面向安全关键嵌入式系统的时钟同步设计①

李 翔, 吴向阳, 张 激

(中国电子科技集团公司 第三十二研究所, 上海 201801)

摘 要: 为了满足安全关键性实时嵌入式系统对实时行为正确性和确定性的要求, 本文应用 ETSEC 网络控制器的硬件时间戳特性, ReWorks 嵌入式实时操作系统的实时特性, 结合 IEEE 1588 精密度时钟同步协议, 设计并实现了一个面向安全关键性领域的时钟同步系统原型. 其同步精度达到 100 纳秒以内, 并通过软件插桩测试和验证了该时钟同步系统的正确性和精度.

关键词: 时钟同步; 安全关键性; IEEE 1588; 实时嵌入式系统

Design of Clock Synchronization System for Security Critical Embedded System

LI Xiang, WU Xiang-Yang, ZHANG Ji

(The 32nd Research Institute of China Electronics Technology Group Corporation, Shanghai 201801, China)

Abstract: In order to meet the requirements of real-time behavior correctness and certainty of safety critical real-time embedded system, the paper shows a design and implementation of clock synchronization system with hardware timestamp characteristics of the of ETSEC Ethernet device controller. The real-time embedded operating system ReWorks and IEEE 1588 precision clock synchronization protocol is adopted for the implementation. The synchronization accuracy is less than 100 ns, and the accuracy and precision of the clock synchronization system are tested and verified.

Key words: clock synchronization; safety critical; IEEE1588; real-time embedded OS

在航空航天、轨道交通、军事以及能源电力等安全攸关领域对实时嵌入式处理节点具有很高的可靠性和安全性要求. 在安全关键性实时嵌入式系统中, 为了减少和降低风险, 除了选用高可靠电子元器件外, 通常采用多模冗余的方式提高实时嵌入式系统的可靠性和降低风险概率. 多模冗余实时嵌入式系统中处理节点间时钟同步是确保系统正确实时性行为的关键, 如在对数据表决时, 要求不管是输入数据还是输出数据, 都要在时刻上保持一致.

在多模冗余实时嵌入式系统中每个实时嵌入式处理节点的时钟属性差异和初始时间不同, 需要使用时钟同步技术来补偿时钟误差, 构建并维护一个统一时间基, 从而确保系统实时行为的正确性. 关键词库的结合大大提高了信息抽取算法的准确性和通用性, 基于 Web 信息抽取的混合交通出行方案生成与表示系统的成功实验也证明了本文提出的 Web 信息抽取算法的实用性.

1 概述

目前安全关键性应用领域主要采用基于报文交换网络的时钟同步技术来补偿时间误差, 同时需要考虑时钟漂移和网络消息延迟不确定性两个主要的误差源. 随着通信频率的降低, 时钟漂移不确定性增加, 而消息延迟不确定性是个常量, 因此时钟漂移通常比消息延迟对时钟同步精度影响更大. 简单的解决方法是降低同步间隔, 由于同步过程中需要通过网络消息传递和处理时间信息, 从而增加同步需要的计算和内存资源, 这意味着同步系统设计时需要在计算资源和同步精度之间做权衡.

针对安全关键性领域应用一般要求时钟同步精度

① 收稿时间:2016-09-25;收到修改稿时间:2016-11-17 [doi:10.15888/j.cnki.csa.005806]

在亚微妙级和实时行为的正确性确定性, 本文应用ETSEC网络控制器的硬件时间戳特性和ReWorks操作系统的实时特性, 结合 IEEE 1588 精密度时钟同步协议, 设计并实现了一个面向安全关键性领域的时钟同步系统原型.

IEEE1588 协议是用使用网络通讯、本地计算和分发等技术来实现的精确同步时钟和控制系统. IEEE 1588 标准的全称是 IEEE 1588 Precision Clock Synchronization Protocol (精密时钟同步协议标准), 简称 PTP(Precision Timing Protocol). PTP 通过以下两步来达到时钟同步, 先将网络中的时钟组织成一个主从层次结构; 再在主从时钟之间交换带时间戳的消息.

目前对网络节点中的时间同步协议的研究主要集中在优化延时和提升同步效果上. 文献[1]和文献[2]研究在不同环境下对 PTP 的影响和实现. 文献[3]采用主时钟选择算法, 改进了报文帧传输过程中的延时, 提升了时钟同步效果. 文献[4]提出了基于区分服务调度模型的同步报文路径延时误差修正方法, 实现了同步时钟误差的修正.

