微處理機系統與介面技術 LAB 2

系所：電機 學號 :612415013 姓名：蕭宥羽

<實驗器材>

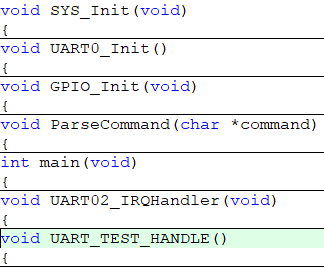
NUC 140 V2.0 開發板



<實驗過程與方法>

實驗要求 : 將鍵盤輸入的字印到putty上面，要在按下Enter時再把字印出來，並根據輸入的字串控制RGB LED燈。

MCU透過rx端接收pc鍵盤按下的字元，將資料存到MCU的記憶體中，按下enter後再將這筆字串與led命令去比較決定led的開與關，最後再將這串字串藉由tx端一個字元一個字元的發送到putty上。



SYS\_Init、UART0\_Init、GPIO\_Init 負責做初始化

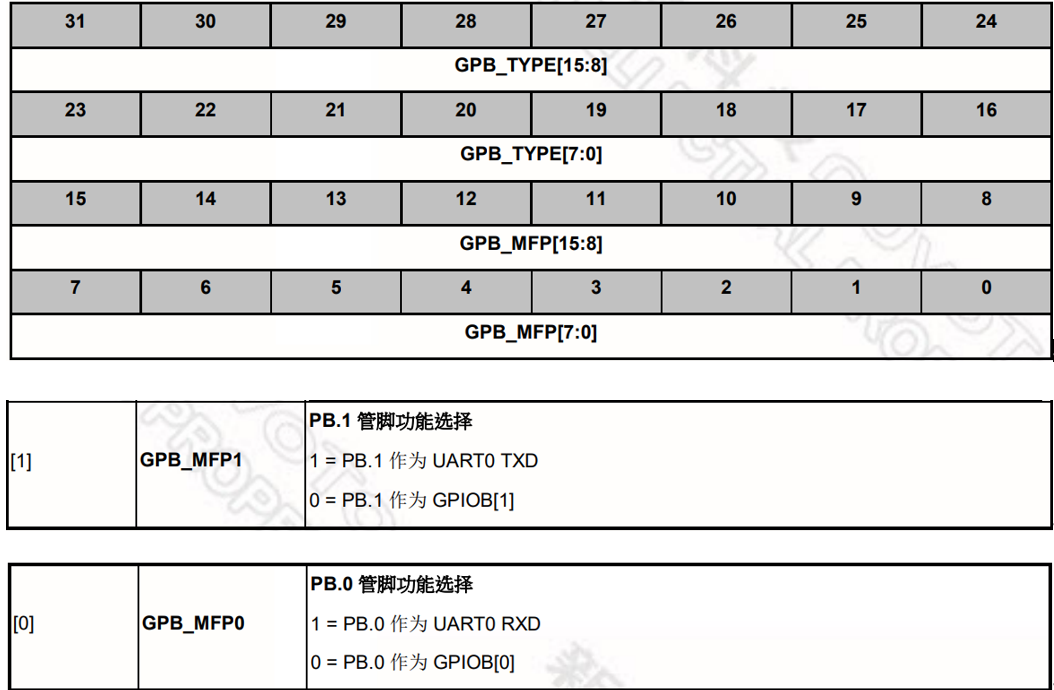
ParseCommand 負責比較輸入得字串是否符合led命令，並控制led開與關

