一种基于单片机的抢占式实时嵌入式操作系统设计

于 坤 崔芮华 孟庆龙 陈堂功 河北工业大学电器研究所(300310)

Abstrat

Embedded system usually requires character of real time.So specific task scheduling algorithm is needed.Furthermore,embedded OS aims at specific situations.Different hardware and environment that the OS runs in corresponded different solution.With the analysis of system requirement,this article gives out a design idea of operation sytem on 80C51.

Keywords:preemptive kernel,priority inversion, hard realtime,thread

摘 要

嵌入式系统一般都有实时性的要求,需要选用特定的任务调度算法。同时,嵌入式操作系统针对性很强,不同的硬件环境选用的设计方案往往也不同。本文在分析系统需求的基础上,给出了一种基于 80C51 的操作系统的设计思路。

关键词:实时嵌入式操作系统,抢占式内核,优先级翻转,硬实时,线程

基于单片机的嵌入式系统一般都没有操作系统，但是当应用趋于复杂时,编写大量相关性很强的代码,即使是富有经验的开发人员也会感到力不从心。这时把应用分为多个独立的任务是一个解决问题的好方法。我们可以采用操作系统来进行任务管理。即使我们不采用操作系统,潜在地也要由用户自己实现多任务。仔细研究一下单片机操作系统实现的基本思想对开发人员而言都是很有好处的。

目前已有不少作者进行了这方面的讨论，但是一般都是相对弱实时的,采用的是非抢占式内核设计。由于实时性比较差,限制了系统的应用范围。但是采用抢占式内核,设计的复杂性和难度大大增加。本文试图通过考察实时操作系统的基本原理,给出一种在单片机上设计操作系统的思路。

1 单片机实时操作系统设计的一般问题

1.1 实时性要求

嵌入式系统必须满足系统的实时性要求。实时性分为软实时和硬实时。

硬实时任务使用抢占式内核。非抢占式内核只在内核中系统调用结束之后才可以切换任务，这确保了系统调用不会被别的任务进程所打断,即它是不可重入的。不可重入的系统调用降低了内核的复杂性,但是,也延迟了关键任务的执行。抢占式内核中必须把系统调用设计成可重入的，这时可以通过把内核中的关键数据的访问放入临界区来实现。

对于硬实时任务使用抢占式内核，同时根据事件发生的规律,采取不同的调度策略。

一般而言,对于周期性的、任务之间耦合程度较低的系统,可以采用单一速率调度算法 RMS。它的优先级是静态的,由周期决定,周期最短的优先级最高。

对于非周期性的或任务之间耦合度高的系统,可以考虑使用期限最近者优先调度 EDF。它以最后期限的顺序指定优先级,因此，任务的优先级是动态的。理论上,EDF 可以达到 100%的 CPU 利用率。但是实际上还应该考虑一些其它因素:任务切换时间、调度算法实现开销、任务间的数据依赖、优先级翻转问题等。

1.2 硬件环境的限制

嵌入式系统必须在有限的内存空间和简单的内存管理单元 MCU 条件下可靠工作,因此操作系统应尽量地短小精悍。通用操作系统中常用的宏展开,用在嵌入式系统中就不适合,可以代之以函数调用,以实现代码的复用。这是以牺牲一定的速度为代价的。只靠速度并不能保证实时性,事件必须在可预测的时间内得到处理,在最坏的情况下,实时性也应得到满足。能保证最坏情况下时间得到及时处理的设计都是合理的。

由于硬件的限制,任务和内核常常都工作在实模式。同时大多数单片机缺乏必要的内存管理硬件。这时任务以线程的形式实现。有的单片机有内存管理硬件,但出于效率上的考虑,有的实时系统并没有使用。

1.3 中断响应

中断延迟时间是指从硬件中断发生到系统开始执行中断服务例程 ISR 所花费的最大时间,它直接关系系统对异步事件做出即时响应的能力,是评价实时系统性能的重要标准。为了确保对关键数据的原子操作,屏蔽中断时必须的。但是在此过程中的中断屏蔽可能造成中断的丢失。同时,ISR 阻塞了任务调度,影响了系统对关键任务的响应时间。因此,ISR 应该尽快地结束,而把进一步的处理交给上层来完成。

1.4 互斥

内核、系统调用、任务都要用互斥来保护临界资源、同步相互之间的关系,因此,系统必须实现互斥。互斥可由如下几种方法实现:一是关中断。这种方法最简单,但对于分布式系统和内部中断无效。同时,如果关中断时间太长,将影响实时性。其次是使用信号量或锁机制。对锁的操作需要使用硬件提供的功能,比如说交换指令或中断屏蔽。对于任务之间的互斥,还可以在内核中设置软锁。当任务进入临界区时对锁数据结构置 1 加锁,退出时置 0 解锁。任务调度例程查看锁状态决定是否能进行任务切换。设想当在加锁后当前任务被中断打断,若已加锁,任务调度程序不进行任务切换,从而实现了原子操作。这种方法比关中断效率高。缺点是使所有其他任务,包括不使用该临界资源的任务也不能运行,因此,临界区应尽可能得短。该方法适合于用户信号量的实现。

