UC/OSii 在 M251上的移植

NuMicro® 32位系列微控制器範例代碼介紹

文件資訊

應用簡述	在 M251 系列單片機上移植 UC/OSii_V2.93 的示例代碼
BSP 版本	M251_M252_M254_M256_M258_Series_BSP_CMSIS_V3.02.003
開發平台	NuMaker-M258KE Ver1.1

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller and microprocessor based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1. 概述

本代碼介紹在 M251 系列單片機上移植 UC/OSii 需要添加哪些文件。

代碼中的工程建立了相互通信的兩個任務。

1.1 原理

爲何要用 UC/OS? 因爲用了 UC/OS,每個任務都像是獨占了 CPU，因此，一個應用可以分成幾個任務，而這些任務，可以由不同的人編寫。

當一個任務被另一任務抢先時，就像是 CPU 去執行中斷代碼了，過一會兒 CPU 就會回來。每一個任務都有獨立的堆叠，在任務開始調度前，每個堆叠被初始化成像是剛剛發生中斷一樣，CPU 寄存器被壓入了堆叠。

```
S_STK *OSTaskStkInit(void (*task)(void *p_arg), void *p_arg, OS_STK *ptos, INT16U opt)
{
    OS_STK *stk;

    (void)opt;                                          /* 'opt' is not used, prevent warning */
    stk       = ptos;                                   /* Load stack pointer  */

                     /* Registers stacked as if auto-saved on exception */
    *(stk)    = (INT32U)0x01000000uL;                   /* xPSR         */
    *(--stk)  = (INT32U)task;                           /* Entry Point */
    *(--stk)  = (INT32U)OS_TaskReturn;                  /* R14 (LR)   */
    *(--stk)  = (INT32U)0x12121212uL;                   /* R12       */
    *(--stk)  = (INT32U)0x03030303uL;                   /* R3       */
    *(--stk)  = (INT32U)0x02020202uL;                   /* R2      */
    *(--stk)  = (INT32U)0x01010101uL;                   /* R1     */
    *(--stk)  = (INT32U)p_arg;                          /* R0 : argument  */

                   /* Remaining registers saved on process stack */
    *(--stk)  = (INT32U)0x07070707uL;                   /* R7          */
    *(--stk)  = (INT32U)0x06060606uL;                   /* R6         */
    *(--stk)  = (INT32U)0x05050505uL;                   /* R5        */
    *(--stk)  = (INT32U)0x04040404uL;                   /* R4       */

    return (stk);
}
```

建好所有的任務並且堆叠初始化好以後，CPU 開始執行最高優先級任務代碼，就像是剛從中斷返回來一樣，這個工作是在函數 OSStart(), OSStartHighRdy() 和 PendSV_Handler 中做的。

任務的切換在函數 PendSV_Handler()中。最高優先級任務堆疊的首地址被讀出來寫入棧指針 PSP，然後執行中斷返回，CPU 就返回到最高優先級任務去執行代碼了。

```
PendSV_Handler
    CPSID   I                           ; Prevent interruption during context switch
    MRS   R0, PSP                       ; PSP is process stack pointer

    CMP   R0, #0
    BEQ   OS_CPU_PendSVHandler_nosave    ; equivalent code to CBZ from M3 arch to M0 arch
                                ; Except that it does not change the condition code flags

    SUBS  R0, R0, #0x10     ; Adjust stack pointer to where memory needs to be stored to
                            ; avoid overwriting
    STM   R0!, {R4-R7}      ; Stores 4 4byte registers, default increments SP after each
                            ; storing
    SUBS    R0, R0, #0x10   ; STM does not automatically call back the SP to initial
                            ; location so we must do this manually
    LDR   R1, =OSTCBCur       ; OSTCBCur->OSTCBStkPtr = SP;
    LDR   R1, [R1]
    STR   R0, [R1]            ; R0 is SP of process being switched out

                              ; At this point, entire context of process has been saved
OS_CPU_PendSVHandler_nosave
    PUSH   {R14}              ; Save LR exc_return value
    LDR   R0, =OSTaskSwHook   ; OSTaskSwHook();
    BLX   R0
    POP    {R0}
    MOV   R14, R0

    LDR   R0, =OSPrioCur      ; OSPrioCur = OSPrioHighRdy;
    LDR   R1, =OSPrioHighRdy
    LDRB  R2, [R1]
    STRB  R2, [R0]

    LDR   R0, =OSTCBCur       ; OSTCBCur  = OSTCBHighRdy;
    LDR   R1, =OSTCBHighRdy
    LDR   R2, [R1]
    STR   R2, [R0]
    LDR   R0, [R2]          ; R0 is new process SP; SP = OSTCBHighRdy->OSTCBStkPtr;
    LDM   R0!, {R4-R7}      ; Restore R4-R7 from new process stack
    MSR   PSP, R0           ; Load PSP with new process SP
    MOV   R0,  R14
    MOVS  R1,  #0x04        ; Immediate move to register
    ORRS  R0,  R1           ; Ensure exception return uses process stack
    MOV   R14, R0
    CPSIE I
    BX    LR                ; Exception return will restore remaining context

    ALIGN                   ; Ensures that ARM instructions start on four-byte boundary
    END
```

