微處理機 LAB 7 ADC

Due ： 兩週後 早上 8:00

PART 1. (10%) 實作題

Lab 7.1 ADC:

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

˙不使用 SysTick 中斷，單純執行一次 ADC 取樣光敏電阻值，並把取樣結果顯示在 7-segmemt 上。

```c
#include "stm32l476xx.h"
#include "helper_functions.h"
#include "usart.h"
#include "adc.h"
#include "led_button.h"
#include "7seg.h"
#include "stdio.h"

#define lab10_2
#define SEG_gpio GPIOC
#define DIN_pin 3
#define CS_pin 4
#define CLK_pin 5
double resistor;
int adcVal = 0;

void GPIO_init(){
    // Setup FPU
    SCB->CPACR |= (0xF << 20);
    __DSB();
    __ISB();

    RCC->AHB2ENR |= RCC_AHB2ENR_GPIOAEN;   // PA5 LED
    RCC->AHB2ENR |= RCC_AHB2ENR_GPIOCEN;   // PC13 Button
    RCC->AHB2ENR |= RCC_AHB2ENR_ADCEN;     // ADC

    // Setup Button
    GPIOC->MODER &= ~GPIO_MODER_MODE13_Msk;
    GPIOC->MODER |= (0x0 << GPIO_MODER_MODE13_Pos);

    // Setup PA5 LED
    GPIOA->MODER &= ~GPIO_MODER_MODE5_Msk;
    GPIOA->MODER |= (0x1 << GPIO_MODER_MODE5_Pos);

    // PC0 ADC
    GPIOC->MODER &= ~GPIO_MODER_MODE0_Msk;
    GPIOC->MODER |= (0x3 << GPIO_MODER_MODE0_Pos);
    GPIOC->ASCR |= GPIO_ASCR_ASC0;

}

void EXTI_Setup(){
    // Enable SYSCFG CLK
    RCC->APB2ENR |= RCC_APB2ENR_SYSCFGEN;
    // Select output bits
    SYSCFG->EXTICR[3] &= ~SYSCFG_EXTICR4_EXTI13_Msk;
    SYSCFG->EXTICR[3] |= (2 << SYSCFG_EXTICR4_EXTI13_Pos);
```

```
    SYSCFG->EXTICR[3] &= ~SYSCFG_EXTICR4_EXTI13_Msk;
    SYSCFG->EXTICR[3] |= (2 << SYSCFG_EXTICR4_EXTI13_Pos);
    // Enable interrupt
    EXTI->IMR1 |= EXTI_IMR1_IM13;
    // Enable Falling Edge
    EXTI->RTSR1 |= EXTI_RTSR1_RT13;
    // Enable NVIC
    NVIC_EnableIRQ(EXTI15_10_IRQn);
}

void ADCInit(){
    ADCResolution(ADC1, 0);              // 12 bits
    ADCContinuousConversion(ADC1, 0);    // enable continuous conversion
    ADCDataAlign(ADC1, 0);               // set right align
    ADCCommonDualMode(0);                // independent mode
    ADCCommonClockMode(1);               // hclk / 1
    ADCCommonPrescaler(0);               // div 1
    ADCCommonDMAMode(0);                 // disable dma
    ADCCommonDelayTwoSampling(0b0100);   // 5 adc clk cycle
    ADCChannel(ADC1, 1, 1, 2);           // channel 1, rank 1, 12.5 adc clock cycle
    ADCWakeup(ADC1);
    ADCInterrupt(ADC1, ADC_IER_EOCIE, 1);
    NVIC_EnableIRQ(ADC1_2_IRQn);
    ADCEnable(ADC1);
}

void SysTick_Handler() {
    if(SysTick->CTRL & SysTick_CTRL_COUNTFLAG_Msk){
        //ADCStartConversion(ADC1);
        light();
        handler();
        toggle_output(GPIOA,5);
    }
    return;
}

void ADC1_2_IRQHandler(){
    if(ADC1->ISR & ADC_ISR_EOC){
        ADC1->ISR &= ADC_ISR_EOC;
        resistor = ADCGetValue(ADC1);
        resistor = (3.3 / 4095) * resistor;
        resistor = (3.3 - resistor) * 2030 / resistor;
    }
}

void startADC() {
    while (!(ADC1->ISR & ADC_ISR_ADRDY)) ADC1->CR |= ADC_CR_ADEN; // TURN ON
    ADC1->ISR = ADC_ISR_EOC | ADC_ISR_EOS | ADC_ISR_OVR; // Clear flags
    ADC1->CR |= ADC_CR_ADSTART; // START CONV
}

void light() {
    startADC();
    while (!(ADC1->ISR & ADC_ISR_EOC));
    adcVal = ADC1->DR;
}

void handler(){
    double Vsample = (3.3/4095)*adcVal;
    double current = (3.3 - Vsample)/10000;
    adcVal = Vsample/current;
    send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_DIGIT_0, adcVal%10);
    adcVal/=10;
    send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_DIGIT_1, adcVal%10);
    adcVal/=10;
    send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_DIGIT_2, adcVal%10);
    adcVal/=10;
    send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_DIGIT_3, adcVal%10);
    adcVal/=10;
    send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_DIGIT_4, adcVal%10);
    adcVal/=10;
```

```c
void lab_10_2(){
    EXTI_Setup();
//  USARTInit();
    ADCInit();
    //SystemClock_Config_Interrupt(4, 1000000);
    init_led(GPIOA,5);
    light();
    handler();
    while(1){
        //light();
        //ADC1_2_IRQHandler();
        //EXTI15_10_IRQHandler();
        //TIM2_IRQHandler();
        //display_two_decimal(SEG_gpio, DIN_pin, CS_pin, CLK_pin, adcVal);
    }
}

int main(){
    GPIO_init();
    if(init_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin) != 0){
                    return -1;
                }
    send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_DECODE_MODE, 0xFF);
    send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_SCAN_LIMIT, 0x04);
    send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_SHUTDOWN, 0x01);

    lab_10_2();
    while(1){}
    return 0;
}
```

