MIAT-STM32嵌入式作業系統uC/OS II移植

WU-YANG

Technology Co., Ltd.

Declared Version

Training Only	
Declare	
Document Version	1.00
Release Date	2009.06.20
Document Title	MIAT-STM32嵌入式作業系統uCOS II移植
Exercise Time	■ Lecture 25 minutes ■ Operating 25 minutes
Platform	■ MIAT_STM32 ■ MIAT_IOB
Peripheral	LCD, LED
Author	■ WU-YANG Technology Co., Ltd.

WU-YANG

Technology Co., Ltd.

實驗目的

- 利用MicroC/OS II多工執行的能力執行多個應用程式
- 了解如何在MicroC/OS II中加入工作

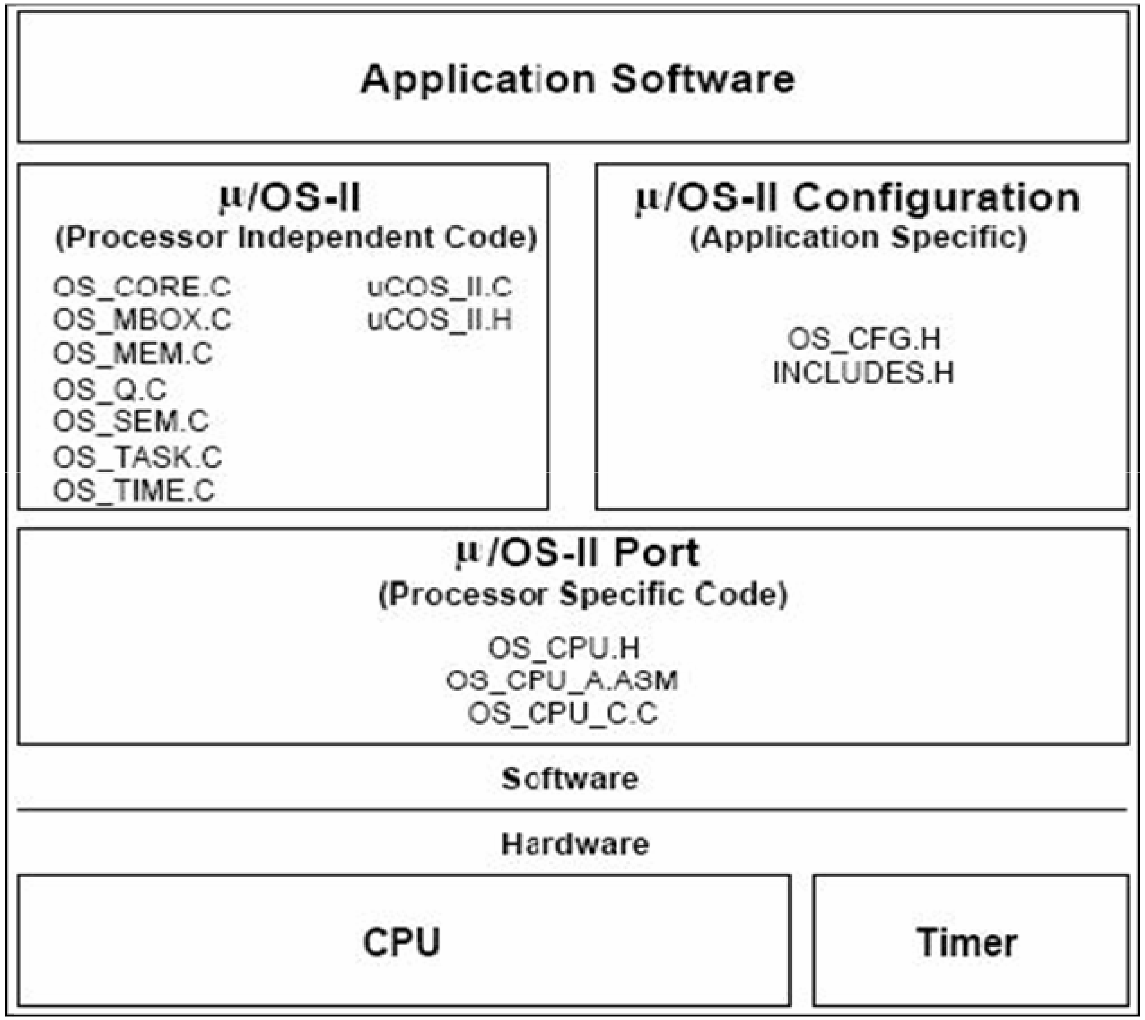
實驗原理

- MicroC/OS II是以多重工作處理為核心的即時多工作業系統
- 具備可移植性、唯讀儲存性、可規劃性及程式碼小等特點
- 官方網站(http://www.micrium.com/)提供多種硬體平台上的移植範例程式

