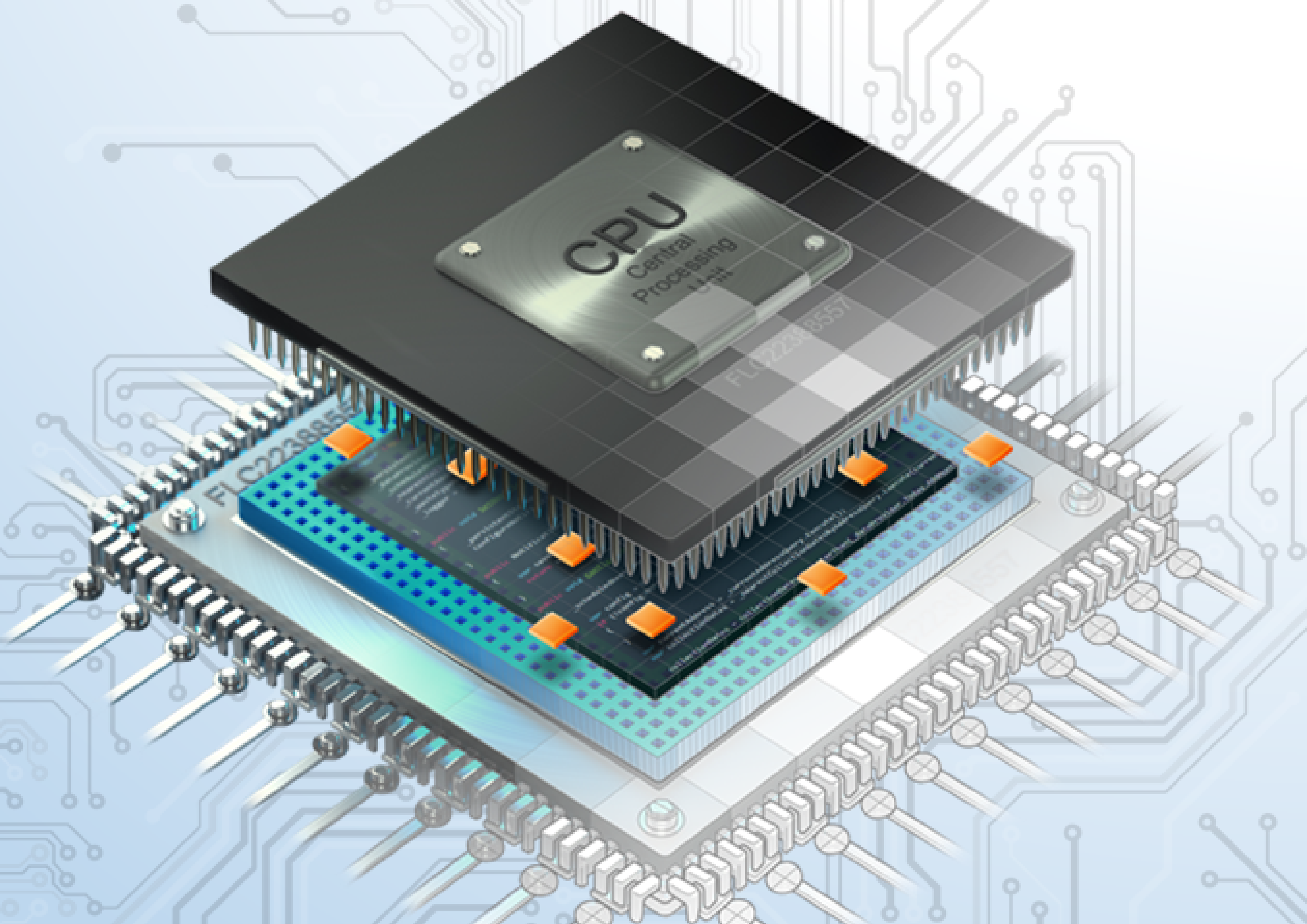
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
COMPUTER ENGINEERING