˙調慢 ADC 取樣間隔到每秒取樣做 ADC 一次，不可改變 sys_clk，然後把取樣結果顯示在 7-segmemt 上。

```c
#include "stm321476xx.h"
#include "helper_functions.h"
#include "usart.h"
#include "adc.h"
#include "led_button.h"
#include "7seg.h"
#include "stdio.h"

#define lab10_2
#define SEG_gpio GPIOC
#define DIN_pin 3
#define CS_pin 4
#define CLK_pin 5
double resistor;
int adcVal = 0;

void GPIO_init(){
    // Setup FPU
    SCB->CPACR |= (0xF << 20);
    __DSB();
    __ISB();

    RCC->AHB2ENR |= RCC_AHB2ENR_GPIOAEN;    // PA5 LED
    RCC->AHB2ENR |= RCC_AHB2ENR_GPIOCEN;    // PC13 Button
    RCC->AHB2ENR |= RCC_AHB2ENR_ADCEN;      // ADC

    // Setup Button
```

```
    // Setup Button
    GPIOC->MODER &= ~GPIO_MODER_MODE13_Msk;
    GPIOC->MODER |= (0x0 << GPIO_MODER_MODE13_Pos);

    // Setup PA5 LED
    GPIOA->MODER &= ~GPIO_MODER_MODE5_Msk;
    GPIOA->MODER |= (0x1 << GPIO_MODER_MODE5_Pos);

    // PC0 ADC
    GPIOC->MODER &= ~GPIO_MODER_MODE0_Msk;
    GPIOC->MODER |= (0x3 << GPIO_MODER_MODE0_Pos);
    GPIOC->ASCR |= GPIO_ASCR_ASC0;
}

void EXTI_Setup(){
    // Enable SYSCFG CLK
    RCC->APB2ENR |= RCC_APB2ENR_SYSCFGEN;
    // Select output bits
    SYSCFG->EXTICR[3] &= ~SYSCFG_EXTICR4_EXTI13_Msk;
    SYSCFG->EXTICR[3] |= (2 << SYSCFG_EXTICR4_EXTI13_Pos);
    // Enable interrupt
    EXTI->IMR1 |= EXTI_IMR1_IM13;
    // Enable Falling Edge
    EXTI->RTSR1 |= EXTI_RTSR1_RT13;
    // Enable NVIC
    NVIC_EnableIRQ(EXTI15_10_IRQn);
}

void ADCInit(){
    ADCResolution(ADC1, 0);              // 12 bits
    ADCContinuousConversion(ADC1, 0);    // enable continuous conversion
    ADCDataAlign(ADC1, 0);               // set right align
    ADCCommonDualMode(0);                // independent mode
    ADCCommonClockMode(1);               // hclk / 1
    ADCCommonPrescaler(0);               // div 1
    ADCCommonDMAMode(0);                 // disable dma
    ADCCommonDelayTwoSampling(0b0100);   // 5 adc clk cycle
    ADCChannel(ADC1, 1, 1, 2);           // channel 1, rank 1, 12.5 adc clock cycle
    ADCWakeup(ADC1);
    ADCInterrupt(ADC1, ADC_IER_EOCIE, 1);
    NVIC_EnableIRQ(ADC1_2_IRQn);
    ADCEnable(ADC1);
}

void SysTick_Handler() {
    if(SysTick->CTRL & SysTick_CTRL_COUNTFLAG_Msk){
        //ADCStartConversion(ADC1);
        light();
        handler();
        toggle_output(GPIOA,5);
    }
    return;
}

void ADC1_2_IRQHandler(){
    if(ADC1->ISR & ADC_ISR_EOC){
        ADC1->ISR &= ADC_ISR_EOC;
        resistor = ADCGetValue(ADC1);
        resistor = (3.3 / 4095) * resistor;
        resistor = (3.3 - resistor) * 2000 / resistor;
    }
}

void startADC() {
    while (!(ADC1->ISR & ADC_ISR_ADRDY)) ADC1->CR |= ADC_CR_ADEN; // TURN ON
    ADC1->ISR = ADC_ISR_EOC | ADC_ISR_EOS | ADC_ISR_OVR; // Clear flags
    ADC1->CR |= ADC_CR_ADSTART; // START CONV
}

void light() {
    startADC();
```

```c
void light() {
    startADC();
    while (!(ADC1->ISR & ADC_ISR_EOC));
    adcVal = ADC1->DR;
}
void handler(){
    double Vsample = (3.3/4095)*adcVal;
    double current = (3.3 - Vsample)/10000;
    adcVal = Vsample/current;
    send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_DIGIT_0, adcVal%10);
    adcVal/=10;
    send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_DIGIT_1, adcVal%10);
    adcVal/=10;
    send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_DIGIT_2, adcVal%10);
    adcVal/=10;
    send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_DIGIT_3, adcVal%10);
    adcVal/=10;
    send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_DIGIT_4, adcVal%10);
    adcVal/=10;
}

void lab_10_2(){
    EXTI_Setup();
//  USARTInit();
    ADCInit();

    SystemClock_Config_Interrupt(4, 1000000);

    init_led(GPIOA,5);
    while(1){
        //light();
        //ADC1_2_IRQHandler();
        //EXTI15_10_IRQHandler();
        //TIM2_IRQHandler();
        //display_two_decimal(SEG_gpio, DIN_pin, CS_pin, CLK_pin, adcVal);
    }
}

int main(){
    GPIO_init();
    if(init_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin) != 0){
                    return -1;
                }
    send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_DECODE_MODE, 0xFF);
    send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_SCAN_LIMIT, 0x04);
    send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_SHUTDOWN, 0x01);

    lab_10_2();
    while(1){}
    return 0;
}
```

