

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



**MICROPROCESSORS  
MICRO-CONTROLLERS  
CO3009 - CC01**

---

**Report**

**Lab 2 - Timer**

---

Advisors: Lê Trọng Nhân  
Nguyễn Trần Hữu Nguyên

Student: Bùi Phát Lộc - 1752326

HO CHI MINH CITY, Q4 2022



## Contents

<b>1</b>	<b>Software Timer</b>	<b>3</b>
<b>2</b>	<b>Exercise 1</b>	<b>4</b>
<b>3</b>	<b>Exercise 2</b>	<b>7</b>
<b>4</b>	<b>Exercise 3 + 4</b>	<b>10</b>
<b>5</b>	<b>Exercise 5</b>	<b>12</b>
<b>6</b>	<b>Exercise 7 + 8</b>	<b>13</b>



This is the drive link contains all file in Lab 1, including STM32 file and Proteus simulation file: [Github Link](#) and [Back up GG Drive Link](#)

## 1 Software Timer

Except exercise 1, all other exercises use the software timer so this `HAL_TIM_PeriodElapsedCallback()` function is used instead

```
1 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
2 {
3     Run_timer();
4 }
```

Program 1.1: `HAL_TIM_PeriodElapsedCallback()` function

```
1
2 #ifndef INC_SOFTWARE_TIMER_H_
3 #define INC_SOFTWARE_TIMER_H_
4
5 extern int timer_flag[];
6
7 void Set_timer(int duration, int arr_pos);
8 void Run_timer();
9
10 #endif /* INC_SOFTWARE_TIMER_H_ */
```

Program 1.2: `software_timer.h`

```
1 #include "software_timer.h"
2
3 const int timer_arr_size = 18;
4 int timer_counter[18] = {
5     0, 0, 0, 0, 0, 0,
6     0, 0, 0, 0, 0, 0,
7     0, 0, 0, 0, 0, 0};
8 int timer_flag[18] = {
9     0, 0, 0, 0, 0, 0,
10    0, 0, 0, 0, 0, 0,
11    0, 0, 0, 0, 0, 0};
12
13
14 void Set_timer(int duration, int arr_pos)
15 {
16     timer_counter[arr_pos] = duration;
17     timer_flag[arr_pos] = 0;
18 }
19
20 void Run_timer()
21 {
22     int i = 0;
23     while(i < timer_arr_size)
24     {
25         if(timer_counter[i] > 0)
26         {
27             timer_counter[i]--;
28             if(timer_counter[i] <= 0)
29             {
30                 timer_flag[i] = 1;
31             }
32         }
33         i++;
34     }
35 }
```

