LECTURE NOTES (E- CONTENTS) for

B.Sc. II Electronics (2021-22)

Semester: IV Paper- IV DSC -1005 D

Advance Communication and Microcontroller 8051

Section II: Microcontroller 8051

Unit- 4: Interfacing of Devices with 8051

Prepared and Circulated for

B .Sc. II Electronics Students

BY

Dr. C. B. Patil

Associate Professor, Department of Electronics

Vivekanand College (Autonomous), Kolhapur

(For Private Circulation only)

UNIT 4
Interfacing of Devices with 8051

Syllabus: Introduction to embedded C, comparison of C and assembly, Data types in C, SFR accessing , I/O programming, logical operations in C. C language programming: Program to generate square wave on port pin, Interfacing of LED , Opto-coupler, Switch, Relay, DC motor and Stepper motor.

What is an Embedded System?

An Embedded System can be best described as a system which has both the hardware and software and is designed to do a specific task.

- Biomedical Instrumentation – ECG Recorder, Blood cell recorder, patient monitor system
- Communication systems – pagers, cellular phones, cable TV terminals, fax and transreceiver, video games.
- Peripheral controllers of a computer – Keyboard controller, DRAM controller, DMA controller, Printer controller, LAN controller, disk drive controller.
- Industrial Instrumentation – Process controller, DC motor controller, robotic systems, CNC machine controller, close loop engine controller, industrial moisture recorder and controller.
- Scientific – digital storage system, CRT display controller, spectrum analyzer.

What is an Embedded C?

Embedded C is one of the most popular and most commonly used Programming Languages in the development of Embedded Systems.

C Language :

- C is a general-purpose programming language, which is widely used to design any type of desktop-based applications.
- It was developed by **Dennis Ritchie** as a system programming language to develop the operating system.
- The main features of C language include low-level access to memory, a simple set of keywords, and clean style, these features make C language suitable for system programming's like OS or compiler development.

Embedded C:

- Embedded C is an extension of C language
- used to develop micro-controller based applications
- The extensions in the Embedded C language from normal C Programming Language is the I/O Hardware Addressing, fixed-point arithmetic operations, accessing address spaces, etc.

Differences between C and Embedded C

	C programming	Embedded C programming
1	C is a general purpose programming language, which can be used to design any type of desktop based applications.	Embedded C is an extension of C language (some of the features are there, which can be used to specific purposes), it is used to develop micro-controller based applications (low-level or/and application level).
2	While, writing a C programming language code there is no need to know about computer hardware i.e. C language is not hardware dependent language.	You must have good knowledge about the hardware for that you're developing any code. Embedded C is fully hardware dependent language.
3	For C language, the standard compilers can be used to compile and execute the program. GCC (GNU Complier collection), Borland turbo C, Intel C++ compiler are some of the popular compilers which are used to compile, execute a C language program.	For Embedded C, you need to some specific compilers that are able to generate particular hardware/micro-controller based output. Keil compiler (An Arm company compilers), BiPOM ELECTRONIC – Embedded training and Development, Green Hill software etc are some of the popular compilers to compile, run an Embedded C language program.
4	In the C programming language, we can use standard function like printf(), scanf() etc for output and input.	These functions may not work, because in an embedded device there may not any standard output device (like monitor, Keyboard etc.). you have to write code to display output to connected display unit like 16X2 LCD, graphics display etc.

5	C language compilers generate operating system dependent executable files that can be run on the same operating system.	Embedded C language compilers generate hardware dependent files that you have to upload in the micro-controller and then you have to switch on the device to check weather code is working or not
6	Readability modifications, bug fixing are very easy in a C language program.	It's not too easy to read, understand, modify and fix the bugs in an Embedded C language program.

Basic Structure of an Embedded C Program (Template for Embedded C Program)

The following part shows the basic structure of an Embedded C Program.

- Multiline Comments . . . . . Denoted using /*……*/
- Single Line Comments . . . . . Denoted using //
- Preprocessor Directives . . . . . #include<…> or #define
- Global Variables . . . . . Accessible anywhere in the program
- Function Declarations . . . . . Declaring Function
- Main Function . . . . . Main Function, execution begins here
  {
  Local Variables . . . . . Variables confined to main function
  Function Calls . . . . . Calling other Functions
  Infinite Loop . . . . . Like while(1) or for(;;)
  Statements . . . . .
  ….
  ….
  }
- Function Definitions . . . . . Defining the Functions
  {
  Local Variables . . . . . Local Variables confined to this Function
  Statements . . . . .
  ….
  ….
  }

Different Components of an Embedded C Program

Comments: Comments are readable text that are written to help us (the reader) understand the code easily. They are ignored by the compiler and do not take up any memory in the final code (after compilation).

There are two ways you can write comments: one is the single line comments denoted by // and the other is multiline comments denoted by /*….*/.

Preprocessor Directive: A Preprocessor Directive in Embedded C is an indication to the compiler that it must look in to this file for symbols that are not defined in the program.

In C Programming Language (also in Embedded C), Preprocessor Directives are usually represented using # symbol like #include… or #define….

In Embedded C Programming, we usually use the preprocessor directive to indicate a header file specific to the microcontroller, which contains all the SFRs and the bits in those SFRs.

In case of 8051, Keil Compiler has the file "reg51.h", which must be written at the beginning of every Embedded C Program.

Global Variables: Global Variables, as the name suggests, are Global to the program i.e., they can be accessed anywhere in the program.

Local Variables: Local Variables, in contrast to Global Variables, are confined to their respective function.

Main Function: Every C or Embedded C Program has one main function, from where the execution of the program begins.

Comparisons of assembly and 8051-C programming

Advantages of ALP

It is useful to generate compact code for the 8051 microcontroller. If program is in assembly language user can control the exact instruction and their sequences.

Disadvantages of ALP

1. It is tedious and time consuming.
2. It has no function library facility.
3. The program updating and modification is complicated in ALP
4. The code generated in ALP is not easily portable to another microcontroller.

Advantages of C

1. It is easier and less time consuming then ALP
2. It is easy to modify and update.
3. User has facility to use a function library
4. C language generated code is transferable for one to other microcontrollers.

Disadvantages of C

It produce hex file which size is greater the ALP.

Super Loop

It is a just an endless loop. It is used for system continually functioning within the loop. The for(;;) and while(1) are the endless/ super loop.

