مرکز آموزش نیـــرا سیستم

nirasystem.com

Interrupt

Ali Mirghasemi

Process Methods

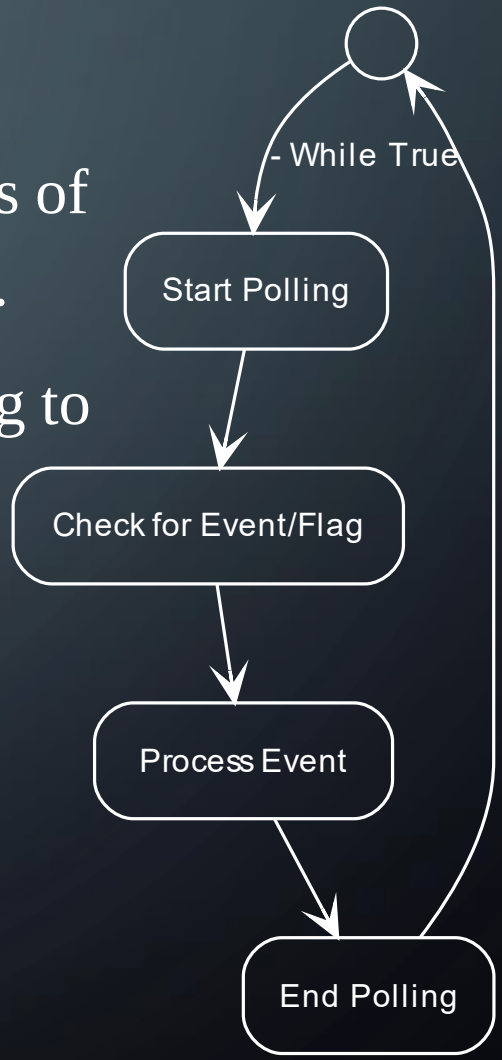
- Polling
- **Interrupt**
- Event Driven

Interrupt Introduction

- Interrupts are an essential mechanism in microcontrollers like the STM32F407.
- They allow the microcontroller to respond promptly to external events without continuously polling for changes.
- When an interrupt occurs, the microcontroller temporarily suspends its current task, executes the interrupt service routine (ISR), and then resumes its previous task.

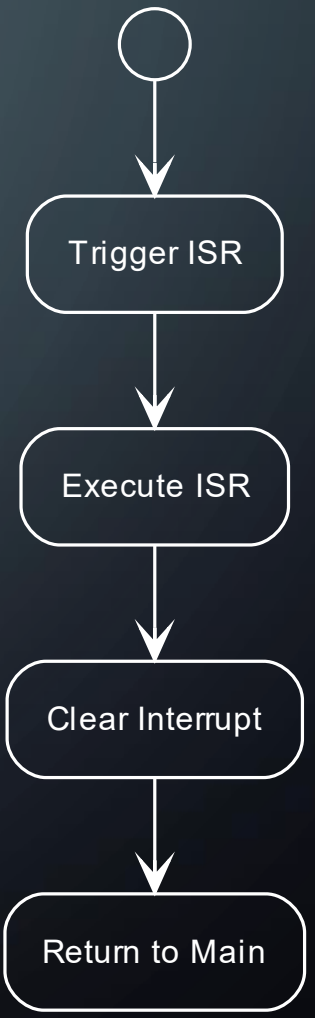
Polling

- In polling, the microcontroller continuously checks the status of a specific input or flag to determine if an event has occurred.
- It consumes CPU cycles even when there is no event, leading to inefficiency.
- Suitable for simple and infrequent events.



