LAB: GPIO Digital InOut 7-segment

Date: 2023-10-06

Author: EunChan Kim 21801017

Demo Video: Youtube link

Introduction

This project involves creating a simple program to control a 7-segment display. The goal of the program is to show decimal numbers (from 0 to 9) increasing each time a button is pressed.

Requirement

Hardware

MCU:

- NUCLEO-F411RE

Actuator/Sensor/Others:

- 7-segment display(5101ASR)
- Array resistor (330 ohm)
- decoder chip(74LS47)
- breadboard

Software

- Keil uVision, CMSIS, EC_HAL library

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 &= ~(0x0000FFFF);
	Set Pin5~9 mode = Output	GPIOA->MODER |= 0x00005555;
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);
Port A Pin Y	Set PinY PUPD=no pullup/down	GPIOA->OTYPER |= (0<<Y);

Problem 1: Connecting 7-Segment Display

Procedure

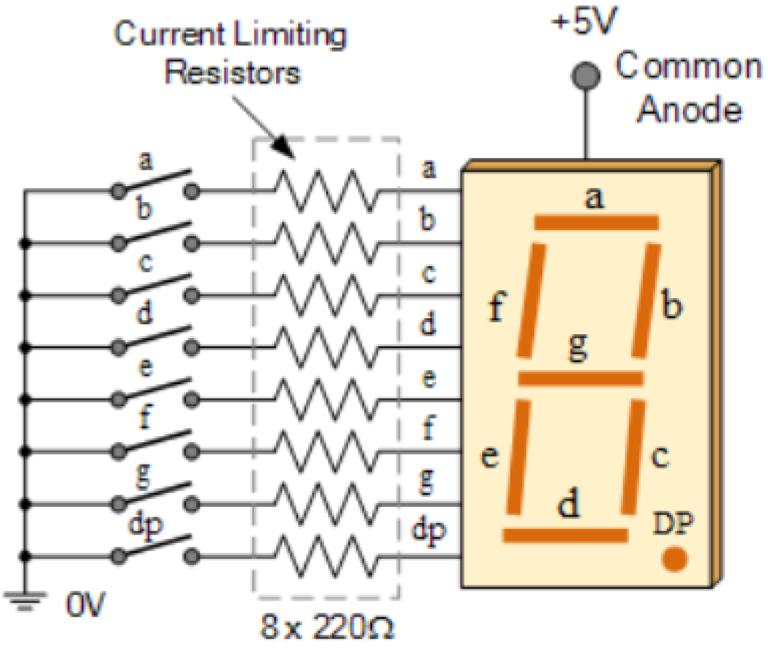


Figure1. 7-Segment Circuit

- 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

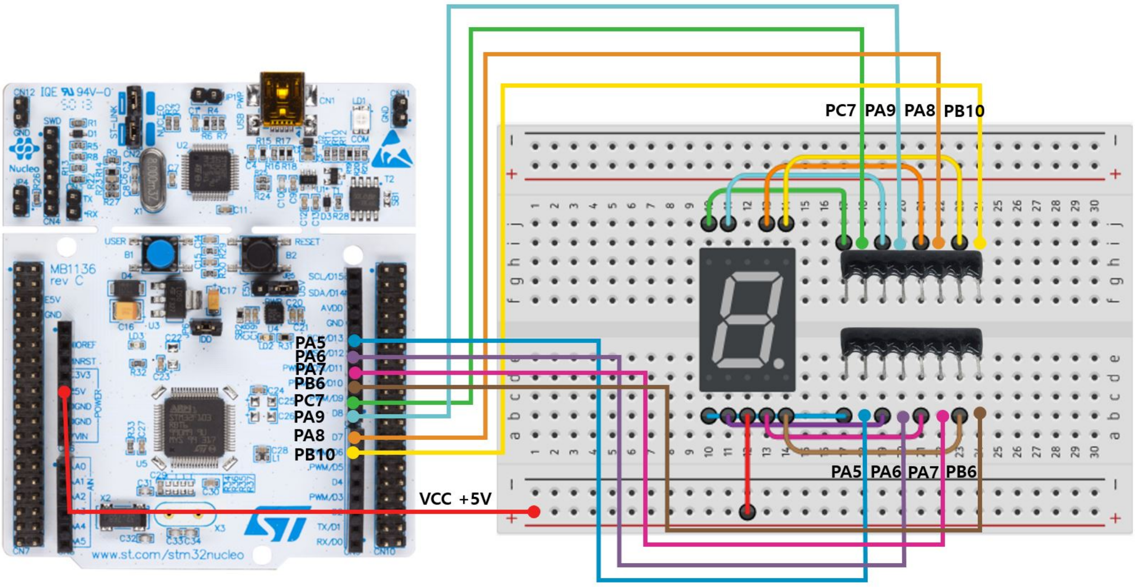


Figure2. 7-Segment circuit with registor array

Discussion

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

Digit	A	B	C	D	a	b	c	d	e	f	g	dp
0	0	0	0	0	1	1	1	1	1	1	0	0
1	0	0	0	1	0	1	1	0	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1	0
3	0	0	1	1	1	1	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0	0	1	1	0
5	0	1	0	1	1	0	1	1	0	1	1	0
6	0	1	1	0	1	0	1	1	1	1	1	0
7	0	1	1	1	1	1	1	0	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1	0
9	1	0	0	1	1	1	1	1	0	1	1	0

Table1. Truth table for the BCD 7-segment decoder

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

   The two types, common cathode and common anode, refer to how these segments are connected.

- Common Cathode

  In this type all the negative terminals (cathodes) of all the 8 LEDs are connected together. The common point is connected to Ground or 0V. To turn on any particular LED, its corresponding pin must be given high signal or VCC.

- Common Anode

  In this type all the positive terminals (Anodes) of all the 8 LEDs are connected together. The common point is connected to VCC. To turn on any particular LED, its corresponding pin must be grounded or given low signal or 0V.

