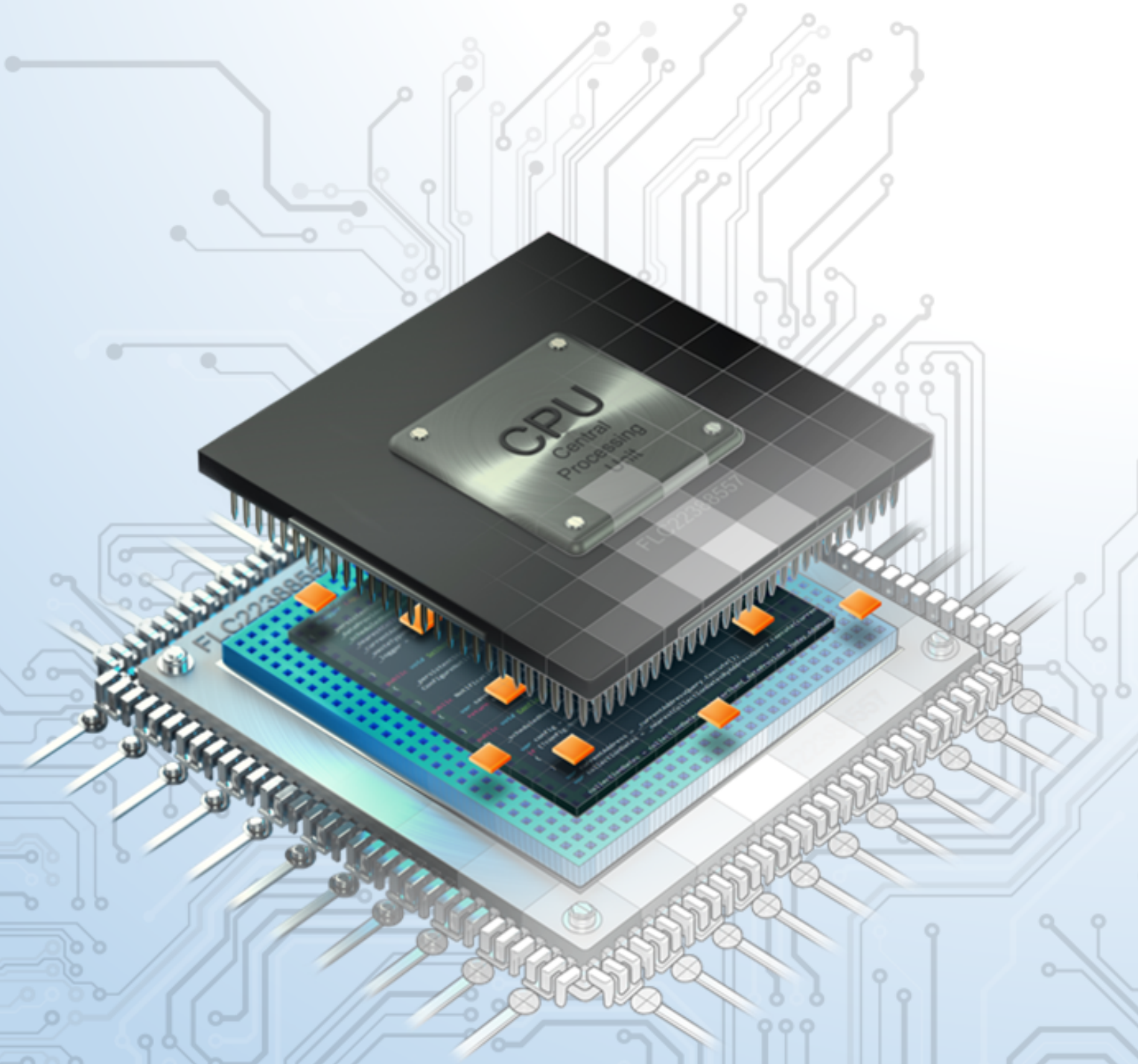
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
COMPUTER ENGINEERING

