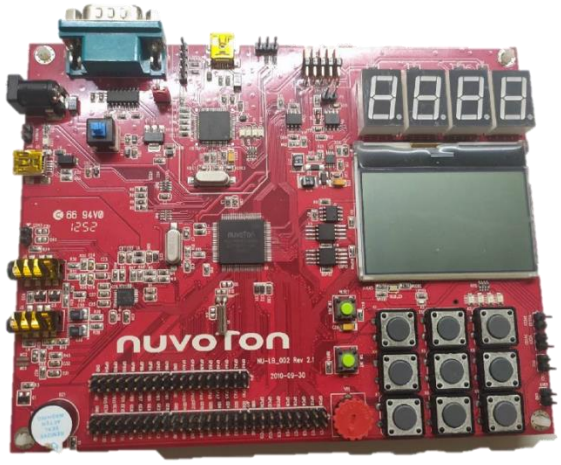
微處理機系統與介面技術 LAB 5

系所：通訊三　　學號 :409430030　　姓名:翁佳煌

<實驗器材>

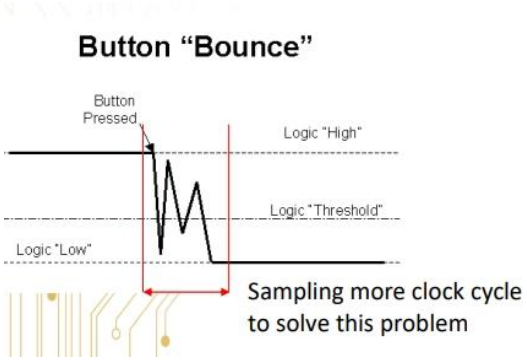
NUC 140 開發板



PL2303 USB to UART 線

<實驗過程與方法>

首先，在使用按鈕生成脈衝波時，可能會導致在進入穩定階段之前的電壓彈跳不穩的問題(**圖 1.1**)，這會使 GPIO Interrupt 讀到很多次設定的 pin 腳的變化，導致執行很多次中斷。



▲圖 1.1

為了解決這問題，我在**圖 1.2** 中，第 139~141 行中，設定 GPIO 為 QUASI bi-direction mode，設定鍵盤並且將 PA2 設定成 Falling edge 觸發。在 147~148 行中設定 debounce。

```
128 int main(void)
129 {
130     /* Unlock protected registers */
131     SYS_UnlockReg();
132     /* Init System, peripheral clock and multi-function I/O */
133     SYS_Init();
134     /* Lock protected registers */
135     SYS_LockReg();
136     /* Init UART0 for printf */
137     UART0_Init();
138     /*  Configure PE.5 as Quasi-bidirection mode and enable interrupt by falling edge trigger */
139     GPIO_SetMode(PA, BIT2, GPIO_PMD_QUASI);
140     /* Use PA.2 to check time-out period time */
141     GPIO_EnableInt(PA, 2, GPIO_INT_FALLING); //GPIO_EnableInt(PA, 2, GPIO_INT_RISING) is also work;
142     NVIC_EnableIRQ(GPAB_IRQn);
143
144
145     PA3=0; PA4=1; PA5=1;
146     /* Enable interrupt de-bounce function and select de-bounce sampling cycle time is 1024 clocks of LIRC clock */
147     GPIO_SET_DEBOUNCE_TIME(GPIO_DBCLKSRC_LIRC, GPIO_DBCLKSEL_1024);
148     GPIO_ENABLE_DEBOUNCE(PA, BIT0 | BIT1 | BIT2);
149
```

▲圖 1.2

下圖第 109~122 行，設定 GPIO 中斷，處理按下鍵盤 KEY1 時的反應，並用一個全域變數 g_flag 去判斷目前按鈕狀態。

```
109 void GPAB_IRQHandler(void){
110     /* To check if PA interrupt occurred */
111     if(GPIO_GET_INT_FLAG(PA, BIT2)) //key1
112     {
113         printf("Change!!!\n");
114         if(g_flag==0){
115             g_flag=1;
116         }
117         else{
118             g_flag=0;
119         }
120         GPIO_CLR_INT_FLAG(PA, BIT2);
121     }
122 }
```

第 150~157 行，**圖 1.3**，如果超過 WDT 所設定的時間還尚未餵狗的話(也就是尚未重設計數時間)，就會 reset，並且會把 WDT_GET_RESET_FLAG 設置為 1，並且這個值並不會被 reset，直到重新執行到 150 行時會讀到為 1，並在 155 行中才使用 WDT_CLEAR_RESET_FLAG 把它清除，並印出相關文字。

```
150     if(WDT_GET_RESET_FLAG() == 1 ) //have reset flag
151         {
152           g_flag=0;
153           /* Use PA.2 to check time-out period time */
154           GPIO_SetMode(PA, BIT2, GPIO_PMD_QUASI);
155           WDT_CLEAR_RESET_FLAG();
156           printf("*** WDT time-out reset occurred (Alarm!!!~~~reset!!!)***\n\n");
157         }
158
```

