Chapter 2.2 Intel 8255A (Contd…)

Example 3

Figure (on the right) shows an interfacing circuit using the 8255A in Mode 1. Port A is designated as the input port for a keyboard with interrupt I/O and port B is designated as the output port for a printer with status check I/O.

a) Find port addresses by analyzing the decode logic.

b) Determine the control word to set up port A as input and port B as output in Mode 1.

c) Determine the BSR word to enable INTEA.

d) Determine the masking byte to verify the OBF' line in status check I/O.

e) Write subroutine to accept character from keyboard and send character to printer.

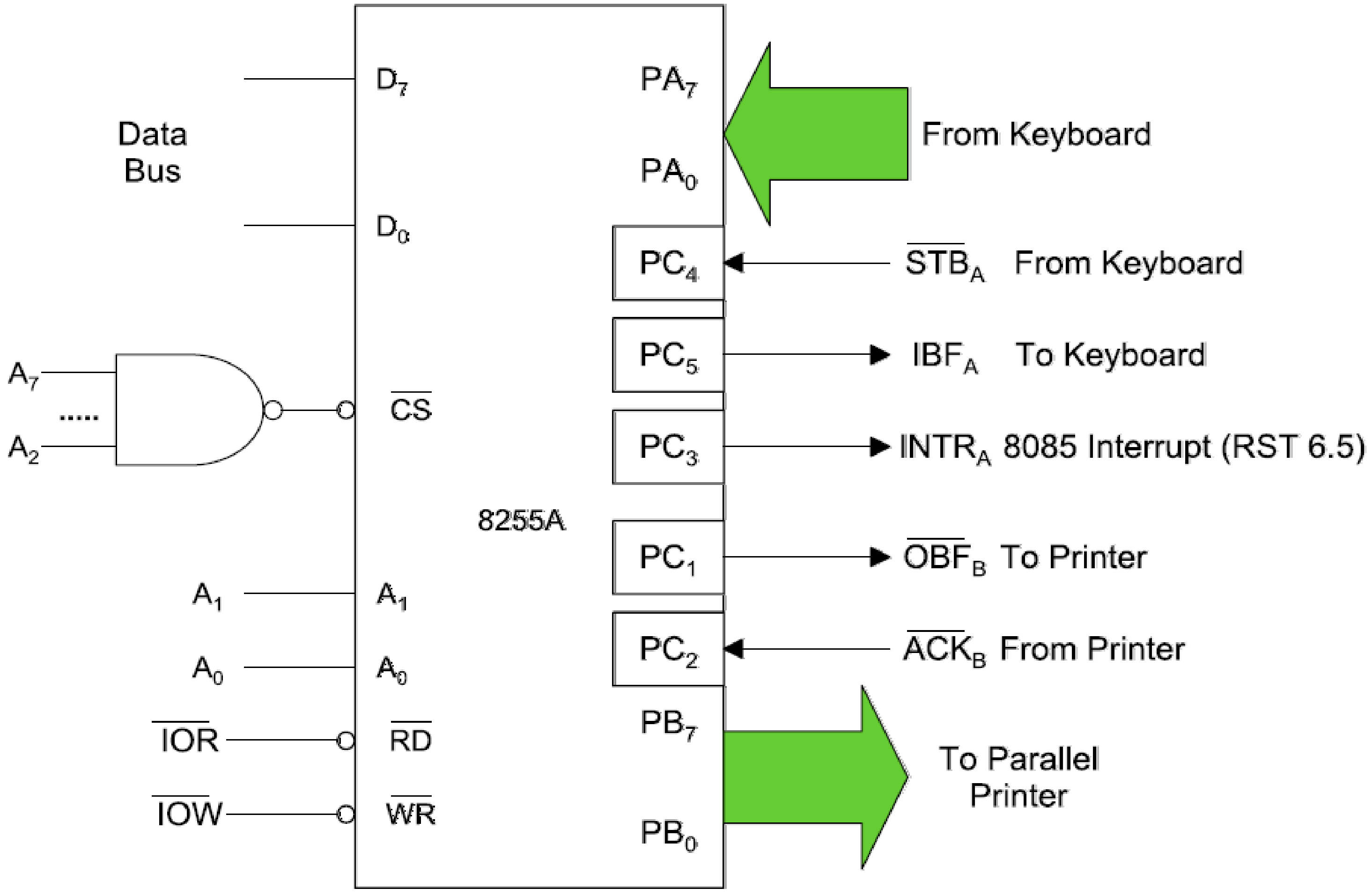


Fig: 8255A Mode 1 Example

Chapter 2.2 Intel 8255A (Contd...)

Solution

a) The 8255A is connected as I/O mapped I/O. When the address lines A7-A2 are all 1, the output of NAND gate goes low and selects 8255A. The port addresses are calculated as 1111 11XX:

Port A	= 1111 1100 (FCH)
Port B	= 1111 1101 (FDH)
Port C	= 1111 1110 (FEH)
Control Register	= 1111 1111 (FFH)

b) Control word to set up port A as input and port B as output Mode 1 is:

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

= B4H

c) BSR word to set $INTE_A$

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

= 09H

d) Status word to check OBF_B'

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
X	X	X	X	X	X	OBF_B'	X

Masking Byte = 02H

Chapter 2.2 Intel 8255A (Contd…)

e) Subroutines to accept character from keyboard and send to printer:

```
        MVI A, B4H   ; Initialize control word
        OUT FFH           ; Using I/O Mode
        MVI A, 09H   ; Set INTEA (PC4)
        OUT FFH           ; Using BSR Mode
        EI                ; Enable Interrupt
        CALL READ  ; Read Character
        CALL PRINT ; Display Character
        HLT               ; Terminate Program
READ:                     ; Keyboard Read Subroutine
        IN FEH            ; Read Port C
        ANI 20H           ; Check IBFA (PC5)
        JZ READ
        IN FCH            ; Read ASCII code of character
        MOV C, A     ; Save Character
        RET
```

```
PRINT:
        IN FEH                ; Read port C
        ANI 02H               ; Check OBFB' (D1)
        JZ PRINT
        MOV A, C     ; Get Character
        OUT FDH      ; Send Character to port B
        RET
```

Chapter 2.2 Intel 8255A (Contd…)

Programming in Mode 2 (Strobe Bidirectional Bus I/O)

- When the 8255 is operated in Mode 2
  - Port A can be used as a bidirectional 8-bit I/O bus using PC_3–PC_7 for handshaking
  - Port B can be programmed only in Mode 0 (PC_0–PC_2 as Input or Output), or in Mode 1 (with PC_0–PC_2 for handshaking).
- Interrupts are generated for both output and input operations on the same $INTR_A$ (PC_3) line.
- The control signal definitions for Mode 2 are:
- **Output Control Signals**
  - **$\overline{\text{OBF}}$ (Output Buffer Full)** : This is an active low output which indicates that the CPU has written data into Port A.
  - **$\overline{\text{ACK}}$ (Acknowledge)** : This is an active low input signal (generated by the peripheral) which enables the tri-state output buffer or Port A and makes Port A data available to the peripheral. In Mode 2, Port A outputs are in tri-state until enabled.
  - **INTE 1** : This is the flip-flop associated with Output Buffer Full. INTE 1 can be used to enable to disable the interrupt by setting or resetting PC_6 in the BSR Mode.
- **Input Control Signals**
  - **$\overline{\text{STB}}$(Strobe Input)** : This is an active low input signal which enables Port A to latch the data available as its input.
  - **IBF (Input Buffer Full Flip-Flop)** : This is an active high output which indicates that data has been loaded into the input latch of Port A.
  - **INTE 2** : This is an Interrupt enable flip-flop associated with Input Buffer Full. It can be controlled by setting or resetting PC_4 in the BSR Mode.

