No. #2 : Parallel Interfacing with Microprocessor Based System (4 Pts)

Interface :

→ Definition, Needs, Types : Serial & Parallel

→ Parallel Interface : Higher data rate, expensive, complex, multiple wires, short distance

2.1 : Methods of Parallel Data Transfer : Simple Input & Output, Strobe I/O, Single Handshake I/O & Double Handshake I/O :

Simple I/O :

→ used if data is always present or I/O device is always present & read such as switch, thermostat as input or LED as output.

→ Crossing on data lines : new data byte becomes valid

→ does not depend on other signals. [see figure]

Simple Strobe I/O :

→ used if valid data is present only at a certain time & must be read at that time, such as ASCII-encoded keyboard

→ when a key is pressed, circuitry on the keyboard sends out the ASCII code of pressed key on eight parallel data lines & then sends out a strobe signal on another line to indicate that valid data is present.

→ This transfer is time dependent - reading of data when strobe pulse is generated.

→ The sending device outputs parallel data on the data lines & then outputs strobe ($\overline{STB}$) signal to indicate the presence of valid data.

→ [see figure]

Single Handshake I/O :

→ Input to Microprocessor :

→ Peripheral outputs some parallel data & sends $\overline{STB}$ signal to up.

→ The up detects asserted $\overline{STB}$ signal & reads in the byte of data.

→ Then then up sends an Acknowledge signal (ACK) to the peripheral to indicate that the data has been read.

→ Output from Microprocessor : (To a printer)

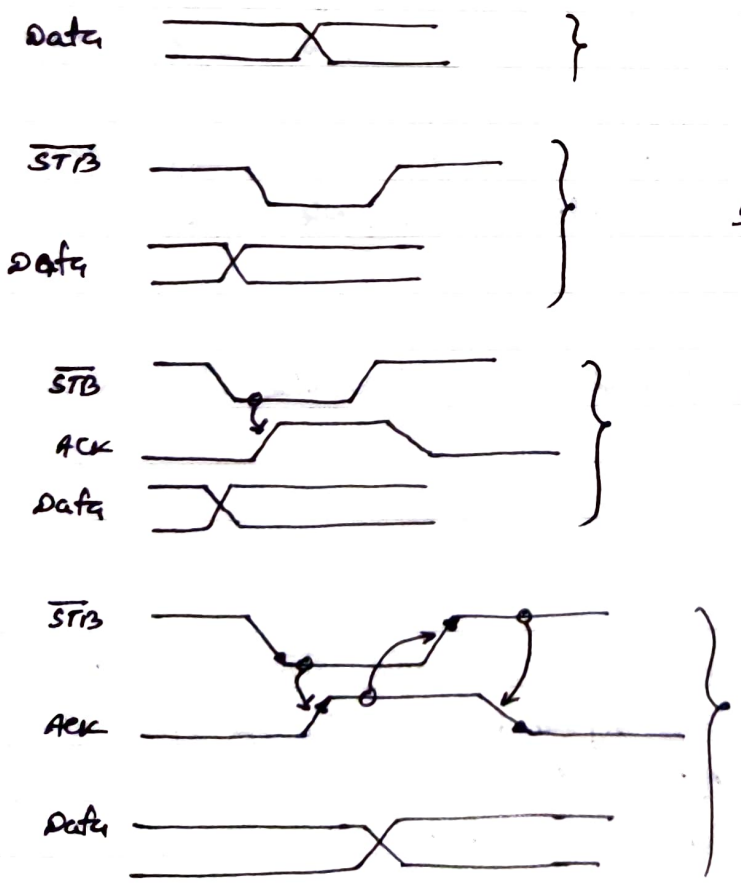
→ The up outputs a character to the printer & asserts a $\overline{STB}$ signal

No. Date

to the printer to tell the printer, "Here is a character for you".
→ when printer is ready, it answers back with ACK signal to tell the processor, "I got that one."
→ [see figure]

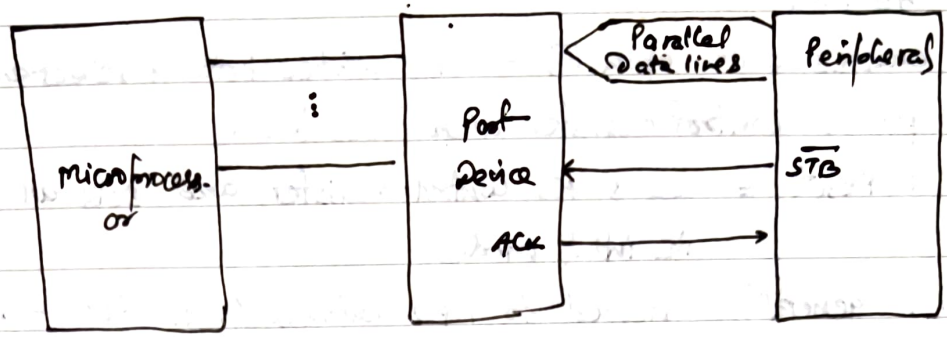
Double Handshake I/O :

- used when even more coordination between sender & receiver is required.
- Each signal (ACK, $\overline{STB}$) edge has meaning.
- The sending device asserts $\overline{STB}$ line to low - Are you ready ?
- The receiving " raises ACK " " high - I am ready.
- The sending " then sends its some valid ~~for you~~ data & raises its $\overline{STB}$ line to high - Here is some valid data for you.
- After it has read in the data, the receiving device drops the ACK line to low - I have the data, waiting your response for next data.
- [see figure]



