第3章 任务的定义与任务切换的实现

主讲内容

3.1 本章目标

3.2 什么是任务

3.3 创建任务

3.4 OS系统初始化

3.5 启动系统

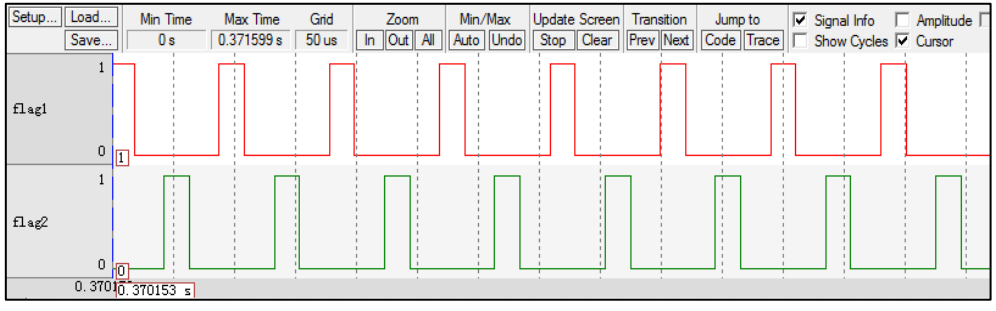
3.6 任务切换

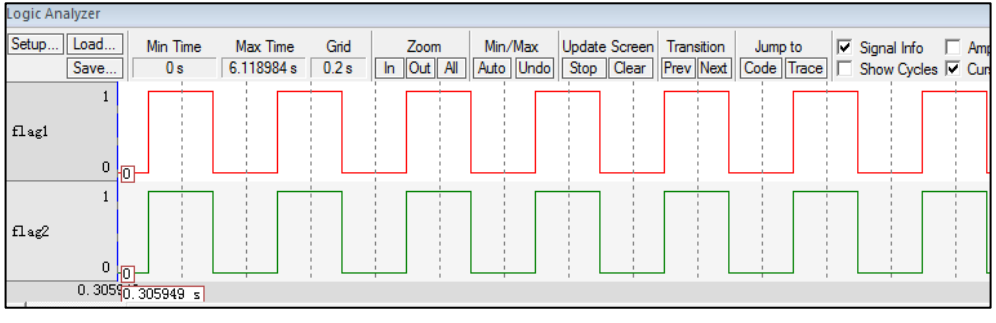
3.7 实验

参考资料:《μCOS-III内核实现与应用开发实战指南》

目标

必须要学会创建任务，并重点掌握任务是如何切换的。





什么是任务

在裸机系统中，系统的主体就是main()函数里面顺序执行的无限循环，这个无限循环里面CPU按照顺序完成各种事情。在多任务系统中，我们根据功能的不同，把整个系统分割成一个个独立的且无法返回的函数，这个函数我们称为任务，也有人称之为线程。

定义任务栈

如果有全局变量，有子函数调用，有中断发生。那么系统在运行的时候，全局变量放在哪里，子函数调用时，局部变量放在哪里，中断发生时，函数返回地址发哪里。

在裸机系统中，他们统统放在一个叫栈的地方，栈是单片机RAM里面一段连续的内存空间，栈的大小由启动文件里面的代码配置

在多任务系统中，每个任务都是独立的，互不干扰的，所以要为每个任务都分配独立的栈空间，这个栈空间通常是一个预先定义好的全局数组。这些一个个的任务栈也是存在于RAM中。


```
static   CPU_STK   Task1Stk[TASK1_STK_SIZE];
static   CPU_STK   Task2Stk[TASK2_STK_SIZE];
```

凡是与CPU类型相关的数据类型则统一在cpu.h中定义，与OS相关的数据类型则在os_type.h定义。

定义任务函数

任务是一个独立的函数，函数主体无限循环且不能返回。

定义任务控制块TCB

任务控制块就相当于任务的身份证，里面存有任务的所有信息，比如任务的栈，任务名称，任务的形参等。

TCB是一个新的数据类型，在os.h定义

实现任务创建函数

任务的栈，任务的函数实体，任务的TCB最终需要联系起来才能由系统进行统一调度。那么这个联系的工作就由任务创建函数OSTaskCreate来实现

该函数在os_task.c中实现

OSTaskStkInit()是任务栈初始化函数。当任务第一次运行的时候，加载到CPU寄存器的参数就放在任务栈里面，在任务创建的时候，预先初始化好栈。

OSTaskStkInit()函数在os_cpu_c.c中实现

任务创建好之后，我们需要把任务添加到一个叫就绪列表的数组里面，表示任务已经就绪，系统随时可以调度。

OS系统初始化

OSInit()函数在文件os_core.c中实现，主要做的工作就是初始化μC/OS-III中定义的全局变量。

启动系统

OSStart()在os_core.c中定义

OSStartHighRdy()

