Rockchip Developer Guide GMAC PTP1588

文件标识：RK-KF-YF-C33

发布版本：V1.0.0

日期：2024-12-20

文件密级：□绝密 □秘密 □内部资料 ■公开

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址： 福建省福州市铜盘路软件园A区18号

网址： www.rock-chips.com

客户服务电话： +86-4007-700-590

客户服务传真： +86-591-83951833

客户服务邮箱： fae@rock-chips.com

前言

概述

本文提供 Rockchip 平台以太网 GMAC 接口的 PTP1588 功能使用文档。

产品版本

芯片名称	内核版本
RK3588、RK3576、RK3568、RK3506、RV1126、RK2118	ALL

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

版本号	作者	修改日期	修改说明
V1.0.0	吴达超	2024-12-20	初始版本

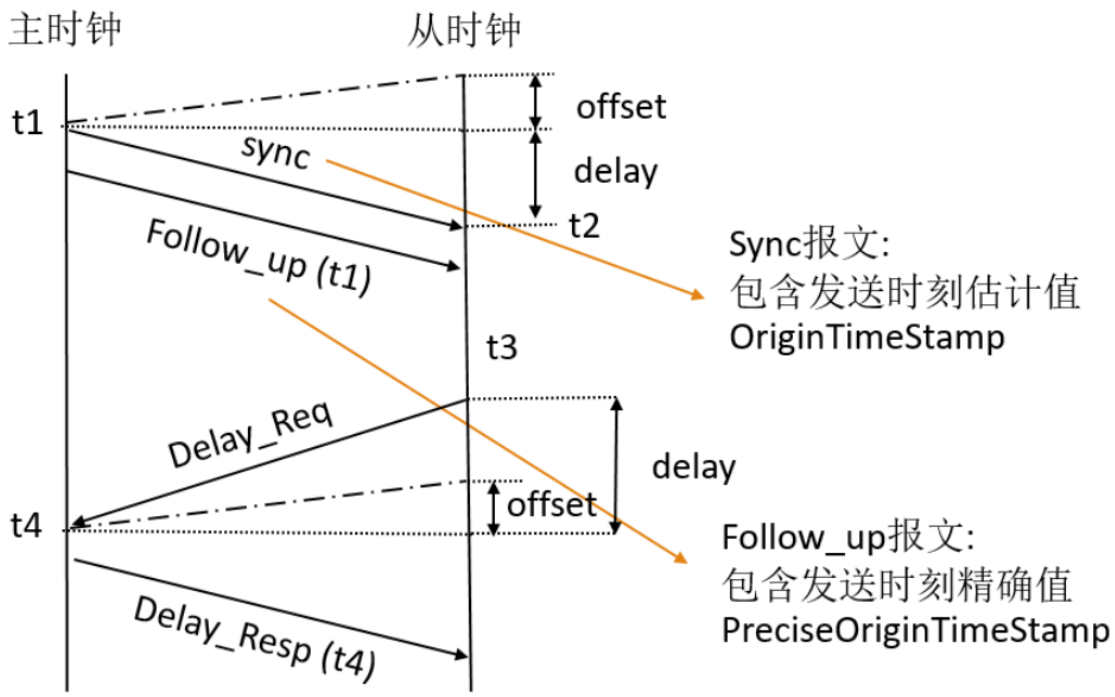
目录

1. PTP1588 实现原理

PTP1588 全称是网络测量和控制系统的精密时钟同步协议标准，基本功能是是分布式网络中所有时钟都与最精确的时钟保持同步，它定义了一种精确的时间协议PTP(precision Time Protocol)，用于对标准以太网或其他支持多播技术的终端设备中的时钟进行亚微妙级别的同步，IEEE 的 PTP1588 用于需要时钟精度比NTP高的局域分布系统。GMAC 控制器支持 PTP1588 v1 和 v2 版本。

常见的 PTP1588 同步流程 **Delay Request-Response Mechanism**（延时响应机制）：

1.1 延时响应机制的报文收发流程



- 主设备周期性的向其所有节点广播 PTP 同步消息，此消息在 t1 离开主设备的系统。记录此数据包离开 GMII 或 MII 上的以太网端口捕获此时间 t1。
- 从设备接收同步消息，并使用其时间参考捕获准确时间 t2。
- 主设备向从设备发送 Follow_up 消息，其中包含 t1 信息以供以后使用。（由于 sync 报文不可能携带精确的报文离开时间，所以我们在之后的 Follow_up 报文中，将 sync 报文精确的发送时间戳 t1 封装起来，发给从时钟）。
- 从设备向主设备发送 Delay_Req 消息，并记录此数据包离开 GMII 或 MII 接口的准确时间 t3。
- 主设备接收消息，捕获消息进入其系统的准确时间 t4。
- 主设备在 Delay_Resp 消息中将 t4 信息发送给从设备。
- 从设备使用 t1、t2、t3 和 t4 四个值将其本地定时参考同步到主设备的定时参考。

