ENT-AN1130
1588 PHY User Integration Guide

July 2018

a Microchip company

Contents

1 Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

1.1 Revision 1.1

Revision 1.1 of this document was published in July 2018. The following is a summary of the changes:

- The document title was updated to more accurately reflect content. Obsolete content was removed.
- The procedure for PTP clock time distribution to 1588 PHYs was clarified. For more information, see Distributing a 1588 System Clock to PHYs in a System.
- The PTP time (LTC) maintenance information was clarified. For more information, see Maintaining the 1588 Local Time Counter.
- 1PPS output and 1588 reference clock details were clarified. For more information, see Purpose of the 1PPS Output and Reference Clock for 1588.
- 1588 timestamp processing details were updated. For more information, see Timestamp Processing within the 1588 PHY.
- Egress TS FIFO information was updated. For more information, see Egress Time Stamp FIFO in the 1588 PHY.
- Transmit and receive timestamp handling details were updated. For more information, see Reading Time Stamps from PTP Event Messages.
- The initialization/disabling of 1588 through API procedure was clarified. For more information, see API Initialization of the 1588 Engine.
- 1588 frame formatting/interoperability information was clarified. For more information, see Inter-operation of Microsemi 1588 PHYs with Other 1588 Frame Formats.
- Common questions on the working of the transparent clock were addressed. For more information, see Solutions.

1.2 Revision 1.0

Revision 1.0 was of this document was published in July 2015. This was the first publication.

2 ENT-AN1130 Common Customer Integration Topics for the 1588 PHYs

This section covers common operational topics when integrating the 1588 time stamping unit in Microsemi PHYs into a PTP system. The following discussion is relevant for all Microsemi 1588 PHYs, except where noted as product-specific information.

2.1 Clock

This section provides solutions for clock-related issues.

2.1.1 Distrib uting a 1588 System Clock to PHYs in a System

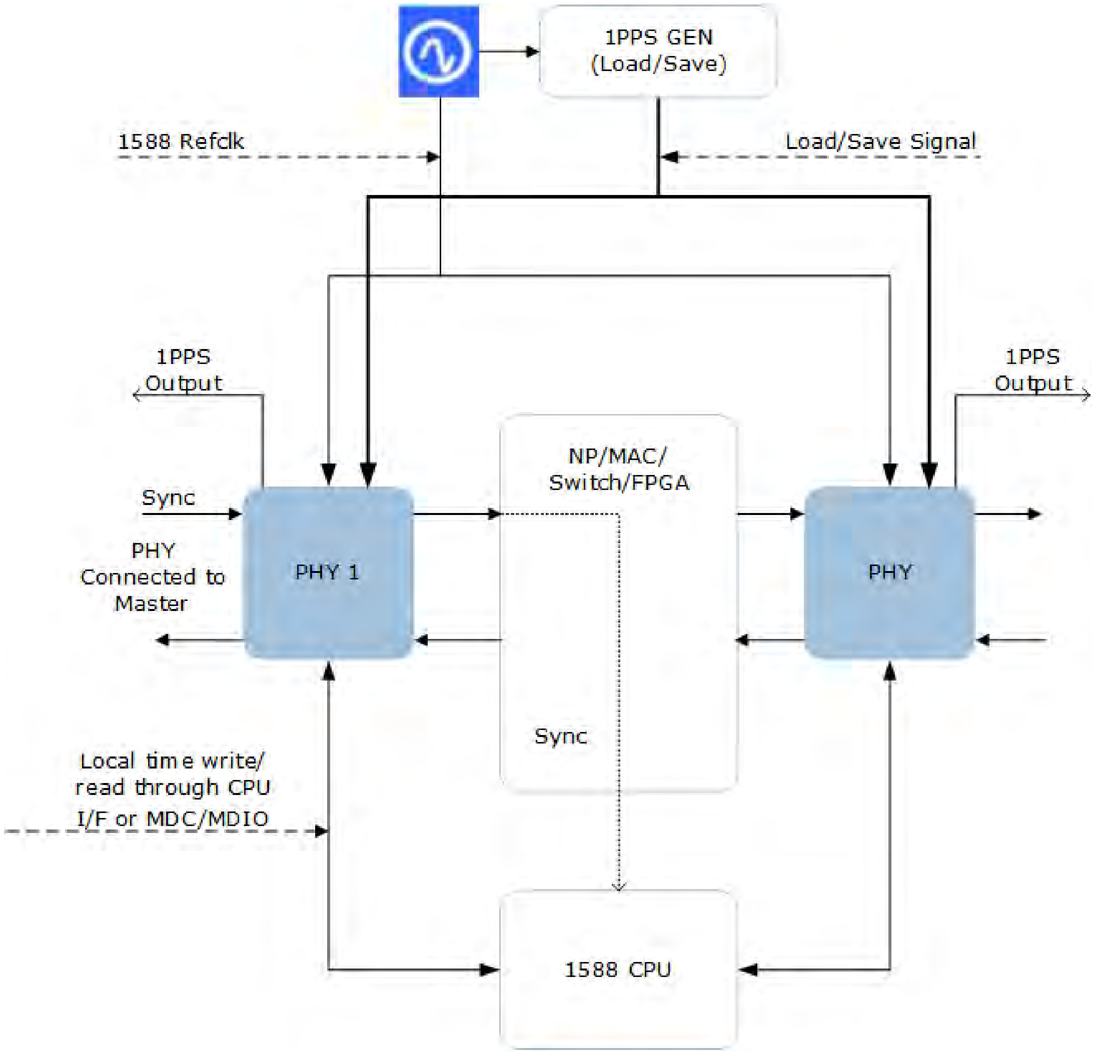
This section describes how to distribute a 1588 system clock to PHYs in a system.

2.1.1.1 Local Time Load and Synchronization

To distribute the 1588 system clock to PHYs in a system:

1. Provide local time to all Microsemi 1588 PHYs through a CPU interface or MDC/MDIO, as shown.

Figure 1 • PHY Synchronization Signals



Note: The VSC8490 and VSC8258 family of PHYs use a CPU interface, while all other 1588 PHYs usually use the MDIO. The VSC8584 family is the sole 1G device that can also use a SPI-slave interface. The VSC8492 family uses a parallel CPU interface.

