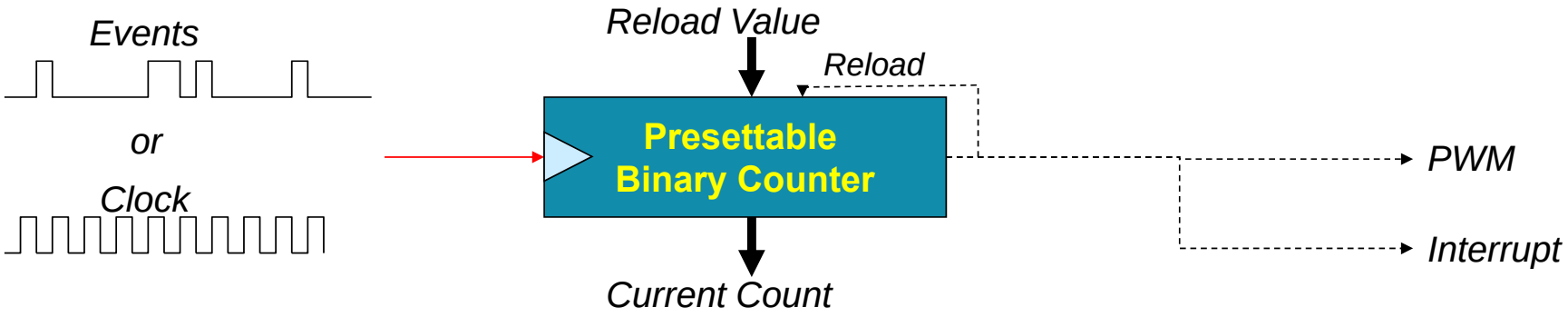
Timer Peripherals

Types of Timer Peripherals

- Interrupt Timer
  - Can generate periodically interrupts or trigger DMA (direct memory access) transfers
- PWM Module
  - Connected to I/O pins, has input capture and output compare support
  - Can generate PWM signals
  - Can generate interrupt requests
- Low-Power Timer
  - Can operate as timer or counter in all power modes
  - Can wake up system with interrupt
  - Can trigger hardware
- Real-Time Clock
  - Powered by external 32.768 kHz crystal
  - Tracks elapsed time (seconds) in register
  - Can set alarm
  - Can generate 1Hz output signal and/or interrupt
  - Can wake up system with interrupt
- SYSTICK
  - Part of CPU core's peripherals
  - Can generate periodic interrupt

ARM

Timer/Counter Peripheral Introduction



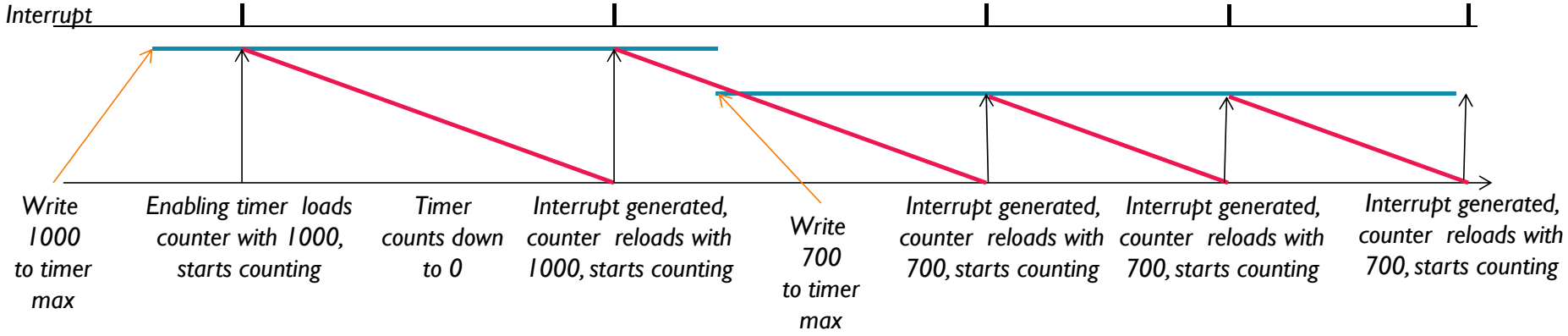
- Common peripheral for microcontrollers
- Based on presettable binary counter, enhanced with configurability
  - Count value can be read and written by MCU
  - Count direction can often be set to up or down
  - Counter's clock source can be selected
    - Counter mode: count pulses which indicate events (e.g. odometer pulses)
    - Timer mode: clock source is periodic, so counter value is proportional to elapsed time (e.g. stopwatch)
  - Counter's overflow/underflow action can be selected
    - Generate interrupt
    - Reload counter with special value and continue counting
    - Toggle hardware output signal

