

嵌入式系统 Linux 下 LCD 显示驱动的开发

李 勇, 张建正

(华东理工大学 电子与通信工程系, 上海 200237)

摘 要: 嵌入式系统通常使用 LCD 作为显示设备, 嵌入式 Linux 下, 如果图形界面接口采用的是 MicroWindows, MiniGUI 或 Qt-Embedded, 则 LCD 设备的驱动程序必须采用 Linux 的帧缓冲设备来处理与 LCD 控制器有关的底层命令。帧缓冲是 Linux 为图形设备提供的一个抽象接口, 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作。文中介绍了 Linux 帧缓冲设备驱动程序框架, 详细分析了帧缓冲设备驱动程序层次结构、核心功能模块和数据结构, 最后基于三星公司 S3C2410x 处理器的开发平台, 系统地给出了 Linux 下帧缓冲驱动程序开发和调试实例。

关键词: 嵌入式系统; Linux 操作系统; 帧缓冲; 网络文件系统

中图分类号: TP316

文献标识码: A

文章编号: 1005-3751(2006)02-0093-03

Development of LCD Device Driver in Linux Embedded System

LI Yong, ZHANG Jian-zheng

(Dept. of Electronic and Information Engineering, East China University of Technology, Shanghai 200237, China)

Abstract: An LCD is a standard display device for hand-held embedded systems. If GUIs (graphic user interface) such as MiniGUI, MicroWindows of Qt-Embedded in the embedded Linux are used, LCD device drivers must be implemented as Linux frame buffer device drivers in addition to those low level operations which only deal with commands provided by LCD controllers. The frame buffer device provides an abstraction for the graphics hardware. It represents the frame buffer of some video hardware, and allows application software to access the graphics hardware through a well-defined interface. This paper introduces the framework of the Linux frame buffer device driver, and gives a detailed analysis of the architecture, key functions and data structures of the Linux frame buffer device driver. By taking the platform of Samsung's S3C2410x processor for example, the paper systematically describes how to develop and debug a Linux frame buffer device driver.

Key words: embedded system; Linux operation system; frame buffer; NFS

0 引 言

随着高性能嵌入式处理器的普及和硬件成本的不断降低, 尤其是 ARM 系列处理器的推出, 嵌入式系统的功能越来越强。在多媒体应用的推动下, 彩色 LCD 也越来越多地应用到了嵌入式系统中, 如新一代手机和掌上电脑多采用 TFT 显示器件, 支持彩色图形界面和视频媒体播放。嵌入式系统操作系统有 Window CE, Palm OS 等。而 Linux 作为开放源代码的操作系统也在市场中占据了一席之地。由于 Linux 成本低廉、源代码开放, 因此成为国内外厂商极力发展的操作系统。

在应用需求的推动下, Linux 下也出现了许多图形界面软件包, 如 MicroWindows, MiniGUI, Embedded QT 等, 其图形界面及开发工具已经与 Windows CE 不相上下。在 Linux 图形软件包的开发和移植工作中都牵扯到 LCD

的驱动问题, 笔者参与了一个基于 ARM9 的嵌入式系统 TFT 真彩色 LCD 驱动的开发, 提出了在嵌入式 Linux 操作系统下调试 LCD 设备驱动的一种有效方法, 所用处理器是三星公司的 S3C2410X^[1], 操作系统 Linux 2.4.18, 应用 GNU GCC 交叉编译器。

1 硬件平台

S3C2410X 是三星公司的基于 ARM920T 的 S3C2410X 芯片。S3C2410X 集成了一个 LCD 控制器(支持 STN 和 TFT 带有触摸屏的液晶显示屏)、并集成 SDRAM, 触摸屏, USB, SPI, SD 和 MMC 等控制器, 处理器工作频率最高达到 203MHz。

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, 完成配置和显示控制。S3C2410X 中的 LCD 控制器可支持单色/彩色 LCD 显示器。支持彩色 TFT 时, 可提供 4/8/12/16 位颜色模式, 其中 16 位颜色模式下可以显示 65536 种颜色。配置 LCD 控制器重要的一步是指定显示缓冲区, 显示的内容就是从缓冲区中读出的, 其大小由屏幕分辨率和显示