2. Provide the global load/save signal to load all the PHY counters at the same precise time.
   Note: The global load/save can be a 1PPS (or low frequency) signal generated from a single timing source that provides one common signal connected to all the PHYs. The 1PPS should be generated in hardware with highly accurate time of delivery. The 1PPS can be generated by dividing the system reference clock at a central location and distributing the resulting signal (1PPS) to all 1588 PHYs in the system.
3. Provide a clock to all the PHYs so that the 1588 counters progress at the same rate (refer to the specific PHY datasheets for frequency requirements).
4. See the 1588 ptptime APIs for arming, setting, and disarming the load bits of all PHY counters.
   Note: The 1588 reference clock can be a free-running oscillator or derived from a system clock.

2.1.1.2 Synchronizing to a PTP Master

This section lists the steps for synchronizing and adjusting local time in multiple PHYs to a PTP master for ordinary clock (OC) and boundary clock (BC) applications.

1. The 1588 CPU receives sync frames time stamped by the ingress PHY1 connected to the PTP master.

2. The CPU goes through the complete IEEE 1588 procedure of exchanging PTP messages to calculate offset from the master.
   Note: The procedure of exchanging PTP messages is repeated several times and a packet delay variation (PDV) filtering algorithm is applied to filter out delay variation.
3. The 1588 CPU sets the save bit in PHY1, and then reads the saved local time, `t0`. The local time is saved in the LTC registers on the next load/save pulse. The local time format is unsigned 48-bit (sec) || unsigned 30-bit (ns).
4. The CPU calculates expected time, `t1`, at the next load/save pulse corrected by the offset error from the PTP master. The next load/save pulse, `tp`, can be one or more one PPS periods.
   `t1 = t0 + Offset + tp + (optional trace delay compensation for load /save signal) + fixed_load_latency`
5. The CPU writes the expected time, `t1`, into PHY1, and then sets the load bit. The local time is loaded into the local time counter on the next load/save pulse.
6. The CPU checks if the offset becomes zero. Otherwise, it adjusts the local time with the ± 1 ns function until the offset becomes zero.
7. The CPU sets the save bit on PHY1, and then reads the local time, `t2`.
8. The CPU then writes all other PHYs with the value `t2 + tp + (optional trace delay compensation for load/save signal) + fixed_load_latency`, and then sets the load bit in each PHY. The local time is loaded into all PHYs on the next pulse on the load/save pin.
   Note: The trace delay is the known PCB trace delay on the load/save pin for each PHY. In most cases, the CPU can load all PHYs within one load or save time period, which is typically one second. If the CPU fails to load all PHYs within one load or save time period, this procedure is repeated over multiple load or save periods.
   Note: The fixed_load_latency is the time difference between when the 1588 PHY captures the load /save rising edge and the actual instant when a loaded value updates inside the local time counter. For example, as shown in VSC8574 and VSC8584 datasheets, it is two cycles of the 1588 counter clock.
9. In the steady state, the CPU sets the save bit in all the PHYs, reads back local time values, and checks to make sure that all values are the same.
10. The CPU continuously checks the local time drifts compared to the recovered PTP time. It corrects local time by issuing ± 1 ns commands to all PHYs.
    Note: For high port-counts, steps 8 and 9 can be broken down on a per-chip basis in order to compare against the local PTP master's time being tracked by the CPU.

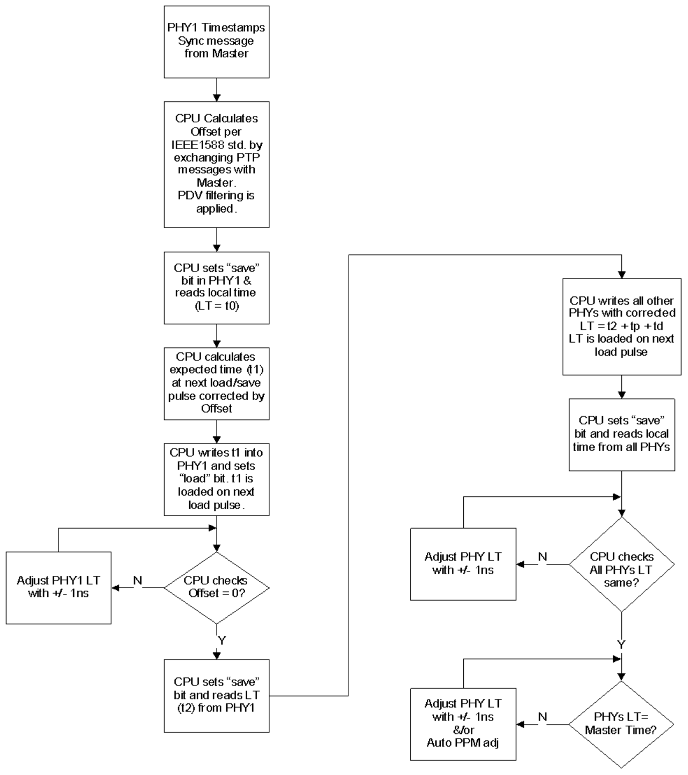
PHYs can also be programmed to automatically adjust to the PPM differences when local time is drifting at a fixed rate compared to the PTP time, as described in Automatic PPM Adjustments.

The following are the related registers.

- `LTC_SAVE_ENA and LTC_LOAD_ENA in LTC_CTRL`
- `LTC_SAVED_SEC_H`
- `LTC_SAVED_SEC_L`
- `LTC_SAVED_NS`
- `LTC_LOAD_SEC_H`
- `LTC_LOAD_SEC_L`
- `LTC_LOAD_NS`
- `LTC_ADD_SUB_1NS_REQ`
- `LTC_ADD_SUB_1NS`
- `LTC_AUTO_ADD_SUB_1NS`
- `LTC_AUTO_ADJUST_NS`

The following illustration shows the workflow used to set and adjust the PHY local time to the master PTP time.