No. Date

Illustration : Handshake Data Transfer



2.2 8255 as a General Purpose Programmable I/O Device and Its ~~Read~~ Interfacing Examples

- Widely used, programmable, parallel I/O device.
- Can be programmed to ~~a~~ transfer but under various conditions, from simple I/O to interrupt I/O.
- Flexible, versatile, economical but complex.
- can be used ~~as~~ with any microprocessor
- Has 24 I/O pins = two 8-bit parallel ports : A & B, with the remaining eight bits as port C.
- Port C - Can be used as individual bits or grouped in two 4-bit ports : Upper & Lower.
- Functions of these ports - Writing a control word in control register.

Block Diagram :

- see figure & explain !

(1.) Data Bus Buffer

- Tri-state, bidirectional 8-bit buffer used to interface 8255 to the system bus of a up.
- Data is transmitted or received by the buffer.
- Also used to carry control words & status information.

(2.) Read/Write Control logic :

- To manage transfer of data, control word & status words.
- Receives address & control bus from up & issues commands to both of the control groups.

No. Date

(i) $\overline{RD}$: enables read operation - MPU reads data from the selected I/O port when low.

(ii) $\overline{WR}$: enables write operation - MPU writes into a selected I/O port or control register when goes low.

(iii) RESET : Reset = clears the control register and sets all ports in input mode.

(iv) A_1, A_0 : generally connected to MPU address lines A_1 & A_0. used to select a port or control register.

(v) $\overline{CS}$: chip select - enables communication between MPU & 8255 when low, master chip select

$\overline{CS}$	A_1	A_0	Selected
0	0	0	Port A
0	0	1	" B
0	1	0	" C
0	1	1	Control Register
1	X	X	8255 is not selected

(3.) Group A & Group B :

→ Explain.

Modes of operation

→ 3 operational modes of 8255

(1.) Mode 0 :

→ Simple input or output without handshaking

→ If both ports A & B are in mode 0 - two halves of port C can be used together as an additional 8-bit port or they can be used individually as two 4-bit ports independently.

→ When used as output, port C lines can be individually set or reset by sending a special control word to control register address.

No. Date

(2.) Mode 1 :

- used if port A or B for a handshake (strobed) input or output
- Some pins of port C function as handshake lines.
- Port B in mode 1 → PC_0, PC_1, PC_2 function as handshake lines
- Port A " " " input – PC_3, PC_4, PC_5 " " " "
- " " " " " output – PC_3, PC_6, PC_7 " " " "
- Rest of port C pins ⇒ as input or output

(3.) Mode 2 :

- Only port A can be initialized - bidirectional handshake data transfer
- Port A in mode 2 = PC_3-PC_7 for handshake lines
  - others : I/O port of or handshake for port B.

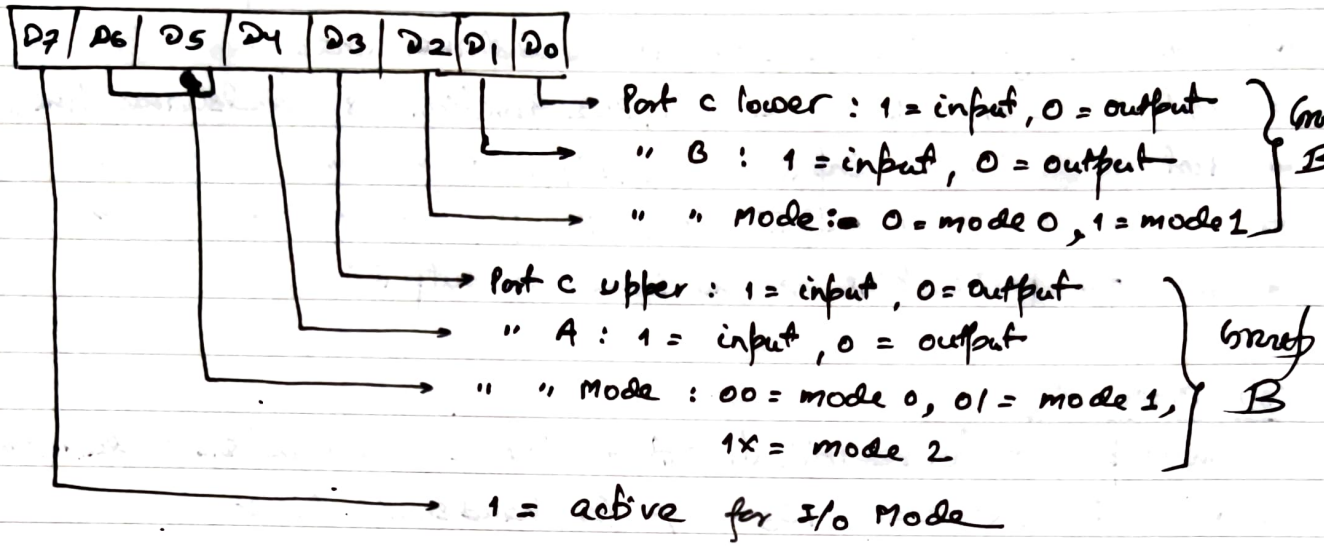
Control Word

- Content of control register = Specify an I/O operation for each port.
- Can be accessed to write a control word when A_0 & A_1 are 1 logic.
- Not accessible for a read operation.
- Bit D_7 of control register = either I/O function or Bit Set/Reset function
- $D_7 = 1$ → I/O functions in various modes
  - $D_7 = 0$ → Port C operates in Bit set/Reset (BSR) mode.
- BSR mode = not alters the functions of port A & port B.

