

TRƯỜNG ĐẠI HỌC BÁCH KHOA – TP. HỒ CHÍ MINH
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH



Lab 2

Timer Interrupt and LED Scanning

Microcontroller - Microprocessor (Lab)

COURSE ID: CO3010 - HK251

Name: Trương Gia Hy - Student ID: 2352458

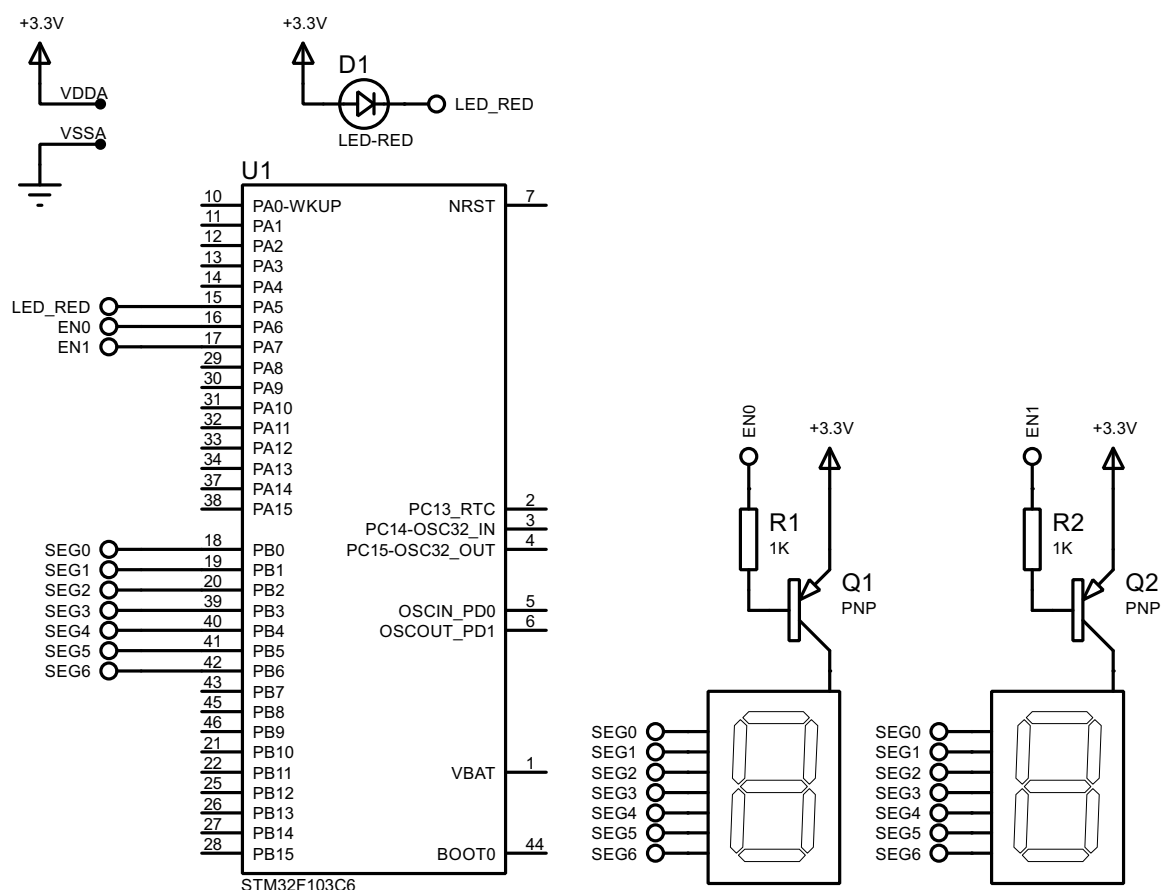
Supervising Lecturer: Dr. VÕ TUẤN BÌNH

1 Exercise

The GitHub link for the lab files: https://github.com/hygameo/VXL-VDK_2352458

1.1 Exercise 1

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



1.1.2 Report 2: Present your source code.

```

1 uint8_t segmentMap[10] = {
2     0b1111110, 0b0110000,
3     0b1101101, 0b1111001,
4     0b0110011, 0b1011011,
5     0b1011111, 0b1110000,
6     0b1111111, 0b1111011
7 };
8
9 uint8_t SegPin[7] = {
10     SEG_A_Pin, SEG_B_Pin,
11     SEG_C_Pin, SEG_D_Pin,
12     SEG_E_Pin, SEG_F_Pin,
13     SEG_G_Pin
14 };
15
16 int counter = 0;

