文章编号：1005-8451（2010）07-0030-05

基于V4L的视频采集系统的设计

王元伟，刘国秀
（西南交通大学 信息科学与技术学院，成都 610031）

摘 要：介绍基于V4L的视频采集系统的设计。简要介绍所用的视频采集卡，着重介绍视频采集卡的驱动，基于Qt开发平台的采集程序和显示程序的设计。

关键词：V4L；视频采集；PCI；Qt
中图分类号：TP391 文献标识码：A

Design of Video Capture System based on V4L

WANG Yuan-wei, LIU Guo-xiu
(School of Information Science and Technology, Southwest Jiaotong University, Chengdu 610031, China)

Abstract: It was introduced the design of Video Capture System based on V4L. First of all, introduced the video capture card that was used, then highlighted the design of video capture card driver, program designing of capturing and displaying. Program designing of capturing and displaying was based on Qt.

Key words: V4L; video capture; PCI; Qt

Linux在TV和多媒体上的应用是目前热门的研究领域，而其中关键的技术则是Linux的Video-4Linux（V4L），即Linux内核里支持影像设备的一组标准接口函数（API）。配合适当的驱动程序，可以实现视频采集和影像CODEC等功能。V4L可为视频设备提供编程接口，使视频采集的应用更加广泛。

1 系统硬件

本系统硬件采用Sony EX view CCD摄像头和DC600 1394视频采集卡。摄像头采集原始模拟数据，视频采集卡的Decoder芯片接收摄像头的数据对数据进行解码，将模拟信号转换为数字信号，数据流通过PCI接口芯片，由PCI接口芯片可以通过DMA传送到内存，然后再保存到硬盘上，或者送到显卡的缓冲区进行预览。整个系统的中心器件是控制器。其工作流程如图1。

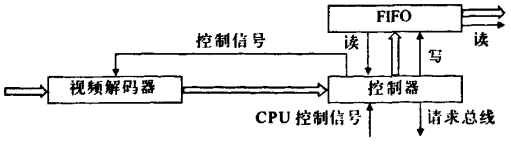

图1 视频采集卡工作流程

收稿日期：2009-12-16

作者简介：王元伟，在读硕士研究生；刘国秀，在读硕士研究生。

本系统所使用的摄像头采集支持NTSC/PAL视频制式。视频采集卡使用可编程数字图像解码芯片SAA7111，控制器负责图像数据采集控制、FIFO读写控制、与PCI接口芯片通信。

2 驱动程序设计

本系统硬件采用PCI接口，其驱动程序设计可分为两部分：PCI接口驱动和采集卡设备芯片初始化。

2.1 PCI接口驱动设计

在写驱动程序前，为了使驱动能正常运行，应先对PCI设备的配置寄存器进行设置。可通过查看/proc/bus/pci/devices和/proc/bus/pci/*/*来查看PCI设备清单和配置寄存器。所有的PCI设备有至少256 bit的地址空间。前64 byte是标准化的，其余是设备相关的。可以查看用户手册进行配置，驱动程序通过配置好的寄存器与设备进行交互。

Linux对于视频采集提供了两种驱动和应用程序编程规范：V4L和V4L2。本系统驱动和上层应用的开发都是按照V4L的规范编写的。PCI卡的设备驱动程序结构上大体相似，由以下几部分组成。

2.1.1 探测PCI设备

探测设备就是使驱动程序找到要驱动的设备，主要有两种PCI设备探测方法：老式PCI探

测和新式探测方法。二者的区别是：老式的探测方法需要手工搜索系统中的PCI设备列表或调用一个可以查找特定PCI设备的函数，新式的探测方法把探测设备任务交给系统的PCI层自动完成，这样可以支持热插拔，因此非常方便。这里用到新的探测方法。

在驱动程序中通过调用一个系统函数pci_register_driver进行设备的自动探测，其函数原型如下：int pci_register_driver (struct pci_driver *)，最重要的工作是添写struct pci_driver的成员，与驱动程序相关的字段如下：

name：驱动程序的名称；

id_table：指向驱动程序感兴趣的设备的列表；

probe：指向PCI驱动程序中探测函数的指针。当PCI核心有一个它认为驱动程序需要控制的PCI设备室便调用该函数；

remove：指向一个移除函数的指针，当移除驱动程序所支持的PCI设备或驱动程序从内核卸载时，PCI核心便调用该函数；

suspend：指向一个挂起函数的指针，设备被挂起时调用的函数；

resume：指向一个恢复函数的指针，当PCI设备恢复时调用的函数。

2.1.2　激活PCI设备

激活设备就是使设备处于工作状态，在驱动程序可以访问PCI设备的任何设备资源前，必须激活PCI设备。驱动程序通过调用pci_enable_device函数激活设备，其原型如下：

int pci_enable_device(sturct pci_dev *dev)。

2.1.3　申请和访问资源

系统资源一般包括I/O空间和内存空间，使用之前必须显式地进行申请，申请实际上是在系统中进行登记，避免出现资源出现混乱局面。对于中断设备来说还要申请中断号，需要DMA传送的设备还应申请DMA缓冲区，在本系统中，需要同时申请终端号和DMA缓冲区。

Linux内核中申请内存的方法有3个：kmalloc、__get_free_pages和vmalloc。前两个系统调用申请连续的物理地址空间，vmalloc所申请的物理空间不连续，而kmalloc只能分配小于128 Kbit的内存，__get_free_pages最大可申请到4 Mbit的内存，所以应采用__get_free_pages来完成内存的申请。

DMA通道的申请由request_dma实现，PCI设备申请DMA缓冲区的函数是pci_alloc_consi-stent，这两个函数的原型如下：

int request_dma(unsigned int, const char*);

void *pci_alloc_consistent(struct pci_dev*, size_t, dma_addr_t)。

I/O端口通过request_region函数申请，该函数原型如下：

struct resource *request_region(unsigned long, unsigned long, const char*) 。

中断号的申请通过内核函数request_irq实现，由于系统只有16根中断信号线，所以在申请的时候要用共享的方式申请。

2.1.4　与设备进行通信

这部分是整个驱动程序的关键，具体操作需要参照设备硬件资料去做。与硬件通信通过读写设备的端口或者存储空间实现。对设备端口的访问通过intb、outb以及其他变形函数实现；通过readb和writeb等函数访问设备的存储空间。有时也需要调用rmb、wmb或mb函数防止编译器优化。

2.1.5　注册设备

由于视频采集卡属于字符设备，因此需要通过调用register_chrdev注册设备，建立设备存取的入口点，主要由register_chrdev中指向struct file_operations结构的指针参数确定，结构如下：

```
static struct file_operations my_video_fops=
{
        .owner = THIS_MODULE;
        .read = video_read();
        .write = video_write();
        .open = video_open();
        .mmap = video_mmap();
        .release = video_release();
        ...
    };
```

通过register_chrdev，使设备号与相应的file_operations结构建立一对一的映射，至此驱动程序完成。

2.1.6　加载驱动程序

在编写驱动程序的时候应以模块的形式编写，

每个模块都有一个入口和出口，通常设备的注册在入口函数中编写，加载模块的时候便调用入口函数。模块卸载的时候调用出口函数完成设备的注销工作。加载模块用insmod或modprobe函数，卸载模块用rmmod函数。

2.2 采集卡芯片初始化和控制

采集卡最核心的部分是PCI接口芯片，Decoder芯片的初始化和控制通过PCI接口芯片实现，因此对PCI接口芯片的初始化和控制至关重要，需要对其进行以下编程：与DMA控制器联系起来进行DMA传输；确定PCI中断触发事件；PCI中断处理程序。经过以上步骤，PCI视频采集卡即可工作。

3 视频采集和显示程序设计

3.1 视频采集程序的设计

根据V4L规范，视频采集流程如图2。

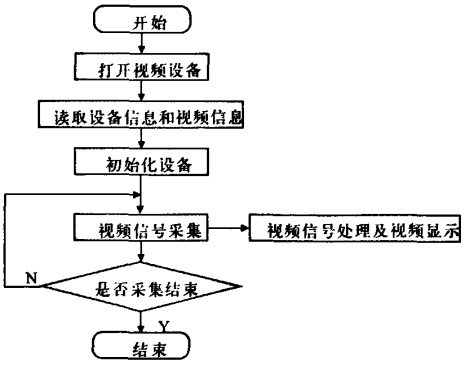

图2 视频采集流程图

根据Linux下的/include/linux/videodev.h文件中定义的数据结构，选择需要的机构，定义自己的视频结构，如下：

```
typedef struct v4l_device
{
  int fd;
  struct video_capability capability;
  struct video_channel channel[4];
  struct video_picture picture;
  struct video_window window;
  struct video_capture capture;
  struct video_buffer buffer;
  struct video_mmap mmap;
  struct video_mbuf mbuf;
  unsigned char *map;
  int frame;
  int framestat[2];
}vd;
```

各成员的具体定义以及下面用到的宏都在/include/linux/videodev.h文件中。

3.1.1 打开和关闭视频设备

在Linux中，采用设备文件，对用户来说，设备文件与普通文件没有差别。视频卡在Linux中属于字符设备，其主设备号是81，在实际操作上，访问控制视频卡也与一般的设备文件没有什么不同。用open打开设备：

```
vd->fd = open("/dev/video0",O_RDWR);
```

关闭视频设备用close（"/dev/video0"，vd->fd)。

3.1.2 读取设备信息和视频信息

下面几步都需要用到ioctl函数，ioctl的原型是int ioctl(int fd, ind cmd, …)，fd是设备的文件描述符，cmd是用户程序对设备的控制命令，省略号一般是一个表示类型长度的参数，可省略。它用于与设备进行对话，是一个非常重要的函数。

读取设备和视频信息的相关代码如下（宏VIDIOCGCAP和VIDIOCGPICT的分别表示获得视频设备的capability和picture)：

```
\\读取设备信息
  int v4l_get_capability(v4l_device *vd){
   if (ioctl(vd->fd, VIDIOCGCAP, &(vd->capability)) < 0) {
     perror("v4l_get_capability:");
     return -1;
   }
  return 0;
}
\\读取视频信息
int v4l_get_picture(v4l_device *vd)  {
  if (ioctl(vd->fd, VIDIOCGPICT, &(vd->picture)) < 0) {
    perror("v4l_get_picture:");
    return -1;
  }
```

```
    return 0;
  }
```

3.1.3 初始化设备

这一步主要设置video_picture以及video_mmap类变量的值，主要代码如下：

```
  vd->picture. palette = VIDEO_PALETTE_RGB32;
   if ( ioctl( fd, VIDIOCSPICT, &vd->picture ) < 0 )
  {
    perror("v4l_set_picture:");
    return -1;
  }
    vd->mmap.width = MAX_WIDTH;
    vd->mmap.height = MAX_HEIGHT;
    vd->mmap.format = vd->picture.palette;
    ioctl( fd, VIDIOCMCAPTURE, &vd->mmap ); // 在设置完成后使设置生效
```

3.1.4 视频信号采集

视频信号采集即获得图像，获得图像的方式有2种，分别是直接用read读取设备和使用mmap内存映射，直接读取设备方式通过内核缓冲区来读取数据；而mmap()通过把设备文件映射到内存中，绕过内核缓冲区，加速I/O访问。本程序采用mmap方法。另外，mmap( )系统调用使得进程之间通过映射同一个普通文件实现共享内存。普通文件被映射到进程地址空间后，进程可以像访问普通内存一样对文件进行访问，不必再调用read()和write（）等操作。mmap的代码如下：

```
int v4l_mmap_init(v4l_device *vd) {
  if (v4l_get_mbuf(vd) < 0)
      return -1;
  if ((vd->map = mmap(0, vd->mbuf.size, PROT_READ|PROT_WRITE,
    MAP_SHARED, vd->fd, 0)) < 0) {
    perror("v4l_mmap_init:mmap");
    return -1;
  }
  return 0;
}
```

mmap执行完以后，接下来便进行真正的图像采集，调用2次ioctl()函数，命令参数是VIDIOCMCAPTURE和VIDIOCSYNC。VIDIOCMCAPTURE参数告诉ioctl()，将图像数据采集到mmap()所映射的内存中。若调用成功，就开始一帧的截取，是否截取完一帧留给VIDIO-CSYNC来判断，若成功，表明一帧截取已完成。可以开始做下一次VIDIOCMCAPTURE。本系统采用连续帧采集，具体代码如下。

（1）获取图像函数

```
int v4l_grab_frame(v4l_device *vd, int frame)
{
  if (vd->frame_using[frame]) {
    fprintf(stderr, "v4l_grab_frame: frame %d is already used.\n", frame);
    return -1;
  }
  vd->mmap.frame = frame;
  if (ioctl(vd->fd, VIDIOCMCAPTURE, &(vd->mmap)) < 0) {
    perror("v4l_grab_frame");
    return -1;
  }
  vd->frame_using[frame] = TRUE;
  vd->frame_current = frame;
  return 0;
}
```

在本函数中是截取双帧图像。

（2）判断采集函数

```
int v4l_grab_sync(v4l_device *vd)
{
  if (ioctl(vd->fd, VIDIOCSYNC, &(vd->frame_current)) < 0)
    perror("v4l_grab_sync");
  vd->frame_using[vd->frame_current] = FALSE;
  return 0;
}
```

该函数返回0说明你想要获取的图像帧已经获取完毕。

至此，图像采集程序完成，下面介绍对采集到的视频信号的处理，使其在PC机上显示出来。

3.2 视频显示

本文采用Qt平台对视频进行显示，Qt中显示

文章编号：1005-8451（2010）07-0034-03

基于Qtopia Core的网络MP3播放器的设计与实现

曲 威，蒋朝根
（西南交通大学 信息科学与技术学院，成都 610031）

摘 要：介绍一种基于Qt/embedded Linux（Qtopia Core）的网络MP3播放器的设计方法，通过FTP，将网络上FTP服务站点中的MP3音乐文件下载到本地，使用文件流对MP3进行软解码，实现文件传输控制、播放和暂停、快进快退、上下曲、音量增减以及显示歌曲状态信息等功能。

关键词：Qt/embedded Linux；Qtopia；FTP网络；MP3播放器
中图分类号：TP393 文献标识码：A

Design and implementation of network MP3 player based on Qtopia Core

QU Wei, JIANG Chao-geng
(School of Information Science &Technology, Southwest Jiaotong University, Chengdu 610031, China)

Abstract: In this paper, it was introduced a design method of Network MP3 player based on Qtopia Core. By MP3, the music was downloaded from FTP server to local file system by FTP protocol. Using file stream to decode MP3 files, it was implementd these primary functions: files transports controllplay and pause, move back, skip forward, adjust volume of music, and song status showing.

Key words: Qt/embedded Linux; Qtopia; FTP network; MP3 Player

近年来，随着嵌入式产品的广泛应用，越来越多的嵌入式设备使用嵌入式Linux作为操作系统，利用Linux丰富的接口搭建复杂的图形用户界面。Qt/Embedded Linux（也称Qtopia Core）以其良好的可移植性逐渐成为一种被广泛使用的GUI系统。它运行在嵌入式Linux系统上，为嵌入式应用程序提供Qt的API标准。利用QT设计MP3图形用户界面，可以提高人机交互的友好性和美观性。

本文以Qtopia Core作为嵌入式Linux操作系统的GUI系统，以主要为freescale的coldfire平

收稿日期：2009-12-15
作者简介：曲 威，在读硕士研究生；蒋朝根，教授。

采集到的视频图像一般有两种方法：

（1）直接读取内存缓冲区中的帧，将其转化为QImage，再用QTimer与QPainter不断更新；

（2）将RGB（或YUV）数据流写入文件，生成Bitmap图像，存入磁盘，再载入QPixmap对象，然后用QPainter绘制。

第2种方法的最大缺点是速度太慢，保存文件和加载文件消耗大量时间，不能满足实时性的要求，因此选用第1种方法。

实现方法如下：重载paintEvent函数，在该函数中调用Qimage，将采集到的帧转化为QImage图像，再重载timerEvent函数，在该函数中调用repaint函数，从而在定时器到时的时候图像更新，由于选用的采集速率是25帧/s，因此Qtimer的初始值应小于40 ms才能得到稳定的图像，避免丢帧。

4 结束语

本文介绍了视频卡驱动和对视频信号进行采集和处理的过程，在实验室中进行了一路视频的采集和显示实验，视频画面清晰、无闪烁，将要做的工作是通过多路视频采集卡对多路视频信号进行采集和处理，以满足实时监控的需求。

参考文献：

[1] Jonathan Corbet, Greg Kroah-Hartman, Alessandro Rubini. Linux设备驱动程序[M].北京：中国电力出版社，2006.
[2] 宋宝华. Linux设备驱动开发详解[M].北京：人民邮电出版社，2008.
[3] Jasmin Blanchette, Mark Summerfield. C++ GUI Qt 4编程[M].北京：电子工业出版社，2009.