Interrupts (Part-2)

Chapter 11

Lecture 6

Bilal Habib

DCSE, UET

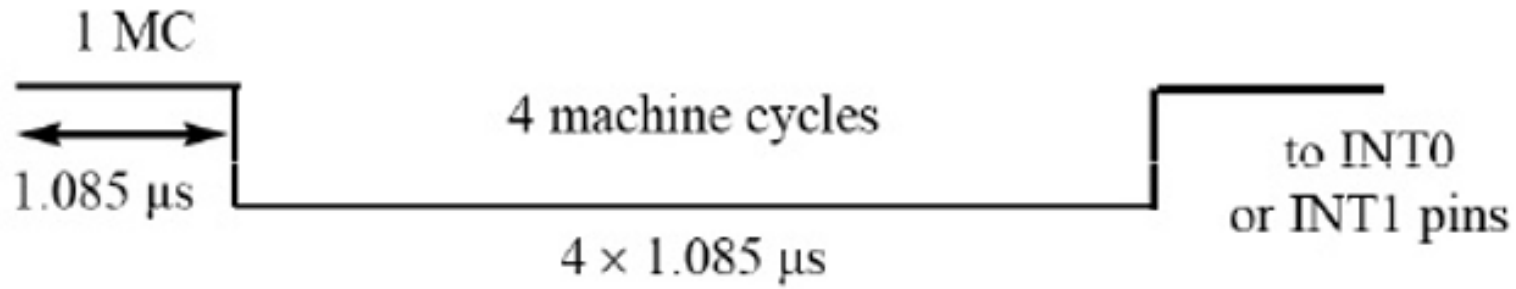
EXTERNAL HARDWARE INTERRUPTS

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
  - Then the microcontroller stops whatever it is doing and jumps to the interrupt vector table to service that 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
- This is called a *level-triggered* or *level-activated* interrupt and is the default mode upon reset of the 8051

Level triggered interrupt

Sampling the low level-triggered interrupt



Note: On RESET, IT0 (TCON.0) and IT1 (TCON.2) are both low, making external interrupts level-triggered.

Assuming fclk = 11.059 MHz

EXTERNAL HARDWARE INTERRUPTS

Edge-Triggered Interrupt

- To make INT0 and INT1 edge-triggered interrupts, we must program the bits of the TCON register
  - The TCON register holds, among other bits, the IT0 and IT1 flag bits that determine level- or edge-triggered mode of the hardware interrupt
    - IT0 and IT1 are bits D0 and D2 of the TCON register
    - They are also referred to as TCON.0 and TCON.2 since the TCON register is bit-addressable

EXTERNAL HARDWARE INTERRUPTS

Edge-Triggered Interrupt (cont')

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

IE1	TCON.3	External interrupt 1 edge flag. Set by CPU when the external interrupt edge (H-to-L transition) is detected. Cleared by CPU when the interrupt is processed
IT1	TCON.2	Interrupt 1 type control bit. Set/cleared by software to specify falling edge/low-level triggered external interrupt
IE0	TCON.1	External interrupt 0 edge flag. Set by CPU when the external interrupt edge (H-to-L transition) is detected. Cleared by CPU when the interrupt is processed
IT0	TCON.0	Interrupt 0 type control bit. Set/cleared by software to specify falling edge/low-level triggered external interrupt

