Microprocessors and Microcontrollers (EE-231)

Lab-11

Objective

- Interrupts Programming in C
  - In Proteus
  - On 8051 development board

Interrupt

- An interrupt is an external or internal event that interrupts the microcontroller to inform it that a device needs its service.
- Difference b/w interrupt and polling?
- A very familiar example of polling
  - while(TI==0);
  - while(TF1==0);
- This way (by interrupt) microcontroller can do multiple tasks and handle multiple devices.
- It can also choose which device to serve which not to serve by masking.

Interrupt Service Routine

- When an interrupt is invoked, the microcontroller runs the interrupt service routine
- For every interrupt, there is a fixed location in memory that holds the address of its ISR
- The group of memory locations set aside to hold the addresses of ISRs is called interrupt vector table (IVT)

Steps in Executing an Interrupt

- Upon activation of an interrupt, the microcontroller goes through the following steps

1. It finishes the instruction it is executing and saves the address of the next instruction (PC) on the stack
2. It also saves the current status of all the interrupts internally
3. It jumps to a fixed location in memory, called the interrupt vector table, that holds the address of the ISR
4. The microcontroller gets the address of the ISR from the interrupt vector table and jumps to it
5. It starts to execute the ISR until it reaches the last instruction of the subroutine which is RETI (return from interrupt)
6. Upon executing the RETI instruction, the microcontroller returns to the place where it was interrupted
7. First, it gets the program counter (PC) address from the stack by popping the top two bytes of the stack into the PC
8. Then it starts to execute from that address

Interrupts in 8051

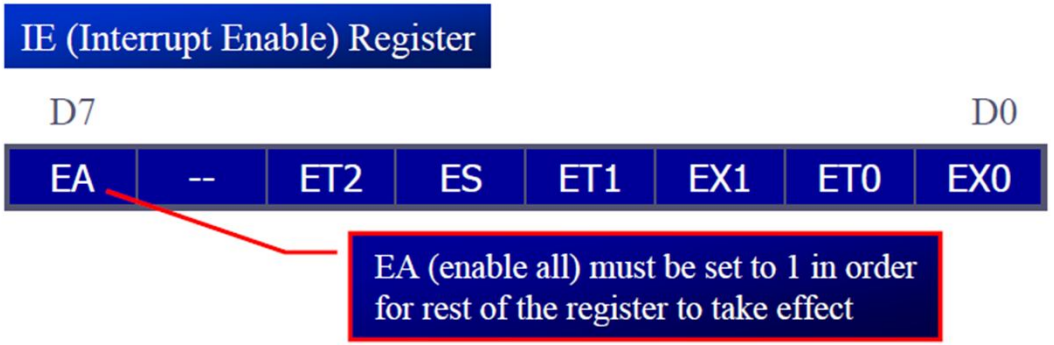
- The 8051 has **6** interrupts given as below:
- i)Reset ii)Timer0 iii)Timer1 iv)External0 v)External1 vi)Serial
- Reset – power-up reset
- Two interrupts are set aside for the timers: one for timer 0 and one for timer 1
- Two interrupts are set aside for hardware external interrupts:P3.2 and P3.3 are for the external hardware interrupts INT0 (or EX1), and INT1 (or EX2)
- Serial communication has a single interrupt for both receive (RI=1) and transfer (TI=1)

Interrupt vector table

Interrupt	ROM Location (hex)	Pin
Reset	0000	9
External HW (INT0)	0003	P3.2 (12)
Timer 0 (TF0)	000B	
External HW (INT1)	0013	P3.3 (13)
Timer 1 (TF1)	001B	
Serial COM (RI and TI)	0023	

IE Register

- Upon reset, all interrupts are disabled (masked), meaning that none will be responded to by the microcontroller if they are activated
- The interrupts must be enabled by software using IE (Interrupt Enable) Register. It is bit addressable.



