VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
Faculty of Computer Science and Engineering

CC02 — Lab Report

Microprocessor - Microcontroller
Lab 2

Supervisors: Nguyen Thien An
Students: Nguyen Minh Dang 2252154

Ho Chi Minh City, October 1, 2024

Contents

1 Exercise

The schematic files and source code for each exercise lab are in this GitHub repository (included PNG, PDSPRJ and C):

https://github.com/dangalpha78/Workspace-for-Microprocessor---Microcontroller/tree/main/Lab2

The schematic for the exercise 1:

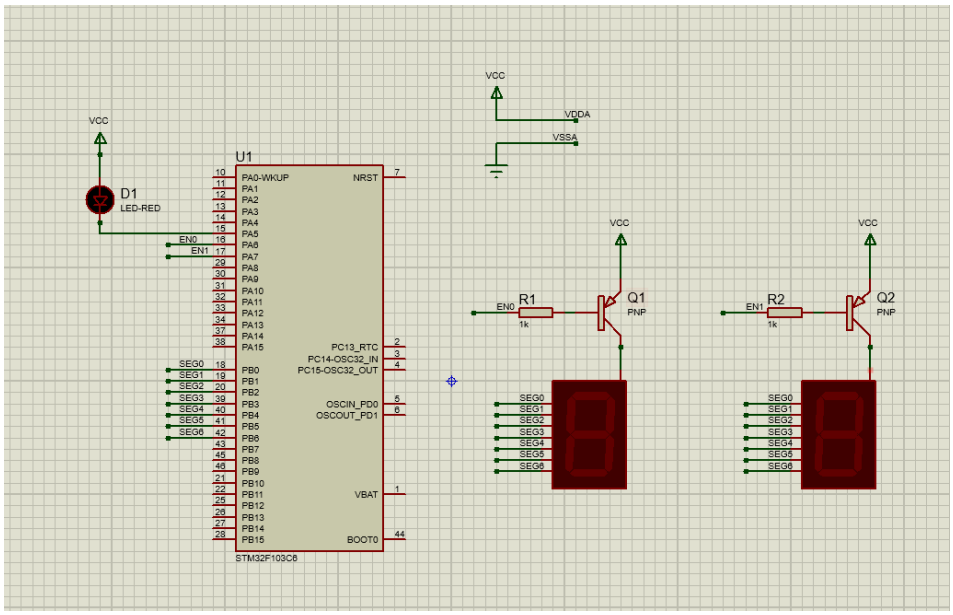

Figure 1: *The schematic for the exercise 1.*

The schematic for the exercises from 2 to 8 is located here:

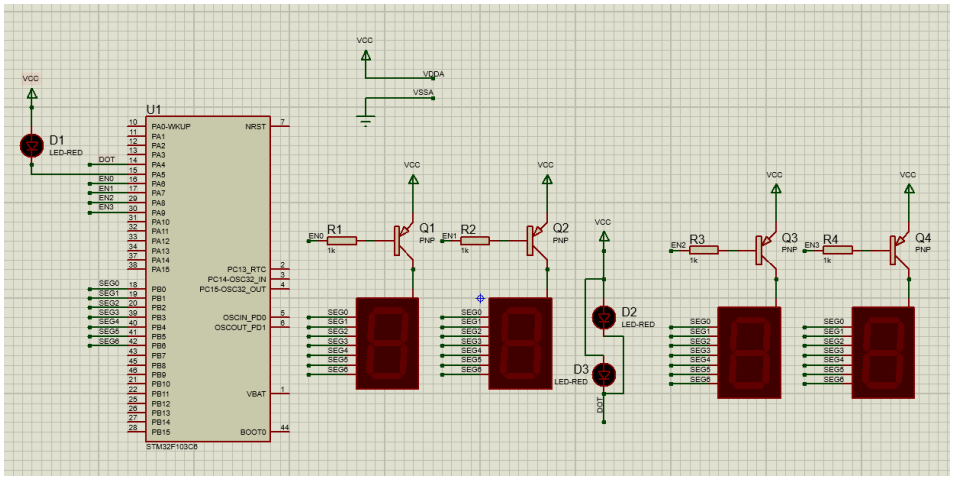

Figure 2: *The schematic for the exercises from 2 to 8.*

The schematic for the exercises 9 and 10 is located here:

