RTL9607C

Time Synchronous Application Note

(CONFIDENTIAL: Development Partners Only)

Rev. 1.0.0
02 Jun 2017

Realtek Semiconductor Corp.
No. 2, Innovation Road II, Hsinchu Science Park, Hsinchu 300, Taiwan
Tel.: +886-3-578-0211 Fax: +886-3-577-6047
www.realtek.com

COPYRIGHT

TRADEMARKS

Realtek is a trademark of Realtek Semiconductor Corporation. Other names mentioned in this document are trademarks/registered trademarks of their respective owners.

DISCLAIMER

Realtek provides this document “as is”, without warranty of any kind, neither expressed nor implied, including, but not limited to, the particular purpose. Realtek may make improvements and/or changes in this document or in the product described in this document at any time. This document could include technical inaccuracies or typographical errors.

USING THIS DOCUMENT

Though every effort has been made to assure that this document is current and accurate, more information may have become available subsequent to the production of this guide. In that event, please contact your Realtek representative for additional information that may help in the development process.

CONFIDENTIALITY

This document is confidential and should not be provided to a third-party without the permission of Realtek Semiconductor Corporation.

REVISION HISTORY

Revision	Release Date	Summary
1.0.0	2017/06/02	First Release

Table of Contents

1. OVERVIEW 1
2. GPON TOD 2
3. EPON TOD 2
4. PTP 4
4.1. DELAY MEASUREMENT 4
4.2. TIME SYNCHRONIZATION 5
4.3. PTP CONFIGURATION 7
5. 1PPS +TOD 9
5.1. PULSE AND DELAY ADJUST 10
5.2. HARDWARE DEFAULT TOD FRAME FORMAT 11
5.3. SOFTWARE DEFINED TOD FRAME 11
5.4. SOFTWARE AID REFERENCE POINT 12
6. SYNCE 12
7. API 13
7.1. GPON&EPON TOD 13
7.1.1. Initialization 13
7.1.2. Set ToD Time 13
7.1.3. Update System Time 14
7.2. PTP 15
7.2.1. Initialization 15
7.2.2. Enable PTP Per Port 15
7.2.3. Get System Time 15
7.2.4. Set PTP_LATCH_TS 16
7.2.5. Set Ingress PTP Action 16
7.2.6. Set Egress PTP Action 16
7.3. 1PPS+TOD 17
7.3.1. Initialization 17
7.3.2. Set ToD Delay Time 17
7.3.3. Set 1PPS Width of Pulse 18
7.3.4. Set Software & Hardware Mode 18
7.3.5. Set Software Aid Reference Point 19
7.3.6. Set Software Defined Frame Length 19
7.3.7. Set Software Defined Frame Value 20
7.3.8. Set Baudrate 22
8. EXAMPLE 23
8.1. GPON&EPON TOD 23
8.2. PTP MASTER 23
8.2.1. One-Step Timestamp by Software 23
8.2.2. One-Step Timestamp by Hardware 25
8.2.3. Two-Step Timestamp 26
8.2.4. Ingress Time of PTP Packet 28
8.3. 1PPS+TOD 29
9. TIME SYNCHRONOUS 30
9.1. PTP GRAND MASTER PROCESS 30

9.2. 1PPS ToD Module 33
9.3. Time Synchronous Process 34

List of Tables

TABLE 1. GPON TOD FORMAT 2
TABLE 2. OSSPDU FORMAT 3
TABLE 3. OSSPDU TRAP ENABLE 3
TABLE 4. PTP TIME FREQENCY 7
TABLE 5. PTP ENABLE 7
TABLE 6. PTP OFFSET 7
TABLE 7. PTP SYSTEM TIME 7
TABLE 8. PTP INGRESS AND EGRESS ACTION 7
TABLE 9. PTP MESSAGE CLASS 8
TABLE 10. PTP MEANPATH DELAY 8
TABLE 11. PTP BEHAVIOR 9
TABLE 12. PULSEWIDTH AND TODDELAY 10
TABLE 13. HARDWARE DEFAULT TOD FRAME 11

List of Figures

FIGURE 1. TIME SYNCHRONOUS OVERVIEW 1
FIGURE 2. FREQUENCY SYNCHRONOUS OVERVIEW 1
FIGURE 3. GPON TOD DECISION FLOW 2
FIGURE 4. EPON TOD OSSPDU FLOW 3
FIGURE 5. TWO-STEP DELAY MEASUREMENT 5
FIGURE 6. ONE-STEP DELAY MEASUREMENT 5
FIGURE 7. TIME SYNCHRONIZATION 6
FIGURE 8. PEER-TO-PEER TRANSPARENT 6
FIGURE 9. 1PPS+TOD OVERVIEW 10
FIGURE 10. PULSEWIDTH AND TODDELAY 11
FIGURE 11. SOFTWARE AID REFERENCE POINT OVERVIEW 12
FIGURE 12. GPON SYNCE BLOCK DIAGRAM 12
FIGURE 13. EPON SYNCE BLOCK DIAGRAM 13
FIGURE 14. ONE-STEP TIMESTAMP BY SOFTWARE 24
FIGURE 15. ONE-STEP TIMESTAMP BY HARDWARE 25
FIGURE 16. TWO-STEP TIMESTAMP 26
FIGURE 17. RX INGRESS TIME 28
FIGURE 18. PTP GRAND MASTER PROCESS 30
FIGURE 19. GM MAIN THREAD STATE MACHINE 31
FIGURE 20. GM MAIN THREAD WAIT EVENT 31
FIGURE 21. GM RX THREAD STATE MACHINE 32
FIGURE 22. PTP MASTER TEST WITH PTP SLAVE 32
FIGURE 23. 1PPS TOD STATE MACHINE 33
FIGURE 24. TIME SYNCHRONOUS ENVIRONMENT 34
FIGURE 25. TIME SYNCHRONOUS SOFTWARE FLOW 34

1. Overview

Time synchronization is critical to transition to Ethernet services. The current synchronous infrastructure and application require frequency, phase, and time of day synchronization. Time Synchronous includes PTP, 1PPS ToD, GPON ToD, EPON ToD and SyncE in this document. ONU synchronizes system time by GPON/EPON ToD. ONU can distribute time by 1PPS+ToD or IEEE1588 PTP. 1PPS+ToD transfers time by uart. IEEE1588 PTP transfers time by Ethernet.

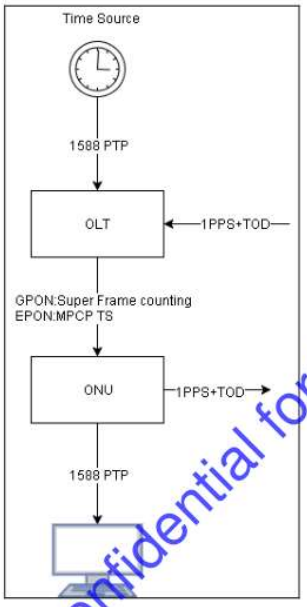

Figure 1. Time Synchronous Overview

ONU can synchronize the frequency by SyncE. OLT sends the signal to ONU. ONU can recover the frequency signal and decides using it or local crystal signal.

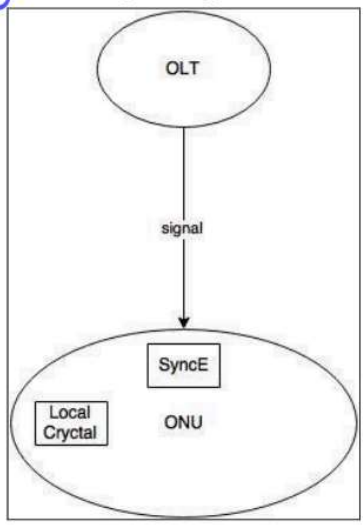

Figure 2. Frequency Synchronous Overview

2. GPON ToD

GPON ToD bases on ITU-T G.984.3 Amendment 2. OLT and ONU use OLT-G(131) Managed entities of OMCI for synchronizing time in GPON. OLT will tell ONU the time of day information before the super frame N arriving ONU. When ONU receives super frame N, ONU will update system time by the time of day information.

Time of Day information of OLT-G is 14 bytes. The first 4 bytes is the super frame counter. OLT decides which super frame to update ONU system time. The last 10 bytes is the same as packet format of timestamp of IEEE 1588. The format of 14 bytes is as the following.

