LAB: GPIO Digital InOut 7-segment

Date: 2022-09-26

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:
  - 7-segment display(5101ASR)
  - Array resistora (330 ohm) x 2
  - breadboard

Software

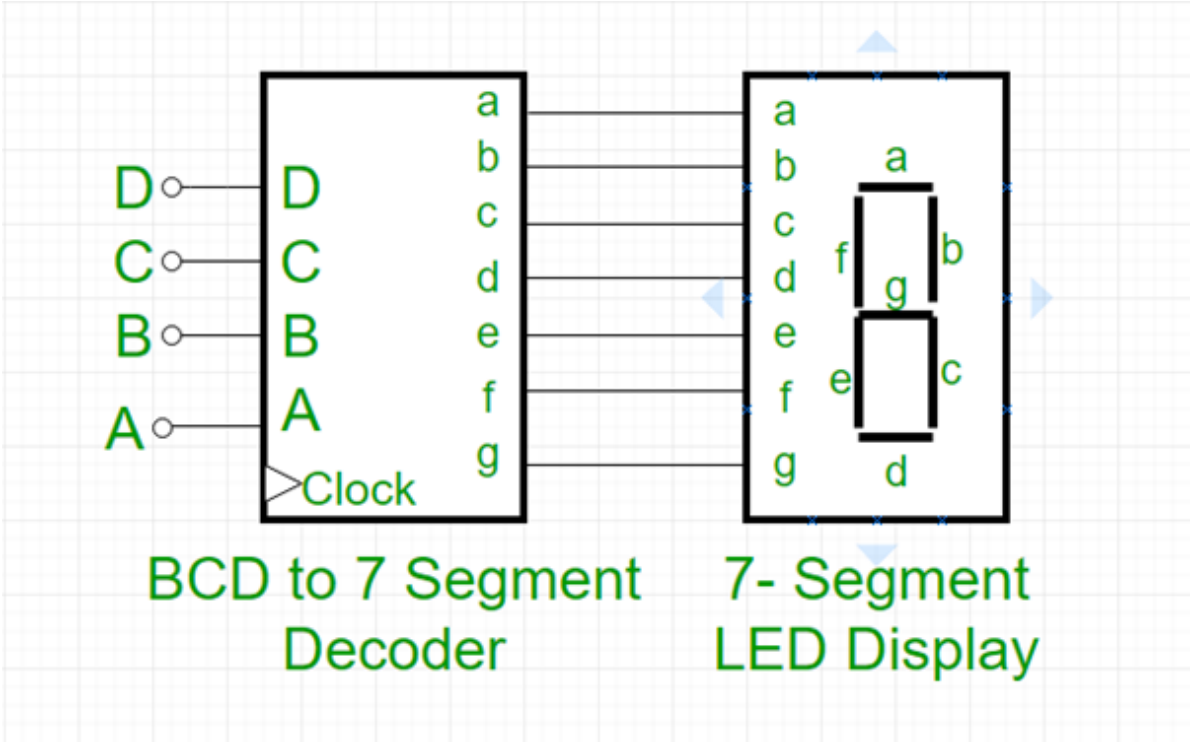
- Keil uVision, CMSIS, EC_HAL library

Problem 1: Connecting 7-Segment

Procedure

The popular BCD 7-segment decoder chips are **74LS47 and CD4511**.

Instead of using the decoder chip, we are going to make the 7-segment decoder with the MCU programming.



