UNIT 2 : IO PORT INTERFACING - I

Syllabus :

Pin Diagram and its functioning, Port structure,
IO Interfacing Requirements
Interfacing of : LED
Keys
Seven segment Multiplexed Display
DAC 0808
ADC 0809
Stepper Motor
Relay
Buzzer
Opto Isolator
Design of Data Acquisition System (DAS):
All programs in C Language.

Text Book :

Mahumad Ali Mazidi, Janice Gillispie Mazidi, Rolin. D Mckinlay
"The 8051 Microcontroller & Embedded systems", PHI 2nd Edition

IO Interfacing

All 8051 Microcontrollers have 4 I/O ports each comprising of 8 bits, which can be configured as inputs or outputs.
32 I/O pins available, enabling the microcontroller to be connected to peripheral devices.

Pin Configuration i.e whether the port pin is to be configured as an input (1) or an output (0) depends on its logic state.
In order to configure a microcontroller pin as input it is necessary to apply a logic 1 to appropriate I/O port bit.
Similarly in order to configure a microcontrollee pin as output it is necessary to apply a logic 0 to appropriate IO port bit.

Port Structure:

Port 0:

Address : 80 H

Port 0 is used as Simple I/O port.
Pull Registor is needed when Port 0 is defined as output port because Port 0 is open drain.

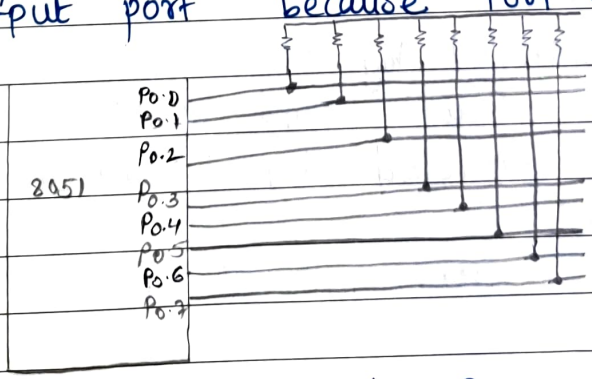
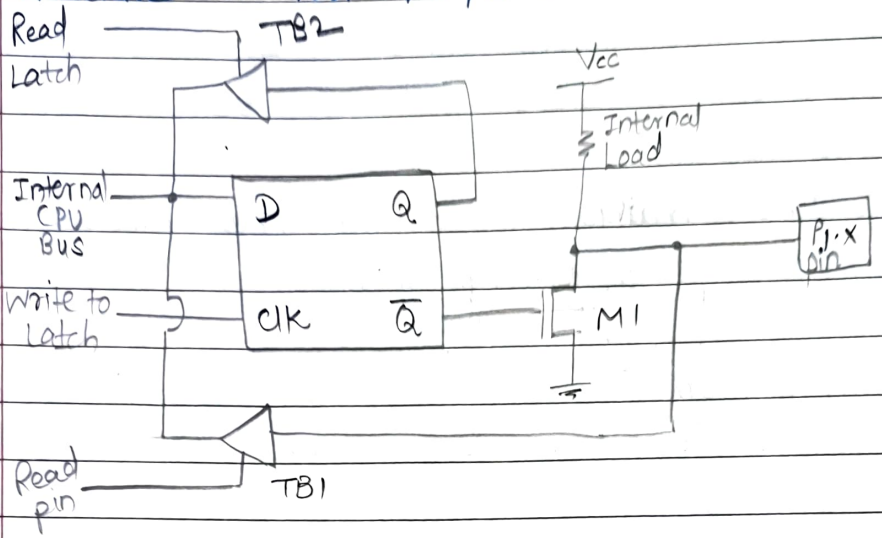


fig: Port0 with Pull up Registor.

Structure of Port 1, 2, 3 of 8051 Microcontroller.



Each I/O port is Connected internally to CPU bus.

The bit latch (one bit in port's SFR) is represented as D flip-flop, which clocks a input from the internal bus in response to "write to latch" signal from CPU

The Q output of flipflop is placed on internal bus in response to "Read Latch" signal from CPU.

The level of port pin itself is placed on the internal bus in response to "read pin" signal from CPU.

Tri state Buffer

TB1 is controlled by "Read Pin"

Read Pin = 1 → Read Data at Pin

TB2 is Controlled by "Read Latch"

Read latch = 1

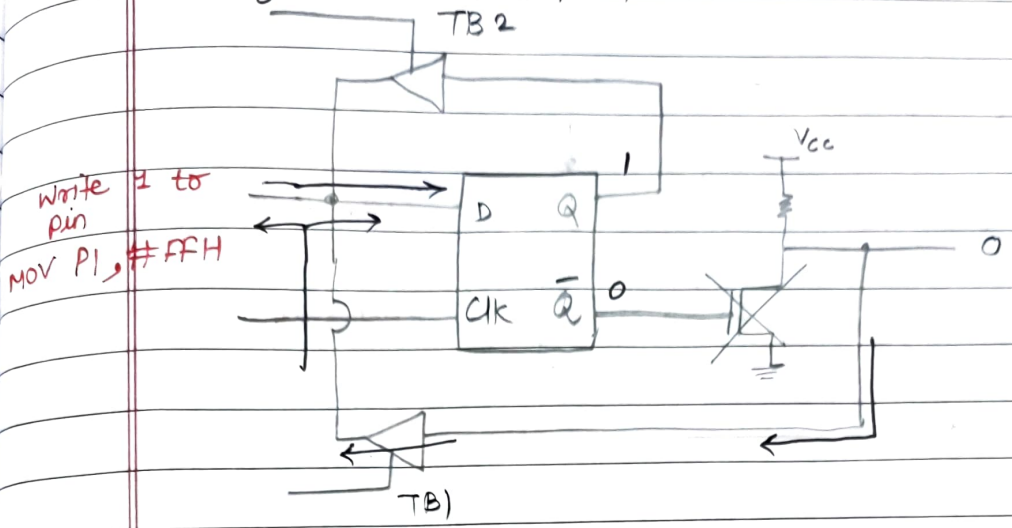
Read Value from internal Latch.

A transistor M1

Gate = 0 open

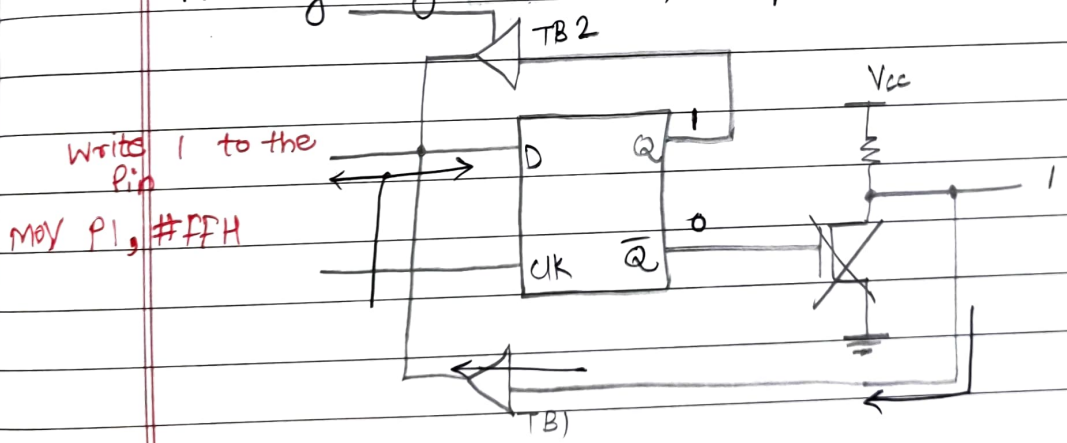
Gate = 1 close

Reading "low" at i/p pin :



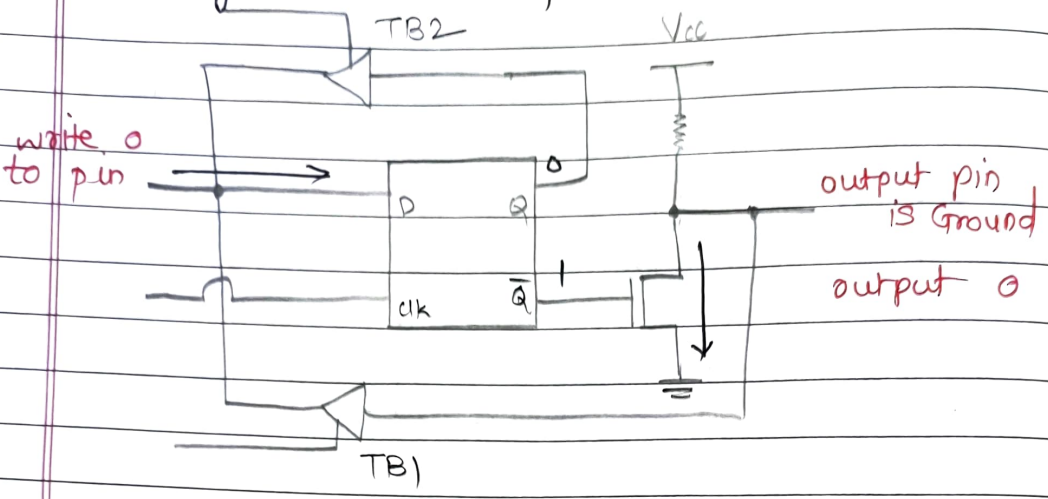
Read Pin = 1
Read Latch = 0
Write to Latch = 1

Reading "High" at input pin :

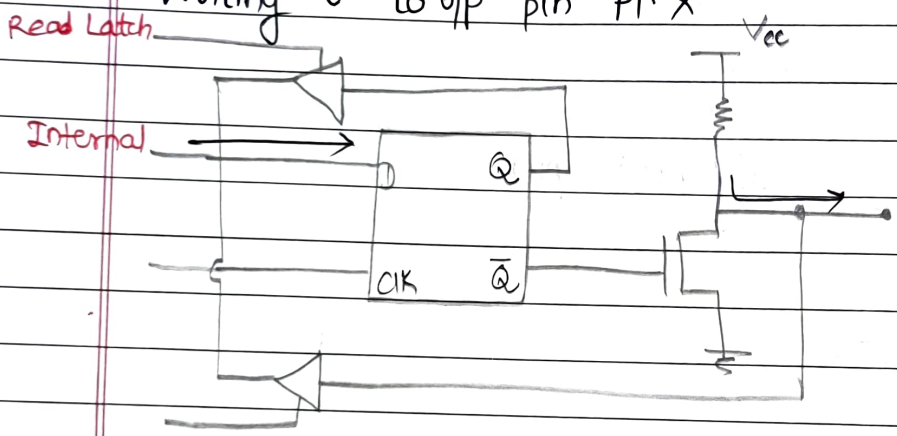


Read Pin = 1
Read Latch = 0
Write to latch = 1

Writing "0" to o/p pin P1.X



Writing 0 to o/p pin P1.X



LED's

LED is semiconductor device.