Program 1.3: software\_timer.c

## 2 Exercise 1

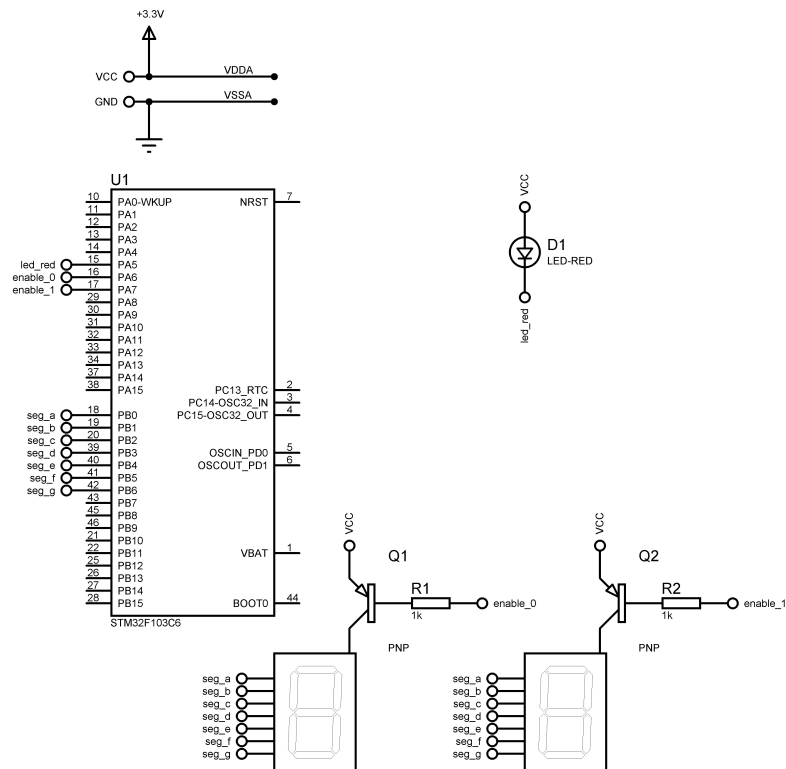


Figure 2.1: [Github Link](#) and [Back up GG Drive Link](#)

```

1 int enable_trigger = 0;
2 int counter = 50;
3 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
4 {
5     counter--;
6     int trigger_num = 0;
7     if(counter <= 0)
8     {
9         counter = 50;
10        HAL_GPIO_TogglePin(led_red_GPIO_Port, led_red_Pin);
11
12        switch (enable_trigger) {
13            case 0:
14                HAL_GPIO_WritePin(enable_0_GPIO_Port, enable_0_Pin, RESET);
15                HAL_GPIO_WritePin(enable_1_GPIO_Port, enable_1_Pin, SET);
16                enable_trigger = 1;
17                trigger_num = 1;
18                break;
19            case 1:

```

```
20     HAL_GPIO_WritePin(enable_1_GPIO_Port, enable_1_Pin, RESET)
21     ;
22     HAL_GPIO_WritePin(enable_0_GPIO_Port, enable_0_Pin, SET);
23     enable_trigger = 0;
24     trigger_num = 2;
25     break;
26     default:
27     break;
28 }
29
30 display7SEG(trigger_num,
31     seg_a_GPIO_Port, seg_a_Pin,
32     seg_b_GPIO_Port, seg_b_Pin,
33     seg_c_GPIO_Port, seg_c_Pin,
34     seg_d_GPIO_Port, seg_d_Pin,
35     seg_e_GPIO_Port, seg_e_Pin,
36     seg_f_GPIO_Port, seg_f_Pin,
37     seg_g_GPIO_Port, seg_g_Pin);
38 }
```

