第 36 卷 第 9 期 计 算 机 工 程 2010 年 5 月
Vol.36 No.9 Computer Engineering May 2010
·开发研究与设计技术· 文章编号：1000—3428(2010)09—0282—03 文献标识码：A 中图分类号：TP301.6

嵌入式 Linux 下 USB 摄像头驱动实现

宋丽华，高 珂
(北方工业大学信息工程学院，北京 100144)

摘 要： 在嵌入式系统中开发 USB 摄像头驱动需要充分利用 USB 总线带宽并保证内存与摄像头之间数据的高速稳定交换。为满足该要求，参考开源项目 GSPCA/SPCA5xx，采纳 Linux 内核建议并遵循 Video4Linux 标准，提出双 URB 分配策略和 MMAP 内存映射机制以最大限度地提高视频采集速度。在 S3C2440AL_ARM 开发平台上设计并实现 USB 摄像头的嵌入式 Linux 设备驱动，对该驱动程序进行功能验证和性能分析。

关键词： 嵌入式 Linux；USB 摄像头；双 URB 策略；MMAP 映射机制

Implementation of USB Camera Drive Under Embedded Linux

SONG Li-hua, GAO Ke
(College of Information Engineering, North China University of Technology, Beijing 100144)

【Abstract】 In the design and development of USB camera drive based on embedded system, it should make full use of USB bus bandwidth and ensure a high and stable data transfer speed between memory and peripherals. In order to meet the real-time video capturing requirements, reference is made to open-source projects GSPCA/SPCA5xx, recommendation of Linux kernel is adopted, and using double-URB strategy and MMAP mapping mechanism to maximize the video capturing speed, eventually realizes this USB camera drive in line with Video4Linux on the S3C2440AL_ARM development platform and provides a functional verification and performance analysis.

【Key words】 embedded Linux; USB camera; double-URB strategy; MMAP mapping mechanism

1 概述

随着 CMOS 和 CCD 图像传感器技术的迅速发展，USB 摄像头由于性价比高、接口统一和支持多种高质量图像输出，因此应用十分广泛，如可视电话、视频会议、视频传感器和手持设备。在此类嵌入式设备中，基于 ARM 结构的高性能、低功耗、低成本的处理器已成为嵌入式解决方案的 RISC 标准。但嵌入式系统中支持 USB 摄像头的驱动很少，因此，研究和开发 USB 摄像头驱动具有较高的实用价值和现实意义。

USB 摄像头由传感器芯片和图像处理芯片组成。传感器芯片负责图像采集，图像处理芯片负责压缩及与主机的通信。本文使用的摄像头传感器芯片为 HYUNDAI 公司的 HV7131C CMOS 芯片，图像处理芯片为 Vimicro 公司的 ZC0301 系列芯片。开发平台为 EMBEDSKY 公司的 SKY2440 开发板(Samsung S3C2440AL ARM9 芯片，主频 400 MHz)，操作系统为 Linux 2.6.13，交叉编译环境为 gcc-3.4.1-glibc-2.3.3。

2 USB驱动程序系统架构

2.1 USB 驱动层结构

Linux 内核主要支持 2 种类型的 USB 驱动程序：主机侧(host)驱动和设备侧(gadget)驱动。主机侧驱动负责控制插入其中的 USB 设备，而设备侧驱动控制该设备如何与主机通信[1-2]。

USB 驱动层如图 1 所示，在主机侧中，最底层是 USB 主控制器驱动，中间层为 USB 核心层，最上层为 USB 设备驱动层。通常，内核本身带有前面 2 层驱动程序，而开发者只需完成 USB 设备驱动层的开发工作。

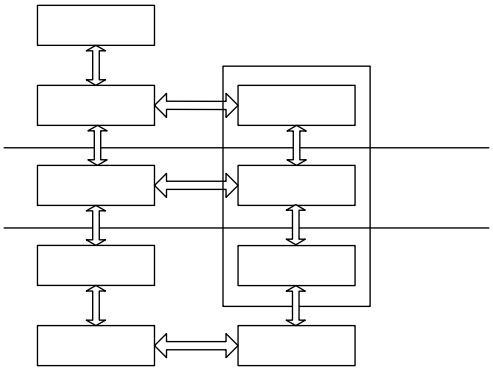

图 1 USB 驱动层结构

2.2 主机侧与设备侧

设备侧非常复杂，生产商按照 USB 协议进行实现，Linux 内核提供的 USB 核心处理了大部分的复杂性。主机侧驱动开发关心的是设备侧的构成，包括配置、接口和端点，以及 USB 主机侧驱动如何绑定到 USB 设备接口上。按照协议[3]，USB 是一种“单主方式”的实现，主机轮询各种不同的外围设备，

基金项目： 国家“十一五”科技支撑计划基金资助重点项目(2009BAI71B02)；北京市属高等学校人才强教计划基金资助项目(PHR201007121)；北方工业大学重点研究基金资助项目(NCUT20090106)

作者简介： 宋丽华(1979－)，女，博士，主研方向：嵌入式系统，网络通信协议；高 珂，硕士研究生

收稿日期： 2009-12-04 **E-mail：** slh2g@126.com

一个 USB 设备不能主动发送数据。

2.3 驱动模块结构

每个 USB 设备由一个 USB 模块驱动，但设备本身可能表现为一个字符设备，如 USB 摄像头。因此，对于一个 USB 摄像头来说，它首先是“USB”的，其次是“视频类”的。USB 模块与 Linux 内核用于支持视频设备的附加层 Video4Linux 一起工作。Video4Linux 驱动程序组划分出了一个通用模块，它导出的符号可供与具体硬件相关的驱动程序使用[4-5]。

3 关键模块的设计与实现

3.1 重要的数据结构

所有 USB 驱动程序都必须创建的结构体是 struct usb_driver。创建一个有效的 usb_driver 只要初始化以下 5 个字段：

```
static struct usb_driver spca5xx_driver = {
        .owner = THIS_MODULE,
        .name = "gspca",
        .id_table = device_table,
        .probe = spca5xx_probe,
        .disconnect = spca5xx_disconnect};
```

其中，id_table 是指向 struct usb_device_id 表的指针，该表包含了驱动程序可以支持的所有不同类型的 USB 设备。probe 是指向驱动程序中探测函数的指针，当 USB 设备在总线上加电启动后，USB 将检测与之匹配的驱动程序，如果找到，调用 spca5xx_probe()函数，该函数主要完成初始化工作，包括：配置设备信息以及连接摄像头操作函数；初始化 zc301 桥芯片和解码器信息；创建设备驱动文件系统等。

感光芯片和桥芯片的寄存器设置与具体硬件相关，初始化细节可参阅相关硬件的 datasheet。disconnect 是指向断开函数的指针，当 USB 设备从系统中移除或者驱动程序卸载时，将调用 spca5xx_disconnect()函数，主要完成内存释放和资源回收工作。struct usb_spca50x 是本驱动中非常重要的结构体，几乎所有的函数调用都会用到它。

```
struct usb_spca50x {
        struct video_device *vdev;
        struct usb_device *dev;
        struct tasklet_struct spca5xx_tasklet;
        struct dec_data maindecode;
        unsigned char iface;
        int alt;
        int epadr;
        int customid;
        struct spca50x_frame frame[2];
        struct spca50x_sbuf[2];
        … };
```

usb_spca50x 是 USB 摄像头驱动的“上下文”结构，包含了“当前”摄像头的硬件配置信息，包括：桥芯片和感光芯片信息，来自设备侧的“配置”、“接口”、“端点”信息，主要编解码信息，与摄像头相关的操作函数及帧缓冲信息。

硬件设备在 Linux 系统中是作为文件而存在的，struct file_operations 结构体实现了“设备文件”与在“设备文件上的操作”的连接。

```
static struct file_operations spca5xx_fops = {
  .owner = THIS_MODULE,
  .open = spca5xx_open,
  .release = spca5xx_close,
  .read = spca5xx_read,
  .mmap = spca5xx_mmap,
  .ioctl = spca5xx_ioctl,
  .llseek = no_llseek,};
```

在上述结构体中，spca5xx_open()函数是对设备文件执行的第一个操作，所有客户端程序对设备的访问都将从它开始。该函数主要完成初始化 usb_spca50x 中的各个缓冲区工作、寻找匹配的 iso 端点、开始同步传输等工作。spca5xx_close()用于释放 file 结构，注意并非每次关闭设备时都会被调用，只要 file 结构被共享，spca5xx_close()就会等到所有的副本都关闭之后才会得到调用[6]。spca5xx_read()用于从设备中读取数据。spca5xx_mmap()用于请求将设备内存映射到进程地址空间，此函数是 MMAP 内存映射机制的主要实现函数。spca5xx_ioctl()用于执行设备相关的特定命令。

