DEEPAK GUPTA

EM TESTED

DC Motor Control Using Keypad

Mr. WASIM

A Direct Current (DC) motor is a device that translates electrical pulses into mechanical movement. The speed of the motor depends on the three factors

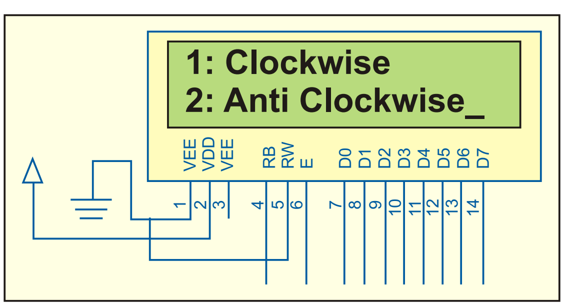
1)LOAD
2)VOLTAGE
2)CURRENT

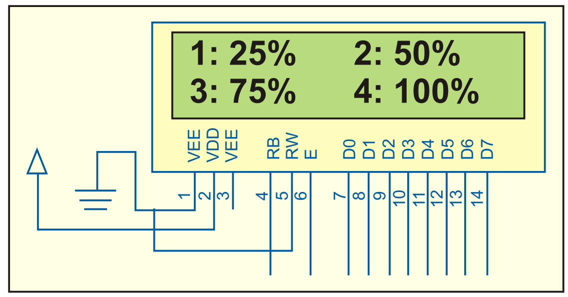
For a given fixed load , we can maintain a steady speed by using a method called PWM(Pulse Width Modulation) concept is used.Some controllers have inbuilt PWM circuitry. For microcontrollers without PWM circuitry ,we can create various duty cycle .

In a DC motor, we have only + and – leads. By reversing the polarity, the DC motor will move in the opposite direction. The maximum speed of a DC motor is indicated in RPM.The RPM is reduced when moving a load and it decreases. DC Motor have high voltage and current ratings.The nominal voltage can vary from 1 to 150V and the nominal current can vary from 25mA to a few ampere.

8051 cannot drive a motor because of its current and voltage ratings. Therefore to connect a motor to 8051, we need a driver.The driver provides sufficient current and voltage to turn on the motor. Also the motor draws huge current at starting,this may prove fatal from 8051.It is also being taken care by the driver.

In the circuit the motor is controlled by the keypad and all the instructions are displayed on LCD.The user is given choice to rotate the motor in clockwise or anticlockwise.Both the messages are displayed on the 16x2LCD.

