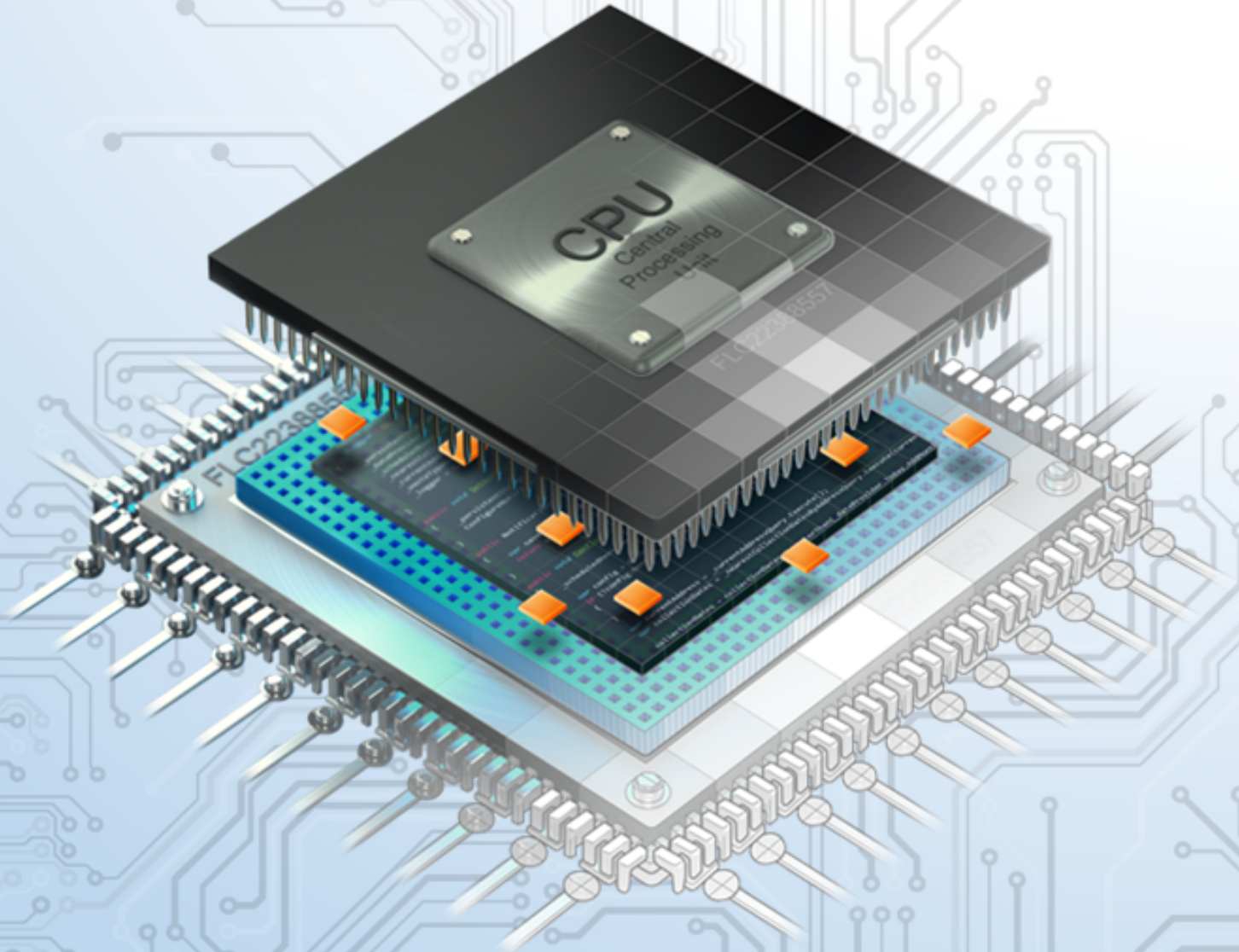
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
COMPUTER ENGINEERING

