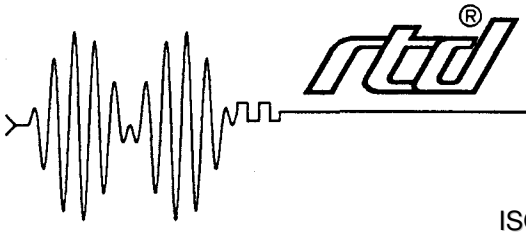
DG24
User's Manual



Real Time Devices, Inc.

"Accessing the Analog World"™

ISO9001 and AS9100 Certified

DG24

User's Manual

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

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

Copyright © 1992 by Real Time Devices, Inc.
All rights reserved

Printed in U.S.A.

Rev. C 9234

Table of Contents

LIST OF ILLUSTRATIONS

INTRODUCTION

The DG24 is a general purpose digital I/O board for use in the IBM PC/XT/AT or compatible computer. Installed within a single short or full-size expansion slot in the computer, the DG24 features:

- 24 TTL/CMOS 8255-based programmable digital I/O lines,
- Optional TTL buffered outputs for high driving capability (/B model),
- Optional pull-up/pull-down resistors,
- Simple I/O or strobed I/O operation,
- Hardware enabled interrupts (IRQ2-IRQ7),
- BASIC, Turbo Pascal, and Turbo C source code.

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 DG24 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 unbuffered 8255 can be operated in any one of the 8255's three modes. If you have purchased the DG24/B with TTL buffers for high driving capacity, the 8255 can be operated in Mode 0 when the buffers are installed. CMOS 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*.

What Comes With Your Board

You receive the following items in your DG24 package:

- DG24 or DG24/B (with TTL buffers) interface board
- Software diskette with 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 DG24 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 DG24 include the TB40 terminal board and XB40 prototype/terminal board for prototype development and easy signal access, and the XP40 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 DG24 board has jumper settings you can change if necessary for your application. 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 switches S1, S2, and S3 has been provided to bypass the 8255 buffers if you have the DG24/B buffered model.

Also note that by installing resistor packs at RN1-RN4 and setting the jumpers on P5, you can configure your digital input lines to be pulled up or pulled down. This procedure is explained near the end of this chapter.

RN5 through RN8 are provided to install resistor packs for ports configured as buffered outputs. These pull-down resistor packs are described 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 DG24 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 P2, the base address jumper, 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)
P2	Sets the base address to 1 of 8 I/O ports	300 hex (768 decimal)
P3	Connects 1 or more of the 3 interrupt sources to an interrupt channel	Interrupt channels disabled
P4	Sets the direction of buffered digital I/O lines on Ports A, B, CL & CH (active on DG24/B model only)	Ports A, B, CL & CH set as inputs (4 jumpers installed on IN pins)
P5	Sets resistor networks RN1-RN4 for Ports A, B, CL & CH as pull-ups (+5V) or pull-downs (GND); active only when optional resistor packs are installed	Disabled (no jumpers installed)
S1	Bypasses 8255 Port A buffers for Mode 2 operation (DG24/B only)	Open (buffers not bypassed)
S2	Bypasses 8255 Port B buffers for Mode 1 operation (DG24/B only)	Open (buffers not bypassed)
S3	Bypasses 8255 Port C buffers for Modes 1 & 2 operation (DG24/B only)	Open (buffers not bypassed)

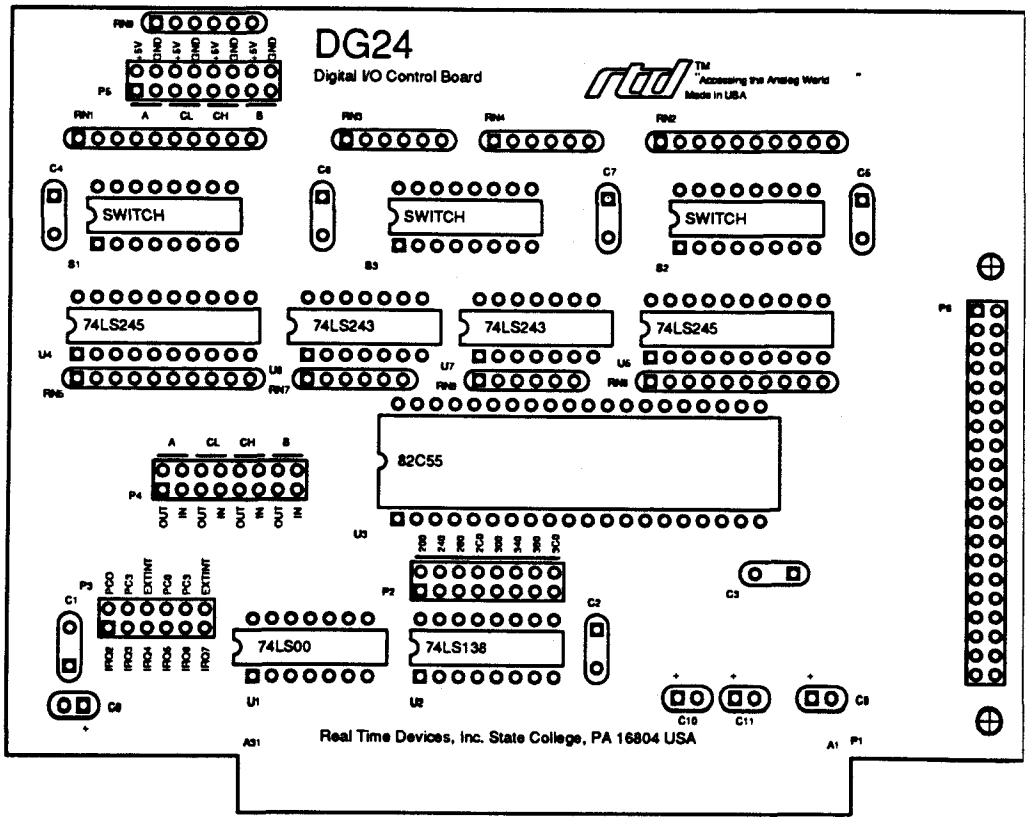


Fig. 1-1 — Board Layout Showing Factory-Configured Settings

P2 — 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 DG24 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 DG24 has a header connector, P2, which lets you select any one of eight 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. These addresses are, from left to right on P2:

Hexadecimal	Decimal
200	512
240	576
280	640
2C0	704
300	768
340	832
380	896
3C0	960

To change the base address setting, remove the jumper from the fifth from right pair of pins (300 hex) and, using Figure 1-2 as a guide, install it in the desired location. Record the new base address setting on the table inside the back cover of this manual.

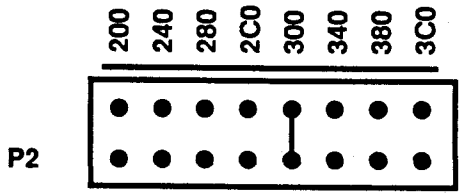


Fig. 1-2 — Base Address Jumper, P2

P3 — Interrupt Source and Channel Select (Factory Setting: Disabled)

This header connector, shown in Figure 1-3, lets you connect one of three interrupt sources to an interrupt channel for interrupt generation. These sources are: PC0, which is the INTRB signal from the 8255 PPI; PC3, which is the INTRA signal from the 8255 PPI; and EXTINT, an external interrupt you can route onto the board through the P6 I/O connector. Each source has two IRQ channels available to avoid contention. When selecting the interrupt and channel you desire, be sure that the IRQ channel is not used by other devices your computer system. Note that it is possible to use more than one interrupt source on the DG24. To connect an interrupt source, place the jumper across the desired set of pins. Figure 1-3a shows PC3 connected to IRQ3 and Figure 1-3b shows EXTINT connected to IRQ4.

It is important to note that the DG24 interrupt sources are not open collector. Therefore, do not attempt to connect one of these interrupts to any other interrupt output.

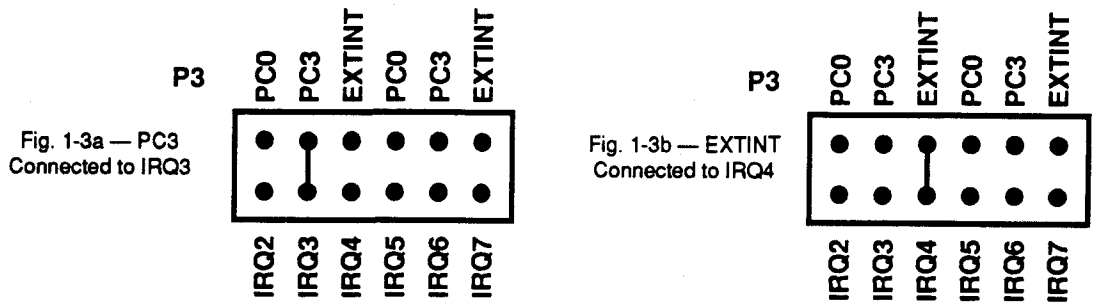


Fig. 1-3 — Interrupt Source and Channel Select Jumper, P3

P4 — Digital I/O Direction (Model DG24/B Only) (Factory Setting: IN)

This header connector, shown in Figure 1-4, sets the direction, input or output, of the buffered digital I/O lines on the DG24/B board. This header is not used if the /B option is not installed.

One jumper is installed for each group of lines, Port A, Port B, Port C lower, and Port C upper. Installing a jumper vertically across the IN pins configures a group as inputs; OUT configures them as outputs. One jumper must be installed for each buffered port for proper operation. If a particular port is shunted by using the port's DIP switch and removing the corresponding buffer, then the jumper on P4 for that port has no effect on operation.

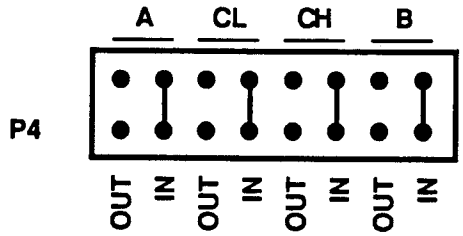


Fig. 1-4 — Digital I/O Direction Jumpers, P4

P5 — Pull-up/Pull-down Select (Factory Setting: Disabled (No Jumpers Installed))

The DG24 board provides four locations to add resistor networks to control the state of the Port A, Port B, Port C lower, and Port C upper I/O lines upon reset. The P5 header connector is used in conjunction with these optional resistor networks to configure them to function as pull-ups or pull-downs. Until a jumper is installed on this header, the corresponding resistor network is disabled. For each group of signals to be pulled up, install a jumper vertically between the +5V pin and the corresponding port pin. To pull a group of signals down, install the jumper between GND and the corresponding port pin. Note that only one jumper can be installed for each group of lines. Figure 1-5 shows all ports pulled up (resistor networks must be installed at RN1 through RN4 for the pull-ups to be active). There are no jumpers installed on this header connector when you receive the board.

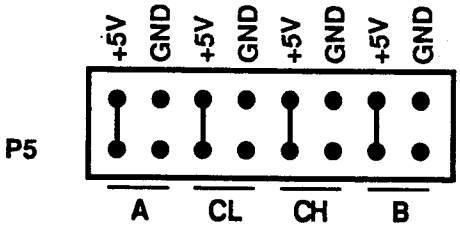


Fig. 1-5 — Pull-up/Pull-down Select Jumpers, P5

S1/S2/S3 — Buffer Bypass Switches (DG24/B Only) (Factory Setting: OPEN (Not Bypassed))

The 8255 can be operated in Mode 0 when buffers are installed on the I/O lines. Mode 1 and Mode 2 operation require some of these buffers to be removed. Additionally, buffers must be removed from any group of lines when you wish to be able to change direction dynamically through software. Table 1-2 shows the ports of the 8255 and their associated buffers and shunt switches.

Table 1-2 — 8255 Buffers and Shunts

Port	Buffer Location	Buffer Type	Shunt DIP Switch
A	U4	74LS245	S1
B	U5	74LS245	S2
CL	U6	74LS243	S3, 1-4
CH	U7	74LS243	S3, 5-8

Mode 0 Operation — If the direction of a port configured for Mode 0 operation is changed dynamically through software, all of the switches on the corresponding DIP switch must be set to the CLOSED position and the buffer must be removed for that port. This is required because the buffers are hardware-configured for a particular direction using the jumper at P4. Therefore, their direction cannot be changed through software. When removing the buffer, also remove the corresponding jumper on P4.

After closing the DIP switches, carefully remove the corresponding buffer from the printed circuit board. Locate the port that requires a DIP switch shunt in Table 1-2, then note the component labels of both the buffer and the associated DIP switch to verify that all settings are as desired.

In the event that shunts are required for only one half of Port C, the switches on DIP switch S3 can be closed in groups of four. Determine their positions from Table 1-2, then close the appropriate group of switches. Only the buffer corresponding to the half of Port C that requires shunts must be removed (refer to the table for its location).

Mode 1 Operation — When operating a group of lines in Mode 1, some of the Port C bits are used as handshaking signals. Therefore, the buffers that are installed at locations U6 and U7 must be removed and DIP switch S3 must have all switches closed to allow for the transmission of these signals in both directions: both to and from Port C. Buffers may still be used for Ports A and B, input or output.

As with Mode 0 operation, buffers cannot be used for Port A or Port B if the Mode 1 direction is changed dynamically under software control. In this case, the appropriate DIP switches must be closed for these ports and the corresponding buffers and P4 jumpers removed.

Mode 2 Operation — When operating the 8255 in Mode 2, the lines of Port A must be bidirectional and the lines of Port C function as control lines, some as outputs and some as inputs. When using Mode 2, both the Port A and Port C buffers must be removed and bypassed. Buffers may still be used for Port B.

Installing and Removing Buffers — Whenever you install a buffer for an 8255 port, be sure to OPEN its corresponding DIP switches and set its direction on P4. When removing a buffer, CLOSE the corresponding DIP switches and remove the jumper from P4. Figure 1-6 shows the Port A buffer circuitry; Figure 1-7 shows the Port B buffer circuitry; and Figure 1-8 shows the Port C buffer circuitry.

