基于 VxWorks 实时控制系统中文交互界面开发平台

查振羽， 熊华钢

(北京航空航天大学 电子信息工程学院，北京 100083)

摘　要：针对基于 VxWorks 的实时控制装置的中文交互界面的实现，抽象出此类交互界面通用的界面元素的概念和相互之间的关系，在此基础上给出针对此类交互界面的开发平台。该平台下层用到 VxWorks 下的 WindML 库，采用了界面元素管理和以事件为驱动的运行机制，提供上层库函数用于具体构建不同需求的界面。

关键词：VxWorks； WindML； 实时控制系统； 嵌入式 GUI； 交互界面框架

中图法分类号：TP316.2　　**文献标识码**：A　　**文章编号**：1000-7024 (2009) 08-1875-04

Development platform for real-time control system Chinese UI based on VxWorks

ZHA Zhen-yu， XIONG Hua-gang

(School of Electronic and Engineering, Beijing University of Aeronautics and Astronautics, Beijing 100083, China)

Abstract：According to the implementation of real-time control system Chinese UI based on VxWorks， some concepts of these common-use UI element and relationships between them are abstracted， then the development platform for this kind system Chinese UI on these concepts is developed. The platform， which used the WindML library of VxWorks at the bottom layer and is organized by the mechanism of UI element manage and event-driving， provided interfaces for upper layer to build certain UI.

Key words：VxWorks； WindML； real-time control system； embedded system GUI； interface framework

0　引　言

VxWorks 是美国 WindRiver 公司开发的专门为实时嵌入式系统设计开发的操作系统内核，提供了高效的实时多任务调度、中断管理、实时的系统资源以及实时的任务间通信，由于其高可靠性，实时性和可剪裁性，因此被广泛的运用于可靠性和实时性要求较高的领域的各类实时控制系统中。

WindML(wind media library)是 VxWorks 提供的一个可裁减的多媒体库，用来提供给开发者一些基本的图形和音频视频应用的支持。WindML 由 DDK(硬件开发包)和 SDK(软件开发包)组成，前者提供实现硬件相关的驱动，后者提供的是与硬件无关的通用 API。

具体层次结构如图 1 所示。

1　实时控制装置交互界面

针对一个实时的控制装置，交互界面的任务一般都需要完成下面几个内容：

(1) 图形界面输出：作为控制装置，需要显示的内容包括发控命令状态、系统自身的运算结果、受控设备的反馈数据和状态等，在屏幕上显示的基本数据类型包括数字(返回的数据显示，包括浮点型或者整形)、文本(命令信息和提示信息等，能实现中英文混合显示)、用颜色和形状的不同属性显示不同

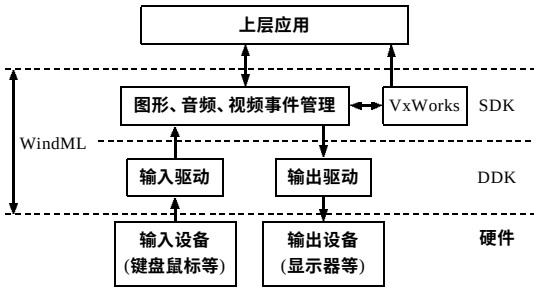

图 1　WindML 层次结构

信息的区域（比如用圆形区域的红色或者绿色模拟红绿灯用来表示预警或者正常状态)等，以及上述组合(比如需要同时变化的显示文本和同时变化颜色表示状态)；

因此，显示界面上开辟一个个的区域，作为显示的基本单元用于显示数据，这些显示的区域的属性包括：形状、颜色、大小、位置等，内部显示的数据可以有汉字和数据等不同类型，这些区域内，可以作为固定显示单元接受从其他任务传来的数据进行存储和显示，也可以作为可输入单元接受输入部分的输入。这些区域是组成整个显示的基础元素。

(2)键盘鼠标输入部分：能够交互界面任务与程序其他任务之间的数据交换。其中，鼠标可以选中上面可以被选中的区域，选中后可以利用键盘进行数字的输入(包括数字、小数

点、空格、回车),键盘中某些按键可以在按下后实现特定的功能,这样以便把键盘一些按键映射成有特殊用途的发控键盘按键,在开发时候可以直接针对通用键盘按键进行开发即可;