PendSV 中斷代碼的流程圖見 圖 1-1. 進入 PendSV 中斷後, 如果 PSP 不等 0，表明 CPU 剛從一個任務跳轉過來，要先保存當前任務的上下文，包括棧指針 SP，然後再獲取新任務的上下文。

PSP=0 表明剛從 OSStart()開始調度任務，不需要保存當前任務的上下文。讀出當前最高優先級任務的上下文，棧指針 PSP 指向最高優先任務的堆疊，然後執行中斷返回，CPU 就去執行最高優先級任務去了。

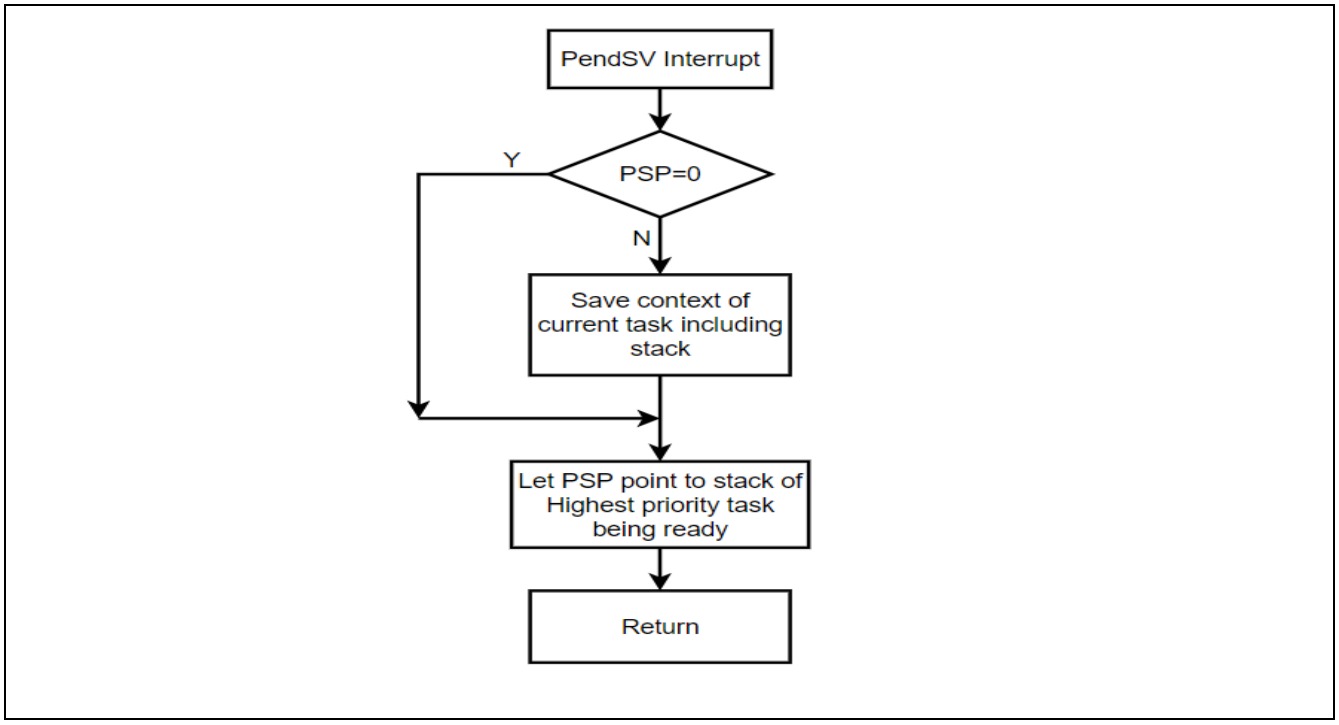

圖 1-1 PendSV 中斷代碼流程圖

1.2 執行結果

任務 2 讓一個 LED 閃爍，閃爍間隔或周期由任務 1 發過來。

2. 代碼介紹

工程結構見圖 2-1, 工程中只包含必要文件。
由于版權問題，軟件包中不含UC/OS內核代碼文件，請到相關網站下載US/OS內核代碼，按如下目錄添加到工程中即可。