1.2 时钟偏差 & 网络延时

offset : 时钟间偏差（主从时钟之间存在时间偏差，偏离值就是 offset）

delay : 网络延时（报文在网络中传输带来的延时）

从时钟可以通过 t1,t2,t3,t4 四个精确时间戳信息，得到主从时钟偏差 offset 和传输延时 delay:

$$delay = \frac{(t_2 - t_1) + (t_4 - t_3)}{2}$$

$$offset = \frac{(t_2 - t_1) - (t_4 - t_3)}{2}$$

从时钟得到 offset 和 delay 之后就可以通过修正本地时钟进行时间同步。

1.3 PTP 报文

- PTP Packets Over IPv4

  IP Multicast Address (IEEE 1588 version 1)，Multicast IPv4 addresses allowed:

  ■ 224.0.1.129

  ■ 224.0.1.130

  ■ 224.0.1.131

  ■ 224.0.1.132

  IP Multicast Address (IEEE 1588 version 2):

  ■ PTP Primary multicast address: 224.0.1.129

  ■ PTP Pdelay multicast address: 224.0.0.107

- PTP Frames Over IPv6

  ■ PTP Primary multicast address: FF0x:0:0:0:0:0:0:0:181 (Hex)

  ■ PTP Pdelay multicast address: FF02:0:0:0:0:0:0:0:6B (Hex)

- PTP Packets Over Ethernet

  All PTP messages can use any of the following multicast addresses:

  ■ 01-1B-19-00-00-00

  ■ 01-80-C2-00-00-0E

1.4 GMAC PTP1588 工作原理

GMAC PTP1588 的系统时间生成器模块是使用参考时钟 (clk_ptp_ref) 进行更新，此时间是获取在 *GMII/RMII 接口上发送或接收的以太网数据包的快照 (时间戳) 的来源。

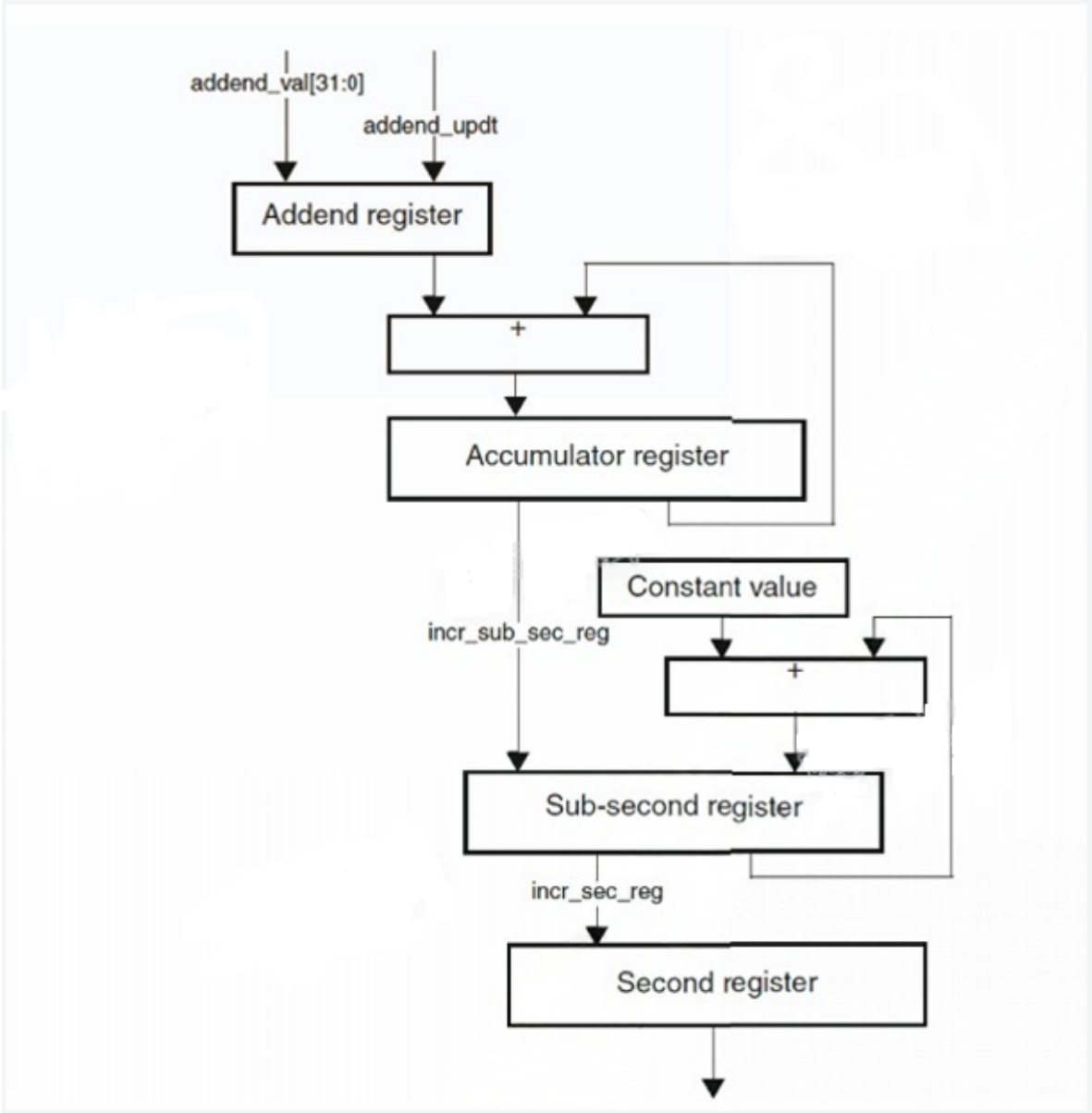
系统时间由两部分组成，分别存在于寄存器 MAC_System_Time_Seconds 和 MAC_System_Time_Nanoseconds，纳秒字段支持以下两种模式：

