# DIO24
# User's Manual

Real Time Devices, Inc.

*"Accessing the Analog World"*™

# DIO24
# User's Manual

**REAL TIME DEVICES, INC.**
Post Office Box 906
State College, Pennsylvania 16804

Phone: (814) 234-8087
FAX: (814) 234-5218

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

Printed in U.S.A.

# Table of Contents

# LIST OF ILLUSTRATIONS

# INTRODUCTION

The DIO24 series boards are general purpose digital I/O board for use in the IBM PC/XT/AT or compatible computer. The DIO24 series has two models: the DIO24-1 and the DIO24-2 with timer/counters. Installed within a single short or full-size expansion slot in the computer, the DIO24 features:

- 24 TTL/CMOS 8255-based programmable digital I/O lines,
- Direct connection to opto-22 I/O system modules,
- Buffered outputs for high driving capability,
- Three 16-bit timer/counters (DIO24-2 model only),
- Optional pull-up/pull-down resistors,
- Simple I/O or strobed I/O operation,
- Software enabled interrupts (IRQ2-IRQ7),
- Assembly, BASIC, Turbo Pascal, and Turbo C source code; diagnostics program.

The following paragraphs briefly describe the major function 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.*

## Digital I/O

The DIO24 has 24 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. The 8255 can be operated in one of two modes: Mode 0 or Mode 1. To ensure high driving capacity, CMOS buffers are installed. TTL buffers are available on request.

Pads for installing and activating pull-up or pull-down resistors are included on the board. Installation procedures are given at the end of Chapter 1, *Board Settings.*

## 8254 Timer/Counter (DIO24-2)

An 8254 programmable interval timer contains three 16-bit, 8 MHz timer/counters to support a wide range of timing and counting functions. The clock, gate, and output pins for each of the timer/counters are available at P6, a 20-pin on-board box header connector.

## What Comes With Your Board

You receive the following items in your DIO24 package:

- DIO24-1 or DIO24-2 (with timer/counters) interface board
- Software and diagnostics diskette with Assembly, BASIC, Turbo Pascal, and Turbo C 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 DIO24 package, Real Time Devices offers a full line of 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.

Accessories for the DIO24 include the TB50 terminal board and XB50 prototype/terminal board for prototype development and easy signal access, the XO50 cable assembly for direct connection to opto-22 systems, and the TW50 twisted pair wire 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 DIO24 board has jumper and switch settings you can change if necessary for your application. The board is factory-configured with the most often used settings. The factory settings are listed and shown on a diagram in the beginning of this chapter. Should you need to change these settings, use these easy-to-follow instructions before you install the board in your computer.

Note that DIP switch S2 has been provided to bypass the Port C buffers and allow Mode 1 operation of the 8255.

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

## Factory-Configured Switch and Jumper Settings

Table 1-1 lists the factory settings of the user-configurable jumper and switches on the DIO24 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 | Connects 1 of 6 interrupt sources to an interrupt channel; pulls tri-state buffer to ground (GND) for multiple interrupt applications | GND (ground for buffer) connected; interrupt channels disabled |
| P4 | Selects the interrupt source | EXT (external interrupt) |
| P5 | Sets the clock sources for the three 8254 timer/counters (TC0-TC2) | CLK0-XTAL; CLK1-OUT0; CLK2–OUT1 (all three timer/counters are cascaded) |
| S1 | Sets the base address | 300 hex (768 decimal) |
| S2 | Bypasses Port C buffers for Mode 1 operation | Open (buffers not bypassed) |

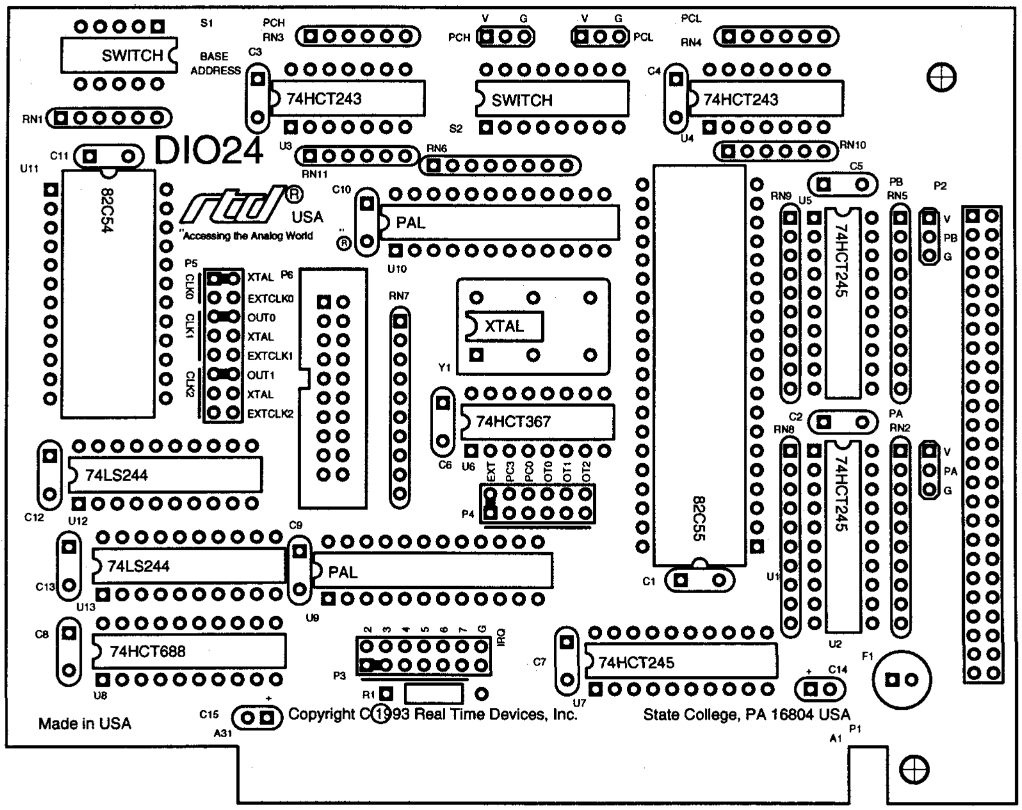


Fig. 1-1 — Board Layout Showing Factory-Configured Settings

### P3 — Interrupt and Interrupt Channels (Factory Setting: GND Connected; Interrupt Channels Disabled)

This header connector, shown in Figure 1-2, lets you connect an interrupt source selected on P4 to an interrupt channel, IRQ2 through IRQ7. To connect the interrupt source to an interrupt channel, you must install a jumper across the desired IRQ channel.

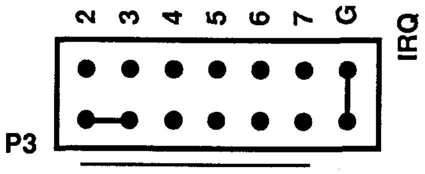


Fig. 1-2 — Interrupt and Interrupt Channel Jumper, P3

The rightmost pair of pins on P3, labeled G, are provided so that you can install a jumper which connects a 1 kilohm 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. So, whenever an interrupt request is made, the tri-state buffer is enabled, forcing the output high and causing an interrupt. You can monitor the interrupt status through bit 0 in the status word (I/O address location BA + 5). After the interrupt has been serviced, the clear command returns the IRQ line low, disabling the tri-state buffers, and pulling the output low again. Figure 1-3 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 ground jumper installed. The rest should be disconnected. Whenever you operate a single board, the G jumper should be installed.

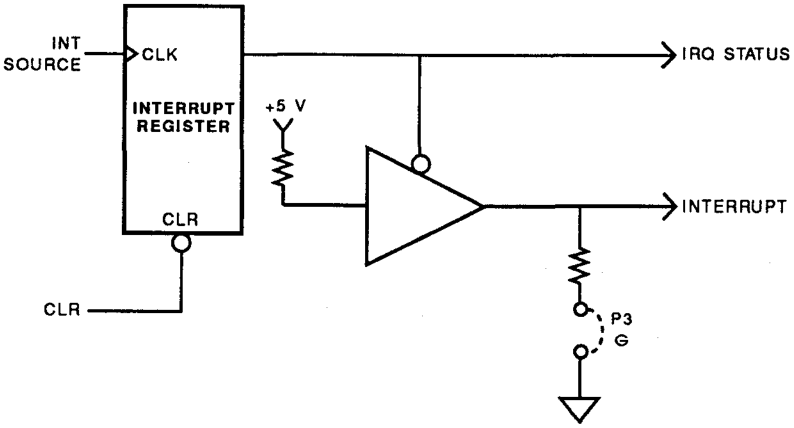


Fig. 1-3 — Pulling Down the Interrupt Request Line

### P4 — Interrupt Source Select (Factory Setting: EXT)

This header connector, shown in Figure 1-4, lets you connect one of six interrupt sources for interrupt generation. These sources are: OT0, OT1, and OT2, which are the three 8254 timer/counter outputs available on the DIO24-2 model; PC3, which is the INTRA signal from the 8255 PPI; PC0, which is the INTRB signal from the 8255 PPI; and EXT, an external interrupt you can route onto the board through the P2 I/O connector. To connect an interrupt source, place the jumper across the desired set of pins. Note that only ONE interrupt source can be activated at a time.

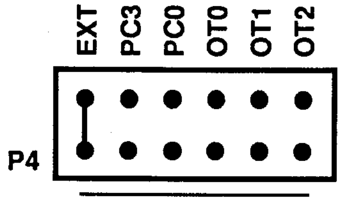


Fig. 1-4 — Interrupt Source Select Jumper, P4

**P5 — 8254 Timer/Counter Clock Sources (Factory Settings: CLK0-XTAL, CLK1-OUT0, CLK2-OUT1)**

This header connector, shown in Figure 1-5, lets you select the clock sources for the 8254 timer/counters, TC0, TC1, and TC2 which are on the DIO24-2 model. The factory setting cascades all three timer/counters, with the clock source for TC0 being the on-board 8 MHz oscillator, the output of TC0 providing the clock for TC1, and the output of TC1 providing the clock for TC2. You can connect any or all of the sources to an external clock input through the P6 on-board I/O connector, or you can set TC1 and TC2 to be clocked by the 8 MHz oscillator. Figure 1-6 shows a block diagram of the timer/counter circuitry to help you with these connections.

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

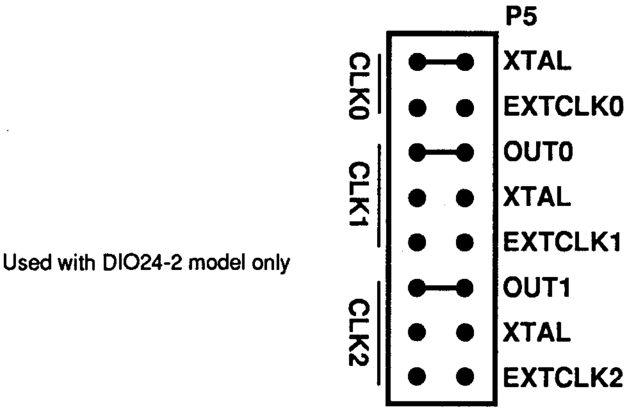


Fig. 1-5 — 8254 Timer/Counter Clock Source Jumpers, P5

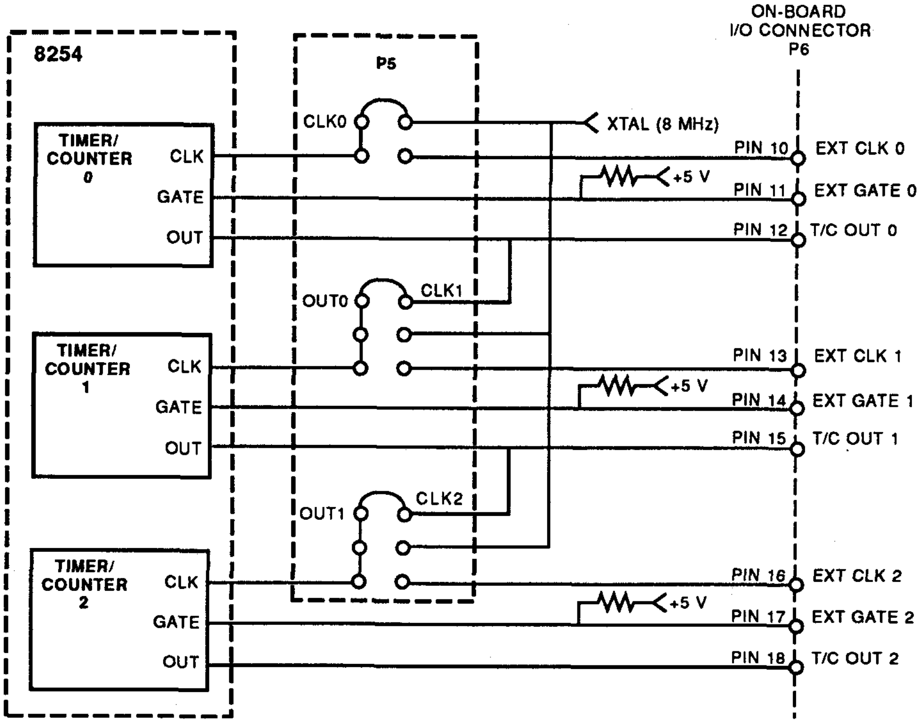


Fig. 1-6 — 8254 Timer/Counter Circuit Block Diagram

### 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 DIO24 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 DIO24 has an easily accessible 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 value shown 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 1-7 shows the DIP switch set for a base address of 300 hex (768 decimal).

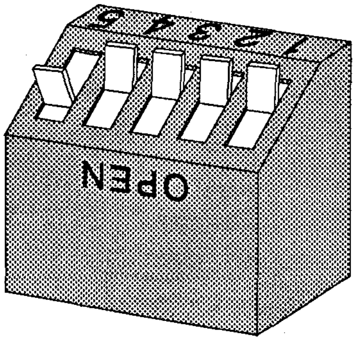


Fig. 1-7 — Base Address Switch, S1

**Table 1-2 — Base Address Switch Settings, S1**

| Base Address<br>Decimal / (Hex) | Switch Setting<br>5 4 3 2 1 | Base Address<br>Decimal / (Hex) | Switch Setting<br>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**

### S2 — Buffer Bypass Switch (Factory Setting: OPEN (Not Bypassed))

When operating the 8255 in Mode 1, the lines of Port C function as control lines, some as outputs and some as inputs. When using Mode 1, the Port C buffers must be removed and bypassed to allow the Port C lines to be individually set as inputs or outputs. Figure 1-8 shows the Port C buffers, and the following steps tell you how to configure the board for Mode 1 operation.

To remove buffering from Port C:

1. Close DIP switches 1 through 8 on S2.
2. Remove U3 from the board.
3. Remove U4 from the board.

**CAUTION:** Remember, whenever you close the switches, **be sure** to remove the buffers, U3 and U4, from the board. Failure to do so may damage the board.

### F1 — External +5-volt Fuse

This 1 ampere fuse protects the +5 volt line available at I/O connector P2, pin 49 from drawing too much current and damaging system equipment.

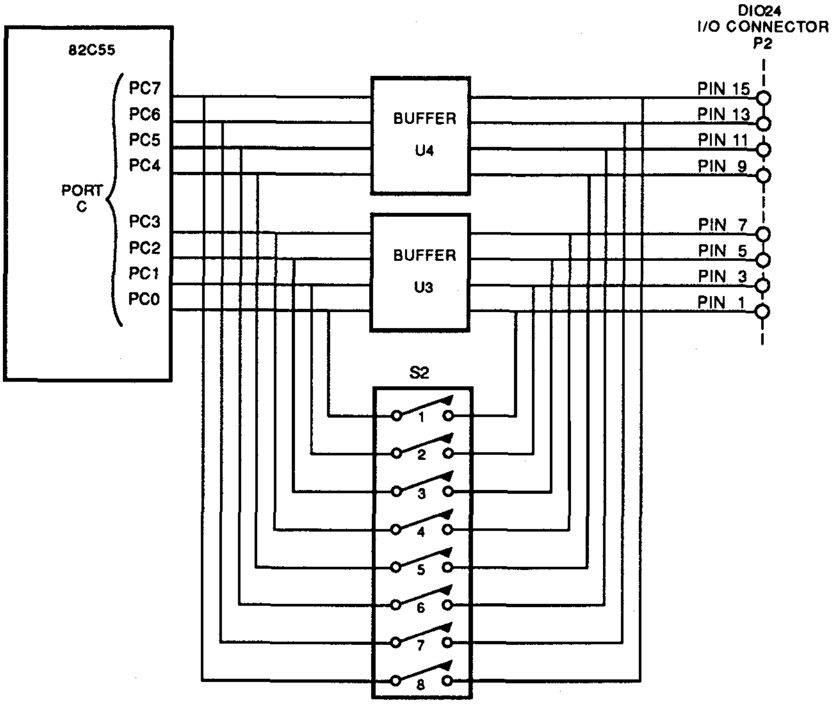


Fig. 1-8 — Port C Buffer Circuitry

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

The 8255 programmable peripheral interface provides 24 parallel TTL/CMOS compatible digital I/O lines which can be interfaced with external devices. The lines are divided into four groups: eight Port A lines, four Port C Lower lines, eight Port B lines, and four Port C Upper lines. You can install and connect pull-up or pull-down resistors for any or all of these four 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 down lines connected to relays which control turning motors on and off. These motors turn on when the digital lines controlling them are high.

To use the pull-up/pull-down feature, you must first install 10 kilohm resistor packs in any or all of the four locations around the 8255, labeled PA, PB, PCL, and PCH. PA and PB take a 10-pin pack, and CL and CH take 6-pin packs. Figure 1-9 shows these 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 near the resistor packs. They are labeled G (for ground) on one end and V (for Vcc) on the other end. The middle hole is common. PA is for Port A, PB for Port B, CL is for Port C Lower, and CH is for Port C Upper. Figure 1-9 shows a blowup of the pads for Port A. 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. For example, Figure 1-10 shows Port A lines with pull-ups, Port C Lower with pull-downs, and Port C Upper with no resistors.

