第 24 卷 第 3 期
2004 年 5 月

长安大学学报(自然科学版)
Journal of Chang'an University (Natural Science Edition)

Vol.24 No.3
May 2004

文章编号: 1671-8879(2004)03-0095-05

嵌入式实时操作系统内核的设计与实现

闫茂德[1,2], 贺昱曜[1], 陈金平[1], 许化龙[2]
(1. 长安大学 信息工程学院, 陕西 西安 710064; 2. 第二炮兵工程学院 自动控制系, 陕西 西安 710025)

摘 要: 针对广泛使用的微处理单元(MCU), 从任务 TCB 设计、任务的创建和删除、任务调度的算法及实现、中断管理、优先级管理和操作系统及用户的接口等 6 个部分出发, 提出了一个嵌入式实时多任务操作系统内核的实现方案, 给出了嵌入式操作系统内核软件的功能、位置、原理和技术参数等, 并给出在 TOSHIBA 815 16 位的微处理器上实现了该内核软件的应用实例, 实际运行于某型号的移动掌上电脑产品中, 运行情况良好。

关键词: 嵌入式操作系统; 任务控制块; 信号量; 操作系统内核
中图分类号: TN39; TP316.2 文献标识码: A

Design and implementation of embedded real-time OS kernel

YAN Mao-de[1,2], HE Yu-yao[1], CHEN Jin-ping[1], XU Hua-long[2]
(1. School of Information Engineering, Chang'an University, Xi'an 710064, China;
2. Department of Automatic Control, Second Artillery Engineering Institute, Xi'an 710025, China)

Abstract: Base on MCU, an embedded real—time operating system kernel was proposed from six aspects including task TCB design, task creating, task scheduling, interrupt handling, priority level management and the interface between operating system and user. The proposed design describes the position, function, principle and technical parameter of embedded real-time operating system kernel. This kernel was runned on the platform of TOSHIBA 815 microprocessor and used in the mobile PDA products. The applications of mobile PDA products was provided to illustrate the performance of the embedded real-time operating system kernel.

Key words: embedded OS; task control block; signal; OS kernel

0 引 言

在传统的嵌入式程序设计中, 如果有多个任务, 一般都是采用一个无限主循环, 依次对每个任务的条件进行判断, 一旦条件成立, 则立即执行相对应的模块完成任务。各模块按照编写的顺序依次执行。在这期间, 一旦发生中断, 由中断服务程序(ISR)负责处理, 整个过程如图 1 所示。

对于简单的应用, 这种方式是可以满足要求的, 随着应用的复杂化, 系统可能要同时控制和管理许多外设, 要求实时响应, 有很多任务要处理。如果采用传统的程序设计方法, 就会出现两个比较突

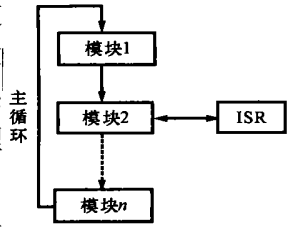

图 1 传统的嵌入式程序设计过程

收稿日期: 2003-09-09
基金项目: 全国优秀博士学位论文作者专项资金资助项目(200250)
作者简介: 闫茂德(1974—), 男, 陕西澄城人, 长安大学博士, 第二炮兵工程学院博士后.

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

出的问题:一是为了保证某项重要任务的实时性,就必须用 ISR 进行处理,这样一来,ISR 的执行时间就会变长;二是任务多,要考虑的各种可能也多,各种资源调度不当就会发生死锁,降低系统可靠性,同时程序量也加大。

解决这些问题的最好方法是使用实时操作系统(RTOS),使用 RTOS 可以有效地对任务进行调度,对各任务赋予不同的优先级可以保证重要任务及时响应。RTOS 也体现了一种新的应用程序设计思想和开放的框架,降低了程序的复杂度。目前的嵌入式实时操作系统价格昂贵,而且很多 RTOS 厂商要求用户在批量生产时要按产品数量交纳版税,对中小系统的成本影响很大。而事实上,中小应用系统并不需要一个完整的 RTOS,只需要 RTOS 的基本功能对任务进行有效地管理,基于这些考虑,设计一个RTOS 内核,采用 C 语言和汇编语言来实现,且系统调用以 C 函数库的形式提供,便于使用,这里给出基本方案。

