

Lecture FIVE

Programmable Peripheral

Interface (PPI)

REAL-TIME SYSTEMS

THE PROGRAMMABLE PERIPHERAL INTERFACE

The 82C55 **programmable peripheral interface** (PPI) is a very popular, low-cost interfacing component found in many applications. This is true even with all the programmable devices available for simple applications. The PPI, which has 24 pins for I/O that are programmable in groups of 12 pins, has groups that operate in three distinct modes of operation. The 82C55 can interface any TTL-compatible I/O device to the microprocessor. The 82C55 (CMOS version) requires the insertion of wait states if operated with a microprocessor using higher than an 8 MHz clock. It also provides at least 2.5 mA of sink (logic 0) current at each output, with a maximum of 4.0 mA. Because I/O devices are inherently slow, wait states used during I/O transfers do not impact significantly upon the speed of the system. The 82C55 still finds application (compatible for programming, although it may not appear in the system as a discrete 82C55), even in the latest Core2-based computer system. The modern computer uses a few 82C55s located inside the chip set for various features on the personal computer. The 82C55 is used for interface to the keyboard and the parallel printer port in many personal computers, but it is found as a function within a interfacing chip set. The chip set also controls the timer and reads data from the keyboard interface.

Basic Description of the 82C55

Its three I/O ports (labeled A, B, and C) are programmed as groups. Group A connections consist of port A (PA7–PA0) and the upper half of port C (PC7–PC4), and group B consists of port B (PB7–PB0) and the lower half of port C (PC3–PC0). The 82C55 is selected by its $\overline{CS}$ pin for programming and for reading or writing to a port. Register selection is accomplished through the A1 and A0 input pins that select an internal register for programming or operation.

4: 8255 PPI CHIP

In this section we study the 8255 chip, one of the most widely used I/O chips. The 8255 is a 40-pin DIP chip (see Figure 11-11). It has three separately accessible ports, A, B, and C, which can be programmed, hence the name PPI (programmable peripheral interface). Notice that the individual ports of the 8255 can be programmed to be input or output. They can also be changed dynamically, in contrast to the 74LS244 and 74LS373, which are hard-wired.

Port A (PA0 - PA7)

This 8-bit port A can be programmed all as input or all as output.

Port B (PB0 - PB7)

This 8-bit port B can be programmed all as input or all as output.

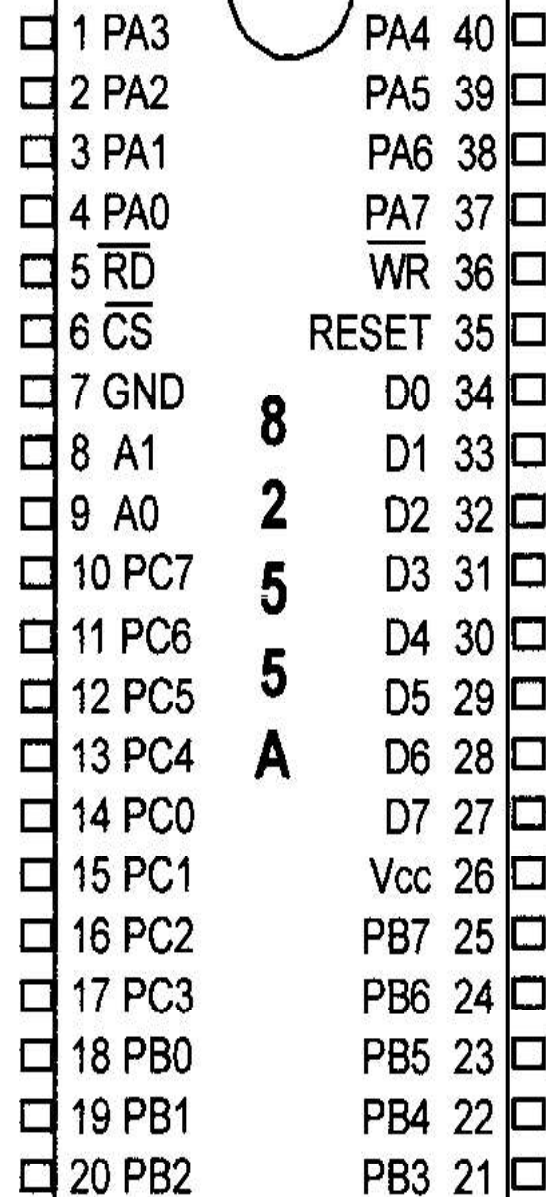


Figure 11-11. 8255 PPI Chip

(Reprinted by permission of Intel Corporation,

Port C (PC0 - PC7)

This 8-bit port C can be all input or all output. It can also be split into two parts, CU (upper bits PC4 - PC7) and CL (lower bits PC0 - PC3). Each can be used for input or output. Any of PC0 to PC7 can be programmed individually.

RD and WR

These two active-low control signals are inputs to the 8255. If the 8255 is using peripheral I/O design, IOR and IOW of the system bus are connected to these two pins. If the port uses memory-mapped I/O, MEMR and MEMW activate them.

RESET

This is an active-high signal input into the 8255 used to clear the control register. When RESET is activated, all ports are initialized as input ports. This pin must be connected to the RESET output of the system bus or ground, making it inactive. Like all IC input pins, it should not be left unconnected.

This pin can be grounded.

A0, A1, and CS

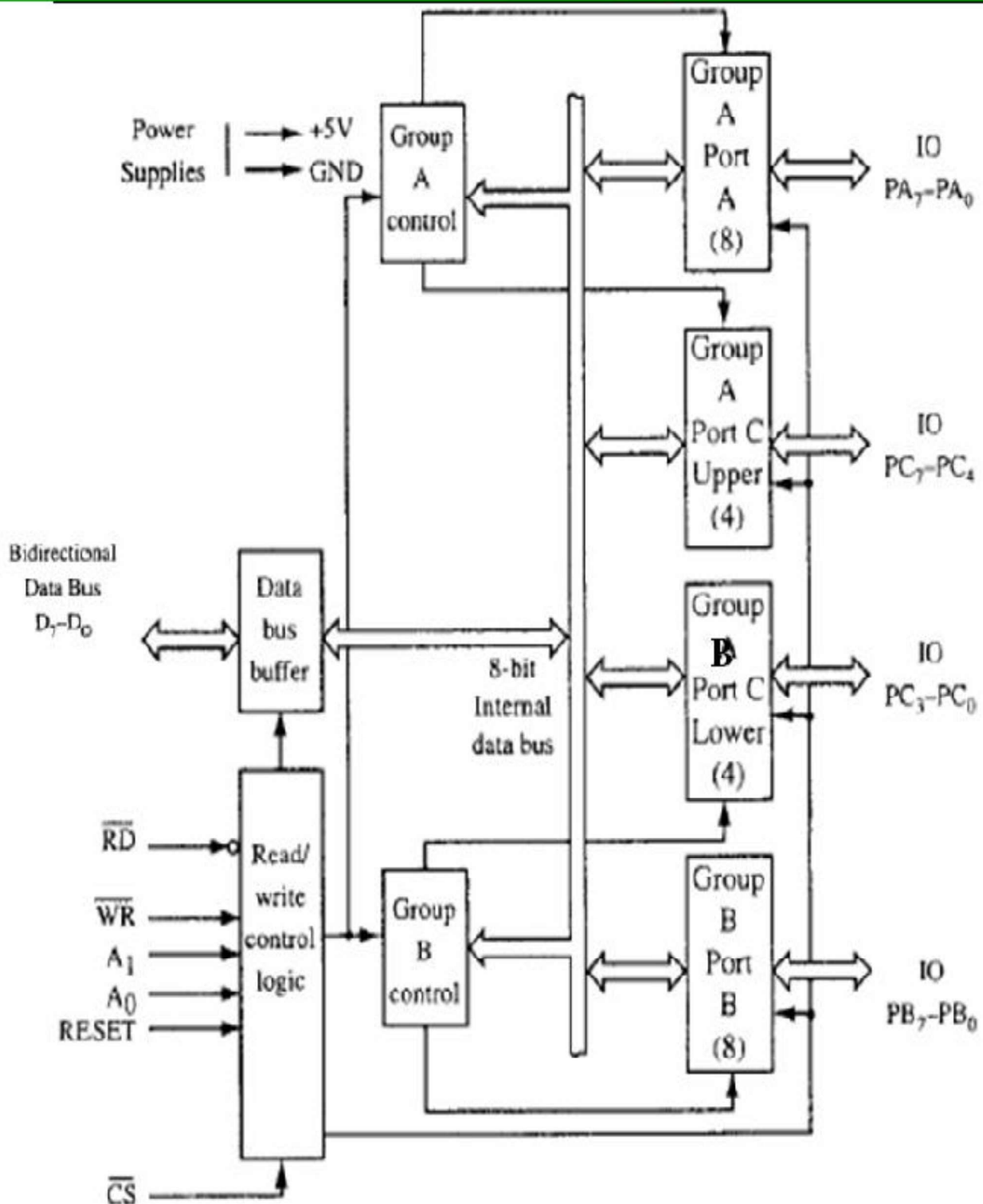
While CS (chip select) selects the entire chip, address pins A0 and A1 select the specific port within the 8255. These three pins are used to access ports A, B, C, or the control register, as shown in Table 11-3.

