UC/OSii 在 M251上的移植

NuMicro® 32位系列微控制器范例代码介绍

文件信息

应用简述	在 M251 系列单片机上移植 UC/OSii_V2.93 的示例代码
BSP 版本	M251_M252_M254_M256_M258_Series_BSP_CMSIS_V3.02.003
开发平台	NuMaker-M258KE Ver1.1

1. 概述

本代码介绍在 M251 系列单片机上移植 UC/OSii 需要添加哪些文件。

代码中的工程建立了相互通信的两个任务。

1.1 原理

为何要用 UC/OS? 因为用了 UC/OS,每个任务都像是独占了 CPU，因此，一个应用可以分成几个任务，而这些任务，可以由不同的人编写。

当一个任务被另一任务打断时，就像是 CPU 去执行中断代码了，过一会儿 CPU 就会回来。每一个任务都有独立的栈区，在任务开始调度前，每个栈区被初始化成像是刚刚发生中断一样，CPU 寄存器被压入了栈区。

```
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);
}
```

建好所有的任务并且栈区初始化好以后，CPU 开始执行最高优先级任务代码，就像是刚从中断返回来一样，这个工作是在函数 OSStart(), OSStartHighRdy() 和 PendSV_Handler 中做的。

任务的切换在函数 PendSV_Handler()中。最高优先级任务栈区的首地址被读出来写入栈指针 PSP，然后执行中断返回，CPU 就返回到最高优先级任务去执行代码了。

```
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 中断代码的流程图见 图 1-1. 进入 PendSV 中断后, 如果 PSP 不等 0，表明 CPU 刚从一个任务跳转过来，要先保存当前任务的上下文，包括栈指针 SP，然后再获取新任务的上下文。

PSP=0 表明刚从 OSStart()开始调度任务，不需要保存当前任务的上下文。读出当前最高优先级任务的上下文，栈指针 PSP 指向最高优先任务的栈区，然后执行中断返回，CPU 就去执行最高优先级任务去了。

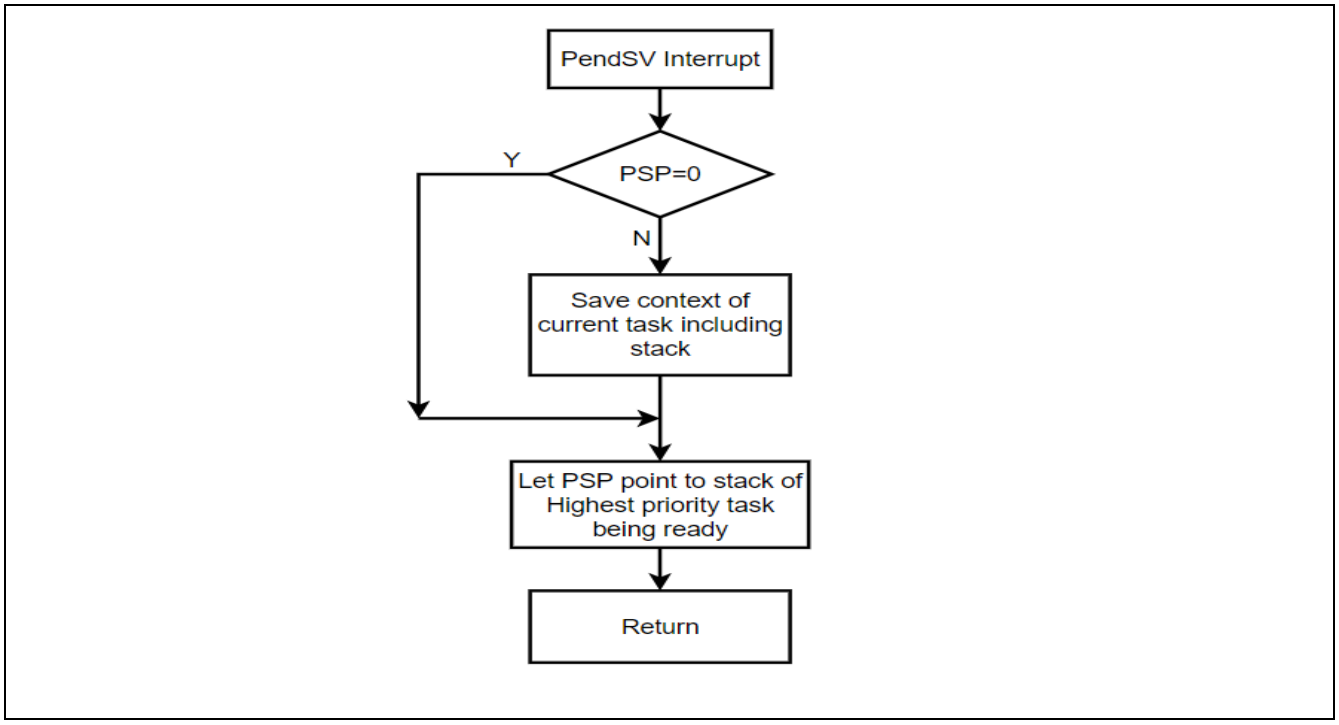

图 1-1 PendSV 中断代码流程图

1.2 执行结果

任务 2 让一个 LED 闪烁，闪烁间隔或周期由任务 1 发过来。

2. 代码介绍