Chapter 2.2 Intel 8255A (Contd…)

Programming in Mode 2 (Strobe Bidirectional Bus I/O) (Contd…)

- **Status Word in Mode 2**
- The status word for Mode 2 (accessed by reading Port C) is shown in above figure. D_7–D_3 of the status word carry information about $\overline{OBF}$, $INTE_1$, IBF_A, $INTE_2$, and $INTR_A$. The status of the bits D_2 – D_0 depend on the mode setting of Group B. If B is programmed in Mode 0, D_2–D_0 carry information about the control signals for B, depending upon whether B is an Input port or Output port respectively.

Chapter 2.2 Intel 8255A (Contd…)

Programming in Mode 2 (Strobe Bidirectional Bus I/O) (Contd…)

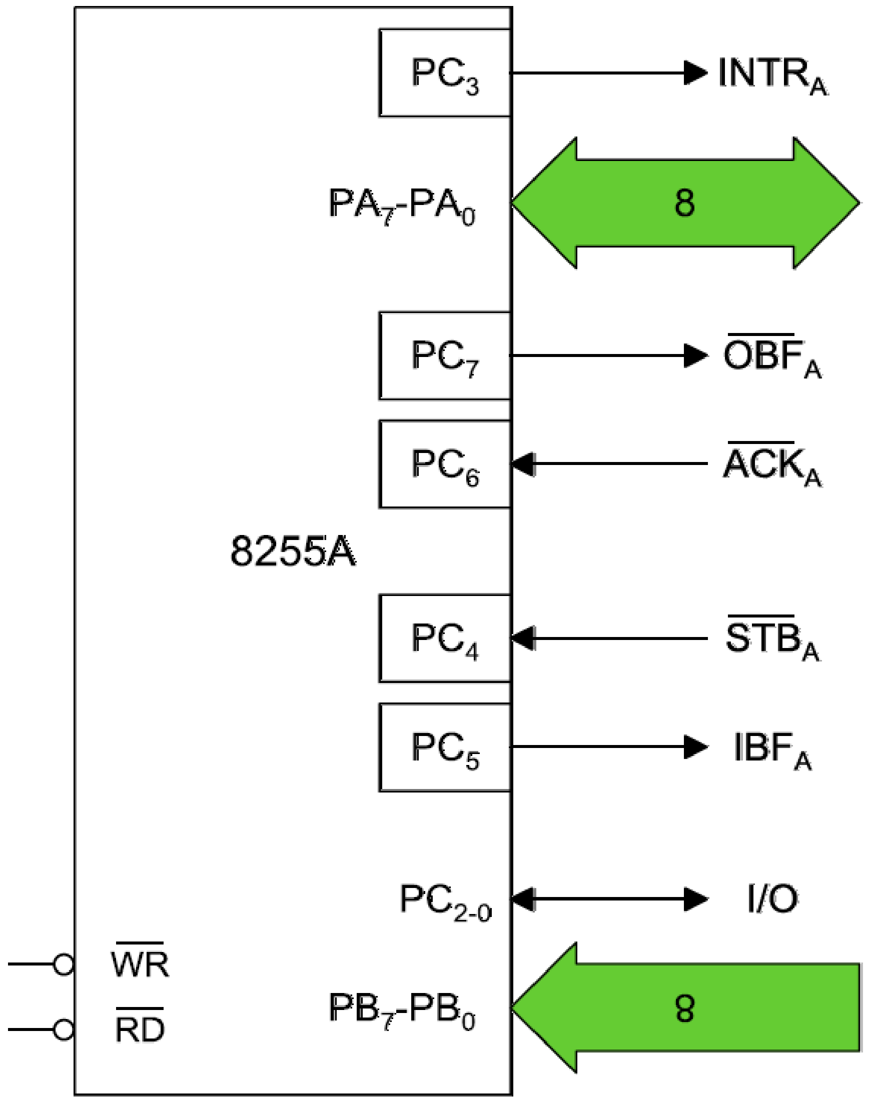


Fig: 8255A Mode 2 and Mode 0 Input Configuration

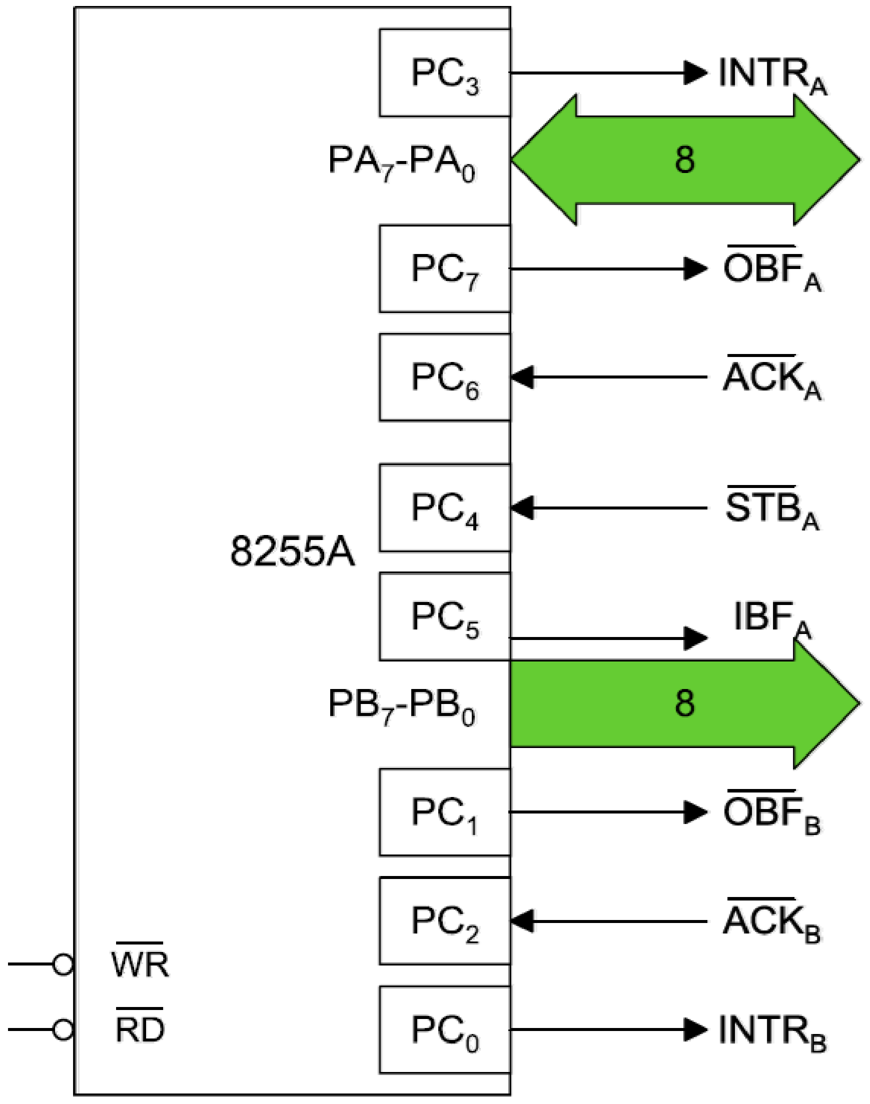


Fig: 8255A Mode 2 and Mode 1 Output Configuration

Chapter 2.2 Intel 8255A (Contd…)