Microcontroller

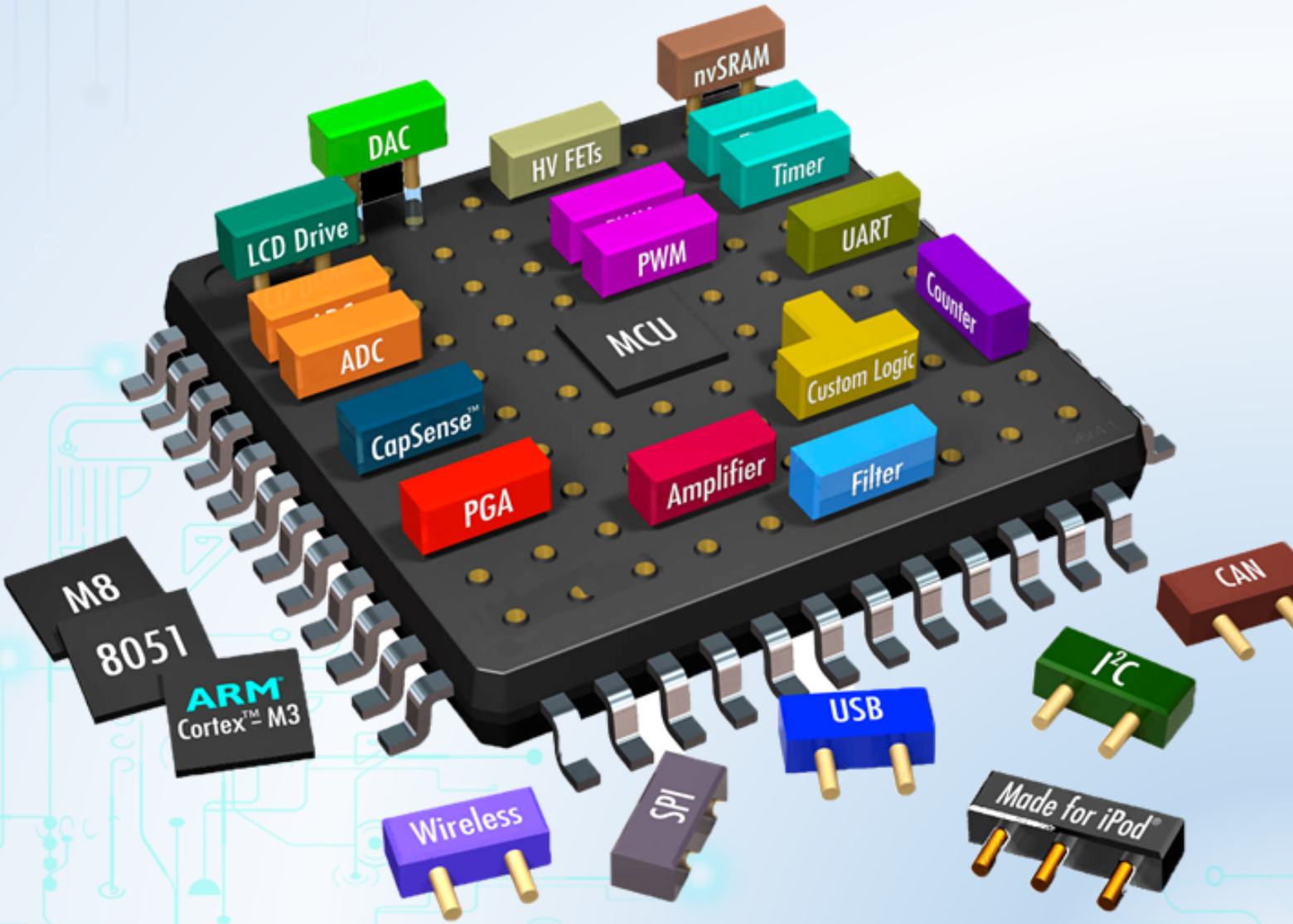


Dr. Le Trong Nhan

Mục lục

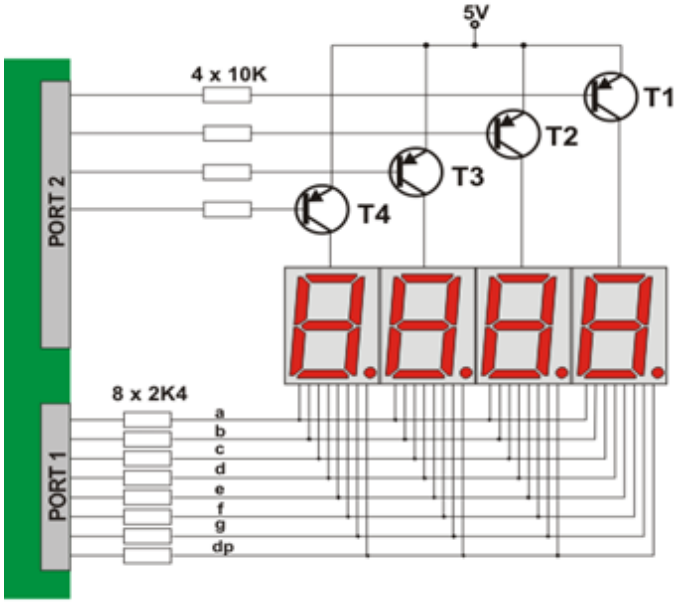
CHƯƠNG 1

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.



Hình 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.

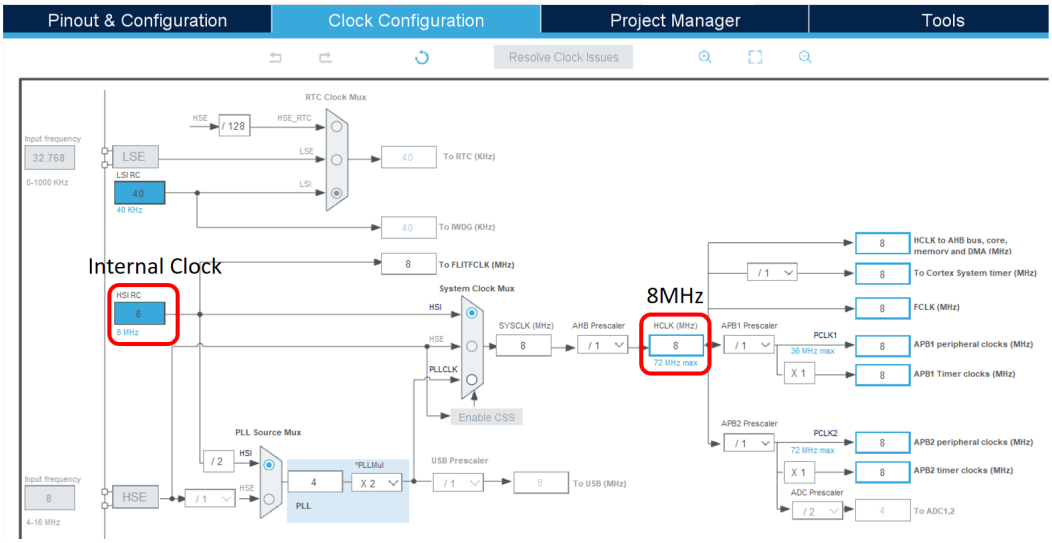
In this manual, the timer interrupt is used to design the interval T_S for LED scanning. Unfortunately, the simulation on Proteus can not execute at high frequency, the frequency f is set to a low value (e.g. 1Hz). In a real implementation, this fre-

quency should be 50Hz.

2 Timer Interrupt Setup

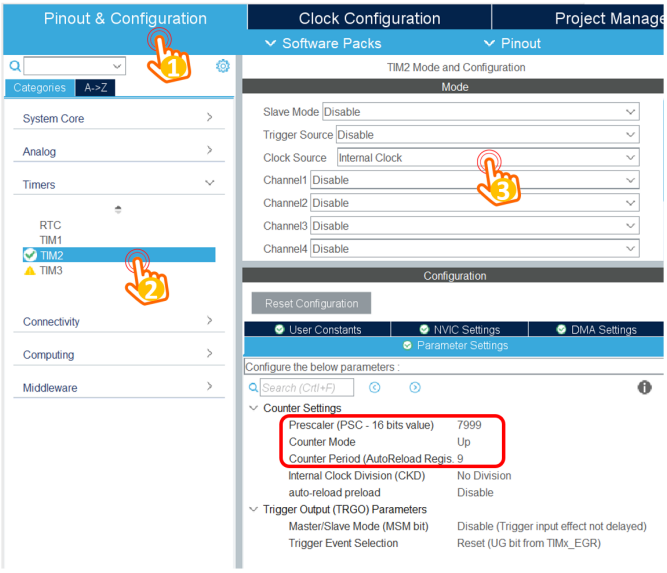
Step 1: Create a simple project, which LED connected to PA5. The manual can be found in the first lab.

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:

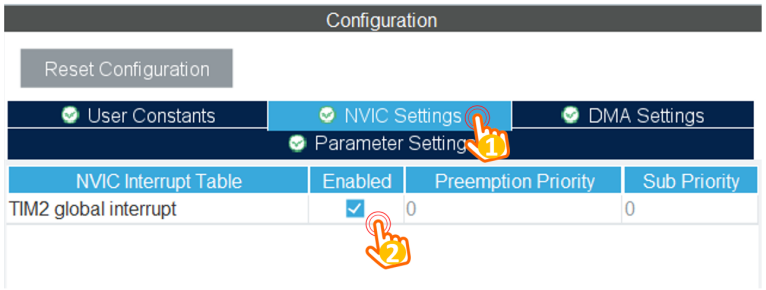


Hình 1.3: Configure for Timer 2

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

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.