Program 2.1: Ex1 source code

### 3 Exercise 2

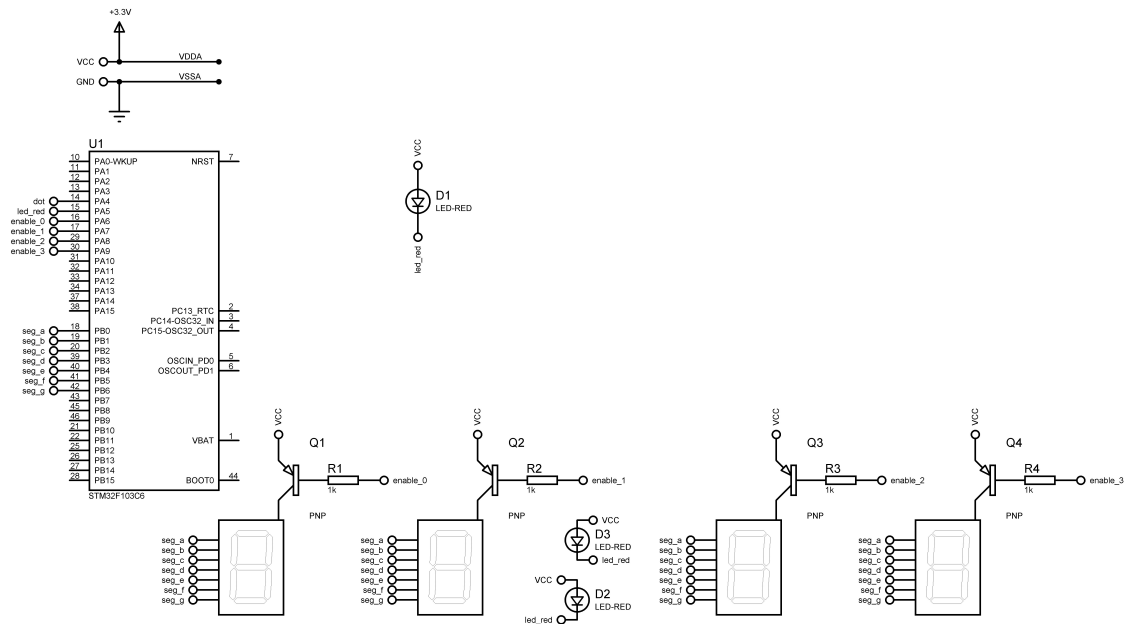


Figure 3.1: [Github Link](#) and [Back up GG Drive Link](#)



```
1 Set_timer(100, 0);
2 Set_timer(50, 1);
3 int enable_trigger = 0;
4 int trigger_num = 0;
5 while (1)
6 {
7     if(timer_flag[0] == 1)
8     {
9         Set_timer(100,0);
10        HAL_GPIO_TogglePin(led_red_GPIO_Port, led_red_Pin);
11    }
12
13    if(timer_flag[1] == 1)
14    {
15        Set_timer(50,1);
16        switch (enable_trigger)
17        {
18            case 0:
19                HAL_GPIO_WritePin(enable_0_GPIO_Port, enable_0_Pin, RESET)
20            ;
21                HAL_GPIO_WritePin(enable_1_GPIO_Port, enable_1_Pin, SET);
22                HAL_GPIO_WritePin(enable_2_GPIO_Port, enable_2_Pin, SET);
23                HAL_GPIO_WritePin(enable_3_GPIO_Port, enable_3_Pin, SET);
24                enable_trigger = 1;
25                trigger_num = 1;
26                break;
27            case 1:
28                HAL_GPIO_WritePin(enable_0_GPIO_Port, enable_0_Pin, SET);
29                HAL_GPIO_WritePin(enable_1_GPIO_Port, enable_1_Pin, RESET)
30            ;
31                HAL_GPIO_WritePin(enable_2_GPIO_Port, enable_2_Pin, SET);
32                HAL_GPIO_WritePin(enable_3_GPIO_Port, enable_3_Pin, SET);
33                enable_trigger = 2;
34                trigger_num = 2;
35                break;
36            case 2:
37                HAL_GPIO_WritePin(enable_0_GPIO_Port, enable_0_Pin, SET);
38                HAL_GPIO_WritePin(enable_1_GPIO_Port, enable_1_Pin, SET);
39                HAL_GPIO_WritePin(enable_2_GPIO_Port, enable_2_Pin, RESET)
40            ;
41                HAL_GPIO_WritePin(enable_3_GPIO_Port, enable_3_Pin, SET);
42                enable_trigger = 3;
43                trigger_num = 3;
44                break;
45            case 3:
46                HAL_GPIO_WritePin(enable_0_GPIO_Port, enable_0_Pin, SET);
47                HAL_GPIO_WritePin(enable_1_GPIO_Port, enable_1_Pin, SET);
48                HAL_GPIO_WritePin(enable_2_GPIO_Port, enable_2_Pin, SET);
49                HAL_GPIO_WritePin(enable_3_GPIO_Port, enable_3_Pin, RESET)
```

```
47     ;
48     enable_trigger = 0;
49     trigger_num = 0;
50     break;
51     default:
52     break;
53 }
54
55 display7SEG(trigger_num,
56     seg_a_GPIO_Port, seg_a_Pin,
57     seg_b_GPIO_Port, seg_b_Pin,
58     seg_c_GPIO_Port, seg_c_Pin,
59     seg_d_GPIO_Port, seg_d_Pin,
60     seg_e_GPIO_Port, seg_e_Pin,
61     seg_f_GPIO_Port, seg_f_Pin,
62     seg_g_GPIO_Port, seg_g_Pin);
63
64 }
65 /* USER CODE END WHILE */
66
67 /* USER CODE BEGIN 3 */
68 }
69 }
```

