Microprocessors and Microcontrollers (EE-231)

Lab-12

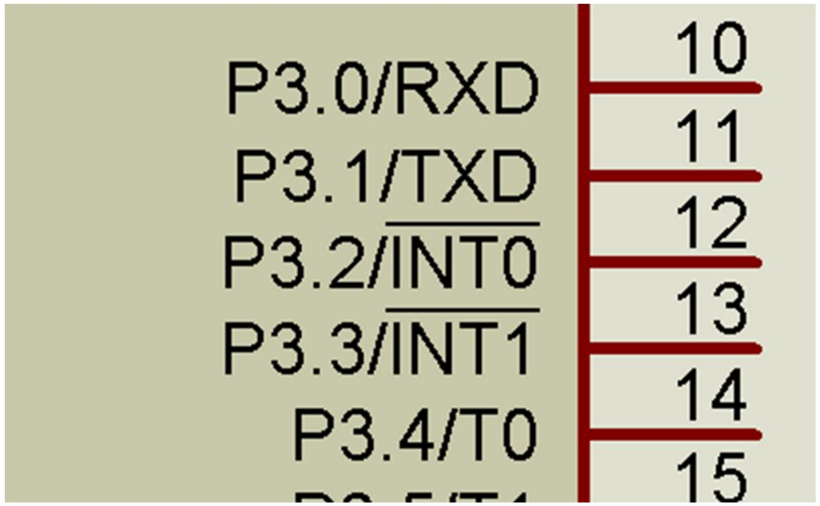
Objective

- Interrupts (External Hardware) Programming in C
  - ➢ In Proteus
  - ➢ On 8051 development board

External Hardware Interrupt

- The 8051 has two external hardware interrupts
- Pin 12 (P3.2) and pin 13 (P3.3) of the 8051, designated as INT0 and INT1, are used as external hardware interrupts
- There are two activation levels for the external hardware interrupts

1. Level trigged
2. Edge trigged

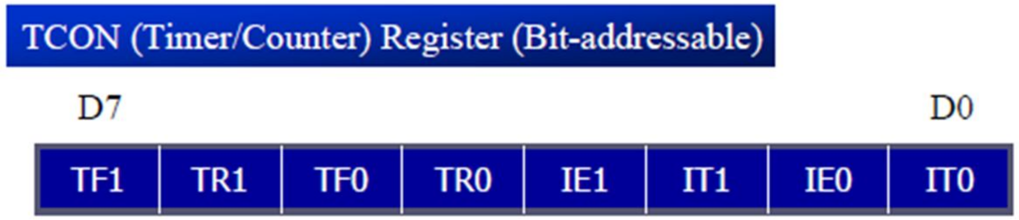


Level Triggered Interrupt

- In the level-triggered mode, INT0 and INT1 pins are normally high
- If a low-level signal is applied to them, it triggers the interrupt
- The low-level signal at the INT pin must be removed before the execution of the last instruction of the ISR, RETI; otherwise, another interrupt will be generated
- Level-triggered interrupt is the default mode upon reset of the 8051
- To ensure the activation of the hardware interrupt at the INTn pin, make sure that the duration of the low-level signal is around 4 machine cycles, but no more

Edge Triggered Interrupt

- To make INT0 and INT1 edge triggered interrupts, we must program the IT0 and IT1 flag bits of the TCON register
- IT0 and IT1 are bits D0 and D2 of the TCON register
- Remember TCON is a bit-addressable register

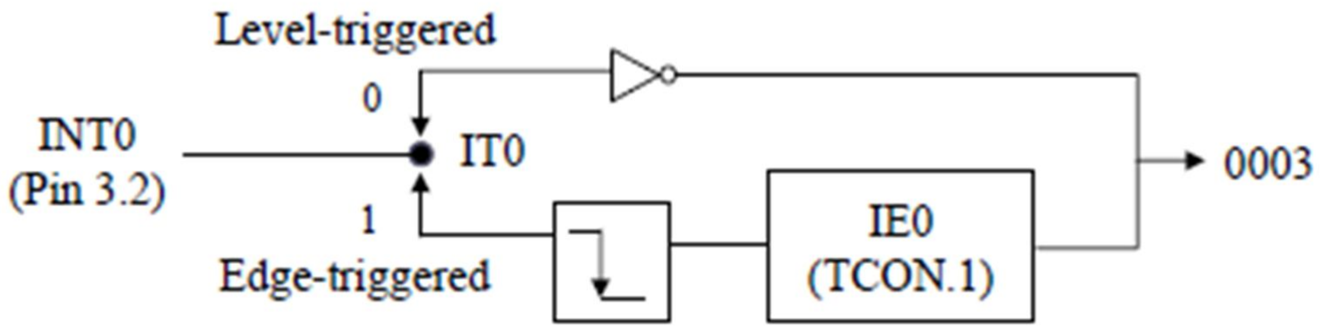


TCON (Timer/Counter) Register (Bit-addressable)