Table 11-3: 8255 Port Selection

CS	A1	A0	Selects
0	0	0	Port A
0	0	1	Port B
0	1	0	Port C
0	1	1	Control register
1	x	x	8255 is not selected

(Reprinted by permission of Intel Corporation, Copyright Intel Corp. 1983)

8255 Block Diagram



Mode selection of the 8255A

While ports A, B, and C are used for I/O data, it is the control register that must be programmed to select the operation mode of the three ports A, B, and C. The ports of the 8255 can be programmed in any of the following modes.

1. Mode 0, simple I/O mode. In this mode, any of the ports A, B, CL, and CU can be programmed as input or output. In this mode, all bits are out or all are in. In other words, there is no control of individual bits. Mode 0 is the most widely used mode in current system I/O interfacing design. For the rest of this and the next chapter we use only this simple mode.
2. Mode 1. In this mode, ports A and B can be used as input or output ports with handshaking capabilities. This mode is not used due to timing incompatibility with devices such as the printer.
3. Mode 2. In this mode, port A can be used as a bidirectional I/O port with handshaking capabilities. This mode is rarely used.

Notice from Figure 11-12 that we must set $D7 = 1$ to get the above I/O modes of 0, 1, and 2. If $D7 = 0$, we have BSR mode. In BSR (bit set/reset) mode, the bits of port C are programmed individually. This mode is also rarely used.

The 8255 chip is programmed in any of the above modes by sending a byte (Intel calls it control word) to the control register of the 8255. For example, to make ports A, B, and C output ports, we make $D7 = 1$ according to Figure 11-12. To select simple I/O mode of 0, we need 1000 0000 as the control word. Similarly, to get PB as input, and PA and all of PC as output, we must have 1000 0010 or 82H for the control word. Examples 11-4, 11-5, and 11-6 demonstrate how the 8255 chip is programmed. Study these three examples in detail since they will be the basis of many lab assignments.

Mode 0: Simple Input or Output

In this mode, ports A, B are used as two simple 8-bit I/O ports
port C as two 4-bit ports.

Each port can be programmed to function as simply an input port or
an output port. The input/output features in Mode 0 are as follows.

1. Outputs are latched.
2. Inputs are not latched.
3. Ports don't have handshake or interrupt capability.

Mode 1: Input or Output with Handshake

In this mode, handshake signals are exchanged between
the MPU and peripherals prior to data transfer.

The features of the mode include the following:

1. Two ports (A and B) function as 8-bit I/O ports.
They can be configured as either as input or output ports.
2. Each port uses three lines from port C as handshake signals.
The remaining two lines of Port C can be used for simple I/O operations.
3. Input and Output data are latched.
4. Interrupt logic is supported.

Mode 2: Bidirectional Data Transfer

This mode is used primarily in applications such as data transfer between two computers.

In this mode, Port A can be configured as the bidirectional port Port B either in Mode 0 or Mode 1.

Port A uses five signals from Port C as handshake signals for data transfer.

The remaining three signals from port C can be used either as simple I/O or as handshake for port B.

There are 2 control words in 8255

Mode Definition (MD) Control word and

Port C Bit Set / Reset (PCBSR) Control Word

MD control word configures the ports of 8255

- as i/p or o/p in Mode 0, 1, or 2

PCBSR control word is used to set to 1 or reset to 0 any one selected bit of Port C

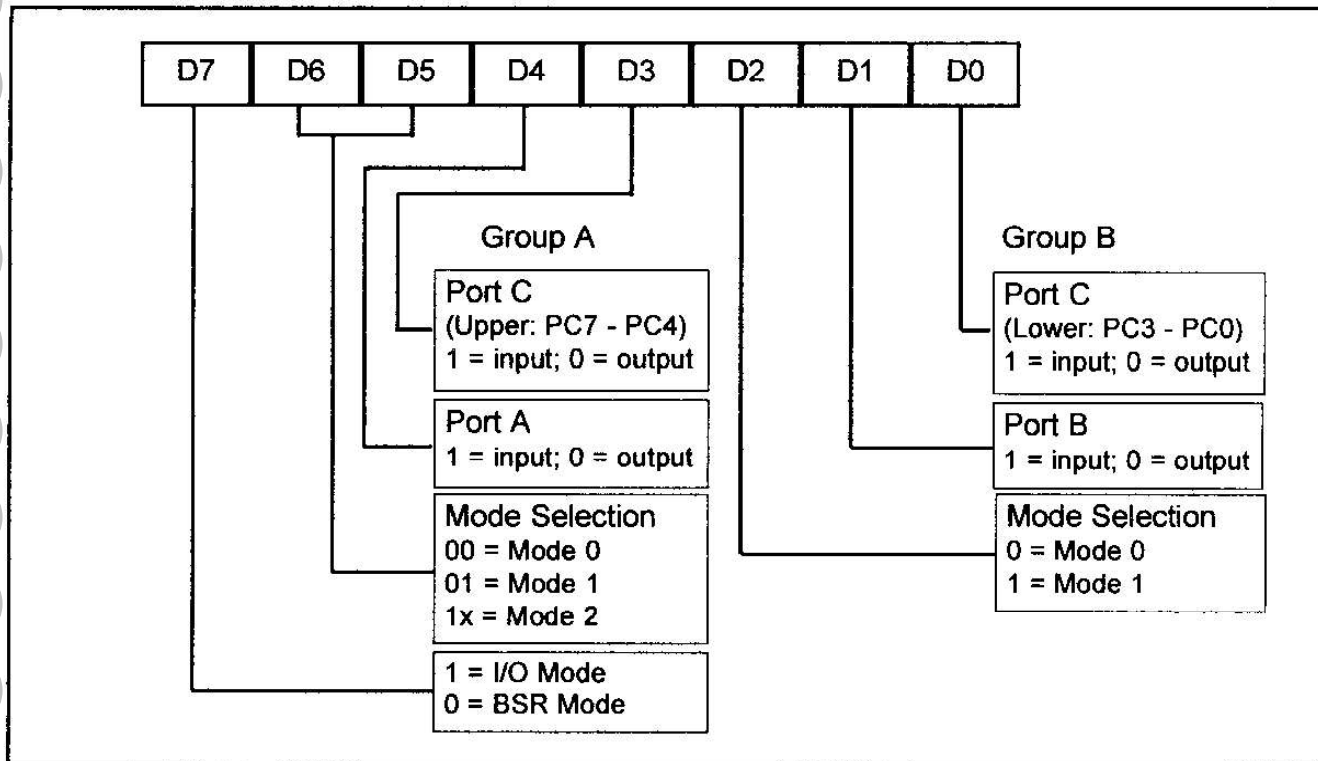


Figure 11-12. 8255 Control Word Format (I/O Mode)

Example 11-4

- (a) Find the control word if PA = out, PB = in, PC0 - PC3 = in, and PC4 - PC7 = out.
 (b) Program the 8255 to get data from port A and send it to port B. In addition, data from PCL is sent out to the PCU.
 Use port addresses of 300H - 303H for the 8255 chip.

Solution:

(a) From Figure 11-12 we get the control word of 1000 0011 in binary or 83H.

(b) The code is as follows:

```

B8255C EQU 300H ;Base address of 8255 chip
CNTL EQU 83H ;PA=out,PB=in,PCL=in,PCU=out

MOV DX,B8255C+3 ;load control reg. address (300H+3=303H)
MOV AL,CNTL ;load control byte
OUT DX,AL ;send it to control register
MOV DX,B8255C+1 ;load PB address
IN AL,DX ;get the data from PB
MOV DX,B8255C ;load PA address
OUT DX,AL ;send it to PA
MOV DX,B8255C+2 ;load PC address
IN AL,DX ;get the bits from PCL
AND AL,0F1H ;mask the upper bits
ROL AL,1
ROL AL,1 ;shift the bits
ROL AL,1 ;to upper position
ROL AL,1
OUT DX,AL ;send it to PCU
  
```

Example 11-5

The 8255 shown in Figure 11-13 is configured as follows: port A as input, B as output, and all the bits of port C as output.

- Find the port addresses assigned to A, B, C, and the control register.
- Find the control byte (word) for this configuration.
- Program the ports to input data from port A and send it to both ports B and C.

Solution:

- The port addresses are as follows:

$\overline{CS}$	A1	A0	Address	Port
11 0001 00	0	0	310H	Port A
11 0001 00	0	1	311H	Port B
11 0001 00	1	0	312H	Port C
11 0001 00	1	1	313H	Control register

- The control word is 90H, or 1001 0000.

