WindML工业平台下开发 S1d13506驱动及显示功能的实现

Development of S1d13506 Driver and Display Programm Under WindML

张 雷 任恩恩 兰州交通大学光电技术与智能控制教育部重点实验室 甘肃 兰州 730070)

摘 要

讨论的内容是使用风河公司提供的 WindML来开发爱普生公司生产的 S1d13506显卡芯片的驱动 并编写显示程序。在整个开发过程中 ,WindML提供了快捷便利的编程环境和程序框架。使用 WindML提供的 API函数 可以开发出用户自己定制的应用程序。同时 ,WindML给出了通用的显卡驱动程序框架 编程者可以参考这些程序来开发显卡驱动。显卡驱动配置好以后 通过编译工具编译 生成相应的驱动模块 加进 VxWorks镜像中 便可以实现显卡的驱动以及建立人机交互的环境。

关键词 :WindML, PID, DDK, SDK, vxworks, AT91RM9200静态存储管理器

Abstract

This paper introduces how to use WindML to develop the diver of s1d13506,which is manufactured by Epson.WindML is supplied by wind river Inc.WindML supports a convenient environment to develop the customed application and distributes the general diver of dsiplay chip.After the configuration of the diver,you can add the application module to the vxWorks image to realize the function you want.

Keywords:WindML,PID,DDK,SDK,vxworks,AT91RM9200,SMC

VxWorks操作系统是现今使用比较广泛的一种嵌入式实时操作系统 由美国风河公司开发。由于当今各种工业平台的需求 风河公司开发了 PID(PlatForm- for- Industrial- Devices)来支持各种工业平台的应用 ,WindML(Wind Media Library)便是专门针对在嵌入式操作系统上的多媒体应用而开发的软件包。它为在各种操作系统下开发标准的多媒体设备驱动提供了一个框架 同时也为有如图像、声音、视频技术的应用开发提供了便利。本文讨论的内容是在 VxWorks操作系统下 开发爱普生公司生产的 S1d13506芯片驱动的及相应的显示程序。开发使用 WindML提供的软件包 以及相应的 API函数。通过这些 API函数 用户可以方便地开发不同的 CPU型号下各种显卡和声卡的驱动以及应用程序。

1 WindML的体系结构

WindML包含两个组件 ,一个是软件开发包 (Software Development Kit, SDK),一个是驱动开发包 (Driver Development Kit, DDK)。软件开发包提供了广泛的 API函数用于图形以及输入处理 多媒体 字体以及内存管理。驱动开发包提供了常用的参考驱动 用户可以在此基础上完成各种多媒体硬件的配置 并最终开发出自己的硬件驱动程序。驱动开发包是可以按照用户的需求进行扩展和定制的。在 WindML提供的驱动开发包中包括很多硬件驱动文件 种类涵盖图形、视频、声音、字体以及窗口管理。 WindML的框架结构如图 所示。

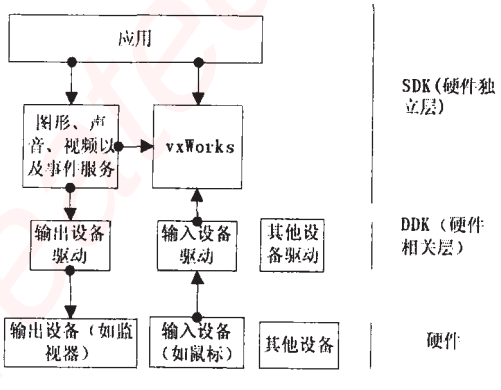

图 1 WindML框架结构

2 使用 WindML开发显卡驱动

2.1 在 BSP中的相关配置

在使用 WindML开发显卡驱动前 首先需要对 BSP文件夹下的 config和 syslib进行配置。要在 config.h中定义包含使用 WindML库的宏 #define INCLUDE_WINDML,然后要在 syslib.中对 WindML所使用的内存区进行配置 ,即在 sysPhysMemDesc的结构数组中 要对内存区的首地址 堆栈的大小以及内存区的一些操作进行配置。同时在 syslib.中还要包括 WindML与系统接口文件 sysWindML.,同时要在 gpio.中对 WindML使用的 I/O管脚进行配置 ,同时还要对 SMC(AT91RM9200中集成的静态存储管理器 进行配置。

