# Architectures for Embedded Systems

# Timers
# Watchdog timers
# PWM Generators
# Laboratory assignment

Arnaldo S. R. Oliveira

Academic year 2024/25

Universidade de Aveiro – Dep. de Eletrónica, Telecomunicações e Informática

# Outline

Timers

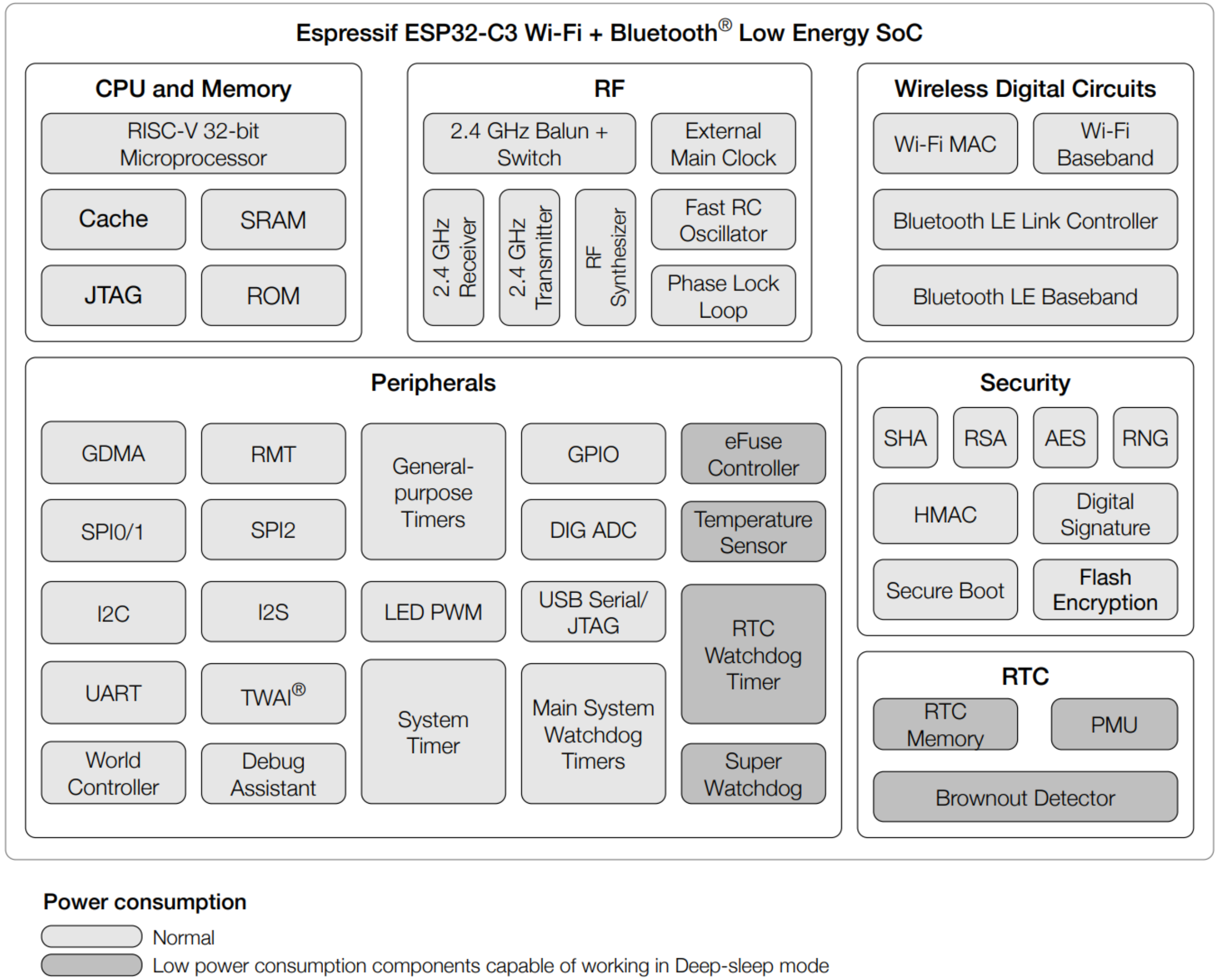
- Basic principles
- System Timer
- Timer Group (general-purpose timers)

Watchdog Timers

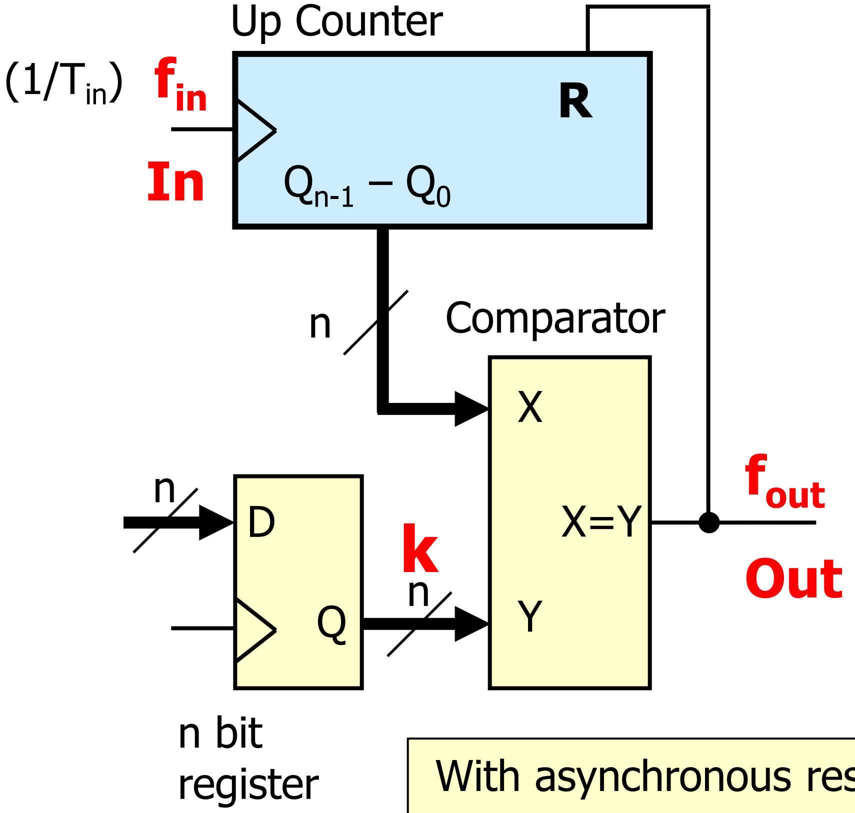
PWM Generators (LED controllers)

