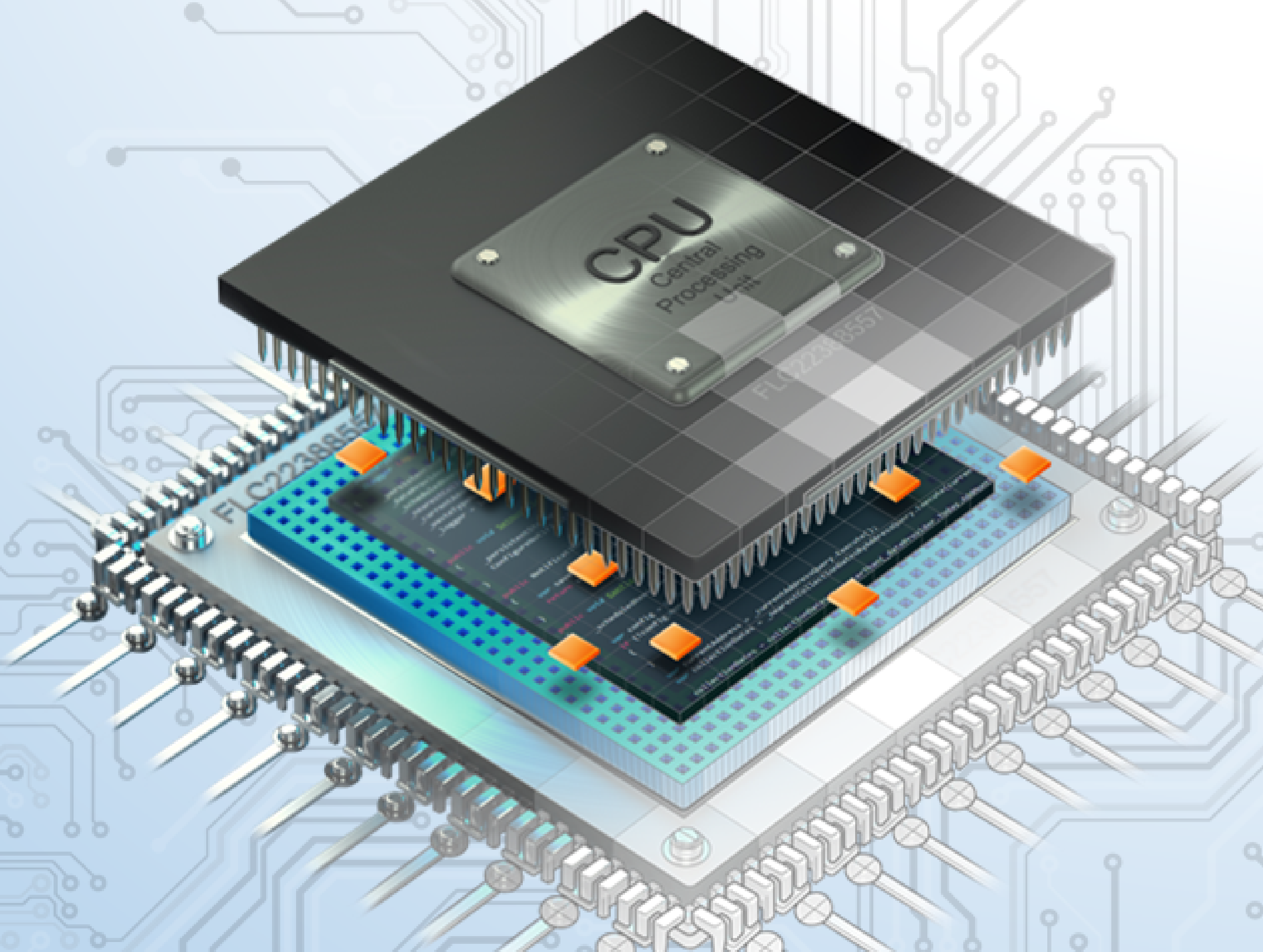
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
COMPUTER ENGINEERING

Microcontroller

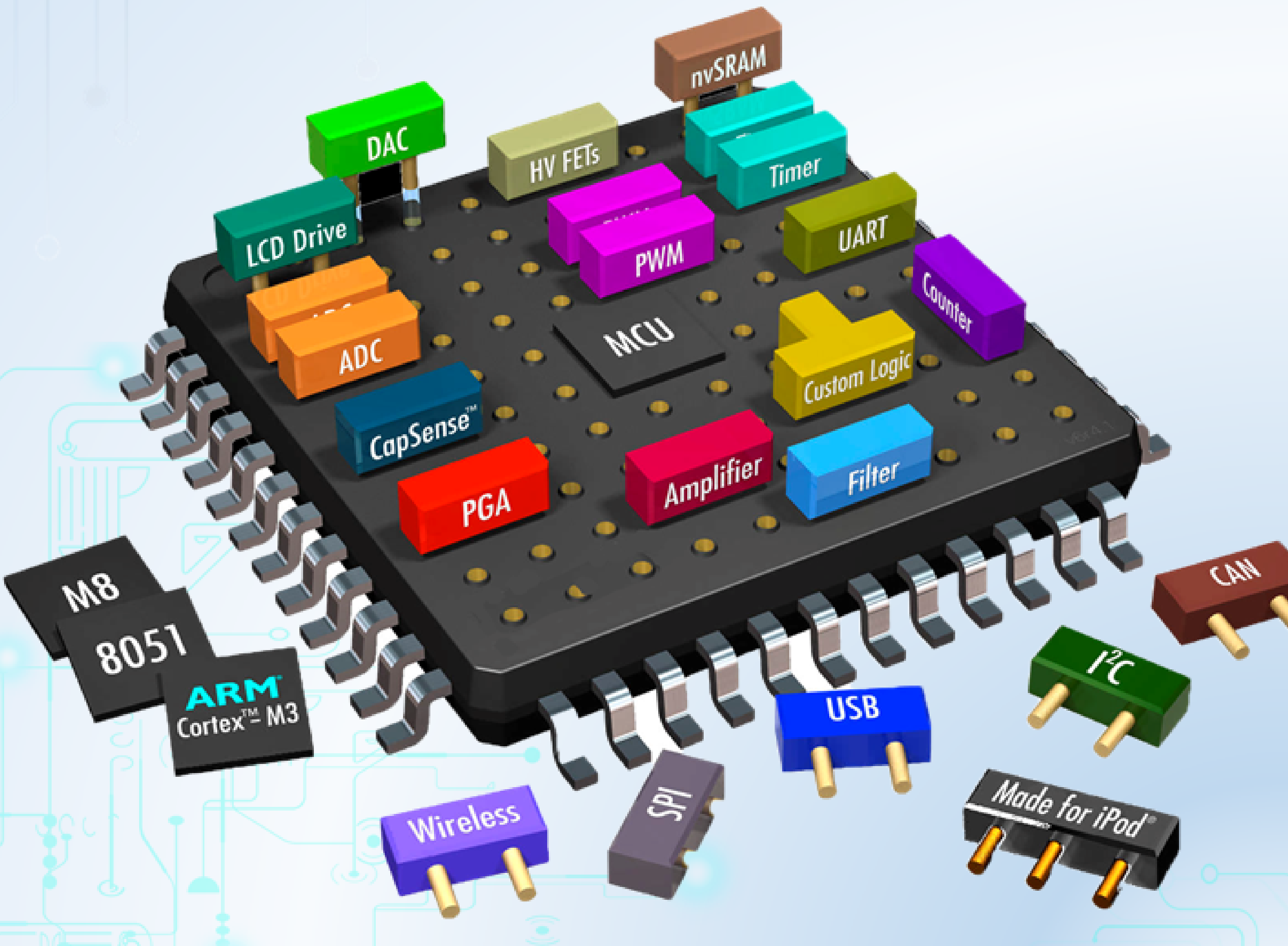


Dr. Le Trong Nhan

Contents

CHAPTER 1

Timer Interrupt and LED Scanning



1 Introduction

Timers are one of the most important features in modern micro-controllers. They allow us to measure how long something takes to execute, create non-blocking code, precisely control pin timing, and even run operating systems. In this manual, how to configure a timer using STM32CubeIDE is presented how to use them to flash an LED. Finally, students are proposed to finalize 10 exercises using timer interrupt for applications based LED Scanning.

