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, October 1, 2024

Contents

1 Exercise

The GitHub link for the lab schematics is at here or in this link: https://github.com/thanhbinh0710/VXL.git.

The schematic for the exercises:

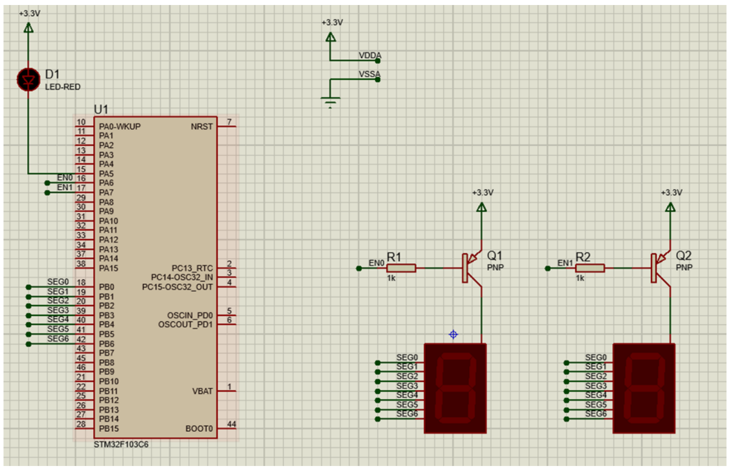

Figure 1: *EX1*

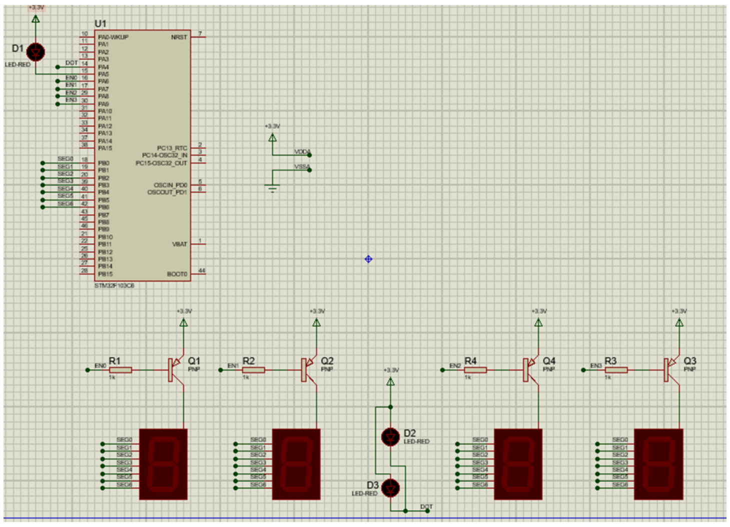

Figure 2: *EX2*

1.1 Exercise 1

1.1.1 Report 1

Can be found at 1.

1.1.2 Report 2

```c
int counter = 100;
void HAL_TIM_PeriodElapsedCallback ( TIM_HandleTypeDef * htim )
{
  if (counter == 100){
    display7SEG(1);
    HAL_GPIO_WritePin(En0_GPIO_Port, En0_Pin, RESET);
    HAL_GPIO_WritePin(En1_GPIO_Port, En1_Pin, SET);
  }else if (counter == 50){
    display7SEG(2);
    HAL_GPIO_WritePin(En0_GPIO_Port, En0_Pin, SET);
    HAL_GPIO_WritePin(En1_GPIO_Port, En1_Pin, RESET);
  }
  counter = (counter > 0) ? counter-1 : 100;
}
```

- 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 = 4Hz

1.2 Exercise 2

1.2.1 Report 1

Can be found at 2.

1.2.2 Report 2

```c
const unsigned int Default = 200;
int counter = Default;
void HAL_TIM_PeriodElapsedCallback ( TIM_HandleTypeDef * htim )
{
  if (counter == Default){
    display7SEG(1);
    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);
    HAL_GPIO_WritePin(Dot_GPIO_Port,Dot_Pin, RESET);
  }else if (counter == (Default*3/4)){
    display7SEG(2);
    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);
    HAL_GPIO_WritePin(Dot_GPIO_Port,Dot_Pin, SET);
  }else if (counter == (Default*2/4)){
    display7SEG(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, RESET);
    HAL_GPIO_WritePin(En3_GPIO_Port, En3_Pin, SET);
    HAL_GPIO_WritePin(Dot_GPIO_Port,Dot_Pin, RESET);
  }else if (counter == (Default*1/4)){
    display7SEG(0);
    HAL_GPIO_WritePin(En2_GPIO_Port, En2_Pin, SET);
    HAL_GPIO_WritePin(En3_GPIO_Port, En3_Pin, RESET);
    HAL_GPIO_WritePin(Dot_GPIO_Port,Dot_Pin, SET);
  }
  //decrease counter
  counter = (counter > 0) ? counter-1 : Default;
    if (dot_counter >= Default) {
          dot_counter = 0;
          HAL_GPIO_TogglePin(Dot_GPIO_Port, Dot_Pin);
      }
      dot_counter++;

}
```

