嵌入式设备的精确时钟同步技术的研究与实现

陈大峰[1]，白瑞林[1]，邹骏宇[2]
CHEN Dafeng[1], BAI Ruilin[1], ZOU Junyu[2]

1.江南大学 轻工过程先进控制教育部重点实验室，江苏 无锡 214122
2.无锡信捷电气有限公司，江苏 无锡 214072
1.Key Laboratory of Advanced Process Control for Light Industry（Ministry of Education）, Jiangnan University, Wuxi, Jiangsu 214122, China
2.Xinjie Electronic Co., Ltd., Wuxi, Jiangsu 214072, China

CHEN Dafeng, BAI Ruilin, ZOU Junyu. Research and implementation of precision clock synchronization in embedded devices. Computer Engineering and Applications, 2013, 49（4）:82-86.

Abstract: In order to solve the issue of the clock synchronization between industrial Ethernet embedded devices, the application scheme of IEEE1588 protocol in embedded devices is proposed. Based on the software and hardware platform of STM32F207IG processor and ucos-II OS, the transplant and modification of LwIP protocol stack is needed to make it compatible with the IEEE1588 protocol. The clock calibration method is configured. The application layer calibrates clock using the enhanced descriptors to access the time stamp captured. The correction algorithm is proposed to solve frequency drift. Pulse per second is used to test clock synchronization. Experiments show that the system achieves 200 ns clock synchronization precision, and meets the needs of most industrial Ethernet devices.

Key words: IEEE1588; clock synchronization; Precision Time Protocol（PTP）; time stamp; embedded device; frequency drift

摘　要: 为解决工业以太网中嵌入式设备之间时钟同步能力不足的问题，提出了IEEE1588协议在嵌入式设备中的应用方案。基于STM32F207IG处理器和ucos-II操作系统软硬件平台，首先移植并修改LwIP协议使其兼容IEEE 1588协议，然后配置系统时间校准模式，将捕获的时间戳以增强型描述符的形式交于应用层进行时钟校正，采用频率漂移校正算法解决从时钟频率漂移的问题，使用秒脉冲信号测试时钟同步。实验测试表明，时钟同步精度约200 ns，满足了大部分工业以太网的需求。

关键词: IEEE1588; 时钟同步; 精确时钟协议（PTP）; 时间戳; 嵌入式设备; 频率漂移
文献标志码: A　　**中图分类号:** TN915.05　　doi:10.3778/j.issn.1002-8331.1207-0056

1　引言

随着工业以太网的广泛应用，对地理位置分散的嵌入式设备协同完成工业任务的需求也越来越高，设备之间时钟同步精度不高或成本过高问题亟待解决。IEEE1588协议[1-3]，又称为精确时钟协议（Precision Time Protocol，PTP），提供了一种低成本高精度的网络时钟同步的解决方案，并具有支持多播技术的网络，占用网络带宽小，对系统资源要求低等优点，非常适合应用于工业嵌入式设备中。

针对IEEE1588协议在工业中的应用，文献[4]采用了纯软件方式实现了IEEE1588协议，设计了时钟频率调整系统进行主从时钟的同步调整，软件实现复杂，同步精度在微秒级。文献[5-6]采用FPGA设计了时间戳捕获、晶振频率补偿和时钟同步算法，但开发FPGA模块过程复杂。文献[7]采用了硬件辅助的方式，构建了以DP83640物理层芯片为主要组成部分的时钟同步功能模块，同步精度可达到纳秒级，但需要替换掉以太网物理层，成本较高。

综合分析后，本文提出了一种兼顾开发成本和时间精度的实现方案，采用在MAC层集成IEEE 1588时间戳功能的STM32F207IG芯片上实现IEEE 1588功能的方案。

2　系统机理分析

在工业以太网中应用IEEE1588协议首先需要组建主

从式拓扑结构网络，以消除环形通信路径，然后采用延迟测量技术测量主从时钟之间传输路径延迟值和时钟偏移值，并修改从时钟的时间以达到与主时钟时钟同步的目的，最后为了提高主从时钟同步精度，使用频率漂移校正算法修正从时钟的频率漂移。

2.1 组建主从式拓扑网络

工业以太网中各个嵌入式设备通过运行“最佳主时钟”（Best Master Clock，BMC）算法[8]，选择最合适的主时钟设备，同时也组建了主从式网络拓扑结构，如图1所示。由于所有设备均采用同样的数据集独立执行同一BMC算法，所以产生的结果也会相同，各设备之间不需要进行任何协商便可选出网络中的最优时钟。主从式拓扑网络结构可以有效地消除环形通信路径，为网络传输路径的对称性提供保证。

