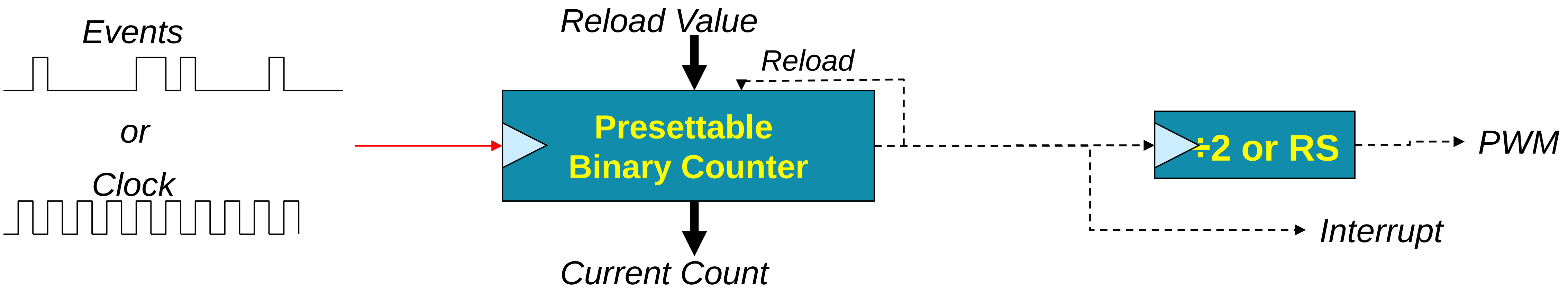
Module 4 - Timers

KL25 Timer Peripherals

- PIT - Periodic Interrupt Timer
  - Can generate periodically generate interrupts or trigger DMA (direct memory access) transfers
- TPM - Timer/PWM Module
  - Connected to I/O pins, has input capture and output compare support
  - Can generate PWM signals
  - Can generate interrupts and DMA requests
- LPTMR - Low-Power Timer
  - Can operate as timer or counter in all power modes (including low-leakage 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 32-bit 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

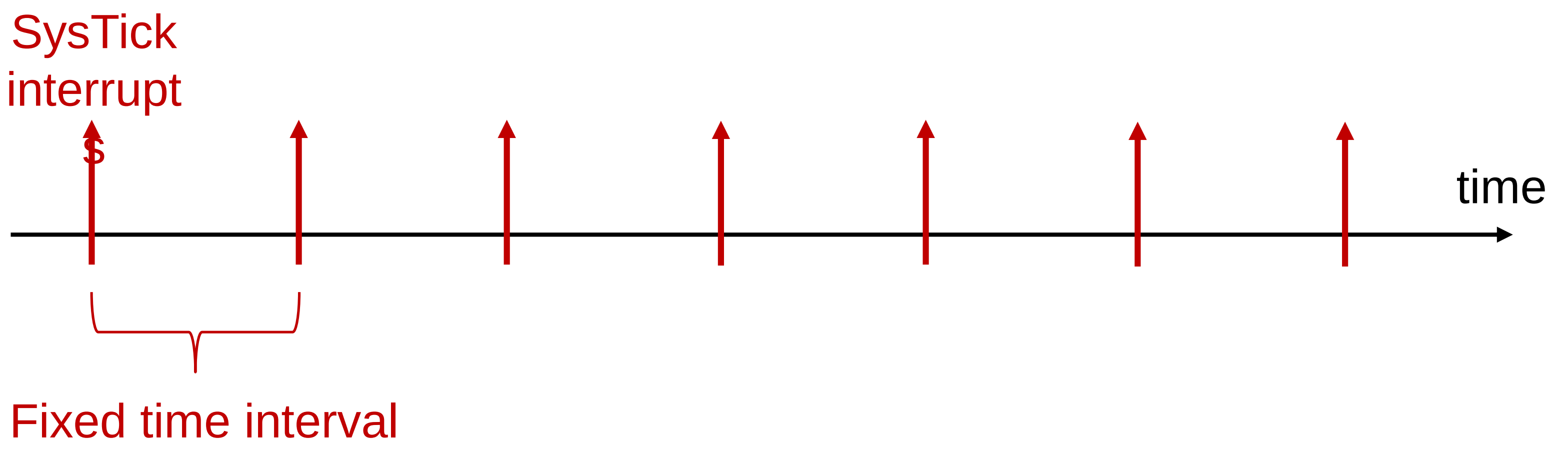
Timer/Counter Peripheral Introduction



- Common peripheral for microcontrollers
- Based on pre-settable 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
    - Stop!

