VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

Microprocessors-Microcontrollers

Assignment

Traffic lights

Advisor: Nguyễn Thiên Ân
Students: Trần Quốc Khải – 2252341

HO CHI MINH CITY, December 2024

Contents

1 Introduction

This project, we will simulate a real life traffic system control unit. The project will has 3 modes, 3 buttons to take input from users; and 2 traffic lights + 1lcd for output.

Modes:

- Mode 1:Normal traffic light
- Mode 2:Display and update value for red
- Mode 3:Display and update value for yellow
- Mode 4:Display and update value for green

Buttons:

- Button 1:Change modes
- Button 2:Increase value (All modes except 1)
- Button 3: Update respected value for mode 1

2 Implementation

2.1 Timer

Timer is one of the key aspect when first think of a traffic light. In this report, the timer interupt of STM32 is used.

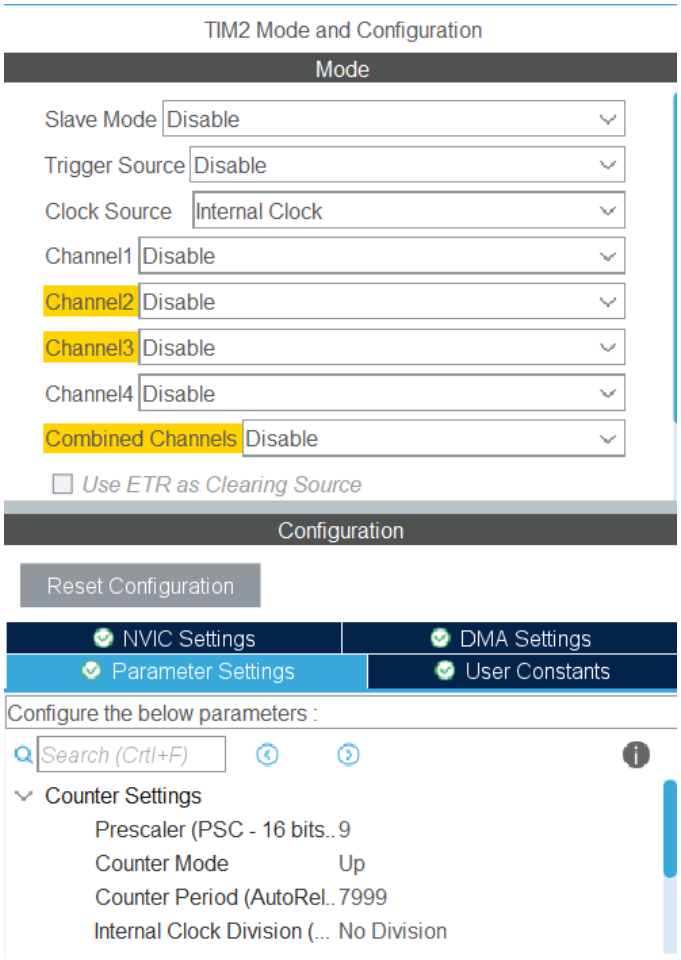

Figure 1: *Timer configuration*

The basic function of timer in this project is setTimer(). With each "set" function will comes with a "timer_counter" variable, and a "timer_flag" variable.

```
void setTimer(int duration){
  timer_counter = duration /TIMER_CYCLE;//TIMER_CYCLE = 10 ms due to configuration
  timer_flag = 0;
}
void timer_run(){
    if(timer_counter > 0){
        timer_counter--;
        if(timer_counter == 0) timer_flag = 1;
    }
}
```

2.2 Buttons and input

Debouncing is the problem when the button output state is chaotic in the first 10 ms. In this report and code, this specific problem is solve using timer and register.

When timer interupt is called, an input from a button will be call. That input will be updated to "memories". When all 3 of the "memories" is the same, program will access that the input is valid (whether it is press (RESET) or not pressed (SET)).

Moreover, the program also has another variable button_flag, and its job is to check the previous valid state is pressed(button_flag=0) or not pressed(button_flag=1). This extra variable

is to not accept a long pressed as 2 pressed.

