Interrupts programming

It is a sub-routine calls that given by the microcontroller when some other program with high priority is request for acquiring the system buses than interrupt occur in current running program.

Interrupts provide a method to postpone or delay the current process, performs a sub-routine task and then restart the standard program again.

Types of interrupt in 8051 Microcontroller

Let's see the five sources of interrupts in 8051 Microcontroller:

- Timer 0 overflow interrupt - TF0
- External hardware interrupt - INT0
- Timer 1 overflow interrupt - TF1
- External hardware interrupt - INT1
- Serial communication interrupt - RI/TI

The timer and serial interrupts are internally produced by the microcontroller, whereas the external interrupts are produced by additional interfacing devices or switches that are externally connected with the microcontroller. These external interrupts can be level triggered or edge triggered.

When interrupt occur then the microcontroller executes the interrupt service routine. Therefore the memory location corresponds to interrupt enables it. Consider the interrupt corresponding to the memory location is shown in the interrupt vector table below.

Interrupt Number	Interrupt Description	Address
0	EXTERNAL INT 0	0003h
1	TIMER/COUNTER 0	000Bh
2	EXTERNAL INT 1	0013h
3	TIMER/COUNTER 1	001Bh
4	SERIAL PORT	0023h

Interrupt Enable (IE) Register

IE register is used for enabling and disabling the interrupt. This is a bit addressable register in which EA value must be set to one for enabling interrupts. The individual bits in this register

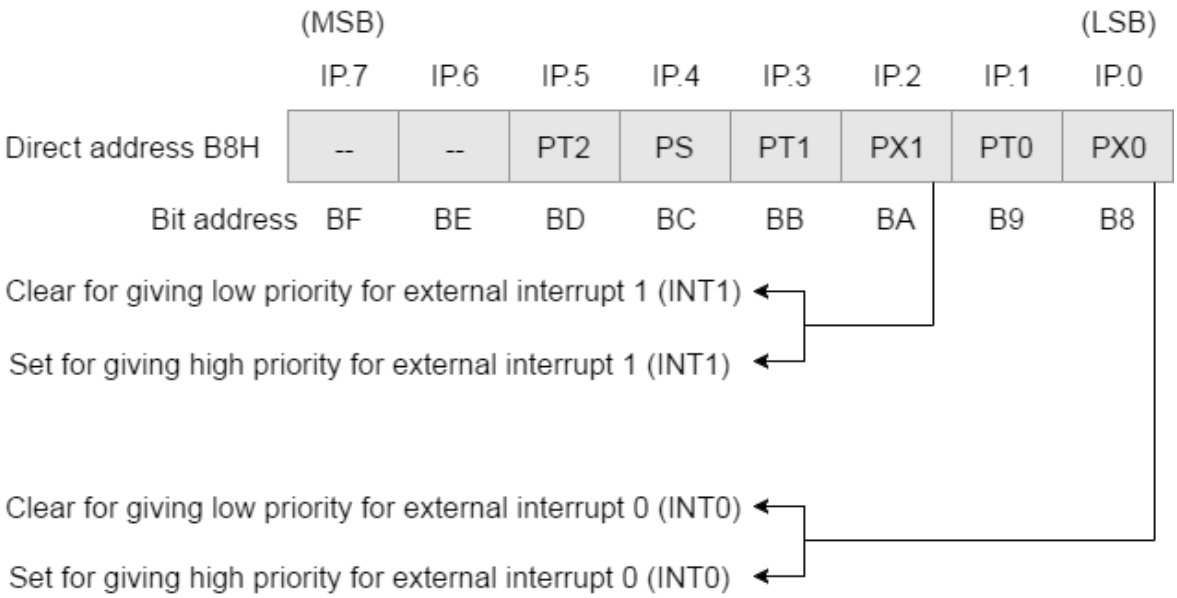
enables the particular interrupt like timer, serial and external inputs. Consider in the below IE register, bit corresponds to 1 activate the interrupt and 0 disable the interrupt.

EA	--	--	ES	ET1	EX1	ET0	EX0

EA	IE.7	Disables all interrupts, If EA=0, no interrupt will be acknowledged. If EA=1, interrupt source is individually enable or disabled by setting or clearing its enable bit.
--	IE.6	Not implemented, reserved for future use*.
--	IE.5	Not implemented, reserved for future use*.
ES	IE.4	Enable or disable the Serial port interrupt.
ET1	IE.3	Enable or disable the Timer 1 overflow interrupt.
EX1	IE.2	Enable or disable External interrupt 1.
ET0	IE.1	Enable or disable the Timer 0 overflow interrupt.
EX0	IE.0	Enable or disable External interrupt 0.

Interrupt Priority Register (IP)

Using IP register it is possible to change the priority levels of an interrupts by clearing or setting the individual bit in the Interrupt priority (IP) register as shown in figure. It allows the low priority interrupt can interrupt the high-priority interrupt, but it prohibits the interruption by using another low-priority interrupt. If the priorities of interrupt are not programmed, then microcontroller executes the instruction in a predefined manner and its order are INT0, TF0, INT1, TF1, and SI.



1. **Timer Interrupt Programming**: In microcontroller Timer 1 and Timer 0 interrupts are generated by time register bits TF0 AND TF1. This timer interrupts programming by C code involves:
    - Selecting the configuration of TMOD register and their mode of operation.
    - Enables the IE registers and corresponding timer bits in it.
    - Choose and load the initial values of TLx and THx by using appropriate mode of operation.
    - Set the timer run bit for starting the timer.
    - Write the subroutine for a timer and clears the value of TRx at the end of the subroutine.

Let's see the timer interrupt programming using Timer0 model for blinking LED using interrupt method:

```
1.  #include< reg51 .h>
2.  sbit Blink Led = P2^0;  // LED is connected to port 2 Zeroth pin
3.  void timer0_ISR (void) interrupt 1  //interrupt no. 1 for Timer0
4.  {
5.  Blink Led=~Blink Led;   // Blink LED on interrupt
6.  TH0=0xFC;   // loading initial values to timer
7.  TL0=0x66;
8.  }
9.  void main()
```

```
10. {
11. TMOD=0x0l;      // mode 1 of Timer0
12. TH0 = 0xFC:    // initial value is loaded to timer
13. TL0 = 0x66:
14. ET0 =1;         // enable timer 0 interrupt
15. TR0 = 1;        // start timer
16. while (1);      // do nothing
17.  }
```