Lab assignment

# ESP32-C3 µC (SoC) Components



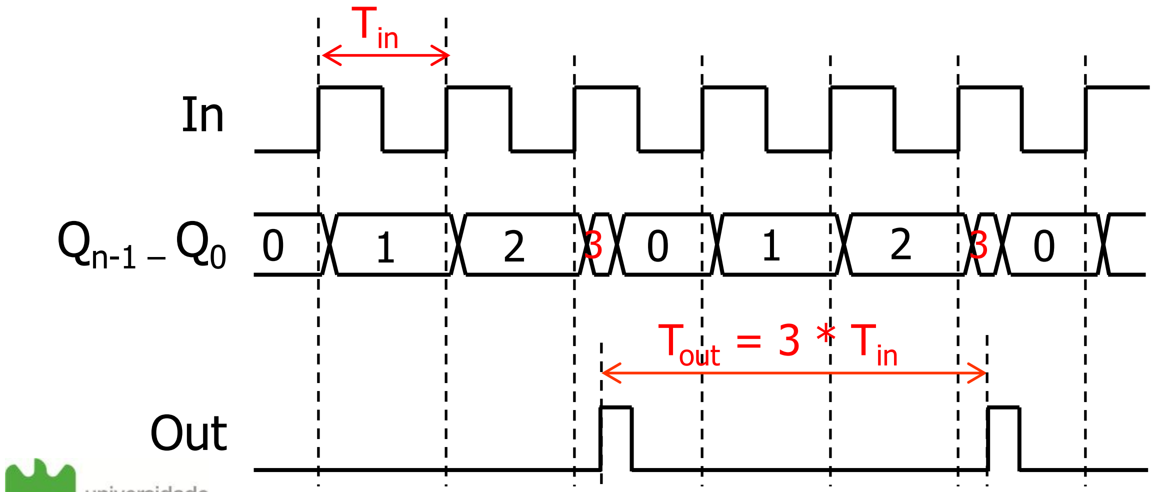
# Timers – frequency control (periodic event) Operation basics (asynchronous counter reset)



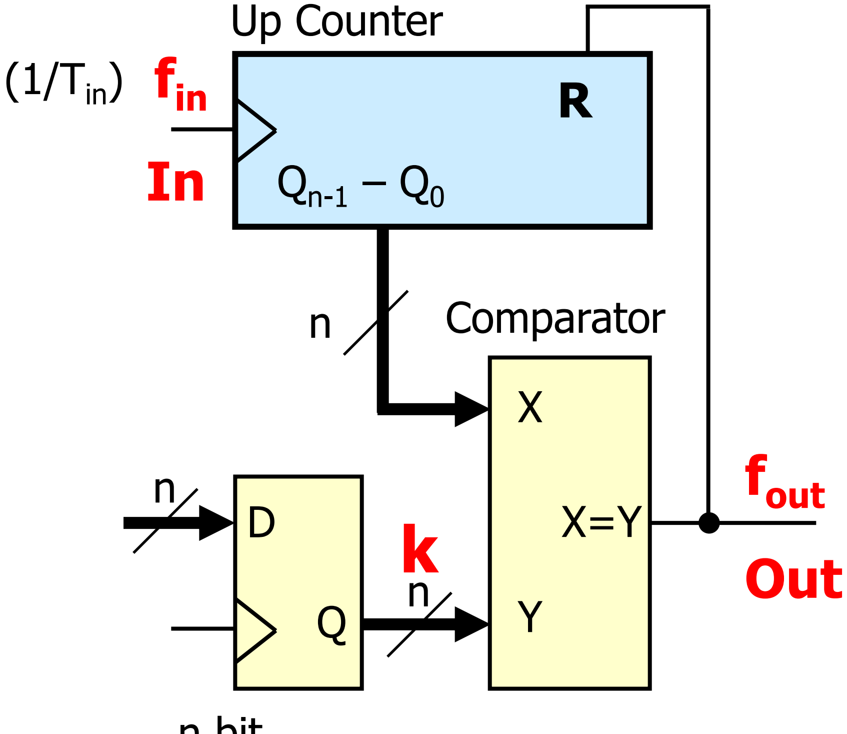
With asynchronous reset:

$$f_{out} = f_{in} / k$$

- Example:
  - Asynchronous counter reset
  - $k = 3$
- Output signal period:
  - $T_{out} = k * T_{in}$, or, $f_{out} = f_{in} / k$
- Output pulse duration not controlable (due to propagation delays)



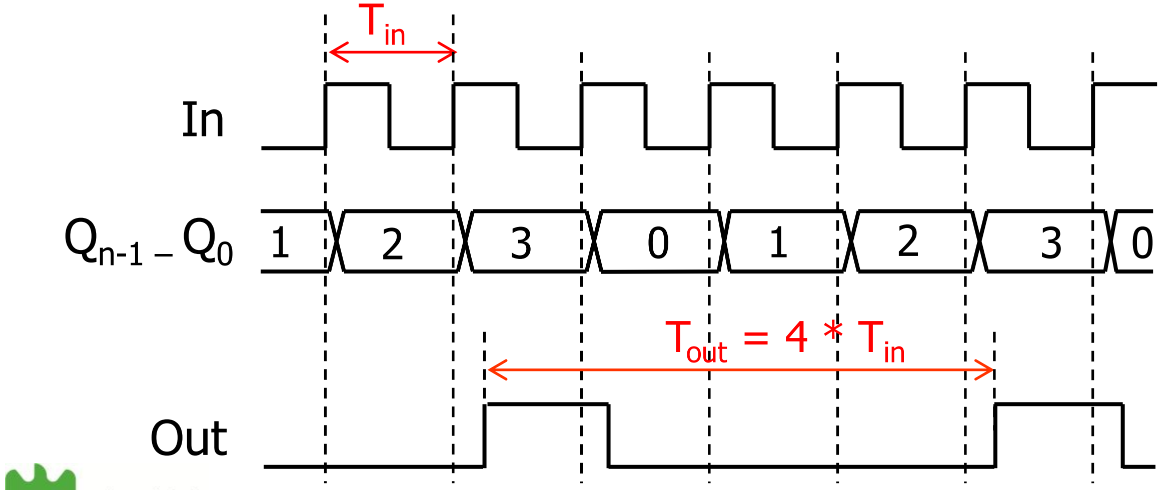
# Timers – frequency control (periodic event) Operation basics (synchronous counter reset)