这里将研究内容定位于优化延时上, 通过修改打时间戳的位置和优化 ETSEC 网卡驱动来提高时钟同步效果.

2　时钟同步系统设计

2.1 BMC 算法(最佳主时钟算法)

PTP 协议的第一步中每个时钟广播自身属性给同一个域的其他时钟. 最佳主时钟算法独立运行在每个节点, 主要是确定当前域中的最佳时钟. 最佳时钟就成为主时钟, 其他时钟是从时钟. BMC 算法将本地网络节点组织成一个主从层次结构如图 1 所示.

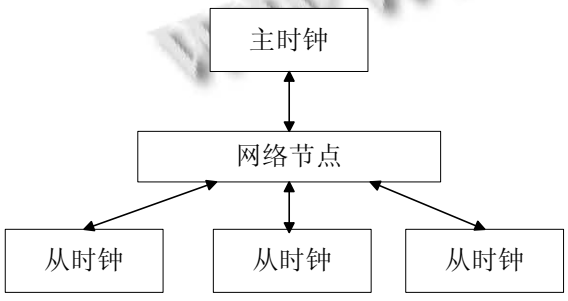

图 1　主从时钟模型

2.2 PTP 时钟同步模型

PTP 时钟同步模型及原理如图 2 所示.

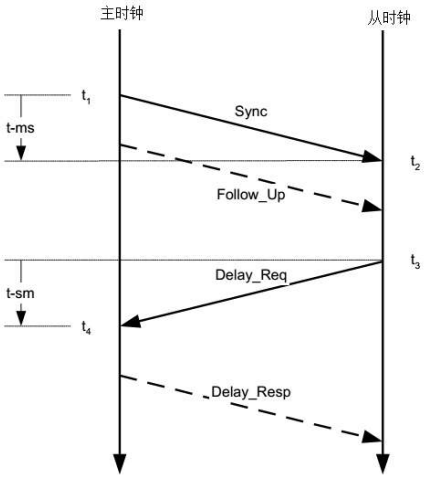

图 2　PTP 时钟同步模型

PTP 时钟同步的思想和过程如下:

① 主时钟周期性地发送 Sync 报文, 并将发送时刻 t1 发给从时钟;

② 从时钟收到 Sync 报文, 并记录收到报文的时刻 t2;

③ 主时钟有以下两种方式将 t1 传递给从时钟: 将 t1 时间戳放入 Sync 报文, 这需要高精度的硬件支持; 将 t1 时间戳放入 Follow Up 报文;

④ 从时钟给主时钟发送 Delay_Req 报文并记录发送时刻 t3;

⑤ 主时钟收到 Delay_Req 报文并记录收到的时刻 t4;

⑥ 主时钟将 t4 放入 Delay_Resp 报文并传递给从时钟.

经过上述过程之后, 可得到两个时钟的平均延时 (t-ms+t-sm)/2, 其中 t-ms = t2-t1, t-sm = t4-t3.

2.3 端口状态管理

所有的时钟都要遵守端口状态机, 而端口状态机共有 9 种状态[6], 如图 3 所示. 网络中所有的端口都在上面 9 种状态中转换, 共同完成同步的过程.

① INITIALIZING. 当一个端口处于 INITIALIZING 状态时, 该端口初始化数据集、硬件和通讯设备. 不允许任何时钟的端口发送 PTP 报文. 如果边界时钟的一个端口处于 INITIALIZING 状态, 那么所有的端口都应处于 INITIALIZING 状态;

② FAULTY. 即协议中的 fault 状态. 当一个端口

处于 FAULTY 状态时, 只响应网络中的 management 报文, 而自身不发出其他任何 PTP 报文;