(3)与其他任务的交互:在VxWorks下,任务之间的数据交换通过消息、全局变量共享内存等各种形式进行,通过信号量等进行进程的同步。同时,由于控制系统的实时性要求,因此交互界面的优先级应该放在网络传输、解算等关键任务之后。

构建具体的用户界面:①需要使用低级图形库在屏幕上建立交互式图形对象;②维护这些图形它们所表示的应用数据间的动态关系;③需要针对不同的操作,应用数据内容必须作相应改变,操作包括操作者输入以及其他任务的交互。因此涉及到显示相关信息的数据表示,屏幕表示以及控制3个部分,也就是MVC(model view controller)模型,此模型将应用系统的输入、处理、输出流程进行分离。

因此,对于一个实时控制装置的交互图形界面的开发,可以在WindML之上建立一层针对上述应用的开发平台,由WindML提供低层的低级图形库支持,上层提供给具体应用的交互界面接口,用于具体界面的建立。

由于WindML和VxWorks已经将底层硬件进行抽象,本身具有比较成熟的移植性,因此建立在其上的开发平台能够在多种硬件平台上运行。

具体层次可如图2所示。

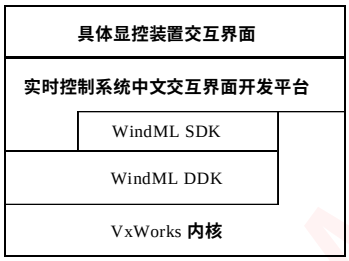

图2 图形界面中间层

对上述需求,我们具体抽象出几个对应概念,构成开发平台的基础构件,我们称之为构件层。

这一层主要作用在于定义界面系统基本元素的内部结构,内外部的操作以及外部接口,提供给MVC层中各个对应部分使用。

1.1 界面显示元素的结构定义和相关操作

显示的基本单元,可以看成是一个个的"控件",有以下的属性:形状,位置,大小,颜色,储存的数据区,即存储在控件内部的并且在需要时候进行显示的数据;

若干个相关的控件(概念相关,比如是显示反馈数据的若干属性),可以用链表结构连成一个"帧",用一个惟一的帧号进行区别。显示的时候或者操作的时候,这些相关控件的数据传输,便可以以帧为单位进行数据的传递。这样保证画面刷新的效率同时由于属性相近而发送任务便于管理。帧的属性有帧号,指向下属所有控件的链表指针等;

而几个帧,如果在显示区域的相同位置,可以归于一个"区域",把相关帧用链表结构进行串接,在需要切换显示的时候,通过帧号进行显示。这样在一个区域内可以实现多帧切换,所以区域有下列属性:所拥有帧的链表,当前显示的帧号,同时,帧的属性里有所属的区域的指针。

三者的关系结构示例如图3所示。

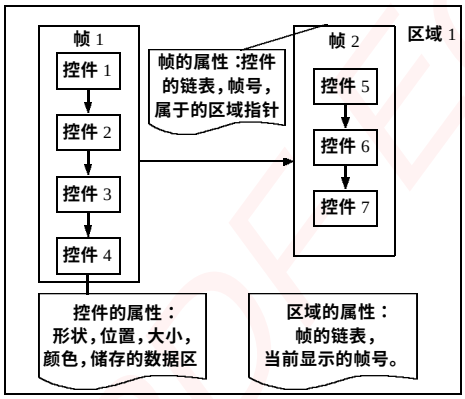

图3 控件、帧、区域关系

界面元素管理器(GUI elements manager)用于维护和管理界面上所有的元素的构成,其中元素包括控件,帧和区域。能够实现的接口操作有元素的建立,新建一个控件,帧或者区域的函数,相互关系的建立和修改,包括帧或者区域加入链表,删除元素等操作。