With synchronous reset:

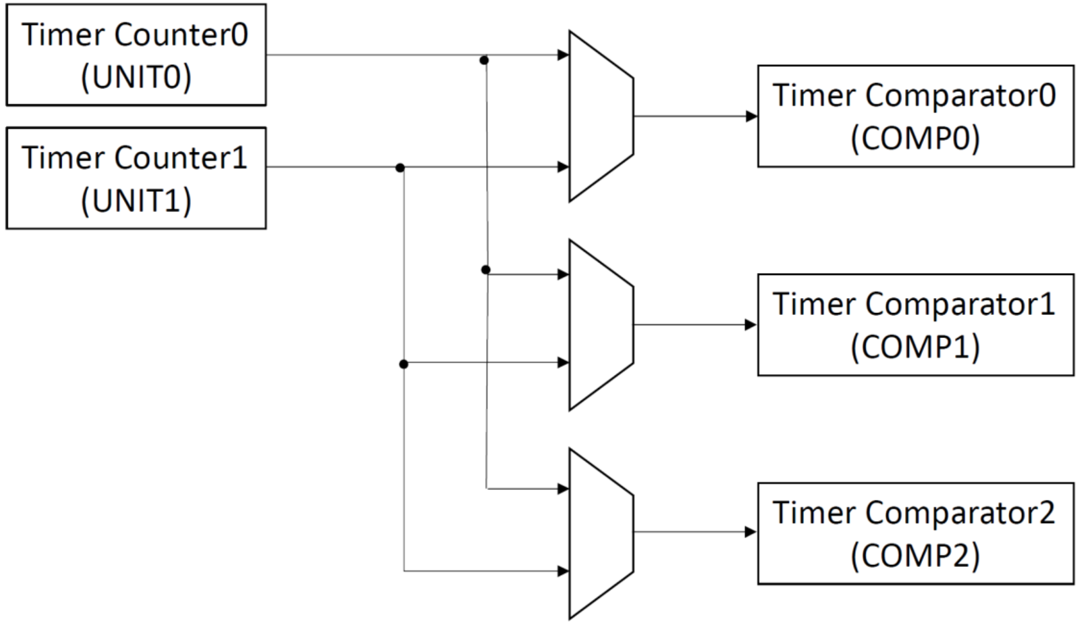
$$f_{out} = f_{in} / (k+1)$$

- Example:
  - Synchronous counter reset
  - k = 3
- Output signal period:
  - $T_{out} = (k+1) * T_{in}$, or, $f_{out} = f_{in} / (k+1)$
- Output pulse duration is 1 clock cycle