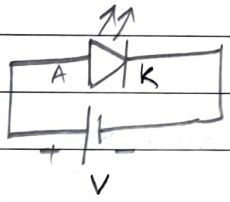
LED's Convert Electrical energy into light energy.

symbol :

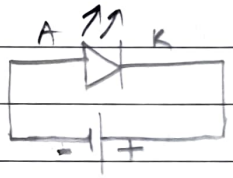
Why LED's ?

Low power Consumption
Small size
fast switching
physically Robust
Long lasting

LED's are available in different colors like Red, Blue, Green, Yellow, white etc.



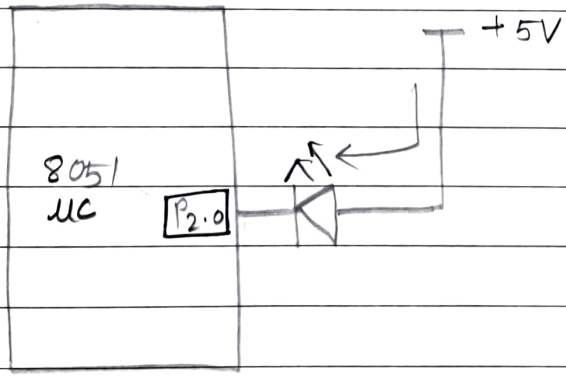
LED ON



LED OFF

Single LED interfacing :

Method 1 :



Common Anode

To make LED = ON
$P_{2.0}$ should be given
0

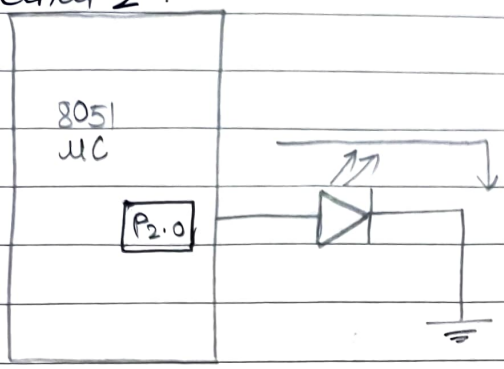
To make LED = OFF
$P_{2.0}$ should be given
1

LED flashing on P2.0

```
#include <reg51.h>

sbit LED = P2^0 ;

void delay()
{
unsigned int i,j ;
    for ( i=0 ; i<1000 ; i++)
     {
       for ( j= 0 ; j< 10,000 ;j++)
       {
        }
     }
}

Void main()
{
 while (1)                    // Do it continuously
{
 LED=0 ;                      // LED ON
 delay();
 LED=1 ;                      // LED OFF.
 delay();
 }
}
```

Single LED interfacing with 8051

Method 2 :



Common Cathode

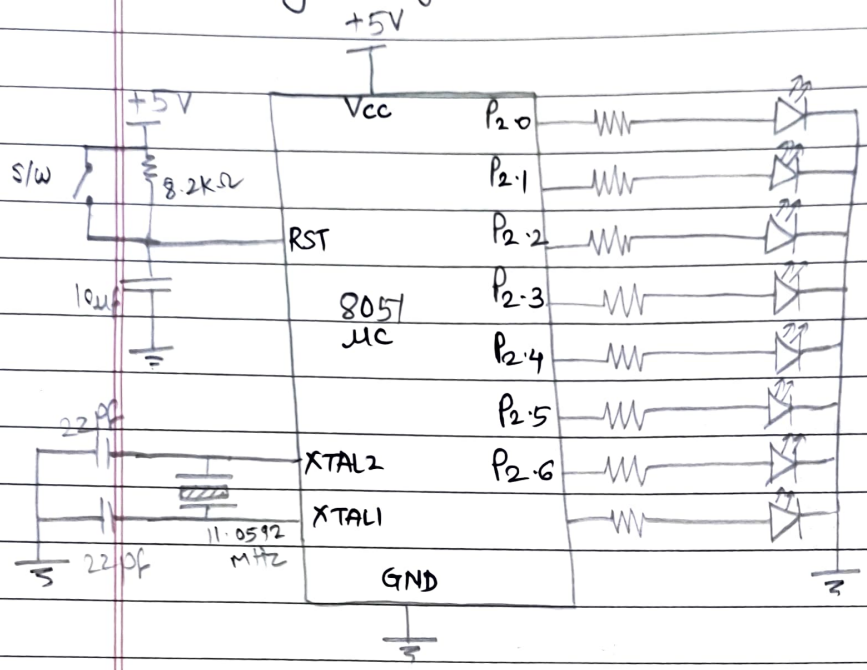
LED flashing on P2.0

```
#include <reg51.h>
sbit LED = P2^0;

Void delay()
{
unsigned int i, j;
for (i=0 ; i<1000 ; i++)
{
 for ( j=0 ; j<10,000, j++)
{
 }
}
}
Void main()
{
while (1)
{
 LED = 1;              // LED ON
 delay();
 LED = 0;              // LED OFF
 delay();
}
}
```

Multiple LED interfacing with 8051

LED's in Common Cathode connected to Port 2

Interfacing Diagram:



LED flashing on P2

```
#include <reg51.h>

Void delay()
{
unsigned int i,j;
for (i=0; i<300; i++)
{
 for (j=0; j<=1000; j++)
{
}
}

Void main()
{
```

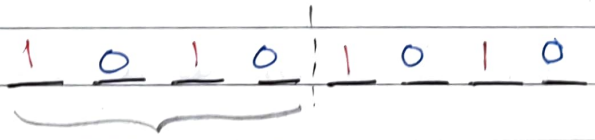
```
while (1)
    {
    P2 = 0xff ;              // LED ON
                                (common cathode)
    delay ();
    P2 = 0x00;
    delay ();                // LED off.
    }
}
```

Multiple LED interfacing with 8051 (Alternate blinking)

Refer same interfacing Diagram:

Logic for Alternate LED Blinking on P2

= 55H



= AA H

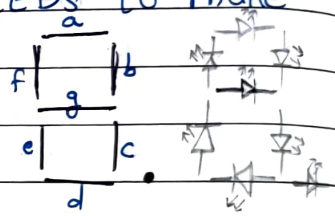
```
#include <reg51.h>
Void delay()
{
unsigned int i,j;
for (i=0 ;i<300 ; i++)
 {
   for (j=0 ;j<1000 ; j++)
    {
    }
 }
}
```

```
Void main ()
{
 while (1)
 {
 P2 = 0X55 ;
 delay ();
 P2 = 0XAA;
 delay ();
 }
}
```

↳ Seven Segment Display:

Seven Segment Display uses seven LEDs to make any Digit.

If all LEDs are ON it shows Digit 8.



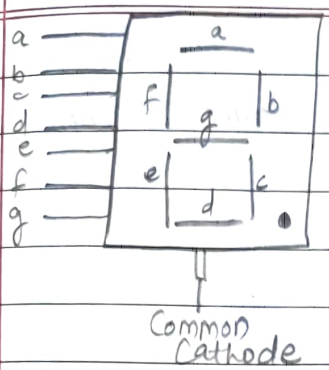
There are Two types of SSD:

Common Cathode:

Cathode of all LEDs is given as a common pin. The Anode is Connected to port pins Common Cathode method requires port pins to source Large current. But 8051 cannot Source current beyond 2mA.

Common Anode:

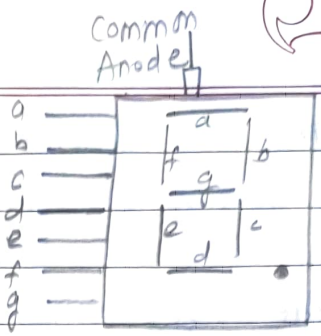
Anode of all LEDs are given as common pin. In this case Cathode is connected to port pins. Hence port pin have to sink 20 mA current. Hence we can use Common anode seven Segment Display and Connect to 8051.



