# Rep_Lab_04_B0829002

## Problem Description:

在這個 Lab 裡，我們要在 basic 實作兩個 debounce button( 1 for left shift 1 bit, the other for right shift 1 bit)以及將 LED output 初始化某個 bit 為 1 排上的其他 bit 為 0。

而 bonus 則是用 timer interrupt 實作一個 button eventOnclick()的狀態轉換將之前做過的四個狀態（基數燈閃爍、偶數燈閃爍、燈號往右移和燈號往左移）實作在這個按鈕內。

## Code and explanations:

### Basic:

#### Part 1: Port, page , in- and output 設定

在這部分基本和上個 lab 一樣，將基本的輸入、輸出、看門狗計數器和頁面設定好。

```c
#include "C8051F040.h"
int N = 100;
void Port_Configuration () {
    XBR2 = 0xc0;
    P1MDIN = 0xff;
    P2MDOUT = 0xff;
}//end of function Port_Configuration

void Default_Config () {
    //turn-off watch-dog timer
    // disable watchdog timer
    WDTCN = 0xde;
    WDTCN = 0xad;

    //initialize SFR setup page
    SFRPAGE = CONFIG_PAGE;              // Switch to configuration page

    Port_Configuration ();

    //set to normal mode
    SFRPAGE = LEGACY_PAGE;
}//end of function Default_Config
```

#### Part 2: main function

這部分為主函數，用來作為 program 入口、function call: port setting, button detection, button event handler。若 button 被偵測到有輸入且是第一個 button 則整排 LED 往右移一格; 第二個則是整排往左移。

```c
int main () {
    int status;
    int button ;
    Default_Config ();
    P1 = 0;
    status = 1;
    P2 = status;
    while (1) {
        button = button_detect();
        if (button == 1){
            if (status == 1){
                status = 128;
            }
            else{
                status = status >> 1;
            }
        }
        if (button == 2){
            if (status == 128){
                status = 1;
            }
            else{
                status = status << 1;
            }
        }
        P2 = status ;
    }//end while (1)
```

## Part 3: button_detect

這部分主要是用來偵測 p1 數入 port 上的按鈕並做 debounce 的動作，當按鈕被按下以後 count 會 count 到 100 然後才會再次偵測按鈕是否被按下，然後回傳 button state 給 main function。

```c
int button_detect () {
    int key_hold, key_release, count, button_state;
    do {
        key_hold = P1;
        button_state = P1 ;
    } while (!key_hold);

    //Stage 2: wait for key released
    key_release = 0;
    count = N;
    while (!key_release) {
        key_hold = P1;
        key_hold ? count = N : (--count == 0 ? key_release = 1 : 1);
    }//Stage 2: wait for key released
    return button_state;
}//end of function button_detect ()
```

## Bonus:

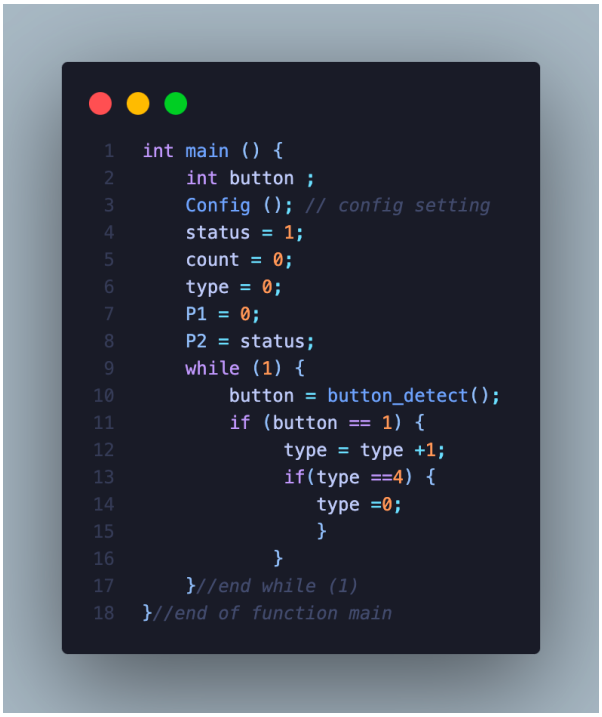
### Part 1: config setting

這部分基本上和 basic 的一樣只是多加了 timer 的設定，因為等一下要用到 timer 做 interrupt 處理。

```c
#include "C8051F040.h"

int N = 100;
int status;
int count;
int tmp, tmp_status;
int type;
void Timer0_ISR ();

void Port_Configuration () {
    //initialize SFR setup page
    SFRPAGE = CONFIG_PAGE;                // Switch to configuration page

    //setup the cross-bar and configure the I/O ports
    XBR2 = 0xc0;
    P2MDOUT = 0xff;

    //set to normal mode
    SFRPAGE = LEGACY_PAGE;
}//end of function Port_Configuration

void Timer_Configuration () {
    // SETTING THE TIMER WORKING MODE
    TMOD = 0x01;
    TCON = 0x10;
    CKCON = 0x10;

    IE = 0x82;
    TL0 = 0;
    TH0 = 0;
}//end of function Timer_Configuration

void Config () {
    //turn-off watch-dog timer
    WDTCN = 0xde;
    WDTCN = 0xad;

    OSCICN = 0x83;
    CLKSEL = 0x00;

    Port_Configuration ();

    Timer_Configuration ();
}//end of function Default_Config
```