Table 1. GPON ToD Format

13	12	11	10	9	8	7	6	5	4	3	2	1	0
SuperFrameCounter N				secondsField						nanosecondsField			

When ONU receives OLT-G OMCI from OLT, ONU will write SuperFrameCounter N into GPON MAC register m_pps_realign_sf.Then, ONU writes secondsField and nanosecondsField bit into PTP_PON_TOD_SEC and PTP_PON_TOD_NSEC(8 nanosecond unit) of PTP register. Then, GPON MAC reverses m_pps_realign_tgl. When GPON MAC receives GTC frame corresponding to superframeCounter N, it will send gpon_tod_serial_data signal to inform switch. Switch will update PTP_PON_TOD_SEC and PTP_PON_TOS_NSEC to PTP_SEC and PTP_NSEC.

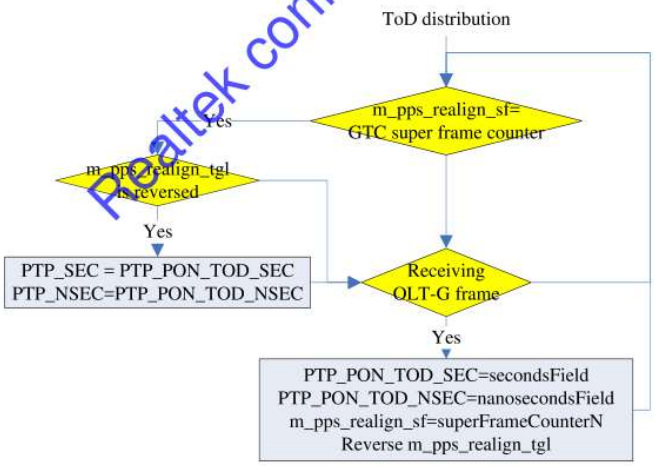

Figure 3. GPON ToD Decision Flow

3. EPON ToD

OLT and ONU use OSSP (organization-specific slow protocol) TIMESYNC Message defined in 802.1AS for synchronizing time in EPON.

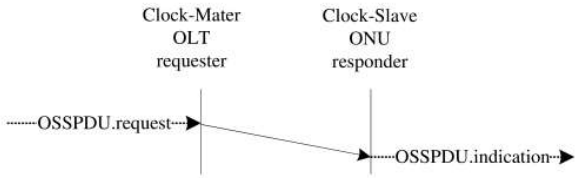

Figure 4. EPON ToD OSSPDU Flow

The ToD packet format of OSSPDU is as the following. OLT pre-calculates colock, delay, etc. before sending OSSPDU to ONU.

Table 2. OSSPDU Format

	Octets
Destination Address(01-80-C2-00-00-02)	6
Source Address	6
Length/Type = 88-09	2
Subtype = 0x0A	1
Organizationally Unique Identifier=00-80-C2	3
Message Identifier = 0x1	2
X(MPCP counter for $ToD_{X,i}$)	4
$ToD_{X,i}$(1588 timestamp)	10
sourcePortIdentity	10
logMessageInterval	1
rateRatio	8
gmTimeBaseIndicator	2
lastGmPhaseChange	12
scaledLastGmFreqChange	4
FCS	4

OSSP_TRAP_EN is for trapping OSSPDU packet to CPU.

Table 3. OSSPDU Trap Enable

Field	Description	Bits
OSSP_TRAP_EN	Trap OSSP packets 0b0: disable 0b1: enable trap OSSP packet to CPU	1

When ONU receives OSSPDU from OLT, CPU will set delay time for timing synchronization MPCP counter S to register EPON_TOS_COUNTER according to MPCP counter X of OSSPDU. ONU will use the delay $ToD_{x,i}$ formula (ref. IEEE 802.1AS 13.1.4 Time Synchronization in EPON) to set

PTP_PON_TOD_SEC and PTP_PON_TOD_NSEC. After that, ONU sets EPON_TOD_SYNC=1. EPON MAC will use timing signal to inform switch when MPCP counter is reaching to EPON_TOD_COUNTER. When switch receives time sync signal, it will use PTP_PON_TOD_SEC and PTP_PON_TOD_NSEC to update PTP_SEC and PTP_NSEC

4. PTP

Precision Time Protocol (PTP) is a protocol used to synchronize clocks by network. On a local area network, it achieves clock accuracy in nanosecond range, making it suitable for measurement and control systems. PTP protocol use Delay Measurement and Time Synchronization for synchronizing time.

4.1. Delay Measurement

Delay Measurement measures propagation delay between two machines. In the two step mode, the Delay Measurement will use the procedures to measure propagation delay.

- Delay Requestor will send “Delay Request (Pdelay_req)” and store egress time T1.
- Delay Responder receives “Delay Request” and store ingress time T2. Then, Delay Responder puts the T2 in “Delay Response (Pdelay_resp)” and sends “Delay Response” to Delay Requestor. Delay Responder stores egress time T3 of “Delay response”. Delay Responsder puts the T3 in the “Delay Response Follow_Up(Pdelay_Resp_Follow_Up)” and sends to Delay Requestor.
- When Delay Requestor receives “Delay Response”, it will store ingress time T4 and store T2 in the “Delay Response”
- When Delay Requestor receives “Delay Response Follow_Up”, it will store T3 in the “Delay Response Follow_Up”. In this moment, Delay Requestor can calculate the delay time Td = ((T4 – T1) – (T3 – T2))/2. Td

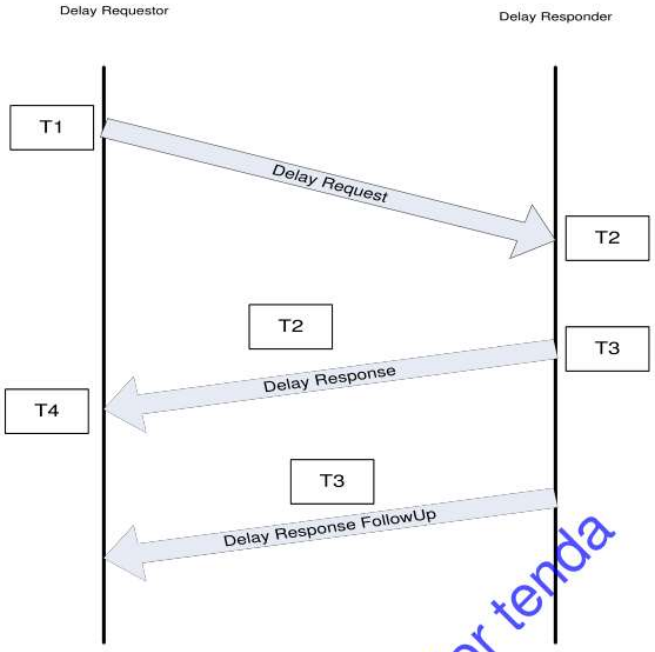

Figure 5. Two-Step Delay Measurement

When measuring the propagation delay in one step mode, the delay of calculation formula is the same as above mention.

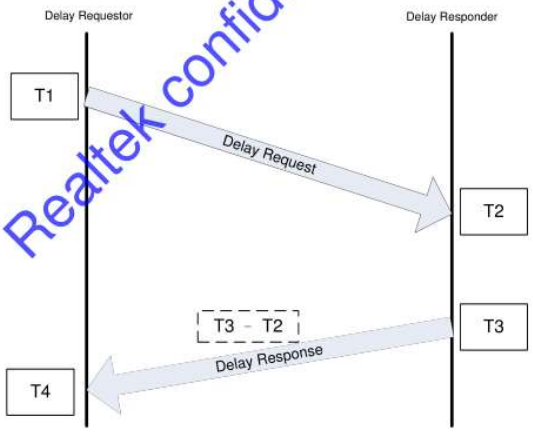

Figure 6. One-Step Delay Measurement

4.2. Time Synchronization

Time Synchronization will let master and slave synchronizing time through Sync message.

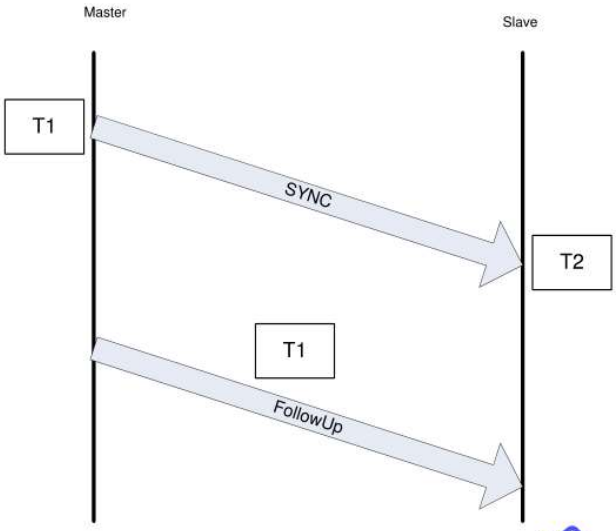

Figure 7. Time Synchronization

- Master sends “Sync” and stores the egress time T1. Master puts T1 in the “FollowUp” and sends it out.
- When Slave receives “Sync”, it will store the ingress time T2.
- When Slave receives “FollowUp”, it will get T1. In this moment, Slave can calculate the offset between Master and Slave. Toffset = T2 – T1 – Td.

