

Programming

When a Control Word is written to a Counter, all Control Logic is immediately reset and OUT goes to a known initial state; no CLK pulses are required for this.

GATE

The GATE input is always sampled on the rising edge of CLK. In Modes 0, 2, 3, and 4 the GATE input is level sensitive, and the logic level is sampled on the rising edge of CLK. In Modes 1, 2, 3, and 5 the GATE input is rising-edge sensitive. In these Modes, a rising edge of GATE (trigger) sets an edge-sensitive flip-flop in the Counter. This flip-flop is then sampled on the next rising edge of CLK; the flip-flop is reset immediately after it is sampled. In this way, a trigger will be detected no matter when it occurs—a high logic level does not have to be maintained until the next rising edge of CLK. Note that in Modes 2 and 3, the GATE input is both edge- and level-sensitive. In Modes 2 and 3, if a CLK source other than the system clock is used, GATE should be pulsed immediately following $\overline{WR}$ of a new count value.

COUNTER

New counts are loaded and Counters are decremented on the falling edge of CLK.

The largest possible initial count is 0; this is equivalent to 2^{16} for binary counting and 10^4 for BCD counting.

The Counter does not stop when it reaches zero. In Modes 0, 1, 4, and 5 the Counter "wraps around" to the highest count, either FFFF hex for binary counting or 9999 for BCD counting, and continues counting. Modes 2 and 3 are periodic; the Counter reloads itself with the initial count and continues counting from there.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias	0°C to 70°C
Storage Temperature	−65° to +150°C
Supply Voltage	−0.5 to +8.0V
Operating Voltage	+4V to +7V
Voltage on any Input	GND −2V to +6.5V
Voltage on any Output	GND−0.5V to V_{CC} + 0.5V
Power Dissipation	1 Watt

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS

(T_A=0°C to 70°C, V_{CC}=5V± 10%, GND=0V) (T_A = −40°C to +85°C for Extended Temperature)

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage	−0.5	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC} + 0.5	V	
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 2.5 mA
V_{OH}	Output High Voltage	3.0 V_{CC} − 0.4		V V	I_{OH} = −2.5 mA I_{OH} = −100 μA
I_{IL}	Input Load Current		±2.0	μA	V_{IN}=V_{CC} to 0V
I_{OFL}	Output Float Leakage Current		±10	μA	V_{OUT}=V_{CC} to 0.0V
I_{CC}	V_{CC} Supply Current		20	mA	Clk Freq= 8MHz 82C54 10MHz 82C54-2
I_{CCSB}	V_{CC} Supply Current-Standby		10	μA	CLK Freq = DC $\overline{CS}$ = V_{CC}. All Inputs/Data Bus V_{CC} All Outputs Floating
I_{CCSB1}	V_{CC} Supply Current-Standby		150	μA	CLK Freq = DC $\overline{CS}$ = V_{CC}. All Other Inputs, I/O Pins = V_{GND}, Outputs Open
C_{IN}	Input Capacitance		10	pF	f_c = 1 MHz
$C_{I/O}$	I/O Capacitance		20	pF	Unmeasured pins returned to GND(5)
C_{OUT}	Output Capacitance		20	pF	

A.C. CHARACTERISTICS

(T_A = 0°C to 70°C, V_{CC} = 5V ±10%, GND =0V) (T_A = −40°C to +85°C for Extended Temperature)

BUS PARAMETERS (Note 1)

READ CYCLE

Symbol	Parameter	82C54		82C54-2		Units
		Min	Max	Min	Max	
t_{AR}	Address Stable Before $\overline{RD}$ ↓	45		30		ns
t_{SR}	$\overline{CS}$ Stable Before $\overline{RD}$ ↓	0		0		ns
t_{RA}	Address Hold Time After $\overline{RD}$ ↑	0		0		ns
t_{RR}	$\overline{RD}$ Pulse Width	150		95		ns
t_{RD}	Data Delay from $\overline{RD}$ ↓		120		85	ns
t_{AD}	Data Delay from Address		220		185	ns
t_{DF}	$\overline{RD}$ ↑ to Data Floating	5	90	5	65	ns
t_{RV}	Command Recovery Time	200		165		ns

NOTE:
1. AC timings measured at V_{OH} = 2.0V, V_{OL} = 0.8V.

A.C. CHARACTERISTICS (Continued)

WRITE CYCLE

Symbol	Parameter	82C54		82C54-2		Units
		Min	Max	Min	Max	
t_{AW}	Address Stable Before $\overline{WR}$ ↓	0		0		ns
t_{SW}	$\overline{CS}$ Stable Before $\overline{WR}$ ↓	0		0		ns
t_{WA}	Address Hold Time After $\overline{WR}$ ↑	0		0		ns
t_{WW}	$\overline{WR}$ Pulse Width	150		95		ns
t_{DW}	Data Setup Time Before $\overline{WR}$ ↑	120		95		ns
t_{WD}	Data Hold Time After $\overline{WR}$ ↑	0		0		ns
t_{RV}	Command Recovery Time	200		165		ns

CLOCK AND GATE

Symbol	Parameter	82C54		82C54-2		Units
		Min	Max	Min	Max	
t_{CLK}	Clock Period	125	DC	100	DC	ns
t_{PWH}	High Pulse Width	60(3)		30(3)		ns
t_{PWL}	Low Pulse Width	60(3)		50(3)		ns
T_R	Clock Rise Time		25		25	ns
t_F	Clock Fall Time		25		25	ns
t_{GW}	Gate Width High	50		50		ns
t_{GL}	Gate Width Low	50		50		ns
t_{GS}	Gate Setup Time to CLK ↑	50		40		ns
t_{GH}	Gate Hold Time After CLK ↑	50(2)		50(2)		ns
T_{OD}	Output Delay from CLK ↓		150		100	ns
t_{ODG}	Output Delay from Gate ↓		120		100	ns
t_{WC}	CLK Delay for Loading(4)	0	55	0	55	ns
t_{WG}	Gate Delay for Sampling(4)	−5	50	−5	40	ns
t_{WO}	OUT Delay from Mode Write		260		240	ns
t_{CL}	CLK Set Up for Count Latch	−40	45	−40	40	ns

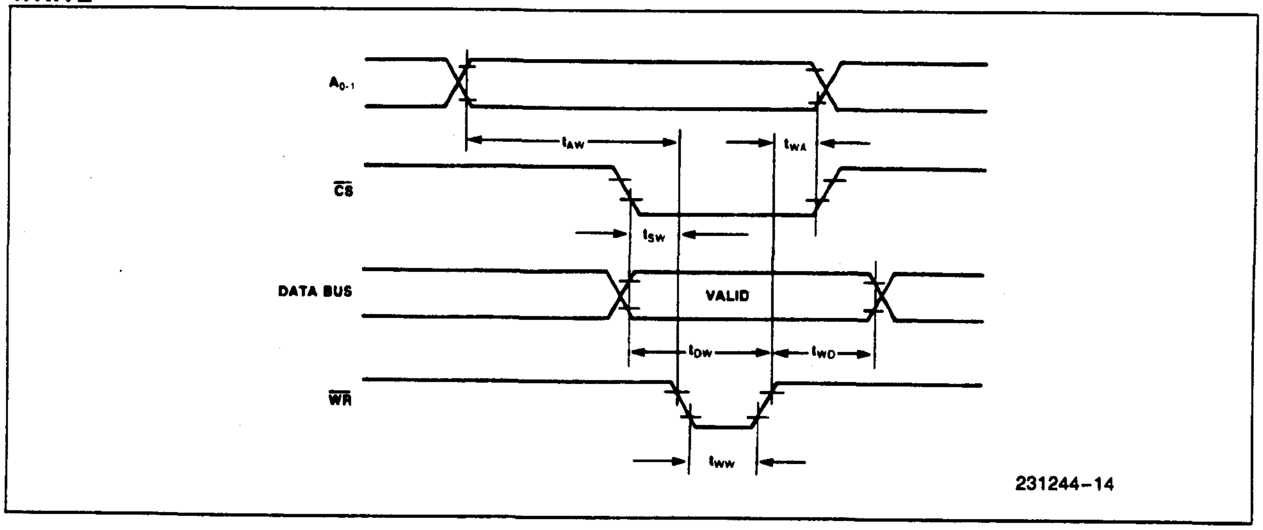
NOTES:
2. In Modes 1 and 5 triggers are sampled on each rising clock edge. A second trigger within 120 ns (70 ns for the 82C54-2) of the rising clock edge may not be detected.
3. Low-going glitches that violate t_{PWH}, t_{PWL} may cause errors requiring counter reprogramming.
4. Except for Extended Temp., See Extended Temp. A.C. Characteristics below.
5. Sampled not 100% tested. T_A = 25°C.
6. If CLK present at T_{WC} min then Count equals N+2 CLK pulses, T_{WC} max equals Count N+1 CLK pulse. T_{WC} min to T_{WC} max, count will be either N+1 or N+2 CLK pulses.
7. In Modes 1 and 5, if GATE is present when writing a new Count value, at T_{WG} min Counter will not be triggered, at T_{WG} max Counter will be triggered.
8. If CLK present when writing a Counter Latch or ReadBack Command, at T_{CL} min CLK will be reflected in count value latched, at T_{CL} max CLK will not be reflected in the count value latched. Writing a Counter Latch or ReadBack Command between T_{CL} min and T_{WL} max will result in a latched count vallue which is ± one least significant bit.

EXTENDED TEMPERATURE (T_A = −40°C to +85°C for Extended Temperature)

Symbol	Parameter	82C54		82C54-2		Units
		Min	Max	Min	Max	
t_{WC}	CLK Delay for Loading	−25	25	−25	25	ns
t_{WG}	Gate Delay for Sampling	−25	25	−25	25	ns

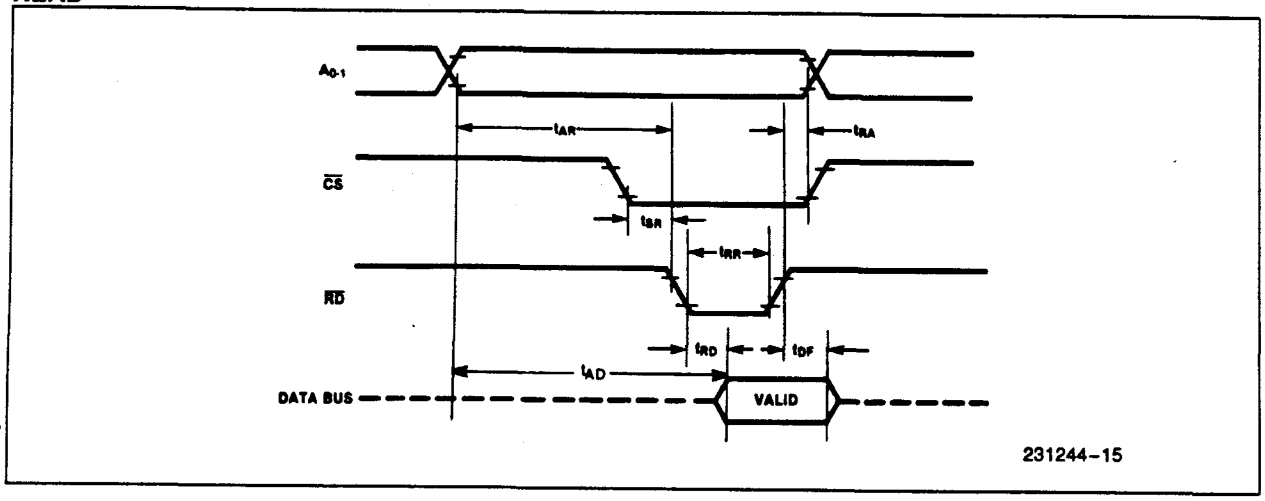
WAVEFORMS

WRITE



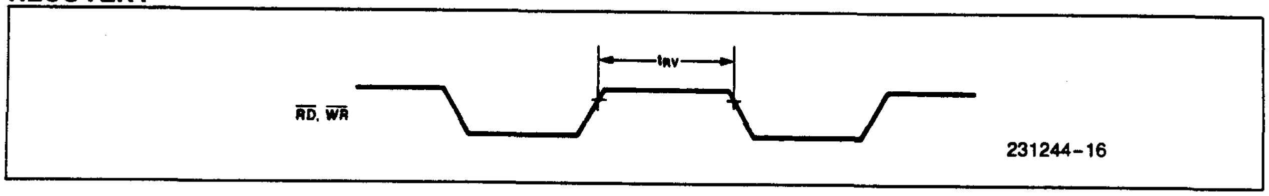
231244-14

READ



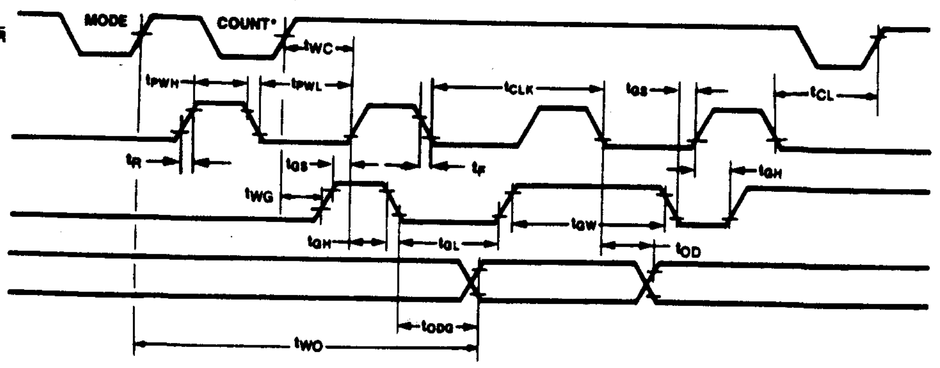
231244-15

RECOVERY



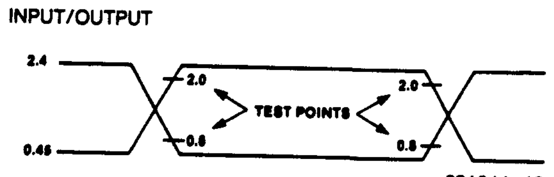
231244-16

CLOCK AND GATE



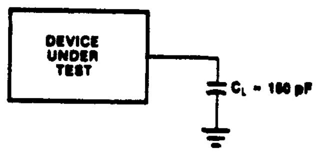
231244–17
* Last byte of count being written

A.C. TESTING INPUT, OUTPUT WAVEFORM



231244–18
A.C. Testing: Inputs are driven at 2.4V for a logic "1" and 0.45V for a logic "0." Timing measurements are made at 2.0V for a logic "1" and 0.8V for a logic "0."