Programming in Mode 2 (Strobe Bidirectional Bus I/O) (Contd…)

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

$PC_{2\text{-}0}$
1 = Input
0 = Output

Fig: Control Word - Mode 2 and Mode 0 Input Configuration

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

Fig: Control Word - Mode 2 and Mode 1 Output Configuration

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
OBF_A	$INTE_A$	IBF_A	$INTE_2$	$INTR_A$	X	X	X

$\underbrace{D_7 \ldots D_3}_{\text{Group A}}$ $\underbrace{D_2 \ldots D_0}_{\text{Defined by Mode 0 and Mode 1 Selection}}$

Fig: Status Word - Mode 2 Configuration

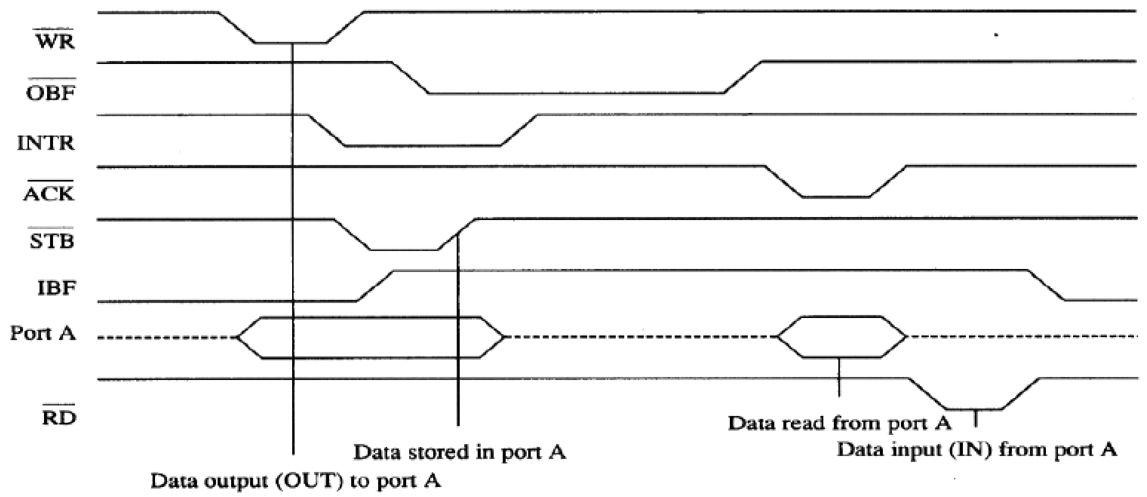


Fig: Timing Waveform for Mode 2 Configuration

EXAMPLE:

2. Design an interfacing circuit to set-up bidirectional data communication in the master-slave format between two 8085A microcomputers. Use the 8255A as the interfacing device between the master and the slave microcomputers. Write necessary program to transfer a block of data from the master to the slave.

Solution:

The block diagram shows two bidirectional data buses interconnected through the 8255A, which serves as a peripheral device of the master MPU. Port A of 8255A is used for bidirectional data transfer and five signals from Port C are used for handshaking. When the master MPU writes a data byte in the 8255A, the OBF’ signal goes low to inform the slave that a byte of data is available, and the slave acknowledges when it reads the byte.

Similarly, two other handshake signals are used when the slave transfers a data byte to the master. The master requires I/O ports to read and write data and check the status of the handshake signals. Similarly, the slave MPU required I/O ports to perform read and write operations. For this example, let’s suppose that both MPUs are set up under the status check mode.

According to assumed decode logic, the addresses of individual ports are selected as:

1. Port A = FCH ($A_1 = 0, A_0 = 0$)
2. Port B = FDH ($A_1 = 0, A_0 = 1$)
3. Port C = FEH ($A_1 = 1, A_0 = 0$)
4. Control Register = FFH ($A_1 = 1, A_0 = 1$)

Control word can be expressed as:

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

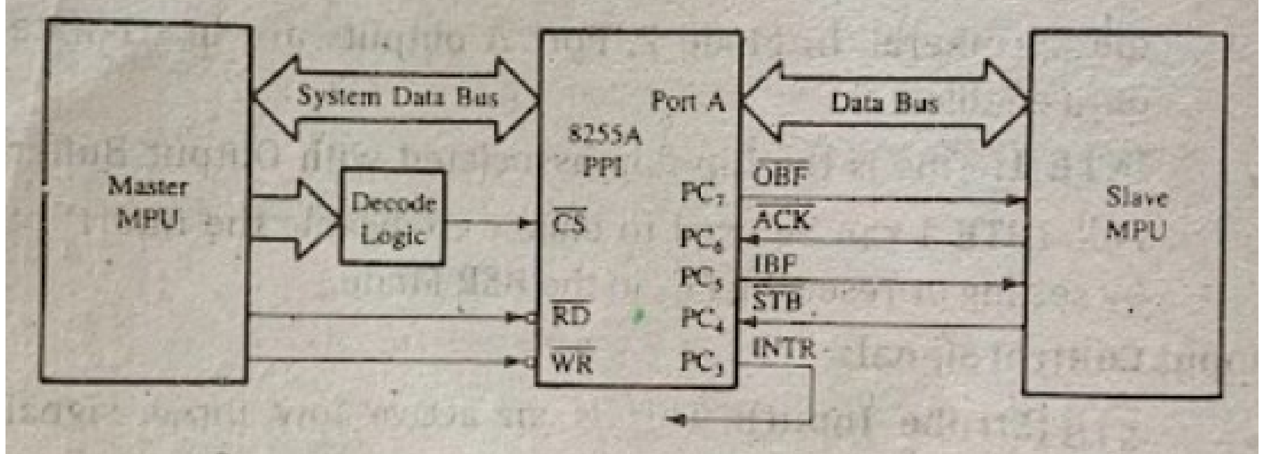


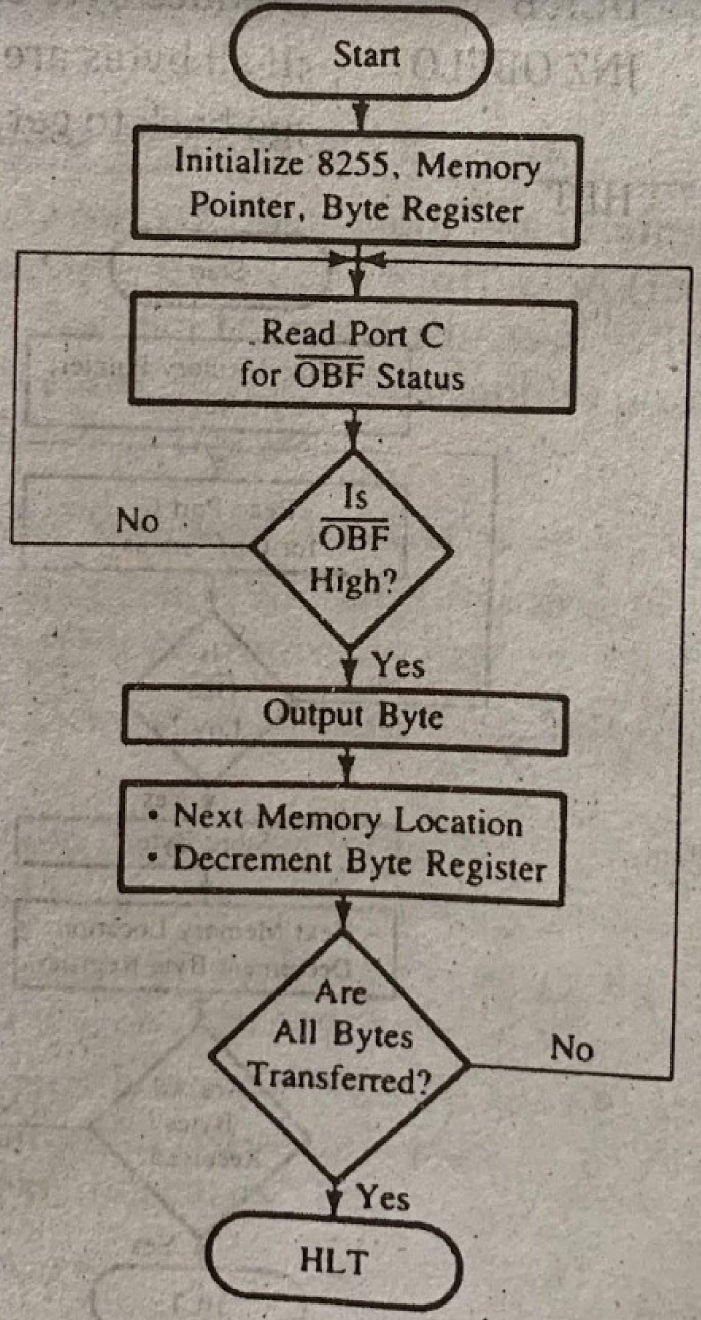
Fig.: Block diagram of bidirectional communication between two computers using the 8255A

Data transfer from master MPU to slave MPU:

1. The master MPU reads the status of OBF’ to verify whether the previous byte has been read by the slave MPU.
2. The master writes data into Port A and the 8255A informs the slave by causing the signal OBF’ to go low.
3. The slave checks OBF’ signal from the master for data availability.
4. The slave MPU reads data from Port A and acknowledges the reading at the same time by making the signal ACK’ low.

Data transfer from slave MPU to master MPU:

1. The slave MPU checks the handshake signal IBF to find out whether Port A is available (i.e. empty) to transfer a data byte.
2. The slave MPU place a data byte on the data bus and informs 8255A by enabling the STB’ signal.
3. The 8255A causes the IBF to go high, and the master MPU reads the signal to find out whether a data byte is available.
4. Master MPU reads the data byte.

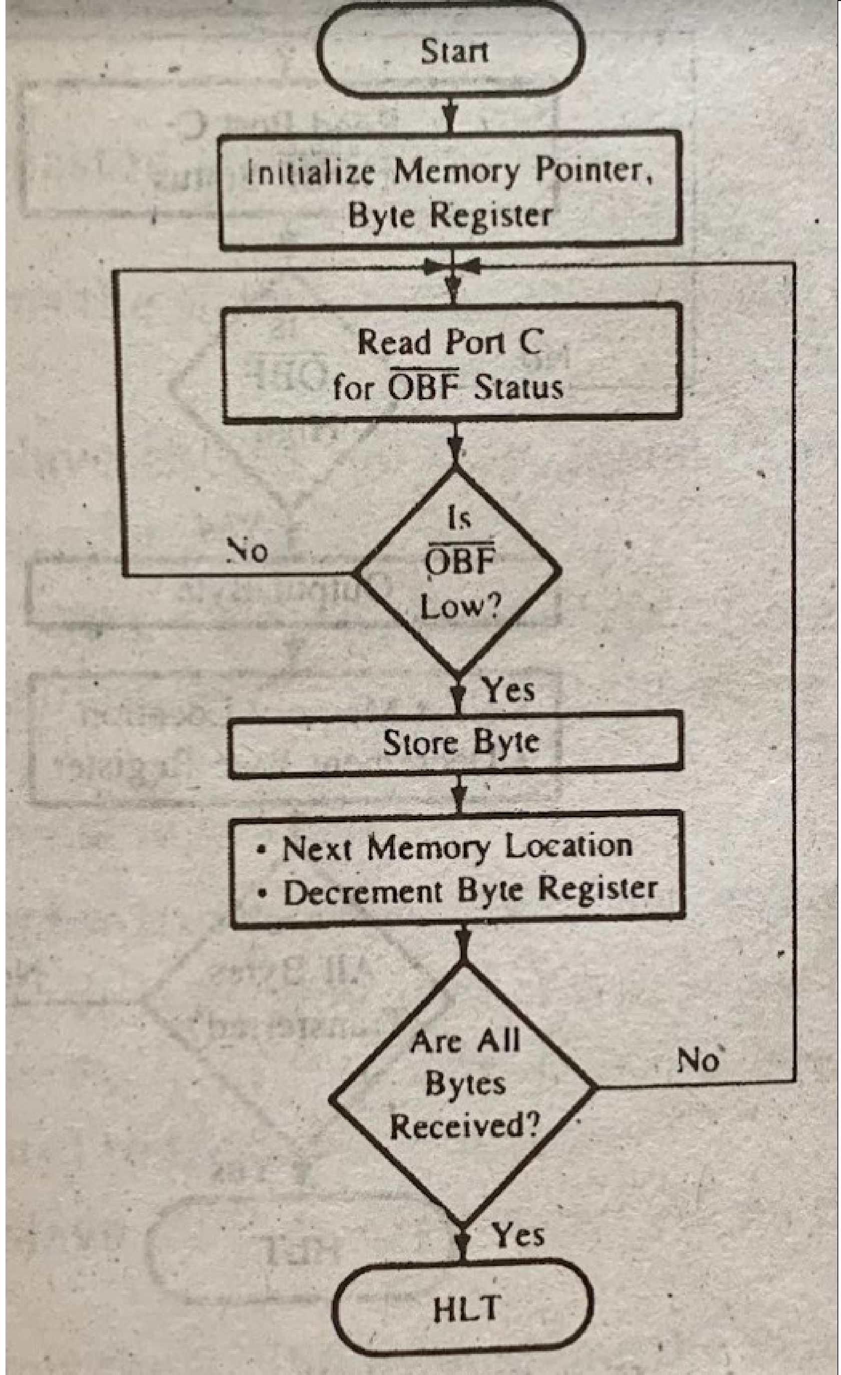


Start
Initialize 8255, Memory Pointer, Byte Register
Read Port C for $\overline{OBF}$ Status
Is $\overline{OBF}$ High?
No
Yes
Output Byte
• Next Memory Location
• Decrement Byte Register
Are All Bytes Transferred?
No
Yes
HLT

