# Day 1 : Interrupts

SAI KRISHNA

March 2025

## 1 Interrupts Vs Polling

In interrupt mode, when a module needs to be serviced, it's notifies the CPU by sending an interrupt signal. When the CPU receives the signal, the CPU interrupts whatever it is doing and services that module.

In polling mode, the CPU continuously monitors the status of the given module.And when a particular status condition is met, the CPU then services the module.

## 2 What is an interrupt and why is it important in embedded systems?

An interrupt is a signal generated by hardware or software that causes the CPU to pause its current tasks and execute a specialized function called an Interrupt Service Routine (ISR). This mechanism is critical in embedded systems because it enables the processor to respond immediately to time-sensitive events—such as sensor inputs or communication signals—without having to continuously poll for changes. This not only improves system responsiveness but also optimizes power consumption and overall efficiency. Interrupts also allow for prioritization, where critical tasks can preempt less important ones, ensuring that the system meets its real-time performance requirements.

## 3 How do you ensure that an Interrupt Service Routine (ISR) is efficient and doesn't degrade system performance?

To ensure an ISR remains efficient, it should be kept as short and simple as possible.

- Minimize Workload: Perform only essential actions, such as reading or writing data, acknowledging the interrupt, and setting flags for deferred processing.

- Avoid Blocking Operations: Do not include delays, complex computations, or long loops that could prevent the system from servicing other interrupts promptly.
- Shared Resources: Manage shared data carefully by using techniques like volatile variables, atomic operations, or temporarily disabling interrupts when accessing critical sections.
- Context Management: Ensure the processor's context is properly saved and restored to maintain system stability.

# 4 What are nested interrupts and how do they impact system design?

Nested interrupts occur when an ISR is itself interrupted by another, higher-priority interrupt. This feature allows systems to handle more urgent tasks even if the processor is already servicing an interrupt. While nested interrupts can significantly improve responsiveness, they add complexity to system design. The designer must ensure that:

- Context Saving: The system correctly saves the state of the currently executing ISR before switching to a higher-priority task.
- Priority Management: Interrupt priorities are well-defined so that critical tasks always preempt less critical ones without causing resource conflicts or stack overflows.
- System Stability: The overall architecture is robust enough to handle multiple levels of interrupt nesting without causing unpredictable behavior.