微處理機系統實習　　Lab7

班級：資訊三丁　學號：D1009212　姓名：邱柏宇

一、【實驗目的】：

What was your design? What were the concepts you have used for your design?

首先我參考老師的 scankey 中斷和 timer 的寫法，把 scankey 改成了純中斷的版本，當發生中斷的時候，會將該列上的 PA0~2 切換 011, 010, 110, 去進行檢查哪個鍵被按下。如此一來就可以不需要利用 timer 去切換 PA0~2 的 01 組合了，為了讓之後也比較方便可以用，在 Library 建立一個 NewScanKey 來放中斷版本的 scankey。

```
/*
      PA2   PA1   PA0
PA3    1     2     3
PA4    4     5     6
PA5    7     8     9
*/
void GPAB_IRQHandler(void) {
    uint8_t i, which_PA_INT;
    // 查看是哪個 PA 觸發中斷
    if (PA->ISRC & BIT0) which_PA_INT = 0;
    if (PA->ISRC & BIT1) which_PA_INT = 1;
    if (PA->ISRC & BIT2) which_PA_INT = 2;

    // 檢查是哪個按鈕被按下
    // 將所有 PA 腳位設為 1，接下來輪流將PA3~5設為 0，並檢查是否有按鈕被按下。
    PA0 = 1; PA1 = 1; PA2 = 1; PA3 = 1; PA4 = 1; PA5 = 1;
    for (i = 3; i <= 5; i++) {
        CLK_SysTickDelay(5000);
        GPIO_PIN_DATA(0, i - 1) = 1;
        GPIO_PIN_DATA(0, i) = 0;
        if (GPIO_PIN_DATA(0, which_PA_INT) == 0) {
            KEY_FLAG = (3 - which_PA_INT) + 3 * (i - 3);
            break;
        }
    }
    PA0 = 1; PA1 = 1; PA2 = 1; PA3 = 0; PA4 = 0; PA5 = 0;
    PA->ISRC = PA->ISRC;
    return;
}
```

7.1

為了隨機取兩位數，首先我將(rand() % 9 + 1) * 10 取到 1~9 的十位數，然後再 rand() % 10 取個位數加在一起。

計算符號用 #define MATH_SYMBOLS "+-*/" 來定義，方便用來 print 在 lcd 上。

並且為了達到不影響七段顯示器，也是都改用 timer 去跑。我覺得這個 timer 版本的七段顯示器之後也會很常用到，所以也在 Library 建立一個 NewSevenSegment 去放這個 timer，那我還寫了個轉換再 NewSevenSegment 中，因為如果可以直接一次 show 超過個位數的功能，會很方便，像是這題需要顯示兩個二位數，每次都要寫一個把整數轉陣列很麻煩，所以也整理進 Library，並且可以決定是否補 0。(-1 為不顯示)

（還有下一頁）

```
}

// 設定七段顯示器的數字
void set_seg_buffer_number(uint16_t number, uint8_t fill_zero) {
    seg_buffer[3] = number / 1000;
    seg_buffer[2] = (number / 100) % 10;
    seg_buffer[1] = (number / 10) % 10;
    seg_buffer[0] = number % 10;
    if (fill_zero == FALSE) {
        if (seg_buffer[3] == 0) seg_buffer[3] = -1;
        if (seg_buffer[2] == 0 && seg_buffer[3] == -1) seg_buffer[2] = -1;
        if (seg_buffer[1] == 0 && seg_buffer[2] == -1) seg_buffer[1] = -1;
    }
}
```

7.2

碼表我認為比較簡單，上面提到已經把 timer 版本的七段顯示器和中斷的 scankey 都放進去 Library，這個小題就只需要在 timer 中計數即可，5ms % 200 = 1s, 1s % 60 = 1 minute，並且在讓七段顯示器顯示數字即可，最後使用 line_index 記錄行數，再用 NewLCD 去印出來就好。

7.3

簡易計算機這題難點是在休眠模式，但因為剛好之前的 NewLCD 有 buffer 的功能，因此休眠的部分我是用 timer 去單純 clear_ lcd，不會清空 buffer 並且當按下按鍵時使用 show_lcd_buffer，就可以做到像休眠模式的效果。

二、【遭遇的問題】：

What problems you faced during design and implementation?

WDT 的喚醒休眠設定是最困擾我的，如果是按照老師所說的需要做到 PowerDown 和喚醒，是我上課期間沒有做到的，只有先做到像休眠模式而已，並沒有真的 PowerDown。

三、【解決方法】：

How did you solve the problems?

