PreLAB: SysTick

Name:leejeayong

ID:22000561

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 the configuration of interrupts

Hardware

- NUCLEO -F411RE

Software

- VS code, 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]

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)

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

Schematic

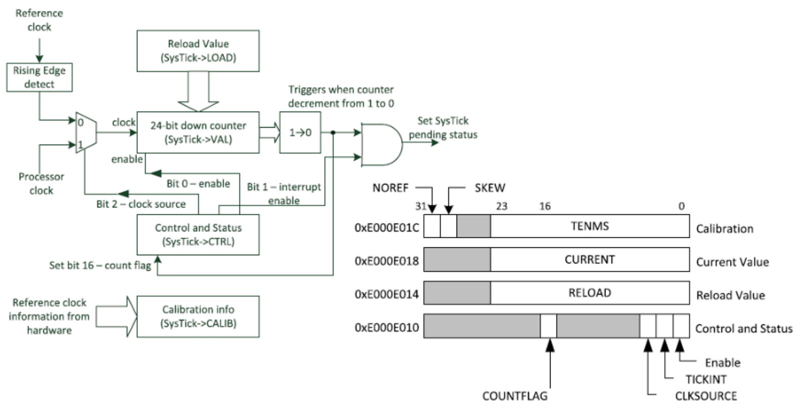


FIGURE 9.15

A simplified block diagram of SysTick timer

Figure 1: Register List

`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)

1. NVIC SysTick Interrupt priority
2. NVIC SysTick Enable

III. Tutorial

A. Programming

This is an example code for turning the LED on/off with the button input trigger with a wait function.

Procedure

- Name the project as '**TU__SysTick**' by creating a new folder as '**tutorial/TU__SysTick**'
- Download the header library files and save under `include\`.
  - `ecSysTick2_student. ecSysTick2_student.c`: Click here to download
  - Rename the files as `ecSysTick2. ecSysTick2.c`
- Download the template code
  - `TU_SysTick_student.c` : Click here to download
- This is an example code for turning LED on/off with the button input trigger with a wait function.
- Fill in the empty spaces in the code.
- Run the program and check your result.
- Your tutorial report must be submitted to the LMS
- This is a sample program that turns LED on/off at 1 second period using SysTick

