LECTURE NOTES (E- CONTENTS) for

B.Sc. III Electronics (2021-22)

Semester: V Paper- V DSE 1005E1

Linear Integrated Circuits, 8051 Microcontroller Interfacing and Embedded C

Section II: 8051 Microcontroller Interfacing and Embedded C

Unit- 2: Real World Interfacing of 8051

Prepared and Circulated for

B .Sc. III Electronics Students

BY

Dr. C. B. Patil

Assistant Professor, Department of Electronics

Vivekanand College (Autonomous), Kolhapur

(For Private Circulation only)

UNIT 2
Real World Interfacing of 8051

Syllabus: Interfacing to output devices – LED, Relay, opto-coupler, LCD, seven segment display, seven segment display (multiplexing mode), DC Motor, Stepper Motor,ADC0804. Interfacing to input devices – Switch, 4X4 matrix keyboard, thumb wheel switch, DAC0808.

Interfacing with 8051 microcontroller:

It can be defined as transferring data from interfacing peripherals such as sensors, motors, machines, circuit components, and so on to 8051 microcontroller and vice versa. By interfacing with 8051, we can have ease of control over complex circuit components or devices. The 8051 microcontroller is frequently used in electronics projects for controlling several operations. In these circuits, generally several electronics devices or peripherals are interfaced with 8051 which are termed as interfacing devices. For example, 7 segment display interfacing with 8051, LCD display interfacing with 8051, Matrix keypad interfacing with 8051, interfacing DS1307 RTC with 8051 microcontroller, interfacing 8051 with servo motor, interfacing DC motor with 8051, 8051 microcontroller interfacing with ADC, and so on.

1. INTERFACING OF LED WITH 8051

Commonly used LEDs has generally barrier potential of 1.5V and current of 10mA. If this voltage and current applied to the LED, it glows with full intensity.

Circuit Description: The power on reset circuit with R1_C3 is connected to RESET pin and for generating clock the crystal and capacitors C1 and C2 of 33pf are connected between XTAL1 and XTAL2 pin of microcontroller.

We cannot connect any pin of the 8051 to the LED directly because required current for LED is more than sinking/sourcing capacity of the 8051 and it is harmful. Therefore transistor (SL100) is used as buffer. It's base terminal is connected to port pin P2.0 through 47K resistor. This resistor limits the base current. The LED is connected in between Vcc and collector of transistor through resistor R4.This resistor limits current through LED (10 mA).

When we make Pin P2.0 high transistor will become ON, then current flows through LED-collector-emitter of transistor and hence LED turns ON .To make LED off we make pin P2.0 low.

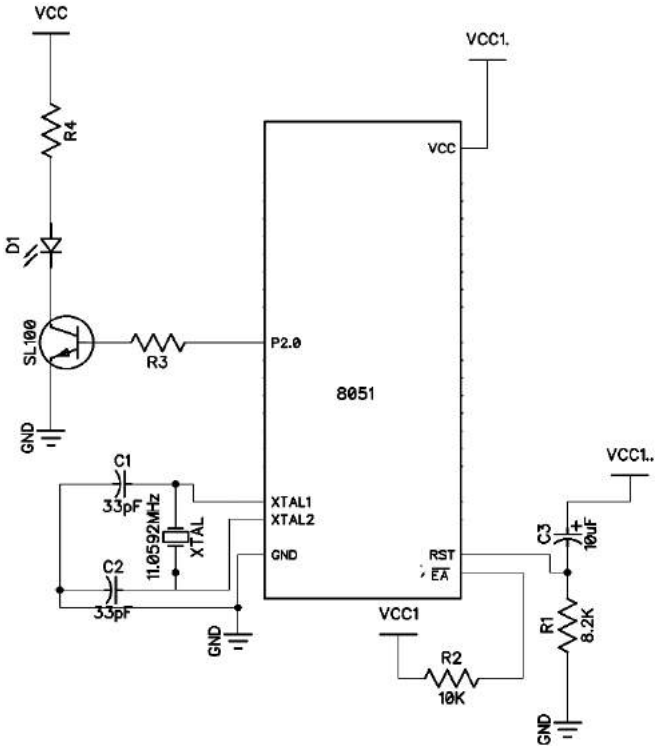


Fig.1 INTERFACING OF LED WITH 8051

Designing:

The value of resistor R4

R4= $(Vcc-VD-VCE)/IC$

Where, Vcc; Supply Voltage (+5V)

VD: LED barrier potential(+1.5V)

VCE: Collector to emitter (when transistor becomes on here 0.6V)

IC: Collector current(here LED Current=10mA)

R4 =330 Ohm

R3= $(VP2.0-VBE)/IB$= $(VP2.0-VBE)\ \beta/IC$

Where VP2.0=Maximum voltage at pin P2.0(Here it is +5V)

VBE=Base to Emitter Voltage when transistor ON(here 0.6V)

IB=base Current

IC: Collector current(here LED Current=10mA)

β= Current gain of transistor(Here 100)

R3=47K

```
// INTERFACING OF LED WITH 8051
#include <REGX51.H>
sbit LED=P2^0;          //assign P2.0 to variable LED
void delay();           //declaration of delay function
void main(void)
{
while(1)        //repeat forever
```

```
{LED=1;          //Turn ON LED
delay();         //Call delay subroutine
LED=0;           //Turn OFF LED
delay();          //Call delay subroutine
}
}
void delay()     //delay function
{
TMOD=0X01;  //select Timer 0, mode 1
 TH0=0X00;     //Load count  00H into TH0
 TL0=0X00;      //Load count  00H into TL0
 TR0=1;                  //start Timer
while(TF0==0);          //wait here until Timer overflows
 TR0=0;                  //stop Timer
 TF0=0;          //reset flag
}
```

2. Interfacing of Relay to 8051

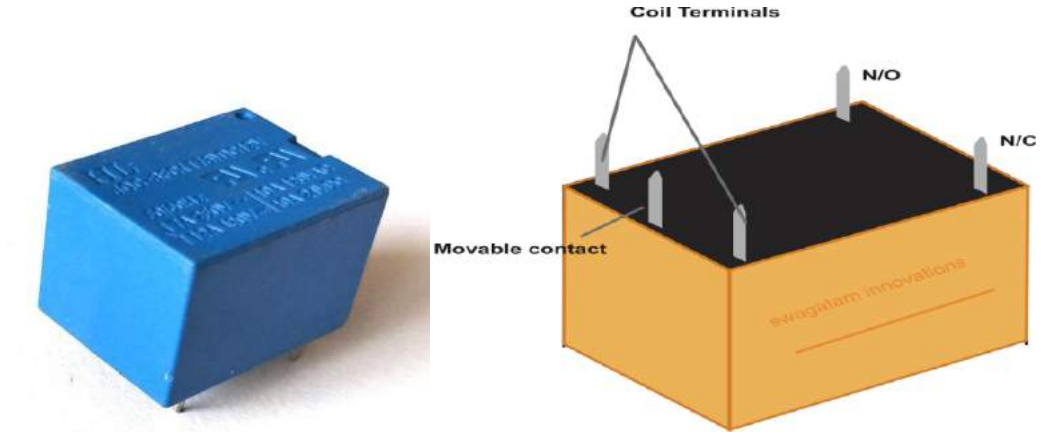


Fig.2: Relay terminals

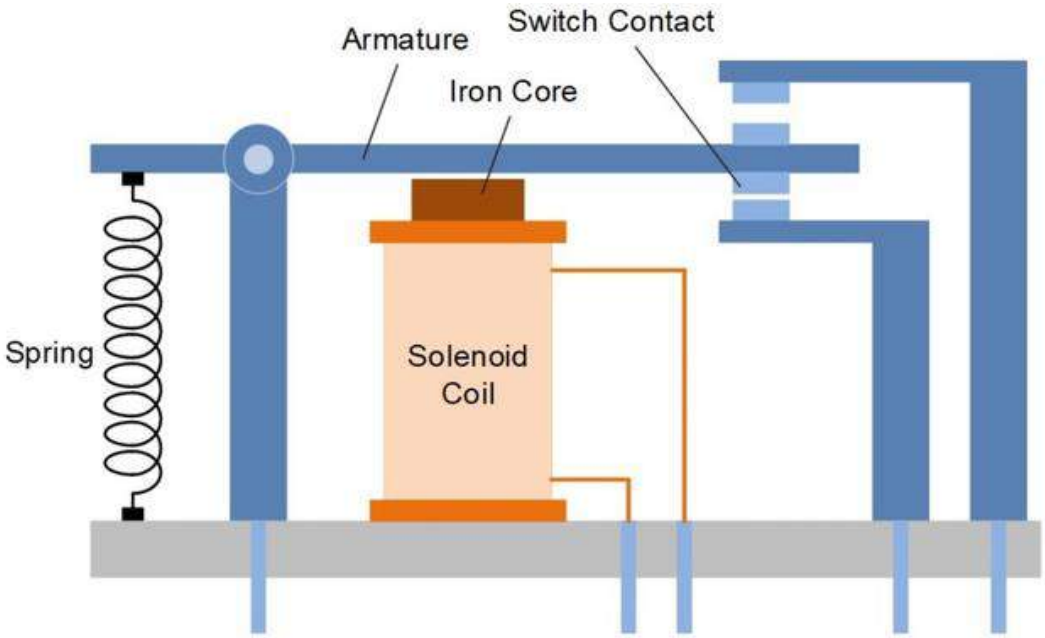


Fig.3: schematic of Relay

An Electromechanical relay is widely used in industry, automobiles etc. It isolate two separate sections of system with two different voltage levels It means one part of relay is connected to +5V or +12V while other is connected to 230 V AC or high DC. Therefore a relays are useful to control devices which operates on high voltage.

Parameters of the relay:

1. **Trigger Voltage**: this is the voltage required to turn on the relay that is to change the contact from Common->NC to Common->NO.

   relays of Trigger values :3V, 5V, 6V,12V

2. **Load Voltage & Current**: this is the amount of voltage or current that the NC,NO or Common terminal of the relay could withstand

   30V and 10A.

   230 V and 10A

The relay is connected in between Vcc (+12V) and collector of transistor. The 8051 cannot drive relay directly, so a transistor is used as a buffer in this circuit. A high voltage device Bulb is connected between common point and Normally Open (N/O) terminals of relay.

When we make Pin P2.0 high transistor will become ON and hence current flows through coil of relay and relay gets activated and connection between C and N/O terminals is developed and connected bulb will be turns ON. To make Relay off we make pin P2.0 low . During relay ON-OFF a back emf (inductive kick back) is generated in coil and this back emf is harmful to components connected in circuit. Here it is transistor. To avoid this back emf a freewheeling diode is connected across the coil of relay.