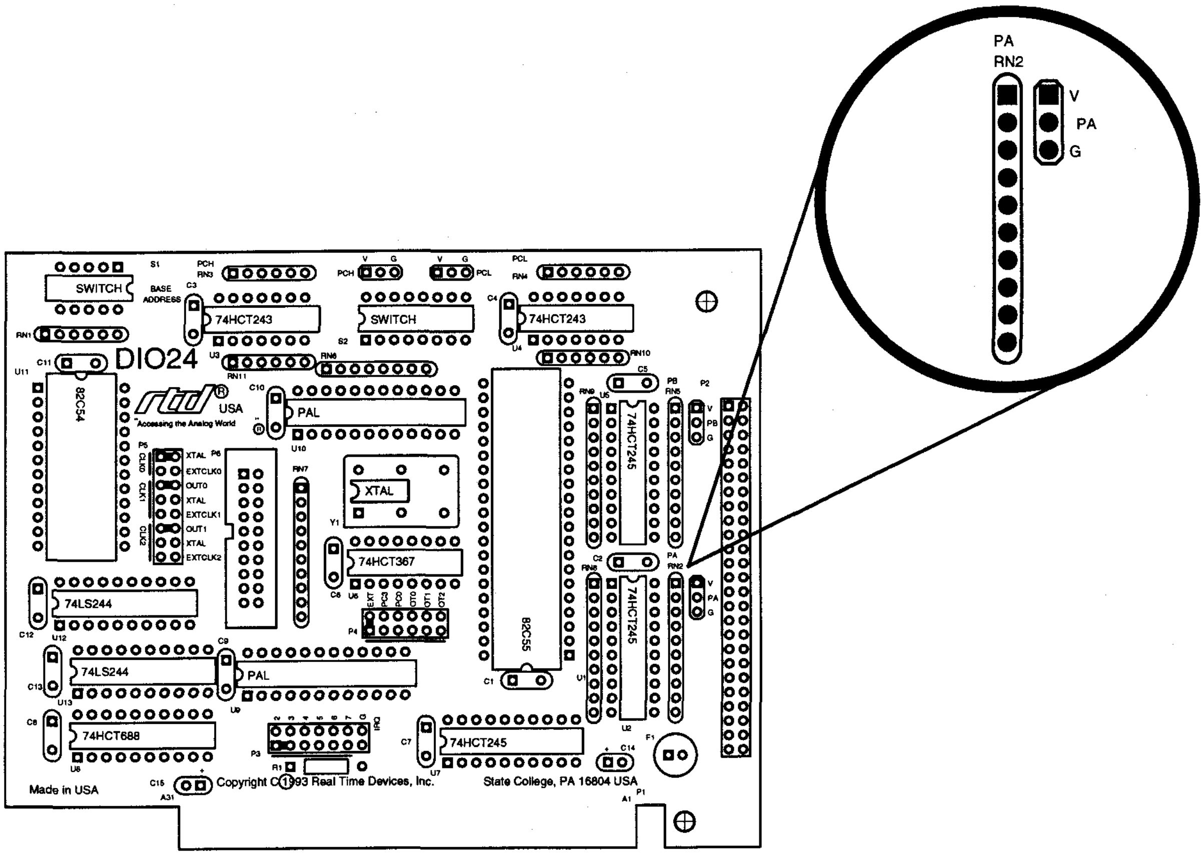


Fig. 1-9 — Port A Pull-up/Pull-down Resistor Circuitry

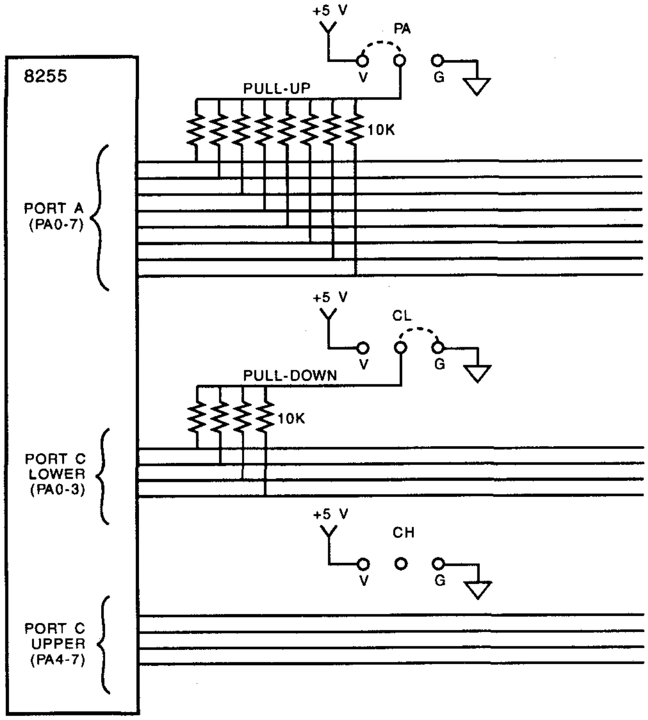


Fig. 1-10 — Adding Pull-ups and Pull-downs to Some Digital I/O Lines

# CHAPTER 2

## BOARD INSTALLATION

The DIO24 is easy to install in your IBM PC/XT/AT. 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 24DIAG 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 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.

Also note that the P2 I/O connector mounting bracket has an oversized cutout to allow space for running the cable to 20-pin on-board connector P6 through the same I/O slot. If you want to run both cables through the same slot, you must make these connections before installing the board.

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 DIO24's P2 I/O connector pinout and P6 on-board I/O connector pinout. Refer to these diagrams as you make your I/O connections.

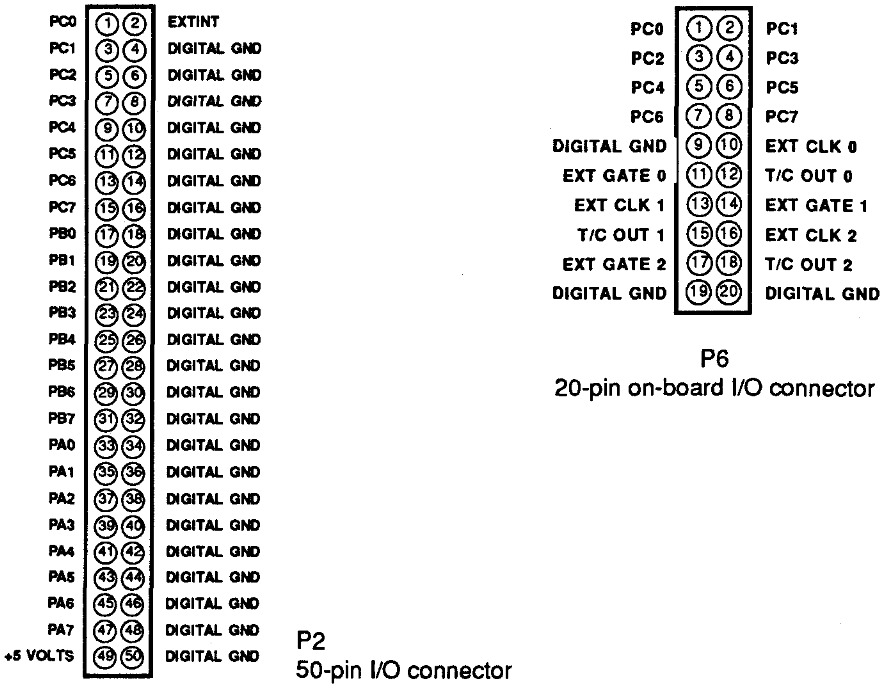


Fig. 2-1 — P2 and P6 I/O Connector Pin Assignments

### Connecting the Digital I/O

The DIO24 is designed for direct connection to industry standard opto-22 isolated I/O racks and system modules. Each digital I/O line on P2 has a digital ground, as shown in Figure 2-1. For all digital I/O connections, the high side of an external signal source or destination device is connected to the appropriate signal pin on the I/O connector, and the low side is connected to the DIGITAL GND. A cable to provide direct connection to opto-22 systems is available as an accessory from RTD.

### Connecting the Timer/Counter I/O

External connections to the timer/counters on the DIO24-2 model can be made by connecting the high side of the external device to the appropriate signal pin on on-board connector P6 and the low side to a P6 DIGITAL GND.

### Connecting the External Interrupt

The DIO24 can receive an externally generated interrupt signal, EXTINT, through I/O connector P2, pin 2 and route it to an IRQ channel through on-board header connectors P3 and P4. Interrupt generation is enabled through software. When interrupts are enabled, a rising edge on the EXTINT line will cause the selected IRQ line to go high, and the IRQ status bit will change from 0 to 1. The pulse applied to the EXTINT pin should have a duration of at least 100 nanoseconds.

## Running the 24DIAG 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, 24DIAG, 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 major features of the DIO24's 8255 based digital I/O and 8254 timer/counters. This chapter also describes the hardware-selectable interrupts.

The DIO24 provides buffered digital I/O lines, as shown Figure 3-1. In addition, the DIO24-2 model providesthree 16-bit timer/counters. This chapter describes the hardware which makes up the digital I/O circuitry, timer/counters, and hardware-selectable interrupts.

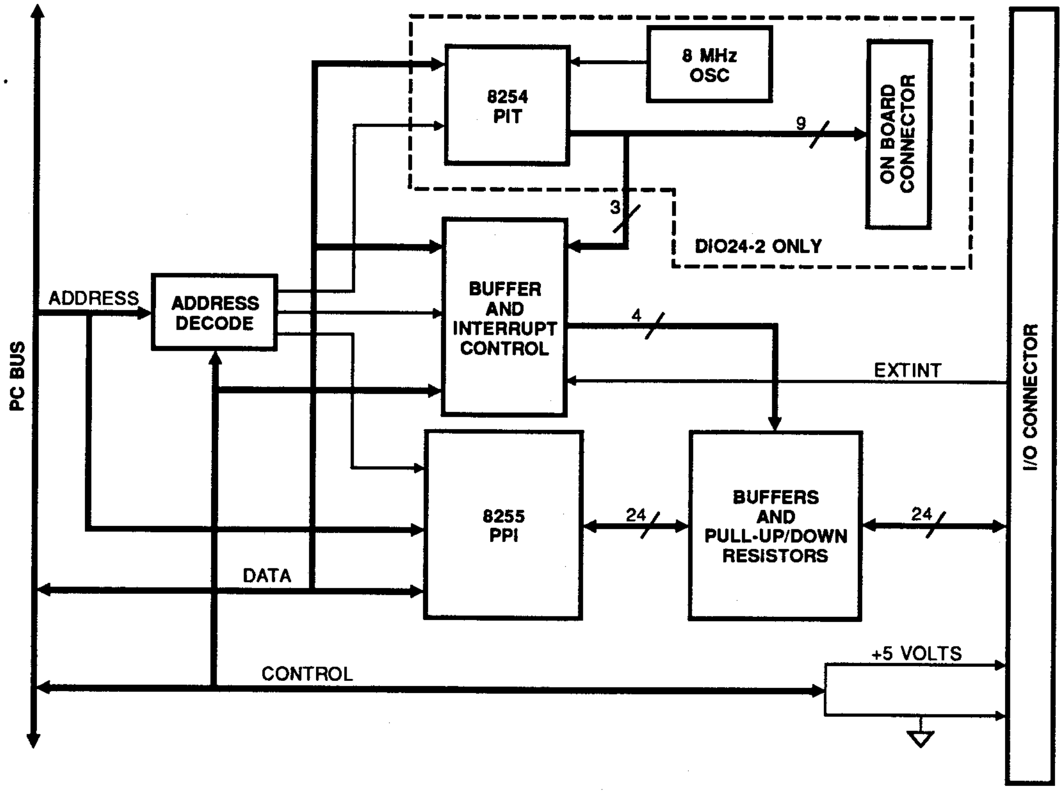


Fig. 3-1 — DIO24 Block Diagram

## Digital I/O, 8255 Programmable Peripheral Interface

The 8255 programmable peripheral interface (PPI) can be easily configured to solve a wide range of digital real-world problems. This high-performance TTL/CMOS compatible chip has 24 parallel programmable 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).

Each group can be programmed for Mode 0 or Mode 1 operation. **Do not try to use Mode 2 operation!** The DIO24 does not support Mode 2. When operating in Mode 1, the on-board buffers must be removed from the Port C lines. This procedure is described in Chapter 1 in the S2 DIP switch discussion. The DIO24 operating modes are:

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 or Port B in conjunction with strobes or handshaking signals.

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

The bidirectional buffers on the 8255's I/O lines monitor the 8255 control word to automatically set their direction. Hardware changes to the buffer circuitry is required only when using Mode 1, where the Port C buffers must be removed as described in Chapter 1.

## Timer/Counters

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

Each timer/counter has two inputs, CLK in and GATE in, and one output, timer/counter OUT. The clock sources for the timer/counters can be selected using jumpers on header connector P5 (see Chapter 1). The timer/counters 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.

## Interrupts

The DIO24 can use any one of six signal sources to generate interrupts. These sources are: OT0, OT1, and OT2, which are the three 8254 timer/counter outputs available on the DIO24-2 model; PC3, which is the INTRA signal from the 8255 PPI; PC0, which is the INTRB signal from the 8255 PPI; and EXT, an external interrupt you can route onto the board through the P2 I/O connector. Chapter 1 tells you how to set the jumpers on interrupt header connectors P3 and P4, and Chapter 4 provides some programming information.

# CHAPTER 4

## BOARD OPERATION AND PROGRAMMING

This chapter shows you how to program and use your DIO24 board. It provides a complete description of the I/O map and a detailed description of programming operations 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, Assembly, and BASIC, include source code to simplify your applications programming.

## Defining the I/O Map

The I/O map for the DIO24 is shown in Table 4-1 below. As shown, the board occupies 12 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. The following sections describe the register contents of each address used in the I/O map.

**Table 4-1 — DIO24 I/O Map**

| Register Description | Read Function | Write Function | Address * (Decimal) |
|---|---|---|---|
| 8255 PPI Port A | Read Port A digital input lines | Program Port A digital output lines | BA + 0 |
| 8255 PPI Port B | Read Port B digital input lines | Program Port B digital output lines | BA + 1 |
| 8255 PPI Port C | Read Port C digital input lines | Program Port C digital output lines | BA + 2 |
| 8255 PPI Control Word | Not used | Program PPI configuration | BA + 3 |
| IRQ Enable | Not used | Enable and disable interrupt generation | BA + 4 |
| Interrupt Status/Clear | Read status of interrupt | Clear interrupt | BA + 5 |
| Reserved | | | BA + 6 |
| Reserved | | | BA + 7 |
| 8254 Timer/Counter 0 | Read TC0 count value | Load TC0 count register | BA + 8 |
| 8254 Timer/Counter 1 | Read TC1 count value | Load TC1 count register | BA + 9 |
| 8254 Timer/Counter 2 | Read TC2 count value | Load TC2 count register | BA + 10 |
| 8254 Control Word | Not used | Program control register | BA + 11 |

* BA = Base Address

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

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

### BA + 1: PPI Port B — Digital I/O (Read/Write)

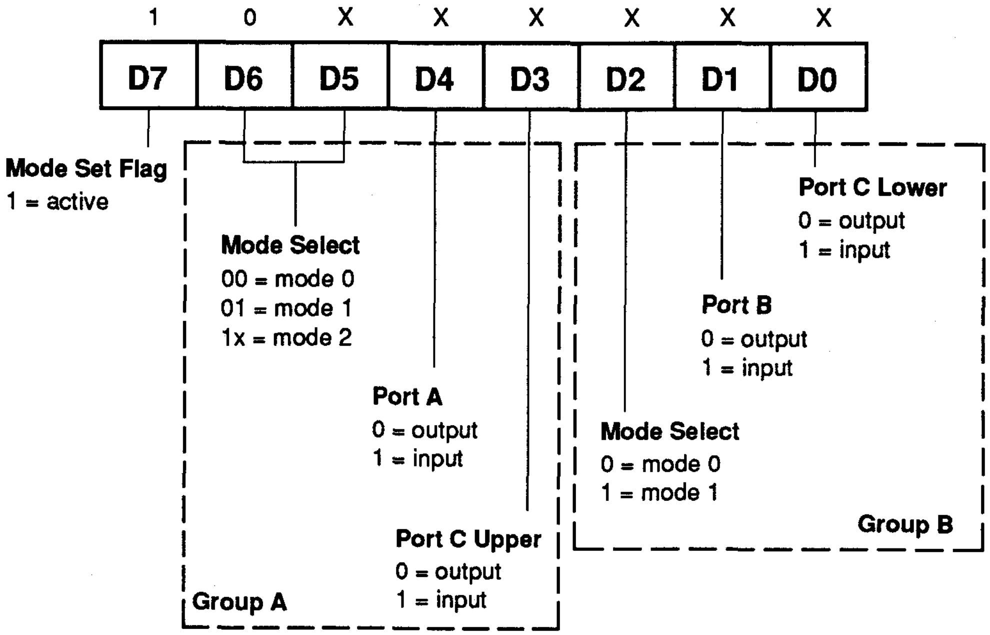
Transfers the 8-bit Port B digital input and digital output data between the board and an external device. A read transfers data from the external device, through P2, and into PPI Port B; a write transfers the written data from Port B through P2 to an external device.

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

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

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

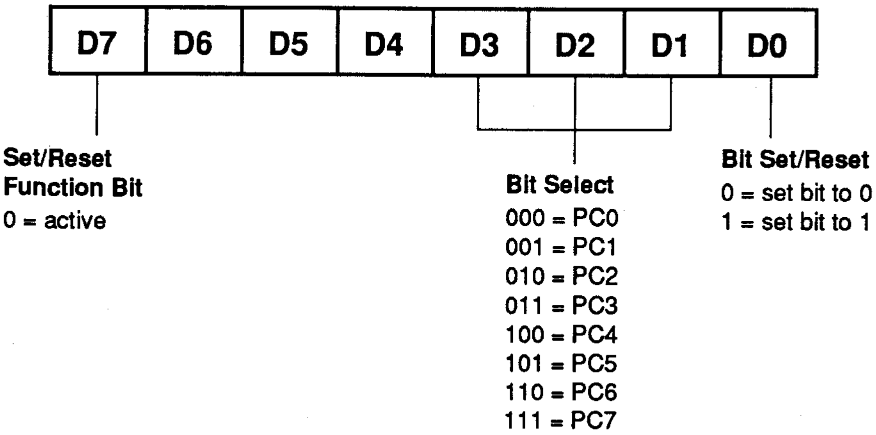
When bit 7 of this word is set to 1, a write programs the PPI configuration. Bit 6 must always be set to 0 (Mode 2 operation is not supported by the DIO24). The table below shows the control words for the 16 possible Mode 0 Port I/O combinations.