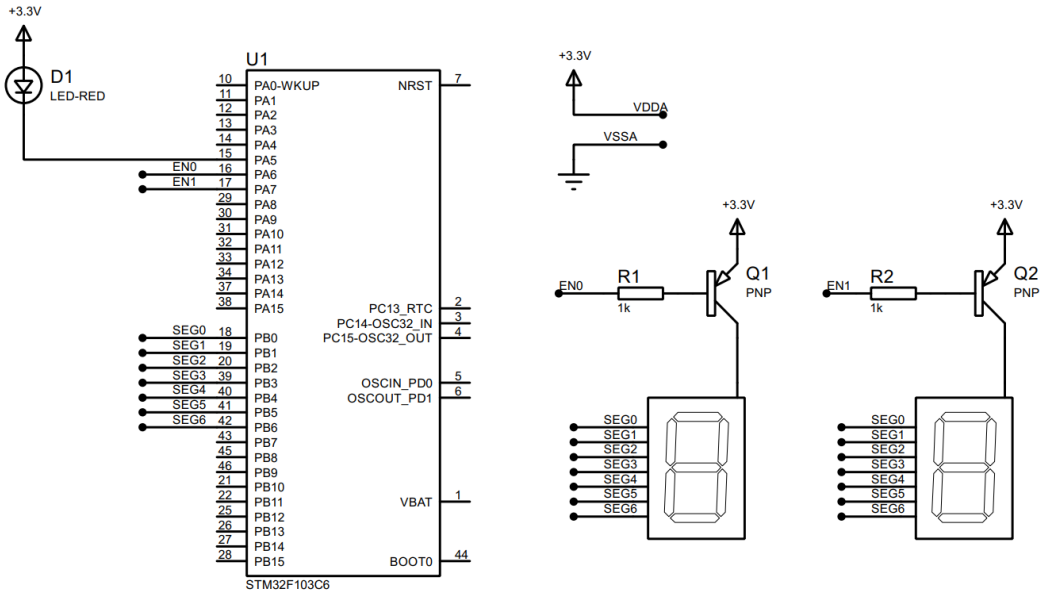
3 Exercise and Report

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:



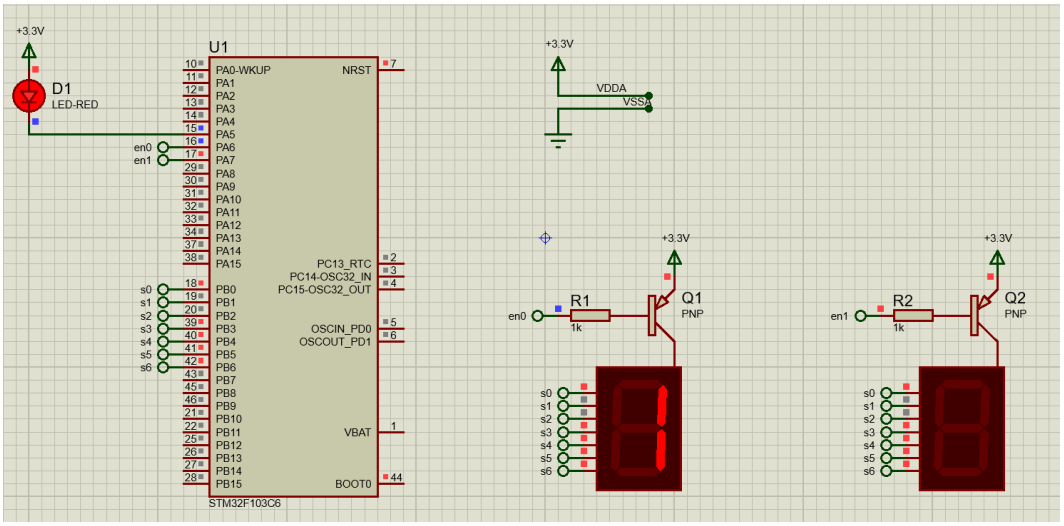
Hình 1.5: *Simulation schematic in Proteus*

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.



Hình 1.6: Simulation schematic in Proteus ex1

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

```
int counter = 50;
//The 'which_led' variable allows a switch between seven-segment LEDs
int which_led = 0;
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
  counter--;
  if(counter <= 0) {
    counter = 50;
    switch(which_led) {
    case 0:
      which_led = 1;
      //Turn on the 1st seven-segment LED. Meanwhile, turn off the remaining seven-segment LED
      HAL_GPIO_WritePin(GPIOA, en0_Pin, RESET);
      HAL_GPIO_WritePin(GPIOA, en1_Pin, SET);
      //display number '1' at 1st seven segment LED
      display7SEG(1);
      break;
    case 1:
      which_led = 0;
```

```
19        //Turn on the 2nd seven-segment LED. Meanwhile, turn
      off the remaining seven-segment LED
20        HAL_GPIO_WritePin(GPIOA, en0_Pin, SET);
21        HAL_GPIO_WritePin(GPIOA, en1_Pin, RESET);
22        //display number '2' at 2nd seven segment LED
23        display7SEG(2);
24        break;
25      default:
26        break;
27      }
28    }
29  }
```

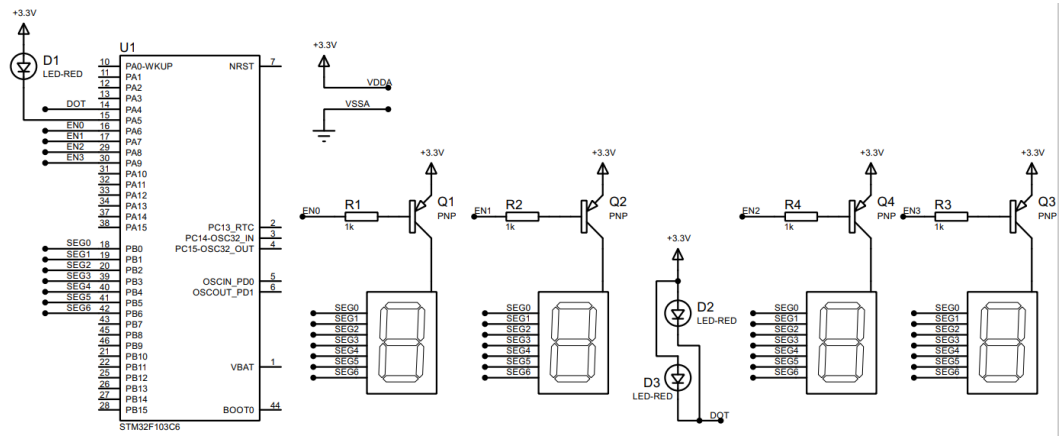
Program 1.4: Code for ex1

Short question: What is the frequency of the scanning process?

Thời gian chuyển đổi giữa từng LED 7 đoạn là nửa giây (500ms). Ta có 2 thanh LED 7 đoạn. Như vậy, tính được chu kỳ là 2*500ms = 1s. Tần số của quá trình quét 2 thanh LED 7 đoạn sẽ là 1/1s = 1Hz.