Program 3.1: main loop

## 4 Exercise 3 + 4

```
1 void update7SEG(int num)
2 {
3     switch (num)
4     {
5         case 0:
6             HAL_GPIO_WritePin(enable_0_GPIO_Port, enable_0_Pin, RESET);
7             HAL_GPIO_WritePin(enable_1_GPIO_Port, enable_1_Pin, SET);
8             HAL_GPIO_WritePin(enable_2_GPIO_Port, enable_2_Pin, SET);
9             HAL_GPIO_WritePin(enable_3_GPIO_Port, enable_3_Pin, SET);
10            break;
11        case 1:
12            HAL_GPIO_WritePin(enable_0_GPIO_Port, enable_0_Pin, SET);
13            HAL_GPIO_WritePin(enable_1_GPIO_Port, enable_1_Pin, RESET);
14            HAL_GPIO_WritePin(enable_2_GPIO_Port, enable_2_Pin, SET);
15            HAL_GPIO_WritePin(enable_3_GPIO_Port, enable_3_Pin, SET);
16            break;
17        case 2:
18            HAL_GPIO_WritePin(enable_0_GPIO_Port, enable_0_Pin, SET);
19            HAL_GPIO_WritePin(enable_1_GPIO_Port, enable_1_Pin, SET);
20            HAL_GPIO_WritePin(enable_2_GPIO_Port, enable_2_Pin, RESET);
21            HAL_GPIO_WritePin(enable_3_GPIO_Port, enable_3_Pin, SET);
22            break;
23        case 3:
24            HAL_GPIO_WritePin(enable_0_GPIO_Port, enable_0_Pin, SET);
25            HAL_GPIO_WritePin(enable_1_GPIO_Port, enable_1_Pin, SET);
26            HAL_GPIO_WritePin(enable_2_GPIO_Port, enable_2_Pin, SET);
27            HAL_GPIO_WritePin(enable_3_GPIO_Port, enable_3_Pin, RESET);
28            break;
29        default:
30            break;
31    }
32 }
```

Program 4.1: **update7SEG()** function

```
1 Set_timer(100, 0);
2 Set_timer(25, 1);
3 const int MAX_LED = 4;
4 int index_led = 0;
5 int led_buffer [4] = {1 , 2 , 3 , 4};
6
7 while (1)
8 {
9     if(index_led >= MAX_LED)
10    {
11        index_led = 0;
12    }
13    if(timer_flag[0] == 1)
14    {
15        Set_timer(100, 0);
16        HAL_GPIO_TogglePin(led_red_GPIO_Port , led_red_Pin);
17    }
18    if(timer_flag[1] == 1)
19    {
20        Set_timer(25, 1);
21        display7SEG(led_buffer[index_led],
22                    seg_a_GPIO_Port, seg_a_Pin,
23                    seg_b_GPIO_Port, seg_b_Pin,
24                    seg_c_GPIO_Port, seg_c_Pin,
25                    seg_d_GPIO_Port, seg_d_Pin,
26                    seg_e_GPIO_Port, seg_e_Pin,
27                    seg_f_GPIO_Port, seg_f_Pin,
28                    seg_g_GPIO_Port, seg_g_Pin);
29        update7SEG(index_led++);
30    }
31    /* USER CODE END WHILE */
32
33    /* USER CODE BEGIN 3 */
34 }
```

Program 4.2: Ex3 + Ex4 source

## 5 Exercise 5

```
1 void updateClockBuffer(int hour, int minute, int* led_buffer)
2 {
3
4     if(hour >= 0 && hour <10)
5     {
6         led_buffer[0] = 0;
7         led_buffer[1] = hour;
8     }
9     else
10    {
11        led_buffer[0] = hour/10;
12        led_buffer[1] = hour%10;
13    }
14
15    if(minute >= 0 && minute <10)
16    {
17        led_buffer[2] = 0;
18        led_buffer[3] = minute;
19    }
20    else
21    {
22        led_buffer[2] = minute/10;
23        led_buffer[3] = minute%10;
24    }
25 }
```

