基于IPv6的Socket通信的实现

Realization of Socket Communication based on IPv6

周细义　刘利强

(湖南理工学院　长沙　414006)

【摘　要】随着计算机网络的发展，IPv6（Internet Protocol Version 6）必然会取代IPv4，开发支持IPv6的网络应用程序已经迫在眉睫。介绍了IPv6下Socket地址结构和IPv6下Socket函数，给出了在Windows环境下用Visual C++ 6.0实现Socket通信的方法，对今后开发基于IPv6的网络应用程序会有极大的帮助。

【关键词】IPv6，Socket，网络编程，Visual C++ 6.0

中图分类号：TP393　　**文献标识码**：A

ABSTRACT　With the quick development of Internet computer network, IPv6 (Internet Protocol Version 6) certainly can replace IPv4, developing and supporting network applications program of IPv6 extremely urgent. This article introduced socket address structure and Socket function of IPv6, provide method of socket programming under Windows with Visual C++ 6.0, the network applications program based on IPv6 can have very big help to developing.

KEYWORDS　IPv6, socket, network programming, Visual C++ 6.0

随着网络的飞速发展，目前的IPv4协议在许多方面已经显得不太适应，如IPv4地址严重不足、缺乏服务质量(QoS)控制、安全性差等问题。因此，为了满足人们对地址空间、性能以及安全性等方面的新需求，Internet工程任务工作小组(IETF)的IPng工作组确定了IPng的协议规范，并称之为“IPv6”[1]。IPv6是为适应未来对于网络基础设施的数量和质量的需求而设计的下一代互联网协议，随着IPv6协议的不断发展和完善，现在也越来越为人们所熟悉，逐渐被主流的操作系统和网络硬件设备所支持，相应的协议和标准还在不断的制订、讨论和变化中，这些都需要我们积极地参与研究、开发和实践。本文讲述了IPv6协议在Windows XP系统下的安装和配置过程，并在此基础上给出了基于IPv6协议的网络应用程序设计方法。

1　IPv6地址及相关数据结构

1.1　IPv6地址

在IPv6中，每个地址长度为128位。IPv6使用冒号十六进制记法，将128位写成8个用冒号隔开的16位的整数，每个16位的整数用4个十六进制数字表示。例如，用点分十进制表示的128位数为：

104.230.140.100.255.255.255.255.0.0.17.128.150.10.255.255 可用冒号十六进制表示为：

68E6:8C64:FFFF:FFFF:0:1180:96A:FFFF

冒号十六进制记法可以允许零压缩，即一连串连续的零可以为一对冒号所取代，例如：

FF05:0:0:0:0:0:0:B3

可以写成：

FF05::B3

IPv6将128位地址空间分为两大部分。第一部分是可变长度的类型前缀，它定义了地址的目的[2]。第二部分是地址的其余部分，其长度也是可变的。IPv6的地址类型前缀的分配如表1所示。

表1　IPv6的地址空间分配表

类型前缀(二进制)	地址的类型	占地址空间的份额
0000 0000	保留(与IPv4兼容)	1/256
0000 001	保留给NSAP地址	1/128
0000 010	保留给IPX地址	1/128
001	可聚合的全球单播地址	1/8
1111 1110 10	本地链路单播地址	1/1024
1111 1110 11	本地网点单播地址	1/1024
1111 1111	多播地址	1/256

1.2　IPv6相关数据结构

在Winsock中，IPv6使用128位地址，也定义了本身的地址结构sockaddr_in6和in6_addr。

```
struct sockaddr_in6 {
u_char    sin6_len;    //结构长度
u_char    sin6_family;    //必须是AF_INET6
u_int16_t    sin6_port;    //传输层端口号
u_int32_t    sin6_flowinfo;    //IPv6拥塞级别
```

与流信息

```
struct in6_addr sin6_addr;    //128 位的 IP 地址
u_int32_t sin6_scope_id; //IPv6 的接口范围
}
```

其中, sin6_len 表示整个结构的长度, sin_family 表示地址所属的协议族, 对于 IPv6 地址来说, 始终为 AF_INET6, ssin6_flowinfo 包含 IPv6 流信息, in6_addr 标识一个128 位的IPv6 地址。in_addr6 结构如下所示。

```
struct in6_addr {
u_int8_t u6_addr8[16];    //IPv6 地址
}
```

2 IPv6 的Socket API函数

IPv6 的Socket API函数中一部分沿用了 IPv4 的 Socket API 函数, 也新增了一些 IPv6 专用的 Socket API函数, 这些Socket API函数如表2 所示。

表2 IPv6 的Socket API函数

IPv6 函数	功能说明
inet_ntop()	字符串地址转为 IP 地址
inet_pton()	IP 地址转为字符串地址
getipnodebyname()	由名字获得 IP 地址
struct hostent * getipnodebyaddr()	IP 地址获得名字
getaddrinfo()	获得全部地址信息
getnameinfo()	获得全部名字信息
socket()	建立Socketbind()
Socket 与地址绑定	send() 发送数据(TCP)
sendto() 发送数据	(UDP) receive()
接收数据(TCP) recv()	接收数据(UDP) accept()
接收连接 listen()	网络监听

Getaddrinfo() 的函数原形为:

int getaddrinfo (const char * hostname, const char * service, const struct addrinfo * hints, struct addrinfo * * res)

其中hostname 可以是主机名地址或者是 IPv6 的地址; service 可以是服务名或十进制端口号, 这给我们的编程也带来了很大的方便; Hints 相当于一个过滤器, 只有符合Hints 结构的内容才会返回到 res 指针中[3]。

socket() 函数原形为: int socket (int domain, int type, int protocol)

其中第一个参数确定协议族, 如符号常数AS_NS 表示用Xerox 网络服务地址族, 符号常数AF_INET6 表示用 IPv6 协议; 第二个参数指明了想使用的通信服务类型, 如符号常数SOCK_DGRAM 表示数据报, 用 SOCK_STREAM 表示字节流; 第三个参数指明此 Socket 请求使用的协议, 如符号常数IPPROTO_TCP 表示用TCP 协议, 用 IPPROTO_UDP 表示用UDP 协议。

listen() 函数的函数原型为: int listen(int sockfd, int backlog)

其中 sockfd 是 socket 函数返回的 socket 描述符; backlog 指定在请求队列中允许的最大请求数。

accept() 函数的函数原型为: int accept(int sockfd, void * addr, int * addrlen) 其中 sockfd 是被监听的 socket 描述符; addr 通常是一个指向 sockaddr_in 变量的指针, 该变量用来存放提出连接请求服务的主机的信息; addrten 通常为一个指向值为 sizeof(struct sockaddr_in)的整型指针变量。

connect() 函数原型为: int connect(int sockfd, struct sockaddr * serv_addr, int addrlen)

其中 sockfd 是 socket 函数返回的 socket 描述符; serv_addr 是包含远端主机 IP 地址和端口号的指针; addrlen 是远端地址结构的长度。

send() 函数原型为: int send(int sockfd, const void * msg, int len, int flags);

其中 sockfd 是你想用来传输数据的 socket 描述符; msg 是一个指向要发送数据的指针; len 是以字节为单位的数据的长度; flags 一般情况下置为0。

recv() 函数原型为: int recv(int sockfd, void * buf, int len, unsigned int flags);

其中 sockfd 是接受数据的 socket 描述符; buf 是存放接收数据的缓冲区; len 是缓冲的长度; flags 也被置为0。

3 Windows XP 下 IPv6 协议的配置

3.1 安装 IPv6 协议

在命令提示符下运行ipv6 install。

3.2 配置 IPv6 地址

①在命令提示符下运行netsh interface ipv6 add address“本地连接”3ffe:3211::2, 该命令设定客户端静态的 IPv6 地址为3ffe:3211::2。

②运行 netsh interface ipv6 add route 3ffe:3211::/64“本地连接”, 该命令设置前缀的路由 3ffe:3211:0:0。

③运行netsh interface ipv6 add route“本地连接”3ffe:3211::254, 该命令设置客户端的默认网关为

3ffe: 3211: : 254。

④运行 netsh interface ipv6 add dns“本地连接” 3ffe: 3211: : 1, 该命令设置客户端DNS 服务器地址为 3ffe: 3211: : 1。

4 Socket 通信的实现

在Windows 下编写网络应程序, 基本上都是利用 Socket 套接字进行数据通信, 分为服务器端套接字和客户端套接字两个设计部分[4], 关键代码如下:

4.1 服务器端程序

```
WSAStartup(MAKEWORD(2, 2), &wsaData); //初始化套接字
hints.ai_family= AF_INET6; //指定用 IPV6 协议
hints.ai_socktype= SOCK_DGRAM; //用数据报
hints.ai_protocol= IPPROTO_UDP; //用UDP 协议
hints.ai_flags= AI(1)NUMERICHOST; //IP 用数字表示
s_send= socket(res-> ai_family, res-> ai_socktype, res-> ai_protocol); //建服务端套接字
rc= getaddrinfo((LPCTSTR)m_local,“2000”, &hints, &res); //解析本机地址
if(bind(s_send, res-> ai_addr, res-> ai_addrlen)= = SOCKET_ERROR) //绑定本地监听端口
AfxMessageBox(“bind 失败”);
else
{AfxMessageBox(“bind 成功”); GetDlgItem(IDOK)-> EnableWindow(TRUE);}
}
int iLen= recv(s_send, sBuf, sizeof(sBuf), 0); //接收UDP 数据
closesocket(s_send); //关闭套接字
```

4.2 客户端程序

```
WSAStartup(MAKEWORD(2, 2), &wsaData); //初始化套接字
hints.ai_family= AF_INET6; //指定用 IPV6 协议
hints.ai_socktype= SOCK_DGRAM; //用数据报
hints.ai_protocol= IPPROTO_UDP; //用UDP 协议
hints.ai_flags= AI_NUMERICHOST; //IP 用数字表示
s_send= socket(res-> ai_family, res-> ai_socktype, res-> ai_protocol); //建客户端套接字
rc= sendto(s_send, m_send, m_send.GetLength(), 0, res-> ai_addr, res-> ai_addrlen); //发送数据
closesocket(s_send); //关闭套接字
```

5 结 论

IPv6 作为下一代网络的基石, 它的重要性是毫无疑问的, 我们无疑应当尽快学习掌握有关 IPv6 套接口的有关内容, 同时紧跟目前尚未完善标准化进程。争取为今后的开发工作打下良好的基础。

参 考 文 献

[1] 张云勇. 基于 IPv6 的下一代互联网 [M]. 北京: 电子工业出版社, 2004.

[2] RFC3493, Basic Socket Interface Extensions for IPv6 [S], 2003.

[3] RFC2292, Advanced Sockets API for IPv6[S], 1998.

[4] 雷 斌, 杨建华. Visual C+ + 6.0 网络编程技术 [M]. 北京: 人民邮电出版社, 2000.

下 期 要 目

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
52. 基于 USB30 的设备自定义请求实现方法
53. IEEE1588 协议在网络测控系统中的应用
54. USB30 物理层中弹性缓冲的设计与实现
55. USB30 的高速信息传输瓶颈研究
56. 基于 IPv6 的 UDP 通信的实现
57. 一种基于 IPv6 的流媒体传送方案研究与实现
58. 基于 IPv4-IPv6 双栈的 MODBUS-TCP 协议实现
59. RS485CAN 网关设计与实现
60. MVB 周期信息的实时调度

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
33. 基于 VxWorks 实时控制系统中文交互界面开发平台
34. 基于 VxWorks 操作系统的 WindML 图形操控界面实现方法
35. 基于 GPU FPGA 芯片原型的 VxWorks 下驱动软件开发
36. VxWorks 下的多串口卡设计
37. VxWorks 内存管理机制的研究
38. T9 输入法在 Tilcon 下的实现
39. 基于 VxWorks 的 WindML 图形界面开发方法
40. 基于 Tilcon 的 IO 控制板可视化测试软件的设计和实现
41. 基于 VxWorks 的通信服务器实时多任务软件设计

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
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
40. 基于嵌入式 Linux 下的 USB30 驱动程序开发方法研究
41. Android 手机应用开发之音乐资源播放器
42. Linux 下以太网的 IPv6 隧道技术的实现
43. Research and design of mobile learning platform based on Android
44. 基于 linux 和 Qt 的串口通信调试器调的设计及应用
45. 在 Linux 平台上基于 QT 的动态图像采集系统的设计
46. 基于 Android 平台的医护查房系统的研究与设计
47. 基于 Android 平台的软件自动化监控工具的设计开发

48. 基于 Android 的视频软硬解码及渲染的对比研究与实现

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
21. DCOM 协议在网络冗余环境下的应用
22. Windows XP Embedded 在变电站通信管理机中的应用
23. XPE 在多功能显控台上的开发与应用
24. 基于 Windows XP Embedded 的 LKJ2000 仿真系统设计与实现
25. 虚拟仪器的 Windows XP Embedded 操作系统开发
26. 基于 EVC 的嵌入式导航电子地图设计

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
20. 基于 PowerPC 的模拟量输入接口扩展
21. 基于 PowerPC 的车载通信系统设计
22. 基于 PowerPC 的嵌入式系统中通用 IO 口的扩展方法
23. 基于 PowerPC440GP 型微控制器的嵌入式系统设计与研究
24. 基于双 PowerPC 7447A 处理器的嵌入式系统硬件设计

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-OS II 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
28. 基于 ARM11 的嵌入式无线视频终端的设计
29. 基于 S3C6410 的 Uboot 分析与移植
30. 基于 ARM 嵌入式系统的高保真无损音乐播放器设计
31. UBoot 在 Mini6410 上的移植
32. 基于 ARM11 的嵌入式 Linux NAND FLASH 模拟 U 盘挂载分析与实现

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

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
25. 基于龙芯 2F 架构的 PMON 分析与优化
26. CPU 与 GPU 之间接口电路的设计与实现
27. 基于龙芯 1A 平台的 PMON 源码编译和启动分析
28. 基于 PC104 工控机的嵌入式直流监控装置的设计
29. GPGPU 技术研究与发展
30. GPU 实现的高速 FIR 数字滤波算法
31. 一种基于 CPUGPU 异构计算的混合编程模型
32. 面向 OpenCL 模型的 GPU 性能优化

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
7. 数据结构考题 - 第 1 章 绪论
8. 数据结构考题 - 第 2 章 线性表
9. 数据结构考题 - 第 2 章 线性表 - 答案