基于VxWorks的WindML图形界面开发方法

蔡　华　卞新高　史中权　丁　坤　河海大学机电工程学院（213022）

Abstract

This paper briefly introduces embedded real-time operating system VxWorks' media library WindML,and discusses WindML designing ways of graphical interfaces,and through an instance expounded the functions' implementation methods such as WindML configuration and loading,2D graphics drawing,text display,the response of input device such as cursor and keyboard, region and window drawing.

Keywords:embedded real-time operating system,VxWorks,graphical interfaces,WindML

摘　要

简要介绍了嵌入式实时操作系统VxWorks的媒体库WindML,讨论了WindML图形界面开发方法,通过实例详细阐述了WindML的配置和加载、二维图形的绘制、文本显示、鼠标和键盘等输入设备的响应以及区域和窗口的绘制等功能的实现方法。

关键词:嵌入式实时操作系统,VxWorks,图形界面,WindML

VxWorks操作系统是美国风河公司(Wind River Systems Inc.)于1983年设计开发的一种嵌入式实时操作系统(Embedded RTOS)。WindML即Wind Media Library(媒体库),是VxWorks库的一部分,它支持多媒体程序运行于嵌入式操作系统,风河公司设计它主要是用来提供基本的图形、视频和声频技术以及为用户提供一个开发标准用户设备驱动程序的框架。并且,WindML提供了一系列工具用来处理输入设备和过程事件。以上这些功能绝大部分都由WindML提供的API来完成。

1　WindML的结构

WindML包括两个组件:软件开发包(Software Development Kit,SDK)和驱动程序开发包(Driver Development Kit,DDK)。SDK组件用于为各种平台开发与硬件无关的应用。它在图形、输出处理、多媒体、字体和内存管理方面提供了完整的API。DDK用于开发驱动程序。它提供了一整套可用于通用硬件配置、软件框架的参考驱动程序,以及支持开发人员从提供的'通用'代码快速创建新驱动程序的API。WindML的层次结构见图1。

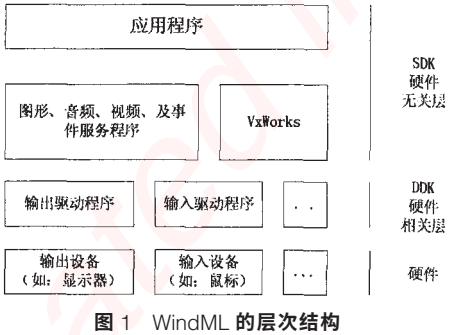

图1　WindML的层次结构

2　WindML的图形界面开发方法

利用WindML进行图形界面开发,首先要对WindML进行相应的配置和编译,然后将WindML加载到VxWorks内核。当WindML配置和添加完成后,就可以在VxWorks的开发环境Tornado中进行编程,从而实现图形界面的开发。利用WindML可以实现二维图形的绘制、文本的显示、鼠标和键盘等输入设备的响应、区域和窗口的绘制等等功能。

2.1　配置和编译WindML

在第一次使用WindML前,必须要配置WindML,WindML配置有两种方法:一是通过Tornado下的WindML配置工具配置WindML,二是通过命令行方式配置WindML。一般都是使用配置工具来完成WindML的配置,如果使用配置工具不能满足配置要求,也可以通过命令行方式配置WindML。在通过配置工具配置WindML时,配置选项包括处理器的选择,图形设备配置,输出设备配置,字体配置,音频配置和杂项配置。用户可以根据自己的要求,选择合适的配置。选好配置后,直接按配置工具上的编译按钮即可进行编译。

2.2　加载WindML

配置和编译好WindML后,可以使用Tornado工程管理工具把WindML加载到VxWorks映像里,加载时,可以根据自己的需要进行相应的选择。一般对于图形界面的开发,我们只要将2D graphics和complete 2D library添加进VxWorks即可。

2.3　图形初始化

WindML配置和加载完成后,就可以用WindML进行编程。编程时,必须首先调用函数uglInitialize()完成初始化。接着要进行颜色的配置,WindML的图形界面是以像素为单位的,一般采用配色表来选择颜色,先在配色表上配置好每一种颜色的R、G、B值,并用其在配色表中的索引值来代表这种颜色。当配置好颜色后,还要创建图形上下文(GC),图形上下文包含了绘图的特征信息,大部分的绘图操作只有通过指定图形上下文,定义了绘图特征才能进行。WindML使用函数uglGcCreate()来创建图形上下文,创建GC后就可以进行画图的基本设置,如图形的前景色和背景色、线的类型和线宽、图形的填充模式、默认的位图、光栅覆盖模式、当前使用的字体等等。