同时提供控件帧或者区域的刷新显示操作,通过控件的属性,调用WindML中底层的2D绘图函数进行显示。

1.2 可被选中的控件数据结构和相关操作

可被选中控件指的是能够被鼠标点击或者键盘上下左右定位选中可以被输入数据的控件。维护一个指向控件指针的二重双向链表,指向对象为可供输入的控件,示例如图4所示。

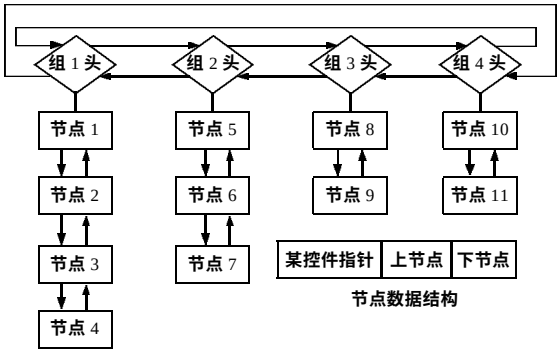

图4 可被选中的控件二重双向链表

其中每一个组头,维持一个双向链表,而这个双向链表中节点属性包括指向控件的指针,以及上下节点的指针。

可被选中控件管理器维护这个双向链表,并且提供下述接口操作:

对应键盘输入上下按键对应选中同一组上下节点的控件。如果到了头或者尾,则跳到上一组的最后一个节点或者下一组的首个节点。

组头之间构成双向链表,针对左右键响应,选中左右组的首个节点的控件。如果到头或者尾,则跳回尾或者头形成循环。

对于鼠标的选中,则从上到下从左向右遍历这个二重链

表，判断鼠标选中的坐标值是否在二重链表中所有的控件的坐标范围内，如果是，选中该控件。

对应控件被选中的标识，即控件被选中后，执行标识函数：通过改变控件背景颜色或者给控件外部加上颜色框等方法显示，标识此控件已经被选中。

对于选中的控件，输入的内容在回车之前存在一个全局变量中，并且在每次输入的时候在控件位置刷新显示，在回车后存入控件的数据区。

2 开发平台底层技术实现

开发平台下层用到的 WindML 提供的 2D graphic 库中包括文字显示、绘图原语、颜色管理和事件服务库的部分 API，其中主要包括：

2.1 中西文混合显示技术

WindML 里默认字库中没有汉字字库，因此我们需要利用到 WindML 支持 Unicode 编码双字节字库，进行汉字显示的支持。

WindML 支持的字体文件形式为 BMF 形式，是一种位图字体，我们可以手工加入制作好的汉字字体，修改配置文件后编译 VxWorks 镜像，这就提供了可显示汉字的 VxWorks 操作系统内核。

在 WindML 中字符串显示接口函数包括 uglTextDraw 函数用于包括西文字符的单字节字符的输出，uglTextDrawW 函数用于包括中字字符的双字节字符的输出，但是没有直接进行混合显示的函数，通过观察汉字编码(gb2312)中任意一个字的编码值中每个字节都不小于 0xA1，而西文 ASCII 的编码均小于此值，因此，可以编写函数对输入的字符串编码进行逐字节的判断：如果小于 0xA1，则判断为单字节的西文字符，调用 uglTextDraw 进行显示，如果大于 0xA1，则下面两个字节的内容为 Unicode 中文编码，调用 uglTextDrawW 进行显示。同时，通过调用 uglTextSizeGet 或者 uglTextSizeGetW 进行下一个字符显示位置的计算。

中西文混合显示技术实现包括下面几步：

(1)首先利用工具将需要的汉字字库转换成 VxWorks 可以用的 bmf 字体形式，把字体文件拷贝到 WIND_BASE\target\src\ugl\font\bmf 目录中；

(2)修改 WindML 库的配置：首先添加字体索引在配置文件 uglBmfCfg.c，对加入字体进行注册，然后在 Torando 中 WindML 配置界面中将新加入的字体包括进去。编译带汉字字库的 WindML 库，进而编译带汉字字库的 VxWorks 镜像；

(3)编写字体初始化函数，分别调用 uglFontFindString、uglFontCreat 找到需要显示字体的字库，创建字体，调用 uglFontSet 选择此字体进行输出；

(4)编写中西文混合显示函数，内部调用了 uglTextDraw、uglTextDrawW、uglTextSizeGet、uglTextSizeGetW 函数，通过判断编码值大小来进行选择调用。

