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

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

```c
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
  Run_timer();
}
```

Program 1.1: **HAL_TIM_PeriodElapsedCallback()** function

```c

#ifndef INC_SOFTWARE_TIMER_H_
#define INC_SOFTWARE_TIMER_H_

extern int timer_flag[];

void Set_timer(int duration, int arr_pos);
void Run_timer();

#endif /* INC_SOFTWARE_TIMER_H_ */
```

Program 1.2: software_timer.h

```c
#include "software_timer.h"

const int timer_arr_size = 18;
int timer_counter[18] = {
    0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0};
int timer_flag[18] = {
    0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0};


void Set_timer(int duration, int arr_pos)
{
  timer_counter[arr_pos] = duration;
  timer_flag[arr_pos] = 0;
}

void Run_timer()
{
  int i = 0;
  while(i < timer_arr_size)
  {
    if(timer_counter[i] > 0)
    {
      timer_counter[i]--;
      if(timer_counter[i] <= 0)
      {
        timer_flag[i] = 1;
      }
    }
    i++;
  }
}
```

Program 1.3: software_timer.c

2 Exercise 1

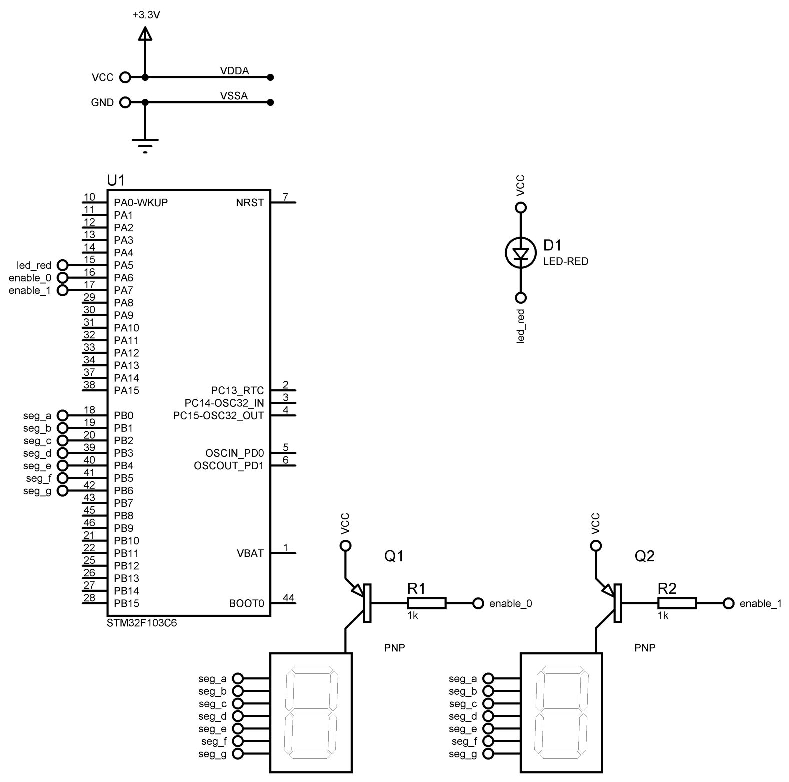

Figure 2.1: Github Link and Back up GG Drive Link

```c
int enable_trigger = 0;
int counter = 50;
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
  counter--;
  int trigger_num = 0;
  if(counter <= 0)
  {
    counter = 50;
    HAL_GPIO_TogglePin(led_red_GPIO_Port, led_red_Pin);

    switch (enable_trigger) {
      case 0:
        HAL_GPIO_WritePin(enable_0_GPIO_Port, enable_0_Pin, RESET);
        HAL_GPIO_WritePin(enable_1_GPIO_Port, enable_1_Pin, SET);
        enable_trigger = 1;
        trigger_num = 1;
        break;
      case 1:
        HAL_GPIO_WritePin(enable_1_GPIO_Port, enable_1_Pin, RESET)
    ;
        HAL_GPIO_WritePin(enable_0_GPIO_Port, enable_0_Pin, SET);
        enable_trigger = 0;
        trigger_num = 2;
        break;
      default:
        break;
    }

    display7SEG(trigger_num,
        seg_a_GPIO_Port, seg_a_Pin,
        seg_b_GPIO_Port, seg_b_Pin,
        seg_c_GPIO_Port, seg_c_Pin,
        seg_d_GPIO_Port, seg_d_Pin,
        seg_e_GPIO_Port, seg_e_Pin,
        seg_f_GPIO_Port, seg_f_Pin,
        seg_g_GPIO_Port, seg_g_Pin);
  }
}
```

