

**VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY**  
**HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY**



**REPORT**

**LAB 2**

**Class: Microprocessors-Microcontrollers – CC01**

**Lecture: NGUYỄN THIÊN ÂN**

No.	Full Name	ID Student
1.	Nguyễn Huy Tài	2110513

*Ho Chi Minh City, October 1<sup>st</sup> 2024*

# Contents

<b>1</b>	<b>Exercise 1</b>	<b>2</b>
1.1	Report 1 . . . . .	2
1.2	Report 2 . . . . .	2
<b>2</b>	<b>Exercise 2</b>	<b>5</b>
2.1	Report 1 . . . . .	5
2.2	Report 2 . . . . .	5
2.3	Answer the question . . . . .	5
<b>3</b>	<b>Exercise 3</b>	<b>6</b>
3.1	Report 1 . . . . .	6
3.2	Report 2 . . . . .	6
<b>4</b>	<b>Exercise 4</b>	<b>6</b>
4.1	Report 1 . . . . .	6
<b>5</b>	<b>Exercise 5</b>	<b>7</b>
5.1	Report 1 . . . . .	7
<b>6</b>	<b>Exercise 6</b>	<b>7</b>
6.1	Report 1 . . . . .	7
6.2	Report 2 . . . . .	8
6.3	Report 3 . . . . .	8
<b>7</b>	<b>Exercise 7</b>	<b>8</b>
7.1	Report 1 . . . . .	8
<b>8</b>	<b>Exercise 8</b>	<b>9</b>
8.1	Report 1 . . . . .	9
<b>9</b>	<b>Exercise 9</b>	<b>11</b>
9.1	Report 1 . . . . .	11
9.2	Report 2 . . . . .	11
<b>10</b>	<b>Exercise 10</b>	<b>12</b>
10.1	Report 1 . . . . .	12
10.2	Report 2 . . . . .	13
<b>11</b>	<b>Source</b>	<b>13</b>

