TOPIC 6

MEMORY AND I/O INTERFACING

MEMORY INTERFACING

i. External ROM (program memory) Interfacing

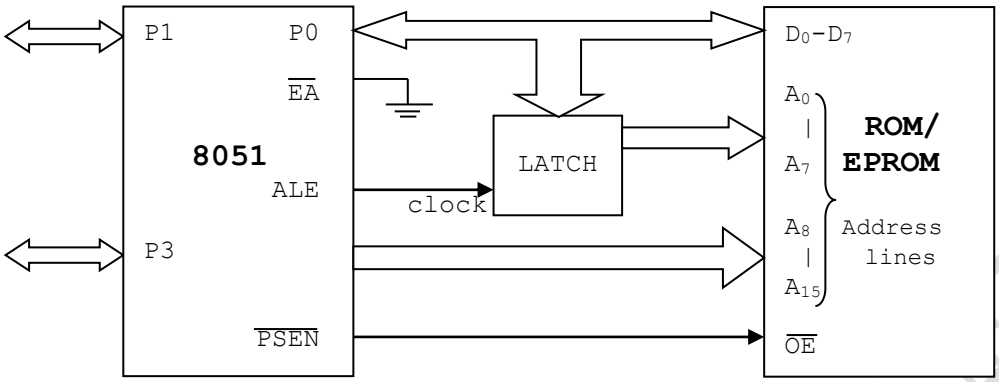


FIGURE 1 INTERFACING OF ROM/EPROM TO µC 8051.

above figure shows how to access or interface ROM to 8051.
port 0 is used as multiplexed data & address lines.
it gives lower order (A_7-A_0) 8 bit address in initial T cycle & higher order (A_8-A_{15}) used as data bus.
8 bit address is latched using external latch & ALE signal from 8051.
port 2 provides higher order (A_{15}-A_8) 8 bit address.
PSEN is used to activate the output enable signal of external ROM/EPROM.

ii. External RAM (data memory) Interfacing

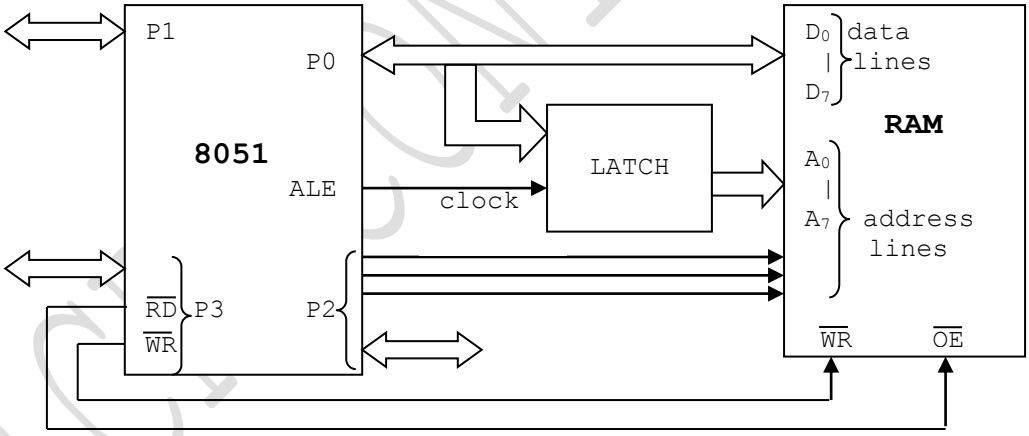


FIGURE 2 INTERFACING OF RAM(DATA MEMORY) TO µC 8051.

above figure shows how to connect or interface external RAM(data memory) to 8051.
port 0 is used as multiplexed data & address lines.
address lines are decoded using external latch & ALE signal from 8051 to provide lower order (A7-A0) address lines.
port 2 gives higher order address lines.
RD & WR signals from 8051 selects the memory read & memory write operations respectively.

RD & WR signals: generally P3.6 & P3.7 pins of port 3 are used to generate meamory read and memory write signals.
remaining pins of port 3 i.e. P3.0-P3.5 can be used for other functions.

LINEAR AND ABSOLUTE DECODING

i. Absolute Decoding

all higher address lines : decoded to select memory chip for specific logic levels.

for other logic levels memory chip is disabled.

generally used in large memory systems.

figure below shows memory interfacing using absolute decoding.

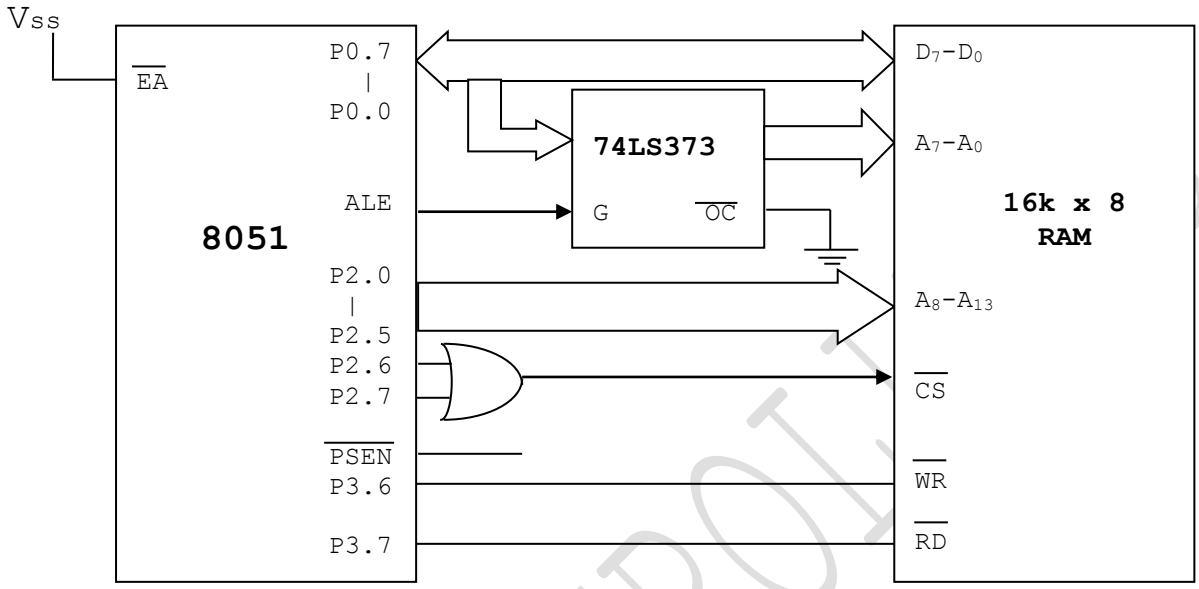


FIGURE 3 MEMORY (RAM) INTERFACING USING ABSOLUTE DECODING.

ii. Linear Decoding (Partial Decoding)

for small systems : individual higher order address lines used to select memory chip.

replacing the hardware by decoding logic.

reducing the cost of decoding, drawback is- multiple addresses.

as shown in figure below, A_{14} line is directly connected to chip select line, A_{15} line not connected anywhere, kept open.

so, status of A_{15}- not considered for generation of chip select signal.

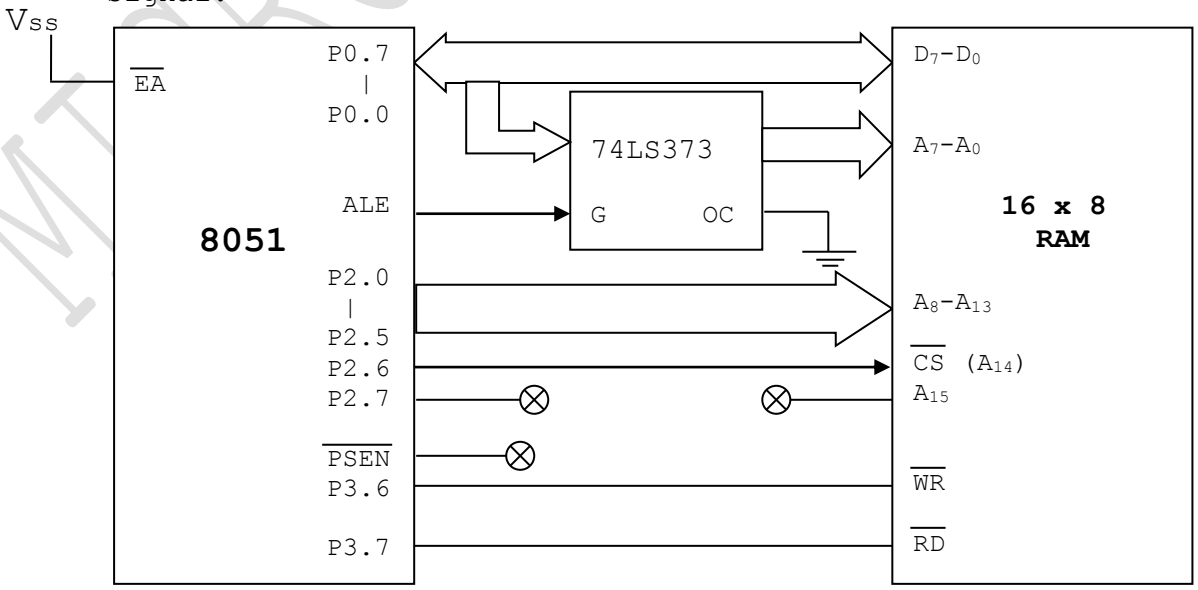


FIGURE 4 MEMORY (RAM) INTERFACING USING LINEAR DECODING.

