Lab7 STM32 Clock and Timer

0510532 楊上萱

1. Lab objectives 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

2. Steps 實驗步驟

(1) Modify system initial clock

驅動部分

先把 GPIO_init()和 delay_1s 的 function 放入.s 檔中並宣告為.global。

硬體部分

PA5 為 LED output pin，PC13 為 user button input pin。

設定 system clock:

關掉 PLL 並等它穩定。

改變 PLLN、PLLM 來設定 RCC->PLLCFGR。

重新開啟 PLL 並等它穩定並把 PLL 作為 SYSTEM CLK。

最後改變 RCC->CFGR 來設定 AHB prescaler。

如何更改頻率:

照著公式推算:

The PLL clock frequency is calculated as f(VCO clock) = f(PLL clock input) × (PLLN / PLLM)

The final output to the system clock frequency is f(PLL_R) = f(VCO clock) / PLLR

(2) Timer 計時器

驅動部分:

將 max7219_init()、GPIO_init()、max7219send 的 function 放入.s 檔中並宣告為.global。

硬體部分:

DIN->PA5，CS->PA6，CLK->PA7，VCC->3v3，GND->GND

程式解釋:

MSI 的 default frequency 是 4MHz, TIM2 的 prescaler 為 39999，auto-reload register 是 99。

這樣子設定的結果，timer frequency 變為 100Hz，counter 每 0.01 秒加一次，然後把 counter/100 的結果印在 max7219 上，一直數到 TIME_SEC*100 為止。

(3) Music keypad

蜂鳴器頻率：

Freq = HCLK / (prescalar + 1) / (AutoReload_reg + 1);

➩ Prescalar = HCLK / Freq / (AutoReload_reg + 1)-1;

PWM：

CCR_reg 初始為 dyty_cycle=50，我們用

改變 CCR_reg(duty_cycle)的值，每按一次”#”，duty_cycle+1，每按一次”*”，duty_cycle-1。

3. Results and analysis 實驗結果與分析

7_1:

每次按 user_button，LED 閃爍的頻率會依序變成 1MHz – 6MHz – 10MHz – 16MHz – 40MHz – 1MHz …。

7_2:

設定 TIME_SEC，程式會一邊數秒並秀在 max7219 上，數到 TIME_SEC 為止。

7_3:

用 keypad 控制蜂鳴器的輸出頻率，keypad 上的 1~8 分別代表標準 Do、Re、Mi、Fa、…、高音 Do，按#會把 duty_cycle+5，按*會把 duty_cycle-5。

4. Conclusions and ideas 心得討論與應用聯想

這次 Lab 比之前都難，因此也花了更多時間在做，不過過程中學到了許多，包括如何對 system clock、timer 做設定，還有用 timer 設定蜂鳴器的頻率也非常有趣。

5. Code

main7_1.c

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

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

void SystemClock_Config();

```

```
int main()
{
    SystemClock_Config();
    gpio_init();
    int flag=0;
    int j=1;
    while (1)
    {
        //if (!prev_btn && curr_btn)
        j=1;
        if(flag)
        {
            switch (state)
            {
            case S1MHZ:
                plln = 16; // 16 7 9
                pllm = 7;
                prescaler = 9;
                j=1;
                break;
            case S6MHz:
                plln = 24;//24 7 0
                pllm = 7;
                prescaler = 0;
                j=1000;
                break;
            case S10MHZ:
                plln = 40;//40 7 0
                pllm = 7;
                prescaler = 0;
                j=1400;
                break;
            case S16MHZ:
                plln = 64;// 64 7 0
                pllm = 7;
                prescaler = 0;
                j=1800;
                break;
            case S40MHZ:
                plln = 20;// 20 0 0
                pllm = 0;
                prescaler = 0;
                j=3000;
                break;
            default:
                break;
            }
            SystemClock_Config();
            state = state == S40MHZ ? S1MHZ : state + 1;
        }
        GPIOA->BSRR = (1 << 5);
        //prev_btn = curr_btn;
        flag=0;
        //int test = GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_13);
        for(int i=0;i<1000;i++){
//          if(flag==0)
//              curr_btn = GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_13);
            if(!GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_13)&&flag==0){
                for(int k=1;k<j;k++)
                        delay_1s();
                flag=1;
            }
            delay_1s();
        }
        GPIOA->BRR = (1 << 5);
        for(int i=0;i<1000;i++){
//          if(flag ==0)
//              curr_btn = GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_13);
            if(!GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_13)&&flag==0){
                for(int k=1;k<j;k++)
                        delay_1s();
                flag=1;
            }
            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
}

```

main7_2.c

```
#include "stm32l476xx.h"
#include "util.h"
#define TIME_SEC 12.7

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);
```

```c
    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();
}

```

main7_3.c