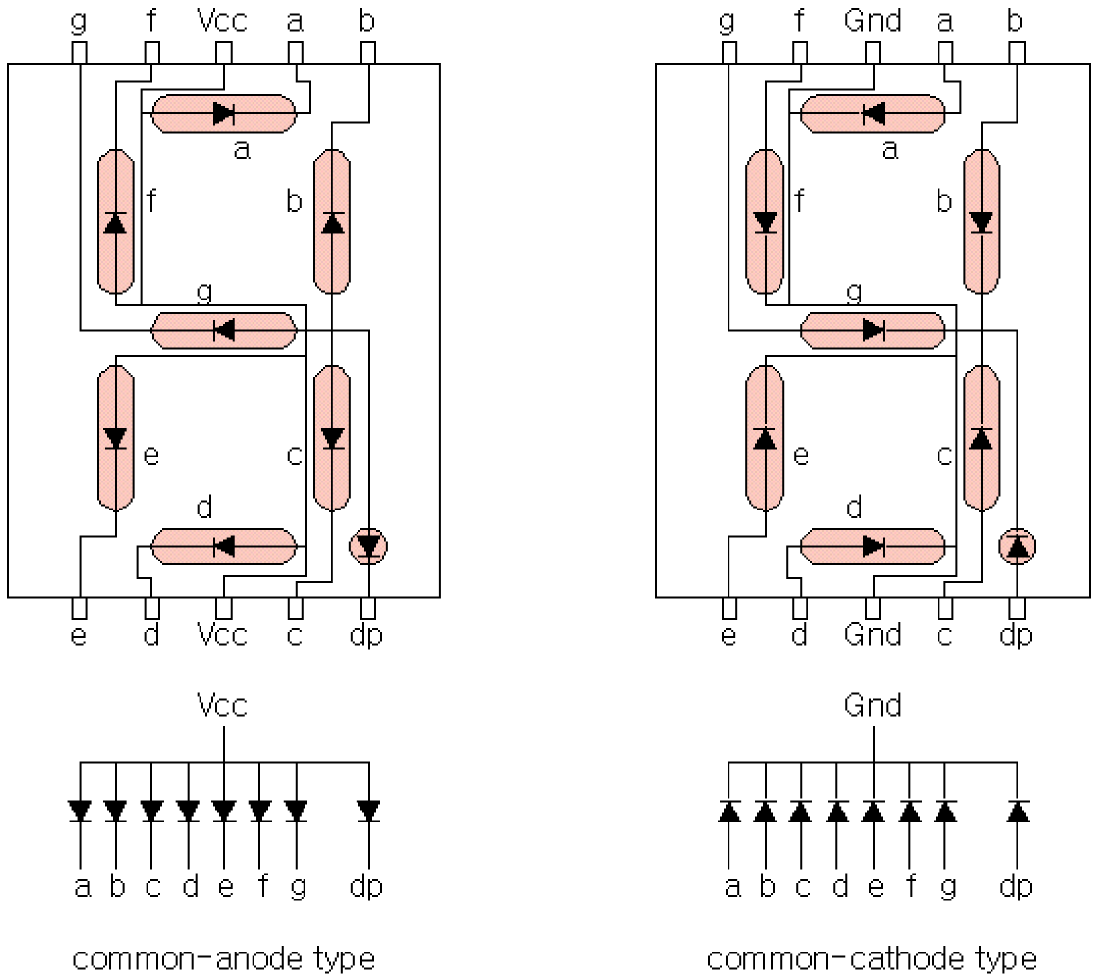


Figure3. common cathode and common anode circuit

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, the LED pin on the common anode does not illuminate when it receives a 'HIGH' signal from the MCU. The common anode type on the 7-segment display is illuminated when each LED segment receives a 'LOW' signal. This is because the common anode is connected to the VCC. Therefore, when the MCU gives a 'HIGH' signal to the LED pin, the LED is switched off.

Problem 2: Display 0~9 with button press

Procedure

- Create the functions `void sevensement_init` and `void sevensement_decoder(uint8_tnum)` to signal the LEDs in the seven segments from the eight pins when the button is pressed.
- Make sure that the numbers turn on sequentially each time you press the button.

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

Code

LAB_GPIO_7segment.c

Defines the GPIO pin number connected to each LED segment. The main function checks the button input within the infinite loop after initial setup, increasing the counter each time the button is pressed and outputting the value to the 7-segment display.

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

#define LED_PA5 5
#define LED_PA6 6
#define LED_PA7 7
#define LED_PB6 6
