LAB: GPIO Digital InOut

Date : 2023.09.23

Author/Partner: Duwon Yang / Yoonseok Choi

Github : LINK

Demo Video : LINK

Introduction

In this lab, We create a simple program that switch multiple LEDs with pushbutton input.

We used Nucleo-F411RE to implement this program, and the library by creating a HAL driver for GPIO digital input and output control.

Requirement

Hardware

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

Software

- Keli uVision, CMSIS, EC_HAL library

Problem 1 : Create EC_HAL library

Procedure

Create the library directory `\repos\EC\lib\.`

Save header library files in this directory.

Create own library for Digital_In and Out : `ecGPIO.h, ecGPIO.c`

ecRCC.h (provided)

```
void    RCC_HSI_init(void);
void    RCC_GPIOA_enable(void);
void    RCC_GPIOB_enable(void);
void    RCC_GPIOC_enable(void);
void    RCC_GPIOD_enable(void);
```

ecGPIO.h (provided)

```
void GPIO_init(GPIO_TypeDef *Port, unsigned int pin, unsigned int mode);
void GPIO_write(GPIO_TypeDef *Port, unsigned int pin, unsigned int Output);
unsigned int  GPIO_read(GPIO_TypeDef *Port, unsigned int pin);
void GPIO_mode(GPIO_TypeDef* Port, unsigned int pin, unsigned int mode);
void GPIO_ospeed(GPIO_TypeDef* Port, unsigned int pin, unsigned int speed);
void GPIO_otype(GPIO_TypeDef* Port, unsigned int pin, unsigned int type);
void GPIO_pupd(GPIO_TypeDef* Port, unsigned int pin, unsigned int pupd);
```

Problem 2 : Toggle LED with Button

Procedure

1. Create a new project under the directory `\repos\EC\lib\`.

- The project name is **"LAB_GPIO_DIO_LED"**
- Name the source file as **"LAB_GPIO_DIO_LED.c"**

2. Include library **ecGPIO.h, ecGPIO.c** in `\repos\EC\lib\`

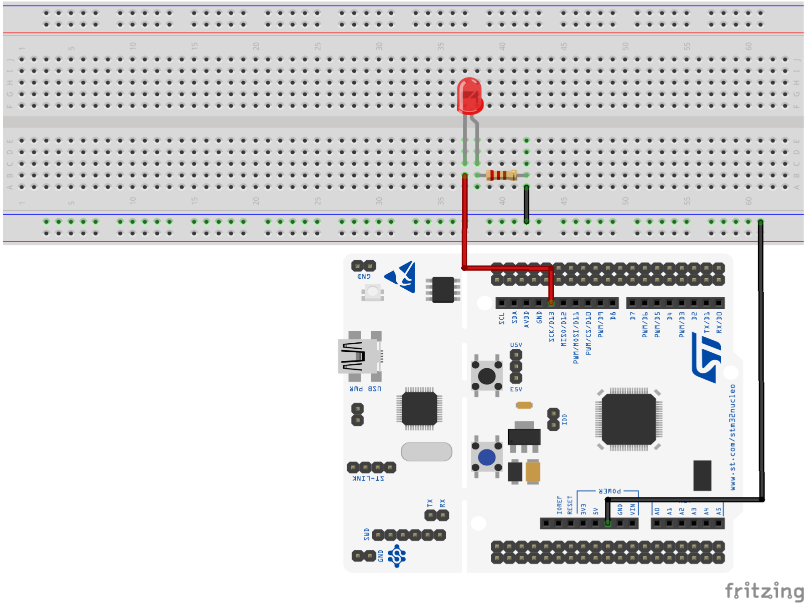
3. Toggle the LED by pushing the button.

- Push button (LED ON), Push Button (LED OFF) and repeat

Configuration

Button(B1)	LED
Digital In	Digital Out
GPIOC, Pin 13	GPIOA, Pin5
PULL-UP	Open-Drain, Pull-up, Medium Speed

Circuit diagram



Description with Code

- Lab source code: LINK
- setup the code : Define setup Function and Pin number.