Address Mapping(Memory Map)

i. Absolute Decoding

Address	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	HEX adrs.
starting	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000H
end	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3FFFH

ii. Linear Decoding

Address	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	HEX adrs.
starting	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000H
end	x	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3FFFH

Comparison between Full address(Absolute) & Partial address (Linear) Decoding.

Full Adress(Absolute) Decoding	Partial Address(Linear Decoding
i. all higher address lines are decoded to select memory or I/O device.	i. few or individual address lines are decoded to select memory or I/O device.
ii. more hardware : decoding logic.	ii. less hardware : decoding logic. (sometimes none.)
iii. decoding circuit : higher cost.	iii. decoding circuit : less cost.
iv. No multiple addresses.	iv. multiple addresses possible.
v. used in large systems.	v. used in small systems.

Solved Examples:

Example 1: Design a µController system using 8051.Interface the external RAM of size 16k x 8.

Solution: Given, Memory size: 16k

that means we require 2^n=16k :: n address lines

here n=14 :: A_0 to A_{13} address lines are required.

A_{14} and A_{15} are connected through OR gate to CS pin of external RAM.

when A_{14} and A_{15} both are low (logic '0'), external data memory(RAM) is selected.

Address Decoding(Memory Map)for 16k x 8 RAM.

Address	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	HEX adrs.
starting	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000H
end	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3FFFH

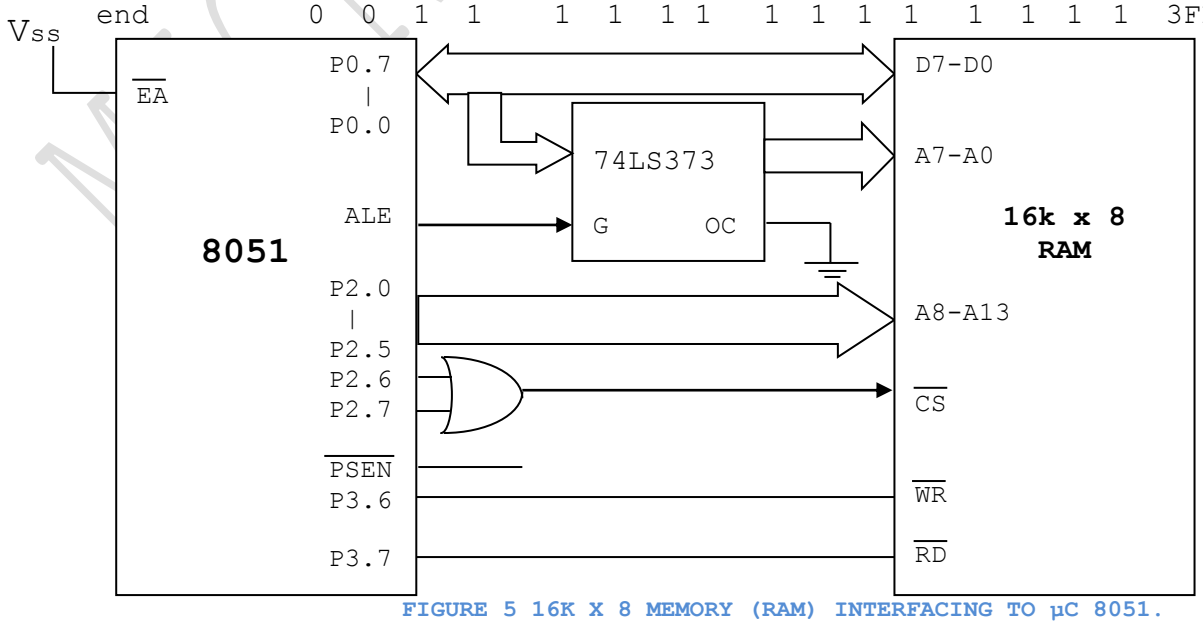


FIGURE 5 16K X 8 MEMORY (RAM) INTERFACING TO µC 8051.

Example 2: Design a µController system using 8051.Interface the external ROM of size 4k x 8.

Solution: Given, Memory size: 4k

that means we require 2^n=4k :: n address lines

here n=12 :: A_0 to A_{11} address lines are required.

remaining lines A_0, A_0, A_0, A_0 & PSEN are connected though OR gate to CS & RD of external ROM.

when A_0 to A_0 are low (logic '0'), only then external ROM is selected.

Address Decoding(Memory Map)for 4k x 8 RAM.

Address	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	HEX adrs.
starting	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000H
end	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0FFFH

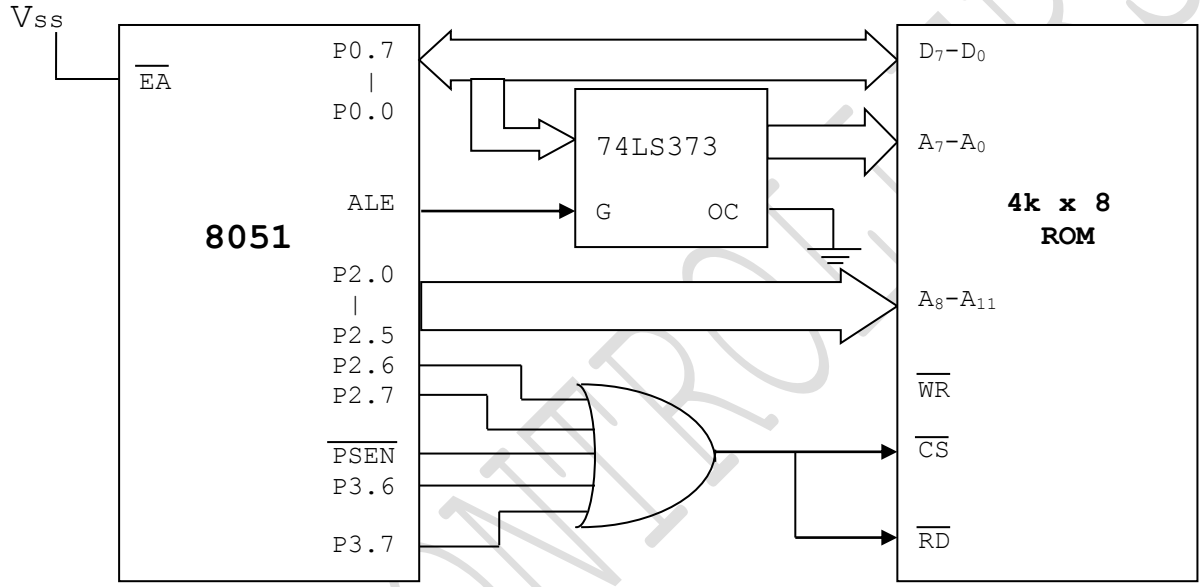


FIGURE 6 4K X 8 MEMORY (ROM) INTERFACING TO µC 8051.

Example 3: Design a µController system using 8051, 16k bytes of ROM & 32k bytes of RAM. Interface the memory such that starting address for ROM is 0000H & RAM is 8000H.

Solution: Given, Memory size- ROM : 16k

that means we require 2^n=16k :: n address lines

here n=14 :: A_0 to A_{13} address lines are required.

A14,A15,PSEN ⟶ ORed ⟶ $\overline{CS}$

when low – ROM is selected.

Memory size- RAM :32k

that means we require 2^n=32k :: n address lines

here n=15 :: A_0 to A_{15} address lines are required.

A15 ⟶ inverted(NOT Gate) ⟶ $\overline{CS}$

when high- RAM is selected.

$\overline{PSEN}$ is used as chip select pin ROM.

$\overline{RD}$ is used as read control signal pin.

$\overline{WR}$ is used as write control signal pin.

for RAM selection.

Address Decoding(Memory Map)for 16k x 8 ROM.

Address	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	HEX adrs.
starting	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000H
end	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3FFFH

Address Decoding(Memory Map)for 32k x 8 RAM.