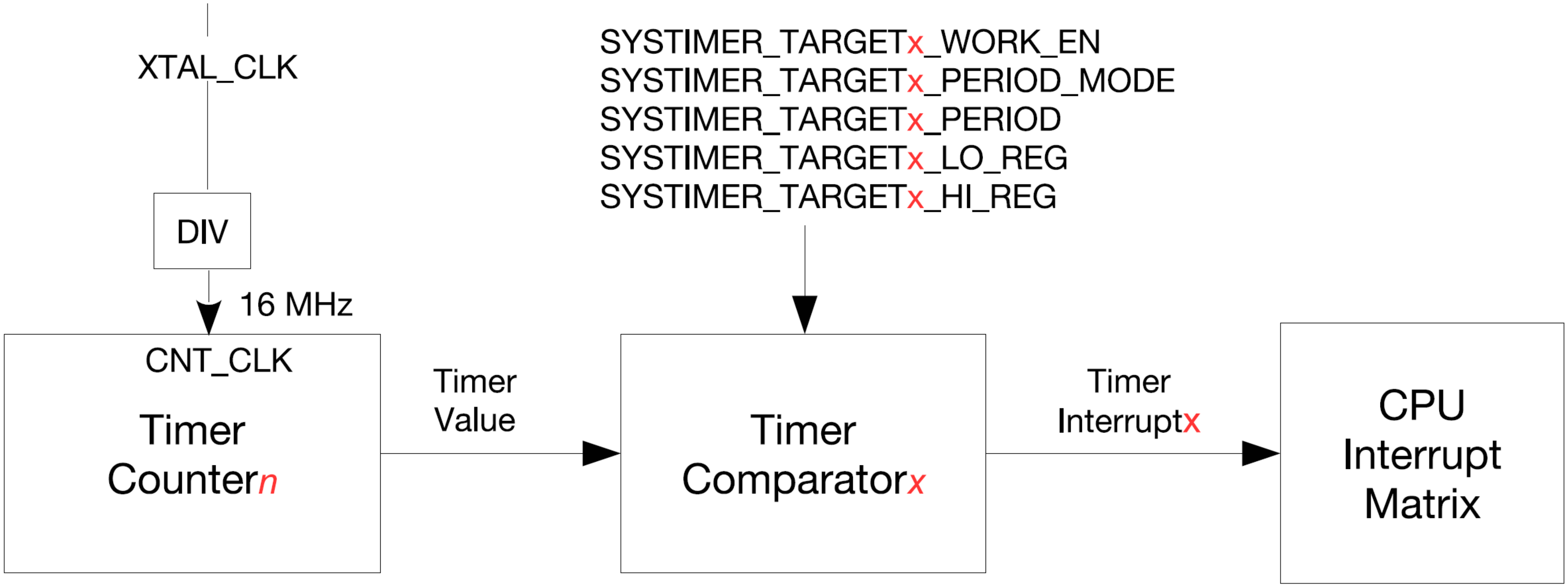
# System Timer @ ESP32

- Consist of two 52-bit counters and three 52-bit comparators
- Software accessing registers is clocked by APB_CLK
- Use CNT_CLK for counting, with an average frequency of 16 MHz in two counting cycles
- Use 40 MHz XTAL_CLK as the clock source of CNT_CLK
- Support for 52-bit alarm values (t) and 26-bit alarm periods (δt)
- Provide two modes to generate alarms:
  - Target mode: only a one-time alarm is generated based on the alarm value (t)
  - Period mode: periodic alarms are generated based on the alarm period (δt)
- Three comparators can generate three independent interrupts based on configured alarm value (t) or alarm period (δt)



Source: ESP32-C3 Technical Reference Manual, page 269

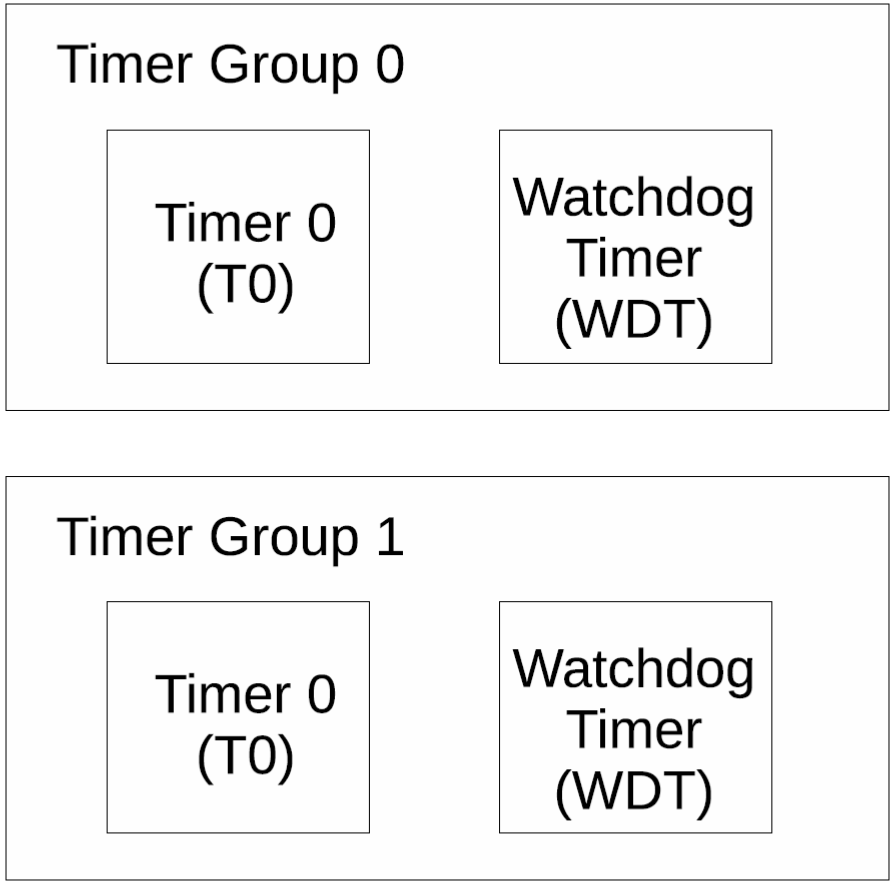
# System Timer @ ESP32



Source: ESP32-C3 Technical Reference Manual, page 270

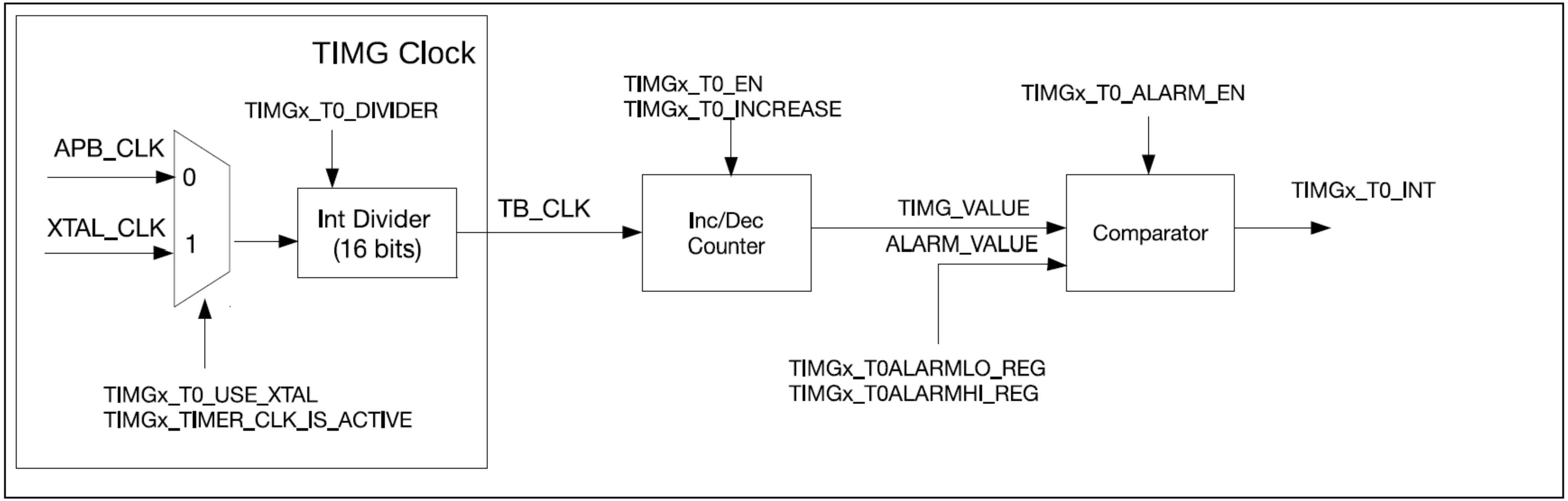
# Timer Group @ ESP32 (general purpose timers)