Fig.: Flowchart of Master program

Master Program

```
LXI SP, STACK1
LXI H, MASTER              ; Memory pointer for data
MVI B, BYTES   ; No. of bytes to be sent
MVI A, CTRL                ; Control word
OUT FFH                    ; Initialize 8255A
OBFLO:IN FEH; Read Port C for status
               RAL      ;Place OBF’ status in CY
               JNC OBFLO ;If OBF’ is low, wait
               MOV A, M ;Get byte
               OUT FCH ;Sent byte to Port A
               INX H ;Next memory location
               DCR B ;Reduce byte count by 1
               JNZ OBFLO ; If all bytes are not
        ;transferred, to back to get next byte
               HLT
```



Start
Initialize Memory Pointer, Byte Register
Read Port C for $\overline{OBF}$ Status
Is $\overline{OBF}$ Low?
No
Yes
Store Byte
• Next Memory Location
• Decrement Byte Register
Are All Bytes Received?
No
Yes
HLT

Fig.: Flowchart of Slave program

Slave Program

```
LXI SP, STACK1
LXI H, SLAVE ; Memory pointer for destination data
MVI B, BYTES   ; No. of bytes to be received

OBFHI: IN FEH; Read Port C for OBF’
       RAL ;Place OBF’ (in D7) status in CY
       JC OBFHI ;If OBF’ is high, wait
       IN FCH ;Read byte
       MOV M, A ;Store data
       INX H ;Next memory location
       DCR B ;Reduce byte count by 1
       JNZ OBFLI ; If all bytes are not received, to back to
                           ;get next byte
       HLT
```

Chapter 2.3 Parallel Interfacing with ISA and PCI bus

I/O buses are used to connect the system bus (address, data, and control buses) for example ISA (8 or 16 bit), EISA (Extended ISA - 32 bit), VESA (Video Electronics Standards Association) local bus (VL Bus), PCI (32 or 64 bit), Accelerated graphics port (AGP), PCI-X (64 bit, 133MHZ), PCI-Express etc.

- **ISA Bus (Industry Standard Architecture)**
- First introduced in 1982 with the first PC (IBM/PC) – [Intel 8088 8 bit microprocessor].
- Originally ISA bus was with 8-bit bus which runs at 4.77 MHz.
- 16 bit version of ISA was introduced in 1984 used with Intl 80286 (16-bit microprocessor).
- Peripheral devices such as sound cards, disk drives, network cards etc. are connected via ISA slots.
- ISA bus is mostly obsolete for PC nowadays, but is still used in many industrial applications due to their low costs and existing cards.

Chapter 2.3 Parallel Interfacing with ISA and PCI bus

- **8-bit ISA bus Architecture**
  - Has data bus width of 8 bits and address bus width of 20 bits.
  - Number of pins in ISA slots/cards are 62.
  - Clock frequency of 4.77 MHz.
  - ISA bus connector contains:
    - 20 bit address bus (A_{19}-A_0)
    - 8 bit data bus
    - MEMR’, MEMW’. IOR’, IOW’ control signal for controlling I/O or memory on the ISA card.
    - Interrupt request lines IRQ_2-IRQ_7
    - DMA request inputs DRQ_1-DRQ_3
    - DMA acknowledgement O/Ps $DACK_0$’-$DACK_3$’
    - Clock signals
    - Power lines and Reset

Fig: 8 Bit ISA Slot

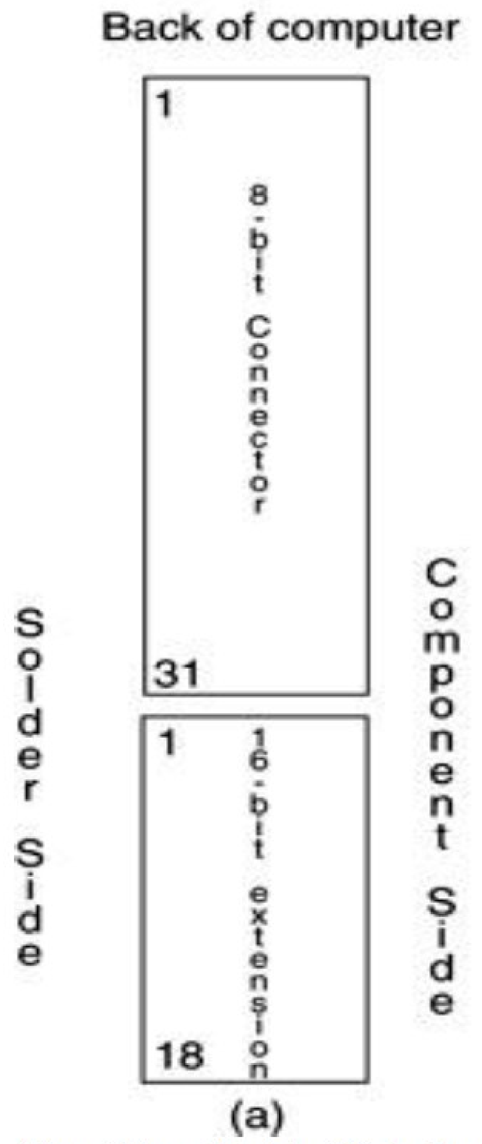
Fig: 16 Bit ISA Slot

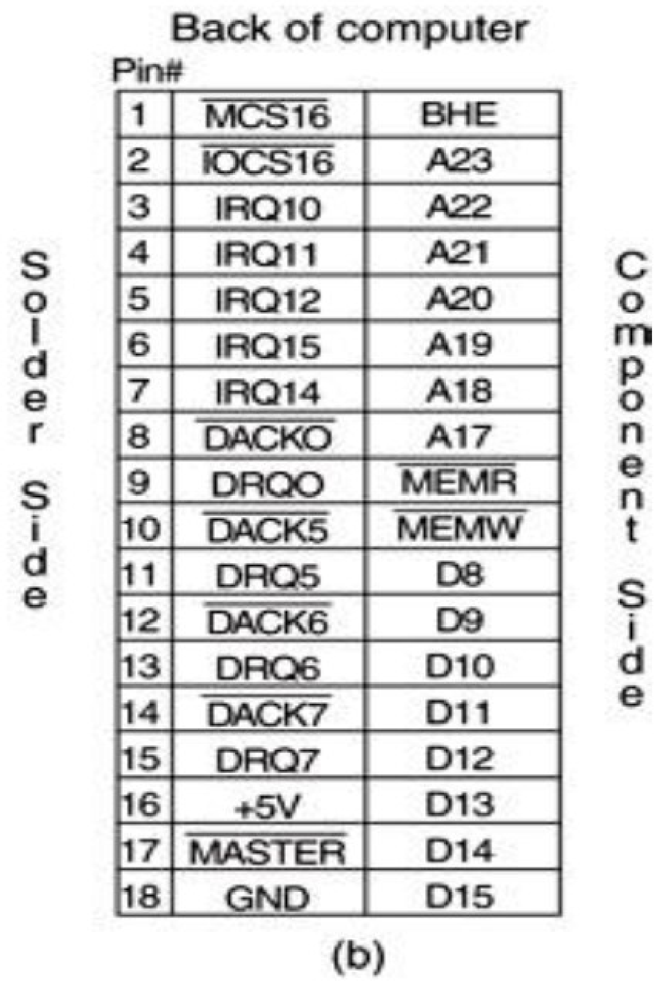
Fig: EISA Slot

Chapter 2.3 Parallel Interfacing with ISA and PCI bus

- **16-bit ISA bus Architecture**
  - Data bus width of 16 bit and address bus width of 24 bits.
  - Number of pins in ISA card/slot are 98
  - Clock frequency of 8.33 MHz
  - Consists of an extra connector with 36 pins behind the 8-bit connector.
  - Compatible with both 8-bit and 16-bit ISA cards.
  - 16-bit card consists of two edge connectors
    - One plugs into the original 8-bit connector
    - Other plugs into the new 16-bit connector
  - Extra connector consists of
    - 4 additional address lines – 24 lines in total
    - 8 additional data lines – 16 lines in total
    - 4-bit DMA channel request and acknowledgement lines
    - Additional Interrupt lines
    - Control lines to select 8 or 16 bit transfer