0 → OFF LED

1 → ON LED

fig: Common Cathode LED SSD

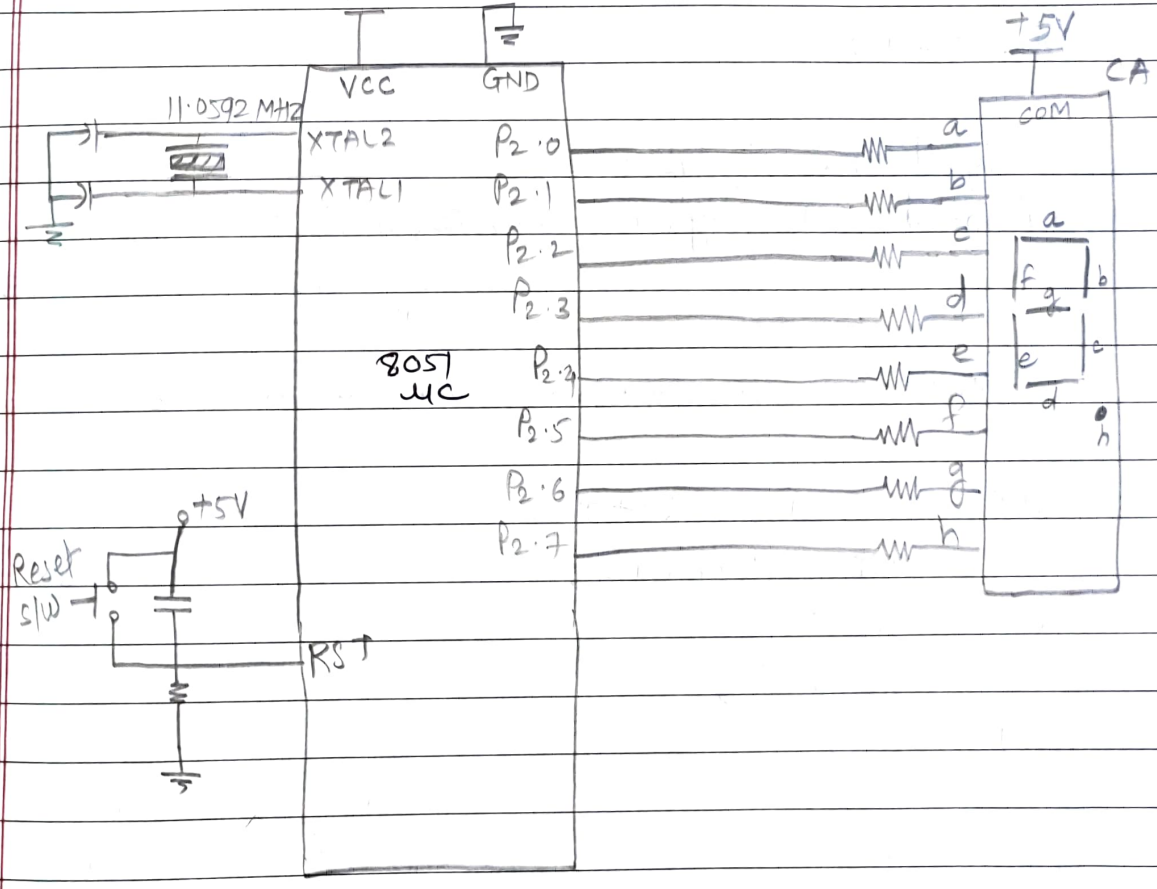


0 → Glow LED

1 → OFF LED

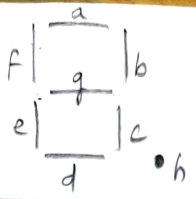
fig: Common Anode LED SSD

7 segment Display Interfacing Diagram (common Anode)



Program to Display 0 to 9 Digits Continously on Seven Segment Display (Decade Counter) Connected to P2

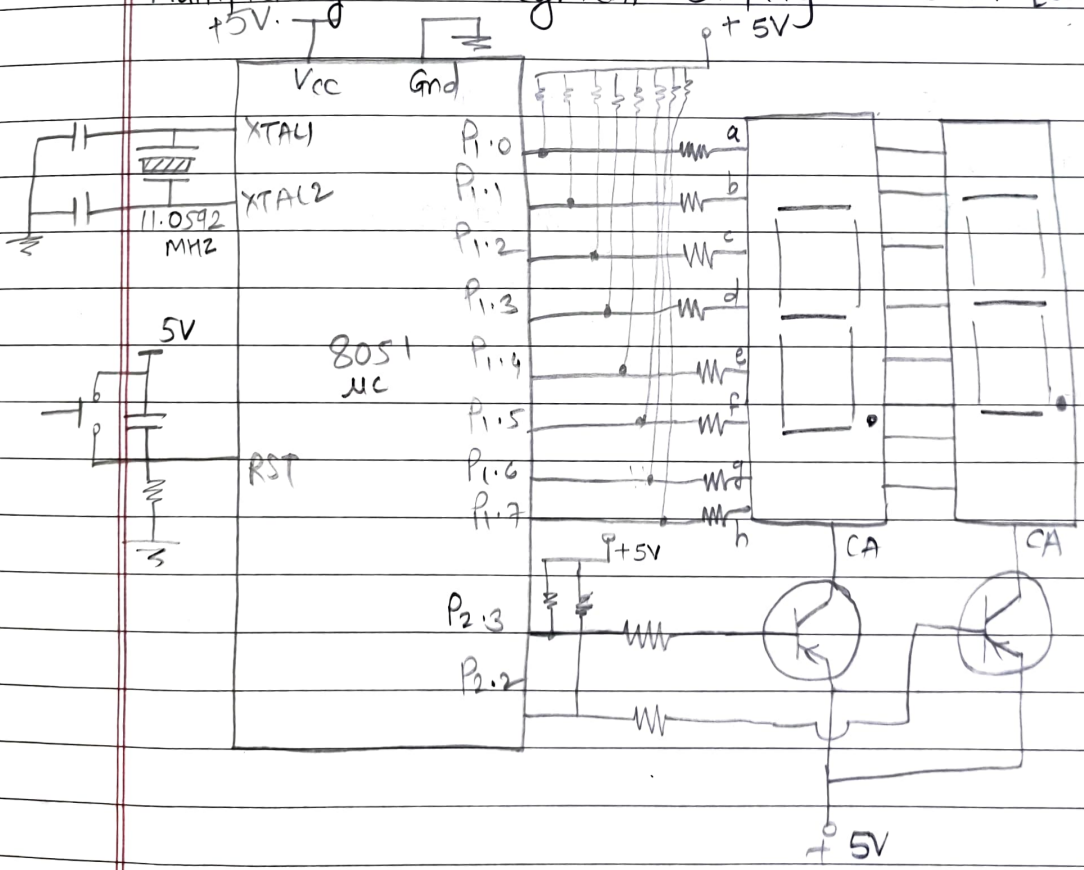
```
#include <reg51.h>

Void delay()
{
int j;
for (j=0 ; j<=1000; j++)
 {
 }
}

Void main()
{
Unsigned char i, dat;
Unsigned int digit[] = {0xC0, 0xF9, 0xA4, 0xB0, 0x99,
                        0x92, 0x82, 0xF8, 0x80, 0x90}

While (1)
{
for (i=0; i<10; i++)
 {   dat = digit[i];
     P2 = dat;
     delay();
 }
}
}
```



Common Anode
0 → ON
1 → OFF

Digit	h	g	f	e	d	c	b	a	Hex Value
0	1	1	0	0	0	0	0	0	C0
1	1	1	1	1	1	0	0	1	f9
2	1	0	1	0	0	1	0	0	A4
3	1	0	1	1	0	0	0	0	B0
4	1	0	0	1	1	0	0	1	99
5	1	0	0	1	0	0	1	0	92
6	1	0	0	0	0	0	1	0	82
7	1	1	1	1	1	0	0	0	F8
8	1	0	0	0	0	0	0	0	80
9	1	0	0	1	0	0	0	0	90

Multiplexing 7 segment Display to 8051 to display "99"



Display 99 on seven segment Display

```
#include <reg51.h>
Void delay()
{
 int i;
 for (i=0; i<1000; i++)
 {
 }
}

Void main()
{

While(1)
{
 P1 = 0x90;                    // value of 9
 delay()

 }
}
```

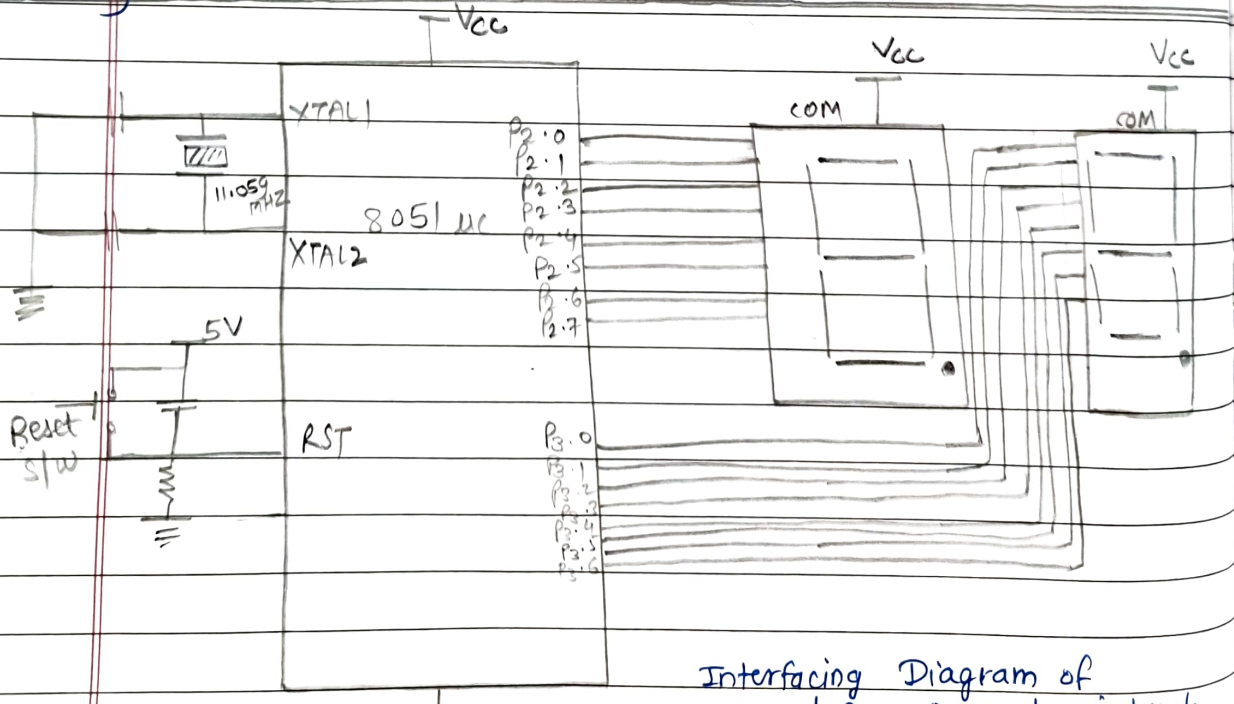


fig: Multiplexed Seven segment Display to count 0 to 99.

