文章编号:1671-4598(2014)05-1619-04　　中图分类号:TP273　　文献标识码:A

嵌入式 Linux 设备的高精度 IEEE 1588 时钟同步实现

朱望纯，钟震林，覃斌毅
（桂林电子科技大学 电子工程与自动化学院，广西 桂林 541004）

摘要：IEEE 1588 时钟同步协议用于解决分布式网络测控系统中远距离仪器设备之间的同步问题；在分析 IEEE 1588 时钟同步实现原理的基础上，提出一种嵌入式 Linux 设备的高精度 IEEE 1588 时钟同步实现方案；采用专用 PHY 芯片 DP83640 在物理层为 PTP 报文加盖硬件时间戳，设计网络设备驱动与 PTP 硬件时钟控制驱动，并在用户层利用 Linux 系统标准 API 实现 IEEE 1588 协议软件；实验结果表明，两台设备直接相连时，时钟同步精度可稳定在±100 ns 以内。

关键词：IEEE 1588；时钟同步；PTP；硬件时间戳；DP83640

Implementation of High－precision IEEE 1588 Clock Synchronization Based on Embedded Linux Device

Zhu Wangchun, Zhong Zhenlin, Qin Binyi
(Department of Electronic Engineering and Automation, Guilin University of Electronic Technology, Guilin 541004, China)

Abstract: The IEEE 1588 Precision Time Protocol is used to achieve synchronization among the long－distance instruments in distributed network measurement and control system. On the basis of analysis of the IEEE 1588 clock synchronization principle, a method is proposed to implement high－precision IEEE 1588 clock synchronization based on the embedded Linux device. The dedicated PHY chip DP83640 is used to time stamp the PTP massages in physical layer, the network device driver and the PTP hardware clock control driver is designed, and the IEEE 1588 protocol software is implemented using the Linux system standard API in user space. Experimental results show that the clock synchronization accuracy can be stabilized within ± 100ns when two devices are directly connected.

Key words: IEEE 1588; clock synchronization; PTP; hardware timestamp; DP83640

0 引言

分布式网络测试以其高集成度、高效性、远程可控性、利于数据共享的优点正在测试领域中得到迅速应用。实现远距离仪器节点间的高精度同步是分布式网络测控系统中的难点。IEEE1588 精密时钟同步协议（Precision Time Protocol，PTP）是为克服以太网通信实时性不足而规定的一种对时机制，它能够通过 LAN 通信使得局域网内的仪器节点达到精确时钟同步[1]。

提高时钟同步精度是长期以来研究的重点。PTP 的实现通常有两种方式：（1）采用纯软件实现，即在软件层为 PTP 报文加盖时间戳，可以获得微秒级的时钟同步精度；（2）采用硬件辅助实现，在物理层对 PTP 报文加盖时间戳，可以消除 PTP 报文穿越网络协议栈以及软件执行所引起的不确定延时和抖动，时钟同步精度可达到纳秒级[2-4]。

目前，嵌入式 Linux 系统凭借其免费开源、可定制、可移植性好以及性能出色等特点在测试和控制领域获得了广泛的应用。而且从 2011 年 7 月发布的 Linux 3.0 内核开始集成了 PTP 时钟类驱动，为实现高精度时钟同步的 IEEE 1588 协议带来了极大的便利。

收稿日期:2013-12-28；　**修回日期**:2014-02-14。

基金项目:广西自然科学基金项目(2013GXNSFAA019332)；广西信息科学实验中心项目(2013ZD024)；广西教育厅科学技术研究项目(2013ZD024)；广西教育厅科研立项项目(201106LX163)。

作者简介:朱望纯(1976-)，男，湖南双峰人，副教授，主要从事虚拟仪器，自动测试系统方向的研究。

1 IEEE1588 时钟同步协议实现原理

IEEE1588 时钟同步的基本实现原理是：在 PTP 系统中，先通过最佳主时钟算法选择一个最稳定、最精确的时钟作为主时钟，其它所以时钟节点作为从时钟；从时钟周期性地与主时钟进行对时校正，实现与主时钟精确时间同步[5]。

