VxWorks 下的多重定时器设计

VxWorks 是一种嵌入式实时操作系统(RTOS)，具有内核小、可裁剪、实时性强等特点。VxWorks 内核(Wind)提供了共享内存、信号量、消息队列、套接字通信和定时器等多种机制。为了实现基于 UDP 网络的可靠通信，本文利用 VxWorks 的多种任务间通信机制和看门狗定时器机制，设计了一种多重定时器模型，该模型可以确保数据包的可靠传递。

1 VxWOrks 的时钟及定时器机制

1．1 VxWorks 延时函数

VxWorks 既提供了延时功能，也提供了时限约束功能。VxWorks 系统有 2 种延时方式：一种是 Wind 内核提供的 taskDelay()函数；另一种是 POSIX 函数 nanosleep()。

taskDelay()函数以 tick 作为延时单位，默认情况下 1 个 tick 为 16.67 ms(1／60 s)，可以通过调用 sysClkRateSet()函数对 tick 进行重新设定。taskDelay()函数使调用该函数的任务在指定时间内主动放弃 CPU，用于任务调度或等待某一外部事件。nanosleep()函数指定一个以 s 和 ns 为单位的睡眠或延时时间。其实，两个延时函数的精度是相同的，都是以 tick 为时间基准。不同之处在于，taskDelay(0)有自身意义，用于相同优先级任务间的任务调度，而 nanosleep(0)是没有意义的。

1．2 VxWorks 定时器机制

VxWorks 提供一种看门狗定时器机制(watchdogtimer)，可以用来处理任务的时限约束。看门狗定时器作为系统时钟中断服务程序的一部分来维护，因此，看门狗定时器的回调函数以系统时钟中断级作为中断服务程序执行。看门狗定时器回调函数受到中断服务程序的限制，不能调用可能引起阻塞的函数，比如试图获取信号量，调用 malloc()和 free()等创建和释放内存函数或执行 I／O 操作。

POSIX 定时器也可以处理任务时限。此外，VxWorks 中一些函数具有时限控制的功能，semTake()、msgQSend()、msgQReceive()函数中都有设定时限控制的参数。超时参数 NO_WAIT 意味着立即返回，而 WAIT_FOREVER 意味着程序永不超时。

2 多重定时器实现要求

在 VxWorks 系统下，利用网络套接字建立基于 UDP 协议的客户端 / 服务器通信模式。由于 UDP 是无连接的协议，发送方并不清楚发出的数据包是否已经正确到达接收方，于是提出一种支持重传和定时等待确认的协议。

这个协议要求发送方发送的数据包与接收方回复的确认包具有对应的序列号，发送方和接收方都可以通过序列号来判断是不是想要得到的数据包。序列号是循环的，考虑到如果序列号太小会出现折返情况产生混淆，所以序列号至少大于 2。如果用 1 个字节来表示序列号，则可以设定序列号为 256。

发送方送出一个数据包后启动一个定时器。这时可能会有 4 种情况发生：

①发送方接收到正确序号的确认包，则发送下一序列号的数据包。

②发送方接收到已经接收过的重复确认包，则丢弃该确认包继续等待。如果在超时前收到了正确确认包，则发送下一序列号的数据包。

③定时器超时，没有收到想要的确认包，则重新发送数据包，启动下一定时器。

④设定的多重定时器超时后，没有收到想要确认包，则通知网络管理设备。

接收方在收到所需序列号的数据包后，回复一个确认包给发送方。如果接收方回复的确认包后没有正确到达发送方，则会引起发送方超时，重新发送原序列号数据包。接收方收到数据包后，需要检查数据包序列号。如果是重复序列号数据包，则丢弃，但是依旧回复确认包给发送方，以免已发送确认包在发送过程中丢失。这里基于支持重传和定时等待确认协议。具体要求是，在客户端通过 UDP 协议发送数据包后启动一个定时器，等待接收服务器端回复的 ACK(acknowl-edgement)确认包。如果成功接收，则继续发送下一序列号的数据包；如果超时后还没有收到需要的确认包，则重新传输原序列号的数据包。图 1 所示为数据包均按时、正确地接收的情况。