- Digital rollover mode：在此模式下，纳秒字段中的最大值为 0x3B9A_C9FF，即（10e9-1）纳秒。

- Binary rollover mode：在此模式下，纳秒字段翻转并递增秒值 0x7FFF_FFFF 之后的字段，精度约为每比特约0.466 ns。

系统时间被更新通过设置 TSUPDT bit，在设置之前需要（添加或减去）与指定的值 MAC_System_Time_Seconds_Update 和 MAC_System_Time_Nanoseconds_Update 寄存器。

如果只更新系统时间，这个方式抖动误差比较大，所以提供一种精度更高的更新时间方式。



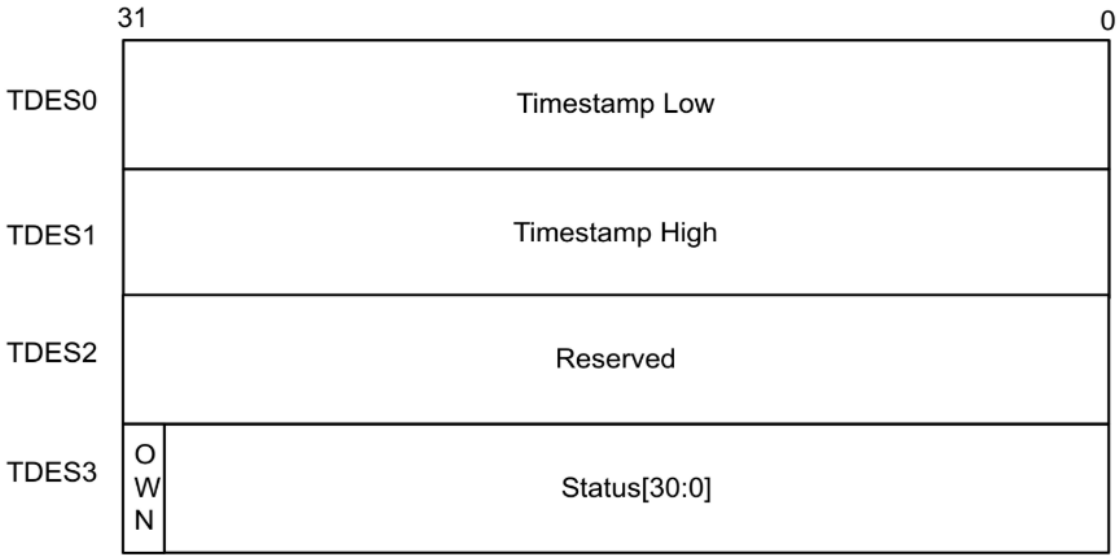
如上图所示，在校正 GMAC PTP1588 的系统时间方法中，从时钟 (clk_ptp_ref) 相对于主时钟 (IEEE 1588-2002 中定义) 的频率偏移或频率漂移在一段时间内进行校正。这有助于保持线性时间，并且不会在 PTP 同步消息间隔之间的参考时间中引入剧烈变化 (或大抖动)。在此方法中，累加器将加数寄存器的内容相加，如上图所示，累加器生成的算术进位用作脉冲来增加系统时间计数器。累加器是 32 位寄存器，充当高精度频率乘法器或分频器。