Address	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	HEX adrs.
starting	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8000H
end	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	FFFFH

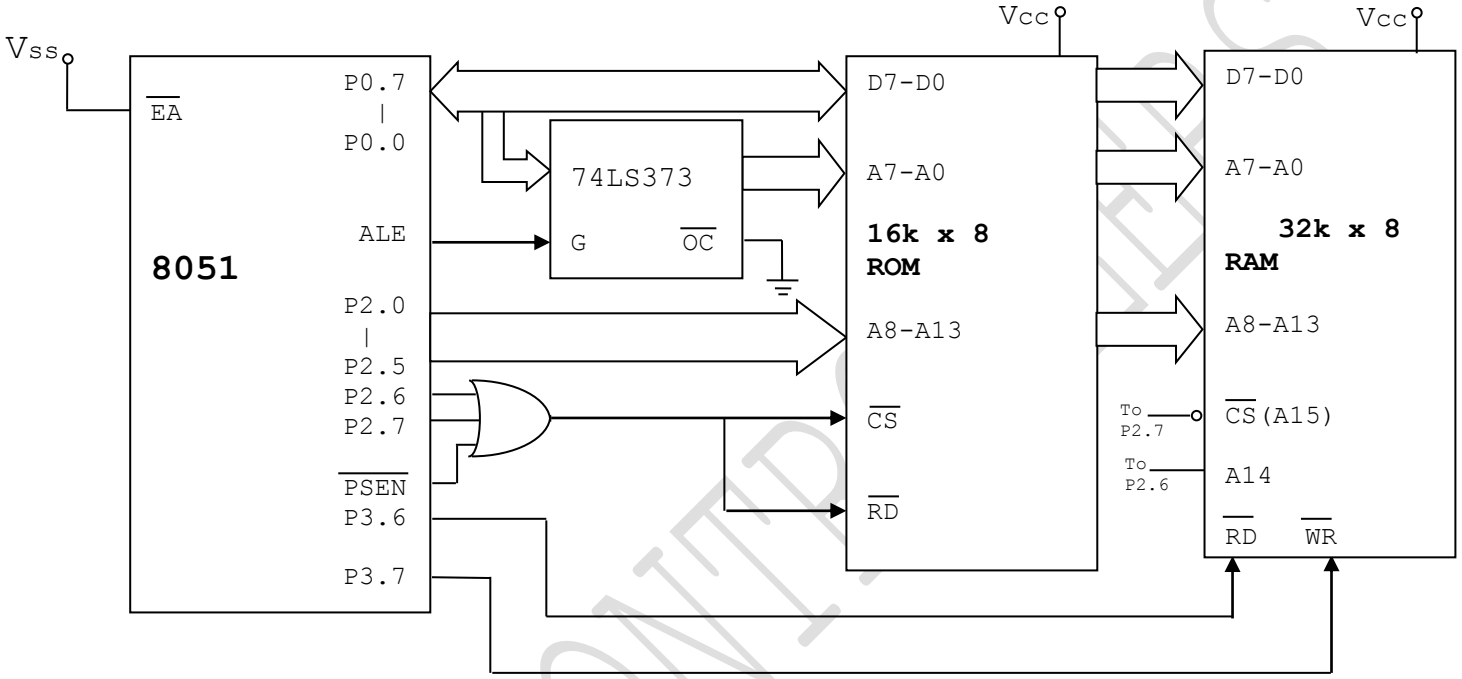


FIGURE 7 16K X 8 ROM AND 32K X 8 RAM INTERFACING TO µC 8051.

Example 4: Design a µController system using 8051, 8k bytes of program ROM & 8k bytes of data RAM. Interface the memory such that starting address for ROM is 0000H & RAM is E000H.

Solution: Given, Memory size- ROM : 8k

that means we require 2^n=8k :: n address lines

here n=13 :: A_0 to A_{12} address lines are required.

A13,A14,A15,PSEN ⟶ ORed ⟶ $\overline{CS}$

when low – program ROM is selected.

Memory size- RAM :8k

that means we require 2^n=8k :: n address lines

here n=13 :: A_0 to A_{12} address lines are required.

A13,A14,A15 ⟶ NANDed ⟶ $\overline{CS}$

when high- data RAM is selected.

$\overline{PSEN}$ is used as chip select pin ROM.

$\overline{RD}$ is used as read control signal pin.

$\overline{WR}$ is used as write control signal pin.

for RAM selection.

Address Decoding(Memory Map)for 8k x 8 ROM.

Address	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	HEX adrs.
starting	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000H
end	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1FFFH

Address Decoding(Memory Map)for 8k x 8 RAM.

Address	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	HEX adrs.
starting	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	E000H
end	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	FFFFH

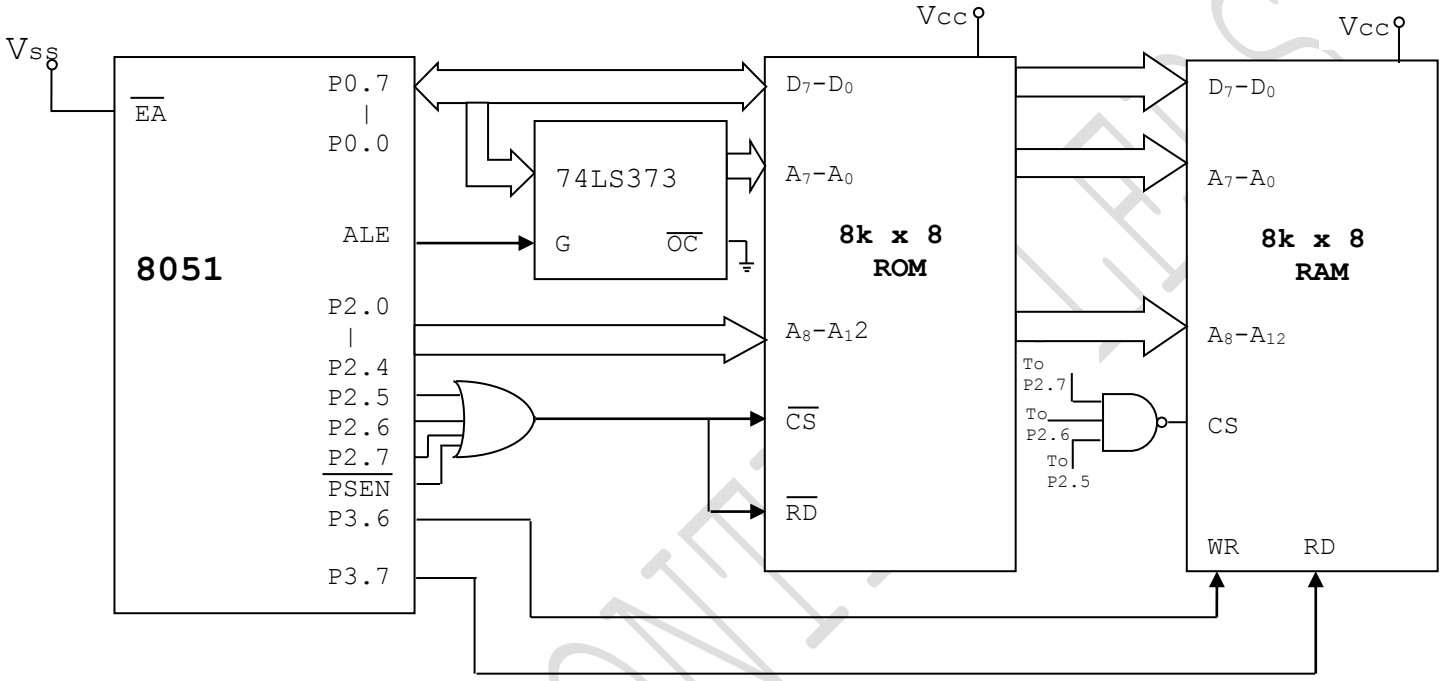


FIGURE 8 8K X 8 ROM AND 8K X 8 RAM INTERFACING TO µC 8051.

I/O interfacing

1. 8255 Programmable Peripheral Interface-

Features:

***** widely used programmable parallel I/O device.
* TTL compatible & compatible with all Intel & most other processors.
* can be programmed to transfer data- simple I/O, Interrupt I/O.
* three 8 bit ports- port A, port B and port C.
* bit set/reset mode: setting/resetting of individual bits of port c.
* can operate in I/O modes-
    i. Mode 0
    ii. Mode 1
    iii. Mode 2

port A programmed as input or output : with or without handshaking signals.

or as a bidirectional port.

port B programmed as input or output : with or without handshaking signals.

port C divided in two parts : port C upper (PC4-PC7) port C lower (PC0-PC3)

can be programmed as an input or output port.

can be individually set/reset to generate control signals.

40 pin DIP chip.

