LAB: GPIO Digital InOut 7-segment

Date: 2022-10-10

Author/Partner: Seongjun Park/ Dongmin Kim

Github: https://github.com/PotPung/EC_sjpark_262/tree/main/docs/LAB_GPIO_7segment_21600262_%EB%B0%95%EC%84%B1%EC%A4%80

Demo Video: https://youtu.be/zCFLeOTo1D0

PDF version: 1.7(Acrobat 8.x)

Introduction

In this lab, I create a program that controls the 7-segment. Each time the button is pressed, the 7-segment display increases by 1 from 0 to 9. If the button is pressed when it becomes 9, reset it to 0. Through this, it is possible to understand the hardware principle that the lead of the 7-segment operates and the connection method of the 7-segment.

Requirement

Hardware

- MCU
    - NUCLEO-F401RE
- Actuator/Sensor/Others:
    - 7-segment display(5101ASR)
    - Array resistor (330 ohm)
    - breadboard

Software

- Keil uVision, CMSIS, EC_HAL library

Problem 1: Connecting 7-Segment

Procedure

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

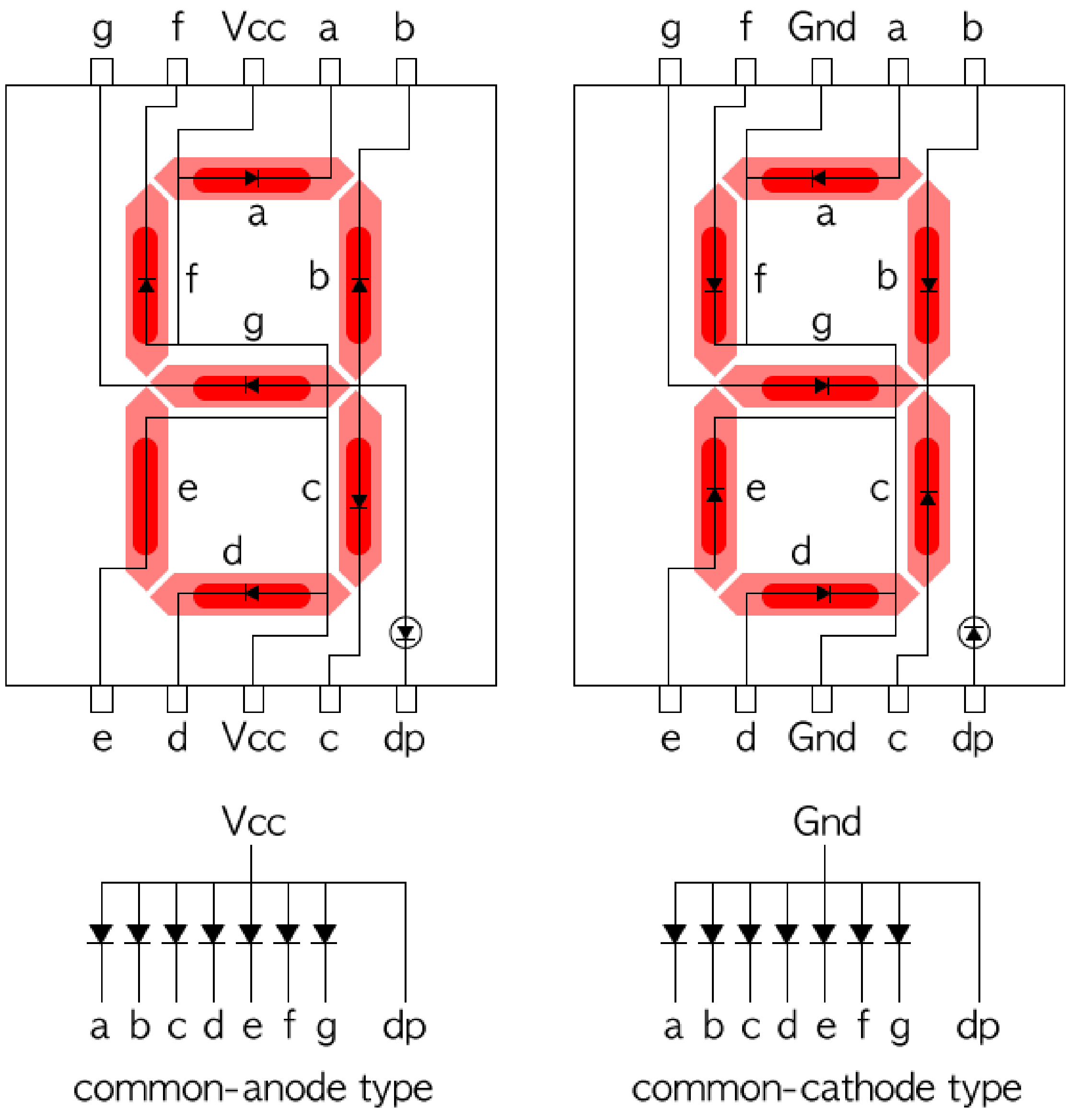

Figure 1. 7-segment connection

1. Connect the common anode 7-segment with the given array resistors.
2. Apply VCC and GND to the 7-segment display.
3. Apply 'H' to any 7-segment pin 'a'~'g' and observe if that LED is turned on or off

