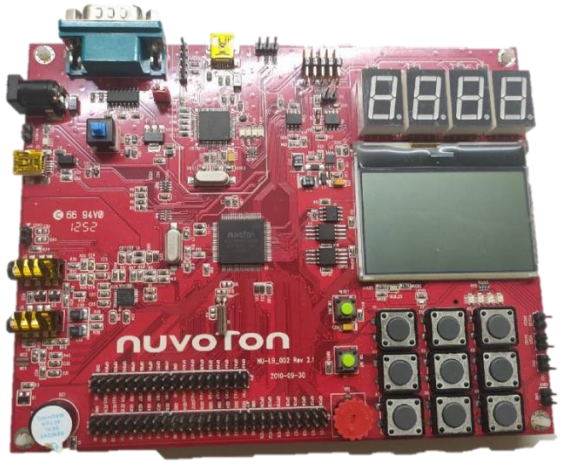
微處理機系統與介面技術 LAB 4

系所：通訊三　　學號 :409430030　　姓名:翁佳煌

<實驗器材>

NUC 140 開發板



PL2303 USB to UART 線

<實驗過程與方法>

首先實作 Basic 的部份，下**圖 1.1** 中，main 中呼叫 SYS_Init()，一開始必然要先使用 CLK_EnableModuleClock(TMR0_MODULE)(在此以 TWM0 為例子)，此外，這函式中寫到的 CLK_SetModuleClock(**圖 1.2**)，是負責選擇 timer 的 clock source，要注意不能選錯，否則之後的計數會錯誤。回到**圖 1.1** 中第 118 行 TIMER_OPEN，裡面三個參數分別放 timer(在此實驗為 TIMER0)、Mode(在此實驗為連續模式)、Freq(在此實驗為 1)，使用 Go to definition 去查看的話可發現(**圖 1.3**)此處為決定如何計數與 prescale 是如何算出 1 秒的部分。回到**圖 1.1**，第 119 行則是設置當時間數完時 Enable 中斷，而第 122 行則是設置中斷發生時對應的 function。第 129 行 TIMER_Start 則是開始計數，當數到 0 時，觸發中斷。中斷發生後會跑到**圖 1.4** 第 32 行的函式，第 37 行為負責清除設置起來的 Interrupt_Flag，以便下次中斷發生時再次使用，第 39 行則是負責儲存每秒計數的值，而回到**圖 1.1** 的第 133~142 行，此部分為負責印出每秒 count 的數值。

▼圖 1.1

```
101  int main(void)
102  {
103      volatile uint32_t u32InitCount;
104
105      /* Unlock protected registers */
106      SYS_UnlockReg();
107
108      /* Init System, peripheral clock and multi-function I/O */
109      SYS_Init();
110
111      /* Lock protected registers */
112      SYS_LockReg();
113
114      /* Init UART0 for printf */
115      UART0_Init();
116
117      /* Open Timer0 in periodic mode, enable interrupt and 1 interrupt tick per second */
118      TIMER_Open(TIMER0, TIMER_PERIODIC_MODE, 1);          //Go to the definition to check.
119      TIMER_EnableInt(TIMER0);                             //when the time is up make enable Interrupt
120
121      /* Enable Timer0 ~ Timer3 NVIC */
122      NVIC_EnableIRQ(TMR0_IRQn);                 //enable corresponding function.  This is very important
123
124      /* Clear Timer0 ~ Timer3 interrupt counts to 0 */
125      g_au32TMRINTCount[0]= 0;
126      u32InitCount = g_au32TMRINTCount[0];
127
128      /* Start Timer0 ~ Timer3 counting */
129      TIMER_Start(TIMER0);                                 //When count to zero will interrupt and into the corresponding func.
130
131      /* Check Timer0 ~ Timer3 interrupt counts */
132      printf("# Timer interrupt counts :\n");
133      while(1)
134      {
135          if(g_au32TMRINTCount[0] != u32InitCount)
136          {
137              printf("TMR0:%3d\n",g_au32TMRINTCount[0]);
138              u32InitCount = g_au32TMRINTCount[0];
139          }
140      }
141      while(1);
142  }
```

▼圖 1.2

```
/* Select Timer 0~3 module clock source */
CLK_SetModuleClock(TMR0_MODULE, CLK_CLKSEL1_TMR0_S_HXT, NULL);
CLK_SetModuleClock(TMR1_MODULE, CLK_CLKSEL1_TMR1_S_HCLK, NULL);
CLK_SetModuleClock(TMR2_MODULE, CLK_CLKSEL1_TMR2_S_HIRC, NULL);
CLK_SetModuleClock(TMR3_MODULE, CLK_CLKSEL1_TMR3_S_HXT, NULL);
```

