# 单片机中基于多线程机制的实时多任务研究

武汉化工学院计算机科学与工程系(430073)　冉　全

摘　要：借鉴操作系统中的多任务和多线程机制，提出一种单片机等嵌入式操作系统对多任务进行实时处理的方法。

关键词：单片机　线程　并行处理

一些嵌入式设备需要操作系统，如掌上电脑、PDA、网络控制器等高性能的手持设备和移动设备。它们往往和无线通信、互联网访问和多媒体处理等复杂而强大的功能联系在一起，并对 CPU 要求很高，通常是以通用 CPU 为原型的各种高端嵌入式处理器。

作为一个完整的操作系统，RTOS(嵌入式实时多任务操作系统)有一个可靠性很高的实时内核，将 CPU 时间、中断、I/O、定时器等资源全部包装，留给用户一个标准的应用程序接口(API)。根据各个任务的优先级，合理地在不同任务之间分配 CPU 的时间，保证程序执行的实时性、可靠性。内核一般都能提供任务调度与管理、时间管理、任务间同步与通信、内存管理和中断服务等功能，部分高档商业化产品如 Windows XP Embedded 甚至支持 32 位地址空间、虚拟存储管理、多进程以及嵌入式操作系统中不多见的动态链接库(DLL)。对于这些 RTOS 来说，多任务实时处理不是困难的事情。

但更多的情况下用户使用的是另一类 CPU——微控制器。例如：大多数单片机往往是按照某一流程执行单一任务。出于成本和技术上的原因，这类软件开发多数还是基于处理器直接编写，没有选配实时多任务操作系统作为开发平台，也不需要将系统软件和应用软件分开处理。但是在实际应用中，有时也会面临同时处理多个并行任务的情况。这就需要一种运行机制来模拟 RTOS 中的处理方式。

## 1　RTOS 中的设计思想

单处理机多道程序系统具有如下特征：

(1)从宏观上看，几道程序“同时运行”，即它们先后开始各自的运行，且均未结束。

(2)从微观上看，几道程序“交替执行”。对于单处理机系统而言，它们只能轮流地占用 CPU。

其实质是几道程序在处理机中“交替执行”。按照现在常用的方法，把一道程序和一个任务对应，将任务中的每个分开的、独立执行的部分称为线程。

RTOS 系统应具备的特征为：

(1)由于实时操作中的多任务引起的并发性和实时性，要求操作系统对资源分配具有更强的控制能力。

通常的办法是设立前台与后台二个作业的分配。前台作业中包含实时采集、控制、处理有关的任务，任务优先级较高；后台作业一般是对数据进行分析、输出数据、响应操作员请求等任务，优先级较低。后台作业中的任务只能在前台任务运行的间隙期间运行。前台作业与后台作业并非完全孤立，后台作业所需数据由前台作业存储在共享内存区内，作业之间通过共享存储区进行数据交换。

(2)实时任务总是由某个事件发生或时间条件满足来激活。

事件有内部事件和外部事件。时间驱动分为按绝对时间驱动和按相对时间驱动。内部事件驱动是指某一程序运行的结果导致另一任务的启动，这个结果可能是数据满足一定条件，也可能是释放了某一资源。最典型的实时任务是由外部事件驱动。在实时系统中，外部事件的发生有时是不可预测的。由外部事件驱动的任务一般是需要立即执行的任务，它的优先级最高。

绝对时间驱动是指在某指定时刻执行的任务，即在自然时钟的绝对时间执行。相对时间驱动是指周期性执行的任务，总是相对上一次执行时间计时，执行时间的间隔一定。除了周期性任务外，还有一些同步任务也可能由相对时间驱动，如等待某种条件到来，或者等待时间是由编程设定的。相对时间可用计算机内部时钟或软时钟计时。

在实际设计中，这二方面的问题都有所体现，所有的事件驱动和时间驱动都体现为设置相应的任务标识和线程标识。从后面的讨论中可以看出：当硬件环境一定时，根据这些标识并通过安排系统内中断响应方式和调整任务调度算法，可以有效解决多任务并行问题。这是因为系统的实时性主要取决于这二点。

## 2 多任务多线程机制的实现

本文介绍的设计是针对双通道和四通道测试的某型医用检验设备。每个通道可以置入样本,设置不同的测试项目,完成测试后输出不同的测试结果和附加的计算结果。

常规的处理方法是:各个通道只能测试同一个项目,按统一步骤同步执行各任务的相同阶段,其处理示意如图1。为简化起见,这里用双通道进行说明。

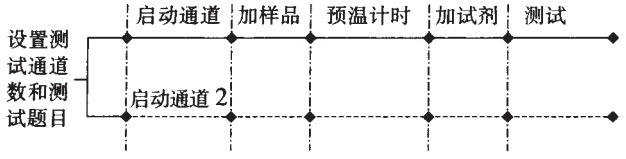


图1 多任务简单重叠

显然,这样做会失去测试的灵活性,例如:不能同时测量不同项目,不能随意在不同通道中测试不同样本,即使有空余通道也不能在上一样本测试过程中启动下一样本的测试。此外还牺牲效率、浪费时间,因为要等每个阶段最慢的一个处理完才能进入下一阶段。这种方法其实是单任务的多次简单重复,设计容易,但蔽端多。

