ARM11 嵌入式系统 Linux 下 LCD 的驱动设计

张伽伟，周安栋，罗　勇

（海军工程大学 电子工程学院，湖北 武汉　430033，E-mail：290519038@qq.com）

摘　要： 对 ARM11 嵌入式系统 Linux 下的 LCD 驱动设计进行了研究，分析了在 Framebuffer 基础上编写 LCD 驱动的方法和两种直接读写 GPIO 的 LCD 驱动方法，通过测试归纳出 3 种 LCD 驱动方法的优缺点。在比较 Framebuffer 和两种直接读写 GPIO 的 LCD 驱动方法优劣的基础上，实现了 Framebuffer 与直接读写 GPIO 驱动结合的 LCD 控制方式。实际应用表明：用该方法控制 LCD 显示，具有显示速度快、驱动简单、移植性强的优点。

关　键　词： Linux；ARM11；LCD 驱动；Framebuffer；GPIO

中图分类号： TN27；TP399　　**文献标识码：** A　　**DOI：** 10.3788/YJYXS20112605.0660

Design of LCD Driving in Linux Based on ARM11 Embedded System

ZHANG Jia-wei, ZHOU An-dong, LUO Yong

(*Department of Electrical Engineering, Naval University of Engineering, Wuhan*　430033, *China*, *E-mail*: 290519038@qq.com)

Abstract: The LCD driver in Linux based on ARM11 embedded system was studied. A method of LCD driving design on the basis of the framebuffer, and two methods on the basis of reading and writing GPIO directly were analyzed. The advantages and disadvantages of the three LCD drivers were summarized by testing. A LCD control mode was implemented by combining the framebuffer and direct both reading and writing GPIO driver. The application indicates that this method can obtain following advantages: faster speed, simpler driver and stronger transplantation.

Key words: Linux; ARM11; LCD driver; Framebuffer; GPIO

1　引　　言

从 20 世纪 70 年代单片机的出现到今天，嵌入式系统已经有了近 40 年的发展历史。ARM 微处理器作为嵌入式系统中的一个重要代表，目前在各个领域得到了广泛应用[1-3]。三星公司生产的 S3C6410 基于 ARM1176JZF-S 核，主频为 533 MHz，最高可达 667 MHz，较 ARM7 主频提高了 10 倍，且拥有更丰富的片上资源，处理速度更快，功能更强。ARM11 可采用 Linux 操作系统。Linux 操作系统拥有性能优良、源码公开、大小可裁减、移植性强、运行速度快、网络性能好等优点。ARM11 与 Linux 操作系统的结合能够充分发挥 ARM11 的功能。

本文以 ARM11 处理器 S3C6410 为平台，在嵌入式 Linux 操作系统下实现了 3 种 LCD 驱动方法，通过实际应用得出了 3 种驱动方法的优点和不足，并实现了一种新的 LCD 控制方式。

2　基于 Framebuffer 机制的 LCD 驱动设计

2.1　Framebuffer 架构

Framebuffer 驱动是显示驱动的标准，是出

现在 Linux2.2(及以后)内核当中的一种驱动接口。Framebuffer 将显示设备抽象为帧缓冲区,用户通过内存映射将其映射到进程地址空间之后,就可以直接进行读写操作,操作可以直接反应到屏幕上[4]。

Framebuffer 驱动的数据结构主要包括 4 部分:fb_info、fb_fix_screeninfo、fb_var_screeninfo 和 fb_ops。fb_info 结构是 Linux 为帧缓冲设备定义的驱动层接口,用于用户在内核空间的调用,它包含了底层函数和有记录设备状态的数据;fb_fix_screeninfo 结构用来描述设备无关、不可变更的信息,用户可以使用 FBIOGET_FSCREENINFO 命令来获得这些信息;fb_var_screeninfo 结构用来描述可更改的配置信息,可以使用 FBIOGET_VSCREENINFO 命令获得这些信息,使用 FBIOPUT_VSCREENINFO 命令写入这些信息;fb_ops 结构定义了所有对帧缓冲设备的操作,可以使用 ioctl 系统调用来操作设备,该结构用来支持 ioctl 的这些操作。Framebuffer 驱动的模块文件都在 linux/drivers/video/中[5]。

2.2 S3C6410 的 LCD 驱动开发

S3C6410 是一款内部集成 LCD 控制器的 ARM11 芯片[6],显卡驱动可以在 ARM9 的显卡驱动 S3C2410fb.c 的基础上进行修改[7]。S3C6410 支持黑白、4 级灰度、16 级灰度、256 色、4 096 色的 STN 液晶屏,支持黑白、4 级灰度、16 级灰度、256 色、64k 色、真彩色的 TFT 液晶屏。两类屏的尺寸可在 8.9～30.2 cm(3.5 ～12.1 in),屏幕分辨率可以达到 1 024×768[8]。选择好相应不带控制器的 LCD,需要在驱动程序中实现和硬件相关,并填充 fb_info 结构。编写驱动完成的工作如图 1 所示。

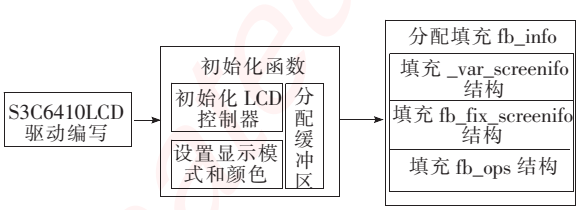

图 1 编写驱动完成的内容

Fig. 1 Content of the driver

在 S3C2410fb.c 中:s3c24xxfb_probe 函数完成了初始化 LCD 控制器,分配显示缓冲区和注册一个 Framebuffer 驱动; s3c2410fb_calculate_stn_lcd_regs 函数设置 STN 液晶屏的显示模式和显示颜色数;s3c2410fb_calculate_tft_lcd_regs 函数设置 TFT 液晶屏的显示模式和显示颜色数;s3c2410fb_calculate_stn_lcd_regs、s3c2410fb_calculate_tft_lcd_regs 和 s3c2410fb_activate_var 3 个函数设置 LCD 控制寄存器[9];s3c2410fb_check_var 函数确认设置液晶屏可变参数;s3c2410fb_set_par 函数用来更改硬件设置。完成 S3C6410 的 LCD 驱动 S3C2410fb.c 并设置好 config 和 Makefile 文件后,就可以把驱动程序编译进 Linux 的内核并下载到 S3C6410 中[10]。

2.3 Framebuffer 机制 LCD 驱动的应用实现

底层驱动加载到内核后,就可以编写相关应用程序对其加以实现。Framebuffer 应用程序的流程图如图 2 所示。

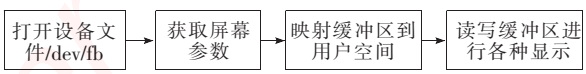

图 2 Framebuffer 应用程序流程图

Fig. 2 Flow chat of Framebuffer

核心程序如下:

```
……
fd=open("/dev/fb",O_RDWR);//打开设备文件
ioctl(fd, FBIOGET_FSCREENINFO, &finfo);//系统调用获取屏的固定参数
ioctl(fd, FBIOGET_VSCREENINFO, &vinfo);//系统调用获取屏的可变参数
/* 设置缓冲区大小 */
screensize = vinfo.xres * vinfo.yres * vinfo.bits_per_pixel / 8;
/* 映射缓冲区到用户空间 */
fbp = (char *)mmap(0, screensize, PROT_READ | PROT_WRITE, MAP_SHARED,fd, 0);
/* 对指针 fbp 进行读写进行各种显示 */
……
```

3 直接读写 GPIO 口 LCD 驱动设计

对于含控制器的 LCD,S3C6410 的 LCD 驱动设计可以不用自带的 LCD 控制器,采用更简单的设计方法直接读写 GPIO。GPIO 驱动是向 LCD 控制器和 LCD 的应用程序提供接口[11]。LCD 控制器有两类信号接口:一类是数据信号接

口;另一类是功能信号接口。本文以 LCD 控制器 TC6963 为例进行介绍,其他类型控制器以此为参考。TC6963 的信号接口主要有:数据接口 DB0～DB7,功能信号接口 WR 写信号、RD 读信号、C/D 通道选择信号和 CE 片选信号。WR、RD、CE 为低电平有效。C/D 为高电平时是命令通道,低电平时是数据通道[12]。直接读写 GPIO 口 LCD 驱动设计的工作就是 GPIO 驱动和上层应用程序配合完成对 LCD 控制器的读写操作。

GPIO 驱动的设计方法有两种:一是仅实现对 GPIO 控制寄存器和数据寄存器的读写操作,LCD 控制器的读写时序全部由应用程序完成;二是 LCD 控制器的读写时序全部在 GPIO 驱动中完成,由 GPIO 驱动向应用程序提供写数据接口、写命令接口和显示数据接口。

3.1 直接读写 GPIO 口 LCD 驱动方法一的实现

方法一是 LCD 控制的读写时序全部在应用程序中完成,GPIO 驱动提供写控制和写数据寄存器的接口,该方法的程序结构如图 3 所示。

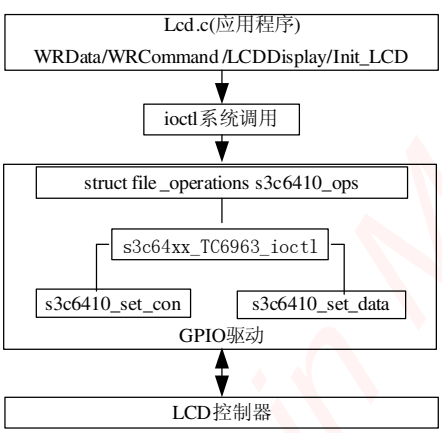

图 3 GPIO 驱动方法一结构图

Fig. 3 Diagram of GPIO driver (method 1)

GPIO 驱动主要包括以下 4 个函数:

(1)读写 GPIO 控制寄存器的函数 s3c6410_set_con。该函数的核心语句是 writel(tmp, S3C64XX_GPICON),其中 writel 是驱动层的写函数,tmp 是写入控制寄存器的数据,S3C64XX_GPICON 是 S3C6410 中第 I 组 GPIO 口的控制寄存器[13]。

(2)读写 GPIO 数据寄存器的函数 s3c6410_set_data。该函数的核心语句是 writel(tmp, S3C64XX_GPIDAT),writel 和 tmp 和 s3c6410_set_con 中的一样,S3C64XX_GPIDAT 是 S3C6410 中第 I 组 GPIO 口的数据寄存器。这里的数据寄存器要与 s3c6410_set_con 中的控制寄存器保持一致。

(3)系统调用接口函数 s3c64xx_TC6963_ioctl(struct file * file, int cmd, long int data)。该接口中定义两个命令分别调用 s3c6410_set_con 和 s3c6410_set_data 函数。

(4)应用程序通过系统调用 iotcl(fd,1,xx)和 iotcl(fd,2,xx)可以实现对 GPIO 口控制器和数据寄存器的读写,按照 LCD 控制器的读写时序[2]进行读写操作就实现了 LCD 的显示控制。

3.2 直接读写 GPIO 口 LCD 驱动方法二的实现

方法二是 LCD 控制的读写时序在 GPIO 驱动中完成,该方法的程序结构如图 4 所示。

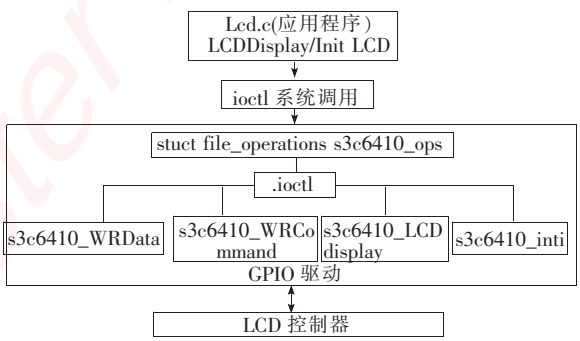

图 4 GPIO 驱动方法二结构图

Fig. 4 Diagram of GPIO driver (method 2)

GPIO 驱动主要包括写数据函数 s3c6410_WRData、写命令函数 s3c6410_WRCommand、显示数据函数 s3c6410_LCDdisplay、初始化函数 s3c6410_inti、系统调用接口函数 s3c64xx_TC6963_ioctl 和系统调用结构体 file_operations 结构的填充。

s3c6410_WRData 和 s3c6410_WRCommand 是按照 LCD 控制器 TC6963 的时序来进行的,以 s3c6410_WRData 为例,程序如下:

```
static void s3c64xx_WRData(int data1)
{ unsigned tmp1=0x00;
tmp1=(readl(S3C64XX_GPIDAT)&～0x0300);
//写命数据有效 C/D=0,片选有效 CE=0
writel(tmp1, S3C64XX_GPIDAT);
data1=data1&0x00ff;
tmp1=(readl(S3C64XX_GPIDAT)&～0x00ff)|
```

```
data1; //将数据送入 GPI0～7
writel(tmp1, S3C64XX_GPIDAT);
tmp1 = (readl(S3C64XX_GPIDAT) & ～0x0400);//WR=0 写允许
writel(tmp1, S3C64XX_GPIDAT);
tmp1=(readl(S3C64XX_GPIDAT)&～0x0400)|0x0400; //WR=1 写禁止
writel(tmp1, S3C64XX_GPIDAT);}
```

写命令函数 s3c6410_WRCommand 也按照写数据函数类似的方法编写即可，显示数据函数 s3c6410_LCDdisplay 通过调用 s3c6410_WRData 和 s3c6410_WRCommand 函数将数据送到液晶屏显示。系统调用函数和系统调用结构的填充与方法一类似。

应用程序只需通过系统调用函数调用 s3c6410_LCDdisplay 就可实现数据的 LCD 显示。

4 三种 LCD 驱动方法比较

S3C6410 中程序在内核态的执行速度远高于程序在用户态执行速度，在执行速度上他们是数量级上的差别，这点决定了选择不同的驱动编写方式，会得到不同的 LCD 驱动效果。本文通过比较得出了 3 种驱动方式的特点如表 1 所示。

表 1 3 种 LCD 驱动方法的优缺点对照

Table 1 Advantages and disadvantages of the three LCD drivers

驱动方法	优 点	缺 点
Framebuffer 机制的 LCD 驱动	直接读写显存，反应速度快、执行效率高；应用程序简单，Framebuffer 接口通用。	底层硬件驱动复杂，改硬件驱动程序调试繁琐。
直接读写 GPIO，方法一：控制时序在应用层	硬件驱动通用性强，改变 LCD 控制器后，GPIO 驱动不用改，只需修改应用程序；大部分工作在应用层，程序调试方便。	应用程序与内核相互调度频繁，速度慢、执行效率低。
直接读写 GPIO，方法二：控制时序在硬件驱动层	驱动向应用层提供的接口少，应用程序简单；应用程序和内核交互少，运行效率高。	通用性差，LCD 控制器改变就需要重新写 GPIO 驱动；调试硬件驱动，调试繁琐。

实际应用中可参考这 3 种 LCD 驱动编写方法的利弊，权衡自己 LCD 程序设计的需求，选择适合的驱动编写方法。

5 Framebuffer 与直接读写 GPIO 结合的 LCD 控制

通过分析 Framebuffer 机制下的 LCD 驱动和直接读写 GPIO 的 LCD 驱动发现：如果将二者结合起来，在底层驱动采用直接读写 GPIO 驱动方式，在内核空间和应用程序数据的传输方式上采用 Framebuffer 提供的帧缓冲区将显存映射到用户空间，底层驱动直接从帧缓冲区读数据，克服了这两种驱动方式的缺点，既提高了程序运行效率，也易编写底层驱动。

采用这种方式的应用程序核心部分如下：

```
……
fd = open("/dev/LCD", 0); //LCD 是 GPIO 驱动注册的设备号
fdd= open("/dev/fb",O_RDWR);//fb 是 Framebuffer 注册的设备号
set_init_lcd(fd); //初始化 LCD
/* 映射缓冲区到用户空间 */
fbp=(int *)mmap(0,60 * 272,PROT_READ|PROT_WRITE,MAP_SHARED,fdd,0);
for(i=0;i<RamLength;i++)
{
*(fbp+i)=tupian[i]; //把要显示的图片数组全部放入 Framebuffer 帧缓冲区
}
/* 相关系统调用，控制 LCD 显示 */
……
```

底层 GPIO 驱动只需要在原 GPIO 驱动的基础上，把 s3c6410_LCDdisplay 函数中数据从应用程序读取改为从帧缓冲区读取，然后编译进 linux 内核即可。这样就实现了 Framebuffer 与直接读写 GPIO 结合的 LCD 驱动设计。采用该驱动实现控制 128×64 单色屏的效果图如图 5 所示，采用前面两种驱动方式的实物效果图和此图相似，

图 5　液晶显示效果图

Fig. 5　Figure of LCD 128×64

他们的区别主要体现在驱动程序的复杂度和液晶读写的时间方面。

6　结　　论

分析并实现了 ARM11 嵌入式 Linux 在 Framebuffer 基础上编写 LCD 驱动的方法和两种直接读写 GPIO 的 LCD 驱动编写方法。在总结这 3 种 LCD 驱动方法优缺点的基础上，实现了 Framebuffer 与直接读写 GPIO 驱动结合的 LCD 控制方式，使得 LCD 控制的显示速度更快、驱动更简单、一致性更强。较为全面地阐述了 ARM11 嵌入式系统 Linux 下 LCD 的驱动设计方法。尽管 LCD 的种类很多，但驱动程序的设计都有可以遵循的模式，可以根据设计需求选择适合的设计方法。该方法对于嵌入式系统开发中 LCD 显示驱动的设计具有一定的借鉴意义。

参　考　文　献：

[1] 赵小欢，夏靖波，李明辉. 基于 ARM 和 FGGA 的视频监控系统设计[J]. 液晶与显示，2010，25(1)：94-98.
[2] 尹柱霞，郑喜凤，于洪涛. ARM＋FPGA 控制的 LED 脱机屏系统设计[J]. 液晶与显示，2010，25(2)：262-267.
[3] 马舜峰，金龙旭，安少婷，等. 一种基于 ARM9 的彩色 TFT-LCD 模块设计及实现[J]. 液晶与显示，2010，25(5)：718-723.
[4] 冯国进. 嵌入式 Linux 驱动程序设计从入门到精通 [M]. 北京：清华大学出版社，2008：151-168.
[5] 张策，孙绪刚. 基于 Framebuffer 的 LCD 驱动设计 [J]. 计算机工程与设计，2009，30(23)：5372-5375.
[6] 李春燕. Linux 内核的嵌入式系统裁减以及 LCD 驱动的实现 [J]. 电脑知识与技术，2008，1(1)：29-37.
[7] 谭显强，吴宁，范彩霞. 基于 S3C2410 的 LCD 驱动系统设计 [J]. 苏州科技学院学报，2009，19(3)：70-74.
[8] 广州友善之比计算机科技有限公司. mini6410 用户手册-20100814 [S]. 广州，2010：11-12.
[9] 常赟杰，张位勇. Framebuffer 的 LCD 驱动程序设计[J]. 电脑编程技巧与维护，2009，15(4)：18-19.
[10] 鲁宝宏，郭磊，魏世民. 嵌入式 L i nux 平台下 LCD 驱动的开发与实现 [J]. 应用设计，2008，45(9)：28-30.
[11] 苏哲欣，刘鸿飞，薛晓. 基于嵌入式 Linux 的 LCD 驱动分析与实现 [J]. 工业控制计算机，2009，22(2)：29-30.
[12] 孙俊喜. LCD 驱动电路、驱动程序设计及典型应用 [M]. 北京：人民邮电出版社，2009：64-72.
[13] Samsung 公司. S3C6410X 用户手册 [R]. Korea，2008：326-329.

嵌入式资源免费下载

总线协议：

1. 基于 PCIe 驱动程序的数据传输卡 DMA 传输
2. 基于 PCIe 总线协议的设备驱动开发
3. CANopen 协议介绍
4. 基于 PXI 总线 RS422 数据通信卡 WDM 驱动程序设计
5. FPGA 实现 PCIe 总线 DMA 设计
6. PCI Express 协议实现与验证
7. VPX 总线技术及其实现
8. 基于 Xilinx FPGA 的 PCIE 接口实现
9. 基于 PCI 总线的 GPS 授时卡设计
10. 基于 CPCI 标准的 6U 信号处理平台的设计
11. USB30 电路保护
12. USB30 协议分析与框架设计
13. USB 30 中的 CRC 校验原理及实现
14. 基于 CPLD 的 UART 设计
15. IPMI 在 VPX 系统中的应用与设计
16. 基于 CPCI 总线的 PMC 载板设计
17. 基于 VPX 总线的工件台运动控制系统研究与开发
18. PCI Express 流控机制的研究与实现
19. UART16C554 的设计
20. 基于 VPX 的高性能计算机设计
21. 基于 CAN 总线技术的嵌入式网关设计
22. Visual C 串行通讯控件使用方法与技巧的研究
23. IEEE1588 精密时钟同步关键技术研究
24. GPS 信号发生器射频模块的一种实现方案
25. 基于 CPCI 接口的视频采集卡的设计
26. 基于 VPX 的 3U 信号处理平台的设计
27. 基于 PCI Express 总线 1394b 网络传输系统 WDM 驱动设计
28. AT89C52 单片机与 ARINC429 航空总线接口设计
29. 基于 CPCI 总线多 DSP 系统的高速主机接口设计
30. 总线协议中的 CRC 及其在 SATA 通信技术中的应用
31. 基于 FPGA 的 SATA 硬盘加解密控制器设计
32. Modbus 协议在串口通讯中的研究及应用
33. 高可用的磁盘阵列 Cache 的设计和实现
34. RAID 阵列中高速 Cache 管理的优化

35. 一种新的基于 RAID 的 CACHE 技术研究与实现
36. 基于 PCIE-104 总线的高速数据接口设计
37. 基于 VPX 标准的 RapidIO 交换和 Flash 存储模块设计
38. 北斗卫星系统在海洋工程中的应用
39. 北斗卫星系统在远洋船舶上应用的研究
40. 基于 CPCI 总线的红外实时信号处理系统
41. 硬件实现 RAID 与软件实现 RAID 的比较
42. 基于 PCI Express 总线系统的热插拔设计
43. 基于 RAID5 的磁盘阵列 Cache 的研究与实现
44. 基于 PCI 总线的 MPEG2 码流播放卡驱动程序开发
45. 基于磁盘异或引擎的 RAID5 小写性能优化
46. 基于 IEEE1588 的时钟同步技术研究
47. 基于 Davinci 平台的 SD 卡读写优化
48. 基于 PCI 总线的图像处理及传输系统的设计
49. 串口和以太网通信技术在油液在线监测系统中的应用
50. USB30 数据传输协议分析及实现

VxWorks：

1. 基于 VxWorks 的多任务程序设计
2. 基于 VxWorks 的数据采集存储装置设计
3. Flash 文件系统分析及其在 VxWorks 中的实现
4. VxWorks 多任务编程中的异常研究
5. VxWorks 应用技巧两例
6. 一种基于 VxWorks 的飞行仿真实时管理系统
7. 在 VxWorks 系统中使用 TrueType 字库
8. 基于 FreeType 的 VxWorks 中文显示方案
9. 基于 Tilcon 的 VxWorks 简单动画开发
10. 基于 Tilcon 的某武器显控系统界面设计
11. 基于 Tilcon 的综合导航信息处理装置界面设计
12. VxWorks 的内存配置和管理
13. 基于 VxWorks 系统的 PCI 配置与应用
14. 基于 MPC8270 的 VxWorks BSP 的移植
15. Bootrom 功能改进经验谈
16. 基于 VxWorks 嵌入式系统的中文平台研究与实现
17. VxBus 的 A429 接口驱动
18. 基于 VxBus 和 MPC8569E 千兆网驱动开发和实现
19. 一种基于 vxBus 的 PPC 与 FPGA 高速互联的驱动设计方法
20. 基于 VxBus 的设备驱动开发

