

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: Nguyen Dang Duy 2252116

Ho Chi Minh City, October 1, 2024



Contents

| | | |
|----------|-----------------------|-----------|
| 1 | Exercise | 2 |
| 1.1 | Exercise 1 | 2 |
| 1.1.1 | Report 1 | 2 |
| 1.1.2 | Report 2 | 2 |
| 1.2 | Exercise 2 | 4 |
| 1.2.1 | Report 1 | 4 |
| 1.2.2 | Report 2 | 4 |
| 1.3 | Exercise 3 | 6 |
| 1.3.1 | Report 1 | 6 |
| 1.3.2 | Report 2 | 7 |
| 1.4 | Exercise 4 | 8 |
| 1.4.1 | Report 1 | 8 |
| 1.5 | Exercise 5 | 8 |
| 1.5.1 | Report 1 | 8 |
| 1.6 | Exercise 6 | 9 |
| 1.6.1 | Report 1 | 9 |
| 1.6.2 | Report 2 | 9 |
| 1.6.3 | Report 3 | 9 |
| 1.7 | Exercise 7 | 10 |
| 1.8 | Exercise 8 | 10 |
| 1.9 | Exercise 9 | 12 |
| 1.10 | Exercise 10 | 14 |
| | References | 16 |

1 Exercise

The GitHub link for the lab schematics is at [here](https://github.com/11ttledino/mcu-mpu) or in this link: <https://github.com/11ttledino/mcu-mpu>.

1.1 Exercise 1

1.1.1 Report 1

This is the schematic of the Exercise 1 using Proteus:

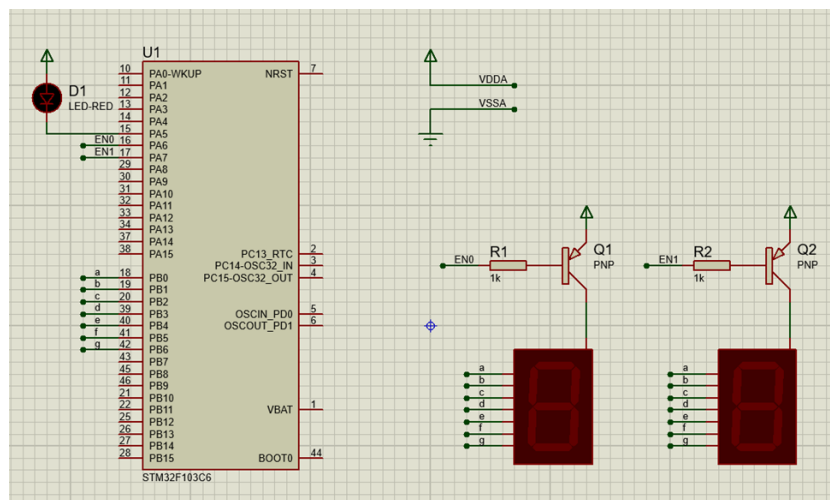


Figure 1: Schematic for exercise 1

1.1.2 Report 2

This is the source code of void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef * htim) function with some supporting functions:

```
1 void init(){
2     display7SEG(1);
3     HAL_GPIO_WritePin(en0_GPIO_Port, en0_Pin, 0);
4     HAL_GPIO_WritePin(en1_GPIO_Port, en1_Pin, 1);
5 } // Set the init stage
6
7 void toggle(){
8     HAL_GPIO_TogglePin(en0_GPIO_Port, en0_Pin);
9     HAL_GPIO_TogglePin(en1_GPIO_Port, en1_Pin);
10 } //Toggle between 2 LEDs
11
12 int clock_1 = 50; //50 * 10ms = 500 ms
13 void HAL_TIM_PeriodElapsedCallback (TIM_HandleTypeDef * htim){
14     clock_1--;
15     if (clock_1 <= 0){
```

```
16     switch (HAL_GPIO_ReadPin(en0_GPIO_Port, en0_Pin)){
17     case 0: {
18         toggle();
19         display7SEG(2);
20         break;
21     }
22     case 1:{
23         toggle();
24         display7SEG(1);
25         break;
26     }
27     }
28     clock_1 = 50;
29 }
30 }
31 int main(void)
32 {
33     HAL_Init();
34     SystemClock_Config();
35     MX_GPIO_Init();
36     MX_TIM2_Init();
37     HAL_TIM_Base_Start_IT(&htim2);
38     init();
39     while (1) {}
40 }
```