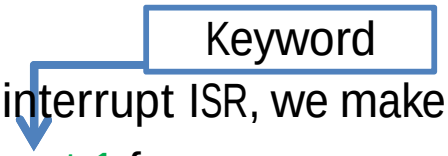
EA	Enable All
ET2	Enable Timer 2
ET1	Enable Timer 1
ES	Enable Serial
EX1	Enable External 1
ET0	Enable Timer 0
EX0	Enable External 0

Interrupt Programming in C

- In C Programming, we don't need to worry about vector address.
- The compiler have a unique number for each interrupt

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
Timer 2 (8052 only)	(TF2)	5

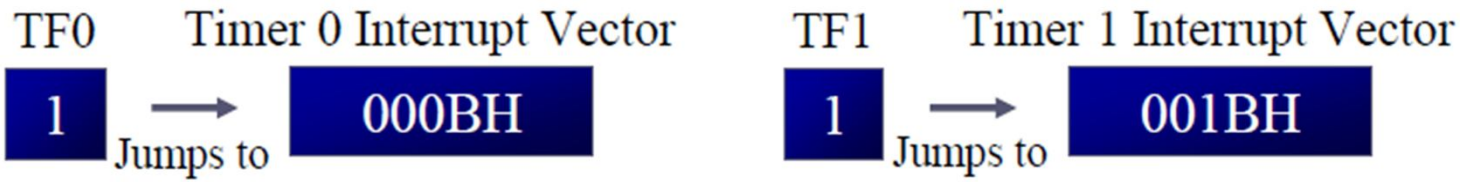
- We activate the interrupt in the main program and we write its ISR as a function of C.
- For example for a timer interrupt ISR, we make a function as below




```
void my_ISR (void) interrupt 1 {
///// Function body }
```

Timer Interrupt

- If the timer interrupt in the IE register is enabled, whenever the timer rolls over, TF is raised, and the microcontroller is interrupted &
- Jumps to the interrupt vector table to service the ISR
- In this way, the microcontroller can do other until it is notified that the timer has rolled over
- No need to continuously monitor i.e. while(TFX==0);



- void timer_one (void) interrupt 3 {

Timer Interrupt 1	(TF1)	3

- void timer_zero (void) interrupt 1 {

Timer Interrupt 0	(TF0)	1

Timer Interrupt

Example 11-14

Write a C program that continuously gets a single bit of data from P1.7 and sends it to P1.0, while simultaneously creating a square wave of 200 µs period on pin P2.5. Use Timer 0 to create the square wave. Assume that XTAL = 11.0592 MHz.

Solution:

We will use timer 0 mode 2 (auto-reload). One half of the period is 100 µs. 100/1.085 µs = 92, and TH0 = 256 - 92 = 164 or A4H

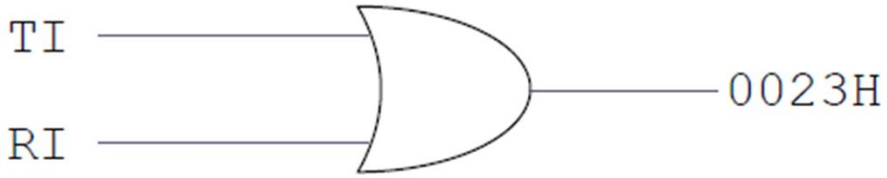
```
#include <reg51.h>
sbit SW   =P1^7;
sbit IND  =P1^0;
sbit WAVE =P2^5;
void timer0(void) interrupt 1 {
  WAVE=~WAVE;  //toggle pin
}
void main() {
  SW=1;          //make switch input
  TMOD=0x02;
  TH0=0xA4;      //TH0=-92
  IE=0x82;       //enable interrupt for timer 0
  while (1) {
    IND=SW;      //send switch to LED
  }
}
```

Making this function an ISR

We make the controller copy P1.7 to P1.0 continuously. While for the square wave we use interrupt. i.e. every 100 us we generate an interrupt to request our controller to toggle the P2.5. After toggling it, controller goes back to its regular job i.e. copying P1.7 to P1.0

Serial Interrupt

- In the 8051 there is only one interrupt set aside for serial communication
- If enabled, this interrupt occurs whenever either of TI or RI flag is raised i.e. =1.
- In that ISR we must examine the TI and RI flags to see which one caused the interrupt and respond accordingly



- The serial interrupt is used mainly for receiving data

Serial Communication	(RI + TI)	4

- void Serial_ISR (void) interrupt 4 {

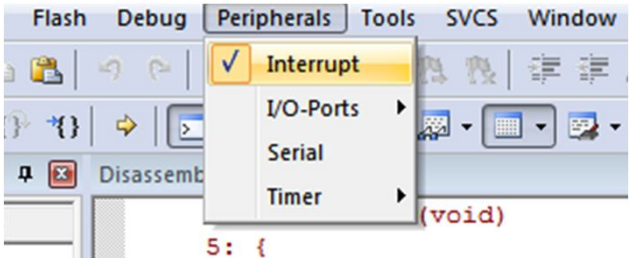
Serial Interrupt

Example:
Write a C code to do the following.
1) Continuously Copy P0 to P1.
2) Receive data serially and send it to P2.

