基于嵌入式 Linux的网络设备驱动设计与实现

管秋梅,胡仁杰
(东南大学电气工程系,江苏省南京市 210096)

摘 要:主要介绍 Intel PXA255平台网络设备驱动的软硬件设计和实现方案,分析了嵌入式 Linux下网络设备驱动的一般特性与编程要点,阐述了网络设备驱动如何处理数据的发送、接收、超时等相应事件,并简要介绍了网卡驱动与上层协议之间的交互。分别利用系统测试和应用测试验证该设计方案,结果表明网卡驱动能够很好地实现网络包数据的传输,达到较快的速度和较高的稳定性。

关键词:信息终端;网卡驱动; Intel PXA255; Linux;嵌入式系统
中图分类号: TP334.7

0 引 言

随着电子信息技术的发展,嵌入式系统应用到了社会生活的各个角落。后 PC时代是嵌入式和网络化的时代,将分散在各处的嵌入式设备连接到网络成为近年来以及未来的发展趋势,随时随地都能通过网络去控制这些智能设备,在多种嵌入式设备之间进行数据通信,例如信息终端、信息家电等。用户可以通过信息终端从网络服务商提供的服务中搜索并浏览有用的信息,实现全面交流和互动,实现新型的信息金融终端的概念,同时,用户也可享受到网上金融信息、网上购物、视频会议等其他服务。

本文提出了一种基于 Linux 2.4操作系统平台和 Intel PXA255处理器的网络设备驱动设计方案,分析了其软硬件实现以及 Linux下网络设备驱动的特性。

1 网卡设备

本方案使用的 CS8900A是一款低功耗的网络控制器,包含一块片上 RAM、10BASE-T接收和发送滤波器、ISA总线接口,提供了多种性能和配置选项,它独立的包页(packet page)体系结构能自适应改变网络通信模式和可用的系统资源,提高了系统效率。该控制器支持 IEEE 802.3的 10 Mbit/s模式,全双工单片以太网解决方案。主要模块有:一个直接 ISA总线接口;一个 802.3 MAC引擎,处理以太网帧发送和接收的各个方面;集成 4 kB的页可缓冲存储器,将发送和接收的帧完全缓冲在网卡中;一个串口 EEPROM接口;完全模拟前端 10BASE-T(10 Mbit/s)和 AUI(粗缆网卡接口)。

2 Linux驱动

Linux驱动分为字符设备、块设备和网络设备驱动。字符设备是能够像字节流一样被访问的设备,大多数字符设备只能被顺序访问。块设备驱动程序提供面向块的设备的访问,这种设备以随机访问的方式传输数据,并且数据总是具有固定大小的块,这些块中包含了 2的几次幂字节的数据,硬盘就是典型的块设备。

在 Linux中,字符设备和块设备都是通过文件系统节点被访问,块驱动程序除了向内核提供与字符驱动程序相同的接口外,还提供了专门面向块驱动设备的接口。

网络接口在系统中的角色与一个已挂装的块设备非常相似。块设备将自己注册到 blk_dev数组以及其他内核结构中,然后通过自己的 request函数在发生请求时"发送"和"接收"数据块。同样,网络接口也必须在特定的数据结构中注册自己,以便在与外界交换数据包时被调用。但是,块设备接口与网络数据包发送接口之间存在不同。普通的文件操作对网络接口来说没有任何意义,因此,Unix的"所有东西都是文件"这一思想无法应用于网络接口。这样,网络接口存在于它们自己的名字空间中,同时导出一组不同的操作。同一网络接口可以由几百个套接字同时复用。块设备驱动只对来自内核的请求做出响应,而网络驱动程序却异步地接收来自外界的数据包。

网络驱动程序同时必须支持大量的管理任务,例如设置地址、修改传输参数以及维护流量和错误统计等。网络驱动程序的 API反映了这种需求。此外,网络驱动程序与内核其余部分之间每次交互处理的是一个网络数据包,因此,驱动程序无需关心协议问题。

3 实现方案

3.1 硬件连接

硬件平台使用 Intel PXA255处理器作为主机,工作频率 400 MHz,网卡的晶振为 20 MHz,网卡的数据线和地址线通过缓冲后与 CPU直接相连,硬件电路如图 1所示。

