基于 VxWorks 操作系统的 WindML 图形操控界面实现方法*

胡　俊

（华中光电技术研究所武汉光电国家实验室　武汉　430074）

摘　要　通过使用 WindML 来实现界面显示，显示主要采用读取 24 位真彩色位图的方式来实现，背景用整个位图显示，界面中需改变的图像只需刷新局部显示内存，给出详细源代码，方法简单，并且界面美观，通用性强。

关键词　WindML；VxWorks；位图；图形界面；Tornado

中图分类号　TP391.41

A Method to Achieve Interface-display under VxWorks

Hu Jun

(Wuhan National Laboratory for Optoelectronics, Huazhong Institute of Electro-Optics, Wuhan　430074)

Abstract　Using WindML to achieve Interface-Display under VxWorks operating system, and the way to realize Interface-Display is to read 24bit bitmaps into buffer, and then display it. Providing particular program codes, the method is simple and the interface is beautiful, and can be widely used.

Key Words　WindML, VxWorks, Bitmap, graphical-Interface, Tornado

Class Number　TP391.41

1　引言

在实际的工程应用中，人机界面要实时显示状态数据，并且要做到美观大方，如果将界面提前在 Windows 用一些画图软件下做成精美的 24 位的真彩色位图，显示的时候，只需读取图片，将图片作为背景显示，当状态数据改变时，只需改变该区域显示内存就可以了。对于简单符号和数字的显示也可以通过读取位图的方式来实现。这不失为 VxWorks 下界面显示的一个简单有效的方法。

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，以及支持开发任意从提供的“通用”代码快速创建新驱动程序的 API。WindML 的层次结构[2]见图 1。

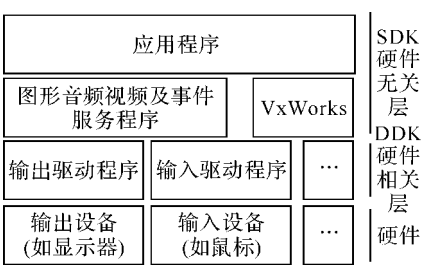

图 1　WindML 的层次结构图

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相应区域和窗口的绘制等等功能。

· 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：1）通过 Tornado 下的 WindML 配置工具配置 WindML；2）通过命令行的方式配置 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、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。

编译成功后，将在\\target\\lib*cpuXX**CPUXX*\\gnu\\文件夹下生成 wndml.o 文件（*cpuXX* 为 BSP 选择的 CPU 型号），将此文件加载进 Macros 里的 EXTRA_MODULES 里，并指明相应的路径。

· 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 映象里。

其中要注意两点：1）在 Tornado 的开发环境中 VxWorks 组件配置中加入 WindML components；2）在工程对应的 BSP 中的 config.h 文件的最开始要加入如下代码“#define INCLUDE_WindML”，将 WindML 包含进来，如果不加入，则不支持 WindML，则无法显示图像。

· 24 位位图的读取

BMP 图像格式是 Windows Bit Map 的缩写，24 位 BMP 格式的图像文件包含的信息是最完整的，本文主要进行 24 位真彩色位图的读取和显示。

在 24 位真彩色 BMP 图像文件中，先是位图文件头，接着是位图信息头，文件的最后部分由实际的像素位组成。它不含有颜色表，因为像素位区域里每一个像素由 24 位组成，为 3 个字节的长度，每个字节分别代表红、绿和蓝的颜色值，这样就可以描述三个颜色的 256 个值，由这三个颜色值可以匹配到各种颜色。像素位区域是由图像的底行开始，沿着图像向上增长的水平行组织的。

24 位真彩色位图文件中包括：文件标识、位图信息头的长度、位图的水平宽度、位图的垂直高度、位图的位平面数、图像区数据的大小、水平每米有多少像素、每个像素用多少位表示、图像所用的颜色数，图像数据区等所需的重要信息，对于整个过程的实现是必不可少的。通过对参考 24 位真彩色位图文件[3]，可以根据偏移量和字段大小找到所需相应的信息，便于读取显示，在此不再赘述。

要读取图片文件，可以使用 fread()函数来进行读取。为了能够按字段读出二进制文件里的每一个数据，可以按照结构体的形式来读取。定义结构体的关键问题是数据类型的限制。诸如 LONG，DWORD，WORD 等类型，在 Tornado 中是无法编译通过的，必须要做如下定义：

```
#define DWORD unsigned int
#define LONG long
#define WORD unsigned short
```