### Part 2: main function

這部分基本上也和 basic 一樣，只是多了一些變數初始化以及 2 個 button 狀態的 state 偵測轉換改成一個按鈕 4 個狀態輪轉。



```
int main () {
    int button ;
    Config (); // config setting
    status = 1;
    count = 0;
    type = 0;
    P1 = 0;
    P2 = status;
    while (1) {
        button = button_detect();
        if (button == 1) {
             type = type +1;
             if(type ==4) {
                type =0;
                }
            }
    }//end while (1)
}//end of function main
```

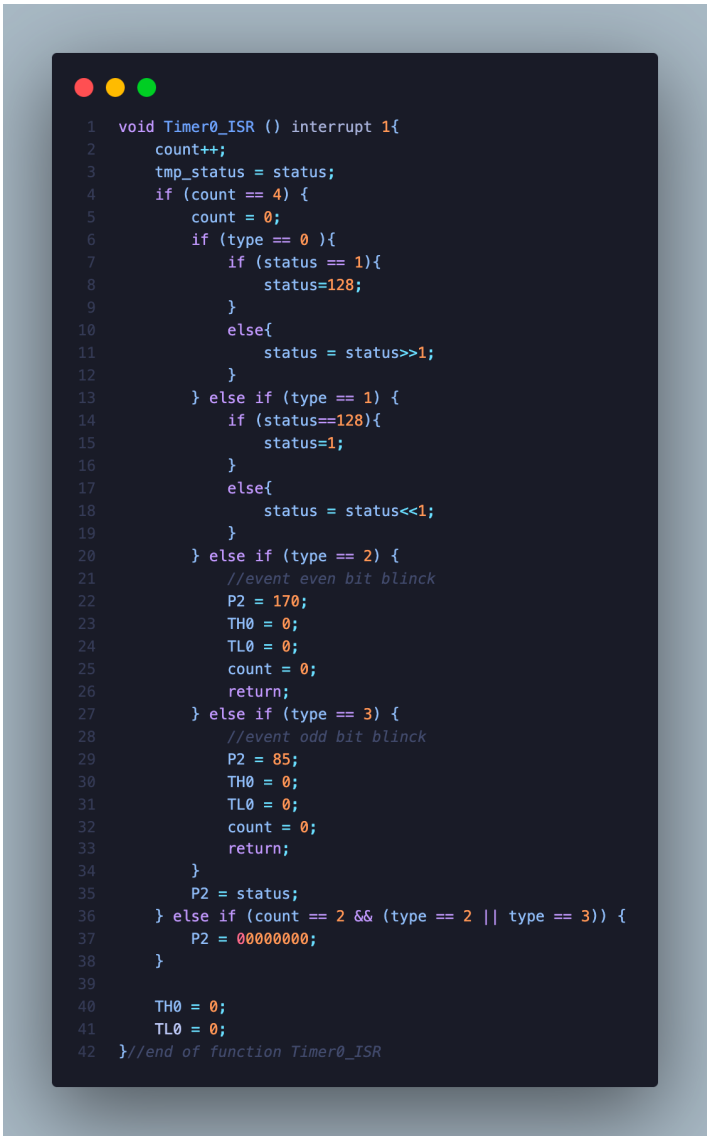
## Part 3: button detection

這與 basic 的 button detection 功能幾乎一模一樣，故參照 basic part 3。

```
int button_detect () {
    int key_hold, key_release;
    int count;
    int button_status;
    do {
        key_hold = P1;
        button_status = P1 ;
    } while (!key_hold);

    //Stage 2: wait for key released
    key_release = 0;
    count = N;
    while (!key_release) {
        key_hold = P1;
        if (key_hold) {
            count = N;
        }
        else {
            --count == 0 ? key_release = 1 : 1;
        }
    }//Stage 2: wait for key released
    return button_status;
}
```

## Part 4: Timer interrupt 處理

這部分主要是用來看現在 button 的狀態（type）是（基數燈閃爍、偶數燈閃爍、燈號往右移和燈號往左移）的哪一個並且將對應的功能應該的訊號寫到 P2（輸出）裡面。而這裡多寫了一個當狀態是奇數和偶數燈閃的時候會在 count 到一半的時候輸出一個全空的訊號到輸出裡面以此達到閃爍的功能。



```
void Timer0_ISR () interrupt 1{
    count++;
    tmp_status = status;
    if (count == 4) {
        count = 0;
        if (type == 0 ){
            if (status == 1){
                status=128;
            }
            else{
                status = status>>1;
            }
        } else if (type == 1) {
            if (status==128){
                status=1;
            }
            else{
                status = status<<1;
            }
        } else if (type == 2) {
            //event even bit blinck
            P2 = 170;
            TH0 = 0;
            TL0 = 0;
            count = 0;
            return;
        } else if (type == 3) {
            //event odd bit blinck
            P2 = 85;
            TH0 = 0;
            TL0 = 0;
            count = 0;
            return;
        }
        P2 = status;
    } else if (count == 2 && (type == 2 || type == 3)) {
        P2 = 00000000;
    }

    TH0 = 0;
    TL0 = 0;
}//end of function Timer0_ISR
```

## Difficulties you've encountered and your solution:

這部分遇到的最大問題就是如何把按鈕狀態寫進一個 variable 內並且在 timer interrupt 的時候偵測狀態輸出對應訊號，這邊我們利用在 main function 對 type 狀態進行輪轉確保他不會出現 0~4 以外的數，而當 timer interrupt 的時候就會去找這個 gaolable var. 的值再進行相對訊號的輸出。

## Discussion:

在這個實驗內我學到了如何在 8051 CPU 做 button debounce 以及如何在 timer interrupt 的時候做出 output 的狀態轉換，以達成（基數燈閃爍、偶數燈閃爍、燈號往右移和燈號往左移）的輸出。