实时操作系统任务调度策略的研究与设计

Research and design of embedded real-time operating system scheduling strategy

桂林工学院 王新政 程小辉 周华茂
WANG XINZHENG CHENG XIAOHUI ZHOU HUAMAO

摘要 实时操作系统的调度策略是影响系统实时性和稳定性的一个重要因素 通过对抢占调度和分时调度的研究 设计了一个可分级抢占和分时调度的实时操作系统。用抢占调度方式来保证系统的实时性 用分时调度来减少进程饥饿现象的发生 提高系统的稳定性 更易于使用。

关键字 实时操作系统 时间片轮转 优先级检索 嵌入式操作系统
中图分类号 :TP316.2 **文献标识码** :A

Abstract:real-time operating system s scheduling strategy is an important factor that can affects system s performance. The system designed in this paper is a multi-level preemptive and round-robin scheduling operating system.It uses preemption to ensure the system s real-time performance and uses round-robin method to reduce procedure starvation makes the system more stable and easy to use.

Key words:real-time operating system,round robing,search priority,embedded operating system.

1 引言

嵌入式系统是一种面向应用的系统 与传统计算机系统相比 它更注重系统的体积、功耗、实时性、可靠性等问题 而传统的采用前/后台方式的系统实时性不高 其响应时间在任务级,最坏的情况下取决于整个循环的执行时间 如果循环修改了,则其响应时间也会随之变化。

如今嵌入式系统做得越来越复杂 要控制的外设和运行的应用程序也越来越多 甚至还需要与各类网络互联 传统设计模式变得越来越不适应 用操作系统来进行资源的调度和管理就显得十分必要 但嵌入式系统与广泛应用的 PC 机比起来具有的硬件资源相对要少得多 因此不可能也没必要把操作系统做得很大 基于此 专家们提出了微内核结构技术 即只把那些最本质、最基本的功能留在内核中 而其它的操作系统功能以单独的服务器进程形式存在。操作系统的性能很大程度上取决于内核的任务调度机制 实时操作系统内核可分为 非抢占式内核和抢占式内核两种 这两种内核都由中断服务程序来处理异步事件。在非抢占式内核中 中断服务程序可以使一个高优先级的任务就绪 但中断完成后仍返回到被中断的任务 直到其主动放弃 CPU 的使用权方可调度高优先级的就绪任务 这种系统的性能比前/后台方式的要好 但实时性还是不高 因为高优先级的任务就绪了也不能马上得到执行 见图 。1在可抢占型内核中 中断服务程序退出时会有两种情况 如果中断服务程序使高优先级任务就绪 则内核挂起被中断了的低优先级任务 使高优先级的任务执行 如果中断服务程序没有使高优级任务就绪 则中断后仍返回被中断了的任务 见图 。2任务在运行过程中也可以动态的创建任务 如果优先级比当前任务的要高也要抢占当前任务。

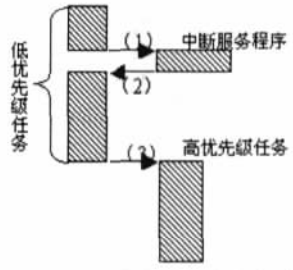

图 1 非抢占式内核任务调度

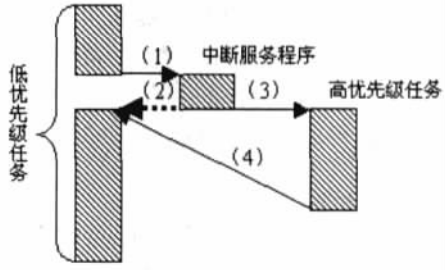

图 2 可抢占内核任务调度

2 任务队列的快速检索

在抢占式系统中如何快速的找到优先级最高的就绪任务是系统所要解决的一个重要问题 当就绪任务里有比当前任务优先级高的任务时 当前任务就要被高优先级的就绪任务抢占 很多主流的操作系统 如(Linux)是通过遍历就绪任务队列来检索高优先级的任务的 其时间复杂度为 O(N)当 增大时遍历的时间也会增大 影响实时性。查表法是美国嵌入式系统专家 Jean J.Labrosse在其设计的 uc/os 操作系统中使用的一种检索最高优先级就绪任务的方法。用查表法来寻找优先级最高的任务 简单、快速并且与当前系统的任务数无关 具有时间可确定性。在大多数的基于优先级的抢占式多任务实时操作系统中 采用的都是固定优先级方式 每个任务在创建的时候由系统的设计者赋予一个明确的与其它任务不同的优先级 用任务