MicroC/OS-II架構

- MicroC/OS-II在架構上主要包含兩個部分的程式碼：
  - 與處理器相依的程式碼(processor specific code)
    - 提供內容交換(context switch)時工作狀態的儲存和回復，以及設定系統時脈(clock tick)相關的機制
  - 與處理器無關的程式碼(processor independent code)等部分
    - 提供其他的作業系統服務，包括：排班(scheduling)、工作之間的通訊(inter-task communication)實作等等

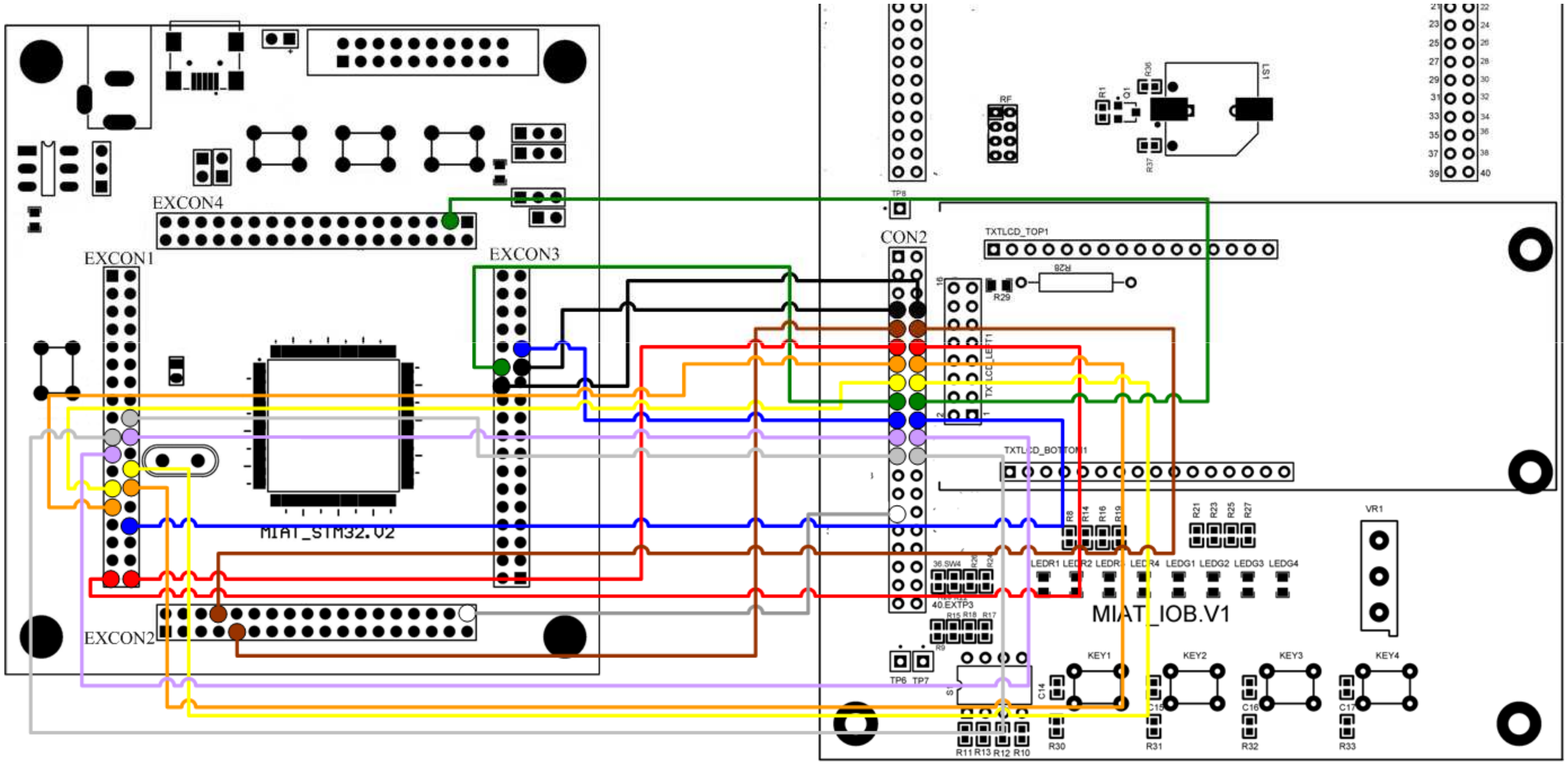
MicroC/OS-II架構圖



硬體電路配置

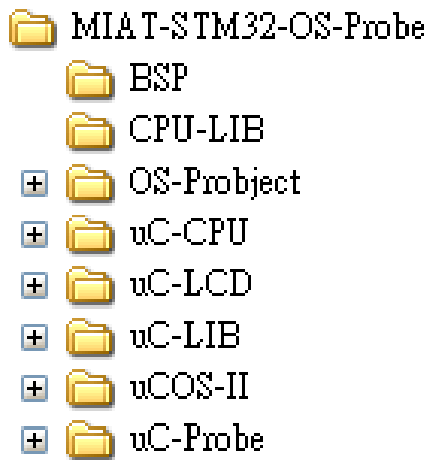
子版腳位名稱	子版腳位編號	母版腳位名稱	母版腳位編號
VCC5V	CON2.11	VCC5V	EXCON1.36
GND	CON2.12	GND	EXCON1.35
LCD_D0	CON2.16	PC0	EXCON1.24
LCD_D1	CON2.15	PC1	EXCON1.25
LCD_D2	CON2.14	PC2	EXCON1.26
LCD_D3	CON2.13	PC3	EXCON1.27
LCD_D4	CON2.8	PC4	EXCON2.8
LCD_D5	CON2.7	PC5	EXCON2.9
LCD_D6	CON2.6	PC6	EXCON3.24
LCD_D7	CON2.5	PC7	EXCON3.25
LCD_EN	CON2.15	PC8	EXCON3.26
LCD_RS	CON2.17	PC9	EXCON3.27
LCD_R/W	CON2.18	PC10	EXCON4.3
LED_G1	CON2.23	PF6	EXCON1.18
LED_G2	CON2.24	PF7	EXCON1.19
LED_G3	CON2.21	PF8	EXCON1.20
LED_G4	CON2.22	PF9	EXCON1.21