Interfacing Diagram of

Count from 0 to 99 using 8051 uC with seven segment Display

```
#include <reg51.h>
void delay();

Void main()
{unsigned int ch[] = {0xc0, 0xf9, 0xA4, 0xB0, 0x99,
                      0x92, 0x82, 0xF8, 0x80, 0x90}
unsigned int i,j;
P3 = 0xc0;
P2 = 0xc0;

while(1)
{
for (j=0; j<10; j++)
{
 for (i=0; i<10; j++)
{
 P3 = ch[i];
 delay();
 }
 if (j!=10)
 P2 = ch[j+1];
 }

if (i==10 && j==10)
 {P3 = 0xc0;
  P2 = 0xc0;
}
}

Void delay()
{ int i;
for (i=0; i<1000; i++)
}
```

ADC 0808/0809.

ADC 0808/0809 with 8 analog channels:

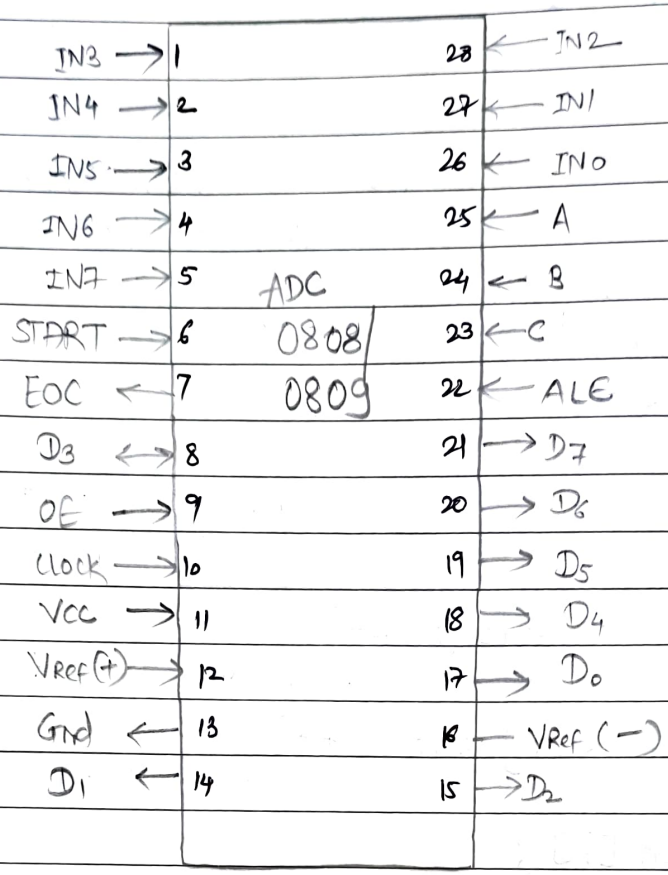


Fig: Pin Diagram of ADC IC 0808/0809

It has 8 analog channels (IN0 — IN7)
User can connect Analog signal input to this 8 channels and IC will convert Analog to Digital signal.
This channels are internally Multiplexed. At a time IC will select one input channel and convert to Digital.
To select channel, IC has address lines (A, B, C)
Each channel has Unique address.

Analog Channel Address.

Channel	C	B	A
IN0	0	0	0
IN1	0	0	1
IN2	0	1	0
IN3	0	1	1
IN4	1	0	0
IN5	1	0	1
IN6	1	1	0
IN7	1	1	1

After placing address of particular channel on address bus user has to latch this address. So for that ALE signal is used. User has to give Low to high pulse to ALE to latch the address.

Now User can send the start of conversion signal. (START). Low to high pulse at START will start Conversion of analog input into Digital.

User can check whether Conversion is Complete or not through EOC [End of Conversion]
EOC → 0 → when Conversion completed
EOC → 1 → when Conversion is going on
User can continously check EOC is 0 or not. If 0 that means Conversion is completed (Analog Data is Converted to Digital)

Digital o/p is stored in the internal Registor of ADC IC.

To place this Digital o/p on Digital bus user has to send OE (output Enable signal)

Low to high pulse should be given to OE signal to place Digital o/p on Digital / Data bus.

ADC has 8 bit Digital bus. ($D_0 - D_7$)
o/p will be available on $D_0 - D_7$.

operation of IC is carried out using CLOCK. clock can be generated using external circuit OR 8051 clock can be connected to ADC 0808/0809 CLOCK pin.

Vcc → +5V suply
Gnd → Ground

Vref (+) → +ve voltage.

Vref (−) → Gnd

Input / output Relation :

8 bit ADC

Steps = 2^8 = 256

Digital o/p

0000 0000
0000 0001
⋮ ⋮
1111 1111

Total 256 steps for Digital o/p.

Step size : how much analog input should change to change Digital o/p by one step.

Step size = $\frac{V_{ref}}{2^8}$ consider $V_{ref} = 2.56$ ⇒ $\frac{2.56}{256} = 10\,mV$.

if i/p analog voltage change by 10 mV, Digital o/p will change by 1 step.

o/p in decimal = $\frac{Vin \times 256}{Vref}$

Assume

Analog i/p voltage = 1 V Vref = 2.56 V

o/p = $\frac{1 \times 256}{2.56}$ = $(100)_{10}$

= $(0110\ 0100)_2$ = 64H

(6 4)

Interfacing Diagram of ADC 0808/0809 to 8051 uc.

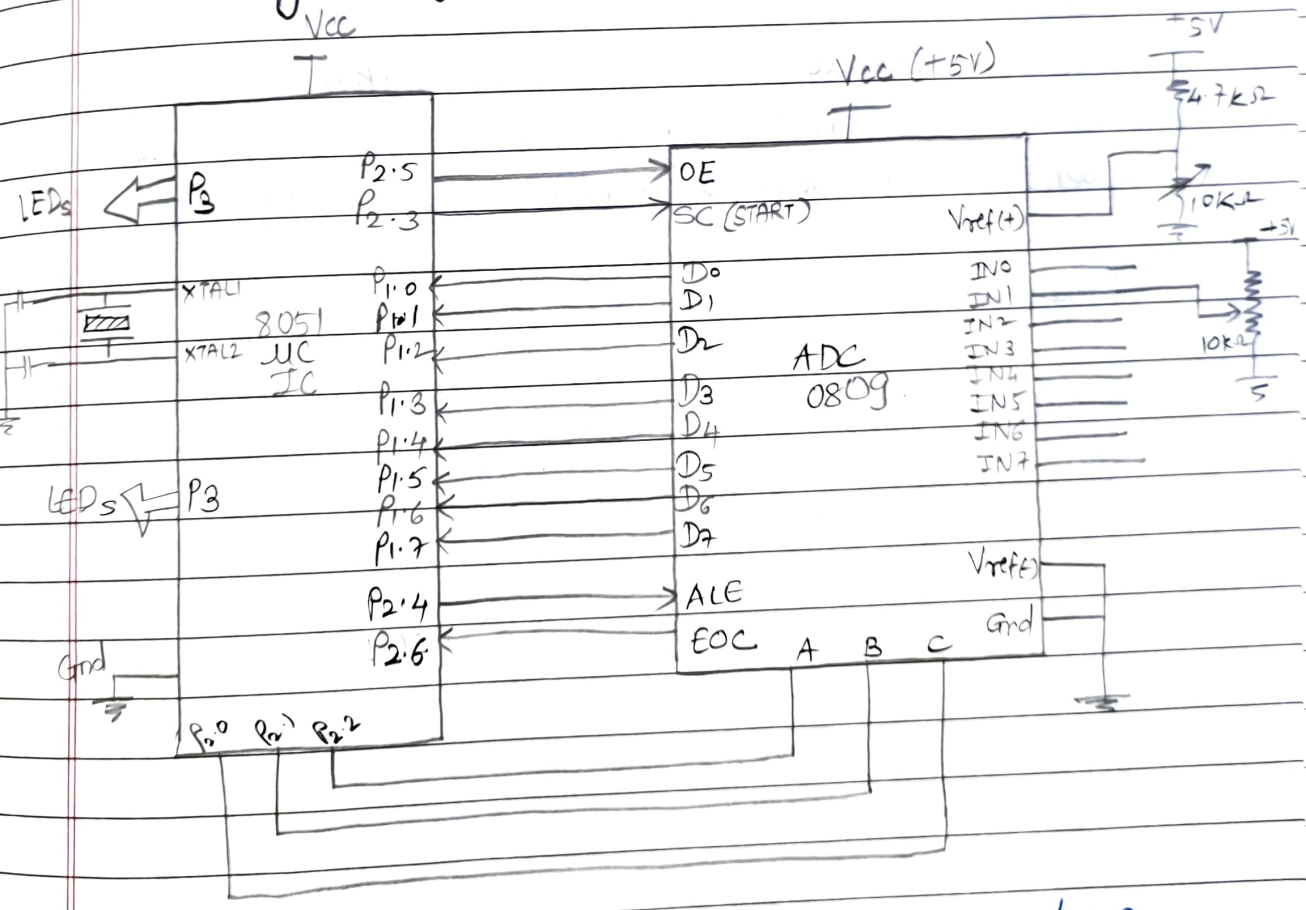


fig: 8051 Connections to ADC 0808/0809.