PTP 系统中有两种时钟类型：普通时钟和边界时钟。普通时钟只有一个 PTP 端口，边界时钟有两个或更多的 PTP 端口。边界时钟用于代替 PTP 系统中插入的网络元件（交换机或路由），可消除网络元件引起的不确定延时和抖动，提高时钟同步精度。普通时钟模型如图 1 所示。

协议软件设计的核心是实现一个协议引擎，它负责了整个协议运行、计算与数据管理。PTP 协议引擎主要由 5 个部分组成：

（1）最佳主时钟算法：最佳主时钟算法 BMC（Best Master Clock algorithm）是 PTP 的核心算法之一。PTP 系统中每个时钟都独立运行最佳主时钟算法，通过计算确定时钟端口的状态，性能最好的时钟将被选择为主时钟，为其他时钟提供参考。同一时刻，同一 PTP 域内中只能有一个端口处于主时钟状态。最佳主时钟算法由两个部分组成[6-7]。

① 数据集比较算法：通过比较两组数据集的优劣，一组描述本地时钟的属性，一组描述外来时钟属性，最终选出质量较好的数据集。

② 状态决策算法：根据数据集比较算法的结果计算出本

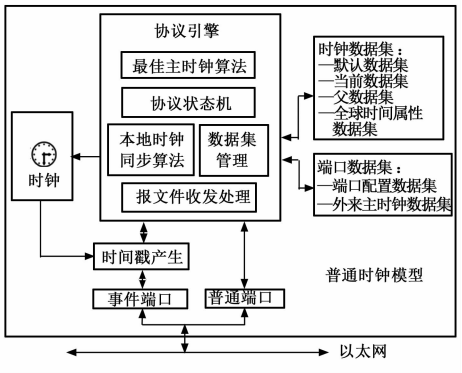

图1 PTP普通时钟模型

地时钟端口当前应该处于的状态，并作相应的端口状态转换，为接下来系统的时钟同步做准备。

(2) 本地时钟同步算法：只有处于从时钟状态的端口才会执行本地时钟调整算法。从时钟通过网络和主时钟交换报文，可以获得4个有效的报文时间戳，计算出和主时钟的时间偏差，并调整本地时钟，实现和主时钟时间同步。参与时钟同步的PTP报文包括4种：同步报文（Sync）、跟随报文（Follow_Up）、延时请求报文（Delay_Req）以及延时应答报文（Delay_Resp）。

(3) 协议状态转换机制：协议状态机实现时钟端口在不同的状态之间进行切换，管理着整个协议软件的运行状况。协议对时钟端口可能的9种状态进行了定义：初始化状态（PTP_INITIALIZING）、超主时钟状态（PTP_PRE_MASTER）、主时钟状态（PTP_MASTER）、从时钟状态（PTP_SLAVE）、故障状态（PTP_FAULTY）、禁止状态（PTP_DISABLED）、听状态（PTP_LISTENING）、被动状态（PTP_PASSIVE）、未标定或暂时状态（PTP_UNCALIBRATED）。一个时钟端口同一时刻只能处于其中的一个状态，状态之间的转换依赖于状态事件的发生。

(4) 报文收发处理：PTP报文分为两类：事件报文（Sync、Delay_Req）和普通报文（Follow_Up、Delay_Resp和Management）。因此，协议定义了两个不同的端口用于网络收发，事件端口用于事件报文的收发，普通端口用于普通报文的收发。根据IEEE1588协议规定，只需要对事件报文加盖时间戳。

