Freescale Semiconductor
Application Note

Document Number: AN3423
Rev. 0, 09/2010

Support for IEEE 1588™ Protocol in PowerQUICC and QorIQ Processors

by *Networking and Multimedia Group*
Freescale Semiconductor, Inc.
Austin, TX

The IEEE 1588™ protocol is used to synchronize the clocks of two systems. Unlike the network time protocol (NTP) which synchronizes with accuracy of few milliseconds, the IEEE 1588™ protocol can achieve synchronization accuracy within tens of nanoseconds. Such high accuracy is especially beneficial for control applications that need synchronized clocks for operation. The advantage of the IEEE 1588 protocol is that it can operate over a regular Ethernet communication stack, so it can synchronize clocks of systems that are physically separated by a reasonable distance. Detailed information on synchronization using the IEEE 1588 protocol is available in the protocol specification.

The Ethernet controller in PowerQUICC and QorIQ devices contains dedicated hardware for registering time-stamps of both incoming and outgoing Ethernet packets, which enables nanoseconds synchronization of clocks. This document describes the implementation of the IEEE 1588 protocol using the dedicated hardware in eTSEC- and dTSEC-based devices. The document does not cover IEEE 1588 protocol implementation in QUICC Engine block-based devices. For a detailed explanation of signals and registers, please refer to the reference manual of the particular device.

Contents

freescale™ semiconductor

1 Clock Synchronization Using the IEEE 1588™ Protocol

In the systems using IEEE 1588™, every slave synchronizes to it's master's clock by exchanging synchronization messages with the master. The synchronization process is divided into two phases: offset measurement and delay measurement.

In the offset measurement phase, the time difference between master and slave is corrected. During this phase, the master periodically transmits a unique synchronization (sync) message to the related slave clocks at defined intervals. This sync message contains an estimated value of the time at which this message was transmitted. The master clock measures the time of transmission (TM1) and the slave clocks measure the time of reception, TS1. The master sends the second message, the follow-up message, and the exact time of transmission of the corresponding sync message to the slave clocks.

Upon reception of the sync message and, for increased accuracy, on reception of the corresponding follow-up message, the slave clock calculates the correction (offset) in relation to the master clock while taking into account the reception time-stamp of the sync message. The slave clock, Ts, must then be corrected by this offset. If there is no delay over the transmission path, both clocks are synchronous.

Figure 1 illustrates this process.

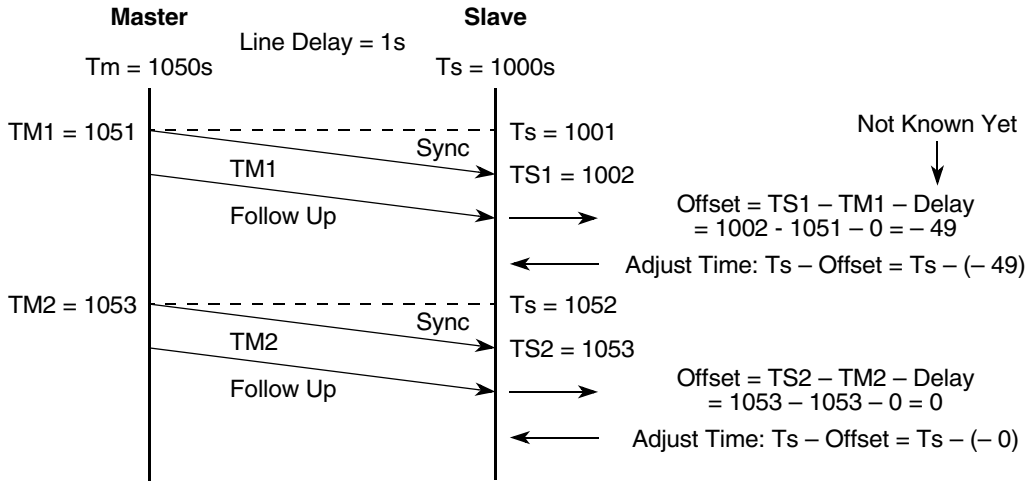

Figure 1. Offset Correction in the IEEE 1588 Protocol

The second phase of the synchronization process is the delay measurement that determines the delay or latency between slave and master. For this purpose, the slave clock sends a delay request packet to the master and records the time of transmission of this message (TS3). When the master receives the packet, it generates a time-stamp and sends the time of reception (TM3) back to the slave in a delay response packet. The slave calculates the delay time between itself and master based on the following information:

- Local time-stamp for transmission
- Time-stamp for reception provided by the master
- Local reception time

This calculation assumes a symmetrical propagation delay—that is, the time for propagation from slave to master is the same as the propagation time from master to slave. Figure 2 illustrates this process.