21. 基于 VxBus 的驱动程序架构分析
22. 基于 VxBus 的高速数据采集卡驱动程序开发
23. Vxworks 下的冗余 CAN 通讯模块设计
24. WindML 工业平台下开发 S1d13506 驱动及显示功能的实现
25. WindML 中 Mesa 的应用
26. VxWorks 下图形用户界面开发中双缓冲技术应用
27. VxWorks 上的一种 GUI 系统的设计与实现
28. VxWorks 环境下 socket 的实现
29. VxWorks 的 WindML 图形界面程序的框架分析
30. VxWorks 实时操作系统及其在 PC104 下以太网编程的应用
31. 实时操作系统任务调度策略的研究与设计

Linux：

1. Linux 程序设计第三版及源代码
2. NAND FLASH 文件系统的设计与实现
3. 多通道串行通信设备的 Linux 驱动程序实现
4. Zsh 开发指南-数组
5. 常用 GDB 命令中文速览
6. 嵌入式 C 进阶之道
7. Linux 串口编程实例
8. 基于 Yocto Project 的嵌入式应用设计
9. Android 应用的反编译
10. 基于 Android 行为的加密应用系统研究
11. 嵌入式 Linux 系统移植步步通
12. 嵌入式 CC++语言精华文章集锦
13. 基于 Linux 的高性能服务器端的设计与研究
14. S3C6410 移植 Android 内核
15. Android 开发指南中文版
16. 图解 Linux 操作系统架构设计与实现原理（第二版）
17. 如何在 Ubuntu 和 Linux Mint 下轻松升级 Linux 内核
18. Android 简单 mp3 播放器源码
19. 嵌入式 Linux 系统实时性的研究
20. Android 嵌入式系统架构及内核浅析
21. 基于嵌入式 Linux 操作系统内核实时性的改进方法研究
22. Linux TCP IP 协议详解
23. Linux 桌面环境下内存去重技术的研究与实现
24. 掌握 Android 7.0 新增特性 Quick Settings
25. Android 应用逆向分析方法研究

26. Android 操作系统的课程教学
27. Android 智能手机操作系统的研究
28. Android 英文朗读功能的实现
29. 基于 Yocto 订制嵌入式 Linux 发行版
30. 基于嵌入式 Linux 的网络设备驱动设计与实现
31. 如何高效学习嵌入式
32. 基于 Android 平台的 GPS 定位系统的设计与实现
33. LINUX ARM 下的 USB 驱动开发
34. Linux 下基于 I2C 协议的 RTC 驱动开发
35. 嵌入式下 Linux 系统设备驱动程序的开发
36. 基于嵌入式 Linux 的 SD 卡驱动程序的设计与实现
37. Linux 系统中进程调度策略

Windows CE：

1. Windows CE.NET 下 YAFFS 文件系统 NAND Flash 驱动程序设计
2. Windows CE 的 CAN 总线驱动程序设计
3. 基于 Windows CE.NET 的 ADC 驱动程序实现与应用的研究
4. 基于 Windows CE.NET 平台的串行通信实现
5. 基于 Windows CE.NET 下的 GPRS 模块的研究与开发
6. win2k 下 NTFS 分区用 ntldr 加载进 dos 源代码
7. Windows 下的 USB 设备驱动程序开发
8. WinCE 的大容量程控数据传输解决方案设计
9. WinCE6.0 安装开发详解
10. DOS 下仿 Windows 的自带计算器程序 C 源码
11. G726 局域网语音通话程序和源代码
12. WinCE 主板加载第三方驱动程序的方法
13. WinCE 下的注册表编辑程序和源代码
14. WinCE 串口通信源代码
15. WINCE 的 SD 卡程序[可实现读写的源码]
16. 基于 WinCE 的 BootLoader 研究
17. Windows CE 环境下无线网卡的自动安装
18. 基于 Windows CE 的可视电话的研究与实现
19. 基于 WinCE 的嵌入式图像采集系统设计
20. 基于 ARM 与 WinCE 的掌纹鉴别系统