Figure 2 • Set and Adjust PHY Local Time to MasterPTP Time


PHY1 Timestamps Sync message from Master
CPU Calculates Offset per IEEE1588 std. by exchanging PTP messages with Master. PDV filtering is applied.
CPU sets "save" bit in PHY1 & reads local time (LT = t0)
CPU calculates expected time (t1) at next load/save pulse corrected by Offset
CPU writes t1 into PHY1 and sets "load" bit. t1 is loaded on next load pulse.
CPU checks Offset = 0?
N
Adjust PHY1 LT with +/- 1ns
Y
CPU sets "save" bit and reads LT (t2) from PHY1
CPU writes all other PHYs with corrected LT = t2 + tp + td LT is loaded on next load pulse
CPU sets "save" bit and reads local time from all PHYs
CPU checks All PHYs LT same?
N
Adjust PHY LT with +/- 1ns
Y
PHYs LT= Master Time?
N
Adjust PHY LT with +/- 1ns &/or Auto PPM adj

The PHYs also have 1PPS output pins that provide equally spaced sync pulses to measure the timing skew between PHYs. By measuring the skew between the 1PPS test output from each PHY, it is possible to measure the nominal correction values that can be incorporated into the system software. Variation due to temperature and system differences should be minimized.

2.1.1.3 Loading Local Time without PTP Master

The process for loading local time into multiple PHYs without synchronizing local time to a master is as follows:

1. The CPU writes all PHYs with local time at the next load or save pulse, and then sets the load bit in each PHY. This time includes the known PCB trace delay on the load or save pin of each PHY, as well as the fixed load latency as defined in Synchronizing to a PTP Master. Local time is loaded into all PHYs on the next pulse on the load/save pin. In most cases, the CPU can load all PHYs within one load or save time period, which is typically one second. Otherwise, this can be achieved by repeating the steps over multiple load or save periods.
2. The CPU sets the save bit in all PHYs, reads back local time values, and then checks to ensure that all the values are the same. The CPU adjusts the PHY local time using the ± 1 ns function as necessary.

2.1.2 Maintaining the 1588 Local Time Counter

The frequency of local time adjustment is system dependent. The PHYs support two methods for local time and PPM adjustment: the ± 1 ns function within the CPU and automatic PPM adjustments.

2.1.2.1 ± 1 ns Function with the CPU

In order to support the adjustments as necessary, the CPU issues ± 1 ns command that adds or subtracts 1 ns from the local time counter.

2.1.2.2 Automatic PPM Adjustments

The PPM offset can be used to calculate the number of the local-time clocks it takes to reach an offset of 1 ns, and this value plus an add or subtract value can be programmed into the PHY for the automatic PPM adjustments.

Example

If a 250 MHz local time counter clock is off by 100 PPM (0.01%), then the 4 ns period is off by 0.0004 ns every clock cycle. A 1 ns adjustment would need to be made to the local time counter every 1 ns/(0.0004 ns) cycles, or every 2500 cycles. Because the clock period is 4 ns, the LTC_AUTO_ADJUST.LTC_AUTO_ADJUST_NS register would be set to 2500 x 4, which is 10,000 ns.

2.1.3 Purpose of the 1PPS Output

There is one 1PPS output pin available per port (that is, PPS0 for port 0, PPS1 for port 1, and so on). The PPS0 pin on port 0 of the device can be programmed to internally mux the 1PPS of any port out on the PPS0 pin.

The 1PPS output pin serves the following purposes:

- Provides a way to ensure that all the PHYs are in sync across all the ports in a system, perhaps by sending all outputs to an FPGA for host CPU skew measurements as mentioned in Synchronizing to a PTP Master.
- Provides a way to calibrate the propagation delay differences among the different PHYs on all PTP ports, if there are any. This can be based on a one-time lab measurement of these outputs that can be used as a correction factor to load from software per PHY.

2.1.4 Reference Clock for 1588

There is a separate 1588 reference clock used for the local time counter (LTC) in the PHY. The clock source for the LTC is passed in through the 1588 configuration as part of the `vtss_phy_ts_init` API.

For the VSC8574, VSC8492, and VSC8487-15 family of devices, the period of the 1588 reference clock sets the time stamp resolution.

For the VSC8584. VSC8490, and VSC8258 family of devices, time stamp resolution is independent of the clock frequency due to high-resolution circuitry. The LTC runs directly off of the 1588 reference clock.

For all devices, the PHY as configured through the 1588 API only supports the following frequencies for an externally-supplied 1588 reference clock: 125, 156.25, 200, and 250 MHz.

2.2 Time Stamping

This section discusses time stamping operations.

2.2.1 Time Format inside the 1588 PHY

The following sections describe the timestamp format within Microsemi 1588 PHYs.

2.2.1.1 Local Time Counter inside the Timestamping Unit

The local time counter (LTC) inside the Microsemi 1588 PHY keeps time of day as an ordered set in accordance with the IEEE 1588v2 standard: unsigned 48-bit (seconds) || unsigned 30-bit (nanoseconds)

While the LTC register width is 80 bits, note that the nanoseconds value wraps at 10^9 (1 billion) nanoseconds when it increments the upper 48-bit seconds value of the ordered set.

2.2.1.2 Timestamp Format

Typical 1588 user implementations process PTP timestamps in an 80-bit format. However, 80-bit timestamps can consume a lot of memory on embedded systems. Thus, the storage capacity of time stamps can be limited, particularly on embedded hardware. The Microsemi 1588 PHY allows 4-byte timestamping for applications that do not require the upper 6 bytes (for example, residence-time calculation). Whether the time stamp format programmed by the user is 4 or 10 bytes, the size of the frame signature (such as for T3 timestamp harvesting) is independently programmable.

2.2.1.3 Ingress Timestamp Format

The RESERVED field used for intra-switch storage of the ingress timestamp can be written to the PTP frame in either of two formats:

- 30-bit value that is nanoseconds-only
- 32-bit value that is modulo-2^{32} of (nanoseconds + seconds* 10^9)

The user configures the RESERVED (Rx) timestamp format as part of the 1588 clock configuration passed into the 1588 API. 30-bit mode is more common, due to its processing simplicity, but 30-bit mode places a tighter time constraint on rollover handling in the host CPU's PTP application.

