Instrumentation II (III/I)

Course Code: EX–602
(Module#2)

Dhawa Sang Dong
(Lecturer)

Kathmandu Engineering College, Kalimati

May 26, 2022

Chapter#2

Parallel Interfacing with microprocessor based system

Class Outline

1. Methods of Parallel Data Transfer
2. General Purpose Programmable I/O device 8255
3. Parallel Interfacing with ISA and PCI Bus

Methods of Parallel Data Transfer

- Generally, when there is need of high speed data transfer then parallel communication comes into exist rising cost with multiple wires.
- It is because the serial communication could not support high speed data transfer that parallel communication exists.

Some common methods of prallel data transfer:

1. Simple I/O
2. Simple Strobe I/O
3. Single Handshake I/O
4. Double Handshake I/O

- Parallel data transmission can not be suitable for long distance communication because of potential losses along the wire.
- Printers, disk drives etc can support high speed communication of parallel data transfer.

Methods of Parallel Data Transfer

- Information exchange between microprocessor and peripheral device consists of data transfer and control signaling through I/O interfacing logic or circuits.
- Control signaling enables microprocessor monitor I/O devices and get communicated when ready to exchange the data.
- If the communicating devices operate at different speed, microprocessor can select a particular speed of operation, and the technique is called synchronization.
- synchronization technique includes previously mentioned methods of parallel data transfer.
- those are Simple, Simple Strobe, Single Handshake or Double Handshake I/O

#1 Simple Input Output (Simple I/O)

- For data from input device, device is connected input port of microprocessor.
- data is always present and ready, and it is supposed to be valid so that microprocessor can read the port at anytime.
- similar to the case output to external device from microprocessor.
- when data is available at output port from microprocessor, the device read port anytime.

#1 Simple Input Output (Simple I/O)

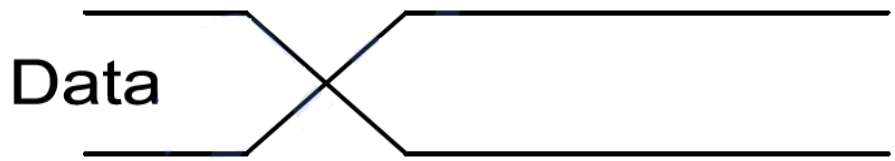


Fig. 1 Simple I/O Timing Diagram

- In the timing diagram, crossed point indicates new valid data on the output lines of port.
- it is simply because data exchange is simple and independent of other control signals that there is no other control signals.

#2 Simple Strobe Input Output (Simple Strobe I/O)

- It is because there is no possibility of data present to be valid all the time that data must be read only at the time when valid data is present in many applications.
- For the indication of valid data, strobe signal is used.
- In this type of communication, receivers has to wait for the strobe signal.

#2 Simple Strobe Input Output (Simple Strobe I/O)

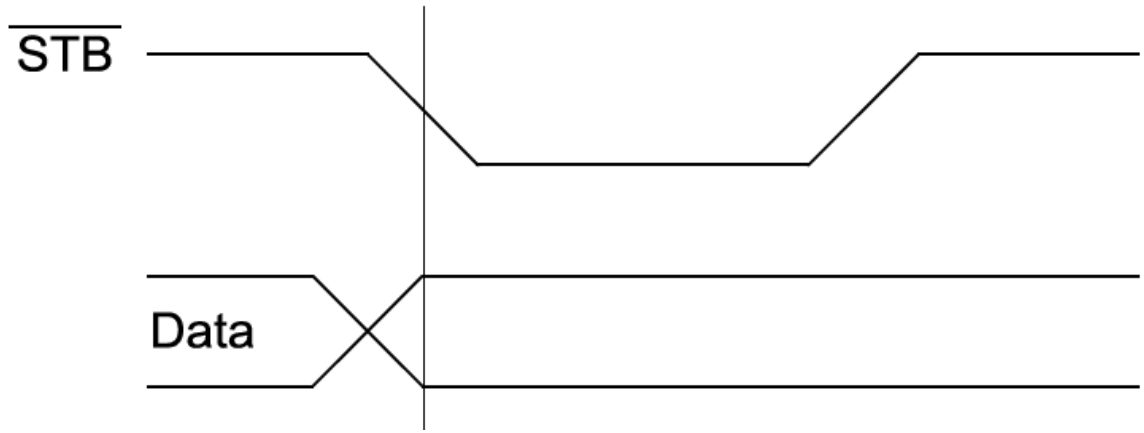


Fig. 2 Simple Strobe I/O Timing Diagram

- sending device puts parallel data at data-bus and $\overline{STB}$ signaling to indicate valid data is present.
- in this type of data transfer, microprocessor needs to wait until the device is ready for the operation.

#3 Single Handshaking Input Output (Single Handshaking I/O)

- Often, high speed microprocessor and slow speed peripherals communicates using either single or double handshaking.
- There are two scenarios for the microprocessor and peripheral communication:

1. Input Handshaking (Peripheral to microprocessor)
2. Output Handshaking (Peripheral from microprocessor)

Single Handshaking Input Output

#1) Input Handshaking

- when peripherals have data to microprocessor then it sends strobe signal to microprocessor indicating data for microprocessor.
- Once microprocessor detects the strobe ($\overline{STB}$) signal asserted it reads the data and acknowledge peripheral that data has been read and ready for the next byte of data.

Single Handshaking Input Output

#2) Output Handshaking

- when microprocessor has data for peripherals it asserts a strobe signal ($\overline{STB}$).
- Once the peripherals read the data, it acknowledge to the microprocessor signaling ready for next data.

#3 Single Handshaking Input Output (Single Handshaking I/O)

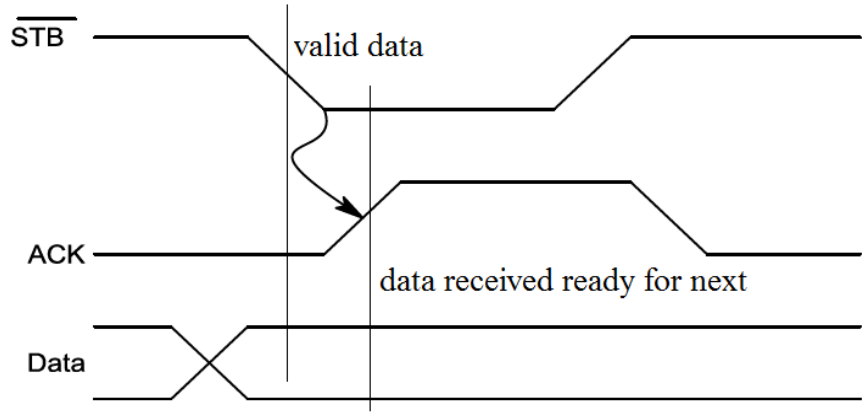


Fig. 3 Single Handshaking I/O Timing Diagram

- sending device puts parallel data at data-bus and asserts $\overline{STB}$ signaling to indicate valid data is present for you.
- receiver detects asserted $\overline{STB}$ signal, reads the data and sends an ACK signal indicating data has been received and ready for next data if any.
- sending device sends data when data is ready for receiving device.

#4 Double Handshaking Input Output (Double Handshaking I/O)

- Double Handshaking scheme, in general, is needed for more co-ordination between communicating devices (microprocessor and peripherals).
- Similar to single handshaking, it has two communication scenarios:

1. Input Handshaking (Peripheral to microprocessor)
2. Output Handshaking (Peripheral from microprocessor)

Double Handshaking Input Output

#1) Input Handshaking

- when peripheral wants to send data to microprocessor, it asserts strobe $\overline{STB}$ signal low to know if microprocessor is ready for data receive.
- If microprocessor acknowledge (ACK) to indicate it is ready for receiving data; then peripherals sends the byte of data and raise strobe ($\overline{STB}$) signal high.
- when microprocessor drops its acknowledge (ACK) line low indicating data has been read and ready for next byte of data.

Double Handshaking Input Output

#2) Output Handshaking

- For sending data to peripherals, microprocessor asserts $\overline{STB}$ signal to know if peripheral is ready for receiving data.
- Peripheral sends ACK to microprocessor indicating it is ready for receiving data byte.

#4 Double Handshaking Input Output (Simple Strobe I/O)

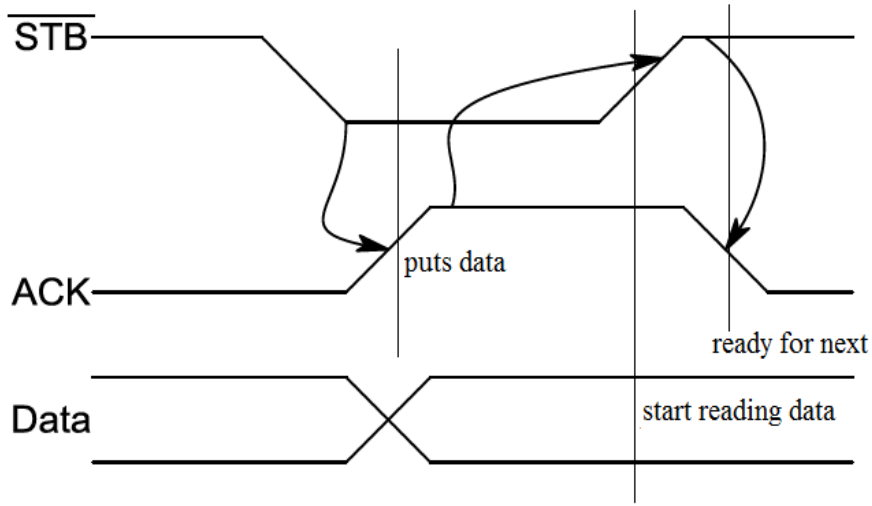


Fig. 4 Double Handshaking I/O Timing Diagram

- sending device put asserts $\overline{STB}$ signal to receiving device if ready for accept data.
- once receiver detects asserted $\overline{STB}$ signal, it raises ACK signal indicating ready to receive.
- sending device sends data and raises $\overline{STB}$ signal signaling valid data available on data bus.
- once receiver read data, it drops ACK signaling sender data has been read and ready for next data.

General Purpose Programmable I/O device 8255(PPI)

- it is general purpose programmable I/O device communicable to Intel microprocessor.
- it has 24 I/O pins categorized as group of 8 pins namely Port-A, Port-B, and Port-C (40 pins in total)
- Port-C can be further group as C-upper and C-lower, and it can also be used as individual pin.
- functions of these ports are defined with control word in control register.
- 8255 can function in two modes:
  1. Bit Set/Reset mode (BSR mode)
  2. I/O mode

General Purpose Programmable I/O device 8255

#1) Bit Set/Reset mode (BSR mode):

- BSR mode is used to set or reset individual pins of Port-C.

#2) I/O mode:

- I/O mode can further be divided into three modes: mode0, mode1 and mode2.

* In mode0, all ports function as simple I/O ports.
* In mode1, PortA and PortB use bits from PortC as handshake signaling, so it is also known as handshake mode.
  - there are two type of I/O data transfer: status check and Interrupt
* In mode2, PortB functions as either input or output, PortA function for bidirectional data transfer using handshake bits from PortC.