3.2 Exercise 2

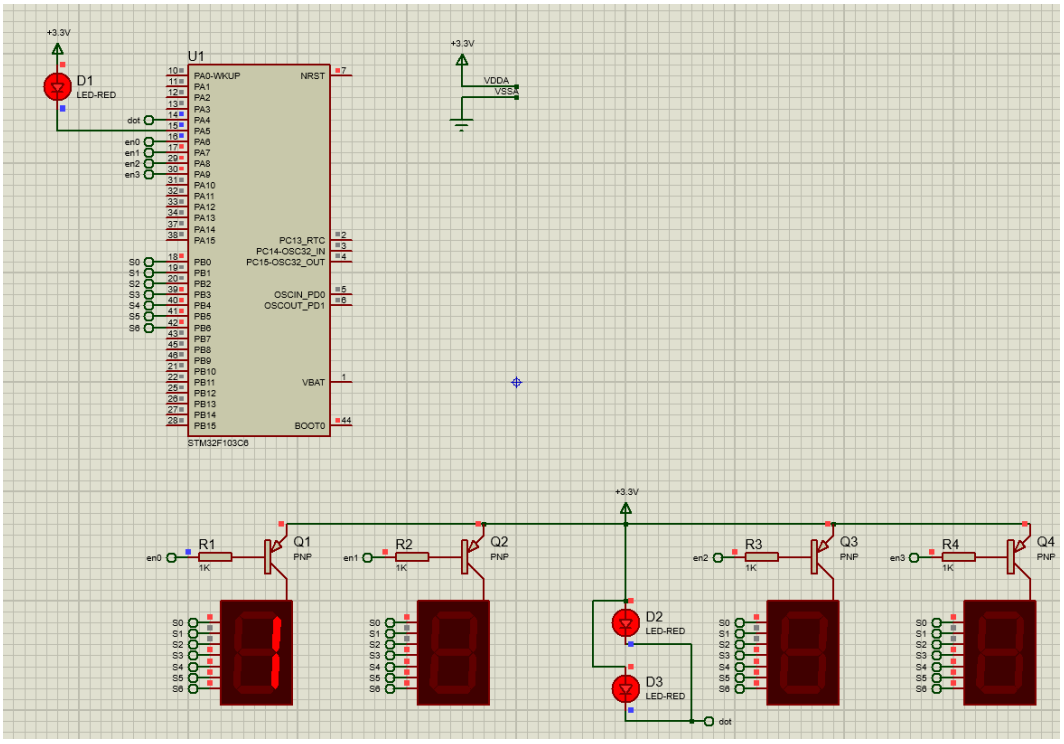
Extend to 4 seven segment LEDs and two LEDs (connected to PA4, labeled as **DOT**) in the middle as following:



Hình 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.



Hình 1.8: Simulation schematic in Proteus ex2

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

```
int counter = 50;
//The 'which_led' variable allows a switch between seven-segment LEDs
int which_led = 0;
//The 'counter_2_leds' variable allows a blinking of the two LEDs every second.
int counter_2_leds = 2;
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
  counter--;
  if(counter <= 0) {
    counter = 50;
    counter_2_leds--;
    switch(which_led) {
    case 0:
      which_led = 1;
      //display number '1' at 1st seven segment LED
      display7SEG(1);
      //Turn on the 1st seven-segment LED. Meanwhile, turn off the remaining seven-segment LEDs
      HAL_GPIO_WritePin(GPIOA, en0_Pin, RESET);
      HAL_GPIO_WritePin(GPIOA, en1_Pin, SET);
      HAL_GPIO_WritePin(GPIOA, en2_Pin, SET);
      HAL_GPIO_WritePin(GPIOA, en3_Pin, SET);
      break;
    case 1:
      which_led = 2;
      //display number '2' at 2nd seven segment LED
      display7SEG(2);
      //Turn on the 2nd seven-segment LED. Meanwhile, turn off the remaining seven-segment LEDs
      HAL_GPIO_WritePin(GPIOA, en0_Pin, SET);
      HAL_GPIO_WritePin(GPIOA, en1_Pin, RESET);
      HAL_GPIO_WritePin(GPIOA, en2_Pin, SET);
      HAL_GPIO_WritePin(GPIOA, en3_Pin, SET);
      break;
    case 2:
      which_led = 3;
      //display number '3' at 3rd seven segment LED
      display7SEG(3);
      //Turn on the 3rd seven-segment LED. Meanwhile, turn off the remaining seven-segment LEDs
      HAL_GPIO_WritePin(GPIOA, en0_Pin, SET);
      HAL_GPIO_WritePin(GPIOA, en1_Pin, SET);
      HAL_GPIO_WritePin(GPIOA, en2_Pin, RESET);
      HAL_GPIO_WritePin(GPIOA, en3_Pin, SET);
```

```
41        break;
42      case 3:
43        which_led = 0;
44        //display numer '0' at 4th seven segment LED
45        display7SEG(0);
46        //Turn on the 4th seven-segment LED. Meanwhile, turn
     off the remaining seven-segment LEDs
47        HAL_GPIO_WritePin(GPIOA, en0_Pin, SET);
48        HAL_GPIO_WritePin(GPIOA, en1_Pin, SET);
49        HAL_GPIO_WritePin(GPIOA, en2_Pin, SET);
50        HAL_GPIO_WritePin(GPIOA, en3_Pin, RESET);
51        break;
52      default:
53        break;
54      }
55      //After scanning four seven-segment LEDs, two LEDs have
     to switch their status.
56      if(counter_2_leds <= 0) {
57        HAL_GPIO_TogglePin(GPIOA, dot_Pin);
58        counter_2_leds = 2;
59      }
60    }
61  }
```

Program 1.5: Code for ex2

Short question: What is the frequency of the scanning process?

Thời gian chuyển đổi giữa từng LED 7 đoạn là nửa giây (500ms). Ta có 4 thanh LED 7 đoạn. Như vậy, tính được chu kỳ là 4*500ms = 2s. Tần số của quá trình quét 4 thanh LED 7 đoạn sẽ là 1/2s = 0.5Hz.

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.

```
const int MAX_LED = 4;
//The 'index_led' variable allows a switch between seven-
   segment LEDs
int index_led = 0;
//The array containing the values will be updated to the
   seven-segment LED which matches its position in the
   array.
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]
    display7SEG(led_buffer[index]);
    index_led = 1;
    //Turn on the 1st seven-segment LED. Meanwhile, turn
```

```
   off the remaining seven-segment LEDs
    HAL_GPIO_WritePin(GPIOA, en0_Pin, RESET);
    HAL_GPIO_WritePin(GPIOA, en1_Pin, SET);
    HAL_GPIO_WritePin(GPIOA, en2_Pin, SET);
    HAL_GPIO_WritePin(GPIOA, en3_Pin, SET);
    break ;
  case 1:
   // Display the second 7 SEG with led_buffer [1]
    display7SEG(led_buffer[index]);
    index_led = 2;
    //Turn on the 2nd seven-segment LED. Meanwhile, turn
   off the remaining seven-segment LEDs
    HAL_GPIO_WritePin(GPIOA, en0_Pin, SET);
    HAL_GPIO_WritePin(GPIOA, en1_Pin, RESET);
    HAL_GPIO_WritePin(GPIOA, en2_Pin, SET);
    HAL_GPIO_WritePin(GPIOA, en3_Pin, SET);
    break ;
  case 2:
  // Display the third 7 SEG with led_buffer [2]
    display7SEG(led_buffer[index]);
    index_led = 3;
    //Turn on the 3rd seven-segment LED. Meanwhile, turn
   off the remaining seven-segment LEDs
    HAL_GPIO_WritePin(GPIOA, en0_Pin, SET);
    HAL_GPIO_WritePin(GPIOA, en1_Pin, SET);
    HAL_GPIO_WritePin(GPIOA, en2_Pin, RESET);
    HAL_GPIO_WritePin(GPIOA, en3_Pin, SET);
    break ;
  case 3:
  // Display the forth 7 SEG with led_buffer [3]
    display7SEG(led_buffer[index]);
    index_led = 0;
    //Turn on the 4th seven-segment LED. Meanwhile, turn
   off the remaining seven-segment LEDs
    HAL_GPIO_WritePin(GPIOA, en0_Pin, SET);
    HAL_GPIO_WritePin(GPIOA, en1_Pin, SET);
    HAL_GPIO_WritePin(GPIOA, en2_Pin, SET);
    HAL_GPIO_WritePin(GPIOA, en3_Pin, RESET);
    break ;
  default :
    break ;
  }
}
```