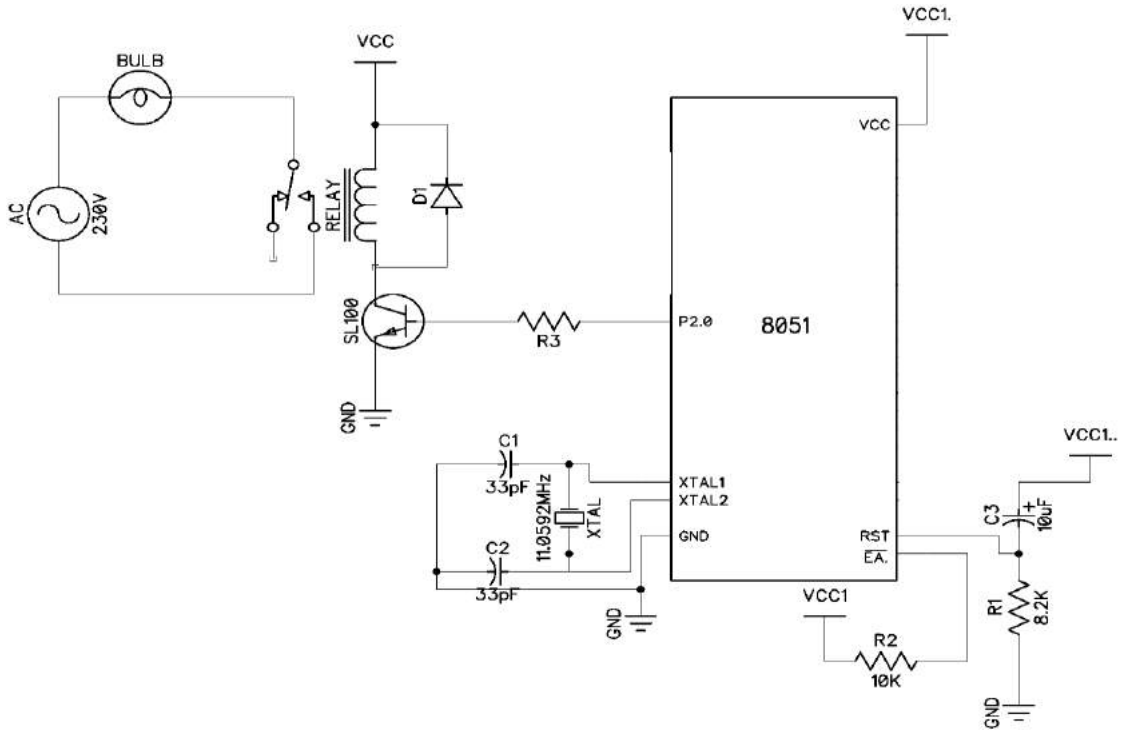
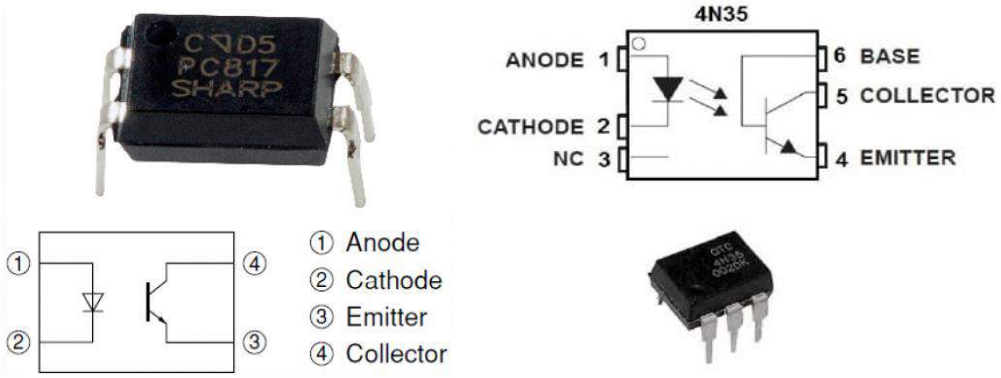


Fig.4 INTERFACING OF RELAY WITH 8051

```
// INTERFACING OF RELAY WITH 8051
#include <REGX51.H>
sbit RELAY=P2^0;       //assign P2.0 to variable RELAY
void delay();          //declaration of delay function
void main(void)
{
while(1)         //repeat forever
{ RELAY =1;      //Turn ON RELAY
delay();                 //Call delay subroutine
RELAY =0;        //Turn OFF RELAY
delay();                 //Call delay subroutine
}
}
void delay()     //delay function
{
TMOD=0X01;  //select Timer 0, mode 1
```

```
 TH0=0X00;     //Load count  00H into TH0
 TL0=0X00;      //Load count  00H into TL0
 TR0=1;                  //start Timer
while(TF0==0);          //wait here until Timer overflows
 TR0=0;                  //stop Timer
 TF0=0;          //reset flag
}
```

3. Interfacing of optocoupler to 8051 :



PC817 OPTOCOUPLER 4N35 OptoCoupler Optoisolator DIP IC - 4N35

Fig.5:optocouplers ICs and their schematics

The inductive devices like motors are produces back emf of voltage spike due to sudden change in current. This back emf is harmful to other devices. To protect the devices from back emf mostly optocouplers are used.

An optocoupler or opto-isolator consists of a light emitter (LED) and a light sensitive receiver which can be a single photo-diode, photo-transistor, photo-resistor, photo-SCR, or a photo-TRIAC. Both the light emitter and photo-sensitive device are enclosed in a light-tight body or package with metal legs for the electrical connections as shown in Figure.

The emitted light falls upon the base of the photo-transistor, causing it to switch-ON and conduct in a similar way to a normal bipolar transistor. When light is not emitted the transistor goes into OFF state.

A LED of optocoupler is connected to port pin 2.0 through a buffer transistor and the motor is connected at collector of optocoupler transistor.

When we make Pin P2.0 high, buffer transistor will become ON and hence current flows through LED of optocoupler. Optocoupler transistor becomes ON and motor turns ON. To make motor off we make pin P2.0 low.

During motor ON-OFF a back emf (inductive kick back) is generated in coil and this back emf is harmful to components connected in circuit. Here it is transistor. To avoid this back emf a freewheeling diode is connected across the motor.

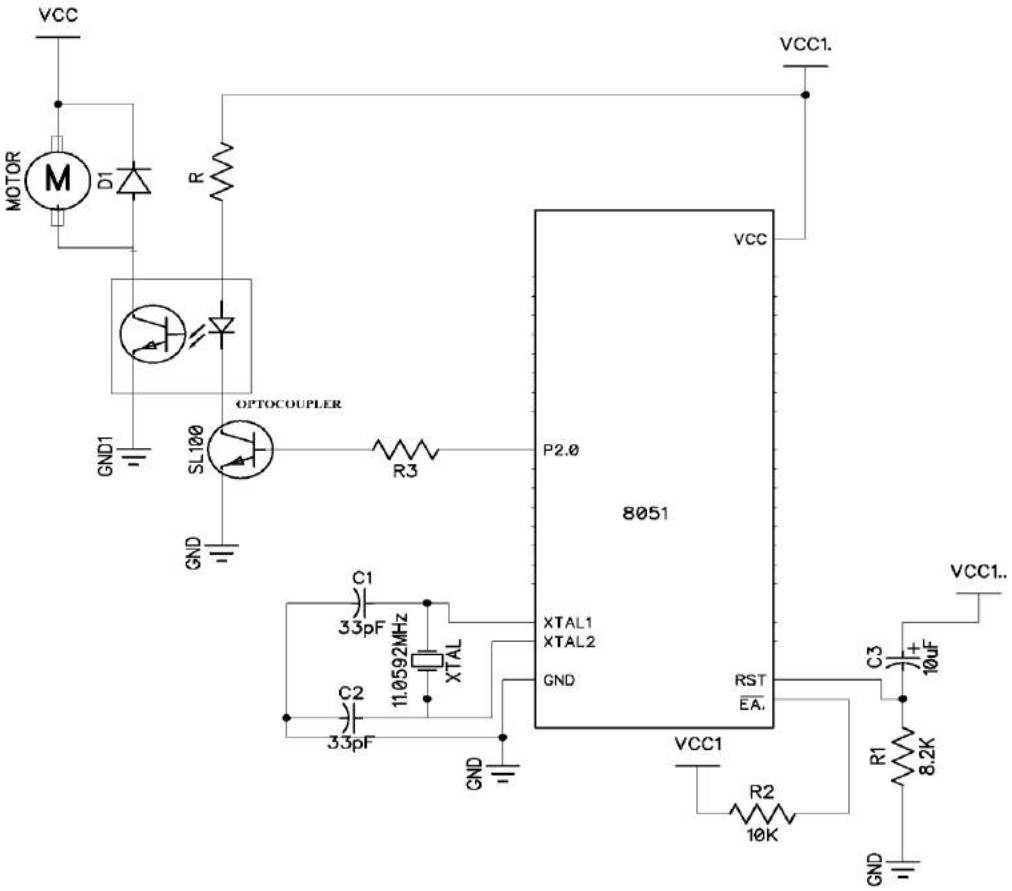


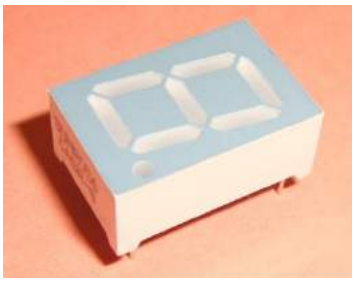
Fig.6 INTERFACING OF OPTOCOUPLER WITH 8051

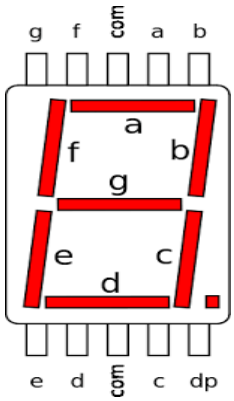
```
// INTERFACING OF OPTPCOUPLER WITH 8051
#include <REGX51.H>
sbitoptocoupler=P2^0;          //assign P2.0 to variable optocoupler
void delay();                  //declaration of delay function
void main(void)
{
while(1)                  //repeat forever
```

```
{ optocoupler =1;       //Turn ON optocoupler
delay();                //Call delay subroutine
optocoupler =0;         //Turn OFF optocoupler
delay();                //Call delay subroutine
}
}
void delay()            //delay function
{
TMOD=0X01;              //select Timer 0, mode 1
 TH0=0X00;              //Load count  00H into TH0
 TL0=0X00;              //Load count  00H into TL0
 TR0=1;                 //start Timer
while(TF0==0);          //wait here until Timer overflows
 TR0=0;                 //stop Timer
 TF0=0;                 //reset flag
}
```

4. Interfacing of single seven segment display with 8051

It is used to shows numbers 0 to 9 and alphabets by glowing seven segments pins $\overline{A}$, $\overline{B}$, $\overline{C}$, $\overline{D}$, $\overline{E}$, $\overline{F}$, $\overline{G}$, $\overline{H}$. It has two types: common anode and common cathode