1 实时操作系统主要实现的功能

嵌入式实时操作系统内核实现的功能包括:

(1)任务创建及调度功能,实现任务在挂起、运行、就绪、休眠和中断 5 个状态间的状态机转换及控制功能。

(2)静态内存管理机制,提供稳定的多任务内存堆栈管理功能。

(3)任务优先级管理功能。

(4)占先式时间片轮转调度的内核服务功能,使操作系统具有较高的实时性。占先式内核首先考虑的是系统的响应时间。最高优先级的任务一旦就绪,总能得到 CPU 的控制权。当一个运行着的任务使一个比它高优先级的任务处于就绪态时,当前任务对 CPU 的使用权就被剥夺,或者说当前任务就被挂起了,高优先级的任务立刻得到 CPU 的控制权。在相同优先级的任务之间采用时间片进行轮转调度,响应时间只取决于任务的切换时间。

(5)任务间通信功能,使应用层能通过设置或清除任务的信号,而改变任务的状态。

(6)定时器管理和时间片(TICK)中断功能,使应用层能根据需求创建任务的定时器,设置定时器,获取当前定时器值和清除定时器。其中 TICK 中断使系统能定时对定时器列表进行扫描,实现定时计数值的自减,清除已满定时器,触发任务调度。

(7)中断锁和中断解锁功能,使操作系统能保护对临界区数据的操作,避免当一个任务在处理一段共享资源时,被其他任务或中断服务程序打断,造成对共享资源的同抢(同时访问)及误操作。

2 位置与接口关系

TASK 操作系统界于 BOOT 区和 API 接口层之间,主要完成操作系统初始化、任务定义、任务起始,以及中断服务、任务调度、信号设置、定时器设置、TICK 中断、信号(优先级)获取、信号(定时器)清除等工作,并利用信号、定时器、优先级、中断、任务调度等机制为应用层提供支撑。向下主要与BOOT 区进行初始化接续。如图 2 所示。

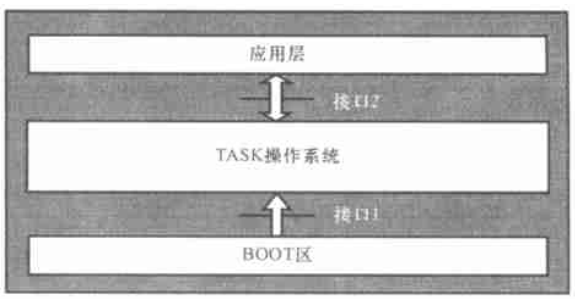

图 2 嵌入式操作系统 TASK 与上下层软件的接口关系

在接口 1,当底层驱动部分调用操作系统 main( )(即将 PC 指针指向主函数)后,操作系统先执行初始化操作,接着进行空任务创建,然后创建用户定制的任务,最后运行用户所需的任务,进入任务运行状态。

在接口 2,操作系统提供 API 函数与应用层间的相交互,包括:信号、定时器、任务、优先级等功能的使用。

3 实时操作系统内核的设计与实现

在操作系统中对任务进行调度有两种基本的调度方式:轮转调度和基于优先级的抢占式调度,前者无法处理各任务间可能存在的优先级差别,后者可以保证高优先级任务的实时响应。但后者也同时带来了新的问题:当系统中存在几个相同优先级的任务时,就有可能使某一任务独占处理器,直至完成。为此在该方案中采用了基于优先级的时间片轮转调度方式,即在不同优先级的任务之间以抢占方式进行调度,在相同优先级的任务之间采用时间片进行轮转调度。

