TUTORIAL: SysTick Interrupt

Timer and Time Delay

Name: Gyeonheal An ID: 21900416

I. Introduction

In this tutorial, we will learn how to use SysTick interrupt. We will create functions to count up numbers at a constant rate using SysTick.

The objectives of this tutorial are how to

- Configure SysTick with NVIC
- Create your own functions for configuration of interrupts

Hardware

NUCLEO -F411RE

Software

Keil uVision IDE, CMSIS, EC_HAL

Documentation

STM32 Reference Manual

II. Basics of SysTick

A. Register List

List of SysTick registers for this tutorial. [**Programming Manual** ch4.3, ch10.2]

Type	Register Name	Description
SYSCFG_	SysTick_CTRL	Clock Control and Status
	SysTick_LOAD	Reload Value
	SysTick_VAL	Current Value

Schematic

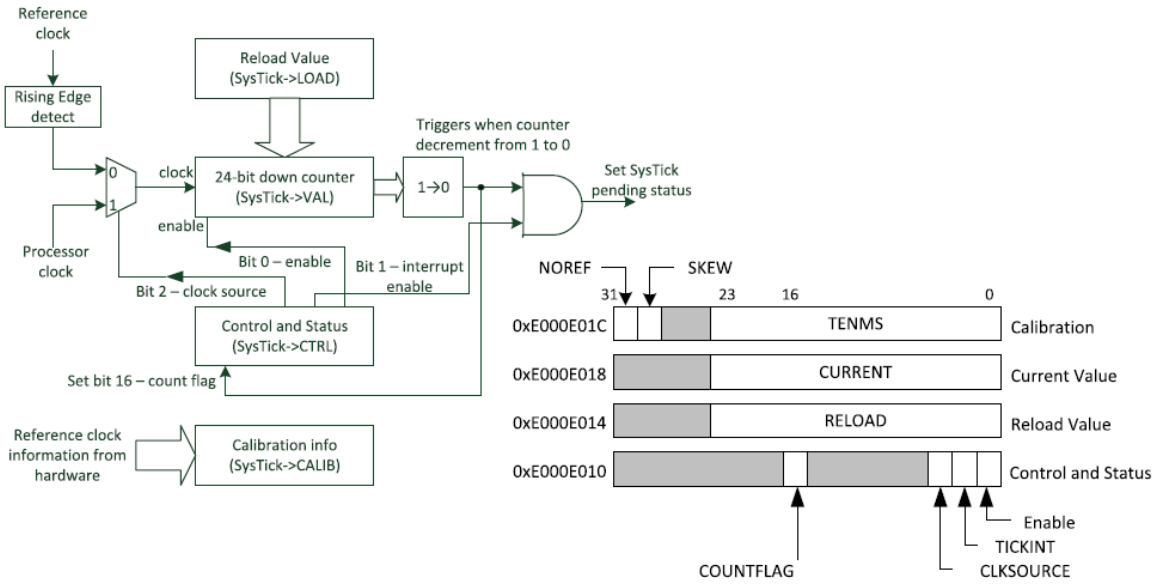


FIGURE 9.15

A simplified block diagram of SysTick timer

B. Register Setting

(RCC system clock)

1. PLL, HCLK= 84MHz

(System Tick Configuration)

1. Disable SysTick Timer
   - SysTick->CTRL ENABLE=0

2. Choose clock signal: System clock or ref. clock(STCLK)
   - SysTick->CTRL CLKSOURCE = 0 or 1

3. Choose to use Tick Interrupt (timer goes 1->0)
   - SysTick->CTRL TICKINT = 0 or 1

4. Write reload Counting value (24-bit)
   - SysTick->LOAD RELOAD = (value-1)

5. Start SysTick Timer
   - SysTick->CTRL ENABLE=1

6. (option) Read or Clear current counting value
   - Read from SysTick->VAL
   - Write clears value

(NVIC Configuration)