收稿日期: 2005-05-12

作者简介: 李 勇(1981—), 男, 上海奉贤人, 硕士研究生, 研究方向为嵌入式系统及其应用; 张建正, 副教授, 研究方向为嵌入式系统及其应用。

颜色数决定。文中采用的是台湾元太 V16C6448AC^[2], 是 TFT 真彩 65536 色, 分辨率为 640×480 的 LCD 显示模块。

2 Linux 下的帧缓冲设备

嵌入式 Linux 下, 如果图形界面接口采用的是 Microwindows, MiniGUI 或 Qt-Embedded, 则 LCD 设备的驱动程序必须采用 Linux 的帧缓冲设备来处理与 LCD 控制器有关的底层命令。

帧缓冲(Frame Buffer)是 Linux 为图形设备提供的一个抽象接口, 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作。这种操作是抽象的、统一的。应用程序不必关心物理显存的位置、换页机制等等具体细节。这些都是由帧缓冲设备驱动来完成的。Linux 的帧缓冲设备对应的设备文件通常为 /dev/fb0~3。

帧缓冲设备属于字符设备, 采用“文件层-驱动层”的接口方式^[3,4]。

文件 fb.h 中定义了帧缓冲设备的驱动层接口 fb_info 结构体, fb_info 定义了当前工作的显示卡的状态, 它仅对内核可见。

文件 fbmem.c 中定义了帧缓冲设备的文件层接口 file_operations 结构体, 它对应用程序可见, 该结构体的定义如下: 该结构体中功能函数 open() 和 release() 不需要底层的支持, 而 read(), write(), mmap() 则需要调用 fb_get_fix(), fb_get_var(), fb_set_var() (这些函数位于结构体 fb_info 中指针 fbops 指向的结构体变量中) 等与底层 LCD 硬件相关的函数的支持。另一个功能函数是 ioctl(), ioctl() 是设备驱动程序中对设备的 I/O 通道进行管理的函数, 应用程序应用 ioctl() 系统调用来调用 fb_get_fix(), fb_get_var(), fb_set_var() 等方法来获得和设置结构体 fb_info 中 var, fix 和 cmap 等变量的信息。在 fbmem.c 中给出了 ioctl() 命令和 fb_info 中结构体 fb_ops 的成员函数的对应关系如下:

```
FBIOGET_VSCREENINFO fb_get_var
FBIOPUT_VSCREENINFO fb_set_var
FBIOGET_FSCREENINFO fb_get_fix
FBIOPUTCMAP fb_set_cmap
FBIOGETCMAP fb_get_cmap
FBIOPAN_DISPLAY fb_pan_display
```

用户应用程序只需要调用 FBIOXXXX 来操作 LCD 硬件。

文件 fbmem.c 中还定义了帧缓冲设备底层驱动的管理函数:

```
register_framebuffer(struct fb_info *fb_info)
unregister_framebuffer(struct fb_info *fb_info)
```

帧缓冲设备在驱动层所要做的工作仅仅是对 Linux 为帧缓冲的驱动层接口 fb_info 进行初始化, 然后调用这两个函数对其注册或注销。帧缓冲设备驱动层接口直接对 LCD 设备硬件进行操作, 而 fbmem.c 可以记录和管理

多个底层设备驱动。

嵌入式 Linux 操作系统对帧缓冲设备的初始化入口也在 fbmem.c 中定义, 如果帧缓冲设备驱动是静态编译进内核的, 必须给予此结构一个入口。如果采用模块加载方式, 则不必要关心这个。

3 帧缓冲驱动的编写

3.1 编写结构体 fb_info 中 fb_ops 对应的成员函数

对于本嵌入式系统的实现, 需要下列 5 个函数^[5]:

```
static struct fb_ops s3c2410fb_ops = {
    owner: THIS_MODULE,
    fb_get_fix: s3c2410fb_get_fix,
    fb_get_var: s3c2410fb_get_var,
    fb_set_var: s3c2410fb_set_var,
    fb_get_cmap: s3c2410fb_get_cmap,
    fb_set_cmap: s3c2410fb_set_cmap,
};
```

结构体 fb_ops 在 include/linux/fb.h 中定义。

这些函数都是用来设置和获取驱动层接口 fb_info 结构体中的成员变量的, 前文已提过当应用程序对设备文件进行 ioctl 操作时候会调用它们。对于 fb_get_fix() 和 fb_get_var() 应用程序传入的是 fb_info 中的变量 fix 和 var, fb_set_var() 函数则是对 var 变量进行设置。同样 fb_get_cmap() 和 fb_set_cmap() 则是对变量 cmap 内容进行读取和设置。在这 5 个函数中, fb_set_var() 设置了显示设备的显示模式, 是最重要的一个函数。文中首先根据需为当前硬件定义一个专有结构体, 该结构体包括一个 fb_info 结构变量, 及其它与所选 LCD 硬件有关的所有参数, 因此对结构体 fb_ops 中成员函数对结构体 fb_info 的操作实际上就是对结构体 S3C2410fb_info 的操作。该结构体定义如下:

```
struct s3c2410fb_info {
    struct fb_info fb; //fb_info 结构变量
    u_int max_bpp;
    u_int max_xres;
    u_int max_yres;
    ... //其他与 LCD 硬件有关的参数
};
```

结构体 fb_ops 中的成员函数的流程相似, 文中在此仅给出 fb_set_var() 流程图(见图 1)。

3.2 编写初始化函数

初始化函数^[5]首先初始化结构体 fb_info, 填充其中的成员变量, 这些成员变量的参数值由 LCD 显示器厂商的手册获得。然后通过 consistent_alloc 函数分配一片连续的空间。笔者采用的 LCD 显示方式为 640×480, 16 位彩色。需要分配的显示缓冲区为 640×480×2=600k 字节, 缓冲区通常分配在片外 SDRAM 中, 起始地址保存在 LCD 控制器寄存器中。最后, 设置显示控制器参数并初

始化 LCD 控制器,最后调用 register_framebuffer(&fb_info) 将 fb_info 登记入内核。文中定义的结构体 S3C2410fb_info 包含一个 fb_info 结构变量,因此对结构体 S3C2410fb_info 的初始化实际上就是对结构体 fb_info 的初始化。

初始化函数如下:

```
int init_s3c2410fb_init(void)
{
    struct s3c2410fb_info * fbi;
    fbi = s3c2410fb_init_fbinfo(); //对结构 s3c2410fb_info 初始化,包括对 fb_info 的初始化,LCD 硬件参数保存到结构中
    ret = s3c2410fb_map_video_memory(fbi);
    //根据 LCD 硬件参数开辟显示存储器空间
    if (ret) { kfree(fbi); return ret; } //出错返回
    s3c2410fb_set_var(&fbi->fb.var, -1, &fbi->fb); //
    设置显示控制器参数
    ret = register_framebuffer(&fbi->fb); //注册显示驱动程序
    if (ret < 0) { kfree(fbi); return ret; } //出错返回
    printk("Installed s3c2410 frame buffer \n");
    //在控制台显示安装显示驱动程序成功
    return 0;
}
```

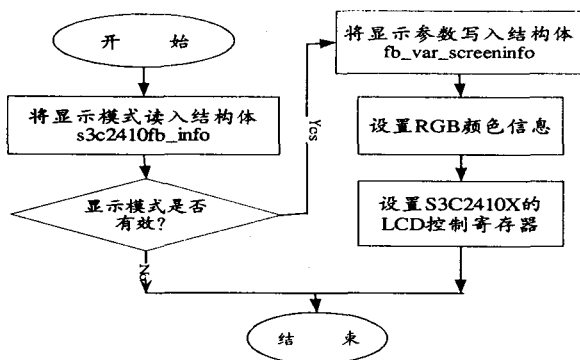


图1 函数 fb_set_var() 流程图

4 帧缓冲驱动的调试

驱动程序编写完成后,开发者可选择将其编译为动态加载模块,或静态地编译入内核中。文中在驱动程序的调试过程中采用 NFS 文件系统动态加载方式进行帧缓冲驱动的调试。

NFS(Network FileSystem)是由 SUN 公司发展,并于 1984 年推出,能够实现不同的系统间的文件共享,它的通讯协定设计与主机及操作系统无关。当使用者想用远端文件时只要使用“mount”命令就可把远程文件系统安装在自己的文件系统之下,使得远端的文件使用上和本地系统上的文件没有两样。使用 NFS 可以使嵌入式应用程序的开发和调试变得更为方便,目标板嵌入式系统可以直接通过局域网访问和使用服务器上的文件,解决了调试过程中反复烧写/擦除目标板闪存所带来的不便。

文中的调试模型(见图2)中,宿主机运行 Redhat linux 9.0,设置 IP 地址:192.168.100.99 并开启 NFS 系统服务。目标板系统内核选用的是 Linux 2.4.18-rmk7-px-al,在三星公司 S3C3410 SMDK 内核配置的基础上将内核的 LCD 驱动的选项配置为动态模块加载方式,并在内核菜单 File systems -> Network File Systems 下配置:

< * > NFS file system support

[*] Provide NFSv3 client support

[*] Root file system on NFS

菜单 Networking options 下配置:

[*] IP: kernel level autoconfiguration

[] IP: DHCP support

[*] IP: BOOTP support

[] IP: RARP support

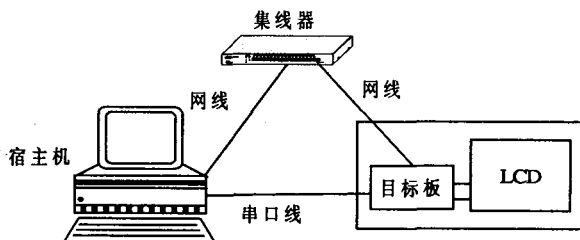


图2 LCD 驱动调试模型

最后在交叉编译内核后,连同 Linux 的引导程序烧写到目标板 FLASH 中。目标板第一次上电时设置引导程序,使之在以后每次引导 Linux 内核的同时自动设置目标板 Linux 操作系统 IP 地址为:192.168.100.100。这样目标板系统上电或复位后就会运行 FLASH 中的 Linux 内核,而目标板 Linux 操作系统所需的文件系统则位于局域网内的宿主机中。文中采用了 Trolletech 公司的 Embedded QT 2.3.6 作为目标板 Linux 图形界面库。目标板系统设置使用串口作为控制台(Console),调试过程中,首先在目标板中启动 Linux 内核,启动内核后,目标板上就加载了宿主机上的 NFS 文件系统,目标板可以直接访问宿主机上的文件,在宿主机中修改并交叉编译帧缓冲驱动后,就可以通过串口(目标板控制台)在目标板上运行 insmod/rmmod 命令动态加载/卸载帧缓冲驱动来进行调试。调试成功后,在目标板控制台执行 ./qpe 就可以运行事先编译好的基于 Embedded QT 2.3.6 的 Linux Qtopia 1.6.2 图形界面了。

5 结束语

文中介绍了 Linux 操作系统下帧缓冲设备驱动程序的原理和框架,编写了一个典型的帧缓冲设备驱动程序,并给出了在嵌入式 Linux 操作系统下调试 LCD 显示设备驱动程序的一种有效的方法。

参考文献:

[1] SAMSUNG ELECTRONICS. USER'S MANUAL S3C2410

(下转第 187 页)

13) end while

上述流程中, $W = (W_1, W_2, \dots, W_L)$, 其中 $W_i = (w_{i1}, w_{i2}, \dots, w_{in})^T$ 是第 i 个粒子的位置; $V = (V_1, V_2, \dots, V_L)$, 其中 $V_i = (v_{i1}, v_{i2}, \dots, v_{in})^T$ 是第 i 个粒子的速度(即移动的距离); G_p 是 m 个粒子迄今搜索到的最优适应值, 其对应的粒子位置矩阵是 $G_p = (G_{p1}, G_{p2}, \dots, G_{pL})$; G_g 是粒子群迄今搜索到的最优适应值, 对应的最优粒子位置是 g_{best} ; 线性减小 H 的策略有多种, 下式是其中较好的一种^[1]:

$$H(t) = h_k - h_L(t/T_{\max}) \quad (7)$$

其中, h_k 和 h_L 是常数, h_k 是 H 的最大值, 而 $h_k - h_L$ 则是 H 的最小值。

上面简要介绍了利用粒子群优化神经网络权值的算法, 最后以某省电网 1998 年 1 月 17 日(星期六)数据在短期电力负荷预测的 RBF 神经网络^[8]应用为例, 试算了用文中提出的优化方法与标准最小二乘法对 RBF 网络权值^[9]进行了优化比较, 结果如表 2 所示, 效果良好, 平均预测误差减小了 0.64%。可见用粒子群优化神经网络的算法优化效果明显。