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

| Group A | | Group B | | Control Word | | |
|---|---|---|---|---|---|---|
| Port A | Port C Upper | Port B | Port C Lower | Binary | Decimal | Hex |
| Output | Output | Output | Output | 1 0 0 0 0 0 0 0 | 128 | 80 |
| Output | Output | Output | Input | 1 0 0 0 0 0 0 1 | 129 | 81 |
| Output | Output | Input | Output | 1 0 0 0 0 0 1 0 | 130 | 82 |
| Output | Output | Input | Input | 1 0 0 0 0 0 1 1 | 131 | 83 |
| Output | Input | Output | Output | 1 0 0 0 1 0 0 0 | 136 | 88 |
| Output | Input | Output | Input | 1 0 0 0 1 0 0 1 | 137 | 89 |
| Output | Input | Input | Output | 1 0 0 0 1 0 1 0 | 138 | 8A |
| Output | Input | Input | Input | 1 0 0 0 1 0 1 1 | 139 | 8B |
| Input | Output | Output | Output | 1 0 0 1 0 0 0 0 | 144 | 90 |
| Input | Output | Output | Input | 1 0 0 1 0 0 0 1 | 145 | 91 |
| Input | Output | Input | Output | 1 0 0 1 0 0 1 0 | 146 | 92 |
| Input | Output | Input | Input | 1 0 0 1 0 0 1 1 | 147 | 93 |
| Input | Input | Output | Output | 1 0 0 1 1 0 0 0 | 152 | 98 |
| Input | Input | Output | Input | 1 0 0 1 1 0 0 1 | 153 | 99 |
| Input | Input | Input | Output | 1 0 0 1 1 0 1 0 | 154 | 9A |
| Input | Input | Input | Input | 1 0 0 1 1 0 1 1 | 155 | 9B |

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



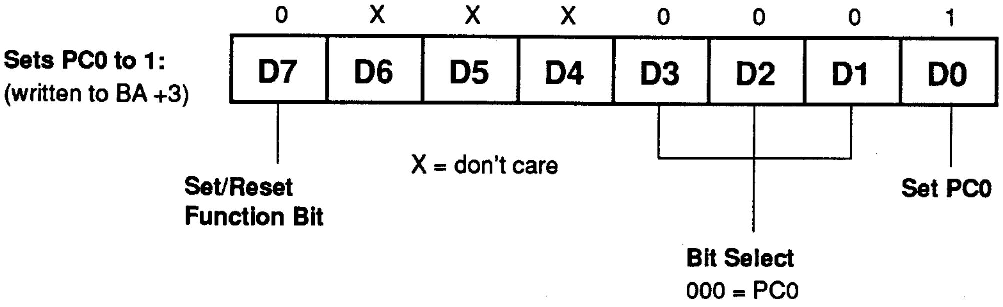
| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|

**Set/Reset Function Bit**
0 = active

**Bit Select**
000 = PC0
001 = PC1
010 = PC2
011 = PC3
100 = PC4
101 = PC5
110 = PC6
111 = PC7

**Bit Set/Reset**
0 = set bit to 0
1 = set bit to 1

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:



**Sets PC0 to 1:**
(written to BA +3)

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

X = don't care

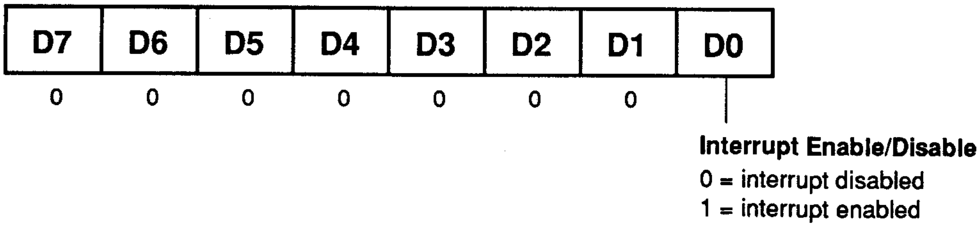
**Set/Reset Function Bit**

**Bit Select**
000 = PC0

**Set PC0**

**BA + 4: IRQ Enable (Write Only)**

Enables and disables interrupt generation. Writing a "1" enables interrupt generation; writing a "0" disables interrupt generation.

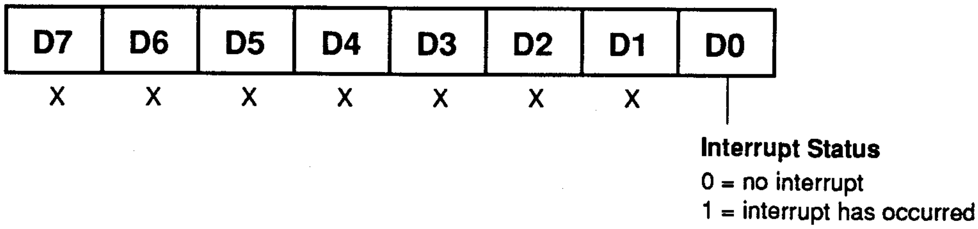


| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

**Interrupt Enable/Disable**
0 = interrupt disabled
1 = interrupt enabled

**BA + 5: Interrupt Status/Clear (Read/Write)**

A read shows the status of the interrupt (bit 0 only) as defined below. A write clears the interrupt (data written is irrelevant). Each time the interrupt status bit goes high, a write should follow to clear the bit.



| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| X | X | X | X | X | X | X | |

**Interrupt Status**
0 = no interrupt
1 = interrupt has occurred

**BA + 6: Reserved**

**BA + 7: Reserved**

**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.

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

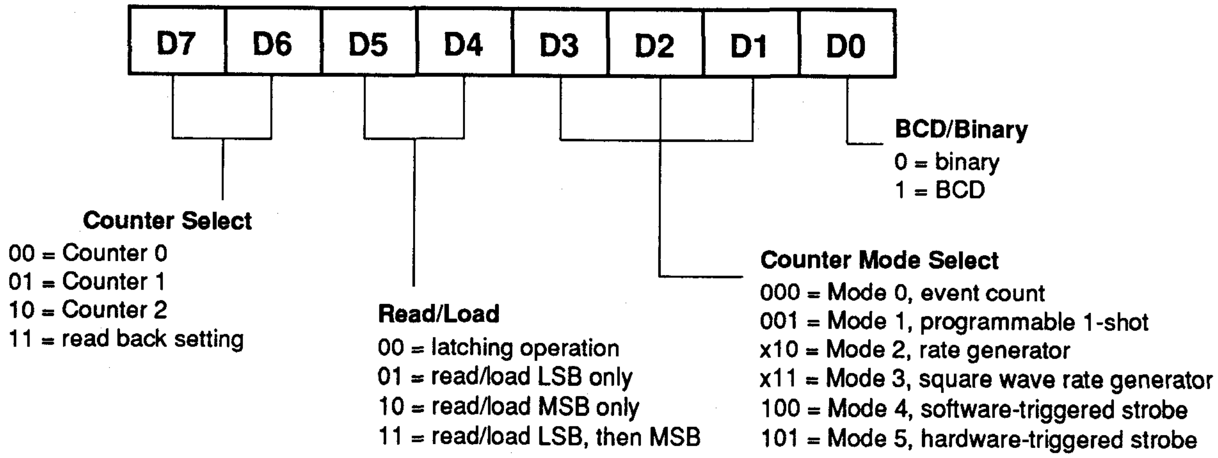
A read shows the count in the counter, and a write loads the counter with a new value. Counting begins as soon as the count is loaded.

**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.

**BA + 11: 8254 Control Word (Write Only)**

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



**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
x10 = Mode 2, rate generator
x11 = Mode 3, square wave rate generator
100 = Mode 4, software-triggered strobe
101 = Mode 5, hardware-triggered strobe

## Programming the DIO24

This section gives you some general information about programming and the DIO24 board, and then walks you through the major DIO24 programming functions. These descriptions will help you as you use the example programs included with the board. All of the program descriptions in this section use decimal values unless otherwise specified.

The DIO24 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<br>in al, dx | mov dx, Address<br>mov al, Data<br>out dx, al |

In addition to being able to read/write the I/O ports on the DIO24, 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 | %<br>a = b % c | /<br>a = b / c | &<br>a = b & c | \|<br>a = b \| c |
| Pascal | MOD<br>a := b MOD c | DIV<br>a := b DIV c | AND<br>a := b AND c | OR<br>a := b OR c |
| BASIC | MOD<br>a = b MOD c | \ (backslash)<br>a = b \ c | AND<br>a = b AND c | OR<br>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 DIO24!**

### 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 DIO24 board functions.

### Initializing the 8255 PPI

Before you can operate the DIO24, the 8255 must be initialized. This step must be executed every time you start up, reset, or reboot your computer.

The 8255 is initialized by writing the appropriate control word to I/O port BA + 3. The contents of your control word will vary, depending on how you want to configure your I/O lines. Use the control word description in the previous I/O map section to help you program the right value. Remember that the DIO24 cannot use Mode 2. In the example below, a decimal value of 128 sets up the 8255 so that all I/O lines are Mode 0 outputs.

| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|
| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

### Digital I/O Operations

Once the 8255 is initialized, you can use the digital I/O lines to control or monitor external devices.

### Timer/Counters

An 8254 programmable interval timer provides three 16-bit, 8-MHz timer/counters for timing and counting functions such as frequency measurement, event counting, and interrupts. All three timer/counters are cascaded at the factory. Figure 4-1 shows the timer/counter circuitry.

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

One of two clock sources, the on-board 8-MHz crystal or an external clock routed through on-board I/O connector P6 can be selected as the clock input to each timer/counter. In addition, the timer/counters can be cascaded by connecting TC0's output to TC1's clock input and TC1's output to TC2's clock input. The diagram shows how these clock sources are connected to the timer/counters.

An external gate source can be connected to each timer/counter through P6. When a gate is disconnected, an on-board pull-up resistor automatically pulls the gate high, enabling the timer/counter.

The output from each timer/counter is available at P6, where it can be used for interrupt generation or for counting functions.

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

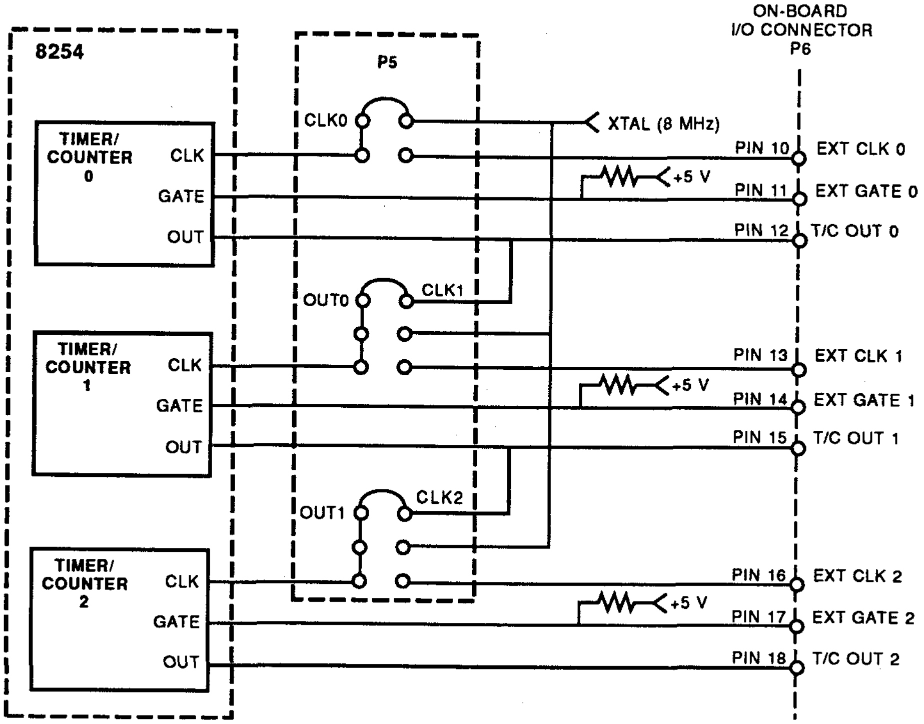


Fig. 4-1 — 8254 Timer/Counter Circuit Block Diagram

**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.

## 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 DIO24 board can interrupt the processor when one of the six interrupt sources is enabled. 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 DIO24 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 will 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 DIO24), 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.

**- 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 DIO24 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 DIO24 by writing any value to BA + 5.
- 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 + 5, 0);     /* Clear DIO24 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 + 5] := 0;      { Clear DIO24 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 DIO24 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 DIO24 and forgetting to issue the EOI command to the 8259 interrupt controller before exiting the ISR.

## Example Programs

Included with the DIO24 is a set of example programs that demonstrate the use of many of the board's features. These examples are in written in C, Pascal, Assembly, and BASIC. Also included is an easy-to-use menu-driven diagnostics program, 24DIAG, which is especially helpful when you are first checking out your board after installation.

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 DIO24 board.

**Digital I/O:**

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

**Timer/Counters:**

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

### BASIC Programs

These programs include both source code files and executable files so that you can run them on your DIO24. All of the executable programs are set up to look for the board at a base address (BA) of 300 hex (768 decimal). If you change the base address of the board, you must also change the BA in your programs.

**Digital I/O:**

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

**Timer/Counters:**

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

# APPENDIX A

## DIO24 SPECIFICATIONS

# DIO24 Characteristics Typical @ 25° C

### Interface

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

| | |
|---|---|
| **Digital I/O** | **CMOS 82C55** |
| Opto-22 compatible | |
| Number of lines | 24 |
| 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 |
| High-level output current, Isource | CMOS buffer: -12 mA, max; TTL buffer: -16 mA, max |
| Low-level output current, Isink | CMOS buffer: 24 mA, max; TTL buffer: 64 mA, max |
| Input load current | ±10 µA |
| Input capacitance | 10 pF |
| Input capacitance, C(IN)@F=1MHz | 10 pF |
| Output capacitance, C(OUT)<@F=1MHz | 20 pF |

| | |
|---|---|
| **Timer/Counters** | **CMOS 82C54** |
| Three 16-bit down counters | |
| 6 programmable operating modes | |
| Counter input source | External clock (8 MHz, max) or on-board 8 MHz clock |
| Counter outputs | Available externally; used as PC interrupts |
| Counter gate source | External gate or always enabled |

### Current Requirements

194 mA @ +5 volts

### Connectors

P2 — 50-pin right angle shrouded box header
P6 — 20-pin box header

### Size

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

# APPENDIX B

## I/O CONNECTOR PIN ASSIGNMENTS

## I/O Connector P2:

| | | | |
|---|---|---|---|
| PC0 | 1 | 2 | EXTINT |
| PC1 | 3 | 4 | DIGITAL GND |
| PC2 | 5 | 6 | DIGITAL GND |
| PC3 | 7 | 8 | DIGITAL GND |
| PC4 | 9 | 10 | DIGITAL GND |
| PC5 | 11 | 12 | DIGITAL GND |
| PC6 | 13 | 14 | DIGITAL GND |
| PC7 | 15 | 16 | DIGITAL GND |
| PB0 | 17 | 18 | DIGITAL GND |
| PB1 | 19 | 20 | DIGITAL GND |
| PB2 | 21 | 22 | DIGITAL GND |
| PB3 | 23 | 24 | DIGITAL GND |
| PB4 | 25 | 26 | DIGITAL GND |
| PB5 | 27 | 28 | DIGITAL GND |
| PB6 | 29 | 30 | DIGITAL GND |
| PB7 | 31 | 32 | DIGITAL GND |
| PA0 | 33 | 34 | DIGITAL GND |
| PA1 | 35 | 36 | DIGITAL GND |
| PA2 | 37 | 38 | DIGITAL GND |
| PA3 | 39 | 40 | DIGITAL GND |
| PA4 | 41 | 42 | DIGITAL GND |
| PA5 | 43 | 44 | DIGITAL GND |
| PA6 | 45 | 46 | DIGITAL GND |
| PA7 | 47 | 48 | DIGITAL GND |
| +5 VOLTS | 49 | 50 | DIGITAL GND |

## On-board Connector P6:

| | | | |
|---|---|---|---|
| PC0 | 1 | 2 | PC1 |
| PC2 | 3 | 4 | PC3 |
| PC4 | 5 | 6 | PC5 |
| PC6 | 7 | 8 | PC7 |
| DIGITAL GND | 9 | 10 | EXT CLK 0 |
| EXT GATE 0 | 11 | 12 | T/C OUT 0 |
| EXT CLK 1 | 13 | 14 | EXT GATE 1 |
| T/C OUT 1 | 15 | 16 | EXT CLK 2 |
| EXT GATE 2 | 17 | 18 | T/C OUT 2 |
| DIGITAL GND | 19 | 20 | DIGITAL GND |

# APPENDIX C

## COMPONENT DATA SHEETS

# Intel 82C55A Programmable Peripheral Interface
# Data Sheet Reprint

intel®

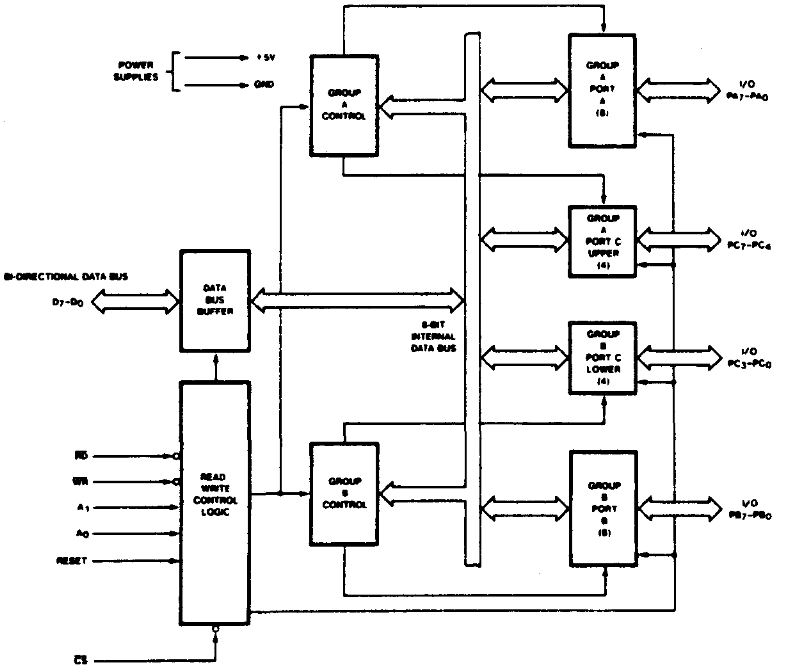
# 82C55A
# CHMOS PROGRAMMABLE PERIPHERAL INTERFACE

- **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.

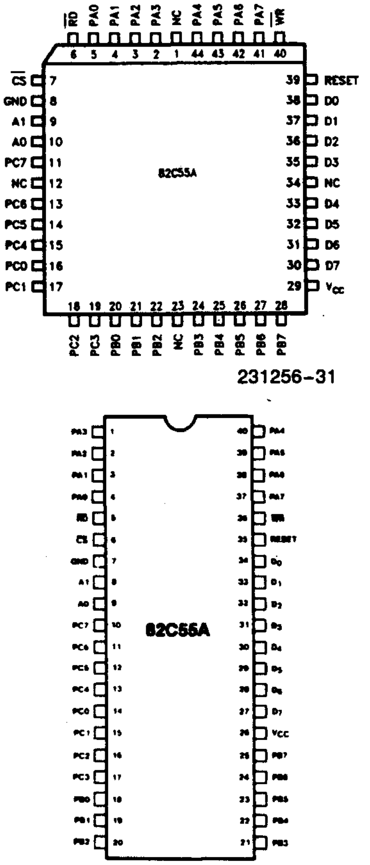
In MODE 0, each group of 12 I/O pins may be programmed in sets of 4 and 8 to be inputs or outputs. In MODE 1, each group may be programmed to have 8 lines of input or output. 3 of the remaining 4 pins are used for handshaking and interrupt control signals. MODE 2 is a strobed bi-directional bus configuration.

