

VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
Faculty of Computer Science and Engineering



CC02 — Lab Report

Microprocessor - Microcontroller Lab 2

Supervisors: Nguyen Thien An
Students: Vu Trinh Thanh Binh 2252085

Ho Chi Minh City, September 30, 2024



Contents

1	Exercise	2
1.1	Exercise 1	3
1.1.1	Report 1	3
1.1.2	Report 2	3
1.2	Exercise 2	3
1.2.1	Report 1	3
1.2.2	Report 2	3
1.3	Exercise 3	5
1.3.1	Report 1	5
1.3.2	Report 2	6
1.4	Exercise 4	6
1.4.1	Report 1	6
1.5	Exercise 5	7
1.5.1	Report	7
1.6	Exercise 6	7
1.6.1	Report 1	7
1.6.2	Report 2	7
1.7	Exercise 7	7
1.8	Exercise 8	8
1.9	Exercise 9	10
1.9.1	Report 1	10
1.9.2	Report 2	10
1.10	Exercise 10	11
	References	15

1 Exercise

The GitHub link for the lab schematics is at [here](https://github.com/thanhbinh0710/VXL.git) or in this link: <https://github.com/thanhbinh0710/VXL.git>.

The schematic for the exercises:

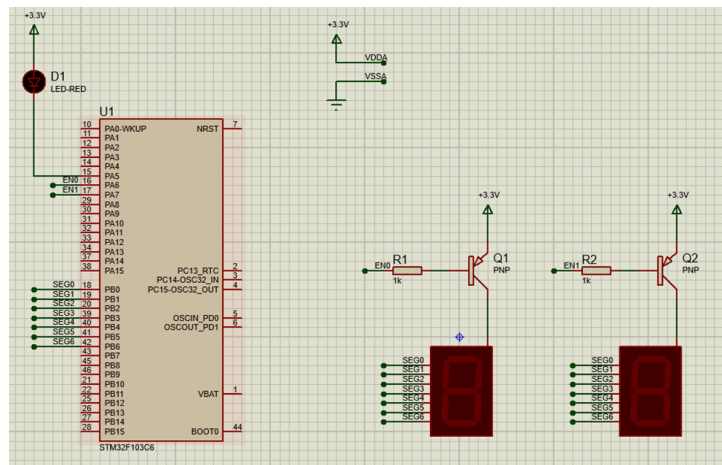


Figure 1: EX1

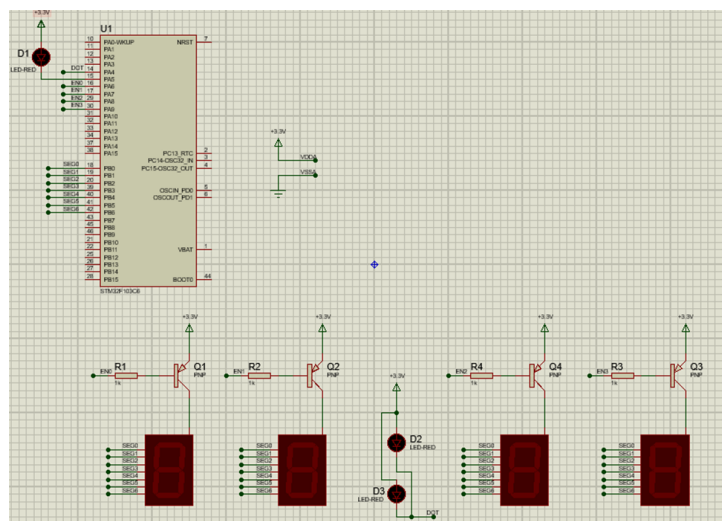


Figure 2: EX2

1.1 Exercise 1

1.1.1 Report 1

Can be found at 1.

