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

****

Lab 1 - LED Animation

Microcontroller - Microprocessor (Lab)

COURSE ID: CO3009 - HK251

Name: Trần Duy Tuấn - Student ID: 2353278

Supervising Lecturer: Dr. Phan Văn Sỹ

TP. Hồ Chí Minh – 2024

1 Exercise

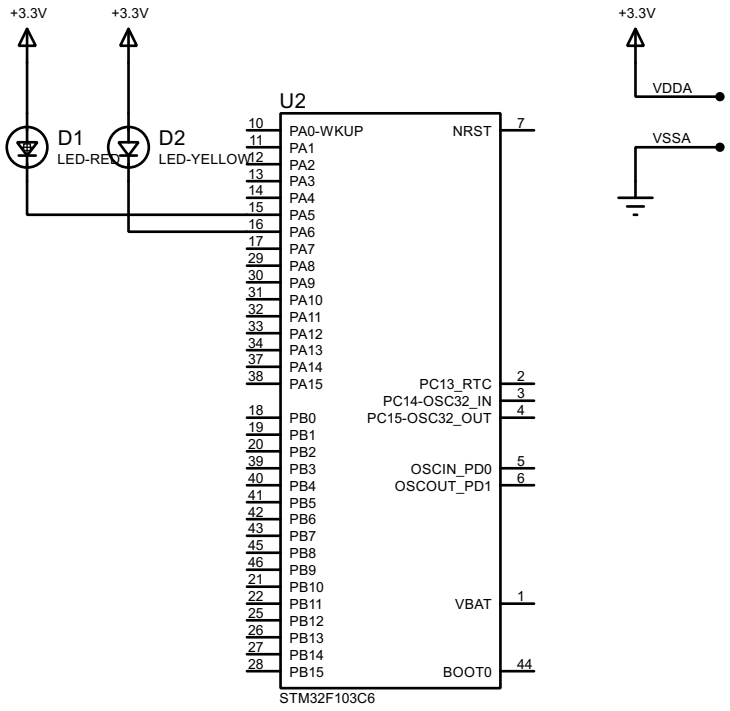
The GitHub link for the lab schematics is at here or in this link: https://github.com/MEgjune621/Lab_MCU-CO3009_Sem-251/

The default while(1) code for most of the exercise is:

```
while (1) {
    // THE FUNCTION INPUT INSERTED HERE
    HAL_Delay(1000);
}
```

1.1 Exercise 1

1.1.1 Report 1:

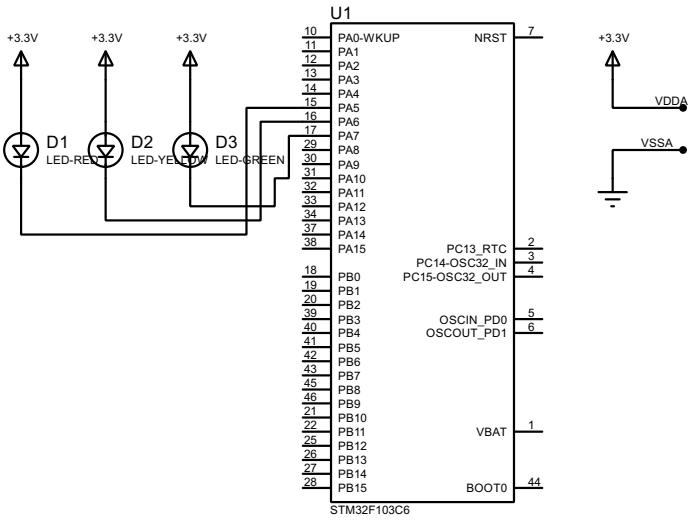


1.1.2 Report 2:

```c
int main(void)
{
    // Initialize LEDs: RED = ON, YELLOW = OFF
    setPin(LED_RED, ON);
    setPin(LED_YELLOW, OFF);

    // Infinite loop
    while (1)
    {
        // Toggle RED LED state
        togglePin(LED_RED);

        // Toggle YELLOW LED state
        togglePin(LED_YELLOW);

        // Wait 2 seconds
        delay(2000);
    }
}
```

