微處理機系統與介面技術 LAB 4

系所：通訊三　　學號：409430022　　姓名：李宜叡

一、實驗器材

NUC 140 V2.0 開發板

PL2303 USB to UART線

二、實驗過程與方法

main.c與這次實驗有關的內容如下:

```
156 int main(void){
157     /* Unlock protected registers */
158     SYS_UnlockReg();
159     /* Init System, peripheral clock and multi-function I/O */
160     SYS_Init();
161     /* Lock protected registers */
162     SYS_LockReg();
163     /* Init UART0 for printf */
164     UART0_Init();
165     printf("\n\nCPU @ %d MHz\n", SystemCoreClock/1000000);
166     printf("+--------------------------------------------------+\n");
167     printf("|                  Start Timer:                    |\n");
168     printf("+--------------------------------------------------+\n\n");
169     printf("# Timer Settings:\n");
170     printf("Clock source 12 MHz;\n");
171     printf("Clock source HCLK(50 MHz);\n");
172     printf("Clock source HIRC(22 MHz);\n");
173     printf("Clock source 12 MHz;\n");
```

Line156~173：前面先初始化，並且印出相關訊息。

```
174
175     //Timer 0: 1 sec 2 times ; Timer 1: 1 sec 1 time; Timer 3: 1 sec 3 times.
176     /* Open Timer0 in periodic mode, enable interrupt and 2 interrupt tick per second */
177     TIMER_Open(TIMER0, TIMER_PERIODIC_MODE, 2);
178     TIMER_EnableInt(TIMER0);
179     /* Open Timer1 in periodic mode, enable interrupt and 1 interrupt ticks per second */
180     TIMER_Open(TIMER1, TIMER_PERIODIC_MODE, 1);
181     TIMER_EnableInt(TIMER1);
182     /* Open Timer3 in periodic mode, enable interrupt and 3 interrupt ticks per second */
183     TIMER_Open(TIMER3, TIMER_PERIODIC_MODE, 3);
184     TIMER_EnableInt(TIMER3);
```

Line174~184：打開Timer，並且參數與這次實驗目標有關，後續會說明。

Timer0 就是一秒數兩次的計時器、Timer1 就是一秒數 1 次的計時器、Timer3 就是一秒數三次的計時器。

```
186     /* Enable Timer0 ,Timer3 NVIC */
187     NVIC_EnableIRQ(TMR0_IRQn);//Interrupt Request
188     NVIC_EnableIRQ(TMR3_IRQn);
189     NVIC_EnableIRQ(TMR1_IRQn);
190
191     /* Clear Timer0 ,Timer3 interrupt counts to 0 */
192     g_au32TMRINTCount[0] = g_au32TMRINTCount[1] = g_au32TMRINTCount[2]= 0;
193
194     /* Start Timer0 ~ Timer3 counting */
195     TIMER_Start(TIMER0);
196     TIMER_Start(TIMER3);
197     TIMER_Start(TIMER1);
```

Line186~197：設定成中斷的模式，清空中斷計數並且啟動計時器(拿掉Mask)。

```
198   /*  Configure PE.5 as Quasi-bidirection mode and enable interrupt by falling edge trigger */
199     GPIO_SetMode(PA, BIT0, GPIO_PMD_QUASI);
200     GPIO_SetMode(PA, BIT1, GPIO_PMD_QUASI);
201     GPIO_SetMode(PA, BIT2, GPIO_PMD_QUASI);
202     GPIO_SetMode(PA, BIT3, GPIO_PMD_QUASI);
203     GPIO_SetMode(PA, BIT4, GPIO_PMD_QUASI);
204     GPIO_SetMode(PA, BIT5, GPIO_PMD_QUASI);
205     GPIO_EnableInt(PA, 0, GPIO_INT_FALLING);//falling edge trigger
206     GPIO_EnableInt(PA, 1, GPIO_INT_FALLING);
207     GPIO_EnableInt(PA, 2, GPIO_INT_FALLING);
208     NVIC_EnableIRQ(GPAB_IRQn);//Enable External Interrupt
209     PA3=0;
210     PA4=1;
211     PA5=1;
```

