LAB: GPIO Digital InOut

Date: 2023-09-25

Author: NohYunKi(21800226)

Demo Video: Link

Introduction

In this lab, three external LED devices will be turned off and on by controlling pin's state. To achieve this goal, there are three necessary step. First, we have to use library code. Second, We have to learn how to toggle internal LED with button. Third, by applying how to toggle internal LED we have to turn on and off each LEDs.

Requirement

Hardware

- MCU
  - NUCLEO-F411RE
- Actuator/Sensor/Others:
  - LEDs x 3
  - Resistor 330 ohm x 3, breadboard

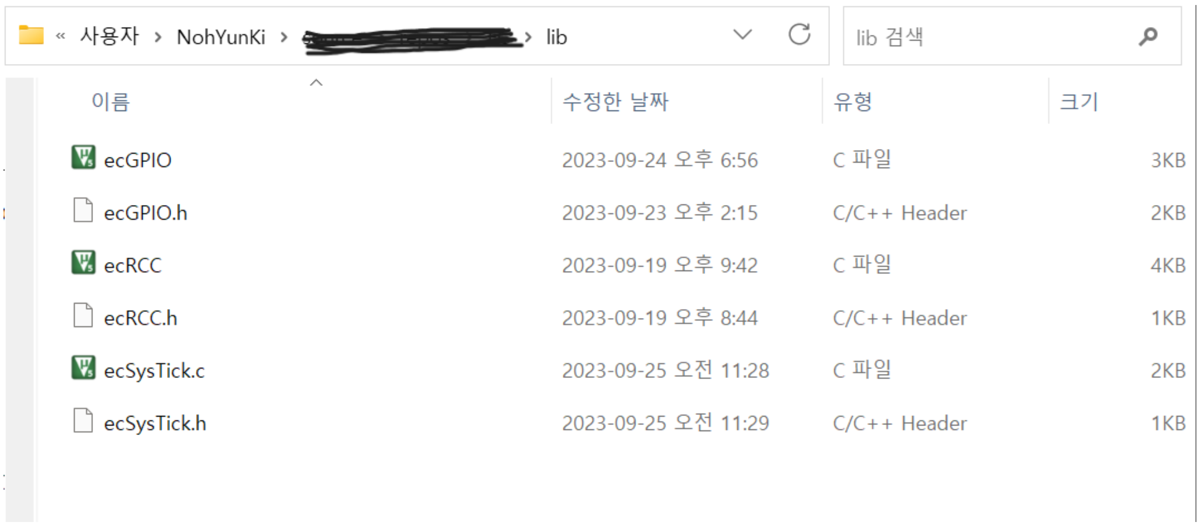
Software

- Keil uVision, CMSIS, EC_HAL library

Problem 1: Create EC_HAL library

Procedure

We recommend to make library file path only with English. To follow this advice, the file path probably will be created in user's folder.


« 사용자 › NohYunKi › [redacted] › lib

lib 검색

이름	수정한 날짜	유형	크기
ecGPIO	2023-09-24 오후 6:56	C 파일	3KB
ecGPIO.h	2023-09-23 오후 2:15	C/C++ Header	2KB
ecRCC	2023-09-19 오후 9:42	C 파일	4KB
ecRCC.h	2023-09-19 오후 8:44	C/C++ Header	1KB
ecSysTick.c	2023-09-25 오전 11:28	C 파일	2KB
ecSysTick.h	2023-09-25 오전 11:29	C/C++ Header	1KB

Image 1. File path

Then send header files which you downloaded to this location.

- Necessary header files: ecRCC.c, ecRCC.h and ecGPIO.c, ecGPIO.h

Download link: header files link

Problem 2: Toggle LED with Button

Procedure

- A port setting(LED)

```
1. Make A port enable
1. Set A port's 5th pin as Output mode
1. Set A port's 5th pin as open_drain mode
1. Set A port's 5th pin as Pull-up mode
1. Set A port's 5th pin as Fast-speed mode
```

- C port setting(Button)

1. Make C port enable
2. Set C port's 13th pin as Input mode
3. Set C port's 13th pin as Pull-up mode

- Logic

When Button's pin receive signal, change the state of Input's pin

Configuration

Button(B1)	LED
Digital In	Digital Out
GPIOC, Pin13	GPIOA, Pin5
PULL-UP	Open-Drain, Pull-up, Medium Speed

