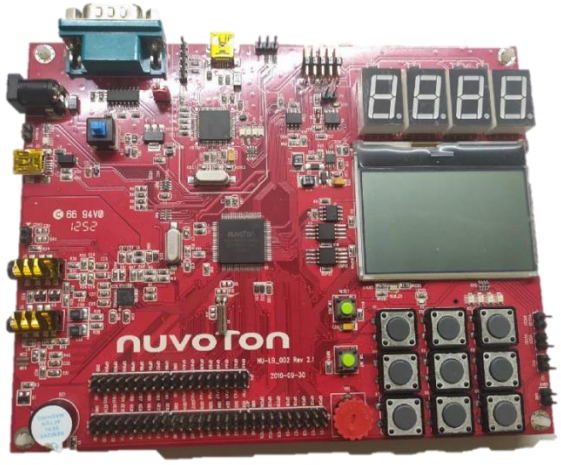
嵌入式作業系統 LAB 1

系所：通訊四　　學號 :409430030　　姓名:翁佳煌

<實驗器材及環境>

NUC 140 開發板　　FreeRTOSv10.4.1



<實驗過程與方法>

下**圖 1** 的 14~23 行，程式開始時引入了必要的標頭檔。這些標頭檔通常包括系統標頭檔、FreeRTOS 相關標頭檔和自定義標頭檔，以及 PLL 運行時的時脈頻率，這將用於系統時鐘設定。

再來第 25~28 行，這一部分宣告了多個靜態函數，這些函數將作為 FreeRTOS 任務運行。

SYS_Init 以及 UART0_Init 都是做簡單的初始化動作，這在微處理的課程中多次提到，這邊就不再贅述。

```
13  *****************************************************
14  #include <stdio.h>
15  #include "NUC100Series.h"
16
17  #include "FreeRTOS.h"
18  #include "task.h"
19  #include "queue.h"
20  #include "croutine.h"
21
22  #define PLL_CLOCK           50000000
23
24
25  static void vTaskMsgPro1(void* pvParameters);
26  static void vTaskMsgPro2(void* pvParameters);
27  static void LEDControlGPC12(void* pvParameters);
28  static void LEDControlGPC13(void* pvParameters);
29
30  static TaskHandle_t xHandleTaskMsgPro1 = NULL;
31  static TaskHandle_t xHandleTaskMsgPro2 = NULL;
32
33  void vStartThreadTasks( void );
34
35
36  /* Function prototype declaration */
37  void SYS_Init(void);
38  void UART0_Init(void);
39
```

▲圖 1

下**圖 2** 的部分是因為 Bonus 指定要控制 LED 閃爍，因此這個函數是用來初始化嵌入式系統中的 GPIO（通用輸出/輸入）引腳，以確保這些引腳的正確操作模式。在這個函數中，使用了 GPIO_SetMode 函數，來設置多個引腳（PC12、PC13、PC14 和 PC15）的操作模式為 GPIO_PMD_OUTPUT，這意味著它們將被配置為數字輸出引腳，以便後續的 LED 控制。

```
52  void GPIO_Init()
53  {
54      GPIO_SetMode(PC, BIT12, GPIO_PMD_OUTPUT);
55      GPIO_SetMode(PC, BIT13, GPIO_PMD_OUTPUT);
56      GPIO_SetMode(PC, BIT14, GPIO_PMD_OUTPUT);
57      GPIO_SetMode(PC, BIT15, GPIO_PMD_OUTPUT);
58  }
```

▲圖 2

接下來看到下**圖 3** 的 138~157 行的 main 主程式，**SYS_UnlockReg()**是用來解除保護註冊的鎖定，以便後續的設定和初始化操作能夠存取需要的系統註冊。在某些嵌入式系統中，某些系統註冊可能會被鎖定，以防止不必要的更改。**SYS_Init()**用於初始化嵌入式系統的主要設定，包括系統時脈設定、外部硬體設定等。這個函數的目的是設定系統的運行環境，以確保系統能夠正確運行。**SYS_LockReg()**這一行程式碼是用來重新鎖定保護註冊，以確保系統在進行運行時不會被不必要地修改。這通常在系統初始化後調用，以提高系統的安全性。**GPIO_Init()**在上**圖 2** 中有提到，用於初始化 GPIO（通用輸出/輸入）引腳的設定。

再來看到第 152 行的 **vStartThreadTasks()**，用於初始化和創建多個執行緒（task）或任務。這些執行緒將在作業系統中運行，執行不同的任務。這些執行緒是使用 FreeRTOS 函數庫創建的。

