

LAB: GPIO Digital InOut 7-segment(eval board)

LAB: GPIO Digital InOut 7-segment

Date: 2025-09-19

Author/Partner: Dongjun,Lee

Github:

<https://github.com/henny041520-commits/EC-DJLee-042/tree/main/lab/LAB%20GPIO%20Digital%20InOut%207-segment>

Demo Video: <https://youtube.com/shorts/G5XIK69HttQ?feature=share>

PDF version: <https://github.com/henny041520-commits/EC-DJLee-042/blob/main/report/%5BEC%202024-2%5D%20LAB%20GPIO%20Digital%20InOut%207-segment.pdf>

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.

You must submit

- LAB Report (*.pdf)
- Zip source files(lab***.c, ecRCC2.h, ecGPIO2.h etc...).
 - Only the source files. Do not submit project files

Requirement

Hardware

- MCU
 - NUCLEO-F411RE
- Actuator/Sensor/Others:
 - eval board

Software

- PlatformIO, CMSIS, EC_HAL library

Exercise

Fill in the table

Port/Pin	Description	Register setting
Port B Pin 5	Clear Pin5 mode	GPIOB->MODER &= ~(3<<(5*2))
Port B Pin 5	Set Pin5 mode = Output	GPIOB->MODER = 1<<(5*2)
Port B Pin 6	Clear Pin6 mode	GPIOB->MODER &= ~(3<<(6*2))
Port B Pin 6	Set Pin6 mode = Output	GPIOB->MODER = 1<<(6*2)
Port B Pin Y	Clear PinY mode	GPIOB->MODER &= ~(3<<(Y*2))
Port B Pin Y	Set PinY mode = Output	GPIOB->MODER = 1<<(Y*2)
Port B Pin 5~9	Clear Pin5~9 mode	for(int i = 5; i < 10; i++) {GPIOB->MODER &= ~(3<<(i*2))}
	Set Pin5~9 mode = Output	for(int i = 5; i < 10; i++) {GPIOB->MODER = 1<<(i*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 B Pin5	Set Pin5 otype=push-pull	GPIOB->OTYPER &= ~(1<<(5*2))
Port B PinY	Set PinY otype=push-pull	GPIOB-> OTYPER &= ~(Y<<(5*2))
Port B Pin5	Set Pin5 ospeed=Fast	GPIOB->OSPEEDR = 2<<(5*2)
Port B PinY	Set PinY ospeed=Fast	GPIOB-> OSPEEDR = 2<<(Y*2)
Port B Pin 5	Set Pin5 PUPD=no pullup/down	GPIOB->OTYPER &= ~(3<<(5*2))
Port B Pin Y	Set PinY PUPD=no pullup/down	GPIOB-> OTYPER &= ~(3<<(Y*2))

Problem 0: Preparation

Procedure

Complete the Tutorial: 7-segment Display.

```
{% embed url="https://ykkim.gitbook.io/ec/ec-course/tutorial/tutorial-7segment-display#option-3.-without-using-a-7-segment-decoder-on-jkit-evaluation-board" %}
```

You must check the 7-segment display can show all the number from 0 to 9.

- Give 'HIGH' signal to each 7-segment pin of 'a'~'g'
- Observe if that LED is turned ON or OFF
- Check another 7-segment display leds
 - Example: Connect VCC to all 'a'~'g' pins

Complete the required functions that displays numbers on 7-segment FND.