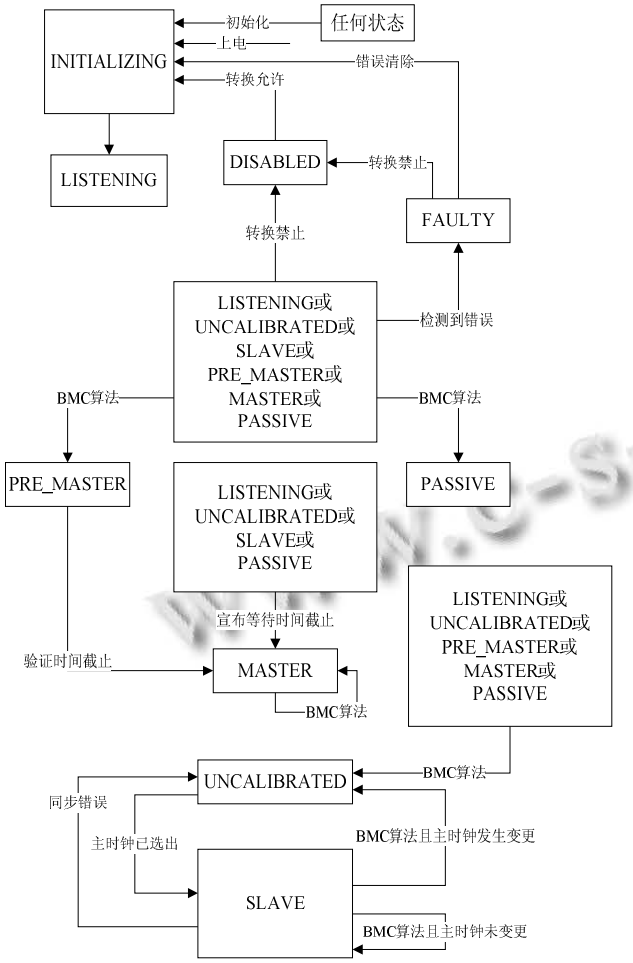

图 3 端口状态转化图

③ DISABLED. 处于DISABLED状态的端口, 自身不发出任何PTP报文, 并丢弃除了management报文之外的所有 PTP 报文;

④ LISTENING. 处于 LISTENING 状态的端口, 在一定时限内侦听等待主时钟发来的 Announce 报文. 目的是能够在子域中顺序地加入时钟. 处于此状态的端口只能发出 Pdelay_Req,Pdelay_Resp,Pdelay_Resp_Follow_Up 报文或是应答其他的 management 报文;

⑤ PRE_MASTER. 处于该状态的端口与 MASTER 状态很类似, 但不同的是只能发出 Pdelay_Req,Pdelay_Resp,Pdelay_Resp_Follow_Up 报文或是 management 报文;

⑥ MASTER. 处于该状态的端口为主时钟端口. 定期地向网络中发送 PTP 同步报文;

⑦ PASSIVE. 处于该状态的端口只能发出 Pdelay_Req,Pdelay_Resp,Pdelay_Resp_Follow_Up 报文或是应答其他的 management 报文;

⑧ UNCALIBRATED. 当在子域中侦测到多个的主时钟时, 端口会处于该状态. 同时, 同步协议会通过算法选出一个最合适的主时钟, 其他端口都准备去和该主时钟同步;

⑨ SLAVE. 处于该状态的端口为从时钟端口, 周期性地与主时钟同步.

2.4 同步协议模型

同步协议主要由初始化、端口状态转换、BMC 算法、报文控制、时间戳解析 5 个部分组成. 它们之间的关系如图 4 所示.

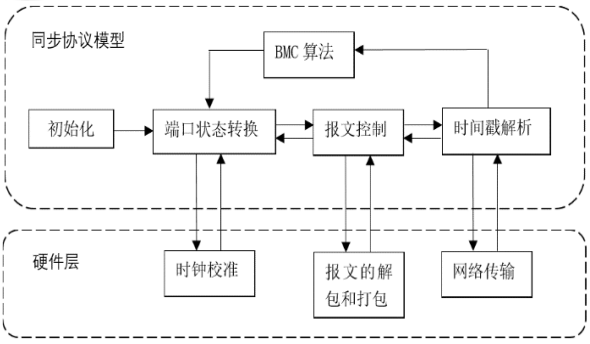

图 4 同步协议模型

系统初始化之后, 各个端口进行初始化并按照 BMC 算法进行调整, 对时钟进行实时的校准. 在 PTP 报文中, 需要打时间戳的是 Sync,Delay_Req,Pdelay_Req,Pdelay_Resp 报文, 而 Announce,Follw_Up,Delay_Resp,Pdelay_Resp_Follow_Up,Management,Signaling 报文不需要打时间戳. 调整过程中周期性往复循环上图过程. 震荡性地将网络中各端口的时钟校准.

2.5 PTP 驱动结构

PTP 底层驱动的设计包括以下几个部分: PTP 时钟操作、时间戳的存储、时间戳的解析和过滤, 和在报文中打入时间戳的功能.