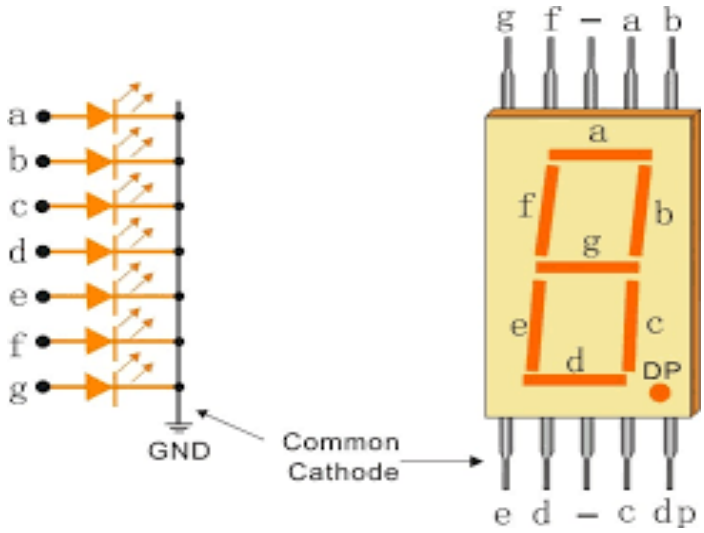


Fig. 7 photograph and schematic

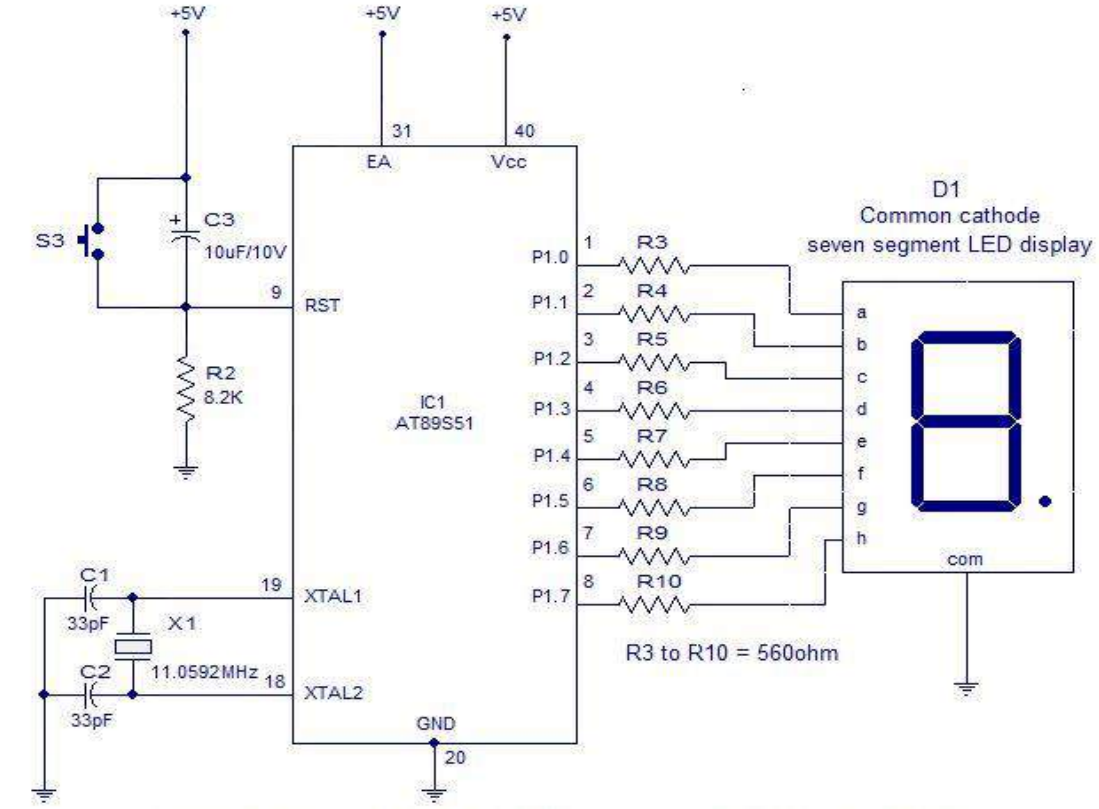


Fig.8: Interfacing of single seven segment display with 8051

```
//Interfacing of single seven segment display with 8051
#include <reg51.h>
int main()
{
   char seg_code[]={0x3F,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};
                        //array of codes for common cathode
```

```
int k;
while (1)
  {
for (k = 0; k <= 9; k++) // loop to display 0-9
     {
        P1 = seg_code[k];
DELAY_ms(1000);
     }
  }
}
voidDELAY_ms(unsigned intms_Count)
{
unsignedint i, j;
for(i=0;i<ms_Count;i++)
  {
for(j=0;j<1275;j++);
  }
}
```

5. Interfacing of seven segment display in multiplexing mode with 8051

The circuit diagram of the interfacing of two seven segment display using transistor is shown in figure. The A, B, C, D, E, F, G pins of both the Displays are interconnected to each other and then connected to port P1. The two pins of the port P3 are used as select lines for selection of the proper seven segment display. Whenever a high input is given to any of the control lines the corresponding seven segment gets activated. As a result the data send by P1for corresponding digit gets displayed. To display the next digit, the next seven segment is activated and the corresponding data is loaded on port P1. The speed at which the two Displays are activated one after the other is so fast that it appears as if a two digit number is displayed and not a single digit one after the other.

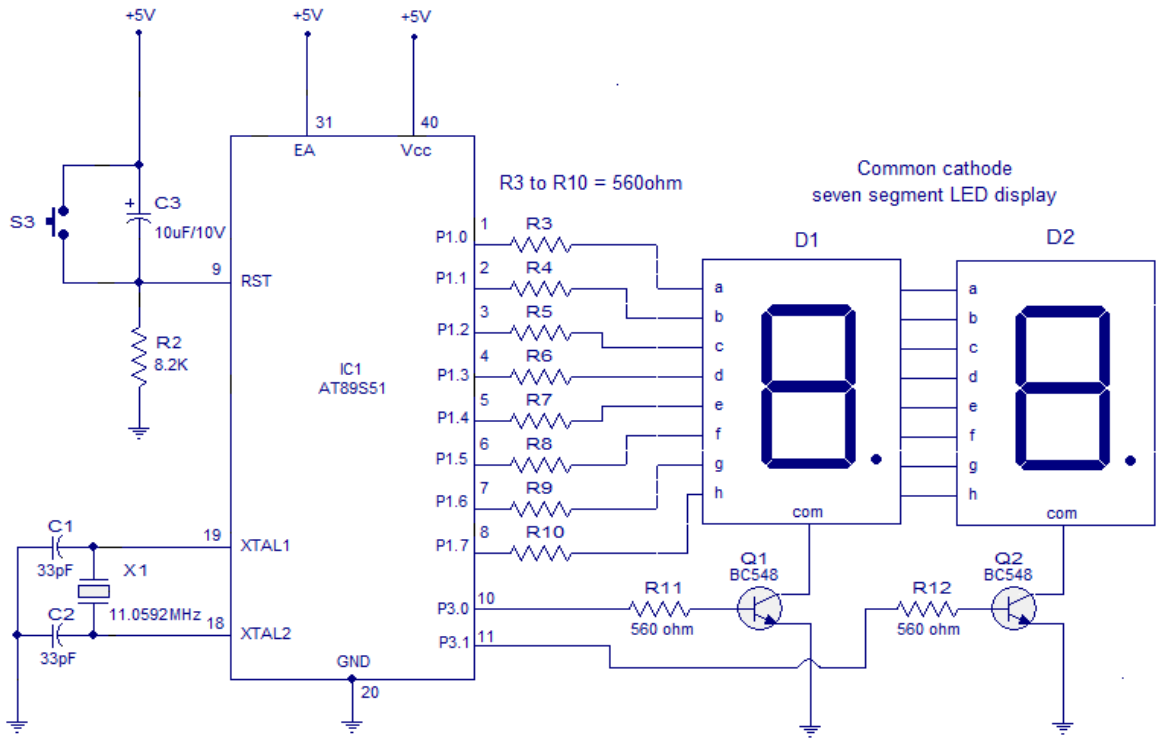


Fig.9: Interfacing of seven segment display in multiplexing mode with 8051

```
//Interfacing of two seven segment display in multiplexing mode with 8051
#include<reg51.h>
sbit a=P3^0;                    // to  on off seven segment at tenth place
sbit b=P3^1;                     // to  on off seven segment at once place )
void main()
{
unsigned char seg-code[]={0x3F,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};
unsignedint i, j, k;
while(1)
  {
for(i=0;i<100;i++)
  {
    a=1;                                // on the 7 seg at tenth place
    b=0;                              // off the 7 seg at ones place
    P1= seg-code [i/10];                // sending 0-9 to port 2 to 7 seg
for(k=0;k<35000;k++);                 // delay
```

```
a=0;                              // off 7 seg at tenth place
 b=1;                             // on 7 seg at ones place
 P1= seg-code[i%10];
for(k=0;k<35000;k++);          // delay
 }
 }
 }
```

6. Interfacing of seven segment display and thumb Wheel switch to 8051

Thumbwheel Switch:A thumbwheel switch is a simple electromechanical device that enables an operator to provide a code number as input to data processing equipment like microcontroller. The decimal version contains a wheel on which numbers are printed, usually in white on black, from 0 through 9, visible one at a time through a window in the face of the switch. When wheel is rotated, number in window increase or decreases according to direction of rotation of thumb wheel. The thumbwheel switch includes a common (input) pin and four output pins on BCD data. The BCD output depends on the switch position. If switch is at position 9BCD output is1001.

Seven segment display: It is used to shows numbers 0 to 9 and alphabets by glowing seven segmentspins $\overline{A}$,$\overline{B}$,$\overline{C}$,$\overline{D}\overline{E}$,$\overline{F}$,$\overline{G}$,$\overline{H}$. It has two types: common anode and common cathode.

Fig.10: photographs ofThumbwheel Switch

Circuit diagram

The interfacing of the thumbwheel switch and seven segment display to 8051 is shown in fig. 3.9. The BCD outputs of thumbwheel are connected to four pins of P2 pins P2.0-P2.3. The seven segment display is connected to the output of decoder IC 7447. IC7447 converts BCD numbers into seven segment form for common anode display. Using four pins of P1 such as P1.0-P1.3. A BCD numbers are applied to the decoder. The BCD data corresponding to decimal number on switch is read and store at accumulator using port 2 pins.Then it is transferred to BCD input of IC7447 through port 1 pins. The seven segment display shows corresponding number.