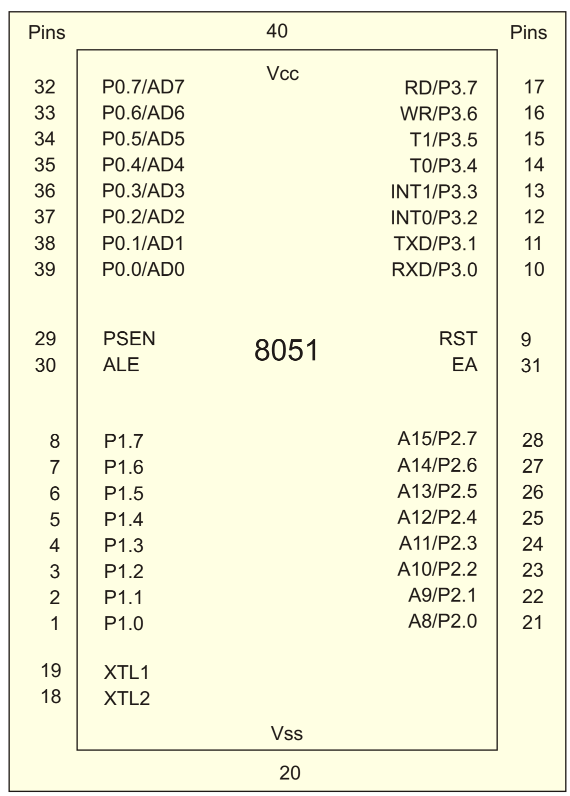


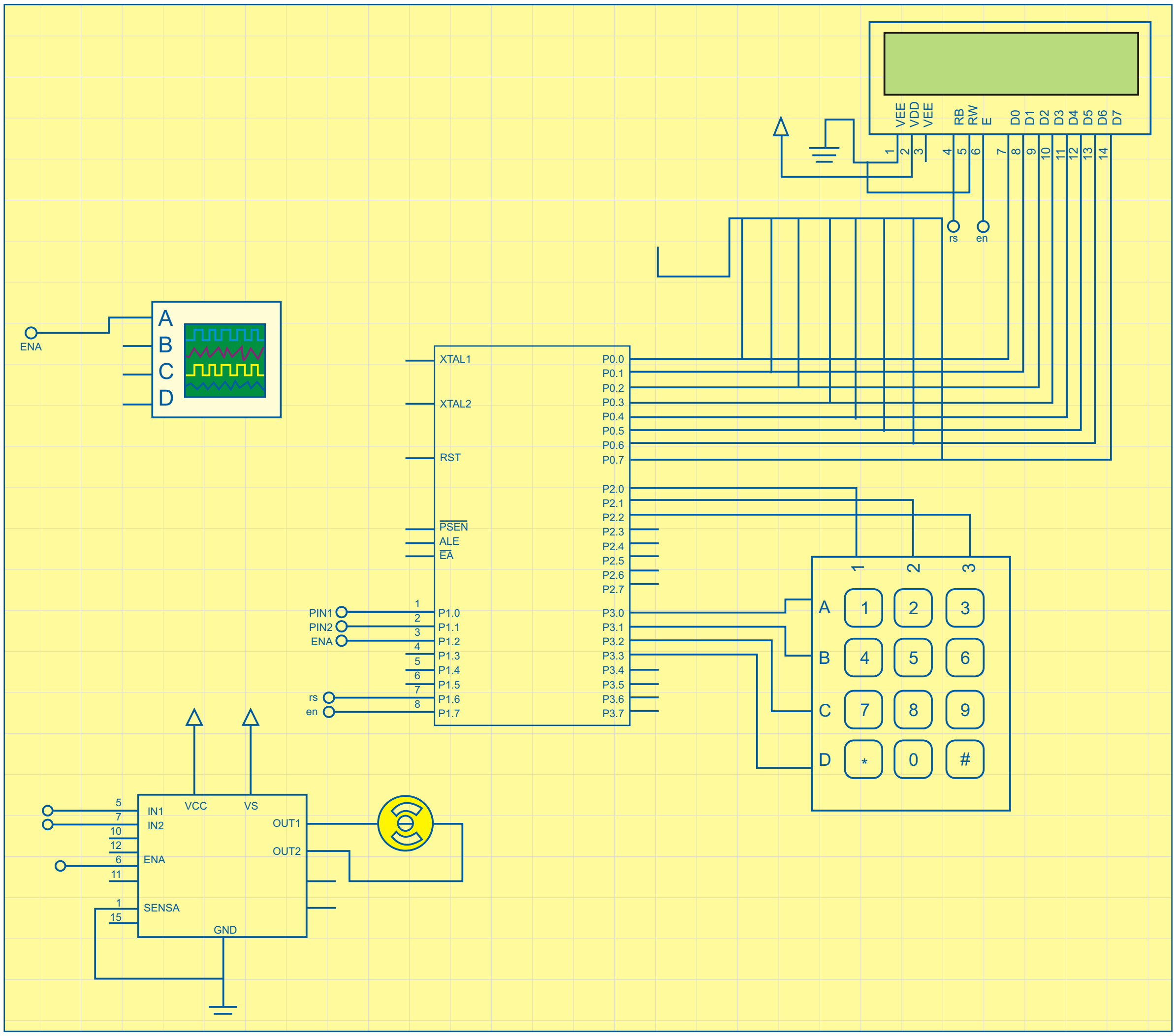


Depending upon the input from the keypad, the user is given four option to run the motor in different duty cycles.The options being 25%,50%,75% and 100%.

On receiving the respective input the motor runs with a steady speed.

