基于 Framebuffer 的 DSP 平台 LCD 驱动程序的实现与应用①

Implementation and Application of DSP -LCD Driver Based on Framebuffer

封岸松 杨 柳 王庆辉 (沈阳化工学院 信息工程学院 辽宁 沈阳 110142)

摘 要: 论述了在 uClinux 中基于 Framebuffer 的 LCD 驱动程序的实现方法。首先主要介绍 Framebuffe 的处理机制以及底层驱动的接口函数，并在 ADSP-BF561 平台上进行了具体实现；其次在 Framebuffer 的基础上，简要介绍了嵌入式图形用户接口 MiniGUI，并在 ADSP-BF561 平台上进行了移植；最后通过实际效果测试，验证了工作的正确性。并将其应用于无线网络视频监控系统中的人机接口中。

关键词: ADSP-BF561 uClinux LCD Framebuffer MiniGUI

1 引言

近几年，嵌入式系统取得了迅猛的发展，在消费类电子产品和工业控制智能仪表等领域得到了广泛的应用。随着 Linux 技术的兴起，uClinux 在嵌入式领域逐渐占有了重要的地位[1]。uClinux 驱动程序的结构和标准 Linux 驱动程序的结构类似，不同的只是 uClinux 不支持内存管理单元[2]。因此，为了充分利用 uClinux 的优势，驱动程序必须按照 uClinux 的方式来写。由于嵌入式硬件性能的不断提升，使得在嵌入式设备上运行精美的图形用户界面成为可能。

MiniGUI 是一个著名的开放式源码的嵌入式图形界面软件，目的是把轻量级图形界面环境引入到运行 uClinux 的嵌入式小型设备上。而实现这一切的基础即为 Framebuffer。FrameBuffer 是 Linux 控制台下的一个通用的图形接口,它拥有良好的平台无关性,可以支持绝大多数的硬件，通过它可以十分方便地构建一个图形系统，因而得到愈来愈多厂家的支持。本设计中采用 Blackfin 系列处理器 ADSP-BF561 作为开发平台。无线网络视频监控系统中的人机接口，需要实现开发平台上 LCD 的 Framebuffer 驱动，才可以完成其图形界面。

2 底层LCD驱动机制

2.1 硬件平台

ADSP-BF561 是 AD 公司基于 DSP 为核心的一款开发平台，集成了多种外围设备，其中包括 TFT 真彩 LCD 带触摸屏及 LCD 控制器。LCD 控制器的功能是产生显示驱动信号，驱动 LCD 显示器。用户只需要通过读写一系列的寄存器，即可完成配置和显示控制。LCD 控制器通过 DMA 将显存中的数据传入显示板来显示图形。ADSP-BF561 中的 TFT 真彩 LCD 为 256 色，320×240 像素的 3.5 寸 LCD，LCD 控制器可支持单色/彩色 LCD 显示。配置 LCD 控制器重要的一步是指定显示缓冲区，显示的内容由缓冲区读出，其大小由屏幕分辨率和显示颜色数决定。

2.2 Framebuffer 的实现机制

在 uCLinux 中，Framebuffer 是一种能够提取图形的硬件设备，是用户图形界面的很好接口。Framebuffer 是显存抽象后的一种设备，它允许上层应用程序在图形模式下直接对显示缓冲区进行读写操

① 基金项目:沈阳市科技项目应用基础研究计划项目(1081236-1-00)
收稿时间:2008-09-26

作[2]。物理显存的位置、换页机制等等具体细节都由 Framebuffer 设备驱动来完成[2]。显示器根据相应指定内存块的数据来显示对应的图形界面，由 LCD 控制器和相应的驱动程序来完成。

Framebuffer 的显示缓冲区位于 uCLinux 中核心态地址空间。而在 uCLinux 中，每个应用程序都有自己的虚拟地址空间，在应用程序中不能直接访问物理缓冲区地址。因此，uCLinux 在文件操作 file—operations 结构中提供了 mmap 函数，可将文件的内容映射到用户空间。对于帧缓冲设备，则可通过映射操作，将屏幕缓冲区的物理地址映射到用户空间的一段虚拟地址中，之后用户就可以通过读写这段虚拟地址访问屏幕缓冲区，在屏幕上绘图[3]。

Framebuffer 中内存块分布如图 1 ：

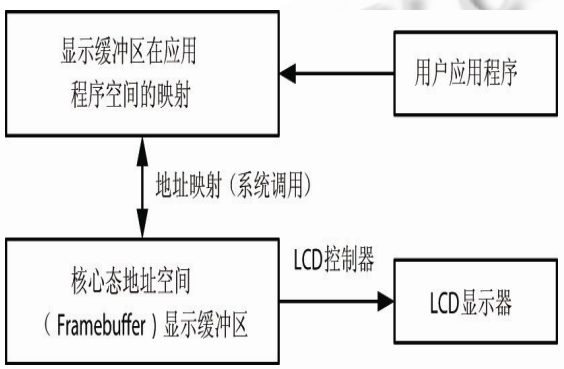

图 1　内存块分布图

3　底层LCD驱动接口

3.1 Linux 下设备驱动程序

对于 LCD 的驱动，uCLinux 采用帧缓冲设备，帧缓冲设备为标准字符设备，其对应的设备文件为 /dev/fb*。本文 ADSP-BF561 平台中只备有一个显卡，/dev/fb 作为当前的帧缓冲设备，指向/dev/fb0。Framebuffer 设备依靠下面的 3 个数据结构，在 fb.h 中声明：