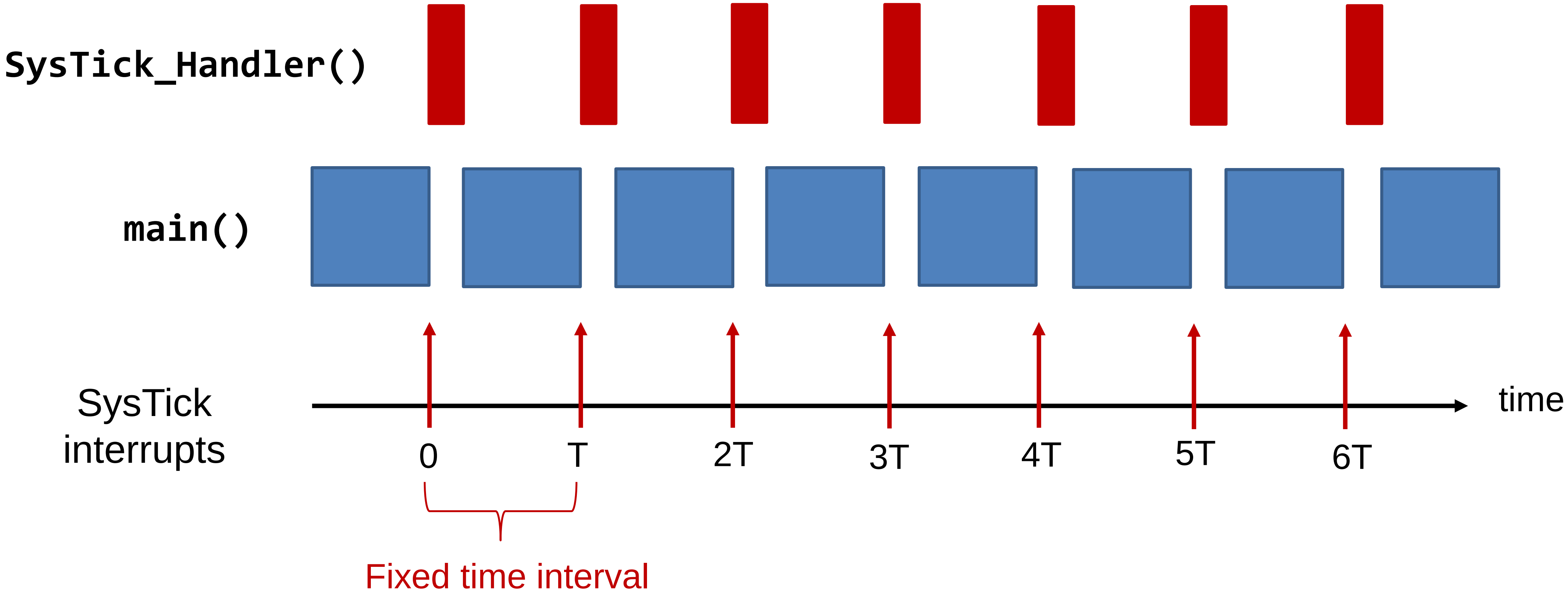
System Timer (SysTick) – *(SysTick Slides are based on the slides of Dr. Yifeng Zhu)*

- Generate SysTick interrupts at a fixed time interval



- Example Usages:
  - Measuring time elapsed, such as time delay function
  - Executing tasks periodically, such as periodic polling, and OS CPU scheduling

System Timer (SysTick)



System Timer (SysTick)

- System timer is a standard hardware component built into ARM Cortex-M.
- This hardware periodically forces the processor to execute the following ISR:

```
void SysTick_Handler(void){
   ...
}
```