PowerPC：

1. Freescale MPC8536 开发板原理图
2. 基于 MPC8548E 的固件设计
3. 基于 MPC8548E 的嵌入式数据处理系统设计
4. 基于 PowerPC 嵌入式网络通信平台的实现
5. PowerPC 在车辆显控系统中的应用
6. 基于 PowerPC 的单板计算机的设计
7. 用 PowerPC860 实现 FPGA 配置
8. 基于 MPC8247 嵌入式电力交换系统的设计与实现
9. 基于设备树的 MPC8247 嵌入式 Linux 系统开发
10. 基于 MPC8313E 嵌入式系统 UBoot 的移植
11. 基于 PowerPC 处理器 SMP 系统的 UBoot 移植
12. 基于 PowerPC 双核处理器嵌入式系统 UBoot 移植
13. 基于 PowerPC 的雷达通用处理机设计
14. PowerPC 平台引导加载程序的移植
15. 基于 PowerPC 嵌入式内核的多串口通信扩展设计
16. 基于 PowerPC 的多网口系统抗干扰设计
17. 基于 MPC860T 与 VxWorks 的图形界面设计
18. 基于 MPC8260 处理器的 PPMC 系统

ARM：

1. 基于 DiskOnChip 2000 的驱动程序设计及应用
2. 基于 ARM 体系的 PC-104 总线设计
3. 基于 ARM 的嵌入式系统中断处理机制研究
4. 设计 ARM 的中断处理
5. 基于 ARM 的数据采集系统并行总线的驱动设计
6. S3C2410 下的 TFT LCD 驱动源码
7. STM32 SD 卡移植 FATFS 文件系统源码
8. STM32 ADC 多通道源码
9. ARM Linux 在 EP7312 上的移植
10. ARM 经典 300 问
11. 基于 S5PV210 的频谱监测设备嵌入式系统设计与实现
12. Uboot 中 start.S 源码的指令级的详尽解析
13. 基于 ARM9 的嵌入式 Zigbee 网关设计与实现
14. 基于 S3C6410 处理器的嵌入式 Linux 系统移植

15. CortexA8 平台的μC-OSⅡ及 LwIP 协议栈的移植与实现
16. 基于 ARM 的嵌入式 Linux 无线网卡设备驱动设计
17. ARM S3C2440 Linux ADC 驱动
18. ARM S3C2440 Linux 触摸屏驱动
19. Linux 和 Cortex-A8 的视频处理及数字微波传输系统设计
20. Nand Flash 启动模式下的 Uboot 移植
21. 基于 ARM 处理器的 UART 设计
22. ARM CortexM3 处理器故障的分析与处理
23. ARM 微处理器启动和调试浅析
24. 基于 ARM 系统下映像文件的执行与中断运行机制的实现
25. 中断调用方式的 ARM 二次开发接口设计

Hardware：

1. DSP 电源的典型设计
2. 高频脉冲电源设计
3. 电源的综合保护设计
4. 任意波形电源的设计
5. 高速 PCB 信号完整性分析及应用
6. DM642 高速图像采集系统的电磁干扰设计
7. 使用 COMExpress Nano 工控板实现 IP 调度设备
8. 基于 COM Express 架构的数据记录仪的设计与实现
9. 基于 COM Express 的信号系统逻辑运算单元设计
10. 基于 COM Express 的回波预处理模块设计
11. 基于 X86 平台的简单多任务内核的分析与实现
12. 基于 UEFI Shell 的 PreOS Application 的开发与研究
13. 基于 UEFI 固件的恶意代码防范技术研究
14. MIPS 架构计算机平台的支持固件研究
15. 基于 UEFI 固件的攻击验证技术研究
16. 基于 UEFI 的 Application 和 Driver 的分析与开发
17. 基于 UEFI 的可信 BIOS 研究与实现
18. 基于 UEFI 的国产计算机平台 BIOS 研究
19. 基于 UEFI 的安全模块设计分析
20. 基于 FPGA NiosⅡ的等精度频率计设计
21. 基于 FPGA 的 SOPC 设计
22. 基于 SOPC 基本信号产生器的设计与实现
23. 基于龙芯平台的 PMON 研究与开发

Programming:

1. 计算机软件基础数据结构 - 算法
2. 高级数据结构对算法的优化
3. 零基础学算法
4. Linux 环境下基于 TCP 的 Socket 编程浅析
5. Linux 环境下基于 UDP 的 socket 编程浅析
6. 基于 Socket 的网络编程技术及其实现