▲圖 1.3

前面有提到，我們設定一個全域變數 g_flag 去判斷目前按鈕狀態，**圖 1.4** 第 161~168 行，如果 g_flag 為 0 表示 safe，如果為 1 表示 alarm。
第 177~182 行，如果目前是 safe 的狀態就把 WDT_Open 的 EnableReset 參數會被設定成 False，如此一來，reset 時**圖 1.3** 中的 WDT_GET_RESET_FLAG 就不會被設置為 1，他會繼續回到 while 迴圈裡印出 safe 文字。如果目前狀態是 alarm，則 WDT_Open 的 EnableReset 參數會設定成 True，reset 時**圖 1.3** 中的 WDT_GET_RESET_FLAG 就會被設置為 1。
第 185~188 行，設置 WDT 中斷的 Enable 與對應的 interrupt function。
第 190~191 行，設置 CLK_SysTickDelay 控制速度以便觀察印在 putty 上的文字。

```
159     printf("Start Lab5\n");
160     while(1){
161         if(g_flag==0){
162           //PB11=1; BUZZER CLOSE
163           printf("Safe!\n");
164         }
165         else{
166             printf("Alarm!!!\n");
167           //PB11=0; BUZZER OPEN
168         }
169
170         /* Because of all bits can be written in WDT Control Register are write-protected;
171            To program it needs to disable register protection first. */
172         SYS_UnlockReg();
173
174         /* Enable WDT time-out reset function and select time-out interval to 2^14 * WDT clock then start WDT counting */
175         g_u8IsWDTTimeoutINT = 0;
176
177         if(g_flag==0)
178           WDT_Open(WDT_TIMEOUT_2POW16, WDT_RESET_DELAY_1026CLK, FALSE, FALSE);
179         //to change 2^16 ,WDT IRQ will wait1024*Twdt, if WDT counter not reset, WDT will generate chip reset signal.
180
181         else  // have to reset
182           WDT_Open(WDT_TIMEOUT_2POW16, WDT_RESET_DELAY_1026CLK, TRUE, FALSE); // will set WDT_GET_RESET_FLAG
183
184         /* Enable WDT interrupt function */
185         WDT_EnableInt();
186
187         /* Enable WDT NVIC */
188         NVIC_EnableIRQ(WDT_IRQn);
189
190         CLK_SysTickDelay(1000000);
191         CLK_SysTickDelay(1000000);
192         //while(1);
193     }
194 }
```

▲圖 1.4

▼WDT_Open

```
51  void WDT_Open(uint32_t u32TimeoutInterval,
52                uint32_t u32ResetDelay,
53                uint32_t u32EnableReset,
54                uint32_t u32EnableWakeup)
55  {
56      WDT->WTCRALT = u32ResetDelay;
57
58      WDT->WTCR = u32TimeoutInterval | WDT_WTCR_WTE_Msk |
59                  (u32EnableReset << WDT_WTCR_WTRE_Pos) |
60                  (u32EnableWakeup << WDT_WTCR_WTWKE_Pos);
61      return;
62  }
```

在 WDT_OPEN 中，可藉由參考下圖的表格設定出想要的 Timeout 時間。

WTIS	Timeout Interval Selection T_{TIS}	Interrupt Period T_{INT}	WTR Timeout startingInterval (WDT_CLK=10 kHz) MIN. T_{WTR} ~ Max. T_{WTR}
000	$2^4 * T_{WDT}$	$1024 * T_{WDT}$	1.6 ms ~ 104 ms
001	$2^6 * T_{WDT}$	$1024 * T_{WDT}$	6.4 ms ~ 108.8 ms
010	$2^8 * T_{WDT}$	$1024 * T_{WDT}$	25.6 ms ~ 128 ms
011	$2^{10} * T_{WDT}$	$1024 * T_{WDT}$	102.4 ms ~ 204.8 ms
100	$2^{12} * T_{WDT}$	$1024 * T_{WDT}$	409.6 ms ~ 512 ms
101	$2^{14} * T_{WDT}$	$1024 * T_{WDT}$	1.6384 s ~ 1.7408 s
110	$2^{16} * T_{WDT}$	$1024 * T_{WDT}$	6.5536 s ~ 6.656 s
111	$2^{18} * T_{WDT}$	$1024 * T_{WDT}$	26.2144 s ~ 26.3168 s

Table 5-8 Watchdog Timeout Interval Selection

圖 1.5 為 WDT 對應的 Interrupt function，在這中斷裡，會在 39 行先清除中斷的 flag，以便下次中斷可以再被偵測。第 44 行則是判斷按鈕狀態印出相關文字。

```
34  void WDT_IRQHandler(void)
35  {
36      if(WDT_GET_TIMEOUT_INT_FLAG() == 1)
37      {
38          /* Clear WDT time-out interrupt flag */
39          WDT_CLEAR_TIMEOUT_INT_FLAG();
40
41          g_u8IsWDTTimeoutINT = 1;  //
42
43          printf("WDT time-out interrupt occurred.(Watch dog timer ocurred!!!)\n");
44          if(g_flag==0)
45          {
46              printf("No problem~~\n");
47          }
48      }
49  }
50
```

▲圖 1.5

<心得與收穫>

這次的LAB讓我理解看門狗計時器的功能以及如何使用，這項實用的技術常常在許多裝置上面都看得到，像是電腦、手機等等，在NUC140 Technical Reference Manual中WDT的功能介紹寫得非常清楚，WDT是一種電腦硬體的計時裝置，當系統的主程式發生某些錯誤事件時，像是當機或未定時的清除看門狗計時器的內含計時值，這時看門狗計時器就會對系統發出重設、重新啟動或關閉的訊號，使系統從懸停狀態回復到正常運作狀態。另外，我覺得有趣的是，老師上課提到有些工程師在工作上遇到產品找不到突然當機的原因，導致無法下班時，他們可能會運用這個功能去解決問題，只要不被發現有問題就是沒有問題，我認為是一個非常有趣的解決方法。