Figure 1. Configuration

Code

```
/**
******************************************************************************
* @author   Noh YunKi
* @Mod      23/09/21
* @brief   Embedded Controller:  LAB Digital In/Out
*                 - Toggle LED LD2 by Button B1 pressing
*
******************************************************************************
*/
```

```
#include "stm32f4xx.h"
#include "ecRCC.h"
#include "ecGPIO.h"

#define LED_PIN    5
#define BUTTON_PIN 13



void setup(void);

int main(void) {
   // Initialiization --------------------------------------------------------
   setup();

   // Inifinite Loop ----------------------------------------------------------
   while(1){
      if(GPIO_read(GPIOC, BUTTON_PIN) == 0){
         if(GPIO_read(GPIOA, LED_PIN) == LOW) GPIO_write(GPIOA, LED_PIN, HIGH);
         else GPIO_write(GPIOA, LED_PIN, LOW);
                 for(volatile int i = 0; i < 200000; i++){}
      }
   }
}


// Initialiization
void setup(void)
{
   RCC_HSI_init();

   /*-------------------------------------------------------
   BUTTON_PIN's settings:
   Digital Input, GPIOC13, Pull-up
   -------------------------------------------------------*/
   GPIO_init(GPIOC, BUTTON_PIN, INPUT);  // 0.Enalbe & 1. Input
   GPIO_pupd(GPIOC, BUTTON_PIN, EC_PU);  // 3. Pull-up


   /*-------------------------------------------------------
   LED_PIN's settings:
   Digital Output, GPIOA5, Open_darin, pull-up, Medium Speed
   -------------------------------------------------------*/
   GPIO_init(GPIOA, LED_PIN, OUTPUT);    // 0.Enable & 1.Output
   GPIO_otype(GPIOA, LED_PIN, 0);            // 2.oepn_darin
   GPIO_pupd(GPIOA, LED_PIN, EC_PU);         // 3.Pull-up
   GPIO_ospeed(GPIOA, LED_PIN, EC_FAST); // 4.Fast speed

}
```

Discussion

1. Typical Solution for Debouncing

- Software Solution: Check time interval

Debouncing is occured because of duplication current flow at device. In this case, when button is pressed and released, several slight shakes occur at the Button. As a result of this, Button is pressed and released several times at one click. To resolve this problem, we will use the time interval information of button. When button's state is changed, we will check whether the value of interval is over than 30ms or not. If time interval is over 30ms, we will decide to change the state of button. In contrast to this case, if time interval is under 30ms, then we will decide this signal doesn't mean the button is pressed and release. By applying this method, we would to avoid duplication problem of Button.