另外在 Tornado 中，最小的单位是 4 个字节，但是位图读取的时候最小单位是 1 个字节，因此，直接按结构体读取的时候是不对的，所以在我们用 C 语言定义 BITMAPINFOHEADER 结构体和 BITMAPFILEHEADER 结构体的时候，必须在定义结构体之前写上“#pragma pack(1)”，将其转换为按 1 字节单位读取，结构体定义结束后，写上“#pragma pack(4)”，恢复为 4 字节的单位。

用 fopen()函数打开位图文件，再使用 fread()函数先后读取 BITMAPFILEHEADER 结构体信息和 BITMAPINFOHEADER 结构体信息，才能读取图像数据。将图像数据读取进分配好的内存空间。

由于 24 位图的图像数据是 3 个 8 位分别表示(R，G，B)，而 VxWorks 下的显示是 4 个 8 位分别表示(R，G，B，α)，因此需要进行转换。这里 α 的值为 0xFF，将每一个像素点多加入 α 即可。

24 位位图的像素位区域是由图像的底行开始，沿着图像向上增长的水平行组织的。但是 vxworks 的显示是由屏幕的最上一行开始，沿着图像向下增长的水平行组织的。因此也需要作转换。可以用 memcpy()逐行转换即可。

· 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，必须调用函数 uglInitialize()完成初始化。接下来获取显示设备 ID，获得输入设备 ID，并创建图形上下文。

· 图形界面的显示

由于这里是使用读取 24 位位图，进行图像处理，然后再显示的方式，因此不需要配置颜色表。将整个图片的内存空间显示在屏幕上，WindML 提供了完整的 API 函数，即 WindML Bitmap API。首先使用 uglBitmapCreate()函数创建标准的 DDB (Device Dependant Bitmap)，再调用 uglBatchStart ()锁住图形上下文，uglBitmapBlt()用于在屏幕上显示标准的 DDB，uglBatchEnd 解锁图形上下文。

如果显示的图像有变化，只需找到变化的部分对应的内存空间，修改内存空间后，先调用 uglBitmapDestroy()销毁之前的 DDB，再重新创建 DDB 重复上述过程即可。

3 程序实例

下面通过一个程序实例来简单说明一下利用 WindML 进行位图读取和显示的图形界面开发过程。下面给出程序框图和部分程序代码并加以说明。

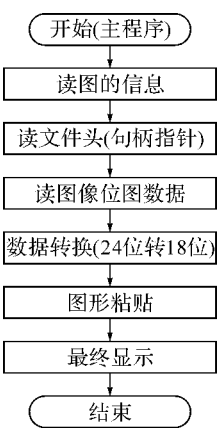

图 2 程序流程框图

位图文件的读取子程序。对位图文件中重要信息包括图文件的信息头，大小，并将把 24 位位图改成适合 VxWorks 显示的 32 位。VxWorks 显示是从屏幕的左上角开始显示，24 位图读取是从左下角开始读取，所以要进行转换。

```
BOOL LoadPic(char * pPicFileName , int * nWidth , int * nHeight , unsigned char * * ppData) /*读位图图片文件*/
{
    FILE * fp;
    BITMAPFILEHEADER FileHeader;
    BITMAPINFOHEADER InfoHeader;
    int value;
    BOOL bRet;

    UGL_UINT8 *pnewData;
    UGL_UINT8 *poldData;

    bRet = TRUE;
    fp = fopen(pPicFileName, "rb");
     memset ( &FileHeader, 0, sizeof ( BITMAPFILEHEADER));
    fread ( &FileHeader , sizeof(BITMAPFILEHEADER),1 ,fp);

     memset ( &InfoHeader, 0, sizeof ( BITMAPINFOHEADER));
    fread ( &InfoHeader ,sizeof(BITMAPINFOHEADER),1, fp);
    /*缓冲区分配内存初始化*/
    pnewData = calloc(InfoHeader.biWidth * InfoHeader.biHeight * 4 , 1);
    poldData = calloc(InfoHeader.biWidth * InfoHeader.biHeight * 4 , 1);

    if (bRet)
    {
        /*读取图像数据*/
        value = fread( pnewData , 1 , InfoHeader.biWidth * InfoHeader.biHeight * 3 , fp);

        /*把 24 位位图改成适合 VXWORKS 显示的 32 位*/
        for( i = InfoHeader.biWidth * InfoHeader.biHeight - 1 ; i>=0 ; i--)
        {
            *(poldData + i* 4 + 0) = *(pnewData + i * 3 + 0);
            *(poldData + i* 4 + 1) = *(pnewData + i * 3 + 1);
            *(poldData + i* 4 + 2) = *(pnewData + i * 3 + 2);
        }
        for( i = InfoHeader.biWidth * InfoHeader.biHeight - 1 ; i>=0 ; i--)
        {
            *(poldData + i* 4 + 3) = 255;
        }

            memcpy(pnewData , poldData , InfoHeader.biWidth * InfoHeader.biHeight * 4 );

        for( i = 0 ; i< InfoHeader.biHeight ; i++)
        {
            memcpy( poldData + i * InfoHeader.biWidth * 4,
                    pnewData + InfoHeader.biWidth * ( InfoHeader.biHeight - 1 - i ) * 4 ,
                    InfoHeader.biWidth * 4 );
        }

        cfree(pnewData);
```

```
        *ppData = poldData;

        *nWidth = InfoHeader.biWidth;
        *nHeight = InfoHeader.biHeight;
    }
    fclose (fp);

    return bRet;
}
```