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**



231256-31

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

<table>
<tr><th rowspan="2">Symbol</th><th colspan="2">Pin Number</th><th rowspan="2">Type</th><th colspan="6" rowspan="2">Name and Function</th></tr>
<tr><th>Dip</th><th>PLCC</th></tr>
<tr><td>$PA_{3-0}$</td><td>1–4</td><td>2–5</td><td>I/O</td><td colspan="6"><b>PORT A, PINS 0–3:</b> Lower nibble of an 8-bit data output latch/buffer and an 8-bit data input latch.</td></tr>
<tr><td>$\overline{RD}$</td><td>5</td><td>6</td><td>I</td><td colspan="6"><b>READ CONTROL:</b> This input is low during CPU read operations.</td></tr>
<tr><td>$\overline{CS}$</td><td>6</td><td>7</td><td>I</td><td colspan="6"><b>CHIP SELECT:</b> A low on this input enables the 82C55A to respond to $\overline{RD}$ and $\overline{WR}$ signals. $\overline{RD}$ and WR are ignored otherwise.</td></tr>
<tr><td>GND</td><td>7</td><td>8</td><td></td><td colspan="6"><b>System Ground</b></td></tr>
<tr><td rowspan="14">$A_{1-0}$</td><td rowspan="14">8–9</td><td rowspan="14">9–10</td><td rowspan="14">I</td><td colspan="6"><b>ADDRESS:</b> These input signals, in conjunction $\overline{RD}$ and $\overline{WR}$, control the selection of one of the three ports or the control word registers.</td></tr>
<tr><td>$A_1$</td><td>$A_0$</td><td>$\overline{RD}$</td><td>$\overline{WR}$</td><td>$\overline{CS}$</td><td><b>Input Operation (Read)</b></td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>Port A - Data Bus</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>Port B - Data Bus</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>Port C - Data Bus</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>Control Word - Data Bus</td></tr>
<tr><td colspan="6"><b>Output Operation (Write)</b></td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>Data Bus - Port A</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>Data Bus - Port B</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>Data Bus - Port C</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>Data Bus - Control</td></tr>
<tr><td colspan="6"><b>Disable Function</b></td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>1</td><td>Data Bus - 3 - State</td></tr>
<tr><td>X</td><td>X</td><td>1</td><td>1</td><td>0</td><td>Data Bus - 3 - State</td></tr>
<tr><td>$PC_{7-4}$</td><td>10–13</td><td>11,13–15</td><td>I/O</td><td colspan="6"><b>PORT C, PINS 4–7:</b> 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.</td></tr>
<tr><td>$PC_{0-3}$</td><td>14–17</td><td>16–19</td><td>I/O</td><td colspan="6"><b>PORT C, PINS 0–3:</b> Lower nibble of Port C.</td></tr>
<tr><td>$PB_{0-7}$</td><td>18–25</td><td>20–22,<br>24–28</td><td>I/O</td><td colspan="6"><b>PORT B, PINS 0–7:</b> An 8-bit data output latch/buffer and an 8-bit data input buffer.</td></tr>
<tr><td>$V_{CC}$</td><td>26</td><td>29</td><td></td><td colspan="6"><b>SYSTEM POWER:</b> + 5V Power Supply.</td></tr>
<tr><td>$D_{7-0}$</td><td>27–34</td><td>30–33,<br>35–38</td><td>I/O</td><td colspan="6"><b>DATA BUS:</b> Bi-directional, tri-state data bus lines, connected to system data bus.</td></tr>
<tr><td>RESET</td><td>35</td><td>39</td><td>I</td><td colspan="6"><b>RESET:</b> A high on this input clears the control register and all ports are set to the input mode.</td></tr>
<tr><td>$\overline{WR}$</td><td>36</td><td>40</td><td>I</td><td colspan="6"><b>WRITE CONTROL:</b> This input is low during CPU write operations.</td></tr>
<tr><td>$PA_{7-4}$</td><td>37–40</td><td>41–44</td><td>I/O</td><td colspan="6"><b>PORT A, PINS 4–7:</b> Upper nibble of an 8-bit data output latch/buffer and an 8-bit data input latch.</td></tr>
<tr><td>NC</td><td></td><td>1, 12,<br>23, 34</td><td></td><td colspan="6">No Connect</td></tr>
</table>

## 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

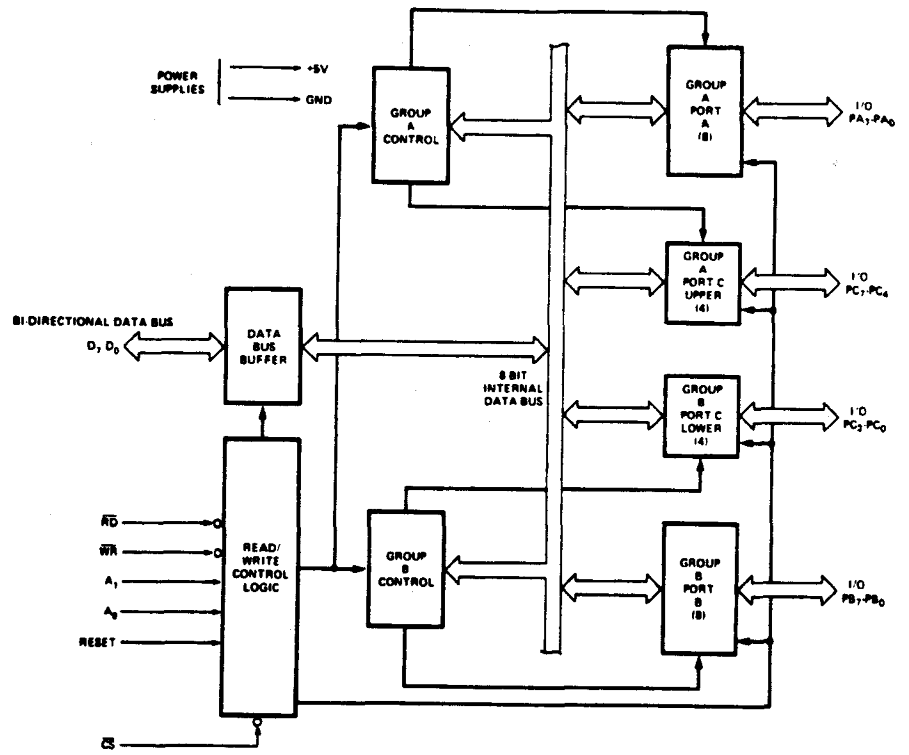
The 82C55A contains three 8-bit ports (A, B, and C). All can be configured in a wide variety of functional characteristics by the system software but each has its own special features or "personality" to further enhance the power and flexibility of the 82C55A.

**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.

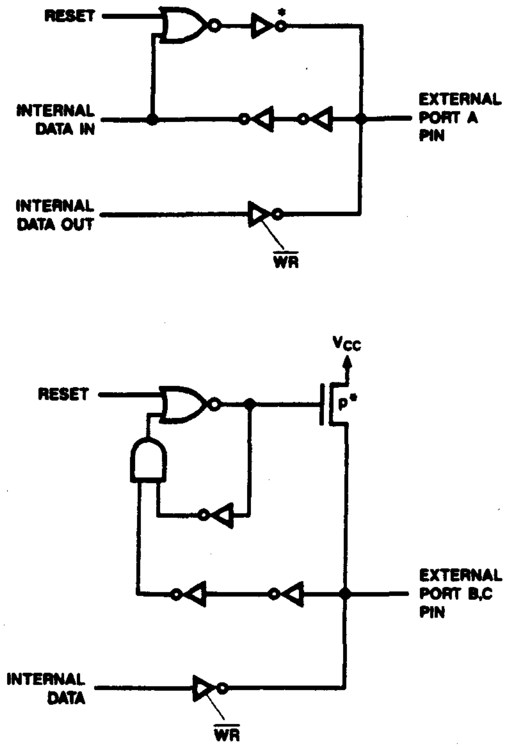
**Port C.** One 8-bit data output latch/buffer and one 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the control signal outputs and status signal inputs in conjunction with 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.



231256-3

**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.

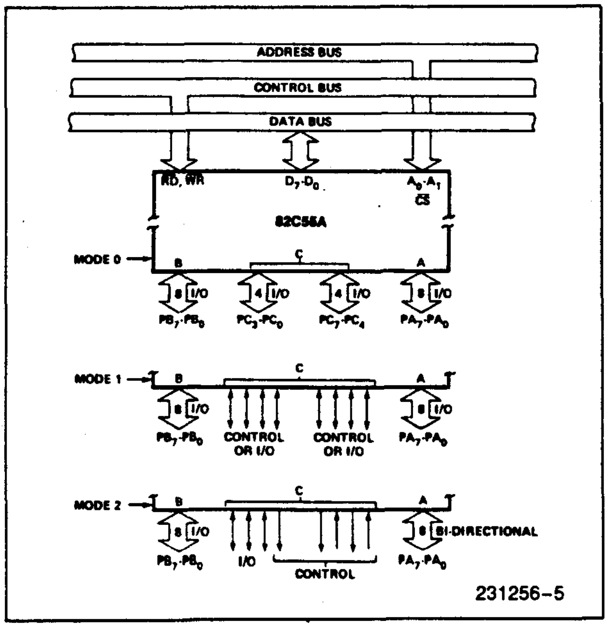


**Figure 5. Basic Mode Definitions and Bus Interface**

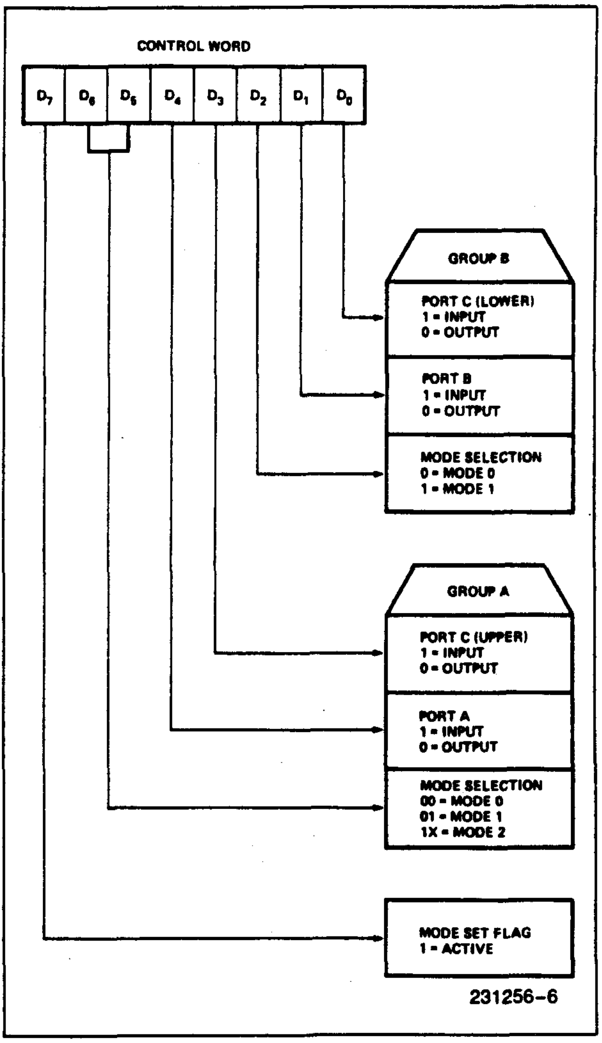


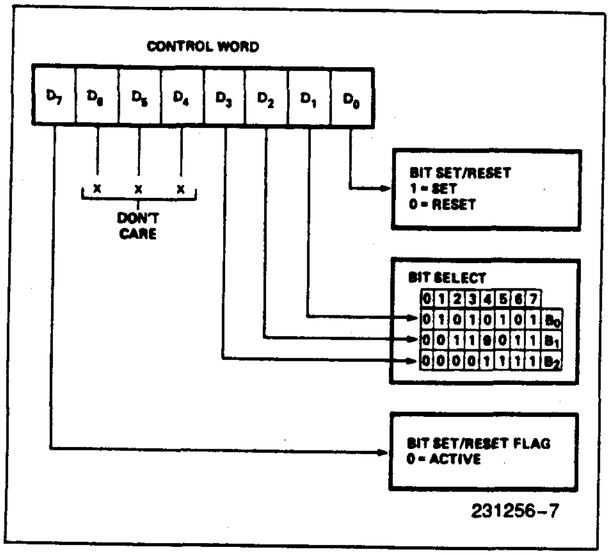
**Figure 6. Mode Definition Format**

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.



| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | $B_0$ |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | $B_1$ |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | $B_2$ |

**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.

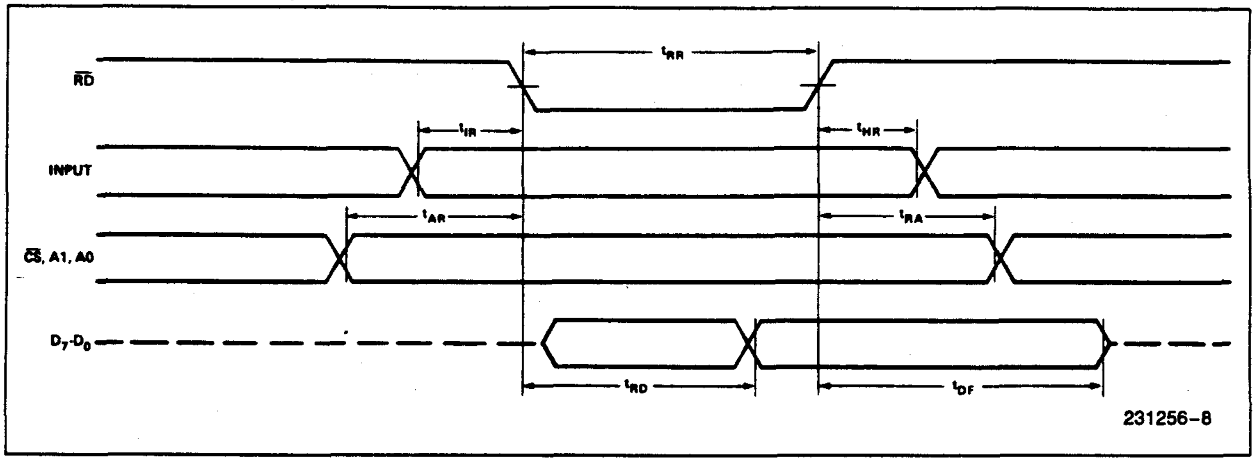
## Operating Modes

**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:

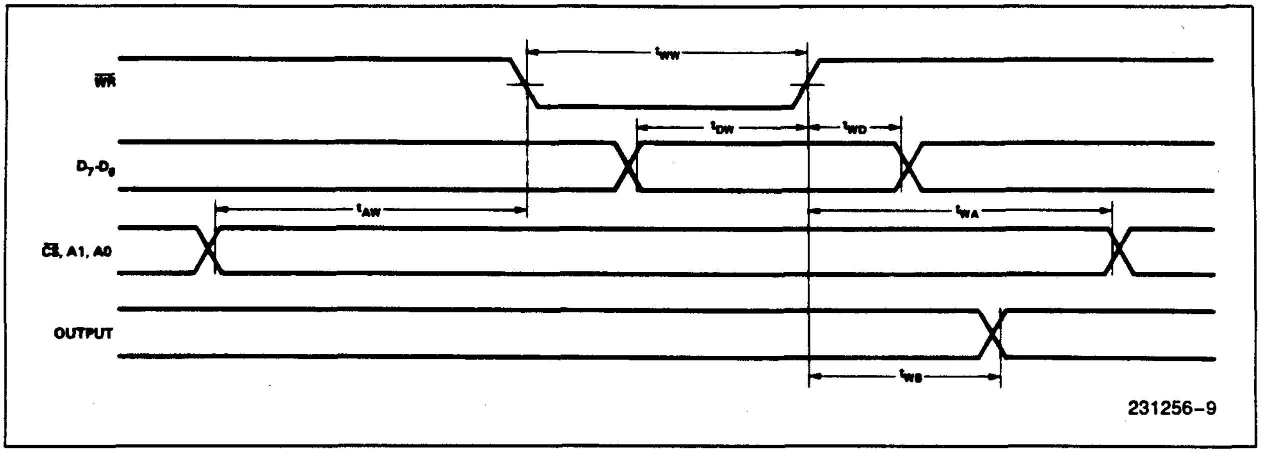
- Two 8-bit ports and two 4-bit ports.
- Any port can be input or output.
- Outputs are latched.
- Inputs are not latched.
- 16 different Input/Output configurations are possible in this Mode.

