ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH

VI XỬ LÝ - VI ĐIỀU KHIỂN

Lab 3

BUTTON - SWITCH

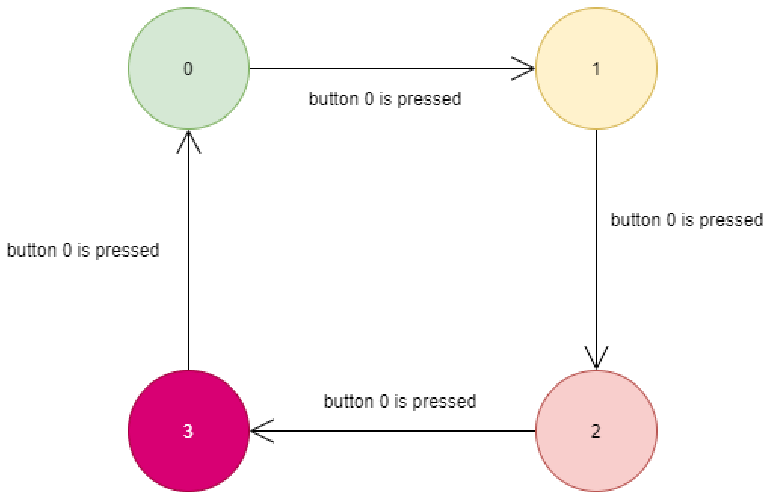
GVHD: **Huỳnh Phúc Nghị**
Thành viên : Đỗ Xuân Thơ 1713365

TP. HỒ CHÍ MINH, THÁNG 09/2021

Mục lục

1 Execises 1

Your task in this exercise is to sketch an FSM that describes your idea of how to solve the problem.

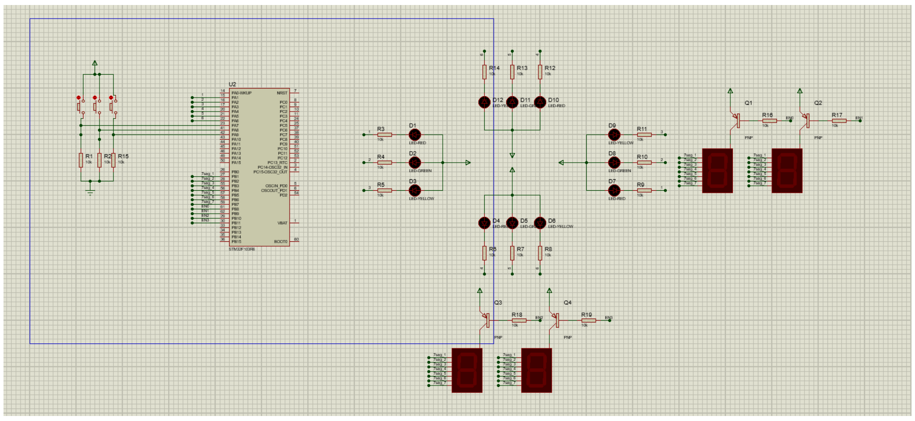


Hình 1: State transition for 4 mode

2 Execises 2

Your task in this exercise is to draw a Proteus schematic for the problem above.