一般地，这里会设置一个默认的 addend 值，比如设置为0x80000000，那么在每经过两个clk_ptp_ref clock 之后，32bit 的累加器会溢出；从图上可以看到如果累加器溢出，Sub-second 寄存器就会增加一个 incr_sub_sec 寄存器的配置单位，如果是100M时钟，这里会配置 MAC_Sub_Second_Increment 寄存器值应为2倍的时钟周期: 0x14；当然这里说的是 digital 模式，如果是 binary 模式，需要一个转化。digital 模式下，在增长到 0x3B9A_C9FF，对应的 Second 寄存器值增加1，表示时间已经累积了1s；如果是binary 模式，增长到 0x7FFF_FFFF ，Second 寄存器值增加1。

当进行 PTP1588 同步时候，在经过每个 clk_ptp_ref 时钟时，加数寄存器的值被添加到累加器，通过更改 addend 寄存器值，从而影响 Sub-second 的累加，可以简单认为修改了 ptp 时钟，最终影响了系统时间的变化；软件必须根据同步消息计算频率漂移，并相应地更新加数 addend 寄存器。

1.5 GMAC PTP1588 timestamp 时间戳获取

从 PTP1588 延时响应机制的报文收发流程来看，如何获得报文在网卡上发送和接收的精确时间至关重要；在硬件实现上，GMAC 在网络包发送出去和接收的同时，会往指定的内存里保存此刻发送和接收的系统时间戳数据，这个指定的内存就是其数据包对应的发送和接收描述符，比如下面的 tx 描述符：



代码段：

```
static inline void dwmac4_get_timestamp(void *desc, u32 ats, u64 *ts)
{
    struct dma_desc *p = (struct dma_desc *)desc;
    u64 ns;

    ns = le32_to_cpu(p->des0);
    /* convert high/sec time stamp value to nanosecond */
    ns += le32_to_cpu(p->des1) * 1000000000ULL;

    *ts = ns;
}
```

而何种报文在接收和发送的时候，硬件会保存下来时间戳，这个需要在 MAC_Timestamp_Control 寄存器中可以设置。

2. PTP1588 测试

通过 ethtool 检查是否支持硬件时间戳，对于硬件时间戳支持，参数列表应包括：

- SOF_TIMESTAMPING_RAW_HARDWARE
- SOF_TIMESTAMPING_TX_HARDWARE
- SOF_TIMESTAMPING_RX_HARDWARE

```
root@linaro-alip:/# ethtool -T eth0
Time stamping parameters for eth0:
Capabilities:
        hardware-transmit       (SOF_TIMESTAMPING_TX_HARDWARE)
```

```
        software-transmit       (SOF_TIMESTAMPING_TX_SOFTWARE)
        hardware-receive        (SOF_TIMESTAMPING_RX_HARDWARE)
        software-receive        (SOF_TIMESTAMPING_RX_SOFTWARE)
        software-system-clock   (SOF_TIMESTAMPING_SOFTWARE)
        hardware-raw-clock      (SOF_TIMESTAMPING_RAW_HARDWARE)
PTP Hardware Clock: 0
Hardware Transmit Timestamp Modes:
        off
        on
Hardware Receive Filter Modes:
        none
        all
        ptpv1-l4-event
        ptpv1-l4-sync
        ptpv1-l4-delay-req
        ptpv2-l4-event
        ptpv2-l4-sync
        ptpv2-l4-delay-req
        ptpv2-event
        ptpv2-sync
```

常用的测试软件有 Linuxptp 提供的 ptp4l，phy2sys，ptp4l 程序通过硬件时间戳将 PTP 硬件时钟与主时钟同步，或通过软件时间戳将系统时钟与主时钟同步；phc2sys 实现系统时钟与网卡硬件时钟之间的同步（仅限硬件时间戳模式）；另外其它测试软件还有 ptpd2等。