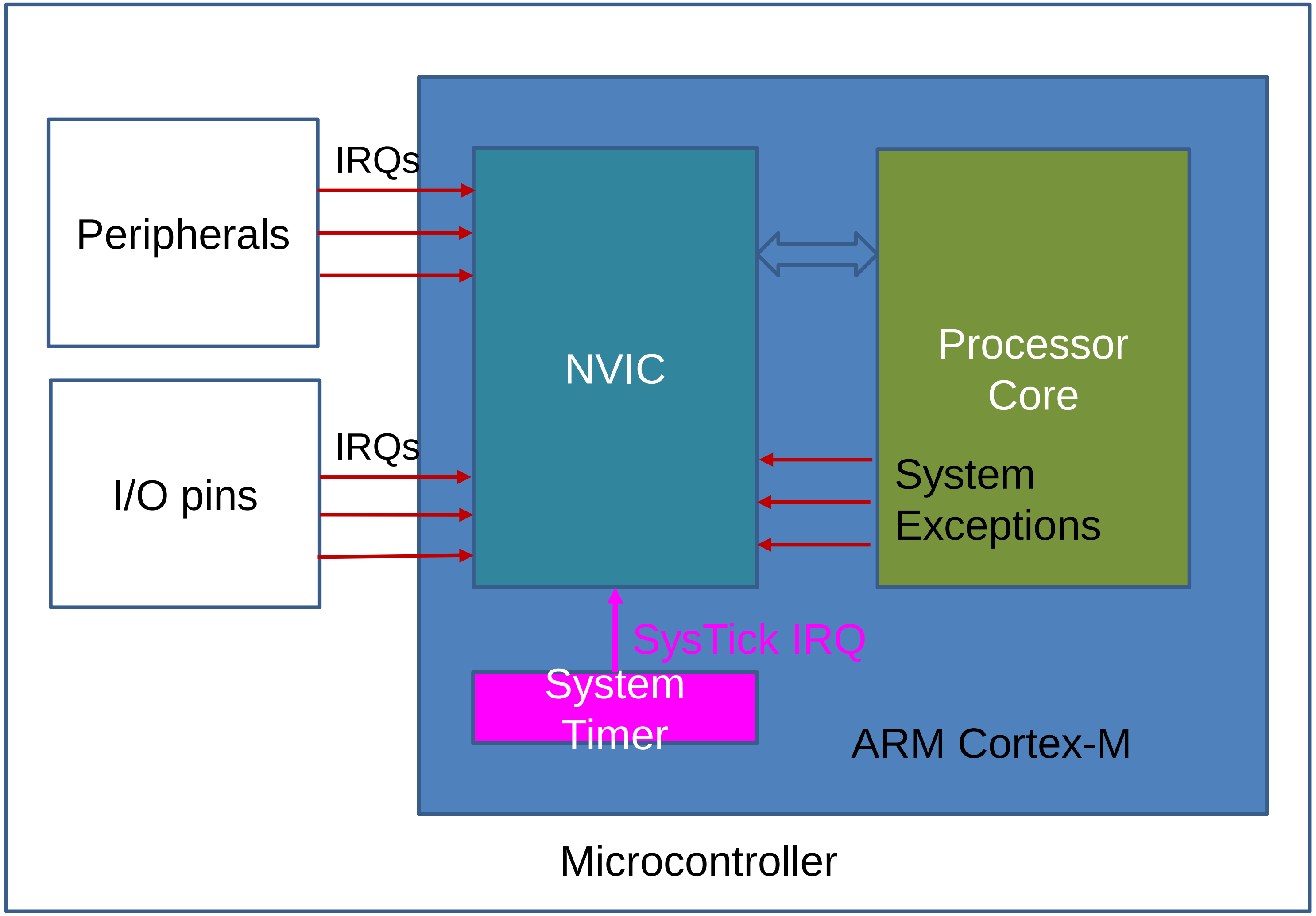
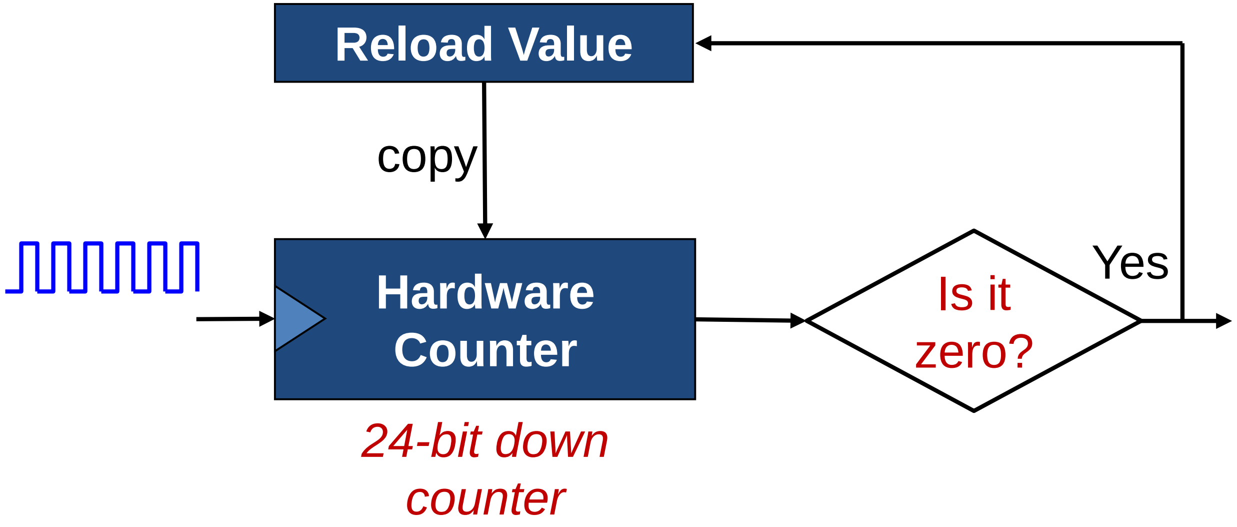