在文件os_cpu_a.s实现

触发PendSV异常

指令名称	作用
EQU	给数字常量取一个符号名，相当于 C 语言中的 define
AREA	汇编一个新的代码段或者数据段
SPACE	分配内存空间
PRESERVE8	当前文件栈需按照 8 字节对齐
EXPORT	声明一个标号具有全局属性，可被外部的文件使用
DCD	以字为单位分配内存，要求 4 字节对齐，并要求初始化这些内存
PROC	定义子程序，与 ENDP 成对使用，表示子程序结束
WEAK	弱定义，如果外部文件声明了一个标号，则优先使用外部文件定义的标号，如果外部文件没有定义也不出错。要注意的是：这个不是 ARM 的指令，是编译器的，这里放在一起只是为了方便。
IMPORT	声明标号来自外部文件，跟 C 语言中的 EXTERN 关键字类似
B	跳转到一个标号
ALIGN	编译器对指令或者数据的存放地址进行对齐，一般需要跟一个立即数，默认表示 4 字节对齐。要注意的是：这个不是 ARM 的指令，是编译器的，这里放在一起只是为了方便。
END	到达文件的末尾，文件结束
IF,ELSE,ENDIF	汇编条件分支语句，跟 C 语言的 if　else 类似

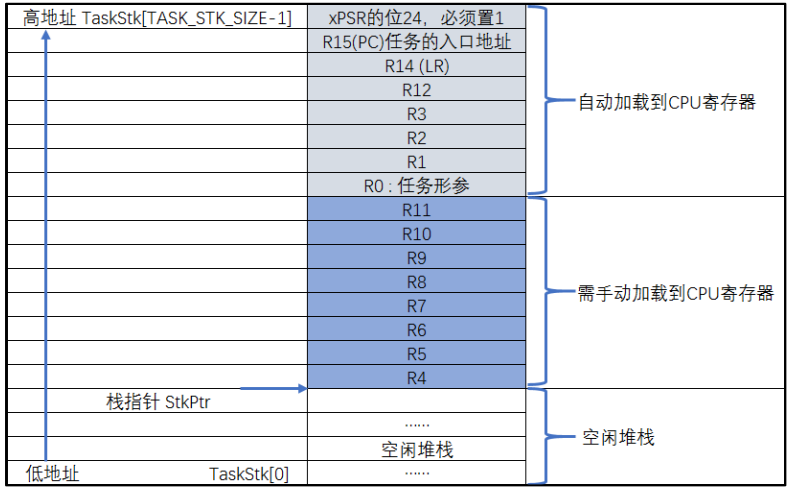
名字	功能描述
PRIMASK	这是个只有单一比特的寄存器。在它被置 1 后，就关掉所有可屏蔽的异常，只剩下 NMI 和硬 FAULT 可以响应。它的默认值是 0，表示没有关中断。
FAULTMASK	这是个只有 1 个位的寄存器。当它置 1 时，只有 NMI 才能响应，所有其他的异常，甚至是硬 FAULT，也通通闭嘴。它的默认值也是 0，表示没有关异常。
BASEPRI	这个寄存器最多有 9 位（由表达优先级的位数决定）。它定义了被屏蔽优先级的阈值。当它被设成某个值后，所有优先级号大于等于此值的中断都被关（优先级号越大，优先级越低）。但若被设成 0，则不关闭任何中断，0 也是默认值。