① 报文接收：报文接收处理模块接收到PTP报文后，按照协议规定格式进行解包，根据不同的报文对应不同的处理方法。

② 报文发送：报文发送处理模块主要实现将PTP报文按照规定格式进行封装，并通过UDP广播发送出去。

(5) 数据集管理：负责时钟数据集和端口数据集的存储与更新。数据集是协议计算或软件运行中使用到的一些参数的集合。数据集分为两种：时钟数据集和端口数据集。

时钟数据集包括：

①默认数据集 — 描述普通时钟的属性；

②当前数据集 — 与时钟同步相关的属性；

③父数据集 — 描述时钟端口的父时钟相关属性；

④全球时间属性数据集 — 描述时间属性。

端口数据集包括：

①端口配置数据集 — 描述时钟端口属性；

②外来主时钟数据集 — 描述外来主时钟的属性。

2 嵌入式Linux设备对PTP的支持

基于硬件辅助的PTP实现与纯软件的PTP实现的不同之处就是它使用PTP报文硬件时间戳，并且拥有独立的PTP硬件时钟而不是使用系统时钟。PTP应用软件的设计需要Linux设备为用户空间提供两方面服务：(1) 如何获得PTP报文的硬件时间戳；(2) 如何实现对PTP硬件时钟的控制。

2.1 获取PTP报文硬件时间戳

Linux系统中，在没有标准系统调用接口支持获取网络报文硬件时间戳之前，一般通过在MAC（媒体访问控制器）驱动实现私有的ioctl函数，将网络报文硬件时间戳返回给用户。由于没有标准的实现方法和接口，开发出来的IEEE 1588协议应用软件严重依赖于具体的硬件平台，可移植性差。因此，Linux系统内核在2.6.30版本开始添加了对网络报文硬件时间戳的支持，为用户层应用程序的实现与网络设备驱动层的实现提供了标准的接口[8]。

(1) 用户层编程接口：

网络报文硬件时间戳的获取参照了Linux此前已有的软件时间戳获取方法，在原有socket设置选项SO_TIMESTAMP和SO_TIMESTAMPNS的基础上，增加了SO_TIMESTAMPING选项。SO_TIMESTAMP和SO_TIMESTAMPNS选项利用系统时钟实现软件加盖时间戳，分别返回微秒精度和纳秒精度的时间值；SO_TIMESTAMPING同时支持了软件加盖时间戳和硬件加盖时间戳两种方式，通过setsockopt()函数对其进行参数设置可以实现不同的返回结果。

若要使用socket选项SO_TIMESTAMPING设置返回硬件时间戳，应调用socket的ioctl()函数对硬件进行初始化，指定需要哪一类接收报文加盖时间戳以及是否对发送报文加盖时间戳，控制命令为SIOCSHWTSTAMP。

网络接收数据时，接收时间戳作为一笔附属数据存放在原始协议级SOL_SOCKET下的SO_TIMESTAMPING类型控制信息里，可以用recvmsg()函数获取接收数据和控制信息，然后使用CMSG_FIRSTHDR()与CMSG_NXTHDR()宏遍历所有的附属数据，最终获得接收硬件时间戳。网络发送数据时，当需要发送时间戳，发送数据结构skb被克隆一份，加入时间戳后被添加到套接字错误消息队列中，这样使得发送时间戳和接收时间戳的获取方法一样，可用recvmsg()函数从错误消息队列里获取发送数据以及控制信息。

(2) 内核驱动接口：

Linux内核为设计具有网络报文硬件时间戳功能的网络设备驱动提供标准接口支持，使得上层协议栈实现与具体硬件设备无关。网络硬件设备包括MAC控制器（数据链路层）和PHY芯片（物理层），硬件时间戳在物理层产生。网络设备驱动主要实现从硬件获得发送报文和接收硬件时间戳，通过内核标准API将其传给协议栈处理。

与硬件时间戳实现有关的内核接口函数声明在Linux/include/linux/skbuff.h文件中，函数实现在/Linux/net/core/timestamping.c文件，主要包括：

skb_hwtstamps()函数用于获得sk_buff数据结构中的硬件时间戳指针；

skb _ clone _ tx _ timestamp（）函数实现对网络发送报文类型的判断，如果需要加盖硬件时间戳，则克隆一份 skb 结构，并通知 PHY 驱动获取发送时间戳；

skb _ complete _ tx _ timestamp（）函数将 PHY 驱动获得的发送硬件时间戳添加到克隆的 skb 结构中，并将该 skb 添加到套接字错误消息队列；

skb _ defer _ rx _ timestamp（）函数实现对网络接收报文类型的判断，如果需要加盖硬件时间戳，则通知 PHY 驱动获取接收时间戳，并将时间戳添加到 skb 中。

