TASK 12

🟦 Microcontroller Systems: Clock & Interrupts

MCU Clock System

- General Topology of Clock Architecture

MCU clocks are crucial timing signals within a microcontroller, ensuring that all components operate in sync and at the correct timing. The Clock Management Unit (CMU) is responsible for managing these clocks, including their sources, distribution, and scaling.

Types of MCU Clocks:

- **System Clock (Core Clock):** Drives the CPU and affects the overall processing speed. For example, a 48 MHz system clock influences the CPU's execution speed.

- **Peripheral Clocks:** Provide timing for peripherals like timers, UARTs, and ADCs. These can be derived from the system clock or generated independently.

- **Real-Time Clock (RTC):** Maintains accurate time and date even during low-power states. It operates with its own oscillator.

Clock Sources:

- **Internal RC Oscillator (HSI/IRC) → quick startup, less accurate.**
- **External Crystal Oscillator (HSE) → stable, accurate, but needs external component.**
- **Low-Speed Oscillators (LSI/LSE) → used for RTC, watchdog timers.**
- **PLL (Phase-Locked Loop)* → multiplies/divides frequency for higher system clock speeds.**

Clock Distribution:

- **Clock Tree:** Distributes the clock signal to various MCU parts, such as the CPU and peripherals.

📌 In STM32 (ARM Cortex-M) you'll see this "Clock Tree" diagram in datasheets.

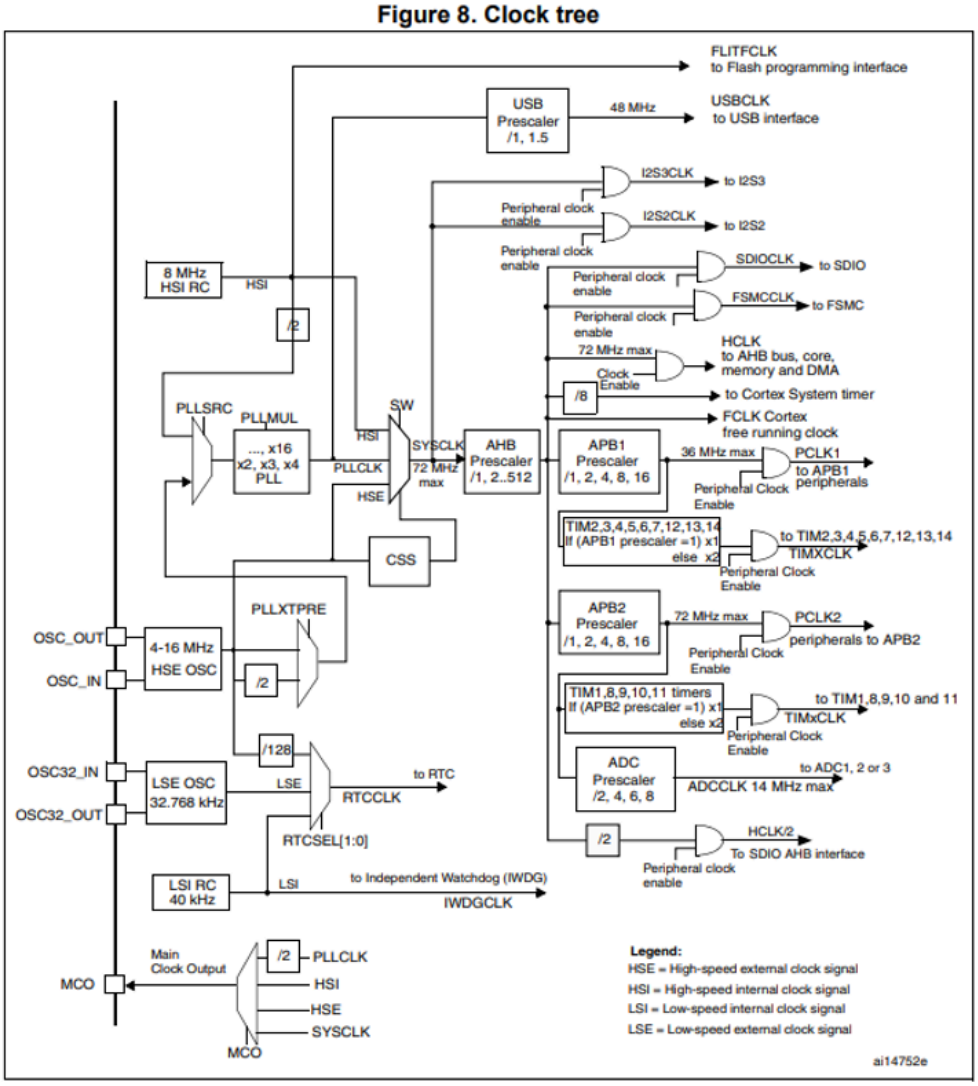
📌 In ATmega (AVR 8-bit) → clock is much simpler (internal RC or external crystal, no complex PLL tree).