The different components used are





89C51 Controller

The 8051 architecture consists of these specific features:

8 bit CPU with registers A (the accumulator) and B

16 bit program counter (PC) and data pointer (DPTR)

8 bit program status word (PSW)

RAM of 128 bytes and ROM of 4K

Four register banks, each containing 8 registers

32I/O pins organized as 4 ports P0-P3

Two 16 bit timers, T0,T1

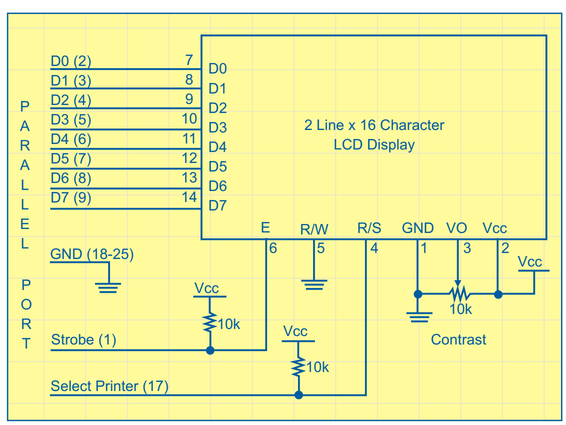
Two external and three internal interrupt sources

Oscillator and cloak circuit

Port 0 is connected to the 16x2 LCD. External pull up resistors of 4.7K are used, since port 0 do not contain the internal pull up resistor. Port 2 and 3 are connected to the columns and rows of the keypad respectively.Port 1 are connected to L298 driver. Also pin 6 and 7 of port 1 are used as control pins for LCD. The circuit is built on Proteus and so the crystal frequency has been set internally.

16x2 LCD

The LCD panel's Enable and Register Select is



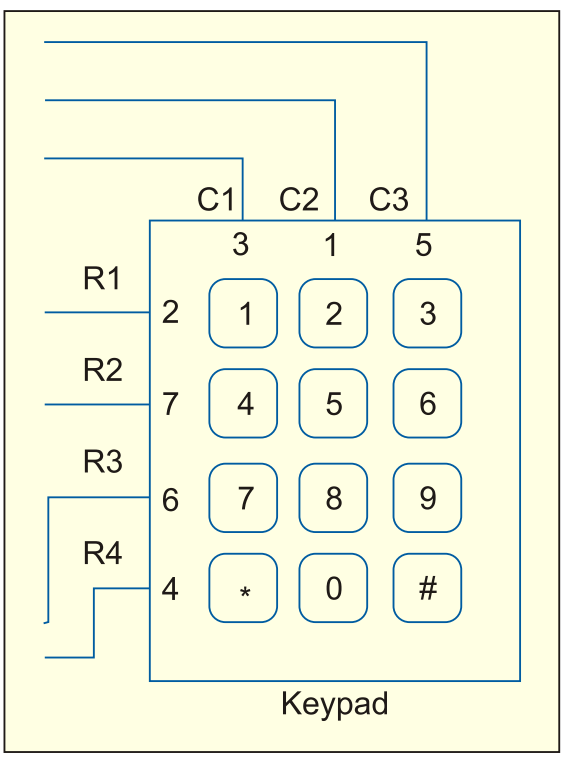
Pin	Function
1	Ground (Vss), 0V
2	Power (Vdd), +5V
3	Contrast Voltage (usually less than 1V)
4	"R/S" Register Select (1 for Data Write, 0 for Command Write)
5	"R/W" Read/Write (1 for Read, 0 for Write)
6	"EN" Enable line (Pulsing high latches a command or data _|¯|_)
7-14	Data Pins (D0-D7) D0 is LSB, in 4-bit mode only D4-D7 are used
15-16	(Optional) Back-light Anode and Cathode, NC, or Not There at All

connected to the Control Port. The Control Port is an open collector / open drain output. While most Parallel Ports have internal pull-up resistors, there are a few which don't. Therefore by incorporating the two 10K external pull up resistors, the circuit is more portable for a wider range of computers, some of which may have no internal pull up resistors. The 10k Potentiometer controls the contrast of the LCD panel.

The 2 line x 16 character LCD modules are available from a wide range of manufacturers and should all be compatible with the HD44780.

The pin out on most LCD's will be 14 to 16 pins in a single row with the standard 100 mil spacing. The 16 pin version has two extra pins to accommodate a back-light. However, sometimes the pins are present but not connected to anything.

There are two basic ways to interface the device: 8-bit mode and 4-bit mode. Most often, the "R/W" line is just tied to ground, and the LCD is only written to and not read. The read function is usually used to poll the "Busy Flag" which appears on D7 while the device is incapable of accepting a command (it's busy..get it). However, this function may be ignored by simply waiting the maximum amount of time for each command to complete (most are completed in less than 200us).