2.1 ptp4l 测试

Slave:

```
ptp4l -i eth0 -m -H -s
```

Master:

```
ptp4l -i eth0 -m -H
```

```
ptp4l[420.071]: selected /dev/ptp0 as PTP clock
ptp4l[420.077]: port 1 (eth0): new foreign master 54e1ad.fffe.dfa454-1
ptp4l[424.077]: selected best master clock 54e1ad.fffe.dfa454
ptp4l[424.078]: port 1 (eth0): LISTENING to UNCALIBRATED on RS_SLAVE
ptp4l[425.084]: master offset      -1466 s2 freq  -20216 path delay     13117
ptp4l[425.084]: port 1 (eth0): UNCALIBRATED to SLAVE on MASTER_CLOCK_SELECTED
ptp4l[426.084]: master offset         -6 s2 freq  -19196 path delay     13117
ptp4l[427.085]: master offset        491 s2 freq  -18701 path delay     13020
ptp4l[428.084]: master offset        330 s2 freq  -18714 path delay     13026
ptp4l[429.084]: master offset        365 s2 freq  -18580 path delay     13041
ptp4l[430.084]: master offset        116 s2 freq  -18720 path delay     13050
ptp4l[431.084]: master offset         30 s2 freq  -18771 path delay     13041
ptp4l[432.084]: master offset         24 s2 freq  -18768 path delay     13032
ptp4l[433.084]: master offset         29 s2 freq  -18756 path delay     13032
ptp4l[434.084]: master offset         39 s2 freq  -18737 path delay     13032
ptp4l[435.084]: master offset        -71 s2 freq  -18836 path delay     13032
ptp4l[437.085]: master offset        201 s2 freq  -18585 path delay     13015
ptp4l[438.085]: master offset        -70 s2 freq  -18796 path delay     13021
```

```
ptp4l[439.085]: master offset        -30 s2 freq  -18777 path delay     13041
ptp4l[440.085]: master offset         29 s2 freq  -18727 path delay     13032
ptp4l[441.085]: master offset        -41 s2 freq  -18788 path delay     13037
ptp4l[442.085]: master offset        -61 s2 freq  -18820 path delay     13037
ptp4l[443.085]: master offset         54 s2 freq  -18723 path delay     13007
ptp4l[444.085]: master offset        -31 s2 freq  -18792 path delay     13007
ptp4l[445.085]: master offset         -6 s2 freq  -18777 path delay     13007
ptp4l[446.085]: master offset         61 s2 freq  -18711 path delay     13045
ptp4l[447.085]: master offset        -89 s2 freq  -18843 path delay     13045
ptp4l[448.085]: master offset         41 s2 freq  -18740 path delay     13030
ptp4l[449.085]: master offset         -4 s2 freq  -18772 path delay     13030
ptp4l[450.085]: master offset        -47 s2 freq  -18817 path delay     13023
ptp4l[451.085]: master offset          5 s2 freq  -18779 path delay     13026
ptp4l[452.085]: master offset         55 s2 freq  -18727 path delay     13026
ptp4l[453.085]: master offset         93 s2 freq  -18673 path delay     13023
ptp4l[454.085]: master offset        -57 s2 freq  -18795 path delay     13023
ptp4l[455.086]: master offset        -32 s2 freq  -18787 path delay     13023
ptp4l[456.085]: master offset         33 s2 freq  -18732 path delay     13023
ptp4l[457.086]: master offset        -26 s2 freq  -18781 path delay     13037
ptp4l[458.086]: master offset        -55 s2 freq  -18817 path delay     13026
ptp4l[459.086]: master offset         -5 s2 freq  -18784 path delay     13026
ptp4l[460.086]: master offset        -33 s2 freq  -18813 path delay     13024
```

master offset表示主从端时间差，单位是ns。s0，s1，s2 : 表示1588的不同状态，s0表示未锁定，s1表示正在同步，s2表示锁定。freq表示时钟的频率调整（以十亿分率 (ppb) 为单位），path delay 值表示从主时钟发送的同步消息的预计延迟（以纳秒为单位），该延迟可通过E2E或P2P方式测量，默认为E2E。

2.2 同步系统时钟

将网卡上的时钟同步到操作系统：

- 主钟

```
sudo phc2sys -m -s CLOCK_REALTIME -c end0 -w &
sudo ptp4l -i end0 -m -H
```

- 从钟

```
phc2sys -m -s eth0 -c CLOCK_REALTIME -w  --step_threshold=1 &
ptp4l -i eth0 -m -H -s
```

- 同步前两块板子上的时间：

```
root@linaro-alip:/# date
2024年 07月 08日 星期一 19:48:25 CST

root@buildroot:/# date
Thu Jan  1 00:14:02 UTC 1970
```

- 同步后两块板子上的时间（中国北京时区CST 时间和零时区UTC时间正好相差8个时区）：

```
root@linaro-alip:/# date
2024年 07月 08日 星期一 19:51:52 CST

root@buildroot:/# date
Mon Jul  8 11:51:52 UTC 2024
```