3.1 任务 TCB 设计

任务 TCB 结构体所含内容是对任务特性的精确描述,在控制处理任务状态中至关重要。本模块

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

中设计的任务 TCB 结构体包含栈指针、栈大小、任务信号、等待信号、任务优先级、任务函数名、TCB 双向链表等。通过栈和栈大小给定任务特性值的存储位置便于查找到任务 TCB。通过任务信号使任务能通过信号设置从就绪态转向运行态等。通过等待信号使任务能自动挂起。通过任务优先级使系统具有按优先级高低进行任务 TCB 链表存储和任务调度的功能。通过任务函数名使系统能找到任务函数, 运行应用层业务。而通过一个紧凑的 TCB 双向链表数据结构成功实现任务 TCB 的级联存储。

```
typedef struct task-tcb-struct {
    void              *sp;          / * 栈指针              */
    void              *stack-limit; / * 栈大小              */
    unsigned short    slices;       / * 任务片              */
    task-sigs-type    sigs;         / * 任务信号            */
    task-sigs-type    wait;         / * 等待信号            */
    task-priority-type pri;         / * 任务优先级          */
    void              ( * task)( );  / *任务函数名           */
    struct                          / *任务 TCB 链表        */
    {
        struct    task -tcb-struct  * next-ptr;
        struct    task -tcb-struct  *prev-ptr;
    }link;
    #if defined FEATURE-TASK-EXTENDED-CONTEXT
        void              *ext;     / * 用于 TCB结构拓展    */
    #endif
} task -tcb-type;
```

3.2　任务优先级管理

在实时多任务操作系统中, 为了突显任务的相对重要性, 实现任务按优先级的时间片轮转调度方式, 引入了任务优先级管理。当低优先级任务正在执行, 高优先级任务已经就绪时, 如果高优先级任务被设置信号, 则系统就挂起低优先级任务, 转入执行高优先级任务。系统根据应用层任务的需求, 分别设置各任务的初始优先级, 在系统运行中, 各任务的优先级不变, 实现静态优先级管理。系统定义空闲任务的优先级为 0, 任务优先级最高为 255。

当任务起始后, 在任务 FOR 循环中首先执行的是等待程序, 等待任务信号的到来, 如果没有等待到任务信号, 则 FOR 循环无限循环执行等待程序, 即任务处于挂起态, 直到中断程序中断该状态, 并给任务设置信号, 这时, 任务才会执行等待程序后面的程序段, 进行相关事件处理。

创建任务函数 Create-task(　); //任务创建

起始任务函数 start-task(　); //任务起始

3.3　任务调度

由于系统采用 C 语言实现的程序, 所以主核心库中已经包含了 main(　)函数, 在 main(　)函数做完操作系统初始化后, 自动调用 Init-task(　)函数, 最后转入调度程序。所以用户程序中不能包含 main(　)函数, 要把 Init-task(　)函数作为应用程序的入口, 并在其中依次创建各任务。当任务创建后, 置为就绪态, 并按照一定的原则加入就绪队列中, 在系统中, 各任务在运行、就绪和挂起三种状态之间变化。

任务调度按照占先式时间片轮转调度的准则。占先式调度使用是指一旦高优先级任务被设置信号进入就绪态时, 就能得到 CPU 的控制权。一旦任务满足高优先级和设置信号两个条件, 则被置为最好任务, TASK 将调用调度程序进行当前任务的现场保护, 将当前任务置为最好任务, 并启动最好任务。在现场保护中, 将 CPU 的寄存器内容保存到任务内存栈中, 当下次调度时, 弹出保存在该任务内存栈的内容, PC 指针从保存的位置继续往后执行。在最好任务启动中, 系统同样弹出该任务的内存栈, 并使 PC 指针从原保存位置往后执行。在相同优先级的任务之间采用时间片进行轮转调度。

任务调度函数: TASK-sched(　)　// 实现任务切换, 并保护现场

3.4　信号量

使用信号机制的目的在于: ①标志某中断事件的发生; ②指示系统切换到高优先级任务执行; ③使两个任务的行为同步。

当某任务已经处理完, 需要切换到其他高优先级任务进行执行时, 给高优先级任务设置信号, 系统会进行任务切换。当任务之间需要协同进行时, 即同步处理时, 低优先级任务给高优先级任务设置信号, 使系统切换到高优先级任务执行相关事件, 执行完相关事件后, 再返回继续执行被挂起的任务, 完成任务同步。