Program 2.1: Ex1 source code

3 Exercise 2

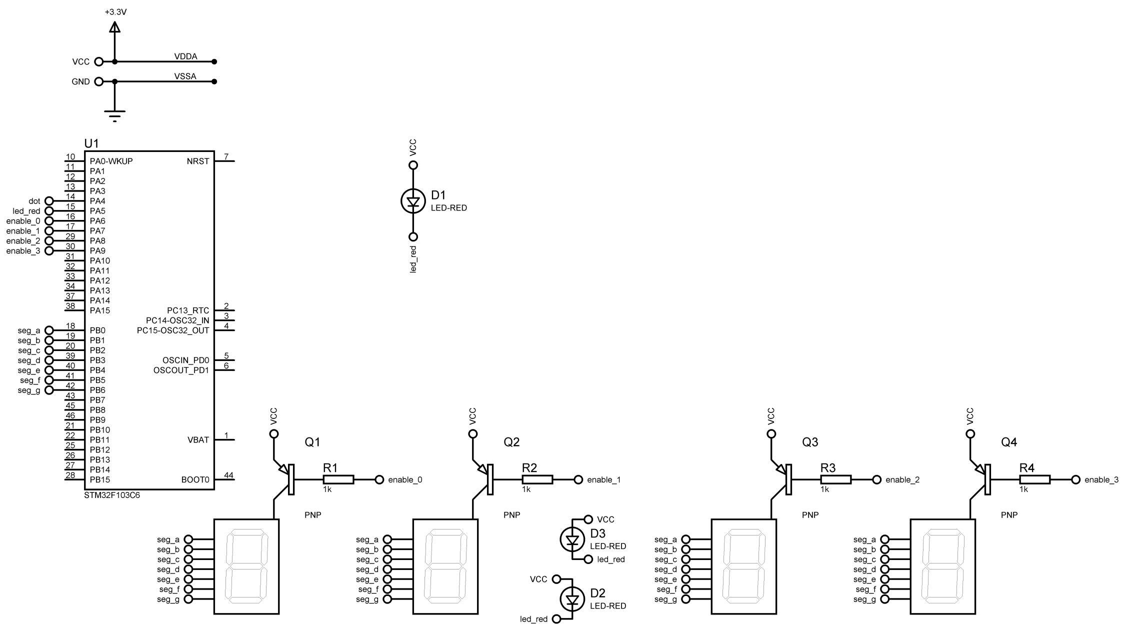

Figure 3.1: Github Link and Back up GG Drive Link

```c
Set_timer(100, 0);
Set_timer(50, 1);
int enable_trigger = 0;
int trigger_num = 0;
while (1)
{
  if(timer_flag[0] == 1)
  {
    Set_timer(100,0);
    HAL_GPIO_TogglePin(led_red_GPIO_Port, led_red_Pin);
  }

  if(timer_flag[1] == 1)
  {
    Set_timer(50,1);
    switch (enable_trigger)
    {
    case 0:
      HAL_GPIO_WritePin(enable_0_GPIO_Port, enable_0_Pin, RESET)
;
      HAL_GPIO_WritePin(enable_1_GPIO_Port, enable_1_Pin, SET);
      HAL_GPIO_WritePin(enable_2_GPIO_Port, enable_2_Pin, SET);
      HAL_GPIO_WritePin(enable_3_GPIO_Port, enable_3_Pin, SET);
      enable_trigger = 1;
      trigger_num = 1;
      break;
    case 1:
      HAL_GPIO_WritePin(enable_0_GPIO_Port, enable_0_Pin, SET);
      HAL_GPIO_WritePin(enable_1_GPIO_Port, enable_1_Pin, RESET)
;
      HAL_GPIO_WritePin(enable_2_GPIO_Port, enable_2_Pin, SET);
      HAL_GPIO_WritePin(enable_3_GPIO_Port, enable_3_Pin, SET);
      enable_trigger = 2;
      trigger_num = 2;
      break;
    case 2:
      HAL_GPIO_WritePin(enable_0_GPIO_Port, enable_0_Pin, SET);
      HAL_GPIO_WritePin(enable_1_GPIO_Port, enable_1_Pin, SET);
      HAL_GPIO_WritePin(enable_2_GPIO_Port, enable_2_Pin, RESET)
;
      HAL_GPIO_WritePin(enable_3_GPIO_Port, enable_3_Pin, SET);
      enable_trigger = 3;
      trigger_num = 3;
      break;
    case 3:
      HAL_GPIO_WritePin(enable_0_GPIO_Port, enable_0_Pin, SET);
      HAL_GPIO_WritePin(enable_1_GPIO_Port, enable_1_Pin, SET);
      HAL_GPIO_WritePin(enable_2_GPIO_Port, enable_2_Pin, SET);
      HAL_GPIO_WritePin(enable_3_GPIO_Port, enable_3_Pin, RESET)
  ;
          enable_trigger = 0;
          trigger_num = 0;
          break;
        default:
          break;
      }

      display7SEG(trigger_num,
          seg_a_GPIO_Port, seg_a_Pin,
          seg_b_GPIO_Port, seg_b_Pin,
          seg_c_GPIO_Port, seg_c_Pin,
          seg_d_GPIO_Port, seg_d_Pin,
          seg_e_GPIO_Port, seg_e_Pin,
          seg_f_GPIO_Port, seg_f_Pin,
          seg_g_GPIO_Port, seg_g_Pin);

    }
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
```

