Implementation of Interrupt Manager in Small - Scaled Embedded Operating System

PU Hanlai LIU Hao LING Ming

(*National Engineering Research Center for Application Specific Integrate Circuit System, Southeast University, Nanjing* 210096, *P. R. China*)

Abstract: At present the general implementation of interrupt manager in small-scaled embedded operating system exhibits high dependence on hardware, poor safety and real time. According to these problems, we raised a new implementation. It decreases the dependence on hardware through layered design. It provides extendable interrupt priority by using soft interrupt vector. It improves ability in real time interrupt management by changing the policy.

Key words: embedded; OS; operating system; interrupt manager; implementation

EEACC: 6330;6210

小型嵌入式操作系统中断管理的实现

浦汉来 刘 昊 凌 明

(东南大学国家专用集成电路系统工程技术研究中心,南京 210096)

摘要:目前小型嵌入式操作系统的中断管理实现的不足主要表现在:对硬件依赖性强;安全性差;实时性差等。本文针对这些问题提出了一种中断管理实现。它采用分层设计,降低硬件依赖性;通过构建软中断向量表,提供可扩展的中断优先级;调整中断管理策略,改善实时性中断的响应。

关键词:嵌入式;操作系统;中断管理

中图分类号:TP316.1/.5 **文献标识码**:A **文章编号**:1005-9490(2003)02-0218-04

近年来,随着硬件复杂性、多样性和应用复杂性的增加,软件开发工作量急剧增长。传统的开发模式已经不能适应系统复杂性的增长,而嵌入式操作系统作为嵌入式软件的运行平台和开发平台,它的引入,极大的方便了嵌入式软件的开发和维护。

然而,不同规模和配置的嵌入式系统,对操作系统的具体要求也不一样。在一些规模很小的嵌入式系统中,比如说CPU主频仅有十几兆、RAM大小在一兆左右的系统,操作系统的设计往往很难在高效性和可靠性之间取得平衡。这一点在操作系统的底层核心模块中断管理上体现的非常明显。

很多这种系统的中断管理实现存在以下问题:① 汇编代码含量高,对硬件依赖性强,移植工作量大;② 安全性差,为了降低复杂度,对硬件异常不作处理或只是简单处理,这同时也使系统调试中的问题定位变得很困难;③ 效率低,在中断处理过程中屏蔽全部中断的时间过长,或者相同优先级的中断不能互相抢占;④ 用户接口定义不明确,使用起来很不方便。这些问题在一些以UCOS-II为典型的嵌入式操作系统中尤其突出。虽然这些问题在Linux等操作系统中已经有了非常好的解决方案,但对于小型嵌入式系统而言,这些方案的代码量过大,对系统配置要求也过高,生搬硬套过来,效率上也会打很大的折扣。

收稿日期: 2003-03-21

作者简介: 浦汉来(1980-),男,东南大学国家专用集成电路工程技术研究中心硕士研究生,研究方向为专用集成电路系统设计,pessia@163.com.

因此，针对上述问题，本文提出了一种相应的中断管理实现。同时，为了提高实时性，在中断处理过程中，尽量减少了中断屏蔽的时间；为了增强可移植性，采用了分层设计，将与硬件相关代码独立开来。

本文第一节介绍该中断管理框架结构和特点；第二节介绍它的主要数据结构；第三节介绍板级支持的实现；第四节介绍中断管理流程；第五节是总结与展望。

1　中断管理框架结构和特点

该中断管理的框架结构如图1。它主要分为两个部分，即板级支持和中断管理。板级支持实现与硬件相关的功能，比如统一中断入口，屏蔽和恢复全部中断等等；中断管理属于上层模块，负责实现中断嵌套、中断识别以及中断处理等功能。通过这样的分层设计，将与硬件相关的处理封装在板级支持这一层中，而上层的中断管理只需要调用板级支持提供的接口，便可实现相应的管理功能，这样既提高了该系统的移植性，又增强了它的可维护性。