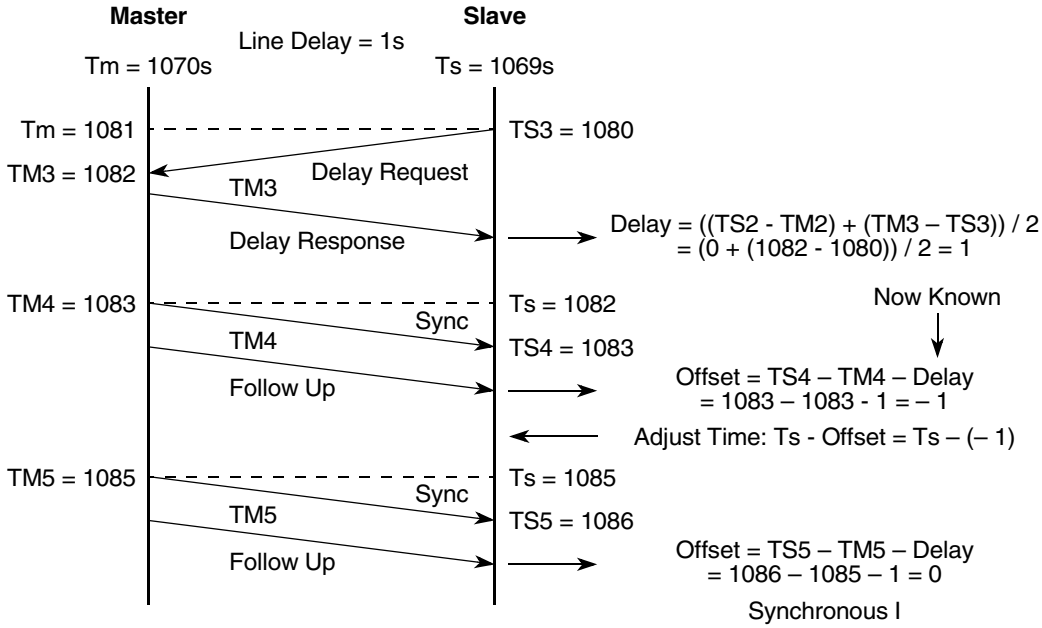

Figure 2. Delay Measurement in the IEEE 1588 Protocol

2 Hardware Support for the IEEE 1588 Protocol

PowerQUICC and QorIQ processors contain a dedicated hardware block for supporting the IEEE 1588 protocol. This section explains the working of the hardware block and its functional usage.

2.1 IEEE 1588 Protocol Registers

Table 1 shows the details of the IEEE 1588 Protocol registers used in PowerQUICC and QorIQ devices, which are discussed subsequently. For the most up to date information, refer to the reference manual of the particular device.

Table 1. IEEE 1588 Registers

Sr. No.	Name	Description
1.	TMR_CTRL	Timer control register
2.	TMR_EVENT	Timer event register
3.	TMR_TEMASK	Timer event mask register
4.	TMR_PEVENT	Timer PTP packet event register
5.	TMR_PEMASK	Timer PTP packet event mask register
6.	TMR_STAT	Timer status register
7.	TMR_CNT_H/L	Timer counter register
8.	TMR_ADD	Timer drift compensation addend register

Table 1. IEEE 1588 Registers (continued)

Sr. No.	Name	Description
9.	TMR_ACC	Timer accumulator register
10.	TMR_PRSC	Timer prescaler register
12.	TMROFF_H/L	Timer offset register
13.	TMR_ALARM1-2_H/L	Alarm time comparator register
13.	TMR_FIPER1-3	Timer fixed interval periodic register
14.	TMR_ETTS1-2_H/L	External trigger stamp register
15.	TMR_TXTS1-2_ID	Transmit time-stamp identification register
16.	TMR_TXTS1-2_H/L	Transmit time-stamp register

2.2 IEEE 1588 Protocol Clock

The IEEE 1588 protocol clock can use either the internal or external reference clock source. The hardware support is built around an accumulator. The input clock can be divided using the accumulator and ADDEND to the desired nominal frequency. For example, if the frequency of the input clock is 150 MHz and the desired nominal frequency is 100 MHz, the value of ADDEND can be calculated as $2^{32} \div$ FreqDivRatio. The frequency division ratio in this case is 1.5 (150 ÷ 100) and hence the ADDEND would be equal to 0xAAAA_AAAA.

As shown in Figure 3, the value of ADDEND, to be program in TMR_ADD, is added to the accumulator on every input clock, and the overflow bit of the accumulator is used to increment the clock counter (TMR_CNT). If ADDEND is equal to 0xAAAA_AAAA, the overflow bit of accumulator would be generated 100,000,000 times if ADDEND gets added to accumulator 150,000,000 times and hence the division.