Program 1.7: Code for ex3

Report 2: Present the source code in the HAL_TIM_PeriodElapsedCallback.

```
int counter = 50;
//The 'counter_2_leds' variable allows a blinking of the
```

```
   two LEDs every second.
int counter_2_leds = 2;
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
    {
  counter--;
  if(counter <= 0) {
    counter = 50;
    counter_2_leds--;
    //Calling the 'update7SEG' function to update all seven
   -segment LEDs.
    update7SEG(index_led);
    //After scanning four seven-segment LEDs, two LEDs have
    to switch their status.
    if(counter_2_leds <= 0) {
      HAL_GPIO_TogglePin(GPIOA, dot_Pin);
      counter_2_leds = 2;
    }
  }
}
```

Program 1.8: Code for ex3

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**.

Solution: Vì theo yêu cầu của đề bài, nếu tăng tầng số quét của 4 thanh LED 7 đoạn lên thành 1HZ (tương đương với chu kỳ là 1s), thời gian chuyển đổi giữa các thanh LED 7 đoạn sẽ chỉ còn là 1s/4 = 0.25s = 250ms. Do vậy, giá trị của biến counter cần phải giảm xuống còn 25.

Đồng thời, để hai LED của DOT vẫn nhấp nháy mỗi giây, cần chu kỳ cho DOT là 2s. Vậy cần phải đếm hết counter = 25 qua 4 lượt thì đổi DOT cho hai LED nhấp nháy. Cần có thêm biến counter_2_leds và giá trị của biến này là 4.

```
int counter = 25;
int counter_2_leds = 4;
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
    {
  counter--;
  if(counter <= 0) {
    counter = 25;
    counter_2_leds--;
    //Calling the 'update7SEG' function to update all seven
   -segment LEDs.
    update7SEG(index_led);
    //After scanning four seven-segment LEDs, two LEDs have
    to switch their status.
    if(counter_2_leds <= 0) {
      HAL_GPIO_TogglePin(GPIOA, dot_Pin);
      counter_2_leds = 4;
    }
  }
}
```

Program 1.9: Code for 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:

```
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.

```
void updateClockBuffer() {
  //update hour
  if(hour >= 10)
    led_buffer[0] = hour/10;
  else
    led_buffer[0] = 0;
  led_buffer[1] = hour%10;

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

Program 1.11: Code for ex5

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:

```
1 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?

Dòng 1 của code trên mất đi, **timer0_counter = 0** theo mặc định (do duration không nhận được giá trị truyền vào), dẫn tới khi thực hiện hàm **timer_run**, **timer0_flag** vẫn giữ nguyên bằng 0 theo mặc định.

Và vì hàm **run_timer** trong hàm **HAL_TIM_PeriodElapsedCallback** sẽ được chạy trước vòng while(1), đồng thời **timer0_flag = 0** (không thỏa điều kiện **if(timer0_flag == 1)** trong vòng while(1)), dẫn tới đèn LED sẽ không được Toggle mà vẫn giữ nguyên trạng thái bắt đầu khi chạy.

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

Dòng 1 của code trên chuyển thành **setTimre0(1)**, thì sau khi chạy hàm **timer_run** thì **timer0_flag** vẫn bằng 0 (do **timer0_counter = duration / TIMER_CYCLE** = 1/10 = 0)

Vì vậy, kết quả ở trường hợp này sẽ giống với trường hợp trên.

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?

Dòng 1 của code trên chuyển thành **setTimer0(10)**, thì sau khi chạy hàm **timer_run** thì **timer0_flag = 1** (do **timer0_counter = duration / TIMER_CYCLE** = 10/10 = 1)

Vì vậy, đèn LED trong trường hợp này sẽ Toggle trạng thái được, ban đầu sẽ Toggle khá nhanh do **setTimer0(10)**, tương ứng với 1ms, và sau đó sẽ Toggle ổn định mỗi 2s.

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.

```
while (1)
{
    if(timer0_flag == 1) {
        //counting second
    second++;
    //increasing minute and updating second
    if (second >= 60) {
          second = 0;
      minute ++;
    }
    //increasing hour and updating minute
    if(minute >= 60) {
        minute = 0;
      hour ++;
    }
    //updating hour
    if(hour >= 24) {
        hour = 0;
    }
    //updating and displaying time to 4 seven-segment LEDs
    updateClockBuffer();
    HAL_GPIO_TogglePin(GPIOA, dot_Pin);
    //Lasting the process for 1s
    setTimer0(1000);
  }
}
```

Program 1.15: Code for 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: Present your source code in the the main function. In the case more extra functions are used (e.g. the second software timer), present them in the report as well.

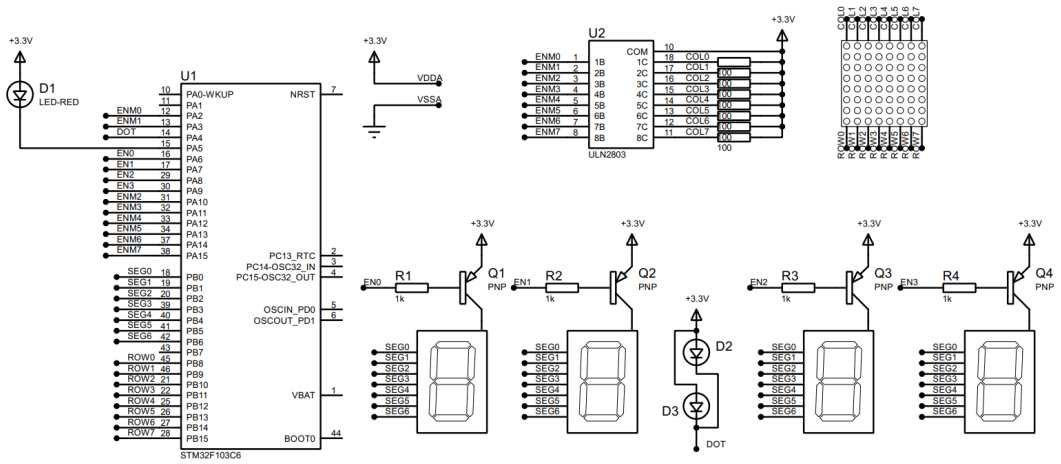
```
//update7SEG
const int MAX_LED = 4;
int index_led = 0;
int led_buffer[4] = {1, 2, 3, 4};