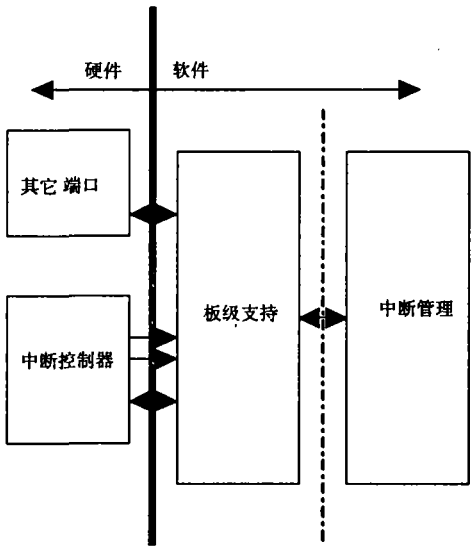

图1　中断管理框架结构图

2　主要数据结构

对开发人员来说，很多小型嵌入式系统中，采用的芯片对中断优先级的支持不是非常完善。比如说，有的芯片根本没有定义中断优先级，所有中断都是同一级别的；有的芯片在同一优先级的中断信号上包含了多个中断源等等，这都给开发人员带来了管理上的困难。该中断管理系统通过定义主次优先级来解决这些问题。也就是说，每个中断都有两个优先级，其中，主优先级由硬件的中断控制器来决定，而次优先级由用户定义，决定该中断在同一主优先级的一组中断中的优先级。通过主次优先级，可以构成一个二维的软中断向量表（如图2），它是对硬件提供的中断向量表的补充和完善。

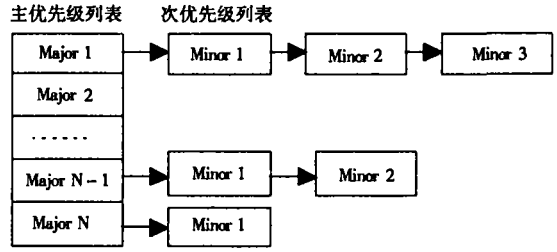

图2　主次优先级构成的二维软中断向量表

软中断向量表中的每个次节点都代表着用户申请的一个中断。当主节点没有对应的次优先级列表的时候，它本身也可以作为一个中断，这种情况下，它通常代表着硬件异常。

该中断管理系统为每个中断定义了一套处理函数接口（如图三所示），用户在申请中断的时候，只需要填充相应的函数接口，减少了其他代码的修改。同时，这样用户只需关心该中断本身的处理便可以了，提高了工作效率。

这套接口中，check_opt()用于检查该中断是否已经发生，这可以区分同一主优先级下的不同次优先级中断；enable_opt()和 disable_opt()用于实现该中断的具体开关操作，这样做可以将所有中断的开关函数接口统一起来；handle_opt()则是该中断的具体处理函数。

```
struct int_operations{
    int( * check_opt)(void);
    void( * enable_opt)(void);
    void( * disable_opt)(void);
    void( * handle_opt)(void);
};
```

图3　中断函数接口

3　板级支持实现

该中断管理系统中，板级支持实现与硬件相关

的功能,包括统一中断入口,屏蔽和恢复全部中断等等操作。

为了提高系统的可移植性,向上层提供统一的中断入口便必不可少。一般来说,每种芯片都会拥有多种不同的异常栈帧定义,并且变化通常很大。该中断管理系统通过填充硬件中断向量表来实现了中断入口的统一。

硬件中断向量表中对应的每个入口函数大致包含以下几个处理步骤:① 将当前堆栈寄存器值放入函数调用帧;② 将该中断的主优先级的值放入函数调用帧;③ 调用中断处理函数 void soft_ int_ handle (void * data, int priority);④ 销毁异常栈帧。

函数 soft_ int_ handle()便是上层中断管理所定义的统一中断入口函数。不同的异常栈帧结构,通过 data 参数传递给用户,而用户则根据不同的栈帧进行相应的处理。priority 参数是用来进行中断嵌套处理的。