传统平台上, 没有硬件支持的 1588 同步协议只有在网络层才能处理 PTP 数据, 属于纯软件实现, 接收时间戳和发送时间戳都非常的滞后, 同步效果较差. P1010 平台所集成的 eTSEC 网卡支持在 MAC 层给报文打时间戳, 利用这一特性可以给 1588 同步协议提供非常精确的时间戳, 提高同步效果[8]. 开启 eTSEC 网

卡硬件打时间戳功能有以下 8 个条件[9]:

① TMR_CTRL[RTPE] = 1. Time control register 寄存器的 RTPE 位(第 16 位)要设为 1;

② TxBD[TOE] = 1. Transmit data buffer descriptors(TxBD)寄存器的 TOE 位(第 14 位)设为 1. TxBD 是管理 MAC 层发送的内存区域;

③ TxBD 的[data buffer pointer]是 8 字节对齐的;

④ TxBD[data length]的值为 8;

⑤ TxFCB[PTP] = 1. Transmit frame control block 寄存器的 PTP 位(第 15 位)要设为 1;

⑥ TxFCB 之后紧跟着至少 16 字节的 TxPAL;

⑦ TxBD 寄存器的第二块 data buffer pointers 指向 L2 的起始点或者帧数据;

⑧ TxBD 寄存器的第二块 data length 域至少 1024 字节或者帧长度.

结合这些特性, 设计出 PTP 底层驱动的模型, 如图 5 所示.

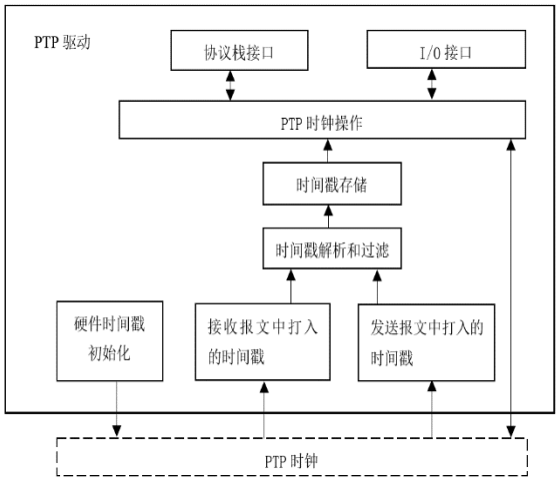

图 5　PTP 驱动结构

3　时钟同步系统实现

3.1 软硬件架构

3.1.1 软件环境

软件环境采用了 ReWorks 操作系统. ReWorks 是中国电子科技集团公司第三十二研究所(华东计算技术研究所)自主研制的嵌入式实时操作系统, 采用先进的面向对象和 POSIX 标准微内核技术开发, 具有强实时性、可裁剪性和可伸缩性, 并特别提供了 VxWorks 兼容层.

3.1.2 硬件环境

硬件环境采用了 PowerPC 构架的 P1010 平台. P1010 是 Freescale 公司 QorIQ 系列通信处理器的一款入门级两核处理器芯片, 具有高性能、低功耗、性价比高的特点.

3.2 部分关键接口的实现

3.2.1 时间戳的解析和过滤接口

该接口解析报文, 并判断其是否为 PTP 报文[10]. 若是 PTP 报文则返回 PTP 头部的位置, 若不是则返回 NULL. 其流程图如图 6 所示.