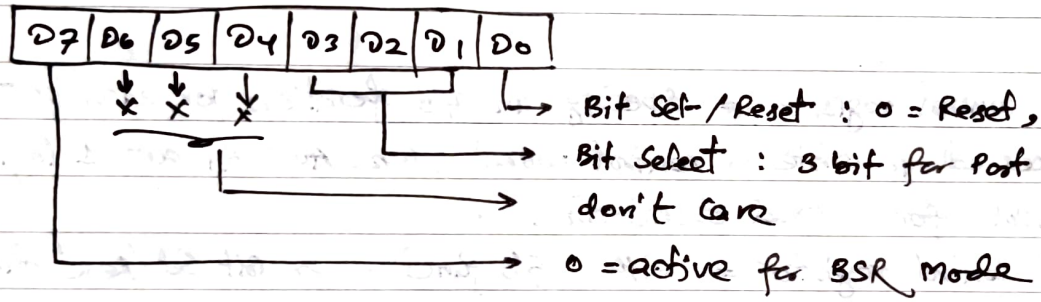
- To communicate with peripherals through 8255, three steps are necessary :
  (1.) Determine the address of ports A, B & C and control register according to $\overline{CS}$, A_1, A_0.
  (2.) Write a control word in the control register.
  (3.) Write I/O instructions to communicate with peripherals through ports A, B & C.

No. Date

Control Word for I/O Mode



Control Word for Port C BSR



Show various examples for control words

8255 Programming

- High on RESET = all ports in input mode, all flip-flops are cleared & interrupts are reset
  = remains same even RESET goes low.
- for any mode = sending out a single output instruction to control register
- The BSR word can also be used for enabling & disabling interrupt signals generated by Port C when 8255 is programmed for mode 1 & mode 2 by setting or resetting the associated bits of their interrupts.

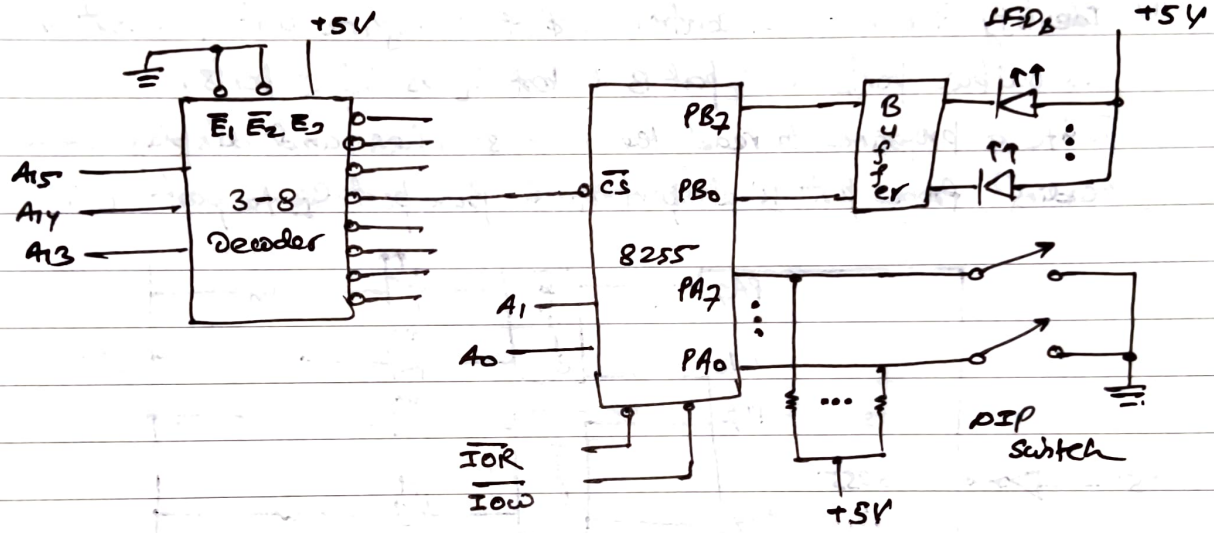
No. Date

Programming in Mode 0 (Basic I/O Mode)

→ Outputs are latched, Inputs are not latched, ports do not have handshake or interrupt capability.

Examples: (1:)

1. Identify port addresses.
2. Identify the Mode 0 control word to configure port A as an input port & port B as output port.
3. Write a program to read DIP switches & display the reading from port A ~~to~~ at port B.



Solution

1. → I/O - mapped I/O → A_{15} - A_8 and AD_7 - AD_0 carry same signals.

→ To enable $\overline{CS}$, $A_{15} = 0$, $A_{14} = 1$, $A_{13} = 0$.

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
0	1	1	x	x	x	x	x	x	x	x	x	x	x	.	.

So, 011 xxx .. = 011 000 ..