A.C. TESTING LOAD CIRCUIT



231244–19
C_L = 150 pF
C_L includes jig capacitance

Intel 82C55A Programmable Peripheral Interface
Data Sheet Reprint

intel®

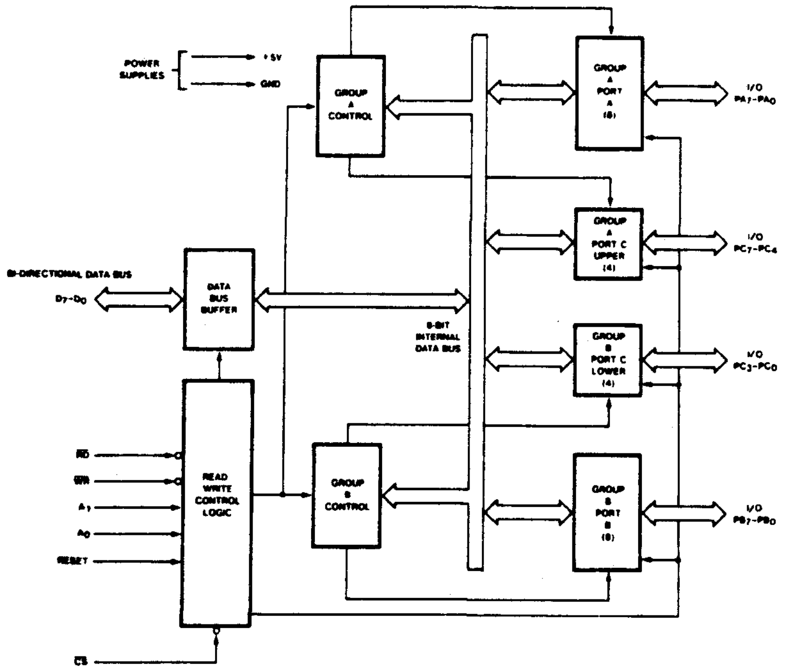
82C55A
CHMOS PROGRAMMABLE PERIPHERAL INTERFACE

- Compatible with all Intel and Most Other Microprocessors
- High Speed, "Zero Wait State" Operation with 8 MHz 8086/88 and 80186/188
- 24 Programmable I/O Pins
- Low Power CHMOS
- Completely TTL Compatible
- Control Word Read-Back Capability
- Direct Bit Set/Reset Capability
- 2.5 mA DC Drive Capability on all I/O Port Outputs
- Available in 40-Pin DIP and 44-Pin PLCC
- Available in EXPRESS
  - —Standard Temperature Range
  - —Extended Temperature Range

The Intel 82C55A is a high-performance, CHMOS version of the industry standard 8255A general purpose programmable I/O device which is designed for use with all Intel and most other microprocessors. It provides 24 I/O pins which may be individually programmed in 2 groups of 12 and used in 3 major modes of operation. The 82C55A is pin compatible with the NMOS 8255A and 8255A-5.

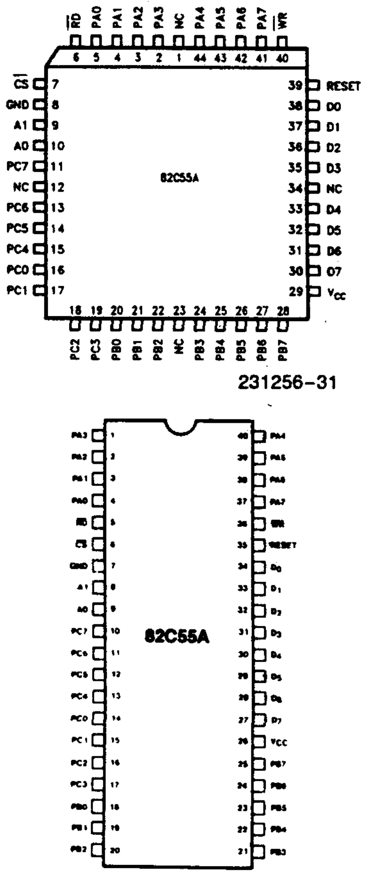
In MODE 0, each group of 12 I/O pins may be programmed in sets of 4 and 8 to be inputs or outputs. In MODE 1, each group may be programmed to have 8 lines of input or output. 3 of the remaining 4 pins are used for handshaking and interrupt control signals. MODE 2 is a strobed bi-directional bus configuration.

The 82C55A is fabricated on Intel's advanced CHMOS III technology which provides low power consumption with performance equal to or greater than the equivalent NMOS product. The 82C55A is available in 40-pin DIP and 44-pin plastic leaded chip carrier (PLCC) packages.



231256-1

Figure 1. 82C55A Block Diagram



231256-2

Figure 2. 82C55A Pinout

Diagrams are for pin reference only. Package sizes are not to scale.

September 1987
Order Number: 231256-004

Table 1. Pin Description

Symbol	Pin Number Dip	Pin Number PLCC	Type	Name and Function
PA_{3-0}	1–4	2–5	I/O	**PORT A, PINS 0–3:** Lower nibble of an 8-bit data output latch/ buffer and an 8-bit data input latch.
$\overline{RD}$	5	6	I	**READ CONTROL:** This input is low during CPU read operations.
$\overline{CS}$	6	7	I	**CHIP SELECT:** A low on this input enables the 82C55A to respond to $\overline{RD}$ and $\overline{WR}$ signals. $\overline{RD}$ and WR are ignored otherwise.
GND	7	8		**System Ground**
A_{1-0}	8–9	9–10	I	**ADDRESS:** These input signals, in conjunction $\overline{RD}$ and $\overline{WR}$, control the selection of one of the three ports or the control word registers.

A_1	A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	**Input Operation (Read)**
0	0	0	1	0	Port A - Data Bus
0	1	0	1	0	Port B - Data Bus
1	0	0	1	0	Port C - Data Bus
1	1	0	1	0	Control Word - Data Bus
					Output Operation (Write)
0	0	1	0	0	Data Bus - Port A
0	1	1	0	0	Data Bus - Port B
1	0	1	0	0	Data Bus - Port C
1	1	1	0	0	Data Bus - Control
					Disable Function
X	X	X	X	1	Data Bus - 3 - State
X	X	1	1	0	Data Bus - 3 - State

Symbol	Pin Number Dip	Pin Number PLCC	Type	Name and Function
PC_{7-4}	10–13	11,13–15	I/O	**PORT C, PINS 4–7:** Upper nibble of an 8-bit data output latch/ buffer and an 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the control signal outputs and status signal inputs in conjunction with ports A and B.
PC_{0-3}	14–17	16–19	I/O	**PORT C, PINS 0–3:** Lower nibble of Port C.
PB_{0-7}	18–25	20–22, 24–28	I/O	**PORT B, PINS 0–7:** An 8-bit data output latch/buffer and an 8-bit data input buffer.
V_{CC}	26	29		**SYSTEM POWER:** + 5V Power Supply.
D_{7-0}	27–34	30–33, 35–38	I/O	**DATA BUS:** Bi-directional, tri-state data bus lines, connected to system data bus.
RESET	35	39	I	**RESET:** A high on this input clears the control register and all ports are set to the input mode.
$\overline{WR}$	36	40	I	**WRITE CONTROL:** This input is low during CPU write operations.
PA_{7-4}	37–40	41–44	I/O	**PORT A, PINS 4–7:** Upper nibble of an 8-bit data output latch/ buffer and an 8-bit data input latch.
NC		1, 12, 23, 34		No Connect

82C55A FUNCTIONAL DESCRIPTION

General

The 82C55A is a programmable peripheral interface device designed for use in Intel microcomputer systems. Its function is that of a general purpose I/O component to interface peripheral equipment to the microcomputer system bus. The functional configuration of the 82C55A is programmed by the system software so that normally no external logic is necessary to interface peripheral devices or structures.

Data Bus Buffer

This 3-state bidirectional 8-bit buffer is used to interface the 82C55A to the system data bus. Data is transmitted or received by the buffer upon execution of input or output instructions by the CPU. Control words and status information are also transferred through the data bus buffer.

Read/Write and Control Logic

The function of this block is to manage all of the internal and external transfers of both Data and Control or Status words. It accepts inputs from the CPU Address and Control busses and in turn, issues commands to both of the Control Groups.

Group A and Group B Controls

The functional configuration of each port is programmed by the systems software. In essence, the CPU "outputs" a control word to the 82C55A. The control word contains information such as "mode", "bit set", "bit reset", etc., that initializes the functional configuration of the 82C55A.

Each of the Control blocks (Group A and Group B) accepts "commands" from the Read/Write Control Logic, receives "control words" from the internal data bus and issues the proper commands to its associated ports.

Control Group A - Port A and Port C upper (C7–C4)
Control Group B - Port B and Port C lower (C3–C0)

The control word register can be both written and read as shown in the address decode table in the pin descriptions. Figure 6 shows the control word format for both Read and Write operations. When the control word is read, bit D7 will always be a logic "1", as this implies control word mode information.

Ports A, B, and C

The 82C55A contains three 8-bit ports (A, B, and C). All can be configured in a wide variety of functional characteristics by the system software but each has its own special features or "personality" to further enhance the power and flexibility of the 82C55A.

Port A. One 8-bit data output latch/buffer and one 8-bit input latch buffer. Both "pull-up" and "pull-down" bus hold devices are present on Port A.

Port B. One 8-bit data input/output latch/buffer. Only "pull-up" bus hold devices are present on Port B.

Port C. One 8-bit data output latch/buffer and one 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the control signal outputs and status signal inputs in conjunction with ports A and B. Only "pull-up" bus hold devices are present on Port C.

See Figure 4 for the bus-hold circuit configuration for Port A, B, and C.

