基于USB3.0的设备自定义请求实现方法

陈 锴

(中国海洋大学 信息科学与工程学院,山东 青岛 266100)

摘要: 该文主要讨论了一种新方法,实现了主机通过一种新的自定义设备请求,获取存储于USB3.0设备固件中详细的UI信息。从而可使OEM/IHV厂商不必再随其生产的USB3.0设备分发任何特定的安装媒体。

关键词: USB3.0;设备自定义;固件;UI;设备请求

中图分类号:TP31　　文献标识码:A　　文章编号:1009-3044(2013)07-1648-03

A Method to Implement the Device Specific Quest Based on Universal Serial Bus 3.0

CHEN Kai

(College of Information Science and Engineering, Ocean University of China, Qingdao 266100, China)

Abstract: This paper discusses the means of implementing a new method offer for storing device specific users interface information into firmware on a USB3.0 device. The USB3.0 device returns the device specific information to an operating system or other application for responding a host specific device request. Thus, the OEM/IHVs can store device specific UI information into the firmware of a USB3.0 device such that installation media does not require to be distributed with each USB3.0 device.

Key words: USB3.0; Device-specific; Firmware; UI; device request

1 背景

USB(Universal Serial Bus)即通用串行总线,是一种灵活,快速的总线接口。USB技术的出现让IT产业的接口产生了巨大的革命。其主要特征是低成本,同步,高速,双向,高速,可即时连接,是对当今计算机体系结构的一种工业标准扩展,因此支持USB的设备现在已经十分广泛。主机通过USB总线对设备的操作,完成对其连接,配置,使用和断开等功能。随着硬件设备的不断发展进步,更高的传输速度和更大的带宽越来越被人们所重视,由此各界对USB 3.0的需求也愈来愈高。2008年11月发行的USB 3.0通用串行总线(Universal Serial Bus)是最新规范,该规范由英特尔等大公司发起,其最高传输速度可达5 Gb/ s,并且兼容USB 2. 0 及以下接口标准。

2 设备请求介绍

设备请求是USB3.0的重要组成部分。设备的信息和功能都存在它的设备描述符(descriptor)中,要想得到这些信息,或是修改这些信息,就要对描述符进行读写操作,完成这些读写的行为称为设备请求。

设备请求包括标准请求、设备类请求和厂商自定义请求。

1) 标准请求:USB3.0协议定义了一系列所有USB3.0设备都必须支持的请求。它们用于配置一个设备、控制USB接口的状态,还有其他特征。

2) 设备类请求:每个USB3.0设备类都可以定义类的特定请求,除了集线器设备之外,这些请求由设备类的协议说明文档定义,它们并不包含在USB3.0协议的主体部分。

3) 厂商自定义请求:一个USB3.0设备可以支持设备厂商自定义的请求,这些请求和设备厂商对设备的具体实现相关。厂商自定义的请求只有设备和主机端对应的设备驱动程序知道。

所有的设备在设备的缺省控制通道处对主机的请求发出响应。这些请求是通过使用控制传输来发送的,请求及请求的参数通过Setup阶段的数据包发向设备,主机负责设置Setup数据包内的每个域的值,每个Setup包有8个字节,见表1。

3 设备自定义请求的实现

高层主机的应用程序和操作系统厂商,会根据实际应用需要对设备定义一些特定的设备请求与响应。而底层的硬件制造商(OEMs)和独立硬件商(IHVs),即设备生产和发行方,会对主机软件和操作系统的这些额外的自定义设备请求提供支持。虽然USB3.0协议主体中规范了多种类型的设备请求,但并没有对上述的主机自定义请求进行详细说明。

表1 Setup数据包格式

偏移量	域	大小	值	描述
0	bmRequestType	1	位图	请求特征::D7: 传输方向,0=主机至设备 1=设备至主机;D6..5: 种类,0=标准,1=类,2=厂商,3=保留;D4..0: 接受者,0=设备,1=接口,2=端点,3=其他;4..31=保留;
1	bRequest	1	值	特定请求
2	wValue	2	值	字长域,根据不同的请求含义改变。
4	wIndex	2	索引或偏移	根据不同的请求含义改变,通常用于传送索引或偏移。
6	wLength	2	计数	如有数据传送阶段,此为数据字节数。