硬體電路配置

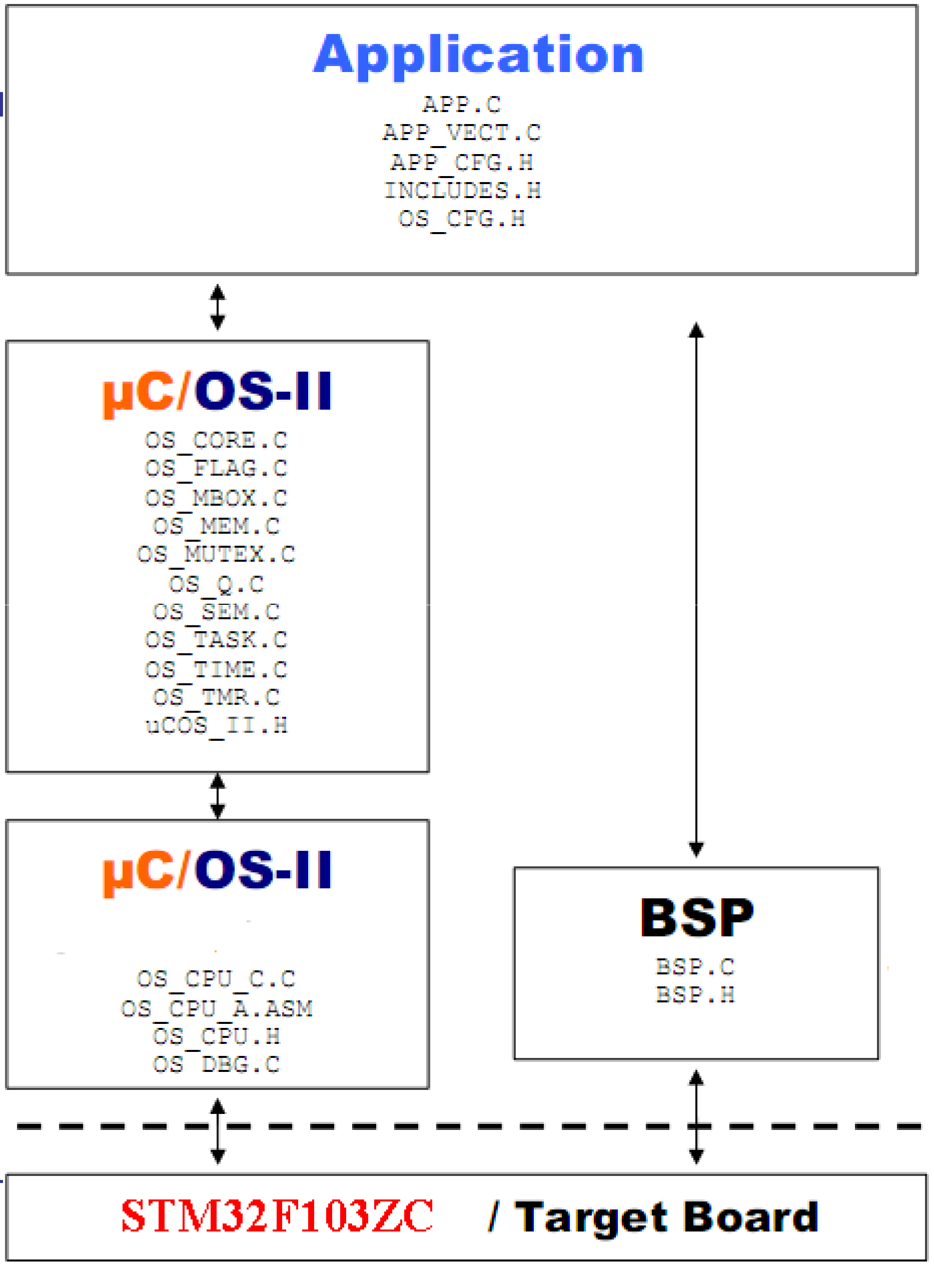

EXCON4
EXCON1
EXCON3
EXCON2
MIAT_STM32.V2
CON2
TXTLCD_TOP1
TXTLCD_BOTTOM1
TXTLCD_LEFT1
R28
R29
RF
LS1
Q1
R36
R37
TP8
TP6
TP7
VR1
MIAT_IOB.V1
LEDR1
LEDR2
LEDR3
LEDR4
LEDG1
LEDG2
LEDG3
LEDG4
KEY1
KEY2
KEY3
KEY4
R30
R31
R32
R33
R11 R13 R12 R10
36.SW4
40.EXTP3

實驗說明

程式架構



- MIAT-STM32-OS-Probe
  - BSP
  - CPU-LIB
  - OS-Probject
  - uC-CPU
  - uC-LCD
  - uC-LIB
  - uCOS-II
  - uC-Probe



加入工作

```c
int  main (void)
{
    CPU_INT08U  os_err;

    BSP_IntDisAll();                                          /* Disable all ints until we are ready to accept them.  */

    OSInit();                                                 /* Initialize "uC/OS-II, The Real-Time Kernel".         */

    os_err = OSTaskCreateExt((void (*)(void *)) App_TaskStart,  /* Create the start task.                             */
                            (void          * ) 0,
                             (OS_STK         * )&App_TaskStartStk[APP_TASK_START_STK_SIZE - 1],
                             (INT8U            ) APP_TASK_START_PRIO,
                             (INT16U           ) APP_TASK_START_PRIO,
                             (OS_STK         * )&App_TaskStartStk[0],
                             (INT32U           ) APP_TASK_START_STK_SIZE,