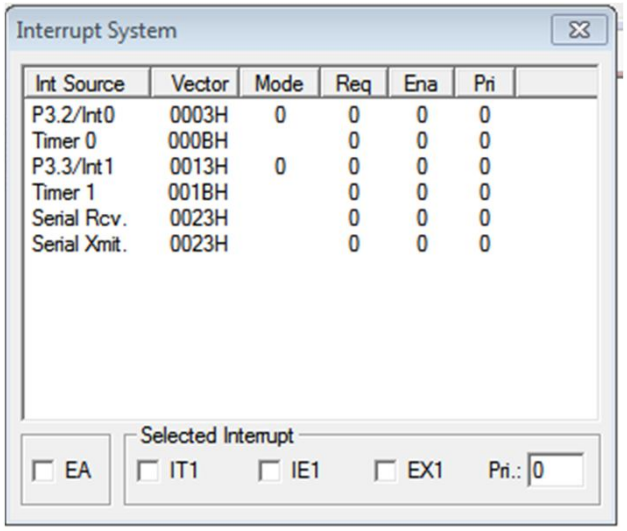
```
#include <reg51.h>
void serial_ISR(void) interrupt 4 {
P2=SBUF;
RI=0 }
void main (void) {
unsigned char mybyte;
P0=0xFF;// Make P0 an input
TMOD=0x20;
TH1=-3; // 9600 baudrate
EA=1; // Enable all
ES=1; // Enable serial interrupt
TR1=1; //Start Timer 1 for baudrate generation
while(1)
mybyte=P0;
P1=mybyte;
}
```

Viewing Interrupt Status in Keil

- In Keil debugger we can view the status of interrupt by going in to Peripheral menu and selecting ‘interrupt’



- In the interrupt window, we can see status and assert manually any flag of the interrupt.



Todays Task 1

- Implement this in Proteus and then on easy 8051 Kit.
- Using interrupt for both timer 0 and 1 generate 2.5 KHz frequency at P2.0 and 3 KHz at P2.1. Also display a count from 0-9 at a seven segment connected to P1 using an appropriate delay. (MSDelay(250); maybe….)
- Firs implement it on Proteus and Then on 8051 Kit.

