[LAB#3] GPIO Digital InOut 7-segment

Date: Dec.6.2023

Author: 21900416 Gyeonheal An

Github: https://github.com/AnGyeonheal/Embedded_Control

Demo Video: https://www.youtube.com/watch?v=A_POSQJDsKc&ab_channel=%ED%95%9C%EB%8F%99%EB%8C%80%ED%95%99%EA%B5%90%EC%95%88%EA%B2%AC%ED%9E%90

I. 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.

i. Requirement

Hardware

- MCU
  - NUCLEO-F411RE
- Actuator/Sensor/Others:
  - 7-segment display(5101ASR)
  - Array resistor (330 ohm)
  - breadboard

Software

- IDE: Clion(with PlatformIO core plugin)
  Library: STM32Cube library package(Official), EC_HAL library
  Compiler: GNU Arm Embedded Toolchain
  Debugger: ST-Link

ii. Exercise

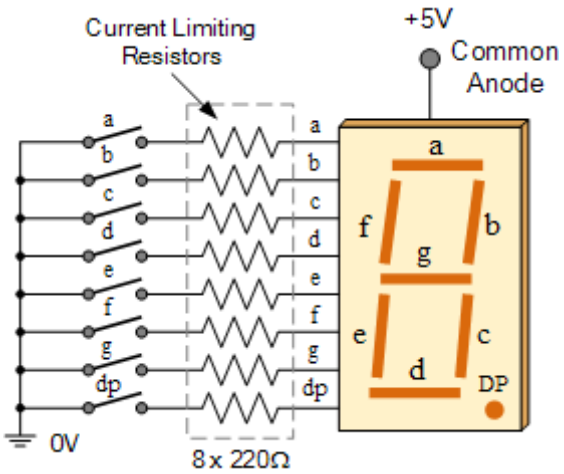
Port/ Pin	Description	Register Setting
Port A Pin 5	Clear Pin5 mode	GPIOA->MODER &=~(3<<(5*2))
Port A Pin 5	Set Pin5 mode = Output	GPIOA->MODER |= (1<<5*2)
Port A Pin 6	Clear Pin6 mode	GPIOA->MODER &=~ ~(3<<(6*2))
Port A Pin 6	Set Pin6 mode = Output	GPIOA->MODER |= (1<<6*2)
Port A Pin Y	Clear PinY mode	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 &=~(0x3FF<<(5*2))
	Set Pin5~9 mode = Output	GPIOA->MODER |= (0x155<<(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 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<<(5*2))
Port A Pin Y	Set PinY ospeed=Fast	GPIOA-> OSPEEDR |=(2<<(Y*2))
Port A Pin 5	Set Pin5 PUPD=no pullup/down	GPIOA->PUPDR |= (0<<(5*2))
Port A Pin Y	Set PinY PUPD=no pullup/down	GPIOA->PUPDR |= (0<<(Y*2))

II. Problem 1: Connecting 7-Segment Display

i. 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.



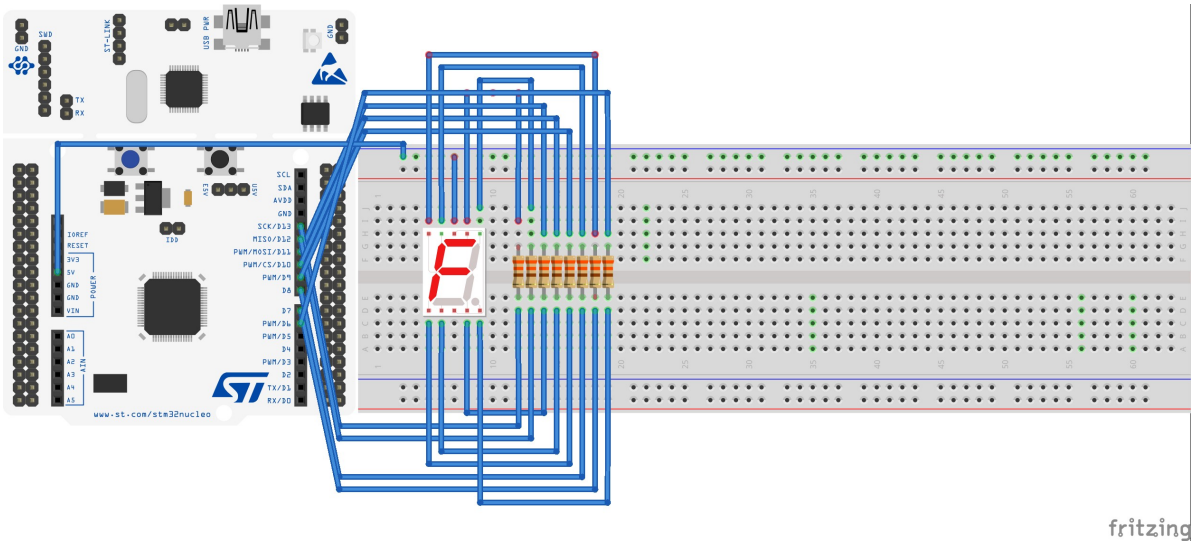
Connect the common anode 7-segment with the given array resistors.