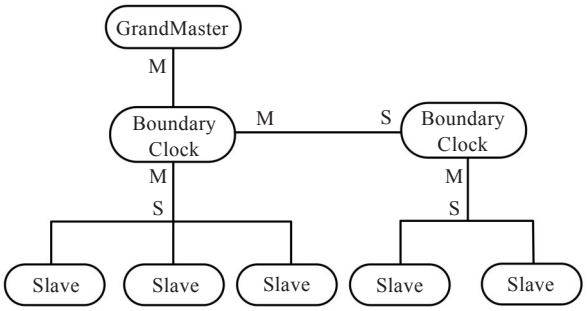

图1 网络拓扑结构图

2.2 主从时钟同步机理

主从时钟同步算法[9]使用Sync、Follow_up、Delay_Req和Delay_Resp报文来测量路径的通信延迟*Delay*，如图2所示，使用前提是网络的对称性即假设路径传输延迟*Delay*不变。Sync和Delay_Req报文需由主时钟和从时钟的时间戳硬件单元捕获发送和接收的准确时间。主从路径延迟*Delay*由Sync报文的发送和接收时间按照公式（1）计算得出。

$$Delay=(Ts1+Offset)-Tm1 \tag{1}$$

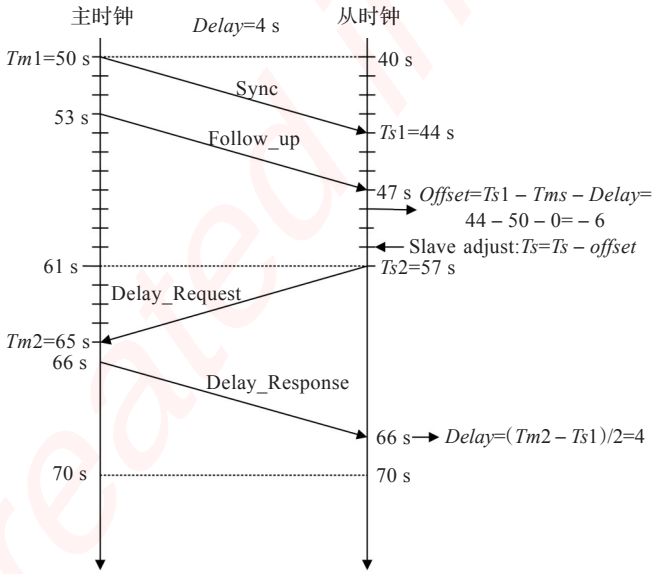

图2 主从时钟同步原理图

主从路径延迟*Delay*也可由Delay_Req报文的发送和接收时间按照公式（2）计算得出。

$$Delay=Tm2-(Ts2+Offset) \tag{2}$$

公式（1）和（2）相加可以得出路径延迟值如公式（3）所示。一旦获得通信路径延迟*Delay*，便可利用公式（1）或（2）计算出从时钟与主时钟的时间差，如公式（4）和（5）所示。

$$Delay=\frac{(Ts1-Tm1)+(Tm2-Ts2)}{2} \tag{3}$$

$$Offset=Tm1-Ts1+Delay \tag{4}$$

$$Offset=Tm2-Ts2-Delay \tag{5}$$

得到主从时钟偏差值*Offset*后，从时钟设备调整各自的时钟频率，使其时间调整到与主时钟一致。

2.3 时钟校准

主从节点运行同步算法后得到*Offset*后，从节点校准时钟一般有两种方式，粗调（Coarse）和精调（Fine）。

粗调，也就是直接补偿，初始化时，系统时间直接设定为初始值；校准时，系统时间直接加上或减去偏移值，从而达到主从时钟时间一致的目的。粗调方式能够在一个时钟周期内就完成校正。

精调，是指在一段时间里纠正从时钟相对主时钟的频率偏移，精调采用相对较长的时间可以维持时间的线性度。

粗调方式虽能很快地校正时钟，但会造成从节点的时间突变，会对系统造成很多的弊端。从时间线性度的角度出发，本文选择使用精调模式进行从时钟调整。

3 IEEE1588在STM32F207中实现方案

在STM32F207IG中实现IEEE1588协议的系统结构如图3所示。IEEE1588属于通信协议，协议的大部分是通过UDP层之上的软件实现的，PTP协议引擎以状态机的形式运行。PTP报文的发送和接收使用UDP的319和320端口，

