云南大学学报（自然科学版），2010，32（2）：152～157　　CN 53－1045/N　ISSN 0258－7971
Journal of Yunnan University

基于 RTX51 Tiny 实时多操作系统的软件设计与应用[*1]

何　勇
（贵州大学 计算机科学与信息学院，贵州 贵阳　550025）

摘要：传统的单片机编程模式难以适应目前越来越复杂和多变的应用需求，而基于小内核操作系统支持的单片机编程可以很好地解决应用时碰到的很多困难. 研究了实时多任务操作系统 RTX51 Tiny 的内核原理和在此基础上的单片机编程方法，并给出了应用实例和在开发中应注意的一些问题.

关键词：单片机；实时操作系统；RTX51 Tiny
中图分类号：TP 273　　**文献标识码**：A　　**文章编号**：0258－7971(2010)02－0152－06

典型的单片机软件结构是一个前、后台处理程序，即将实时性要求不高的任务放在主函数中，主函数是一个无限循环结构，任务在循环中依次轮流执行，实时性高的则使用中断技术进行处理，任务模块间通过软件标志、全局变量进行通信[1]. 这种传统的以应用为核心、软件直接建立在硬件上的编程方式当处理任务较多、任务之间关系复杂、系统对实时性要求较高时程序的复杂度会急剧增大，因为其中模块的调度以及相互通信完全依靠开发人员的经验和编程技巧，而且程序的可读性和可维护性也很难保证. 随着应用系统的规模和复杂度不断扩大，这种单片机编程模式难以适应目前实际应用的要求.

随着芯片技术的快速发展，单片机器件中已经普遍集成了较大容量的 Flash Rom，代码的规模大小已经不再是主要矛盾. 在单片机软件开发中采用基于实时操作系统的编程模式，虽然代码的规模变大了，但可以提高系统的可靠性和开发效率，降低开发成本，使开发人员能集中精力于应用本身，有利于系统的扩展和移植[2].

1　RTX51 Tiny 工作过程

1.1　内核原理　RTX51 Tiny 是德国 Keil 公司推出的专为 MCS51 系列单片机定制的小型嵌入式实时操作系统，并内嵌在集成开发环境 μVision 中[3]，其特点是既能保证对外界的信息以足够快的速度进行处理，又能并行运行多个任务，具有实时性和并行性的特点，可以无缝嵌入到单片机应用程序中. 该系统是完全免费且源码公开的操作系统，源码全部由汇编语言写成，运行速度快，占有硬件资源低，可进行方便灵活地配置，并且具有很小的内核，最大占用约 900 字节的程序存储空间，可以在没有任何外部存储器的单片机上运行[4].

RTX51 Tiny 的内核是按时间片轮转的方式对系统中运行的多个任务进行调度管理，它利用单片机定时器 T0 中断产生的周期报时信号，作为定时驱动操作系统的时钟，最多可支持管理 16 个任务，所有的任务可以同时被激活，支持非抢占式的任务切换. 系统为每一个任务分配一个独立的堆栈区，在任务切换的同时改变堆栈的指针，并自动保存和恢复寄存器的值. 由于单片机片上资源非常有限，而 RTX51 Tiny 运行时只占用了定时器 T0 和少量存储器空间，可以说是专为单片机量身定做的，最适用于要求多任务管理而

* **收稿日期**：2009－06－06
基金项目：贵州省自然科学基金资助项目(3027).
作者简介：何　勇(1974－)，男，硕士，讲师，主要从事嵌入式系统、网络通信方面的研究.

对实时性要求不是特别高的场合.

内核完全集成在 Keil C51 编译器中,以系统调用的方式提供运行,内核提供以下函数供应用程序使用,如创建任务、从任务队列中删除任务、发送信号给一个任务、停止当前任务并等待一个或几个事件,完全可以满足一个典型的单片机开发应用. 其内核代码用汇编语言写成,完成系统调用的所有函数,在 2.0 版本中,主要由若干个模块文件组成,其文件名对应相应的系统调用函数名.