These functions must be moved to `ecGPIO2.h, ecGPIO2.c`

Update your library header

- **ecGPIO2.h, ecGPIO2.c**

```
PinName_t svgpinsFND[12]={PB_7, PB_6, PB_5, PB_4, PB_3, PB_2, PB_1, PB_0,
PC_3, PC_4, PA_11, PA_10};
PinName_t svgpinsnum[8]={PB_7, PB_6, PB_5, PB_4, PB_3, PB_2, PB_1, PB_0};
PinName_t svgpinsSelect[4]={ PC_3, PC_4, PA_11, PA_10};
/*
-----
==Pin mapping==
- svgpinsFND[0..7] : segment a,b,c,d,e,f,g,dp (8 pins)
- svgpinsFND[8..11] : digit select DIG0..DIG3 (4 pins)
- svgpinsnum[]       : segment pins ordered for bit-to-pin writes
- svgpinsSelect[]    : digit select pins
-----
*/
uint8_t svgvalue[10] = {
  0b00111111, //0
  0b000000110, //1
  0b01011011, //2
  0b01001111, //3
  0b01100110, //4
  0b01101101, //5
  0b01111101, //6
  0b001000111, //7
  0b01111111, //8
  0b01101111, //9
};

/*
-----
==Digit bit patterns for 0-9 (common-cathode)==
```

```

- Bit order (LSB→MSB): a b c d e f g dp
- Example: 0b00111111 means g=0, f=1, e=1, d=1, c=1, b=1, a=1 (dp=0)
- dp is OFF by default
----- */
void seven_seg_FND_display(uint8_t num, uint8_t select){
    // Activate the chosen digit (High)
    GPIO_write(svgpinsSelect[select], 1);
    for (int i=0; i<8;i++){
        //svgpinsnum[7~0] -> svgvalue[0~7]
        GPIO_write(svgpinsnum[7-i], (svgvalue[num]>>i )& 0x01);
    }
}
/*
=====seven_seg_FND_display=====
Show a single decimal digit (num: 0~9) on the selected position (select: 0~3).
Sequence:
1) Drive the selected digit line HIGH
2) Output the segment pattern of svgvalue[num] to svgpinsnum.
for array
arrangement : 01234567
for bit
arrangement : 76543210
this is why we reverse svgpinsnum's arrangement before we write svgvalue[num]
to it.
(if we set svgpinsnum[8]={PB_0,PB_1,PB_2, PB_3,PB_4,PB_5,PB_6,PB_7} like this,
reversing is not required)

*/
void seven_seg_FND_init(void){
    for(int i=0;i<12;i++)
    {
        GPIO_init(svgpinsFND[i], OUTPUT);
        GPIO_init(PA_4, INPUT);
        GPIO_otype(svgpinsFND[i], pushpull);
        GPIO_ospeed(svgpinsFND[i], mediumspeed);
        GPIO_pupd(svgpinsFND[i], nopupd);
    }
    GPIO_pupd(PA_4,pullup);
}
/*
=====seven_seg_FND_init=====
Initialize:
- 12 FND pins (8 segments + 4 digit selects) as Push-Pull, No Pull-up-Pull-

```

```

down, Medium Speed.
- Button on PA_4 as input with pull-up.
-----
*/
int readButtonRising(PinName_t pinName){
    static int prev = 1; // previous state
    int now = GPIO_read(pinName) ? 1 : 0; // 1 = released, 0 = pressed
    int rising = 0;
    // Detect 0→1 transition (release)
    if(prev==0 && now==1){
        //debounce(60~100 ms)
        for(volatile int i=0;i<160000;i++);
        rising = 1;
    }
    prev = now; // update state
    return rising; // Detect 0→1 transition -> 1 , not -> 0
}
/*
==readButtonRising(ben made for other labs, not necessary for 7-segment
Display)
0→1 transition detected on the given pin return 1, else 0.
Operation:
1) Read current level (now).
2) If prev==0 and now==1, a rising edge occurred.
   1.start : now = 1, prev = 1 -> if(x) -> now = 1, prev = 1
   2.pressed : now = 0, prev = 1 -> if(x) -> now = 0, prev = 0
   3.released : now = 1, prev = 0 -> if(o) -> now = 1, prev = 1
3) if(0) -> Apply a short delay as a debounce, rising = 1
               else rising = 0
4) return rising.
-----
*/

```

Problem 1: Display a Number with Button Press

Procedure

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**”
- Update `platformio.ini` for VS.Code : [Read here for detail](#)

Create a code that increases the displayed number from 0 to 9 with each button press.

