第 33 卷 第 8 期 计 算 机 工 程 2007 年 4 月
Vol.33 No.8 Computer Engineering April 2007
·开发研究与设计技术· 文章编号：1000—3428(2007)08—0270—03 文献标识码：A 中图分类号：TP316

基于 Linux 的 MPEG-4 视频压缩卡驱动程序

程晓宇，毕笃彦
(空军工程大学工程学院信号与信息处理实验室，西安 710038)

摘 要：阐述了 Linux 系统下实现设备驱动程序的基本框架。在介绍 MPEG-4 压缩卡工作原理的基础上，讨论了 Linux 系统下 PCI 接口的 MPEG-4 视频压缩卡驱动程序的设计与实现过程，并提供了相应的函数及程序的编写。实验结果证明了该驱动程序的实用性。
关键词：Linux；驱动程序；中断；MPEG-4

MPEG-4 Video Compress Card Driver on Linux

CHENG Xiaoyu, BI Duyan
(Lab of Signal and Information Processing, Engineering Institute, Air Force Engineering University, Xi'an 710038)

【Abstract】Based on the basic principle of MPEG-4 video compress card, this paper presents the general framework of developing device driver on Linux platform and describes the design and the implementation of the MPEG-4 video compress card driver, including relevant functions and programs. The experiment results prove the feasibility of the driver.
【Key words】Linux; Driver; Interrupt; MPEG-4

Linux 是 Unix 操作系统的一个变种，由于其源代码完全公开，且具有较强的稳定性、可裁剪性及强大的网络功能，越来越多的开发人员开始采用 Linux 平台来开发自己的产品。开发产品过程中，编写设备驱动程序的工作量要占整个系统工作量的 1/3 甚至更多，而且设备驱动程序编写的好坏直接影响到硬件设备性能的发挥，在整个系统的开发中占有极为重要的地位。

MPEG-4 是新一代的音视频压缩标准，能够实现基于内容的交互并具有很高的压缩效率，在视频监控和网络视频等领域有着广泛的应用。本文主要介绍 Linux 系统下 PCI 接口的 MPEG-4 视频压缩卡驱动程序的设计及其实现细节。

1 Linux 系统设备驱动程序概述

在多任务操作系统中，系统通过设备驱动程序完成对特定硬件的控制。设备驱动程序实际是处理或操作硬件控制器的软件[1]，从本质上讲，它们是内核中拥有高特权的，驻留内存的，可共享的底层硬件处理例程。

Linux 通过设备驱动程序为应用程序提供了统一抽象的接口，从而隐藏了大量不同设备之间的区别和细节。在 Linux 中所有对硬件设备的操作和普通文件一样，利用标准的系统调用在设备上进行打开、关闭、读取和写入操作。系统中每个设备由“设备特殊文件”来代表。每个由相同设备驱动程序控制的设备具有相同的主设备号，而次设备号则用来区分同类设备中的不同设备。设备特殊文件的虚拟文件系统(Virtual File System, VFS)索引节点中包含设备号信息。如果通过系统调用访问设备，则内核可通过该 VFS 索引节点中的设备号信息调用适当的设备驱动程序。

2 MPEG-4 视频压缩卡的工作原理

压缩卡中，MPEG-4 压缩编码芯片采用INTIME公司的IME6400 芯片。IME6400 是一块多通道实时数字音视频MPEG-4/2/1 压缩芯片[2]，它不仅支持原始的音、视频信息，还支持PCM编码的音频信号等。IME6400 是按照内部的firmware工作的[3]。在实现上，firmware软件既可以存放在一个外挂的ROM中，也可以从外部主机通过IME6400 的Host Interface(主机接口)下载到芯片中。本系统中采用外挂ROM的方法以便选择firmware的版本并定期进行软件更新。除上述功能外，IME6400 还可以根据主机自定义的运动检测要求实现运动检测，并将检测到的信息以数据包的形式发送给主机。IME6400 的原理框图如图 1 所示[2]。模拟视频、音频信号分别经A/D转换后，其数据流输入到IME6400 中进行MPEG-4压缩编码，编码后的数据流经SDRAM缓存最后通过主机接口送到外部主机。

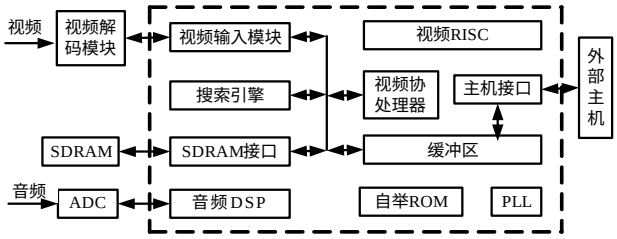

图 1 IME6400 的原理框图

压缩卡的PCI总线控制模块采用的芯片是PHILIPS公司的SAA7146A PCI多媒体桥[4]，它完全符合PCI 2.1 总线规范，集成了MMU、BPS和HPS等单元模块。它还包含了数据扩展总线接口DEBI和DMSD2 兼容(16 位YUV)的视频输入接口等众多接口，可以和包括IME6400 在内的众多音、视频处理芯片实现无缝连接，所以较为广泛地应用于多媒体数据的传输和处理。

本压缩卡中，模拟视频信号经解码和PLD逻辑控制后变为RGB信号，连同音频解码模块的输出一起分别送入IME6400的视、音频接口，由IME6400 完成视频和音频信号的MPEG-4

作者简介：程晓宇(1979－)，男，博士生，主研方向：图像侦查与传输，Linux 操作系统；毕笃彦，教授、博导
收稿日期：2006-05-08 **E-mail：**ubwom@163.com

实时压缩编码。IME6400 利用主机接口和PCI总线控制模块SAA7146A相连，构成数据和控制通道，同时用外挂SDRAM来存储待编码的码流，而IME6400 内部 1KB的FIFO用来实现已编码码流的快速传输。外部主机可以按照规定的流程对主机接口控制寄存器进行读或写操作，还可以完成直接寄存器读写、IME6400 系统内存的访问、FIFO缓冲区的访问和firmware软件的下载等操作。压缩卡的原理框图如图 2 所示。

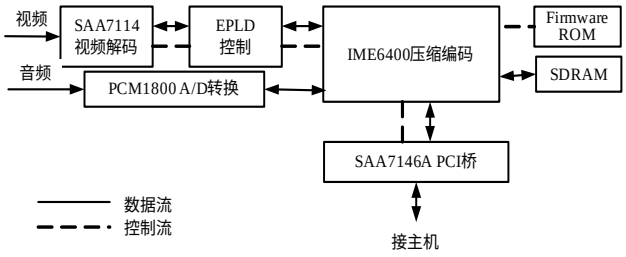

图 2 压缩卡原理框图

3 基于 Linux 2.4 内核的驱动程序设计与实现

下面详细介绍基于 Linux2.4 内核下 PCI 接口的 MPEG-4 视频采集压缩卡驱动程序的设计与实现。内容包括视频压缩卡的初始化、驱动程序各入口函数的编写、中断处理及驱动程序的装/卸载和应用程序的编写等。

3.1 压缩卡初始化

根据 SAA7146A 的厂商号 1131H 和设备号 7146H，利用内核提供的 pci_find_device 函数来扫描 PCI 总线，找出 PCI 槽上共有多少块压缩卡。每找到一块即读取卡上的中断线、I/O 端口的基地址 iobase 及每块卡的总线号、设备号和功能号。根据卡的总线号和功能号为这块卡定义一个识别号，以便将来的程序识别此卡，最后将这些信息保存到结构体 m4c_obj 的变量数组 pcidev 中。pcidev[i]是一个结构体变量数组元素，对应每张压缩卡的上述信息。

接下来使用系统提供的register_chrdev()函数注册字符设备 m4cap 模块，参数包括主设备号、设备名称以及指向 file_operation 结构的指针：

```
if((major = register_chrdev(0, DEVICE_NAME, &device_fops)) < 0) {
printk(DEVICE_NAME ": Device registration failed (%d)\n", major);
}
```

若申请失败则应从系统中卸载此模块，并释放相应的主设备号及已申请成功的资源。注册成功后马上为卡申请空间以保存所读取的卡上的各种信息，最后注册中断。注册中断时首先根据已找到的设备读取系统分配的中断号，然后根据此中断号和设备名称注册相应的中断服务程序，本驱动程序中使用共享的中断号：

```
pci_read_config_byte(pdev->dev, PCI_INTERRUPT_LINE, &pdev->irq);
result = request_irq(pdev->irq, m4c_irq_handler, SA_SHIRQ | SA_INTERRUPT, "m4cap", pdev);
if (result) { /*register failed*/
        pdev->irq = -1;
        printk("request_irq failed!\n");}
pdev->task.sync = 0;
pdev->task.routine = m4c_bh_interrupt;
pdev->task.data = NULL;
```

主机对SAA7114 的初始化是通过IME6400 来实现的，IME6400 相当于连接SAA7146A和SAA7114 的桥，IME6400 和SAA7114 之间是通过I^2C总线实现通信的。

3.2 入口函数的编写

Linux 系统中上层的应用程序不能直接对硬件进行操作，需要通过驱动程序的 IOCTL 方法来实现。作为 IME6400 驱动程序的入口，file_operation 结构的各成员函数具体如下：

```
struct file_operations device_fops = {
open:    m4c_open,
release: m4c_release,
read:    m4c_read,
write: m4c_write,
ioctl:   m4c_ioctl}
```

其中：m4c_read 和 m4c_write 的实现为空操作，仅用来填充结构体 device_fops。m4c_open、m4c_release 分别完成设备计数的+1 和-1。

驱动程序中最主要的硬件控制集中在 IOCTL 系统调用上，应用程序就是通过这些 IOCTL 系统调用和驱动程序进行通讯地，下面详细介绍各 IOCTL()调用的实现。

(1)IOCTL_M4C_WR_REG: 对板卡上 SAA7146A 和 IME6400 等芯片的寄存器进行写操作；

(2)IOCTL_M4C_RD_REG: 根据应用程序的指令对板卡上 SAA7146A 和 IME6400 等芯片的寄存器进行读操作；

(3)IOCTL_7114_INIT: 对SAA7114 进行初始化，具体的初始化工作由驱动程序通过I^2C总线的读写来完成；

(4)IOCTL_M4C_DEBI_TRANSFER: 根据应用程序传入的参数来决定数据扩展总线接口数据的传输方向是读还是写，并完成相应的读写操作，程序如下：

```
case IOCTL_M4C_DEBI_TRANSFER:
io = (IO_PARA *)arg;
BYTE* ptempio = (BYTE *)io->offset;
/*从用户空间中传递应用程序的参数*/
BYTE direction = *ptempio;
WORD debi_address = *(WORD *)(ptempio + 1);
WORD debi_data = *(WORD *)(ptempio + 3);
if(direction == 1)
  m4c_debi_read(debi_address, (WORD *)(ptempio + 3));
/*direction 为 1 表明是读*/
else
  m4c_debi_write(debi_address, debi_data);
/*direction 为 0 表明是写*/
return 0;
```

(5)IOCTL_M4C_CAP_START: 调用 m4c_cap_start()函数来捕获视频，m4c_cap_start()函数主要完成 firmware 命令寄存器(0x2000)的配置和内核态下内存资源的申请和清零工作；

(6)IOCTL_M4C_CAP_STOP: IOCTL_M4C_CAP_START 相对应，对 firmware 命令寄存器(0x2000)下达停止压缩的命令并释放已申请成功的内存资源；

(7)IOCTL_M4C_GET_POS_BLOCKED: 从内核空间向用户空间传递已拷贝数据缓冲区的数目。

3.3 中断处理

IME6400 的内部有一个1KB的FIFO，当这个FIFO满时，它会置标志位同时等待外部主机及时将数据取走，否则出现超时，压缩将被临时挂起直到缓冲区的数据被读取。

为了提高数据传输效率和避免硬件的死锁问题，驱动程序中数据缓存采用了乒乓结构[5]，开辟了 2 块 100KB的内存空间作为缓存，每次中断时将IME6400 内部FIFO 1KB的数据

读出放到一块缓存中，当一块缓存写满时，即进行标识，然后向另一块未标识的内存写入数据，同时唤醒在此等待数据传输的进程队列，及时将数据复制到用户空间的缓冲区中，内存中数据被拷贝后对标识复位。如此循环，利用 2 块缓存来完成压缩码流从内核空间向用户空间的传输，避免了由于系统效率问题造成的缓存中数据覆盖现象，保证了记录数据的正确性和完整性。

对硬件中断的处理是驱动程序中最难的部分，如果处理不好会造成资源竞争、程序死锁、数据丢失等事件发生。本驱动程序中，中断处理函数仅完成检测数据可用后置标志位并将中断底半部[1]插入任务队列，其余功能在更安全时间内执行的中断底半部和任务队列中完成，包括将IME6400 内部 FIFO 1KB的数据读出放到缓存中，协调驱动程序中所开的 2 块缓存的使用等。下面是IME6400 的中断服务程序：

```
static void m4c_irq_handler(int irq, void *m4c_id, struct pt_regs *regs){
  m4c_obj *pdev = (m4c_obj *)m4c_id;
  if (m4c_read_reg(ISR) == 0){
    printk("No saa7146 interrupt!\n");
    return; }
  else{data_come = 1; /*置标志位*/
    queue_task(&pdev->task, &tq_immediate); /*插入立即队列*/
    mark_bh(IMMEDIATE_BH); /*标记底半部*/
    return;}
  }
```

3.4 驱动程序的装、卸载和应用程序编写

为了让驱动有更广泛的兼容性，本驱动在分配主设备号时使用主机随机分配。装载时通过一个脚本来获取随机的主设备号从而正确建立设备文件，部分程序如下：

(1)调用 insmod 来装载驱动

```
/sbin/insmod -f ./$module.o $* || exit 1
```

(2)用 awk 工具获取主设备号

```
major=`cat /proc/devices | awk "\\$2==\"$module\" {print \\$1}"
```

(3)先删除设备文件再创建设备节点

```
rm -f /dev/$device
mknod /dev/$device c $major 0
```

(4)设置设备访问权限

```
chgrp $group /dev/$device
chmod $mode /dev/$device
```

为了调试方便，在调试阶段将驱动做成动态加载模块以便于随时加载和卸载。驱动调试成功后通过在 /etc/rc.d/rc.local 文件中设置 m4cap_load(上面的脚本文件名)的运行实现驱动程序的自动加载。驱动程序卸载时直接使用 rmmod 命令将设备驱动模块从系统中移除即可。

应用程序 test_m4cap 主要完成设备的打开、初始化及视频数据的记录工作。其中最主要的工作就是通过调用驱动程序接口对记录视频参数进行设置，然后循环读取缓存中的视频数据，并将它们存入记录载体中。

4 结论

本文所介绍的驱动程序在 Linux 2.4.18 内核上调试成功。驱动装载后启动应用程序 test_m4cap 进行长时间记录，记录文件回放时内容完整、图像清晰、无马赛克现象，符合压缩的性能要求。本压缩卡和驱动程序拟应用于某型飞机的机载视频记录系统。

参考文献

1 Rubin A, Corbet J. Linux 设备驱动程序[M]. 魏永明, 译. 北京: 中国电力出版社, 2002.

2 INTiME Corporation. IME6400 MPEG4/2/1 Encoder Hardware Reference Manual[Z]. 2002. http://www.intime.co.kr.

3 INTiME Corporation. IME6400 MPEG4/2/1 Encoder Firmware Reference Manual[Z]. 2003. http://www.intime.co.kr.

4 Semiconductors P. SAA7146A Multimedia Bridge, High Performance Scaler and PCI Circuit(SPCI)[Z]. 1998. http://www.philips.com.

5 张立冬, 程晓宇. 某型机载数字音视频记录系统的软件设计与开发[J]. 计算机工程, 2005, 31(6): 213-215.

(上接第 269 页)

supplier: SUPPLIER endstatename { Insert_supplier (StateNode,$2); };

表示某个状态在事件 even 触发下的结果状态 supplier 保存到 StateNode 的 EndState 分量中。

4 UML 语法分析器的仿真

把上述生成的 4 个例程加入 VC6.0 环境进行仿真，结果如图 3 所示。

图 3 图 2 状态图的仿真结果

5 结论

通过 UML 语法分析器，可以自动提取*.mdl 文档的信息，利用提取的信息结合已成熟的测试算法就可以自动生成测试集，实现了从测试分析过程到生成测试集的自动化，提高了软件测试的效率。

参考文献

1 Object Management Group. UML Specification[EB/OL]. 2003. http://www.omg.org/uml.

2 陈火旺, 刘春林, 谭庆平, 等. 程序设计语言编译原理[M]. 北京: 国防工业出版社, 2000.

3 张保卫, 张毅坤, 赵 明, 等. 基于 UML 的面向对象软件测试系统[J]. 计算机工程, 2005, 31(6): 70-72.

4 Donnelly C, Stallman R. Bison[EB/OL]. 2004. http://www.gnu.org/software/bison/manual/pdf/bison.pdf.

5 Vern Paxson. Flex[EB/OL]. 1998. http://www.gnu.org/software/flex/manual/html_mono/flex.html.