Instrumentation (II/II)

Course Code: ENEX - 252
(Module#4)

Dhawa Sang Dong, MSc Eng.
(Sr. Lecturer)

Kathmandu Engineering College
Kalimati, Kathmandu

June 2025

Chapter#4

Interfacing of Instrumentation System

✓ Class Outline

1. Microprocessor and microcontroller
2. General Purpose Programmable I/O device 8255
3. Microcontroller – ATMEGA328 and STEM32

Microprocessor and microcontroller

Microprocessor (MPU):

- A general-purpose central processing unit (CPU) on a single chip (Integrated Chip – IC).
- Needs external components like RAM, ROM, I/O ports, and timers to function.

  Examples: Intel 8085, Intel 8086, Intel Core i7, AMD Ryzen.

Features:

- Focused on computation and processing power
- High speed, multi-tasking support
- Used in PCs, laptops, and servers

Microprocessor and microcontroller

Microcontroller (MCU):

- A compact integrated chip with CPU, RAM, ROM, I/O ports, and timers all in one.
- Designed for specific control applications.

Examples: 8051, PIC16F877A, ATmega328 (Arduino), STM32

Features:

- Optimized for real-time control
- Low power, small size
- Used in embedded systems like appliances, cars, medical devices

Microprocessor and microcontroller

Architecture Comparison

Feature	Microprocessor	Microcontroller
CPU	Present	Present
Memory	External (RAM/ROM)	On-chip RAM and ROM
I/O Ports	External	On-chip I/O ports
Cost and Power	High cost, high power	Low cost, low power
Application	General-purpose	Specific task control
Example Use Case	Desktop, Laptop	Washing Machine, Microwave Oven

Microprocessor and microcontroller

Selection Criteria for Choosing MPU/MCU

1. **Application Type**
   - For general computing → Microprocessor
   - For real-time control systems → Microcontroller

2. **Performance Requirements**
   - Need high speed & multitasking? → Microprocessor
   - Need efficient, real-time control? → Microcontroller

3. **Power Consumption**
   - Battery-powered systems → Low-power MCUs
   - Desktop applications → Power-hungry MPUs acceptable

Microprocessor and microcontroller

Selection Criteria for Choosing MPU/MCU

❹ **Cost Constraints**

- MCU is cheaper and better for mass production
- MPUs are expensive, suitable for high-end devices

❺ **System Complexity**

- Simple system (like a digital thermometer) → Microcontroller
- Complex OS-based system → Microprocessor

❻ **Integration Needs**

- More integration (sensor interface, PWM, ADC) → Microcontroller
- More flexibility and expansion → Microprocessor

Microprocessor and microcontroller

Microprocessor Applications

- Personal computers
- Gaming consoles
- Workstations
- High-speed scientific calculators
- ATM machines

Microprocessor and microcontroller

Microcontroller Applications:

- Automatic washing machines
- Smart home devices (e.g., smart bulbs, thermostats)
- Medical instruments (heart rate monitor)
- Robotics (Arduino/STM32-based robots)
- Automotive systems (ABS, airbag control)
- IoT devices

General Purpose Programmable I/O device Programmable Peripheral Interface 8255(PPI)

- it is general purpose programmable I/O Interfacing device communicable to Intel microprocessor (initially).
- it has 24 Input/Output 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.

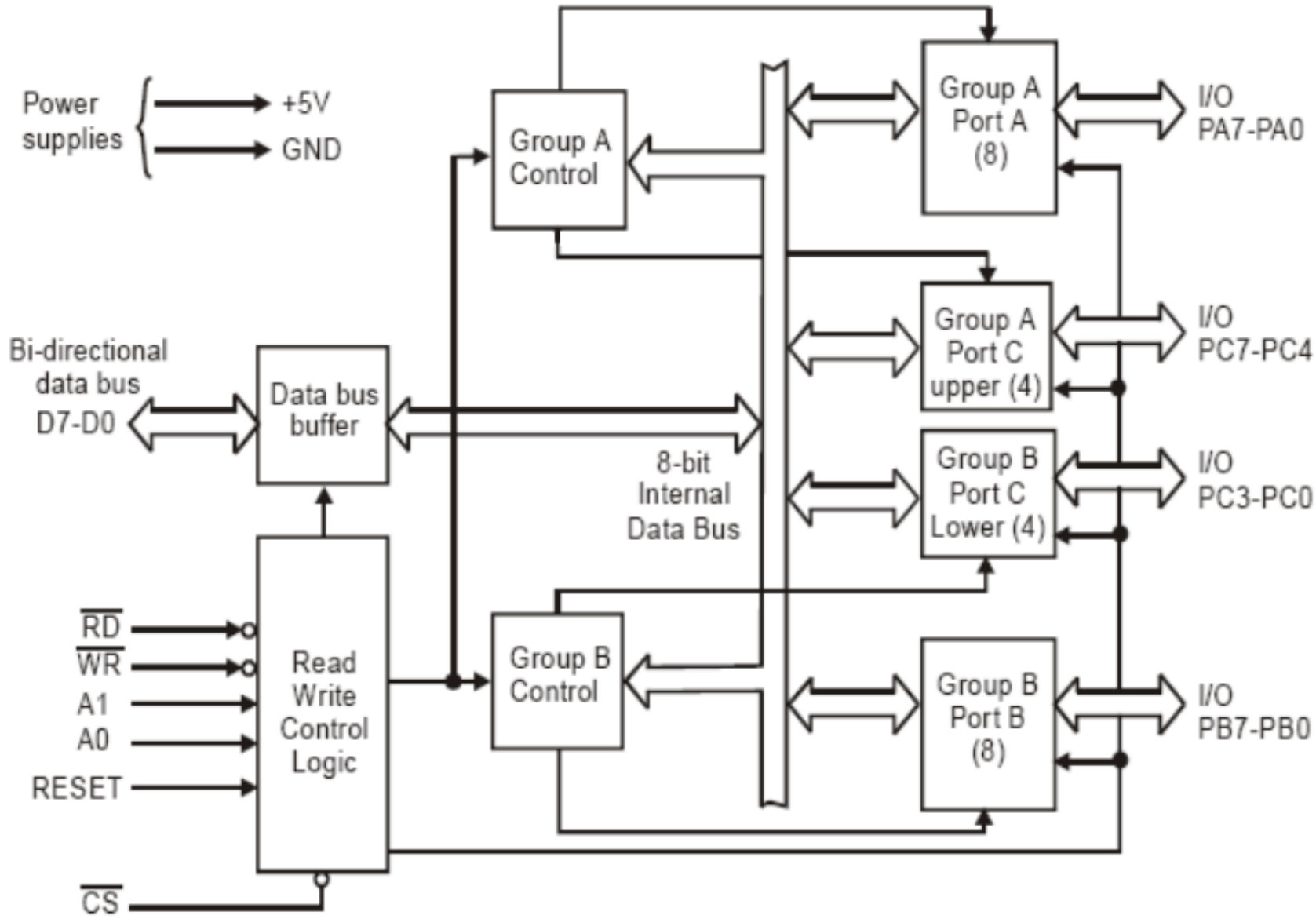
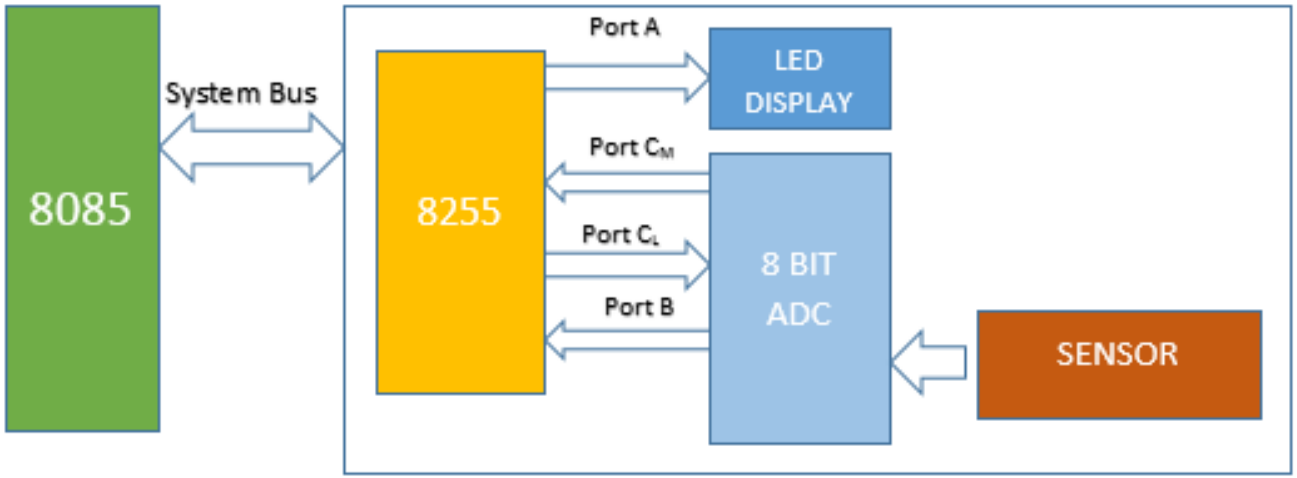


Fig. 1 8255 Internal Block Diagram

Interfacing block diagram



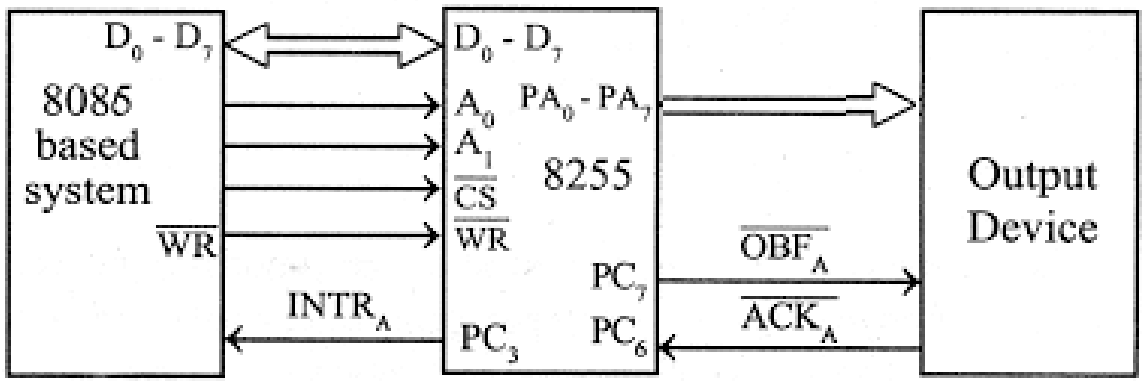


Fig. 2 Simple 8255 interfacing to I/O devices block diagram

Some main block components of 8255

Data Bus Buffer

- tri-state 8-bit buffer 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.

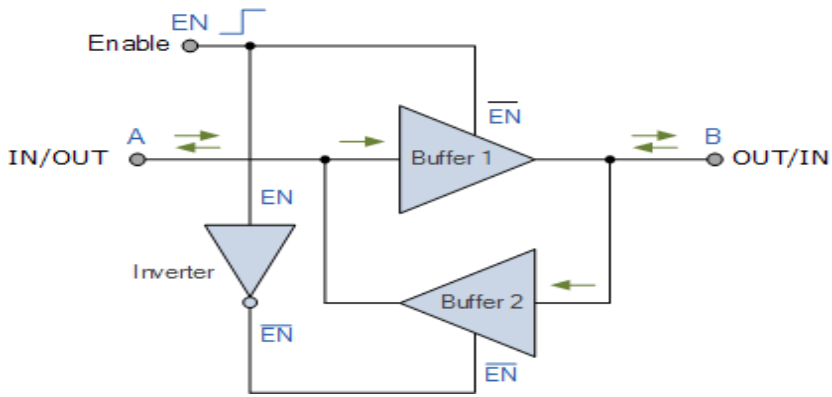


Fig. 3 tri-state bi-directional buffer for single bus

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 mod~~

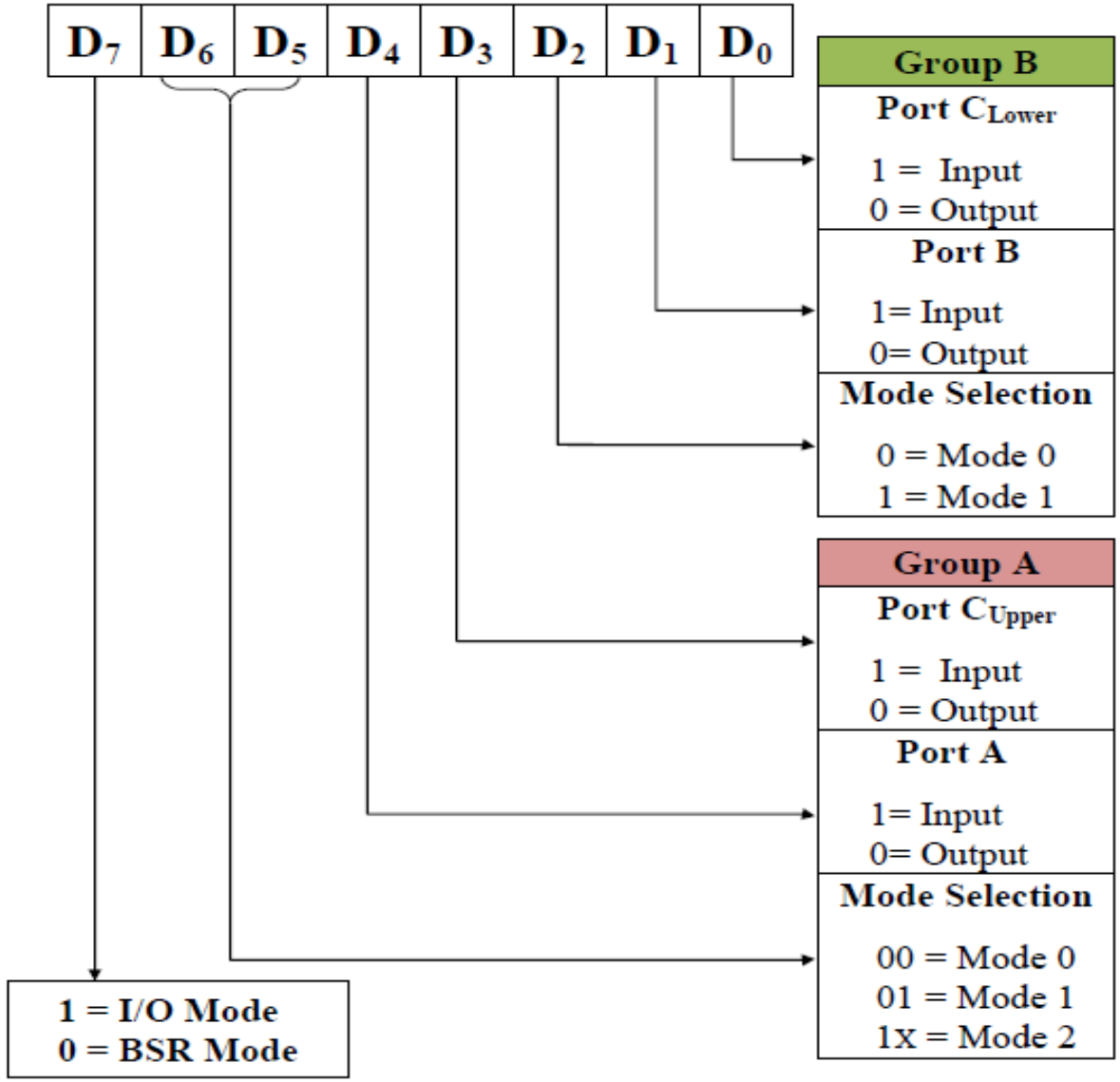
✓ if $D_7 = 1$, D_6-D_0 determines the I/O functions in various modes (Mode0, Mode1 or Mode2).