2.2.2 Timestamp Resolution within the 1588 Timestamping Unit

Time stamp resolution is differentiated by product family.

For the VSC8487-15, VSC8574, and VSC8492 families of 1588 PHYs, the time stamp resolution is equal to the 1588 reference clock period. For example, the resolution is 4 ns for a 250 MHz reference clock and 8 ns for a 125 MHz clock.

For the newer VSC8584, VSC8490, and VSC8258 families of 1588 PHYs, the time stamp resolution is always 1 ns due to high-resolution circuitry within its newer 1588 timestamping processor.

2.2.3 Timestamp Processing within the 1588 PHY

The following sections describe the timestamp processing of Microsemi 1588 PHYs.

2.2.3.1 Support for 1588v2 Clock Modes

The 1588 PHY supports the following clock modes in one-step or two-step modes:

- OC master or slave
- BC
- E2E TC
- Peer-to-peer transparent clock (P2P TC)

From a PTP frame-processing perspective, the clock mode configuration determines how the 1588 timestamping engine inside the PHY acts upon processing PTP event messages.

Note: In addition to timestamping, TC Mode B can support frequency recovery using the 1588 ToD (syntonization), as when the sync messages are forwarded through the 1588 engine. This allows a local TC to frequency-lock to a PTP master without resorting to more sophisticated methods (for instance, Sync-E clock recovery).

2.2.3.2 Timestamping 1588v2 Frames

The 1588 engine inside the PHY timestamps PTP event messages such as: Sync, Delay_Req, Pdelay_Req, and Pdelay_Resp. General PTP messages (such as Announce) are not time stamped by the 1588 engine.

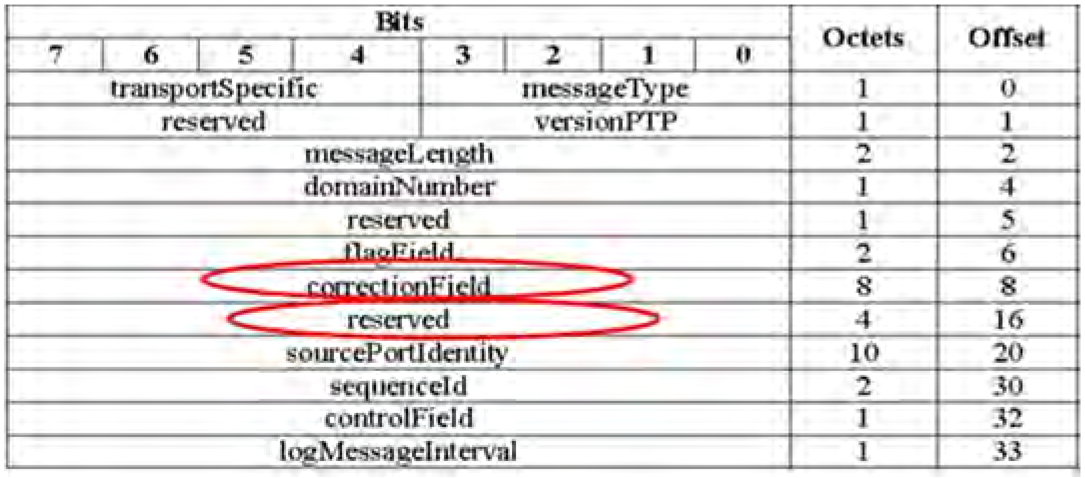
The PTP payload contains a number of fields that can be modified by the PHY as follows:

- The correction field is updated according to the IEEE 1588v2 protocol
- The reserved bytes (32-bit field) are overwritten by the 1588 engine for ingress time stamp transfer internal to the PTP application stack. For PTP messages with overwritten reserved bytes, the reserved field is zeroed by the 1588 engine at packet egress.

- Transmit time stamps are overwritten in the PTP message for one-step applications and are saved in a egress timestamp FIFO for T3 harvesting in the PTP slave node, or two-step applications, or for path delay calculations.

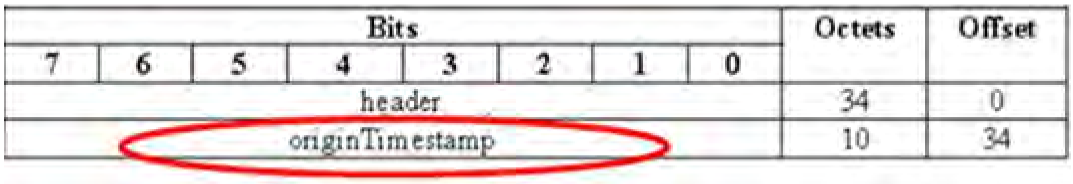
The relevant fields in the PTP payload which may be modified by the 1588 engine inside the PHY are shown in the following.

Figure 3 • 1588v2 Header



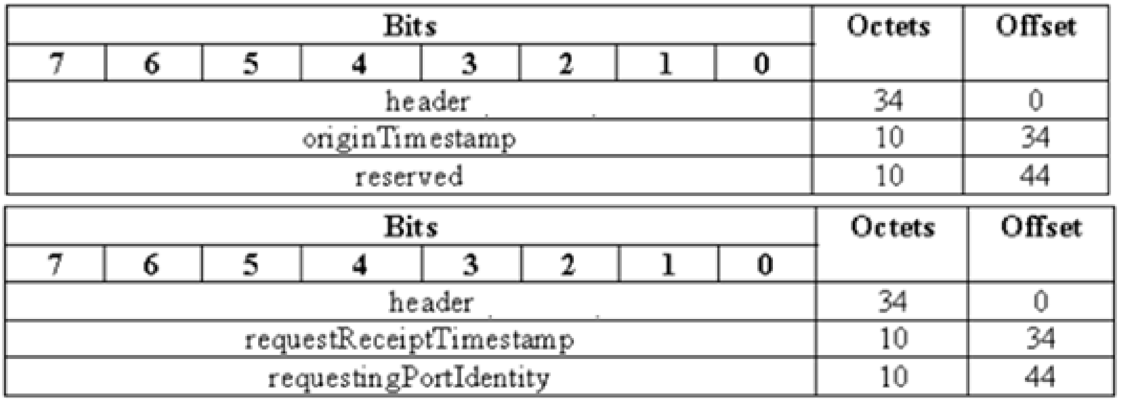
Bits								Octets	Offset
7	6	5	4	3	2	1	0		
transportSpecific				messageType				1	0
reserved				versionPTP				1	1
messageLength								2	2
domainNumber								1	4
reserved								1	5
flagField								2	6
correctionField								8	8
reserved								4	16
sourcePortIdentity								10	20
sequenceId								2	30
controlField								1	32
logMessageInterval								1	33