Data types in C

The C compiler provides all data types of standard C data type. It also provides some additional data types which are related to the 8051 microcontroller. Data Types in C Programming Language help us declaring variables in the program.

The following table shows widely used data types in 8051-C

Sr. No.	Data types	Size in bits	Data Range
1.	unsigned char	8-bit	0 to 255
2.	(signed) char	8-bit	-128 to +127
3.	unsigned int/short	16-bit	0 to 65535
4.	(signed) int/short	16-bit	-32768 to +32767
5.	sbit	1-bit	SFR bit addressable only
6.	Bit	1-bit	RAM bit address area only
7.	Sfr	8-bit	SFR address 80H to FFH i.e RAM location whose address from 80H to FFH

Table 1 Data Types

1 unsigned char:

i. It is one of the most widely used data type for the 8051 microcontroller.
ii. It is an 8 bit (1 Byte)data type
iii. It takes a value range 0-255(00H-FFH)
iv. It is generally used for setting any counter value in which there is no need of singed char.

Assignment 1: Write an 8051-C program to send values 00H-FFH to P1.

```
#include<REGX51.H>
//header file for the generic 8051 microcontroller. It is useful for define all SFR, SBIT etc of the controller. It is different for different microcontroller.
void main(void)  // main function without return value
        {       while(1)
                        {       unsigned char z;
                                for(z=0; z<=255; z++)
                                P1=z;
                        }
        }
```

Assignment: 2 Write an 8051 C program to send hex values of ASCII character UNISHIVAJI to P1.

```
#include<REGX51.H>
void main(void)
        {       unsigned  char mydata[]="UNISHIVAJI";
                unsigned  char z;
                while(1)
                        {
                                for(z=0; z<10; z++)
                                P1=mydata[z];
                        }
        }
```

2 Signed char:

It is an 8-bit signed data type.

i. MSB (i.e. D7 bit) represent sign positive or negative.
ii. The range of signed character is -128 to +127.
iii. If keyword unsigned not used the default value is considered signed.

Assignment 3: Write an 8051 C program to send values of temperature range -4 to +4 to P1

```
#include<REGX51.H>
void main(void)
        {
                char mydata[]={-4,-3,-2,-1,1,2,3,4};  // data types default is signed
                unsigned char z;
                while(1)
                        {
                                for(z=0;z<8;z++)
                                P1=mydata[z];
                        }
        }
```

3 unsigned int:

i. It is a 16-bit data(2 byte) type.
ii. It takes a value in the range of 0-65535(i.e. 0000H to FFFFH)
iii. Unsigned integer are used to define variable such as memory address, to set counter value more than 255

Assignment 4: Write an 8051 C program to toggle P1 50,000 times.

```
#include<RGEX51.H>
void main(void)
        {       unsigned int  z;
                for(z=0; z<50000; z++)
                        {
                                P1=0x55;
                                P1=0xAA;
                        }
        }
```

4 signed int:

i. It is an 16-bit signed data types
ii. It takes a value in the range of -32768 to +32767
iii. MSB (i.e. D15) represent sign positive or negative.
iv. If keyword unsigned not used the default value is considered signed

5 sbit:

i. The sbit keyword widely used in 8051-C data types
ii. It used to access single bit addressable SFR (i.e. bit addressable register such as A,B, P0, etc)
iii. This data type is declared before writing void main function

Assignment 5: Write an 8051 C program to toggle bit D0 of the port P1 (P1.0) 50000 times

```
#include<RGEX51.H>
sbit MYBIT = P1^0;

void main(void)
        {       unsigned int  z;
                for(z=0; z<50000; z++)
                        {
                                MYBIT =0;
                                MYBIT =0;
                        }
        }
```

6 bit:

The bit is keyword allows accessing a single bit of bit addressable area from 20H to 2FH or declaring single bit variable.
In 8051 the sbit is a data type used for only bit address sfr only. Sometimes we need to store some data in bit address section of data RAM space 20H -2FH then we used bit data type.

Assignment 6: write an 8051-C program to get status of bit P1.0, save it, and send it to P2.7 continuously, use bit keyword for save data.

```
#include<REGX51.H>
sbit inbit =P1^0;
sbit outbit=P2^7;
bit membit ;            //memory bit of bit addressable area
void main(void)
        {
                inbit=1;   //making input
                while(1)
                        {
                                membit=inbit;
                                outbit=membit;
                        }
        }
```

7 sfr:

i. It used to access SFRs with their addresses

Assignment 7: Write an 8051 C program to toggle the bits of port P0 & P1 continuously with a 250 ms delay. Use the sfr data type to declare port addresses.

```
//accessing ports as SFRs using sfr data type
sfr P0=0x80;
sfr P1=0x90;
void MSdelay(unsigned int);
void main(void)
{
while(1)
{
P0=0x55;
P1=0x55;
MSdelay(250);
P0=0xAA;
P1=0xAA;
MSdelay(250);
}
}
void MSdelay(unsigned int itime)
{
unsigned int i,j;
for (i=0;i<=itime; i++)
for (j=0;j<=1275; j++);
}
```

Bit addressable I/O Programming in 8051-C

In 8051 I/O ports P0-P3 are byte as well as bit addressable. We can access a single bit without worrying rest of the bits of port. Sbit is a data type used to access a single bit of P0-P3. One way to do that is to use Px^y form, where x= port no(0,1,2,3) and y=port pin no.(0,1,2,3,4,5,6,7). e.g. P1^7 port 1 pin no. 7.

Assignment 8: Write an 8051-C to toggle only bit P2.4 continuously without disturbing other remaining bits of P2.

```
#include<REGX51.H>
Sbit mybit=P1^4;
void main(void)
        {
                for(;;)
                        {
                                mybit=0;
                                mybit=1;
                        }

        }
```

Assignment 9: Write an 8051-C to monitor bit P1.5, if it is high send 55H to P0 otherwise send AAH to P2.

```
#include<REGX51.H>
sbitmybit=P1^5;
void main(void)
{
        mybit=1; //making mybit as an input
        for(;;)
        {
        if(mybit==1)
        P0=0x55;
        else
        P2=0x55;
        }
}
```

Assignment 10: Write an 8051-C program to turn bit P1.5 on and off 50,000 times using sbit keyword.

```
sbit mybit=0x95;   //accessing single bit using sbit keyword with address
void main(void)
    {
        unsignedint z;
        for(z=0;z<50000;z++)
            {
                mybit=1;
                mybit=0;
            }
    }
```

Accessing SFR addresses 80H to FFH

Another way to access SFR RAM space 80H to FFH is to use the ***sfr*** data type. In that case there is no required #include <REGX51.H> statement. This is method widely used for new generation of 8051 microcontroller.