```

```
17 int DisplayNum = 0;
18
19 void display7SEG(int num) {
20     uint8_t bitmask = segmentMap[num];
21
22     for(int i = 0; i < 7; i++) HAL_GPIO_WritePin(GPIOB, SegPin[i], (
23         bitmask & (1 << (6 - i))) ? RESET : SET);
24 }
25
26 void HAL_TIM_PeriodElapsedCallback ( TIM_HandleTypeDef * htim )
27 {
28     counter++;
29     if(counter >= 50){
30         counter = 0;
31
32         HAL_GPIO_WritePin(GPIOA, EN0_Pin | EN1_Pin, GPIO_PIN_SET);
33
34         if(DisplayNum == 0){
35             HAL_GPIO_WritePin(GPIOA, EN0_Pin, GPIO_PIN_RESET);
36             display7SEG(1);
37         }
38         if(DisplayNum == 1){
39             HAL_GPIO_TogglePin(GPIOA, LED_RED_Pin);
40             HAL_GPIO_WritePin(GPIOA, EN1_Pin, GPIO_PIN_RESET);
41             display7SEG(2);
42         }
43
44         DisplayNum = (DisplayNum + 1) % 2;
45     }
46 }
```

1.1.3 Question: What is the frequency of the scanning process?

The frequency of the scanning process is 1 Hz, as the switching time between the two seven-segment displays is half a second (0.5 seconds), resulting in one complete cycle (switching from the first to the second and back) every 1 second.

1.2.3 Question: What is the frequency of the scanning process?

The frequency of the scanning process is 0.5 Hz. This is because each seven-segment LED is displayed for 500ms (0.5 seconds), and the system cycles through all 4 LEDs, completing one full scan every 2 seconds ($4 \times 0.5s = 2s$). The frequency is calculated as $\frac{1}{2}$ Hz.

1.3 Exercise 3

1.3.1 Report 1: Present the source code of the update7SEG function.

```
1 void update7SEG(int index){
2     HAL_GPIO_WritePin(GPIOA, EN0_Pin | EN1_Pin | EN2_Pin | EN3_Pin,
3         GPIO_PIN_SET);
4
5     switch (index){
6         case 0:
7             HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, GPIO_PIN_RESET);
8             display7SEG(led_buffer[0]);
9             break;
10        case 1:
11            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, GPIO_PIN_RESET);
12            display7SEG(led_buffer[1]);
13            break;
14        case 2:
15            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_RESET);
16            display7SEG(led_buffer[2]);
17            break;
18        case 3:
19            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, GPIO_PIN_RESET);
20            display7SEG(led_buffer[3]);
21            break;
22        default:
23            break;
24    }
```

1.3.2 Report 2: Present the source code.

```
1 void HAL_TIM_PeriodElapsedCallback ( TIM_HandleTypeDef * htim )
2 {
3     counter++;
4     if(counter0 >= 50){
5         counter = 0;
6
7         if(index_led == 1 || index_led == 3) HAL_GPIO_TogglePin(GPIOA,
8             LED_RED_Pin | DOT_Pin);
9         update7SEG(index_led++);
10        if(index_led >= 4) index_led = 0;
11    }
```

1.4 Exercise 4

1.4.1 Report 1: Present the source code.

```
1 void HAL_TIM_PeriodElapsedCallback ( TIM_HandleTypeDef * htim )
2 {
3     counter++;
4     if(counter >= 25){
5         counter = 0;
6
7         if(index_led == 3) HAL_GPIO_TogglePin(GPIOA, LED_RED_Pin |
8             DOT_Pin);
9         update7SEG(index_led++);
10        if(index_led >= 4) index_led = 0;
11    }
```