CAUTION: Remember, whenever you close the switches on S1, S2, or S3, **be sure** to remove the corresponding buffers from the board. Failure to do so may damage the board.

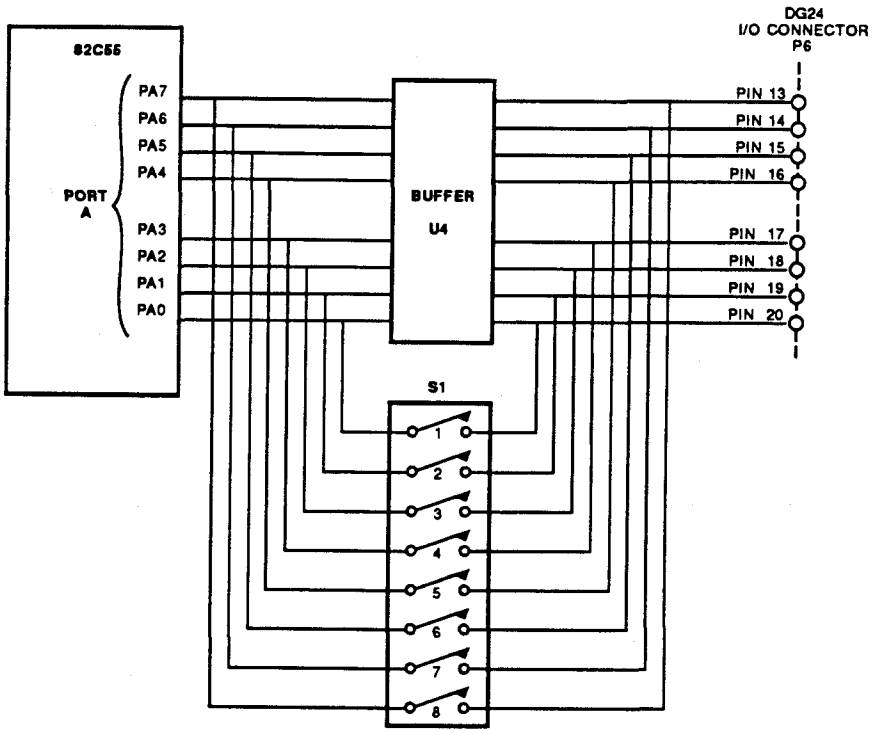


Fig. 1-6 — Port A Buffer Circuitry

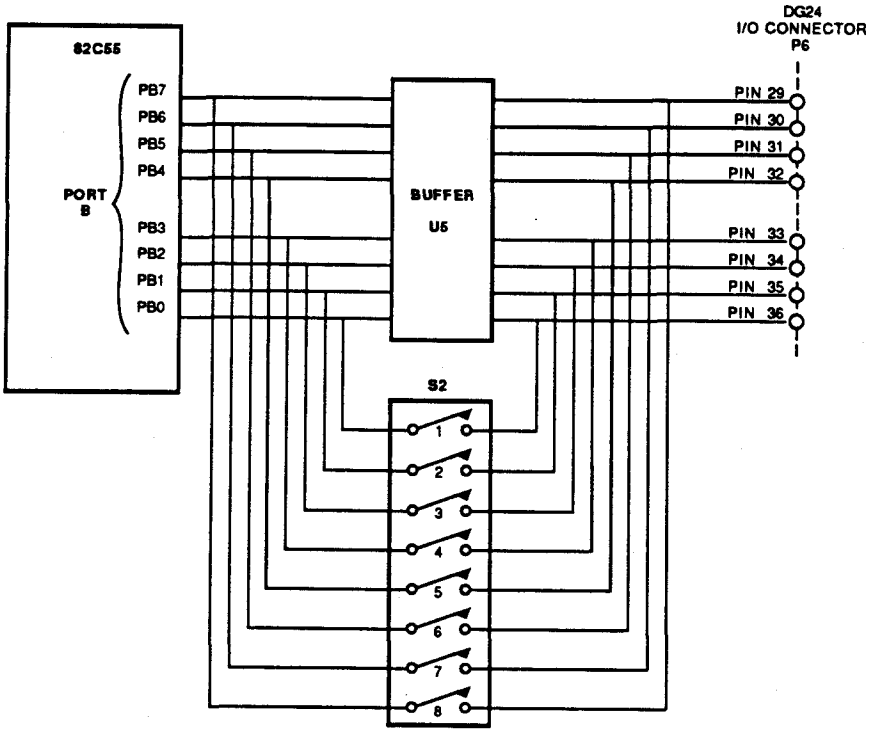


Fig. 1-7 — Port B Buffer Circuitry

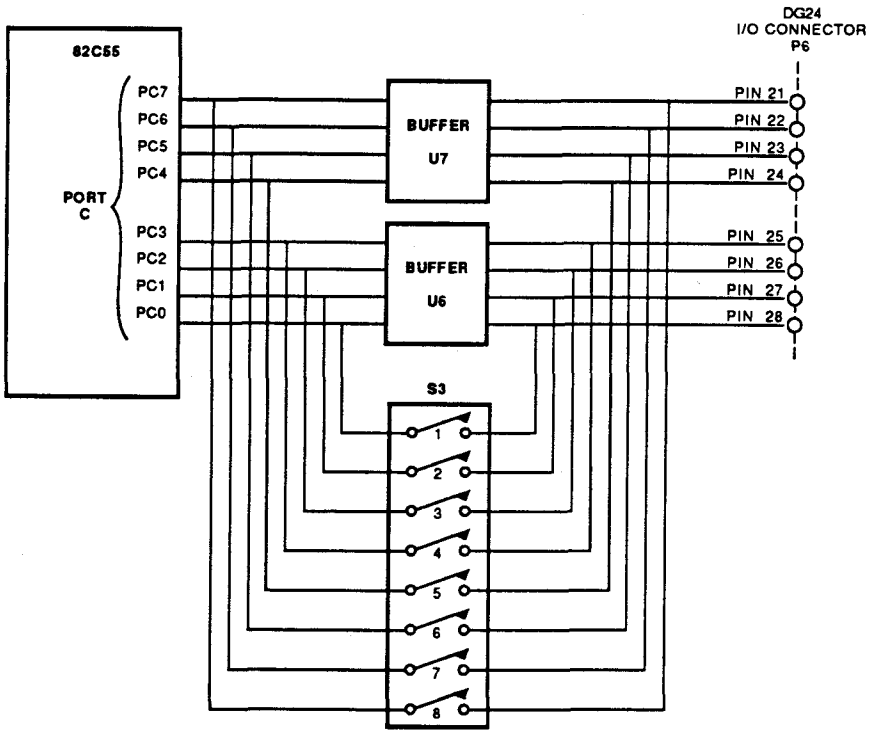


Fig. 1-8 — Port C Buffer Circuitry

Pull-up/Pull-down Resistors on Digital Input Lines (RN1-RN4)

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, eight Port B lines, four Port C Lower 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 when they are configured as input ports.

To use the pull-up/pull-down feature, you must first install 10 kilohm resistor packs (recommended value) in any or all of the four locations, RN1 through RN4, as shown in the table below. Note that these resistor networks are independent of the pull-down resistor networks RN5 through RN8 used on buffered output lines as described in the following section.

Port	SIP Pack	Input Port Resistor Network
A	10 pin	RN1
B	10 pin	RN2
CL	6 pin	RN3
CH	6 pin	RN4

After the resistor packs are installed, you must connect them into the circuit as pull-ups or pull-downs. This is done by placing the corresponding jumper on P5 for each port's resistor network across the +5V pins (pull-up) or across the GND pins (pull-down).

Pull-down Resistors on Buffered Digital Output Lines (RN5-RN8)

When you configure a port to provide buffered outputs, you may want to install a pull-down resistor network in the appropriate location on the board as shown in the table below to keep the buffered output lines low during the time between system power-up or reset and initialization of the PPI. A recommended value of resistance for these pull-downs is 10 kilohms.

Port	SIP Pack	Buffered Output Resistor Network
A	10 pin	RN5
B	10 pin	RN6
CL	6 pin	RN7
CH	6 pin	RN8

CHAPTER 2

BOARD INSTALLATION

The DG24 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.

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.

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 DG24's P6 I/O connector pinout. Refer to this diagram as you make your I/O connections.

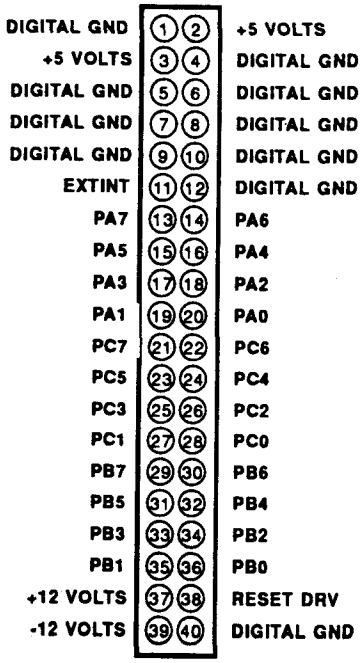


Fig. 2-1 — P6 I/O Connector Pin Assignments

Connecting the Digital I/O

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 a DIGITAL GND (P6, pins 1, 4-10, and 40).

Connecting the External Interrupt

The DG24 can receive an externally generated interrupt signal, EXTINT, through I/O connector P6, pin 11 and route it to an IRQ channel through on-board header connector P3. Interrupt generation is enabled through hardware. 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. You must take the EXTINT line high until the interrupt routine is serviced.

Connecting the Reset Drv Pin

The RESET DRV pin (P6-38) can be used to connect the RESET signal generated by the PC to external circuitry. The RESET DRV is an active high signal (i.e., the line goes high during a RESET condition).

CHAPTER 3

HARDWARE DESCRIPTION

This chapter describes the major features of the DG24's 8255 based digital I/O. This chapter also describes the hardware-selectable interrupts.

The DG24 provides 24 digital I/O lines, with buffered lines available on the DG24/B model, as shown Figure 3-1. This chapter describes the hardware which makes up the digital I/O circuitry and hardware-selectable interrupts.

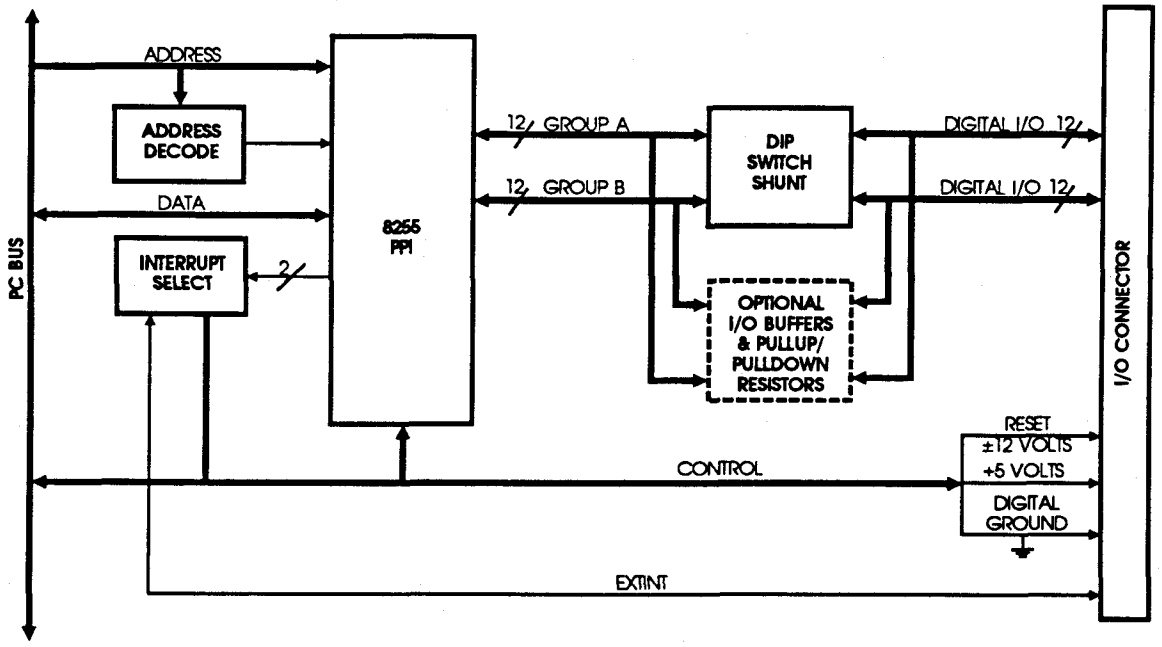


Fig. 3-1 — DG24 Block Diagram

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).

The PPI has three modes of operation:

Mode 0 — Basic input/output. Provides simple input and output operations for each port. Data is written to or read from a specified port.

Mode 1 — Strobed input/output. Provides a means for transferring I/O data to or from Port A or Port B in conjunction with strobes or handshaking signals.

Mode 2 — Strobed bidirectional input/output. Provides a bidirectional means of communicating with another device on a single eight-bit bus. Handshaking signals are similar to mode 1. This mode applies to Port A only.

In Mode 0, all four ports (A, B, C lower, and C upper) are available as I/O lines. Sixteen configurations are possible in this mode, and any port can be configured as an input or an output. The outputs are latched, but the inputs are not latched.

In Mode 1, the four ports are grouped into two groups. Each group contains one eight-bit data port (Port A or Port B) and one four-bit control/data port (Port C lower or Port C upper) which is used for control and status of the eight-bit port. The eight-bit data port in each group can be configured as an input or an output. Both inputs and outputs are latched. Because Port C is used bidirectionally in this mode, Port C buffers must be removed from the /B board and bypassed for Mode 1 (see Chapter 1).

In Mode 2, Port A is an eight-bit bidirectional bus and Port C is a five-bit control port. Port B cannot be used in this mode, but is available for use in Mode 0 or Mode 1 while Port A is in Mode 2. Both inputs and outputs are latched. On the /B board, Port A and Port C buffers must be removed and bypassed when using Mode 2 (see Chapter 1).

The PPI is configured by writing a control word to the appropriate I/O address location, as described in Chapter 4. The control word can also be used to individually set or reset the Port C bits. This feature allows any bit of Port C to be set or reset without affecting the other port C bits.