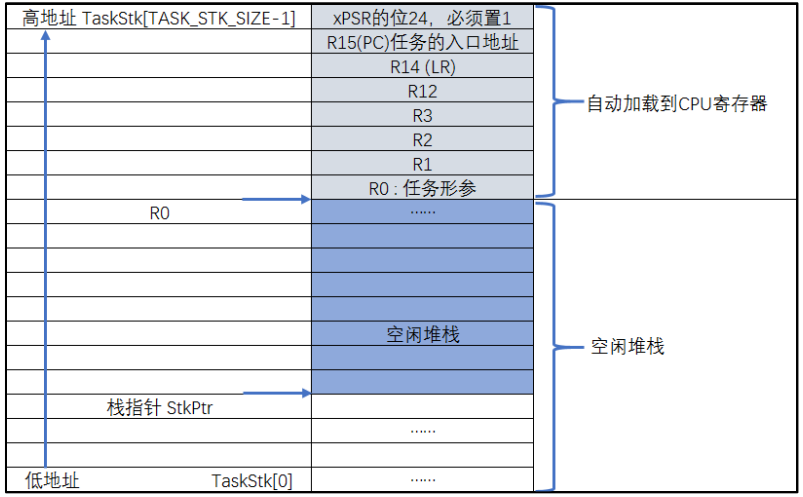
任务切换

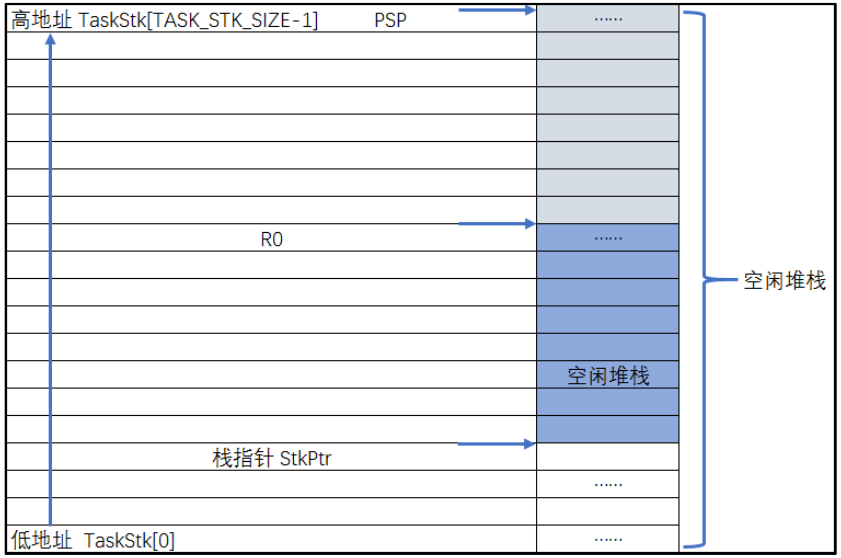
当调用OSStartHighRdy()函数，触发PendSV异常后，就需要编写PendSV异常服务函数，然后在里面进行任务切换。

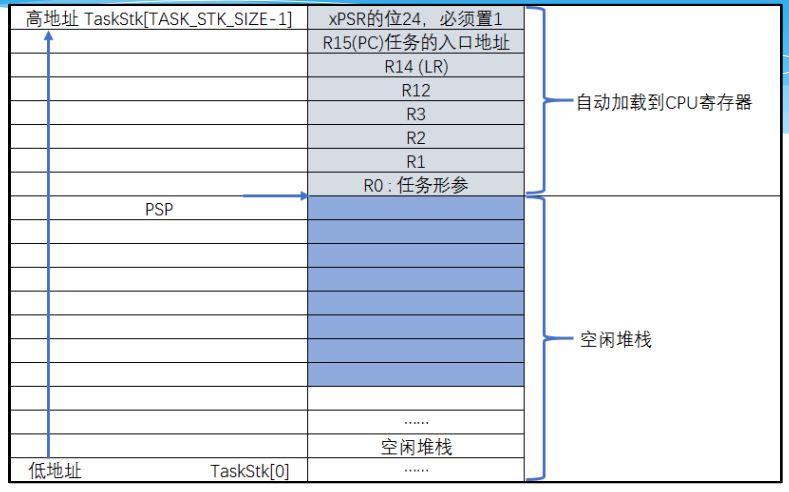
PendSV异常服务中主要完成两个工作，一是保存上文，即保存当前正在运行的任务的环境参数；二是切换下文，即把下一个需要运行的任务的环境参数从任务栈中加载到CPU寄存器，从而实现任务的切换。

