

ĐẠI HỌC QUỐC GIA, THÀNH PHỐ HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC BÁCH KHOA  
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH



**Vi xử lý - Vi điều khiển (TN) (CO3010)**

---

**LAP 3**

# **Microprocessor- Microcontroller**

---

GVHD: TÔN HUỖNH LONG  
SV thực hiện: BÙI NGUYỄN NHẬT TIẾN 2213444

TP Hồ Chí Minh, Tháng 10 Năm 2025

## Mục lục

<b>1</b>	<b>Bài tập tổng quan</b>	<b>2</b>
1.1	Bài 1 . . . . .	2
1.2	Bài 2 . . . . .	2
1.3	Bài 3 . . . . .	3
1.4	Bài 4 . . . . .	3
1.4.1	Công thức tính toán . . . . .	3
1.5	Bài 5 – Chống dội phím (Button Debounce) . . . . .	3
1.5.1	Mã nguồn . . . . .	3
1.6	Bài 6 – Hiển thị LED 7 đoạn và đèn giao thông . . . . .	5
1.6.1	LED 7 đoạn . . . . .	5
1.6.2	Đèn giao thông . . . . .	6
1.7	Bài 7 – 9: Máy trạng thái hữu hạn (FSM) . . . . .	6
1.8	Bài 10 . . . . .	7

# 1 Bài tập tổng quan

Liên kết GitHub chứa toàn bộ sơ đồ và mã nguồn của bài thí nghiệm: [https://github.com/NhatTiens/vx1\\_lap3](https://github.com/NhatTiens/vx1_lap3)

## 1.1 Bài 1

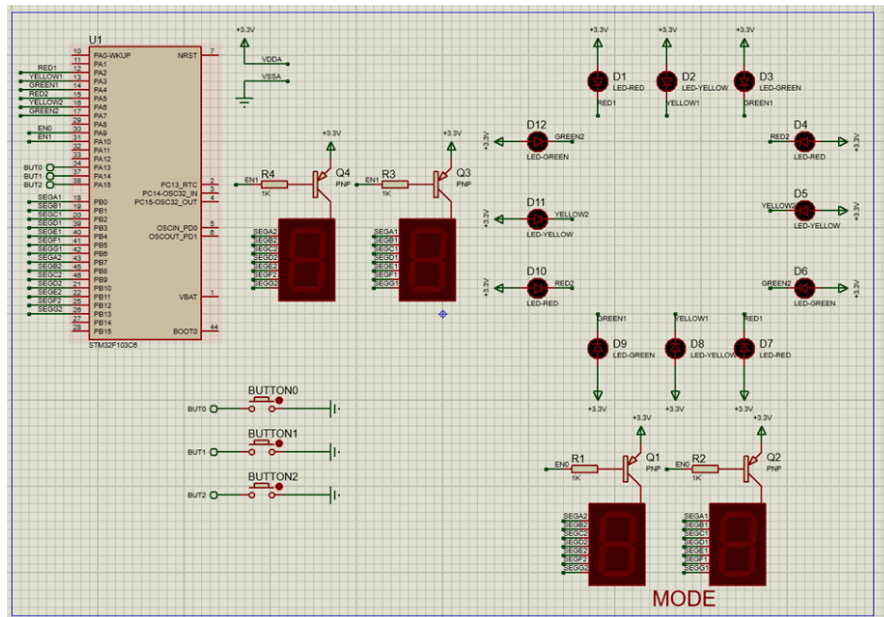


Figure 1: Sơ đồ Proteus – Bài 1

## 1.2 Bài 2



$$T_0 = \frac{(\text{prescaler} + 1) \times (\text{counterPeriod} + 1)}{f}$$

Figure 3: Các công thức tính toán tần số Timer

Listing 1: Đọc và xử lý nút nhấn

```
#include "input_reading.h"

int Max_press = 500;
int Max_hold = 500;
int button_flag[NO_OF_BUTTONS];

static GPIO_PinState debounceButtonBuffer1[NO_OF_BUTTONS];
static GPIO_PinState debounceButtonBuffer2[NO_OF_BUTTONS];
static GPIO_PinState debounceButtonBuffer3[NO_OF_BUTTONS];
static GPIO_PinState debounceButtonBuffer4[NO_OF_BUTTONS];

static uint8_t press1s_flag[NO_OF_BUTTONS];
static uint16_t press1s_counter[NO_OF_BUTTONS];

#define INPUT_PORT GPIOA
uint16_t BUTTON_PIN[NO_OF_BUTTONS] = {0x2000, 0x4000, 0x6000};

void setButton() {
    for (unsigned char i = 0; i < NO_OF_BUTTONS; i++) {
        debounceButtonBuffer1[i] = BUTTON_IS_RELEASED;
        debounceButtonBuffer2[i] = BUTTON_IS_RELEASED;
        debounceButtonBuffer3[i] = BUTTON_IS_RELEASED;
        debounceButtonBuffer4[i] = BUTTON_IS_RELEASED;
        button_flag[i] = BUTTON_FLAG_RESET;
        press1s_flag[i] = BUTTON_FLAG_RESET;
        press1s_counter[i] = 0;
    }
}
```

