嵌入式操作系统RTX51 Tiny的分析及应用

阳 艳,蒋 礼,杨科灵,罗少轩
(中南大学 物理科学与技术学院,湖南 长沙 410083)

摘 要:在嵌入式开发中,嵌入式操作系统的应用是重要的一环。文中对一种适用51单片机的实时多任务操作系统——RTX51 Tiny进行应用。首先从任务管理和内存管理的角度来分析该操作系统,在此基础上,用C语言编写应用程序traffic。应用实践得到以下结论:该操作系统的应用程序启动简单,没有操作系统的启动过程;短小精悍,代码不足50行;运行可靠,仿真、硬件运行结果充分证实了这一点。从而得到RTX51 Tiny操作系统对任务和堆栈管理的有效性及其代码精简和运行可靠的基本特点。

关键词:嵌入式操作系统; 实时操作系统; RTX51 Tiny
中图分类号:TP316.2; TP317 **文献标识码**:A **文章编号**:1673-629X(2006)06-0089-03

Analysis and Application of an Embedded Operating System: RTX51 Tiny

YANG Yan, JIANG Li, YANG Ke-ling, LUO Shao-xuan
(School of Physics Science and Technology, Central South University, Changsha 410083, China)

Abstract: The utilization of the embedded operating system is an important step in the development of the embedded system. A RTOS—RTX51 Tiny which is adapted for 51 single-chip computer is applied. The OS is designed for two purposes: task management and RAM management. On the basis of it, an application program traffic is edited with C. A program is given to examine its validity. The successful running of program adequately testify the two functions of the OS, it gives a powerful management of CPU and RAM. The paper shows that RTX51 Tiny has following characters: it is effective to task management, the codes are simple and it has reliability in operation.

Key words: embedded operating system; real-time operating system; RTX51 Tiny

0 引 言

随着后PC时代的到来,嵌入式产品的功能越来越复杂。为了降低开发难度,很多嵌入式产品已经广泛采用了实时多任务操作系统(RTOS),嵌入式RTOS成为了当今计算机领域的研究热点。在嵌入式产品中,8位和32位处理器是两大支柱产品。适用于32位处理器的操作系统种类繁多,而适用于8位单片机的操作系统则很少。8位处理器虽然面临各种新生代产品的挑战,地位仍岿然不动。目前就成本而言,在较长的一段时期内,很多控制应用领域只需8位单片机就可胜任。对于这样的低端产品开发,要完成较为复杂的任务,也可将操作系统应用在相应的产品开发中。所以在这种情况下,研究适用于51单片机的操作系统显得非常有必要[1]。

一般说来,获得适于8位单片机的操作系统有两种途径:一是精简现有的功能强大的操作系统;二是开发一种专门针对8位单片机的新的操作系统。目前,已有将uc/os移植到51单片机上的报道,但应用时需外扩RAM和ROM。对于51单片机这种资源较少的处理器使用受限。解决的办法是采用新的操作系统,例如RTX51。

收稿日期:2005-09-26

作者简介:阳 艳(1981-),女,湖南衡阳人,硕士研究生,研究方向为嵌入式系统;蒋 礼,教授,从事电子器件的介观尺度效应和热可靠性研究。

1 RTX51 Tiny分析

RTX51是由德国Keil公司开发的,专门针对8051兼容MCU所作的多任务实时操作系统。它有完全版(RTX51 Full)和小型版(RTX51 Tiny)。RTX51 Tiny是一个RTX51 Full的子集,它可以很容易地在没有任何外部存储器的51系统上运转。通常,由于8位单片机的处理速度、内部寄存器资源等因素的限制,由此定制的操作系统功能有限。通过分析发现RTX51 Tiny操作系统主要包括了任务管理和内存管理。RTX51 Tiny在任务管理方面:仅支持时间片轮转任务切换和使用信号进行任务切换,不支持抢先式的任务切换,不包括消息历程;在内存管理方面:没有存储器池分配程序,内核使用keilc51编译器将内存管理简化为堆栈管理[2,3]。

1.1 RTX51 Tiny的任务和事件

RTX51 Tiny任务管理的主要工作是按某种调度策略使应该运行的任务占用CPU,同时,要对各任务在切换时的地址进行保存,即压栈,以便下次运行该任务时,能够恢复运行(下文有详细的分析)。首先看看RTX51 Tiny的任务状态。RTX51 Tiny的用户任务具有以下几个状态。