1.2　任务管理和同步机制　一个任务可看成是独立的执行进程,并与其他的并发任务竞争 CPU 时间. 任务的创建通过 os_create_task 函数完成,在 RTX51 Tiny 管理下,执行队列中的任务按轮转方式在分配的时间片内执行. 调度规则是:当前正运行的任务主动让出 CPU 资源;或当前任务的时间片已经用完而仍未完成的情况下,由操作系统进行任务切换.

在系统中一个用户任务可能有以下 5 种状态,任务之间的转换关系如图 1 所示.

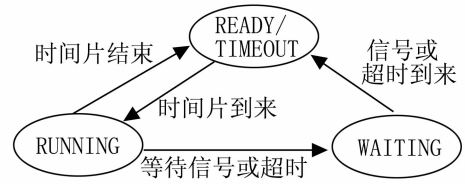

图 1　任务状态转换图

Fig. 1 The state transition diagram of task

(1)运行(RUNNING)　任务正处于运行中,同一时刻只有一个任务可以处于运行状态.

(2)就绪(READY)　等待运行的任务处于“READY”状态. 表示该任务处于就绪状态,可随时执行.

(3)等待(WAITING)　等待一个事件的任务处于“WAITING”状态. 如果等待的事件发生,则此任务进入“READY”状态.

(4)超时(TIMEOUT)　表示该任务处于就绪状态,可随时执行,该任务是在时间片内未完成而终止的.

(5)删除(DELETED)　表示这个任务已经从队列中删除,不会参与调度.

同步机制是为了能保证多任务在执行次序上的协调,内核用特定事件进行任务间的通信和同步. RTX51 Tiny 中提供了以下几类事件及事件的或组合对同步机制进行支持.

(1)超时　挂起运行的任务,并等待指定数量的时钟周期,超时信号不累计.

(2)间隔　类似于超时,但是软件定时器没有复位,超时信号要累计.

(3)信号　用于任务之间的同步协调.

例如,某个应用中任务 A 必须在另一个任务 B 完成之后才能执行,可以使用内核提供的信号事件来实现. 使用 os_wait 函数,通过不同的事件参数,可以实现挂起当前任务,等待 1 个或几个间隔(K_IVL)、超时(K_TMO)或信号(K_SIG)事件,如果所等待的事件已经发生,继续执行当前任务;如果所等待的事件没有发生,则置相应的等待标志后,挂起该任务,系统会切换到下一任务. 发送信号可通过 os_send_signal 函数或 isr_send_signal 函数完成.

使用 RTX51 Tiny 编程时,应用程序不需要主函数,因为主函数实际已经内置到操作系统中,但至少应该有一个任务 0. 单片机上电后 RTX51 Tiny 操作系统首先取得控制权,然后自动执行任务 0,在任务 0 中进行系统初始化操作并创建其他任务. 所有任务构成一个任务队列,很多情况下任务是一个无限循环的过程. 通过 RTX51 Tiny 在后台的调度,每个任务占用一个时间片,如果一个时间片到而该任务还未完成,操作系统会强行退出该任务,切换至队列中的下一个任务. 当队列中所有任务都被执行一遍之后,再调度执行第 1 个任务,如此不断循环. 如果对性能有更高要求,可直接调用 os_switch_task 函数进行任务切换,则不需等待内核自动切换.

1.3　存储管理　内核对全局变量和局部变量采取静态分配存储空间的策略,因而存储器管理简化为堆栈管理. 内核为每个任务都保留一个单独的堆栈区,全部堆栈管理都在 IDATA 空间进行. 为了给当前正在运行的任务分配尽可能大的栈区,所以各个任务所用的堆栈位置是动态的,并用 STKP[task_id]来记录和任

务堆栈栈底位置. 当堆栈自由空间小于 FREESTACK(默认为 20)个字节时,就会调用宏 STACK_ERROR,进行堆栈出错处理.