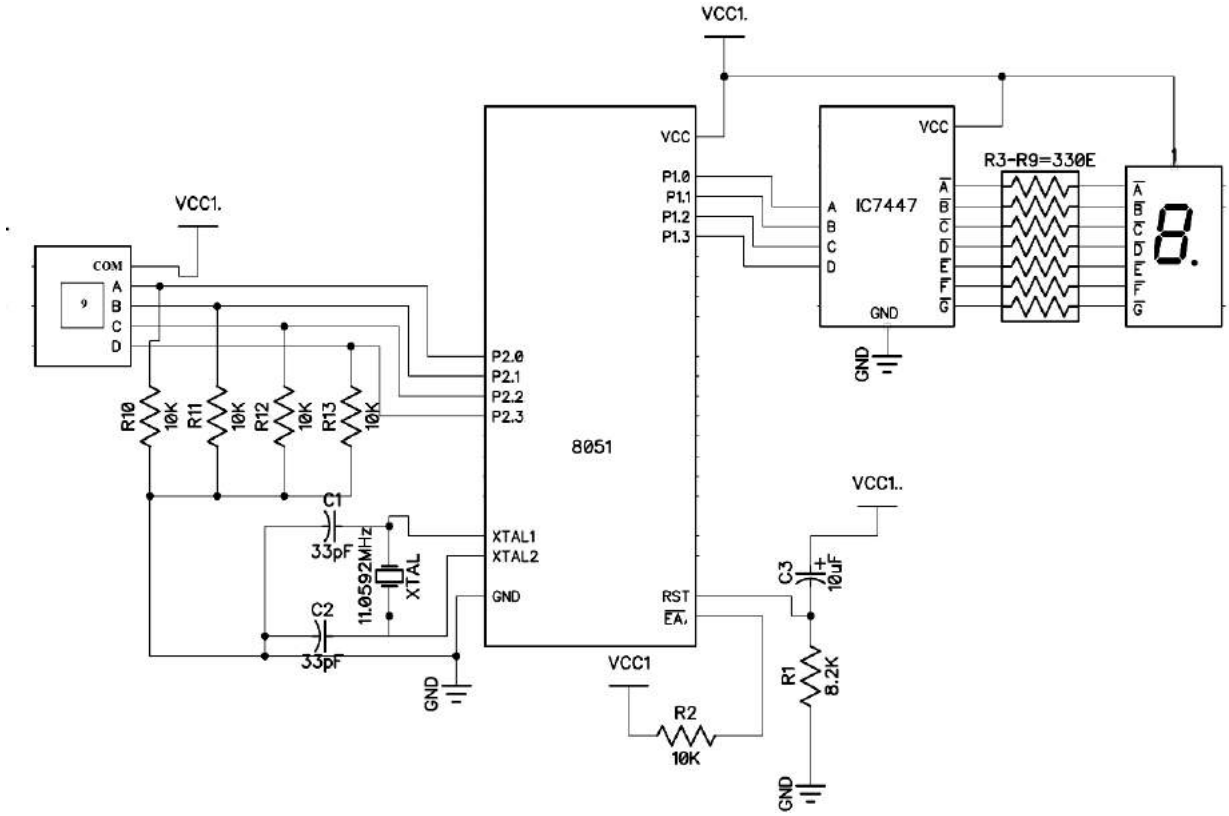


Fig.11: Interfacing of seven segment display and thumb Wheel switch to 8051with decoder IC 7447

```
// Interfacing of seven segment display and thumb Wheel switch with decoder IC 7447
#include <REGX51.H>
sfr thumbwheel=P2;
sfr display=P1;
void main(void)
{while(1)
{thumbwheel=0XFF;
Display=thumbwheel);
}}
```

Interfacing of seven segment display and thumb Wheel switch to 8051 without decoder IC 7447

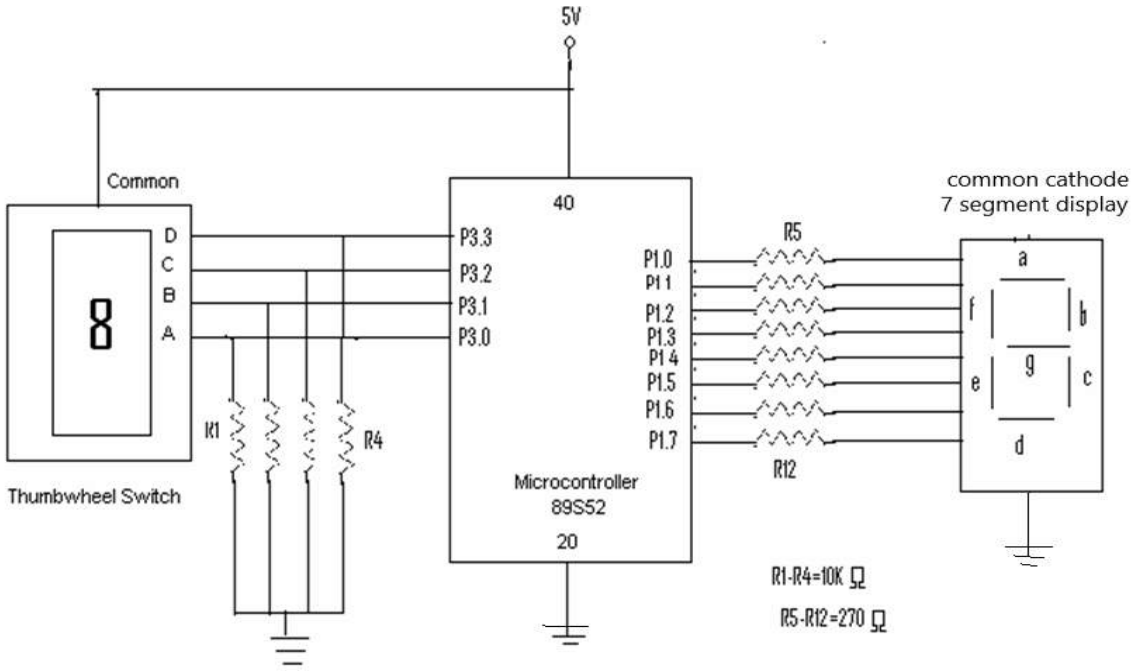


Fig.12: Interfacing of seven segment display and thumb Wheel switch to 8051

```
//Interfacing of seven segment display and thumb Wheel switch to 8051 without decoder IC
//7447
#include <REGX51.H>
void main (void)
{ P3=0xFF;        // make P3 as input port
while(1)
{ P3=0x0F&&P3;     // mask upper nibble of P3
if (P3==0x00)       // if P3=00 then send code 3F to display digit zero.
{  P1=0x3F;}
else if(P3==0x01)   // if P3=01 then send code 06 to display digit one
{ P1=0x06;}
else if(P3==0x02)    // if P3=02 then send code 5B to display digit two
{P1=0x5B;}
else if(P3==0x03)    // if P3=03 then send code 4F to display digit three
{P1=0x4F;}
else if(P3==0x04)    // if P3=04 then send code 66 to display digit four
{P1=0x66;}
```

```
else if(P3==0x05)   // if P3=05 then send code 6D to display digit five
{P1=0x6D;}
else if(P3==0x06)    // if P3=06 then send code 7D to display digit six
{P1=0x7D;}
else if(P3==0x07)    // if P3=07 then send code 07 to display digit seven
{P1=0x07;}
else if(P3==0x08)    // if P3=08 then send code 7F to display digit eight
{P1=0x7F;}
else                 //otherwise send code 6F to display digit nine
{P1=0x6F;}}}
```

7. Interfacing of LCD with 8051

LCD means Liquid Crystal Display. It is used to displaying numbers, character, etc. The 16×2 LCD module is a very common type of LCD module that is used in 8051 based embedded projects. It consists 16 rows and 2 columns of 5×7 or 5×8 LCD dot matrices. It is available in a 16 pin package with back light,contrast adjustment function and each dot matrix has 5×8 dot resolution. The pin numbers, their name and corresponding functions are shown in the following table.

Pin No.	Name	Function
1	VSS	This pin must be connected to the ground
2	VCC	Positive supply voltage pin (5V DC)
3	VEE	Contrast adjustment
4	RS	Register selection RS=1 selects data register RS=0 Selects Command Register
5	R/W	Read or write To read from reg R/W=1 to write on reg. R/W=0
6	E	Enable High to low pulse enables command or data reg.
7-14	D0-D7	LCD Data Lines
15	LED+	Back light LED+
16	LED-	Back light LED-

Table 1:Pin functions of LCD

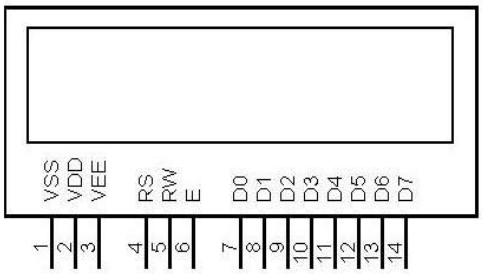


Fig. 13 schematic of LCD

Command	Function
0FH	LCD ON, Cursor ON, Cursor blinking ON
01H	Clear screen
02H	Return home
04H	Decrement cursor
06H	Increment cursor
0EH	Display ON ,Cursor blinking
80H	Force cursor to the beginning of 1st line
C0H	Force cursor to the beginning of 2nd line
38H	Use 2 lines and 5×7 matrix
08H	Display OFF, Cursor OFF

Table 2: LCD commands

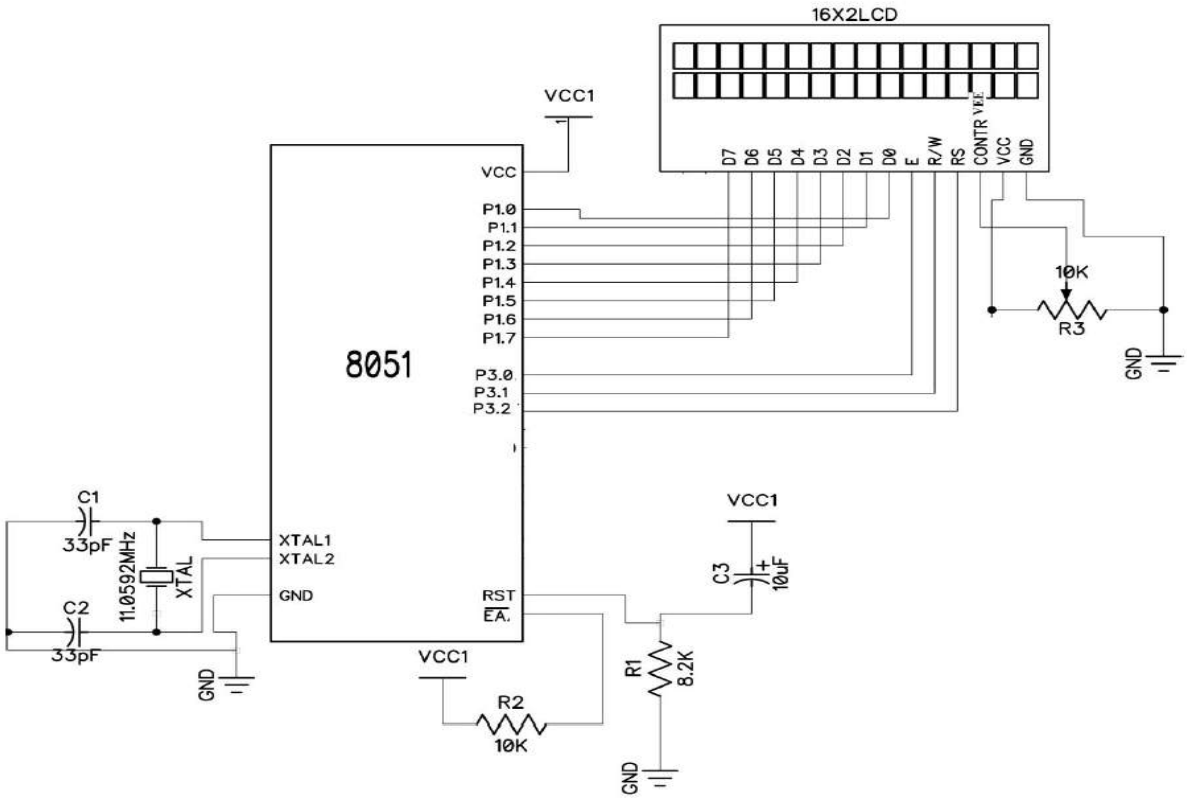


Fig.14: Interfacing of LCD with 8051

Ex: Write an 8051 C program to send letters ‘M’, ‘D’, and ‘E’ to the LCDusing the busy flag method.

