VxWorks 下字符设备的驱动开发

郭德磊

(中国飞行试验研究院,陕西 西安 710089)

摘要:随着 VxWorks 操作系统在嵌入式系统中的应用,VxWorks 下产品的开发和应用也越来越广泛。开发嵌入式设备经常遇到的一个问题就是如何编写高效可靠的设备驱动程序。该文对 VxWorks 下设备驱动程序开发的几个主要方面以及如何配置 PCI 总线设备等进行了详细介绍,为相关的开发人员提供了很好的参考。

关键词:VxWorks;字符设备;驱动开发;PCI 设备;select 功能

中图分类号:TP311　　**文献标识码**:A　　**文章编号**:1009-3044(2011)01-0121-02

VxWorks Device Driver Development Under the Character

GUO De-lei

(China Flight Test Establishment, Xi'an 710089, China)

Abstract: With the VxWorks operating system in embedded system, VxWorks under the product development and applications are increasingly widespread. Development of embedded devices is a problem often encountered in how to write efficient and reliable device drivers. In this paper, VxWorks device driver development under the main aspects and how to configure the PCI bus devices are described in detail for the relevant developers a good reference.

Key words: VxWorks; character device; driver development; PCI device; select function

VxWorks 是一个基于抢占式的实时操作系统,它以其高度的可靠性、优秀的实时性、灵活的可裁性广泛应用到越来越多的嵌入式装置中。在这些嵌入式装置中,往往有一些自研的板卡设备,要使这些板卡支持 VxWorks 操作系统,就必须开发相应的设备驱动程序,而这些板卡设备绝大多数又都属于字符设备类。因此,本文针对这种情况,详细讨论了在 VxWorks 操作系统下字符设备的驱动程序开发问题。

1 相关概念

为了让读者对驱动程序开发有一个全面的了解,首先介绍一下设备驱动程序的相关概念。

设备驱动程序是指直接控制设备操作的那部分程序。通俗地说,就是对设备的 I/O 端口地址进行读写操作的那类程序。

设备驱动程序按设备类型的不同可划分为字符设备、块设备和网络设备 3 大类,这其中的字符设备是指能够像字节流一样被访问的设备,你可以简单地把它当成一个文件来访问,绝大多数板卡设备都属于字符设备。

设备驱动程序通常应具有 6 个主要功能:设备初始化、打开设备、关闭设备、读设备、写设备以及对设备进行控制。

2 调用方式

设备驱动程序本身不能主动执行,它只能被操作系统或用户程序调用。嵌入式系统调用设备驱动程序的方式有 3 种:即应用程序直接调用、应用程序通过操作系统内核调用以及应用程序通过操作系统的扩展模块来调用。这里,我们以最符合开发规范的第 2 种方式来讲解,因为据此开发的应用程序具有良好的可移植性。

在这种方式下,应用程序访问板卡设备是通过 VxWorks 的 I/O 子系统进行的。I/O 子系统提供了一套标准的与设备无关的 I/O 接口函数,对于字符设备来说,一共有 7 个函数:creat(),remove(),open(),close(),read(),write(),ioctl()。它们之间的关系见图 1。

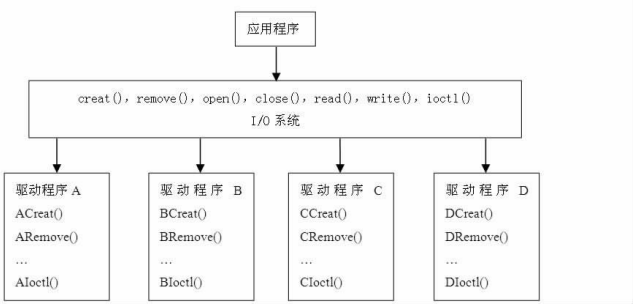

图 1　调用层次图

3 应用程序和设备驱动程序之间的联系

VxWorks 的 I/O 子系统还提供了 3 张表:设备列表、文件描述符表和设备驱动程序描述表,正是通过它们实现了应用程序和设备驱动程序之间的联系。

设备列表是一个大小可变的动态双向链表,每添加一个新设备时,链表就增加一个环节。例如:添加两个相同的设备,虽然这两个设备的驱动程序相同,但链表仍旧会增加个两环节而不是一个。设备列表中的每一个节点被称为设备描述符,它的结构组成如下:

Typedef struct xxDEV

```
{
DEV_HDR devHdr;// 设备头
BOOL ReadyToRead;// 设备读就绪
BOOL ReadyToWrite;// 设备写就绪
…
SEL_WAKEUP_LIST selwakeuplist;//select 功能
}xxDEV;
```

此结构中的设备头部分是必需的,斜体部分是用户根据需要自己添加的。设备头结构的原型如下:

```
Typedef struct DEV_HDR
{
DL_NODE node;// 设备列表的连接节点
short drvNum;// 设备的驱动程序索引号
char * name;// 设备名称
}DEV_HDR;
```

设备驱动程序描述表用于存储驱动程序的各个函数的入口地址,其结构如图 2 所示。

驱动程序索引号	creat	remove	open	close	read	write	ioctl
0	…	…	…	…	…	…	…
1	…	…	…	…	…	…	…
2	xxDevCreat	NULL	xxOpen	xxClose	xxRead	xxWrite	xxIoctl
3	xxxOpen	NULL	xxxOpen	xxxClose	xxxRead	NULL	xxxIoctl

图 2　设备驱动程序描述表

通过图 2 可知,并非所有的 I/O 接口函数都要实现,甚至有些接口函数的实现是完全相同的。

文件描述符表的作用是将板卡设备与设备驱动程序关联起来。其结构如图 3 所示。

文件描述符	驱动程序索引号	驱动程序分配的设备号
879022	3	0
239703	7	0
256987	7	1

图 3　文件描述符表

其中文件描述符就是 xxDevCreat()或 xxOpen()的返回值。

驱动程序索引号就是驱动程序安装函数 iosDrvInstall()的返回值。

字符设备的设备驱动程序通常由下面几部分组成:设备驱动程序的注册函数(xxDrv)、设备创建函数(xxDevCreat)、设备卸载函数(xxDevRemove)、设备打开和关闭函数(xxOpen、xxClose)、设备读/写函数(xxRead、xxWrite)、设备控制函数(xxIoctl)和设备的中断服务函数(xxIntHdlr)。

在使用上述驱动函数之前,应先在设备驱动程序注册函数中的通过 iosDrvInstall()函数将它们挂接到系统中,iosDrvInstall()函数会将驱动程序的各个函数的入口地址填入到设备驱动程序描述表中;再由设备创建函数中的 iosDevAdd()函数将设备加入到设备列表中。

之后,当应用程序象下面这样打开一个设备时:

```
fd=xxOpen (xxDev,devName,mode);// xxDev 是指向设备描述符的指针;devName 是指向设备名的指针;mode 用于指示设备打开的方式;fd 是返回的文件描述符。
```

I/O 子系统就开始在设备列表中搜索与 devName 最匹配的设备;找到后,再根据设备列表结构的 drvNum 字段值在设备驱动程序描述表中查找驱动程序的各个函数的入口地址;最后 I/O 子系统会把文件描述符 fd,还有与 fd 对应的驱动程序索引号以及由驱动程序分配的设备号添加到文件描述符表以供应用程序的后续访问。至此,应用程序就与设备驱动程序联系起来了。

4 中断服务函数的注意事项

现在的板卡设备许多都提供有对中断的支持,因此设计设备的中断服务函数也往往是必需的一环。由于中断服务函数没有任务控制块,不在一个固定的任务上下文中执行,故所有的中断服务函数是共享一个堆栈的。这使得设计中断服务函数会有以下一些限制:

1)不可以有 I/O 系统调用。(如 printf())

2)不可以有阻塞操作。(如 semTake(),free(),taskDelay(),malloc())

3) 不可以有请求任务上下文的操作。(如 taskSuspend())

4)不可以有浮点协处理器函数的调用。

另外,如果板卡设备支持中断复用,则中断服务函数应判断相关板卡设备是否产生了中断;中断服务函数还应该尽可能的减少无关操作。

5 如何处理 PCI 设备

对于 PCI 总线的板卡设备,其设备驱动程序只是多了两个需要注意的方面。即 PCI 总线设备初始化和 PCI 设备中断复用。

根据 PCI 规范,每一个 PCI 设备都有一个 256 字节的 PCI 配置空间,它的前 64 个字节(头标区)的信息对每一个 PCI 设备都是大致相同的,后 192 个字节的信息则属于该设备的专有信息,很多的 PCI 设备并不用这一块。 (下转第 126 页)

