文章编号:1007 - 2985(2011)04 - 0064 - 04

# 基于 QNX 的 CPCI 板卡通信测试系统设计与实现*

余朝兵
(吉首大学张家界学院,湖南 张家界 427000)

摘 要:以 PC104 工控机为处理平台,探讨了基于此平台之上的板卡通信测试系统的设计. 软件选用 QNX 实时操作系统,硬件上结合了 CPCI 总线技术,设计并实现了性能良好的测试系统,该设计方法为实时系统下的各种总线标准板卡间通信实现提供了一定的借鉴.

关键词:QNX;CPCI 总线;实时系统
中图分类号:TP303$^{+}$. 2 文献标志码:A

## 1 QNX 体系结构与资源管理器

### 1.1 QNX 体系结构

QNX 是以微内核为架构、建立在完全地址保护空间的实时操作系统,支持多种架构 CPU,如 X86,PENTIUM,POWER 等,支持多种总线标准,如 ISA,EISA,PCI,CPCI 等,支持各种外设,如 SCI、IDE、各种网卡等,完全遵循 POSIX. 1 和 POSIX. 2 接口标准,部分遵循 POSIX. 1b 标准,表现出非常强大的可移植性. 与 LINUX 不同的是,其系统内核非常之小,仅 1 个 1. 44 MB 软盘就能完全保存其内核系统文件,原因在于 QNX 只提供了一些最为基础的服务和基于消息间的进程通信机制,这些服务包括:消息、线程、信号、同步传递、时序安排、定时器服务. QNX 发展至今,其最新内核版本为 Nutrino6. 5,广泛应用于航空航天、汽车电子、手持设备等领域.[1—2]

### 1.2 QNX 资源管理器

在 QNX 中,所有的服务器进程被虚构为 1 个设备,所有的设备都由 1 个叫作设备资源管理器的进程管理. 这样,不论是物理设备,还是虚拟的文件设备,都可以视为 1 个服务器进程. 其另 1 个重要优势在于,资源管理器运行于内核态之中,目的在于以传递消息来提供最基础的服务,既实现了实时性的需求,又达到了保护系统稳定的目的. 从编程角度来讲,QNX 与 LINUX 的设备驱动程序编写步骤具有类比性,一般步骤为:

(1) 初始化设备;

(2) 待设备正常初始化后,利用标准接口在内核与设备间进行数据存取;

(3) 在用户程序中,调用系统接口完成应用与设备驱动程序间的数据交互;

(4) 设备操作的异常处理,如中断处理、直接存取错误处理等.

1 个资源管理器进程对应着 1 个设备. 当资源管理器运行后,用户即客户端,可以通过消息传递服务与资源管理器进行通信,从而间接完成与相应的设备交互,相关的操作有 open()、read()、write()调用等. 在 QNX 中,有 1 个路径名映射机制将资源管理器与 1 个路径,如/dev/cpci_a 关联起来. 通讯过程说明如下:

* 收稿日期:2011 - 05 - 21
作者简介:余朝兵(1980 -),男,湖南吉首人,吉首大学张家界学院教师,主要从事计算机科学与技术研究.

(1) 客户查询. 用户程序向进程管理器查询所要访问的资源管理器是否存在;

(2) 回复查询结果. 收到查询消息后,进程管理器在路径名空间中查找,如不存在,表示用户所需访问的设备不存在或资源管理器未能运行,返回错误,客户端进入异常处理;

(3) 建立连接通道. 如第(2)步返回成功,则用户向资源管理器发送 1 个连接消息请求建立 1 个通信通道,表示用户需要设备服务;

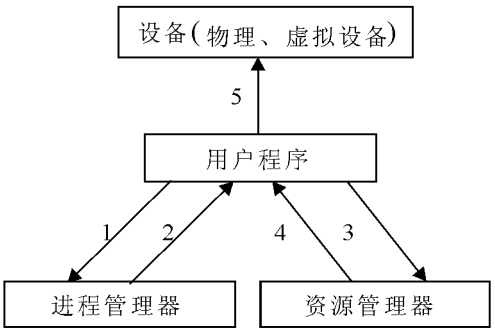


图 1　资源管理器、客户端和进程管理器间通信模型

(4) 资源管理器返回建立通道结果. 无论建立成功与否,回复信息中均包含本次连接的文件描述符;客户通过判断文件描述符(fd)的值来确定是否已经成功建立通道,fd 小于 0,表示失败.

(5) 设备通信服务. 当成功建立通道后,fd 就对应了 1 个通信通道,后续的交互操作,客户端就可以使用此 fd. 当有多个客户端需要同时与设备进行通信时,资源管理器将为客户返回不同的 fd.

## 2　测试系统功能模块设计及实现

### 2.1 测试系统组成

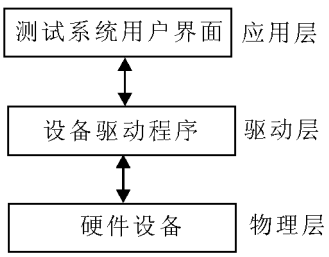


图 2　测试系统软件体系结构

测试系统软件体系结构如图 2 所示,图 2 中的设备驱动程序实际就是 QNX 中的设备资源管理器,用户界面通过驱动程序与硬件设备进行交互.