工程结构见图 2-1，工程中只包含必要文件。
由于版权问题，软件包中不含UC/OS内核代码文件，请到相关网站下载US/OS内核代码，按如下目录添加到工程中即可。

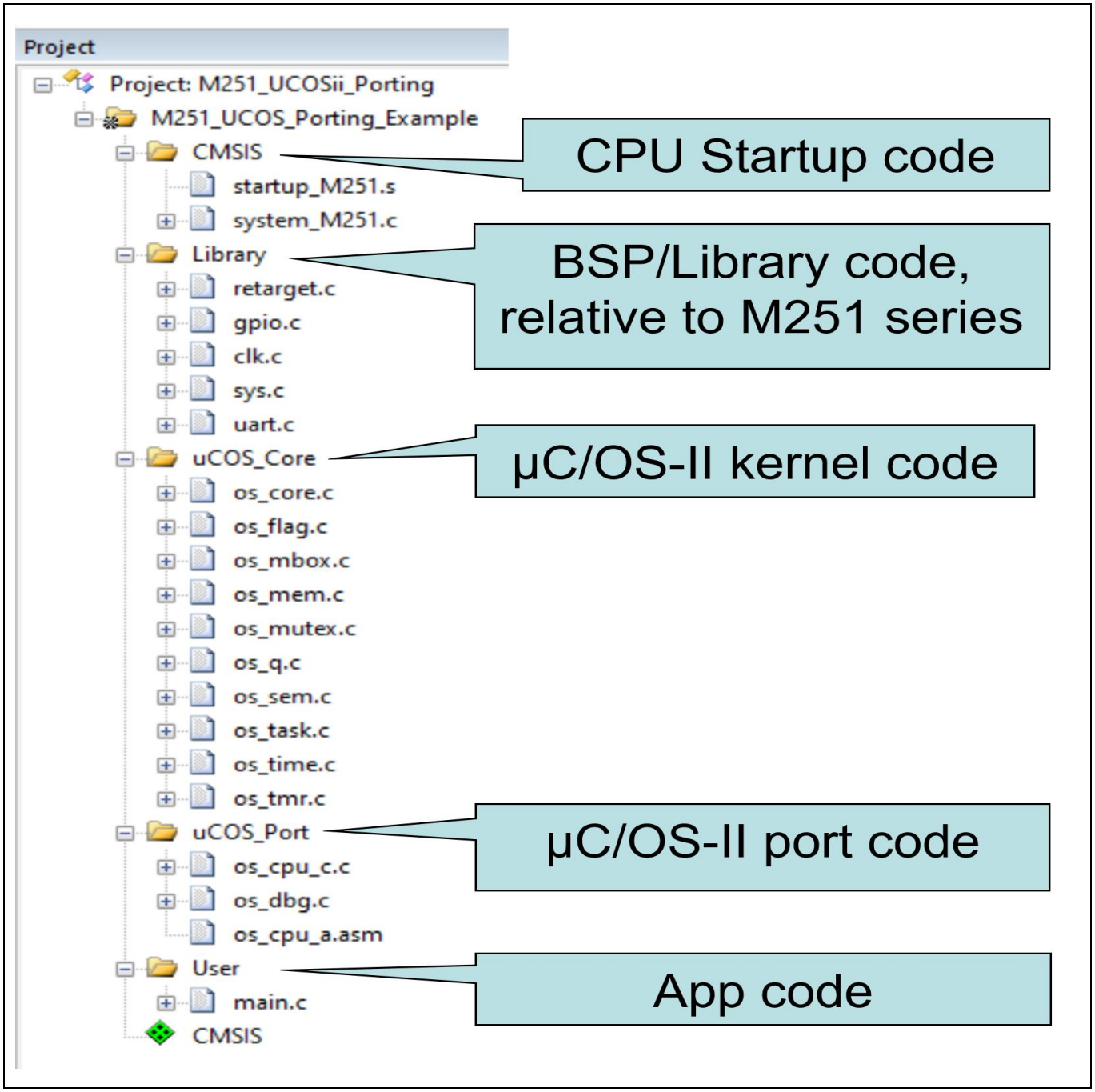

图 2-1 工程结构

M251 的各外设初始化后，先建立信号量、邮箱和队列，然后建立 任务。执行 OSStart()之后，代码开始任务调度。

```
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.
}
```

每个任务都是一个 while(1){ } 循环。在循环中必须调用系统函数，如 OSTimeDaley(), OSMboxPost()等，如此，其它任务才有时间执行。

系统节拍定时器必须在最高优先级的任务里启动，否则，如果在任务调度之前发生了节拍定时器中断，系统可能崩溃。

任务 1 优先级最高，开头启动 SysTick 定时器。while(1)循环里，每 3 秒钟给任务 2 发一次数据。

```
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
    }
}
```

任务 2 是让 LED 闪烁，闪烁周期，按任务 1 发送过来的时间。

```
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. 软件与硬件需求

3.1 软件需求

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

3.2 硬件需求

- 电路组件
  - NuMaker- M258KE Ver1.1

4. 目录信息

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

图 4-1 目录信息

5. 范例程序执行

1. 根据目录信息章节进入 ExampleCode 路径中的 KEIL 文件夹，双击 *M251_UCOSii_Porting.uvprojx*。
2. 进入编译模式界面
   - 编译
   - 下载代码至内存
   - 进入 / 离开仿真模式
3. 进入仿真模式界面
   - 执行代码

6. 修订纪录

Date	Revision	Description
2022.07.27	1.00	初始发布。

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.