Solution:

```
#include <reg51.h>
sfr ldata = 0x90; //P1=LCD data pins
sbitrs = P2^0;
sbitrw = P2^1;
sbit en = P2^2;
sbit busy = P1^7;
void main()
{
lcdcmd(0x38);
lcdcmd(0x0E);
lcdcmd(0x01);
lcdcmd(0x06);
lcdcmd(0x86); //line 1, position 6
```

```
lcddata('M');
lcddata('D');
lcddata('E');
}
voidlcdcmd(unsigned char value)
{
lcdready(); //check the LCD busy flag
ldata = value; //put the value on the pins
rs = 0;
rw = 0;
en = 1; //strobe the enable pin
MSDelay(1);
en = 0;
return;
}
voidlcddata(unsigned char value)
{
lcdready(); //check the LCD busy flag
ldata = value; //put the value on the pins
rs = 1;
rw = 0;
en = 1; //strobe the enable pin
MSDelay(1);
en = 0;
return;
}
voidlcdready()
{
busy = 1; //make the busy pin at input
rs = 0;
rw = 1;
en = 1; //strobe the enable pin
```

```
MSDelay(1);
en = 0;
while(busy==1){ //wait here for busy flag
}
voidlcddelay(unsigned intitime)
{
unsignedint i, j;
for(i=0;i<itime;i++)
for(j=0;j<1275;j++);
}
```

8. Interfacing of ADC0804 with 8051

ADC0804 is a very commonly used 8-bit analog to digital convertor. It is a single channel IC, i.e., it can take only one analog signal as input. The digital outputs vary from 0 to a maximum of 255(00H to FFH). Fig 3. 17 shows the pin diagram of ADC 0804.

CS: It is an active low input pin. The ADC0804 is selected by making this pin low.

RD: It is an active low input pin and used to get converted digital at data lines by applying a high to low pulse on this pin

WR: It is an active low input pin. It also known as start of conversion pin. When low to high pulse is applied to this pin, ADC0804 start to convert analog data into digital.

INTR: It is an active low output pin. It also known as end of conversion pin. It is normally high, when ADC completes its conversion this pin become low and indicates conversion is completed.

DB0-DB7: Data lines. The converted 8-bit digital data is available on these pins.

CLK IN & CLK R: ADC0804 has on chip clock generator. These pins are used to generate an internal clock by connecting a resistor (10K) and capacitor (150pF). It gives an internal clock frequency of 606 KHz. The conversion time (110 µS) of ADC is depends on this frequency.

Vref/2 :The resolution/step size of the ADC is depends on the voltage at Vref/2. It can be externally adjusted to convert smaller input voltage range to full 8 bit resolution. If Vref/2 left open means input voltage span is 0-5V and step size is 5/255=19.6 mV. The table 3.4 shows Vref/2 voltage, input voltage range and step size.

Vin(+) & Vin(-): The analog input is applied to this pin.

VCC: +5V is applied at this pin.

AGND and DGND: ADC0804 has a separate analog and digital ground. AGND is an analog ground connected to the analog input ground like sensors. DGND is a digital ground connected to the digital system here it is microcontroller 8051.

Vref/2 (pin9) (V)	Input voltage range (V)	Step size (mV)
Unconnected	0 – 5	5/255 = 19.6
2	0 – 4	4/255 = 15.69
1.5	0 – 3	3/255 = 11.76
1.28	0 – 2.56	2.56/255 = 10.04
1.0	0 – 2	2/255 = 7.84
0.5	0 – 1	1/255 = 3.92

Steps for converting the analogue input to digital form

1. Make $\overline{CS}$=0
2. Send a low to high pulse to $\overline{WR}$ pin to start the conversion.
3. Keep monitoring the $\overline{INTR}$ pin. If $\overline{INTR}$=1 conversion is not finished and if $\overline{INTR}$=0 conversion is finished.
4. Send a high to low pulse to $\overline{RD}$ pin to read the data from the ADC
5. Make $\overline{CS}$=1
6. Store read data at safe place/send any display connected to microcontroller
7. Go to step 1 for next data conversion

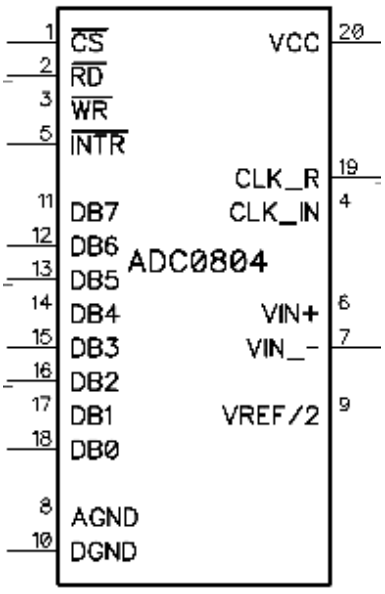


Fig. 15: Pin diagram of ADC 0804

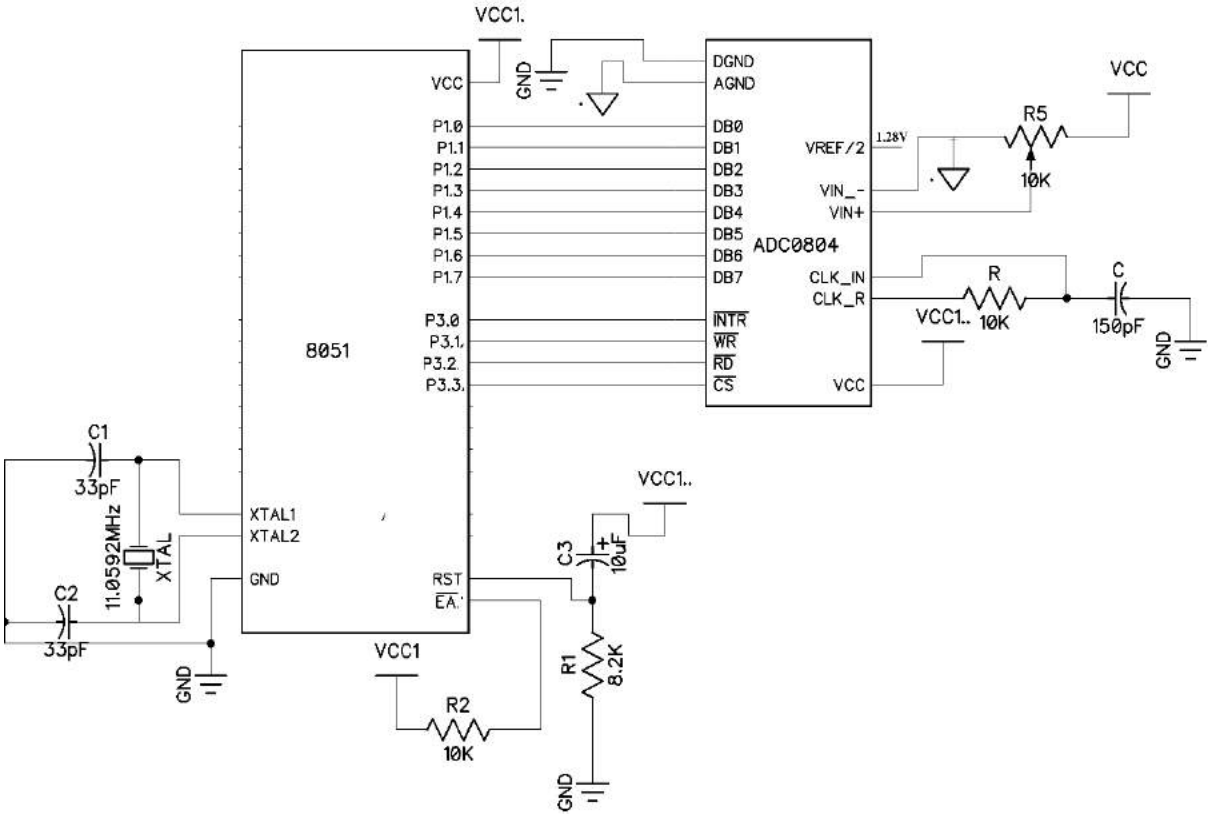


Fig.16: ADC0804 interfacing with 8051

```
//PROGRAM TO INTERFACE ADC 0804
#include <REGX51.H>
sbit INTR = P3^0;
sbit WR = P3^1;
sbit RD =P3^2;
sbit CS = P3^3;
void main()
{
  P1=0xFF;       //make P1 as input port
  CS=0; //make CS low
  INTR=1;        //make INTR high
  RD=1; //make RD high
  WR=1;          //make WR high
while(1)         //repeat forever
  {
          WR=0;                  //make WR low
```

```
        WR=1;                   //make WR high (start conversion)
        while(INTR==1);                 //check INTR pin for conversion complete
        RD=0;                   //read converted data
        P2=P1;                  //send converted digital data to port P2
        RD=1;                   //stop reading
  }
}
```

9. DAC0808 interfacing with 8051

The digital-to-analog converter (DAC) is a device widely used to convert digital pulses to analog signals. The majority of DAC including uses the R/2R method for conversion of digital signal into analog. The DAC0808 is one of the R/2R type DAC. It has 8 digital inputs, therefore it provides 256 discrete current levels at the output.

By connecting a resistor (or OPAMP current to voltage converter circuit) at Iout pin we can convert the result to voltage. The total current provided by the Iout pin is a function of the binary numbers at the A1-A8 inputs of the DAC0808 and the reference current (Iref). The relation between Iout, Iref and digital input A1-A8is as follows,

$$\text{Iout} = \text{Iref}\left(\frac{A1}{2}+\frac{A2}{4}+\frac{A3}{8}+\frac{A4}{16}+\frac{A5}{32}+\frac{A6}{64}+\frac{A7}{128}+\frac{A8}{256}\right)$$

Where A8 is the LSB, A1 is the MSB for the inputs and Iref is the input current that must be applied to pin 14(Vref+).

The pin diagram of DAC0808 as shown in fig.3.19 and the interfacing of DAC0808 with 8051 microcontroller is as shown in fig. 3.20. The Iref current is generally set to 2.0 mA. This reference current is generated by using thestandard 5-V power supply and 2.5K (1K and 1.5K-ohm standard) resistors(5/1+1.5K=2mA) and is applied to Vref+ pin. Digital input to the DAC is applied using port 1 pins P1.0 to P1.7 by connecting A8 to A1. The output of the DAC is current (Iout) form. This output is coupled to current to voltage circuit designed by op-amp with 5K feedback resistor and RC filter. The analog output voltage can observed on CRO or measured by using multitier.

