

**87家公司发出退市警告，你买的股票也在其中吗？**已有**98746**人通过检查规避了风险

投资有风险 入市需谨慎

| 马上输入股票代码

立即诊断

LookAt_theStar的博客

PendSV与堆栈操作

2015-04-24 21:23:33

转载

摘自：http://www.cnblogs.com/sky1991/p/stepbystep_stm32_os_3.html

一、什么是PendSV

PendSV是可悬起异常，如果我们把它配置最低优先级，那么如果同时有多个异常被触发，它会在其他异常执行完毕后再执行，而且任何异常都可以中断它。更详细的内容在《Cortex-M3 权威指南》里有介绍，下面我摘抄了一段。

OS 可以利用它“缓期执行”一个异常——直到其它重要的任务完成后才执行动作。悬起PendSV的方法是：手工往NVIC的PendSV悬起寄存器中写1。悬起后，如果优先级不够高，则将缓期待待执行。

PendSV的典型使用场合是在上下文切换时（在不同任务之间切换）。例如，一个系统中有两个就绪的任务，上下文切换被触发的场合可以是：1、执行一个系统调用2、系统滴答定时器（SYSTICK）中断，（轮转调度中需要）

让我们举个简单的例子来辅助理解。假设有这么一个系统，里面有两个就绪的任务，并且通过SysTick异常启动上下文切换。但若在产生SysTick异常时正在响应一个中断，则SysTick异常会抢占其ISR。在这种情况下，OS是不能执行上下文切换的，否则将使中断请求被延迟，而且在真实系统中延迟时间还往往不可预知——任何有一丁点实时要求的系统都决不能容忍这种事。因此，在CM3中也是严禁没商量——如果OS在某中断活跃时尝试切入线程模式，将触犯用法fault异常。

为解决此问题，早期的OS大多会检测当前是否有中断在活跃中，只有在无任何中断需要响应时，才执行上下文切换（切换期间无法响应中断）。然而，这种方法的弊端在于，它可以把任务切换动作拖延很久（因为如果抢占了IRQ，则本次SysTick在执行后不得作上下文切换，只能等待下一次SysTick异常），尤其是当某中断源的频率和SysTick异常的频率比较接近时，会发生“共振”，使上下文切换迟迟不能进行。现在好了，PendSV来完美解决这个问题了。PendSV异常会自动延迟上下文切换的请求，直到其它的ISR都完成了处理后才放行。为实现这个机制，需要把PendSV编程为最低优先级的异常。如果OS检测到某IRQ正在活动并且被SysTick抢占，它将悬起一个PendSV异常，以便缓期执行上下文切换。

使用PendSV控制上下文切换中事件的流水账记录如下：

1. 任务A呼叫SVC来请求任务切换（例如，等待某些工作完成）

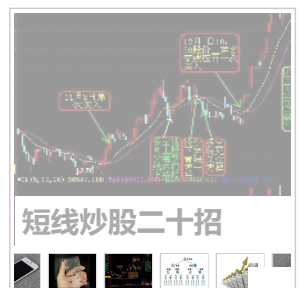


LookAt_theStar

加关注

博客等级：**12**博客积分：**689**博客访问：**20,384**关注人气：**4**

荣誉徽章：



相关博文

背带裤女孩俏皮写真

摄影师leale老秘

前卫潮女穿衣打扮亮瞎眼

曹作兰

街拍：虽然立秋已半月，美女不舍

曹作兰艺术行走

街拍：仪态万方、风姿绰约的老外

2. OS接收到请求，做好上下文切换的准备，并且悬起一个 PendSV异常。
3. 当 CPU退出 SVC后，它立即进入 PendSV，从而执行上下文切换。
4. 当 PendSV执行完毕后，将返回到任务 B，同时进入线程模式。
5. 发生了一个中断，并且中断服务程序开始执行
6. 在 ISR执行过程中，发生 SysTick异常，并且抢占了该 ISR。
7. OS执行必要的操作，然后悬起 PendSV异常以作好上下文切换的准备。
8. 当 SysTick退出后，回到先前被抢占的 ISR中，ISR继续执行
9. ISR执行完毕并退出后，PendSV服务例程开始执行，并且在里面执行上下文切换
10. 当 PendSV执行完毕后，回到任务 A，同时系统再次进入线程模式。

我们在uCOS的PendSV的处理代码中可以看到：

```
OS_CPU_PendSVHandler
{
    CPSID I ; 关中断
    ;保存上文
    ;.....
    ;切换下文
    CPSIE I ;开中断
    BX LR ;异常返回
}
```

它在异常一开始就关闭了中端，结束时开启中断，中间的代码为临界区代码，即不可被中断的操作。PendSV异常是任务切换的堆栈部分的核心，由他来完成上下文切换。PendSV的操作也很简单，主要有设置优先级和触发异常两部分：

```
NVIC_INT_CTRL EQU 0xE000ED04 ; 中断控制寄存器NVIC_SYSPRI14 EQU 0xE000ED22 ; 系统优先级寄存器
LDR R1, =NVIC_PENDSV_PRI
STRB R1, [R0] ; 触发PendSV异常LDR R0, =NVIC_INT_CTRL
LDR R1, =NVIC_PENDSVSET
STR R1, [R0]
```