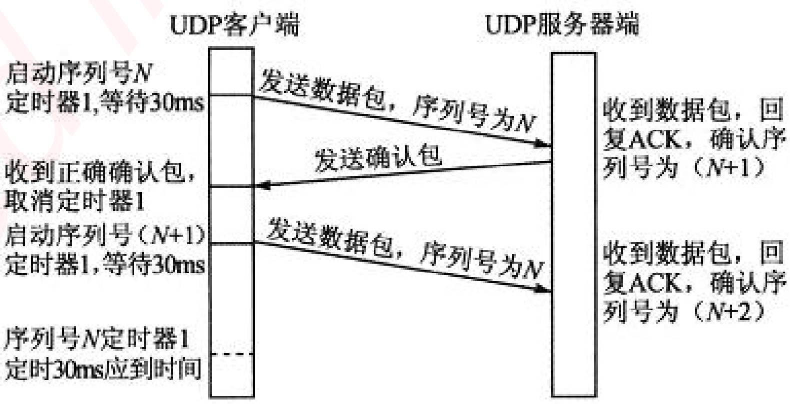

图 1 数据包正确接收情况

一般情况下，假定启动定时器 30 ms 内可以完成从发送数据包到接收 ACK 确认包的全过程，但是由于某些原因使得 30 ms 内无法收到确认包，则会重传原数据包，并启动一个稍长的 40 ms 定时器。如果 40 ms 还无法收到确认，则再次重传原数据包，并启动一个考虑到最差情况的 60 ms 定时器。如果依旧无法收到确认则不再发送，通知网络管理设备。

出现定时器超时情况有 3 种可能：发送方发送数据包过程中丢包；接收方发送确认包过程中丢包；从发送数据包到确认包到达发送方过程中，延时时间超过定时时间。造成超时有两方面原因：一是，双方终端在接收数据包时由于缓冲问题不能及时处理，使得终端出现延时接收数据包或丢包；二是，通信链路发生断链情况，导致双方无法进行通信。从图 2 中可以看到，如果链路没有断开，则包含 3 种情况的三重定时器超时情况。

3 多重定时器设计

3．1 设计方案

选用看门狗定时器机制来设计。看门狗定时器操作较为简单，只有 4 个函数，即 wdCreate()、wdDelete()、wdStart()、wdCancel()。看门狗定时器与调用任务异步执行，并不阻塞调用任务，所以看门狗定时器很适合多任务的非阻塞计时。

当看门狗定时器启动后，如果在规定的 30 ms 内收到了正确的确认包，就会将定时器取消掉，继续发送下面的数据包。如果 30 ms 规定时间内没有收到确认数据包 ACK，则需要重新发送数据包，并启动第 2 个 40 ms 的定时器。VxWorks 中单 CPU 的任务间常用通信机制是消息队列。当定时器到时后利用消息队列向发送任务发送消息，通知发送任务重新发送数据包，启动下一定时器。看门狗定时器的回调函数可以执行 msgQSend()这种向消息队列发送消息的函数，我们通过 msgQSend()函数向主任务发送时限已达消息。但是，将 msgQSend()的延时参数设为 wAIT_FOREVER 时，消息队列中一旦没有了可用缓冲，则进入等待状态。由于中断服务程序优先级高，而从消息队列中接收消息的优先级低，当有任务准备从消息队列中取消息时，要等待中断服务程序执行完毕，则消息队列始终处于已满状态，造成系统死锁。如果将 msgQSend()函数中的延时参数改为 NO_WAIT，则可避免一直等待向消息队列发消息的情况，一旦消息队列已满就将该消息丢弃。但这样一来，向接收端发送数据包任务接收不到定时器超时消息，不会重发原序列号数据包和启动下一定时器，所以使用参数 NO_WAIT 也不可行。