第 153 行的 **vTaskStartScheduler()**，這是 FreeRTOS 函數庫的一個函數，用於啟動作業系統的調度器。一旦調度器啟動，將開始執行各個執行緒，並按照其優先級執行。

```
138  int main(void)
139  {
140      /* Unlock protected registers */
141      SYS_UnlockReg();
142      /* Init system, IP clock and multi-function I/O. */
143      SYS_Init();
144      /* Lock protected registers */
145      SYS_LockReg();
146
147          GPIO_Init();  // LED initial
148
149          UART0_Init();
150
151          printf("test\r\n");
152          vStartThreadTasks();
153          vTaskStartScheduler();
154
155          while(1);
156
157  }
```

▲圖 3

下圖 4 的 160~166 行，是使用 FreeRTOS 函數庫創建和管理多個執行緒（task）或任務，以實現並行執行不同的任務。這裡創建了四個不同的執行緒，分別為 vTaskMsgPro1、vTaskMsgPro2、LEDControlGPC12 和 LEDControlGPC13。這四個執行緒將在相同的優先級下運行。

xTaskCreate()會創建一個新的執行緒，並指定其執行的函數（例如 vTaskMsgPro1）、名稱、stack 深度、參數、Priority 以及一個指向執行緒的指標。

```
160  void vStartThreadTasks( void )
161  {
162          xTaskCreate(vTaskMsgPro1, "vTaskMsgPro1",512,NULL,2,( xTaskHandle * ) NULL );  /
163          xTaskCreate(vTaskMsgPro2, "vTaskMsgPro2",512,NULL,2,( xTaskHandle * ) NULL );
164          xTaskCreate(LEDControlGPC12, "LEDControl1",128,NULL,2,( xTaskHandle * ) NULL );
165          xTaskCreate(LEDControlGPC13, "LEDControl1",128,NULL,2,( xTaskHandle * ) NULL );
166  }
167
```

```
• BaseType_t xTaskCreate
(   TaskFunction_t pxTaskCode,
    const char * const pcName,
    const uint16_t usStackDepth,
    void * const pvParameters,
    UBaseType_t uxPriority,
    TaskHandle_t * const pxCreatedTask )
```

▲圖 4

下**圖 5** 的 168~184 行是實作 LAB1 中 Basic 的重要部分，vTaskMsgPro1 和 vTaskMsgPro2 是執行緒函數，不斷地執行一個無窮迴圈，分別每次都會輸出 "Task1 -> 500ms" 以及 "Task2 -> 1000ms"，然後使用 vTaskDelay 函數使其暫停 500ms 和 1000ms。

```
168  static void vTaskMsgPro1(void* pvParameters)
169  {
170          while(1)
171          {
172              printf("Task1 -> 500ms\r\n");
173              vTaskDelay(500);
174          }
175  }
176
177  static void vTaskMsgPro2(void* pvParameters)
178  {
179          while(1)
180          {
181              printf("Task2 -> 1000ms\r\n");
182              vTaskDelay(1000);
183          }
184  }
```

▲圖 5

下**圖 6** 的 186~207 行，是 Bonus 控制 LED 的主要部分，LEDControlGPC12 和 LEDControlGPC13 執行緒分別控制 PC12 以及 PC13 引腳，分別為每隔 100ms 和 1000ms 切換引腳狀態，即 ON 和 OFF，達到 LED 閃爍的效果。

```
186  static void LEDControlGPC12(void* pvParameters)
187  {
188          while(1)
189          {
190              //printf("GPC12 ON every 100ms\r\n");
191              PC12=0;
192              vTaskDelay(100);
193              PC12=1;
194              vTaskDelay(100);
195          }
196  }
197  static void LEDControlGPC13(void* pvParameters)
198  {
199          while(1)
200          {
201              //printf("GPC13 ON every 1000ms\r\n");
202              PC13=0;
203              vTaskDelay(1000);
204              PC13=1;
205              vTaskDelay(1000);
206          }
207  }
```

▲圖 6

<心得與收穫>

由於之前修過微處理與介面系統設計和作業系統相關課程，這次的 LAB1 在 Code 方面並未遇到太大的困難。主要問題是一開始建環境時遇到一些困難，感謝助教願意撥出時間指導我，解決了我在環境配置方面的問題，這使我能夠順利完成本次 LAB。希望在未來的學習和實驗中繼續積累更多的知識和經驗，並不斷提升自己的技能。