8255 PPI

- PPI

  Programmable Peripheral Interface

Intel 8255 PPI

PPI – Programmable Peripheral Interface

It is an I/O port chip used for interfacing I/O devices with microprocessor

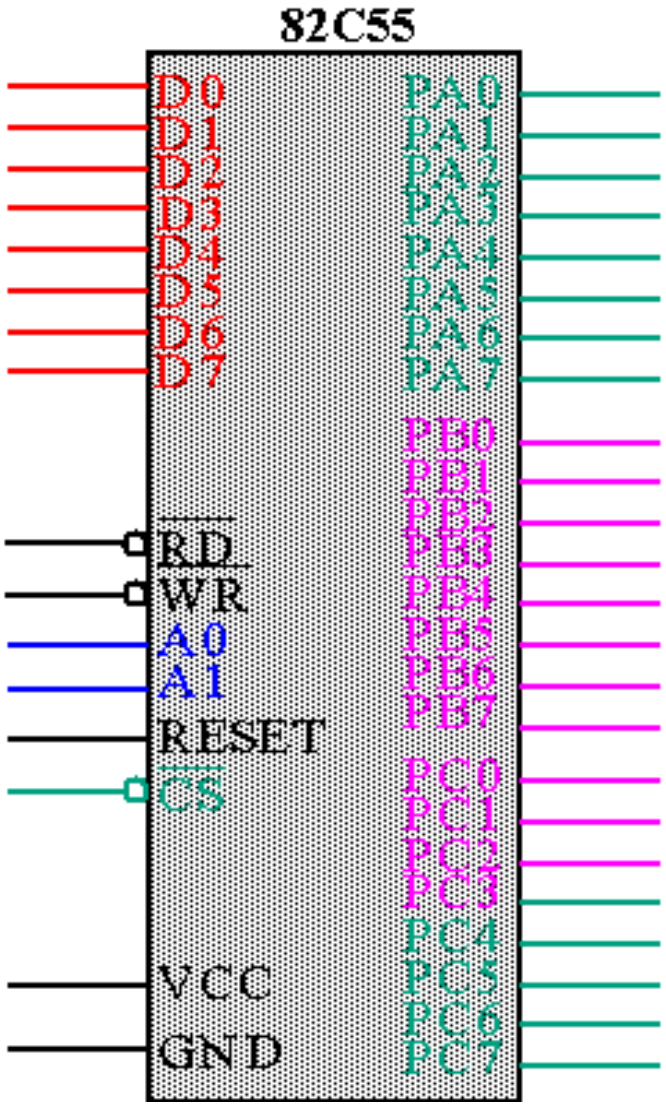
Very commonly used peripheral chip

Knowledge of 8255 essential for students in the Microprocessors lab for Interfacing experiments

About 82C55

- The 82C55 is a popular interfacing component, that can interface any TTL-compatible I/O device to a microprocessor.
- It is used to interface to the keyboard and a parallel printer port in PCs (usually as part of an integrated chipset).
- Requires insertion of wait states if used with a microprocessor using higher that an 8 MHz clock.
- PPI has 24 pins for I/O that are programmable in groups of 12 pins and has three distinct modes of operation.

82C55 : Pin Layout



Group A

Port A (PA7-PA0) and upper half of port C (PC7 - PC4)

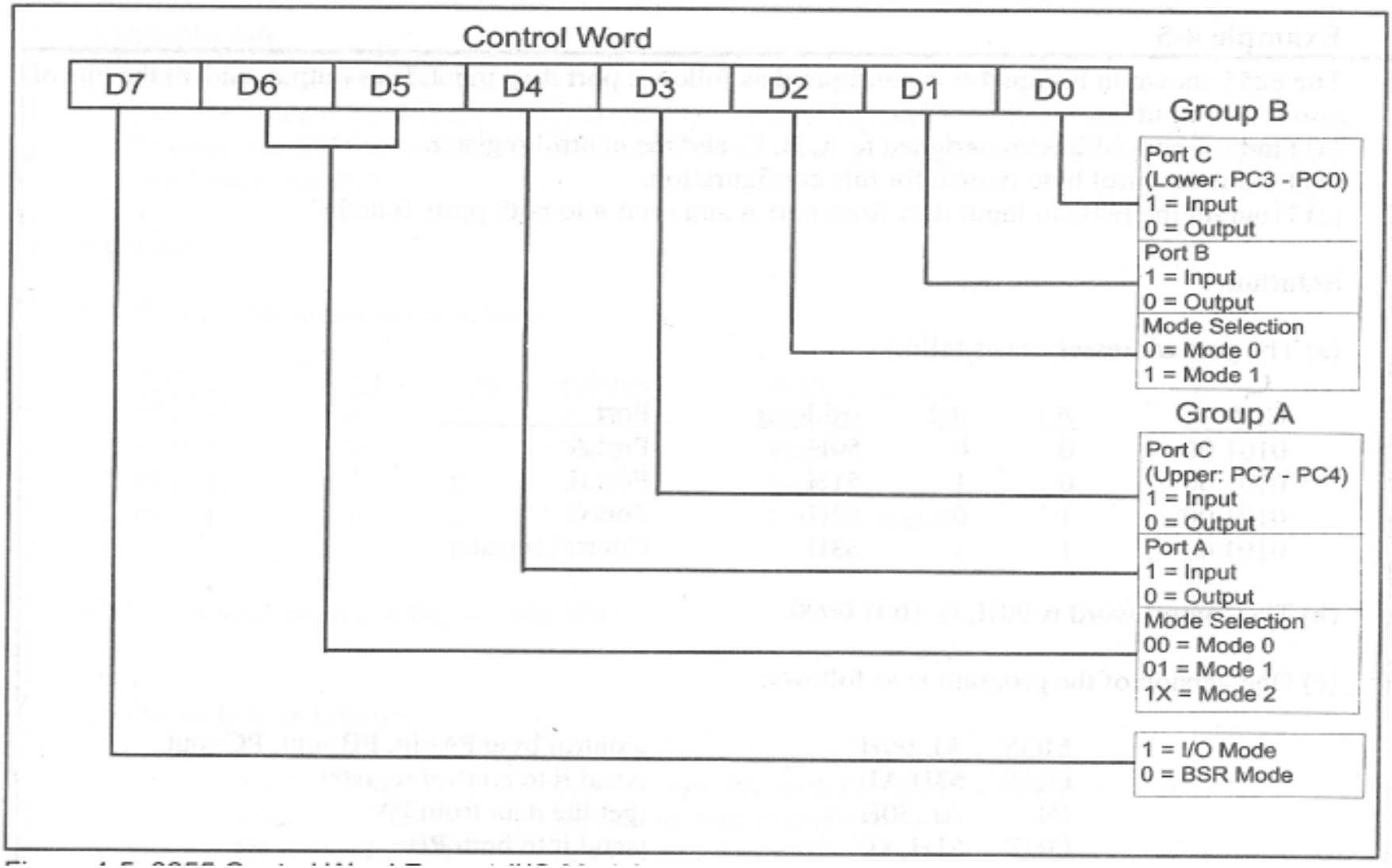
Group B

Port B (PB7-PB0) and lower half of port C (PC3 - PC0)

