

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



REPORT
LAB 2

Class: Microprocessors - Microcontrollers – CC04

Lecturer: PHAN VĂN SỸ

No.	Full Name	ID Student
1.	Lê Mạnh Quân	2352993

Ho Chi Minh City, September 2025

Contents

1	Exercise 1	2
1.1	Report 1	2
1.2	Report 2	2
1.3	Short question	5
2	Exercise 2	6
2.1	Report 1	6
2.2	Report 2	6
2.3	Short question	7
3	Exercise 3	8
3.1	Report 1	8
3.2	Report 2	8
4	Exercise 4	9
4.1	Report 1	9
5	Exercise 5	10
5.1	Report	10
6	Exercise 6	10
6.1	Report 1	10
6.2	Report 2	10
6.3	Report 3	10
7	Exercise 7	10
7.1	Report	10
8	Exercise 8	11
8.1	Report 1	11
9	Exercise 9	13
10	Exercise 10	14
11	Source	15

20 }

Listing 1: Source code in the HAL TIM PeriodElapsedCallback function

```
1 void clear7SEG() {
2 HAL_GPIO_WritePin(a0_GPIO_Port, a0_Pin, GPIO_PIN_SET);
3 HAL_GPIO_WritePin(a1_GPIO_Port, a1_Pin, GPIO_PIN_SET);
4 HAL_GPIO_WritePin(a2_GPIO_Port, a2_Pin, GPIO_PIN_SET);
5 HAL_GPIO_WritePin(a3_GPIO_Port, a3_Pin, GPIO_PIN_SET);
6 HAL_GPIO_WritePin(a4_GPIO_Port, a4_Pin, GPIO_PIN_SET);
7 HAL_GPIO_WritePin(a5_GPIO_Port, a5_Pin, GPIO_PIN_SET);
8 HAL_GPIO_WritePin(a6_GPIO_Port, a6_Pin, GPIO_PIN_SET);
9 }
10
11 void display7SEG (int num) {
12     clear7SEG();
13     switch (num) {
14         case 0: HAL_GPIO_WritePin(a0_GPIO_Port, a0_Pin,
15                                 GPIO_PIN_RESET);
16                 HAL_GPIO_WritePin(a1_GPIO_Port, a1_Pin,
17                                 GPIO_PIN_RESET);
18                 HAL_GPIO_WritePin(a2_GPIO_Port, a2_Pin,
19                                 GPIO_PIN_RESET);
20                 HAL_GPIO_WritePin(a3_GPIO_Port, a3_Pin,
21                                 GPIO_PIN_RESET);
22                 HAL_GPIO_WritePin(a4_GPIO_Port, a4_Pin,
23                                 GPIO_PIN_RESET);
24                 HAL_GPIO_WritePin(a5_GPIO_Port, a5_Pin,
25                                 GPIO_PIN_RESET);
26                 break;
27         case 1: HAL_GPIO_WritePin(a1_GPIO_Port, a1_Pin,
28                                 GPIO_PIN_RESET);
29                 HAL_GPIO_WritePin(a2_GPIO_Port, a2_Pin,
30                                 GPIO_PIN_RESET);
31                 break;
32         case 2: HAL_GPIO_WritePin(a0_GPIO_Port, a0_Pin,
33                                 GPIO_PIN_RESET);
34                 HAL_GPIO_WritePin(a1_GPIO_Port, a1_Pin,
35                                 GPIO_PIN_RESET);
36                 HAL_GPIO_WritePin(a3_GPIO_Port, a3_Pin,
37                                 GPIO_PIN_RESET);
38                 HAL_GPIO_WritePin(a4_GPIO_Port, a4_Pin,
39                                 GPIO_PIN_RESET);
40                 HAL_GPIO_WritePin(a6_GPIO_Port, a6_Pin,
41                                 GPIO_PIN_RESET);
42                 break;
43         case 3: HAL_GPIO_WritePin(a0_GPIO_Port, a0_Pin,
44                                 GPIO_PIN_RESET);
45                 HAL_GPIO_WritePin(a1_GPIO_Port, a1_Pin,
46                                 GPIO_PIN_RESET);
47                 HAL_GPIO_WritePin(a2_GPIO_Port, a2_Pin,
48                                 GPIO_PIN_RESET);
49                 HAL_GPIO_WritePin(a3_GPIO_Port, a3_Pin,
50                                 GPIO_PIN_RESET);
51                 HAL_GPIO_WritePin(a6_GPIO_Port, a6_Pin,
52                                 GPIO_PIN_RESET);
```

```

35         break;
36     case 4: HAL_GPIO_WritePin(a1_GPIO_Port,a1_Pin,
37         GPIO_PIN_RESET);
38         HAL_GPIO_WritePin(a2_GPIO_Port,a2_Pin,
39         GPIO_PIN_RESET);
40         HAL_GPIO_WritePin(a5_GPIO_Port,a5_Pin,
41         GPIO_PIN_RESET);
42         HAL_GPIO_WritePin(a6_GPIO_Port,a6_Pin,
43         GPIO_PIN_RESET);
44         break;
45     case 5: HAL_GPIO_WritePin(a0_GPIO_Port,a0_Pin,
46         GPIO_PIN_RESET);
47         HAL_GPIO_WritePin(a2_GPIO_Port,a2_Pin,
48         GPIO_PIN_RESET);
49         HAL_GPIO_WritePin(a3_GPIO_Port,a3_Pin,
50         GPIO_PIN_RESET);
51         HAL_GPIO_WritePin(a5_GPIO_Port,a5_Pin,
52         GPIO_PIN_RESET);
53         HAL_GPIO_WritePin(a6_GPIO_Port,a6_Pin,
54         GPIO_PIN_RESET);
55         break;
56     case 6: HAL_GPIO_WritePin(a0_GPIO_Port,a0_Pin,
57         GPIO_PIN_RESET);
58         HAL_GPIO_WritePin(a2_GPIO_Port,a2_Pin,
59         GPIO_PIN_RESET);
60         HAL_GPIO_WritePin(a3_GPIO_Port,a3_Pin,
61         GPIO_PIN_RESET);
62         HAL_GPIO_WritePin(a4_GPIO_Port,a4_Pin,
63         GPIO_PIN_RESET);
64         HAL_GPIO_WritePin(a5_GPIO_Port,a5_Pin,
65         GPIO_PIN_RESET);
66         HAL_GPIO_WritePin(a6_GPIO_Port,a6_Pin,
67         GPIO_PIN_RESET);
68         break;
69     case 7: HAL_GPIO_WritePin(a0_GPIO_Port,a0_Pin,
70         GPIO_PIN_RESET);
71         HAL_GPIO_WritePin(a1_GPIO_Port,a1_Pin,
72         GPIO_PIN_RESET);
73         HAL_GPIO_WritePin(a2_GPIO_Port,a2_Pin,
74         GPIO_PIN_RESET);
75         break;
76     case 8: HAL_GPIO_WritePin(a0_GPIO_Port,a0_Pin,
77         GPIO_PIN_RESET);
78         HAL_GPIO_WritePin(a1_GPIO_Port,a1_Pin,
79         GPIO_PIN_RESET);
80         HAL_GPIO_WritePin(a2_GPIO_Port,a2_Pin,
81         GPIO_PIN_RESET);
82         HAL_GPIO_WritePin(a3_GPIO_Port,a3_Pin,
83         GPIO_PIN_RESET);
84         HAL_GPIO_WritePin(a4_GPIO_Port,a4_Pin,
85         GPIO_PIN_RESET);
86         HAL_GPIO_WritePin(a5_GPIO_Port,a5_Pin,
87         GPIO_PIN_RESET);
88         HAL_GPIO_WritePin(a6_GPIO_Port,a6_Pin,
89         GPIO_PIN_RESET);
90         break;