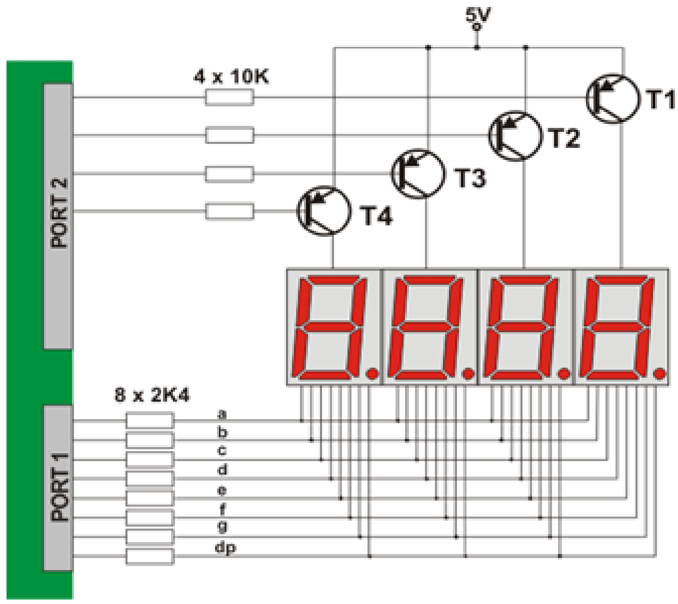


Figure 1.1: *Four seven segment LED interface for a micro-controller*

Design an interface for with multiple LED (seven segment or matrix) displays which is to be controlled is depends on the number of input and output pins needed for controlling all the LEDs in the given matrix display, the amount of current that each pin can source and sink and the speed at which the micro-controller can send out control signals. With all these specifications, interfacing can be done for 4 seven segment LEDs with a micro-controller is proposed in the figure above.

In the above diagram each seven segment display is having 8 internal LEDs, leading to the total number of LEDs is 32. However, not all the LEDs are required to turn ON, but one of them is needed. Therefore, only 12 lines are needed to control the whole 4 seven segment LEDs. By controlling with the micro-controller, we can turn ON an LED during a same interval T_S. Therfore, the period for controlling all 4 seven segment LEDs is $4T_S$. In other words, these LEDs are scanned at frequecy $f = 1/4T_S$. Finally, it is obviously that if the frequency is greater than 30Hz (e.g. f = 50Hz), it seems that all LEDs are turn ON at the same time.

In this manual, the timer interrupt is used to design the interval T_S for LED scanning. Unfortunately, the simulation on Proteus can not execute at high frequency, the frequency f is set to a low value (e.g. 1Hz). In a real implementation, this frequency should be 50Hz.

2 Timer Interrupt Setup

Step 1: Create a simple project, which LED connected to PA5. The manual can be found in the first lab.

Step 2: Check the clock source of the system on the tab **Clock Configuration** (from *.ioc file). In the default configuration, the internal clock source is used with 8MHz, as shown in the figure bellow.

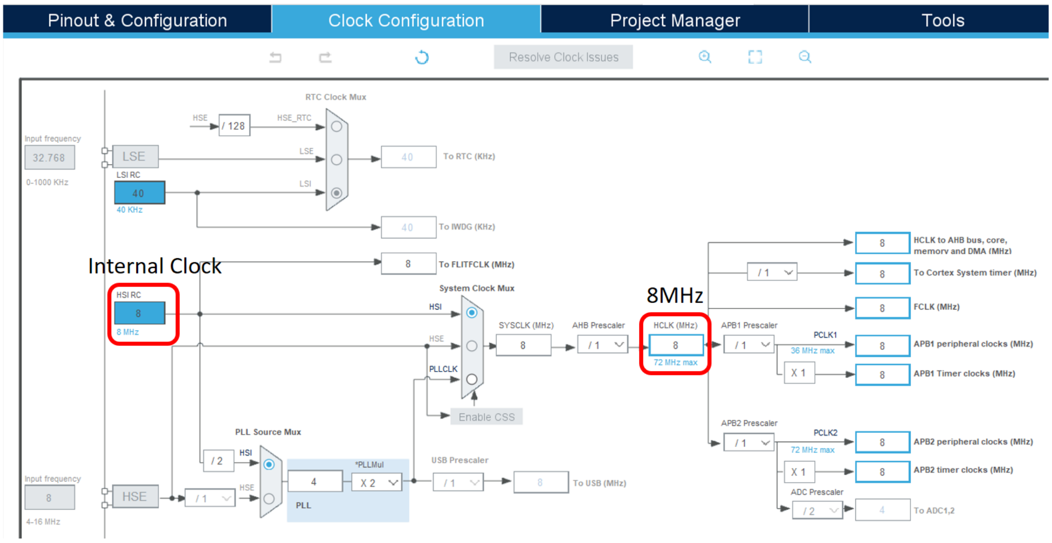


Figure 1.2: Default clock source for the system

Step 3: Configure the timer on the **Parameter Settings**, as follows:

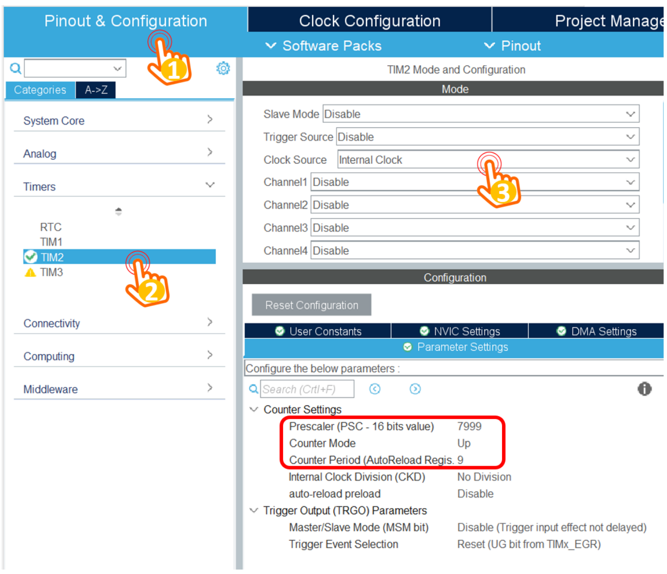


Figure 1.3: Configure for Timer 2

Select the clock source for timer 2 to the **Internal Clock**. Finally, set the prescaller and the counter to 7999 and 9, respectively. These values are explained as follows:

- The target is to set an interrupt timer to 10ms

- The clock source is 8MHz, by setting the prescaller to 7999, the input clock source to the timer is **8MHz/(7999+1) = 1000Hz**.

