**Microcontrollers - 8051 Interrupts:**

There are two main types of interrupts:

1. **Hardware Interrupts** – Triggered by external signals or peripheral events.
2. **Software Interrupts** – Initiated by specific program instructions to execute predefined routines.

* Each interrupt has an associated vector address where the ISR (Interrupt Service Routine) is stored.
* When an interrupt occurs, the microcontroller saves its current state, executes the ISR, and then resumes the interrupted task.

**8051 Microcontroller Interrupt:**

In the 8051 microcontroller, **timer and serial interrupts** are internally generated, while **external interrupts** come from peripheral devices or switches. External interrupts are classified into:

1. **Edge-triggered interrupts** – Activated by a change in signal edge (rising or falling).
2. **Level-triggered interrupts** – Triggered when the signal remains at a specific level (high or low).

**Interrupt Handling:**

Two Special Function Registers (SFRs) control interrupt operations:

1. **IE (Interrupt Enable Register)** – Enables or disables specific interrupts.
2. **IP (Interrupt Priority Register)** – Assigns priority levels to interrupts.

The **priority system** consists of three levels:

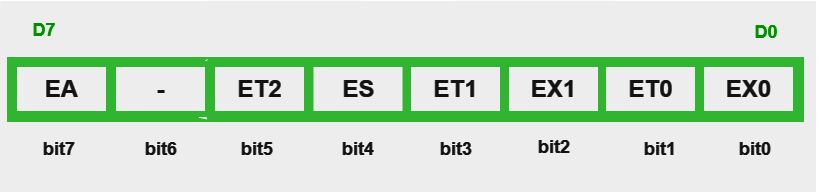
* **Reset**: Stops all operations and restarts the microcontroller.
* **Priority 1**: Can be disabled only by a reset.
* **Priority 0**: Can be disabled by both a reset and another interrupt.

**IE (Interrupt Enable) Register in 8051 Microcontroller:**

* The **IE register** is responsible for enabling or disabling interrupts in the **8051 microcontroller**.
* It consists of multiple bits that control different interrupts:

**Key Bits in the IE Register:**

1. **EA (Global Interrupt Enable/Disable):**
   * 1 → Enables all interrupts.
   * 0 → Disables all interrupts.
2. **ES (Serial Communication Interrupt Enable):**
   * 1 → Enables UART system interrupt.
   * 0 → Disables UART system interrupt.
3. **ET0 (Timer 0 Interrupt Enable):**
   * 1 → Enables Timer 0 interrupt.
   * 0 → Disables Timer 0 interrupt.
4. **ET1 (Timer 1 Interrupt Enable):**
   * 1 → Enables Timer 1 interrupt.
   * 0 → Disables Timer 1 interrupt.
5. **EX0 (External Interrupt 0 Enable):**
   * 1 → Enables External Interrupt 0.
   * 0 → Disables External Interrupt 0.
6. **EX1 (External Interrupt 1 Enable):**
   * 1 → Enables External Interrupt 1.
   * 0 → Disables External Interrupt 1.
7. **IT0 and IT1 (Interrupt Type Control for EX0 and EX1):**
   * **Determines trigger type** for external interrupts (edge or level-triggered).



**IP (Interrupt Priority) Register in 8051 Microcontroller:**

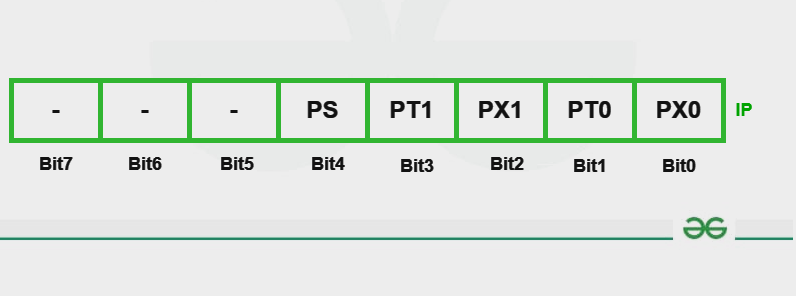
The **IP register** manages the priority of interrupts in the **8051 microcontroller**. Since multiple interrupts can occur simultaneously, the IP register helps determine which interrupt should be serviced first.