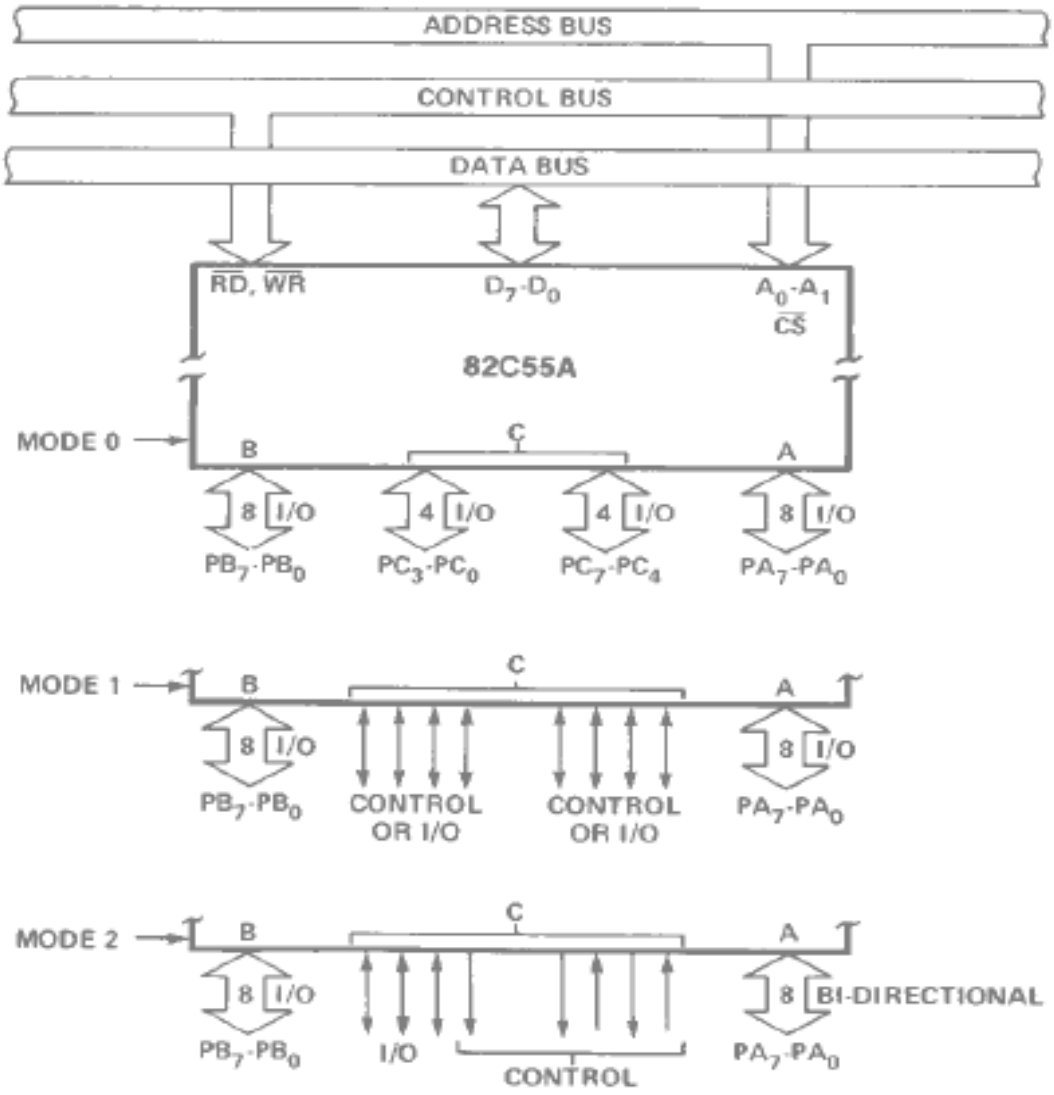
I/O Port Assignments

A_1	A_0	Function
0	0	Port A
0	1	Port B
1	0	Port C
1	1	Command Register

8255 Control Word



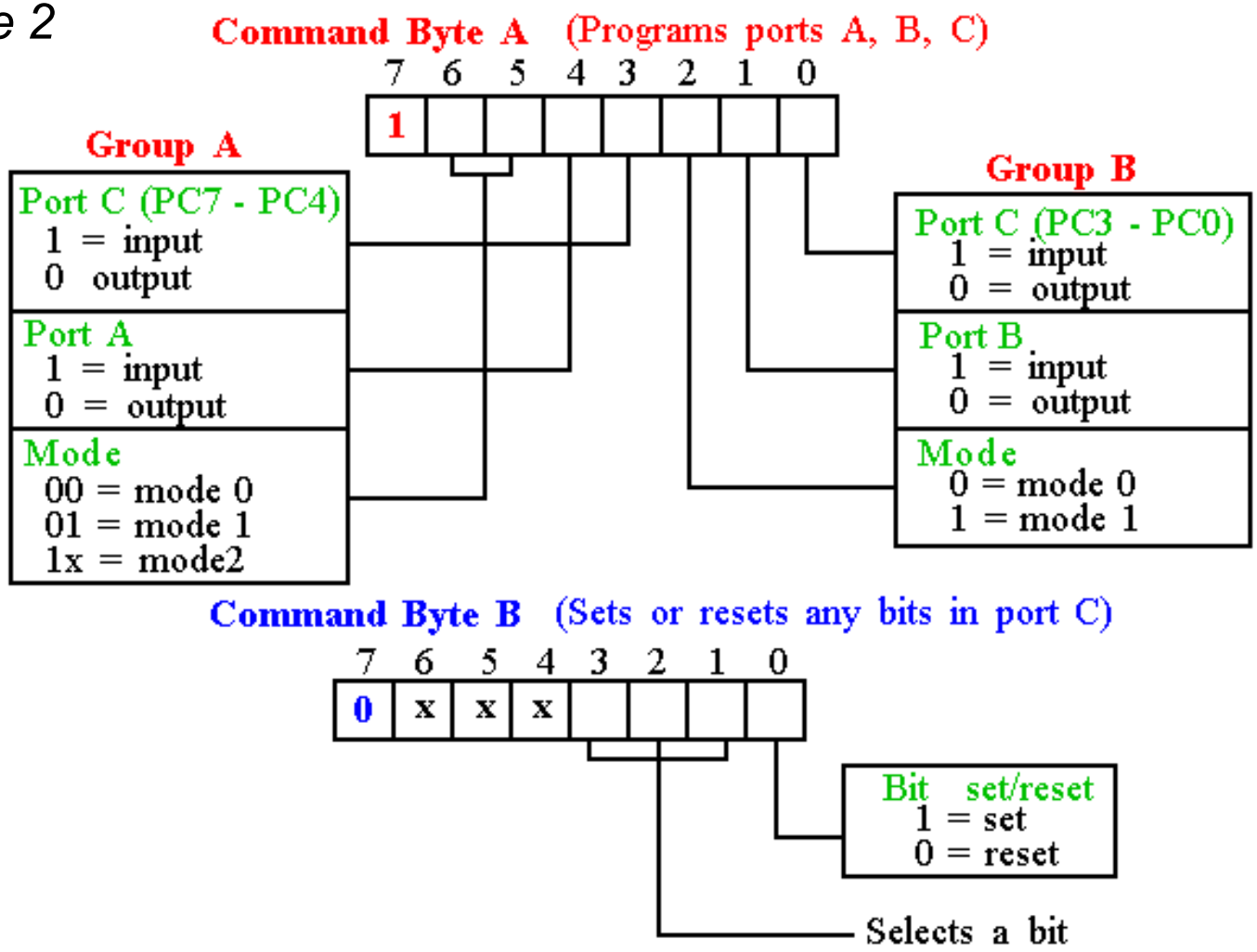
Basic Mode Definitions and Bus Int



- Mode 0
  - Basic I/O
- Mode 1
  - Strobe I/O
- Mode 2
  - Bi-Dir Bus

Programming 8255

- 8255 has three operation modes: *mode 0, mode 1, and mode 2*



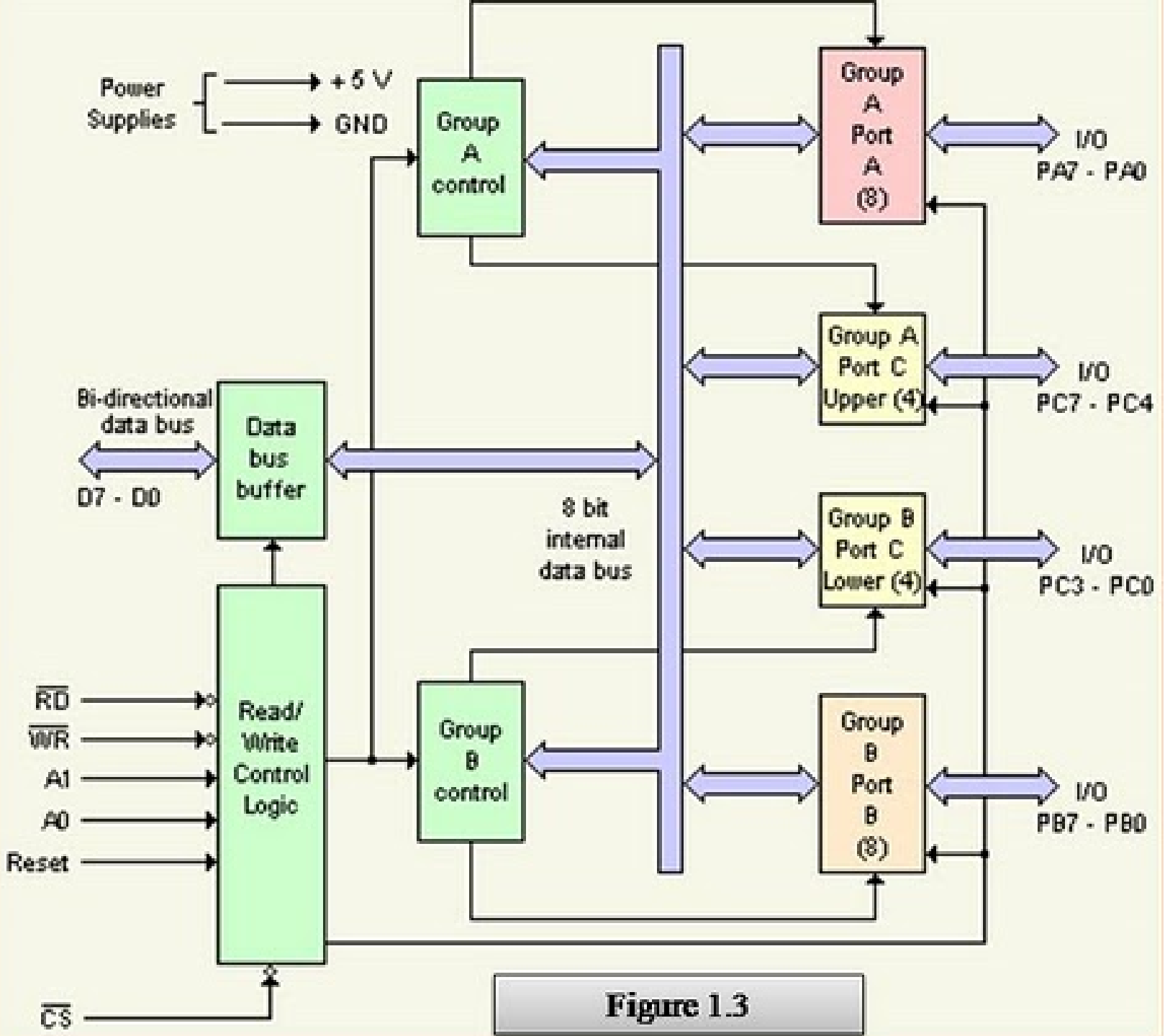

Power Supplies
+5 V
GND
Group A control
Group A Port A (8)
I/O PA7 - PA0
Group A Port C Upper (4)
I/O PC7 - PC4
Bi-directional data bus
D7 - D0
Data bus buffer
8 bit internal data bus
Group B Port C Lower (4)
I/O PC3 - PC0
RD
WR
A1
A0
Reset
Read/ Write Control Logic
Group B control
Group B Port B (8)
I/O PB7 - PB0
CS

Figure 1.3

8255 PPI contd.

3 ports in 8255 from user's point of view

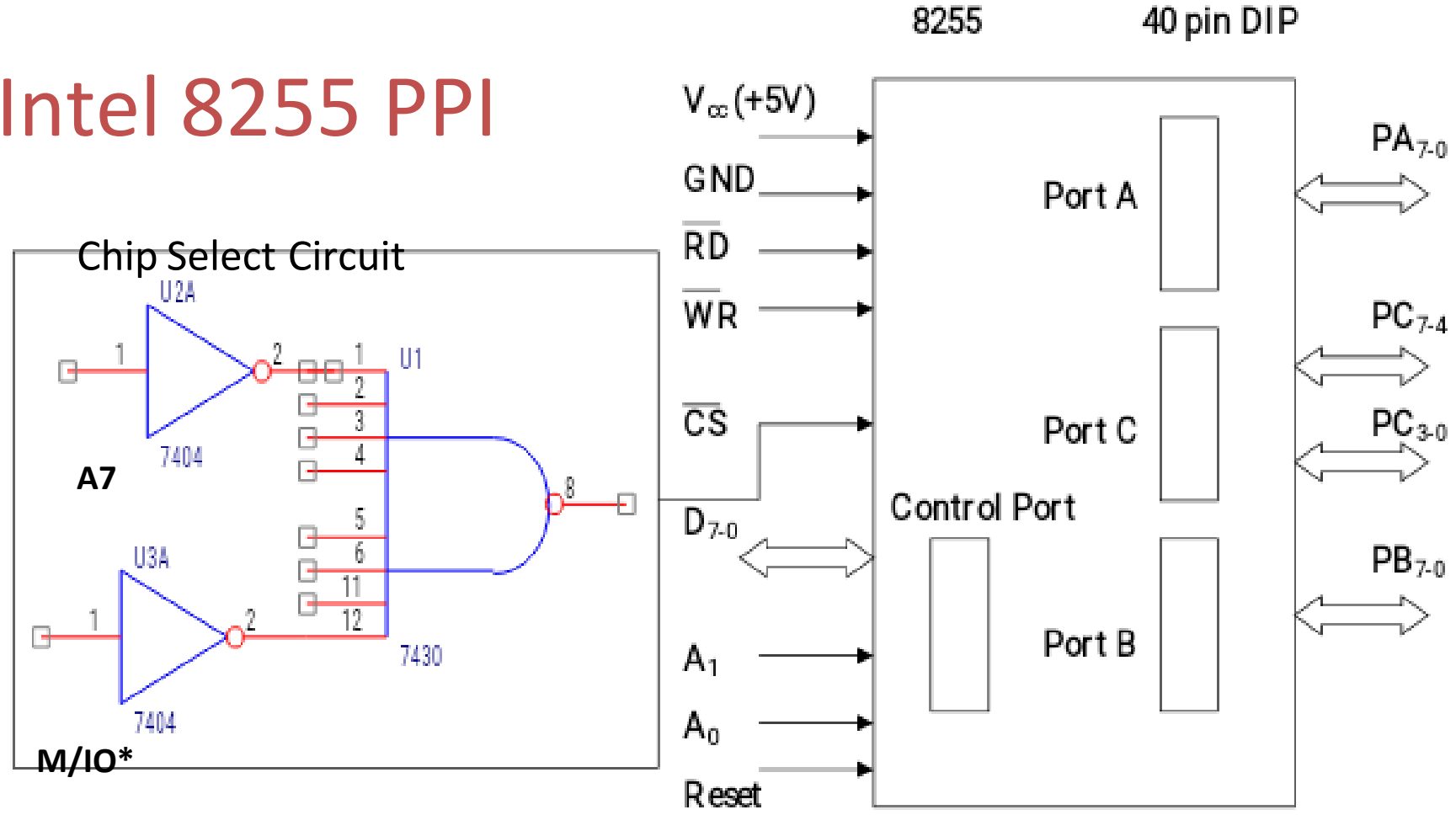
- Port A, Port B and Port C.

