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

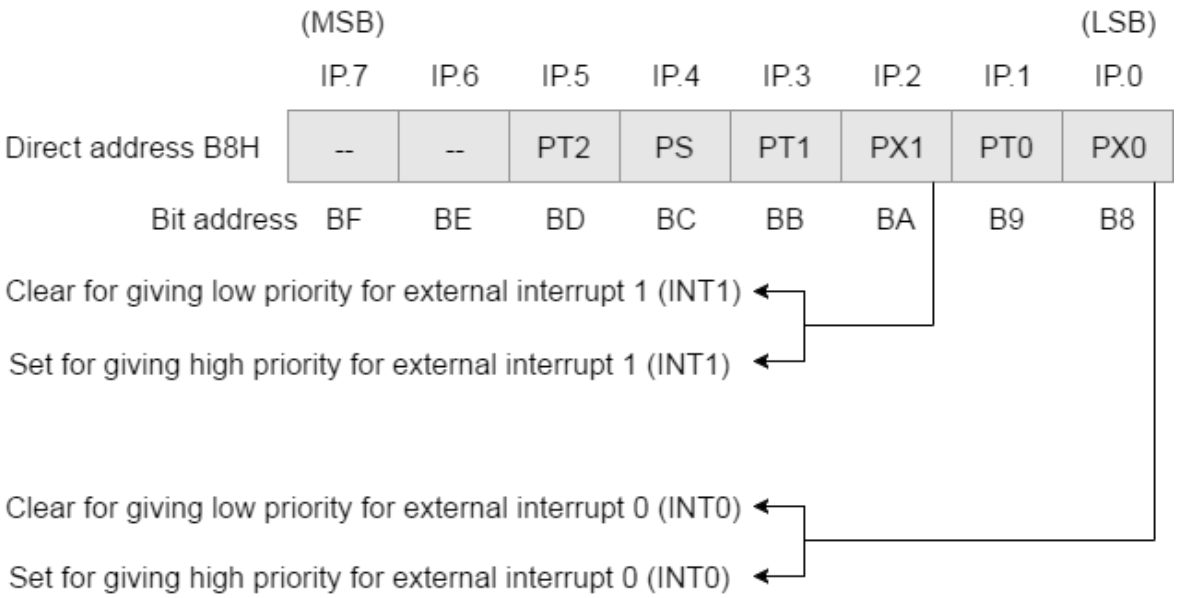
enables the particular interrupt like timer, serial and external inputs. Consider in the below IE register, bit corresponds to 1 activate the interrupt and 0 disable the interrupt.

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. **Serial Communication Interrupt Programming** It is used when there is a need to send or receive data. Since one interrupt bit is used for both Transfer Interrupt (TI) and Receiver Interrupt (RI) flags, Interrupt Service Routine (ISR) must examine these flags for knowing the actual interrupt. By the logical OR operation of RI and TI flags causes the interrupt and it is clear by the software alone. Consider the steps involved in serial communication interrupt programming are:-
    - Configure the Interrupt Enable register for enabling serial interrupt.
    - Configure the SCON register for performing transferring and receiving operation.
    - Write a subroutine for given interrupt with appropriate function.

    Let's see the program for sending 'E' through serial port with 9600 baud rate using Serial Interrupt:

```
1. void main()
2. {
3. TMOD = 0x20:
4. TH1= 0xFD;      // baud rate for 9600 bps
5. SCON = 0x50;
6. TR1=1;
7. EA=I;
8. whlle(l);
9. }
10. void ISR_Serial(void) interrupt 4
```

```
11. {
12. if(TI==I)
13. {
14. SBUF= ?E?;
15. TI=0;
16. }
17. else
18. RI =0;
19. }
```