Ho Chi Minh city University of Technology
Computer Science and Engineering Faculty

Project – LAB2

Microcontroller – Microprocessor

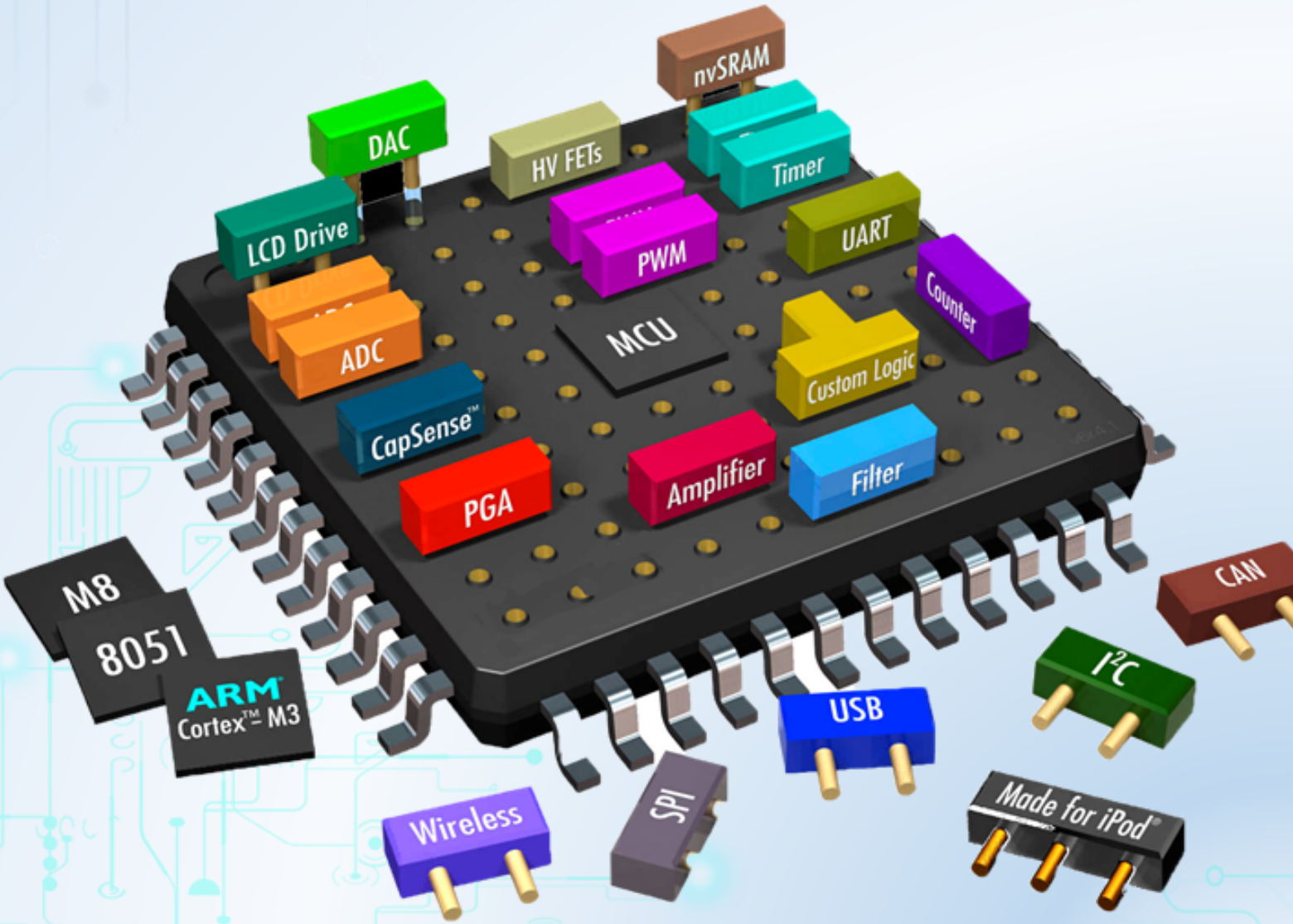
Instructor: Dr. Le Trong Nhan

Student: Nguyen Quoc Kiet - 1952802

Mục lục

CHƯƠNG 1

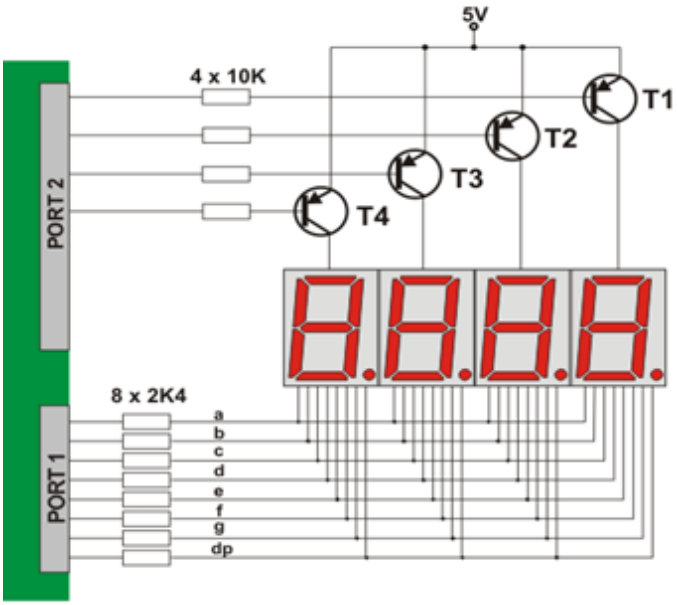
Timer Interrupt and LED Scanning



GitHub Repository URL and Student Information:
https://github.com/codeKietne/LAB2

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.



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

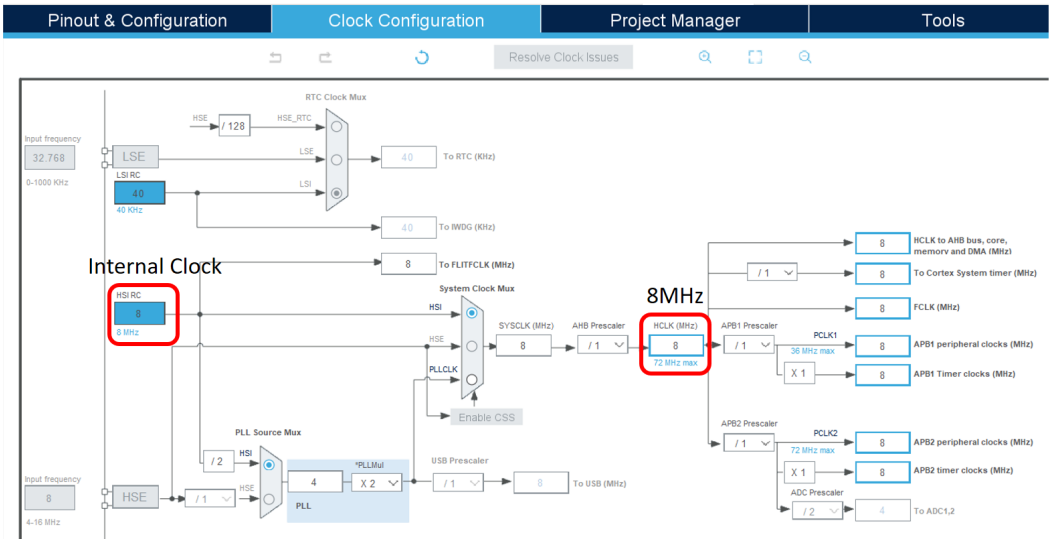
In the above diagram each seven segment display is having 8 internal LEDs, leading to the total number of LEDs is 32. However, not all the LEDs are required to turn ON, but one of them is needed. Therefore, only 12 lines are needed to control the whole 4 seven segment LEDs. T_S. Therfore, the period 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).

