**CHAPTER 04: 8051 INTERFACING APPLICATION**

**Preview:**

In this chapter students will be able to know how to interface different external devices like 7 segment LED display, 16x2 LCD display, stepper motor, 4x4 matrix keyboard, ADC and DAC with microcontroller 805. Interfacing diagram and program for interfacing using ‘Keil C’ language and simulation using proteus software. That is they will learn how these devices will behave in real life.

**4.1 Interfacing of seven segment display & LCD display Interfacing**

**Diagram & pin out of 2x16 LCD**

**4.1.1 Interfacing 7 Segment Display to 8051**

1.Seven segment displays are used to indicate numerical information. Seven segments display can display digits from 0 to 9 and even we can display few characters like A, b, C, H, E, e, F, etc.

2. Seven segment displays internally consist of 8 LEDs. In these LEDs, 7 LEDs are used to indicate the digits 0 to 9 and single LED is used for indicating decimal point.

3. Generally seven segments are two types, one is common cathode and the other is common anode.

In **common cathode,** all the cathodes of LEDs are tied together and labeled as com. and the anode are left alone.

In **common anode,** seven segment display all the anodes are tied together and cathodes are left freely.

Below figure shows the internal connections of seven segment Display.

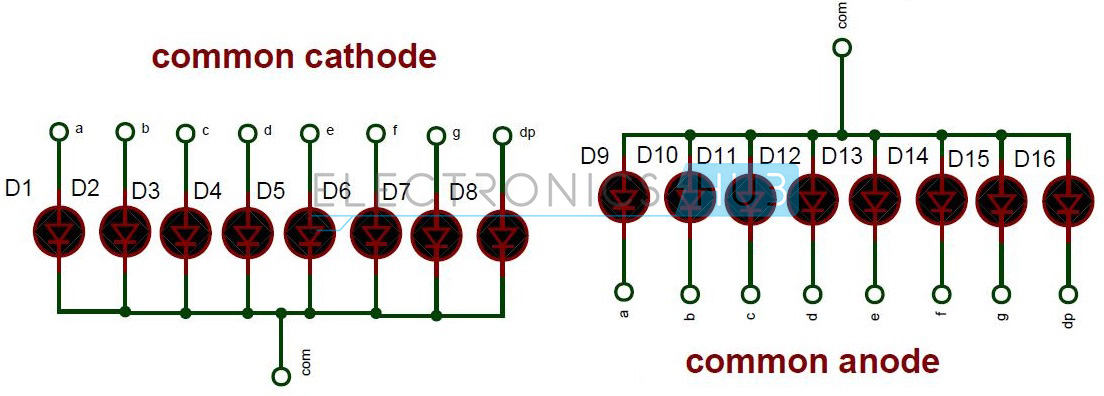
[](http://www.electronicshub.org/wp-content/uploads/2014/07/Internal-connections-of-seven-segment.jpg)

Fig 4.1 Internal Connections of Seven Segment

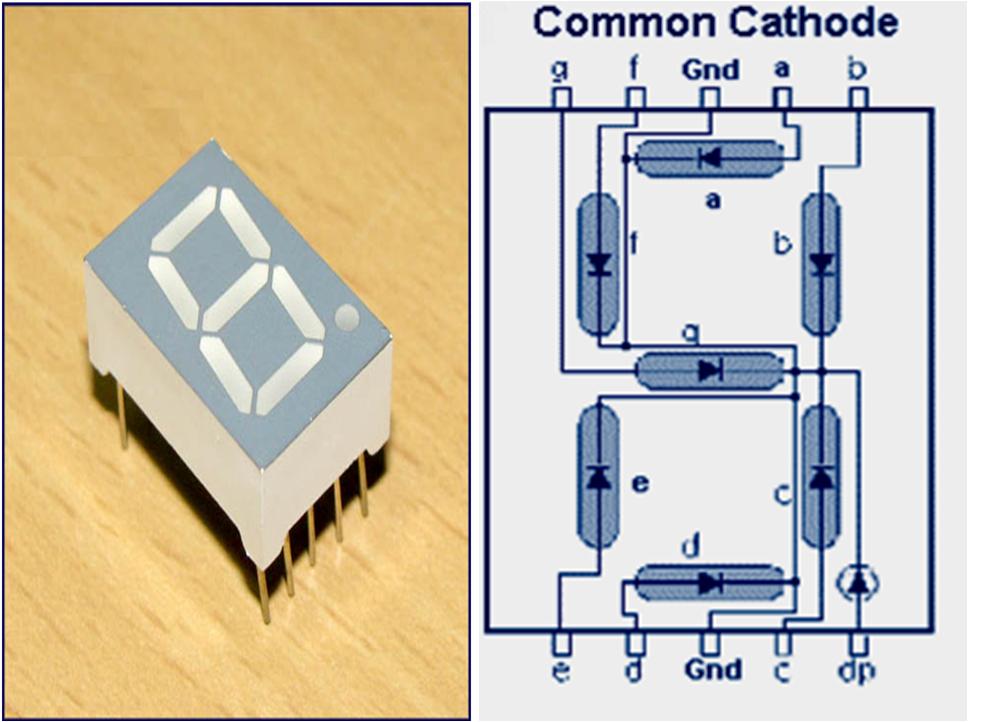
[](http://www.electronicshub.org/wp-content/uploads/2014/07/Common-cathode-7-segment-Display.jpg)