Assignment 11: Write an 8051-C program to send 55H to P0,P1 and P2 usingsfr keyword to declare the port address.

```
sfr P0=0x80 ;      //accessing SFR using sfr keyword
sfr P1=0x90 ;
sfr P2=0xA0 ;
void main(void)
        {
            while(1)
                {
                    P0=0x55;
                    P1=0x55;
                    P2=0x55;
                }
        }
```

Time delay generation in 8051-C

There are two ways to create a time delay in 8051-C

1. Using Simple ***for*** Loop
2. Using 8051 timer (T0 ad T1)

Using Simple *for* loop

In 8051-C, the time delay can be generated by using simple ***for*** loop.

Assignment 12: Write an 8051-C program to generate square wave at P1 with some frequency.

```
#include<REGX51.H>
void main(void)
        {
                unsigned char m;
                while(1)
                        {
                                P1=0x00;
                                for(m=0;m<100;m++);
                                P1=0x00;
                                for(m=0;m<100;m++);
                        }
        }
```

Assignment 13: Write an 8051-C program to toggle bits of P1 continuously forever with 250mS delay.

```
#include<REGX51.H>
voidmsdelay(unsigned int);
void main(void)
        {
                while(1)
                        {
                                P1=0x55;
                                msdelay(250);
                                P1=0xAA;
                                msdelay(250);
                        }
        }
voidmsdelay(unsigned int z)
// this is an standard program for delay for zvalues in millisecond. If z=250
then //generated is 250mS
        {
                unsignedinti,j;
                for(i=0;i<z;i++)
                for(j=0;j<1275;j++);
        }
```

Time delay generation using Timer

Using internal two timing register T0 and T1 user can generate time delay in 8051 microcontroller.

Steps of timer 8051-C programming in mode 1

i) Load TMOD register to select mode 1
ii) Load TLx and THx register with initial count.
iii) Start the timer TRx=1.
iv) Keep monitoring of timer flag by using while(TFx==0). It get out from loop when TFx becomes 1
v) Stop the timer by using TRx=0 instruction
vi) For next delay clear the TFx flag by TFx=0 instruction
vii) For next delay go to the step ii

Steps for delay calculation in mode 1

i) Divide required time delay with 1.085µS for 11.0592MHz OR 1µS for 12MHz
ii) Then result of step i subtract from 65536 (i.e.65536-x)
iii) Convert the result of step ii into hex form. If YYXX is hex form then YY is loaded into THx register and XX loaded into TLx register.

Assignment 14: Write an 8051-C to generate a square wave on port P1 using timer delay

```
#include<REGX51.H>
mybit=P1^5;
void delay(void)
void main(main)
    {
        while(1)
            {
              P1=0xFF;
              delay();
              P1=0x00;
              dealy();
              }
        }
void delay()
        {
        TMOD=0X01;
        TH0=0X03;
        TL0=0XFF;
        TR0=1;
        while(TF0==0);
        TR0=0;
        TF0=0;
        }
```

Assignment 15: Write an 8051-C program to generate delay of 5mS at port pin P1.5 using timer delay in mode 1. Assume f=11.0592MHz.

Delay 5mS

i) x=5mS/1.085 µS=4608
ii) 65536-x=60928_{10}
iii) YYXX=EE00H i.e. TH0=EEH,TL0=00H

60928÷16	0(0) Remainder
3808÷16	0(0)
238÷16	14(E)
14÷16	14(E)

```
#include<REGX51.H>
mybit=P1^5;
void delay(void);
void main(void)
{
while(1)
{
mybit=1;
delay();
mybit=0;
dealy();
}
}
void delay()
        {
        TMOD=0X01;
        TH0=0XEE;
        TL0=0x00;
        TR0=1;
        While(TF0==0);
        TR0=0;
        TF0=0;
        }
```

Factors affects on accuracy of time delay:

There are three factors which affect on accuracy of the time delay which are given below

a. Machine Cycle:-

Time delay depends on clock period and it varies with machine cycle. For different types of microcontrollers has different machine cycle and clock period. For 8051/52 microcontroller design uses 12 clock period per machine cycle. For new version microcontroller has few clock period per machine cycle. For DS5000 4 clock period per machine cycle and DS89C40 uses 1 clock period per machine cycle.

b. Crystal:-

Time delay also depends on crystal frequency connected at X1 and X2 pins of microcontroller, because duration of the clock period for machine cycle is a function of crystal frequency.

c. Compiler:-

The 3rd factor that affect the time delay is the compiler used to compile C program. When program in assembly language we can control the exact instruction and their sequences used in time delay subroutine. In case of C compiler that converts C statement into assembly language.

Byte addressable I/O Programming in 8051-C:

In 8051 microcontroller there are four ports P0,P1, P2 and P3. All ports are bit as well as byte addressable. In this we focused on byte addressable I/O programming.

Assignment 16: LED's are connected to the ports P1 and P2. Write an 8051-C program that shows cont from 00H to FFh on the LED

```
#include<REGX51.H>
void main(void)
        {       unsigned char z;
                for(;;)
                        {       P1=z;
                                P2=z;
                                z=z+1;
                        }
        }
```

Assignment 17: Write an 8051-C program that read data from P1 wait ½ second(i.e. 500ms) than send it P2,

```
#include<REGX51.H>
voidmsdelay(unsigned int);
void main(void)
{unsigned char z;
P1=0xFF;    //making P1 as an Input
for(;;)
{
z=P1;  //read data from P1
msdelay(500);
P2=z;// send data to P2
}
}
```

```
void msdelay(unsigned int z)  // this is an standard program for delay
for z //values in millisecond. If z=500 then it generates 500mS delay
{
        unsignedinti,j;
        for(i=0;i<z;i++)
        for(j=0;j<1275;j++);
}
```

Logical Operation in 8051-C

Logical operations are also known as bitwise operation in 8051-C. There are six important logical operation in C, logical AND, OR, XOR,NOT,RIGHT SHIFT and LEFT SHIFT. Following table shows logical operation with its symbol.