3.2 双 USB 请求块及 MMAP 映射

Linux 内核通过 URBUSB 请求块(USB Request Block, URB)与所有的 USB 设备通信。该请求块使用 struct urb 结构体表示。根据 USB 协议，URB 的传输有 4 种：控制，中断，等时和批量。对于 USB 摄像头，USB 传输方式必须为等时传输[7]。驱动程序对 URB 的分配和提交策略及核态空间和用户空间转换开销是影响 USB 摄像头传输性能的 2 个关键因素[7-8]。本文采用的双 URB 策略和 MMAP 内存映射机制可以最大限度地提高视频采集速度。

3.2.1 双 URB 策略

驱动程序可以为单个端点分配多个 URB，也可以一个 URB 被多个端点重用。关于 URB 的分配策略，Linux 内核是这么建议的：当 URB 被驱动提交后，通常都在排队。对于音频或视频之类数据流设备，为了能以固定的速率传输，驱动程序在安排传输的 URB 时，至少应该是双缓冲的，并且在回调函数中应明确地重新提交该 URB。主要代码如下：

```
int gspca_init_transfert(struct usb_spca50x *spca50x)
{ …
  for (n = 0; n < 2; n++) {
  urb = usb_alloc_urb(FRAMES_PER_DESC, GFP_KERNEL));
  spca50x->sbuf[n].data=usb_buffer_alloc(spca50x->dev, psize *
FRAMES_PER_DESC,
GFP_KERNEL, &urb->transfer_dma);
  urb->pipe = usb_rcvisocpipe(spca50x->dev, ep->desc.bEndpoint
Address);
  urb->transfer_flags=URB_ISO_ASAP | URB_NO_ TRANSFER_
DMA_MAP;
  urb->interval = ep->desc.bInterval;
  urb->transfer_buffer = spca50x->sbuf[n].data;
  urb->complete = spca50x_isoc_irq;
  … }
  for (n = 0; n < 2; n++)
usb_submit_urb(spca50x->sbuf[n].urb, GFP_KERNEL);
}
```

首先调用 usb_alloc_urb()分配 2 个 URB，由于等时 URB 没有初始化函数，因此此处必须显示初始化。调用宏 usb_rcvisopipe()设置接收 URB 的设备端点地址；URB_ISO_ASAP|URB_NO_TRANSFER_DMA_MAP 用于指定 URB 传输方式为 iso 等时传输，并采用 DMA 方式以提高传输速度；transfer_buffer 指定 DMA 关联数据的缓冲区地址；最后调用 usb_submit_urb 分别提交这 2 个 URB。

spca50x_isoc_irq 为指向该 URB 生命周期结束后的回调函数地址，它主要完成一帧数据获取和解码工作，并完成重

新提交该 URB 的任务。主要代码如下：

```
void spca50x_isoc_irq(struct urb *, struct pt_regs *)
{if (spca50x->curframe >=0)
/* 解码并将数据传输到缓冲区中 */
        len = spca50x_move_data(spca50x,urb);
 else if (waitqueue_active(&spca50x->wq))
        wake_up_interruptible(&spca50x->wq);
 urb->dev = spca50x->dev;
 urb->status = 0;
 usb_submit_urb(urb, GFP_ATOMIC);
 … }
```

3.2.2 MMAP 内存映射机制

在 Linux 系统中，文件操作通常是由 read, write 系统调用完成。这些系统调用在驱动中的解决方法是用 copy_to_user(), copy_from_user()等函数在核态、用户态空间互相拷贝，如图 2 所示。但对大量的图像数据来说，这种频繁互拷会增加系统开销，特别对资源相对较少的嵌入式系统来说更加明显。

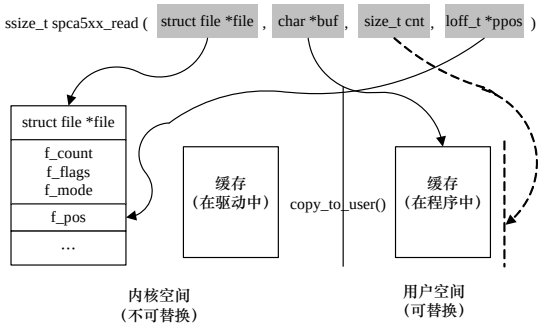

图 2　read 系统调用

MMAP 内存映射可以提供给用户直接访问设备内存的能力，以减少系统开销。主要代码如下：

```
ssize_t spca5xx_mmap(struct file *, struct vm_area_struct *)
{…
down_interruptible(&spca50x->lock);
page = kvirt_to_pa(pos);
if (remap_pfn_range(vma, start, page >> PAGE_SHIFT,
PAGE_SIZE, PAGE_SHARED)) {
up(&spca50x->lock);
return -EAGAIN;
}}
```