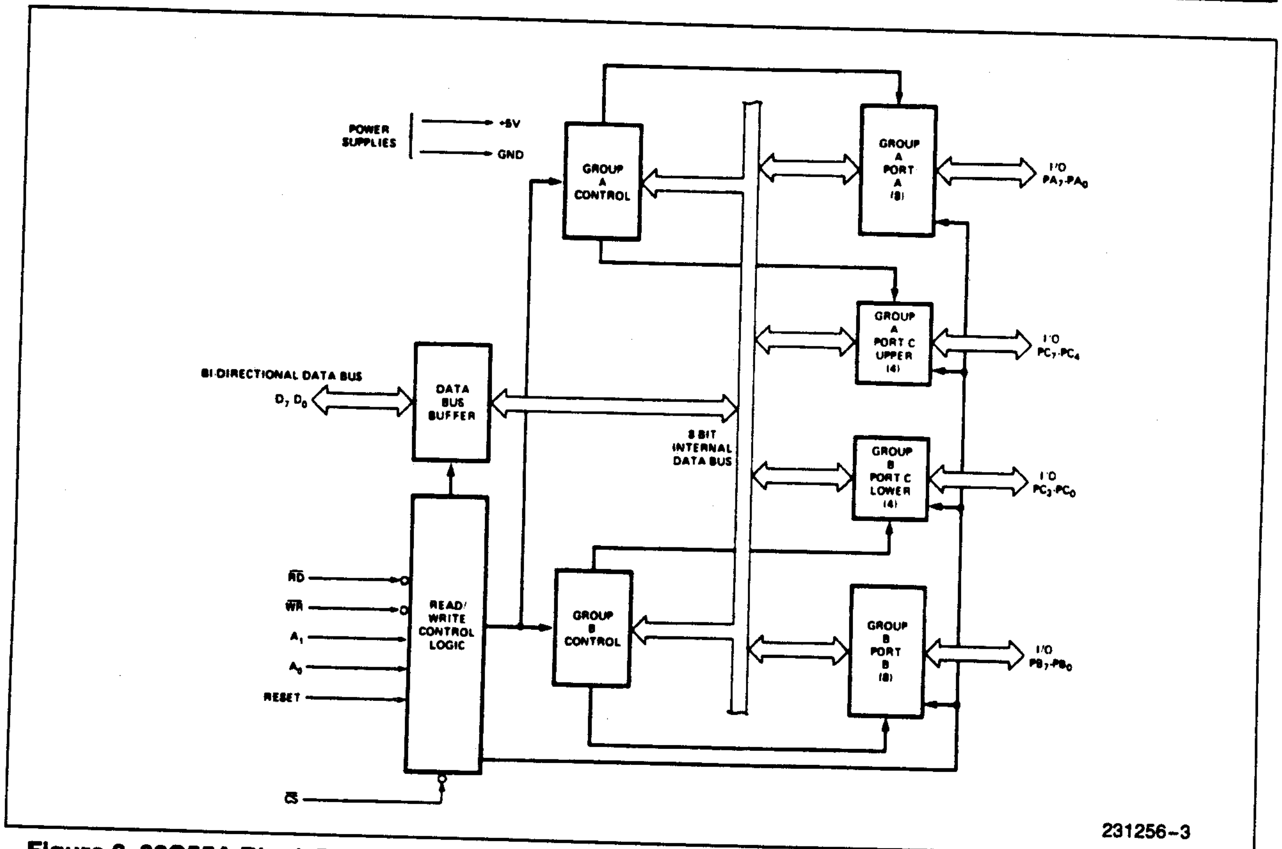
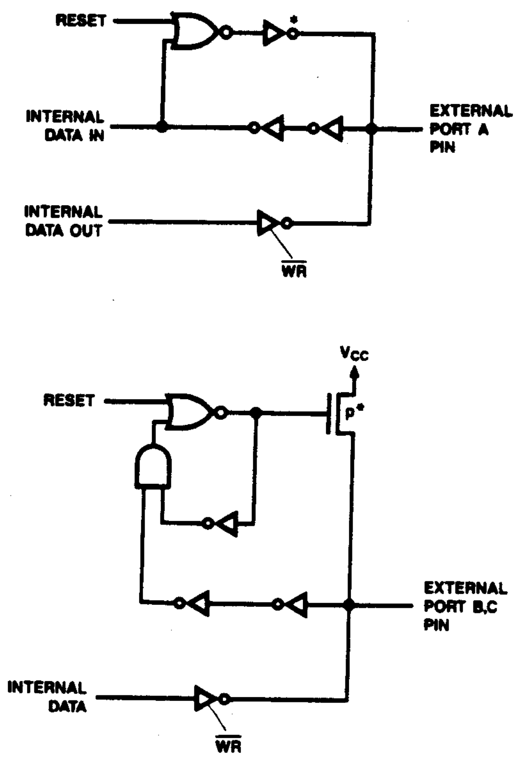


Figure 3. 82C55A Block Diagram Showing Data Bus Buffer and Read/Write Control Logic Functions



231256-4

***NOTE:**
Port pins loaded with more than 20 pF capacitance may not have their logic level guaranteed following a hardware reset.

Figure 4. Port A, B, C, Bus-hold Configuration

82C55A OPERATIONAL DESCRIPTION

Mode Selection

There are three basic modes of operation that can be selected by the system software:

Mode 0 — Basic input/output
Mode 1 — Strobed Input/output
Mode 2 — Bi-directional Bus

When the reset input goes "high" all ports will be set to the input mode with all 24 port lines held at a logic "one" level by the internal bus hold devices (see Figure 4 Note). After the reset is removed the 82C55A can remain in the input mode with no additional initialization required. This eliminates the need for pullup or pulldown devices in "all CMOS" designs. During the execution of the system program, any of the other modes may be selected by using a single output instruction. This allows a single 82C55A to service a variety of peripheral devices with a simple software maintenance routine.

The modes for Port A and Port B can be separately defined, while Port C is divided into two portions as required by the Port A and Port B definitions. All of the output registers, including the status flip-flops, will be reset whenever the mode is changed. Modes may be combined so that their functional definition can be "tailored" to almost any I/O structure. For instance; Group B can be programmed in Mode 0 to monitor simple switch closings or display computational results, Group A could be programmed in Mode 1 to monitor a keyboard or tape reader on an interrupt-driven basis.

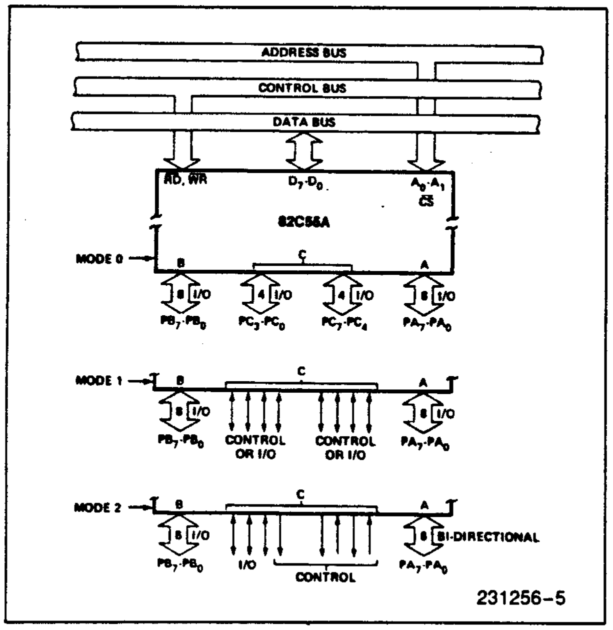


Figure 5. Basic Mode Definitions and Bus Interface

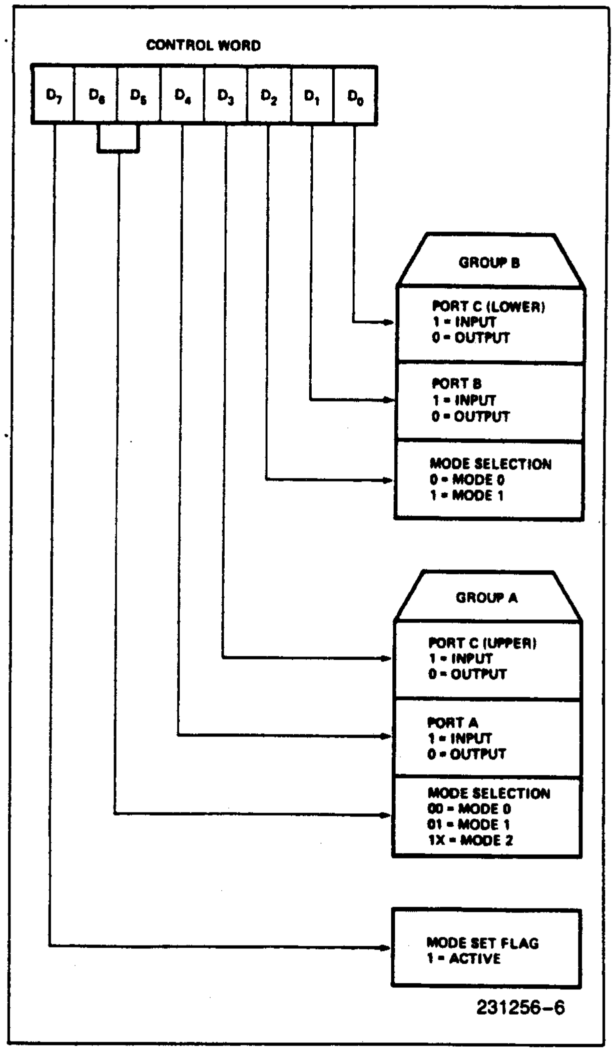


Figure 6. Mode Definition Format

The mode definitions and possible mode combinations may seem confusing at first but after a cursory review of the complete device operation a simple, logical I/O approach will surface. The design of the 82C55A has taken into account things such as efficient PC board layout, control signal definition vs PC layout and complete functional flexibility to support almost any peripheral device with no external logic. Such design represents the maximum use of the available pins.

Single Bit Set/Reset Feature

Any of the eight bits of Port C can be Set or Reset using a single OUTput instruction. This feature reduces software requirements in Control-based applications.

When Port C is being used as status/control for Port A or B, these bits can be set or reset by using the Bit Set/Reset operation just as if they were data output ports.

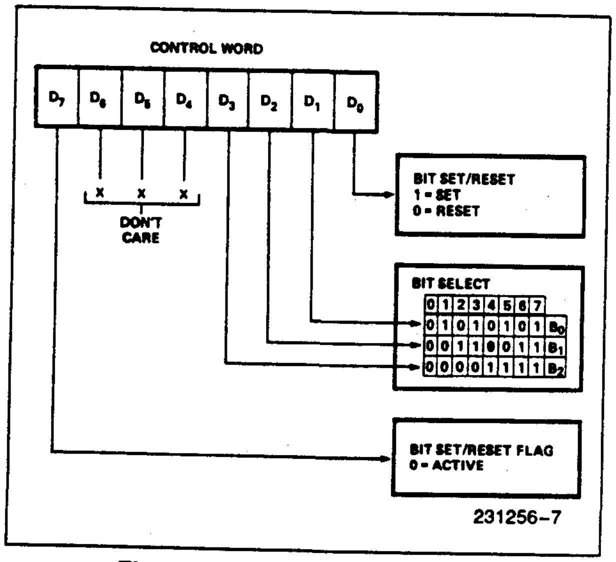


Figure 7. Bit Set/Reset Format

Interrupt Control Functions

When the 82C55A is programmed to operate in mode 1 or mode 2, control signals are provided that can be used as interrupt request inputs to the CPU. The interrupt request signals, generated from port C, can be inhibited or enabled by setting or resetting the associated INTE flip-flop, using the bit set/reset function of port C.

This function allows the Programmer to disallow or allow a specific I/O device to interrupt the CPU without affecting any other device in the interrupt structure.

INTE flip-flop definition:

(BIT-SET)—INTE is SET—Interrupt enable
(BIT-RESET)—INTE is RESET—Interrupt disable

Note:
All Mask flip-flops are automatically reset during mode selection and device Reset.

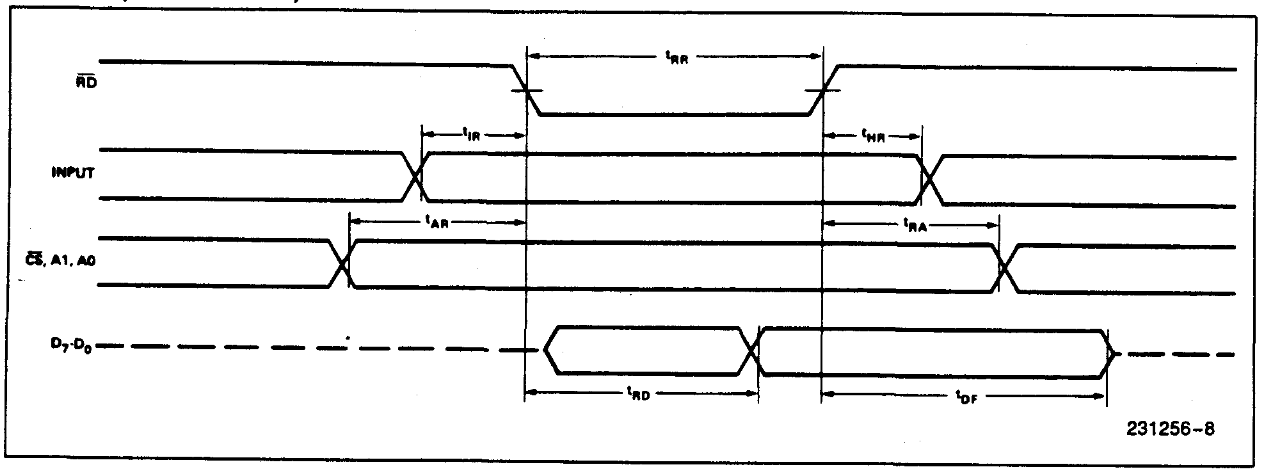
Operating Modes

Mode 0 (Basic Input/Output). This functional configuration provides simple input and output operations for each of the three ports. No "handshaking" is required, data is simply written to or read from a specified port.