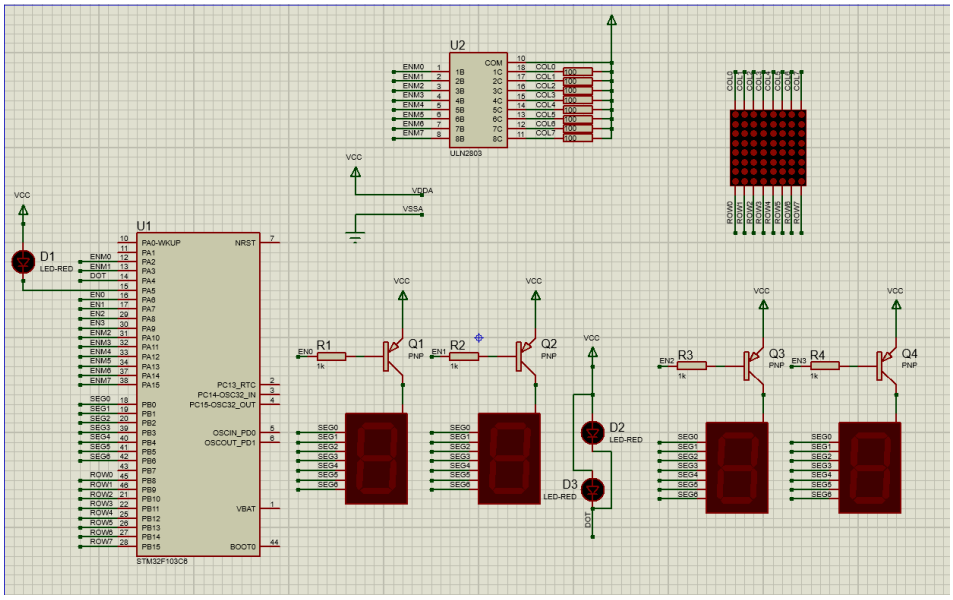

Figure 3: *The schematic for the exercises 9 and 10.*

1.1 Exercise 1

1.1.1 Report 1

The schematic for exercise 1:

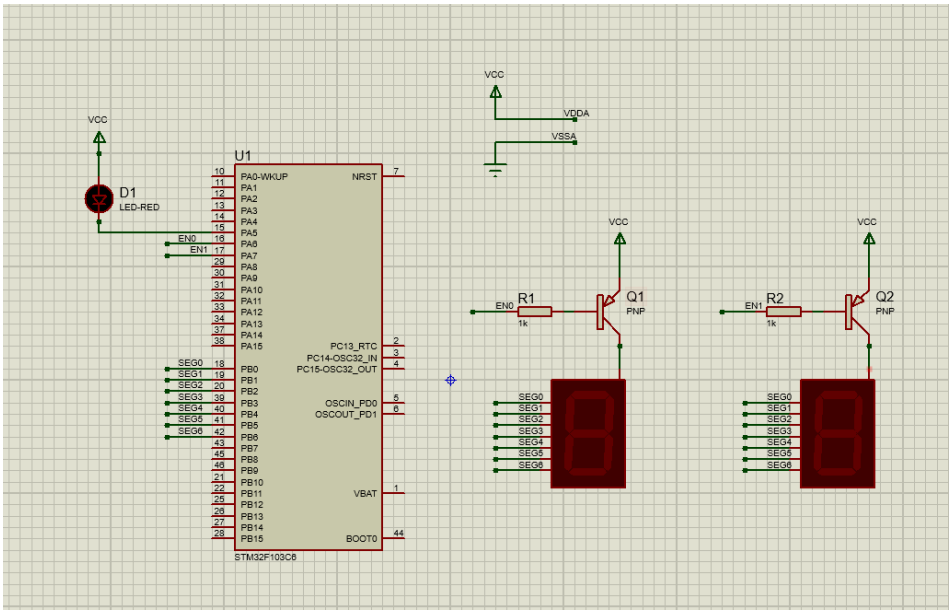

Figure 4: *The schematic for the exercise 1.*

Or can be found at GitHub Schematic Ex1, Proteus, Source Code Ex 1

1.1.2 Report 2

```
/* USER CODE BEGIN 0 */
void display7SEG(int num); //this func already declared in lab 1

void led_red() //control the led red
{
  HAL_GPIO_TogglePin(LED5_GPIO_Port, LED5_Pin);
}

void SW(int sw){
  switch(sw){
  case 1: //turn on left led ("1" led)
    HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, RESET);
    HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, SET);
    break;
  case 2: //turn on right led ("2" led)
    HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, SET);
    HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, RESET);
    break;