Figure 4 • Sync and Delay_Req Frame Format



Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header								34	0
originTimestamp								10	34

Figure 5 • Pdelay_Req_Pdelay_Resp_Frame_Format



Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header								34	0
originTimestamp								10	34
reserved								10	44

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header								34	0
requestReceiptTimestamp								10	34
requestingPortIdentity								10	44

2.2.3.3 Preamble Reduction Mode

The 1588 engine within the PHY supports a non-standard operational mode to update ingress timestamps that do not use the reserved bytes of the PTP header. This mode reduces the preamble of ingress packets by 4 bytes for all frames. For any matching PTP frames, it replaces the incoming 4-byte FCS with a 4-byte time stamp and appends a new, valid 4 byte FCS.

2.2.4 Calculating End-to-End Transparent Clock Residence Time

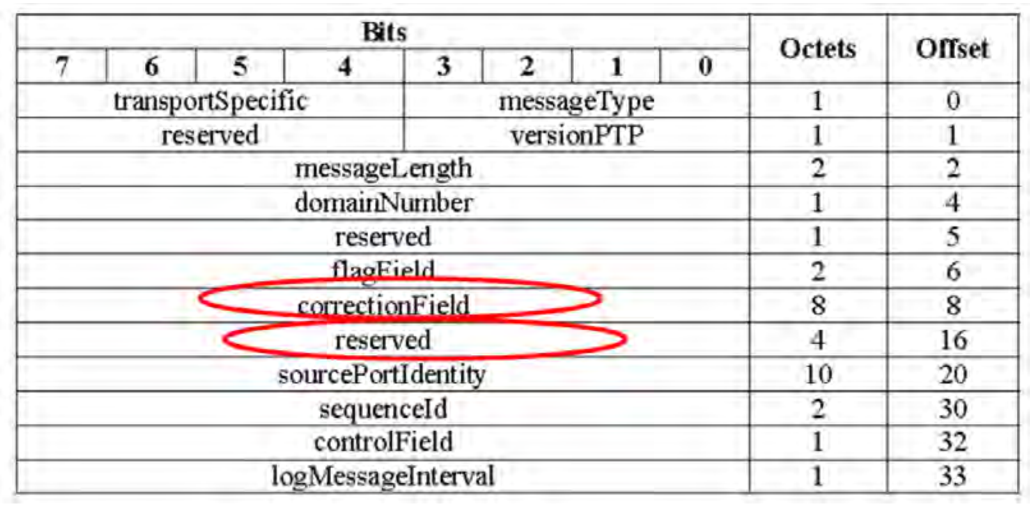
Microsemi 1588 PHYs support the following types of end-to-end Transparent Clock (E2E TC) residence time calculations.

2.2.4.1 Normal Residence Time Calculation

The residence time in Transparent Clock Mode B is calculated as follows:

1. On ingress, the Rx time stamp is written into the four reserved bytes in the PTP header as shown.

Figure 6 • PTP Header



Bits								Octets	Offset
7	6	5	4	3	2	1	0		
transportSpecific				messageType				1	0
reserved				versionPTP				1	1
messageLength								2	2
domainNumber								1	4
reserved								1	5
flagField								2	6
correctionField								8	8
reserved								4	16
sourcePortIdentity								10	20
sequenceId								2	30
controlField								1	32
logMessageInterval								1	33

2. On egress, the PHY extracts the 32-bit Rx time stamp from the four reserved bytes, subtracts the Rx time stamp from the TX time stamp, and adds this value to the CF. In other words, CF= CF + Residence Time.
3. The PHY clears the reserved bytes back to 0.

The normal residence E2E TC mode allows the PTP engine to sniff the PTP sync frames that have the Rx time stamp located in the reserved bytes. This allows the PTP engine to recover the frequency of the PTP system and syntonize the clock of the system to match the PTP frequency or let the PHY compensate for the frequency differences between the system clock and the PTP clock.

2.2.4.2 Alternate Residence Time Calculation

The residence time in Transparent Clock Modes A or C is calculated as follows:

1. The ingress PHY subtracts the receive time stamp (Rx_TS) from the original CF.
2. The egress PHY adds the transmit time stamp (Tx_TS) to the CF. In other words, CF= CF + Residence Time.

This works well in end-to-end transparent clocks, but it does not allow the TC to recover the frequency based on the PTP frames to syntonize the clock of the system and minimize the residence time error that is caused by the frequency difference of the system compared to the PTP frequency. The residence time error is equal to residence time multiplied by ppm offset.

If the TC is already locked to a PRC by other means (SyncE or E1) or if the residence time is low, the error is insignificant.

Transparent Clock modes A and C perform the same kind of operations on the CF, but the precision of those operations will differ. Mode C is preferred as it uses two's-complement arithmetic, so rollover is implicitly handled in the operations. For more information about rollover handling in other TC modes, see ENT-AN1132 Residence Time Calculation in Transparent Clocks.

2.2.5 Egress Time Stamp FIFO in the 1588 PHY

The following sections describe the egress time stamp FIFO in Microsemi 1588 PHYs.