D7							D0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

- IE1 & IE0 contain the interrupt status. If it occurs then these bits are 1.
- IT0 & IT1 are interrupt type bits. When set i.e. 1, indicate edge-triggered interrupt. (negative edge triggered)
- In edge-triggered interrupts the external source must be held high for at least one machine cycle, and then held low for at least one machine cycle

External Hardware Interrupt



- To program we use following interrupt number

External Interrupt 0	(INT0)	0

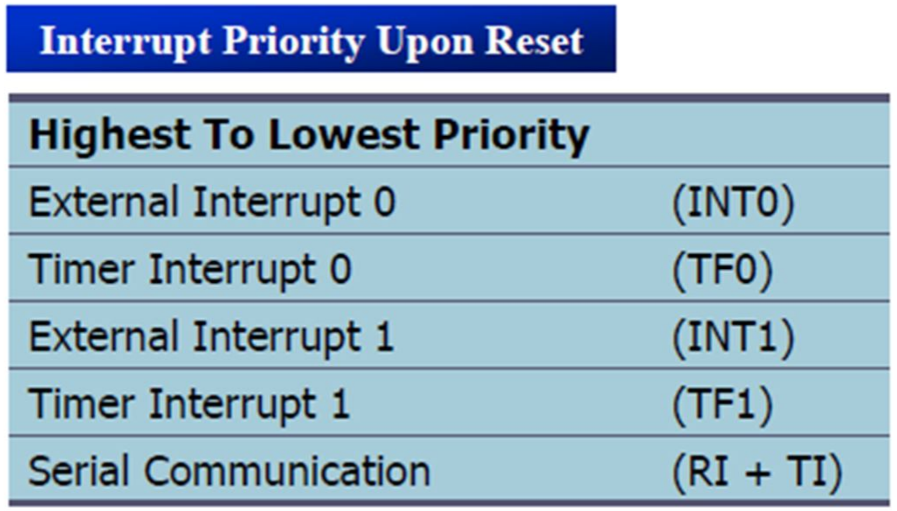
- void Ext0_Interrupt(void) interrupt 0 {

External Interrupt 1	(INT1)	2

- void Ext1_Interrupt(void) interrupt 1 {

Interrupt Priority

- When the 8051 is powered up, the priorities are assigned according to the following



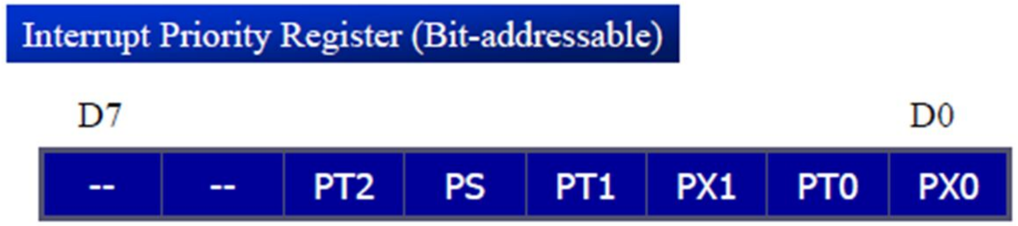
Interrupt Priority Upon Reset

Highest To Lowest Priority	
External Interrupt 0	(INT0)
Timer Interrupt 0	(TF0)
External Interrupt 1	(INT1)
Timer Interrupt 1	(TF1)
Serial Communication	(RI + TI)

- In the 8051 a low-priority interrupt can be interrupted by a higher-priority interrupt but not by another low priority interrupt

Interrupt Priority

- We can alter the sequence of interrupt priority by assigning a higher priority to any one of the interrupts by programming a register called IP (interrupt priority)
- To give a higher priority to any of the interrupts, we make the corresponding bit in the IP register high
- When two or more interrupt bits in the IP register are set to high What happens then ?