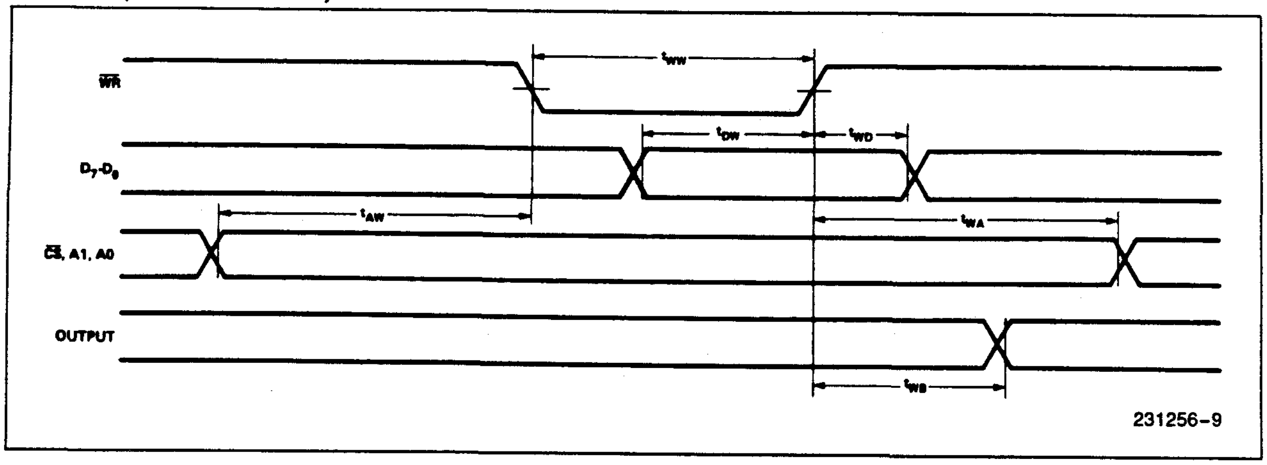
Mode 0 Basic Functional Definitions:

- Two 8-bit ports and two 4-bit ports.
- Any port can be input or output.
- Outputs are latched.
- Inputs are not latched.
- 16 different Input/Output configurations are possible in this Mode.

MODE 0 (BASIC INPUT)



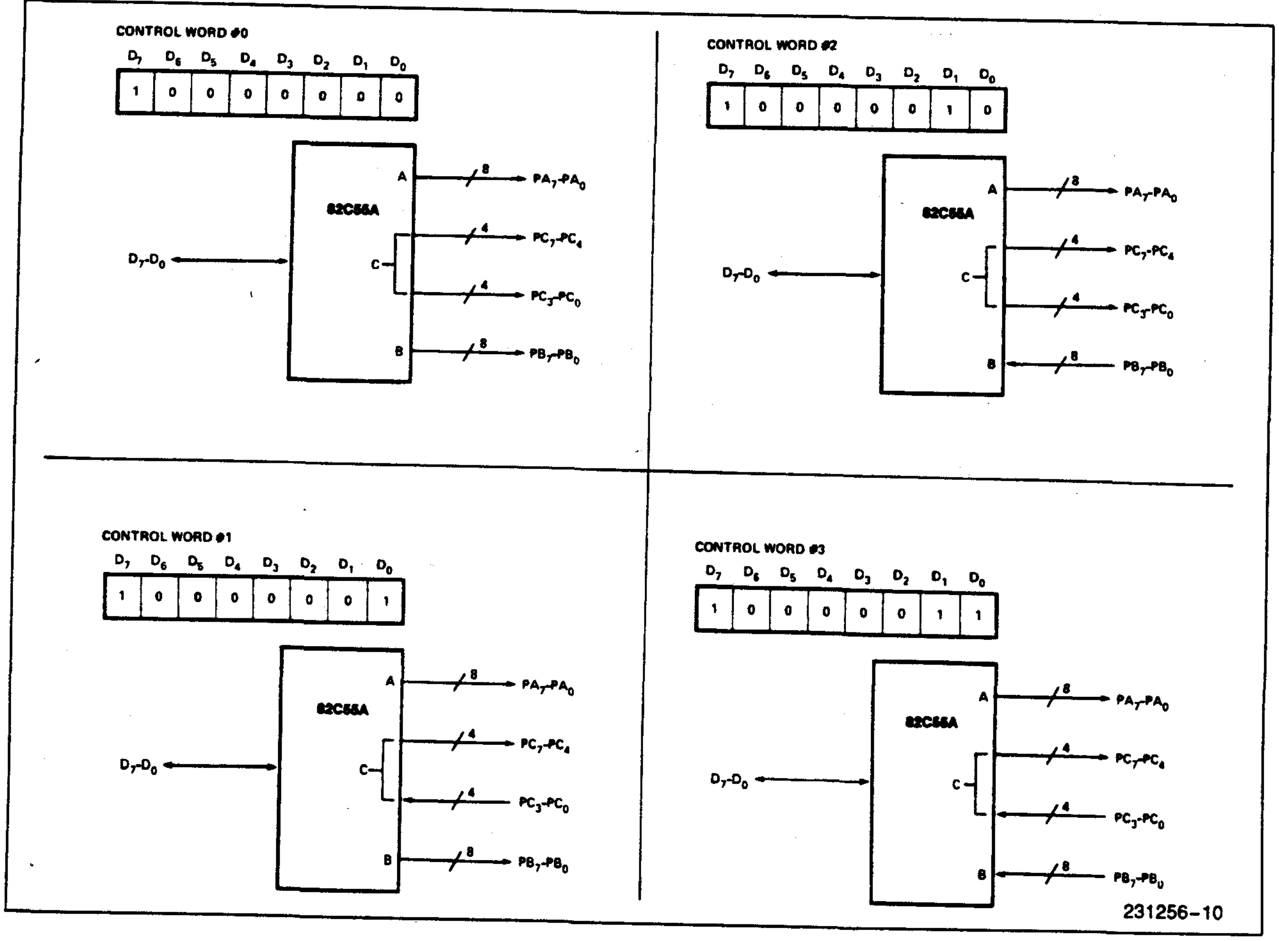
MODE 0 (BASIC OUTPUT)



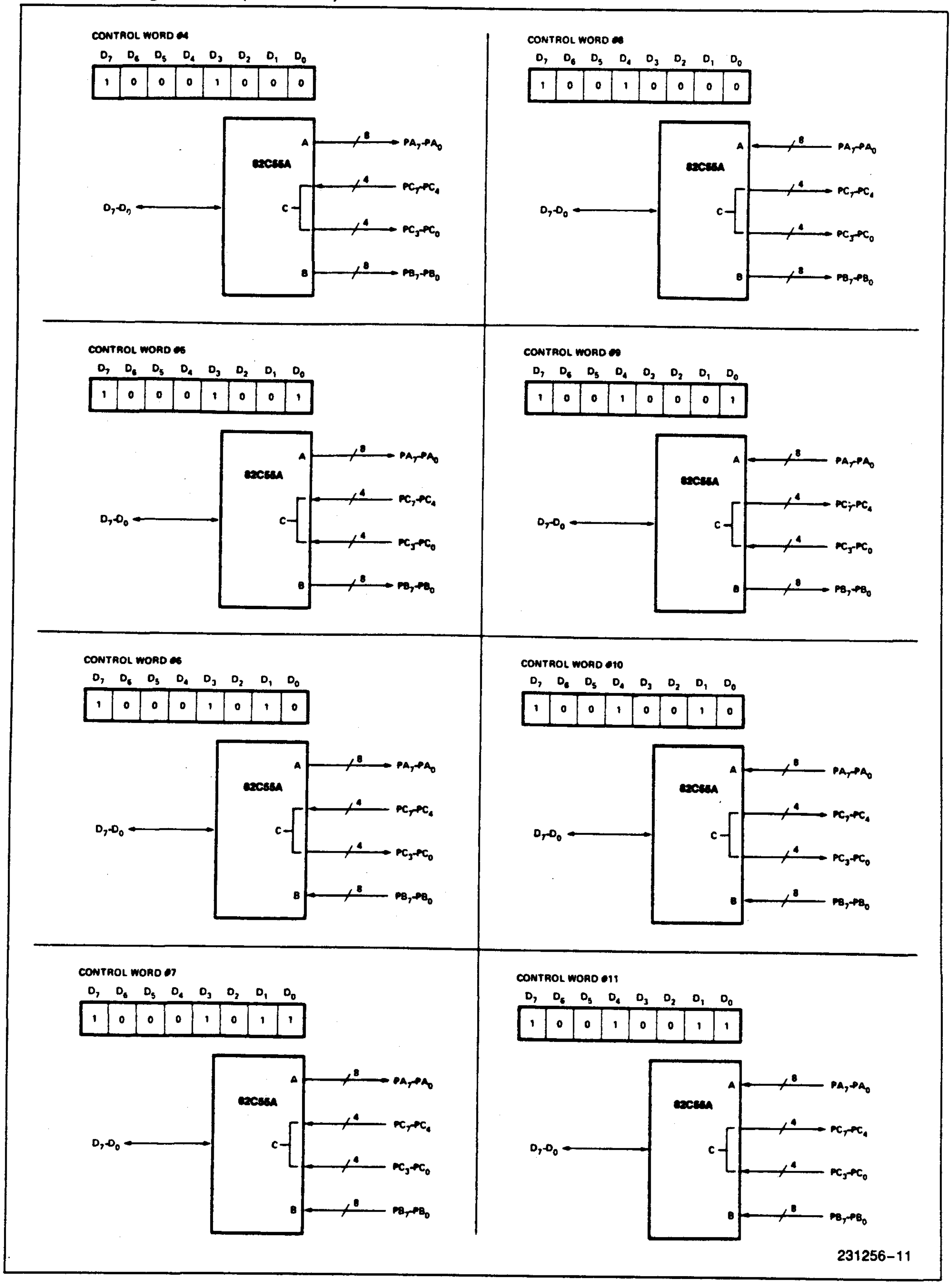
MODE 0 Port Definition

A		B		GROUP A			GROUP B	
D_4	D_3	D_1	D_0	PORT A	PORT C (UPPER)	#	PORT B	PORT C (LOWER)
0	0	0	0	OUTPUT	OUTPUT	0	OUTPUT	OUTPUT
0	0	0	1	OUTPUT	OUTPUT	1	OUTPUT	INPUT
0	0	1	0	OUTPUT	OUTPUT	2	INPUT	OUTPUT
0	0	1	1	OUTPUT	OUTPUT	3	INPUT	INPUT
0	1	0	0	OUTPUT	INPUT	4	OUTPUT	OUTPUT
0	1	0	1	OUTPUT	INPUT	5	OUTPUT	INPUT
0	1	1	0	OUTPUT	INPUT	6	INPUT	OUTPUT
0	1	1	1	OUTPUT	INPUT	7	INPUT	INPUT
1	0	0	0	INPUT	OUTPUT	8	OUTPUT	OUTPUT
1	0	0	1	INPUT	OUTPUT	9	OUTPUT	INPUT
1	0	1	0	INPUT	OUTPUT	10	INPUT	OUTPUT
1	0	1	1	INPUT	OUTPUT	11	INPUT	INPUT
1	1	0	0	INPUT	INPUT	12	OUTPUT	OUTPUT
1	1	0	1	INPUT	INPUT	13	OUTPUT	INPUT
1	1	1	0	INPUT	INPUT	14	INPUT	OUTPUT
1	1	1	1	INPUT	INPUT	15	INPUT	INPUT

MODE 0 Configurations

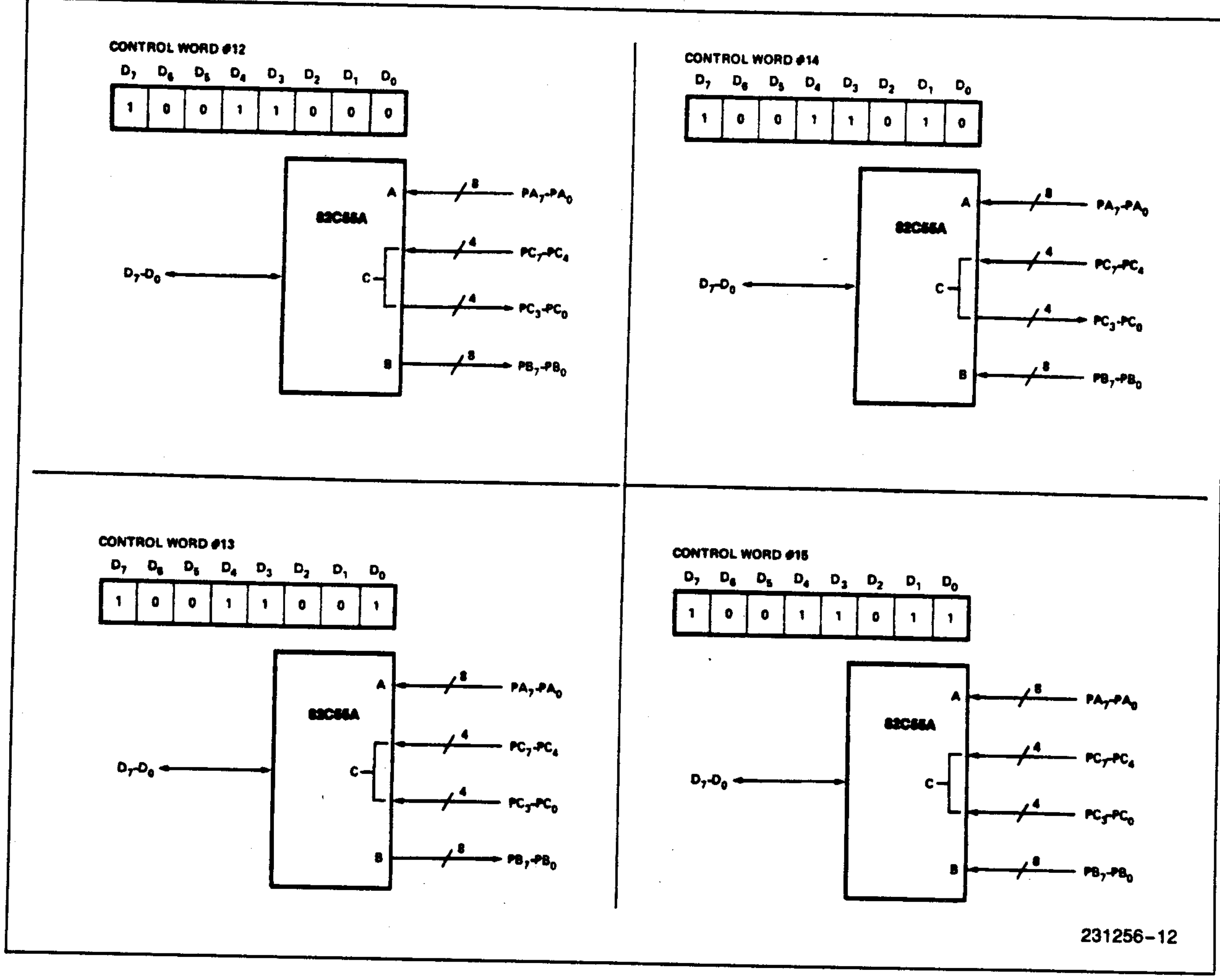


MODE 0 Configurations (Continued)



231256-11

MODE 0 Configurations (Continued)



231256-12

Operating Modes

MODE 1 (Strobed Input/Output). This functional configuration provides a means for transferring I/O data to or from a specified port in conjunction with strobes or "handshaking" signals. In mode 1, Port A and Port B use the lines on Port C to generate or accept these "handshaking" signals.