Line198~211：設定鍵盤並且將PA0～2 設定成 Falling edge 觸發。

```
213   /* Enable interrupt de-bounce and select sampling cycle time is 16 clocks*/
214     GPIO_SET_DEBOUNCE_TIME(GPIO_DBCLKSRC_LIRC, GPIO_DBCLKSEL_16);
215     GPIO_ENABLE_DEBOUNCE(PA, BIT0 | BIT1 | BIT2);
216
217     printf("# Timer interrupt counts :\n");
```

Line213~217：設定 debounce。

```
218   /* Check Timer0 ~ Timer3 interrupt counts *
219     while(1){//waiting for interrupt
220       if(flag[2]==1){
221         //1 sec 1 time, timer1
222         flag[2] = 0;
223         count_t();
224         print_time();
225       }
226       if(flag[0] == 1){
227         flag[0] = 0;//timer 0, 1 sec 2 time
228         print_time();
229       }
230       if(flag[1] == 1){
231         flag[1] = 0;//timer 3, 1 sec 3 times
232         print_time();
233       }
234       if(cmd[2] == 1){
235         cmd[2] = 0;
236         print_time();
237       }
238     }
239 }
```

Line218~239：中斷用迴圈。在此迴圈中會檢查旗標的數值，如圖中 timer 1 對應到 flag[2] ，當flag[2]被前面設定成1時就印出數值並清為零。

```
30 void TMR0_IRQHandler(void){
31    //1 is time-out interrupt occered.
32     if(TIMER_GetIntFlag(TIMER0) == 1){
33
34         /* Clear Timer0 time-out interrupt flag */
35         TIMER_ClearIntFlag(TIMER0);
36
37       if(!cmd[0]){
38         g_au32TMRINTCount[0]++;
39         flag[0] = 1;
40       }
41     }
42 }
43
44
45 void TMR3_IRQHandler(void){
46     if(TIMER_GetIntFlag(TIMER3) == 1){
47         /* Clear Timer3 time-out interrupt flag */
48         TIMER_ClearIntFlag(TIMER3);
49       if(!cmd[1]){
50         g_au32TMRINTCount[1]++;
51         flag[1] = 1;
52       }
53     }
54 }
```

Line30~54：Timer0、3的 IRQHandler，處理 time-out interrupt。

```
56 void TMR1_IRQHandler(void){
57     if(TIMER_GetIntFlag(TIMER1) == 1){
58         /* Clear Timer1 time-out interrupt flag */
59         TIMER_ClearIntFlag(TIMER1);
60       if(!cmd[2]){
61         g_au32TMRINTCount[2]++;
62         flag[2] = 1;
63       }
64     }
65 }
```

Line56~65：同上，Timer 1的IRQHandler。

```
67 void GPAB_IRQHandler(void){
68     //gpio_INT
69     //if we press the key, then PA interrupt occures.
70     if(GPIO_GET_INT_FLAG(PA, BIT0)){
71       GPIO_CLR_INT_FLAG(PA, BIT0);
72       g_au32TMRINTCount[0] = g_au32TMRINTCount[1] = 0;
73       cmd[2] = 1;
74     }
75     else if(GPIO_GET_INT_FLAG(PA, BIT1)){
76       GPIO_CLR_INT_FLAG(PA, BIT1);
77       cmd[1] = !cmd[1];
78     }
79     else if(GPIO_GET_INT_FLAG(PA, BIT2)){
80       GPIO_CLR_INT_FLAG(PA, BIT2);
81       cmd[0] = !cmd[0];
82     }
83     else{
84       //Un-expected interrupt. Just clear all PA, PB interrupts
85         PA->ISRC = PA->ISRC;
86         PB->ISRC = PB->ISRC;
87     }
88 }
```

Line67~88：處理按下鍵盤按鍵時的反應，分別有兩個按下暫停計數以及一個全部清空。

三、問題與資料

1. 解讀TIMER_Open()

在main中，我們呼叫了TIMER_Open()，這個函式定義在timer.c 裡面，它的功用就是在特定模式與頻率下打開timer 。有關模式，定義寫可以用下面四種模式打開：