Sr. No.	Logic gate	Symbol	Logical Operation
1	AND	&	Logical AND operation
2	OR	|	Logical OR operation
3	XOR	^	Logical XOR operation
4	NOT	~	Logical NOT operation
5	Right Shift	>>	Logical right shifting number of times given in statement
6	Left Shift	<<	Logical left shifting number of times given in statement

Table 2 Logical Operation

Assignment 18: Write an 8051-C program to generate square wave at P0 and P1 with 250mS ON and OFF time using logical NOT operation.

```
#include<REGX51.H>
void msdelay(unsigned int);
void main(void)
        {
                P0=0xFF;
                P1=0xFF;
                while(1)
                        {       P0=~P0;
                                P1=~P1;
                                msdelay(250);
                        }
        }
void msdelay(unsigned int z)
        {
        unsignedinti,j;
        for(i=0;i<z;i++)
        for(j=0;j<1275;j++);
        }
```

Assignment 19: Write an 8051-C program to toggle all bits of P0 and P1 continuously forever with 250mS delay using logical XOR operation.

```
#include<REGX51.H>
Void msdelay(unsigned int);
void main(void)
{
P0=0x55;
P1=0x55;
while(1)
{
P0=P0^0xFF; //XOR gate change the o/p for unequal I/P
P1=P1^0xFF;
msdelay(250);
}
}
void msdelay(unsigned int z)
{
        Unsigned int i, j;
        for(i=0;i<z;i++)
        for(j=0;j<1275;j++);
}
```

Assignment 20: Write an 8051-C program to perform logical NOT operation of the bit P1.0 and send result to P2.7.

```
#include<REGX51.H>
sbit mybit1=P1^0;
sbit mybit2=P2^7;
bit mybit3=0x20;//mybit3 defined at bit addressable memory location 0x20
void main(void)
{
mybit1=1; //making P1.0 as an input
for(;;)
{
mybit3=mybit1;
mybit2=~mybit3;
}
}
```

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.

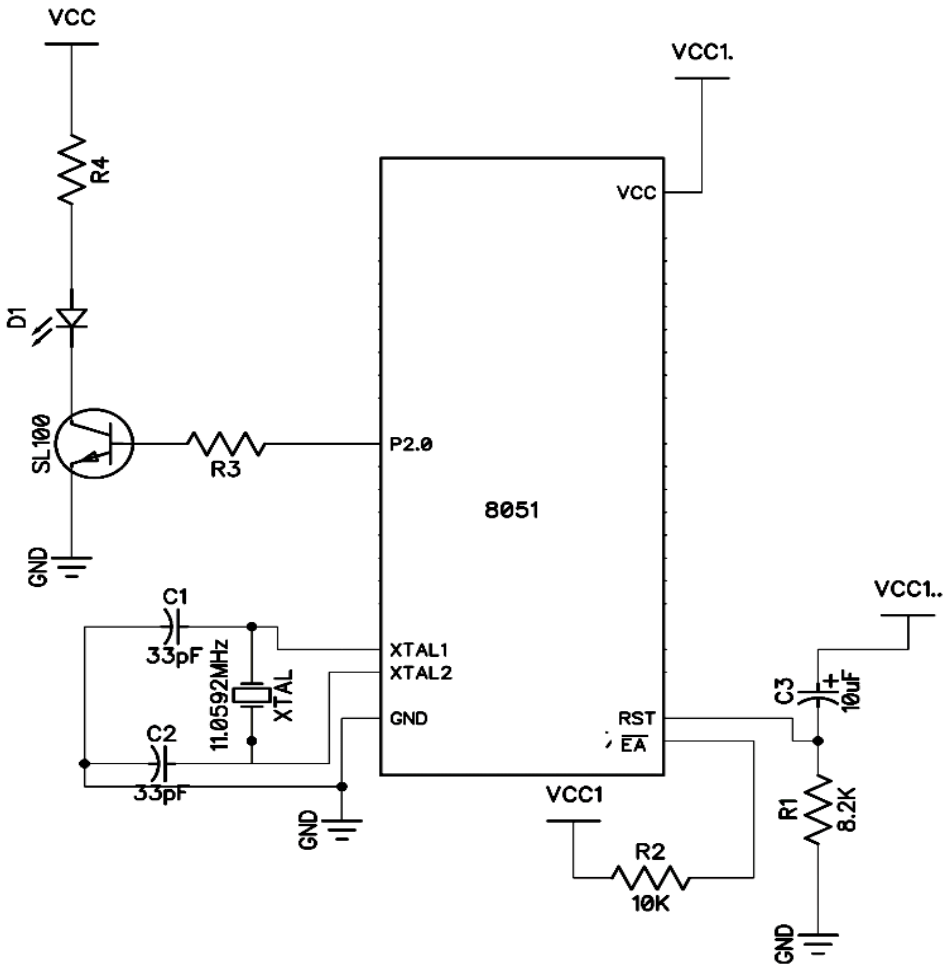


Fig.1 INTERFACING OF LED WITH 8051

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

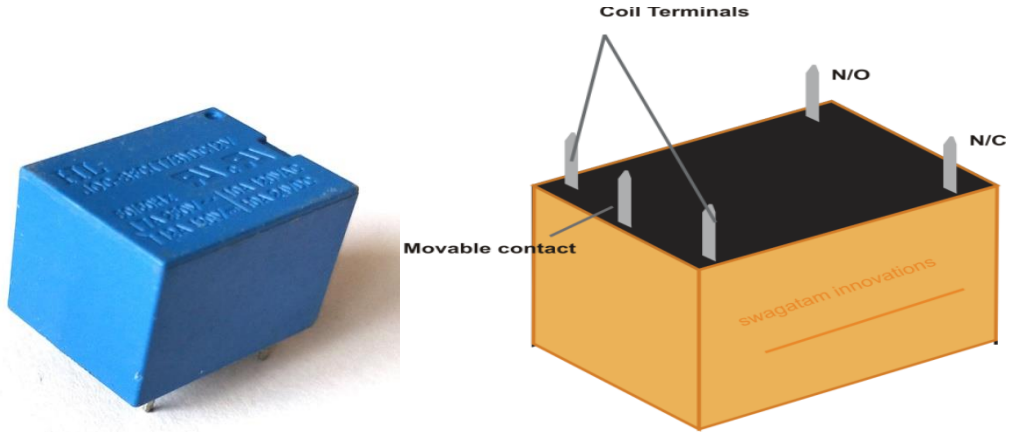
IC: Collector current(here LED Current=10mA)