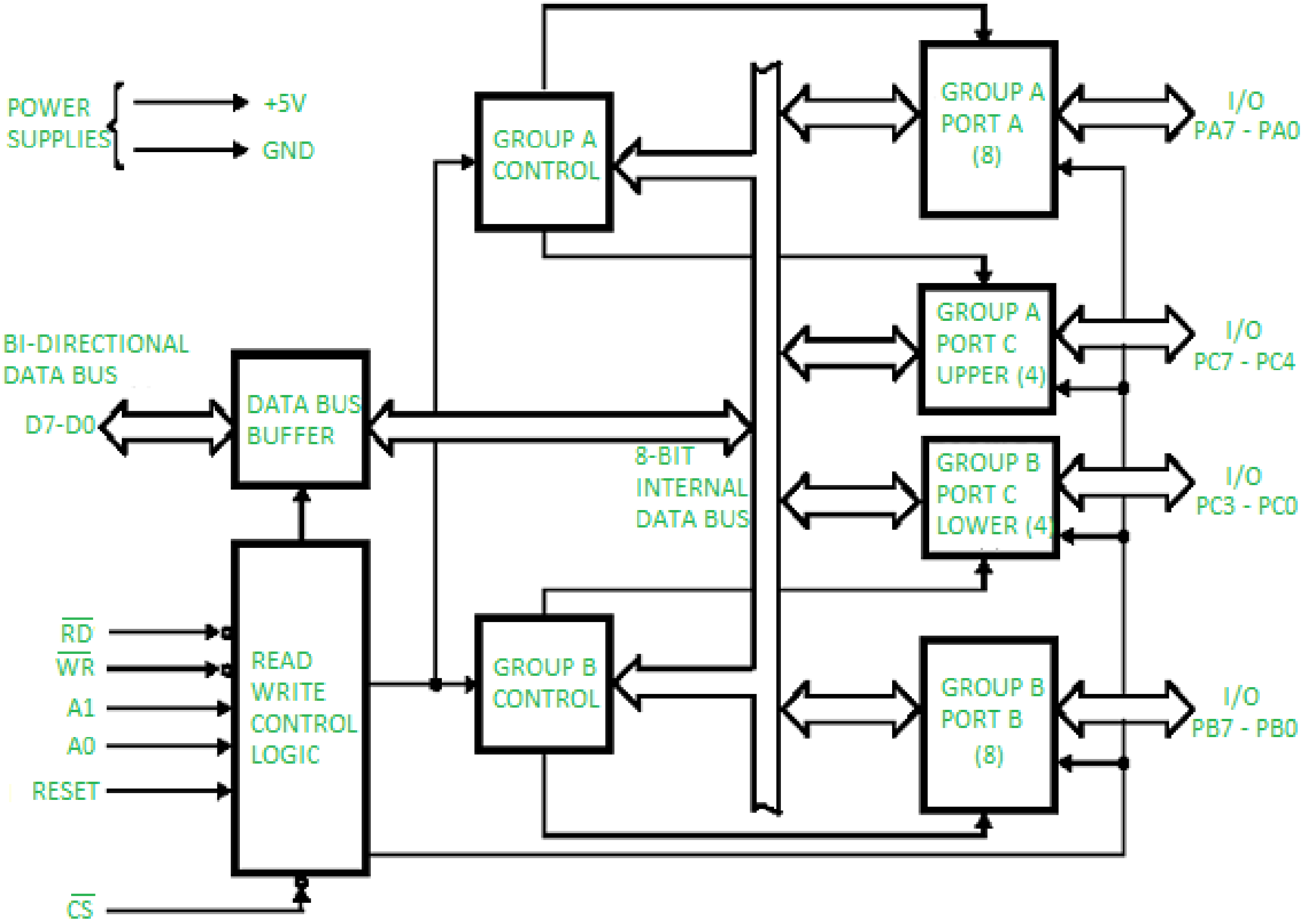
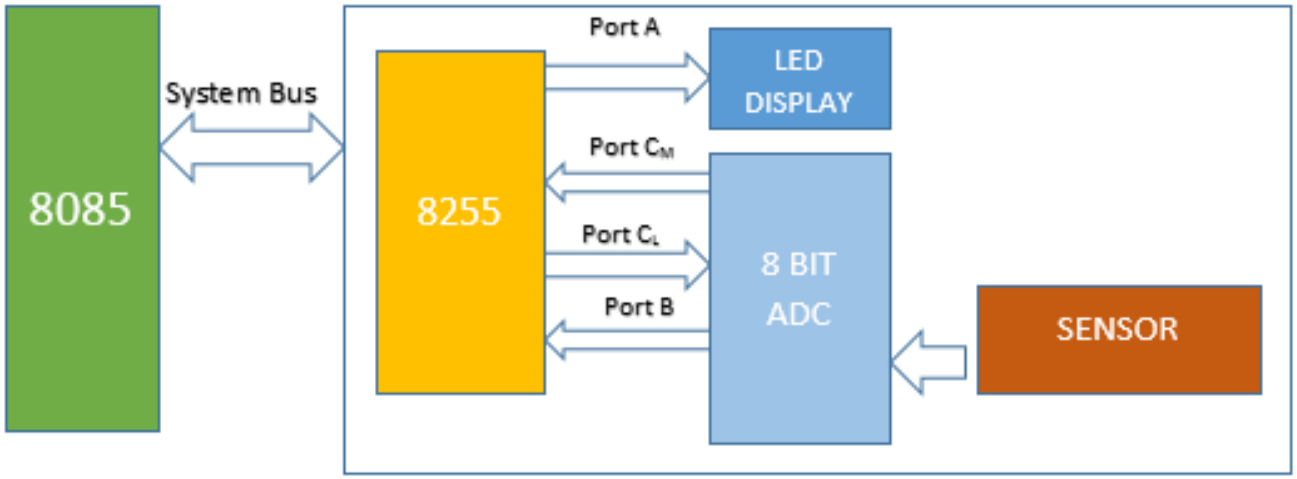


Fig. 5 8255 Internal Block Diagram

Interfacing block diagram



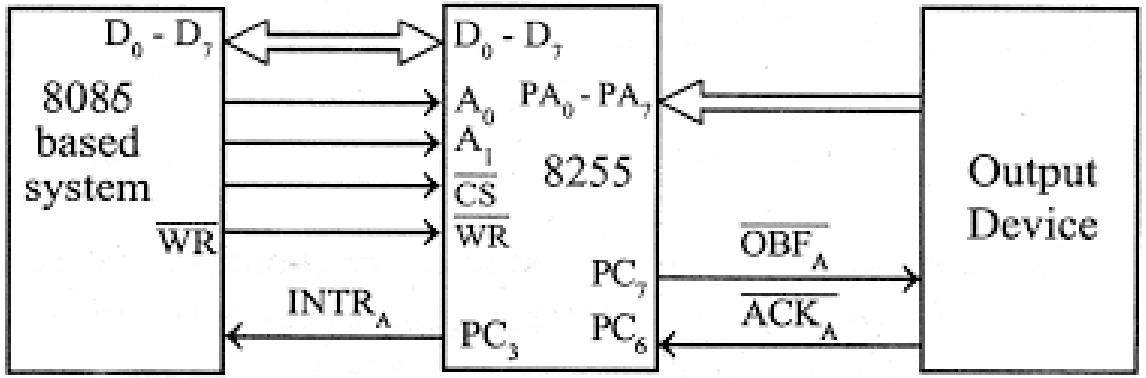


Fig. 6 Simple 8255 interfacing to I/O devices block diagram

Some main block components of 8255

Data Bus Buffer

- tri-state bi-directional buffer of 8-bit is used to interface 8255A to the system data-bus
- depending up on instruction either data is received or transmitted through the buffer by processor (CPU).
- control words and status information are also transferred through the data bus buffer.

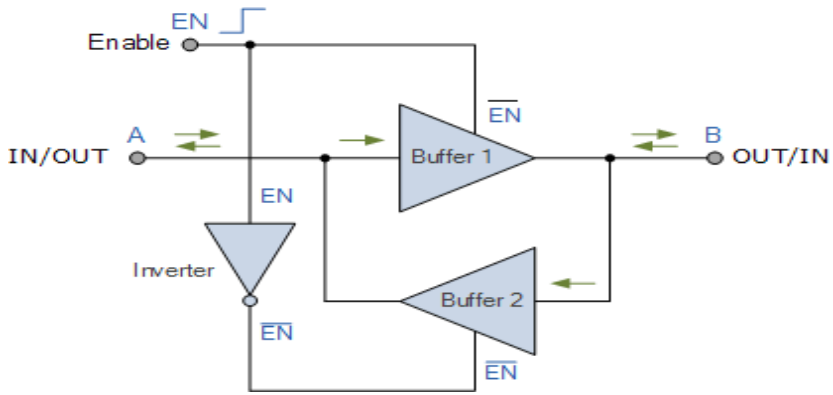


Fig. 7 tri-state bi-directional buffer for single bus

Some main block components of 8255

Read/Write Control logic

- it manages all the internal and external transfers of both data and control words.
- it accepts inputs from address bus and control buses of CPU.
- issues commands to both of the control GroupA and GroupB.

Chip Select ($\overline{CS}$)

- it enables the communication between 8255 and CPU.

Read($\overline{RD}$)

- it enables 8255 to send data or status information to CPU.

Write($\overline{WR}$)

- it enables CPU to write data or control words into 8255.

Reset(RESET

- it clears the content of control registers, sets all ports A, B, and C into input mode.

Some main block components of 8255

A_0 and A_1

- these are least significant bits of the address.
- these pins are used to select one of three ports or control word register.

Following can be state of the ports and control status for different combination of A_0A_1

$\overline{CS}$	A_1	A_0	Selected
0	0	0	Port A ~~(80H)~~
0	0	1	Port B ~~(81H)~~
0	1	0	Port C ~~(82H)~~
0	1	1	Control register ~~(83H)~~
1	X	X	8255 is not selected

Some main block components of 8255

GroupA and GroupB Controls

- CPU outs control words to 8255A
- Control word contains information such as mode, bit set, bit reset etc.
- Control blocks (GroupA & GroupB) accepts commands from Read/Write control logic.
- Receives control words from the internal data bus and issues the proper commands to its associated ports.
- control GroupA – controls PortA and PortC-upper (C_7 - C_4)
- control GroupB – controls PortB and PortC-lower (C_3 - C_0)

Some main block components of 8255

GroupA and GroupB Controls

- when A_1A_0 pins have values [1, 1], address mapping addresses the control register (8-bit register)
- The content of the register is control word which specifies the I/O function for each port.

Control word Format:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

- MSB (D_7) of control word determines either I/O function or BSR mode.

GroupA and GroupB Controls

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

Control word Format

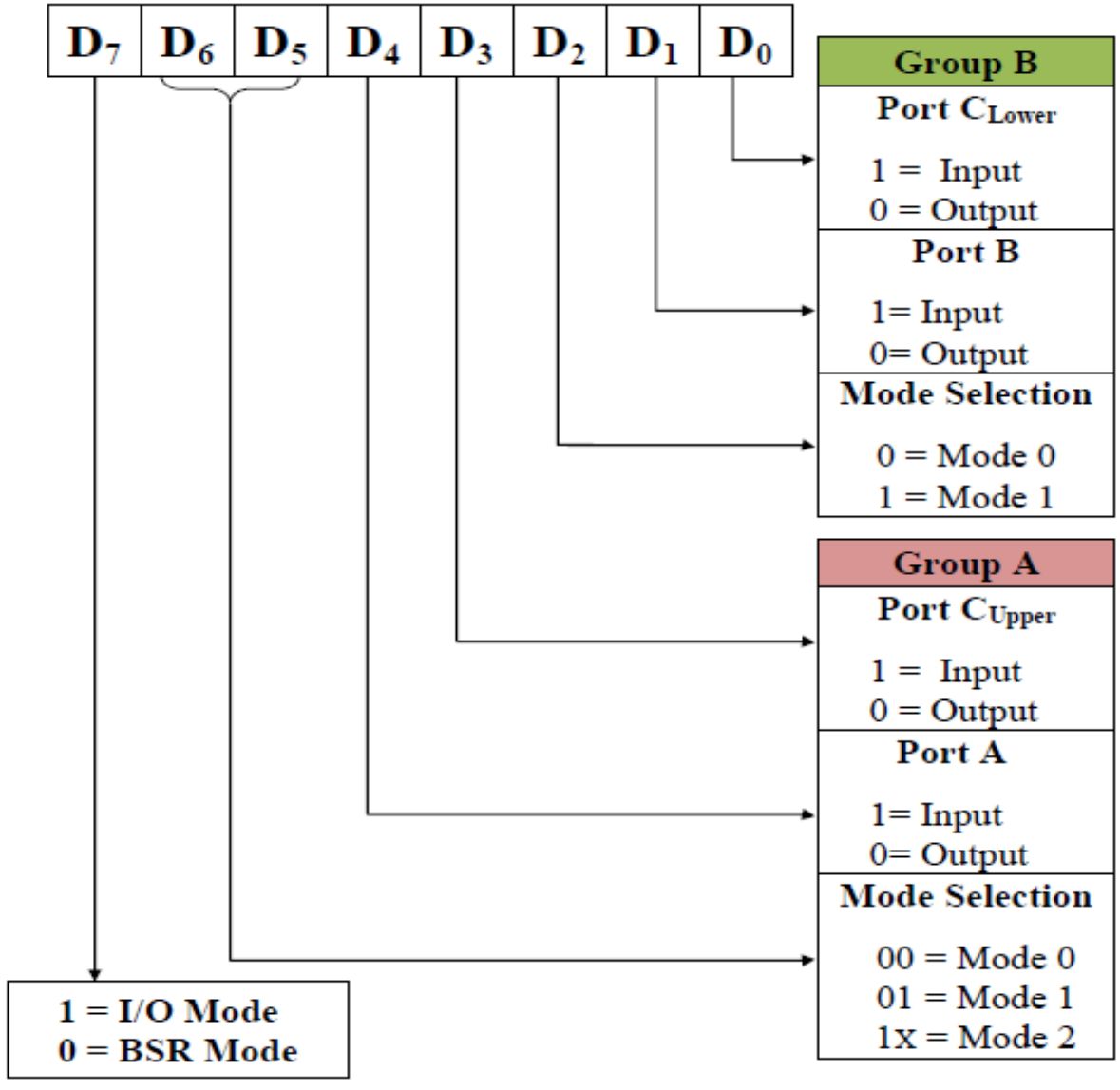
- ~~MSB (D_7) of control word determines either I/O function or BSR mode.~~

- if $D_7 = 1$, D_6-D_0 determines the I/O functions in various modes (Mode0, Mode1 or Mode2).
- if $D_7 = 0$, PortC operates in BSR mode. BSR function does not affect the function of PortA and PortB.

To communicate with peripherals through 8255

1. Find addresses of PortA, PortB, PortC and Control Register according to address lines A_1A_0 and Chip Select ($\overline{CS}$).
2. write control word in control register.
3. write I/O instruction to communicate with peripherals through PortA, PortB, and PortC.

Control Word Format



Control word Format for I/O mode

Some main block components of 8255

I/O Control word Examples

1. PortA $\Rightarrow$ mode1, output;
   PortB $\Rightarrow$ mode0, output;
   PortC $\Rightarrow$ lower pins as output and remaining as output.

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

2. PortA $\Rightarrow$ mode0, output;
   PortB $\Rightarrow$ mode0, output;
   PortC $\Rightarrow$ lower pins as output and remaining as input.
3. PortA $\Rightarrow$ mode2, bidirectional;
   PortB $\Rightarrow$ mode0, input;
   PortC $\Rightarrow$ lower pins as output.

Operating mode of 8255

Mode 0 (Basic I/O)

- this is simple I/O configuration for all ports (A,B,C)
- there is no handshaking
- output are latched but not input.

- Two 8-bit ports A and B
- Two 4-bit ports (PortC-upper and PortC-lower)
- Any port can be configured as I/P or O/P.