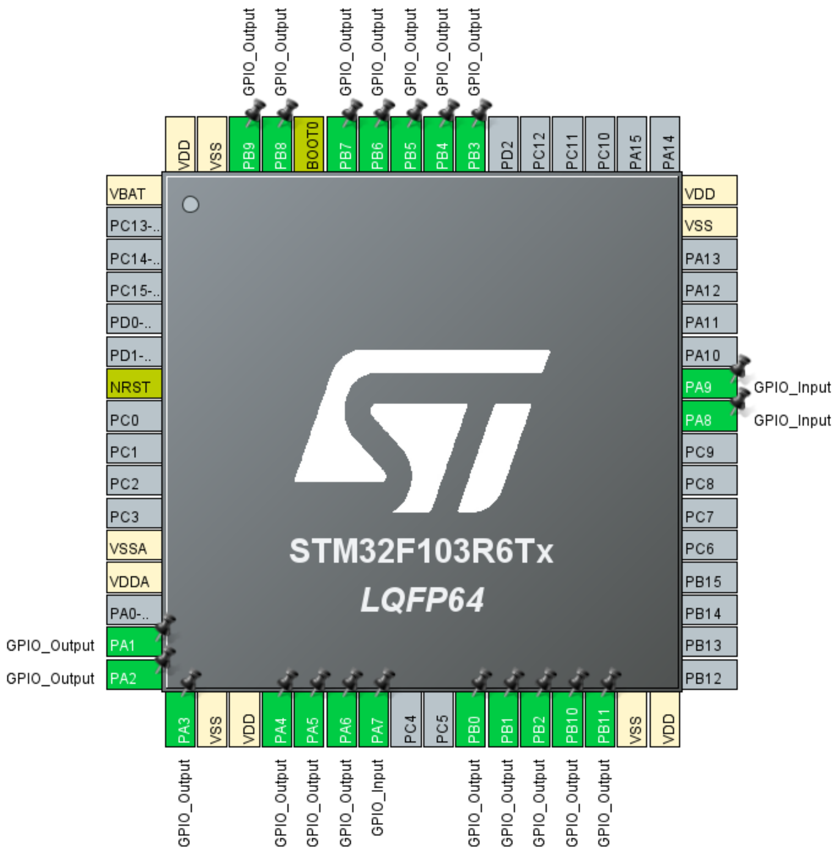
2.1 Schematic



Hình 2: Proteus Schematic

3 Execises 3

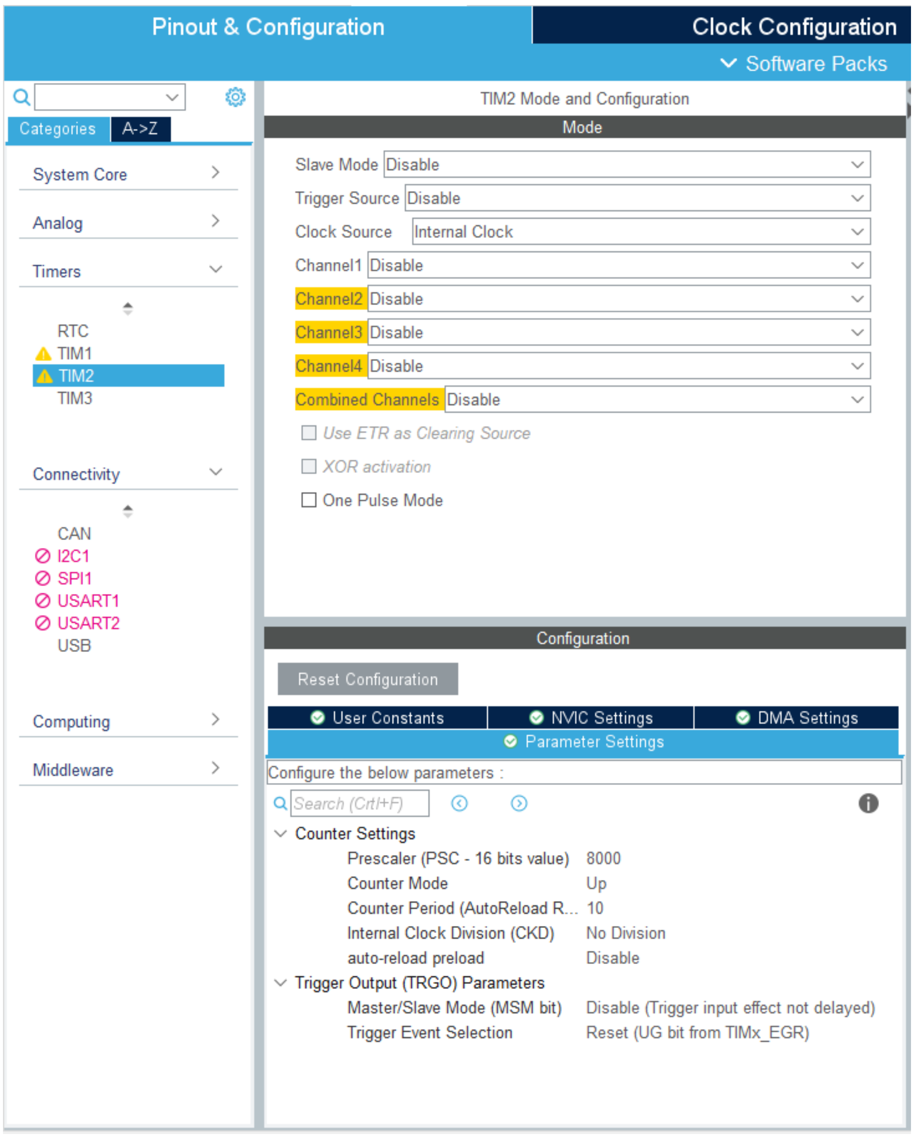
Your task in this exercise is to create a project that has pin corresponding to the Proteus schematic that you draw in previous section. You need to set up your timer interrupt is about 10ms.



