IEEE1588 协议在网络测控系统中的应用*

杨传顺，彦廷江

(江苏自动化研究所，江苏 连云港 222006)

摘　要：在基于网络的测量与控制系统中，对分散节点间的时钟同步有很高的要求。IEEE 1588 标准的精确时钟协议为测控系统的时钟同步提供了一种简单可行的途径。分析了精确时钟协议实现的原理和算法，针对测控系统中对时钟同步精度的需求，研究了系统内选择最优主时钟的算法和实现，以及硬件辅助检测时间戳的方法和应用。最后以以太网为例，对设计的方法进行了精度测试，验证了在测控系统中应用精确时钟同步协议能够满足系统的应用需求。

关键词：测控系统；IEEE 1588 协议；精确时钟协议；最优主时钟算法；时间戳；硬件辅助

中图分类号：TP273　　**文献标识码**：A　　doi：10.3969/j.issn.1001-893x.2011.11.017

Application of IEEE 1588 Protocol in Networked Measurement and Control Systems

YANG Chuan-shun, YAN Ting-jiang

(Jiangsu Automation Research Institute, Lianyungang 222006, China)

Abstract: In a networked measurement and control system, high accuracy of the clock synchronization between the distributed nodes is required. IEEE 1588 standard of precision time protocol for measurement and control system provides a simple and feasible way to the clock synchronization. This paper analyses the principle and algorithm of precision time protocol, according to the requirement of measurement and control system for clock synchronization accuracy, studies the algorithm and implementation to choose the best master clock in the system, and the hardware-assisted method and application of detecting time stamp. Finally, taking the ethernet for example, it tests the accuracy of the clock synchronization designed and the resut verifies that the application of precision clock synchronization protocol in measurement and control systems can satisfy the system application requirements.

Key words: measurement and control system; IEEE 1588 protocol; precision time protocol; best master clock algorithm; time stamp; hardware-assisted

在基于网络的测量控制系统中，对时钟的应用主要有基于绝对时间的应用和基于频率的应用两种，绝对时间用于记录系统内事项的发生次序，频率用于控制器、执行器以及传感器的同步采样、回馈和执行。因此，分散节点间的时钟同步对于基于网络的测控系统来说非常重要，而 IEEE 1588 协议的推出正是为了解决测量仪器和工业控制系统中网络定时同步能力不足的问题。

1　精确时钟同步协议

IEEE 1588 协议[1]的全称是“网络测量和控制系统的精确时钟同步协议”，简称 PTP (Precision Time

Protocol)。国际电工协会也把此标准纳入到了 IEC 标准中,对应的标准是 IEC 61588[2]。它具有配置容易、收敛快速以及对网络带宽和资源消耗少等优点。它的主要原理是采用分层的主从模式通过一个同步信号周期性地对网络中所有节点的时钟进行校正;通过记录报文发送和接收的时间戳信息,计算出报文传输的时间延迟和主从时钟的时钟偏移,并相应地调整本地时间,使从时钟保持与主时钟一致的频率与相位,使系统中分散节点间的时钟达到高精度同步[3]。

PTP 定义了 4 种多点传送的报文[4]:同步报文(Sync)、跟随报文(Follow_up)、延迟请求报文(Delay_Req)和延迟应答报文(Delay_Resp)。这些报文的交互顺序将在第 2 节中说明,收到的信息响应与时钟当前的状态有关。

为了描述测控系统的时钟同步问题,我们以一个主时钟和一个从时钟的同步过程为例,如图 1 所示。

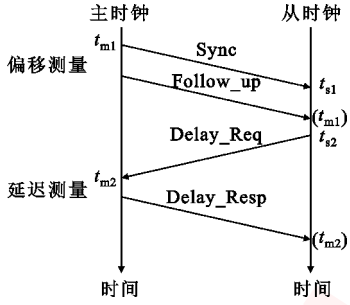

图 1 时钟同步过程示意图

Fig. 1 Schematic diagram of the clock synchronization

时钟同步过程如下所述。

(1) 主时钟按照定义的间隔时间周期性地向从时钟发送“同步报文(Sync)”,这个报文的时间戳是主时钟打上的预计发送时间,同时主时钟记下实际发送时间 t_{m1};此报文在接收端被从时钟记下接收时间 t_{s1}。