### 2.2 CPCI 设备资源管理器的编写

设备资源管理器是 1 个用户级服务器进程,它使用"发送—接收—回复"消息协议完成与客户端的 1 次会话. 设备资源管理器体系结构如图 3 所示.

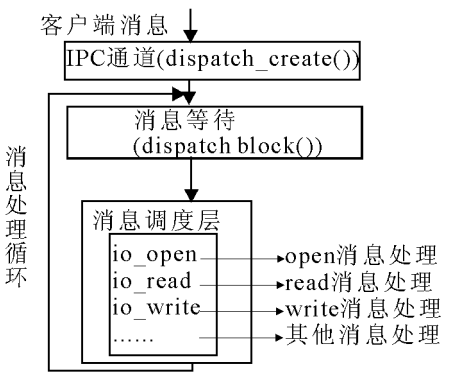


图 3　资源管理器体系结构

QNX 将设备资源管理器在逻辑上划分为 4 层,即功能函数层,消息调度层,消息转发层,线程池层. 功能函数层用于实现设备的功能,处于最高一级,由一系列符合 POSIX 标准文件系统的输入输出函数组成. 通常在这层里为设备处理消息函数接口提供定义,比如 read()和 write()接口. 对于未定义消息接口,系统将提供 1 个默认的消息处理函数,此类函数的形式如下iofunc_*_default()(其中"*"表示文件处理操作如 open、read、write、close 等,下同). 功能函数层的所有函数和相应数据结构都被定义在头文件<sys/iofunc.h>中,命名格式为 iofunc_*. 消息调度层完成对接收到的消息进行检查,检查内容包括消息类型、格式等,在消息检查正确时,调用上层用户自定义或系统默认的消息处理函数,否则,作出错误处理. 消息转发层(dispatch)用来循环提取客户端发来的一条消息,通过 dispatch_handler()来调用相应的处理函数,特别是,当 1 条消息处理未完成时,此层接口可以将后续到来的消息放入消息队列,直到前 1 条消息处理完毕再向上层提交消息,这样可以避免造成消息处理序列混乱. 线程池层主要用于指定设备运行的线程属性,即单线程或多线程. 对于资源管理器来说,大部分情况下,利用 1 条线程池启动语句即可.

QNX 中设备资源管理器编程的规范如下:

(1) 资源管理器的初始化. 具体包括以下几步:(a)初始化消息转发层 dispatch 接口. 资源管理器是基于消息传递的. 系统 dispatch_create()函数用于创建并返回一个消息传递通道. 返回的通道数据结构里包

含了通道 ID.(b)初始化资源管理器属性.资源管理器属性结构主要用来指定该资源管理器与客户端消息传递的一些限制.如可用于回复服务(nparts_max)的 IOV(输入输出数组向量大小)数据结构个数、消息最小接收缓冲区大小(msg_max_size).(c)初始化消息处理函数.实现对于特定设备我们所关心的客户端消息,其他消息则由系统提供默认的处理接口处理.(d)初始化设备用属性结构.主要是初始化打开控制块(OCB)结构.(e)向进程管理器注册路径名信息.将 1 个设备资源管理器关联到 1 个特定的唯一的路径名,用于客户端在进程系统中找到需要控制的设备.(f)启动资源管理器消息处理.此消息处理应是 1 个 while(1)类型的无限循环过程,当客户端发来 1 个消息时,资源管理器检查消息类型、查询是否有用户实现的处理接口并调用,当没有针对该消息处理的用户接口时,则调用系统默认处理接口.

(2) 定义 io_devctl()与硬件设备交换数据的结构体,并实现此函数.Io_devctl()函数是 1 个非常重要的调用,功能非常强大,几乎能将所有的消息处理都放在此接口中实现.但实际应用中,出于可读性的考虑,只处理用户自定义消息.POSIX 标准消息如读、写等,则不被包括在 io_devctl 中处理.客户端是通过系统调用来完成与资源管理器的 1 次信息交换,如 open()、read()等,资源管理器应该实现这些接口.所以,在设备资源管理器中,我们会通过 io_devctl()来完成 1 次比 read()和 write()更多功能的数据交换.

(3) 初始化 CPCI 设备.CPCI 在 QNX 中的软件管理与配置与 PCI 是一致的.此步主要是侦测 PCI 设备是否存在、读出 PCI 设备在系统中的配置.QNX 规定,只要在程序中用到诸如 pci_开头的函数系列,就必须先调用 pci_attach(),否则内核会产生 1 个访问错误.当成功连接上 PCI 服务器后,接下来应该侦测某个 PCI 设备是否存在,相关函数为 pci_find_device(),它使用两个重要的参数来搜索设备,即 PCI 设备厂商号和设备号.如果 PCI 设备存在,读取其系统配置信息:I/O 基地址、中断号等以备后用.PCI 设备配置信息放在 PCI 设备信息结构体 pci_dev_info 中,此结构体的系统定义见表 1.