的优先级即可唯一的标志任务，任务和优先级之间是一一对应的关系，而在本设计中的一个优先级对应着一个任务队列，不同的任务可以具有相同的优先级，但必需具有唯一的任务号。

查表法的关键是要建立查找表，查找表可以看作一个预先设计好的常量数组，SearchTable[]，数组有256个元素，从SearchTable[0]到SearchTable[255]各元素的值通过计算得到，这样每次可对一个8位数进行查表，确定最高优先级任务所在位置。SearchTable表的设计方法为首先把数组下标(0~255)用8位二进制来表示，最低位为第0位，最高位为第7位，再看为1的最低位在第几位，然后把其值填入到数组下标所指定的元素中。例如，SearchTable[86]，其下标为86，转换成二进制为01010110b，为1的最低位在第1位，所以把1填到下标为86的元素中。系统最多可支持64个优先级，把这64个优先级分为8组，每组8个，通过两次查表即可当前就绪表中的最高优先级。用ReadyGroup中的位来指示任务的优先级组，ReadyGroup的定义为INT8U ReadyGroup(INT8U为无符号的8位整数。)，数组ReadyPrioTbl[]来指示优先级，ReadyPrioTbl[]的定义为INT8U ReadyPrioTbl[8]，ReadyPrioTbl里的元素只要不为0，则ReadyGroup对应位置1。ReadyPrioTbl和ReadyGroup的关系可用图1来表示：

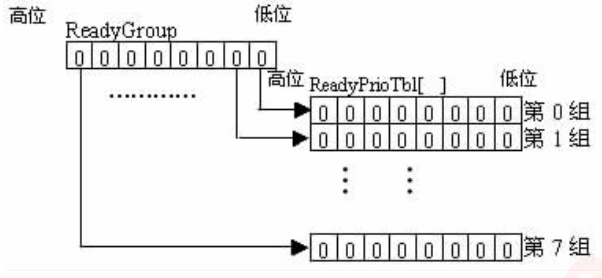

图1 优先级表位图

调度任务时，先用ReadyGroup查SearchTable表，找到任务优先级所在的组，假设查到的值为n，再用ReadyPrioTbl[n]值查表找到任务所在组里的位置，假查到的值为m，即可得到任务的优先级prio=n*8+m;

3 任务调度的实现

3.1 时间片轮转的意义

在实时操作系统中对任务按其重要程度和响应要求的不同赋予不同的的优先级，能够使那些对时间要求苛刻的任务优先的得到响应，提高了系统的实时性。然而，在实际的应用中，并不是所有任务重要程度都是不同的，往往会有多个任务具有同等的重要性，它们应该被赋予相同的优先级，在系统调度时得到同等的对待，如果对两个重要程度相同的任务强行的赋予不同的优先级，很容易出现一个任务长时间的占用处理机，而另一个同等重要的任务却得不到处理的情况，造成进程饥饿，甚至饿死，这同样会影响系统的实时性，降低系统性能，甚至会引起灾难性的后果。因此要使系统的多个任务能共一个优先级，对不同优先级的任务采用抢占式调度，提高实时性，相同优先级的任务采用时间片轮转方式分时调度减少进程饥饿发生，增强系统性能，更易于应用程序的设计。