### MODE 0 (BASIC INPUT)



231256-8

### MODE 0 (BASIC OUTPUT)

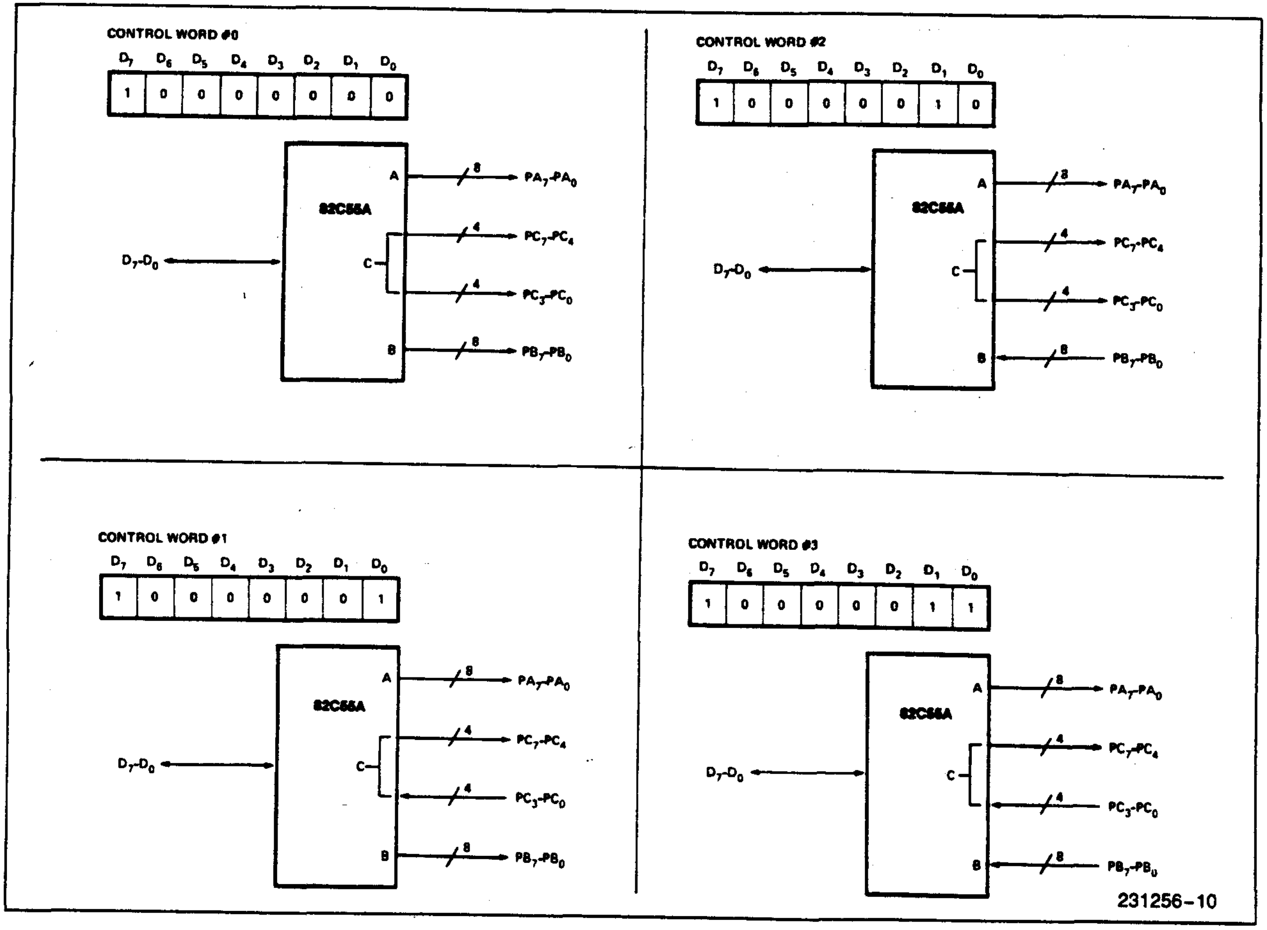


231256-9

## 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

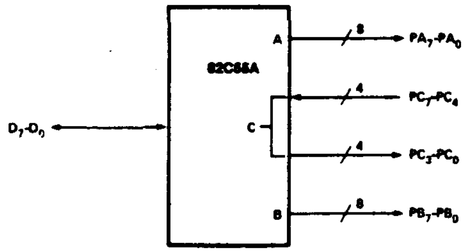


231256-10

## MODE 0 Configurations (Continued)

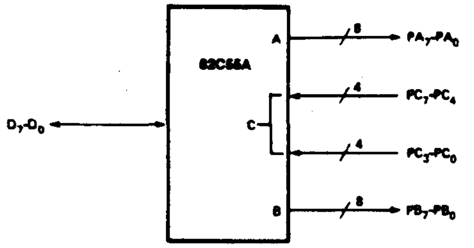
CONTROL WORD #4

| $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |



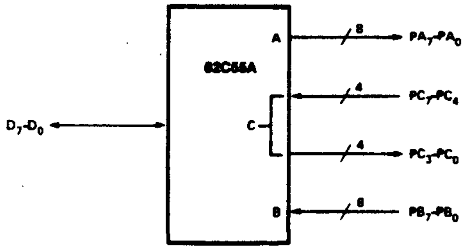
CONTROL WORD #5

| $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |



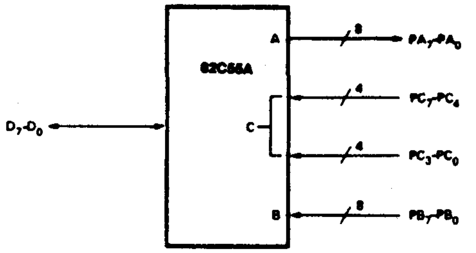
CONTROL WORD #6

| $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |



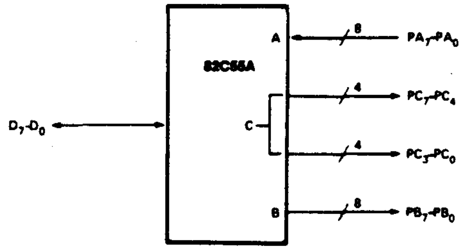
CONTROL WORD #7

| $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |



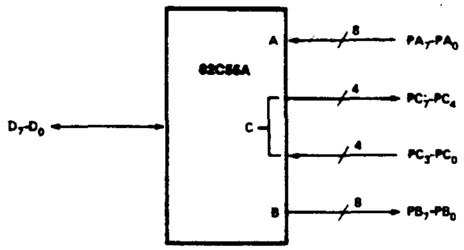
CONTROL WORD #8

| $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |



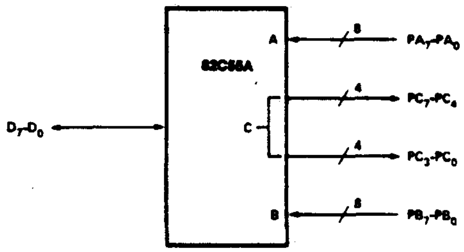
CONTROL WORD #9

| $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |



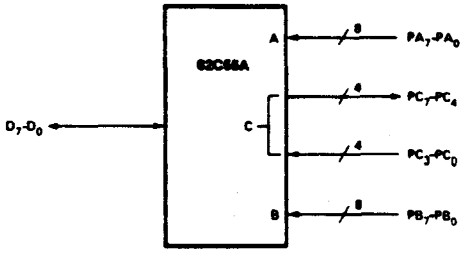
CONTROL WORD #10

| $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |



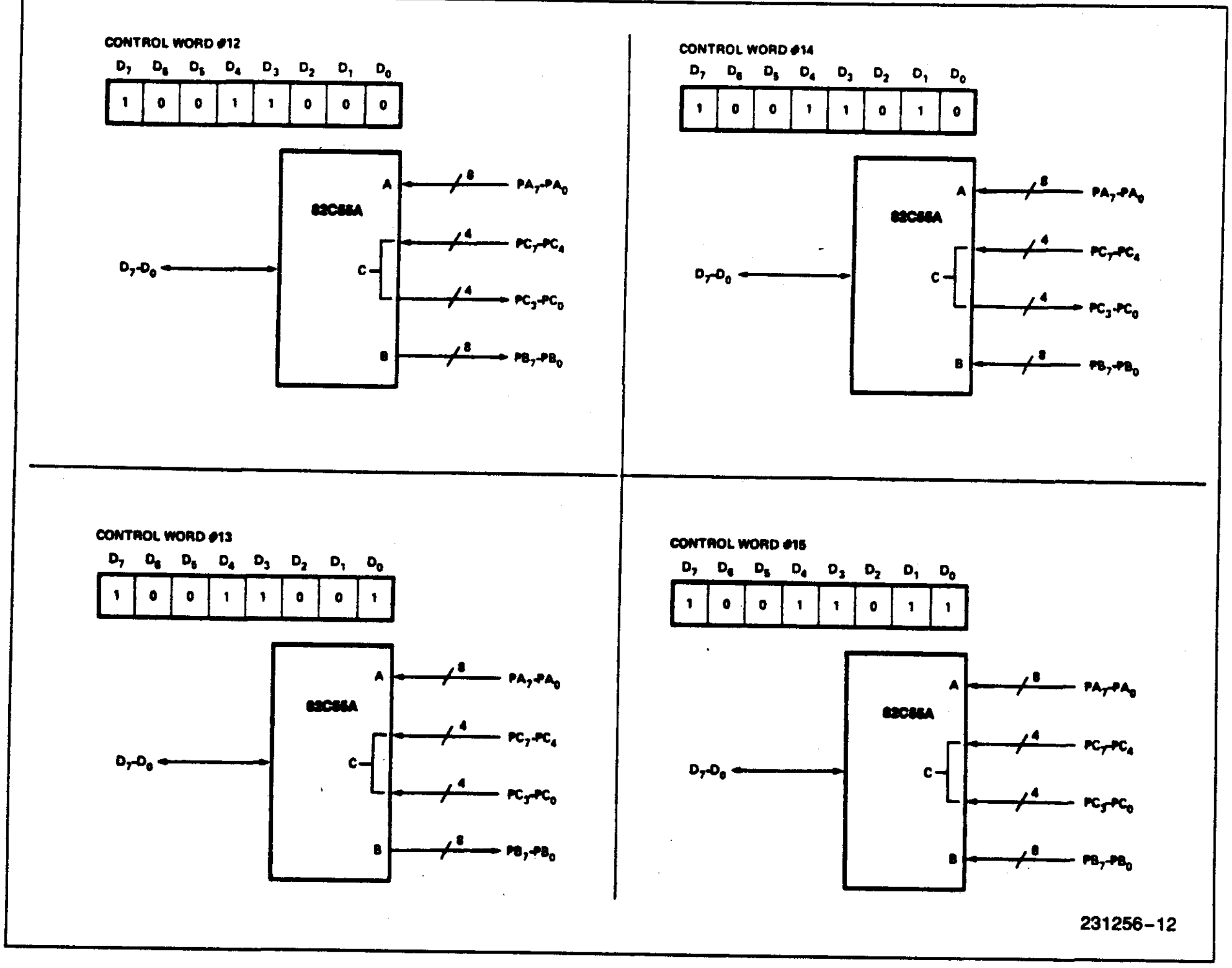
CONTROL WORD #11

| $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |



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

**$\overline{\text{STB}}$ (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 the data has been loaded into the input latch; in essence, an acknowledgement. IBF is set by $\overline{\text{STB}}$ input being low and is reset by the rising edge of the $\overline{\text{RD}}$ 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 $\overline{\text{STB}}$ is a "one", IBF is a "one" and INTE is a "one". It is reset by the falling edge of $\overline{\text{RD}}$. 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 $PC_2$.

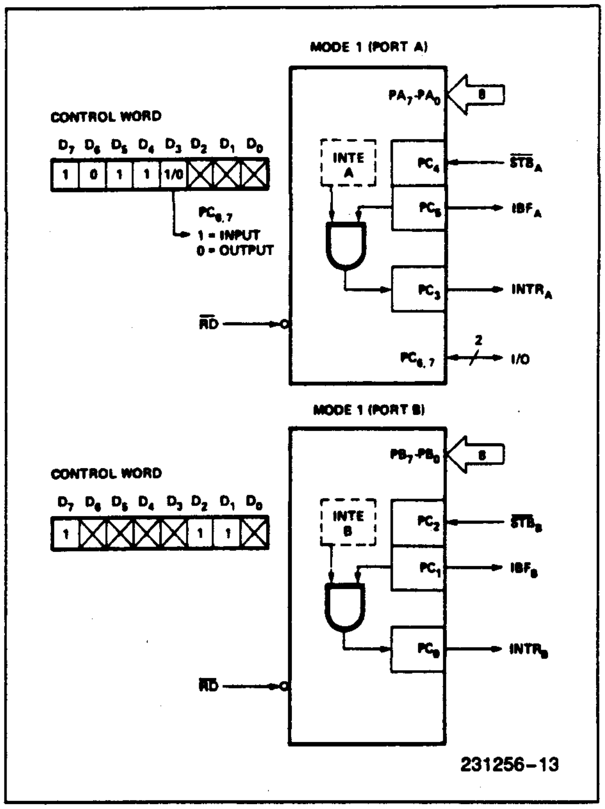


**Figure 8. MODE 1 Input**

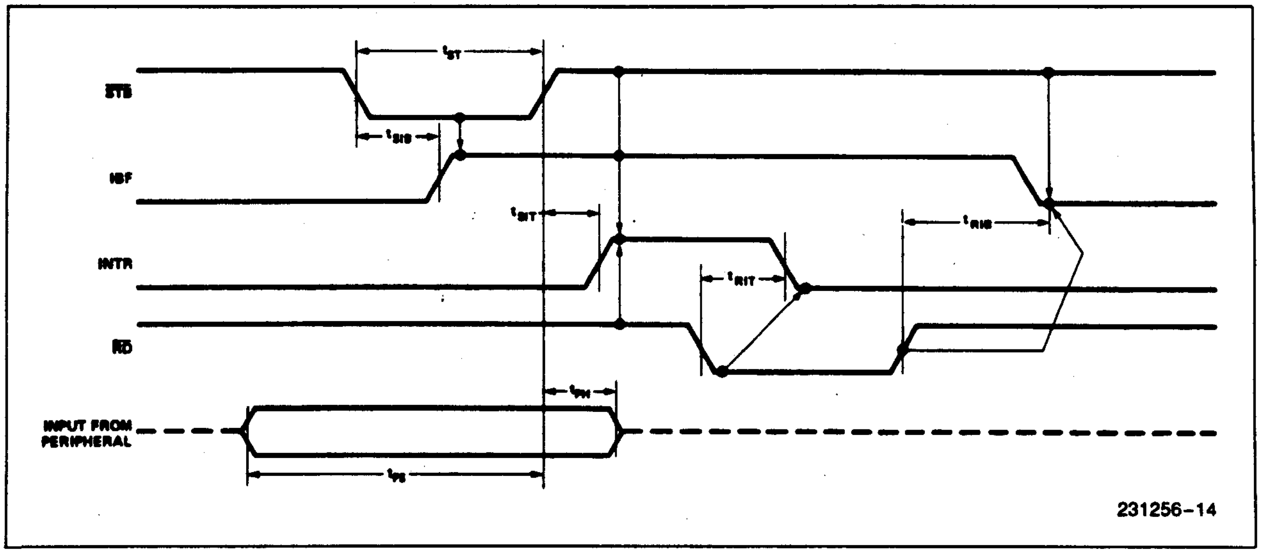


**Figure 9. MODE 1 (Strobed Input)**

### Output Control Signal Definition

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

**$\overline{ACK}$ (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.

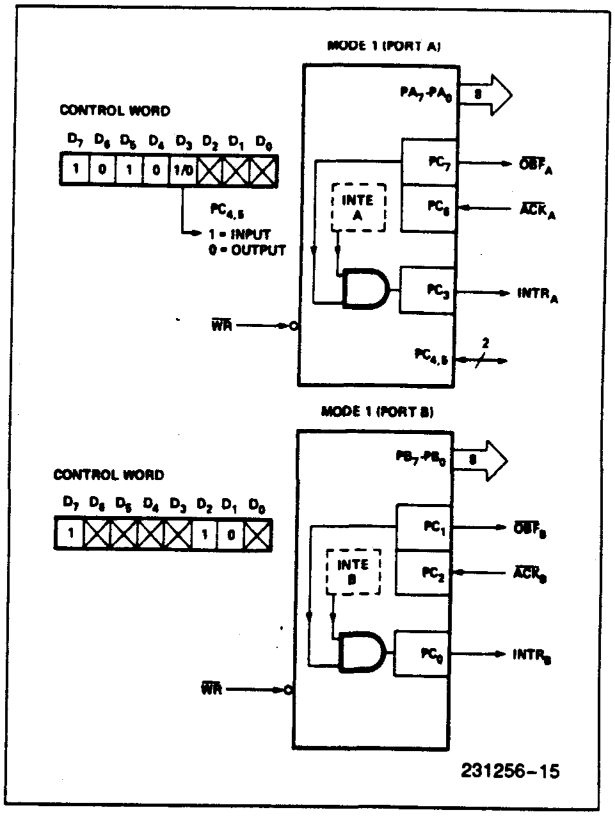
**INTR (Interrupt Request).** A "high" on this output can be used to interrupt the CPU when an output device has accepted data transmitted by the CPU. INTR is set when $\overline{ACK}$ is a "one", $\overline{OBF}$ is a "one" and INTE is a "one". It is reset by the falling edge of $\overline{WR}$.

**INTE A**

Controlled by bit set/reset of $PC_6$.

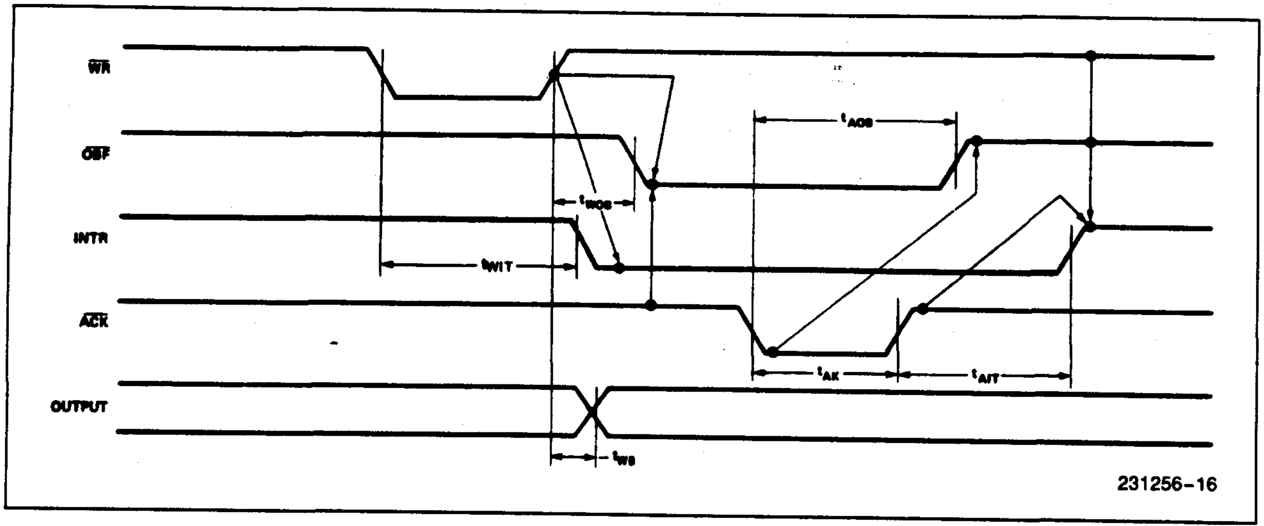
**INTE B**

Controlled by bit set/reset of $PC_2$.