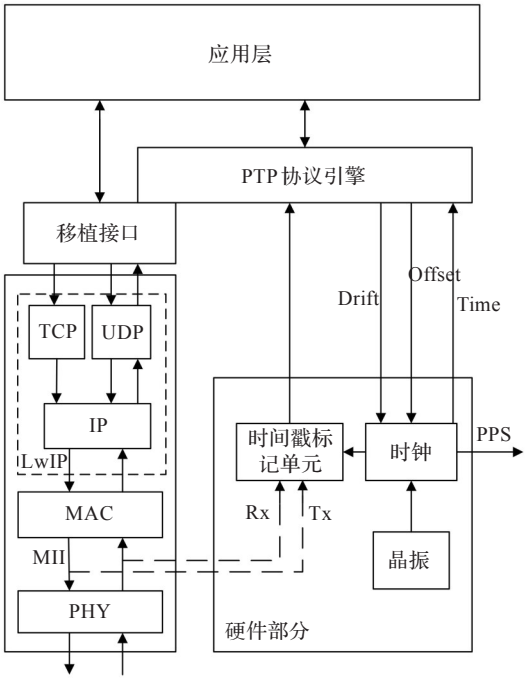

图3 系统结构图

当PTP报文经过MAC和PHY之间的MII时，时间戳单元（Time Stamp Unit，TSU）将记录该报文发送和接收的准确时间[10]并将该时间信息以增强型发送/接收描述符的形式返回给PTP协议引擎，PTP协议引擎运行主从时钟同步算法后将时间偏差值递交给时钟，并进行时钟频率调整，进而达到系统时间的校准，实现主从时钟同步。秒脉冲（Pulse Per Second，PPS）信号的输出是便于进行时钟同步的检测。移植接口保证了非PTP报文的传输不受PTP协议的影响。

3.1 IEEE1588硬件配置

STM32F207IG芯片的以太网模块集成了IEEE 1588功能，针对硬件部分设计了ETH_PTPStart()函数，主要功能是配置时钟校准模式为精调模式，程序流程如图4所示。首先屏蔽时间戳触发中断并使能时间戳功能，亚秒累加寄存器的数值设置可由公式(6)计算得出。

$$Increament = \frac{Tick \times 2^{31}}{10^9} \tag{6}$$

其中 *Increament* 表示亚秒累加器中设定的数值，*Tick* 表示系统时间的精确度。

主从设备的时钟通常是由脉冲计数的方式来实现。当 *Increament* 设置为常数43，则使得系统时间精度约为20 ns，即系统时间累加的步长是20 ns，因此系统时钟更新电路需要50 MHz的时钟频率以使系统时间精度达到20 ns。如果系统运行的基准时钟是66 MHz，则得出频率比为66/50=1.32，然后计算2^{32}/1.32=0xC1F07C1F，将0xC1F07C1F存放在时间加数寄存器。累加寄存器的值在每个时钟脉冲增加0xC1F07C1F，累加器的算术进位产生脉冲的时间就是20 ns，此脉冲可使亚秒累加器累加设定的常数值43，而亚秒累加器的算术进位可使系统时间增加1 s。显然可知，通过修改频率比，即修改时间加数寄存器中的值，就可以微调系统时间，修正主从时钟的偏差 *Offset* ，此种方法较好地校准了系统时间。

初始化时，时间戳更新寄存器的值即为要设定的系统时间初始值。当时间戳计数器初始化为时间戳更新寄存器的值后，它就开始计数工作了。如果关闭了时间戳功能，那么在重新打开此功能时，必须重复图4所示的流程。

3.2 频率漂移校正

从时钟相对于主时钟的频率漂移问题，可视为从节点系统运行的基准时钟频率发生了变化，因此修改频率比能够校正从节点的基准时钟频率。在软件设计上需要利用Sync消息计算出频率的漂移，并相应地更新加数寄存器的值。具体实现算法如下所述：