Fig 4.2 Common Cathode

**4.1.1.1 Digit Drive Pattern:**

To display the digits on 7 segment, we need to glow different logic combinations of segments. For example if you want to display the digit 3 on seven segment then you need to glow the segments a, b, c, d and g.

The below table show you the Hex decimal values what we need to send from PORT2 to Display the digits from 0 to 9.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Digit** | **Dp** | **g** | **f** | **e** | **d** | **c** | **b** | **a** | **Hex value** |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0x3f |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0x06 |
| 2 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0x5b |
| 3 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0x4f |
| 4 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0x66 |
| 5 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0x6d |
| 6 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0x7d |
| 7 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0x07 |
| 8 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0x7f |
| 9 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0x67 |

Table 4.1 Hex decimal values

**4.1.1.2 Algorithm to display values :**

1. First initialize all the segment hex values of the digits in an array.

unsigned char arr[10]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x67};

1. Now take for loop and assign array values to the PORT2 with some time delay.

for (i=0;i<10;i++)

                        {

                                 P2=arr[i];

                                    delay\_ms(500);

                        }

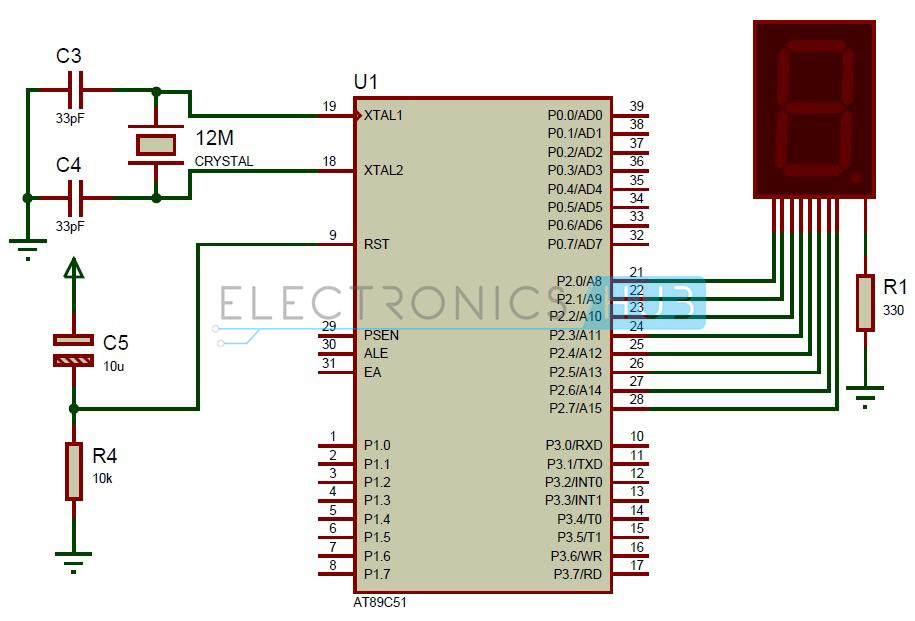
**[](http://www.electronicshub.org/wp-content/uploads/2014/07/Interfacing-7-segment-display-to-8051.jpg)**

Fig 4.3 Interfacing diagram of 7 segment display

**Circuit Components:**

* AT89C51 microcontroller
* AT89C51 programming board
* programming cable
* 12V DC battery or adaptor
* Common cathode 7 segment Display
* Resistors – 10k, 330 ohm
* 2 Ceramic capacitors – 33pF
* 12 MHz crystal
* Electrolytic capacitor – 10uF
* Connecting wires.