二、堆栈操作

Cortex M4有两个堆栈寄存器，主堆栈指针（MSP）与进程堆栈指针（PSP），而且任一时刻只能使用其中的一个。MSP为复位后缺省使用的堆栈指针，异常永远使用MSP，如果手动开启PSP，那么线程使用PSP，否则也使用MSP。怎么开启PSP？

```
MSR      PSP, R0                ; Load PSP with new
                                ; value
ORR      LR, LR, #0x04          ; Ensure exception
很容易就看出来了，置LR的位2为1，那么异常返回后，线程使用PSP。
```

写OS首先要将内存分配搞明白，单片机内存本来就很小，所以我们当然要斤斤计较一下。在OS运行之前，我们首先要初始化MSP和PSP，OS_CPU_ExceptStkBase是外部变量，假如我们给主堆栈分配1KB（256*4）的内存即OS_CPU_ExceptStk[256],则OS_CPU_ExceptStkBase=&OS_CPU_ExceptStk[256-1]。

曹作兰艺术行走

陈益峰：风水理论中的龙边与虎边
北京风水师北京风水大师

思婷美眉的近照
绵竹2008

段子来了大（五）大（dai）行（xi）
李秀亭de专栏

古人做事：始于立心，得于人和，止于
青岛城阳律师网

屌丝死于霍欣欣，中产死于郎咸平，穷人
用户330152631

大盘回补缺口通信
巨丰投顾

【早盘策略】市场普跌等待周五
史月波

下周继续挤压式调整
聊四方

查看更多

推荐博文

《朗读者》《见字如面2》《见字如面》
新能源汽车真的能替代游
电影天堂的好莱坞是如何设计出 “
不只是朝核危机！当世界开始疯狂
“211” “985”：打破 “世
“海归县长” 落马暴露干部日常监
很多中国人的坏毛病，一条没有可
一颗被冷冻了5年的头颅，被它一
杭州纵火案保姆被检方以放火罪和
互联网电视能否杀出重围？



标新立异的前
卫潮女

壮观！800盏
龙母水灯燃放



柳岩麻花辫俏
皮大片

飞扬吧，你的
小青春



这里有太多人
的回忆

2017华夏幸福
北京马拉松

查看更多

```

EXTERN  OS_CPU_ExceptStkBase    ;PSP清零，作为首次上下文切换的标志
        MOVS    R0, #0
        MSR     PSP, R0    ;将MSP设为我们为其分配的内存地址    LDR     R0, =OS_CPU_Exc
        LDR     R1, [R0]
        MSR     MSP, R1

```

然后就是PendSV上下文切换中的堆栈操作了，如果不使用FPU，则进入异常自动压栈
xPSR，PC，LR，R12，R0-R3，我们还要把R4-R11入栈。如果开启了FPU，自动压栈的
寄存器还有S0-S15，还需吧S16-S31压栈。

```

MRS     R0, PSP
        SUBS    R0, R0, #0x20    ;压入R4-R11    STM     R0, {R4-R11}

        LDR     R1, =Cur_TCB_Point    ;当前任务的指针    LDR     R1, [R1]    STR
出栈类似，但要注意顺序

```

```

LDR     R1, =TCB_Point    ;要切换的任务指针    LDR     R2, [R1]
        LDR     R0, [R2]    ; R0为要切换的任务堆栈地址
        LDM     R0, {R4-R11}    ; 弹出R4-R11    ADDS    R0, R0, #0x20

        MSR     PSP, R0    ;更新PSP

```

三、OS实战

新建os_port.asm文件，内容如下：

```

NVIC_INT_CTRL    EQU    0xE000ED04    ; Interrupt con
RSEG CODE:CODE:NOROOT(2)
THUMB

EXTERN  g_OS_CPU_ExceptStkBase

EXTERN  g_OS_Tcb_CurP
EXTERN  g_OS_Tcb_HighRdyP

PUBLIC OSStart_Asm
PUBLIC PendSV_Handler
PUBLIC OSCtxSw

OSCtxSw
        LDR     R0, =NVIC_INT_CTRL
        LDR     R1, =NVIC_PENDSVSET    STR     R1, [R0]
        BX     LR    ; Enable interrup
        LDR     R0, =NVIC_SYSPRI14    ; Set the PendS
        STRB    R1, [R0]

        MOVS    R0, #0    ; Set the PSP t

        LDR     R0, =g_OS_CPU_ExceptStkBase    ; Initialize
        MSR     MSP, R1

        LDR     R0, =NVIC_INT_CTRL    ; Trigger the P

```

```

CPSIE    I                                ; Enable interr
B        OSStartHang                      ; Should never
PendSV_Handler
CPSID    I                                ; Prevent inter
MRS      R0, PSP                          ; PSP is proces
CBZ      R0, OS_CPU_PendSVHandler_nosave ; Skip register
SUBS     R0, R0, #0x20                     ; Save remainin

LDR      R1, =g_OS_Tcb_CurP               ; OSTCBCur

; At this point

LDR      R0, =g_OS_Tcb_CurP               ; OSTCBCur
LDR      R2, [R1]    STR      R2, [R0]