(2) 主时钟向从时钟发送“跟随报文(Follow_up)”,此报文包含先前的同步报文准确的发送时间 t_{m1}。

记网络延迟时间为 *Delay*,主从时钟偏移为 *Offset*,则:

$$t_{s1} = t_{m1} + Offset + Delay \tag{1}$$

(3) 从时钟向主时钟发送“延时请求报文(Delay_Request)”(延时请求报文的间隔是独立设置的),报文发送后是从时钟记下准确的发送时间 t_{s2},在主时钟端记录准确的接收时间 t_{m2}。

(4) 主时钟向从时钟返回一个“延时响应报文(Delay_Response)”,此报文包含先前的“延时请求报文”的准确接收时间 t_{m2},从时钟利用 t_{s2}和发送时间 t_{m2}。可以得到:

$$t_{m2} = t_{s2} - Offset + Delay \tag{2}$$

根据链路的对称性,由式(1)和(2)可计算出:

$$Delay = [(t_{s1} - t_{m1}) + (t_{m2} - t_{s2})]/2 \tag{3}$$

$$Offset = [(t_{s1} - t_{m1}) + (t_{m2} - t_{s2})]/2 \tag{4}$$

如果目前的从时钟时间为 t_{sOld},则可以计算出目前应该同步到的新的从时钟时间 t_{sNew}:

$$t_{sNew} = t_{sOld} - Offset - Delay \tag{5}$$

据此,在接下来的一个同步周期内,就可以实现从时钟和主时钟的同步。

以上是时钟同步过程的算法思想,在实际的应用中,同步报文和跟随报文的发送是按照定义的时间间隔周期性发送的;延迟请求报文和延迟应答报文发送间隔可以是不规则的,其测量间隔时间比偏移值测量间隔时间要大,这样做可以减轻网络尤其是终端设备的负荷[5]。

2 IEEE 1588 在网络测控中的应用

在测控系统中,测控装置的时钟一般来源于主站或后台;为了提高同步精度,一般采取系统内配置 GPS 装置,接收 GPS 的标准授时信号,其它节点和 GPS 的时钟同步,保证系统的时钟同步精确度要求;当 GPS 故障时,可选择利用与监控中心的时钟同步来校正,这正好符合 IEEE 1588 标准的要求。

在 IEEE 1588 标准中,系统内的时钟分为普通时钟和边界时钟两种。只有一个 PTP 通信埠的时钟是普通时钟,有两个及以上 PTP 埠的时钟是边界时钟。每个 PTP 埠的状态主要有主状态、从状态和被动状态 3 种。埠处于主状态的时钟称为主时钟,处于从状态的时钟称为从时钟,主时钟负责同步系统内所有的从时钟。如果埠处于被动状态,则意味着对应的时钟不参与时钟同步。理论上任何时钟都能实现主时钟和从时钟的功能,但一个系统内只能有一个主时钟,整个系统中的最优时钟为主时钟(Grandmaster Clock),具有最好的稳定性、精确性等。根据各节点的时钟精度、级别以及通用协调时间(UTC)的可追溯性等,由最优主时钟算法来自动选择

系统内的主时钟[6]。图 2 是一个网络测控系统中典型的主、从时钟关系示意图,最上面的时钟作为主时钟,其它设备的时钟通过分层结构都与主时钟同步。

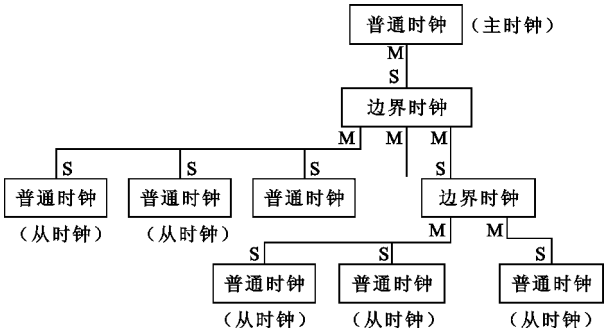

图 2　主时钟、从时钟关系示意图

Fig. 2 Schematic diagram of the relationship between master clock and slave clock

2.1　最优主时钟算法

最佳时钟(Best Master Clock,BMC)算法独立运行于系统内的每个时钟的端口上,在系统的所有时钟中(包括本地时钟)选出一个性能最好的,指定它作为系统中的最优主时钟,该算法避免系统内出现 2 个主时钟或没有一个主时钟的情况[7]。