8255 Block Diagram:

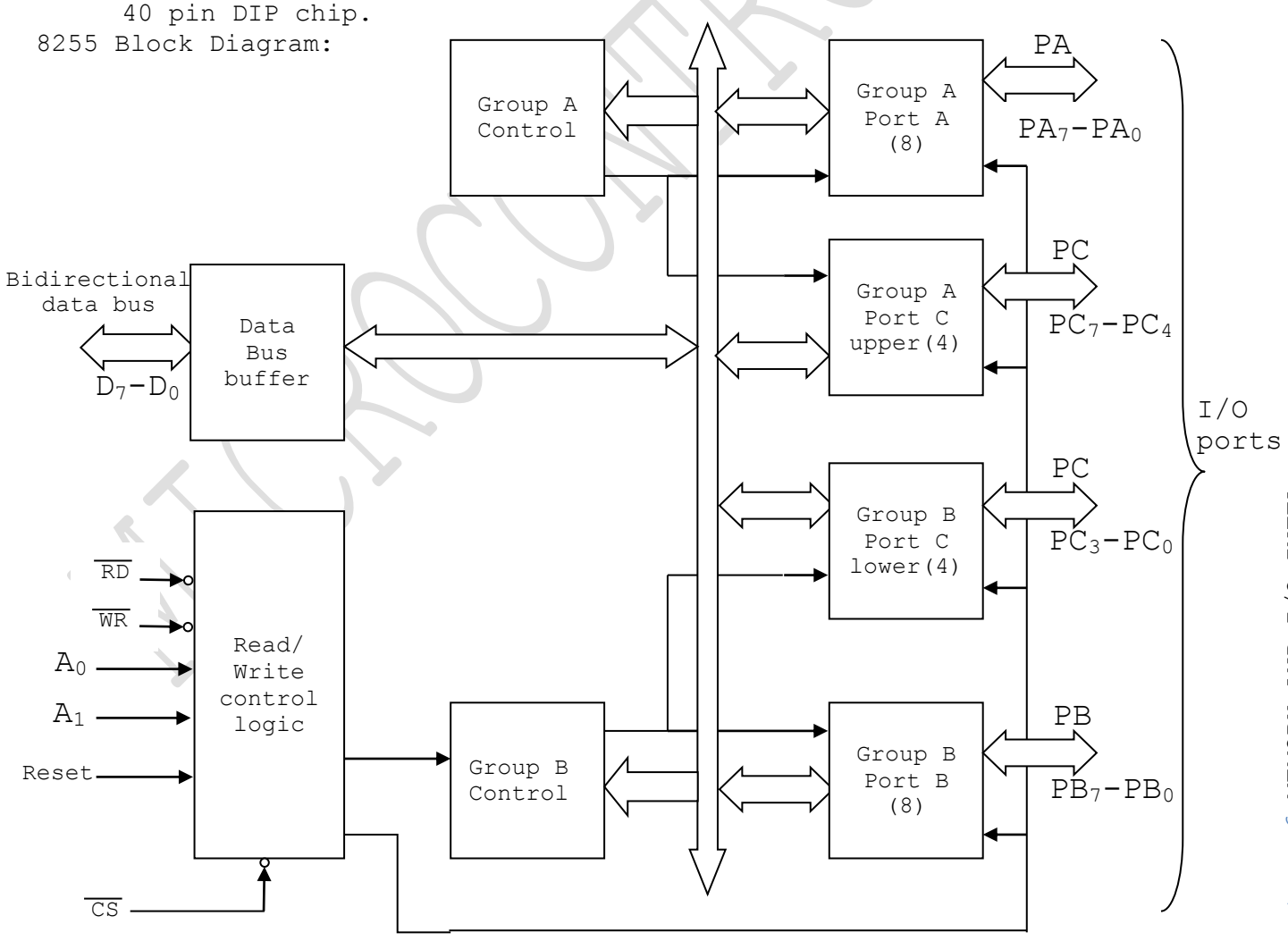


FIGURE 9 BLOCK DIAGRAM OF 8255A.

above figure shows internal block diagram of 8255.
consist of data bus buffer, control logic- group A and group B.

DATA BUS BUFFERS-

tri-state bidirectional buffer.
interfaces internal databus of 8255 to the system databus.
i/p or o/p instructions executed by the CPU either read data from or write data into buffer.
data from & to CPU always passed through buffer.

CONTROL LOGIC-

accepts control bus signals, i/ps from address bus & issues command to the individual group control blocks.(Group A & Group B control)
it issues appropriate enabling signals to the access required data/control words or status words.

following are the inputs for the control logic section,

Group A and Group B control-
receives control words from CPU & issues appropriate commands to ports associated with it.
Group A controls the Port A and Port C upper part i.e. PC_7-PC_4.
Group B controls the Port B and Port C lower part i.e. PC_3-PC_0.

Port A-8 bit latched inputs.
8 bit latched & buffered output.
can be programmed in 3 modes viz. mode 0, mode 1 and mode 2.

Port B-8 bit data input buffer.
8 bit data I/O latch/buffer.
can be programmed in mode 0 & mode 1.

Port C-8 bit unlatched input buffer.
8 bit output latch/buffer.
divide into two parts- each one used as control signal for port A & B in handshake mode.
can be programmed for bit set/reset operation.

8255 PIN DIAGRAM

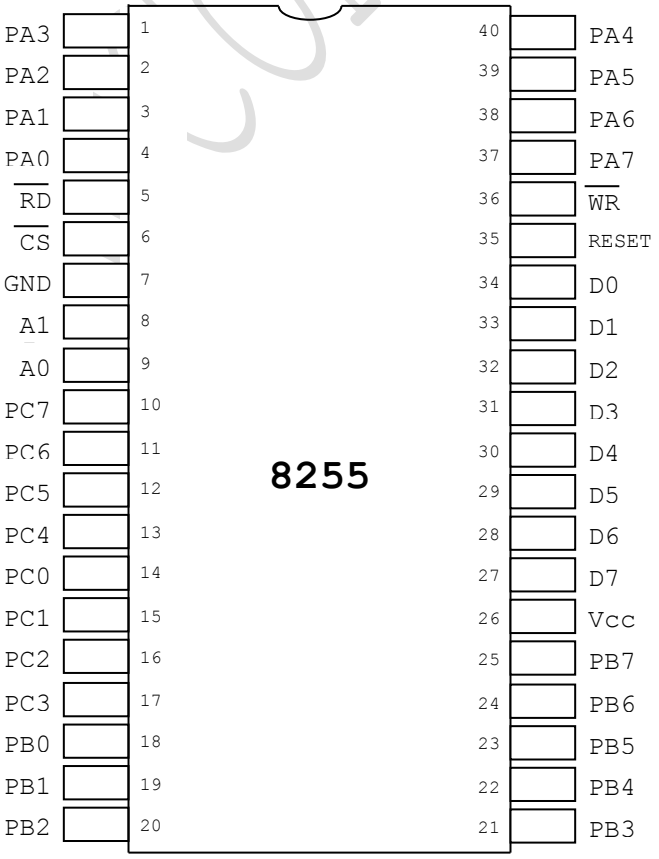


FIGURE 9 PIN DIAGRAM OF 8255.

figure above shows the pin diagram of 8255.

PA0-PA7 (Port A)

8 bit bidirectional I/O pins.
to send/receive data to & from I/O devices.
functions as 8 bit data i/p buffer, 8 bit data o/p buffer/latch.

D0-D7 (Data Bus)

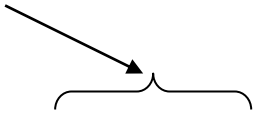
8 bit bidirectional, tri-state data bus.
connected to system data bus.
used to transfer data & control word from processor to 8255 or to receive data/status word from 8255 to processor.

PB0-PB7 (Port B)

8 bit bidirectional I/O pins.
to send/receive data to & from I/O devices.
functions as 8 bit data i/p buffer, 8 bit data o/p buffer/latch.

PC0-PC7 (Port C)

8 bit bidirectional I/O pins.



PC U (PC7-PC4) PC L (PC3-PC0)

for handshaking purpose.
to send receive data in & out.

RD (Read)

when low, CPU can read the data in port or status word through buffer.

WR (Write)

when low, CPU can write the data on ports or in control registers through buffers.

CS (Chip Select)

when low, chip is activated/enabled for data transfer between CPU and 8255.

RESET

active high input to reset 8255.
when RESET= 1 :: control register is cleared & all ports are set to input mode.
usually RESET OUT from processor is used to reset 8255.

A0 & A1 selects specific ports & control register.

A1	A0	RD	WR	CS	OPERATION
					Input (read) operation
0	0	0	1	0	port A to data bus.
0	1	0	1	0	port B to data Bus.
1	0	0	1	0	port C to data bus.
					Output (write) operation
0	0	1	0	0	data bus to port A.
0	1	1	0	0	data bus to port B.
1	0	1	0	0	data bus to port C.
1	1	1	0	0	data bus to control register.
					Disable Function
X	X	X	X	1	data bus tri-stated.
1	1	0	1	0	illegal condition.
X	X	1	1	0	data bus tri-stated.

Operating Modes

Bit Set Reset Mode (BSR mode)

individual bits of port C can be set or reset by sending out single OUT instruction to the control register, whenever port C is used for control/status operation.