```

```

66     case 9: HAL_GPIO_WritePin(a0_GPIO_Port,a0_Pin,
67         GPIO_PIN_RESET);
68         HAL_GPIO_WritePin(a1_GPIO_Port,a1_Pin,
69             GPIO_PIN_RESET);
70         HAL_GPIO_WritePin(a2_GPIO_Port,a2_Pin,
71             GPIO_PIN_RESET);
72         HAL_GPIO_WritePin(a3_GPIO_Port,a3_Pin,
73             GPIO_PIN_RESET);
74         HAL_GPIO_WritePin(a5_GPIO_Port,a5_Pin,
75             GPIO_PIN_RESET);
76         HAL_GPIO_WritePin(a6_GPIO_Port,a6_Pin,
77             GPIO_PIN_RESET);
78         break;
79     default: clear7SEG(); break;
80 }
81 }
82
83 void initState() {
84     clear7SEG();
85     HAL_GPIO_WritePin(en0_GPIO_Port, en0_Pin, GPIO_PIN_SET);
86     HAL_GPIO_WritePin(en1_GPIO_Port, en1_Pin, GPIO_PIN_SET);
87     HAL_GPIO_WritePin(en2_GPIO_Port, en2_Pin, GPIO_PIN_SET);
88     HAL_GPIO_WritePin(en3_GPIO_Port, en3_Pin, GPIO_PIN_SET);
89     HAL_GPIO_WritePin(DOT_GPIO_Port, DOT_Pin, GPIO_PIN_RESET);
90     HAL_GPIO_WritePin(led_GPIO_Port, led_Pin, GPIO_PIN_RESET);
91 }
92
93 void clearEN() {
94     HAL_GPIO_WritePin(en0_GPIO_Port, en0_Pin, GPIO_PIN_SET);
95     HAL_GPIO_WritePin(en1_GPIO_Port, en1_Pin, GPIO_PIN_SET);
96     HAL_GPIO_WritePin(en2_GPIO_Port, en2_Pin, GPIO_PIN_SET);
97     HAL_GPIO_WritePin(en3_GPIO_Port, en3_Pin, GPIO_PIN_SET);
98 }