2.3 gPTP 测试

gPTP 是 general precise time protocol 的简称，是 PTP 协议的派生，它又叫做 802.1AS，gPTP 的目的是确保所有局域网里的节点的时间完全同步。gPTP协议是基于L2层的传播，那么就决定了一个特性，只能在局域网里传播。

- Slave:

```
ptp4l -i eth0 -m -H -f ./gPTP.cfg --step_threshold=1 -s
```

- Master:

```
ptp4l -i end0 -m -H -f ./gPTP.cfg --step_threshold=1
```

```
ptp4l[196.636]: selected /dev/ptp0 as PTP clock -f gPTP.cfg --step_threshold=1 -
s
ptp4l[196.670]: port 1 (eth0): INITIALIZING to LISTENING on INIT_COMPLETE
ptp4l[196.671]: port 0 (/var/run/ptp4l): INITIALIZING to LISTENING on
INIT_COMPLETE
ptp4l[196.671]: port 0 (/var/run/ptp4lro): INITIALIZING to LISTENING on
INIT_COMPLETE

ptp4l[200.262]: port 1 (eth0): new foreign master 9ae370.fffe.1cb4e0-1
ptp4l[200.547]: selected local clock 568583.fffe.b70879 as best master
ptp4l[202.262]: selected best master clock 9ae370.fffe.1cb4e0
ptp4l[202.262]: port 1 (eth0): LISTENING to UNCALIBRATED on RS_SLAVE
ptp4l[202.942]: port 1 (eth0): UNCALIBRATED to SLAVE on MASTER_CLOCK_SELECTED
ptp4l[203.568]: rms 853264659523145600 max 1706529319046291712 freq  -4554 +/-
1736 delay 15780 +/-   0
ptp4l[204.570]: rms  305 max  360 freq  -5595 +/-  53 delay 15780 +/-   0
ptp4l[205.571]: rms  205 max  290 freq  -5713 +/-  37 delay 15770 +/-   0
ptp4l[206.573]: rms   53 max  129 freq  -5592 +/-  58 delay 15769 +/-   0
ptp4l[207.574]: rms   46 max   85 freq  -5572 +/-  63 delay 15765 +/-   0
ptp4l[208.576]: rms   62 max  115 freq  -5510 +/-  73 delay 15765 +/-   0
ptp4l[209.577]: rms   52 max   95 freq  -5502 +/-  66 delay 15765 +/-   0
ptp4l[210.579]: rms   49 max   75 freq  -5545 +/-  66 delay 15765 +/-   0
ptp4l[211.580]: rms   56 max  115 freq  -5506 +/-  75 delay 15765 +/-   0
ptp4l[212.582]: rms   42 max   65 freq  -5552 +/-  53 delay 15765 +/-   0
ptp4l[213.584]: rms   53 max  114 freq  -5553 +/-  72 delay 15774 +/-   0
ptp4l[214.585]: rms   51 max  106 freq  -5513 +/-  66 delay 15774 +/-   0
ptp4l[215.587]: rms   59 max   74 freq  -5552 +/-  81 delay 15774 +/-   0
ptp4l[216.588]: rms   57 max   96 freq  -5503 +/-  71 delay 15764 +/-   0
ptp4l[217.590]: rms   50 max   99 freq  -5566 +/-  62 delay 15779 +/-   0
ptp4l[218.591]: rms   68 max  101 freq  -5538 +/-  93 delay 15779 +/-   0
ptp4l[219.593]: rms   66 max  106 freq  -5538 +/-  91 delay 15774 +/-   0
ptp4l[220.595]: rms   56 max  106 freq  -5551 +/-  78 delay 15774 +/-   0
ptp4l[221.596]: rms   55 max   84 freq  -5575 +/-  72 delay 15784 +/-   0
```

```
ptp4l[222.598]: rms   36 max   71 freq  -5539 +/-  47 delay 15789 +/-   0
ptp4l[223.599]: rms   53 max   76 freq  -5561 +/-  74 delay 15784 +/-   0
ptp4l[224.601]: rms   65 max  116 freq  -5533 +/-  87 delay 15784 +/-   0
ptp4l[225.602]: rms   64 max  136 freq  -5577 +/-  86 delay 15784 +/-   0
ptp4l[226.604]: rms   66 max  136 freq  -5538 +/-  88 delay 15784 +/-   0
ptp4l[227.606]: rms   46 max   61 freq  -5553 +/-  63 delay 15779 +/-   0
ptp4l[228.607]: rms   58 max  139 freq  -5547 +/-  80 delay 15779 +/-   0
ptp4l[229.609]: rms   42 max   61 freq  -5553 +/-  58 delay 15779 +/-   0
```