231256–15

**Figure 10. MODE 1 Output**



231256–16

**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.

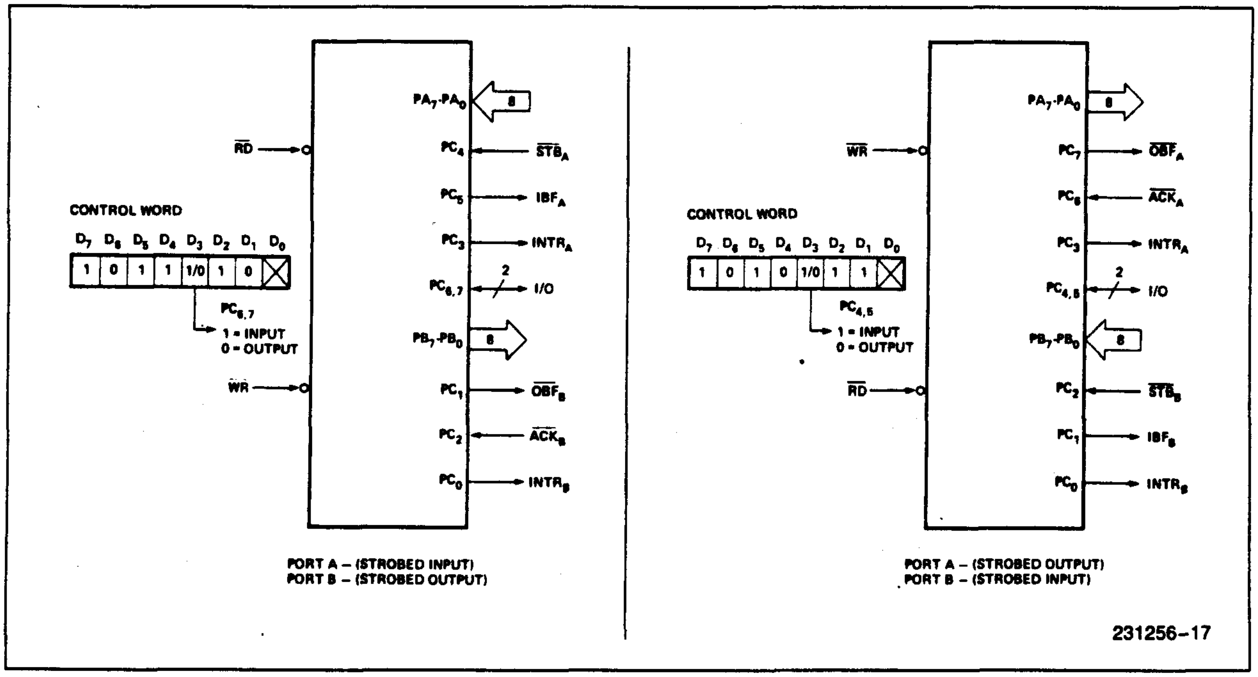


**Figure 12. Combinations of MODE 1**

### 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 $PC_4$.

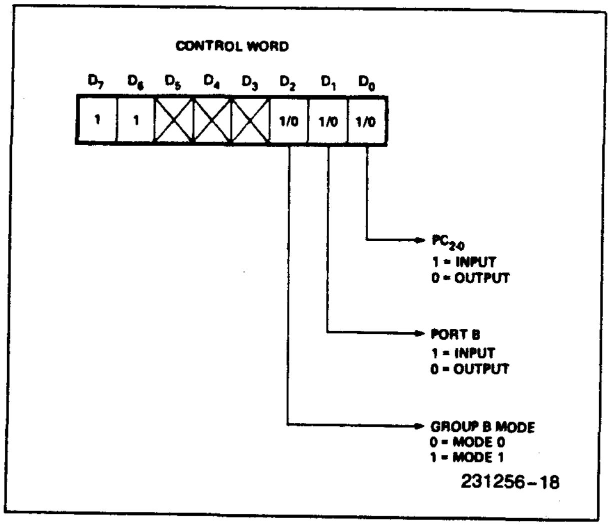


**Figure 13. MODE Control Word**

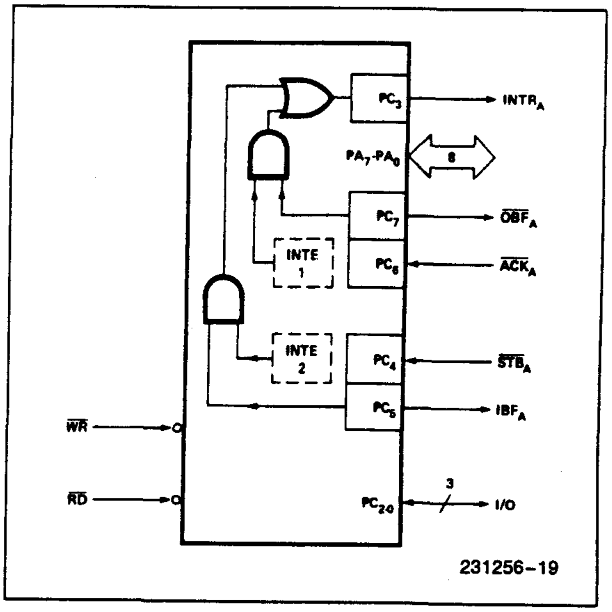


**Figure 14. MODE 2**

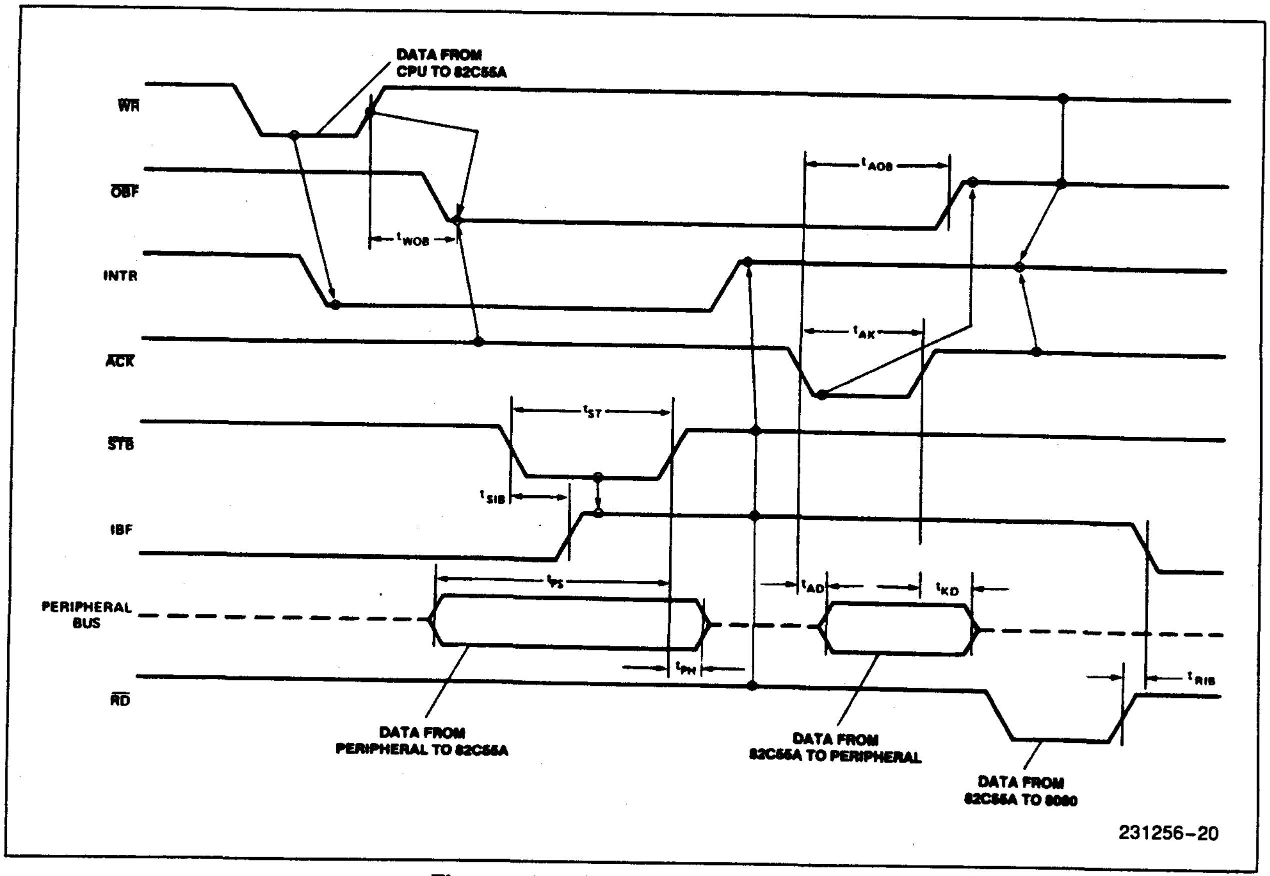


**Figure 15. MODE 2 (Bidirectional)**

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

### 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$_B$ | $\overline{\text{OBF}}_B$ | I/O |
| PC$_2$ | IN | OUT | $\overline{\text{STB}}_B$ | $\overline{\text{ACK}}_B$ | I/O |
| PC$_3$ | IN | OUT | INTR$_A$ | INTR$_A$ | INTR$_A$ |
| PC$_4$ | IN | OUT | $\overline{\text{STB}}_A$ | I/O | $\overline{\text{STB}}_A$ |
| PC$_5$ | IN | OUT | IBF$_A$ | I/O | IBF$_A$ |
| PC$_6$ | IN | OUT | I/O | $\overline{\text{ACK}}_A$ | $\overline{\text{ACK}}_A$ |
| PC$_7$ | IN | OUT | I/O | $\overline{\text{OBF}}_A$ | $\overline{\text{OBF}}_A$ |

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 $\overline{\text{ACK}}$ and $\overline{\text{STB}}$ lines, will be placed on the data bus. In place of the $\overline{\text{ACK}}$ and $\overline{\text{STB}}$ 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 $\overline{\text{OBF}}$) can be written, or an interrupt enable flag can be either set or reset. Port C lines programmed as inputs, including $\overline{\text{ACK}}$ and $\overline{\text{STB}}$ 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 $\overline{\text{ACK}}$ and $\overline{\text{STB}}$ 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.

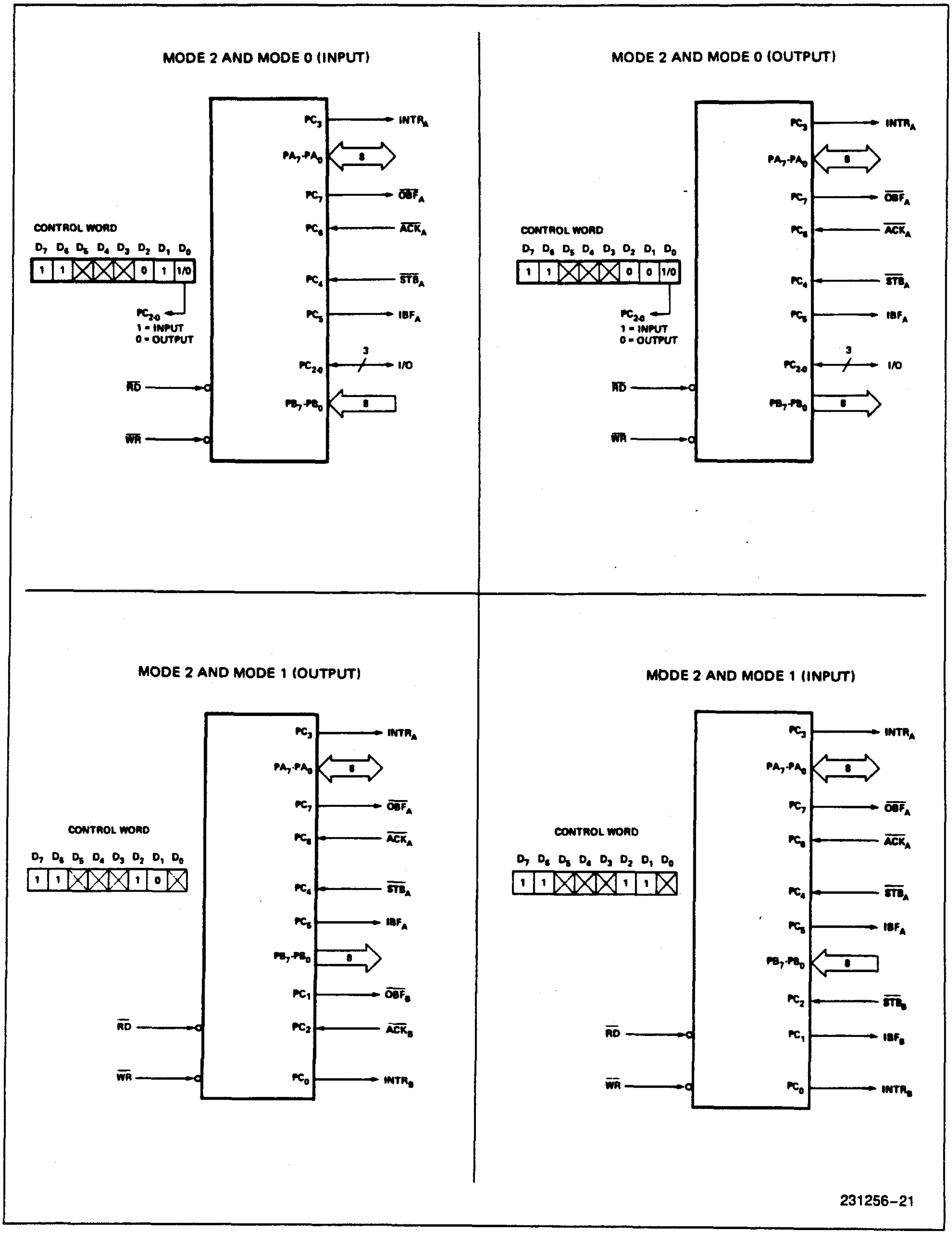


**Figure 16. MODE 1/4 Combinations**

## 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<br>$V_{CC}$ − 0.4 | | V<br>V | $I_{OH}$ = −2.5 mA<br>$I_{OH}$ = −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<br>$R_{ext}$ = 500Ω<br>$V_{ext}$ = 1.7V |
| $I_{PHL}$ | Port Hold Low Leakage Current | +50 | +300 | μA | $V_{OUT}$ = 1.0V<br>Port A only |
| $I_{PHH}$ | Port Hold High Leakage Current | −50 | −300 | μA | $V_{OUT}$ = 3.0V<br>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<br>$V_{IN}$ = $V_{CC}$ or GND<br>Port Conditions<br>If I/P = Open/High<br>O/P = Open Only<br>With Data Bus = High/Low<br>$\overline{CS}$ = High<br>Reset = Low<br>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<br>$f_c$ = 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 ±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 $\overline{\text{RD}}$ ↓ | 0 | | ns | |
| $t_{RA}$ | Address Hold Time After $\overline{\text{RD}}$ ↑ | 0 | | ns | |
| $t_{RR}$ | $\overline{\text{RD}}$ Pulse Width | 150 | | ns | |
| $t_{RD}$ | Data Delay from $\overline{\text{RD}}$ ↓ | | 120 | ns | |
| $t_{DF}$ | $\overline{\text{RD}}$ ↑ to Data Floating | 10 | 75 | ns | |
| $t_{RV}$ | Recovery Time between $\overline{\text{RD}}$/$\overline{\text{WR}}$ | 200 | | ns | |

### WRITE CYCLE

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

## OTHER TIMINGS

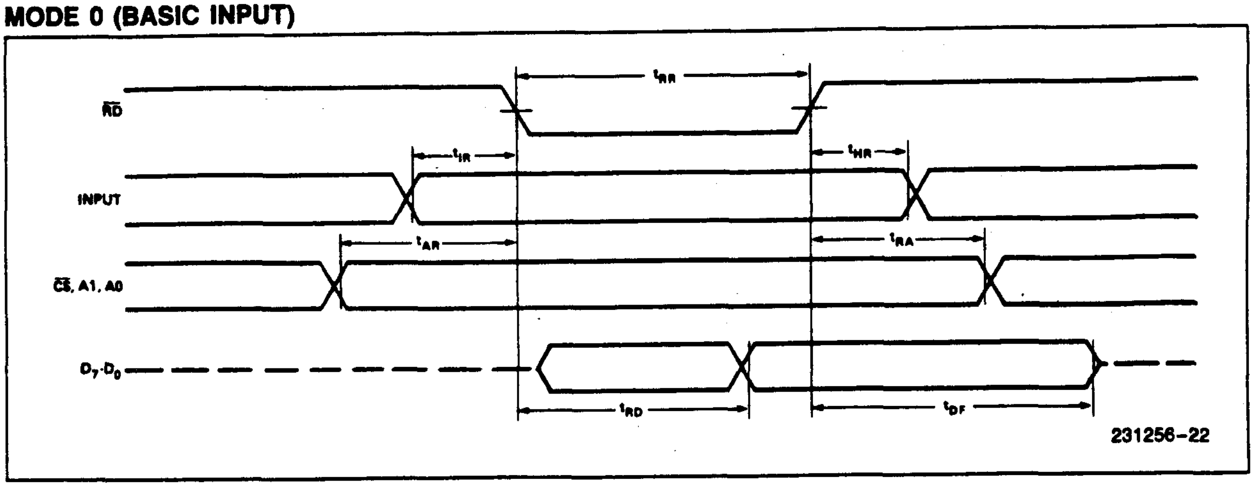
| Symbol | Parameter | 82C55A-2 | | Units Conditions | Test |
|---|---|---|---|---|---|
| | | Min | Max | | |
| $t_{WB}$ | $\overline{\text{WR}}$ = 1 to Output | | 350 | ns | |
| $t_{IR}$ | Peripheral Data Before $\overline{\text{RD}}$ | 0 | | ns | |
| $t_{HR}$ | Peripheral Data After $\overline{\text{RD}}$ | 0 | | ns | |
| $t_{AK}$ | $\overline{\text{ACK}}$ Pulse Width | 200 | | ns | |
| $t_{ST}$ | $\overline{\text{STB}}$ Pulse Width | 100 | | ns | |
| $t_{PS}$ | Per. Data Before $\overline{\text{STB}}$ High | 20 | | ns | |
| $t_{PH}$ | Per. Data After $\overline{\text{STB}}$High | 50 | | ns | |
| $t_{AD}$ | $\overline{\text{ACK}}$ = 0 to Output | | 175 | ns | |
| $t_{KD}$ | $\overline{\text{ACK}}$ = 1 to Output Float | 20 | 250 | ns | |
| $t_{WOB}$ | $\overline{\text{WR}}$ = 1 to $\overline{\text{OBF}}$ = O | | 150 | ns | |
| $t_{AOB}$ | $\overline{\text{ACK}}$ = 0 to $\overline{\text{OBF}}$ = 1 | | 150 | ns | |
| $t_{SIB}$ | $\overline{\text{STB}}$ = 0 to IBF = 1 | | 150 | ns | |
| $t_{RIB}$ | $\overline{\text{RD}}$ = 1 to IBF = 0 | | 150 | ns | |
| $t_{RIT}$ | $\overline{\text{RD}}$ = 0 to INTR = 0 | | 200 | ns | |
| $t_{SIT}$ | $\overline{\text{STB}}$ = 1 to INTR = 1 | | 150 | ns | |
| $t_{AIT}$ | $\overline{\text{ACK}}$ = 1 to INTR = 1 | | 150 | ns | |
| $t_{WIT}$ | $\overline{\text{WR}}$ = 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 $\overline{\text{WR}}$ ↓.
2. Pulse width of initial Reset pulse after power on must be at least 50 μSec. Subsequent Reset pulses may be 500 ns minimum.