Microcontroller

LABORATORY REPORT



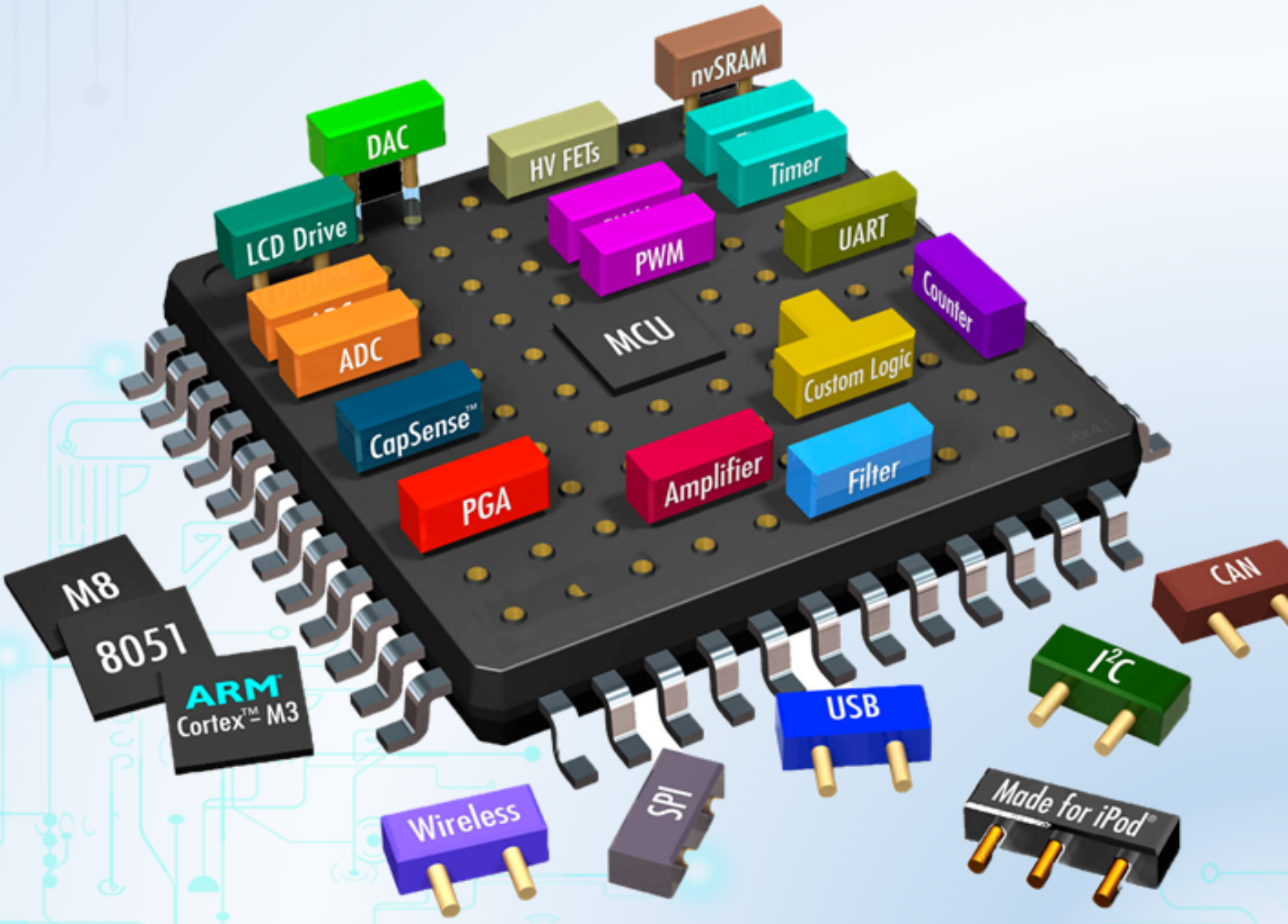
Sep - Nov 2022

Cao Minh Quang - 2052221

Contents

CHAPTER 1

Buttons/Switches



1 Specifications

You are required to build an application of a traffic light in a cross road which includes some features as described below:

- The application has 12 LEDs including 4 red LEDs, 4 amber LEDs, 4 green LEDs.
- The application has 4 seven segment LEDs to display time with 2 for each road. The 2 seven segment LEDs will show time for each color LED corresponding to each road.
- The application has three buttons which are used

  - to select modes,

  - to modify the time for each color led on the fly, and

  - to set the chosen value.
- The application has at least 4 modes which is controlled by the first button. Mode 1 is a normal mode, while modes 2 3 4 are modification modes. You can press the first button to change the mode. Modes will change from 1 to 4 and back to 1 again.