✓ 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 (PortC-upper and PortC-lower) 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 (PortA and PortB).

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, and GroupB either mode0 or mode1

✔ PortA, 8-bit bidirectional bus and 5-bit control from PortC.

✔ both input and output data are latched.

✔ 5-bit from PortC is used for control and status of 8-bit bidirectional data portA, remaining PortC bit/pin can be used for PortB control signal.

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/control.

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)

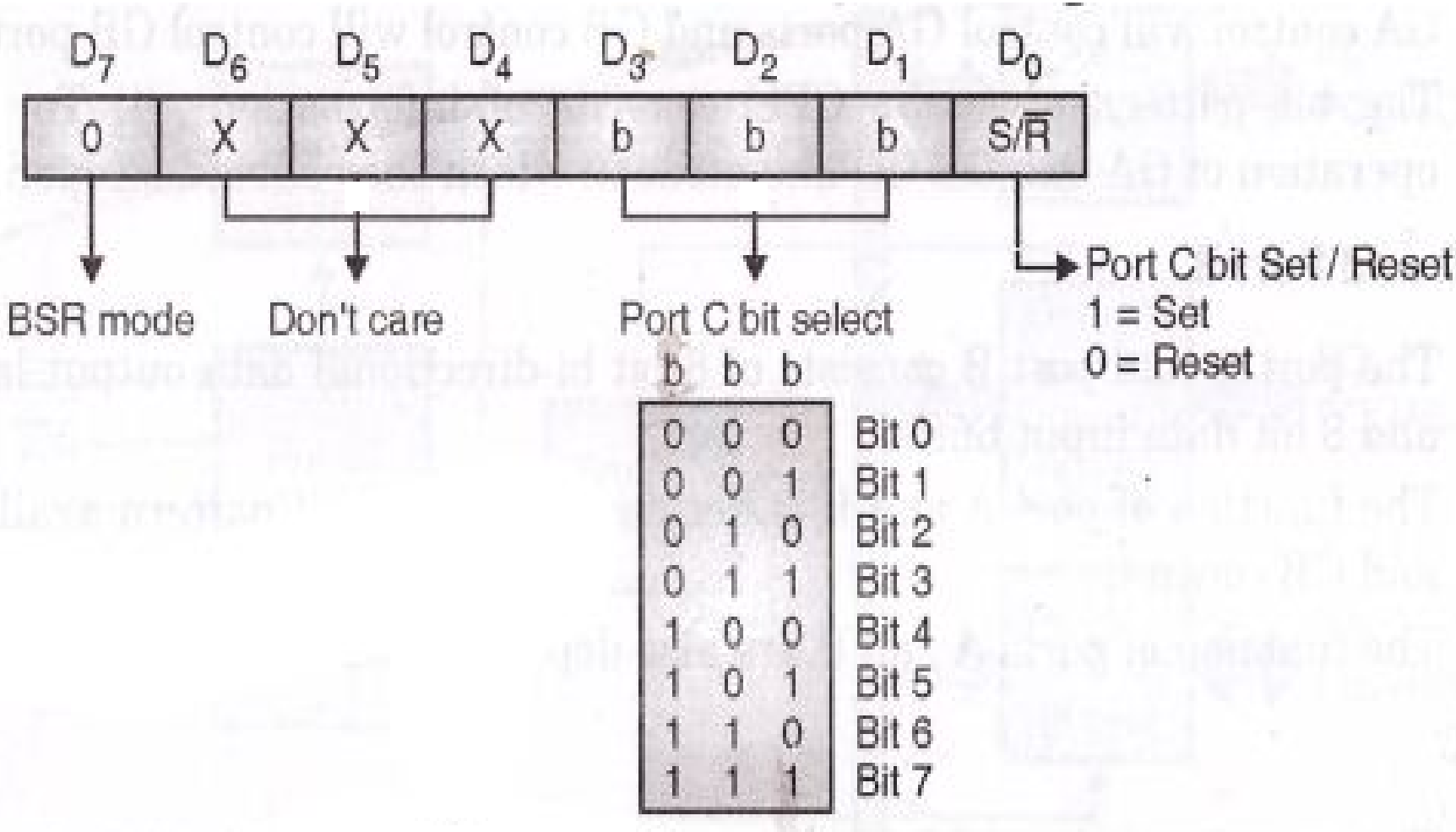


Fig. 4 Control word Format BSR mode

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_4

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

3. To set PC_2

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

Programming and Operation of 8255

To Sum up, Control words can be:

- ✓ either mode definition word or
- ✓ Bit Set/Reset word.

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.

Programming and Operation of 8255

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

I/O Mode or BSR Mode | Mode Definition

- D_7 of control word defines the control word to be either I/O Mode definition or BSR Mode.
- 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 configured 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 functioning 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

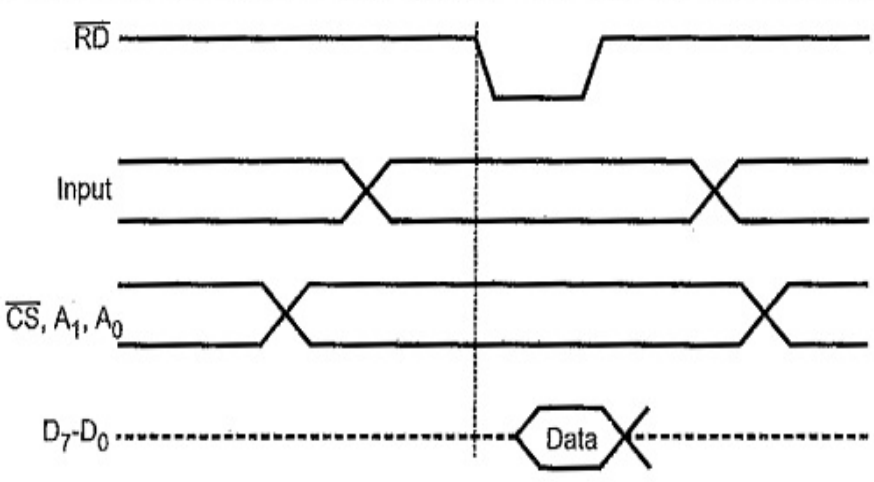


Fig. 5 Timing Mode0/Input

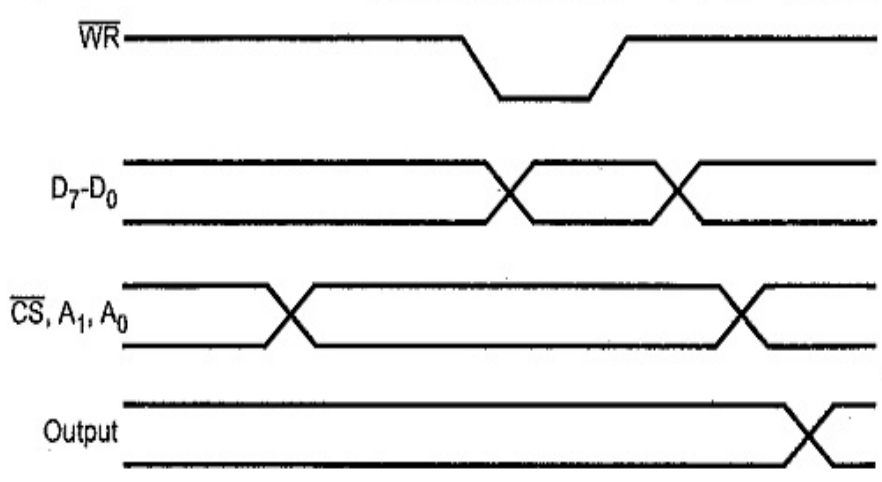
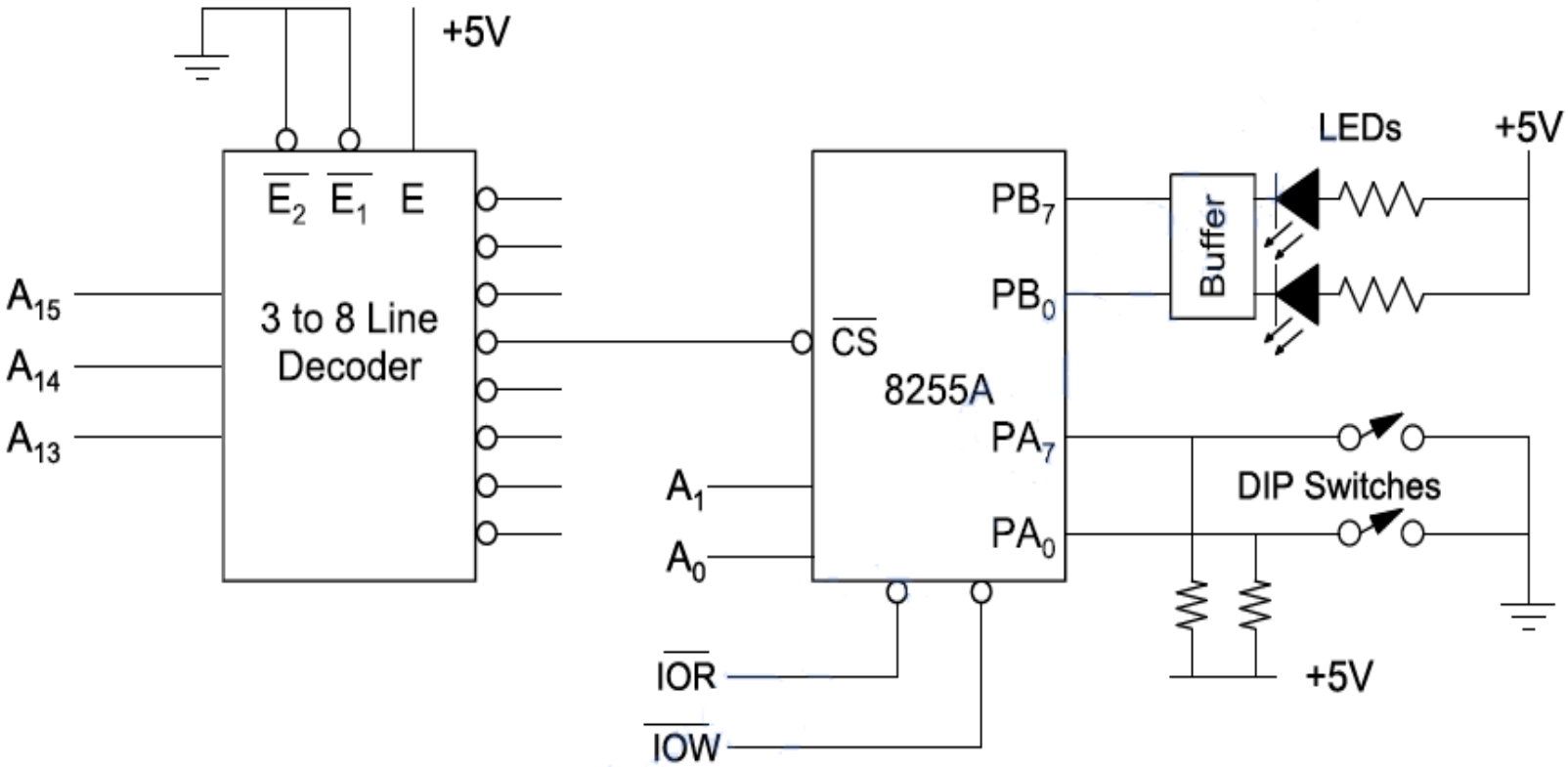


Fig. 6 Timing Mode0/Output

Mode 0 Example

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



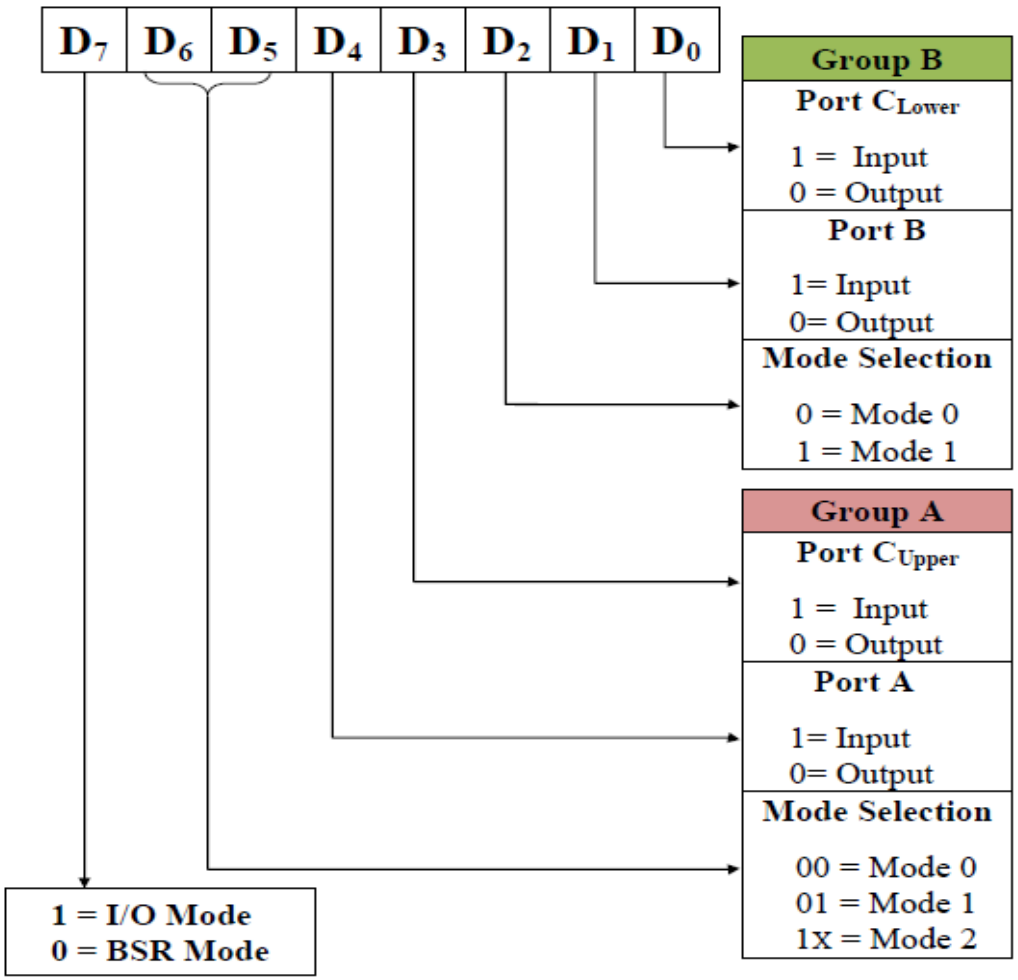
$\overline{CS}$ to be low:

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

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.