- The interrupt is raised when the timer counter is counted from 0 to 9, meaning that the frequency is divided by 10, which is 100Hz.

- The frequency of the timer interrupt is 100Hz, meaning that the period is **1/100Hz = 10ms**.

Step 4: Enable the timer interrupt by switching to **NIVC Settings** tab, as follows:

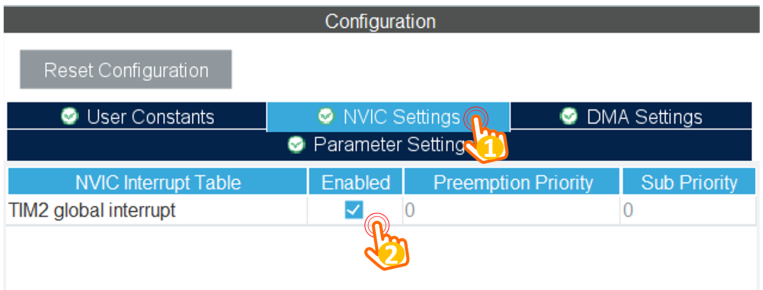


Figure 1.4: Enable timer interrupt

Finally, save the configuration file to generate the source code.

Step 5: On the **main()** function, call the timer init function, as follows:

```c
int main(void)
{
  HAL_Init();
  SystemClock_Config();

  MX_GPIO_Init();
  MX_TIM2_Init();

  /* USER CODE BEGIN 2 */
  HAL_TIM_Base_Start_IT(&htim2);
  /* USER CODE END 2 */

  while (1){

  }
}
```

Program 1.1: Init the timer interrupt in main

Please put the init function in a right place to avoid conflicts when code generation is executed (e.g. ioc file is updated).

Step 6: Add the interrupt service routine function, this function is invoked every 10ms, as follows:

```
/* USER CODE BEGIN 4 */
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
  {

}
/* USER CODE END 4 */
```

Program 1.2: Add an interrupt service routine

Step 7: To run a LED Blinky demo using interrupt, a short manual is presented as follows:

```
/* USER CODE BEGIN 4 */
int counter = 100;
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
  {
  counter--;
  if(counter <= 0){
    counter = 100;
    HAL_GPIO_TogglePin(LED_RED_GPIO_Port, LED_RED_Pin);
  }
}
/* USER CODE END 4 */
```

Program 1.3: LED Blinky using timer interrupt

The **HAL_TIM_PeriodElapsedCallback** function is an infinite loop, which is invoked every cycle of the timer 2, in this case, is 10ms.

3 Exercise and Report

3.1 Exercise 1

The first exercise show how to interface for multiple seven segment LEDs to STM32F103C6 micro-controller (MCU). Seven segment displays are common anode type, meaning that the anode of all LEDs are tied together as a single terminal and cathodes are left alone as individual pins.

In order to save the resource of the MCU, individual cathode pins from all the seven segment LEDs are connected together, and connect to 7 pins of the MCU. These pins are popular known as the **signal pins**. Meanwhile, the anode pin of each seven segment LEDs are controlled under a power enabling circuit, for instance, an PNP transistor. At a given time, only one seven segment LED is turned on. However, if the delay is small enough, it seems that all LEDs are enabling.

Implement the circuit simulation in Proteus with two 7-SEGMENT LEDs as following:

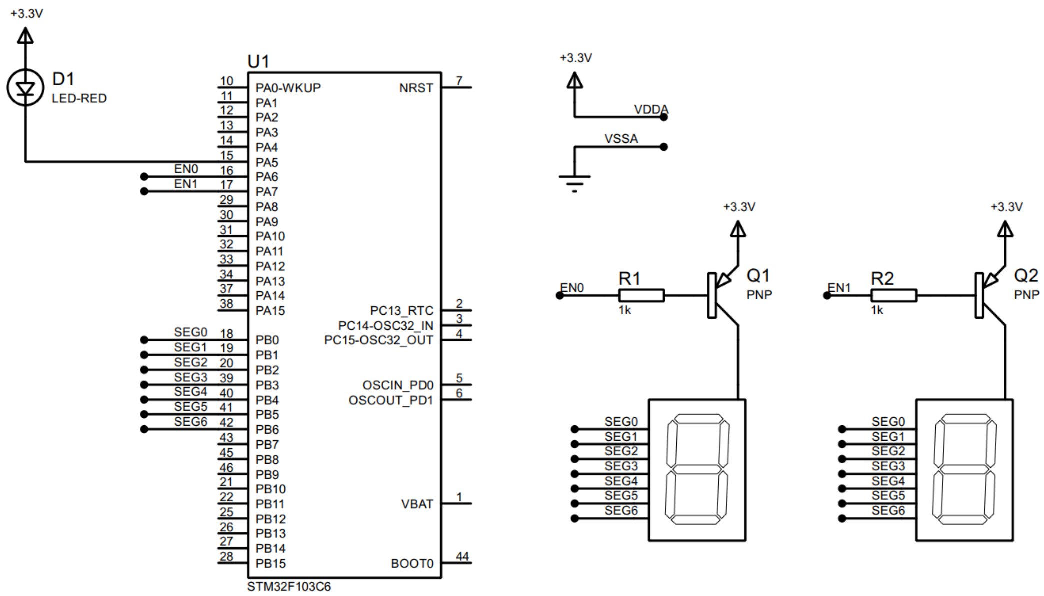


Figure 1.5: Simulation schematic in Proteus

Components used in the schematic are listed bellow:

- 7SEG-COM-ANODE (connected from PB0 to PB6)
- LED-RED
- PNP
- RES
- STM32F103C6

Students are proposed to use the function **display7SEG(int num)** in the Lab 1 in this exercise. Implement the source code in the interrupt callback function to display number **"1"** on the first seven segment and number **"2"** for second one. The switching time between

2 LEDs is half of second.

Report 1: Capture your schematic from Proteus and show in the report.

