8051 Interrupts

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:

1. **Interrupt**
2. **Polling**

Interrupt Vs Polling

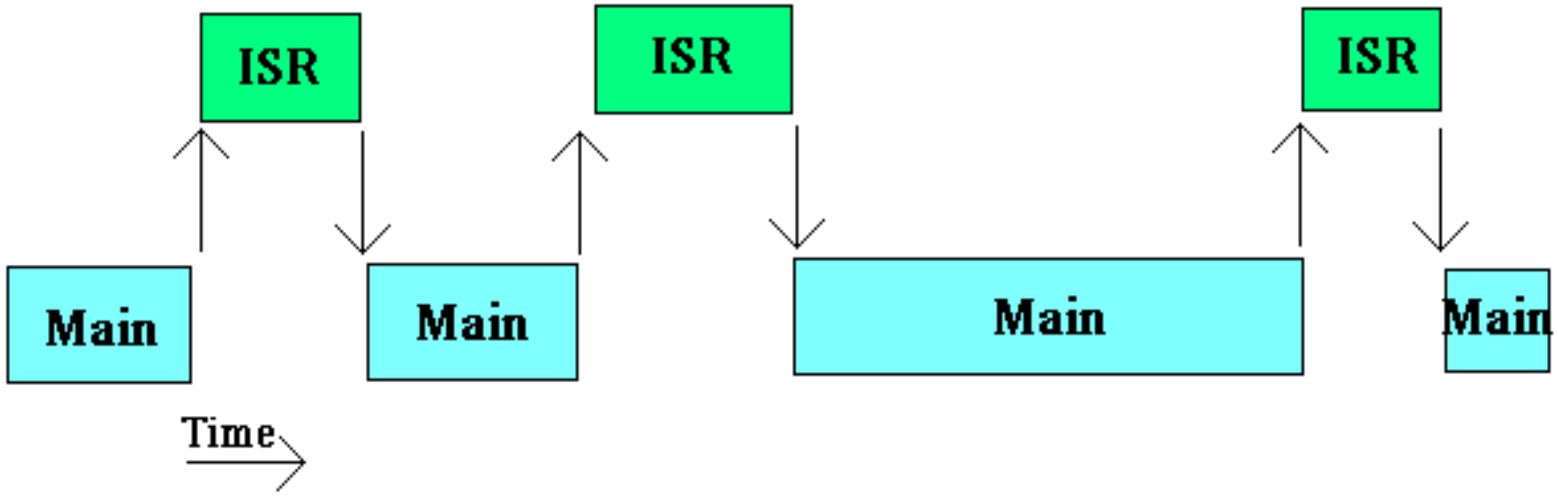
1. **Interrupts**
   - Whenever any device needs its service, the device notifies the microcontroller by sending it an **interrupt signal**.
   - Upon receiving an interrupt signal, the **microcontroller interrupts** whatever it is doing and serves the device.
   - The program which is associated with the interrupt is called the **interrupt service routine (ISR)** or interrupt handler.
2. **Polling**
   - The microcontroller **continuously monitors** the status of a given device.
   - When the **conditions** met, it performs the service.
   - After that, it moves on to monitor the **next device** until every one is serviced.

Interrupt Vs Polling

- The **polling method is not efficient**, since it wastes much of the microcontroller's time by polling devices that do not need service.
- The **advantage of interrupts** is that the microcontroller can serve many devices (not all at the same time).
- Each devices can get the attention of the microcontroller based on the **assigned priority**.
- For the polling method, it is **not possible** to assign priority since it checks all devices in a round-robin fashion.
- The microcontroller can also **ignore (mask)** a device request for service in Interrupt.

Program execution without intrrupts :

Program execution with intrrupts :



ISR : Intrrupt Service Routin

Six Interrupts in 8051

Six interrupts are allocated as follows:

1. **Reset – power-up reset.**

2. **Two interrupts are set aside for the timers.**
   - one for timer 0 and one for timer 1

3. **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)

4. **Serial communication has a single interrupt that belongs to both receive and transfer.**

What events can trigger Interrupts?

- We can configure the 8051 so that any of the following events will cause an interrupt:
  - Timer 0 Overflow.
  - Timer 1 Overflow.
  - Reception/Transmission of Serial Character.
  - External Event 0.
  - External Event 1.
- We can configure the 8051 so that when Timer 0 Overflows or when a character is sent/received, the appropriate interrupt handler routines are called.

8051 Interrupt Vectors

INTERRUPT VECTORS

When the original 8051 and 8031 were introduced, only 5 interrupts were provided.

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

8051 Interrupt related Registers

- The various registers associated with the use of interrupts are:
  - TCON - Edge and Type bits for External Interrupts 0/1
  - SCON - RI and TI interrupt flags for RS232
  - IE - Enable interrupt sources
  - IP - Specify priority of interrupts

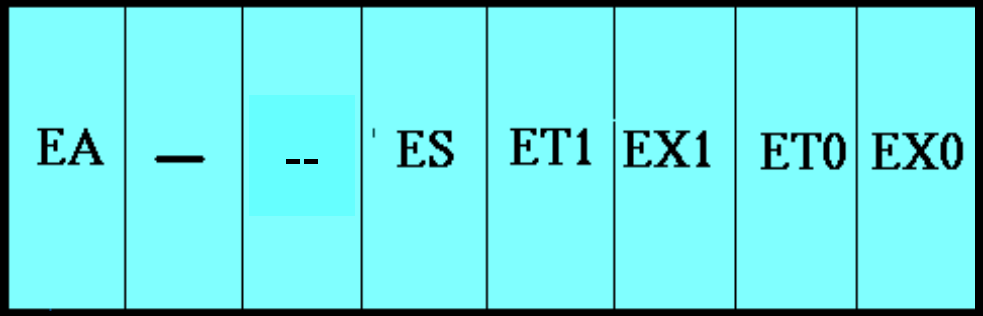
Enabling and Disabling an Interrupt

- 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.