Steps to Find Out the Clock Tree in an MCU:

1. **Examine the Clock Management Unit in the Block Diagram:** Understand the layout and connections.
2. **Check the Base Addresses of the Clock Management Unit:** Locate the registers for clock configuration.
3. **Consult the Technical Reference Manual (TRM):** Review the clock tree structure and configuration options.

STM32F103xx_TRM

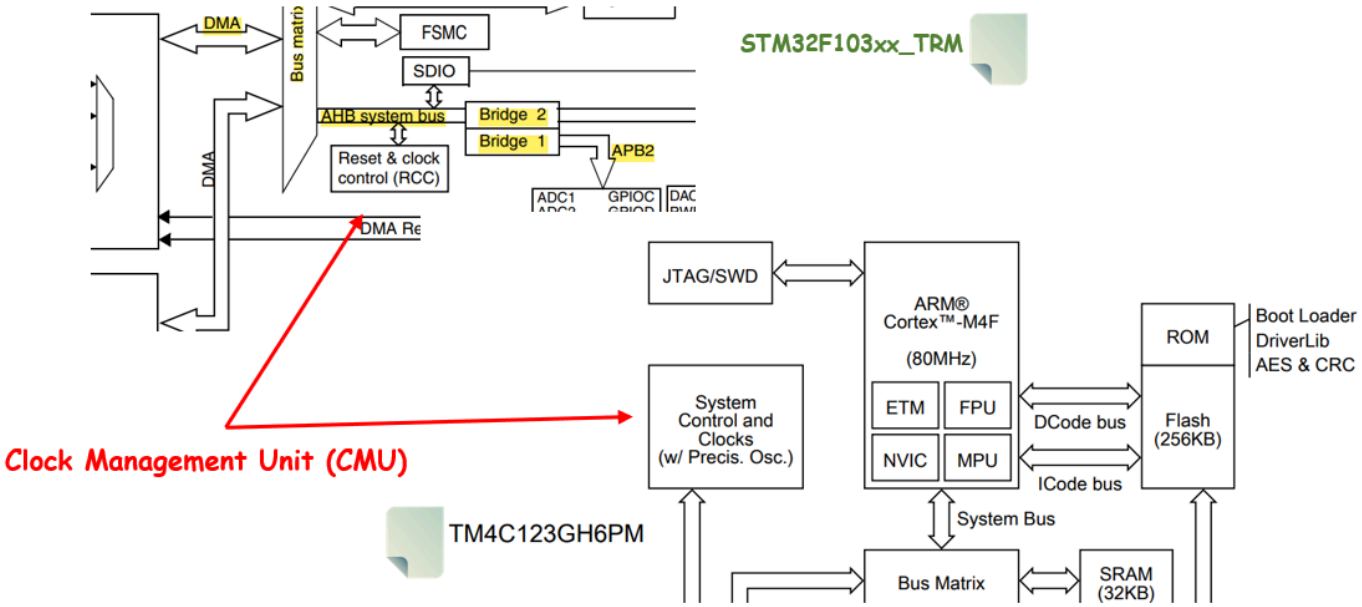
Figure 8. Clock tree



Clock Management Unit (CMU) Components:

- **Clock Sources:** Internal RC oscillators, external crystals, and PLLs.