- TIMER_ONESHOT_MODE：計數只會進行一次，到閥值之後就發送中斷訊號。
- TIMER_PERIODIC_MODE：計數不斷循環，每次到閥值就發送中斷訊號，這個訊號會發送到NVIC去通知CPU。
- TIMER_TOGGLE_MODE：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。只要中斷旗標(interrupt flag)被軟體清除，計數器馬上就會再次回到初始值重新進行計數。
- TIMER_CONTINUOUS_MODE：到閥值也能持續數下去的模式。

```
44  uint32_t TIMER_Open(TIMER_T *timer, uint32_t u32Mode, uint32_t u32Freq)
45  {
46      uint32_t u32Clk = TIMER_GetModuleClock(timer);
47      uint32_t u32Cmpr = 0, u32Prescale = 0;
48
49      // Fastest possible timer working freq is (u32Clk / 2). While cmpr = 2, pre-scale = 0.
50      if(u32Freq > (u32Clk / 2))
51      {
52          u32Cmpr = 2;
53      }
```

Line44~53：宣告參數，其中Cmpr就是目標閥值，數到這個數字會停下；而Prescale則是除頻的數值，讀取到Clk的頻率後要進行除頻，後面會附上除頻的資料。而50行開始就是處理讀取到的頻率，首先要判斷是否超過最快工作頻率，最快只能是讀取到的一半，這點沒有查清楚，倒是查到電晶體的工作速度極限大約是$10^{17}Hz$，但因為現實因素大概到$10^{14}Hz$就是極限，這都是因為普朗克關係式$E = h\omega$導致的。

```
54      else
55      {
56          if(u32Clk >= 0x4000000)
57          {
58              u32Prescale = 7;    // real prescaler value is 8
59              u32Clk >>= 3;//u32Clk devided by 2^3=8
60          }
61          else if(u32Clk >= 0x2000000)//33554432~33Mhz
62          {
63              u32Prescale = 3;    // real prescaler value is 4
64              u32Clk >>= 2;
65          }
66          else if(u32Clk >= 0x1000000)//16777216
67          {
68              u32Prescale = 1;    // real prescaler value is 2
69              u32Clk >>= 1;
70          }
71
72          u32Cmpr = u32Clk / u32Freq;
73      }
```

Line54~73：衡量讀取到的頻率大小，根據情況除頻。至於為什麼Prescale設定 1 但是實際除頻數是 2 ，其原因是**計數器在數到目標閥值之後還會再等待一個週**

期才重置，在程式碼中我們是設定Falling edge trigger，也就是在目標閥值之後的下一個負緣才會觸發。而在72行中我們可以看到關係式$Cmpr = \frac{Clk}{Frequency}$。

```
75      timer->TCSR = u32Mode | u32Prescale;
76      timer->TCMPR = u32Cmpr;
77
78      return(u32Clk / (u32Cmpr * (u32Prescale + 1)));
79  }
```

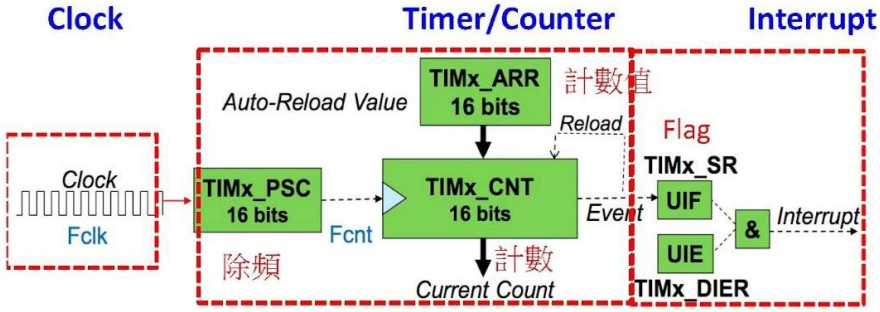
Line75~79：設定TCSR(Control and Status)，藉由Mode與Prescale取交集去設定它的除頻數以及模式。還有設定目標閥值，前面那個關係式能夠讓我們以比例去劃分出目標閥值而不用手動去調整它。最後返還的式子如下：

$$return = \frac{Clock}{Cmpr * (Prescale + 1)}$$

式子中，prescale要加一的原因前面已經提到過。在整個函式中，一開始傳入的frequency(也就是我們設定的頻率值)會影響Cmpr的數值，下面會簡短整理工作原理。

