INTERRUPTS AND THEIR HANDLING IN 8051

A single microcontroller can serve several devices. There are two ways to do that:

- Interrupts
- Polling.

Interrupts

In the interrupt method, whenever the device needs its service, the device notifies the microcontroller by sending an interrupt signal. Upon receiving an interrupt signal, the microcontroller interrupts whatever it is doing and serves the device. The program associated with the interrupt is called the interrupt service routine (ISR) or interrupt handler.

Interrupt service routine

For every interrupt, there must be an interrupt service routine (ISR), or interrupt handler. When an interrupt arises, the microcontroller runs the interrupt service routine.

Interrupt vector table

The group of memory locations set aside to hold the addresses of ISRs is called the interrupt vector table.

Advantages of interrupts

The advantage of interrupts is that the microcontroller can serve many devices. Each device can get the attention of the microcontroller based on the priority assigned to it.

In the interrupt method the microcontroller can also ignore (mask) a device request for service.

Steps in executing an interrupt

When an interrupt is activated, 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 (i.e., not on the stack).
3. It jumps to a fixed location in memory called the interrupt vector table and holds the address of the interrupt service routine.
4. The microcontroller gets the address of the ISR from the interrupt vector table and jumps to it. It starts to execute the interrupt service subroutine until it reaches the last instruction of the subroutine, which is RETI (return from interrupt).

5. Upon executing the RETI instruction, the microcontroller returns to the place where it was interrupted. First, it gets the program counter (PC) address from the stack by popping the top two bytes of the stack into the PC. Then it starts to execute from that address.

Interrupt vector table

The interrupts in 8051 are allocated as follows

Interrupt	**ROM Location(Hex)**
Reset	0000
External hardware interrupt 0(INT0)	0003
Timer 0 interrupt(TF0)	000B
External hardware interrupt 1(INT1)	0013
Timer 1 interrupt(TF1)	001B
Serial COM interrupt (RI and TI)	0023

Enabling and disabling an interrupt

Upon reset, all interrupts are disabled (masked) i.e., none will be responded to microcontroller. There is a register called IE (interrupt enable) that is responsible for enabling (unmasking) and disabling (masking) the interrupts.

IE (interrupt enable) Register

D7							D0
EA	X	X	ES	ET1	EX0	ET0	EX0

EA (enable all):

If EA=0, no interrupt will be acknowledged.

If EA=1, each interrupt source is individually enabled or disabled by setting the enable bit.

ES - Enables or disables Serial port interrupt:

If ES=0, serial port interrupt is disabled.

ET1 Enables or disables the Timer 1 overflow interrupt:

If ET1 = 0, Timer 1 overflow interrupt is disabled

EX1 Enables or disables the external interrupt 1

If EX1 =0, external interrupt 1 is disabled

ET0 Enables or disables Timer 0 overflow interrupt

If ET0 = 0, Timer 0 overflow interrupt is disabled

EX0 Enables or disables the external interrupt 0

If EX0 = 0, external interrupt 0 is disabled

External interrupts INTO and INT1

8051 has two external hardware interrupts such as

1. INT0
2. INT1.

External hardware interrupts can be activated by two methods

1. Level triggered
2. Edge triggered.

Level-triggered interrupt

In the level-triggered mode, INT0 and INT1 pins are normally high and 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 the interrupt. This is called a level- triggered or level-activated interrupt. The low-level signal at the INT pin must be removed before the execution of the last instruction of the interrupt service routine, (RETI) otherwise, another interrupt will be generated.

The IT0 and IT1 flag bits that determine level- or edge-triggered mode of the hardware interrupts.

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Fig: TCON Register

During RESET TCON.0 (IT0) and TCON.2 (IT1) are both 0‟s. i.e., hardware interrupts INT0 and INT1 are low level triggered.

Edge-triggered interrupts

To make INT0 and INT1 as edge-triggered interrupts, TCON bits must be programmed. By making the TCON.0 and TCON.2 bits high with instructions such as “SETB TCON. 0″ and “SETB TCON. 2″, the external hardware interrupts of INT0 and INT1 become edge-triggered.

For example, the instruction “SETB CON. 2″ makes INT1 as edge-triggered interrupt, in which, a high-to-low signal is applied to pin P3.3 (INT1).

Timer Interrupts

8051 has two interrupts. They are

1) Timer 0 interrupt (T0)
2) Timer 1 interrupt (T1)

Whenever the timer rolls over, TF is raised. The timer interrupt in Interrupt Enable (IE) register is enabled and the microcontroller is interrupted and jumps to the interrupt vector table to service the interrupt service routine (ISR) which is located at $000B_H$.

Serial Communication Interrupt

The serial communication interrupt in 8051 are Transmit Interrupt (TI) and Receive Interrupt (RI).

Transmit interrupt (TI) is raised when the last bit of the framed data i.e., stop bit is transferred Receive Interrupt (RI) is raised when the entire frame of data including stop bit is received.

Interrupt Priority in 8051

When the 8051 is powered up, the priorities are assigned

Interrupt	Priority
External interrupt 0(INT0) Timer interrupt 0(TF0) External interrupt 1(INT1) Timer interrupt 1(TF1) Serial communication (RI&TI)	Highest Lowest

Interrupt Priority (IP) Register

D7 D0

-	-	-	PS	PT1	PX1	PT0	PX0

PS - Serial Port Interrupt

Priority level PT1 - Timer 1

Interrupt Priority level

PX1- External

Interrupt 1 Priority level PT0

- Timer 0 Interrupt Priority

level PX0

- External Interrupt 0 Priority level

Priority bit=1 assigns high priority. Priority bit =0 assigns low priority