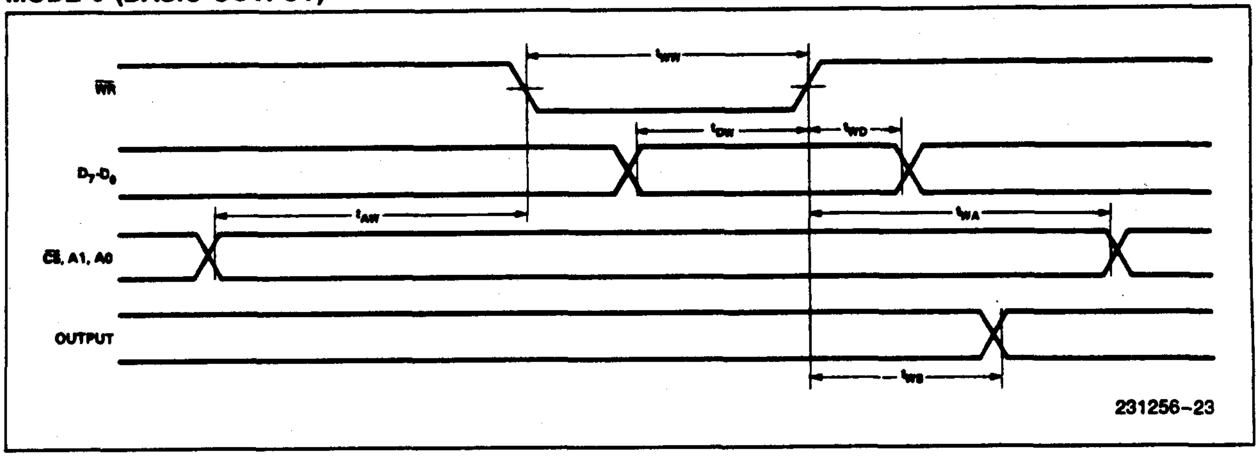
## WAVEFORMS

### MODE 0 (BASIC INPUT)



231256–22

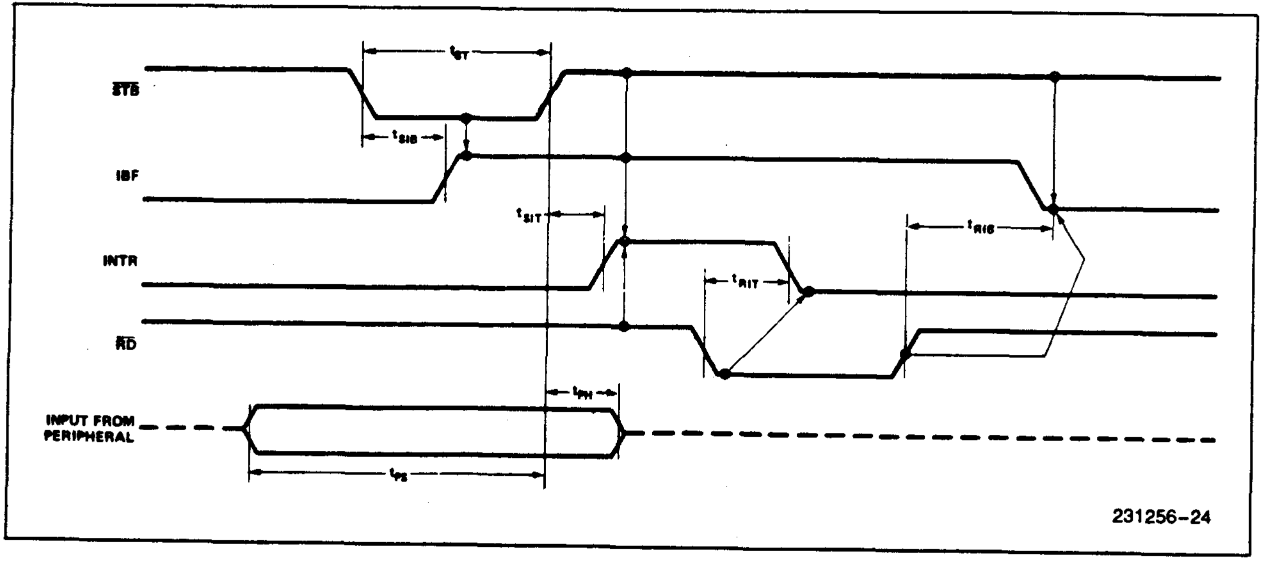
### MODE 0 (BASIC OUTPUT)



231256–23

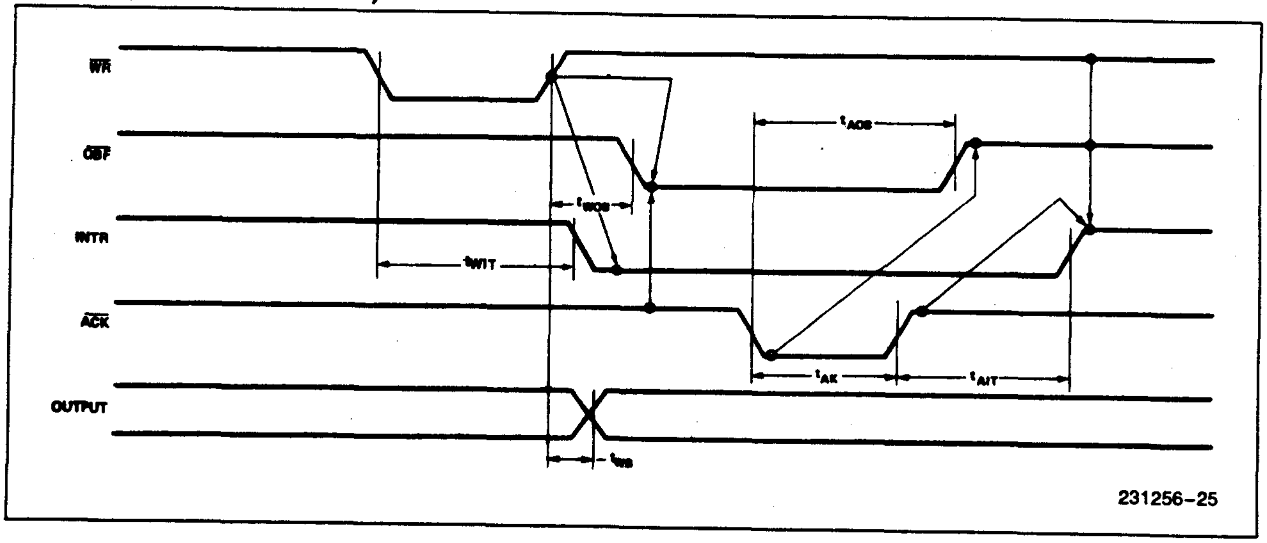
## WAVEFORMS (Continued)

### MODE 1 (STROBED INPUT)



231256-24

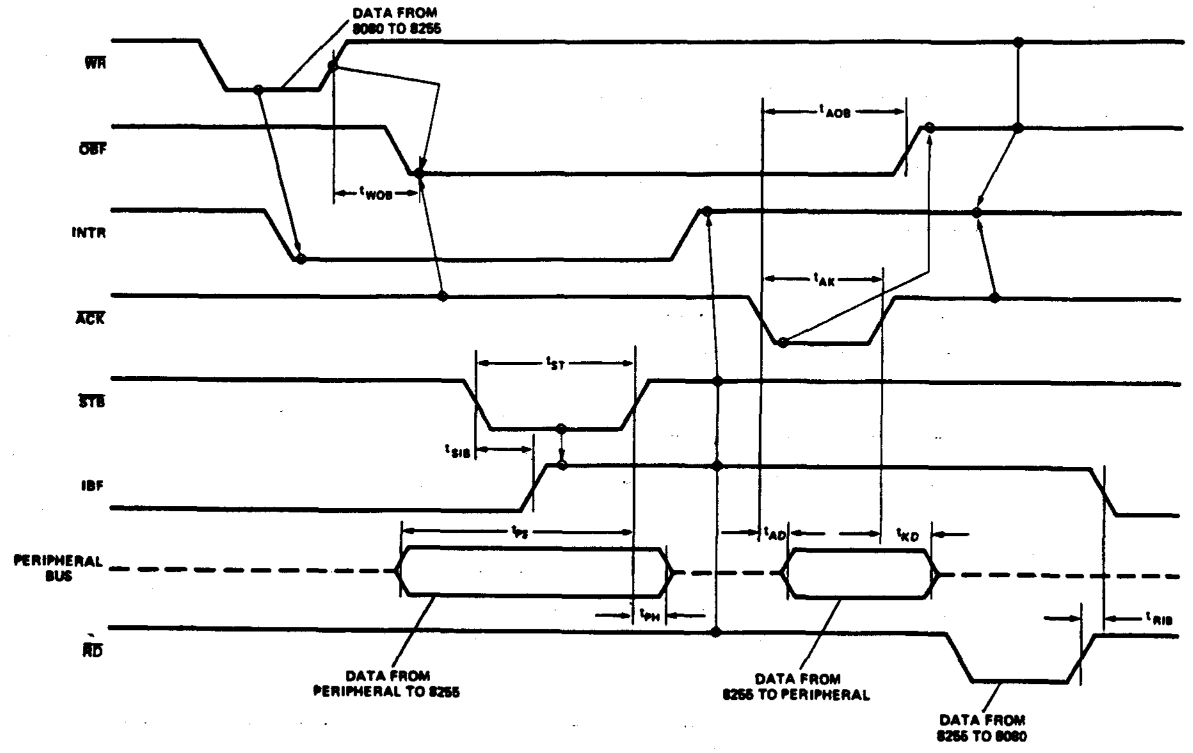
### MODE 1 (STROBED OUTPUT)



231256-25

## WAVEFORMS (Continued)

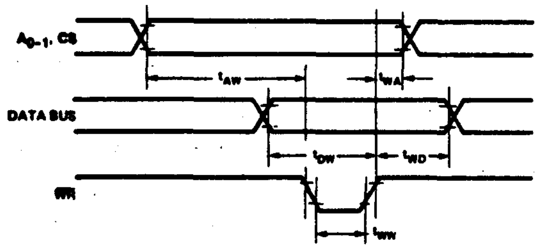
### MODE 2 (BIDIRECTIONAL)



231256–26

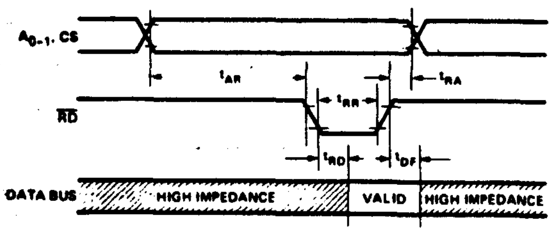
**Note:**
Any sequence where $\overline{WR}$ occurs before $\overline{ACK}$ AND $\overline{STB}$ occurs before $\overline{RD}$ is permissible.
(INTR = IBF • $\overline{MASK}$ • $\overline{STB}$ • $\overline{RD}$ + $\overline{OBF}$ • $\overline{MASK}$ • $\overline{ACK}$ • $\overline{WR}$)

### WRITE TIMING



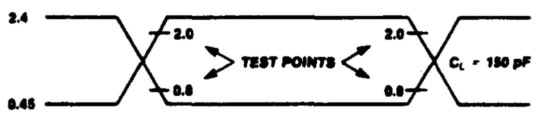
231256–27

### READ TIMING



231256–28

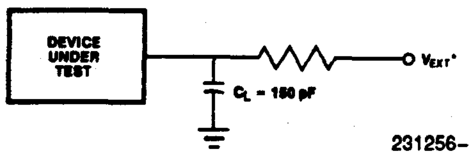
### A.C. TESTING INPUT, OUTPUT WAVEFORM



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.

# Intel 82C54 Programmable Interval Timer
# Data Sheet Reprint

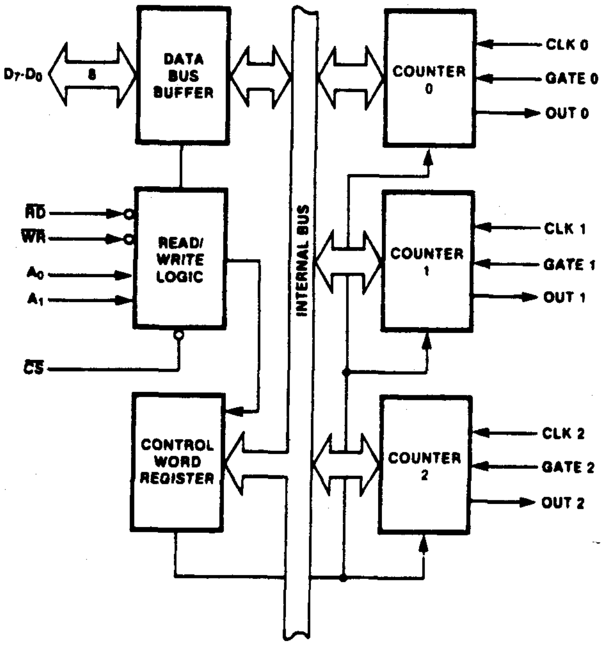
# 82C54
# CHMOS PROGRAMMABLE INTERVAL TIMER

- **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.

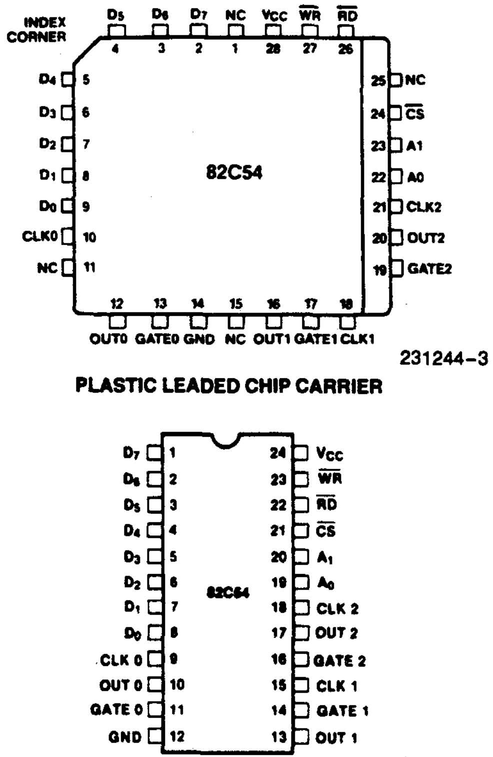
Six programmable timer modes allow the 82C54 to be used as an event counter, elapsed time indicator, programmable one-shot, and in many other applications.

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**



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

**Figure 2. 82C54 Pinout**

**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.<br>$A_1$ \| $A_0$ \| Selects<br>0 \| 0 \| Counter 0<br>0 \| 1 \| Counter 1<br>1 \| 0 \| Counter 2<br>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.

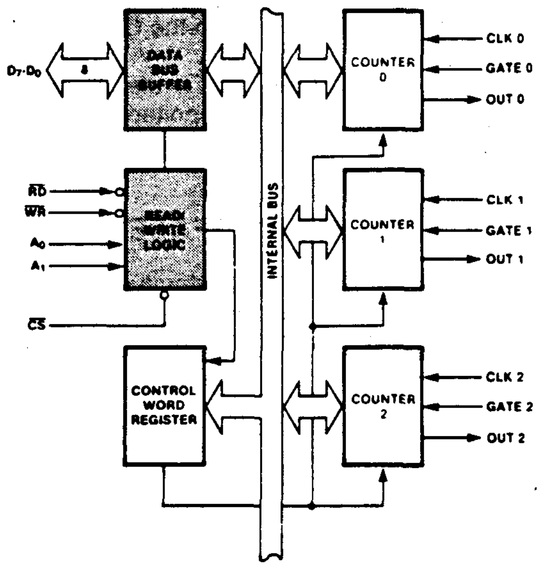
Some of the other counter/timer functions common to microcomputers which can be implemented with the 82C54 are:

- 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).



231244-4

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

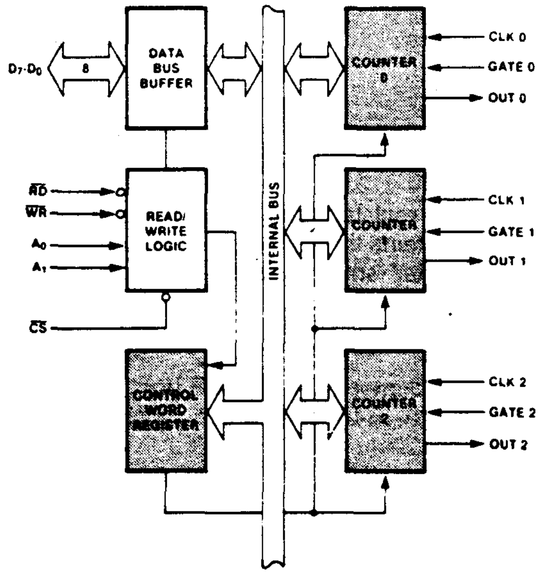
### READ/WRITE LOGIC

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.



231244-5

**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.

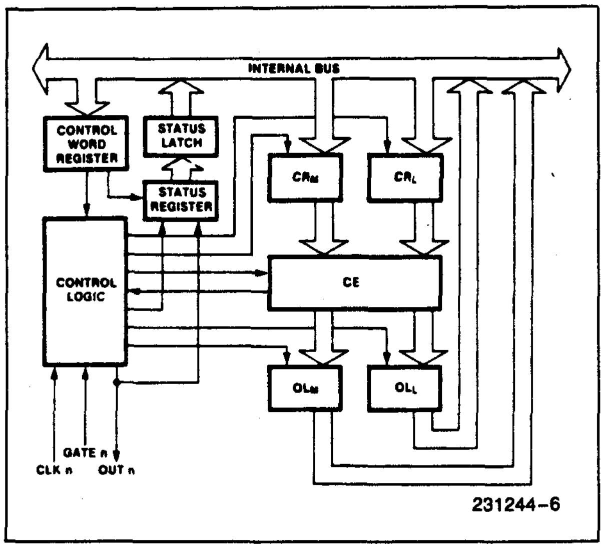


**Figure 5. Internal Block Diagram of a Counter**

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.

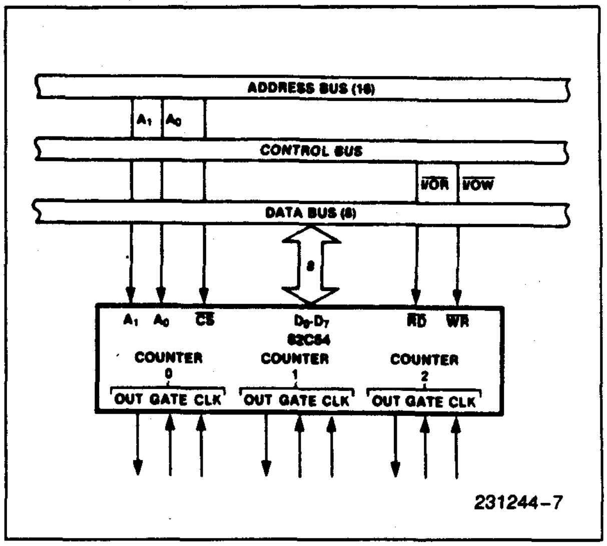


**Figure 6. 82C54 System Interface**

## 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) |

**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 |

**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. |

**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.

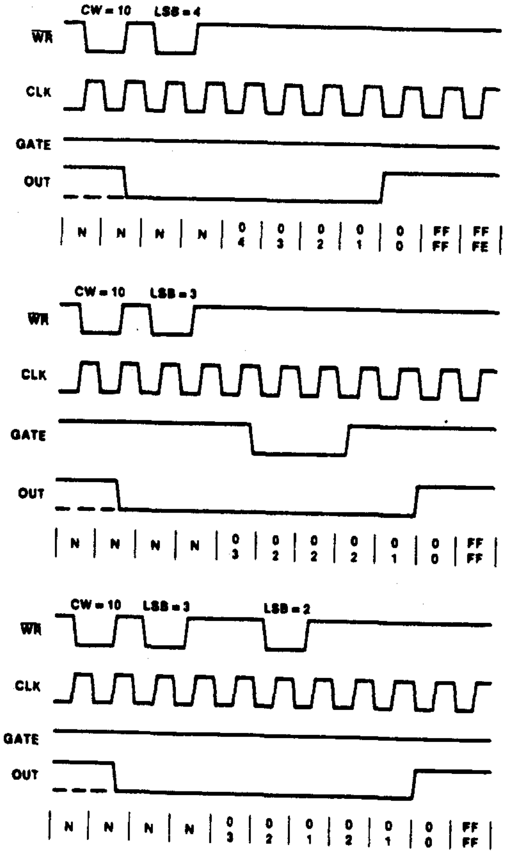
After the Control Word and initial count are written to a Counter, the initial count will be loaded on the next CLK pulse. This CLK pulse does not decrement the count, so for an initial count of N, OUT does not go high until N + 1 CLK pulses after the initial count is written.

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.