Interrupt Priority Register (Bit-addressable)

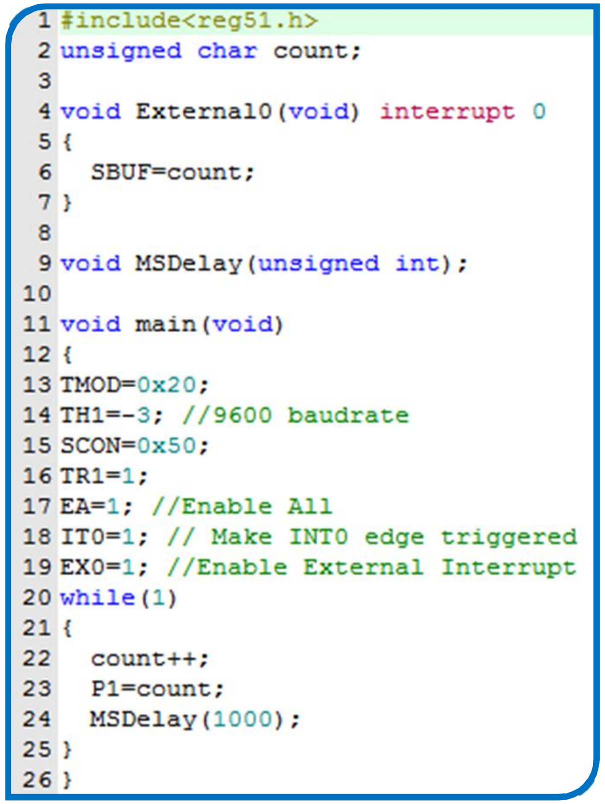
D7							D0
--	--	PT2	PS	PT1	PX1	PT0	PX0

- Bit=0 Low Priority
- Bit=1 High Priority

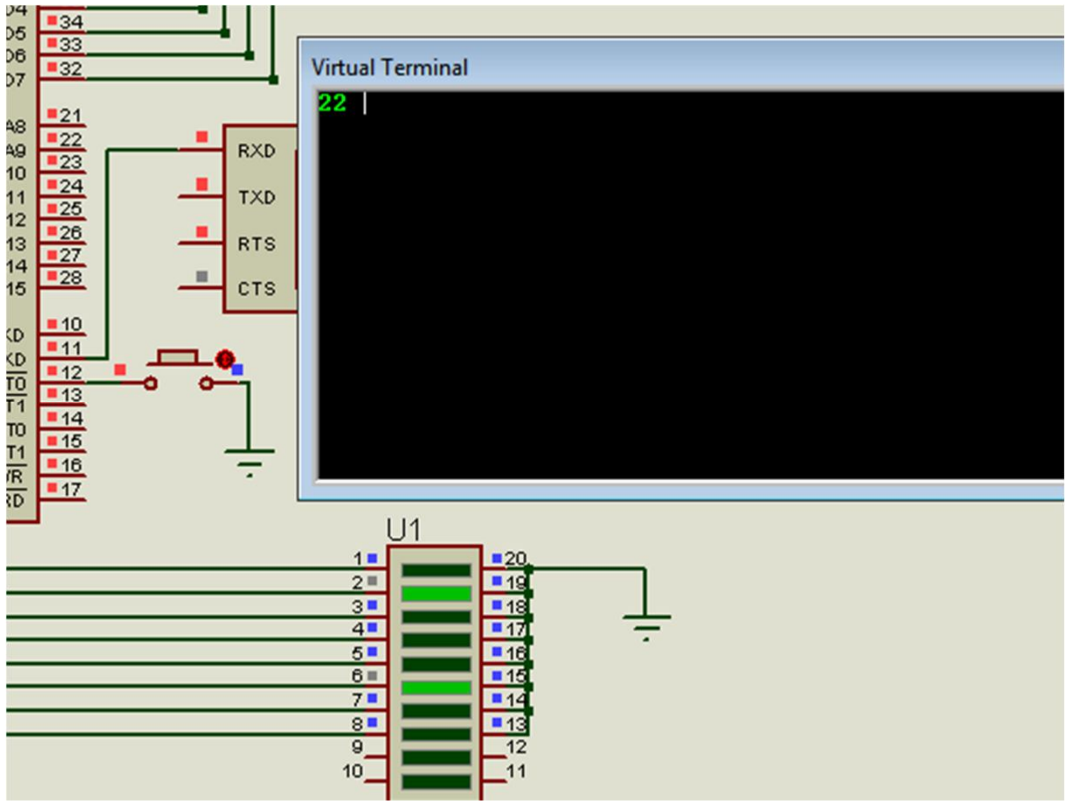
Example

Example:

Write a code in c for 8051. Make a counter and display its value on P2/P1. Whenever an edge-triggered hardware interrupt occurs, the value of count is transmitted via serial port to the PC.