The PPI can also be used to generate interrupts in Mode 1 or Mode 2 operation. In these modes, the interrupt enable (INTE) mask is used to enable the INTRA (PC3) and INTRB (PC0) interrupt signals.

To enhance its capabilities, the PPI can be ordered with the /B TTL buffer option. The buffer circuitry allows the PPI to drive long cables with output signals and provides noise immunity for input signals. However, as noted above, buffers cannot be used for some ports when operating in Modes 1 or 2, or when dynamically changing the port direction through software control. On-board DIP switches are included to bypass the buffers. When these DIP switches are closed and their corresponding buffers are removed, then the I/O lines controlled by them are shunted. Each of the four ports, A, B, CL, or CH, is controlled by one DIP switch and buffer. Chapter 1 describes how to set the switches and remove buffers.

Interrupts

The DG24 can use any one of three signal sources to generate interrupts. These sources are: PC3, which is the INTRA signal from the 8255 PPI; PC0, which is the INTRB signal from the 8255 PPI; and EXTINT, an external interrupt you can route onto the board through the P6 I/O connector. Chapter 1 tells you how to set the jumpers on interrupt header connector P3, and Chapter 4 provides some programming information.

CHAPTER 4

BOARD OPERATION AND PROGRAMMING

This chapter shows you how to program and use your DG24 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, and BASIC, include source code to simplify your applications programming.

Defining the I/O Map

The I/O map for the DG24 is shown in Table 4-1 below. As shown, the board occupies four consecutive I/O port locations. The base address (designated as BA) can be selected using header connector P2 as described in Chapter 1, *Board Settings*. The following sections describe the register contents of each address used in the I/O map.

Table 4-1 — DG24 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	Reserved	Program PPI configuration	BA + 3

* 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 P6, and into PPI Port A; a write transfers the written data from Port A through P6 to an external device.

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

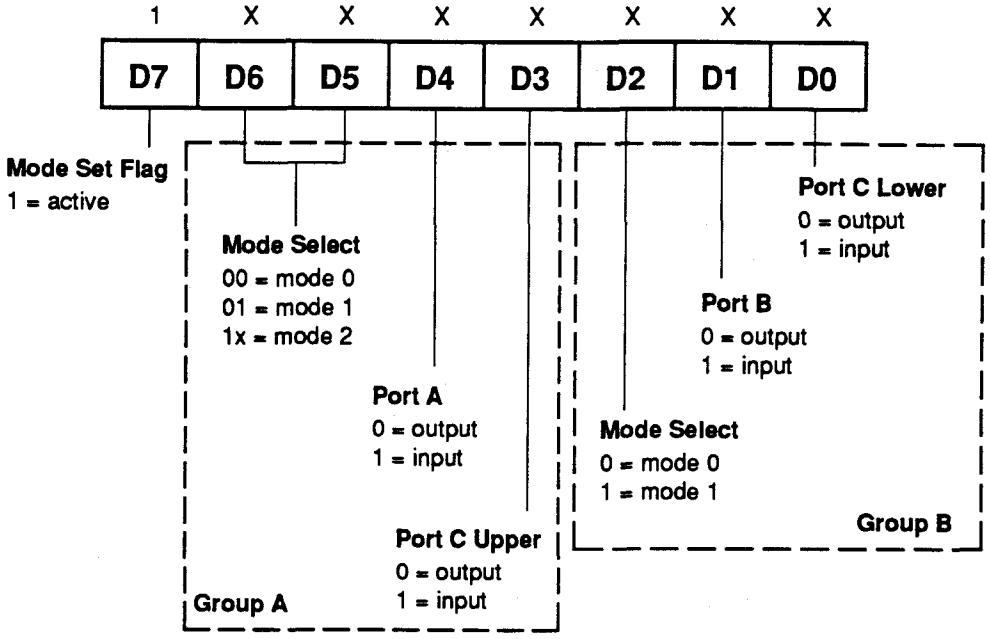
Transfers the 8-bit Port B digital input and digital output data between the board and an external device. A read transfers data from the external device, through P6, and into PPI Port B; a write transfers the written data from Port B through P6 to an external device.

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 P6, and into PPI Port C; a write transfers the written data from Port C through P6 to an external device.

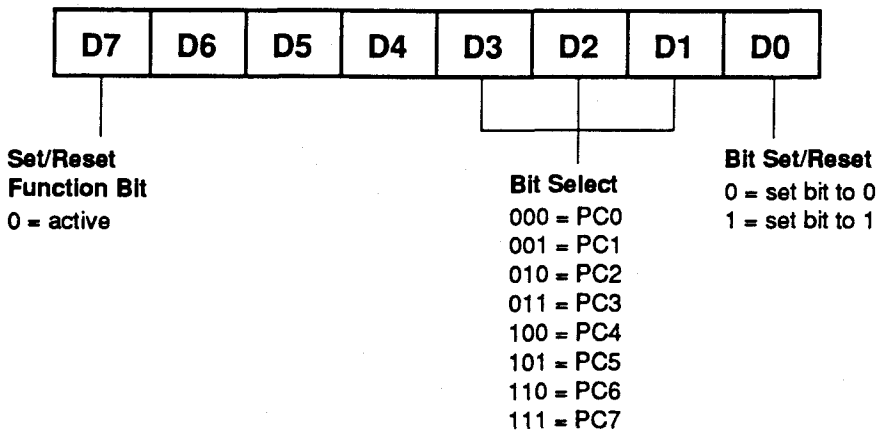
BA + 3: 8255 PPI Control Word (Write Only)

When bit 7 of this word is set to 1, a write programs the PPI configuration. Note that the D2 and D6 Mode Select bits should be set for 0 (Mode 0 operation) in the fully buffered /B board. The table below shows the control words for the 16 possible Mode 0 Port I/O combinations.

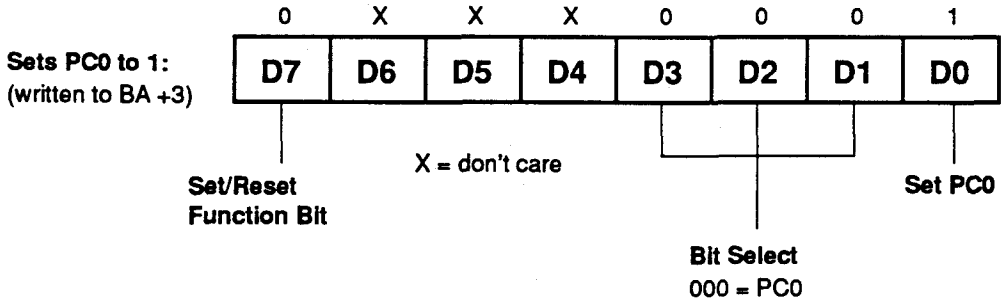


8255 Port I/O Flow Direction and Control Words, Mode 0						
Group A		Group B		Control Word		
Port A	Port C Upper	Port B	Port C Lower	Binary	Decimal	Hex
Output	Output	Output	Output	10000000	128	80
Output	Output	Output	Input	10000001	129	81
Output	Output	Input	Output	10000010	130	82
Output	Output	Input	Input	10000011	131	83
Output	Input	Output	Output	10001000	136	88
Output	Input	Output	Input	10001001	137	89
Output	Input	Input	Output	10001010	138	8A
Output	Input	Input	Input	10001011	139	8B
Input	Output	Output	Output	10010000	144	90
Input	Output	Output	Input	10010001	145	91
Input	Output	Input	Output	10010010	146	92
Input	Output	Input	Input	10010011	147	93
Input	Input	Output	Output	10011000	152	98
Input	Input	Output	Input	10011001	153	99
Input	Input	Input	Output	10011010	154	9A
Input	Input	Input	Input	10011011	155	9B

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



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



Programming the DG24

This section gives you some general information about programming and the DG24 board, and then walks you through the major DG24 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 DG24 is programmed by writing to and reading from the correct I/O port locations on the board. These I/O ports were defined in the previous section. Most high-level languages such as BASIC, Pascal, C, and C++, and of course assembly language, make it very easy to read/write these ports. The table below shows you how to read from and write to I/O ports using some popular programming languages.

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

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

In addition to being able to read/write the I/O ports on the DG24, you must be able to perform a variety of operations that you might not normally use in your programming. The table below shows you some of the operators discussed in this section, with an example of how each is used with Pascal, C, and BASIC. Note that the modulus operator is used to retrieve the least significant byte (LSB) of a two-byte word, and the integer division operator is used to retrieve the most significant byte (MSB).

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

Many compilers have functions that can read/write either 8 or 16 bits from/to an I/O port. For example, Turbo Pascal uses **Port** for 8-bit port operations and **PortW** for 16 bits, Turbo C uses **inportb** for an 8-bit read of a port and **inport** for a 16-bit read. **Be sure to use only 8-bit operations with the DG24!**