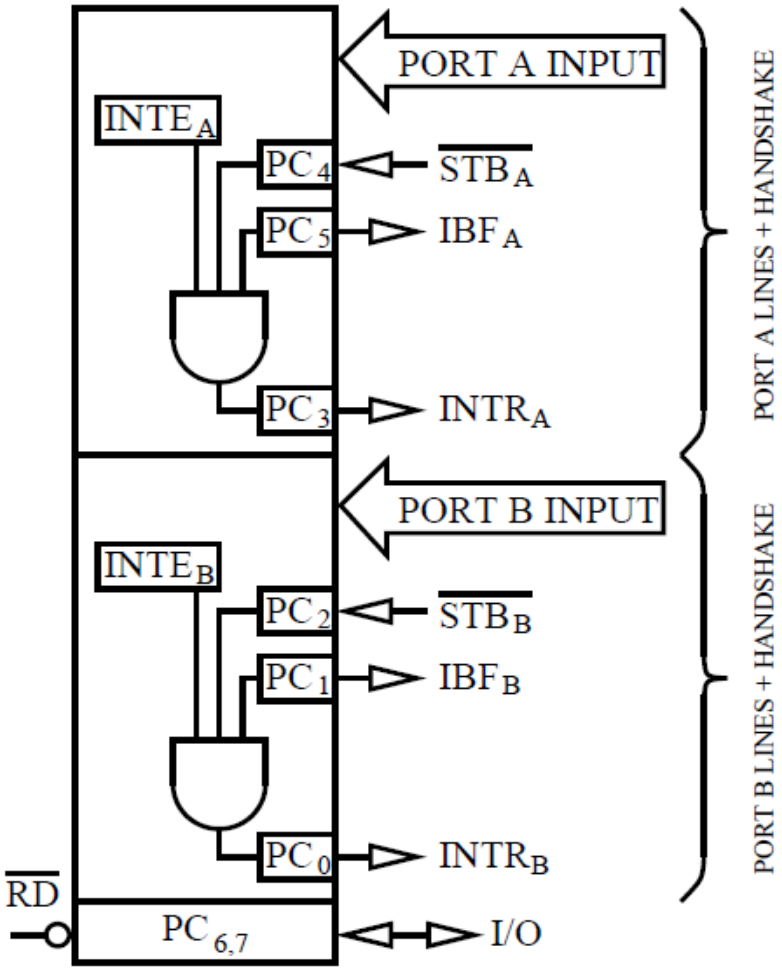
PortB

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 ⇒ 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.

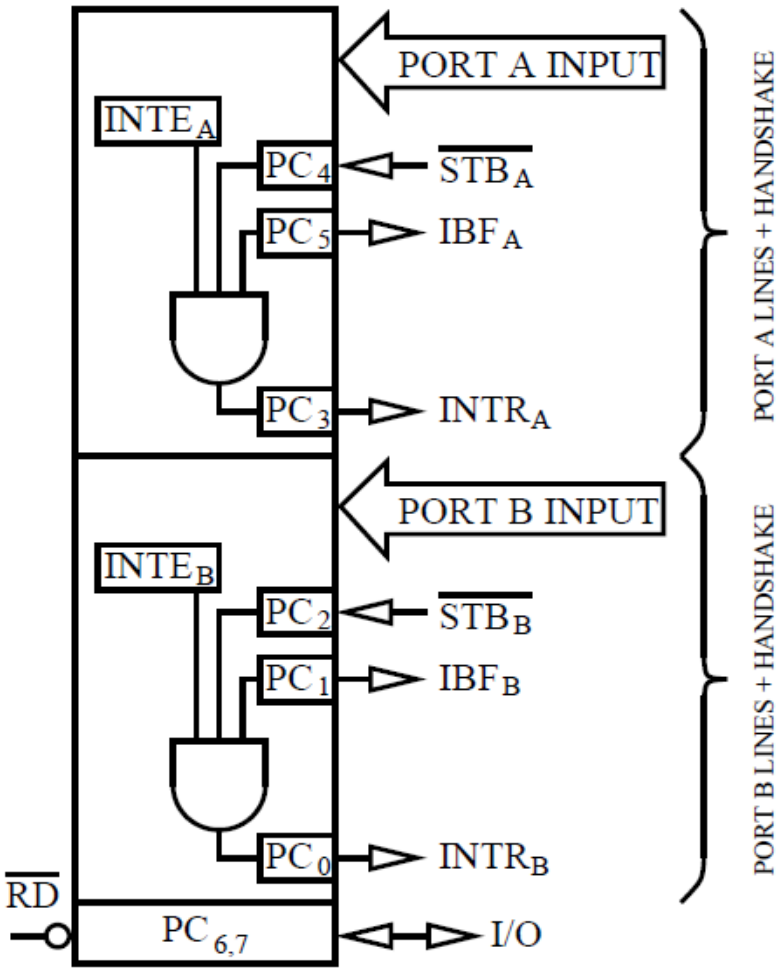
.....

Fig. 7 Mode1 input configuration

Programming and Operation of 8255

Programming in Mode 1(PortA and PortB ⇒ Input)

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



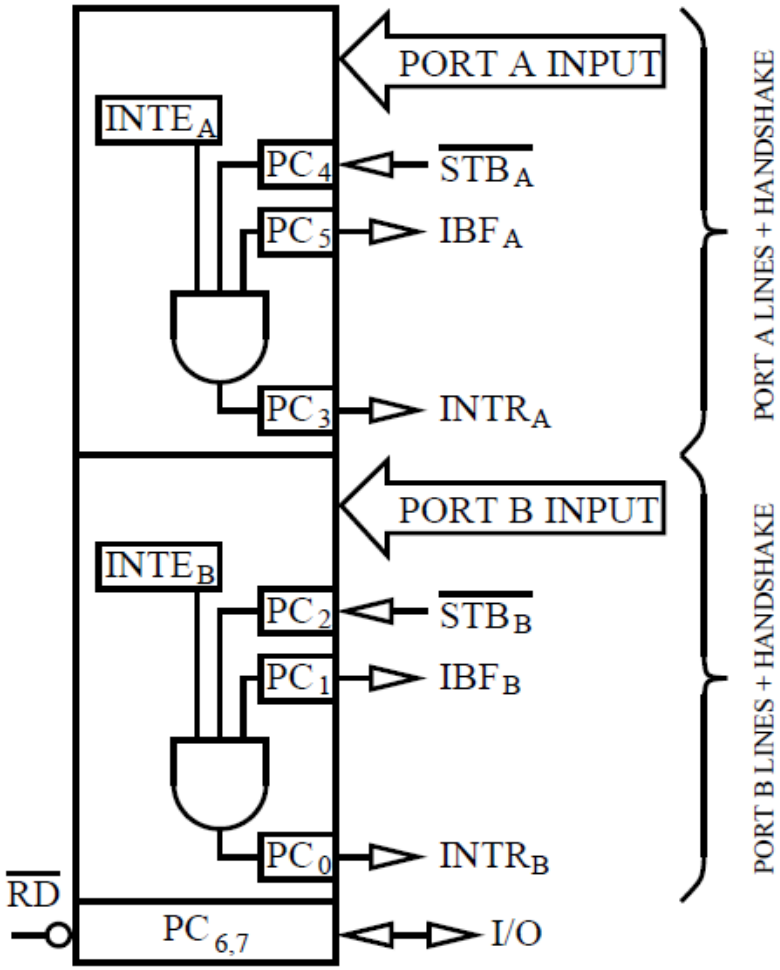
Mode1 input configuration

.....

- ✔ $INTR_A$ 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

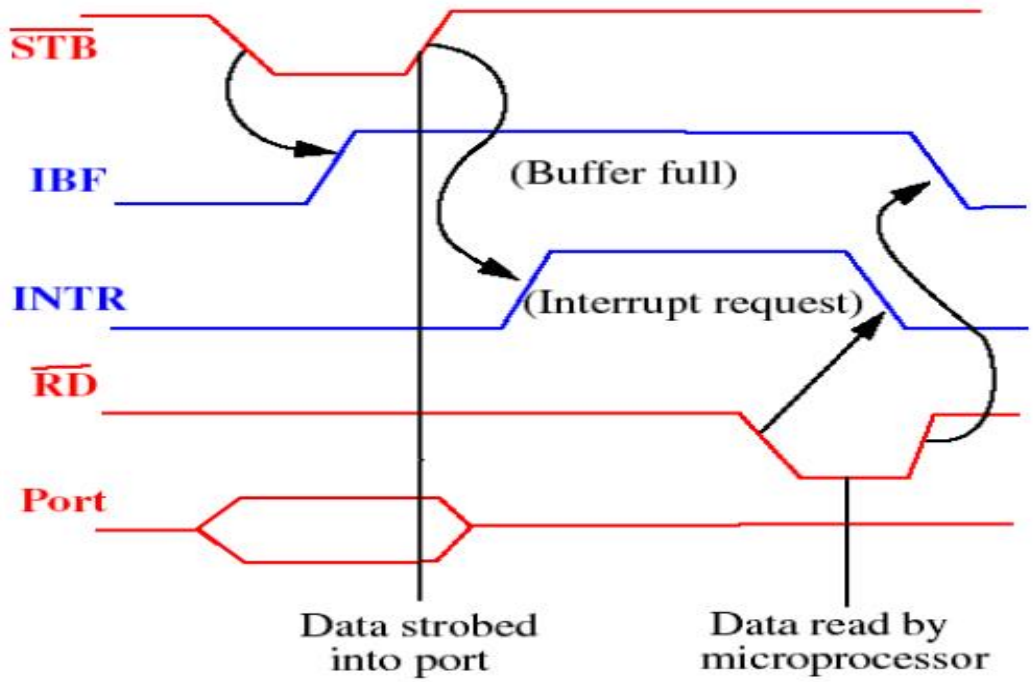
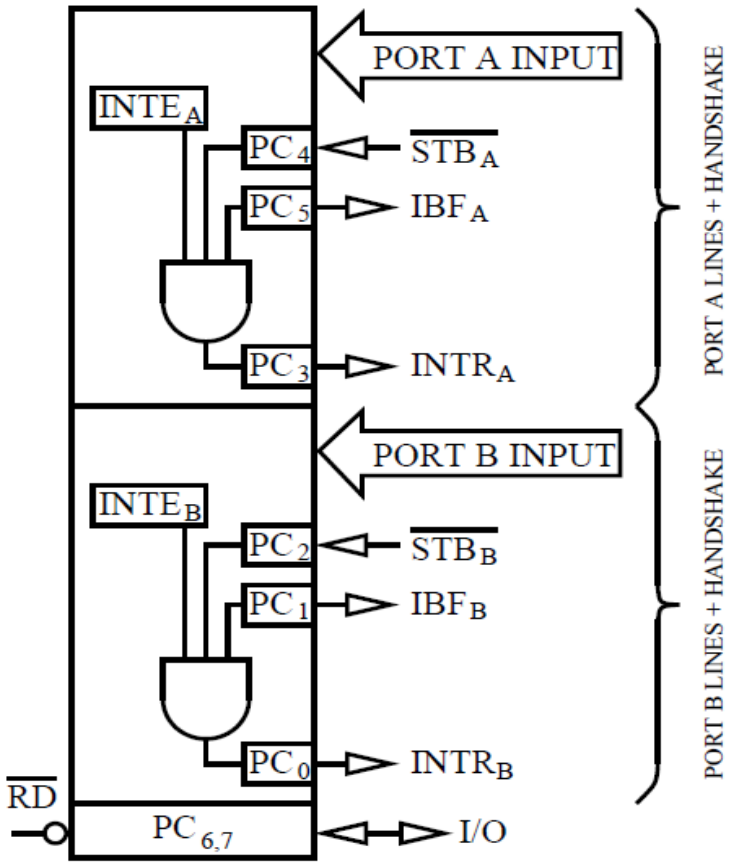


Fig. 8 Timing Diagram for Mode1/Input (Strobed Input)

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 ⇒ 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 (from 8255) to interrupt CPU.
- generated when $\overline{STB}$, IBF and INTE are all logic one.
- it is reset when $\overline{RD}$ is rising.

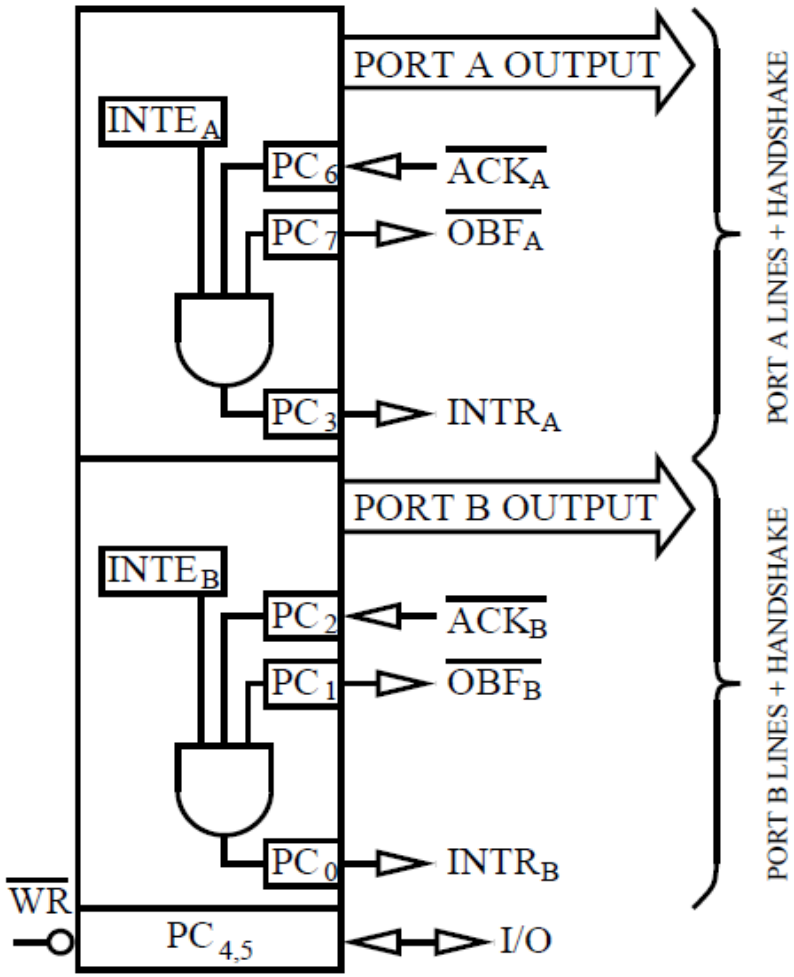
INTE(Interrupt Enable)

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



- ✔ 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/STB signal to I/O Peripherals) goes low on the rising edge of $\overline{WR}$ signal - once CPU writes data into 8255.

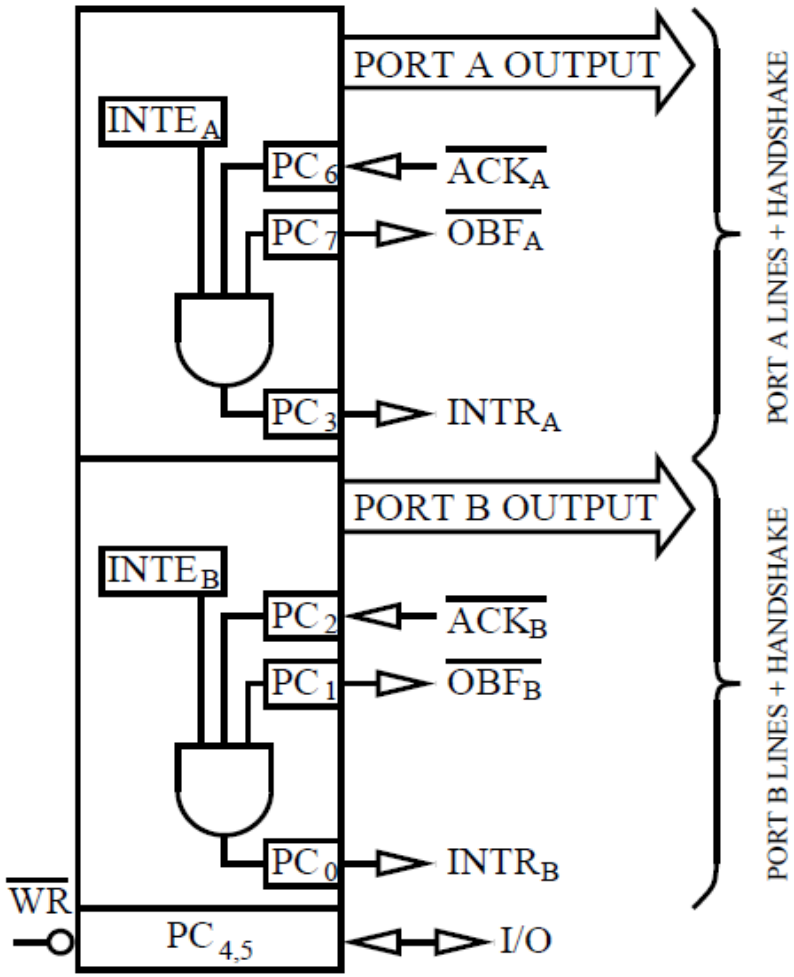
.....

