LAB: GPIO Digital Multi InOut

Date: 2022-09-28

Author/Partner (student ID): SeungEung Hwang / Jeahyun Oh (21800805 / 21800447)

Github: repository link

Demo Video: Youtube link

PDF version:

Introduction

In this lab, you are required to create a simple program to control a 7-segment display to show a decimal number (0~9) that increases by pressing a push-button.

Requirement

Hardware

- MCU
  - NUCLEO-F401RE
- Actuator/Sensor/Others:
  - LEDs x 3
  - Array Resistor 220 ohm
  - breadboard

Software

- Keil uVision, CMSIS, EC_HAL library

Problem 1: Create EC_HAL library

Procedure

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

List of functions for Digital_In and Out

ecRCC.h

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

ecGPIO.h

```
// GPIO Mode
#define INPUT  0x00
#define OUTPUT 0x01
```

```
#define AF     0x02
#define ANALOG 0x03

#define HIGH 1
#define LOW  0

// GPIO Speed          : Low speed (00), Medium speed (01), Fast speed (10), High
speed (11)
#define LSPEED  0x00
#define MSPEED  0x01
#define FSPEED  0x02
#define HSPEED  0x03

// Output type
// Pullup Pulldown
#define PUSHPULL    0
#define OPDRAIN         1


// GPIO Push-Pull    : No pull-up, pull-down (00), Pull-up (01), Pull-down (10),
Reserved (11)
#define NOPUPD  0x00
#define PULLUP  0x01
#define PULLDO  0x02
#define RESERV  0x03


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

ecGPIO.c

```
void GPIO_init(GPIO_TypeDef *Port, int pin, int mode){
    // mode  : Input(0), Output(1), AlterFunc(2), Analog(3)
    if (Port == GPIOA)
        RCC_GPIOA_enable();

    if (Port == GPIOB)
        RCC_GPIOB_enable();

    if (Port == GPIOC)
        RCC_GPIOC_enable();

    if (Port == GPIOD)
        RCC_GPIOA_enable();

    if (Port == GPIOE)
        RCC_GPIOB_enable();

    GPIO_mode(Port, pin, mode);

}
```

```
// GPIO Mode          : Input(00), Output(01), AlterFunc(10), Analog(11)
void GPIO_mode(GPIO_TypeDef *Port, int pin, int mode){
   Port->MODER &= ~(3UL<<(2*pin));        // mask
   Port->MODER |= mode <<(2*pin);         // choose the mode
}

// GPIO Speed          : Low speed (00), Medium speed (01), Fast speed (10), High
speed (11)
void GPIO_ospeed(GPIO_TypeDef *Port, int pin, int speed){

        Port->OSPEEDR &= ~(3UL  << (pin * 2));      // mask
        Port->OSPEEDR |=  speed << (pin * 2);       // choose the speed

}

// GPIO Output Type: Output push-pull (0, reset), Output open drain (1)
void GPIO_otype(GPIO_TypeDef *Port, int pin, int type){

        Port -> OTYPER &= ~(type << pin);       // 0:Push-Pull

}

// GPIO Push-Pull    : No pull-up, pull-down (00), Pull-up (01), Pull-down (10),
Reserved (11)
void GPIO_pupd(GPIO_TypeDef *Port, int pin, int pupd){

        Port->PUPDR  &=     ~(3UL << (pin * 2));        // 00: claer

        Port->PUPDR  |=     (pupd   << (pin * 2));          // 00: write

}

int GPIO_read(GPIO_TypeDef *Port, int pin){

    // 0 or 1만 읽기 위해서 사용하는 방법
    return ((Port -> IDR) >> pin) & 1UL;

}

void GPIO_write(GPIO_TypeDef *Port, int pin,  int output){

         if(output == 1)

                Port->ODR |= (1UL << pin);

            else

                Port->ODR &= ~(1UL << pin);
}
```

Problem 2: Toggle LED with Button

Procedure

1. Create a new project under the directory `\repos\EC\LAB\`
2. Include your library **ecGPIO.h, ecGPIO.c** in `\repos\EC\lib\`
3. Toggle the LED by pushing button.

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

Configuration

- Button (B1)
  - Digital In
  - GPIOC, Pin 13 (**PC13**)
  - PULL-UP
- LED
  - Digital Out
  - GPIOA, Pin 5 (**PA5**)
  - Open-Drain
  - Pull-up
  - Medium Speed

Code