2.2 开发 S1d13506驱动的步骤

2.2.1 驱动文件 ude506Direct介绍

在本文中 要开发 S1d13506的驱动 并且要建立人机交互的环境。因此 在编写相应显卡驱动的同时 还要编写显示器的显示程序。开发 S1d13506的驱动 要在 install/target/src/ugl/diver/graphics/Epson/s1d13506/16bit下找到相应的驱动源文件 ude506Direct.进行具体的配置和修改。由于本文中的输入设备没有使用 WindML开发 因此可以屏蔽掉输入设备的配置这一块。在该文件中有两个函数很重要 ,一个是 uglEpson13506_16BitDevCreate(UGL_UINT32 instance, UGL_UINT32 notUsed0, UGL_UINT32 notUsed1),它主要完成对 s1d13506驱动的初始化并返回一个设备控制结构的指针 UGL_UGI_DRIVER*。UGL_UGI_DRIVER结构中定义了开发驱动程序所需的所有的 API函数的指针 程序编写者可以添加自己的 API函数指针到该结构中来完成对驱动的定制。 WindML中还包含一个 UGL_GENERIC_DRIVER结构 它包含通用的驱动程序所需的 API函数指针。 uglEpson13506_16BitDevCreate(UGL_UINT32 instance, UGL_UINT32 notUsed0, UGL_UINT32 notUsed1)函数的具体工作是完成显卡设备的获取 内存空间的分配 以及 UGL_UGI_DRIVER结构的初始化 并返回指向该结构的指针。显卡设备的获取是通过调用 uglGraphicsDevOpen函数 在 uglEpson13506_16BitDevCreate()函数中被调用 来实现 该函数在 udcHwAbsLayer.中定义 而该函数又

(UINT32 devType, UINT32 instance, UINT32 vendorID, UINT32 deviceID)函数来完成系统对设备的初始化。这样，底层驱动函数和系统函数之间的调用关系便建立起来了。

ude506Direct文件中另一个主要的函数是 uglEpsonSetRegs(int mode)，它完成对显卡设备寄存器的配置。爱普生公司提供了两个标准的批处理文件 mode640x480x60.dat和mode800x600x60.dat，这两个文件和 ude506Direct在同一个文件夹下。通过这两个文件，可以完成对 s1d13505显卡在两种显示模式下的寄存器配置，十分方便。

2.2.2 WindML的配置

WindML提供给用户的都是一些源文件，因此如果要在系统中使用 WindML提供的功能，必须对其进行配置和编译。配置的方法很简单，打开 Tornado编译环境，在 Tools下拉菜单中，选择 WindML，然后出现配置对话框，点击 config，出现 WindML Config对话框，选择 File下拉菜单建立自己定义的配置文件，然后选择适当的处理器和编译工具，并将 Additional Builds下的选项全部选上，在 Devices下的配置，要根据用户使用的芯片厂商提供的批处理文件来进行配置，本例中是爱普生公司提供的批处理文件，所有选项配置好以后，进行编译，编译将在 installl\target\lib\arm\ARMARCH4\common下生成一个 wndml.o模块，在最终生成 vxWorks镜像时，要在 EXTRA_MODULES选项中加入 $(WIND_BASE)\target\lib\arm\ARMARCH4\common\wndml.o。WindML的配置对话框如图所示。

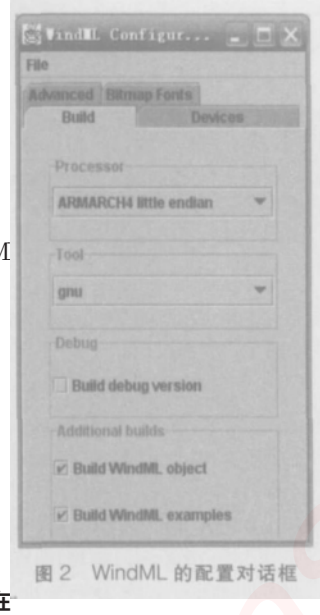

图 2 WindML 的配置对话框

3开发 RT的显示程序

完成了对显卡驱动的配置，还需要编写相应的显示程序。对于 WindML来说，这里的显示程序相当 WindML下的一个应用程序的开发。本文中定义的显示程序文件名为 at91rm9200vga.c，和系统提供的 pcConsole.c文件一起放在 install/target/src/drv/serial下。