- A 16-bit clock prescaler, from 2 to 65536
- A 54-bit time-base counter programmable to incrementing or decrementing
- Able to read real-time value of the time-base counter
- Halting and resuming the time-base counter
- Programmable alarm generation
- Timer value reload (Auto-reload at alarm or software-controlled instant reload)
- Level interrupt generation



Source: ESP32-C3 Technical Reference Manual, page 287

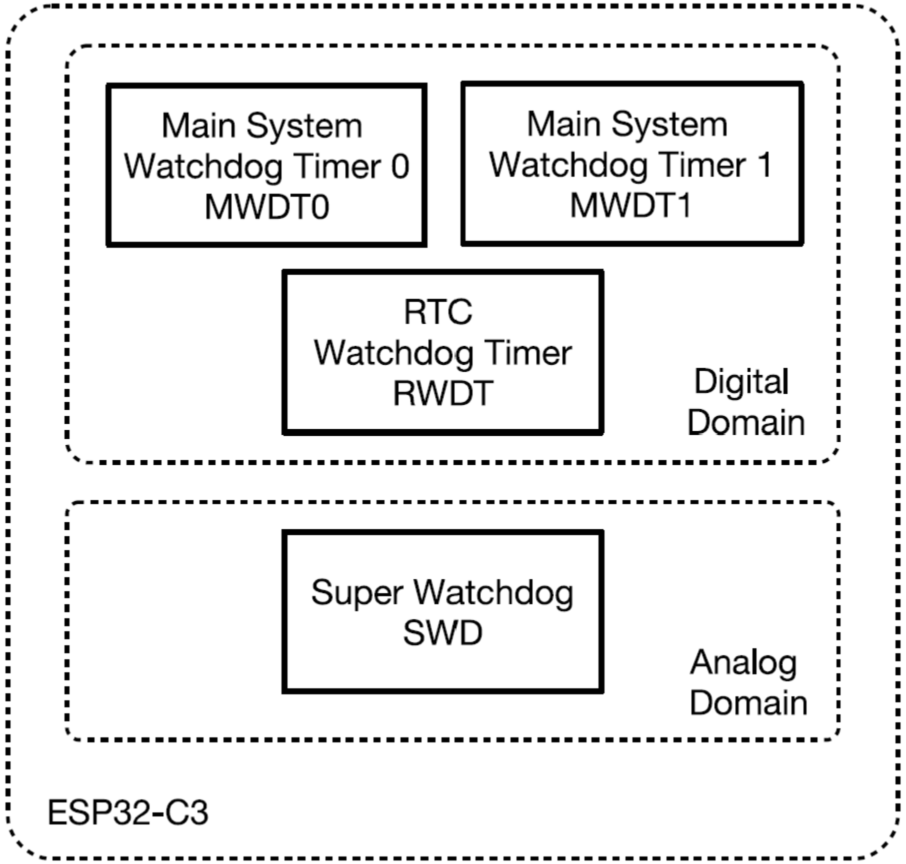
# Timer Group @ ESP32 (general purpose timers)