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

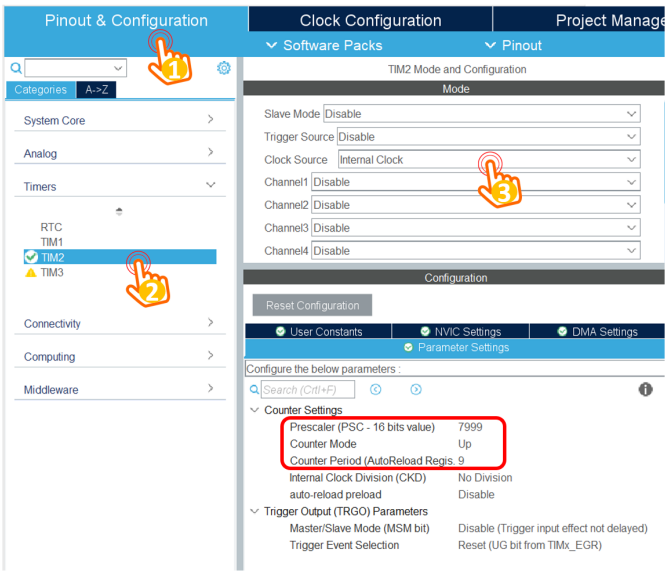
Step 1: Create a simple project, which LED connected to PA5. The manual can be found in the first lab.

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.



Hình 1.2: Default clock source for the system

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

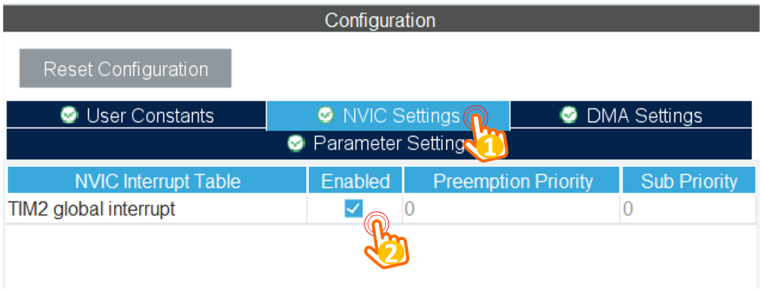


Hình 1.3: Configure for Timer 2

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:



Hình 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:

```
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 */g3

  while (1){

  }
}
```

Program 1.1: Init the timer interrupt in main

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 */
```

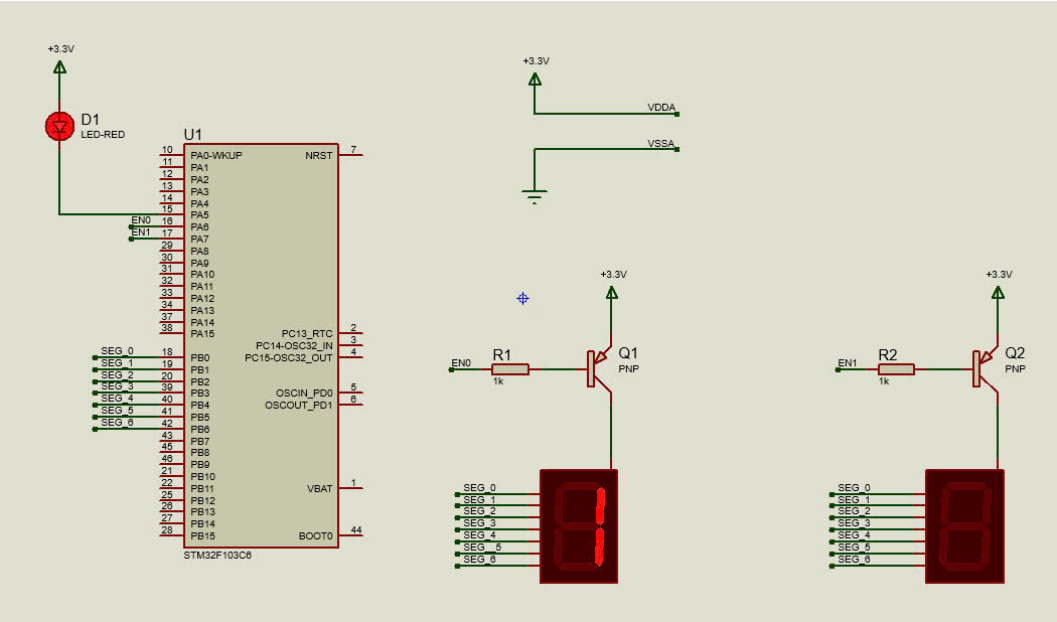
Program 1.3: LED Blinky using timer interrupt

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

Report 1: Implement the circuit simulation in Proteus with two 7-SEGMENT LEDs as following:



Hình 1.5: Simulation schematic in Proteus

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 2: Source code in the **HAL_TIM_PeriodElapsedCallback** function.

```
int counter = 50;
int enable = 0;
int counter1= 100;
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
  if(enable == 0)
  {
    counter--;
    HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, RESET);
    HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, SET);
    display7SEG(1);
    if(counter <= 0)
    {
      counter = 50;
      enable = 1;
    }
  }
```