2 基于 80C51 的单片机实时操作系统的实现

本系统主面向工业控制,系统规模较小,一般任务数在 5~10 左右。系统可靠性、安全性要求较高,设计时必须保证毫秒级的实时性。同时事件发生随机性很强。因此采用了基于最后期限的多优先级调度算法、抢占式多任务内核。由于 C51 没有特权级和内存管理模块,任务管理采用实地址线程模式。没有使用微内核设计,实际上也没有典型意义上的用户态。为了提高可移植性,任务调度和中断处理用汇编语言实现,内核其它部分由 C 语言实现。

2.1 中断处理流程

80C51 有两级中断优先级，使用中我们把所有中断源设定为低优先级，简化了设计。为了提高中断响应速度，采用三级处理方式，中断响应-中断服务例程 ISR-设备驱动任务。

中断响应保存任务现场，并根据中断源选择中断处理例程 ISR。reenter 记录中断嵌套层次，每次进入中断处理 reenter 加 1，当中断服务 ISR 退出时 reenter 减 1。当 reenter 为 0 时，所有中断处理完毕，重新调度任务。ISR 分析中断源，对于处理简单、时间比较急迫的事件，ISR 实现，而对于较为复杂的事件处理，则 ISR 交相应任务完成。

退出内核时要进行任务调度，这通过堆栈指针指向 current 线程来实现。中断可能改变任务的优先级，由 current 指示优先级最高的任务。并且在开中断的任意点都可以响应另一次中断。

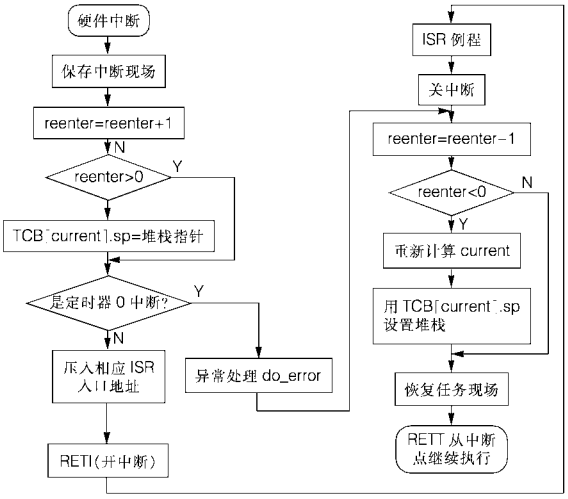

图 1 流程图

2.2 任务的实现和任务调度

设备驱动程序以线程的形式实现，线程由任务控制块和可执行代码、私有堆栈构成。内核中还设置全局变量区，由所有线程共享。线程由任务表项 TCB 标识。TCB 结构如下：

```
struct TCB
{int deadline;
 unsighed state:1;
 unsighed rt:1;
 int sp;
 struct TCB * prev,next;
 struct TASK * p_task;
 proc_t * do_error;
 }TCB[n_tasks];
```

任务表项存储在内核的任务表中，系统最大可支持 n_tasks 个任务，一般 n_tasks 不超过 15。

根据任务的性质和状态，任务表项被链接到以下 4 个双向链表中：硬实时就绪队列、软实时就绪队列、硬实时等待队列、软实时等待队列。

TCB 结构中，rt 标识任务是软实时还是硬实时。当需要由任务完成中断事件的后续处理时，ISR 将设置 deadline，填充一个 TASK 结构，并把 state 置为就绪，插入相应的就绪队列。TASK 结构存放任务说明。事件将被记录在 p_task 指示的 TASK 结构的队列中，当事件发生太快，任务来不及响应时，可以延期处理，保证中断不会丢失。

当最外层中断退出时还调用调度程序，设置 current 变量指向将获得 CPU 的任务线程。

设置 current 很简单。任务队列按最后期限的顺序链接，而最后期限就是任务的优先级，最后期限越近，优先级越高。硬实时任务优先级大于软实时任务。典型的运行环境的任务数不超过 10，因此把任务插入有序队列时只要简单地顺链查找，不需要复杂的算法，即可以保证较高的效率。

系统切换任务的同时，还必须对定时器 0 以该任务距 deadline 的时间开始计时。

线程的创建是静态的，所有线程在系统启动时都已设置好，线程不需要创建和删除。

任务调度采用多优先级 EDF。用户在规划系统时必须保证负载小于最大可承载负荷。万一硬实时中断超过最后期限时，定时器 0 将触发一个定时中断，启动异常例程，其入口地址存放在 TCB 的 do_error 域。异常不作为任务实现，它是内核中的一个模块，执行时关中断，并尽快地使系统恢复到安全状态。用户设置该模块，自己设定安全恢复操作。

2.3 同步、互斥及任务间通讯

内核控制路径间的互斥用关中断实现。线程间使用互斥信号量。信号量由软锁实现以提供更好的中断响应。由于没有用户地址空间的概念，线程间的通讯以共享内存的方式实现，安全性较低。用户必须仔细调试系统，确保系统的可靠运行。

3 结束语

设计可抢占的操作系统，需要做很多工作来避免由于无规律的抢占而可能引起的数据不一致。在本例中这一工作交给了用户线程自己。当读写共享数据时，必须使用信号量来互斥访问。实现可抢占式实时系统是复杂的，但是它也可以适应应用的更高要求，拓宽系统的使用范围，提高产品的竞争力。

参考文献

1 Wayne Wolf.Principles of Embeded Computing System Design[M].北京：机械工业出版社，2002

[收稿日期：2003.7.15]

广告索引