  Mode 1 - Normal mode:

  - The traffic light application is running normally.

  Mode 2 - Modify time duration for the red LEDs: This mode allows you to change the time duration of the red LED in the main road. The expected behaviours of this mode include:

  - All single red LEDs are blinking in 2 Hz.

  - Use two seven-segment LEDs to display the value.

  - Use the other two seven-segment LEDs to display the mode.

  - The second button is used to increase the time duration value for the red LEDs.

  - The value of time duration is in a range of 1 - 99.

  - The third button is used to set the value.

  Mode 3 - Modify time duration for the amber LEDs: Similar for the red LEDs described above with the amber LEDs.

  Mode 4 - Modify time duration for the green LEDs: Similar for the red LEDs described above with the green LEDs.

2 Github for version control

- Please access the following link to gain full control of this lab:
- `https://github.com/cmq2002/Simple_Traffic_Lights_with_Modes`

3 Exercise 1: Sketch an FSM

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

Please add your report here.

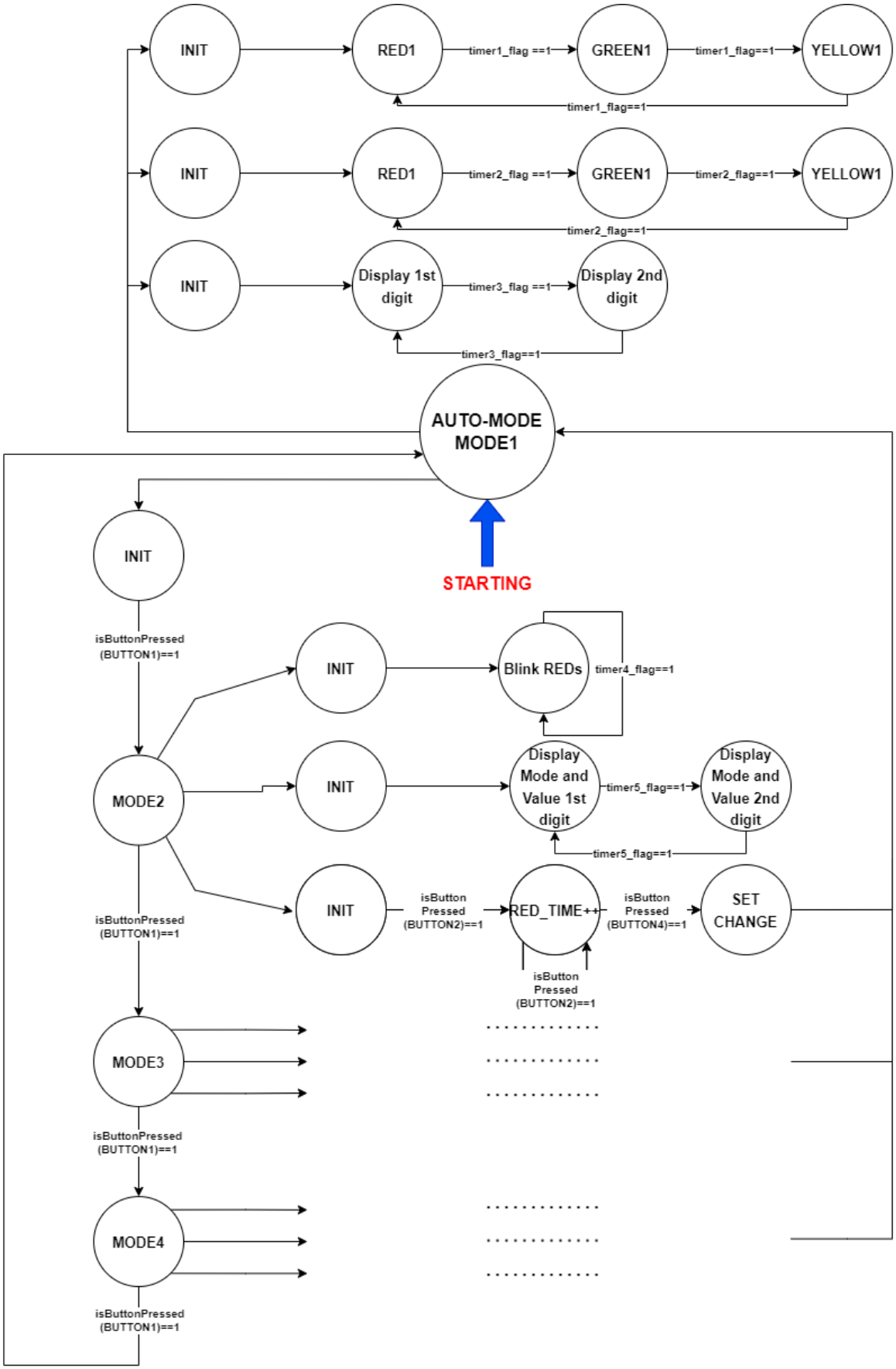


Figure 1.1: The FSM for the system.

- In Mode 3 and Mode 4, since the behaviors are the same with that of Mode 2, I have left dots symbol to simplified our diagram.
- Additionally, I have also attempted to implemented another button which is used for decreasing the waiting time.
- Hence, the mechanism for Mode 2 - Mode 4 with the ability to increase and decrease waiting time will be a little bit different from the above.

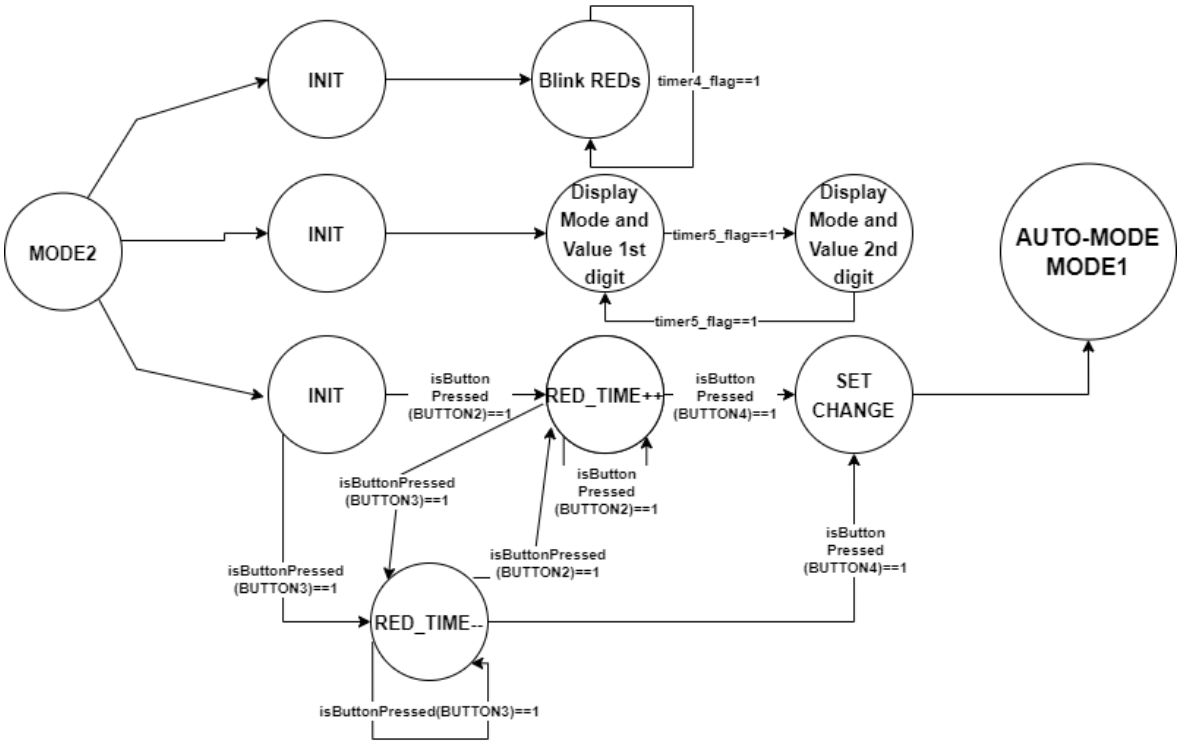


Figure 1.2: The FSM for the decreasing waiting time function.

