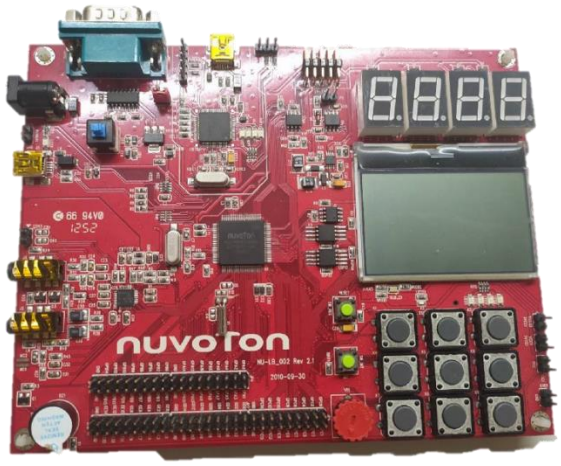
微處理機系統與介面技術 LAB 2

系所：通訊三　　學號 :409430030　　姓名:翁佳煌

<實驗器材>

NUC 140 開發板



<實驗過程與方法>

如圖 1.1，一開始的 22~25 行，分別宣告變數，用 volatile 宣告的型別變量表示可以被某些編譯器未知的因素更改，比如：**作業系統**、硬體或者其它執行緒等；它可以用來提醒編譯器後面所定義的變數隨時有可能改變，因此編譯後的程式每次需要儲存或讀取這個變數的時候，都會直接從變數位址中讀取資料。g_u32comRbytes 用來計算幾個 bytes、g_u32comRhead 和 g_u32comRtail 來記錄 RecData Buffer 的開頭與結束的位置、g_bWait 則是後面會用來判斷是否結束程式。

另外，第 98~100 行，則是設定 RGB LED 腳位為 Output mode。

```
19  #define RXBUFSIZE 1024
20  uint8_t g_u8RecData[RXBUFSIZE]  = {0};
21
22  volatile uint32_t g_u32comRbytes = 0;
23  volatile uint32_t g_u32comRhead  = 0;
24  volatile uint32_t g_u32comRtail  = 0;
25  volatile int32_t g_bWait         = TRUE;
26
27  void SYS_Init(void)
28  {
67  void UART0_Init()
68  {
79  /*---------------------------------------------------------------
80  /* MAIN function
81  /*---------------------------------------------------------------
82  void UART_TEST_HANDLE(void);
83  void UART_FunctionTest(void);
84  int main()
85  {
86    /* Unlock protected registers */
87    SYS_UnlockReg();
88
89    /* Init System, peripheral clock and multi-function I/O *
90    SYS_Init();
91
92    /* Lock protected registers */
93    SYS_LockReg();
94
95    /* Init UART0 for printf and testing */
96    UART0_Init();
97
98    GPIO_SetMode(PA,BIT12,GPIO_PMD_OUTPUT);
99    GPIO_SetMode(PA,BIT13,GPIO_PMD_OUTPUT);
100   GPIO_SetMode(PA,BIT14,GPIO_PMD_OUTPUT);
101
102   UART_FunctionTest();
103   while(1);
104 }
```

▲圖 1.1

呼叫 UART_FunctionTest()後，發現下圖 1.2 中第 196 行的 UART_EnableInt 裡面有三個負責控制中斷的參數，分別為(1)UART_IER_RDA_IEN_Msk 、(2)UART_IER_THRE_IEN_Msk 、(3) UART_IER_TOUT_IEN_Msk，這三種參數去做位元運算。查看 NuMicro 的 Manual 會發現 UA_IER(Interrupt Enable Register) 裡面有很多種中斷，上述三種就包含在此。

(1)為 Receive Data Available Interrupt Enable，判斷是否有輸入，1 = Enable INT_RDA，而 0 = Mask off INT_RDA。

(2)為 Transmit Holding Register Empty Interrupt Enable，1 = Enable INT_THR，0 = Mask off INT_THRE。

(3)為 Receiver buffer time out interrupt。

此外，我發現把(2)和(3)拿掉對本次實驗結果並無影響，我認為是因為本次實驗只需要藉由輸入來判斷是否中斷，因此只會用到(1)並不會用到(2)和(3)的中斷判斷條件。

```
175  void UART_FunctionTest()
176  {
177    /*
178      printf("+------------------------------------------------------------+\n");
179      printf("|  UART Function Test                                        |\n");
180      printf("+------------------------------------------------------------+\n");
181      printf("|  Description :                                             |\n");
182      printf("|    The sample code will print input char on terminal       |\n");
183      printf("|    Please enter any to start     (Press '0' to exit)       |\n");
184      printf("+------------------------------------------------------------+\n");
185    */
186      /*
187          Using a RS232 cable to connect UART0 and PC.
188          UART0 is set to debug port. UART0 is enable RDA and RLS interrupt.
189          When inputting char to terminal screen, RDA interrupt will happen and
190          UART0 will print the received char on screen.
191      */
192
193      printf("LAB2-UART\n");
194
195      /* Enable Interrupt and install the call back function */
196      UART_EnableInt(UART0, (UART_IER_RDA_IEN_Msk | UART_IER_THRE_IEN_Msk | UART_IER_TOUT_IEN_Msk));
197      while(g_bWait);
198
199      /* Disable Interrupt */
200      UART_DisableInt(UART0, (UART_IER_RDA_IEN_Msk | UART_IER_THRE_IEN_Msk | UART_IER_TOUT_IEN_Msk));
201      g_bWait = TRUE;
202      printf("\nUART Sample Demo End.\n");
203
204  }
205
```

