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 Thanh Tai 2252722

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/ThanhTaiNguyen24/mcu-mpu-lab1.

The schematic for the exercises from 1 to 10 is located here:

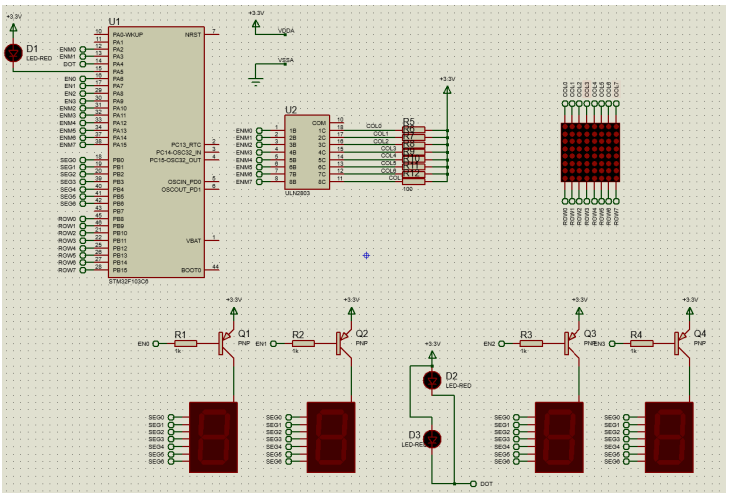

Figure 1: *The schematic for the exercises from 1 to 10.*

1.1 Exercise 1

1.1.1 Report 1

Can be found at 1.

1.1.2 Report 2

This is the header file library for the exercise 1 :

```
int counter = 100;
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim){

  if(counter == 100){
  led7segmentdisplay(1);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, GPIO_PIN_RESET);
  } else if (counter == 50){
  led7segmentdisplay(2);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, GPIO_PIN_RESET);
  }
  counter--;
  if (counter == 0) counter = 100;
}
```

Short question: the frequency of the scanning process is 1Hz.

1.2 Exercise 2

1.2.1 Report 1

Can be found at 1.

1.2.2 Report 2

This is the source code for the exercise 2:

```
int counter = 200;
int led_counter = 100;
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim){
  if(counter == 200){
  led7segmentdisplay(1);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, GPIO_PIN_SET);
  } else if (counter == 150){
  led7segmentdisplay(2);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, GPIO_PIN_SET);

  } else if(counter == 100){
  led7segmentdisplay(3);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, GPIO_PIN_SET);

  } else if (counter == 50){
  led7segmentdisplay(0);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, GPIO_PIN_RESET);
  }
  counter--;
  led_counter --;
  if (led_counter == 50){
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, GPIO_PIN_SET);
  }
  if (led_counter == 0){
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, GPIO_PIN_RESET);
    led_counter = 100;
  }
  if (counter == 0){
```

```
41      counter = 200;
42    }
43  }
```

1.3 Exercise 3

1.3.1 Report 1

This is the source code for update7SEG function:

```
void update7SEG (int index){
  switch(index){
  case 0:
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, GPIO_PIN_SET);
    led7segmentdisplay(led_buffer[0]);
    break;
  case 1:
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, GPIO_PIN_SET);
    led7segmentdisplay(led_buffer[1]);
    break;
  case 2:
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, GPIO_PIN_SET);
    led7segmentdisplay(led_buffer[2]);
    break;
  case 3:
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, GPIO_PIN_SET);
    led7segmentdisplay(led_buffer[3]);
    break;
  default:
    break;
  }
}
```

1.3.2 Report 2

This is the source code for PeriodElapsedCallback function:

```
int counter = 50;
int led_counter = 100;
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim){
  update7SEG(index_led);
  counter--;
```

```
  if(counter == 0){
    index_led++;
    counter = 50;
    if (index_led > 3) index_led = 0;
  }


  led_counter --;
  if (led_counter == 50){
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, GPIO_PIN_SET);
  }
  if (led_counter == 0){
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, GPIO_PIN_RESET);
    led_counter = 100;
  }
}
```

1.4 Exercise 4

1.4.1 Report 1

This is the source code for the exercise 4:

```
int counter = 25;
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim){
  update7SEG(index_led);
  counter--;
  if(counter == 0){
    index_led++;
    counter = 25;
    if (index_led > 3) index_led = 0;
  }
  led_counter --;
  if (led_counter == 50){
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, GPIO_PIN_SET);
  }
  if (led_counter == 0){
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, GPIO_PIN_RESET);
    led_counter = 100;
  }
}
```

1.5 Exercise 5

1.5.1 Report 1

This is the source code for the exercise 5:

```
void updateClockBuffer (){
  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

Answer: if **setTimer0(1000)** is removed, **timer0_counter** always equal to 0, the LED will not toggle.

1.6.2 Report 2

Answer: **void setTimer0** receive input parameter named **duration** as an **integer**. If **duration** equal to 1, **timer0_counter** (which is an **integer** value equal to 0, the LED will not toggle.

1.6.3 Report 3

Answer: If **duration** equal to 10, **timer0_counter** (which is an **integer** value equal to 1, the **timer_run()** function will invokes in this case, the LED will toggle.

1.7 Exercise 7

This is the code for the exercise 7:

```
setTimer1(1000);
setTimer2(500);
while (1)
 {
    if(timer1_flag == 1) {
    second++;
    if (second >= 60){
      second = 0;
      minute++;
    }
    if (minute >= 60) {
      minute = 0;
      hour++;
    }
    if (hour >= 24) {
      hour = 0;   24
    }
    updateClockBuffer();
    setTimer1(1000);
    }
    if(timer2_flag == 1){
    HAL_GPIO_TogglePin (GPIOA ,GPIO_PIN_4);
    setTimer2(500);
    }
}
```

1.8 Exercise 8

1.8.1 Report 1

This is the code for the exercise 8:

```
setTimer1(10);
setTimer2(10);
setTimer3(20);
while (1)
 {
    if(timer1_flag == 1) {
        second++;
    if (second >= 60){
      second = 0;
      minute++;
    }
    if (minute >= 60) {
      minute = 0;
      hour++;
    }
        if (hour >= 24) {
      hour = 0;
        }
        updateClockBuffer();
        update7SEG(index_led);
        HAL_GPIO_TogglePin (GPIOA ,GPIO_PIN_4);
        index_led++;
        if (index_led > 3) index_led = 0;
        setTimer1(1000);
    }
}

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim){
  timerRun();
}

int timer1_counter = 0;
int timer1_flag = 0;
int TIMER_CYCLE = 10;

void setTimer1(int duration){
  timer1_counter = duration/TIMER_CYCLE;
  timer1_flag = 0;
}
void setTimer2(int duration){
  timer2_counter = duration/TIMER_CYCLE;
  timer2_flag = 0;
}
void setTimer3(int duration){
  timer3_counter = duration/TIMER_CYCLE;
```

```
46    timer3_flag = 0;
47  }
48
49  void timerRun(){
50      if (timer1_counter > 0){
51    timer1_counter--;
52    if(timer1_counter <= 0){
53      timer1_flag = 1;
54    }
55      }
56      if (timer2_counter > 0){
57    timer2_counter--;
58    if(timer2_counter <= 0){
59      timer2_flag = 1;
60    }
61      }
62      if (timer3_counter > 0){
63    timer3_counter--;
64    if(timer3_counter <= 0){
65      timer3_flag = 1;
66    }
67      }
68  }
```

1.9 Exercise 9

1.9.1 Report 1

Can be found at 1.

1.9.2 Report 2

This is the source code for the exercise 9:

```
uint8_t matrix_buffer [8] = {0x18,0x3c,0x66,0x66,0x7e,0x7e,0x66,0x66};
void updateLedMatrix(int index){
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_9, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_10, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_11, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_12, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_13, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_14, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_15, GPIO_PIN_SET);

  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_11, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_12, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_13, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_14, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_15, GPIO_PIN_SET);

  uint8_t numberCode = matrix_buffer[index];
  uint8_t value[8];
  switch(index){
  case 0: HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, GPIO_PIN_RESET); break;
  case 1: HAL_GPIO_WritePin(GPIOB, GPIO_PIN_9, GPIO_PIN_RESET); break;
  case 2: HAL_GPIO_WritePin(GPIOB, GPIO_PIN_10, GPIO_PIN_RESET); break;
  case 3: HAL_GPIO_WritePin(GPIOB, GPIO_PIN_11, GPIO_PIN_RESET); break;
  case 4: HAL_GPIO_WritePin(GPIOB, GPIO_PIN_12, GPIO_PIN_RESET); break;
  case 5: HAL_GPIO_WritePin(GPIOB, GPIO_PIN_13, GPIO_PIN_RESET); break;
  case 6: HAL_GPIO_WritePin(GPIOB, GPIO_PIN_14, GPIO_PIN_RESET); break;
  case 7: HAL_GPIO_WritePin(GPIOB, GPIO_PIN_15, GPIO_PIN_RESET); break;
  default: break;
  }
  for (index_led_matrix = 0; index_led_matrix < MAX_LED_MATRIX; index_led_matrix
    ++){

    value[index_led_matrix] = (numberCode >> (7 - index_led_matrix)) & 0x01;
    value[index_led_matrix] = (value[index_led_matrix] == 1) ? 0 : 1;
  }
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, value[0]);
```

```
40    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3, value[1]);
41    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, value[2]);
42    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_11, value[3]);
43    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_12, value[4]);
44    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_13, value[5]);
45    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_14, value[6]);
46    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_15, value[7]);
47  }
```

1.10 Exercise 10

This is the code for the exercise 10:

```
setTimer1(10);
setTimer2(10);
setTimer3(20);
while (1){
    if(timer2_flag == 1){
        updateLedMatrix(index_led_matrix);
        index_led_matrix++;
        if (index_led_matrix > MAX_LED_MATRIX) index_led_matrix = 0;
        setTimer2(40);
    }
    if(timer3_flag == 1){
    HAL_GPIO_TogglePin (GPIOA, GPIO_PIN_5);
    ledShiftRight();
    setTimer3(500);
    }
}
void ledShiftRight(){
  for (int i = 0; i < MAX_LED_MATRIX;i++){
    uint8_t leftToRight = (matrix_buffer[i] & 0x01) << 7;
    matrix_buffer[i] = (matrix_buffer[i] >> 1) | leftToRight;
  }
}
```

Explaination: The letter 'A' is shifted to the right. If borderless, the letter 'A' will appear in the first left column of the LED MATRIX again.

References