#define LED_PC7 7
#define LED_PA9 9
#define LED_PA8 8
#define LED_PB10 10
#define BUTTON_PIN 13

void setup(void);

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

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

        if(GPIO_read(GPIOC, BUTTON_PIN) == 0){
            cnt++;
            sevensegment_decoder(cnt % 10);
        }

        for(volatile int i = 0; i < 500000; i++){}
    }
}

// Initialiization
void setup(void){
    RCC_HSI_init();
    sevensegment_init();
    GPIO_init(GPIOC, BUTTON_PIN, INPUT);  // calls RCC_GPIOC_enable()
    GPIO_pupd(GPIOC, BUTTON_PIN, EC_PU); // pull-up resistor
}
```

ecGPIO.c

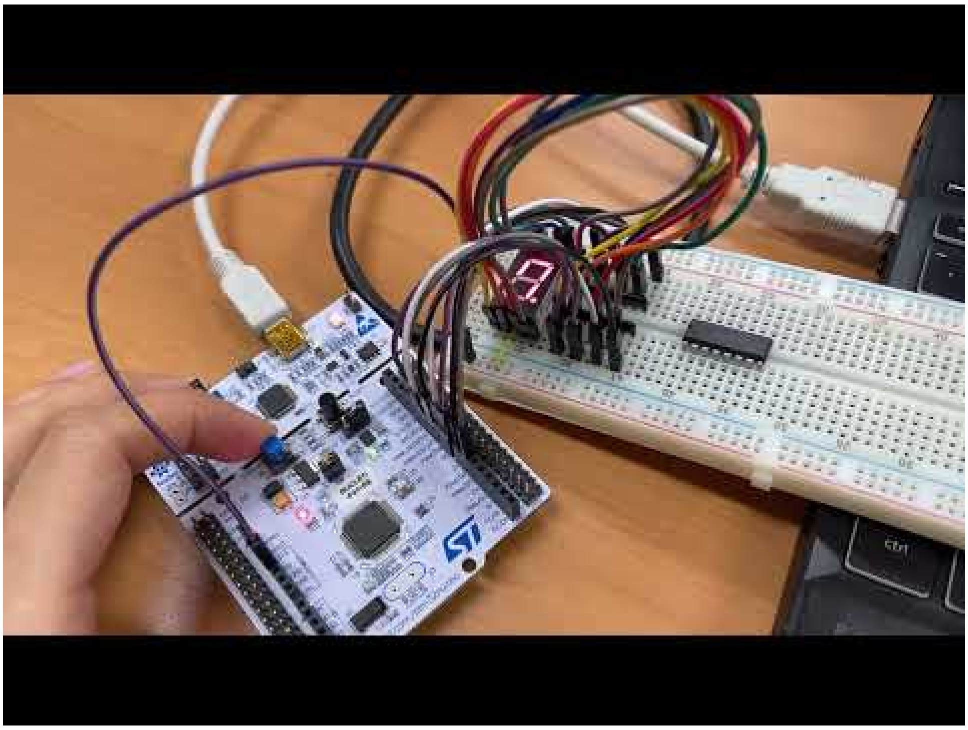
Defines a structure for storing information about GPIO pins. Each pin is associated with a specific port, which is represented by the GPIO_TypeDef* type. Initialize each LED pin connected to the 7-segment display according to the configuration.
Create an `unsigned int state[10][8]` array to define how each number from 0 to 9 should appear on the 7-segment display. It was created according to the Truth table. The `void segment_decoder(uint8_t num)` function outputs LED segment patterns between 0 and 9 given. The state of each LED pin is set using the 'state' array value corresponding to the number entered.