BMC 算法由两部分组成:一是数据集比较算法,比较两组数据的优劣,一组可能是本地时钟的缺省数据集,一组可能是本地时钟的某个端口接收到的时钟数据集,应用此算法,从中选出质量较优的数据集;二是状态决策算法,根据数据集比较算法的结果计算本地时钟的每个埠当前应该所处的状态,并作相应的埠状态转换,为接下来系统的时钟同步作准备。

BMC 算法规定了数据比较的顺序和判据。时钟埠在设备上电时进行缺省初始化配置,为各数据集分配缓冲区等操作。然后,系统开始侦听网络上的时钟同步报文,如果埠收到相应的 PTP 报文,就调用 BMC 算法。

BMC 算法的典型软件实现步骤如下:

Step 1:对于具有 N 个埠的时钟 C_0的其中一个端口 r,通过数据集比较算法比较从这个端口接收到的其它时钟埠的有效同步报文的数据集,计算出最优的 E_{rbest};

Step 2:对 C_0的所有埠比较 E_{rbest},计算出 N 个埠最优的 E_{best};

Step 3:对 C_0每个埠,根据 E_{rbest}和 E_{best}以及缺省资料集 D_0,通过状态决策算法决策出端口应该所处的状态,并更新所有端口的数据集;

Step 4:把决策出的端口状态返馈到 PTP 引擎状态机,由状态机控制和实现埠状态的转化。

实现时钟端口的 BMC 算法的软件流程图如图 3 所示。对于系统的每个时钟,每个端口都运行 BMC 算法,这个运算是连续不断的,因此能适应时钟和埠的变化。

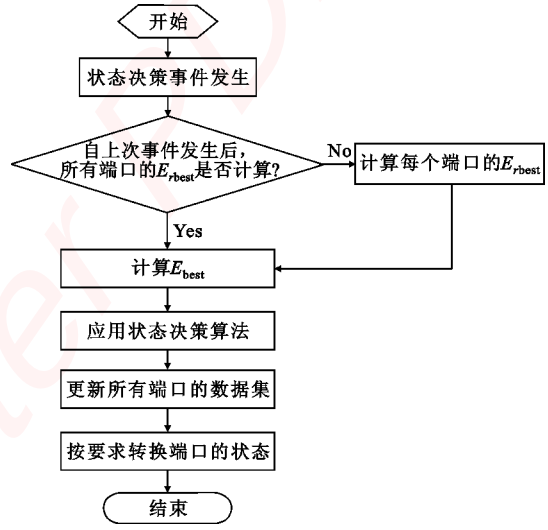

图 3　BMC 算法的软件流程图

Fig. 3 Flowchart of BMC software algorithm

2.2　时间戳记录

PTP 支持在物理介质层、驱动程序层或应用软件层检测和记录报文发送和接收的时间戳;越靠近物理介质层,同步精度越高;越靠近应用软件层,同步精度越低[8]。早期的网络时间协议(NTP)是只有软件控制的同步协议,在应用软件层打上时间戳,因此同步的精度不高;PTP 为了提高系统的时钟同步精度,既使用软件,亦同时使用硬件和软件配合,获得更精确的定时同步;PTP 支持硬件辅助在物理介质层检测和记录报文的时间,从而把报文从驱动程序到应用软件处理的不确定时间延迟排除在系统精度的分析之外。图 4 是在物理介质层硬件辅助记录时间戳的示意图,图中 t_{m1}、t_{m2}、t_{s1}和 t_{s2}分别是在物理介质层由硬件记录的同步报文的发送和接收时间,然后传送给应用软件,由应用软件打包记录到报文中,通过网络发送出去。

