LAB: GPIO Digital InOut

LAB: GPIO Digital InOut

Date: 2024-09-16

Author/Partner: Jong-Hyeon Kim

Github: https://github.com/Dandelion-Kim/Embedded-Controller

Demo Video: Problem0 / Problem1 / Problem3

Introduction

In this lab, you are required to create a simple program that toggle multiple LEDs with a push-button input. Create HAL drivers for GPIO digital in and out control and use your library.

Requirement

Hardware

- MCU
  - NUCLEO-F411RE
- Actuator/Sensor/Others:
  - LEDs x 4
  - Array Resistor, breadboard

Software

- Keil uVision, CMSIS, EC_HAL library

Problem 0: STM-Arduino

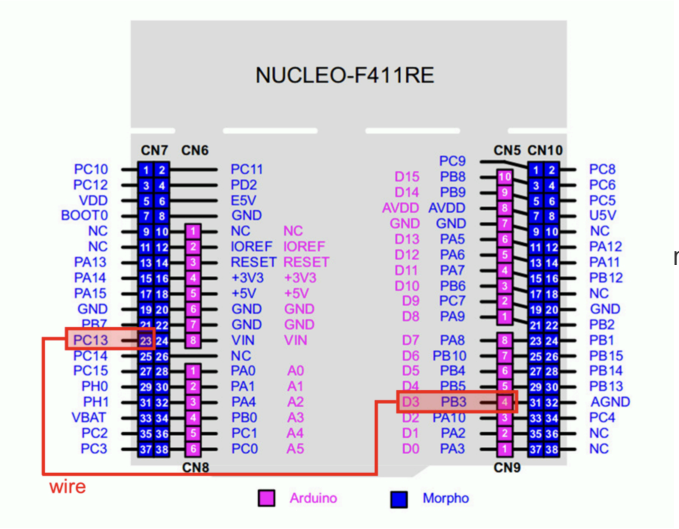
We are going to create a simple program that turns LED(LD2) on and off by pressing the user button(BT1), using Arduino Syntax

Code

```
const int btnPin = 3;
const int ledPin = 13;

int btnState = HIGH;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(btnPin, INPUT);
}
```

```
void loop() {
  btnState = digitalRead(btnPin);

  if (btnState == HIGH)
    digitalWrite(ledPin, LOW);

  else
    digitalWrite(ledPin, HIGH);

}
```

The user button pin is `PC13`, but this pin cannot be used in arduino. So, let's connect `PC13` to `pinName` `D3` by using wire.



m:25%;" />

Click on **upload** button. Push the reset button(black) and check the performance.

The LED(LD2) is turned on when the button is pressed.

[Click to watch the result of Problem_0 by Arduino IDE]

Problem 1: Create EC_HAL library

ecGPIO2.h

```
#ifndef __ECGPIO2_H
#define __ECGPIO2_H

#include "stm32f411xe.h"
#include "ecRCC2.h"
#include "ecPinNames.h"

//(GPIOx_MODER) GPIO port mode register
#define INPUT  0x00
#define OUTPUT 0x01
#define AF     0x02     //Alternate function
#define ANALOG 0x03   //Analog
```

```
//(GPIOx_OTYPER) GPIO port output type register
#define PSPL     0x0        //Push-Pull
#define OPDR     0x1        //Open Drain
//(GPIOx_OSPEEDR) GPIO port output speed register
#define LSPEED 0x00     //Low speed
#define MSPEED 0x01     //Medium speed
#define FSPEED 0x02     //Fast speed
#define HSPEED 0x03     //High speed
//(GPIOx_PUPDR) GPIO port pull-up/pull-down register
#define EC_NP  0x00     //No pull-up, pull down
#define EC_PU  0x01     //Pull-up
#define EC_PD    0x02       //Pull-down
#define EC_RE  0x03     //Reserved

#define HIGH 1
#define LOW  0

#define LED_PIN    //Find LED Port&Pin and Fill the blank
#define BUTTON_PIN //Find BTN Port&Pin and Fill the blank

#ifdef __cplusplus
 extern "C" {
#endif /* __cplusplus */

void GPIO_init(PinName_t pinName, uint32_t mode);
void GPIO_write(PinName_t pinName, int Output);
int  GPIO_read(PinName_t pinName);
void GPIO_mode(PinName_t pinName, uint32_t mode);
void GPIO_ospeed(PinName_t pinName, int speed);
void GPIO_otype(PinName_t pinName, int type);
void GPIO_pupd(PinName_t pinName, int pupd);



#ifdef __cplusplus
}
#endif /* __cplusplus */

#endif // __ECGPIO2_H
```