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



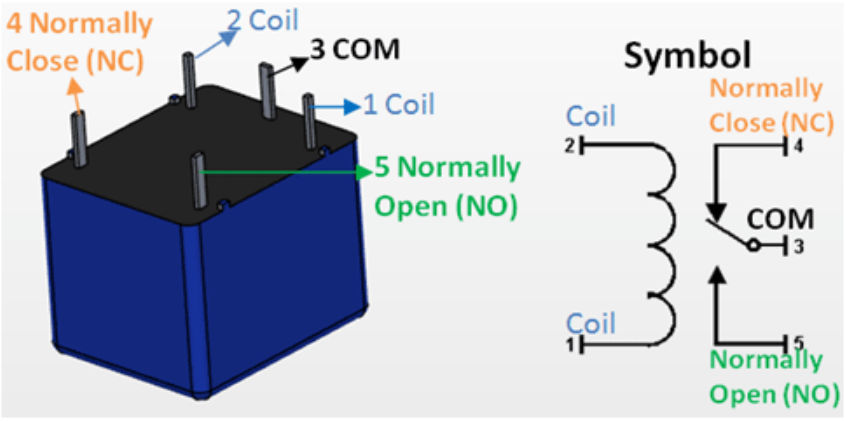


Fig.2: Relay terminals

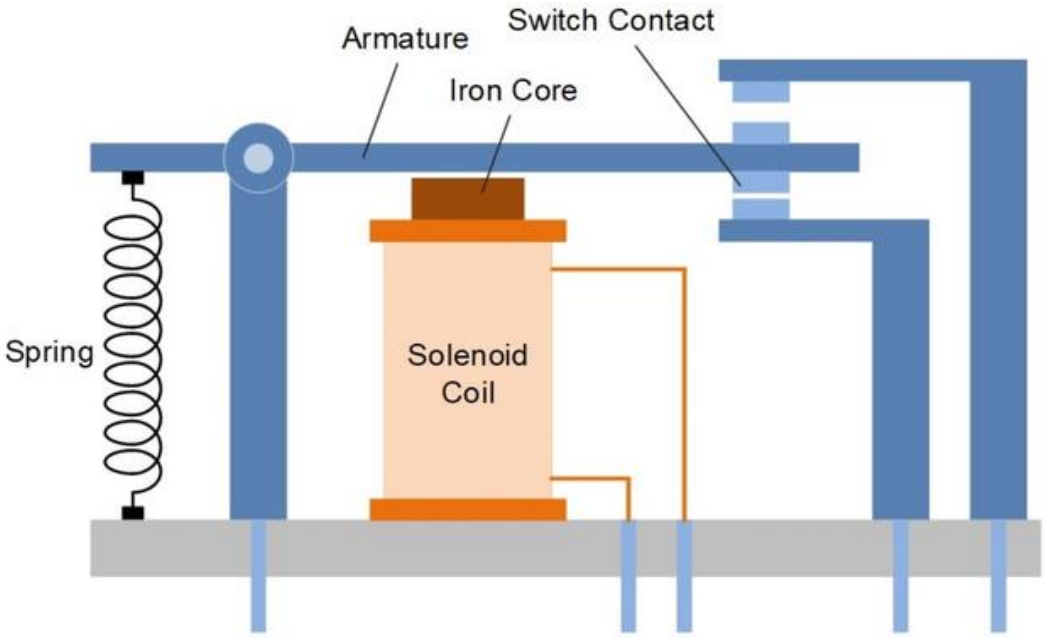


Fig.3: schematic of Relay

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.

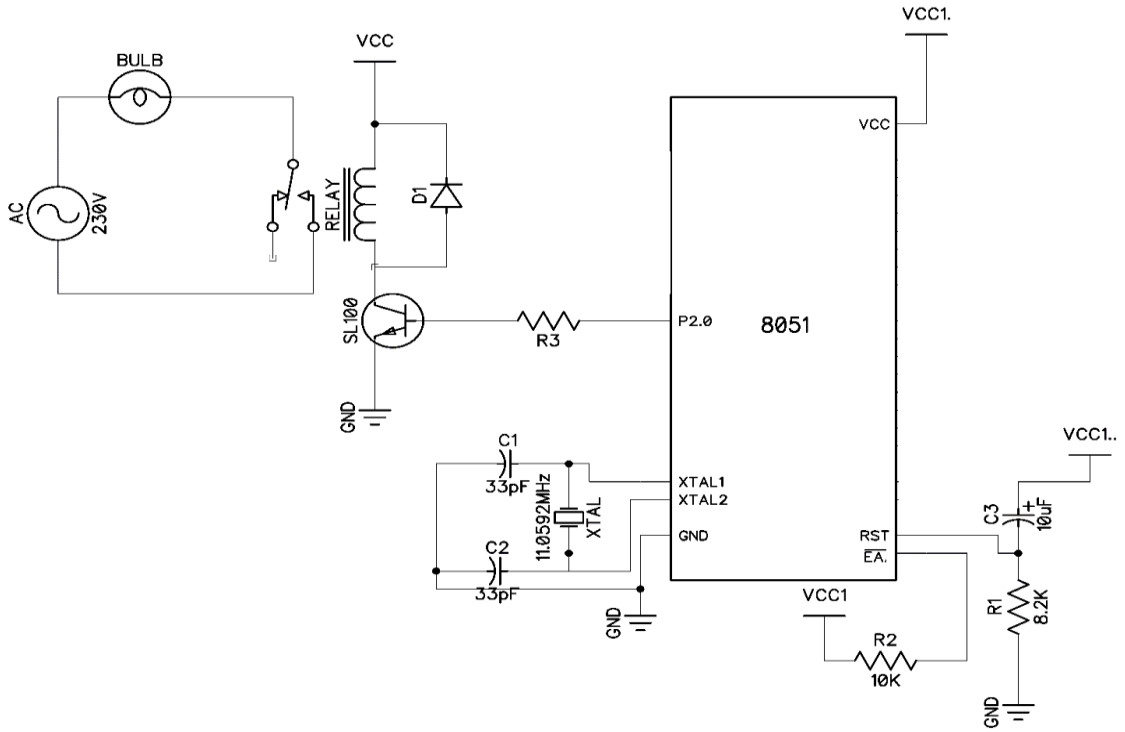
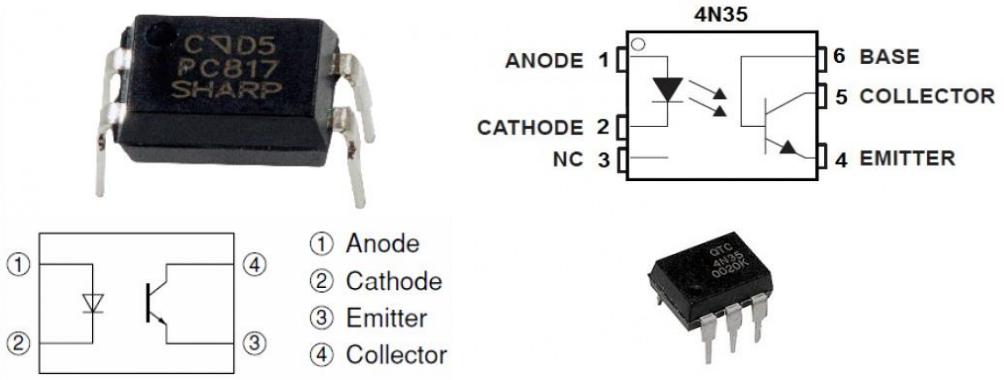