- 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 x 0.5sec = 2sec

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

f = 1/T = 1/2 = 0.5Hz

1.3 Exercise 3

1.3.1 Report 1

Code of the update7SEG:

```
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:
    display7SEG(led_buffer[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:
    display7SEG(led_buffer[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:
    display7SEG(led_buffer[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:
    display7SEG(led_buffer[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:
    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, SET);
    break;
  }
}
```

1.3.2 Report 2

Code in the HAL_TIM_PeriodElapsedCallback:

```c
    void HAL_TIM_PeriodElapsedCallback ( TIM_HandleTypeDef * htim )
{
  if (counter == Default){
    update7SEG(index_led++);
  }else if (counter == (Default*3/4)){
    update7SEG(index_led++);
  }else if (counter == (Default*2/4)){
    update7SEG(index_led++);
  }else if (counter == (Default*1/4)){
    update7SEG(index_led++);
  }
  index_led = (index_led < MAX_LED) ? index_led : 0;
  // g i m  counter
  counter = (counter > 0) ? counter-1 : Default;
    if (dot_counter >= Default) {
          dot_counter = 0;
          HAL_GPIO_TogglePin(Dot_GPIO_Port, Dot_Pin);
      }
      dot_counter++;

}
```

1.4 Exercise 4

1.4.1 Report 1

```c
const unsigned int Default = 100;
int counter = Default;
int dot_counter = 0;
void HAL_TIM_PeriodElapsedCallback ( TIM_HandleTypeDef * htim )
{
  if (counter == Default) {
          update7SEG(index_led++);
          index_led = (index_led < MAX_LED) ? index_led : 0;
      } else if (counter == (Default * 3 / 4)) {
          HAL_GPIO_WritePin(Dot_GPIO_Port, Dot_Pin, RESET);
      } else if (counter == (Default * 1 / 4)) {
          HAL_GPIO_WritePin(Dot_GPIO_Port, Dot_Pin, SET);
      }

      // Reduce counter
      counter = (counter > 0) ? counter - 1 : Default;


      if (dot_counter >= Default) {
          dot_counter = 0;
          HAL_GPIO_TogglePin(Dot_GPIO_Port, Dot_Pin);
      }
      dot_counter++;
}

```

1.5 Exercise 5

1.5.1 Report

Code in the updateClockBuffer function

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

1.6 Exercise 6

1.6.1 Report 1

If the `setTimer0(1000)` line is missing, the timer will not be initialized, and the value of `timer0_counter` will remain 0 (or whatever previous value was left). This means that `timer0_flag` will never be reset to control the time interval between the LED toggling. As a result, the LED will never blink because the condition `if(timer0_flag == 1)` will never be true, since `timer0_flag` is not being set through the countdown of the timer.

1.6.2 Report 2

If `setTimer0(1)` is used instead of `setTimer0(1000)`, the software timer will be set with a very short delay, as it will only wait for 1 ms (instead of 1000 ms) before setting `timer0_flag` to 1. As a result, the LED will blink much faster, toggling almost immediately after each iteration of the loop. Essentially, the LED will blink at a very high frequency, which might be too fast for the human eye to detect individual blinks, making the LED appear to be continuously on.

1.6.3 Report 3

When `setTimer0(10)` is used instead of `setTimer0(1000)`:

1. **Compared to the original setTimer0(1000)**: The timer will now be set for 10 ms instead of 1000 ms. This means the LED will blink much faster. Instead of waiting for 1 second (1000 ms) before toggling, it will only wait for 10 ms. The result is that the LED will blink at a much higher frequency, about 100 times faster than in the original setup, potentially too fast for the human eye to perceive individual blinks.
2. **Compared to setTimer0(1)**: The difference is less dramatic compared to `setTimer0(1)`. Instead of waiting for just 1 ms, the timer will now wait for 10 ms. The LED will still blink quickly, but it will be 10 times slower than the case with `setTimer0(1)`. This might still be too fast for the human eye to perceive as distinct blinks, but it will be slower than the 1 ms delay.

In both cases, the LED will blink much faster than with the original 1000 ms delay, but `setTimer0(10)` provides a more moderate speed compared to `setTimer0(1)`.

1.7 Exercise 7

```
HAL_GPIO_WritePin(Dot_GPIO_Port,Dot_Pin, RESET);
int hour = 15, minute =  59, second = 59;
setTimer0(1000);
int duration = 1000;
updateClockBuffer(hour, minute);
while (1)
{

  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
if (timer0_flag == 1){
  HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);
  HAL_GPIO_TogglePin(Dot_GPIO_Port, Dot_Pin);
  setTimer0(2000);
  second++;
     if (second >= 60) {
       second = 0;
       minute ++;
     }
     if(minute >= 60) {
       minute = 0;
       hour ++;
     }
     if(hour >=24){
       hour = 0;
     }
      updateClockBuffer(hour, minute);
      setTimer0(duration);
}
}
```

1.8 Exercise 8

Extra code:

```c
int timer0_counter = 0;
int timer0_flag = 1;
int TIMER_CYCLE = 10;
int timer1_counter = 0;
int timer1_flag = 1;

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;
  }
}
```