Connection Diagram

7segment Circuit diagram

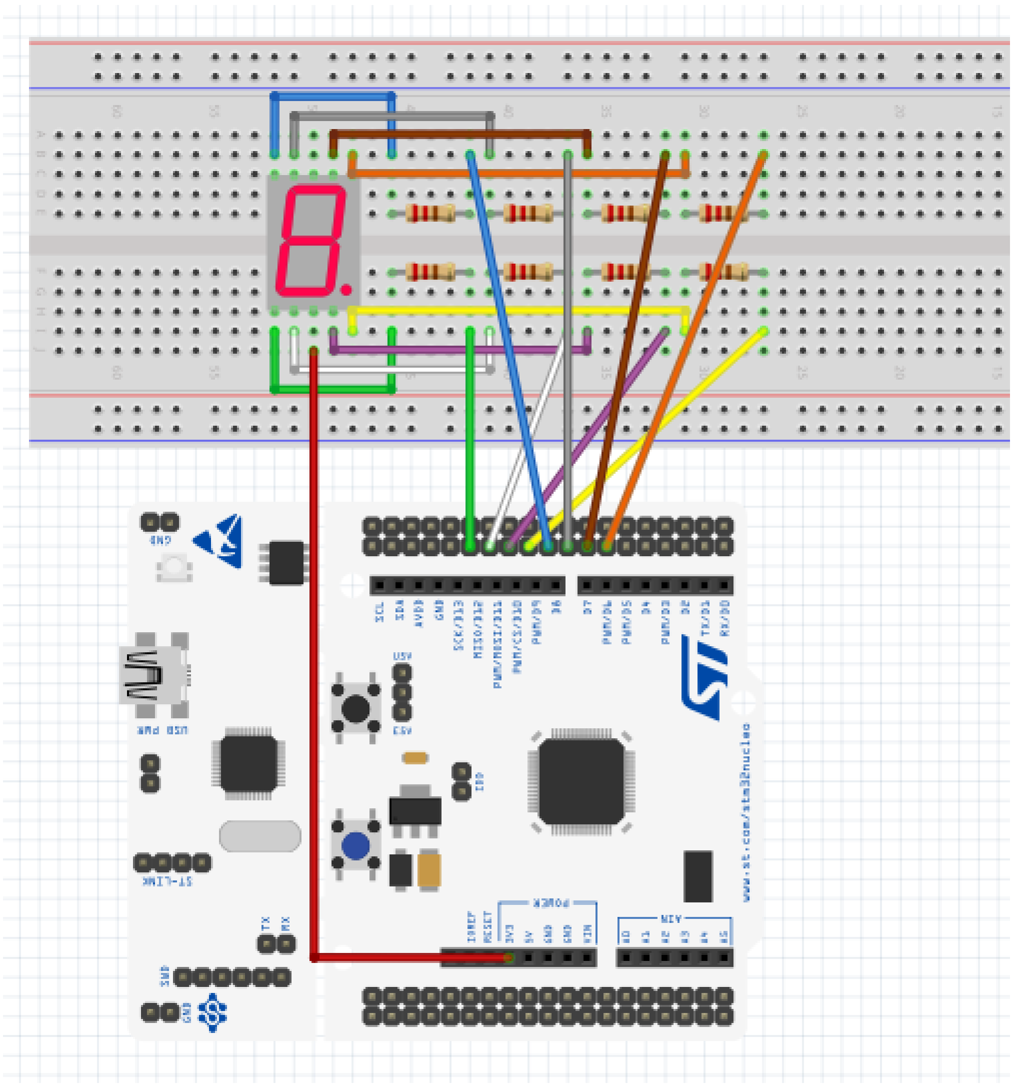

Figure 2. 7-segment Circuit diagram

Discussion

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

Binary Pattern (4-bit input)				a	b	c	d	e	f	g	h	Decimal Display
0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	1	1	0	0	1	1	1	1	0	1
0	0	1	0	0	0	1	0	0	1	0	0	2
0	0	1	1	0	0	0	0	1	1	0	0	3
0	1	0	0	1	0	0	1	1	0	0	0	4
0	1	0	1	0	1	0	0	1	0	0	0	5
0	1	1	0	0	1	0	0	0	0	0	0	6
0	1	1	1	0	0	0	1	1	0	1	0	7
1	0	0	0	0	0	0	0	0	0	0	0	8
1	0	0	1	0	0	0	0	1	0	0	0	9
1	0	1	0	X	X	X	X	X	X	X	X	10
1	0	1	1	X	X	X	X	X	X	X	X	11
1	1	0	0	X	X	X	X	X	X	X	X	12
1	1	0	1	X	X	X	X	X	X	X	X	13
1	1	1	0	X	X	X	X	X	X	X	X	14
1	1	1	1	X	X	X	X	X	X	X	X	15

Figure 3. Truth table for BCD 7-segment with 4-bit input

This table shows the common-cathode type in Figure 1, and h means Dp.

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

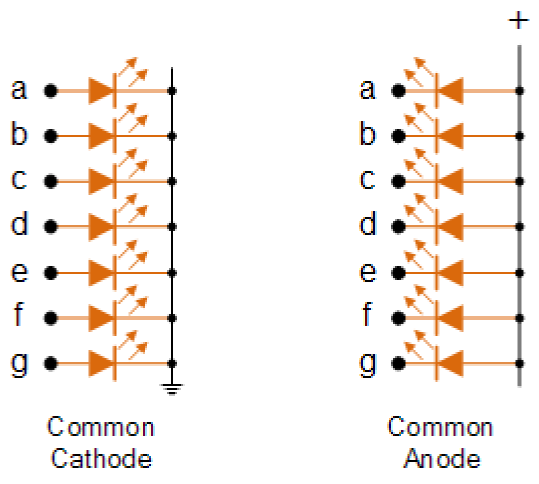

Figure 4. Common Cathode & Common Anode

In the Common Cathode, all cathodes are commonly connected to the ground like the left picture in Figure 4, and in the Common Anode, all anodes are commonly connected to + poles like the right picture in Figure 4. In other words, since the voltage difference occurs only when the input of the common cathode is "H", the LED is turned on, and the voltage difference occurs only when the input of the common cathode is "L", so the LED is turned on.

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?

Based on the answer No. 2, when 'HIGH' is given as the MCU's LED pin, the LED of the 7-segment display (common anode) pin is turned off.

Problem 2: Display 0~9 with button press

Procedure

1. setup - configuration initialization
2. 7-segment decode function
   - The values of 0[LOW-LED ON],1[HIGH-LED OFF] for each led(a~h) are expressed using a 2D array according to each display of numbers 0 to 9.
   - When the button is pressed, it counts, and select the display array that matches the counted number.
3. Debounce through delay.