2.2　PTP 硬件时钟控制

为了完善对实现高精度 IEEE 1588 时钟同步的支持，Linux 3.0 内核版本开始集成了 PTP 时钟类驱动。类驱动采用面向对象的思想，在内核中实现了具有相同特征的一类硬件设备的共同属性部分，简化了具体驱动的开发工作，并为应用层提供一致的接口。PTP 时钟类驱动支持基本的时钟操作和辅助的时钟功能[9]。

基本的时钟操作包括：设置时间、获取时间、调整时间偏差以及调整时钟频率；

辅助的时钟功能包括：创建单次或周期性报警、为外部事件加盖时间戳、可配置的周期信号输出以及 PPS 系统等。

（1）用户层编程接口：

某一具体的硬件时钟设备驱动注册为 PTP 时钟类驱动的同时，也被注册为字符设备驱动。因此，用户空间的接口也分为两部分：字符设备操作接口以及 POSIX 时钟接口。字符设备接口用于实现辅助的时钟功能部分，即用户可以通过 ioctl（）函数设置单次或周期性报警、使能外部事件时间戳、配置周期信号输出以及使能 PPS 等，通过 read（）和 poll（）函数可读取生成的事件时间戳；POSIX 时钟接口实现时钟的基本操作，接口函数包括：

①clock _ gettime（），获取当前时间；

②clock _ settime（），设置时间；

③clock _ adjtime（），根据调整时间偏移或时钟频率。

（2）内核驱动接口：

Linux 内核 linux/ptp _ clock _ kernel.h 文件里定义了一个 struct ptp _ clock _ info 类型的数据结构，该结构对硬件时钟设备进行了抽象。定义如下：

```
struct ptp_clock_info {
  struct module *owner;
  char name[16]; //设备名
  s32 max_adj; //最大频率调整值
  int n_alarm; //可编程报警的数目
  int n_ext_ts; //事件时间戳的数目
  int n_per_out; //周期信号输出的数目
  int pps; //PPS 使能
  int (*adjfreq)(struct ptp_clock_info *ptp, s32 delta);
  int (*adjtime)(struct ptp_clock_info *ptp, s64 delta);
  int (*gettime)(struct ptp_clock_info *ptp,
          struct timespec *ts);
  int (*settime)(struct ptp_clock_info *ptp,
          const struct timespec *ts);
  int (*enable)(struct ptp_clock_info *ptp,
          struct ptp_clock_request *request, int on);
};
```

可见，内核接口与用户层接口是对应的，enable（）函数实现辅助的时钟功能。硬件时钟设备驱动的开发就是对该数据结构中的 adjfreq、adjtime、gettime、settime、enable 等方法进行具体的实现，然后将其注册到 PTP 时钟类驱动中，剩余的工作由类驱动来完成。

3　系统设计

3.1　硬件系统

DP83640 是美国国家半导体公司专门针对高精度时间同步 IEEE1588 的应用而设计的一款集成了硬件时间标记单元的 PHY（物理层芯片），支持 IEEE1588 v1 和 v2 版本，主要特征包括以下 3 个方面[10-11]。

（1）产生 IEEE 1588 时钟：

外部时钟振荡源通过 DP83640 内部相位产生模块（PGM）可以产生一个最小分辨率为 8 ns 的 IEEE1588 参考时钟，可通过直接读写、时间加/减调整、频率控制和临时速率调整 4 种方式对其进行操作。

（2）为 PTP 报文加盖硬件时间戳：

DP83640 内部发送端和接收端都有一个数据包解析器，在发送和接收数据时能够自动检测出 PTP 事件报文，并对其加盖硬件时间戳。

（3）定时触发与外部事件时间戳：

DP83640 提供了 12 个用于 IEEE 1588 时基触发或者事件捕获的 GPIO。通过配置可实现不同的触发信号输出；当配置用作事件捕获时，GPIO 能够监测外部输入事件，并对其加盖时间戳。

处理器采用三星公司 ARM920T 内核的 S3C2440，以太网 MAC 控制器选用 DAVICOM 公司 DM9000。DM9000 通过 MII 接口与 DP83640 相连，采用 MII 连接模式时，DP83640 时钟输入端要求接入一个 25MHz 的时钟源，硬件系统设计如图 2 所示。

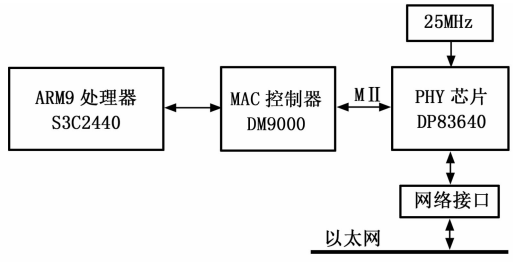

图 2　硬件系统框图

3.2　设备驱动实现

操作系统中，设备驱动为上层应用屏蔽了底层硬件的具体实现细节，用户层只需要调用应用程序编程接口就可以让硬件完成要求的功能，采用这种软硬件分离的思想，大大提高了应用程序的可移植性。

设备驱动包括 DM9000 和 DP83640 两个部分，其中 DP83640 驱动又分为了 PHY 功能部分和 IEEE 1588 时钟功能部分，如图 3 所示。DP83640 驱动的 PHY 功能部分注册为挂靠在 MII 总线的一个 PHY 设备，与 DM9000 驱动组成网络设备驱动，实现网络数据收发以及 PTP 报文硬件时间戳；1588 时钟功能部分通过定义 ptp _ clock _ info 结构体，注册为 PTP 时钟类驱动下的一个特定硬件时钟设备驱动，实现基本的时钟

操作和辅助的时钟功能。

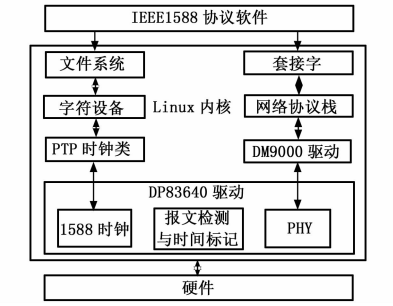

图 3 设备驱动实现

设备驱动的实现使用了内核标准 API 函数，由内核完成与用户层的对接，只需要完成与具体硬件相关的操作，开发难度大大降低。

3.3 IEEE 1588 协议软件实现

基于上述 linux 系统标准的硬件时间戳获取 API 和 PTP 时钟控制 API，修改和移植了开源 IEEE1588 协议软件 PTPd (PTP daemon)。PTPd 基于 GNU/ Linux 操作系统，采用纯软件的方法实现了一个 PTP 普通时钟模型，可以获得微秒级时钟同步精度，它已成为开源社区中的一个参考例程。

4 时钟同步测试

4.1 两台设备通过交叉网线直接相连

两台自行设计的时钟同步设备通过 2 m 交叉网线直接相连，同时运行 IEEE 1588 协议软件。通过执行最佳主时钟算法，其中一台设备被选择为主时钟，另一台设备作为从时钟，设置同步时间间隔为 2 s，进行了 3 个小时的同步测试。从时钟将每次计算得到的与主时钟的时间偏差值通过串口输出，PC 机收集到每次的时间偏差用 Matlab 进行统计分析，结果如图 4 所示，从测试开始，经历一段时间后，从时钟和主时钟的时间偏差能够稳定在±100 ns 以内。