```
typedef struct {
    GPIO_TypeDef* port;
    uint16_t pin;
} GPIO;

void sevensegment_init(void) {
    GPIO pins[8] = {
        {GPIOA, 5},
        {GPIOA, 6},
        {GPIOA, 7},
        {GPIOB, 6},
        {GPIOC, 7},
        {GPIOA, 9},
        {GPIOA, 8},
        {GPIOB, 10}
    };

    // Initialize GPIO pins
    for (int i = 0; i < 8; ++i) {

        GPIO_init(pins[i].port, pins[i].pin, OUTPUT);
        GPIO_pupd(pins[i].port, pins[i].pin, EC_NONE);    // No Pull-up-Pull-
down
        GPIO_otype(pins[i].port, pins[i].pin, EC_PUSH_PULL); // Set LED as Push-
Pull
        GPIO_ospeed(pins[i].port, pins[i].pin, EC_MEDIUM);  // Set medium speed
        GPIO_write(pins[i].port, pins[i].pin, HIGH);       // Set state as HIGH
    }

    // initial state = 0
    for (int i = 0; i < 8; ++i) {
        if (pins[i].port == GPIOC && pins[i].pin == 7){
            GPIO_write(pins[i].port, pins[i].pin, HIGH);
        }
                else{
            GPIO_write(pins[i].port, pins[i].pin, LOW); // other led is LOW
        }
    }
}


unsigned int state[10][8]={

    {0,0,0,0,0,0,1,0}, //zero
    {1,0,0,1,1,1,1,0}, //one
    {0,0,1,0,0,1,0,0}, //two
    {0,0,0,0,1,1,0,0}, //three
    {1,0,0,1,1,0,0,0}, //four
    {0,1,0,0,1,0,0,0}, //five
    {0,1,0,0,0,0,0,0}, //six
    {0,0,0,1,1,0,1,0}, //seven
    {0,0,0,0,0,0,0,0}, //eight
    {0,0,0,0,1,0,0,0}, //nine

};