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

The frequency of the scanning process will be:

$$f = \frac{1}{T} = \frac{1}{0.5(\text{s}) + 0.5(\text{s})} = 1(\text{Hz})$$

1.2 Exercise 2

1.2.1 Report 1

This is the schematic of the Exercise 2 using Proteus:

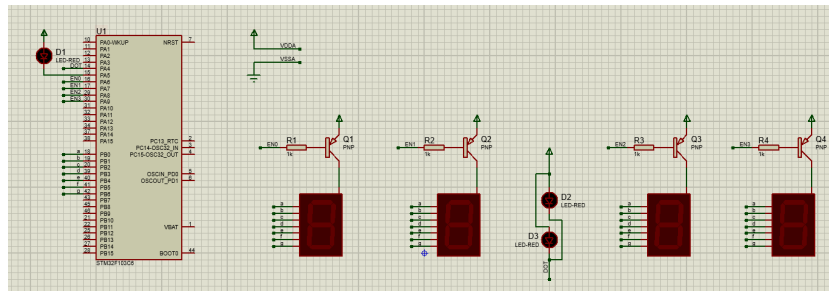


Figure 2: Schematic for exercise 2

1.2.2 Report 2

This is the source code of `void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef * htim)` function with some supporting functions:

```
1 // Exercise2
2 int clock_1 = 100;
3 int clock7Seg = 50;
4 int stage = 0; //init stage
5
6 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef * htim){
7     clock_1--;
8     if (clock_1 <= 0) { //Flag
9         HAL_GPIO_TogglePin(dot_GPIO_Port, dot_Pin);
10        clock_1 = 100;
11    } //Blinking dots
12    clock7Seg--;
13    if (clock7Seg <= 0){ //Flag
14        switch (stage){
15            case 0:
16                {
17                    HAL_GPIO_WritePin(en0_GPIO_Port, en0_Pin, 0);
18                    HAL_GPIO_WritePin(en1_GPIO_Port, en1_Pin, 1);
19                    HAL_GPIO_WritePin(en2_GPIO_Port, en2_Pin, 1);
20                    HAL_GPIO_WritePin(en3_GPIO_Port, en3_Pin, 1);
21                    display7SEG(1);
22                    stage = 1;
23                    break;
24                }
25            case 1:
26                {
27                    HAL_GPIO_WritePin(en0_GPIO_Port, en0_Pin, 1);
28                    HAL_GPIO_WritePin(en1_GPIO_Port, en1_Pin, 0);
```

```
29     HAL_GPIO_WritePin(en2_GPIO_Port, en2_Pin, 1);
30     HAL_GPIO_WritePin(en3_GPIO_Port, en3_Pin, 1);
31     display7SEG(2);
32     stage = 2;
33     break;
34 }
35 case 2:
36 {
37     HAL_GPIO_WritePin(en0_GPIO_Port, en0_Pin, 1);
38     HAL_GPIO_WritePin(en1_GPIO_Port, en1_Pin, 1);
39     HAL_GPIO_WritePin(en2_GPIO_Port, en2_Pin, 0);
40     HAL_GPIO_WritePin(en3_GPIO_Port, en3_Pin, 1);
41     display7SEG(3);
42     stage = 3;
43     break;
44 }
45 case 3:
46 {
47     HAL_GPIO_WritePin(en0_GPIO_Port, en0_Pin, 1);
48     HAL_GPIO_WritePin(en1_GPIO_Port, en1_Pin, 1);
49     HAL_GPIO_WritePin(en2_GPIO_Port, en2_Pin, 1);
50     HAL_GPIO_WritePin(en3_GPIO_Port, en3_Pin, 0);
51     display7SEG(0);
52     stage = 0;
53     break;
54 }
55 }
56 clock7Seg = 50; //reset clock
57 }
58 }
```