3.2 优先级队列头结点设计

用一个优先级能对应多个任务的方法不但可以提高系统的性能，使系统更符合实际的应用，而且还扩大了系统所能支持的任务数。在系统里同一个优先级对应着多个任务，因此优先级不能作为任务的标志，而是采用任务号Task#作为任务的唯一标志，用优先级来指示任务队列。每个任务的任务号在系统中必须是唯一的，Task#是一个8位的无符号整数，总共可以表示256个任务。任务优先级指针数组QueueTable[]的每个指针分别指向一个优先级队列的头结点QUEUE，任务就绪时把它挂到与QUEUE结构点优先级相同的任务队列上。任务优先级指针数组QueueTable[]，优先级队列头结点QUEUE和任务控制块TASK_TCB的关系见图2。QUEUE的结构定义如下：

```
typedef struct queue{
    UINT8 QueuePriority; //8位无符号整数，保存队列的优先级
    UINT8 QueueTaskNum; //队列上的任务数，当此值为0时，
将任务就绪表OSRdyTbl[]中相应的位清0
    TASK_TCB *QueueHead; //队列头指针，在时间片轮转调
度时总是执行此指针所指任务，若任务的时间片用完而任务未
结束，则将任务挂于队列尾指针QueueEnd上，将QueueHead指
向下一个任务，并进行任务切换
    TASK_TCB *QueueEnd; //队列尾指针，与队列头指针一
起，便于任务的插入和删除，任务就绪时总是把它插在尾指针上
}QUEUE
```

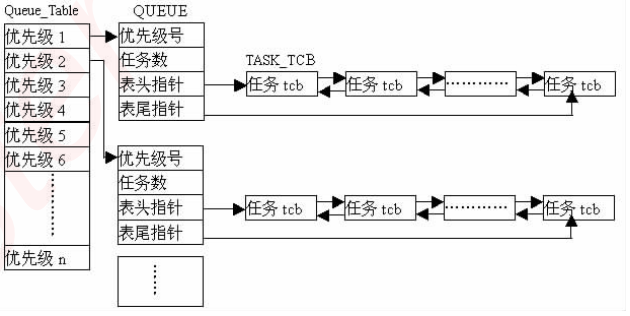

图2 任务优先级队列

3.3 任务的管理

为了便于直观的表示任务，在任务控制块中增加了一个存放任务名的数据变量，可存放10个字符，在建立任务时可给任务一个直观的名字，但系统不对任务名作唯一性检查，两个不同的任务其名字有可能相同，因此任务名不能用来标志任务，在系统里能唯一标志任务的只有任务的Task#号。要实现时间片轮转分时调度，需要在任务控制块TASK_TCB里增加Total-Time和RemainTime两项来保存系统的时间片值和剩余时间值。任务初始化时，TotalTime和RemainTime被赋予相同的值，都等于SliceTime，SliceTime是在os_cfg.h里定义的常数，用户可根据实际需要进行设定。系统时间片的量度为tick，它由定时器产生定时的溢出中断来实现，两个溢出中断之间的时间长度即为一个tick，中断产生后通过调用系统的时钟中断服务程序TickISR()来进行中断处理，TickISR()在保存现场后调用节拍函数TimeTick()对所有被延时或在超时等待的任务的时间值TaskDelay进行减1操作，如果系统延时或等待结束，则把任务置为就绪态并重新进行任务调度，如果当前任务不是优先级最高的任务，中断服务程序使得高优先级任务就绪，则把当前任务挂起，让高优先级的就绪任务运行，即进行抢占调度，否则对当前任务队列头结点的QueueTaskNum进行测试，如果QueueTaskNum大于1，则说明当前任务级所在队列上有多个任务，需要把当前任务的RemainTime值减1，如果减1后RemainTime

为 则调用函数 RollToNext(void)进行同优先级任务的时间片轮转调度。时间片轮转调度与抢占调度是不同的 在抢占调度时需要重新查表来确定任务优先级队列 而在时间片轮转调度时任务的优先级队列已知 只需把当前任务挂到队列尾 使结点头指针指向下一个任务 即可进行任务切换。 TASK_TCB的数据结构如下 :

```
typedef struct task_tcb{
TASK_ STK *TaskStkPt;     用于保存任务上下文的堆栈指针
struct task_tcb *TaskNext; 指向下一个任务控制块的指针
struct task_tcb *TaskPrev; 指向前一个任务控制块的指针
INT8U TaskId;             任务号 唯一的标志任务
Char TaskName[ 11];       用来保存任务名的字符数组
INT16U TotalTime ;        时间片长度值
INT16U RemainTime;        剩余的时间值
INT16U TaskDelay;         任务延时
INT8U TaskState;          任务状态
}TASK_TCB
```