# 1 Exercise 1

## 1.1 Report 1

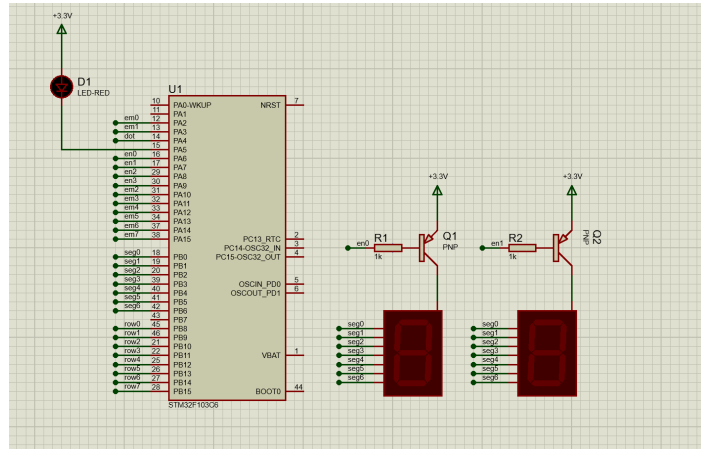


Figure 1: *The schematic of exercise 1*

## 1.2 Report 2

```
1 void display7seg(int nth, int value) {
2
3     if (nth == 1)
4     {
5         HAL_GPIO_WritePin(en0_GPIO_Port, en0_Pin,
6         GPIO_PIN_SET);
7         HAL_GPIO_WritePin(en1_GPIO_Port, en1_Pin,
8         GPIO_PIN_RESET);
9         HAL_GPIO_WritePin(en2_GPIO_Port, en2_Pin,
10        GPIO_PIN_SET);
11        HAL_GPIO_WritePin(en3_GPIO_Port, en3_Pin,
12        GPIO_PIN_SET);
13    }
14    else if (nth == 0)
15    {
16        HAL_GPIO_WritePin(en0_GPIO_Port, en0_Pin,
17        GPIO_PIN_RESET);
18        HAL_GPIO_WritePin(en1_GPIO_Port, en1_Pin,
19        GPIO_PIN_SET);
20        HAL_GPIO_WritePin(en2_GPIO_Port, en2_Pin,
21        GPIO_PIN_SET);
22        HAL_GPIO_WritePin(en3_GPIO_Port, en3_Pin,
23        GPIO_PIN_SET);
24    }
25    else if (nth == 2)
26    {
27        HAL_GPIO_WritePin(en0_GPIO_Port, en0_Pin,
28        GPIO_PIN_SET);
29    }
30 }
```

```

20     HAL_GPIO_WritePin(en1_GPIO_Port, en1_Pin,
    GPIO_PIN_SET);
21     HAL_GPIO_WritePin(en2_GPIO_Port, en2_Pin,
    GPIO_PIN_RESET);
22     HAL_GPIO_WritePin(en3_GPIO_Port, en3_Pin,
    GPIO_PIN_SET);
23 }
24 else if (nth == 3)
25 {
26     HAL_GPIO_WritePin(en0_GPIO_Port, en0_Pin,
    GPIO_PIN_SET);
27     HAL_GPIO_WritePin(en1_GPIO_Port, en1_Pin,
    GPIO_PIN_SET);
28     HAL_GPIO_WritePin(en2_GPIO_Port, en2_Pin,
    GPIO_PIN_SET);
29     HAL_GPIO_WritePin(en3_GPIO_Port, en3_Pin,
    GPIO_PIN_RESET);
30 }
31
32
33     uint8_t segments = segmentMap[value]; // Get segment
    pattern for the value
34
35     // Assuming 7-segment pins are connected to GPIOB Pins
    0-6
36     HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, (segments & 0x01)
    ? GPIO_PIN_SET : GPIO_PIN_RESET);
37     HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1, (segments & 0x02)
    ? GPIO_PIN_SET : GPIO_PIN_RESET);
38     HAL_GPIO_WritePin(GPIOB, GPIO_PIN_2, (segments & 0x04)
    ? GPIO_PIN_SET : GPIO_PIN_RESET);
39     HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3, (segments & 0x08)
    ? GPIO_PIN_SET : GPIO_PIN_RESET);
40     HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, (segments & 0x10)
    ? GPIO_PIN_SET : GPIO_PIN_RESET);
41     HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, (segments & 0x20)
    ? GPIO_PIN_SET : GPIO_PIN_RESET);
42     HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, (segments & 0x40)
    ? GPIO_PIN_SET : GPIO_PIN_RESET);
43 }
44
45 void clearLED() {
46     // Turn off all segments (assuming GPIOB pins 0-6
    control the segments)
47     HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0 | GPIO_PIN_1 |
    GPIO_PIN_2 | GPIO_PIN_3 |
48         GPIO_PIN_4 | GPIO_PIN_5 | GPIO_PIN_6,
    GPIO_PIN_RESET);

```

49 }

Program 1: Function code in exercise1

```
1 int counter = 100, nth = 0;
2 int buffer[2] = {1, 2};
3
4 void HAL_TIM_PeriodElapsedCallback (TIM_HandleTypeDef *htim
   ) {
5     counter--;
6     if (counter == 50 || counter == 0) {
7         if (counter == 0) {
8             counter = 100;
9             HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5); //
10            Toggle GPIOA PIN 5
11        }
12        // Alternate between 0 and 1 for 'nth'
13        nth = (nth == 0) ? 1 : 0;
14
15        clearLED(); // Clear current display
16
17        // Display the value on the 7-segment display
18        display7seg(nth, buffer[nth]);
19    }
```