一般情况下设备安装过程为：USB3.0设备与主机连接后,用户需要安装OEM/IHV提供的光盘或应用程序。安装程序通常包括设备特定的设置和资源,如下所示：

A.设备驱动

B.用户界面(UI),包含图标,字体,图片,标志,帮助页面,通用资源定位符(URL)。

C.详细的说明,设置和资源定位信息文件。

在安装过程中,操作系统通过设备提供的USB标准类和子类信息码,来判断此设备是否为通用设备,或可用默认驱动来控制此设备。通常情况下大部分USB3.0设备都可使用系统默认驱动控制,所以用户就可不必再安装OEM/IHV提供驱动。

随着USB3.0设备大范围的普及,OEM/IHV厂商们更希望操作系统能够自行加载设备某些自定义信息(如,图标,字体,图片,标志,帮助页面,通用资源定位符(URL)),这样即使用户不用安装任何额外的程序或驱动,操作系统也能为设备提供恰当的UI和信息。如果解决了这个问题,将大大简化了设备安装过程,提高了设备的易用性。同样,也降低了OEM/IHV厂商的成本。

本文根据这个问题,提出一种解决方案。即把设备自定义的UI信息存储于USB3.0设备固件中。当主机与设备通信时,设备响应主机的请求,把设备的UI,配置信息传输给主机。设备使用系统默认驱动时,不用安装厂商提供的任何文件,也能完成对设备的正确配置。

如图1示例系统,设备自定义的设置和资源信息都存储于固件中。操作系统或应用程序可以通过主机自定义请求读取这些信息,从而为用户提供恰当的UI和设备信息。

系统由主机与设备组成,两部分通过USB总线相连。主机一般为PC或其他类型计算机。主机中具有一个或多个处理器,存储设备(如内存,硬盘等)。操作系统如(Windows)和各种应用程序会存储于硬盘和内存中,通过处理器运行。主机中还包含操作系统所支持的USB驱动和端口。主机与设备的通信过程为:高层的应用程序对操作系统的系统服务进行操作，然后由系统服务配置低层的通信细节后完成通信,最后把设备返回的信息反馈给应用程序。

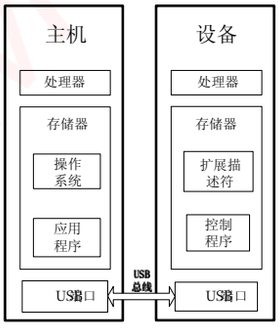

图1 系统整体结构

图1中的设备可以是硬盘,闪存,数码相机,鼠标等等。设备中也包含了一个或多个处理器,各种类型的存储器和USB总线端口。其中存储器存储了控制设备与主机通信的控制逻辑,并可由处理器运算后执行。

设备通过USB总线,对主机的请求发出响应。这些请求是通过使用控制传输来发送的,请求及请求的参数通过Setup包发向设备。设备接收到Setup数据包后,响应主机返回数据。

图1设备中定义了一种新的设备自定义描述符–扩展属性描述符。描述符中存储着设备自定义的UI信息。这些UI信息的格式是由操作系统厂商所定义的。OEM/IHV厂商依据规范格式可把UI信息(如:图标,字体,标志,URL等)写入扩展属性描述符,并存储于设备存储器,即固件中。

此系统可以实现:设备使用前,主机发送自定义请求给设备,命令返回相应的设备自定义UI信息,然后设备响应主机请求返回数据。最后主机根据设备返回的UI信息配置操作系统,提供相应的UI及组件。

举例说明:普通数码摄像机和PC相连后,Windows系统通常只会识别为逻辑磁盘设备(只显示默认卷标),且只提供磁盘存储设备的UI。但是,如果设备加入了扩展属性描述符,而且经过主机自定义请求后获得此描述符,操作系统就可以根据描述符中的数据

获得有关数码相机的详细信息。这些信息可使Windows为数码相机提供更加恰当的标识(如,卷标显示Digital Camera)和相机相关的UI,而不仅仅只当成磁盘存储设备。