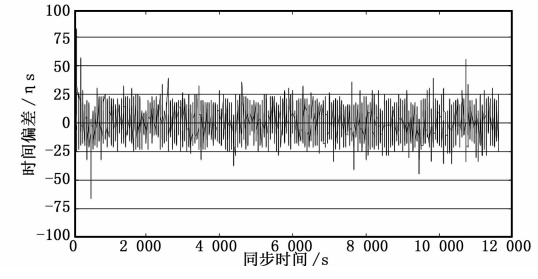

图 4 同步测试结果（直接相连）

4.2 两台设备通过交换机相连

通过交互机相连进行测试的目的是为了观察增加网络元件对时钟同步精度的影响。一台设备作为主时钟，另一台设备作为从时钟，同步时间间隔设为 2 s，进行了 2 个小时的同步测试。测试结果如图 5 所示，观察发现时钟同步精度基本能稳定在±300 ns 以内。与直连方式测试数据进行比较可知，增加网络元件会影响时钟同步的精度，因此在组建 PTP 系统时，应使用边界时钟代替普通的交换机或路由器，以获得更高精度的时钟同步。

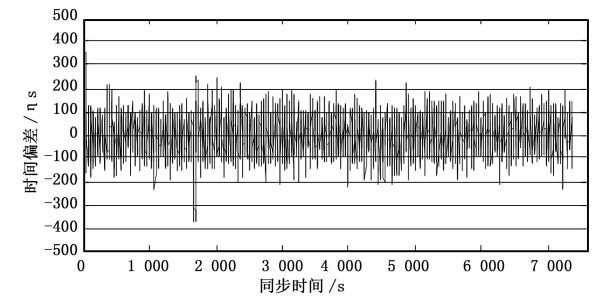

图 5 同步测试结果（通过交换机相连）

5 结论

测试结果表明，利用在物理层对同步事件报文加盖时间戳的方式实现 IEEE1588 时钟同步协议，可以实现纳秒级的时钟同步精度。另外，本地硬件时钟的精度与稳定度以及网络元件的插入也是影响时钟同步精度的重要因素。目前，IEEE 1588 时钟同步协议获得了广泛的研究与应用，但大多数实现方法严重依赖于具体的硬件平台，通用性差。本文基于 Linux 系统的标准 API 实现了高精度时钟同步的 IEEE 1588 协议，程序可移植性好，实现方法更具一般性，可为各种 IEEE 1588 应用设计提供参考。

参考文献：

[1] Correll K, Barendt N, Branicky M. Design considerations for software only implementations of the IEEE 1588 precision time protocol [A]. in Proceedings of the IEEE 1588, Conference [C], Zurich, October 2005.

[2] 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.

[3] 刘兆庆，潘邵武，张毅刚. 基于 DP83640 的 IEEE1588 应用研究 [J]. 测控技术，2012，31 (9)：80－82.

[4] 孙中尉. IEEE1588 高精度网络时间同步应用研究 [D]. 陕西：中国科学院研究生院，2010.

[5] IEEE Instrumentation and Measurement Society. IEEE standard for a precision clock synchronization protocol for networked measurement and control systems [S]. New York，USA：IEEE，2002.

[6] 庾智兰，李 智. 精确时钟同步协议最佳主时钟算法 [J]. 电力自动化设备. 2009，29 (11)：74－77.

[7] 杨传顺，李国华，钱幸存. 精确时钟协议的最优主时钟算法 [J]. 计算机测量与控制. 2011，19 (9)：2269－2271.

[8] Ohly P. Precision time protocol — temporary fork with support for hardware timestamping [EB/OL]. http：//github.com/pohly/ptpd，2009.

[9] Richard Cochran , Cristian Marinescu. Design and Implementation of a PTP Clock Infrastructure for the Linux Kernel [A]. International IEEE Symposium [C]，2010.

[10] National Semiconductor Corporation. Precision PHYTER－IEEE 1588 Precision Time Protocol Transceiver [DB/OL]. http：//www.docin.com/p－203979498. html，2010.

[11] National Semiconductor Corporation. National Semiconductor Ethernet PHYTER Software Development Guide [DB/OL]. Revision 1.96，http：//wenku.baidu.com，2009.