                             (void           * )0,
                             (INT16U           )(OS_TASK_OPT_STK_CLR | OS_TASK_OPT_STK_CHK));

#if (OS_TASK_NAME_SIZE >= 11)
    OSTaskNameSet(APP_TASK_START_PRIO, (CPU_INT08U *)"Start Task", &os_err);
#endif

    OSStart();                                                /* Start multitasking (i.e. give control to uC/OS-II).  */

    return (0);
}
```

建立工作

- **OSTaskCreateExt(**

```
void(*task)(void *pd),
void        *pdata,
OS_STK      *ptos,
INT8U       prio,
INT16U      id,
OS_STK      *pbos,
INT32U      stk_size,
void        *pext,
INT16U      *opt
)
```

- OSTaskCreateExt()用來建立一個新的工作，建立工作可以在多工環境開始之前或正在執行的工作中建立，但不能從中斷服務常式裡建立。

- task　指向工作程式碼的指標
- pdata 一個指標，指向一個類型可以選擇的資料區，該資料區的資料用在建立工作時與工作有關的參數，pdata用在向建立的工作傳遞參數
- ptos　指向工作堆疊頂端的指標

  OS_STK_GROWTH (OS_CPU.H)設為1時，堆疊由高位址向低位址遞減，ptos應指向工作堆疊的最高位址；

  OS_STK_GROWTH設為0時，堆疊由高位址向低位址遞增，ptos應指向工作堆疊的最低位址

- prio　工作優先權，數字越小權限越高
- id　　工作標誌，無實際用途，可把id與prio設為相同
- pbos 指向工作堆疊底端的指標

  OS_STK_GROWTH設為1時，ptos應指向工作堆疊的最低位址；

  OS_STK_GROWTH設為0時， ptos應指向工作堆疊的最高位址

- stk_size 指定工作堆疊的大小
- pext 一個定義使用者資料結構的指標
- opt 與工作相關的操作號誌
  - OS_TASK_OPT_NONE
  - OS_TASK_OPT_STK_CHK 堆疊檢查
  - OS_TASK_OPT_STK_CLR 清空堆疊
  - OS_TASK_OPT_SAVE_FP 儲存浮點數暫存器的值(處理器需有浮點數硬體)