**4.1.1.3 How to Operate Interfacing 7 Segment Display to 8051 Circuit?**

1. Initially burn the program to the microcontroller
2. Give the connections as per the circuit diagram
3. Make sure that a to g pins of 7 segment are connected to the P2.0 to P2.6 respectively.
4. Switch on the supply, you can observe that digits 0 to 9 will display continuously with some delay.
5. Switch of the supply.

**4.1.1.4 Interfacing 7 Segment Display to 8051 Circuit Applications:**

* Seven segments are widely used in digital clocks to display the time.
* These are used in electronic meters for displaying the numerical information.
* Used in Instrument panels
* Used in digital readout displays.

**4.1.1.5 Limitations of the Circuit:**

* The complexity is increased to display large information.
* It is not possible to display the symbols on seven segment.

**Program:**

**Aim:** Write a program to display 0-9 continuously on 7 segment LED display.

#include<regx51.h>

void msdelay(unsigned int);

void main()

{

unsigned int arr[10]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,

0x7d,0x07,0x7f,0x67};

P1=0x00;

while(1)

{

for(i=1;i<=10;i++)

{ P2=arr[i];

msdelay(50); }

}

}

void msdelay(unsigned itime)

{ unsigned int x,y;

for(x=0;x<=itime;x++)

for(y=0;y<=1275;y++);

}

**4.1.2 LCD INTERFACING**

**4.1.2.1 LCD Operation**

By replacing LEDs now a days LCD are widely used because it has good features like

* The declining prices of LCD, the ability to display numbers, characters, and graphics
* Ease of programming for characters and graphics 16×2 character lcd display is very basic module which is commonly used in electronics devices and projects. It can display 2 lines of 16 characters.
* LCD can be interfaced with microcontroller in 4 Bit or 8 Bit mode. These differs in how data is send to LCD.
* In 8 bit mode to write a character, 8 bit ASCII data is send through the data lines D0 – D7 and data strobe is given through E of the LCD. LCD commands which are also 8 bit are written to LCD in similar way.
* But 4 Bit Mode uses only 4 data lines D4 – D7. In this mode 8 bit character ASCII data and command data are divided into two parts and send sequentially through data lines. The idea of 4 bit communication is used to save pins of microcontroller.
* 4 bit communication is a bit slower than 8 bit communication but this speed difference can be neglected since LCDs are slow speed devices. Thus 4 bit mode data transfer is most commonly used.

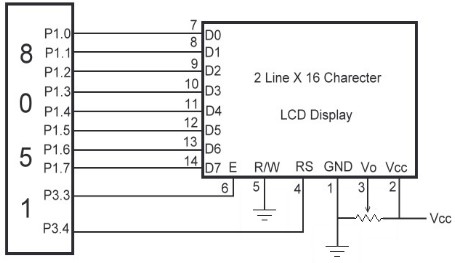
**Descriptions for LCD**

Fig 4.4 LCD Display using 8- bit mode

**4.1.2.2 Pin Description of 16x2 LCD**

**Pin Symbol I/O Descriptions**

1 VSS -- Ground

2 VCC -- +5V power supply

3 VEE -- Power supply to control contrast

4 RS I RS=0 to select command register,

RS=1 to select data register

5 R/W I R/W=0 for write,

R/W=1 for read

6 E I/O Enable

7 DB0 I/O The 8-bit data bus

8 DB1 I/O The 8-bit data bus

9 DB2 I/O The 8-bit data bus

10 DB3 I/O The 8-bit data bus

11 DB4 I/O The 8-bit data bus

12 DB5 I/O The 8-bit data bus

13 DB6 I/O The 8-bit data bus

14 DB7 I/O The 8-bit data bus

* So by reading the above table you can get a idea how to display a character.
* For displaying a character you should enable the enable pin (pin 6) by giving a pulse of 450ns,
* After enabling the pin6 you should select the register select pin (pin4) in write mode.
* To select the register select pin in write mode you have to make this pin high (RS=1), after selecting the register select you have to configure the R/W to write mode that is R/W should be low (R/W=0).

