report.md

實驗三 實驗結報

0316213 蒲郁文 & 0316323 薛世恩

實驗名稱

STM32 Clock and Timer

實驗目的

- 瞭解 STM32 的各種 clock source 使用與修改
- 瞭解 STM32 的 timer 使用原理
- 瞭解 STM32 的 PWM 使用原理與應用

實驗步驟

Modify system initial clock

```
#include "stm32l476xx.h"
#include "utils.h"

int plln = 16, pllm = 7, prescaler = 9;
enum {S1MHZ, S6MHz, S10MHZ, S16MHZ, S40MHZ} state = S1MHZ;
int prev_btn = 1, curr_btn = 1;

void SystemClock_Config();

int main()
{
    SystemClock_Config();
    gpio_init();
    while (1)
    {
        if (!prev_btn && curr_btn)
        {
            switch (state)
            {
            case S1MHZ:
                plln = 16;
                pllm = 7;
                prescaler = 9;
                break;
            case S6MHz:
                plln = 24;
                pllm = 7;
                prescaler = 0;
                break;
            case S10MHZ:
                plln = 40;
                pllm = 7;
                prescaler = 0;
                break;
            case S16MHZ:
                plln = 64;
                pllm = 7;
                prescaler = 0;
                break;
            case S40MHZ:
                plln = 20;
                pllm = 0;
```

```
                prescaler =  0;
                 break;
            default:
                 break;
            }
            SystemClock_Config();
            state = state == S40MHZ ? S1MHZ : state +   1;
        }
        GPIOA->BSRR = ( 1 << 5);
        delay_1s();
        GPIOA->BRR = ( 1 << 5);
        delay_1s();
        prev_btn = curr_btn;
        curr_btn =  GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_13);
    }
}

void SystemClock_Config()
{
    RCC->CFGR = 0x00000000;
    // CFGR reset value
    RCC->CR &= 0xFEFFFFFF;
    // main PLL enable: PLL off
    while (RCC->CR & 0x02000000);
    // main PLL clock ready flag: PLL locked
    RCC->PLLCFGR = 0x01000001;
    // main PLL PLLCLK output enable: PLLCLK output enable
    // main PLL entry clock source: MSI clock selected as PLL clock entry
    RCC->PLLCFGR |= plln <<  8;
    // main PLL multiplication factor for VCO
    RCC->PLLCFGR |= pllm <<  4;
    // division factor for the main PLL input clock
    // f(VCO clock) = f(PLL clock input) × (PLLN / PLLM)
    // f(PLL_R) = f(VCO clock) / PLLR
    RCC->CR |= 0x01000000;
    // main PLL enable: PLL on
    while (!(RCC->CR & 0x02000000));
    // main PLL clock ready flag: PLL locked
    RCC->CFGR = 0x00000003;
    // system clock switch: PLL selected as system clock
    RCC->CFGR |= prescaler <<  4;
    // AHB prescaler: SYSCLK divided by N
}
```

計時器

```
#include "stm32l476xx.h"
#include "utils.h"
#define TIME_SEC 11.99

int cal_len(int a)
{
    int sum = 0;
    while (a > 0)
    {
        a /= 10;
        sum++;
    }
    return sum;
}

void timer_init()
{
    RCC->APB1ENR1 |=  0b1;
    TIM2->ARR = (uint32_t) (TIME_SEC * (4000000 / 40000)); // reload value
    TIM2->PSC = (uint32_t) 39999; // prescaler
    TIM2->EGR = TIM_EGR_UG;  // reinitialize the counter
}

void timer_start()
{
    TIM2->CR1 |= TIM_CR1_CEN;
    display(0, -1003);
```

```
    if (TIME_SEC <= 0 || TIME_SEC > 10000)
    {
        TIM2->CR1 &= ~TIM_CR1_CEN;
        return;
    }
    int pre_val = 0;
    while (1)
    {
        int now_val = TIM2->CNT;
        if (pre_val > now_val)
        {
            TIM2->CR1 &= ~TIM_CR1_CEN;
            return;
        }
        pre_val = now_val;
        int len = cal_len(now_val);
        if (now_val < 100)
            len = 3;
        display(now_val, -1000 - len);
    }
}

int main()
{
    gpio_init();
    max7219_init();
    timer_init();
    timer_start();
}
```