Microcontroller

Lab's Report



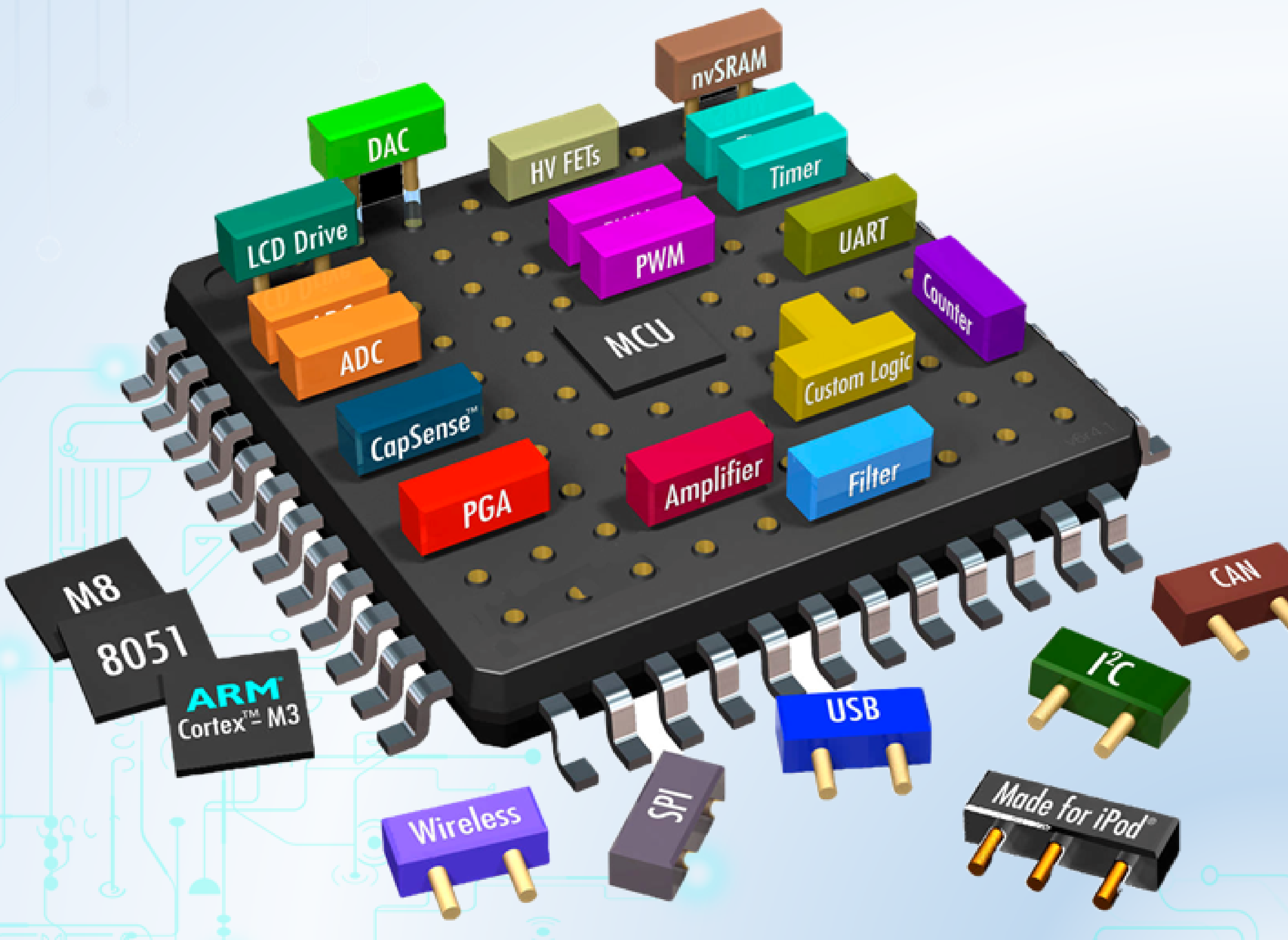
Date:15/09/2022

Cao Minh Quang - 2052221

Contents

CHAPTER 1

Timer Interrupt and LED Scanning



1 Github for version control

- Please access the following link to gain full control of this lab:
- `https://github.com/cmq2002/Microcontroller_Microprocessor_LAB2.git`