∴ Port A Address = 0110 0000 = 60H

" B " = 0110 0001 = 61H

" C " = 0110 0010 = 62H

Control Register " = 0110 0011 = 63H.

2. Control word : 1001 x0 0x = 1001 0000 = ~~92H~~ 90H

No. Date

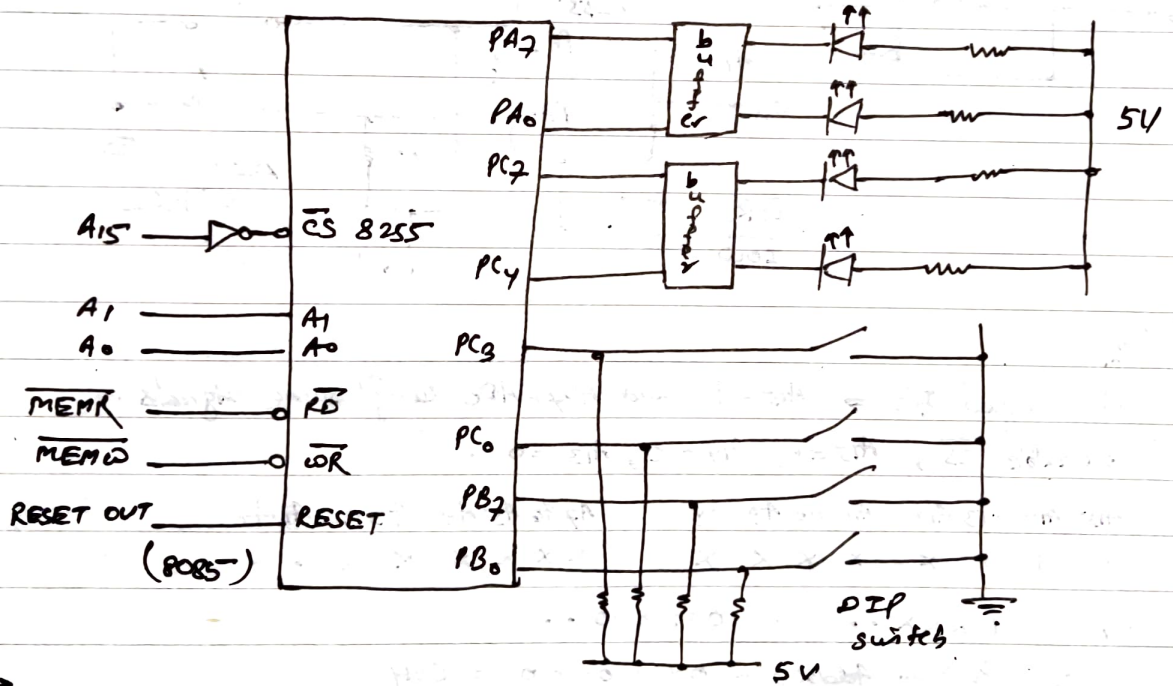
3. ⇒ Program Subroutine :

```
MVI  A, 90H ;  Load Acc with control word
OUT  63H    ;  Write control word into control register
               to initialize ports
IN   60H    ;  Read switches at port A
OUT  61H    ;  Display the reading at port B
RET
```

(2.) (a) Identify the port addresses.

(b) Identify the Mode 0 control word to configure port A & port C_U as output ports and port B & port C_L as input ports.

(c) Write a program to read the DIP switches and display the reading from port B at port A & from port C_L to port C_U.



⇒

(a) Memory-mapped I/O, A15 = 1 enables $\overline{CS}$

all don't care = 0 then

Port A address = 1000 0000 0000 0000 = 8000H

" B " = 1000 0000 0000 0001 = 8001H

" C " = 1000 0000 0000 0010 = 8002H

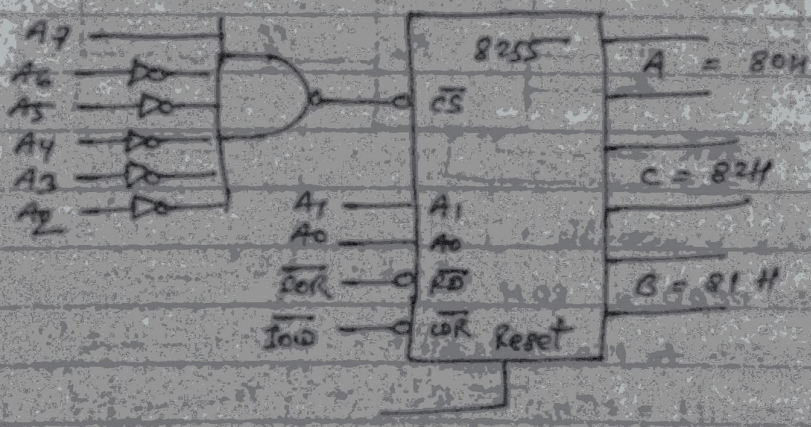
Control register " = 1000 0000 0000 0011 = 8003H

(b) Control word = 10000011 = 83H

(c) Program:

```
MVI  A, 83H   ; load acc with control word
STA  8003H    ; write word into control reg. to initialize ports
LDA  8001H    ; Read switches at port B
STA  8000H    ; Display the reading at port A
LDA  8002H    ; Read switches at port C
ANI  0FH      ; Mask upper as not input data
RLC           ; Rotate
RLC
RLC
RLC
STA  8002H    ; Display at port C
HLT
```

(3) Write a BSR control word subroutine to set bits PC_7 & PC_3 and reset them after 10 ms.