当任务状态发生改变时,系统将进行相应的堆栈管理:

(1)任务切换　将全部自由堆栈空间分配给正在运行的任务;

(2)任务创建　将自由堆栈空间的 2 个字节,分配给新创新的任务 task_id,并将 ENTRY[task_id]放入其堆栈;

(3)任务删除　回收被删除的任务 task_id 的堆栈空间,并转换为自由堆栈空间.

RTX51 Tiny 操作系统非常小巧,但也占用了一定的存储空间. 其中有关存储空间方面的信息是:RAM 需求为 7 字节 DATA,外加每个任务占用 3 字节 IDATA 空间,代码量即 ROM 约 900 字节[4].

1.4　系统配置　RTX51 Tiny 的配置参数在 conf_tny. a51 文件中进行定义,用来确定堆栈出错时要执行的宏和系统运行需要的全局变量如内存大小、堆栈空间和时间片等,可以根据实际情况灵活修改这些参数.

可配置的参数主要与可用的硬件资源和系统运行的方式相关. 如任务轮转时的时间片大小,是由 2 个参数 INT_CLOCK 和 TIMESHARING 决定. INT_CLOCK 是时钟中断使用的周期数,默认值是 10 000,即 10 000个机器周期,这个数值实际上是操作系统的最小时间单位;TIMESHARING 表示中断次数,默认值是 5,2 个参数的乘积决定了 1 个任务 1 次使用的最大时间片,也即任务轮转的基本时间单位. 典型情况下若使用 12MHz 晶振,默认配置下的 1 个时间片是 50 ms.

RAMTOP 参数定义堆栈可使用的最高 RAM 地址,默认值为地址 0FFH(表示 52 系列单片机). FREE_STACK 参数配置堆栈大小,默认值为 20.

2　基于 RTX51 Tiny 的软件设计方法

2.1　软件设计原则　总体设计阶段可将待完成的工作分解为若干任务,任务实际是系统的各个基本的功能模块,而各个任务的管理、相互的通信由操作系统来完成. 这种面向任务的编程方式符合人处理问题的思维,能更方便地对实时处理情景进行描述,程序的可读性、可维护性都得到提高.

应用程序中应包含头文件 rtx51tny. h,且不能禁用中断使能(EA = 0),因为操作系统使用了定时器 T0 中断. 程序中不需要写主函数,任务 task 0 就是应用程序的入口,在 task 0 中可调用 os_create_task 函数创建其他任务.

中断服务程序与系统是并行工作的,但应注意缺省状态下,所有的中断是同一优先级别,如果在执行其他中断服务程序的过程中又发生了时间片任务调度中断,那么时间片调度将由于优先级不高的原因而不执行,导致任务无法切换. 这种情况可以采用信号机制进行处理,即可在中服务里使用 isr_send_signal 函数,把信号发给另外一个任务,让另外一个任务来接管中断服务,这样内核会自动做好相应的设置以保证时间片的调度工作正常进行.

一般将配置文件 conf_tny. a51 的一份拷贝添加到项目文件中[5],并根据实际情况调整配置参数. 如堆栈的大小要设置得合适,太大浪费资源,太小又会出现堆栈错误. 而轮转时间片的长度也不宜设置得过大或过小,设置得过大,则一些持续时间较短的事件无法及时响应. 如果轮转时间设置得过小,则 CPU 的时间会过多消耗在任务切换上了,需要根据具体的情况权衡.

2.2　任务划分原则　结合 RTX51 Tiny 的准并发特性,对应用软件要完成的功能进行大小适当地划分,即按照一定原则划分若干个任务模块,并对各个任务间的通信和时延进行确认. 采用软件工程中的信息隐藏原则,如尽量使用数据耦合,少用控制耦合,限制公共环境耦合的范围,完全不用内容耦合[6].

