WindML中Mesa的应用

毛子鹰[1],高贵明[2**]

(1.南京信息工程大学 电子与信息工程学院,南京 210044; 2.南京船舶雷达研究所,南京 210003)

摘 要:研究了在WindML3.0中应用Mesa来改进嵌入式图形界面显示效果的方法。针对实际产生的问题进行了简要的分析,提出了相应的解决方法,达到了预期的效果。

关键词:WindML; Mesa; VxWorks; 图形界面

中图分类号:TP311.5 **文献标识码**:A **文章编号**:1009-0401(2009)03-0067-04

The application of Mesa in WindML

MAO Zi-ying[1], *GAO Gui-ming*[2]

(1. *College of Electronics and Information Engineering, Nanjing University of Information Science & Technology, Nanjing* 210044; 2. *Nanjing Marine Radar Institute, Nanjing* 210003)

Abstract: A method of applying Mesa in WindML3.0 to improve the display effect of the embedded GUI is introduced. Based on the problems in the test, the brief analysis is made with the corresponding solutions to achieve the expectant effect.

Keywords: WindML; Mesa; VxWorks; GUI

1 引 言

WindML2.0为VxWorks提供了基本的图形、音频和视频的支持,满足了VxWorks下图形界面设计的基本需求。但是,使用WindML2.0设计图形界面也有一些无法克服的缺点:首先,WindML2.0提供的窗口系统比较原始,这造成了使用窗口系统的界面程序比较复杂;其次,WindML2.0仅提供了包含点、直线、矩形、椭圆、多边形这5种基本的2D图形绘制函数,而且没有反走样的处理,严重影响了图形的显示效果。WindML3.0是对WindML2.0的一次比较大的改进,窗口管理器的运用简化了窗口图形界面的设计,丰富了窗口界面的效果。但是,WindML3.0沿用了WindML2.0的基本2D图形库,因此在图形绘制上的缺陷没有得到根本上的改观。本文所描述的就是在WindML3.0下利用Mesa来弥补这一缺陷的方法。

2 Mesa简介

Mesa是对于OpenGL规范的一个软件上的实现,原先是为UNIX/X11所设计的。由于这是一个比较成熟的开源软件,Mesa可以方便的应用于很多的系统之中,如Windows, VMS, OS/2。与Mesa的最新版本相比,Mesa4.0完整地提供了对WindML的支持代码,为在WindML中使用提供了详细的解决方案。

Mesa以源码的方式提供,需要由用户自己编译成可用的库。编译的方法可以参考文献[4]。

Stephane Raimbault所编写的《WindML Driver for Mesa 4.0》[3]中给出了在WindML中使用Mesa的基本流程:

(1) 调用 ugMesaCreateContext( ),创建UGL/Mesa下文环境,并获取屏幕显示格式。

(2) 调用 ugMakeCurrentContext( ),将UGL/Mesa的图形缓冲区与创建的UGL/Mesa上下文绑定,并将该上下文设定为当前上下文。

(3) 在需要的时候调用OpenGL函数。

(4) 采用双缓冲策略的时候,调用 ugMesaSwapBuffers( ),更换绘图页面。

(5) 在UGL销毁前,调用 MesaDestroyContext( )。

3 Mesa4.0的修改与使用

Mesa4.0中的支持代码原本是针对WindML2.0开发的,可以正确地应用于WindML2.0开发的图形界面。但是,当在WindML3.0的窗口系统中使用Mesa4.0开发图形界面的时候,却经常造成窗口失去响应。只有不采用窗口系统的情况下,才能够达到与WindML2.0相同的结果。

3.1 问题产生的原因

上述问题的产生主要源于以下两个方面:

首先,Mesa4.0与WindML3.0的窗口系统在绘图页面的选择上不同。Mesa4.0中主要的绘图页面是WindML2.0中采用的Page0,其地址为帧缓冲区(framebuffer)的地址。然而,在WindML3.0的窗口系统中,并没有将帧缓冲区默认为绘图页面,而是采用了一个独立于屏幕和帧缓冲区的专门的页面。在绘图页面的选择上的差异造成了WindML绘制的图形和Mesa绘制的图形难以在同一时间显示在同一屏幕上。

其次,WindML3.0提供的基本的窗口管理器不适用于双缓冲输出。考虑到Mesa4.0和WindML3.0的窗口系统都支持双缓冲绘制,如果能够同步两者的双缓冲页面,上述问题并不难以解决。然而,WindML3.0所提供的默认的窗口管理器在绘图模式上并没有采用双缓冲的方式。这种不一致也是导致双缓冲模式下WindML3.0的窗口系统失去响应的重要原因。修改WindML3.0的窗口管理器可以解决这一问题,但修改量却比较大,不能够从根本上解决问题。双缓冲绘制的方式带来了额外的负担,并不是解决上述问题的最好方法。

综合这两方当面的问题,为了在WindML3.0中使用Mesa,必须修改Mesa的源代码,为其绘图提供正确的页面,并且,WindML3.0在窗口系统下的绘图模式不应选择双缓冲模式。

3.2 Mesa的修改

对于Mesa4.0的修改主要就是更改绘图页面的选择,并处理相应产生的问题。

(1) 更改绘图页面

在Mesa4.0对于WindML提供支持的程序中,使用了firstPage和secondPage这两个结构变量维护绘图页与显示页。在Mesa的直接绘图模式下,firstPage既是绘图页,也是显示页;在双缓冲模式下,firstPage与secondPage轮流维护绘图页和缓冲页。虽然从原理上来说,WindML3.0的off-screen实际上是双缓冲的方法,但是,由于双缓冲页面的更换是由窗口系统自动完成的,因此,在实际应用中,对于Mesa而言,off-screen模式更加接近于直接绘图模式。由于双缓冲的绘图方法不适用于WindML3.0的窗口系统,我们只需要修改Mesa中有关直接绘图模式的相应代码,将其中firstPage的pageId设为绘图页面的pageId,将firstPage所维护的缓冲区地址设为绘图页面的地址,如图1所示。

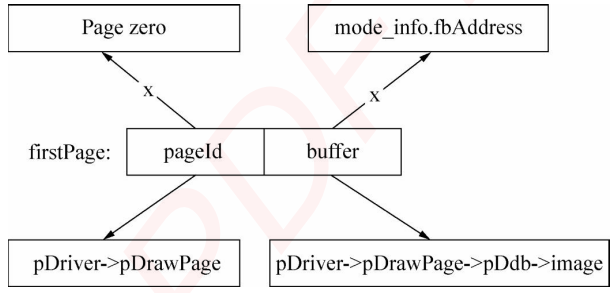

图1 firstPage的更改

主要代码如下(在UGL/MESA上下文环境的创建函数中添加):

```
UGL_GENERIC_DRIVER * pDriver = (UGL_GENERIC_DRIVER *) umc->devId;
UGL_GEN_DDB * pBitmap = (UGL_GEN_DDB *) pDriver->pDrawPage->pDdb;
umc->firstPage->buffer = pBitmap->image;
umc->firstPage->pageId = pDriver->pDrawPage
```

(2) 更改图形上下文的相应代码

Mesa4.0对于WindML的支持代码中自身维护了一个图形上下文,由于绘图需要发生在某个窗口界面中,这一图形上下文必然与WindML3.0的窗口系统所创建的图形上下文产生冲突。因此,必须修改原有的有关图形上下文的代码。

首先,在直接绘图模式中的绘图页面设置之前,应当将Mesa中创建的图形上下文销毁,并将UGL/Mesa上下文中的图形上下文设定为UGL所维护的图形上下文:

```
UGL_GC_ID gc = pDriver->gc;
uglGcDestroy(umc->gc);
umc->gc = gc;
```

其次,由于做了上述的修改,在销毁Mesa的时候(ugMesaDestroyContext),必须在销毁图形上下文之前加上判断,只有UGL/Mesa上下文中的图形上下文与UGL中的图形上下文不同才加以销毁,避免误销毁WindML所维护的图形上下文。

(3) 视口的设置

完成了页面的设置并不表示图形就能够正常地输出,对于计算机图形来说,还必须考虑视口(Viewport)的问题。只有在视口区域中的图形操作才能够输出在屏幕之上。对于视口的错误操作必然导致图形输出位置的错误,甚至导致屏幕上不能够输出图形。WindML2.0中,图形绘制往往发生在全屏幕上,因此,在与之适应的 Mesa的代码中,将视口设置为了全屏幕。但是在 WindML3.0中,这种做法是不可取的。WindML3.0提供了比较完善的窗口系统,对于视口基本上是由窗口系统自动调整的:在响应绘图消息的时候,窗口系统根据当前窗口在屏幕上的位置,自动完成视口的调整。因此,在窗口系统外部,例如在绘图消息的响应函数中,对于视口的调整必须考虑之前视口的状态,否则将与窗口系统的视口设置发生矛盾,导致视口设置的混乱。解决方法有很多,我们采用的方法是将 Mesa中关于视口设置的函数删除。这样做的好处在于,如同普通绘图那样,绘图时的视口设置完全交由 WindML3.0的窗口系统完成,无需添加额外的视口设置代码,也大大降低了程序出错的可能性。

3.3 修改后 Mesa的使用

本文并没有对 Mesa做出根本性的改变,因此在 WindML3.0中使用 Mesa的基本流程也就没有大的改变。但是,由于修改后的 Mesa需要应用在 WindML3.0的窗口系统中,在使用的过程中有一些必须注意的问题。

(1) 窗口绘制模式的选择

如前所述,在 WindML3.0的窗口系统中使用 Mesa时,图形界面的绘制不应当采用双缓冲的方式绘制,推荐采用 off-screen模式绘制。由于不采用双缓冲绘图,并需要利用 WindML对界面绘制进行管理,在第2节所述的流程中,在创建 UGL/Mesa上下文的时候 ugMesaCreateNewContext的参数 mode应当设为"UGL_MESA_SINGLE | UGL_MESA_WINDML_EXCLUSIVE"。

(2) 窗口应用程序的栈空间的选择

WindML3.0中的窗口应用是由函数 winAppCreate所发起的一个任务所维护的。对于普通的窗口应用,在窗口绘制的过程中不会需要很多的函数调用,因此,在创建这个任务的时候,并不需要分配很大的栈空间。一般来说,winAppCreate 所提供的默认大小(10000bytes)就足够了,因此,其参数 stackSize可以设定为默认值 0。但是,Mesa是对于 OpenGL的一种软件上的实现,牵涉到了大量的函数调用,显然这一默认大小就远远不够了。图 2是在一个采用了 Mesa绘制图形的程序中,用 Tornado的 Browser工具观察到的某个瞬间的栈使用情况的截图。

图 2 任务占用栈的大小

在该图中,"ppi窗口"这个任务采用了 Mesa绘图。在这个瞬间,所占用的栈空间达到了 62012bytes,显然已经远远超过了默认情况下分配的栈的大小。为该任务分配不足的栈空间将导致系统出现访问异常而崩溃。因此必须根据实际情况为窗口任务分配合适的栈空间。对于图中所示任务,根据实验数据,考虑了充分的余量后,120000bytes的栈空间是一个比较好的选择。

(3) OpenGL函数的调用

在 WindML3.0的窗口系统中,为了能够正确地调用 OpenGL函数,第 2节中所述的流程应当完整的体现在每一个对 MSG_DRAW消息响应的回调函数中。在此基础上,可以按照通常的方式调用 OpenGL函数,实现 OpenGL所提供的特性。为了验证其效果,我们利用 VxWorks在 Windows下的模拟器做了相应实验。图 3和图 4通过截取局部细节展示了使用 OpenGL绘图在反走样上带来的明显改进。