2 Exercise 1

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

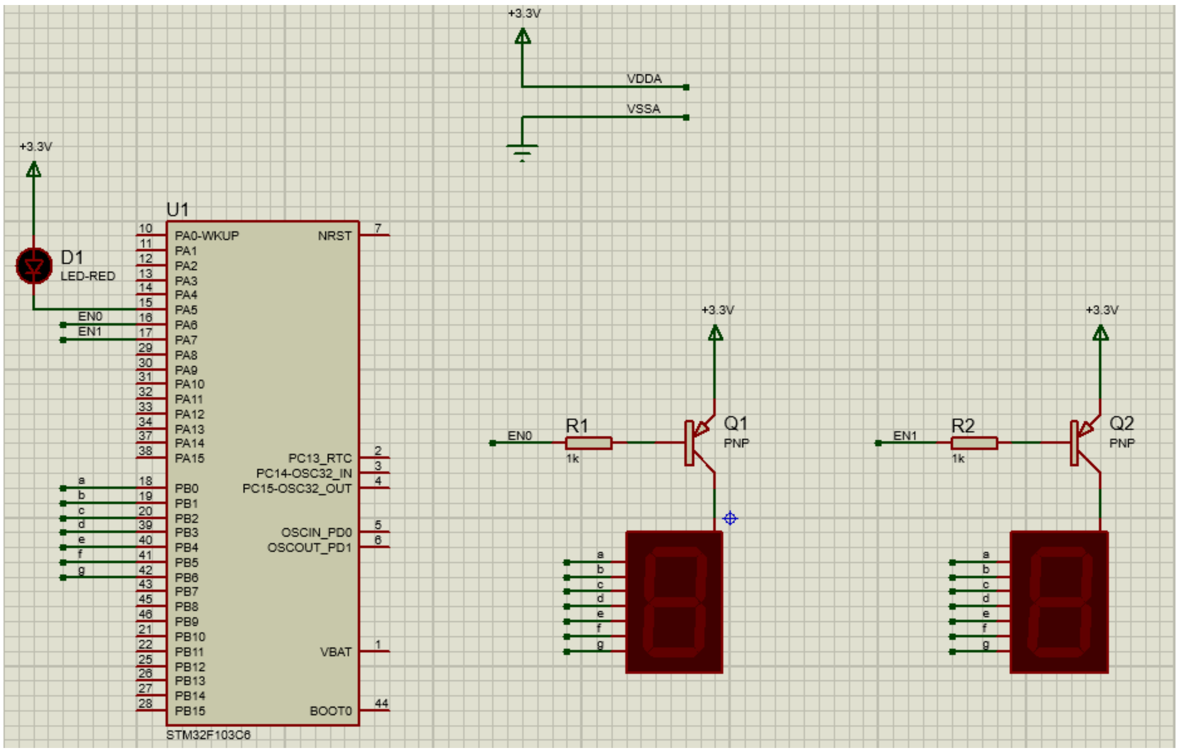


Figure 1.1: Schematic design from Proteus.

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

```
void HAL_TIM_PeriodElapsedCallback ( TIM_HandleTypeDef *
    htim )
{
  systemRun();
}
```

The section below contains all support functions:

```
#define setCounter 50
enum ledState {led1=1, led2=2};
enum ledState status = led1;

```

```
//Transfering sequence: gfedcba
// MSB = g; LSB = a -> Active Low
static uint8_t sevenSegTable[10] = {
      0x40 //0
    , 0x79 //1
    , 0x24 //2
    , 0x30 //3
    , 0x19 //4
    , 0x12 //5
    , 0x02 //6
    , 0x78 //7
    , 0x00 //8
    , 0x10 //9
};

void display7SEG (int num){
  switch (num){
    case led1:
      HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, RESET);
      HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, SET);
      break;
    case led2:
      HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, SET);
      HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, RESET);
      break;
    default:
      break;
  }
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, ((sevenSegTable[num
   ]>>0)&0x01));
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1, ((sevenSegTable[num
   ]>>1)&0x01));
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_2, ((sevenSegTable[num
   ]>>2)&0x01));
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3, ((sevenSegTable[num
   ]>>3)&0x01));
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, ((sevenSegTable[num
   ]>>4)&0x01));
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, ((sevenSegTable[num
   ]>>5)&0x01));
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, ((sevenSegTable[num
   ]>>6)&0x01));
}

static uint16_t counter = setCounter;
void systemRun(void){
  counter--;
  if (counter <=0){
    counter = setCounter;
    HAL_GPIO_TogglePin(LED_RED_GPIO_Port, LED_RED_Pin);
    display7SEG(status);
    if (status == led1) status = led2;
    else status = led1;
  }
}
```

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

- The system consists of the following states: Num1 -> Num2 -> Num1
- As the switching time between each state is 0.5s, in total, the full cycle is 1s. This give us the frequency of the scanning process $f = \frac{1}{1} = 1(Hz)$