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

The frequency of the scanning process will be:

$$f = \frac{1}{T} = \frac{1}{0.5(\text{s}) + 0.5 \times 3(\text{s})} = 0.5(\text{Hz})$$

1.3 Exercise 3

1.3.1 Report 1

This is the source code of the void update7SEG(int index) function with some supporting functions (if any):

```
1
2 const int MAX_LED = 4;
3 int index_led = 0;
4 int led_buffer[4] = {1, 7, 0, 5};
5
6 void enSelect(int idx){ // This function is for choosing LEDs
7     switch (idx){
8         case 0:
9             {
10                 HAL_GPIO_WritePin(en0_GPIO_Port, en0_Pin, 0);
11                 HAL_GPIO_WritePin(en1_GPIO_Port, en1_Pin, 1);
12                 HAL_GPIO_WritePin(en2_GPIO_Port, en2_Pin, 1);
13                 HAL_GPIO_WritePin(en3_GPIO_Port, en3_Pin, 1);
14                 break;
15             }
16         case 1:
17             {
18                 HAL_GPIO_WritePin(en0_GPIO_Port, en0_Pin, 1);
19                 HAL_GPIO_WritePin(en1_GPIO_Port, en1_Pin, 0);
20                 HAL_GPIO_WritePin(en2_GPIO_Port, en2_Pin, 1);
21                 HAL_GPIO_WritePin(en3_GPIO_Port, en3_Pin, 1);
22                 break;
23             }
24         case 2:
25             {
26                 HAL_GPIO_WritePin(en0_GPIO_Port, en0_Pin, 1);
27                 HAL_GPIO_WritePin(en1_GPIO_Port, en1_Pin, 1);
28                 HAL_GPIO_WritePin(en2_GPIO_Port, en2_Pin, 0);
29                 HAL_GPIO_WritePin(en3_GPIO_Port, en3_Pin, 1);
30                 break;
31             }
32         case 3:
33             {
34                 HAL_GPIO_WritePin(en0_GPIO_Port, en0_Pin, 1);
35                 HAL_GPIO_WritePin(en1_GPIO_Port, en1_Pin, 1);
36                 HAL_GPIO_WritePin(en2_GPIO_Port, en2_Pin, 1);
37                 HAL_GPIO_WritePin(en3_GPIO_Port, en3_Pin, 0);
38                 break;
39             }
40         default:
41             {
42                 HAL_GPIO_WritePin(en0_GPIO_Port, en0_Pin, 1);
43                 HAL_GPIO_WritePin(en1_GPIO_Port, en1_Pin, 1);
```

```
44     HAL_GPIO_WritePin(en2_GPIO_Port, en2_Pin, 1);
45     HAL_GPIO_WritePin(en3_GPIO_Port, en3_Pin, 1);
46     break;
47 }
48 }
49 }
50
51 void update7SEG (int index){
52     if (index_led >= MAX_LED) index_led = 0; //Check index_led
53     index = index % 4;
54     switch (index){
55         case 0:{
56             enSelect(index);
57             display7SEG(led_buffer[index]);
58             break;
59         }
60         case 1:{
61             enSelect(index);
62             display7SEG(led_buffer[index]);
63             break;
64         }
65         case 2:{
66             enSelect(index);
67             display7SEG(led_buffer[index]);
68             break;
69         }
70         case 3:{
71             enSelect(index);
72             display7SEG(led_buffer[index]);
73             break;
74         }
75         default:
76             enSelect(-1); // Turn off the LED
77             break;
78     }
79 }
```

1.3.2 Report 2

This is the source code for void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef * htim) function:

```
1 int clock_1 = 50;
2 int clockBlink = 100;
3 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef * htim){
4     clock_1--;
5     clockBlink--;
6     if (clock_1 <= 0) {
7         update7SEG(index_led++);
```



```
8     clock_1 = 50;
9 }
10 if (clockBlink <= 0){
11     HAL_GPIO_TogglePin(dot_GPIO_Port, dot_Pin);
12     clockBlink = 100;
13 }
14 }
```