2.2.5.1 Purpose of the Egress Time Stamp FIFO

The time stamp FIFO within the 1588 PHY stores egress time stamps along with the frame signature information. This information can be read out by a CPU or pushed out on a dedicated serial time stamp output Interface (for example, for use in a two-step mode to create follow-up messages). The timestamp FIFO is also used in one-step mode to save time stamps for the Delay_Req/Pdelay_Req messages used in path delay measurements (also known as "timestamp harvesting").

2.2.5.2 Number of Timestamps in the Egress Timestamp FIFO

The depth of the egress time stamp FIFO is 8 entries of 208 bits per entry. Seven hardware register locations are used to access all available entries within the FIFO; the timestamp access API automatically pops all entries from the hardware FIFO each time the API is called.
The size of the signature stored in the FIFO is 16 bytes, and the size of the egress timestamp stored in the FIFO is 10 bytes. The length is not configurable from the API.

2.2.5.3 Identifying Timestamps in the Egress Timestamp FIFO

Each time stamp uses a unique frame signature for its identification. The frame signature contains user-selectable portions of the PTP header, such as sequence ID and MAC address. The IEEE 1588 protocol uses the sequence_ID in order to match a saved timestamp with a transmitted frame. The sequence_ID is a 16-bit number generated by the source of the PTP frame. The number must be incremented for each frame that is transmitted to a destination, and the source assigns a different pool of sequence IDs to each port (IEEE 1588-2008 Clause 7.3.7).
In more complex PT networks (with several PTP domains per port), it might be necessary to save the domain number or the destination address as well. Whichever combination of PTP frame data is ultimately chosen for the signature, that information must be transported up to the application along with the corresponding timestamp for the PTP stack to work correctly. The Microsemi Unified API provides retrieval of the timestamp/signature pairs to the PTP application, but processing by the PTP application stack is beyond its scope.

2.2.5.4 Using the Egress Time Stamp FIFO

The PTP application can read egress timestamps through the SPI master interface or management interfaces (that may have multiple options), depending on the 1588 PHY used. Detailed setup of the access method is described in **ENT-AN1051**. Beyond the API usage basics described in that document, the user must also consider the following.

- Reading T3 timestamps from the CPU management interface versus serial timestamp interface—service of the egress timestamp FIFO must at least meet the maximum PTP event message rate for the relevant port in order to avoid FIFO overflow on that port, which is particularly important for CPU management access. For serial timestamp interface (push-out), that interface is a global bus that services all ports within a 1588 PHY chip. Consult the relevant 1588 PHY datasheet for detailed timing information and descriptions of the formatted timestamp/frame signature word pushed out on its interface (for example, for writing driver-level access).

- Polling versus event notification for egress FIFO handling—polling is the simplest implementation, but may be inefficient compared to interrupt-driven servicing of the FIFO. As an example of event notification, it may be enabled using the 1588 PHY API in order to interrupt the PTP application for egress FIFO handling, as shown.

```
#define VTSS_PHY_TS_EGR_TIMESTAMP_CAPTURED 0x20 /**< Timestamp
captured in Tx TSFIFO */
#define VTSS_PHY_TS_EGR_FIFO_OVERFLOW 0x40 /**< Tx TSFIFO overflow
**/
vtss_rc rc;
vtss_phy_ts_event_t ev_mask=0;
rc = vtss_phy_ts_event_enable_get(board->inst, port_no, &ev_mask);
ev_mask |= VTSS_PHY_TS_EGR_TIMESTAMP_CAPTURED;
rc = vtss_phy_ts_event_enable_set(board->inst, port_no, enable,
ev_mask);
printf ("Event Mask: 0x%x: Enable: 0x%x; for port %d :", ev_mask,
enable, port_no);
```

2.2.6 Reading Time Stamps from PTP Event Messages

For most PTP applications, the guidelines described in the following sections must be considered in the PTP application stack.

2.2.6.1 Transmit Time Stamps in PTP Frames

Transmit time stamps are captured through peripheral connections to the egress timestamp FIFO, as described in this document.

2.2.6.2 Receive Time Stamps in PTP Frames

Received PTP event messages must be extracted from the packet stream through the network transport stack (TCP/IP). Then, the timestamp within the PTP frame must be merged with the upper seconds of the ToD counter. The implementation for merging depends on the receive timestamping length, and the methods described assume the CPU ToD counter and the 1588 PHY LTC counter accumulate based on the same clock source (for example, the source driving the 1588_DIFF_CLK pins of the PHY).

2.2.6.2.1 30-bit Ingress Timestamp Format

To interpret the 30-bit time stamp updated in PHY at ingress into reserved bytes of PTP header, refer to the following sample pseudocode that reads the timestamp in seconds and nanoseconds that was written into the packet.

```
u32 reserved, sec, nsec ;
typedef struct {
u64 seconds;
u32 subseconds;
} time_of_day_t;
time_of_day_t corr_time_of_day;
reserved        = extract_reserved(ptpheader); /* This function should be provided by app

sec = 0; /* always less than 10^9 ns */
nsec = reserved;  /* same as LTC sub-seconds i.e. %1000000000 */
corr_time_of_day = ToD_counter_read(); /* application-specific call to the host ToD count

if (corr_time_of_day.ns < nsec) {
// a 1-second rollover occurred since timestamp was written
corr_time_of_day.seconds--;
}
corr_time_of_day.subseconds = nsec;
```

2.2.6.2.2 32-bit Ingress Timestamp Format

Conversion of a 32-bit ingress timestamp must correctly account for wrapping of the RESERVED field (modulo-2^{32}). The generated value written into the RESERVED field by the 1588 PHY is modulo-2^{32} of (nanoseconds + seconds* 10^9) where the nanoseconds field is the 30 least-significant bits of the 1588 PHY's LTC, and the seconds field is the 48 most-significant bits of the PHY's LTC.

Note: The nanoseconds field wraps every 10^9 nanoseconds (that is, not 2^{30}).

