微處理機 LAB 5 Clock and Timer

Due ： 兩週後 早上 8:00

110611052 郭宗諺
110611063 林穎沛

PART 1. (10%) 實作題

Lab 5.1 Modify system initial clock:

請完成實驗 錄影紀錄實驗結果並附上程式碼(main.c 及 include 之.h, .c 檔案)

• 在範例程式 Lab 5.1 中新增 20 MHz 的 system clock 頻率，讓 LED 閃爍頻率做出對應改變。

```
#include "stm32l476xx.h"
#include "helper_functions.h"
#include "7seg.h"
#include "keypad.h"
#include "led_button.h"
#include "timer.h"

// Define pins for 7seg
#define SEG_gpio GPIOC
#define DIN_pin 3
#define CS_pin 4
#define CLK_pin 5

// Define pins for keypad
#define COL_gpio GPIOA
#define COL_pin 6       // 6 7 8 9
#define ROW_gpio GPIOB
#define ROW_pin 3       // 3 4 5 6

// Define pins for led (default use on-board led PA5)
#define LED_gpio GPIOA
#define LED_pin 5

// Define pins for button (default use on-board button PC13)
#define BUTTON_gpio GPIOC
#define BUTTON_pin 13

// Define Counter timer
```

```c
#define COUNTER_timer TIM2

// Buzzer is fixed to PA0 due to its need for PWM signal
// Can change to other ports if needed, but need to look up the reference

// Use to decide which part of the code will run
// Use define & ifdef to control
#define lab_modify_system_clock
//#define lab_counter
//#define lab_music_keyboard
//#define lab_music_song

int main(){
    // Cause we want to use floating points we need to init FPU
    FPU_init();

#ifdef lab_modify_system_clock

    if(init_led(LED_gpio, LED_pin) != 0){
        // Fail to init led
        return -1;
    }
    if(init_button(BUTTON_gpio, BUTTON_pin) != 0){
        // Fail to init button
        return -1;
    }

    int speed=0, trans[5]={1, 6, 10, 16, 20};
    SystemClock_Config(trans[speed]);

    // Used to indicate led state: un-lit(0) or lit(1)
    int state=0;
    // button_press_cycle_per_second (How many button press segments in a
second)
    int button_press_cycle_per_second = 10;
    // Use to state how many cycles to check per button_press_cycle
    int debounce_cycles = 100;
    // Use to state the threshold when we consider a button press
```

```
    int debounce_threshold = debounce_cycles*0.7;
    // Used to implement negative edge trigger 0=not-presses 1=pressed
    int last_button_state=0;

    while(1){


        for(int a=0;a<button_press_cycle_per_second;a++){
            // Simple Debounce without interrupt
            int pos_cnt=0;
            for(int a=0;a<debounce_cycles;a++){
                // If button press add count
                if(read_gpio(BUTTON_gpio, BUTTON_pin)==0){
                    pos_cnt++;
                }
                delay_without_interrupt(1000/(button_press_cycle_per_sec
ond*debounce_cycles));
            }
            // Check if need to change state
            if(pos_cnt>debounce_threshold){
                if(last_button_state==0){
                    // Pressed button - Pos edge
                    // Do nothing
                }
                else{
                    // Pressed button - Continued pressing
                    // Do nothing
                }
                last_button_state = 1;
            }
            else{
                if(last_button_state==0){
                    // Released button - Not pressing
                    // Do nothing
                }
                else{
                    // Released button - Neg edge
                    // Change speed and change system clock
                    speed = (speed+1)%5;
```

```
                    SystemClock_Config(trans[speed]);
                }
                last_button_state = 0;
            }
        }
        if(state==1){
            reset_gpio(LED_gpio, LED_pin);
        }
        else{
            set_gpio(LED_gpio, LED_pin);
        }
        state = 1-state;

    }

#endif



    // Leave a empty while loop in order to stop it from
    // jumping back to startup script's LoopForever
    //
    // Useful when debugging using debugger:
    //   If jump to "LoopForever" means unexpected error happens
    //   Else, the code has ended
    while(1){}

    return 0;
}
```

PART 2. (40%) 實作題

Lab 5.2 Timer

請完成實驗 錄影及截圖紀錄實驗結果並附上程式碼(main.c 及 include 之.h, .c 檔案)

- 使用 STM32 timer 實做一個計時器會從 0 上數(Upcounting) TIME_SEC 秒(自訂)的時間。顯示到小數點以下第二位，結束時 7-SEG LED 停留在 TIME_SEC 的數字。(建議使用擁用比較高 counter resolution 的 TIM2~TIM5 timer)，取得 timer CNT register 值並換算成時間顯示到 7-SEG LED 上。