在 spca5xx_mmap()函数中，首先将驱动中帧缓存地址转换成页，然后调用 remap_page_range()函数将其逐页映射到用户空间中，如图 3 所示。

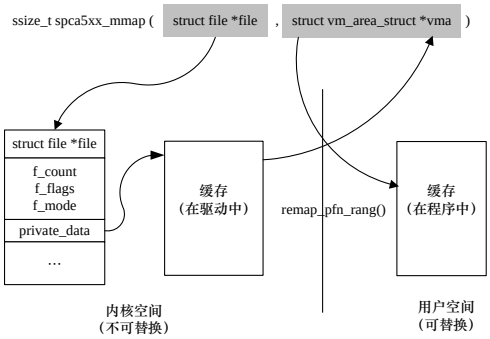

图 3　MMAP 映射机制

4　测试与性能分析

4.1　功能测试

嵌入式驱动测试平台见图 4，由 ARM9 CPU、64 MB Flash 存储器、3.5 英寸 TFT LCD 屏幕以及 2 个主机端 USB 接口组成，可以支持低速和全速的 USB 设备。

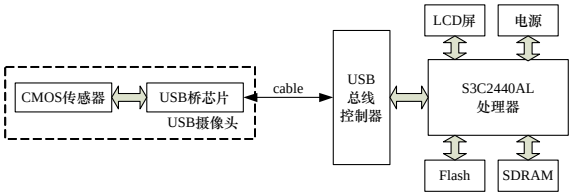

图 4　USB 摄像头驱动测试平台

驱动程序可以快速正确地完成 CMOS 传感器探测、桥芯片设置和初始化。运行客户端软件后，顺利出图，视频显示平滑流畅，可连续工作数十个小时，稳定可靠。

4.2　性能分析

搭建好 ARM-GCC 交叉编译环境[9]后，分别测试不同时间间隔下摄像头的帧速情况，测试参数和结果见表 1。测试平台为 400 MHz ARM9 处理器。由于嵌入式系统限制，因此在 320×240 分辨率下视频采集平均帧速大于 11 f/s，能够满足大多数嵌入式系统的需求。

表 1　帧速测试数据

测试时间/ms	解码帧数	帧速/(f·s^{-1})
8 398	113	13
15 509	145	10
20 577	237	11
27 769	328	11

5　结束语

本文介绍了 USB 驱动的层次结构以及主机侧和设备侧的关系，阐述了开发符合 Video4Linux 标准的 USB 摄像头驱动的方法。根据嵌入式系统的视频实时性要求，采用了双 URB 分配策略和 MMAP 内存映射机制，最大限度地提高了视频采集速度，实际测试表明其具有良好的性能和可靠性。这对其他平台的相关驱动开发工作也具有一定的指导意义。

参考文献

[1] 魏永明, 耿　岳, 钟书毅. Linux 设备驱动程序[M]. 3 版. 北京: 中国电力出版社, 2005.

[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] Compaq, Intel, Microsoft, NEC. Revision 1.1 Universal Serial Bus Specification[Z]. 1998.

[4] Schimek M H, Dirks B, Verkuil H, et al. Video for Linux Two API Specification Reversion 0.24[EB/OL]. (2009-09-03). http://v4l2spec.bytesex.org/v4l2spec/v4l2.pdf.

[5] Cox A. Video4Linux Programming[EB/OL]. (2003-11-29). http://kernelbook.sourceforge.net/videobook.html.

[6] Fliegl D. Programming Guide for Linux USB Device Drivers[Z]. (2000-03-01). http://usb.cs.tum.edu/download/usbdoc.

[7] 杨　伟, 刘　强, 顾　新. Linux 下 USB 设备驱动研究与开发[J]. 计算机工程, 2006, 32(19): 283-285.

[8] 刘　飞, 张曦煌. 基于嵌入式平台的 USB 摄像头驱动程序的实现[J]. 计算机工程与设计, 2008, 29(8): 1994-1996.

[9] Yaghmour K, Masters J, Ben-Yossef G, et al. Building Embedded Linux Systems[M]. 2nd ed. [S. 1.]: O'Reilly Media, Inc., 2008.

编辑　张正兴