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

```
1.  #include "stm32l476xx.h"
2.  #include "util.h"
3.
4.  int plln = 16, pllm = 7, prescaler = 9;
5.  enum {S1MHZ, S6MHz, S10MHZ, S16MHZ, S40MHZ} state = S40MHZ;
6.  int prev_btn = 1, curr_btn = 1;
7.
8.  void SystemClock_Config();
9.
```

```
10. int main()
11. {
12.     SystemClock_Config();
13.     gpio_init();
14.     int flag=0;
15.     int j=1;
16.     while (1)
17.     {
18.         //if (!prev_btn && curr_btn)
19.         j=1;
20.         if(flag)
21.         {
22.             switch (state)
23.             {
24.             case S1MHZ:
25.                 plln = 16; // 16 7 9
26.                 pllm = 7;
27.                 prescaler = 9;
28.                 j=1;
29.                 break;
30.             case S6MHz:
31.                 plln = 24;//24 7 0
32.                 pllm = 7;
33.                 prescaler = 0;
34.                 j=1000;
35.                 break;
36.             case S10MHZ:
37.                 plln = 40;//40 7 0
38.                 pllm = 7;
39.                 prescaler = 0;
40.                 j=1400;
41.                 break;
42.             case S16MHZ:
43.                 plln = 64;// 64 7 0
44.                 pllm = 7;
45.                 prescaler = 0;
46.                 j=1800;
47.                 break;
48.             case S40MHZ:
49.                 plln = 20;// 20 0 0
50.                 pllm = 0;
51.                 prescaler = 0;
52.                 j=3000;
53.                 break;
54.             default:
55.                 break;
56.             }
57.             SystemClock_Config();
```

```
58.             state = state == S40MHZ ? S1MHZ : state + 1;
59.         }
60.         GPIOA->BSRR = (1 << 5);
61.         //prev_btn = curr_btn;
62.         flag=0;
63.         //int test = GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_13);
64.         for(int i=0;i<1000;i++){
65. //          if(flag==0)
66. //              curr_btn = GPIO_ReadInputDataBit(GPIOC,
    GPIO_Pin_13);
67.             if(!GPIO_ReadInputDataBit(GPIOC,
    GPIO_Pin_13)&&flag==0){
68.                 for(int k=1;k<j;k++)
69.                         delay_1s();
70.                 flag=1;
71.             }
72.             delay_1s();
73.         }
74.         GPIOA->BRR = (1 << 5);
75.         for(int i=0;i<1000;i++){
76. //          if(flag ==0)
77. //              curr_btn = GPIO_ReadInputDataBit(GPIOC,
    GPIO_Pin_13);
78.             if(!GPIO_ReadInputDataBit(GPIOC,
    GPIO_Pin_13)&&flag==0){
79.                 for(int k=1;k<j;k++)
80.                         delay_1s();
81.                 flag=1;
82.             }
83.             delay_1s();
84.         }
85.
86.         //prev_btn = curr_btn;
87.         //curr_btn = GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_13);
88.     }
89. }
90.
91. void SystemClock_Config()
92. {
93.     RCC->CFGR = 0x00000000;
94.     // CFGR reset value
95.     RCC->CR &= 0xFEFFFFFF;
96.     // main PLL enable: PLL off
97.     while (RCC->CR & 0x02000000);
98.     // main PLL clock ready flag: PLL locked
99.     RCC->PLLCFGR = 0x01000001;
100.       // main PLL PLLCLK output enable: PLLCLK output enable
101.       // main PLL entry clock source: MSI clock selected as PLL
```

```
    clock entry
102.        RCC->PLLCFGR |= plln << 8;
103.        // main PLL multiplication factor for VCO
104.        RCC->PLLCFGR |= pllm << 4;
105.        // division factor for the main PLL input clock
106.        // f(VCO clock) = f(PLL clock input) × (PLLN / PLLM)
107.        // f(PLL_R) = f(VCO clock) / PLLR
108.        RCC->CR |= 0x01000000;
109.        // main PLL enable: PLL on
110.        while (!(RCC->CR & 0x02000000));
111.        // main PLL clock ready flag: PLL locked
112.        RCC->CFGR = 0x00000003;
113.        // system clock switch: PLL selected as system clock
114.        RCC->CFGR |= prescaler << 4;
115.        // AHB prescaler: SYSCLK divided by N
116.    }
117.