I/O modes-

1. Mode 0: simple Input/ Output

   Port A & B :: two simple 8 bit I/O ports.
   Port C :: two 4 bit ports.
   port(any) can be programmed to function as simply input or output port.
   I/O features are as follows-
   i. outputs are latched.
   ii. inputs are buffered, not latched.
   iii. ports don't have handshake or interrupt capability.

2. Mode 1: Input/ Output with handshake.

   i/p or o/p data transfer is controlled by handshaking signals.

   to transfer data between devices whose data transfer speeds are not same.

   Features-
   i. Port A & B- 8 bit I/O ports :: either input or output port.
   ii. Port C-3 lines for handshaking signals &
   2 lines for I/O functions.
   iii. input and output data are latched.
   iv. interrupt logic is supported.

3. Mode 2: Bidirectional I/O data transfer.

   allows bidirectional data transfer.

   make use of handshaking signals (PC3-PC7) for Port A only.

   both inputs and outputs are latched.
   when peripheral request processor, data is sent from CPU through Port A appears on bus.
   Port C lines are used for simple I/O functions.
   Port B- programmed in Mode 0 or Mode 1.

   PC0-PC2 used for handshaking.

CONTROL WORDS FORMATS

8255 PPI got inbuilt control registers.

also called as control or command word register.

loaded 8 bit pattern decides I/O functions of each port & mode of operation of port.

1. Bit set/ reset mode:

   bit set/ reset control word format is shown below.
   BSR word – written for each bit that is to be set or reset.
   also used for enabling/ disabling interrupt signals by setting/ resetting the associated bits of interrupts.

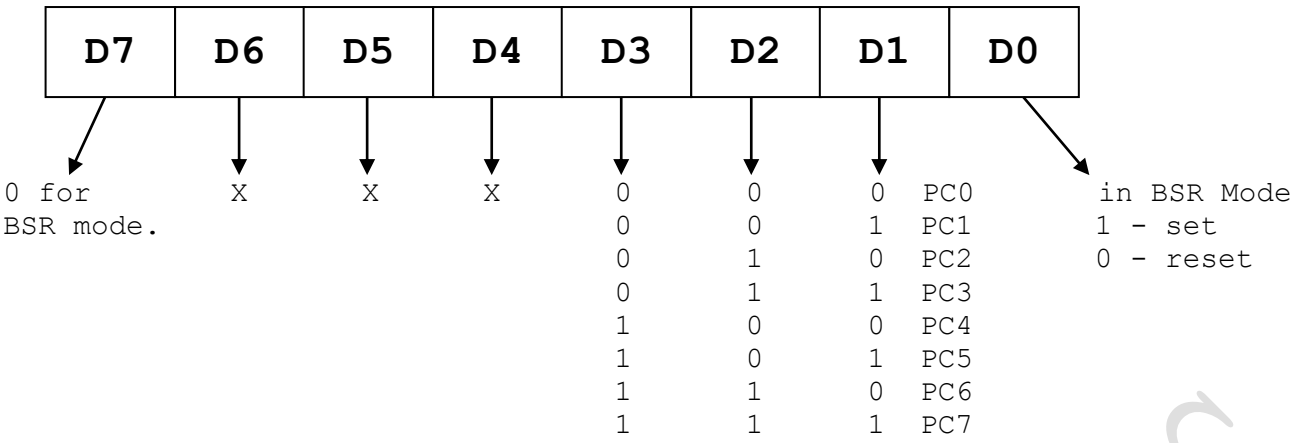


FIGURE 10 BSR CONTROL WORD FORMAT.

2. for I/O mode:

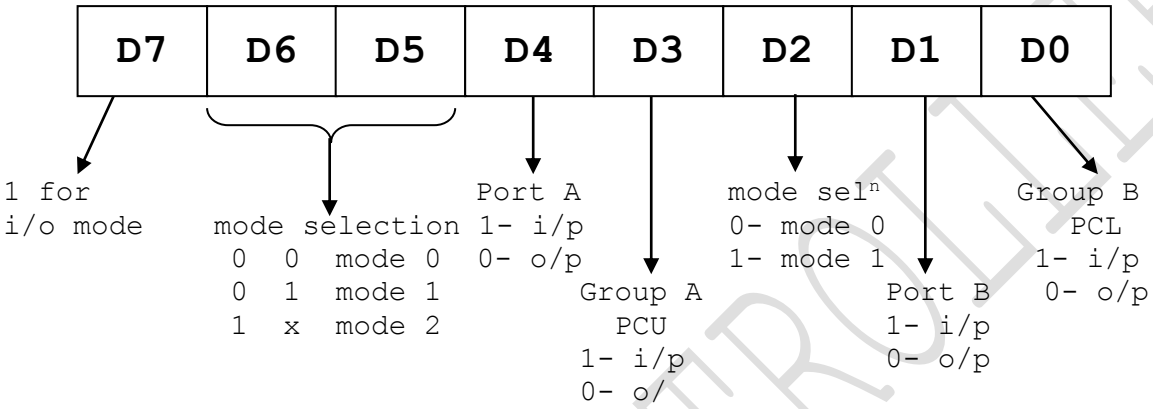


FIGURE 11 BSR CONTROL WORD FORMAT FOR I/O MODE.

I/O PORT EXPANSION USING 8255

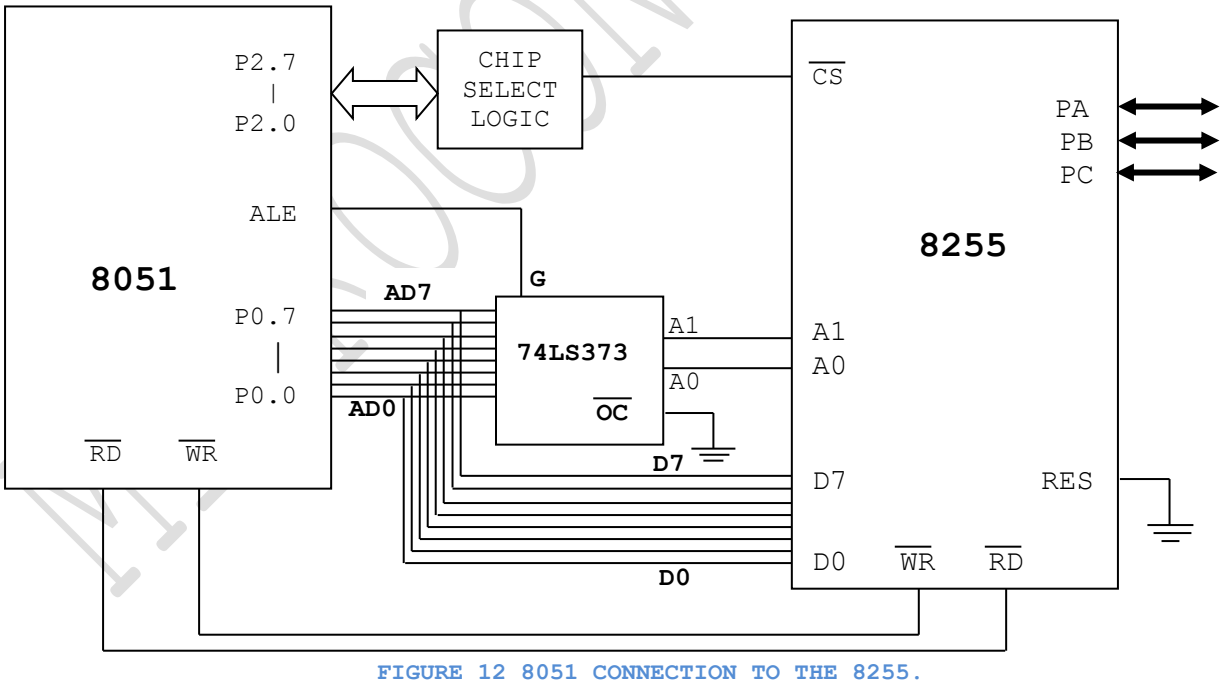


FIGURE 12 8051 CONNECTION TO THE 8255.

as seen earlier, for external memory interfacing to 8051 port 0 & port 2 are used as multiplexed address/data bus & higher order data bus respectively.
if circuit needs on-chip peripherals(e.g. serial I/O & interrupts) then only port is available for I/O.

in such situation, I/O expansions is necessary & it is achieved by using 8255.
data bus of 8255 is connected to the port 0.
address line A_0 & A_1, after latches are connected to A_0 & A_1 of the 8255.