Fig.4 INTERFACING OF RELAY WITH 8051

```
// 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.

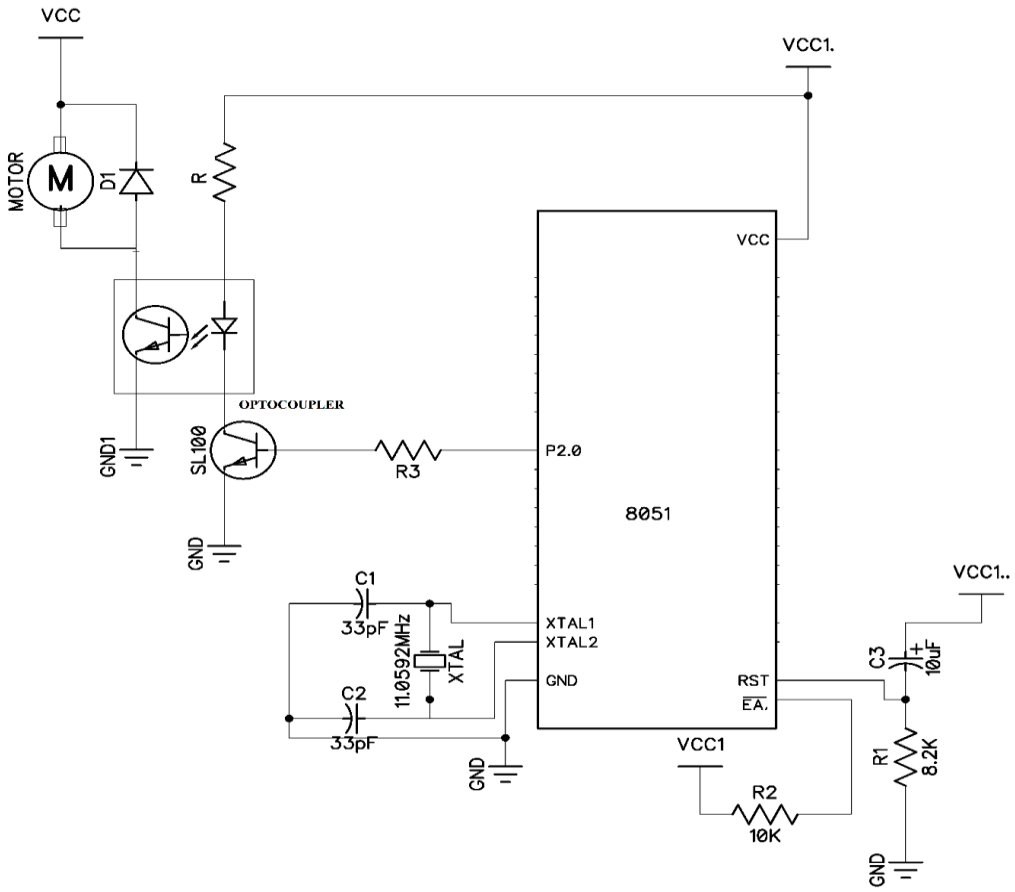


Fig.6 INTERFACING OF OPTOCOUPLER WITH 8051

```
// INTERFACING OF OPTPCOUPLER WITH 8051
#include <REGX51.H>
sbit optocoupler=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 switch with 8051:

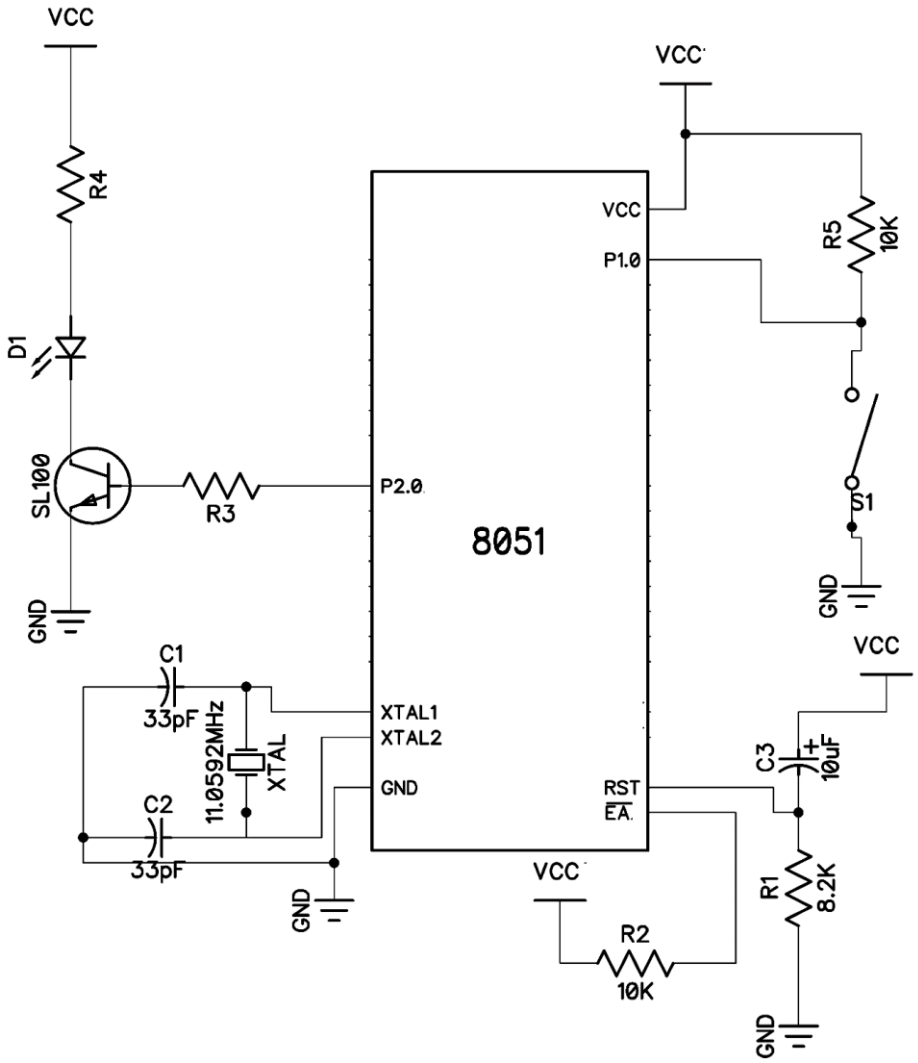


Fig.7 INTERFACING OF SWITCH AND LED WITH 8051

We can connect a number of switches at ports of the 8051. A switch is connected at the port pin P1.0 and we read status of switch is displayed on LED connected at P2.0

Circuit Description:

A Switch is connected at port pin P1.0 and it is normally open. One terminal of the switch is connected to +Vcc through pull up resister and one terminal is grounded. When switch is pressed port pin becomes low and when switch is open the port pin becomes high.

The statues of the switch is read by using MOV A,P1 and it display on LED. In this case port P1 must configured as an input port.LED is connected at pin P2.0 through buffer transistor. The data from the accumulator is transferred to port 2 by giving instruction MOV P2,A after complementing.

```
// Interfacing of SWITCH and LED WITH 8051
#include <REGX51.H>
sbit SWITCH=P1^0;
sbit LED=P2^0;
void main(void)
{
SWITCH=1;
while(1)
{
if (SWITCH==0)
{
LED=0;
}
else
{
LED=1;
}
}
}
```