1.1.2 Report 2

```
1 int counter = 100;
2 void HAL_TIM_PeriodElapsedCallback ( TIM_HandleTypeDef * htim )
3 {
4     if (counter == 100){
5         display7SEG(1);
6         HAL_GPIO_WritePin(En0_GPIO_Port, En0_Pin, RESET);
7         HAL_GPIO_WritePin(En1_GPIO_Port, En1_Pin, SET);
8     }else if (counter == 50){
9         display7SEG(2);
10        HAL_GPIO_WritePin(En0_GPIO_Port, En0_Pin, SET);
11        HAL_GPIO_WritePin(En1_GPIO_Port, En1_Pin, RESET);
12    }
13    counter = (counter > 0) ? counter-1 : 100;
14 }
```

- Answer: From the description provided, the switching between the two displays is performed in such a way that **2 LEDs (displays) take half a second**. Therefore, each LED is scanned once in half of this time. To calculate the frequency:

- Total time for 2 displays = 0.5 seconds
- Time for each display = 0.5 seconds / 2 = 0.25 seconds (or 250 ms)

The frequency f is the reciprocal of the time period T : $f = 1/T = 1/0.25 = 4\text{Hz}$

1.2 Exercise 2

1.2.1 Report 1

Can be found at 2.

1.2.2 Report 2

```
1 const unsigned int Default = 200;
2 int counter = Default;
3 void HAL_TIM_PeriodElapsedCallback ( TIM_HandleTypeDef * htim )
4 {
5     if (counter == Default){
6         display7SEG(1);
7         HAL_GPIO_WritePin(En0_GPIO_Port, En0_Pin, RESET);
8         HAL_GPIO_WritePin(En1_GPIO_Port, En1_Pin, SET);
9         HAL_GPIO_WritePin(En2_GPIO_Port, En2_Pin, SET);
10    }
```

```
10     HAL_GPIO_WritePin(En3_GPIO_Port, En3_Pin, SET);
11     HAL_GPIO_WritePin(Dot_GPIO_Port, Dot_Pin, RESET);
12 }else if (counter == (Default*3/4)){
13     display7SEG(2);
14     HAL_GPIO_WritePin(En0_GPIO_Port, En0_Pin, SET);
15     HAL_GPIO_WritePin(En1_GPIO_Port, En1_Pin, RESET);
16     HAL_GPIO_WritePin(En2_GPIO_Port, En2_Pin, SET);
17     HAL_GPIO_WritePin(En3_GPIO_Port, En3_Pin, SET);
18     HAL_GPIO_WritePin(Dot_GPIO_Port, Dot_Pin, SET);
19 }else if (counter == (Default*2/4)){
20     display7SEG(3);
21     HAL_GPIO_WritePin(En0_GPIO_Port, En0_Pin, SET);
22     HAL_GPIO_WritePin(En1_GPIO_Port, En1_Pin, SET);
23     HAL_GPIO_WritePin(En2_GPIO_Port, En2_Pin, RESET);
24     HAL_GPIO_WritePin(En3_GPIO_Port, En3_Pin, SET);
25     HAL_GPIO_WritePin(Dot_GPIO_Port, Dot_Pin, RESET);
26 }else if (counter == (Default*1/4)){
27     display7SEG(0);
28     HAL_GPIO_WritePin(En2_GPIO_Port, En2_Pin, SET);
29     HAL_GPIO_WritePin(En3_GPIO_Port, En3_Pin, RESET);
30     HAL_GPIO_WritePin(Dot_GPIO_Port, Dot_Pin, SET);
31 }
32 //decrease counter
33 counter = (counter > 0) ? counter-1 : Default;
34
35 }
```

- Answer: According to the description, **the switching time for each 7-segment display is 0.5 seconds (500 ms)**. Since there are 4 displays, the total time to scan through all displays is:

$$T = 4 \times 0.5\text{sec} = 2\text{sec}$$

The frequency f of the scanning process is the reciprocal of the period T :