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

#define LED_PIN     5
#define BUTTON_PIN 13

void setup(void);


// Initialiization
void setup(void)
{
    RCC_HSI_init();
    GPIO_init(GPIOC, BUTTON_PIN, INPUT);        // calls RCC_GPIOC_enable()
    GPIO_mode(GPIOC, BUTTON_PIN, INPUT);        // Set GPIOC as INPUT
    GPIO_pupd(GPIOC, BUTTON_PIN, EC_PU);        // Set GPIOC as PULL_UP


    GPIO_init(GPIOA, LED_PIN, OUTPUT);          // calls RCC_GPIOA_enable()
    GPIO_mode(GPIOA, LED_PIN, OUTPUT);          // Set GPIOC as OUTPUT
    GPIO_pupd(GPIOA, LED_PIN, EC_PU);           // Set GPIOC as PULL_UP
    GPIO_otype(GPIOA, LED_PIN, EC_PUSH_PULL);   // Set GPIOC as PUSH_PULL
    GPIO_ospeed(GPIOA, LED_PIN, EC_MEDIUM);     // Set GPIOC as MEDIUM SPEED
}
```

- main code : as button(B1) is pressed, let a LED ON. If is not, LED OFF. Because of Open-Drain setting, the LED ON occurs when LOW(0) input. and then, LED OFF occurs when HIGH(1) input.

```
int main(void) {
    // Initialiization
    setup();
    int delay = 0;
    int button_state = 0;
    // Inifinite Loop
    while(1){
        //when the button is pressed
        if(GPIO_read(GPIOC, BUTTON_PIN) == 0)
        {
            if(delay>10000 && button_state == 0)
            {
                GPIO_write(GPIOA, LED_PIN, LOW);
                delay = 0;
            }
            else if(delay>10000 && button_state == 1)
            {
                GPIO_write(GPIOA, LED_PIN, HIGH);
                delay = 0;
            }
            else
                delay++;
        }
        //state update
        else{
            if(button_state == 0)
                button_state = 1;
            else if(button_state == 1)
                button_state = 0;

