LIBRARY · REFERENCE GUIDE

CLOCKS AND TIMERS

A clock paces everything a microcontroller does, and timers count that clock to measure time or fire on a schedule. How a board blinks at 1 Hz or samples at exactly 250 Hz.

ONE THOUSAND DRONES ENGINEERING TEAM · VERIFIED 2026-07

Everything a microcontroller does is paced by a clock: a steady stream of pulses that steps the processor forward one tick at a time. Timers are hardware counters that count those pulses, so the chip can measure elapsed time and trigger events on an exact schedule. Together they are how a board blinks at `1 Hz` or samples a sensor at exactly `250 Hz`.

THE CLOCK SETS THE PACE

The system clock is an oscillator running at a fixed frequency, from a few megahertz to hundreds of megahertz. Every instruction the CPU runs and every peripheral it drives is timed off that clock. A faster clock does more work per second but burns more power, which is one of the first knobs a low-power design turns down when it does not need the speed.

A TIMER COUNTS TICKS

A timer is a counter wired to the clock, or to a divided-down version of it. It increments on every tick. Read it to measure how much time has passed, or set it to fire an event when it reaches a target count. Because it is counting a known frequency, a count converts straight into a span of time, and a target count converts into a delay.

$$t = N / f_{clk}$$

A timer that counts `N` ticks of a clock running at `f_clk` has measured `t` seconds. Turn it around, and to get a `1 ms` tick from a `1 MHz` timer clock you count to `1000`.

PERIODIC EVENTS INSTEAD OF BUSY-WAITING

To do something at an exact rate, set a timer to fire periodically and let it call you (an interrupt, the next lesson), rather than counting in a loop. A software delay loop blocks the CPU the whole time and drifts, because anything else the code does throws the timing off. A hardware timer keeps time in the background while your code gets on with other work, and its rate is as steady as the clock. The ESP32's general-purpose timers do exactly this.

- Espressif. ESP-IDF Programming Guide: General Purpose Timer (GPTimer). docs.espressif.com

DEEP DIVE · WHY EXACT TIMING MATTERS WHEN YOU SAMPLE

A data logger or a sensor front-end needs its samples spaced evenly in time, or the uneven spacing shows up as noise and error in the data later. If you sample inside a plain loop, the spacing wobbles with whatever else the loop is doing that pass. A hardware timer set to the sample rate fixes it: it fires an interrupt every period, and the handler takes exactly one reading, so the spacing is as steady as the clock crystal rather than as steady as your code. That is how a board holds a true `250 Hz` or `1 kHz` sample rate. (Espressif GPTimer)

A TIMER COUNTS CLOCK TICKS AND FIRES EVERY N OF THEM, KEEPING AN EXACT RATE.

CHECKPOINT

1. How does a board sample a sensor at an exact rate?

a. A timer counts the clock and fires periodically

b. It reads as fast as the main loop happens to come around

c. It uses more memory

ANSWER · A

A hardware timer paces the samples off the clock, so the spacing stays exact.

2. What paces everything a microcontroller does?

a. The supply voltage

b. Its system clock

c. The number of GPIO pins

ANSWER · B

The clock steps the CPU and peripherals; its frequency sets the speed.

3. Why is a hardware timer better than a software delay loop for timing?

a. It uses less flash memory

b. It reads more cleanly in the code

c. It keeps time in the background without blocking the CPU, and does not drift

ANSWER · C

A delay loop ties up the CPU and drifts; a timer counts the clock independently and stays exact.

- Next: interrupts