**Key Features of the IP Register:**

* **Higher-priority interrupts** can **pause lower-priority ones** if they occur simultaneously.
* **Reset (RST)** has the highest priority and **disables all interrupts** when triggered.
* **Interrupt Priority 1 (High)** can only be disabled by **Reset**.
* **Interrupt Priority 0 (Low)** can be disabled by **both Reset and Priority 1 interrupts**.
* If two interrupts of **equal priority** occur, the second one must **wait** until the first is completed.

**Bit Assignments in the IP Register:**

| **Bit** | **Function** | **0 (Low Priority)** | **1 (High Priority)** |
| --- | --- | --- | --- |
| **PX0** | External Interrupt 0 Priority | Low | High |
| **PT0** | Timer 0 Interrupt Priority | Low | High |
| **PX1** | External Interrupt 1 Priority | Low | High |
| **PT1** | Timer 1 Interrupt Priority | Low | High |
| **PS** | Serial Communication Priority | Low | High |
| **Bits 5, 6, 7** | Reserved Bits | - | - |



**Types of 8051 Microcontroller Interrupts:**

The **8051 microcontroller** has **five different types of interrupts** that can temporarily halt the main program execution to handle specific tasks. These interrupts include **external hardware interrupts, timer interrupts, and serial communication interrupts**.

**List of Interrupts in 8051:**

| **Interrupt Type** | **Interrupt Flag** | **Description** |
| --- | --- | --- |
| **Timer 0 Overflow** | TF0 | Triggered when Timer 0 overflows. |
| **Timer 1 Overflow** | TF1 | Triggered when Timer 1 overflows. |
| **External Interrupt 0** | INT0 | Triggered by an external signal at PORT3.2. |
| **External Interrupt 1** | INT1 | Triggered by an external signal at PORT3.3. |
| **Serial Communication** | RI/TI | Triggered by UART transmission/reception. |

**1. External Hardware Interrupts (INT0 & INT1)**

The **8051 microcontroller** can respond to **external events** through two external interrupts: **INT0** and **INT1**.

**External Interrupt 0 (INT0)**

* Connected to **PORT3.2**.
* Triggered by a **low-to-high** transition at the pin.
* The **IE (Interrupt Enable)** register must be set to enable INT0.
* The **IT0 (Interrupt Type 0)** bit in the **TCON** register determines **edge-triggered or level-triggered mode**.

**External Interrupt 1 (INT1)**

* Connected to **PORT3.3**.
* Triggered by a **low-to-high** transition at the pin.
* The **IE (Interrupt Enable)** register must be set to enable INT1.
* The **IT1 (Interrupt Type 1)** bit in the **TCON** register determines **edge-triggered or level-triggered mode**.

**2. Timer Interrupts (Timer0 and Timer1)**

**Timers** are **hardware counters** used to measure time intervals and generate precise delays.

**Timer 0 Interrupt (TF0)**

* **8-bit timer** with a count range of **0 to 255**.
* Operates in **13-bit or 16-bit mode**.
* Uses **TH0 (Timer 0 High) and TL0 (Timer 0 Low)** registers in **13-bit mode**.
* Generates an **interrupt when it overflows**.
* The microcontroller executes a specific **ISR (Interrupt Service Routine)** upon overflow.

**Timer 1 Interrupt (TF1)**

* **16-bit timer** with a count range of **0 to 65,535**.
* Operates in **16-bit or 8-bit mode**.
* Uses **TL1 (Timer 1 Low) and TH1 (Timer 1 High)** registers in **8-bit mode**.
* Generates an **interrupt when it overflows**, triggering an ISR.

**3. Serial Communication Interrupt (UART - RI/TI)**

The **8051 microcontroller** uses **UART (Universal Asynchronous Receiver/Transmitter)** for serial communication.

**Steps in UART Interrupt Handling:**

1. **Initialize UART**: Set the **data format**, **baud rate**, and **enable UART module** using configuration registers.
2. **Enable Interrupts**: Enable the **Transmit Interrupt (TI)** for data transmission. Enable the **Receive Interrupt (RI)** for data reception.
3. **Write an Interrupt Service Routine (ISR)**: Check if data has been **received (RI)** or **transmitted (TI)**.
4. **Clear the Interrupt Flag**: Reset **RI or TI** to acknowledge the interrupt and prepare for the next one.