If Device type is peer-to-peer transparent, the transparent flow is like the following figure. meanPathDelay is pre-calculated by propagation delay measurement

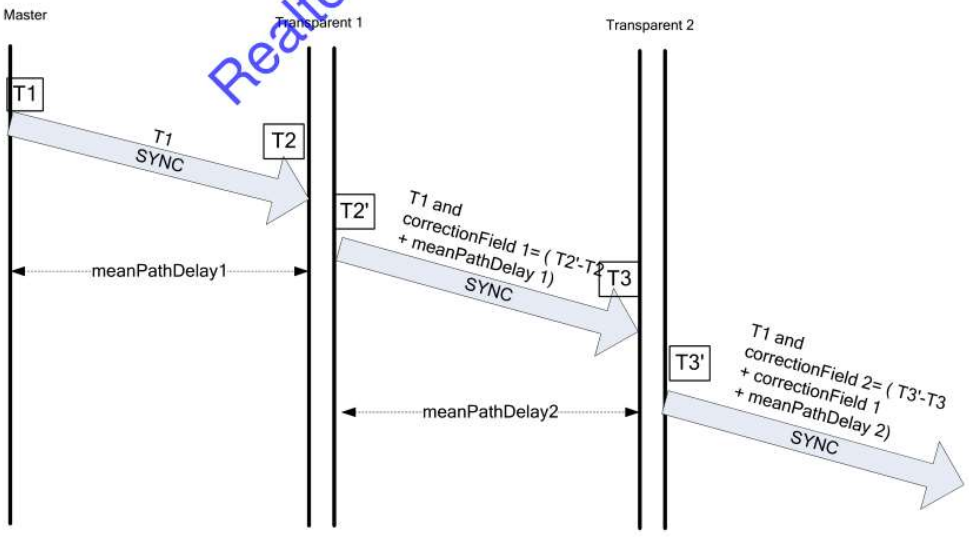

Figure 8. Peer-to-peer Transparent

4.3. PTP Configuration

PTP_TIME_FREQ is for adjusting frequency.

Table 4. PTP Time Freqency

Field Name	Description	Bits
PTP_TIME_FREQ	Configure frequency of system time(clock drift granularity about 0.008ppm)	28

PTP_EN is for awaring ingress MAC and egress MAC.

Table 5. PTP Enable

Field Name	Description	Bits
PTP_EN	Per-port enable PTP message aware function	1

PTP_OFFSET_SEC and PTP_OFFSET_NSEC is for adjusting time. If the system time of device is later, putting the time gap to the PTP_OFFSET_SEC and PTP_OFFSET_NSEC. It the system time of device is earlier, putting the (Maximum value – time gap value + 1) to the offset register.

Table 6. PTP Offset

Field Name	Description	Bits
PTP_OFFSET_SEC	Configure offset of system time in second	48
PTP_OFFSET_NSEC	Configure offset of system time in 8 nanosecond	26
PTP_TIME_CTRL_CMD	Enable system time tuning with system offset timer, write to clear	1

PTP_LATCH_TS is for hardware latching the system time to PTP_SEC and PTP_NSEC.

Table 7. PTP System Time

Field Name	Description	Bits
PTP_SEC	System time in second	48
PTP_NSEC	System time in 8 nanosecond	26
PTP_LATCH_TS	Software configurable ingress timestamp of PTP message(unit 16ns)	3+26

PTP_IGR_MSGx_ACT is for deciding ingress action of PTP message. PTP_EGR_MSGx_ACT is for deciding egress action of PTP message. TRANSPARENT_PMSK is for ingress action “transparent active port mask”.

Table 8. PTP Ingress and Egress Action

Field Name	Description	Bits
PTP_IGR_MSGx_ACT	Action for ingress PTP message class x(0-9) 0b00: none	2

	0b01: trap to CPU with PTP timestamp CPU-tag 0b10: Forward to transparent active port mask 0b11: Forward to transparent active port mask and Rx mirror to CPU with PTP timestamp CPU-tag	
PTP_EGR_MSGx_ACT	Action for egress PTP message class x(0-9) 0b00: none 0b01: Latch egress timestamp and fill to packet 0b10: Tx mirror to CPU with PTP timestamp CPU-tag(without original packet content) 0b11: Modify correctionField	2
TRANSPARENT_PMSK	Transparent active port mask for PTP messages can be forwarded	7

Message Class is referenced by PTP_IGR_MSGx_ACT and PTP_EGR_MSGx_ACT.

Table 9. PTP Message Class

Message Class	Description
0x0	Sync message for two-step
0x1	Delay_Req message
0x2	Pdelay_Req message for two-step
0x3	Pdelay_Resp message
0x4	Follow_Up
0x5	Delay_Resp
0x6	Pdelay_Resp_Follow_Up message
0x7	Announce/Signaling and other PTP message
0x8	Sync message for one-step
0x9	Pdelay_Req message for one-step

PTP_MEANPATH_DELAY will be added to correction field when egress action is setting to “Modify correctionField”.

Table 10. PTP Meanpath Delay

Field Name	Description	Bits
PTP_MEANPATH_DELAY	Mean path delay time as measured by peer delay mechanism for the link connected to the ingress port,, mapping to correctionField[46:16]	31

When setting device to different PTP device type, the behavior of different PTP message type should follow the following table.

Table 11. PTP Behavior

Device Type		Ordinary/Boundary		Transparent	
Message Type		Two-Step	One-Step	Two-Step	One-Step
Ingress	Sync	Trap to CPU		STM = 0 : Forward to transparent active port mask STM=1: Forward to transparent active port mask and send to CPU port with PTP timestamp	
	Delay_Req			End-to-end:Forward to transparent active port mask Peer-to-peer: Drop	
	Delay_Resp				
	Pdelay_Req			End-to-end: Forward to transparent active port mask Peer-to-peer: Trap to CPU	
	Pdelay_Resp				
	Pdelay_Resp_Follow_Up				
	Follow_Up			Forward to transparent active port mask	
	Others				
Egress	Sync	Latch egress timestamp / forward timestamp CPU	Modify correctionField	Modify correctionField	
	Delay_Req		Latch egress timestamp/ forward timestamp CPU		
	Pdelay_Req			None	Modify correctionField
	Pdelay_Resp		Modify correctionField		
	Follow_Up	None		Modify correctionField	
	Delay_Resp				
	Pdelay_Resp_Follow_Up			End-to-end: Modify correctionField Peer-to-peer: None	
	others			None	

5. 1PPS +ToD

A Pulse Per Second (PPS or 1PPS) is an electrical signal that has a width of less than one second and a sharply rising or abruptly falling edge that accurately repeats once per second. Time of Day (ToD) is current time and sent in the corresponding 1PPS.

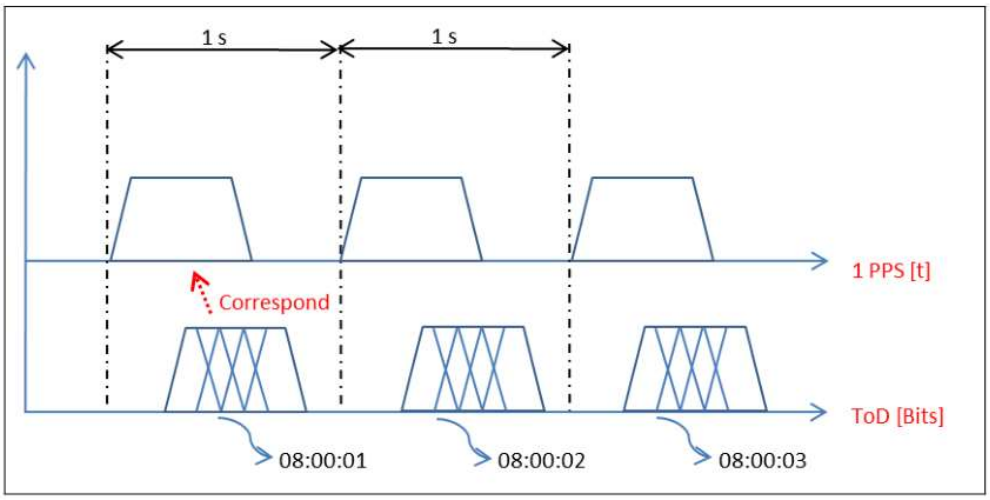

Figure 9. 1PPS+ToD overview

5.1. Pulse and Delay Adjust

ONU can use PulseWidth to adjust the pulse width and ToDDelay to adjust delay time of tod sending.