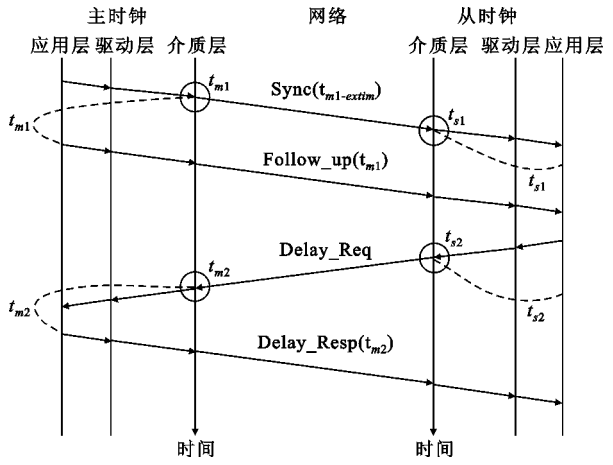

图 4　硬件辅助记录时间戳示意图

Fig. 4 Schematic diagram of the hardware-assisted record time stamp

目前已有多种芯片和微控制器支持 PTP 时间戳硬件实现,常见的有美国国家半导体的 DP83640 芯片、Silicon Labs 公司生产的 Si5315 芯片、飞思卡尔的 ColdFire MPU 等,也可以采用 FPGA 设计实现。当采用硬件辅助检测和记录 PTP 报文的时间戳时,即在网络的物理层(PHY)中放置这些器件,一旦线路上有 PTP 包,即被这些器件读取。采用 DP83640 芯片实现系统时钟同步的系统结构图如图 5 所示。

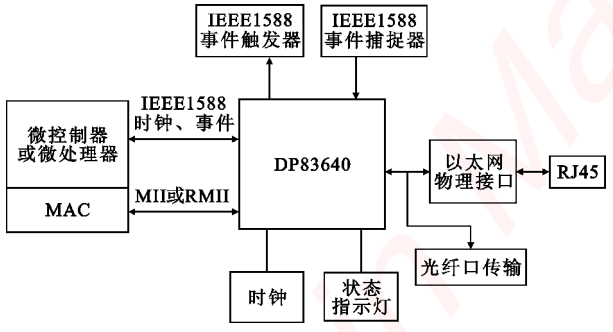

图 5　基于 DP83640 芯片实现时钟同步的系统结构图

Fig. 5 System structure of clock synchronization based DP83640 chip

3　精度测试

当在测控系统中采用 PTP 时,本地时钟相对于主时钟的同步时钟精度可由标准差(或称均方差)表示,PTP 的标准差基于 Allan 标准差,Allan 标准差 $\sigma_y(\tau)$ 的估计如下[9]:

$$\sigma_y(\tau) = \frac{1}{2(N-2)\tau^2}\sum_{k=1}^{N-2}(x_{k+2} - 2x_{k+1} + x_k)^2 \quad (6)$$

式中,x_k、x_{k+1}、x_{k+2}是对应时间 t_k、$t_k+\tau$、$t_k+2\tau$的样本测量值,τ是样本测量的时间间隔,N 是样本总数。Allan 标准差是对时基振荡器频率变化的二阶统计,PTP 标准差则是对测量时钟和本地参考时钟的偏差统计。

PTP 方差定义为

$$\sigma^2_{\mathrm{PTP}} = \tau^2 \times \frac{1}{3}\sigma_y^2 \quad (7)$$

由式(6)和(7)得 PTP 的标准差无偏估计的计算如下:

$$\sigma_{\mathrm{PTP}} = \frac{1}{6(N-2)}\sum_{k=1}^{N-2}(x_{k+2} - 2x_{k+1} + x_k)^2 \quad (8)$$

测试系统时钟同步精度的方法有多种,最常用的是通过示波器观察秒脉冲信号(PPS)的波形。为了使测试的精度能够满足实际的工程应用,以采用以太网组网为例,对不同情况下的网络负载分别进行了相应的测试。

设置主时钟和从时钟在某一频率点产生一个时钟输出,然后比较这两路时钟信号。采用秒脉冲测量是由于秒脉冲信号一般从数字输出中产生,虽然会对同步结果增添额外的误差,但是附加的误差仅会影响数字输入和输出,而不是同步时钟本身,因此不会包括在同步测量中。测试结果如表 1 所示。

表 1　秒脉冲的时钟同步精度测试结果

Table 1 Precision clock synchronization test results of second pulse

网络流量(利用率)	标准差/ ns
20 %	13.9
50 %	15.7
80 %	28.0

由表 1 可知,当网络负载在 80 %时,从时钟和主时钟的同步精度在28.0 ns的标准差内,完全能够满足网络测控系统的时钟需求。

4　结　论

随着测量和控制系统规模的不断增大,分散节点间的时钟同步变得愈加重要。IEEE 1588 精确时钟同步协议的出现,为测控系统的时钟同步提供了一种切实可行的途径。由于 PTP 不需要为时钟传递布置特别的网络,只需在原有网络上添加时钟同步报文,这些报文只占用少许网络资源,它们和控制数据包或其它信息包共享网络,具有较低的网络开销,在不需要太多资源的情况下,就能实现微秒级的高

精度时钟同步,完全能够满足系统对时钟的同步精度需求。

参考文献:

[1] IEEE 1588 - 2008, IEEE Standard for a precision Clock Synchronization Protocol for Networked Measurement and Control Systems[S].