void sevensegment_decoder(uint8_t  num){

    GPIO_write(GPIOA, 8, state[num][0]);        // led a
    GPIO_write(GPIOB, 10,state[num][1]);        // led b
    GPIO_write(GPIOA, 7, state[num][2]);        // led c
    GPIO_write(GPIOA, 6, state[num][3]);        // led d
    GPIO_write(GPIOA, 5, state[num][4]);        // led e
    GPIO_write(GPIOA, 9, state[num][5]);        // led f
    GPIO_write(GPIOC, 7, state[num][6]);        // led g
    GPIO_write(GPIOB, 6, state[num][7]);        // led dp
}
```

Results

After creating and operating the function, the 7 segment is set to 0 at initialization and the number goes up one by one as the button is pressed. After 9, through 0, it grows back to 1.

Click Below ↓



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

Procedure

- Create a function to operate seven segments using decoder chip (74LS47).
- Verify that the button works as in Problem 2 when pressed.

Connection Diagram

Circuit diagram

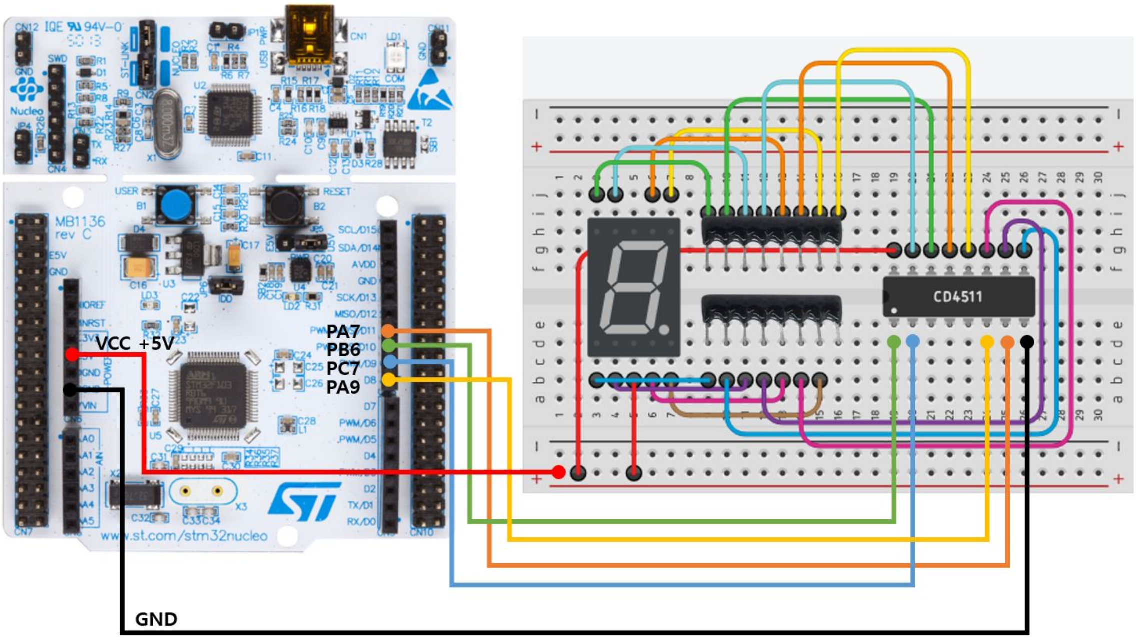


Figure4. 7-Segment circuit with Decoder

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

Code

ecGPIO.c

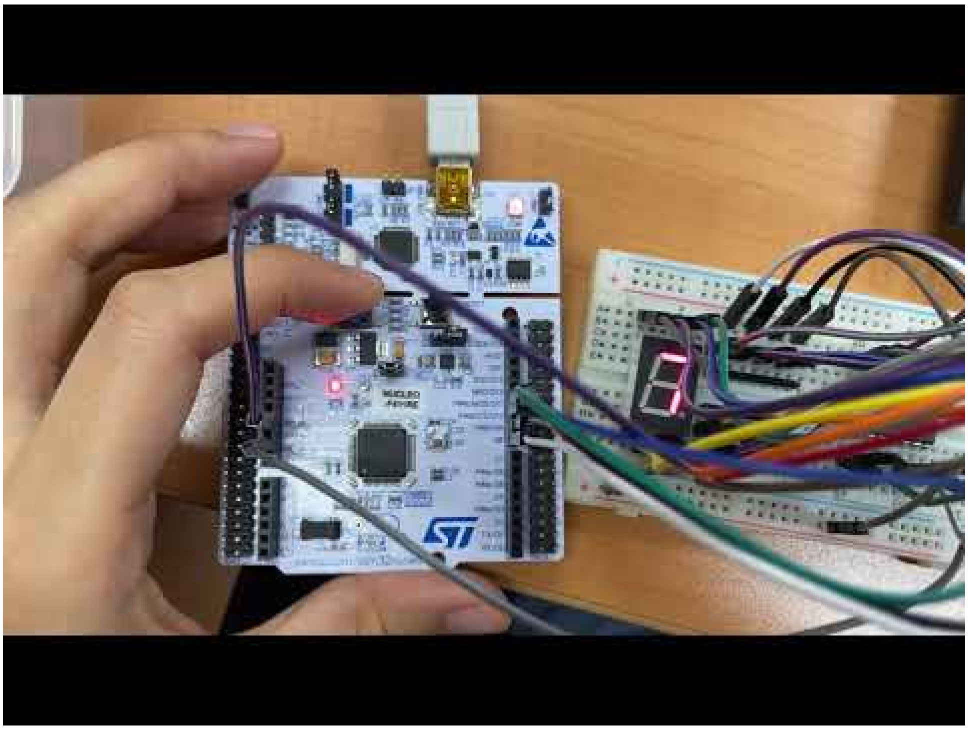
`void specification_display_init` resets the GPIO pin to output mode. The `unsigned int binary[10]` array represents each number from 0 to 9 in binary form. The function `void specification_display(uint8_tnum)` converts it into a binary form corresponding to a given number and passes the value to each GPIO pin. `(binary[num]>i) & 1` works in a BCD by using bit operation to extract the corresponding bit value.