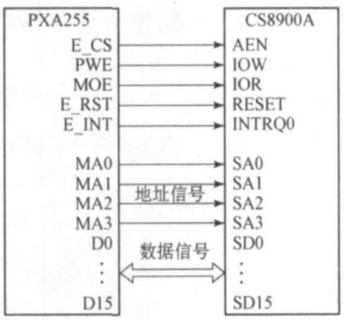

图 1 硬件连接

网卡的读写使能引脚 IOR/IOW信号主要由 CPU采用 Flash存储器的方式给出,与 PWE和 MOE两个引脚分别连接,其有效时间由 CPU内部的寄存器配置而得。

网卡内部寄存器使能引脚 AEN由 CPU片选脚 E_CS组合得出,而网卡复位引脚则通过 CPU的一个 GPIO口给出。

从电路中可以看出只使用了 4根地址线,因此,对网卡中的寄存器的访问采用 packet page寄存器间接寻址,即通过地址线先对 packet page访问,将欲访问的寄存器 R的偏移地址送到 packet page寄存器中,然后通过数据线去读或者写此寄存器。

3.2 软件实现

网卡驱动可看做是连接网络层与实际网络设备的一个软件层。在网卡设备驱动中,通过维护一个名为 net_device结构的 dev指针来实现上层对驱动程序的访问。这个结构包括了对管理网络设备所必须完成的操作,包括设备的打开、关闭、发送数据、接收数据、获得状态、设置网卡地址、处理超时等。

以下主要介绍这些操作的实现,包括设备的初始化和注销、对中断事件的处理。

3.2.1 设备初始化

由于需要通过 Memory模式访问网卡设备,因此需要配置存储器控制寄存器 VLIO(Variable Latency)模式。关联设备的初始化探测函数 cirrus_probe,在 cirrus_probe函数中主要是填充该设备的 Dev结构。网络接口探测方式不同,所以不能在编译阶段进行 Dev结构的设置。这样从 Dev→init返回时,Dev结构中就应该填充正确的值。Dev结构中包含相关的网卡操作函数的指针,为上层调用网卡驱动提供方法。内核提供了 ethr_setup函数,可处理以太网的某些缺省设置。在探测函数中还要分配 net_device的 priv字段,该字段所指向的数据项通常包含接口上的统计信息。用户可以在任何时刻通过调用 ifconfig获得统计数据,向内核注册网络设备驱动程序。

这里简要介绍网络设备驱动中两个重要的结构体:Linux内核提供的统一网络设备结构 net_device,定义了系统统一的访问接口,此结构体位于网络驱动层的核心地位;net_device_stats结构体用来记录网络驱动发送和接收数据量的统计信息,例如发送包的数目以及丢失包的数目。当应用程序需要获得接口的统计信息时,将调用该方法。

3.2.2 设备的注销

设备的注销过程与设备初始化的过程相反,释放网卡的地址空间及网卡驱动使用的其他相关资源,注销网络设备的 Dev结构。

3.2.3 中断处理

网卡工作有轮循和中断两种模式。轮循模式会大量占用 CPU资源,影响嵌入式系统性能,因此这里选择中断模式。有 5种情况会触发中断,每种情况对应不同的中断寄存器,中断发生会将某一模式寄存器的相应位置位,并且把这种模式寄存器中的内容映射到中断状态寄存器。当有中断事件发生后,进入中断例程,如图 2所示,首先读取中断状态寄存器的内容,然后根据其中的信息判断是哪种情况触发的中断,最后进入相应的分支进行处理。

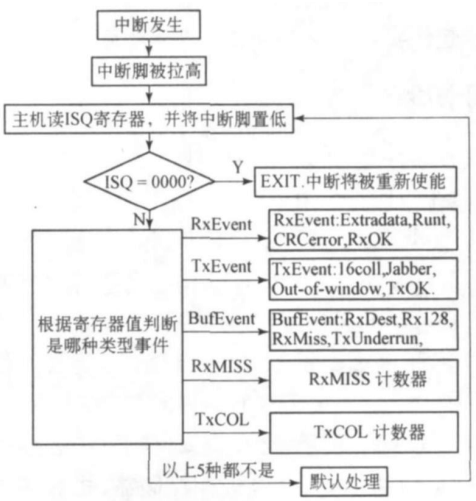

图 2 中断处理流程

3.2.4 对 5种中断状态的处理

1)接收帧

如图 3所示,当网卡接收到数据就会触发一次中断,在中断例程中调用接收包的函数 cirrus_receive。

首先读取接收的状态和长度,使用 dev_alloc_skb分配一段足够大小的缓冲区,得到该包的网络协议 ID号,在一个申请好的 sk_buff的缓冲区中保留一块空间,这个空间一般是用做下一层协议的头空间;接着将网卡接收缓冲区中的数据读到 skb缓冲区中,再使用 netif_rx通知上层协定有新的封包传入。当一个封包传送完成后,必须将缓冲区释放。