```

```
  default:
    HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, SET);
    HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, SET);
    break;
  }
}
/* USER CODE END 0 */

//...
//...

/* USER CODE BEGIN 4 */
const int TIME_COUNTER = 50; //1Hz
int counter = TIME_COUNTER; //init counter
int sw_counter = 1; //init switch status
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
//this func be called every 10ms
{
  counter--; //execute this command 50 times => 10ms x 50 = 500ms
  if( counter <= 0) { //switch state
    counter = TIME_COUNTER;
    led_red();
    if (sw_counter == 1) sw_counter = 2;  //switch "1" led to "2" led
    else sw_counter = 1;                  //switch "2" led to "1" led
  }
  SW(sw_counter);
  display7SEG(sw_counter);
}
/* USER CODE END 4 */
```

The frequency of the scanning process is of 2 seven-segment LEDs **1 HZ**. The switching time between the two displays is 0.5 seconds, so the total cycle time is $0.5 \times 2 = 1$ second. And the $f = 1/cycle = 1/1 = 1Hz$.

1.2 Exercise 2

1.2.1 Report 1

The schematic for exercise 2:

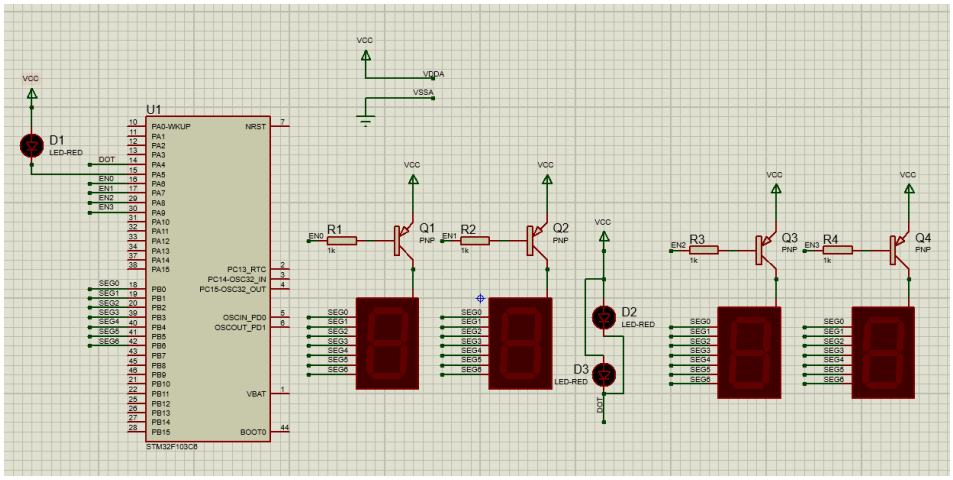

Figure 5: *The schematic for the exercise 2.*

Or can be found at GitHub Schematic Ex2, Proteus, Source Code Ex 2

1.2.2 Report 2

```
/* USER CODE BEGIN 0 */
void display7SEG(int num); //this func already declared in lab 1

void led_red(){
  HAL_GPIO_TogglePin(LED5_GPIO_Port, LED5_Pin);
  HAL_GPIO_TogglePin(DOT_GPIO_Port, DOT_Pin);
}

//This function toggles the state between EN0, EN1, EN2, EN3 (LED 1, 2, 3, and 4)
void SW(int sw){
  switch(sw){
  case 1:
    HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, RESET);
    HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, SET);
    HAL_GPIO_WritePin(EN2_GPIO_Port, EN2_Pin, SET);
    HAL_GPIO_WritePin(EN3_GPIO_Port, EN3_Pin, SET);
    break;
  case 2:
    HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, SET);
    HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, RESET);
    HAL_GPIO_WritePin(EN2_GPIO_Port, EN2_Pin, SET);
    HAL_GPIO_WritePin(EN3_GPIO_Port, EN3_Pin, SET);
    break;
```

```
24    case 3:
25      HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, SET);
26      HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, SET);
27      HAL_GPIO_WritePin(EN2_GPIO_Port, EN2_Pin, RESET);
28      HAL_GPIO_WritePin(EN3_GPIO_Port, EN3_Pin, SET);
29      break;
30    case 0:
31      HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, SET);
32      HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, SET);
33      HAL_GPIO_WritePin(EN2_GPIO_Port, EN2_Pin, SET);
34      HAL_GPIO_WritePin(EN3_GPIO_Port, EN3_Pin, RESET);
35      break;
36    default:
37      HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, SET);
38      HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, SET);
39      HAL_GPIO_WritePin(EN2_GPIO_Port, EN2_Pin, SET);
40      HAL_GPIO_WritePin(EN3_GPIO_Port, EN3_Pin, SET);
41      break;
42    }
43  }
44  /* USER CODE END 0 */
45  //...
46  //THERE IS STILL CODE HERE
47  //...
48
49  /* USER CODE BEGIN 4 */
50  const int TIME_COUNTER = 50; //0.5Hz
51  int counter = TIME_COUNTER;
52  int sw_counter = 1;
53  int dot_led_counter = 100;
54  void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
55  //this func be called every 10ms
56  {
57    if(counter <= 0) {
58      counter = TIME_COUNTER;
59      sw_counter++;
60      if (counter >= 4) counter = 0;
61    }
62    if (dot_led_counter <= 0) //dot led switch state
63          {
64      led_red();
65      dot_led_counter = 100;
66    }
67    SW(sw_counter);
68    display7SEG(sw_counter);
69    counter--;
70    dot_led_counter--;
71  }
72  /* USER CODE END 4 */
```

The frequency of the scanning process of 4 seven-segment LEDs is **0.5 HZ**. The switching time between the two displays is 0.5 seconds, so the total cycle time is $0.5 \times 4 = 2$ seconds. And the $f = 1/cycle = 1/2 = 0.5Hz$.

1.3 Exercise 3

1.3.1 Report 1

Can be found at GitHub Schematic Ex3, Proteus, Source Code Ex 3

1.3.2 Report 2

```
/* USER CODE BEGIN 0 */
//... display7SEG(int num)
//... led_red(), SW(int sw) (can find it in the previous exercise)
const int MAX_LED = 4;
int index_led = 0;
int led_buffer[4] = {1 , 2 , 3 , 0};
void update7SEG ( int index ) {
    switch ( index ) {
    case 0:
      SW(index);
      display7SEG(led_buffer[index]);
        break ;
    case 1:
      SW(index);
      display7SEG(led_buffer[index]);
        break ;
    case 2:
      SW(index);
      display7SEG(led_buffer[index]);
        break ;
    case 3:
      SW(index);
      display7SEG(led_buffer[index]);
        break ;
    default:
        break ;
    }
}
/* USER CODE END 0 */

//...
//THERE IS STILL CODE HERE
//...

/* USER CODE BEGIN 4 */
const int TIME_COUNTER = 50; //SET 0.5Hz
int counter = TIME_COUNTER;
//int sw_counter = 0;
int num = 1;
int dot_led_counter = 100; //SET 2Hz
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
//this func be called every 10ms
```

```
{
  if( counter <= 0) {
    counter = TIME_COUNTER;
    index_led++;
    index_led = index_led % 4;
  }
  if (dot_led_counter <= 0){
    led_red();
    dot_led_counter = 100;
  }
  update7SEG(index_led);
  //display7SEG(index_led);
  counter--;
  dot_led_counter--;
}
/* USER CODE END 4 */
```

1.4 Exercise 4

Exercise 4 is almost as same as exercise 3 so I use same source code for both. Just modify $TIME\_COUNTER = 25$

Or can be found at GitHub Schematic Ex4, Proteus, Source Code Ex 4

1.4.1 Report 1

```
//...
//THERE IS CODE HERE
//...
/* USER CODE BEGIN 4 */
const int TIME_COUNTER = 25; // SET 1Hz
int counter = TIME_COUNTER;
//int sw_counter = 0;
int num = 1;
int dot_led_counter = 100; // SET 1Hz
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
//this func be called every 10ms
{

  if( counter <= 0) {
    counter = TIME_COUNTER;
    index_led++;
    index_led = index_led % 4;
  }
  if (dot_led_counter <= 0){
    led_red();
    dot_led_counter = 100;
  }
  update7SEG(index_led);
  //display7SEG(index_led);
  counter--;
  dot_led_counter--;
}
/* USER CODE END 4 */
```

The frequency of the scanning process of 4 seven-segment LEDs is **1 HZ**. The switching time between the two displays is 0.25 seconds, so the total cycle time is $0.25 \times 4 = 1$ second. And the $f = 1/cycle = 1/1 = 1Hz$.

1.5 Exercise 5

1.5.1 Report 1

Can be found at GitHub Schematic Ex5, Proteus, Source Code Ex 5

```
//...
//THERE IS STILL CODE HERE
//...
int hour = 15 , minute = 8 , second = 50;

void updateClockBuffer(){
  led_buffer[0] = hour / 10;
  led_buffer[1] = hour % 10;
  led_buffer[2] = minute / 10;
  led_buffer[3] = minute % 10;
}
//...
//THERE IS STILL CODE HERE
//...
```

1.6 Exercise 6

Can be found at GitHub Schematic Ex6, Proteus, Source Code Ex 6

1.6.1 Report 1

```
1  setTimer0(1000);
2  while (1)
3    {
4      /* USER CODE END WHILE */
5      if( timer0_flag == 1) {
6        HAL_GPIO_TogglePin (LED5_GPIO_Port, LED5_Pin);
7        setTimer0(2000);
8      }
9      /* USER CODE BEGIN 3 */
10   }
11   /* USER CODE END 3 */
```

If in line 1 of the code above is miss, what happens after that and why?

Answer: If the command `setTimer0(1000);` disappears, the value of `timer0_counter` will be 0, which means that the condition `if(timer0_counter > 0)` in the `timer_run()` function will never be accessed, and `timer0_flag` will never be equal to 1. This prevents the statements within `if(timer0_flag == 1)` in the while loop from executing. As a result, the LED will remain in its initial state (in this case, the initial state is ON).

1.6.2 Report 2

If in line 1 of the code above is changed to `setTimer0(1)`, what happens after that and why?

Answer: If `setTimer0(1)` is called, then `timer0_counter = duration / TIMER_CYCLE;` will have a value of 0 (since `duration = 1` and `TIMER_CYCLE = 10`). Therefore, `timer0_counter` remains 0, and the result is the same as in Report 1. This means that the LED will retain its initial state (in this case, the initial state is ON).

1.6.3 Report 3

If in line 1 of the code above is changed to `setTimer0(10)`, what is changed compared to 2 first questions and why?

Answer: At this point, `timer0_counter` will have a value of 1 (instead of being 0 as in the previous two reports) and will execute `if(timer0_counter > 0)` once. At that moment, `timer0_flag` will be 1, satisfying the condition `if(timer0_flag == 1)` in the while loop, causing LED5 to toggle its state within 10 ms. After that, `setTimer0(2000)` is called, and the process repeats sequentially, but the time for state transition will now be 2 seconds.

1.7 Exercise 7

Exercise 8 is as same as exercise 7 so I use same source code for both.

Or can be found at GitHub Schematic Ex7, Proteus, Source Code Ex 7

1.7.1 Report 1

```
#define NUM_TIMERS 3
int timer_counter[NUM_TIMERS] = {0};
int timer_flag[NUM_TIMERS] = {0};
int TIMER_CYCLE = 10;

void setTimer(int index, int duration) {
  timer_counter[index] = duration / TIMER_CYCLE ;
  timer_flag[index] = 0;
}

void timer_run() {
  for (int i = 0; i < NUM_TIMERS; i++){
    if(timer_counter[i] > 0) {
      timer_counter[i]--;
      if(timer_counter[i] == 0) timer_flag[i] = 1;
    }
  }
}

setTimer(0, 1000); //set time counter for clock
setTimer(1, 250); //set time counter for 4 digits
setTimer(2, 1000); //set time counter for dots
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    if (timer_flag[1] == 1) // 4 digits
    {
        update7SEG(index_led++);
        if (index_led >= 4) index_led = 0;
        setTimer(1, 250);
    }

    if (timer_flag[0] == 1) //clock
    {
        second++;
        if ( second >= 60) {
            second = 0;
            minute ++;
        }
        if( minute >= 60) {
```

```
43              minute = 0;
44              hour ++;
45          }
46          if( hour >=24) {
47              hour = 0;
48          }
49          updateClockBuffer();
50          setTimer(0, 1000);
51      }
52
53      if (timer_flag[2] == 1) // 2 dot toggle
54      {
55          led_red();
56          setTimer(2, 1000);
57      }
58
59
60      /* USER CODE BEGIN 3 */
61    }
62    /* USER CODE END 3 */
```

1.8 Exercise 8

Exercise 8 is as same as exercise 7 so I use same source code for both.

Or can be found at GitHub Schematic Ex8, Proteus, Source Code Ex 8

1.8.1 Report 1

```
#define NUM_TIMERS 3
int timer_counter[NUM_TIMERS] = {0};
int timer_flag[NUM_TIMERS] = {0};
int TIMER_CYCLE = 10;

void setTimer(int index, int duration) {
  timer_counter[index] = duration / TIMER_CYCLE ;
  timer_flag[index] = 0;
}

void timer_run() {
  for (int i = 0; i < NUM_TIMERS; i++){
    if(timer_counter[i] > 0) {
      timer_counter[i]--;
      if(timer_counter[i] == 0) timer_flag[i] = 1;
    }
  }
}

setTimer(0, 1000); //set time counter for clock
setTimer(1, 250); //set time counter for 4 digits
setTimer(2, 1000); //set time counter for dots
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    if (timer_flag[1] == 1) // 4 digits
    {
        update7SEG(index_led++);
        if (index_led >= 4) index_led = 0;
        setTimer(1, 250);
    }

    if (timer_flag[0] == 1) //clock
    {
        second++;
        if ( second >= 60) {
            second = 0;
            minute ++;
        }
        if( minute >= 60) {
```

```
43              minute = 0;
44              hour ++;
45          }
46          if( hour >=24) {
47              hour = 0;
48          }
49          updateClockBuffer();
50          setTimer(0, 1000);
51      }
52
53      if (timer_flag[2] == 1) // 2 dot toggle
54      {
55          led_red();
56          setTimer(2, 1000);
57      }
58      /* USER CODE BEGIN 3 */
59  }
60  /* USER CODE END 3 */
```

1.9 Exercise 9

Exercise 9 is almost as same as exercise 10 so I use same source code for both.

Or can be found at GitHub Schematic Ex9, Proteus, Source Code Ex 9

1.9.1 Report 1

The schematic for exercise 9:

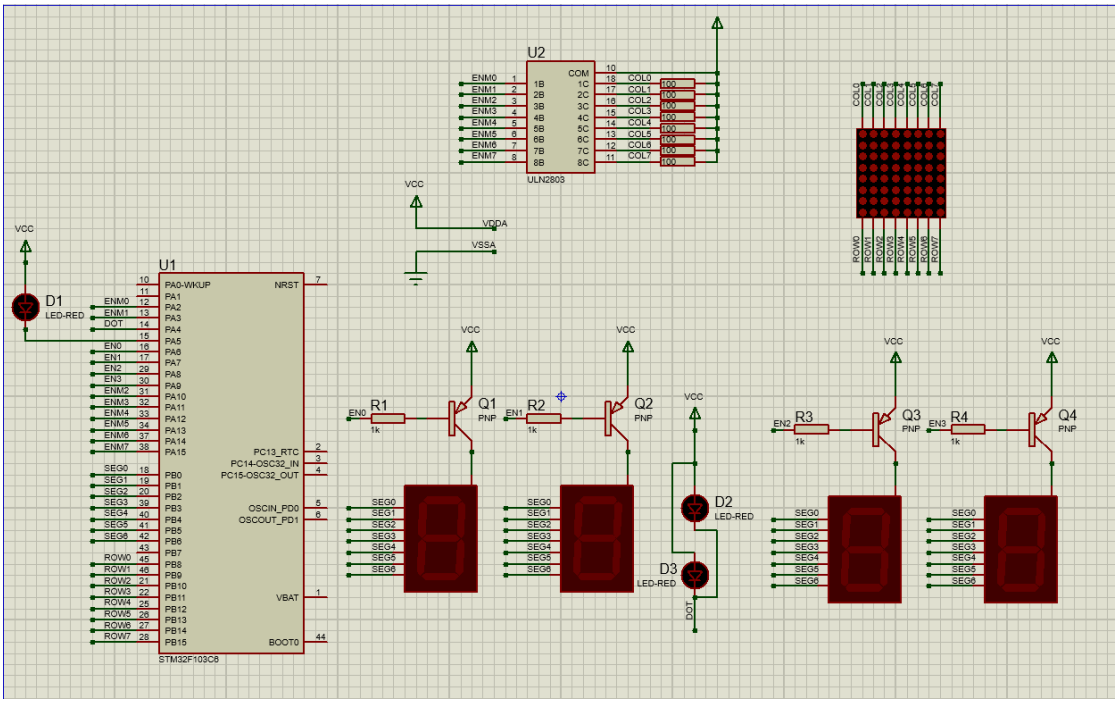

Figure 6: *The schematic for the exercise 9.*

```
#define NUM_TIMERS 5
int timer_counter[NUM_TIMERS] = {0};
int timer_flag[NUM_TIMERS] = {0};
int TIMER_CYCLE = 10;

void setTimer(int index, int duration) {
  timer_counter[index] = duration / TIMER_CYCLE ;
  timer_flag[index] = 0;
}

void timer_run() {
  for (int i = 0; i < NUM_TIMERS; i++){
    if(timer_counter[i] > 0) {
      timer_counter[i]--;
      if(timer_counter[i] == 0) timer_flag[i] = 1;
    }
```

```
  }
}

void resetCol() //turn off all col
{
  HAL_GPIO_WritePin(ENM0_GPIO_Port, ENM0_Pin, SET);
  HAL_GPIO_WritePin(ENM1_GPIO_Port, ENM1_Pin, SET);
  HAL_GPIO_WritePin(ENM2_GPIO_Port, ENM2_Pin, SET);
  HAL_GPIO_WritePin(ENM3_GPIO_Port, ENM3_Pin, SET);
  HAL_GPIO_WritePin(ENM4_GPIO_Port, ENM4_Pin, SET);
  HAL_GPIO_WritePin(ENM5_GPIO_Port, ENM5_Pin, SET);
  HAL_GPIO_WritePin(ENM6_GPIO_Port, ENM6_Pin, SET);
  HAL_GPIO_WritePin(ENM7_GPIO_Port, ENM7_Pin, SET);
}

void resetRow() //turn on all row
{
  HAL_GPIO_WritePin(ROW0_GPIO_Port, ROW0_Pin, RESET);
  HAL_GPIO_WritePin(ROW1_GPIO_Port, ROW1_Pin, RESET);
  HAL_GPIO_WritePin(ROW2_GPIO_Port, ROW2_Pin, RESET);
  HAL_GPIO_WritePin(ROW3_GPIO_Port, ROW3_Pin, RESET);
  HAL_GPIO_WritePin(ROW4_GPIO_Port, ROW4_Pin, RESET);
  HAL_GPIO_WritePin(ROW5_GPIO_Port, ROW5_Pin, RESET);
  HAL_GPIO_WritePin(ROW6_GPIO_Port, ROW6_Pin, RESET);
  HAL_GPIO_WritePin(ROW7_GPIO_Port, ROW7_Pin, RESET);
}

void displayRow(uint8_t hex_value){
  resetRow();
  //int binary_value = bitset<8>(hex_value);
  for (int i = 0; i < 8; i++){
  int bit_value = (hex_value >> i) & 1;
  //int bit_value = binary_value % 10;
  //binary_value = binary_value / 10;
    if (bit_value <= 0){
      if (i == 0){
          HAL_GPIO_WritePin(ROW0_GPIO_Port, ROW0_Pin, SET);
      }
      if (i == 1){
          HAL_GPIO_WritePin(ROW1_GPIO_Port, ROW1_Pin, SET);
      }
      if (i == 2){
          HAL_GPIO_WritePin(ROW2_GPIO_Port, ROW2_Pin, SET);
      }
      if (i == 3){
                HAL_GPIO_WritePin(ROW3_GPIO_Port, ROW3_Pin, SET);
      }
      if (i == 4){
                HAL_GPIO_WritePin(ROW4_GPIO_Port, ROW4_Pin, SET);
```

```
66        }
67        if (i == 5){
68                  HAL_GPIO_WritePin(ROW5_GPIO_Port, ROW5_Pin, SET);
69        }
70              if (i == 6){
71                  HAL_GPIO_WritePin(ROW6_GPIO_Port, ROW6_Pin, SET);
72              }
73              if (i == 7){
74                  HAL_GPIO_WritePin(ROW7_GPIO_Port, ROW7_Pin, SET);
75              }
76    }
77      }
78  }
79
80  void colSW(int columnSW){
81    switch(columnSW){
82    case 0:
83      HAL_GPIO_WritePin(ENM0_GPIO_Port, ENM0_Pin, RESET);
84      break;
85    case 1:
86      HAL_GPIO_WritePin(ENM1_GPIO_Port, ENM1_Pin, RESET);
87      break;
88    case 2:
89      HAL_GPIO_WritePin(ENM2_GPIO_Port, ENM2_Pin, RESET);
90      break;
91    case 3:
92      HAL_GPIO_WritePin(ENM3_GPIO_Port, ENM3_Pin, RESET);
93      break;
94    case 4:
95      HAL_GPIO_WritePin(ENM4_GPIO_Port, ENM4_Pin, RESET);
96      break;
97    case 5:
98      HAL_GPIO_WritePin(ENM5_GPIO_Port, ENM5_Pin, RESET);
99      break;
100   case 6:
101     HAL_GPIO_WritePin(ENM6_GPIO_Port, ENM6_Pin, RESET);
102     break;
103   case 7:
104     HAL_GPIO_WritePin(ENM7_GPIO_Port, ENM7_Pin, RESET);
105     break;
106   default:
107     //resetCol();
108         break;
109   }
110 }
111
112 const int MAX_LED_MATRIX = 8;
113 int index_led_matrix = 0;
```

```
114 //uint8_t matrix_buffer[8] = {0b00000000 , 0b01111100 , 0b00010010 , 0b00010011 ,
        0b00010011 , 0b00010010 , 0b01111100 , 0b00000000};
115 uint8_t matrix_buffer[8] = {0x00, 0x7C, 0x12, 0x13, 0x13, 0x12, 0x7C, 0x00}; //
        letter A
116
117 void updateLEDMatrix (int index) {
118     resetCol();
119     //resetRow();
120   switch(index) {
121     case 0:
122       colSW(index);
123       displayRow(matrix_buffer[index]);
124         break;
125     case 1:
126       colSW(index);
127       displayRow(matrix_buffer[index]);
128         break;
129     case 2:
130       colSW(index);
131       displayRow(matrix_buffer[index]);
132         break;
133     case 3:
134       colSW(index);
135       displayRow(matrix_buffer[index]);
136         break;
137     case 4:
138       colSW(index);
139       displayRow(matrix_buffer[index]);
140         break;
141     case 5:
142       colSW(index);
143       displayRow(matrix_buffer[index]);
144         break;
145     case 6:
146       colSW(index);
147       displayRow(matrix_buffer[index]);
148         break;
149     case 7:
150       colSW(index);
151       displayRow(matrix_buffer[index]);
152         break;
153     default:
154         break;
155   }
156 }
157
158 void shiftL() //this func is served for ex 10
159 {
160   uint8_t temp = matrix_buffer[0];
```

```
  for (int i = 1; i < 8; i++){
    matrix_buffer[i - 1] = matrix_buffer[i];
  }
  matrix_buffer[7] = temp;
}

//...
//THERE IS STILL CODE HERE
//...

setTimer(0, 1000); //set time counter for clock
setTimer(1, 250); //set time counter for 4 digits
setTimer(2, 1000); //set time counter for dots
setTimer(3, 10); //set time counter for led matrix
setTimer(4, 1000); //set time counter for shift Left,this is for ex 10

  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */
    if (timer_flag[1] == 1) // 4 digits
    {
      update7SEG(index_led++);
      if (index_led >= MAX_LED) index_led = 0;
      setTimer(1, 250);
    }

    if (timer_flag[0] == 1) //clock
    {
      second++;
      if ( second >= 60) {
          second = 0;
          minute ++;
      }
      if( minute >= 60) {
          minute = 0;
          hour ++;
      }
      if( hour >=24) {
          hour = 0;
      }
      updateClockBuffer();
      setTimer(0, 1000);
    }

    if (timer_flag[2] == 1) // 2 dot toggle
    {
      led_red();
      setTimer(2, 1000);
```

```
210     }
211
212     if (timer_flag[3] == 1)
213     {
214       updateLEDMatrix(index_led_matrix++);
215       if (index_led_matrix >= MAX_LED_MATRIX) index_led_matrix = 0;
216       setTimer(3, 10);
217     }
218
219     if (timer_flag[4] == 1) //this is for ex 10
220     {
221       shiftL();
222       setTimer(4, 1000);
223     }
224     /* USER CODE BEGIN 3 */
225   }
226   /* USER CODE END 3 */
227 }
```

1.10 Exercise 10

Exercise 9 is almost as same as exercise 10 so I use same source code for both.

Or can be found at GitHub Schematic Ex10, Proteus, Source Code Ex 10

1.10.1 Report 1

```
void shiftL() //this func is served for ex 10
{
  uint8_t temp = matrix_buffer[0];
  for (int i = 1; i < 8; i++){
    matrix_buffer[i - 1] = matrix_buffer[i];
  }
  matrix_buffer[7] = temp;
}

//INIT TIME COUNTER FOR SHIFT LEFT
setTimer(4, 1000);
//THIS IN WHILE LOOP
if (timer_flag[4] == 1) //this is for ex 10
{
    shiftL();
    setTimer(4, 1000);
}
```

References