```
void setup(void);
void bitToggle ( GPIO_TypeDef *Port, int pin);

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

    // Inifinite Loop
    while(1){

        if(GPIO_read(GPIOC, BUTTON_PIN) == 0)       { // when button pressed

            bitToggle(GPIOA, LED_PIN);                // bittoggle function
        }

        delay_ms(30);                                 // software debouncing

    }
}

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

    RCC_HSI_init();
    GPIO_init(GPIOC, BUTTON_PIN, INPUT);    // calls RCC_GPIOC_enable()
    GPIO_init(GPIOA, LED_PIN, OUTPUT);      // calls RCC_GPIOA_enable()

    GPIO_otype(GPIOA, LED_PIN, OPDRAIN);    // LED open drain

    GPIO_pupd(GPIOA, LED_PIN, PULLUP);      // LED PULLUP
```

```
    GPIO_ospeed(GPIOA, LED_PIN, MSPEED);     // LED Medium speed
    GPIO_pupd(GPIOC, BUTTON_PIN, PULLUP);    // button pull up

    GPIO_write(GPIOA, LED_PIN, LOW);         // claer LED

}

void bitToggle ( GPIO_TypeDef *Port, int pin){

    Port -> ODR ^= (1 << pin);

}
```

Discussion

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

   software debouncing

   The software debouncing is using delay for few seconds and

   Hardware debouncing

   The Hardware debouncing is use RC circuit and schmitt trigger. RC circuit make ripples signal, so it is insensitive to switch signal changes, and uses schmitt triggers to debounce more effectively.

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

   In NUCELO's every GPIO Registers have schmitt trigger without RC circuit. So in this lab I used software debouncing, because only hardware debouncing can not prevent the sensing error.

Problem 3: Toggle LED with Button

Procedure

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

- The project name is "**LAB_GPIO_DIO_multiLED".**
- Name the source file as "**LAB_GPIO_DIO_multiLED.c"**
  1. Include your library **ecGPIO.h, ecGPIO.c** in `\repos\lib\`.
  2. Connect 4 LEDs externally with necessary load resistors.
- As Button B1 is Pressed, light one LED at a time, in sequence.
- Example: LED0--> LED1--> ...LED3--> ...LED0....

Configuration

- Button (B1)
  - Digital In
  - Pin : **PC13**
  - PULL-UP
- LED

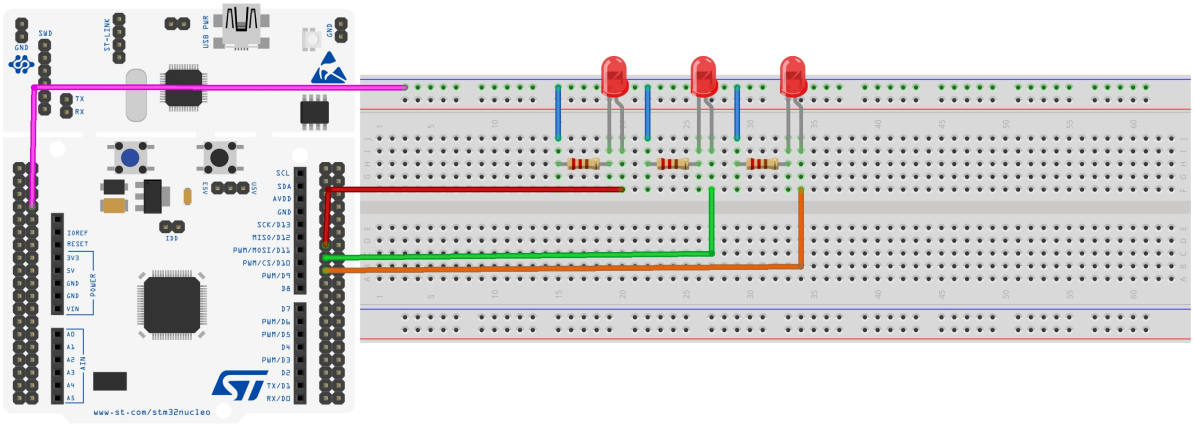
- Digital Out
- pin : **PA5**, **PA6**, **PA7**, **PB6**
- Open-Drain
- Pull-up
- Medium Speed

Code