**4.1.2.3 Follow these simple steps for displaying a character of data**

* E= 1; enable pin should be high
* RS= 1; Register select should be high
* R/W= 0; Read/write pin should be high.

**4.1.2.4 To send a command to the LCD just follows these steps.**

* E=1; enable pin should be high
* RS=0; Register select should be low
* R/W =1; Read /write pin should be high.



Table 4.2 LCD Command Codes

**Aim:** Write a Program to display “WELCOME” on 2x16 LCD.

**Solution:**

#include<regx51.h>

sbit rs=P2^0;

sbit rw=P2^1;

sbit en=P2^2;

void delay(unsigned int);

void lcdcmd(unsigned char);

void lcddata(unsigned char);

void main()

{

lcdcmd(0x38);

delay(50);

lcdcmd(0x0E);

delay(50);

lcdcmd(0x01);

delay(50);

lcdcmd(0x80);

delay(50);

lcdcmd(0x06);

delay(50);

lcddata('W');

delay(50);

lcddata('E');

delay(50);

lcddata('L');

delay(50);

lcdcmd(0xC0);

delay(50);

lcdcmd(0x06);

delay(50);

lcddata('C');

delay(50);

lcddata('O');

delay(50);

lcddata('M');

delay(50);

lcddata('E');

while(1);

}

void lcdcmd(unsigned char c)

{

rs=0;

rw=0;

P3=c;

en=1;

delay(10);

en=0;

}

void lcddata(unsigned char dt)

{

rs=1;

rw=0;

P3=dt;

en=1;

delay(10);

en=0;

}

void delay(unsigned int t)

{

unsigned int x,y;

for(x=0;x<=t;x++)

for(y=0;y<=1275;y++);

}

**4.2 Interfacing of 4x4 Keyboard, ADC & DAC- interfacing diagram & programming.**

**4.2.1 KEYBOARD INTERFACING**

* Keyboards are organized in a matrix of rows and columns. The CPU accesses both rows and columns through ports.
* Therefore, with two 8-bit ports, an 8 x 8 matrix of keys can be connected to a microprocessor
* When a key is pressed, a row and a column make a contact Otherwise, there is no connection between rows and columns.
* A 4x4 matrix connected to two ports. The rows are connected to an output port and the columns are connected to an input port

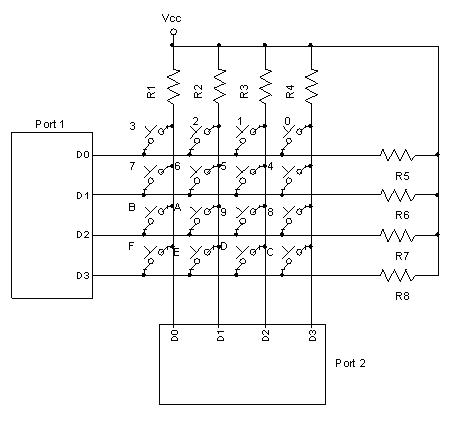
****

Fig 4.5: Matrix Keyboard Connection to ports

* If no key has been pressed, reading the input port will yield 1s for all columns since they are all connected to high (Vcc).
* If all the rows are grounded and a key is pressed, one of the columns will have 0 since the key pressed provides the path to ground.
* It is the function of the microcontroller to scan the keyboard continuously to detect and identify the key pressed. To detect a pressed key, the microcontroller grounds all rows by providing 0 to the output latch, then it reads the columns.
* If the data read from columns is D3 – D0 = 1111, no key has been pressed and the process continues till key press is detected. If one of the column bits has a zero, this means that a key press has occurred.
* For example, if D3 – D0 = 1101, this means that a key in the D1 column has been pressed
* After detecting a key press, microcontroller will go through the process of identifying the key

Starting with the top row, the microcontroller grounds it by providing a low to row D0 only

* It reads the columns, if the data read is all 1s, no key in that row is activated and the process is moved to the next row. It grounds the next row, reads the columns, and checks for any zero
* This process continues until the row is identified. After identification of the row in which the key has been pressed.

**For Example:** From Figure identify the row and column of the pressed key for each of the following.

(a) D3 – D0 = 1110 for the row, D3 – D0 = 1011 for the column

(b) D3 – D0 = 1101 for the row, D3 – D0 = 0111 for the column

**Solution :**

From Figure the row and column can be used to identify the key.

(a) The row belongs to D0 and the column belongs to D2; therefore, key number 2 was pressed.