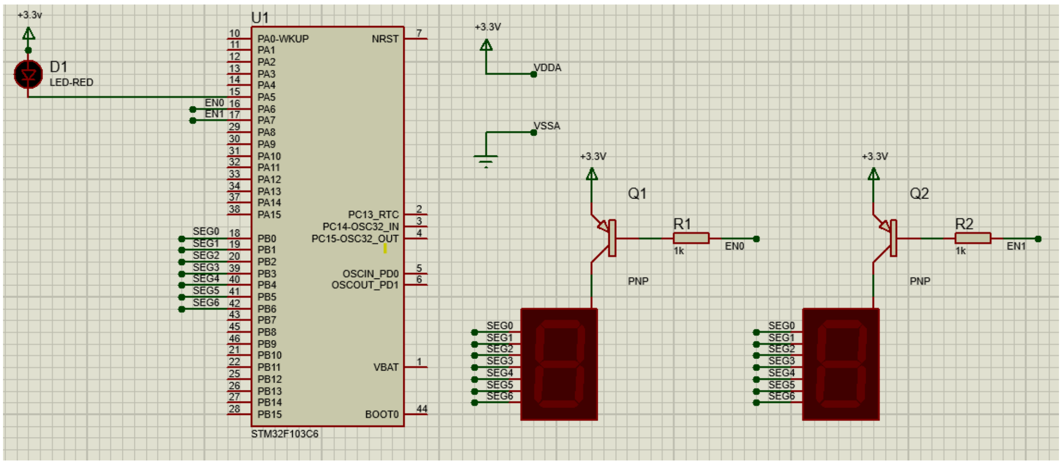


Figure 1.6: Simulation schematic in Proteus

Report 2: Present your source code in the **HAL_TIM_PeriodElapsedCallback** function.

```
/* USER CODE BEGIN 4 */
int counter = 100; // 1 second
void HAL_TIM_PeriodElapsedCallback( TIM_HandleTypeDef *htim ){
  counter --;
  if(counter >= 50){  // From From 1st second to 0.5th second, segment 1 turns on
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, 0);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, 0);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, 1);
    display7SEG(1);
  }
  else if(counter >= 0){  // From the 0.5th second to the 0th second, segment 2 turns on
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, 1);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, 1);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, 0);
    display7SEG(2);
  }else counter = 100; // return 1 second
}
/* USER CODE END 4 */
```

Program 1.4: 7-SEGMENT LED using timer interrupt

Short question: What is the frequency of the scanning process?
-> The frequency of the timer interrupt is 100Hz.

3.2 Exercise 2

Extend to 4 seven segment LEDs and two LEDs (connected to PA4, labeled as **DOT**) in the middle as following:

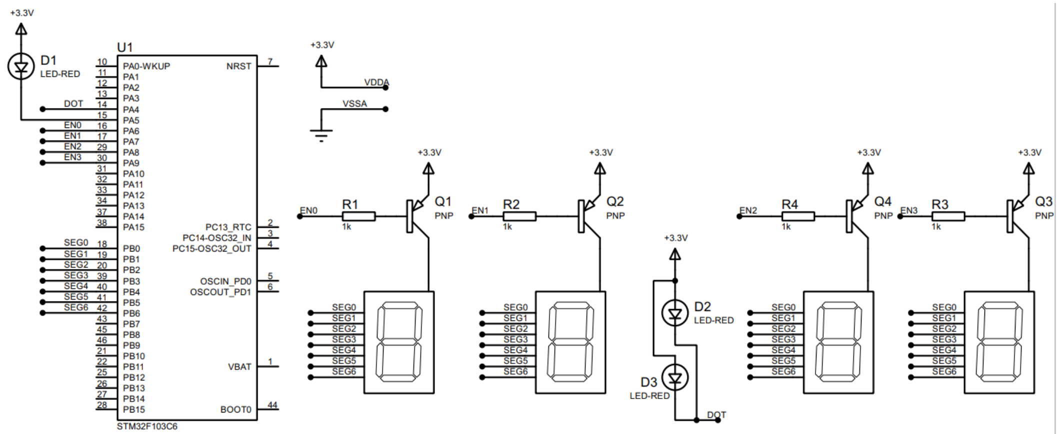


Figure 1.7: Simulation schematic in Proteus

Blink the two LEDs every second. Meanwhile, number 3 is displayed on the third seven segment and number 0 is displayed on the last one (to present 12 hour and a half). The switching time for each seven segment LED is also a half of second (500ms). **Implement your code in the timer interrupt function.**

Report 1: Capture your schematic from Proteus and show in the report.

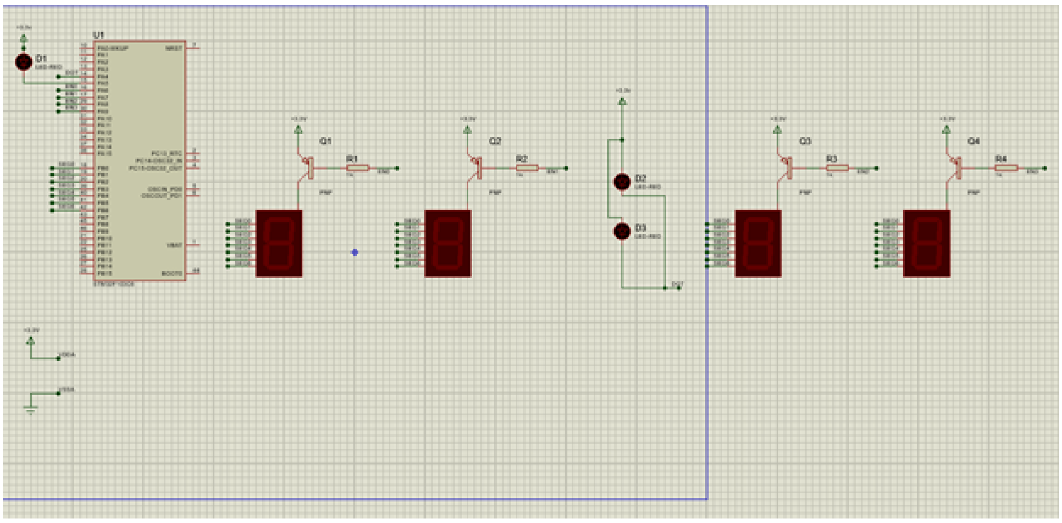


Figure 1.8: Simulation schematic in Proteus

Report 2: Present your source code in the **HAL_TIM_PeriodElapsedCallback** function.

```
/* USER CODE BEGIN 4 */
// Function to display 7-segment LED
void display7SEG(int counter){
      switch(counter){
```

```
    case 0:
      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, 1);
      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5, 0);
      break;
    case 1:
      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0|GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6, 1);
      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1|GPIO_PIN_2, 0);
      break;
    case 2:
      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_2|GPIO_PIN_5, 1);
      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_6, 0);
      break;
    case 3:
      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4|GPIO_PIN_5, 1);
      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_6, 0);
      break;
    case 4:
      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0|GPIO_PIN_3|GPIO_PIN_4, 1);
      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_5|GPIO_PIN_6, 0);
      break;
    case 5:
      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1|GPIO_PIN_4, 1);
      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0|GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_5|GPIO_PIN_6, 0);
      break;
    case 6:
      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1, 1);
      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0|GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6, 0);
      break;
    case 7:
      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6, 1);
      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2, 0);
      break;
    case 8:
      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6, 0);
      break;
    case 9:
      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, 1);
```

```
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0|GPIO_PIN_1|
  GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_5|GPIO_PIN_6, 0);
        break;
      }
    }

// Function to turn off 7-segment LEDs
void clearAllClock(){
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6| GPIO_PIN_7|
  GPIO_PIN_8| GPIO_PIN_9, 1);
  }

// Function to display 7 segment led with a certain number
void setNumberOnClock(int num){
    switch(num){
    case 0: HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, 0);
        break;
    case 1: HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, 0);
        break;
    case 2: HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, 0);
        break;
    case 3: HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, 0);
        break;
    }

  }


int counter = 200; // 2 seconds is the total display time
  of 4 led 7 segments
int counter_led = 50; // 0.5 second to blink two LEDs
void HAL_TIM_PeriodElapsedCallback ( TIM_HandleTypeDef *
  htim ){
  counter--;
  counter_led--;

  if(counter_led <= 0){ // If counter_led is less than 0
  then reverse the LED pin and return 50
    HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_4);
    counter_led = 50;
  }

  if(counter >= 150){  // If counter is greater than or
  equal to 150, turn off all 7-segment LEDs, and turn on
  the first 7-segment led.
    clearAllClock();
    setNumberOnClock(0);
    display7SEG(1);
  }
```

```
  else if(counter >= 100){ //If counter is greater than or
   equal to 100, turn off all 7-segment LEDs, and turn on
   the second 7-segment led.
    clearAllClock();
    setNumberOnClock(1);
    display7SEG(2);
  }
  else if(counter >= 50 ){ // If counter is greater than or
    equal to 50, turn off all 7-segment LEDs, and turn on
   the third 7-segment led.
    clearAllClock();
    setNumberOnClock(2);
    display7SEG(3);
  }
  else if(counter >= 0){ // If counter is greater than or
   equal to 0, turn off all 7-segment LEDs, and turn on the
    fourth 7-segment led.
    clearAllClock();
    setNumberOnClock(3);
    display7SEG(0);
  }
else {
    counter= 200; // return 200
  }

}

/* USER CODE END 4 */
```