INTERFACING TO µCONTROLLER 8051

1. Interfacing Push button switches(Keys & LEDs)

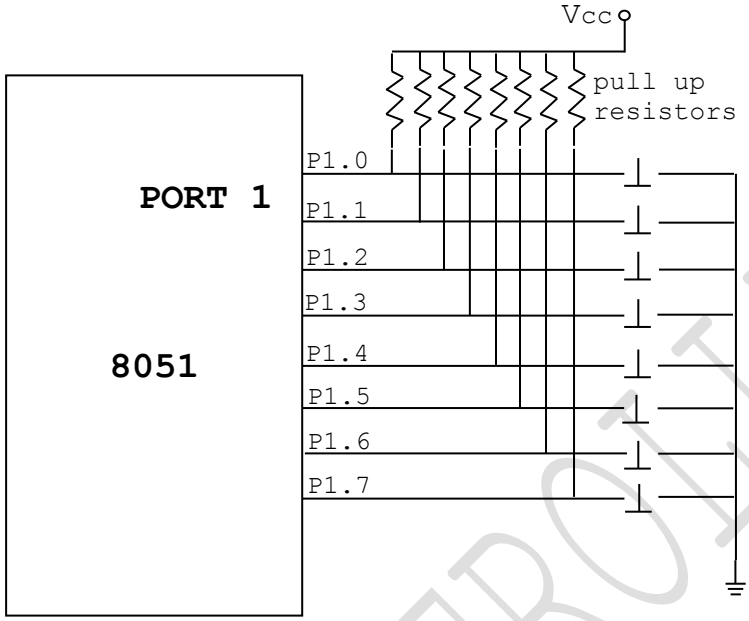


FIGURE 13-A 8051 PORT 1 CONNECTION TO SWITCHES(KEYS).

AS shown in above figure 13-A 8 push buttons are connected to port 1.

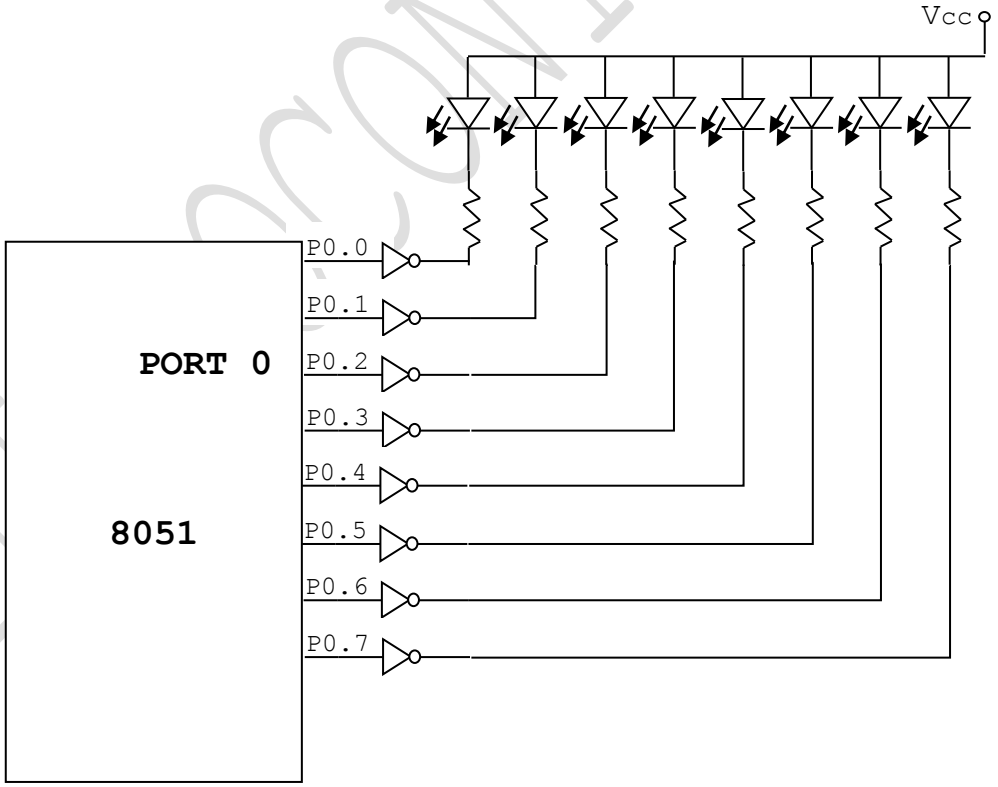


FIGURE 13-B 8051 PORT 0 CONNECTION TO LEDS.

as shown in above figure 13-B 8 LEDs (in common anode configuration) are connected to port 0.
PORT 1- input port
PORT 0- output port

when push button pressed, bounce(make & break) take place ... before firm contact.
sol^{n} to this is wait for 10- 20 sec till key is settled & then key is checked again.

PROGRAMMING STEPS
i. check if key is pressed.
ii. wait for key debounce.
iii. identify the key in binary format.
iv. display the key condition using 8 LEDs.

SAMPLE PROGRAM: key_led.asm

```
 labels      mnemonics                comments
-------------------------------------------------------------------------
            mov p1, #0ffh          ;make p1 as input port.
start:      MOV A, P1              ;read the data from P1.
            cjne a, #0ffh, check   ;key pressed- branched to check1.
            sjmp start             ;branch to start.
check:      acall delay            ;call delay.
            mov a, p1              ;read data from port 1.
            cpl a                  ;complement a.
            mov p0, a              ;send data to LED.
            ajmp start             ;branch to start.
delay:      mov r6, #20h           ;delay routine. load r6 with 20h.
next:       mov r7, #0ffh          ;load r6 with 20h.
here:       djnz r7, here          ;wait until r7 becomes 0.
            djnz r6, next          ;wait until r6 becomes 0.
            ret                    ;return to next instruction.
            end                    ;end address of program.
```

---

2. Interfacing Seven Segment Display

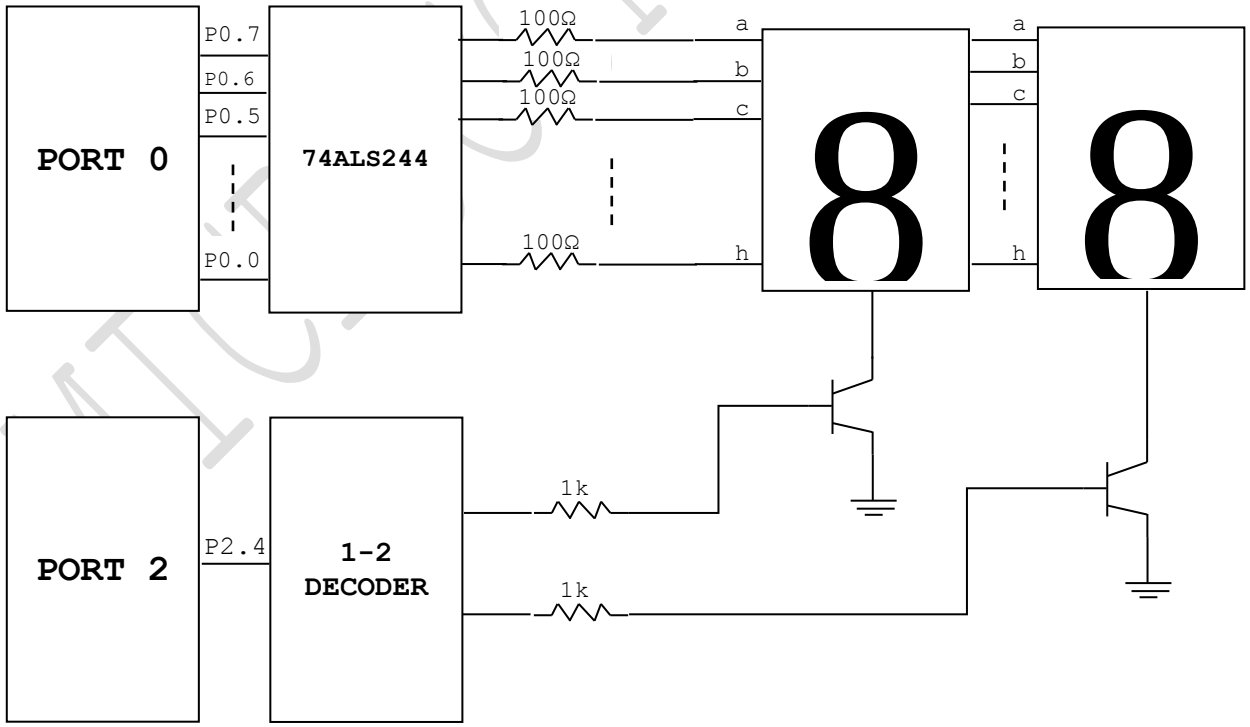


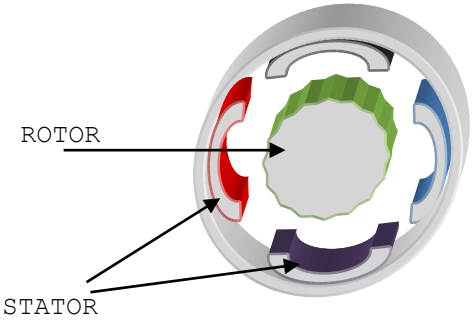
FIGURE 14 8051 PORT 0 CONNECTION TO 7 SEGMENT DISPLAY.