3 Exercise 2

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

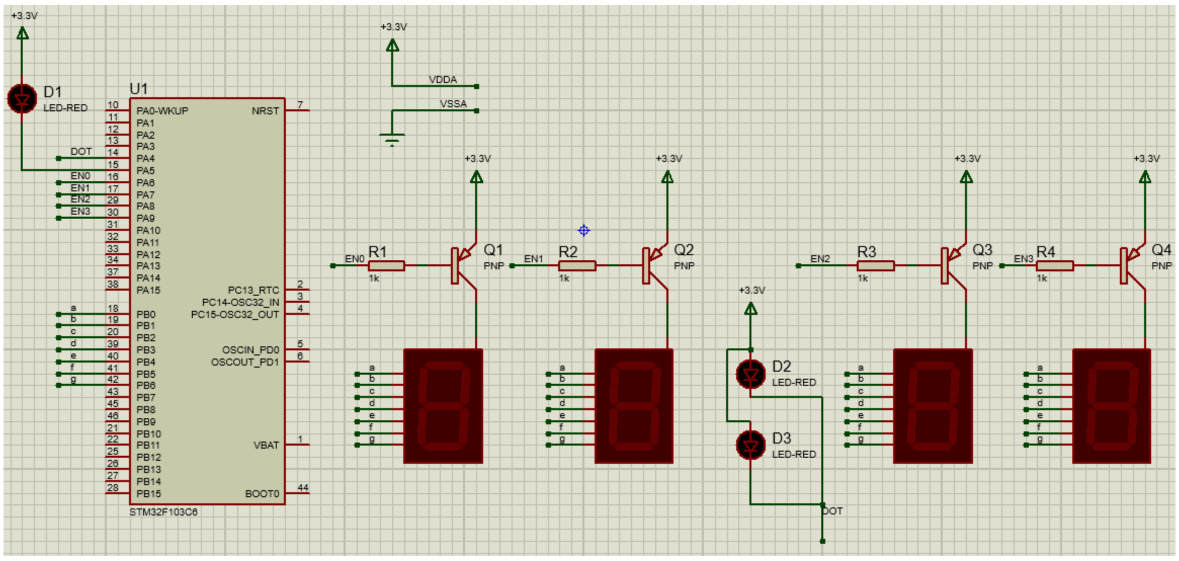


Figure 1.2: Schematic design from Proteus.

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

```
void HAL_TIM_PeriodElapsedCallback ( TIM_HandleTypeDef *
   htim )
{
  systemRun();
}
```

The section below contains all support functions:

```c
#define setCounterDot 100
#define setCounterLed 50
#define setCounter7SEG 50

enum ledState {led1=1, led2=2, led3=3, led4=0};
enum ledState status = led1;
//Transfering sequence: gfedcba
// MSB = g; LSB = a -> Active Low
static uint8_t sevenSegTable[10] = {
    0x40 //0
  , 0x79 //1
  , 0x24 //2
  , 0x30 //3
  , 0x19 //4
  , 0x12 //5
  , 0x02 //6
  , 0x78 //7
  , 0x00 //8
  , 0x10 //9
};

void display7SEG (int num){
  switch (num){
    case led1:
      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);
      status = led2;
      break;
    case led2:
      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);
      status = led3;
      break;
    case led3:
      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);
      status = led4;
      break;
    case led4:
      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);
      status = led1;
      break;
    default:
      break;
  }
  GPIOB->ODR = sevenSegTable[num];
}

static uint16_t counterLed = setCounterLed;
static uint16_t counter7SEG = setCounter7SEG;
static uint16_t counterDot = setCounterDot;

void systemRun(void){
  counterLed--;
  counter7SEG--;
  counterDot--;
  if (counterLed <= 0){
    counterLed = setCounterLed;
    HAL_GPIO_TogglePin(LED_RED_GPIO_Port, LED_RED_Pin);
  }
  if (counter7SEG <= 0){
    counter7SEG = setCounter7SEG;
    display7SEG(status);
  }
  if (counterDot <= 0){
    counterDot =setCounterDot;
    HAL_GPIO_TogglePin(DOT_GPIO_Port, DOT_Pin);
  }
}
```

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

- In this circumstance, our system now has 4 state:
- Num1->Num2->Num3->Num0->Num1. This makes our full cycle of 2s.
- As a consequenc, the scanning frequency is $f = \frac{1}{2} = 0.5Hz$