→ BSR control words:

To set PC_7 = 0 000 111 1 = 0FH

" reset " = 0000 111 0 = 0EH

" set PC_3 = 0000 011 1 = 07H

" reset " = 0000 011 0 = 06H

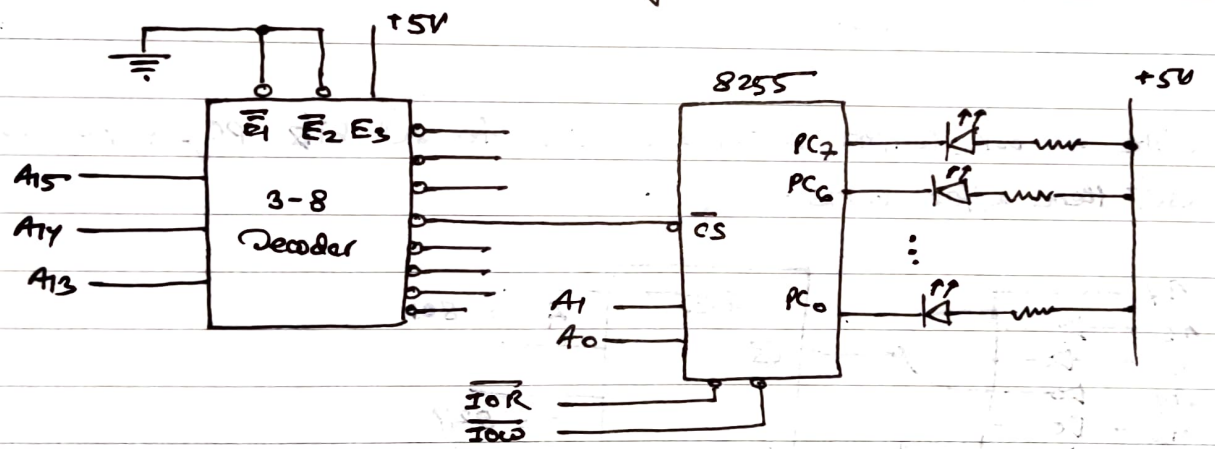
→ Port address: Control register address = 1000 0011 = 83H

No. Date

➡ Subroutine :

```
MVI  A, 0FH   ; load byte in acc to set PC7
OUT  83H      ; set PC7 = 1
MVI  A, 07H   ; load byte in acc to set PC3
OUT  83H      ; set PC3 = 1
CALL Delay    ; 10 ms delay subroutine
MVI  A, 06H   ; load byte in acc to reset PC3
OUT  83H      ; reset PC3 = 0
MVI  A, 0EH   ; load byte in acc to reset PC7
OUT  83H      ; reset PC7 = 0
RET
```

(4.) Write a BSR control word subroutine to set $PC_7, PC_6, \ldots, PC_1, PC_0$ and reset each after 1 second delay.



➡ Port address [as earlier – example 1]

➡ Control word : 1 000 0000 = 80H

➡ BSR control words for PC7 to PC0 as

To set PC7 = 0 0001111 = 0FH

" reset " = 0 0001110 = 0EH

" set PC_6 = 0000 1101 = 0DH

" reset " = 0000 1100 = 0CH

⋮

" set PC_0 = 0000 0001 = 01 H

" reset " = 0000 0000 = 00H

No. Date

⇒ Subroutine :

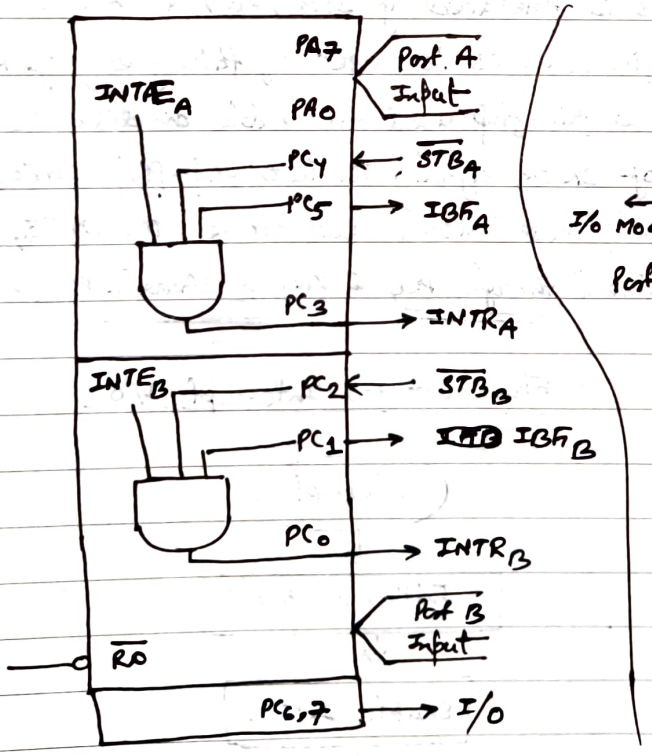
```
        MVI  A, 80H
        OUT  63H
        MVI  A, 0FH
again : OUT  63H
        CALL Delay
        DCR  A
        ANI  0FH
        JMP  again

Delay : MVI  C, 0AH
loop  : MVI  D, 64H
loop1 : MVI  E, 0EH
loop2 : DCR  E
        JNZ  loop2
        DCR  D
        JNZ  loop1
        DCR  C
        JNZ  loop
        RET
```

Programming in Mode 1 (Strobe I/O Mode)

⇒ Input & output data are latched.