//updateClockBuffer
int hour = 15 , minute = 8 , second = 50;

//setTimer0
int timer0_counter = 0;
int timer0_flag = 0;

//setTimer1
int timer1_counter = 0;
int timer1_flag = 0;

int TIMER_CYCLE = 10;

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

void setTimer1(int duration) {
  timer1_counter = duration / TIMER_CYCLE;
  timer1_flag = 0;
}

void timer_run() {
    //use for setTimer0
  if(timer0_counter > 0) {
    timer0_counter--;
    if(timer0_counter == 0)
      timer0_flag = 1;
  }

    //use for setTimer0
  if(timer1_counter > 0) {
```

```
    timer1_counter--;
    if(timer1_counter == 0)
      timer1_flag = 1;
  }
}

setTimer0(10);
setTimer1(10);

while (1)
{
    if(timer0_flag == 1) {
        //counting second
    second++;
    //increasing minute and updating second
    if (second >= 60) {
          second = 0;
      minute ++;
    }
    //increasing hour and updating minute
    if(minute >= 60) {
        minute = 0;
      hour ++;
    }
    //updating hour
    if(hour >= 24) {
        hour = 0;
    }
      //updating time to 4 seven-segment LEDs
        updateClockBuffer();
        HAL_GPIO_TogglePin(GPIOA, dot_Pin);
        //Lasting the process for 1s
        setTimer0(1000);
  }

  if(timer1_flag == 1) {
      //updating changes and displaying time to 4 seven-
   segment LEDs
    update7SEG(index_led);
    //Lasting the process for 250ms
    setTimer1(250);
  }
}
```

Program 1.16: Code for 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:



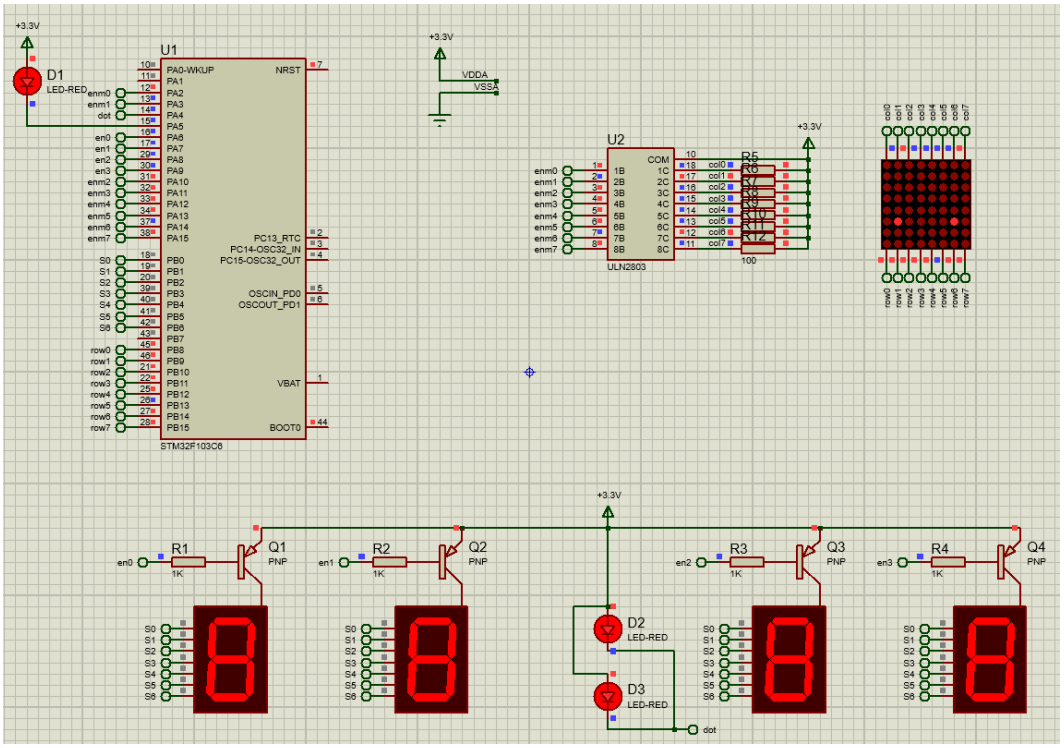
Hình 1.9: LED matrix is added to the simulation

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.

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] = {0x01, 0x02, 0x03, 0x04, 0x05, 0
   x06, 0x07, 0x08};
void updateLEDMatrix(int index){
    switch (index){
        case 0:
            break;
        case 1:
            break;
        case 2:
            break;
        case 3:
            break;
        case 4:
            break;
        case 5:
            break;
```



Hình 1.10: LED matrix is added to the simulation

```
        case 6:
            break;
        case 7:
            break;
        default:
            break;
    }
}
```

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"**.