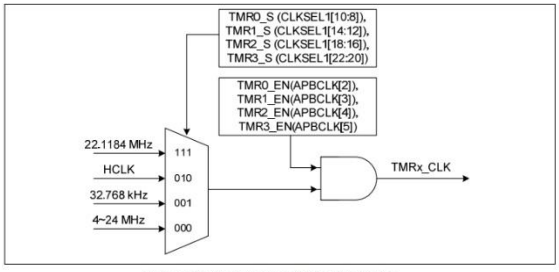


Figure 5-64 Clock Source of Timer Controller

▼圖 1.3

```
uint32_t TIMER_Open(TIMER_T *timer, uint32_t u32Mode, uint32_t u32Freq)
{
    uint32_t u32Clk = TIMER_GetModuleClock(timer);
    uint32_t u32Cmpr = 0, u32Prescale = 0;

    // Fastest possible timer working freq is (u32Clk / 2). While cmpr = 2, pre-scale = 0.
    if(u32Freq > (u32Clk / 2))
    {
        u32Cmpr = 2;
    }
    else
    {
        if(u32Clk >= 0x4000000)
        {
            u32Prescale = 7;    // real prescaler value is 8
            u32Clk >>= 3;
        }
        else if(u32Clk >= 0x2000000)
        {
            u32Prescale = 3;    // real prescaler value is 4
            u32Clk >>= 2;
        }
        else if(u32Clk >= 0x1000000)
        {
            u32Prescale = 1;    // real prescaler value is 2
            u32Clk >>= 1;
        }

        u32Cmpr = u32Clk / u32Freq;
    }

    timer->TCSR = u32Mode | u32Prescale;
    timer->TCMPR = u32Cmpr;

    return(u32Clk / (u32Cmpr * (u32Prescale + 1)));
}
```

TIMER_ONESHOT_MODE(One－Shot 模式)：計數只會進行一次，數到 0 後就發送中斷訊號。

TIMER_PERIODIC_MODE(Periodic 模式)：計數不斷循環，每數到 0 就發送中斷訊號。

TIMER_TOGGLE_MODE (Toggle 模式): 設定 CEN 的值來控制計數器是否要持續計數，1 就是持續 0 則反之。在這狀態下每次數到閥值就會發送一個中斷訊號，然後回到初始值再重新數一次。只要 interrupt flag 被清除，TIMER 就會回到初始值重新進行計數。

▼圖 1.4

```
volatile uint32_t g_au32TMRINTCount[4] = {0};


/**
 * @brief       Timer0 IRQ
 *
 * @param       None
 *
 * @return      None
 *
 * @details     The Timer0 default IRQ, declared in startup_NUC100Series.s.
 */
void TMR0_IRQHandler(void)
{
    if(TIMER_GetIntFlag(TIMER0) == 1)
    {
        /* Clear Timer0 time-out interrupt flag */
        TIMER_ClearIntFlag(TIMER0);

        g_au32TMRINTCount[0]++;
    }
}
```

再來為 Bonus 的部分，這裡的概念與 Basic 的部分幾乎一樣，只是多了一個 Timer 和控制 TIMER 的部分，首先如下**圖 1.5** 第 143 以及第 147 行，TIMER_Open 裡面，頻率的部分，前者選擇一秒數 2 次，後者為一秒數 3 次，第 160~161 行開始兩個 TIMER 的計數，當數到 0 時發生中斷(這部分與 Basic 相似)。至於控制 TIMER 的部分，首先第 126~127 行我先設置兩個變數 key1 與 key2 為 0，用來表示尚未按下按鍵，接著再第 166 行讀入使用鍵盤所輸入的字元，如果輸入為 1，且如果 key1 為 0，則 TIMER 暫停，如果 key1 為 1 則繼續開始計數，輸入為 2 則與上述同理。