(b) The row belongs to D1 and the column belongs to D3; therefore, key number 7 was pressed.



Fig 4.6: To check which key is pressed

Aim : Write a Program to Interface 4x4 matrix keyboard with 8051 microcontroller.

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* KEYPAD

// In this program the keypad is connected to the port 1.

// All the rows are connected to P1.0 to P1.3 and columns are connected to P1.4 to P1.7.

// In this we have two functions

// 1. key\_pressed();

// 2. key\_check();

// 1. key\_pressed() function will wait for the key stroke and it will return corresponding key value.

// 2. key\_check() is a function which checks whether key is pressed or not.

// If any key is pressed,it will return that particular key value else it will return 0.

//

#include<reg51.h>

#include<stdio.h>

#include<lcd.h>

#include<lcd.c>

#define STRING " Gramin Polytechnic "

#define KEYPORT P1

unsigned char code keyboard[4][4] = {'C','B','A','/',

'E','E','D','\*',

'H','G','F','-',

'n','0','=','+'}; //keypad values.

unsigned char keydata, row, column, position=1,key\_value,key;

unsigned char key\_check();

unsigned char get\_key();

void delay(unsigned int );

void main(void)

{

lcd\_init(); //Initializing LCD

lcd\_clear( ALL ); //Clear the total LCD display

lcd\_gotoxy( 1,LINE1 ); //Move the LCD cursor position to 1st of the LINE 1

lcd\_printf( STRING ); //Display the message on the LCD

lcd\_gotoxy( 1,LINE2 ); //Move the LCD cursor position to 1st of the LINE 2

lcd\_cursor( OFF ); //Display cursor on the LCD

KEYPORT |= 0xf0; //Making all columns as input

while(1)

{

do {

KEYPORT &= 0xf0; //Connecting all rows to ground

keydata = KEYPORT;

keydata &= 0xf0;

}while(keydata != 0xf0); //Checking all keys are released or not

do

{

delay(2);

keydata = KEYPORT;

keydata &= 0xf0;

}while(keydata == 0xf0); //Checking whether key is pressed or not

key\_value = get\_key();

lcd\_send(DATA,key\_value);

}

//KEY\_CHECK FUNCTION

unsigned char key\_check()

{

KEYPORT &= 0xf0; //Connecting all rows to ground

do

{

keydata = KEYPORT;

keydata &= 0xf0;

}while(keydata != 0xf0); //Checking all keys are released or not

delay(2);

keydata = KEYPORT;

keydata &= 0xf0;

if(keydata == 0xf0) //Checking whether key is pressed or not

return(0);

else

return(get\_key());

}

//\* GET\_KEY FUNCTION \*

unsigned char get\_key()

{

while(1)

{

KEYPORT=0xfe; //Checking whether the key is from 1st row or not

keydata = KEYPORT;

keydata &= 0xf0;

if(keydata != 0xf0)

{

row=0;

goto next;

}

KEYPORT = 0xf0;

KEYPORT=0xfd; //Checking whether the key is from 2nd row or not

keydata = KEYPORT;

keydata &= 0xf0;

if(keydata != 0xf0)

{

row=1;

goto next;

}

KEYPORT = 0xf0;

KEYPORT=0xfb; //Checking whether the key is from 3rd row or not

keydata = KEYPORT;

keydata &= 0xf0;

if(keydata != 0xf0)

{

row=2;

goto next;

}

KEYPORT = 0xf0;

KEYPORT=0xf7; //Checking whether the key is from 4rth row or not

keydata = KEYPORT;

keydata &= 0xf0;

if(keydata != 0xf0)

{

row=3;

goto next;

}

break;

}

next: //checking for column position

if(keydata==0xe0)

column=0;

else

if(keydata==0xd0)

column=1;

else

if(keydata==0xb0)

column=2;

else

if(keydata==0x70)

column=3;

return(keyboard[row][column]);

}

// DELAY FUNCTION \*\*\*\*\*\*\*\*

void delay(unsigned int delay\_time) //creating delay

{

unsigned int reload,loop;

for(reload = 1; reload <= delay\_time; reload++)

for(loop = 1; loop <= 1275; loop++);

}

**4.2.2 ADC & DAC- interfacing**

ADCs (analog-to-digital converters) are among the most widely used devices for data acquisition