Program 1.5: 4 SEVEN-SEGMENT LEDs using timer interrupt

Short question: What is the frequency of the scanning process?
-> The frequency of the timer interrupt is 100Hz.

3.3 Exercise 3

Implement a function named **update7SEG(int index)**. An array of 4 integer numbers are declared in this case. The code skeleton in this exercise is presented as following:

```
const int MAX_LED = 4;
int index_led = 0;
int led_buffer[4] = {1, 2, 3, 4};
void update7SEG(int index){
    switch (index){
        case 0:
            //Display the first 7SEG with led_buffer[0]
            break;
        case 1:
            //Display the second 7SEG with led_buffer[1]
            break;
        case 2:
            //Display the third 7SEG with led_buffer[2]
            break;
        case 3:
            //Display the forth 7SEG with led_buffer[3]
            break;
        default:
            break;
    }
}
```

Program 1.6: An example for your source code

This function should be invoked in the timer interrupt, e.g update7SEG(index_led++). The variable **index_led** is updated to stay in a valid range, which is from 0 to 3.

Report 1: Present the source code of the update7SEG function.

```

// Function to display 7-segment LED
void display7SEG(int counter){
      switch(counter){
      case 0:
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, 1);
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0|GPIO_PIN_1|
   GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5, 0);
        break;
      case 1:
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0|GPIO_PIN_3|
   GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6, 1);
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1|GPIO_PIN_2, 0);
        break;
      case 2:
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_2|GPIO_PIN_5, 1);
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0|GPIO_PIN_1|
```

```
GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_6, 0);
        break;
      case 3:
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4|GPIO_PIN_5, 1);
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0|GPIO_PIN_1|
GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_6, 0);
        break;
      case 4:
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0|GPIO_PIN_3|
GPIO_PIN_4, 1);
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1|GPIO_PIN_2|
GPIO_PIN_5|GPIO_PIN_6, 0);
        break;
      case 5:
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1|GPIO_PIN_4, 1);
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0|GPIO_PIN_2|
GPIO_PIN_3|GPIO_PIN_5|GPIO_PIN_6, 0);
        break;
      case 6:
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1, 1);
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0|GPIO_PIN_2|
GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6, 0);
        break;
      case 7:
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3|GPIO_PIN_4|
GPIO_PIN_5|GPIO_PIN_6, 1);
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0|GPIO_PIN_1|
GPIO_PIN_2, 0);
        break;
      case 8:
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0|GPIO_PIN_1|
GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6,
0);
        break;
      case 9:
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, 1);
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0|GPIO_PIN_1|
GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_5|GPIO_PIN_6, 0);
        break;
      }
    }

// Function to turn off 7-segment LEDs
void clearAllClock(){
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6| GPIO_PIN_7|
GPIO_PIN_8| GPIO_PIN_9, 1);
  }


```

```c
int led_buffer [4] = {1 , 2 , 3 , 4};
void update7SEG ( int index ) {
  switch ( index ) {
  case 0:
    // Display the first 7 SEG with led_buffer [0]
    clearAllClock(); // Tunrn off 7-segment LEDs
    display7SEG(led_buffer[0]); // Display 7-segment LED with led_buffer[0]
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, 0); // Turn on the first 7-segment led.
    break ;
  case 1:
    // Display the second 7 SEG with led_buffer [1]
    clearAllClock();
    display7SEG(led_buffer[1]);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, 0);
    break ;
  case 2:
    // Display the third 7 SEG with led_buffer [2]
    clearAllClock();
    display7SEG(led_buffer[2]);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, 0);
    break ;
  case 3:
    // Display the forth 7 SEG with led_buffer [3]
    clearAllClock();
    display7SEG(led_buffer[3]);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, 0);
    break ;
  default :
    break ;
  }
}
```

Program 1.7: Implement a function named **update7SEG(int index)**

Report 2: Present the source code in the HAL_TIM_PeriodElapsedCallback.

```c

const int MAX_LED = 4;
int index_led = 0;
int counter = 50; // The switching time for each seven segment LED is a half of second (500ms) and to  blink two  LEDs
void HAL_TIM_PeriodElapsedCallback ( TIM_HandleTypeDef * htim ){
  counter--;
  if(counter <= 0){ // If  counter  is less  than 0
    HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_4); // then  reverse the  LED  pin
```

```
    update7SEG(index_led);  // update 7-segment led
    index_led++;            // increase index
    if(index_led >= MAX_LED){  // If index_led is greater than MAX_LED then return 0
      index_led =0;
    }
    Counter = 50; // return  50
  }
}
```

Program 1.8: The source code in the HAL_TIM_PeriodElapsedCallback.

Students are proposed to change the values in the **led_buffer** array for unit test this function, which is used afterward.

3.4 Exercise 4

Change the period of invoking update7SEG function in order to set the frequency of 4 seven segment LEDs to 1Hz. The DOT is still blinking every second.

Report 1: Present the source code in the **HAL_TIM_PeriodElapsedCallback**.

```

const int MAX_LED = 4; // 4 led 7 segments
int index_led = 0;  // current 7 segment led index: the first 7-segment  led
int counter = 25; // The switching time for each seven segment LED is a half of second (250ms)
int counter_led = 50; // 0.5  second  to  blink  two  LEDs

void HAL_TIM_PeriodElapsedCallback ( TIM_HandleTypeDef * htim ){
  counter--;
  counter_led--;
  if(counter_led <= 0){ // If  counter_led  is less  than 0  then  reverse  the  LED  pin  and  return  50
    HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_4);
    counter_led =50;
  }

  if(counter <= 0){ // If counter is small or equal to 0
    update7SEG(index_led); //  update 7-segment  led
    index_led++; // increase index
    if(index_led == MAX_LED){ // If  index_led  is  greater  than  MAX_LED  then  return 0
      index_led =0;
    }
    counter= 25; //  return  25
  }
```

```
}
```

Program 1.9: The source code in the **HAL_TIM_PeriodElapsedCallback**

3.5 Exercise 5

Implement a digital clock with **hour** and **minute** information displayed by 2 seven segment LEDs. The code skeleton in the **main** function is presented as follows:

```
int hour = 15, minute = 8, second = 50;

while(1){
    second++;
    if (second >= 60){
        second = 0;
        minute++;
    }
    if(minute >= 60){
        minute = 0;
        hour++;
    }
    if(hour >=24){
        hour = 0;
    }
    updateClockBuffer();
    HAL_Delay(1000);
}
```

Program 1.10: An example for your source code

The function **updateClockBuffer** will generate values for the array **led_buffer** according to the values of hour and minute. In the case these values are 1 digit number, digit 0 is added.