本文选择了完全并行的设计方法,即要求所有通道可以完全独立工作、任意启动和停止、彼此没有约束、时间上可以任意重叠,是几个独立的任务,如图2。

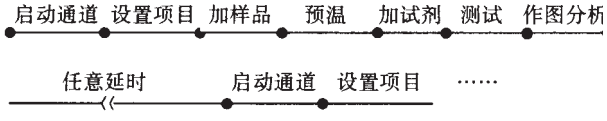


图2 独立并行任务

这里把对每一个启动通道进行测试的程序叫做一个任务。把各自任务下的每一个单独的、分开处理的程序段叫做一个线程。每个线程依靠自己的标识来识别。一个通道的测试任务可分为启动、设置、加样品、预温计时、加试剂与搅拌、通道轮流采样、数据处理和作图打印等多个线程。另外有一个温度的实时监控独立线程,它的优先级次于通道的测试采样。

这些线程可分属于前台和后台二类。前台主要是一些中断的处理,如二路温度的实时监控,每100ms内对各通道循环检测一遍,采用中断方式的键盘干预等。后台主要是扫描方式下的响应操作员的按键请求、数据处理、图形显示、打印报告等内容。

整个实现机制可以简单地概括如下:前台通过合理安排中断的响应和服务方式对多个任务的实时线程进行处理;后台操作则主要以循环方式扫描各个任务的线程标识,满足条件的线程被激活并进行处理。

限于篇幅,这里不可能把整个设计方案详细介绍,只能给出各测试通道工作任务的前台和后台线程划分及流程。本文还给出了一个中断退出后返回到任意地址的函数,它比C51自己的Setjmp和longjmp全程跳转函数的使用要方便很多。实时任务中,中断服务结束后一般不返回到断点地址执行原有程序,而是强制返回到某一地址去执行新程序。本文采用设置环境变量的方法,使中断退出后可以任意返回到多个设置入口中的某一个去执行。这种方法有效地解决了前台和后台任务线程的灵活切换这一关键问题。这里使用的CPU是97c52,编程语言为Keil C51 6.0版。

图3是主定时器中断服务,12c887提供中断请求信号至int0,12c887的3个中断触发服务中温度扫描是独立线程,4次500ms"周期中断"(即每2秒)后执行1遍;需要屏幕显示预温倒计时的时候使用"更新中断",每秒1次,各测试任务的倒计时线程依靠各自的标识启动和停止;"报警中断"需要时设置为每分钟1次,用于主菜单界面显示当前时间和长定时的返回。

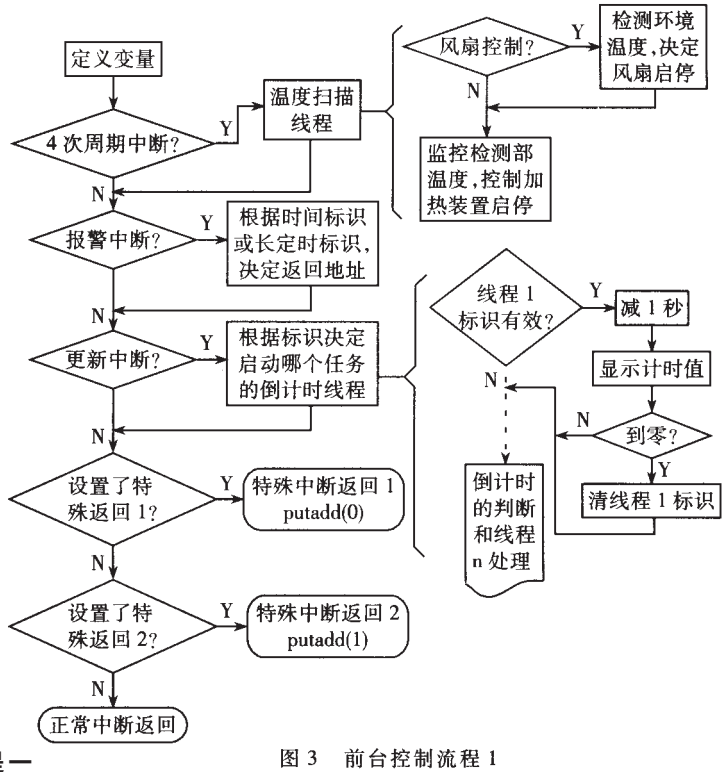


图3 前台控制流程1

图4是CPU内部定时器0的中断服务,用于A/D转换。每个测试任务的A/D分为2个线程:检测试剂加入和测试试剂样品的反应曲线。虽然都是通过对光学传感器的输出进行检测,但处理方法完全不同,数据量也不一样。定时器0设定为每100ms中断一次。因为采用高精度Σ-Δ转换器件,CPU必须直接控制器件的整个转换过程,所以要注意所有通道轮扫一遍A/D的时间不能超过100ms。