- One version of the program is as follows:

```

MOV AL,90H          ;control byte PA=in, PB=out, PC=out
MOV DX,313H         ;load control reg address
OUT DX,AL           ;send it to control register
MOV DX,310H         ;load PA address
IN AL,DX            ;get the data from PA
MOV DX,311H         ;load PB address
OUT DX,AL           ;send it to PB
MOV DX,312H         ;load PC address
OUT DX,AL           ;and to PC

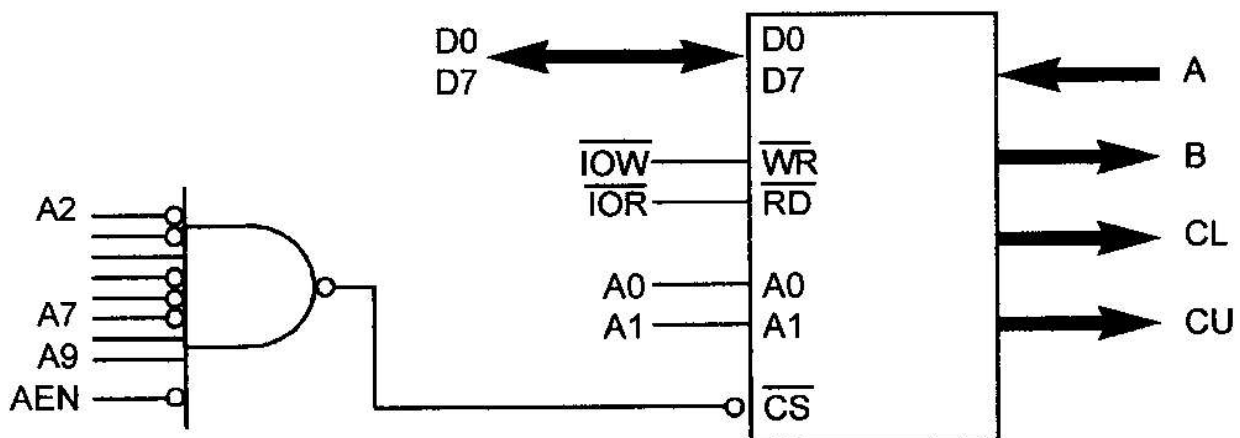
```

Using the EQU directive one can rewrite the above program as follows:

```

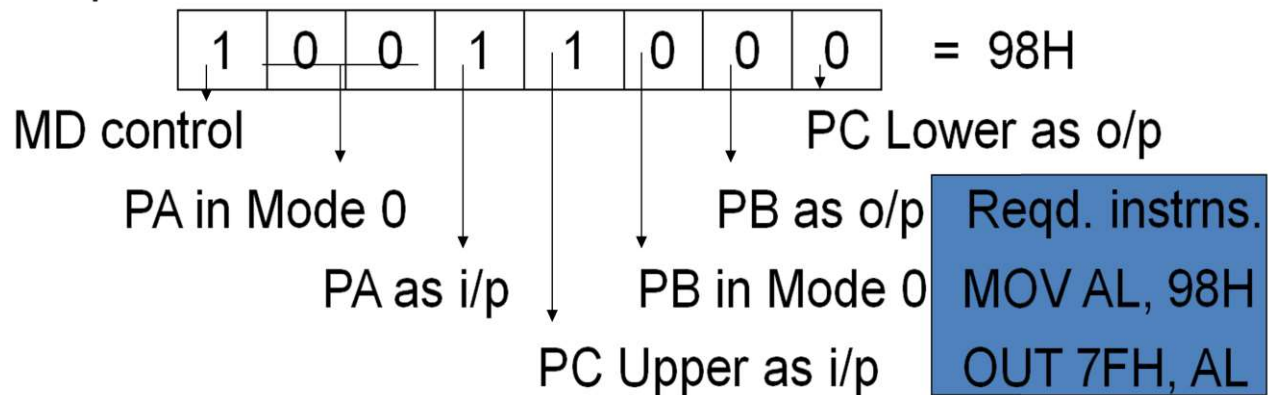
CNTLBYTE EQU 90H    ;PA=in, PB=out, PC=out
PORTA EQU 310H
PORTB EQU 311H
PORTC EQU 312H
CNTLREG EQU 313H
...
...
MOV AL,CNTLBYTE
MOV DX,CNTLREG
OUT DX,AL
MOV DX,PORTA
IN AL,DX
;and so on.

```



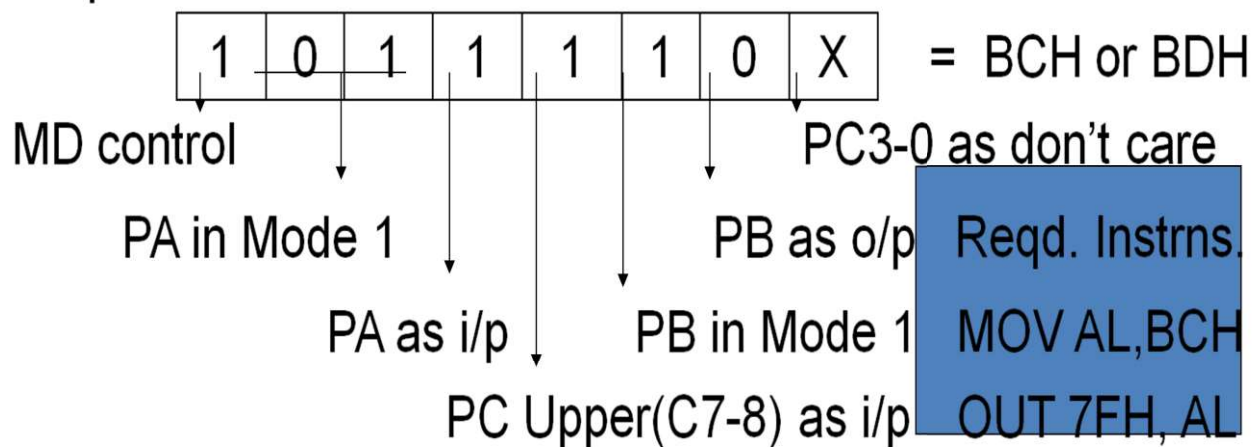
Ex. 1: Configure Port A as i/p in Mode 0, Port B as o/p in mode 0, Port C (Lower) as o/p and Port C (Upper) as i/p ports.

Required MD control word:



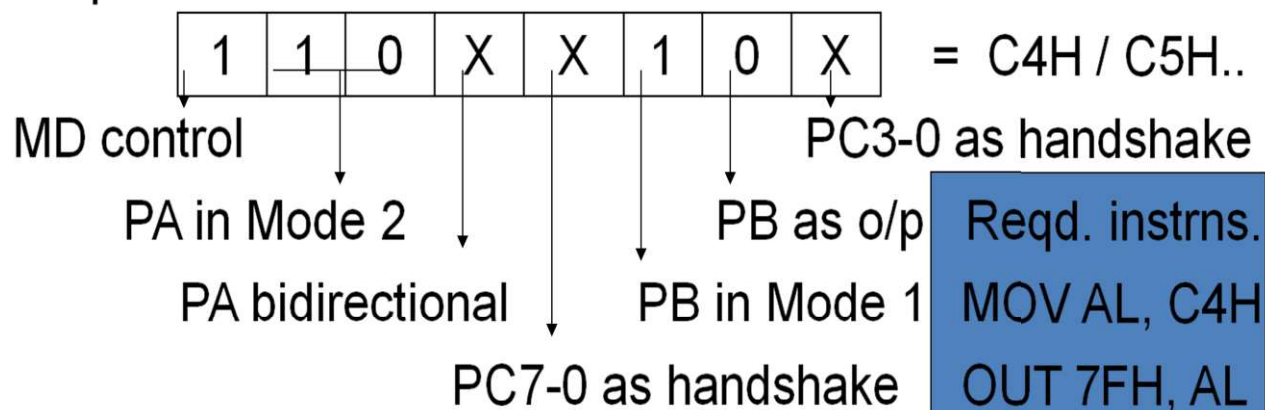
Ex. 2: Configure Port A as i/p in Mode 1, Port B as o/p in mode 1, Port C7-8 as i/p ports. (PC5-0 are handshake lines, some i/p lines and others o/p. So they are shown as X)

Required MD control word:

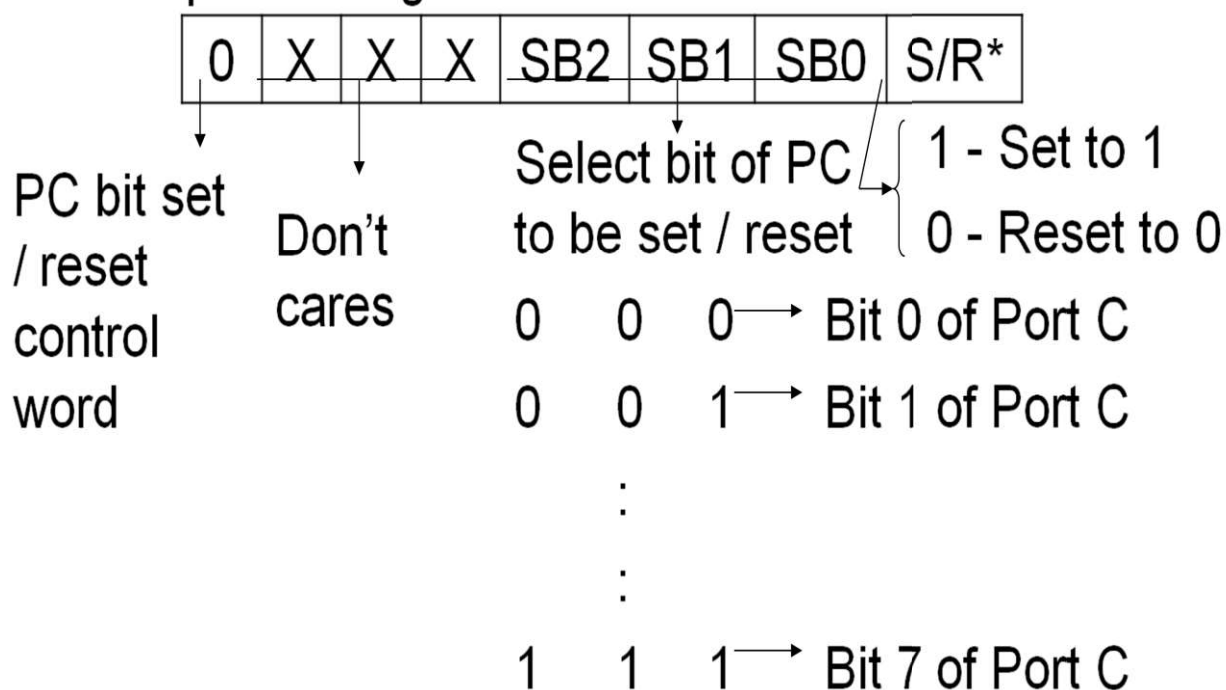


Ex. 3: Configure Port A in Mode 2, Port B as o/p in mode 1.
(PC5-0 are handshake lines for Port A and PC2-0 are handshake signals for port B)

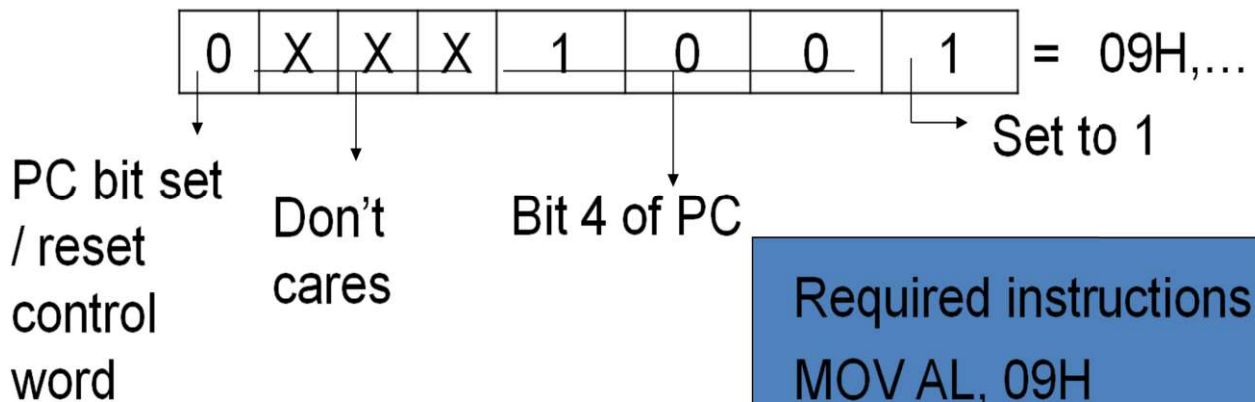
Required MD control word:



Control port having Port C Bit Set / Reset control word

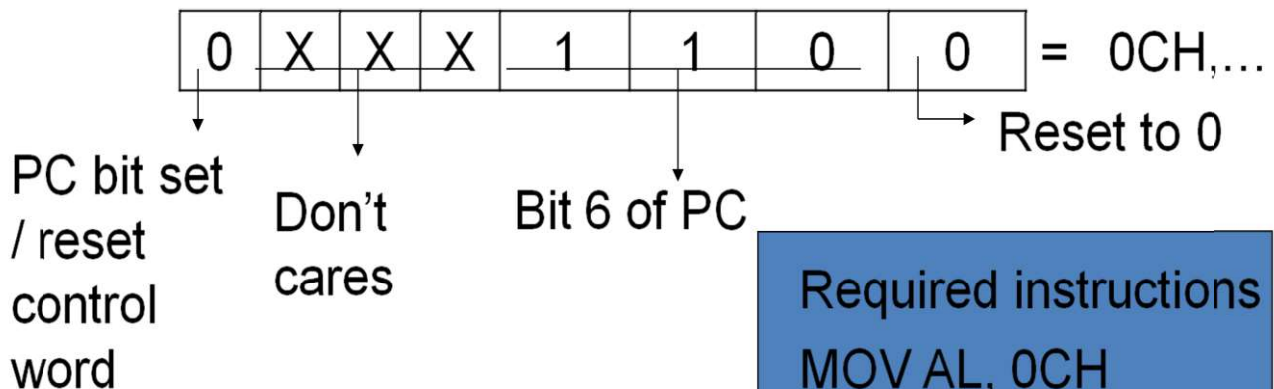


Ex. 1: Set to 1 bit 4 of Port C



Required instructions
MOV AL, 09H
OUT 7FH, AL

Ex. 2: Reset to 0 bit 6 of Port C

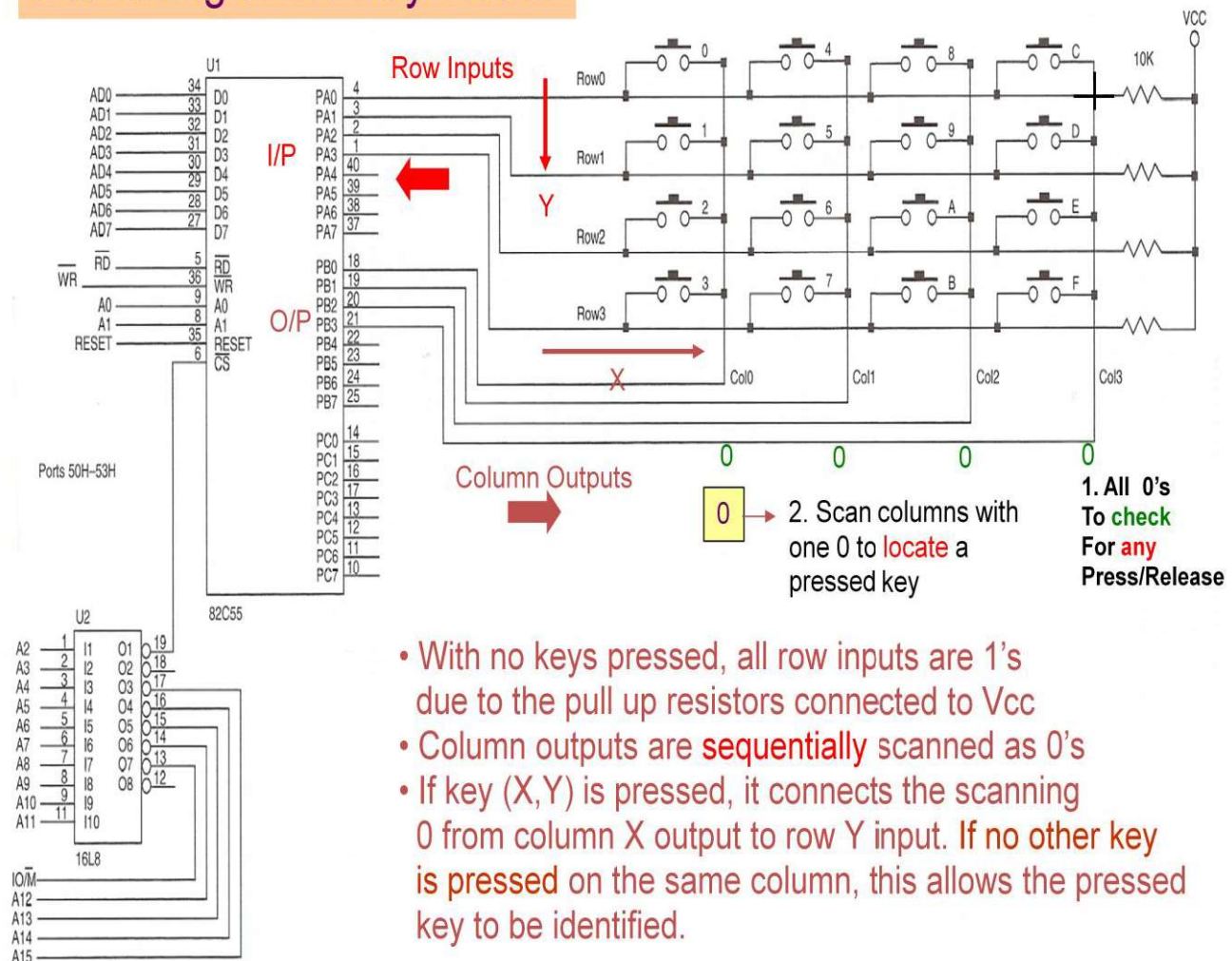


Required instructions
MOV AL, 0CH
OUT 7FH, AL

82C55: Mode 0, Scan Display

- Mode 0 operation causes the 82C55 to function as a buffered input device or as a latched output device.
- In previous example, both ports A and B are programmed as (mode 0) simple latched output ports.
- Port A provides the segment data inputs to display and port B provides a means of selecting one display position at a time.
- Different values are displayed in each digit via fast time multiplexing.

Interfacing a 4x4 Key Matrix



PROGRAM

DATA SEGMENT

PORTA EQU 0FFF8H

PORTB EQU 0FFF9F

CTRLPORT EQU 0FFFBH

DELAY EQU 012CH

DIGITS DB 1,2,3,4,5,6,7,8

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE,DS:DATA

START: MOV AX,DATA

MOV DS,AX

MOV DX,CNTRLPORT

;PORTA ,PORTB O/P PORTS

MOV AL,80H

REPEAT: OUT DX,AL

MOV BH,8

;INITIALIZE DIGIT COUNT

LEA SI,DIGITS

;GET ADDRESS OF THE DIGIT TABLE

MOV BL,0FEH

;CODE TO TURN ON Q0

BACK: MOV AL,BL

MOV DX,PORTA

;TURN ON Q0

OUT DX,AL

MOV AL,[SI]

MOV DX,PORTB

;GET DIGIT TO BE DISPLAYED

SELF: OUT DX,AL

;SEND IT TO 7447 FOR DISPLAY

MOV CX,DELAY

;DELAY CONSTATNT FOR 1MS

LOOP SELF

INC SI

ROL BL,1

;CODE TO TURN ON NEXT TRANSISTOR

DEC BH

;DECREMENT DIGIT COUNT

JNZ BACK

JMP REPEAT

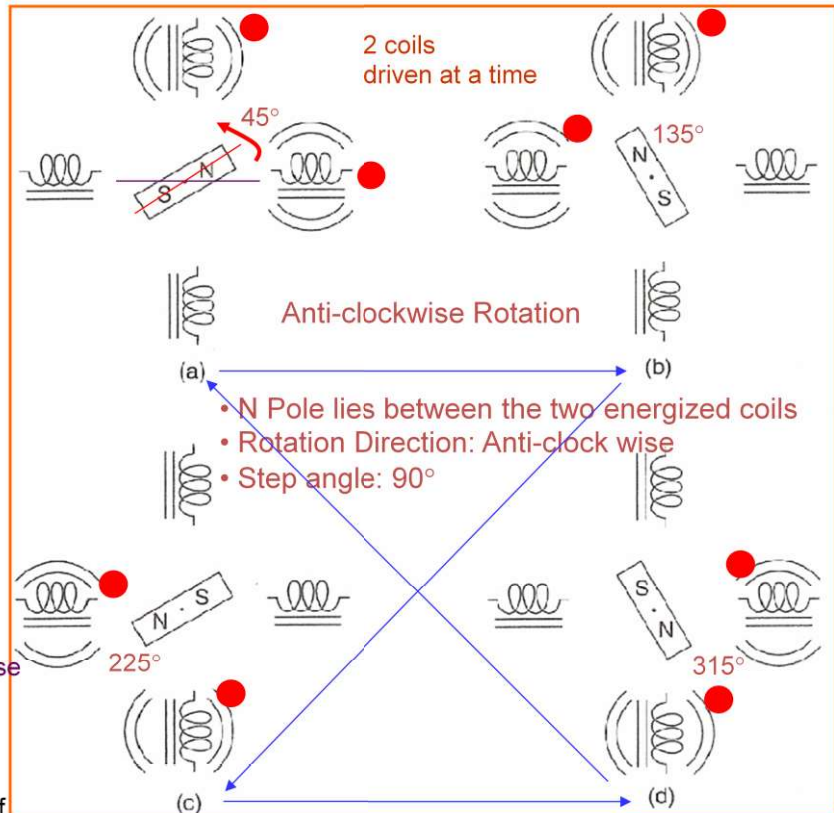
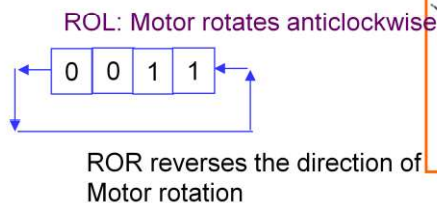
CODE ENDS

END START

Stepper Motor Interface

Wk 10

- Stepper motor is **digital** in nature
- It rotates in a sequence of discrete steps controlled by sequentially energizing a set of coils (windings)
- Step angles vary from 1° to 15° depending on precision required (and cost)

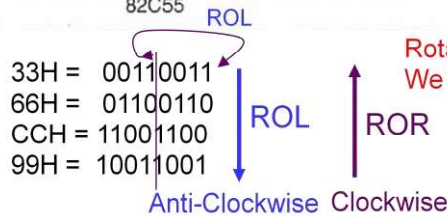


ROTATION PER SEQUENCE = $360/NT$

NT= NUM.OF TURNS

FOUR PATTERN SWITCHING SEQUENCE

Clockwise	Step #	Winding A	Winding B	Winding C	Winding D	Counter-clockwise
↓	1	1	0	0	1	↑
	2	1	1	0	0	
	3	0	1	1	0	
	4	0	0	1	1	



POS	33H or 66H or CCH or 99H
-----	--------------------------

Current angular
Position stored at location
POS (in memory)

NT = NO.OF TEETH ON ROTOR = 200
SPEED OF MOTOR =12 ROTATIONS/MINUTE
CPU FREQUENCY = 10MHZ

ALGORITHM

THE DELAY BETWEEN EACH PATTERN IS CALCULATED AS FOLLOWS

SPEED = 12 ROTATIONS/MINUTE

TO COMPLETE ONE ROTATION 5 SEC REQUIRED

200 TEETH ROTATION = 5 SEC

1 TOOTH ROTATION = $5/200 = 1/40$ SEC = 25MILLI.SEC

DELAY BETWEEN EACH PATTERN = 25msec

CPU FREQ = 10MHZ

1 CLOCK CYCLE = 100nsec

LOOP INSTRUCTION TAKES 17CLOCK CYCLES

TIME TAKEN FOR 1 ITERATION $17 \times 100\text{ns} = 1.7\text{micro sec}$

No.of iteration(count) requires for 25m.sec delay = $25 \times 1000 / 1.7 = 14705$

SEND THE FIRST VALUE AS 33H. ROTATE IT BY ONE POSITION TO GET NEXT PATTERN.