ecRCC2.h

```
#ifndef __EC_RCC2_H
#define __EC_RCC2_H

#ifdef __cplusplus
 extern "C" {
#endif /* __cplusplus */

//#include "stm32f411xe.h"

void RCC_HSI_init(void);
void RCC_PLL_init(void);
```

```
void RCC_GPIOA_enable(void);
void RCC_GPIOB_enable(void);
void RCC_GPIOC_enable(void);
void RCC_GPIOD_enable(void);
void RCC_GPIOE_enable(void);
void RCC_GPIOH_enable(void);
// void RCC_GPIO_enable(GPIO_TypeDef * GPIOx);

extern int EC_SYSCL;

#ifdef __cplusplus
}
#endif /* __cplusplus */

#endif // __EC_RCC2_H
```

ecGPIO2.c

```
#include "stm32f4xx.h"
#include "stm32f411xe.h"
#include "ecGPIO2.h"


void GPIO_init(PinName_t pinName, uint32_t mode){
    GPIO_TypeDef * Port;
    unsigned int pin;
    ecPinmap(pinName, &Port, &pin);

    // mode  : Input(0), Output(1), AlterFunc(2), Analog(3)
    if (Port == GPIOA)
        RCC_GPIOA_enable();
    if (Port == GPIOC)
        RCC_GPIOC_enable();
    if (Port == GPIOB)
        RCC_GPIOB_enable();
    if (Port == GPIOD)
        RCC_GPIOD_enable();
    if (Port == GPIOE)
        RCC_GPIOE_enable();
    if (Port == GPIOH)
        RCC_GPIOH_enable();
    //[TO-DO] YOUR CODE GOES HERE
    // Make it for GPIOB, GPIOD..GPIOH

    // You can also make a more general function of
// void RCC_GPIO_enable(GPIO_TypeDef *Port);

    GPIO_mode(pinName, mode);
    //GPIO_ospeed(pinName, mode);
    //GPIO_otype(pinName, mode);
    //GPIO_pupd(pinName, mode);
}
```

```
// GPIO Mode          : Input(00), Output(01), AlterFunc(10), Analog(11)
void GPIO_mode(PinName_t pinName, uint32_t mode){
   GPIO_TypeDef * Port;
   unsigned int pin;
   ecPinmap(pinName,&Port,&pin);
   Port->MODER &= ~(3UL<<(2*pin));
   Port->MODER |= mode<<(2*pin);
}


// GPIO Speed          : Low speed (00), Medium speed (01), Fast speed (10), High
speed (11)
void GPIO_ospeed(PinName_t pinName, int speed){
     GPIO_TypeDef * Port;
   unsigned int pin;
   ecPinmap(pinName,&Port,&pin);
   Port->OSPEEDR &= ~(3UL<<(2*pin));
   Port->OSPEEDR |= speed<<(2*pin);
}

// GPIO Output Type: Output push-pull (0, reset), Output open drain (1)
void GPIO_otype(PinName_t pinName, int type){
    GPIO_TypeDef * Port;
   unsigned int pin;
   ecPinmap(pinName,&Port,&pin);
   Port->OTYPER &= ~(1UL<<(pin));
   Port->OTYPER |= type<<(pin);
}

// GPIO Push-Pull    : No pull-up, pull-down (00), Pull-up (01), Pull-down (10),
Reserved (11)
void GPIO_pupd(PinName_t pinName, int pupd){
    GPIO_TypeDef * Port;
   unsigned int pin;
   ecPinmap(pinName,&Port,&pin);
   Port->PUPDR &= ~(3UL<<(2*pin));
   Port->PUPDR |= pupd<<(2*pin);
}

int GPIO_read(PinName_t pinName){
   GPIO_TypeDef * Port;
   unsigned int pin;
   ecPinmap(pinName,&Port,&pin);
    return Port->IDR&(1<<pin); //Heewon
        //return Port->IDR>>pin & 1UL;
}

void GPIO_write(PinName_t pinName, int Output){
     GPIO_TypeDef * Port;
   unsigned int pin;
   ecPinmap(pinName,&Port,&pin);

        Port->ODR &= ~(1 << pin);
        Port->ODR |= (Output << pin);
```

```
}
```

Problem 2: Toggle LED with Button

Procedure

- The project name is “**LAB_GPIO_DIO_LED”.**
- 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, Pin 5
PULL-UP	Open-Drain, Pull-up, Medium Speed

