MCU Project

製件者:JetWen

目錄

[關於此文件 3](#_Toc160713252)

[軟體開發實驗 5](#_Toc160713253)

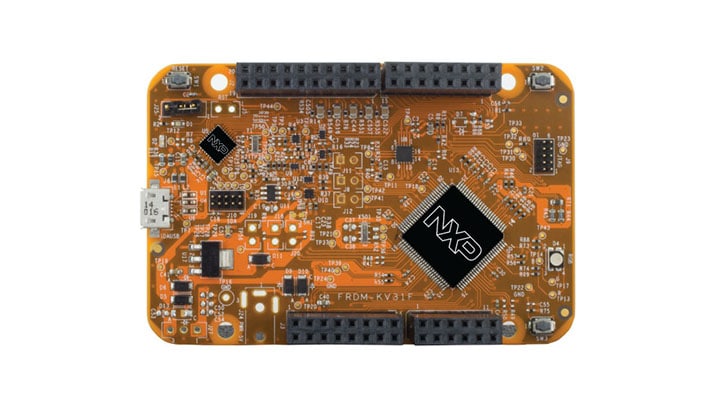
[專案項目 6](#_Toc160713254)

# 關於此文件

1. 目的:

以學習MCU 為主，以利未來專案開發。

1. 學習開發版為FRDM-KV31(如圖一)



(圖一:FRDM-KV31)

1. 硬體架構(如下表一):

(表一)

|  |  |
| --- | --- |
| **Feature Category** | **Feature Description** |
| Microcontroller | KV31F512VLL12 MCU: Arm® Cortex®-M4, 120 MHz, 512 kB flash, 96 kB SRAM |
| Analog-Digital Converters | 2x 16-bit ADCs |
| Timers/PWM | 4x FlexTimers/PWM up to 20-channels with quadrature decoder dedicated to motor and power control, |
| plus 2x 12-bit DACs |
| Connectivity | Motor control auxiliary connector, form factor compatible with Arduino® Rev3 pin layout |
| Power Management | Power selectable between 3.3 V/1.8 V, |
| flexible power supply options – USB or external source |
| Debugging | OpenSDA debug interface, Mass storage device flash programming interface (default), |
| MBED Debug interface, CMSIS-DAP interface |
| Sensors | Six-axis sensor combining accelerometer and magnetometer (FXOS8700CQ), |
| Thermistor sensor for measuring temperature |
| User Components | Tri-color user-controllable LEDs, |
| Two (2) user push-button switches for NMI interrupts and LLWU wake-up (SW2/SW3) |

# 軟體開發實驗

1. 概要:

本章將介紹開發環境及專案目的

1. 開發環境及軟體架構(如表一)

(表二)

|  |  |
| --- | --- |
| 項目 | 內容 |
| 開發環境 | Linux Ubuntu |
| IDE | VS Code |
| Debugger | OpenSDA及J-Link |

1. 專案目的:
   1. 了解MCU運作流程(如:啟動及中斷向量表)
   2. 了解FreeRTOS (如:Task/Queue/Semaphore…etc.)
   3. 驅動程式開發(如:SPI/UART/I2C…etc.)
   4. 學會使用Git和GitHub版本管理

## 專案項目

1. 概要:

本章將詳細介紹專案項目及各項細節。

1. 開發專案一欄表(如:表三):

(表三)

|  |  |  |
| --- | --- | --- |
| 項目編號 | 項目名稱 | 內容 |
| A | FreeRTOS\_hello | 驅動UART，打印Hello world |
| FreeRTOS\_LED | 驅動GPIO，控制三色Led燈亮滅 |
| B | Task Priority調整 | 理解vTaskSuspend(),vTaskDelay(),taskYEILD() |
| C | Semaphore使用 | 理解Semaphore及Mutex的使用 |
| D | Queue的使用 | 理解Task之間參數改 |
| E | SoftwareTimer的使用 | 理解SoftwareTimer定時呼叫Task的功能 |

1. 專案內容:
   1. FreeRTOS\_hello及FreeRTOS\_LEDs:

先Create一個Task用來驅動UART並輸出字元到終端機。

再Create 三個Task用以驅動GPIO點亮R, G, B三色小燈

* 1. Task Priority調整:

再A專案的基礎上，加入vTaskSuspend(), vTaskDelay(), taskYEILD()，來幫助理解Task之間的運行。

* 1. Semaphore使用:

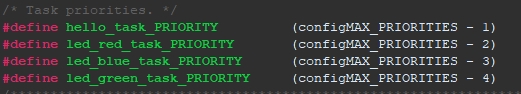
再B專案的基礎上，加入xSemaphoreTake()及xSemaphoreGive()函式

* 1. Queue的使用:創建Queue修改global variable.
  2. Software Timer:建立Software Timer執行定時呼叫Task的功能。

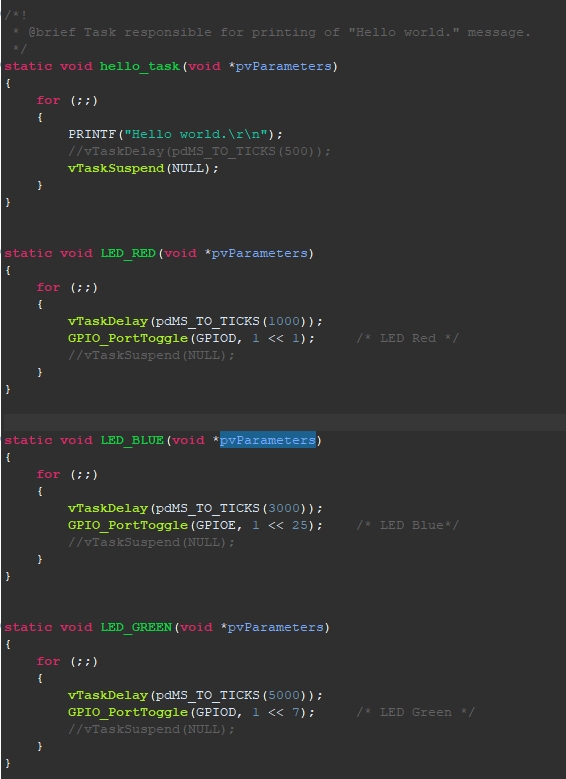
1. 程式解析
2. (實驗一):
   1. 單純建立四個Task，hello\_task(),

LED\_RED(), LED\_BLUE(), LED\_GREEN()

* 1. Task之間與優先級關係如下圖A.1圖A.2所示。



(圖A.1)



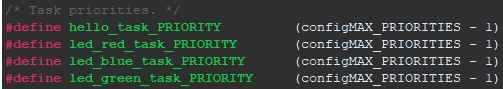
(圖A.2)

* 1. 說明:

因為hello\_task()的Priority最高所以透過vTaskSuspend()來執行LED三色小燈亮滅。

再透過vTaskDelay(),使LED R, G, B三色出現不同顏色組合閃爍。

1. (實驗二):
   1. 延續實驗一，新增taskYIELD ()函式及輸出字串，並將Priority設置相同去理解task的交替執行。
   2. Task與優先級關係如下圖B.1圖B.2所示。



(圖B.1)



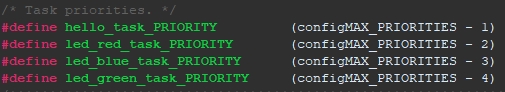
(圖B.2)

* 1. 說明:

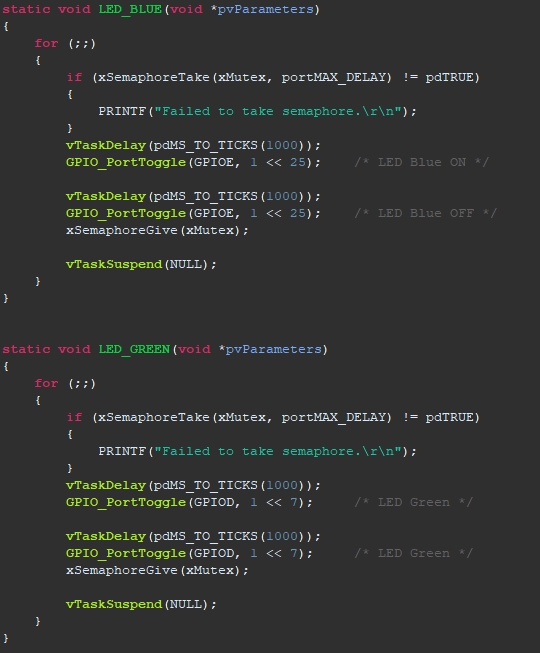
因為每個Task都有taskYIELD()函式，所以task執行結束後會交給下一個task去執行。

輸出結果可以看到終端機成功輸出各個顏色小燈對應的字串但是LED卻顯示白色小燈，這是因為執行速度太快導致肉眼無法辨別，實際上Task是有交替執行的。

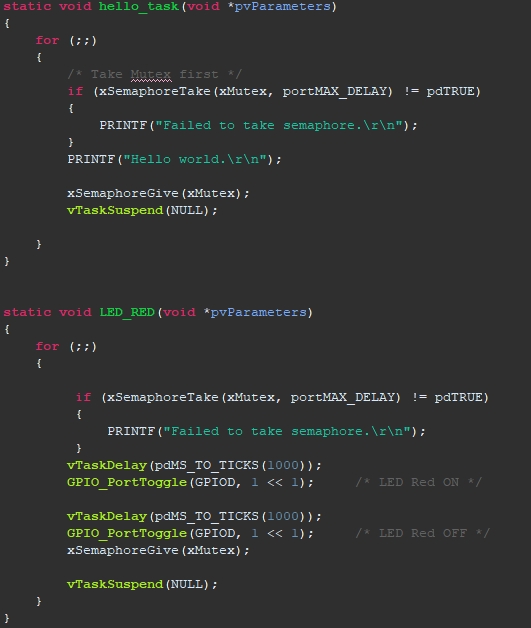
1. (實驗三)
   1. 延續實驗二，新增xSemaphoreTake()及 xSemaphoreGive()函式理解Semaphore用法。
   2. Task與優先級關係如下圖C.1,圖C.2, 圖C.3所示。



(圖C.1)



(圖C.2)



(圖C.3)

* 1. 說明:

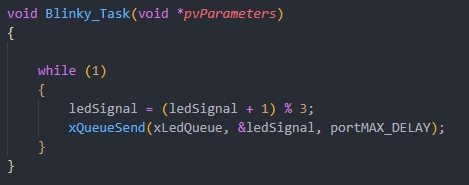
因為Mutex take 後其他Task不可以access CPU必須等到Mutex Give後才可以換下一個Task執行，因此hello\_task打印出hello之後換三個小燈以R -> G -> B順序1秒執行亮滅。

1. (實驗四):Queue
   1. 建立一個Task，Blinky\_task()函式控制R, G, B三色小燈閃爍

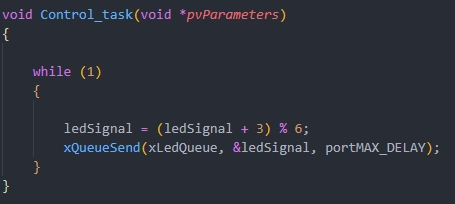
(如圖D.1)

* 1. 建立一個Task，Control\_task()函式控制組合燈閃爍

(如圖D.2)

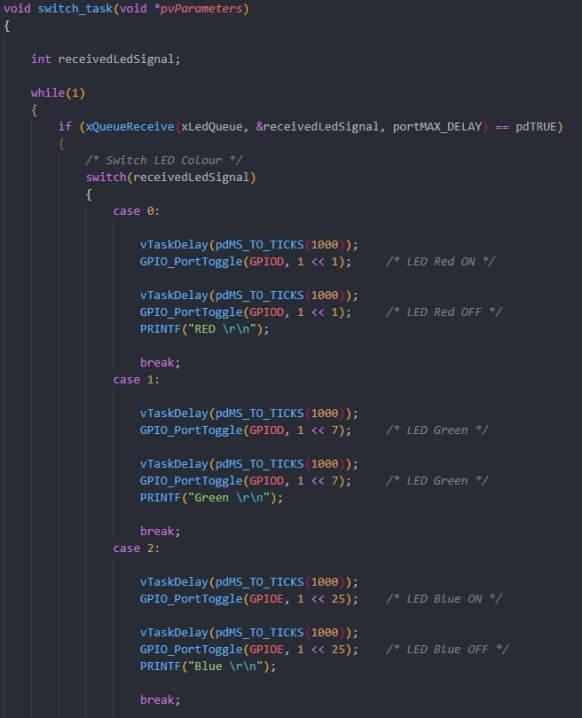


(圖D.1)



(圖D.2)

* 1. 創建一個Task，switch\_task()函式執行RGB三色燈閃爍切換以及組合閃爍切換(如圖D.3和D.4)。



(圖D.3)

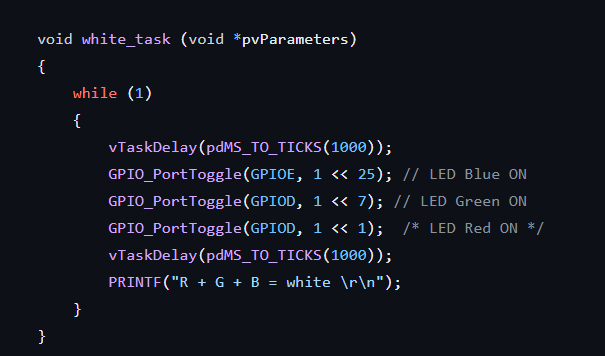


(圖D.4)

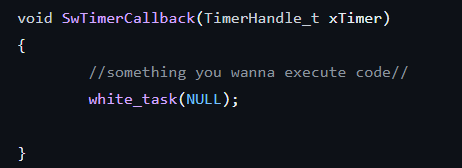
* 1. 說明:

利用Blinky\_task()函式利用Queue修改ledSiganl參數控制RGB閃爍，再利用Control\_task()函式利用Queue修改ledSiganl參數組合燈閃爍，最後switch\_task()函式接收Queue中ledSignal的值，並執行閃爍對應顏色小燈。

1. (實驗五):Software Timer
2. 先創建一個Task同時點亮白光(如圖E.1)，再創建一個Software Timer用做計時效果(如圖E.2)。
3. 設定計時器每三秒閃爍一次。



(圖E.1)



(圖E.2)

1. 說明:

利用Software Timer於固定時間(3秒)去呼叫white\_task去執行閃爍白燈的task。