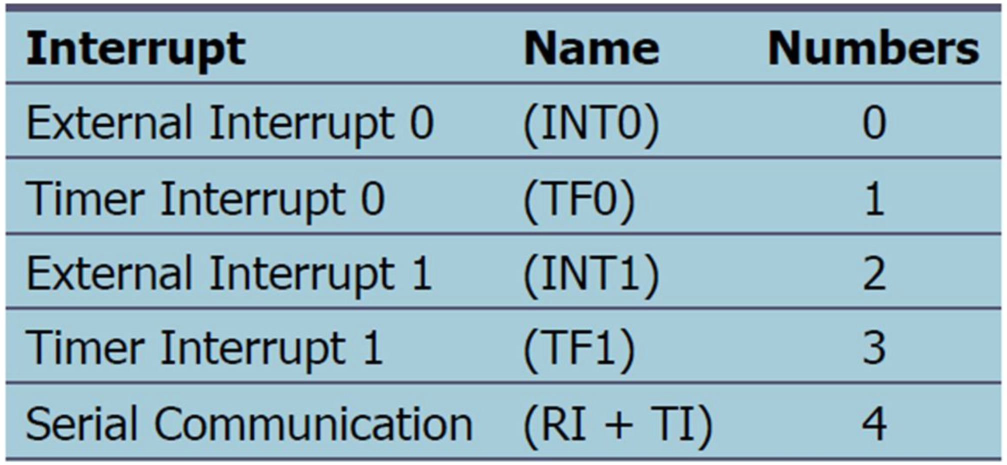
```c
#include<reg51.h>
unsigned char count;

void External0(void) interrupt 0
{
  SBUF=count;
}

void MSDelay(unsigned int);

void main(void)
{
TMOD=0x20;
TH1=-3; //9600 baudrate
SCON=0x50;
TR1=1;
EA=1; //Enable All
IT0=1; // Make INT0 edge triggered
EX0=1; //Enable External Interrupt
while(1)
{
  count++;
  P1=count;
  MSDelay(1000);
}
}
```



```
22
```