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);
```

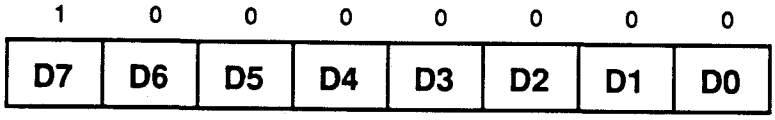
A final note: Don't be intimidated by the binary operators AND and OR and try to use operators for which you have a better intuition. For instance, if you are tempted to use addition and subtraction to set and clear bits in place of the methods shown above, DON'T! Addition and subtraction may seem logical, 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 DG24 board functions.

Initializing the 8255 PPI

Before you can operate the DG24, 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 certain modes are not supported when the digital I/O lines are buffered (/B board). In the example below, a decimal value of 128 sets up the 8255 so that all I/O lines are Mode 0 outputs.



Digital I/O Operations

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

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 DG24 board can interrupt the processor when any of the three interrupt sources is enabled (jumpers installed on P3). 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 DG24 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 DG24), 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 write an end-of-interrupt command to the 8259 controller. Finally, when exiting the ISR, in addition to popping all the registers you pushed on entrance, you must use the IRET instruction and **not** a plain RET. The IRET automatically pops the flags, CS, and IP that were pushed when the interrupt was called.

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

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

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

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

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

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

Your ISR should have this structure:

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

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

In C:

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

In Pascal:

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

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

The next step after writing the ISR is to save the startup state of the interrupt mask register and the interrupt vector that you will be using. The IMR is located at I/O port 21H. The interrupt vector you will be using is located in the interrupt vector table which is simply an array of 256-bit (4-byte) pointers and is located in the first 1024 bytes of memory (Segment = 0, Offset = 0). You can read this value directly, but it is a better practice to use DOS function 35H (get interrupt vector). Most C and Pascal compilers provide a library routine for reading the value of a vector. The vectors for the hardware interrupts are vectors 8 through 15, where IRQ0 uses vector 8, IRQ1 uses vector 9, and so on. Thus, if the DG24 will be using IRQ3, you should save the value of interrupt vector 11.

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