Table 12. PulseWidth and ToDDelay

Bits	Field	Description	Type	Default	Note
[5:0]	PulseWidth	Delay 10 ms each step (default pulse width: 100 ms)		0xA	Adjustable pulse width from 0s~630ms
[5:0]	ToDDelay	Delay 1 ms each step		0xA	0~63ms

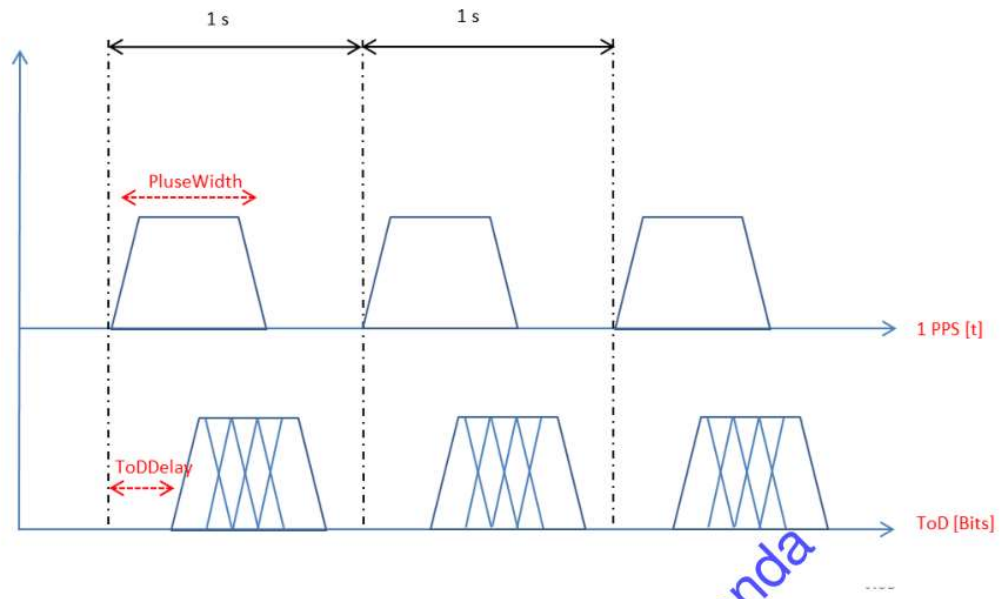

Figure 10. PulseWidth and ToDDelay

5.2. Hardware Default ToD Frame Format

The default format of ToD frame bases on format of China Mobile as the following.

Table 13. Hardware Default ToD Frame

1byte	1byte	1byte	1byte	2bytes	Nbytes	1byte
SYNC CHAR1	SYNC CHAR2	CLASS	ID	LENGTH Big Endian	Payload Big Endian	FCS

ToD header is composed of SYNC CHAR 1 and SYNC CHAR 2. SYNC CHAR 1 and SYNC CHAR 2 are fixed value (0x43 and 0x4D). Class is the basic classification of ToD messages. ID is the identic number of ToD message. Length is only the length of payload. Payload is composed of ToD message and some information. FCS is crc8 calculated by formula G(x) = x8 + x5 + x4 + 1.

5.3. Software Defined ToD Frame

Considering the future enhancement of ToD frame format, chip defines 32 byte registers (SW_ToD_Frame_0~SW_ToD_Frame_31) for ToD frame software fills in. Register Tod_Frame_Manual is for selecting ToD frame of hardware or software. Register Packet_length is the total length of software defined frame sending by ASIC.

5.4. *Software Aid Reference Point*

Software Aid Reference Point is preventing from ASIC taking too much processing time to calculate GPS time of week(TOW) transferred by UTC seconds. The reference point should be set by recent time.

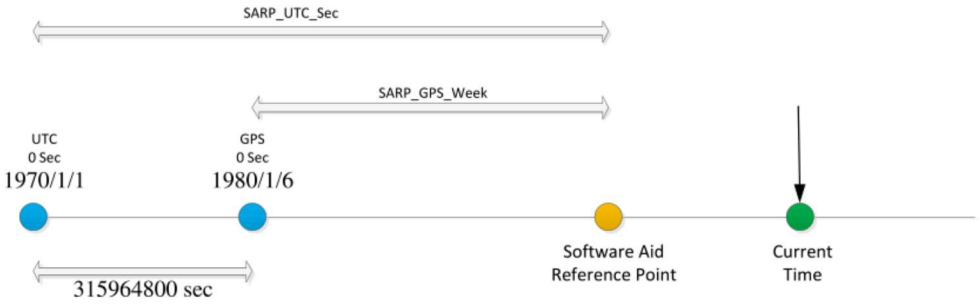

Figure 11. Software Aid Reference Point overview

The ToW translating formula is TOWSecond = UTCSecond – (WEEK*604800+315964800).

6. SyncE

Synchronous Ethernet (SyncE) is to provide a synchronization signal to those network resources that may eventually require such a type of signal. The Synchronous Ethernet signal is transmitted over physical layer. ONU receives signal from OLT by serdes and recovers frequency (CKREF_GPHY_CDR). CKREF_GPHY_CDR can supply frequency to TX module and PHY. The source of PHY can be selected by ckgphy_sel to decide using recover clock from RX or local XTAL.

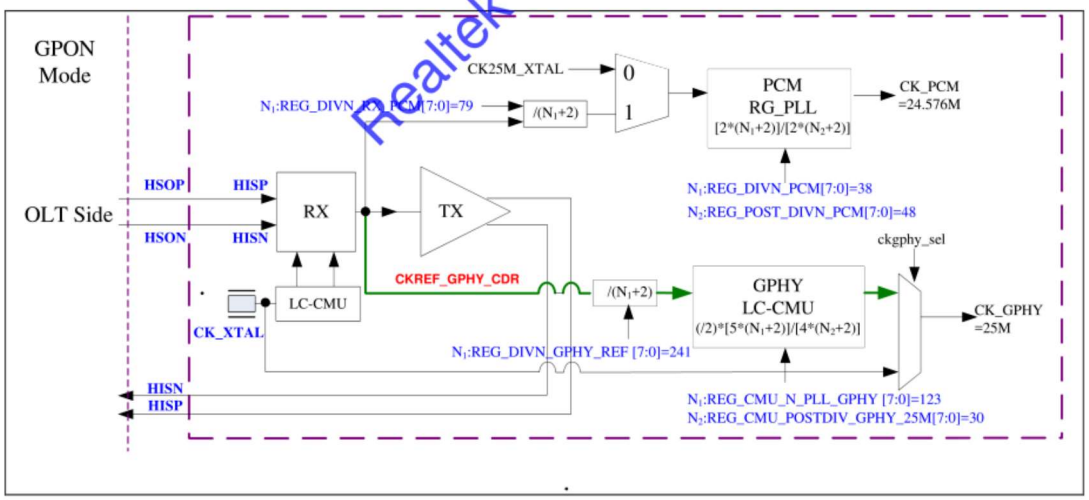

Figure 12. GPON SyncE Block Diagram

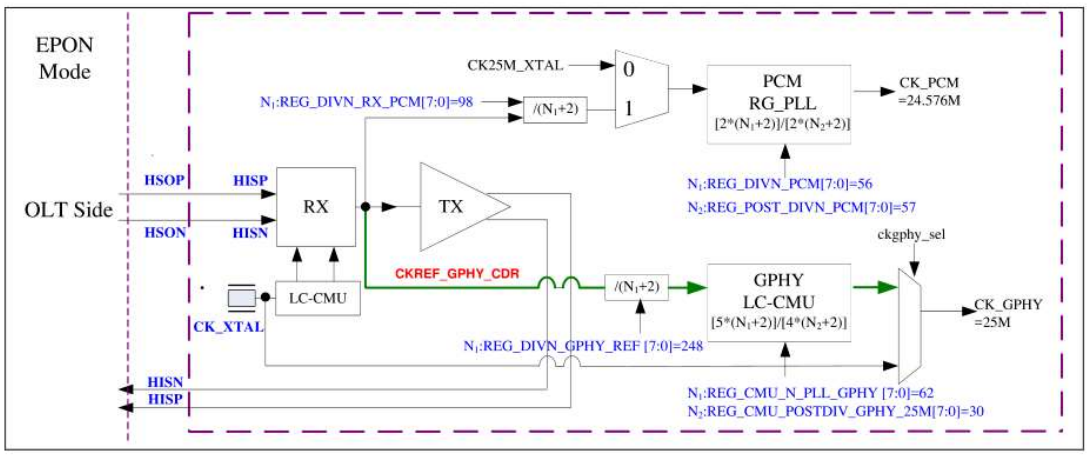

Figure 13. EPON SyncE Block Diagram

7. API

7.1. GPON&EPON ToD

7.1.1. Initialization

rtk_time_init is the first API user should invoke before setup any configuration.