表 1 pci_dev_info 结构系统定义

| 类型 | 成员 | 含义 |
|---|---|---|
| uint16_t | DeviceId | PCI 设备号 |
| uint16_t | VendorId | PCI 设备厂商号码 |
| uint16_t | SubsystemId | 子系统 ID |
| uint16_t | SubsystemVendorId | 子系统厂商 ID |
| uint8_t | BusNumber | 总线号 |
| uint8_t | DevFunc | 设备功能号 |
| …… | …… | …… |
| uint32_t | Irq | PCI 设备中断号 |
| uint64_t | PciBaseAddress | PCI 基地址矩阵 |
| uint32_t | BaseAddressSize | 各基地址长度 |
| uint64_t | PciRom | PCI ROM 地址 |
| uint64_t | RomSize | ROM 地址长度 |
| uint64_t | CpuRom | CPU ROM 地址 |
| uint64_t | CpuBaseAddress | CPU 基地址矩阵 |

成功调用 pci_attach_device()获取系统获取 PCI 设备分配的信息,返回 pci_dev_info 结构,其结构成员包括:中断号 Irq、设备基地址 PciBaseAddress 等.Irq 是设备外部唯一中断号,而设备基地址是 1 个长度为 6 的数组,如何从这个数组中找出所需要的内存和 I/O 端口基地址呢?可以利用系统定义 PCI 宏:PCI_IS_IO、PCI_IS_MEM、PCI_IO_ADDR 和 PCI_MEM_ADDR,它们只有一个参数即地址值,将 PCI 设备基地址数组中的六个成员分别传入 PCI_IS_IO 和 PCI_IS_MEM 可以判断出 I/0 基地址和内存基地址,当得到基地址后,将相应的值传入 PCI_IO_ADDR 和 PCI_MEM_ADDR 则可以转换为后续函数可用作地址访问的基地址值形式.最后,根据获取的信息为 PCI 设备配置具体应用环境,如设置中断屏蔽寄存器等.

(4) 编写中断处理程序.QNX 作为实时微内核结构的系统,事件响应延时非常短,其中断实现也非常简单,即由一条线程通过系统库函数挂接一个中断处理规程 ISR(中断报务程序),客户端线程在一个循环里无限地调用 InterruptWait()阻塞,直到某个中断发生时,ISR 将被内核直接调用一次来处理硬件中断,然后进行下一次中断…….

(5) 启动设备资源管理器.设备资源管理器分单线程和多线程环境.在单线程环境下,需要程序安排消息循环处理,多线程环境下,使用线程池技术启动,即简单地调用 thread_pool_start().

(6) 设备资源管理器的自动运行处理.通过配置系统 SHELL 脚本实现设备资源管理器的自动运行.

## 3　结语

基于 QNX 的测试系统,使用 C/C++作为开发语言,采用了 POSIX 应用程序接口、CPCI 技术以及大量的进程、线程通讯等比较主流的技术,对于实时领域中的应用开发都具有参考价值,且采用了模块化开发理念,也更易于日后维护和升级.

**参考文献:**

[1] 魏永明. Linux 设备驱动程序 [M]. 第 2 版. 北京:中国电力出版社,2002:121－130.

[2] 王　勇. GNU/Linux 编程指南 [M]. 北京:清华大学出版社,2000:97－114.

[3] QNX Software Systems Ltd Limited. System Architecture [EB/OL]. [2011－05－21]. http://www.qnx.com.

## Design and Implementation of Communication Test System for QNX Based on CPCI Boards

SHE Zhao-bing

(Zhangjiajie College, Jishou University, Zhangjiajie 427000, China)

**Abstract**: This paper explores the design of communication test system based on the PC104 IPC processing platform. As for software, the popular QNX real-time operating system is used; as for hardware, the CPCI bus technology is used. Through fully using the greatest potential of both aspects, the test system with good performance is successfully designed and implemented. In addition, under the real-time system the methods described in this paper are also to provide reference for a variety of bus standards to achieve communication between the boards.

**Key words**: QNX; CPCI bus; real-time systems

(责任编辑　陈炳权)

(上接第 59 页)

## Study of the Powder Particles' Friction in HVC Based on Fractal Theory

GU Cheng-ling

(Laiwu Vocational and Technical College, Laiwu 271100, China)

**Abstract**: According to the original proposal of Bowden and Tabor and based on the fractal theory, the total adhesive friction coefficient is expressed as a combination of the adhesive friction coefficient in elastic and plastic regime. Then a fractal model of adhesive friction coefficient is proposed. And a mathematical model of the relationship of powder particles' internal friction and fractal parameter is built. Through the function curves, it is analyzed that the fractal dimension of the surface of the powder particle has impact on the total adhesive friction coefficient and the inner friction. The study shows that for certain fractal dimension, the total adhesive friction coefficient reduces sharply as the normalized contact area increases and the inner friction can reach the value of plastic shearing intensity rapidly as the fractal dimension and the temperature of the surface of the powder particle increase.

**Key words**: fractal dimension; HVC; adhesive friction coefficient; internal friction

(责任编辑　陈炳权)