Operating mode of 8255

Mode 1 (Strobe I/O)

- transfer of I/O data to or from a specified port takes place in conjunction with with strobes or handshaking.
- PortA and PortB use signals or lines from Portc for handshaking.

- Contains two groups Group-A and Group-B
- Each group contains one 8-bit data ports and one 4-bit control/data port
- Both Input and Output data are latched
- 4-bit port (PortC-Upper/PortC-Lower) is used for control and status of 8-bit data port.

Operating mode of 8255

Mode 2 (Strobe bidirectional I/O)

- handshaking is used to maintain proper flow discipline
- interrupt and enable/disable functions are also available.

- used in GroupA only
- one 8-bit bidirectional bus (PortA) and 5-bit control port(PortC).
- both i/p and o/p data are latched.
- 5-bit port (PortC) is used for control and status of 8-bit data port.

Operating mode of 8255

BSR Mode (Bit Set/Reset)

- this mode is concern only to eight bits from PortC.
- in this mode, portC can be set or reset with appropriate control words in the control register.
- control word with $D_7 = 0$ is recognized as BSR mode control while previously transmitted control word configuration will not be changed.

  that means I/O operation of portA and PortB are not affected by BSR control word.
- individual pins/bits of PortC can be used for applications as switch.

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

Operating mode of 8255

BSR Mode (Bit Set/Reset)

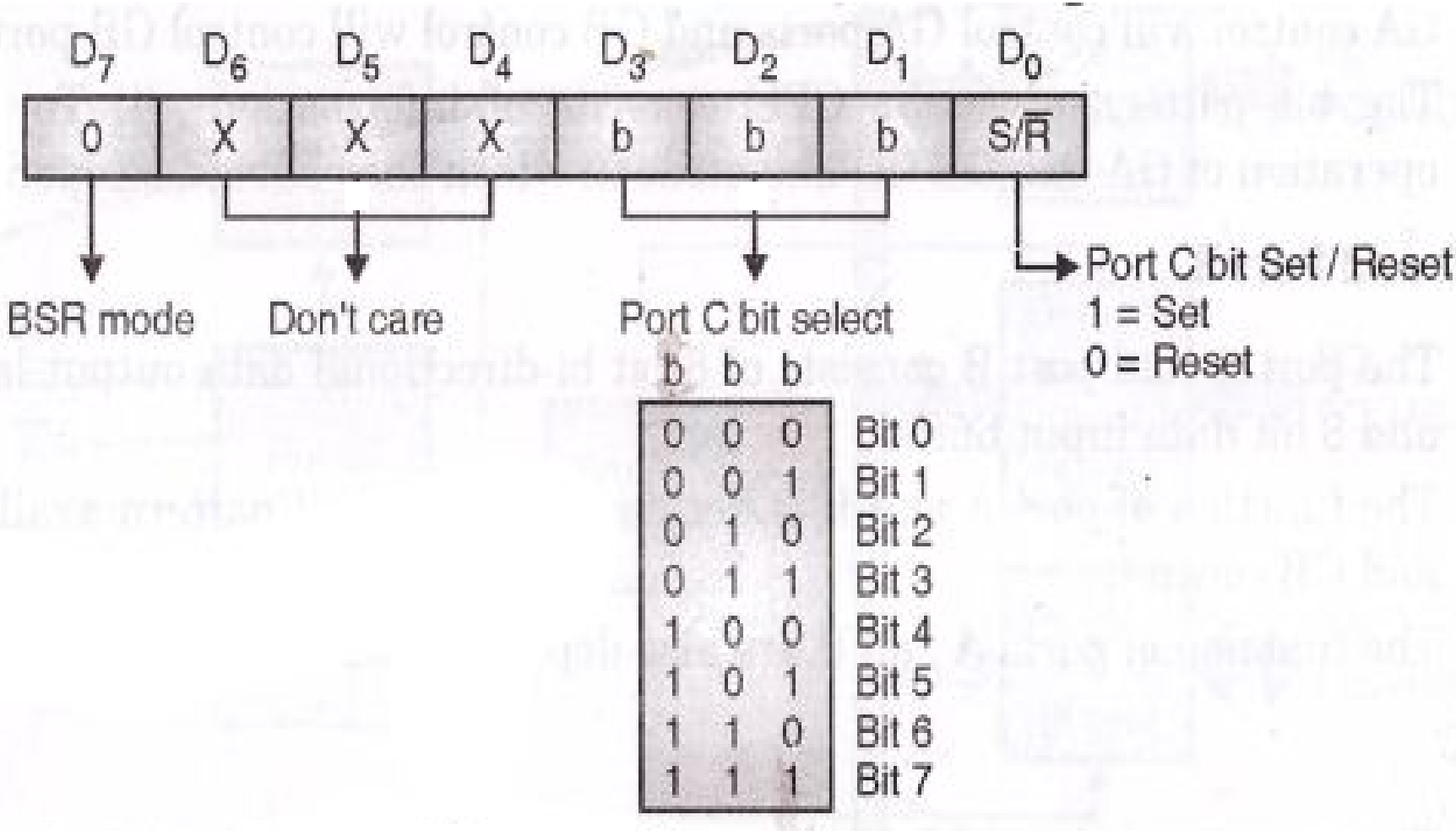


Fig. 8 Control word Format BSR mode

Operating mode of 8255

BSR Control word Examples

1. To set PC_7:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	x	x	x	-	-	-	-

2. To Reset PC_3

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	x	x	x	-	-	-	-

To Sum up, Control words can be:

1. either mode definition word or
2. Bit Set/Reset word.

Programming and Operation of 8255

A High to RESET pin

- once high to RESET pin, all PortA, PortB, PortC pins are converted to input mode
- changing mode of ports can be caused with control word to control register.

Modes for GroupA and GroupB

- mode definition control word defines the mode of GroupA and GroupB with portC taking responsibilities(control signal).
- for instance, if Group A is programmed for Mode 0, GroupB for Mode 1, PortA and PortC-upper can be either input or output while PortB can be programmed for either input or output using PC_0, PC_1, PC_2 for handshaking.
- similarly, when PortA is programmed to operate in Mode 1 (input), $PC_3, PC_4, PC5$ (for output PC_3, PC_6, PC_7) are used for handshaking.

Programming and Operation of 8255

Mode Definition or BSR

- D_7 of control word defines the control word to be either mode definition or BSR.
- if $D_7 = 0$, the control word is BSR, and if $D_7 = 1$, the control word is mode definition control word.
- all the control words are loaded to the same control register in a respective way.
- control register is accessed when $A_0A_1 = 11$ and both $\overline{WR}$ and $\overline{CS}$ to be 0.

Programming and Operation of 8255

Programming in BSR Mode(Bit Set/Reset Mode)

- this mode is configure when PortC is used as control for PortA and PortB with bit set/reset.
- BSR control word with $D_7 = 0$ does not affect the I/O configuration of PortA and PortB.

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

Programming and Operation of 8255

case	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	HexCOde
Reset PC_0	0	X	X	X	-	-	-	0	00H
Set PC_0	0	X	X	X	-	-	-	1	01H
Reset PC_1	0	X	X	X	-	-	-	0	02H
Set PC_1	0	X	X	X	-	-	-	1	03H
Reset PC_2	0	X	X	X	-	-	-	0	04H
Set PC_2	0	X	X	X	-	-	-	1	05H
Reset PC_3	0	X	X	X	-	-	-	0	06H
Set PC_3	0	X	X	X	-	-	-	1	07H
Reset PC_4	0	X	X	X	-	-	-	0	08H
Set PC_4	0	X	X	X	-	-	-	1	09H
Reset PC_5	0	X	X	X	-	-	-	0	0AH
Set PC_5	0	X	X	X	-	-	-	1	0BH
Reset PC_6	0	X	X	X	-	-	-	0	0CH
Set PC_6	0	X	X	X	-	-	-	1	0DH
Reset PC_7	0	X	X	X	-	-	-	0	0EH
Set PC_7	0	X	X	X	-	-	-	1	0FH

Programming and Operation of 8255

Programming in Mode 0(Basic I/O)

- PortA, PortB, and PortC is configured as simple I/O port using appropriate control word loaded in the control register.
- D_6, D_5 is set to 0 to configure GroupA Port to Mode 0 and D_2 set to 0 to configure GroupB port to Mode 0.
- Depending up on the value of D_4, D_3, D_1, D_0 PortA, PortB, PortC-upper and PortC-lower will be either output type or input type.

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

Mode 0 Example

Timing Diagram for Mode-0 Input/Output

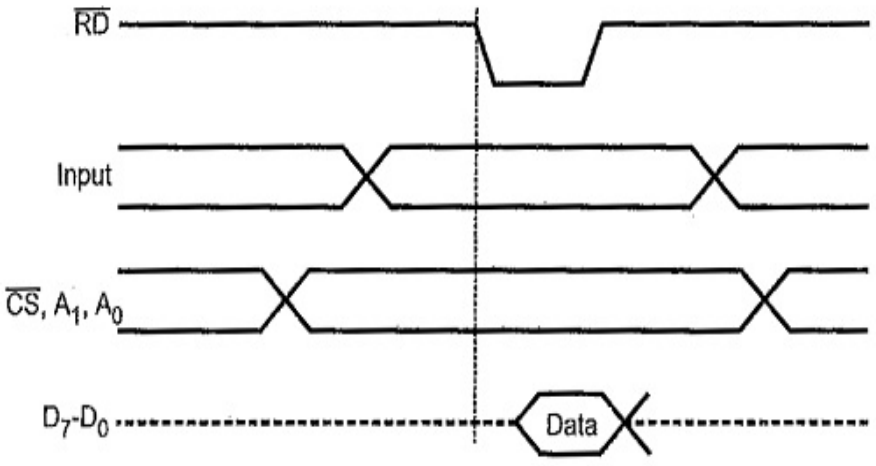


Fig. 9 Timing Diagram for Mode0/Input

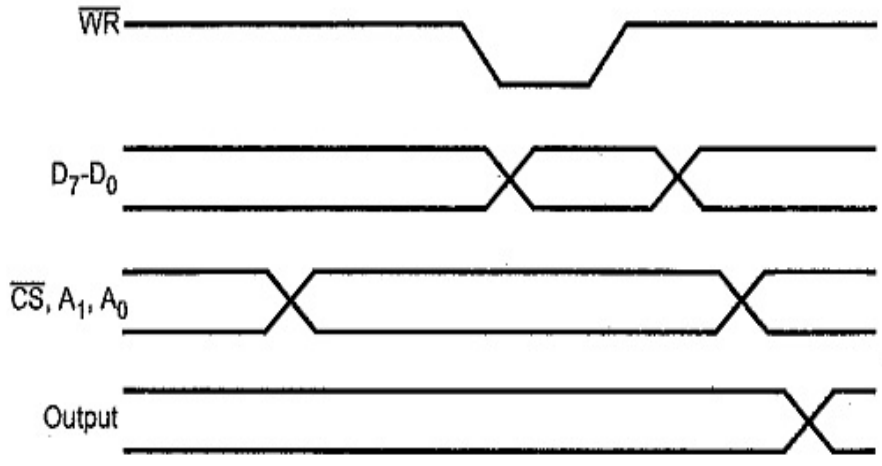
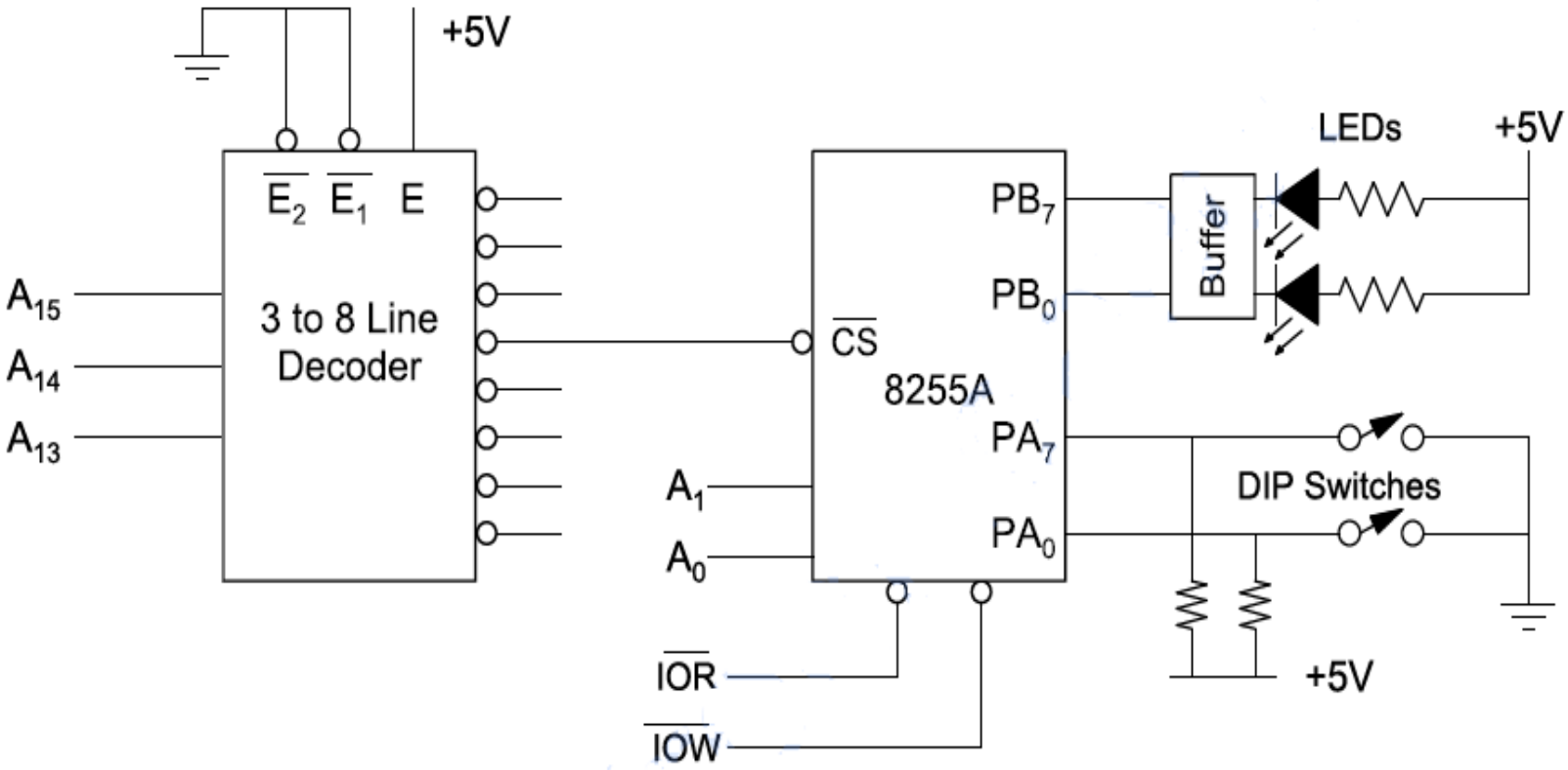


Fig. 10 Timing Diagram for Mode0/Output

Mode 0 Example

Write a program to read Dip (Dual-in-line package) switches and display the reading from portA to portB.