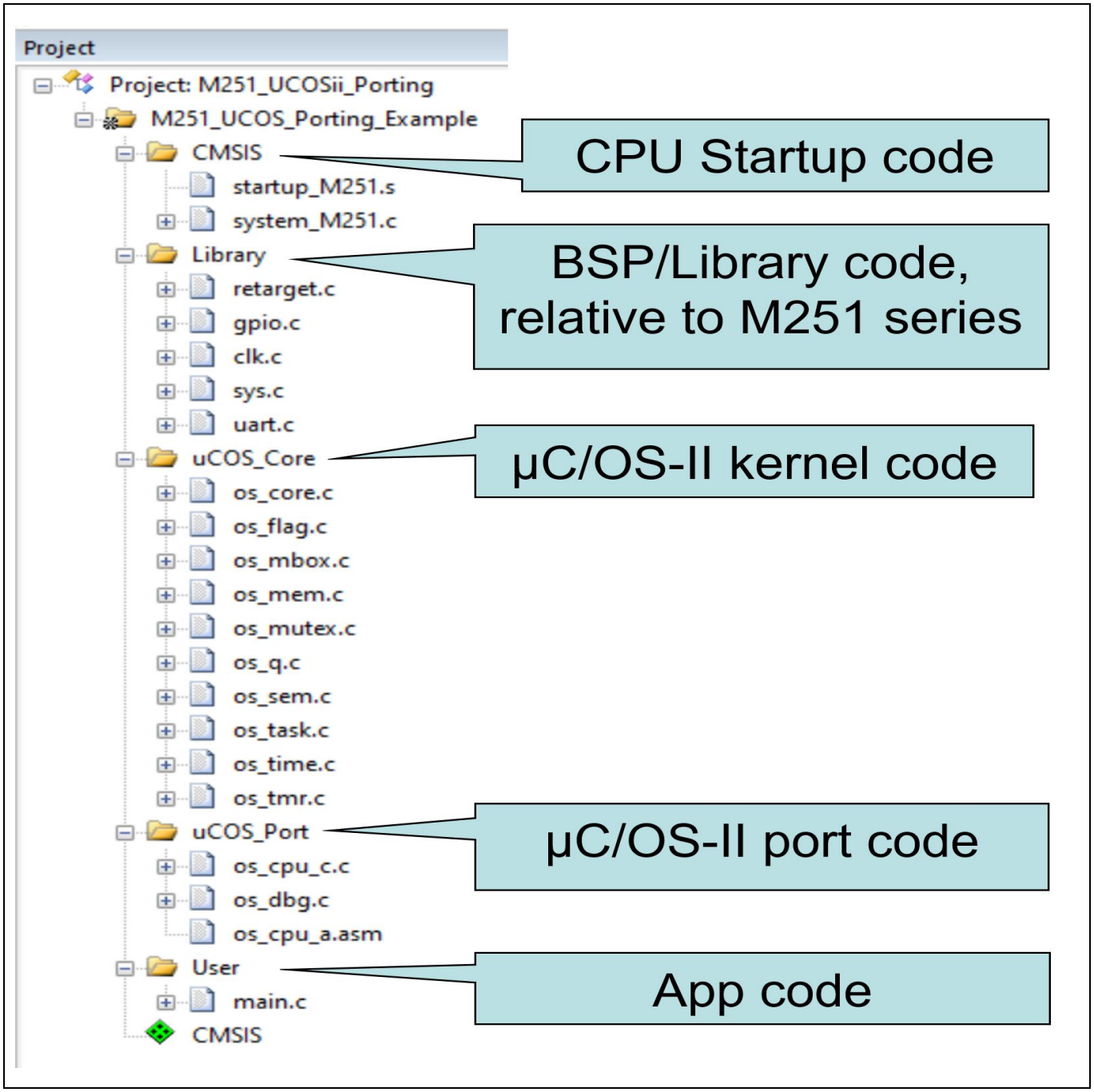

圖 2-1 工程結構

M251 的各周边初始化後，先建立信號量、郵箱和隊列，然後建立 任務。執行 OSStart()之後，代碼開始任務調度。

```
int main(void)
{
    int32_t Cnt32;

    SYS_UnlockReg();        /* Unlock protected registers */
    SYS_Init();             /* Init System, peripheral clock and multi-function I/O */
    SYS_LockReg();          /* Lock protected registers */

    UART0_Init();           /* Init UART0 for printf */
    printf("\n\nCPU @ %u Hz\n", SystemCoreClock);

    OSInit();                                                   // uCOS initialization.
    p_MailBox_1 = OSMboxCreate((void *)0);                      // create a mail box

    OSTaskCreateExt((void(*)(void *))task1_Highest,             // create a task
                    (void *)0,
                    (OS_STK *)&task1_stk_H[LED_TASK_STK_SIZE - 1],
                    (INT8U)TASK1_PRIO_Highest,
                    (INT16U)TASK1_PRIO_Highest,
                    (OS_STK *)&task1_stk_H[0],
                    (INT32U)LED_TASK_STK_SIZE,
                    (void *)0,
                (INT16U)OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR | OS_TASK_OPT_SAVE_FP);

    OSTaskCreate((void(*)(void *))task2_LED,                            // create another task
                 (void *)0,
                 (OS_STK *)&task2_stk_led[LED_TASK_STK_SIZE - 1],
                 (INT8U)TASK2_PRIO_LED);

    OSStart();                                                          // Start schedule

    while (1) {} ;                              // This sentence would be never executed.
}
```

每個任務都是一個 while(1){ } 循環。在循環中必須調用系統函數，如 OSTimeDaley(), OSMboxPost()等，如此，其它任務才有時間執行。

SysTick 定時器必須在最高優先級的任務裏啓動，否則，如果在任務調度之前發生了 SysTick 定時器中斷，系統可能崩潰。

任務 1 優先級最高，開頭啓動 SysTick 定時器。while(1)循環裏，每 3 秒鐘讓給任務 2 發一次數據。

```
void task1_Highest(void *pvParameters)