4 Exercise 2: Proteus Schematic

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

Please add your report here.

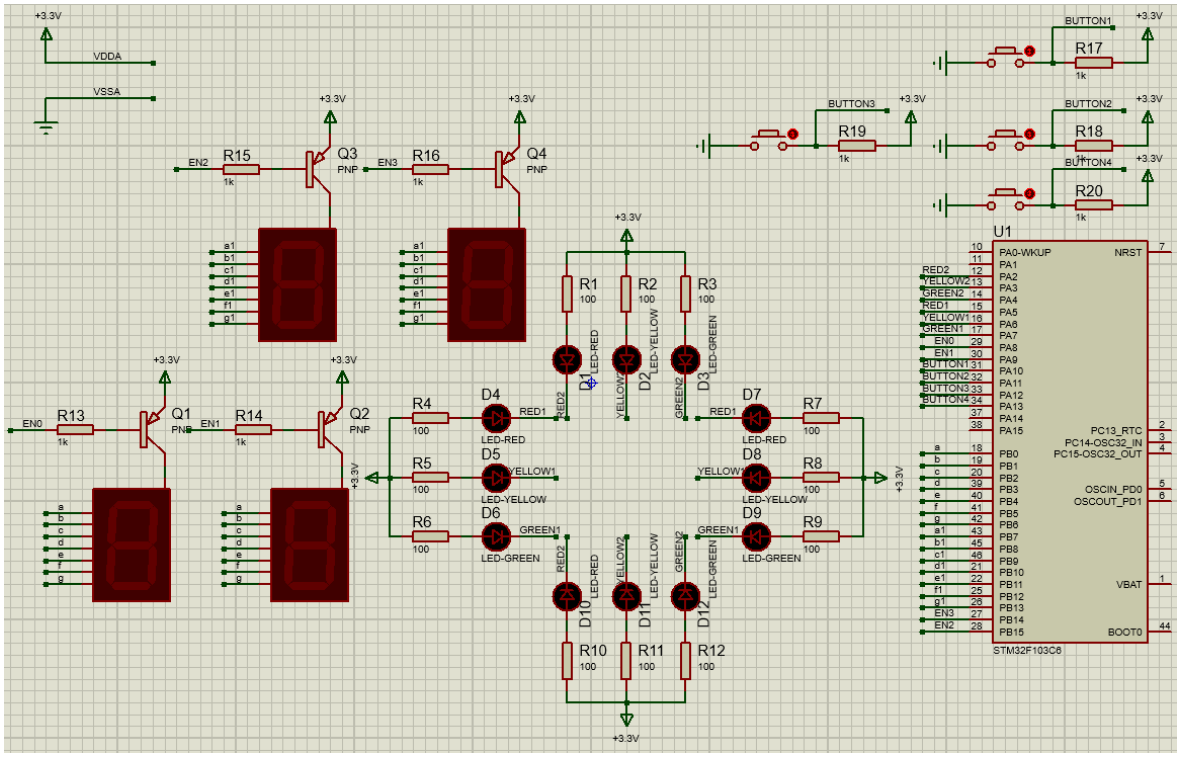


Figure 1.3: Schematic design in Proteus.

5 Exercise 3: Create STM32 Project

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.

Please add your report here.