节拍函数 TimeTick 清单如下 :

```
Void TimeTick ( )
{
TASK_TCB *TickPtr;
TickPtr=TaskTcbList;          获得已创建任务队列链表的头指针
while(*TickPtr! =null)        是否为队列尾
{
if(TickPtr- >TaskDelay! =0)
    如果减 后 task_delay为 并且不是被挂起状态则 使任务就绪
if((- - TickPtr- >TaskDelay=0)&(TaskState! =SUSPEND))
Ready_Task(TickPtr) ;         使任务就绪
else
TickPtr- >TaskDelay=1;        继续被挂起
}
if(TaskCurQueue- > QueueTaskNum>1) 当前任务所在队列有多个任务
CurrentTaskPtr- >RemainTime- - ;  剩余时间减 1
If(CurrentTaskPtr- >RemainTime= =0) ; 时间片用完
RollToNext( );                切换到队列的下一个任务
}
```

任务进入就绪态时除了把任务对应优先级位和其所属的组的相应位置 外 还要把任务挂到它所的优先级队列上 并把队列的任务数加 。在进行任务调度时 首先用查表的方法找到就绪表中的最高优先级组 再用优先级组的值 QueueTable[n] 为 0-找到任务队列 如果队列的任务数 .QueueTaskNum大于 则表示要进行时间片轮转调度 在时间中断服务时要对任务的剩余时间值 .RemainTim减 如果队列上的任务数不大于 则说明任务在抢占模式运行 不主动放弃 CPU除非有高优先级任务就绪对其进行抢占。任务完成或被挂起要退出就绪表时要判断队列任务数 QueueTaskNum是否为 0若为 则把 OS-RdyTbl里相应的优先级位清 如果此时优先级所属组的各位都为 0则还需把 OSRdyGrp里相应位清 0然后调用 OSSched()重新查表调度就绪任务里优先级最高的任务运行。

4总结

基于操作系统的嵌入式系统开发是嵌入式发展的一种趋势 也是嵌入式领域走向工业标准化的必经之路。在嵌入式系统中采用多任务的实时操作系统中能提高系统的实时性、稳定性、可靠性和加快产品的开发速度。用查表法来检索任务优先级队列能在确定的时间里快速的找到优先级最高的任务 查找任务的时间与当前的任务数无关。本文作者的创新点 结合了优先级抢占调度和分时调度的优点 既保证了系统的实时性又减少了进程饥饿现象的发生 更易于操作系统应用程序的开发 ,同时系统具有很大的灵活性 当每个优先级只给一个任务时 ,系统变成了完全的抢占式系统 如果把所有任务都分配在一个优先级上则又变成了分时系统。系统最大能支持 64个优先级 ,多达 256个任务 能满足绝大部分应用系统的要求。

参考文献

[1]陈明计,周立功 嵌入式实时操作系统 Small RTOS51原理及应用 [M]北京航空航天大学出版社, 2004

[2]沈胜庆 嵌入式操作系统的内核研究 [J]微计算机信息 .2006.2:72~74

[3]Xiao- hui Cheng,Ming- qiang Li,Xin- zheng Wang.Embedded real- time operating system micro kernel design.2005.ICMIT 2005[C]. Proc. of SPIE Vol. 6041: 60410F- 3~60410F- 4

[4]Jean J.Labrosse著,邵贝贝等译 嵌入式实时操作系统 uc/os II 第(版)[M]北京航空航天大学出版社, 2003

[5]子瀛,哲凤屏等 计算机操作系统 [M]西安电子科技大学出版社 . 1996

作者简介 王新政 (1981-男 汉族 广西桂林人 硕士研究生 ,主要研究方向 嵌入式系统、数据库。

Biography:Wang Xinzheng (1981 -),male,postgraduate student. Research: Embedded system, Database.

(541004广西桂林 桂林工学院 电子与计算机系 王新政 程小辉 周华茂

(Department of Electronics and Computer Science, Guilin University of Technology,Guilin 541004)Wang Xin - zheng Cheng Xiao- hui Zhou Hua- mao

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

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

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

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

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

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

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