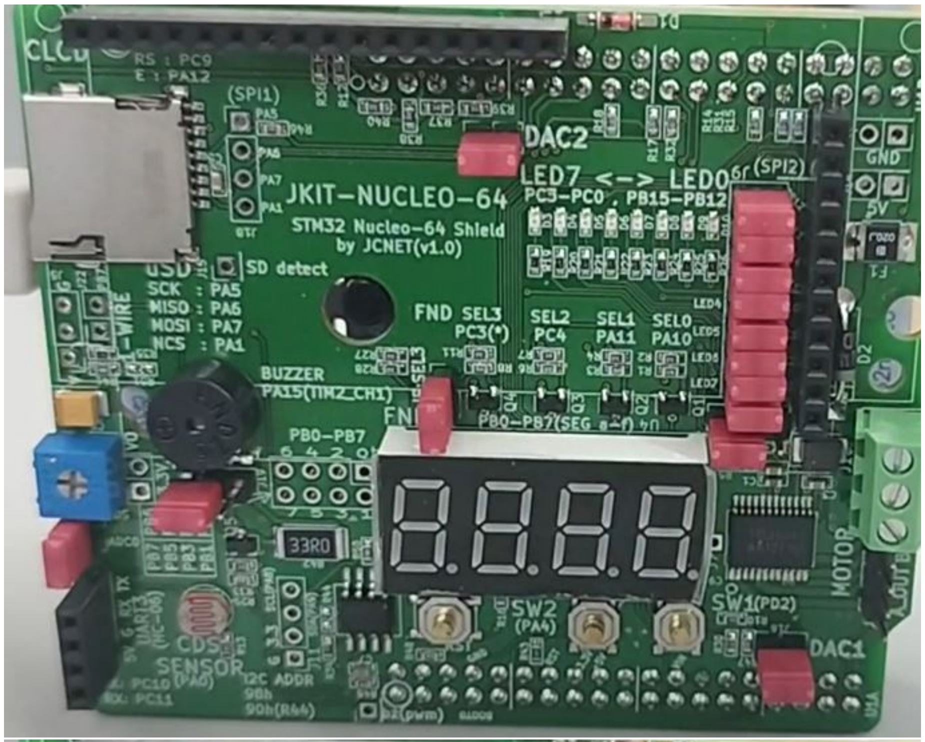
Example Code

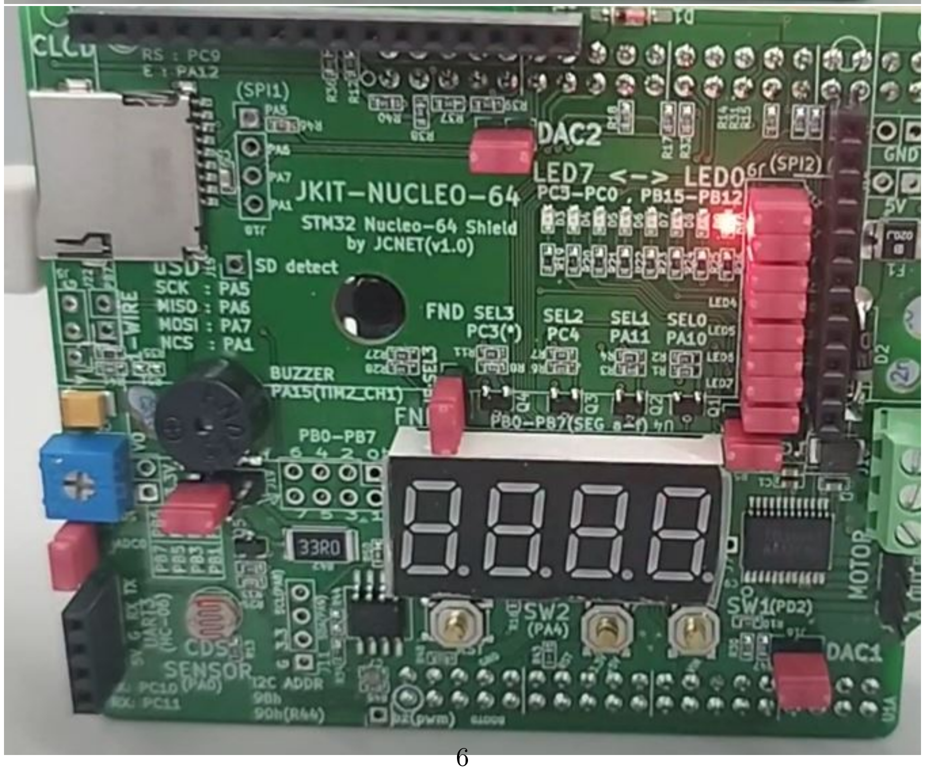
- Understand the code definition for void SysTick_init() : in `ecSysTick2.h`
- Read the code definition for void delay_ms( ) in `ecSysTick2.h`
- You can modify previous LAB code to include delay_ms()