当然,函数 soft_ int_ handle()中还必须包含任务运行环境的保存和恢复操作,但由于它们的具体实现是由具体的操作系统来决定,这里便不再详细描述。

为了上层实现临界区操作,板级支持还提供了屏蔽和恢复全部中断两个操作。因为有的芯片是通过修改状态寄存器来屏蔽和恢复全部中断;有的可以通过中断控制寄存器来实现,通过板级支持的封装,实现了上层临界区操作不受硬件影响,移植时不用再做修改。

4 中断管理流程

中断管理属于上层模块,它负责实现中断嵌套、中断识别以及中断处理等功能。图 4 和图 5 简要描述了中断管理流程的实现,即函数 soft_ int_ handle()的流程实现。

如图 4 所示,首先判断中断是否嵌套,分以下两种情况讨论:

1) 如果没有发生嵌套

进行正常的次优先级中断处理流程(如图 5)。即如图所示,将嵌套数加 1、更新优先级,调用中断处理模块。完毕后便可以恢复优先级、嵌套数减 1。

2) 如果中断发生嵌套

首先判断该中断与正在处理的中断的主优先级是否相同,如果相同,则意味着目前的中断处理过程中,发生了中断遗漏,在高次优先级中断处理流程中进行处理。它与图 5 的流程非常相似,只有两处不同:第一,如果次优先级列表为空,则直接退出。第二,图中循环的范围发生了变化,即它只搜索比当前正在处理的中断拥有更高次优先级的次节点。如果有高次优先级中断发生,则调用其处理函数,没有则退出。

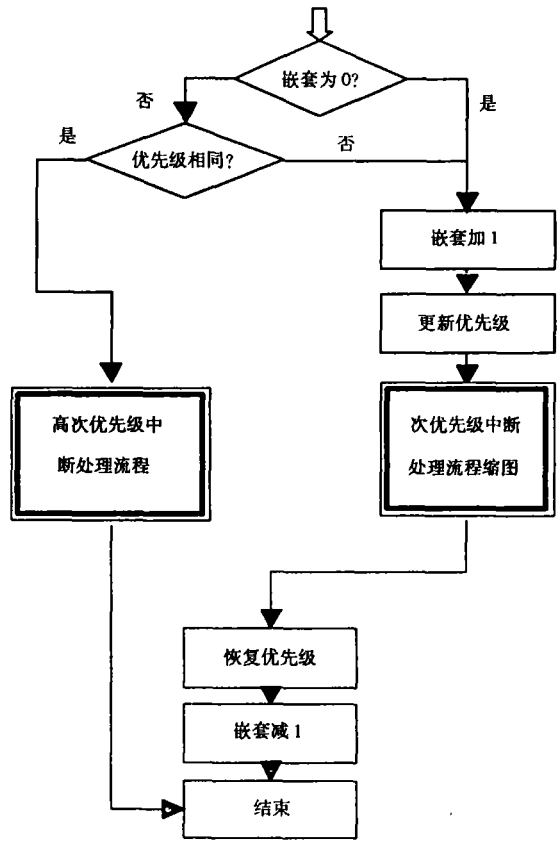

图 4 中断管理流程的实现

如果发现该中断与正在处理的中断的主优先级并不相同(更确切的说,应该是主优先级更高),则进行正常的次优先级中断处理流程。

需要说明的是,图 4 中所示"结束"这一步骤,对于不同的操作系统,可能有着不同的意义,其实现也不尽相同。在这一步里,一般的实现都通过判断嵌套数是否为 0 来决定如何退出:① 如果不为 0,意味着目前系统仍然在中断处理过程中,则直接退出;② 如果为 0,系统可以引发一次任务调度,然后再退出。

下面来描述一下中断处理流程缩图中的内容(如图 5 所示)。首先判断该主优先级对应的次优先级列表是否为空,分为两种情况讨论:

1) 次优先级列表为空,这时应该检测该主节点

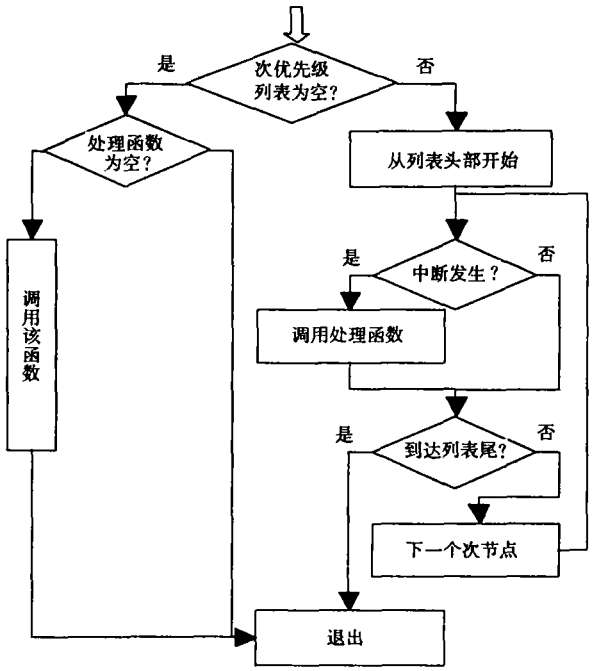

图 5　次优先级中断处理流程缩图

是否拥有自己的处理函数，如果有则执行它。这种情况主要用于错误异常的处理，比如除 0 异常、寻址错误异常等等。

2) 次优先级列表不为空时，则循环检查各个次节点对应的中断是否已经发生(通过调用 check_opt 函数)，对已发生的中断调用其处理函数。

在图 4 和图 5 中，很多环节都处于临界区之中。由于这些环节所需的处理时间都非常短(比如"嵌套加 1"等等)，所以，目前采用关闭中断作为进入临界区的方法。这样做，既保证了代码的可重入性，又节省了临界区的时间。

当然，中断管理层也为用户提供了一系列简单易用的操作函数接口，如申请中断、删除中断、使能中断等等，本文便不再赘述。

5　总结与展望

本文介绍的中断管理系统是专为小型嵌入式系统设计的。它针对传统的中断管理设计的缺点进行了大量的改进和完善。它采用分层设计，在提高了系统的可移植性的同时，也保证系统的高效性和高可靠性。但是，该系统也不可避免的存在一些不足之处，首先板级支持的接口定义不完善，降低了系统的可移植性；其次，目前的数据结构给开发人员提供的自定义接口不多，降低了系统的可扩展性。鉴于此，在后续工作中，将会针对上述诸点，进一步改进和完善该系统。

参 考 文 献

[1] (美) Uresh Vahalia 著，聊鸿斌 曲广之 王元鹏 等译，UNIX 高级教程系统技术内幕[M]. 清华大学出版社. 2000,(3).

[2] (美) William Stallings 著，魏迎梅 王涌 等译，操作系统——内核与设计原理(第四版)[M]. 电子工业出版社. 2001,(1).

[3] ANDREW S. TANENBAUM ALBERT S. WOODHULL 著，王鹏 尤晋元 朱鹏 敖青云 译校，操作系统：设计与实现(第 2 版)(上、下册)[M]. 电子工业出版社. 2000,(3).

[4] DANEL P. BOVET & MARCO CESATI 著，陈莉君 冯锐 牛欣源 译，深入理解 LINUX 内核[M]. 中国电力出版社. 2001,(1).

[5] 陈莉君 编著，Linux 操作系统内核分析[M]. 人民邮电出版社，2000,(1).

[6] JEAN J. LABROSSE 著，邵贝贝 译，μC/OS - II——源码公开的实时嵌入式操作系统[M]. 中国电力出版社. 2001,(1).