等待信号函数　task -sigs-type task-wait (p-sigs); //信号等待

清除信号函数　task -sigs-type task -clr-sigs(p-tcb, p-sigs); //信号清除

设置信号函数　task -sigs-type task -set-sigs(p-tcb, p-sigs); //信号设置

取信号函数　task -sigs-type task -get-sigs(p-tcb); //信号获取

3.5　时间片中断

系统占用了 Time 0 作为时间片(TICK)发生器, 通过设置, 使其每 15ms 产生一次中断, 当中断

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

触发后,操作系统停止正执行的任务,转而执行中断服务程序。中断服务程序首先保存现场,然后调用task-tick( )函数,对运行状态和某些等待状态的任务的时间片计数减1。如对定时器链表的扫描操作,在task-tick( )函数中,扫描定时器链表时,如果定时器计数值小于或等于tick步长,则计数值为0(定时器已满),并从定时器链表中删除该定时器,给任务设置信号。设置信号时,首先执行中断锁,在判断所设信号是任务的等待信号后,再判断是否任务的优先级高于当前任务curr-task,如果高的话,则执行调度程序,因为此时所有中断皆锁,不允许其他中断打入(包括tick中断),所以只有等调度程序执行完后,再回到设置信号程序的结尾,结尾是解中断锁,最后返回。返回到tick中断服务程序中的task-tick( )函数,继续执行对定时器列表的扫描,直到对所有的定时器扫描完毕后,函数执行完毕。然后恢复所有CPU寄存器,执行中断返回,返回到被中断的任务中。在设置信号中,如果任务的优先级低于或等于curr-task的优先级,则不执行调度,而是回到设置信号程序的结尾,执行中断解锁。

时间片中断功能函数:task-tick( );//TICK中断服务子功能。

3.6 定时器管理

设置定时器有3个目的:①让事件滞后触发;②要求事件在规定的时间内完成;③挂起当前任务,使之定时等待信号的到来。

TASK模块的定时器的时钟基准是TICK中断,采用倒数计数机制实现。TICK中断的时长由CPU时钟分频确定。TASK模块只提供TICK中断服务程序,在该服务程序中对定时器的计数值进行倒数自减处理。设置定时器时,首先要设定一个计数值cnt,当定时器启动后,内核以一定TICK中断的时长为步长对cnt进行自减。当cnt为0时,定时器为满。此时TASK模块给该定时器所属任务设置信号,使该任务转为就绪态,当该任务的优先级为就绪任务中最高时,则运行该任务。TASK模块设置了一个双向链表对定时器进行管理。内核根据tick中断的到来,对定时器链表进行扫描,对所有计数值进行自减,并判断是否为0.如果为0,则给该定时器所属的任务设置信号,进行相关事件处理。

创建定时器函数int task-def-timer( );//定时器创建

清除定时器函数int task-clr-timer( );//定时器清除

设置定时器函数task-timer-cnt-type task-set-timer(p-timer, p-cnt);//定时器设置

取定时器值函数task-timer-cnt-type task-get-timer(p-timer);//定时器值获取

3.7 内存管理及任务状态机

采用静态内存管理机制,提供稳定的多任务内存堆栈管理功能。任务状态机转换见图3。

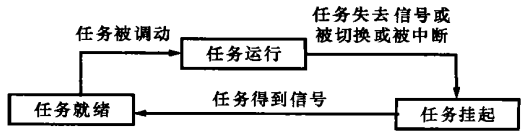

图3 任务状态机转换

定义3个状态:

(1)运行态。任务FOR循环执行状态。只要程序进入任务FOR循环,则认为任务已进入运行态。

(2)挂起态。任务已被创建起来(即任务栈已建立),且已运行过,CPU因某种原因(该任务失去信号、被调度、被中断)而要保存该任务的现场,去执行其他事件处理,该任务即为挂起态。

(3)就绪态。任务已取得信号,但未取得CPU占先权,正等待取得CPU占先权调度执行的状态。

任务状态机是任务启动后状态的转换机制,如图3所示。任务首先要被创建和启动,才进入运行状态。任务因没有信号或者因被切换或者因中断,而进入挂起态。任务在挂起态,因得到信号而转入就绪态,当任务取得CPU占先权后,即被调度,可回到运行态。

一个任务就是一个无限执行的FOR循环,在循环中处理任务的各项事件。任务状态机在多任务运行中体现。当任务创建时,任务TCB指针按任务优先级高低插入双向链表中,优先级高的任务始终居于链表头。链表尾是空闲任务。操作系统起始时,从优先级最高的任务开始,将各任务执行一遍,但不给任何任务赋于信号,从而各任务按优先级高低依次被挂起。操作系统在运行某个任务中,当中断来临时,CPU中断该任务,保存该任务现场。该任务即被挂起,CPU处理中断服务程序。当中断服务程序结束后,CPU恢复被保存的任务现场,继续执行该任务。当低优先级任务给高优先级任务设置信号时,高优先级任务进入就绪态。CPU执行调度程序,使低优先级任务挂起,运行高优先级任务,当高优先级任务信号被清除后,TASK通过任务等待程序挂起高优先级任务,回到低优先级就绪任务,运行