```
18    else if(enable == 1)
19    {
20      counter--;
21      HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, SET);
22      HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, RESET);
23      display7SEG (2);
24      if(counter<= 0)
25      {
26        counter= 50;
27        enable = 0;
28      }
29
30      counter1--;
31      if(counter1 <= 0)
32      {
33        counter1 = 100;
34        HAL_GPIO_WritePin(LED_RED_GPIO_Port, LED_RED_Pin,
     RESET);
35      }
36    }
37  }
```

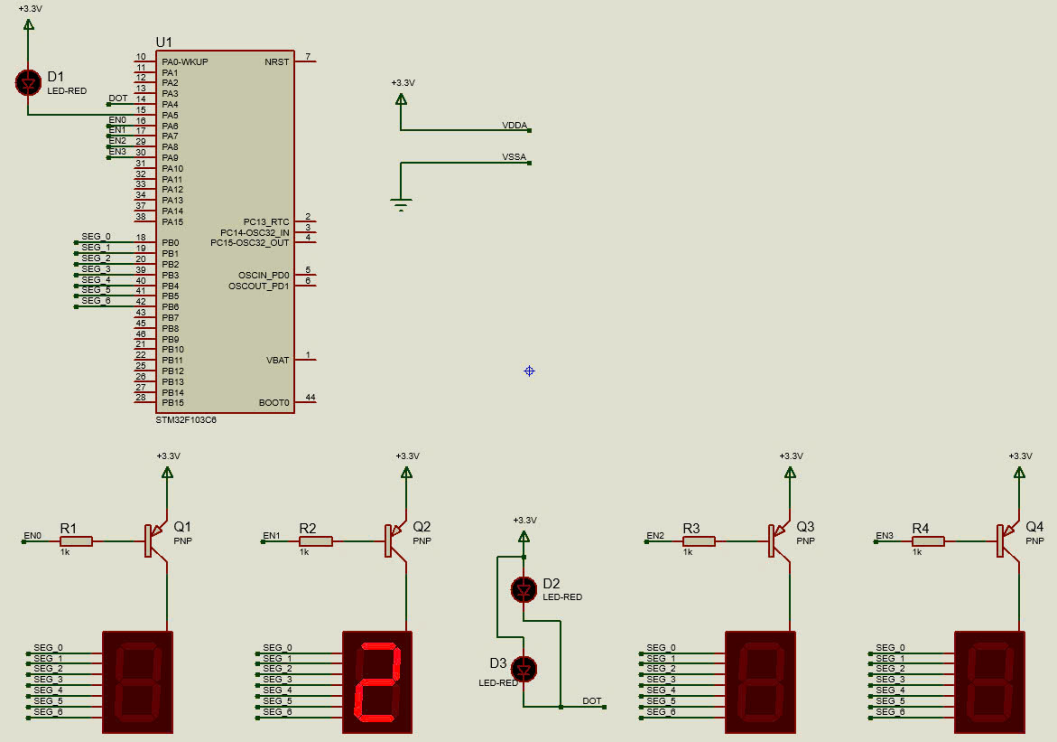
Program 1.4: Source code of Ex1

Short question: What is the frequency of the scanning process?
The working period of each 7-segment is 50ms. So, the total period is 2x500 = 1000ms. Scanning frequency is 1/1000ms = 1Hz.

3.2 Exercise 2

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

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: Schematic from Proteus and show in the report.



Hình 1.6: Simulation schematic in Proteus

Report 2: Source code in the **HAL_TIM_PeriodElapsedCallback** function.

```
int counter = 100;
int counter1 = 100;
int state = 0;
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
  if(state == 0)
  {
    counter--;
    HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, RESET);
    HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, SET);
    HAL_GPIO_WritePin(EN2_GPIO_Port, EN2_Pin, SET);
    HAL_GPIO_WritePin(EN3_GPIO_Port, EN3_Pin, SET);
    display7SEG(1);
    if(counter <= 0)
```

```
    {
      counter = 100;
      state = 1;
    }
  }
  else if (state == 1)
  {
    counter--;
    HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, SET);
    HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, RESET);
    HAL_GPIO_WritePin(EN2_GPIO_Port, EN2_Pin, SET);
    HAL_GPIO_WritePin(EN3_GPIO_Port, EN3_Pin, SET);
    display7SEG(2);
    if(counter <= 0)
    {
      counter = 100;
      state = 2;
    }
  }
  else if (state == 2)
  {
    counter--;
    HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, SET);
    HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, SET);
    HAL_GPIO_WritePin(EN2_GPIO_Port, EN2_Pin, RESET);
    HAL_GPIO_WritePin(EN3_GPIO_Port, EN3_Pin, SET);
    display7SEG(3);
    if(counter <= 0)
    {
      counter = 100;
      state = 3;
    }
  }
  else if (state == 3)
  {
    counter--;
    HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, SET);
    HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, SET);
    HAL_GPIO_WritePin(EN2_GPIO_Port, EN2_Pin, SET);
    HAL_GPIO_WritePin(EN3_GPIO_Port, EN3_Pin, RESET);
    display7SEG(4);
    if(counter <= 0)
    {
      counter = 100;
      state = 0;
    }
  }
    counter1--;
    if(counter1 <= 0)
```

```
64      {
65      counter1 = 100;
66      HAL_GPIO_TogglePin ( LED_RED_GPIO_Port , LED_RED_Pin );
67      HAL_GPIO_TogglePin ( DOT_GPIO_Port , DOT_Pin );
68      }
69  }
```

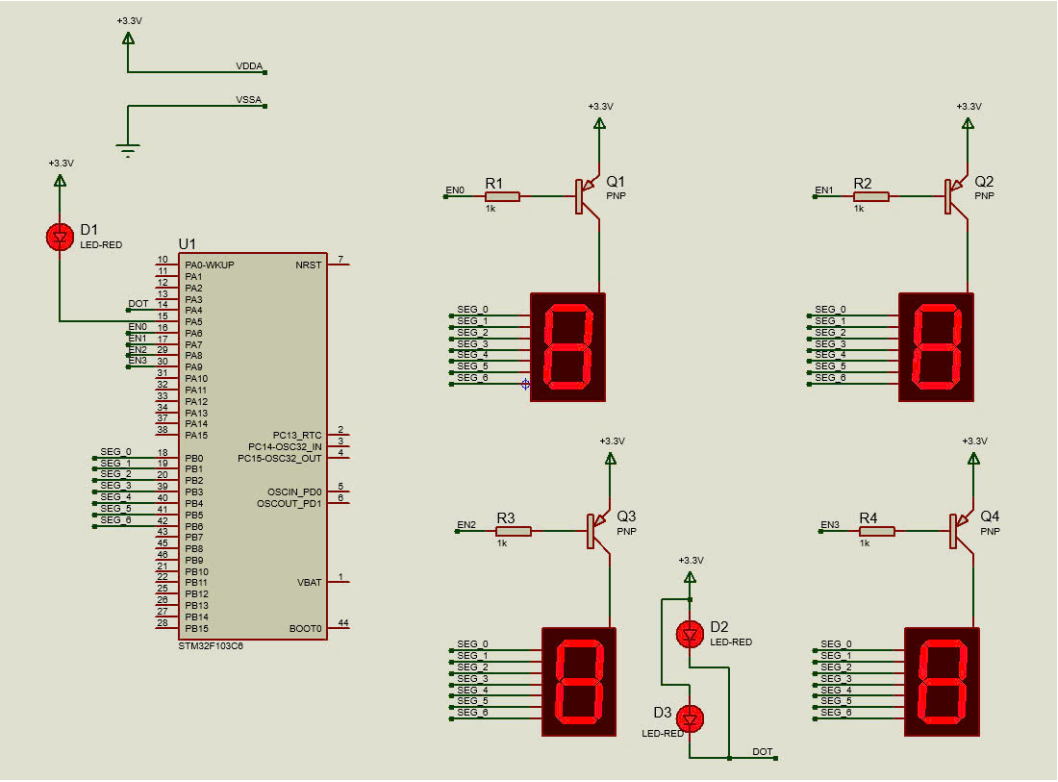
Program 1.5: Source code of Ex2

Short question: What is the frequency of the scanning process?
The working period of each 7-segment is 100ms. So, the total period is 4x1000 = 4000ms. Scanning frequency is 1/4000ms = 0.25Hz.

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:

Report 1: Schematic from Proteus and show in the report.



Hình 1.7: Simulation schematic in Proteus

Report 2: Source code in the **HAL_TIM_PeriodElapsedCallback** function.