```

```

R0, [R2]                                ; R0 is new process
R0, {R4-R11}                            ; Restore r4-11

PSP, R0                                ; Load PSP with
LR, LR, #0x04                          ; Ensure except
I
LR                                        ; Exception ret

```

视频直播交



下：

```

#include <stdio.h>#define OS_EXCEPT_STK_SIZE 1024#define TASK_1_STK_SIZE 1024#d
typedef void (*OS_TASK)(void);

```

```

typedef struct OS_TCB
{
    OS_STK *StkAddr;
}OS_TCB,*OS_TCBP;

```

```

OS_TCBP g_OS_Tcb_CurP;
OS_TCBP g_OS_Tcb_HighRdyP;static OS_STK OS_CPU_ExceptStk[OS_EXCEPT_STK_SIZE];
OS_STK *g_OS_CPU_ExceptStkBase;static OS_TCB TCB_1;static OS_TCB TCB_2;static 0
{ if(g_OS_Tcb_CurP == &TCB_1)
    g_OS_Tcb_HighRdyP=&TCB_2; else
    g_OS_Tcb_HighRdyP=&TCB_1;

    OSCtxSw();
}void task_1()
{
    printf("Task 1 Running!!!\n");
    Task_Switch();
    printf("Task 1 Running!!!\n");
    Task_Switch();
}void task_2()
{

    printf("Task 2 Running!!!\n");
    Task_Switch();
}

```

```

    printf("Task 2 Running!!!\n");
    Task_Switch();
}void Task_End(void)
{
    printf("Task End\n"); while(1)
    {}
}void Task_Create(OS_TCB *tcb, OS_TASK task, OS_STK *stk)
{
    OS_STK *p_stk;
    p_stk = stk;
    p_stk = (OS_STK *) ((OS_STK) (p_stk) & 0xFFFFFFFF8u);
    *--p_stk = (OS_STK) 0x01000000uL; //xPSR
    *--p_stk = (OS_STK) task; // Entry Point
    *--p_stk = (OS_STK) Task_End; // R14 (
    *--p_stk = (OS_STK) 0x12121212uL; // R12
    *--p_stk = (OS_STK) 0x03030303uL; // R3
    *--p_stk = (OS_STK) 0x02020202uL; // R2
    *--p_stk = (OS_STK) 0x01010101uL; // R1
    *--p_stk = (OS_STK) 0x00000000u; // R0

    *--p_stk = (OS_STK) 0x11111111uL; // R11
    *--p_stk = (OS_STK) 0x10101010uL; // R10
    *--p_stk = (OS_STK) 0x09090909uL; // R9
    *--p_stk = (OS_STK) 0x08080808uL; // R8
    *--p_stk = (OS_STK) 0x07070707uL; // R7
    *--p_stk = (OS_STK) 0x06060606uL; // R6
    *--p_stk = (OS_STK) 0x05050505uL; // R5
    *--p_stk = (OS_STK) 0x04040404uL; // R4
    tcb->StkAddr=p_stk;
}int main()
{

    g_OS_CPU_ExceptStkBase = OS_CPU_ExceptStk + OS_EXCEPT_STK_SIZE - 1;

    Task_Create(&TCB_1, task_1, &TASK_1_STK[TASK_1_STK_SIZE-1]);
    Task_Create(&TCB_2, task_2, &TASK_2_STK[TASK_1_STK_SIZE-1]);

    g_OS_Tcb_HighRdyP=&TCB_1;

    OSStart_Asm();
    return 0;
}

```

编译下载并调试：

在此处设置断点

此时寄存器的值，可以看到R4-R11正是我们给的值，单步运行几次，可以看到进入了我们的任务task_1或task_2，任务里打印信息，然后调用Task_Switch进行切换，OSCtxSw触发PendSV异常。

IO输出如下：

至此我们成功实现了使用PenSV进行两个任务的互相切换。之后，我们使用使用SysTick实现比较完整的多任务切换。

更多博文

- ADO.NET与数据库操作
- [转载]（转）塞林格：破碎故事之心
- C#中调用百度地图API应用（.net + JavaScrip...
- 线程同步技术剖析：临界区、时间、信号量、...
- Labview 同步——信号量

0

喜欢

阅读(368) 评论(0) 收藏(0) 转载(0) 举报

分享

前一篇: ADO.NET与数据库操作

后一篇:uCOSII汇编文件os_cpu_a.asm

评论

重要提示：警惕虚假中奖信息

0 条评论 展开

发评论

相关阅读




曹作兰艺术... 街拍：虽然立秋已半月，美女不舍露脐装

虽然入秋已经半个月，但是北京的秋老虎依然厉害。应对居高不下的气温，美女依然选择露脐装。露脐装，应该是最清凉的方式。当上面露肩、下面露腿，中间再露出腰身和肚皮，...[详细]



2017-08-29 08:24:33



曹作兰艺术... 街拍：仪态万方、风姿绰约的老外美女