```

main7_2.c

```
1. #include "stm32l476xx.h"
2. #include "util.h"
3. #define TIME_SEC 12.7
4.
5. int cal_len(int a)
6. {
7.     int sum = 0;
8.     while (a > 0)
9.     {
10.        a /= 10;
11.        sum++;
12.    }
13.    return sum;
14.}
15.
16.void timer_init()
17.{
18.    RCC->APB1ENR1 |= 0b1;
19.    TIM2->ARR = (uint32_t) (TIME_SEC * (4000000 / 40000)); //
   reload value
20.    TIM2->PSC = (uint32_t) 39999; // prescaler
21.    TIM2->EGR = TIM_EGR_UG; // reinitialize the counter
22.}
23.
24.void timer_start()
25.{
26.    TIM2->CR1 |= TIM_CR1_CEN;
27.    display(0, -1003);
```

```
28.     if (TIME_SEC <= 0 || TIME_SEC > 10000)
29.     {
30.         TIM2->CR1 &= ~TIM_CR1_CEN;
31.         return;
32.     }
33.     int pre_val = 0;
34.     while (1)
35.     {
36.         int now_val = TIM2->CNT;
37.         if (pre_val > now_val)
38.         {
39.             TIM2->CR1 &= ~TIM_CR1_CEN;
40.             return;
41.         }
42.         pre_val = now_val;
43.         int len = cal_len(now_val);
44.         if (now_val < 100)
45.             len = 3;
46.         display(now_val, -1000 - len);
47.     }
48.}
49.
50.int main()
51.{
52.     gpio_init();
53.     max7219_init();
54.     timer_init();
55.     timer_start();
56.}
57.
```

main7_3.c

```
1. #include "stm32l476xx.h"
2. #include "keypad.h"
3. #include "7-seg.h"
4.
5. int intlen (int n);
6. void timer_init (TIM_TypeDef *timer);
7. void timer_start (TIM_TypeDef *timer);
8. void timer_stop (TIM_TypeDef *timer);
9. void C4 (TIM_TypeDef *timer);
10.void D4 (TIM_TypeDef *timer);
11.void E4 (TIM_TypeDef *timer);
12.void F4 (TIM_TypeDef *timer);
13.void G4 (TIM_TypeDef *timer);
14.void A4 (TIM_TypeDef *timer);
15.void B4 (TIM_TypeDef *timer);
```

```
16.void C5 (TIM_TypeDef *timer);
17.
18.void timer_init (TIM_TypeDef *timer)
19.{
20.    // Sound freq = 4 MHz / (pres + 1) / 100
21.    // pres = 4 MHz / Sound freq / 100 - 1
22.    timer->PSC = (uint32_t) 152;
23.    timer->ARR = (uint32_t) 99;
24.
25.    /* CH1 */
26.    timer->CCR1 = 50;
27.    timer->CCMR1 |= TIM_CCMR1_OC1M_2 | TIM_CCMR1_OC1M_1;
28.
29.    timer->CR1 |= TIM_CR1_ARPE;
30.    timer->EGR = TIM_EGR_UG;
31.    timer->CCER |= TIM_CCER_CC1E;   /* CH1 */
32.}
33.
34.void timer_start (TIM_TypeDef *timer)
35.{
36.    timer->CR1 |= TIM_CR1_CEN;
37.}
38.
39.void timer_stop (TIM_TypeDef *timer)
40.{
41.    timer->CR1 &= ~TIM_CR1_CEN;
42.}
43.
44.int main (void)
45.{
46.    int     t = 0, key = 0, duty = 50;
47.    TIM_TypeDef *timer = TIM3;
48.
49.    max7219_init ();
50.    display (duty, intlen (duty));
51.    keypad_init ();
52.
53.    /* GPIO: set PB4 as alternate function */
54.    RCC->AHB2ENR |= 0x1 << 1;   /* enable AHB2 clock for port B */
55.    GPIOB->MODER |= GPIO_MODER_MODE4_1;
56.    GPIOB->AFR[0] |= GPIO_AFRL_AFSEL4_1;    /* PB4: AF2 (TIM3_CH1)
   */
57.
58.    RCC->APB1ENR1 |= RCC_APB1ENR1_TIM3EN;
59.    timer_init (timer);
60.
61.    while (1) {
62.        if (!row4 () && !row3 () && !row2 () && !row1 ()) {
```

```
63.             t = 0;
64.             key = 0;
65.             timer_stop (timer);
66.         } else {
67.             if (key == 0)
68.                 key = keypad_scan ();
69.             if (t == 700) {
70.                 if (key & 0x1 << 1) {
71.                     C4 (timer);
72.                     timer_start (timer);
73.                 } else if (key & 0x1 << 2) {
74.                     D4 (timer);
75.                     timer_start (timer);
76.                 } else if (key & 0x1 << 3) {
77.                     E4 (timer);
78.                     timer_start (timer);
79.                 } else if (key & 0x1 << 4) {
80.                     F4 (timer);
81.                     timer_start (timer);
82.                 } else if (key & 0x1 << 5) {
83.                     G4 (timer);
84.                     timer_start (timer);
85.                 } else if (key & 0x1 << 6) {
86.                     A4 (timer);
87.                     timer_start (timer);
88.                 } else if (key & 0x1 << 7) {
89.                     B4 (timer);
90.                     timer_start (timer);
91.                 } else if (key & 0x1 << 8) {
92.                     C5 (timer);
93.                     timer_start (timer);
94.                 } else if (key & 0x1 << 14) {
95.                     duty += 500;
96.                     if (duty > 100000)
97.                         duty = 90;
98.                     display (duty, intlen (duty));
99.                     timer->CCR1 = duty;
100.                } else if (key & 0x1 << 15) {
101.                    duty -= 5;
102.                    if (duty < 10)
103.                        duty = 10;
104.                    display (duty, intlen (duty));
105.                    timer->CCR1 = duty;
106.                }
107.            }
108.            allhigh ();
109.            ++t;
110.        }
```

```
111.        }
112.
113.        return 0;
114.    }
115.
116.    void C4 (TIM_TypeDef *timer)
117.    {
118.        timer->PSC = (uint32_t) 152;    // 4 MHz / 261.6 Hz / 100 -
    1 = 151.90 = 152;
119.    }
120.
121.    void D4 (TIM_TypeDef *timer)
122.    {
123.        timer->PSC = (uint32_t) 135;    // 4 MHz / 293.7 Hz / 100 -
    1 = 135.19 = 135
124.    }
125.
126.    void E4 (TIM_TypeDef *timer)
127.    {
128.        timer->PSC = (uint32_t) 120;    // 4 MHz / 329.6 Hz / 100 -
    1 = 120.36 = 120
129.    }
130.
131.    void F4 (TIM_TypeDef *timer)
132.    {
133.        timer->PSC = (uint32_t) 114;    // 4 MHz / 349.2 Hz / 100 -
    1 = 113.55 = 114
134.    }
135.
136.    void G4 (TIM_TypeDef *timer)
137.    {
138.        timer->PSC = (uint32_t) 101;    // 4 MHz / 392.0 Hz / 100 -
    1 = 101.04 = 101
139.    }
140.
141.    void A4 (TIM_TypeDef *timer)
142.    {
143.        timer->PSC = (uint32_t) 90; // 4 MHz / 440.0 Hz / 100 - 1 =
    89.91 = 90
144.    }
145.
146.    void B4 (TIM_TypeDef *timer)
147.    {
148.        timer->PSC = (uint32_t) 80; // 4 MHz / 493.9 Hz / 100 - 1 =
    79.99 = 80
149.    }
150.
151.    void C5 (TIM_TypeDef *timer)
```

```
152.  {
153.      timer->PSC = (uint32_t) 75; // 4 MHz / 523.3 Hz / 100 - 1 =
   75.44 = 75
154.  }
155.
156.  int intlen (int n)
157.  {
158.      int len = 1;
159.      while (n > 9) {
160.          n /= 10;
161.          ++len;
162.      }
163.      return len;
164.  }
165.
```