```
int counter = 50;
int counter1 = 100;
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
  counter--;
  if(counter <= 0)
  {
    counter = 50;
    update7SEG(index_led++);
  }
    if(index_led > 3)
    {
      index_led = 0;
    }

    counter1--;
    if(counter1 <= 0)
    {
      counter1 = 100;
    HAL_GPIO_WritePin(LED_RED_GPIO_Port, LED_RED_Pin, RESET
   );
    HAL_GPIO_WritePin(DOT_GPIO_Port, DOT_Pin, RESET);
    }
}
```

Program 1.6: Source code of Ex3

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: Source code in the **HAL_TIM_PeriodElapsedCallback** function.

```
int counter = 25;
int counter1 = 100;
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
  counter--;
  if(counter <= 0)
  {
    counter = 25;
    update7SEG(index_led++);
  }
    if(index_led > 3)
```

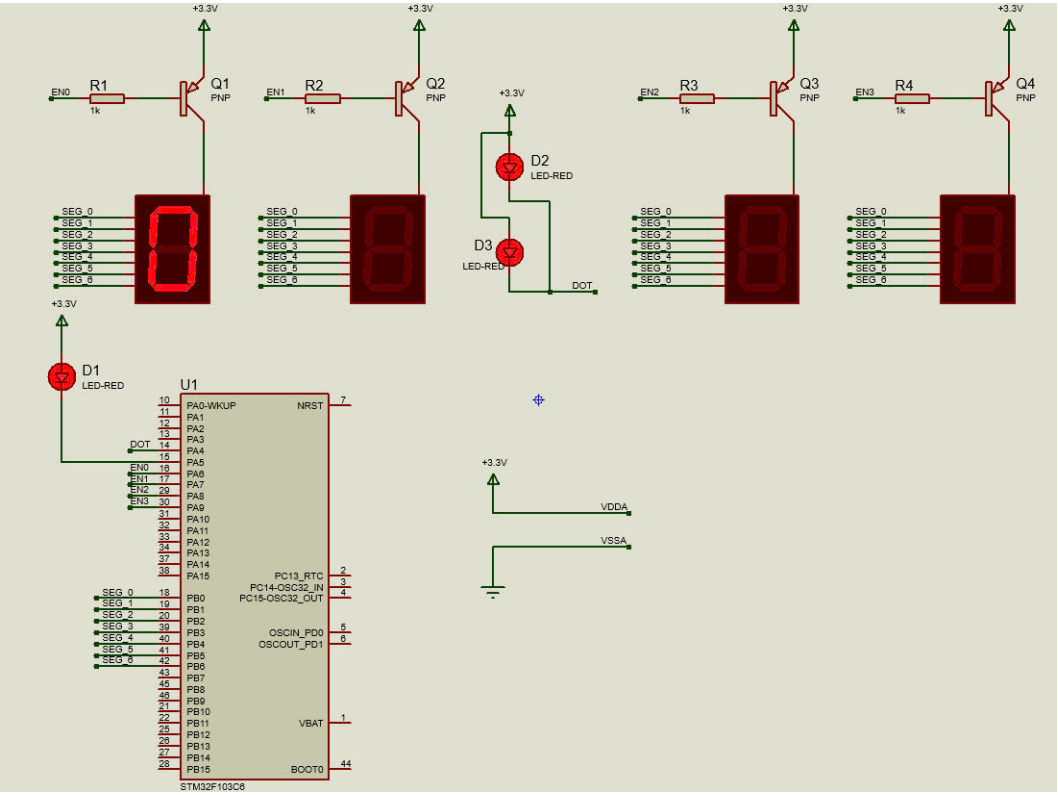
```
12      {
13        index_led = 0;
14      }
15
16      counter1--;
17      if(counter1 <= 0)
18      {
19        counter1 = 100;
20      HAL_GPIO_WritePin(LED_RED_GPIO_Port, LED_RED_Pin, RESET
        );
21      HAL_GPIO_WritePin(DOT_GPIO_Port, DOT_Pin, RESET);
22      }
23  }
```

Program 1.7: Source code of Ex4

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:

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



Hình 1.8: Simulation schematic in Proteus

Report 2: Source code in the **HAL_TIM_PeriodElapsedCallback** function.

```
void updateClockBuffer(int hour, int minute, int led_buffer
    [])
{
  if( hour >= 0 && hour <= 9)
  {
    led_buffer[0] = 0;
    led_buffer[1] = hour;
  }
  else
  {
    led_buffer[0] = hour / 10;
    led_buffer[1] = hour % 10;
  }

  if(minute >= 0 && minute <= 9)
  {
```

```
16      led_buffer[2] = 0;
17      led_buffer[3] = minute;
18    }
19    else
20    {
21      led_buffer[2] = minute / 10;
22      led_buffer[3] = minute % 10;
23    }
24  }
```

Program 1.8: Source code of Ex5

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.

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.**

Report 1: if in line 1 of the code above is miss, what happens after that and why?
Answer: There is not set time, the flag is not 1 because there is no information to set the flag

Report 2: if in line 1 of the code above is changed to setTimer0(1), what happens after that and why?
Answer: Set Timer = 1, it is too small compared to hardware timer = 10ms(1/10 = 0.1). So, it can not run. **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?
Answer: In this case, the flag is changed due to timer = 10 (10/10 = 1). So, it work

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: Source code in the while loop on main function.