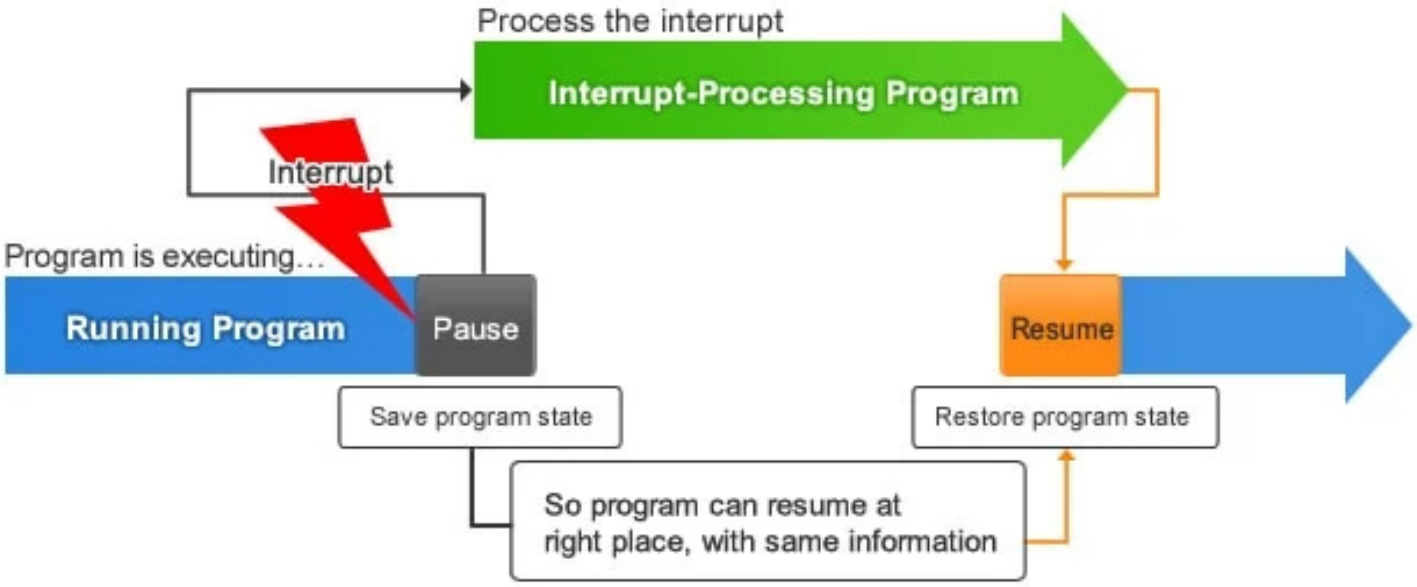
- **Clock Dividers:** Generate different frequencies from a single source.
- **Multiplexers (MUX):** Switch between different clock sources.
- **Clock Gating Units:** Control clock signals to various modules.
- **Clock Monitoring:** Includes mechanisms for monitoring clock integrity and switching to backup sources if needed.



MCU Interrupts

What is an Interrupt?

The interrupt is a signal emitted by hardware or software when a process or an event needs immediate attention. It alerts the processor to a high-priority process requiring interruption of the current working process. In I/O devices one of the bus control lines is dedicated for this purpose and is called the ***Interrupt Service Routine (ISR)***.



- **Types of Interrupts**

Hardware Interrupts → triggered by peripherals (timer overflow, UART receive, GPIO change, ADC done).

Software Interrupts (Exceptions) → triggered by software instructions (like SVC in ARM Cortex).

Maskable (IRQ) → can be enabled/disabled by software.

Non-Maskable Interrupt (NMI) → cannot be disabled, used for critical faults.

- Interrupt Vector Table (IVT) .

A table stored in memory that holds the addresses of Interrupt Service Routines (ISR).

- **Interrupt Handling & Startup Process**

When an interrupt occurs, the MCU follows a standard process:

1. Event Triggered:

A hardware event (e.g., Timer overflow) sets an *interrupt flag* in the peripheral.

2. Interrupt Request (IRQ) Sent:

The peripheral signals the interrupt controller.

3. Priority Check:

NVIC (or equivalent) checks if the interrupt is enabled and if it has the highest priority among pending interrupts.

4. Context Saving (Automatic by Hardware):

CPU pushes Program Counter (PC), Program Status Register (PSR) , and sometimes general-purpose registers onto the stack.

Ensures the main program can resume later.

5. Vector Fetch:

CPU jumps to the ISR address from the interrupt vector table.

6. ISR Execution:

The CPU executes the *Interrupt Service Routine* (user-defined function).