Chapter 2.3 Parallel Interfacing with ISA and PCI bus





Back of computer
Solder Side
Component Side
(b)

Pin#		
1	$\overline{\text{MCS16}}$	BHE
2	$\overline{\text{IOCS16}}$	A23
3	IRQ10	A22
4	IRQ11	A21
5	IRQ12	A20
6	IRQ15	A19
7	IRQ14	A18
8	$\overline{\text{DACK0}}$	A17
9	DRQ0	$\overline{\text{MEMR}}$
10	$\overline{\text{DACK5}}$	$\overline{\text{MEMW}}$
11	DRQ5	D8
12	$\overline{\text{DACK6}}$	D9
13	DRQ6	D10
14	$\overline{\text{DACK7}}$	D11
15	DRQ7	D12
16	+5V	D13
17	$\overline{\text{MASTER}}$	D14
18	GND	D15

Fig: The 16-bit ISA bus. (a) Both 8- and 16-bit connectors and (b) the pinout of the 16-bit connector.

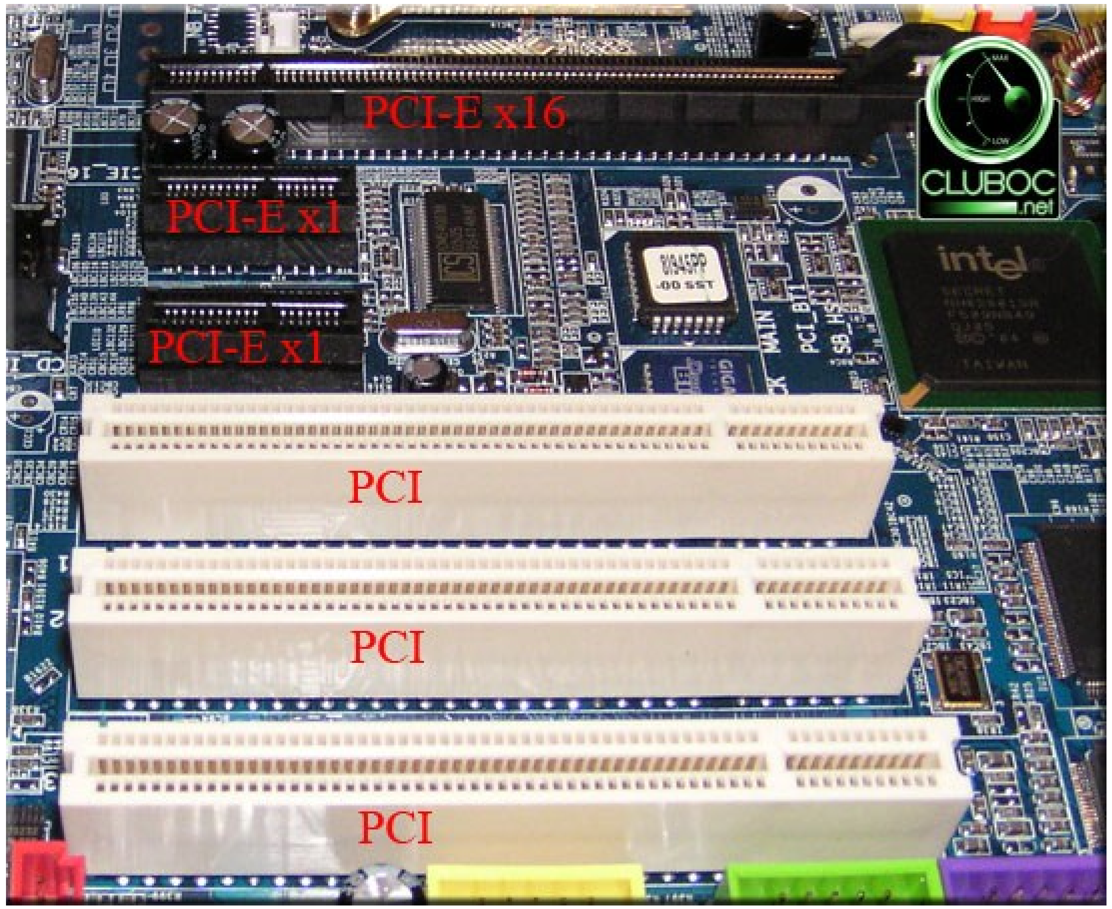
Chapter 2.3 Parallel Interfacing with ISA and PCI bus

- **Reasons for elimination of ISA Bus**
    - ISA bus is slow, hard to use and bulky.
    - Once each ISA slot/card uses dedicated interrupt lines, only limited number of cards can be used.
    - Since address lines of 24 bits, a maximum of 2^{24}=16 MB of RAM can only be accessed for DMA.
    - Since data bus size is 16 bits only, higher bits data (32-bits) communication would reduce system performance.
    - ISA cards do not have plug and play (PnP) technology i.e. they can't be configures automatically by BIOS or operating system.
    - ISA cards must be controlled manually by setting the I/O addresses, interrupts and clock speed using jumpers and switches on the card itself.
- **Improvements in ISA bus**
    - EISA (Extended ISA) of 32-bits, 8 MHz; now obsolete
    - ISA PnP for plug and play; now obsolete
    - VL-Bus of 32-bits operated at the speed of local bus (CPU)
    - Used only for graphics cards
    - Possibility of interference with the performance of the CPU

Chapter 2.3 Parallel Interfacing with ISA and PCI bus

- **PCI Bus (Peripheral Component Interconnect)**
  - Introduced in 1990 by Intel
  - Provides direct access to the CPU and system memory but uses a bridge to connect to the system bus to eliminate the potential for interference with CPU.
  - PCI bus is independent of processor type or speed
  - Originally operated at 33 MHz using 32-bit data lines
  - Revised standard at 66 MHz using 64-bit data lines
  - The 32-bit PCI connector has 124 pins and 64-bit PCI connector has 188 pins
  - The PCI bus is able to work with so few pins because of hardware multiplexing i.e. the device sends more than one signal over a single pin
  - Also, PCI supports devices that use 5v signalling voltage levels
  - PCI card support plug and play (PnP) feature i.e. PCI devices are automatically recognized and configured to work in system.
- **Advancement in PCI bus**
  - PCI-X (PCI extended): runs at 133 MHz, 32-bit and 1.06 GBps data rate
  - PCI-E (PCI express): replaced PCI, PCI-X & AGP standards
  - The microprocessor connects to the PCI bus through an IC called a PCI bridge
  - Virtually any processor can interface to PCI with a bridge
  - The resident local bus is often called front side bus