```
int hour = 7, minute = 25, second = 20;
int counter = 0;
int led_buffer [4] = {1, 2, 3, 4};
setTimer0 (100);
setTimer1 (100);
while (1)
{
  if(timer0_flag == 1)
  {
    setTimer0(100);
    HAL_GPIO_TogglePin(LED_RED_GPIO_Port, LED_RED_Pin);
  }

 if(timer1_flag == 1)
 {
   setTimer1(100);
   second++;
   if(second >= 60)
   {
     minute++;
     second = 0;
   }
   if(minute >= 60)
   {
     hour++;
     minute = 0;
   }
   if(hour >= 24)
   {
     hour = 0;
   }

   HAL_GPIO_TogglePin(DOT_GPIO_Port, DOT_Pin);
   updateClockBuffer(hour, minute, led_buffer);
   update7SEG(counter, led_buffer);

   counter++;
   if(counter > 3)
   {
     counter = 0;
```

```
    }
  }
```

Program 1.9: Source code in Ex7

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: source code in the the main function. In the case more extra functions are used (e.g. the second software timer).

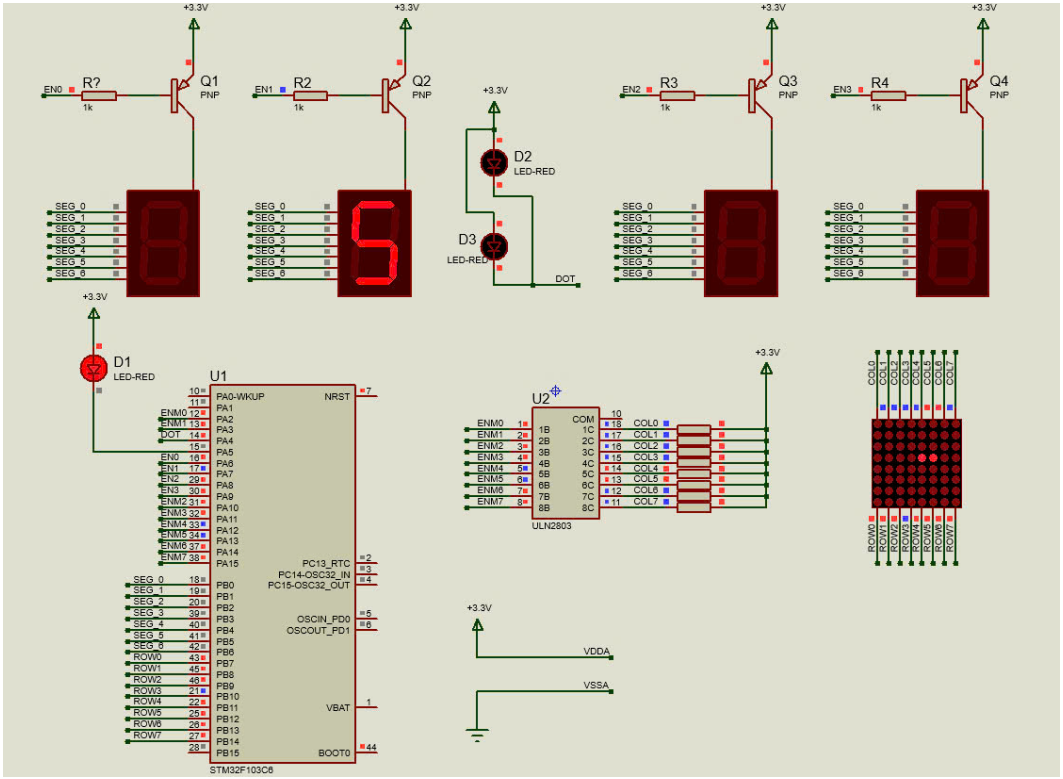
```
int hour = 18, minute = 40, second = 30;
int counter = 0;
int led_buffer [4] = {1, 2, 3, 4};
setTimer0 (100);
setTimer1 (100);
setTimer2 (200);

while (1)
{
  if(timer0_flag == 1)
  {
    setTimer0(100);
    HAL_GPIO_TogglePin(LED_RED_GPIO_Port, LED_RED_Pin);
  }

 if(timer1_flag == 1)
 {
    setTimer1(100);
    updateClockBuffer(hour, minute, led_buffer);
    second++;
    if(second >= 60)
    {
      minute++;
      second = 0;
    }
    if(minute >= 60)
    {
      hour++;
      minute = 0;
    }
    if(hour >= 24)
    {
      hour = 0;
    }
```

```
35      HAL_GPIO_TogglePin(DOT_GPIO_Port, DOT_Pin);
36     }
37
38    if(timer2_flag == 1)
39    {
40      setTimer2(200);
41      update7SEG(counter, led_buffer);
42
43      counter++;
44      if(counter > 3)
45      {
46        counter = 0;
47      }
48    }
49     /* USER CODE END WHILE */
50
51     /* USER CODE BEGIN 3 */
52   }
53   /* USER CODE END 3 */
54 }
```

Program 1.10: Source code in Ex8

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:
Report 1: Schematic of system in Proteus.



Hình 1.9: LED matrix is added to the simulation

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

```
void updateLEDMatrix(uint8_t index){
  setMatrix();
  switch (index){
  case 0:
    setCol(matrix_buffer[0]);
    HAL_GPIO_WritePin(ROW0_GPIO_Port, ROW0_Pin, RESET);
    break;
  case 1:
    setCol(matrix_buffer[1]);
    HAL_GPIO_WritePin(ROW1_GPIO_Port, ROW1_Pin, RESET);
    break;
  case 2:
    setCol(matrix_buffer[2]);
    HAL_GPIO_WritePin(ROW2_GPIO_Port, ROW2_Pin, RESET);
    break;
  case 3:
    setCol(matrix_buffer[3]);
```

```
    HAL_GPIO_WritePin(ROW3_GPIO_Port, ROW3_Pin, RESET);
    break;
  case 4:
    setCol(matrix_buffer[4]);
    HAL_GPIO_WritePin(ROW4_GPIO_Port, ROW4_Pin, RESET);
    break;
  case 5:
    setCol(matrix_buffer[5]);
    HAL_GPIO_WritePin(ROW5_GPIO_Port, ROW5_Pin, RESET);
    break;
  case 6:
    setCol(matrix_buffer[6]);
    HAL_GPIO_WritePin(ROW6_GPIO_Port, ROW6_Pin, RESET);
    break;
  case 7:
    setCol(matrix_buffer[7]);
    HAL_GPIO_WritePin(ROW7_GPIO_Port, ROW7_Pin, RESET);
    break;
  default:
    break;
  }
}
```

Program 1.11: Source code in Ex9

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 solution and present source code in the report.