- After the number '9', it should start from '0' again.
-

Configuration

Configure the MCU GPIO

Digital In for Button (B1)	Digital Out for 7-Segment
Digital In	Digital Out
PA4	PB7,PB6,PB5,PB4,PB3,PB2,PB1,PB0 ('a'~'h', respectively) PC3,PC4,PA11,PA10 ('FND_0'~FND_3, respectively)
PULL-UP	Push-Pull, No Pull-up-Pull-down, Medium Speed

Algorithm

Overview

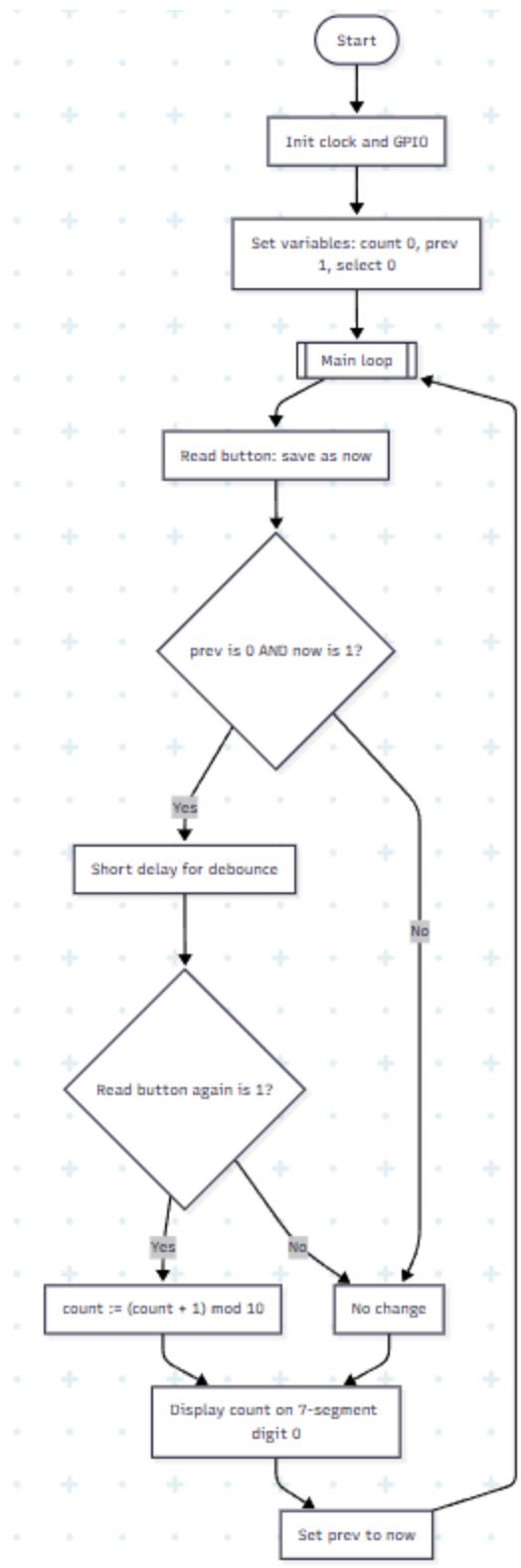
Mealy FSM Table — Button Counter with 7-Segment Display (1 Digit)

Present State	Input X (Button)	Condition	Next State	Output (Immediate)
S0 (IDLE_UP)	X=0 (pressed)	button goes low	S1	—
S0 (IDLE_UP)	X=1 (released)	still released	S0	—
S1 (PRESSED)	X=0 (pressed)	still pressed	S1	—
S1 (PRESSED)	X=1 (released)	rising edge detected	S0	count ← (count+1) mod 10; show on 7-seg digit 0

Output Logic

- `selectFND = 0` (only one digit enabled)
- `SEG = svgvalue[count]` (7-segment pattern from table `svgvalue []``)

Flow chart



Code

Sample Code.