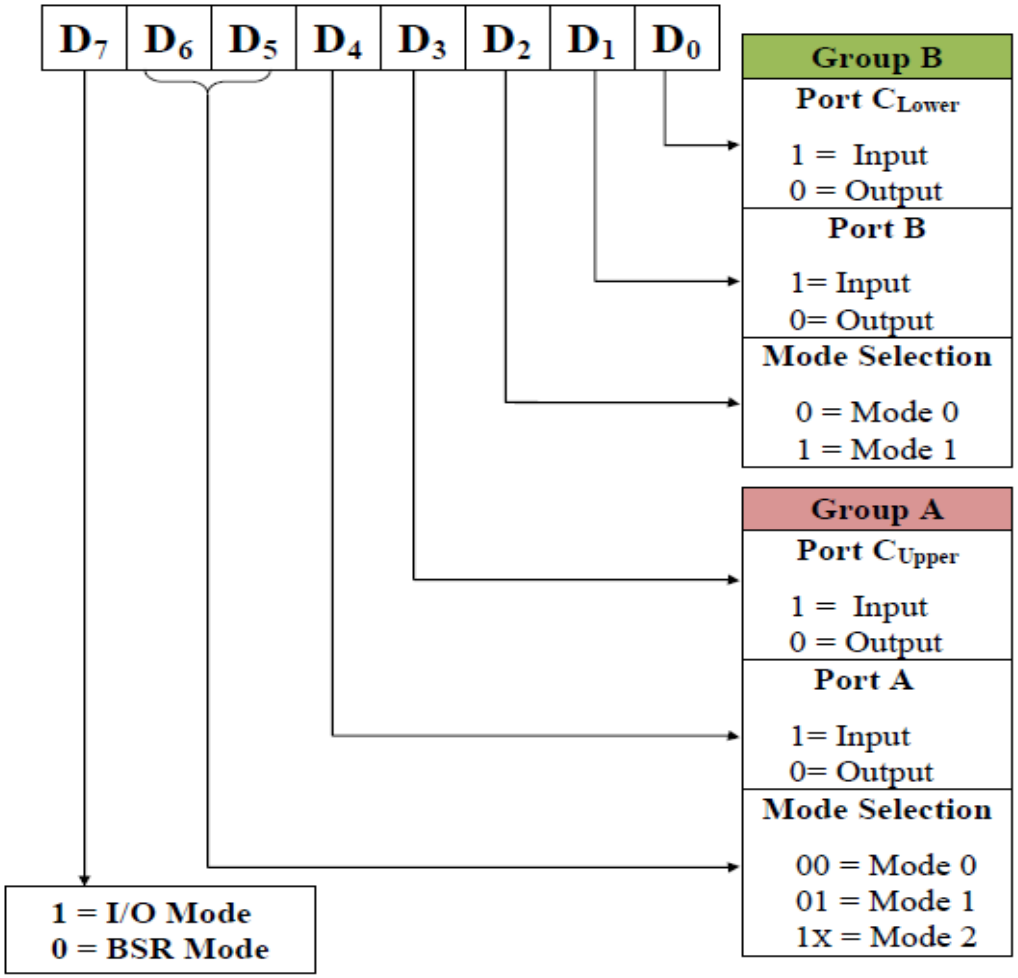
$\overline{CS}$ to be low:

$A_{15}A_{14}A_{13}A_{12}.....A_1A_0$ = 011X-XXXX-XXXX-XXA_1A_0

Mode 0 Example

Solution:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
-	-	-	-	-	-	-	-



Port Address:

Port A: ——

Port B: ——

Port C: ——

Program or subroutine:

——: Control-Word to Acc

——: Acc to Control Reg.

——: read switch at PortA

——: out Acc to PortB.

RET: (Return to main)

Control word Format for I/O mode

Programming and Operation of 8255

Programming in Mode 1(Strobe I/O)

In Mode1, handshaking signals are exchanged between MPU and Peripherals for data exchange.

PortA

When function as input

- PC_3, PC_4, PC_5 are control signals
- PC_6, PC_7 are I/O as defined by D_3 in control word.

When function as Output

- PC_3, PC_6, PC_7 are control signals
- PC_4, PC_5 are I/O as defined by D_3 in control word.

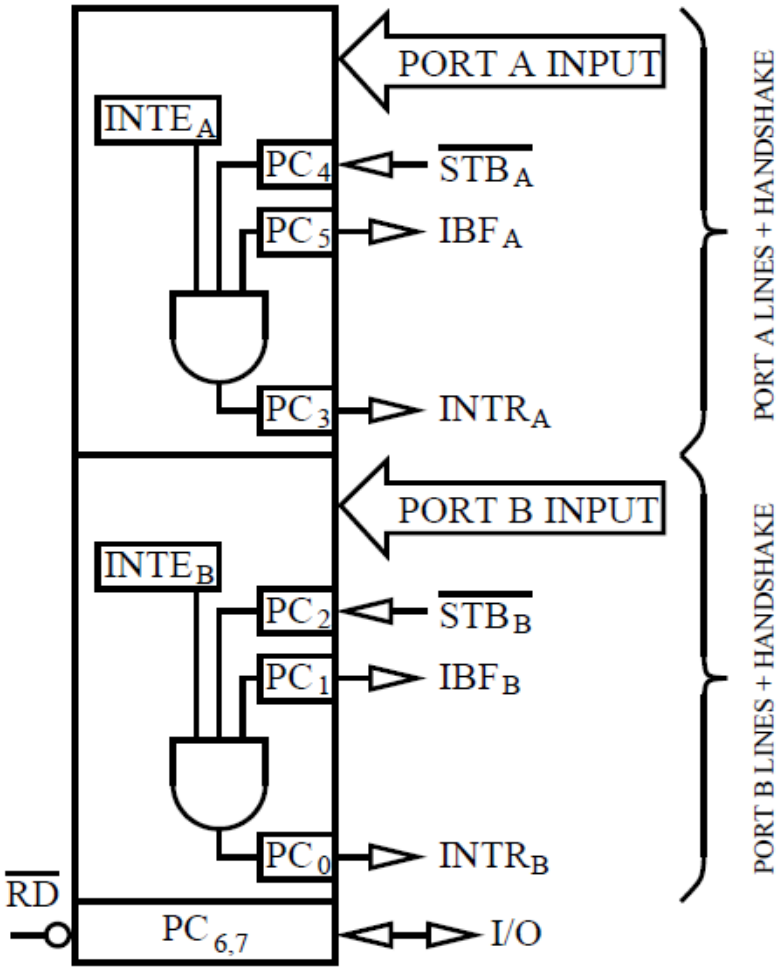
PortB

For both input or output configuration, PC_0, PC_1, PC_2 are control signals.

Programming and Operation of 8255

Programming in Mode 1(PortA and PortB $\Rightarrow$ Input)

Control word: $D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0 =$ $\boxed{1}, \boxed{0,1}, \boxed{1}, \boxed{b}, \boxed{1}, \boxed{1}, \boxed{X}$



- low $\overline{STB}$ is generated by peripherals when data is loaded.
- 8255 responses generating IBF_A(Input Buffer Full) $INTR_A$(Interrupt Request)
- IBF_A signaling signifies input latch received a byte, and once the data is read, it will be reset.
- input/output data are latched.

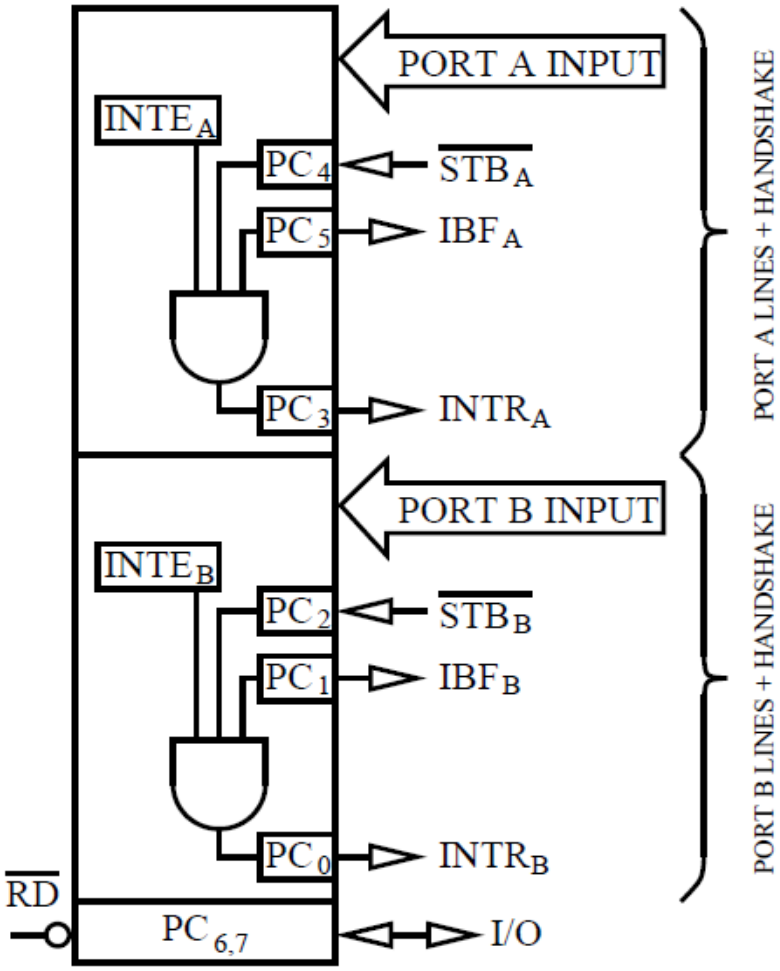
.....

Fig. 11 Mode1 input configuration

Programming and Operation of 8255

Programming in Mode 1(PortA and PortB ⇒ Input)

Control word: $D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0 = \boxed{1}, \boxed{0,1}, \boxed{1}, \boxed{b}, \boxed{1}, \boxed{1}, \boxed{X}$



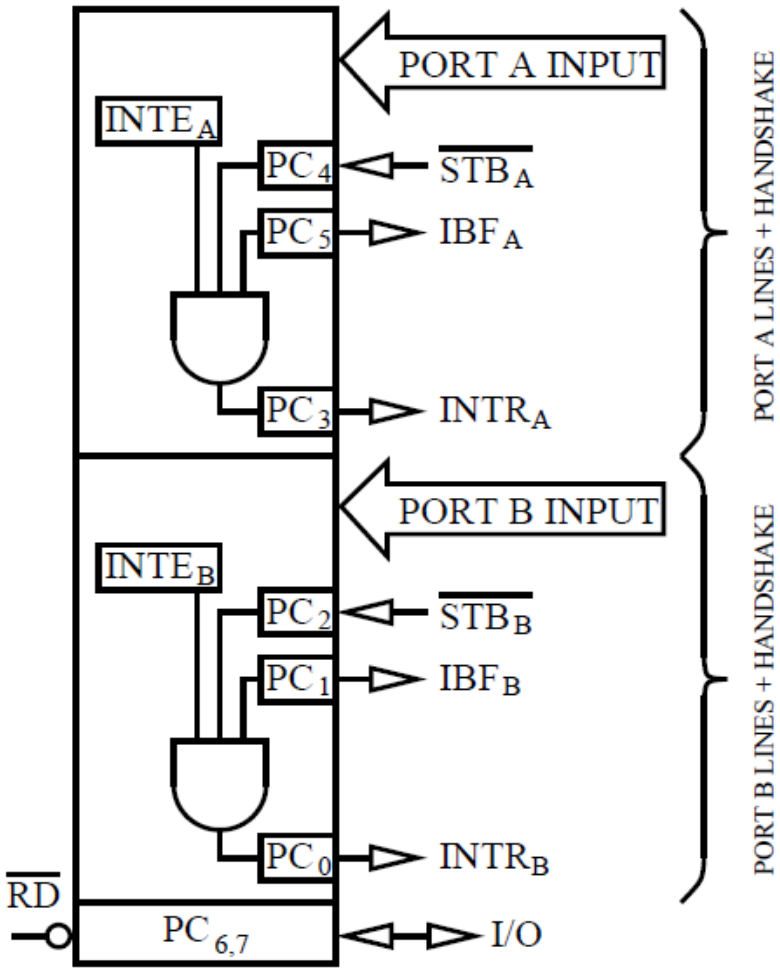
Mode1 input configuration

.....

