RT Embedded http://www.kontronn.com

基于 PCI总线的 MPEG - 2码流播放卡驱动程序开发

□李乐乐,黄新平,李兵兵
(西安电子科技大学 ISN国家重点实验室,陕西 西安 710071)

摘 要:结合 DriverStudio软件包,详细阐述了 MPEG - 2码流播放卡驱动程序的开发,重点介绍了对硬件资源的访问、DMA数据传输过程和中断处理部分,并简要介绍了驱动程序安装文件的修改。

关键词:设备驱动程序;WDM; PCI总线;DMA; DriverStudio
中图分类号: TN949. 197 **文献标识码**: B **文章编号**: 1007 - 7022 (2005) 17 - 1719 - 04

Driver Development for MPEG - 2 TS Playing Device Based on PCI Bus

□LI Le-le, HUANG Xin-ping, LI Bing-bing
(National Key Laboratory on ISN of Xi'an Electronic Science & Technology University, Shaanxi Xi'an 710071, China)

Abstract: This paper expatiates the development of driver for MPEG - 2 TS playing device with DriverStudio. The emphasis is the access of hardware, DMA transfer and interrupt service routine. The modification of the driver's INF file is also referred.

Key words: device driver; WDM (Windows Driver Model); PCI bus; DMA; DriverStudio

MPEG - 2码流播放卡可作为高清晰度数字电视系统的信号源,它能将计算机硬盘上的 MPEG - 2码流文件传送给后级的高清晰度电视传输系统,对高清晰度数字电视系统的调试具有重要的意义。MPEG - 2码流播放卡是我们自主开发的基于 PCI总线的设备,由于 Windows操作系统限制了应用程序直接操作硬件设备,只能通过驱动程序来读写和控制设备,所以驱动程序在整个系统中具有重要的作用。

MPEG - 2码流播放卡驱动程序提供硬件和应用程序的接口,驱动程序要负责访问硬件资源、响应硬件中断以及应用程序和硬件之间数据的传输。考虑到跨平台的兼容性,驱动程序采用了 WDM (Windows Driver Model)模型;使用 DriverStudio软件包开发,开发效率高;码流播放卡驱动程序支持高速率码流的传输,在码流速率高达 120 Mbps时数据传输正确,其 DMA数据传输过程和中断处理过程具有一定的创新性,其开发方法对其他基于 PCI总线的设备驱动程序开发具有参考价值。

1 驱动程序的结构和开发工具的选择

驱动程序包含许多例程,用来响应操作系统或应用程序的调用。有一些例程是必需的,比如 DriverEntry和 AddDevice和 OnStartDevice,它们与操作系统相配合完成初始化、建立中断连接等任务,使硬件成为操作系统可访问的设备;另外一些例程要根据硬件要实

RT Embedded http://www.kontronn.com

现的功能进行选择,码流播放卡驱动程序除了上述必需的例程之外,还有中断处理例程、延迟过程调用例程、DMA传输等一些特定的例程。

开发驱动程序可以直接使用微软为各操作系统提供的开发软件包 Device Driver Kits(DDK),也可以使用第三方提供的开发工具,如 NuMega公司的 DriverStudio软件包,后者包含的 DriverWorks软件开发驱动的时间短、效率高,并且提供了一系列类,这些类封装了许多更为底层的操作,使得驱动程序开发采用面向对象的框架结构,DriverWorks包含一个代码生成工具(DriverWizard),能快速生成 WDM 驱动程序框架,使开发过程简单,鉴于这些优点,采用 DriverStudio软件包开发 MPEG-2码流播放卡的驱动程序。

2 码流播放卡驱动程序的功能

码流播放卡硬件主要由 PCI9054桥接芯片、可编程逻辑器件 FPGA、存储器 FIFO和 AD9851频率合成器芯片构成,组成框图如图 1所示,PCI9054用于实现 PCI总线和本地用户总线的连接,FPGA作为码流播放卡硬件的逻辑控制模块,FIFO作为缓冲器存储码流数据,AD9851频率合成器芯片根据码流播放速率产生一个时钟控制码流输出的速度,根据驱动程序设置的频率控制字产生时钟信号。

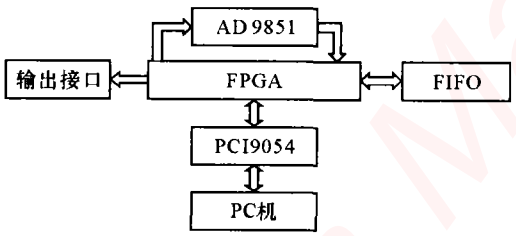

图 1 码流播放卡硬件组成框图

为了使码流播放卡能正常工作,驱动程序要完成以下功能:

(1)从应用程序得到码流的速率,根据此速率设置硬件的 AD9851芯片的控制字,由 AD9851产生时钟作为码流播放时的读取码流数据的时钟信号。

(2)在码流播放过程中,将码流数据写入硬件的外部存储器 FIFO中,为了达到码流速率高达 120Mbps时能流畅地播放,码流数据传输采用 DMA方式。

3)由于外部存储器 FIFO中的数据按照 AD9851的时钟(即码流的速率)连续不断输出,为了保证画面不间断地正常播放,必须保证 FIFO不能空,系统采用如下的方案:当 FIFO半空的时候,由硬件发出中断(称为本地中断),驱动程序响应本地中断并通知应用程序,让应用程序再次调用驱动程序进行 DMA传输。

(4)上述数据传输过程不断进行,直到驱动程序接收到应用程序停止播放的命令时,驱动程序停止硬件工作,释放资源。

3 硬件资源访问

驱动程序首先必须完成 PCI总线地址空间与核心模式驱动程序的虚拟地址空间的映射,然后才可以访问硬件资源。为了完成映射,需要获得硬件的 I/O和 Memory在 PCI总线上的地址,笔者使用 DriverStudio软件包自带的调试工具 SoftICE,首先获得 PCI总线上所有的硬件信息,然后根据硬件设备的 ID找到该设备的信息,可以显示硬件的 I/O和 Memory在 PCI总线上的序号、起始地址和范围等详细的信息。

在码流播放卡驱动程序中对 AD9851写入控制字配置码率以及访问桥接芯片 PCI9054的中断寄存器和 DMA寄存器都采用了 I/O映射方式,利用 DriverWorks软件提供的 KIoRange类,产生该类的一个实例,在设备启动例程(OnStartDevice)中初始化该实例,即可实现对硬件的两个 I/O地址空间的映射,其中一个 I/O地址空间用于访问桥接芯片 PCI9054的寄存器,另一个 I/O地址空间用于访问本地端的设备。如果映射成功,就可以使用 KIoRange类的成员函数访问硬件资源,常用的成员函数有 inb、inw、ind、outb、outw和 outd。以 inb和 outb为例说明如下:

VOID inb(ULONG Byteoffset, PUCHAR Buffer, ULONG Count):从偏移为 Byteoffset的地址开始连续读取 Count个字节数据放在 Buffer缓冲区中。

VOID outb(ULONG Byteoffset, PUCHAR Buffer, ULONG Count):从 Buffer中连续写 Count个字节数据放在偏移地址为 Byteoffset的地方。

其他的函数只是读写数据类型不同,w代表字(16位),d代表双字(32位)。

4 DMA传输和中断处理

DMA传输和中断处理是数据传输的两个重要组成部分,整个数据传输的流程图如图 2所示,图中具有灰色底纹的是数据传输中的中断处理部分。

(1)DMA传输过程

为了实现高速率码流的播放,桥接芯片 PCI9054采用 DMA工作模式,驱动程序也要配合硬件支持 DMA传输方式。DriverWorks提供了 3个类用于实现 DMA操作,KDmaAdapter类用于建立一个 DMA适配器说明 DMA通道的特性,KDmaTransfer类用于 DMA传输控制,KCommonDmaBuffer类用于申请系统提供的

RT Embedded http://www.kontronn.com

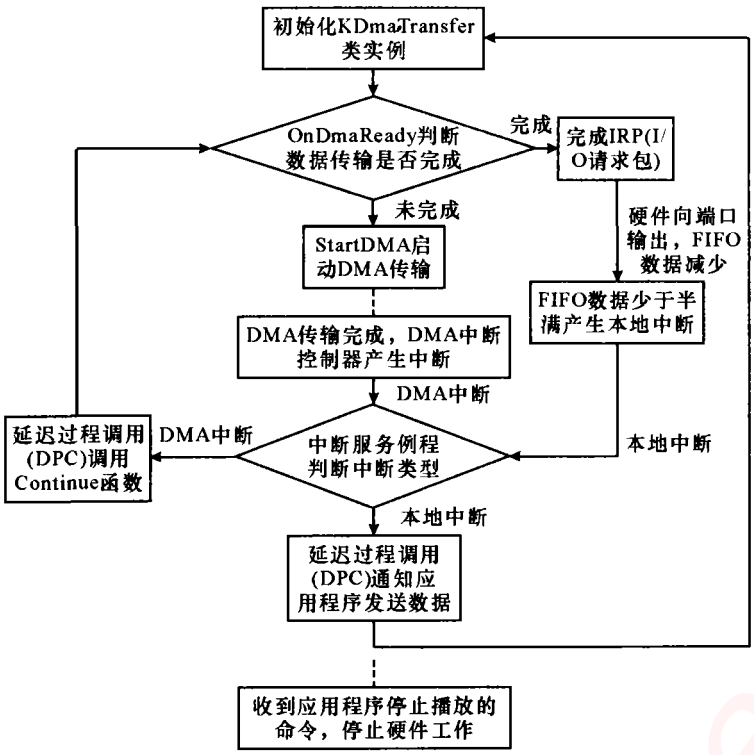

图 2　DMA 传输和中断处理流程

通用缓冲区。

①产生 DMA 适配器对象。在设备启动例程 (OnStartDevice)中创建一个 KDmaAdapter类的实例,并且在适配器对象描述表中正确描述适配器对象。描述表中的信息要与硬件一致,然后用 KDmaAdapter类的成员函数 Initialize初始化该 DMA 适配器。MPEG - 2码流播放卡是总线主控设备,不支持分散 聚集。

②在设备启动例程 (OnStartDevice)中创建一个 KCommonDmaBuffer类实例,调用该类的成员函数 Initialize初始化,需要用到 DMA 适配器,并指定通用缓冲区的大小。

③创建一个 KDmaTransfer类实例,并调用该类的成员函数 Initialize初始化,根据驱动程序使用的内存类型的不同有两种不同的初始化方式。驱动程序使用的内存类型分为“packet”和“common buffer”两种。“packet”形式的驱动程序采用直接 I/O 内存缓冲策略,应用程序将存放数据的源地址或目的地址传入驱动程序,驱动程序用 IRP中的 MDL 信息获取;“common buffer”是通用缓冲区,它是系统分配的一个物理连续内存缓冲区,处理器和设备都可以存取,这种方法适合经常传输数据的总线主控设备适配器,适用于不支持 scatter/gather的设备传输大数据。鉴于硬件设备不支持 scatter/gather,而且每收到一次本地中断就要进行一次 DMA 传输 256 kBytes数据,数据传输频繁,所以码流播放卡的驱动程序中使用的内存类型是“common buffer”。

④编写 KDmaTransfer类实例的回调函数 OnDmaReady,由宏 DMAREADY_CALLBACK 定义。回调函数调用 KDmaTransfer类的成员函数 BytesRemaining,根据其返回值是否为 0 来判断数据传输是否已经完成。如果传输已完成,则调用该类的成员函数 Terminate完成相应的 IRP;如果未完成,则调用该类的成员函数 GetTransferDescriptors,获得当前传输数据的物理地址、传输数据的字节数,然后进入 StartDMA 例程设置硬件的桥接芯片 PCI9054 的 DMA 寄存器,包括 DMA 模式、PCI端的地址、本地端的地址、数据传输方向和需要传输的字节数,开启 DMA 通道,开始真正的数据传输。

(2)中断处理部分

①数据传输完成时 PCI9054的 DMA 中断控制器产生中断 (称为 DMA 中断),驱动程序的中断服务例程 (ISR)做出响应;前面提到了 FIFO 中的数据量少于一半容量时 FPGA 产生本地中断,驱动程序同样会进入中断服务例程,两种中断发生时驱动程序要执行不同的操作,这就要求在中断服务例程中区分这两种中断。两种中断依靠桥接芯片 PCI9054芯片 DMA 中断寄存器不同的比特发生变化来区分,在 ISR例程中读取 DMA 中断寄存器,判断中断类型并设置驱动程序中的一个全局变量标识 LocInt,然后清除本次中断,连接延迟过程调用 (DPC)例程。

②在 DPC例程中根据全局变量标识 LocInt的不同,执行不同的操作:如果是硬件的本地中断 (代表了 FIFO 中的数据量少于一半容量),则通知应用程序发送数据再次进行数据传输;如果是 DMA 传输完成的中断,则调用 KDmaTransfer类的 Continue()成员函数,进而又进入回调函数 OnDmaReady,此时数据传输已经完成,调用 KDmaTransfer类的成员函数 Terminate并且完成相应的 IRP。

5　安装文件的修改

DriverWorks软件生成的工程中包含了一个后缀名为.inf的安装文件,要修改其内容与硬件信息一致。注册表项 HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Class中包含了 Microsoft定义的一系列 GUID和设备类名。我们修改码流播放卡的安

RT Embedded http://www.kontronn.com

装文件,使之加入 MEDIA设备类,在设备管理器中显示为“声音、视频和游戏控制器”。将驱动程序安装文件中 Version Section的 Class和 ClassGUID修改为注册表中 MEDIA设备类的类名和 GUID,并将 ClassInstall/ClassInstall32 Section的 HKR,,Icon,,“”中的数据修改为注册表中该设备类 Icon的数据值,就修改了在设备管理器中设备显示的图标。

码流播放卡驱动程序安装文件中的对应项分别修改为如下内容:

Class=MEDIA

ClassGUID={4D36E96C-E325-11CE-BFC1-08002BE10318}

HKR,,Icon,,"3004"

为了在硬件属性中显示设备的制造商、驱动的提供商和设备名等信息,还要修改安装文件中的[Strings]节。

参考文献:

[1] 武安河,邰铭于,洪 涛. Windows2000/XP WDM设备驱动程序开发[M]. 北京:电子工业出版社,2003.

[2] 张惠娟,周利华,翟鸿鸣. Windows环境下的设备驱动程序设计[M]. 西安:西安电子科技大学出版社,2002.

[3] 蔡祁鹏,王宏远. HDTV信号源中 PCIDMA的 WDM程序开发[J]. 电视技术,2001,(4):17-20.

[4] 姚成虎,王 磊. 细说 Windows 2000设备驱动程序的 INF安装文件[J]. 计算机辅助工程,2003,(4):69-74.

光发射机的使用与维护

□赵 震 (江苏兰丹网络技术有限公司,江苏 镇江 212001)

光发射机是构建有线电视 HFC网络的关键设备,特别是采用 1 550 nm系统的 HFC网络,整个网络只有 1台光发射机,为确保网络的安全可靠运行,光发射机在使用中有以下几个环节需要注意。

1 接地问题

光发射机的激光器属静电敏感器件,1 550 nm的光发射机比 1 310 nm光发射机更为敏感,因此在使用光发射机时必须与接地相连,接地电阻小于 4Ω。有很多机房因对接地体缺乏维护,接地电阻往往较大,这是一个很大的隐患,对接地电阻应定期检测。同时注意光发射机在加电工作前,应先确认机壳和电源插座的接地端可靠接地,以免静电损坏激光器件,并防止机壳带电对人体造成伤害。

2 供电和温度

在安装光发射机时,应确认光发射机的供电电压(特别是进口设备),对于 220 V供电系统进线采用三线制,中线为接地线,为了避免雷击使光发射机损坏,必须在供电系统接上避雷装置,在电网电压不稳定或电压波形较差的地区,应为设备配置专用的交流稳压工作电源,最好是配置 UPS不间断稳压电源系统。

光发射机的激光器寿命受温度影响较大,一般要求光发射机的工作温度小于 50℃(设备的理想工作环境温度为 25℃),在环境温度变化太大或机房环境较差的地区,用户应为设备配置专用的空调设备,以改善设备的工作环境。

3 射频输入电平

为达到较好的系统指标,光发射机在出厂时,调制度已经由厂家设定好并给出一个建议电平,GY/T106-99规定光发射机输入电平为 75～85 dBμV频道。若光接收机输出图像有明显网纹,就可能是光发射机输入电平偏高或输入的射频信号的交互调指标太差,若射频输入信号电平太高就可能损坏激光器,若输入光发射机的射频信号电平太低就会导致光系统链路的载噪比指标太低,因而光接收机输出的信号低,图像有明显的噪点。

另外,光发射机的激光器虽然有反向击穿保护,但还是应尽量避免供电系统意外断电和在光发射机还未供电工作的情况下,将 RF信号直接接到光发射机,因为激光器在正常工作状态下处于正向偏置状态,如光发射机未通电工作,RF信号驱动电平加到激光器上就可能出现反向电压,如果反压超过 1 V就可能导致激光器反向击穿,特别是在设备调试时,应严格遵守先开发射机再加载 RF信号、切断时先去掉 RF信号再关发射机的操作规程。

4 信号输出口

光发射机的输出接口是设备光信号的输出端口,接口类型的常用规格有 FC/APC和 SC/APC两种,在设备正常工作后,此端口有不可见的激光束射出,应避免该端口对准人体特别是眼睛,以免造成意外伤害。

5 输出光功率检测

检测光发射机输出光功率,首先要用质地良好的擦镜纸或医用脱脂酒精棉小心清洗跳纤活动接头。清洗干净的光纤活动连接头接回适配器时,应注意用力适当,以免用力过猛使适配器内的陶瓷破裂;如用脱脂酒精棉清洗,清洗完毕还需等片刻,让活动连接头表面晾干,然后接入光功率计,检测输出光功率是否正常。若光功率偏低,更换跳线后还是偏低,则有可能是机内的另一个连接头或适配器被污染。拆卸适配器应小心操作,以免损伤机内光纤,若还不正常可能是光发射机有故障。

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

Creat by Master PDF Editor

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

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

Linux：

Create Master PDF Editor

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

Created with Master PDF Editor

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

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

ARM：

Created by Master PDF Editor

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

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

Created with Master PDF Editor

16. 基于 UEFI 的 Application 和 Driver 的分析与开发

Programming:

1. 计算机软件基础数据结构 - 算法

Created in Master PDF Editor