Program 2: Source code in the **HAL\_TIM\_PeriodElapsedCallback** function

## 2 Exercise 2

### 2.1 Report 1

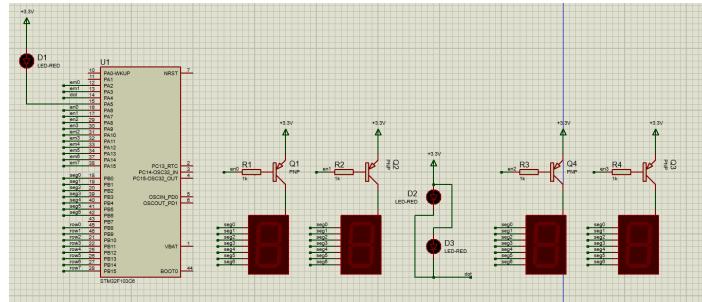


Figure 2: *The schematic of exercise 2*

### 2.2 Report 2

int counter

```
1
2 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
3 {
4     counter--;
5     if (counter == 50 || counter == 0) {
6         if (counter == 0) {
7             counter = 100;
8             HAL_GPIO_TogglePin(GPIOA, led_red_Pin | dot_Pin);
9         }
10
11         clearLED();
12         display7seg(idx, buffer[idx]);
13
14         idx = (idx + 1) % 4;
15     }
16 }
```

Program 3: Source code in the **HAL\_TIM\_PeriodElapsedCallback** function

### 2.3 Answer the question

The frequency of the clock is 2 Hz

## 3 Exercise 3

### 3.1 Report 1

```
1 void update7SEG(int index)
2 {
3     int led_buffer [4] = {1 , 2 , 3 , 4};
4     clearEnableVsLED();
5     display7seg(index, led_buffer[index]);
6 }
```

Program 4: Source code in update7SEG(int index) function

### 3.2 Report 2

```
1 int counter = 100, idx = 0;
2
3 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
4 {
5     counter--;
6
7     if (counter == 50 || counter == 0) {
8         if (counter == 0) {
9             counter = 100;
10            HAL_GPIO_TogglePin(GPIOA, led_red_Pin | dot_Pin);
11        }
12        // clearLED();
13        update7SEG(idx);
14
15        idx = (idx + 1) % 4;
16    }
17 }
```

Program 5: Source code in the **HAL\_TIM\_PeriodElapsedCallback** function

## 4 Exercise 4

### 4.1 Report 1

```
1 /*
2  * inorder to display 1hz for all 4 leds, which means each
3  * leds should bright within 250ms
4  */
5
6 int counter = 50, idx = 0;
7
8 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
9 {
10     counter--;
```

```

8
9  if (counter == 25 || counter == 0) {
10     if (counter == 0) {
11         counter = 50;
12         HAL_GPIO_TogglePin(GPIOA, led_red_Pin | dot_Pin);
13     }
14
15     //     clearLED();
16
17     idx++;
18
19     if (idx >=4)
20     {
21         idx = 0;
22         update7SEG(idx);
23     }
24
25     else update7SEG(idx);
26 }
27 }

```

Program 6: Source code in the **HAL\_TIM\_PeriodElapsedCallback** function

## 5 Exercise 5

### 5.1 Report 1

```

1 void updateClockBuffer(int *clock_buffer, int hour, int min
  , int second) {
2     clock_buffer[0] = hour / 10;
3     clock_buffer[1] = hour % 10;
4     clock_buffer[2] = min / 10;
5     clock_buffer[3] = min % 10;
6 }

```

Program 7: Source code in the **updateClockBuffer** function

## 6 Exercise 6

### 6.1 Report 1

If line 1 of the code is missed, what happens after that and why?

If line 1 of the code is missed, the value `timer0_flag` is kept at 0 and can not be set to 1, so the LED will not blink



## 6.2 Report 2

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

If line 1 of the code is changed to `setTimer0(1)`, the LED will not blink, because if `duration = 1`, we get `timer0_counter = 0` (since `timer0_counter` is of type `int`), then when executing `timer_run()`, the value of `timer0_counter` can not satisfy the if condition, thus `timer0_flag` is kept at 0 and can not be set to 1

## 6.3 Report 3

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

If line 1 of the code is changed to `setTimer0(10)`, we get `timer0_counter = 1`, this value satisfy the if condition in `timer_run()` and the `timer0_flag` is set to 1 right away, so the LED will be invoked and start blinking properly.