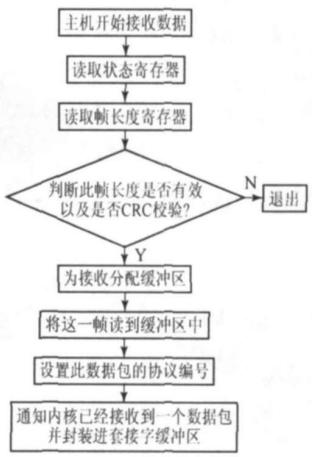

图 3 接收帧流程

2)发送帧

当主机要发送数据时调用 hard_start_xmit方法将数据放入外发队列,而此方法关联的函数就是 cirrus_send_start。内核处理后的每个数据包位于一个套接字缓冲区(struct sk_buff)结构,核心处理的每个包包含在一个套接字缓冲区结构,输入输出缓冲区都是 sk_buff结构的链表,传递给 hard_start_xmit的套接字缓冲区含有物理包,它具有传输层的包头。接口不需要修改被发送的数据。实际的硬件接口是异步传输数据包的,硬件中有缓冲区以保存要外发的数据包,但是此缓冲空间非常有限。如图 4所示,在此函数中首先通知上层表明硬件缓冲区已经用完不要再送封包下来,然后向网卡发命令表示要开始发送数据并且指明发送数据的长度,接着读 BusST寄存器判断当前网卡状态,如果主机中的发送缓冲区可用,则将 SKB中数据发到网卡的缓冲区中,最后释放 SKB。如果网卡将数据发送成功,则会触发一次中断,在中断处理例程中将 net_device_stats结构体中的 tx_packets(表示发送的包的个数)元素递增并通知上层可继续送包下来。

3) BufEvent

Buffer事件 当 RxMiss置位表示由于数据从缓冲区中搬移到主机速度较慢而丢失了一些接收的帧,读寄存器获取丢失的包的数目。当 TxUnderrun置位表示在帧结束前网卡运行已过时。改变网络状态结构体中对应元素的值,接着通知上层可往下发送包数据。

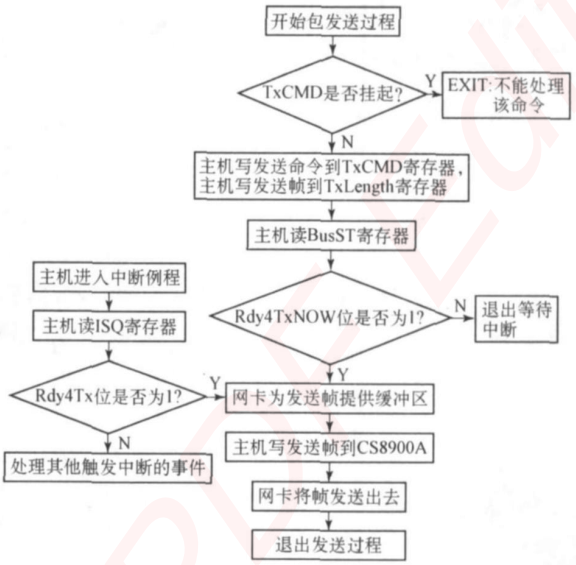

图 4 发送帧流程

4) TxCOL

当传输出现冲突错误时,通过读寄存器值得到当前冲突的个数,加到统计结构体中的对应元素值上。

5) RxMISS

读寄存器获取丢失帧的个数。

3.2.5 传输超时的处理

驱动程序要处理硬件不能正确响应的情况,比如中断丢失或者接口工作异常等。网络系统使用大量定时器控制的多个状态机之间流转来检测传输超时,因此本驱动程序中无需自己检测超时问题而只是通过 net_device结构的 watchdog_timeo字段设置超时周期为 10 ms,对于一般的传输超时,此值能满足要求。当检测到系统时间超过设备传输开始的时间至少 1个超时周期,网络层将调用 cirrus_transmit_timeout方法,解决超时所要做的工作,主要是在统计信息中标记该错误,同时调用 netif_wake_queue重新启动传输队列。

3.2.6 测试结果

网络设备驱动测试主要从底层硬件测试和上层网络测试两个方面进行。底层硬件测试使用如下方法:在内存中开辟两块区域 M1、M2,并按照网络包的格式设置初始值,将 M1中的数据通过网卡发送出去,但是以回环方式,最终还是由主机端接收到,存放到 M2中,再将 M2中的数据通过网卡发送出去,并接收回来。经过多次循环,比较最终的数据包和先前设置的数据包是否一致,经过此项测试表明在最底层数据发送和接收无误。上层网络测试使用如下方法:目标机和 PC端通过网线相连,目标机端开启 pure-ftp服务,主机端使用 FlashFXP软件,从主机分别传送不同大小(从 0 MB到 20 MB)文件到目标机,再从目标机传送

到主机,执行多次比较传送文件是否有误,并计算网络传送速度。测试结果表明网卡驱动能较好地实现网络应用的功能,速度达到 2 Mbit/s～5 Mbit/s。

4 结束语

本文提出了一种基于嵌入式 Linux的网卡驱动实现方案,并介绍了 Linux下网络设备驱动的一些特性。此方案已应用到基于 Intel PXA255处理器的金融信息终端研发中,使用和测试表明此方案稳定可靠,也可为其他嵌入式系统的网络设备开发提供一定的参考。

参 考 文 献

[1] RUBINIA, CORBET J. LINUX设备驱动程序 [M]. 魏永明,骆刚,姜君, 译. 2版. 北京:中国电力出版社, 2002: 346-393.

[2] 毛德操, 胡希明. Linux内核源代码情景分析 (下册) [M]. 杭州: 浙江大学出版社, 2001: 119-606.

[3] BOVET D P, CESATIM. 深入理解 Linux内核 [M]. 陈莉君,冯锐,牛欣源, 译. 北京:中国电力出版社, 2004.

[4] 毛德操, 胡希明. 嵌入式系统 采用公开源代码和 StrongARM/XScale处理器 [M]. 杭州:浙江大学出版社, 2003.

[5] Linux操作系统网络驱动程序编写 [EB/OL]. http://fanqiang.chinaunix.net/a4/b7/20010419/122928.html

[6] 李卓桓,瞿华, 等. Linux网络编程 [M]. 北京:机械工业出版社, 2000.

管秋梅 (1977-),女,硕士研究生,主要研究方向为电路与系统。

Design and Implementation of Network Device Driver Based on Embedded Linux

GUAN Qiumei, HU Renjie

(Southeast University, Nanjing 210096, China)

Abstract: This paper introduces the design and implementation of network device driver based on Intel PXA255, and analyses the general characteristics and programming methods of network device driver on embedded Linux. The paper describes how to deal with corresponding events such as transmitting, receiving, timeout and introduces the interaction between driver and upper layer user protocol. Finally, both system test and application test are used to verify the scheme, the result of which indicates that the driver supports data communication very well, thus obtaining fast speed and high stability.

Keywords: information terminal; Network device driver; Intel PXA255; Linux; embedded system

Analysis and Implementation of Simple Microcomputer Based on FPGA

HU Yuanwang[1], YE Pinju[2]

(1. Department of Electronic and Information Engineering, Changzhou College of Information Technology, Changzhou 213164, China;
2. Department of Computer Science and Technology, Changzhou College of Information Technology, Changzhou 213164, China)

Abstract: This paper presents three methods on how to design and implement a simple computer based on FPGA. Because the structure and operation of a simple computer is usually abstract, it's hard for people to understand. To improve this kind of situation, three approaches including bus wire, multiplexer and functional description, which can implement simple computers respectively based on FPGA, are introduced and analyzed. In addition, it is good for mastering EDA and designing ASICs in the future. Finally, the FPGA implementation are provided, and many experiments were carried out and desired results were obtained.

Keywords: microcomputer; FPGA; VHDL

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

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

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

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

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

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

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

Programming:

1. 计算机软件基础数据结构 - 算法