2.計數器工作原理：如何湊出1秒

在這裡借用交大鄭雲謙教授的簡報說明：



除頻後頻率送到計數器啟動計數，並且前面設定好了計數值(目標閥值)，計數器就會以除頻後的頻率運作，數到目標閥值後才會中斷。又由於前面提到過在目標閥值數滿後，會等到下一個Cycle才會重置，因此相關式子中的除頻數與計數值(如果沒有像我們開發版已經寫好前面的Cmpr關係式的話，計數值就要手動調整而非依比例計算)都要加一再做計算，至於設定觸發在前面有提到：

```
205     GPIO_EnableInt(PA, 0, GPIO_INT_FALLING);//falling edge trigger
206     GPIO_EnableInt(PA, 1, GPIO_INT_FALLING);
207     GPIO_EnableInt(PA, 2, GPIO_INT_FALLING);
```

回到正題，公式如下：

$$Time(時間) = 計數值 * 除頻數 * 系統clock週期$$

$$time = (Cmpr) * (Prescale + 1) * Clk$$

或是$time = \frac{Cmpr*(Prescale+1)}{ClkFrequency}$

若要手動設定Cmpr，舉例來說想要設40，則必須賦值39。

當今天系統頻率是5000000hz，要湊到1秒可以設除頻數為49999，並假設CMPR是100，則有：$\frac{(49999+1)*100}{5000000} = 1sec$，這樣就湊出一秒。不過Cmpr跟Prescale在TIMER_Open()都已經被包裝好，在這裡我們只要調Frequency就能得到效果。

3.怎麼會有22.1184Mhz這種奇怪頻率?

在Technical reference p.148、p.328跟官方給的sample code以及system_NUC100Series.h中，我們都可以發現這個頻率的身影。

```
/*---------------------------------------------------------------------------
  Define SYSCLK
 *---------------------------------------------------------------------------*/
#define __HXT       (12000000UL)    /*!< External Crystal Clock Frequency     */
#define __LXT       (32768UL)       /*!< External Crystal Clock Frequency 32.768KHz */
#define __HIRC      (22118400UL)    /*!< Internal 22M RC Oscillator Frequency */
#define __LIRC      (10000UL)       /*!< Internal 10K RC Oscillator Frequency */
#define __HSI       (50000000UL)    /*!< PLL default output is 50MHz */

extern uint32_t SystemCoreClock;    /*!< System Clock Frequency (Core Clock) */
extern uint32_t CyclesPerUs;        /*!< Cycles per micro second              */
extern uint32_t PllClock;           /*!< PLL Output Clock Frequency           */
```

PLL Control Register (PLLCON)

The PLL reference clock input is from the external 4~24 MHz high speed crystal clock input or from the internal 22.1184 MHz high speed oscillator. These registers are use to control the PLL output frequency and PLL operating mode

System clock = internal 22.1184 MHz high speed oscillator						
Baud rate	Mode0		Mode1		Mode2	
	Parameter	Register	Parameter	Register	Parameter	Register
921600	x	x	A=0,B=11	0x2B00_0000	A=22	0x3000_0016
460800	A=1	0x0000_0001	A=1,B=15 A=2,B=11	0x2F00_0001 0x2B00_0002	A=46	0x3000_002E

以網路查到的資料來說，原因就是在使用串列傳輸(serial)時，標準的Baud rate是192才導致我們做的震盪頻率要做成22.1184Mhz，由關係式可以看出來：

$$\frac{22.1184}{115200} = 192$$

四、心得

官方的Technical Reference還是寫得蠻糟的，在計數器的四種模式那邊看得有點頭痛，它的文法讓人看了都充滿遺憾，我簡直可以引用浮士德的句子「人生缺少遺憾，在此得到補償。」來形容它的文法。

這次沒有再進一步查PLL的資料，反正鎖相環電路是電子二類比的東西，而且我只有跟著交大學到回授前面就停了，後面要看也要多花時間才能懂，就不深入再爬了。看完TIMER_Open()還是可以發現其實新唐很貼心，都已經先幫忙包裝好程式，到最後我們只要設定頻率而不用動手下去改Cmpr跟Prescale的數值，省了一些功夫。