UNIX环境中实现PCI接口设备驱动

别其璋*

(重庆工商大学计算机与信息工程学院 重庆 南岸 400033)

【摘要】以UNIX操作系统Solaris 8环境中设计PMC-Sirra 7364卡驱动程序为例，探讨在UNIX系统下PCI接口设备驱动程序的开发技术，包括设计思想、基本步骤、程序构架、常用函数以及PCI设备的系统管理命令等。其方法可以推广应用于UNIX环境中对CompactPCI接口设备的驱动程序开发，为使用UNIX主机代替PC工业控制机实现高稳定、高可靠电信设备和控制设备提供了一条新的途径。

关 键 词 UNIX系统; PCI接口; Solaris环境; 设备驱动程序

中图分类号 TP311.1 文献标识码 A

Development of PCI Device Driver under UNIX

Bie Qizhang

(College of Computer and Information Engineering, Chongqing Technology and Business Univ. Chongqing Nanan 400033)

Abstract Based on the design of PMC-Sirra 7364 card device driver under Solaris 8 of UNIX system, the development technology of PCI device driver under UNIX system is discussed. It presents the idea of design, basic programming procedure, program structural framework, some useful functions and the system management commands. This method can be further applied to the development of CompactPCI device driver under UNIX, which provides a new approach to replacing PC industrial control computer with UNIX host to implement telecommunication and control devices with high stability and reliability.

Key words UNIX system; PCI; solaris; device driver

随着信息工业的发展，传统通信行业和计算机行业的结合越来越紧密，在呼叫中心，短信中心，智能网等电信应用中，广泛采用PC工业控制机来实现电信网设备和应用系统的连接控制单元。PC工业控制机虽然结构简单，开发方便，接口丰富，但也有先天的缺陷。硬件体系上，PC工业控制机是对传统PC主板进行电路和结构优化设计，使用普通的Intel Pentium处理器；系统软件上，采用windows系统或者Linux系统，这两种操作系统在作为网络或客户端应用时还可以接受，而在实时性、稳定性、可靠性要求很高的电信级设备应用中则无法满足电信部门的要求。所以如何设计一种更安全可靠的连接控制设备是许多电信设备生产商正在积极探讨的问题。

UNIX操作系统及其主机系统以其高可靠性、稳定性以及严密安全性被广泛应用于电信、银行、工业控制等领域。在电信部门中承担数字计算、数据存储等任务的交换机都使用UNIX的主机来实现。因此，如果能用UNIX主机代替上述通过PC工业控制机实现电信网络和计算机系统的连接控制单元，就可满足电信级可靠性、稳定性和安全性的要求。

2002年12月30日收稿

* 男 57岁 副教授 主要从事电子与信息技术应用方面的研究

1 系统构架

本段以一个模拟短消息业务互连IWMSC设备为例来介绍使用UNIX主机代替上述通过PC工业控制机实现电信网络和计算机系统的连接控制单元的方法。该设备要求具备收发七号信令的功能，同时在系统中还需完成数据整合、数据查询等功能，因此采用计算机系统连接信令链路接口单元，构成以计算机系统为核心，以链路接口板收发七号信令网信息的体系结构。该系统构架如图1所示。

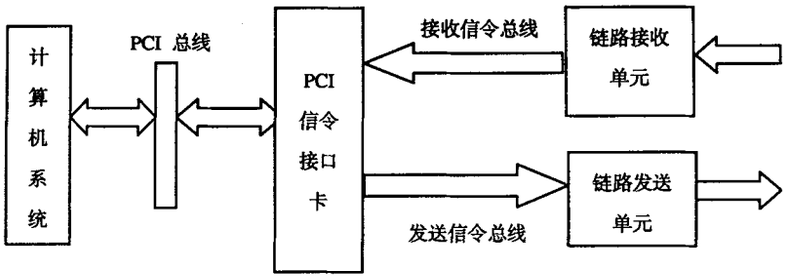

图 1 系统构架

1.1 链路接口单元

需完成对国内E1接口(2 048 Kbit/s)链路的接收和发送，采用高阻或终端方式接入的2 M信号通过解码器LXT360进行解码，再经过成帧器PM6344进行信号再生和成帧过程，然后进入交换器MT8985，完成信令的时隙交叉，最后进入PCI信令接口卡。

1.2 PCI信令接口卡

采用PMC-Sirra公司的PCI总线接口的多通道HDLC控制器7364芯片为核心，设计一个支持32个E1口链路，128个双向HDLC通道的接口板。该接口板可实现计算机系统通过PCI总线完成信令消息的接收、控制和发送过程。