        }
}
}
```

Discussion

1. Find out a typical solution for software debouncing and hardware debouncing

Software debouncing

At there, a timer can be used to perform a debouncing role. if the input change is deemed valid for a defined period of time, perform an operation. otherwise if deemed invalid, not perform the operation until the input stabilizes.

Hardware debouncing

In hardware, a low-frequency filter using resistors and capacitors serves to reduce noise. Additionally, the Schmitt trigger can be used to convert analog signals into clean digital signals with hysteresis. This allows the output to be maintained stably until the input signal exceeds a certain threshold.

2. What method of debouncing did this NUCLEO board use for the push-button(B1)?

Generally, use the method of Hardware debouncing. The push-button(B1) on the NUCLEO board has debouncing implemented in a hardware manner, through which noise and bounce effects occurring in button input are filterd out and converted into stable digital signals. This manners are achieved through the RC low-frequency filter and Schmitt trigger mentioned above. However, because ther is a limit to increasing the capacitor value, a method of forcing a time delay can be used by executing the debouncing delay used by increasing the delay in a while statement and initializing it every time a push-button is pressed.

Problem 3 : Toggle multiLED with Button

Procedure

1. Create a new project under the directory `\repos\EC\lib\.`

- The project name is **"LAB_GPIO_DIO_multiLED"**
- Name the source file as **"LAB_GPIO_DIO_multiLED.c"**

2. Include library **ecGPIO.h, ecGPIO.c** in `\repos\EC\lib\`

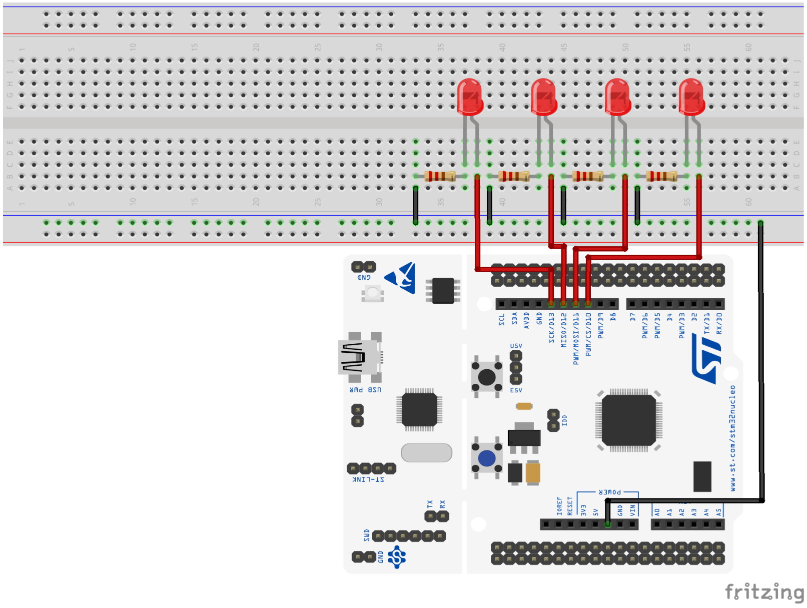
3. Connect 4 LEDs externally with necessary load resistors. Toggle the LED sequentially by pushing the button.

- As button B1 is pressed, light one LED at a time, in sequence.
- Example: LED0->LED1->...LED3->...LED0...

Configuration

Button(B1)	LED
Digital In	Digital Out
GPIOC, Pin 13	PA5, PA6, PA7, PB6
PULL-UP	Push-Pull, Pull-up, Medium Speed

Circuit diagram



Description with Code

- Lab source code: LINK

- setup the code : Define setup Function and Pin number(C13, A5, A6, A7, B6). set Button Pin as PULL-UP, and LEDs as Push-Pull, Pull-up, Medium speed.

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

#define LED_PIN_1   5
#define LED_PIN_2   6
#define LED_PIN_3   7
#define LED_PIN_4   6
#define BUTTON_PIN 13

void setup(void);

void setup(void)
{
    RCC_HSI_init();
    GPIO_init(GPIOC, BUTTON_PIN, INPUT);        // calls RCC_GPIOC_enable()
    GPIO_mode(GPIOC, BUTTON_PIN, INPUT);        // Set GPIOC as INPUT
    GPIO_pupd(GPIOC, BUTTON_PIN, EC_PU);        // Set GPIOC as PULL_UP

    GPIO_init(GPIOA, LED_PIN_1, OUTPUT);            // calls RCC_GPIOA_enable()
    GPIO_mode(GPIOA, LED_PIN_1, OUTPUT);            // Set GPIOC as OUTPUT
    GPIO_pupd(GPIOA, LED_PIN_1, EC_PU);             // Set GPIOC as PULL_UP
    GPIO_otype(GPIOA, LED_PIN_1, EC_PUSH_PULL); // Set GPIOC as PUSH_PULL
    GPIO_ospeed(GPIOA, LED_PIN_1, EC_MEDIUM);   // Set GPIOC as MEDIUM SPEED

    GPIO_init(GPIOA, LED_PIN_2, OUTPUT);            // calls RCC_GPIOA_enable()
    GPIO_mode(GPIOA, LED_PIN_2, OUTPUT);            // Set GPIOC as OUTPUT
    GPIO_pupd(GPIOA, LED_PIN_2, EC_PU);             // Set GPIOC as PULL_UP
    GPIO_otype(GPIOA, LED_PIN_2, EC_PUSH_PULL); // Set GPIOC as PUSH_PULL
    GPIO_ospeed(GPIOA, LED_PIN_2, EC_MEDIUM);   // Set GPIOC as MEDIUM SPEED

    GPIO_init(GPIOA, LED_PIN_3, OUTPUT);            // calls RCC_GPIOA_enable()
    GPIO_mode(GPIOA, LED_PIN_3, OUTPUT);            // Set GPIOC as OUTPUT
    GPIO_pupd(GPIOA, LED_PIN_3, EC_PU);             // Set GPIOC as PULL_UP
    GPIO_otype(GPIOA, LED_PIN_3, EC_PUSH_PULL); // Set GPIOC as PUSH_PULL
    GPIO_ospeed(GPIOA, LED_PIN_3, EC_MEDIUM);   // Set GPIOC as MEDIUM SPEED

    GPIO_init(GPIOB, LED_PIN_4, OUTPUT);            // calls RCC_GPIOA_enable()
    GPIO_mode(GPIOB, LED_PIN_4, OUTPUT);            // Set GPIOC as OUTPUT
    GPIO_pupd(GPIOB, LED_PIN_4, EC_PU);             // Set GPIOC as PULL_UP
    GPIO_otype(GPIOB, LED_PIN_4, EC_PUSH_PULL); // Set GPIOC as PUSH_PULL
    GPIO_ospeed(GPIOB, LED_PIN_4, EC_MEDIUM);   // Set GPIOC as MEDIUM SPEED
}
```