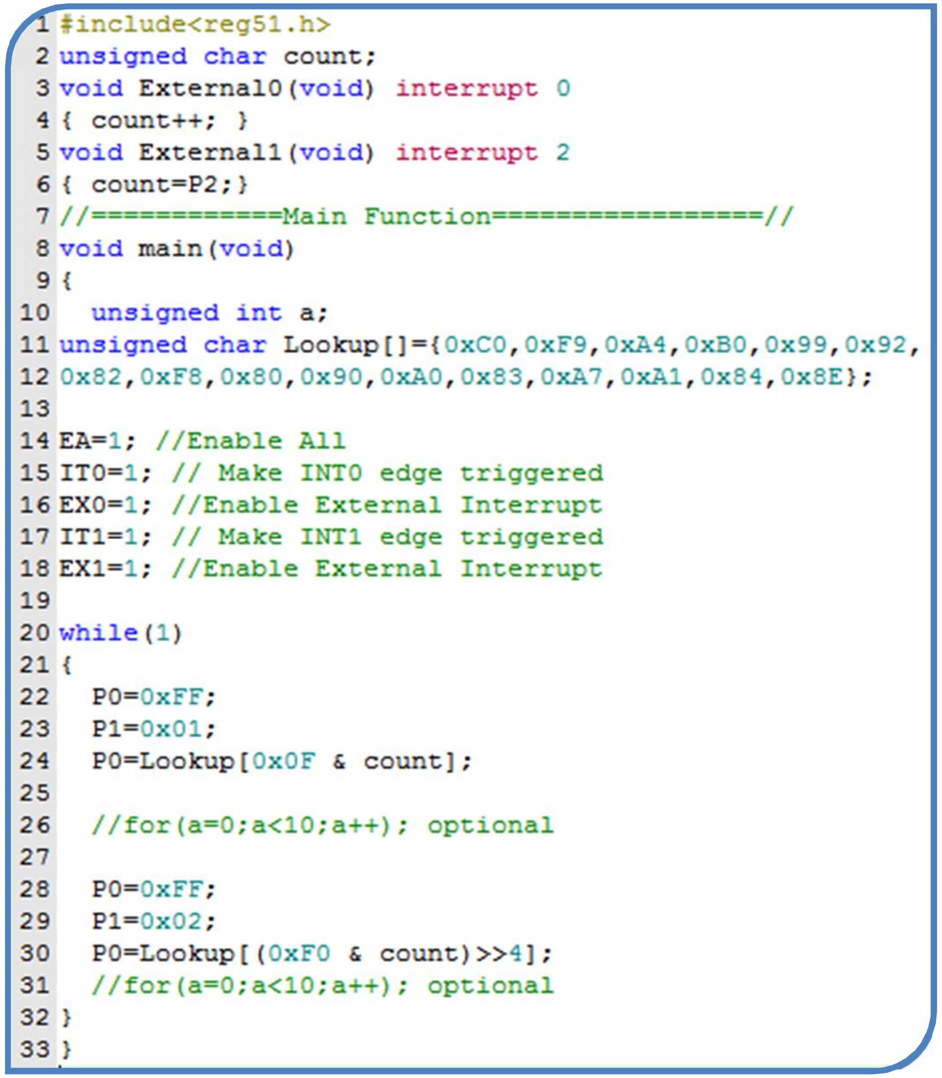
Todays Task 1

- Implement this on easy 8051 Kit.
- Make a counter from 00-FF and display its value on Seven Segment. The counter counts up when External Interrupt 0 is given.
- A DIPSW is connected to P2. Whenever the External Interrupt 1 is given, the value of DIPSW i.e. at P2 will be copied as a new value of count. And count will continue from this value.
- Make the interrupt edge triggered.



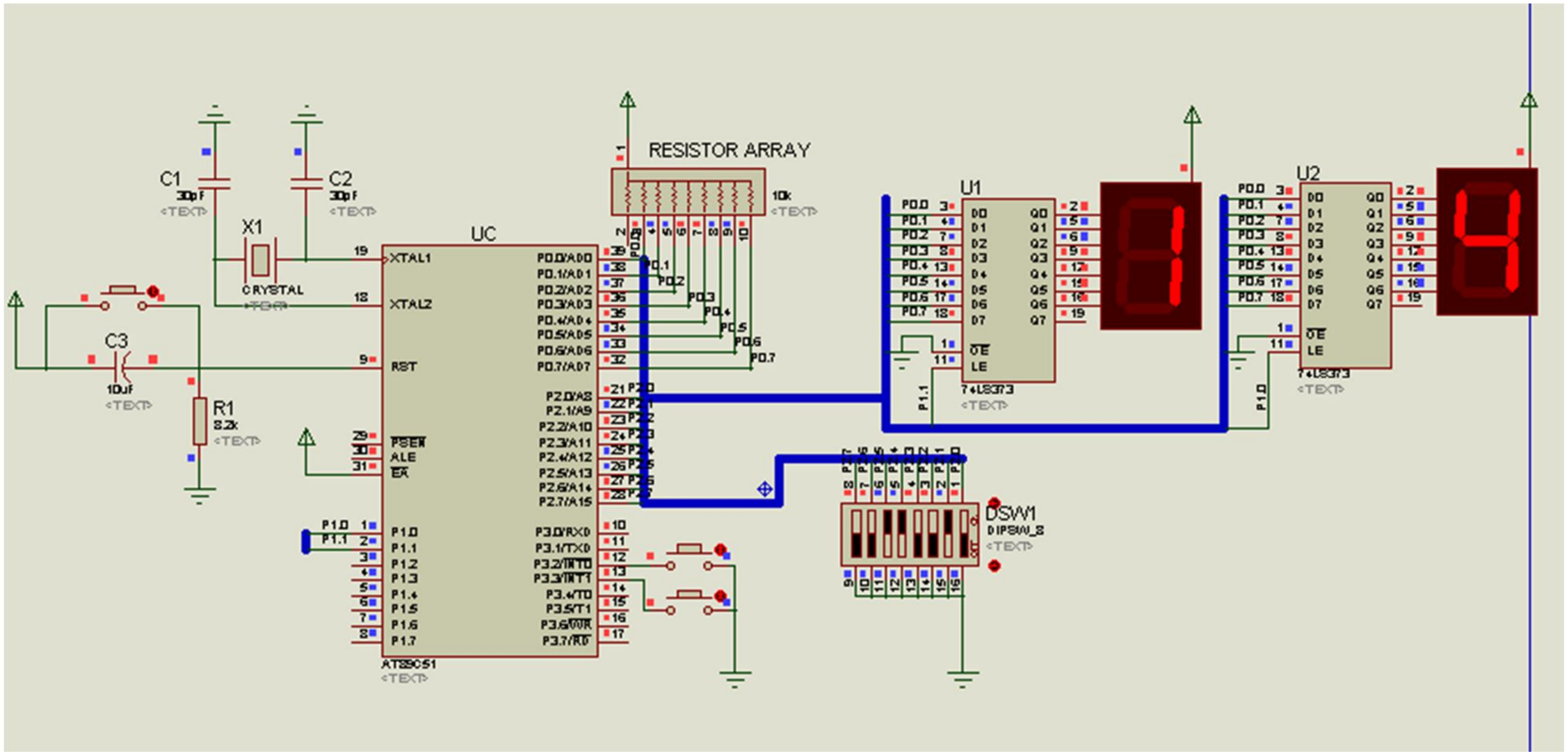
Interrupt	Name	Numbers
External Interrupt 0	(INT0)	0
Timer Interrupt 0	(TF0)	1
External Interrupt 1	(INT1)	2
Timer Interrupt 1	(TF1)	3
Serial Communication	(RI + TI)	4