$$f = 1/T = 1/2 = 0.5\text{Hz}$$

1.3 Exercise 3

1.3.1 Report 1

Code of the update7SEG:

```
1  const int MAX_LED = 4;
2  int index_led = 0;
3  int led_buffer[4] = {1,2,3,4};
4  void update7SEG(int index){
5      switch (index){
6          case 0:
7              display7SEG(led_buffer[0]);
8              HAL_GPIO_WritePin(En0_GPIO_Port, En0_Pin, RESET);
9              HAL_GPIO_WritePin(En1_GPIO_Port, En1_Pin, SET);
10             HAL_GPIO_WritePin(En2_GPIO_Port, En2_Pin, SET);
11             HAL_GPIO_WritePin(En3_GPIO_Port, En3_Pin, SET);
12             break;
13         case 1:
14             display7SEG(led_buffer[1]);
15             HAL_GPIO_WritePin(En0_GPIO_Port, En0_Pin, SET);
16             HAL_GPIO_WritePin(En1_GPIO_Port, En1_Pin, RESET);
17             HAL_GPIO_WritePin(En2_GPIO_Port, En2_Pin, SET);
18             HAL_GPIO_WritePin(En3_GPIO_Port, En3_Pin, SET);
19             break;
20         case 2:
21             display7SEG(led_buffer[2]);
22             HAL_GPIO_WritePin(En0_GPIO_Port, En0_Pin, SET);
23             HAL_GPIO_WritePin(En1_GPIO_Port, En1_Pin, SET);
24             HAL_GPIO_WritePin(En2_GPIO_Port, En2_Pin, RESET);
25             HAL_GPIO_WritePin(En3_GPIO_Port, En3_Pin, SET);
26             break;
27         case 3:
28             display7SEG(led_buffer[3]);
29             HAL_GPIO_WritePin(En0_GPIO_Port, En0_Pin, SET);
30             HAL_GPIO_WritePin(En1_GPIO_Port, En1_Pin, SET);
31             HAL_GPIO_WritePin(En2_GPIO_Port, En2_Pin, SET);
32             HAL_GPIO_WritePin(En3_GPIO_Port, En3_Pin, RESET);
33             break;
34         default:
35             HAL_GPIO_WritePin(En0_GPIO_Port, En0_Pin, SET);
36             HAL_GPIO_WritePin(En1_GPIO_Port, En1_Pin, SET);
37             HAL_GPIO_WritePin(En2_GPIO_Port, En2_Pin, SET);
38             HAL_GPIO_WritePin(En3_GPIO_Port, En3_Pin, SET);
39             break;
40     }
41 }
```



1.3.2 Report 2

Code in the HAL_TIM_PeriodElapsedCallback:

```
1  void HAL_TIM_PeriodElapsedCallback ( TIM_HandleTypeDef * htim )
2  {
3      if (counter == Default){
4          update7SEG(index_led++);
5      }else if (counter == (Default*3/4)){
6          update7SEG(index_led++);
7      }else if (counter == (Default*2/4)){
8          update7SEG(index_led++);
9      }else if (counter == (Default*1/4)){
10         update7SEG(index_led++);
11     }
12     index_led = (index_led < MAX_LED) ? index_led : 0;
13     // g i m counter
14     counter = (counter > 0) ? counter-1 : Default;
15
16 }
```

1.4 Exercise 4

1.4.1 Report 1

```
1  const unsigned int Default = 100;
2  int counter = Default;
3  int dot_counter = 0;
4  void HAL_TIM_PeriodElapsedCallback ( TIM_HandleTypeDef * htim )
5  {
6      if (counter == Default) {
7          update7SEG(index_led++);
8          index_led = (index_led < MAX_LED) ? index_led : 0;
9      } else if (counter == (Default * 3 / 4)) {
10         HAL_GPIO_WritePin(Dot_GPIO_Port, Dot_Pin, RESET);
11     } else if (counter == (Default * 1 / 4)) {
12         HAL_GPIO_WritePin(Dot_GPIO_Port, Dot_Pin, SET);
13     }
14
15     // Reduce counter
16     counter = (counter > 0) ? counter - 1 : Default;
17
18
19     if (dot_counter >= Default) {
20         dot_counter = 0;
21         HAL_GPIO_TogglePin(Dot_GPIO_Port, Dot_Pin);
22     }
23     dot_counter++;
24 }
25
```

1.5 Exercise 5

1.5.1 Report

Code in the updateClockBuffer function

```
1 void updateClockBuffer(int hour, int minute){
2     led_buffer[0] = hour / 10;
3     led_buffer[1] = hour % 10;
4     led_buffer[2] = minute / 10;
5     led_buffer[3] = minute % 10;
6 }
```

1.6 Exercise 6

1.6.1 Report 1

Can be found at ??.