Chapter 2.3 Parallel Interfacing with ISA and PCI bus

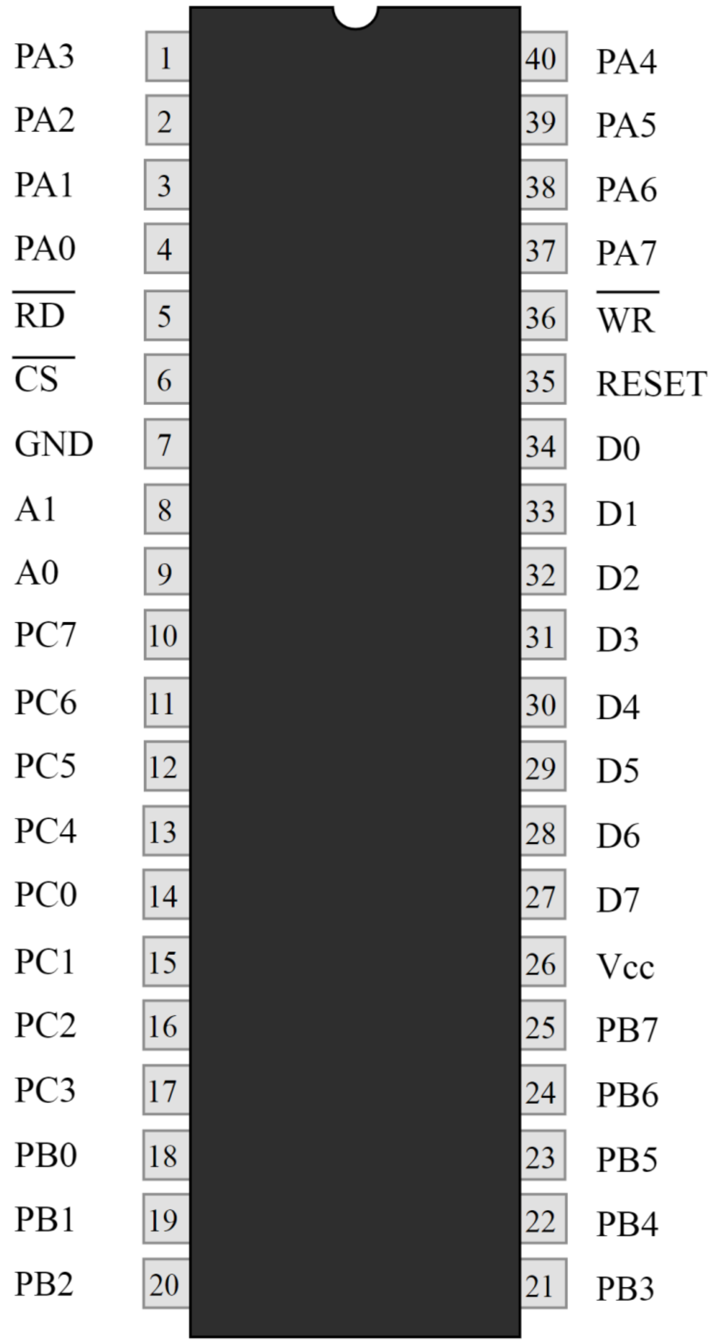


Back of computer

Pin #	Solder Side	Component Side
1	−12V	$\overline{\text{TRST}}$
2	TCK	+12V
3	GND	TMS
4	TD0	TD1
5	+5V	+5V
6	+5V	$\overline{\text{INTA}}$
7	$\overline{\text{INTB}}$	$\overline{\text{INTC}}$
8	$\overline{\text{INTD}}$	+5V
9	$\overline{\text{PRSNT 1}}$	
10		+VI/O
11	$\overline{\text{PRSNT 2}}$	
12	KEY	KEY
13	KEY	KEY
14		
15	GND	$\overline{\text{RST}}$
16	CLK	VI/O
17	GND	$\overline{\text{VNT}}$
18	$\overline{\text{REQ}}$	GND
19	+V I0	
20	AD31	AD30
21	AD29	+3.3V
22	GND	AD28
23	AD27	AD26
24	AD25	GND
25	+3.3V	AD24
26	C/$\overline{\text{BE3}}$	IDSEL
27	AD23	+3.3V
28	GND	AD22
29	AD21	AD20
30	AD19	GND
31	+3 3V	AD18
32	AD17	AD16
33	C/$\overline{\text{BE2}}$	+3 3V
34	GND	$\overline{\text{FRAME}}$
35	$\overline{\text{IRDY}}$	GND
36	+3 3V	$\overline{\text{TRDY}}$
37	$\overline{\text{DEVSEL}}$	GND
38	GND	$\overline{\text{STOP}}$
39	$\overline{\text{LOCK}}$	+3 3V
40	$\overline{\text{PERR}}$	SDONE

Notes: (1) pins 63–94 exist only on the 64-bit PCI card
(2) + VI/O is 3.3V on a 3.3V board and +5V on a 5V board
(3) blank pins are reserved

Pin #	Solder Side	Component Side
41	+3 3V	$\overline{\text{SBO}}$
42	$\overline{\text{SERR}}$	GND
43	+3 3V	PAR
44	C/$\overline{\text{BE1}}$	AD15
45	AD14	+3.3V
46	GND	AD13
47	AD12	AD11
48	AD10	GND
49	GND	AD9
50	KEY	KEY
51	KEY	KEY
52	AD8	C/$\overline{\text{BE0}}$
53	AD7	+3.3V
54	+3 3V	AD6
55	AD5	AD4
56	AD3	GND
57	GND	AD2
58	AD1	AD0
59	+V IO	+V IO
60	$\overline{\text{ACK64F}}$	$\overline{\text{REQ64}}$
61	+5V	+5V
62	+5V	+5V
63		GND
64	GND	C/$\overline{\text{BE7}}$
65	C/$\overline{\text{BE6}}$	C/$\overline{\text{BE5}}$
66	C/$\overline{\text{BE4}}$	+V IO
67	GND	PAR64
68	AD63	AD62
69	AD61	GND
70	+V IO	AD60
71	AD59	AD58
72	AD57	GND
73	GND	AD56
74	AD55	AD54
75	AD53	+V IO
76	GND	AD52
77	AD51	AD50
78	AD49	GND
79	+V IO	AD48
80	AD47	AD46
81	AD45	GND
82	GND	AD44
83	AD43	AD42
84	AD41	+VI/O
85	GND	AD40
86	AD39	AD38
87	AD37	GND
88	+VI/O	AD36
89	AD35	AD34
90	AD33	GND
91	GND	AD32
92		
93		GND
94	GND	



Pins and their usage in Intel 8255

Figure source: https://en.wikipedia.org/wiki/Intel_8255

TUTORIAL EXAMPLES:

- **Tutorials:**
- 1. Assume that your group has decided to make a PC based control system for a wine company. After studying the system, your group found out that the following to be implemented for controlling purpose:
  - Pressure measurement (6 points)
  - Temperature measurement (5 points)
  - Weight measurement (1 point)
  - Volume measurement for filling (5 points)
- Your group also decided to use 8255A PPI card at base address 0550H.

  a) List out collected documents and components

  b) List out different signals you need to derive and/or can be directly connected to your interfacing circuit.

  c) Draw minimum mapping circuit for above system

  d) What are the address captured by card

  e) Generate necessary control word

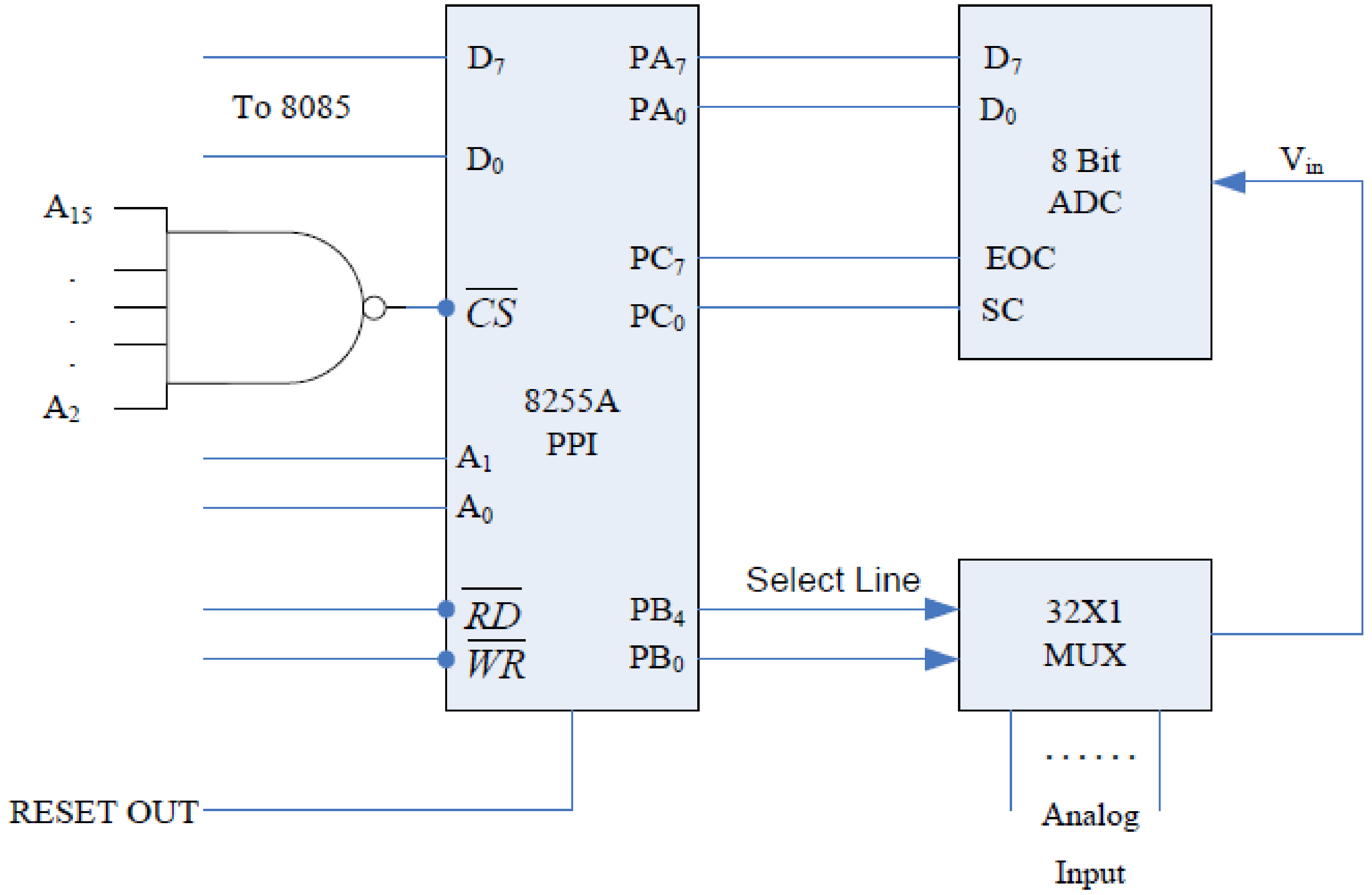
  f) Write a program module for measuring the pressure of all the points and control if the pressure is not in a range, Assume suitable data if necessary.

Solution:

a) Components: 8255A card, ADC, MUX, Memory, Processor, connecting wires, power supplies (+5V, GND), gates etc.
Documents: Data sheets and technical documentation of above components

b) Signals needed to be derived on directly connected to circuit
- A1, A2, Chip Select ($\overline{CS}$) for Port selection of of 8255A, RESET signal
- Read ($\overline{RD}$) and Write ($\overline{WR}$) signals
- Start Conversion (SC) and End of Conversion (EOC)

c) The minimum mapping circuit is as given below:



d) The base address of card is 0550H, following are address captured by card.

Port	**Address**	**A15**	**A14**	**A13**	**A12**	**A11**	**A10**	**A9**	**A8**	**A7**	**A6**	**A5**	**A4**	**A3**	**A2**	**A1**	**A0**
A	0550H	0	0	0	0	0	1	0	1	0	1	0	1	0	0	**0**	**0**
B	0551H	0	0	0	0	0	1	0	1	0	1	0	1	0	0	**0**	**1**
C	0552H	0	0	0	0	0	1	0	1	0	1	0	1	0	0	**1**	**0**
CR	0553H	0	0	0	0	0	1	0	1	0	1	0	1	0	0	**1**	**1**

The total numbers of monitoring points are 17. If we use 1 ADC for all of them, we need to select any one at given time. So, we can use 32X1 MUX which would then have 2^5=32 i.e. 5 selection lines (B_0 to B_4). These lines can have defined for any of the 17 lines.
In the above circuit,
Port A → Input port to read data from ADC in mode 0
Port B → Output port to select any one of 17 lines from MUX in mode 0
Port C → Output port (PC_0 as SC) and Input port (PC_7 as EOC)

e) Control word and BSR words:
Control word to set up port ports in above configuration:

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

= 98H

BSR word to set PC_0

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

= 01H

f) Program Module:

```
           LXI H, MEMORY
           MVI A, 98H
           STA 0553H;    write control word in CR
           MVI C, 06H;   set counter to read 6 pressure points
           MVI B, 00H;   selection of points for MUX
NEXT:  MOV A, B
           STA 0551H;    select first pressure point
           MVI A, 01H;   load A with BSR word to set PC0
           STA 0553H;    set SC line
           CALL DELAY
           MVI A, 00H;   load A with BSR word to reset PC0
           STA 0553H;    reset SC line
READ:  LDA 0552H;    read port C
           RAL
           JNC READ;     check for PC7
           LDA 0550H;    read data from port A
           MOV M, A;     store value in memory
           CPI MAX_VALUE;   compare with maximum value
           JNC CONTROL;     control value
           CPI MIN_VALUE;   compare with minimum value
           JC CONTROL;      control value
           INR B
           INX H
           DCR C
           JNZ NEXT
```