4 Exercise 3

Report 1: Present the source code of the update7SEG function.
This section also consists of all support functions.

```c
#define setCounterDot 100
#define setCounter 50

//Transfer sequence: gfedcba
//MSB=g, LSB=a <- Active low
```

```c
//Ex: To display 0 -> sequence = 1000000
const uint8_t sevenSegTable[10] = {
      0x0040 //0
    , 0x0079 //1
    , 0x0024 //2
    , 0x0030 //3
    , 0x0019 //4
    , 0x0012 //5
    , 0x0002 //6
    , 0x0078 //7
    , 0x0000 //8
    , 0x0010 //9
};

void display7SEG (int num){
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, ((sevenSegTable[num
   ]>>0)&0x01));
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1, ((sevenSegTable[num
   ]>>1)&0x01));
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_2, ((sevenSegTable[num
   ]>>2)&0x01));
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3, ((sevenSegTable[num
   ]>>3)&0x01));
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, ((sevenSegTable[num
   ]>>4)&0x01));
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, ((sevenSegTable[num
   ]>>5)&0x01));
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, ((sevenSegTable[num
   ]>>6)&0x01));
}

static uint8_t led_buffer[4] = {1, 2, 5, 8};
void update7SEG (int index){
  switch (index){
    case 0:
      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);
      break;
    case 1:
      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);
      break;
    case 2:
      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);
      break;
    case 3:
      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);
      break;
    default:
      break;
  }
  display7SEG(led_buffer[index]);
}

const uint8_t MAX_LED = 4;
static uint8_t index_led = 0;

static uint16_t counterLed = setCounter;
static uint16_t counter7SEG = setCounter;
static uint16_t counterDot = setCounterDot;

void systemRun(void){
  counterLed--;
  counter7SEG--;
  counterDot--;
  if (counterLed <= 0){
    counterLed = setCounter;
    HAL_GPIO_TogglePin(LED_RED_GPIO_Port, LED_RED_Pin);
  }
  if (counter7SEG <= 0){
    counter7SEG = setCounter;
    update7SEG(index_led);
    index_led = (index_led + 1) % MAX_LED;
  }
  if (counterDot <= 0){
    counterDot = setCounterDot;
    HAL_GPIO_TogglePin(DOT_GPIO_Port, DOT_Pin);
  }
}
```

Report 2: Present the source code in the HAL_TIM_PeriodElapsedCallback.

```
void HAL_TIM_PeriodElapsedCallback ( TIM_HandleTypeDef *
   htim )
{
  systemRun();
}
```

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

5 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.

The solution:

- When we attempt to change the scanning frequency to 1Hz, the full cycle of the system is 1Hz.
- However, the number of states is still the same as in Exercise 3 so that the switching time between each state is $\frac{1}{4} = 0.25s$.
- In general, we only need to modify the setCounter macro to 25.

```
    #define setCounter 25

```

6 Exercise 5

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

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

7 Exercise 6

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 1.1: 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 1.2: 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 1.3: 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?

- When we delete the line 1, **timer0-counter** will be 0 as it has been initialized.
- Consequently, **timer0-flag** will never be 1 and the Led can not blink.

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

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?

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

```
/* Infinite loop */
 /* USER CODE BEGIN WHILE */
 int hr = 21, min=31, sec = 56;
 updateClockBuffer(hr, min);
 setTimer1(setCounterDot);
 setTimer3(setCounterDot);
 while (1)
 {
 if (timer1_flag == 1){
     sec++;
   if (sec >= 60){sec = 0; min++;}
   if (min >= 60){min = 0; hr++;}
   if (hr >= 24) {hr = 0;}
   updateClockBuffer(hr, min);
   setTimer1(setCounterDot);
 }
 //Ex7
 if (timer3_flag == 1){
   HAL_GPIO_TogglePin(DOT_GPIO_Port,DOT_Pin);
   setTimer3(setCounterDot);
 }
   /* USER CODE END WHILE */

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

9 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 : 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.

```
/* Private user code ---------------------------------*/
/* USER CODE BEGIN 0 */
static uint8_t led_buffer[4] = {0, 0, 0, 0};
```

```c
const uint8_t MAX_LED = 4;
static uint8_t index_led = 0;
void updateClockBuffer(int hr, int min){
  led_buffer[0] = hr/10;
  led_buffer[1] = hr%10;
  led_buffer[2] = min/10;
  led_buffer[3] = min%10;
}

//Transfer sequence: gfedcba
//MSB=g, LSB=a <- Active low
//Ex: To display 0 -> sequence = 1000000
const uint8_t sevenSegTable[10] = {
      0x40 //0
    , 0x79 //1
    , 0x24 //2
    , 0x30 //3
    , 0x19 //4
    , 0x12 //5
    , 0x02 //6
    , 0x78 //7
    , 0x00 //8
    , 0x10 /*9*/};

void display7SEG (int num){
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, ((sevenSegTable[num
   ]>>0)&0x01));
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1, ((sevenSegTable[num
   ]>>1)&0x01));
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_2, ((sevenSegTable[num
   ]>>2)&0x01));
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3, ((sevenSegTable[num
   ]>>3)&0x01));
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, ((sevenSegTable[num
   ]>>4)&0x01));
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, ((sevenSegTable[num
   ]>>5)&0x01));
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, ((sevenSegTable[num
   ]>>6)&0x01));
}

void update7SEG (int index){
  switch (index){
    case 0:
      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);
      break;
    case 1:
      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);
      break;
    case 2:
      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);
      break;
    case 3:
      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);
      break;
    default:
      break;
  }
  display7SEG(led_buffer[index]);
}
/* USER CODE END 0 */


int main(void)
{
  HAL_Init();