Code

```
#include "stm32f4xx.h"
#include "ecRCC2.h"
#include "ecGPIO2.h"

PinName_t LED_pin = PA_5;                                   // GPIOA Pin5
PinName_t button_pin = PC_13;                               // GPIOC Pin13

void setup(void);

int main(void) {
    // Initialiization --------------------------------------------------------
    setup();
    int output = HIGH;
    // Inifinite Loop ---------------------------------------------------------
    while(1){
            if(GPIO_read(button_pin) == 0)              // If the button(B1) is
pushed
            {
                GPIO_write(LED_pin, output);            // Turn the LED(LD2) on
                output = ~output;                       // Toggling
                while(GPIO_read(button_pin) == 0);      // Examine if the
button(B1) released.
            }
        }
    }
```

```
// Initialiization
void setup(void)
{
    RCC_HSI_init();
    // Setup for the button pin
    GPIO_init(button_pin, INPUT);       // calls RCC_GPIOC_enable()
    GPIO_pupd(button_pin, EC_PU);       // Pull-Up

    // Setup for the LED
    GPIO_init(LED_pin, OUTPUT);         // calls RCC_GPIOA_enable()
    GPIO_ospeed(LED_pin, MSPEED);       // Medium-Speed
    GPIO_pupd(LED_pin, EC_PU);          // Pull-Up
    GPIO_otype(LED_pin, OPDR);          // Open-Drain
}
```