- NVIC SysTick Interrupt priority
- NVIC SysTick Enable

III. Tutorial

A. Register Configuration

Fill in the table

Port/Pin	Description	Register setting
SYSTICK	Disable SysTick Timer	**SysTick -> CTRL |= 0;**
	Choose clock signal: System clock	**SysTick->CTRL |= 1UL << 2UL;**
	Write reload Counting value Make it 1ms / 1s	**SysTick->LOAD = 1ms * 84*10^6 - 1 = 83999;**
	Start SysTick Timer	**SysTick->CTRL |= 1UL << 1UL**
	Read from SysTick value	**val = SysTick->VAL**
	Clear from SysTick value	**SysTick->VAL = 0**
NVIC	Set SysTick Interrupt priority =16	NVIC_SetPriority(SysTick_IRQn, 15);
	NVIC SysTick Enable	NVIC_EnableIRQ(SysTick_IRQn);

B. Programming

Procedure

- Create a new folder '**EC/Tutorial/TU_ SysTick /**'
- Open the program 'Keil uVision5' and create a new project.
- Name the project as '**TU_ SysTick**'.
- Create a new item called '**TU_SysTick.c**'
- Use the given source code Click here to download
- Fill in the empty spaces in the code.
- Run the program and check your result.
- Your tutorial report must be submitted to LMS

Exercise

- Create a simple program that turns LED on/off at 1 second period.
- Set the SysTick to be 1msec.
- You can define the necessary timer or time wait function in the handler function of

  void Systick_Handler(void)

  Example:

  ```
  void SysTick_Handler(void){ msTicks++;}
  ```

- There are other methods for making time delay functions. Check here for examples
- You can check some sample codes here

```c
#include "stm32f411xe.h"
#include "ecRCC.h"
#include "ecGPIO.h"

#define MCU_CLK_PLL 16000000  // 84000000
#define MCU_CLK_HSI 16000000

volatile uint32_t msTicks = 0;
volatile uint32_t curTicks;

void setup(void);
void LED_toggle(void);
void SysTick_Handler(void);

int main(void) {

// System CLOCK, GPIO Initialiization ----------------------------------------
  setup();

// SysTick Initialiization ---------------------------------------------------
  //  SysTick Control and Status Register
  SysTick->CTRL = 0;        // Disable SysTick IRQ and SysTick Counter

  // Select processor clock
  // 1 = processor clock;  0 = external clock
  SysTick->CTRL |= 1UL << 2UL;

  // uint32_t MCU_CLK=EC_SYSTEM_CLK
  // SysTick Reload Value Register
  SysTick->LOAD = (MCU_CLK_PLL / 1000) - 1;         // 1ms

  // Clear SysTick Current Value
  SysTick->VAL = 0;

  // Enables SysTick exception request
  // 1 = counting down to zero asserts the SysTick exception request
  SysTick->CTRL |= 1UL << 1UL;

  // Enable SysTick IRQ and SysTick Timer
  SysTick->CTRL |= 1UL << 0UL;

  NVIC_SetPriority(SysTick_IRQn, 16);   // Set Priority to 1
  NVIC_EnableIRQ(SysTick_IRQn);     // Enable interrupt in NVIC


// While loop ----------------------------------------------------------
  msTicks = 0;

  while(1){
    curTicks = msTicks;
    while ((msTicks - curTicks) < 1000);
    msTicks = 0;
    LED_toggle();
  }
}
void LED_toggle(void){
  static unsigned int out = 0;
  out ^= 1UL;
  GPIO_write(GPIOA, LED_PIN, out);
}

void SysTick_Handler(void){
  msTicks++;
}
 void setup(void)
{
  RCC_PLL_init();                           // System Clock = 84MHz
  GPIO_init(GPIOA, LED_PIN, OUTPUT);    // calls RCC_GPIOA_enable()
}
```

Appendix

[See here for MCU resources]