**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.

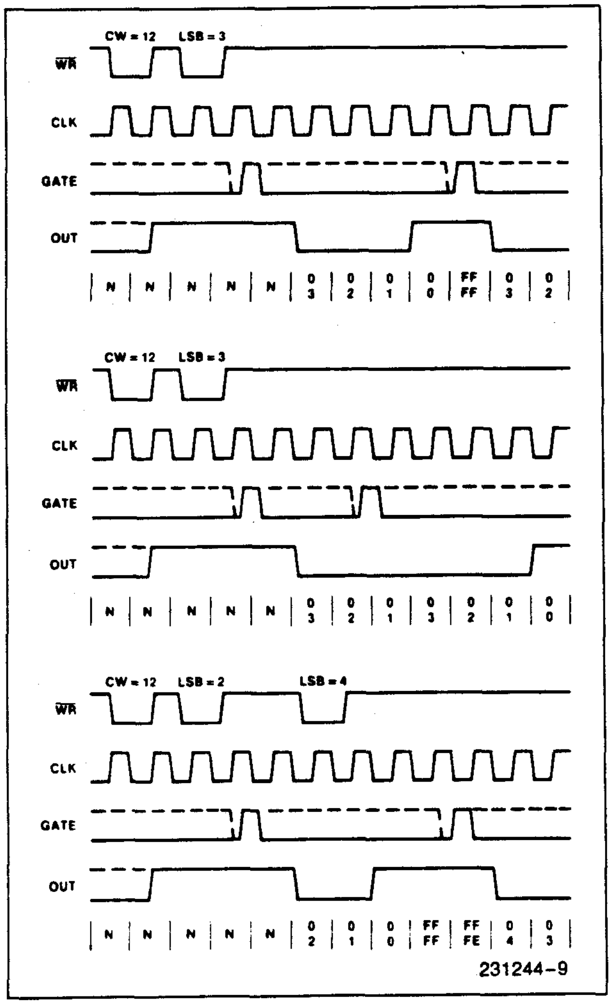

CW = 12 LSB = 3
WR
CLK
GATE
OUT
N N N N N 0 3 0 2 0 1 0 0 FF FF 0 3 0 2
CW = 12 LSB = 3
WR
CLK
GATE
OUT
N N N N N 0 3 0 2 0 1 0 3 0 2 0 1 0 0
CW = 12 LSB = 2 LSB = 4
WR
CLK
GATE
OUT
N N N N N 0 2 0 1 0 0 FF FF FF FE 0 4 0 3
231244-9

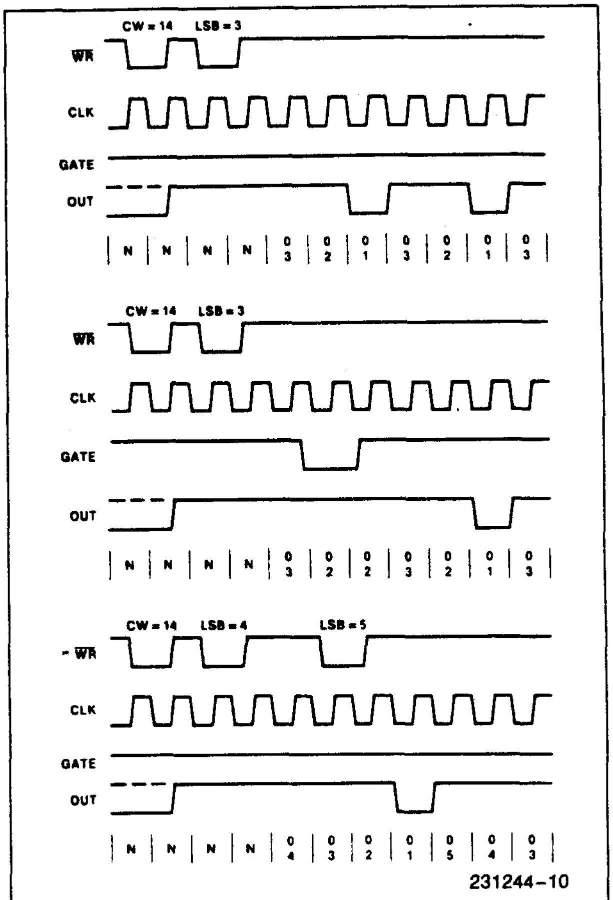
Figure 16. Mode 1

## 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.

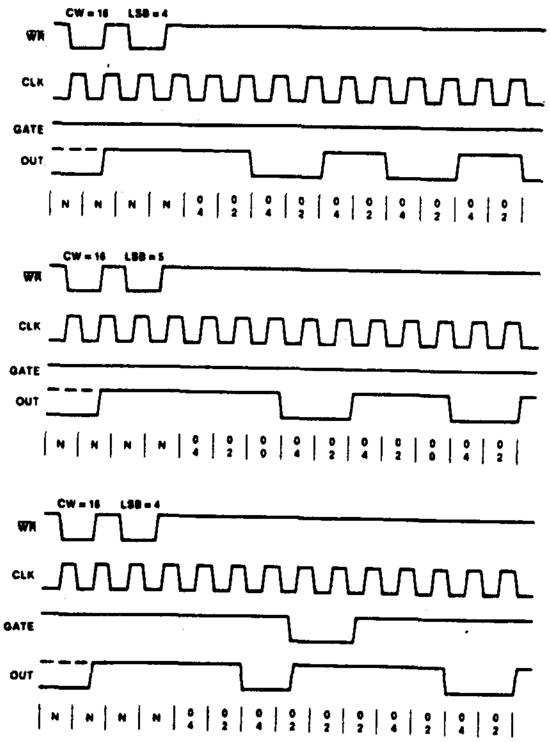
After writing a Control Word and initial count, the Counter will be loaded on the next CLK pulse. This allows the Counter to be synchronized by software also.

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.


CW = 16
LSB = 4
WR
CLK
GATE
OUT
N N N N 0 4 0 2 0 4 0 2 0 4 0 2 0 4 0 2 0 4 0 2
CW = 16
LSB = 5
WR
CLK
GATE
OUT
N N N N 0 4 0 2 0 0 0 4 0 2 0 4 0 2 0 0 0 4 0 2
CW = 16
LSB = 4
WR
CLK
GATE
OUT
N N N N 0 4 0 2 0 4 0 2 0 2 0 2 0 4 0 2 0 4 0 2

231244-11

**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.

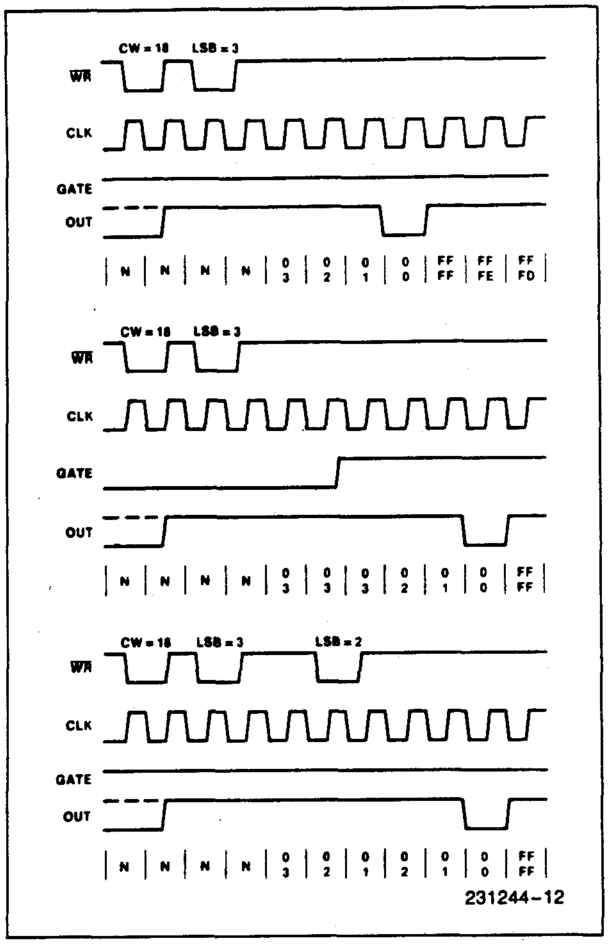

CW=18 LSB=3
WR
CLK
GATE
OUT
N N N N 0 3 0 2 0 1 0 0 FF FF FF FE FF FD
CW=18 LSB=3
WR
CLK
GATE
OUT
N N N N 0 3 0 3 0 3 0 2 0 1 0 0 FF FF
CW=18 LSB=3 LSB=2
WR
CLK
GATE
OUT
N N N N 0 3 0 2 0 1 0 2 0 1 0 0 FF FF
231244-12

**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.

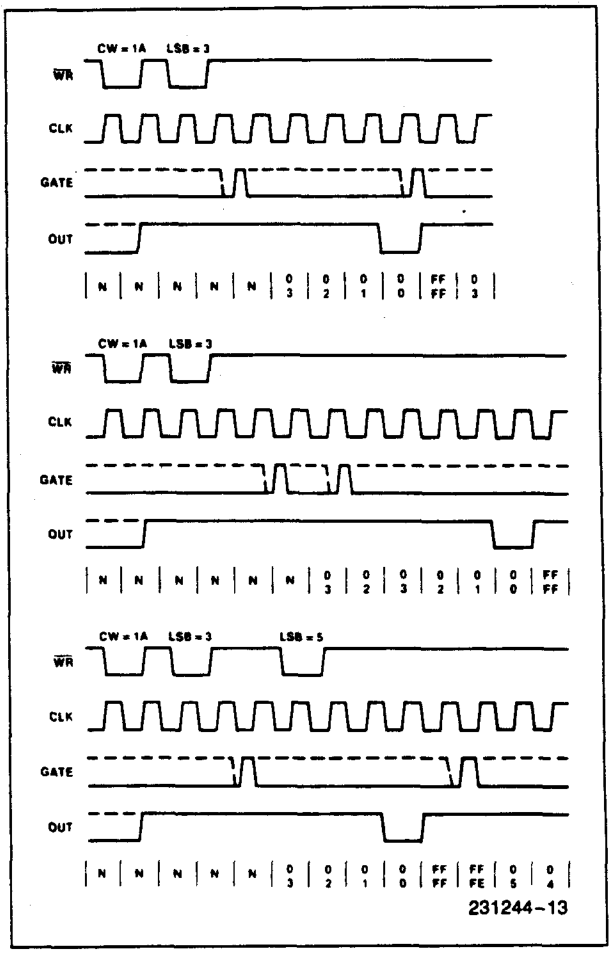

CW=1A LSB=3
WR
CLK
GATE
OUT
N N N N N 0 3 0 2 0 1 0 0 FF FF 0 3
CW=1A LSB=3
WR
CLK
GATE
OUT
N N N N N N 0 3 0 2 0 3 0 2 0 1 0 0 FF FF
CW=1A LSB=3 LSB=5
WR
CLK
GATE
OUT
N N N N N 0 3 0 2 0 1 0 0 FF FF FF FE 0 5 0 4
231244-13

**Figure 20. Mode 5**

| Signal Status Modes | Low Or Going Low | Rising | High |
|---|---|---|---|
| 0 | Disables counting | — | Enables counting |
| 1 | — | 1) Initiates counting<br>2) Resets output after next clock | — |
| 2 | 1) Disables counting<br>2) Sets output immediately high | Initiates counting | Enables counting |
| 3 | 1) Disables counting<br>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<br>$V_{CC}$ − 0.4 | | V<br>V | $I_{OH}$ = −2.5 mA<br>$I_{OH}$ = −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<br>10MHz 82C54-2 |
| $I_{CCSB}$ | $V_{CC}$ Supply Current-Standby | | 10 | μA | CLK Freq = DC<br>$\overline{CS}$ = $V_{CC}$.<br>All Inputs/Data Bus $V_{CC}$<br>All Outputs Floating |
| $I_{CCSB1}$ | $V_{CC}$ Supply Current-Standby | | 150 | μA | CLK Freq = DC<br>$\overline{CS}$ = $V_{CC}$. All Other Inputs,<br>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:**

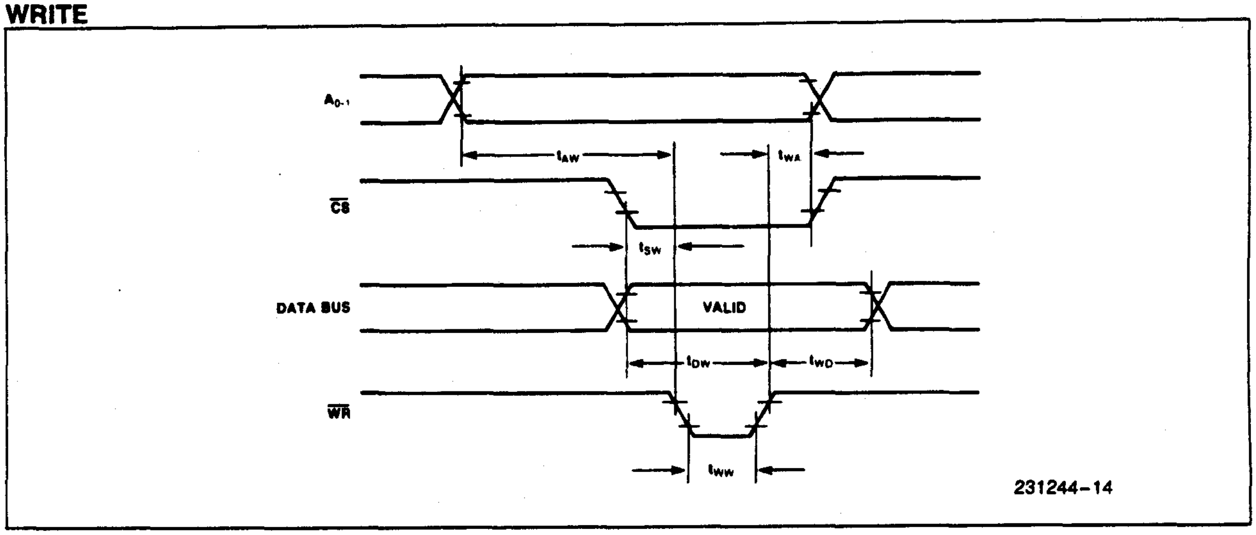
2. In Modes 1 and 5 triggers are sampled on each rising clock edge. A second trigger within 120 ns (70 ns for the 82C54-2) of the rising clock edge may not be detected.
3. Low-going glitches that violate $t_{PWH}$, $t_{PWL}$ may cause errors requiring counter reprogramming.
4. Except for Extended Temp., 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 |

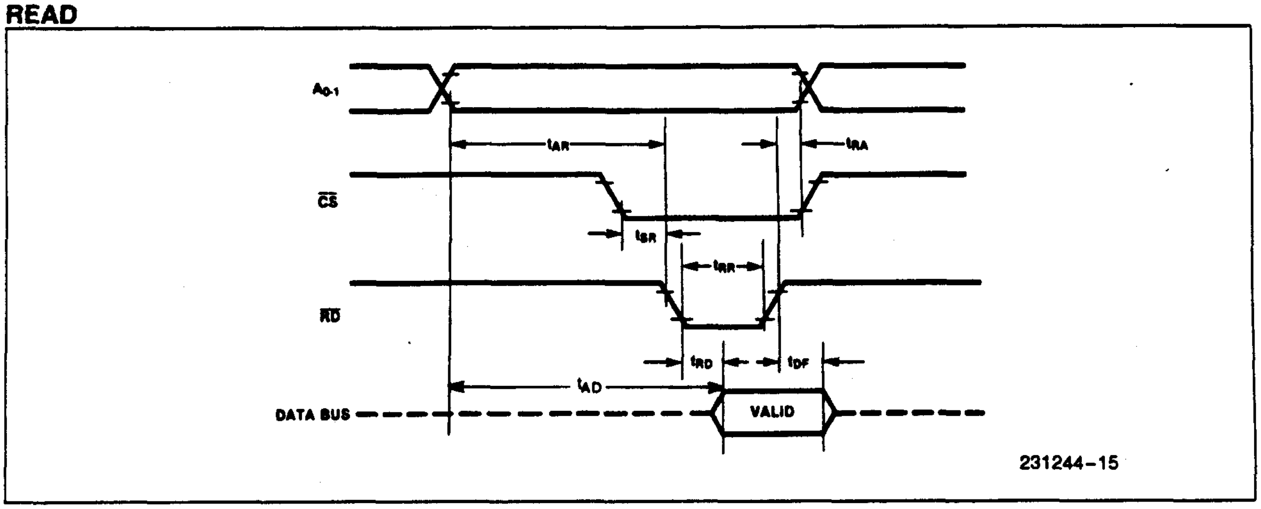
## WAVEFORMS

### WRITE



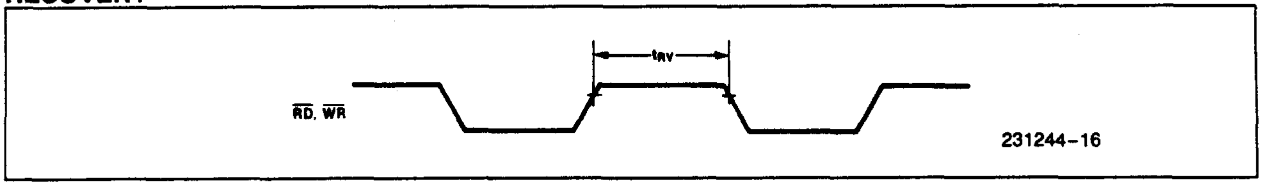
231244-14

### READ



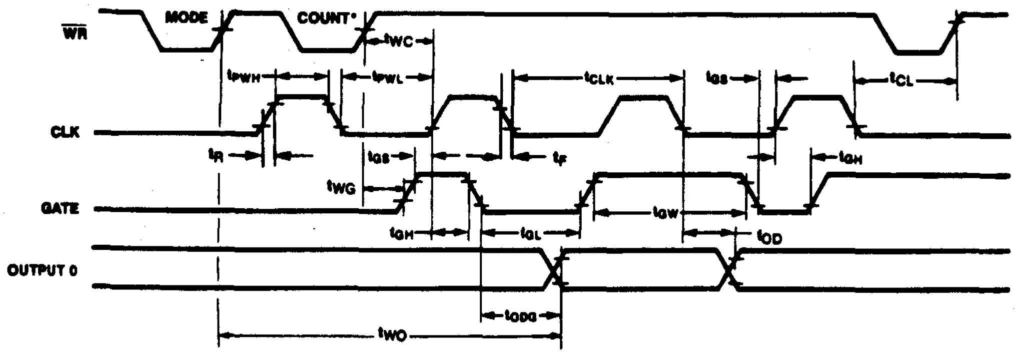
231244-15

### RECOVERY



231244-16

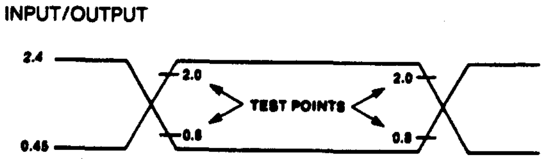
## CLOCK AND GATE



231244-17

* Last byte of count being written

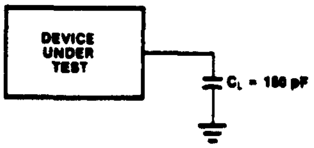
## A.C. TESTING INPUT, OUTPUT WAVEFORM



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

# 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.

## DIO24 Board User-Selected Settings

**Base I/O Address:**

| | |
|---|---|
| (hex) | (decimal) |