2.5 打印增压曲线程序的框图如图 5

把文件中的变量保存在数组变量中,根据这些变量绘制的曲线和保压时的曲线完全相同。为了提高曲线的打印质量,不采用图形控件中的打印功能,而是先传送正文和图形给 Printer 对象,再用 NewPage 和 EndDoc 方法打印。

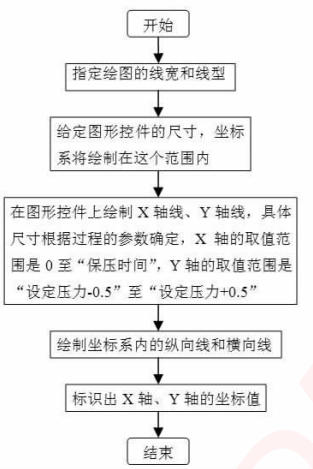

图 4

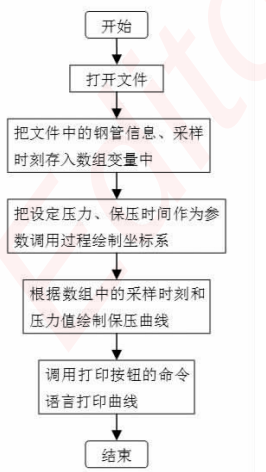

图 5

3 结语

该程序由三部分组成,曲线在屏幕上的绘制;曲线数据在文件中的保存;曲线从打印机的输出。绘制曲线时主要通过图形控件和计数器控件的功能来完成;保存曲线可以通过新建、打开、读出、写入、关闭等文件功能来实现;打印曲线使用 Printer 对象来完成,Printer 对象是一个与设备无关的图片空间,支持用 Print、PSet、Line、PaintPicture 和 Circle 等方法来创建文本和图形,当完成在 Printer 对象中放置信息后,用 NewPage 和 EndDoc 方法将输出传送到打印机,就可以完成曲线的打印。限于篇幅不能写出具体的程序代码,只能用程序框图来表示。该文介绍的方法已经在水压机水压监控系统程序中得到应用,已经在产品中正常使用。

参考文献:

[1] 程胜利.Visual Basic 语言程序设计教程[M].北京:中国水利水电出版社,2007.
[2] Coombs T,Campbell J.Visual Basic 编程实用大全(精华版)[M].邓少鹃,邓云佳.译.北京:中国水利水电出版社,2005.

(上接第 122 页)

在 PCI 配置空间头标区里,从 0Dh—24h 是 24 个字节的基地址寄存器区,这些寄存器用于确定 PCI 设备需要多少存储器空间和 IO 空间。故在 PCI 总线设备初始化时,要先确定空间类别和容量需求。按照 PCI 规范,通过读取各个基地址寄存器,由返回值的第 0 位来判断需要的空间类别,1 表示是 IO 空间,0 表示是存储器空间。对于存储器空间,则向该基地址寄存器写入 0xFFFFFFFF,再读回其值,用该值与 0xFFFFFFF0 进行“位与”运算,再把结果取反就可以得到需要的存储器空间的大小了。最后用 sysMmuMapAdd()函数将需要的存储器空间映射到系统的物理地址空间中去。

对于 PCI 设备中断复用的问题,VxWorks 专门提供了 pciIntConnect()函数,它可以为我们将中断服务函数挂接到系统中,我们只要在中断服务函数中判断一下此中断是否是自己的板卡设备产生的就可以了。

6 结束语

上文详细地介绍了 VxWorks 下字符设备驱动程序的组成、调用机制以及 PCI 驱动程序开发的具体事项。文中提到的函数在 VxWorks 的联机帮助中都有详细介绍,这里不再赘述。依照上述思路开发的某板卡设备的驱动程序已经在某测试系统中得到了成功应用。

参考文献:

[1] Wind river.tornado device driver workshop[M].2002.
[2] Wind river.VxWorks 5.5 Programmers Guide[M].2002.
[3] 李贵山,戚德虎.PCI 局部总线开发者指南[M].西安:西安电子科技大学出版社,2001.