7.1.2. Set ToD Time

rtk_time_ponTodTime_set is for setting time from OLT OLT-G or OSSPDU, superframe count(GPON) and local time count(EPON).

```
/* Example of Setting ToD Time
GPON set super frame to 100
EPON set local time to 10
Time set 10 second and 500 nanosecond
*/
rtk_pon_tod_t ponTod;
if(PON==GPON)
{
    ponTod.ponMode = 1;//gpon
```

```
        ponTod.startPoint.superFrame = 100;
}
else if(PON==EPON)
{
        ponTod.ponMode = 0;//epon
        ponTod.startPoint.localTime = 10;
}
ponTod.timeStamp.sec = 10;
ponTod.timeStamp.nsec = 500;
 if((ret = rtk_time_ponTodTime_set (PTP_MSG_TYPE_SYNC_ONE_STEP,
 PTP_EGR_ACTION_MODIFY_CORRECTION)) != RT_ERR_OK)
 {
        return ret;
 }
```


7.1.3. Update System Time

rtk_time_todEnable_set is for triggering ASIC updating system when superframe(GPON) is coming or the delay time(EPON) is reaching.

```
/* Example of Enabling Updating System Time
*/
 if((ret = rtk_time_todEnable_set (ENABLED)) != RT_ERR_OK)
 {
        return ret;
 }
```

7.2. PTP

7.2.1. Initialization

rtk_time_init is the first API user should invoke before setup any configuration.
Enable

7.2.2. Enable PTP Per Port

rtk_time_portPtpEnable_get is per-port enabling PTP message awaring function

7.2.3. Get System Time

rtk_time_curTime_latch is latching the system time. ***rtk_time_curTime_get*** is getting the system time latched by ***rtk_time_curTime_latch***. CPU can use system time to put the system time in timestamp of sync packet.

```
/* Example of one-step timestamp
CPU gets the system time from ASIC
*/
 if((ret = rtk_time_curTime_latch ()) != RT_ERR_OK)
 {
        return ret;
 }
rtk_time_timeStamp_t timeStamp;
if((ret = rtk_time_curTime_get(&timeStamp)) != RT_ERR_OK)
 {
        return ret;
 }
```

7.2.4. Set PTP_LATCH_TS

rtk_time_rxTime_set is setting the PTP_LATCH_TS. After setting PTP_LATCH_TS, CPU sends the PTP packet to ASIC. ASIC modified the correction field of PTP packet with existing time of PTP packet in ASIC.

```
/* Example of one-step timestamp
CPU sets the system time to register PTP_LATCH_TS
*/
if((ret = rtk_time_rxTime_set(timeStamp)) != RT_ERR_OK)
 {
        return ret;
 }
```


7.2.5. Set Ingress PTP Action

rtk_time_ptpIgrMsgAction_set is setting the ingress PTP packet action.

```
/* Example of one-step timestamp
CPU sets register PTP_IGR_MSGx_ACT to 0b01(trap to CPU with PTP timestamp CPU-tag). After setting
the PTP_IGR_MSGx_ACT of delay_req. The ASIC will trap delay_req to CPU with the ingress time.
*/
 if((ret = rtk_time_ptpIgrMsgAction_set (PTP_MSG_TYPE_DELAY_REQ,
 PTP_IGR_ACTION_TRAP2CPU)) != RT_ERR_OK)
 {
        return ret;
 }
```

7.2.6. Set Egress PTP Action

rtk_time_ptpEgrMsgAction_set is setting the egress PTP packet action.

```
/* Example of one-step timestamp
CPU sets register PTP_EGR_MSGx_ACT to 0b01 (Modify correctionField)
*/
if((ret = rtk_time_ptpEgrMsgAction_set (PTP_MSG_TYPE_SYNC_ONE_STEP,
PTP_EGR_ACTION_MODIFY_CORRECTION)) != RT_ERR_OK)
{
    return ret;
}
```

```
/* Example of two-step timestamp
CPU sets register PTP_EGR_MSGx_ACT to 0b10 (Tx mirror to CPU with PTP timestamp CPU-tag). After
setting PTP_EGR_MSGx_ACT of sync packet to 0b10, CPU sends sync packet and the ASIC will send-back
the mirroring sync packet with egress time of sync packet to CPU.
*/
if((ret = rtk_time_ptpEgrMsgAction_set (PTP_MSG_TYPE_SYNC,
PTP_EGR_ACTION_LATCH_TIME_AND_MIRROR2CPU)) != RT_ERR_OK)
{
    return ret;
}
```


7.3. 1PPS+ToD

7.3.1. Initialization

rtk_ppstod_init is the first API user should invoke before setup any configuration.

7.3.2. Set ToD Delay Time

rtk_ppstod_delay_set is setting delay time of ToD starting sending ToD frame.

```
/* Example
Set the delay to 10ms
*/
 if((ret = rtk_ppstod_delay_set(10)) != RT_ERR_OK)
 {
        return ret;
 }
```

7.3.3. Set 1PPS Width of Pulse

rtk_ppstod_pulseWidth_set is setting the width of pulse for 1PPS. Unit of parameter is 10ms.

```
/* Example
Set the delay to 100ms
*/
 if((ret = rtk_ppstod_pulseWidth_set(10)) != RT_ERR_OK)
 {
        return ret;
 }
```


7.3.4. Set Software & Hardware Mode

rtk_ppstod_mode_set is setting using Hardware default ToD frame or Software defined ToD frame to be the frame format of ToD.

```
/* Example
Set 1PPS ToD mode to software
*/
 if((ret = rtk_ppstod_mode_set (PPSTOD_MANUAL_MODE_SW)) != RT_ERR_OK)
 {
```

```
    return ret;
}
```

7.3.5. Set Software Aid Reference Point

rtk_ppstod_sarpGpsWeek_set is setting the Software Aid Reference Point by difference of week form GPS starting time. It will calculate UTC seconds automatically and set UTC seconds to register SARP_UTC_SEC.

```
/* Example
Set Software AiS Reference Point to March 14,2016
*/
 if((ret = rtk_ppstod_sarpGpsWeek_set (0x760)) != RT_ERR_OK)
 {
     return ret;
 }
```


7.3.6. Set Software Defined Frame Length

rtk_ppstod_frameLen_set is setting the frame length. The length is for ASIC sending length of software defined ToD frame.

```
/* Example
Set length of software defined ToD frame to 23
*/
 if((ret = rtk_ppstod_frameLen_set (0x17)) != RT_ERR_OK)
 {
     return ret;
 }
```