（1）主节点向从节点发送Sync报文，此时的时间记为 $MSTime(n)$ ，从节点收到Sync报文的时间记为 $STime(n)$ ，计算此时主节点的时间 $MTime(n)$，$Delay(n)$ 为主从路径延迟，可由公式(3)求出。

$$MTime(n) = MSTime(n) + Delay(n)$$

（2）当前Sync周期，主时钟时间差 $MDiff(n)$ 为：

$$MDiff(n) = MTime(n) - MTime(n-1)$$

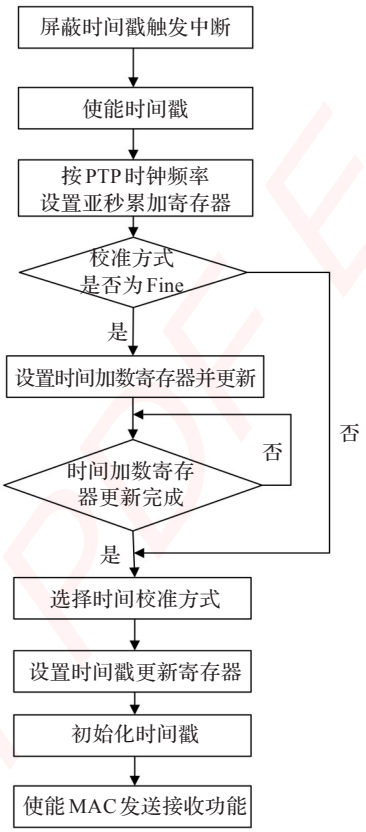

图4 PTP硬件配置流程

（3）当前Sync周期，从时钟时间差 $SDiff(n)$ 为：

$$SDiff(n) = STime(n) - STime(n-1)$$

（4）当前Sync周期，主时钟时间差和从时钟时间差的差别 $Diff(n)$ ：

$$Diff(n) = MDiff(n) - SDiff(n)$$

（5）从时钟的频率比系数 $Factor(n)$ 为：

$$Factor(n) = (MDiff(n) + Diff(n))/SDiff(n)$$

（6）时间加数寄存器的频率补偿值 $Value(n)$ 为：

$$Value(n) = Factor(n) \times Value(n-1)$$

该算法具有自校准功能，在不增加网络负载的情况下，可以很好解决从时钟频率漂移的问题，理论上可以在一个Sync周期内锁定频率，然而由于网络传播延时等，实际上需要几个Sync周期。

3.3 时间戳的处理

时间戳的捕获在MAC层和PHY层之间的介质无关接口（Media Independent Interface，MII）处，记录方式使用64位格式，高32位表示时间的second值，低32位表示时间的subsecond值。

当PTP报文封装成的以太网帧传输到MAC控制器，MAC并不通过处理帧本身的方式来识别PTP帧，而是根据帧发送描述符里的TDES0中的“发送时间戳使能位”来控制该帧是否需要捕获时间戳，该位在帧发送前进行置位。帧发送完成后记录下来的时间戳会放在帧的发送描述符中，使用发送中断返回给应用程序。这样应用程序就可以自动地把时间戳和PTP帧联系起来。64位的时间戳分别

写入TDES6和TDES7域中，如图5所示。接收时，MAC会在MII端记录下所有收到帧的时间戳，当帧接收完成后，MAC把时间戳写入到DMA接收描述符中。

	31							0
TDES0	占有位	控制位[30:26]	发送时间戳使能位	保留[24]	控制位[23:20]	保留[19:18]	发送时间戳状态位	状态位[16:0]
TDES1	保留[31:29]	发送缓冲区2字节数[28:16]			保留[15:13]	发送缓冲区1字节数[12:0]		
TDES2	发送缓冲区地址1[31:0]							
TDES3	发送缓冲区地址2[31:0]							
TDES4	保留							
TDES5	保留							
TDES6	发送时间戳低位[31:0]							
TDES7	发送时间戳高位[31:0]							

图5　增强型发送描述符

3.4　LwIP兼容性修改

LwIP是轻型的TCP/IP协议栈，主要关注的是怎么减少内存的使用和代码的大小，非常适用于资源有限嵌入式设备中。官方发布的LwIP并不允许从以太网接口发送和接收PTP报文，为使IEEE 1588协议能正确运行，须对LwIP进行兼容性修改。

首先是LwIP中结构体的修改。结构体pbuf中增加time_sec，time_nsec两个成员来记录时间戳信息；结构体FrameTypeDef中增加ETH_DMADESCTypeDef类型的指针以专门用于时间戳队列的处理。

其次是LwIP底层驱动函数的修改。

low_level_output()函数功能是底层处理pbuf中指向的数据包的发送；low_level_input()函数功能是底层处理接收数据包并将数据地址放入到分配好的pbuf中。为了能够准确地把DMA发送和接收描述符中的时间戳信息返回给应用程序，就需要对这两个函数的处理过程进行修改。udp_sendto_if()函数功能向IP层转发pbuf，存在pbuf复制的过程，须对结构体中新增的时间戳信息进行处理。因为在数据包发送接收时对时间戳的检测，底层驱动函数需要等待至时间戳检测完成并放入到DMA描述符中，处理速度将变慢。