Fig. 9 Mode1 output configuration

Programming and Operation of 8255

Programming in Mode 1(PortA and PortB ⇒ 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 output configuration

.....

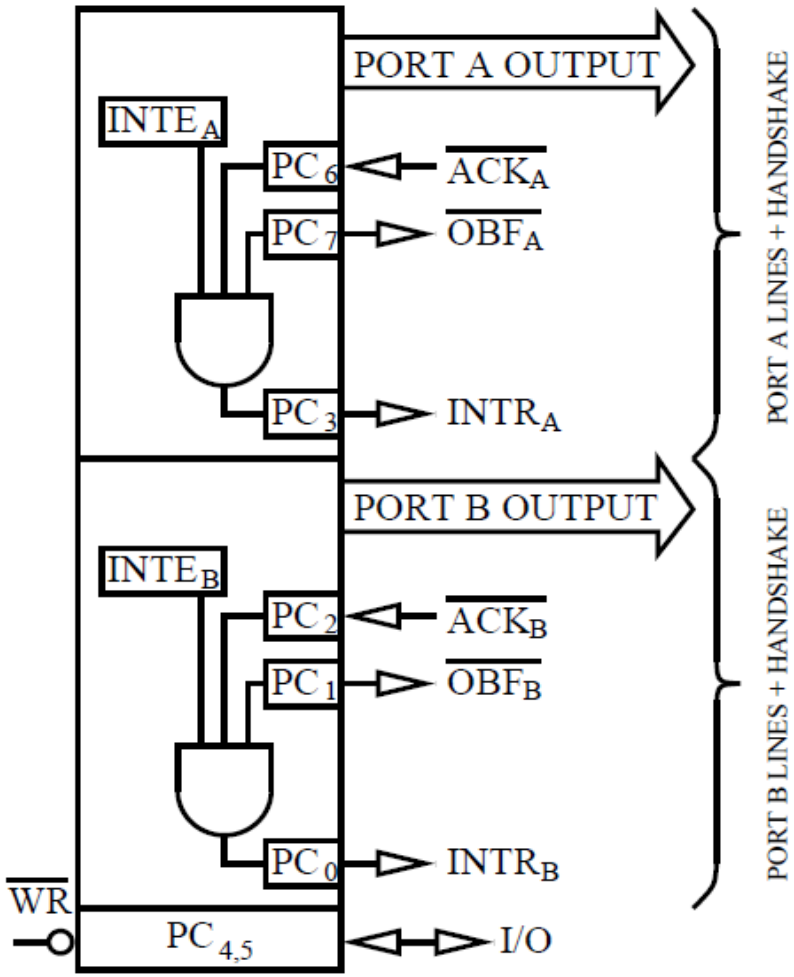
- ✔ $\overline{OBF_A}$ can be treated as strobe signal to the peripheral to latch the content of PortA.
- ✔ Once the data has been read, peripheral acknowledges by low $\overline{ACK}$.
- ✔ low $\overline{ACK}$ resets $\overline{OBF_A}$ indicating buffer is free and ready for next data.

...

Programming and Operation of 8255

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

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



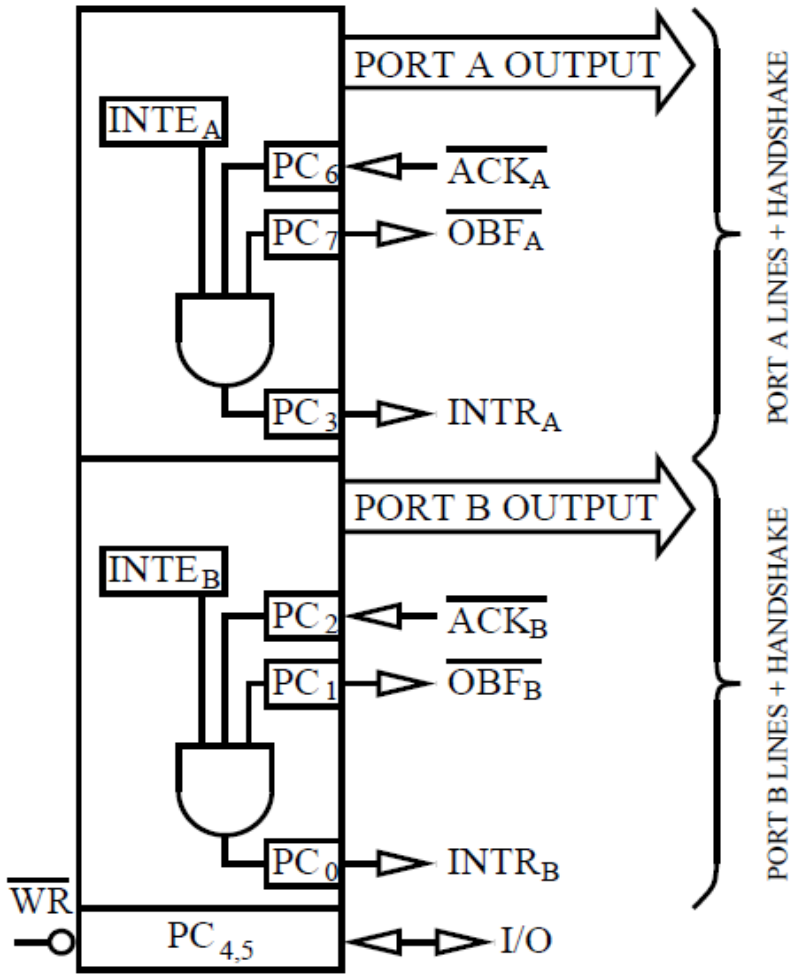
Mode1 output configuration

.....

- ✔ $INTR_A$ 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 $\overline{WR}$ 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/ output

Mode1/Output Timing Diagram



Mode1 output configuration

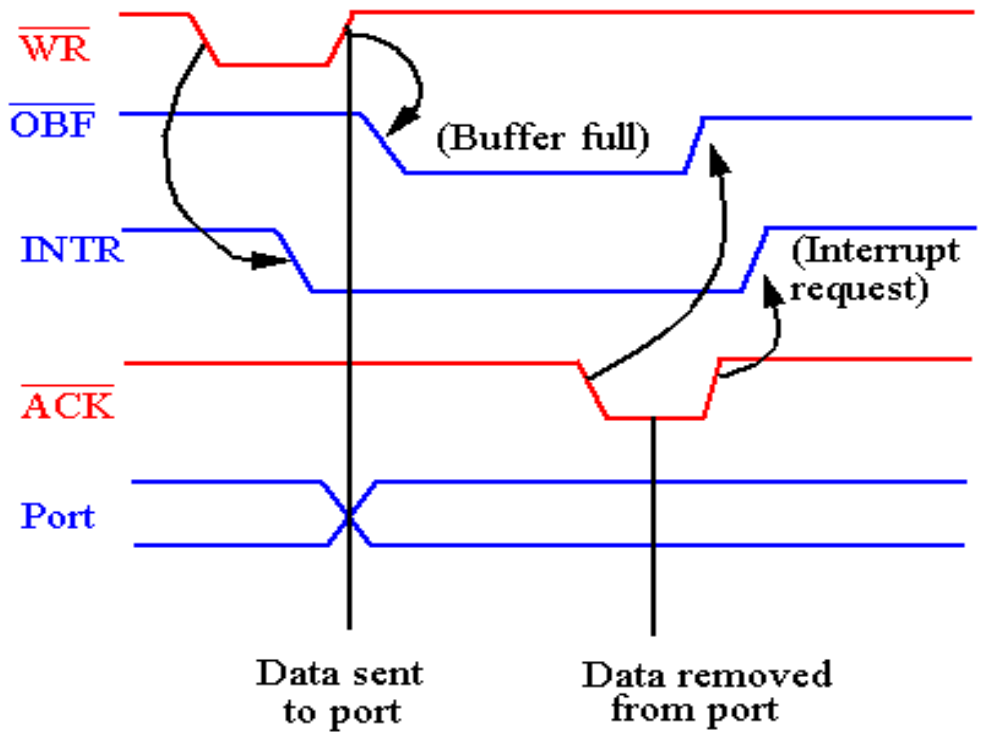


Fig. 10 Timing Diagram for Mode1/Output

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)

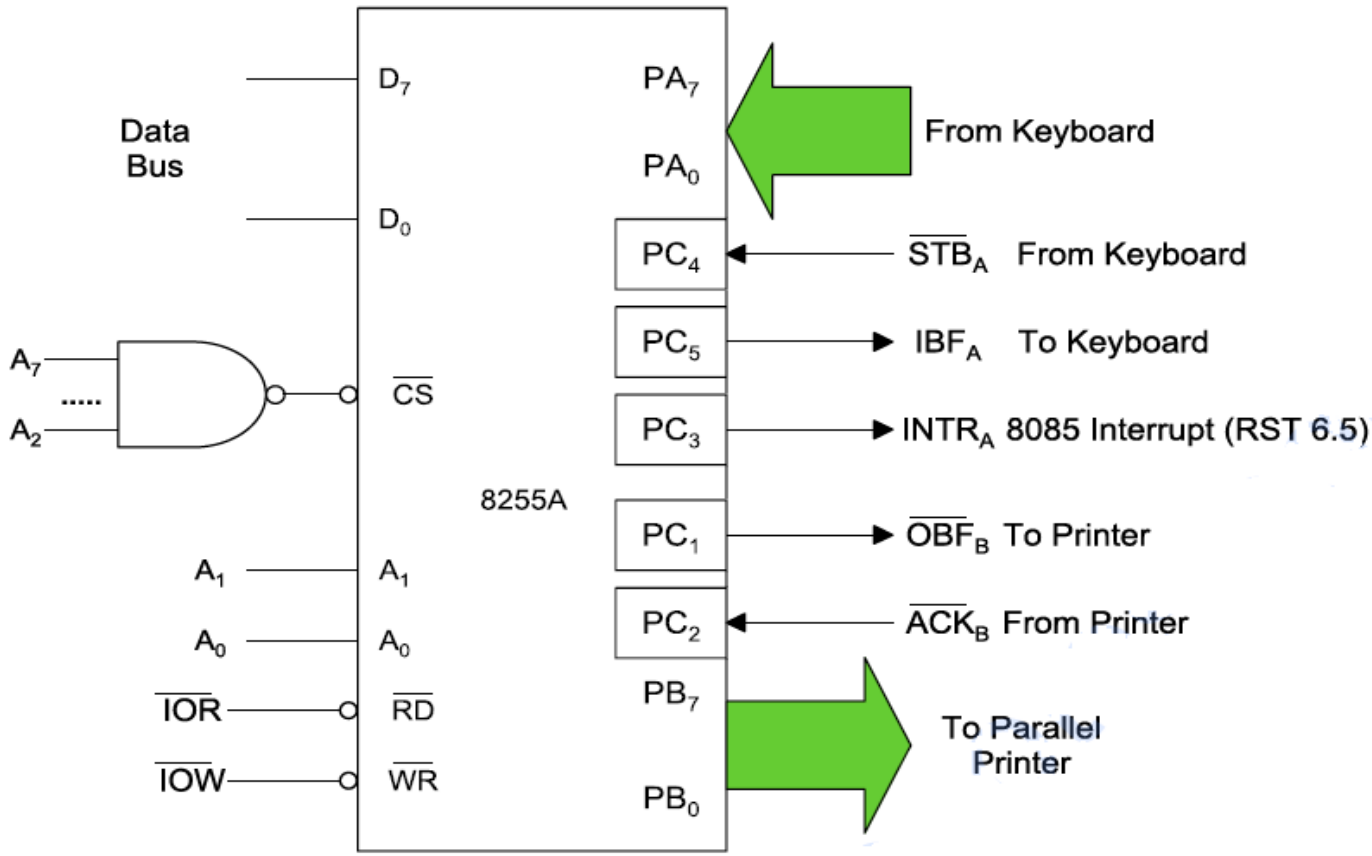
- high on this pin is used to interrupt the CPU signaling peripheral received the data.
- when an INTE, OBF and ACK all are one INTR is high and reset by falling edge of $\overline{WR}$

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

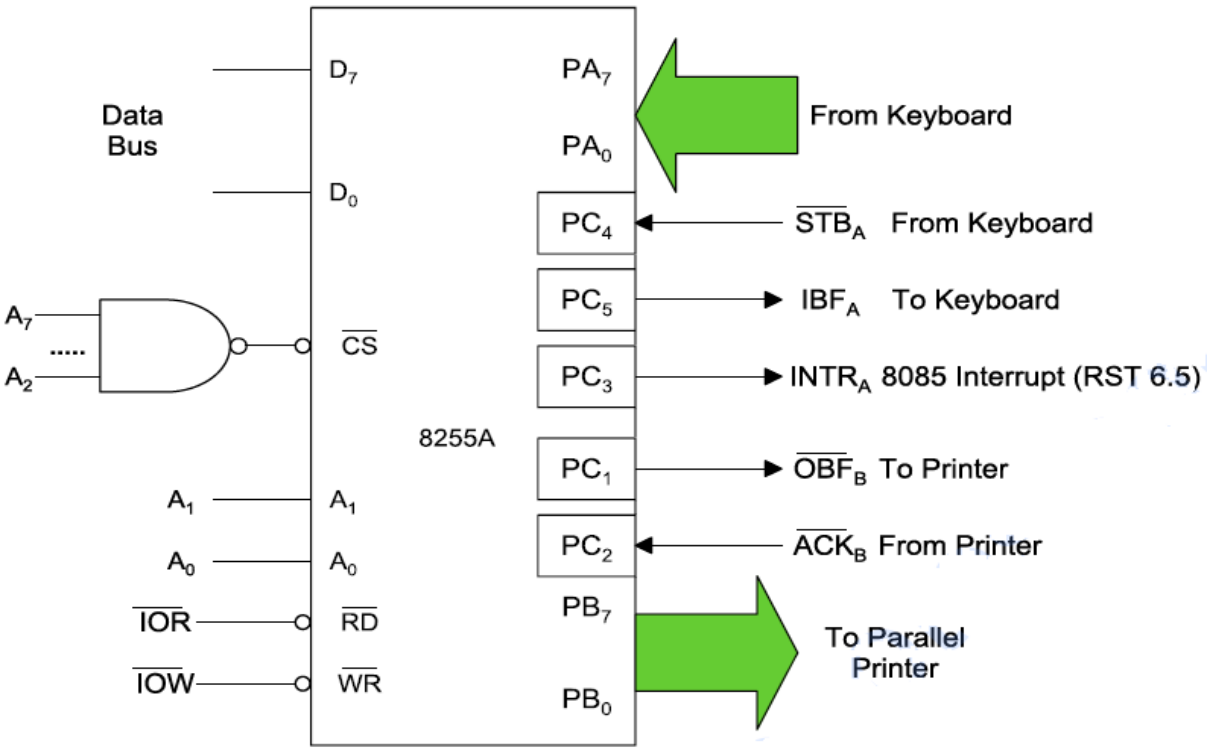
For given diagram; PortA accept keyboard input with interrupt I/O and portB output for printer with status check I/O.
Write a subroutine for the design scenario described.