```
#include "main.h"
#include "software_timer.h"
TIM_HandleTypeDef htim2;
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_TIM2_Init(void);
void display7SEG(int num);
const int MAX_LED = 4;
const int MAX_LED_MATRIX = 8;
int index_led = 0;
int index_led_matrix = 0;
int matrix_shift_flag = 0;
uint8_t shift = 0;
uint8_t matrix_buffer[8] = {0xE7,0xDB,0xBD,0x7E,0x00,0x00,0
   x7E,0x7E};
```

```
15 void update7SEG (int index, int led_buffer[])
16 {
17   switch ( index )
18   {
19   case 0:
20     HAL_GPIO_WritePin ( EN0_GPIO_Port , EN0_Pin ,
    GPIO_PIN_RESET );
21     HAL_GPIO_WritePin ( EN1_GPIO_Port , EN1_Pin ,
    GPIO_PIN_SET );
22     HAL_GPIO_WritePin ( EN2_GPIO_Port , EN2_Pin ,
    GPIO_PIN_SET );
23     HAL_GPIO_WritePin ( EN3_GPIO_Port , EN3_Pin ,
    GPIO_PIN_SET );
24     display7SEG ( led_buffer [0]) ;
25     break;
26   case 1:
27     HAL_GPIO_WritePin ( EN0_GPIO_Port , EN0_Pin ,
    GPIO_PIN_SET );
28     HAL_GPIO_WritePin ( EN1_GPIO_Port , EN1_Pin ,
    GPIO_PIN_RESET );
29     HAL_GPIO_WritePin ( EN2_GPIO_Port , EN2_Pin ,
    GPIO_PIN_SET );
30     HAL_GPIO_WritePin ( EN3_GPIO_Port , EN3_Pin ,
    GPIO_PIN_SET );
31     display7SEG ( led_buffer [1]) ;
32     break;
33   case 2:
34     HAL_GPIO_WritePin ( EN0_GPIO_Port , EN0_Pin ,
    GPIO_PIN_SET );
35     HAL_GPIO_WritePin ( EN1_GPIO_Port , EN1_Pin ,
    GPIO_PIN_SET );
36     HAL_GPIO_WritePin ( EN2_GPIO_Port , EN2_Pin ,
    GPIO_PIN_RESET );
37     HAL_GPIO_WritePin ( EN3_GPIO_Port , EN3_Pin ,
    GPIO_PIN_SET );
38     display7SEG ( led_buffer [2]) ;
39     break;
40   case 3:
41     HAL_GPIO_WritePin ( EN0_GPIO_Port , EN0_Pin ,
    GPIO_PIN_SET );
42     HAL_GPIO_WritePin ( EN1_GPIO_Port , EN1_Pin ,
    GPIO_PIN_SET );
43     HAL_GPIO_WritePin ( EN2_GPIO_Port , EN2_Pin ,
    GPIO_PIN_SET );
44     HAL_GPIO_WritePin ( EN3_GPIO_Port , EN3_Pin ,
    GPIO_PIN_RESET );
45     display7SEG ( led_buffer [3]) ;
46     break;
47   default:
```

```
48      break;
49    }
50  }
51  //int led_buffer [4] = {0, 0, 0, 0};
52  int hour = 15, minute = 25, second = 50;
53  void updateClockBuffer ( int hour, int minute, int
       led_buffer[])
54  {
55    if ( hour >= 0 && hour <= 9)
56      {
57        led_buffer [1] = hour;
58        led_buffer [0] = 0;
59      }
60      else
61      {
62        led_buffer [1] = hour % 10;
63        led_buffer [0] = hour / 10;
64      }
65    if (minute >= 0 && minute <= 9)
66    {
67      led_buffer [3] = minute;
68      led_buffer [2] = 0;
69    }
70    else
71    {
72      led_buffer [3] = minute % 10;
73      led_buffer [2] = minute / 10;
74    }
75  }
76  void setCol(uint8_t val)
77  {
78    HAL_GPIO_WritePin(ENM0_GPIO_Port, ENM0_Pin, ((val>>7)&0
       x01));
79    HAL_GPIO_WritePin(ENM1_GPIO_Port, ENM1_Pin, ((val>>6)&0
       x01));
80    HAL_GPIO_WritePin(ENM2_GPIO_Port, ENM2_Pin, ((val>>5)&0
       x01));
81    HAL_GPIO_WritePin(ENM3_GPIO_Port, ENM3_Pin, ((val>>4)&0
       x01));
82    HAL_GPIO_WritePin(ENM4_GPIO_Port, ENM4_Pin, ((val>>3)&0
       x01));
83    HAL_GPIO_WritePin(ENM5_GPIO_Port, ENM5_Pin, ((val>>2)&0
       x01));
84    HAL_GPIO_WritePin(ENM6_GPIO_Port, ENM6_Pin, ((val>>1)&0
       x01));
85    HAL_GPIO_WritePin(ENM7_GPIO_Port, ENM7_Pin, ((val>>0)&0
       x01));
86  }
87  void setMatrix(void)
```

```
{
  HAL_GPIO_WritePin(ROW0_GPIO_Port, ROW0_Pin, SET);
  HAL_GPIO_WritePin(ROW1_GPIO_Port, ROW1_Pin, SET);
  HAL_GPIO_WritePin(ROW2_GPIO_Port, ROW2_Pin, SET);
  HAL_GPIO_WritePin(ROW3_GPIO_Port, ROW3_Pin, SET);
  HAL_GPIO_WritePin(ROW4_GPIO_Port, ROW4_Pin, SET);
  HAL_GPIO_WritePin(ROW5_GPIO_Port, ROW5_Pin, SET);
  HAL_GPIO_WritePin(ROW6_GPIO_Port, ROW6_Pin, SET);
  HAL_GPIO_WritePin(ROW7_GPIO_Port, ROW7_Pin, SET);
}
void updateLEDMatrix(uint8_t index, uint8_t shift)
{
  setMatrix();
  uint8_t matrix_buffer_shift = (matrix_buffer[index] <<
   shift) | (matrix_buffer[index] >> (8-shift));
  switch (index)
  {
  case 0:
    setCol(matrix_buffer_shift);
    HAL_GPIO_WritePin(ROW0_GPIO_Port, ROW0_Pin, RESET);
    break;
  case 1:
    setCol(matrix_buffer_shift);
    HAL_GPIO_WritePin(ROW1_GPIO_Port, ROW1_Pin, RESET);
    break;
  case 2:
    setCol(matrix_buffer_shift);
    HAL_GPIO_WritePin(ROW2_GPIO_Port, ROW2_Pin, RESET);
    break;
  case 3:
    setCol(matrix_buffer_shift);
    HAL_GPIO_WritePin(ROW3_GPIO_Port, ROW3_Pin, RESET);
    break;
  case 4:
    setCol(matrix_buffer_shift);
    HAL_GPIO_WritePin(ROW4_GPIO_Port, ROW4_Pin, RESET);
    break;
  case 5:
    setCol(matrix_buffer_shift);
    HAL_GPIO_WritePin(ROW5_GPIO_Port, ROW5_Pin, RESET);
    break;
  case 6:
    setCol(matrix_buffer_shift);
    HAL_GPIO_WritePin(ROW6_GPIO_Port, ROW6_Pin, RESET);
    break;
  case 7:
    setCol(matrix_buffer_shift);
    HAL_GPIO_WritePin(ROW7_GPIO_Port, ROW7_Pin, RESET);
    break;
```