对同一个外设的访问放在一个任务中,这样可以有效避免系统中的资源冲突和重入问题,降低设计的复杂度,这样无论何时切换任务,都不会对独立的设备资源造成影响.

通过任务分割提高系统的实时性. 执行时间较长的任务对使用不可剥夺型实时内核的响应时间影响

明显,由于 CPU 长时间停留在长任务中,其他任务得不到实时的响应,系统的实时性会明显下降. 解决长任务问题的有效途径是进行任务分割,将影响系统实时性的长任务分割成若干小任务.

2.3　应用系统功能及程序结构　系统硬件部分主要包括单片机、温度采集电路、键盘接口、显示电路、报警电路、存储器电路、驱动电路及串口通信等[7-8].

系统工作过程如下:单片机首先把通过传感器测到的现场温度与预先设置的温度进行比较,当温度超过设定值时,系统自动产生报警指示,并输出信号通过继电器驱动控制加热器的工作,从而实现温度控制,同时通过 RS-232 串行口与上位机通信,使系统具有远程温度控制能力和远程报警能力.

设计时将系统的功能分解为 6 个任务模块,分别完成各个基本功能,软件入口通过任务 0 完成,整个软件的可读性非常好. 任务的管理、调度以及任务之间的联系完全在 RTX51 Tiny 的控制下进行工作,具体是通过各个系统调用函数完成.

以下是主程序部分内容,具体说明参见注释.

```
#include <rtx51tny.h>          //操作系统头文件
#include "DS18B20.h"           //温度采集驱动模块
#include "1602LCD.h"           //液晶显示器驱动模块
#include "24C01.h"             //存储器驱动模块
//任务标识号 ID 的定义
//初始化和创建其他任务的工作定义为任务 0,被操作系统首先自动执行
//显示程序定义为任务 1,报警程序定义为任务 2,调整温度定义为任务 3
//键盘操作定义为任务 4,温度读取定义为任务 5,通信处理定义为任务 6
#define Init          0
#define Disp          1
#define Beep          2
#define Adjust        3
#define Key           4
#define Temperature   5
#define Comm          6
void job0 (void) _task_ Init{
//任务 0,初始化各硬件寄存器
//设定串口工作方式
  os_create_task (Disp);      //分别创建各个任务模块
  os_create_task Beep);
  os_create_task (Adjust);
  os_create_task (Key);
  os_create_task (Temperature);
  os_create_task (Comm);
  os_delete_task (Init);      //初始化任务完成后即可删除,以免占用系统资源
}
//串口中断服务程序
//接收 PC 发来的控制命令,该中断和 RTX51 Tiny 并行工作
void Receive() interrupt 4 using 2{
  isr_send_signal(Comm); //中断处理后送出信号,该函数是中断和 RTX51 Tiny 的接口
```

```
}
//任务1:显示当前读取的温度和设定的温度范围
void DisplayLCD(void) _task_ Disp{
while(1)
{  }}
//任务2:等待信号,当该事件满足,任务状态从 waiting 变成 ready,等待调度
void Beep_Flash(void) _task_Beep{
  os_wait(K_SIG,1,0);
}
//任务3:驱动加热器工作
void AdjustTemp(void) _task_ Adjust{
  os_wait(K_SIG,1,0);
}
//任务4:设定报警温度,并保存在24C01中
void KeyProcess(void) _task_ Key{
  while(1) { //os_wait(K_IVL,2,0); //键盘延时防抖 }
}
//任务5:采集温度,如果超出设定的温度,发信号报警并调整温度
void ReadTemperature(void) _task_ Temperature{
  os_send_signal(Beep);
  os_send_signal(Comm);
  os_send_signal(Adjust);
}
//任务6:发送温度到上位机PC中报警
void CommPC(void) _task_ Comm{
os_wait(K_SIG,1,0);
}
```

2.4 应用总结 应用时还应注意以下几个问题:

(1)由于 RTX51 Tiny 会占用少量系统资源,在设计程序时要特别注意,避免冲突. 程序中不能使用定时器 T0,其他中断资源可以正常使用,并和操作系统并行工作,但一般不要修改中断的优先级,以免影响操作系统的正常调度.