```
void sevensegment_display_init(void){
        GPIO pins[4] = {
        {GPIOA, 7},
        {GPIOB, 6},
        {GPIOC, 7},
        {GPIOA, 9},
    };
        // Initialize GPIO pins
    for (int i = 0; i < 4; ++i) {
        GPIO_init(pins[i].port, pins[i].pin, OUTPUT);
        GPIO_pupd(pins[i].port, pins[i].pin, EC_NONE);     // No Pull-up-Pull-down
        GPIO_otype(pins[i].port, pins[i].pin, EC_PUSH_PULL); // Set LED as Push-Pull
        GPIO_ospeed(pins[i].port, pins[i].pin, EC_MEDIUM);   // Set medium speed
        GPIO_write(pins[i].port, pins[i].pin, LOW);          // Set to LOW
    }

}

unsigned int binary[10] = {
    0b0000, // zero
    0b0001, // one
    0b0010, // two
    0b0011, // three
    0b0100, // four
    0b0101, // five
    0b0110, // six
    0b0111, // seven
    0b1000, // eight
    0b1001, // nine
};

void sevensegment_display(uint8_t num){

    GPIO pins[4] = {
        {GPIOA, 7},
        {GPIOB, 6},
        {GPIOC, 7},
        {GPIOA, 9},
   };

    // Send the binary code to the GPIO pins
    for (int i = 0; i < 4; i++) {
         GPIO_write(pins[i].port, pins[i].pin, (binary[num] >> i) & 1);
    }

}
```

Result

It has been confirmed that pressing the button works the same as Problem2. Using four input pins and a decoder, The operation of seven segments in BCD method was completed using four input pins and a decoder.

Click Below ↓



Reference

https://jcdgods.tistory.com/456

Apendix

Entire code

```
/*----------------------
   LAB_GPIO_7segment.c
-----------------------*/

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

#define LED_PA5 5
#define LED_PA6 6
#define LED_PA7 7
#define LED_PB6 6
#define LED_PC7 7
#define LED_PA9 9
#define LED_PA8 8
#define LED_PB10 10
#define BUTTON_PIN 13

void setup(void);

int main(void) {
    // Initialiization --------------------------------------------------------
    setup();
    unsigned int cnt = 0;
    // Inifinite Loop ----------------------------------------------------------
    while(1){

        if(GPIO_read(GPIOC, BUTTON_PIN) == 0){
            cnt++;
            //sevensegment_decoder(cnt % 10);
            sevensegment_display(cnt % 10);
        }

        for(volatile int i = 0; i < 500000; i++){}

    }

}


// Initialiization
void setup(void){
    RCC_HSI_init();
    //sevensegment_init();
    sevensegment_display_init();
    GPIO_init(GPIOC, BUTTON_PIN, INPUT);  // calls RCC_GPIOC_enable()
    GPIO_pupd(GPIOC, BUTTON_PIN, EC_PU); // pull-up resistor
}