Program 3.1: main loop

4 Exercise 3 + 4

```c
void update7SEG(int num)
{
    switch (num)
    {
    case 0:
      HAL_GPIO_WritePin(enable_0_GPIO_Port, enable_0_Pin, RESET);
      HAL_GPIO_WritePin(enable_1_GPIO_Port, enable_1_Pin, SET);
      HAL_GPIO_WritePin(enable_2_GPIO_Port, enable_2_Pin, SET);
      HAL_GPIO_WritePin(enable_3_GPIO_Port, enable_3_Pin, SET);
      break;
    case 1:
      HAL_GPIO_WritePin(enable_0_GPIO_Port, enable_0_Pin, SET);
      HAL_GPIO_WritePin(enable_1_GPIO_Port, enable_1_Pin, RESET);
      HAL_GPIO_WritePin(enable_2_GPIO_Port, enable_2_Pin, SET);
      HAL_GPIO_WritePin(enable_3_GPIO_Port, enable_3_Pin, SET);
      break;
    case 2:
      HAL_GPIO_WritePin(enable_0_GPIO_Port, enable_0_Pin, SET);
      HAL_GPIO_WritePin(enable_1_GPIO_Port, enable_1_Pin, SET);
      HAL_GPIO_WritePin(enable_2_GPIO_Port, enable_2_Pin, RESET);
      HAL_GPIO_WritePin(enable_3_GPIO_Port, enable_3_Pin, SET);
      break;
    case 3:
      HAL_GPIO_WritePin(enable_0_GPIO_Port, enable_0_Pin, SET);
      HAL_GPIO_WritePin(enable_1_GPIO_Port, enable_1_Pin, SET);
      HAL_GPIO_WritePin(enable_2_GPIO_Port, enable_2_Pin, SET);
      HAL_GPIO_WritePin(enable_3_GPIO_Port, enable_3_Pin, RESET);
      break;
    default:
      break;
    }
}
```

Program 4.1: **update7SEG()** function

```c
Set_timer(100, 0);
Set_timer(25, 1);
const int MAX_LED = 4;
int index_led = 0;
int led_buffer [4] = {1 , 2 , 3 , 4};

while (1)
{
  if(index_led >= MAX_LED)
  {
    index_led = 0;
  }
  if(timer_flag[0] == 1)
  {
    Set_timer(100, 0);
    HAL_GPIO_TogglePin(led_red_GPIO_Port, led_red_Pin);
  }
  if(timer_flag[1] == 1)
  {
    Set_timer(25, 1);
    display7SEG(led_buffer[index_led],
        seg_a_GPIO_Port, seg_a_Pin,
        seg_b_GPIO_Port, seg_b_Pin,
        seg_c_GPIO_Port, seg_c_Pin,
        seg_d_GPIO_Port, seg_d_Pin,
        seg_e_GPIO_Port, seg_e_Pin,
        seg_f_GPIO_Port, seg_f_Pin,
        seg_g_GPIO_Port, seg_g_Pin);
    update7SEG(index_led++);
  }
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
}
```

