VxWorks 的 WindML 图形界面程序的框架分析

Analysis of WindML Graphica

李海亮　石鹏程　哈尔滨工程大学水声工程学院 黑龙江 哈尔滨 150001

摘　要

WindML作为VxWorks下的图形界面开发组件，有其特有的框架结构。介绍了WindML环境的建立，并通过对一个简洁的WindML例程来分析WindML程序的框架结构。

关键词 图形界面；VxWorks；框架结构

Abstract

WindML, one of the graphical interface empold the environment of WindML, and using a compact

Keywords: graphical interface; VxWorks; WindM

美国风河公司的VxWorks操作系统自1992年登陆中国，经过十年时间，逐渐进入了国内通信、国防、工业控制、医疗设备等嵌入式实时应用领域，在国内拥有众多的用户。特别是最近几年，VxWorks操作系统越来越多地占据了国内嵌入式实时应用市场。

为了尽量减小嵌入到实际系统中运行内核的大小，一般的嵌入式系统不提供图形界面。但是为了提高系统的易用性，VxWorks提供了一个图形开发组件WindML利用它可以更快、更简单地开发出资源消耗更少的图形界面。

我们学习任何一种图形界面开发工具都会从hello这个简单的例程学起，比如Windows编程、wxWindows、FLTK等。但是，虽然WindML提供了一个简单却又完整地体现了WindML的结构的例程。本文就编写一个WindML例程，并通过对该例程的分析，使WindML程序的框架结构很清晰的显示出来。

1 WindML概述

WindML即WindML 3.0，是VxWorks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、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 3.0提供了一系列工具用来处理输入设备和过程事件。

WindML由两部分组成：软件开发工具箱（SDK）和硬件开发工具箱（DDK）。

SDK用来开发应用程序，在图形、输入处理、多媒体、字体和内存管理等方面提供了程序接口API。DDK提供了普通硬件的驱动，是SDK和硬件的中间层。

WindML的结构关系图如图1所示。

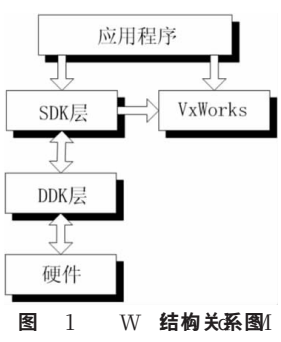

图1 WindML结构关系图

2 WindML程序框架简述

几乎所有的编程语言都有一个框架可以遵循。利用WindML开发的图形界面也不例外。在编程之前，很好的理解并掌握该语言程序的框架结构，能够使我们减少编程时的错误，而且可以提高编程的效率。下面就给出WindML图形开发设计的总体框架：

1）对程序进行初始化，调用ugl来完成UGL组件的初始化。

设备驱动装载，包括显示设备、输入设备、字体、声音等。

创建图形环境变量，初始化颜色，并分配预定义的GC值等。

利用一个无限循环构成事件处理机制。这一步是程序的关键部分，所有的鼠标、键盘等交互过程都在这里完成，循环体只有在特定情况下才会跳出，从而执行下面的内容。

完成对所有资源的释放，并退出。

上面的解释使WindML程序的框架结构基本体现出来了，但是非常的笼统。第3节将通过一个简洁的hello示例的WindML程序分析，剖析来具体说明WindML程序框架结构的每一部分。