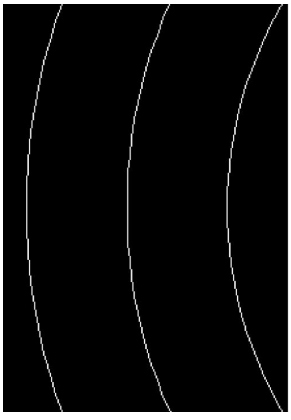

图 3 反走样前

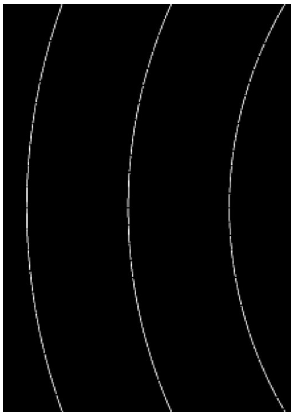

图 4 反走样后

4 结束语

为了在 WindML3.0中使用 Mesa,对于 WindML和 Mesa在绘图页面上的选择做出了分析。在此基础上修改了 Mesa4.0的源代码,并列举了其在 WindML3.0中的使用中通常需要考虑的问题和解决方法。实验的结果表明,本方案基本是成功的,能够保证 Mesa在 WindML3.0中的正常使用。

参考文献:

[1] WindML SDK Programmer's Guide 3.0[M/DK]. WindRiver Systems, Inc. 2002.

[2] VxWorks Programmer's Guide 5.5[M/DK]. WindRiver Systems, Inc. 2002.

[3] Stephane Raimbault WindML Driver for Mesa 4.0 [M/OL]. http://www.mesa3d.org

[4] 在 VxWorks下进行 OpenGL编程的环境搭建教学 [EB/OL]. http://www.cevx.com/bbs/viewthread.php?tid=11343&extra=&page=1.

(上接第 63页)

$$\triangle t = \frac{860\ P}{Q_v \cdot C_p \cdot \gamma} = 9℃$$

经校核计算出功放组件表面传热系数 $h = 18.15\ W/m^2 \cdot ℃$,根据强迫风冷的热传递公式 $Q = h \cdot A \cdot \triangle t$得到 $\triangle t = 21℃$,可以知道功放组件表面温度为 30 + 21 = 51℃,低于最高允许温度 54.2℃。

4 结 论

固态发射机冷却系统的计算和校核证明在车载方舱的工作环境下,强制空气冷却满足系统散热要求,固态发射机能够正常工作。目前,根据此方案设计的固态发射机已经在车载方舱的工作条件下顺利通过了各种试验及连续开机工作测试,系统工作稳定。

参考文献:

[1] 谢德仁. 电子设备热设计 [M]. 南京:东南大学出版社, 1989.

[2] 刘永智. 某固态发射机的热设计 [J]. 现代电子, 2001(3).

[3] 李德伟. 某雷达发射机系统热设计分析 [J]. 电子工程, 1994(4).

(上接第 66页)

完成以上工作后,分别选择 GME工具栏上的 Generate Package Descriptor、Generate Domain Descriptor、Generate DeploymentPlan就可以生成工程的部署描述文件,这些文件包括构件本身的描述(CCD文件)、构件实现描述(CID文件)、封装描述(CPD文件)、配置计划描述(CPD文件)等。将所有输出文件保存到特定目录下,一般放在 descriptors文件夹中,这样就完成了工程的描述。生成的文件就以 XML语言描述了整个工程之间的关系,在把 A/R显示构件和驱动构件进行部署时就需要这些文件。

5 结束语

使用 CoSMIC将 IDL转换成 PICML,再使用 GME,将 PICML导入,然后利用模型,可视化操作,最终可以生成 CCM工程的 XML描述文件。该方法使用方便,在开发过程中可以减少不必要的错误,同时也节约了时间。

参考文献:

[1] Vanderbilt University. GME Overview [OL]. http://www.isis.vanderbilt.edu/Projects/gme/.

[2] Ming Xiong, , Building a Stock Quoter with CoSMIC and DAnCE [EB/OL]. http://www.dre.vanderbilt.edu/~mxiong/CoSMIC/.

[3] 张云勇,张智江,刘锦德,等. 中间件技术原理与应用 [M]. 北京:清华大学出版社, 2004.

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

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：

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

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：

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

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