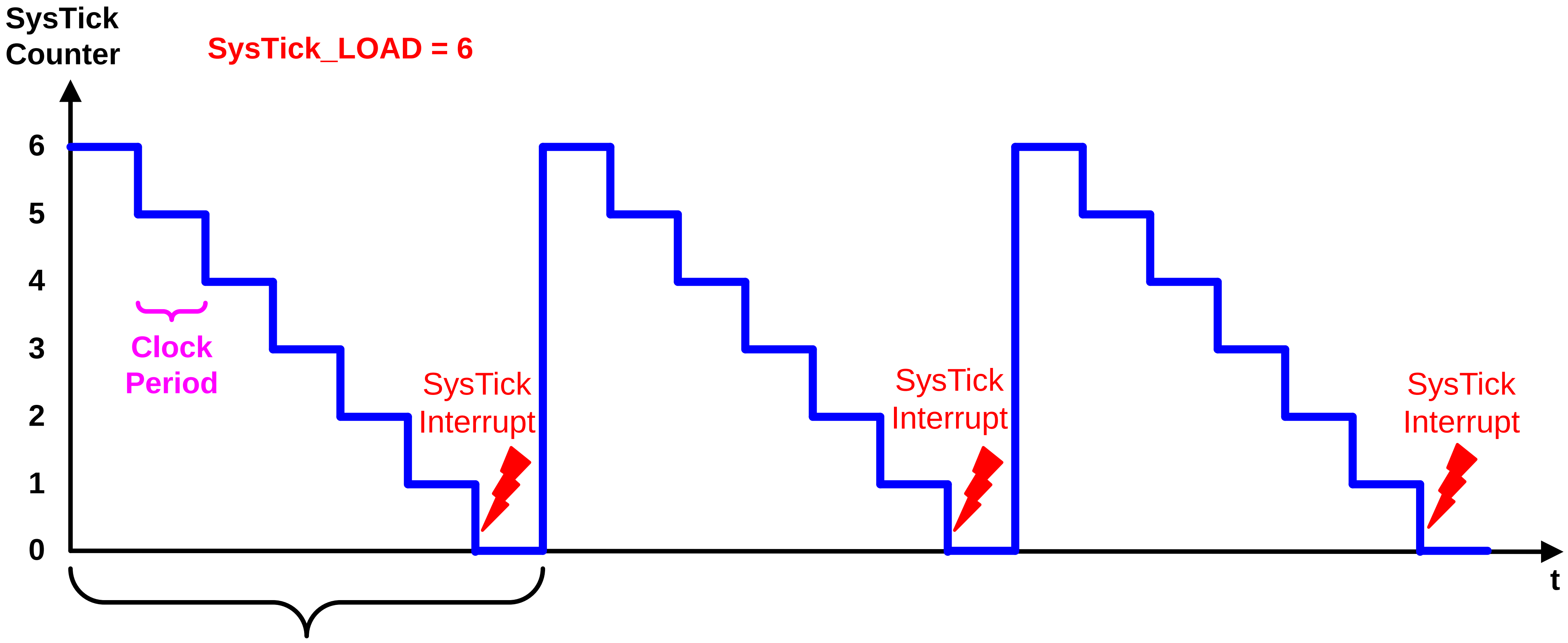