Main loop:

```
  HAL_GPIO_WritePin(Dot_GPIO_Port,Dot_Pin, RESET);
int hour = 15, minute =  59, second = 59;
setTimer0(1000);
int duration = 1000;
updateClockBuffer(hour, minute);
while (1)
{

  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
if (timer0_flag == 1){
  HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);
  HAL_GPIO_TogglePin(Dot_GPIO_Port, Dot_Pin);
  second++;
     if (second >= 60) {
       second = 0;
       minute ++;
     }
     if(minute >= 60) {
       minute = 0;
       hour ++;
     }
     if(hour >=24){
       hour = 0;
     }
      updateClockBuffer(hour, minute);
      setTimer0(duration);
}
if (timer1_flag == 1){
  update7SEG(index_led++);
  index_led = (index_led < MAX_LED) ? index_led : 0;
  setTimer1(duration/MAX_LED);
}
}
```

1.9 Exercise 9

1.9.1 Report 1

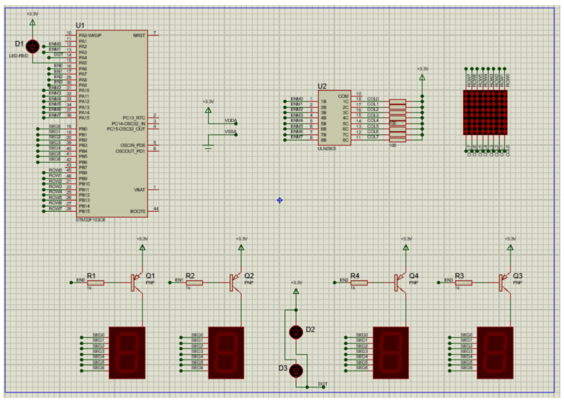

Figure 3: *Ex9*

1.9.2 Report 2

```
   void updateLEDMatrix(int index){
 switch (index){
  case 0:
    MatrixColLed(index);
    displayLEDMatrix(index);
    break;
  case 1:
    MatrixColLed(index);
    displayLEDMatrix(index);
    break;
  case 2:
    MatrixColLed(index);
    displayLEDMatrix(index);
    break;
  case 3:
    MatrixColLed(index);
    displayLEDMatrix(index);
    break;
  case 4:
    MatrixColLed(index);
    displayLEDMatrix(index);
    break;
  case 5:
    MatrixColLed(index);
    displayLEDMatrix(index);
    break;
  case 6:
    MatrixColLed(index);
    displayLEDMatrix(index);
```

```
      break;
    case 7:
      MatrixColLed(index);
      displayLEDMatrix(index);
      break;
    default:
      break;
  }
}
```

1.10 Exercise 10

1. Timer variable setup:

```
int timer0_counter = 0;
int timer0_flag = 1;
int timer1_counter = 0;
int timer1_flag = 1;
int timer2_counter = 0;
int timer2_flag = 1;
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:

```
    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 setTimer2(int duration) {
    timer2_counter = duration / TIMER_CYCLE;
    timer2_flag = 0;
}
```

-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:

```c
    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;
    }
    if (timer2_counter > 0) {
        timer2_counter--;
        if (timer2_counter == 0) timer2_flag = 1;
    }
}
```

- 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:

```c
const uint8_t MatrixCol[8] = {0xfe, 0xfd, 0xfb, 0xf7, 0xef, 0xdf, 0xbf, 0x7f};
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
    };
const uint16_t RowPin[8] = {Row0_Pin, Row1_Pin, Row2_Pin, Row3_Pin, Row4_Pin,
    Row5_Pin, Row6_Pin, Row7_Pin};
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
    };
const uint16_t ColPin[8] = {Enm0_Pin, Enm1_Pin, Enm2_Pin, Enm3_Pin, Enm4_Pin,
    Enm5_Pin, Enm6_Pin, Enm7_Pin};
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

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


```

-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

```
    void updateMatrixBuffer() {
    matrix_buffer[7] = matrix_buffer[0];
    for (int i = 0; i < 7; i++) {
        matrix_buffer[i] = matrix_buffer[i + 1];
    }
}


```

-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

```
    while (1) {
    // USER CODE END WHILE
    // USER CODE BEGIN 3
    if (timer0_flag == 1) {
        updateMatrixBuffer();
        HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);
        HAL_GPIO_TogglePin(Dot_GPIO_Port, Dot_Pin);
        second++;
        if (second >= 60) {
            second = 0;
            minute++;
        }
        if (minute >= 60) {
            minute = 0;
            hour++;
        }
        if (hour >= 24) {
            hour = 0;
        }
        updateClockBuffer(hour, minute);
        updateMatrixBuffer();
        setTimer0(1000);
    }
    if (timer1_flag == 1) {
        update7SEG(index_led++);
        updateMatrixBuffer();
        index_led = (index_led < MAX_LED) ? index_led : 0;
        setTimer1(250);
    }
    if (timer2_flag == 1) {
        setTimer2(10);
        if (index_led_matrix >= 8) index_led_matrix = 0;
        updateLEDMatrix(index_led_matrix++);
    }
}
```

-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