Program 4.2: Ex3 + Ex4 source

5 Exercise 5

```c
void updateClockBuffer(int hour,  int minute, int* led_buffer)
{

  if(hour >= 0 && hour <10)
  {
    led_buffer[0] = 0;
    led_buffer[1] = hour;
  }
  else
  {
    led_buffer[0] = hour/10;
    led_buffer[1] = hour%10;
  }

  if(minute >= 0 && minute <10)
  {
    led_buffer[2] = 0;
    led_buffer[3] = minute;
  }
  else
  {
    led_buffer[2] = minute/10;
    led_buffer[3] = minute%10;
  }
}
```

Program 5.1: **updateClockBuffer()** function

6 Exercise 7 + 8

```
Set_timer(100, 0);
Set_timer(50, 1);
int hour = 24 , minute = 59 , second = 50;
const int MAX_LED = 4;
int led_buffer [4] = {1 , 2 , 3 , 4};
int index_led = 0;
while (1)
{
  if(timer_flag[0] == 1)
  {
    Set_timer(100, 0);
    HAL_GPIO_TogglePin(led_red_GPIO_Port, led_red_Pin);
    HAL_GPIO_TogglePin(dot_GPIO_Port, dot_Pin);
    second++;
    if(second >= 60)
    {
      second = 0;
      minute++;
    }
    if(minute >= 60)
    {
      minute = 0;
      hour++;
    }
    if(hour >= 24)
    {
      hour = 0;
    }
  }

  updateClockBuffer(hour, minute, led_buffer);

  if(timer_flag[1] == 1)
  {
    Set_timer(50, 1);
    if(index_led >= MAX_LED)
    {
      index_led = 0;
    }
    switch (index_led)
    {
    case 0:
      HAL_GPIO_WritePin(enable_0_GPIO_Port, enable_0_Pin, RESET)
;
      HAL_GPIO_WritePin(enable_1_GPIO_Port, enable_1_Pin, SET);
      HAL_GPIO_WritePin(enable_2_GPIO_Port, enable_2_Pin, SET);
      HAL_GPIO_WritePin(enable_3_GPIO_Port, enable_3_Pin, SET);
```

```c
      break;
    case 1:
      HAL_GPIO_WritePin(enable_0_GPIO_Port, enable_0_Pin, SET);
      HAL_GPIO_WritePin(enable_1_GPIO_Port, enable_1_Pin, RESET);
      HAL_GPIO_WritePin(enable_2_GPIO_Port, enable_2_Pin, SET);
      HAL_GPIO_WritePin(enable_3_GPIO_Port, enable_3_Pin, SET);
      break;
    case 2:
      HAL_GPIO_WritePin(enable_0_GPIO_Port, enable_0_Pin, SET);
      HAL_GPIO_WritePin(enable_1_GPIO_Port, enable_1_Pin, SET);
      HAL_GPIO_WritePin(enable_2_GPIO_Port, enable_2_Pin, RESET);
      HAL_GPIO_WritePin(enable_3_GPIO_Port, enable_3_Pin, SET);
      break;
    case 3:
      HAL_GPIO_WritePin(enable_0_GPIO_Port, enable_0_Pin, SET);
      HAL_GPIO_WritePin(enable_1_GPIO_Port, enable_1_Pin, SET);
      HAL_GPIO_WritePin(enable_2_GPIO_Port, enable_2_Pin, SET);
      HAL_GPIO_WritePin(enable_3_GPIO_Port, enable_3_Pin, RESET);
      break;
    default:
      break;
    }

    display7SEG(led_buffer[index_led],
              seg_a_GPIO_Port, seg_a_Pin,
              seg_b_GPIO_Port, seg_b_Pin,
              seg_c_GPIO_Port, seg_c_Pin,
              seg_d_GPIO_Port, seg_d_Pin,
              seg_e_GPIO_Port, seg_e_Pin,
              seg_f_GPIO_Port, seg_f_Pin,
              seg_g_GPIO_Port, seg_g_Pin);
    index_led++;
  }

  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
}
```

Program 6.1: Ex7 + Ex8 source