5. Interfacing of DC motor with 8051

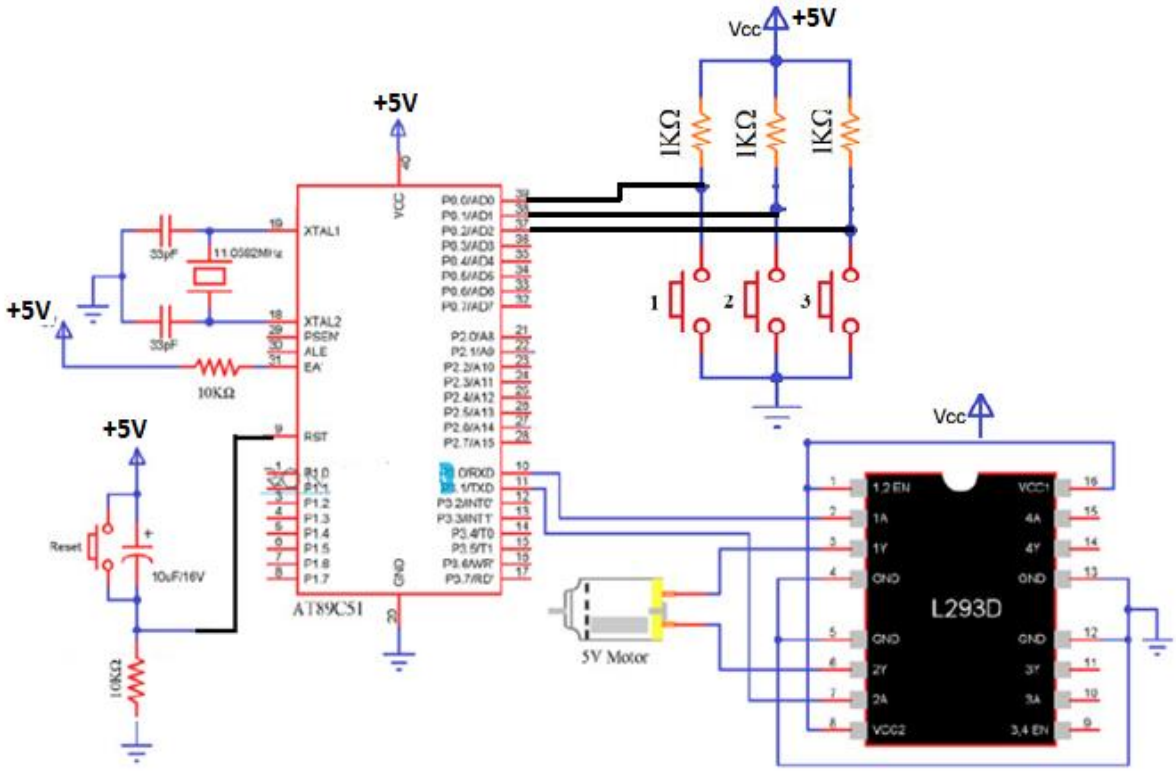


Fig.8: DC motor interfacing with 8051

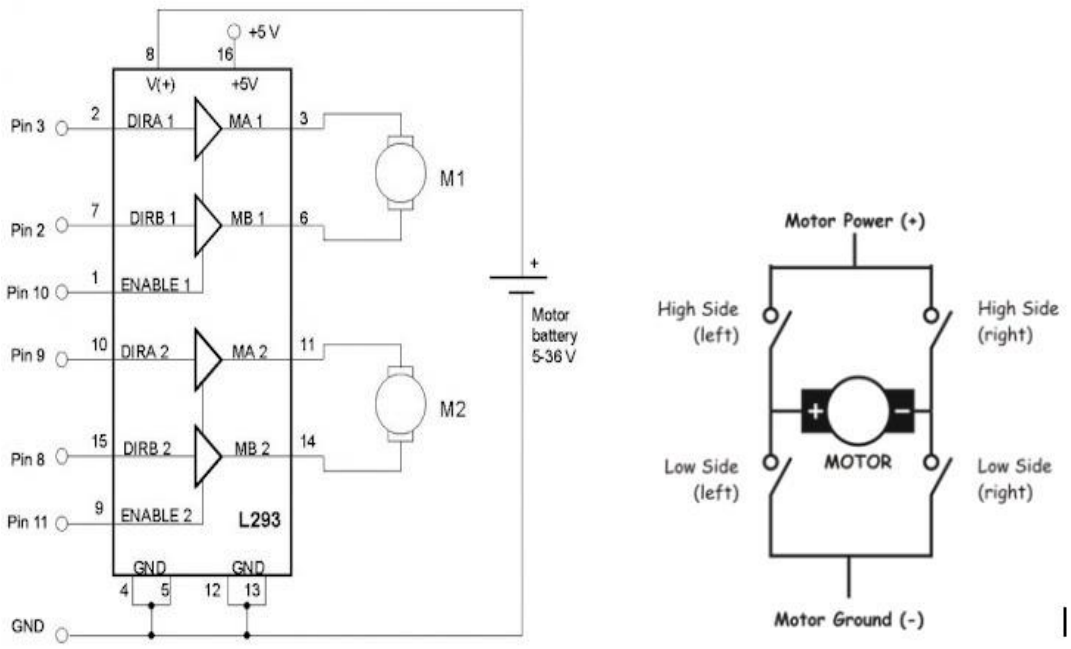


Fig.9: IC L293

```
// 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);
        }
        }
}
```

6. 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. 10: 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.

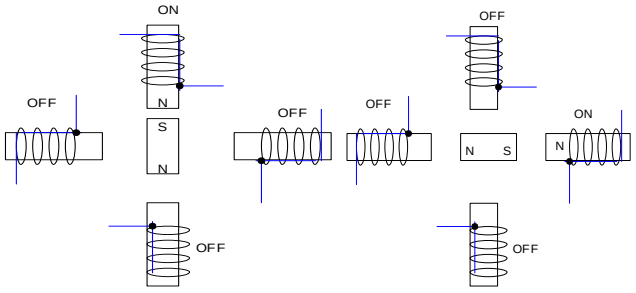


Fig.11: Full Stepping Method

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.

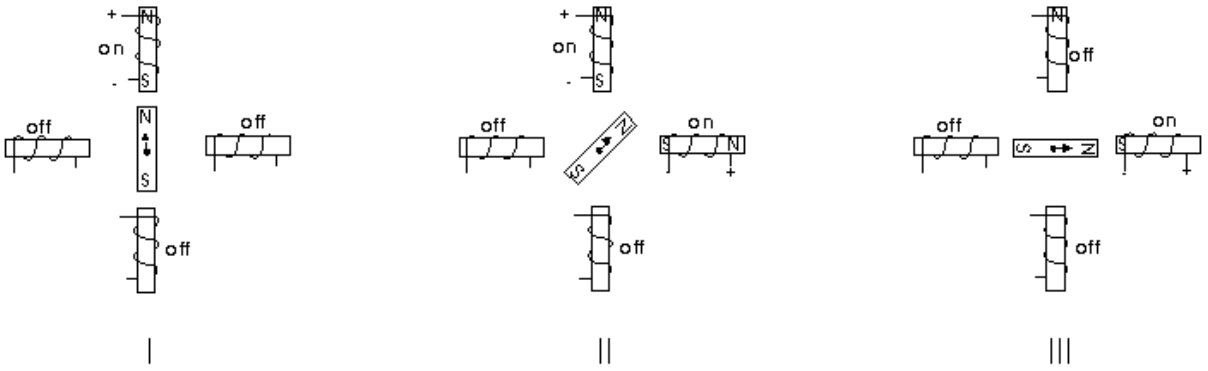


Fig.12: Half stepping

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 3: Stepper Motor step sequence

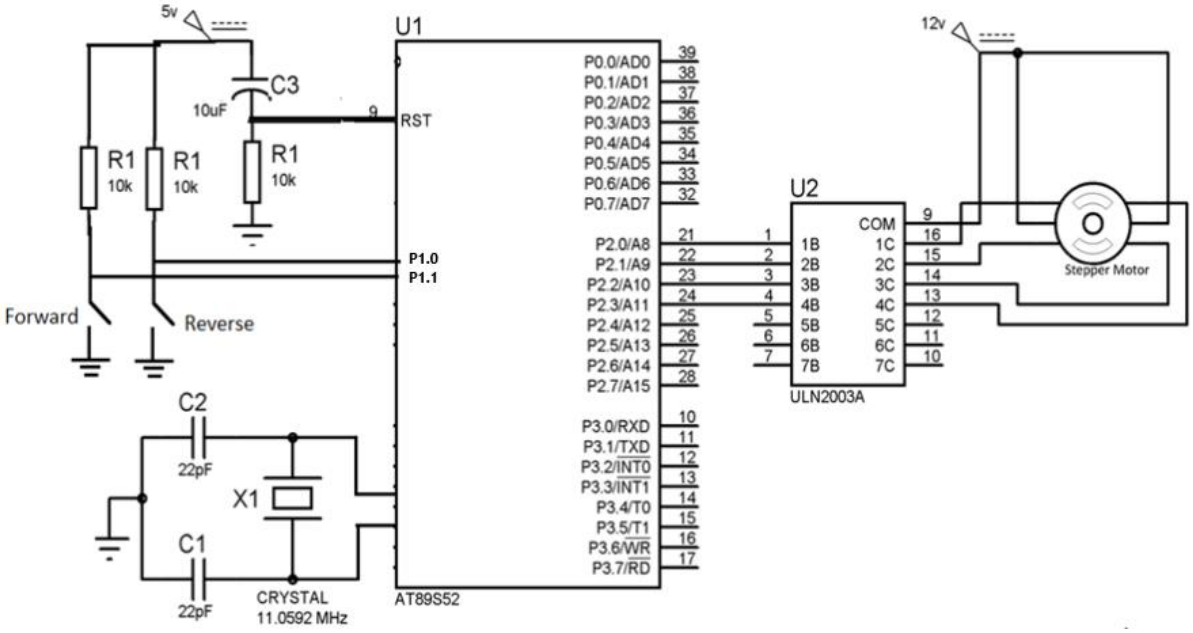


Fig.13: Stepper motor interfacing with 8051

*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.

```
#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);
}}}
```