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

- example: Set 'H' on PA5 of MCU and connect to 'a' of the 7-segment.

ii. Connection Diagram

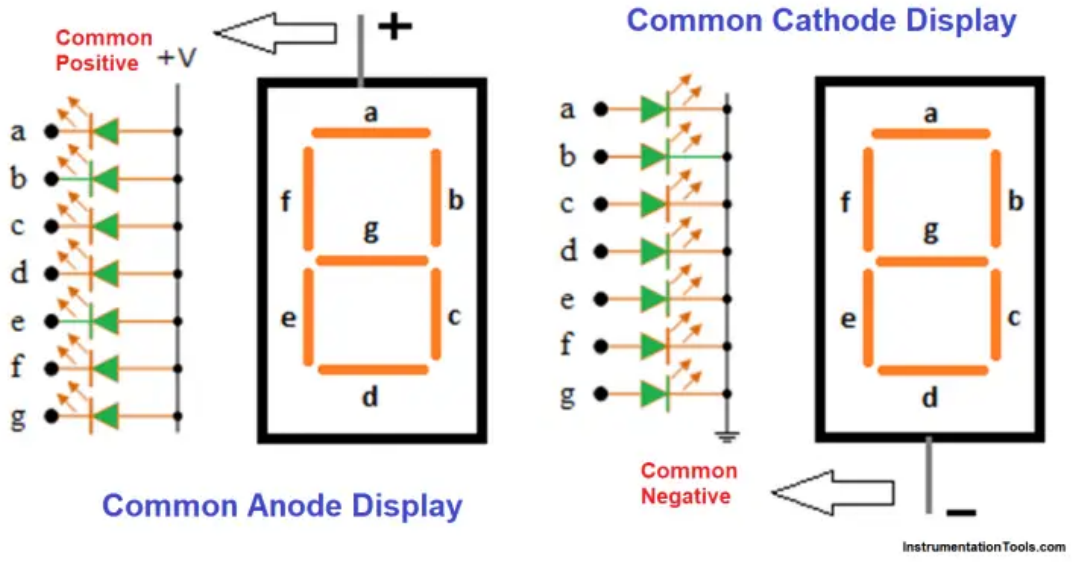


iii. Discussion

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

DECIMAL	D	C	B	A	a	b	c	d	e	f	g
0	L	L	L	L	L	L	L	L	L	L	H
1	L	L	L	H	H	L	L	H	H	H	H
2	L	L	H	L	L	L	H	L	L	H	L
3	L	L	H	H	L	L	L	L	H	H	L
4	L	H	L	L	H	L	L	H	H	L	L
5	L	H	L	H	L	H	L	L	H	L	L
6	L	H	H	L	H	H	L	L	L	L	L
7	L	H	H	H	L	L	L	H	H	H	H
8	H	L	L	L	L	L	L	L	L	L	L
9	H	L	L	H	L	L	L	H	H	L	L
X	X	X	X	X	X	X	X	X	X	X	X

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



Common cathode 7-segment display

- In a common cathode display, all the cathodes (negative terminals) of the LED segments are connected together and shared as a common connection.