Edge triggered vs level triggered interrupt

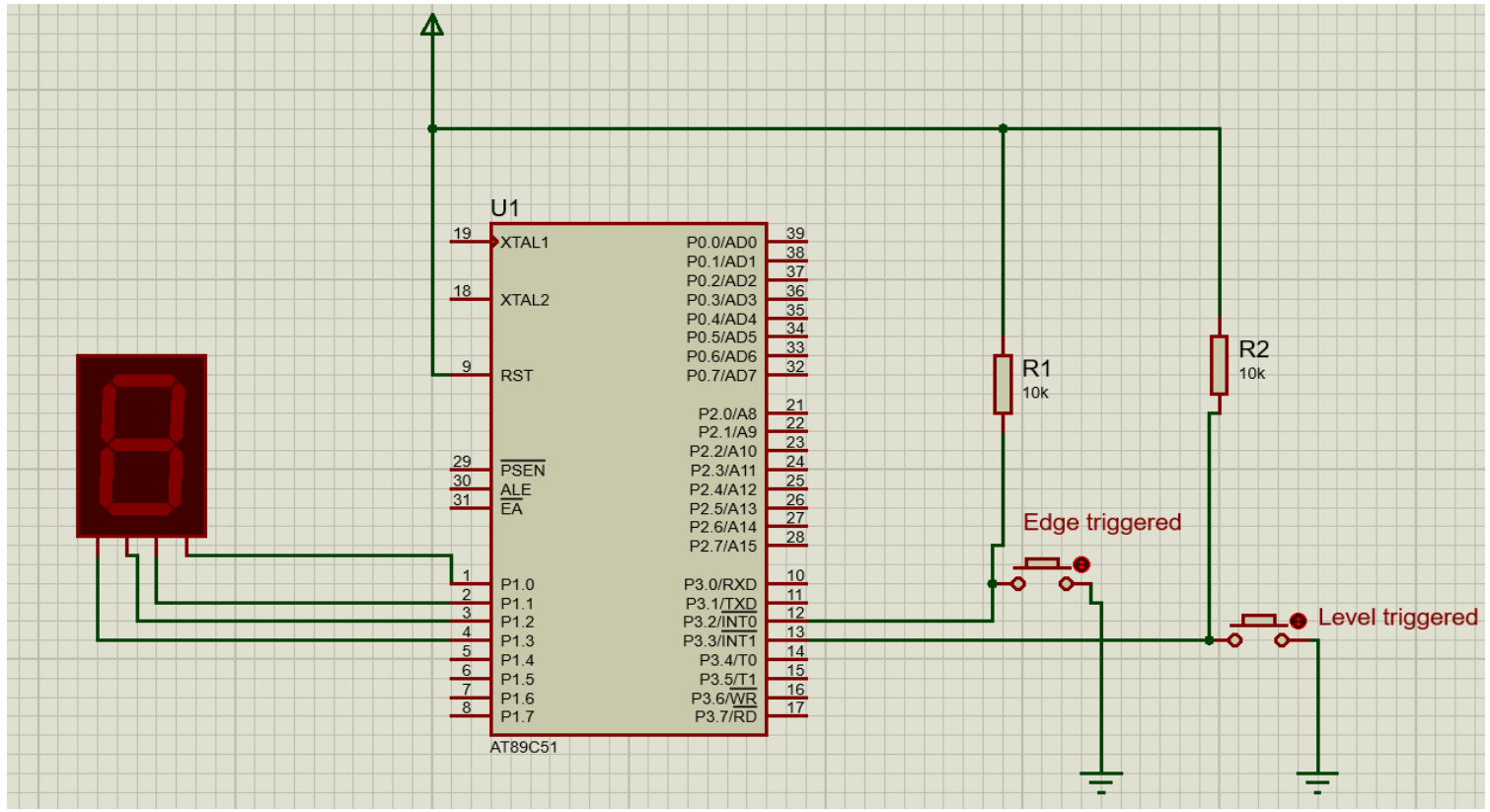
```c
#include <reg51.h>
#include <stdio.h>

unsigned char count=0;

void ext_int_0() interrupt 0
{
    P1 = count++;
}

void ext_int_1() interrupt 2
{
   P1 = count--;
}

void main()
{
   P3 |= 0x0c; // Configure the INT0 & INT1 pins as Inputs
   EX0 = 1;    // Enable INT0
   EX1 = 1;    // Enable INT1
   EA  = 1;    // Enable Global Interrupt bit
   IT0 = 1;     // INT0 control bit H->L (Edge triggered)
   IT1 = 0;     // INT1 control (Level triggered)
   P1 = 0;
   while(1)
   {   }
}
```