Click to watch the result of Problem_2

Problem 3: Toggle 4-LEDs with Button

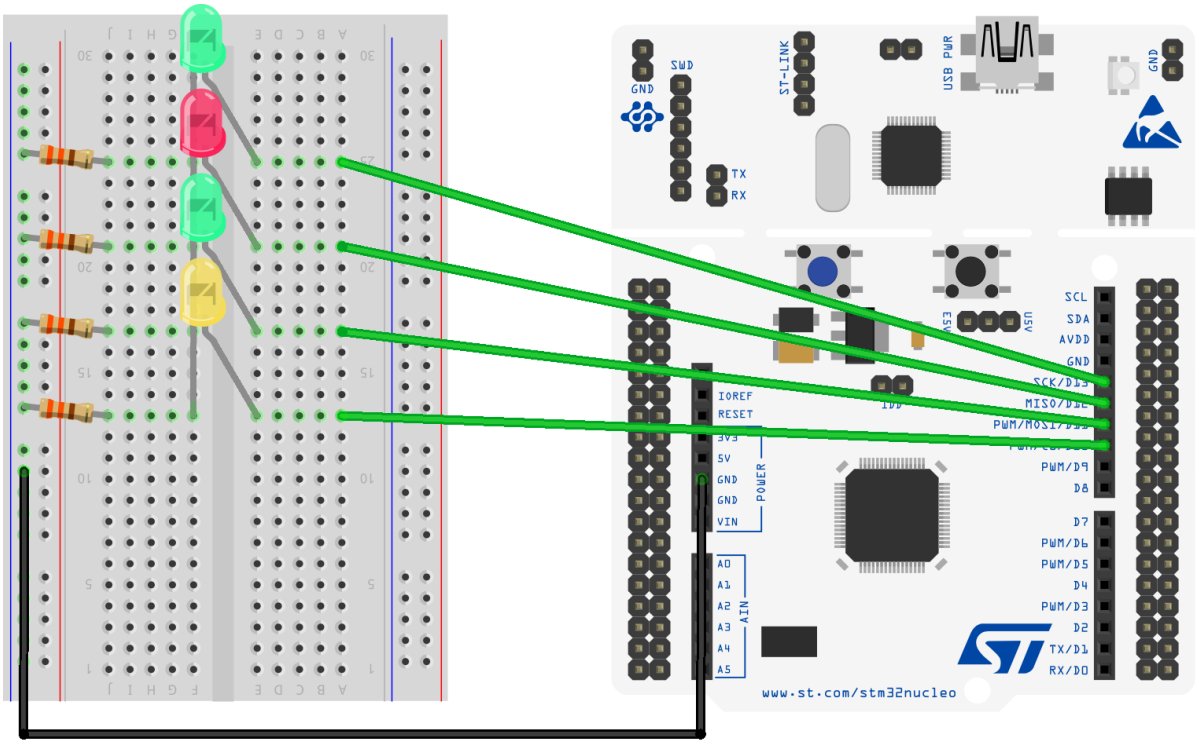
Procedure

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

Configuration

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

Circuit Diagram

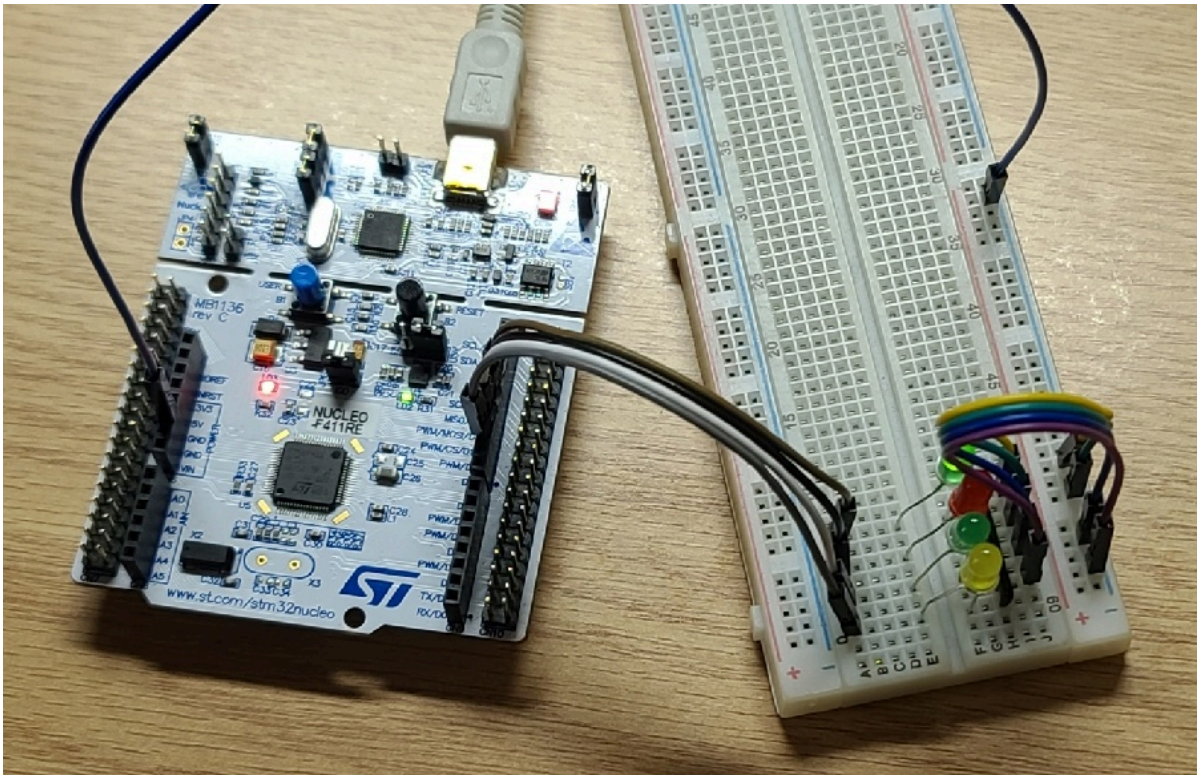


Code

```
#include "stm32f4xx.h"
#include "ecRCC2.h"
#include "ecGPIO2.h"

PinName_t LED_pin0 = PA_5;                                  // GPIOA Pin5
PinName_t LED_pin1 = PA_6;                                  // GPIOA Pin6
PinName_t LED_pin2 = PA_7;                                  // GPIOA Pin7
PinName_t LED_pin3 = PB_6;                                  // GPIOB Pin6

PinName_t button_pin = PC_13;                               // GPIOC Pin13

void setup(void);

int main(void) {
    // Initialiization --------------------------------------------------------
    setup();
    int count=0;
    int LED_ary[4]={LED_pin0, LED_pin1, LED_pin2, LED_pin3};        // Build a
Toggling Range

    // Inifinite Loop ---------------------------------------------------------
    while(1){
            if((GPIO_read(button_pin) == 0)){

                if(count==0){                                   // Start with the
LED_pin0
                GPIO_write(LED_ary[3],LOW);
                }
```

```
                else{
                GPIO_write(LED_ary[count-1],LOW);           // Turn off the
previous LED
                }
                GPIO_write(LED_ary[count],HIGH);            // Turn on the
present LED
                count++;                                    // Go to next count

                if(count==4) count=0;                       // It is the last
index. Go to first index of LED array.

                while(GPIO_read(button_pin)==0);            // Examine if the
button(B1) released.
        }
    }
}


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

    // Configuration button pin
    GPIO_init(button_pin, INPUT);   // calls RCC_GPIOC_enable()
    GPIO_pupd(button_pin, EC_PU);   // Pull-Up

    // Configuration LEDs
    //PA_5
    GPIO_init(LED_pin0, OUTPUT);    // calls RCC_GPIOA_enable()
    GPIO_otype(LED_pin0, PSPL);     // Push-Pull
    GPIO_ospeed(LED_pin0, MSPEED);  // Medium-Speed