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

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

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

– Restoring the Startup IMR and Interrupt Vector

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

- Common Interrupt Mistakes

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

Example Programs

Included with the DG24 is a set of example programs that demonstrate the use of many of the board's features. These examples are in written in C, Pascal, and BASIC. Also included is an easy-to-use menu-driven diagnostics program, DG24DIAG, 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 DG24 board.

Digital I/O:

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

BASIC Programs

These programs are source code files so that you can easily develop your own custom software for your DG24 board.

Digital I/O:

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

APPENDIX A

DG24 SPECIFICATIONS

DG24 Characteristics Typical @ 25° C

Interface

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

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

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 ..... Unbuffered: -100 µA, max;
/B TTL buffer: -15 mA, max
Low-level output current, Isink ..... Unbuffered: 1.7 mA, max
/B TTL buffer: 24 mA, max
Darlington drive current ..... -1 mA, min; -5 mA max
(Available on any 8 pins on Ports B & C)
Input load current ..... Unbuffered: ±10 µA
/B TTL buffer: +20/-200 µA
Input capacitance ..... 10 pF
Input capacitance,
C(IN)@F=1MHz ..... 10 pF
Output capacitance,
C(OUT)<@F=1MHz ..... 20 pF

Miscellaneous I/Os

±12V, +5V, Digital GND (PC bus-sourced)
EXTINT
RESET DRV

Current Requirements

12 mA @ +5 volts (unbuffered)
125 mA @ +5 volts (/B buffers installed)

Connectors

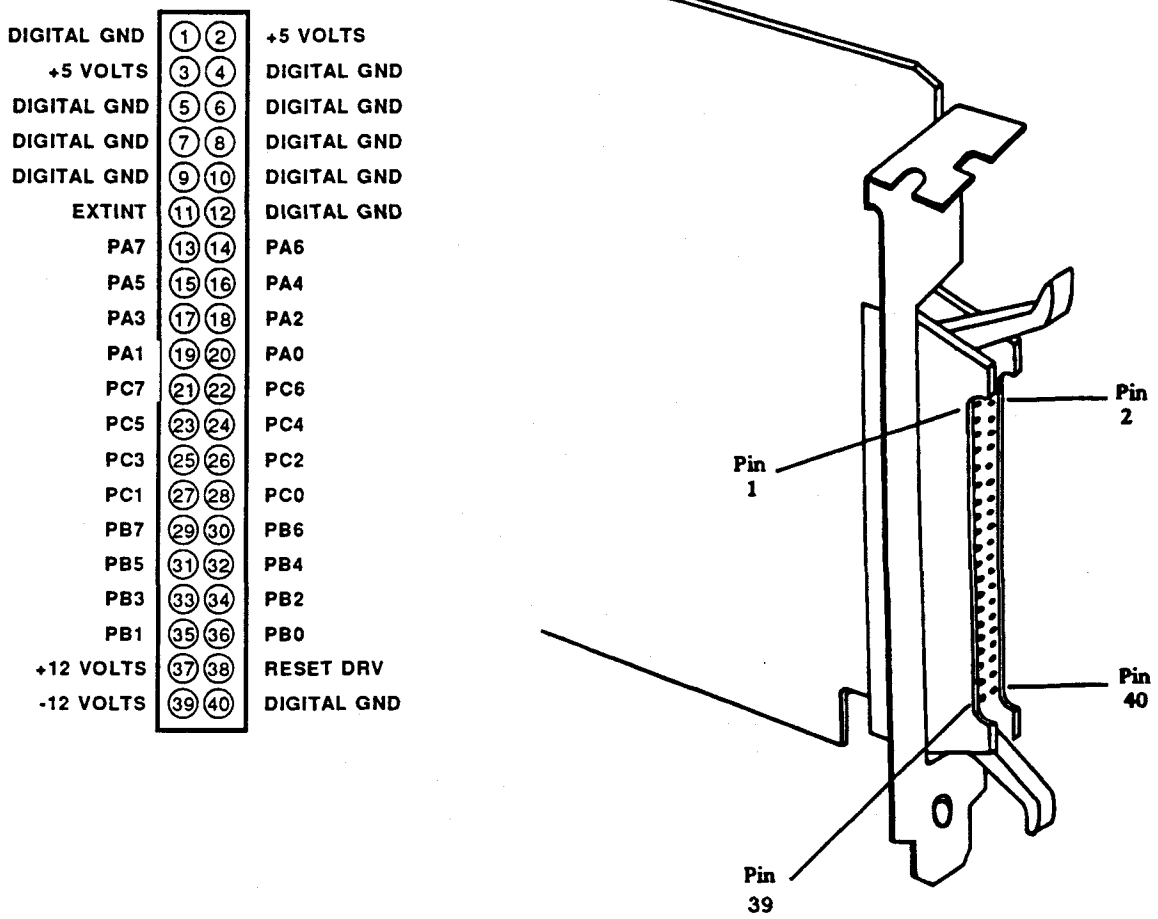
P6 — 40-pin right angle shrouded header with ejector tabs

Size

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

APPENDIX B

P6 CONNECTOR PIN ASSIGNMENTS



DG24 P6 Connector/Mating Connector		
Manufacturer	**DG24 P6 Connector**	**P6 Mating Connector**
Fujitsu 3M Robinson Nugent MIL C-83503	FCN-705Q040-AU/M	FCN-707B040-AU/B 3417-7040 IDS-C40PK-C-SR-TG M83503/7-09

APPENDIX C

COMPONENT DATA SHEETS

Intel 82C55A Programmable Peripheral Interface
Data Sheet Reprint

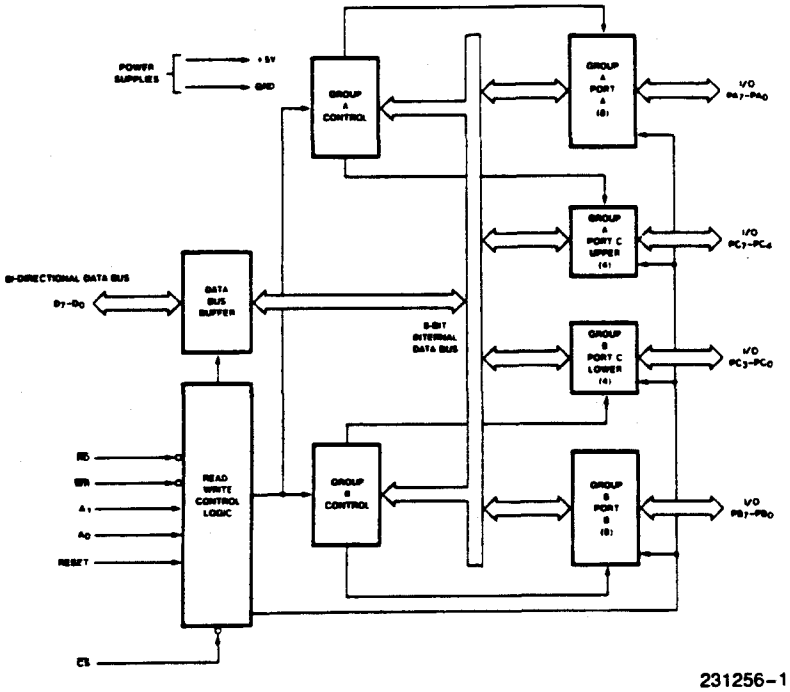
82C55A
CHMOS PROGRAMMABLE PERIPHERAL INTERFACE

- **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.

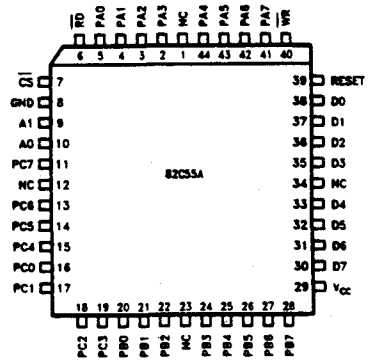
In MODE 0, each group of 12 I/O pins may be programmed in sets of 4 and 8 to be inputs or outputs. In MODE 1, each group may be programmed to have 8 lines of input or output. 3 of the remaining 4 pins are used for handshaking and interrupt control signals. MODE 2 is a strobed bi-directional bus configuration.

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.

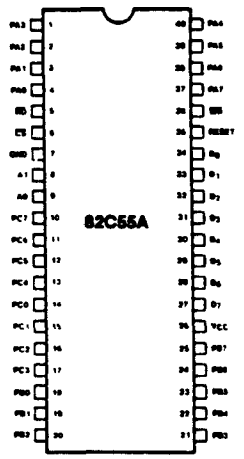


231256-1

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

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