- The anodes (positive terminals) of each segment are connected to individual pins or lines.
- To light up a particular segment in a common cathode display, you apply a positive voltage (typically 5V or 3.3V) to the anode of that segment while grounding the common cathode.
- When we ground the common cathode, the LED segments with a positive voltage on their anodes will light up, displaying the desired character.

Common anode 7-Segment display

- In a common anode display, all the anodes (positive terminals) of the LED segments are connected together and shared as a common connection.
- The cathodes (negative terminals) of each segment are connected to individual pins or lines.
- To light up a particular segment in a common anode display, you apply ground (0V) to the cathode of that segment while applying a positive voltage (typically 5V or 3.3V) to the common anode.
- When we apply a positive voltage to the common anode, the LED segments with their cathodes grounded will light up, displaying the desired character.

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?**

   No. It turn ON when 'LOW' is given to the LED pin for MCU

III. Problem 2: Display 0~9 with button press

i. Procedure

1. Create a new project under the directory `\repos\EC\LAB\LAB_GPIO_7segment`
2. Include your updated library in `\repos\EC\lib\` to your project.

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

3. Declare and Define the following functions in your library

ecGPIO.h

```
void seven_segment_init(void);
void sevensegment_decoder(uint8_t  num);
```

- First, check if every number, 0 to 9, can be displayed properly
- Then, create a code to display the number from 0 to 9 with each button press. After the number '9', it should start from '0' again.