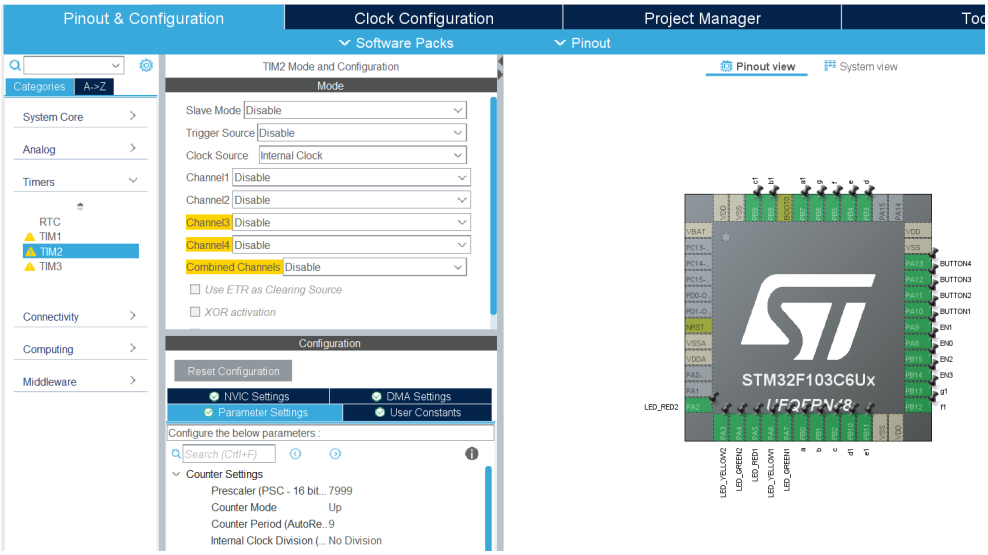


Figure 1.4: Timer interrupt setup.

- Prescaler = 7999 and Counter = 9 give us the 10ms timer interrupt.

6 Exercise 4: Modify Timer Parameters

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.

Please add your report here.

- When changing the time to invoke the interrupt, to make the system's nature unchanged, we need to use the software timer in order to synchronize the operation as we desire.
- The below code segment is the implementation of the software timer. We will create a header file and the corresponding C source file.

```
#ifndef INC_SOFTWARE_TIMER_H_
#define INC_SOFTWARE_TIMER_H_

extern int timer1_flag;

void setTimer1(int duration);

void timerRun();

#endif /* INC_SOFTWARE_TIMER_H_ */

```

Program 1.1: softwareTimer.h

```
#include "software_timer.h"

int timer1_counter = 0;
int timer1_flag = 0;

void setTimer1(int duration){
  timer1_counter = duration;
  timer1_flag = 0;
}

void timerRun(){
  if(timer1_counter > 0){
    timer1_counter--;
    if(timer1_counter == 0){
      timer1_flag = 1;
    }
```

```
17        }
18    }
19
```

Program 1.2: softwareTimer.c

- The mechanism behind is quite simple. We only need to set the timer with the number we wanted. timerRun() is expected to be put inside the function:

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *
    htim)
{
    timerRun();
}
```

- timerRun() simply counts down, whenever it reaches 0, it will raise a flag. And then this is the signal for us to perform our task.
- Let consider the situation when interrupt is invoked every 2ms not 10ms anymore. We want to maintain the 2Hz blinking Led, so we have to setTimer(125) since the duration is 0.5s, which means led on in 0.25s and off 0.25s. The number 125 here indicates that we will blink the led only when the interrupt is invoked 125 times.
- Overall, all we need to do is that to compute the right number the function setTimer() no matter how long the interrupt is invoked. Futhermore, we can also clone the variable and the function in the software timer if we need.

7 Exercise 5: Adding code for button debouncing

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.

Please add your report here.

- Code for button reading and button debouncing:

```
#ifndef INC_BUTTON_H_
#define INC_BUTTON_H_

#include "main.h"
#include "global.h"

#define NORMAL_STATE GPIO_PIN_SET
#define PRESSED_STATE GPIO_PIN_RESET
```

```
9
10 int isButtonPressed();
11
12 void getKeyInput();
13
14 #endif /* INC_BUTTON_H_ */
```

Program 1.3: button.h