Diagram of System Timer (SysTick)

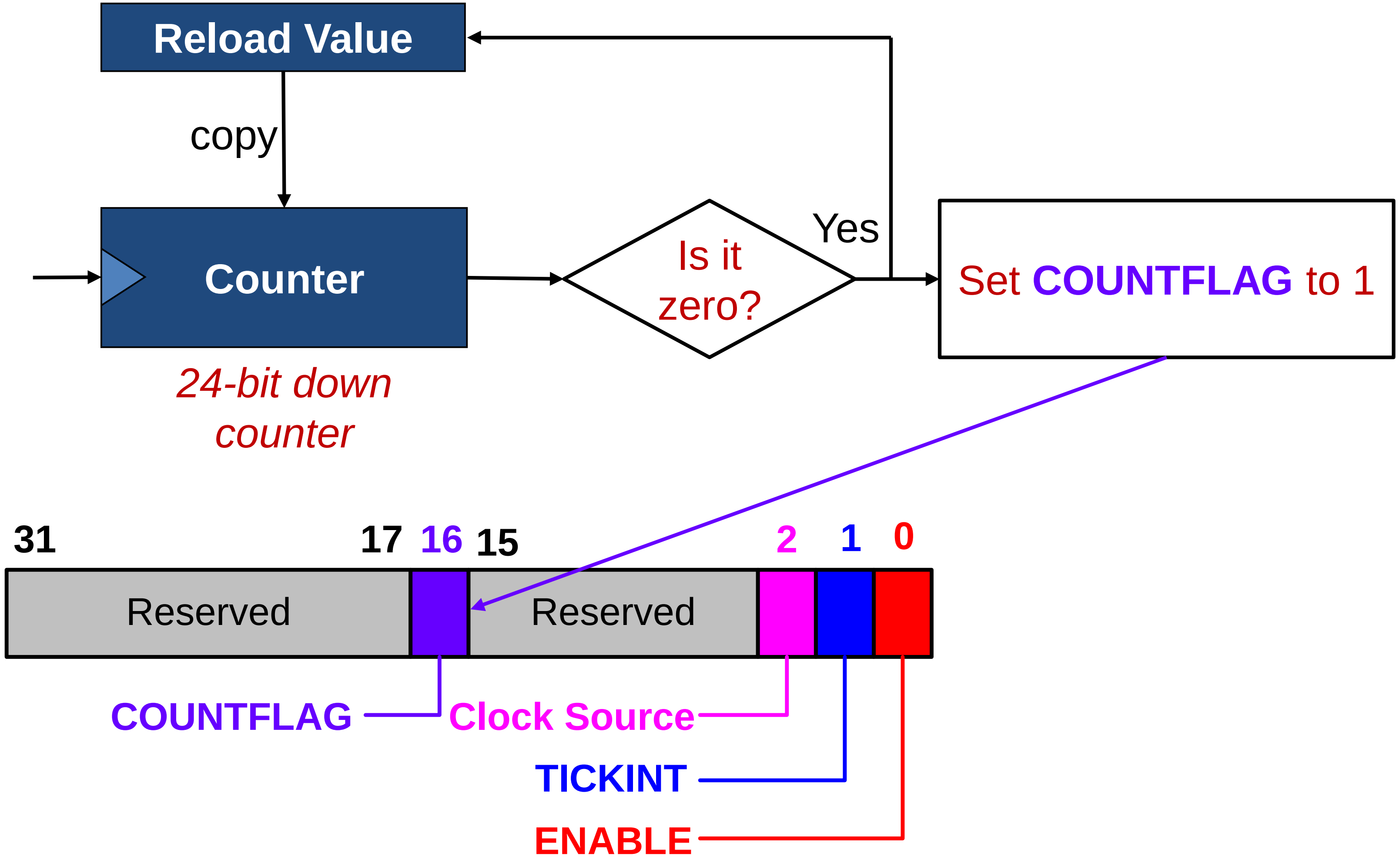


System Timer



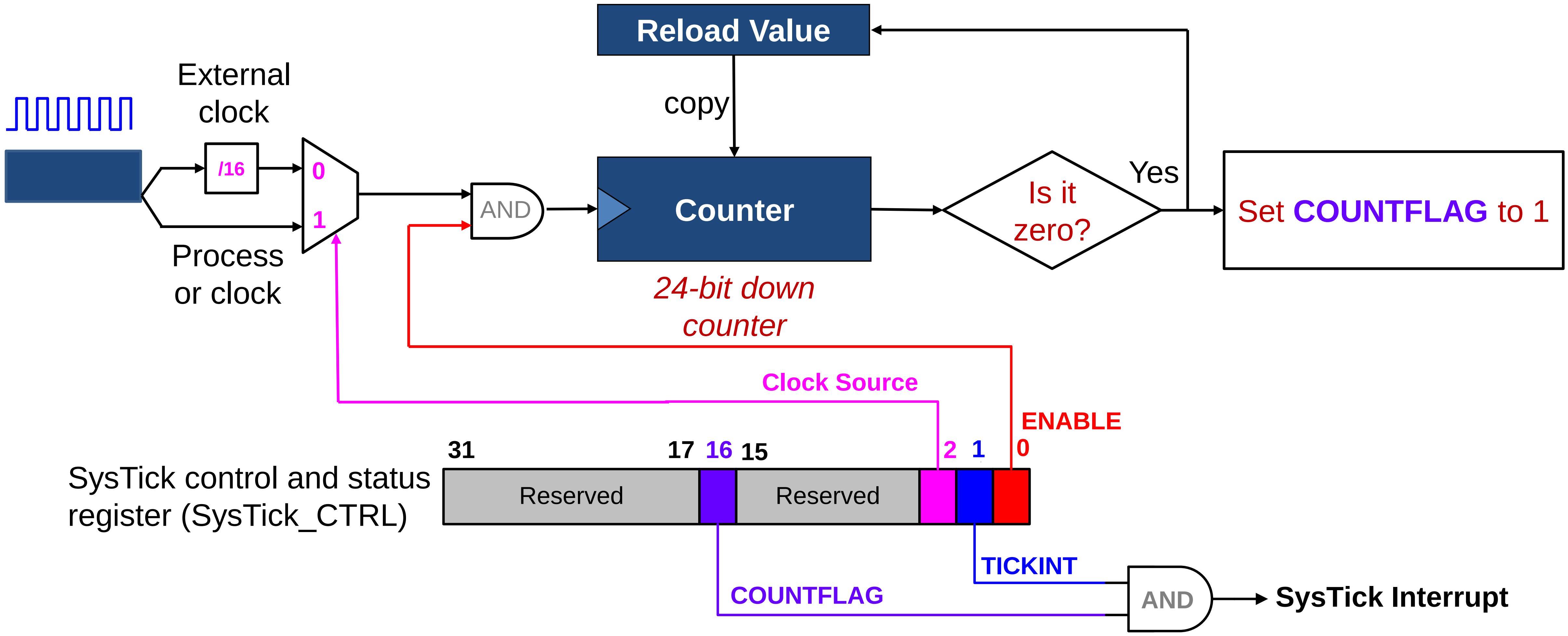
SysTick Interrupt Time Period = (SysTick_LOAD + 1) × Clock Period = 7 × Clock Period

Diagram of System Timer (SysTick)



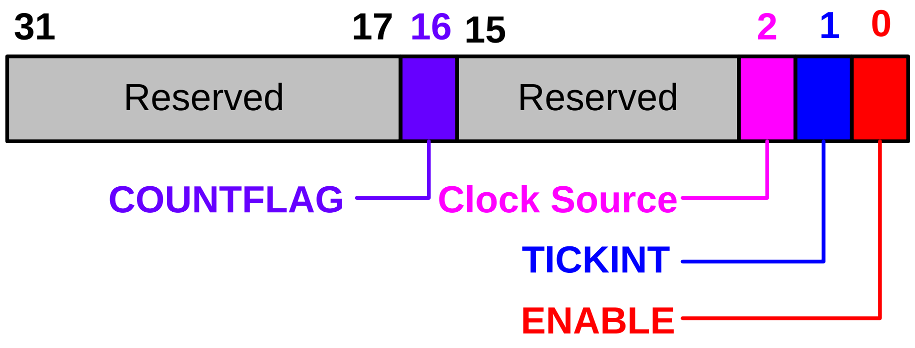
SysTick control and status register (SysTick_CTRL)