Mode 1 Basic functional Definitions:

- Two Groups (Group A and Group B).
- Each group contains one 8-bit data port and one 4-bit control/data port.
- The 8-bit data port can be either input or output Both inputs and outputs are latched.
- The 4-bit port is used for control and status of the 8-bit data port.

Input Control Signal Definition

$\overline{STB}$ (Strobe Input). A "low" on this input loads data into the input latch.

IBF (Input Buffer Full F/F)

A "high" on this output indicates that the data has been loaded into the input latch; in essence, an acknowledgement. IBF is set by $\overline{STB}$ input being low and is reset by the rising edge of the $\overline{RD}$ input.

INTR (Interrupt Request)

A "high" on this output can be used to interrupt the CPU when an input device is requesting service. INTR is set by the $\overline{STB}$ is a "one", IBF is a "one" and INTE is a "one". It is reset by the falling edge of $\overline{RD}$. This procedure allows an input device to request service from the CPU by simply strobing its data into the port.

INTE A

Controlled by bit set/reset of PC_4.

INTE B

Controlled by bit set/reset of PC_2.

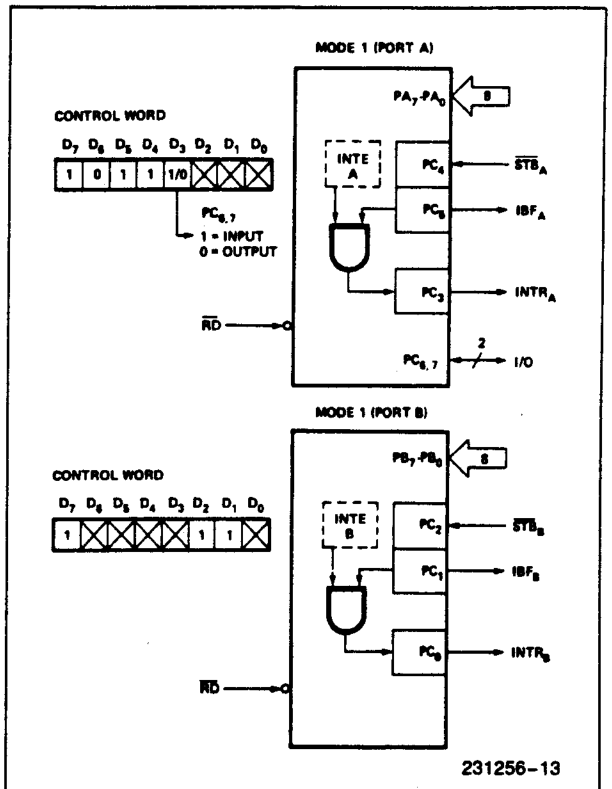


Figure 8. MODE 1 Input

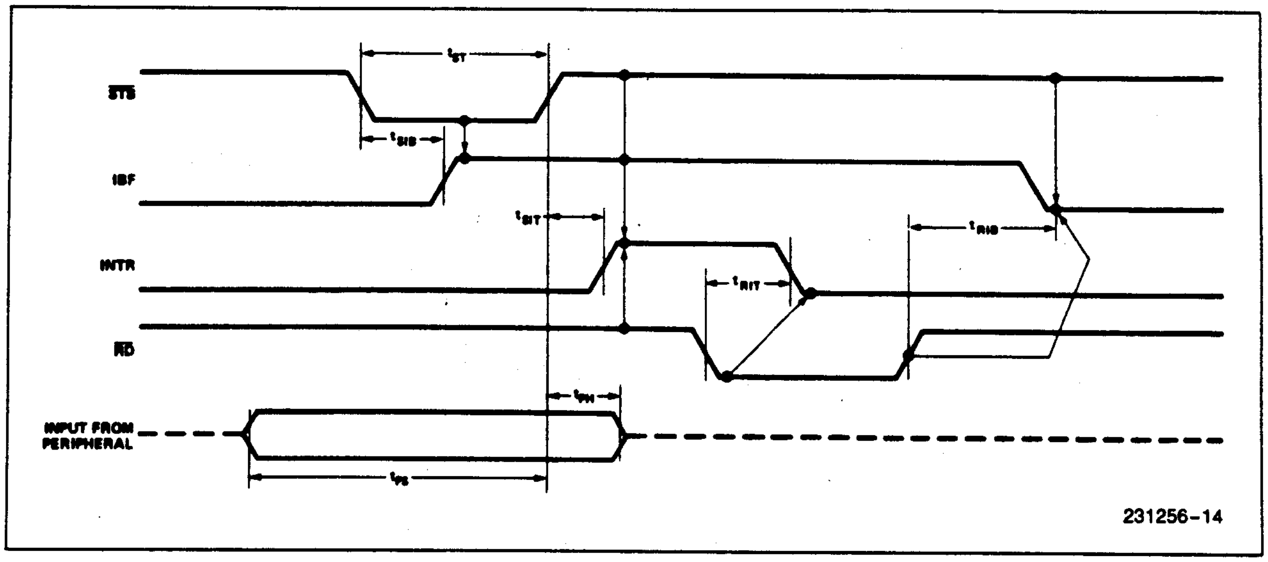


Figure 9. MODE 1 (Strobed Input)

Output Control Signal Definition

$\overline{\text{OBF}}$ (Output Buffer Full F/F). The $\overline{\text{OBF}}$ output will go "low" to indicate that the CPU has written data out to the specified port. The $\overline{\text{OBF}}$ F/F will be set by the rising edge of the $\overline{\text{WR}}$ input and reset by $\overline{\text{ACK}}$ Input being low.

$\overline{\text{ACK}}$ (Acknowledge Input). A "low" on this input informs the 82C55A that the data from Port A or Port B has been accepted. In essence, a response from the peripheral device indicating that it has received the data output by the CPU.

INTR (Interrupt Request). A "high" on this output can be used to interrupt the CPU when an output device has accepted data transmitted by the CPU. INTR is set when $\overline{\text{ACK}}$ is a "one", $\overline{\text{OBF}}$ is a "one" and INTE is a "one". It is reset by the falling edge of $\overline{\text{WR}}$.

INTE A

Controlled by bit set/reset of PC_6.

INTE B

Controlled by bit set/reset of PC_2.

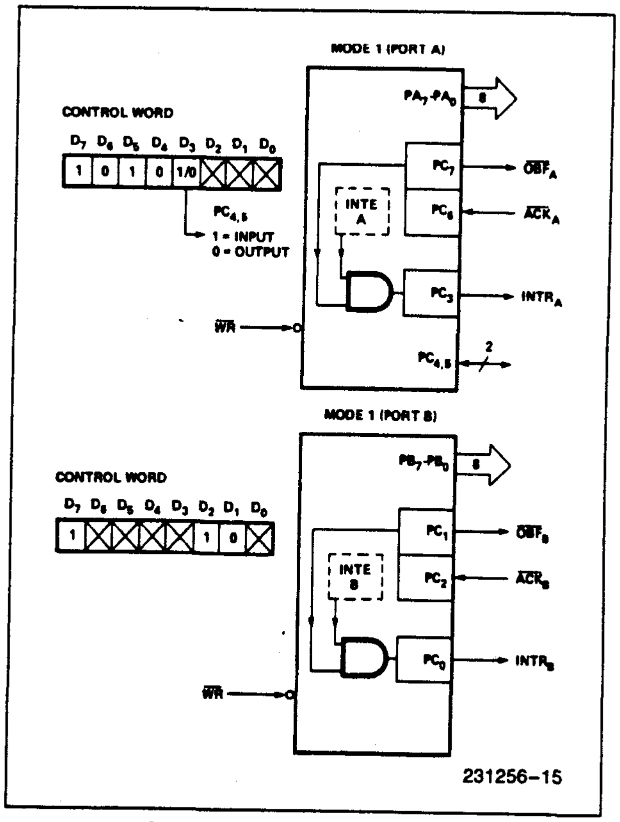


Figure 10. MODE 1 Output

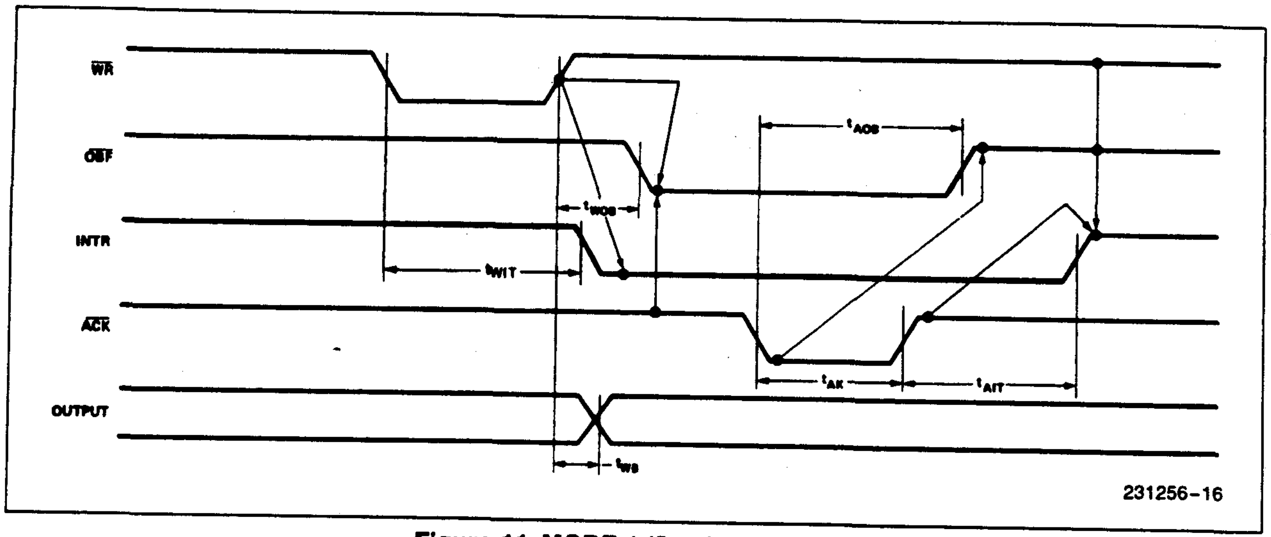


Figure 11. MODE 1 (Strobed Output)

Combinations of MODE 1

Port A and Port B can be individually defined as input or output in Mode 1 to support a wide variety of strobed I/O applications.