Hình 3: Execise 3 - Proteus project file

4 Execises 4

Your task in this exercise is to modify the timer settings so that when we want to change the time duration of the timer interrupt, we change it the least and it will not affect the overall system. For example, the current system we have implemented is that it can blink an LED in 2 Hz, with the timer interrupt duration is 10ms. However, when we want to change the timer interrupt duration to 1ms or 100ms, it will not affect the 2Hz blinking LED



TIM2 Mode and Configuration

Mode

Slave Mode Disable
Trigger Source Disable
Clock Source Internal Clock
Channel1 Disable
Channel2 Disable
Channel3 Disable
Channel4 Disable
Combined Channels Disable
☐ Use ETR as Clearing Source
☐ XOR activation
☐ One Pulse Mode

Configuration

Configure the below parameters :

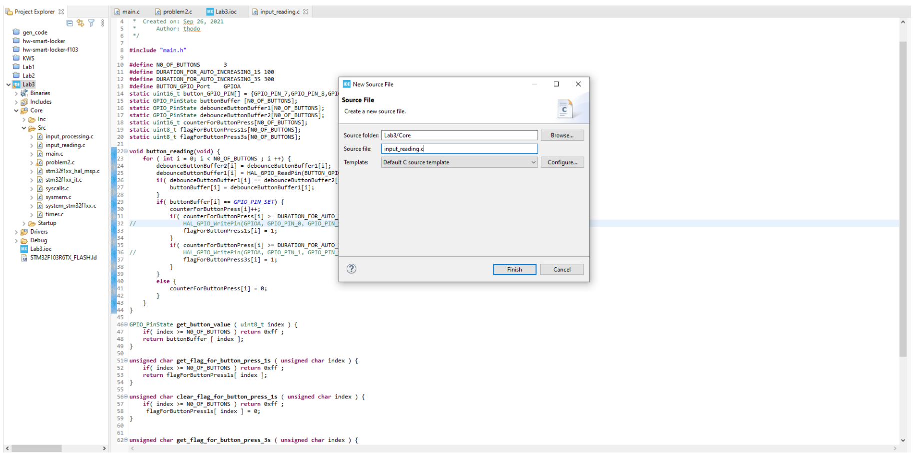
Counter Settings	
Prescaler (PSC - 16 bits value)	8000
Counter Mode	Up
Counter Period (AutoReload R...	10
Internal Clock Division (CKD)	No Division
auto-reload preload	Disable
Trigger Output (TRGO) Parameters	
Master/Slave Mode (MSM bit)	Disable (Trigger input effect not delayed)
Trigger Event Selection	Reset (UG bit from TIMx_EGR)

Hình 4: Timer Configuration

5 Execises 5

Following the example of button reading and debouncing in the previous section, your tasks in this exercise are:

- To add new files for input reading and output display
- To add code for button debouncing
- To add code for increasing mode when the first button is pressed



```
 *  Created on: Sep 26, 2021
 *      Author: thodo
 */

#include "main.h"

#define N0_OF_BUTTONS       3
#define DURATION_FOR_AUTO_INCREASING_1S 100
#define DURATION_FOR_AUTO_INCREASING_3S 300
#define BUTTON_GPIO_Port    GPIOA
static uint16_t button_GPIO_PIN[] = {GPIO_PIN_7,GPIO_PIN_8,GPI
static GPIO_PinState buttonBuffer [N0_OF_BUTTONS];
static GPIO_PinState debounceButtonBuffer1[N0_OF_BUTTONS];
static GPIO_PinState debounceButtonBuffer2[N0_OF_BUTTONS];
static uint16_t counterForButtonPress[N0_OF_BUTTONS];
static uint8_t flagForButtonPress1s[N0_OF_BUTTONS];
static uint8_t flagForButtonPress3s[N0_OF_BUTTONS];

void button_reading(void) {
    for ( int i = 0; i < N0_OF_BUTTONS ; i ++) {
        debounceButtonBuffer2[i] = debounceButtonBuffer1[i];
        debounceButtonBuffer1[i] = HAL_GPIO_ReadPin(BUTTON_GPI
        if( debounceButtonBuffer1[i] == debounceButtonBuffer2[
            buttonBuffer[i] = debounceButtonBuffer1[i];
        }
        if( buttonBuffer[i] == GPIO_PIN_SET) {
            counterForButtonPress[i]++;
            if( counterForButtonPress[i] >= DURATION_FOR_AUTO_
//              HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_
                flagForButtonPress1s[i] = 1;
            }
            if( counterForButtonPress[i] >= DURATION_FOR_AUTO_
//              HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_
                flagForButtonPress3s[i] = 1;
            }
        }
        else {
            counterForButtonPress[i] = 0;
        }
    }
}

GPIO_PinState get_button_value ( uint8_t index ) {
    if( index >= N0_OF_BUTTONS ) return 0xff ;
    return buttonBuffer [ index ];
}

unsigned char get_flag_for_button_press_1s ( unsigned char index ) {
    if( index >= N0_OF_BUTTONS ) return 0xff ;
    return flagForButtonPress1s[ index ];
}

unsigned char clear_flag_for_button_press_1s ( unsigned char index ) {
    if( index >= N0_OF_BUTTONS ) return 0xff ;
     flagForButtonPress1s[ index ] = 0;
}


unsigned char get_flag_for_button_press_3s ( unsigned char index ) {
```

Source folder: Lab3/Core
Source file: input_reading.c
Template: Default C source template

Hình 5: Create input_reading.c file

5.1 Source code

- Button Debouncing

```
/*
 * input_processing.c
 *
 *  Created on: Sep 26, 2021
 *      Author: thodo
 */

#include "main.h"

#define NO_OF_BUTTONS    3
#define DURATION_FOR_AUTO_INCREASING_1S 100
#define DURATION_FOR_AUTO_INCREASING_3S 300
#define BUTTON_GPIO_Port GPIOA
static uint16_t button_GPIO_PIN[] = {GPIO_PIN_7,GPIO_PIN_8,GPIO_PIN_9};
static GPIO_PinState buttonBuffer [NO_OF_BUTTONS];
static GPIO_PinState debounceButtonBuffer1[NO_OF_BUTTONS];
static GPIO_PinState debounceButtonBuffer2[NO_OF_BUTTONS];
static uint16_t counterForButtonPress[NO_OF_BUTTONS];
```

```c
static uint8_t flagForButtonPress1s[NO_OF_BUTTONS];
static uint8_t flagForButtonPress3s[NO_OF_BUTTONS];

void button_reading(void) {
   for ( int i = 0; i < NO_OF_BUTTONS ; i ++) {
      debounceButtonBuffer2[i] = debounceButtonBuffer1[i];
      debounceButtonBuffer1[i] = HAL_GPIO_ReadPin(BUTTON_GPIO_Port,button_GPIO_PIN[i]);
      if( debounceButtonBuffer1[i] == debounceButtonBuffer2[i]){
         buttonBuffer[i] = debounceButtonBuffer1[i];
      }
      if( buttonBuffer[i] == GPIO_PIN_SET) {
         counterForButtonPress[i]++;
         if( counterForButtonPress[i] >= DURATION_FOR_AUTO_INCREASING_1S ) {
//          HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_SET);
            flagForButtonPress1s[i] = 1;
         }
         if( counterForButtonPress[i] >= DURATION_FOR_AUTO_INCREASING_3S ) {
//          HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_SET);
            flagForButtonPress3s[i] = 1;
         }
      }
      else {
         counterForButtonPress[i] = 0;
      }
   }
}

GPIO_PinState get_button_value ( uint8_t index ) {
   if( index >= NO_OF_BUTTONS ) return 0xff ;
   return buttonBuffer [ index ];
}

unsigned char get_flag_for_button_press_1s ( unsigned char index ) {
   if( index >= NO_OF_BUTTONS ) return 0xff ;
   return flagForButtonPress1s[ index ];
}

unsigned char clear_flag_for_button_press_1s ( unsigned char index ) {
   if( index >= NO_OF_BUTTONS ) return 0xff ;
    flagForButtonPress1s[ index ] = 0;
}


unsigned char get_flag_for_button_press_3s ( unsigned char index ) {
   if( index >= NO_OF_BUTTONS ) return 0xff ;
   return flagForButtonPress3s[ index ];
}

unsigned char clear_flag_for_button_press_3s ( unsigned char index ) {
   if( index >= NO_OF_BUTTONS ) return 0xff;
    flagForButtonPress3s[ index ] = 0;
}
```

```
unsigned char clear_counter_for_button_press ( unsigned char index ) {
   if( index >= NO_OF_BUTTONS ) return 0xff ;
    counterForButtonPress[ index ] = 0;
}
```