* A physical quantity, like temperature, pressure, humidity, and velocity, etc., is converted to electrical (voltage, current) signals using a device called a transducer, or sensor
* We need an analog-to-digital converter to translate the analog signals to digital numbers, so microcontroller can read them

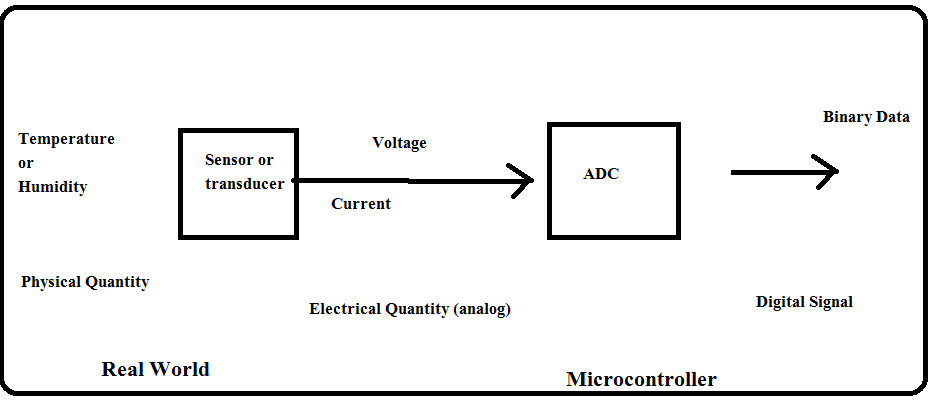


Fig 4.7: Need of ADC

**4.2.2.1 Types of ADC**

There are two types of ADC

1. Parallel ADC
2. Serial ADC

* **Parallel ADC**

In this 8 or more pins dedicated to bring out the binary [digital] data.

1. ADC 0804IC
2. ADC0808IC

**4.2.2.2 ADC808/809 Chip**

* ADC808 has 8 analog inputs
* It allows us to monitor up to 8 different transducers using only a single chip
* The chip has 8-bit data output just like the ADC804
* The 8 analog input channels are multiplexed and selected according to table below using three address pins, A, B, and C

**4.2.2.3 ADC808 Analog Channel Selection**

* **Selected Analog 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

* **ADC808/809**



Fig 4.8 ADC 0808/0809

**Steps to Program ADC808/809**

1. Select an analog channel by providing bits to A, B, and C addresses
2. Activate the ALE pin, It needs an L-to-H pulse to latch in the address
3. Activate SC (start conversion ) by an H-to-L pulse to initiate conversion
4. Monitor EOC (end of conversion) to see whether conversion is finished
5. Activate OE (output enable ) to read data out of the ADC chip, An H-to-L pulse

to the OE pin will bring digital data out of the chip

**Interfacing diagram**

**Program**

#include<regx51.h>

sbit rs=P0^2;

sbit rw=P0^1;

sbit e=P0^0;

sbit ALE=P2^4;

sbit OE=P2^5;

sbit SC=P2^6;

sbit EOC=P2^7;

sbit Add\_A=P2^0;

sbit Add\_B=P2^1;

sbit Add\_C=P2^2;

void delay (unsigned int);

void lcdcmd (unsigned char);

void lcddta (unsigned char);

void hex2ascii(unsigned char);

unsigned char adc\_value;

void main(void)

{

rs=0;

rw=0;

e=0;

lcdcmd(0x38);

delay(10);

lcdcmd(0x0e);

delay(10);

lcdcmd(0x06);

delay(10);

P1=0xff;

EOC=1;

ALE=0;

OE=0;

SC=0;

while(1)

{

lcdcmd(0x85);

delay(10);

Add\_A=0;

Add\_B=0;

Add\_C=0;

delay(1);

ALE=1;

delay(2);

SC=1;

delay(1);

ALE=0;

SC=0;

while(EOC==1);

while(EOC==0);

OE=1;

delay(1);

adc\_value=P1;

OE=0;

hex2ascii(adc\_value);

}

}

void hex2ascii(unsigned char dt)

{

unsigned char x,d1,d2,d3;

x=dt/10;

d1=(dt%10)+0x30;

d2=(x%10)+0x30;

d3=(x/10)+0x30;

lcddta(d3);

lcddta(d2);

lcddta(d1);

}

void lcdcmd (unsigned char cmd)

{

rs=0;

rw=0;

P3=cmd;

e=1;

delay(2);

e=0;

}

void lcddta(unsigned char dt)

{

rs=1;

rw=0;

P3=dt;

e=1;

delay(2);

e=0;

}

void delay (unsigned int t)

{

unsigned int x,y;

for(x=0;x<=t;x++)

for(y=0;y<=1275;y++);

}

**DAC Interfacing [Digital to analog converter]**

This device is used to convert digital pulse to analog signal.