INTERRUPT PRIORITY

- When the 8051 is powered up, the priorities are assigned according to the following
  - In reality, the priority scheme is nothing but an internal polling sequence in which the 8051 polls the interrupts in the sequence listed and responds accordingly

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)

Interrupt Priority

- **Example 11-11**
- Discuss what happens if interrupts INT0, TF0, and INT1 are activated at the same time. Assume priority levels were set by the power-up reset and the external hardware interrupts are edge-triggered.
- **Solution:**
- If these three interrupts are activated at the same time, they are latched and kept internally. Then the 8051 checks all five interrupts according to the sequence listed in Table 11-3. If any is activated, it services it in sequence. Therefore, when the above three interrupts are activated, IE0 (external interrupt 0) is serviced first, then timer 0 (TF0), and finally IE1 (external interrupt 1).

INTERRUPT PRIORITY (cont')

- 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
    - While these interrupts have a higher priority than others, they are serviced according to the sequence of Table 11-13

INTERRUPT PRIORITY (cont')

Interrupt Priority Register (Bit-addressable)

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

--	IP.7	Reserved
--	IP.6	Reserved
PT2	IP.5	Timer 2 interrupt priority bit (8052 only)
PS	IP.4	Serial port interrupt priority bit
PT1	IP.3	Timer 1 interrupt priority bit
PX1	IP.2	External interrupt 1 priority bit
PT0	IP.1	Timer 0 interrupt priority bit
PX0	IP.0	External interrupt 0 priority bit