```
136   default:
137     break;
138   }
139 }
140 int main(void)
141 {
142   HAL_Init();
143   SystemClock_Config();
144   MX_GPIO_Init();
145   MX_TIM2_Init();
146   HAL_TIM_Base_Start_IT (& htim2 );
147   /* USER CODE BEGIN WHILE */
148   int led_buffer [4] = {1, 2, 3, 4};
149   setTimer0 (500) ;
150   setTimer1 (250) ;
151   setTimer2 (200) ;
152   setTimer3 (1000) ;
153   while (1)
154   {
155     // LED BLINK AND DOT
156     if(timer0_flag == 1)
157     {
158     setTimer0(500);
159     HAL_GPIO_TogglePin(LED_RED_GPIO_Port, LED_RED_Pin);
160     }
161     // UPDATE TIME
162     if(timer1_flag == 1)
163     {
164       setTimer1(250);
165       updateClockBuffer(hour,minute,led_buffer);
166       second++;
167       if (second >= 60)
168       {
169         second = 0;
170         minute++;
171       }
172       if(minute >= 60)
173       {
174         minute = 0;
175         hour++;
176       }
177       if(hour >= 24)
178       {
179         hour = 0;
180       }
181       HAL_GPIO_TogglePin(DOT_GPIO_Port, DOT_Pin);
182     }
183     // UPDATE LED7SEG & LED MATRIX
184     if(timer2_flag == 1)
```

```
    {
    setTimer2(200);
    update7SEG(index_led, led_buffer);
    index_led++;
    if(index_led >= 4)
     index_led = 0;
    }
    if(timer3_flag == 1)
    {
    setTimer3(1000);
    updateLEDMatrix(index_led_matrix, shift);
    index_led_matrix++;
    if(index_led_matrix >= 8)
    {
    index_led_matrix = 0;
    shift ++;
    if(shift >= 8) shift = 0;
    }
    }
  }
}
void display7SEG(int num)
{
    if (num == 0)
    {
        HAL_GPIO_WritePin(a_GPIO_Port, a_Pin, RESET);
        HAL_GPIO_WritePin(b_GPIO_Port, b_Pin, RESET);
        HAL_GPIO_WritePin(c_GPIO_Port, c_Pin, RESET);
        HAL_GPIO_WritePin(d_GPIO_Port, d_Pin, RESET);
        HAL_GPIO_WritePin(e_GPIO_Port, e_Pin, RESET);
        HAL_GPIO_WritePin(f_GPIO_Port, f_Pin, RESET);
        HAL_GPIO_WritePin(g_GPIO_Port, g_Pin, SET);
    }
    else if (num == 1)
    {
        HAL_GPIO_WritePin(a_GPIO_Port, a_Pin, SET);
        HAL_GPIO_WritePin(b_GPIO_Port, b_Pin, RESET);
        HAL_GPIO_WritePin(c_GPIO_Port, c_Pin, RESET);
        HAL_GPIO_WritePin(d_GPIO_Port, d_Pin, SET);
        HAL_GPIO_WritePin(e_GPIO_Port, e_Pin, SET);
        HAL_GPIO_WritePin(f_GPIO_Port, f_Pin, SET);
        HAL_GPIO_WritePin(g_GPIO_Port, g_Pin, SET);
    }
    else if (num == 2)
    {
        HAL_GPIO_WritePin(a_GPIO_Port, a_Pin, RESET);
        HAL_GPIO_WritePin(b_GPIO_Port, b_Pin, RESET);
        HAL_GPIO_WritePin(c_GPIO_Port, c_Pin, SET);
        HAL_GPIO_WritePin(d_GPIO_Port, d_Pin, RESET);
```

```
234         HAL_GPIO_WritePin(e_GPIO_Port, e_Pin, RESET);
235         HAL_GPIO_WritePin(f_GPIO_Port, f_Pin, SET);
236         HAL_GPIO_WritePin(g_GPIO_Port, g_Pin, RESET);
237     }
238     else if (num == 3)
239     {
240         HAL_GPIO_WritePin(a_GPIO_Port, a_Pin, RESET);
241         HAL_GPIO_WritePin(b_GPIO_Port, b_Pin, RESET);
242         HAL_GPIO_WritePin(c_GPIO_Port, c_Pin, RESET);
243         HAL_GPIO_WritePin(d_GPIO_Port, d_Pin, RESET);
244         HAL_GPIO_WritePin(e_GPIO_Port, e_Pin, SET);
245         HAL_GPIO_WritePin(f_GPIO_Port, f_Pin, SET);
246         HAL_GPIO_WritePin(g_GPIO_Port, g_Pin, RESET);
247     }
248 
249     else if (num == 4)
250     {
251         HAL_GPIO_WritePin(a_GPIO_Port, a_Pin, SET);
252         HAL_GPIO_WritePin(b_GPIO_Port, b_Pin, RESET);
253         HAL_GPIO_WritePin(c_GPIO_Port, c_Pin, RESET);
254         HAL_GPIO_WritePin(d_GPIO_Port, d_Pin, SET);
255         HAL_GPIO_WritePin(e_GPIO_Port, e_Pin, SET);
256         HAL_GPIO_WritePin(f_GPIO_Port, f_Pin, RESET);
257         HAL_GPIO_WritePin(g_GPIO_Port, g_Pin, RESET);
258     }
259     else if (num == 5)
260     {
261         HAL_GPIO_WritePin(a_GPIO_Port, a_Pin, RESET);
262         HAL_GPIO_WritePin(b_GPIO_Port, b_Pin, SET);
263         HAL_GPIO_WritePin(c_GPIO_Port, c_Pin, RESET);
264         HAL_GPIO_WritePin(d_GPIO_Port, d_Pin, RESET);
265         HAL_GPIO_WritePin(e_GPIO_Port, e_Pin, SET);
266         HAL_GPIO_WritePin(f_GPIO_Port, f_Pin, RESET);
267         HAL_GPIO_WritePin(g_GPIO_Port, g_Pin, RESET);
268     }
269     else if (num == 6)
270     {
271         HAL_GPIO_WritePin(a_GPIO_Port, a_Pin, RESET);
272         HAL_GPIO_WritePin(b_GPIO_Port, b_Pin, SET);
273         HAL_GPIO_WritePin(c_GPIO_Port, c_Pin, RESET);
274         HAL_GPIO_WritePin(d_GPIO_Port, d_Pin, RESET);
275         HAL_GPIO_WritePin(e_GPIO_Port, e_Pin, RESET);
276         HAL_GPIO_WritePin(f_GPIO_Port, f_Pin, RESET);
277         HAL_GPIO_WritePin(g_GPIO_Port, g_Pin, RESET);
278     }
279     else if (num == 7)
280     {
281         HAL_GPIO_WritePin(a_GPIO_Port, a_Pin, RESET);
282         HAL_GPIO_WritePin(b_GPIO_Port, b_Pin, RESET);
```