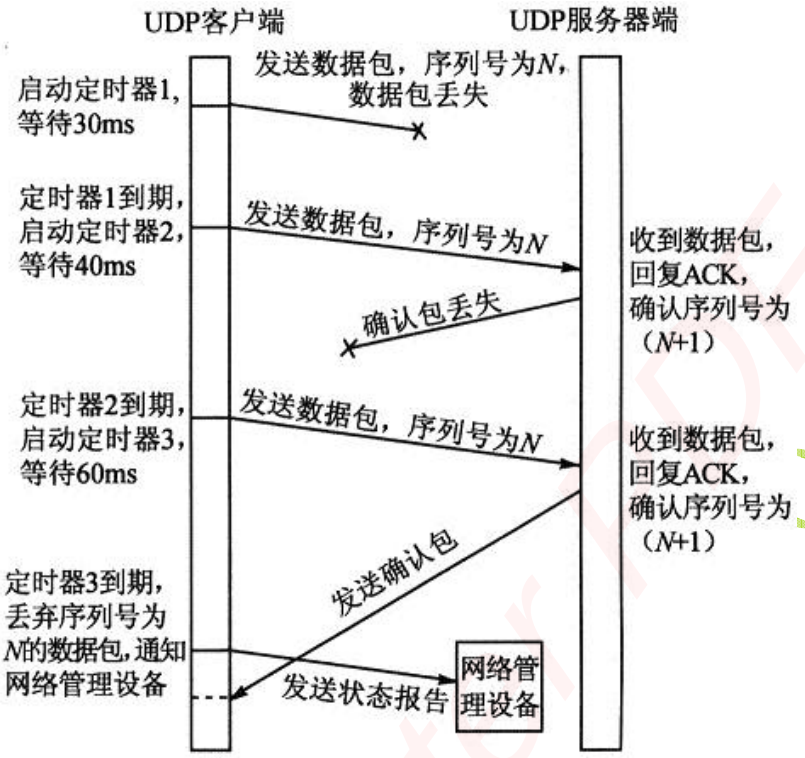

图 2　连续执行三重定时器情况

这里提出一种避免上述情况造成系统死锁的方法，即使用信号量机制来使 msgQSend()不在中断服务程序中执行。通过使用信号量的任务间同步机制来实现这个功能。释放信号量函数 semGive()不像 msgQSend()那样需要在消息队列中等待，一旦执行就可以马上释放信号量，从而避免了冲突。

由于任务中事件发生有一定间隔，不必选用计数器信号量，所以选用最常用的二进制信号量。首先建立 3 个先进先出的二进制信号量，设定可调用信号量为空。然后在看门狗定时器的回调函数中使用 semGive()函数来释放信号量，重建一个任务在任务起始使用 semTake()函数来获取信号量。当获得信号量后，通过 msgQSend(，，，WAIT_FOREVER，)函数向消息队列中发送超时消息，而且保证只要消息队列有可用缓冲，就一定可以将消息送出。本文给出一个多重定时器的任务框架，如图 3 所示。

