

YZ-RTOS:一个面向继电保护装置的 嵌入式实时操作系统

郭为群,朱利,王亚辉,刘福财
(西安交通大学 软件学院,陕西 西安 710049)

摘要:针对继电保护装置对实时性、可靠性要求很高的特殊应用需求,提出一种基于逻辑栈和三值信号量的嵌入式实时操作系统的模型—YZ-RTOS。在内核算法实现上,按照任务实时性要求的不同,将任务分成8个等级。给出了基于有限状态机的调度和同步描述。测试结果表明,YZ-RTOS占有很小的程序空间和数据空间,在实时性、可靠性方面具有很高的性能,满足了继电保护的应用需求。

关键词:继电保护装置;实时操作系统;逻辑栈;信号量同步

中图分类号:TP391

文献标识码:A

文章编号:1005-3751(2006)04-0113-03

YZ-RTOS: A Relay-Oriented Protection Set Embedded Real-Time Operation System

GUO Wei-qun, ZHU Li, WANG Ya-hui, LIU Fu-cai
(Software Institute, Xi'an Jiaotong University, Xi'an 710049, China)

Abstract: Relay protection set emphasizes on high real-time and reliability, in order to meet these special application requirements, in this paper, an embedded real-time operation system model based on logical stack and three-value semaphore: YZ-RTOS is proposed. In the core algorithm implementation, tasks are divided into eight levels according to different timely requirements. Finite state machine based specification for schedule and synchronization is presented. Test results indicates that YZ-RTOS consumes very little program space and data space, and a high performance on real-time and reliability is achieved. These may be sufficient to the requirement of relay protection application.

Key words: relay protection set; real-time operation system; logical stack; semaphore synchronization

0 引言

随着电力系统应用的复杂化,继电保护装置需要同时控制或监视多个外设,可能会产生多个实时任务动作死线(dead line)^[1],因而如何满足多个任务并发执行的实时性和可靠性是一个实时操作系统(Real Time Operation System, RTOS)较难解决的问题。目前,可用的嵌入式操作系统已有很多,如 VxWorks^[2], pSOS^[3], uC/OS^[4], Windows CE 等等。这些操作系统大部分是为专有系统而开发的,或者针对某种特定的处理器,或者对系统资源有严格要求,价格昂贵。源代码开放的 RTOS,如 uC/OS-II^[4]和 Linux,实时性好(它的中断间隔一般为 10~200 毫秒)、可靠性高、功能完善、维护性好,但 RAM/ROM 的资源需求较高。如 uC/OS-II 对栈资源需求较高,尤其是 uC/OS-II 的内核不能解决优先级反转的问题而影响了应用。针

对以上问题,笔者自行开发了一个面向继电保护装置的嵌入式实时操作系统 YZ-RTOS,它实时性好(它的中断间隔 ≤ 5 毫秒)、可靠性高、面向应用裁减、可维护性好。

1 基于逻辑栈的 YZ-RTOS 设计

1.1 继电保护装置的软件组成

继电保护装置系统的软件分为 4 个部分(如图 1 所示),包括初始化、核心调度部分、应用接口部分和多任务部分。装置上电后,程序首先进行初始化部分的执行,在这个过程中,对各种寄存器、CPU、中断、EEROM、键盘等初始化,完成软硬件模块的配置工作。定时中断启动后,首先进行数据采集,然后启动 YZ-RTOS。YZ-RTOS 修改任务计数值和就序表进入任务切换,根据任务的优先级调度任务。由此整个系统的资源被 YZ-RTOS 管理,从而完成各个任务。