```

Listing 2: Function code in exercise1

1.3 Short question

Short Answer: The frequency of the scanning process is 1Hz

2 Exercise 2

2.1 Report 1

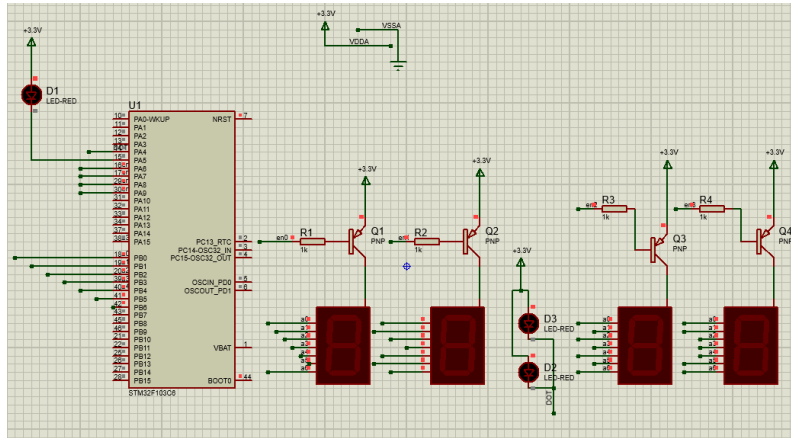


Figure 2: The schematic of exercise 2

2.2 Report 2

```
1 int counter = 100;
2 int idx = 0;
3 int buffer[4] = {1, 2, 3, 0};
4 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
5     counter--;
6
7     if (counter == 50 || counter == 0) {
8         if (counter == 0) {
9
10            counter = 100;
11            HAL_GPIO_TogglePin(GPIOA, led_Pin | DOT_Pin); //
12                nh y LED
13        }
14        clearEN();
15        display7SEG(buffer[idx]);
16        switch (idx) {
17            case 0: HAL_GPIO_WritePin(en0_GPIO_Port, en0_Pin,
18                GPIO_PIN_RESET); break;
19            case 1: HAL_GPIO_WritePin(en1_GPIO_Port, en1_Pin,
20                GPIO_PIN_RESET); break;
21            case 2: HAL_GPIO_WritePin(en2_GPIO_Port, en2_Pin,
22                GPIO_PIN_RESET); break;
23            case 3: HAL_GPIO_WritePin(en3_GPIO_Port, en3_Pin,
24                GPIO_PIN_RESET); break;
25        }
26        idx = (idx + 1) % 4;
27    }
28 }
```

24 }

Listing 3: Source code in the HAL TIM PeriodElapsedCallback function

2.3 Short question

Short Answer: The frequency of the scanning process is 2Hz

3 Exercise 3

3.1 Report 1

```
1 const int MAX_LED = 4;
2 int counter = 100;
3 int index_led = 0;
4 void update7SEG(int index) {
5     int led_buffer[4] = {1, 2, 3, 4};
6     clearEN();
7     switch (index) {
8         case 0:
9             display7SEG(led_buffer[0]);
10            HAL_GPIO_WritePin(en0_GPIO_Port, en0_Pin,
11                               GPIO_PIN_RESET);
12            break;
13        case 1:
14            display7SEG(led_buffer[1]);
15            HAL_GPIO_WritePin(en1_GPIO_Port, en1_Pin,
16                               GPIO_PIN_RESET);
17            break;
18        case 2:
19            display7SEG(led_buffer[2]);
20            HAL_GPIO_WritePin(en2_GPIO_Port, en2_Pin,
21                               GPIO_PIN_RESET);
22            break;
23        case 3:
24            display7SEG(led_buffer[3]);
25            HAL_GPIO_WritePin(en3_GPIO_Port, en3_Pin,
26                               GPIO_PIN_RESET);
27            break;
28        default:
29            break;
30    }
31 }
```