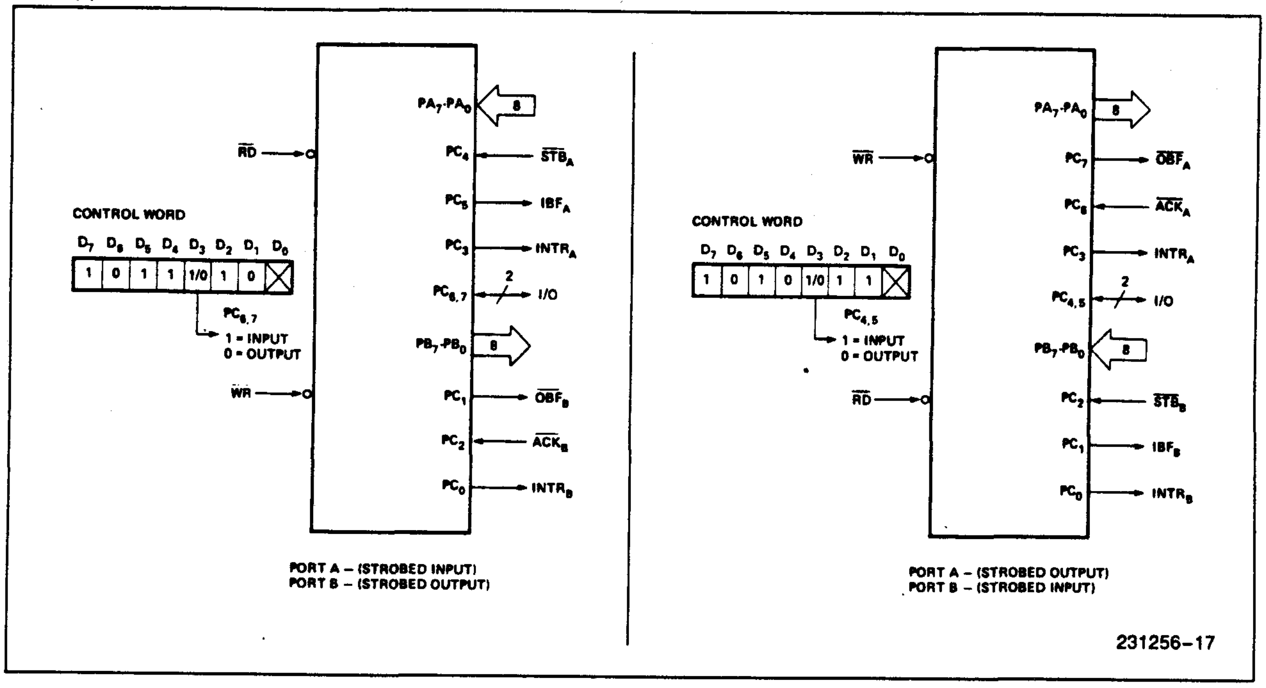


Figure 12. Combinations of MODE 1

Operating Modes

MODE 2 (Strobed Bidirectional Bus I/O). This functional configuration provides a means for communicating with a peripheral device or structure on a single 8-bit bus for both transmitting and receiving data (bidirectional bus I/O). "Handshaking" signals are provided to maintain proper bus flow discipline in a similar manner to MODE 1. Interrupt generation and enable/disable functions are also available.

MODE 2 Basic Functional Definitions:

- Used in Group A **only.**
- One 8-bit, bi-directional bus port (Port A) and a 5-bit control port (Port C).
- Both inputs and outputs are latched.
- The 5-bit control port (Port C) is used for control and status for the 8-bit, bi-directional bus port (Port A).

Bidirectional Bus I/O Control Signal Definition

INTR (Interrupt Request). A high on this output can be used to interrupt the CPU for input or output operations.

Output Operations

$\overline{\text{OBF}}$ (Output Buffer Full). The $\overline{\text{OBF}}$ output will go "low" to indicate that the CPU has written data out to port A.

$\overline{\text{ACK}}$ (Acknowledge). A "low" on this input enables the tri-state output buffer of Port A to send out the data. Otherwise, the output buffer will be in the high impedance state.

INTE 1 (The INTE Flip-Flop Associated with $\overline{\text{OBF}}$). Controlled by bit set/reset of PC_6.

Input Operations

$\overline{\text{STB}}$ (Strobe Input). A "low" on this input loads data into the input latch.

IBF (Input Buffer Full F/F). A "high" on this output indicates that data has been loaded into the input latch.

INTE 2 (The INTE Flip-Flop Associated with IBF). Controlled by bit set/reset of PC_4.

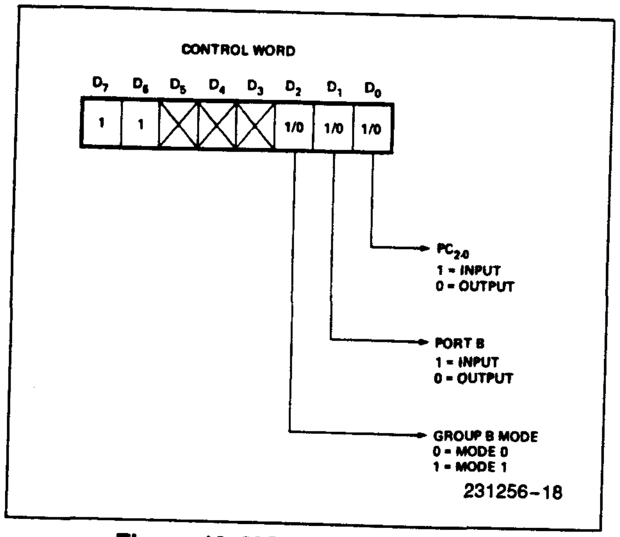


Figure 13. MODE Control Word

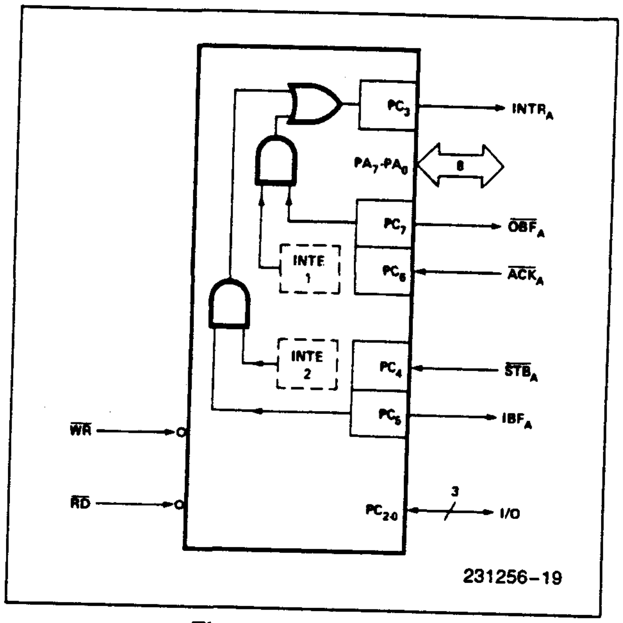


Figure 14. MODE 2

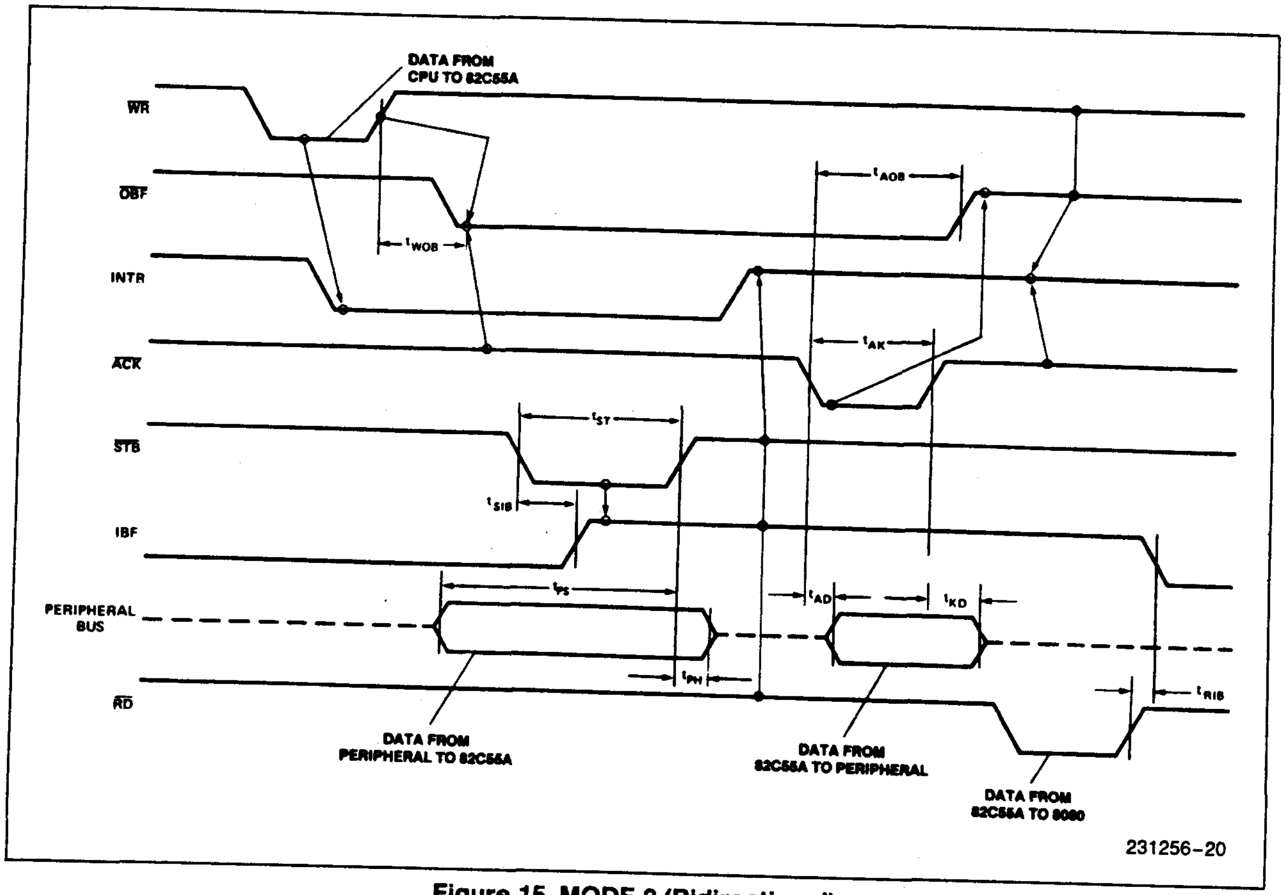


Figure 15. MODE 2 (Bidirectional)

NOTE:
Any sequence where $\overline{WR}$ occurs before $\overline{ACK}$, and $\overline{STB}$ occurs before $\overline{RD}$ is permissible.
(INTR = IBF • $\overline{MASK}$ • $\overline{STB}$ • $\overline{RD}$ + $\overline{OBF}$ • $\overline{MASK}$ • $\overline{ACK}$ • $\overline{WR}$)