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 8255 PPI example

Main Program:

```
MVI A, B4H          ; I/O control to ACC
OUT FF             ; ACC to Control Reg
MVI A, 09H          ; To set PC4 / INTE_A
OUT FF             ; BSR to Control Reg
EI                  ; Enable Interrupt

CALL READ           ; Read character
CALL PRINT          ; Print character
HLT                 ; Terminate
```

Programming 8255 PPI example

Subroutines:

READ:

```
IN FC              ; Read from Port A
MOV C, A             ; Save ACC content
RET
```

PRINT:

```
IN FE              ; Read Port C
ANI 02H              ; Mask PC1 (OBFB)
JZ PRINT             ; Wait for printer ready
MOV A, C             ; Move content to ACC
OUT FD             ; Output to printer
RET
```

Programming 8255 PPI example

a# Control Register and Port Addresses:

An 8085 μp is interfaced with 8255 PPI to control the seven-segment display as shown below; Outline an assembly language to display the last six digits of your Mobile No.

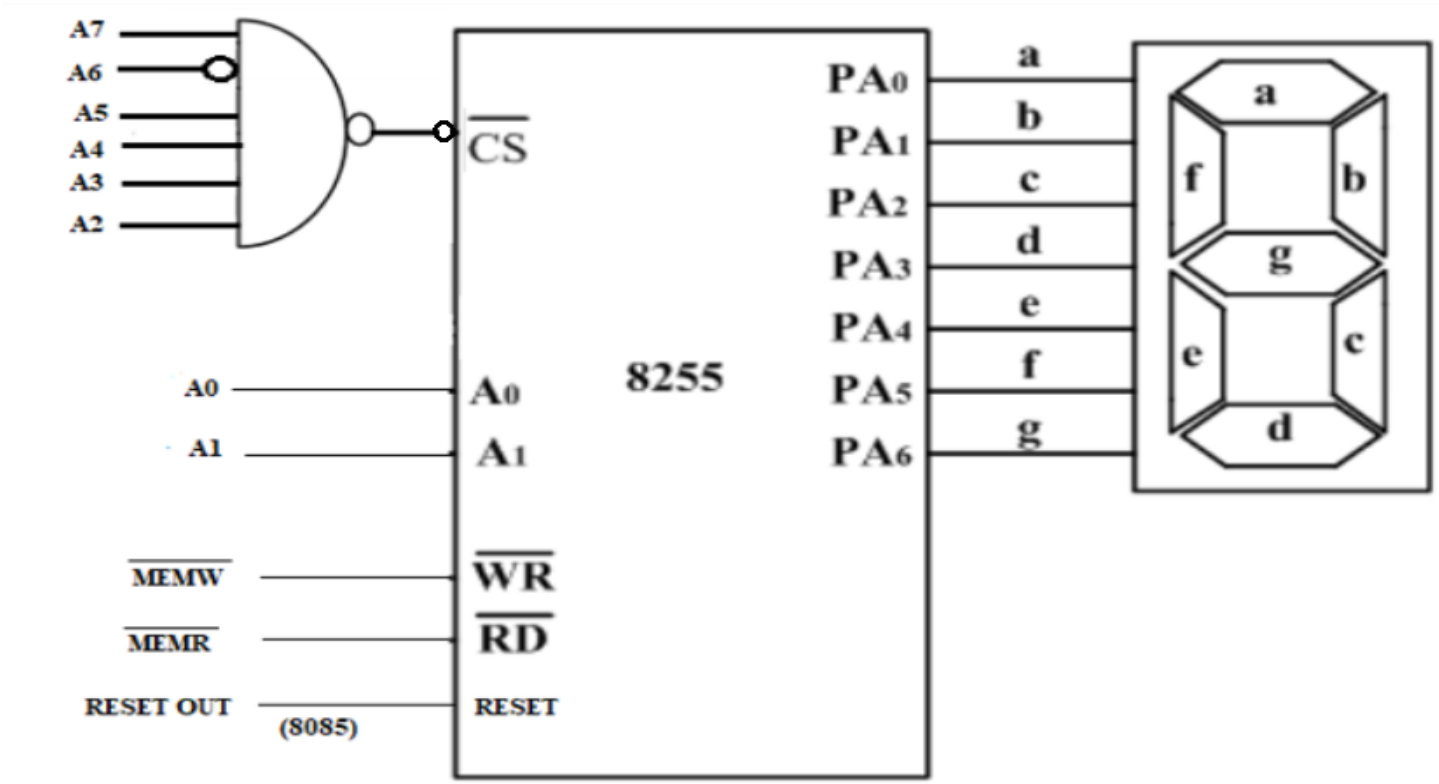


Fig. 11 Interfacing Seven-Segment Display in Mode-1

Programming 8255 PPI example

a# Control Register and Port Addresses:

```
; Initialize 8255 for Mode 1 Output on Port A
MVI A, A8 H  ; A8H = 10101000: Mode 1 Output (Port A),
    ; PC upper input
OUT BF H     ; Write to Control Word Register (CWR)

; Display digits: 6 5 4 3 2 1
; Each CALL HANDSHAKE_WRITE waits for
; ACK before sending next digit

MVI A, 7DH          ; Digit 6
CALL HANDSHAKE_WRITE

MVI A, 6DH          ; Digit 5
CALL HANDSHAKE_WRITE
```

Programming 8255 PPI example

a# Control Register and Port Addresses:

```
MVI A, 66H          ; Digit 4
CALL HANDSHAKE_WRITE

MVI A, 4FH          ; Digit 3
CALL HANDSHAKE_WRITE

MVI A, 5BH          ; Digit 2
CALL HANDSHAKE_WRITE

MVI A, 06H          ; Digit 1
CALL HANDSHAKE_WRITE

HLT                 ; End of program
```

Programming 8255 PPI example

a# Control Register and Port Addresses:

```
; Subroutine: HANDSHAKE_WRITE
HANDSHAKE_WRITE:
OUT BC H              ; Write data to Port A
WAIT_ACK:
IN BE H               ; Read Port C (status)
ANI 80 H              ; Check PC7 (OBF signal)
JZ WAIT_ACK          ; Wait until OBF is HIGH
RET
```

Examples of ADC and DAC Interface

Integrating 8-bit/ 8-channel (0808/0809) ADC using status check

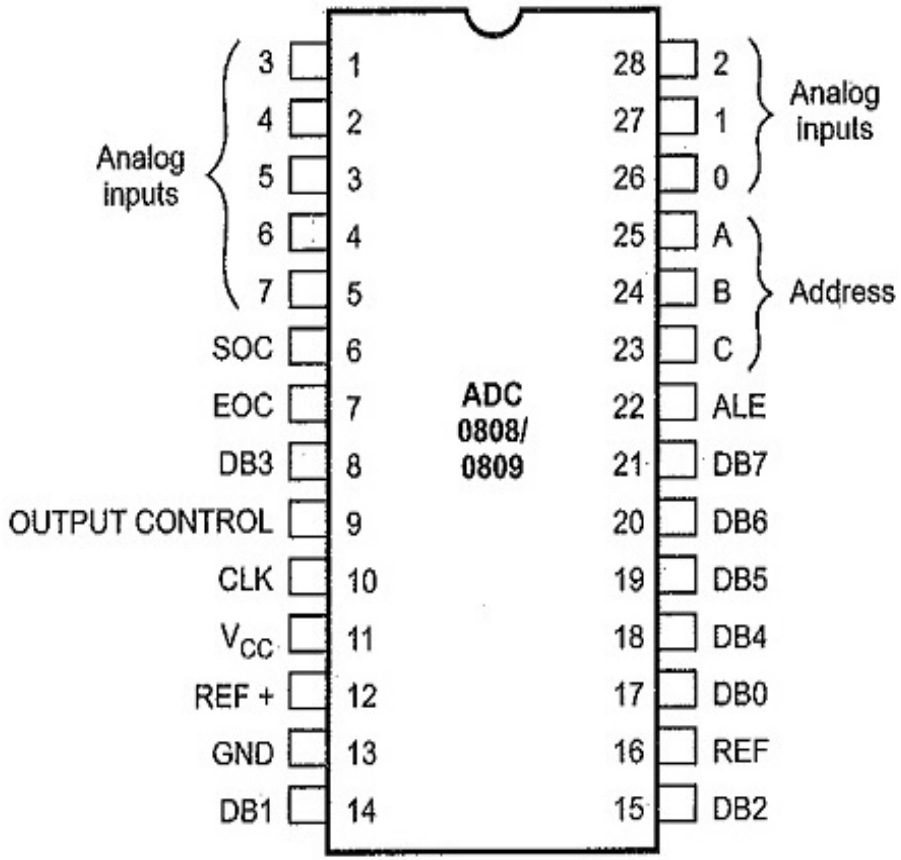


Fig. 12 Pin Diagram of ADC (0808/0809)

Examples of ADC and DAC Interface

Integrating 8-bit/ 8-channel (0808/0809) ADC using status check

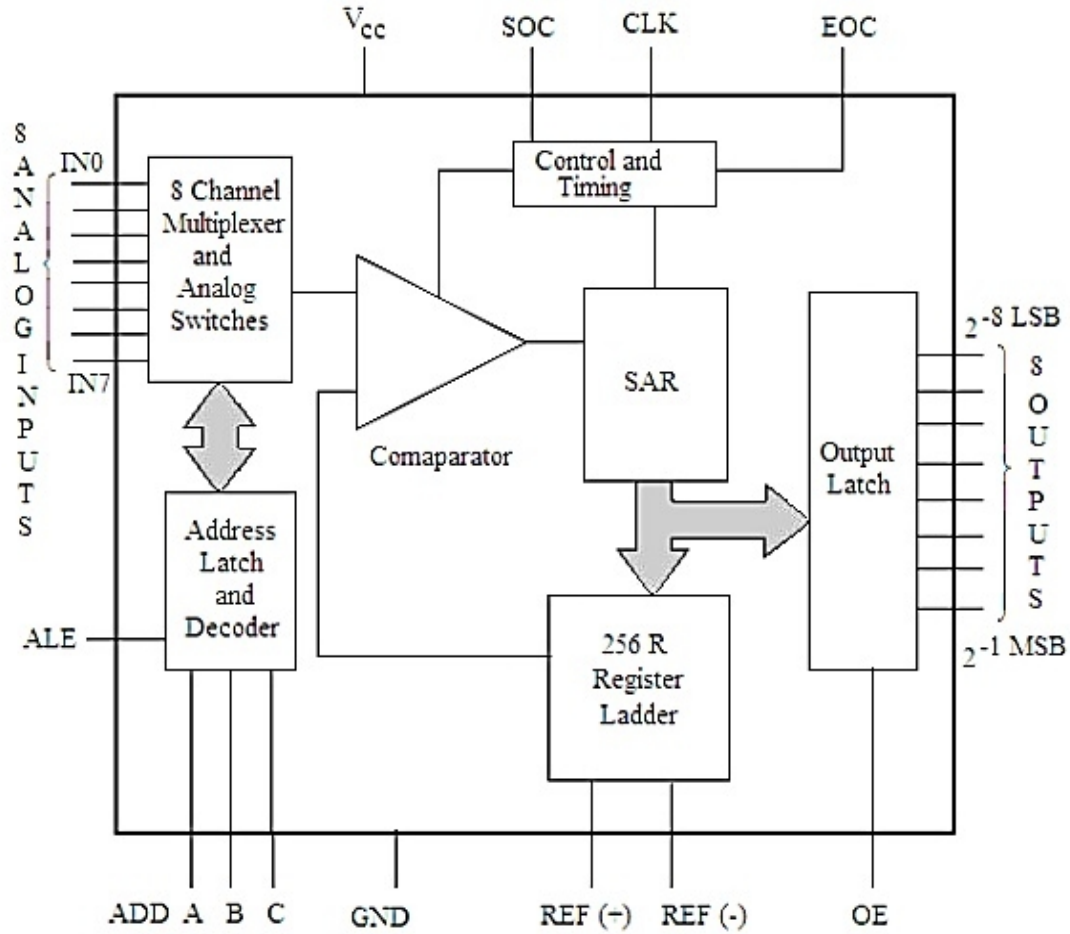


Fig. 13 Detail Internal Block Diagram of ADC (0808/0809)

Examples of ADC and DAC Interface

Integrating 8-bit/ 8-channel (0808/0809) ADC using status check

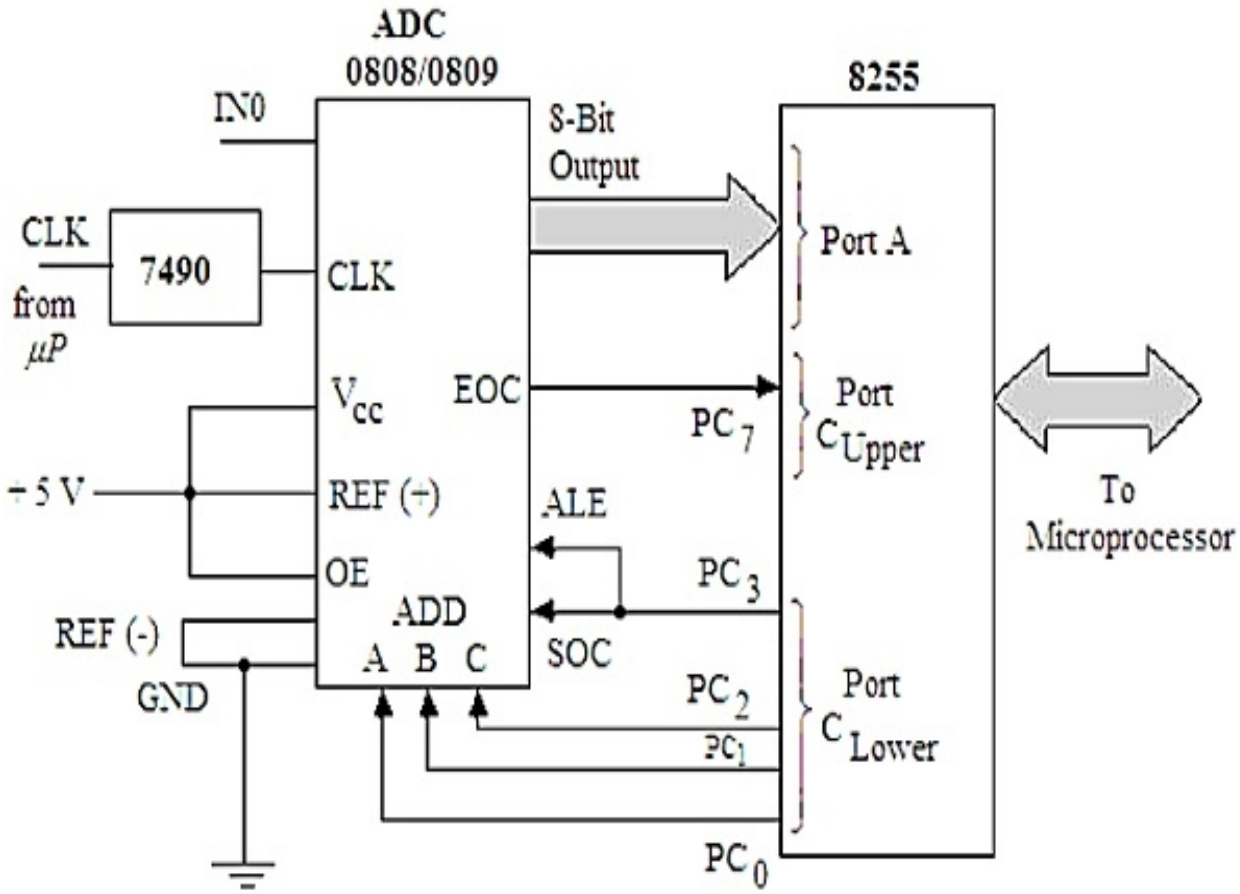


Fig. 14 Interfacing ADC(0808/0809) using 8255 PPI (base address 00H)

Programming 8255 PPI example

b# Different Control words:

Here we assume Base Address (PortA) to be 80H

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
EOC	x	x	x	x	x	x	x

Examples of ADC and DAC Interface

Integrating 8-bit/ 8-channel (0808/0809) ADC using status check

MVI A, B8H : Port A/Port B (input), PCu(input)

OUT 83H : Control word to Control Reg.

