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

The GitHub link for the lab schematics is at here or in this link: https://github.com/l1ttled1no/mcu-mpu.

1.1 Exercise 1

1.1.1 Report 1

This is the schematic of the Exercise 1 using Proteus:

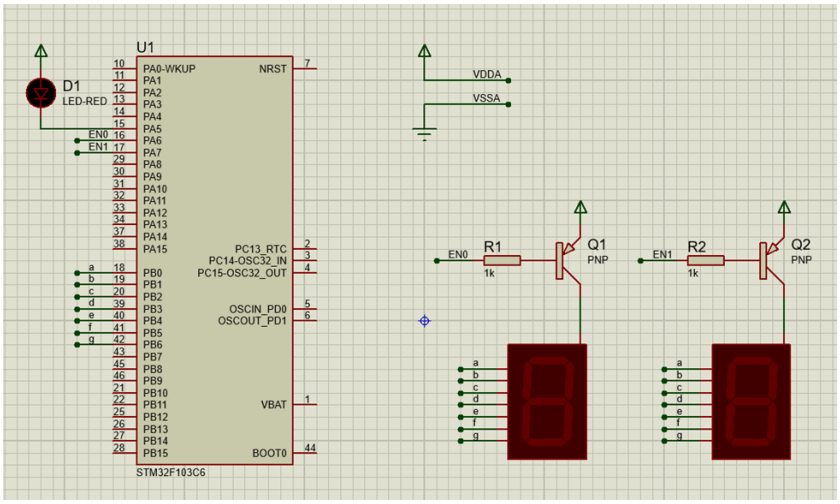

Figure 1: *Schematic for exercise 1*

1.1.2 Report 2

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

```
void init(){
  display7SEG(1);
  HAL_GPIO_WritePin(en0_GPIO_Port, en0_Pin, 0);
  HAL_GPIO_WritePin(en1_GPIO_Port, en1_Pin, 1);
} // Set the init stage

void toggle(){
  HAL_GPIO_TogglePin(en0_GPIO_Port, en0_Pin);
  HAL_GPIO_TogglePin(en1_GPIO_Port, en1_Pin);
} //Toggle between 2 LEDs

int clock_1 = 50; //50 * 10ms = 500 ms
void HAL_TIM_PeriodElapsedCallback (TIM_HandleTypeDef * htim){
  clock_1--;
  if (clock_1 <= 0){
```

```
    switch (HAL_GPIO_ReadPin(en0_GPIO_Port, en0_Pin)){
    case 0: {
      toggle();
      display7SEG(2);
      break;
      }
    case 1:{
      toggle();
      display7SEG(1);
            break;
      }
    }
    clock_1 = 50;
  }
}
int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_TIM2_Init();
  HAL_TIM_Base_Start_IT(&htim2);
  init();
  while (1) {}
}
```

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:

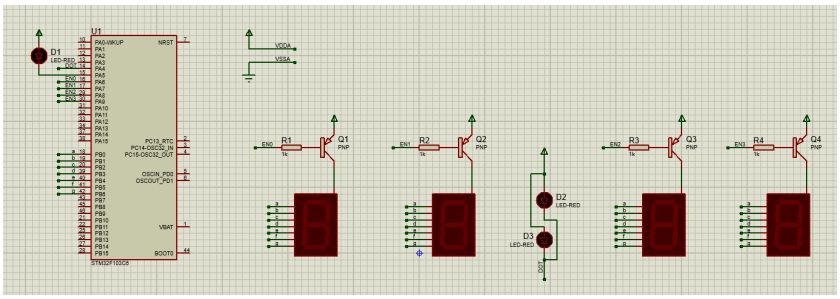

Figure 2: *Schematic for exercise 2*

1.2.2 Report 2

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

```
// Exercise2
int clock_1 = 100;
int clock7Seg = 50;
int stage = 0; //init stage

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef * htim){
  clock_1 --;
  if (clock_1 <= 0) {     //Flag
    HAL_GPIO_TogglePin(dot_GPIO_Port, dot_Pin);
    clock_1 = 100;
  } //Blinking dots
  clock7Seg--;
  if (clock7Seg <= 0){    //Flag
    switch (stage){
      case 0:
      {
        HAL_GPIO_WritePin(en0_GPIO_Port, en0_Pin, 0);
        HAL_GPIO_WritePin(en1_GPIO_Port, en1_Pin, 1);
        HAL_GPIO_WritePin(en2_GPIO_Port, en2_Pin, 1);
        HAL_GPIO_WritePin(en3_GPIO_Port, en3_Pin, 1);
        display7SEG(1);
        stage = 1;
        break;
      }
      case 1:
      {
        HAL_GPIO_WritePin(en0_GPIO_Port, en0_Pin, 1);
        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):