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

3.2 Report 2

```
1 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
2     counter--;
3     if (counter == 50 || counter == 0) {
4         if (counter == 0) {
5             counter = 100;
6             HAL_GPIO_TogglePin(GPIOA, led_Pin | DOT_Pin);
7         }
8         update7SEG(index_led);
9         index_led = (index_led + 1) % MAX_LED;
10    }
11 }
```

Listing 5: Source code in the HAL_TIM_PeriodElapsedCallback function

4 Exercise 4

4.1 Report 1

```
1 int counter = 50;
2 int idx = 0;
3 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
4     counter--;
5     if (counter == 25 || counter == 0) {
6         if (counter == 0) {
7             counter = 50; // Reset sau 0.5s
8             HAL_GPIO_TogglePin(GPIOA, led_Pin | DOT_Pin);
9         }
10        update7SEG(idx);
11        idx = (idx + 1) % 4;
12    }
13 }
```

Listing 6: source code in the `HAL_TIM_PeriodElapsedCallback`

5 Exercise 5

5.1 Report

```
1 void updateClockBuffer(int hour, int minute) {  
2     led_buffer[0] = hour / 10;  
3     led_buffer[1] = hour % 10;  
4     led_buffer[2] = minute / 10;  
5     led_buffer[3] = minute % 10;  
6 }
```

Listing 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 timer0flag is kept at 0 and cannot 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 timer0counter=0 (since timer0counter is of type int), then when executing timerrun(), the value of timer0counter cannot satisfy the if condition, thus timer0flag is kept at 0 and cannot be set to 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?

If line 1 of the code is changed to setTimer0(10), we get timer0counter=1, this value satisfies the if condition in timerrun() and the timer0flag is set to 1 right away, so the LED will be invoked and start blinking properly.

7 Exercise 7

7.1 Report

```
1 int hour = 15, minute = 8, second = 55;  
2     setTimer0(1000);  
3     while (1)  
4     {
```

```

5      /* USER CODE END WHILE */
6
7
8      /* USER CODE BEGIN 3 */
9          if (timer0_flag == 1) {
10              second++;
11              if (second >= 60) {
12                  second = 0;
13                  minute++;
14              }
15              if (minute >= 60) {
16                  minute = 0;
17                  hour++;
18              }
19              if (hour >= 24) {
20                  hour = 0;
21              }
22
23              updateClockBuffer(hour,
24                               minute);
25              setTimer0(1000);
26          }
27      }

```

Listing 8: Source code in the while loop on main function

8 Exercise 8

8.1 Report 1

```

1      int hour = 15, minute = 8, second = 55;
2      int index_led = 0;
3      const int MAX_LED = 4;
4      int led_buffer[4] = {1, 5, 0, 8};
5      int dot_state = 0;
6      setTimer0(1000);
7      setTimer1(250);
8      while (1)
9      {
10         if (timer0_flag == 1) {
11             second++;
12             if (second >= 60) {
13                 second = 0;
14                 minute++;
15             }
16             if (minute >= 60) {
17                 minute = 0;
18                 hour++;
19             }
20             if (hour >= 24) {
21                 hour = 0;
22             }

```

```
23
24         updateClockBuffer(hour,
25             minute);
26         dot_state = !dot_state;
27         HAL_GPIO_WritePin(GPIOA,
28             GPIO_PIN_4, dot_state ?
29             GPIO_PIN_SET :
30             GPIO_PIN_RESET);
31         setTimer0(1000);
32     }
33     if (timer1_flag == 1) {
34         update7SEG(index_led);
35         index_led = (index_led + 1) % MAX_LED;
36         setTimer1(250);
37     }
```

