

Embedded Systems summary

◆ Why Use Interrupts?

- **Efficiency:** Processor can sleep or perform other tasks.
- **Responsiveness:** Reacts immediately when the event occurs.
- **Power Saving:** Reduces power usage since CPU is idle until an interrupt occurs.

◆ Interrupt vs Polling:

Feature	Polling	Interrupts
CPU Behavior	Constantly checks flag	Waits until interrupt occurs
Power Consumption	High	Low
Efficiency	Less efficient	More efficient
Reaction Time	Depends on polling frequency	Immediate

◆ Interrupt Service Routine (ISR):

- A function that runs when an interrupt is triggered.
- Must be short and fast to avoid missing other interrupts.
- Usually disables other interrupts temporarily.

```
void __interrupt() myISR(void) {  
    if (INTF == 1) {  
        // Handle external interrupt  
        INTF = 0; // Clear interrupt flag  
    }  
}
```

◆ Interrupt Vector Table:

- A table that maps each interrupt to its ISR address.
 - In PIC microcontrollers, certain memory addresses are reserved for ISR.
-

◆ Interrupt Priorities:

- Some microcontrollers allow priority settings (high and low priority).
 - Ensures that important interrupts are handled first.
-

◆ Types of Interrupts:

- **External Interrupts:** Triggered by hardware signals on specific pins (e.g., INT0).
 - **Timer Interrupts:** Triggered when timers overflow or match a value.
 - **Peripheral Interrupts:** From UART, ADC, I2C, etc.
-

◆ Enabling Interrupts:

- Use control registers (e.g., INTCON, PIE1, PIR1) to enable/disable specific interrupts.
 - **GIE:** Global Interrupt Enable
 - **PEIE:** Peripheral Interrupt Enable
-

◆ Applications of Interrupts:

- Real-time systems
 - Measuring time between events
 - Communication protocols
 - Low-power embedded systems
-

📄 Example Use Case:

- **Button Press Detection:**
 - External interrupt triggered on falling edge.
 - ISR toggles LED state.

```

void __interrupt() ISR(void) {
    if (INTF) {
        LED = !LED; // Toggle LED
        INTF = 0;   // Clear interrupt flag
    }
}

```

✓ Key Takeaways:

- Interrupts improve efficiency and responsiveness.
 - ISRs should be concise and optimized.
 - Interrupts can coexist with polling if managed properly.
 - Must configure interrupt sources and flags carefully to avoid unexpected behavior.
-

Feature	Microcontroller	Microprocessor
Purpose	Dedicated application	General-purpose processing
Components Included	CPU, RAM, ROM, I/O ports, timers all in one chip	Only CPU, requires external peripherals
Power Consumption	Low	Higher
Cost	Lower	Higher
Complexity	Less complex system design	More complex due to many external parts
Applications	Embedded devices (e.g., washing machines, IoT)	PCs, laptops, servers

1) System on Chip (SoC):

- a) SoC integrates all components of a computer or other electronic system into a single chip.
- b) Commonly used in mobile devices.

c) Considered an evolution of microcontrollers with more power and integrated features.

4. Harvard vs Von Neumann Architecture:

Aspect	Von Neumann Architecture	Harvard Architecture
Memory	Single memory for program and data	Separate memory for program and data
Speed	Slower due to shared bus	Faster as it can access program & data simultaneously
Complexity	Less complex	More complex due to dual buses

✓ Advantages of Microcontrollers:

- Cost-effective for mass production.
- Lower power consumption.
- Easy to develop for simple, real-time control tasks.
- Compact and integrated.

✗ Limitations of Microcontrollers:

- Limited processing power.
- Less memory and storage.
- Not suitable for complex computations or multitasking applications.

Interrupt Types Overview


Interrupt Type

Description

Hardware Interrupts

Triggered by external devices or events (e.g., button press, sensor signal).

Interrupt Type	Description
Software Interrupts	Triggered by executing a specific instruction in code.

 Interrupt Characteristics	
Characteristic	Description
Speed	Immediate response to time-sensitive events.
Priority Levels	Allows more critical interrupts to preempt lower-priority ones.

Interrupt Vector Table	Lookup table that associates interrupts with handler routines.
------------------------	--

Key Concepts and Flow

1. **Global Interrupt Enable (GIE)** must be set for any interrupt to occur.
 2. Individual interrupts must also be enabled.
 3. When an interrupt occurs:
 - MCU stops main execution.
 - Jumps to the appropriate ISR (Interrupt Service Routine).
 - Executes the ISR.
 - Returns to the main program.
-

Setup in Registers

Register

Purpose

INTCON Main control for global and peripheral interrupts.

PIE1, PIE2 Enable bits for specific interrupts.

PIR1, PIR2 Flags indicating which interrupt occurred.

Best Practices

- Keep ISRs short and efficient.
- Clear interrupt flags inside ISR.
- Avoid long delays inside ISRs.

Common Applications

- Handling button presses
- Monitoring sensors
- Timers and delays
- Serial communication handling