Task Code



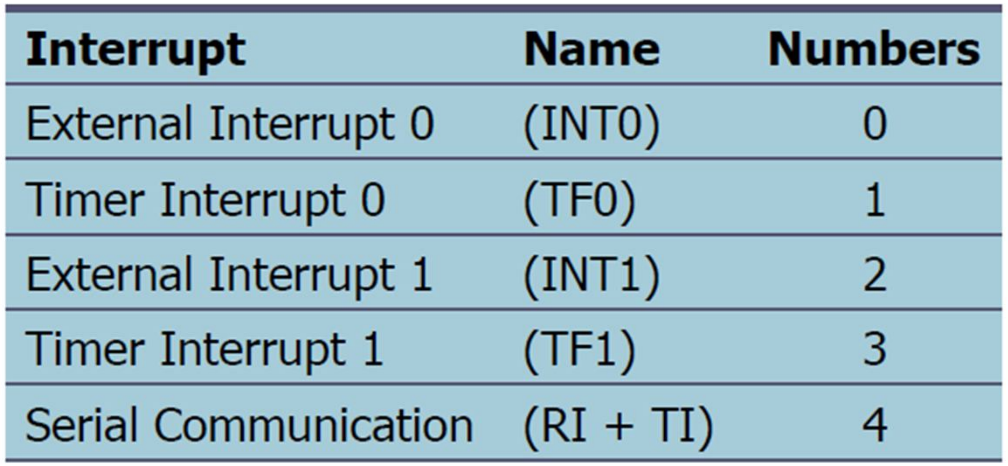
```c
#include<reg51.h>
unsigned char count;
void External0(void) interrupt 0
{ count++; }
void External1(void) interrupt 2
{ count=P2;}
//============Main Function=================//
void main(void)
{
  unsigned int a;
unsigned char Lookup[]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,
0x82,0xF8,0x80,0x90,0xA0,0x83,0xA7,0xA1,0x84,0x8E};

EA=1; //Enable All
IT0=1; // Make INT0 edge triggered
EX0=1; //Enable External Interrupt
IT1=1; // Make INT1 edge triggered
EX1=1; //Enable External Interrupt

while(1)
{
  P0=0xFF;
  P1=0x01;
  P0=Lookup[0x0F & count];

  //for(a=0;a<10;a++); optional

  P0=0xFF;
  P1=0x02;
  P0=Lookup[(0xF0 & count)>>4];
  //for(a=0;a<10;a++); optional
}
}
```