1.5 Exercise 5

1.5.1 Report 1: Present the source code in the updateClockBuffer function.

```
1 void updateClockBuffer() {
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: if in line 1 of the code above is miss, what happens after that and why?

If the line `setTimer0(1000)` is removed before the while (1) loop, the global variable `timer0_counter` will remain at its default value of 0. As a result, when `timer_run()` is called, the condition `if (timer0_counter > 0)` will be false, and `timer0_counter` will not decrement. Consequently, `timer0_flag` will stay at 0 and never be set to 1. In the while (1) loop, the condition `if (timer0_flag == 1)` will never be true, so `HAL_GPIO_TogglePin(GPIOA, LED_RED_Pin)` and `setTimer0(2000)` will not execute.

Since `LED_RED` is connected with one pin to 3.3V and the signal pin to `PA5`, and `PA5` remains in its initial state (`GPIO_PIN_RESET`), the LED will turn on continuously. This happens because the timer mechanism does not start without initializing `timer0_counter` with a positive 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?

If `setTimer0(1000)` is changed to `setTimer0(1)`, the `LED_RED` will not toggle at all, and the program will remain in an idle state within the while (1) loop. This occurs because `timer0_counter` is set to 0 due to integer division, preventing the `timer_run` function from decrementing it or setting `timer0_flag` to 1, thus blocking the intended logic from executing. The LED will stay active like Report 1.

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?

Compared to first question (`setTimer0(1000)`): The initial delay changes from 1 second to 10ms, making the first LED toggle occur much sooner, though subsequent toggles remain on a 2-second interval.

Compared to second question (`setTimer0(1)`): The system now functions instead of being stuck, because `timer0_counter` is 1 rather than 0, allowing the timer to run.

Reason: The behavior depends on the value of `timer0_counter` set by `duration / TIMER_CYCLE`. A value of 1 (from `setTimer0(10)`) provides a minimal but functional delay, while 100 (from `setTimer0(1000)`) gives a 1-second delay, and 0 (from `setTimer0(1)`) disables the timer.

1.7 Exercise 7

1.7.1 Report 1: Present the source code of this function.

```
1 void clearAllClock(void) {
2     for (int i = 0; i < 12; i++) HAL_GPIO_WritePin(GPIOA, ACLOCK_Pins[
        i], GPIO_PIN_RESET);
3 }
```

1.8 Exercise 8

1.8.1 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.

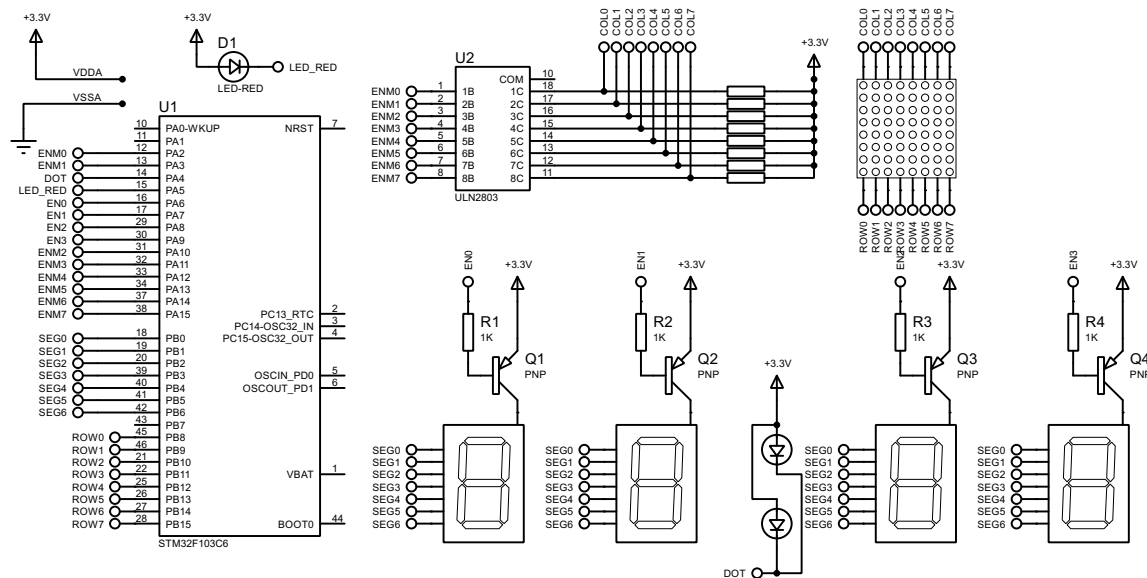
```
1 int timer0_counter = 0;
2 int timer0_flag = 0;
3 int timer1_counter = 0;
4 int timer1_flag = 0;
5 int TIMER_CYCLE = 10;
6
7 void setTimer0 ( int duration ){
8     timer0_counter = duration / TIMER_CYCLE ;
9     timer0_flag = 0;
10 }
11
12 void setTimer1(int duration) {
13     timer1_counter = duration / 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 }
27
28 int main(void)
29 {
30     setTimer0(1000) ;
31     setTimer1(250) ;
32     while (1)
33     {
34         if(timer0_flag == 1){
35             if(counter >= 2){
36                 HAL_GPIO_TogglePin(GPIOA, LED_RED_Pin);
37                 counter = 0;
38             }
39         }
40     }
```