2.4　二维图形的绘制及文本的显示

当完成初始化后,就可以调用WindML API函数来完成二维图形的绘制。WindML具有画直线、矩形、椭圆、点、多边形等简单图形的功能,也就是提供了画这些二维图形的函数,例如绘制直线的函数uglLine()、绘制矩形的函数uglRectangle()等等。WindML可以使用多线程或多任务,但资源是一定的,为防止多

线程之间产生资源冲突，需要使用互斥信号量锁定资源。WindML 中，一般在使用一组画图函数前，用 uglBatchStart(gc)通过互斥信号量锁定图形上下文、图形设备及缓冲，并且隐藏光标。在画图操作完成后，再用 uglBatchEnd(gc)释放被锁定的资源以被其他的画图函数所使用。

WindML 文本渲染和字体管理 API 提供了方便的文字显示方法。在文本显示之前由函数 uglInitialize()完成字体驱动和字体引擎的初始化。然后使用函数 uglDriverFind()获得字体驱动的 ID，再由函数 uglFontDriverInfo()设置字体驱动程序。由于许多嵌入式系统只提供少数的有限字体选择，因此一般还需要应用函数 uglFontFindString()找到系统提供的与所要显示字体属性最相匹配的字体。当一个字体找到后，由函数 uglFontCreate()生成该字体。当要使用一个已经创建了的字体时，必须首先使用函数 uglFontSet()将该字体的 ID 选入图形上下文（GC），然后就可以使用显示函数 uglTextDraw()(uglTextDrawW 用于双字节)用当前图形上下文中的字体在显示器上显示文本。

2.5 区域和窗口的绘制

WindML 的区域管理 API 可以在界面上定义一个区域，这个区域是由一组矩形区围绕而成，可以在这个区域里画二维图形。如果是更为复杂的应用程序，或者要在多任务或多线程之间共享显示，则要用到窗口 API。窗口 API 可以创建、显示和操作多个窗口，并且可以处理这些窗口中的事件。窗口可以移动，改变大小，重叠，或者放在其他窗口内部。这样，多个任务就可以在一个屏幕上的不同窗口内同时显示图形了。绘制区域一般是先通过函数 uglRegionCreate()来创建区域，然后就可以在创建的区域里画二维图形，同时可以通过函数 uglRegionRectInclude()包含一个矩形到区域里和 uglRegionRectExclude()从区域里去除一个矩形。还可以通过区域的相关函数进行区域的移动，复制，清除等等操作。而绘制窗口则要创建事件路由，并要创建窗口应用上下文。窗口除了具有区域的功能外，还可以响应外界的鼠标、键盘等输入设备的事件。具体的程序实现方法可以参见下面的实例。

2.6 扩展功能的实现

WindML 本身不支持中文显示，因此要想实现中文显示，可以利用 WindML 对双字节编码的支持，实现对汉字的点阵存储、点阵获取、点阵显示的全过程，并利用 WindML 的双字节显示函数实现汉字码到汉字的显示。这种方法需要自己编写汉字字库，然后将汉字字库添加进 WindML 的内核，显示中文的编程方法和文本显示是一样的，只要将汉字当作双字节显示即可，不过这种方法具有局限性，就是能显示的中文必须是汉字字库里的，这样能显示的汉字受到所添加的汉字字库的限制。

WindML 组件中提供的窗口 API 不能实现 Windows 系统中的窗口功能，但是我们可以利用 WindML 现有的功能仿真实现，现在简要阐述一下实现过程。首先，我们利用 WindML 的窗口 API 画出窗口外观，然后在窗口的最上方用不同颜色画一个与窗口同宽的矩形来模拟窗口的标题栏，在标题栏的左上角可以写字，在右上角可以画出最小化图标和关闭图标，当鼠标点击最小化图标区域，就将当前窗口关闭，在屏幕的左下角重新画出一个小矩形窗口来作为最小化后的窗口，当鼠标再次点击最小化后的窗口时，再在原来的窗口位置重新画一下窗口；当鼠标点击关闭图标区域时，将当前窗口关闭即可。这样就模拟了 Windows 系统中的窗口的最小化和关闭功能。