RUNNING:任务处于运行中。同一时间只能有一个任务处于运行态。

READY:任务正在等待运行。在当前运行的任务时间片完成之后,RTX51 Tiny运行下一个处于ready状态的任务。

WAITING:任务等待一个事件。如果所等待的事件发生的话,任务进入ready状态。

DELETED:任务不处于执行队列。

TIMEOUT:任务由于时间片用完而处于timeout状态,并等待再次运行。该状态与ready状态相似。

图1所示为任务状态转换图。

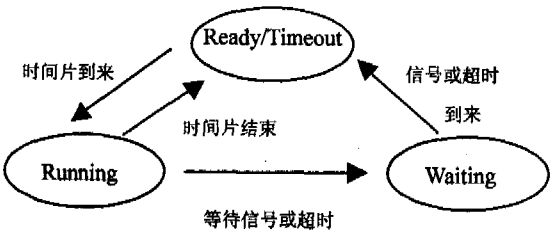

图1 任务状态转换图

一般情况下,任务切换由时间片控制。但有时候,用户需要用事件来控制任务切换。RTX51 Tiny事件有超时(timeout)、间隔(interval)和信号(signal)3种。

Timeout:挂起运行任务指定数量的时钟周期,调用os_wait函数的任务将被挂起,直到延时结束才返回到ready状态,并可被再次执行。

Interval:类似于超时,但是任务的节拍计数器不复位。典型应用是产生时钟。

Signal:用于任务间通信,如果一个任务调用了os_wait等待signal而signal未置位,则该任务被挂起到signal位置,才返回到ready状态,可被再次执行。

系统内核中的os_wait函数(详见下文)挂起一个任务来等待一个事件的发生。当任务等待的事件没有发生的时候,系统挂起这个任务;当事件发生时,系统根据任务切换规则切换任务。

1.2 存储器管理

存储器的管理主要是如何划分存储空间和采用什么原则将要运行的程序安置到内存的适当位置上。RTX51采用的存储管理法是移动法。它以一个任务为单元分配内存,每个任务分配一连续的存储空间。并且堆栈按任务的标号由大到小排列。

RTX51 Tiny是把51单片机的内部RAM作为内存使用的。对于8052芯片,由于00H~2FH是工作寄存器区和位寻址区,设置堆栈要从30H开始,直到FFH都是可用的。任务可用的堆栈从当前SP~FFH。在主程序中,赋给每个任务的堆栈栈底都是RAMTOP。当这个任务被创建时,将会在适当的位置为它腾出两字节的空间,存放任务的入口地址。当这个任务处于运行状态时,被替换下来的任务堆栈空间将被转移到当前任务。同时,时钟0中断程序将会检查这段当前任务所用的堆栈是否有足够的空间,即FREESTACK个空间。当这个任务处于就绪状态时,和任务被创建时一样,只占用两字节的空间,其余的空间被释放。当这个任务被删除时,它所使用的空间将被顺序搬移到正要运行的任务堆栈。

RTX51 Tiny总是将全部闲置的内存分配为运行任务的堆栈区。这样,运行的任务得到了最大的可利用的栈空间。图2说明了应用程序有3个任务时,堆栈的搬移过程。

?STACK表示堆栈的起始地址,堆栈的下面是全局变量、寄存器和位寻址区。栈顶RAMTOP可由配置文件定义。

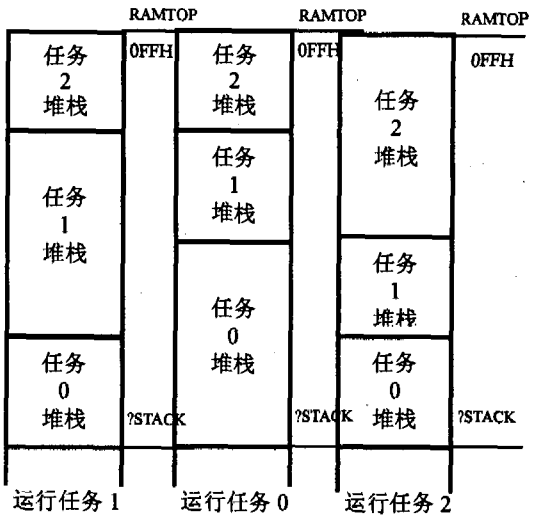

图2 堆栈搬移图

1.3 应用RTX51 Tiny的说明

在对RTX51 Tiny进行应用开发时,将会涉及到一些全局变量及系统调用函数的问题。

表1是对一些全局变量的说明,用户可以通过改变这些变量,达到适应具体系统应用的目的。表2是几个系统调用函数的说明,用户通过使用系统调用函数来达到采用操作系统简化程序设计的目的。

表1 一些全局参量的说明

变量	文字说明
INT_CLOCK	定义系统时钟间隔。系统使用这个间隔产生中断
TIMESHARING	定义时间片轮转任务切换的超时时间
RAMTOP	表明8051派生系列内存储器存储单元的最大尺寸。8051应设为7FH;8052应设为0FFH
FREE_STACK	定义了自由堆栈区的大小。当切换任务时,RTX51 Tiny检验堆栈区指定数量的有效字节