    GPIO_pupd(LED_pin0, EC_PU);     // Pull-Up

    //PA_6
    GPIO_init(LED_pin1, OUTPUT);    // calls RCC_GPIOA_enable()
    GPIO_otype(LED_pin1, PSPL);     // Push-Pull
    GPIO_ospeed(LED_pin1, MSPEED);  // Medium-Speed
    GPIO_pupd(LED_pin1, EC_PU);     // Pull-Up

    //PA_7
    GPIO_init(LED_pin2, OUTPUT);    // calls RCC_GPIOA_enable()
    GPIO_otype(LED_pin2, PSPL);     // Push-Pull
    GPIO_ospeed(LED_pin2, MSPEED);  // Medium-Speed
    GPIO_pupd(LED_pin2, EC_PU);     // Pull-Up

    //PB_6
    GPIO_init(LED_pin3, OUTPUT);    // calls RCC_GPIOA_enable()
    GPIO_otype(LED_pin3, PSPL);     // Push-Pull
    GPIO_ospeed(LED_pin3, MSPEED);  // Medium-Speed
    GPIO_pupd(LED_pin3, EC_PU);     // Pull-Up
}
```

Results

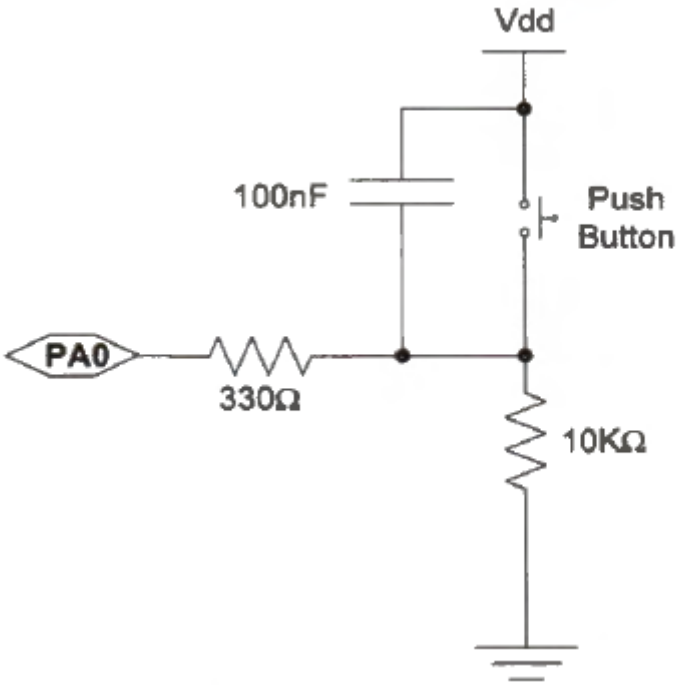


Click to watch the result of Problem_3

Discussion

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

For the **hardware bouncing**, as the picture below, if switch is opened, the capacitor will be fully charged. Then, if the button is pushed, the capacitor is quickly discharged and the processor pin(PA0) will be changed to High level. for the debouncing part, if button is opened immediately because the two metal meets and contact bang together, the capacitor charge speed cannot effect on processor pin. So, this circuit make hardware debouncing.



Software debouncing contains **'wait and see debouncer'** and **'counter debouncer'**. For the wait and see debouncer, it just examine if the still button state is same with the time(=ex)50ms) (Repeating 255 time in for or while loop consider as 1ms). But, this method is not good at overall, so we can use the counter debouncer. counter debouncer divde each examining flag. For example, when checking if the button state is same, counter debouner method check it by a specific number of repetitions, see if it is satisfied, and hand over the value.

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

NUCLEO board use the push-button(B1). The push-button is already applied hardware debouncing. So, it does not need software debouncing. **NUCLEO did hardware debouncing.**

Reference


```
- Y.Zhu(2017). Embedded Systems with Arm Cortex-M Microcontrollers in Assembly
Language and C: Third Edition
- https://ykkim.gitbook.io/ec
```