·显示背景初始化程序 BOOL InitScreen()，**包括显示背景图片的宽度、高度等信息。**

```
pBKData = NULL;
    LoadPic ( "ata0/main.bmp", &bkWidth, &bkHeight, &pBKData);

    /*给 screenDib 赋值*/
    screenDib.width =SCREEN_WIDTH;
    screenDib.height =SCREEN_HEIGHT;

    以及获取显示设备 ID
    devId = (UGL_DEVICE_ID)uglRegistryFind (UGL_DISPLAY_TYPE, 0, 0,0)->id;
    获得输入设备 ID
    inputServiceId = (UGL_INPUT_SERVICE_ID)uglRegistryFind (UGL_INPUT_SERVICE_TYPE, 0, 0,0)->id;
```

· 在完成所有位图文件中缓存，指针等变量的使用后要释放所有在初始化中创建的 UGL **资源，并销毁之前的** DDB

```
Void UninitScreen()
{
    /*销毁 DDB destroys a Device Dependant Bitmap (DDB)*/
    uglBitmapDestroy(devId, testBitmap);
    /*销毁图形上下文*/
    uglGcDestroy (gc);
    /*释放所有在初始化中创建的 UGL 资源*/
    uglDeinitialize();
}

void refreshScreen()
{
    uglBitmapDestroy(devId, testBitmap);
    testBitmap = uglBitmapCreate(devId, &screenDib, UGL_DIB_INIT_DATA, 0,
    UGL_NULL);
    uglBatchStart(gc);
    uglBitmapBlt(gc, testBitmap,
    0,0,SCREEN_WIDTH, SCREEN_HEIGHT,
    UGL_DEFAULT_ID,
    10, 10);
    uglBatchEnd(gc);
}
```

本例是利用 WINML **读取** 24 **位位图，作为背景显示出来，当需要改动时，读取其他位图，覆盖在存储背景位图的内存的相应位置，重新刷新输出即可。程序具有很强的通用性，可以直接移植到其他地方直接使用。**

4 结语

本文详细介绍了一种基于 VxWorks **的** WindML **图形界面实现方法，实现了** 24 **位真彩色位图在** VxWorks **下的读取和显示，上面提到的图形界面实现方法已经被作者应用于实际的海军装备工程开发，并取得了比较好的效果。**

参考文献

[1] 谭浩强. C 语言程序设计[M]. 北京：清华大学出版社，1999

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

[3] 陈传波，金先级. 数字图像处理[M]. 北京：机械工业出版社，2005

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

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

[6] 王学龙. 嵌入式 VxWorks 系统开发与应用[M]. 北京：人民邮电出版社，2003：194～252

[7] 周启平. 张杨 VxWorks 程序员速查手册[M]. 北京：机械工业出版社，2005：187～200

[8] 赵刚，张翀，江勇. 计算机图形显示加速及实现技术[M]. 北京：电子工业出版社，2009

[9] 陈传波，陆枫. 计算机图形学基础[M]. 北京：电子工业出版社，2002

[10] 何永前，陈建勋，李建璜，等. C 程序的动态测试[J]. 舰船电子工程，2009，29(8)

[11] 伏玉琛，周洞汝. C 语言程序设计[M]. 武汉：华中科技大学出版社，2003

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

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

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

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

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

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

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

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