⇒ Interrupt logic is supported.

⇒ Mode 1 Input Control Signals : Port A & Port B as Input



Control Word

↳ To set up ports A, B as input mode 1

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

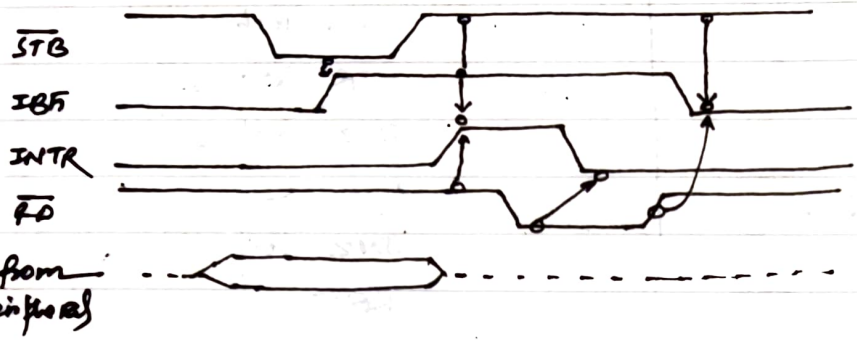
- D7 = 1 : I/O Mode
- D6 = 0 : Port A Mode 1
- D5 = 1 : Port A I/P
- D4 = 1 : Port A I/P
- D3 = 1/0 : PC6,7 : 0 = output & 1 = input
- D2 = 1 : Port B Mode 1
- D1 = 1 : Port B I/P
- D0 = X

No. Date

- **Status word** ⇒ Placed in acc if port C is read.

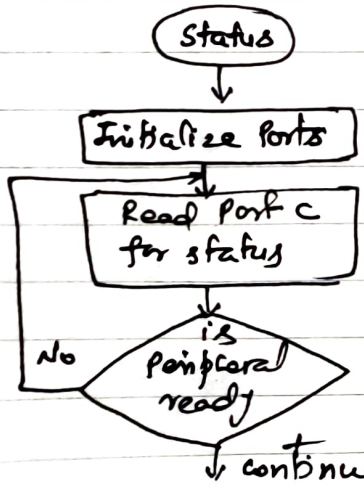
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
I/O	I/O	IBF_A	$INTE_A$	$INTR_A$	$INTE_B$	IBF_B	$INTR_B$

- **Timing Waveform**

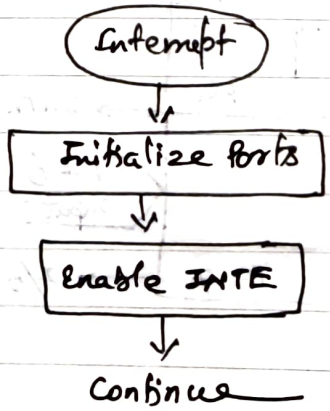


- **STB (Strobe I/p)** : generated by a peripheral to indicate that it has transmitted a byte of data. The 8255 then generates IBF & INTR as shown above.
- **IBF (Input Buffer Full)** : an ack by 8255 to indicate that input latch has received the data byte. This is reset when the MPU reads the data.
- **INTR (Interrupt Req)** : an output signal that may be used to interrupt the MPU. This is generated if $\overline{STB}$, IBF & INTE are all at logic 1. This is reset by falling edge of $\overline{RD}$.
- **INTE (Interrupt Enable)** : an internal flip-flop used to enable or disable the generation of INTR signal. The two flip-flops $INTE_A$ & $INTE_B$ are set/reset using BSR mode. The $INTE_A$ enabled or disabled through PC_4 & $INTE_B$ through PC_2.