Listing 9: Source code in the the main function

9 Exercise 9

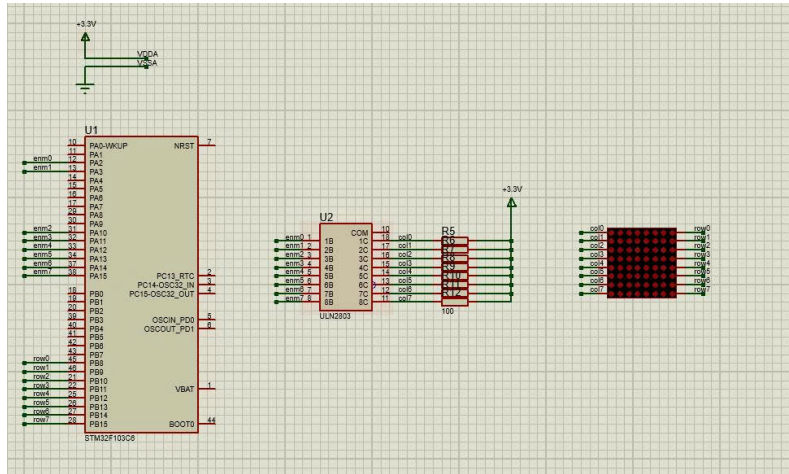


Figure 3: The schematic of exercise 1

```

1  const int MAX_LED_MATRIX = 8;
2  int index_matrix = 0;
3  uint8_t matrix_buffer[8] = {
4      0x18, // 00011000
5      0x24, // 00100100
6      0x42, // 01000010
7      0x42, // 01000010
8      0x7E, // 01111110
9      0x42, // 01000010
10     0x42, // 01000010
11     0x00  // 00000000
12 };
13
14 void setRow(int row) {
15     HAL_GPIO_WritePin(GPIOB, ROW0_Pin, (row == 0) ? GPIO_PIN_SET :
16         GPIO_PIN_RESET);
17     HAL_GPIO_WritePin(GPIOB, ROW1_Pin, (row == 1) ? GPIO_PIN_SET :
18         GPIO_PIN_RESET);
19     HAL_GPIO_WritePin(GPIOB, ROW2_Pin, (row == 2) ? GPIO_PIN_SET :
20         GPIO_PIN_RESET);
21     HAL_GPIO_WritePin(GPIOB, ROW3_Pin, (row == 3) ? GPIO_PIN_SET :
22         GPIO_PIN_RESET);
23     HAL_GPIO_WritePin(GPIOB, ROW4_Pin, (row == 4) ? GPIO_PIN_SET :
24         GPIO_PIN_RESET);
25     HAL_GPIO_WritePin(GPIOB, ROW5_Pin, (row == 5) ? GPIO_PIN_SET :
26         GPIO_PIN_RESET);
27     HAL_GPIO_WritePin(GPIOB, ROW6_Pin, (row == 6) ? GPIO_PIN_SET :
28         GPIO_PIN_RESET);
29     HAL_GPIO_WritePin(GPIOB, ROW7_Pin, (row == 7) ? GPIO_PIN_SET :
30         GPIO_PIN_RESET);
31 }
32
33 void setColumn(int value) {

```

```

26     HAL_GPIO_WritePin(GPIOA, ENM0_Pin, (value & 0x01) ?
        GPIO_PIN_SET : GPIO_PIN_RESET);
27     HAL_GPIO_WritePin(GPIOA, ENM1_Pin, (value & 0x02) ?
        GPIO_PIN_SET : GPIO_PIN_RESET);
28     HAL_GPIO_WritePin(GPIOA, ENM2_Pin, (value & 0x04) ?
        GPIO_PIN_SET : GPIO_PIN_RESET);
29     HAL_GPIO_WritePin(GPIOA, ENM3_Pin, (value & 0x08) ?
        GPIO_PIN_SET : GPIO_PIN_RESET);
30     HAL_GPIO_WritePin(GPIOA, ENM4_Pin, (value & 0x10) ?
        GPIO_PIN_SET : GPIO_PIN_RESET);
31     HAL_GPIO_WritePin(GPIOA, ENM5_Pin, (value & 0x20) ?
        GPIO_PIN_SET : GPIO_PIN_RESET);
32     HAL_GPIO_WritePin(GPIOA, ENM6_Pin, (value & 0x40) ?
        GPIO_PIN_SET : GPIO_PIN_RESET);
33     HAL_GPIO_WritePin(GPIOA, ENM7_Pin, (value & 0x80) ?
        GPIO_PIN_SET : GPIO_PIN_RESET);
34 }
35
36 void updateLEDMatrix(int index) {
37     setRow(index);
38     setColumn(matrix_buffer[index]);
39 }
40
41 }

```

Listing 10: Code in private code

```

1  while (1)
2  {
3      updateLEDMatrix(index_matrix);
4      index_matrix++;
5      HAL_Delay(1);
6      if (index_matrix >= 8) {
7          index_matrix = 0;
8      }
9  }

```

Listing 11: Code in while loop

10 Exercise 10

```

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

```

Listing 12: Code in private code

```

1  while (1)
2  {
3      updateLEDMatrix(index_matrix);
4      index_matrix++;

```

```
5         HAL_Delay(5);
6         if (index_matrix >= 8) {
7             shiftLeft(matrix_buffer);
8             index_matrix = 0;
9         }
10 }
```

Listing 13: Code in while loop

11 Source

GG Drive Link: [My Source Code](#) Github Link: [My Source Code](#)