Port C composed of two independent 4-bit ports

- PC7-4 (PC Upper) and PC3-0 (PC Lower)

A1	A0	Selected port
0	0	Port A
0	1	Port B
1	0	Port C
1	1	Control port

Intel 8255 PPI



A7=0, A6=1, A5=1, A4=1, A3=1, A2=1, & M/IO*= 0

8255 PPI Contd.

There is also a Control port from the Processor point of view. Its contents decides the working of 8255.

When CS (Chip select) is 0, 8255 is selected for communication by the processor. The chip select circuit connected to the CS pin assigns addresses to the ports of 8255.

For the chip select circuit shown, the chip is selected when A7=0, A6=1, A5=1, A4=1, A3=1, A2=1, & M/IO*= 0

Port A, Port B, Port C and Control port will have the addresses as 7CH, 7DH, 7EH, and 7FH respectively.

8255 PPI Contd.

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.

8255 PPI Contd.

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 ort 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.

8255 PPI Contd.

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.

8255 Handshake signals

Where are the Handshake signals?

Port C pins act as handshake signals, when Port A and Port B are configured for other than Mode 0.

Port A in Mode 2 and Port B in Mode 1 is possible, as it needs only 5+3 = 8 handshake signals

After Reset of 8255, Port A , Port B , and Port C are configured for Mode 0 operation as input ports.

8255 Handshake signals Contd.

PC2-0 are used as handshake signals by Port B when configured in Mode 1. This is immaterial whether Port B is configured as i/p or o/p port.

PC5-3 are used as handshake signals by Port A when configured as i/p port in Mode 1.

PC7,6,3 are used as handshake signals by Port A when configured as o/p port in Mode 1.

PC7-3 are used as handshake signals by Port A when configured in Mode 2.

8255 PPI Contd.

Port A can work in Mode 0, Mode 1, or Mode 2

Port B can work in Mode 0, or Mode 1

Port C can work in Mode 0 only, if at all

Port A, Port B and Port C can work in Mode 0

Port A and Port B can work in Mode 1

Only Port A can work in Mode 2

8255 MD Control word

Control port having Mode Definition (MD) control word

Means Mode Definition control word

1 - PC_U as input

0 - PC_U as output

1 - PA as input

0 - PA as output

1 -PC_L as input

0 -PC_L as output

1 - PB as input

0 - PB as output

1 – Port B in Mode 1

0 – Port B in Mode 0

M2A M1A

0 0 → Port A in Mode 0

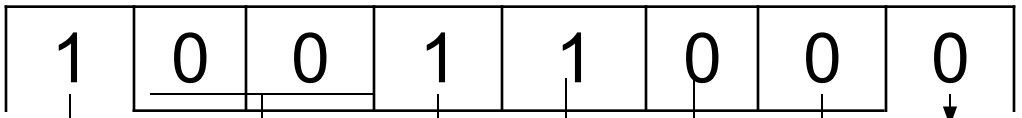
0 1 → Port A in Mode 1

1 0/1 → Port A in Mode 2

8255 MD Control word Contd.

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:



MD control

PA in Mode 0

PA as i/p

PC Upper as i/p

PB in Mode 0

PB as o/p

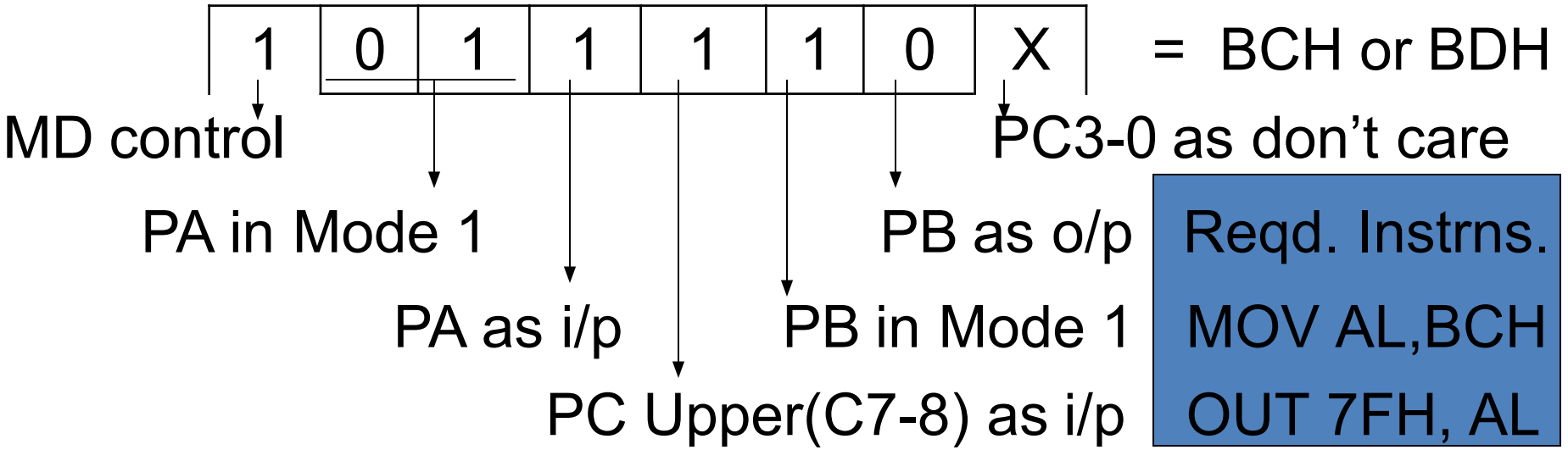
PC Lower as o/p

Reqd. instrns.
MOV AL, 98H
OUT 7FH, AL

8255 MD Control word Contd.

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:



8255 Contd.

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

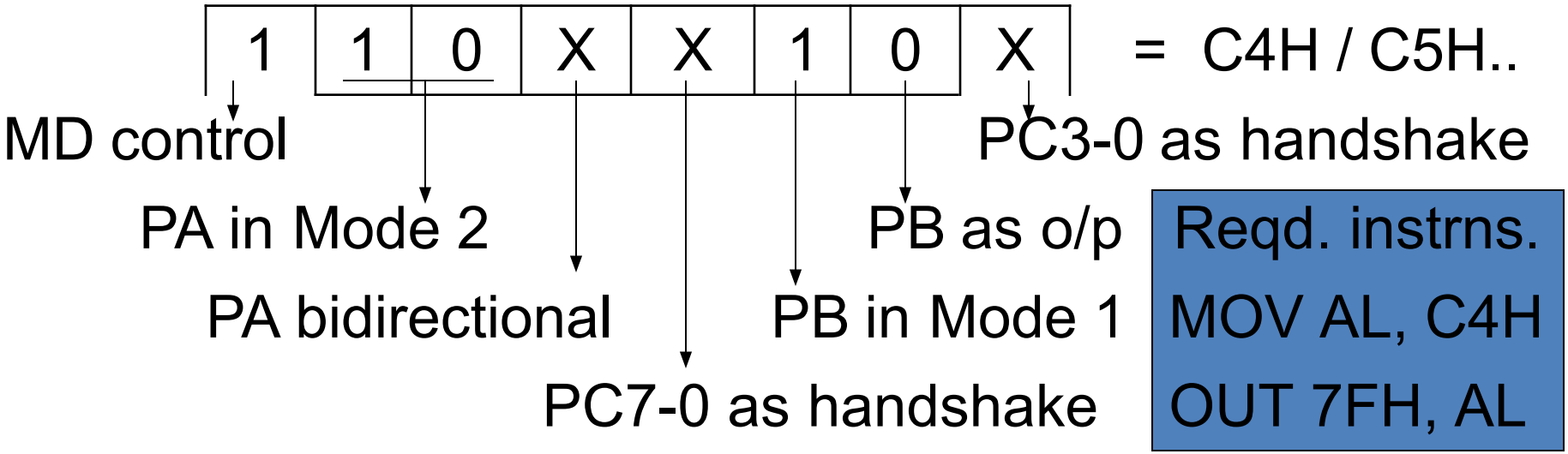
- as i/p or o/p in Mode 0, 1, or 2

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

8255 MD Control word Contd.

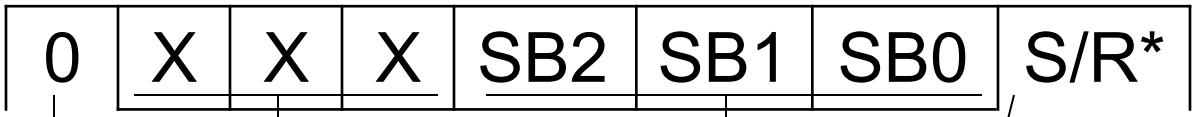
Ex. 3:Configure Port A in Mode 2, Port B as o/p in mode 1. (PC5-0 are handshake lines for Port A and PC2-0 are handshake signals for port B)

Required MD control word:



8255 PCBSR Control word

Control port having Port C Bit Set / Reset control word



PC bit set / reset control word

Don't cares

Select bit of PC to be set / reset

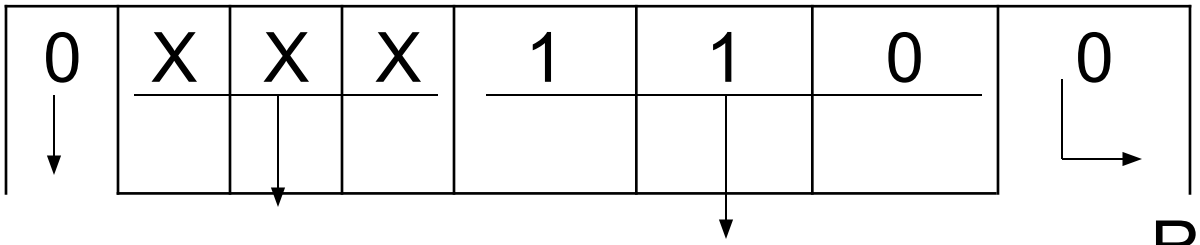
1 - Set to 1

0 - Reset to 0

0	0	0	→ Bit 0 of Port C
0	0	1	→ Bit 1 of Port C
:			
:			
1	1	1	→ Bit 7 of Port C

8255 PCBSR Control word contd.

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



PC bit set / reset control word

Don't cares

Bit 6 of PC

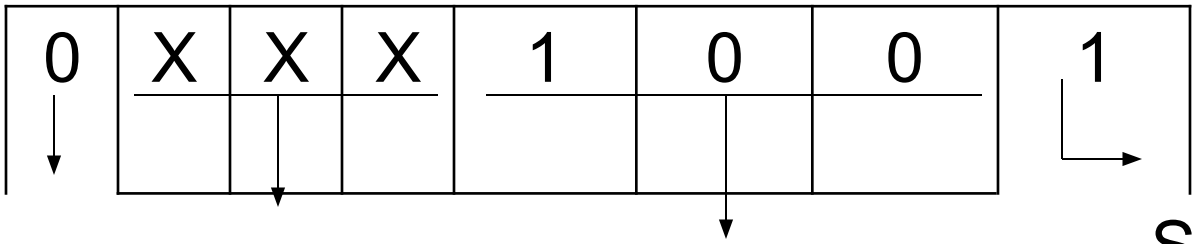
Required instructions

MOV AL, 0CH

OUT 7FH, AL

8255 PCBSR Control word contd.

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



PC bit set / reset control word

Don't cares

Bit 4 of PC

Required instructions

MOV AL, 09H

OUT 7FH, AL

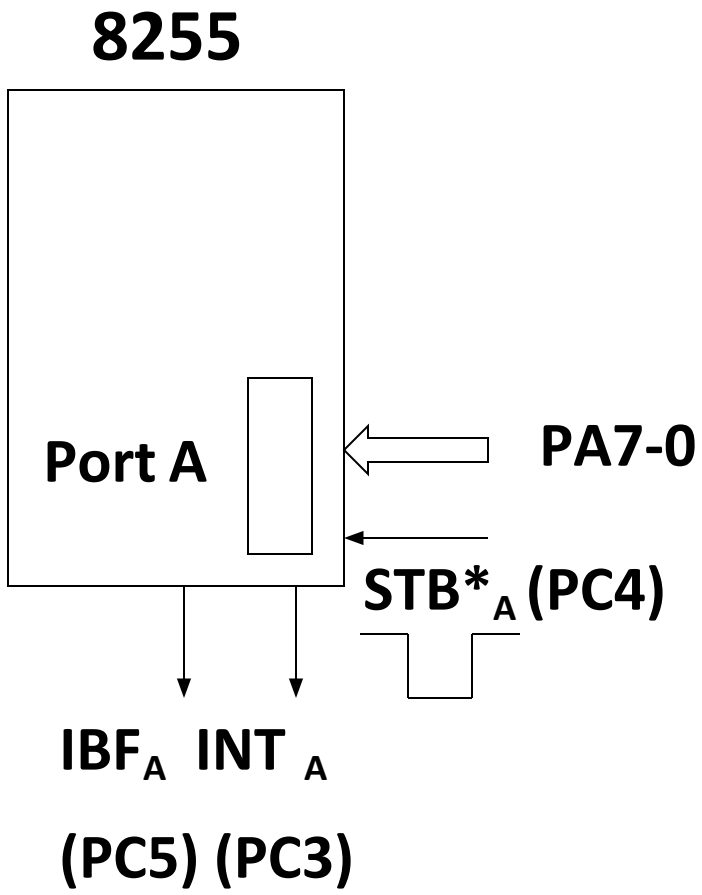
Handshake Interrupt i/p port

For Port A as handshake interrupt input port:

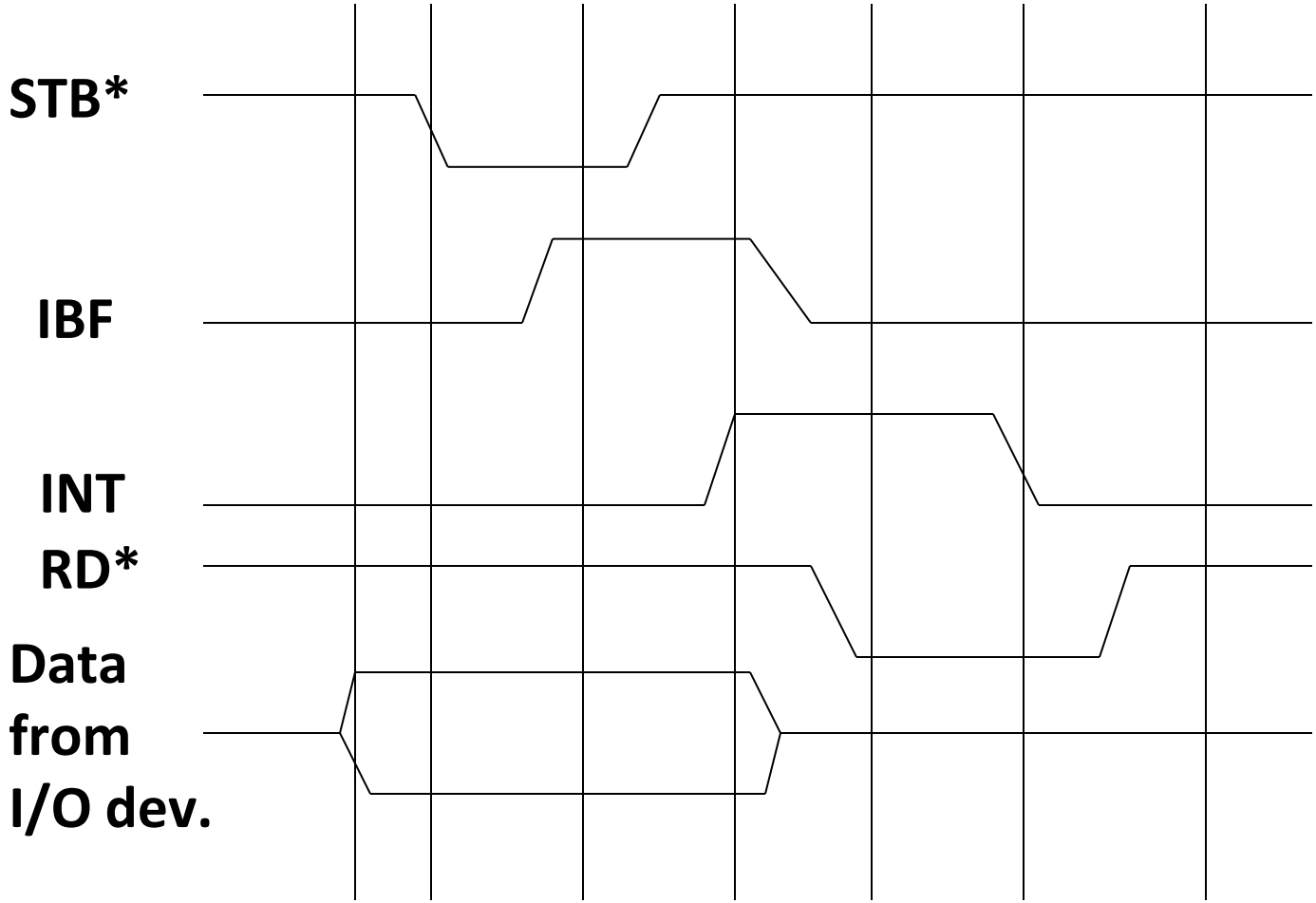
INT_A is PC3

STB^*_A is PC4

IBF_A is PC5



Handshake Interrupt i/p port



Handshake interrupt i/p port

When i/p device has data to send it checks if IBF (input buffer full) signal is 0.

If 0, it sends data on PB7-0 and activates STB* (Strobe) signal. STB* is active low.

When STB* goes high, the data enters the port and IBF gets activated.

If the Port interrupt is enabled, INT is activated. This interrupts the processor.

Processor reads the port during the ISS. Then IBF and INT get deactivated.

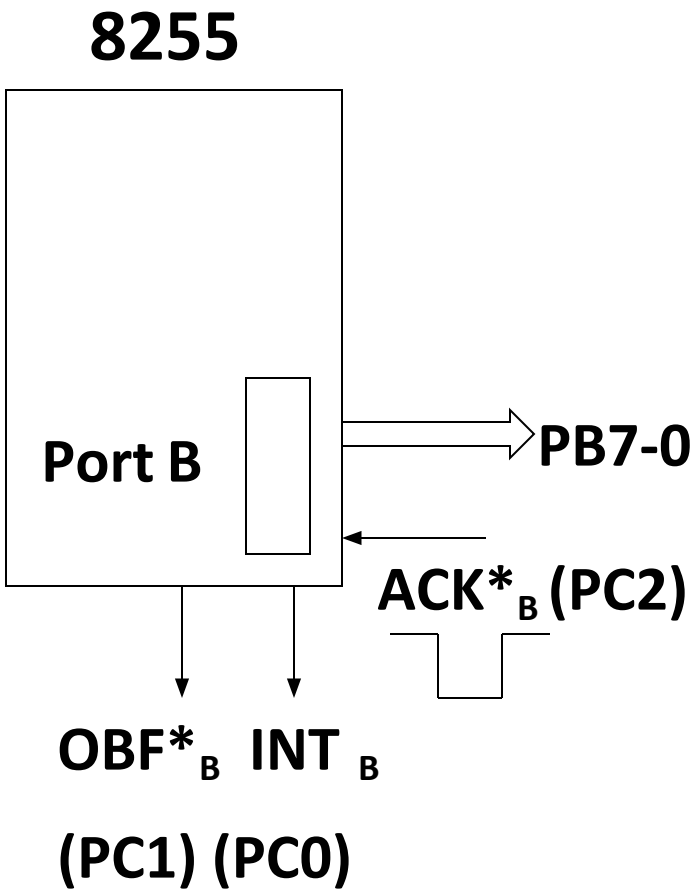
Handshake interrupt o/p port

For Port A as handshake interrupt output port:

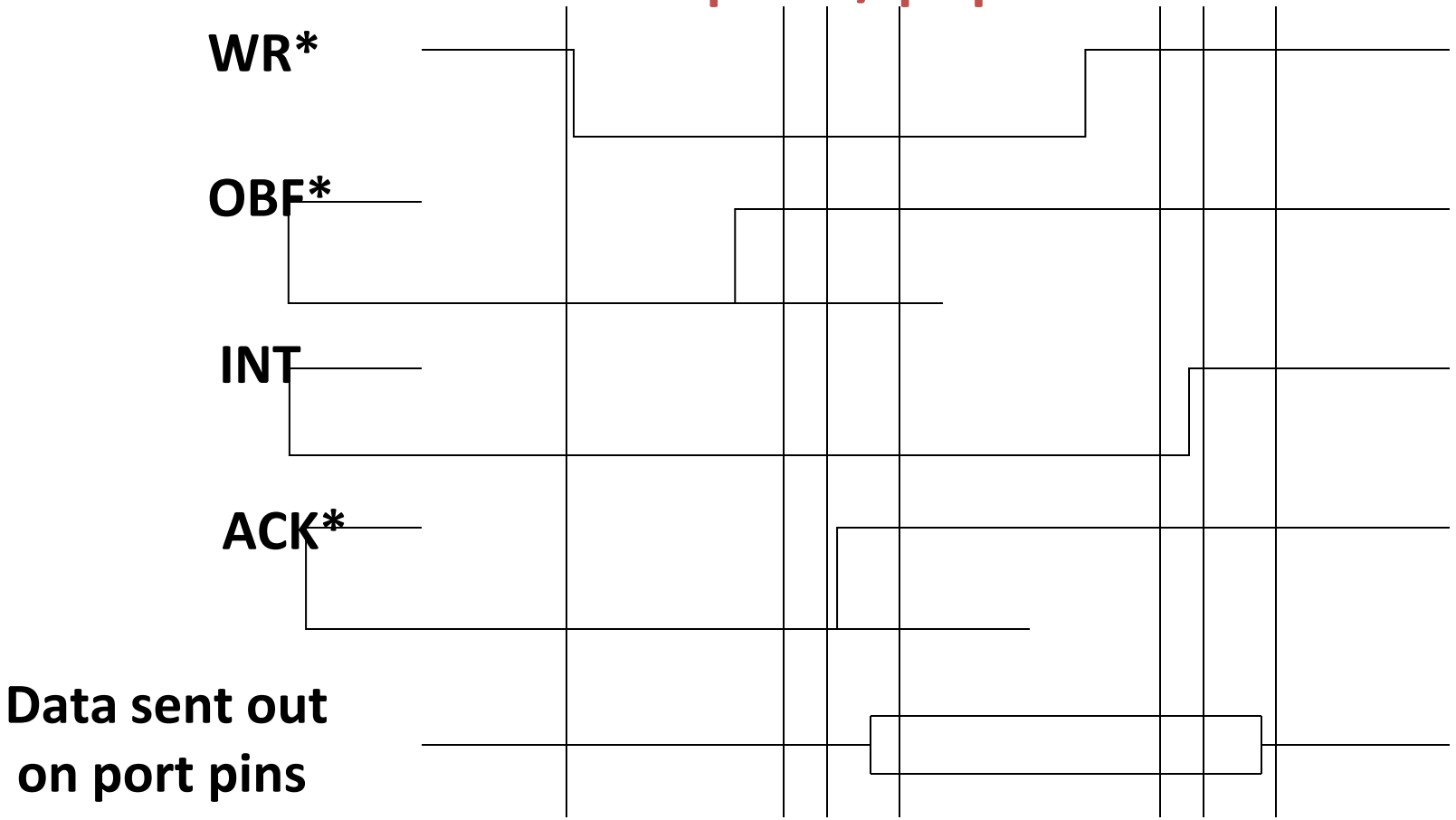
INT_B is PC0

ACK^*_B is PC2

OBF^*_B is PC1



Handshake interrupt o/p port



Handshake interrupt o/p port

When o/p device wants to receive data it checks if OBF* (output buffer full) signal is 0.

If 0, it receives data on PB7-0 and activates ACK* (Acknowledge) signal. ACK* is active low.

When ACK* goes high, the data goes out of the port and OBF* is set to 1.

If the Port interrupt is enabled, INT is activated. This interrupts the processor.

Processor sends another byte to the port during the ISS. Then OBF* and INT are reset to 0.

Handshake Status Check I/O

Interrupt is disabled for the port using PCBSR

Even if new data is entered into I/p buffer by I/O device INT o/p is not going to be activated for i/p operation

How processor knows that the i/p buffer has new data?

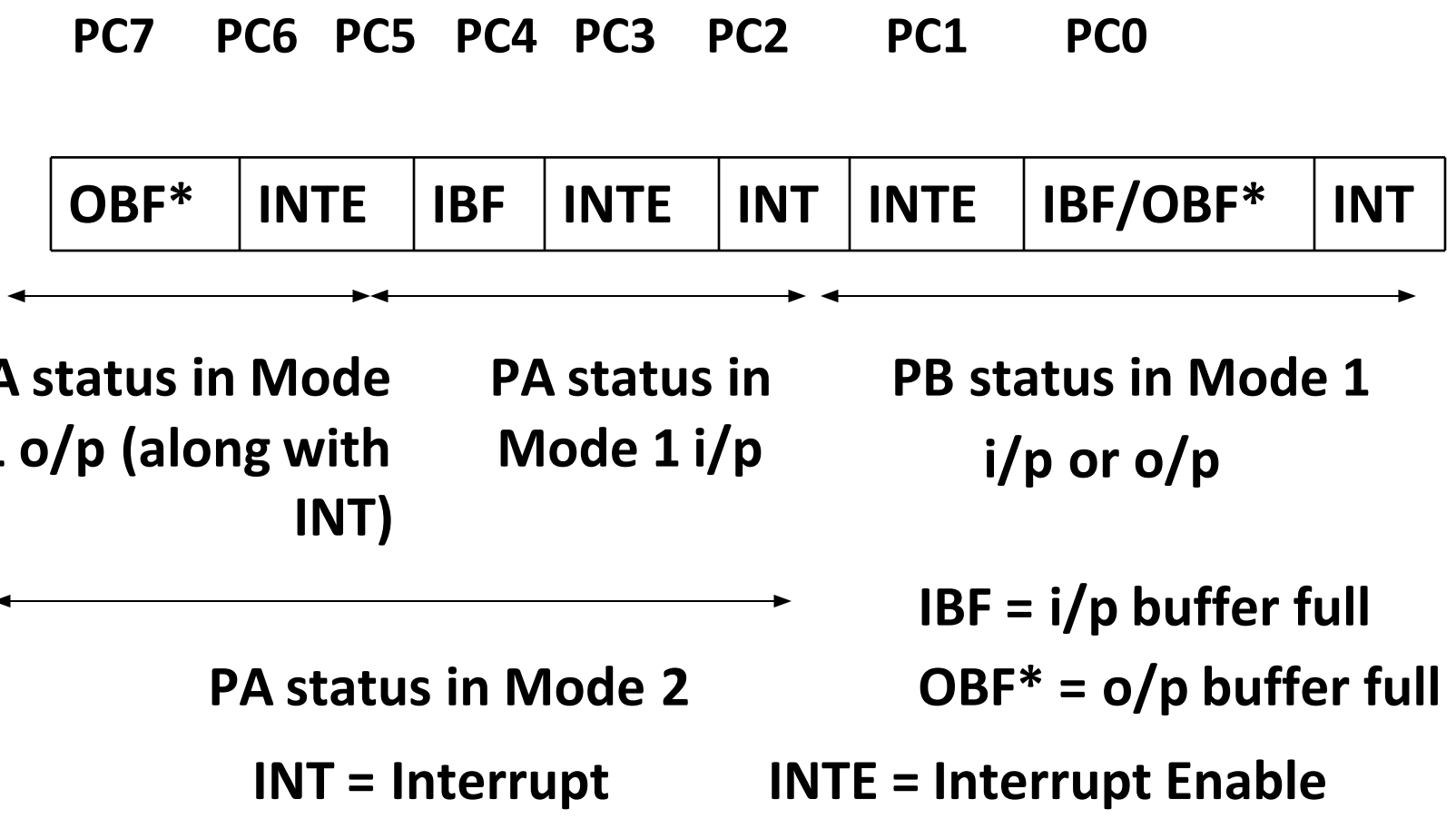
Even if I/O device has emptied the o/p buffer, INT o/p is not going to be activated for o/p operation

How the processor knows that the o/p buffer is empty?

Processor reads the status of the port for this purpose

Port C as provider of Status

PC provides status info of PA & PB when not in mode 0



Handshake status check i/p port

Suppose Port B is in mode 1 status check i/p

Processor reads bit 1 (IBF) of Port C repeatedly till it is set and then the processor reads Port B

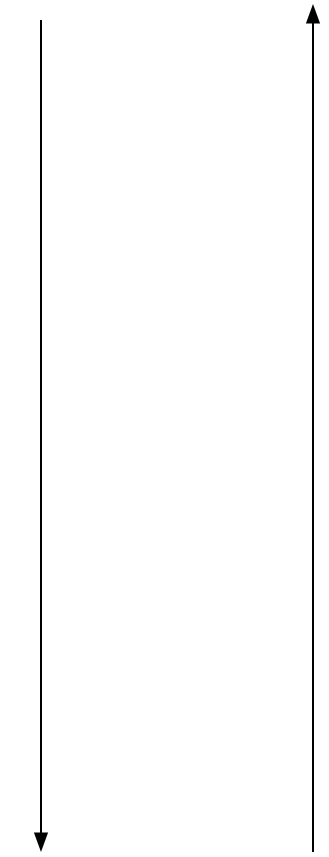
```
AGAIN:  IN AL, 7EH; Read Port C
        ROR AL, 1;
        ROR AL, 1;  Check bit 1 of Port C
        JNC AGAIN; If it is 0, repeat checking
        IN AL, 7DH ; Read from Port B
```