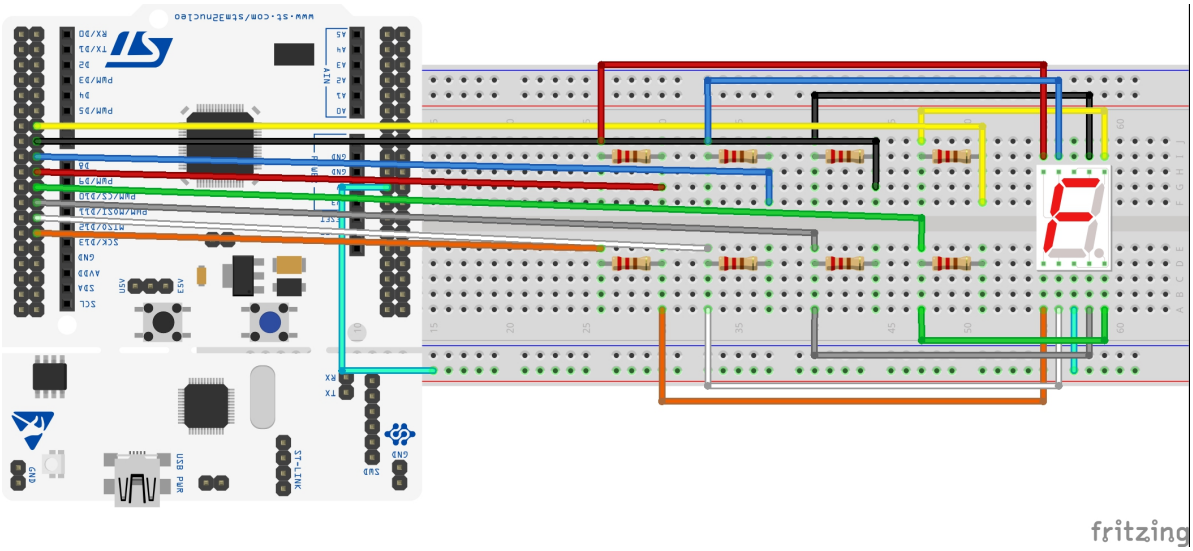
Connect the common anode 7-segment with the given array resistors.

Apply VCC and GND to the 7-segment display.

Apply 'H' to any 7-segment pin 'a'~'g' and observe if that LED is turned on or off