33H IS CHOSEN IN PLACE OF 03H SO THAT ROTATION OF 8-BIT DATA GIVES CORRECT VALUE

SEND ALL PATTERNS AND CONTINUE THE SET OF PATTERN INDEFINITELY

PROGRAM

DATA SEGMENT

PORTC EQU 8004H

CNTRLPR1 EQU 8006H

DELAY EQU 14705

DATA ENDS

CODE SEGMENT

ASSUME CS: CODE, DS: DATA

MOV AX, DATA

MOV DS, AX

MOV AL, 80H

; ALL PORTS AS O/P PORTS

MOV DX, CNTRLPR1

OUT DX, AL

MOV AL, 33H

; SELECT THE FIRST SWITCH PATTERN

MOV DX, PORTC

OUT DX, AL

ROR AL, 1

; NEXT SWITCH PATTERN FOR CLOCK WISE ROTATION

MOV CX, DELAY

LOOP SELF

JMP BACK

CODE ENDS

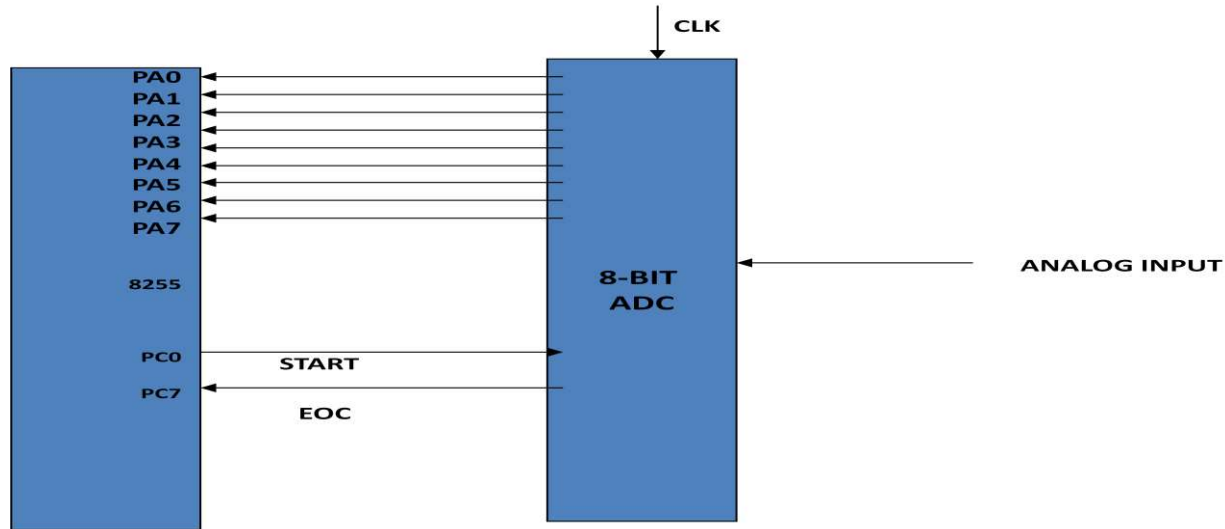
END START

PROGRAM TO ROTATE STEPPER MOTOR IN ANTI CLOCKWISE ROTATION
FOR 180 FOR THE ABOVE SPECIFICATION

EACH STEP = $360/NT = 360/200 = 1.8\text{DEG}$

THEREFORE $N = 180/1.8 = 100$

Analog to Digital Converter



WRITE A PROGRAM FOR 8-BIT ADC TO SAMPLE
ANALOG INPUT AND STORE THE DIGITAL VALUE IN MEMORY

ALGORITHM

1. SEND THE START PULSE TO ADC
2. WAIT FOR EOC TO BECOME ACTIVE
3. READ THE DATA FROM ADC AND STORE IT IN MEMORY

MD=98H
PCBSR = 00 (RESET)/ 01(SET)

DATA SEGMENT

```
PORTA EQU 0FFE0H
PORTC EQU 0FFE4H
CNTPRT EQU 0FFE6H
MEM DW 2000H
```

DATA ENDS

CODE SEGMENT

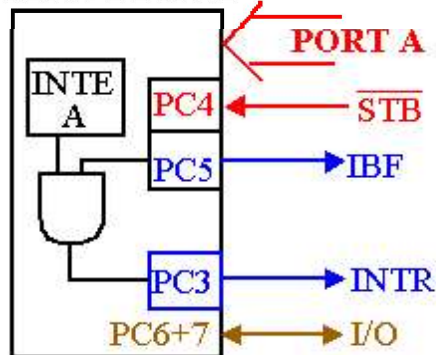
```
ASSUME CS:CODE,DS:DATA
START: MOV AX,DATA
MOV DS,AX
MOV DX,CNTPRT
MOV AL,98H
OUT DX,AL
MOV AL,01H
OUT DX,AL
MOV AL,00
```

```
OUT DX,AL
MOV DX,PORTC
CHK: IN AL,DX
AND AL,80H
JZ CHK
MOV DX,PORTA
IN AL,DX
MOV MEM,AL
INT 3H
CODE ENDS
END START
```

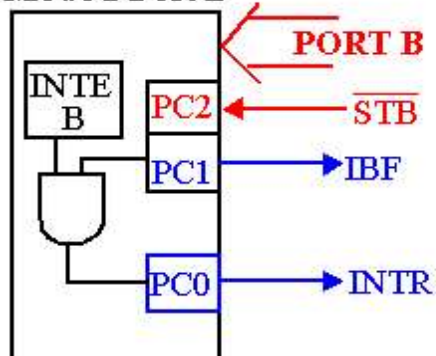
82C55: Mode 1 Strobed Input

- **$\overline{\text{STB}}$** : The strobe input loads data into the port latch on a 0-to-1 transition.
- **IBF** : Input buffer full is an output indicating that the input latch contain information.
- **INTR** : Interrupt request is an output that requests an interrupts.
- **INTE** : The interrupt enable signal is neither an input nor an output; it is an internal bit programmed via the PC4 (port A) or PC2 (port B) bits.
- **PC7,PC6** : The port C pins 7 and 6 are general purpose I/O pins that are available for any purpose.

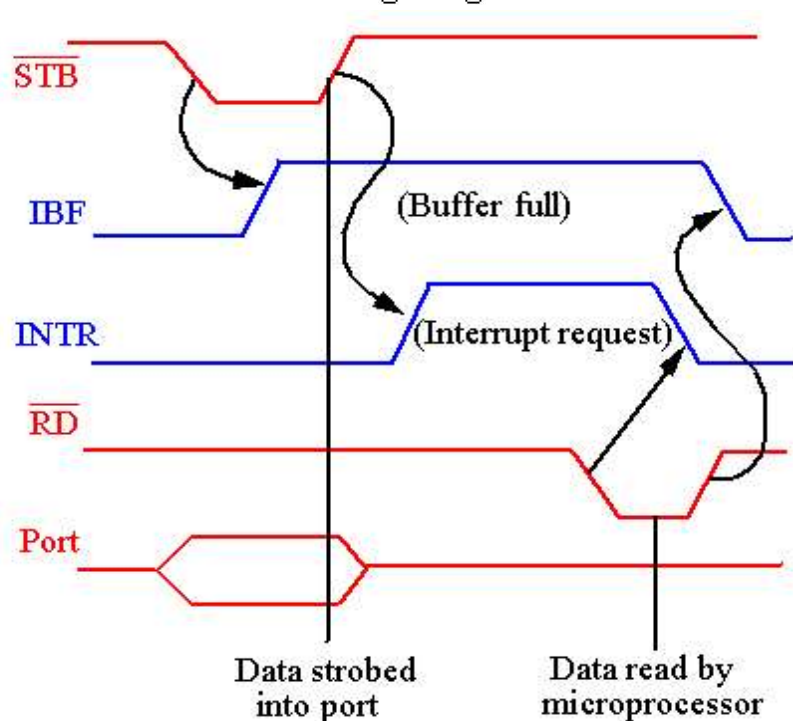
Mode 1 Port A



Mode 1 Port B



Timing Diagram



Interfacing a Keyboard to μP using Port A in Mode 1 (Strobed Input)

Handshake (Polling) Method

;A procedure that reads the keyboard and
;returns the ASCII key code in AL

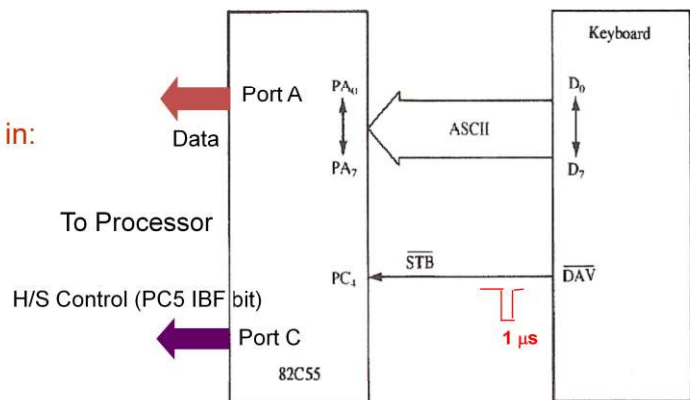
```

BIT5 EQU 20H ;00100000 Mask defining PC5 (IBF for Port A)
PORTC EQU 22H
PORTA EQU 20H
READ PROC NEAR
    .REPEAT ;poll IBF bit
        IN AL,PORTC
        TEST AL,BIT5
    .UNTIL !ZERO? ; Quit polling when bit 5 read is not ZERO (IBF=1)
    IN AL, PORTA ; get ASCII value of key pressed from keyboard
    RET
READ ENDP

```

8255 should be programmed for operation in:

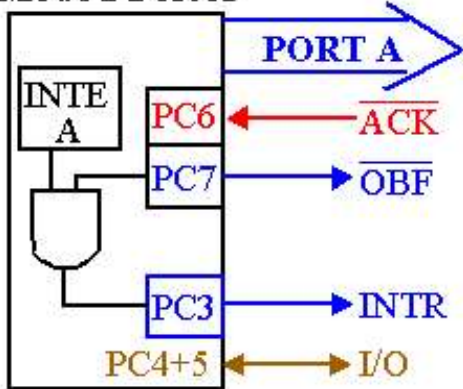
- Group A in Mode 1
- Port A is input



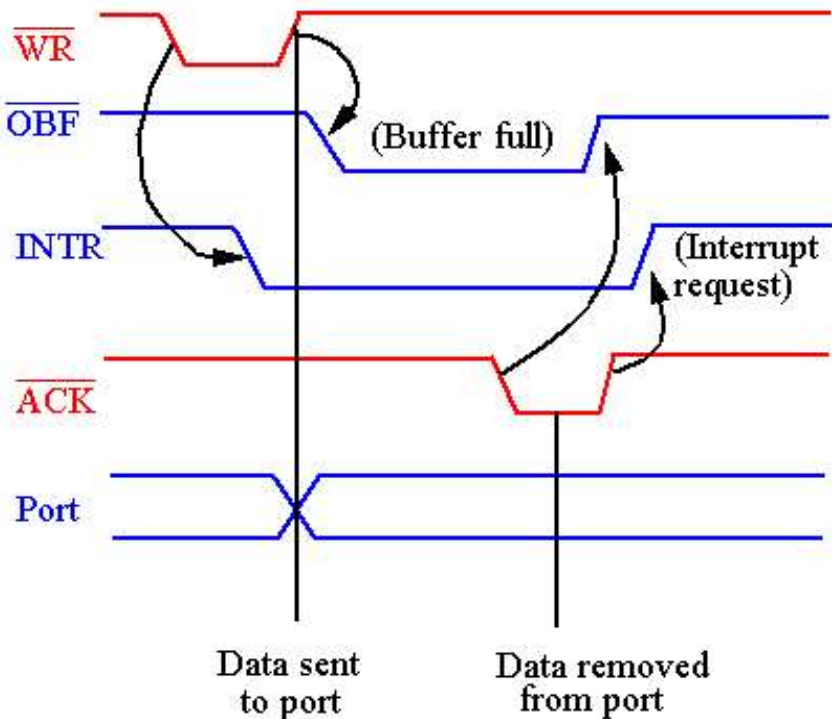
82C55 : Mode 1 Output Exam.

- **~OBF** : Output buffer full is an output that goes low when data is latched in either port A or port B. Goes low on ~ACK.
- **~ACK** : The acknowledge signal causes the ~OBF pin return to 0. This is a response from an external device.
- **INTR** : Interrupt request is an output that requests an interrupt.
- **INTE** : The interrupt enable signal is neither an input nor an output; it is an internal bit programmed via the PC6(Port A) or PC2(port B) bits.
- **PC5,PC4** : The port C pins 5 and 4 are general-purpose I/O pins that are available for any purpose.

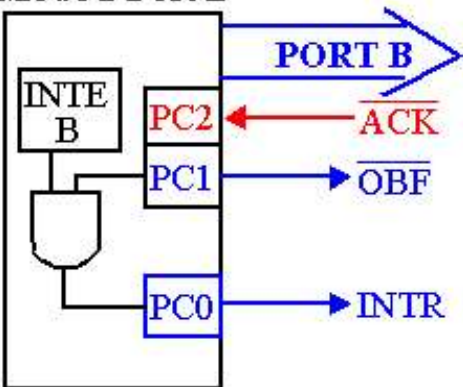
Mode 1 Port A



Timing Diagram



Mode 1 Port B

Interfacing a Printer to μP using Port B in Mode 1 (Strobed Output)

;A procedure that transfers an ASCII character from AH to the printer

;connected to port B

BIT1 EQU 2 ; Bit PC1 = #OBF for port B

PORTC EQU 62H

PORTB EQU 61H

CMD EQU 63H; The 8255 command byte address

PRINT PROC NEAR

.REPEAT ;Wait for printer ready to receive a new char- Poll #OBF till high

IN AL,PORTC

TEST AL,BIT1

μP Polling for #OBF = 1

.UNTIL !ZERO? ; #OBF = 1 No data in output buffer, so can write into it!

MOV AL,AH ; Write ASCII char data into port latch

OUT PORTB,AL

MOV AL,8 ;Generate data strobe pulse
; on PC4

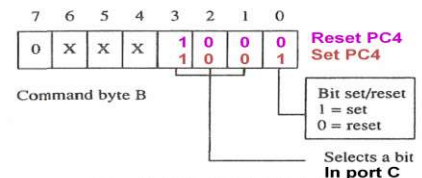
OUT CMD,AL

MOV AL,9

OUT CMD,AL

RET

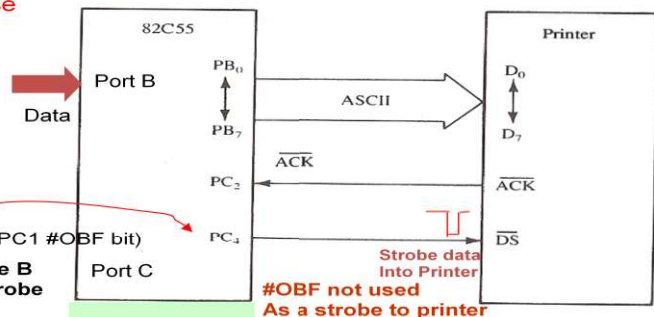
PRINT ENDP



Data from μP to
port latch

Data from port latch
to Printer

Ordinary C I/O bit
Set and reset with command byte B
To generate the #DS pulse (Strobe
output) signal to Ex Device



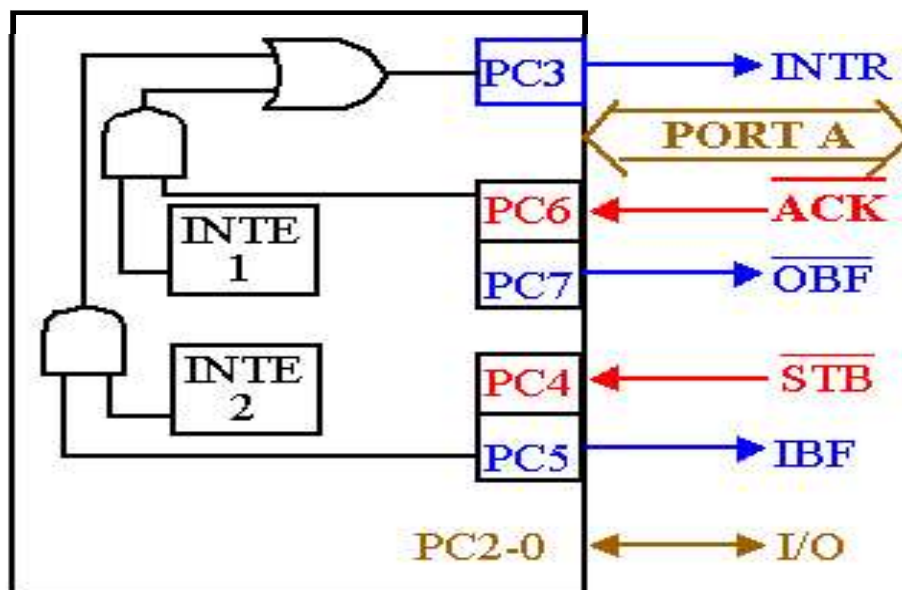
82C55: Mode 2 Bi-directional Operation

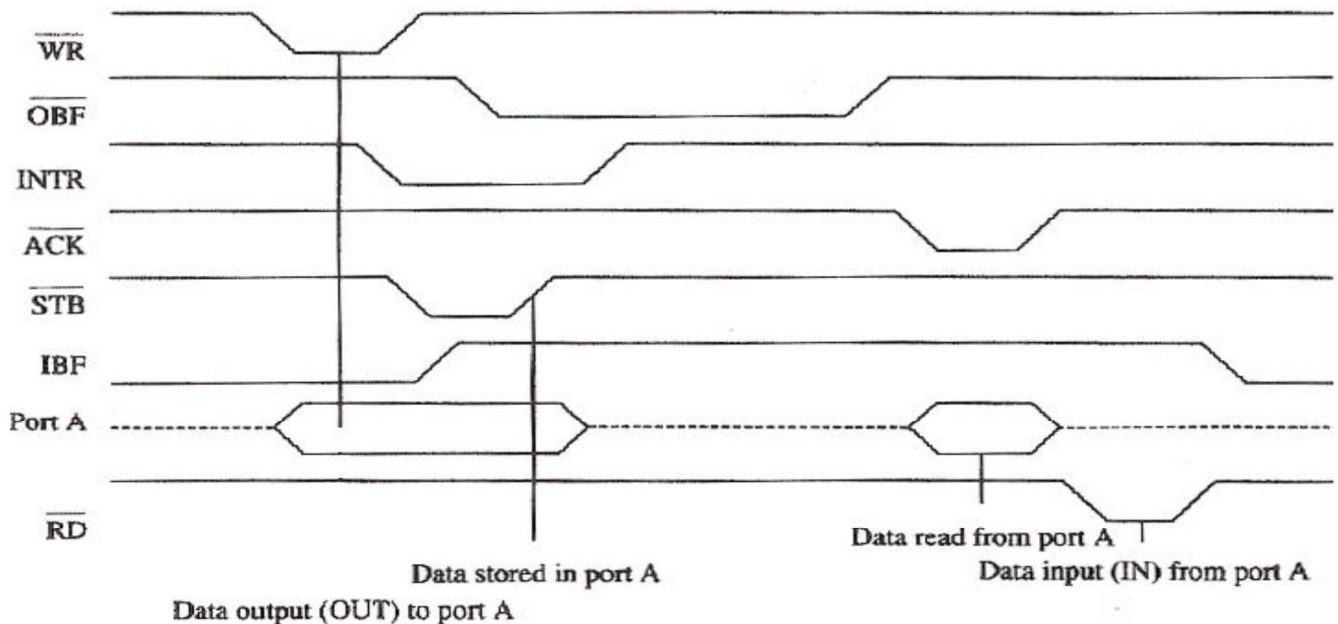
- This functional configuration provides a means for communicating with a peripheral device or structure on a single 8-bit bus for both transmitting and receiving data (bidirectional bus I/O).
- “Handshaking” signals are provided to maintain proper bus flow discipline in a similar manner to MODE 1.
- 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).