1.4 Exercise 4

1.4.1 Report 1

In exercise 2, the frequency of each 7-segment LED is 0.5Hz. To achieve a frequency of 1Hz on each 7-segment LED, we simply reduce the timer by half, resulting in a frequency of 1Hz:

```
1 int clock_1 = 25;
2 int clockBlink = 100;
3 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef * htim){
4     clock_1--;
5     clockBlink--;
6     if (clock_1 <= 0) {
7         update7SEG(index_led++);
8         clock_1 = 25;
9     }
10    if (clockBlink <= 0){
11        HAL_GPIO_TogglePin(dot_GPIO_Port, dot_Pin);
12        clockBlink = 100;
13    }
14 }
```

So the frequency of each 7-segment LED now will be:

$$f = \frac{1}{T} = \frac{1}{0.25(s) + 0.25 \times 3(s)} = 1(\text{Hz})$$

1.5 Exercise 5

1.5.1 Report 1

This is the source code of void updateClockBuffer():

```
1 void updateClockBuffer(){
2     led_buffer[0] = hrs / 10; // Get the 1st digit of hour
3     led_buffer[1] = hrs % 10; // Get the 2nd digit of hour
4     led_buffer[2] = min / 10; // Get the 1st digit of minutes
5     led_buffer[3] = min % 10; // Get the 2nd digit of minutes
6 }
```



1.6 Exercise 6

1.6.1 Report 1

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

The LED will not blink because the trigger flag is set **equal** to 0 instead of **less than or equal** to 0. Consequently, the flag will never be triggered, and the clock will never be reset to its default value.

1.6.2 Report 2

If in line 1 of the code above is changed to `setTimer0(1)`, what happens after that and why?

The LED will not blink because, in the `setTimer0` function, the duration is divided by the cycle. Therefore, with a duration less than 10 (the current cycle in this exercise), `timer0_counter` is set to 0. This effectively negates the purpose of the `setTimer0` function, making it similar to not having it before the `while` loop.

1.6.3 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?

The `timer0_counter` is initially set to 1. Within the first 10ms, the LED will toggle to the Off state. Subsequently, it will toggle on and off every 2 seconds within the `while` loop.

1.7 Exercise 7

This is the while loop after removing all the HAL_Delay and moving the blinking dots to while loop:

```
1 setBlink(10);
2 while (1)
3 {
4     if(clockBlink_flag == 1){ //end 1s
5         HAL_GPIO_TogglePin(dot_GPIO_Port, dot_Pin); // Toggle DOT after 1s
6         sec++;
7         if (sec >= 60) {
8             sec = 0;
9             min++;
10        }
11        if (min >= 60){
12            min = 0;
13            hrs++;
14        }
15        if (hrs >= 24) hrs = 0; // Running the clock at 1s
16        setBlink(1000); // Reset the clock
17    }
18    updateClockBuffer();
19 }
```

1.8 Exercise 8

This is the while loop after removing all the HAL_Delay and moving the blinking dots and scanning 7 LED-segments to while loop:

```
1 while (1)
2 {
3     /* USER CODE END WHILE */
4     if(clockBlink_flag == 1){ //end 1s
5         HAL_GPIO_TogglePin(dot_GPIO_Port, dot_Pin); // Toggle DOT after 1s
6         sec++;
7         if (sec >= 60) {
8             sec = 0;
9             min++;
10        }
11        if (min >= 60){
12            min = 0;
13            hrs++;
14        }
15        if (hrs >= 24) hrs = 0; // Running the clock at 1s
16
17        setBlink(1000);
18    }
19
20    if (clock0_flag == 1){ //end 0.25s
```



```
21     update7SEG(index_led++);
22     setclock(250);
23 }
24 /* USER CODE BEGIN 3 */
25     updateClockBuffer();
26
27 }
28 /* USER CODE END 3 */
29 }
```