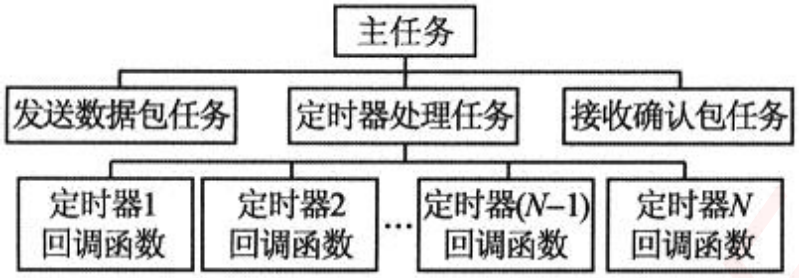

图 3　多重定时器任务框架

3．2 主要实现代码

一个三重定时器的主要实现代码如下：

```
STATUS timerdemo(void) {
  sysClkRateSet(100);              /* 设定 1 个 tick 为 10 ms */
  syncSem1=semBCreate(SEM_Q_FIFO,SEM_EMPTY);
                                   /* 创建 3 个二进制信号量 */
  syncSem2=semBCreate(SEM_Q_FIFO,SEM_EMPTY);
  syncSem3=semBCreate(SEM_Q_FIFO,SEM_EMPTY);
  wdId=wdCreate();                 /* 初始化看门狗定时器 */
  msgQId1=msgQCreate();            /* 创建消息队列 1 */
  msgQId2=msgQCreate();            /* 创建消息队列 2 */
  taskSpawn( (FUNCPTR)centerTask ); /* 启动 6 个任务 */
  taskSpawn( (FUNCPTR)serverTask );
  taskSpawn( (FUNCPTR)clientTask );
  taskSpawn( (FUNCPTR)semTake1 );
  taskSpawn( (FUNCPTR)semTake2 );
  taskSpawn( (FUNCPTR)semTake3 );
          /* 终止并释放信号量、看门狗定时器、消息队列 */
  ……
}
LOCAL STATUS centerTask(void) {
  msgQSend(msgQId1);/* 发送数据包命令发给消息队列 1 */
  wdStart(3,(FUNCPTR)sencondWd);
       /* 启动第 1 个定时器(30ms),回调函数为 sencondWd() */
```

```
    FOREVER {
      msgQReceive(msgQId2); /* 从消息队列 2 中接收消息 */
      if(RCV_ACK)           /* 如果收到正确确认数据包 */
      wdCancel(wdId);     /* 取消正在计时的看门狗定时器 */
      msgQSend(msgQId1); /* 发送数据包命令给消息队列 1 */
      if(TIMEOUT) {              /* 收到定时器超时消息 */
        if(FIR_TIMEOUT) {                /* 第 1 次超时 */
          msgQSend(msgQId1);
              /* 发送重复序列号数据包命令给消息队列 1 */
          wdStart(4,(FUNCPTR)thirdWd);
          /* 启动第 2 个定时器(40 ms),回调函数为 thirdWd() */
        }
        if(SEC_TIMEOUT) {                /* 第 2 次超时 */
          msgQSend(msgQId1);
              /* 发送重复序列号数据包命令给消息队列 1 */
          wdStart(6,(FUNCPTR)deadlineHandler);
    /* 启动第 3 个定时器(60 ms),回调函数为 deadlineHandler() */
        }
        if(THI_TIMEOUT)                  /* 第 3 次超时 */
          msgQSend(msgQId1);
        /* 通知消息队列 1 不用再重发,并通知网络管理设备 */
        }
      }
}
LOCAL STATUS serverTask(void) {
  FOREVER{
      msgQReceive(msgQId2);
      if(SND_DATA)
        sendto();
            /* 如果收到发送命令,则向接收方发送数据包 */
      else if(RESND_DATA)
        sendto();  /* 如果收到重复发送命令,则向接收方发
                      送重复序列号数据包 */
  }
}
LOCAL STATUS clientTask(void) {
  FOREVER{ recvfrom();
        if(RCV_ACK);           /* 判断是否是确认信号 */
        msgQSend(msgQId2);
                          /* 将确认包发送到消息队列 2 */
          }
}
LOCAL void SecondWd(void) {
```

```
                                        /* 获得信号量 syncSem1 */
        msgQSend(msgQId);
            /* 发送超时消息,说明是第 1 次超时(FIR_TIMEOUT) */
    }
}
LOCAL void ThirdWd(void) {
    semGive(syncSem2);          /* 释放同步信号量 syncSem2 */
}
LOCAL void SemTake2(void) {
    FOREVER{
        semTake(syncSem2,WAIT_FOREVER);
                                        /* 获得信号量 syncSem2 */
        msgQSend(msgQId);
         /* 发送超时消息,说明是第 2 次超时(SEC_TIMEOUT) */
    }
}
LOCAL void deadlineHandler(void) {
    logMsg("\nI have lost the send_packet!\n",0,0,0,0,0,0);
    semGive(syncSem3);          /* 释放同步信号量 syncSem3 */
}
LOCAL void SemTake3(void) {
    FOREVER{
        semTake(syncSem3,WAIT_FOREVER);
                                        /* 获得信号量 syncSem3 */
        msgQSend(msgQId);
         /* 发送超时消息,说明是第 3 次超时(THI_TIMEOUT) */
    }
}
```

以上程序中通过 sysClkRateSet(100)将最小延时单位 tick 修改成 10 ms，它是几个定时时间(30 ms、40 ms、60ms)的最大公约数。通过抓包软件 Ethereal 抓包，查看发送时间。以 30 ms 为例，抓包 100 次的平均定时时间在 25 ms 左右。出现这种情况的原因是，延时 N 个 tick 实际是延时(N-1)tick~N • tick。由于是等可能概率，则它的数学期望是(N+1 / 2)。对于 tick 为 10 ms，30 ms 即 N=3，数学期望为 25 ms。示意图如图 4 所示。

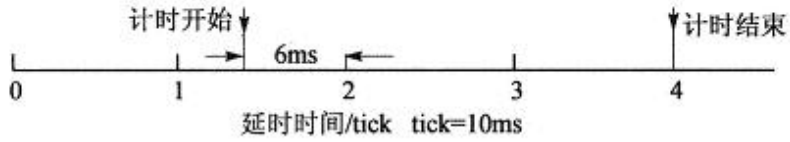

图 4 延时 30 ms 实际延时 26 ms

延时精度为 1 / N 秒，N 越大越精确。于是调用函数 synClkRateSet(500)，可以使定时的最大误差不超过 2 ms。但是如果时钟频率太高，会造成系统在时钟中断处理方面开销太大，影响系统的任务调度，最好通过实验选用较为合适的时钟频率。这里选用 sysClkRate-Set(200)。