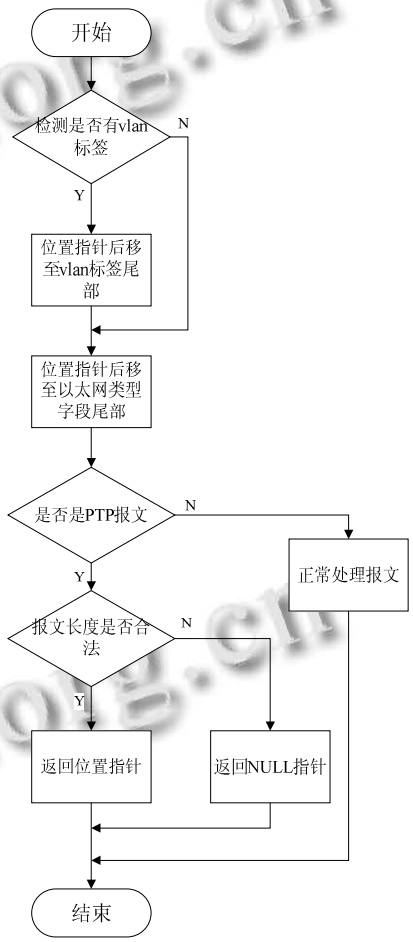

图 6　时间戳的解析和过滤接口流程图

关键代码段如下:

```
static void *motTsec_ptp_parse_packet(M_BLK *pMblk, u16 *eth_type)
    switch (*eth_type)

    {
    case ETH_P_1588:
        ptp_loc = pos;
```

```
            if ((ptp_loc[0] & 0xF) <= 3) {
                /*长度是否足够 */
                if (pMblk->mBlkHdr.mLen >= (((char *)ptp_loc - pMblk->mBlkHdr.mData) +MOT_TSEC_PTP_HEADER_SZE))
                    return ptp_loc;
            }
            break;
        case ETHERTYPE_IP:
            iph = (struct ip *)pos;
            if (ntohs(iph->ip_p) != IPPROTO_UDP)
                return NULL;
            pos += iph->ip_hl * 4;
            udph = (struct udphdr *)pos;
        if(ntohs(udph->dest)!=MOT_TSEC_PTP_EVENT_PORT)
                return NULL;
            ptp_loc = pos + sizeof(struct udphdr);
       if(pMblk->mBlkHdr.mLen>=((ptp_loc-pMblk->mBlkHdr.mData)+MOT_TSEC_PTP_HEADER_SZE))
                return ptp_loc;
            break;
        default:
            break;
        }
```

3.2.2 存储时间戳接口

该接口记录接收报文时的硬件时钟戳, 并将其写入报文, 流程图如图 7 所示.

关键代码段如下:

```
      Void motTsec_ptp_store_rxstamp(ETSEC_DRV_CTRL *pDrvCtrl, M_BLK *pMblk, struct motTsec_ptp_time *rx_ts)
        ptp_loc = motTsec_ptp_parse_packet(pMblk, &eth_type);
        if (ptp_loc == NULL)
            return;
        switch (eth_type) {
        case ETHERTYPE_IP:
        tmp_rx_time.ident.netw_prot=MOT_TSEC_PTP_PROT_IPV4;
            break;
        case ETHERTYPE_IPV6:
        tmp_rx_time.ident.netw_prot=MOT_TSEC_PTP_PROT_IPV6;
            break;
        case ETH_P_1588:
        tmp_rx_time.ident.netw_prot=MOT_TSEC_PTP_PROT_802_3;
            break;
        default:
            return;
        }
        /*data part:identify the PTP message*/
        tmp_rx_time.ident.version = (*(u8 *)(ptp_loc + 1)) & 0X0F;
        tmp_rx_time.ident.msg_type=(*(u8 )(ptp_loc+MOT_TSEC_PTP_HEADER_MES_OFFS)) & 0x0F;
         tmp_rx_time.ident.seq_id=*(unsignedshort*)(ptp_loc+MOT_TSEC_PTP_HEADER_SEQ_OFFS);
memcpy(tmp_rx_time.ident.snd_port_uuid,ptp_loc+MOT_TSEC_PTP_SPID_OFFS,MOT_TSEC_PTP_SOURCE_PORT_LENGTH);
       tmp_rx_time.ts = *rx_ts;
        /* insert timestamp in circular buffer */
        motTsec_ptp_insert(&(pDrvCtrl->rx_timestamps), &tmp_rx_time);
        return;
```

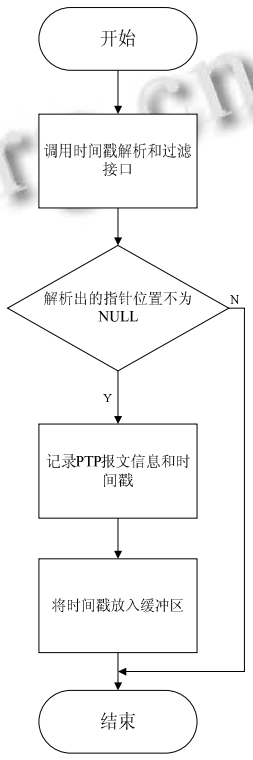

图 7 存储时间戳接口流程图

4 验证与测试

选取两台 P1010 设备, 使用网线分别连接两台设备的 eTSEC 0 网口和 eTSEC1 网口并分别设置其 ip 为 192.168.1.100 和 192.168.1.101, 保证它们在同一网段. 验证与测试方法采用软件插桩测试. 在底层驱动中加入 printk 打印出两台设备的时钟误差. 使用 SecureCRT 记录设备串口的打印. 在两台设备上启动 ReWorks 操作系统后执行 PTP 校准命令开始同步. 并记录两台设备的时钟误差, 得到图 8.