BSP中的 bootconfig.c文件完成显示器的初始化函数 pcConDrv的调用，在 pcConDrv中调用 pcConDrvHrdInit()完成对显示设备的初始化，这里我自己定义了一个函数 vgaHrdInit()来完成对显示环境的初始化。vgaHrdInit()函数在 pcConDrvHrdInit中被调用。函数实例如下：

```
STATUS vgaHrdInit (void)
{
  UGL_REG_DATA *pRegistryData;
  UGL_FONT_DRIVER_ID fontDrvId;
  UGL_FONT_DEF fixedFontDef;
  UGL_MODE_INFO modeInfo;
  if(uglInitialize()==ERROR) /*初始化 UGL*/
    return(ERROR);
  pRegistryData = uglRegistryFind (UGL_DISPLAY_TYPE, 0, 0, 0); /*获得显示设备的标识符*/
  {
  uglDeinitialize();
            return(ERROR);
            }
      devId = (UGL_DEVICE_ID)pRegistryData->id; /*UGL_DEVICE_ID就是一个指向 UGL_UGL_DRIVER结构的指针*/
  pRegistryData = uglRegistryFind (UGL_FONT_ENGINE_TYPE, 0, 0, 0); /*获得字体驱动的标识符*/
      if (pRegistryData == UGL_NULL)
      {
          uglDeinitialize();
          return(ERROR);
      }
      fontDrvId = (UGL_FONT_DRIVER_ID)pRegistryData->id;

  uglInfo(devId, UGL_MODE_INFO_REQ, &modeInfo); /*设置显示模式*/
  displayWidth = modeInfo.width;
  displayHeight = modeInfo.height;
    uglColorAlloc (devId, &colorTable [BLACK].rgbColor, UGL_NULL, &colorTable[BLACK].uglColor, 1); /*为黑色进行初始化*/
    ......
  ....../*UGL定义了颜色数组，可以对其中定义的颜色进行初始化*/
   uglFontFindString (fontDrvId, "pixelSize = 15 name=Fixed", &fixedFontDef); /*生成字体，并规定其大小*/
   if ((fontEcho = uglFontCreate (fontDrvId, &fixedFontDef)) == UGL_NULL)
        {
          printf("( Font not found. Exiting.\n");
          return(ERROR);
        }
  /*生成整个图形环境的标识，在后面很多子函数中把该标识当作参数，在当前的显示上下文中进行操作*/
    gc = uglGcCreate(devId);
        vgaClear (gc); /*清除屏幕*/
   cursorInit;(); /*生成光标*/
  ....../*建立光标闪烁的任务*/
}
```

完成了显示环境的初始化，还有一个重要的函数 int vgaWriteString(FAST PC_CON_DEV *pPcCoDv)用来完成字符的显示。函数实例如下：

```
int vgaWriteString(FAST PC_CON_DEV *pPcCoDv)
{
  ......
  int nBytes=0;
  while (RNG_ELEM_GET (ringId,&ch,dummy) == 0) /*从环形缓冲队列中取字符*/
  {
        nBytes++;
       if(ch!=8) /*如果不是退格键*/
         {
             textEcho(ch); /*显示字符*/
         }
          else
         {
                  x-=textWidth;
                  charClear(gc); /*清除该字符*/
          }
    }
```

操作通常用中断请求来通知 A/转换结束 并需要在下一次A/执行前将转换结果转存到另一位置。但是 ADC12有 12个转换寄存器 使得 A/可以进行多次转换而不需要软件干预。与一般的 AD比较 它具有高速通用的特点 特别适合精密的数据采集和转换。 ADC12有 8个外部模拟信号采样通道和 4个内部通道 各个通道的参考电平可以由用户根据需要编程选择。本温度计根据实际需要使用了 2个外部信号采样通道并选用内部参考电平 2.5V

MSP430F13具有 FLASH存储器 这一特点使得它的开发工具相当简便。利用单片机本身带有的 JTAG接口 可以在一台PC机及一个结构小巧的 JTAG控制器的帮助下实现程序的下载 完成程序调试 方便了软件设计。

2.数字温度传感器 DS18B20

平衡式温度计是利用热电偶将温度转换为电势作为信号输入的。只有当热电偶冷端温度保持不变时 热电势才是被测温度的单值函数。在应用时 由于热电偶工作端与冷端距离很近 冷端又暴露于空间 容易受到周围环境温度波动的影响 因而冷端温度难以保持恒定 为此需对热电偶冷端进行补偿。传统做法一般是用电桥补偿 改进前的温度计正是完全靠硬件对热电偶进行温度补偿 使得电路非常复杂 调试困难。现在采用特殊的温度补偿方式 利用数字温度传感器 DS18B20进行实时的环境温度测量 在软件上补偿热电偶冷端受到的环境温度的影响 大大简化了电路 改善了非线性问题。