Algorithm:

1) Send logic 1 at Port 1 all terminals and P2.5 pin (This pins are used as i/p pins to 8051)

2) Connect analog i/p voltage at IN1
3) Send address of channel on address line (A, B, C) i.e 001 (channel 1)
4) Send ALE to latch address (L-H)
5) Send Start of Conversion signal (L-H)
6) Wait for EOC. IF EOC is 0 then go to next step.
7) Send Output Enable (OE) signal (L-H)
8) Read Converted Digital data from P1 and send it on port 3. where LEDs are connected.

Program:

```
#include <reg51.h>
sbit ALE  P2^4;
sbit OE   P2^5;
sbit SC   P2^3;
sbit EOC  P2^6;
sbit A = P2^2
sbit B = P2^1
sbit C = P2^0

sfr mybyte = P1...;
sfr senddata = P3;          // LED's are connected
Void dataconversion (unsigned char);
Void display (unsigned char);
Void delay ();
```

classmate
Date
Page

② Void main ()
{
Unsigned char Value ;
A = 0 ;
B = 0 ;
C = 0 ;
ALE = 0 ;
OE = 0 ;
EOC = 1 // i/p pin
mybyte = 0XFF // i/p port.

while (1)
{
A = 1 ; // select analog i/p channel.
IN1 = 001
B = 0 ;
C = 0 ;
ALE = 1 ; // High pulse on ALE
delay ();
SC = 1 ; // Start Conversion
delay ();
ALE = 0 ; // Low pulse on ALE
SC = 0 ; // Low pulse on SC
while (EOC == 1); // wait for data Conversion
while (EOC == 0);
OE = 1 ;
Value = mybyte ;
senddata = value ;
OE = 0 ; // H - L on OE

}
}

② void delay ()
{
int j ;
for (j=0; j<1000; j++)
{
}
}

DAC

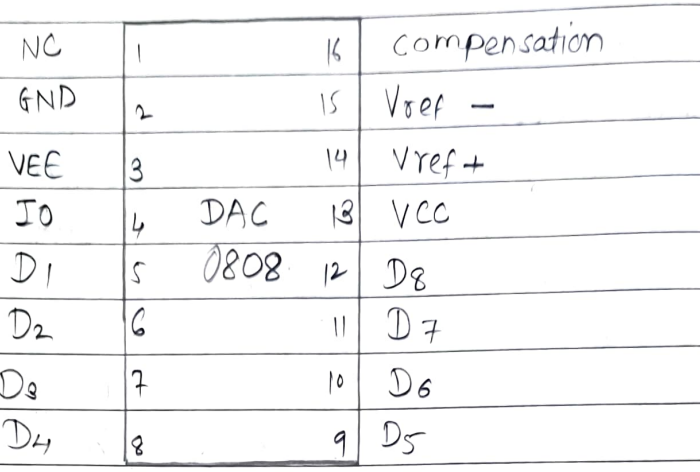


fig: pin diagram of DAC 0808.

$D_0 - D_7$: Digital input

D_0 D_1 D_2 D_3 D_4 D_5 D_6 D_7

MSB LSB

$D_0 - D_7$ are Connected to Microcontroller

Analog o/p will be available on pin 4 of DAC 0808

Analog o/p is in the form of Current.

Externally we need to connect current to voltage converter which Converts output current in voltage We obtain voltage analog signal

Pin 2 Ground

Pin 13 VCC (+ve supply of 5V)

Pin 3 VEE (-ve Supply, Generally Connected to Ground)

Pin 14 Vref + (positive reference voltage)
+5 or +10 V can be applied
The o/p c/n depends on value of Vref +

Pin 15 Vref − (negative reference voltage)
Generally Connected to Ground

Pin 16 Compensation
Capacitor can be Connected at this pin to Reduce noise

Pin 1 Not connected.

Input output Relation:

$$I_o = I_{ref}\left[\frac{D_0}{2} + \frac{D_1}{4} + \frac{D_2}{8} + \frac{D_3}{16} + \frac{D_4}{32} + \frac{D_5}{64} + \frac{D_6}{128} + \frac{D_7}{256}\right]$$

$$V_o = I_o \times R \quad (R \rightarrow 2.5\,k\Omega)$$

Date
Page

Interfacing DAC 0808 to 8051

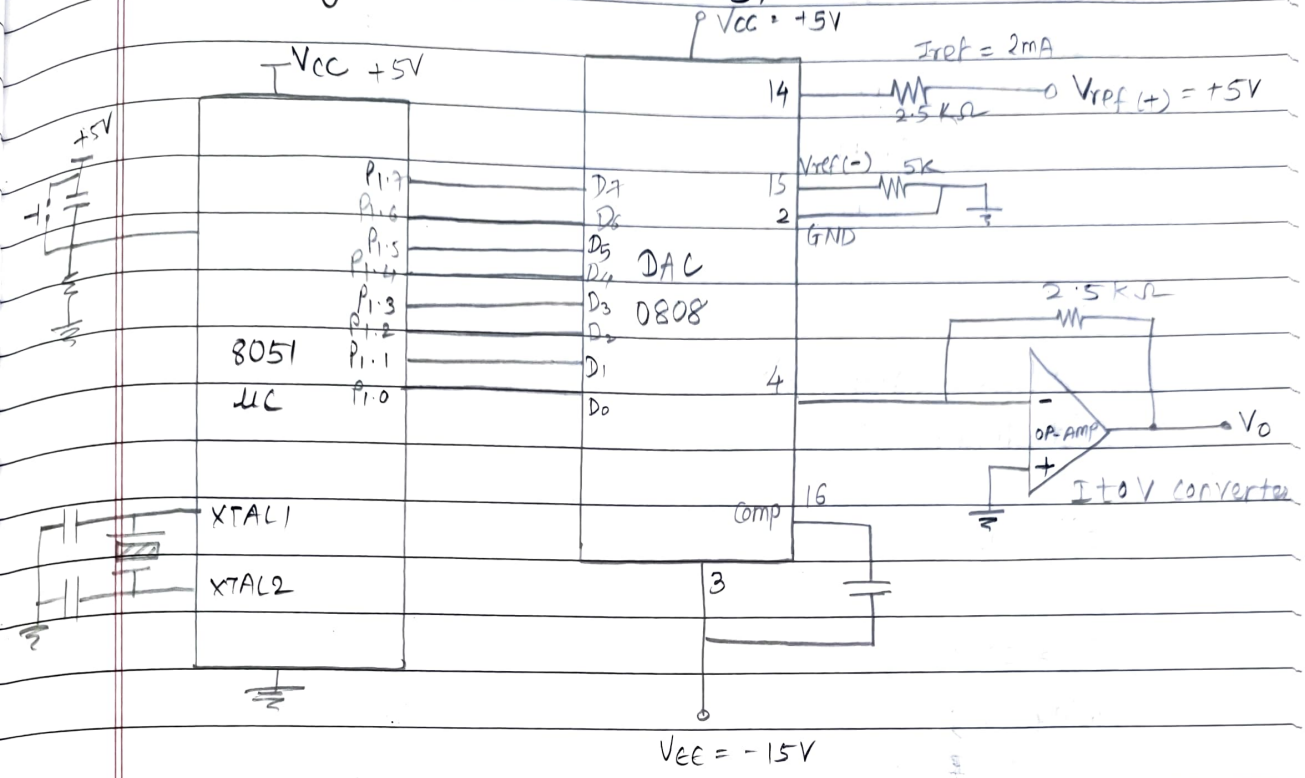
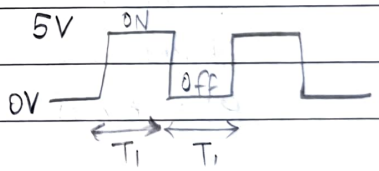


fig: Interfacing Diagram of 0808 DAC to 8051 Microcontroller

Program to Generate square wave:




```
#include <reg51.h>
void delay();
Void main()
{

while(1)
{
P1 = 0xff;
delay();
P1 = 0x00;
delay();
}
}
```

```
Void delay()
{int i;
 for(i=0; i<1000; i++)
  {
  }
}
```

if i/p 00H ⇒ I_o = 0A

$V_o = 0 \times 2.5k = 0V$

if i/p ff ⇒ $I_o = 2mA\left[\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} + \frac{1}{64} + \frac{1}{128} + \frac{1}{256}\right]$

$I_o = 2mA$

$V_o = 2.5k \times 2mA = 5V$

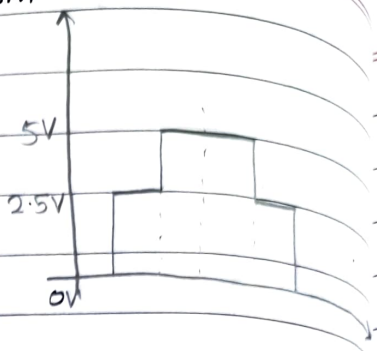
Program to generate staircase waveform:

Logic:

for 0V = 00H

for 2.5V = 7FH

for 5V = ffH.




```
#include <reg51.h>
Void delay();

Void main()
{
P1 = 0x00;
delay();
P1 = 0x7f;
delay();
P1 = 0xff;
delay();
delay();
P1 = 0x7f;
delay();
}

Void delay()
{
int i;
for (i=0; i<1000; i++)
{
}
}
```

Relay :

Relay is an electrically controllable switch.

Microcontroller use relays to control a heavy electrical load.

Relay allows isolation between high power and low power sections of a system having different voltage sources.