```
1  #include "button.h"
2
3  int buttonList[NUM_OF_BUTTON] = {BUTTON1_Pin, BUTTON2_Pin,
       BUTTON3_Pin, BUTTON4_Pin};
4
5  int KeyReg0[NUM_OF_BUTTON] = {NORMAL_STATE, NORMAL_STATE,
       NORMAL_STATE, NORMAL_STATE};
6  int KeyReg1[NUM_OF_BUTTON] = {NORMAL_STATE, NORMAL_STATE,
       NORMAL_STATE, NORMAL_STATE};
7  int KeyReg2[NUM_OF_BUTTON] = {NORMAL_STATE, NORMAL_STATE,
       NORMAL_STATE, NORMAL_STATE};
8  int KeyReg3[NUM_OF_BUTTON] = {NORMAL_STATE, NORMAL_STATE,
       NORMAL_STATE, NORMAL_STATE};
9
10 int TimeOutForKeyPress =  500;
11 int button_flag[NUM_OF_BUTTON] = {0, 0, 0, 0};
12 int button_record[NUM_OF_BUTTON] = {0, 0, 0, 0};
13
14 int isButtonPressed(int index){
15   if(button_flag[index] == 1){
16     button_flag[index] = 0;
17     return 1;
18   }
19   return 0;
20 }
21
22 void subKeyProcess(int index){
23   button_flag[index] = 1;
24 }
25
26 void getKeyInput(){
27   for(int i=0; i<NUM_OF_BUTTON; i++){
28     KeyReg2[i] = KeyReg1[i];
29     KeyReg1[i] = KeyReg0[i];
30     KeyReg0[i] = HAL_GPIO_ReadPin(GPIOA, buttonList[i]);
31     if ((KeyReg1[i] == KeyReg0[i]) && (KeyReg1[i] ==
       KeyReg2[i])){
32     if (KeyReg2[i] != KeyReg3[i]){
33       KeyReg3[i] = KeyReg2[i];
34
35       if (KeyReg3[i] == PRESSED_STATE){
```

```
36          TimeOutForKeyPress = 500;
37          subKeyProcess(i);
38          }
39      }else{
40           TimeOutForKeyPress --;
41          if (TimeOutForKeyPress == 0){
42            KeyReg3[i] = NORMAL_STATE;
43          }
44      }
45      }
46    }
47  }
```

Program 1.4: button.c

8 Exercise 6: Adding code for displaying modes

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.

Please add your report here.

```
void fsm_manual_run(){
  if (mode == MODE2){
    //Blink LEDs
    switch (statusMODE2_1){
      case INIT:
        statusMODE2_1 = TOGGLE;
        setTimer4(1);
        break;
      case TOGGLE:
        if (timer4_flag == 1){
          toggleREDs();
          setTimer4(25);
        }
        break;
      default:
        break;
    }

    //Show mode and value
    switch (statusMODE2_2){
      case INIT:
        statusMODE2_2 = DOZEN;
        setTimer5(1);
        break;
      case DOZEN:
        if (timer5_flag == 1){
          blinkDigit1(mode, AUTO_RED);
          statusMODE2_2 = UNIT;
          setTimer5(25);
        }
        break;
      case UNIT:
        if (timer5_flag == 1){
          blinkDigit2(mode, AUTO_RED);
          statusMODE2_2 = DOZEN;
          setTimer5(25);
        }
        break;
      default:
        break;
```

```
    }
}
```

9 Exercise 7: Adding code for increasing time duration value for the red LEDs

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.

Please add your report here.

```
void fsm_manual_run(){
    if (mode == MODE2){
    // Changing Red Light Waiting Time
    switch(statusMODE2_3){
      case INIT:
        if (isButtonPressed(BUTTON2)==1){
          statusMODE2_3 = CHANGE;
          AUTO_RED+=1;
        }
        break;
      case CHANGE:
        if (isButtonPressed(BUTTON2)==1){
          if (AUTO_RED > UPPER_BOUND) AUTO_RED = UPPER_BOUND;
          if (AUTO_RED < LOWER_BOUND) AUTO_RED = LOWER_BOUND;
          AUTO_RED += 1;
        }
        if (isButtonPressed(BUTTON4)==1) statusMODE2_3 = SAVE;
        break;
      case SAVE:
        mode = MODE1;
        initVar();
        break;
      default:
        break;
    }

    if (isButtonPressed(BUTTON1)==1){
      mode = MODE3;
    }
  }
}
```

10 Exercise 8: Adding code for increasing time duration value for the amber LEDs

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.

Please add your report here.

```
void fsm_manual_run(){
  if (mode == MODE3){
    //Blink LEDs
    switch (statusMODE3_1){
      case INIT:
        statusMODE3_1 = TOGGLE;
        setTimer6(1);
        break;
      case TOGGLE:
        if (timer6_flag == 1){
          toggleGREENs();
          setTimer6(25);
        }
        break;
      default:
        break;
    }

    //Show mode and value
    switch (statusMODE3_2){
      case INIT:
        statusMODE3_2 = DOZEN;
        setTimer7(1);
        break;
      case DOZEN:
        if (timer7_flag == 1){
          blinkDigit1(mode, AUTO_GREEN);
          statusMODE3_2 = UNIT;
          setTimer7(25);
        }
        break;
      case UNIT:
        if (timer7_flag == 1){
          blinkDigit2(mode, AUTO_GREEN);
          statusMODE3_2 = DOZEN;
          setTimer7(25);
        }
        break;
```

```
      default:
        break;
    }

    // Changing Green Light Waiting Time
    switch(statusMODE3_3){
      case INIT:
        if (isButtonPressed(BUTTON2)==1){
          statusMODE3_3 = CHANGE;
          AUTO_GREEN+=1;
        }
        break;
      case CHANGE:
        if (isButtonPressed(BUTTON2)==1){
          if (AUTO_GREEN > UPPER_BOUND) AUTO_GREEN =
  UPPER_BOUND;
          if (AUTO_GREEN < LOWER_BOUND) AUTO_GREEN =
  LOWER_BOUND;
          AUTO_GREEN += 1;
        }
        if (isButtonPressed(BUTTON4)==1) statusMODE3_3 =
  SAVE;
        break;
      case SAVE:
        mode = MODE1;
        initVar();
        break;
      default:
        break;
    }

    if (isButtonPressed(BUTTON1)==1){
      mode = MODE4;
    }
  }
}
```

11 Exercise 9: Adding code for increasing time duration value for the green LEDs

Your tasks in this exercise are:

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

Please add your report here.

```c
void fsm_manual_run(){
  if (mode == MODE4){
    //Blink LEDs
    switch (statusMODE4_1){
      case INIT:
        statusMODE4_1 = TOGGLE;
        setTimer8(1);
        break;
      case TOGGLE:
        if (timer8_flag == 1){
          toggleYELLOWs();
          setTimer8(25);
        }
        break;
      default:
        break;
    }

    //Show mode and value
    switch (statusMODE4_2){
      case INIT:
        statusMODE4_2 = DOZEN;
        setTimer9(1);
        break;
      case DOZEN:
        if (timer9_flag == 1){
          blinkDigit1(mode, AUTO_YELLOW);
          statusMODE4_2 = UNIT;
          setTimer9(25);
        }
        break;
      case UNIT:
        if (timer9_flag == 1){
          blinkDigit2(mode, AUTO_YELLOW);
          statusMODE4_2 = DOZEN;
          setTimer9(25);
        }
        break;
      default:
        break;
    }

    // Changing YELLOW Light Waiting Time
    switch(statusMODE4_3){
      case INIT:
        if (isButtonPressed(BUTTON2)==1){
          statusMODE4_3 = CHANGE;
          AUTO_YELLOW+=1;
        }
```

```
50          break;
51        case CHANGE:
52          if (isButtonPressed(BUTTON2)==1){
53            if (AUTO_YELLOW > UPPER_BOUND) AUTO_YELLOW =
    UPPER_BOUND;
54            if (AUTO_YELLOW < LOWER_BOUND) AUTO_YELLOW =
    LOWER_BOUND;
55            AUTO_YELLOW += 1;
56          }
57          if (isButtonPressed(BUTTON4)==1) statusMODE4_3 =
    SAVE;
58          break;
59        case SAVE:
60          mode = MODE1;
61          initVar();
62          break;
63        default:
64          break;
65      }
66
67      if (isButtonPressed(BUTTON1)==1){
68        mode = MODE1;
69        initVar();
70      }
71    }
72  }
```

12 Exercise 10: To finish the project

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