ii. 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

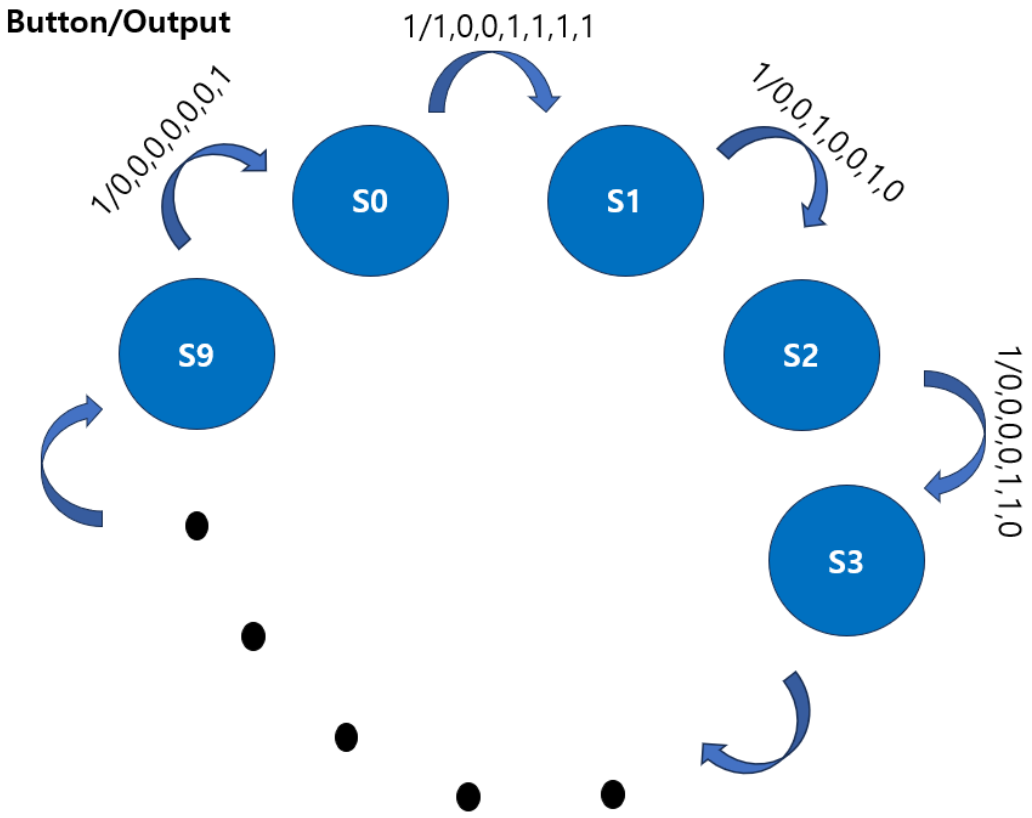
iii. Code

main.c

```
/*----------------------------------------------------------------\
@ Embedded Controller by Young-Keun Kim - Handong Global University
Author           : Gyeonheal An
Created          : 05-03-2021
Modified         : 10-03-2023
Language/ver     : C++ in Keil uVision

Description      : LAB_GPIO_7segment
/----------------------------------------------------------------*/

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

unsigned char state = S0;
unsigned  char next_state = S0;
unsigned int input = 1;

void setup(void);

int main(void) {
    int delay=0;

    // Initialization --------------------------------------------------------
    setup();
    //------------------------ problem 2 ------------------------
    seven_segment_decode(S0);
    GPIO_write(GPIOB, pin_dp, HIGH);      // DP

    //------------------------ problem 3 ------------------------
    // sevensegment_display(S0);

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

        if((GPIO_read(GPIOC, BUTTON_PIN) == 0) && (delay > 100000)){
            input = 0;
            next_state = FSM[state].next[input];
            state = next_state;
            // ---------------- problem 2 ----------------
            seven_segment_decode(state);
            // ---------------- problem 3 ----------------
            // sevensegment_display(state);

            delay = 0;
            input = 1;
        }
        delay++;
    }
}

void setup(void)
{
```

```
    RCC_HSI_init();
    GPIO_init(GPIOC, BUTTON_PIN, INPUT);  // calls RCC_GPIOC_enable()
    GPIO_pupd(GPIOC, BUTTON_PIN, EC_PU);
    // seven_segment_init();
    sevensegment_display_init();
}
```

The basic setting of GPIO pins was performed through the setup function, the initialization was performed through the step_segment_decode() function, and the DP also fixed the output to zero. Loop() switches to the next state when the button is pressed and after the delay time passes, and outputs the output of the corresponding state. Then, prepare to receive the next button input by initializing the next delay and input.



This is FSM state graph of 7-segment with out Decoder.

GPIO.c

```
State_P1 FSM[10] = {
        {{S1, S0}, {0,0,0,0,0,0,1}},    // 0
        {{S2, S1}, {1,0,0,1,1,1,1}},    // 1
        {{S3, S2}, {0,0,1,0,0,1,0}},    // 2
        {{S4, S3}, {0,0,0,0,1,1,0}},    // 3
        {{S5, S4}, {1,0,0,1,1,0,0}},    // 4
        {{S6, S5}, {0,1,0,0,1,0,0}},    // 5
        {{S7, S6}, {1,1,0,0,0,0,0}},    // 6
        {{S8, S7}, {0,0,0,1,1,1,1}},    // 7
        {{S9, S8}, {0,0,0,0,0,0,0}},    // 8
        {{S0, S9}, {0,0,0,1,1,0,0}}     // 9

};
```