/*----------------------
        ecGPIO.c
-----------------------*/

typedef struct {
    GPIO_TypeDef* port;
    uint16_t pin;
} GPIO;

void sevensegment_init(void) {
    GPIO pins[8] = {
        {GPIOA, 5},
        {GPIOA, 6},
        {GPIOA, 7},
        {GPIOB, 6},
        {GPIOC, 7},
        {GPIOA, 9},
        {GPIOA, 8},
        {GPIOB, 10}
    };

    // Initialize GPIO pins
    for (int i = 0; i < 8; ++i) {

        GPIO_init(pins[i].port, pins[i].pin, OUTPUT);
        GPIO_pupd(pins[i].port, pins[i].pin, EC_NONE);     // No Pull-up-Pull-
down
        GPIO_otype(pins[i].port, pins[i].pin, EC_PUSH_PULL); // Set LED as Push-
Pull
        GPIO_ospeed(pins[i].port, pins[i].pin, EC_MEDIUM);   // Set medium speed
        GPIO_write(pins[i].port, pins[i].pin, HIGH);        // Set state as HIGH
    }
    // initial state = 0
    for (int i = 0; i < 8; ++i) {

        if (pins[i].port == GPIOC && pins[i].pin == 7){
            GPIO_write(pins[i].port, pins[i].pin, HIGH);
        }
                else{
            GPIO_write(pins[i].port, pins[i].pin, LOW); // other led is LOW
        }
    }
}


unsigned int state[10][8]={

    {0,0,0,0,0,0,1,0}, //zero
    {1,0,0,1,1,1,1,0}, //one
    {0,0,1,0,0,1,0,0}, //two
    {0,0,0,0,1,1,0,0}, //three
    {1,0,0,1,1,0,0,0}, //four
    {0,1,0,0,1,0,0,0}, //five
    {0,1,0,0,0,0,0,0}, //six
    {0,0,0,1,1,0,1,0}, //seven
    {0,0,0,0,0,0,0,0}, //eight
    {0,0,0,0,1,0,0,0}, //nine

};


void sevensegment_decoder(uint8_t  num){

    GPIO_write(GPIOA, 8, state[num][0]);        // led a
    GPIO_write(GPIOB, 10,state[num][1]);        // led b
    GPIO_write(GPIOA, 7, state[num][2]);        // led c
    GPIO_write(GPIOA, 6, state[num][3]);        // led d
    GPIO_write(GPIOA, 5, state[num][4]);        // led e
    GPIO_write(GPIOA, 9, state[num][5]);        // led f
    GPIO_write(GPIOC, 7, state[num][6]);        // led g
    GPIO_write(GPIOB, 6, state[num][7]);        // led dp
}

void sevensegment_display_init(void){
        GPIO pins[4] = {
        {GPIOA, 7},
        {GPIOB, 6},
        {GPIOC, 7},
        {GPIOA, 9},
    };
        // Initialize GPIO pins
    for (int i = 0; i < 4; ++i) {

        GPIO_init(pins[i].port, pins[i].pin, OUTPUT);
        GPIO_pupd(pins[i].port, pins[i].pin, EC_NONE);     // No Pull-up-Pull-
down
        GPIO_otype(pins[i].port, pins[i].pin, EC_PUSH_PULL); // Set LED as Push-
Pull
        GPIO_ospeed(pins[i].port, pins[i].pin, EC_MEDIUM);   // Set medium speed
        GPIO_write(pins[i].port, pins[i].pin, LOW);          // Set to LOW
    }

}

unsigned int binary[10] = {
    0b0000, // zero
    0b0001, // one
    0b0010, // two
    0b0011, // three
    0b0100, // four
    0b0101, // five
    0b0110, // six
    0b0111, // seven
    0b1000, // eight
    0b1001, // nine
};

void sevensegment_display(uint8_t num){

    GPIO pins[4] = {
        {GPIOA, 7},
        {GPIOB, 6},
        {GPIOC, 7},
        {GPIOA, 9},
   };

    // Send the binary code to the GPIO pins
    for (int i = 0; i < 4; i++) {
         GPIO_write(pins[i].port, pins[i].pin, (binary[num] >> i) & 1);
    }

}


/*----------------------
        ecGPIO.h
-----------------------*/
void sevensegment_init(void);

void sevensegment_decoder(uint8_t num);

void sevensegment_display_init(void);

void sevensegment_display(uint8_t  num);
```