基于ARM 的嵌入式系统中断处理机制研究

金　浩，韩江洪，李阳铭
（合肥工业大学 计算机与信息学院　安徽 合肥　230009）

摘　要：实时嵌入式系统要求嵌入式处理器具有支持多任务、中断响应时间短的特点，基于ARM 体系结构的处理器支持实时多任务，但中断异常响应机制比较复杂，效率不高。以LPC2119 为硬件平台分析了基于ARM 嵌入式系统的异常响应机制，阐明了关键词“_irq”的作用；然后探讨了μC/OS II嵌入式操作系统下可重入性中断处理函数的设计思路；最后，在分析操作系统的基础上，针对ARM 体系结构特点，对μC/OS II下的可重入性中断处理函数进行了优化。

关键词：状态寄存器；异常模式；中断处理；任务切换；可重入函数
中图分类号：TP316.2　　**文献标识码**：A　　**文章编号**：1004 373X（2005）22 001 03

Research of Exception Handler Mechanism for Embedded System Based on ARM

JIN Hao, HAN Jianghong, LI Yangming
(School of Computer and Information, Hefei University of Technology, Hefei, 230009, China)

Abstract: Real time embedded system required CPU to support multi tasks and respond to interrupt exceptions quickly. ARM processors did well in supporting real time multi tasks, but the principle of exception processing was so complex that whole efficiency was low. The mechanism in which ARM processor responded to sorts of exceptions was analyzed, and functions of "_irq" was also introduced. Then the way in which reentrant interrupt service program should be designed had been discussed on basis of "μC/OS_II" system. At last the sequence how reentrant interrupt service works was optimized based on characteristics of "μC/OS II" and the optimized scheme was put up in detail.

Keywords: state register; exception mode; exception handler; task switch; reentrant function

1　引　言

随着嵌入式系统在实时控制领域的广泛应用，系统对嵌入式微处理器和嵌入式软件的要求越来越高。不仅要求系统支持实时多任务，支持抢占式任务切换；而且具有较短的中断响应时间，及时处理现场环境的实时请求。

ARM（Advanced RISC Machines）是ARM 公司设计开发的通用32 位RISC 微处理器体系结构，目前被公认为业界领先的一种体系结构。这种体系结构支持实时多任务，提供专用通道处理外部请求，基于ARM 的微处理器功能强大且功耗低，不仅适合于网络通信设备，在各类工业级现场控制系统中也得到了广泛应用。

ARM 内核采用先进的三级流水线技术，支持7 种处理器模式，中断当前指令流时，必须计算PC（Program Counter）值，并需要进行系统模式的切换，这使得各种异常信号的响应机制更加复杂。本文以一款基于ARM 7TDMI S 内核的嵌入式控制芯片LPC2119 为硬件平台，分析中断异常响应的基本原理，并对μC/OS II操作系统下的中断响应机制提出改进方案。

收稿日期：2005 08 06
基金项目：安徽省“十五”二期科技攻关计划项目（2001BA104G）

2　ARM 系统的中断异常响应机制

ARM 系统结构支持7 种处理器模式：用户模式、快中断模式、中断模式、管理模式、终止模式、未定义模式和系统模式，执行特定指令可以切换处理器模式，不同模式对应的堆栈空间也不相同[1]。ARM 系统结构响应多种异常事件，本文主要讨论IRQ 中断异常的响应机制。ARM 体系结构的中断异常响在具有如下几个特点[2]：

（1）响应中断时，必须转换处理器的工作模式，由任务所处模式转换到中断模式；

（2）退出中断处理函数时，必须计算返回地址且转换处理器模式，回到任务模式；

（3）使用C 语言调用中断处理函数时，必须按照ATPCS（ARM/Thumb Procedure Call Standard）处理寄存器。

下面依次介绍ARM 处理器响应中断异常时的工作流程：

系统响应中断请求后，即进入中断模式，内核按下面的步骤工作：将下一条指令的地址保存到中断模式的连接寄存器；复制CPSR 到中断模式SPSR；设置CPSR 寄存器

模式位10010，置IRQ禁止位，强制PC从相应的中断向量地址取指令，进入中断处理函数。这些工作是由ARM内核完成的，不需要用户程序参与。中断处理完成后，系统需要回到发生中断的模式，继续执行被中断的程序，中断处理函数要完成下面的工作：恢复状态寄存器CPSR；清除中断禁止标志；将LR中的值减去偏移量计算返回地址。这些工作必须由用户在中断处理函数中实现。

通过以上分析，可以发现异常处理函数与普通函数不同，在处理异常事件前和处理后需要进行一系列与ARM内核相关的操作，普通函数经过封装后才可以作为中断处理函数链接到异常向量表中[3]。标准的ARM指令编译器提供了一个用来声明中断处理函数的关键字_irq，使用此关键字声明的函数可以被编译器识别为中断处理函数，并且生成必要的接口代码[4]。