Listing 2: Hàm đọc trạng thái nút nhấn

```
void readButton(void) {
    for (unsigned char i = 0; i < NO_OF_BUTTONS; i++) {
        debounceButtonBuffer3[i] = debounceButtonBuffer2[i];
        debounceButtonBuffer2[i] = debounceButtonBuffer1[i];
        debounceButtonBuffer1[i] = HAL_GPIO_ReadPin(INPUT_PORT, BUTTON_PIN[i]);
        if ((debounceButtonBuffer2[i] == debounceButtonBuffer1[i]) &&
            (debounceButtonBuffer2[i] == debounceButtonBuffer3[i])) {
            if (debounceButtonBuffer3[i] != debounceButtonBuffer4[i]) {
                debounceButtonBuffer4[i] = debounceButtonBuffer3[i];
                if (debounceButtonBuffer4[i] == BUTTON_IS_PRESSED) {
```

```

        Max_press = 500;
        button_flag[i] = BUTTON_FLAG_SET;
    } else {
        Max_press--;
        press1s_counter[i] = 0;
        debounceButtonBuffer4[i] = BUTTON_IS_RELEASED;
    }
} else {
    if (press1s_counter[i] < DURATION_FOR_AUTO_INCREASING)
        press1s_counter[i]++;
    else press1s_flag[i] = BUTTON_FLAG_SET;
}
}
}
}
}

```

Listing 3: Hàm kiểm tra sự kiện nhấn

```

unsigned char Press_button(unsigned char index) {
    if (index >= NO_OF_BUTTONS) return 0;
    if (button_flag[index] == BUTTON_FLAG_SET) {
        button_flag[index] = BUTTON_FLAG_RESET;
        return 1;
    }
    return 0;
}

unsigned char Press1s_button(unsigned char index) {
    if (index >= NO_OF_BUTTONS) return 0;
    if (press1s_flag[index] == BUTTON_FLAG_SET) {
        press1s_flag[index] = BUTTON_FLAG_RESET;
        press1s_counter[index] = 0;
        return 1;
    }
    return 0;
}

```

## 1.6 Bài 6 – Hiển thị LED 7 đoạn và đèn giao thông

### 1.6.1 LED 7 đoạn

Listing 4: Điều khiển LED 7 đoạn

```

#include "segment_display.h"
#include "global.h"

void display7SEG1(int num) {
    switch (num) {
        case 0:
            HAL_GPIO_WritePin(SEGA1_GPIO_Port, SEGA1_Pin, RESET);

```

```

        HAL_GPIO_WritePin(SEGB1_GPIO_Port, SEGB1_Pin, RESET);
        HAL_GPIO_WritePin(SEGC1_GPIO_Port, SEGC1_Pin, RESET);
        HAL_GPIO_WritePin(SEGD1_GPIO_Port, SEGD1_Pin, RESET);
        HAL_GPIO_WritePin(SEGE1_GPIO_Port, SEGE1_Pin, RESET);
        HAL_GPIO_WritePin(SEGF1_GPIO_Port, SEGF1_Pin, RESET);
        HAL_GPIO_WritePin(SEGG1_GPIO_Port, SEGG1_Pin, SET);
        break;
    ...
}
}

```

### 1.6.2 Đèn giao thông

Listing 5: Hàm điều khiển đèn giao thông

```

#include "led_display.h"

void setRed1() {
    HAL_GPIO_WritePin(RED1_GPIO_Port, RED1_Pin, RESET);
    HAL_GPIO_WritePin(GREEN1_GPIO_Port, GREEN1_Pin, SET);
    HAL_GPIO_WritePin(YELLOW1_GPIO_Port, YELLOW1_Pin, SET);
}
...
void clearLed() {
    HAL_GPIO_WritePin(RED1_GPIO_Port, RED1_Pin, SET);
    HAL_GPIO_WritePin(GREEN1_GPIO_Port, GREEN1_Pin, SET);
    HAL_GPIO_WritePin(YELLOW1_GPIO_Port, YELLOW1_Pin, SET);
    HAL_GPIO_WritePin(RED2_GPIO_Port, RED2_Pin, SET);
    HAL_GPIO_WritePin(GREEN2_GPIO_Port, GREEN2_Pin, SET);
    HAL_GPIO_WritePin(YELLOW2_GPIO_Port, YELLOW2_Pin, SET);
}

```

## 1.7 Bài 7 – 9: Máy trạng thái hữu hạn (FSM)

Listing 6: FSM chỉnh sửa thời gian các pha đèn

```

void fsm_modify() {
    switch(status3) {
        case RED1_MODIFY:
            mode = 2;
            if (endTimer1()) {
                setTimer1(100);
                toggleLEDs(mode);
            }
            if (Press_button(SET_VALUE)) {
                clearLed();
                red1_duration = time_input * 100;
                time_input = 1;
            }
        }
    }
}

```

```

        status3 = YELLOW1_MODIFY;
    }
    break;
...
case GREEN2_MODIFY:
    mode = 7;
    if (endTimer1()) {
        setTimer1(100);
        toggleLEDs(mode);
    }
    if (Press_button(SET_VALUE)) {
        clearLed();
        green2_duration = time_input * 100;
        time_input = 1;
        return_flag1 = return_flag2 = 1;
        status3 = DELAY;
        led1 = &countdown1;
        led2 = &countdown2;
    }
    break;
}
}
}

```

## 1.8 Bài 10

Toàn bộ mã nguồn nằm trong thư mục GitHub kèm theo.