START_CONVERSION:

MVI A, 00H : Channel 0 address (A=0, B=0, C=0)

OUT 82H : Output to Port C lower bits to select channel

MVI A, 08H : Set ALE (PC3) high to latch the address

OUT 82H : Output to Port C to latch the address

NOP : Small delay to ensure ALE is high

MVI A, 00H : Set ALE/START(PC3) low

OUT 82H : Output to Port C to start the conversion

Examples of ADC and DAC Interface

Integrating 8-bit/ 8-channel (0808/0809) ADC using status check

WAIT_EOC:

IN 82H : Read from Port C (EOC)

ANI 80H : Mask all but the EOC bit (PC7)

JZ WAIT_EOC : Wait until EOC is high

IN 80H : Read from Port A (data lines)

HLT/RET : Halt the processor

Programming in Mode1(ADC Interfacing)

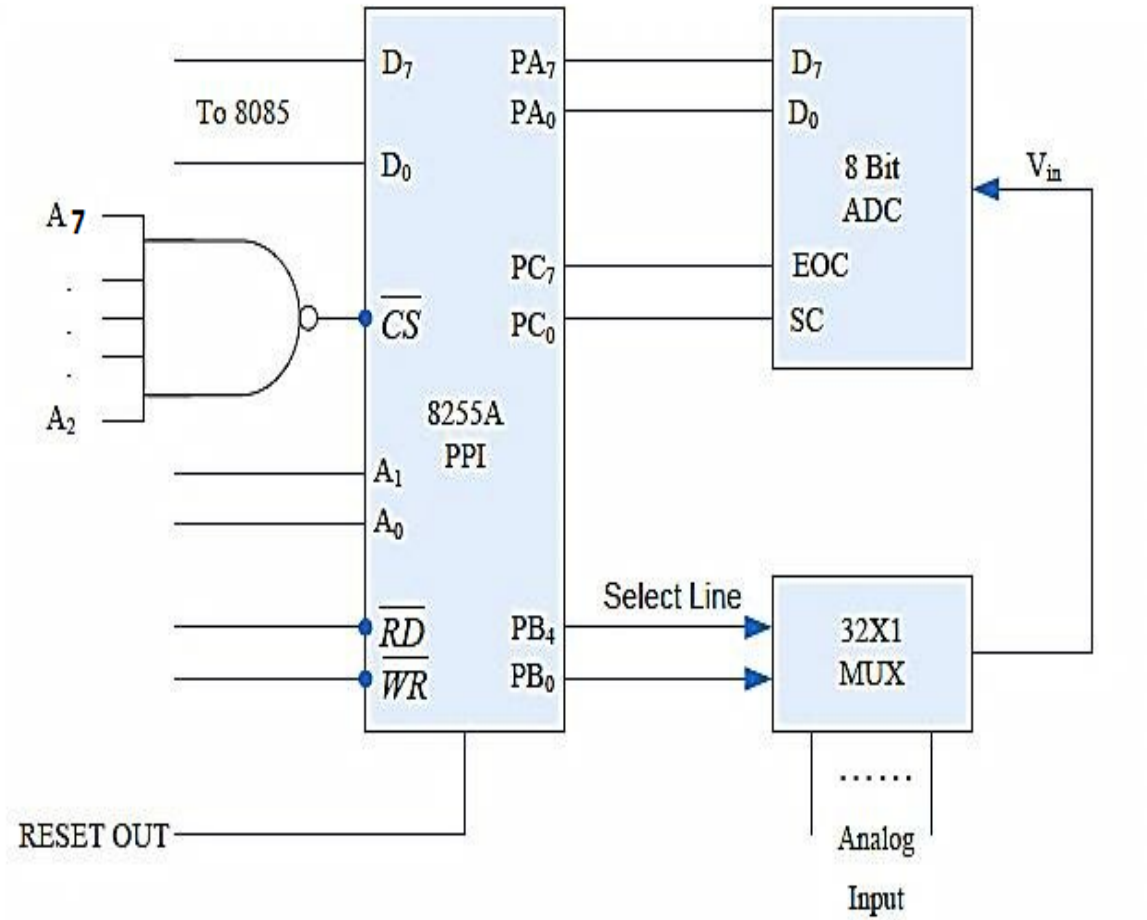


Fig. 15 Read digital output data for multiple input signals

Programming in Mode1(ADC Interfacing)

```
; Step 1: Initialize 8255 in Mode 1 for Port A,
      Mode 0 for Port B
MVI A, B8H          ; 10111000: Port A=Input (Mode1),
        Port B=Output, PC0=output, PC7=input
OUT FFH             ; Control register address

; Step 2: Select Analog Channel 03 (MUX)
MVI A, 03H          ; Select analog channel 3
OUT FDH             ; Send to Port B (MUX select)

; Step 3: Send SC pulse to ADC (PC0 = 1 then 0)
MVI A, 01H          ; Set PC0 = 1 (SC = HIGH)
OUT FEH             ; Send control to ADC
MVI A, 00H          ; Set PC0 = 0 (SC = LOW)
OUT FEH
```

Programming in Mode1(ADC Interfacing)

```
; Step 4: Wait for EOC (PC7 = 1)
WAIT_EOC:
IN FEH              ; Read Port C
ANI 80H             ; Mask PC7 (10000000)
JZ WAIT_EOC         ; Wait until EOC is HIGH

; Step 5: Read ADC data (from Port A)
IN FCH              ; Read from Port A
MOV B, A            ; Store ADC result in B

; Step 6: Store result to memory
LXI H, 3000H
MOV M, B
HLT
```

Programming in Mode1(DAC Interfacing)

Digital to Analog Converter (DAC) - 1408/0808 DAC

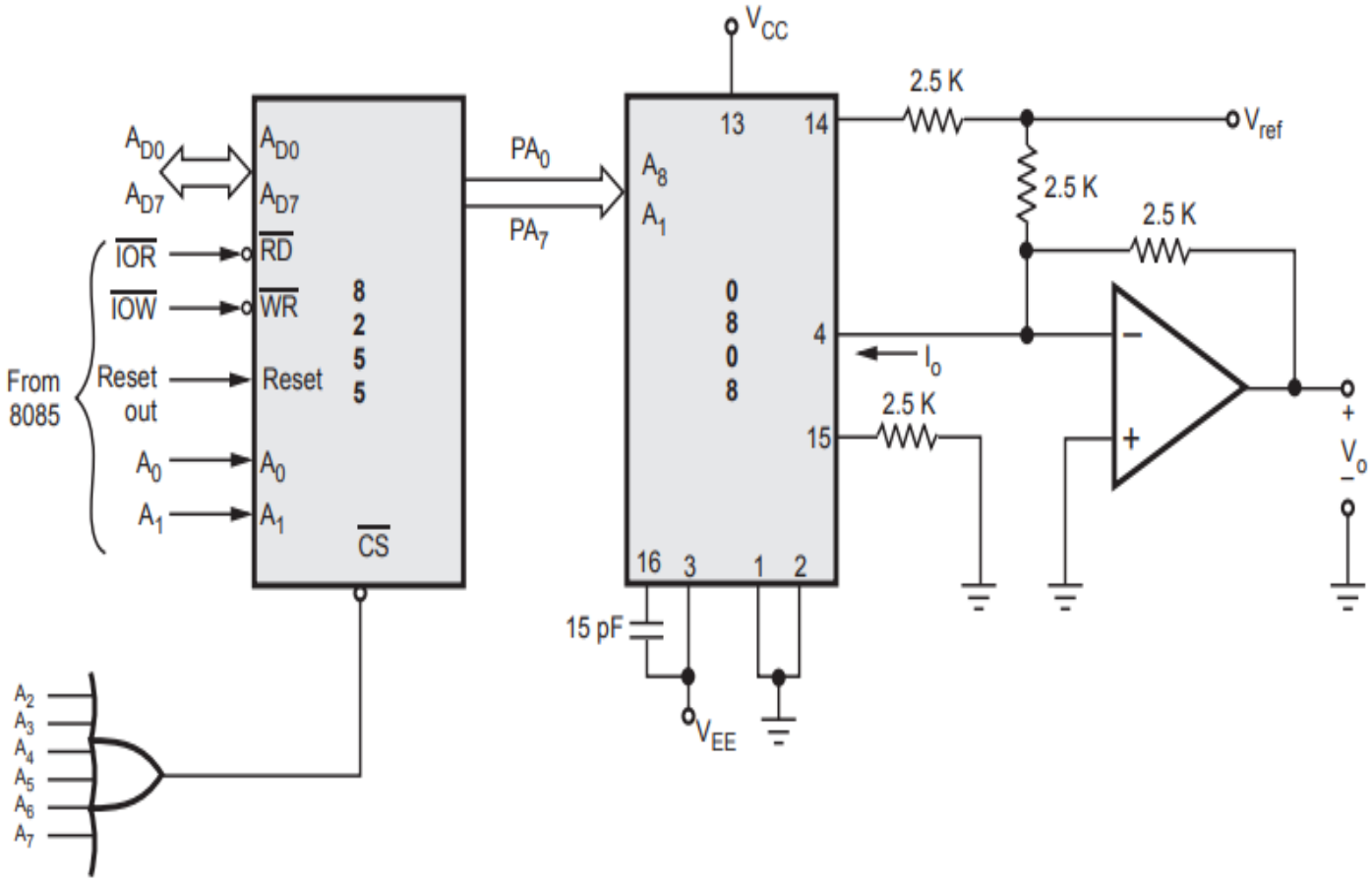


Fig. 16 Interfacing 0808 with microprocessor

Programming in Mode1(ADC Interfacing)

I/O control word:

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

```
MVI A, 80H          ; Control word to config. 8255
; Set all ports (A, B, C) as output in Mode 0
OUT 03H             ; Write to control register

MVI A, data         ; Load 8-bit data to be sent to DAC
OUT 00H             ; Send data to Port A (DAC input)
```

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

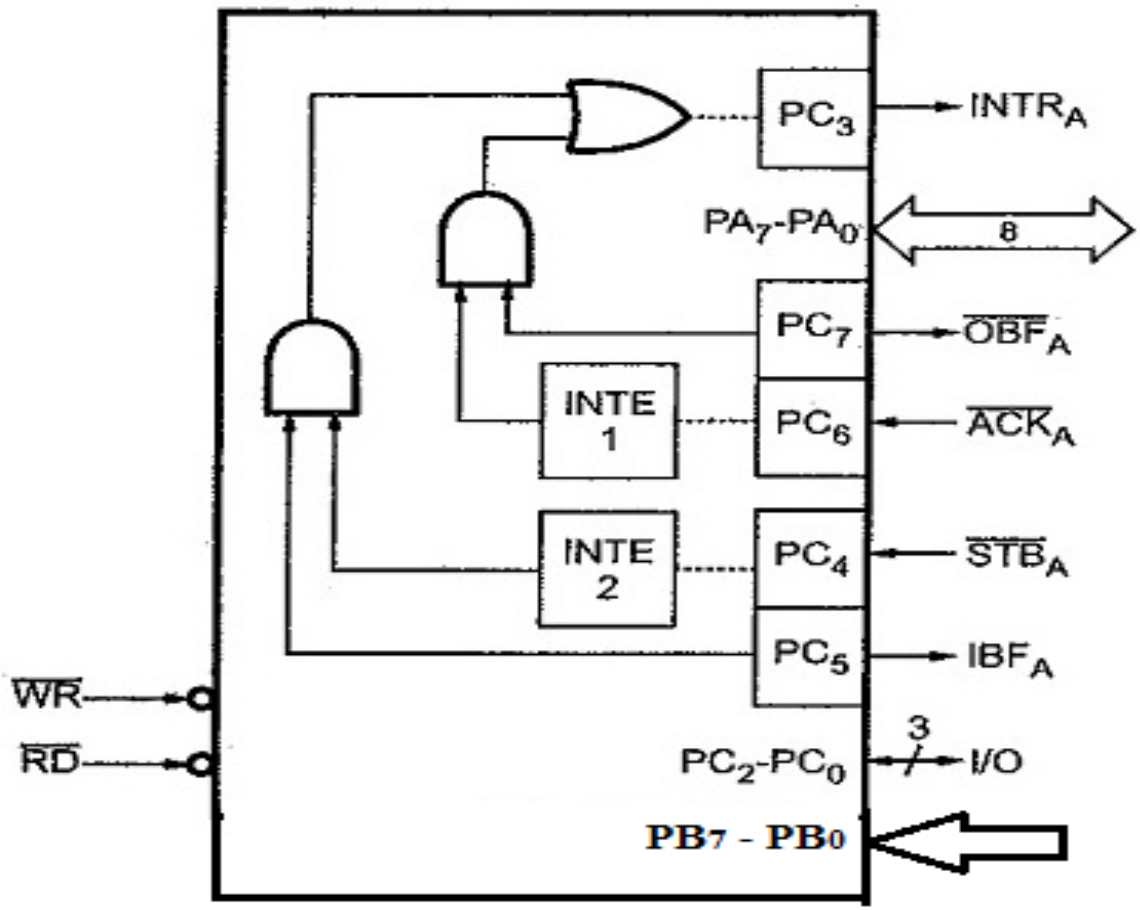


Fig. 17 8255 configured in Mode2, PortB mode0

Programming in Mode2(Strobe bidirectional)