- Increasing mode

```
void fsm_processing(void){
   switch (mode) {
      case 0:
         if(get_button_value(0)){
            mode = 1;
            segment_index = 0;
            reset_timer1();
         }
         else{
            if(get_flag_timer0()){
               counter_down_left_right -- ;
               counter_down_top_down -- ;
               if(counter_down_left_right <= 0){
                  switch (led_left_right) {
                     case 0:
                        counter_down_left_right = MAX_GREEN_COUNTER;
                        break;
                     case 1:
                        counter_down_left_right = MAX_YELLOW_COUNTER;
                        break;
                     case 2:
                        counter_down_left_right = MAX_RED_COUNTER;
                        break;
                     default:
                        break;
                  }
                  led_left_right = (led_left_right+1)%3;
               }
               if(counter_down_top_down <= 0){
                  switch (led_top_down) {
                     case 0:
                        counter_down_top_down = MAX_GREEN_COUNTER;
                        break;
                     case 1:
                        counter_down_top_down = MAX_YELLOW_COUNTER;
                        break;
                     case 2:
                        counter_down_top_down = MAX_RED_COUNTER;
                        break;
                     default:
                        break;
                  }
                  led_top_down = (led_top_down+1)%3;
               }
               set_Timer0(1000);
```

```
        }
        display_7seg();
        display_LED_Traffic();
    }
    break;
case 1:
    if(get_button_value(0)){
        mode = 2;
        segment_index = 0;
        reset_timer1();
        reset_timer2();
    }
    else{
        if(get_button_value(1)){
            temp_counter = (temp_counter+1)%100;
        }
        if(get_button_value(2)){
            MAX_RED_COUNTER = temp_counter;
        }
        //blink 0 ~ red
        blink_index(0);
        display_7seg_modify();
    }
    break;
case 2:
    if(get_button_value(0)){
        mode = 3;
        segment_index = 0;
        reset_timer1();
        reset_timer2();
    }
    else{
        if(get_button_value(1)){
            temp_counter = (temp_counter+1)%100;
        }
        if(get_button_value(2)){
            MAX_GREEN_COUNTER = temp_counter;
        }
        //blink 0 ~ red
        blink_index(1);
        display_7seg_modify();
    }
    break;
case 3:
    if(get_button_value(0)){
        mode = 0;
        segment_index = 0;
        reset_timer1();
        reset_timer2();
    }
    else{
        if(get_button_value(1)){
            temp_counter = (temp_counter+1)%100;
            }
            if(get_button_value(2)){
               MAX_YELLOW_COUNTER = temp_counter;
            }
            //blink 0 ~ red
            blink_index(2);
            display_7seg_modify();
         }
         break;
      case 4:

         break;
      default:
         break;
   }
}
```

6 Execises 6

Your tasks in this exercise are:

- To add code for display mode on seven-segment LEDs, and
- To add code for blinking LEDs depending on the mode that is selected.