Interrupt Enable (IE) Register



- EA : Global enable/disable.
- --- : Reserved for additional interrupt hardware.
- ES : Enable Serial port interrupt.
- ET1 : Enable Timer 1 control bit.
- EX1 : Enable External 1 interrupt.
- ET0 : Enable Timer 0 control bit.
- EX0 : Enable External 0 interrupt.

MOV IE,#08h
or
SETB ET1

Enabling and Disabling an Interrupt

- **Example:** Show the instructions to **(a)** enable the serial interrupt, timer 0 interrupt, and external hardware interrupt 1 and **(b)** disable (mask) the timer 0 interrupt, then **(c)** show how to disable all the interrupts with a single instruction.
- **Solution:**
  - **(a) MOV IE,#10010110B** ;enable serial, timer 0, EX1
    - Another way to perform the same manipulation is:
      - **SETB IE.7** ;EA=1, global enable
      - **SETB IE.4** ;enable serial interrupt
      - **SETB IE.1** ;enable Timer 0 interrupt
      - **SETB IE.2** ;enable EX1
  - **(b) CLR IE.1** ;mask (disable) timer 0 interrupt only
  - **(c) CLR IE.7** ;disable all interrupts

Interrupt Priority

- When the 8051 is powered up, the priorities are assigned according to the following.

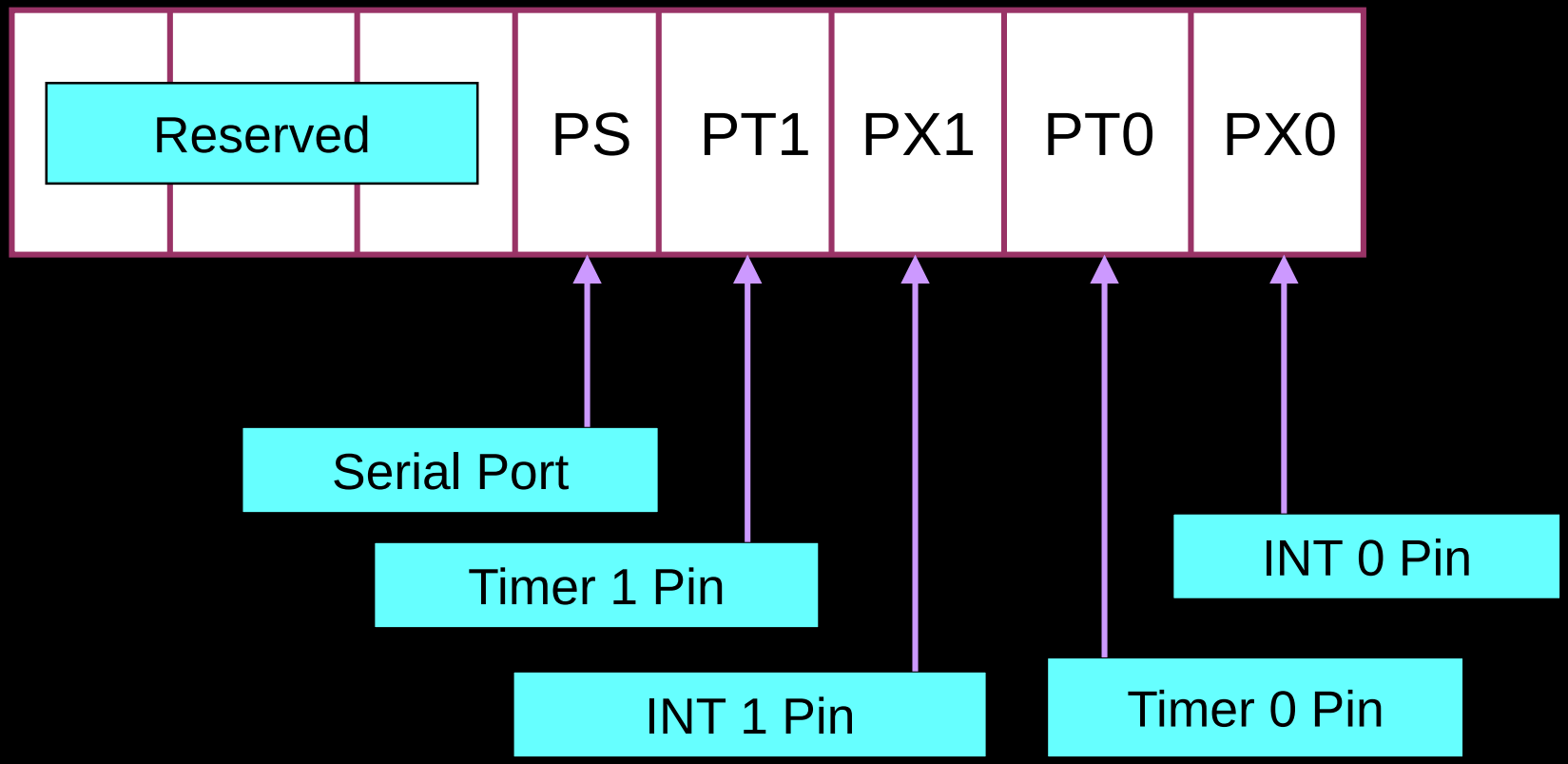
- In reality, the priority scheme is nothing but an internal polling sequence in which the 8051 polls the interrupts in the sequence listed and responds accordingly.

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

- **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.**

Interrupt Priority (IP) Register



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