PART 2. (40%) 實作題

Lab 7.2: ADC 變壓輸入:

請完成實驗 錄影紀錄實驗結果

- 選用三個不同電阻取代光敏電阻，計算出這三個電阻跨壓，不可高於 5V，用 ADC 分別讀取這三個跨壓，，然後把取樣結果顯示在 7-segmemt 上。
- 計算出電阻值與 ADC 讀值的轉換關係式並寫入程式中，按按鍵(blue button)切換顯示電阻值與 ADC 讀值。

```c
#include "stm32l476xx.h"
#include "helper_functions.h"
#include "usart.h"
#include "adc.h"
#include "led_button.h"
#include "7seg.h"
#include "stdio.h"

#define BUTTON_gpio GPIOC
#define BUTTON_pin 13
#define lab10_2
#define SEG_gpio GPIOC
#define DIN_pin 3
#define CS_pin 4
#define CLK_pin 5

double resistor;
int adcVal = 0;
int state = 0;
void GPIO_init(){
    // Setup FPU
    SCB->CPACR |= (0xF << 20);
    __DSB();
    __ISB();

    RCC->AHB2ENR |= RCC_AHB2ENR_GPIOAEN;   // PA5 LED
    RCC->AHB2ENR |= RCC_AHB2ENR_GPIOCEN;   // PC13 Button
    RCC->AHB2ENR |= RCC_AHB2ENR_ADCEN;     // ADC

    // Setup Button
    GPIOC->MODER &= ~GPIO_MODER_MODE13_Msk;
    GPIOC->MODER |= (0x0 << GPIO_MODER_MODE13_Pos);

    // Setup PA5 LED
    GPIOA->MODER &= ~GPIO_MODER_MODE5_Msk;
    GPIOA->MODER |= (0x1 << GPIO_MODER_MODE5_Pos);

    // PC0 ADC
    GPIOC->MODER &= ~GPIO_MODER_MODE0_Msk;
    GPIOC->MODER |= (0x3 << GPIO_MODER_MODE0_Pos);
    GPIOC->ASCR |= GPIO_ASCR_ASC0;
}

void EXTI_Setup(){
    // Enable SYSCFG CLK
    RCC->APB2ENR |= RCC_APB2ENR_SYSCFGEN;
    // Select output bits
    SYSCFG->EXTICR[3] &= ~SYSCFG_EXTICR4_EXTI13_Msk;
    SYSCFG->EXTICR[3] |= (2 << SYSCFG_EXTICR4_EXTI13_Pos);
    // Enable interrupt
    EXTI->IMR1 |= EXTI_IMR1_IM13;
```

```
    EXTI->IMR1 |= EXTI_IMR1_IM13;
    // Enable Falling Edge
    EXTI->RTSR1 |= EXTI_RTSR1_RT13;
    // Enable NVIC
    NVIC_EnableIRQ(EXTI15_10_IRQn);
}

void ADCInit(){
    ADCResolution(ADC1, 0);              // 12 bits
    ADCContinuousConversion(ADC1, 0);    // enable continuous conversion
    ADCDataAlign(ADC1, 0);               // set right align
    ADCCommonDualMode(0);                // independent mode
    ADCCommonClockMode(1);               // hclk / 1
    ADCCommonPrescaler(0);               // div 1
    ADCCommonDMAMode(0);                 // disable dma
    ADCCommonDelayTwoSampling(0b0100);   // 5 adc clk cycle
    ADCChannel(ADC1, 1, 1, 2);           // channel 1, rank 1, 12.5 adc clock cycle
    ADCWakeup(ADC1);
    ADCInterrupt(ADC1, ADC_IER_EOCIE, 1);
    NVIC_EnableIRQ(ADC1_2_IRQn);
    ADCEnable(ADC1);
}

void SysTick_Handler() {
    if(SysTick->CTRL & SysTick_CTRL_COUNTFLAG_Msk){
        //ADCStartConversion(ADC1);
        light();
        handler();
        toggle_output(GPIOA,5);
    }
    return;
}

void ADC1_2_IRQHandler(){
    if(ADC1->ISR & ADC_ISR_EOC){
        ADC1->ISR &= ADC_ISR_EOC;
        resistor = ADCGetValue(ADC1);
        resistor = (3.3 / 4095) * resistor;