- **Flow chart for status check I/O**

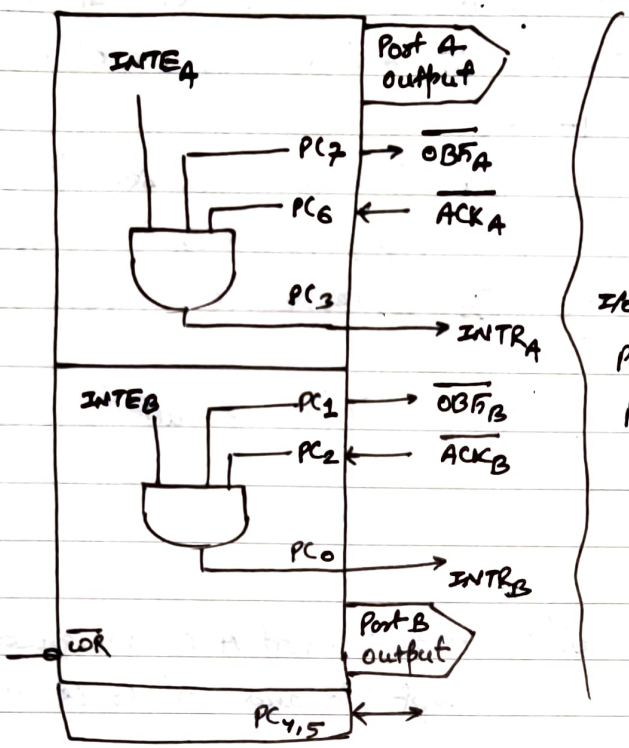


Flowchart for Interrupt I/O



No. Date

→ **Mode 1: Output Control Signals**



Control word – Mode 1 Output

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

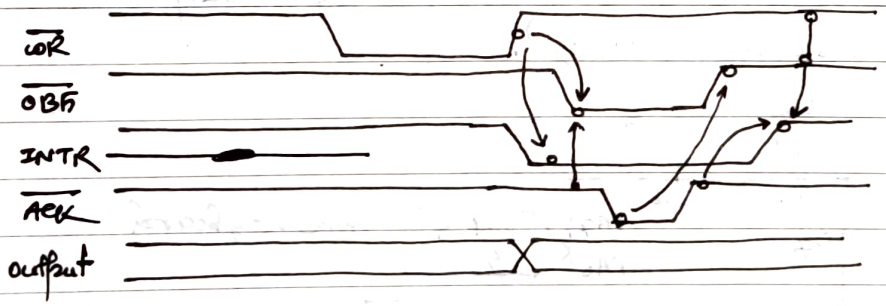
- D7 → I/O mode
- D6 → Port A mode 1
- D5 → Port A O/P
- D4 → Port B O/P
- D3 → " " mode 1
- D2 → PC$_{4,5}$: 1 = input, 0 = output
- D1 → Port B mode 1
- D0 → Port B O/P

→ To set up ports A & B as O/P ports in mode 1

→ **Status word:** ⇒ Placed in acc if port c is read.

D7	D6	D5	D4	D3	D2	D1	D0
$\overline{OBF_A}$	INTE$_A$	I/O	I/O	INTR$_A$	INTE$_B$	$\overline{OBF_B}$	INTR$_B$

→ **Timing waveform:**



→ ~~INTE: an internal flip flop~~

→ **$\overline{OBF}$ (Output Buffer Full):** an output signal that goes low when the MPU writes data into output latch of 8255. It indicates to an output peripheral that new data are ready to be read. It goes high again after the 8255 receives $\overline{ACK}$ from peripheral.

→ **$\overline{ACK}$ (Acknowledge):** an input signal from a peripheral that must output a low when peripheral receives the data from 8255 ports.

No.

Date

- **INTR** : an output signal & is set by rising edge of $\overline{ACK}$ signal. can be used to interrupt MPU to request the next data byte for output. It is set when $\overline{OBF}$, $\overline{ACK}$ & INTE are all one & reset by falling edge of $\overline{WR}$.
- **INTE** ; an internal flip-flop to a port & needs to be set to generate the INTR signal. Two flip-flops $INTE_A$ & $INTE_B$ are controlled by bits PC_6 & PC_2, respectively through BSR mode.
- **$PC_{4,5}$** : Two lines can be set up either as input or output.

Examples :

(1.) Figure shows an interfacing circuit in mode 1. Port A is designed as input port for a keyboard & with interrupt I/O and port B is " as the output port & for a printer with status check I/O.

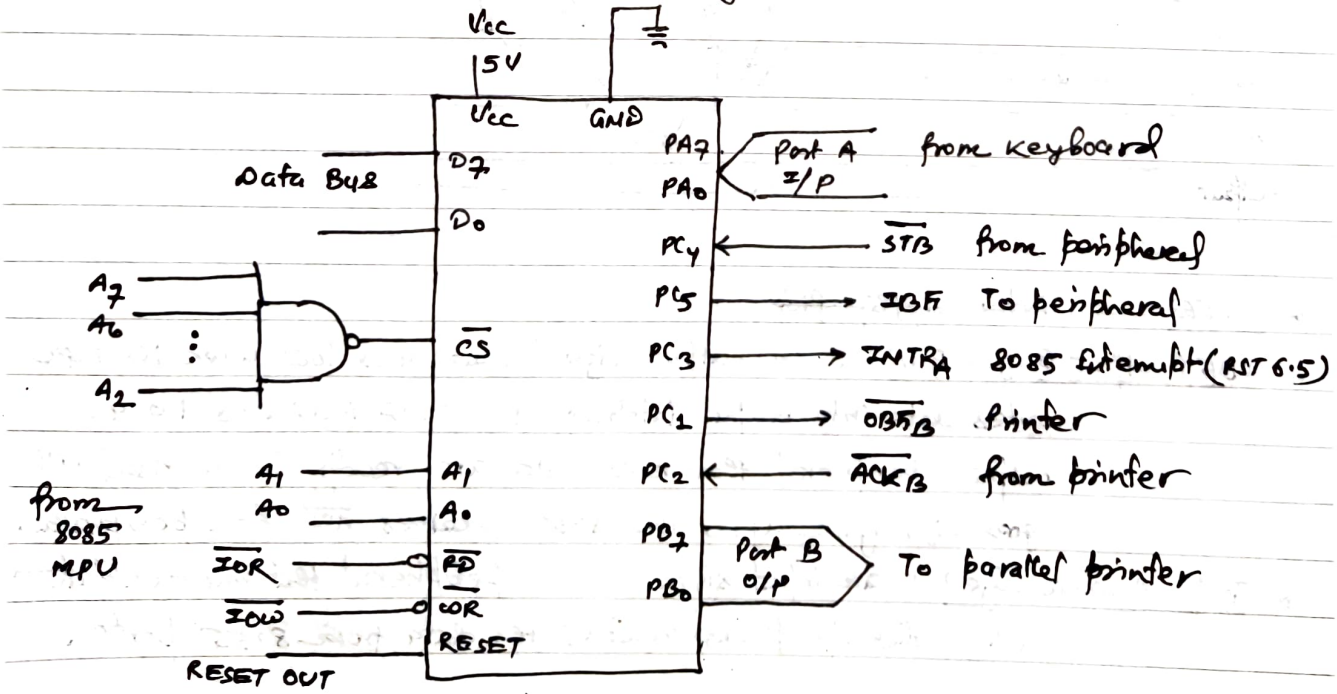
(a) Find port addresses by analyzing the decode logic.