7.3.7. Set Software Defined Frame Value

rtk_ppstod_frameData_set is setting the data of software defined ToD frame. Total length of software defined ToD frame is 32 bytes.

```
/* Example
offset      value
 0          0x43
 1          0x4D
 2          0x20
 3          0x01
 4          0x00
 5          0x10
 6          0x00
 …           …
 15         0x00
 16         0x11
 17         0x04
 18         0xFF
 19         0x00
 …           …
 31         0x00
*/
 if((ret = rtk_ppstod_frameData_set (0,0x43)) != RT_ERR_OK)
 {
      return ret;
 }
 if((ret = rtk_ppstod_frameData_set (1,0x4D)) != RT_ERR_OK)
 {
      return ret;
```

```
}
if((ret = rtk_ppstod_frameData_set (2,0x20)) != RT_ERR_OK)
{
    return ret;
}
if((ret = rtk_ppstod_frameData_set (3,0x01)) != RT_ERR_OK)
{
    return ret;
}
if((ret = rtk_ppstod_frameData_set (4,0x00)) != RT_ERR_OK)
{
    return ret;
}
if((ret = rtk_ppstod_frameData_set (5,0x10)) != RT_ERR_OK)
{
    return ret;
}
for(i=6;i<=15;i++)
{
    if((ret = rtk_ppstod_frameData_set (i,0x00)) != RT_ERR_OK)
    {
        return ret;
    }
}
if((ret = rtk_ppstod_frameData_set (16,0x11)) != RT_ERR_OK)
{
    return ret;
}
```

```
if((ret = rtk_ppstod_frameData_set (17,0x04)) != RT_ERR_OK)
{
    return ret;
}
if((ret = rtk_ppstod_frameData_set (18,0xFF)) != RT_ERR_OK)
{
    return ret;
}
for(i=19;i<=31;i++)
{
    if((ret = rtk_ppstod_frameData_set (i,0x00)) != RT_ERR_OK)
    {
        return ret;
    }
}
```


7.3.8. Set Baudrate

rtk_ppstod_baudrate_set is setting the baudrate of uart for sending ToD frame.

```
/* Example
Set baudrate to 9600
*/
 if((ret = rtk_ppstod_baudrate_set (9600)) != RT_ERR_OK)
{
    return ret;
}
```

8. Example

8.1. GPON&EPON ToD

We can set GPON or EPON ToD Time by the following command. For GPON, parameter of startpoint represents superframe count. For EPON, parameter of startpoint represents delay time. Parameter of state is triggering ASIC updating system when superframe counter (GPON) or the local time counter(EPON) is reaching.

We present example of setting EPON ToD to update system time by Diagshell command.

```
/* Example
*/
RTK.0> time set ponmode epon startpoint 1000 pon-tod-time 100 10 state enable
RTK.0> epon get local-time
Local time:0x00000004
/*Waiting the local time is reaching 1000*/
RTK.0> time latch cur-time
RTK.0> time get cur-time
/*Checking system time */
```


8.2. PTP Master

8.2.1. One-Step Timestamp by Software

One-step IEEE1588 is only sending sync packet with egress time in timestamp field. ASIC supports getting system time and modifies correction field. CPU needs to get system time and attaches it to PTP event packet. CPU sets the PTP_LATCH_TS to system time. After that, CPU sends the PTP event packet. ASIC will add the {EGRESS time- PTP_LATCH_TS } to correction field of PTP event packet.

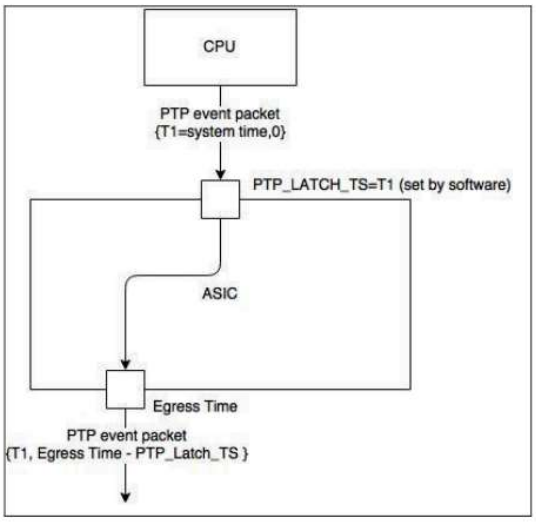

Figure 14. One-Step Timestamp by Software

We present example of sending one sync packet as the following by Diagshell command..

```
/* Example
*/
RTK.0> time init
RTK.0> time set ptp 0-3 state enable
RTK.0> time set egress type sync-one-step action modify-correction
RTK.0> debug packet tx clear
RTK.0> time latch cur-time
RTK.0> time get cur-time
Current Time: 1895 sec      847461456 nanosec
RTK.0> time set rxtime 1895 847461456
RTK.0> debug packet tx set padding 0 0 60
RTK.0> debug packet tx set addr 01:1b:19:00:00:00 00:11:22:33:44:55
RTK.0> debug packet tx set l2payload 0
88f70002002c0000000000000000000000000000000000001122ffff3344550002043d0000000000000000000000000000
RTK.0> debug packet tx send
```

8.2.2. One-Step Timestamp by Hardware

ASIC supports attaching timestamp by Hardware.

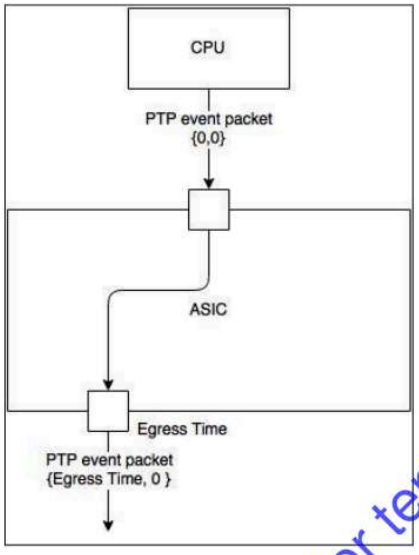

Figure 15. One-Step Timestamp by Hardware

We present example of sending one sync packet as the following by Diagshell command..

```
/* Example
*/
 RTK.0> time init
 RTK.0> time set ptp 0-3 state enable
 RTK.0> time set egress type sync-one-step action latch-time
 RTK.0> debug packet tx clear
 RTK.0> debug packet tx set padding 0 0 60
 RTK.0> debug packet tx set addr 01:1b:19:00:00:00 00:11:22:33:44:55
 RTK.0> debug packet tx set l2payload 0
88f70002002c00000000000000000000000000000000001122ffff3344550002043d00000000000000000000000000
 RTK.0> debug packet tx send
```

8.2.3. Two-Step Timestamp

Two-step IEEE1588 is only sending sync packet and latches the egress time of sync packet. Then, follow_up packet will be sent after the sync packet. Follow_up packet will attach the egress time of sync packet in timestamp field. ASIC supports mirroring the PTP event packet and sends back the mirroring PTP event packet with egress time to CPU. ASIC supports setting the PTP_LATCH_TS and CPU sends the PTP event packet. ASIC will add the {EGRESS time- PTP_LATCH_TS } to correction field of PTP event packet.