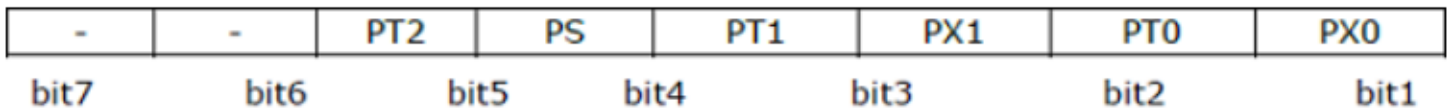
Priority bit=1 assigns high priority

Priority bit=0 assigns low priority

IP (Interrupt Priority) Register

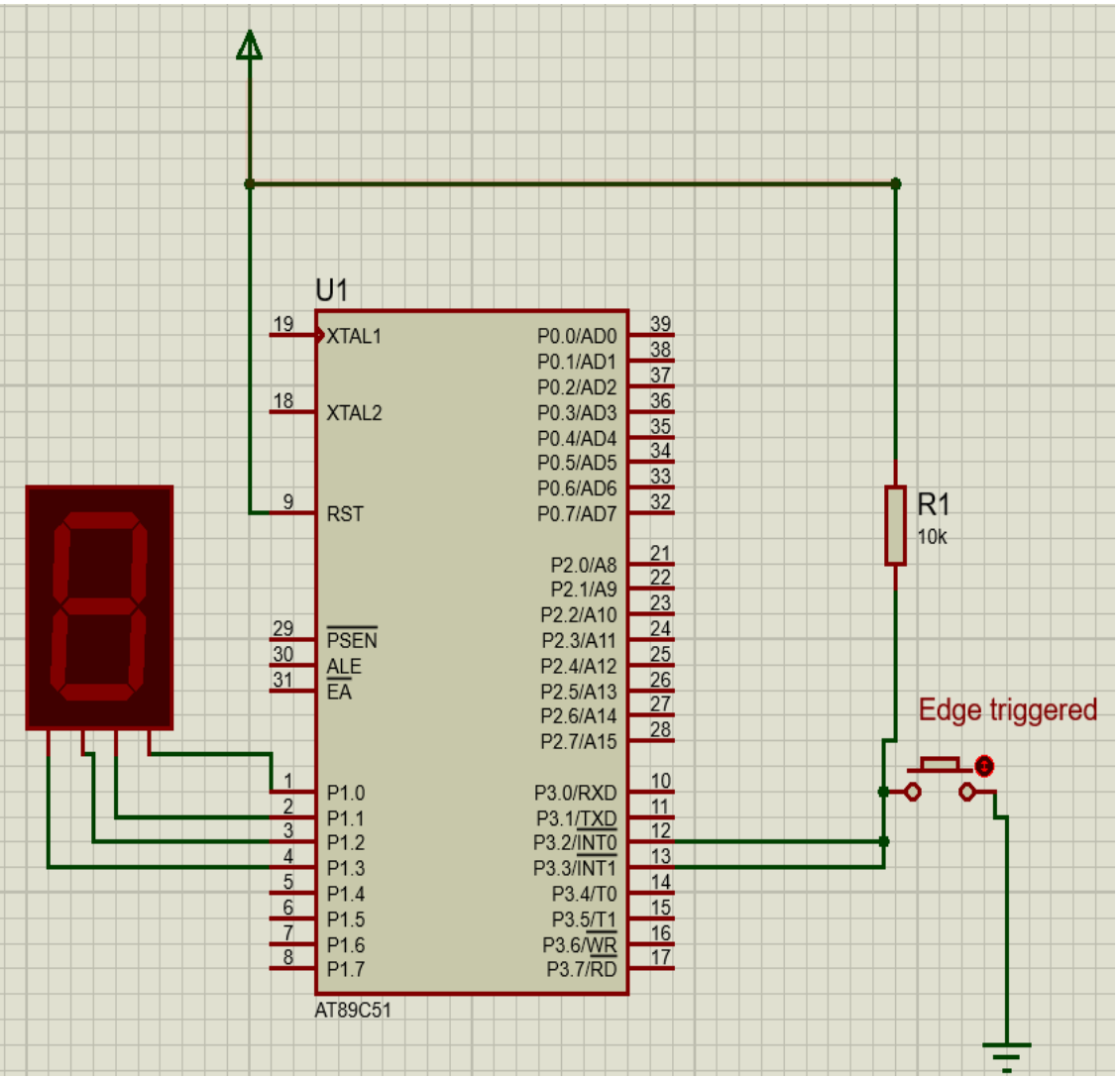
We can change the priority levels of the interrupts by changing the corresponding bit in the Interrupt Priority (IP) register as shown in the following figure.

- A low priority interrupt can only be interrupted by the high priority interrupt, but not interrupted by another low priority interrupt.
- If two interrupts of different priority levels are received simultaneously, the request of higher priority level is served.
- If the requests of the same priority levels are received simultaneously, then the internal polling sequence determines which request is to be serviced.



-	IP.6	Reserved for future use.
-	IP.5	Reserved for future use.
PS	IP.4	It defines the serial port interrupt priority level.
PT1	IP.3	It defines the timer interrupt of 1 priority.
PX1	IP.2	It defines the external interrupt priority level.
PT0	IP.1	It defines the timer0 interrupt priority level.
PX0	IP.0	It defines the external interrupt of 0 priority level.

Interrupt Priority




```c
#include <reg51.h>
#include <stdio.h>
unsigned char count=0;
void ext_int_0() interrupt 0 //Edge triggered
{
     P1 = count++;
}
void ext_int_1() interrupt 2 //Edge triggered
{
    P1 = count--;
}
void main()
{
    P3 |= 0x0c; // Configure the INT0 & INT1 pins as Inputs
/////////////// IE ////////////////////
    EX0 = 1;     // Enable INT0
    EX1 = 1;     // Enable INT1
    EA  = 1;     // Enable Global Interrupt bit
/////////////// TCON ////////////////////
    IT0 = 1;      // INT0 control bit H->L (Edge triggered)
    IT1 = 1;      // INT1 control bit H->L (Edge triggered)
/////////////// IP ////////////////////
    PX1 = 1;      // INT1 will have higher priority
    PX0 = 0;      // INT0 will have lower priority
    P1 = 0;
     while(1)
    {    }
}
```