The digital pulses are converted to current.

By connecting resistor to the pin we can convert the result to voltage.

The total current provided by pin is a function of the binary numbers at the D0-D7 inputs of the DAC 0808 and reference current (Iref) and is as follows.

)

Where D0iste LSB ,D7 is the MSB for the input and Iref is the input current that must be applied to pin 14. The Iref current is generally set to 2.0 mA. Figure shows the generation of current reference (setting Iref=2mA) by using the standard 5V power supply and 1K and 1.5K-ohm standard resistors.

Converting Iout to voltage in DAC0808

Ideally we connect the output pin Iout to resistor convert this voltage and monitor the output on the scope. In real life, however this can cause inaccuracy since the input resistance of the load where it is connected will also affect the output voltage.

For this reason, the Iref current output is isolated by connecting it to op-amp such as the 714 with Rf=5K ohms for feedback resistor. Assuming the binary input.

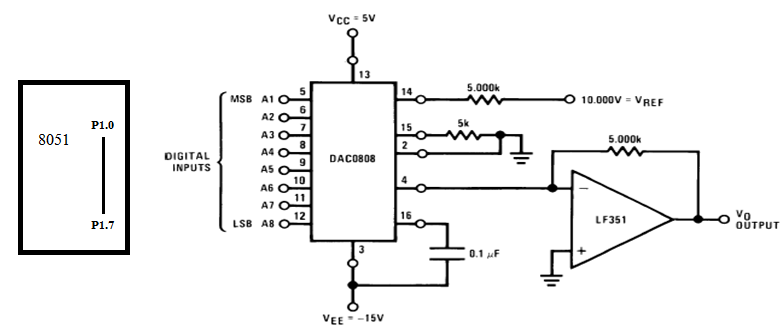


Fig 4.9: DAC Interfacing with 8051

**Program**

Write a Program for interfacing DAC to generate Saw tooth, Square wave, triangular wave, staircase wave.

#include<regx51.h>

void delay (unsigned int);

void main ()

{

unsigned char x;

while (1)

{

P2=0x00;

delay (1);

P2=0xFF;

delay (1);

}

}

void delay(unsigned int t)

{

unsigned int i,j;

for(i=0;i<=t;i++)

for(j=0;j<=1275;j++);

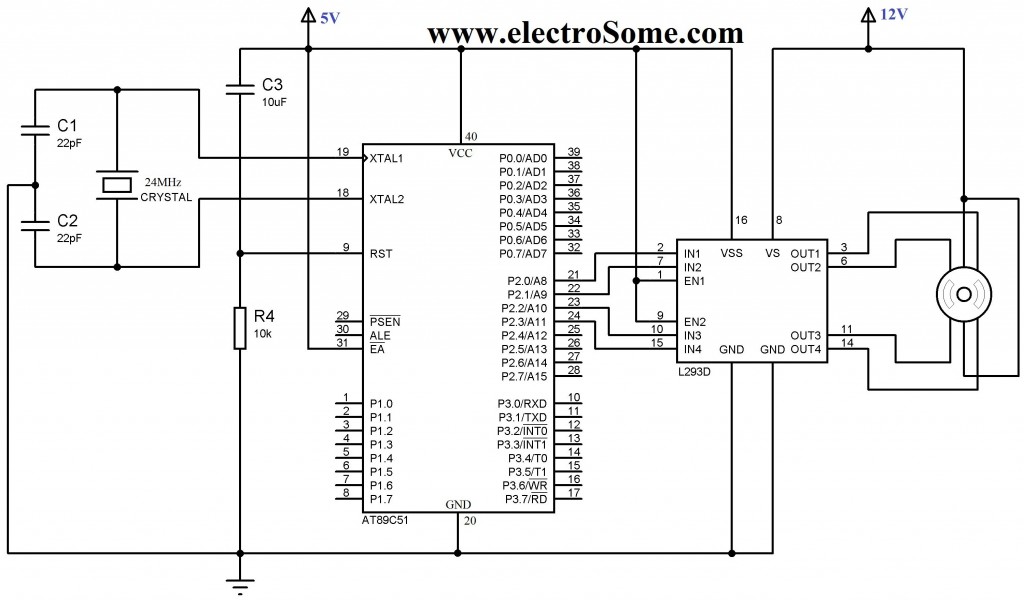
}

**4.3 Interfacing of stepper motor- interfacing diagram & Programming**

**Function**

**Stepper Motor Control using Microcontroller Circuit Principle:**

The main principle of this circuit is to rotate the stepper motor step wise at a particular step angle. The ULN2003 IC is used to drive the  stepper motor  as the controller cannot provide current required by the motor.