Music keypad

```
#include "stm32l476xx.h"
#include "utils.h"

#define DO 261.6
#define RE 293.7
#define MI 329.6
#define FA 349.2
#define SO 392.0
#define LA 440.0
#define SI 493.9
#define HI_DO 523.3

float freq = -1;
int curr = -2, prev = -3, check = -4;
int duty_cycle = 50;

void timer_init()
{
    RCC->APB1ENR1 |= RCC_APB1ENR1_TIM2EN;
    TIM2->ARR = (uint32_t) 100;
    TIM2->PSC = (uint32_t) (4000000 / freq / 100);
    TIM2->EGR = TIM_EGR_UG;
}

void timer_start()
{
    while (1)
    {
        prev = curr;
        curr = keypad_scan();
        if (curr == prev)
            check = 86400;
        else
            check = curr;
        switch (check)
        {
        case 1:
            TIM2->CR1 &= ~TIM_CR1_CEN;
            freq = DO;
            timer_init();
            TIM2->CR1 |= TIM_CR1_CEN;
```

```
            break;
        case 2:
            TIM2->CR1 &= ~TIM_CR1_CEN;
            freq = RE;
            timer_init();
            TIM2->CR1 |= TIM_CR1_CEN;
            break;
        case 3:
            TIM2->CR1 &= ~TIM_CR1_CEN;
            freq = MI;
            timer_init();
            TIM2->CR1 |= TIM_CR1_CEN;
            break;
        case 4:
            TIM2->CR1 &= ~TIM_CR1_CEN;
            freq = FA;
            timer_init();
            TIM2->CR1 |= TIM_CR1_CEN;
            break;
        case 5:
            TIM2->CR1 &= ~TIM_CR1_CEN;
            freq = SO;
            timer_init();
            TIM2->CR1 |= TIM_CR1_CEN;
            break;
        case 6:
            TIM2->CR1 &= ~TIM_CR1_CEN;
            freq = LA;
            timer_init();
            TIM2->CR1 |= TIM_CR1_CEN;
            break;
        case 7:
            TIM2->CR1 &= ~TIM_CR1_CEN;
            freq = SI;
            timer_init();
            TIM2->CR1 |= TIM_CR1_CEN;
            break;
        case 8:
            TIM2->CR1 &= ~TIM_CR1_CEN;
            freq = HI_DO;
            timer_init();
            TIM2->CR1 |= TIM_CR1_CEN;
            break;
        case 10:
            duty_cycle = duty_cycle ==  90 ? duty_cycle : duty_cycle +  5;
            break;
        case 11:
            duty_cycle = duty_cycle ==  10 ? duty_cycle : duty_cycle -  5;
            break;
        case 86400:
            break;
        default:
            TIM2->CR1 &= ~TIM_CR1_CEN;
            freq = - 1;
            break;
        }
        if (freq > 0)
        {
            if (TIM2->CNT < duty_cycle)
                GPIOB->BSRR = ( 1 << 8);
            else
                GPIOB->BRR = ( 1 << 8);
        }
        else
            GPIOB->BRR = ( 1 << 8);
    }
}

int main()
{
    fpu_enable();
    gpio_init();
    keypad_init();
    timer_start();
}
```