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

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

82C55A FUNCTIONAL DESCRIPTION

General

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

Data Bus Buffer

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

Read/Write and Control Logic

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

Group A and Group B Controls

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

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

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

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

Ports A, B, and C

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

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

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

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

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

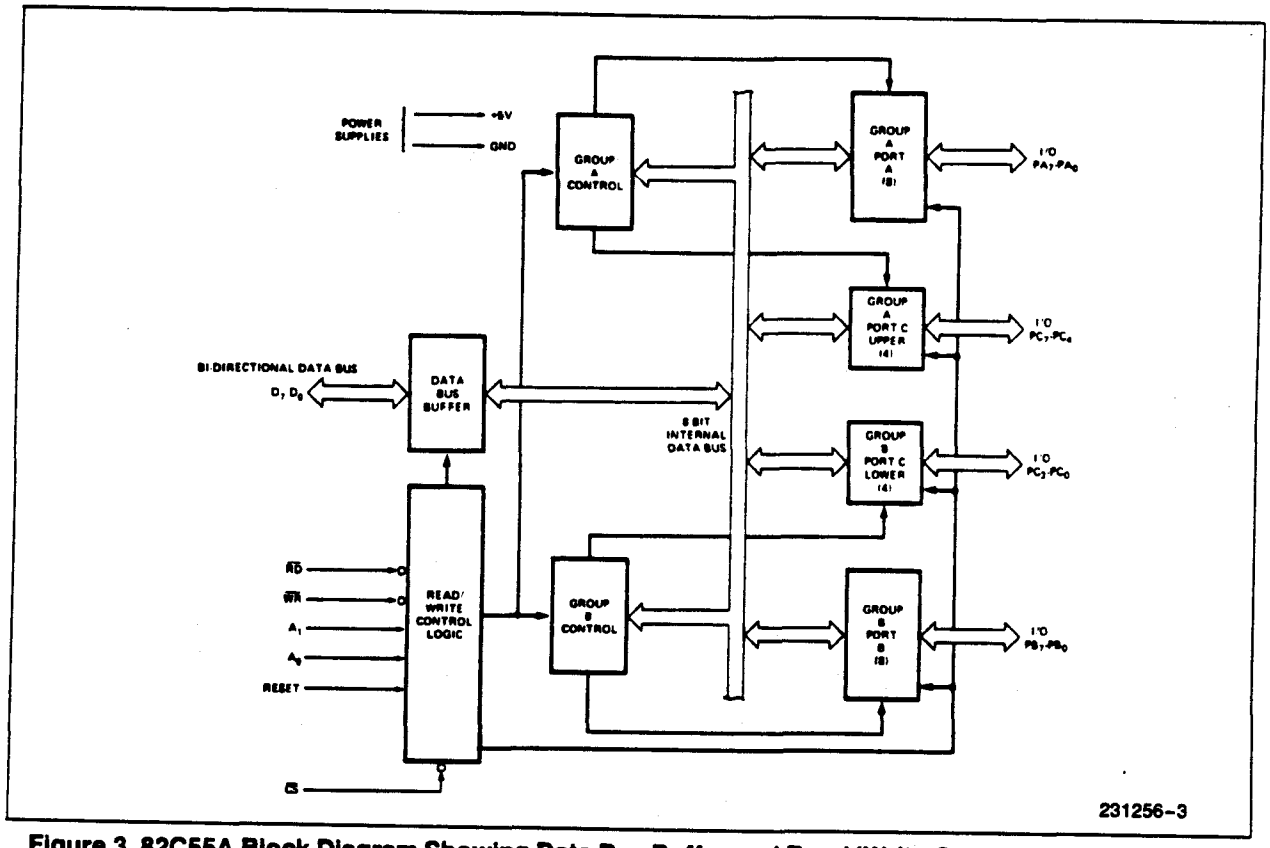
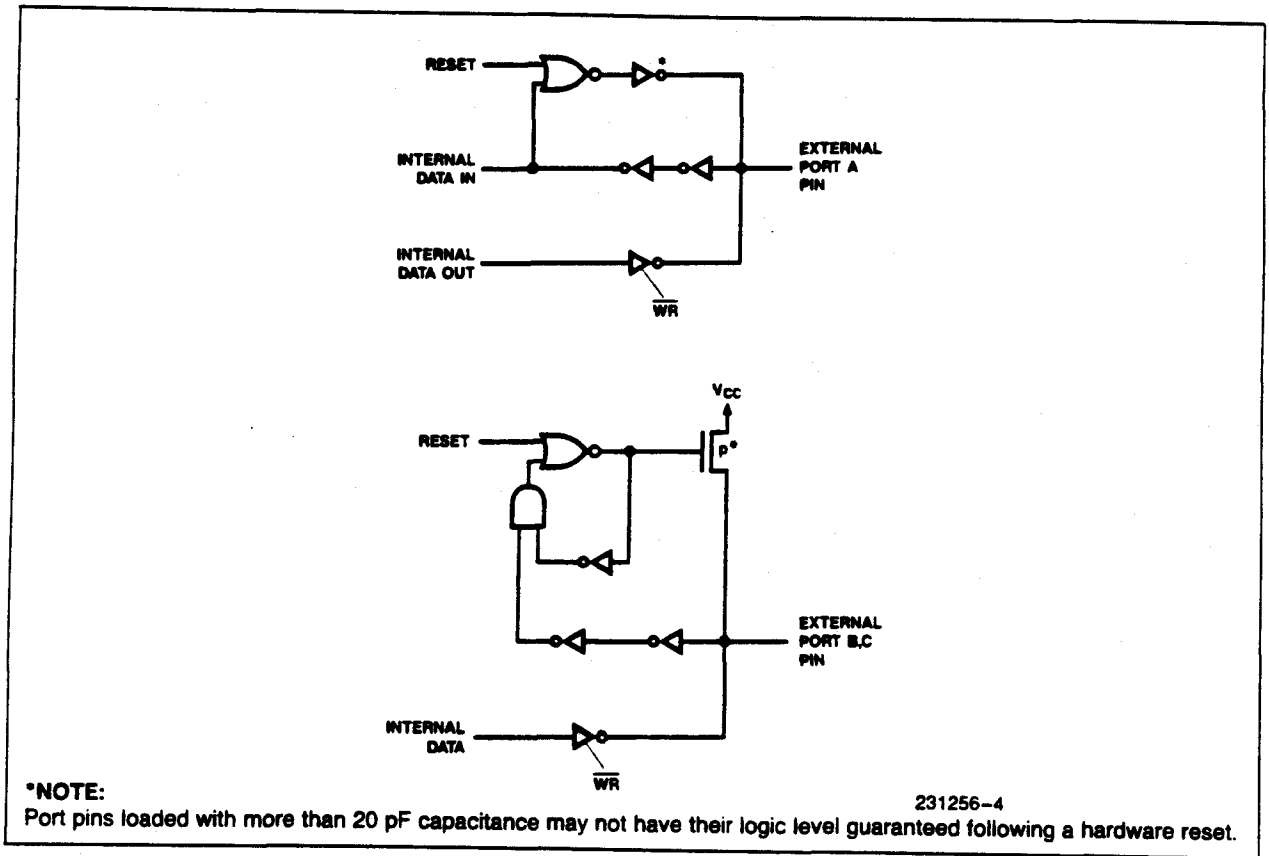


Figure 3. 82C55A Block Diagram Showing Data Bus Buffer and Read/Write Control Logic Functions



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

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

82C55A OPERATIONAL DESCRIPTION

Mode Selection

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

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

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

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

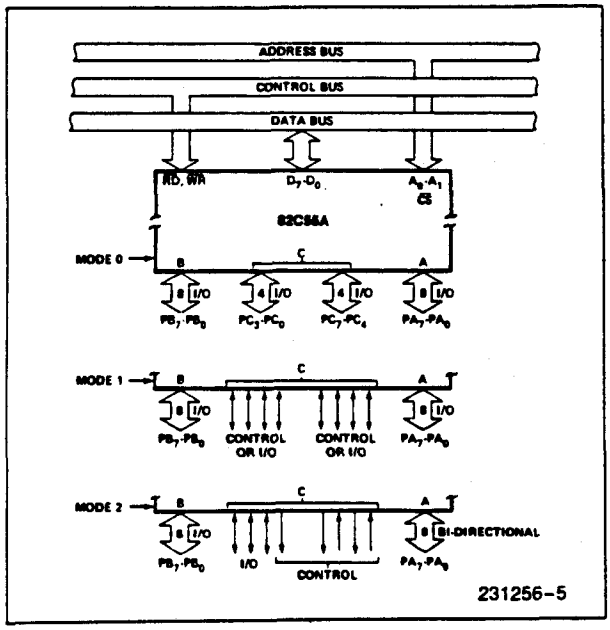


Figure 5. Basic Mode Definitions and Bus Interface

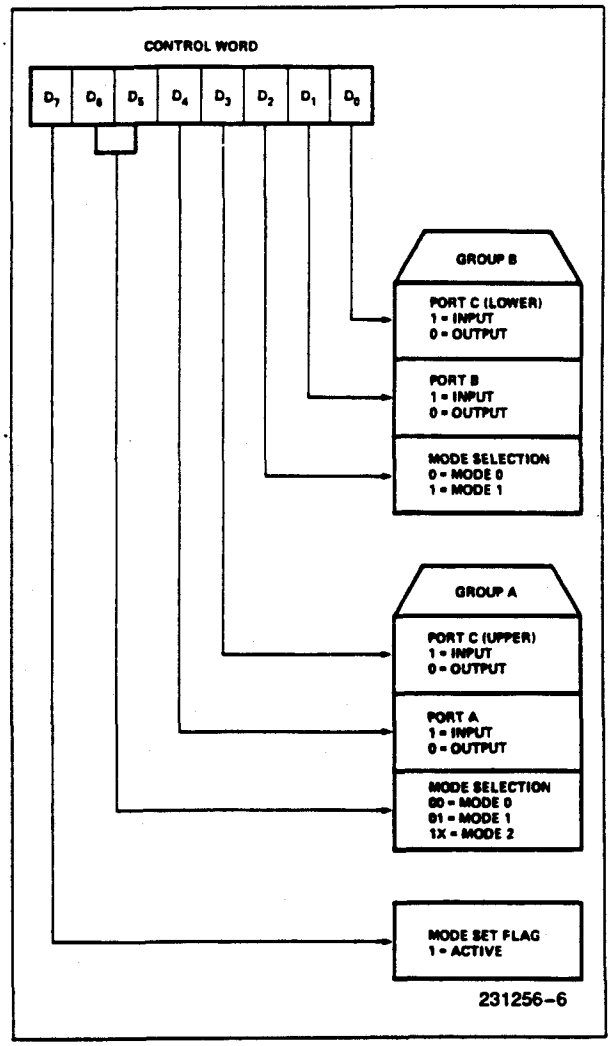


Figure 6. Mode Definition Format

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

Single Bit Set/Reset Feature

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

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

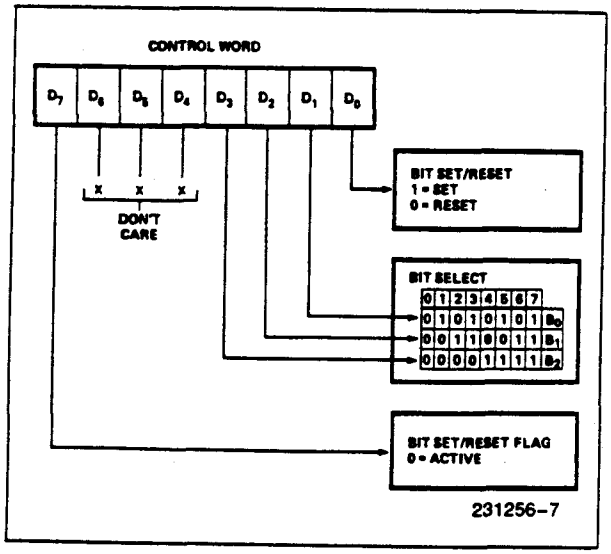


Figure 7. Bit Set/Reset Format

Interrupt Control Functions

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

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

INTE flip-flop definition:

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

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

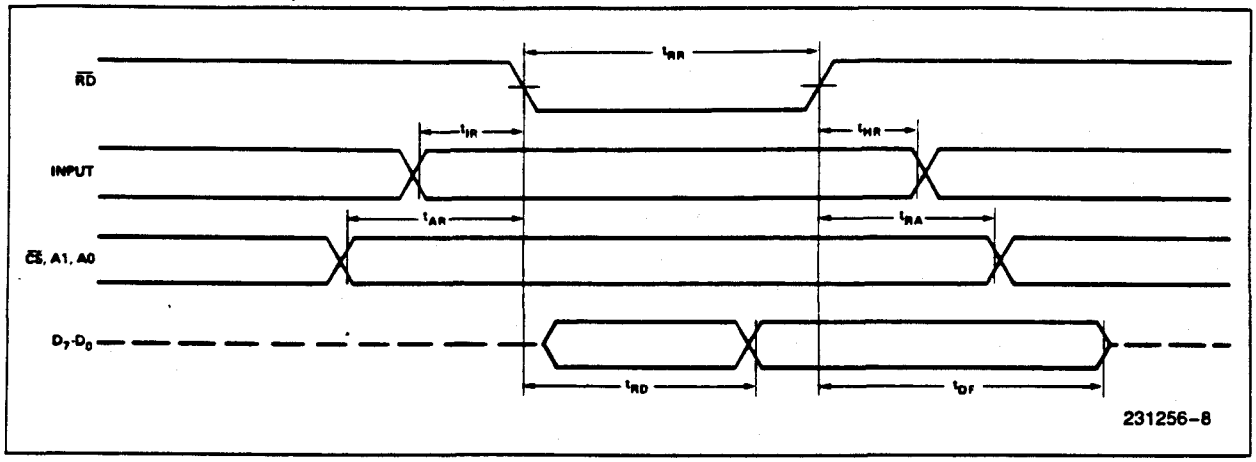
Operating Modes

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

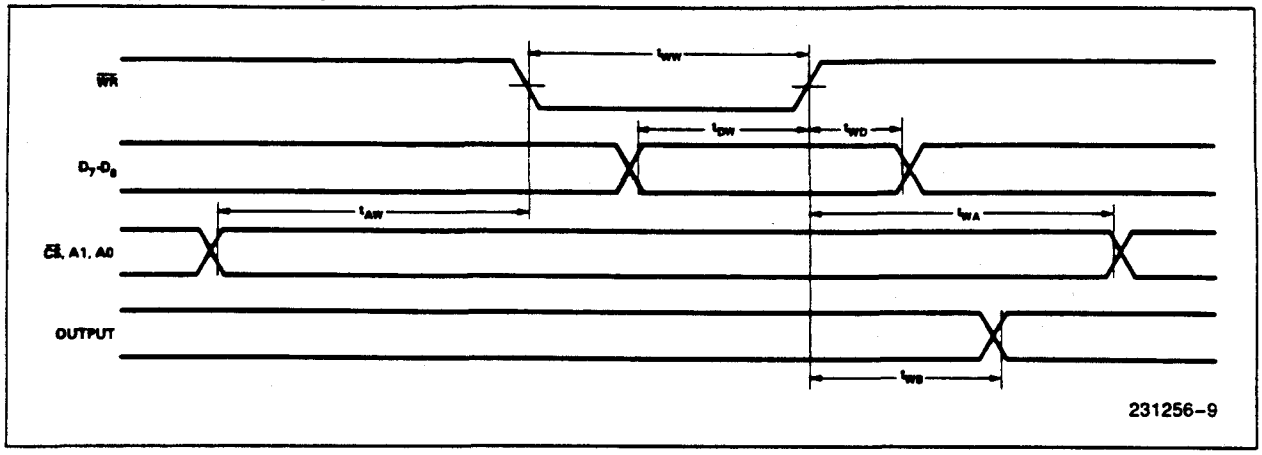
Mode 0 Basic Functional Definitions:

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

MODE 0 (BASIC INPUT)



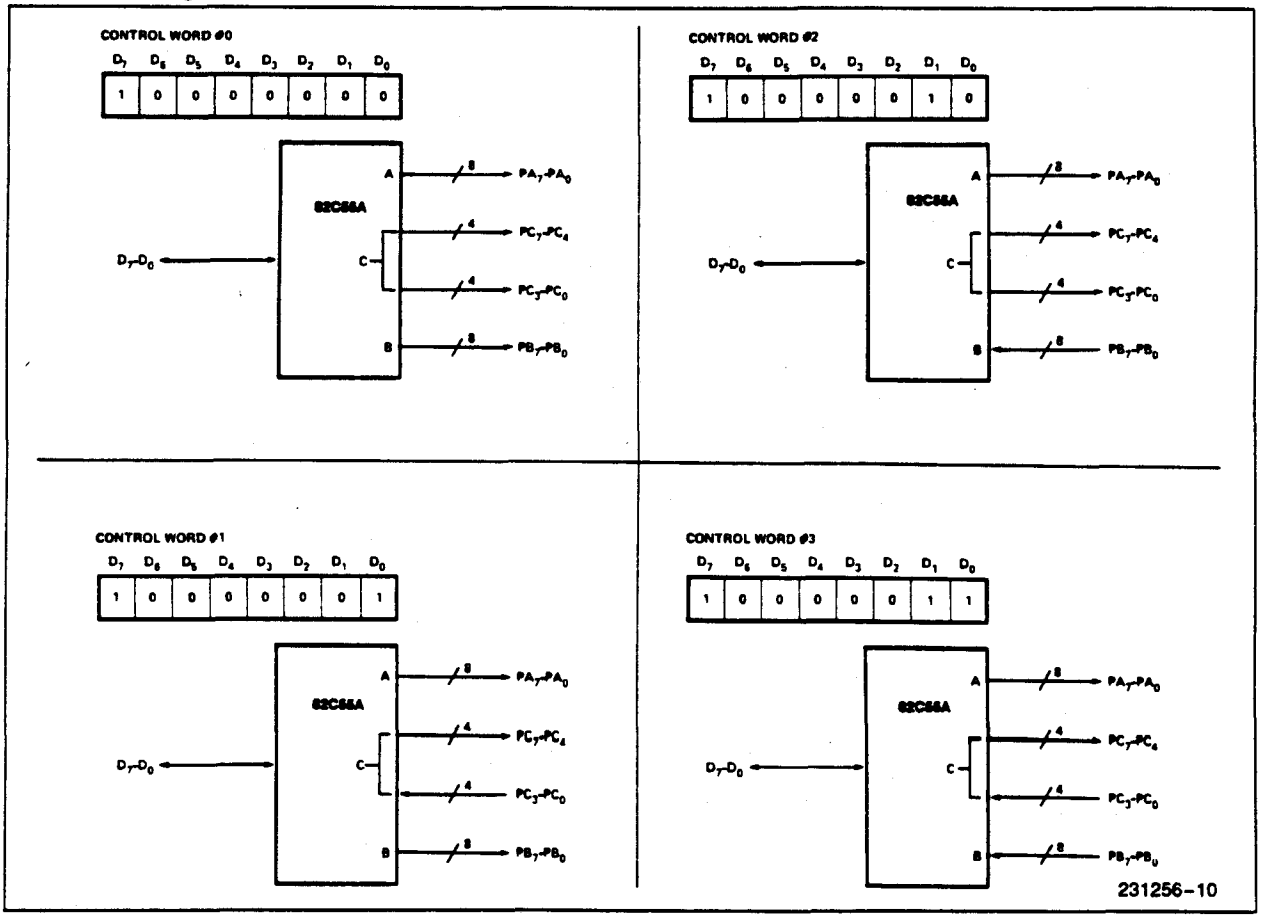
MODE 0 (BASIC OUTPUT)



MODE 0 Port Definition

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

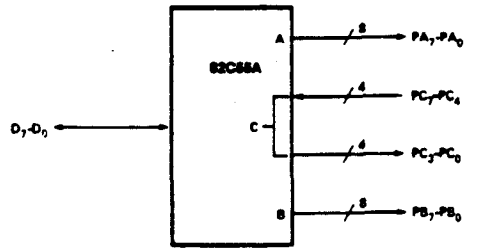
MODE 0 Configurations



MODE 0 Configurations (Continued)

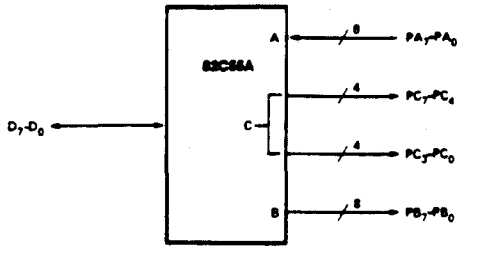
CONTROL WORD #4

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	0	0



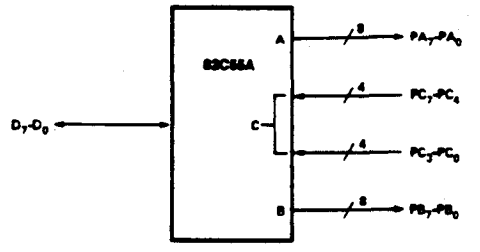
CONTROL WORD #8

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	0	0



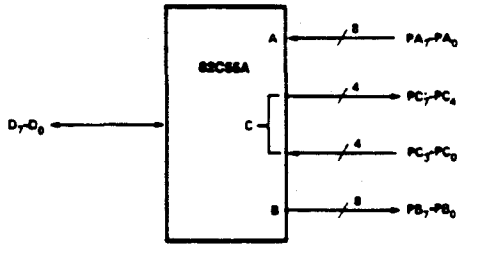
CONTROL WORD #5

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	0	1