1.2 基于逻辑栈的 YZ-RTOS 设计

实时操作系统 RTOS 分为 4 个层次(如图 2 所示),即硬件层、硬件抽象层、系统内核层和应用接口层。其中,内

收稿日期:2005-08-11

作者简介:郭为群(1970-),女,河南开封人,硕士研究生,从事嵌入式系统的开发和应用工作;朱利,副教授,博士,研究方向为新型网络、多媒体通信和嵌入式系统。

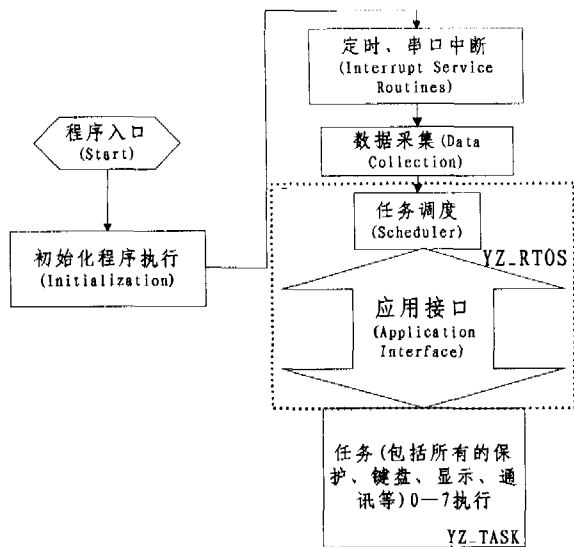


图 1 继电保护装置软件结构图

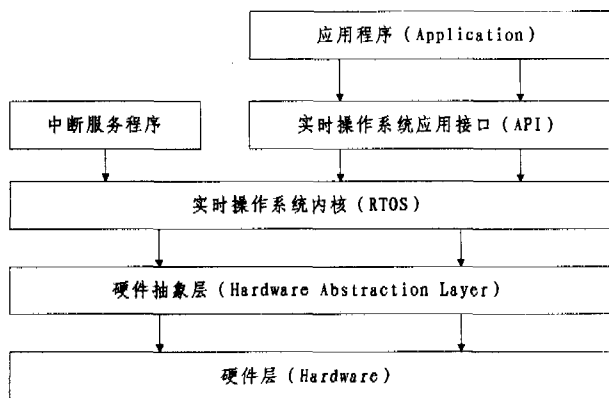


图 2 实时操作系统 YZ-RTOS 结构模型

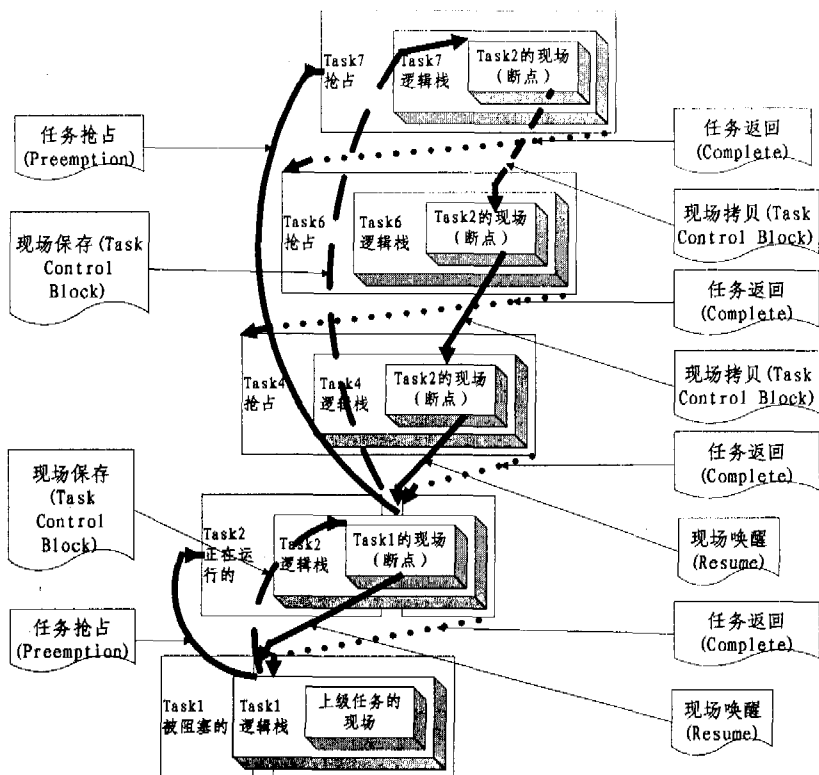


图 3 内核调度有限状态图

核是 RTOS 的关键,它的功能是实现任务调度。应用接口层可剪裁地提供如下内核对象服务:包括任务管理、时间管理、信号量管理、事件控制、邮箱管理、消息队列管理、内存管理等^[4]。针对继电保护装置的特性,即要求高的实时性,只须实现信号量这一个内核服务,用于解决资源冲突的问题。