比较下面两段代码可以看出“_irq”的作用：

```
__irq void IRQHandler(void)
{
  volatile unsigned int *base=(unsigned int*)0x80000000;
  if(*base==1){C_int_handler();}
  *(base+1)=0;
}
```

上面的C语言代码经过编译后生成的汇编代码如下：

```
IRQHandler PROC
    STMFD    sp!,{r0  r4,r12,lr}  ;处理异常事件前的操作
    MOV      r4,#0x80000000
    LDR r0,  [r4,#0]
    SUB      sp,sp,#4
    CMP      r0,#1
    BLEQ     C_int_handler
    MOV      r0,#0
    STR      r0,[r4,#4]
    ADD      sp,sp,#4
    LDMFD    sp!,{r0  r4,r12,lr}
                              ;异常事件处理结束后的操作
    SUBS     pc,lr,#4
ENDP
```

编译后的代码在处理异常事件前保存现场信息(第一句代码)，处理异常事件后对现场信息进行恢复(最后两句代码)。因此，_irq关键字适合用于定义中断处理函数和快速中断处理函数。但是，_irq声明的异常处理函数在响应中断时，并没有保存SPSR寄存器，而且一直处在关中断的IRQ模式，这导致了中断处理函数是不可重入的，即系统在退出整个中断处理程序之前不能响应其他中断。

3 可重入中断处理函数的设计

3.1 操作系统下中断处理函数

嵌入式实时操作系统要求系统及时响应外界请求，而且中断处理函数应该是可中断、可重入的。μC/OS II操作系统是一种基于优先级的抢占式实时操作系统，当中断来临时，系统把正在执行的任务挂起，保护现场，进行中断处理；中断处理完成后，检查是否有新任务进入就绪状态，比较所有处于就绪状态的任务的优先级，通过任务切换去执行最高优先级的任务；如果没有新任务进入就绪状态，中断返回后继续执行原任务[5]。因此，退出中断处理函数时需要进行任务切换，而不是单纯地恢复保存的CPU现场，这一切换功能由OSIntExit()通过调用OSIntCtxSw()完成。

运行在ARM平台的操作系统进行中断处理需要考虑这样的问题：任务运行时，系统默认的任务模式是用户模式或系统模式，响应中断后，系统进入IRQ异常模式，两种模式使用不同的堆栈空间，现场寄存器如何保存到任务堆栈中？因而，在操作系统μC/OS II中实现可重入性中断需要重新设计中断处理函数接口，接口主要是对CPU各寄存器进行操作，最好使用汇编语言编写。一般设计方案的流程如图1所示。

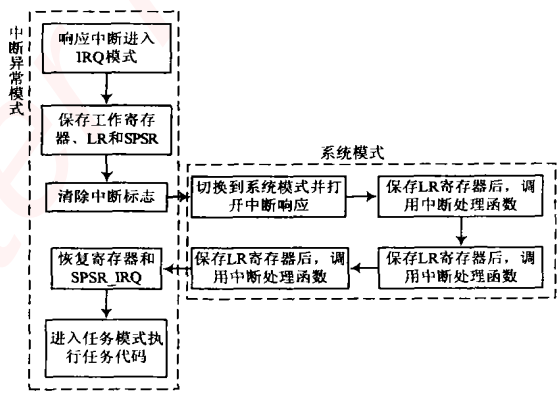

图1 可重入中断处理函数流程

响应中断后，保存CPU现场及SPSR寄存器到IRQ堆栈，并清除中断标志，使得系统可以继续响应其他中断信号，实现了中断处理的可重入性；然后切换到系统模式，保存现场后调用中断处理处理程序；最后回到中断模式，恢复寄存器后进入任务模式，执行处于就绪状态的最高优先级任务[6]。因为一般进程都是运行在用户模式的，而且用户模式与系统模式共用所有寄存器和堆栈，所以进入系统模式后现场保存到了任务堆栈中。这种设计方式能够实现可重入中断，但也带来了新的问题：每次响应中断系统都需要两次切换工作模式，两次保存CPU工作现场，既浪费了存储空间，又降低了整个中断响应过程的时间性能。

3.2 中断处理函数的改进方案

标准的中断处理函数接口存在时空浪费现象，下面讨论如何改进中断处理函数流程。由ATPCS规范可知系统调用C语言函数时，使用R0～R3来传递参数，C语言程序不会直接使用CPU寄存器，R4～R11是不会改变的，因而也就不需要入栈保存。系统响应中断时，寄存器是保存到IRQ模式堆栈的，切换到系统模式后，再次保存现场信

息到用户堆栈。两次保存的CPU 现场信息基本是相同的，因而其中一个保存操作不是必需的[7]。可以这样分析：系统从中断返回时，如果没有更高优先级的任务进入就绪状态，则不需要进行任务切换，只需将IRQ 模式堆栈中保存的CPU 现场信息恢复即可；如果有更高优先级的任务进入就绪状态，则需要进行任务切换，此时需要将CPU 现场信息保存到任务堆栈中。得到如下改进方案：