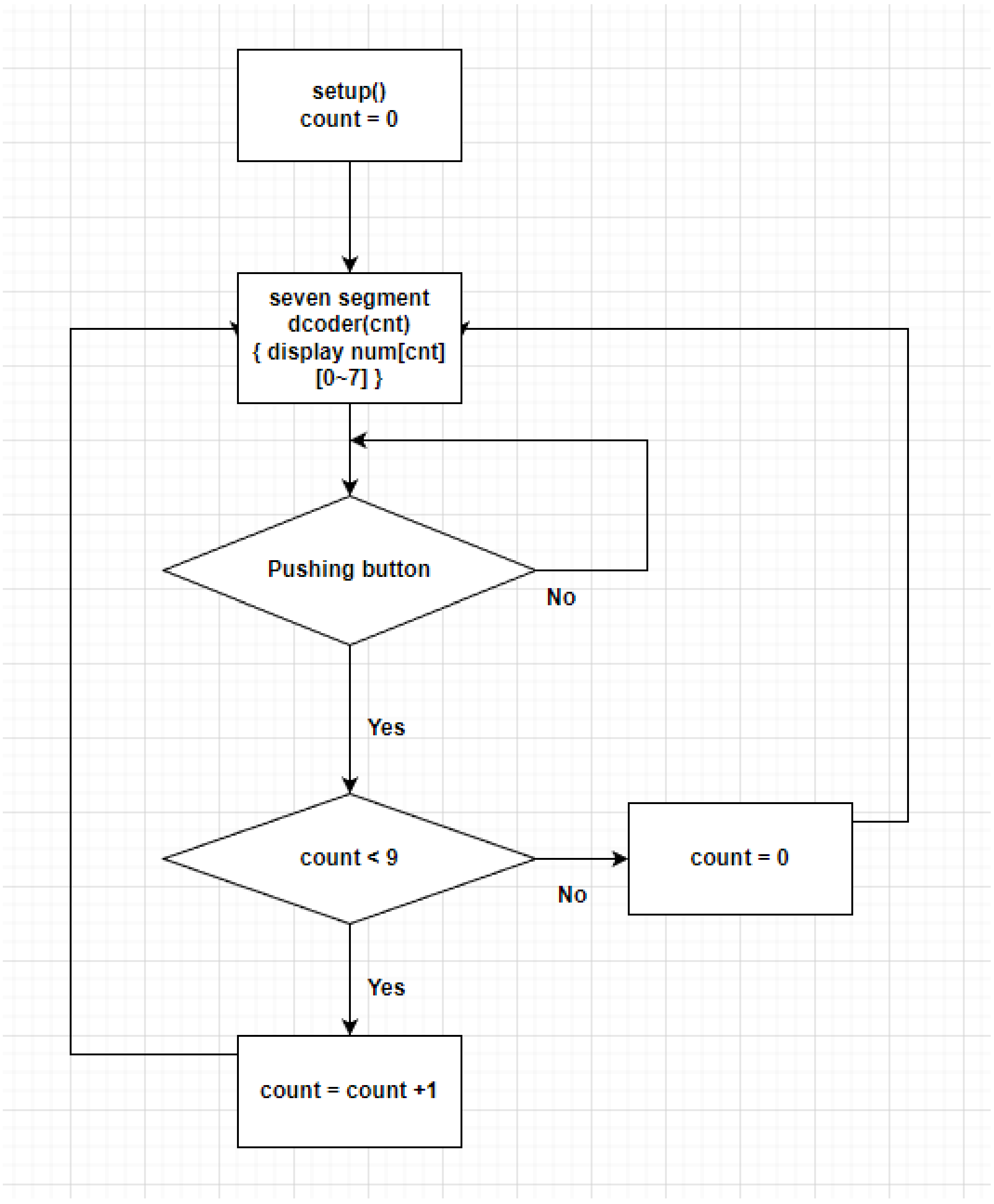

Figure 5. Flow chart for 7-segment

Configuration

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

Exercise

Fill in the table

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<<(2*Y)
Port A Pin 5~9	Clear Pin5~9 mode	GPIOA->MODER &= ~(0x3FF<<(2*5))
	Set Pin5~9 mode = Output	GPIOA->MODER |= 0x155<<(2*5)
Port X Pin Y	Clear Pin Y mode	GPIOX->MODER &= ~(3<<(Y*2))
	Set Pin Y mode = Output	GPIOX->MODER |= 1<<(2*Y)
Port A Pin 5	Set Pin5 otype=push-pull	GPIOA->OTYPER = 0<<5
Port A Pin Y	Set PinY otype=push-pull	GPIOA-> OTYPER = 0<<Y
Port A Pin 5	Set Pin5 ospeed=Fast	GPIOA->OSPEEDR = 2<<(2*5)
Port A Pin Y	Set PinY ospeed=Fast	GPIOA-> OSPEEDR = 2<<(2*Y)
Port A Pin 5	Set Pin5 PUPD=no pullup/down	GPIOA->PUPDR = 0<<(2*5)
Port A Pin Y	Set PinY PUPD=no pullup/down	GPIOA->PUPDR = 0<<(2*Y)

Code

- **Lab source code**:
- **Main code** - LAB_GPIO_7segment.c

```c
int main(void) {
    // Initialiization ---------------------------------------------------------
    setup();
    unsigned int cnt = 0; //count

    // Inifinite Loop ----------------------------------------------------------
    while(1){
        sevensegment_decode(cnt % 10);
        if(GPIO_read(GPIOC, BUTTON_PIN) == 0) cnt++; // count by pushing button
        if (cnt > 9) cnt = 0; // count initialization
        for(int i = 0; i < 300000;i++){} //debouncing
    }
}
```