使用中断时还应注意寄存器组的占用问题,由于 RTX51 Tiny 默认使用片内第1组寄存器,所以在使用中断时,应尽量避开,例如使用中断0时,可以通过编译器指令改变默认使用的寄存器组,如可采用这样的方式:函数名( ) interrupt 0 using 2,通过 using 2 使用第2组寄存器,这样可以避开使用第1组寄存器.

(2)在 RTX51Tiny 中实现共享资源独占也是常碰到的问题,有几个解决的方法. 其一可通过 TIME-SHARING 这个系统变量来禁止时间片轮转,使其值为0来禁止时间片轮转,就可以实现禁止任务切换,当前任务就可以独占共享资源;其二可以关闭中断来实现,使 EA = 0,定时器 T0 的中断被关闭,不能再为时间片轮转提供基准,从而禁止了任务切换. 但这2种方法都带有一定的局限性,前一种方法只能适用于实时性要求不高的场合,后一种方法由于 T0 中断关闭时间不能太长,只能适用于一些简单变量操作的场合. 更完善的解决办法可以借鉴其他实时操作系统的信号量机制,该机制可以很好地支持对共享资源的管理和使用,但由于 RTX51 Tiny 本身不支持信号量机制,只能在程序中自己定义信号量及其操作函数来扩展

系统对共享资源的管理,这方面的内容本文不再赘述.

3　结　论

选择免费、源码开放,并且可以灵活定制的实时操作系统是开发单片机应用时的理想选择. RTX51 Tiny 完全符合上述条件,是专门为单片机量身定制的,能够充分保证系统的稳定性和应用程序的性能,所以非常适合作为单片机应用软件的支撑环境.

本文分析了 RTX51 Tiny 实时多任务操作系统的内核原理,重点是与应用最密切相关的多任务调度机制,研究了相关软件设计方法并给出应用实例,最后进行了应用总结. 实践证明在 RTX51 Tiny 的支持下设计单片机软件可以增强系统的稳定性,并明显缩短了开发周期.

参考文献:

[1]　Jean J Labrosse. μC/OS – II—源码公开的实时嵌入式操作系统[M]. 邵贝贝,译. 北京:中国电力出版社,2001.
[2]　朱珍民,隋雪青,段斌. 嵌入式实时操作系统及其应用开发[M]. 北京:北京邮电大学出版社,2006.
[3]　冯桂平,鲁怀伟. 关于 RTX51 Tiny 的分析与探讨[J]. 单片机与嵌入式系统应用,2008(5):68-70.
[4]　Keil 公司. RTX51 Timy user's guide[M]. Keil,2004.
[5]　马忠梅,刘宾,戚军,等. 单片机 C 语言 Windows 环境编程宝典[M]. 北京:北京航空航天大学出版社,2003.
[6]　张海藩. 软件工程导论[M]. 北京:清华大学出版社,2008.
[7]　敬岚,朱海君,张硕成,等. 基于 AT89C51 的自动测量和控制系统设计[J]. 仪器仪表与传感器,2004,(12):35-37.
[8]　张菁. 单片机温度控制方案的研究[J]. 上海交通大学学报,2007,41(1):142-144.

Design and application of software based on RTX51 Tiny real – time operating system

HE Yong
(College of Computer Science & Information, Guizhou University, Guiyang 550025, China)

Abstract: Currently the programming method of single – chip microcontroller really faced big challenge owing to more and more complex requirement of application, therefore it is a better choice using this programming method under supporting of real – time operating system than conventional programming method. This paper researched corresponding programming method of the embedded application and working process of RTX51 Tiny, which is a real – time multitask operating system. In addition gives application instance and problems which should be paid attention to during development.

Key words: single – chip microcontroller; real time operating system; RTX51 Tiny