By default INT0 has higher priority

Interrupt Priority

- Example 11-12
- (a) Program the IP register to assign the highest priority to INT1(external interrupt 1), then (b) discuss what happens if **INT0**, **INT1**, and **TF0** are activated at the same time. Assume the interrupts are both edge-triggered.
- Solution:
- (a) MOV IP,#00000100B ;IP.2=1 assign INT1 higher priority. The instruction SETB IP.2 also will do the same thing as the above line since IP is bit-addressable.
- (b) The instruction in Step (a) assigned a higher priority to **INT1** than the others; therefore, when INT0, INT1, and TF0 interrupts are activated at the same time, the 8051 services INT1 first, then it services INT0, then TF0. This is due to the fact that INT1 has a higher priority than the other two because of the instruction in Step (a). The instruction in Step (a) makes both the INT0 and TF0 bits in the IP register 0. As a result, the sequence in Table 11-3 is followed which gives a higher priority to INT0 over TF0

Interrupt Priority

- **Example 11-13**
- Assume that after reset, the interrupt priority is set the instruction MOV IP,#00001100B. Discuss the sequence in which the interrupts are serviced.
- **Solution:**
- The instruction “MOV IP #00001100B” (B is for binary) and timer 1 (TF1)to a higher priority level compared with the reset of the interrupts. However, since they are polled according to Table, they will have the following priority.

  Highest Priority External Interrupt 1 (INT1)

  Timer Interrupt 1 (TF1)

  External Interrupt 0 (INT0)

  Timer Interrupt 0 (TF0)

  Lowest Priority Serial Communication (RI+TI)

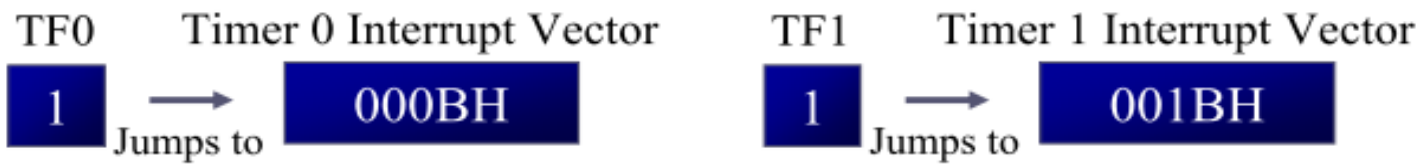
INTERRUPT PRIORITY

Interrupt inside an Interrupt

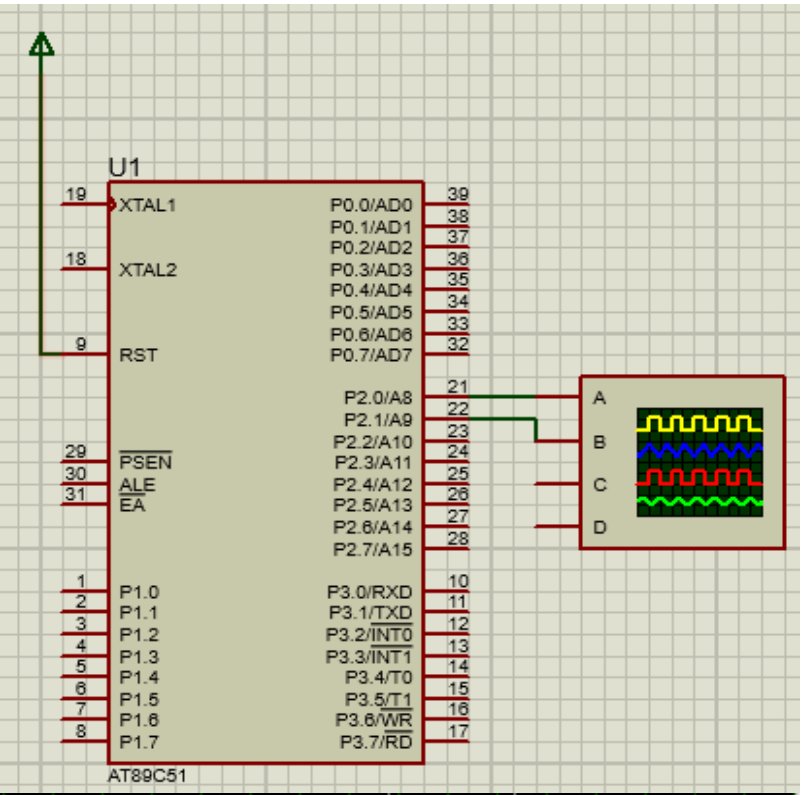
- In the 8051 a low-priority interrupt can be interrupted by a higher-priority interrupt but not by another low-priority interrupt
  - Although all the interrupts are latched and kept internally, no low-priority interrupt can get the immediate attention of the CPU until the 8051 has finished servicing the high-priority interrupts