The above FSM graph was implemented as a structure. And this was called from the header file below.

```
// GPIO.h
typedef struct{
    unsigned int next[2];
    unsigned int out[7];
} State_P1;

extern State_P1 FSM[10];  // {{next_state}, {a,b,c,d,e,f,g}}
FSM[state].next[Button_input]
```

```
void seven_segment_init(void){
    RCC_HSI_init();
    GPIO_init(GPIOA, pin_a, OUTPUT);
    GPIO_init(GPIOA, pin_b, OUTPUT);
    GPIO_init(GPIOA, pin_c, OUTPUT);
    GPIO_init(GPIOB, pin_d, OUTPUT);
    GPIO_init(GPIOC, pin_e, OUTPUT);
    GPIO_init(GPIOA, pin_f, OUTPUT);
    GPIO_init(GPIOA, pin_g, OUTPUT);
    GPIO_init(GPIOB, pin_dp, OUTPUT);

    GPIO_otype(GPIOA, pin_a, EC_PUSH_PULL);
    GPIO_otype(GPIOA, pin_b, EC_PUSH_PULL);
    GPIO_otype(GPIOA, pin_c, EC_PUSH_PULL);
    GPIO_otype(GPIOB, pin_d, EC_PUSH_PULL);
    GPIO_otype(GPIOC, pin_e, EC_PUSH_PULL);
    GPIO_otype(GPIOA, pin_f, EC_PUSH_PULL);
    GPIO_otype(GPIOA, pin_g, EC_PUSH_PULL);
    GPIO_otype(GPIOB, pin_dp, EC_PUSH_PULL);

    GPIO_pupd(GPIOA, pin_a, EC_NONE);
    GPIO_pupd(GPIOA, pin_b, EC_NONE);
    GPIO_pupd(GPIOA, pin_c, EC_NONE);
    GPIO_pupd(GPIOB, pin_d, EC_NONE);