3 WindML下hello world程序分析

3.1 WindML环境的建立

完成本例程所依托的调试环境如下：

主机：Windows XP Pro sp2

由于使用仿真器VxSim，因此不需要单独的目标机。

具体步骤如下所示：

第一步：安装Tornado，然后在安装目录下安装Wind ML 3.0。

第二步：配置WindML库。

使用WindML配置工具对配置文件进行修改。这里处理器选择为SIMNT。

第三步：编译链接生成库文件。WindML配置工具会在编译链接完成后在base path\target的lib目录下生成名为libwindml.a的库文件。

第四步：将库文件链接到应用程序，并使用WindML提供的API函数来开发图形界面程序。

下面就以上面建立的环境编写一个hello world程序，并通过对该例程的分析来认识WindML程序的框架结构。

（2）Hello world程序代码

```
#include <stdio.h>
#include <ugl/ugl.h>
#include <ugl/uglos.h>
#include <ugl/uglinput.
UGL_LOCAL;int He*声明函数声明*o/rl
UGL_LOCAL vol;ati布尔类型的变L
量 用于控制程序的结束* /
```

```
UGL_LOCAL UGL_INPUT_SERVICE_ID inputServiceId    * 定义输入设备的 ID */
static UGL_FONT_ID fontSystem                    * 定义系统字体 ID */
static UGL_FONT_DRIVER_ID fontDrvId              * 定义字体驱动 ID */
UGL_LOCAL struct _colorStruct                    * 以下是创建一个配色表 包含了两种颜色 */
    {
      UGL_ARGB rgbColor;
      UGL_COLOR uglColor;
    }
colorTable[] =
    {
    {   UGL_MAKE_ARGB(0xff, 0, 0, 0),},0 ,
    {   UGL_MAKE_ARGB(0xff, 168, 0, 0),},0
    };
#define BLACK                (0)
#define RED                  (1)
UGL_LOCAL void HelloPause(void)          * 循环体函数 用于接受并响应事件 */
    {
      while( stopHello)
        {
          UGL_MSG msg;
          if((uglInputMsgGet(inputServiceId,&msg,100) ==
UGL_STATUS_Q_EMPTY) &&msg.type == MSG_KEYBOARD)
              stopHello = UGL_TRUE;          * 响应键盘事件 */
        }
}
void hello(void)                         * 入口函数 创建并激活主任务 */
    {
      stopHello = UGL_FALSE;
      printf( To stop hello, type 'HelloStop'\n );
      uglOSTaskCreate( tWindMLHello, (UGL_FPTR)HelloWorld, 110,
                0, 10240,0,0,0,0,0)
}
void HelloStop (void)          * 调用该函数可以终止程序的运行 */
    {
      stopHello = UGL_TRUE;
}
UGL_LOCAL int HelloWorld (void) * 主函数 整个程序的生命线 */
    {
      UGL_GC_ID gc;                          * 定义图形上下文 */
UGL_DEVICE_ID devId;                     * 定义显示设备 ID */
UGL_ORD textOrigin = UGL_FONT_TEXT_UPPER_LEFT;
UGL_FONT_DEF systemFontDef;              * 定义系统字体 */
 * / uGL组件的初始化 包括输入输出设备的创建 注册 事件队列的创建与注册等 */
      uglInitialize()
 * 获得显示设备 ID */
devId = (UGL_DEVICE_ID)uglRegistryFind (UGL_DISPLAY_TYPE,
0, 0,0)->id;
inputServiceId = (UGL_INPUT_SERVICE_ID)
uglRegistryFind (UGL_INPUT_SERVICE_TYPE,  0, 0,0)->id  * 获得输入设备 ID */
fontDrvId = (UGL_FONT_DRIVER_ID)
uglRegistryFind (UGL_FONT_ENGINE_TYPE,  0, 0,0)->id * 获得字体驱动 ID */
      uglFontDriverInfo(fontDrvId, UGL_FONT_TEXT_ORIGIN, &tex-
tOrigin)
      uglFontFindString(fontDrvId," familyName=Lucida pixelSize =
24 , &systemFontDef)
      if ((fontSystem = uglFontCreate(fontDrvId, &systemFontDef))
== UGL_NULL)
        {
          printf( Font not found. Exiting.\n )
          return;
        }
gc = uglGcCreate(devId)                  * 创建图形上下文 */
* 使用配色表为系统配色 */
      uglColorAlloc (devId,  &colorTable [BLACK].rgbColor,
UGL_NULL,
                  &colorTable[BLACK].uglColor, 1)
      uglColorAlloc(devId, &colorTable[RED].rgbColor, UGL_NULL,
                  &colorTable[RED].uglColor, 1)
 * 通过取信号量的方法来锁住图形上下文 ,实现成批绘图 与 uglBatchEnd成为一对 */
      uglBatchStart(gc)
      uglBackgroundColorSet(gc, colorTable[BLACK].uglColor) * /
设置背景色为黑色 */
      uglForegroundColorSet(gc, colorTable[RED].uglColor)  * /
设置前景色为红色 */
      uglFontSet(gc, fontSystem)                  * 选择字体为系统字体 */
      uglTextDraw(gc, 300, 200, -1," hello world\n" )      * /输出文字 hello world /
      uglBatchEnd(gc)    * 解锁图形上下文 */
      HelloPause()       * 执行循环体函数 接受并响应事件 */
      uglGcDestroy (gc)  * 销毁图形上下文 */
      uglDeinitialize()  * 释放所有在初始化中创建的 ugl资源 */
      return(0)
    }
```