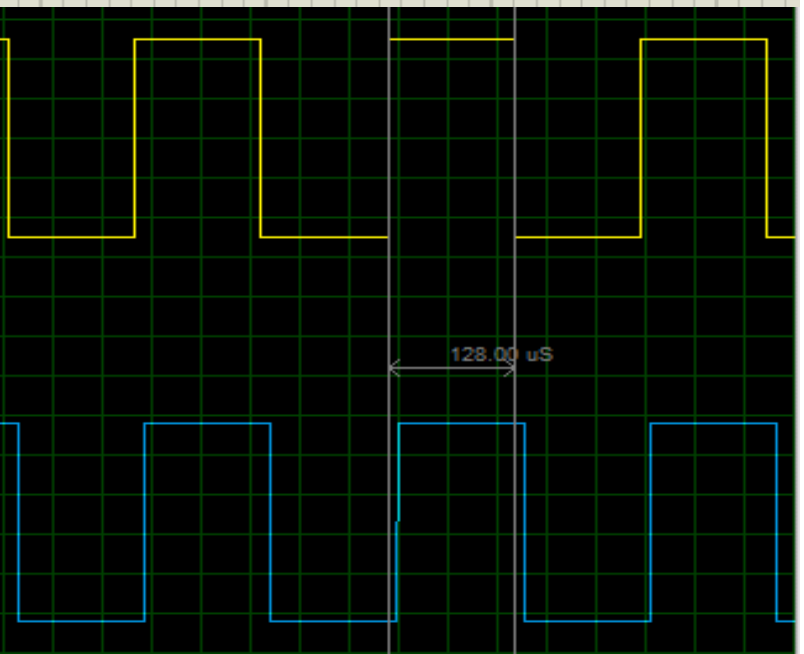
TIMER INTERRUPTS

- The timer flag (TF) is raised when the timer rolls over
  - In polling TF, we have to wait until the TF is raised
    - The problem with this method is that the microcontroller is tied down while waiting for TF to be raised, and can not do anything else
  - Using interrupts solves this problem and, avoids tying down the controller
    - If the timer interrupt in the IE register is enabled, whenever the timer rolls over, TF is raised, and the microcontroller is interrupted in whatever it is doing, and 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



Using two timers simultaneously






```c
#include <reg51.h>
sbit pin1 = P2^0;
sbit pin2 = P2^1;
void timer0() interrupt 1
{
    pin1 = ~pin1;
}
void timer1() interrupt 3
{
   pin2 = ~pin2;
}
void main()
{
   TMOD = 0x22; // Both timers in Auto-Reload mode
   IE = 0x8A; // Enable Global Interrupt bit and both Timer bits
   TH0 = 0x80;
   TH1 = 0x80;
   TR0 = 1;     // Start Timer 0
   TR1 = 1;     // Start Timer 1
   while(1);
}
```

PROGRAMMING IN C

- The 8051 compiler have extensive support for the interrupts
  - They assign a unique number to each of the 8051 interrupts

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

  - It can assign a register bank to an ISR
    - This avoids code overhead due to the pushes and pops of the R0 – R7 registers