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

#define BUTTON_PIN PA_4

void setup(void){
    // Initialize System Clock
    RCC_HSI_init();
    GPIO_init(BUTTON_PIN, INPUT); // calls RCC_GPIOC_enable()
    // and Others
    // [YOUR CODE GOES HERE]
    seven_seg_FND_init();
}

int main(void) {
    setup();
    uint8 numDisplay=8;
    uint8 selectFND=0;

    while (1) {
        // [YOUR CODE GOES HERE]
        seven_seg_FND_display(numDisplay,selectFND);
        // [YOUR CODE GOES HERE]
        // [YOUR CODE GOES HERE]
    }
}
```

Your code goes here: [ecGPIO2.c](#)

| Explain your source code with necessary comments.

```
*/
void seven_seg_FND_display(uint8_t num, uint8_t select){
    // Activate the chosen digit (High)
    GPIO_write(svgpinsSelect[select], 1);
    for (int i=0; i<8;i++){
        //svgpinsnum[7~0] -> svgvalue[0~7]
        GPIO_write(svgpinsnum[7-i], (svgvalue[num]>>i )& 0x01);
    }
}
```

```
}

/*
-----  

==seven_seg_FND_display==  

Show a single decimal digit (num: 0-9) on the selected position (select: 0-3).  

Sequence:  

1) Drive the selected digit line HIGH  

2) Output the segment pattern of svgvalue[num] to svgpinsnum.  

for array  

arrangement : 01234567  

for bit  

arrangement : 76543210  

this is why we reverse svgpinsnum's arrangement before we write svgvalue[num]  

to it.  

(if we set svgpinsnum[8]={PB_0,PB_1,PB_2, PB_3,PB_4,PB_5,PB_6,PB_7} like this,  

reversing is not required)
```

```
/*
int readButtonRising(PinName_t pinName){
    static int prev = 1; // previous state
    int now = GPIO_read(pinName) ? 1 : 0; // 1 = released, 0 = pressed
    int rising = 0;
    // Detect 0→1 transition (release)
    if(prev==0 && now==1){
        //debounce(busy-wait)
        for(volatile int i=0;i<160000;i++);
        rising = 1;
    }
    prev = now; // update state
    return rising; // Detect 0→1 transition -> 1 , not -> 0
}
/*
```

```
==readButtonRising(been made for other labs, not necessary for 7-segment  

Display)
```

0→1 transition detected on the given pin return 1, else 0.

Operation:

- 1) Read current level (now).
- 2) If prev==0 and now==1, a rising edge occurred.
 - 1.start : now = 1, prev = 1 -> if(x) -> now = 1, prev = 1
 - 2.pressed : now = 0, prev = 1 -> if(x) -> now = 0, prev = 0
 - 3.released : now = 1, prev = 0 -> if(o) -> now = 1, prev = 1
- 3) if(0) -> Apply a short delay as a debounce, rising = 1
else rising = 0
- 4) return rising.