YZ-RTOS 的任务调度是基于逻辑栈的,按优先级策略进行抢占式调度。根据应用需求,任务的级别设计为 8 个优先级:任务 7 优先级最高,任务 0 最低。调度算法用有限状态机表示,如图 3 所示。

为了简化任务描述,定义 T_h, T_l 分别为高、低优先级的任务, S_h, S_l 分别表示 T_h, T_l 的逻辑栈,其中, $l < h$, 且 $l, h \in [0, 7]$; C_i 表示断点处的上下文关系, $i \in [0, 6]$; $>$ 表示压栈操作。那么,当 T_h, T_l 并发执行时, T_h 抢占 T_l , 伴随动作: $C_l > S_h$ 。如果 T_h 完成,且 $\exists T_l, l < i$, 那么 $C_l > S_l$, 执行 T_l ; 否则 $C_l < S_h$ 。

1.3 三值信号量同步

在继电保护装置中,任务的粒度是异构的,对实时性的要求不一样。其中,基于中断的采样任务每 1667 微秒执行一次,获取得原始的数据和开入量;保护运算、保护判断、动作输出、分合闸回路断线报警任务、手动合分闸处理等任务,每 10 毫秒执行一次;测量数据,如电压、电流、功率,每秒执行一次;键盘处理,每 100 毫秒执行一次;通讯任务,它的触发取决于上位机的命令。

在如上的任务划分中,可以看到测量数据是实时性要求不太高的任务,因此在执行这个任务的过程中,会被许多高优先级的任务打断。如果测量数据取的是采样任务中的原始数据,那么参与测量的数据很容易被破坏,导致测量数据的误差较大。而如果把测量数据任务提高任务等级,一方面没有必要,另一方面又加重 CPU 的负担。针对这个问题,设计了一个三值信号量,来对采集数据进行保护。

信号量是一种约定机制,主要用于控制共享资源的使用权、标志某事件的发生、使两个任务的行为同步。笔者制定的是三值信号量:

$$S = \begin{cases} 0, & \text{initialization, copy1} \\ 1, & \text{copy2} \\ 2, & \text{computation} \end{cases}$$

信号量的初始化值是 0,当采集任务发现这个信号量为 0 时,就把采集到的原始数据(一个周波的)拷贝到一个设定好的缓冲空间里,信号量置为 1;顺序再拷贝 3 个点的电流值以用于无功功率运算,信号量置为 2;当测量任务就绪并检测到信号量为 2 时,激活计算任务。计

算完成后,将信号量重新清零,再去取数据,如此往复。这样既没有浪费 CPU 的时间,又保证了测量的精度。信号量的状态转移图如图 4 所示。

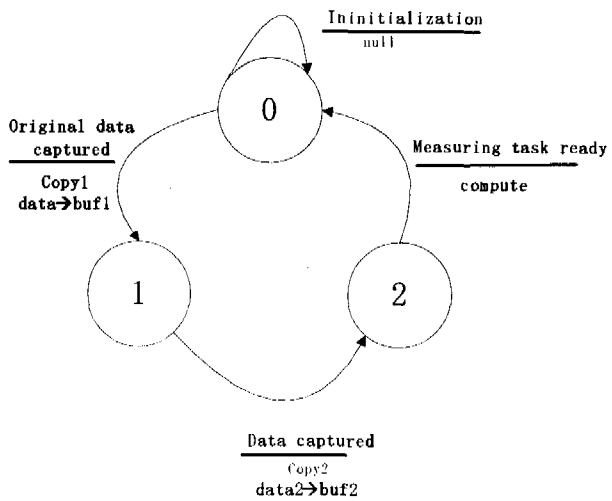


图 4 三值信号量的有限状态机

2 内核算法描述

基于优先级的内核采用的是基于静态优先级的可抢占式的固定周期轮转调度。也就是说,正在运行的任务总是就绪任务中级别最高的那个任务。这样才能使得任务的响应时间最优化。遵循资源占用最小的原则,YZ-RTOS 的任务具有 4 种调度状态:运行态、就绪态、阻塞态和等待态。图 5 显示了任务调度状态及其转换。