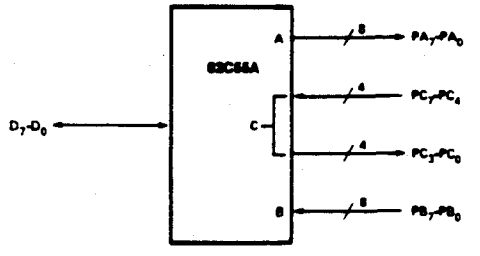
CONTROL WORD #9

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	0	1



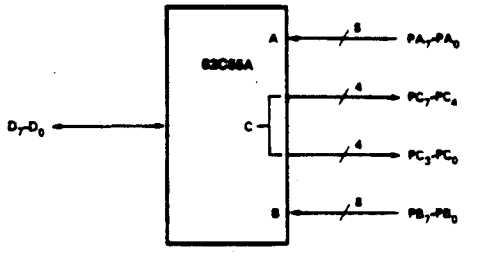
CONTROL WORD #6

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	1	0



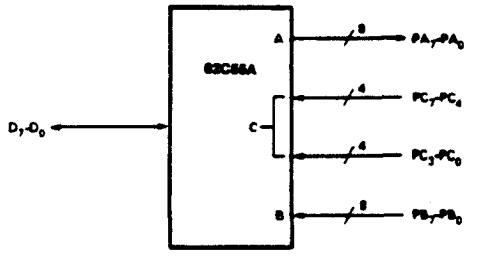
CONTROL WORD #10

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	1	0



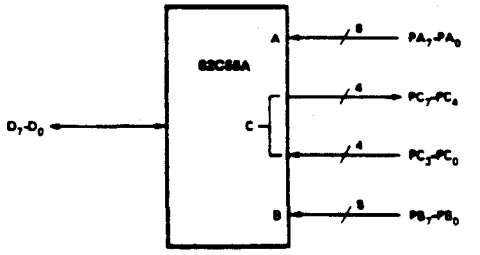
CONTROL WORD #7

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	1	1



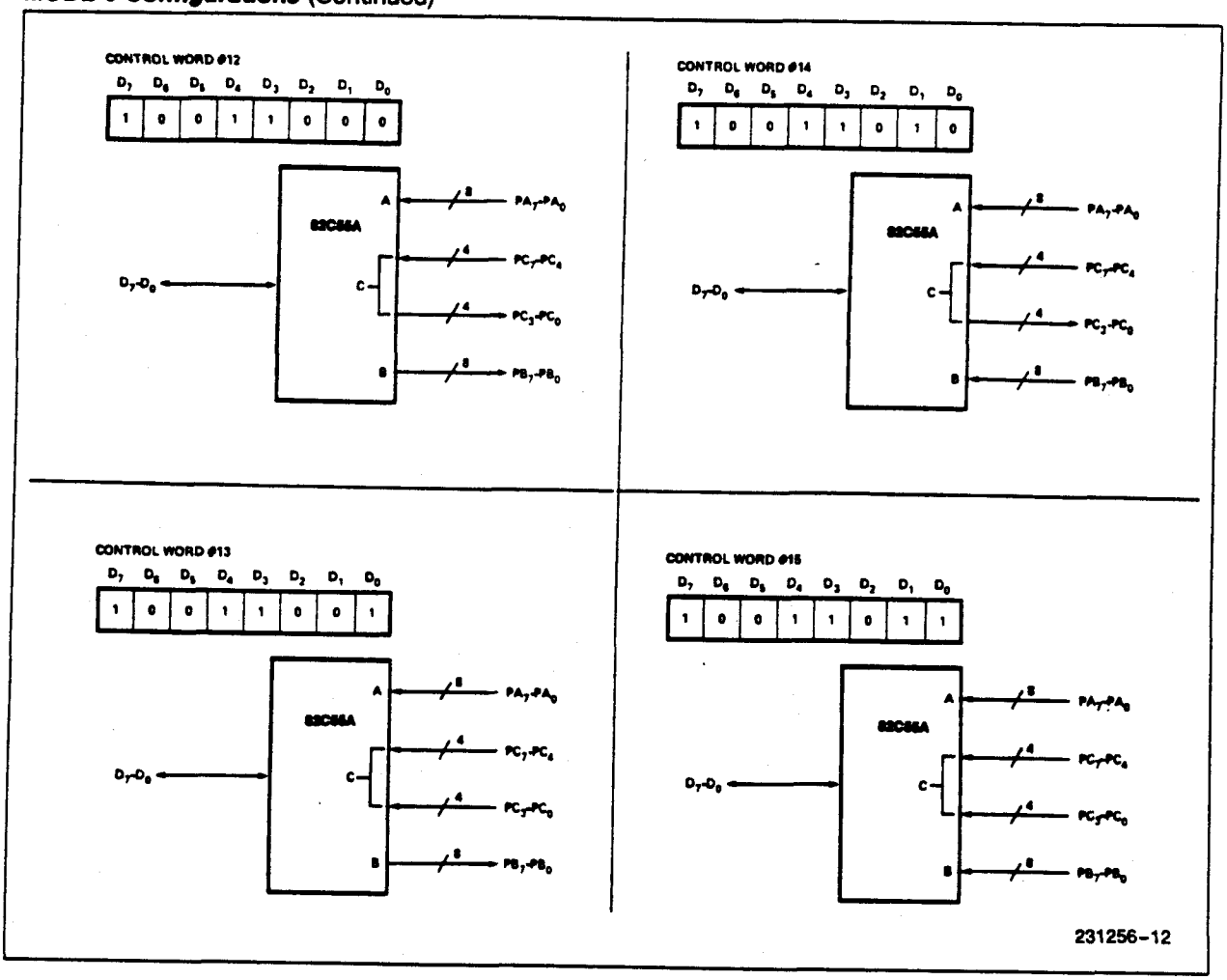
CONTROL WORD #11

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)



Operating Modes

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

Mode 1 Basic functional Definitions:

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

Input Control Signal Definition

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

IBF (Input Buffer Full F/F)

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

INTR (Interrupt Request)

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

INTE A

Controlled by bit set/reset of PC_4.

INTE B

Controlled by bit set/reset of PC_2.

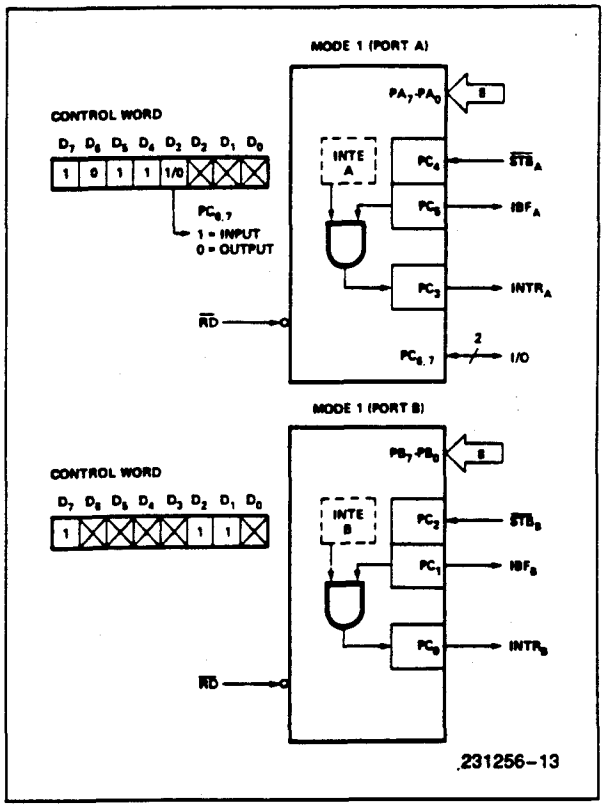


Figure 8. MODE 1 Input

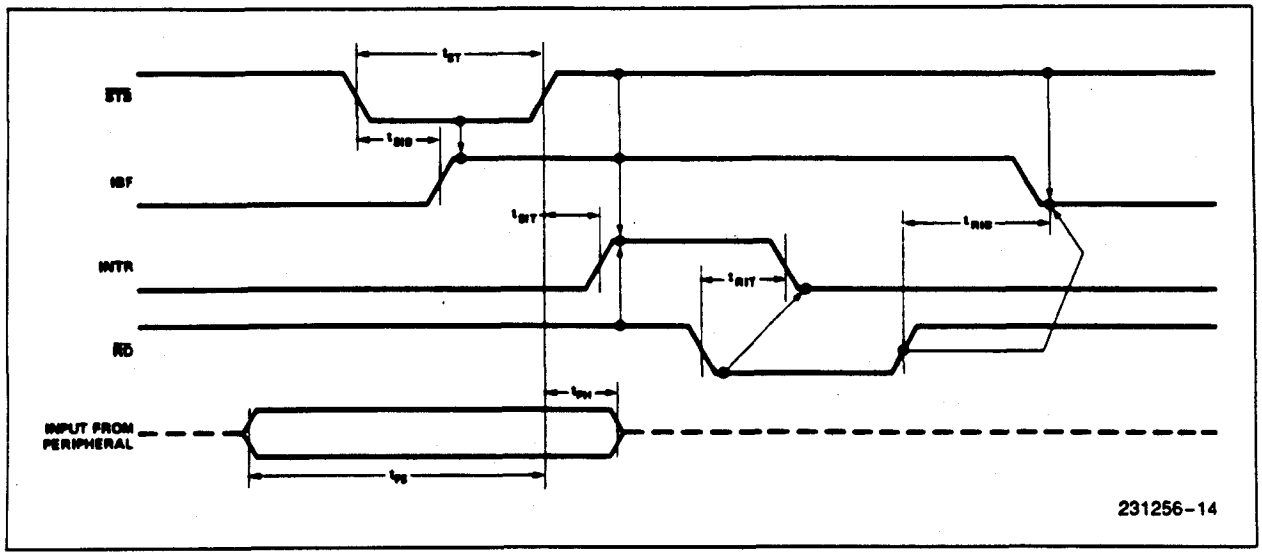


Figure 9. MODE 1 (Strobed Input)

Output Control Signal Definition

$\overline{OBF}$ (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.

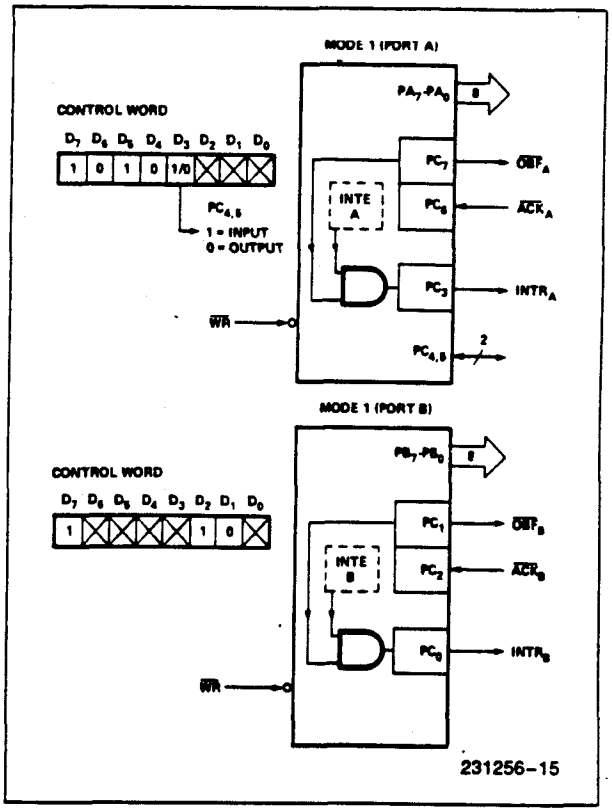


Figure 10. MODE 1 Output

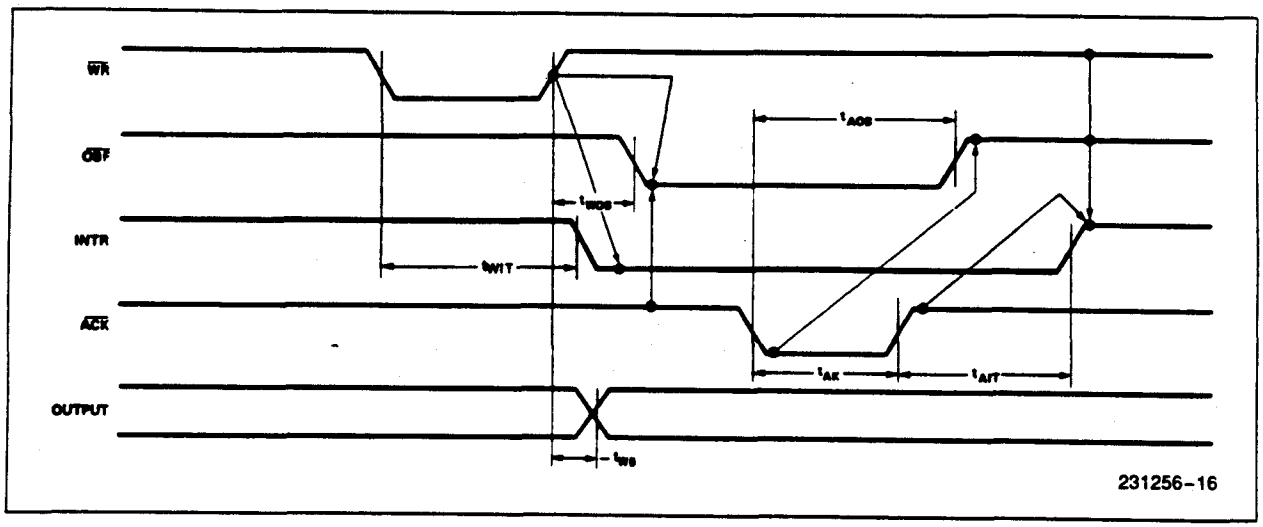


Figure 11. MODE 1 (Strobed Output)

Combinations of MODE 1

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

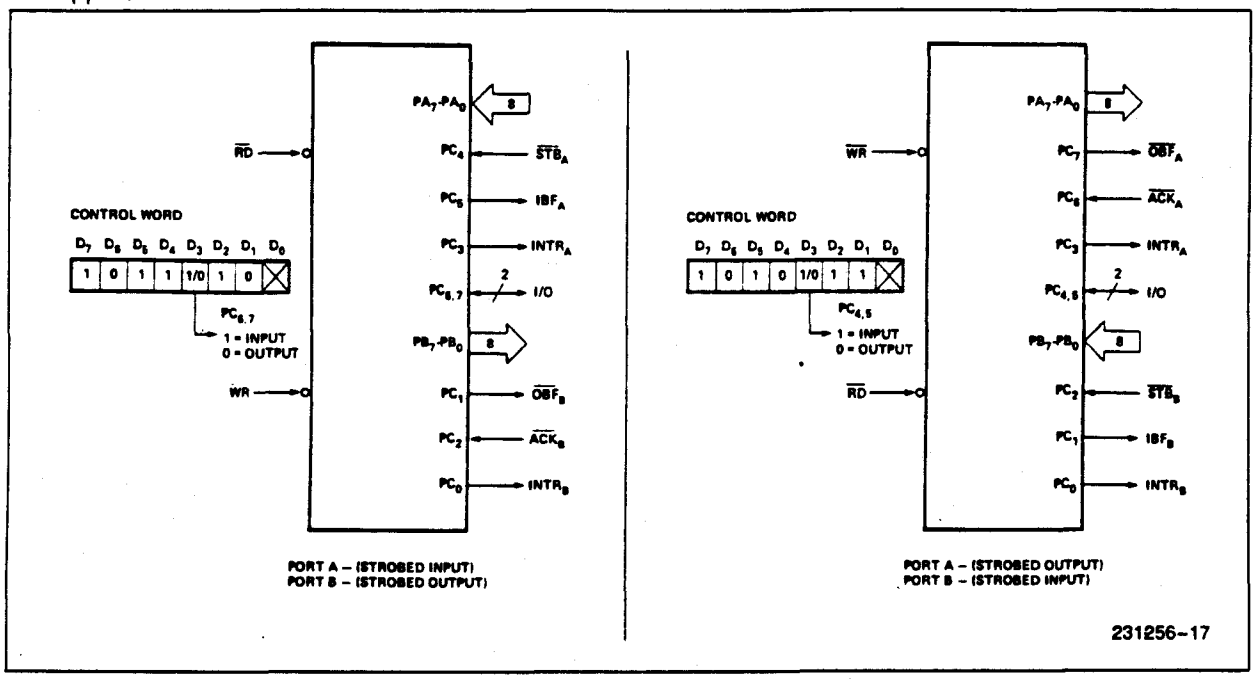


Figure 12. Combinations of MODE 1

Operating Modes

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

MODE 2 Basic Functional Definitions:

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

Bidirectional Bus I/O Control Signal Definition

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

Output Operations

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

$\overline{ACK}$ (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 $\overline{OBF}$). Controlled by bit set/reset of PC_6.

Input Operations

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

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

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

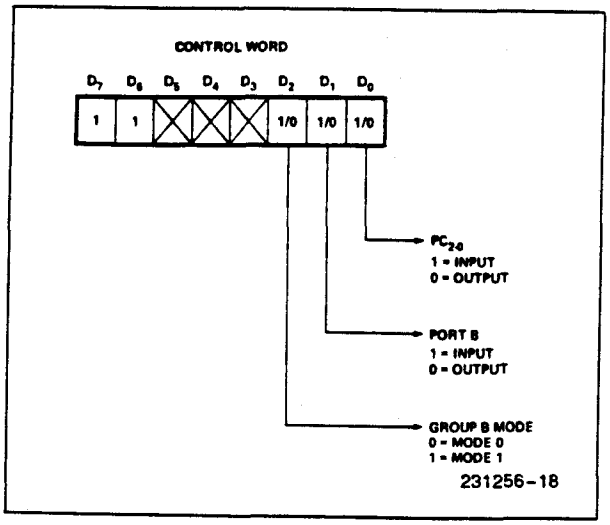


Figure 13. MODE Control Word

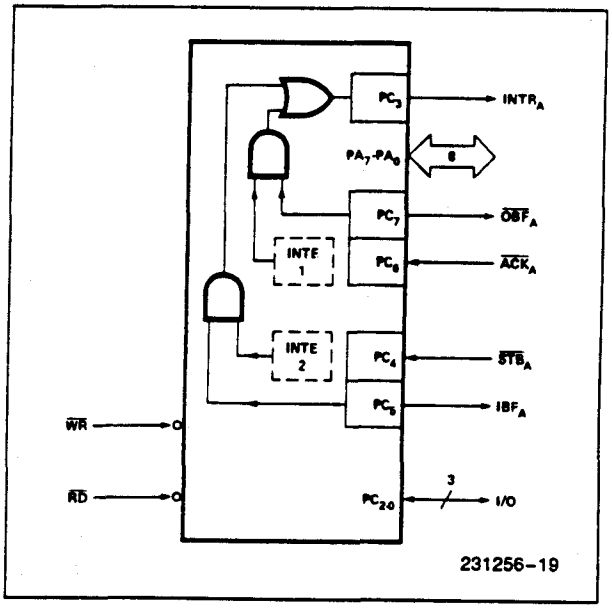


Figure 14. MODE 2

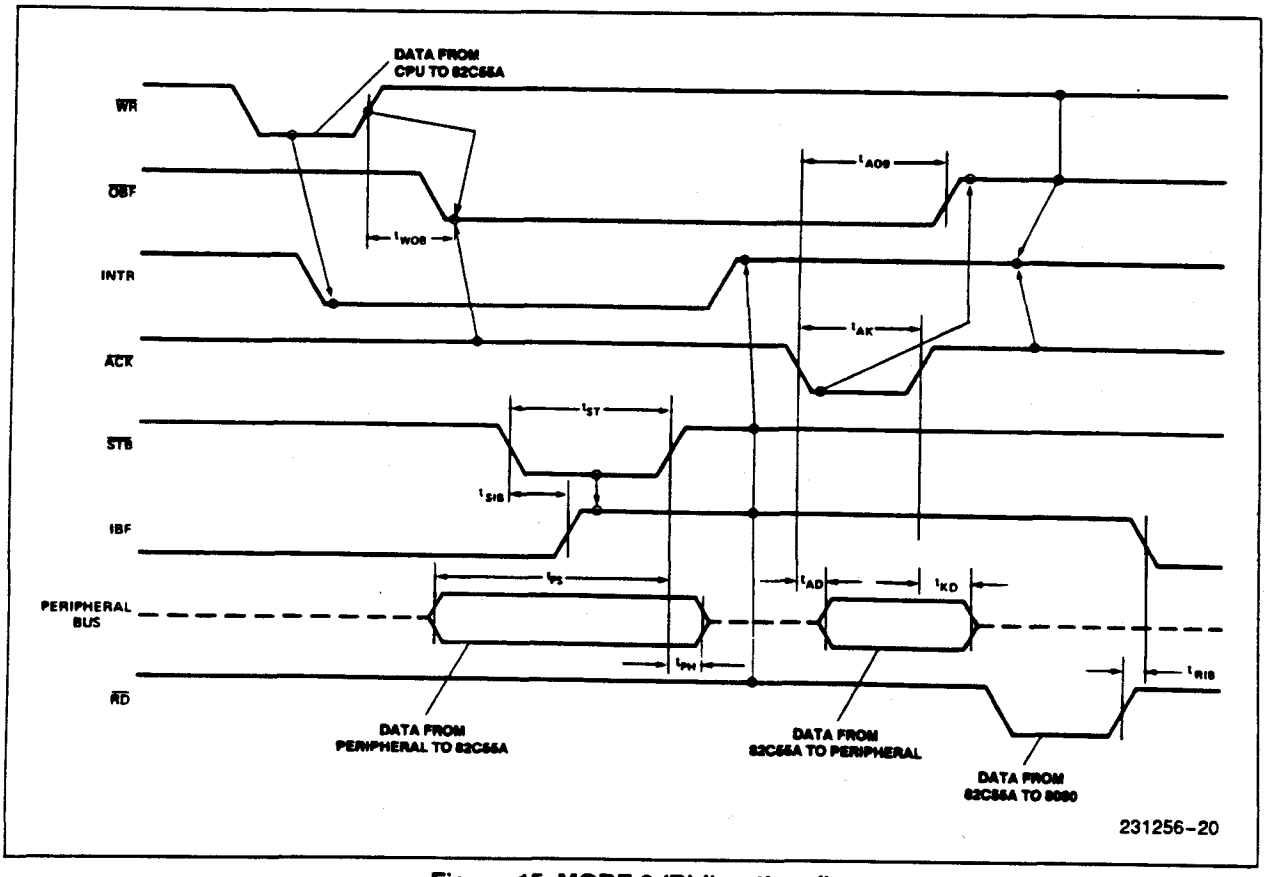


Figure 15. MODE 2 (Bidirectional)

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

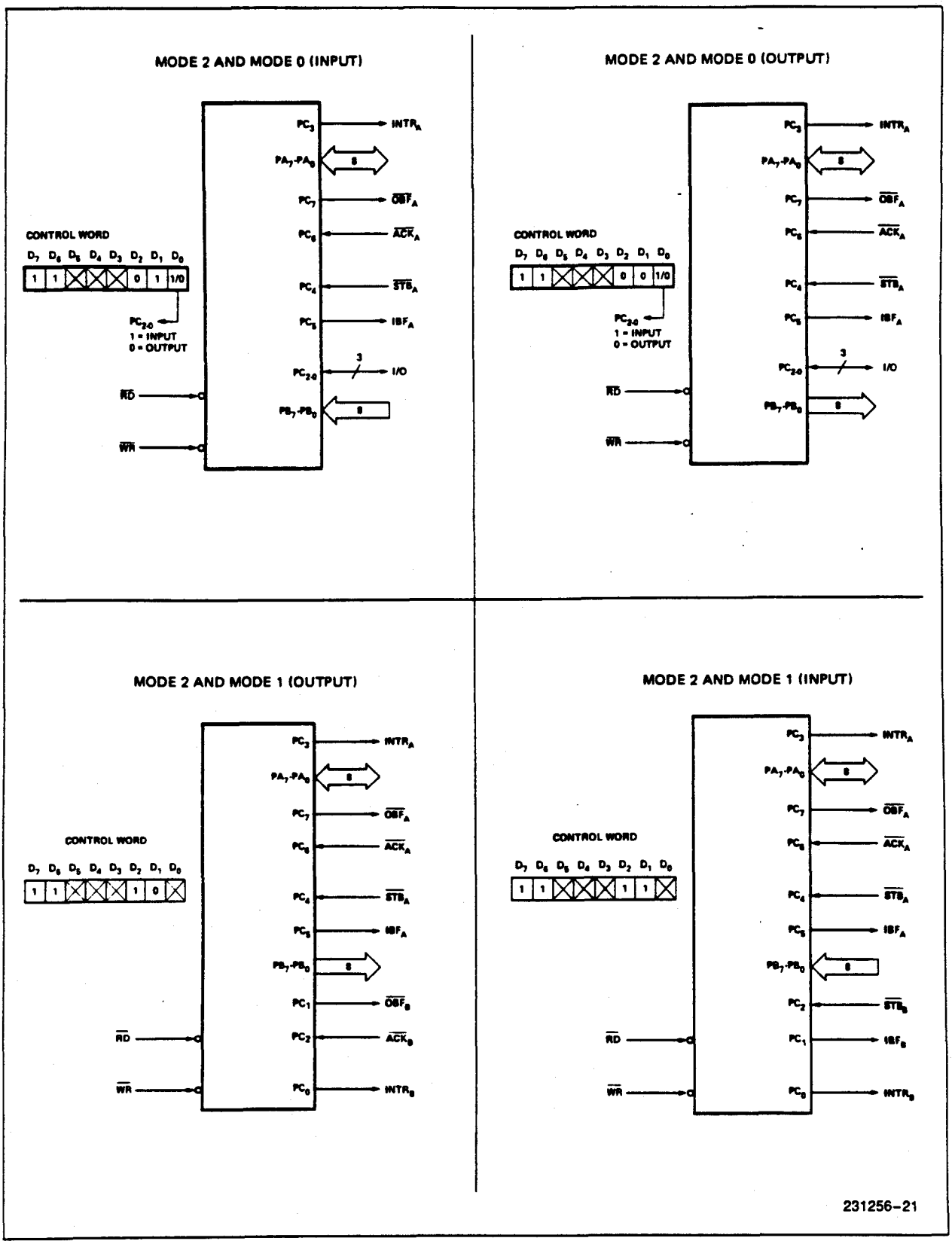


Figure 16. MODE ¼ Combinations

Mode Definition Summary

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

Special Mode Combination Considerations

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

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

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

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

Current Drive Capability