## 7 Exercise 7

### 7.1 Report 1

```
1 int index = 0;
2 int hour = 12, min = 38, sec = 55;
3 int TIMEstate = 0;
4 int index_matrix = 0;
5 int count_dot = 0;
6 setTimer1(250);
7 int bufferClock[4] = {0,0,0,0};
8 updateClockBuffer(bufferClock, hour, min, sec);
9 display7seg(0, bufferClock[0]);
10 index++;
11
12 // flag[1] is trigger for second
13 // count_dot is trigger for dot_led
14
15 while (1)
16 {
17     if (count_dot == 2)
18     {
19         // LED & DOT
20         HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_4);
21         HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
22         count_dot = 0;
23     }
24
25     switch(TIMEstate)
26     {
```

```

27     case 0:
28         sec++;
29         setTimer0(1000);
30         TIMEstate = 1;
31         break;
32     case 1:
33         if (flags[0] == 1)
34         {
35             if (sec >= 60)
36             {
37                 sec = 0;
38                 min++;
39             }
40             if (min >= 60)
41             {
42                 min = 0;
43                 hour++;
44             }
45             if (hour >= 24)
46             {
47                 hour = 0;
48             }
49             TIMEstate = 0;
50         }
51         break;
52     }
53
54     updateClockBuffer(bufferClock, hour, min, sec);
55
56     if (index >= 4) index = 0;
57 }

```

Program 8: Source code in the while loop

## 8 Exercise 8

### 8.1 Report 1

```

1  int index = 0;
2  int hour = 12, min = 38, sec = 55;
3  int TIMEstate = 0;
4  int index_matrix = 0;
5  int count_dot = 0;
6  setTimer1(250);
7  int bufferClock[4] = {0,0,0,0};
8  updateClockBuffer(bufferClock, hour, min, sec);
9  display7seg(0, bufferClock[0]);
10 index++;
11

```

```

12 // flag[0] is trigger for second
13 // count_dot is trigger for dot_led
14 // flag[1] is trigger for scanning
15
16 while (1)
17 {
18     if (count_dot == 2)
19     {
20         // LED & DOT
21         HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_4);
22         HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
23         count_dot = 0;
24     }
25
26     if (flags[1] == 1)
27     {
28         clearLED();
29         display7seg(index, bufferClock[index]);
30         index++;
31         count_dot++;
32         setTimer1(250);
33     }
34
35     switch(TIMEstate)
36     {
37         case 0:
38             sec++;
39             setTimer0(1000);
40             TIMEstate = 1;
41             break;
42         case 1:
43             if (flags[0] == 1)
44             {
45                 if (sec >= 60)
46                 {
47                     sec = 0;
48                     min++;
49                 }
50                 if (min >= 60)
51                 {
52                     min = 0;
53                     hour++;
54                 }
55                 if (hour >= 24)
56                 {
57                     hour = 0;
58                 }
59                 TIMEstate = 0;
60             }

```

```

61         break;
62     }
63
64     updateClockBuffer(bufferClock, hour, min, sec);
65
66     if (index >= 4) index = 0;
67 }

```

Program 9: Source code in the while loop

## 9 Exercise 9

### 9.1 Report 1

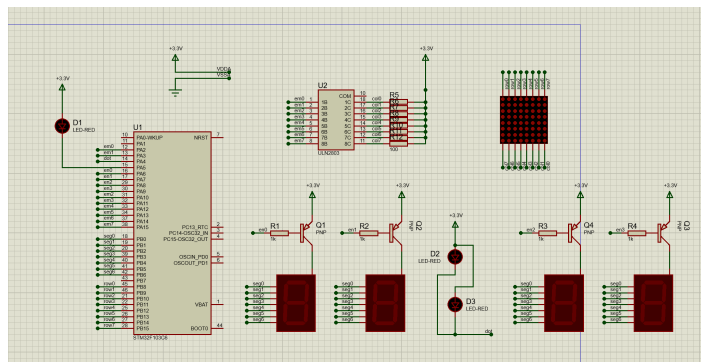


Figure 3: *The schematic of exercise 9*

### 9.2 Report 2

```

1 void setRow(int row)
2 {
3     HAL_GPIO_WritePin(GPIOB, row0_Pin, (row == 0)?
4     GPIO_PIN_SET : GPIO_PIN_RESET);
5     HAL_GPIO_WritePin(GPIOB, row1_Pin, (row == 1)?
6     GPIO_PIN_SET : GPIO_PIN_RESET);
7     HAL_GPIO_WritePin(GPIOB, row2_Pin, (row == 2)?
8     GPIO_PIN_SET : GPIO_PIN_RESET);
9     HAL_GPIO_WritePin(GPIOB, row3_Pin, (row == 3)?
10    GPIO_PIN_SET : GPIO_PIN_RESET);
11    HAL_GPIO_WritePin(GPIOB, row4_Pin, (row == 4)?
12    GPIO_PIN_SET : GPIO_PIN_RESET);
13    HAL_GPIO_WritePin(GPIOB, row5_Pin, (row == 5)?
14    GPIO_PIN_SET : GPIO_PIN_RESET);
15    HAL_GPIO_WritePin(GPIOB, row6_Pin, (row == 6)?
16    GPIO_PIN_SET : GPIO_PIN_RESET);
17    HAL_GPIO_WritePin(GPIOB, row7_Pin, (row == 7)?
18    GPIO_PIN_SET : GPIO_PIN_RESET);
19 }

```

```

12
13 void setColumn(int value)
14 {
15     HAL_GPIO_WritePin(GPIOA, em0_Pin, (value & 0x01) ?
GPIO_PIN_SET : GPIO_PIN_RESET);
16     HAL_GPIO_WritePin(GPIOA, em1_Pin, (value & 0x02) ?
GPIO_PIN_SET : GPIO_PIN_RESET);
17     HAL_GPIO_WritePin(GPIOA, em2_Pin, (value & 0x04) ?
GPIO_PIN_SET : GPIO_PIN_RESET);
18     HAL_GPIO_WritePin(GPIOA, em3_Pin, (value & 0x08) ?
GPIO_PIN_SET : GPIO_PIN_RESET);
19     HAL_GPIO_WritePin(GPIOA, em4_Pin, (value & 0x10) ?
GPIO_PIN_SET : GPIO_PIN_RESET);
20     HAL_GPIO_WritePin(GPIOA, em5_Pin, (value & 0x20) ?
GPIO_PIN_SET : GPIO_PIN_RESET);
21     HAL_GPIO_WritePin(GPIOA, em6_Pin, (value & 0x40) ?
GPIO_PIN_SET : GPIO_PIN_RESET);
22     HAL_GPIO_WritePin(GPIOA, em7_Pin, (value & 0x80) ?
GPIO_PIN_SET : GPIO_PIN_RESET);
23 }
24
25 void updateLEDMatrix(int index)
26 {
27     setRow(index);
28     setColumn(matrix_buffer[index]);
29 }

```

Program 10: Source code of functions matrixled in functions.h

```

1 while (1)
2 {
3     if (flags[3] == 1)
4     {
5         updateLEDMatrix(index_matrix);
6         index_matrix ++;
7         setTimer3(20);
8     }
9     if (index_matrix >= 8 )
10    {
11        index_matrix = 0;
12    }
13 }

```

Program 11: Source code in the while loop

## 10 Exercise 10

### 10.1 Report 1

```

1 void shiftLeft(uint8_t matrix_buffer[8])
2 {
3     for (int i = 0; i < 8; i++) {
4         uint8_t leftBit = (matrix_buffer[i] & 0x80) >> 7;
5         matrix_buffer[i] = (matrix_buffer[i] << 1) |
leftBit;
6     }
7 }

```

Program 12: **shiftleft** function in functions.h

```

1 while (1)
2 {
3     if (flags[3] == 1)
4     {
5         updateLEDMatrix(index_matrix);
6         index_matrix++;
7         setTimer3(20);
8     }
9     if (index_matrix >= 8 )
10    {
11        shiftLeft(matrix_buffer);
12        index_matrix = 0;
13    }
14 }

```

Program 13: Source code in the while loop

## 10.2 Report 2

I put make the shift left by using concatenation method and put in the if condition

## 11 Source

You can find the source code on my GitHub repository: **My GitHub Repository**.