```
39         counter++;
40         updateClockBuffer();
41         second ++;
42
43         if ( second >= 60) {
44             second = 0;
45             minute++;
46         }
47         if( minute >= 60) {
48             minute = 0;
49             hour++;
50         }
51         if( hour >= 24){
52             hour = 0;
53         }
54
55         HAL_GPIO_TogglePin(GPIOA , DOT_Pin);
56         setTimer0(1000) ;
57
58     }
59     if(timer1_flag == 1){
60         update7SEG(index_led++);
61         if(index_led >= 4) index_led = 0;
62         setTimer1(250) ;
63     }
64 }
65 }
```

1.9 Exercise 9

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



1.9.2 Report 2: Display character "A".

```

1  const int MAX_LED_MATRIX = 8;
2  int index_led_matrix = 0;
3  uint8_t matrix_buffer [8] = {0x00, 0x00, 0x7C, 0x12, 0x12, 0x7C, 0x00,
4  0x00};
5  void updateLEDMatrix (int index){
6      HAL_GPIO_WritePin(GPIOA, ENM0_Pin | ENM1_Pin | ENM2_Pin | ENM3_Pin
7      | ENM4_Pin | ENM5_Pin | ENM6_Pin | ENM7_Pin, GPIO_PIN_SET);
8
9      HAL_GPIO_WritePin(GPIOA, ENMPin[index], GPIO_PIN_RESET);
10
11     for(int i = 0; i < 8; i++) HAL_GPIO_WritePin(GPIOB, ROWPin[i], (
12         matrix_buffer[index] & (1 << i)) ? GPIO_PIN_RESET : GPIO_PIN_SET
13     );
14 }
15
16 int main(void)
17 {
18     setTimer0(1000) ;
19     setTimer1(250) ;
20     setTimer2(10) ;
21     while (1)
22     {
23         if(timer0_flag == 1){
24             if(counter >= 2){
25                 HAL_GPIO_TogglePin(GPIOA, LED_RED_Pin);
26                 counter = 0;
27             }
28             counter++;
29         }
30     }
31 }

```

```
26         updateClockBuffer();
27         second ++;
28
29         if ( second >= 60) {
30             second = 0;
31             minute++;
32         }
33         if( minute >= 60) {
34             minute = 0;
35             hour++;
36         }
37         if( hour >= 24){
38             hour = 0;
39         }
40
41         HAL_GPIO_TogglePin(GPIOA , DOT_Pin);
42         setTimer0(1000) ;
43
44     }
45     if(timer1_flag == 1){
46         update7SEG(index_led++);
47         if(index_led >= 4) index_led = 0;
48         setTimer1(250) ;
49     }
50     if(timer2_flag == 1){
51         updateLEDMatrix(index_led_matrix);
52         index_led_matrix = (index_led_matrix + 1) % MAX_LED_MATRIX
53         ;
54         setTimer2(10) ;
55     }
56 }
```

1.10 Exercise 10

1.10.1 Report 1: Present your source code in the report.