UART02\_IRQHandler發生UART中斷時會執行裡面的程式碼

UART\_TEST\_HANDLE負責處理rx tx傳輸

<Mian function code>

1. SYS\_Init

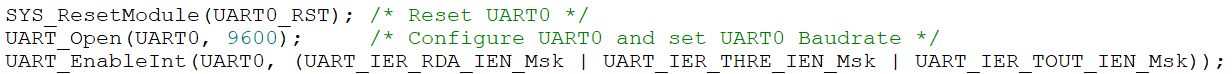




將GPB\_MFP暫存器的bit0 bit1設為1， PB1作為UART0 TXD，PB.0 作為 UART0 RXD

先保留其他位元的狀態 再將bit1 2 設為1

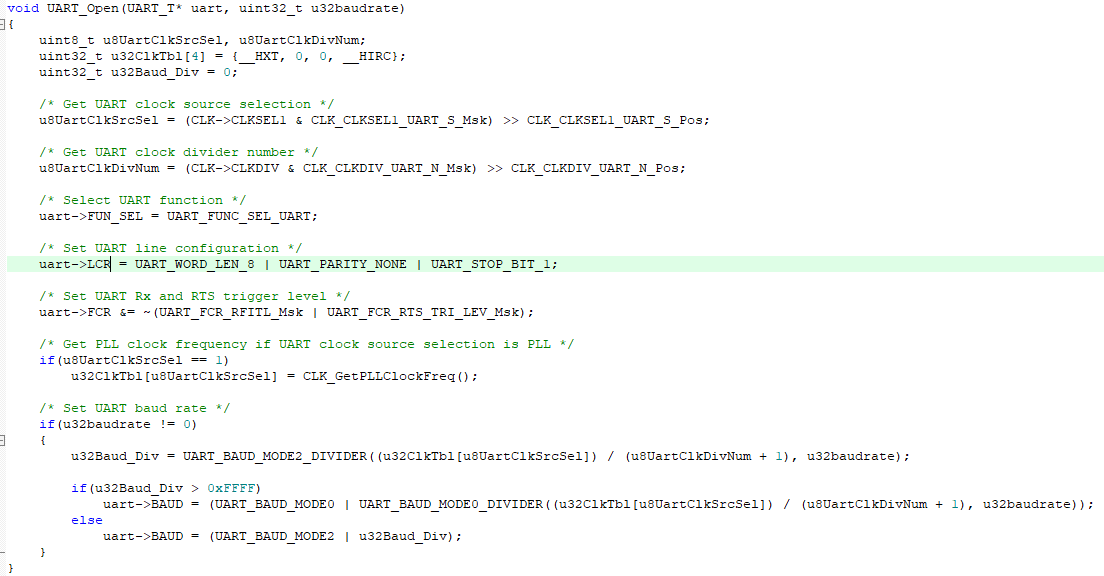
1. UART0\_Init



這邊對 uart 做初始化，設定baud rate、開啟中斷等等

* 1. UART\_Open(UART0, 9600);

將uart0的baud rate設為9600

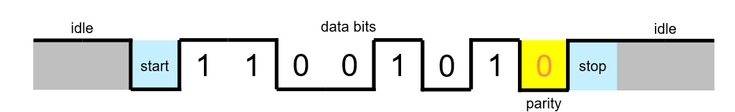


* Baud rate 的設置方式是將UART\_CLK進行分頻之後得到我們想要的傳輸速度。
* uart->FUN\_SEL = UART\_FUNC\_SEL\_UART;

將 UART 設置為標準 UART 模式，不啟用其他功能模式（如 IrDA 或 RS-485）。

* uart->LCR = UART\_WORD\_LEN\_8 | UART\_PARITY\_NONE | UART\_STOP\_BIT\_1;

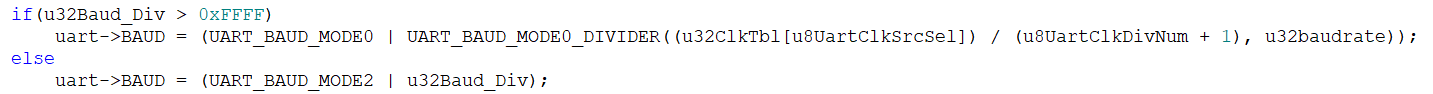
設置 UART 的線路參數，8 位元資料長度 (UART\_WORD\_LEN\_8)、無奇偶校驗 (UART\_PARITY\_NONE)、1 個停止位 (UART\_STOP\_BIT\_1)

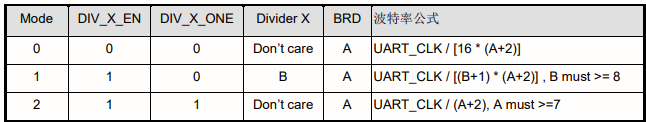


* 

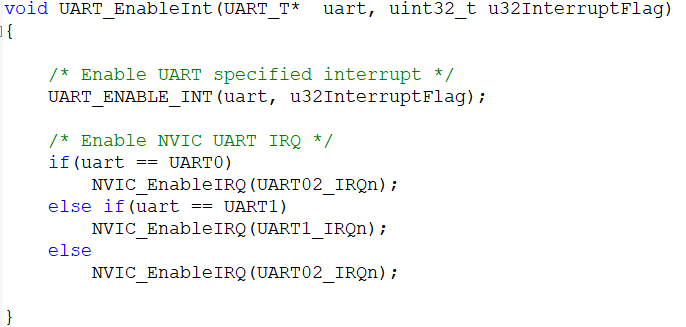
**計算波特率除數**，根據時鐘頻率和目標baud rate來計算 UART 的除數值。

