Linux 下基于 I²C 协议的 RTC 驱动开发

孟令公,朱　宏,杨忠孝,杨裕辉
(电子科技大学　四川 成都　610054)

摘　要:在嵌入式中,Linux 渐渐成为一种流行操作系统,Linux 驱动开发也成为嵌入式开发中的必备环节。介绍 Linux 环境下基于 I²C 协议的 RTC 驱动程序开发与实现。首先研究了 Linux 环境下字符设备驱动程序框架,然后介绍 I²C 协议,在此基础上开发基于 I²C 协议的 RTC 字符设备驱动程序。对于驱动程序,这里详细介绍其整体架构和各模块实现细节。最终成功实现了基于 I²C 协议的 RTC 驱动程序,并移植到 Linux 操作系统中。

关键词:嵌入式系统;Linux;驱动程序;I²C
中图分类号:TP316.81　　**文献标识码**:B　　**文章编号**:1004-373X(2009)20-032-04

Development of RTC Driver Based on I²C under Linux

MENG Linggong, ZHU Hong, YANG Zhongxiao, YANG Yuhui
(University of Electronic Science and Technology of China, Chengdu, 610054, China)

Abstract: In embedded field, Linux OS has been more and more popular, the development of Linux driver is becoming a constituent job. Reaserch and development of RTC driver based on I²C under Linux OS are introduced. First, research on the character device driver framework is given. Then, I²C protocol is introduced, RTC driver based on I²C under Linux OS is developed, the scheme and realization are proposed. Finally, RTC driver based on I²C is realized and transplanted into Linux.

Keywords: embedded system; Linux; driver; I²C

0　引　言

近年来嵌入式系统的研究与开发渐渐成为热点,在嵌入式系统中,Linux 操作系统以其开源、稳定、可移植等种种优点,渐渐成为一种流行的操作系统。Linux 下各种驱动程序的开发经常是软件开发中必不可缺的环节,Linux 对其驱动程序提供了很好的支持框架。I²C 总线是一种由 Philips 公司开发的两线式串行总线,用于连接微控制器及其外围设备。I²C 总线产生于在 20 世纪80 年代,最主要的优点是其简单性和有效性。这里介绍 Linux 字符设备驱动以及 I²C 总线协议,并在此基础上开发基于 I²C 总线的 RTC 驱动程序。

1　Linux 字符设备驱动框架

在 Linux 内核中每个字符驱动程序都是基于以下框架进行设计的[1,2]:

```
struct file - operations {
    ssize - t ( *read) (struct file  *,char  - - user  *,size -
t,loff - t  *);
    ssize - t ( *write) (struct file  *,const char  - - user  *,
size - t,loff - t  *);
int ( *ioctl) (struct inode  *,struct file  *,unsigned int,
unsigned long);
    int ( *open) (struct inode  *,struct file  *);
    int ( *release) (struct inode  *,struct file  *);
    … };
```

其中每个驱动程序必须实现的函数主要有:

(1) open()

在 Linux 或应用程序使用设备前都会调用驱动程序的这个方法。在驱动程序中 open() 方法一般用于对驱动程序支持的设备进行初始化。

(2) release()

当 Linux 或应用程序不再使用设备时会调用驱动程序的这个方法。release() 方法主要用于在使用完毕支持驱动程序的设备后,对设备进行关闭操作。

(3) read()

read() 方法主要用于供应用程序或 Linux 从字符设备中读取数据。read() 方法读取的数据会从内核态拷贝到用户态,供应用程序使用[3]。

(4) write()

Linux 或应用程序会调用 write() 方法写多个字节数据到字符设备中。应用程序的数据会首先从用户态拷贝到内核态,然后再传给 write() 方法。

(5) ioctl()

ioctl() 方法向应用程序提供了一些独特的操作,这

些操作不易通过 read()方法或 write()方法来实现[4,5]。

2 I²C 协议以及 RTC 芯片介绍

2.1 I²C 协议

I²C 总线是一种由 Philips 公司开发的两线式串行总线标准,用于链接微控制器及其外围设备。I²C 由数据线(SDA)和时钟线(SCL)构成的同步串行总线,可发送和接收数据。在处理器与被控制芯片之间,芯片与芯片之间进行双向传送[6]。

I²C 总线在传送数据过程中共有三种类型信号,它们分别是开始信号、结束信号和应答信号[3]。

开始信号　SCL 为高电平时,SDA 由高电平向低电平跳变,开始传送数据。

结束信号　SCL 为低电平时,SDA 由低电平向高电平跳变,结束传送数据。

应答信号　接收数据的设备在接收到 8 位数据后,向发送数据的设备发出特定的低电平脉冲,表示已收到数据。

(1) 控制字节。在起始条件之后,必须是器件控制字节,其中高4 位为器件类型识别符(不同芯片类型有不同定义),接着 3 位为地址片选,最后 1 位为读写位,当为 1 时读操作,为 0 时写操作。

(2) 写操作。在写入时,I²C 主控设备先发起起始位(S),抢占总线;然后,发送 7 位设备地址和 1 位 0,表示对设备的写入;接着就是向设备传送数据。

(3) 读操作。在读取时,要改变总线数据的传输方向,其流程如下:I²C 主控设备发起起始位(S),抢占总线;发送7 位设备地址和 1 位 0,表示对设备写入;发送要写入的寄存器地址;再次发起起始位(S);发送7 位的设备地址和 1 位 1,表示对设备读取;从设备读取数据[7]。

2.2 RTC 时钟芯片介绍

在此,采用精密时钟 ISL1208 向系统提供实时时间。ISL1208 是 Intersil 公司的一款低功耗实时时钟集成电路。ISL1208 寄存器可由后备电池供电。在从地址"1101111x"之后,可以访问,可以读或写到地址[00h～13h],可以通过对任意寄存器地址直接以字节写或页面写操作来修改寄存器的内容。

寄存器分成 4 段,它们是:实时时钟(7 B):地址为 00h～06h;控制与状态(5 B):地址为 07h～0Bh;报警(6 B):地址为 0ch～11h;用户 SRAM (2 B):地址为 12h～13h。

由于只使用了其中的实时时钟寄存器,所以只对其中用到的寄存器进行描述。实时时钟寄存器主要包括 SC(秒)、MN(分)、HR(时)、DT(日期)、MO(月)、YR(年)、DW(星期)。

2.3 IXP425 网络处理器对 I²C 协议的支持

IXP425 本身没有基于 I²C 协议的端口,但是 ixp425 有 16 个 GPIO(GPIO0～GPIO15),即通用输入输出端口。这些 GPIO 可以由软件控制输出 0 或 1,并且也可由软件去读取 GPIO 上的状态[8]。虽然 Ixp425 没有基于 I²C 协议的接口,但是可以采用它的 GPIO 来实现 I²C 信号发送与接受。具体采用:

GPIO7	SDA	数据信号发送与接受
GPIO6	SCK	时钟信号发送

3 基于 I²C 协议的 RTC 驱动程序设计

3.1 驱动程序总体框架

整个驱动程序可以分成两个模块,第一个模块是 I²C 协议实现部分,称之为 I²C 协议模块。第二个模块负责与 Linux 内核及应用程序交互,称之为 RTC 驱动模块[9],如图 1 所示。

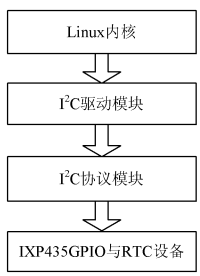

图 1　模块图

3.2 I²C 协议实现模块

该模块中用到的宏与函数:

```
# define I²C - SDA - BIT   7
                          //gpio7 配置为数据信号口 SDA
# define I²C - SCK - BIT   6
                          //gpio6 配置为时钟信号口 SCL
GPIO - DATA - BIT - WRITE - HIGH();
                          //设置 GPIO 状态为高
GPIO - DATA - BIT - WRITE - LOW();
                          //设置 GPIO 状态为底
```

(1) void GpioI²CStart(void)

功能:产生一次 I²C 开始信号。

实现:在 SCL 为高的情况下,SDA 由高电平向低电平跳变。

函数流程图如图 2 所示。

(2) void GpioI²CStop(void)

功能:产生一次 I²C 传送结束信号。

实现:在 SCL 为高的情况下,SDA 由低电平向高电平跳变。

函数流程图如图 3 所示。

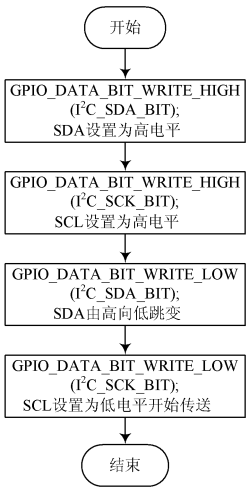

图 2　函数流程图(一)

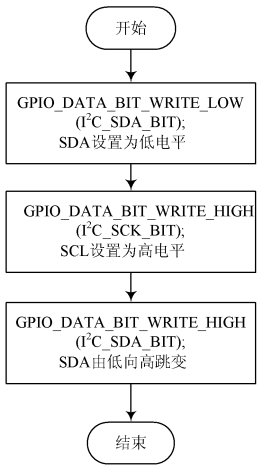

图 3　函数流程图(二)

(3) void GpioI²CSendByte(BYTE ucData)

功能:在 I²C 总线上发送一个字节(8 位数据)。

参数:BYTE ucData　要发送的字节(8 位数据)

实现:现在总线上写一个 8 位数据,然后等待 ACK。函数流程如图 4 所示。

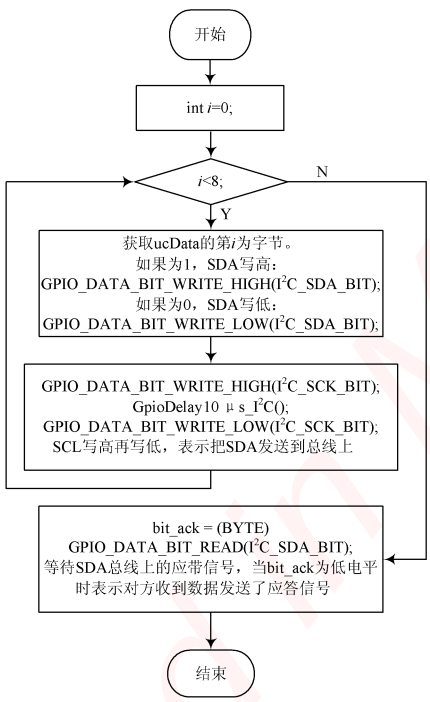

图 4　函数流程图(三)

(4) UINT8 GpioI²CReceiveByte(void)

功能:在 I²C 总线上读取一个字节(8 位数据)。

返回值:在 I²C 总线上读取到的数据。

实现:在总线上等待数据的读取。

函数流程如图 5 所示。

3.3　RTC 驱动实现模块

RTC 驱动模块主要实现 struct file - operations 结构框架。这里主要实现了该框架的 read,write,ioctl,release 几个函数成员。其中主要功能在 ioctl 中实现。

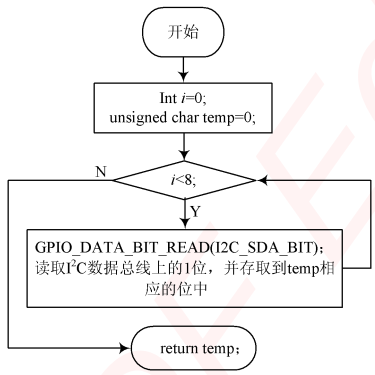

图 5　函数流程图(四)

该结构的定义如下[10]:

```
struct file - operations I²C - rtc - fops =
{
   .open          = I²CRtcDrvOpen,
   .release       = I²CRtcDrvClose,
   .read          = I²CRtcDrvRead,
   .write         = I²CRtcDrvWrite,
   .ioctl         = I²CRtcDrvIoctl,
};
```

由于所有功能都在 ioctl 函数成员中实现,所以以下将主要介绍 ioctl 函数成员。

(1) ioctl 函数用到的数据结构与调用的函数

```
typedef struct
{
   unsigned char second;                    //秒
   unsigned char minute;                    //分钟
   unsigned char hour;                      //小时
   unsigned char date;                      //日
   unsigned char month;                     //月
   unsigned char year;                      //年
}I²C - RTC - DATA - T;
```

该结构表示系统时间。在 I²C 驱动中,主要用来存储从 I²C 总线上读取的 RTC 时间。I²C 协议模块把从 RTC 时钟中读取的时间转化为该类型,然后传给 RTC 驱动模块。RTC 驱动模块再把该类型的系统时间传给用户态。

```
INT32   GpioI²CRead(UINT8 * address,UINT16 rom - address,UINT16 count);
INT32   GpioI²CWrite(UINT8 * address,UINT16 rom - address,UINT16 count);
```

这两个函数用于读和写 RTC 时钟上的寄存器,其中 address 参数为内存区缓冲地址,rom - address 为 RTC 上寄存器地址,count 为要读取的字节数。它们是基于 I²C 协议的调用来实现的。在以下函数中将会用到。

(2) int I²CRtcDrvIoctl(struct inode * inode, struct file *file,UINT32 cmd,unsigned long arg)

功能:这个是 I²C 驱动模块中的 ioctl 函数成员。参数描述如表 1 所示。

表 1　函数的参数描述

参数名	类型	说明
cmd	UINT32	用户态向内核态传入的命令
arg	unsigned long	用户态向内核态传入的参数

cmd 中传入的命令:

RTC - RD - TIME　　读取 RTC 时钟时间
RTC - SET - TIME　　设置 RTC 时钟时间

arg 参数为一个指向 I²C - RTC - DATA - T 类型的指针,用于存取时间。

函数流程如图 6 所示,图中 I2C - RTC - DATA - T gI2cRtcData;全局数据,表示时间。

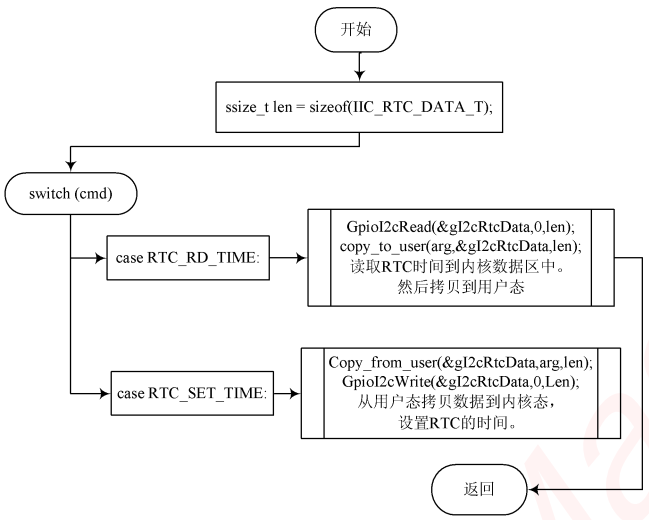

图 6　函数流程图(五)

该函数读取用户传入的命令(cmd),如果用户要获取时间,则调用底层函数读 RTC 寄存器,获取时间。如果用户要设置时间,则调用底层函数写 RTC 寄存器,设置时间。

4　结　语

这里介绍了 Linux 操作系统下基于 I²C 协议的 RTC 驱动程序的开发,主要介绍了 I²C 协议以及 Linux 字符设备驱动程序框架,并在此基础上给出了基于 IXP425 处理器、I²C 协议的 RTC 字符设备驱动程序。该设备驱动程序包括最底层的协议开发,以及上层的驱动程序框架,具有很强的移植性。最终成功开发并实现了该驱动程序。

参　考　文　献

[1] 刘名博,邓中亮. 基于 ARM 的嵌入式 Linux 操作系统移植的研究[J]. 计算机系统应用,2006,44(11):87 - 88.

[2] 刘军芳,胡和智. 基于 ARM 的嵌入式 Linux 操作系统移植研究[J]. 科技信息:学术版,2006,22(5):29 - 31.

[3] 王淑贞. U - Boot 在 S3C2410 上的移植[J]. 微计算机应用,2008,29(4):95 - 97.

[4] 刘广路. 基于 ARM 的 μCLinux 片级移植[J]. 现代计算机,2008,33(1):131 - 133.

[5] 孙纪坤,张小全. 嵌入式 Linux 系统技术开发详解 ——基于 ARM[M]. 北京:人民邮电出版社,2006.

[6] 周立功,陈明计,陈渝. ARM 嵌入式 Linux 系统构建与驱动开发范例[M]. 北京:北京航空航天大学出版社,2006.

[7] 许振山,刘峥嵘. 嵌入式 Linux 系统应用开发详解[M]. 北京:电力工业出版社,2007.

[8] 李驹光,郑秋,江泽明. 嵌入式 Linux 系统技术开发详解 ——基于 EP93XX 系列 ARM[M]. 北京:清华大学出版社,2006.

[9] Daniel P Bovet, Marco Cesa. 深入理解 Linux 内核[M]. 2 版. 陈莉君,冯锐,牛欣源,译. 北京:中国电子出版社,2004.

[10] 陈渝,李明,杨晔. 源码开放的嵌入式系统软件分析与实践[M]. 北京:北京航空航天大学出版社,2004.

作者简介　孟令公　男,1985 年出生,山东枣庄人,电子科技大学在读硕士研究生。主要研究方向为嵌入式 Linux 底层软件开发。

(上接第 31 页)

[8] 汪安民,陈明欣,朱明. TMS320C54xx DSP 实用技术[M]. 北京:清华大学出版社,2007.

[9] 刘鹏飞,韩九强,周挺. 基于多 DSP 的六自由度机器人伺服控制系统研究[J]. 微电子学与计算机,2005,22(8):5 - 7.

[10] 苏小红,陈慧鹏,孙志刚. C 语言大学实用教程[M]. 北京:电子工业出版社,2007.

[11] Zhang Z, Huang Q, Jin Q, *et al*. Kinematics Analysis of a Humanoid Leg with Redundancy Freedom[A]. Proceedings of the IEEE International Conference on Mechatronics and Automation[C]. Piscataway, NJ, USA: IEEE, 2006: 1 080 - 1 085.

[12] 陶龙,张国良,孙大卫. 基于 PC/104 与单片机的仿人机器人控制系统设计[J]. 现代电子技术,2009,32(2):

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

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

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

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

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

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

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