[](https://electrosome.com/wp-content/uploads/2013/05/Interfacing-Unipolar-Stepper-Motor-with-8051-using-L293D.jpg)

Stepper motor has 6 pins. In these six pins, 2 pins are connected to the supply of 12V and the remaining are connected to the output of the stepper motor. Stepper rotates at a given step angle. Each step in rotation is a fraction of full cycle. This depends on the mechanical parts and the driving method.

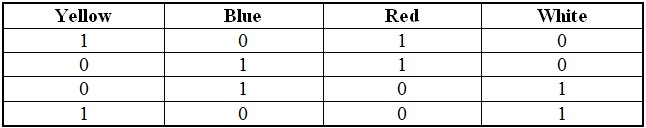
Similar to all the motors, stepper motors will have stator and rotor. Rotor has permanent magnet and stator has coil. The basic stepper motor has 4 coils with 90 degrees rotation step. These four coils are activated in the cyclic order. The below figure shows you the direction of rotation of the shaft. There are different methods to drive a stepper motor. Some of these are explained below.

**Full Step Drive:**In this method two coils are energized at a time. Thus, here two opposite coils are excited at a time.

**Half Step Drive:**In this method coils are energized alternatively. Thus it rotates with half step angle. In this method, two coils can be energized at a time or single coil can be energized. Thus it increases the number of rotations per cycle. It is shown in the below figure.

**How to Operate this Stepper Motor Driver Circuit?**

* Initially , switch on the circuit.
* Microcontroller  starts driving the stepper motor.
* One can observe the rotation of the stepper motor
* The stepper motor has four wires. They are yellow, blue, red and white. These are energized alternatively as given below.
* In full step driving, use the following sequence

[](http://www.electronicshub.org/wp-content/uploads/2014/07/Full-Step-Driving.jpg)

* To drive the motor in half step angle, use the following sequence

[](http://www.electronicshub.org/wp-content/uploads/2014/07/Half-Step-Angle.jpg)

**Stepper Motor Controller Circuit Advantages:**

* It consumes less power.
* It requires low operating voltage

**Stepper Motor Control Applications:**

* This circuit can be used in the robotic applications.
* This can also be used in mechantronics applications.
* The stepper motors can be used in disk drives, matrix printers, etc.

A Stepper Motor is a brushless, synchronous DC Motor. It has many applications in the field of robotics and mechatronics. The total rotation of the motor is divided into steps. The angle of a single step is known as the stepper angle of the motor. There are two types of stepper motors **Unipolar** and **Bipolar**..

Unipolar stepper motors can be used in three modes namely the **Wave Drive**, **Full Drive** and **Half Drive** mode. Each drive have its own advantages and disadvantages

#include <REG51.H>

//#include<INTEL\8051.H>

//#include<STANDARD.H>

#include<stdio.h>

void delay\_ms(unsigned int );

# define PHASEA 0x09

# define PHASEB 0x0c

# define PHASEC 0x06

# define PHASED 0x03

void main()

{

while(1)

{

P0=PHASEA;

delay\_ms(210);

P0=PHASEB;

delay\_ms(210);

P0=PHASEC;

delay\_ms(210);

P0=PHASED;

delay\_ms(210);

}

}

void delay\_ms(unsigned int i )

{

unsigned int ii,j;

for(ii=0;ii<i;ii++)

for(j=0;j<1275;j++);

}

**QUESTION BANK**

**4 Marks**

1. **State the condition of RS, RW & E pins while sending data on LCD.**
2. **How to interface LCD with 8051μc? Draw an interfacing diagram and explain it.**
3. **How to interface 7 segment LED with 8051μc? Draw an interfacing diagram and explain it.**
4. **How to interface stepper motor with 8051μc? Draw an interfacing diagram and explain it.**