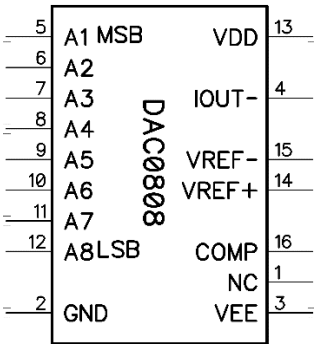


Fig. 17: Pin diagram of DAC0808

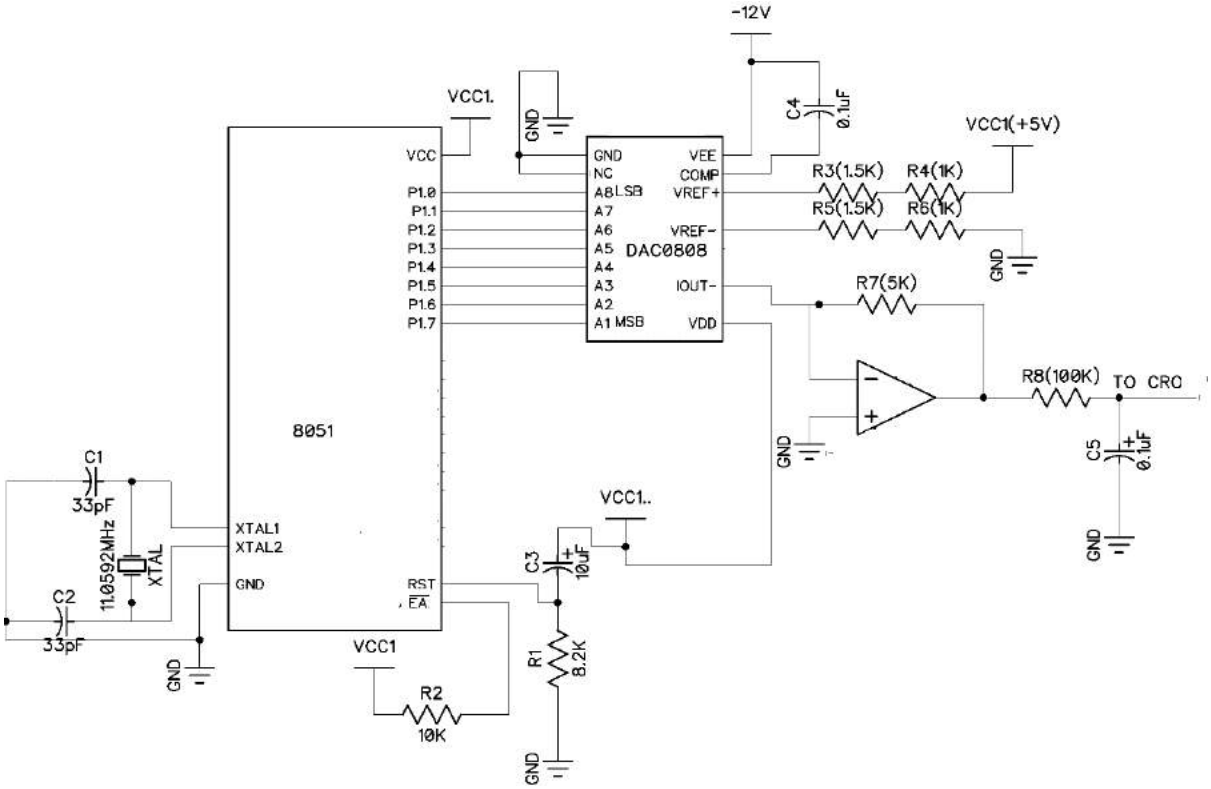


Fig.18: DAC 0808 interfacing with 8051

```
//program to generate triangular wave
#include< reg51.h>
unsigned char d;
void delay();
void main(void)
{
while(1)
 {
for(d=0; d<255; d++)
     {
        P1 = d;
```

```
            delay();
        }
for(d=255; d>0; d--)
        {
           P1 = d;
            delay();
        }
    }
}
void delay()
{
TMOD=0X01;
 TH0=0X00;
 TL0=0X00;
 TR0=1;
while(TF0==0);
 TR0=0;
 TF0=0;
}
```

10. Interfacing of DC motor with 8051

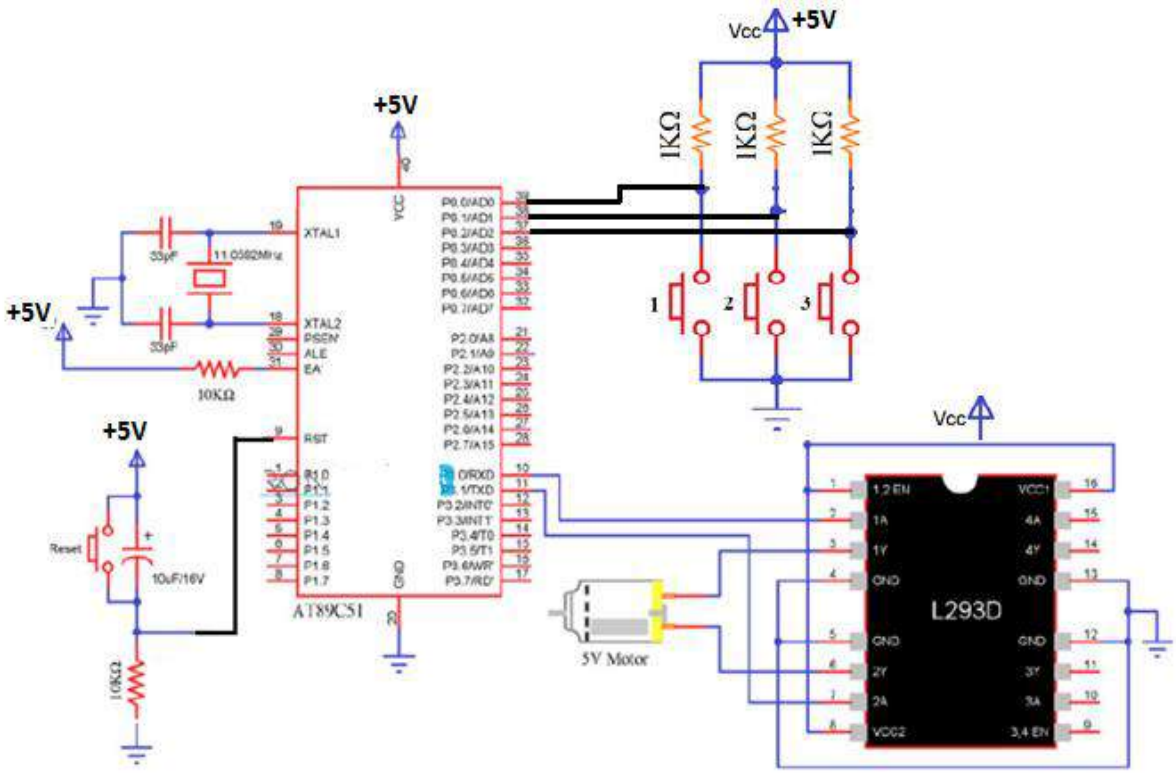


Fig.19: DC motor interfacing with 8051

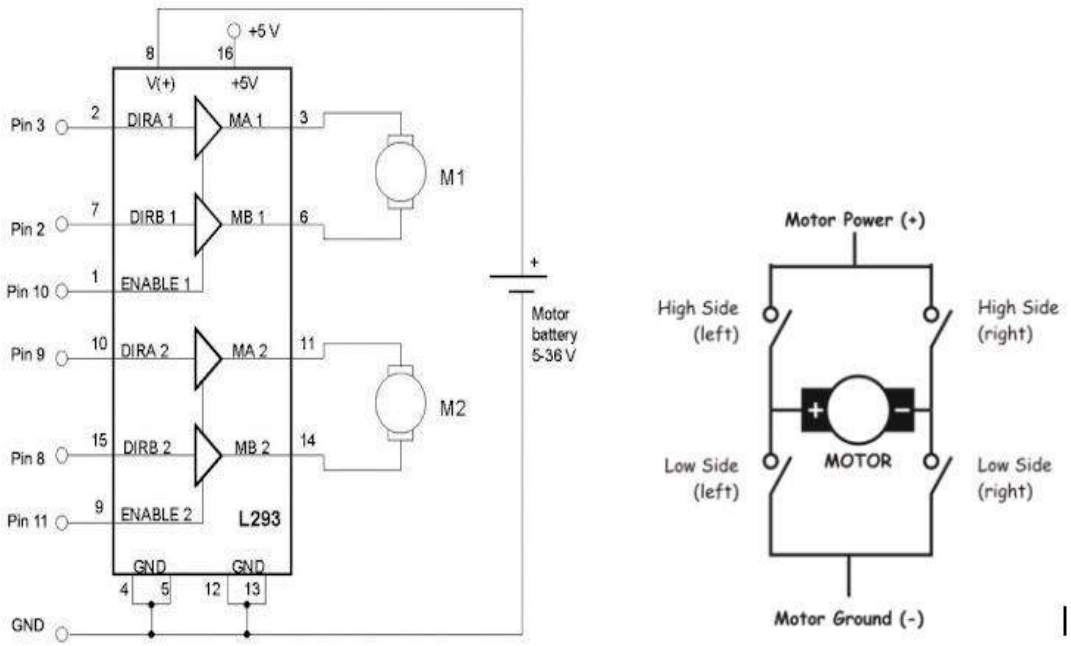


Fig.20: IC L293

```
// interfacing of DC motor to 8051
#include<reg51.h>
sbit A1 = P3^0;
sbit A2 = P3^1;
sbit forward = P0^0;
sbit backward = P0^1;
sbit stop = P0^2;
void main()
{       A1=0;
        A2=0;
        forward=1;     //make forward switch as input
        backward=1;   //make backward switch as input
        stop=1;          //make stop switch as input
        while(1)
        {
        if(forward==0)
        {
            do
            {A1=1;      //rotate motor in forward direction
```

```
            A2=0;      //rotate motor in forward direction
        } while(forward==0);
                }
        elseif(backward==0)
                {
            do
            { A1=0;     //rotate motor in backward direction
            A2=1;       //rotate motor in backward direction
            } while(backward==0);
             }
        elseif(stop==0)
           {A1=0;       //stop motor
            A2=0;       //stop motor
             while(stop==0);
        }
        }
}
```

11. Interfacing of stepper motor to 8051

A Stepper Motor is a brushless, synchronous motor which divides a full rotation into a number of steps. DC motor which rotates continuously when a fixed DC voltage is applied to it,while a step motor rotates in discrete step angles. The number of steps required to complete one complete rotations known as steps per revolution. If stepper motor has 12, 24, 72, 144, 180 and 200 resulting stepping angles are 30, 15, 5, 2.5, 2, and 1.8 degrees per step (step angle= 360/step angle).

