0316323 薛世恩　0316213 蒲郁文

1. 實驗目的

瞭解 STM32 SysTick timer 設定
瞭解 STM32 NVIC 和 External interrupt 設定和原理

使用到：蜂鳴器 keypad led 燈 七段顯示器

2.實驗步驟

(1) SysTick timer interrupt

```
#include "stm32l476xx.h"
#include "utils.h"
int plln = 40, pllm = 7, prescaler = 0; // 10 MHz SYSCLK
int prev_btn = 1, curr_btn = 1;
void SystemClock_Config();
void SysTick_UserConfig();
void SysTick_Handler();
int main()
{
SystemClock_Config();
SysTick_UserConfig();
gpio_init();
while (1)
{
if (!prev_btn && curr_btn)
SysTick->CTRL = (SysTick->CTRL & 0xFFFFFFFE) | ~(SysTick->CTRL & 0x00000001);
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
}
void SysTick_UserConfig()
{
SysTick->CTRL |= 0x00000004;
SysTick->LOAD = 5000000; // 0.5 second
SysTick->VAL = 0;
SysTick->CTRL |= 0x00000003;
}
void SysTick_Handler()
{
GPIOA->ODR = (GPIOA->ODR & 0xFFFFFFDF) | ~(GPIOA->ODR & 0x00000020);
}
```

作法: 設定時如同上面註解，當 user button 按鍵被按下時，將 systick 的 ctrl 最後一個 bit 做切換，以達到 enable 跟 disable 的效果，也要透過 plln，pllm 以及 pllr 來調整 clock。透過 SysTick_UserConfig() 來設定時間，在 SysTick_Handler() 時將其輸出。

(2) Keypad external interrupt

```
#include "stm32l476xx.h"
#include "utils.h"
int scan_state = 0, key_value = 0, prev = 0;
void SysTick_UserConfig();
void SysTick_Handler();
void EXTI4_IRQHandler();
void EXTI9_5_IRQHandler();

int main()
{
SysTick_UserConfig();
gpio_init();
max7219_init();
keypad_init();
exti_init();
while (1)
{
display(key_value, 2);
prev = YPORT->IDR;
}
}
void SysTick_UserConfig()
{
SysTick->CTRL |= 0x00000004;
SysTick->LOAD = 400000; // 0.1 second
SysTick->VAL = 0;
SysTick->CTRL |= 0x00000003;
}
void SysTick_Handler()
{
EXTI->IMR1 ^= EXTI_IMR1_IM4 | EXTI_IMR1_IM5 | EXTI_IMR1_IM6 | EXTI_IMR1_IM7;
scan_state = scan_state == 3 ? 0 : scan_state + 1;
switch (scan_state)
{
case 0:
XPORT->BSRR = X0;
XPORT->BRR = X1;
XPORT->BRR = X2;
XPORT->BRR = X3;
break;
case 1:
XPORT->BRR = X0;
XPORT->BSRR = X1;
XPORT->BRR = X2;
XPORT->BRR = X3;
break;
case 2:
XPORT->BRR = X0;
XPORT->BRR = X1;
XPORT->BSRR = X2;
XPORT->BRR = X3;
break;
case 3:
XPORT->BRR = X0;
XPORT->BRR = X1;
XPORT->BRR = X2;
XPORT->BSRR = X3;
break;
}
EXTI->IMR1 |= EXTI_IMR1_IM4 | EXTI_IMR1_IM5 | EXTI_IMR1_IM6 | EXTI_IMR1_IM7;
}
void EXTI4_IRQHandler()
```

```
{
uint32_t *ptr;
ptr = (uint32_t *) NVIC_ICPR;
ptr[0] = 0x00000400;
EXTI->PR1 |= EXTI_PR1_PIF4;
int now = YPORT->IDR;
switch (scan_state)
{
case 0:
if ((prev & Y0) && !(now & Y0))
key_value = 15;
if ((prev & Y1) && !(now & Y1))
key_value = 7;
if ((prev & Y2) && !(now & Y2))
key_value = 4;
if ((prev & Y3) && !(now & Y3))
key_value = 1;
break;
case 1:
if ((prev & Y0) && !(now & Y0))
key_value = 0;
if ((prev & Y1) && !(now & Y1))
key_value = 8;
if ((prev & Y2) && !(now & Y2))
key_value = 5;
if ((prev & Y3) && !(now & Y3))
key_value = 2;
break;
case 2:
if ((prev & Y0) && !(now & Y0))
key_value = 14;
if ((prev & Y1) && !(now & Y1))
key_value = 9;
if ((prev & Y2) && !(now & Y2))
key_value = 6;
if ((prev & Y3) && !(now & Y3))
key_value = 3;
break;
case 3:
if ((prev & Y0) && !(now & Y0))
key_value = 13;
if ((prev & Y1) && !(now & Y1))
key_value = 12;
if ((prev & Y2) && !(now & Y2))
key_value = 11;
if ((prev & Y3) && !(now & Y3))
key_value = 10;
break;
```

```
}
}
void EXTI9_5_IRQHandler()
{
uint32_t *ptr;
ptr = (uint32_t *) NVIC_ICPR;
ptr[0] = 0x00800000;
EXTI->PR1 |= EXTI_PR1_PIF5 | EXTI_PR1_PIF6 | EXTI_PR1_PIF7;
int now = YPORT->IDR;
switch (scan_state)
{
case 0:
if ((prev & Y0) && !(now & Y0))
key_value = 15;
if ((prev & Y1) && !(now & Y1))
key_value = 7;
if ((prev & Y2) && !(now & Y2))
key_value = 4;
if ((prev & Y3) && !(now & Y3))
key_value = 1;
break;
case 1:
if ((prev & Y0) && !(now & Y0))
key_value = 0;
if ((prev & Y1) && !(now & Y1))
key_value = 8;
if ((prev & Y2) && !(now & Y2))
key_value = 5;
if ((prev & Y3) && !(now & Y3))
key_value = 2;
break;
case 2:
if ((prev & Y0) && !(now & Y0))
key_value = 14;
if ((prev & Y1) && !(now & Y1))
key_value = 9;
if ((prev & Y2) && !(now & Y2))
key_value = 6;
if ((prev & Y3) && !(now & Y3))
key_value = 3;
break;
case 3:
if ((prev & Y0) && !(now & Y0))
key_value = 13;
if ((prev & Y1) && !(now & Y1))
key_value = 12;
if ((prev & Y2) && !(now & Y2))
key_value = 11;
```

```
if ((prev & Y3) && !(now & Y3))
key_value = 10;
break;
}
}
```