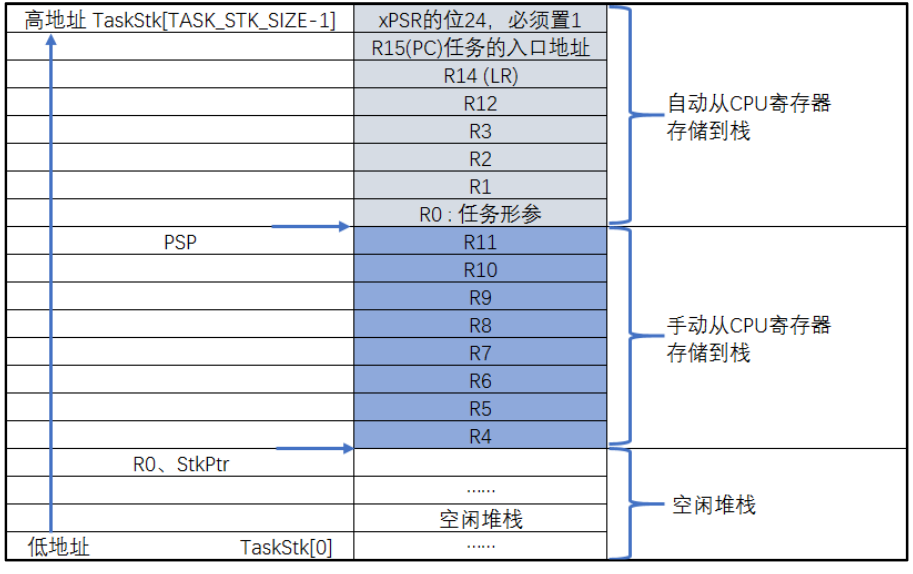
PendSV异常服务中用到了OSTCBCurPtr和OSTCBHighRdyPtr这两个全局变量，这两个全局变量在os.h中定义


高地址 TaskStk[TASK_STK_SIZE-1]
xPSR的位24，必须置1
R15(PC)任务的入口地址
R14 (LR)
R12
R3
R2
R1
R0 : 任务形参
R11
R10
R9
R8
R7
R6
R5
R4
栈指针 StkPtr
.....
空闲堆栈
低地址 TaskStk[0]
.....
自动加载到CPU寄存器
需手动加载到CPU寄存器
空闲堆栈


高地址 TaskStk[TASK_STK_SIZE-1]
xPSR的位24，必须置1
R15(PC)任务的入口地址
R14 (LR)
R12
R3
R2
R1
R0 : 任务形参
R0
.....
空闲堆栈
栈指针 StkPtr
.....
低地址 TaskStk[0]
.....
自动加载到CPU寄存器
空闲堆栈


高地址 TaskStk[TASK_STK_SIZE-1]
PSP
.....
R0
.....
空闲堆栈
栈指针 StkPtr
.....
低地址 TaskStk[0]
.....
空闲堆栈


高地址 TaskStk[TASK_STK_SIZE-1]
xPSR的位24，必须置1
R15(PC)任务的入口地址
R14 (LR)
R12
R3
R2
R1
R0 : 任务形参
PSP
.....
空闲堆栈
低地址
TaskStk[0]
.....
自动加载到CPU寄存器
空闲堆栈


高地址 TaskStk[TASK_STK_SIZE-1]
xPSR的位24，必须置1
R15(PC)任务的入口地址
R14 (LR)
R12
R3
R2
R1
R0 : 任务形参
PSP
R11
R10
R9
R8
R7
R6
R5
R4
R0、StkPtr
.....
空闲堆栈
低地址
TaskStk[0]
.....
自动从CPU寄存器存储到栈
手动从CPU寄存器存储到栈
空闲堆栈

指令名称	作用
MRS	加载特殊功能寄存器的值到通用寄存器
MSR	存储通用寄存器的值到特殊功能寄存器
CBZ	比较，如果结果为 0 就转移
CBNZ	比较，如果结果非 0 就转移
LDR	从存储器中加载字到一个寄存器中
LDR[伪指令]	加载一个立即数或者一个地址值到一个寄存器。举例：LDR Rd, = label，如果 label 是立即数，那 Rd 等于立即数，如果 label 是一个标识符，比如指针，那存到 Rd 的就是 label 这个标识符的地址
LDRH	从存储器中加载半字到一个寄存器中
LDRB	从存储器中加载字节到一个寄存器中
STR	把一个寄存器按字存储到存储器中
STRH	把一个寄存器存器的低半字存储到存储器中
STRB	把一个寄存器的低字节存储到存储器中
LDMIA	加载多个字，并且在加载后自增基址寄存器
STMIA	存储多个字，并且在存储后自增基址寄存器
ORR	按位或
BX	直接跳转到由寄存器给定的地址
BL	跳转到标号对应的地址，并且把跳转前的下条指令地址保存到 LR
BLX	跳转到由寄存器 REG 给出的的地址，并根据 REG 的 LSB 切换处理器状态，还要把转移前的下条指令地址保存到 LR。ARM(LSB=0)，Thumb(LSB=1)。CM3 只在 Thumb 中运行，就必须保证 reg 的 LSB=1，否则一个 fault 打过来

编写main()函数

仿真实验

THANKS

淘宝：fire-stm32.taobao.com

论坛： www.firebbs.cn

扫描进入淘宝店铺