  SystemClock_Config();

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_TIM2_Init();
  /* USER CODE BEGIN 2 */
  HAL_TIM_Base_Start_IT (& htim2 ) ;
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  int hr = 21, min=31, sec = 56;
  updateClockBuffer(hr, min);
  setTimer1(setCounterDot);
  setTimer2(setCounter);
  setTimer3(setCounterDot);
  while (1)
  {
  if (timer1_flag == 1){
      sec++;
    if (sec >= 60){sec = 0; min++;}
    if (min >= 60){min = 0; hr++;}
    if (hr >= 24) {hr = 0;}
    updateClockBuffer(hr, min);
    setTimer1(setCounterDot);
    //Ex5
    //HAL_Delay(1000);
  }

  //Ex8
  if (timer2_flag == 1){
    HAL_GPIO_TogglePin(LED_RED_GPIO_Port, LED_RED_Pin);
    update7SEG(index_led);
    index_led = (index_led + 1) % MAX_LED;
    setTimer2(setCounter);
  }

  //Ex7
  if (timer3_flag == 1){
    HAL_GPIO_TogglePin(DOT_GPIO_Port,DOT_Pin);
    setTimer3(setCounterDot);
  }
    /* USER CODE END WHILE */

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

10 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:

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.

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"**.

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

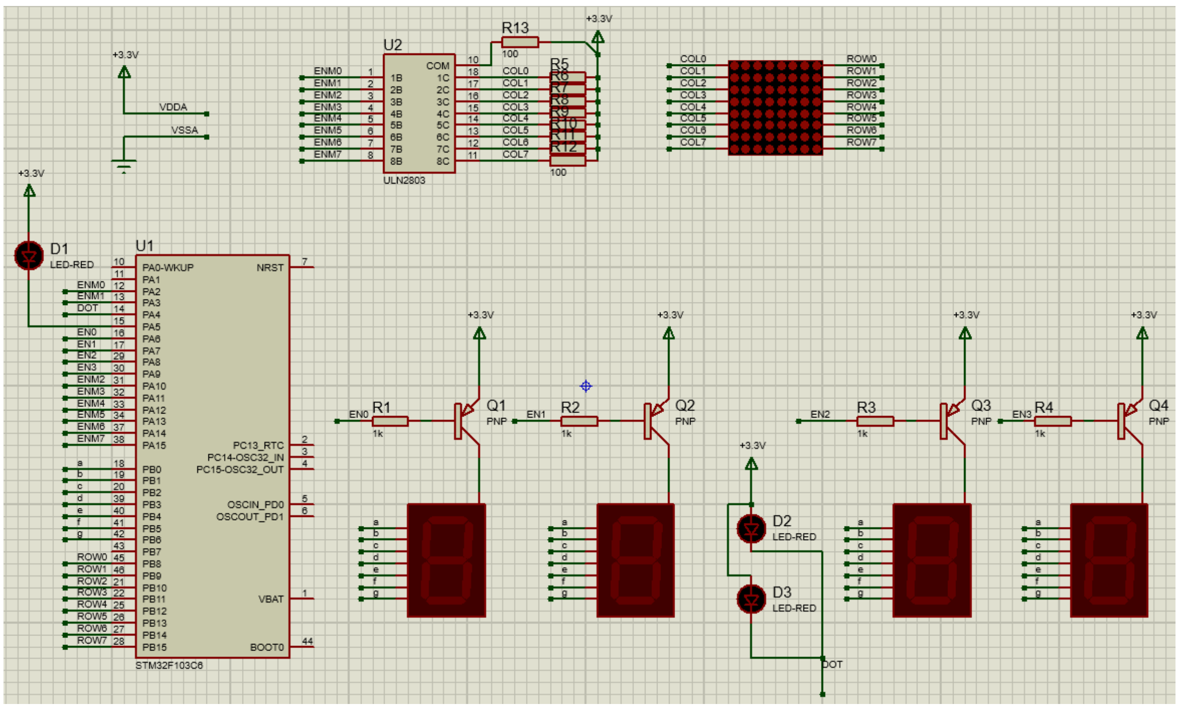


Figure 1.3: Schematic design from Proteus.