作法: 首先將所有要用到的東西先做初始化,因為此次實驗要做 negative edge 所以要先將其設定好,再來將將 SysTick_UserConfig()設定好為 0.1 秒,再來在 Systick_Handler 的地方做鍵盤的掃描 imr 也要同時做設定,一次掃一條 column,我們以 scan_state 來實做,讓它在 0 到 3 之間一直切換,再透過 IRQHandler 判斷到底是哪個按鍵被按下去,最後我們透過 interrupt 然後可以顯示出值在七段顯示器上。

(3) 製作簡單鬧鐘

```
#include "stm32l476xx.h"
#include "utils.h"
#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
#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;
unsigned int x_pin = {X0, X1, X2, X3};
unsigned int y_pin = {Y0, Y1, Y2, Y3};
unsigned int total, len;
char set[14];
```

```
int rem = 0;
int plln = 40, pllm = 7, prescaler = 0; // 10 MHz SYSCLK
int prev_btn = 1, curr_btn = 1;
int state = 0;
void timer_init()
{
RCC->APB1ENR1 |= RCC_APB1ENR1_TIM2EN;
// enable TIM2 timer clock
GPIOB->AFR[0] |= GPIO_AFRL_AFSEL3_0;
// select AF1 for PB3 (PB3 is TIM2_CH2)
TIM2->CR1 |= TIM_CR1_DIR;
// counter used as downcounter
TIM2->CR1 |= TIM_CR1_ARPE;
// enable auto-reload preload (buffer TIM2_ARR)
TIM2->ARR = (uint32_t) 100;
// auto-reload prescaler value
TIM2->CCMR1 &= 0xFFFFFCFF;
// select compare 2 (channel 2 is configured as output)
TIM2->CCMR1 |= (TIM_CCMR1_OC2M_2 | TIM_CCMR1_OC2M_1);
// set output compare 2 mode to PWM mode 1
TIM2->CCMR1 |= TIM_CCMR1_OC2PE;
// enable output compare 2 preload register on TIM2_CCR2
TIM2->CCER |= TIM_CCER_CC2E;
// enable compare 2 output
TIM2->EGR = TIM_EGR_UG;
// re-initialize the counter and generates an update of the registers
}
void timer_config()
{
TIM2->PSC = (uint32_t) (4000000 / freq / 100);
// prescaler value
TIM2->CCR2 = duty_cycle;
// compare 2 preload value
}
void start_systick_timer()
{
SysTick->CTRL |= 0x00000001;
state = 1;
}
void stop_systick_timer()
{
SysTick->CTRL &= 0xFFFFFFFE;
state = 0;
}
#define SHPR3 0xE000ED20
void SysTick_UserConfig()
{
```

```
SysTick->CTRL |= 0x00000004;
SysTick->LOAD = 10000000; // 1.0 second
SysTick->VAL = 0;
SysTick->CTRL |= 0x00000002;
uint32_t *ptr;
ptr = (uint32_t *) SHPR3;
*ptr = *ptr | 0xF0000000;
}
void SysTick_Handler()
{
total = total == 0 ? total : total - 1;
int temp = total == 0 ? 1 : cal_len(total);
display(total, temp);
if (total == 0)
{
freq = DO;
timer_config();
TIM2->CR1 |= TIM_CR1_CEN;
while (state == 1);
TIM2->CR1 &= ~TIM_CR1_CEN;
freq = -1;
total = 0;
len = 0;
rem = 0;
}
}
void set_clear()
{
for (int i = 0; i < 14; i++)
set[i] = 0;
}
void set_insert(int i)
{
if (i >= 0 && i < 14)
set[i] = 1;
}
int set_reduce()
{
int sum = 0;
for (int i = 0; i < 14; i++)
if (set[i])
sum += i;
return sum;
}
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) ¡Ñ (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