在 shell下输入 HelloStop或者按下键盘任意键就可以退出程序。

(3) hello world程序分析

下面就通过分析这不足一百行的小程序来认识 WindML程序的基本框架结构以及 WindML的运行机制。

总览整个程序 一,共分为五个部分 头部的包含及定义部分以及四个函数体。我们一一进行分析。

头部的包含及定义部分。包含了四个头文件

```
#include <stdio.h>标准输入输出库
#include <ugl/ugl.h>声明了 ugl的大部分 2的 API
#include <ugl/uglos.h>声明了 ugl的 VxWorks系统抽象的 API
#include <ugl/uglinput.h>声明了鼠标键盘等输入设备的 API
```

然后重要的一部分是定义了一个 colorTable这为主程序的颜色配置提供了一个配色表。 WindML的图形界面是以像素为单位的 一,般是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 RGB值 并用其在配色表中的索引值来代表这种颜色。由于该例程只使用了黑红两种颜色 因此在配色表中只配置了这两种颜色。

函数 hello(void)这是整个程序的入口函数。在 shell下输入 hello就可运行该程序。该函数体中只使用了一个函数 uglOSTaskCreate(这是 ugl封装 VxWorks函数 taskSpawn( ),作用是创建并激活一个任务 这里就是创建了程序的主任务 tWindMLHello。

函数 HelloWorld(void)这是主任务的函数体 就姑且称其为主函数 该函数也的确称得上是整个程序的主体部分 , ulg

下转第 4页)

常进行数据提供了。

3 ATLAB的 DD功能

MATLA作为客户程序 它支持文本格式的数据传送 可以使用 MATLAB中的 DD客户端模块提供的函数与服务器应用程序进行通讯。这些函数包括 :

1) DD服务初始化函数 ddeinit()该函数调用格式如下 : channel=ddeinit (service,topic)其中 : service为 DD会话的服务器名称 , topic为会话的主题。如果 Channel的返回值不等于 链接成功 否则链接失败。

2) DD链接建立函数 ddeadv()该函数调用格式如下 : rc= ddeadv (channel,item,callback,upmtx,format,timeout)其中 : channel为 DD通道句柄 即上面初始化建立的函数。 item为 DD服务器中启动链接的项目名称。 callbac是回调函数 即服务器端一旦数据变化要调用的函数。 upmtx是一个矩阵 它保存服务器送来的数据 如果这个矩阵名字是一个空字符串 则建立一个温链。如果矩阵名字是非空字符串 则建立一个热链。 forma是传送的数据格式 默认值为 [1 0] timeout是一个数值 其单位为毫秒。如果这段时间内无法建立链接 此函数调用失败。

3) DD数据请求函数 ddereq。()该函数的调用格式为 : data=ddereq(channel,item,format,timeout)它向服务器应用程序请求数据。

4) DD数据发送函数 ddepoke()该函数的调用格式为 : rc=ddepoke(channel,item,data,format,timeout)它将客户端的数据写到 DD服务器指定位置 该位置由 item设定。

5) DD链接释放函数 ddeunadv()该函数的调用格式为 : rc=ddeunadv(channel,item,format,timeout)它释放 MATLAB与 DD服务器应用程序之间的链接。

6) DD链接终止函数 ddeterm。()该函数的调用格式为 : rc=ddeterm(channel)它终止 MATLA与 DD服务器应用程序之间的链接。

MATLA作为 DD客户机程序的工作过程如图 所示。首先 用 ddeinit函数与服务器建立对话 建立成功则该函数返回一个通道号。以后的操作均对这个通道号进行。然后用 ddeadv函数请求建立热链。 ddereq函数向服务器索要数据 ,

图 1 MATLA与 WinC的 DD通讯链接图

返回值是存有数据的矩阵。 ddepoke函数向服务器发送数据。传送结束后 用 ddeunadv函数请求撤消与服务器已建立的热链连接 , ddeterm 函数解除与服务器建立的对话。以下是 MATLA以热链方式与服务器通信的 文件清单。

```
channel=ddeinit('Server','Topic')    建立 DD对话 返回通道号 channel
rc=ddeadv(channel,'text1','disp(x)','x',[1 0]) 与服务器建立热链接
pause                                暂停
rc=ddepoke('channel','text2',result)  向服务器发送数据
rc=ddeunadv(channel,'text1',[1 0])    释放与服务器的热链接
rc=ddeterm(channel)                   终止与服务器的 DD对话
```

该程序运行时与服务器建立热链连接 然后进入暂停状态。服务器程序中一旦 text的内容有变化 则 矩阵获得变化后的数据 然后在函数 disp()中进行运算 并返回结果至服务器程序 text中。在实际应用中 , disp()函数可以换成任意复杂的数据处理函数和控制算法函数。

4结束语

通过 DD技术实现 MATLA与 WinC的数据通讯 充分发挥 MATLA计算仿真功能强大的特点 使其在实际控制中得到广泛应用 ,可大大简化控制系统中组态软件的复杂控制算法的编制及运算过程。

参考文献

[石莹 洪悦 钱晓龙 .MATLA与组态软件的数据交换技术 [J].仪器仪表学报 , 2003, (S1) : 337- 340

(上接第 4页)

的初始化 设备的驱动 注册 设置字体 系统颜色的配置 以及释放资源等的动作都在这里完成 ,它的结束标志着整个程序的结束。

函数 HelloPause(void)该函数体是一个无限循环 也是整个程序的心脏部分 提供了维持程序运行的动力。可称其为循环体函数。 UglInputMsgGet函数会不停地从消息队列中取得消息 通过消息的判断来完成不同的任务。其作用相当于 windows编程中的消息循环和窗口回调函数的作用。这里体现了 WindM的事件驱动机制 通过响应键盘按下 鼠标按下等事件来实现不同的任务。因此程序的复杂程度在这个函数中得到体现。该程序中只是响应了键盘的事件处理 当按下任意键时 退出循环体 最终结束整个程序。

函数 HelloStop(void) 这是最简单的一个函数 也起到退出程序的作用。如果在界面上设计了退出程序的功能 那么这个函数就不是必需的。

可见 一个 WindM程序的主体框架就是三个函数体 入口函数 创建并激活主任务 、主函数 主任务激活 便运行该函数 , 是整个程序的生命线、它的结束意味着整个程序的结束 、循环体函数 起到心脏的作用 通过不停地从消息队列中取得消息 , 来维持整个程序的运行 。)

4结束语

WindM显示技术是个复杂的系统 涉及了图形管理、事件管理、内存管理等各种技术 一个 hello word程序是不具备任何功能的。要想实现众多的功能 还需要在循环体函数中添加各种事件响应的内容。

参考文献

[1] WindML DDK Programmer's Guide 3.0.USA:Wind River Systems Inc,2002

[2] WindML SDK Programmer's Guide 3.0. USA: Wind River Systems Inc,2002

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

收稿日期 : 2006.6.14]

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

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

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
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

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

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

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