(utils.h)

```
#ifndef UTILS_H_
#define UTILS_H_

/**
 * these functions are inside the assembly source file
 */
extern void gpio_init();
extern void max7219_init();
extern void max7219_send(unsigned char address, unsigned char data);
extern void delay_1s();
extern void fpu_enable();

/**
 * show data on 7-segment display via max7219_send
 *
 * input:
 *   data: decimal value
 *   num_digs: number of digits to show on 7-segment display
 *
 * return:
 *   0: success
 *   -1: illegal data range (out of 8 digits)
 */
int display(int data, int num_digs)
{
    int show_dec_pt = 0;
    if (num_digs <= -1000)
    {
        num_digs = -1000 - num_digs ;
        show_dec_pt = 1;
    }
    num_digs = num_digs > 8 ? 8 : num_digs;
    int data2 = data, i;
    for (i = 1; i <= num_digs; i++)
    {
        if (data2 < 0 && i == num_digs);
        else if (show_dec_pt && i == 3 && data % 10 < 0)
            max7219_send(i, -data % 10 | 0b10000000);
        else if (show_dec_pt && i == 3)
            max7219_send(i, data % 10 | 0b10000000);
        else if (data % 10 < 0)
            max7219_send(i, -data % 10);
        else
            max7219_send(i, data % 10);
        data /= 10;
    }
    if (data2 < 0)
        max7219_send(num_digs, 10);
    for ( ; i <= 8; i++)
        max7219_send(i, 15);
    return (data > 99999999 || data < -9999999) ? -1 : 0;
}

/**
 * GPIO pin macros
 */
#define GPIO_Pin_0  0b0000000000000001
#define GPIO_Pin_1  0b0000000000000010
#define GPIO_Pin_2  0b0000000000000100
#define GPIO_Pin_3  0b0000000000001000
#define GPIO_Pin_4  0b0000000000010000
#define GPIO_Pin_5  0b0000000000100000
#define GPIO_Pin_6  0b0000000001000000
#define GPIO_Pin_7  0b0000000010000000
#define GPIO_Pin_8  0b0000000100000000
#define GPIO_Pin_9  0b0000001000000000
#define GPIO_Pin_10 0b0000010000000000
#define GPIO_Pin_11 0b0000100000000000
#define GPIO_Pin_12 0b0001000000000000
#define GPIO_Pin_13 0b0010000000000000
#define GPIO_Pin_14 0b0100000000000000
#define GPIO_Pin_15 0b1000000000000000
```

```
/**
 * read GPIO data
 *
 * input:
 *   port: pointer to GPIO port structure
 *   pin: GPIO pin macro (see above)
 *
 * return:
 *   0: not set
 *   others: set
 */
int GPIO_ReadInputDataBit (GPIO_TypeDef *port,  uint16_t pin) {
    return port->IDR & pin;
}

/**
 * floating point version of display
 */
int displayf(float data, int num_digs)
{
    if (num_digs > 8)
        return display(-1, 2);
    if ((int) (data * 100) % 100)
        return display(data * 100, -1002 - num_digs);
    else
        return display(data, num_digs);
}

/**
 * RCC PLL configuration structure definition
 */
typedef struct
{
    uint32_t PLLState;   // The new state of the PLL
    uint32_t PLLSource;  // PLL entry clock source
    uint32_t PLLM;       // Division factor for PLL VCO input clock
    uint32_t PLLN;       // Multiplication factor for PLL VCO output clock
    uint32_t PLLP;       // Division factor for SAI clock
    uint32_t PLLQ;       // PLLQ: Division factor for SDMMC1, RNG and USB clocks
    uint32_t PLLR;       // Division for the main system clock
} RCC_PLLInitTypeDef;

/**
 * RCC Internal/External Oscillator configuration structure definition
 */
typedef struct
{
    uint32_t OscillatorType;       // The oscillators to be configured
    uint32_t HSEState;             // The new state of the HSE
    uint32_t LSEState;             // The new state of the LSE
    uint32_t HSIState;             // The new state of the HSI
    uint32_t HSICalibrationValue;  // The calibration trimming value
    uint32_t LSIState;             // The new state of the LSI