1.6.2 Report 2

```
1     int counter = 0;
2     while (1)
3     {
4         /* USER CODE END WHILE */
5         while ( counter <= 12) {
6             HAL_GPIO_WritePin ( GPIOA , (1 << ( counter + 4) ) , GPIO_PIN_RESET );
7             HAL_Delay (500) ;
8             HAL_GPIO_WritePin ( GPIOA , (1 << ( counter + 4) ) , GPIO_PIN_SET );
9             counter ++;
10        }
```

1.7 Exercise 7

```
1     HAL_GPIO_WritePin(Dot_GPIO_Port,Dot_Pin, RESET);
2     int hour = 15, minute = 59, second = 59;
3     setTimer0(1000);
4     int duration = 1000;
5     updateClockBuffer(hour, minute);
6     while (1)
7     {
8
9         /* USER CODE END WHILE */
10
11        /* USER CODE BEGIN 3 */
12        if (timer0_flag == 1){
13            HAL_GPIO_TogglePin(LED_GPIO_Port , LED_Pin);
14            HAL_GPIO_TogglePin(Dot_GPIO_Port , Dot_Pin);
15            setTimer0(2000);
```



```
16     second++;
17     if (second >= 60) {
18         second = 0;
19         minute ++;
20     }
21     if(minute >= 60) {
22         minute = 0;
23         hour ++;
24     }
25     if(hour >=24){
26         hour = 0;
27     }
28     updateClockBuffer(hour, minute);
29     setTimer0(duration);
30 }
31 }
```

1.8 Exercise 8

Extra code:

```
1  int timer0_counter = 0;
2  int timer0_flag = 1;
3  int TIMER_CYCLE = 10;
4  int timer1_counter = 0;
5  int timer1_flag = 1;
6
7  void setTimer0 (int duration){
8      timer0_counter = duration /TIMER_CYCLE;
9      timer0_flag = 0;
10 }
11
12 void setTimer1 (int duration){
13     timer1_counter = duration/TIMER_CYCLE;
14     timer1_flag = 0;
15 }
16
17 void timer_run(){
18     if (timer0_counter > 0){
19         timer0_counter--;
20         if (timer0_counter == 0) timer0_flag = 1;
21     }
22     if (timer1_counter > 0){
23         timer1_counter--;
24         if (timer1_counter == 0) timer1_flag = 1;
25     }
26 }
```

Main loop:

```
1   HAL_GPIO_WritePin(Dot_GPIO_Port, Dot_Pin, RESET);
2   int hour = 15, minute = 59, second = 59;
3   setTimer0(1000);
4   int duration = 1000;
5   updateClockBuffer(hour, minute);
6   while (1)
7   {
8
9       /* USER CODE END WHILE */
10
11      /* USER CODE BEGIN 3 */
12      if (timer0_flag == 1){
13          HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);
14          HAL_GPIO_TogglePin(Dot_GPIO_Port, Dot_Pin);
15          second++;
16          if (second >= 60) {
17              second = 0;
18              minute ++;
19          }
20          if(minute >= 60) {
21              minute = 0;
22              hour ++;
23          }
24          if(hour >=24){
25              hour = 0;
26          }
27          updateClockBuffer(hour, minute);
28          setTimer0(duration);
29      }
30      if (timer1_flag == 1){
31          update7SEG(index_led++);
32          index_led = (index_led < MAX_LED) ? index_led : 0;
33          setTimer1(duration/MAX_LED);
34      }
35  }
```

