LECTURE NOTES (E- CONTENTS) for

B.Sc. III Electronics (2022-23)

Semester: V Paper- V DSE 1005E1

Linear Integrated Circuits, 8051 Microcontroller Interfacing and

Embedded C

Section II: 8051 Microcontroller Interfacing and Embedded C

Unit- 1: Introduction to embedded C

Prepared and Circulated

for B.Sc. III Electronics Students

BY

Dr. C. B. Patil

Associate Professor, Department of Electronics

Vivekanand College (Autonomous), Kolhapur

Unit 1
Introduction to embedded C

Syllabus: *Advantages and disadvantages of programming in 8051-C & Assembly Language. Data types, operators and loops, I/O programming, Accessing SFR addresses, Logical operation. Data conversion programs, Accessing ROM space, programming for Time delay generation (using timer), external interrupts (Level and edge triggering).*

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.

<u>Basic Structure of an Embedded C Program (Template for Embedded C Program)</u>

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.

```
#include<REGX51.H>
sbit mybit=0x95;   //accessing single bit using sbit keyword with address
void MSdelay(unsigned int);
void main(void)
                    {
                      unsigned int z;
                              for(z=0;z<50000;z++)
                                        {
                                                  mybit=0;
                                                  MSdelay(500);
                                                  mybit=1;
                                        }
                    }
void MSdelay(unsigned int itime)
{
unsigned int i,j;
for(i=0;i<=itime; i++)
for(j=0;j<=1275; j++);
}
```

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 using sfr 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=0xFF;
                                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>
void msdelay(unsigned int);
void main(void)
        {
                while(1)
                        {
                                P1=0x55;
                                msdelay(250);
                                P1=0xAA;
                                msdelay(250);
                        }
        }
```

```
void msdelay(unsigned int z)
// this is a standard program for delay for z values in millisecond. If z=250 then
//generated time is 250mS
        {
                unsigned int i,j;
                for(i=0;i<z;i++)
                for(j=0;j<1275;j++);
        }
```

<u>Time delay generation using Timer</u>

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

Data conversion programming in C

In the number times there is need to convert the Packed BCD to unpacked BCD, Packed BCD to ASCII or vice Versa. E.g for sending data BCD 99H to LCD it is needed first convert it into unpacked form and then ASCII.

Assignment 21: Write an 8051 C program to convert packed BCD 0x29 to unpacked BCD and display byte on P1 and P2

(Packed BCD 0x29 and its unpacked form is 0x02 and 0x09)

```
#include<REGX51.H>
void main(void)
        {
                unsigned char x,y;
                unsigned char mynum=0x29;
                x=mynum & 0x0F;     //x=0x09 unpacked first number
                y=mynum & 0xF0;     //y=0x20;
                y=y>>4;                 //y=0x02; unpacked second number
                P1=x;                //sending first unpacked number to P1
                P2=y;      // sending second unpacked number to P2
        }
```

Assignment 22: Write an 8051 C program to convert packed BCD 0x29 to ASCII and display byte on P1 and P2. (Packed BCD 0x29 and its ASCII form is 0x32 and 0x39)

```
#include<REGX51.H>
void main(void)
        {
                unsigned char x,y;
                unsigned char mynum=0x29;
                x=mynum & 0x0F;     //x=0x09 unpacked first number
                y=mynum & 0xF0;     //y=0x20;
                y=y>>4;                  //y=0x02 unpacked second number
                x=x|0x30;  //first unpacked number conversion into ASCII
                y=y|0x30;  //first unpacked number conversion into ASCII
                P1=x;                 //sending first ASCII number to P1
                P2=y;      // sending second ASCII number to P2
        }
```

Assignment 23: Write an 8051 C program to convert ASCII digit ‘4’ and ‘7’ to packed BCD and display result on P1.
(ASCII of ‘4’=0x3**4** and ‘7’=0x3**7** packed BCD form is 0x**47**)

```
#include<REGX51.h>
void main(void)
        {
                unsigned char bcdnum;
                unsigned char x=’4’;     //x=0x34
                unsigned char y= ‘7’;    //y=0x37
                x=x & 0x0F;       //masking lower four bit x=0x04;
                x=x<<4;           // left shifting x contains four times x=0x40
                y=y & 0x0F;       //masking lower four bit y=0x07
                bcdnum=x|y   //packed BCD number bcdnum=0x47
                P1=bcdnum;     //sending BCD  number to P1
        }
```

Assignment 24: Write an 8051 C program to convert 1111 1101(FDH) to decimal and display the digit on P0, P1, P2.