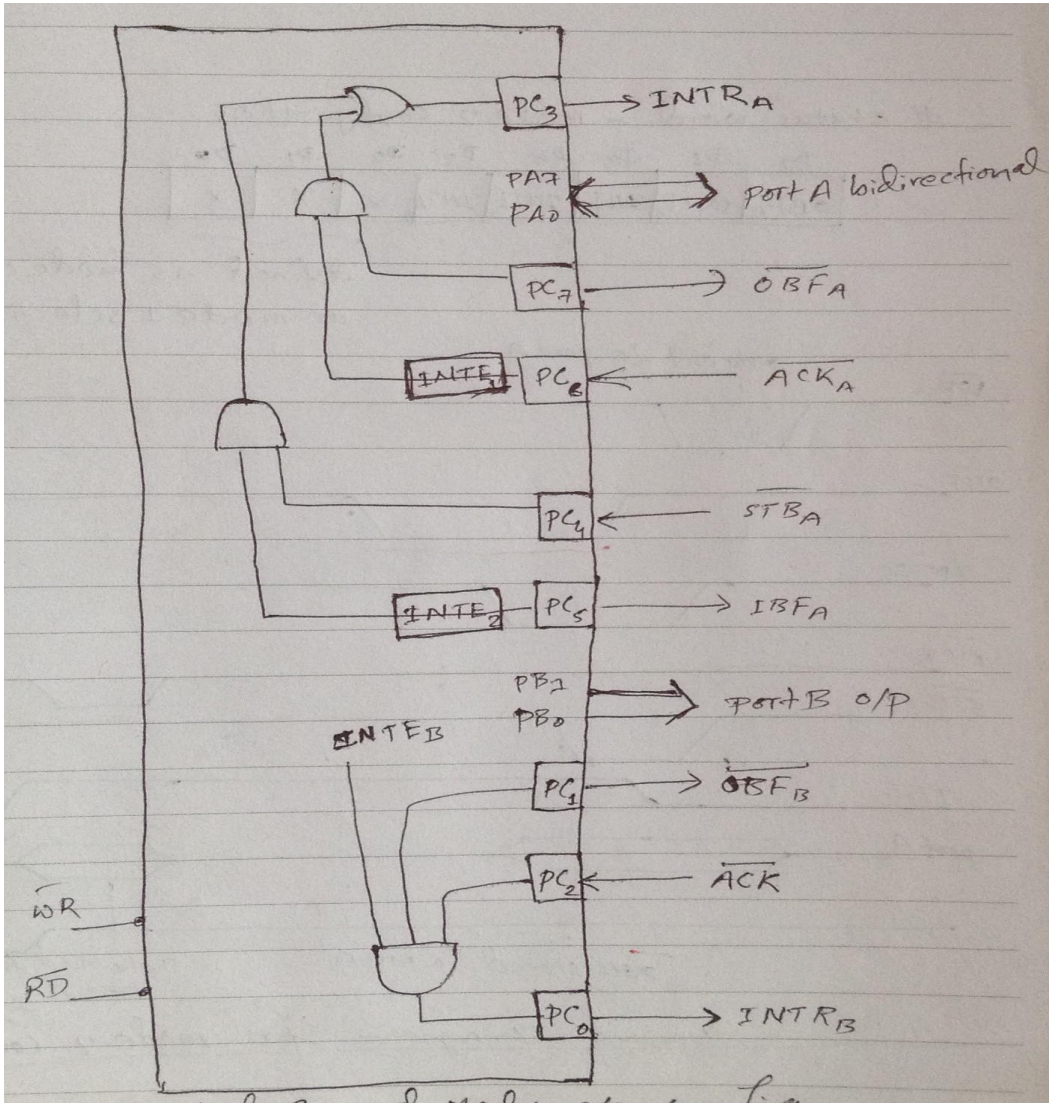


Fig. 18 8255 confiugred in Mode2; PortB mode1, O/P

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)

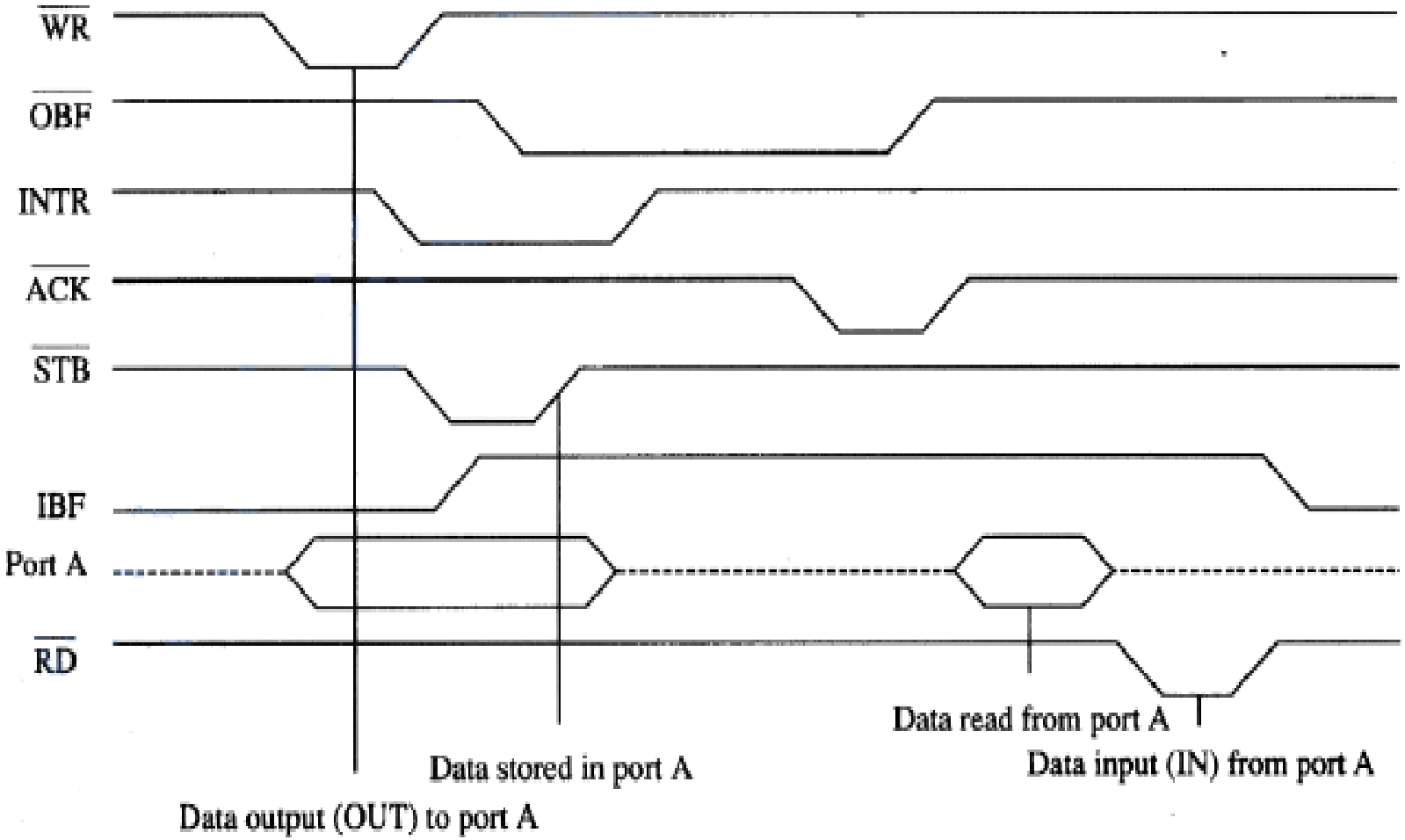


Fig. 19 Timing Diagram for Mode 2 Configuration

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

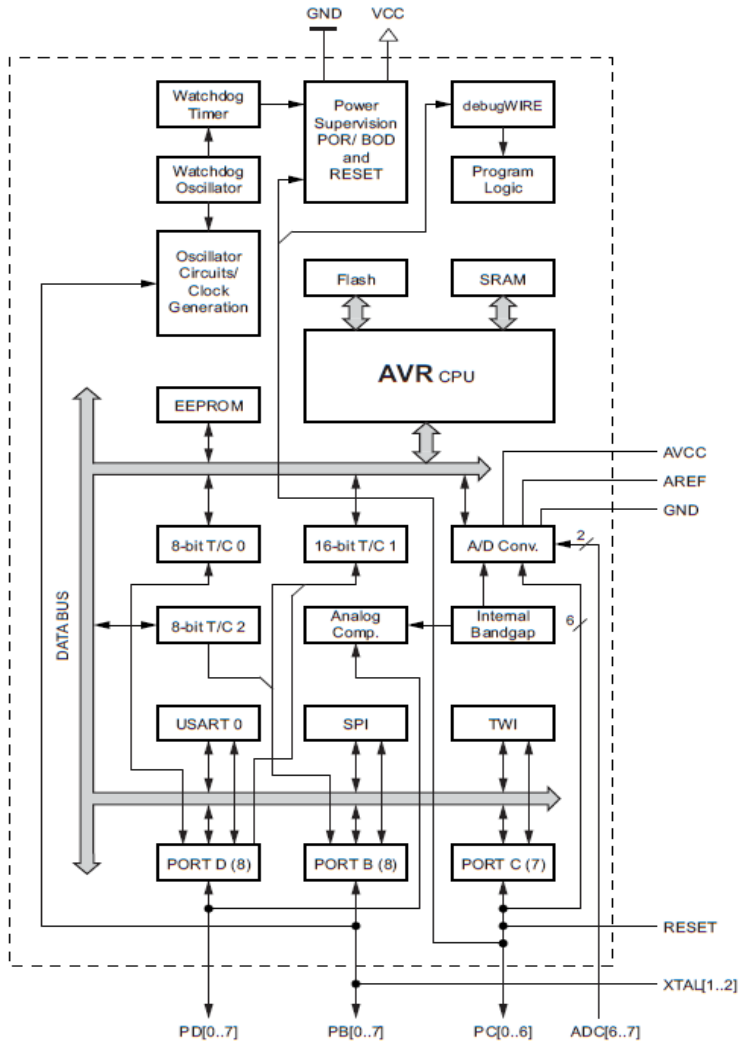
- the internal flip-flop associated to input buffer full.
- used to enable or disable the interrupt by setting PC_4 in BSR mode.

Microcontroller – ATMEGA328 and STEM32

ATMEGA328 (by Microchip) Architecture

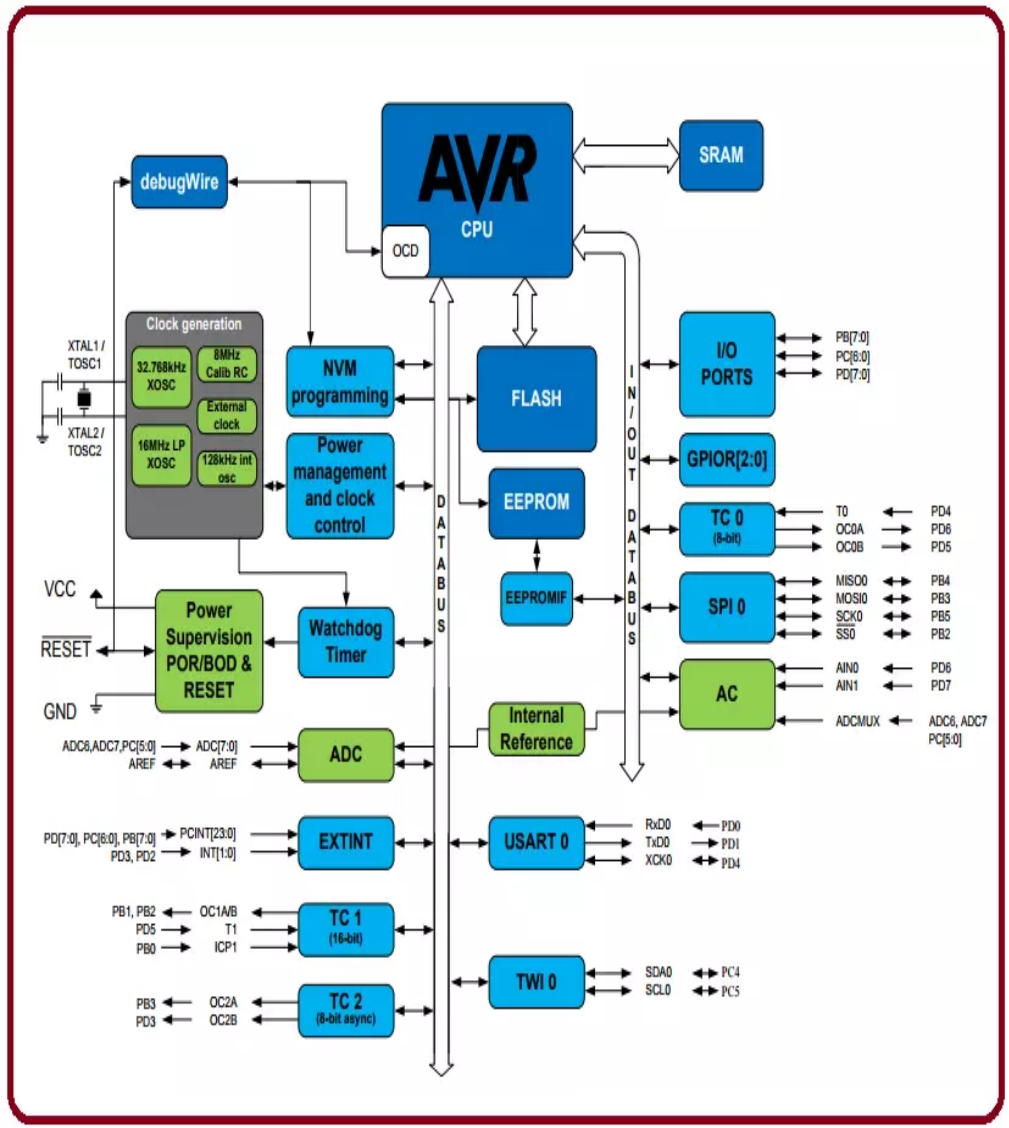
- ✔ Advanced Virtual RISC (AVR) μc
- ✔ supports 8-bit data processing
- ✔ 32KB internal flash memory
- ✔ 1KB EEPROM, 2KB SRAM
- ✔ 32 General Purpose (3 - I/O) Registers
- ✔ low power, 8MHz(int), 20MHz(ext)
- ✔ real timer counter - separate oscillator
- ✔ 8 Pins ADC operations (8 channels).
  - PortC (PC0 - PC5), ADC (6,7).
- ✔ Operating voltage – 1.8 to 5.5 V

- ➠ ATmega328 (DIP version) = 28 pins
- ➠ ATmega328 (TQFP/QFN) = 32 pins



Microcontroller – ATMEGA328 and STEM32

ATMEGA328 (by Microchip) Architecture



Microcontroller – ATMEGA328 and STEM32

Applications of ATMEGA328

- ✔ A complete package including ATmega328 and Arduino can be used in several different real-life applications.
- ✔ It can be used in Embedded Systems Projects.
- ✔ It can also be used in robotics.
- ✔ Quad-copter and even small aero-plane can also be designed through it.
- ✔ Power monitoring and management systems can also be prepared using this device.
- ✔ To design Home Security System

For more information click Here.