Source: ESP32-C3 Technical Reference Manual, page 288

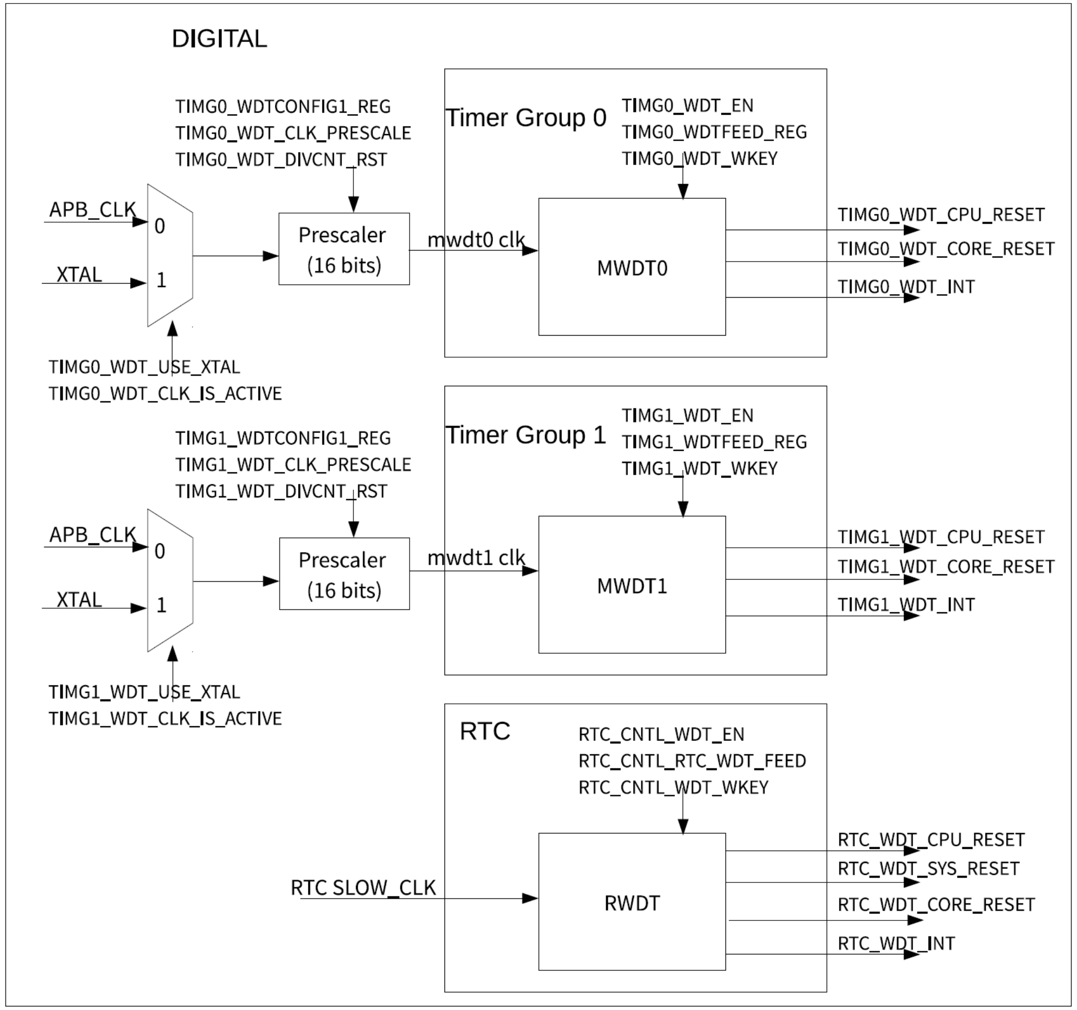
# Watchdog Timers @ ESP32

- Four stages, each with a programmable timeout value. Each stage can be configured and enabled/disabled separately
- Three timeout actions (interrupt, CPU reset, or core reset) for MWDT and four timeout actions (interrupt, CPU reset, core reset, or system reset) for RWDT upon expiry of each stage
- 32-bit expiry counter
- Write protection, to prevent RWDT and MWDT configuration from being altered inadvertently
- Flash boot protection - If the boot process from an SPI flash does not complete within a predetermined period of time, the watchdog will reboot the entire main system



Source: ESP32-C3 Technical Reference Manual, page 304

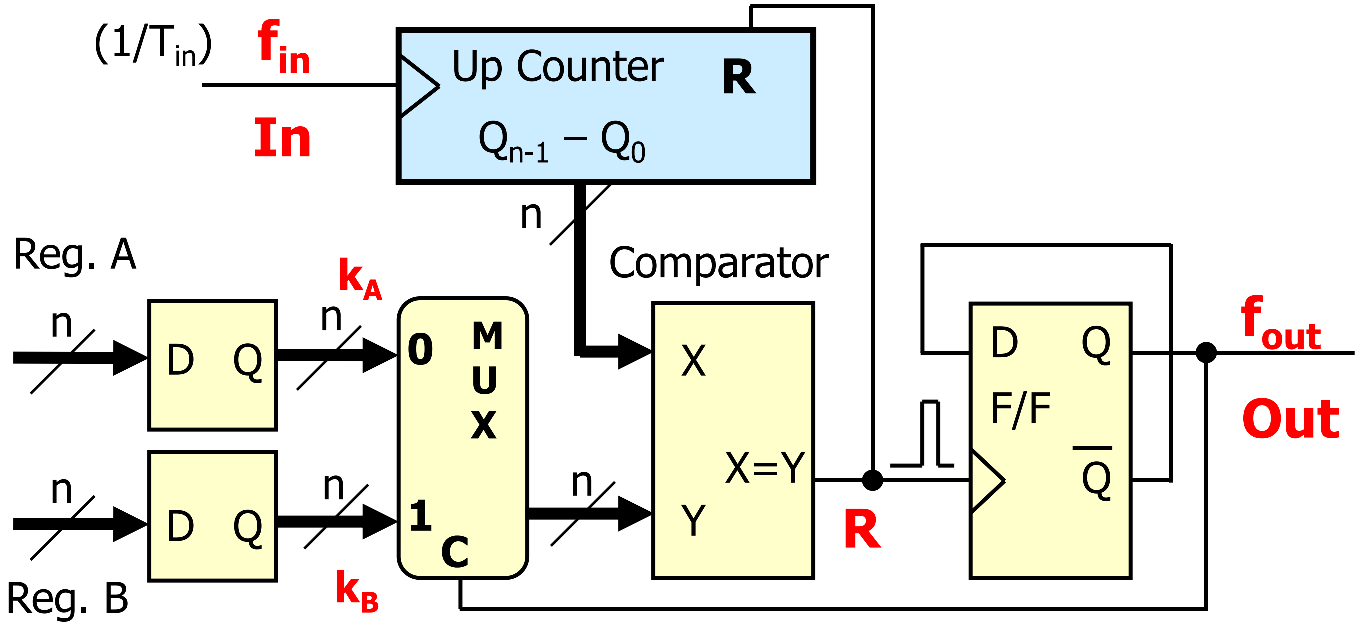
# Watchdog Timers @ ESP32



Source:
ESP32-C3 Technical Reference Manual
page 306

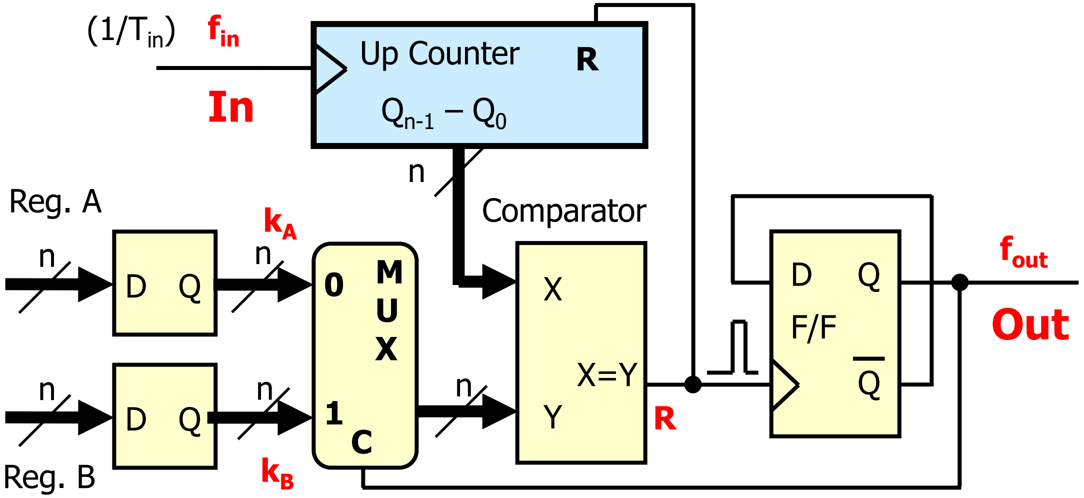
# Timers – Frequency and duty-cycle control (PWM generator operation basics)