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

该任务。任务状态机是操作系统内核实现任务切换和中断服务的基本原理。

3.8　中断处理

在系统中, 中断不隶属于任何一个任务, 每当中断发生后, 中断控制器将中断的优先级发给 CPU。如果多个中断同时产生, 中断控制器将具有最高优先级的中断发给 CPU。如果 CPU 此时正在处理中断, 则 CPU 比较该最高优先级中断与当前中断的优先级, 决定是否接收中断控制器发来的中断。如果 CPU 没有处理中断, 则接收该最高优先级中断。接收中断后, 操作系统停止正执行的任务, 转而执行中断服务程序。中断服务程序首先作快速预处理, 再恢复现场, 执行中断返回。预处理中的设置信号唤醒某个任务以作进一步的处理, 中断服务和任务切换机制的结合使得操作系统具有处理系统内和应用层事件的能力, 可以使得中断服务程序足够短, 同时又保证了数据的及时处理。

3.9　任务间通讯

为了减少系统对资源的占用, 该系统没有提供任务间的直接通讯手段, 但由于各任务都可以直接访问物理内存, 所以如果需要在任务之间传递数据, 可以借助于全局变量或者指定的内存来实现。

4　应用实例

本模块以约 700 行(包含约 250 行的注释)的紧凑代码实现了一个通用嵌入式实时多任务操作系统内核。主要功能包括: ①多任务创建及调度; ②高实时性的占先式处理功能; ③任务间通信; ④定时器管理; ⑤任务优先级管理; ⑥稳定的多任务静态内存管理等。模块自身包括系统初始化, 内核调度和功能丰富的应用程序接口 API。

模块主要技术参数如表 1 所示。

表 1　ELK 模块技术参数

特　性	参　数	备　注
TICK 中断步长/ ms	15	适用于 TOSHIBA 815 16 位 CPU
CPU 位数/ 位	16	
内核代码量/ kB	＜1	
信号位	16	
优先级范围	0～65 535	

某型移动掌上电脑采用 16 位东芝 CPU, 存储器为 256K RAM, 1M flash。支持 4 个任务。采用静态内存管理, 定制任务控制块存储地址、数据段存储地址、中断向量表存储地址。实现定制任务优先级、信号位 16 位和静态创建任务等。成功支持话音、短消息、Internet 上网浏览、电话本、用户界面等业务, 并已通过测试, 现已批量生产, 大批供应国内外市场。

5　结　语

本文所介绍的仅是嵌入式实时多任务操作系统内核中一部分的实现方法, 所开发的移动终端等产品中, 使用了该内核, 实际运行情况良好, 既在一定程度上满足了实时性要求, 又实现了任务的开发执行, 而且简化了程序设计, 程序模块分明。对于操作系统的可剪裁性、时钟管理等, 由于篇幅的原因未能作详细的介绍。另外, 该内核运行在自主开发硬件的设备驱动模块(BOOT)和退出模块, 因此对不同的 CPU 开发不同的 BOOT 区, 可以运行该操作系统内核软件; 也可以进一步完善用户编程接口, 增加可剪裁性、时钟管理、实现动态内存管理、动态优先级管理、动态任务创建等, 可以在该内核的基础上完成完整的嵌入式操作系统。

参考文献:

[1]　刘腾红. 计算机操作系统[M]. 北京: 科学出版社, 2000.

[2]　庞丽萍. 操作系统原理[M]. 武汉: 华中理工大学出版社, 1994.

[3]　唐　寅. 实时操作系统应用开发指南[M]. 北京: 中国电力出版社, 2002.

[4]　孔祥营, 柏桂枝. 嵌入式实时操作系统 VxWorks 及其开发环境 Tornado[M]. 北京: 中国电力出版社, 2003.

[5]　屈振新. 嵌入式实时操作系统核心的设计与实现[J]. 计算机工程与应用, 2002, (9): 123－128.

[6]　凌　明. 嵌入式操作系统内核的实现[J]. 电子器件, 1999, 4(4): 263－269.

[7]　William S. Operating System (Third Edition) [M]. Prentice-Hall, 1998.

[责任编辑　郭庆健]

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net