3. Interfacing Of Stepper Motor

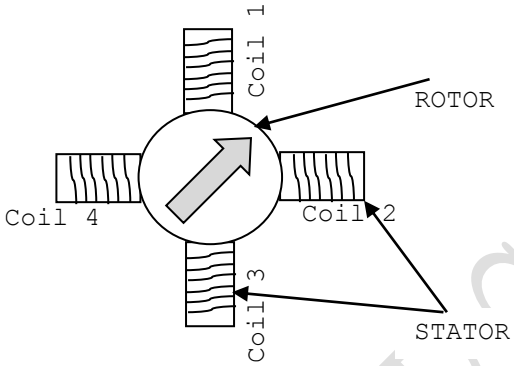
stepper motor: stepping or step motor

rotates in fixed steps.

if rotor rotates 90° in each step(from one pole segment to another)- full step motor.

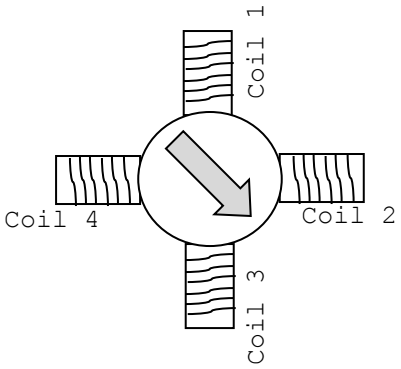


(14-a)

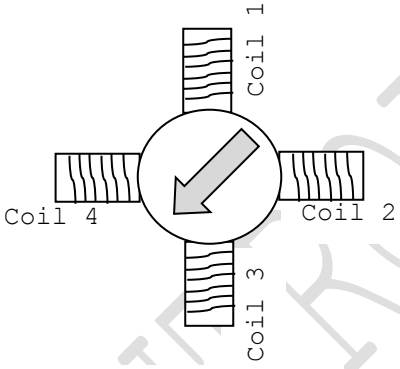


(14-b 1)

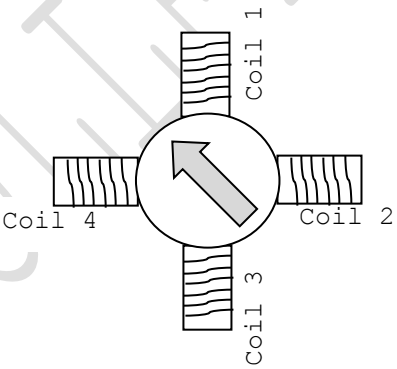
FIGURE 14-a INSIGHT OF STEPPER MOTOR.



(14-b 2)



(14-b 3)



(14-b 4)

FIGURE 14-b FULL STEP STEPPER MOTOR DRIVE SEQUENCE.

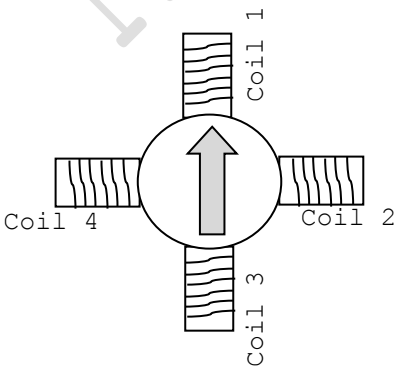
stepper motor may also be operated with half step sequence.

rotor rotates through 45° in each step.

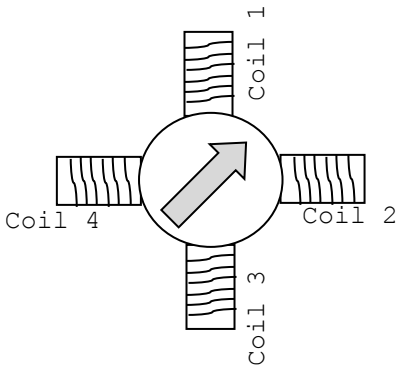
common stepping sequence- 1.8°, 7.5° and 15° etc.

step sequence	No. of steps required for 1 rotation
1.8°	(360°/1.8°=) 200 steps
7.5°	(360°/7.5°=) 48 steps
15°	(360°/15°=) 24 steps

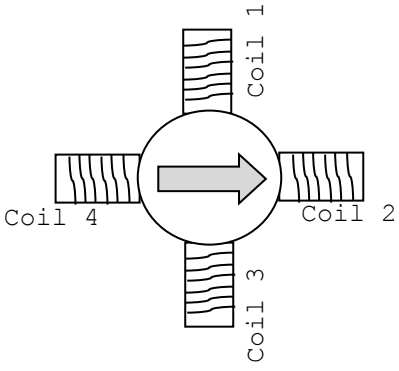
figure below shows the drive sequence for half step stepper motor.



(15.1)



(15.2)



(15.3)

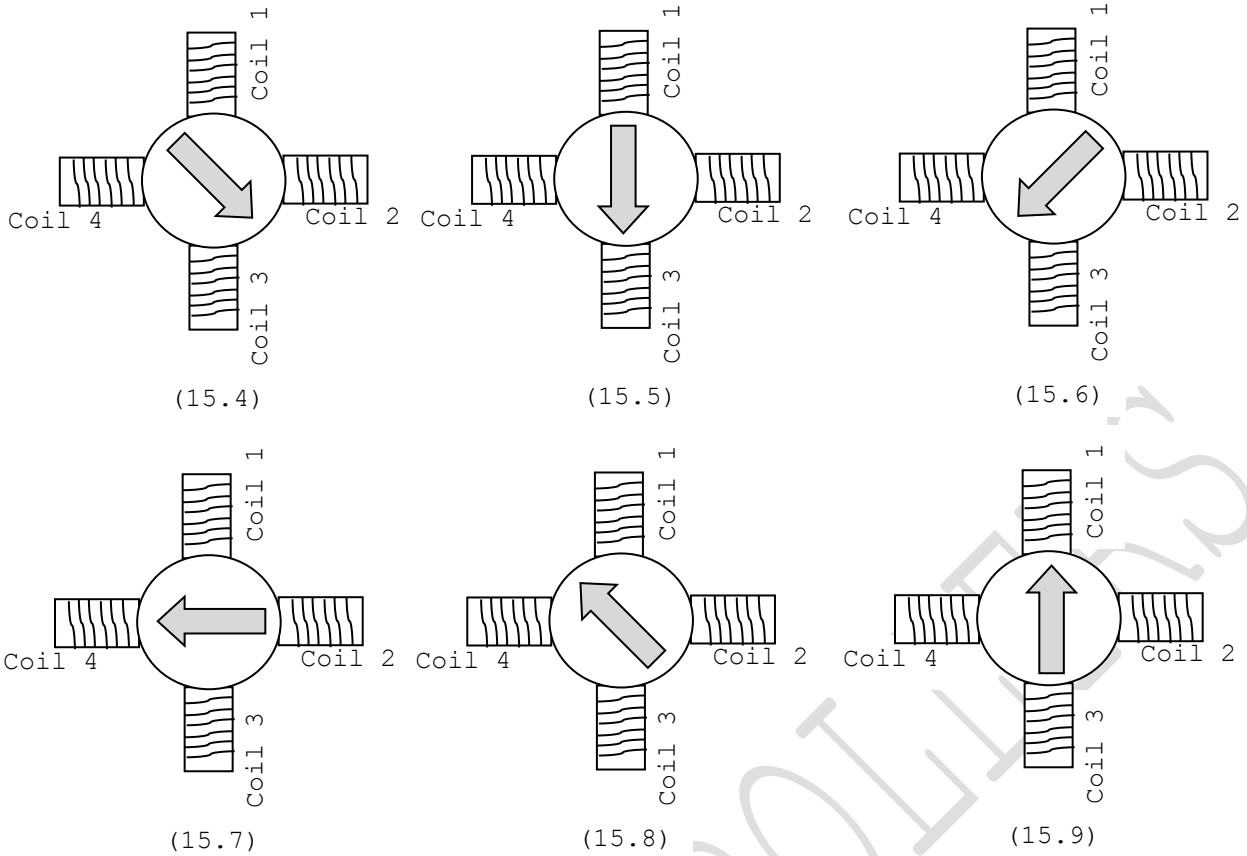


FIGURE 15.1-15.9 HALF STEP STEPPER MOTOR DRIVE SEQUENCE.

stepper motors can be driven directly by transistor to supply high currents.

µC provide drive pattern at the output which causes the motor to rotate.

diodes – flyback diodes which protects transistors from reverse bias.