▲圖 1.2

再來卡進 197 行的 while 後，中斷是由硬體發出的訊號，因作業系統(OS)的緣故，程式會跑到下圖 1.3 的 106 行的 UART02_IRQHandler()，這個函式又呼叫了 UART_TEST_HANDLE()，為什麼不把 UART_TEST_HANDLE()的內容直接寫入，而是另外呼叫的理由是因為，中斷的大原則為不能在中斷裡面做不確定要花多久時間的事情，因此在 UART02_IRQHandler()裡是放重要的中斷，是一定要連續做完，不能被打擾，而另外比較不重要、可被打擾的中斷就把它的 function 寫在外面。

```
106  void UART02_IRQHandler(void)
107  {
108      UART_TEST_HANDLE();
109  }
110
```

◂圖 1.3

在下圖 1.4 中的 115 行用 u32IntSts=UART->ISR 作為目前中斷狀態的暫存器，在第 126 行的 if 判斷式中，將目前中斷狀態與 Msk 做交集運算，判斷是否為 RDA(Receive Data Available Interrupt Indicator (Read Only))中斷，也就是判斷鍵盤是否有按下按鍵，在 130 行中代表如果讀到 0，則 g_bWait=FALSE，將使得程式跳出上圖 1.2 中第 197 行的 while 迴圈，執行第 200 行的 UART_DisableInt 來關掉中斷。

```
112  void UART_TEST_HANDLE()
113  {
114      uint8_t u8InChar = 0xFF;
115      uint32_t u32IntSts = UART0->ISR; //ask
116      uint8_t *txt=g_u8RecData;
117
118      char RGB_LED[6][20];
119      strcpy(RGB_LED[0],"blue on");
120      strcpy(RGB_LED[1],"blue off");
121      strcpy(RGB_LED[2],"green on");
122      strcpy(RGB_LED[3],"green off");
123      strcpy(RGB_LED[4],"red on");
124      strcpy(RGB_LED[5],"red off");
125
126      if(u32IntSts & UART_ISR_RDA_INT_Msk) //
127      {
128          //if(!g_u8RecData[0]) printf("Input:");
129          u8InChar = UART_READ(UART0);
130          if(u8InChar == '0') //input 0 to end.
131            {
132                  g_bWait = FALSE;
133            }
```

▲圖 1.4

在下圖 1.5 的第 134~174 行，當按下 Enter 按鍵，也就是 u8InChar 讀到 0x0D 時，就會輸出字串，而控制 RGD_LED 的部分則是使用 strcmp 的函示來做判斷，因為 RGB_LED 為 low-active driven，0 但表 on，1 代表 off。

```
134      if(u8InChar==0X0D){
135          g_u8RecData[g_u32comRtail]='\0'; //end of string
136          printf("Input:%s\n",(char*)txt);
137
138          if(strcmp(RGB_LED[0],(char*)txt)==0)
139          {
140              PA12=0;
141          }
142          if(strcmp(RGB_LED[1],(char*)txt)==0)
143          {
144              PA12=1;
145          }
146          if(strcmp(RGB_LED[2],(char*)txt)==0)
147          {
148              PA13=0;
149          }
150          if(strcmp(RGB_LED[3],(char*)txt)==0)
151          {
152              PA13=1;
153          }
154          if(strcmp(RGB_LED[4],(char*)txt)==0)
155          {
156              PA14=0;
157          }
158          if(strcmp(RGB_LED[5],(char*)txt)==0)
159          {
160              PA14=1;
161          }
162          //reset
163          g_u8RecData[0]=0;
164          g_u32comRtail=0;
165          g_u32comRbytes=0;
166      }
167        else if(g_u32comRbytes < RXBUFSIZE)
168          {
169              g_u8RecData[g_u32comRtail] = u8InChar; //put the char in g_u8RecData
170              g_u32comRtail = (g_u32comRtail == (RXBUFSIZE - 1)) ? 0 : (g_u32comRtail + 1); //check oversize
171              g_u32comRbytes++;
172          }
173      }
174  }
```

▲圖 1.5

<心得與收穫>

這次的實驗對我來說又是一大挑戰，和同組組員一開始討論就遇到裝置管理員中，連接埠無法讀取、不支援 WIN11 的問題，上網查到需要舊版本的驅動器，處理了好久才終於可以正式開始進入實驗。在看完 pdf 後，當然是腦袋一片空白，上網查了許多 UART 的關鍵字後才終於理解一點點皮毛，最後嘗試了超過一個禮拜才終於把結果弄出來，但許多內容和函式仍然是一知半解，Demo 完後藉由助教得知，原來參雜了那麼多作業系統的知識，此外，雖然 Manual 裡面放了許多資訊，但我嘗試看了發現許多東西還是看不太明白，但是多多少少能夠理解大概的運作原理，接下來的 lab 我期許自己多去查看 Manual 裡面的內容，再更加努力嘗試搞懂 function 的功能，絕對能讓自己更加進步。