As the timestamp written into the RESERVED field by the 1588 PHY occurs earlier than PTP field extraction within the application, the RESERVED field can be merged with ToD in the application stack to arrive at a corrected ToD value by applying a time correction similar to the following pseudo-code.

```
u32 reserved, sec, nsec ;
u64 T2_correction;
u32 T2_correction_secs, T2_correction_nsecs;
typedef struct {
u64 seconds;
u32 subseconds;
} time_of_day_t;
time_of_day_t time_of_day;
time_of_day_t corr_time_of_day;

time_of_day.seconds=0; /* 48 bit value, seconds units */
time_of_day.subseconds = 0; /* 30 bit variable, nanoseconds units corr_time_of_day.second

corr_time_of_day.subseconds = 0; /* 30 bit variable, nanoseconds units */
time_of_day = ToD_counter_read(); /* application-specific call to the host ToD counter, w

reserved = extract_reserved(ptpheader); /* This function should be provided
by application, Extract Reserved bits time stamped at PHY in to reserved
field(offset 16 from start of PTP Hdr) of PTP header. */
T2_correction = ((time_of_day.seconds*10**9 + time_of_day.subseconds) % 2**32) - reserved

T2_correction_secs = T2_correction/10**9;
T2_correction_nsecs = T2_correction%10**9;
corr_time_of_day.seconds = time_of_day.seconds - T2_correction_secs;
corr_time_of_day.subseconds = time_of_day.subseconds - T2_correction_nsecs;
```

Note: To successfully merge ToD as shown, the PTP node must transport PTP event messages quickly enough in order to avoid aliasing of ToD seconds versus the 4-byte timestamp's seconds (that is, at the time when T2 was written by the 1588 PHY into the ingress frame). This implies that while in 30-bit timestamping mode, the application must perform the described calculation in less than 1 second after receiving a PTP event message.

32-bit timestamping mode offers 3 additional seconds of processing time to perform the described operation; however, the application may be required to perform additional timestamp manipulations, as indicated in the pseudo-code, depending on the PTP application stack.

Finally, the error of the host ToD counter (that is, versus UTC) must be subtracted from the "slack" in the PTP frame processing budget as well. The 1588 PHY LTC can be used for a simple, hardware-based accumulator solution to track ToD within the application, with minimal such error. For example, the application can schedule a periodic task (~50 milliseconds) to save and readback the LTC time within the 1588 PHY. The upper 48 bits of the LTC provide ToD seconds, and can be adjusted based on comparison of the lower 30 bits (sub-seconds) of the LTC. CPU-based (soft) ToD implementations also work, but may be more complex due to accuracy limitations within the application's host system.

Overall, implementation of a successful ingress timestamp decode requires customer investigation and, potentially, system integration efforts in order to connect all layers of the PTP application stack. Those details are beyond the scope of this document.

2.2.7 Constant Time Error Adjustments

There are separate programmable values for local correction and path asymmetry. Asymmetry is a signed 32-bit scaled nanoseconds value (allow full range of the correction-field in accordance with IEEE 1588v2, clause 11.6) and local correction is an unsigned 32-bit value in nanoseconds. The local correction values are configured independently for each port and for each direction (ingress/egress), which is useful for port-specific, external compensation (such as external triple-speed SFPs).

3 1588 PHY FAQs

3.1 API Initialization of the 1588 Engine

Issue

How do I enable/disable the 1588 engine inside the PHY?

Resolution

3.1.1 Initializing Time Stamping

To initialize a 1588 block in PHY, invoke the listed APIs in the following order.

1. `vtss_phy_ts_init`
2. `vtss_phy_ts_mode_set`
3. `vtss_phy_ts_ingress_engine_init`
4. `vtss_phy_ts_egress_engine_init`
5. `vtss_phy_ts_ingress_engine_conf_set`
6. `vtss_phy_ts_egress_engine_conf_set`

Steps 3 and 4 shall be called for each encapsulation used by the 1588 application.

Steps 5 and 6 shall be called for each flow used in a particular encapsulation.

After all flow configurations are complete for an encapsulation, configure actions for ingress by calling `vtss_phy_ts_ingress_engine_action_set` and `vtss_phy_ts_egress_engine_action_set`.

The latter two APIs only need to be called once, as they are global for all enabled flows.

3.1.2 Disabling Time Stamping

To disable the time stamping feature of the PHY, there are two options:

- Leave the compile flag, `VTSS_FEATURE_PHY_TIMESTAMP`, undefined (that is, omit in the `vtss_api/include/vtss_options.h` file). This will statically remove the 1588 API from the application image.
- Dynamically turn off timestamping using the `vtss_phy_ts_mode_set` API with the enablement Boolean set to `FALSE`. See 1588 programming application notes or demo application code for usage.

3.2 Inter-operation of Microsemi 1588 PHYs with Other 1588 Frame Formats

Issue

How do I configure the 1588 PHY to handle applications which use non-Microsemi PTP frame formats, such as transparent clocks with multi-vendor PHY timestampers?

Resolution

Microsemi PHYs provide time stamping based on 1588v2 standards so they are compatible with other standard frame formats. However, in non-standard applications the recommend inter-operation approach is to convert other RESERVED or CF formats to Microsemi format.

It is also possible to use the Microsemi 1588 timestamping unit to modify one of the reserved flag bits to indicate the Microsemi format to a host router that is performing internal PTP field conversions.

3.3 Network Stack Support for UDP over IPv6

Issue

How can I support PTP timestamping in a UDP over IPv6 frame encapsulation?

Resolution

UDP over IPv6 is one of the PTP encapsulations that the 1588 PHY can timestamp. In accordance with IEEE 1588v2 Annex E, a transmitting node shall extend the UDP payload of all PTP event messages with a pair of pad bytes. The host system is required to provide those pad bytes, as the Microsemi 1588 PHY cannot. For any such frame it timestamps, the 1588 PHY will update the pad byte field so that the IPv6 checksum field does not need modification.

Note: For information about general encapsulation support in Microsemi 1588 PHYs, please refer to ENT-AN1040 Encapsulation Support in Microsemi IEEE 1588v2 PHYs.