繼續上面的 WDT 喚醒問題，我後來翻閱了技術手冊和寫了一個小測試去驗證一下想法。

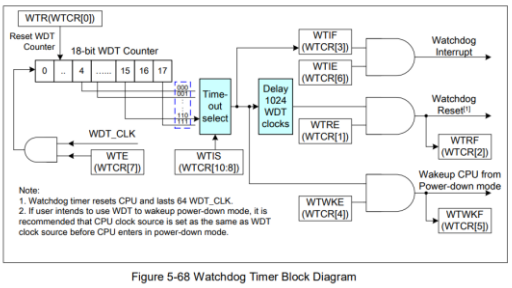


Figure 5-68 Watchdog Timer Block Diagram

（還有下一頁）

可以看到左下角有一個 Note 提到，如果想要使用 WDT 的喚醒功能，那麼建議將 WDT 的 CLK 也設定為和 CPU 一樣的 CLK，所以我在 mcu_init.h 中設定 MCU_CLOCK_SOURCE_LXT 和 TMR3_CLOCK_SOURCE_LXT，去把兩者 CLK 都用 LXT，這樣一來 Enter_PowerDown 和 WDT_IRQHandler 中的 Leave_PowerDown 確實可以進入和離開了，技術手冊中也提到那些 CLK 在 Power-Down 模式中還會運作。

5.3.5 Power-down Mode Clock

When chip enters into Power-down mode, system clocks, some clock sources, and some peripheral clocks will be disabled. Some clock sources and peripherals clock are still active in Power-down mode.

The clocks still active are listed below:

- Clock Generator
  - Internal 10 kHz low speed oscillator clock
  - External 32.768 kHz low speed crystal clock
- Peripherals Clock (When these IP adopt external 32.768 kHz low speed crystal or 10 kHz low speed oscillator as clock source)

四、【未能解決的問題】：

Was there any problem that you were unable to solve? Why was it unsolvable?

繼續上述問題，我發現如果使用 SPI 的 LCD 經過喚醒後，還是無法正常工作，我猜想 SPI 也是使用 CPU 的 CLK，Enter_PowerDown 關閉 CLK 後，使用 Leave_PowerDown 把設定恢復，不過 SPI 可能還需要再次初始化。

五、【投影片的問題】：

1. 再 MCU_init.h 中，#define WWDT_CLOCK_SOURCE_LIRC // LIRC, HCLK_DIV2048, LXT
在 sample 中 使用的是 WWDT_CLOCK_SOURCE_ LXT
請幫我解釋 LIRC, HCLK, LXT，是甚麼意思?有甚麼差別?

首先我想從 WDT 和 WWDT 去了解。

```
#ifdef MCU_INTERFACE_WDT
CLK_EnableModuleClock(WDT_MODULE);
    #ifdef WDT_CLOCK_SOURCE_LIRC
    CLK_SetModuleClock(WDT_MODULE, CLK_CLKSEL1_WDT_S_LIRC, 0);
    #endif
    #ifdef CLK_CLKSEL1_WDT_S_HCLK_DIV2048
    CLK_SetModuleClock(WDT_MODULE, CLK_CLKSEL1_WDT_S_HCLK_DIV2048, 0);
    #endif
    #ifdef WDT_CLOCK_SOURCE_LXT
    CLK_SetModuleClock(WDT_MODULE, CLK_CLKSEL1_WDT_S_LXT, 0);
        #endif
#endif

#ifdef MCU_INTERFACE_WWDT
CLK_EnableModuleClock(WWDT_MODULE);
    #ifdef WWDT_CLOCK_SOURCE_HCLK_DIV2048
    CLK_SetModuleClock(WWDT_MODULE, CLK_CLKSEL2_WWDT_S_HCLK_DIV2048, 0);
    #endif
    #ifdef WWDT_CLOCK_SOURCE_LIRC
    CLK_SetModuleClock(WWDT_MODULE, CLK_CLKSEL2_WWDT_S_LIRC, 0);
    #endif
#endif
```

可以在 SYS_init.c 中發現到 WDT 和 WWDT 之間有些不同，並且需要定義 WDT 才可以使用 WDT 開頭的模式，其中 WDT 有三個，WWDT 只有兩個。