1.2 Exercise 2

1.2.1 Report 1:



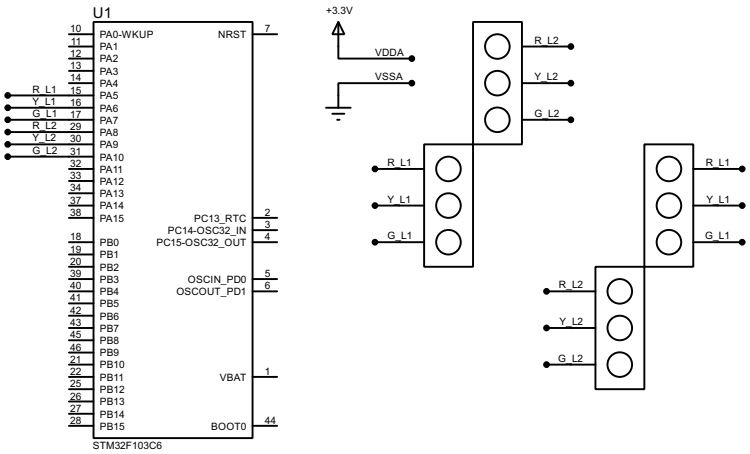
1.2.2 Report 2:

```c
// Define traffic light states
enum TrafficState { RED, YELLOW, GREEN };

// Structure to hold traffic light state and countdown timer
struct TrafficLight {
    TrafficState state;
    int timer;   // countdown in seconds
};

// Initialize traffic light (start RED, 5s)
TrafficLight light = { RED, 5 };

// Function: update state when timer expires
void updateTrafficLight(TrafficLight *light) {
    if (light->timer > 0) {
        light->timer--;   // keep counting down
        return;
    }

    switch (light->state) {
        case RED:
            light->state = GREEN;
            light->timer = 3;   // stay GREEN for 3s
            break;
        case GREEN:
            light->state = YELLOW;
            light->timer = 2;   // stay YELLOW for 2s
```

```
            break;
        case YELLOW:
            light->state = RED;
            light->timer = 5;    // stay RED for 5s
            break;
    }
}

// Function: apply the current light state to LEDs
void applyTrafficLight(TrafficLight *light) {
    if (light->state == RED) {
        turnOn(RED_LED);
        turnOff(YELLOW_LED);
        turnOff(GREEN_LED);
    }
    else if (light->state == YELLOW) {
        turnOff(RED_LED);
        turnOn(YELLOW_LED);
        turnOff(GREEN_LED);
    }
    else if (light->state == GREEN) {
        turnOff(RED_LED);
        turnOff(YELLOW_LED);
        turnOn(GREEN_LED);
    }
}

int main(void) {
    // Loop forever
    while (1) {
        updateTrafficLight(&light);    // update state machine
        applyTrafficLight(&light);     // update LEDs
        delay(1000);                   // wait 1s before next tick
    }
}
```

1.3 Exercise 3

1.3.1 Report 1:



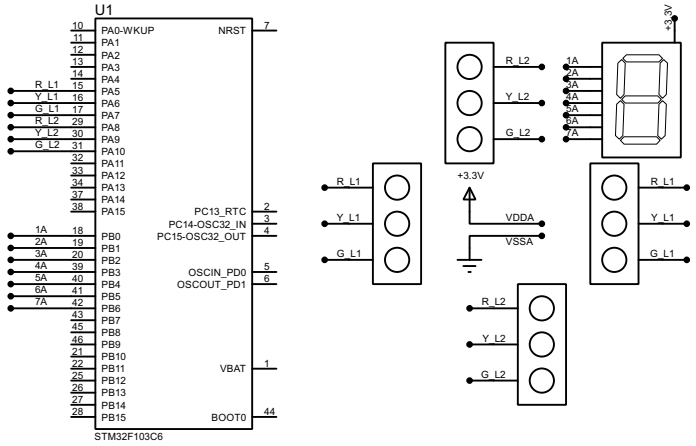
1.3.2 Report 2:

```c
// Use the pseudocode from Exercise 2, but extend it to handle TWO traffic lights
// Each traffic light has its own pins and timers.

// Traffic light structure with its own pins and timer
struct TrafficLight {
    TrafficState state;
    int timer;
    int redPin;
    int yellowPin;
    int greenPin;
};

// Initialize two traffic lights with starting states and durations
TrafficLight light1 = { RED,   5, PIN5,  PIN6,  PIN7  };   // starts RED for 5s
TrafficLight light2 = { GREEN, 3, PIN8,  PIN9,  PIN10 };   // starts GREEN for 3s

int main(void) {
    // Infinite loop
    while (1) {
        // Update timers and states
        updateTrafficLight(&light1);
        updateTrafficLight(&light2);

        // Apply states to LEDs
        applyTrafficLight(&light1);
        applyTrafficLight(&light2);

        // Wait 1 second
```

```
        delay(1000);
    }
}
```