表2 系统调用函数的说明

系统调用函数	文字说明
os_creat(task_id)	启动一个新任务,标记为ready状态
os_delete(task_id)	在任务队列中删除一个任务
os_wait(interval/timeout/signal,ticks)	根据参数的不同,使当前任务停止并等待超时、间隔或信号。ticks是一个interval(间隔事件)或timeout(超时事件)的报时信号数目
os_send_signal(task_id)	从一个任务向任务task_id发送一个信号
os_clear_signal(task_id)	清除task_id任务的信号标志位

2　软件设计及应用

2.1　软件环境

使用 RTX51 Tiny 要求软件环境(即 Keil uVision)包括:C51 编译器、A51 汇编器和 BL51 或 Lx51 链接器。此外,RTX51TNY.LIB 和 RTX51BT.LIB 必须在 \ KEIL \ C51 \ LIB 目录下。RTX51TNY.H 必须在 \ KEIL \ C51 \ INC 目录下[4]。

RTX51 Tiny 程序用标准的 C 语言编写,用 Keil C 51 编译器编译。用 C 语言编写的好处就是能轻松地申明函数而不用理会堆栈的管理和变量结构的定义。

编写 RTX51 Tiny 程序只要求包括 rtx51tny.h 头文件而且使用_task_函数属性声明你的任务。

RTX51 Tiny 程序不需要一个 C 语言主函数(main)。连接过程将包含首先执行任务 0 的程序代码。

另外,用户可以修改配置文件,来改变操作系统的各种参数,如时间片轮转超时值、报时中断的寄存器组、系统计时器的时间间隔等。

2.2　应用设计

为了检验 RTX51 Tiny 操作系统任务管理及内存管理的有效性,设计了一个交通灯实验。车辆和行人交通灯依次点亮。若行人要通过时,可以按下按键,则车辆的通行时间将缩短,以便让行人尽早通过。

2.2.1　硬件组成

单片机 AT89S52(包括 8051 核、256 字节 RAM、8kFLASH、3 个定时/计数器),按键一个,LED 指示灯 5 个,其中用于行人的指示灯 2 个,红绿各一个;用于车辆的指示灯 3 个,红黄绿各一个。

2.2.2　应用软件

在这个交通灯程序中,用 P2.0～P2.4 作为输出,其中 P2.0～P2.2 是控制车辆的,P2.3～P2.4 是控制行人的。P1.0 作为输入,当行人想要通过时,按下一个按键,这个键连接 P1.0,这时车辆的通行时间将缩短,以便让行人尽早通过。5 个指示灯交替显示。循环顺序及时间如表 3 所示。

表 3　交通灯的显示顺序及时间

对应输出 P2 口	交通控制灯	显示时间(s)
P2.0	yellow	1.5
P2.1	green	2/4.5
	yellow	1.5
P2.2	red	1.5
P2.3	walk	5
P2.4	stop	1.5

在这个程序中,根据应用的需要和 RTX51 的特点设计了 3 个任务:初始化任务 0、交通灯控制任务 1 和检测输入按键任务 2。初始化任务 0 将任务 1 和任务 2 激活,放入任务队列,并将任务 0 结束;任务 1 控制交通灯的循环显示周期及显示时间;任务 2 检测按键是否按下,如果按下,则向任务 1 发送信号,用以缩短车辆通行时间。至于任务的切换采用时间片轮转。时间片设为 50ms。

交通灯程序主要部分如下:

```
# include<reg52.h>
# include<rtx51tny.h>
.....
void init (void)_task_ INIT {
  os_create_task (LIGHTS);
  os_create_task (KEYREAD);
  os_delete_task (INIT);
}
.....
void lights (void)_task_LIGHTS {
  .....
  os_clear_signal (LIGHTS);
  os_wait (K_TMO, 200, 0);
  os_wait (K_TMO + K_SIG, 250, 0);
  .....
}
.....
void keyread (void)_task_KEYREAD {
  while (1) {
    if (key) {
    os_send_signal (LIGHTS);
     }
    os_wait (K_TMO, 2, 0);
  }
}
```

程序编写完毕后,在 Keil 环境下,经过包含 RTX51 Tiny 实时操作系统的编译后,仿真通过,并生成 HEX 文件,将程序下载到 AT89S52 芯片的 ROM。程序运行的结果和设想的一样。实验表明,RTX51 Tiny 能够适应多任务的程序设计,有效地进行了任务管理和内存管理(任务的状态和堆栈的使用情况可以通过仿真观察到)。