Proteus Simulation



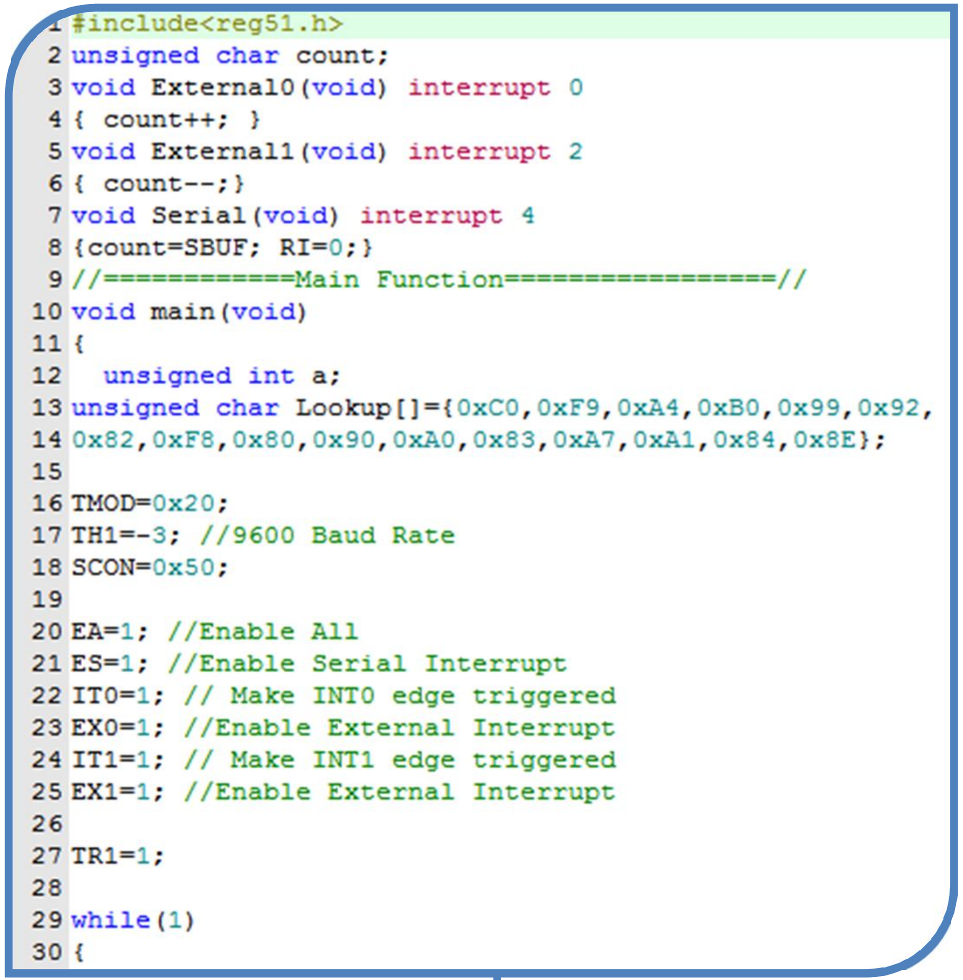
Todays Task 2

- Implement this on easy 8051 Kit.
- Make a counter from 00-FF and display its value on Seven Segment. The counter counts up when External Interrupt 0 is given. And it counts down when External Interrupt 1 is given.
- Make the interrupt edge triggered.
- Using the Serial Receive Interrupt receive a byte and load it as the new value of count.

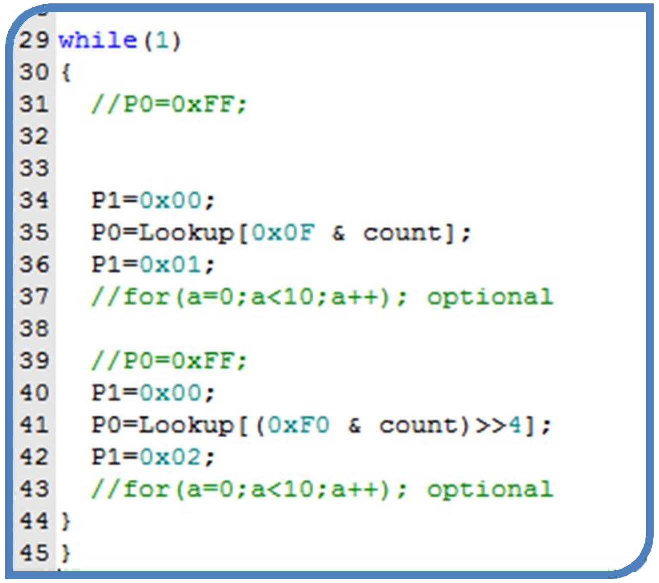


Interrupt	Name	Numbers
External Interrupt 0	(INT0)	0
Timer Interrupt 0	(TF0)	1
External Interrupt 1	(INT1)	2
Timer Interrupt 1	(TF1)	3
Serial Communication	(RI + TI)	4