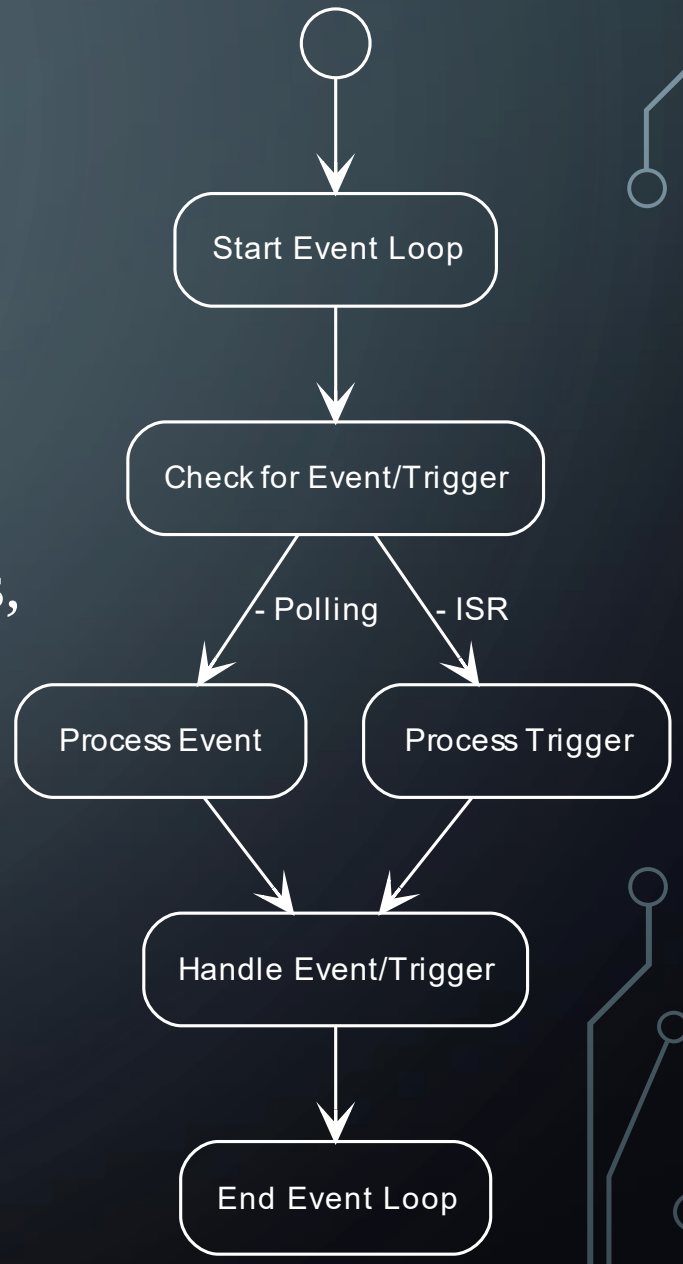
Interrupt

- In an interrupt-driven approach, the microcontroller responds to external events by automatically triggering an ISR when an event occurs.
- This method is more efficient than polling, as the CPU is freed from continuously checking for events.
- Ideal for handling time-critical or real-time events.



Event Driven

- Event-driven systems use a combination of polling and interrupts to respond to different types of events.
- Polling is used for less critical or non-time-sensitive events, while interrupts are utilized for critical and time-sensitive events.
- Provides a balance between efficiency and responsiveness.



Applications

- **Real-Time Control**
  - Interrupts are crucial for handling real-time events in motor control, robotics, and industrial automation.
- **Communication**
  - Interrupts enable the microcontroller to respond quickly to incoming data from UART, SPI, or I2C interfaces.
- **Sensor Handling**
  - Interrupts can be used to detect changes from sensors, such as detecting button presses or monitoring motion sensors.
- **Timer and PWM Control**
  - Interrupts facilitate precise timing control, such as generating PWM signals or capturing time intervals.

Examples

- **External Interrupts**
  - Configuring GPIO pins to generate external interrupts when a button is pressed, allowing quick response to user input.
- **UART Interrupts**
  - Using UART receive interrupts to process incoming data in real-time, avoiding data loss and CPU overutilization.
- **Timer Interrupts**
  - Employing timer interrupts to trigger periodic tasks, like updating display content or acquiring sensor data at fixed intervals.
- **ADC Interrupts**
  - Using ADC interrupts to respond to analog input changes and process analog data efficiently.

Parameters

- $T_T = Task\ Time(s)$
- $T_R = Task\ Rate\ (s)$

- $Valid\ Implementation = T_T < T_R$
  - System will work properly
- $Unvalid\ Implementation = T_T \geq T_R$
  - System Stuck in virtual loop
  - Loss other interrupts