- Hardware Solution: Schmitt Trigger

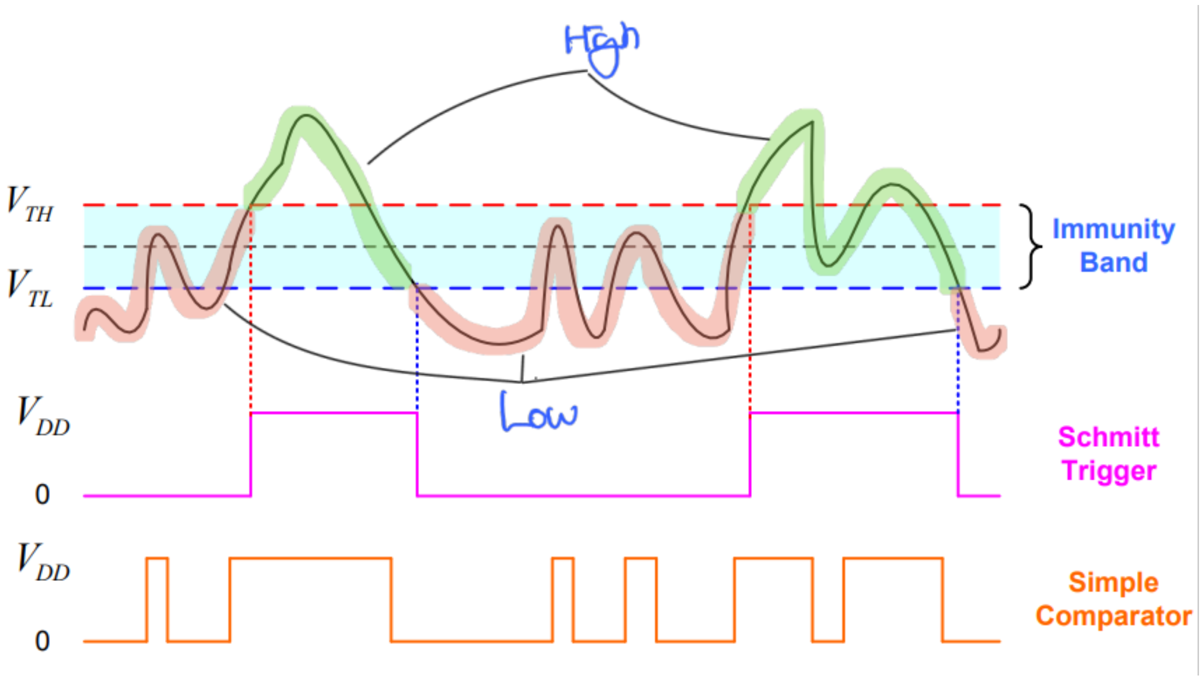


Image 2. Schmitt Trigger

Schmitt Trigger is device to remove noise. There is 'Immunity Band' area. This area exists for filtering. When High signal which is over V_TH is lowered before V_TL value, this series of signal will be decided as HIGH signal. In contrast of this case, when Low signal which is under V_TL value goes up before V_TH, this series of signal will be decided as LOW signal. By applying this device, noises between Immunity Band could be removed.

Problem 3: Toggle LED with Button

Procedure

1. Set circuit

To connect each pin(PA6, PA7, PB6) to each LED, three LEDs are installed on the breadboard. One side of LED will be connected with pin's line and the other side of LED will be connected with resistance to be supplied power.

2. Connect each pins of stem board with LED

To turn on and off each LED, three pins are used. With the first LED, Port A's 6th pin is connected. And with the second LED, Port A's 7th pin is connected. Lastly, with the third LED, Port B's 13th pin is connected.

3. Programming

To use button and each LED, initial setting is performed. After this, by considering the state of each LED pin when Button is pressed, turn on and off each LED pin sequentially. More detail will be provided in 'Code' section.

Configuration

Button(B1)	**LED**
Digital in	Digital out
GPIOC, pin 13	PA5, PA6, PA7, PB6
Pull-up	Push-pull, Pull-up, Medium-speed