ARM

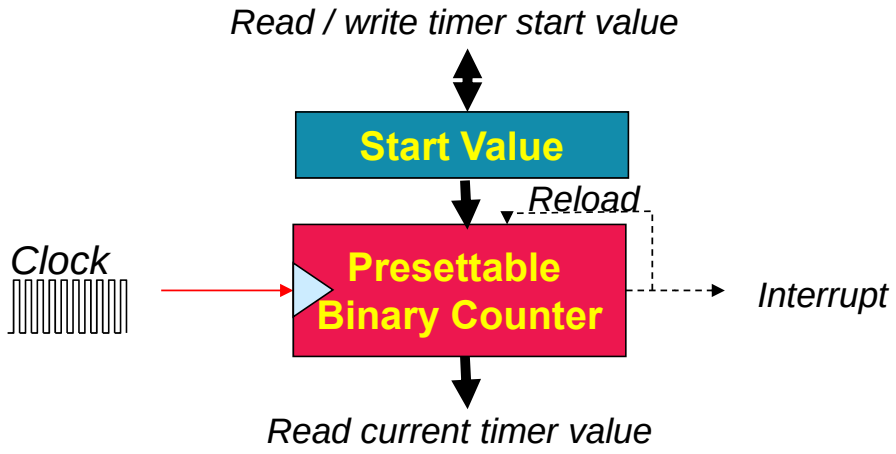
INTERRUPT TIMER

ARM

Interrupt Timer



- Load start value from register
- Counter counts down with each clock pulse
- When timer value reaches zero
  - Generates interrupt
  - Reloads timer with start value



Calculating Max Value

- Goal: generate an interrupt every T seconds
- Max value = round(T * Freq)
  - Round since register is an integer, not a real number
  - Rounding provides closest integer to desired value, resulting in minimum timing error
- Example: Interrupt every 137.41 ms, assuming clock frequency 24 MHz
  - 137.41 ms * 24 MHz = 3297840
- Example: Interrupt with a frequency of 91 Hz with a 12 MHz clock
  - (1/91 Hz) * 12 MHz = round (131868.1318) = 131868
- Use macros, interrupt 1000 times per second:
  - CLK_FREQ / 1000

Configuring the Interrupt Timer

- Setup timer, set to tick at 10 Hz
  - timer_init(CLK_FREQ / 10);
- Set interrupt
  - timer_set_callback(timer_isr);
- Enable module
  - timer_enable();
- Disable module
  - timer_disable();

Example: Stopwatch

- Measure time with 100 us resolution
- Display elapsed time, updating screen every 10 ms
- Controls
  - S1: toggle start/stop
- Use interrupt timer
  - Counter increment every 100 us
    - Set to timer to expire every 100 us
    - Calculate max value, e.g. at 24 MHz = round (100 us * 24 MHz -1) = 2399
  - LCD Update every 10 ms
    - Update LCD every nth ISR
    - n = 10 ms/100us = 100
    - Don't update LCD in ISR! Too slow.
    - Instead set flag in ISR, poll it in main loop

TIMER / PWM MODULE

ARM

Timer / PWM Module

- Core Counter
  - Clock options - external or internal
  - Prescaler to divide clock
  - Can reload with set value, or overflow and wrap around

