5.2 An Overview of Asynchronous Tools in FreeRTOS

Software Timers: For Application-Scheduled Events

A **FreeRTOS Software Timer** is a tool used to schedule the execution of a function at a future time. It's like setting an alarm clock within your software. When the timer expires, the RTOS automatically calls a predefined function, known as a **callback function**.

Key Characteristics:

- **Managed by the RTOS:** Software timers are managed by a dedicated RTOS task (the "timer daemon"). This means they do not consume CPU time while they are waiting to expire.
- **Tied to the System Tick:** The resolution of a software timer is determined by the FreeRTOS system tick rate (`configTICK_RATE_HZ`). You cannot schedule a timer for a period shorter than one tick.
- **Use Case:** Ideal for repetitive, low-priority, or application-level timing. For example, you might use a software timer to:
  - Read a temperature sensor every five seconds.
  - Update a clock display once per minute.
  - Turn off an LED 500ms after it was turned on.

Types of Software Timers:

1. **One-Shot Timer:** Executes its callback function only once after it is started.
2. **Auto-Reload Timer:** Executes its callback function repeatedly at a fixed interval until it is explicitly stopped.

Hardware Interrupts: For Hardware-Triggered Events

A **Hardware Interrupt** is a mechanism for a hardware peripheral to signal the CPU that it needs immediate attention. Unlike a software timer, which is scheduled by your application, an interrupt

is triggered by an external, physical event.

Key Characteristics:

- **High Priority:** An interrupt will immediately preempt the currently running code, regardless of the task's priority. The CPU will save its current state and jump to execute the ISR.
- **Hardware-Driven:** They are generated by peripherals like GPIO pins (e.g., a button press), hardware timers (for precise timing), or communication interfaces like UART/SPI (e.g., data has arrived).
- **Use Case:** Essential for time-critical operations and reacting to external events with minimal latency. For example, you would use a hardware interrupt to:
  - Count pulses from a motor encoder to measure its speed.
  - Immediately stop a machine when a safety limit switch is triggered.
  - Capture incoming data from a high-speed sensor before it is overwritten.

In summary, the choice between them is driven by the source of the event:

- Use a **Software Timer** when the event is driven by the logic of your application ("I need to do X in 500 milliseconds").
- Use a **Hardware Interrupt** when the event is driven by an external hardware signal that requires an immediate response ("The hardware needs attention NOW").

Revision #1
Created 2025-08-28 12:43:12 UTC by GI
Updated 2025-08-28 12:43:50 UTC by GI