- **Setup/initialization code** - LAB_GPIO_7segment.c

```c
void setup(void)
{
    RCC_HSI_init();
    // Dgital In for Button(B1)
    GPIO_init(GPIOC, BUTTON_PIN, INPUT); // calls RCC_GPIOC_enable()
    GPIO_pupd(GPIOC, BUTTON_PIN, EC_PD); // Button pin -> pull down

    // Dgital Out for 7-Segment
    sevensegment_init();
}
```

- **Sevensegment initialization(configuration) code** - ecGPIO.c

```c
void sevensegment_init(void){
    for (int i = PIN_5; i< PIN_11 ; i++){ // led a, b, c, d, e, f
        int gpioType = 0;
        if (i == PIN_10){
            gpioType = GPIOB;
        }
        else{
            gpioType = GPIOA;
        }
    GPIO_init(gpioType, i, OUTPUT);
    GPIO_otype(gpioType, i, EC_PP); // Push-Pull
    GPIO_pupd(gpioType, i, EC_NON); // NO Pull-up-Pull-down
    GPIO_ospeed(gpioType, i, EC_MS); // Medium speed
    }

    GPIO_init(GPIOC, PIN_7, OUTPUT); //led g
    GPIO_otype(GPIOC, PIN_7, EC_PP); //Push-Pull
    GPIO_pupd(GPIOC, PIN_7, EC_NON); // NO Pull-up-Pull-down
    GPIO_ospeed(GPIOC, PIN_7, EC_MS); // Medium speed

    GPIO_init(GPIOB, PIN_6, OUTPUT); // led h
    GPIO_otype(GPIOB, PIN_6, EC_PP); //Push-Pull
    GPIO_pupd(GPIOB, PIN_6, EC_NON); // NO Pull-up-Pull-down
    GPIO_ospeed(GPIOB, PIN_6, EC_MS); // Medium speed
}
```

- **2D array for 7-segment & sevensegment decode code** - ecGPIO.c

```
int number[10][8]={                                 //{a,b,c,d,e,f,g,h}
                   {0,0,0,0,0,0,1,1},          //zero '0'
                   {1,0,0,1,1,1,1,1},          //one '1'
                   {0,0,1,0,0,1,0,1},          //two '2'
                   {0,0,0,0,1,1,0,1},          //three '3'
                   {1,0,0,1,1,0,0,1},          //four '4'
                   {0,1,0,0,1,0,0,1},          //five '5'
                   {0,1,0,0,0,0,0,1},          //six '6'
                   {0,0,0,1,1,0,1,1},          //seven '7'
                   {0,0,0,0,0,0,0,1},          //eight '8'
                   {0,0,0,0,1,0,0,1},          //nine '9'
                 };

void sevensegment_decode (uint8_t num){
    // LEDs for each of the 7-segment
    GPIO_write(GPIOA, 8, number[num][0]); //led a
    GPIO_write(GPIOB, 10, number[num][1]); //led b
    GPIO_write(GPIOA, 7, number[num][2]); //led c
    GPIO_write(GPIOA, 6, number[num][3]); //led d
    GPIO_write(GPIOA, 5, number[num][4]); //led e
    GPIO_write(GPIOA, 9, number[num][5]); //led f
    GPIO_write(GPIOC, 7, number[num][6]); //led g
    GPIO_write(GPIOB, 6, number[num][7]); //led g
}
```

Results

video link : https://youtu.be/zCFLeOTo1D0

Conclusion

Through this lab, I was able to learn the correct connection through understanding the 7-segment hardware. The 7-segment used this time is a common-anode type, and the LED becomes ON when the LOW input is input through the MCU, and the LED becomes OFF when the HIGH input is input. Based on this, when creating a code that controls the 7-segment, it was possible to create a much more efficient code using the 2D-arrary.

Reference

Display Decoder. (n.d.). Electroncis Tutorials. https://www.electronics-tutorials.ws/combination/comb_6.html

Troubleshooting

When creating the code for the first time, the 'h' portion corresponding to the dp_led was completely excluded from the 2D-array. As a result, there was no problem, but when it was necessary to control the dp part, it was determined that it was necessary to add the dp part to be easily used by anyone and went through the process of revising it again. I looked back on the role of an engineer not only to think about the results in front of me, but also to consider various results.