Solution: Để thuận tiện việc quét và hiện chữ A lên LED matrix 8x8, em bổ sung thêm hàm **displayLEDMatrix**, trong hàm này có mảng **dis_arr[8]** mô tả trạng thái chữ A khi hiện trên LED matrix 8x8. Mảng **matrix_buffer** thay vào đó, sẽ được sử dụng ở exercise 10, nhằm cập nhật hiệu ứng cuộn chữ theo chiều dọc. Đồng thời các giá trị sẽ được cập nhật thành như sau:
uint8_t matrix_buffer[8] = 0x00,0x01,0x02,0x03,0x04,0x05,0x06,0x07;
ứng với index mảng **dis_arr[8]** của từng hàng LED matrix 8x8.

```
void displayLEDMatrix(int num) {
    //Variable var2 takes the code inverse of the 'index'
   row from the dis_arr[8] array, decodes select which LED
   in which column is allowed to light by bit shifting, and
    turns on the selected LED.
  int dis_arr[8] = {0x18,0x24,0x42,0x42,0x7E,0x42,0x42,0x42
```

```
    };
  int bit_var, var2;

  var2 = ~dis_arr[num];
  bit_var = var2 & 0x01;
  HAL_GPIO_WritePin(enm0_GPIO_Port, enm0_Pin, bit_var);

  bit_var = (var2>>1) & 0x01;
  HAL_GPIO_WritePin(enm1_GPIO_Port, enm1_Pin, bit_var);

  bit_var = (var2>>2) & 0x01;
  HAL_GPIO_WritePin(enm2_GPIO_Port, enm2_Pin, bit_var);

  bit_var = (var2>>3) & 0x01;
  HAL_GPIO_WritePin(enm3_GPIO_Port, enm3_Pin, bit_var);

  bit_var = (var2>>4) & 0x01;
  HAL_GPIO_WritePin(enm4_GPIO_Port, enm4_Pin, bit_var);

  bit_var = (var2>>5) & 0x01;
  HAL_GPIO_WritePin(enm5_GPIO_Port, enm5_Pin, bit_var);

  bit_var = (var2>>6) & 0x01;
  HAL_GPIO_WritePin(enm6_GPIO_Port, enm6_Pin, bit_var);

  bit_var = (var2>>7) & 0x01;
  HAL_GPIO_WritePin(enm7_GPIO_Port, enm7_Pin, bit_var);
}

//updateLEDMatrix
const int MAX_LED_MATRIX = 8;
int index_led_matrix = 0;
uint8_t matrix_buffer[8] = {0x00,0x01,0x02,0x03,0x04,0x05,0
   x06,0x07};
void updateLEDMatrix(int index) {
    //Get the display effect based on the matrix_buffer[
   index] array and update the index_led_matrix to jump to
   the next state when calling the updateLEDMatrix function
   . Also, enable the 'index' row to be turned on and the
   remaining rows off.
  switch(index) {
    case 0:
      displayLEDMatrix(matrix_buffer[index]);
      index_led_matrix = 1;
      HAL_GPIO_WritePin(row0_GPIO_Port, row0_Pin, RESET);
      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);
```

```
47        HAL_GPIO_WritePin(row5_GPIO_Port, row5_Pin, SET);
48        HAL_GPIO_WritePin(row6_GPIO_Port, row6_Pin, SET);
49        HAL_GPIO_WritePin(row7_GPIO_Port, row7_Pin, SET);
50        break;
51      case 1:
52        displayLEDMatrix(matrix_buffer[index]);
53        index_led_matrix = 2;
54        HAL_GPIO_WritePin(row0_GPIO_Port, row0_Pin, SET);
55        HAL_GPIO_WritePin(row1_GPIO_Port, row1_Pin, RESET);
56        HAL_GPIO_WritePin(row2_GPIO_Port, row2_Pin, SET);
57        HAL_GPIO_WritePin(row3_GPIO_Port, row3_Pin, SET);
58        HAL_GPIO_WritePin(row4_GPIO_Port, row4_Pin, SET);
59        HAL_GPIO_WritePin(row5_GPIO_Port, row5_Pin, SET);
60        HAL_GPIO_WritePin(row6_GPIO_Port, row6_Pin, SET);
61        HAL_GPIO_WritePin(row7_GPIO_Port, row7_Pin, SET);
62        break;
63      case 2:
64        displayLEDMatrix(matrix_buffer[index]);
65        index_led_matrix = 3;
66        HAL_GPIO_WritePin(row0_GPIO_Port, row0_Pin, SET);
67        HAL_GPIO_WritePin(row1_GPIO_Port, row1_Pin, SET);
68        HAL_GPIO_WritePin(row2_GPIO_Port, row2_Pin, RESET);
69        HAL_GPIO_WritePin(row3_GPIO_Port, row3_Pin, SET);
70        HAL_GPIO_WritePin(row4_GPIO_Port, row4_Pin, SET);
71        HAL_GPIO_WritePin(row5_GPIO_Port, row5_Pin, SET);
72        HAL_GPIO_WritePin(row6_GPIO_Port, row6_Pin, SET);
73        HAL_GPIO_WritePin(row7_GPIO_Port, row7_Pin, SET);
74        break;
75      case 3:
76        displayLEDMatrix(matrix_buffer[index]);
77        index_led_matrix = 4;
78        HAL_GPIO_WritePin(row0_GPIO_Port, row0_Pin, SET);
79        HAL_GPIO_WritePin(row1_GPIO_Port, row1_Pin, SET);
80        HAL_GPIO_WritePin(row2_GPIO_Port, row2_Pin, SET);
81        HAL_GPIO_WritePin(row3_GPIO_Port, row3_Pin, RESET);
82        HAL_GPIO_WritePin(row4_GPIO_Port, row4_Pin, SET);
83        HAL_GPIO_WritePin(row5_GPIO_Port, row5_Pin, SET);
84        HAL_GPIO_WritePin(row6_GPIO_Port, row6_Pin, SET);
85        HAL_GPIO_WritePin(row7_GPIO_Port, row7_Pin, SET);
86        break;
87      case 4:
88        displayLEDMatrix(matrix_buffer[index]);
89        index_led_matrix = 5;
90        HAL_GPIO_WritePin(row0_GPIO_Port, row0_Pin, SET);
91        HAL_GPIO_WritePin(row1_GPIO_Port, row1_Pin, SET);
92        HAL_GPIO_WritePin(row2_GPIO_Port, row2_Pin, SET);
93        HAL_GPIO_WritePin(row3_GPIO_Port, row3_Pin, SET);
94        HAL_GPIO_WritePin(row4_GPIO_Port, row4_Pin, RESET);
95        HAL_GPIO_WritePin(row5_GPIO_Port, row5_Pin, SET);
```

```
96        HAL_GPIO_WritePin(row6_GPIO_Port, row6_Pin, SET);
97        HAL_GPIO_WritePin(row7_GPIO_Port, row7_Pin, SET);
98        break;
99      case 5:
100       displayLEDMatrix(matrix_buffer[index]);
101       index_led_matrix = 6;
102       HAL_GPIO_WritePin(row0_GPIO_Port, row0_Pin, SET);
103       HAL_GPIO_WritePin(row1_GPIO_Port, row1_Pin, SET);
104       HAL_GPIO_WritePin(row2_GPIO_Port, row2_Pin, SET);
105       HAL_GPIO_WritePin(row3_GPIO_Port, row3_Pin, SET);
106       HAL_GPIO_WritePin(row4_GPIO_Port, row4_Pin, SET);
107       HAL_GPIO_WritePin(row5_GPIO_Port, row5_Pin, RESET);
108       HAL_GPIO_WritePin(row6_GPIO_Port, row6_Pin, SET);
109       HAL_GPIO_WritePin(row7_GPIO_Port, row7_Pin, SET);
110       break;
111     case 6:
112       displayLEDMatrix(matrix_buffer[index]);
113       index_led_matrix = 7;
114       HAL_GPIO_WritePin(row0_GPIO_Port, row0_Pin, SET);
115       HAL_GPIO_WritePin(row1_GPIO_Port, row1_Pin, SET);
116       HAL_GPIO_WritePin(row2_GPIO_Port, row2_Pin, SET);
117       HAL_GPIO_WritePin(row3_GPIO_Port, row3_Pin, SET);
118       HAL_GPIO_WritePin(row4_GPIO_Port, row4_Pin, SET);
119       HAL_GPIO_WritePin(row5_GPIO_Port, row5_Pin, SET);
120       HAL_GPIO_WritePin(row6_GPIO_Port, row6_Pin, RESET);
121       HAL_GPIO_WritePin(row7_GPIO_Port, row7_Pin, SET);
122       break;
123     case 7:
124       displayLEDMatrix(matrix_buffer[index]);
125       index_led_matrix = 0;
126       HAL_GPIO_WritePin(row0_GPIO_Port, row0_Pin, SET);
127       HAL_GPIO_WritePin(row1_GPIO_Port, row1_Pin, SET);
128       HAL_GPIO_WritePin(row2_GPIO_Port, row2_Pin, SET);
129       HAL_GPIO_WritePin(row3_GPIO_Port, row3_Pin, SET);
130       HAL_GPIO_WritePin(row4_GPIO_Port, row4_Pin, SET);
131       HAL_GPIO_WritePin(row5_GPIO_Port, row5_Pin, SET);
132       HAL_GPIO_WritePin(row6_GPIO_Port, row6_Pin, SET);
133       HAL_GPIO_WritePin(row7_GPIO_Port, row7_Pin, RESET);
134       break;
135     default:
136       break;
137   }
138 }
```