Fig. 21: Stepper motors

Working (Stepper Motor)

Stepper motors consist of a permanent magnetic rotating shaft, called the rotor and electromagnets on the stationary portion that surrounds the motor, called the stator. Fig.3.12 illustrates one step rotation of a stepper motor. At position 1, we can see that the rotor is beginning at the upper electromagnet, which is currently active (has voltage applied to it). To move the rotor clockwise (CW), the upper electromagnet is deactivated and the

right electromagnet is activated, causing the rotor to move 90 degrees CW, aligning itself with the active magnet. This process is repeated in the same manner step by stepupto starting position.In this example step angle is 90 degree and it will requires 4 steps to complete one rotation. This is full stepping method.

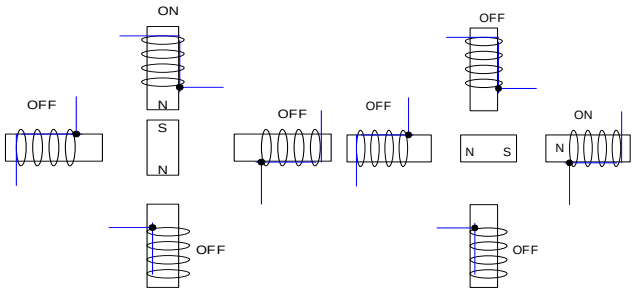


Fig.22: Full Stepping Method

We can double the resolution (step angle) of some motors by using half-stepping method. Instead of switching the next electromagnet in the rotation on one at a time, with half stepping you turn on both electromagnets, causing an equal attraction between due to this stepper motor required eight steps to complete one revolution, thereby doubling the resolution. From Fig3.13, in the first position only the upper electromagnet is active, and the rotor is drawn completely to it. In position 2, both the top and right electromagnets are active, causing the rotor to position itself between the two active poles. Finally, in position 3, the top magnet is deactivated and the rotor is drawn all the way right. This process can then be repeated for the entire rotation.

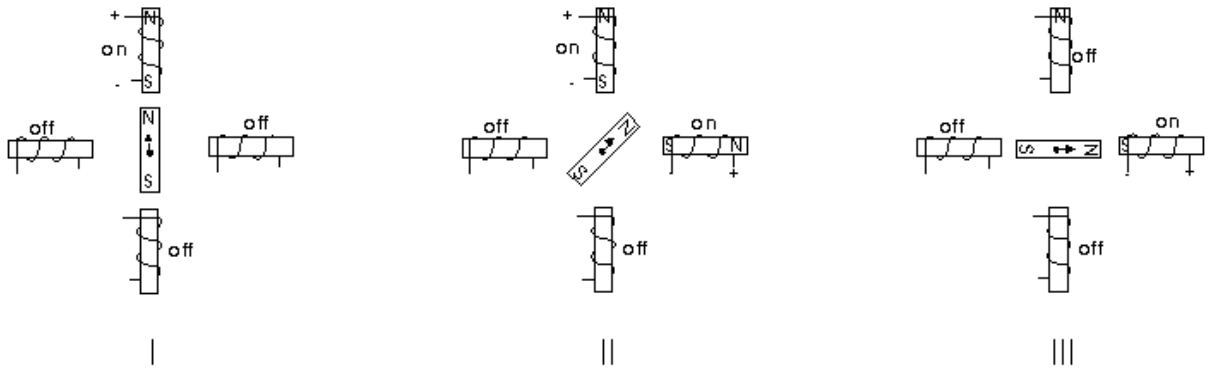


Fig.23: Half stepping

To rotate a stepper motor we use normal 4 step sequence as shown in table 3.1,

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

Table Stepper Motor step sequence

*We can start from any step e.g. from step 4= 0011= 6H. Therefore, if we load number 66H in to the accumulator and rotate accumulator left or right bit pattern of lower or upper nibble is same shown in table 3.

The interfacing of stepper motor to 8051 using ULN2003 is shown in fig. 3.14

The port P1 pins P1.0-P1.3 are connected to the ULN2003.The ULN2003A is a current driver IC. It is used to drive the current of the stepper motor as it requires more current. It consist seven array of Darlington pairs with common emitter. It has 16 pins in which 7 are input pins, 7 are output pins of the Darlington pair and remaining are common(Vcc) and enable (ground). The first four input pins of ULN2003 are connected to the microcontroller(P1.0-P1.3) and four output pins are connected to the stepper motor windings A,B,C,D and common terminals are connected to +5V.

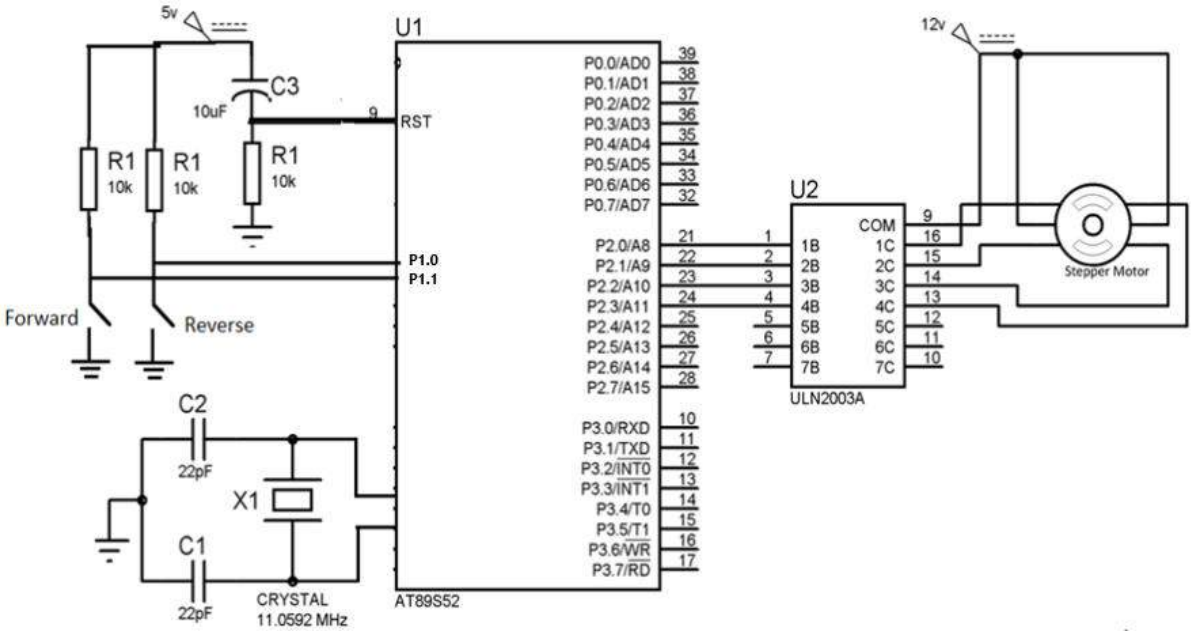


Fig.24: Stepper motor interfacing with 8051

```
#include<reg51.h>

#define stepper P2

void Delay_ms (unsigned char);

sbit forward = P1^0;

sbit backward = P1^1;
```

```
void main()
{stepper = 0x00H
forward=1;
backward=1;
while(1)
{stepper = 0x00H
if(forward==0)
{do
{stepper = 0x01H;
Delay_ms(50);
stepper = 0x02H;
Delay_ms(50);
 stepper = 0x04H;
Delay_ms(50);
 stepper = 0x08H;
Delay_ms(50);
} while(forward==0);
}
elseif(backward==0)
{ do
{stepper = 0x08H;
Delay_ms(50);
stepper = 0x04H;
Delay_ms(50);
stepper = 0x02H;
Delay_ms(50);
stepper = 0x01H;
Delay_ms(50)
} while(backward==0);
}}}
```

12. Interfacing of 4X4 matrix keyboard with 8051

The logic behind detecting a keypress using a microcontroller

To detect the key that's pressed, two ports of the microcontroller are connected to the rows and columns of the matrix respectively. The port which is connected to the rows of the matrix is configured as an output port, hence making each row logic 0. On the contrary, the port which is connected to the columns is configured as an input port, making the column at logic 1.

When a button is pressed, it changes the logic of that particular column to 0 because the button causes a short circuit between the row and the column. As an example, let us say the key with the number 1 is pressed. This causes the D1 column to go to logic 0 as it shorts row D0 with it.

For Port 2 the values are currently D3D2D1D0 = 1101. The 0 indicates that a button in the D1 column is pressed. So we have identified the column now. Next, we will identify the row. Looking at the picture above, you might be wondering that the same output would be produced if any of the keys in the column D1 were pressed. To find the row which has the pressed key, the microcontroller grounds the rows one by one and checks in which case the column is logic zero. This helps the microcontroller to find which key was pressed.

To summarize, when a key is pressed, the column containing the key is driven to logic 0. The microcontroller then checks each row and identifies the row where the button is pressed.