(1) 系统响应中断请求时，不保存所有寄存器，只将可能改变的寄存器（R0，R1，R2，R3）以及连接寄存器、状态寄存器保存到IRQ 模式堆栈；

(2) R0～R3 寄存器只保存到IRQ 堆栈，进入系统模式后，不再重复保存；

(3) 系统从中断响应返回时，先判断是否需要进行任务切换。如果需要，则将所有寄存器保存到任务堆栈，其中R0～R3 寄存器是从IRQ 模式堆栈拷贝到任务堆栈的，其他寄存器是直接在系统模式下保存入栈的；如果不需要进行任务切换，则可以在IRQ 模式下直接将现场恢复，改变CPSR 状态，最后进入任务模式继续执行。

中断函数入口代码重新设计如下：

```
SUB       LR,LR,#4                    ;计算返回地址
STMFD     SP!,{R0 R3,R12,LR}          ;保存寄存器和返回地址
MRS       R3,SPSR
STMFD     SP!,{R3,SP,LR}^
                                      ;保存状态寄存器、堆栈指针和LR
MSR       CPSR_c,#(EnInt|SYS32Mode)
                                      ;切换到系统模式
BL        $IRQ_Exception_Function
                                      ;调用C 语言的中断处理程序
BL        OSIntExit                   ;需要重新定义OSIntExit
MSR       CPSR_c,#(NoInt|IRQ32Mode)   ;切换回irq 模式
LDMFD     SP,{R3,SP,LR}^              ;恢复用户状态的R3,SP,LR
LDR       R0,=OSTCBHighRdy
LDR       R0,[R0]
LDR       R1,=OSTCBCur
LDR       R1,[R1]
CMP       R0,R1
ADD       SP,SP,#4*3;
MSR       SPSR_cxsf,R3
LDMEQFD   SP!,{R0- R3,R12,PC}^
                                      ;不进行任务切换，直接恢复寄存器
LDR       PC,=OSIntCtxSw              ;进行任务切换
```

在需要进行任务切换时才调用OSIntCtxSw（）函数，主要功能是保存被中断任务的现场信息、恢复并运行最高优先级任务。具体流程如图2 所示，整个过程分为3 大步骤：首先，将被中断任务的现场信息（17 个寄存器内容）全部入栈（任务堆栈）保存；然后，取得最高优先级任务的指针；最后，恢复最高优先级任务的现场，并运行新任务。

4 结 语

基于ARM 体系结构的处理器支持实时多任务，但中断响应需要切换工作模式，占用不同堆栈空间，使得中断处理过程非常复杂。μC/OS II操作系统是一种基于优先级的抢占式实时操作系统，本文在深入分析操作系统原理的基础上，对中断异常响应机制作了改进，提出了优化方案。在数字矿井项目中，基于LPC2119 的开发板作为嵌入式网关工作，运行μC/OS II操作系统及通讯软件，采用外部中断方式接收现场总线数据，验证了本文改进方案的实用性和高效性。

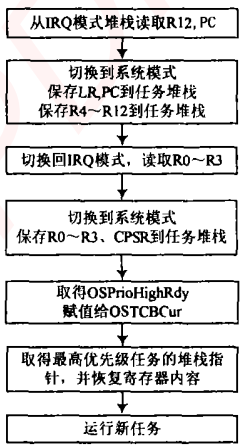

图2 OSIntCtxSw()函数流程图

参 考 文 献

[1] Philips Semiconductors. Data Sheet LPC2100. 2003.

[2] 费浙平. 基于ARM 的嵌入式系统程序开发要点[J]. 单片机与嵌入式系统应用，2003，(11)：82 85.

[3] 蔡国永，王志华. 异常处理技术及其编程应用[J]. 桂林电子工业学院学报，2000，(1)：112 116.

[4] ARM Architecture Refrence Manual [EB/OL]. Advanced RISC Machine Ltd. (ARM)，2000.

[5] [美] Jean Labrosse J. 嵌入式实时操作系统μC/OS II[M]. 第2 版. 邵贝贝译. 北京：北京航空航天大学出版社，2003.

[6] 周立功. ARM 微控制器基础与实践[M]. 北京：北京航空航天大学出版社，2003.

[7] 来卫国. ARM 内核的中断技术[J]. 单片机与嵌入式应用，2002，(5)：124 126.

[8] 蒋亚群，张春元. ARM 微处理器体系结构及其嵌入式SoC[J]. 计算机工程，2002，(11)：4 6.

[9] 王力翔，冀力强. 嵌入式CPU 异常处理的设计及其硬件实现[J]. 半导体技术，2001，(8)：27 30.

作者简介 金 浩 男，1980 年出生，合肥工业大学硕士研究生。主要研究方向为嵌入式系统和计算机控制。

韩江洪 男，1954 年出生，合肥工业大学教授、博士生导师。主要研究方向为计算机控制、计算机网络、汽车电子等。

李阳铭 男，1980 年出生，合肥工业大学硕士研究生。主要研究方向为计算机控制。