结 语

本文针对 VxWorks 下 UDP 网络通信中的可靠传输问题，提出了一个支持重传和定时等待确认的协议，并利用 VxWorks 系统提供的信号量同步、消息队列和看门狗定时器等多种机制，综合设计了一种可扩展的三重定时器。针对遇到的具体问题，笔者还进行了一定的优化处理。这种多重定时器模型已在笔者所研究的项目中得到利用，验证了其可行性和相对稳定性。这种多重定时器模型并不完全适合所有环境，需要根据具体情况改进和优化。

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
55. USB30 的高速信息传输瓶颈研究
56. 基于 IPv6 的 UDP 通信的实现
57. 一种基于 IPv6 的流媒体传送方案研究与实现
58. 基于 IPv4-IPv6 双栈的 MODBUS-TCP 协议实现
59. RS485CAN 网关设计与实现
60. MVB 周期信息的实时调度
61. RS485 和 PROFINET 网关设计
62. 基于 IPv6 的 Socket 通信的实现
63. MVB 网络重复器的设计
64. 一种新型 MVB 通信板的探究
65. 具有 MVB 接口的输入输出设备的分析
66. 基于 STM32 的 GSM 模块综合应用
67. 基于 ARM7 的 MVB CAN 网关设计
68. 机车车辆的 MVB CAN 总线网关设计
69. 智能变电站冗余网络中 IEEE1588 协议的应用
70. CAN 总线的浅析 CANopen 协议
71. 基于 CANopen 协议实现多电机系统实时控制
72. 以太网时钟同步协议的研究
73. 基于 CANopen 的列车通信网络实现研究
74. 基于 SJA1000 的 CAN 总线智能控制系统设计
75. 基于 CANopen 的运动控制单元的设计
76. 基于 STM32F107VC 的 IEEE 1588 精密时钟同步分析与实现

77. 分布式控制系统精确时钟同步技术
78. 基于 IEEE 1588 的时钟同步技术在分布式系统中应用
79. 基于 SJA1000 的 CAN 总线通讯模块的实现
80. 嵌入式设备的精确时钟同步技术的研究与实现
81. 基于 SJA1000 的 CAN 网桥设计
82. 基于 CAN 总线分布式温室监控系统的设计与实现
83. 基于 DSP 的 CANopen 通讯协议的实现

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
33. 基于 VxWorks 实时控制系统中文交互界面开发平台
34. 基于 VxWorks 操作系统的 WindML 图形操控界面实现方法
35. 基于 GPU FPGA 芯片原型的 VxWorks 下驱动软件开发
36. VxWorks 下的多串口卡设计
37. VxWorks 内存管理机制的研究
38. T9 输入法在 Tilcon 下的实现
39. 基于 VxWorks 的 WindML 图形界面开发方法
40. 基于 Tilcon 的 IO 控制板可视化测试软件的设计和实现
41. 基于 VxWorks 的通信服务器实时多任务软件设计
42. 基于 VXWORKS 的 RS485MVB 网关的设计与实现
43. 实时操作系统 VxWorks 在微机保护中的应用
44. 基于 VxWorks 的多任务程序设计及通信管理
45. 基于 Tilcon 的 VxWorks 图形界面开发技术
46. 嵌入式图形系统 Tilcon 及应用研究
47. 基于 VxWorks 的数据采集与重演软件的图形界面的设计与实现
48. 基于嵌入式的 Tilcon 用户图形界面设计与开发
49. 基于 Tilcon 的交互式多页面的设计
50. 基于 Tilcon 的嵌入式系统人机界面开发技术
51. 基于 Tilcon 的指控系统多任务人机交互软件设计
52. 基于 Tilcon 航海标绘台界面设计
53. 基于 Tornado 和 Tilcon 的嵌入式 GIS 图形编辑软件的开发

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
40. 基于嵌入式 Linux 下的 USB30 驱动程序开发方法研究
41. Android 手机应用开发之音乐资源播放器
42. Linux 下以太网的 IPv6 隧道技术的实现
43. Research and design of mobile learning platform based on Android
44. 基于 linux 和 Qt 的串口通信调试器调的设计及应用
45. 在 Linux 平台上基于 QT 的动态图像采集系统的设计
46. 基于 Android 平台的医护查房系统的研究与设计
47. 基于 Android 平台的软件自动化监控工具的设计开发
48. 基于 Android 的视频软硬解码及渲染的对比研究与实现
49. 基于 Android 移动设备的加速度传感器技术研究
50. 基于 Android 系统振动测试仪研究
51. 基于缓存竞争优化的 Linux 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
21. DCOM 协议在网络冗余环境下的应用
22. Windows XP Embedded 在变电站通信管理机中的应用
23. XPE 在多功能显控台上的开发与应用
24. 基于 Windows XP Embedded 的 LKJ2000 仿真系统设计与实现
25. 虚拟仪器的 Windows XP Embedded 操作系统开发
26. 基于 EVC 的嵌入式导航电子地图设计
27. 基于 XPEmbedded 的警务区 SMS 指挥平台的设计与实现
28. 基于 XPE 的数字残币兑换工具开发