DS18B20是 DALLAS公司生产的单线智能温度传感器 属于新一代适配微处理器的数字传感器 具有小体积封装形式 全部传感元件及转换电路集成在形如一只三极管的集成电路内 ;支持 3V～5.5V的电压范围 使系统设计更加灵活、方便 温度测量范围为 -55℃～+125℃ ,平衡式温度计使用时的环境温度完全在其测量范围内 可编程为 9～12位 A/转换精度 测温分辨率可达0.0625℃ 符合平衡式温度计精度要求 独特的单线接口方式 ,在与单片机连接时仅需要一条口线即可实现单片机与DS18B20的双向通讯 被测温度用符号扩展的 16位数字量串行输出送入单片机内。

2.3 16位数模转换器

本温度计选用 AD公司的 16位数模转换器 AD421将比较稳定可靠 外接运算放大器后可输出 -5～+5V电压 从而实现了表盘指针的双向偏转。

3软件设计要点

3.1访问 DS18B20

由于 DS18B20采用的是单线总线协议方式 即在 数据线上实现数据的双向传输 而单片机硬件上不支持单总线协议 ,因此必须采用软件方法来模拟单总线的协议是序来完成对DS18B20芯片的访问。由于 DS18B20是在一根 I/线上读写数据 因此对读写的数据位有严格的时序要求 它有严格的通信协议来保证各位数据传输的正确性和完整性。该协议定义了几种信号的时序 初始化时序、读时序、写时序。所有时序都是将单片机作为主设备 ,DS18B20作为从设备 ,而每一次命令和数据的传输都是从主机主动启动写时序开始 如果要求 DS18B20回送数据 在进行写命令后 主机需启动读时序完成数据接收。

3.2软件实现温度补偿

DS18B20将测得的环境温度以 16位数字量串行送入单片机 通过软件计算出其对应的温度值 查表找出这个温度值对应的热电偶电势 用最小二乘法拟合成热电偶信号所应转换成的A值 从而达到对热电偶温度补偿的目的。

3.3软件实现温度测量

相较于改进前温度计完全用硬件实现测量功能 ,本温度计利用软件拟合温度与 A值的关系要简单得多。用最小二乘法拟合出由热电偶电信号转换成的 A值与温度的多项式即可。

4结束语

本文介绍的平衡式温度计以 MSP430F13为核心 以DS18B20作热电偶温度补偿 ,用很少的外部元件实现温度测量 简化了硬件电路 减少了故障发生率 数字化冷端补偿 降低了仪器调试难度 保证了仪器要求的精度 符合原测试规程。

参考文献

[1]胡大可 .MSP43系列 Flash型超低功耗 16位单片机 [M].北京 北京航空航天大学出版社 ,2001
[2]王化祥 张淑英 传感器原理及应用 [M].天津 天津大学出版社 ,2004
[3]沙占有 智能化集成温度传感器原理与应用 [M].北京 机械工业出版社 ,2002

收稿日期 :2006.7.9]

(上接第 47页)

```
    uglCursorMove(devId,(x+1),(y- textHeight+5));/* 移动光标 */
                ......
    return(nBytes);
}
```

textEcho()函数为编程者自己定义 主要负责将普通字符显示出来 其中用到的 specialKey(char key)函数 该函数用于识别特殊字符和普通字符。

同样 也是在 bootconfig中完成对显示器所需输入输出缓冲区大小的设定 通过 STATUS pcConDevCreate (char* name, FAST int channel int rdBufSize int wrtBufSize)函数 该函数通过调用 tyDevInit (&pPcCoDv->tyDev, rdBufSize, wrtBufSize, vgaWriteString)完成对 TY_DEV结构的初始化 并调用 iosDevAdd (DEV_HDR* pDevHdr char* name int drvNum)将输入输出设备添加到系统当中。

4结束语

本文已经实现了 s1d13506驱动的配置 并且建立了人机交互界面 运行正常。 WindML为所有用户提供了一个简单便捷的开发图形应用及显卡驱动的环境 通过该软件包中提供的通用驱动框架 可以使用户快捷的开发特定环境下的芯片驱动。并且使用 WindML提供的 API函数 也可以非常方便的开发相应的应用程序。但用户除了要编写相应的应用程序外 还需要编写与操作系统接口的文件 在本例中是 sysWindML.c用户还需要对相应的内存区进行设置。在所有用到的文件编写完成后 要使用 WindML配置工具生成配置模块 在编译镜像的时候将该模块添加进来 实现用户需要的显示功能。

参考文献

[1] WIND MEDIA LIBRARY Programmer's Guide- DDK[Z].USA:WindRiver Systemn,Inc,2003
[2] WIND MEDIA LIBRARY Programmer's Guide- SDK[Z].USA:WindRiver Systemn,Inc,2003
[3] Wind River Platforms Getting Started [Z].USA:WindRiver Systemn,Inc,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

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

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

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

Programming:

1. 计算机软件基础数据结构 - 算法