1.4 Exercise 4

1.4.1 Report 1:



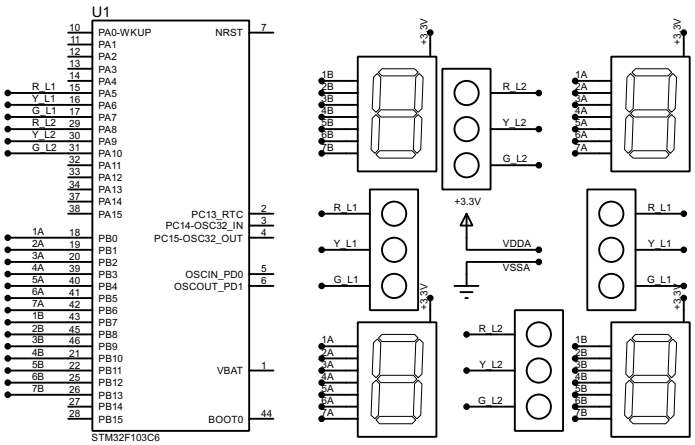
1.4.2 Report 2:

```
// Use the pseudocode from Exercise 3, and add support for a single 7-
    segment display
// To show the countdown of one traffic light.

// 7-segment patterns for digits 0-9
byte seg_digits[10] = {
    0b11000000, // 0
    0b11111001, // 1
    0b10100100, // 2
    0b10110000, // 3
    0b10011001, // 4
    0b10010010, // 5
    0b10000010, // 6
    0b11111000, // 7
    0b10000000, // 8
    0b10010000  // 9
};

// Display number on 7-seg
void display7SEG(int firstPin, int num) {
    if (num > 9) return;
    byte pattern = seg_digits[num];

    for (int i = 0; i < 7; i++) {
        if (pattern & (1 << i))
```

```
            turnOn(firstPin + i);
        else
            turnOff(firstPin + i);
    }
}

// Main program
int main(void) {
    while (1) {
        // Update state machines
        updateTrafficLight(&light1);
        updateTrafficLight(&light2);

        // Show light1 countdown on 7-seg
        display7SEG(PIN0, light1.timer);

        // Apply states to LEDs
        applyTrafficLight(&light1);
        applyTrafficLight(&light2);

        // Wait 1 second
        delay(1000);
    }
}
```

1.5 Exercise 5

1.5.1 Report 1:



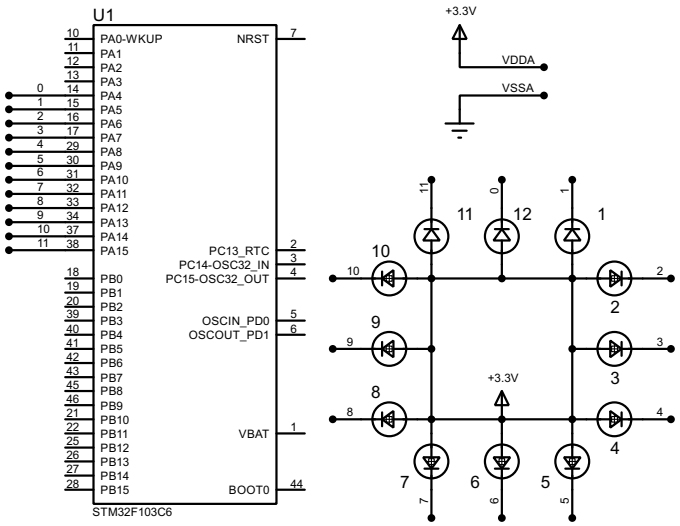
1.5.2 Report 2:

```
// Use the pseudocode from Exercise 4, and add support for displaying
   countdowns of BOTH traffic lights on two 7-segment displays.

// Main program
```

```
int main(void) {
    while (1) {
        // Update both lights
        updateTrafficLight(&light1);
        updateTrafficLight(&light2);

        // Display both countdowns
        display7SEG(PIN0, light1.timer);
        display7SEG(PIN7, light2.timer);

        // Apply states to LEDs
        applyTrafficLight(&light1);
        applyTrafficLight(&light2);

        // Wait 1 second
        delay(1000);
    }
}
```