- $INTR_A$ is used to interrupt MPU, will be generate when $\overline{STB_A}$, IBF_A and $INTE_A$ has logic 1
- $INTE_A$ is internal flip-flop which is set/reset in BSR mode with PC_4.
- $INTE_B$ is internal flip-flop which is set/reset in BSR mode with PC_2 set and similar to PortB as described above for portA.
- upon reading PortA, falling edge of $\overline{RD}$ reset $INTR_A$ and rising edge of $\overline{RD}$ reset IBF_A ⇒ input buffer empty.

8255 functioning as Mode 1/ Input

Mode1/Input Timing Diagram



Mode1 input configuration

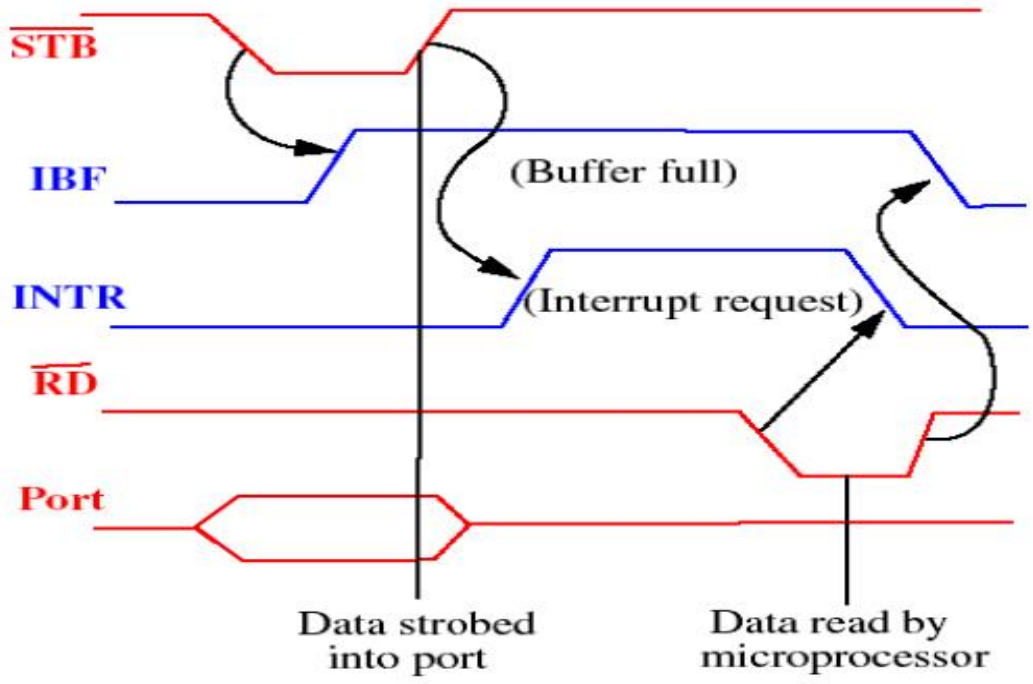
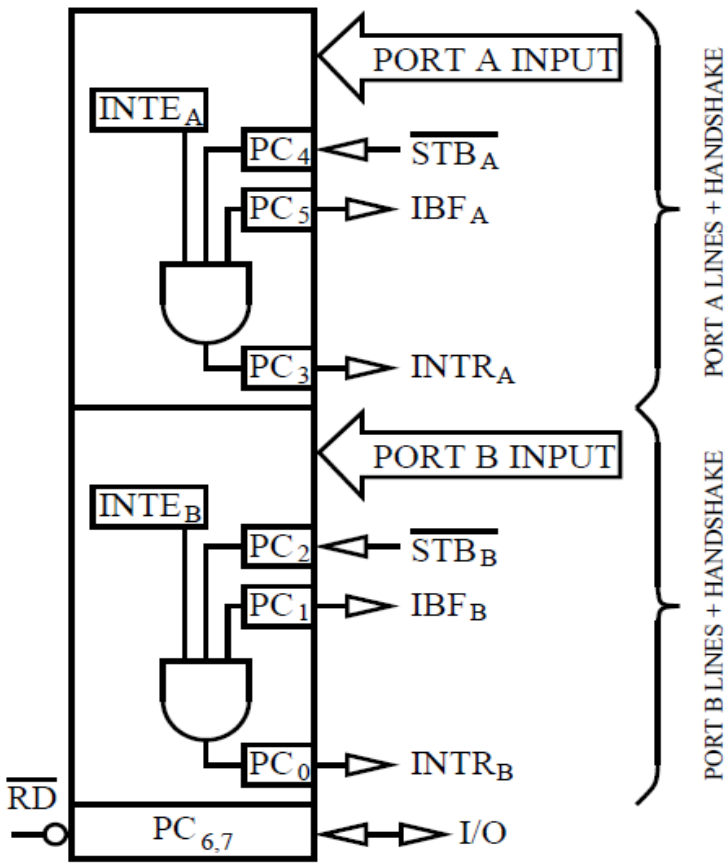


Fig. 12 Timing Diagram for Mode1/Input (Strobed Input)

Programming and Operation of 8255

Programming in Mode 1(PortA and PortB ⇒ Input)



Control word:

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

- if the CPU is busy with other system operation, it can read data Once it is interrupted – Interrupt controlled I/O.

- If the CPU is not busy, it continuously read the status word to check if IFB_A is full - program controlled I/O.

Status word for Mode1 /input

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	$INTR_B$

Programming and Operation of 8255

Programming in Mode 1(PortA/PortB $\Rightarrow$ Input)/Control Signals

Status-Word for Mode1/Input

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	$INTR_B$

$\overline{STB}$(Strobe Input)

- low on this input loads data in the input latch.
- in response to $\overline{STB}$, 8255 generates IBF and INTR.

IBF(Input Buffer Full)

- A high on this output indicates that the data bus has been loaded into the input latch.
- IBF is set by $\overline{STB}$ input being low
- it is reset when $\overline{RD}$ is rising.

Programming and Operation of 8255

Programming in Mode 1(PortA/PortB $\Rightarrow$ Input)/Control Signals

Status-Word for Mode1/Input

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	$INTR_B$

INTR(Interrupt Request)

- is an output signal to interrupt CPU.
- generated when $\overline{STB}$, IBF and INTE are all logic one.
- it is reset when $\overline{RD}$ is rising.

INTE(Interrupt Enable)

- it is internal flip-flop used to enable/disable the generation of INTR signal.
- two flip-flop $INTE_A$ and $INTE_B$ are set/reset using BSR mode through PC_4 and PC_2 respectively.

Programming and Operation of 8255

Programming in Mode 1(PortA and PortB $\Rightarrow$ OutPut)

Control word: $D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0 = \boxed{1}, \boxed{0,1}, \boxed{0}, \boxed{b}, \boxed{1}, \boxed{0}, \boxed{X}$

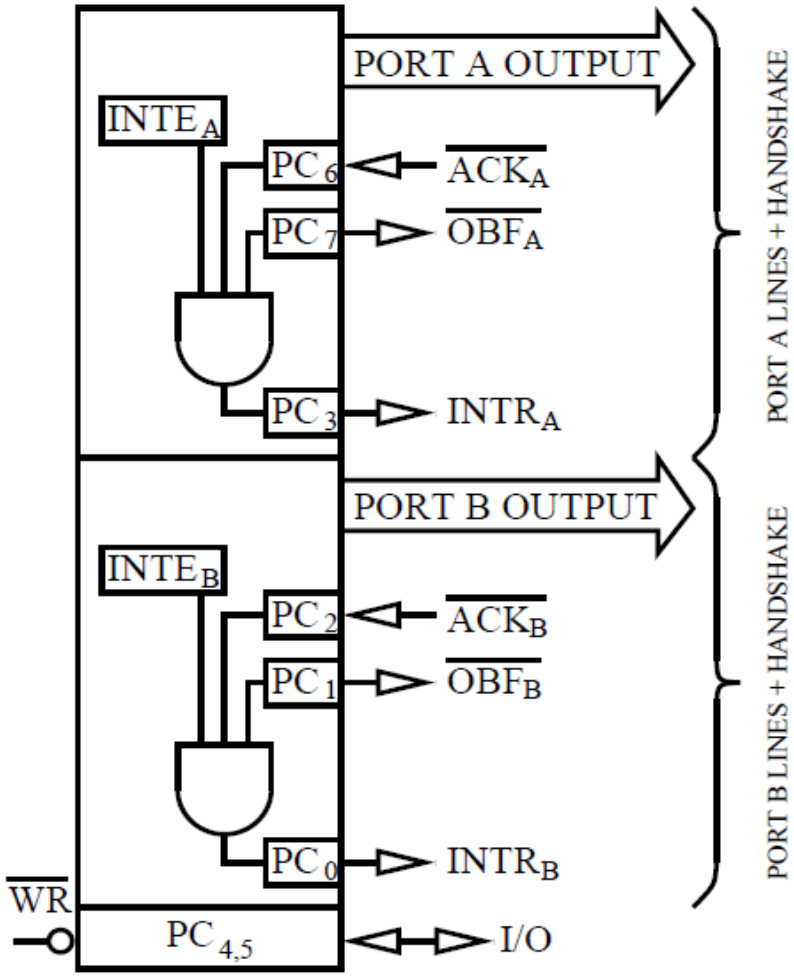


Fig. 13 Mode1 input configuration

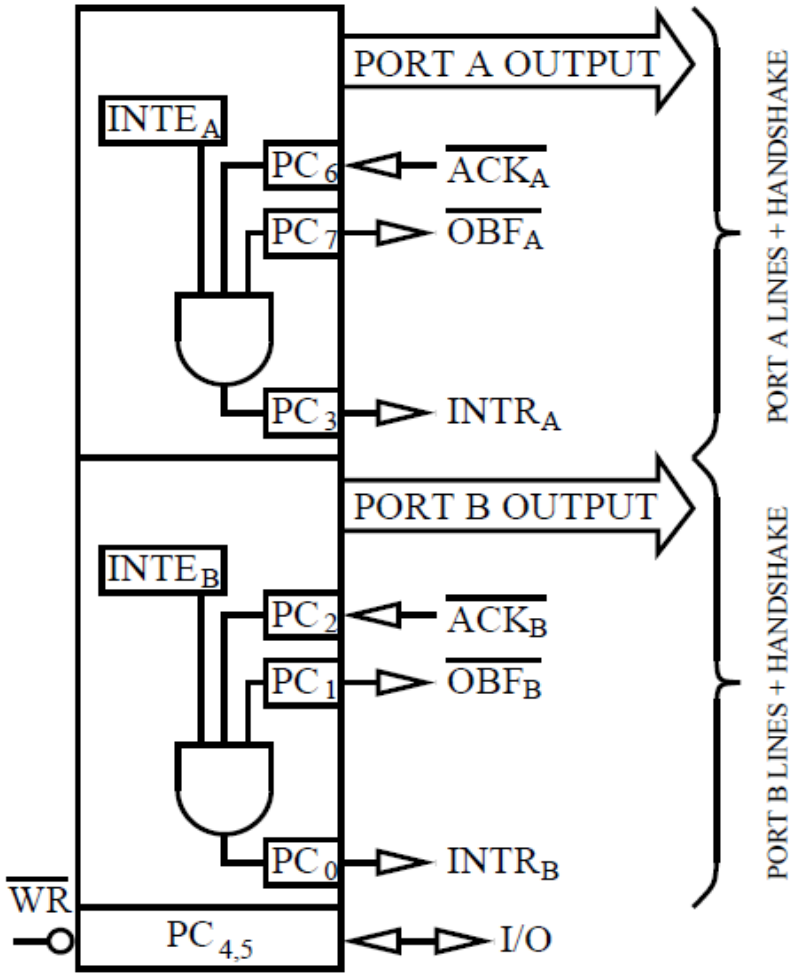
- When control word is loaded, GroupA and GroupB ports are configured to mode1/Output
- CPU then send data to peripheral through PortA(PortB) as on output port.
- $\overline{OBF_A}$ (Output Buffer Full) goes low on the rising edge of $\overline{WR}$ signal - once CPU writes data into 8255

.....

Programming and Operation of 8255

Programming in Mode 1(PortA and PortB $\Rightarrow$ Output)

Control word: $D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0 = \boxed{1}, \boxed{0,1}, \boxed{0}, \boxed{b}, \boxed{1}, \boxed{0}, \boxed{X}$



Mode1 input configuration

.....