最后增加PTP报文处理和时间调整的函数。

PTPRxPkt_ChainMode()和PTPTxPkt_ChainMode()是分别用于处理PTP报文包接收和发送DMA描述符的函数。PTPTime_AdjFreq()用来调整时钟频率，PTPTime_UpdateOffset()用于更新通信路径延迟值。

3.5　秒脉冲PPS使能

PPS秒脉冲输出是设备在每秒跳变处产生脉冲信号，可以用来检查网络中各个节点的同步情况。

PPS输出使能可以通过设置TIM2_OR寄存器的bit 11和bit 10为“01”，然后就可以检测PTP_PPS端口的输出信号了。默认频率是1 Hz，脉冲宽度是125 ms。

4　验证与测试分析

由于条件限制，本文只针对两台设备间时钟同步进行测试，在所有测试中，均将STM3220G_EVAL demon板1的IP设置为192.168.0.155，demon板2的IP设置为192.168.0.15，电脑端IP为192.168.0.2。时间同步间隔设置为1 s。

4.1　网络连通验证

在demon板上移植ucos-II和LwIP，并对LwIP做兼容性修改后，为了验证ucos-II和LwIP能够正常工作，使用网线将demon板与电脑相连，上电后，可在cmd.exe中使用ping命令测试网络是否连通。将一个网页测试程序作为ucos-II的一个任务运行，在电脑PC端的浏览器中输入demo板的IP地址正常打开程序中设定的网页，即可验证ucos-II和LwIP均能正常工作。

4.2　报文信息验证

首先使用网线连接STM3220G_EVAL demon板与电脑，将IEEE1588协议作为ucos-II的一个任务在demon板上运行，在电脑端使用Wireshark软件捕获网络中所有的信息包，当Wireshark能够捕获到demon板发送的PTP报文，即可验证demon板上运行的IEEE1588协议能够正常的运行。