        resistor = (3.3 - resistor) * 2000 / resistor;
    }
}

void startADC() {
    while (!(ADC1->ISR & ADC_ISR_ADRDY)) ADC1->CR |= ADC_CR_ADEN; // TURN ON
    ADC1->ISR = ADC_ISR_EOC | ADC_ISR_EOS | ADC_ISR_OVR; // Clear flags
    ADC1->CR |= ADC_CR_ADSTART; // START CONV
}

void light() {
    startADC();
    while (!(ADC1->ISR & ADC_ISR_EOC));
    adcVal = ADC1->DR;
}

void handler(){
    double Vsample = (3.3/4095)*adcVal;
    double current =  Vsample/10000;
    if(state==1){
        display_two_decimal(SEG_gpio, DIN_pin, CS_pin, CLK_pin, (3.3 - Vsample));
    }else{
        int k=0;
        int temp;
        adcVal = (3.3 - Vsample)/current;
        while(adcVal !=0){
            adcVal = adcVal/10;
            k++;
        }
        if(k>8){
            k=0;
        }
        adcVal = (3.3 - Vsample)/current;
```

```
        adcVal = (3.3 - Vsample)/current;
        display_number(SEG_gpio, DIN_pin, CS_pin, CLK_pin, adcVal, k);
    }
}

void lab_10_2(){
    //EXTI_Setup();
//  USARTInit();
    ADCInit();
    SystemClock_Config_Interrupt(4, 1000000);
    init_led(GPIOA,5);
    while(1){
        //light();
        //ADC1_2_IRQHandler();
        //EXTI15_10_IRQHandler();
        //TIM2_IRQHandler();
        //display_two_decimal(SEG_gpio, DIN_pin, CS_pin, CLK_pin, adcVal);

        int ch_state = 0;
        int button_press_persecond_cycle = 10;
        int debounce_cycle = 100;
        int debounce_threshold = 70;
        int last_botton_pos =1;

        for(int i=0; i<button_press_persecond_cycle; i++){
            int pos_cnt = 0; //count
            for(int a=0; a<debounce_cycle; a++){
                if(read_gpio(BUTTON_gpio, BUTTON_pin) == 0){
                    pos_cnt++;
                }
            }
            if(pos_cnt > debounce_threshold){
                if(last_botton_pos == 1){
                    last_botton_pos = 0;
                }
            }else{
                if(last_botton_pos == 0){
                    if(state == 0){
                        state = 1;
                    }else{
                        state = 0;
                    }
                    last_botton_pos = 1;
                }
            }
        }
    }
}

int main(){
    GPIO_init();
    if(init_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin) != 0){
                    return -1;
                }
    send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_DECODE_MODE, 0xFF);
    send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_SCAN_LIMIT, 0xFF);
    send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_SHUTDOWN, 0x01);

    lab_10_2();
    while(1){}
    return 0;
}
```