- N channels
  - 3 modes
    - Capture Mode: Capture timer's value when input signal changes
    - Output Compare: Change an output signal when timer reaches certain value
    - PWM: Generate pulse-width-modulated signal. Width of pulse is proportional to specified value.
  - Possible triggering of interrupt, hardware trigger on overflow
  - One I/O pin per channel

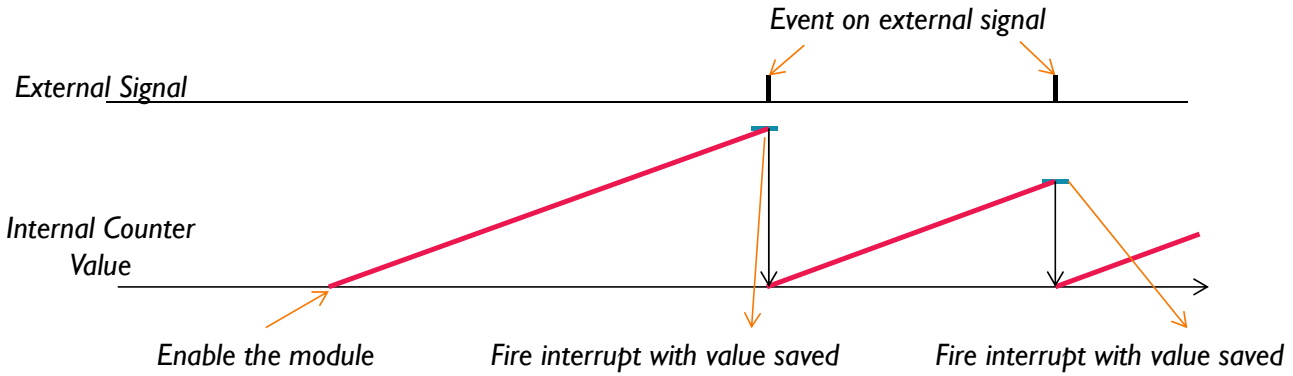
ARM

Major Channel Modes

- Input Capture Mode
  - Capture timer's value when input signal changes
    - Rising edge, falling edge, both
  - How long after I started the timer did the input change?
    - Measure time delay
- Output Compare Mode
  - Modify output signal when timer reaches specified value
    - Set, clear, pulse, toggle (invert)
  - Make a pulse of specified width
  - Make a pulse after specified delay
- Pulse Width Modulation
  - Make a series of pulses of specified width and frequency

ARM

Input Capture Mode



- I/O pin operates as input on edge
- When valid edge is detected on pin…
  - Current value of counter is stored
  - Interrupt is called

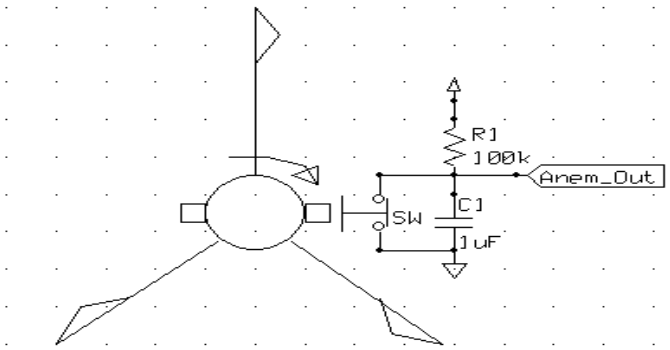
ARM

Wind Speed Indicator (Anemometer)

- Rotational speed (and pulse frequency) is proportional to wind velocity
- Two measurement options:
  - Frequency (best for high speeds)
  - Width (best for low speeds)
- Can solve for wind velocity v

$$v_{wind} = \frac{K * f_{clk}}{T_{anemometer}}$$

- How can we use the timer for this?
  - Use Input Capture Mode to measure period of input signal