▼圖 1.5

```
123  int main(void)
124 {
125     volatile uint32_t u32InitCount;
126     int key1=0;
127     int key2=0;
128     uint8_t u8InChar = 0xFF;
129
130     /* Unlock protected registers */
131     SYS_UnlockReg();
132
133     /* Init System, peripheral clock and multi-function I/O */
134     SYS_Init();
135
136     /* Lock protected registers */
137     SYS_LockReg();
138
139     /* Init UART0 for printf */
140     UART0_Init();
141
142     /* Open Timer0 in periodic mode, enable interrupt and 1 interrupt tick per second */
143     TIMER_Open(TIMER0, TIMER_PERIODIC_MODE, 2);     //Go to the definition to check.
144     TIMER_EnableInt(TIMER0);                        //when the time is up enable Interrupt
145
146     /* Open Timer1 in periodic mode, enable interrupt and 2 interrupt ticks per second */
147     TIMER_Open(TIMER1, TIMER_PERIODIC_MODE, 3);
148     TIMER_EnableInt(TIMER1);
149
150     /* Enable Timer0 ~ Timer1 NVIC */
151     NVIC_EnableIRQ(TMR0_IRQn);                      //enable corresponding function.  This is very important
152     NVIC_EnableIRQ(TMR1_IRQn);
153
154
```

```
155     /* Clear Timer0 ~ Timer3 interrupt counts to 0 */
156     g_au32TMRINTCount[0] = g_au32TMRINTCount[1] = 0;
157     u32InitCount = g_au32TMRINTCount[0];
158
159     /* Start Timer0 ~ Timer1 counting */
160     TIMER_Start(TIMER0);
161     TIMER_Start(TIMER1);
162     //When count to zero will interrupt and into the corresponding func.
163     printf("\tStart TIMER\n(Key1 to control TWM0, Key2 to contrlo TWM1)\n");
164     while(1)
165     {
166     u8InChar = UART_READ(UART0);
167         if(u8InChar == '1')
168         {
169             if(key1==0){
170               TIMER_Stop(TIMER0);
171               key1=1;
172             }
173             else{
174               TIMER_Start(TIMER0);
175               key1=0;
176               }
177         }
178         else if(u8InChar == '2')
179         {
180             if(key2==0){
181               TIMER_Stop(TIMER1);
182               key2=1;
183             }
184             else{
185               TIMER_Start(TIMER1);
186               key2=0;
187               }
188         }
189          printf("\rTMR0:%3d\tTMR1:%3d ",g_au32TMRINTCount[0], g_au32TMRINTCount[1]);
190     }
191 }
```

除了以上控制方法，還可使用九宮格的按鈕，設置 GPIO，利用按鈕按下來發生中斷，如下**圖 1.6**，首先要先設置 GPIO_SetMode，Quasi-bidirection 為雙向模式，GPIO_EnableInt 則是當 falling edge trigger 發生時 enable 中斷，換句話說，當按下按鈕，也就是中斷發生時，會進入 GPAB_IRQHandler 函式裡頭，這部分為負責控制 TIMER 的暫停與繼續計數。

▼圖 1.6

```
/*  Configure PE.5 as Quasi-bidirection mode and enable interrupt by falling edge trigger */
GPIO_SetMode(PA, BIT0, GPIO_PMD_QUASI);
GPIO_SetMode(PA, BIT1, GPIO_PMD_QUASI);
GPIO_SetMode(PA, BIT2, GPIO_PMD_QUASI);
GPIO_SetMode(PA, BIT3, GPIO_PMD_QUASI);
GPIO_SetMode(PA, BIT4, GPIO_PMD_QUASI);
GPIO_SetMode(PA, BIT5, GPIO_PMD_QUASI);

GPIO_EnableInt(PA, 0, GPIO_INT_FALLING);
GPIO_EnableInt(PA, 1, GPIO_INT_FALLING);
GPIO_EnableInt(PA, 2, GPIO_INT_FALLING);

NVIC_EnableIRQ(GPAB_IRQn);

PA3=0; PA4=1; PA5=1;

/* Enable interrupt de-bounce function and select de-bounce sampling cycle time is 1024 clocks of LIRC clock */
  GPIO_SET_DEBOUNCE_TIME(GPIO_DBCLKSRC_LIRC, GPIO_DBCLKSEL_1024);
  GPIO_ENABLE_DEBOUNCE(PA, BIT0 | BIT1 | BIT2);
```

```
void GPAB_IRQHandler(void){
    /* To check if PA interrupt occurred */
  if(GPIO_GET_INT_FLAG(PA, BIT0))          //key3: clear to zero
    {
        GPIO_CLR_INT_FLAG(PA, BIT0);
        g_au32TMRINTCount[0] = g_au32TMRINTCount[1] = 0;
        t_cmd[2] = 1;
    }
    else if(GPIO_GET_INT_FLAG(PA, BIT1))   //key2
    {
        GPIO_CLR_INT_FLAG(PA, BIT1);
        t_cmd[1] = !t_cmd[1];
    }
    else if(GPIO_GET_INT_FLAG(PA, BIT2))   //key1
    {
        GPIO_CLR_INT_FLAG(PA, BIT2);
        t_cmd[0] = !t_cmd[0];
    }
}
```

<心得與收穫>

這次的主題我認為最難理解的部分就是他如何算出 1 秒，雖然應用很簡單，只需要去手動更改數值就可以，但其背後的運算我看了 Datasheet 的內容，也還是一知半解的感覺，實在是讓我頭痛。另外，這次的實驗與我們這組要報告的 PWM 有部分名詞有些重疊，像是當初看到 timer、prescaler 等等的名詞都不太懂他們到底負責做什麼，但做完這次的 LAB 也有初步的認識與了解。此外，在 Bonus 的部分，我誤以為是要用電腦鍵盤上的數字鍵 1 和 2 來控制，但後來有重新理解，並運用 MCU 上九宮格的按鈕來設置兩個 GPIO 控制 TIMER，GPIO 的部分原來可以如此深度的應用，我這才發現我對 GPIO 許多模式不太熟悉，看來必須再回頭看這部分的資料。