对于上层显示函数，只需要调用字体库初始化函数后，在需要显示汉字的地方调用中西文混合显示函数进行显示。

2.2 鼠标键盘技术实现

输入部分主要包括键盘和鼠标的输入，利用到了 WindML 对输入的事件服务进行处理，其中底层驱动监测键鼠的输入，如果检测到后将得到的流数据封装成 WindML 事件的标准数据格式，由事件句柄首先调用注册过的回调函数处理后，再传递给输入程序的事件队列中，如图 5 所示。

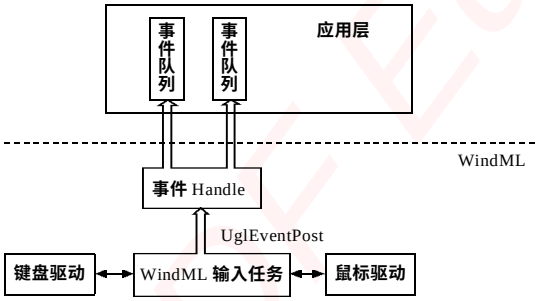

图 5 WindML 事件服务

作为框架程序中输入的实现，只需对得到的输入事件进行解析，按照需求，处理时先将输入事件队列中事件按照鼠标键盘的分类，然后分别对应用回调函数进行处理，将事件中的数据进行解析，得到鼠标左键按下后坐标值和键盘按下的键码，对应再进行后续处理。

对于我们用到的框架程序，具体实现步骤如下：

(1) 首先初始化输入事件队列；

(2) 启动输入任务，循环获得事件队列的队列，解析得到的 UGL_EVENT 结构的事件，通过 .header.type 得到事件是键盘按键按下或者是鼠标左键按下；

(3) 分别调用对应不同的回调处理函数，传入参数通过事件解析的到鼠标的坐标(x，y 值)和按键码(scancode 和 key)，回调函数相应再进行处理。

2.3 2D 绘图原语

WindML 的 2DGraphic 库中绘图的原语包括 uglEllipse() 画椭圆、uglLine()画线、uglPixelSet()画像素、uglPolygon()画多边形和 uglRectangle () 画矩形。因此我们控件的形状主要有椭圆(包括圆形)和矩形，分别调用 uglEllipse()和 uglRectangle() 进行显示。

在进行多组控件同时绘图的时候，前后用 uglBatchStart 和 uglBatchEnd 进行组绘图，保证屏幕刷新的效率。

3 框架具体实现流程

具体实现启动两个任务，一个用来处理显示单元的数据消息队列，一个用于处理输入事件队列。

程序按 MVC 的模式，采用消息和事件作为控制器的驱动，通过控制器进行消息解析处理，对应模型层的数据进行改动，根据数据改动调用对应 View 视图层函数进行显示。所有数据结构和对应操作由构件层提供。

3.1 显示处理任务

显示任务分成初始化和显示处理主函数两个部分，初始化分成两个部分，第一个是底层系统的初始化，完成 ugl 初始化、gc 的创建、颜色的分配、字体的载入和消息队列的生成；第二部分是显示数据的初始化，建立所需要显示的数据结构，根据具体需求创建控件、帧和区域，并且进行对应相关属性和关

联性的设定。

框架程序提供的函数有初始化显示和初始化显示数据函数，其中初始化显示数据运用到了工厂模式，完成对构件的对象实例化和相关关系建立的过程。

初始化后，系统已经建立显示的所有数据以及相应关联，并且建立对其他任务的显示数据的消息队列。

显示主处理函数为循环不断接收其他任务给显示任务传输的消息，外部任务以帧为单位的消息传输，消息内容包括帧号，该帧第一个控件显示数据指针。对消息进行解析，通过帧号得到需要刷新的控件数目和位置，通过指针所指的位置读出相应数据进行更新。如果此帧号的帧为所在区域的当前帧，即显示的帧，则刷新显示否则只刷新所有控件的数据区。

外部其他任务需要更新屏幕上某一个帧的数据，则调用框架类的传送显示信息的函数，函数实现将传入的帧号和所有控件的显示数据放在一个全局的环形缓冲中，再自动把环形缓冲当前指针和帧号打包成消息发送给显示的信息队列中。

对于利用框架层之上的应用，只需要每次在初始化显示数据函数中调用框架层函数去定义自身需求下需要的控件、帧和区域的属性，规定全局惟一帧号。同时外部任务更改显示时按规定调用框架中帧更新函数自动进行消息传递，则可以实现自动的实时的显示和更新。