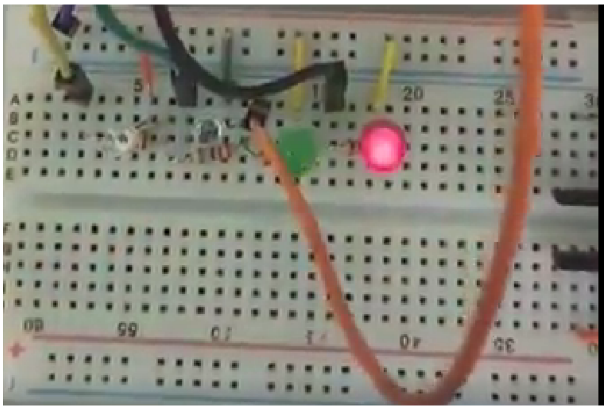
- main code : as button(B1) is pressed, the delay continues to be added within an infinite loop, and LED PIN 1-4 is turned on and off sequentially by debouncing under the condition that it exceeds a certain value(100,000).

```
int main(void) {
    // Initialiization --------------------------------------------------------
    setup();
    int rep = 1, state = 1;
    int delay = 0;


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



        if(GPIO_read(GPIOC, BUTTON_PIN) == 0) {
            if((state == 1)&&(delay > 10000)){
                GPIO_write(GPIOA, LED_PIN_1, HIGH);
                GPIO_write(GPIOB, LED_PIN_4, LOW);
                delay = 0; rep = 2;
            }
            else if((state == 2)&&(delay > 10000)){
                GPIO_write(GPIOA, LED_PIN_1, LOW);
                GPIO_write(GPIOA, LED_PIN_2, HIGH);
                delay = 0; rep = 3;
            }
            else if((state == 3)&&(delay > 10000)){
                GPIO_write(GPIOA, LED_PIN_2, LOW);
                GPIO_write(GPIOA, LED_PIN_3, HIGH);
                delay = 0; rep = 4;
            }
            else if((state == 4)&&(delay > 10000)){
                GPIO_write(GPIOA, LED_PIN_3, LOW);
                GPIO_write(GPIOB, LED_PIN_4, HIGH);
                delay = 0; rep = 1;
            }
        }

        else{
            if(rep == 1)
                state = 1;
            else if(rep == 2)
                state = 2;
            else if(rep == 3)
                state = 3;
            else if(rep == 4)
                state = 4;
        }
                delay++;
    }
}
```

Results

At this experiment, We implemented a program on the Nucleo board that can turn on four LEDs sequentially as the button is pressed. In order to implement the above function, we understand the need for the debounce function and learn how to apply it.

Demo Video



LINK

Discussion

1. Find out a typical solution for software debouncing and hardware debouncing. What method of debouncing did this NUCLEO board use for the push-button(B1)?

General hardware debouncing is acoomplished through RC low-frequency filter and Schimitt trigger. On the other hand, software debouncing is generally done by using a timer to check that the input is stable and then providing an output.

The push-button(B1) on the NUCLEO board has debouncing implemented in a hardware manner, through which noise and bounce effects occurring in button input are filterd out and converted into stable digital signals. This manners are achieved through the RC low-frequency filter and Schmitt trigger. However, because ther is a limit to increasing the capacitor value, a method of forcing a time delay can be used by executing the debouncing delay used by increasing the delay in a while statement and initializing it every time a push-button is pressed.

Troubleshooting

- Push-Pull vs Open-Drain

When the pin was connected to the anode and the ground to the cathode, the LED turns on when a HIGH input is given in push-pull mode and turns off when a LOW input is given. However, when switching to Open-Drain mode, the LED light was dimly lit. This was because one end of the Open-Drain was open to accept external voltage. Accordingly, we connected an external source(3.3V) to the anode and a pin to the cathode, the LED turns on when a LOW input is given, and the LED turns off when a HIGH input is given.

- Debouncing

Before the software debouncing, 4 LEDs turned on simultaneously. This was a phenomenon in which the LEDs turned on and off so quickly that it appeared as if four of them were turning on at the same time. Accordingly, by continuously adding delay within the while statement and setting one more delay conditional statement, a debounce effect was createed to force the LEDs to turn off and on sequnetially.

Reference

https://ykkim.gitbook.io/ec/ec-course/lab/lab-report-template

https://ykkim.gitbook.io/ec/ec-course/lab/lab-gpio-digital-inout