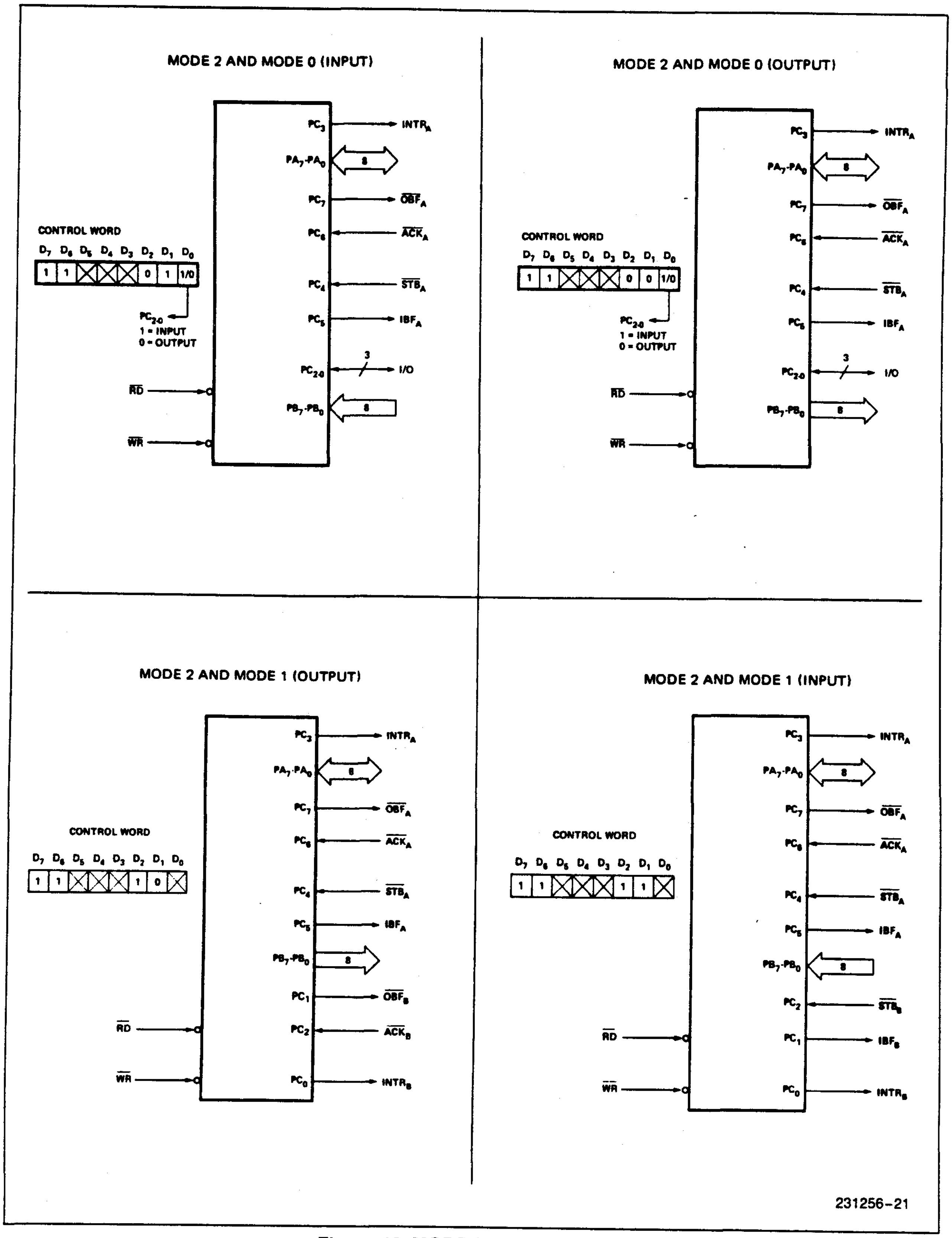


Figure 16. MODE 1/4 Combinations

Mode Definition Summary

	MODE 0		MODE 1		MODE 2
	IN	OUT	IN	OUT	GROUP A ONLY
PA_0	IN	OUT	IN	OUT	↔
PA_1	IN	OUT	IN	OUT	↔
PA_2	IN	OUT	IN	OUT	↔
PA_3	IN	OUT	IN	OUT	↔
PA_4	IN	OUT	IN	OUT	↔
PA_5	IN	OUT	IN	OUT	↔
PA_6	IN	OUT	IN	OUT	↔
PA_7	IN	OUT	IN	OUT	↔
PB_0	IN	OUT	IN	OUT	—
PB_1	IN	OUT	IN	OUT	—
PB_2	IN	OUT	IN	OUT	—
PB_3	IN	OUT	IN	OUT	—
PB_4	IN	OUT	IN	OUT	—
PB_5	IN	OUT	IN	OUT	—
PB_6	IN	OUT	IN	OUT	—
PB_7	IN	OUT	IN	OUT	—
PC_0	IN	OUT	$INTR_B$	$INTR_B$	I/O
PC_1	IN	OUT	IBF_B	$\overline{OBF}_B$	I/O
PC_2	IN	OUT	$\overline{STB}_B$	$\overline{ACK}_B$	I/O
PC_3	IN	OUT	$INTR_A$	$INTR_A$	$INTR_A$
PC_4	IN	OUT	$\overline{STB}_A$	I/O	$\overline{STB}_A$
PC_5	IN	OUT	IBF_A	I/O	IBF_A
PC_6	IN	OUT	I/O	$\overline{ACK}_A$	$\overline{ACK}_A$
PC_7	IN	OUT	I/O	$\overline{OBF}_A$	$\overline{OBF}_A$

MODE 0 OR MODE 1 ONLY

Special Mode Combination Considerations

There are several combinations of modes possible. For any combination, some or all of the Port C lines are used for control or status. The remaining bits are either inputs or outputs as defined by a "Set Mode" command.

During a read of Port C, the state of all the Port C lines, except the $\overline{ACK}$ and $\overline{STB}$ lines, will be placed on the data bus. In place of the $\overline{ACK}$ and $\overline{STB}$ line states, flag status will appear on the data bus in the PC2, PC4, and PC6 bit positions as illustrated by Figure 18.

Through a "Write Port C" command, only the Port C pins programmed as outputs in a Mode 0 group can be written. No other pins can be affected by a "Write Port C" command, nor can the interrupt enable flags be accessed. To write to any Port C output programmed as an output in a Mode 1 group or to change an interrupt enable flag, the "Set/Reset Port C Bit" command must be used.

With a "Set/Reset Port C Bit" command, any Port C line programmed as an output (including INTR, IBF and $\overline{OBF}$) can be written, or an interrupt enable flag can be either set or reset. Port C lines programmed as inputs, including $\overline{ACK}$ and $\overline{STB}$ lines, associated with Port C are not affected by a "Set/Reset Port C Bit" command. Writing to the corresponding Port C bit positions of the $\overline{ACK}$ and $\overline{STB}$ lines with the "Set/Reset Port C Bit" command will affect the Group A and Group B interrupt enable flags, as illustrated in Figure 18.

Current Drive Capability

Any output on Port A, B or C can sink or source 2.5 mA. This feature allows the 82C55A to directly drive Darlington type drivers and high-voltage displays that require such sink or source current.

Reading Port C Status

In Mode 0, Port C transfers data to or from the peripheral device. When the 82C55A is programmed to function in Modes 1 or 2, Port C generates or accepts "hand-shaking" signals with the peripheral device. Reading the contents of Port C allows the programmer to test or verify the "status" of each peripheral device and change the program flow accordingly.

There is no special instruction to read the status information from Port C. A normal read operation of Port C is executed to perform this function.

INPUT CONFIGURATION

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
I/O	I/O	IBF_A	$INTE_A$	$INTR_A$	$INTE_B$	IBF_B	$INTR_B$
GROUP A					GROUP B		

OUTPUT CONFIGURATIONS

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
$\overline{OBF}_A$	$INTE_A$	I/O	I/O	$INTR_A$	$INTE_B$	$\overline{OBF}_B$	$INTR_B$
GROUP A					GROUP B		

Figure 17a. MODE 1 Status Word Format

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
$\overline{OBF}_A$	$INTE_1$	IBF_A	$INTE_2$	$INTR_A$			
GROUP A					GROUP B		

(Defined By Mode 0 or Mode 1 Selection)

Figure 17b. MODE 2 Status Word Format

Interrupt Enable Flag	Position	Alternate Port C Pin Signal (Mode)
INTE B	PC2	$\overline{ACK}_B$ (Output Mode 1) or $\overline{STB}_B$ (Input Mode 1)
INTE A2	PC4	$\overline{STB}_A$ (Input Mode 1 or Mode 2)
INTE A1	PC6	$\overline{ACK}_A$ (Output Mode 1 or Mode 2

Figure 18. Interrupt Enable Flags in Modes 1 and 2

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias	0°C to + 70°C
Storage Temperature	− 65°C to + 150°C
Supply Voltage	− 0.5 to + 8.0V
Operating Voltage	+ 4V to + 7V
Voltage on any Input	GND − 2V to + 6.5V
Voltage on any Output	GND − 0.5V to V_{CC} + 0.5V
Power Dissipation	1 Watt

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = +5V ±10%, GND = 0V (T_A = −40°C to +85°C for Extended Temperture)

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage	−0.5	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 2.5 mA
V_{OH}	Output High Voltage	3.0 V_{CC} − 0.4		V V	I_{OH} = −2.5 mA I_{OH} = −100 μA
I_{IL}	Input Leakage Current		±1	μA	V_{IN} = V_{CC} to 0V (Note 1)
I_{OFL}	Output Float Leakage Current		±10	μA	V_{IN} = V_{CC} to 0V (Note 2)
I_{DAR}	Darlington Drive Current	±2.5	(Note 4)	mA	Ports A, B, C R_{ext} = 500Ω V_{ext} = 1.7V
I_{PHL}	Port Hold Low Leakage Current	+50	+300	μA	V_{OUT} = 1.0V Port A only
I_{PHH}	Port Hold High Leakage Current	−50	−300	μA	V_{OUT} = 3.0V Ports A, B, C
I_{PHLO}	Port Hold Low Overdrive Current	−350		μA	V_{OUT} = 0.8V
I_{PHHO}	Port Hold High Overdrive Current	+350		μA	V_{OUT} = 3.0V
I_{CC}	V_{CC} Supply Current		10	mA	(Note 3)
I_{CCSB}	V_{CC} Supply Current-Standby		10	μA	V_{CC} = 5.5V V_{IN} = V_{CC} or GND Port Conditions If I/P = Open/High O/P = Open Only With Data Bus = High/Low $\overline{CS}$ = High Reset = Low Pure Inputs = Low/High

NOTES:
1. Pins A_1, A_0, $\overline{CS}$, $\overline{WR}$, $\overline{RD}$, Reset.
2. Data Bus; Ports B, C.
3. Outputs open.
4. Limit output current to 4.0 mA.

CAPACITANCE

$T_A = 25°C, V_{CC} = GND = 0V$

Symbol	Parameter	Min	Max	Units	Test Conditions
C_{IN}	Input Capacitance		10	pF	Unmeasured pins returned to GND $f_c = 1$ MHz[(5)]
$C_{I/O}$	I/O Capacitance		20	pF	

NOTE:
5. Sampled not 100% tested.

A.C. CHARACTERISTICS

$T_A = 0°$ to $70°C, V_{CC} = +5V \pm 10\%, GND = 0V$

$T_A = -40°C$ to $+85°C$ for Extended Temperature

BUS PARAMETERS

READ CYCLE

Symbol	Parameter	82C55A-2		Units	Test Conditions
		Min	Max		
t_{AR}	Address Stable Before $\overline{RD}\downarrow$	0		ns	
t_{RA}	Address Hold Time After $\overline{RD}\uparrow$	0		ns	
t_{RR}	$\overline{RD}$ Pulse Width	150		ns	
t_{RD}	Data Delay from $\overline{RD}\downarrow$		120	ns	
t_{DF}	$\overline{RD}\uparrow$ to Data Floating	10	75	ns	
t_{RV}	Recovery Time between $\overline{RD}/\overline{WR}$	200		ns	

WRITE CYCLE

Symbol	Parameter	82C55A-2		Units	Test Conditions
		Min	Max		
t_{AW}	Address Stable Before $\overline{WR}\downarrow$	0		ns	
t_{WA}	Address Hold Time After $\overline{WR}\uparrow$	20		ns	Ports A & B
		20		ns	Port C
t_{WW}	$\overline{WR}$ Pulse Width	100		ns	
t_{DW}	Data Setup Time Before $\overline{WR}\uparrow$	100		ns	
t_{WD}	Data Hold Time After $\overline{WR}\uparrow$	30		ns	Ports A & B
		30		ns	Port C

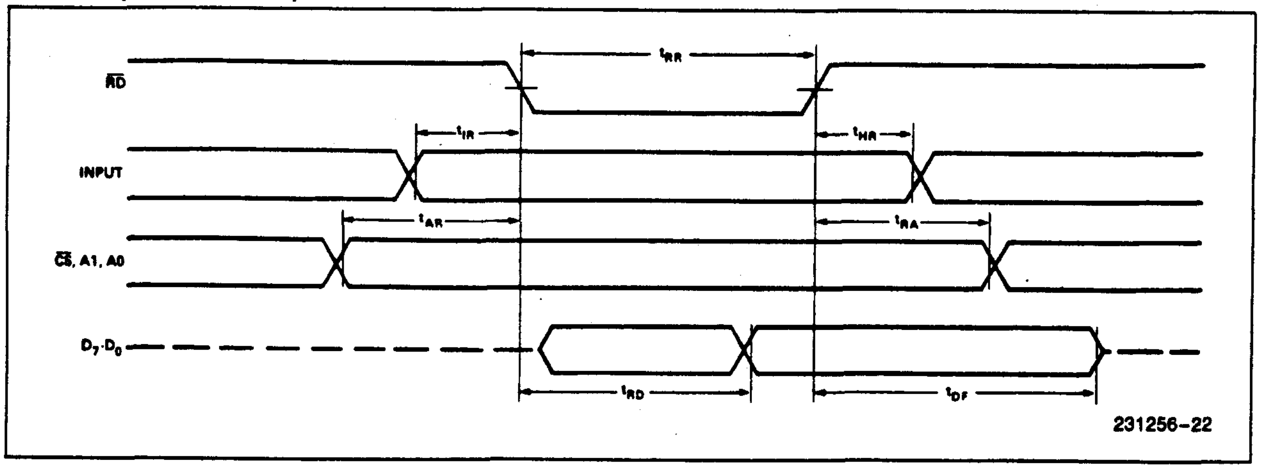
OTHER TIMINGS

Symbol	Parameter	82C55A-2		Units Conditions	Test
		Min	Max		
t_{WB}	$\overline{WR}$ = 1 to Output		350	ns	
t_{IR}	Peripheral Data Before $\overline{RD}$	0		ns	
t_{HR}	Peripheral Data After $\overline{RD}$	0		ns	
t_{AK}	$\overline{ACK}$ Pulse Width	200		ns	
t_{ST}	$\overline{STB}$ Pulse Width	100		ns	
t_{PS}	Per. Data Before $\overline{STB}$ High	20		ns	
t_{PH}	Per. Data After $\overline{STB}$High	50		ns	
t_{AD}	$\overline{ACK}$ = 0 to Output		175	ns	
t_{KD}	$\overline{ACK}$ = 1 to Output Float	20	250	ns	
t_{WOB}	$\overline{WR}$ = 1 to $\overline{OBF}$ = O		150	ns	
t_{AOB}	$\overline{ACK}$ = 0 to $\overline{OBF}$ = 1		150	ns	
t_{SIB}	$\overline{STB}$ = 0 to IBF = 1		150	ns	
t_{RIB}	$\overline{RD}$ = 1 to IBF = 0		150	ns	
t_{RIT}	$\overline{RD}$ = 0 to INTR = 0		200	ns	
t_{SIT}	$\overline{STB}$ = 1 to INTR = 1		150	ns	
t_{AIT}	$\overline{ACK}$ = 1 to INTR = 1		150	ns	
t_{WIT}	$\overline{WR}$ = 0 to INTR = 0		200	ns	see note 1
t_{RES}	Reset Pulse Width	500		ns	see note 2

NOTE:
1. INTR ↑ may occur as early as $\overline{WR}$ ↓.
2. Pulse width of initial Reset pulse after power on must be at least 50 μSec. Subsequent Reset pulses may be 500 ns minimum.