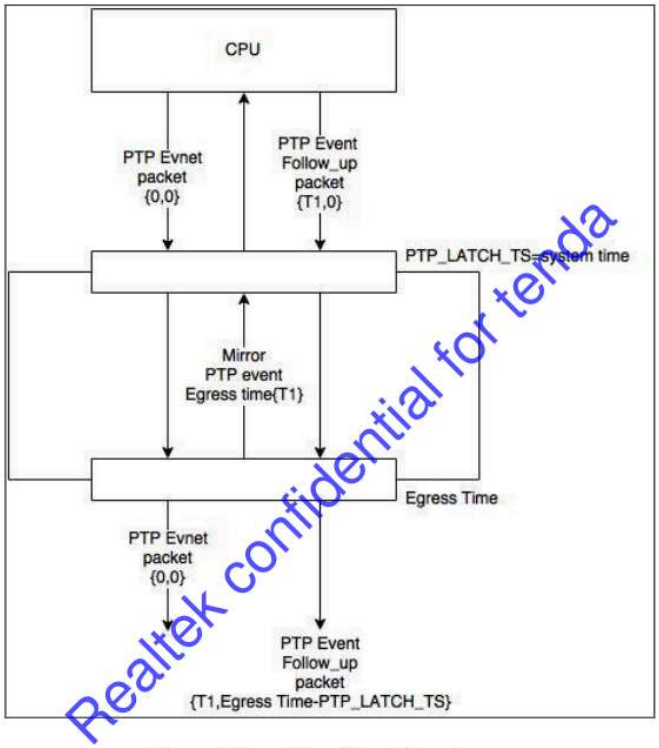

Figure 16. Two-Step Timestamp

We present example of sending sync packet and follow_up packet as the following by Diagshell command.

```
/* Example
*/
 RTK.0> time init
 RTK.0> time set ptp 0-3 state enable
 RTK.0> time set egress type sync action latch-time-and-mirror-to-cpu
 RTK.0> time set egress type follow-up action modify-correction
 RTK.0> debug packet rx enable
```

```
RTK.0> debug packet tx clear

RTK.0> debug packet tx set padding 0 0 60

RTK.0> debug packet tx set addr 01:1b:19:00:00:00 00:11:22:33:44:55

RTK.0> debug packet tx set l2payload 0
88f70002002c00000200000000000000000000000000001122ffff3344550002043d000000000000000000000000

RTK.0> debug packet tx send

RTK.0> debug packet rx dump

Packet dump: enabled

Actual packet length 0x34

Maximum dump length 0x80

Descriptor:

    opts1      addr      opts2      opts3

0x3601a038 0xa0b0aedc 0x80000000 0x001e6020

own eor fs ls crc l3csf l4csf rcdf frag ppptag rwt pkttype rout orifmt pctrl len

  0   0  1  1   0     1     1    0    0     0  0       0    1       1     0  56

cputag ptp_in_cpu_tag_exist svlan_tag_exist pon_stream_id ctagva cvlan_tag

     1                  0                0            0      0        0

src_port_num dst_port_mask reason internal_pri ext_port_ttl

           0              0  243           0            1

       00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f

0000 ff ff ff ff ff ff 00 11 22 33 44 55 00 00 03 ea

0001 1f ea 31 48 00 00 00 00 00 00 00 00 00 00 00 00

0002 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0003 00 00 00 00

RTK.0>debug packet tx set l2payload 0
88f70802002c00000000000000000000000000000000001122ffff3344550002043d00000000000003ea1fea3148

RTK.0> debug packet tx send
```

8.2.4. Ingress Time of PTP Packet

ASIC supports latching ingress time of receiving PTP packet and trapping PTP packet to CPU with the ingress time.

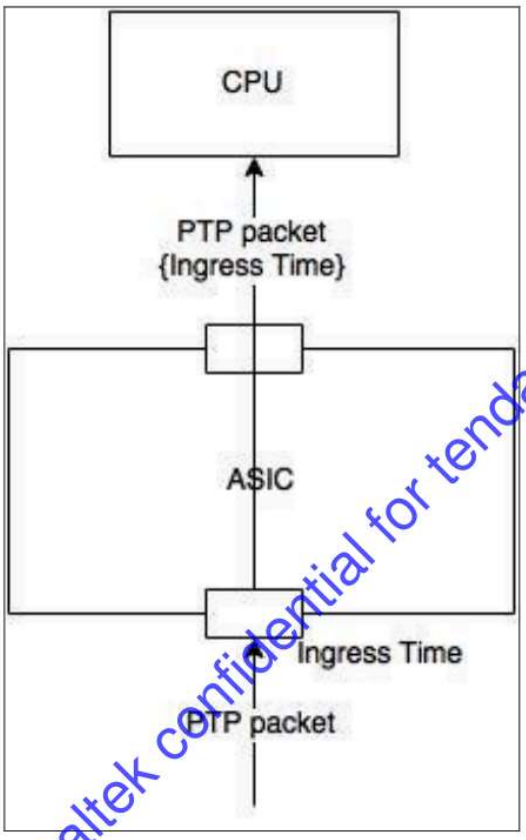

Figure 17. RX Ingress Time

We present example of receiving delay_req packet by Diagshell command..

```
/* Example
*/
RTK.0> time init
RTK.0> time set ptp 0-3 state enable
RTK.0> time set ingress type delay-req action trap-to-cpu
RTK.0> debug packet rx enable
RTK.0> debug packet rx dump
Packet dump: enabled
```

```
Actual packet length 0x4c

Maximum dump length 0x80

Descriptor:

     opts1         addr        opts2       opts3

0x36000050 0xa0ac4ccc 0xc3d00000 0x001e0000

own eor fs ls crc l3csf l4csf rcdf frag ppptag rwt pkttype rout orifmt pctrl len

   0     0   1   1     0        1        1      0      0         0     0          0      0         0        0    80

cputag ptp_in_cpu_tag_exist svlan_tag_exist pon_stream_id ctagva cvlan_tag

         1                              1                    0                61          0             0

src_port_num dst_port_mask reason internal_pri ext_port_ttl

                  0                    0       240                  0                  0

         00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f

0000 00 11 22 33 44 55 00 00 00 00 00 01 88 99 04 f0

0001 01 e8 00 00 00 00 1f 26 24 11 6d b8 88 f7 01 02

0002 00 2c 00 00 04 00 00 00 00 00 00 00 00 00 00 00

0003 00 00 00 80 63 ff ff 00 09 ba 00 02 04 3d 00 00

0004 00 00 45 b1 11 49 2e 32 42 63 00 00
```


8.3. 1PPS+ToD

We present example of setting software mode to be ToD frame format by Diagshell command..

```
/* Example

*/

RTK.0> ppstod init

RTK.0> ppstod set baud-rate 115200

RTK.0> ppstod set frame data offset 0 value 0x11

RTK.0> ppstod set frame data offset 1 value 0x22

RTK.0> ppstod set frame data offset 2 value 0x33
```

```
RTK.0> ppstod set frame data offset 3 value 0x44
RTK.0> ppstod set frame data offset 4 value 0x55
RTK.0> ppstod set frame data offset 5 value 0x66
RTK.0> ppstod set frame data offset 6 value 0x77
RTK.0> ppstod set frame data offset 7 value 0x88
RTK.0> ppstod set frame data offset 8 value 0x99
RTK.0> ppstod set frame data offset 9 value 0xaa
RTK.0> ppstod set frame data offset 10 value 0xbb
RTK.0> ppstod set frame len 11
RTK.0> ppstod set mode software
```


9. Time Synchronous

9.1. PTP Grand Master Process

PTP Grand Master Process will create two threads, one is GM_main_thread for handling PTP packet, timer of sending announce message and timer of sending sync message, the other is GM_rx_thread for reciving PTP packet and informing GM_main_thread.

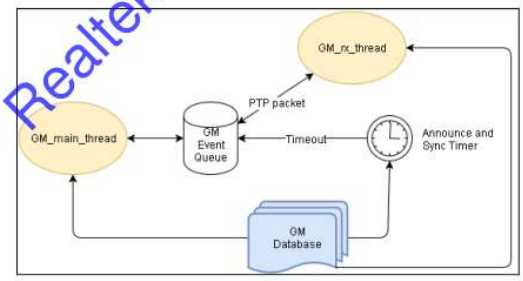

Figure 18. PTP Grand Master Process

The state machines of GM_main_thread are Stop, Start, Wait event, Time Handle, Two Step Message Handle and Message Handle.

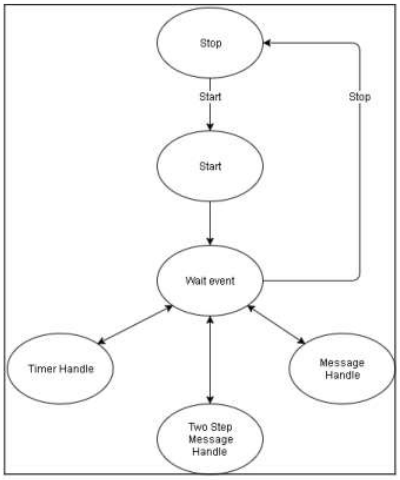

Figure 19. GM Main Thread State Machine

Stop state is initial state. Start state is creating GM event queue. Wait event state is waiting PTP packet, announce time out or sync time out. Two Step Message Handle state is handling two step PTP packet. Message Handle state is handling one step PTP packet.

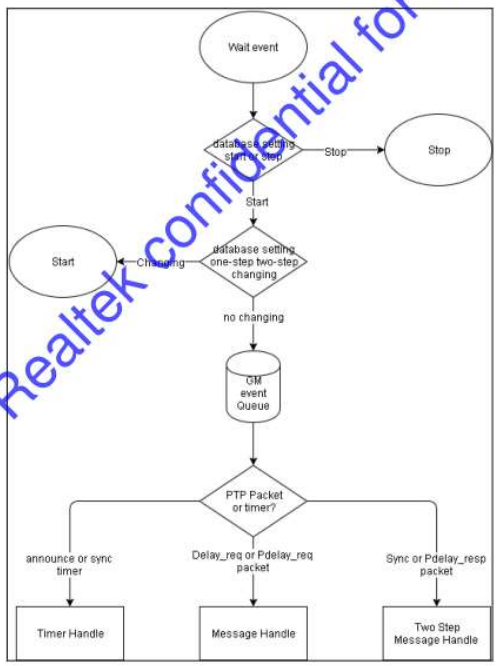

Figure 20. GM Main Thread Wait Event

The state machines of GM_rx_thread are Stop, Start, Waiting PTP packet and Informing. Stop state is initial state. Start state is creating socket for listening PTP packet. Waiting PTP packet is listening socket. Informing state is informing GM_main_thread when PTP packet is coming.