* 將計算出的除數寫入 UART 的暫存器，確保 UART 能夠正確地以指定的波特率運行。

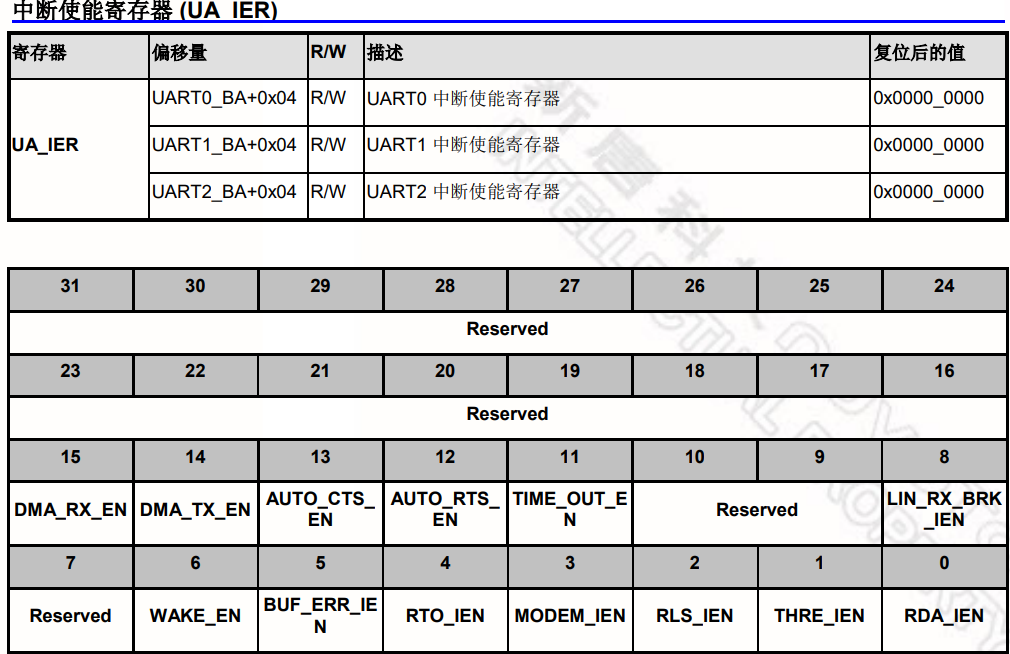




* 1. UART\_EnableInt (UART0, (UART\_IER\_RDA\_IEN\_Msk | UART\_IER\_THRE\_IEN\_Msk | UART\_IER\_TOUT\_IEN\_Msk));







將UA\_IER暫存器的bit1 bit0 bit11 設為1開啟



* TIME\_OUT\_EN

啟用接收資料的超時檢測，防止資料傳輸中斷。

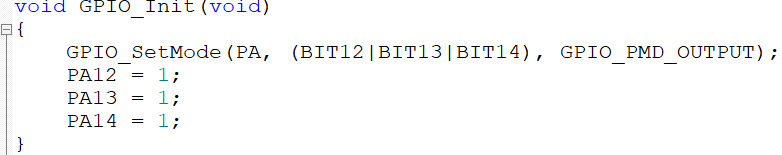
* THRE\_IEN

檢測 UART 的發送緩衝區是否空閒，通知系統可以發送新的資料。

* RDA\_IEN

檢測接收緩衝區是否有新資料可讀，通知系統讀取接收到的資料。

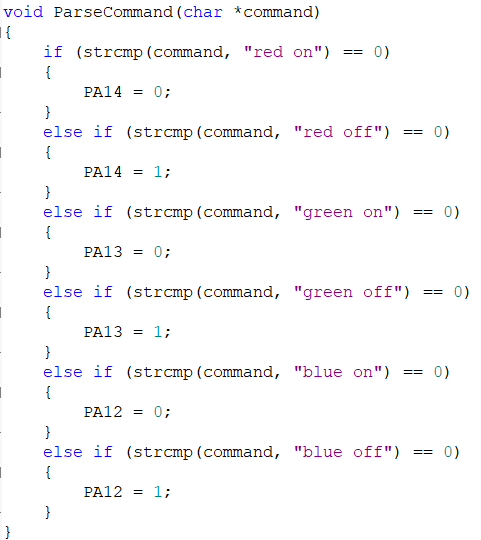
1. GPIO\_Init





啟用LED，GPA12-14打開，並將初始值設為1(關)