Operating voltage : 5V, 6V, 12V, 24V

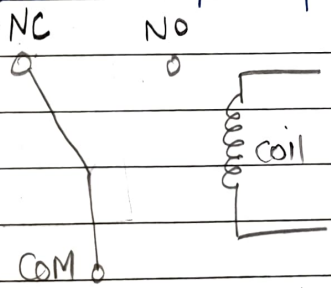
Basic Types of Relay :

Electromechanical Relays :

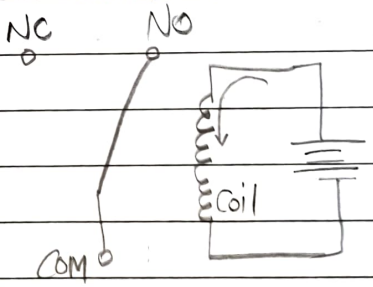
- They have coils, spring and contact
- Switching is low
- Generates noise.

Solid State Relays :

- Uses semiconductor devices for switching
- No movable parts and higher life cycle.
- faster switching time
- use of optocouplers provide electrical isolation & control



coil Not Energised
Relay off



coil is Energised.
Relay ON

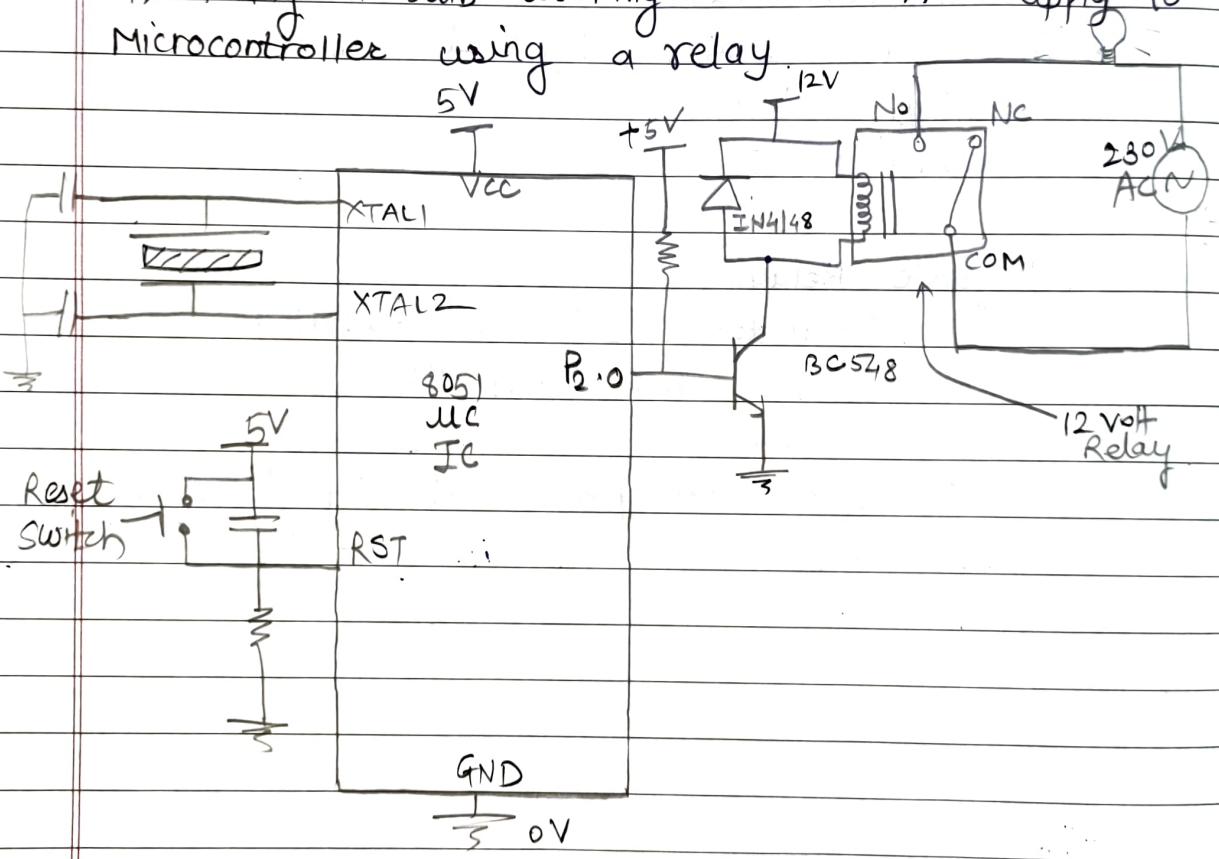
coil : Energise when given voltage
COM : Common Terminal
NO : Normally open Terminal
NC : Normally Close Terminal

Relay off → COM connected to NC
Relay ON → COM Connected to NO.

why driver circuit is needed for interfacing Relay?

A Microcontroller is not able to supply current required for the proper working of relay.
The back emf produced by relay coil can damage port pins.
freewheeling diode is used to supress back emf.
Driver circuitry can be implented using power transistor & Diode, ICs

Interfacing a bulb working on 230V AC supply to Microcontroller using a relay.



When P2.0 = 1 ⟶ 1 on the Base of (npn) Transistor make Transistor ON, this will connect coil terminal to Ground → Relay gets supply and coil gets energised → As a result COM terminal breaks from NC and gets attracted to NO terminal & Bulb gets power. (ON)

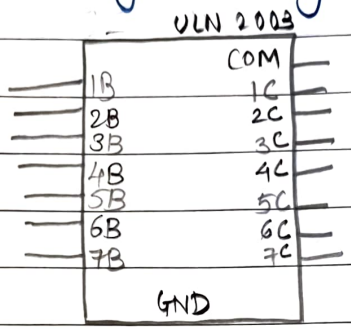
when $P2.0 = 0$, transistor OFF → disconnect coil from Ground so coil gets deenergised and COM terminal which is connected to NO terminal will get attracted back to NC → this disconnect the bulb from one end of power supply hence bulb gets turned off.

C program:

```
#include <reg51.h>
void delay ()
sbit  relay = P2^0;

Void main ()
{
  relay = 0 ;                    // P2.0 as output port.
  while (1)
  {
   Relay = 1 ;                   // Relay ON → Bulb ON
   delay ()
   relay = 0 ;                   // Relay off → Bulb off
   delay ();
  }
}

void delay ()
{
 int i,j;
 for (i=0; i<10; i++)
  {
   for (j=0 ; j<1000 ; j++)
    { }
  }
}
```

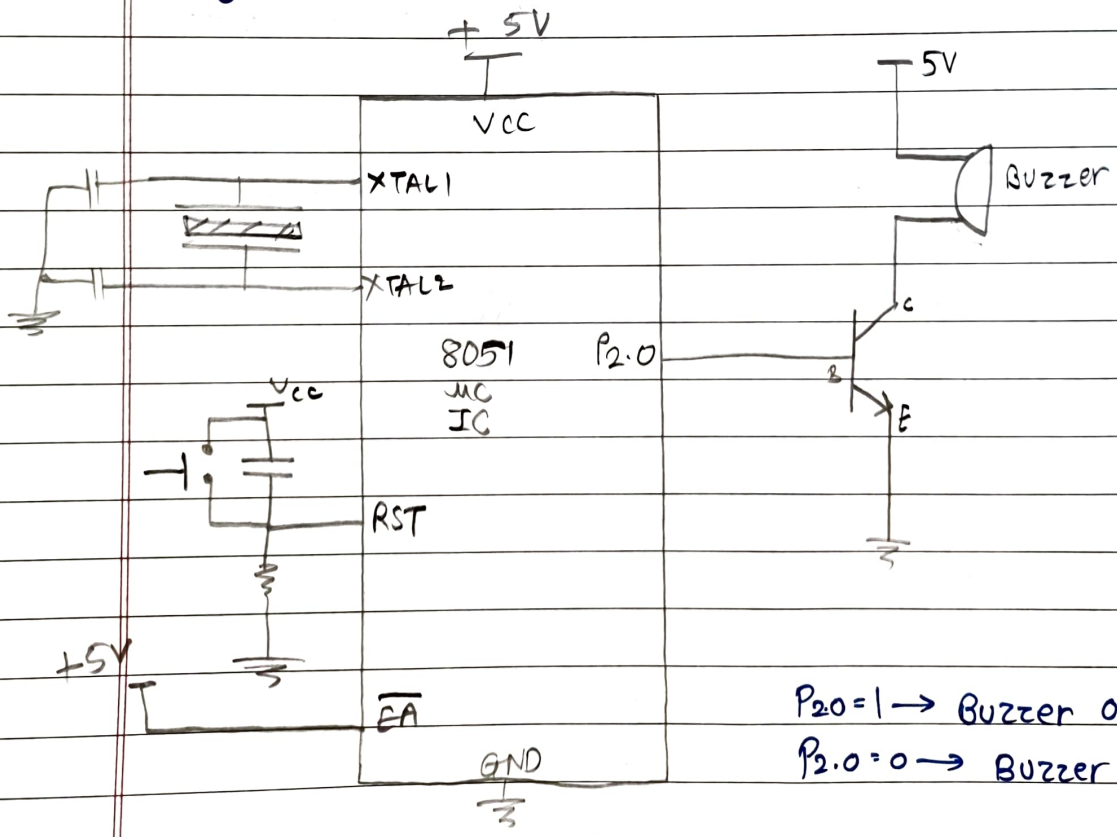
Relay interfacing with 8051 using ULN2003 Driver (Refer PPT for Diagram)

ULN 2003 - Seven relays can be interfaced.
ULN 2803 - Eight relays can be interfaced.