- 0.01 ≤ TIME_SEC ≤ 10000.00 (超過範圍請直接顯示 0.00)
- Note: 7-SEG LED 驅動請利用之前 Lab 所實作的 GPIO_init()、7-segment_init()與 Display()等等函式呈現(須改成可呈現 2 個小數位)。

```
#include "stm32l476xx.h"
#include "helper_functions.h"
#include "7seg.h"
#include "keypad.h"
#include "led_button.h"
#include "timer.h"

// Define pins for 7seg
#define SEG_gpio GPIOC
#define DIN_pin 3
#define CS_pin 4
#define CLK_pin 5

// Define pins for keypad
#define COL_gpio GPIOA
#define COL_pin 6       // 6 7 8 9
#define ROW_gpio GPIOB
#define ROW_pin 3       // 3 4 5 6

// Define pins for led (default use on-board led PA5)
#define LED_gpio GPIOA
#define LED_pin 5

// Define pins for button (default use on-board button PC13)
#define BUTTON_gpio GPIOC
#define BUTTON_pin 13

// Define Counter timer
#define COUNTER_timer TIM2

// Buzzer is fixed to PA0 due to its need for PWM signal
// Can change to other ports if needed, but need to look up the reference

// Use to decide which part of the code will run
// Use define & ifdef to control
```

```c
#define lab_modify_system_clock
//#define lab_counter
//#define lab_music_keyboard
//#define lab_music_song

int main(){
    // Cause we want to use floating points we need to init FPU
    FPU_init();

    #ifdef lab_modify_system_clock

    int SEG_ADDRESS_DIGIT[8] = {
        SEG_ADDRESS_DIGIT_0,
        SEG_ADDRESS_DIGIT_1,
        SEG_ADDRESS_DIGIT_2,
        SEG_ADDRESS_DIGIT_3,
        SEG_ADDRESS_DIGIT_4,
        SEG_ADDRESS_DIGIT_5,
        SEG_ADDRESS_DIGIT_6,
        SEG_ADDRESS_DIGIT_7,
        /*SEG_ADDRESS_DECODE_MODE,
        SEG_ADDRESS_ITENSITY,
        SEG_ADDRESS_SCAN_LIMIT,
        SEG_ADDRESS_SHUTDOWN,
        SEG_ADDRESS_DISPLAY_TEST*/

    };

    if(init_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin) != 0){
            // Fail to init 7seg
        return -1;
    }
    if(init_button(BUTTON_gpio, BUTTON_pin) != 0 ){
        return -1;
    }
    // Set Decode Mode to non-decode mode
    send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin,
SEG_ADDRESS_DECODE_MODE, 0xFF);
```

```
    // Set Scan Limit to digit 0 only
    send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin,
SEG_ADDRESS_SCAN_LIMIT, 0xFF);
    // Wakeup 7seg
    send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin,
SEG_ADDRESS_SHUTDOWN, 0x01);



    for(int i=0; i<8; i++){//reset display
            send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin,
SEG_ADDRESS_DIGIT[i],SEG_DATA_DECODE_BLANK);

    }
    //begin with 0
    send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin,
SEG_ADDRESS_DIGIT[0],0);



    if(init_button(BUTTON_gpio, BUTTON_pin) != 0){
        // Fail to init button
        return -1;
    }

    int state=0;
    // button_press_cycle_per_second (How many button press segments
in a second)
    int button_press_cycle_per_second = 10;
    // Use to state how many cycles to check per button_press_cycle
    int debounce_cycles = 100;
    // Use to state the threshold when we consider a button press
    int debounce_threshold = debounce_cycles*0.7;
    // Used to implement negative edge trigger 0=not-presses 1=pressed
    int last_button_state=0;

    double TIME_SEC = 123.45;
```

```
    if(TIME_SEC < 0.0 || TIME_SEC >10000.0){
        display_two_decimal(SEG_gpio, DIN_pin, CS_pin, CLK_pin,
0.0);
    }else{
        //Enable timer
        timer_enable(COUNTER_timer);
        // init the timer
        timer_init(COUNTER_timer, 40000, 100);
        // Start the timer
        timer_start(COUNTER_timer);
    }

    int sec=0, last=0;

    while(1){
        if(last!=COUNTER_timer->CNT){
            if(COUNTER_timer->CNT==0){
                // one second has pass
            sec++;
            }
            last = COUNTER_timer->CNT;
            double now_time = sec + COUNTER_timer->CNT/100.0;
            if(TIME_SEC < 0.0 || TIME_SEC >10000.0){
                display_two_decimal(SEG_gpio, DIN_pin, CS_pin,
CLK_pin, 0.0);
            }else{
            display_two_decimal(SEG_gpio, DIN_pin, CS_pin, CLK_pin,
now_time);
            }
            if(now_time==TIME_SEC){
                break;
            }
        }


    }

#endif
```

```
    // Leave a empty while loop in order to stop it from
    // jumping back to startup script's LoopForever
    //
    // Useful when debugging using debugger:
    //   If jump to "LoopForever" means unexpected error happens
    //   Else, the code has ended
    while(1){}

    return 0;
}
```