3　应用特点及结论

应用 RTX51 Tiny 时,发现它具有以下特点:

a. RTX51 采用时间片轮转方式切换任务。这和基于优先级的实时操作系统不一样。采用时间片轮转的特点就是,每个任务占有一定的时间片,而这个时间非常短,几十毫秒,这样就好像几个任务同时运行一样。例如,用单片机控制两小块显示屏时,无论是编程者还是使用者肯定希望它们同时工作,而不希望两块屏的显示有时间差。

b. 启动过程比较简单。RTX51 Tiny 不像有些操作系统一样,需要把内核编译成一个映像文件写入 ROM 中,上电复位后,从 ROM 中把文件加载到 RAM 中去,然后再运行应用程序。RTX51 内核是和应用程序一起编译成一个文件的,使用者只需要把这个文件下载转换成 HEX 格式,写入 ROM 就可以了。上电后,它像普通的单片机程序一样运行。

c. 简化了复杂的软件设计,缩短了项目周期。采用

(下转第 94 页)

等于 360°,则该点位于三角形的内部。

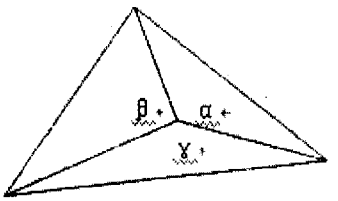

图 2 点在三角形内部的判断

1.3 算法优化

经过包围盒做粗略检测,快速排除了很多不相交的物体,碰撞检测的速度已大有提高。但仍有进一步提高的可能性。当碰撞的发生概率是小概率事件时,可以有效地减少计算次数以及每次计算的复杂度。因此,对以上提出的两步碰撞检测法可进一步引入限时计算的思想[4],采取设定可变时间片长度的方法。

时间步长问题的实质是计算频率问题[5],原因是在有些时间片中物体根本不可能发生碰撞。时间步长方法的目标是避免不必要的时间片中的计算。当碰撞无法发生时,就增大时间步长。当空间中的碰撞发生频率比较低时,这种方法可以节省大量计算时间。

2 实验结果

基于 VC++6.0 和 OpenGL,开发了泰兴东收费站室外三维交互漫游系统,如图 3 所示。整个场景采用 3D 模型,共有 29833 个三角面。在系统中运用文中上述算法进行了碰撞检测,取得了很好的效果,说明该算法是可行和有效的。

3 结束语

关于碰撞检测的研究,目前已经有很多很好的实现算法。对于不同的问题,往往采用不同的检测碰撞的算法。常用算法会带来很大的计算开销,效率不高。文中将两种方法结合,把碰撞检测的过程分成粗略检测和精确检测两步进行,并采用限时计算的思想加以优化,提高了碰撞检测的效率。

图 3 泰兴东收费站室外三维交互漫游系统

参考文献:

[1] 王志强,洪嘉振,杨 辉.碰撞检测问题研究综述[J].软件学报,1999,10(5):545-551.

[2] Akenine-Moller T,Haines E.实时计算机图形学(第 2 版)[M].普建涛译.北京:北京大学出版社,2004.

[3] Van den Bergen G.Efficient collision detection of complex deformable models using AABB trees[J].Journal of Graphics Tools,1997,2(4):1-13.

[4] Hubbard P M.Collision detection for interactive graphics applications[J].IEEE Transon Visualization and Computer Graphics,1995,1(3):218-230.

[5] 石教英.虚拟现实基础及实用算法[M].北京:科学出版社,2002.

(上接第 91 页)

RTX51 Tiny 使得程序的编写和调试变得简单起来。上述交通灯的程序如果不采用操作系统,将要编写大概 200 行的源程序,而采用了 RTX51 Tiny 后,程序还不到 50 行。当程序越复杂时,它作为操作系统精简代码的优点就更加明显。然而,由于资源的限制,它缺乏一般操作系统所必要的支持,没有功能强大的软件包,用户得自己编写驱动程序。

这些应用特点及程序的运行充分说明了 RTX51 Tiny 操作系统代码精简和运行可靠的基本特点[5]。

当然,一个嵌入式工作者要面对的不仅仅是决定采用什么操作系统和开发工具的问题,还必须学会适应一个可能完全外语化的编程环境。并且还不得不处理如系统资源限制、硬件设备驱动、存储分配等一系列问题。

参考文献:

[1] Stepner D, Rajan N, Hui D. Embedded Application Design Using a Real-Time OS[A]. Proceedings of the 1999 36th Annual Design Automation Conference (DAC)[C]. New Orleans, LA, USA:IEEE,1999.151-156.

[2] 李仕涌, 谭南林. 多任务操作系统在嵌入式系统开发中的应用[J]. 北方交通大学学报, 2002,26(4):79-82.

[3] 赵学军. 单片机实时嵌入式操作系统微内核的设计[J]. 桂林电子工业学院学报, 2002,22(3):76-79.

[4] 刘天泉, 黄 海, 王树青. RTX51 在运动控制系统中的应用[J]. 机电工程, 2004,21(1):30-32.

[5] 李小文,万家富,梁慧冰. μC/OS-II 在 MCS-51 系列中的应用[J]. 微计算机应用, 2003,24(5):274-277.