    GPIO_pupd(GPIOC, pin_e, EC_NONE);
    GPIO_pupd(GPIOA, pin_f, EC_NONE);
    GPIO_pupd(GPIOA, pin_g, EC_NONE);
    GPIO_pupd(GPIOB, pin_dp, EC_NONE);

    GPIO_ospeed(GPIOA, pin_a, EC_MEDIUM);
    GPIO_ospeed(GPIOA, pin_b, EC_MEDIUM);
    GPIO_ospeed(GPIOA, pin_c, EC_MEDIUM);
    GPIO_ospeed(GPIOB, pin_d, EC_MEDIUM);
    GPIO_ospeed(GPIOC, pin_e, EC_MEDIUM);
    GPIO_ospeed(GPIOA, pin_f, EC_MEDIUM);
    GPIO_ospeed(GPIOA, pin_g, EC_MEDIUM);
    GPIO_ospeed(GPIOB, pin_dp, EC_MEDIUM);
}
```

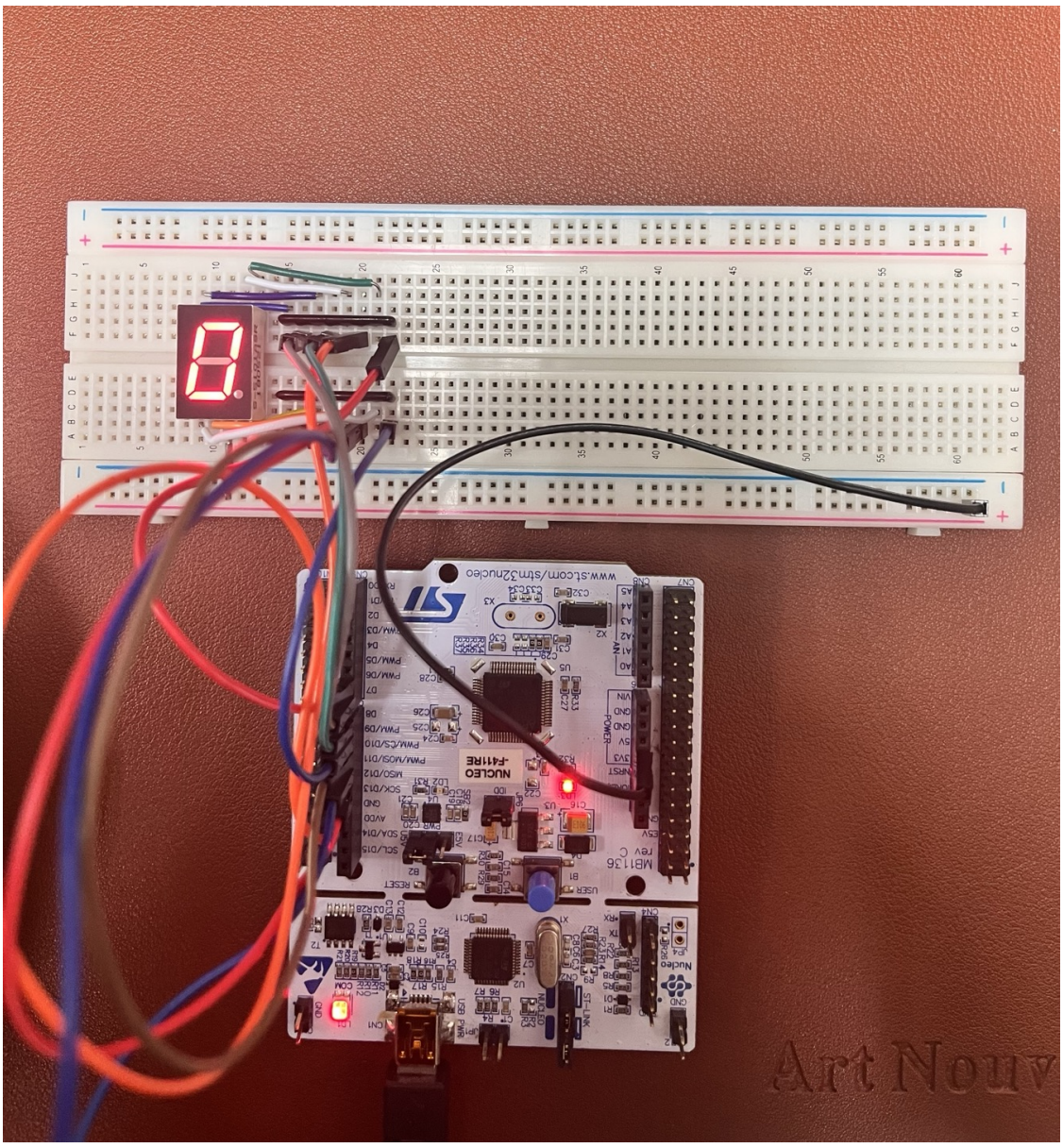
```
void seven_segment_decode(uint8_t state){
    unsigned int a, b, c, d, e, f, g;

    a = FSM[state].out[0];
    b = FSM[state].out[1];
```

```
    c = FSM[state].out[2];
    d = FSM[state].out[3];
    e = FSM[state].out[4];
    f = FSM[state].out[5];
    g = FSM[state].out[6];

    GPIO_write(GPIOA, pin_a, a);     // a
    GPIO_write(GPIOA, pin_b, b);     // b
    GPIO_write(GPIOA, pin_c, c);     // c
    GPIO_write(GPIOB, pin_d, d);     // d
    GPIO_write(GPIOC, pin_e, e);     // e
    GPIO_write(GPIOA, pin_f, f);     // f
    GPIO_write(GPIOA, pin_g, g);     // g
}
```