Report 1: Present the source code in the **updateClockBuffer** function.

```
int hour = 15 , minute = 8 , second = 50;

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

Program 1.11: The source code in the **updateClockBuffer** function

3.6 Exercise 6

The main target from this exercise to reduce the complexity (or reduce code processing) in the timer interrupt. The time consumed in the interrupt can lead to the nested interrupt issue, which can crash the whole system. A simple solution can disable the timer whenever the interrupt occurs, the enable it again. However, the real-time processing is not guaranteed anymore.

In this exercise, a software timer is created and its counter is count down every timer interrupt is raised (every 10ms). By using this timer, the **Hal_Delay(1000)** in the main function is removed. In a MCU system, non-blocking delay is better than blocking delay. The details to create a software timer are presented bellow. The source code is added to your current program, **do not delete the source code you have on Exercise 5.**

Step 1: Declare variables and functions for a software timer, as following:

```
/* USER CODE BEGIN 0 */
int timer0_counter = 0;
int timer0_flag = 0;
int TIMER_CYCLE = 10;
void setTimer0(int duration){
  timer0_counter = duration /TIMER_CYCLE;
  timer0_flag = 0;
}
void timer_run(){
  if(timer0_counter > 0){
    timer0_counter--;
    if(timer0_counter == 0) timer0_flag = 1;
  }
}
/* USER CODE END 0 */
```

Program 1.12: Software timer based timer interrupt

Please change the **TIMER_CYCLE** to your timer interrupt period. In the manual code above, it is **10ms**.

Step 2: The **timer_run()** is invoked in the timer interrupt as following:

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{

  timer_run();

  //YOUR OTHER CODE
}
```

Program 1.13: Software timer based timer interrupt

Step 3: Use the timer in the main function by invoked setTimer0 function, then check for its flag (timer0_flag). An example to blink an LED connected to PA5 using software timer is shown as follows:

```
setTimer0(1000);
```

```
2 while (1){
3     if(timer0_flag == 1){
4         HAL_GPIO_TogglePin(LED_RED_GPIO_Port, LED_RED_Pin);
5         setTimer0(2000);
6     }
7 }
```

Program 1.14: Software timer is used in main fuction to blink the LED

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

Reply: LED RED does not change
-> If you skip line 1: setTimer0(1000), the timer0_counter value is always 0, leading to timer0_flag is always 0 so the LED RED status does not change.

Report 2: if in line 1 of the code above is changed to setTimer0(1), what happens after that and why?

Reply: LED RED does not change
-> After executing setTimer0(1), timer0_counter is still zero, so timer0_flag is always zero, so Led RED status does not change.

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?

Reply: Led Red Blinking
-> After executing setTimer0(10), timer0_counter = 1 > 0, so after executing time_run(), timer0_flag = 1, resulting in the if function in the while loop being executed, so the state of Led Red is reversed. Continuing to execute the setTimer0(2000) function, timer0_counter = 200 > 0, after timer0_counter decreases to 0, timer0_flag = 1, resulting in the if function in the while loop being executed, so the state of Led Red is reversed.

3.7 Exercise 7

Upgrade the source code in Exercise 5 (update values for hour, minute and second) by using the software timer and remove the HAL_Delay function at the end. Moreover, the DOT (connected to PA4) of the digital clock is also moved to main function.

Report 1: Present your source code in the while loop on main function.

```

int timer0_counter = 0; // Time to update 7SEG
int timer0_flag = 0;
int TIMER_CYCLE = 10;

void setTimer0(int duration){
  timer0_counter = duration /TIMER_CYCLE;
  timer0_flag = 0;
}

```

```
void timer_run(){
  if(timer0_counter > 0){
    timer0_counter--;
    if(timer0_counter == 0) timer0_flag = 1;
  }
}

setTimer0(1000); // The switching time for each
 sevensegment LED
while (1)
{
  /* USER CODE END WHILE */

  if(timer0_flag == 1){
      HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_4);
      second ++;
      if ( second >= 60) {
        second = 0;
        minute ++;
      }
      if( minute >= 60) {
        minute = 0;
        hour ++;
      }
      if( hour >=24) {
        hour = 0;
      }
      updateClockBuffer () ;
      setTimer0(1000);
  }

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

Program 1.15: The source code in the while loop on main function

3.8 Exercise 8

Move also the update7SEG() function from the interrupt timer to the main. Finally, the timer interrupt only used to handle software timers. All processing (or complex computations) is move to an infinite loop on the main function, optimizing the complexity of the interrupt handler function.

Report 1: Present your source code in the main function. In the case more extra functions are used (e.g. the second software timer), present them in the report as well.

```
int hour = 15 , minute = 8 , second = 57;
int led_buffer [4] = {1 , 2 , 3 , 4};