Struct fb—var—screeninfo

Struct fb—fix—screeninfo

Struct fb—info

第 1 个结构是用来描述图形卡的特性，通常由用户设置。第 2 个结构定义了图形卡的硬件特性，不能改变，用户选定了 LCD 控制器和显示器后，它的硬件特性也就相应确定下来。第 3 个结构定义了当前图形卡 Framebuffer 设备的独立状态，本设计中只有一个 Framebuffer，这个结构是惟一内核空间可见的。

3.2 Framebuffer 驱动程序的实现

应用程序通过内核对 Framebuffer 的控制，主要有下面 3 种方式：

1)读/写/dev/fb。相当于读/写屏幕缓冲区。

2)映射(map)操作。通过映射操作，可将屏幕缓冲区的物理地址映射到用户空间的一段虚拟地址中，之后用户就可以通过读写这段虚拟地址访问屏幕缓冲区，在屏幕上绘图。

3) I/O 控制。对于帧缓冲设备，设备文件的 ioctl 操作可读取/设置显示设备及屏幕的参数，如分辨率、显示颜色数、屏幕大小等等。ioctl 的操作是由底层的驱动程序来完成的。

因此，Framebuffer 驱动要完成的工作是分配显存的大小、初始化 LCD 控制寄存器、设置修改硬件设备相应的 var 信息和 fix 信息等。

帧缓冲设备采用“文件层-驱动层”的接口方式。在文件层次上，uCLinux 为其定义了一系列的读写控制操作：

```
static struct file—operations fb—fops = {
    owner: THIS—MODULE,
    read: fb—read, /*读操作*/
    write: fb—write, /*写操作*/
    ioctl: fb—ioctl, /*控制操作*/
    mmap: fb—mmap, /*映射操作*/
    open: fb—open, /*打开操作*/
    release: fb—release, /*关闭操作*/
};
```

应用程序需按文件的方式打开一个帧缓冲设备，如果打开成功，则可对帧缓冲设备进行读、写等操作[4]。而读写帧缓冲设备最主要的任务就是，获取帧缓冲设备在内存中的物理地址空间以及相应 LCD 的一些特性。

图 2 为 ADSP-BF561 平台应用程序写帧缓冲设备显示图形的全过程。

依据上面所述，基于 ADSP-BF561 平台编写帧缓冲驱动需要做如下实际工作：

1)编写初始化函数。初始化函数首先初始化 LCD 控制器，通过写寄存器设置显示模式和显示颜色数，然后分配 LCD 显示缓冲区。在 Linux 可通过 kmalloc 函数分配一片连续的空间。

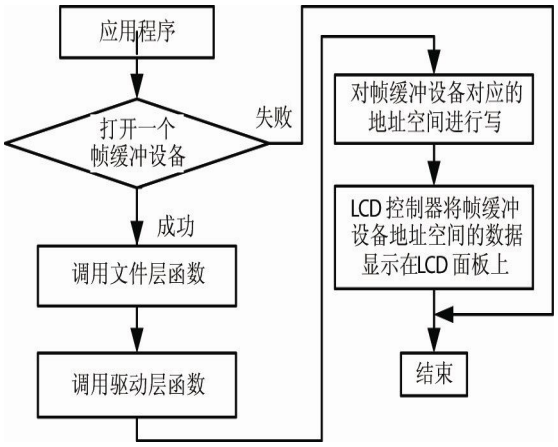

图 2　驱动程序实现框图

本文 ADSP-BF561 平台中，使用的是 24 位真彩色数字屏。其数据传入方式是以 8 位 RGB 方式传入的。根据计算公式：图像数据信息大小=(图像宽度*图像高度*记录像素的位数)/8，即需要分配的显示缓冲区为(320×240×24)/8=225K 字节，缓冲区通常分配在大容量的片外 SDRAM 中，起始地址保存在 LCD 控制器寄存器中。最后是初始化一个 fb—info 结构，填充其中的成员变量，并调用 register—framebuffer(&fb—info)，将 fb—info 登记入内核。下面是初始化帧缓冲设备的程序代码：

```
int init ADSP-BF561fb—init(void)
    {
.....
struct known—lcd—panels * p—lcd;//LCD Panel feature
    /* Init LCD Controller */
lcd—Init(p—lcd->bpp);
lcd—DispON(); //Open LCD display
    /* Init Framebuffer Parameter */
fb—parameter—init(p—lcd);
memset(&fb—info.gen, 0, sizeof(fb—info.gen));
    /* Print some information */
printk("ADSP-BF561framebuffer driver: Init ready\n");
.....
return 0;
    }
```

2)编写结构 fb—info 中函数指针 fb—ops 对应的成员函数，并依照硬件配置寄存器。

struct fb—ops 在 include/linux/fb.h 中定义。这些函数都用来设置/获取 fb—info 结构中的成员变量。当应用程序对设备文件进行 loctl 操作时候会调用它们。只需要下列 3 个函数就可以实现：

```
struct fb—ops {
.....
int (*fb—get—fix)(struct fb—fix—screeninfo *fix, int con, struct fb—info *info);
int (*fb—get—var)(struct fb—var—screeninfo *var, int con, struct fb—info *info);
int (*fb—set—var)(struct fb—var—screeninfo *var, int con,struct fb—info *info);
.....
};
```

struct fb—ops 在 include/linux/fb.h 中定义。这些函数都用来设置/获取 fb—info 结构中的成员变量。当应用程序对设备文件进行 loctl 操作时候会调用它们。

编写基于 ADSP-BF561 的驱动程序，主要的工作就是根据 ADSP-BF561 的硬件设置相应的寄存器。主要设置有：BF561fb_mach_info 结构，主要定义了所用显示屏的一些信息，如时钟、大小等；BF561fb_activate_var 函数关于寄存器的设置，涉及到 ADSP-BF561 LCD 控制器的有关设置，这些寄存器的设置根据所用 TFT-LCD 屏幕来进行设置；BF561fb_set_controller_regs 和 BF561fb_ lcd_init 函数的设置，涉及到 CPU 与 LCD 的物理连接，需要根据 LCD 与 CPU 的具体连接来设置各个 CPIO 寄存器。

3)测试驱动程序。编写完成的 LCD 驱动程序静态添加到内核当中后，将镜像下载到开发板中。在 LCD 显示屏上的左上角会显示一个小企鹅的图标。查看设备文件：

```
[root]# ls -al /dev/fb/0
```

LCD 已经成功驱动。用测试程序(test.c)测试驱动程序，可在显示屏上显示任意颜色的线条。

3.3 Framebuffer 设备的应用

在无线网络视频监控系统中，人机接口是一个关键部分。本实验中，对于人机接口的图形界面部分我们采用了 MiniGUI，这是一种针对嵌入式设备的，跨操作系统的轻量级图形界面支持系统。MiniGUI 的图形接口支持是 framebuffer 设备的典型应用。

MiniGUI 采用了独特的体系结构，从原理上采用分层设计的方法，能够在不影响其他层次的基础上针对不同的应用进行改编或者重写。最顶层的 API 是提供给用户使用的编程接口；中间层 MiniGUI 核心包括了窗口系统的各个模块；最底层的 GAL 和 IAL 为 MiniGUI 提供了底层的图形接口支持以及输入设备如鼠标、键盘等的驱动[5]。

GAL 的作用在于把不同的图形设备抽象为一个逻辑设备，提供一些基本的绘图功能，向上层模块提供统一的接口，建立在 FrameBuffer 上，不必修改上层模块。通过更改与显示屏相关的 Framebuffer 驱动程序，使 MiniGUI 移植入 ADSP-BF561，从而实现图形界面。

整个移植过程分为三阶段：第一，开发和调试基于 ADSP-BF561 平台硬件的 framebuffer 驱动程序，并且调试图形引擎。这一部分是移植过程的关键。第二，定制和调试设备的输入引擎。第三，开发自己的应用程序，并且交叉编译和配置整个 MiniGUI，完成图形界面。

4 实验结果与分析

在本实验中，uClinux 在文件操作 file—operations 结构中提供 mmap 函数，将文件的内容直接映射到用户空间。Framebuffer 通过映射操作，将屏幕缓冲区的物理地址映射到用户空间的一段虚拟地址中，用户通过读写这段虚拟地址访问屏幕缓冲区，在屏幕上绘图。

移植 MiniGUI 系统，需要首先实现 uClinux 内核中的 Framebuffer 驱动。对应于 ADSP-BF561 中的 LCD 控制器，该部分驱动程序必须以静态方式编译进内核，在系统启动时由传递进内核的启动参数激活设备。

实验中，根据 ADSP-BF561 的硬件设置相应的寄存器以及各个配置值的设置非常关键，应严格按照硬件平台和系统为准。否则会导致图像变形或无法显示图形界面。

5 结束语

经过测试，本文中的 LCD 驱动程序已在 ADSP-BF561 平台上成功实现。Framebuffer 作为基础图形设施，把显存抽象成一种设备，通过对此设备的读写直接对显存进行操作。在 MiniGUI 移植入 ADSP-BF561 的过程中，起到了关键性的作用。使轻小级图形界面在 DSP 平台上能够成功搭建，实现了无线网络视频监控系统的人机接口。

参考文献

1 于明俭,陈向阳,方汉.Linux 程序设计权威指南.北京:机械工业出版社, 2001.

2 Alex Buell. FrameBuffer How To, 1999,1(1):22.

3 田泽.嵌入式系统开发与应用.北京:北京航空航天大学出版社, 2005.

4 邹思轶.嵌入式 Linux 设计与应用.北京:清华大学出版社, 2002.

5 北京飞漫软件技术有限公司.MiniGUI 技术白皮书. http://www.minigui.org, 2005.