```

const int MAX_LED = 4;
int index_led = 0;
int led_buffer[4] = {1, 7, 0, 5};

void enSelect(int idx){ // This function is for choosing LEDs
  switch (idx){
    case 0:
      {
          HAL_GPIO_WritePin(en0_GPIO_Port, en0_Pin, 0);
          HAL_GPIO_WritePin(en1_GPIO_Port, en1_Pin, 1);
          HAL_GPIO_WritePin(en2_GPIO_Port, en2_Pin, 1);
          HAL_GPIO_WritePin(en3_GPIO_Port, en3_Pin, 1);
          break;
        }
        case 1:
        {
          HAL_GPIO_WritePin(en0_GPIO_Port, en0_Pin, 1);
          HAL_GPIO_WritePin(en1_GPIO_Port, en1_Pin, 0);
          HAL_GPIO_WritePin(en2_GPIO_Port, en2_Pin, 1);
          HAL_GPIO_WritePin(en3_GPIO_Port, en3_Pin, 1);
          break;
        }
        case 2:
        {
          HAL_GPIO_WritePin(en0_GPIO_Port, en0_Pin, 1);
          HAL_GPIO_WritePin(en1_GPIO_Port, en1_Pin, 1);
          HAL_GPIO_WritePin(en2_GPIO_Port, en2_Pin, 0);
          HAL_GPIO_WritePin(en3_GPIO_Port, en3_Pin, 1);
          break;
        }
        case 3:
        {
          HAL_GPIO_WritePin(en0_GPIO_Port, en0_Pin, 1);
          HAL_GPIO_WritePin(en1_GPIO_Port, en1_Pin, 1);
          HAL_GPIO_WritePin(en2_GPIO_Port, en2_Pin, 1);
          HAL_GPIO_WritePin(en3_GPIO_Port, en3_Pin, 0);
          break;
        }
        default:
        {
          HAL_GPIO_WritePin(en0_GPIO_Port, en0_Pin, 1);
          HAL_GPIO_WritePin(en1_GPIO_Port, en1_Pin, 1);
```

```
            HAL_GPIO_WritePin(en2_GPIO_Port, en2_Pin, 1);
            HAL_GPIO_WritePin(en3_GPIO_Port, en3_Pin, 1);
            break;
          }
        }
}

void update7SEG (int index){
  if (index_led >= MAX_LED) index_led = 0; //Check index_led
  index = index % 4;
  switch (index){
    case 0:{
      enSelect(index);
      display7SEG(led_buffer[index]);
      break;
    }
    case 1:{
      enSelect(index);
      display7SEG(led_buffer[index]);
      break;
    }
    case 2:{
      enSelect(index);
      display7SEG(led_buffer[index]);
      break;
    }
    case 3:{
      enSelect(index);
      display7SEG(led_buffer[index]);
      break;
    }
    default:
      enSelect(-1); // Turn off the LED
      break;
  }
}
```

1.3.2 Report 2

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

```
int clock_1 = 50;
int clockBlink = 100;
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef * htim){
  clock_1--;
  clockBlink--;
  if (clock_1 <= 0) {
    update7SEG(index_led++);
```

```
    clock_1 = 50;
  }
  if (clockBlink <= 0){
    HAL_GPIO_TogglePin(dot_GPIO_Port, dot_Pin);
    clockBlink = 100;
  }
}
```

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:

```
int clock_1 = 25;
int clockBlink = 100;
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef * htim){
  clock_1--;
  clockBlink--;
  if (clock_1 <= 0) {
    update7SEG(index_led++);
    clock_1 = 25;
  }
  if (clockBlink <= 0){
    HAL_GPIO_TogglePin(dot_GPIO_Port, dot_Pin);
    clockBlink = 100;
  }
}
```

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

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

1.5 Exercise 5

1.5.1 Report 1

This is the source code of `void updateClockBuffer()`:

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

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:

```
setBlink(10);
while (1)
  {
    if(clockBlink_flag == 1){ //end 1s
      HAL_GPIO_TogglePin(dot_GPIO_Port, dot_Pin); // Toggle DOT after 1s
      sec++;
      if (sec >= 60) {
        sec = 0;
        min++;
      }
      if (min >= 60){
        min = 0;
        hrs++;
      }
      if (hrs >= 24) hrs = 0; // Running the clock at 1s
      setBlink(1000); // Reset the clock
    }
    updateClockBuffer();
  }
```

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:

```
while (1)
  {
    /* USER CODE END WHILE */
    if(clockBlink_flag == 1){ //end 1s
      HAL_GPIO_TogglePin(dot_GPIO_Port, dot_Pin); // Toggle DOT after 1s
      sec++;
      if (sec >= 60) {
        sec = 0;
        min++;
      }
      if (min >= 60){
        min = 0;
        hrs++;
      }
      if (hrs >= 24) hrs = 0; // Running the clock at 1s

      setBlink(1000);
    }

    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:

```
int matrix_idx = 0;
uint8_t matrix_buffer_A[8] = {0x3C, 0x66, 0x66,0x7E,0x66,0x66,0x66,0x00};

/* TIMER STARTS HERE */
int counter = 0, flag = 0;

int const cycle = 10;

void set(int duration){
  counter = duration / cycle;
  flag = 0;
}

void run(void){
  counter--;
  if (counter == 0) flag = 1;
}

/* TIMER ENDS HERE */
int convertedBinary[8] = {0,0,0,0,0,0,0,0};
int arr[16]; //16-bit array
int arr2[16];
int shift_counter = 0;
void convertToBinary(uint8_t num){
    for(int i = 7; i >= 0; i--){
        convertedBinary[i] = num % 2;
        num = num / 2;
    }
}


void updateLEDMatrix(int index){
  for (int i = 0; i < 8; i++){
    if (index == i){
      convertToBinary(matrix_buffer_A[i]);
      for(int j = 0; j < 8; j++){
        HAL_GPIO_WritePin(ENM0_GPIO_Port, ENM0_Pin << j, arr[j]);
      }
      break;
    }
  }
}

void resetState(){
  for (int i = 0; i < 8; i++){
    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:

```
int matrix_idx = 0;

uint8_t matrix_buffer_A[8] = {0x3C, 0x66, 0x66,0x7E,0x66,0x66,0x66,0x00};

/* TIMER STARTS HERE */
int counter = 0, flag = 0;

int const cycle = 10;

void set(int duration){
  counter = duration / cycle;
  flag = 0;
}

void run(void){
  counter--;
  if (counter == 0) flag = 1;
}

/* TIMER ENDS HERE */
int convertedBinary[8] = {0,0,0,0,0,0,0,0};
int arr[16]; //16-bit array
int arr2[16];
int shift_counter = 0;
void convertToBinary(uint8_t num){
    for(int i = 7; i >= 0; i--){
        convertedBinary[i] = num % 2;
        num = num / 2;
    }
}
void convertShift(uint8_t num){
  for(int i = 15; i >= 0; i--){
    arr[i] = num % 2;
    arr2[i] = arr[i];
    num = num / 2;
  }
  for (int i = 0; i < 16; i++){
    int newidx = i - shift_counter;
    if (newidx >= 0) arr[newidx] = arr2[i];
    else arr[15 - shift_counter + 1] = arr2[i];
  }
}


void updateLEDMatrix(int index){
  for (int i = 0; i < 8; i++){
```

```
    if (index == i){
//      convertToBinary(matrix_buffer_A[i]);
      convertShift(matrix_buffer_A[i]);
      for(int j = 0; j < 8; j++){
        HAL_GPIO_WritePin(ENM0_GPIO_Port, ENM0_Pin << j, arr[j + 4]);
      }
      break;
    }
  }
}

void resetState(){
  for (int i = 0; i < 8; i++){
    HAL_GPIO_WritePin(ROW0_GPIO_Port, ROW0_Pin << i, 0);
  }
}
void displayLEDMatrix(){
  if (matrix_idx >= 8) {
    matrix_idx = matrix_idx % 8;
    shift_counter++;
    if (shift_counter >= 16) shift_counter = shift_counter%16;
  }
  resetState();
  HAL_GPIO_WritePin(ROW0_GPIO_Port, ROW0_Pin << matrix_idx, SET);
  updateLEDMatrix(matrix_idx);
}
```

References