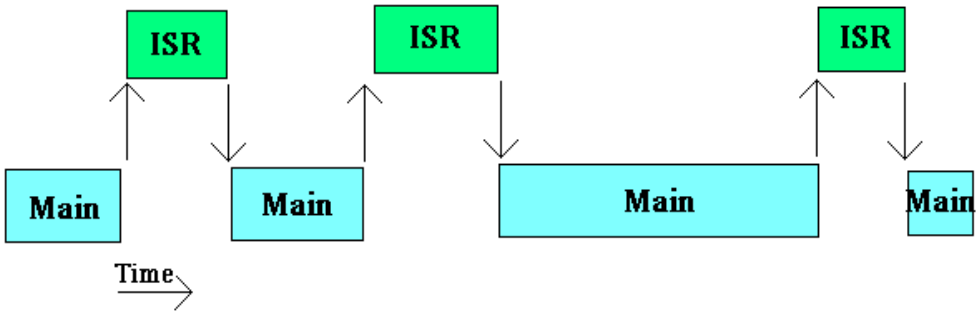
INTERRUPTS AND NESTED INTERRRUPTS HANDLING

- An interrupt is an external or internal event that interrupts the microcontroller to inform it that a device needs its service
- A single microcontroller can serve several devices by two ways:
  1. **Interrupt**
  2. **Polling**

Program execution without intrrupts :

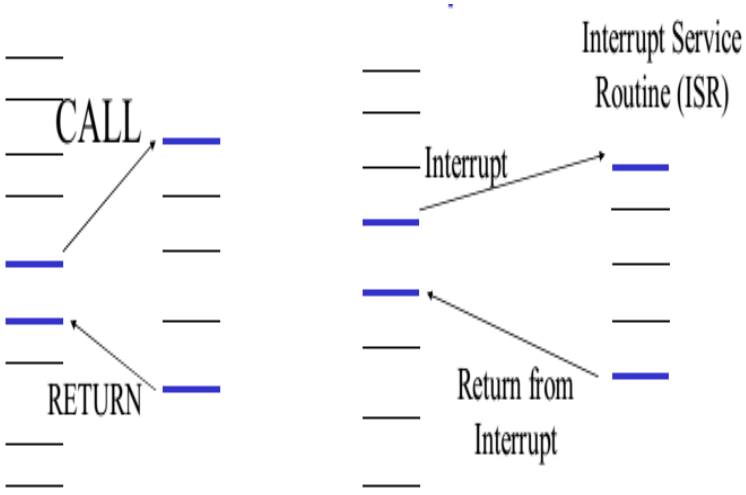
Program execution with intrrupts :



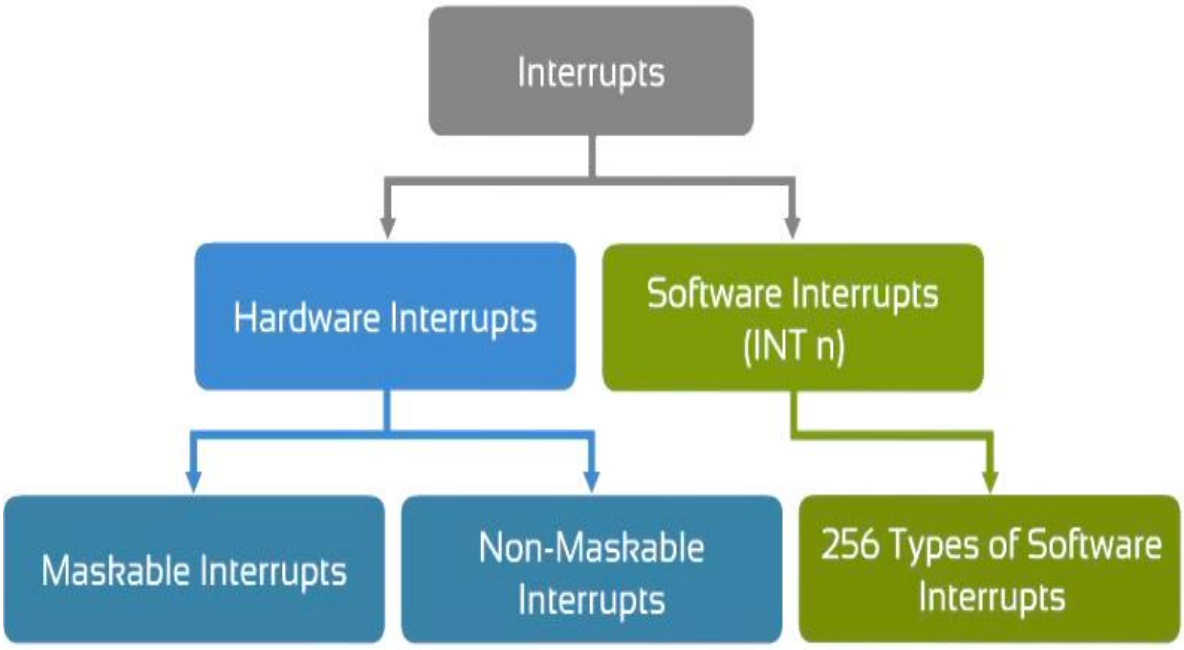
ISR : Intrrupt Service Routin

Steps in executing an interrupt

- Finish current instruction and saves the PC on stack.
- Jumps to a fixed location in memory depend on type of interrupt
- Starts to execute the interrupt service routine until **RETI** (return from interrupt)
- Upon executing the **RETI** the microcontroller returns to the place where it was interrupted. Get pop PC from stack