6.1 Source Code

- display_7SEG()

```
uint8_t MAX_RED_COUNTER  = 5; //seconds
uint8_t MAX_GREEN_COUNTER = 3 ;
uint8_t MAX_YELLOW_COUNTER  = 2;




uint8_t counter_down_left_right = 5;
uint8_t counter_down_top_down = 3;
uint8_t temp_counter = 0;

/*
 * 0: red
 * 1: green
 * 2: yellow
 */
uint8_t led_left_right = 0;
uint8_t led_top_down = 1;


uint8_t segment_index = 0;

uint8_t keep_bt2_flag = 0;


uint8_t stop_clk_minutes = 0;
uint8_t stop_clk_seconds = 0;
uint8_t stop_clk_1_100second = 0;

uint16_t gpio_pinmask_led[] =
    {GPIO_PIN_1,GPIO_PIN_2,GPIO_PIN_3,GPIO_PIN_4,GPIO_PIN_5,GPIO_PIN_6};
uint32_t gpio_port_led_enable = GPIOA;

uint16_t gpio_pinmask_enable_7segment[] = {GPIO_PIN_8,GPIO_PIN_9,GPIO_PIN_10,GPIO_PIN_11};
uint32_t gpio_port_enable = GPIOB;

uint16_t gpio_pinmask_7seg[] =
    {GPIO_PIN_1,GPIO_PIN_2,GPIO_PIN_3,GPIO_PIN_4,GPIO_PIN_5,GPIO_PIN_6,GPIO_PIN_7};
uint32_t gpio_port_7seg = GPIOB;
/*
 * mode 0 : run normal clk
 * mode 1 : increase hours
 * mode 2 : increase minutes
 * mode 3 : increase seconds
 * mode 4 : run stop clk
 */
uint8_t mode = 0;


void Clear_7Seg(){
   HAL_GPIO_WritePin(gpio_port_7seg,
       gpio_pinmask_7seg[0]|gpio_pinmask_7seg[1]|gpio_pinmask_7seg[2]|gpio_pinmask_7seg[3]|gpio_pinmask_7se
       GPIO_PIN_SET);
}
void display7SEG(uint8_t number){
   Clear_7Seg();
   switch (number) {
      case 0:
         /*
          * b'abcdefg = b'1111110
          */
         HAL_GPIO_WritePin(gpio_port_7seg,
             gpio_pinmask_7seg[0]|gpio_pinmask_7seg[1]|gpio_pinmask_7seg[2]|gpio_pinmask_7seg[3]|gpio_pinma
             GPIO_PIN_RESET);
         break;
      case 1:
         /*
          * b'abcdefg = b'0110000
          */
         HAL_GPIO_WritePin(gpio_port_7seg, gpio_pinmask_7seg[1]|gpio_pinmask_7seg[2],
             GPIO_PIN_RESET);
         break;
      case 2:
         /*
          * b'abcdefg = b'1101101
          */
         HAL_GPIO_WritePin(gpio_port_7seg,
             gpio_pinmask_7seg[0]|gpio_pinmask_7seg[1]|gpio_pinmask_7seg[3]|gpio_pinmask_7seg[4]|gpio_pinma
             GPIO_PIN_RESET);
         break;
      case 3:
         /*
          * b'abcdefg = b'1111001
          */
         HAL_GPIO_WritePin(gpio_port_7seg,
             gpio_pinmask_7seg[0]|gpio_pinmask_7seg[1]|gpio_pinmask_7seg[2]|gpio_pinmask_7seg[3]|gpio_pinma
             GPIO_PIN_RESET);
         break;
      case 4:
         /*
          * b'abcdefg = b'0110011
          */
         HAL_GPIO_WritePin(gpio_port_7seg,
             gpio_pinmask_7seg[1]|gpio_pinmask_7seg[2]|gpio_pinmask_7seg[5]|gpio_pinmask_7seg[6],
             GPIO_PIN_RESET);
```

```
            break;
        case 5:
            /*
             * b'abcdefg = b'1011011
             */
            HAL_GPIO_WritePin(gpio_port_7seg,
                gpio_pinmask_7seg[0]|gpio_pinmask_7seg[2]|gpio_pinmask_7seg[3]|gpio_pinmask_7seg[5]|gpio_pinma
                GPIO_PIN_RESET);
            break;
        case 6:
            /*
             * b'abcdefg = b'1011111
             */
            HAL_GPIO_WritePin(gpio_port_7seg,
                gpio_pinmask_7seg[0]|gpio_pinmask_7seg[2]|gpio_pinmask_7seg[3]|gpio_pinmask_7seg[4]|gpio_pinma
                GPIO_PIN_RESET);
            break;
        case 7:
            /*
             * b'abcdefg = b'1110000
             */
            HAL_GPIO_WritePin(gpio_port_7seg,
                gpio_pinmask_7seg[0]|gpio_pinmask_7seg[1]|gpio_pinmask_7seg[2], GPIO_PIN_RESET);
            break;
        case 8:
            /*
             * b'abcdefg = b'1111111
             */
            HAL_GPIO_WritePin(gpio_port_7seg,
                gpio_pinmask_7seg[0]|gpio_pinmask_7seg[1]|gpio_pinmask_7seg[2]|gpio_pinmask_7seg[3]|gpio_pinma
                GPIO_PIN_RESET);
            break;
        case 9:
            /*
             * b'abcdefg = b'1111011
             */
            HAL_GPIO_WritePin(gpio_port_7seg,
                gpio_pinmask_7seg[0]|gpio_pinmask_7seg[1]|gpio_pinmask_7seg[2]|gpio_pinmask_7seg[3]|gpio_pinma
                GPIO_PIN_RESET);
            break;
        default:
            break;
    }
}

void clear_enable(void){
    uint8_t no_enable = sizeof(gpio_pinmask_enable_7segment);
    for (int var = 0; var < no_enable ; ++var) {
        HAL_GPIO_WritePin(gpio_port_enable, gpio_pinmask_enable_7segment[var], GPIO_PIN_SET);
    }
}

void set_enable(uint8_t index){
   HAL_GPIO_WritePin(gpio_port_enable, gpio_pinmask_enable_7segment[index], GPIO_PIN_RESET);
}
void display7SEG_Index(uint8_t index , uint8_t number){
   clear_enable();
   display7SEG(number);
   set_enable(index);
}

void display_7seg(){
   uint8_t number;
   if(get_flag_timer1()){
      switch (segment_index){
         case 0:
            number = counter_down_left_right/10;
            break;
         case 1:
            number = counter_down_left_right%10;
            break;
         case 2:
            number = counter_down_top_down/10;
            break;
         case 3:
            number = counter_down_top_down%10;
            break;
         default:
            break;
      }
      display7SEG_Index(segment_index, number);
      segment_index = (segment_index+1)%4;
//       HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_7);
      set_Timer1(10);
   }
}
```