1.3 计算机系统

采用SUN公司的UltraSPARC架构的UNIX工作站或者服务器，操作系统采用SUN公司版本的UNIX：Solaris 8 for SPARC操作系统。

2 计算机系统和PCI信令接口卡的分析和程序处理

2.1 PMC-Sirra 7364特性及内部结构

PMC-Sirra 7364是一个支持32个E1口链路，128个双向HDLC通道的大规模集成电路,它具有如下特性：时隙工作方式下，支持32个E1口，128个多时隙链路的收发控制；支持32个速率达2 M/s的无时隙HDLC通道的收发控制；支持32 bit，33 MHz PCI 2.1接口及升级接口的电气特性和功能特性。PMC-Sirra 7364的内部功能模块如图2所示。

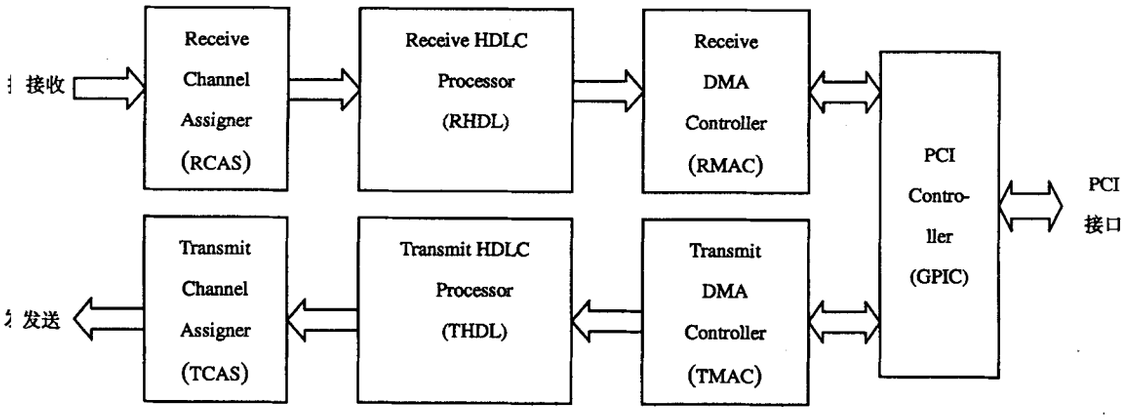

图 2 PMC-Sirra 7364 的内部功能模块结构

该模块工作原理如下：

1) 在接收侧，E1口链路数据通过接收通道分配器(RCAS)完成时隙分配和串并变换，进入接收HDLC处理器(RHDL)，再由它接收数据发送到部分包缓冲区，部分包缓冲区通过接收DMA控制器(RMAC)完成部分包缓冲区到主机内存的DMA存取操作，从而主机应用程序可接收信令消息。

2) 在发送侧，主机应用程序产生需要发送的信令信息，写入主机内存，由发送DMA控制器以DMA方式传送到部分包缓冲区，发送HDLC处理器处理部分包缓冲区中信息，完成HDLC成帧过程。再将信息发送到发送通道分配器(TCAS)，由发送通道分配器，根据时隙配置发送到相应E1口链路中。

3) 实现信令收发功能，关键是要完成PMC-Sirra 7364卡工作模式设置，寄存器映射，以及申请连续的物理内存来完成DMA操作。

2.2 设备驱动程序特点

PMC-Sirra 7364卡的芯片采用DMA工作方式，通过映射主机物理内存来实现DMA工作。因此，要完成对PMC-Sirra 7364卡的设备驱动程序设计，在计算机系统中需完成如下工作：完成对PMC-Sirra 7364卡物理设备的标识；存取和修改PMC-Sirra 7364卡的PCI配置空间；获取并映射PMC-Sirra 7364卡的配置地址寄存器，实现对配置地址寄存器的读写；申请高达2 MB连续的物理地址来完成DMA操作。

2.3 Solaris系统设备驱动程序设计

SUN Solaris 8操作系统提供DDK/DDI开发环境，可使用Forte C++, gcc等开发工具进行设备驱动程序开发。

2.3.1 Solaris对PCI设备开发过程

在SUN主机系统中插入PCI卡，打开计算机，按如下步骤开发PCI卡驱动程序[1, 2]：

1) 找到该PCI卡在Solaris系统的设备树下的位置，使用prtconf命令，在设备树中的PCI树下可发现“pci11f8,7364”名称，记录该卡的父节点，可能为“/pci@1f, 4000/pci@1 /pci11f8, 7364”。