1.6 Exercise 6

1.6.1 Report 1:



1.6.2 Report 2:

```
// Map clock positions to PA4..PA15
const int CLOCK_PIN[12] = {
    GPIO_PIN_4,  GPIO_PIN_5,  GPIO_PIN_6,  GPIO_PIN_7,
    GPIO_PIN_8,  GPIO_PIN_9,  GPIO_PIN_10, GPIO_PIN_11,
    GPIO_PIN_12, GPIO_PIN_13, GPIO_PIN_14, GPIO_PIN_15
};

```

```
// Test sequence: light each LED in order briefly
void testAllLEDsSequence(void) {
    while (1) {
        for (int i = 0; i < 12; i++) {
            // turn this LED ON (active low assumed)
            HAL_GPIO_WritePin(GPIOA, CLOCK_PIN[i], GPIO_PIN_RESET);
            HAL_Delay(300); // visible delay
            // turn it OFF again
            HAL_GPIO_WritePin(GPIOA, CLOCK_PIN[i], GPIO_PIN_SET);
            HAL_Delay(50);
        }
    }
}
```

1.7 Exercise 7

1.7.1 Report 1:

Can be found at Exercise 6/Report 1

1.7.2 Report 2:

```
void clearAllClock(void) {
    // Assumes active-low LEDs: GPIO_PIN_SET = OFF
    for (int i = 0; i < 12; i++) {
        HAL_GPIO_WritePin(GPIOA, CLOCK_PIN[i], GPIO_PIN_SET);
    }
}
```

1.8 Exercise 8

1.8.1 Report 1:

Can be found at Exercise 6/Report 1

1.8.2 Report 2:

```
void setNumberOnClock(int num) {
    if (num < 0 || num > 11) return;
    // Active-low: GPIO_PIN_RESET turns LED ON
    HAL_GPIO_WritePin(GPIOA, CLOCK_PIN[num], GPIO_PIN_RESET);
}
```

1.9 Exercise 9

1.9.1 Report 1:

Can be found at Exercise 6/Report 1

1.9.2 Report 2:

```c
void clearNumberOnClock(int num) {
    if (num < 0 || num > 11) return;
    // Active-low: GPIO_PIN_SET turns LED OFF
    HAL_GPIO_WritePin(GPIOA, CLOCK_PIN[num], GPIO_PIN_SET);
}
```

1.10 Exercise 10

1.10.1 Report 1:

Can be found at Exercise 6/Report 1

1.10.2 Report 2:

```c
void displayClock(int hour, int minute, int second) {
    clearAllClock();

    int posHour   = hour % 12;             // 0..11
    int posMinute = (minute / 5) % 12;     // map 0..59 -> 0..11
    int posSecond = (second / 5) % 12;     // map 0..59 -> 0..11

    setNumberOnClock(posHour);
    setNumberOnClock(posMinute);
    setNumberOnClock(posSecond);
}

int main(void) {
    // init HAL, clocks, MX_GPIO_Init() ...
    while (1) {
        for (int hour = 0; hour < 12; hour++) {
            for (int minute = 0; minute < 60; minute++) {
                for (int second = 0; second < 60; second++) {
                    displayClock(hour, minute, second);
                    HAL_Delay(1000);
                }
            }
        }
    }
}
```