ARM

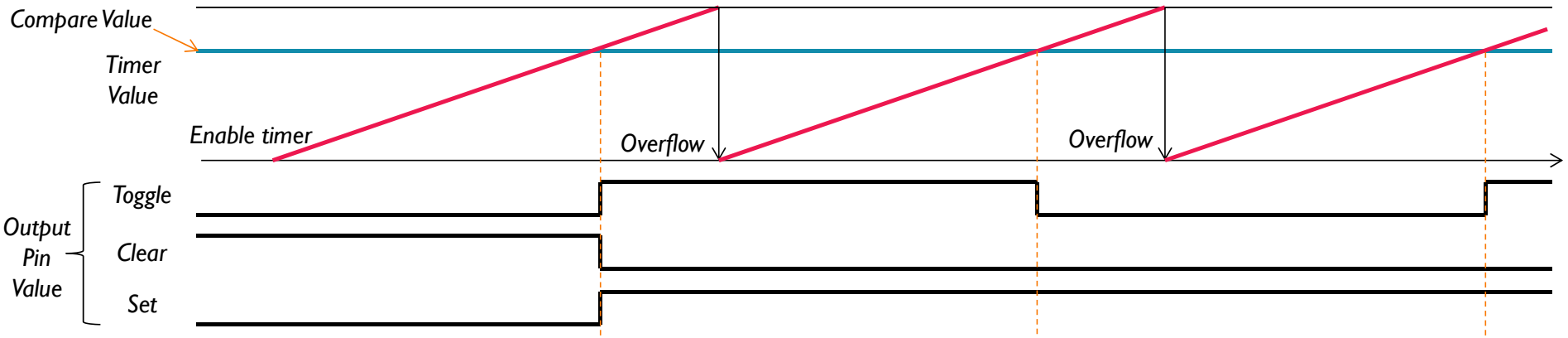
TPM Capture Mode for Anemometer

- Configuration
  - Set up module to count at given speed from internal clock
  - Set up channel for input capture on rising edge

- Operation: Repeat
  - First interrupt - on rising edge
    - Reconfigure channel for input capture on falling edge
    - Clear counter, start it counting
  - Second interrupt - on falling edge
    - Read capture value, save for later use in wind speed calculation
    - Reconfigure channel for input capture on rising edge
    - Clear counter, start it counting

ARM

Output Compare Mode

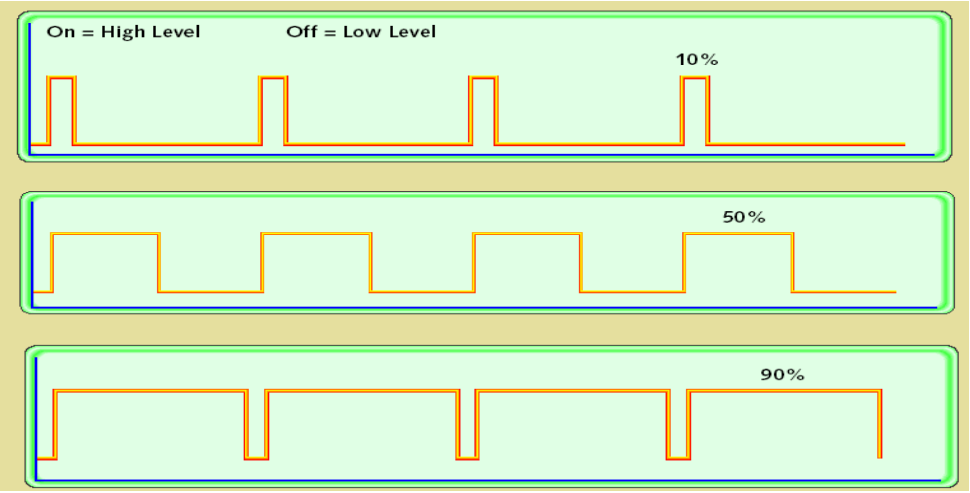


- Action on match
  - Toggle
  - Clear
  - Set
- When counter matches value …
  - Output signal is generated
  - Interrupt is called (if enabled)