- Blinking LED

```
        void blink_index(uint8_t color){
   if(get_flag_timer2()){
      switch (color) {
         case 0:
            HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_1);
            HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_4);
            break;
         case 1:
            HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_2);
            HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
            break;
         case 2:
            HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_3);
            HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_6);
            break;
         default:
            break;
```

```
    }
    set_Timer2(500);
  }
}
```

7 Execises 7

Your tasks in this exercise are:

- to use the second button to increase the time duration value of the red LEDs
- to use the third button to set the value for the red LEDs

7.1 Source code

```
    case 1:
      if(get_button_value(0)){
        mode = 2;
        segment_index = 0;
        reset_timer1();
        reset_timer2();
      }
      else{
        if(get_button_value(1)){
          temp_counter = (temp_counter+1)%100;
        }
        if(get_button_value(2)){
          MAX_RED_COUNTER = temp_counter;
        }
        //blink 0 ~ red
        blink_index(0);
        display_7seg_modify();
      }
      break;
```

8 Execises 8

Your tasks in this exercise are:

- to use the second button to increase the time duration value of the amber LEDs
- to use the third button to set the value for the amber LEDs

8.1 Source code

```
    case 2:
      if(get_button_value(0)){
        mode = 3;
        segment_index = 0;
        reset_timer1();
        reset_timer2();
      }
      else{
        if(get_button_value(1)){
          temp_counter = (temp_counter+1)%100;
        }
        if(get_button_value(2)){
          MAX_GREEN_COUNTER = temp_counter;
        }
        //blink 0 ~ red
        blink_index(1);
        display_7seg_modify();
      }
      break;
```

9 Execises 9

Your tasks in this exercise are:

- to use the second button to increase the time duration value of the red LEDs
- to use the third button to set the value for the red LEDs

9.1 Source code

```
    case 3:
      if(get_button_value(0)){
        mode = 0;
        segment_index = 0;
        reset_timer1();
        reset_timer2();
      }
      else{
        if(get_button_value(1)){
          temp_counter = (temp_counter+1)%100;
        }
        if(get_button_value(2)){
          MAX_YELLOW_COUNTER = temp_counter;
        }
        //blink 0 ~ red
        blink_index(2);
        display_7seg_modify();
      }
      break;
```

10 Execises 10

Your tasks in this exercise are:

- To integrate all the previous tasks to one final project
- To create a video to show all features in the specification
- To add a report to describe your solution for each exercise.
- To submit your report and code on the BKeL

Link Video: https://drive.google.com/file/d/1xBqAWVH-eUiJ0WgHUnHj4oME2YEM1FC0/view?usp=sharing