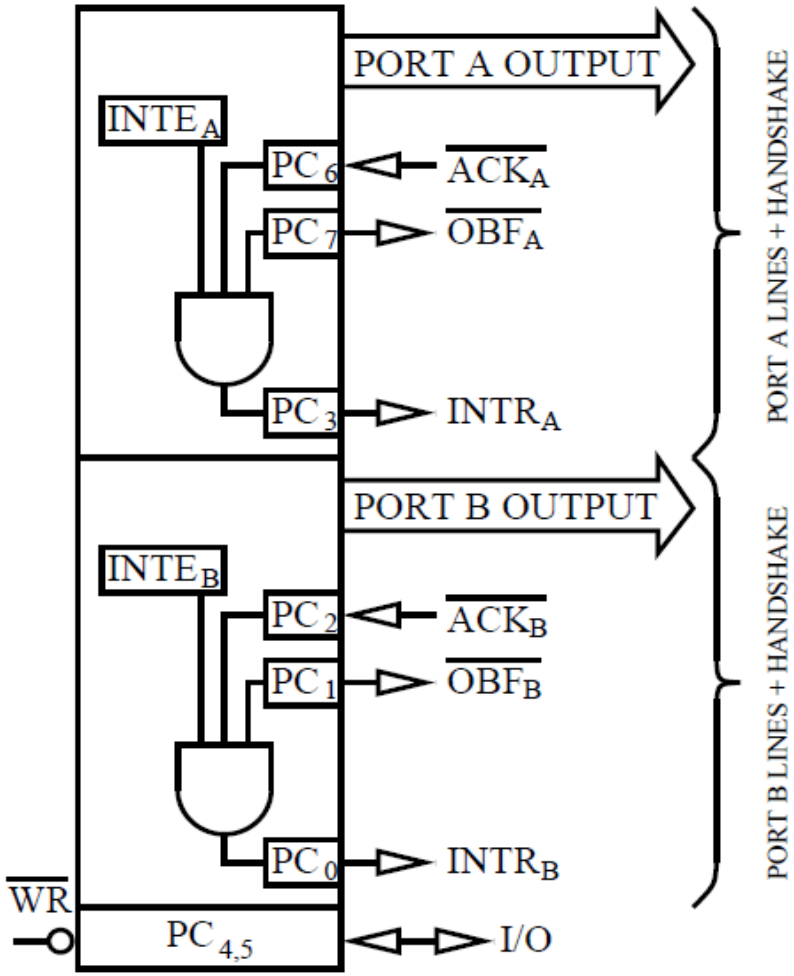
- $\overline{OBF_A}$ can be treated as strobe signal to the peripheral to latch the content of PortA.
- Once the data has been read, peripheral acknowledges by low $\overline{ACK}$.
- low $\overline{ACK}$ resets $\overline{OBF_A}$ indicating buffer is free and ready for next data.

...

Programming and Operation of 8255

Programming in Mode 1(PortA and PortB $\Rightarrow$ Output)

Control word: $D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0 = \boxed{1}, \boxed{0,1}, \boxed{0}, \boxed{b}, \boxed{1}, \boxed{0}, \boxed{X}$



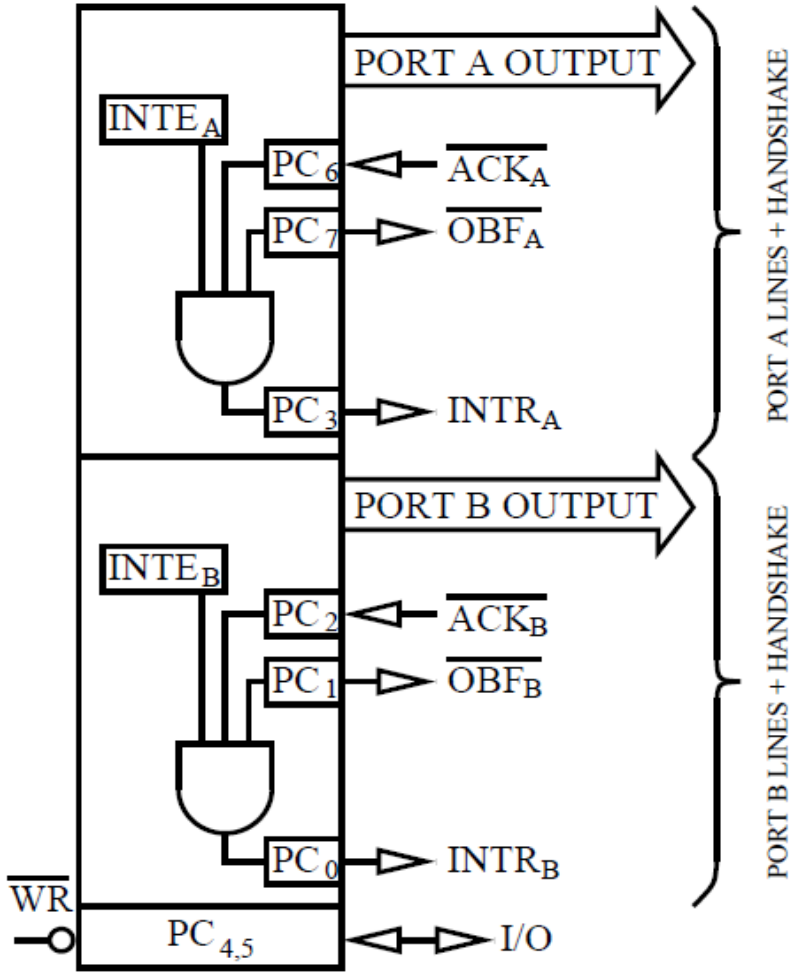
Mode1 input configuration

.....

- $\overline{INTR_A}$ is high when $\overline{ACK}$ and $\overline{OBF_A}$ both are high.
- $INTR_A$ is used to interrupt CPU whenever output buffer is empty.
- falling edge of $\overline{WT}$ resets $INTR_A$ and CPU writes data onto portA.
- to reset $INTR_A$, PC_6 is loaded with 0 in BSR mode which reset $INTE_A$.

8255 functioning as Mode 1/ Input

Mode1/Output Timing Diagram



Mode1 input configuration

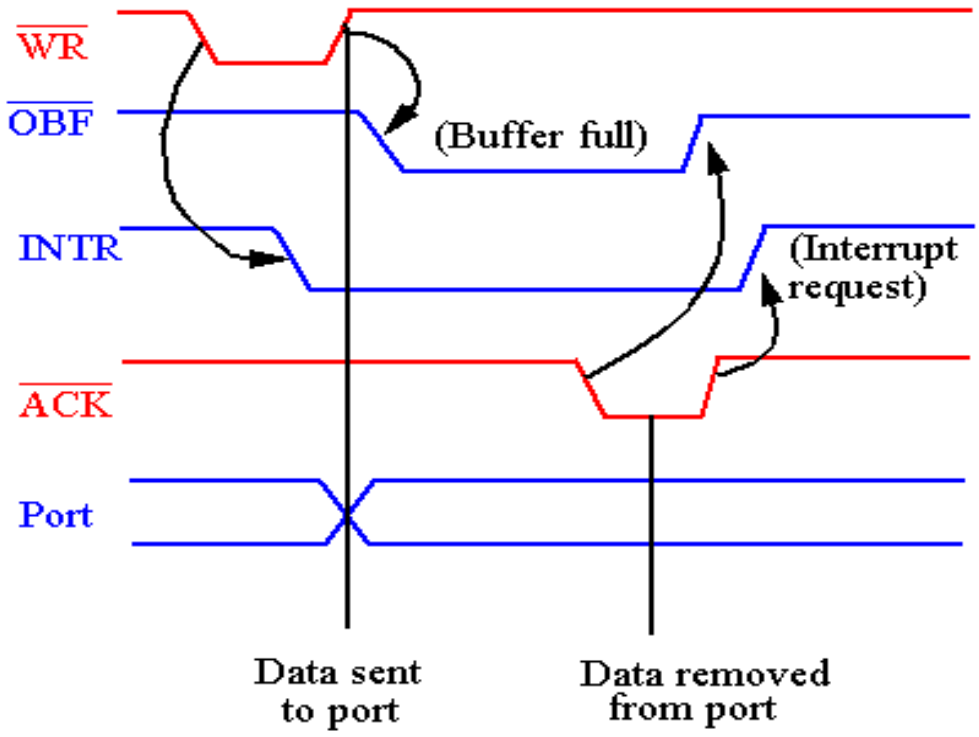


Fig. 14 Timing Diagram for Mode1/Output

Programming and Operation of 8255

Programming in Mode 1(PortA/PortB $\Rightarrow$ Output)/Control Signals

Status-Word for Mode1/Input

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
$\overline{OBF_A}$	$INTE_A$	I/O	I/O	$INTR_A$	$INTE_B$	$\overline{OBF_B}$	$INTR_B$

$\overline{OBF}$(Output Buffer Full)

- $\overline{OBF}$ will go low to indicate the CPU has written data out to specified port.
- $\overline{OBF}$ will be set with the rising edge of $\overline{WR}$ input and reset by $\overline{ACK}$ input being low.

$\overline{ACK}$(Acknowledgment Input)

- low on this informs 8255 that the data from portA/B has been accepted.

Programming and Operation of 8255

Programming in Mode 1(PortA/PortB $\Rightarrow$ Output)/Control Signals

Status-Word for Mode1/Input

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
$\overline{OBF_A}$	$INTE_A$	I/O	I/O	$INTR_A$	$INTE_B$	$\overline{OBF_B}$	$INTR_B$

INTR(Interrupt Request)

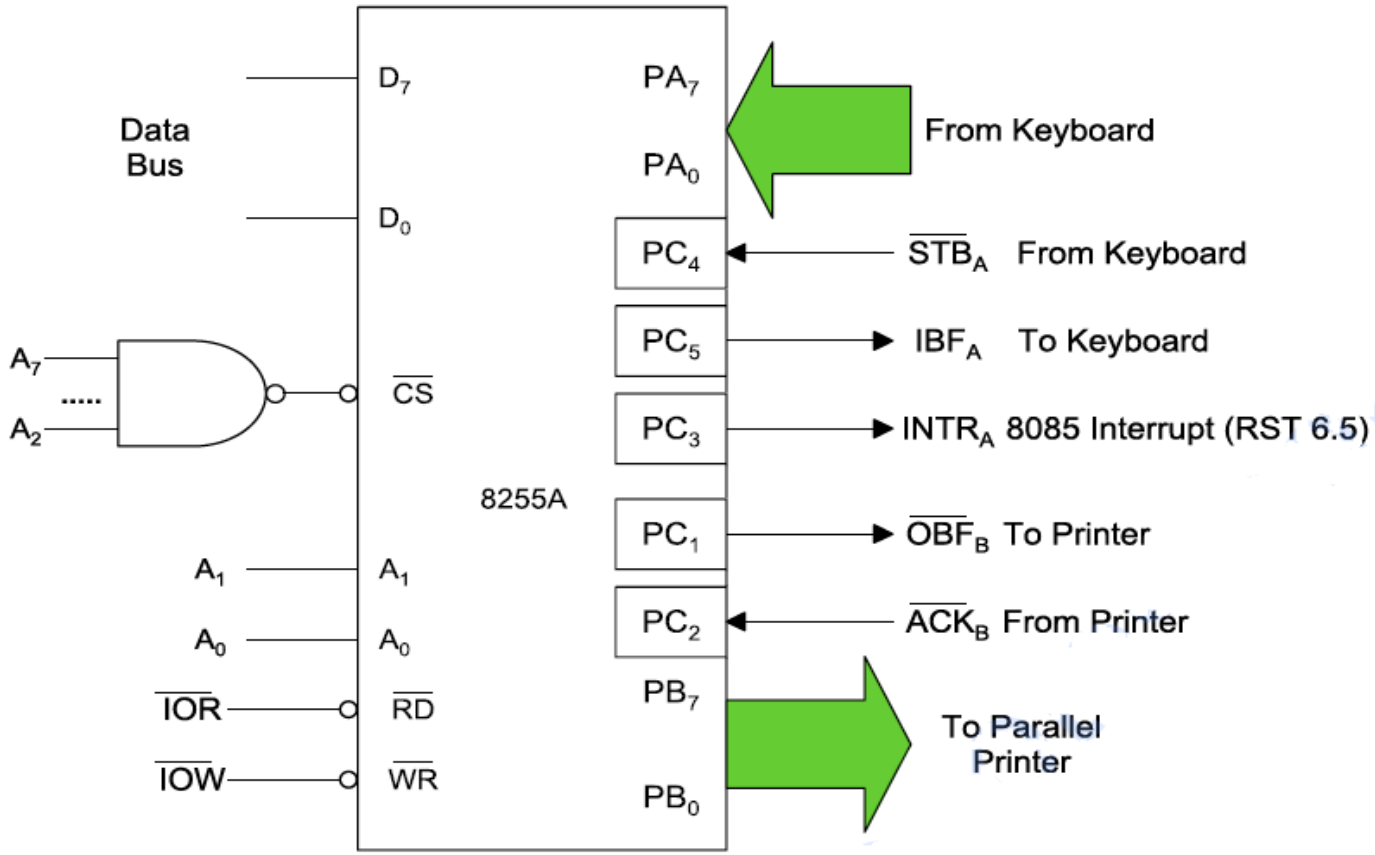
- high on this pin is used to interrupt the CPU signaling peripheral received the data.
- when an INTE, OBF and ACK all are one and reset by falling edge of $\overline{WR}$

INTE(Interrupt Enable)

- this is an internal flip-flop to a port and needs to be set to generate the INTR signal
- the two flip-flop $INTE_A, INTE_B$ are set/reset using the BSR mode through PC_6 and PC_2 respectively.