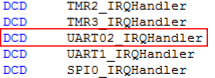
1. ParseCommand



char \*command：這是一個字串，代表要解析的命令，如 "red on"、"green off" 等。

strcmp(command, "red on") == 0：使用 strcmp 函數來比較傳入的命令字串與具體的命令，如 "red on"。如果相等，表示此命令被觸發。最後對應不同顏色的命令，函數會對控制 LED 的 GPIO 進行操作。

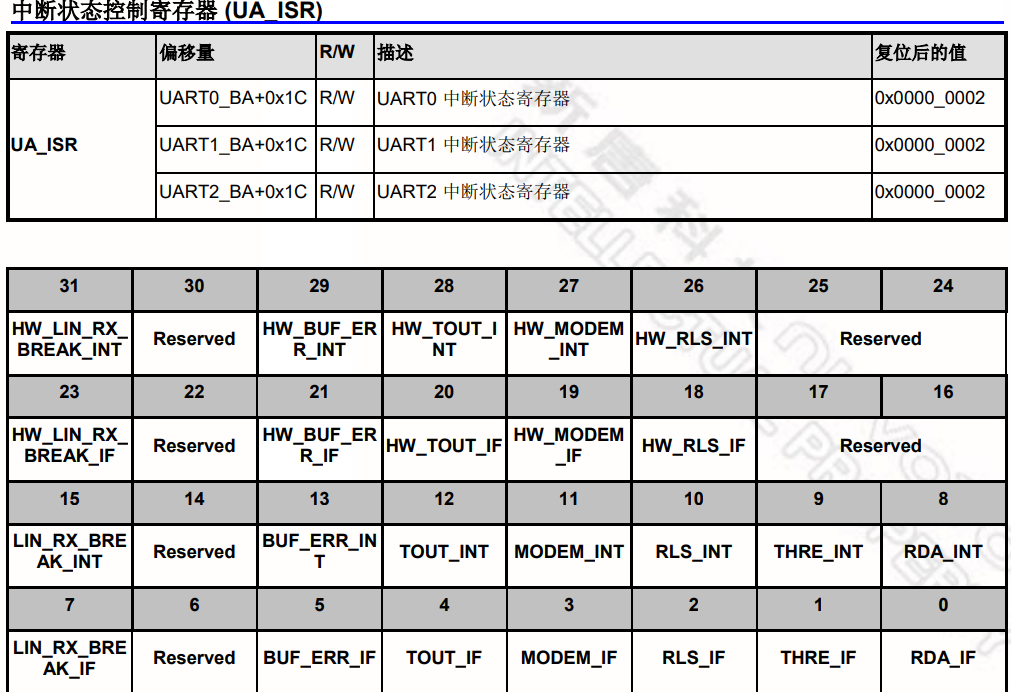
1. UART02\_IRQHandler



DCD 表示「定義代碼指令」，用來將每個硬體中斷源與它的處理函數（IRQHandler）相關聯。每當發生中斷時，這個向量表會引導程序跳轉到對應的中斷處理函數。

UART02\_IRQHandler，表示當 UART0 或 UART2 觸發中斷時，系統會跳轉到 此函數處理該中斷。

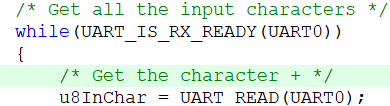
1. UART\_TEST\_HANDLE



* u8InChar：用來存儲接收到的字元。
* u32IntSts：儲存UART中斷狀態暫存器的值（UART0->ISR），用來判斷當前的中斷類型。
* 中斷部分:本次實驗共使用到兩個中斷

1. 

檢測是否發生了RDA中斷。如果發生了接收中斷，表示 UART 接收緩衝區有資料準備好，可以讀取。

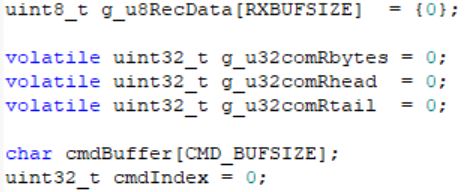


UART\_IS\_RX\_READY 確認 UART 是否有資料可讀，如果有資料，則進入迴圈並從 UART 讀取一個字元，並將其存入 u8InChar。

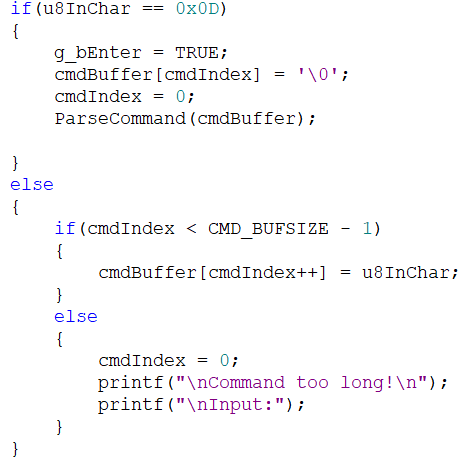
1. 