Microcontroller – ATMEGA328 and STEM32

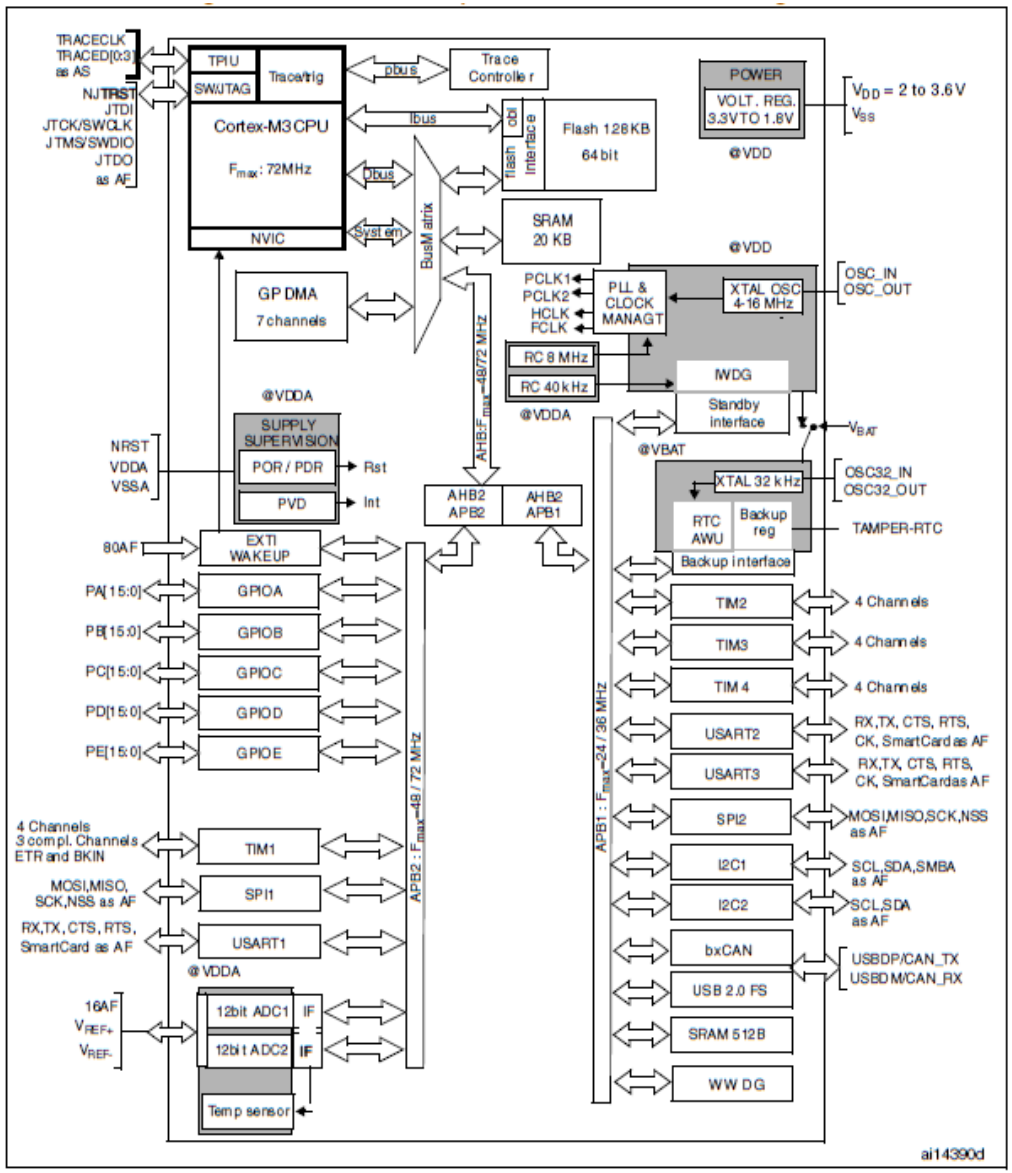
Different port in serial communication: ATMEGA328

Protocol	Function	Port	Pin	Arduino Pin
USART	TX	PD1	3	D1
	RX	PD0	2	D0
SPI	MOSI	PB3	17	D11
	MISO	PB4	18	D12
	SCK	PB5	19	D13
	SS	PB2	16	D10
I²C	SDA	PC4	27	A4
	SCL	PC5	28	A5

Microcontroller – ATMEGA328 and STEM32

STEM32F103 Architecture

- ✔ ARM Cortex-M3 CPU
- ✔ Operating at up to 72 MHz
- ✔ Memory:
  - Flash: Up to 128 KB
  - SRAM: 20 KB
- ✔ Multiple I/O
- ✔ ports: GPIOA to GPIOE
- ✔ Multiple serial, analog,
- ✔ and timer peripherals



Pin Configuration: | STEM32F103 Architecture

Port	Pins	Functions (Examples)
GPIOA	PA0–PA15	ADC1, USART2, SPI1, TIM2, etc.
GPIOB	PB0–PB15	I2C1, CAN, TIM3, SPI1 (alternate), etc.
GPIOC	PC13–PC15	Oscillator pins, general I/O
GPIOD/E	Optional	Available on higher pin-count packages only

Pin Configuration: | STEM32F103 Architecture

Pin/Port	Primary Use
PA0–PA7	ADC, TIM2, USART2, SPI, general I/O
PB6, PB7	I2C1
PB10, PB11	USART3
PA9, PA10	USART1 (default serial interface)
PC13	User LED (Blue Pill)
PA5, PA6, PA7	SPI1
PA11, PA12	USB D– / D+
PA13, PA14	SWDIO / SWCLK (debugging)
BOOT0	Select boot mode (from Flash, RAM, or System Memory)

Sensor Interfacing with ATMEGA328 (by Microchip)

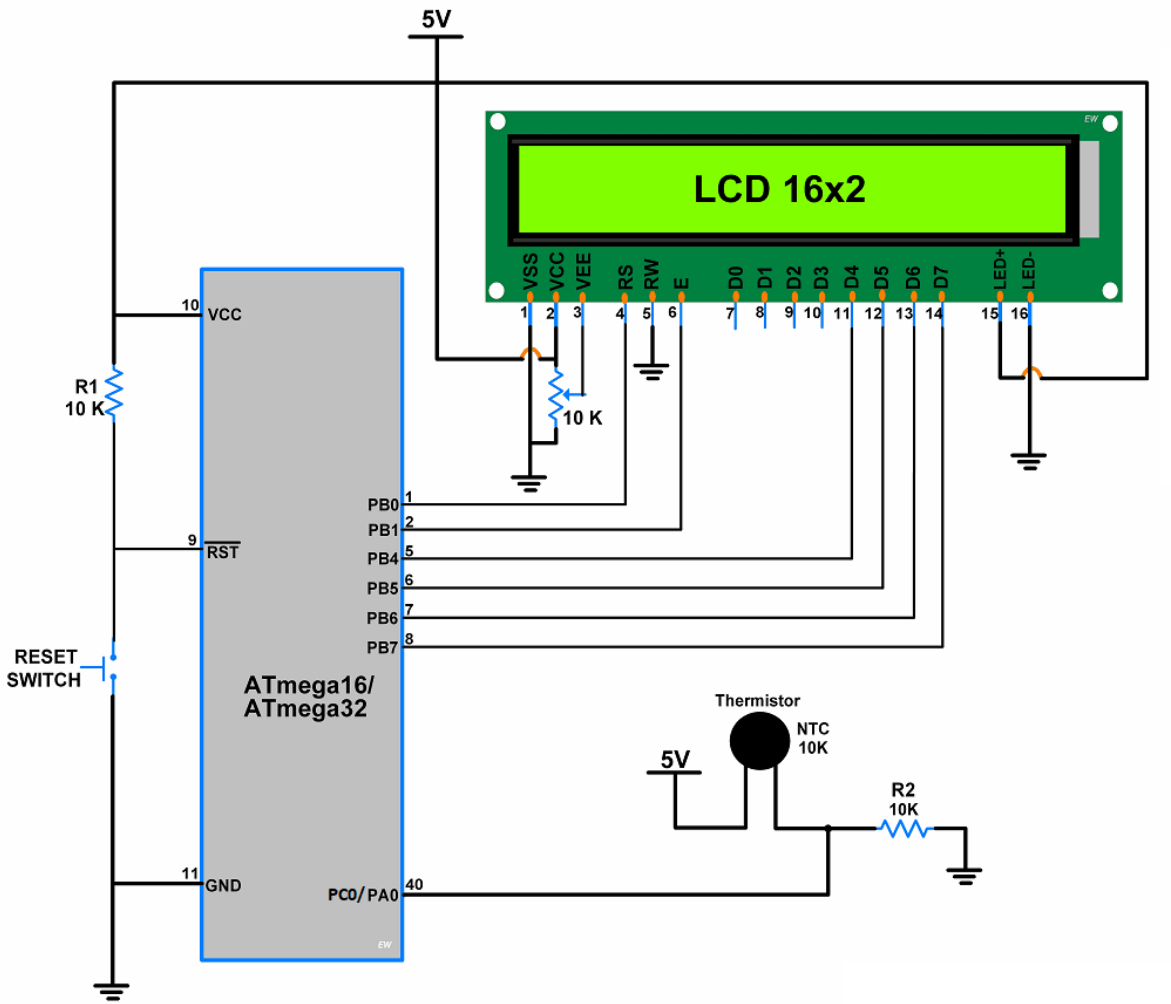


Fig. 20 Interfacing Thermistor With AVR ATmega32

Sensor Interfacing with ATMEGA328 (by Microchip)

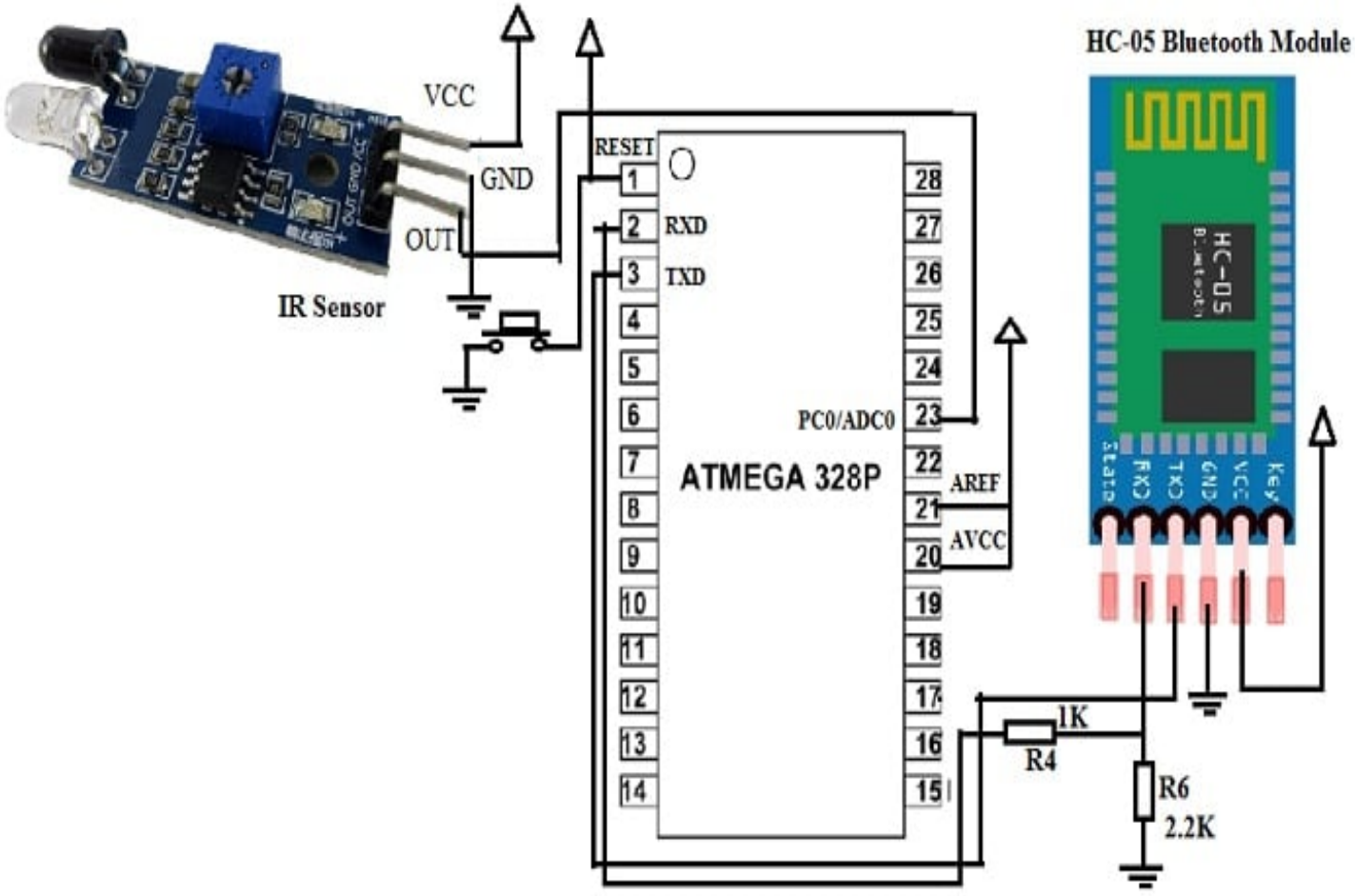


Fig. 21 Bluetooth & IR sensor Interfacing with ATmega328P

Microcontroller – ATMEGA328 and STEM32

Sensor Interfacing with ATMEGA328 (by Microchip)

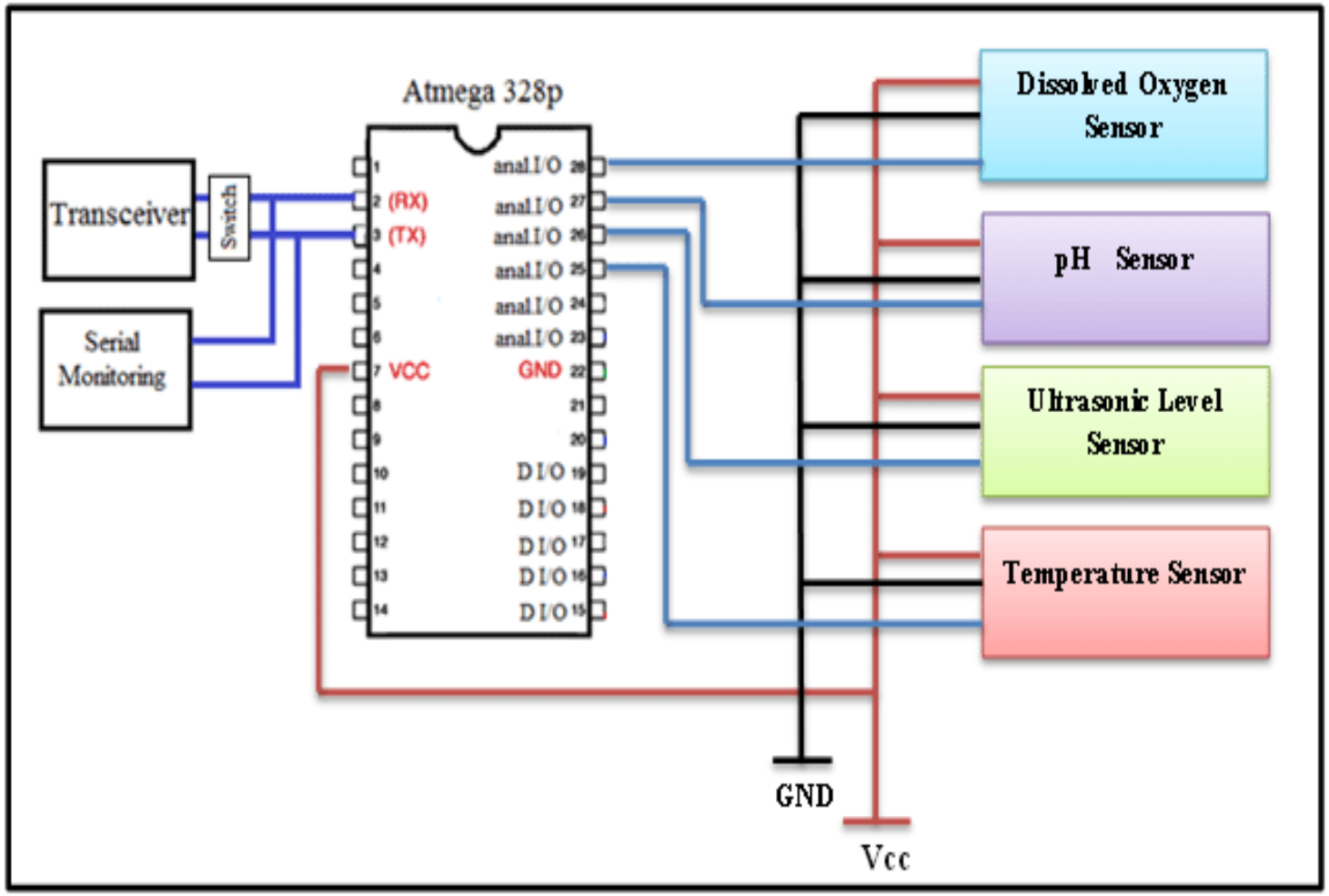


Fig. 22 Weather module using ATMEGA328

As you go
Assignment

Assignment Module#4 is available at MS-Team.

Deadline for submission: 8th July 2025 (*Before 3:00 PM*)