Connection Diagram

Circuit diagram



image

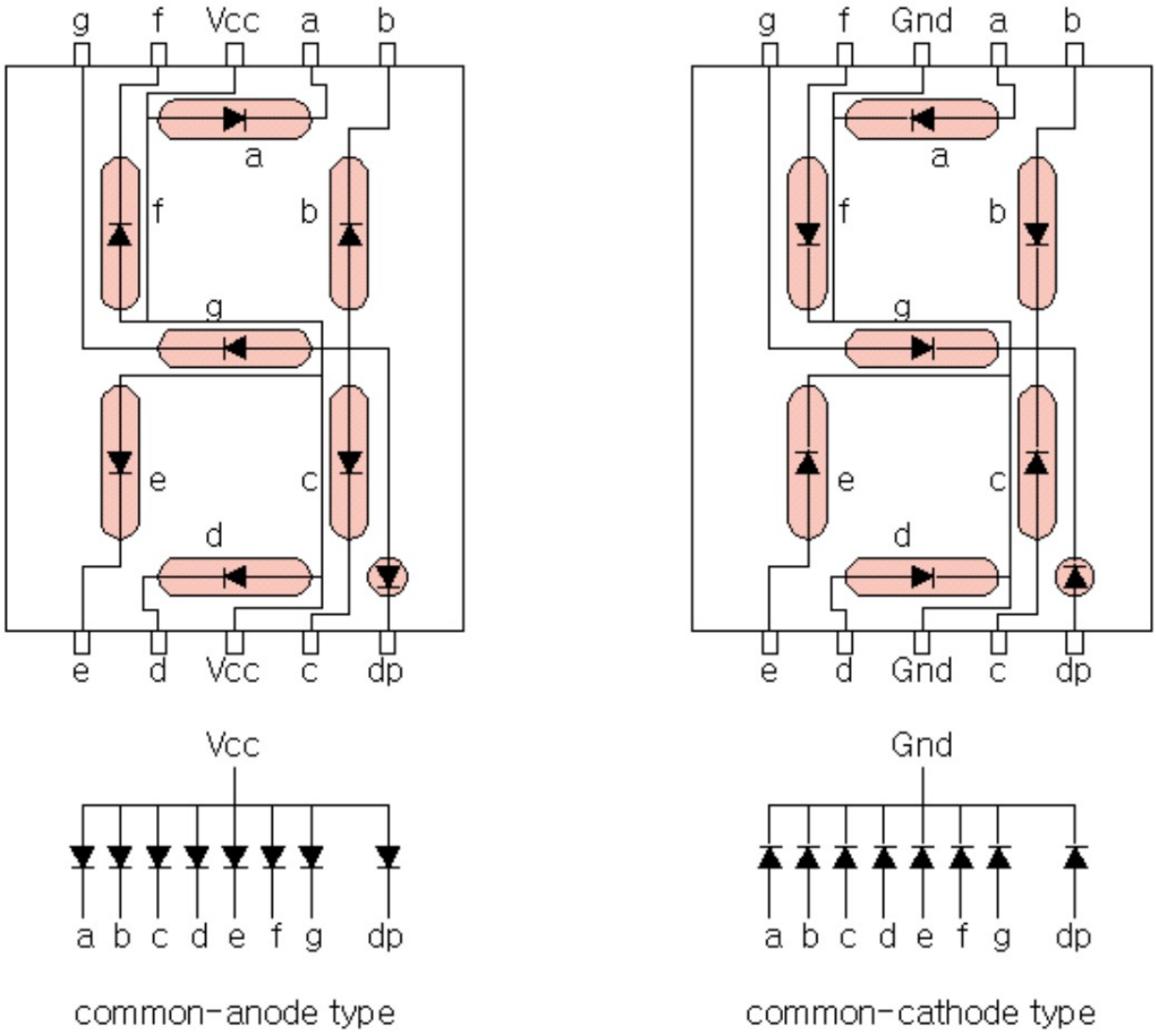
Discussion

1. Draw the truth table for the BCD 7-segment decoder with the 4-bit input.

Decimal	Binary				a	b	c	d	e	f	g	dp
0	0	0	0	0	L	L	L	L	L	L	H	H
1	0	0	0	1	H	L	L	H	H	H	H	H
2	0	0	1	0	L	L	H	L	L	H	L	H
3	0	0	1	1	L	L	L	L	H	H	L	H
4	0	1	0	0	H	L	L	H	H	L	L	H
5	0	1	0	1	L	H	L	L	H	L	L	H
6	0	1	1	0	H	H	L	L	L	L	L	H
7	0	1	1	1	L	L	L	H	H	L	H	H
8	1	0	0	0	L	L	L	L	L	L	L	H
9	1	0	0	1	L	L	L	H	H	L	L	H
dot	1	0	1	0	H	H	H	H	H	H	H	L
11	1	0	1	1	X	X	X	X	X	X	X	X
12	1	1	0	0	X	X	X	X	X	X	X	X
13	1	1	0	1	X	X	X	X	X	X	X	X
14	1	1	1	0	X	X	X	X	X	X	X	X
15	1	1	1	1	X	X	X	X	X	X	X	X

2. What are the common cathode and common anode of 7-segment display?

The common cathode and common anode have opposite results when they are HIGH and LOW, respectively.



3. Does the LED of a 7-segment display (common anode) pin turn ON when 'HIGH' is given to the LED pin from the MCU?

The 7-segment device used in this experiment is a common anode. Therefore, as shown in the picture of problem 2, since Vcc is being supplied, the output of each pin must be LOW in order for a potential difference to occur in the LED.

Problem 2: Display 0~9 with button press

Procedure

1. Create a new project under the directory `\repos\EC\LAB\LAB_GPIO_7segment`

- The project name is "**LAB_GPIO_7segment".**
- Create a new source file named as "**LAB_GPIO_7segment.c"**

2. Include your updated library in `\repos\EC\lib\` to your project.

- **ecGPIO.h, ecGPIO.c**
- **ecRCC.h, ecRCC.c**

- **ecSysTick.h, ecSysTick.c**
- **ecInclude.h**

3. Declare and Define the following functions in your library

- **ecFunc.h, ecFunc.c**

```
void sevensegment_init(void);

void sevensegment_decoder(uint8_t  num);
```