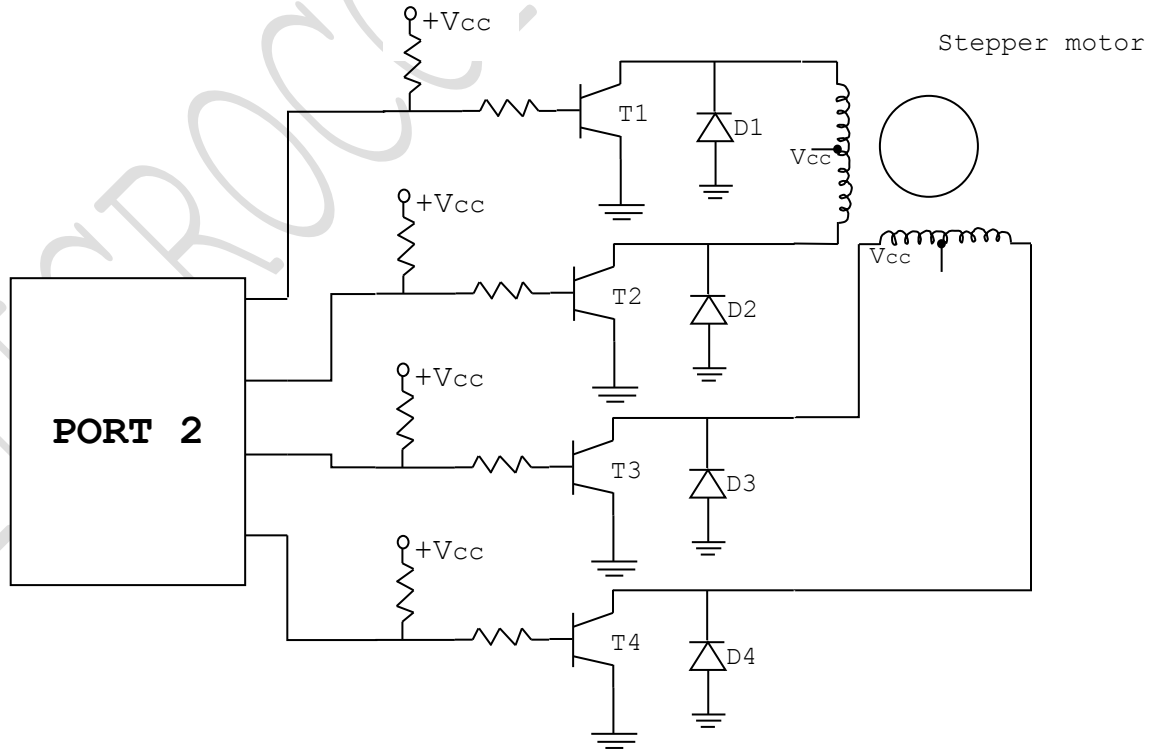


FIGURE 16 STEPPER MOTOR DRIVE CIRCUIT INTERFACING WITH PORT2.

PROGRAMMING STEPS:
i. port 2 is used as output port for interfacing the stepper motor.

SAMPLE PROGRAM: steppermotor_interfacing.asm

```
  Labels          Mnemonics                Comments
-------------------------------------------------------------------------
                  ORG 0000H              ; start address of program.
  LOOP:           MOV P2, #03H           ; load step sequence 03h.
                  ACALL DELAY            ; call delay.
                  MOV P2, #09H           ; load step sequence 09h.
                  ACALL DELAY            ; call delay.
                  MOV P2, #0CH           ; load step sequence 0ch.
                  ACALL DELAY            ; call delay.
                  MOV P2, #06H           ; load step sequence 06h.
                  ACALL DELAY            ; call delay.
                  AJMP LOOP              ; repeat.
  DELAY:          MOV R0, #0FFH          ; load r5 with 0ffh.
  DELAY1:         MOV R1, #0FFH          ; load r7 with 0ffh.
  HERE:           DJNZ R1, HERE          ; wait until r1=0.
                  DJNZ R0, DELAY1        ; wait until r0=0.
                  RET                    ; return to main program.
                  END                    ; end address of program.
```

the step sequence is described below,
depending on no. of steps the step sequence is given below.

a. normal 4 step sequence-

Step	A	B	C	D	Hex Equivalent
1	0	0	1	1	03H
2	1	0	0	1	09H
3	1	1	0	0	0CH
4	0	1	1	0	06H

Clockwise ↓ Anti-clockwise ↑

b. normal 8 step sequence-

Step	A	B	C	D	Hex Equivalent
1	0	0	1	1	03H
2	0	0	0	1	01H
3	1	0	0	1	09H
4	1	0	0	0	08H
5	1	1	0	0	0CH
6	0	1	0	0	04H
7	0	1	1	0	06H
8	0	0	1	0	02H

Clockwise ↓ Anticlockwise ↑

4. Interfacing Relay

µC: pins at output provides Max. 1.32mA current.
relay coil needs around 10mA.
hence µC lacks sufficient driving current for relay.
relay driver – ULN2803 or power transistors employed for this purpose.
figure below shows the connection of relay driver ULN2803 between µC & relay.

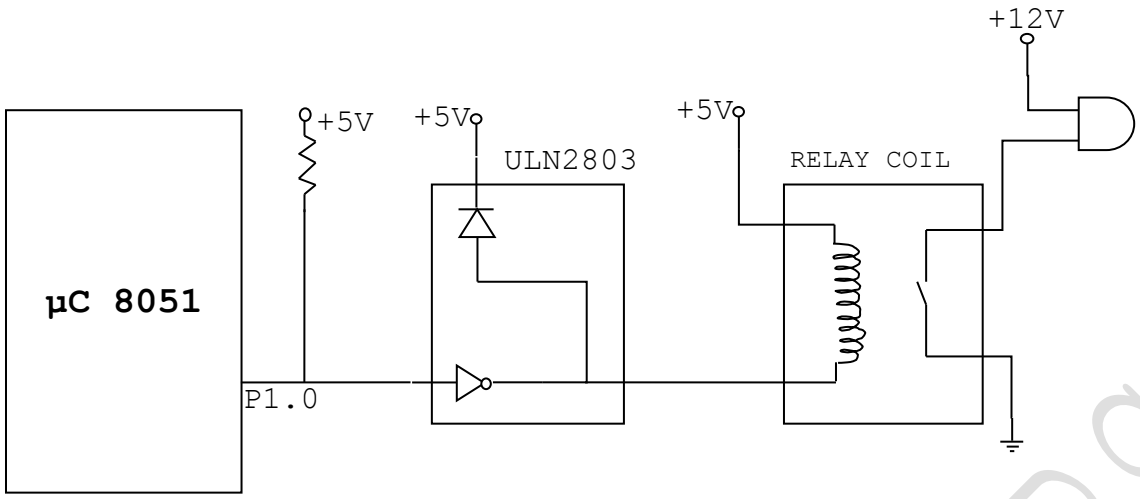


FIGURE 17 RELAY DRIVER & RELAY COIL INTERFACING WITH PORT 1.

solid state relay:

no coil, spring or mechanical contact switch.
no mechanical parts made of semiconductor.
extreamely low input current requirement& small packaging makes it ideal for µC & logic control switching.
widely used in controlling pumps, solenoids, alarms & other power applications.

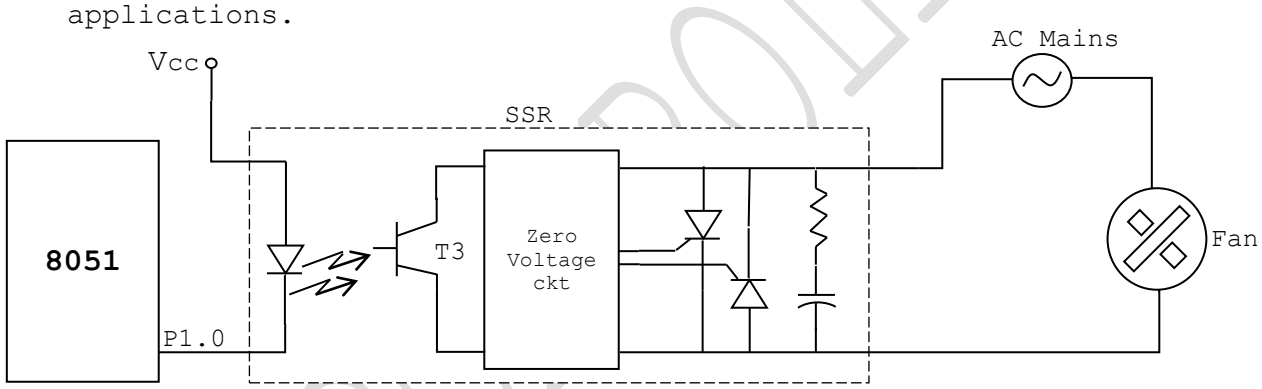


FIGURE 18 FAN CONTROL USING A SOLID STATE RELAY (SSR).

above figure shows control of a fan using a solid state relay(SSR).