1.9 Exercise 9

1.9.1 Report 1

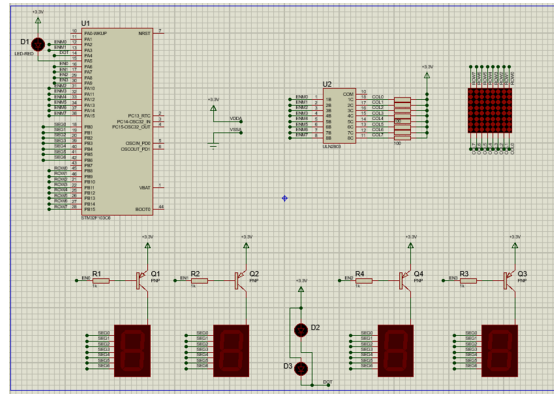


Figure 3: *Ex9*

1.9.2 Report 2

```

1  void updateLEDMatrix(int index){
2  switch (index){
3      case 0:
4          MatrixColLed(index);
5          displayLEDMatrix(index);
6          break;
7      case 1:
8          MatrixColLed(index);
9          displayLEDMatrix(index);
10         break;
11     case 2:
12         MatrixColLed(index);
13         displayLEDMatrix(index);
14         break;
15     case 3:
16         MatrixColLed(index);
17         displayLEDMatrix(index);
18         break;
19     case 4:
20         MatrixColLed(index);
21         displayLEDMatrix(index);
22         break;
23     case 5:
24         MatrixColLed(index);
25         displayLEDMatrix(index);
26         break;
27     case 6:
28         MatrixColLed(index);
29         displayLEDMatrix(index);

```

```
30     break;
31     case 7:
32         MatrixColLed(index);
33         displayLEDMatrix(index);
34         break;
35     default:
36         break;
37 }
38 }
```

1.10 Exercise 10

1. Timer variable setup:

```
1 int timer0_counter = 0;
2 int timer0_flag = 1;
3 int timer1_counter = 0;
4 int timer1_flag = 1;
5 int timer2_counter = 0;
6 int timer2_flag = 1;
7 int TIMER_CYCLE = 10;
```

-These variables are used to manage the timing functionality of the program. Each timer has a counter to track the elapsed time and a flag to indicate when the timer has expired.

-TIMER_CYCLE: This constant defines the time interval (in milliseconds) for each timer tick. It is set to 10 ms, meaning that each timer will count down in increments of 10 ms.

2. Timer setup function:

```
1 void setTimer0(int duration) {
2     timer0_counter = duration / TIMER_CYCLE;
3     timer0_flag = 0;
4 }
5
6 void setTimer1(int duration) {
7     timer1_counter = duration / TIMER_CYCLE;
8     timer1_flag = 0;
9 }
10
11 void setTimer2(int duration) {
12     timer2_counter = duration / TIMER_CYCLE;
13     timer2_flag = 0;
14 }
```

-These functions are used to initialize the timers with a specific duration. Each function takes a duration (in milliseconds) and calculates how many cycles of TIMER_CYCLE fit into that duration. It then sets the corresponding counter and resets the flag to indicate that the timer is active.

3. Timer running:

```
1 void timer_run() {
2     if (timer0_counter > 0) {
3         timer0_counter--;
4         if (timer0_counter == 0) timer0_flag = 1;
5     }
6     if (timer1_counter > 0) {
7         timer1_counter--;
8         if (timer1_counter == 0) timer1_flag = 1;
9     }
10    if (timer2_counter > 0) {
11        timer2_counter--;
12        if (timer2_counter == 0) timer2_flag = 1;
13    }
14 }
```

- This function is called periodically to decrement the counters for each timer. For each timer, if the counter is greater than zero, it is decremented. When a counter reaches zero, the corresponding flag is set to indicate that the timer has expired, allowing the program to take the necessary actions based on that timer.