    uint32_t MSIState;             // The new state of the MSI
    uint32_t MSICalibrationValue;  // The calibration trimming value
    uint32_t MSIClockRange;        // The MSI frequency range
    uint32_t HSI48State;           // The new state of the HSI48
    RCC_PLLInitTypeDef PLL;        // Main PLL structure parameters
} RCC_OscInitTypeDef;

/**
 * RCC System, AHB and APB busses clock configuration structure definition
 */
typedef struct
{
    uint32_t ClockType;       // The clock to be configured
    uint32_t SYSCLKSource;    // The clock source used as system clock (SYSCLK)
    uint32_t AHBCLKDivider;   // The AHB clock (HCLK) divider
    uint32_t APB1CLKDivider;  // The APB1 clock (PCLK1) divider
    uint32_t APB2CLKDivider;  // The APB2 clock (PCLK2) divider
} RCC_ClkInitTypeDef;

/**
 * calculate length of number
```

```
 */
int len(int n)
{
    int sum = 0;
    while (n > 0)
    {
        n /= 10;
        sum++;
    }
    return sum;
}

/**
 * keypad settings, used by keypad_scan
 */
#define XPORT GPIOC
#define YPORT GPIOB
#define X0 GPIO_Pin_0
#define X1 GPIO_Pin_1
#define X2 GPIO_Pin_2
#define X3 GPIO_Pin_3
#define Y0 GPIO_Pin_6
#define Y1 GPIO_Pin_5
#define Y2 GPIO_Pin_4
#define Y3 GPIO_Pin_3

/**
 * initialize keypad GPIO pin, X as output and Y as input
 */
void keypad_init()
{
    RCC->AHB2ENR   |= 0b00000000000000000000000000000110 ;

    GPIOC->MODER   &= 0b11111111111111111111111100000000 ;
    GPIOC->MODER   |= 0b00000000000000000000000001010101 ;
    GPIOC->PUPDR   &= 0b11111111111111111111111100000000 ;
    GPIOC->PUPDR   |= 0b00000000000000000000000001010101 ;
    GPIOC->OSPEEDR &= 0b11111111111111111111111100000000 ;
    GPIOC->OSPEEDR |= 0b00000000000000000000000001010101 ;
    GPIOC->ODR     |= 0b00000000000000000000000000001111 ;

    GPIOB->MODER   &= 0b11111111111111111100000000111111 ;
    GPIOB->PUPDR   &= 0b11111111111111111100000000111111 ;
    GPIOB->PUPDR   |= 0b00000000000000000010101010000000 ;
    GPIOB->OSPEEDR &= 0b11111111111111111111111100000000 ;
    GPIOB->OSPEEDR |= 0b00000000000000000000000001010101 ;
}

/**
 * scan keypad value
 *
 * return:
 *   >=0: key press value
 *   -1: no key press
 */
signed char keypad_scan()
{
    XPORT->BSRR = X0;
    XPORT->BRR  = X1;
    XPORT->BRR  = X2;
    XPORT->BRR  = X3;

    if (GPIO_ReadInputDataBit(YPORT, Y0))
        return 15;
    if (GPIO_ReadInputDataBit(YPORT, Y1))
        return 7;
    if (GPIO_ReadInputDataBit(YPORT, Y2))
        return 4;
    if (GPIO_ReadInputDataBit(YPORT, Y3))
        return 1;

    XPORT->BRR  = X0;
    XPORT->BSRR = X1;
    XPORT->BRR  = X2;
    XPORT->BRR  = X3;
```

```
    if (GPIO_ReadInputDataBit(YPORT, Y0))
        return 0;
    if (GPIO_ReadInputDataBit(YPORT, Y1))
        return 8;
    if (GPIO_ReadInputDataBit(YPORT, Y2))
        return 5;
    if (GPIO_ReadInputDataBit(YPORT, Y3))
        return 2;

    XPORT->BRR  = X0;
    XPORT->BRR  = X1;
    XPORT->BSRR = X2;
    XPORT->BRR  = X3;

    if (GPIO_ReadInputDataBit(YPORT, Y0))
        return 14;
    if (GPIO_ReadInputDataBit(YPORT, Y1))
        return 9;
    if (GPIO_ReadInputDataBit(YPORT, Y2))
        return 6;
    if (GPIO_ReadInputDataBit(YPORT, Y3))
        return 3;

    XPORT->BRR  = X0;
    XPORT->BRR  = X1;
    XPORT->BRR  = X2;
    XPORT->BSRR = X3;

    if (GPIO_ReadInputDataBit(YPORT, Y0))
        return 13;
    if (GPIO_ReadInputDataBit(YPORT, Y1))
        return 12;
    if (GPIO_ReadInputDataBit(YPORT, Y2))
        return 11;
    if (GPIO_ReadInputDataBit(YPORT, Y3))
        return 10;

    return -1;
}

#endif /* UTILS_H_ */
```