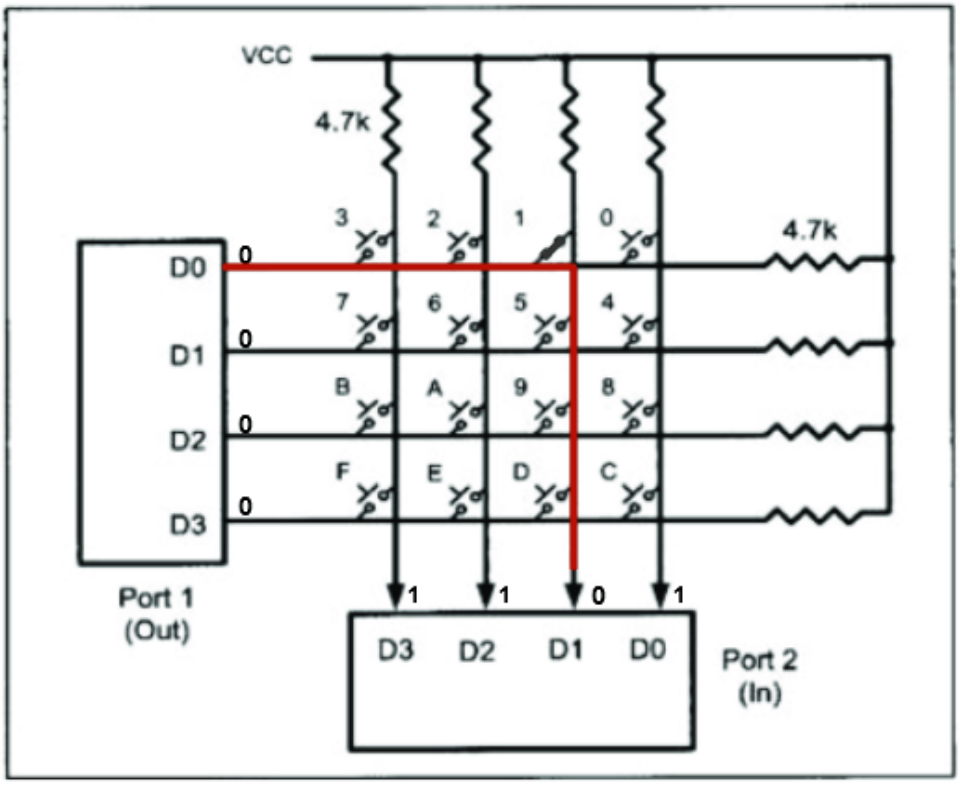


Fig.25: Keypad connection

Flowchart to detect a key press using a keypad

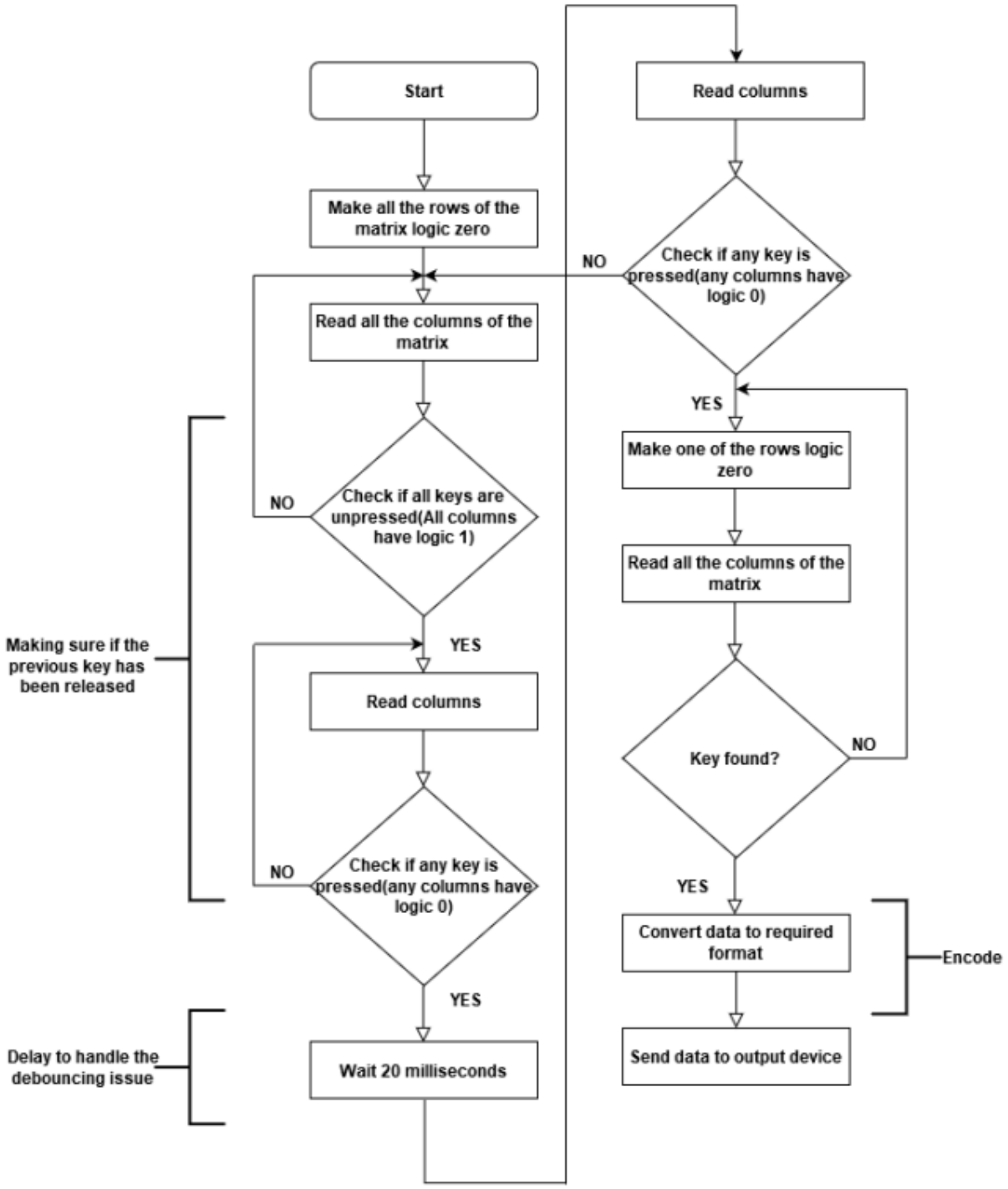


Fig.26: Flowchart to detect keypress

Algorithm to detect a key press using a keypad

- **Debouncing:** The main job of any switch is to connect two terminals to complete a circuit. But here is the issue, this process is not instantaneous. What this means is that it takes some time for the switch to complete the circuit. Due to this, the signal to the microcontroller is noisy until the contacts of the switch make proper contact with the underlying wires. Although this problem can be solved using capacitors and Schmitt triggers, it is better to use a software approach to solve the problem. To solve this problem, the microcontroller needs to wait for a certain amount of time and allow the signal to stabilize.

- **Making sure the previous key has been released**: The microcontroller needs to start scanning the matrix after a keypress has been detected. To do this, the microcontroller must make all the rows logic zero and see if all the columns have a logic 1. If all the columns have logic 1, it means that none of the keys are pressed, and the microcontroller needs to look for a new keypress.
- **Encoding the keypress:** detecting a keypress is one thing, but it is also essential to convert the information about the keypress into information that can be shown on an LCD or any other output peripheral for that matter. Here's a flowchart that will help you understand the entire process of detecting a keypress. It's simpler than it looks.

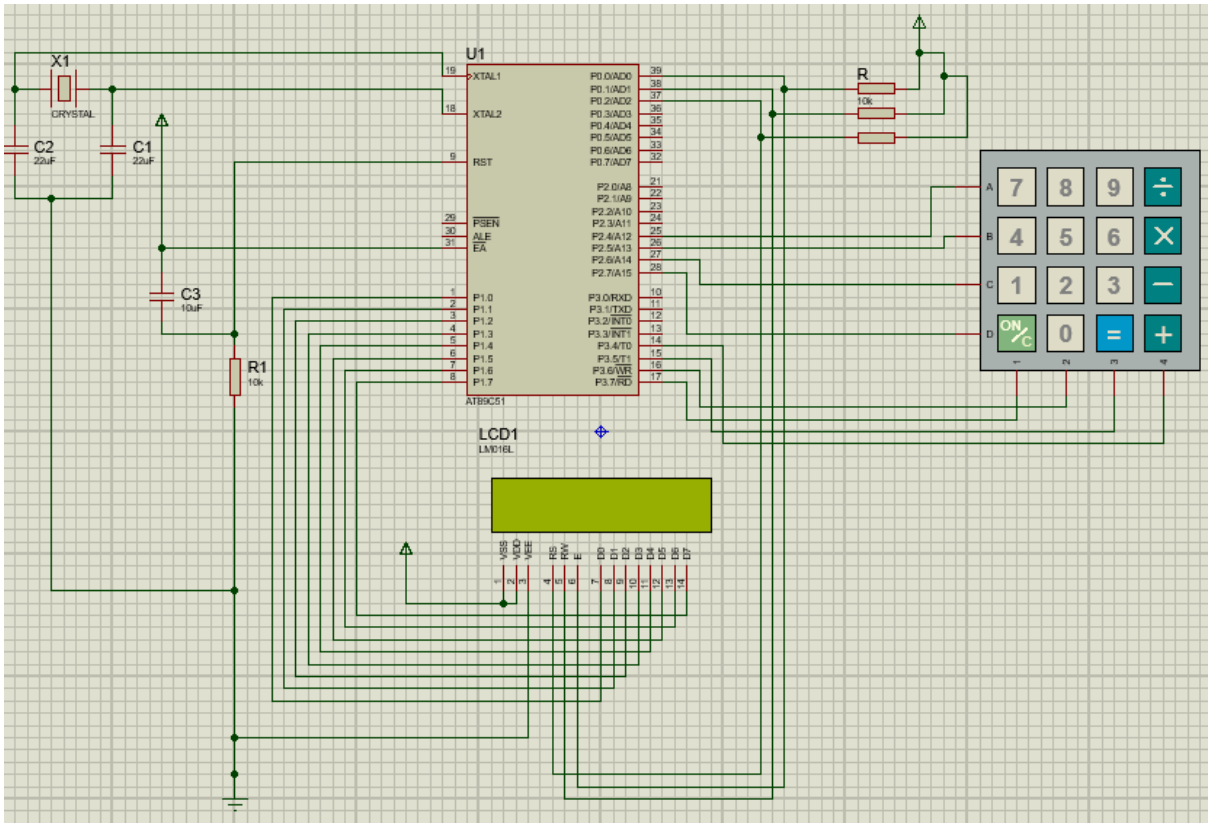


Fig.27: 4x4 keyboard interfacing with 8051

```
#include <reg51.h>
voidMSDelay(unsigned int );
voidLCDtransfer(unsigned char );
voidsend_command(unsigned char);
voidsend_data(unsigned char);
sfr COL = 0xB0;
sfr ROW = 0xA0;
sfrldata = 0x90; //P1 = LCD data pins
sbitrs = P0^2;
sbitrw = P0^1;
sbit en = P0^0;
unsigned char keypad[4][4] = {'7','8','9','/',
                      '4','5','6','x',
                      '1','2','3','-',
                     'e','0','=','+'};

void main()
{
unsigned char col_location, row_location;
COL = 0xFF;     // Sent port 3 as output port
while(1)
{
do
{
ROW =0X00;    //sets the row matrix as an input port
col_location = COL;
col_location &= 0x0F;
} while(col_location != 0x0F); // waits for one of the column's to be logic 0
do
{
do
{
MSDelay(20);
col_location = COL;
col_location &= 0x0F;
}while(col_location == 0x0F);
MSDelay(20);  // delay function to prevent debouncing
col_location = COL;
col_location &=0x0F;
```

```
}while(col_location == 0x0F);
while(1)          // finds which row the key belong to
{
ROW = 0XFE;
col_location = COL;
col_location &=0x0F;
if(col_location != 0x0F)
{
Row_location = 0;
break;
}
ROW = 0XFD;
col_location = COL;
col_location &=0x0F;
if(col_location != 0x0F)
{
row_location = 1;
break;
}
ROW = 0XFB;
col_location = COL;
col_location &=0x0F;
if(col_location != 0x0F)
{
row_location = 2;
break;
}
ROW = 0XF7;
col_location = COL;
col_location &=0x0F;
row_location = 3;
break;
}
send_command(0x38); // 16_bit mode; Enable 5x7 dot matrix for each character
MSDelay(250);
send_command(0x0E); // Display ON & cursor ON
MSDelay(250);
send_command(0x01); // Clear the Display
MSDelay(250);
send_command(0x06); // To shift the cursor right
MSDelay(250);
```

```
send_command(0x83); // Set the cursor at line 1, position 3
MSDelay(250);
if(col_location ==0x0E)
LCDtransfer(keypad[row_location][0]);
else if(col_location == 0x0D)
LCDtransfer(keypad[row_location][1]);
else if(col_location == 0x0B)
LCDtransfer(keypad[row_location][2]);
else
LCDtransfer(keypad[row_location][3]);
}
}

voidMSDelay(unsigned int value)
{
unsignedint x, y;
for(x=0;x<1275;x++)
for(y=0;y<value;y++);

}
voidLCDtransfer(unsigned char keypress)
{
ldata = keypress;
rs = 1;
rw = 0;
en = 1;
MSDelay(10);
en = 0;
return;
}
voidsend_command(unsigned char value)
{
ldata = value;
rs = 0;
rw = 0;
en = 1;
MSDelay(10);
en = 0;
return;
}
```