工作內容

- 工作函式中必須呼叫下列其中一個函式
  - OSMboxPend()
  - OSFlagPend()
  - OSMutexPend()
  - OSQPend()
  - OSSemPend()
  - OSTimeDly()
  - OSTimeDlyHMSM()
  - OSTaskSuspend()
  - OSTaskDel()

```
static  void  App_TaskStart (void *p_arg)
{
    CPU_INT32U  i;
    CPU_INT32U  j;

    (void)p_arg;

    BSP_Init();
    OS_CPU_SysTickInit();

    App_TaskCreate();

    while (DEF_TRUE) {
        for (j = 0; j < 4; j++) {
            for (i = 1; i <= 4; i++) {
                BSP_LED_On(i);
                OSTimeDlyHMSM(0, 0, 0, 100);
                BSP_LED_Off(i);
                OSTimeDlyHMSM(0, 0, 0, 50);
            }
            for (i = 4; i >= 1; i--) {
                BSP_LED_On(i);
                OSTimeDlyHMSM(0, 0, 0, 50);
                BSP_LED_Off(i);
                OSTimeDlyHMSM(0, 0, 0, 100);
            }
        }
        for (i = 0; i < 4; i++) {
            BSP_LED_On(0);
            OSTimeDlyHMSM(0, 0, 0, 200);
            BSP_LED_Off(0);
            OSTimeDlyHMSM(0, 0, 0, 200);
        }
    }
}
```

實驗目標

- 在MicroC/OS II移植程式中加入LCD與LED兩個工作
  - MicroC/OS II的STM32F103ZC版本可在http://www.micrium.com/st/STM32.html下載

實驗步驟

- 1.在bsp.h中加入hw_config.h與lcd_func.h

```
#include   <cpu.h>
#include   <lib_def.h>
#include   <lib_mem.h>
#include   <lib_str.h>

#include   <stm32f10x_conf.h>
#include   <stm32f10x_lib.h>

#include   <app_cfg.h>
#include   <lcd.h>
#include   <bsp.h>

#include <hw_config.h>
#include <lcd_func.h>

#include   <ucos_ii.h>

#if (APP_OS_PROBE_EN == DEF_ENABLED)
#include   <os_probe.h>
#endif
```

實驗步驟

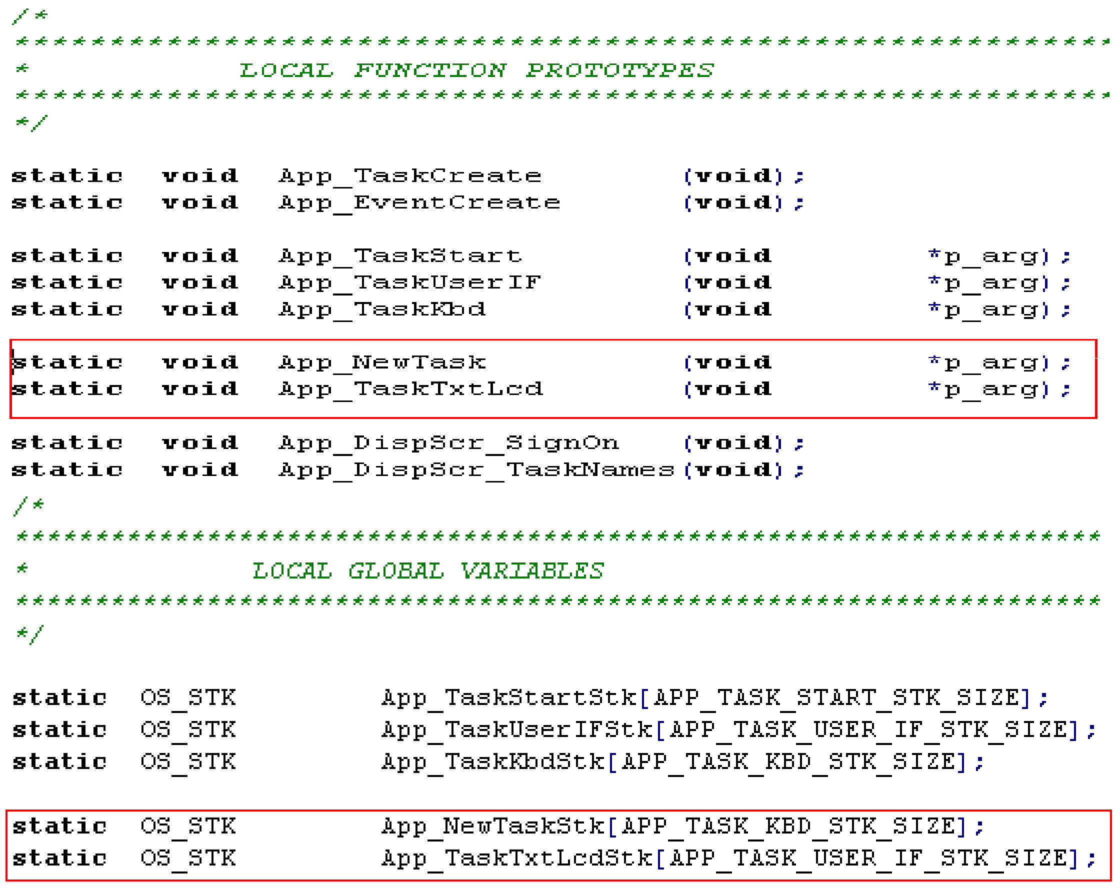
- 2.在app.c中撰寫工作要執行的內容

```
/*******************************************************
static  void  App_TaskTxtLcd(void *p_arg)
{
    CPU_INT32U  i;
    GPIO_Configuration();
    init_lcd();
    (void)p_arg;

  /* 開始顯示資料 */
  while(DEF_TRUE)
  {
     for(i=0;i<3;i++)
     {
        print(i,"MIAT_STM32");/* 顯示訊息1 */
        print(i+1,"HELLO world.....");/* 顯示訊息2 */
        prline1(15, '1');/* 第1行顯示字元 */
        prline2(15, '2');/* 第2行顯示字元 */
        OSTimeDlyHMSM(0, 0, 0, 500);
     }
  }
}
```

```
/**********************************************
static  void  App_NewTask (void *p_arg)
{
    CPU_INT32U  i;
    CPU_INT32U  j;

    (void)p_arg;
    BSP_Init();
    OS_CPU_SysTickInit();
    while (DEF_TRUE) {
        for (j = 0; j < 4; j++) {
            for (i = 1; i <= 4; i++) {
                BSP_LED_On(i);
                OSTimeDlyHMSM(0, 0, 0, 100);
            }

            for (i = 1; i <= 4; i++) {
                BSP_LED_Off(i);
                OSTimeDlyHMSM(0, 0, 0, 100);
            }
        }
    }
}
```

實驗步驟

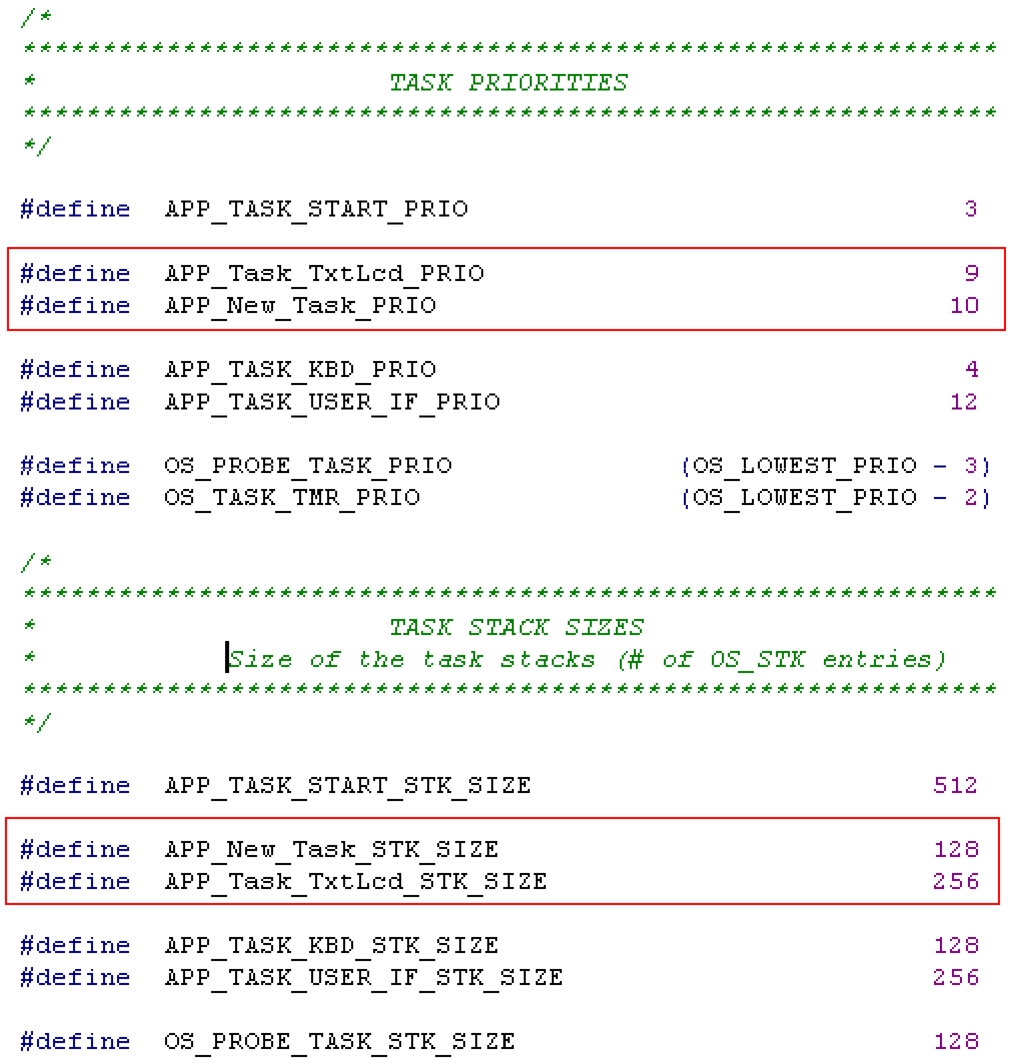
- 3.在app.c宣告工作函式與變數



```
/*
*********************************************************************
*              LOCAL FUNCTION PROTOTYPES
*********************************************************************
*/

static  void  App_TaskCreate           (void);
static  void  App_EventCreate          (void);

static  void  App_TaskStart            (void          *p_arg);
static  void  App_TaskUserIF           (void          *p_arg);
static  void  App_TaskKbd              (void          *p_arg);

static  void  App_NewTask              (void          *p_arg);
static  void  App_TaskTxtLcd           (void          *p_arg);

static  void  App_DispScr_SignOn       (void);
static  void  App_DispScr_TaskNames(void);
/*
*********************************************************************
*               LOCAL GLOBAL VARIABLES
*********************************************************************
*/

static  OS_STK          App_TaskStartStk[APP_TASK_START_STK_SIZE];
static  OS_STK          App_TaskUserIFStk[APP_TASK_USER_IF_STK_SIZE];
static  OS_STK          App_TaskKbdStk[APP_TASK_KBD_STK_SIZE];

static  OS_STK          App_NewTaskStk[APP_TASK_KBD_STK_SIZE];
static  OS_STK          App_TaskTxtLcdStk[APP_TASK_USER_IF_STK_SIZE];
```

實驗步驟

- 4.在app_cfg.h中設定工作優先權與工作堆疊大小



```
/*
*********************************************************************
*                         TASK PRIORITIES
*********************************************************************
*/

#define  APP_TASK_START_PRIO                              3

#define  APP_Task_TxtLcd_PRIO                             9
#define  APP_New_Task_PRIO                               10

#define  APP_TASK_KBD_PRIO                                4
#define  APP_TASK_USER_IF_PRIO                           12

#define  OS_PROBE_TASK_PRIO               (OS_LOWEST_PRIO - 3)
#define  OS_TASK_TMR_PRIO                 (OS_LOWEST_PRIO - 2)

/*
*********************************************************************
*                         TASK STACK SIZES
*             Size of the task stacks (# of OS_STK entries)
*********************************************************************
*/

#define  APP_TASK_START_STK_SIZE                        512

#define  APP_New_Task_STK_SIZE                          128
#define  APP_Task_TxtLcd_STK_SIZE                       256

#define  APP_TASK_KBD_STK_SIZE                          128
#define  APP_TASK_USER_IF_STK_SIZE                      256

#define  OS_PROBE_TASK_STK_SIZE                         128
```

實驗步驟

- 5.修改app.c中App_TaskStart()的內容，右圖為修改前，下圖為修改後

```
static  void  App_TaskStart (void *p_arg)
{
    CPU_INT32U  i;
    CPU_INT32U  j;
    (void)p_arg;

    BSP_Init();
    OS_CPU_SysTickInit();

#if (OS_TASK_STAT_EN > 0)
    OSStatInit();
#endif

    App_TaskCreate();

    while (DEF_TRUE) {
            OSTimeDlyHMSM(0, 0, 0, 200);
    }
}
```

```
static  void  App_TaskStart (void *p_arg)
{
    CPU_INT32U  i;
    CPU_INT32U  j;
    (void)p_arg;

    BSP_Init();
    OS_CPU_SysTickInit();

#if (OS_TASK_STAT_EN > 0)
    OSStatInit();
#endif

    App_TaskCreate();

    while (DEF_TRUE) {
        for (j = 0; j < 4; j++) {
            for (i = 1; i <= 4; i++) {
                BSP_LED_On(i);
                OSTimeDlyHMSM(0, 0, 0, 100);
                BSP_LED_Off(i);
                OSTimeDlyHMSM(0, 0, 0, 50);
            }
            for (i = 4; i >= 1; i--) {
                BSP_LED_On(i);
                OSTimeDlyHMSM(0, 0, 0, 50);
                BSP_LED_Off(i);
                OSTimeDlyHMSM(0, 0, 0, 100);
            }
        }
        for (i = 0; i < 4; i++) {
            BSP_LED_On(0);
            OSTimeDlyHMSM(0, 0, 0, 200);
            BSP_LED_Off(0);
            OSTimeDlyHMSM(0, 0, 0, 200);
        }
    }
}
```

實驗步驟

- 6.在App_TaskCreate()中執行兩項工作

```
static  void  App_TaskCreate (void)
{
    CPU_INT08U  os_err;

    os_err = OSTaskCreateExt(App_NewTask,
                             0,
                             &App_NewTaskStk[APP_New_Task_STK_SIZE - 1],
                             (INT8U           ) APP_New_Task_PRIO,
                             (INT16U          ) APP_New_Task_PRIO,
                             &App_NewTaskStk[0],
                             (INT32U          ) APP_New_Task_STK_SIZE,
                             0,
                             (INT16U          )(OS_TASK_OPT_STK_CLR | OS_TASK_OPT_STK_CHK));

    os_err = OSTaskCreateExt( App_TaskTxtLcd,
                             0,
                             &App_TaskTxtLcdStk[APP_Task_TxtLcd_STK_SIZE - 1],
                             (INT8U           ) APP_Task_TxtLcd_PRIO,
                             (INT16U          ) APP_Task_TxtLcd_PRIO,
                             &App_TaskTxtLcdStk[0],
                             (INT32U          ) APP_Task_TxtLcd_STK_SIZE,
                             (void          * ) 0,
                             (INT16U          )(OS_TASK_OPT_STK_CLR | OS_TASK_OPT_STK_CHK));

}
```

實驗步驟

- 7. dfu燒錄
  - 7.1 必須在main()中加入 NVIC_SetVectorTable(0x08003000,0x0);
  - 7.2 在STM32_Flash.scat裡將

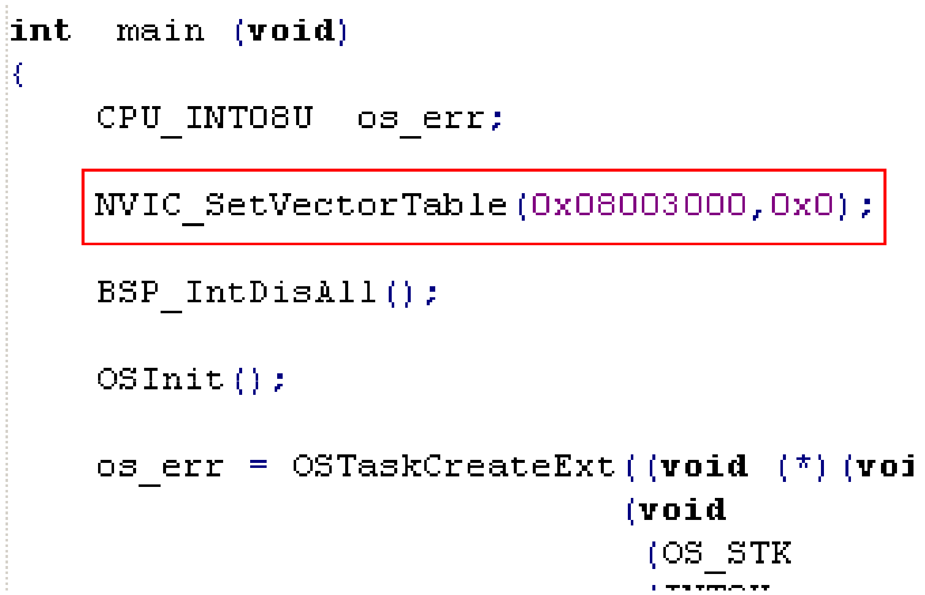
    LR_IROM1 0x08000000 0x00020000 與

    ER_IROM1 0x08000000 0x08020000 改為

    LR_IROM1 0x08003000 0x00020000 與

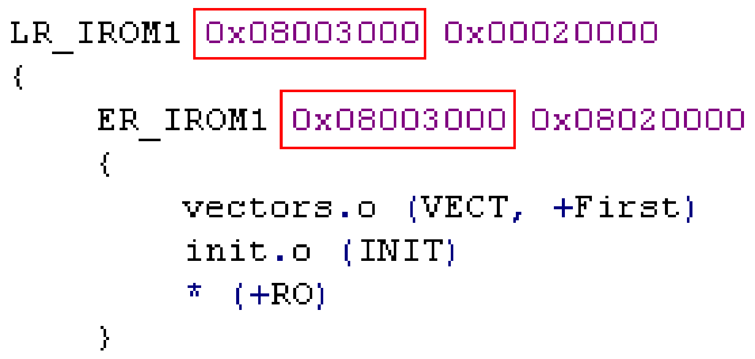
    ER_IROM1 0x08003000 0x08020000

7.1



```
int  main (void)
{
    CPU_INT08U  os_err;

    NVIC_SetVectorTable(0x08003000,0x0);

    BSP_IntDisAll();

    OSInit();

    os_err = OSTaskCreateExt((void (*)(voi
                             (void
                              (OS_STK
```

7.2



```
LR_IROM1 0x08003000 0x00020000
{
    ER_IROM1 0x08003000 0x08020000
    {
        vectors.o (VECT, +First)
        init.o (INIT)
        * (+RO)
    }
```