Programming 8255 PPI example

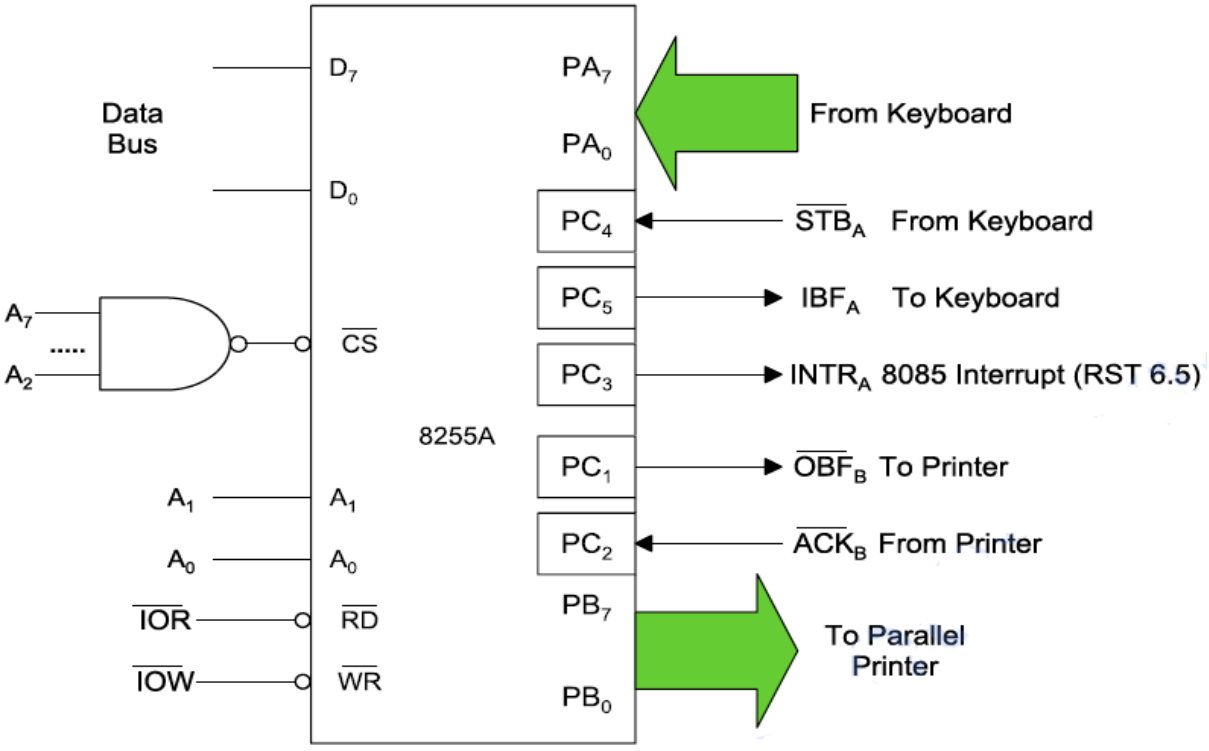
For given diagram; PortA accept keyboard input with interrupt I/O and portB output for printer with status check I/O.
Write a subroutine for the design scenario described.



Mode1 I/O example

Programming 8255 PPI example

a# Control Register and Port Addresses:



Address for different Ports:
PortA:——
PortB:——
PortC:——
Control Register:
——

Mode1 I/O example

Programming 8255 PPI example

b# Different Control words:

I/O control word:

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

BSR Control word:

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

Status word check for $\overline{OBF_B}$:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
x	x	x	x	x	x	$\overline{OBF_B}$	x

Programming 8255 PPI example

——— : I/O control to ACC
——— : ACC to Control Reg.
——— : To set $PC_4/INTE_A$
——— : BSR to Control Reg.
——— : Enable Interrupt
CALL READ:
CALL PRINT:
HLT: terminate

READ:
——— : Read PortA
——— : save ACC content
RET :

PRINT:
——— : PortC to check status
——— : check status word
JZ PRINT
——— : content C to ACC
——— : out ACC to PortB
RET

Programming in Mode2(Strobe bidirectional)

When 8255 is configured in mode2, $PC_7 - PC_3$ are used for handshaking by portA.

PortB can be either configured in mode0 or mode1

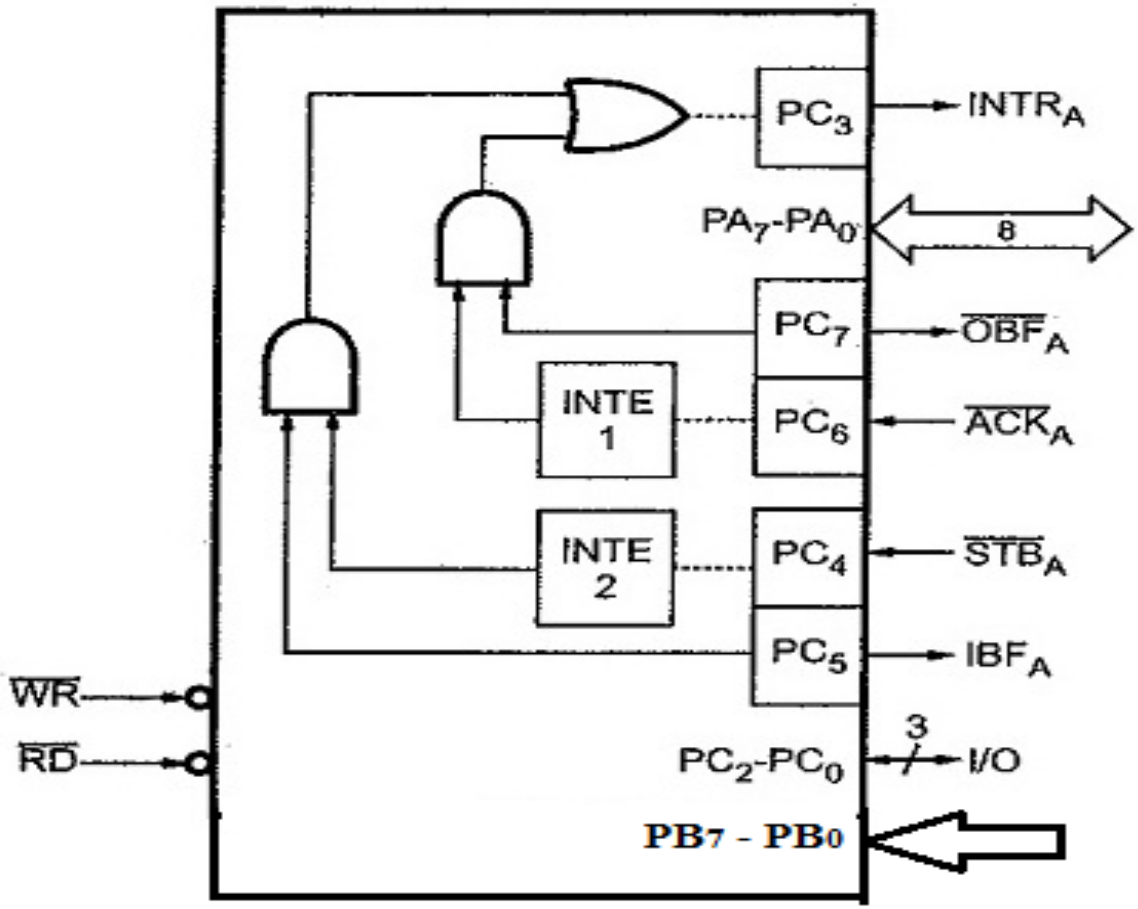


Fig. 15 8255 confiugred in Mode2, PortB mode0

Programming in Mode2(Strobe bidirectional)

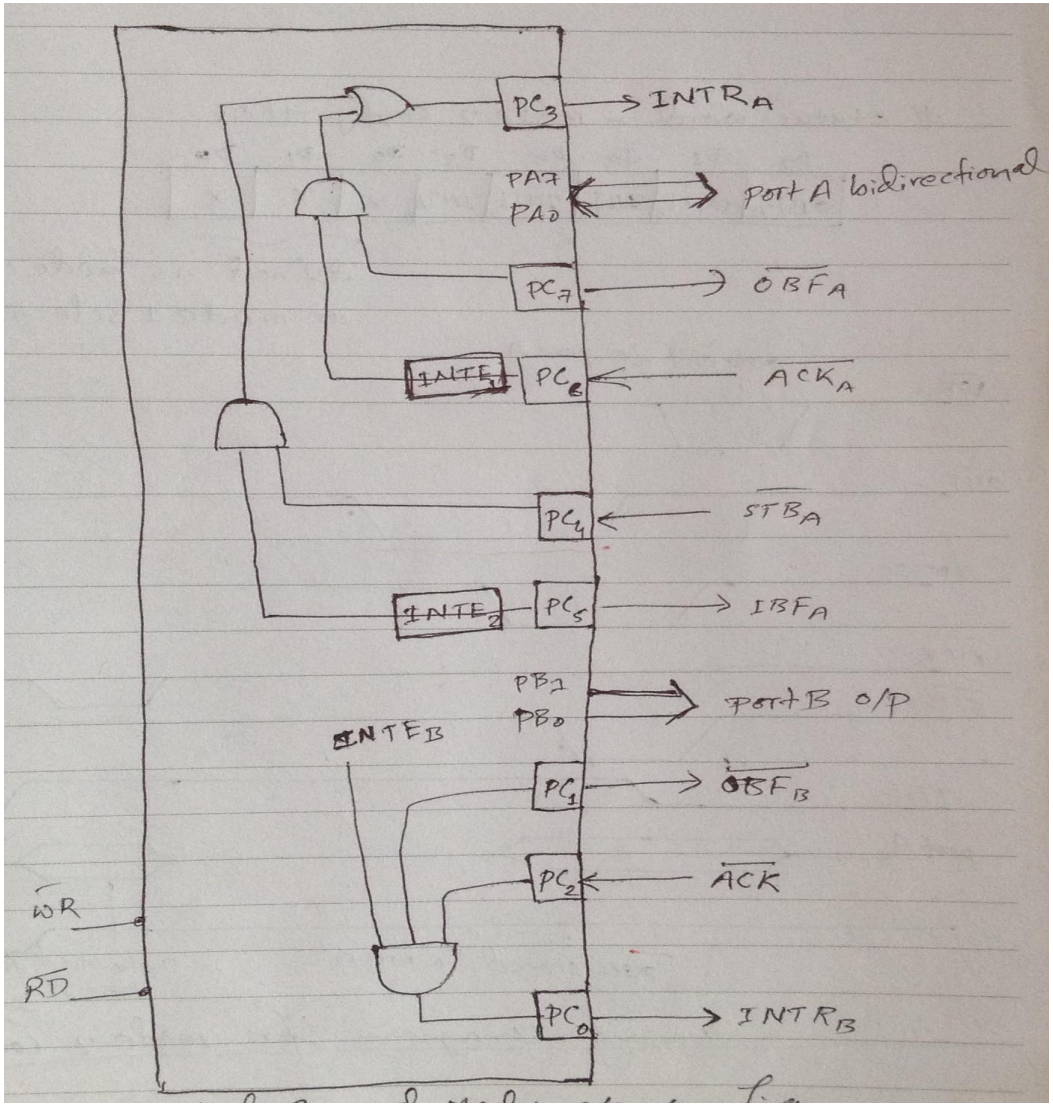


Fig. 16 8255 confiugred in Mode2; PortB mode1, O/P

Control and status words for Mode2

Control word for Mode 2; PortB mode0/ input

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

Control Word for Mode2 and PortB mode1/ output

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

Status Word for Mode2 and PortB mode1, In/Out

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
$\overline{OBF_A}$	$INTE_{A1}$	IBF_A	$INTE_{A2}$	$INTR_A$	b	b	b

Programming in Mode2(Strobe bidirectional)

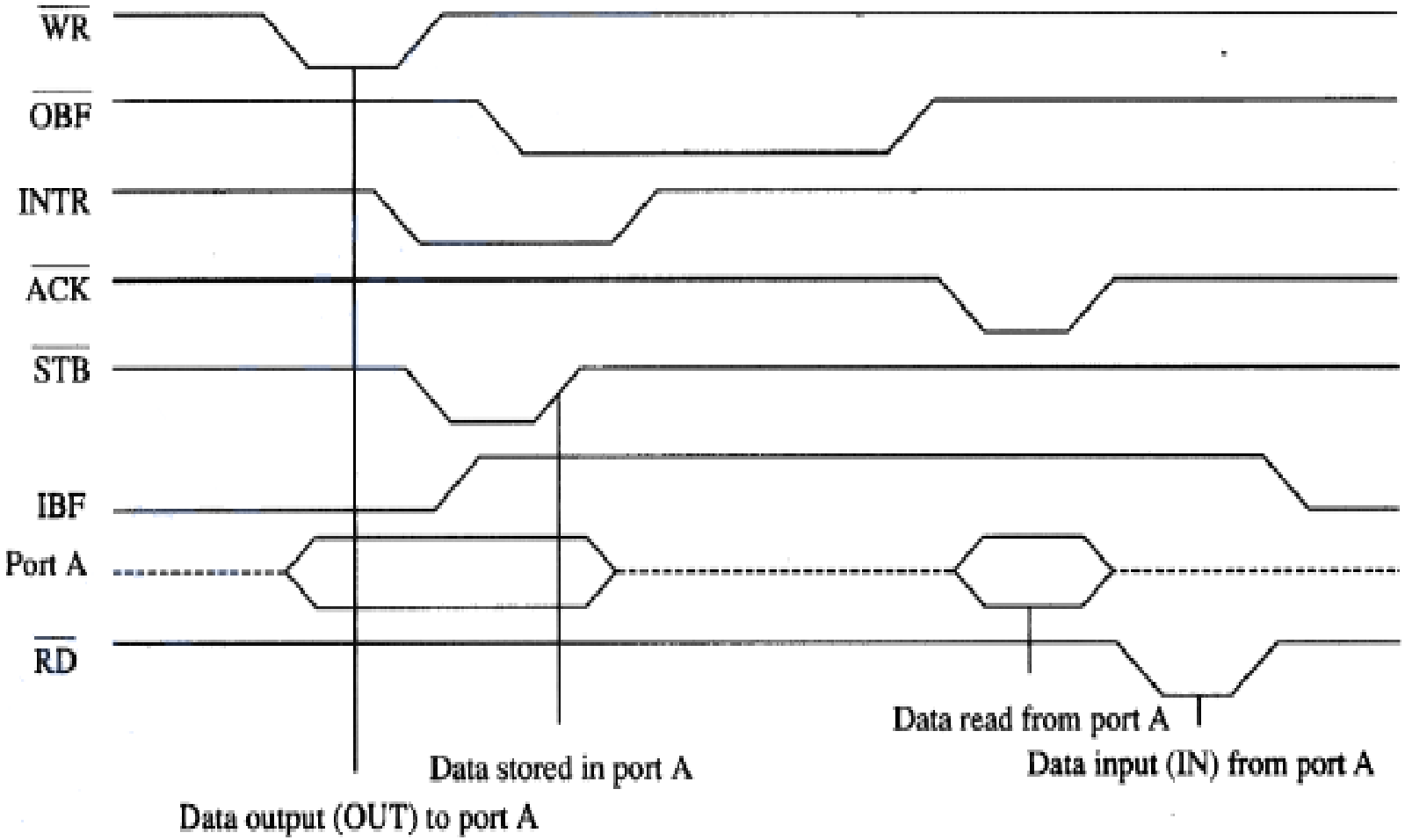


Fig. 17 Timing Diagram for Mode 2 Configuration

Programming 8255 in Mode 2

Output Control Signals

$\overline{OBF}$ (Output Buffer Full)

- is active low output which indicates CPU has write data into portA.