1.9 Exercise 9

This is the code for displaying A character:

```
1 int matrix_idx = 0;
2 uint8_t matrix_buffer_A[8] = {0x3C, 0x66, 0x66, 0x7E, 0x66, 0x66, 0x66, 0x00};
3
4 /* TIMER STARTS HERE */
5 int counter = 0, flag = 0;
6
7 int const cycle = 10;
8
9 void set(int duration){
10     counter = duration / cycle;
11     flag = 0;
12 }
13
14 void run(void){
15     counter--;
16     if (counter == 0) flag = 1;
17 }
18
19 /* TIMER ENDS HERE */
20 int convertedBinary[8] = {0,0,0,0,0,0,0,0};
21 int arr[16]; //16-bit array
22 int arr2[16];
23 int shift_counter = 0;
24 void convertToBinary(uint8_t num){
25     for(int i = 7; i >= 0; i--){
26         convertedBinary[i] = num % 2;
27         num = num / 2;
28     }
29 }
30
31
32 void updateLEDMatrix(int index){
33     for (int i = 0; i < 8; i++){
34         if (index == i){
35             convertToBinary(matrix_buffer_A[i]);
36             for(int j = 0; j < 8; j++){
37                 HAL_GPIO_WritePin(ENMO_GPIO_Port, ENMO_Pin << j, arr[j]);
38             }
39             break;
40         }
41     }
42 }
43
44 void resetState(){
45     for (int i = 0; i < 8; i++){
46         HAL_GPIO_WritePin(ROW0_GPIO_Port, ROW0_Pin << i, 0);
```



```
47     }  
48 }  
49 void displayLEDMatrix(){  
50     if (matrix_idx >= 8) {  
51         matrix_idx = matrix_idx % 8;  
52         shift_counter++;  
53         if (shift_counter >= 16) shift_counter = shift_counter%16;  
54     }  
55     resetState();  
56     HAL_GPIO_WritePin(ROW0_GPIO_Port, ROW0_Pin << matrix_idx, SET);  
57     updateLEDMatrix(matrix_idx);  
58 }
```

1.10 Exercise 10

This is the code for the exercise 10:

```
1 int matrix_idx = 0;
2
3 uint8_t matrix_buffer_A[8] = {0x3C, 0x66, 0x66, 0x7E, 0x66, 0x66, 0x66, 0x00};
4
5 /* TIMER STARTS HERE */
6 int counter = 0, flag = 0;
7
8 int const cycle = 10;
9
10 void set(int duration){
11     counter = duration / cycle;
12     flag = 0;
13 }
14
15 void run(void){
16     counter--;
17     if (counter == 0) flag = 1;
18 }
19
20 /* TIMER ENDS HERE */
21 int convertedBinary[8] = {0,0,0,0,0,0,0,0};
22 int arr[16]; //16-bit array
23 int arr2[16];
24 int shift_counter = 0;
25 void convertToBinary(uint8_t num){
26     for(int i = 7; i >= 0; i--){
27         convertedBinary[i] = num % 2;
28         num = num / 2;
29     }
30 }
31 void convertShift(uint8_t num){
32     for(int i = 15; i >= 0; i--){
33         arr[i] = num % 2;
34         arr2[i] = arr[i];
35         num = num / 2;
36     }
37     for (int i = 0; i < 16; i++){
38         int newidx = i - shift_counter;
39         if (newidx >= 0) arr[newidx] = arr2[i];
40         else arr[15 - shift_counter + 1] = arr2[i];
41     }
42 }
43
44
45 void updateLEDMatrix(int index){
46     for (int i = 0; i < 8; i++){
```



```
47     if (index == i){
48         //     convertToBinary(matrix_buffer_A[i]);
49         convertShift(matrix_buffer_A[i]);
50         for(int j = 0; j < 8; j++){
51             HAL_GPIO_WritePin(ENMO_GPIO_Port, ENMO_Pin << j, arr[j + 4]);
52         }
53         break;
54     }
55 }
56 }
57
58 void resetState(){
59     for (int i = 0; i < 8; i++){
60         HAL_GPIO_WritePin(ROW0_GPIO_Port, ROW0_Pin << i, 0);
61     }
62 }
63 void displayLEDMatrix(){
64     if (matrix_idx >= 8) {
65         matrix_idx = matrix_idx % 8;
66         shift_counter++;
67         if (shift_counter >= 16) shift_counter = shift_counter%16;
68     }
69     resetState();
70     HAL_GPIO_WritePin(ROW0_GPIO_Port, ROW0_Pin << matrix_idx, SET);
71     updateLEDMatrix(matrix_idx);
72 }
```




References