本文所提出的方法,即为各式USB3.0设备定义了可嵌入在USB3.0标准协议中,一种新的设备请求-主机自定义设备请求。此主机自定义设备请求可由操作系统厂商或应用程序规范,设备厂商能够从底层支持这些请求。

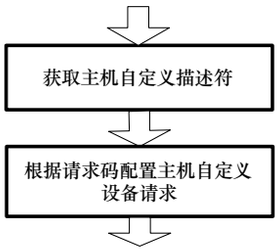

图2 主机获得设备自定义请求码的主体流程

如图2所示,在初始阶段主机发送USB3.0协议规范的设备请求,命令返回设备自定义请求码。顺利返回之后主机根据收到请求码,配置下一阶段发送的主机自定义请求,从而获取扩展属性描述符。

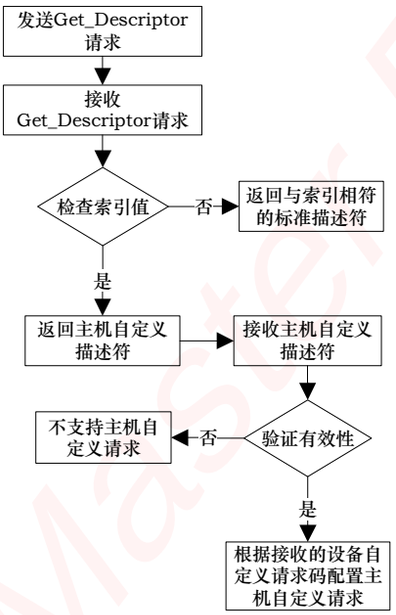

图3 主机获取设备自定义请求码详细流程

图3 是图2实现的具体过程。由此主机在控制传输阶段,首先通过Setup数据包发出USB3.0协议规范的标准设备请求Get_Descriptor,发送请求码bReques=Get_Descriptor到设备。控制传输中各个域设置如下:

bmRequestType=10000000,表示从设备到主机,标准,规范,接受者为设备的请求类型。

bRequest=Get_Descriptor,主机读取设备指定的描述符。

wValue=03F9h,高字节03h表明了主机所读取的描述符的类型值为字符串描述符。低字节F9h中描述符的索引值是常数(可设任意值),其含义为此请求为主机自定义设备请求。

wIndex=0。

wLength=08h,指定主机自定义请求描述符返回的字节数,本例中为08h。

data 返回的主机自定义请求描述符具体数据。

设备接收到主机发出的Get_Descriptor请求后,对此请求中的wValue域的低字节,即描述符索引值进行判断。如果其值不等于所对应的自定义描述符的预定义索引值(即F9),则设备返回与索引值相符的标准描述符。若是设备中具有索引值等于F9的描述符,即主机自定义描述符。则把此描述符返回给主机。此描述符所含以下域:

bLength 描述符长度,本例中为08H。

bDescriptorType 描述符类型,本例中为字符串类型。

bSign 确认描述符有效性,可标识主机自定义设备请求版本型号。例如MS30标识此描述符对应Windows操作系统的自定义设备请求,版本为3.0。

bVendorCode 存储于设备中和主机自定义请求相关联的设备自定义请求码。

主机接收到自定义描述符后,根据bSign域中的值,进一步检查版本型号验证描述符的有效性。如果出现错误则认为设备不支

(下转第1703页)

参考文献：

[1] 栾峻峰.水墨画仿真系统的设计和实现[D].济南:山东大学,2009.
[2] 曹毅.基于图像的中国水墨画绘制方法的研究[D].长春:吉林大学,2012.
[3] 孙美君.中国水墨画的设色扩散与风格化绘制研究[D].天津:天津大学,2009.
[4] Cassidy J Curtis, Sean E Anderson, Joshua E.Seims.et.al. Computer-generated watercolor[C].In Proc.ACM.SIGGRAPH, LosAngeles, US, Aug.1997, 421-430.
[5] Guo Qinglian,Tosiyasu L,Kunii. Modeling the Diffuse Paintings of 'Sumie',Modeling in Computer Graphics,1991:329-338.
[6] Nelson Sin, Hang Chu, Chiew-Lan Tai, Real-Time Ink Dispersion in Absorbent Paper. ACM Transactions on Graphics,2005,24(3):504-511.
[7] Strassmann S. Hairy brushes. Proc. of SIGGRA PH'86,1986:225-232.
[8] Chua Y. Bezier brush strokes. Computer Aided Design.1990, 22(9):5505.
[9] Guo Q, Kunii TL. Modeling the diffuse painting of sumie, IFIP Modeling in Computer Graphics, 1991.
[10] Pahm B. Expressive Brush Strokes. Graphical Models and Image Processing, 1991,53(1):1-6.
[11] 石永鑫,孙济洲,张海江,等.基于粒子系统的中国水墨画仿真算法[J].计算机辅助设计与图形学学报,2003,15(6): 667-672.
[12] 王秀锦,孙济洲.基于渗流力学的水墨画仿真研究[J].系统仿真学报,2008,20(10):2614-2619.
[13] 康丽锋.非真实感水彩画的研究与模拟[D].辽宁:辽宁师范大学,2009.
[14] Jos S. Stable Fluids[C].In Proceedings of SIGGRAPH[M]. New York: ACM Press, 1999: 121-128.
[15] Jos Stam.Real-Time Fluid Dynamics for Games[C]. Proceedings of the Game Developer Conference, March 2003.
[16] 赵杨,杨剑兰.基于流体动力学的水墨画绘制算法研究与设计[J].辽宁大学学报(自然科学版),2012(2).

(上接第1650页)

持主机的自定义请求,从而不会进行下一步操作。反之验证无误,则主机识别出设备支持自定义设备请求。读取bVendorCode域中的设备自定义请求码。随后,主机根据此设备自定义请求码,配置下一阶段的主机自定义请求。主机自定义请求中各个域的含义和数值,不再遵循与表1中的标准设备请求的规范。其实际含义由系统厂商所自定义。

从而接下来的主机控制传输中各个域值设置成:

bmRequestType=1100001,表示此次请求是设备到主机,厂商自定义,接受者为设备的请求类型。

bRequest=bVendorCode域中相应的值,指明主机所要读取的设备自定义描述符。

wValue和wIndex域中根据不同请求含义不同。

wLength域表示有多少字节的数据返回。

由此,对主机自定义请求配置完毕后,通过USB3.0总线发送到设备,随后设备响应主机自定义请求,返回设备固件中所存储的自定义描述符。如图1所搭建系统中的扩展属性描述表。从而操作系统获取描述表中有关此设备的UI信息。

4 小结

对比于传统的USB3.0设备使用前,用户必须使用OEM/IHV厂商为USB3.0设备所提供的特定安装程序,才能提供恰当的UI信息而言。该文所论述的方法在兼容USB3.0协议基础上。通过实现一种新的主机自定义设备请求,直接从USB3.0设备中获取其自定义的UI信息配置操作系统或程序。从而OEM/IHV厂商,只需在USB3.0设备中加入对应于主机自定义设备请求的自定义描述符,就可使用户无需安装任何文件,系统也能为其提供相应的UI,提高了USB3.0设备的灵活性,易用性。为OEM/IHV厂商提供了方便,降低了成本。

参考文献：

[1] Intel,Microsoft,NEC,Universal Serial Bus Specification 3.0.2008-11.
[2] Suguru Ishii.Peripheral Device And Method Of Connecting Peripheral Device With Host Device,United States Patent Application,2011.
[3] Moore,Terrill M. Usb Hub Supporting Unequal Numbers Of High-Speed And Super-Speed Ports, United States Patent Application,2011.
[4] ADERSON Don .USB系统体系,2007.

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
51. IEEE 1588 协议在工业以太网中的实现

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
32. 军事指挥系统中 VxWorks 下汉字显示技术

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
38. 嵌入式 Linux 实时性方法
39. 基于实时 Linux 计算机联锁系统实时性分析与改进

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
19. 基于 PowerPC 的控制器研究与设计

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
25. 中断调用方式的 ARM 二次开发接口设计
26. ARM11 嵌入式系统 Linux 下 LCD 的驱动设计
27. Uboot 在 S3C2440 上的移植

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
24. 基于 X86 平台的嵌入式 BIOS 可配置设计

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