```c
#include "stm32l476xx.h"
#include "keypad.h"
#include "7-seg.h"

int intlen (int n);
void timer_init (TIM_TypeDef *timer);
void timer_start (TIM_TypeDef *timer);
void timer_stop (TIM_TypeDef *timer);
void C4 (TIM_TypeDef *timer);
void D4 (TIM_TypeDef *timer);
void E4 (TIM_TypeDef *timer);
void F4 (TIM_TypeDef *timer);
void G4 (TIM_TypeDef *timer);
void A4 (TIM_TypeDef *timer);
void B4 (TIM_TypeDef *timer);
void C5 (TIM_TypeDef *timer);

void timer_init (TIM_TypeDef *timer)
{
    // Sound freq = 4 MHz / (pres + 1) / 100
    // pres = 4 MHz / Sound freq / 100 - 1
    timer->PSC = (uint32_t) 152;
    timer->ARR = (uint32_t) 99;

    /* CH1 */
    timer->CCR1 = 50;
    timer->CCMR1 |= TIM_CCMR1_OC1M_2 | TIM_CCMR1_OC1M_1;

    timer->CR1 |= TIM_CR1_ARPE;
    timer->EGR = TIM_EGR_UG;
    timer->CCER |= TIM_CCER_CC1E;   /* CH1 */
}

void timer_start (TIM_TypeDef *timer)
{
    timer->CR1 |= TIM_CR1_CEN;
}

void timer_stop (TIM_TypeDef *timer)
{
    timer->CR1 &= ~TIM_CR1_CEN;
}

int main (void)
{
    int     t = 0, key = 0, duty = 50;
    TIM_TypeDef *timer = TIM3;

    max7219_init ();
    display (duty, intlen (duty));
    keypad_init ();

    /* GPIO: set PB4 as alternate function */
    RCC->AHB2ENR |= 0x1 << 1;   /* enable AHB2 clock for port B */
    GPIOB->MODER |= GPIO_MODER_MODE4_1;
    GPIOB->AFR[0] |= GPIO_AFRL_AFSEL4_1;    /* PB4: AF2 (TIM3_CH1) */

    RCC->APB1ENR1 |= RCC_APB1ENR1_TIM3EN;
    timer_init (timer);

    while (1) {
        if (!row4 () && !row3 () && !row2 () && !row1 ()) {
            t = 0;
            key = 0;
            timer_stop (timer);
        } else {
            if (key == 0)
                key = keypad_scan ();
            if (t == 700) {
                if (key & 0x1 << 1) {
                    C4 (timer);
                    timer_start (timer);
                } else if (key & 0x1 << 2) {
                    D4 (timer);
                    timer_start (timer);
                } else if (key & 0x1 << 3) {
                    E4 (timer);
                    timer_start (timer);
                } else if (key & 0x1 << 4) {
                    F4 (timer);
                    timer_start (timer);
                } else if (key & 0x1 << 5) {
                    G4 (timer);
                    timer_start (timer);
                } else if (key & 0x1 << 6) {
                    A4 (timer);
                    timer_start (timer);
                } else if (key & 0x1 << 7) {
                    B4 (timer);
                    timer_start (timer);
                } else if (key & 0x1 << 8) {
                    C5 (timer);
                    timer_start (timer);
                } else if (key & 0x1 << 14) {
                    duty += 500;
                    if (duty > 100000)
                        duty = 90;
                    display (duty, intlen (duty));
                    timer->CCR1 = duty;
                  } else if (key & 0x1 << 15) {
                      duty -= 5;
                      if (duty < 10)
                          duty = 10;
                      display (duty, intlen (duty));
                      timer->CCR1 = duty;
                  }
              }
              allhigh ();
              ++t;
          }
```

```c
    }

    return 0;
}

void C4 (TIM_TypeDef *timer)
{
    timer->PSC = (uint32_t) 152;    // 4 MHz / 261.6 Hz / 100 - 1 = 151.90 = 152;
}

void D4 (TIM_TypeDef *timer)
{
    timer->PSC = (uint32_t) 135;    // 4 MHz / 293.7 Hz / 100 - 1 = 135.19 = 135
}

void E4 (TIM_TypeDef *timer)
{
    timer->PSC = (uint32_t) 120;    // 4 MHz / 329.6 Hz / 100 - 1 = 120.36 = 120
}

void F4 (TIM_TypeDef *timer)
{
    timer->PSC = (uint32_t) 114;    // 4 MHz / 349.2 Hz / 100 - 1 = 113.55 = 114
}

void G4 (TIM_TypeDef *timer)
{
    timer->PSC = (uint32_t) 101;    // 4 MHz / 392.0 Hz / 100 - 1 = 101.04 = 101
}

void A4 (TIM_TypeDef *timer)
{
    timer->PSC = (uint32_t) 90; // 4 MHz / 440.0 Hz / 100 - 1 = 89.91 = 90
}

void B4 (TIM_TypeDef *timer)
{
    timer->PSC = (uint32_t) 80; // 4 MHz / 493.9 Hz / 100 - 1 = 79.99 = 80
}

void C5 (TIM_TypeDef *timer)
```

```c
{
    timer->PSC = (uint32_t) 75; // 4 MHz / 523.3 Hz / 100 - 1 = 75.44 = 75
}

int intlen (int n)
{
    int len = 1;
    while (n > 9) {
        n /= 10;
        ++len;
    }
    return len;
}

```