1. Introduction to Interrupt

An interrupt is a mechanism used in microcontroller programming to pause the execution of the current program and call a specific routine or function when a particular event occurs. These events, defined by the programmer, can range from a specific condition on an input pin to a timer overflow or other hardware-defined triggers. Once the routine or function finishes executing, the program resumes from the exact point where it was interrupted.

The Arduino Uno (ATMega328P) provides two primary types of interrupts: External Interrupts and Timer Interrupts. While their functions and triggers differ, they share the same goal: interrupting the main program flow to handle specific events immediately.

External Interrupt

An External Interrupt is triggered by a change in the voltage level on specific input pins of the microcontroller. On the Arduino Uno, there are two pins dedicated to external interrupts: Pin 2 and Pin 3. These are the most commonly used pins when implementing interrupts via the Arduino programming language.

External Interrupts: INT0 and INT1

While the Arduino IDE refers to these simply as Pin 2 and Pin 3, the ATMega328P datasheet identifies them as **INT0** and **INT1**. These are hardware-level designations that correspond to specific physical pins.

- **INT0 (Digital Pin 2):** This is the first external interrupt. It has a higher priority in the Interrupt Vector Table than INT1, meaning if both occur at the exact same time, the microcontroller will handle INT0 first.
- **INT1 (Digital Pin 3):** This is the second external interrupt.

Trigger Modes

Both INT0 and INT1 can be configured to trigger the Interrupt Service Routine (ISR) based on four specific signal states:

1. **LOW:** Triggered whenever the pin is at a logic low level.
2. **CHANGE:** Triggered whenever the pin changes value (High to Low or Low to High).
3. **RISING:** Triggered specifically when the pin goes from Low to High.
4. **FALLING:** Triggered specifically when the pin goes from High to Low.

Internal Interrupt

An Internal Interrupt is triggered by modules located inside the microcontroller itself. In the ATMega328P, these are generated by timers and are referred to as Timer Interrupts. A Timer Interrupt is specifically triggered by a timer overflow. The Arduino Uno features three internal timers: Timer0, Timer1, and Timer2. (For more details on how timers operate, please refer to the previous module).

To expand on the specific hardware components of the ATMega328P (the heart of the Arduino Uno), we need to look at how the microcontroller labels and manages these specific interrupt sources.

Internal Interrupts: Timer0, Timer1, and Timer2

The Arduino Uno has three hardware timers, each capable of generating interrupts. These are essential for tasks that require precise timing without blocking the `void loop()`.

Name	Size	Possible Interrupts	Uses in Arduino
TIMER0	8 bits (0 - 255)	• Compare Match • Overflow	• delay(), millis(), micros() • analogWrite() pins 5, 6
TIMER1	16 bits (0 - 65,535)	• Compare Match • Overflow • Input Capture	• Servo functions • analogWrite() pins 9, 10
TIMER2	8 bits (0 - 255)	• Compare Match • Overflow	• tone() • analogWrite() pins 3, 11

1. Timer0 (8-bit)

- **Role:** This timer is used by the Arduino internal functions like `delay()`, `millis()`, and `micros()`.
- **Interrupt Potential:** It can trigger an **Overflow Interrupt** (when the counter hits 255 and resets to 0) or a **Compare Match Interrupt**.
- **Note:** Modifying Timer0 registers directly is generally discouraged because it will break the Arduino's built-in time-keeping functions.

2. Timer1 (16-bit)

- **Role:** Because it is a 16-bit timer, it can count up to 65,535. This allows for much longer and more precise timing intervals than Timer0 or Timer2.
- **Use Case:** It is frequently used by the `Servo` library.

- **Interrupt Potential:** Like Timer0, it supports Overflow and Compare Match interrupts, but with much higher resolution.

3. Timer2 (8-bit)

- **Role:** This is another 8-bit timer, similar to Timer0, but it is "independent" of the main time-keeping functions.
- **Use Case:** It is commonly used by the `tone()` library for generating audio frequencies.
- **Interrupt Potential:** It is an excellent choice for a custom periodic interrupt (e.g., checking a sensor every 1ms) without interfering with `delay()`.

Revision #3
Created 2026-03-11 15:25:28 UTC by CH
Updated 2026-03-11 17:11:23 UTC by CH