//  Function  update led_buffer[]
```

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

  int timer0_counter = 0; // Variable count to LED blink
  int timer0_flag = 0; // Flag to signal blink LED
  int timer1_counter = 0; // Variable count to update 7SEG
  int timer1_flag = 0; // Flag to signal update 7SEG
  int TIMER_CYCLE = 10; // timer cycle is 10ms

// Function setTimer0 to establish again timer0_counter and
    timer0_flag
  void setTimer0(int duration){
      timer0_counter = duration /TIMER_CYCLE;
      timer0_flag = 0;
  }

// Function setTimer1 to establish again timer1_counter and
    timer1_flag
  void setTimer1(int duration){
    timer1_counter = duration /TIMER_CYCLE;
    timer1_flag = 0;
  }

// The timer_run()is invoked in the timer interrupt
   void timer_run(){
    if(timer0_counter > 0){ // If timer0_counter > 0
        timer0_counter--; // Decrease timer0_counter
        // If timer0_counter = 0, then timer0_flag = 1 to
   blink LED
        if(timer0_counter == 0) timer0_flag = 1;
    }

    if(timer1_counter > 0){ // If timer1_counter > 0
      timer1_counter--; // Decrease timer1_counter
      // If timer1_counter = 0, then timer1_flag = 1 to
   update 7SEG
      if(timer1_counter == 0) timer1_flag = 1;
    }
  }

//  Function to display 7-segment LED
  void display7SEG(int counter){
      switch(counter){
      case 0:
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, 1);
```

```
            HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5, 0);
            break;
        case 1:
            HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0|GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6, 1);
            HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1|GPIO_PIN_2, 0);
            break;
        case 2:
            HAL_GPIO_WritePin(GPIOB, GPIO_PIN_2|GPIO_PIN_5, 1);
            HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_6, 0);
            break;
        case 3:
            HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4|GPIO_PIN_5, 1);
            HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_6, 0);
            break;
        case 4:
            HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0|GPIO_PIN_3|GPIO_PIN_4, 1);
            HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_5|GPIO_PIN_6, 0);
            break;
        case 5:
            HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1|GPIO_PIN_4, 1);
            HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0|GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_5|GPIO_PIN_6, 0);
            break;
        case 6:
            HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1, 1);
            HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0|GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6, 0);
            break;
        case 7:
            HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6, 1);
            HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2, 0);
            break;
        case 8:
            HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6, 0);
            break;
        case 9:
            HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, 1);
            HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_5|GPIO_PIN_6, 0);
```

```
        break;
      }
    }

//  Function to turn off 7-segment LEDs
void clearAllClock(){
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6| GPIO_PIN_7|
   GPIO_PIN_8| GPIO_PIN_9, 1);
}

// Function update7SEG to display led 7SEG with led_buffer
   []
void update7SEG ( int index ) {
  switch ( index ) {
  case 0:
    // Display the first 7 SEG with led_buffer [0]
    clearAllClock(); // Tunrn  off 7-segment  LEDs
    //  Display 7-segment  LED with  led_buffer [0]
    display7SEG(led_buffer[0]);
    // Turn on the first 7-segment  led.
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, 0);
    break ;
  case 1:
    // Display the second 7 SEG with led_buffer [1]
    clearAllClock();
    display7SEG(led_buffer[1]);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, 0);
    break ;
  case 2:
    // Display the third 7 SEG with led_buffer [2]
    clearAllClock();
    display7SEG(led_buffer[2]);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, 0);
    break ;
  case 3:
    // Display the forth 7 SEG with led_buffer [3]
    clearAllClock();
    display7SEG(led_buffer[3]);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, 0);
    break ;
  default :
    break ;
  }
}


// The switching time for each seven segment LED is a half
   of second (250ms)
setTimer1(250);
```

```
  setTimer0(500); // set time to blink LED is 500ms
const int MAX_LED = 4;
int index_led = 0;
while (1)
{
    // If counter is less than 0
    if(timer0_flag == 1){
      // then reverse the LED pin
      HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
      HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_4);
      setTimer0(500); // set again 500 ms
    }

      if(timer1_flag == 1){
      update7SEG(index_led); //   update 7-segment led
      index_led++;  // increase index

// If index_led > MAX_LED, then update clock (4 * 250 ms =
    1000ms = 1s)
      if(index_led > MAX_LED){
        index_led =0;
        second ++;
        if ( second >= 60) {
          second = 0;
            minute ++;
        }
        if( minute >= 60) {
            minute = 0;
            hour ++;
        }
        if( hour >=24) {
            hour = 0;
        }
      }
      updateClockBuffer () ; // update led_buffer[]
        setTimer1(250); // set again 250ms
    }
  }

// Software timer is used in main fuction
void HAL_TIM_PeriodElapsedCallback ( TIM_HandleTypeDef *
   htim ){

  timer_run();

}
```

Program 1.16: The source code in the main function

3.9 Exercise 9

This is an extra works for this lab. A LED Matrix is added to the system. A reference design is shown in figure bellow:

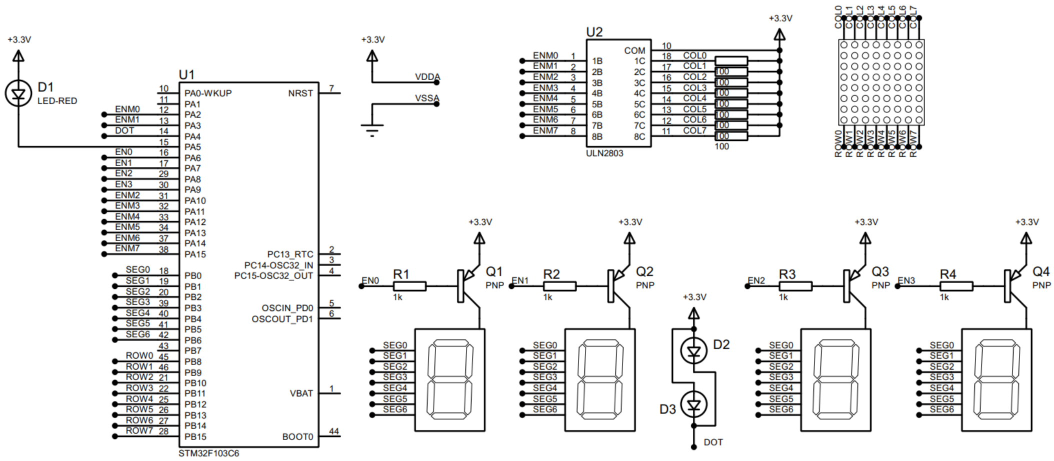


Figure 1.9: LED matrix is added to the simulation

In this schematic, two new components are added, including the **MATRIX-8X8-RED** and **ULN2803**, which is an NPN transistor array to enable the power supply for a column of the LED matrix. Students can change the enable signal (from ENM0 to ENM7) if needed. Finally, the data signal (from ROW0 to ROW7) is connected to PB8 to PB15.

Report 1: Present the schematic of your system by capturing the screen in Proteus.