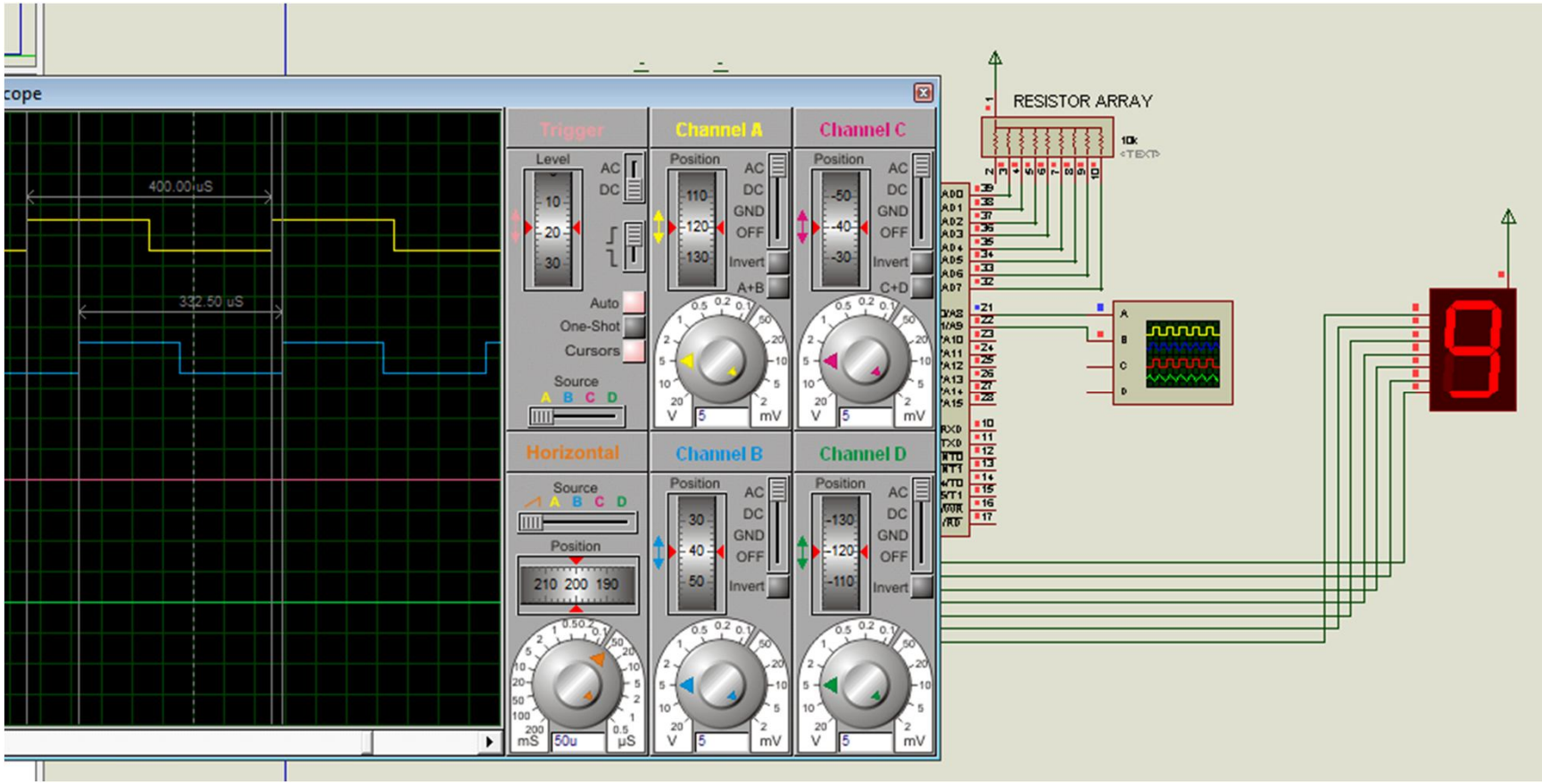
Task Code

```
#include<reg51.h>
sbit wave1=P2^0;
sbit wave2=P2^1;
void Timer_0_ISR() interrupt 1
{
wave1=~wave1;
}

void Timer1_1_ISR(void) interrupt 3
{
wave2=~wave2;
}

void MSDelay (unsigned int);
void main(void)
{
unsigned char x;
unsigned char Lookup[]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,
0x82,0xF8,0x80,0x90,0xA0,0x83,0xA7,0xA1,0x84,0x8E};
TMOD=0x22;
TH0=-184; // For 2.5 KHz
TH1=-153; // For 3 KHz
EA=1; //Enable All
ET0=1; //Enable Timer 0 Interrupt
ET1=1; //Enable Timer 1 Interrupt
TR0=1; // Start Timer 0
TR1=1; // Start Timer 1
while(1)
{
  for(x=0;x<10;x++)
  {
  P1=Lookup[x];
  MSDelay(5);
  }
}
```

Proteus Simulation



Todays Task 2

- Implement this in Proteus and then on easy 8051 Kit.
- Generate a 2.5 KHz Square wave on P2.0.
- While simultaneously implement a seven segment based 0-F up-down counter. Seven segment be connected with P1, will count up if serial port receives ‘u’ and will count down if serial port receives ‘d’. Use serial interrupt to do that.

Task Code

```
#include<reg51.h>
sbit wave=P2^0;
bit dir;
void Timer_0_ISR() interrupt 1
{
wave=~wave;
}
void Serial_ISR(void) interrupt 4
{
  unsigned char y;
  y=SBUF;
  RI=0;
  if(y=='u')
    dir=1;
  else if(y=='d')
    dir=0;
}
void MSDelay (unsigned int);
void main(void)
{
unsigned char x;
unsigned char Lookup[]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,
0x82,0xF8,0x80,0x90,0xA0,0x83,0xA7,0xA1,0x84,0x8E};
TMOD=0x22;
SCON=0x50;
TH0=-184; // For 2.5 KHz
TH1=-3; // For 9600 baudrate
EA=1; //Enable All
ET0=1; //Enable Timer 0 Interrupt
ES=1; //Enable Serial Interrupt
TR0=1; // Start Timer 0
TR1=1; // Start Timer 1 for Baudrate Generation
```

```
while(1)
{
  if(dir==1)
    x++;
  else
    x--;
  P1=Lookup[x&0x0F];
  MSDelay(250);
}
}
void MSDelay (unsigned int itime)
{
unsigned int x,y;
for(x=0;x<itime;x++)
for(y=0;y<114;y++);
}
```

Proteus Simulation