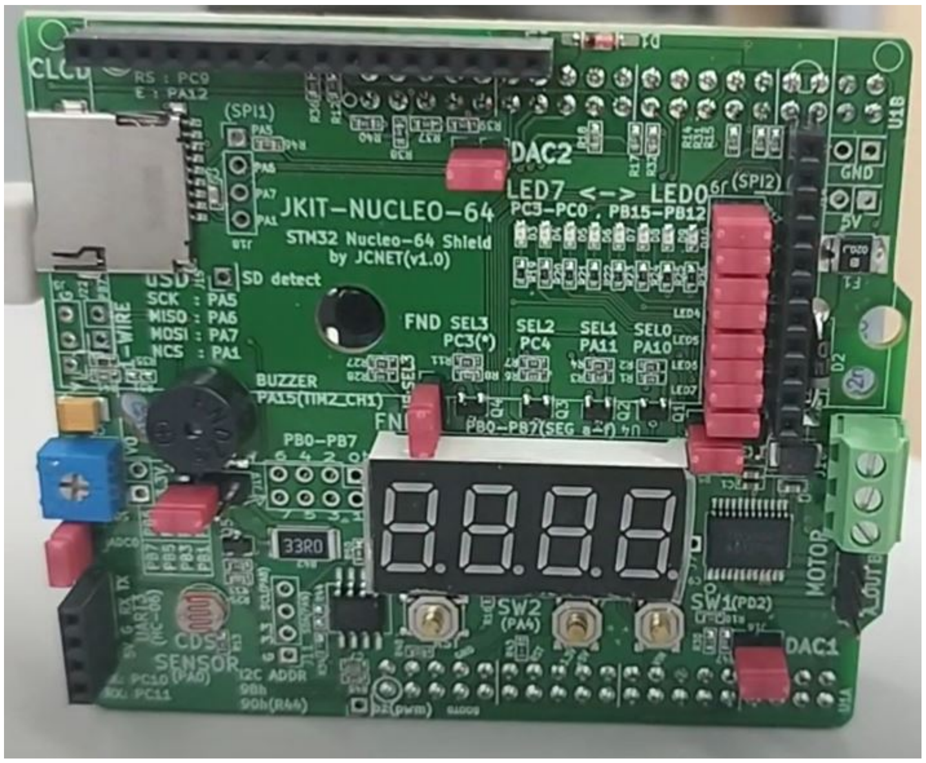
```
/**
****************************************************************************
* @author  SSSLAB
* @Mod       2025-9-25 by YKKIM
* @brief   Embedded Controller:  Tutorial ___
*leejeayong
****************************************************************************
*/



//#include "ecSTM32F4v2.h"
#include "ecRCC2.h"
#include "ecGPIO2.h"

#define LED_PIN   PB_12         //EVAL board JKIT
#define BUTTON_PIN PA_4         //EVAL board JKIT

void LED_toggle(PinName_t pinName);
void EXTI_init_tutorial(PinName_t pinName);

// Initialiization
void setup(void)
{
    RCC_PLL_init();                         // System Clock = 84MHz
    // Initialize GPIOB_12 for Output
    GPIO_init(LED_PIN, OUTPUT);    // LED for EVAL board
    // Initialize GPIOA_4 for Input Button
    GPIO_init(BUTTON_PIN, INPUT);  // OUTPUT for EVAL borad
    EXTI_init_tutorial(PA_4);

}

// MAIN  ----------------------------------------
int main(void) {

    setup();

    while (1);
}
```

```c
// EXTI Initialiization ------------------------------------------------------
void EXTI_init_tutorial(PinName_t pinName)
{
    GPIO_TypeDef *Port;
    unsigned int pin;
    ecPinmap(pinName, &Port, &pin);

    // SYSCFG peripheral clock enable
    RCC->APB2ENR |= RCC_APB2ENR_SYSCFGEN;

    // Connect External Line to the GPIO
    // Button: PA_4 -> EXTICR2(EXTI4)
    SYSCFG->EXTICR[1] &= ~SYSCFG_EXTICR2_EXTI4;
    SYSCFG->EXTICR[1] |= SYSCFG_EXTICR2_EXTI4_PA;

    // Falling trigger enable (Button: pull-up)
    EXTI->FTSR |= (1UL << 4);

    // Unmask (Enable) EXT interrupt
    EXTI->IMR |= (1UL << 4);

    // Interrupt IRQn, Priority
    NVIC_SetPriority(EXTI4_IRQn, 0);        // Set EXTI priority as 0
    NVIC_EnableIRQ(EXTI4_IRQn);             // Enable EXTI
}

void EXTI4_IRQHandler(void) {
    if ((EXTI->PR & EXTI_PR_PR4) == EXTI_PR_PR4) {
        LED_toggle(LED_PIN);
        EXTI->PR |= EXTI_PR_PR4; // cleared by writing '1'
    }
}


void LED_toggle(PinName_t pinName){
    GPIO_TypeDef *Port;
    unsigned int pin;
    ecPinmap(pinName,&Port,&pin);
    // YOUR CODE GOES HERE
    GPIO_write(pinName, !GPIO_read(pinName));
}
```

results


JKIT-NUCLEO-64
STM32 Nucleo-64 Shield
by JCNET(v1.0)
LED7 <-> LED0
PC3-PC0 , PB15-PB12
DAC2
SD detect
FND SEL3 PC3(*)
SEL2 PC4
SEL1 PA11
SEL0 PA10
BUZZER PA15(TIM2_CH1)
PB0-PB7(SEG a-f)
SW2 (PA4)
SW1(PD2)
DAC1
MOTOR
CDS SENSOR
I2C ADDR


JKIT-NUCLEO-64
STM32 Nucleo-64 Shield
by JCNET(v1.0)
LED7 <-> LED0
PC3-PC0 , PB15-PB12
DAC2
SD detect
FND SEL3 PC3(*)
SEL2 PC4
SEL1 PA11
SEL0 PA10
BUZZER PA15(TIM2_CH1)
PB0-PB7(SEG a-f)
SW2 (PA4)
SW1(PD2)
DAC1
MOTOR
CDS SENSOR
I2C ADDR



https://youtu.be/LO1svLcCZ5c