Types of Interrupts

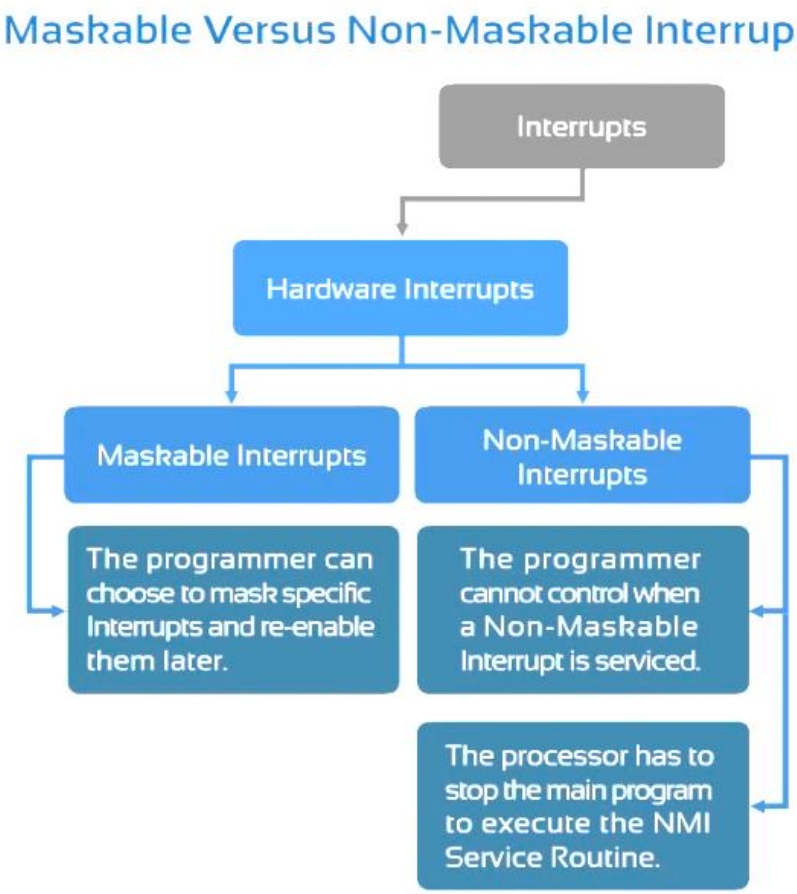


Hardware Interrupts

- Generated by hardware devices.
- Can be External (from outside the chip) or Internal (from on-chip peripherals).