4. LED Matrix array:

```
1 const uint8_t MatrixCol[8] = {0xfe, 0xfd, 0xfb, 0xf7, 0xef, 0xdf, 0xbf, 0x7f};
2 GPIO_TypeDef * RowPort[8] = {Row0_GPIO_Port, Row1_GPIO_Port, Row2_GPIO_Port,
    Row3_GPIO_Port, Row4_GPIO_Port, Row5_GPIO_Port, Row6_GPIO_Port, Row7_GPIO_Port
    };
3 const uint16_t RowPin[8] = {Row0_Pin, Row1_Pin, Row2_Pin, Row3_Pin, Row4_Pin,
    Row5_Pin, Row6_Pin, Row7_Pin};
4 GPIO_TypeDef * ColPort[8] = {Enm0_GPIO_Port, Enm1_GPIO_Port, Enm2_GPIO_Port,
    Enm3_GPIO_Port, Enm4_GPIO_Port, Enm5_GPIO_Port, Enm6_GPIO_Port, Enm7_GPIO_Port
    };
5 const uint16_t ColPin[8] = {Enm0_Pin, Enm1_Pin, Enm2_Pin, Enm3_Pin, Enm4_Pin,
    Enm5_Pin, Enm6_Pin, Enm7_Pin};
6 const int MAX_LED_MATRIX = 8;
```

-MatrixCol[]: Defines the state for each column of the LED matrix. Each entry corresponds to a column configuration for turning on/off LEDs.

-RowPort[] and ColPort[]: Arrays that map to the GPIO ports for the rows and columns of the matrix.

-RowPin[] and ColPin[]: Arrays that map to the specific pin numbers for controlling each row and column.

-MAX_LED_MATRIX: A constant representing the maximum number of LEDs (or columns) in the matrix.

5. Update LED Matrix

```
1 void displayLEDMatrix(int num) {
2     for (int i = 0; i < MAX_LED_MATRIX; i++) {
3         HAL_GPIO_WritePin(RowPort[i], RowPin[i], (matrix_buffer[num] >> i) & 0x01)
4     }
5 }
```

```
4     }  
5 }  
6  
7
```

-This function is responsible for updating the LED matrix to display the current data based on the given index. It writes the state of each row in the LED matrix by shifting the bits from the `matrix_buffer` corresponding to the specified index. Each bit in the buffer controls whether a specific LED in that row is on or off.

6. Update Matrix buffer

```
1 void updateMatrixBuffer() {  
2     matrix_buffer[7] = matrix_buffer[0];  
3     for (int i = 0; i < 7; i++) {  
4         matrix_buffer[i] = matrix_buffer[i + 1];  
5     }  
6 }  
7  
8
```

-The last element of the `matrix_buffer` is set to the first element, and then each element is moved one position to the left. This creates an effect where the displayed character appears to shift left across the LED matrix.

7. Main loop

```
1 while (1) {  
2     // USER CODE END WHILE  
3     // USER CODE BEGIN 3  
4     if (timer0_flag == 1) {  
5         updateMatrixBuffer();  
6         HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);  
7         HAL_GPIO_TogglePin(Dot_GPIO_Port, Dot_Pin);  
8         second++;  
9         if (second >= 60) {  
10            second = 0;  
11            minute++;  
12        }  
13        if (minute >= 60) {  
14            minute = 0;  
15            hour++;  
16        }  
17        if (hour >= 24) {  
18            hour = 0;  
19        }  
20        updateClockBuffer(hour, minute);  
21        updateMatrixBuffer();  
22        setTimer0(1000);  
23    }  
24    if (timer1_flag == 1) {
```

```
25     update7SEG(index_led++);
26     updateMatrixBuffer();
27     index_led = (index_led < MAX_LED) ? index_led : 0;
28     setTimer1(250);
29 }
30 if (timer2_flag == 1) {
31     setTimer2(10);
32     if (index_led_matrix >= 8) index_led_matrix = 0;
33     updateLEDMatrix(index_led_matrix++);
34 }
35 }
```

-The loop checks the flags for each timer:

- **Timer0:** When `timer0_flag` is set, it updates the matrix buffer to create a shifting effect, toggles additional LEDs (like LED and Dot), updates the clock time, and resets the timer for the next second.

-**Timer1:** When `timer1_flag` is set, it updates the display on the 7-segment display and resets the index for the next number to be displayed. It also sets a timer for 250 ms for the next update.

-**Timer2:** When `timer2_flag` is set, it updates the LED matrix based on the current index and resets the timer.



References