(utils.s)

```
    .syntax unified
    .cpu cortex-m4
    .thumb

.text
    .global gpio_init
    .global max7219_init
    .global max7219_send
    .global delay_1s
    .global fpu_enable

    .equ RCC_AHB2ENR,  0x4002104C

    .equ DECODE_MODE,  0x09
    .equ DISPLAY_TEST, 0x0F
    .equ SCAN_LIMIT,   0x0B
    .equ INTENSITY,    0x0A
    .equ SHUTDOWN,     0x0C

    .equ MAX7219_DIN,  0x20 @ PA5
    .equ MAX7219_CS,   0x40 @ PA6
    .equ MAX7219_CLK,  0x80 @ PA7

    .equ GPIOA_BASE,   0x48000000
    .equ BSRR_OFFSET,  0x18 @ set bit
    .equ BRR_OFFSET,   0x28 @ clear bit

    .equ GPIOB_BASE,   0x48000400
```

```
    .equ AFRL_OFFSET,  0x20
    .equ AFRH_OFFSET,  0x24

    .equ GPIOC_BASE,   0x48000800

gpio_init:
    push {r0, r1, r2, lr}

    mov  r0, 0b00000000000000000000000000000111
    ldr  r1, =RCC_AHB2ENR
    str  r0, [r1]

    ldr  r1, =GPIOA_BASE @ GPIOA_MODER
    ldr  r2, [r1]
    and  r2, 0b11111111111111110000001111111111
    orr  r2, 0b00000000000000000101010000000000
    str  r2, [r1]

    add  r1, 0x4 @ GPIOA_OTYPER
    ldr  r2, [r1]
    and  r2, 0b11111111111111111111111100011111
    str  r2, [r1]

    add  r1, 0x4 @ GPIOA_SPEEDER
    ldr  r2, [r1]
    and  r2, 0b11111111111111110000001111111111
    orr  r2, 0b00000000000000000101010000000000
    str  r2, [r1]

    ldr  r1, =GPIOB_BASE @ GPIOB_MODER
    ldr  r2, [r1]
    and  r2, 0b11111111111111100111111111111111
    orr  r2, 0b00000000000000001000000000000000
    str  r2, [r1]

    add  r1, 0x8 @ GPIOB_SPEEDER
    ldr  r2, [r1]
    and  r2, 0b11111111111111100111111111111111
    orr  r2, 0b00000000000000001000000000000000
    str  r2, [r1]

    ldr  r1, =GPIOC_BASE @ GPIOC_MODER
    ldr  r2, [r1]
    and  r2, 0b11110011111111111111111111111111
    str  r2, [r1]

    pop  {r0, r1, r2, pc}

max7219_init:
    push {r0, r1, lr}

    ldr  r0, =DECODE_MODE
    ldr  r1, =0xFF
    bl   max7219_send

    ldr  r0, =DISPLAY_TEST
    ldr  r1, =0x0
    bl   max7219_send

    ldr  r0, =SCAN_LIMIT
    ldr  r1, =0x7
    bl   max7219_send

    ldr  r0, =INTENSITY
    ldr  r1, =0xA
    bl   max7219_send

    ldr  r0, =SHUTDOWN
    ldr  r1, =0x1
    bl   max7219_send

    pop  {r0, r1, pc}

max7219_send:
    @ input parameter: r0 is ADDRESS , r1 is DATA
```

```
    push {r0, r1, r2, r3, r4, r5, r6, r7, r8, lr}
    lsl  r0, r0, 0x8
    add  r0, r1
    ldr  r1, =GPIOA_BASE
    ldr  r2, =MAX7219_CS
    ldr  r3, =MAX7219_DIN
    ldr  r4, =MAX7219_CLK
    ldr  r5, =BSRR_OFFSET
    ldr  r6, =BRR_OFFSET
    ldr  r7, =0x0F @ currently sending r7 -th bit

max7219_send_loop:
    mov  r8, #0x1
    lsl  r8, r8, r7
    str  r4, [r1, r6] @ clk -> 0
    tst  r0, r8 @ ANDS but discard result
    beq  max7219_send_clear_bit
    str  r3, [r1, r5] @ din -> 1
    b    max7219_send_check_done

max7219_send_clear_bit:
    str  r3, [r1, r6] @ din -> 0

max7219_send_check_done:
    str  r4, [r1, r5] @ clk -> 1
    subs r7, #0x1
    bge  max7219_send_loop
    str  r2, [r1, r6] @ cs -> 0
    str  r2, [r1, r5] @ cs -> 1
    pop  {r0, r1, r2, r3, r4, r5, r6, r7, r8, pc}

delay_1s:
    push {r0, lr}
    ldr  r0, =4000000 @ delay 1s
    movs r0, r0
    b    delay_1s_loop

delay_1s_loop:
    beq  delay_1s_end
    subs r0, 0x4
    b    delay_1s_loop

delay_1s_end:
    pop  {r0, pc}

fpu_enable:
    push  {r0, r1, lr}
    ldr.w r0, =0xE000ED88
    ldr   r1, [r0]
    orr   r1, r1, #(0xF << 20)
    str   r1, [r0]
    dsb
    isb
    pop   {r0, r1, pc}
```

實驗結果與問題回答

Modify system initial clock

- 根據文件的指示改變系統 clock，觀察 LED 燈閃爍速率的變化。

計時器

- 根據 ARR 和 prescaler 的 CNT，每 0.01 秒加 1，顯示在七段顯示器上。
- 七段顯示器的第三位加上小數點。

Music keypad

- 採用課程投影片上的說明，變化 prescaler 來改變頻率。

心得討論與應用聯想

- 改變 duty cycle 會改變音色。
- 時間完全不夠。
- 這門課不但讓我學了微處理機，還讓我學了危機處理，雖然這門課已經改為選修了，不過我一定會推薦學弟妹來修的。