2) 编辑该PCI卡的配置文件，格式为设备名.conf文件，本项目中为pci7364.conf。该文件记录设备的名称、属性、父节点等信息，把该文件拷贝到/usr/kernel/drv/sparcv9目录，供添加设备addr_drv命令使用，内容为：

name =“pci7364”class =“pci”parent =“/pci@1f,4000/pci@1”

其中：name表示设备驱动程序名称，class表示设备驱动程序的类型，parent表示该设备在PCI设备树中的父节点的设备名。

3) 编辑PCI卡设备驱动程序文件，本例为pci7364.c，编译后文件为pci7364, 把该文件拷贝到/usr/kernel/drv/sparcv9目录。

4) 调用add_drv添加设备驱动程序，本例为：

add_drv-m‘0 0600 root sys’-i“pci11f8,7364”'pci7364

5) 建立设备链接文件名。调用add_drv后，在/devices目录树中可找到“pci11f8,7364”，该文件为PCI卡的驱动程序，须在/dev目录下生成该驱动程序的链接文件供应用程序调用。调用如下命令实现：

ln–s /devices/pci@1f,4000/pci@1/pci11f8,7364@0,0:a /dev/pci7364

这样，完成驱动程序设计，可编辑调用该驱动程序的应用程序。

2.3.2 Solaris环境下编译设备驱动程序

Solaris下采用C语言来编译设备驱动程序，可采用Forte C或gcc编译器，Forte C是SUN公司提供的专门用于Solaris环境的C开发环境，使用它编译设备驱动程序的过程如下：

1) 编译生成.o文件

cc-D_KERNEL-xarch=v9 -xcode=abs32-c pci7364.c

其中,-D_KERNEL这个编译选项表示编译在系统核心环境中运行的代码，设备驱动程序需使用该选项。

2) 生成设备驱动程序代码

ld-r-o pci7364 pci7364.o

这样产生该PCI卡的设备驱动程序pci7364。按2.3.1节步骤可加载该驱动程序。

2.3.3 Solaris下编写PMC-Sirra 7364卡设备驱动程序

下面编制PMC-Sirra 7364卡源文件pci7364.c,限于篇幅，本文提供一个编程概要，某些过程或函数的具体意义可参见SUN公司的DDK/DDI编程手册。

1) 构建Solaris环境下设备驱动程序框架

当系统装载和卸载驱动程序时，会使用modlinkage结构来调用_init()或者 _fini()过程，modlinkage结构为设备驱动程序的模块连接结构，主要包含以下结构:

```
modlinkage->
        modldrv->
                dev_ops->
                        cb_ops
```

其中dev_ops结构定义该设备驱动程序支持的操作入口函数，它包含一个cb_ops结构的参数，用于定义驱动程序支持的各种操作，类似UNIX下文件系统所支持的open(), close(), read(), write(), ioctl()等操作。完成数据结构定义后，可编写系统装载设备的入口函数_init()和卸载设备的入口函数_fini()。然后，根据设备要求实现设备支持的入口函数设计。

2) 实现PCI配置空间存取和物理内存申请

要完成对PMC-Sirra 7364卡的设备驱动，在计算机系统中需存取和修改它的PCI配置空间，获取并映射它的配置地址寄存器地址，申请高达2 MB连续的物理地址来完成DMA操作。具体如下：

(1) 存取和修改PCI配置空间

DDK/DDI提供pci_config_setup()函数建立PCI设备，然后可调用pci_config_get16()，pci_config_put16()等函数实现配置存取和修改，最后调用pci_config_teardown()完成修改过程。

(2) 获取并映射配置地址寄存器地址

每个PCI设备有4个配置地址寄存器，PMC-Sirra 7364卡使用第二个(序号为1)地址寄存器来存放各种配置信息，因此需映射该地址寄存器的供设备驱动程序使用。首先，获得地址寄存器的尺寸：

```
ddi_dev_regsize(rs->dip,1,&nregsize[1]);
```

接着建立该地址寄存器的映射：

```
ddi_regs_map_setup(rs->dip,nbaseaddrreg,&rs->mem_reg,0,
nregsize[nbaseaddrreg], &endian_attr, &rs->handle1)；
```