Diagram of System Timer (SysTick)

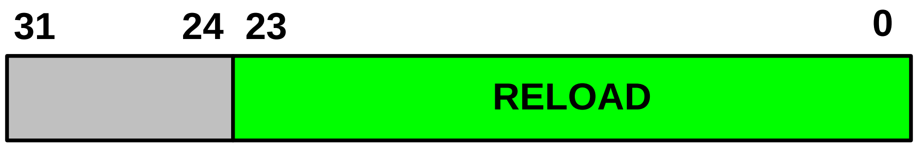


Registers of System Timer

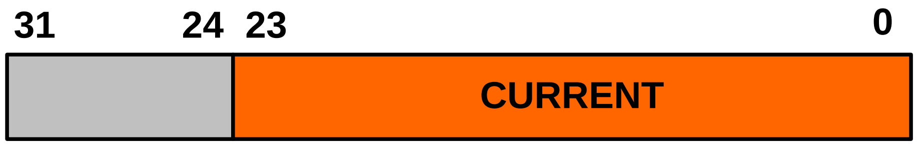
SysTick control and status register (SysTick_CTRL)



SysTick reload value register (SysTick_LOAD)

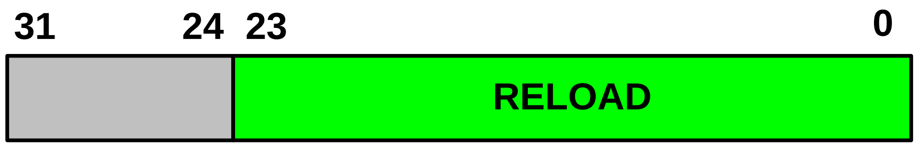


SysTick current value register (SysTick_VAL)



Registers of System Timer

SysTick reload value register (SysTick_LOAD)



- 24 bits, maximum value `0x00FF.FFFF` (16,777,215)
- Counter counts down from RELOAD value to 0.
- Writing RELOAD to 0 disables SysTick, independently of TICKINT
- Time interval between two SysTick interrupts

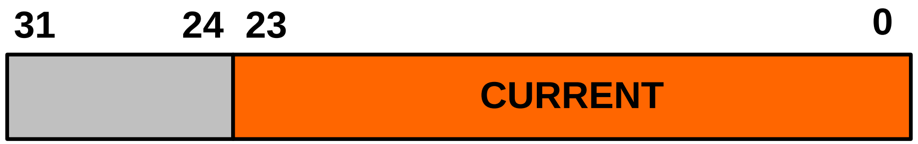
`Interval = (RELOAD + 1) × Source_Clock_Period`

- If `100` clock periods between two SysTick interrupts

`RELOAD = 99`

Registers of System Timer

SysTick current value register (SysTick_VAL)



- Reading it returns the current value of the counter
- When it transits from `1` to `0`, it generates an interrupt
- Writing to SysTick_VAL clears the counter and COUNTFLAG to zero
  - Cause the counter to reload on the next timer clock
  - But, does not trigger an SysTick interrupt
- It has random value on reset.
  - Always clear it before enabling the timer

Calculating Reload Value

- Suppose clock source = `80MHz`
- Goal: SysTick Interval = `10ms`
- What is RELOAD value?

$$Reload = \frac{10\ ms}{Clock\ Period} - 1$$

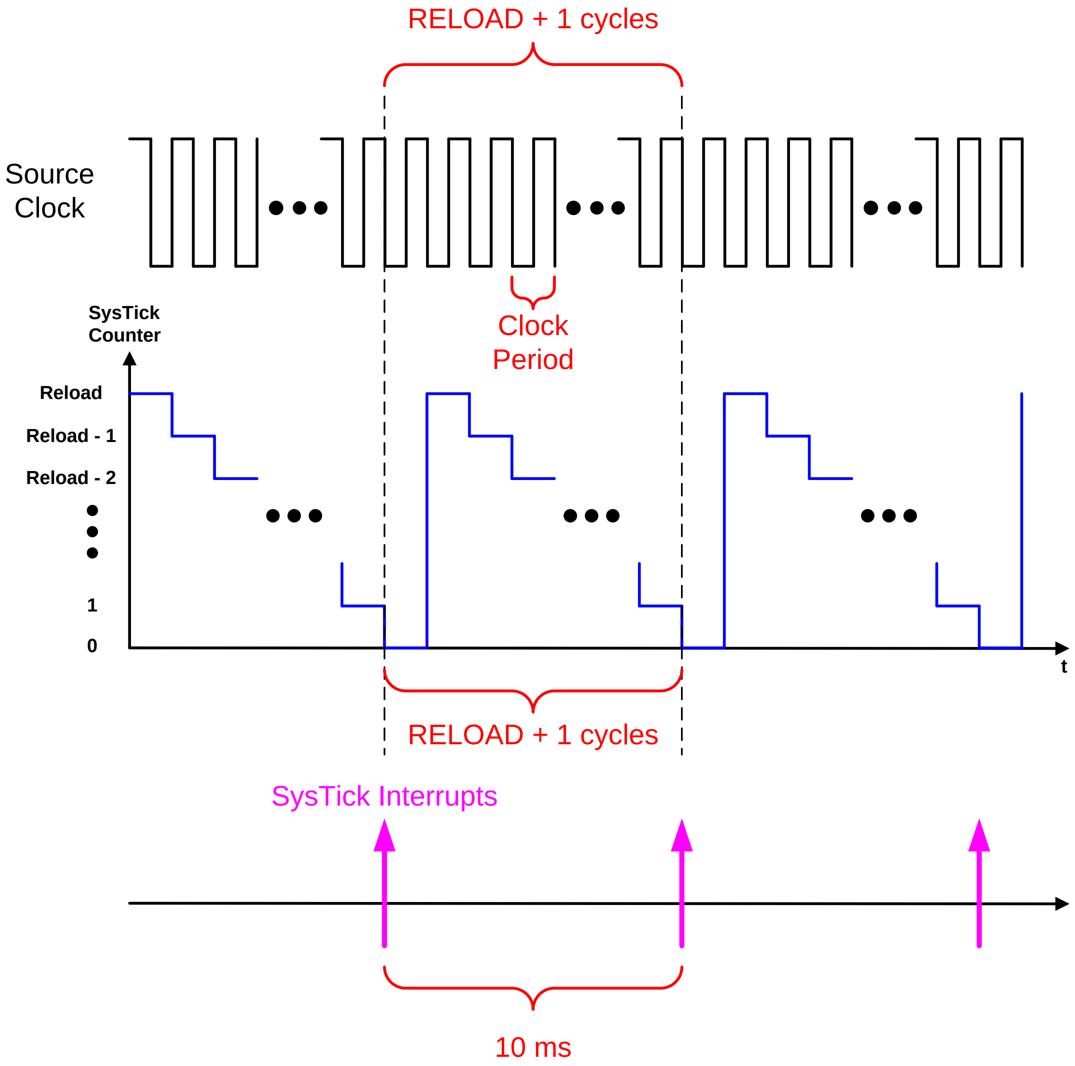
$$= 10ms \times Clock\ Frequency\ - 1$$

$$= 10ms \times 80MHz - 1$$

$$= 10 \times 10^{-3} \times 80 \times 10^{6} - 1$$

$$= 800000 - 1$$

$$= 799999$$



Example Code (Textbook page 189)

```
void Init_SysTick (void) {
    SysTick->CTRL = 0;            // Disable SysTick
    SysTick->LOAD = 0x13FFFF;    // Set reload register to get 1s interrupts clk=20971520
    NVIC_SetPriority(SysTick_IRQn, 3);
    SysTick->VAL = 0;             // Reset the SysTick counter value
    SysTick->CTRL |= SysTick_CTRL_TICKINT_Msk | SysTick_CTRL_ENABLE_Msk;
}

void SysTick_Handler() {
static int n=0;
Control_RGB_LEDs(n&1,n&1,n&1);
n++;
}
```