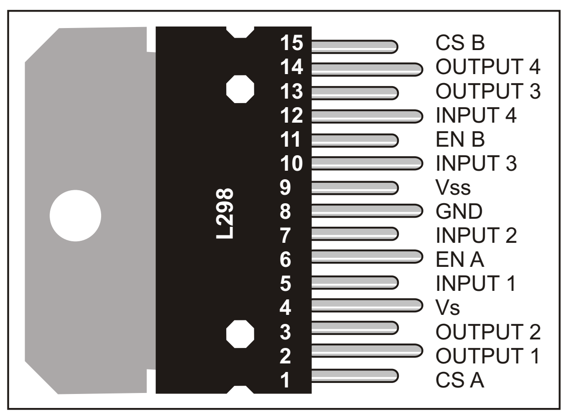


4x4 Keypad

The rows are connected to the output port and columns connected to the input port. If no key has been pressed ,reading the input port will yield '1' for all columns since they are all connected to Vcc. If all the rows are grounded and a key is pressed, one of the columns will have zero since all the keys provide path to ground. It is the function of the controller to scan the keyboard continuously to detect and identify the key pressed.

L298

The L298 is an integrated circuit motor driver that can be used for simultaneous, bidirectional control of two small motors. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to
enable or disable the device independently of the input signals.



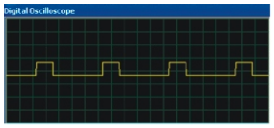
DC Motor

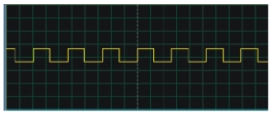
A DC motor wil have only + and – leads.Connecting them to a DC voltage source moves the rotor in one direction.By reversing the polarity,the DC motor will move in opposite direction.The maximum speed of motor is indicated in RPM and specified in datasheet.