Task Code

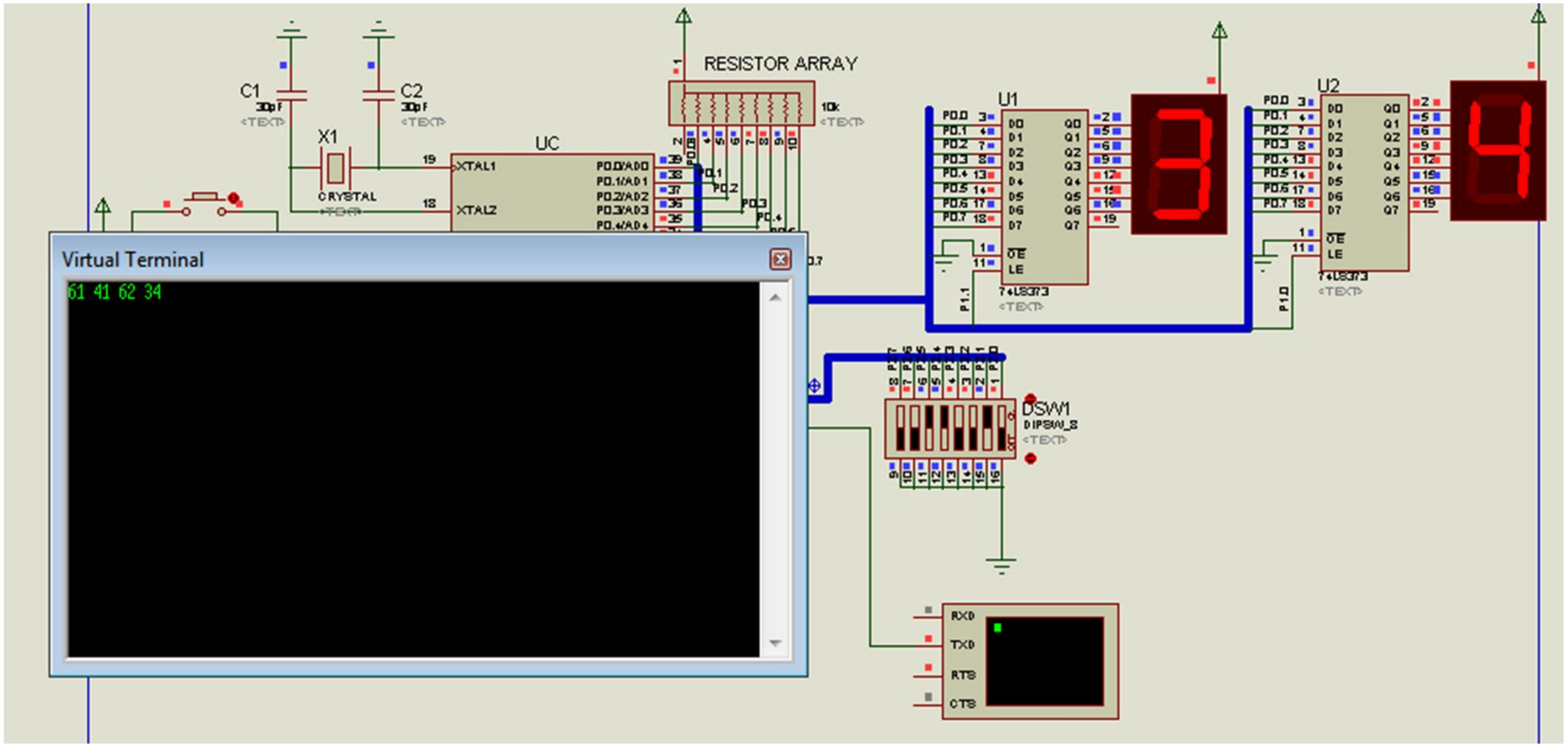


```c
#include<reg51.h>
unsigned char count;
void External0(void) interrupt 0
{ count++; }
void External1(void) interrupt 2
{ count--;}
void Serial(void) interrupt 4
{count=SBUF; RI=0;}
//============Main Function=================//
void main(void)
{
  unsigned int a;
unsigned char Lookup[]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,
0x82,0xF8,0x80,0x90,0xA0,0x83,0xA7,0xA1,0x84,0x8E};

TMOD=0x20;
TH1=-3; //9600 Baud Rate
SCON=0x50;

EA=1; //Enable All
ES=1; //Enable Serial Interrupt
IT0=1; // Make INT0 edge triggered
EX0=1; //Enable External Interrupt
IT1=1; // Make INT1 edge triggered
EX1=1; //Enable External Interrupt

TR1=1;

while(1)
{
```



```c
while(1)
{
  //P0=0xFF;


  P1=0x00;
  P0=Lookup[0x0F & count];
  P1=0x01;
  //for(a=0;a<10;a++); optional

  //P0=0xFF;
  P1=0x00;
  P0=Lookup[(0xF0 & count)>>4];
  P1=0x02;
  //for(a=0;a<10;a++); optional
}
}
```

Proteus Simulation



```
61 41 62 34
```

Assignment Next Week

- Implement Task1 on breadboard and bring along in next lab.
- Use P0 for 7Segment 1 and Use P1 for 7Segment 2. Hence no need to use latch.