- Control the output signal period, as well as the time this signal is set to "1"



- When Q output of the flip-flop is set to "1", counter is compared with $k_B$
- Otherwise, counter is compared with $k_A$
- Therefore, the time during which the output signal is
  - set to "1", depends on $k_B$
  - set to "0", depends on $k_A$

# PWM Generator Operation Example

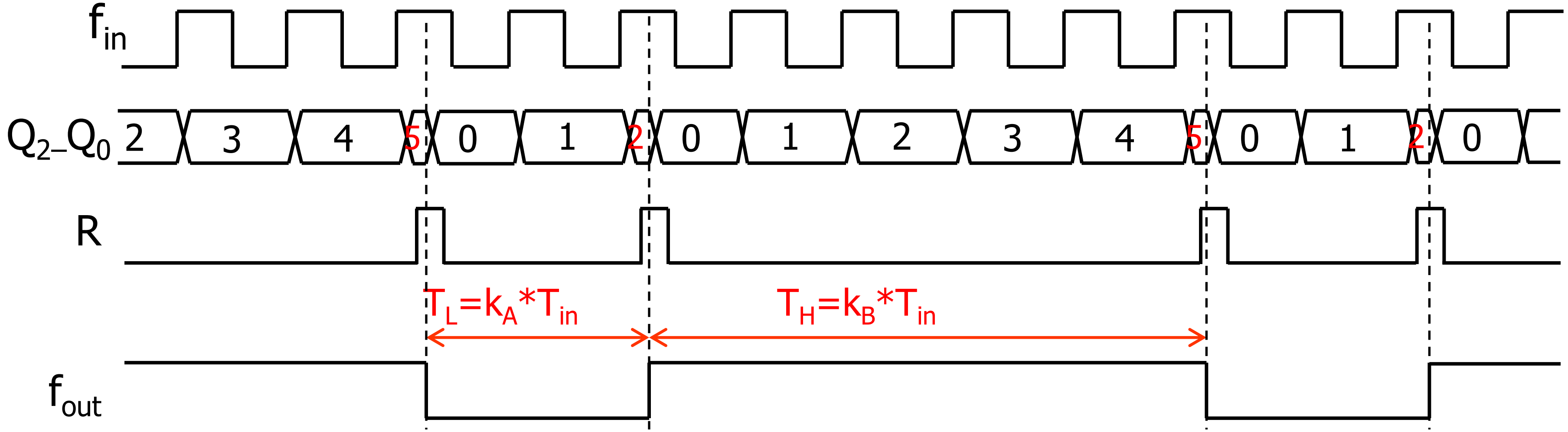


With asynchronous reset:

$f_{out} = f_{in} / (k_A+k_B)$ **D. Cycle** $= k_B/(k_A+k_B)$

With synchronous reset:

$f_{out} = f_{in} / (k_A+k_B+2)$ **D. Cycle** $= (k_B+1)/(k_A+k_B+2)$

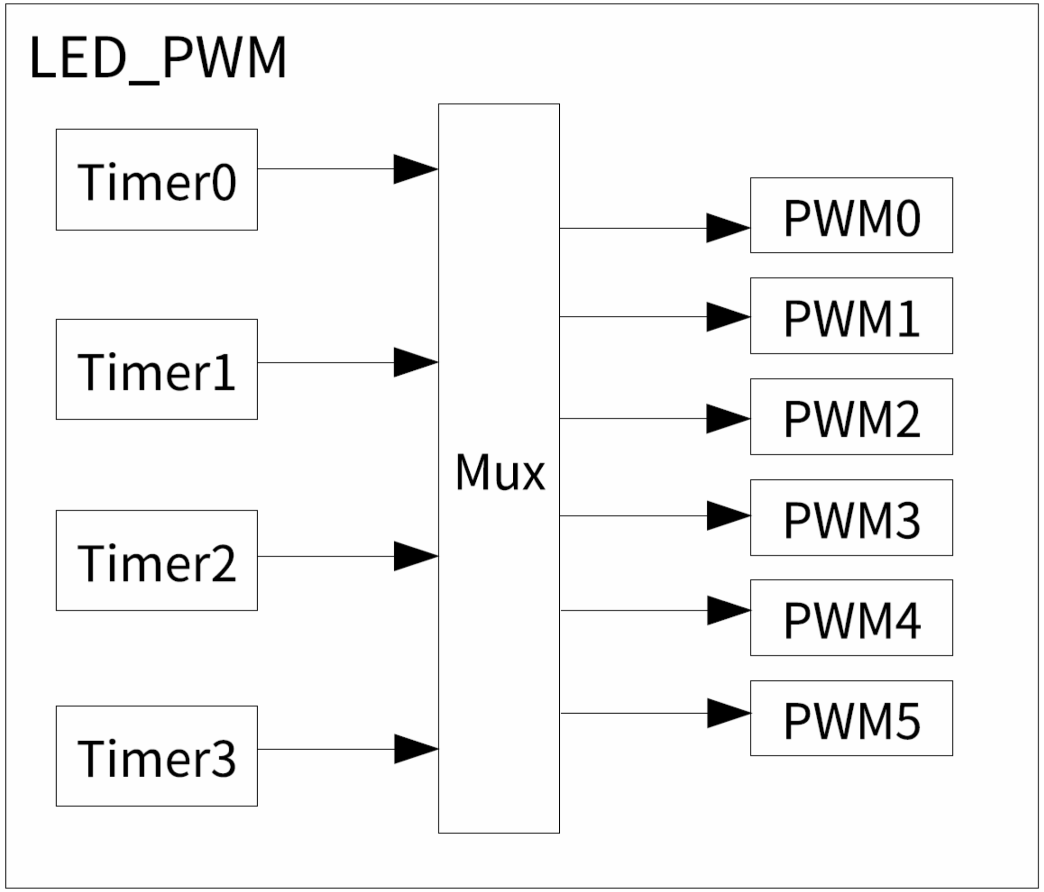


Example with $\mathbf{k_A=2}$, $\mathbf{k_B=5}$, (counter with asynchronous reset)