INTERFACING WITH STEPPER MOTOR

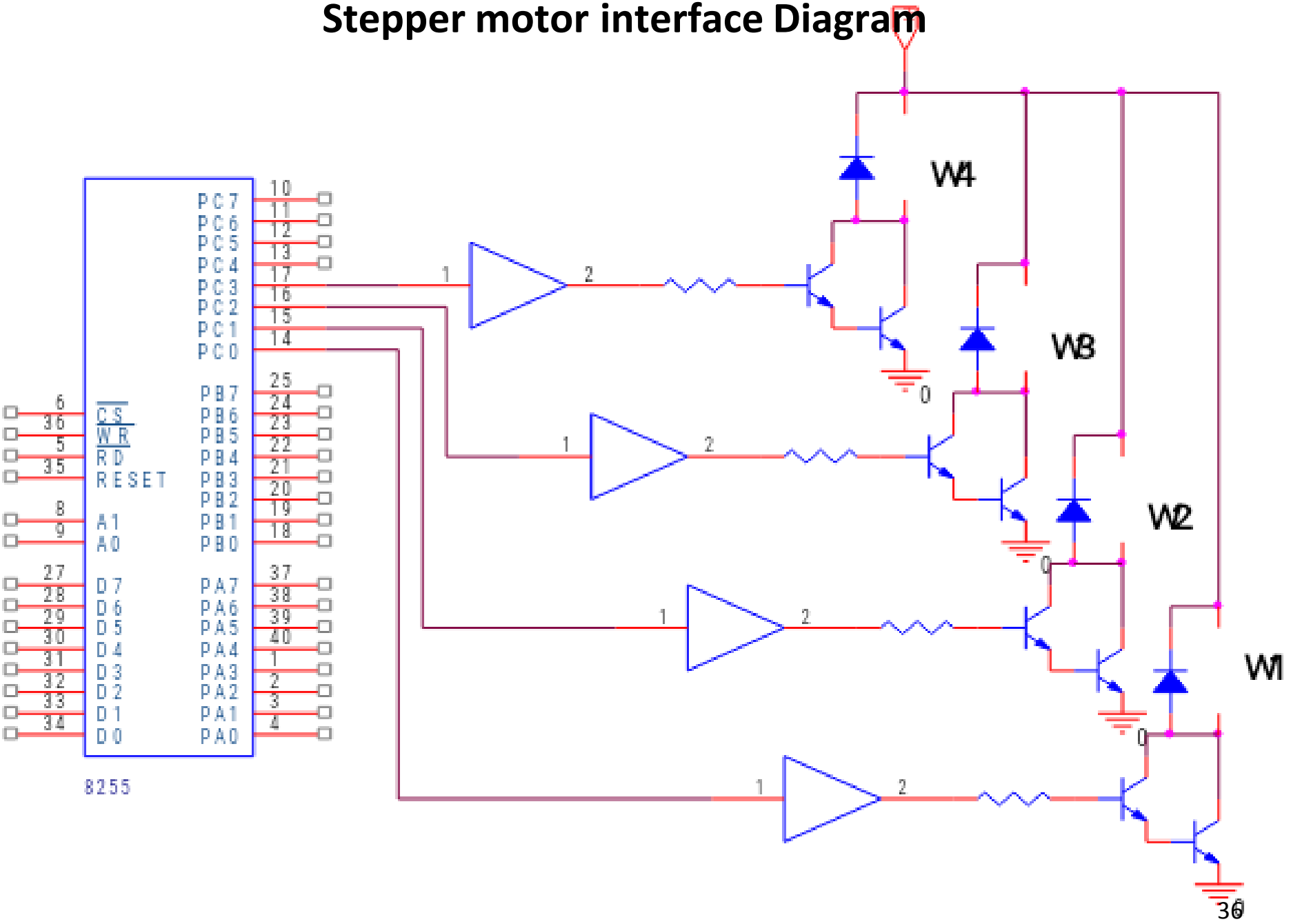
ROTATION PER SEQUENCE = 360/NT
NT= NUM.OF TURNS

FOUR PATTERN SWITCHING SEQUENCE

W4	W3	W2	W1
0	0	1	1
1	0	0	1
1	1	0	0
0	1	1	0
0	0	1	1



Stepper motor interface Diagram



PROGRAM TO ROTATE THE STEPPER MOTOR CONTINUOUSLY IN CLK.WISE DIRECTION FOR FOLLOWING SPECIFICATION

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 17X 100ns=1.7micro sec
No.of iteration(count) requires for 25m.sec delay = 25 x 1000/1.7 = 14705

SEND THE FIRST VALUE AS 33H. ROTATE IT BY ONE POSITION TO GET NEXT PATTERN.
33H IS CHOOSEN IN PLAC E OF 03H SO THAT ROTATION OF 8-BIT DATA GIVES CORRECT VALUE

SEND ALL PATTERNS AND CONTINUE THE SET OF PATTERN INDEFINITELY

PROGRAM

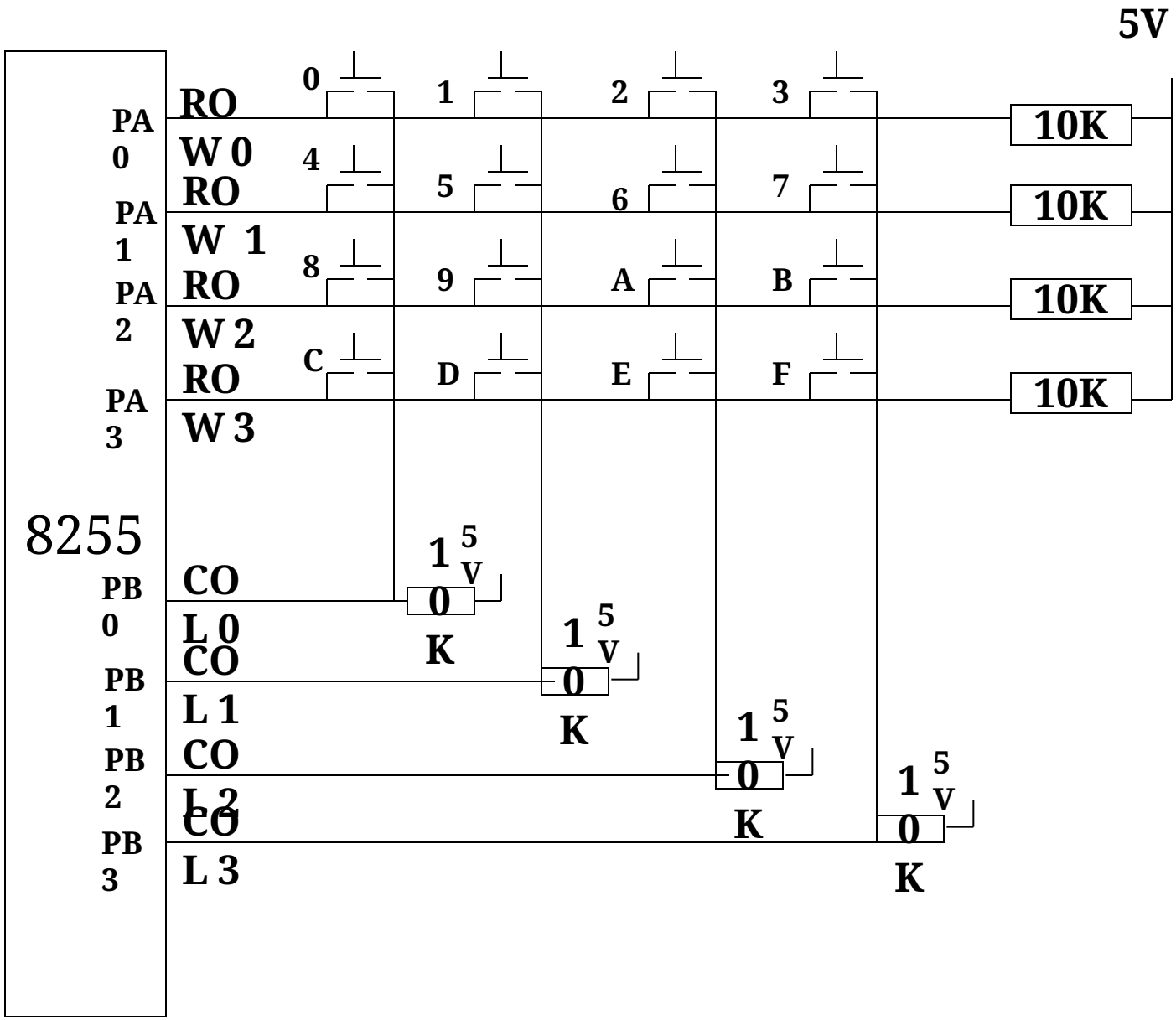
```
        DATA SEGMENT
        PORTC EQU 8004H
        CNTRLPRT EQU 8006H
        DELAY EQU 14705
        DATA ENDS
        CODE SEGMENT
        ASSUME CS: CODE, DS:DATA
START:  MOV AX,DATA
        MOV DS,AX
        MOV AL,80H          ;ALL PORTS AS O/P PORTS
        MOV DX,CNTRLPRT
BACK:   OUT DX,AL
        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
SELF:   LOOP SELF
        JMP BACK
        CODE ENDS
        END START
```

ROL INSTEAD OF ROR FOR COUNTER CLOCK WISE ROTATION

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

EACH STEP = 360/NT=360/200 = 1.8DEG
THERE FORE N = 180/1.8 = 100

INTERFACING KEYBOARD



8086 HAS TO

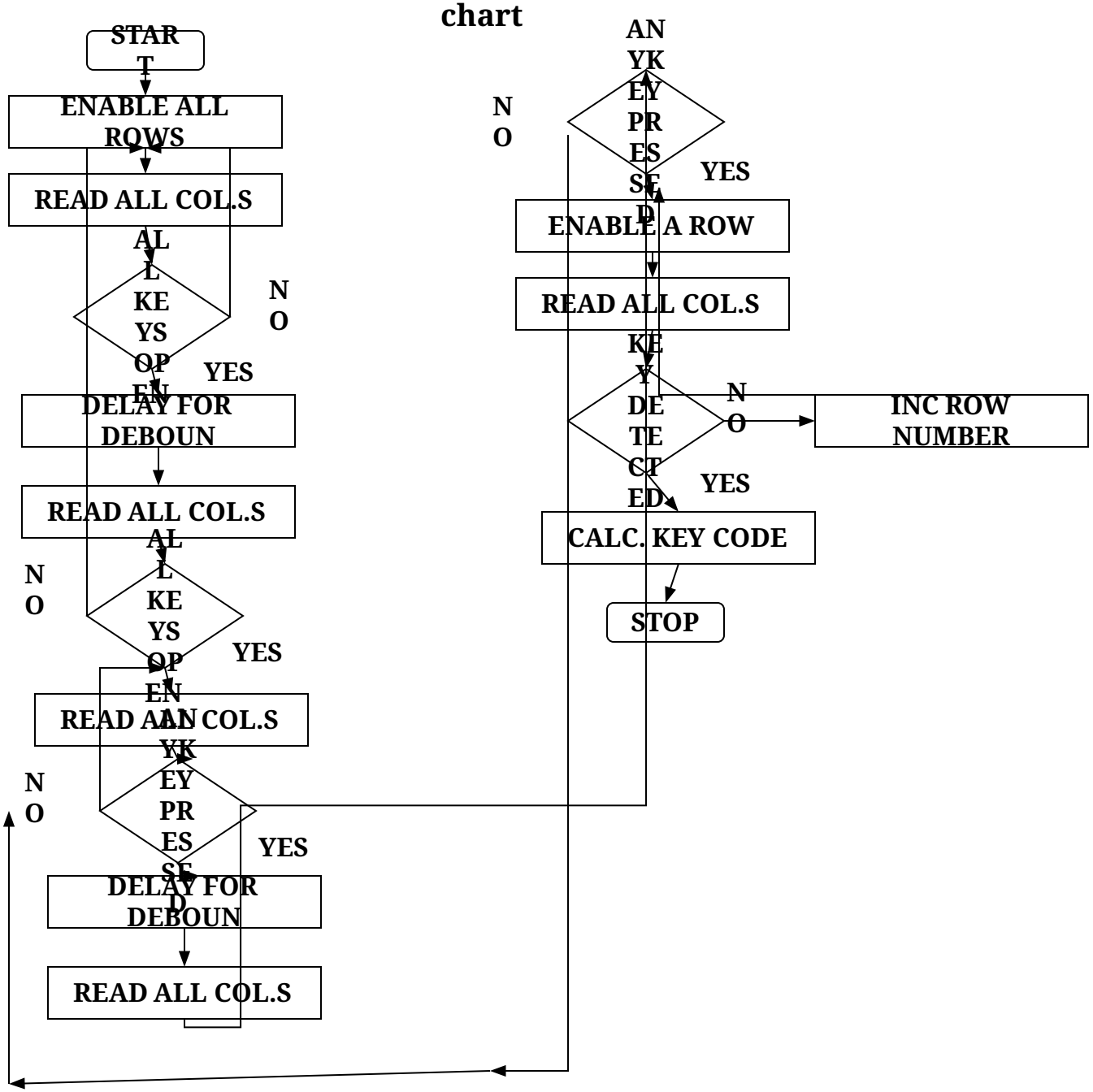
1. DETECT A KEY PRESS
2. DEBOUNCE A KEY PRESS
3. GENERATE A CODE CORRESPONDING TO THE KEY BEING PRESSED