Figure 2. Configuration

Circuit Diagram

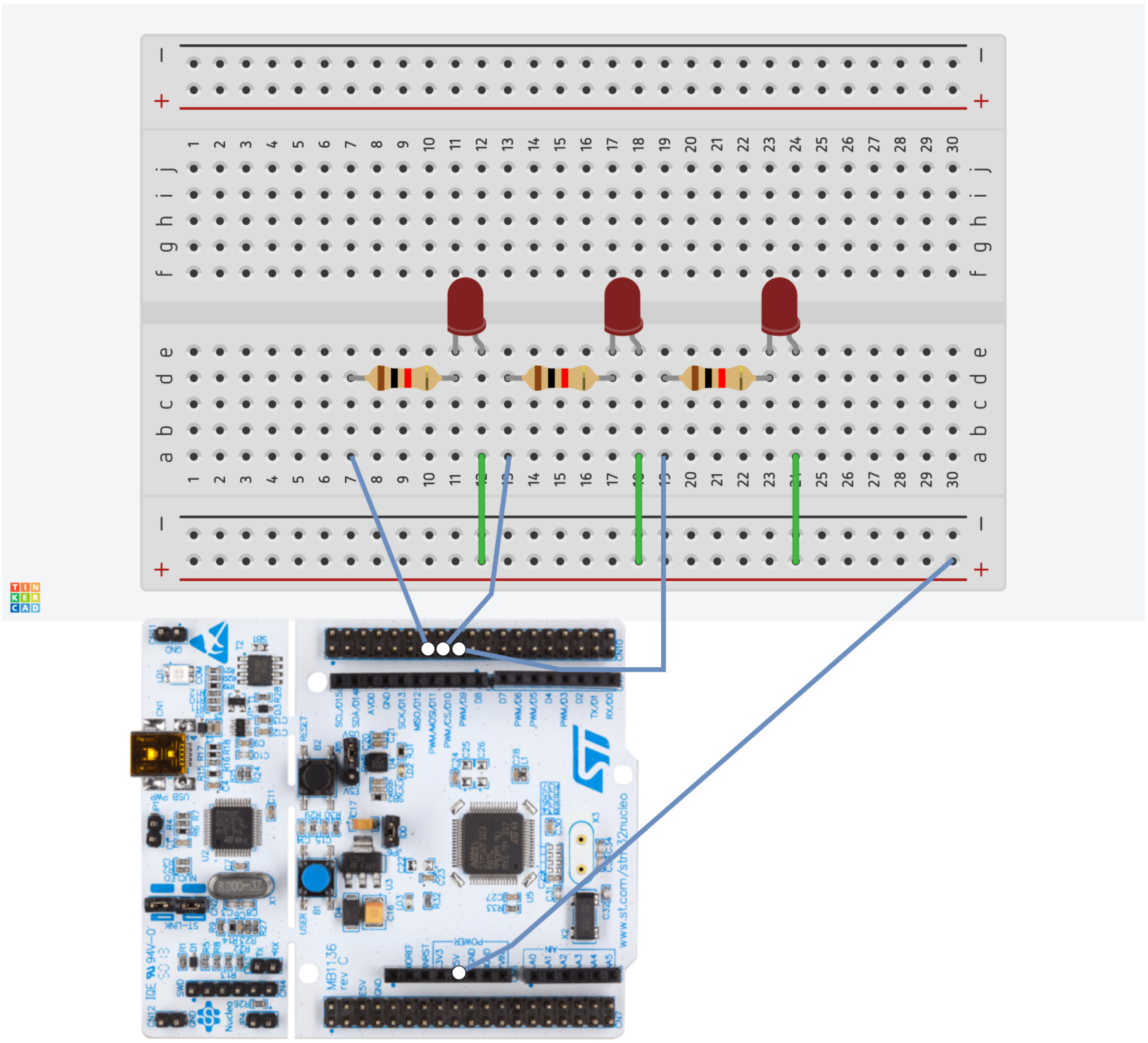


Figure 2. Circuit Diagram

Code

```
/**
******************************************************************************
* @author    NohYunKi
* @Mod       23/09/23
* @brief    Embedded Controller:  LAB Digital In/Out
*                 - Toggle multiple LEDs by Button B1 pressing
*
******************************************************************************
*/

#include "stm32f4xx.h"
#include "ecRCC.h"
#include "ecGPIO.h"

#define LED_PIN     5 // PA5
#define BUTTON_PIN 13
```

```
#define PA6 6
#define PA7 7
#define PB6 6

void setup(void);

int main(void) {
   // Initialiization --------------------------------------------------------
   setup();
    int count = 0;
   int signal = 0;

   // Inifinite Loop ----------------------------------------------------------
    while(1){

       if(GPIO_read(GPIOC, BUTTON_PIN) == 0){
                 signal = count % 4;

                 switch(signal){

                     case 0 :
                         GPIOA->ODR ^= (1<<LED_PIN); //ON
                         if(count > 3){
                                GPIOB->ODR ^= (1<<PB6); //OFF
                         }
                         break;

                     case 1:
                            GPIOA->ODR ^= (1<<LED_PIN); //OFF
                            GPIOA->ODR ^= (1<<PA6); //ON
                            break;

                     case 2:
                            GPIOA->ODR ^= (1<<PA6); //OFF
                            GPIOA->ODR ^= (1<<PA7); //ON
                            break;

                     case 3:
                            GPIOA->ODR ^= (1<<PA7); //OFF
                            GPIOB->ODR ^= (1<<PB6); //ON
            }
                for(volatile int i = 0; i < 200000; i++){}
                count ++;
        }
    }
}


// Initialiization
void setup(void)
{
   RCC_HSI_init();

   /*----------------------------------------------------------
   BUTTON_PIN's settings:
   Digital Input, GPIOC13, Pull-up
   -------------------------------------------------------*/
   GPIO_init(GPIOC, BUTTON_PIN, INPUT);  // 0.Enalbe & 1. Input
   GPIO_pupd(GPIOC, BUTTON_PIN, EC_PU);  // 3. Pull-up

   /*-----------------------------------------------------
   LED_PIN's settings:
   Digital Output, (GPIOA5,GPIOA6,GPIOA7,GPIOB6), Push-Pull, Pull-up, Medium
Speed
   -------------------------------------------------------*/
   GPIO_init(GPIOA, LED_PIN, OUTPUT);    // PA5: 0.Enable & 1.Output
   GPIO_init(GPIOA, PA6, OUTPUT);            // PA6: 0.Enable & 1.Output
   GPIO_init(GPIOA, PA7, OUTPUT);            // PA7: 0.Enable & 1.Output
   GPIO_init(GPIOB, PB6, OUTPUT);            // PB6: 0.Enable & 1.Output

   /*GPIOA5*/
   GPIO_otype(GPIOA, LED_PIN, EC_PUSH_PULL);   // 2.Push-Pull
   GPIO_pupd(GPIOA, LED_PIN, EC_PU);               // 3.Pull-up
   GPIO_ospeed(GPIOA, LED_PIN, EC_MEDIUM);    // 4.Medium speed