显示任务基本实现流程如图 6 所示，其中灰色部分为上层调用部分。

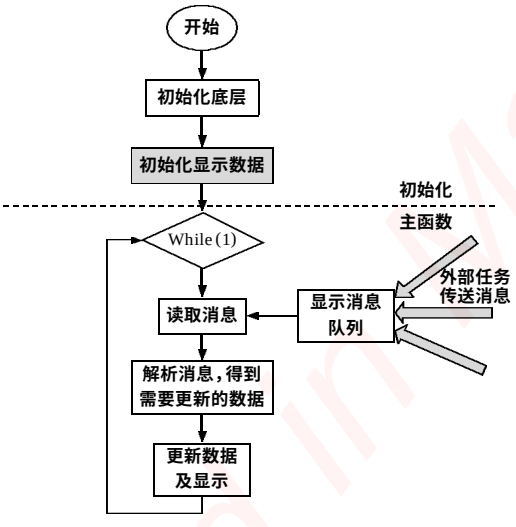

图 6　显示任务流程

3.2　输入处理任务

输入处理任务也包括初始化和输入处理主函数两个部分，初始化主要包括键盘鼠标的底层初始化，二重双向链表的建立和特殊按键的回调函数的设定。建立二重双向链表需要调用函数将控件的关系建立起来，因此必须确定在控件已经建立的基础上。所以输入的初始化必须在显示初始化之后。

输入主处理函数为循环接收键盘和鼠标事件，针对不同的事件进行不同的处理，对于鼠标事件，处理左键事件，对左键按下后通过坐标值进行选定，遍历链表进行定位，键盘中所有数字键回车键和退格键针对当前选中控件进行操作，特殊按键的功能，通过设定上层用户自己的回调函数，规定按下某个按键时执行回调函数的内容。因此可以实现对特定按键的有可设定的特殊用途。

对于上层用户，需要初始化输入数据建立二重链表以及编写针对特殊按键的回调函数，并与特殊按键绑定。

输入处理任务的基本流程如图 7 所示。

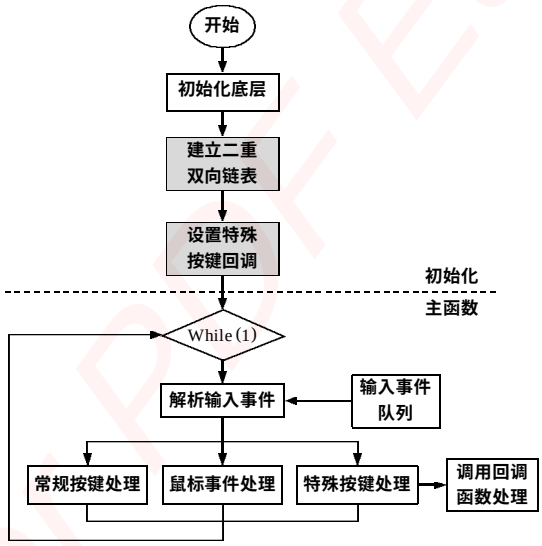

图 7　输入任务流程

4　结束语

VxWorks 由于其可靠性实时性等性质在实时控制领域得到了很大的发挥，但是由于 VxWorks 本身所带 WindML 库比较底层，因此开发实时控制设备的界面需要从底层开始向上建模实现，花费较长时间，并且概念不统一，代码不可重用。

本文提出了针对实时控制设备的交互界面框架的概念，图形框架针对实时控制设备抽象出多个概念，在 WindML 库的基础上开发出可重用的图形框架层，提供一套解决方法和接口函数。上层用户根据需求，利用图形框架层 API，建立自己的交互界面，而界面元素的管理，运行实现和界面的运行自动由下面的框架层自动执行，很大程度上缩短了开发周期，并且界面元素便于管理和扩展。

利用此框架层已开发某型舰船上发控设备和模拟设备界面，程序运行稳定良好。

在此框架基础上，建立控制系统图形界面 IDE，利用图形用户界面工具和代码生成器的结合，可以使得控制系统的图形界面开发更加直观和快速，同时，更加丰富界面元素的抽象，使得显示的形式更加多样化，这些都是下一步需要改进和进展的地方。

参考文献：

[1] WindML programmer' s guide 3.0[M]. USA: Wind River Systems Inc,2002.