PART 3. (50%) 實作題

Lab 7.3: 練習馬達控制

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)

利用 SysTick timer 和 SG90 伺服馬達，每 1 秒順時鐘轉動 45 度，4 秒達到 180 度後，停頓一秒後每秒往逆時鐘方向旋轉 45 度直到回到原點。

```c
#include "stm32l476xx.h"
#include "timer.h"

int times = 1;
void SysTick_Handler() {
    if(SysTick->CTRL & SysTick_CTRL_COUNTFLAG_Msk){
        times ++;
        //-------- timer_init(TIM2, 1000, 20000); ---------//
        TIM2->PSC = (uint32_t)(1000-1);          // PreScalser
        TIM2->ARR = (uint32_t)(150000-1);          // Reload value
        TIM2->EGR |= TIM_EGR_UG;                // Reinitialize the counter

        //-------- timer_start(TIM2); ---------//
        TIM2->CR1 |= TIM_CR1_CEN;
        if (times % 4 == 1 || times % 4 == 2)
            TIM2->CCR1 = 20;
        else
            TIM2->CCR1 = (uint32_t)(1);
    }
}

int main()
{
    // Configure SysTick clk to 10 Mhz and interrupt every 4s
    SystemClock_Config_Interrupt(10, 40000000);

    //-------- GPIO_init_AF(); ---------//
    RCC->AHB2ENR |= RCC_AHB2ENR_GPIOAEN;
    // Set to Alternate function mode
    GPIOA->MODER &= ~GPIO_MODER_MODE0_Msk;
    GPIOA->MODER |= (2 << GPIO_MODER_MODE0_Pos);
    // Set AFRL
    GPIOA->AFR[0] &= ~GPIO_AFRL_AFSEL0_Msk;
    GPIOA->AFR[0] |= (1 << GPIO_AFRL_AFSEL0_Pos);

    //-------- timer_enable(TIM2); ---------//
    RCC->APB1ENR1 |= RCC_APB1ENR1_TIM2EN;

    // -------- timer_init(TIM2, 1000, 20000); ---------//
    TIM2->PSC = (uint32_t)(10000-1);          // PreScalser
    TIM2->ARR = (uint32_t)(1000-1);          // Reload value
    TIM2->EGR |= TIM_EGR_UG;                // Reinitialize the counter

    //-------- sg90_init(); ---------//
    // PA0 for PWM
    // PWM mode 1
    TIM2->CCMR1 &= ~TIM_CCMR1_OC1M_Msk;
    TIM2->CCMR1 |= (6 << TIM_CCMR1_OC1M_Pos);
    // OCPreload_Enable
```

```
49      // OCPreload_Enable
50      TIM2->CCMR1 &= ~TIM_CCMR1_OC1PE_Msk;
51      TIM2->CCMR1 |= (1 << TIM_CCMR1_OC1PE_Pos);
52      // Active high for channel 1 polarity
53      TIM2->CCER &= ~TIM_CCER_CC1P_Msk;
54      TIM2->CCER |= (0 << TIM_CCER_CC1P_Pos);
55      // Enable for channel 1 output
56      TIM2->CCER &= ~TIM_CCER_CC1E_Msk;
57      TIM2->CCER |= (1 << TIM_CCER_CC1E_Pos);
58
59      //-------- timer_init(TIM2, 1000, 20000); ---------//
60      TIM2->PSC = (uint32_t)(10000-1);          // PreScalser
61      TIM2->ARR = (uint32_t)(1000-1);          // Reload value
62      TIM2->EGR |= TIM_EGR_UG;                // Reinitialize the counter
63
64      //-------- timer_start(TIM2); ---------//
65      TIM2->CR1 |= TIM_CR1_CEN;
66
67      while (1){
68
69      }
70      return 0;
71  }
```

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