}
void exti_init2()
{
// setup SYSCFG
SYSCFG->EXTICR[3] = SYSCFG_EXTICR4_EXTI13_PC;
// setup EXTI
EXTI->IMR1 |= EXTI_IMR1_IM13;
EXTI->FTSR1 |= EXTI_FTSR1_FT13;
EXTI->PR1 |= EXTI_PR1_PIF13;
// enable interrupts
asm("cpsie i;");
// setup NVIC
// EXTI15_10_IRQn = 40
uint32_t *ptr;
ptr = (uint32_t *) NVIC_IPR;
ptr[10] = 0x00000010;
ptr = (uint32_t *) NVIC_ICPR;
ptr[1] = 0x00000100;
ptr = (uint32_t *) NVIC_ISER;
ptr[1] = 0x00000100;
}
void EXTI15_10_IRQHandler()
{
uint32_t *ptr;
ptr = (uint32_t *) NVIC_ICPR;
```

```
ptr[1] = 0x00000100;
EXTI->PR1 |= EXTI_PR1_PIF13;
while (1)
{
prev_btn = curr_btn;
curr_btn = GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_13);
if (state == 0 && !prev_btn && curr_btn)
{
if (total == 0)
break;
start_systick_timer();
break;
}
else if (state == 1 && !prev_btn && curr_btn)
{
stop_systick_timer();
break;
}
}
}
int main()
{
fpu_enable();
SystemClock_Config();
SysTick_UserConfig();
gpio_init();
max7219_init();
keypad_init();
timer_init();
exti_init2();
TIM2->CR1 &= ~TIM_CR1_CEN;
freq = -1;
total = 0;
len = 0;
int cnt = 0;
int input = -1, prev_input = -1;
set_clear();
while (1)
{
if (state == 0)
{
prev_input = input;
input = keypad_scan();
if (input == 0)
rem = 1;
if (input == prev_input)
{
cnt++;
if (cnt > 12000)
{
cnt = 0;
if (input < 14)
goto A;
}
}
else if (input >= 14)
{
total = 0;
len = 0;
set_clear();
display(total, len);
}
else if (input != -1)
set_insert(input);
else
{
input = set_reduce();
A:
set_clear();
if (input >= 10 && len + 2 <= 8)
{
total = total * 100 + input;
len += 2;
}
else if (input < 10 && input >= 0 && len + 1 <= 8)
{
if (input == 0 && rem == 0);
else
{
total = total * 10 + input;
if (total == 0 && rem == 1);
else
len += 1;
rem = 0;
}
}
display(total, len);
}
}
}
}
```

作法: 先將前幾次的 code 拿出可以用的部份,分別是計算機那次有一個是可以輸入數字在七段顯示器上面,然後再來是判斷 user button 是否被按下,如果被按下則 interrupt,之後要重新寫 SysTick 的碼表倒數並且也要設定頻率為 10M,然後當他到數完後要讓蜂鳴器以任意頻率（自訂）響起,因為我們有用 state 來區別,所以當 user button 被按下開始倒數和蜂鳴器響起時,keyboard 都不會讀到任何東西。最後在按一次按鍵 interrupt 才打斷蜂鳴器並回到等待輸入狀態。有一像要求是輸入零沒有反應,只要判斷 total==0 然後又是輸入 0 時就不要理它就好。總之這一題的各項設定都是從前面的 lab 所抓出來的,所以前面只要有一次 lab 不會,這一題一定寫不出來,最重要的是設定 priority。

心得:
這次作業真的很困難,有太多東西要調整,這張板子的 reference 又寫得很亂,可讀性真的不是太好,又常常互相參考,一直要點連結跳去不同的地方,如果以後要設計硬體跟寫 reference 要引以為戒,再來是這次作業要求真的都很多,尤其是第三題根本是大雜燴,然後二三題 bug 真的很多,又因為常常剪剪貼貼會漏掉 init 然後整個七段顯示器無法顯示,或是 keyboard 吃不到按鈕之類的,又或者 keyboard 的值會亂跳,讓我們吃盡苦頭。然後 bug 都要用 debugger 一行一行的看。沒有辦法像以前直接看燈或是七段顯示器的輸出來直接 debug,必須仔細看每一個細節是否出錯跟參數的傳遞。不過作業的量實在很多,有點超過負荷。這個禮拜我跟我的 partner 每天都熬夜,三天大概只睡了十初頭個小時。interrupts 是一個很重要部份,但是因為不習慣吧,做起來是處處碰到困難。