用來檢測是否發生了THRE(發送保持暫存器空中斷)的標誌。表示 UART 的發送緩衝區已經空了，可以接受新的資料進行傳輸。

* 在程式最前面有設置兩個buffer，g\_u8RecDate(size:1024)、cmdBuffer(size: 64)



* cmdBuffer : 存放接收到的資料，用於做字串比較

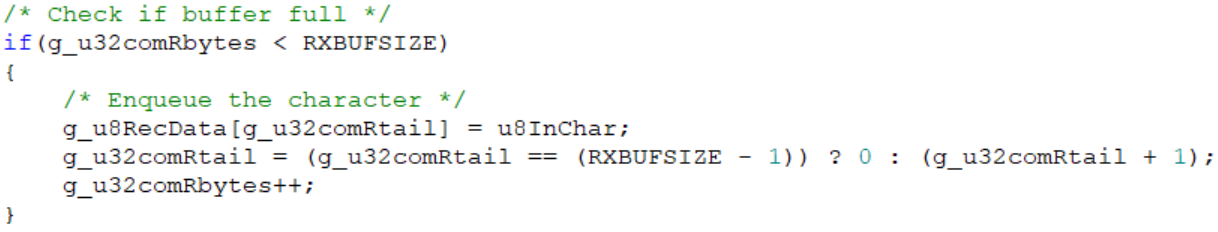


當enter沒被按下的時候，將rx接收到的字元存到cmdBuffer中，如果超過buffer容量的時候，會將index指回0(清空的作用)；當enter被按下的時候，先將g\_bEnter設成true，並設定cmdBuffer[cmdIndex] = '\0'; 將字串的結尾設定為空字元，代表字串結束。

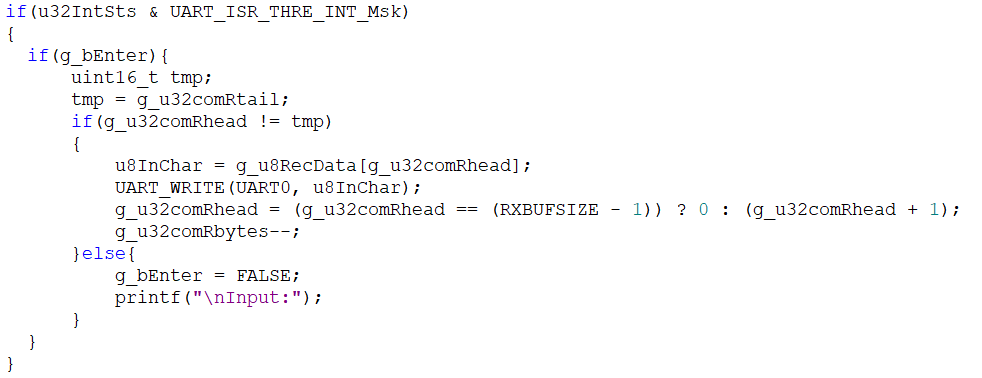
接著將cmdIndex設為0，讓下一次使用時可以用一個空的buffer

ParseCommand(cmdBuffer); 比較傳入的字串與led命令，進行LED的控制。

* g\_u8RecDate : 處理rx tx資料，用頭尾指標的方式去指向buffer的index，當接收到資料時會先將資料放到buffer[tail]的位置，並將tail+1以方便下一次放置資料；當要發送資料時，會將buffer[head]的內容由tx送出，當發送完後會將 head+1，下一次發送就會發送下一個位置的資料，如果tail==head的話代表接收到的資料已經都被發送完了，另外當tail head要超過BUFSIZE時會將其設為0，重新開始新的一圈。
  + - Rx



* + - Tx : 這邊設置當g\_bEbter==true時才會開始發送



* 經過這樣的流程之後，就可以達成鍵盤輸入的字印到putty上面，要在按下Enter時再把字印出來，並根據輸入的字串控制RGB LED燈

<心得與收穫>