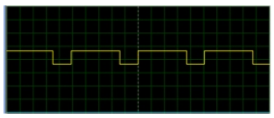
Program Code

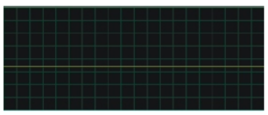
```
#include<reg51.h>
#define row P3
#define col P2
sbit pin1=P1^0;
sbit pin2=P1^1;
sbit rs= P1^6;
sbit en= P1^7;
sbit ENA=P1^2;
 void mdelay()
 {
          TMOD=0x10;
          TH1=0xFF;
          TL1=0xF6;
          TR1=1;
          while(TF1==0);
          TR1=0;
          TF1=0;
}

void delay(unsigned char itime)
{
          unsigned int i,j;
          for(i=0;i<itime;i++)
          for(j=0;j<1275;j++);
}
//////////////////////////////////LCD////////////////////
///


void lcdcmd(unsigned char value)
{
          P0=value;
          rs=0;
//        rw=0;
          en=1;
          delay(1);
          en=0;
}
void lcd_data(unsigned char value)
{
          P0=value;
          rs=1;
//        rw=0;
          en=1;
          delay(1);
          en=0;
}
void lcd_dis(unsigned char *ptr)
{   while(*ptr)
  lcd_data(*ptr++);
}
void lcd_init()
{
          lcdcmd(0x38);
          //mdelay(250);
          lcdcmd(0x0E);
          //mdelay(250);
          lcdcmd(0x01);
          //mdelay(250);
          lcdcmd(0x06);
          //mdelay(250);
          lcdcmd(0x80);
}


////////////////////KEYPAD///////////////////////////
unsigned int accept()
{
          unsigned char keypad[4][3]={    '1','2','3',

                                          '4','5','6',

                                          '7','8','9',

                                          '8','0','#'};
          unsigned char rowloc,colloc,temp;
                    col=0xff;

                              do{
                                        row=0x00;
                                        colloc=col;

colloc=colloc&0x07;
                        }while(colloc!=0x07);
                              delay(1);
                              do
                              {
                                        delay(1);
                                        colloc=col;
```

```
colloc=colloc&0x07;
                }while(colloc!=0x07);

        delay(1);                       while(1)
                {
                        r o w = 0 x 0 E ;
//grounding 1st pin
                        colloc=col;
                        colloc=colloc&0x07;
                        i f ( c o l l o c ! = 0 x 0 7 )
//scanning column
                        {
                                rowloc=0;
                                break;
                        }
                        r o w = 0 x 0 D ;
        //grounding 2nd pin
                        colloc=col;
                        colloc=colloc&0x07;
                        if(colloc!=0x07)
                        {
                                rowloc=1;
                                break;
                        }
                        row=0x0B;
                        colloc=col;
                        colloc=colloc&0x07;
                        if(colloc!=0x07)
                        {
                                rowloc=2;
                                break;
                        }
                        row=0x07;
                        colloc=col;
                        colloc=colloc&0x07;
                        if(colloc!=0x07)
                        {
                                rowloc=3;
                                break;
                        }
                } //while loop closed

                if (colloc==0x06)
                {
                                temp=(keypad[rowloc][0]);
                                pin2=1;
                        }
                        else if (colloc==0x05)
                        {
                                temp=(keypad[rowloc][1]);
                        }
                        else
                        {
                                temp=(keypad[rowloc][2]);
                        }                               return
temp;
}

void clockwise()
{
        unsigned char temp;
        lcd_dis("1:25% 2:50%");
        lcdcmd(0xc0);
        lcd_dis("3:75% 4:100%");
        temp=accept();
        lcdcmd(0x01);
        pin1=1;
        pin2=0;
        if (temp==0x31)
        {
                lcd_dis("25% DUTY CYCLE");
                while(1)
                {
                        ENA=1;
                        mdelay();
                        ENA=0;
                        mdelay();
                        mdelay();
                        mdelay();
                }
        }
        else if(temp==0x32)
        {
                lcd_dis("50% DUTY CYCLE");
                while(1)
                {
                        ENA=1;
                        mdelay();
                        ENA=0;
```









```
mdelay();
                }
}
else if(temp==0x33)
{
                lcd_dis("75% DUTY CYCLE");
                while(1)
                {
                ENA=1;
                mdelay();
                mdelay();
                mdelay();
                ENA=0;
                mdelay();
                }
        }
        else
        {
                lcd_dis("100% DUTY CYCLE");
                while(1)
                {
                        ENA=1;
                        pin1=1;
                        while(1)
                        mdelay();
                        }       }
}                               void
anticlockwise()
{
        unsigned char temp;
        lcd_dis("1:25% 2:50%");
        lcdcmd(0xc0);
        lcd_dis("3:75% 4:100%");
        temp=accept();
        lcdcmd(0x01);
        pin1=0;
        pin2=1;
        if (temp==1)
        {
                lcdcmd(0x80);
                lcd_dis("25% DUTY CYCLE");
                while(1)
                {
                        ENA=1;
                        mdelay();
                        ENA=0;
                        mdelay();
                        mdelay();
                        mdelay();
                }
        }
        else if(temp==2)
        {
                lcd_dis("50% DUTY CYCLE");
                while(1)
                {
                        ENA=1;
                        mdelay();
                        ENA=0;
                        mdelay();
                }
        }
        else if(temp==3)
        {
                lcd_dis("75% DUTY CYCLE");
                while(1)
                {
                        ENA=1;
                        mdelay();
                        mdelay();
                        mdelay();
                        ENA=0;
                        mdelay();
                }
        }
        else
        {
                lcd_dis("100% DUTY CYCLE");
                while(1)
                {
                        ENA=1;
                        mdelay();
                        }       }
}
/////////////////////MAIN//////////////////////////////
////
void main()
{
        unsigned char temp;
        P1=0x00;
        lcd_init();
        lcd_dis("1:Clockwise");
        lcdcmd(0xc0);
        lcd_dis("2:AntiClockwise");
        temp=accept();
                if (temp==0x31)
                {
                        lcdcmd(0x01);
                        lcdcmd(0x80);
                        lcd_dis("Clockwise");
                        delay(100);
                        lcdcmd(0x01);
                        clockwise();
                }
                else if(temp==0x32)
                {
                        lcdcmd(0x01);
                        lcd_dis("Anti Clockwise");
                        delay(100);
                        lcdcmd(0x01);
                        anticlockwise();
                }
                else lcd_dis("Wrong Choice");   }
```

The output can also be seen on the oscilloscope.
25%