2.7 实例

下面通过一个程序实例来简单说明一下利用 WindML 进行图形界面开发的过程。下面给出部分程序代码并加以说明。

```
.....
UGL_GC_ID gc;                                   //定义图形上下文
UGL_DEVICE_ID devId;                            //定义显示设备 ID
UGL_FONT_DRIVER_ID fontDrvId;                   // 定义字体驱动 ID
UGL_FONT_ID fontText;                           // 定义字体 ID
UGL_FONT_DEF fontDef;
UGL_EVENT_SERVICE_ID eventServiceId;            // 定义事件服务 ID
UGL_EVENT event;                                // 定义事件
UGL_EVENT_ROUTER_ID eventRouterId;              // 定义事件路由 ID
UGL_APP_ID appId;                               // 定义应用程序 ID
UGL_WINDOW_ID windowId;                         // 定义窗口 ID
uglInitialize( );                               // 初始化
uglDriverFind (UGL_DISPLAY_TYPE, 0, (UGL_UINT32 *)&devId);
//获得显示设备 ID
gc = uglGcCreate(devId);                        // 创建图形上下文
.....
uglDriverFind (UGL_FONT_ENGINE_TYPE, 0, (UGL_UINT32 *)
&fontDrvId);
uglFontDriverInfo (fontDrvId, UGL_FONT_TEXT_ORIGIN, &textOri-
gin);                                           // 获得字体驱动
uglFontFindString(fontDrvId, "familyName=Song; pixelSize = 16",
&fontDef);
if ((fontText= uglFontCreate (fontDrvId, &systemFontDef)) ==
UGL_NULL)
{
printf("Font not found. Exiting.\n");
return;
}                                               // 查找及生成匹配字体
uglFontSet(gc, fontText);                       // 设置字符样式
uglDriverFind (UGL_EVENT_SERVICE_TYPE, 0, (UGL_UINT32 *)
&eventServiceId )；                             // 获得输入事件服务设备驱动
eventRouterId =winEventRouterCreate (eventServiceId, 100);
// 创建事件路由
appId=winAppCreate (eventRouterId,100); // 创建窗口应用上下文
windowId = winCreate (appId, 10, 20, 100, 200);  // 创建窗口
winAttach (UGL_NULL_ID, windowId,UGL_NULL_ID); // 关联窗口
if (winEventGet (appId, &event, 200) != UGL_STATUS_OK)
        continue;
switch (event.header.type)                      // 判断事件类型
    {
     case WIN_EVENT_TYPE_DRAW:      // 画窗口事件
         {
          UGL_WINDOW_ID windowId = (UGL_WINDOW_ID)
event.header.objectId;
          UGL_RECT windowRect;
          winDrawStart (windowId, gc, UGL_TRUE);// 开始画窗
口，锁定窗口
          winRectGet (windowId, &windowRect);
          UGL_RECT_MOVE_TO (windowRect, 0, 0);  // 在窗
口坐标中，将窗口左上点定义为原点
          uglLineWidthSet (gc, 1);                  // 设置线宽
          uglForegroundColorSet (gc, colorTable[WHITE].uglcol-
or);       // 设置前景色
          uglBackgroundColorSet (gc, colorTable[BLUE].uglcol-
or);        // 设置背景色
          uglRectangle (gc, windowRect.left, windowRect.top,
                        windowRect.right, windowRect.bot-
```

```
tom ); // 画窗口外边框
            uglTextDrawW (gc, windowRect.left + 100, windowRect.top + 30, -1, "文字窗口演示\000"); // 显示汉字
            ......
winDrawEnd (windowId, gc, UGL_TRUE); // 画窗口结束,释放窗口
}
            break;
case UGL_EVENT_TYPE_POINTER:                    // 鼠标事件
            {
             UGL_WINDOW_ID windowId = (UGL_WINDOW_ID) event.header.objectId;
             if (UGL_BUTTON1_DOWN (&event) || UGL_BUTTON2_DOWN (&event))    // 如果是按下左键或右键
            ......
if (UGL_BUTTON1_DRAG (&event) || UGL_BUTTON2_DRAG (&event))    // 如果是拖动左键或右键
 ......
            }
            break;
        case UGL_EVENT_TYPE_KEYBOARD:       // 键盘事件
            ......
            break;
}
......
```

本例是利用 WindML 画出两个窗口，这两个窗口一个是用来显示汉字文本,一个是用来显示基本的二维图形,这两个窗口分别是由两个任务发起的，可以分别响应来自鼠标和键盘的事件。将程序编译,并下载到目标仿真器,运行后,在终端显示出的图形界面如图 2 所示。

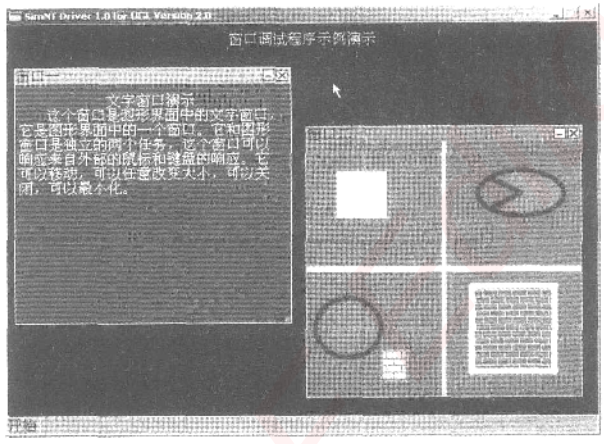

图 2　图形界面

3　结束语

本文对嵌入式实时操作系统 VxWorks 的媒体库 WindML 及其图形界面开发的方法作了比较详细的介绍,上面提到的图形界面开发的方法已经被作者成功地应用于实际工程开发,并取得了比较好的效果。

参考文献

1　孔祥营,柏桂枝编著.嵌入式实时操作系统 VxWorks 及其开发环境 Tornado[M].中国电力出版社,2002

2　WindRiver.VxWorks Programmer´s Guide5.4[S].WindRiver Systems Inc.,1999

收稿日期：2005.4.22]

风河强力支持电信运营商级刀片服务器 Linux 软件开发

全球领先的设备软件优化解决方案提供商风河系统公司与 Artesyn 技术有限公司联合宣布，将会在 Artesyn 公司的 AdvancedTCA 电信级刀片服务器中提供风河网络设备专用平台——Wind River Platform for Network Equipment,同时 Artesyn 公司也将成为风河联盟计划中的平台合作伙伴成员。同时,风河公司还与 AdvancedTCA 电信刀片服务器的处理器提供商 Freescale 公司联合宣布,对 Freescale 公司最新的 PowerPC 和 PowerQUICC 处理器提供所有 Wind River DSO 平台支持。对于网络设备市场来说,Wind River 和 Freescale 联合提供集成化的 DSO 解决方案,将帮助网络设备制造厂商以更加简便、快速、低成本的方式开发可靠的电信设备和网络解决方案。

风河与 Artesyn 将携手开发面向 Artesyn 电信刀片服务器的运营商级 Linux 和 VxWorks 板级支持包(BSP)。这项工作将会首先从基于 PowerPC 处理器的 KatanaQp AdvancedTCA 刀片服务器开始。风河公司将会为新的 BSP 提供验证和认证。两家公司将会联合开展市场推广活动。

风河网络设备者用平台 (Wind River Platform for Network Equipment)包括最新 OSDL 运营商级 Linux(版本 2.6.1)的主流版本以及 Wind River 公司基于 Eclipse 的 Workbench 开发套件,并且带有丰富的网络中间件。这个平台可以使电信设备开发商非常方便地开发和部署网络设备应用,适用领域包括核心部分、企业应用、接入应用和边缘应用。

KatanaQp 是一款高性能的 AdvancedTCA 电信刀片服务器，其中包配备了两个 Freescale MPC7447A 处理器，带有 4 个 PTMC 扩展插座、冗余的 IPMI 系统管理接口、PICMG 3.1 兼容的 AdvancedTCA 接口和 10 Gigabit 以太网通道。其中的高速 Freescale processor 处理器、交换式网络 AdvancedTCA 接口、灵活的中间层扩展和集成化的系统管理可以方便地进行系统客户化,实现灵活多样的控制和包处理应用,包括 WAN 访问、SS7/SIGTRAN 信令、媒体网关、流量处理、无线基站和软交换。

风河公司中国首席代表韩青指出：“风河网络设备专用平台 Wind River Platform for Network Equipment 为网络设备制造商提供了可靠、高效的开发环境,而 Artesyn 公司的 AdvancedTCA 刀片服务器和 Freescale 公司的高性能 PowerPC 处理器在通信市场历史悠久,已经被通信市场中的众多厂商广泛采用。我们三家厂商建立更紧密的合作关系,将会让为网络通信设备制造商利用最新技术开发高性能的网络设备提供有力的帮助,同时让他们显著提高开发效率、降低开发成本。”

由美国风河系统公司北京办事处供稿)

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

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

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

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

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

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

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

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

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