The output of the timer is a variable Per_flag, outside timer function can look up its own variable that keeps the previous value of Per_flag, if different mean a button is pressed.

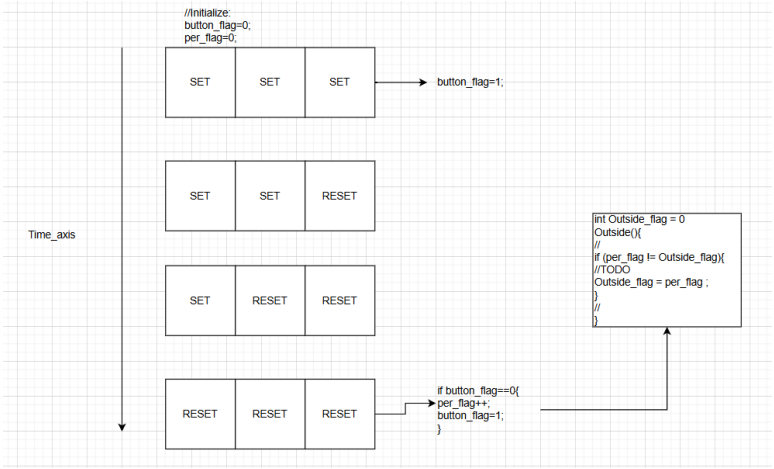

Figure 2: *Example of debounce*

2.3 LCD output

LCD will use an exist file i2c-ldc.c, and this project will use function lcd_goto_XY(x,y) and lcd_send_string().

lcd_goto_XY(input) will set the index of the other funtion, lcd_send_string() then will print out the input onto lcd on x,y.

Example how to use:

```
char* input;
input = "Hello";
lcd_goto_XY(0,0);
lcd_send_string(input);

char input_array[5] = "Hello";
input = &input_array[0];
lcd_goto_XY(1,0);
lcd_send_string(input);
```

2.4 Process mode function of traffic lights

This is a main part of this project's program, and it will be put at while loop in main function; and also call via timer flag every 250 ms.

The core structure of process mode function

```
int temp1,temp2,temp3;
int Per_flag1_buffer = 0;
//Other variables
while(1){
    if (timer0_flag!=1) continue;
    setTimer0(250);
    if (Per_flag!= Per_flag1_buffer){//Per\_flag is declared and update on timer interupt
        Per_flag1_buffer=Per_flag;
        mode_1_counter=0;//Will come in use in case 1-modes
        modes++;
        if (modes>=5 || modes<=0)modes=1;
```

```
        temp1 = LED_Timer[0];
        temp2 = LED_Timer[2];
        temp3 = LED_Timer[1];
      }
    char* row1;
    char* row2;
    char temp1_array[9]="A =      ";
    char temp2_array[12]="B =       \0";
    char temp3_array[12]="Var =    \0";
    switch (modes){
    case 1:
        //CODE
        break;
    case 2:
        //CODE
        break;
    case 3:
        //CODE
        break;
    case 4:
        //CODE
        break;
    default:
        break
    }
    lcd_goto_XY(0, 0);
    lcd_send_string(row1);
    lcd_goto_XY(1, 0);
    lcd_send_string(row2);
}
```

The first statement of above code is to check time_flag and check if button 1 is pressed, which is important because if the button will change the modes hence change the behaviour of our program.

There are also row1, and row2. It will output to lcd (the code to update can be seen at the end).