[2] VxWoks programmer' s guide [M]. USA: Wind River Systems Inc,2002.

[3] 李方敏. VxWorks 高级程序设计 [M]. 北京: 清华大学出版社, 2004.

(下转第 1881 页)

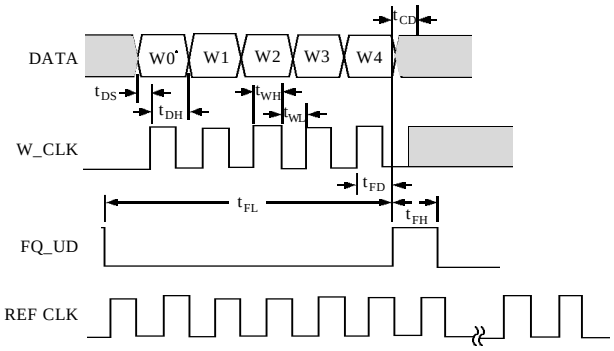

图 3　AD9850 并行控制字时序

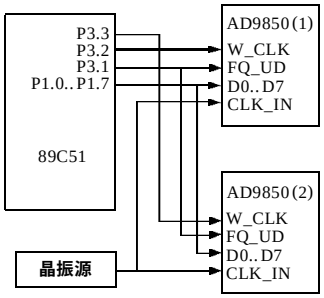

图 4　硬件原理

$M_2 = 2^{32}*0.21/40 = 22548578 = 0x1581062$

40位控制字装入程序如下(以产生210KHz的正弦信号为例)：

```
MOV   A, #00H
MOV   P1, A
SETB   P3.2
CLR    P3.2
MOV   A, #01H
MOV   P1, A
SETB   P3.2
CLR    P3.2
MOV   A, #58H
MOV   P1, A
SETB   P3.2
CLR    P3.2
MOV   A, #10H
MOV   P1, A
SETB   P3.2
CLR    P3.2
MOV   A, #62H
MOV   P1, A
SETB   P3.2
CLR    P3.2
SETB   P3.1
CLR    P3.1
```

4　实验结果

如图 5 所示，产生的频率分别为 210KHz 和 200KHz 的正弦信号。其中，黄线是频率为 210KHz 的波形，蓝线是频率为 200KHz 的波形。

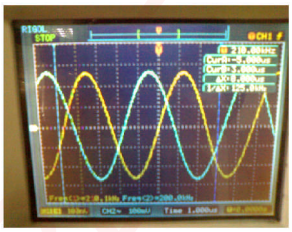

图 5　AD9850 产生的主振和本振信号

5　结束语

实验证明，用 AD9850 作为正弦信号发生器，工作稳定、结构简单，可产生零点几赫兹到几十兆赫兹的正弦信号。DDS 技术是一种重要的频率合成技术，在相位式激光测距仪中的可以产生频率和相位相对稳定的本振和主振信号，较好地解决了频率漂移和相位抖动的问题，这就保证了相位差的准确性，从而保证测量距离的正确。

参考文献:

[1] 陈慧兴.相位式光波测距仪[M].上海:科学技术出版社,1981.

[2] 孙彦景,张徵.基于μC/OS-Ⅱ的嵌入式脉冲激光测距系统[J].计算机工程与设计,2008,29(1):126-128.

[3] 王秀芳,王江,杨向东.相位激光测距技术研究概述[J].激光杂志,2006,27(2):4-5.

[4] DDS AD9850. Analog devices data sheet [Z]. Analog Devices Inc,2003.

[5] 刘川,孙利群,章恩耀.基于 FPGA 和 DDS 技术的激光测距仪[J].光学技术,2007,32(3):379-385.

[6] 肖汉波.一种基于 DDS 芯片 AD9850 的信号源[J].电讯技术,2003(2):47-50.

[7] 石雄,杨加功,彭世蕤.DDS 芯片 AD9850 的工作原理及其与单片机的接口[J].国外电子元器件,2001(5):53-56.

[8] 张竞,王向军.相位式激光测宽仪的设计和实现[J].电子测量技术,2007,30(9):125-128.

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

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

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

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

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