Summarization – Task 12

1. MCU Clock System

- The Clock System is the basis for running any Microcontroller.

- It generates a Clock Signal that controls the execution speed of instructions.

- Clock Sources:

- Internal Oscillator (RC Oscillator) → low-cost but less accurate.

- External Crystal Oscillator → more accurate and stable.

- PLL (Phase Locked Loop) can be used to multiply the clock frequency for higher performance.

- Clock configuration is controlled through MCU registers to select the source and distribute frequency to CPU, peripherals, and timers.

2. Interrupt Fundamentals & Architecture

- An Interrupt is a mechanism that allows the CPU to temporarily stop the current program and respond to an urgent event.

- Purpose: reduce CPU polling and improve efficiency.

- Types of interrupts:

- Hardware Interrupts → triggered by external devices (button, timer, UART…).

- Software Interrupts → generated by software (Exception or SWI).

- Interrupt Architecture Components:

- Interrupt Vector Table (IVT): stores the addresses of ISR routines.

- Interrupt Controller: manages priorities and routes interrupts to CPU.

- Mask/Enable Registers: allow enabling or disabling specific interrupts.

3. Interrupt Handling & Startup Process

- Steps of interrupt handling:

1. An external event occurs (e.g., button press).

2. Interrupt controller sends a signal to the CPU.

3. CPU halts current execution and saves Program Counter + some registers.

4. CPU jumps to the address in IVT and executes the ISR (Interrupt Service Routine).

5. After ISR completes, context is restored and main program continues.

- Startup Process:

- After reset, IVT is loaded from memory.

- Stack Pointer is initialized.

- Global Interrupt Enable bit is set (using special instruction like SEI or a control register).

4. Software Mechanisms & Concepts for Managing Interrupts

- ISR Design Rules:

- Must be short and fast.

- Avoid complex functions or long loops.

- Priority & Nesting:

- Some MCUs support priority levels or nested interrupts (higher priority ISR can interrupt a lower one).

- Debouncing: necessary for buttons to avoid multiple false triggers.

- Critical Sections:

- When shared data is accessed by both main code and ISR, interrupts should be temporarily disabled.

- Software Flags:

- ISR should set a flag, and the main loop should process it to reduce ISR execution time.