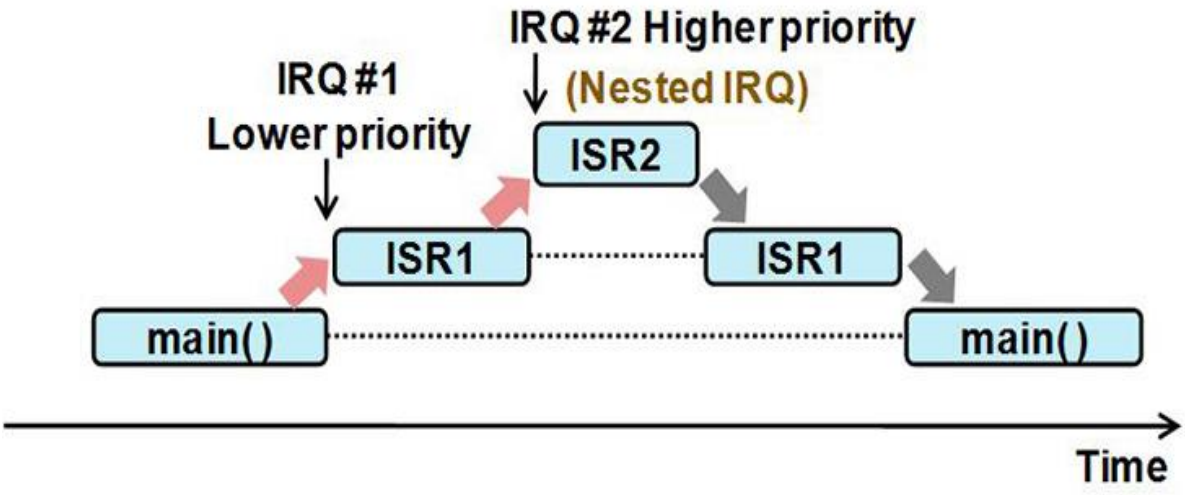
Software Interrupts

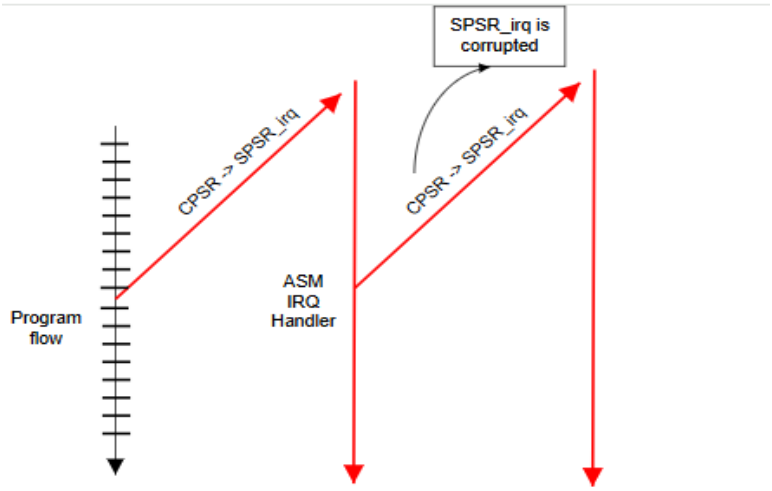
- Generated by software instructions.
- Used for exceptions and operating-system services.



Nested interrupt handling

Nested interrupt handling is where the software is prepared to accept another interrupt, even before it finishes handling the current interrupt. This enables you to prioritize interrupts and make significant improvements to the latency of high priority events at the cost of additional complexity. It is worth noting that nested interrupt handling is a choice made by the software, by virtue of interrupt priority configuration and interrupt control, rather than imposed by hardware.





NVIC: The interrupt controller belongs to the Cortex®-M4 CPU enabling a close coupling with the processor core. The main features are:

• 102 interrupt sources,

• 16 programmable priority levels,

• Low-latency exception and interrupt handling,

• Automatic nesting,

• Power management control.

Interrupt priority determines which interrupt should be serviced first when multiple interrupts occur.

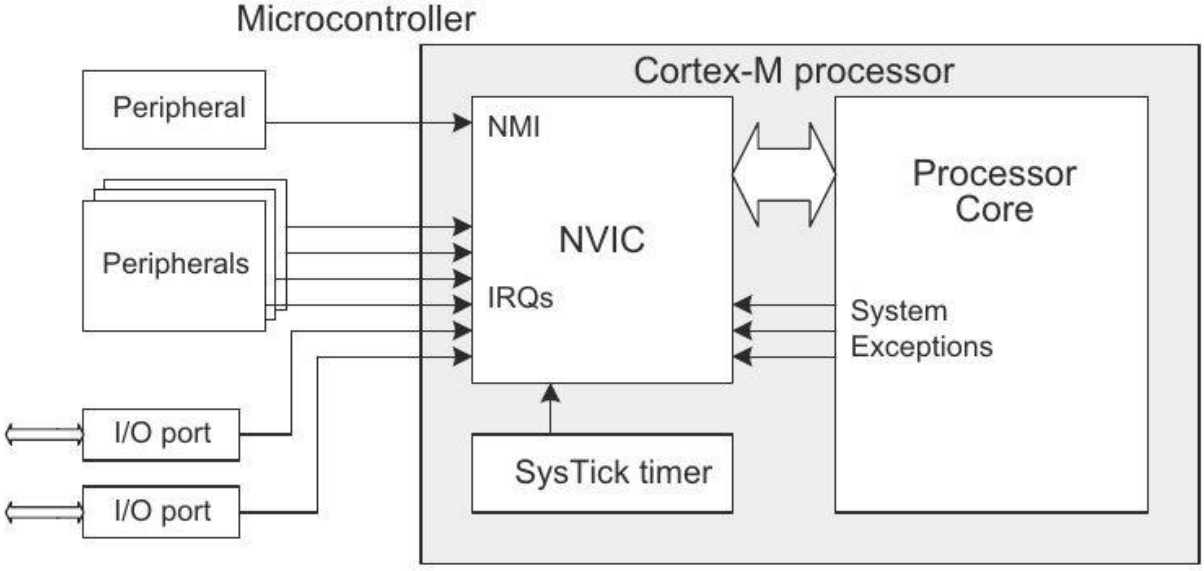
Interrupt Vector Table

The Interrupt Vector Table (IVT) is a table stored in memory that contains the starting addresses of exception and interrupt service routines (ISRs).

When an interrupt occurs, the processor uses the vector table to find which ISR must be executed.

The sequence is:

Peripheral generates → NVIC receives → NVIC checks priority → Vector table identifies ISR → Cortex-M executes ISR → processor returns to the main program.



The Cortex-M processor contains an NVIC that manages interrupt requests from various peripherals and I/O ports. When a peripheral generates an interrupt, the request is sent to the NVIC. The NVIC checks whether the interrupt is enabled and determines its priority. The highest-priority pending interrupt is selected, and the processor core is directed to the corresponding interrupt service routine using the interrupt vector mechanism. The processor temporarily suspends the main program, executes the ISR, and then returns to the previously interrupted program. The SysTick timer can also generate periodic interrupts through the interrupt system, while NMI and system exceptions provide special interrupt/exception mechanisms for critical system events.