```
1  uint8_t segmentMap[10] = {
2      0b1111110, 0b0110000,
3      0b1101101, 0b1111001,
4      0b0110011, 0b1011011,
5      0b1011111, 0b1110000,
6      0b1111111, 0b1111011
7  };
8
9  uint8_t SegPin[7] = {
10     SEG_A_Pin, SEG_B_Pin,
11     SEG_C_Pin, SEG_D_Pin,
12     SEG_E_Pin, SEG_F_Pin,
13     SEG_G_Pin
14 };
15
16 uint16_t ENMPin[8] = {
17     ENM0_Pin, ENM1_Pin,
18     ENM2_Pin, ENM3_Pin,
19     ENM4_Pin, ENM5_Pin,
20     ENM6_Pin, ENM7_Pin
21 };
22
23 uint16_t ROWPin[8] = {
24     ROW0_Pin, ROW1_Pin,
25     ROW2_Pin, ROW3_Pin,
26     ROW4_Pin, ROW5_Pin,
27     ROW6_Pin, ROW7_Pin
28 };
29
30 int timer0_counter = 0;
31 int timer0_flag = 0;
32 int timer1_counter = 0;
33 int timer1_flag = 0;
34 int timer2_counter = 0;
35 int timer2_flag = 0;
36 int TIMER_CYCLE = 10;
37
38 const int MAX_LED = 4;
39 int index_led = 0;
40 int led_buffer [4] = {1, 2, 3, 4};
41 int hour = 15, minute = 8, second = 50;
42
43 const int MAX_LED_MATRIX = 8;
44 int index_led_matrix = 0;
45 uint8_t matrix_buffer [8] = {0x00, 0x00, 0x7C, 0x12, 0x12, 0x7C, 0x00,
46     0x00};
47
48 int counter = 0;
49 void setTimer0 ( int duration ){
```

```
50     timer0_counter = duration / TIMER_CYCLE ;
51     timer0_flag = 0;
52 }
53
54 void setTimer1(int duration) {
55     timer1_counter = duration / TIMER_CYCLE;
56     timer1_flag = 0;
57 }
58 void setTimer2(int duration) {
59     timer2_counter = duration / TIMER_CYCLE;
60     timer2_flag = 0;
61 }
62
63 void timer_run() {
64     if (timer0_counter > 0) {
65         timer0_counter--;
66         if (timer0_counter == 0) timer0_flag = 1;
67     }
68     if (timer1_counter > 0) {
69         timer1_counter--;
70         if (timer1_counter == 0) timer1_flag = 1;
71     }
72     if (timer2_counter > 0) {
73         timer2_counter--;
74         if (timer2_counter == 0) timer2_flag = 1;
75     }
76 }
77
78 void display7SEG(int num) {
79     for(int i = 0; i < 7; i++) HAL_GPIO_WritePin(GPIOB, SegPin[i], (
80         segmentMap[num] & (1 << (6 - i))) ? RESET : SET);
81 }
82
83 void update7SEG(int index){
84     HAL_GPIO_WritePin(GPIOA, EN0_Pin | EN1_Pin | EN2_Pin | EN3_Pin,
85         GPIO_PIN_SET);
86
87     switch (index){
88         case 0:
89             HAL_GPIO_WritePin(GPIOA, EN0_Pin, GPIO_PIN_RESET);
90             display7SEG(led_buffer[0]);
91             break;
92         case 1:
93             HAL_GPIO_WritePin(GPIOA, EN1_Pin, GPIO_PIN_RESET);
94             display7SEG(led_buffer[1]);
95             break;
96         case 2:
97             HAL_GPIO_WritePin(GPIOA, EN2_Pin, GPIO_PIN_RESET);
98             display7SEG(led_buffer[2]);
99             break;
100         case 3:
101             HAL_GPIO_WritePin(GPIOA, EN3_Pin, GPIO_PIN_RESET);
102             display7SEG(led_buffer[3]);
103             break;
```