ARM

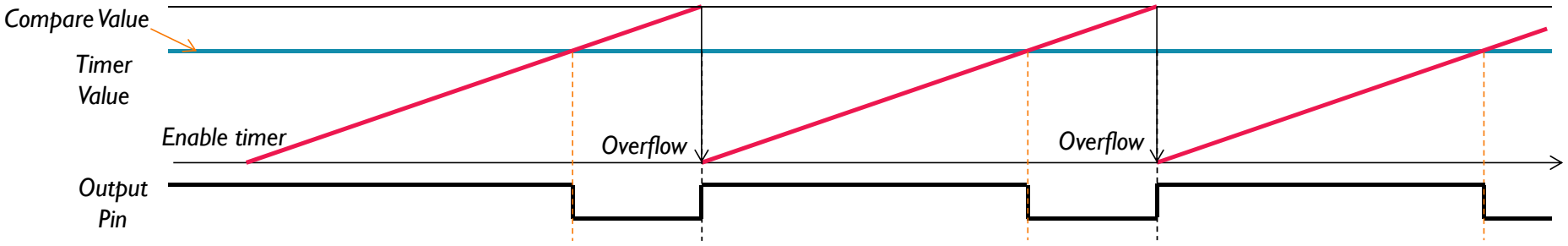
Pulse-Width Modulation

- Uses of PWM
  - Digital power amplifiers are more efficient and less expensive than analog power amplifiers
    - Applications: motor speed control, light dimmer, switch-mode power conversion
    - Load (motor, light, etc.) responds slowly, averages PWM signal
  - Digital communication is less sensitive to noise than analog methods
    - PWM provides a digital encoding of an analog value
    - Much less vulnerable to noise
- PWM signal characteristics
  - Modulation frequency – how many pulses occur per second (fixed)
  - Period – 1/(modulation frequency)
  - On-time – amount of time that each pulse is on (asserted)
  - Duty-cycle – on-time/period
  - Adjust on-time (hence duty cycle) to represent the analog value



ARM

PWM Mode

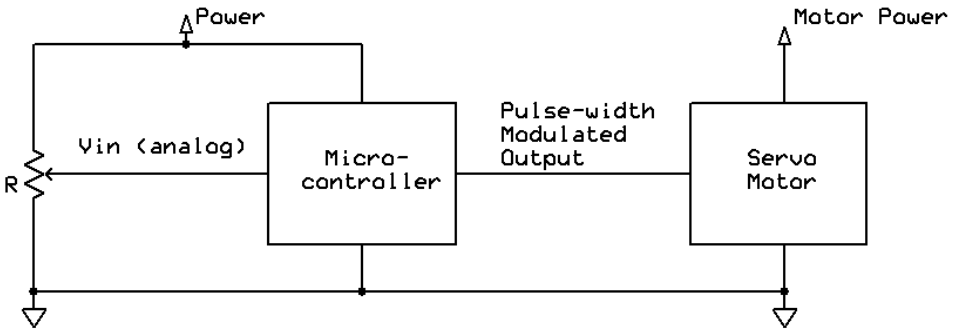


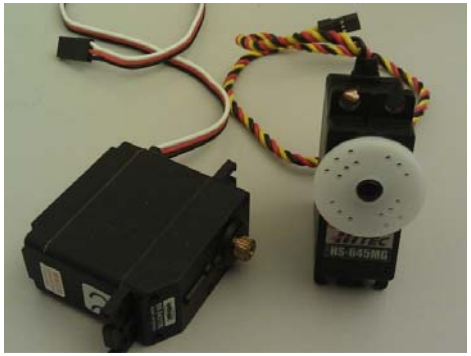
- PWM duty cycle proportional to compare value
  - Period = max timer value
  - Pulse width = compare value