Look closer into case 1

```
if (mode_1_counter==0){
  mode_1_modes0 = 0;
  mode_1_modes1 = 1;
  UpdateLedBuffer(LED_Timer[0]);
  UpdateLedBuffer2(LED_Timer[1]);
HAL_GPIO_WritePin(OUTPUT_1_GPIO_Port, OUTPUT_1_Pin, SET);
HAL_GPIO_WritePin(OUTPUT_2_GPIO_Port, OUTPUT_2_Pin, SET);

HAL_GPIO_WritePin(OUTPUT_3_GPIO_Port, OUTPUT_3_Pin, RESET);
HAL_GPIO_WritePin(OUTPUT_4_GPIO_Port, OUTPUT_4_Pin, SET);

}
  if (mode_1_counter<4){//250*4=1
      mode_1_counter++;
      break;
  }
  mode_1_counter=1;
  //Start code for 1st intersection
  int temp_var1 = 0;
  temp_var1 = GetLEDBuffer();
  temp_var1--;
  if (temp_var1<=0){
      mode_1_modes0++;
```

```
      if (mode_1_modes0>=3||mode_1_modes0<0) mode_1_modes0=0;
      temp_var1 = LED_Timer[mode_1_modes0];
  }
  switch (mode_1_modes0){
    case 0:
        HAL_GPIO_WritePin(OUTPUT_1_GPIO_Port, OUTPUT_1_Pin, SET);
        HAL_GPIO_WritePin(OUTPUT_2_GPIO_Port, OUTPUT_2_Pin, SET);
        break;
    case 1:
        HAL_GPIO_WritePin(OUTPUT_1_GPIO_Port, OUTPUT_1_Pin, SET);
        HAL_GPIO_WritePin(OUTPUT_2_GPIO_Port, OUTPUT_2_Pin, RESET);
        break;
    case 2:
        HAL_GPIO_WritePin(OUTPUT_1_GPIO_Port, OUTPUT_1_Pin, RESET);
        HAL_GPIO_WritePin(OUTPUT_2_GPIO_Port, OUTPUT_2_Pin, SET);
        break;
    default:
      break;
  }
    UpdateLedBuffer(temp1);
    To_char2(&temp1_array[3],temp1/10);
    To_char2(&temp1_array[4],temp1%10);
    row1 = &temp1_array[0];
    //End code for 1st intersection
////////////////////////////////////////////////
    //For 2nd intersection
    break;//break for case of modes
```

Other relevant code to case 1:

```
int LED0 = 0;
int GetLEDBuffer(){
  return LED0;
}
void UpdateLedBuffer(int in){
  LED0 = in;
}
int LED1 = 0;
int GetLEDBuffer2(){
  return LED1;
}
void UpdateLedBuffer2(int in){
  LED1 = in;
}
```

The main purpose of above code, is to remember the counting value of mode 1.

Look closer into case 2

```
case 2:
    if (Per_Toggle_LED==0){
    Per_Toggle_LED=1;
    HAL_GPIO_WritePin(OUTPUT_1_GPIO_Port, OUTPUT_1_Pin, SET);
    HAL_GPIO_WritePin(OUTPUT_2_GPIO_Port, OUTPUT_2_Pin, SET);
    HAL_GPIO_WritePin(OUTPUT_3_GPIO_Port, OUTPUT_3_Pin, SET);
    HAL_GPIO_WritePin(OUTPUT_4_GPIO_Port, OUTPUT_4_Pin, SET);
    }
    else{
    Per_Toggle_LED=0;
    HAL_GPIO_WritePin(OUTPUT_1_GPIO_Port, OUTPUT_1_Pin, RESET);
    HAL_GPIO_WritePin(OUTPUT_2_GPIO_Port, OUTPUT_2_Pin, RESET);
    HAL_GPIO_WritePin(OUTPUT_3_GPIO_Port, OUTPUT_3_Pin, RESET);
    HAL_GPIO_WritePin(OUTPUT_4_GPIO_Port, OUTPUT_4_Pin, RESET);
    }
```

```
    row1 = "Mode: 2";
    row2 = "Counter = ";


    if (Per_flag2 != Per_flag2_buffer ){
    Per_flag2_buffer = Per_flag2;
      temp2++;
      if (temp2>=100)temp2=0;

      }

    if (Per_flag3_buffer != Per_flag3 ){
      Per_flag3_buffer = Per_flag3;
      LED_Timer[2] = LED_Timer[2] + (temp2-LED_Timer[0]);
      LED_Timer[0] = temp2;
      }

    To_char2(&temp3_array[5],temp2/10);
    To_char2(&temp3_array[6],temp2%10);
    row2 = &temp3_array[0];
    break;
```

Case 3, and 4 is similar to a degree of case2

The thing to noted that, when

update red an amount, the same amount is added to green. Yellow will change green. Green will change yellow.