```
*/
```

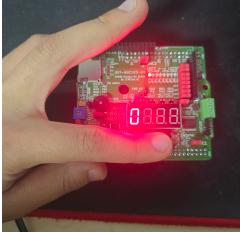
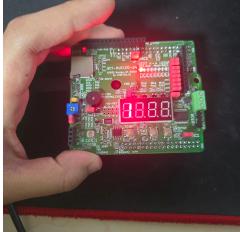
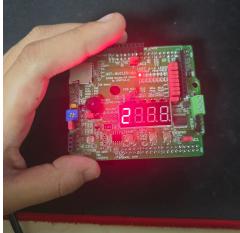
Your code goes here: [LAB GPIO Digital InOut 7-segment.c](#)

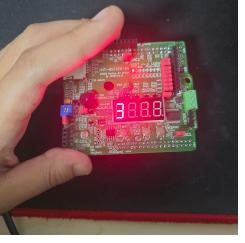
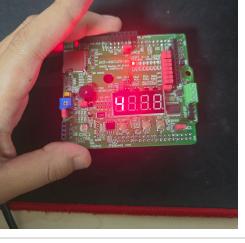
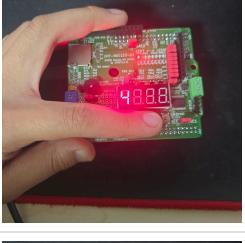
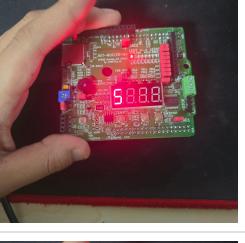
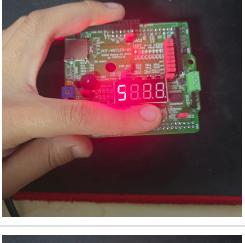
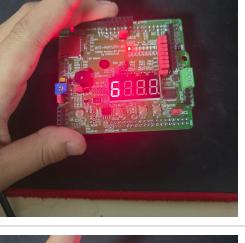
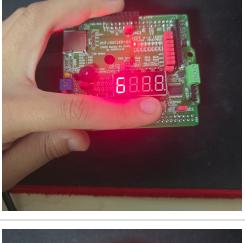
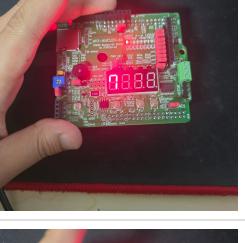
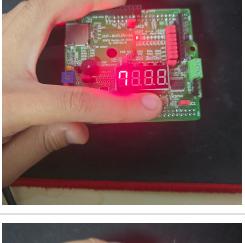
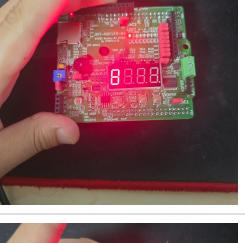
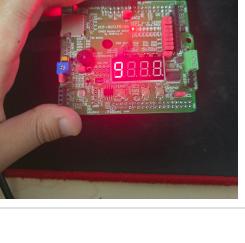
Explain your source code with the necessary comments.

```
int main(void) {
    setup();

    uint8_t selectFND=0; // Which digit position to light (0..3)
    uint8_t count = 0; // Value to display (0..9)
    while (1) {
        // Edge-triggered count: button release -> increment once per .
        // readButtonRising(PA_4) should return 1 only on a 0->1 transition
        if(readButtonRising(PA_4)){
            count++; //one step advance per release event
        }
        if(count==10)
        {
            count=0; // reset display number to 0
        }
        seven_seg_FND_display(count,selectFND);
    }
}
```

Results

number	press	release
0		
1		

number	press	release
2		
3		
4		
5		
6		
7		
8		

number	press	release
9		

<https://youtube.com/shorts/G5XIK69HtQ?feature=share>

Discussion

1. Analyze the result and explain any other necessary discussion.
 2. Draw the truth table for the BCD 7-segment decoder with the 4-bit input.

Answer :

Button release (0→1) is detected with `readButtonRising(PA_4)` → counter increments once per release

One digit is shown with `seven_seg_FND_display(count, selectFND)`

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

Answer :

Common Cathode (CC)(active-HIGH segments): all LED segment cathodes are tied together to GND. To turn a segment ON -> HIGH

Common Anode (CA)(active-LOW segments): all LED segment anodes are tied together to VCC. To turn a segment ON -> LOW
(VCC → common anode → LED → MCU pin to GND).

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

Answer :

In common-anode, the common pin is at VCC. A segment lights when its cathode side is pulled LOW so current can flow.

Turn ON a segment -> MCU LOW

Reference

BCD to 7 Segment Decoder : <https://www.geeksforgeeks.org/digital-logic/bcd-to-7-segment-decoder/>