[2] IEC 61588 - 2009, Precision Clock Synchronization Protocol for NetworkedMeasurement and Control Systems [S].

[3] 赵上林,胡敏强,窦晓波,等. 基于 IEEE1588 的数字化变电站时钟同步技术研究[J]. 电网技术,2008(21):97 - 102.
ZHAO Shang - lin, HU Min - qiang, DOU Xiao - bo, et al. Research of Time Synchronization in Digital Substation Based on IEEE 1588[J]. Power System Technology, 2008(21):97 - 102. (in Chinese)

[4] 于鹏飞,喻强,邓辉,等. IEEE 1588 精确时间同步协议的应用方案[J]. 电力系统自动化,2009(13):95 - 99.
YU Peng - fei, YU Qiang, DENG Hui, et al. The Research of Precision Time Protocol IEEE 1588[J]. Automation of Electric Power Systems, 2009(13):95 - 99. (in Chinese)

[5] 张振,许扬,陆于平,等. IEEE1588 协议在分布式系统保护信息传输中应用[J]. 电力自动化设备,2009(12):83 - 87,97.
ZHANG Zhen, XU Yang, LU Yu - ping, et al. Application of IEEE1588 protocol in information transmission of distributed protection system[J]. Electric Power Automation Equipment, 2009(12):83 - 87,97. (in Chinese)

[6] 李学桥,陈园,梁爽. 基于 IEEE1588 协议的精确时钟同步算法改进[J]. 计算机工程与科学,2011(2):42 - 45.
LI Xue - qiao, CHEN Yuan, LIANG Shuang. Improvement of the Precise Time Synchronization Algorithm Based on IEEE1588[J]. Computer Engineering & Science, 2011(2):42 - 45. (in Chinese)

[7] 刘明哲,徐皑冬,赵伟. 基于 IEEE1588 的时钟同步算法软件实现[J]. 仪器仪表学报,2006(S3):2009 - 2011.
LIU Ming - zhe, XU Ai - dong, ZHAO Wei. Implementation of clock synchronization algorithm by software method base on IEEE 1588 protocol[J]. Chinese Journal of Scientific Instrument, 2006(S3):2009 - 2011. (in Chinese)

[8] 马龙华,陈孝良,程晓斌,等. 分布式声学测量设备中精确时钟同步研究[J]. 计算机工程与设计,2011(1):351 - 353,365.
MA Long - hua, CHEN Xiao - liang, CHENG Xiao - bin, et al. Precise time synchronization for distributed acoustic measurement equipment[J]. Computer Engineering and Design, 2011(1):351 - 353,365. (in Chinese)

[9] 师伟,刘昕,万晓东. 基于 IEEE 1588 同步协议的高空台采集同步系统开发[J]. 计算机测量与控制,2010(2):476 - 478.
SHI Wei, LIU Xin, WAN Xiao - dong. Development of High Altitude Test Cell's Data Acquisition Synchronization System Based on IEEE 1588 Synchronization Protocol[J]. Computer Measurement & Control, 2010(2):476 - 478. (in Chinese)

作者简介:

杨传顺(1978—),男,山东菏泽人,2001 年于山东大学数学系获学士学位,2007 年于华北计算机系统工程研究所获硕士学位,现为工程师,主要研究方向为计算机控制和系统工程;

YANG Chuan - shun was born in Heze, Shandong Province, in 1978. He received the B. S. degree from Shandong University and the M. S. degree from the North China Computer System Engineering Research Institute in 2001 and 2007, respectively. He is now an engineer. His research interests include computer control and system engineering.

Email: yang97718@yahoo.com.cn

彦廷江(1963—),男,吉林人,1990 年于东南大学自动化系获硕士学位,现为高级工程师,主要研究方向为电子设备及显控台总体技术。

YAN Ting - jiang was born in Jilin Province, in 1963. He received the M. S. degree from Southeast University in 1990. He is now a senior engineer. His research concerns system technique of electronic equipment and console.

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

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