Buzzer:

A Buzzer is an audio signalling Device which may be mechanical, electromechanical or piezoelectric

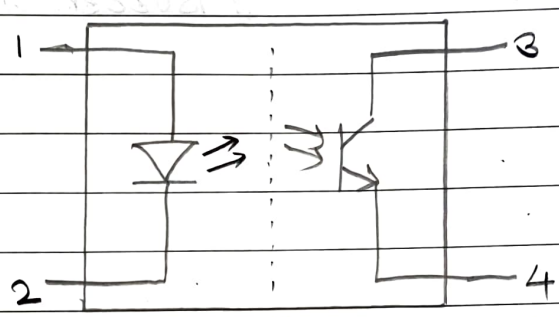


P20 = 1 → Buzzer ON
P2.0 = 0 → Buzzer off

```
#include <reg51.h>

void delay();
sbit pin_buzzer = P2^0;

void main()
{while(1)
 {
  pin_buzzer = 0;          // Buzzer off
  delay();
  pin_buzzer = 1;          // Buzzer ON.
  delay();
 }
}
void delay()
{
 int i,j;
for (i=0 ; i<20; i++)
{
for (j=0 ; j<1000; j++)
{ }
}
}
```

Opto. Isolator :

In electronics, an opto isolator are called an optocoupler, photocoupler or optical isolator is a component that transfers electrical signal between two isolated circuits by using light. Opto Isolator prevents high voltages from affecting the system receiving the signal. Commercially available opto isolators withstand i/p to o/p voltages upto 10 KV & voltage transients with speeds upto 10 KV/μs.



A common type of opto isolator consist of an LED and phototransistor.
other type of source - sensor combination include LED - photodiode, LED - LASCR, lamp - photoresistor

Usually opto Isolator transfer digital signals.

Rating of opto Isolator depends on output voltage drive capacity of opto Isolator. (30V, 70V & 80V are commonly used O/p voltage Rating)

4N25, 4N26, ILD74, PC817 are mostly used opto Isolator.

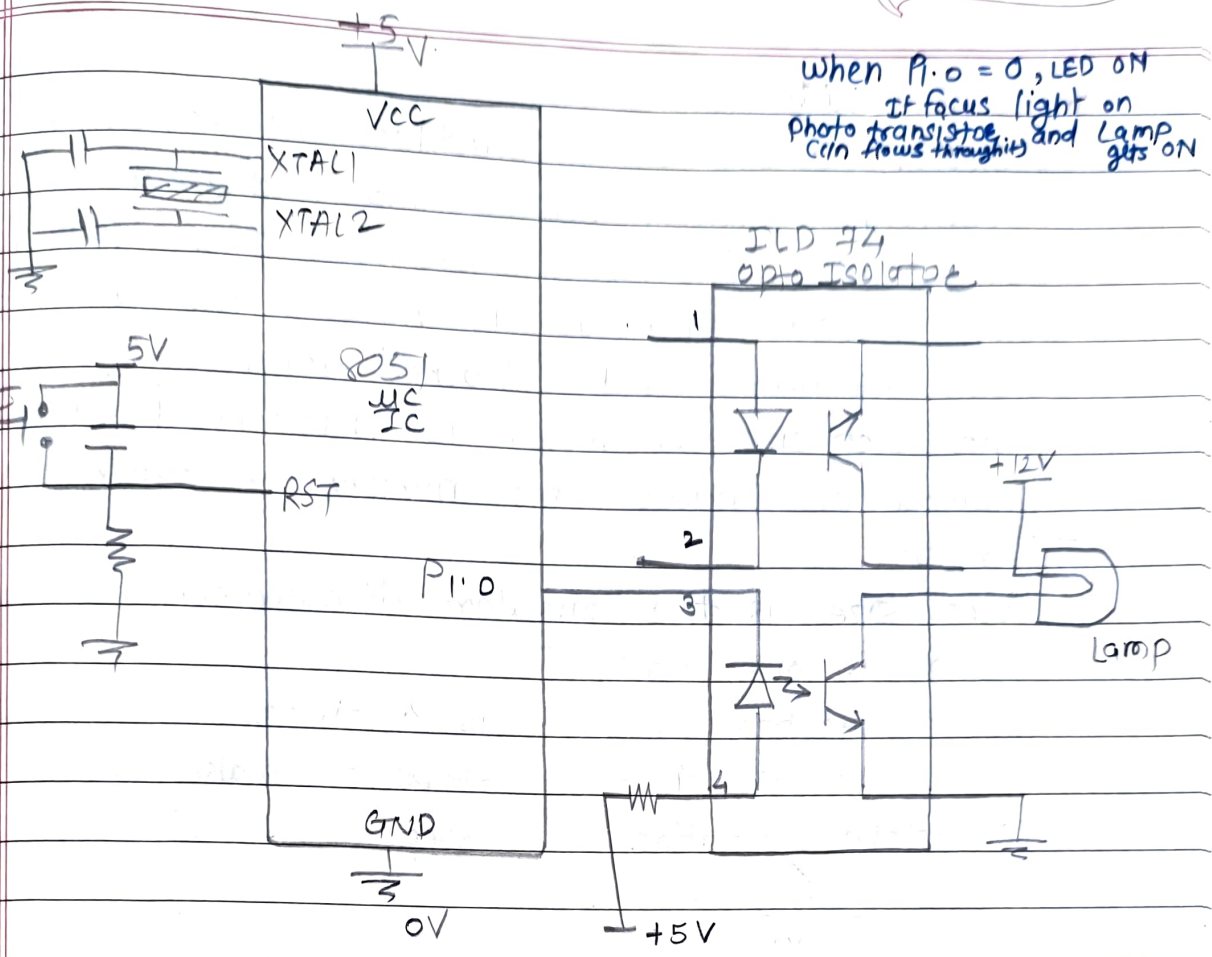


fig: Interfacing Diagram of opto Isolator with 8051

```
# include <reg51.h>
void delay();
        sbit  opt_iso = P1^0;
void main()
{
 while(1)
 {
  opt_iso = 0;          // lamp ON
  delay();
  opt_iso = 1;          // lamp off
 delay();
 }
```

```
void delay()
{
int i;
for(i=0;i<1000;i++)
{
}
}
```

Stepper Motor:

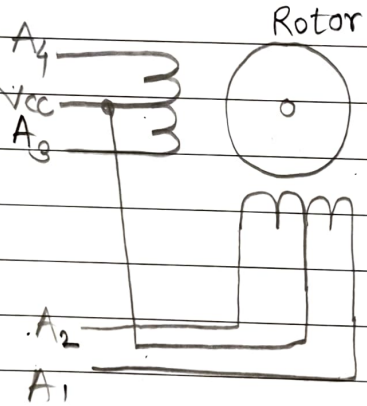
Stepper Motor is a brushless Motor which converts electrical pulses into Mechanical rotation. It rotates in steps according to the input pulses. A stepper motor usually have a number of field coil (phases) and a toothed rotor.

The step size of motor is determined by the number of phases and number of teeth on the rotor.

Step size is the angular displacement of the rotor in one step.

If a stepper motor has 4 phases and 50 teeth, it takes $50 \times 4 = 200$ steps to make one complete rotation. so step angle will be

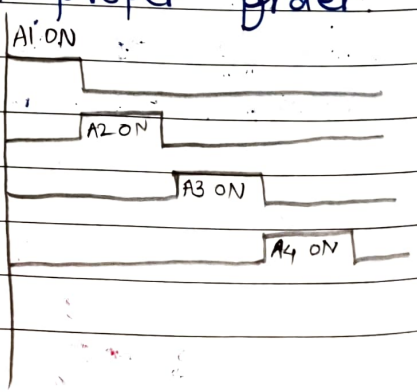
$$\frac{360}{200} = 1.8^\circ.$$



A1 = phase 1
A2 = phase 2
A3 = phase 3
A4 = phase 4.

To Rotate this motor we have to give excitation to each phase in proper order.

The stepper motor is rotated by switching individual phase ON for a given time one by one



Stepper Motor step angles

Step angle	Steps per Revolution	
0.72	500	
1.8	200	Generally this step angle Motor is used
2.0	180	
2.5	144	
5.0	72	
7.5	48	
15	24	

$$\text{Steps per second} = \frac{\text{rpm} \times \text{Steps per Revolution}}{60}$$

Modes to Drive Stepper Motor:

Full step Mode: Two electromagnets are energised at a single time

Torque which is generated will be greater.

Power Consumption is higher in it.

This type of mode is choosen when the torque is more important than power Consumption.

Half Step Mode: one electromagnet is energized in this mode.

It has low torque but the angular resolution is doubled

This type of drive is chosen when we want to increase the angular resolution of the motor.

