基于 RT-Linux 的雕刻机数控系统开发

Development of Engraving Machine Numerical Control System Based on RT-Linux

刘仁中 谢明红 孙友生 周国义 (华侨大学机电及自动化学院,福建 泉州 362021)

摘 要

针对现有雕刻机的一些不足,提出了一种基于 RT-Linux 的雕刻机数控系统。介绍了 RT-Linux 系统的系统结构和工作原理,叙述了雕刻机控制系统软硬件结构,并且通过一个控制实例,阐述了基于 RT-Linux 雕刻机数控系统的设计方法。实践证明,基于 RT-Linux 的雕刻机数控系统满足实时性要求,模块化的软件设计有利于系统的扩展与维护。

关键词:RT-Linux,雕刻机,数控系统,通信,模块

Abstract

In this paper,aiming at some of the existing shortage of engraving machine,a RT-Linux based on the engraving machine numerical control system is proposed.This paper introduces the system structure and working principle of RT-Linux, describes the structure of hardware and software of engraving machine control system,and explains the design mothed of the RT-Linux based on the engraving machine numerical control system through a control example.

Keywords:RT-Linux,engraving machine,numerical control system,communication,module

雕刻机的控制系统是数控雕刻机的核心,其控制功能的强弱,控制性能的优劣直接影响雕刻加工的效率和质量。目前应用的数控雕刻机主要有两类:

1)以 8 位单片机位内核的雕刻机。此类雕刻机有价格较低、设计比较简单,能够达到一般要求的优点,但也存在存储量小、实时性不强、定位精度不高、人机交互不理想而造成操作不方便等缺点。同时,其独立工作能力较弱,较强功能要与 PC 机联机才能实现。

2)基于 PC 机+运动控制卡的雕刻机。其弥补了 8 位单片机为内核雕刻机实时性不强、精度不够高等缺点,但此种雕刻机大多采用 DOS 或 Windows 操作系统,在软件的成本上有所增加,而且无论是作为单任务、过程式操作系统的 DOS 还是实时性能欠佳的 Windows 都有很多不如人意的地方。

基于以上雕刻机存在的不足,本文提出了一种以 RT-Linux 操作系统为开发平台的雕刻机数控系统。RT-Linux 作为一种硬实时操作系统,具有出色的实时性和稳定性,能很好地满足数控加工过程中多任务高精度的要求。文章结合数控系统实例,给出了在 RT-Linux 平台上构建软数控系统的方法,并对其中的一些关键技术进行了研究。

1 RT-Linux 简介

RT-Linux 实时内核是由美国墨西哥理工学院开发的,继承了 MERT 系统的设计思想,以 Linux 内核为基础,在系统中将严格的硬实时服务和所有的标准 POSIX 服务有机的结合起来,提供了一个透明的、模块化、可扩充的实时操作系统。RT-Linux 通过修改 Linux 内核的硬件层,采用中断仿真技术,在内核和硬件之间实现了一个小而高效的实时内核,并在实时内核的基础上形成了小型的实时系统。RT-Linux 实时内核采用的是一个基于优先级的可抢占的调度算法。实时内核将标准 Linux 系统当作它的一个最低优先级任务(idle task),它可以在需要的时候随时抢占 Linux 系统。由于实时内核同时接管了系统的中断控制,故能对硬件中断做出判断:若硬件中断请求一个实时任务的运行,实时内核能及时处理中断,先占其他优先级较低的任务,转而运行与之相应的中断服务程序;若硬件中断请求 Linux 任务进行处理,实时内核将截获这一中断,并在适当的时候模拟一个软件中断,通知 Linux 内核运行相应的中断服务。实时任务被编写成特殊的 Linux 模块,这些模块能被动态地装入内存中,而不会引发 Linux 系统调用。非周期性运行的任务通过使用中断即可获得支持。

RT-Linux 中的实时线程可以通过共享内存或 FIFO 管道与 Linux 中的进程通讯,这样实时应用程序就能够利用 Linux 的所有功能,比如图形功能、窗口系统、设备驱动以及 POSIX 标准 API。系统结构图如图 1 所示。

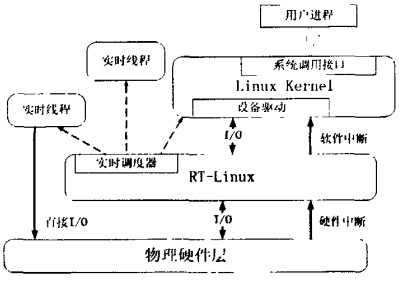

图 1 RT-Linux 系统结构图

RT-Linux 采用双内核共存的策略,有效地实现了硬实时特性。据测试,在一台 386 的机器上,RT-Linux 中断延迟时间不会超过 15μs,对于一个周期性任务在 35μs 内一定会执行。由此可见,RT-Linux 完全满足作为一个数控系统软件平台的要求。

2 数控系统的整体结构

数控系统是一个实时多任务控制系统,而 Linux 与 RT-Linux 的组合,以其出色的硬实时性和稳定性在工业控制领域获得了广泛的应用。这里我们选用 Linux-2.4.28+Rtlinux-3.2-rc1 作为系统的实时软件平台,以工控机作为系统的硬件平台,数控系统与伺服系统接口采用基于 ISA 总线的接口扩展卡 DAC-7330,该卡提供了 32 路隔离数字量输入输出通道及 32 个 TTL 数字量输入输出通道,可以将计算机输出的位置速度等指令转化为相应的脉冲串,驱动伺服系统,从而实现对机床执行件的运动控制。该系统的整体结构如图 2 所示。

实验目标机是一台激光数控雕刻机,雕刻机主体由两个步进电机通过导杆来实现激光刀头在平面上的位置控制。该雕刻机结构简单,很适合作实验用。

3 数控系统的软件结构

RT-Linux 中的任务进程运行在两种模式下:用户空间和核心空间。实时任务运行于核心空间,而非实时任务运行于用户空

间。基于 RT-Linux 的数控系统由硬件层、软件层构成。软件层又分为两部分：实时域和非实时域，其中实时域由用户定义的中断处理程序和实时任务组成，在数控系统中主要是完成位置控制、插补、刀具补偿、速度处理、实时状态监测等功能的模块，它们的运行具有强实时性，对资源的占有方式是独占。非实时域包括较高优先级的总控模块、程序读入译码模块和较低优先级的人机界面模块，它们都是普通的 Linux 程序，运行在 Linux 空间，它们没有实时性要求，对资源的占有方式是分时占有。RT-Linux 的实现机制要求基于它的实时应用程序必须划分成实时域和非实时域两部分，所有有实时性要求的任务均应设计成实时线程或中断处理函数，在实时内核中处理，所有可能引起资源竞争的操作必须放在非实时的 Linux 空间。数控系统的软件结构如图 3 所示。

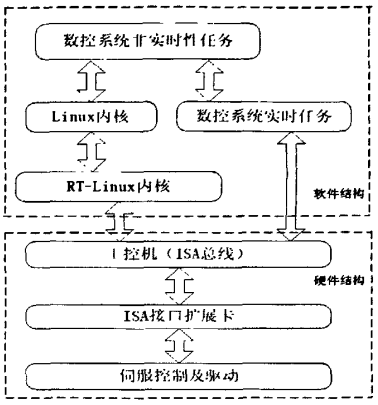

图 2 数控系统整体结构

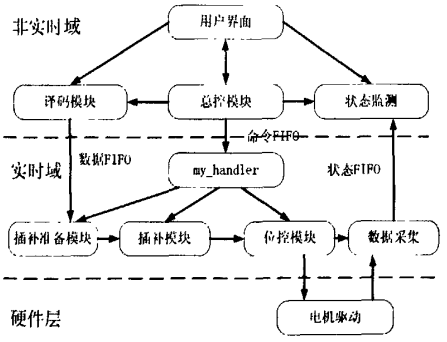

图 3 数控系统软件结构

如图 3 所示，用户通过人机界面模块读入 NC 代码，译码后经由数据 FIFO 到达插补准备模块；插补准备模块在读取数据 FIFO 中的内容后，间歇性地向插补模块发送数据，并完成相应的计算工作；插补模块则进行周期性的插补运算并将运算结果送至位控模块，位控模块依照插补结果实时地驱动电机；而 my_handler 函数随时监听命令 FIFO 管道，若有命令从总控模块传来，就立即处理相应的模块；数据采集模块获取位置及各种参数，然后由状态 FIFO 管道送至状态监测处理，并将需要的数据显示在用户界面上。

4 控制实例

这里通过一个实例来说明基于 RT-Linux 数控的实现方法，其主要功能是通过对雕刻机两个步进电机来实现平面坐标系第一象限的直接插补，所采用的插补方法是逐点比较法。

4.1 程序的基本结构

为简便起见，我们在实验中可以将插补准备模块中的速度等计算由插补模块处理，故此我们只需建两个周期性实时线程：插补线程和位控线程。如图 4 所示，首先插补线程收到用户空间传递的数据，然后 my_handler 函数监听命令 FIFO 管道，一断有开始命令后，立即唤醒插补线程，然后实现插补。位控线程同样通过 rt_handler 监听 FIFO 管道，一旦收到插补线程的内容后将自己唤醒，经处理后传至接口扩展卡来对相应的端口进行脉冲处理，从而实现对电机的控制。

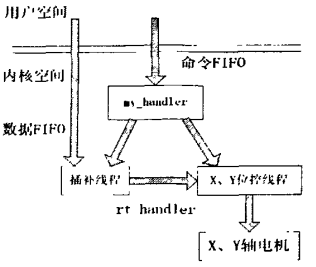

图 4 实验程序基本结构

整个实现过程中，比较关键的就是数据的传递。由于内核空间的模块和用户程序需要交换数据，所以我们一般通过定义一个共同的结构体的方法。由于只是第一象限的直线插补，所以译码处理的结构体只需要 X、Y 轴坐标以及进给速度，而命令管道传输的结构体应包括命令类型、插补周期、进给加速度等。

4.2 用户界面的实现

由于安装 RT-Linux 的基本系统平台为 Red Hat Linux9，可选用系统配有的 Qt3 来编写。Qt 自带的 Qt Designer 能很方便地编写用户界面程序。根据实验要求做的界面如图 5 所示。

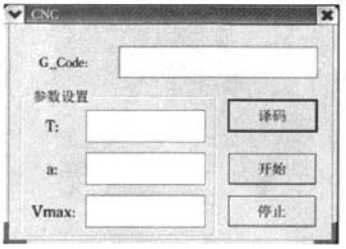

图 5 CNC 直线插补实验界面

如图 5 所示，参数设置栏里用于接受用户设置的插补周期、加速度、速度上限，G_Code 栏用于接收用户输入的 NC 代码。三个按键分别为译码、开始、停止。当点击译码时，界面调用译码模块，并将译码得到的结构体写入数据 FIFO；点击开始时，插补线程被唤醒；点击停止时，插补线程被挂起。

其程序结构如下：

```
#include <*.h>
void CNCDialog::init()  //初始化函数
{打开命令 FIFO;
...}
void CNCDialog::compile()
{...}
void CNCDialog::start()
{ msg.command = START;
  ...}
...
```

4.3 实时模块的实现

本例中实时模块包括 2 个周期性实时线程：插补线程、位控线程。插补线程首先获取由数据 FIFO 传来的译码结构体，然后通过 my_handler 函数监听命令 FIFO 管道，如果收到开始命令时，便唤醒自己，进行插补处理，若收到停止命令时，就将自己挂起。每进行一次插补循环，就唤醒一次位控线程，由位控线程对扩展卡进行读写来实现电机的控制。把这个思路用代码写出来的大体结构如下：

```
#include <*.h>
.....
void *chabu_code(void *t)  //插补线程
{....
rtf_get(1,&my_code,sizeof(my_code));  //读取数据 FIFO 中内容
..... //数据处理
While(1)
{int ret,err;
rtf_get(3,&msg,sizeof(msg));
```

```
...
switch(msg.command)
   {case START:
          pthread_make_periodic_np (pthread_self (),gethrtime(),
msg.T);  //唤醒线程
          break;
       case STOP:
          pthread_suspend_np(pthread_self());  //挂起线程
          break:
    ...}
While(n!) {    //终点判断
ret = pthread_wait_np;  //插补周期等待
.....  //偏差判断等处理
rtf_put(4, &ptrl,sizeof(ptrl)); //将插补处理结果传到位控线程
}return 0;}
void *poco_code(void) {...}   //位控线程
int my_handler(unsigned int fifo){...}   //设置命令 FIFO 句柄,监听
命令 FIFO
int rt_handler(unsigned int fifo){...}   //设置数据传输 FIFO 句柄,监
听命令 FIFO
int init_module(void)   //初始化模块
{...
rtf_creat_handler(2,&my_handler);   //建立响应命令的 handler,监听
2 号 FIFO
rtf_creat_rt_handler (4,&rt_handler);    //建立响应数据传输的 han-
dler,监听 4 号 FIFO
return 0}
void cleanup_module(void){...}   //清除模块
```

这段代码是很典型的线程代码,函数起动后,就进入线程等待,监听命令 FIFO 通道上的数据,通过 rtf_get 收到的命令来对线程进行处理,通过线程间的通道来实现通信,另外初始化和卸载是用线程来进行控制编程所必备的。

4.4 程序运行

整个程序包括实时域和非实时域,必须要分别编译成可执行文件。实时域部分我们通过编写 makefile 文件,然后 make 编译成模块文件 cnc_module.o,非实时域部分可以通过 Qt 专用的 qmake 工具编译成可执行文件 cnc_app。运行整个程序时,首先要加载实时模块,通过命令 insmod cnc_module.o,然后执行文件 cnc_app;程序执行完后,需要卸载实时模块,通过命令 rmmod cnc_module 即可。

5 结束语

实时性操作系统 RT-Linux 以其优异的实时性和稳定性,以及源代码公开等特点,在实时控制领域受到了越来越多的重视,并取得了一定的成绩。本文针对一般雕刻机的弊端,提出了一个基于 RT-Linux 来构建雕刻机数控系统的方法,而且通过一个实例说明其编程方法,更重要的是系统采用模块化设计,能具备很好的可扩展性。在以后的开发中若利用 Linux 优秀的网络处理功能以及 SMP 功能,会有更加好的效果。

参考文献

[1]殷苏民,邓英杰,程昌.嵌入式控制系统在雕刻机中的应用研究[J].控制与检测,2008(1)

[2]郭晋峰.基于 RT-Linux 的数控系统关键技术研究及软件开发[M].泉州:华侨大学,2001

[3]晏密英,刘刚.RT-Linux 的实时性检验与内核裁剪的研究[J].信息技术,2003,27

[4]王爱玲,张吉堂,吴雁.现代数控原理及控制系统[M].北京:国防工业出版社,2002

[5][加拿大]Jasmin Blanchette,[英]Mark Summerfield.C++ GUI Qt3 编程[M].北京:北京航空航天大学出版社,2006

[6]RT -Linux 内核驱动 http://baredog.at.infoseek.co.jp/intl/chn/robot/quad2/titechwire05.htm

[收稿日期:2009.4.27]

(上接第 33 页)

送,分系统较多,发送对象也较多,远程控制系统采用问答方式来接收数据,网络的开销较大。如果采用广播的方式发送指令或数据,需要接收的信息的分系统就去接收,而不需要该信息的分系统就不动作。同时,也可以进行分时段发送数据或指令,即错时发送。这样,可以有效减少信道负担,避免网络冲突和拥堵。在该系统设计中,网络延时小于 10ms。

2)在通信过程中,会出现发出的指令或请求后,对方没有响应,但远程控制计算机必须接收到反馈信号后才能继续动作,这种情况下网络就会一直待机或死锁。为解决基于 UDP 协议的通信死锁的问题,在应用程序中设计了超时计时器,当发出指令或请求时,计时器开始计时,在规定的时间内(如 300ms)如果没有收到反馈信号,即重发第二次,如果第二次发送在规定的时间内仍未得到反馈信号,则给出系统故障指示。

3)基于 VC++6.0 编程的串口通信实现方式有 MSComm 控件、WinAPI 函数和 CserialPort 类三种途径,这三种方式实现串口通信数据的接收有数据驱动(消息响应)和定时查询(时钟中断)的方法。许多测控系统的串口通信常使用定时查询的方法,即设定一个定时器定时访问串口,获取分系统发到串口的数据,这种方法有一个比较严重的缺点就是接收和发送时间不同步,定时查询时可能获得的是"乱包"或不完整的包,使数据的传输可靠性大大降低。采用数据驱动的方法,即添加串口响应函数,只有当数据发送到串口时才响应,接收数据,这样就能保证在每次读串口时,数据是一个完整的包,有效避免了"乱包"或"半包"现象,大大提高了数据传输的可靠性。

3.3 软件的防错性

在软件实际设计过程中,设计人员发现:软件的防错设计是非常重要的,尤其是现在很多程序,设计人员和操作人员是分开的,软件的设计应能够判断操作人员的误操作,并给予相应的提示,对于错误的输入或操作拒绝执行。编程人员在编码时,采用不易混淆的变量名或函数名等来命名,尽量采用经过实践考验可重用的软件模块。

4 结束语

通过对基于分布式测控系统可靠性设计的工程经验总结,得到以下结论:系统的可靠性从设计开始,从软硬件的设计着手,遵循可靠性设计的规范。可靠性设计也不是一蹴而就的,通过可靠性分析软件或在试验中反复验证,不断修改,将可靠性设计贯穿工程研制的始终,最终达到提高系统可靠性的目的。

参考文献

[1]徐惠钢,薄煜明,吴钦.分布式火控系统实时通信的设计实现[J].火力与指挥控制,2005,30(6):109-112

[2]龚庆祥.型号可靠性工程手册[M].北京:国防工业出版社,2007

[3]潘石柱,于仲安.VC 实现串行通讯的三种途径[J].电子工程师,2002,28(9):14-17

[收稿日期:2009.2.8]