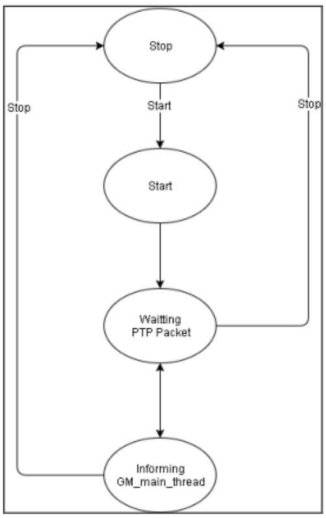

Figure 21. GM Rx Thread State Machine

PTP Master has tested with our PTP Slave and the calculating offset of PTP Slave is converging to +-20ns.

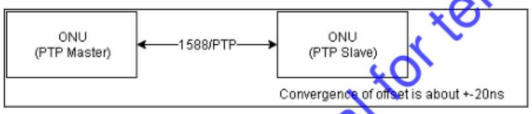

Figure 22. PTP Master test with PTP Slave

PTP Grand Master supports many options. We can set the option to decide to run one-step or two-step mode. We can control announce and sync time interval. We can control if PTP Master send announce message or not and define the parameter of announce message.

```
/* PTP Master support option
*/
 # ptpmaster -h
 -s        PTP Mode 1:one-step 2:two-step
 -at       Announce Interval Time
 -st       Syc Interval Time ex: -st sec msec
 -pa       Send Announce 1:y 2:n
 -pp       Peer Delay Support 1:y 2:n
 -p1       Announce Priority 1
 -p2       Announce Priority 2
 -cq       Announce Clock Quality
 -ts       Announce Time Source
```

-da	Daemon 0:disable 1:enable
-du	Dump PTP Master Configuration
-d	Debug printf 0,1,2
-h	Help

9.2. 1PPS ToD Module

1PPS ToD Module is software which form ToD frame and output frame to ToD pin or uart. 1PPS ToD Module will listen 1PPS interrupt and output the GPS Time which is translated by UTC system time. 1PPS ToD Module will calculate CRC of ToD frame. The state machines of 1PPS ToD Module are Stop, Start, Waiting PTP Interrupt and Output. Stop state is initial state. Start state is doing the initial setting like baudrate, output to uart and etc. Waiting PTP Interrupt state is the interrupt triggered every 1 second. Output state is calculating GPS time and ouput the ToD by the format of China Mobile.

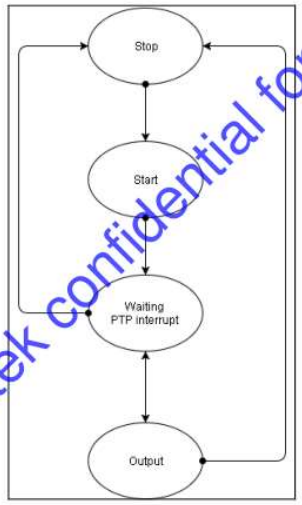

Figure 23. 1PPS ToD state machine

For ToD format of China Mobile, it ouputs 23bytes at baudrate of 9600bps. Each byte includes one start bit and one stop bit. Outputting 23bytes ideally takes about (23*10)/9600=0.02395sec=23.95ms. The actual test of 1PPSToD module outputting 23bytes at baudrate of 9600bps to uart is about 24ms.

1PPS ToD Module supports proc options which we can define baudrate, output to ToD pin or uart, reverse the byte order and print the debug message on uart.

```
/* 1PPSToD support proc option
*/
# ls /proc/ppstod_drv/
baudrate  enable  output  print  reverse
```

9.3. Time Synchronous Process

Time Synchronous Process includes controlling PON SyncE, PON ToD, PTP Master and 1PPSToD.

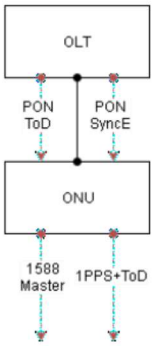

Figure 24. Time Synchronous Environment

Time Synchronous Procsee enables or disables PTP Master Process and 1PPSToD Module after checking status of PON SyncE and PON ToD.

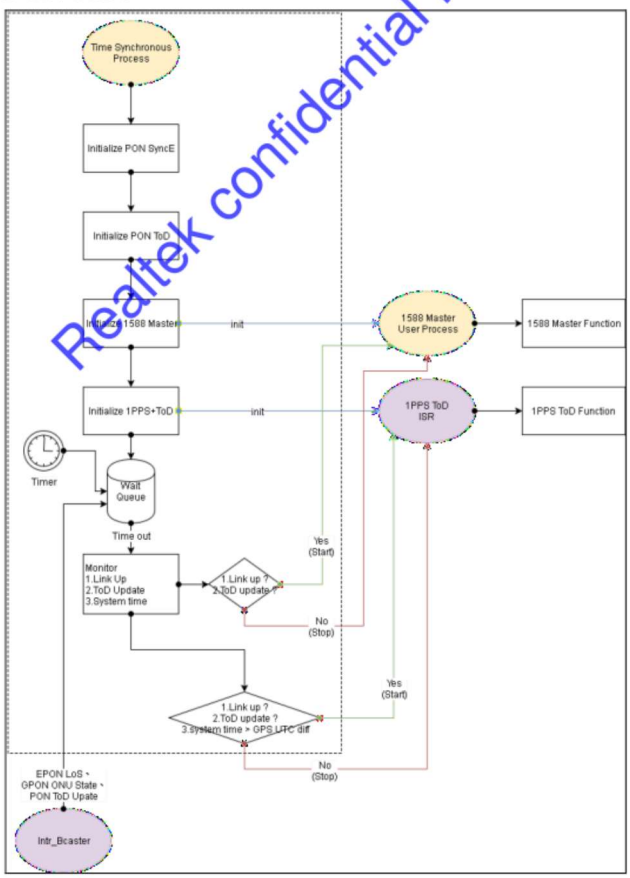

Figure 25. Time Synchronous Software Flow

Time Synchronous Process uses shell script to start or stop PTP Master Process and 1PPSToD Module. We can change the script to control PTP Master Process and 1PPSToD Module.

```
/* /etc/ptpmaster.sh
*/
#!/bin/sh
case "$1" in
        start)
                echo -n "Starting PTP Master"
                pidof ptpmaster 2>/dev/null 1>/dev/null || /bin/ptpmaster
                echo "."
                ;;
        stop)
                echo -n "Stopping PTP Master"
                killall ptpmaster 2>/dev/null 1>/dev/null
                echo "."
                ;;
        restart)
                echo -n "Stopping PTP Master"
                killall ptpmaster 2>/dev/null 1>/dev/null
                echo "."
                echo -n "Starting PTP Master"
                pidof ptpmaster 2>/dev/null 1>/dev/null || /bin/ptpmaster
                echo "."
                ;;
        *)
                echo "Usage: /etc/ptpmaster.sh start|stop|restart"
                exit 1
                ;;
```

```
/* /etc/ppstod.sh
*/
#!/bin/sh
case "$1" in
        start)
                echo -n "Starting 1PPSToD"
                echo 1 > /proc/ppstod_drv/enable
                echo "."
                ;;
        stop)
                echo -n "Stopping 1PPSToD"
                echo 0 > /proc/ppstod_drv/enable
                echo "."
                ;;
        restart)
                echo -n "Stopping 1PPSToD"
                echo 0 > /proc/ppstod_drv/enable
                echo "."
                echo -n "Starting 1PPSToD"
                echo 1 > /proc/ppstod_drv/enable
                echo "."
                ;;
        *)
                echo "Usage: /etc/ppstod.sh start|stop|restart"
                exit 1
                ;;
```


Time Synchronous Process supports using the other path to run start/stop script of PTP Master Process and 1PPSToD Module.

```
/* Time Synchronous support option
*/
# timesync -h
-d                  Debug printf 0,1,2
-fptpmaster         PTPMaster script path
-fppstod            1PPSToD script path
```

全文阅读已结束，下载本文需要使用

400 积分

下载此文档

阅读了该文档的用户还阅读了这些文档

RTL9607C_LED_Apr

RTL9607C_RTK_Por

RTL9607C_L2_Table

RTL9607C_Storm_Fi

EngNote_RTL9607C note_V01(2019111

Application_Nc

发表评论

验证码： 换一张

匿名评论

提交