3.4 Limitations Due to Latency of Frames through the 1588 Timestamping Unit

Issue

Does the Microsemi 1588 PHY support full-duplex flow control when timestamping is enabled? Does the Microsemi 1588 PHY support half-duplex operation when timestamping is enabled?

Resolution

Latency limits the fiber distance that can be used with effective and lossless flow control, depending on internal buffers in the system. Full-duplex flow control using PAUSE frames can still be used, but half duplex mode of operation (CSMA/CD) is unsupported by the PHY when 1588 is operational.

3.5 CF Wrapping in One-Step Transparent Clock Mode

Issue

How can I take care of rollover in the correction field at ingress/egress in one-step TC mode?

Resolution

For first-generation 1588 PHY devices (such as the VSC8574), CF overflow is latched as an error event and a MDINT interrupt can be enabled. However, there is no attempt to update the ingress CF value when overflow occurs. Thus, CPU exception handling must be checked for interoperability with other timestamping devices within the TC node, though the network design usually ensures that CF never overflows. These 1588 PHY devices offer an approximately 5-hour rollover time due to 44-bit arithmetic to the CF.

Second-generation 1588 PHY devices (such as the VSC8584, VSC8490, VSC8258) support a wrapping two's-complement 48-bit field, so there is no need to explicitly track rollover due to the CF arithmetic performed by the PHY.

3.6 Maximum Number of Defined PTP Actions per Engine

Issue

There are two PTP actions defined for one engine. What's the purpose of defining two PTP actions for one engine?

```
typedef struct {
BOOL  action_ptp;  /**< is the action for PTP or OAM */
union {
       vtss_phy_ts_ptp_engine_action_t   ptp_conf[2]; /**< Max 2
       PTP action per engine */
       vtss_phy_ts_oam_engine_action_t   oam_conf[6]; /**< Max 6
       OAM action per engine */
} action; /**< PTP/OAM action config */
} vtss_phy_ts_engine_action_t;
```

Resolution

Defining two sets of PTP actions for a single 1588 channel allows configuring two clock types on the same port or pair of ports (channel)—for example, for users desiring a boundary clock and transparent clock operating on a single 1588 port.

The 1588 PHY hardware does not support more than two sets of PTP actions for a single 1588 channel.

The following code block shows a use case of two PTP actions.

```
ptp_action = &action_conf->action.ptp_conf[0];
              ptp_action->enable = TRUE;
              ptp_action->channel_map = 0
              ptp_action->clk_mode = VTSS_PHY_TS_PTP_CLOCK_MODE_BC1STEP;
              ptp_action->delaym_type = VTSS_PHY_TS_PTP_DELAYM_E2E/
VTSS_PHY_TS_PTP_DELAYM_P2P;
                ptp_action->ptp_conf.range_en = TRUE;
                ptp_action->ptp_conf.domain.range.lower = 0;
                ptp_action->ptp_conf.domain.range.upper = 0;

ptp_action = &action_conf->action.ptp_conf[1];
              ptp_action->enable = TRUE;
              ptp_action->channel_map = 0
              ptp_action->clk_mode = VTSS_PHY_TS_PTP_CLOCK_MODE_TC1STEP/
VTSS_PHY_TS_PTP_CLOCK_MODE_TC2STEP;
                ptp_action->delaym_type = VTSS_PHY_TS_PTP_DELAYM_E2E/
VTSS_PHY_TS_PTP_DELAYM_P2P;
                ptp_action->ptp_conf.range_en = TRUE;
                ptp_action->ptp_conf.domain.range.lower = 1;
                ptp_action->ptp_conf.domain.range.upper = 1;
vtss_phy_ts_egress_engine_action_set(API_INST_DEFAULT, port_no, eng_id,
action_conf)
```

3.7 Working of Transparent Clock in Microsemi PHY

The following table lists several questions raised by our customers regarding the working of the transparent clock in our PHYs and the solutions suggested by our support team.

Table 1 • Solutions

Question	Answer
Can we enable the transparent clock and master on the same port?	Yes, the transparent clock and master can be enabled on the same port. However, the actions should be shared by two channels using the same 1588 block. This can be done by keeping the two clocks in different domain, as shown in the previous section.
Can Microsemi PHYs ignore calculation of the correction field based on some bit in the reserve field of the PTP header?	This can be done only through static configuration, not dynamically enabled and disabled based on packet content.

Question	Answer
For a port with transparent and boundary /ordinary clock defined on the same port, will the master be differentiated based on the source IP address?	When a port is configured for two clocks, the differentiation can be made based on the clock domain or port number, not on the IP address. The IP address is used for flow matching and drop decision, not to support 1588 actions. The 1588 actions are based on the configuration of the PTP header fields.

Microsemi Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996
Email: sales.support@microsemi.com
www.microsemi.com

© 2015–2018 Microsemi. All rights reserved. Microsemi and the Microsemi logo are trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners.

Microsemi makes no warranty, representation, or guarantee regarding the information contained herein or the suitability of its products and services for any particular purpose, nor does Microsemi assume any liability whatsoever arising out of the application or use of any product or circuit. The products sold hereunder and any other products sold by Microsemi have been subject to limited testing and should not be used in conjunction with mission-critical equipment or applications. Any performance specifications are believed to be reliable but are not verified, and Buyer must conduct and complete all performance and other testing of the products, alone and together with, or installed in, any end-products. Buyer shall not rely on any data and performance specifications or parameters provided by Microsemi. It is the Buyer's responsibility to independently determine suitability of any products and to test and verify the same. The information provided by Microsemi hereunder is provided "as is, where is" and with all faults, and the entire risk associated with such information is entirely with the Buyer. Microsemi does not grant, explicitly or implicitly, to any party any patent rights, licenses, or any other IP rights, whether with regard to such information itself or anything described by such information. Information provided in this document is proprietary to Microsemi, and Microsemi reserves the right to make any changes to the information in this document or to any products and services at any time without notice.

Microsemi, a wholly owned subsidiary of Microchip Technology Inc. (Nasdaq: MCHP), offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions; security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, California, and has approximately 4,800 employees globally. Learn more at www microsemi.com.

VPPD-04007