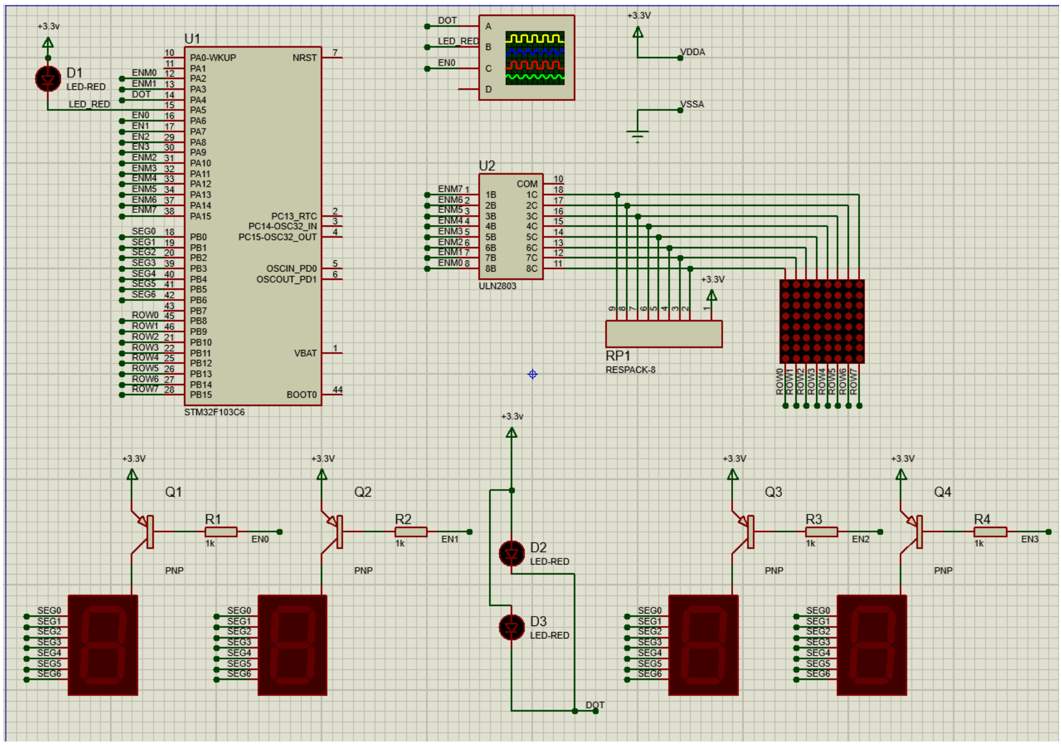


Figure 1.10: LED matrix is added to the simulation

Report 2: Implement the function, updateLEDMatrix(int index), which is similarly to 4 seven led segments.

```
const int MAX_LED_MATRIX = 8;
  int index_led_matrix = 0;
  uint8_t matrix_buffer[8] = {0x00, 0xFC, 0xFE, 0x33, 0x33, 0xFE, 0xFC, 0x0};

// Function remove all the column of led matrix
  void clearMatrix(){
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_10
        |GPIO_PIN_11|GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15, 1);
  }

  void updateLEDMatrix(int index){
   // Set and clear B15->B8 while preserving the state of all other pins in the port
    GPIOB->BSRR = 0b1111111100000000; // Set B15->B8 (HIGH)
    // Clear matrix_buffer[index] << 8 (LOW)
    GPIOB->BRR = matrix_buffer[index] << 8 ;
    clearMatrix(); //Remove all the column of led matrix
      switch(index){
      case 0:
          // Turn on column 1
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, 0);
        break;
      case 1:
          // Turn on column 2
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3, 0);
        break ;
      case 2:
          // Turn on column 3
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, 0);
        break ;
      case 3:
          // Turn on column 4
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_11, 0);
        break ;
      case 4:
          // Turn on column 5
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_12, 0);
        break ;
      case 5:
          // Turn on column 6
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_13, 0);
        break ;
      case 6:
          // Turn on column 7
```

```
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_14, 0);
        break;
      case 7:
          // Turn on column 8
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_15, 0);
        break;
      default:
        break;

      }
    }

int timer2_counter = 0; // Time to update LED MATRIX
int timer2_flag = 0; // Flag to update LED MATRIX
int TIMER_CYCLE = 10; // timer cycle (10ms)

  // Function setTimer2 to establish again timer2_counter and timer2_flag
  void setTimer2(int duration){
      timer2_counter = duration /TIMER_CYCLE;
      timer2_flag = 0;
    }

// The timer_run () is invoked in the timer interrupt
void timer_run(){
  if(timer2_counter > 0){ // If timer2_counter > 0
      timer2_counter--; // Decrease timer2_counter
      // If timer2_counter = 0, then timer2_flag = 1 to update LED MATRIX
      if(timer2_counter == 0) timer2_flag = 1;
  }
}

  setTimer2(10); // set time to update LED MATRIX is 10ms


  while (1)
  {
      if(timer2_flag == 1){
        updateLEDMatrix(index_led_matrix);  // update LED MATRIX
        index_led_matrix++;
        if(index_led_matrix >= MAX_LED_MATRIX){
          index_led_matrix =0;
        }
        setTimer2(10); // set again 10ms
      }
  }

```

```
  // Software timer is used in main fuction
void HAL_TIM_PeriodElapsedCallback ( TIM_HandleTypeDef * htim ){
  timer_run();
}
```

Program 1.17: Function to display data on LED Matrix

Student are free to choose the invoking frequency of this function. However, this function is supposed to invoked in main function. Finally, please update the **matrix_buffer** to display character **"A"**.

3.10 Exercise 10

Create an animation on LED matrix, for example, the character is shifted to the left.
Report 1: Briefly describe your solution and present your source code in the report.

```
int timer2_counter = 0; // Time to update LED MATRIX
 int timer2_flag = 0; // Flag to update LED MATRIX
 int TIMER_CYCLE = 1; // timer  cycle  (1ms)

/* USER CODE END 0 */

 void setTimer2(int duration){
      timer2_counter = duration /TIMER_CYCLE;
      timer2_flag = 0;
     }

 void timer_run(){
    if(timer2_counter > 0){ // If timer2_counter > 0
        timer2_counter--; //   Decrease timer2_counter
   // If timer2_counter = 0, then timer2_flag = 1 to update LED MATRIX
        if(timer2_counter == 0) timer2_flag = 1;
    }
   }

const int MAX_LED_MATRIX = 8;
  int index_led_matrix = 0;

 // Array matrix_buffer[8] corresponds to 8 columns of LED MATRIX
  uint8_t matrix_buffer[8] = {0x00, 0xFC, 0xFE, 0x33, 0x33, 0xFE, 0xFC, 0x0};

  void clearMatrix(){
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_10
         |GPIO_PIN_11|GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15, 1);
  }
```

```
// The updateLEDMatrix function includes two input
   parameters
// index is the current column index
// index_run is the number of shifts left
void updateLEDMatrix(int index, int index_run){
// Set  and  clear B15 ->B8  while  preserving  the  state
    of all  other  pins in the  port
    GPIOB->BSRR = 0b1111111100000000; // Set B15 ->B8 (HIGH
   )

    // Matrix_buffer index of current column index =
   current column index + number of shifts left
    // If matrix_buffer index is greater than
   MAX_LED_MATRIX then subtract MAX_LED_MATRIX
    if(index + index_run >= MAX_LED_MATRIX){
          // Clear matrix_buffer[index] << 8 (LOW)
      GPIOB->BRR = matrix_buffer[index + index_run -
   MAX_LED_MATRIX] << 8 ;
    }
    else GPIOB->BRR = matrix_buffer[index + index_run] << 8
    ; // Clear matrix_buffer[index] << 8 (LOW)


    clearMatrix();// Remove  all the  column  of led
   matrix
      switch(index){
      case 0:
          // Turn on  column 1
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, 0);
        break;
      case 1:
          // Turn on  column 2
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3, 0);
        break ;
      case 2:
        // Turn on  column 3
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, 0);
        break ;
      case 3:
            // Turn on  column 4
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_11, 0);
        break ;
      case 4:
            // Turn on  column 5
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_12, 0);
        break ;
      case 5:
            // Turn on  column 6
```

```
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_13, 0);
        break;
      case 6:
            // Turn on  column 7
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_14, 0);
        break;
      case 7:
             // Turn on  column 8
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_15, 0);
        break;
      default:
        break;

      }
    }

setTimer2(2); // set time to update LED MATRIX is 2ms
int index_led_run = 0; // number of shifts left

while (1)
  {
    /* USER CODE END WHILE */

  if(timer2_flag == 1){
        // update  LED MATRIX
      updateLEDMatrix(index_led_matrix, index_led_run);

      index_led_matrix++;
      if(index_led_matrix >= MAX_LED_MATRIX){
          index_led_matrix = 0;

        // After displaying all 8 columns, increase the
   number of translations
        index_led_run ++;
        // If the number of translations exceeds
   MAX_LED_MATRIX, then set to 0
        if(index_led_run >= MAX_LED_MATRIX){
            index_led_run = 0;
        }

      }

      setTimer2(2);// set  again  2ms
    }

}
```

Program 1.18: Function to display data on LED Matrix