Full Step Mode

Step	A	B	C	D
1	1	1	0	0
2	0	1	1	0
3	0	0	1	1
4	1	0	0	1

Anticlockwise

Clockwise

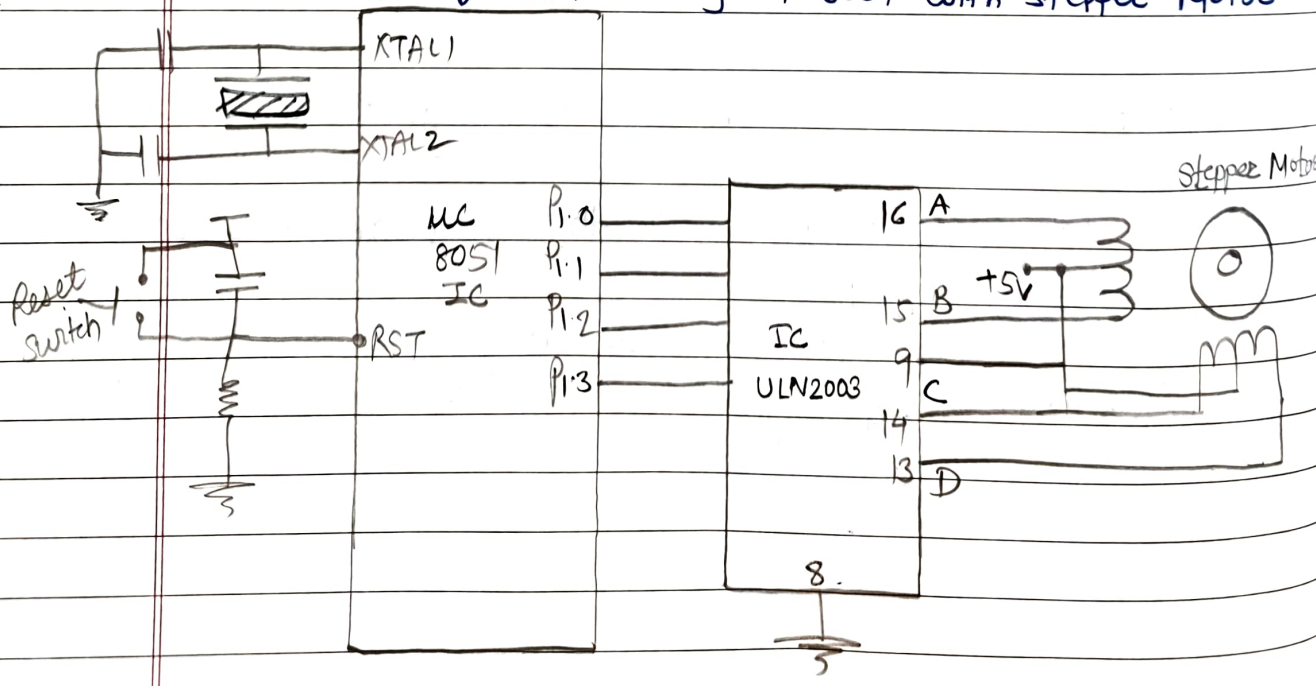
Half Step Mode

Step	A	B	C	D
1	1	0	0	0
2	1	1	0	0
3	0	1	0	0
4	0	1	1	0
5	0	0	1	0
6	0	0	1	1
7	0	0	0	1
8	1	0	0	1

Anticlockwise

Clockwise

fig: Interfacing of 8051 with stepper Motor



ULN2003 → Driver IC

Used for Driving individual phases of Motor

It is Darlington transistor array IC

It is used for driving high current loads such as relays, Motor .....

It has 8 individual channels, each channel with 1 Ampere capacity.

Anticlockwise ↑ / Clockwise ↓

P1	P1.7	P1.6	P1.5	P1.4	D P1.3	C P1.2	B P1.1	A P1.0
09H	X	X	X	X	0	0	1	1
06H	X	X	X	X	0	1	1	0
0CH	X	X	X	X	1	1	0	0
09H	X	X	X	X	1	0	0	1

C program to rotate stepper Motor clockwise in full step Mode

```
# include <reg51.h>
Void delay();
Void main()
{ P1 = 0x00;          //P1 as o/p Port
 while(1)
 {
  P1 = 0x03;
  delay();
  P1 = 0x06;
  delay();
  P1 = 0x0C;
  delay();
  P1 = 0x09;
  delay();
 }
}
```

```
Void delay()
{int i,j;
 for(i=0;i<100;i++)
 {
 for(j=0;j<1275;j++)
 {
 }
 }
}
```

C program to rotate Stepper Motor in clockwise direction with Half stepping Mode.

				D	C	B	A	
$P_{1.7}$	$P_{1.6}$	$P_{1.5}$	$P_{1.4}$	$P_{1.3}$	$P_{1.2}$	$P_{1.1}$	$P_{1.0}$	P_1 Value
X	X	X	X	0	0	0	1	01H
X	X	X	X	0	0	1	1	03H
X	X	X	X	0	0	1	0	02H
X	X	X	X	0	1	1	0	06H
X	X	X	X	0	1	0	0	04H
X	X	X	X	1	1	0	0	0CH
X	X	X	X	1	0	0	0	08H
X	X	X	X	1	0	0	1	09H

```
#include <reg51.h>
void delay();
void main()
{ while(1){
 P1 = 0x01;
 delay();
 P1 = 0x03;
 delay();
 P1 = 0x02;
 delay();
 P1 = 0x06;
 delay();
 P1 = 0x04;
 delay();
 P1 = 0x0C;
 delay();
 P1 = 0x08;
 delay();
 P1 = 0x09;
 delay();
}}
```

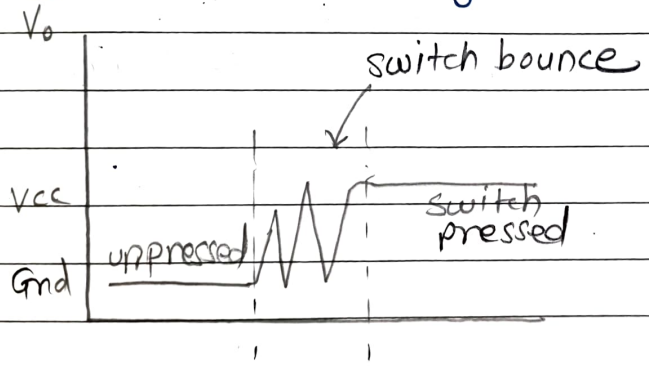
```
void delay()
{
int i,j;
for(i=0; i<100; i++)
{
for(j=0; j<1275; j++)
{
}
}
}
```

Keys :

A Keyboard is arranged as an array of switches
- mechanical
- membrane
- Capacitors
- Hall effect in construction.

Mechanical switches are most popular for keyboards.
- mechanical switches have a problem called contact bounce. closing a mechanical switch generates a series of pulse because the switch contacts donot come to rest immediately.

- In addition a human cannot type more than 50 keys in a second. Reading a keyboard more than 50 times a second will read the same key stroke too many times.



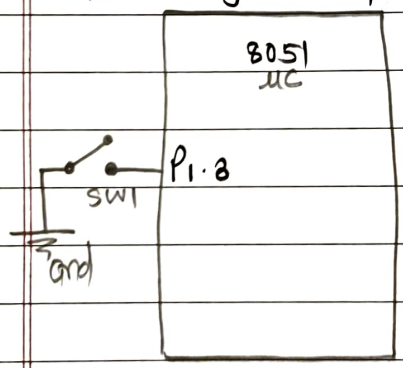
When we press a pushbutton or toggle switch two metal parts come into contact to short the supply.
But they don't connect instantly, the metal parts connect & disconnect several times before the actual connection is made.
The same thing happens while releasing the button This results in false or multiple triggering like the button is pressed multiple times.

Its like falling a bouncing ball from a height & it keeps bouncing on surface until it comes to rest

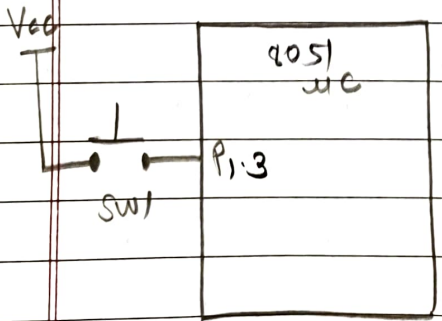
Switch bouncing :
Switch bouncing is the non ideal behavior of any switch which generates multiple transitions of single i/p.
switch bouncing cause problems while we are dealing with logic or digital circuits.
Hence to remove the bouncing from the circuit debouncing circuit is used.

Two types of debouncing :
① Software Debouncing → add delays, interrupt.
② Hardware Debouncing.

Interfacing simple keys to 8051

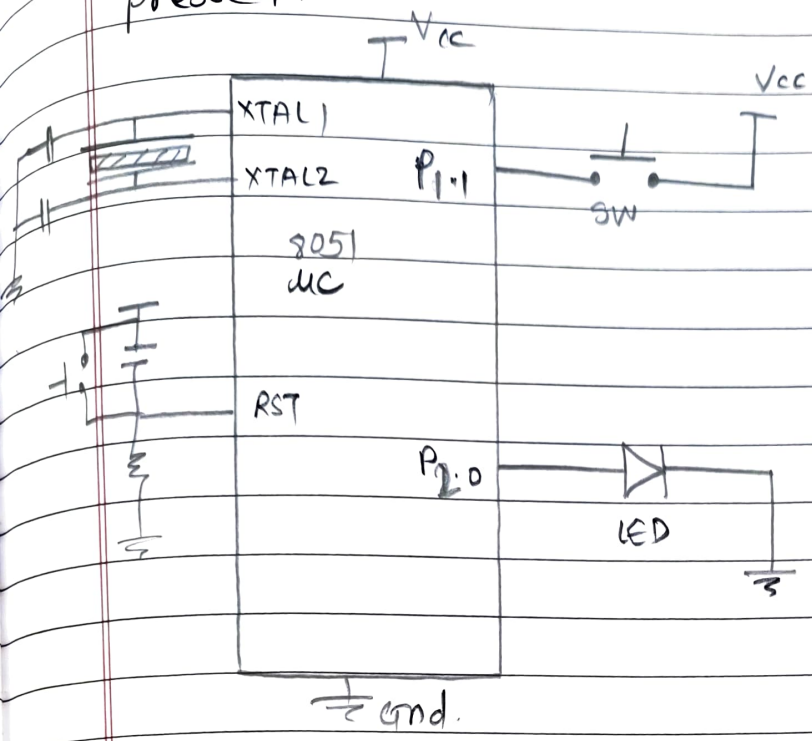


switch off → logic 1
switch ON → logic 0.



switch off → logic 0
Switch ON → logic 1

8051 c program to blink LED connected on P2.0 when switch connected on P1.1 is pressed.




```
#include <reg51.h>
sbit LED = P2^0;
sbit switch = P1^1;

void main ()
{P0 = 0xFF;          // P0 as i/p port.
while (1)
{
 if (switch == 1)
{
 LED = 1 ;
}
 else
{LED = 0 ;
}
}
}
```