The function "sevent_segment_decode()" takes the output corresponding to the input state from the FSM structure and outputs it to the LED.

iv. Results

Circuit of 7-segment without Decode



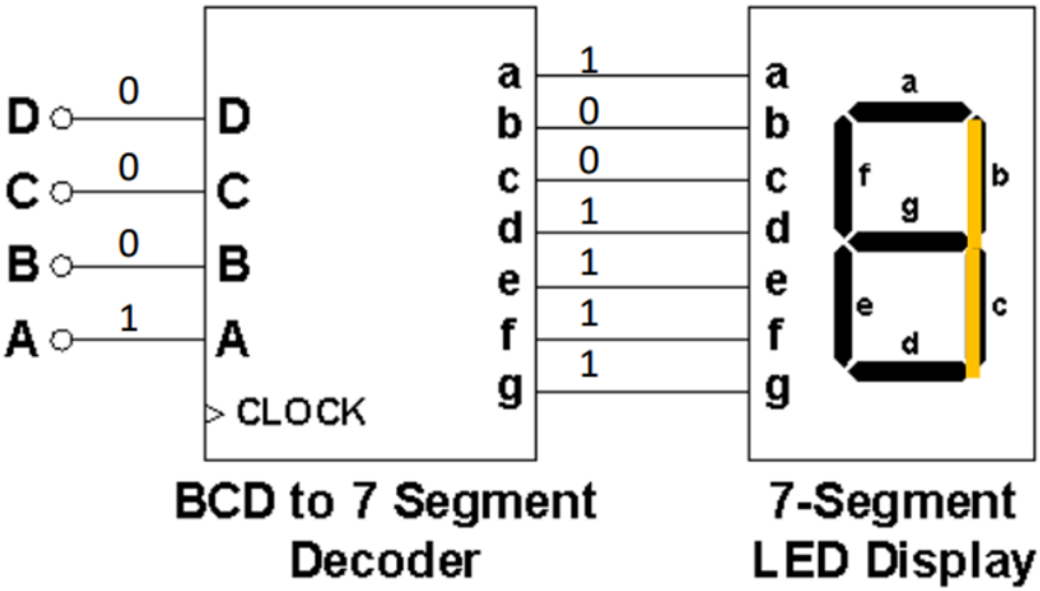
Demo Video

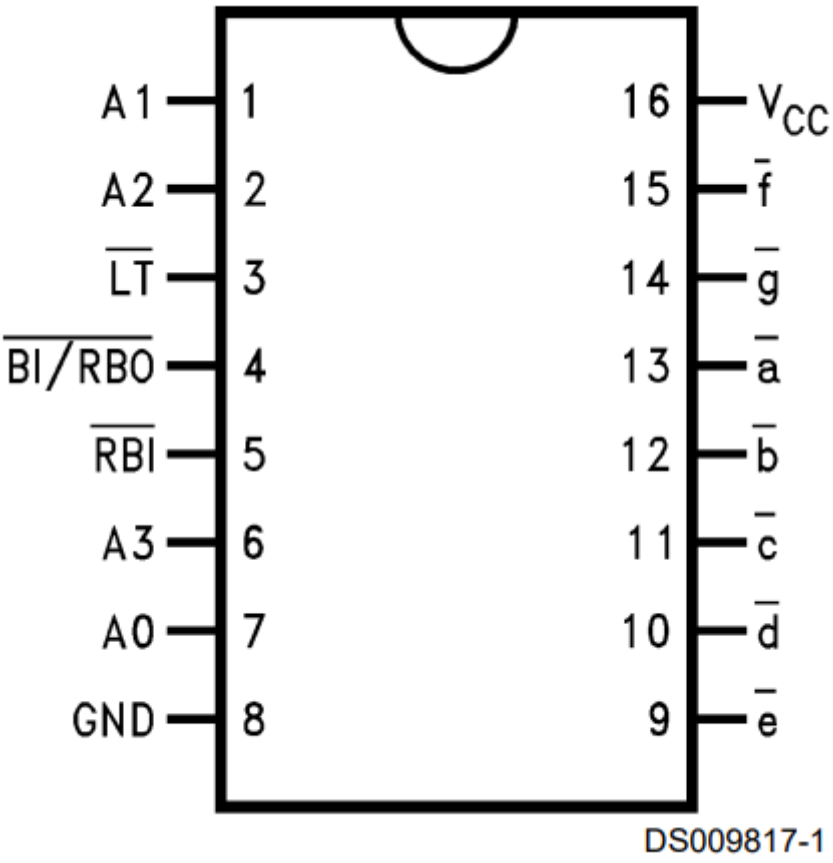
III. Problem 3: Using both 7-Segment Decoder and 7-segment display

i. Procedure

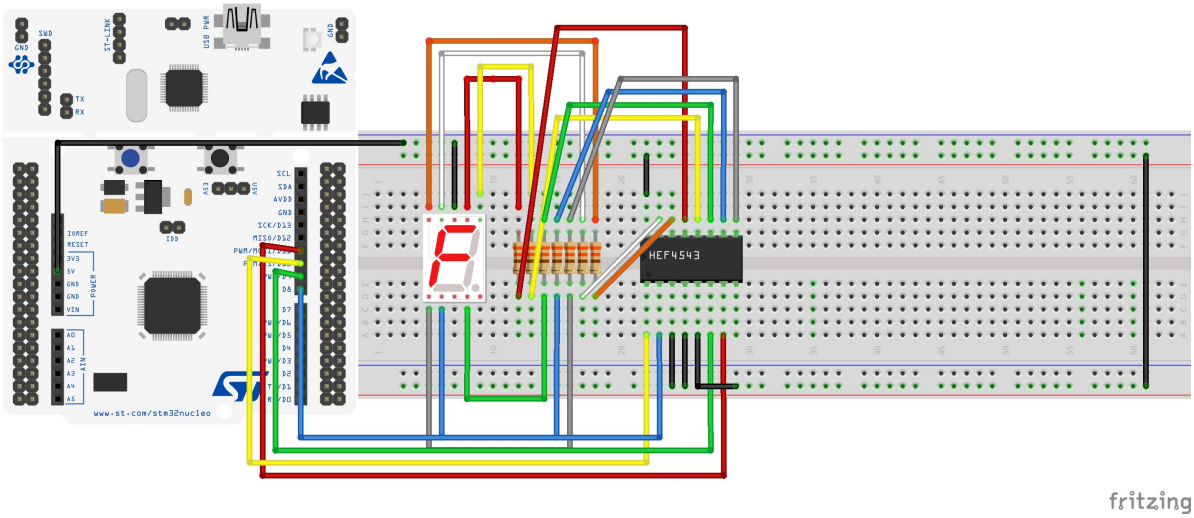
Now, use the decoder chip (**74LS47**). Connect it to the bread board.