然后可直接根据地址偏移量读取配置寄存器,读操作为：

```
ddi_get32(handle,(uint32_t*)(mem_reg+dReg));
```

写操作为：

```
ddi_put32(handle, (uint32_t*)(mem_reg+dReg), dValue);
```

(3) 申请连续的物理地址来完成DMA操作

首先建立一个DMA内存对象的属性描述结构：ddi_dma_attr_t，包括DMA操作的版本号，地址范围，PCI处理突发数据的尺寸，最小传输数据量，最大传输数据量等信息。然后，获得DMA操作的句柄:

```
ddi_dma_alloc_handle(rs->dip, &pci7364_dma_attr, DDI_DMA_SLEEP,
    NULL, &rs->dma_handlep_rbuf)；
```

申请2 M连续物理内存：

```
ddi_dma_mem_alloc(rs->dma_handlep_rbuf, 2*1024*1024, &dev_attr,
    DDI_DMA_CONSISTENT, DDI_DMA_SLEEP, NULL, (caddr_t*)&rs->rlVIRptr,
    &real_len, &rs->acc_handlep_rbuf)；
```

申请的物理内存所对应的核心虚拟内存为rs->rlVIRptr。DDI_DMA_CONSISTENT表示连续的物理内存。

由于PMC-Sirra 7364卡在配置DMA操作时需把申请的物理内存的物理地址写入配置寄存器中，所以还需获得内存的物理地址：

```
ddi_dma_addr_bind_handle (rs->dma_handlep_rbuf, NULL, (caddr_t)rs->rlVIRptr, real_len, DDI_DMA_
```

RDWR|DDI_DMA_CONSISTENT, DDI_DMA_SLEEP, NULL, &rs->dma_cookiep_rbuf, &count)；

其中，返回的rs->dma_cookiep_rbuf结构中的dmac_address域为物理内存的物理地址。

3) 完成PMC-Sirra 7364卡自身寄存器配置

完成对PCI卡配置空间，配置地址寄存器和连续的物理内存申请后，可根据具体PCI卡的要求进行PCI卡本身寄存器的配置工作。这里就不做详细介绍。

4) 建立驱动程序和用户程序间调用接口

设备驱动程序安装后，可在应用程序中进行调用，UNIX把设备驱动程序看作特殊文件系统，可采用文件系统的调用方式进行设备驱动程序操作。

打开设备驱动程序：

open(“/dev/pci7364”, O_RDWR)；

对该设备进行IO操作：

ioctl(fd,IOCTL_SETSLOT,LINKTSLOT)；

应用程序数据和设备驱动程序数据的交互[3]：

方法一：可采用内存映射方式，在设备驱动程序申请一块内存，然后在设备驱动程序中实现devmap()入口函数。在应用程序中调用内存映射函数mmap()建立内存映射。

方法二：实现设备驱动程序的read(),write()操作，应用程序就可以象读写文件一样读写设备驱动程序。

方法三：在设备驱动程序的ioctl()中可实现应用程序和驱动程序间数据传递。

3 结 束 语

通过上述过程，在SUN的UNIX系统Solaris 8上完成PCI卡的设备驱动设计，实现在UNIX系统下电信网络和计算机系统的连接控制单元设计，使产品既满足接口的灵活性，又在性能、安全、稳定等方面满足电信部门的要求。同时还可以把UNIX下PCI设备驱动程序的设计技术推广应用于UNIX对CompactPCI接口卡的驱动程序开发，在UNIX环境下实现对PCI接口和CompactPCI接口设备的驱动，从而为电信系统，工业控制等部门提供高可靠性、高稳定性的基于UNIX系统的控制设备。

参 考 文 献

1 Nemeth E著. UNIX系统管理技术手册[M]. (第三版). 董俊华译. 北京：人民邮电出版社, 2002

2 Robbins K A, Robbins S著. 实用UNIX编程[M]. 刘宗田，孙志勇，秦宗贵译. 北京：机械工业出版社，1999

3 Rubini A著. LINUX设备驱动程序[M]. Lisoleg译. 北京:中国电力出版社, 2001

编　辑　王　燕

·科研成果介绍·

大功率微波发射机运行参量遥传与显示分机

主研人员：雷　霖　高昕艳　成永东　谢文楷　钱光弟

大功率微波发射机运行参量遥传与显示分机采用PLC及配套的I/O模块，对大功率发射机运行参数、工作状态等进行实时采集与遥传显示等功能。采用VC++语言和面向对象程序设计技术，实现上位微机的基于windows98中文图形界面的监控程序，采用梯形图程序设计技术实现PLC程序。

·渠　涌·