(b) Determine control word to set up port A as I/p & port B as o/p in mode 1.

(c) Determine the BSR word to enable $INTE_A$ (port A)

(d) " the masking byte to verify $\overline{OBF}_B$ line in the status check I/O (Port B)

(e) Write initialization instructions & a printer subroutine to output characters that are stored in memory.



No. Date

Solution:

(a) To select 8255, A_7 to A_2 lines are all 1.

∴ Port A address = 1111 1100 = FCH

" B " = 1111 1101 = FDH

" C " = 1111 1110 = FEH

" D " = 1111 1111 = FFH

(b) Control word = 1011 X10X = 10110100 = B4 H

(c) BSR word to set $INTE_A$: bit PC_4 should be 1.

∴ 0 XXX 1001 = 0000 1001 = 09H

(d) Status word to check $\overline{OBF_B}$:

XXXXXX $\overline{OBF_B}$ X = 0000 0010 = 02H

(e) Initialization Program :

```
MVI A, B4H   ; word to initial ports B A & B as needed
OUT FFH      ; write con in control register.
MVI A, 09H   ; Set INTE4 (PC4) or enable INTEA
OUT FFH      ; using BSR mode
EI           ; Enable interrupt
CALL Print   ; 
; Continue other tasks      or HLT
```

Print Subroutine:

```
Print : LXI H, MEM   ; Pointer to desired location
        MVI B, Count ; No. of characters to be printed
Next :  MOV A,M      ; Get a character
        MOV C, A     ; Save the " to C
Status: IN FEH       ; Read port C for status of OBF
        ANI 02H      ; Mask all bits except D1
        JZ status
        MOV A,C      ; retrieve the character
        OUT FDH      ; Send the " to port B
        INX H
        DCR B
        JNZ next
        RET
```

No. Date

(f) Write a subroutine to accept character from keyboard & send it to printer.

→

```
        MVI  A, B4 H   ; Mode selection
        OUT  FF H
        MVI  A, 09H    ; check Enabling of INTE4
        OUT  FF H
        EI
        CALL Read
        CALL Print
        HLT

Read :  IN   FC H      ; read port A
        MOV  C, A      ; store char in C
        RET

Print : IN   FE H      ; read port C
        ANI  02H       ; check OBF
        JZ   Print
        MOV  A, C
        OUT  FD H
        RET
```

No.

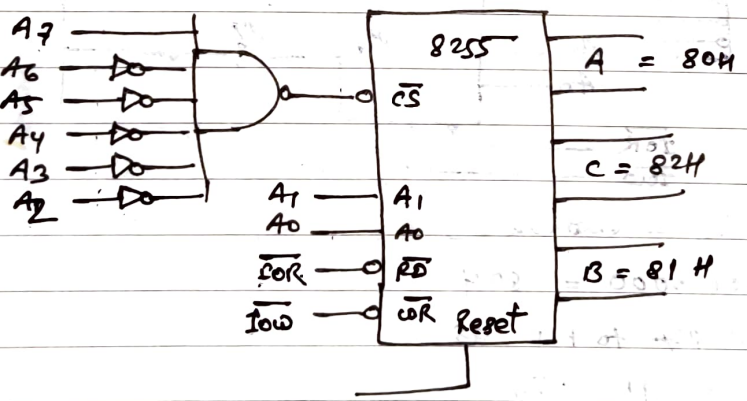
Date

(b) Control Word = 10000011 = 83H

(c) Program:

```
MVI  A, 83H   ; load acc with control word
STA  8003H    ; write word into control reg. to initialize ports
LDA  8001H    ; Read switches at port B
STA  8000H    ; Display the reading at port A
LDA  8002H    ; Read switches at port C
ANI  0FH      ; Mask upper as not input data
RLC           ; Rotate
RLC
RLC
RLC
STA  8002H    ; Display at port C
HLT.
```

(3) Write a BSR control word subroutine to set bits PC_7 & PC_3 and reset them after 10 ms.



⇒ BSR Control words:

To set PC_7 = 0 000 1111 = 0FH

" reset " = 0000 1110 = 0EH

" Set PC_3 = 0000 0111 = 07H

" reset " = 0000 0110 = 06H

⇒ Port address: Control register address = 1000 0011 = 83H