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
20. 基于 PowerPC 的模拟量输入接口扩展
21. 基于 PowerPC 的车载通信系统设计
22. 基于 PowerPC 的嵌入式系统中通用 IO 口的扩展方法
23. 基于 PowerPC440GP 型微控制器的嵌入式系统设计与研究
24. 基于双 PowerPC 7447A 处理器的嵌入式系统硬件设计
25. 基于 PowerPC603e 通用处理模块的设计与实现
26. 嵌入式微机 MPC555 驻留片内监控器的开发与实现
27. 基于 PowerPC 和 DSP 的电能质量在线监测装置的研制
28. 基于 PowerPC 架构多核处理器嵌入式系统硬件设计
29. 基于 PowerPC 的多屏系统设计
30. 基于 PowerPC 的嵌入式 SMP 系统设计

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
29. 基于 S3C6410 的 Uboot 分析与移植
30. 基于 ARM 嵌入式系统的高保真无损音乐播放器设计
31. UBoot 在 Mini6410 上的移植
32. 基于 ARM11 的嵌入式 Linux NAND FLASH 模拟 U 盘挂载分析与实现
33. 基于 ARM11 的电源完整性分析
34. 基于 ARM S3C6410 的 uboot 分析与移植
35. 基于 S5PC100 移动视频监控终端的设计与实现

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
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
25. 基于龙芯 2F 架构的 PMON 分析与优化
26. CPU 与 GPU 之间接口电路的设计与实现
27. 基于龙芯 1A 平台的 PMON 源码编译和启动分析
28. 基于 PC104 工控机的嵌入式直流监控装置的设计
29. GPGPU 技术研究与发展
30. GPU 实现的高速 FIR 数字滤波算法
31. 一种基于 CPUGPU 异构计算的混合编程模型
32. 面向 OpenCL 模型的 GPU 性能优化
33. 基于 GPU 的 FDTD 算法
34. 基于 GPU 的瑕疵检测
35. 基于 GPU 通用计算的分析与研究
36. 面向 OpenCL 架构的 GPGPU 量化性能模型
37. 基于 OpenCL 的图像积分图算法优化研究
38. 基于 OpenCL 的均值平移算法在多个众核平台的性能优化研究
39. 基于 OpenCL 的异构系统并行编程
40. 嵌入式系统中热备份双机切换技术研究

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

6. 基于 Socket 的网络编程技术及其实现
7. 数据结构考题 - 第 1 章 绪论
8. 数据结构考题 - 第 2 章 线性表
9. 数据结构考题 - 第 2 章 线性表 - 答案
10. 基于小波变换与偏微分方程的图像分解及边缘检测
11. 基于图像能量的布匹瑕疵检测方法
12. 基于 OpenCL 的拉普拉斯图像增强算法优化研究
13. 异构平台上基于 OpenCL 的 FFT 实现与优化
14. 数据结构考题 - 第 4 章 串
15. 数据结构考题 - 第 4 章 串答案
16.

FPGA / CPLD:

1. 一种基于并行处理器的快速车道线检测系统及 FPGA 实现
2. 基于 FPGA 和 DSP 的 DBF 实现
3. 高速浮点运算单元的 FPGA 实现
4. DLMS 算法的脉动阵结构设计及 FPGA 实现
5. 一种基于 FPGA 的 3DES 加密算法实现
6. 可编程 FIR 滤波器的 FPGA 实现
7. 基于 FPGA 的 AES 加密算法的高速实现
8. 基于 FPGA 的精确时钟同步方法
9. 应用分布式算法在 FPGA 平台实现 FIR 低通滤波器
10. 流水线技术在用 FPGA 实现高速 DSP 运算中的应用
11. 基于 FPGA 的 CAN 总线通信节点设计
12. 基于 FPGA 的高速时钟数据恢复电路的实现
13. 基于 FPGA 的高阶高速 FIR 滤波器设计与实现
14.