（還有下一頁）

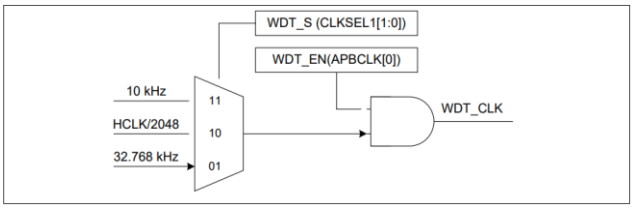


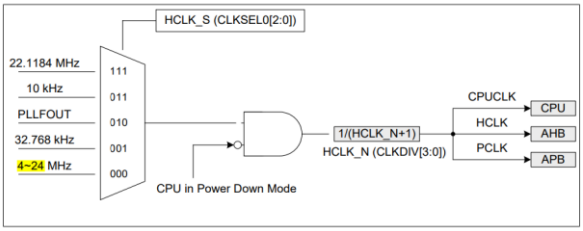
Figure 5-67 Watchdog Timer Clock Control

在技術手冊中我們可以看到 WDT_CLK 是有三個可以去選擇 10kHz、HCLK、32.768kHz，那麼 LIRC, HCLK, LXT，是多少 Hz 可以通過 trace code 更快了解到，以下為 clk.h 的截圖。

```
/*---------------------------------------------------------------------------------------------------------*/
#define CLK_CLKSEL1_WDT_S_LXT           (0x1UL<<CLK_CLKSEL1_WDT_S_Pos)    /*!< Setting WDT clock source as external X'tal 32.768KHz*/
#define CLK_CLKSEL1_WDT_S_HCLK_DIV2048  (0x2UL<<CLK_CLKSEL1_WDT_S_Pos)    /*!< Setting WDT clock source as HCLK/2048 */
#define CLK_CLKSEL1_WDT_S_LIRC          (0x3UL<<CLK_CLKSEL1_WDT_S_Pos)    /*!< Setting WDT clock source as internal 10KHz RC clock */
```

可以到看到這裡將 11 10 01 寫到 CLKSEL。

LXT 為 32.768kHz 的 CLK
LIRC 為 10kHz 的 CLK
那為甚麼沒有寫到 HCLK 的呢？
從技術手冊 System Clock and SysTick Clock 可以看到，HCLK 並不是一個真正有固定 hz 的 CLK，而是一個通過 HCLK_S 去決定的，總共有五種可以選擇。




```
/*---------------------------------------------------------------------------------------------------------*/
#define CLK_CLKSEL0_HCLK_S_HXT          (0x0UL<<CLK_CLKSEL0_HCLK_S_Pos)   /*!< Setting HCLK clock source as external X'tal */
#define CLK_CLKSEL0_HCLK_S_LXT          (0x1UL<<CLK_CLKSEL0_HCLK_S_Pos)   /*!< Setting HCLK clock source as external X'tal 32.768KHz*/
#define CLK_CLKSEL0_HCLK_S_PLL          (0x2UL<<CLK_CLKSEL0_HCLK_S_Pos)   /*!< Setting HCLK clock source as PLL output */
#define CLK_CLKSEL0_HCLK_S_LIRC         (0x3UL<<CLK_CLKSEL0_HCLK_S_Pos)   /*!< Setting HCLK clock source as internal 10KHz RC clock */
#define CLK_CLKSEL0_HCLK_S_HIRC         (0x7UL<<CLK_CLKSEL0_HCLK_S_Pos)   /*!< Setting HCLK clock source as internal 22.1184MHz RC clock */
```

那麼 HCLK 的 Hz 要怎麼設定的呢？

```
// select Chip clock source & set clock source
CLK_EnableXtalRC(MCU_CLOCK_SOURCE_MASK);    // Enable HXT external 12MHz crystal
CLK_WaitClockReady(MCU_CLOCK_STABLE_MASK);
CLK_SetCoreClock(MCU_CLOCK_FREQUENCY);      // Set HCLK clock to 32MHz
```

在 SYS_init.c 當中可以看到初始化時 CLK_SetCoreClock 是使用 mcu_init.h 中 define 的 MCU_CLOCK_FREQUENCY，而 sample 中是用 50000000 Hz，也就是最高 Hz。

```
// Define Clock source
#define MCU_CLOCK_SOURCE
#define MCU_CLOCK_SOURCE_HXT          // HXT, LXT, HIRC, LIRC
#define MCU_CLOCK_FREQUENCY 50000000  // Hz
```

```
uint32_t CLK_SetCoreClock(uint32_t u32Hclk)
{
    uint32_t u32HIRCSTB;

    /* Read HIRC clock source stable flag */
    u32HIRCSTB = CLK->CLKSTATUS & CLK_CLKSTATUS_OSC22M_STB_Msk;

    /* Boundary Check */
    if(u32Hclk > FREQ_50MHZ)
        u32Hclk = FREQ_50MHZ;
    if(u32Hclk < FREQ_25MHZ)
        u32Hclk = FREQ_25MHZ;
```