Configuration

- Digital In for Button (B1)
  - Digital In
  - Pin : **PC13**
  - PULL-UP
- Digital Out for 7-Segment
  - Digital Out
  - pin : **PA5, PA6, PA7, PB6, PC7, PA9, PA8, PB10** ('a'~'h', respectively)
  - Push-Pull
  - No Pull-up-Pull-down
  - Medium Speed

Exercise

Port/Pin	Description	Register setting
Port A Pin 5	Clear Pin5 mode	GPIOA→ MODER &=~(3<<(5*2))
Port A Pin 5	Set Pin5 mode = Output	GPIOA→ MODER |=1<<(5*2)
Port A Pin 6	Clear Pin6 mode	GPIOA→ MODER &=~(3<<(6*2))
Port A Pin 6	Set Pin6 mode = Output	GPIOA→ MODER |=1<<(6*2)
Port A Pin Y	Clear PinY mode	GPIOA→ MODER &=~(3<<(Y*2))
Port A Pin Y	Set PinY mode = Output	GPIOA→ MODER |=1<<(Y*2)
Port A Pin 5~9	Clear Pin5~9 mode	GPIOA→ MODER &=~(1023 << (5*2))
	Set Pin5~9 mode = Output	GPIOA→ MODER |= 341 << (5*2)
Port X Pin Y	Clear Pin Y mode	GPIOX→ MODER &=~ (3 << (Y*2))
	Set Pin Y mode = Output	GPIOX→ MODER |= 1 << (Y*2)
Port A Pin5	Set Pin5 otype=push-pull	GPIOA→ OTYPER = (0 << 5)
Port A PinY	Set PinY otype=push-pull	GPIOA→ OTYPER = (0 << Y)
Port A Pin5	Set Pin5 ospeed=Fast	GPIOA → OSPEEDR = 2 << (5*2);
Port A PinY	Set PinY ospeed=Fast	GPIOA → OSPEEDR = 2 << (Y*2);
Port A Pin 5	Set Pin5 PUPD=no pullup/down	GPIOA→ OTYPER = 0 << (5*2);
Port A Pin Y	Set PinY PUPD=no pullup/down	GPIOA→ OTYPER =0 << (Y*2);

Code

[main code github]

```
/**
```

```c
/**********************************************************************************
* @author  SSSLAB
* @Mod       2022-09-29 by Seung-Eun Hwang
* @brief   Embedded Controller:  - LED 7 segment
*
*
**********************************************************************************
*/

#include "ecInclude.h"


void setup(void);

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

    // matrix initialozation -------------------------------------------------

    int cnt = 0;
    int flag = 0;

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

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

            cnt++;
            flag = cnt % 10;
            sevensegment_decoder(flag);

        }

        delay_ms(50);                                       // software
debouncing

    }
}

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, NOPUPD); // button pull up

    // seven segment setup
    sevensegment_init();


}
```