Program 5.1: **updateClockBuffer()** function

## 6 Exercise 7 + 8

```
1  Set_timer(100, 0);
2  Set_timer(50, 1);
3  int hour = 24 , minute = 59 , second = 50;
4  const int MAX_LED = 4;
5  int led_buffer [4] = {1 , 2 , 3 , 4};
6  int index_led = 0;
7  while (1)
8  {
9      if(timer_flag[0] == 1)
10     {
11         Set_timer(100, 0);
12         HAL_GPIO_TogglePin(led_red_GPIO_Port , led_red_Pin);
13         HAL_GPIO_TogglePin(dot_GPIO_Port , dot_Pin);
14         second++;
15         if(second >= 60)
16         {
17             second = 0;
18             minute++;
19         }
20         if(minute >= 60)
21         {
22             minute = 0;
23             hour++;
24         }
25         if(hour >= 24)
26         {
27             hour = 0;
28         }
29     }
30
31     updateClockBuffer(hour , minute , led_buffer);
32
33     if(timer_flag[1] == 1)
34     {
35         Set_timer(50, 1);
36         if(index_led >= MAX_LED)
37         {
38             index_led = 0;
39         }
40         switch (index_led)
41         {
42             case 0:
43                 HAL_GPIO_WritePin(enable_0_GPIO_Port , enable_0_Pin , RESET)
44                 ;
45                 HAL_GPIO_WritePin(enable_1_GPIO_Port , enable_1_Pin , SET);
46                 HAL_GPIO_WritePin(enable_2_GPIO_Port , enable_2_Pin , SET);
47                 HAL_GPIO_WritePin(enable_3_GPIO_Port , enable_3_Pin , SET);
```

```
47         break;
48     case 1:
49         HAL_GPIO_WritePin(enable_0_GPIO_Port, enable_0_Pin, SET);
50         HAL_GPIO_WritePin(enable_1_GPIO_Port, enable_1_Pin, RESET)
51     ;
52         HAL_GPIO_WritePin(enable_2_GPIO_Port, enable_2_Pin, SET);
53         HAL_GPIO_WritePin(enable_3_GPIO_Port, enable_3_Pin, SET);
54         break;
55     case 2:
56         HAL_GPIO_WritePin(enable_0_GPIO_Port, enable_0_Pin, SET);
57         HAL_GPIO_WritePin(enable_1_GPIO_Port, enable_1_Pin, SET);
58         HAL_GPIO_WritePin(enable_2_GPIO_Port, enable_2_Pin, RESET)
59     ;
60         HAL_GPIO_WritePin(enable_3_GPIO_Port, enable_3_Pin, SET);
61         break;
62     case 3:
63         HAL_GPIO_WritePin(enable_0_GPIO_Port, enable_0_Pin, SET);
64         HAL_GPIO_WritePin(enable_1_GPIO_Port, enable_1_Pin, SET);
65         HAL_GPIO_WritePin(enable_2_GPIO_Port, enable_2_Pin, SET);
66         HAL_GPIO_WritePin(enable_3_GPIO_Port, enable_3_Pin, RESET)
67     ;
68         break;
69     default:
70         break;
71 }
72
73 display7SEG(led_buffer[index_led],
74             seg_a_GPIO_Port, seg_a_Pin,
75             seg_b_GPIO_Port, seg_b_Pin,
76             seg_c_GPIO_Port, seg_c_Pin,
77             seg_d_GPIO_Port, seg_d_Pin,
78             seg_e_GPIO_Port, seg_e_Pin,
79             seg_f_GPIO_Port, seg_f_Pin,
80             seg_g_GPIO_Port, seg_g_Pin);
81 index_led++;
82 }
83
84 /* USER CODE END WHILE */
85
86 /* USER CODE BEGIN 3 */
87 }
```

Program 6.1: Ex7 + Ex8 source