这里使用的是均方根显示的 offset，更加平滑。

3. PPS

3.1 Fixed PPS

PPS: Pulse-Per-Second；
默认情况下，GMAC 处于固定每秒脉冲输出模式，指示 1 秒间隔。可以通过设置 MAC_PPS_Control 寄存器中的 PPSCTRL0 字段来更改 PPS 输出的频率。所以在这个模式下，只要配置好 iomux 就能测试到 PPS 脉冲输出信号。PPSCTRL 的默认值为 0000，PPS 输出为每秒 1 个脉冲，脉冲宽度为 一个 clk_ptp_ref 周期。

从上图可以看到的是，两台设备在经过 PTP1588 同步后，经过一段时间输出的 PPS 信号在示波器上看到的抖动余晖时间与 ptp4l 测试的 master offset 时间差结果是基本相符的。

3.2 binary mode VS digital mode

前面章节已经说到过，这两个模式下系统时间更新有差异，对于 PPS 输出，这两个模式下的频率与占空比是有差异：对于 PPSCTRL 的其他值，PPS 输出将成为以下频率的生成时钟：

- 0001：binary 翻转模式为 2 Hz，digital 翻转模式为 1 Hz。
- 0010：binary 翻转模式为 4 Hz，digital 翻转模式为 2 Hz。
- 0011：binary 翻转模式为 8 Hz，digital 翻转模式为 4 Hz。
- 0100：binary 翻转模式为 16 Hz，digital 翻转模式为 8 Hz。
- ...
- 1111：binary 翻转模式为 32.768 KHz，digital 翻转模式为 16.384 KHz。

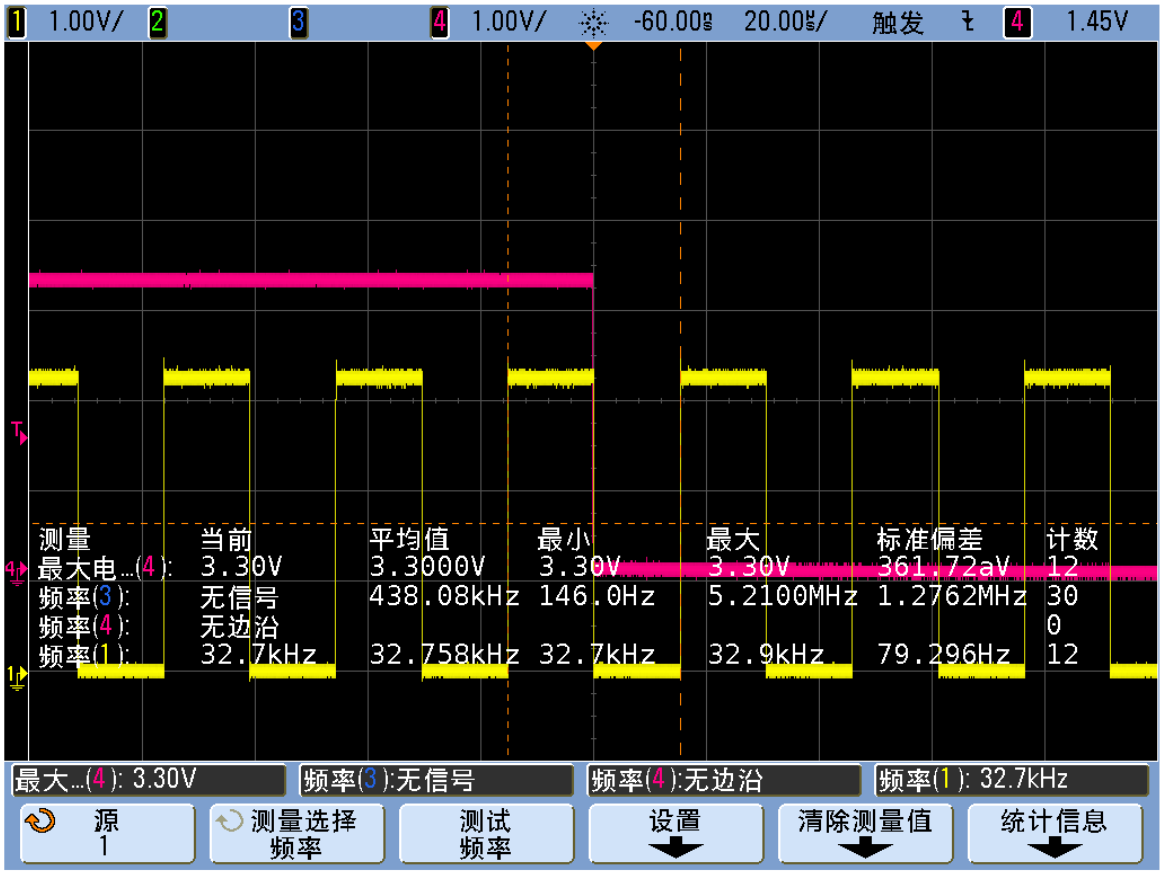
在 binary 翻转模式下，PPS 输出 (ptp_pps_o) 在这些频率下的占空比为 50%。