```
102     default:
103     break;
104 }
105 }
106
107 void updateClockBuffer() {
108     led_buffer[0] = hour / 10;
109     led_buffer[1] = hour % 10;
110     led_buffer[2] = minute / 10;
111     led_buffer[3] = minute % 10;
112 }
113
114 void updateLEDMatrix (int index){
115     HAL_GPIO_WritePin(GPIOA, ENM0_Pin | ENM1_Pin | ENM2_Pin | ENM3_Pin
116         | ENM4_Pin | ENM5_Pin | ENM6_Pin | ENM7_Pin, GPIO_PIN_SET);
117
118     HAL_GPIO_WritePin(GPIOA, ENMPin[index], GPIO_PIN_RESET);
119
120     for(int i = 0; i < 8; i++) HAL_GPIO_WritePin(GPIOB, ROWPin[i], (
121         matrix_buffer[index] & (1 << i)) ? GPIO_PIN_RESET : GPIO_PIN_SET
122     );
123 }
124
125 void shiftArray(uint8_t *array, int size) {
126     uint8_t temp = array[0];
127     for (int i = 0; i < size - 1; i++) array[i] = array[i + 1];
128     array[size - 1] = temp;
129 }
130
131 int main(void)
132 {
133     setTimer0(1000) ;
134     setTimer1(250) ;
135     setTimer2(20) ;
136     while (1)
137     {
138         /* USER CODE END WHILE */
139
140         /* USER CODE BEGIN 3 */
141         if(timer0_flag == 1){
142             if(counter >= 2){
143                 HAL_GPIO_TogglePin(GPIOA, LED_RED_Pin);
144                 counter = 0;
145             }
146             counter++;
147             updateClockBuffer();
148             second ++;
149
150             if ( second >= 60) {
151                 second = 0;
152                 minute++;
153             }
154             if( minute >= 60) {
155                 minute = 0;
```

```
153         hour++;
154     }
155     if( hour >= 24){
156         hour = 0;
157     }
158
159     HAL_GPIO_TogglePin(GPIOA , DOT_Pin);
160     setTimer0(1000) ;
161
162 }
163 if(timer1_flag == 1){
164     update7SEG(index_led++);
165     if(index_led >= 4) index_led = 0;
166     shiftArray(matrix_buffer, MAX_LED_MATRIX);
167     setTimer1(250) ;
168 }
169 if(timer2_flag == 1){
170     updateLEDMatrix(index_led_matrix);
171     index_led_matrix = (index_led_matrix + 1) % MAX_LED_MATRIX
172     ;
173     setTimer2(20) ;
174 }
175 }
```

1.10.2 Report 2: Briefly describe your solution.

To make the letter "A" displayed on the LED matrix shift to the left, I utilized the `shiftArray` function in the code. The solution involves updating the `matrix_buffer` array, which stores the 8-bit patterns representing the LED matrix columns, to shift its contents leftward over time. Specifically, the `shiftArray` function moves each element one position to the left, with the first element moving to the last position, effectively creating a leftward scrolling effect. In the while (1) loop, the `timer1_flag` check calls `shiftArray(matrix_buffer, MAX_LED_MATRIX)` every 250ms (as set by `setTimer1(250)`), ensuring the pattern shifts incrementally. The initial `matrix_buffer` is preloaded with a pattern representing the letter "A" (e.g., 0x7C, 0x12, 0x12, 0x7C in the middle), and the continuous shifting moves this pattern left across the 8-column matrix, updating the display via `updateLEDMatrix` to visually scroll the letter "A" to the left.