PART 3. (40%) 實作題

Lab 5.3 電子琴

請完成實驗 錄影及截圖紀錄實驗結果並附上程式碼(main.c 及 include 之.h, .c 檔案)

- 製作電子琴，用 Timer 製作出所需要的頻率波形並連到變頻喇叭發出 Do, Re, Mi 等音符，八個 keypad 按鍵 0~7 分別對應投影片中音階頻率表中第三度的 Do ~Si 和高音 Do。
-

注意

1.輸出正反變換一次為一個週期。或者可使用 PWM mode 輸出，參考 Reference Manual 26.3.11 PWM mode。

2.喇叭接 Vcc 和輸出接腳 or GND 和輸出接腳 都可。

	X0	X1	X2	X3
Y0	Do	Re	Mi	
Y1	Fa	So	La	
Y2	Si	HDo		
Y3				

Keypad 對應音名

音名	Do	Re	Mi	Fa	So	La	Si	HDo
頻率(Hz)	261.6	293.7	329.6	349.2	392.0	440.0	493.9	523.3

音名頻率對應表

```c
#include "stm32l476xx.h"
#include "helper_functions.h"
#include "7seg.h"
#include "keypad.h"
#include "led_button.h"
#include "timer.h"

// Define pins for 7seg
#define SEG_gpio GPIOC
#define DIN_pin 3
#define CS_pin 4
#define CLK_pin 5

// Define pins for keypad
#define COL_gpio GPIOA
#define COL_pin 5// 6 7 8 9
#define ROW_gpio GPIOB
#define ROW_pin 3// 3 4 5 6

// Define pins for led (default use on-board led PA5)
#define LED_gpio GPIOA
#define LED_pin 5

// Define pins for button (default use on-board button PC13)
#define BUTTON_gpio GPIOC
#define BUTTON_pin 13

// Define Counter timer
#define COUNTER_timer TIM2

// Buzzer is fixed to PA0 due to its need for PWM signal
// Can change to other ports if needed, but need to look up the reference

// Use to decide which part of the code will run
// Use define & ifdef to control
#define lab_modify_system_clock
//#define lab_counter
//#define lab_music_keyboard
```

```
//#define lab_music_song

int key_last = 0;

int main(){


    if(init_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin) != 0){
        // Fail to init 7seg
        return -1;
    }

    // Set Decode Mode to Code B decode mode
    send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_DECODE_MODE,
0xFF);
    // Set Scan Limit to all digits
    send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_SCAN_LIMIT,
0x07);
    // Wakeup 7seg
    send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_SHUTDOWN,
0x01);

    if(init_keypad(ROW_gpio, COL_gpio, ROW_pin, COL_pin) != 0){
        // Fail to init keypad
        return -1;
    }
    GPIO_init_AF();
    timer_enable(TIM2);

    PWM_channel_init();
    //timer_init(TIM2,383,20); //170.3 383
    timer_start(TIM2);
    while(1){
        int input = 0;
        for(int i=0;i<4;i++){
            for(int j=0;j<4;j++){
                if(check_keypad_input_one(ROW_gpio, COL_gpio, ROW_pin,
COL_pin, i, j)){
                    input = 1;
```

```
                    display_number(SEG_gpio, DIN_pin, CS_pin, CLK_pin,
keypad[i][j], num_digits(keypad[i][j]));
                    sound(keypad[i][j]);
                }else{

                }
            }
        }
        if(input == 0){
            display_number(SEG_gpio, DIN_pin, CS_pin, CLK_pin, 0, 0);
            sound(0);
        }

    }



    while(1){}

    return 0;
}

void sound(int key){
    if(key!=key_last){
        key_last = key;
        timer_stop(TIM2);
        switch(key){
            case 0:
                timer_init(TIM2,0,0);
                break;
            case 1:
                timer_init(TIM2,383,20);
                break;
            case 2:
                timer_init(TIM2,341,20);
                break;
            case 3:
                timer_init(TIM2,303,20);
                break;
```

```
            case 4:
                timer_init(TIM2,286,20);
                break;
            case 5:
                timer_init(TIM2,255,20);
                break;
            case 6:
                timer_init(TIM2,227,20);
                break;
            case 7:
                timer_init(TIM2,202,20);
                break;
            case 8:
                timer_init(TIM2,191,20);
                break;
        }
        timer_start(TIM2);
    }

}
```

