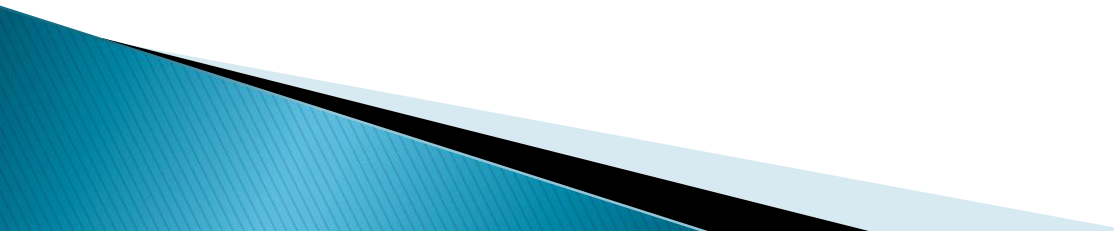


MICROCONTROLLER

UNIT-III

Lecture-6


INTERRUPTS

- ▶ An interrupt is an external or internal event that interrupts the microcontroller to inform it that a device needs its service
 - ▶ A single microcontroller can serve several devices by two ways:
 - ▶ Interrupts: Whenever any device needs its service, the device notifies the microcontroller by sending it an interrupt signal:
- 

Contd.

- ▶ Upon receiving an interrupt signal, microcontroller interrupts whatever it is doing and serves the device

Six Interrupts in 8051

- ▶ Six interrupts are allocated as follows:
 - ▶ 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 that belongs to both receive and transfer
- 

Contd.

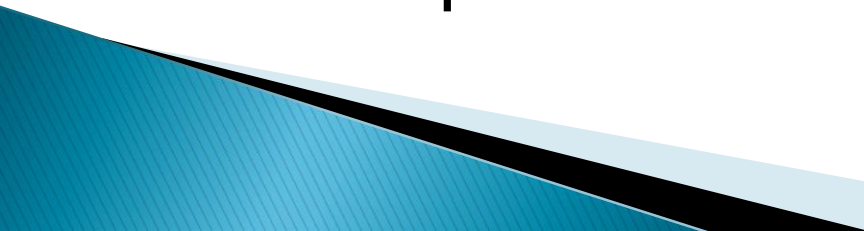
Interrupt vector table

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	

```
ORG 0      ;wake-up ROM reset location
LJMP MAIN  ;by-pass int. vector table
;---- the wake-up program
ORG 30H
MAIN:
    . . . .
    END
```

Only three bytes of ROM space assigned to the reset pin. We put the LJMP as the first instruction and redirect the processor away from the interrupt vector table.

Enabling and Disabling an Interrupt

- ▶ 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 in order for the microcontroller to respond to them
 - ▶ There is a register called IE (interrupt enable) that is responsible for enabling (unmasking) and disabling (masking) the interrupts
- 

Contd.

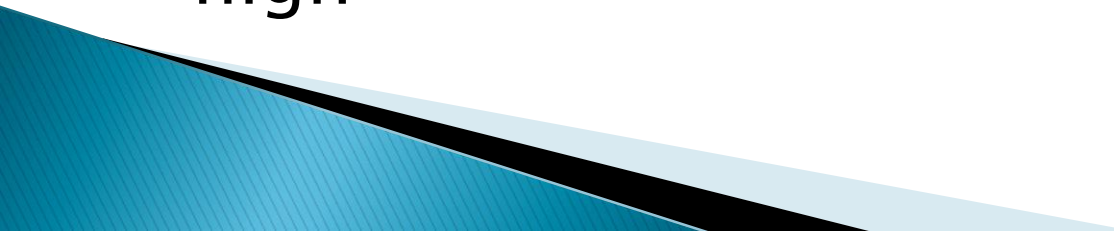
IE (Interrupt Enable) Register



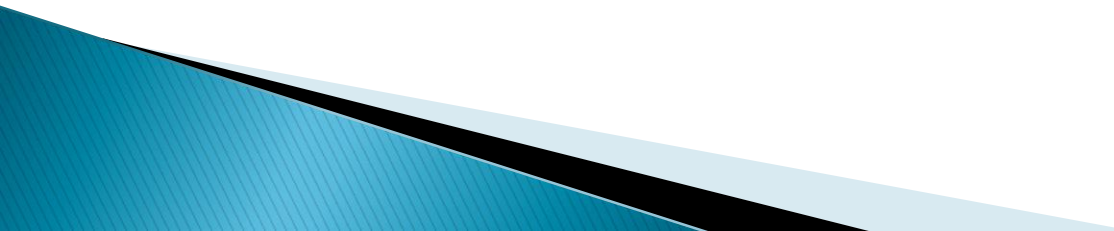
EA (enable all) must be set to 1 in order for rest of the register to take effect

EA	IE.7	Disables all interrupts
--	IE.6	Not implemented, reserved for future use
ET2	IE.5	Enables or disables timer 2 overflow or capture interrupt (8952)
ES	IE.4	Enables or disables the serial port interrupt
ET1	IE.3	Enables or disables timer 1 overflow interrupt
EX1	IE.2	Enables or disables external interrupt 1
ET0	IE.1	Enables or disables timer 0 overflow interrupt
EX0	IE.0	Enables or disables external interrupt 0

Contd.

- ▶ To enable an interrupt, we take the following steps: (1.) Bit D7 of the IE register (EA) must be set to high to allow the rest of register to take effect
 - (2.) The value of EA: If $EA = 1$, interrupts are enabled and will be responded to if their corresponding bits in IE are high
 - ▶ If $EA = 0$, no interrupt will be responded to, even if the associated bit in the IE register is high
- 

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
- 

Contd.

- ▶ Using interrupts solves this problem and, avoids tying down the controller:
- ▶ If timer interrupt in IE register is enabled, whenever timer rolls over, TF is raised, and microcontroller is interrupted in whatever it is doing, and jumps to interrupt vector table to service the ISR