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

```c
void updateLedMatrix(int num){
  displayCol(num);
  displayRow(num);
}
```

More specific explanation and code will be included in the next exercise.

11 Exercise 10

Create an animation on LED matrix, for example, the character is shifted to the left.

Report: Briefly describe your solution and present your source code in the report.

The solution:

- We all know that to turn on a led in the matrix, its corresponding ENM and ROW's signal should be SET. When ENM is SET, its correlated COL signal is RESET.
- Base on that mechanism, in order to display the letter 'A', we first sketch the diagram of turning on led as follow:

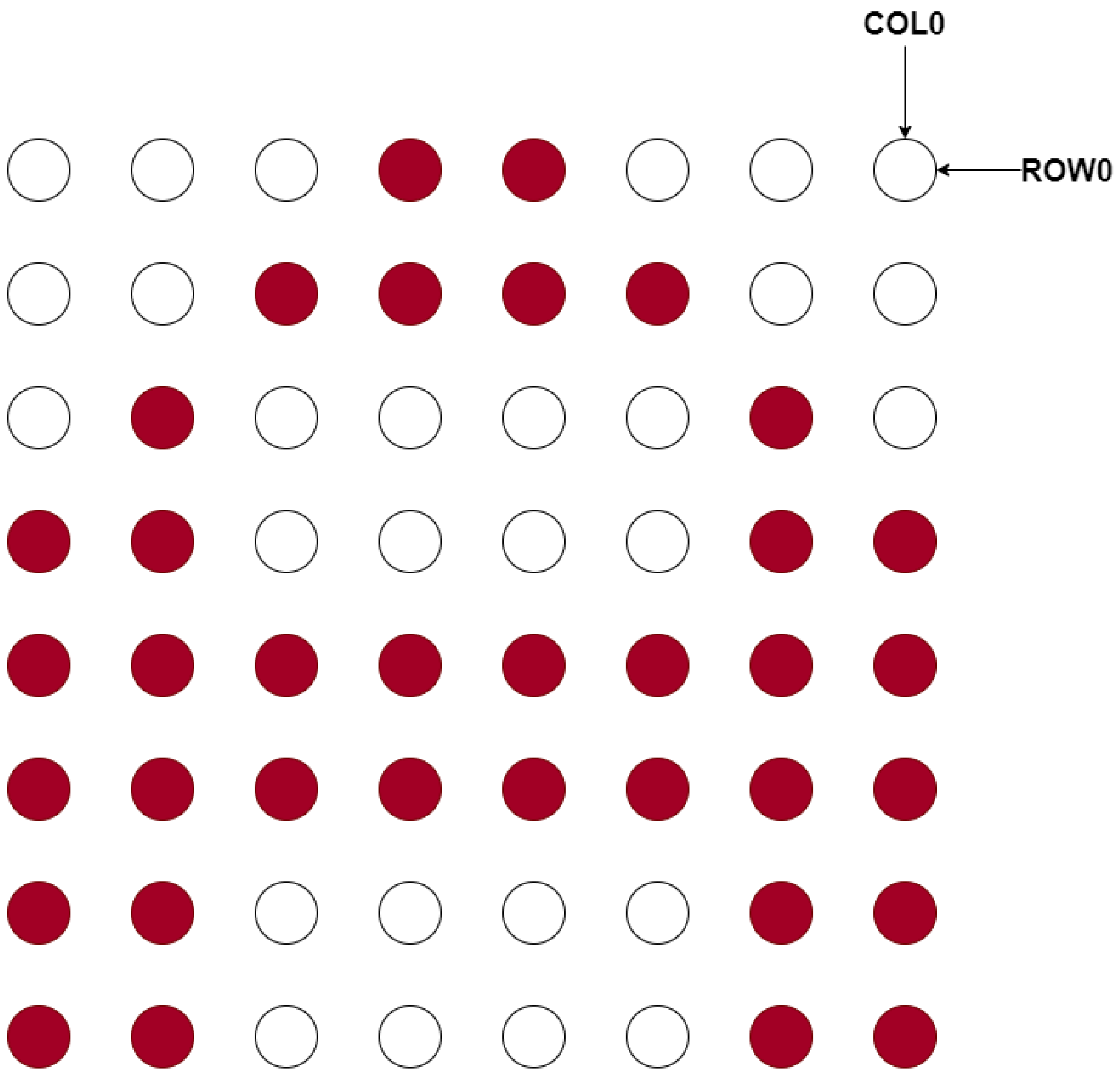


Figure 1.4: Letter A on the led matrix.