1. Pin Configuration of NUCLE-F401RE

Figure 18. NUCLEO-F401RE

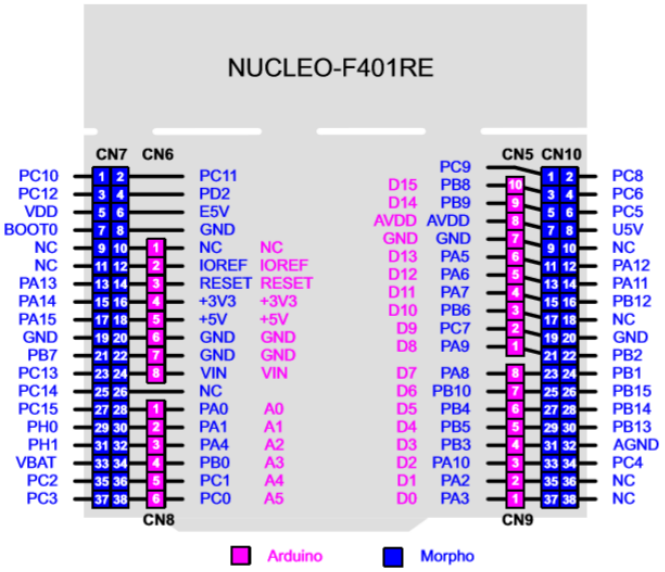
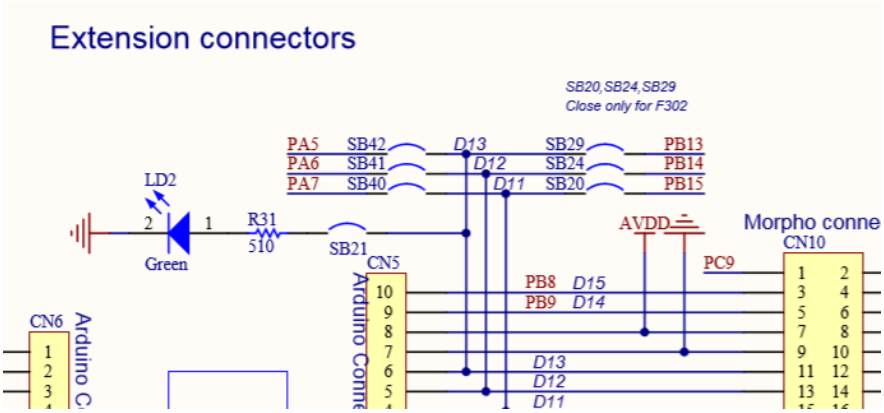


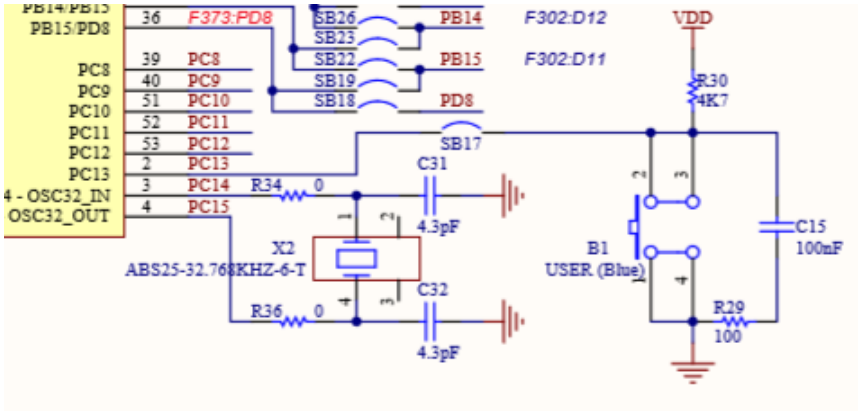
Table 29. ST morpho connector on NUCLEO-F401RE, NUCLEO-F411RE, NUCLEO-F446RE

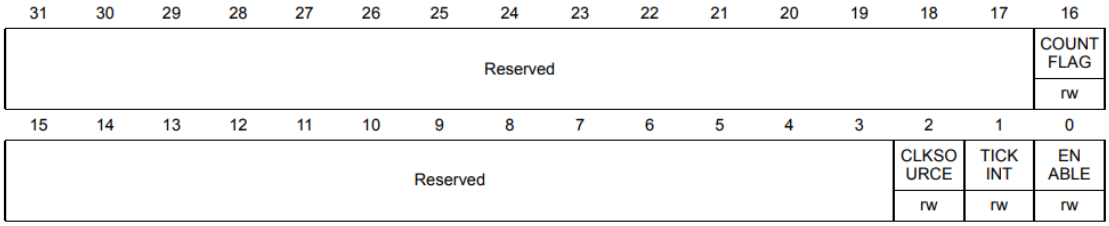
CN7 odd pins		CN7 even pins		CN10 odd pins		CN10 even pins	
Pin	Name	Name	Pin	Pin	Name	Name	Pin
1	PC10	PC11	2	1	PC9	PC8	2
3	PC12	PD2	4	3	PB8	PC6	4
5	VDD	E5V	6	5	PB9	PC5	6
7	BOOT0(1)	GND	8	7	AVDD	U5V(2)	8
9	-	-	10	9	GND	-	10
11	-	IOREF	12	11	PA5	PA12	12
13	PA13(3)	RESET	14	13	PA6	PA11	14
15	PA14(3)	+3.3V	16	15	PA7	PB12	16
17	PA15	+5V	18	17	PB6	-	18
19	GND	GND	20	19	PC7	GND	20
21	PB7	GND	22	21	PA9	PB2	22
23	PC13	VIN	24	23	PA8	PB1	24
25	PC14	-	26	25	PB10	PB15	26
27	PC15	PA0	28	27	PB4	PB14	28
29	PH0	PA1	30	29	PB5	PB13	30
31	PH1	PA4	32	31	PB3	AGND	32
33	VBAT	PB0	34	33	PA10	PC4	34
35	PC2	PC1 or PB9(4)	36	35	PA2	-	36
37	PC3	PC0 or PB8(4)	38	37	PA3	-	38