在 digital 翻转模式下，PPS 输出频率是一个平均数。实际时钟的频率不同，每秒都会同步。例如：

- 当 PPSCTRL = 0001 时，PPS (1 Hz) 的低周期为 537 毫秒，高周期为 463 毫秒
- 当 PPSCTRL = 0010 时，PPS (2 Hz) 是以下序列：一个占空比为 50% 且周期为 537 毫秒的时钟，第二个时钟周期为 463 毫秒（268 毫秒低，195 毫秒高）
- 当 PPSCTRL = 0011 时，PPS (4 Hz) 是以下序列：三个占空比为 50% 且周期为 268 毫秒的时钟，第四个时钟周期为 195 毫秒（134 毫秒低，61 毫秒高）

所以，digital 模式下，PPS 输出的周期并不是均匀的，如果需要均匀周期的输出，需要配置成 binary 模式。

需要注意的是，当 PPSCTRL > =1 时，PPS 信号这里是从低电平开始，也就是下降沿对齐：



3.3 Flex PPS

相对于 fixed PPS 在整秒输出脉冲，flex PPS 更加灵活，输出选项包括开始时间，占空比，频率等。

- 开启 pps:

```
echo "0 0 0 1 1" > /sys/class/ptp/ptp0/period
```

- 也可以设置 pps 开始时间，比如是 100s:

```
echo "0 100 0 1 1" > /sys/class/ptp/ptp0/period
```

- 两台设备时间同步后，开始跑 pps

  后台跑 ptp4l：

```
ptp4l -i eth0 -m -H &
ptp4l -i eth0 -m -H -s &
```

  约定同一时刻开启 pps，比如200s，两台机器分别输入：

```
echo "0 200 0 1 1" > /sys/class/ptp/ptp0/period
```

  将两台设备 pps 信号都输出到同一个示波器，在 GMAC 系统时间等于 200s 的时候，就会同时输出 PPS 信号，测得二者抖动时间是百 ns 级别，与 ptp4l 测试数据相符。

3.4 支持 PPS 的芯片

目前支持 PTP1588 的芯片中，除了 RK3568 和 RV1126 外其它都支持 PPS 功能。

4. RT-Thread 下的 PTP1588 功能

除了 Linux，在 RT-Thread 下也可以支持 PTP1588 功能，比如 RK2118 芯片用的是 RT-Thread。

5. 常见问题

5.1 tx timeout

对一些 版本的 ptp4l 测试出现错误timed out while polling for tx timestamp 时，可以修改 ptp4l.conf 的 tx_timestamp_timeout = 400，然后通过 -f 指定修改过的 ptp4l.conf 配置文件，也可以通过直接加命令来配置：

```
--tx_timestamp_timeout=400
```

5.2 4.19 内核不支持 master 模式

加入补丁：

```
diff --git a/drivers/net/ethernet/stmicro/stmmac/stmmac_main.c
b/drivers/net/ethernet/stmicro/stmmac/stmmac_main.c
index 0511062a9e0d..1e6078a17511 100644
--- a/drivers/net/ethernet/stmicro/stmmac/stmmac_main.c
+++ b/drivers/net/ethernet/stmicro/stmmac/stmmac_main.c
@@ -644,7 +644,8 @@ static int stmmac_hwtstamp_set(struct net_device *dev,
struct ifreq *ifr)
                        config.rx_filter = HWTSTAMP_FILTER_PTP_V2_EVENT;
                        ptp_v2 = PTP_TCR_TSVER2ENA;
                        snap_type_sel = PTP_TCR_SNAPTYPSEL_1;
-                       ts_event_en = PTP_TCR_TSEVNTENA;
+                       if (priv->synopsys_id < DWMAC_CORE_4_10)
+                               ts_event_en = PTP_TCR_TSEVNTENA;
                        ptp_over_ipv4_udp = PTP_TCR_TSIPV4ENA;
                        ptp_over_ipv6_udp = PTP_TCR_TSIPV6ENA;
                        ptp_over_ethernet = PTP_TCR_TSIPENA;
```