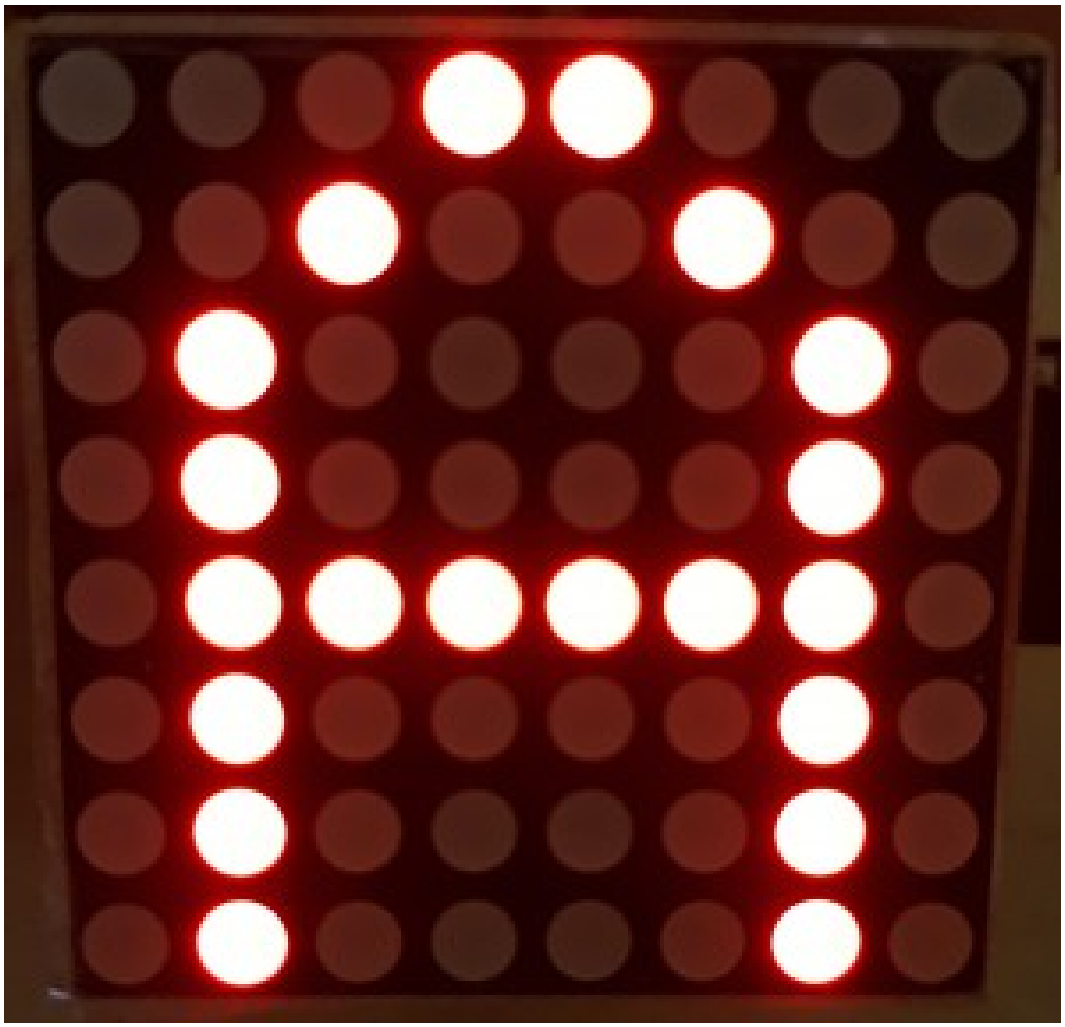
Program 1.18: Code for ex9

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.

Solution: ở exercise 9, LED matrix quét hiện chữ A theo chiều từ trên xuống (quét tuần tự từ ROW0 xuống ROW7). Ở câu này, em chọn hiệu ứng cho chạy chữ A cuộn từ dưới lên trên (chọn cuộn theo chiều dọc vì cùng chiều với chiều quét hiện chữ, tạo sự thuận tiện trong việc viết và debug code). Tái sử dụng hàm**updateLEDMatrix**, bổ sung thêm hàm **displayLEDMatrix** (điều khiển 8 LED trong 1 hàng của LED matrix 8x8) và hàm **updateMatrixBuffer** (cập nhật trạng thái mới cho từng hàng của LED matrix 8x8).



Hình 1.11: chữ A trên LED matrix 8x8

Description:
Bước 1: LED matrix 8x8 sẽ mất 1s (sử dụng **setTimer1(125)**) để quét qua hết 8 hàng.

Bước 2: LED matrix 8x8 sẽ bắt đầu chạy hiệu ứng cuộn chữ lên theo chiều dọc 1 hàng (sử dụng **setTimer0(1000)**), và trong 1s tiếp theo LED matrix 8x8 sẽ quét ra kết quả hiệu ứng các hàng, cụ thể là:
hàng 1 chuyển xuống hàng 8
hàng 2 lên hàng 1
hàng 3 lên hàng 2
...
hàng 8 lên hàng 7
Các giây sau sẽ tiếp tục thực hiện tương tự **bước 2** như trên.

Note: do proteus máy em không chạy đồng thời được 4 thanh LED 7 đoạn và LED matrix 8x8, vòng while em sẽ chỉ để duy nhất các hàm liên quan đến LED matrix.

```
//updateLEDMatrix
const int MAX_LED_MATRIX = 8;
int index_led_matrix = 0;
uint8_t matrix_buffer[8] = {0x00,0x01,0x02,0x03,0x04,0x05,0
   x06,0x07};

//setTimer0
int timer0_counter = 0;
int timer0_flag = 0;

//setTimer1
int timer1_counter = 0;
int timer1_flag = 0;

int TIMER_CYCLE = 10;

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

void setTimer1(int duration) {
  timer1_counter = duration / TIMER_CYCLE;
  timer1_flag = 0;
}

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

  if(timer1_counter > 0) {
    timer1_counter--;
    if(timer1_counter == 0)
      timer1_flag = 1;
```

```
  }
}

setTimer0(1000);
setTimer1(125);
while (1)
{
  if(timer0_flag == 1) {
    updateMatrixBuffer();
    setTimer0(1000);
  }

  if(timer1_flag == 1) {
    updateLEDMatrix(index_led_matrix);
    setTimer1(125);
  }
}

//New status update for each row of LED matrix 8x8
uint8_t temp;
void updateMatrixBuffer() {
  temp = matrix_buffer[0];
  matrix_buffer[0] = matrix_buffer[1];
  matrix_buffer[1] = matrix_buffer[2];
  matrix_buffer[2] = matrix_buffer[3];
  matrix_buffer[3] = matrix_buffer[4];
  matrix_buffer[4] = matrix_buffer[5];
  matrix_buffer[5] = matrix_buffer[6];
  matrix_buffer[6] = matrix_buffer[7];
  matrix_buffer[7] = temp;
}
```

Program 1.19: Code for ex10

Link github project STM32 và proteus lab 2:

https://github.com/minhdok20/MCU_lab2