code link

ecFunc.c ecFunc.h

```
void Seven_Segment_init(void) {

    // led a setup
    GPIO_init(GPIOA, PA8, OUTPUT);      // calls RCC_GPIOA_enable()
    GPIO_otype(GPIOA, PA8, PUSHPULL);   // LED open drain
    GPIO_pupd(GPIOA, PA8, NOPUPD);      // LED NOPUPD
    GPIO_ospeed(GPIOA, PA8, MSPEED);        // LED Medium speed
    GPIO_write(GPIOA, PA8, LOW);        // claer LED

    // led b setup
    GPIO_init(GPIOB, PB10, OUTPUT);     // calls RCC_GPIOA_enable()
    GPIO_otype(GPIOB, PB10, PUSHPULL);  // LED open drain
    GPIO_pupd(GPIOB, PB10, NOPUPD);     // LED NOPUPD
    GPIO_ospeed(GPIOB, PB10, MSPEED);       // LED Medium speed
    GPIO_write(GPIOB, PB10, LOW);       // claer LED

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

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

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

    // led f setup
    GPIO_init(GPIOA, PA9, OUTPUT);      // calls RCC_GPIOA_enable()
    GPIO_otype(GPIOA, PA9, PUSHPULL);   // LED open drain
    GPIO_pupd(GPIOA, PA9, NOPUPD);      // LED NOPUPD
    GPIO_ospeed(GPIOA, PA9, MSPEED);        // LED Medium speed
    GPIO_write(GPIOA, PA9, LOW);        // claer LED

    // led g setup
    GPIO_init(GPIOC, PC7, OUTPUT);      // calls RCC_GPIOA_enable()
    GPIO_otype(GPIOC, PC7, PUSHPULL);   // LED open drain
    GPIO_pupd(GPIOC, PC7, NOPUPD);      // LED NOPUPD
    GPIO_ospeed(GPIOC, PC7, MSPEED);        // LED Medium speed
    GPIO_write(GPIOC, PC7, HIGH);       // claer LED

    // led DP setup
    GPIO_init(GPIOB, PB6, OUTPUT);      // calls RCC_GPIOA_enable()
```

```
    GPIO_otype(GPIOB, PB6, PUSHPULL);   // LED open drain
    GPIO_pupd(GPIOB, PB6, NOPUPD);      // LED NOPUPD
    GPIO_ospeed(GPIOB, PB6, MSPEED);        // LED Medium speed
    GPIO_write(GPIOB, PB6, HIGH);       // claer LED

}


void Seven_segment_decoder(int flag) {

    int seven_segment[11][8] = {
                {LOW,   LOW,    LOW,    LOW,    LOW,    LOW,    HIGH,   HIGH},
       //zero
                {HIGH,  LOW,    LOW,    HIGH,   HIGH,   HIGH,   HIGH,   HIGH},
       //one
                {LOW,   LOW,    HIGH,   LOW,    LOW,    HIGH,   LOW,    HIGH},
       //two
                {LOW,   LOW,    LOW,    LOW,    HIGH,   HIGH,   LOW,    HIGH},
       //three
                {HIGH,  LOW,    LOW,    HIGH,   HIGH,   LOW,    LOW,    HIGH},
       //four
                {LOW,   HIGH,   LOW,    LOW,    HIGH,   LOW,    LOW,    HIGH},
       //five
                {LOW,   HIGH,   LOW,    LOW,    LOW,    LOW,    LOW,    HIGH},
       //six
                {LOW,   LOW,    LOW,    HIGH,   HIGH,   HIGH,   HIGH,   HIGH},
       //seven
                {LOW,   LOW,    LOW,    LOW,    LOW,    LOW,    LOW,    HIGH},
       //eight
                {LOW,   LOW,    LOW,    HIGH,   HIGH,   LOW,    LOW,    HIGH},
       //nine
                {HIGH,  HIGH,   HIGH,   HIGH,   HIGH,   HIGH,   HIGH,   LOW}
     //dot
    };


    GPIO_write(GPIOA, PA8, seven_segment[flag][0]);         // a
    GPIO_write(GPIOB, PB10, seven_segment[flag][1]);        // b
    GPIO_write(GPIOA, PA7, seven_segment[flag][2]);         // c
    GPIO_write(GPIOA, PA6, seven_segment[flag][3]);         // d
    GPIO_write(GPIOA, PA5, seven_segment[flag][4]);         // e
    GPIO_write(GPIOA, PA9, seven_segment[flag][5]);         // f
    GPIO_write(GPIOC, PC7, seven_segment[flag][6]);         // g
    GPIO_write(GPIOB, PB6, seven_segment[flag][7]);         // dp


}
```

ecGPIO.c ecGPIO.h

ecRCC.h ecGPIO.h

ecSysTick.c ecSysTick.h

ecSysTick.c ecSysTick.h

ecInclude.h

Results

Experiment images and results

7 segment demo video link

Reference

ykkim's github

Troubleshooting

Since the names of the pins of GPIO are similar, it is difficult to debug when the code is written incorrectly.

So, once you write the code, you have to focus on writing it.