表 2 优化前后电力负荷预测结果的比较

时刻	实际 负荷	预测负荷 (最小二乘 /粒子群)	相对误差 (最小二乘 /粒子群)	时刻	实际 负荷	预测负荷 (最小二乘 /粒子群)	相对误差 (最小二乘 /粒子群)
01:00	672	662/680	1.49/1.20	13:00	752	720/768	4.25/2.13
02:00	659	653/664	0.91/0.76	14:00	739	750/746	1.49/0.95
03:00	637	660/652	3.61/2.35	15:00	714	728/702	1.96/1.68
04:00	627	631/636	0.64/1.43	16:00	735	758/747	3.13/1.63
05:00	648	661/654	2.00/0.92	17:00	740	762/761	2.97/2.84
06:00	667	685/651	2.70/2.40	18:00	741	760/752	2.56/1.48
07:00	710	743/683	4.65/3.80	19:00	736	761/754	3.39/2.44
08:00	708	698/707	1.41/0.14	20:00	739	751/748	1.62/1.22
09:00	719	701/708	2.50/1.53	21:00	688	708/706	2.90/2.61
10:00	716	708/710	1.11/0.84	22:00	679	711/688	4.71/1.32
11:00	711	721/712	1.41/0.14	23:00	672	673/697	0.15/3.72
12:00	711	732/731	2.95/2.81	24:00	690	722/713	4.63/3.33

实际日最高负荷为 752;

实际日最低负荷为 627;

预测日最高负荷(最小二乘)为 762; 预测日最高负荷(粒子群)为 768;

预测日最低负荷(最小二乘)为 631; 预测日最低负荷(粒子群)为 636;

平均预测误差(最小二乘)为 2.46; 平均预测误差(粒子群)为 1.82;

注: 负荷单位(MW); 误差单位(%)。

3 结束语

文中采用了粒子群优化 RBF 网络学习的算法, 即分组训练合成优化。该算法利用粒子之间的合作与竞争以实现多维复杂空间的高维搜索能力, 找出神经网络权值的最优解, 以达到优化神经网络学习的目的。结果表明用文中提出的算法所优化的神经网络收敛效果明显、收敛速度较快。但也存在局部最优的问题, 在以后的研究中可以把不同的网络的学习率分开调整, 这样大的步长可以保证粒子在全局内寻找最优解而不会陷于局部最优解, 小的步长可以保证不越出最优解的寻找范围。

参考文献:

- [1] 王岁花, 李爱国. 基于粒子群优化的 BP 网络学习算法[J]. 计算机应用与软件, 2003(8): 1-2.
- [2] Shi Y, Eberhart R. A modified particle swarm optimizer[A]. IEEE World Congress on Computational Intelligence[C]. Piscataway, NJ: IEEE Press, 1998. 69-73.
- [3] 杨维, 李奇强. 粒子群优化算法综述[J]. 中国工程科学, 2004(5): 2-5.
- [4] 阎平凡, 张长水. 人工神经网络与模拟进化计算[M]. 北京: 清华大学出版社, 2000.
- [5] 王耀南. 计算智能信息处理技术及应用[M]. 长沙: 湖南大学出版社, 1999.
- [6] 褚蕾蕾, 陈绥阳, 周梦. 计算智能的数学基础[M]. 北京: 科学出版社, 2002.
- [7] 张讲社, 徐字本, 郑亚林. 计算智能中的仿生学: 理论与算法[M]. 北京: 科学出版社, 2003.
- [8] 胡迎松, 陈明, 范志明. 一种电力系统短期负荷预测的 RBF 优化算法[J]. 华中科技大学学报, 2003(3): 2-3.
- [9] 韩民晓, 许振华, 俞有瑛. RBF 神经网络在电力负荷预测中的应用[J]. 华中电力学院学报, 1994(4): 1-5.

(上接第 95 页)

- X32 - Bit RISC Microprocessor[EB/OL]. <http://www.samsung.com/Products/Semiconductor/SystemLSI/MobileSolutions/MobileASSP/MobileComputing/S3C2410/S3C2410.htm>, 2001-05.
- PRIME VIEW INTERNATIONAL. USER'S MANUAL V16 C6448AC[EB/OL]. <http://www.jazzbear.idv.tw:8080/PKM/Projects/LART/User/8912066/present/pdf/V16C6448AC.pdf>, 2000-10.

- Rubini A. LINUX 设备驱动程序[M]. 北京: 中国电力出版社, 2004.
- Uytterhoeven G. The Frame Buffer Device[EB/OL]. <http://www.faqs.org/docs/Linux-HOWTO/Framebuffer-HOWTO.html>, 1998-09.
- Simmons J. Linux Frame Buffer Driver Writing HOWTO[EB/OL]. <http://linuxconsole.sourceforge.net/fbdev/HOWTO/>, 2001-10.