4.3　同步精度测试分析

分析时间同步最常用的方法就是比较秒脉冲信号[11]，同步精度测试示意图如图6所示。

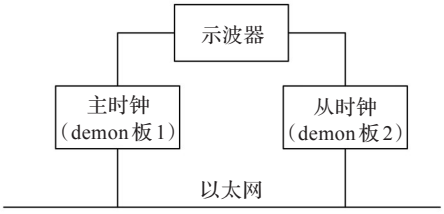

图6　同步测试示意图

使用网线将demon板1和板2连接起来，各自上电运行，使用示波器两通道分别检测PPS信号，结果在示波器中显示出来，图7显示为主从时钟的秒脉冲信号比对。通过

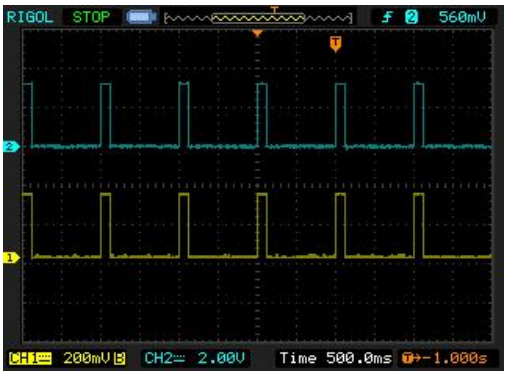

图7　主从时钟秒脉冲信号对比

比较PPS脉冲信号的上升沿可分析出其同步误差，由图8可比对出时间同步精度约为200 ns。

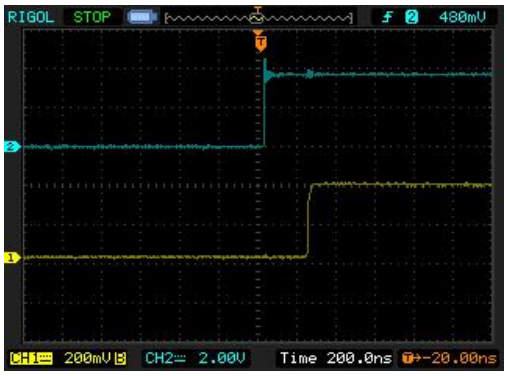

图8　时钟同步误差

5　结论

本文针对工业以太网中嵌入式设备时钟同步的需求，在STM32F207IG芯片上实现了IEEE1588功能，主要特点是：

（1）实现时钟校准模式的配置，使其能准确地捕获时间戳并调整从时钟频率实现与主时钟同步，并使用频率漂移校正算法很好地解决了从时钟频率漂移的问题。

（2）完成LwIP的修改以兼容IEEE1588协议，PTP报文只占用UDP的319和320端口，不影响非PTP报文的发送和接收。

同步精度测试实验表明，主从时钟时间同步精度约200 ns，很好地解决了以太网同步能力差的瓶颈，适合于以太网内各个节点间的时间同步。

参考文献：

[1] IEEE Std 1588-2002.IEEE standard for a precision clock synchronization protocol for networked measurement and control system[S].2002.

[2] IEEE Std 1588-2008.IEEE Standard for a precision clock synchronization protocol for networked measurement and control systems[S].2008.

[3] 李晓珍，苏建峰.基于IEEE1588高精度网络时钟同步的研究[J].通信技术，2011，44(3)：105-107.

[4] Correll K，Barendt N.Design considerations for software only implementations of the IEEE1588 precision time protocol[C]// Conference on IEEE-1588 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems.Winterthur：NIST and IEEE，2005.

[5] 遇彬，黄文君.基于FPGA的精确时钟同步方法研究[D].杭州：浙江大学，2007.

[6] 关世友，李智，胡聪.IEEE1588协议硬件时间戳标记电路设计[J].计算机测量与控制，2011，19(10)：2532-2535.

[7] 王康，胡永辉，何在民.基于DP83640硬件辅助的IEEE 1588研究及实现[J].时间频率学报，2011，34(1)：1-8.

[8] 戴宝峰，崔少辉，常健.IEEE1588最佳主时钟算法的分析与实现[J].仪表技术，2008(2)：29-31.

[9] 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，2011，33(2)：42-45.

[10] 魏丰，孙文杰.IEEE-1588协议时钟同步报文的精确时间标记方法研究[J].仪器仪表学报，2009，30(1)：162-169.

[11] Application Note 1728：IEEE 1588 Precision Time Protocol Time Synchronization Performance[EB/OL].National Semiconductor Corporation.(2007).http://www.ti.com/lit/an/snla098/snla098.pdf.

（上接42页）

[2] Feund R W，Nachtigal N M.QMR：a quasi-minimal residual method for non-hermitian linear systems[J].Numeerische Mathematik，1991，60：315-339.

[3] Feund R W，Golub G H，Nachtigal N M.Iterative solution of linear systems[M].Acta Numerica，1991：57-100.

[4] Georges P，Wsrzee G.High-performance PCG solves for FEM structural analysis[J].International Journal for Numerical Methods in Engineeing，1996，39：1313-1340.

[5] Arany I.Numerical experiences of solving elasticity systems by PCG methods[J].Computers and Mathematics with Applications，2001，42：1025-1033.

[6] 徐树方.矩阵计算的理论与方法[M].北京：北京大学出版社，1995：161-200.

[7] Gupta M M，Manohar R P.，Stephenson J W.A fourth order cost effective and stable finite difference scheme for the convection-diffusion equation[J].Numerical Properties and Methodologies in Heat Transfer，1983：201-209.

[8] Saad Y.Iterative methods for sparse linear systems[M].Boston：PWS Publishing，1996.

[9] Saad Y，Schultz H M.GMRES：a generalized minimal residual algorithm for solving nonsymmetric linear systems[J].SIAM J Sci Stat Comput，1986，7(3)：856-869.

[10] Van der Vorst H A，Vuik C.GMRESR：a family of nested GMRES methods[J].Numerical Linear Algebra with Application，1994，1：369-386.

[11] Meijerink J A，van der Vorst H A.An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix[J].Math Comp，1977，31(137)：148-162.

[12] Gusstafsson I.A class of first order factorization methods[J].BIT Numerical Mathematics，1978，18(2)：142-156.

[13] Saad Y.ILUT：a dual threshold incomplete ILU preconditioner[J].Numer Linear Algebra，1994，1(14)：387-402.

[14] Freund R W.A transpose-free quasi-minimal residual algorithm for non-hermitian linear systems[J].SIAM J Sci Comput，1993，14(2)：470-482.

[15] Saad Y.A flexible inner-outer preconditioned GMRES algorithm[J].SIAM J Sci Stat Comput，1993，14(2)：461-469.

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

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

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.