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

Reading Port C Status

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

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

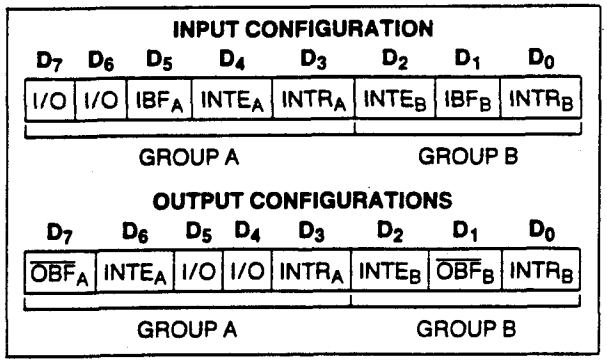


Figure 17a. MODE 1 Status Word Format

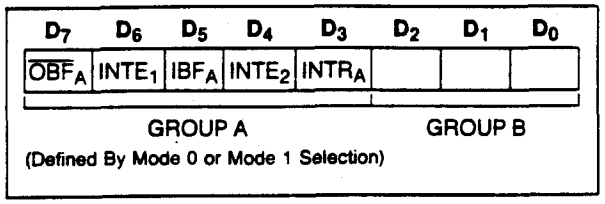


Figure 17b. MODE 2 Status Word Format

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

Figure 18. Interrupt Enable Flags in Modes 1 and 2

ABSOLUTE MAXIMUM RATINGS*

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

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

D.C. CHARACTERISTICS

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

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

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

CAPACITANCE

T_A = 25°C, V_{CC} = GND = 0V

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

NOTE:
5. Sampled not 100% tested.

A.C. CHARACTERISTICS

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

T_A = −40°C to +85°C for Extended Temperature

BUS PARAMETERS

READ CYCLE

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

WRITE CYCLE

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

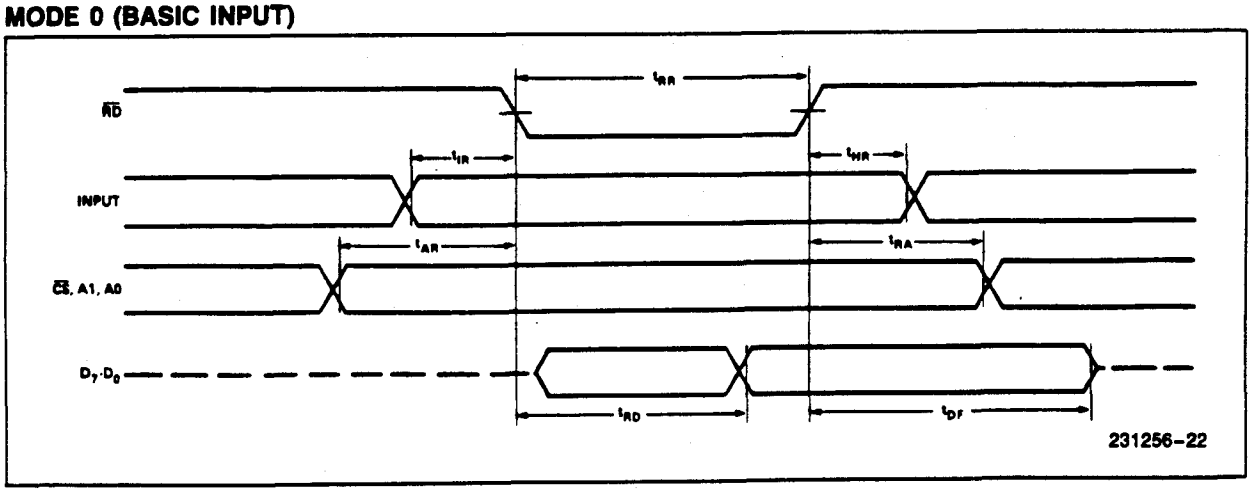
OTHER TIMINGS

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

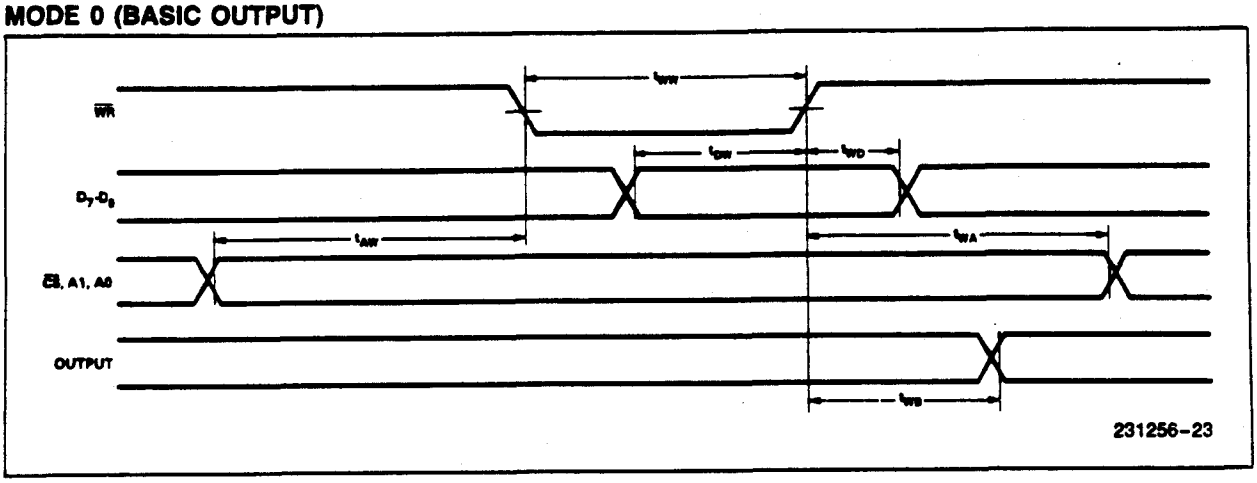
NOTE:
1. INTR ↑ may occur as early as $\overline{WR}$ ↓.
2. Pulse width of initial Reset pulse after power on must be at least 50 μSec. Subsequent Reset pulses may be 500 ns minimum.

WAVEFORMS

MODE 0 (BASIC INPUT)

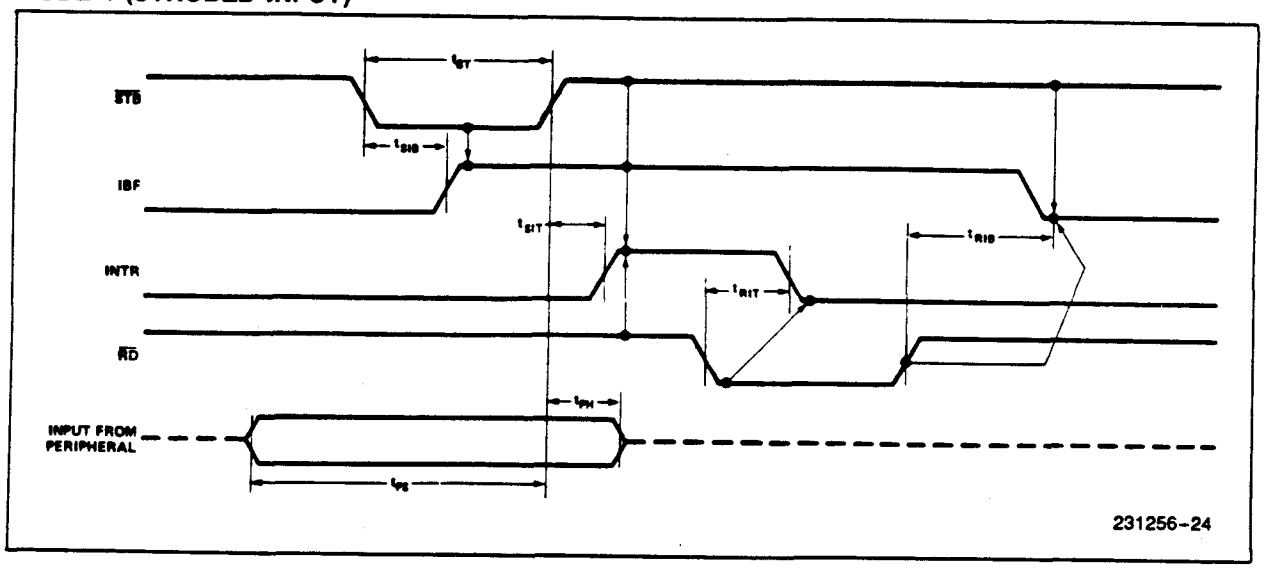


MODE 0 (BASIC OUTPUT)

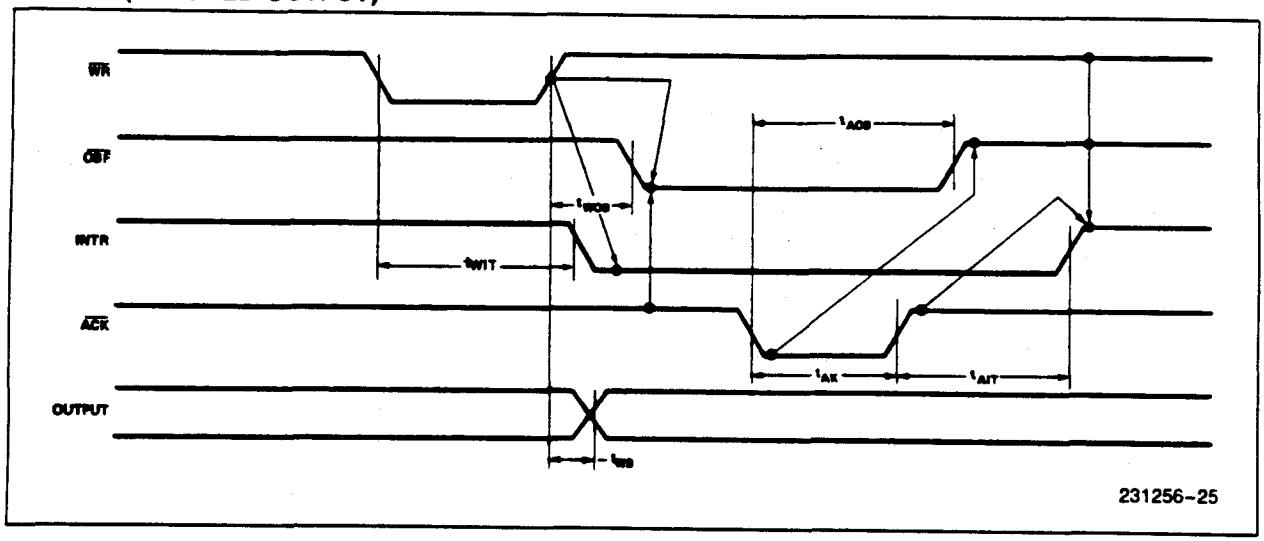


WAVEFORMS (Continued)

MODE 1 (STROBED INPUT)

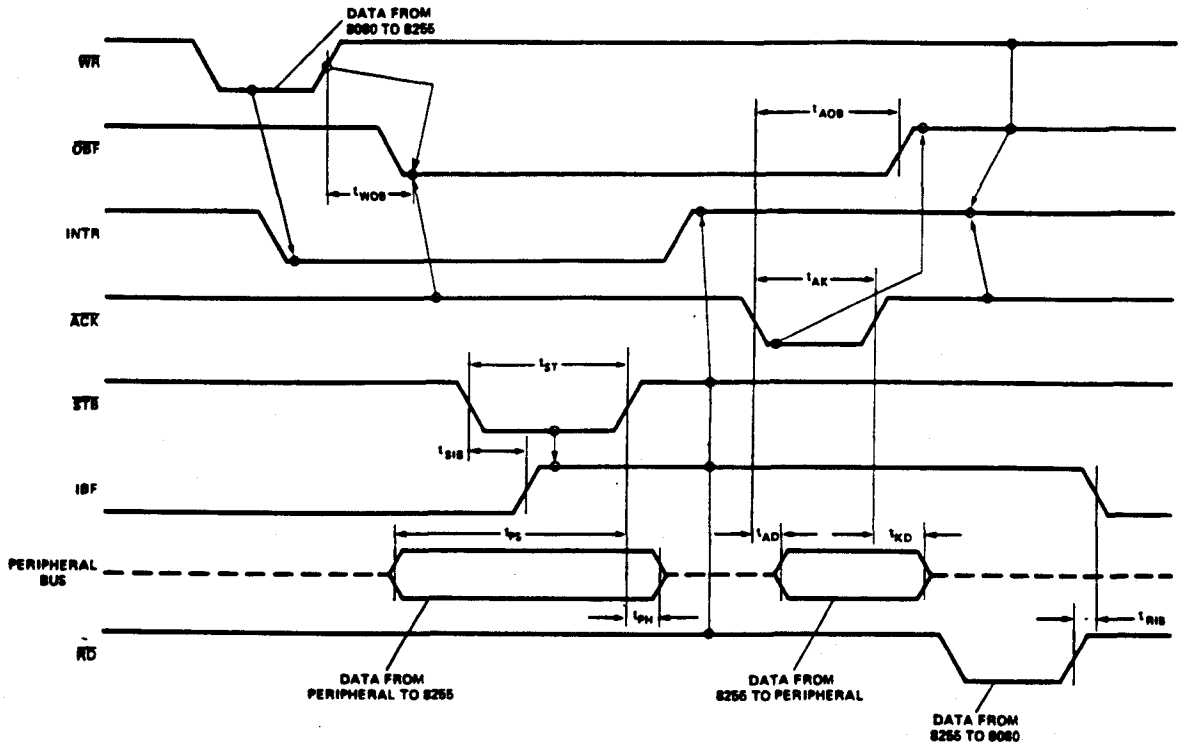


MODE 1 (STROBED OUTPUT)



WAVEFORMS (Continued)

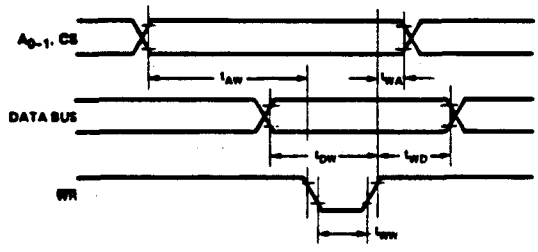
MODE 2 (BIDIRECTIONAL)



231256-26

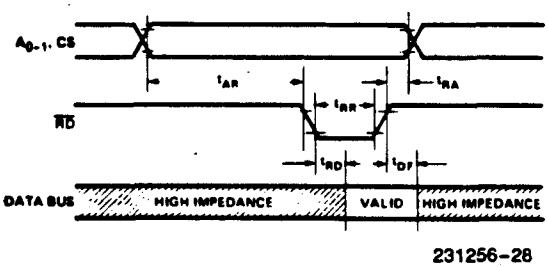
Note:
Any sequence where $\overline{WR}$ occurs before $\overline{ACK}$ AND $\overline{STB}$ occurs before $\overline{RD}$ is permissible.
($INTR = IBF \bullet \overline{MASK} \bullet \overline{STB} \bullet \overline{RD} + \overline{OBF} \bullet \overline{MASK} \bullet \overline{ACK} \bullet \overline{WR}$)

WRITE TIMING



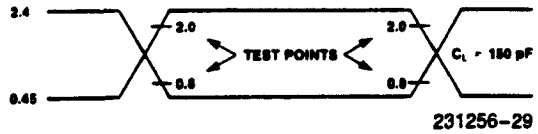
231256-27

READ TIMING



231256-28

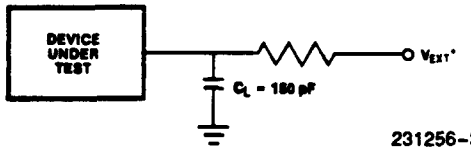
A.C. TESTING INPUT, OUTPUT WAVEFORM



231256-29

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

A.C. TESTING LOAD CIRCUIT



231256-30

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

APPENDIX D

WARRANTY

LIMITED WARRANTY

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

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

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

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

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

DG24 Board User-Selected Settings	
Base I/O Address:	
(hex)	(decimal)
Interrupt Channel Selection:	
8255 PC3	IRQ Channel:
8255 PC0	IRQ Channel:
EXTINT	IRQ Channel: