SysTick Timer | STM32F1xx

Overview

- **SysTick** is a 24-bit **down counter** integrated into the **Cortex-M3 core**.
- **Independent** of other peripherals.
- Can generate **interrupts** on reaching 0.
- Reloads automatically to the **value in the RELOAD register**.

Applications:

- Delay generation (ms, µs, s)
- Time interval measurement
- RTOS tick (scheduler)
- PWM measurement (software-based)

Clock Sources

SysTick can count based on two sources:

CLKSOURCE	Description
0 (external/processor clock / 8)	Slower, low-power usage
1 (processor clock / HCLK)	Fast, same as CPU clock

Default after reset: HCLK/8

Registers

SysTick has **four 32-bit registers** at **0xE000E010 – 0xE000E01C**.

Register	Offset	Bits	Description
`SYST_CSR`	0xE010	0-16	Control and status
`SYST_RVR`	0xE014	0-23	Reload value (24-bit)
`SYST_CVR`	0xE018	0-23	Current counter value
`SYST_CALIB`	0xE01C	0-23	Calibration (optional)

Control and Status Register (SYST_CSR)

Bit	Name	Description
16	COUNTFLAG	1 if counter reached 0 since last read
2	CLKSOURCE	0 = HCLK/8, 1 = HCLK
1	TICKINT	1 = interrupt on count to 0
0	ENABLE	1 = enable counter

Example: Enable SysTick, HCLK, interrupt:

```
SysTick->CTRL = (1 << 2) | (1 << 1) | (1 << 0);
```

Reload Value Register (SYST_RVR)

- 24-bit value: max = 0xFFFFFF = 16,777,215
- Defines the starting count (timer reload)
- When counter reaches 0 → fires interrupt, reloads

Formula:

$$T = \frac{RVR + 1}{f_{CLK}}$$

Example: 1 ms tick at 72 MHz HCLK

```
SysTick->LOAD = (72000000/1000) - 1; // 1ms
```

Current Value Register (SYST_CVR)

- 24-bit down-counter
- Read: current value
- Write any value: **resets counter to 0**
- Automatic reload from `LOAD` when counting to 0

Calibration Register (SYST_CALIB)

- Factory-calibrated for 10 ms tick (optional)
- Useful for delay generation without using `SystemCoreClock`

SysTick Operation Modes

1. **Polling (no interrupt):**
   - Set `ENABLE` bit
   - Read `COUNTFLAG` to detect timer rollover
2. **Interrupt Mode:**
   - Set `TICKINT = 1`
   - Implement `SysTick_Handler()` ISR

Bare-Metal Initialization Examples

1ms Interrupt Tick (HCLK = 72 MHz)

```
#include "stm32f1xx.h"

volatile uint32_t msTicks = 0;

void SysTick_Handler(void) {
    msTicks++; // increments every 1 ms
}

void SysTick_Init(void)
{
    SysTick->LOAD = (72000000/1000) - 1; // 1ms interval
    SysTick->VAL = 0;                    // Reset current counter
    SysTick->CTRL = 0x07;                // ENABLE | TICKINT | CLKSOURCE
}
```

Delay Using SysTick Interrupt

```
void delay_ms(uint32_t ms)
{
    uint32_t start = msTicks;
    while ((msTicks - start) < ms);
}
```

Polling Mode (No Interrupt)

```
void delay_us(uint32_t us)
{
    SysTick->LOAD = (72 - 1); // 1µs tick at 72 MHz
    SysTick->VAL = 0;
    SysTick->CTRL = (1 << 0) | (1 << 2); // ENABLE + CLKSOURCE
    for(uint32_t i=0; i<us; i++)
        while((SysTick->CTRL & (1<<16)) == 0); // wait COUNTFLAG
    SysTick->CTRL = 0; // disable
}
```

Advantages of SysTick

- **Independent of peripherals**
- **Simple to configure**
- **Supports both interrupt and polling**
- **Can create precise time intervals**

Tips & Notes

- **24-bit counter max:**

$$MaxTime = \frac{2^{24} - 1}{f_{CLK}}$$

  At 72 MHz: ~0.233 s. Use software counter for longer delays.

- Always clear `SYST_CVR` before enabling
- Combine with `volatile` variable to track time
- For longer delays: increment counter in ISR

Summary Table

Feature	Register	Bits	Usage
Enable	CSR	ENABLE	start counting
Interrupt	CSR	TICKINT	fire ISR
Clock	CSR	CLKSOURCE	HCLK or HCLK/8
Reload	RVR	23:0	starting count value
Current	CVR	23:0	read/reset counter
Countflag	CSR	16	set when counter reaches 0