- The idea here is to sequentially enable each column with the scanning frequency around 60 - 100 Hz to form a complete letter by taking advantage of the image-retention phenomenon. So as to obtain COL0 exactly as we expected, the signal must be as following: COL0 = 0; COL1 - COL7 = 1, meanwhile, ROW0-ROW2 = 0; ROW3-ROW7 = 1. We will perform the same operation for the others column.

- To simplify our job, we will convert the signal sequence from binary to hexadecimal number. For instance, the column buffer to display the 1st column is 00000001 = $0x01$ and the row buffer is 11111000 = $0xF8$, where COL0 and ROW0 are LSB. At each moment, we will fetch out these sequence and shift bit by bit to the corresponding port.

The source code:

```
/* Private user code ---------------------------------------*/
/* USER CODE BEGIN 0 */
const int MAX_LED_MATRIX = 8;
static int index_led_matrix = 0;
static uint8_t rowBuffer[8] = {0xf8,0xfc,0x32,0x33,0x33,0
   x32,0xfc,0xf8};
static uint8_t colBuffer[8] = {0x01,0x02,0x04,0x08,0x10,0
   x20,0x40,0x80};

void displayRow (int num){
    HAL_GPIO_WritePin(ROW0_GPIO_Port,ROW0_Pin, ((rowBuffer[
   num]>>0)&0x01));
    HAL_GPIO_WritePin(ROW1_GPIO_Port,ROW1_Pin, ((rowBuffer[
   num]>>1)&0x01));
    HAL_GPIO_WritePin(ROW2_GPIO_Port,ROW2_Pin, ((rowBuffer[
   num]>>2)&0x01));
    HAL_GPIO_WritePin(ROW3_GPIO_Port,ROW3_Pin, ((rowBuffer[
   num]>>3)&0x01));
    HAL_GPIO_WritePin(ROW4_GPIO_Port,ROW4_Pin, ((rowBuffer[
   num]>>4)&0x01));
    HAL_GPIO_WritePin(ROW5_GPIO_Port,ROW5_Pin, ((rowBuffer[
   num]>>5)&0x01));
    HAL_GPIO_WritePin(ROW6_GPIO_Port,ROW6_Pin, ((rowBuffer[
   num]>>6)&0x01));
    HAL_GPIO_WritePin(ROW7_GPIO_Port,ROW7_Pin, ((rowBuffer[
   num]>>7)&0x01));
}

void displayCol (int num){
    HAL_GPIO_WritePin(ENM0_GPIO_Port, ENM0_Pin, ((colBuffer
   [num]>>0)&0x01));
    HAL_GPIO_WritePin(ENM1_GPIO_Port, ENM1_Pin, ((colBuffer
   [num]>>1)&0x01));
    HAL_GPIO_WritePin(ENM2_GPIO_Port, ENM2_Pin, ((colBuffer
   [num]>>2)&0x01));
    HAL_GPIO_WritePin(ENM3_GPIO_Port, ENM3_Pin, ((colBuffer
   [num]>>3)&0x01));
    HAL_GPIO_WritePin(ENM4_GPIO_Port, ENM4_Pin, ((colBuffer
   [num]>>4)&0x01));
```

```c
    HAL_GPIO_WritePin(ENM5_GPIO_Port, ENM5_Pin, ((colBuffer[num]>>5)&0x01));
    HAL_GPIO_WritePin(ENM6_GPIO_Port, ENM6_Pin, ((colBuffer[num]>>6)&0x01));
    HAL_GPIO_WritePin(ENM7_GPIO_Port, ENM7_Pin, ((colBuffer[num]>>7)&0x01));
}

void updateLedMatrix(int num){
  displayCol(num);
  displayRow(num);
}
/* USER CODE END 0 */

int main(void)
{
  HAL_Init();

  /* Configure the system clock */
  SystemClock_Config();

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_TIM2_Init();
  /* USER CODE BEGIN 2 */
  HAL_TIM_Base_Start_IT (& htim2 );
  /* USER CODE END 2 */

  /* Infinite loop */
  setTimer1(1);
  while (1)
  {
    if (timer1_flag == 1){
      updateLedMatrix(index_led_matrix);
      index_led_matrix = (index_led_matrix+1) % MAX_LED_MATRIX;
      setTimer1(1);
    }
  }

}
```