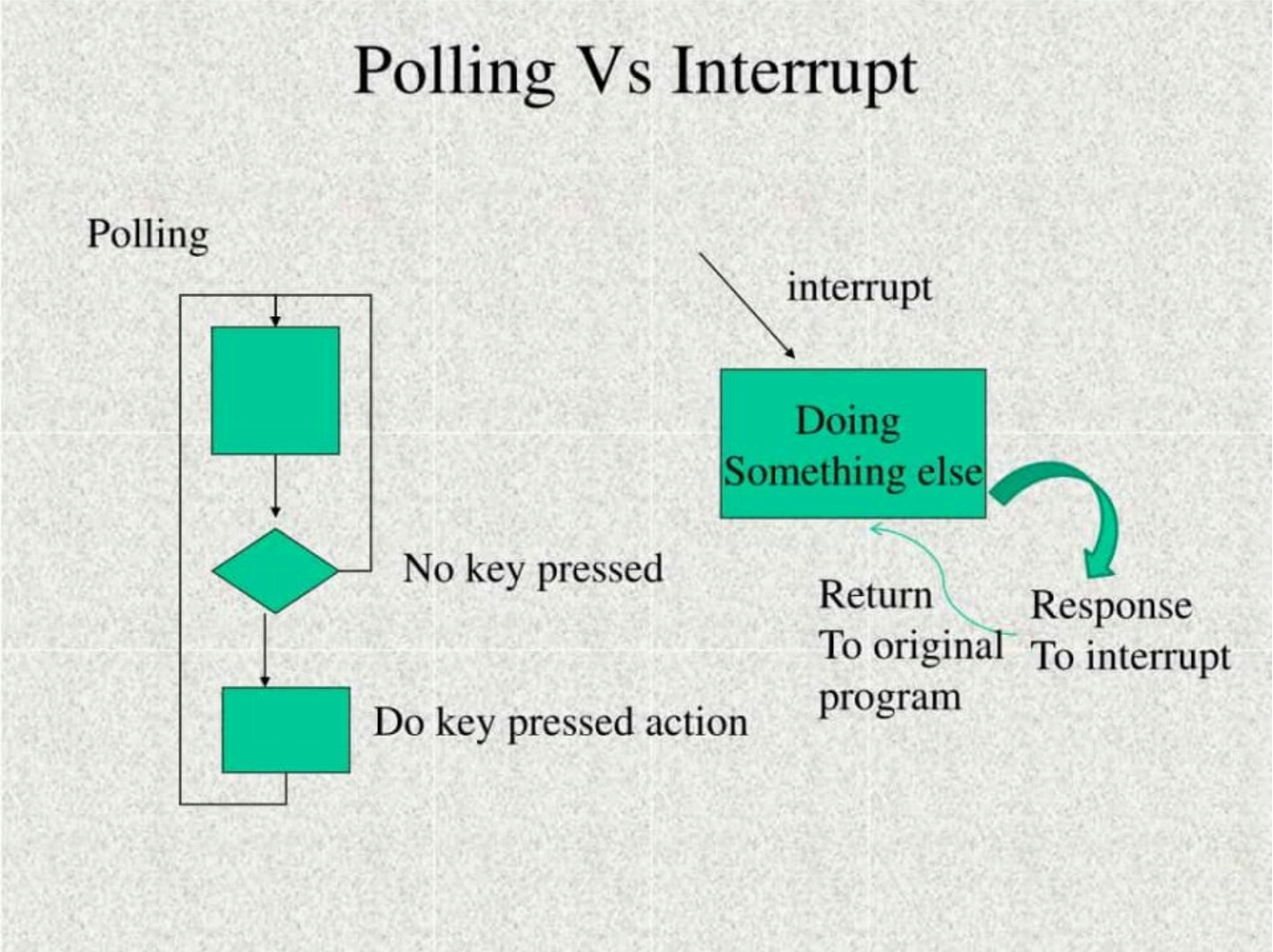
7. Interrupt Return:

CPU pops saved registers from the stack.

Execution resumes from where it was interrupted.

- Polling

Polling is a method where the CPU continuously checks the status of a device or flag in a loop. This means the processor remains active, repeatedly querying peripherals for data or status updates, even when nothing has changed.



- **Determinism and Responsiveness**

These are real-time system concepts:

Determinism → The system behaves predictably; events occur at well-defined times.

Example: A timer ISR always runs every 1 ms, exactly on time.

Responsiveness → How quickly the MCU reacts to an external event (interrupt latency).

Example: Button press → ISR executed within 10 µs.

Trade-off: High responsiveness may reduce determinism if too many interrupts preempt each other.

- **Superloop System**

A program structure where the CPU continuously executes a single infinite loop that checks and runs tasks sequentially. It is simple, deterministic, and predictable but can suffer from poor responsiveness to time-critical events.

- **Foreground–Background System**

A program structure where the background executes in a continuous loop (like the superloop), while the foreground handles urgent events through interrupts. This improves responsiveness and efficiency while still maintaining deterministic background execution.