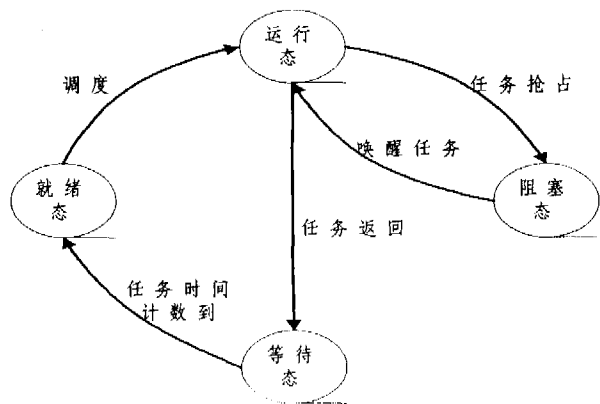


图 5 任务调度状态及其转换

采用基于静态优先级调度^[5]的策略,避免了基于动态优先级出现的优先级反转和占用资源过多的问题。调度使用了系统栈和任务现场保护逻辑栈,它对资源的需求是静态的、固定的,而且资源的需求很小,(YZ-RTOS 的一个任务现场保护逻辑栈为 26 个字节)。而基于动态优先级调度对资源的需求是动态的、不定的,而且其任务栈空间很大,容易出现问題。例如 uC/OS-II,在动态分配中,由于反复地建立和删除任务,内存堆中可能会出现大量的内存碎片,导致没有足够大的连续内存区域可用作任务堆栈。

3 性能指标实验验证

在实时操作系统内核中,时间是最为重要的指标。定义: D_i 为中断延迟时间, R_i 为中断响应时间、 T_R 为中断恢复时间。则根据文献[4],有:

$$D_i = \max(T_i) + T_{s(i)}, i \in [1, 2, 3, \dots]$$
 (1)

其中, T_i 表示关中断时间, $T_{s(i)}$ 表示开始执行中断服务子程序第 1 条指令的时间。

$$R_i = D_i + \sum S(R_i) + T_c, i \in [1, 2, 3, \dots]$$
 (2)

其中, $S(R_i)$ 表示寄存器入栈时间, T_c 表示实时操作系统内核算法在中断服务程序内的执行时间。

$$T_R = T_h + R_h + I_R$$
 (3)

其中, T_h 表示判断是否有优先级更高的任务进入了就绪态的时间, R_h 表示恢复优先级更高的任务的 CPU 内部寄存器的时间, I_R 为执行中断返回指令的时间。

装置的 CPU 选用了 80196KC,主频为 16M,采样周期为 1667 微秒,即一个周波采集 12 个点,每采 3 个点后调度一次,内核的时钟节拍为 5 毫秒。使用某种实时内核,一般地说,关中断的时间最长不超过内核本身关中断的时间,这样就不会影响系统的中断延迟。根据公式(1)、(2)和(3),YZ-RTOS 实际执行时的性能参数测量结果如表 1 所示,可充分满足应用需求。

表 1 性能参数测量结果

	最大	最小	平均
中断延迟	38 微秒	35 微秒	36.5 微秒
内核响应时间	22 微秒	12 微秒	17 微秒
中断响应时间	55 微秒	52 微秒	53.5 微秒
任务切换时间	37 微秒	20 微秒	28.5 微秒
唤醒阻塞任务时间	17 微秒	11 微秒	14 微秒
中断恢复时间	41.8 微秒	16.25 微秒	29 微秒
内核代码空间	650 字节	650 字节	650 字节
数据使用空间	237 字节	237 字节	237 字节

4 总 结

针对继电保护装置设计并实现了一个高实时性的操作系统:YZ-RTOS。实际运行结果表明系统的稳定性好、可靠性高。YZ-RTOS 在用于继电保护装置后,通过了国家继电器质量监督检验中心的检测。

参考文献:

[1] Li Qing, Yao C. Real-Time Concepts for Embedded Systems [M]. 王安生译.北京:北京航空航天大学出版社,2004. 13-15.
[2] 美国风河公司. 产品说明[EB/OL]. <http://www.windriver.com/products/device-technologies/os/vxworks6/>, 2005.
[3] 美国风河公司. 产品说明[EB/OL]. <http://www.windriver.com/products/device-technologies/os/psosystem-3/>, 2005.
[4] 邵贝贝. uC/OS-II-源码公开的实时嵌入式操作系统 [M]. 北京:中国电力出版社,2001. 10-70.
[5] 徐甲同. 操作系统教程[M]. 西安:西安电子科技大学出版社,1992. 92-93.