```
283         HAL_GPIO_WritePin(c_GPIO_Port, c_Pin, RESET);
284         HAL_GPIO_WritePin(d_GPIO_Port, d_Pin, SET);
285         HAL_GPIO_WritePin(e_GPIO_Port, e_Pin, SET);
286         HAL_GPIO_WritePin(f_GPIO_Port, f_Pin, SET);
287         HAL_GPIO_WritePin(g_GPIO_Port, g_Pin, SET);
288     }
289 
290     else if (num == 8)
291     {
292         HAL_GPIO_WritePin(a_GPIO_Port, a_Pin, RESET);
293         HAL_GPIO_WritePin(b_GPIO_Port, b_Pin, RESET);
294         HAL_GPIO_WritePin(c_GPIO_Port, c_Pin, RESET);
295         HAL_GPIO_WritePin(d_GPIO_Port, d_Pin, RESET);
296         HAL_GPIO_WritePin(e_GPIO_Port, e_Pin, RESET);
297         HAL_GPIO_WritePin(f_GPIO_Port, f_Pin, RESET);
298         HAL_GPIO_WritePin(g_GPIO_Port, g_Pin, RESET);
299     }
300     else if (num == 9)
301     {
302         HAL_GPIO_WritePin(a_GPIO_Port, a_Pin, RESET);
303         HAL_GPIO_WritePin(b_GPIO_Port, b_Pin, RESET);
304         HAL_GPIO_WritePin(c_GPIO_Port, c_Pin, RESET);
305         HAL_GPIO_WritePin(d_GPIO_Port, d_Pin, RESET);
306         HAL_GPIO_WritePin(e_GPIO_Port, e_Pin, SET);
307         HAL_GPIO_WritePin(f_GPIO_Port, f_Pin, RESET);
308         HAL_GPIO_WritePin(g_GPIO_Port, g_Pin, RESET);
309     }
310 }
311 void SystemClock_Config(void)
312 {
313   RCC_OscInitTypeDef RCC_OscInitStruct = {0};
314   RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
315   RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI
      ;
316   RCC_OscInitStruct.HSIState = RCC_HSI_ON;
317   RCC_OscInitStruct.HSICalibrationValue =
      RCC_HSICALIBRATION_DEFAULT;
318   RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
319   if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
320   {
321     Error_Handler();
322   }
323   RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|
      RCC_CLOCKTYPE_SYSCLK
324                                 |RCC_CLOCKTYPE_PCLK1|
      RCC_CLOCKTYPE_PCLK2;
325   RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HSI;
326   RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
327   RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
```

```
328   RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
329
330   if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct,
       FLASH_LATENCY_0) != HAL_OK)
331   {
332     Error_Handler();
333   }
334 }
335 static void MX_TIM2_Init(void)
336 {
337   TIM_ClockConfigTypeDef sClockSourceConfig = {0};
338   TIM_MasterConfigTypeDef sMasterConfig = {0};
339   htim2.Instance = TIM2;
340   htim2.Init.Prescaler = 7999;
341   htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
342   htim2.Init.Period = 9;
343   htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
344   htim2.Init.AutoReloadPreload =
       TIM_AUTORELOAD_PRELOAD_DISABLE;
345   if (HAL_TIM_Base_Init(&htim2) != HAL_OK)
346   {
347     Error_Handler();
348   }
349   sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL
       ;
350   if (HAL_TIM_ConfigClockSource(&htim2, &sClockSourceConfig
       ) != HAL_OK)
351   {
352     Error_Handler();
353   }
354   sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
355   sMasterConfig.MasterSlaveMode =
       TIM_MASTERSLAVEMODE_DISABLE;
356   if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &
       sMasterConfig) != HAL_OK)
357   {
358     Error_Handler();
359   }
360 }
361 static void MX_GPIO_Init(void)
362 {
363   GPIO_InitTypeDef GPIO_InitStruct = {0};
364   __HAL_RCC_GPIOA_CLK_ENABLE();
365   __HAL_RCC_GPIOB_CLK_ENABLE();
366   HAL_GPIO_WritePin(GPIOA, ENM0_Pin|ENM1_Pin|DOT_Pin|
       LED_RED_Pin
367                           |EN0_Pin|EN1_Pin|EN2_Pin|EN3_Pin
368                           |ENM2_Pin|ENM3_Pin|ENM4_Pin|
       ENM5_Pin
```

```
369                             |ENM6_Pin|ENM7_Pin,
    GPIO_PIN_RESET);
370   HAL_GPIO_WritePin(GPIOB, a_Pin|b_Pin|c_Pin|ROW3_Pin
371                             |ROW4_Pin|ROW5_Pin|ROW6_Pin|
    ROW7_Pin
372                             |d_Pin|e_Pin|f_Pin|g_Pin
373                             |ROW0_Pin|ROW1_Pin|ROW2_Pin,
    GPIO_PIN_RESET);
374
375   /*Configure GPIO pins : ENM0_Pin ENM1_Pin DOT_Pin
    LED_RED_Pin
376                              EN0_Pin EN1_Pin EN2_Pin EN3_Pin
377                              ENM2_Pin ENM3_Pin ENM4_Pin
    ENM5_Pin
378                              ENM6_Pin ENM7_Pin */
379   GPIO_InitStruct.Pin = ENM0_Pin|ENM1_Pin|DOT_Pin|
    LED_RED_Pin
380                             |EN0_Pin|EN1_Pin|EN2_Pin|EN3_Pin
381                             |ENM2_Pin|ENM3_Pin|ENM4_Pin|
    ENM5_Pin
382                             |ENM6_Pin|ENM7_Pin;
383   GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
384   GPIO_InitStruct.Pull = GPIO_NOPULL;
385   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
386   HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
387
388   /*Configure GPIO pins : a_Pin b_Pin c_Pin ROW3_Pin
389                              ROW4_Pin ROW5_Pin ROW6_Pin
    ROW7_Pin
390                              d_Pin e_Pin f_Pin g_Pin
391                              ROW0_Pin ROW1_Pin ROW2_Pin */
392   GPIO_InitStruct.Pin = a_Pin|b_Pin|c_Pin|ROW3_Pin
393                             |ROW4_Pin|ROW5_Pin|ROW6_Pin|
    ROW7_Pin
394                             |d_Pin|e_Pin|f_Pin|g_Pin
395                             |ROW0_Pin|ROW1_Pin|ROW2_Pin;
396   GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
397   GPIO_InitStruct.Pull = GPIO_NOPULL;
398   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
399   HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
400
401 }
402 void HAL_TIM_PeriodElapsedCallback ( TIM_HandleTypeDef *
    htim )
403 {
404   HAL_GPIO_WritePin(LED_RED_GPIO_Port, LED_RED_Pin,RESET);
405   timerRun0();
406   timerRun1();
407   timerRun2();
```

```
408   timerRun3();
409 }
410 void Error_Handler(void)
411 {
412   __disable_irq();
413   while (1)
414   {
415   }
416 }
417 #ifdef  USE_FULL_ASSERT
418 void assert_failed(uint8_t *file, uint32_t line)
419 {
420 }
421 #endif /* USE_FULL_ASSERT */
```

Program 1.12: Source code in Ex10