$\overline{ACK}$ (Acknowledgement)

- is an active low input from peripherals
- it enables the tri-state output buffer or PortA making PortA data available to the peripherals.

INTE1

- the internal flip-flop associated to output buffer full.
- used to enable or disable the interrupt by setting PC_6 in BSR mode.

Programming 8255 in Mode 2

Input Control Signals

$\overline{STB}$(Strobe Input)

- is an active low input signal which enable PortA to latch available as its input.

IBF(Input Buffer Full)

- is an active high output which signifies the data has been loaded into the input latch of PortA

INTE2

- the internal flip-flop associated to input buffer full.
- used to enable or disable the interrupt by setting PC_4 in BSR mode.

Parallel Interfacing with ISA and PCI Bus

- I/O bus are used to connect the system bus
- ISA(8bit or 16bit), EISA(Extended ISA 32bit), PCI(32bit or 64bit), AGP(Accelerated graphics port), PCI-X(PCI extended 64bit,133 MHz), PCI-Express

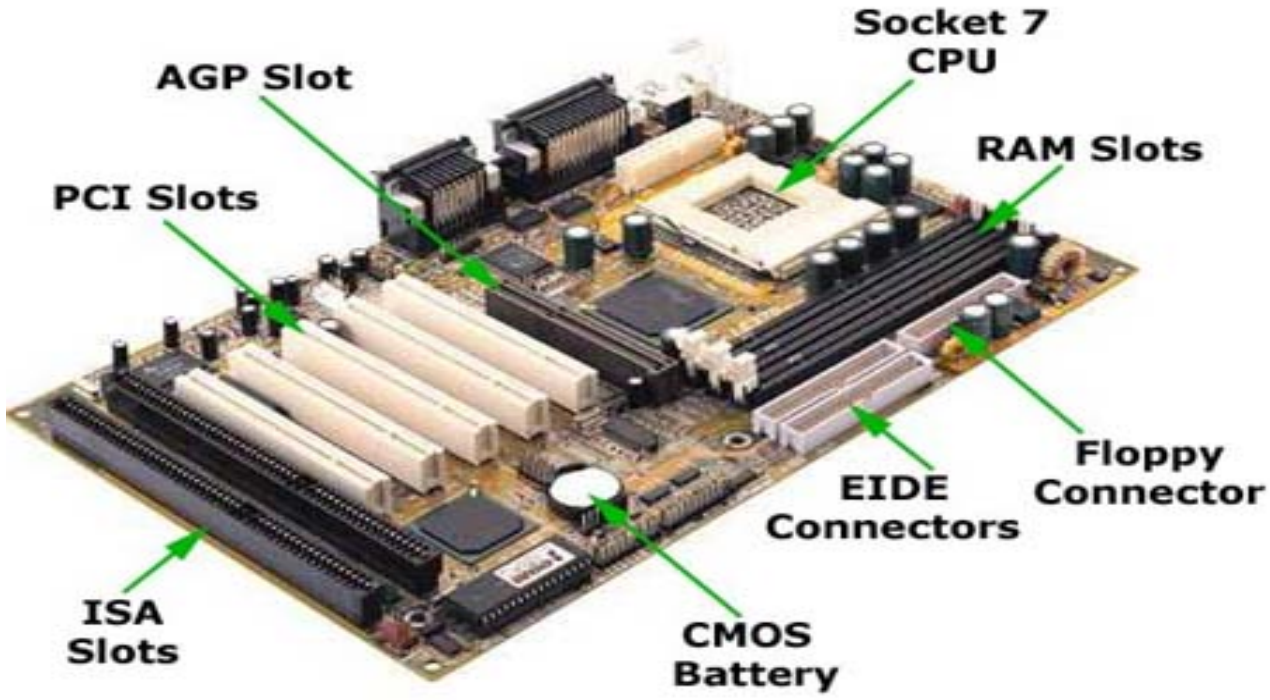


Fig. 18 Different I/O Bus slots

Industry standard Architecture (ISA) bus

ISA Bus (Industry Standard Architecture)

- First introduced by IBM/PC in 1982 used with 8088/8bit microprocessor.
- Originally, ISA bus was 8bit bus running at 4.77 MHz
- 16bit version was introduced in 1984 used with Intl 80286/16bit microprocessor.
- peripherals such as sound cards, disk drives, network cards are connected via ISA slots.
- ISA is almost absolute these day PCs, but it is still used in may industrial applications for their low cost.

Industry standard Architecture (ISA) bus

8bit ISA Bus Architecture

- has 8bit data bus and address bus of 20bit
- ISA slot/card has 62 pins
- Clock frequency of 4.77MHz
- ISA bus connector contains:
  - 20bit address bus
  - 8bit data bus
  - $\overline{MEMR}$, $\overline{MEMW}$, $\overline{IOR}$, $\overline{IOW}$ control signals to control I/O or memory on the ISA card
  - Interrupt request line $IRQ_2 - IRQ_7$
  - DMA request inputs $DRQ_1 - DRQ_3$
  - DMA acknowledgement O/Ps $\overline{DACK_0} - \overline{DACK_3}$
  - Clock Signals
  - Power lines and Reset.

Industry standard Architecture (ISA) bus

16bit ISA Bus Architecture

- has 16bit data bus and address bus of 24bit
- ISA slot/card has 98 pins
- Clock frequency of 8.33MHz
- Consists extra 36 pins behind 8bit Connectors
- Compatible with both 8bit and 16bit ISA cards.
- 16bit card consists of two edge connectors
  - one plugs into the original 8bit connector
  - other plugs into the new 16bit connector.
- extra connectors consists of:
  - 4 additional address lines - 24 lines in total
  - 8 additional data lines - 16 lines in total
  - 4bit DMA channel request and acknowledgement lines
  - additional interrupt lines
  - control lines to select 8bit or 16bit transfer.

Industry standard Architecture (ISA) bus

Reason for elimination of ISA Bus

- ISA bus is slow, hard to use and bulky.
- once a ISA uses dedicated interrupt line, only limited number of cards can be used
- For limited number of address bits (24bit address line), small sized (16MB for 24bit) RAM can be accessed for DMA.
- limited data bit line, resulting slower performance for higher bit data.
- ISA cant be automatically configured by BIOS or operating system as USB devices.
- ISA cards must be controlled manually by setting I/O addresses, interrupts and clock speed using jumpers and switches on the card itself.

Peripheral Component Interconnect (PCI) Bus

- Introduced in 1990 by Intel.
- Provides direct access to the CPU and system memory but uses bridge to connect the system bus to eliminate the potential for interference with CPU.
- PCI bus is independent of processor type or speed
- Originally operated at 33MHz using 32bit data lines
- Revised standard at 66MHz using 64bit data lines.
- 32bit PCI connector has 124 pins and 64bit has 188 pins.
- PCI bus is able to work with few pins because of hardware multiplexing.
- PCI supports devices that uses 5v signaling voltage levels
- PCI card supports plug and play features; PCI devices are automatically configured by BIOS or operating system.

Peripheral Component Interconnect (PCI) Bus

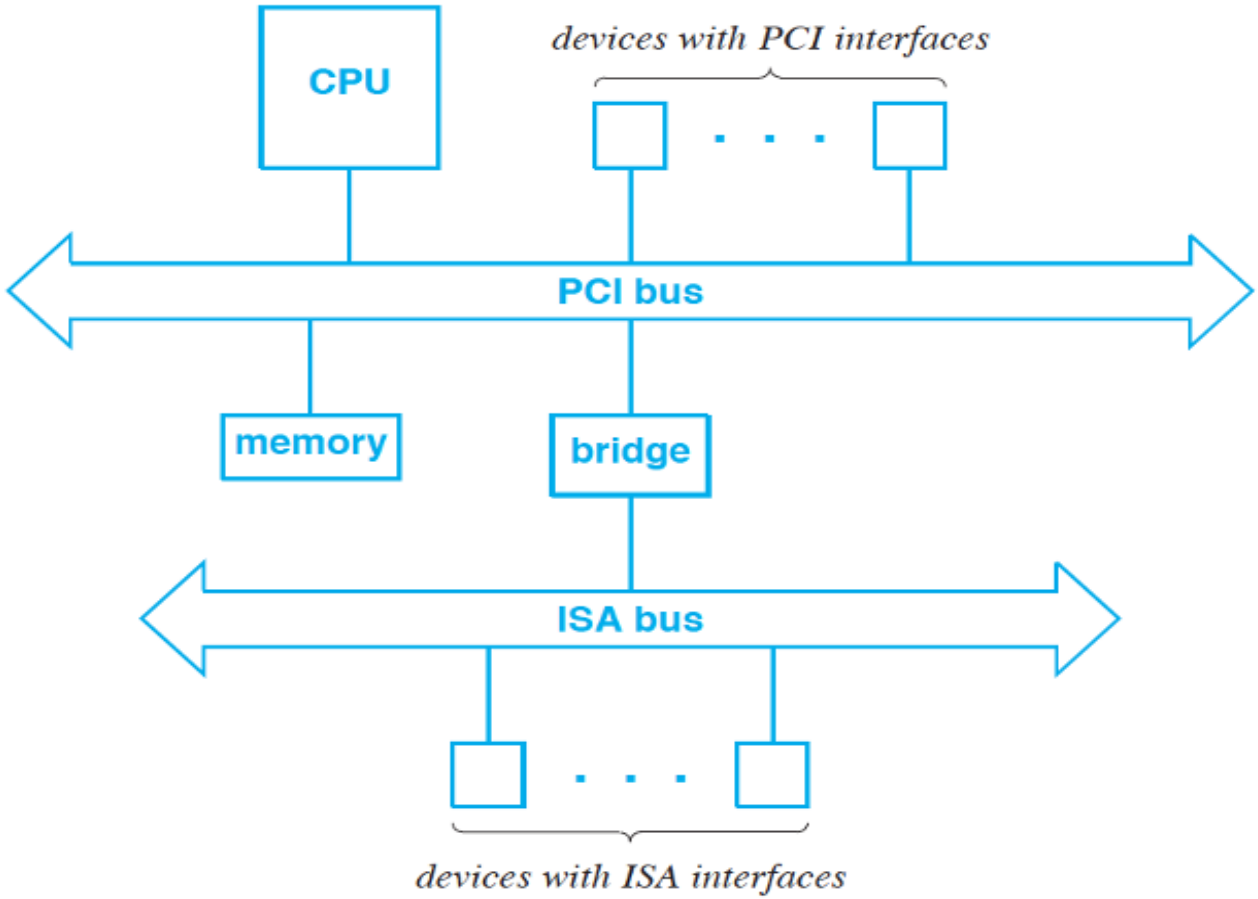


Fig. 19 PC architecture with PCI and ISA bus using bridge

- MPU/PC connects to PCI bus through IC called PCI bridge.
- virtually any processor can interface to PCI with bridge.

Peripheral Component Interconnect (PCI) Bus

Advancement in PCI bus

- PCI-X(PCI Extended) runs at 133MHz, 32bt and 1.06GBps data rate
- PCI-E (PCI-Express) replaced PCI, PCI-X and AGP standards.

Peripheral Component Interconnect (PCI) Bus

Example#1:

Assume that your group has decided to make a PC based controls system for a wine company. After studying the system, your group found out that the following to be implemented for controlling:

- Pressure measurement (6points)
- Temperature measurement (5points)
- Weight measurement (1point)
- Volume measurement for filling (5points)

8255A PPI card at base address 0550H is to used:

- Collect documents and components
- List out different Signals connected to interface
- Draw minimum mapping circuit for the control system
- what are the address captured by card
- generate necessary control words
- write subroutine for measuring pressure of all points and control if the pressure is not in the range, assume suitable data if necessary;

As you go
Assignment

Assignment Module#2 is available at MS-Team.

Deadline for submission: June 10, 2022 (*Before Midnight*)