SOFTWRAE ASPECTS

ALGORITHM

1. WAIT till all keys are released. Use s.w debounce for eac
   key check
2. Wait for key closure
3. Confirm key closure
4. Find number of row and column to which key belongs
5. Convert the row and col information to entry number o
   the
   table which contains ASCII code
6. Get code and repeat in infinite loop

Flow chart

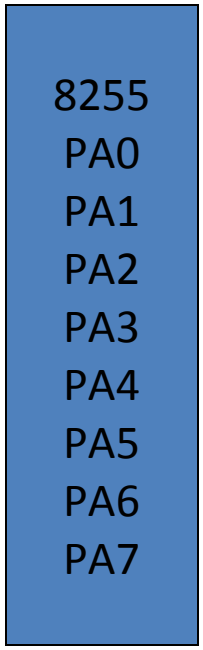

START
ENABLE ALL ROWS
READ ALL COL.S
ALL KEYS OPEN
NO
YES
DELAY FOR DEBOUN
READ ALL COL.S
ALL KEYS OPEN
NO
YES
READ ALL COL.S
ANY KEY PRESSED
NO
YES
DELAY FOR DEBOUN
READ ALL COL.S
ANY KEY PRESSED
NO
YES
ENABLE A ROW
READ ALL COL.S
KEY DETECTED
NO
YES
INC ROW NUMBER
CALC. KEY CODE
STOP

PROGRAM

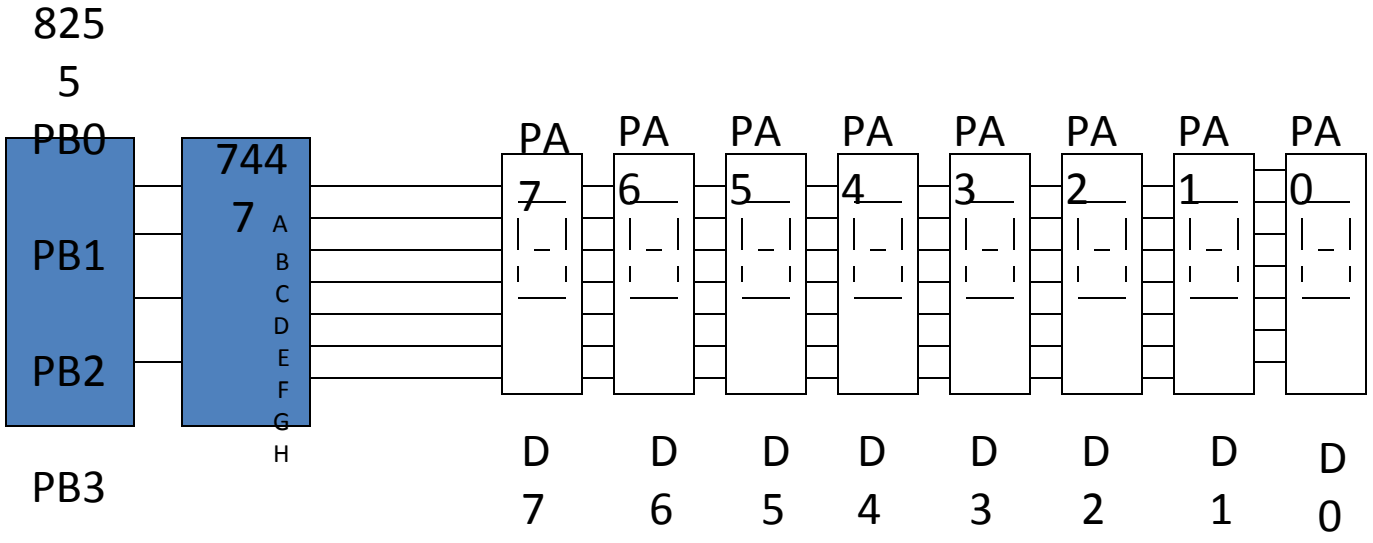
```
        DATA SEGMENT
        CNTRPRT EQU 8003H
        PORTA EQU 8000H
        PORTB EQU 8001H
        DELAY EQU 6666
        TABLE DB 30H,31H,32H,…..39H,41H,….46H ;ASCII CODES FROM 0 TO F
        DATA ENDS
        CODE  SEGMENT
        ASUUME CS:CODE,DS:DATA
START:  MOV AX,DATA
        MOV DS,AX
        MOV AL,82H
        MOV DX,CNTRPRT      :PORT A AS I/P PORT PORT B AS O/P PORT
        OUT DX,AL
        XOR AL,AL
        MOV DX,PORTA
        OUT DX,AL           ;ENABLE ALL ROWS
        MOV DX,PORTB
RDCOL:      IN AL,DX            ;GTE COL STATUS
        AND AL,0FH          ;MASK UNWANTED BITS
        CMP AL,0FH          ;GET READY FOR CHKING COL SATTUS
        JNE RDCOL           ;IS ANY COL ACTIVE?IF YES CHK AGAIN
        MOV CX,DELAY        ;NO DEBOUNCE DEALY
SELF:   LOOP SELF
        IN AL,DX
        AND AL,0FH          ;CONFIRM COL STATUS AGAIN
        CMP AL,0FH
        JNE RDCOL           ;IF NOT CONFIRMED CHECK AGAIN
RDAGN:      IN AL,DX            ;CONFIRMED THAT ALL KEYS ARE OPEN,GET COL STATUS AGAIN
        AND AL,0FH
        CMP AL,0FH          ;CHECK FOR ANY KEY CLOSURE,IF NO CONTINUE TO CHECK,IF YES
        JE RDAGN            ;NEXT STEP
        MOV CX,DELAY
SELF1:  LOOP SELF1
```

```
        IN AL,DX
        AND AL,0FH          ;CONFIRM COL STATUS AGAIN
        JE RDAGN
        MOV AL,0FEH ;KEY CLOSURE CONFIRMES,SELECT ROW PATTTERN TO ENABLE A ROW
        MOV BL,AL           ;SAVE IT
ENROW:  MOV DX,PORTA
        OUT DX,AL           ;ENABLE CORRESPONDING ROW
        MOV DX,PORTB
        IN AL,DX            ;GET COL STATUS
        AND AL,0FH
        CMP AL,0FH          ;CHECK IF COL IS ACTIVE
        JNE CCODE           ;IF YES, GO TO CALCULATE ASCII CODE OF KEY PRESSED
        ROL BL,1            ;PREPARE TO ENABLE NEXT ROW
        MOV AL,BL
        JMP ENROW
CCODE:  MOV CL,0            ;AL CONTAINS COL PATTERN,BL CONTAINS ROW PATTERN
                ;INITIALIZE COL COUNT TO 0
NXTCOL: ROR AL,1            ;COL STATUS GOES TO CARRY FLAG
        JNC CHKROW  ;IS COL ACTIVE, IF YES, CL CONTAINS COL.NUMBER
        INC CL      ;NO INCREMENT COL COUNT
        JMP NXTCOL  ;CHECK NEXT COL
CHKROW: MOV DL,0            ;CL CONTAINS COL NUMBER
                ;INITIALIZE ROW COUNT TO ZERO
NXTROW: ROR BL,1            ;ROW STATUS GOES TO CARRY FLAG
        JNC CALADR  ;IS ROW ACTIVE? IF YES, DL CONTAINS ROW NUMBER
        ADD DL,04H          ;ROW COUNT+4 ?ROW COUNT
        JMP NXTROW CHECK NEXT ROW
CALADR: ADD DL,CL           ;ROW +COL
        MOV AL,DL
        LEA BX,TABLE
        XLAT        ;GET ASCII CODE OF THE KEY PRESSED
        INT3H
        JMP START
        CODE ENDS
        END SATRT
```

INTERFACING THE LED DISPALY



CONNECT PA TO DISPLAY THROUGH PNP TRANSISTOR



ALGORITHM

1. TURN ON Q0 BY APPLYING A LOGICAL LOW TO BASE OF PNP TRANSISTOR
2. SEND 7-SEGMENT CODE FOR D0 (DIGIT 0)
3. AFTER 1MS TURN OFF QO,TURN ON Q1, OFF QO,Q2-Q7
4. SEND 7-SEGMENT CODE FOR D1(DIGIT 1)
5. AFTER 1MS TURN OFF Q1,TURN ON Q2 REMAINING Q'S OFF
6. REPEAT THE PROCESS FOR ALL 8-DIGITS.IT COMPLETES ONE CYCLE
7. START CYCLE AGAIN

PROGRAM

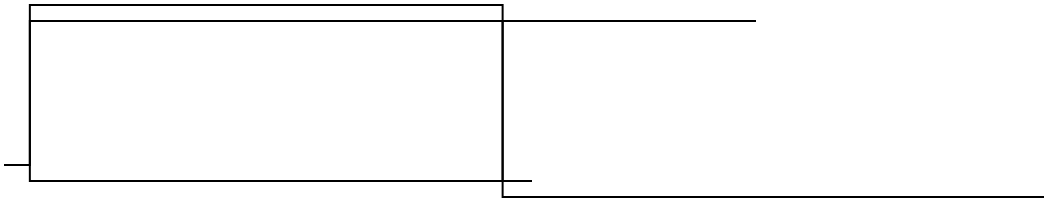
```
            DATA SEGMENT
            PORTA EQU 0FFF8H
            PORTB EQU 0FFF9F
            CTRLPORT EQU 0FFFBH
            DELAY EQU 012CH
            DIGITS DB 1,2,3,4,5,6,7,8
            DATA ENDS
            CODE  SEGMENT
            ASSUME CS:CODE,DS:DATA
START:      MOV AX,DATA
            MOV DS,AX
            MOV DX,CNTRLPORT        ;PORTA ,PORTB O/P PORTS
            MOV AL,80H
            OUT DX,AL
REPEAT:     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
            OUT DX,AL               ;SEND IT TO 7447 FOR DISPLAY
SELF:       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
```