PART 4. (10%) 問答題

1. 說明 Sysclk 和 timer 內 clk 差異 (5%)
   Sysclk 會直接影響程序運行的速度，但 timer 內 clk 的運行是獨立於主程序的運行的
2. 說明如何在 ARM 中設定生成 PWM，參考 Reference Manual 26.3.11 PWM。
   使用 timer 設定 duty cycle 決定 PWM 頻率。

PART 4.加分題

5.4. Music 音色實驗(15%)

請完成實驗 錄影紀錄實驗結果並附上程式碼(main.s 及 include 之 pin.s 檔案)在前一實驗(Lab 5.3 電子琴)中的 keypad 增加 2 個功能按鈕用以調整 PWM 輸出的 Duty cycle(範圍 10%~90%，每按一次鍵調整 5%)，觀察是否會影響蜂鳴器所發出的聲音大小或音色。

Note: 須注意頻率與 duty cycle 的關係來設定 timer ARR 與 CCR registers。可用 LED 或者錄音測試 duty cycle 是否有改變，成功應會看到 LED 隨著 duty cycle 不同而有明暗變化。

```c
#include "stm32l476xx.h"
#include "helper_functions.h"
#include "7seg.h"
#include "keypad.h"
#include "led_button.h"
#include "timer.h"

// Define pins for 7seg
#define SEG_gpio GPIOC
#define DIN_pin 3
#define CS_pin 4
#define CLK_pin 5

// Define pins for keypad
#define COL_gpio GPIOA
#define COL_pin 5// 6 7 8 9
#define ROW_gpio GPIOB
#define ROW_pin 3// 3 4 5 6

// Define pins for led (default use on-board led PA5)
#define LED_gpio GPIOA
#define LED_pin 5

// Define pins for button (default use on-board button PC13)
#define BUTTON_gpio GPIOC
#define BUTTON_pin 13

// Define Counter timer
#define COUNTER_timer TIM2

// Buzzer is fixed to PA0 due to its need for PWM signal
// Can change to other ports if needed, but need to look up the reference

// Use to decide which part of the code will run
// Use define & ifdef to control
#define lab_modify_system_clock
//#define lab_counter
//#define lab_music_keyboard
```

```
//#define lab_music_song

int key_last = 0;
double PWM_percent = 1;
int main(){


    if(init_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin) != 0){
        // Fail to init 7seg
        return -1;
    }

    // Set Decode Mode to Code B decode mode
    send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_DECODE_MODE,
0xFF);
    // Set Scan Limit to all digits
    send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_SCAN_LIMIT,
0x07);
    // Wakeup 7seg
    send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_SHUTDOWN,
0x01);

    if(init_keypad(ROW_gpio, COL_gpio, ROW_pin, COL_pin) != 0){
        // Fail to init keypad
        return -1;
    }
    GPIO_init_AF();
    timer_enable(TIM2);

    PWM_channel_init();
    //timer_init(TIM2,383,20); //170.3 383
    timer_start(TIM2);
    while(1){
        int input = 0;
        for(int i=0;i<4;i++){
            for(int j=0;j<4;j++){
                if(check_keypad_input_one(ROW_gpio, COL_gpio, ROW_pin,
COL_pin, i, j)){
                    input = 1;
```

```
                    display_number(SEG_gpio, DIN_pin, CS_pin, CLK_pin,
keypad[i][j], num_digits(keypad[i][j]));
                    sound(keypad[i][j]);
                }else{

                }
            }
        }
        if(input == 0){
            display_number(SEG_gpio, DIN_pin, CS_pin, CLK_pin, 0, 0);
            sound(0);
        }

    }



    while(1){}

    return 0;
}

void sound(int key){
    if(key!=key_last){
        key_last = key;
        timer_stop(TIM2);
        switch(key){
            case 0:
                timer_init(TIM2,0,0);
                break;
            case 1:
                timer_init(TIM2,383*PWM_percent,20);
                break;
            case 2:
                timer_init(TIM2,341*PWM_percent,20);
                break;
            case 3:
                timer_init(TIM2,303*PWM_percent,20);
                break;
```

```
            case 4:
                timer_init(TIM2,286*PWM_percent,20);
                break;
            case 5:
                timer_init(TIM2,255*PWM_percent,20);
                break;
            case 6:
                timer_init(TIM2,227*PWM_percent,20);
                break;
            case 7:
                timer_init(TIM2,202*PWM_percent,20);
                break;
            case 8:
                timer_init(TIM2,191*PWM_percent,20);
                break;
            case 13:
                if(PWM_percent <= 0.95){
                    PWM_percent=PWM_percent + 0.05;
                }
                break;
            case 14:
                if(PWM_percent >= 0.15){
                    PWM_percent=PWM_percent - 0.05;
                }
                break;
        }
        timer_start(TIM2);
    }

}
```

本作業參考自: DCP1155 Microprocessor System Lab 2016
曹孝櫟教授 國立交通大學 資訊工程學系 Lab7