Then, you need only 4 Digital out pins of MCU to display from 0 to 9.





ii. Connection Diagram



iii. Configuration

Digital In for Button (B1)	Digital Out for 7-Segment
Digital In	Digital Out
PC13	PA7, PB6, PC7, PA9
PULL-UP	Push-Pull, No Pull-up-Pull-down, Medium Speed

iv. Code

```
State_P2 FSM_D[10] = {
        {{S1, S0}, {0,0,0,0}},    // 0
        {{S2, S1}, {0,0,0,1,}},    // 1
        {{S3, S2}, {0,0,1,0,}},    // 2
        {{S4, S3}, {0,0,1,1,}},    // 3
        {{S5, S4}, {0,1,0,0,}},    // 4
        {{S6, S5}, {0,1,0,1,}},    // 5
        {{S7, S6}, {0,1,1,0,}},    // 6
        {{S8, S7}, {0,1,1,1,}},    // 7
        {{S9, S8}, {1,0,0,0,}},    // 8
        {{S0, S9}, {1,0,0,1,}}     // 9

};
```

The above FSM graph was implemented as a structure. And this was called from the header file below.

```
// GPIO.h
typedef struct{
    unsigned int next[2];
    unsigned int out[4];
} State_P2;

extern State_P2 FSM_D[10];
// {{next_state}, {D,C,B,A}}  FSM[state].next[Button_input]
```

```
void sevensegment_display_init(void){
    RCC_HSI_init();

    GPIO_init(GPIOA, pin_D, OUTPUT);
    GPIO_init(GPIOB, pin_C, OUTPUT);
    GPIO_init(GPIOC, pin_B, OUTPUT);
    GPIO_init(GPIOA, pin_A, OUTPUT);

    GPIO_otype(GPIOA, pin_D, EC_PUSH_PULL);
    GPIO_otype(GPIOB, pin_C, EC_PUSH_PULL);
    GPIO_otype(GPIOC, pin_B, EC_PUSH_PULL);
    GPIO_otype(GPIOA, pin_A, EC_PUSH_PULL);

    GPIO_pupd(GPIOA, pin_D, EC_NONE);
    GPIO_pupd(GPIOB, pin_C, EC_NONE);
    GPIO_pupd(GPIOC, pin_B, EC_NONE);
    GPIO_pupd(GPIOA, pin_A, EC_NONE);

    GPIO_ospeed(GPIOA, pin_D, EC_MEDIUM);
    GPIO_ospeed(GPIOB, pin_C, EC_MEDIUM);
    GPIO_ospeed(GPIOC, pin_B, EC_MEDIUM);
    GPIO_ospeed(GPIOA, pin_A, EC_MEDIUM);
}
```

It initialize the GPIO pins

```
void sevensegment_display(uint8_t  state){
    unsigned int D, C, B, A;

    D = FSM_D[state].out[0];
    C = FSM_D[state].out[1];
    B = FSM_D[state].out[2];
    A = FSM_D[state].out[3];

    GPIO_write(GPIOA, pin_c, D);
    GPIO_write(GPIOB, pin_d, C);
    GPIO_write(GPIOC, pin_e, B);
    GPIO_write(GPIOA, pin_f, A);

}
```

The function "sevensegment_display()" takes the output corresponding to the input state from the FSM_D structure and outputs it to the LED.

IV. Reference

https://ykkim.gitbook.io/ec/ec-course/lab/lab-gpio-digital-inout-7segment