- **INTR** : Interrupt request is an output that requests an interrupt.
- **~OBF** : Output Buffer Full is an output indicating that that output buffer contains data for the bi-directional bus.
- **~ACK** : Acknowledge is an input that enables tri-state buffers which are otherwise in their high-impedance state.
- **~STB** : The strobe input loads data into the port A latch.
- **IBF** : Input buffer full is an output indicating that the input latch contains information for the external bi-directional bus.
- **INTE** : Interrupt enable are internal bits that enable the INTR pin. BIT PC6(INTE1) and PC4(INTE2).
- **PC2,PC1,PC0** : These port C pins are general-purpose I/O pins that are available for any purpose.



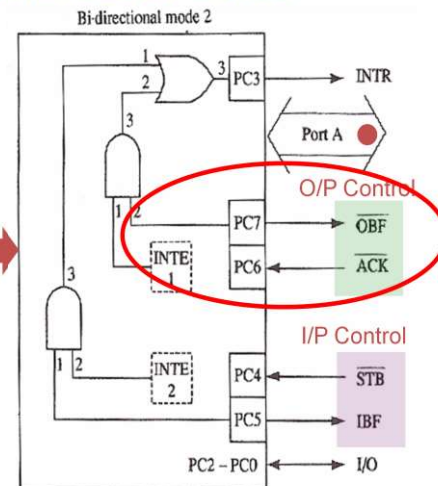


Mode 2 Example: Processor Sends Data to External Device on the Bidirectional Bus

To send Data from processor to external device:

1. Processor checks if #OBF = 1 (No data pending in port)
2. Processor OUTs data to port (Writes it into Port A latch- not on external bus yet)
3. Port automatically lowers #OBF O/P to alert device to take data
4. Device detects 3 and lowers #ACK input to port
5. This raises #OBF high
6. #ACK enables Port external bus to carry latch data so it can be taken by device
7. After device takes data it raises #ACK high

From μP



A procedure transmits AH through the bidirectional bus

```
BIT7 EQU 80H
PORTC EQU 62H
PORTA EQU 60H
```

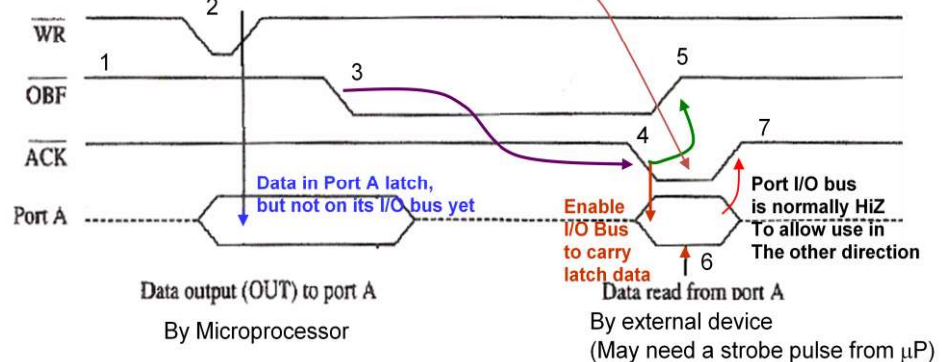
```
TRANSPROC NEAR
```

```
.REPEAT
    IN AL,PORTC
    TEST AL,BIT7
    .UNTIL IZERO?
    MOV AL,AH
    OUT PORTA,AL
    RET
```

```
TRANS    ENDP
```

Wait for #OBF = 1
Result = 1 (Not zero)

I understand you have data for me in your latch. Please put it on the bus so I can take it!

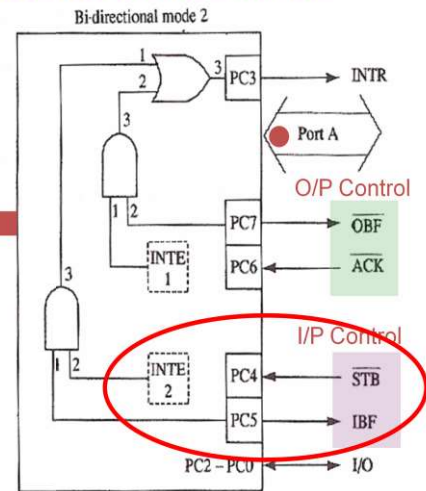


Mode 2 Example: Processor Receives Data from External Device on the Bidirectional Bus

To Receive Data:

1. External device sending data checks if #IBF = 0 (No pending data in port latch not read by processor) (Hardware Polling)
2. Then it puts its data on external bus and strobes it into port latch using #STB
3. IBF automatically goes high until data is read by processor
4. Processor polls IBF for IBF = 1 to make sure data is in port latch (software polling)
5. Processor reads data from port
6. This automatically lowers IBF to enable further writes

To μP



;A procedure that reads data from the bidirectional bus into AL

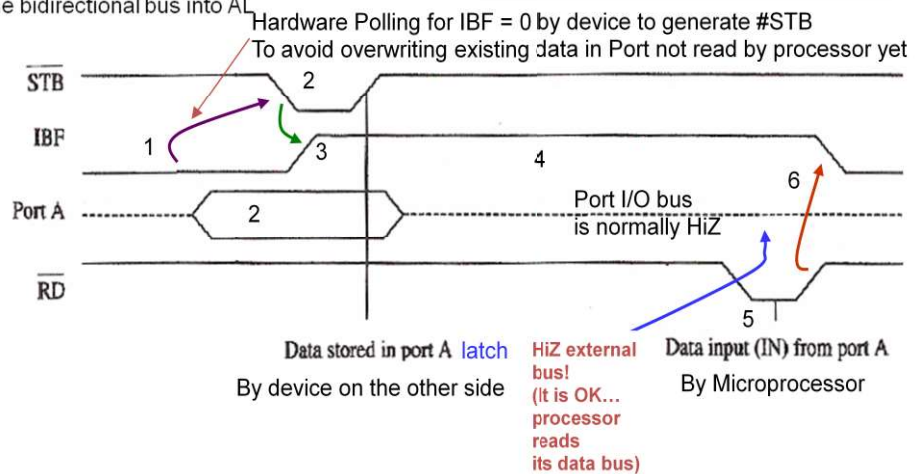
```
BIT5 EQU 20H
PORTC EQU 62H
PORTA EQU 60H
```

```
READ PROC NEAR
```

```
    .REPEAT ;Wait for IBF = 1
    IN AL,PORTC
    TEST AL,BIT5
    .UNTIL IZERO?
    IN AL,PORTA
    RET
```

Step 5

```
READ PROC NEAR
```



Summary of port functions in various modes

	Mode 0	Mode 1	Mode 2
Port A	IN	IN	I/O
Port B	IN	IN	Not used
0		INTR _B	I/O
1		IBF _B	I/O
2		STB _B	I/O
Port C	IN	INTR _A	INTR
3		STB _A	STB
4		IBF _A	IBF
5		I/O	ACK
6		I/O	ACK
7		I/O	OBF

EX: Design an 8086 microprocessor interfacing circuit by using 8255 PPI to operate 8bits binary counter and display on LEDs where port A address=39H, Control=F9H?

MOV AL, 80H

OUT F9H, AL

START:

MOV AL, 00H

**** : OUT 39H, AL**

CALL DELAY

INC AL

JMP **

INT 3

DELAY

MOV CX, FFFFH

***** MOV BL, 03H**

NOP

**** DEC BL**

JNZ **

LOOP ***

RET