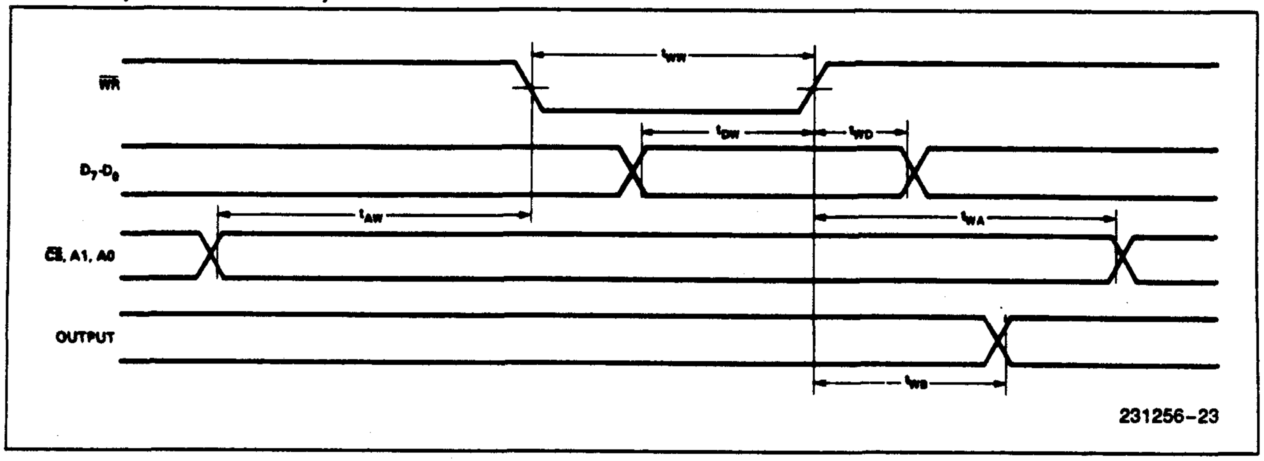
WAVEFORMS

MODE 0 (BASIC INPUT)



231256-22

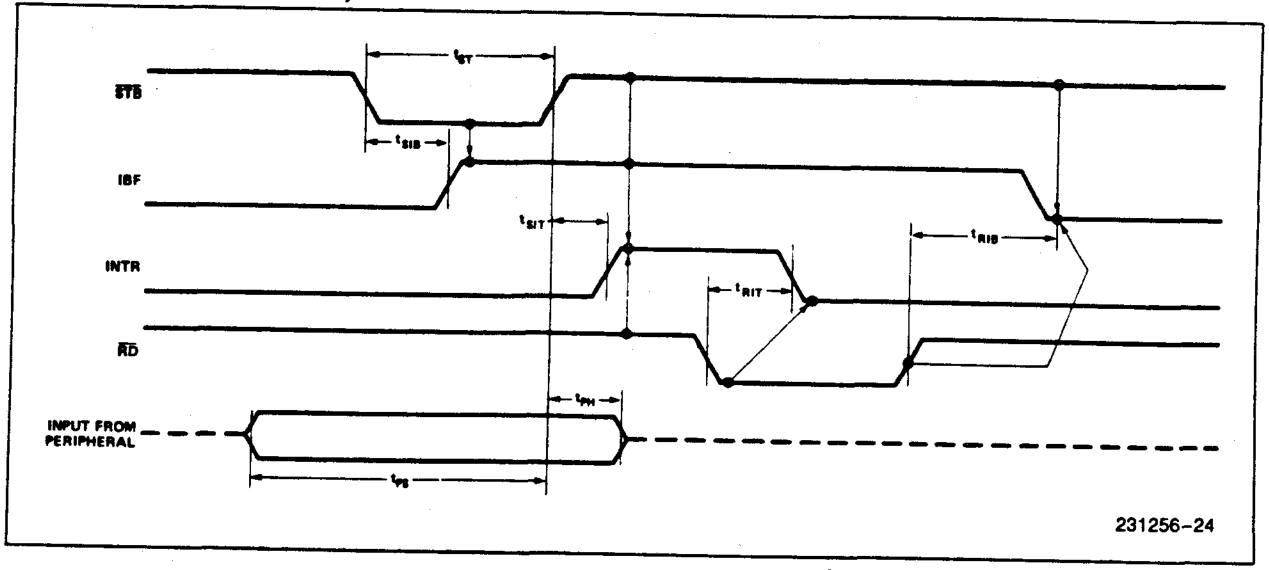
MODE 0 (BASIC OUTPUT)



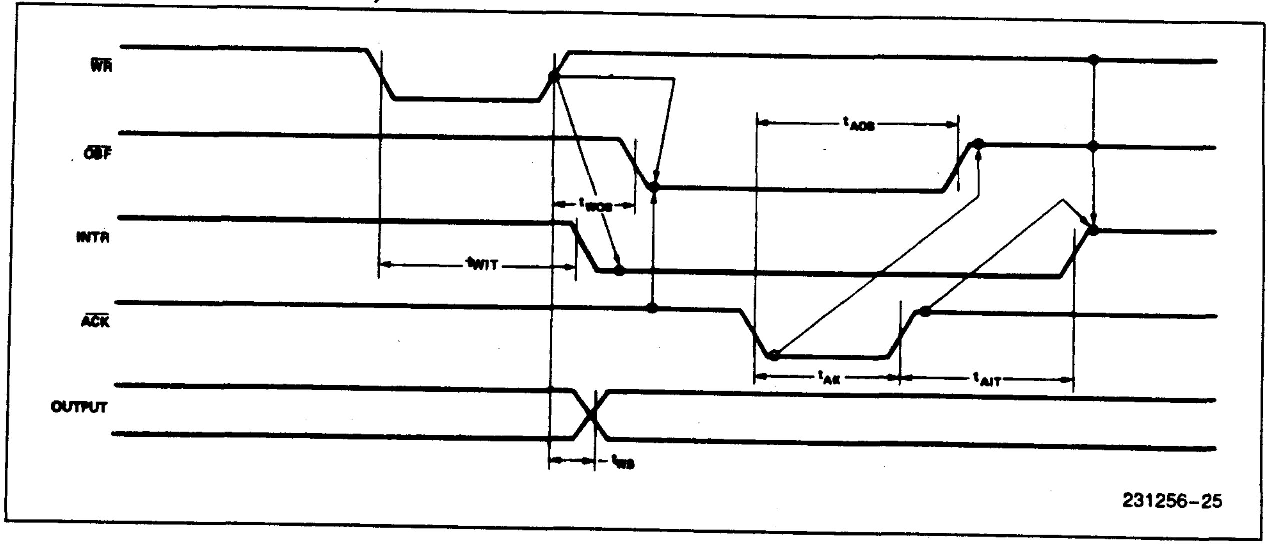
231256-23

WAVEFORMS (Continued)

MODE 1 (STROBED INPUT)

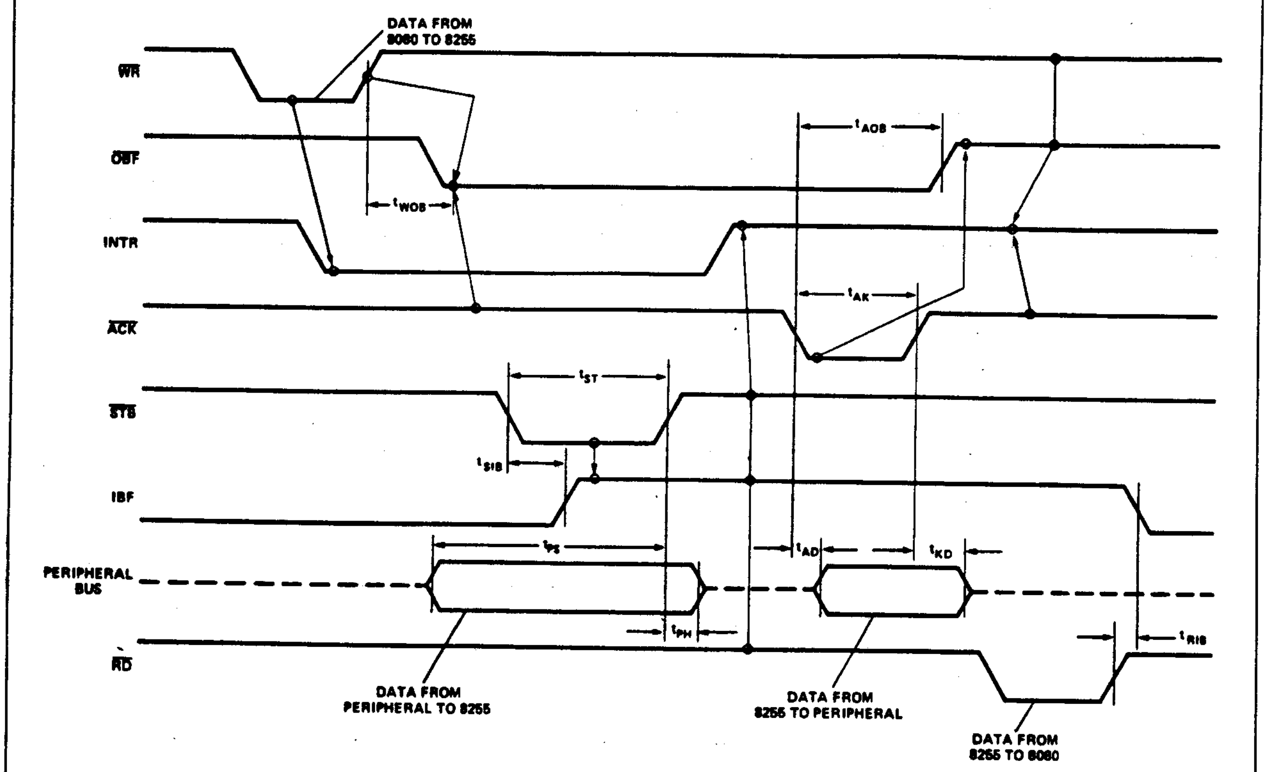


MODE 1 (STROBED OUTPUT)



WAVEFORMS (Continued)

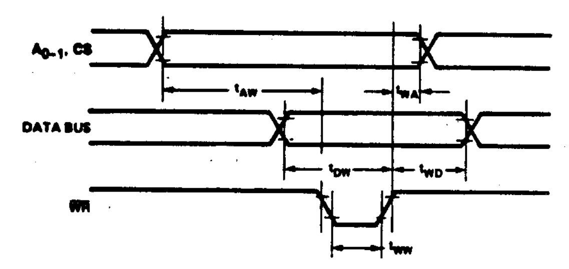
MODE 2 (BIDIRECTIONAL)



231256-26

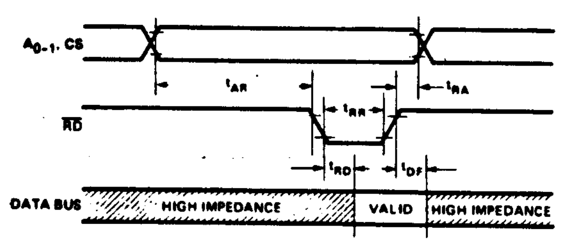
Note:
Any sequence where $\overline{WR}$ occurs before $\overline{ACK}$ AND $\overline{STB}$ occurs before $\overline{RD}$ is permissible.
(INTR = IBF • $\overline{MASK}$ • $\overline{STB}$ • $\overline{RD}$ + $\overline{OBF}$ • $\overline{MASK}$ • $\overline{ACK}$ • $\overline{WR}$)

WRITE TIMING



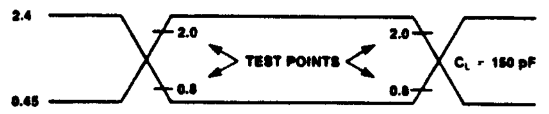
231256-27

READ TIMING



231256-28

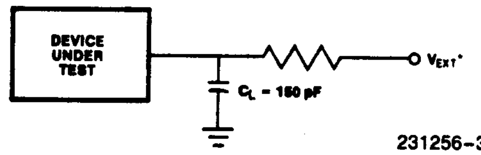
A.C. TESTING INPUT, OUTPUT WAVEFORM



231256-29

A.C. Testing Inputs Are Driven At 2.4V For A Logic 1 And 0.45V For A Logic 0 Timing Measurements Are Made At 2.0V For A Logic 1 And 0.8 For A Logic 0.

A.C. TESTING LOAD CIRCUIT



231256-30

*V_{EXT} Is Set At Various Voltages During Testing To Guarantee The Specification. C_L Includes Jig Capacitance.

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.

AD1210/ADA1210 User Settings	
Base I/O Address:	
(hex)	(decimal)
IRQ Channel:	
DMA Channel:	