1. Default state of BOOT0 is 0. It can be set to 1 when a jumper is on pin5-7 of CN7.Two unused jumpers are available on CN11 and CN12 (bottom side of the board).
2. U5V is 5 V power from ST-LINK/V2-1 USB connector and it rises before +5V.
3. PA13 and PA14 share with SWD signals connected to ST-LINK/V2-1, it is not recommend to use them as IO pins if ST-LINK part is not cut.
4. Refer to *Table 10: Solder bridges* for details.

2. LED/Button Circuit Diagram







Bits 31:17 Reserved, must be kept cleared.

Bit 16 COUNTFLAG:
Returns 1 if timer counted to 0 since last time this was read.

Bits 15:3 Reserved, must be kept cleared.

Bit 2 **CLKSOURCE:** Clock source selection
Selects the clock source.
0: AHB/8
1: Processor clock (AHB)

Bit 1 **TICKINT**: SysTick exception request enable
0: Counting down to zero does not assert the SysTick exception request
1: Counting down to zero to asserts the SysTick exception request.
Note: Software can use COUNTFLAG to determine if SysTick has ever counted to zero.

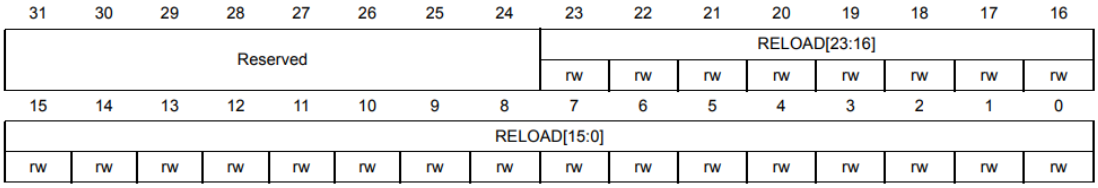
Bit 0 **ENABLE**: Counter enable
Enables the counter. When ENABLE is set to 1, the counter loads the RELOAD value from the LOAD register and then counts down. On reaching 0, it sets the COUNTFLAG to 1 and optionally asserts the SysTick depending on the value of TICKINT. It then loads the RELOAD value again, and begins counting.
0: Counter disabled
1: Counter enabled

4.5.2 SysTick reload value register (STK_LOAD)

Address offset: 0x04

Reset value: 0x0000 0000

Required privilege: Privileged



Bits 31:24 Reserved, must be kept cleared.

Bits 23:0 **RELOAD:** RELOAD value
The LOAD register specifies the start value to load into the STK_VAL register when the counter is enabled and when it reaches 0.
Calculating the RELOAD value
The RELOAD value can be any value in the range 0x00000001-0x00FFFFFF. A start value of 0 is possible, but has no effect because the SysTick exception request and COUNTFLAG are activated when counting from 1 to 0.
The RELOAD value is calculated according to its use:

- To generate a multi-shot timer with a period of N processor clock cycles, use a RELOAD value of N-1. For example, if the SysTick interrupt is required every 100 clock pulses, set RELOAD to 99.
- To deliver a single SysTick interrupt after a delay of N processor clock cycles, use a RELOAD of value N. For example, if a SysTick interrupt is required after 100 clock pulses, set RELOAD to 99.