$$Duty\ Cycle = \frac{Compare\ Value}{Max\ Value} \cdot 100\%$$

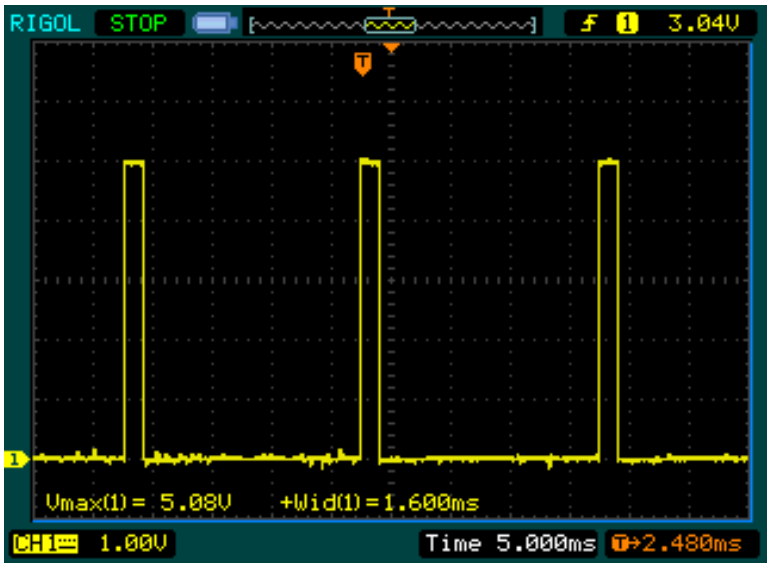
ARM

PWM to Drive Servo Motor





- Servo PWM signal
  - 20 ms period
  - 1 to 2 ms pulse width

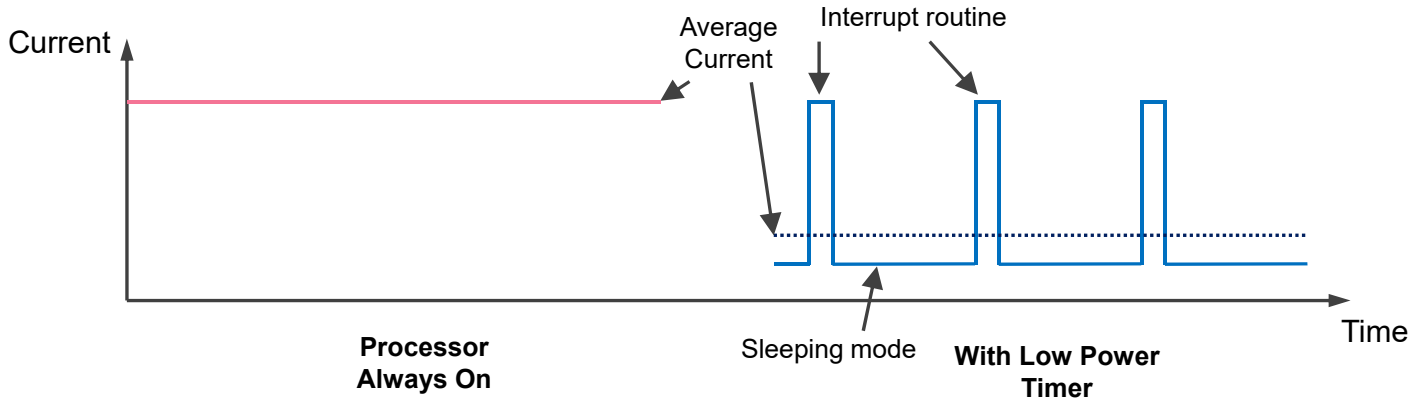


ARM

LOW POWER TIMER

ARM

Low Power Timer Overview



- Features
  - Count time or external pulses
  - Generate interrupt when counter matches compare value
  - Interrupt wakes MCU from any low power mode
- Current draw can be reduced to microamps or even nanoamps!
- Use the WFI instruction (Wait For Instruction)
  - Puts CPU in low power mode until interrupt request

ARM