     GPIO_write(GPIOA, LED_PIN, LOW);

   /*GPIOA6*/
   GPIO_otype(GPIOA, PA6, EC_PUSH_PULL);   // 2.Push-Pull
   GPIO_pupd(GPIOA, PA6, EC_PU);               // 3.Pull-up
   GPIO_ospeed(GPIOA, PA6, EC_MEDIUM);    // 4.Medium speed
     GPIO_write(GPIOA, PA6, HIGH);

   /*GPIOA7*/
   GPIO_otype(GPIOA, PA7, EC_PUSH_PULL);   // 2.Push-Pull
   GPIO_pupd(GPIOA, PA7, EC_PU);               // 3.Pull-up
   GPIO_ospeed(GPIOA, PA7, EC_MEDIUM);    // 4.Medium speed
     GPIO_write(GPIOA, PA7, HIGH);

   /*GPIOB6*/
   GPIO_otype(GPIOB, PB6, EC_PUSH_PULL);   // 2.Push-Pull
   GPIO_pupd(GPIOB, PB6, EC_PU);               // 3.Pull-up
   GPIO_ospeed(GPIOB, PB6, EC_MEDIUM);    // 4.Medium speed
   GPIO_write(GPIOB, PB6, HIGH);

}
```

Results

Result: youtube_link

Discussion

Before starting this lab, I understood the fact that if pin's state(High and Low) is changed, the flow of current will be changed. However, I didn't exactly know how external devices are performed with each Port's pins. By connecting each pins with led and pressing Button, I could understand the mechanism that current flows between LED devices and each pins. If I didn't perform this lab, I wouldn't know how STM32 board works.

Trouble Shooting

```
  while(1){

        if(GPIO_read(GPIOC, BUTTON_PIN) == 0){
                  signal = count % 4;

                  switch(signal){

                      case 0 :
                          GPIOA->ODR ^= (1<<LED_PIN); //ON
                          if(count > 3){
                                 GPIOB->ODR ^= (1<<PB6); //OFF
                          }
                          break;

                      case 1:
                             GPIOA->ODR ^= (1<<LED_PIN); //OFF
                             GPIOA->ODR ^= (1<<PA6); //ON
                             break;

                      case 2:
                             GPIOA->ODR ^= (1<<PA6); //OFF
                             GPIOA->ODR ^= (1<<PA7); //ON
                             break;

                      case 3:
                             GPIOA->ODR ^= (1<<PA7); //OFF
                             GPIOB->ODR ^= (1<<PB6); //ON
            }
                 for(volatile int i = 0; i < 200000; i++){}
                 count ++;
        }
    }
 }
```

However I could learn '**volatile**' keyword is very important when performing repeated sentence. There were problem to my team mates' lab. When he used 'int' variable in the '**for sentence**' to delay after change led state, four pins are not turned on and off regularly. However, when he used '**volatile int**' instead of '**int**', his program worked well. At this problem, we could find that volatile variable's purpose. In the interface of Arduino, when program optimizes, some variables can be changed. So, if we want to preserve specific variable's value, we must to use 'volatile' keyword. Then, we can performing as we want to work with this variable.