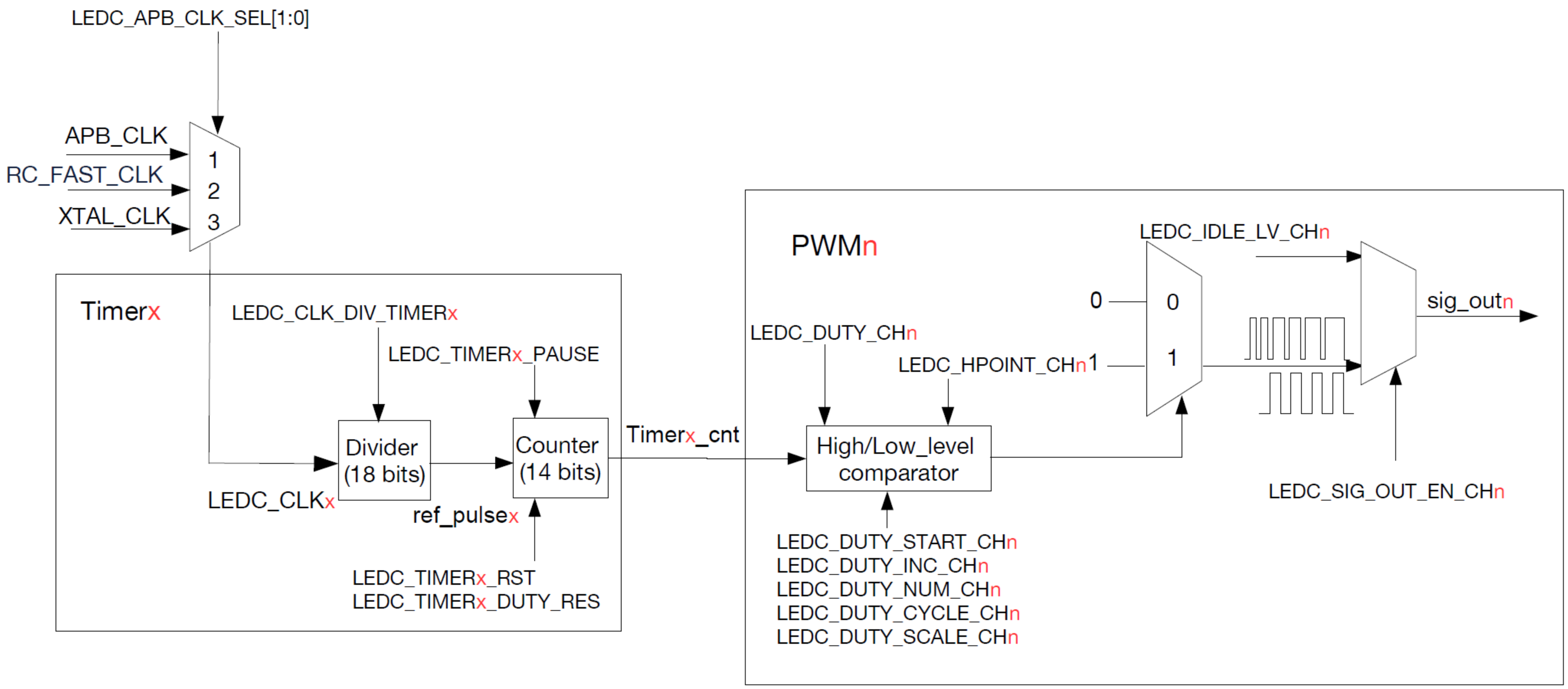
# PWM Generators (LED controllers @ ESP32)

- Six independent PWM generators (i.e. six channels)
- Four independent timers that support division by fractions
- Automatic duty cycle fading (i.e. gradual increase/decrease of a PWM’s duty cycle without interference from the processor) with interrupt generation on fade completion
- Adjustable phase of PWM signal output
- PWM signal output in low-power mode (Light-sleep mode)
- Maximum PWM resolution: 14 bits

Source: ESP32-C3 Technical Reference Manual, page 826



# PWM Generators (LED controllers @ ESP32)



Source: ESP32-C3 Technical Reference Manual, page 827

# Information Sources (to be autonomously explored and used in the lab assignment)

**System Timer**

API: https://docs.espressif.com/projects/esp-idf/en/v5.4/esp32c3/api-reference/system/esp_timer.html

Example: C:\Espressif\frameworks\esp-idf-v5.4\examples\system\esp_timer

**Timer Group (general purpose timers)**

API: https://docs.espressif.com/projects/esp-idf/en/v5.4/esp32c3/api-reference/peripherals/gptimer.html

Example: C:\Espressif\frameworks\esp-idf-v5.4\examples\peripherals\timer_group\gptimer

**Watchdog timers**

API: https://docs.espressif.com/projects/esp-idf/en/v5.4/esp32c3/api-reference/system/wdts.html

**PWM Generators (LED controllers)**

API: https://docs.espressif.com/projects/esp-idf/en/v5.4/esp32c3/api-reference/peripherals/ledc.html

Example: C:\Espressif\frameworks\esp-idf-v5.4\examples\peripherals\ledc\ledc_basic

# Laboratory Assignment – Hardware supported LED brightness control

- Create a new project to control the brightness of the LED based on the following specifications:
  - Supports 10 levels of brightness
  - Stay in each level for 2 seconds before changing to the next level
  - Wrap-up around to the first level after staying 2 seconds in the last level
  - **Use ESP32 hardware timers and PWM generators**
- Compile and test the project (naked eye and oscilloscope)

**Hint:** use the “Information Sources” provided in a previous slide

**Restriction:** the timings specified above cannot be based on the **`usleep()`** or **`vTaskDelay()`** functions

# Final Remarks

- At the end of this week, you should be familiar with:
  - General purpose timers (basics, usage, programming and testing)
  - Watchdog timers (basics and usage)
  - PWM generators (basics, usage, programming and testing)