```
#include<REGX51.h>
void main(void)
        {       unsigned char x, binbyte, d1,d2,d3;
                binbyte=0xFD;  //binary hex byte
                x=binbyte/10;   //divide by 10
                d1=binbyte%10;   // d1 is remainder of binbyte/10
                d2=x%10;      //d2 is remainder of x/10
                d3=x/10;      //d3 is quiescent of x/10
                P0=d1;   //sending conversion result to ports
                P1=d2;
                P3=d3;
            }
```

Interrupt Programming in 8051-C

Interrupt is an external or internal event that interrupts the microcontroller to inform that a device needs its service. In 8051, there are 5 interrupts. Following table shows their names and vectored memory locations.

Interrupt Names	ROM location	Pin
External hardware (INT0)	0003 H	P3.2 (12)
Timer 0 (TF0)	000B H	
External hardware (INT1)	0013 H	P3.2 (13)
Timer 1 (TF1)	001B H	
Serial Communication (RI and TI)	0023 H	

Table 3 Interrupts in C

Programming external Hardware interrupt:

There are two external hardware interrupt in 8051, INT 0 and INT 1. They are located on pins P3.2 and P3.3 of port 3 respectively. The interrupt vector location for INT 0 is 0003H and INT 1 is 0013H. These interrupts are enabled by using register IE. The INT 0 has highest priority. Using IP register, the priority can be changed. The external interrupt are activated using one of the two ways a) low level triggered interrupt b) Falling edge triggered interrupt. This types of the interrupt can be selected by using IE0, IE1, IT0 and IT1 bits in TCON register. Explanation of IE0, IE1, IT0 and IT1 is as follows

1. IE1(TCON.3):- External interrupt 1 edge flag: It is set to 1 by hardware when falling edge is detected at external pin INT1 and when interrupt is processed or ISR routine is completed this flag is cleared.

2. IT1(TCON.2):- Interrupt 1 type control bit. Set/cleared by software to specify falling edge/lowlevel triggered external interrupt. When IT1=1 interrupt is falling edge triggered interrupt. When IT1=0 interrupt is low level triggered interrupt.

3. IE0(TCON.0):- External interrupt 0 edge flag. It is set to 1 by hardware when falling edge is detected at external pin INT0 and when interrupt is processed or ISR routine is completed this flag is cleared.

4. IT0(TCON.0):- Interrupt 0 type control bit .When IT1=0 interrupt is falling edge triggered interrupt. When IT1=0 interrupt is low level triggered interrupt. Edge trigged interrupt is detected falling edge (high to low transition) and low level triggered interrupt is detected when interrupt pin is low for the four machine cycles.

Assignment 25: Write an 8051-C program for two switches are connected to pins P3.2 and P3.3. When switch is pressed the corresponding pin goes low. Write a program to a) ON all LED connected at port 0 if first switch is pressed. b) ON all LED connected at port 1 if second switch is pressed.

Answer:- LED should be ON when switch is low, so level triggered interrupt is used. To enable both external interrupt , IE is loaded with 85H .

```
Program:-
#include<REGX51.H>
void external 0(void) interrupt 0
{
P0=0xFF;    //on LED at P0
}
void external 1(void) interrupt 2
{
P1=0xFF;     //on LED at P1
}
void main(void)
{
IE=0x85;// Enables interrupt INT0 and INT1
while(1); // Super loop
}
```

Assignment 26: Write an 8051-C program for generate square wave on all the pins of ports 1 and frequency of square wave is half of the frequency applied at INT0

Answer:- Here it is required to enable the edge triggered interrupt 0 because every falling edge interrupt is enabled at this point only compliment the port 1. This interrupt enabled by loading 81H into IE.

```
#include<REGX51.H>
void external 0(void) interrupt 0
{
        P1=~P1;    // Compliment P1
```

```
 }
void main(void)
  {
      IE=0x81;       // Enables interrupt INT0
      TCON=0x01; //enabling INT0 as edge triggered
      while(1);      // Super loop
   }
```