{                                                   // MUST active tick-timer in highest task
    SysTick_Config(SystemCoreClock / OS_TICKS_PER_SEC);    // active tick-timer

    while (1)
    {
        OSTimeDlyHMSM(0, 0, 3, 0);                          // Delay 3 seconds

        if (Mail_Message[0] != 100) Mail_Message[0] = 100;
        else Mail_Message[0] = 500 ;                    // Configure message in mail box

        OSMboxPost(p_MailBox_1, Mail_Message);          // Send messgae via mail box
    }
}
```

任務 2 是讓 LED 閃爍，閃爍周期，按任務 1 發送過來的時間

```
void task2_LED(void *pvParameters)
{                                                   // MUST active tick-timer in highest task
    int32_t *ptr ;
    int32_t  static Time = 300 ;

    while (1)
    {
       ptr =OSMboxAccept(p_MailBox_1);  //Check whether there are messages in the mailbox

        if (ptr !=(void*)0) Time = *ptr; //Read out if there is any message in the mailbox

        OSTimeDlyHMSM(0, 0, 0, Time);             // Delay time according to message.
        PB14 ^= 1 ;                               // LED flash
    }}
```

3. 軟體與硬體需求

3.1 軟體需求

- BSP 版本
  - M251_M252_M254_M256_M258_Series_BSP_CMSIS_V3.02.003
- IDE 版本
  - Keil uVersion 5.26

3.2 硬體需求

- 電路元件
  - NuMaker- M258KE Ver1.1

4. 目錄資訊

EC_ M251_UCOSii_Porting_V1.00	
Library	Sample code header and source files
CMSIS	Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp.
Device	CMSIS compliant device header file
StdDriver	All peripheral driver header and source files
SampleCode	
ExampleCode	Source file of example code
uCOS_II	Files relative to uCOS

圖 4-1 目錄資訊

5. 範例程式執行

1. 根據目錄資訊章節進入 ExampleCode 路徑中的 KEIL 資料夾，雙擊 *M251_UCOSii_Porting.uvprojx*。
2. 進入編譯模式介面
   - 編譯
   - 下載代碼至記憶體
   - 進入 / 離開除錯模式
3. 進入除錯模式介面
   - 執行代碼

6. 修訂紀錄

Date	Revision	Description
2022.07.27	1.00	初始發布。

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.