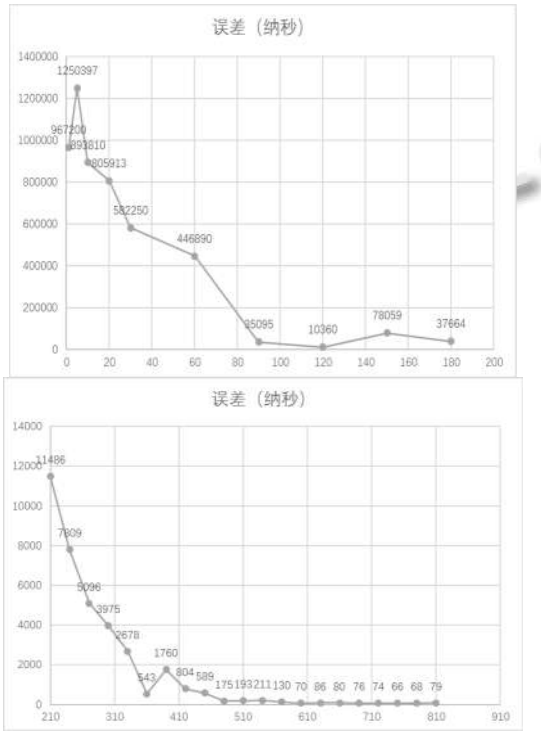

图 8　时钟同步测试结果图

观察上图的数据, 发现在前 3 分钟误差值较大且振荡幅度很大但误差在向减小的趋势变化. 随着时间的进行, 在 400 秒之后两时钟的误差在稳定减小且在 10 分钟之后控制在 80ns 左右. 由此得出结论, PTP 的算法实现是有效的, 而且传统的 PTP 实现只能将误差控制在微秒级[11], 通过本文方法实现的 PTP 算法可以将误差缩小至纳秒级.

5 结语

本文简要分析了时钟同步技术在安全关键性实时嵌入式系统应用的重要性; 详细描述面向安全实时应用的时钟同步系统原型设计; 并在国产实时嵌入式系统 ReWorks 上实现了时钟同步系统原型, 应用 ETSEC 网络控制器的硬件时间戳特性和 ReWorks 操作系统的实时特性, 结合 IEEE 1588 高精度时钟同步协议, 时钟同步系统精度达到 100ns 以内.

本文工作一方面拓展了 IEEE 1588 高精度时钟同步协议和国产实时嵌入式 Reworks 在安全关键性实时嵌入式系统中的应用, 另一方面针对安全关键性实时嵌入式系统对安全性和故障导向安全原则要求, 需要在 IEEE 1588 高精度时钟同步协议上进一步考虑, 这正是未来需要研究的重点.

参考文献

1 梁姣.基于 IEEE1588 协议的航电以太网时钟同步系统的研究与实现[硕士学位论文].南京:东南大学,2015.

2 朱望纯,钟震林,覃斌毅.嵌入式 Linux 设备的高精度 IEEE 1588 时钟同步实现.计算机测量与控制,2014,22(5):1619–1622.

3 王跃飞,杨锦,张利,张本宏.汽车 CAN 系统精确时钟同步机制研究.电子测量与仪器学报,2014,28(1):22–28.

4 黎锐烽,曾祥君,李泽文,王阳.IEEE 1588 同步时钟网络时延误差的分析及修正.电力系统自动化,2012,36(12):82–87.

5 IEEE Std 1588 TM –2008 IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems.2008.

6 赵洪锟.基于嵌入式 linux 的 IEEE1588 协议的分析与实现[硕士学位论文].济南:山东大学,2012.

7 IEEE Std 1588TM–2002 IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems.2002.

8 陈永标,方兴其,岑宗浩.IEEE1588 协议中时钟同步性能的影响因素以及时间戳的生成方式分析.微型电脑应用,2009,25(4):4–6.

9 P1010 QorIQ Integrated Processor Reference Manual. Freescale. 2015.

10 Song EY, Kang L. An application framework for the IEEE 1588 standard. IEEE International Symposium on Precision Clock Synchronization for Measurement. 2008.23–28.

11 戴宝峰,崔少辉,王岩.基于 IEEE1588 协议的时间戳的生成与分析.仪表技术,2007,(7):16–17.