D TO A CONVERTER

D/A CONVERTER CAN BE DIRECTLY CONNECTED TO 8255

LET US ASSUME THAT 8-BIT D/A CONVERTER USED IS HAVING FULL SCALE O/P VOLTAG EOF 0-5V. IT IS CONNECTED TO PORT A OF 8255. THE BASE ADDRESS SOF 8255 IS 8000H. CLOCK FREQUENCY IS 5MHZ

GENERATE A SQUARE WAVE OF 5VOLTS, 1KHZ FREQ



5VOLTS,500MICR
O SEC

ALGORITHM

- SEND A VALUE 0 TO PORT A
- DELAY 500MICRO SEC
- SEND A VLAUE FFH TO PORT A(FOR +5V)
- REPEAT CYCLE INDIFINITELY

DELAY CALCULATIONS

LOOP INSTRUCTION USED FOR GENERATING REQUIRED DELAY,TAKES 17 CYCLES
TIME FOR 17 CYCLES = 17 X 200ns(CPU FREQ = 5MHZ, 1 CYCLE = 200NS)
- 3.4 MICRO SEC
HENCE ONE LOOP INSTRUCTION = 3.4 MICRO SEC
DELAY REQUIRED = 500MICRO SEC
LOOP INSTRUCTION SHOULD BE REPEATED FOR N WHERE
N = 500/3.4 = 147

PROGRAM

```
DATA SEGMENT
PORT EQU 8000H
CNTPRT EQU 8003H
DELAY EQU 147
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START:      MOV AX,DATA
MOV DS,AX
MOV AL,80H
MOV DX,CNTPRT
OUT DX,AL
MOV DX,PORTA
BACK:       MOV AL,00
OUT DX,AL
MOV CX,DELAY
SELF: LOOP SELF
MOV AL,0FFH
OUT DX,AL
MOV CX,DELAY
SELF: LOOP SEWLF
JMP BACK
INT 3H
CODE ENDS
END START
```

GENERATE RECTANGULAR WAVE OF 1V TO 4V,25% DUTY CYCLE, 500KHZ FREQ

ALGORITHM

1. SEND A VALUE CORRESAPONDING TO 1VOLT TO PORT A
2. AFTER 1500 MICRO SEC DELAY SEND
   A VALUE CORRESPONDING TO 4VOLTS TO PORT A
3. AFTER 500 MICRO SEC SEND FIRST VALUE(CORRESPONDING TO 1VOLT)
4. REPEAT CYCLE INDIFINITELY

DELAY CALCULATIONS

DELAY CONSTANT FOR 500 MICRO = 147
DELAY CONSTANT FOR 1500 MICRO = 147 X 3 = 441
BINARY VALUE FOR 5VOLT = FFH
BINARY VALUE FOR 1 VOLT = FF/5H= 255/5 = 51 = 33H
BINARY VALUE FOR 4VOLTS = 33H X 4 = CCH

PROGRAM

```
DADA SEGMENT
PORT EQU 8000H
CNTPRT EQU 8003H
DELAYH EQU 147
DELAYL EQU 441
LVOLT DB 33H
HVOLT DB 0CCH
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START:      MOV AX,DATA
MOV DS,AX
MOV AL,80H
MOV DX,CNTPRT
OUT DX,AL
BACK:       MOV AL,LVOLT
MOV DX,PORTA
OUT DX,AL
MOV CX,DELAYL
SELF: LOOP SELF
MOV AL,HVOLTH
OUT DX,AL
MOV CX,DELAYH
SELF: LOOP SEWLF
JMP BACK
INT 3H
CODE ENDS
END START
```

GENERATE TRIANGULAR WAVE OF 0 TO 5V

ALGORITHM

1. SEND A VALUE CORRESPONDING TO 0V ON PORT A
2. INCREMENT THE VALUE BY 1 AND KEEP SENDING IT TILL IT REACHES HIGH VOLTAGE
3. DECREMENT THE VALUE BY 1 AND KEEP SENDING IT TILL VALLU REACHES 0VOLT
4. INCREMENT AGAIN AND REPEAT THE CYCLE INDIFINITELY
5. BINARY VALUE FOR 0V = 00H
6. BINARY VALUE FOR 5V =FFH

PROGRAM

```
DADA SEGMENT
PORT EQU 8000H
CNTPRT EQU 8003H
DELAYH EQU 147
DELAYL EQU 441
LVOLT DB 00H
HVOLT DB 0FFH
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE,DS:
DATA
START:      MOV AX,DATA
MOV DS,AX
MOV AL,80H
MOV DX,CNTPRT
OUT DX,AL
MOV AL,LVOLT
MOV DX,PORTA
BACK:       OUT DX,AL
INC AL
CMP AL,HVOLT
JNZ BACK
BK:OUT DX,AL
DEC AL
CMP AL,LVOLT
JNZ BK
JMP BACK
INT 3H
CODE ENDS
END START
```

GENERATE STAIRCASE WAVE WITH THE FOLLOWING SPECIFICATIONS
NUM.OF STEPS = 5
HEIGHT OF STEP = 1VOLT
WIDTH OF STEP = 5MILLI SEC

ALGORITHM

1. **SEND A VALUE OF 0 CORRESPONDING TO 0 VOLTS TO PORT A**
2. **GIVE DELAY OF 5 MILLI SEC**
3. **CALCULATE NEXT VALUE BY ADDING STEP HEIGHT**
4. **SEND IT TO PORT A AND DELAY AGAIN**
5. **REPEAT THIS TILL ALL STEPS ARE OVER**
6. **CONTINUE THE CYCLE INDIFINITELY**

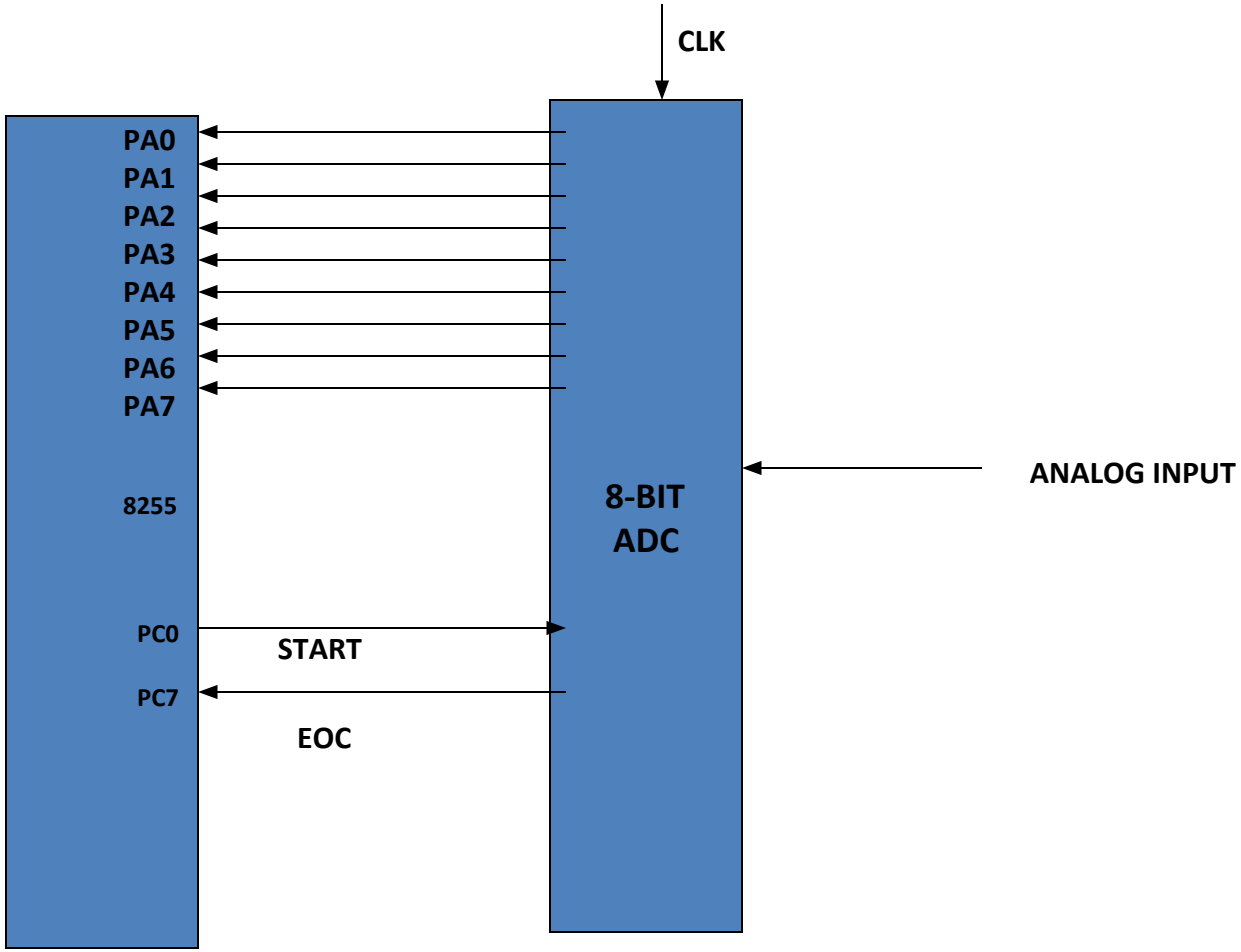
DEALY CALCULATIONS

3.4 MICRO SEC X DELAY CONSTANT = 5000 MICRO SEC
DELAY CONSTANT = 5000 MICRO SEC/ 3.4 = 1470
STEP HEIGHT = 1 VOLT = FF/5 H = 255 / 5 = 51 = 33H
(LVOLT) LOW VALUE = 0H
HVOLT HIGH VALUE = 0FFH

PROGRAM

```
DATA SEGMENT
PORTA EQU 8000H
CNTPRT EQU 8003H
LVOLT EQU 0H
HVOLT DB 0FFH
STEPH DB 33H
STEPCNT DB 06H ; NO.OF STEPS PLUS ONE = STEPCOUNT
DELAY EQU 1470
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START:      MOV AX,DATA
MOV DS,AX
MOV AL,80H
MOV DX,CNTPRT
OUT DX,AL
MOV AL,LVOLT
MOV DX,PORTA
BEGIN:      MOV BL,STEPCNT
MOV AL,00H
BACK:        OUT DX,AL
MOV CX,DELAY
SELF: LOOP SELF
ADD AL,STEPH
DEC BL
JNZ BACK
JMP BEGIN
INT 3H
CODE ENDS
END START
```

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
```