```
#define PA5         5
#define PA6         6
#define PA7         7
#define PB6         6

#define BUTTON_PIN 13

void setup(void);

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

    // Inifinite Loop

    int count_button = 0;
    int flag = 0;

    while(1){

        if(GPIO_read(GPIOC, BUTTON_PIN) == 0)       {

            count_button ++;
            flag = count_button % 4;

            switch(flag){

                case 1:

                    bitToggle(GPIOA, PA5);      // led 1 on : stm led
                    if(count_button > 4){
                        bitToggle(GPIOB, PB6);  // if after sencond terms led 4
off
                    }
                    break;

                case 2:

                    bitToggle(GPIOA, PA5);      // led 1 off : red led
                    bitToggle(GPIOA, PA6);      // led 2 on
                    break;

                case 3:

                    bitToggle(GPIOA, PA6);      // led 2 off : green led
                    bitToggle(GPIOA, PA7);    // led 3 on
                    break;
```

```
                case 0:

                    bitToggle(GPIOA, PA7);      // led 3 off : white led
                    bitToggle(GPIOB, PB6);    // led 4 on
                    break;
            }
        }

        delay_ms(30);                                       // software
debouncing

    }
}


// Initialiization
void setup(void)
{

    RCC_HSI_init();
    SysTick_init();

    // button setup
    GPIO_init(GPIOC, BUTTON_PIN, INPUT);        // calls RCC_GPIOC_enable()
    GPIO_pupd(GPIOC, BUTTON_PIN, PULLUP);       // button pull up

    // led 1 setup
    GPIO_init       (GPIOA, PA5, OUTPUT);       // calls RCC_GPIOA_enable()
    GPIO_otype      (GPIOA, PA5, PUSHPULL);     // LED open drain
    GPIO_pupd       (GPIOA, PA5, PULLUP);       // LED PULLUP
    GPIO_ospeed     (GPIOA, PA5, MSPEED);       // LED Medium speed
    GPIO_write      (GPIOA, PA5, LOW);          // claer LED

    // led 2 setup
    GPIO_init       (GPIOA, PA6, OUTPUT);       // calls RCC_GPIOA_enable()
    GPIO_otype      (GPIOA, PA6, PUSHPULL);     // LED open drain
    GPIO_pupd       (GPIOA, PA6, PULLUP);       // LED PULLUP
    GPIO_ospeed     (GPIOA, PA6, MSPEED);       // LED Medium speed
    GPIO_write      (GPIOA, PA6, LOW);          // claer LED

    // led 3 setup
    GPIO_init       (GPIOA, PA7, OUTPUT);       // calls RCC_GPIOA_enable()
    GPIO_otype      (GPIOA, PA7, PUSHPULL);     // LED open drain
    GPIO_pupd       (GPIOA, PA7, PULLUP);       // LED PULLUP
    GPIO_ospeed     (GPIOA, PA7, MSPEED);       // LED Medium speed
    GPIO_write      (GPIOA, PA7, LOW);          // claer LED

    // led 4 setup
    GPIO_init       (GPIOB, PB6, OUTPUT);       // calls RCC_GPIOA_enable()
    GPIO_otype      (GPIOB, PB6, PUSHPULL);     // LED open drain
    GPIO_pupd       (GPIOB, PB6, PULLUP);       // LED PULLUP
    GPIO_ospeed     (GPIOB, PB6, MSPEED);       // LED Medium speed
    GPIO_write      (GPIOB, PB6, LOW);          // claer LED

}
```

Circuit Diagram



Code

[ecRCC.h]

[ecGPIO.h]

[ecGPIO.c]

[Problem 2]

[Problem 3]

Results

[demo video]

Discussion

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

   NUCLEO needs software debouncing because schimitt trigger was embedded in NUCLEO. but it could not prevent incorrect input. Therefore I use delay_ms function for time delay.

Reference

[ykkim's github]