TCLK_PERIOD can be derived from the nominal clock. A value of 10 in TCLK_PERIOD, when nominal frequency is 100 MHz, would ensure that the clock counter (TMR_CNT) has a granularity of nanoseconds.

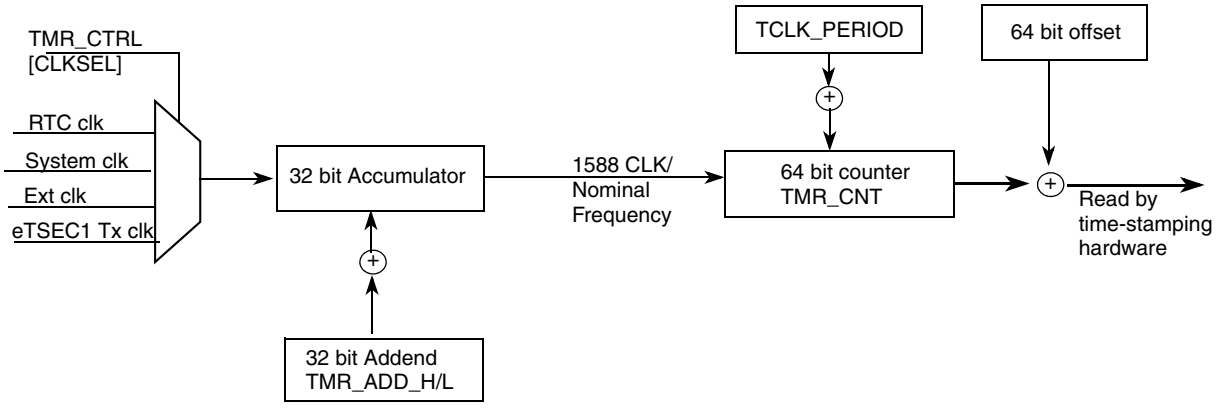

Figure 3. IEEE 1588 Protocol Basic Clock Diagram

In order to synchronize, the Precision Time Protocol (PTP) application running on the PTP slave, may change the value of TMR_CNT when the difference between the master and slave is large. When the difference is small, applications may tweak ADDEND to make nominal clock slower or faster.

2.3 Generating FIPER and ALARM

After the clocks are synchronized, some systems may need phase-aligned pulses to drive other peripherals. The fixed interval period register (FIPER) can be used to generate periodic pulses. A down count register loads the value programmed in the FIPER. The FIPER register must be programmed before the timer is enabled. At every tick of the timer accumulator overflow, the counter decrements by the value of TMR_CTRL[TCLK_PERIOD]. It generates a pulse when the down counter value reaches zero. It reloads the down counter in the cycle following a pulse.

As shown in Figure 4, FIPER gets decremented by the TCLK_PERIOD on every pulse of the nominal clock. This implies that FIPER and TMR_CNT decrements at the same pace, however they are not aligned. FIPER1 can be aligned with TMR_CNT by using ALARM1 a trigger. Other FIPERs do not have this functionality.

For example, if a user wish to use the FIPER1 register to generate a 1 PPS (pulse per second) event aligned with TMR_CNT, the following procedure should be used:

1. Program TMR_FIPER1 to a value that will generate a pulse every second.
2. To trigger FIPER1 by ALARM1, set TMR_CTRL[FS].
3. Program ALARM1 to a value which is a whole number of seconds, and greater than the present TMR_CNT. In the formula below, *n* seconds are added to TMR_CNT. The sum is divided by 10^9 and the integer quotient results in a whole number of seconds.

$$TMR\_ALARM1 = floor\{(TMR\_CNT + n)/ 10^9\} \quad \textit{Eqn. 1}$$

   For example, if TMR_CNT = 5.3 seconds and n = 2 seconds,

   TMR_ALARM1 = floor($7.3*10^9$, 10^9) = 7 seconds

4. Enable the timer

The hardware waits for TMR_ALARM1 to expire before enabling the count down of TMR_FIPER1. The end result is that TMR_FIPER1 pulses every second after the original timer ALARM1 expires.

If the PPS signals need to be phase-aligned to the prescale output clock, the alarm value should be programmed 1 clock period less than the desired value.

To track the prescale output clock, users must the FIPER by writing a new value to the register before enabling the FIPER. The FIPER value can be calculated as shown in the following equation:

$$Value\ of\ FIPER = (prescale\_value \times tclk\_per \times N) - tclk\_per \quad \textit{Eqn. 2}$$

For example, if prescale = 9 and clock period = 10, the FIPER can obtain the following values: 80, 170, 260.....

Figure 4 shows the FIPER and ALARM.

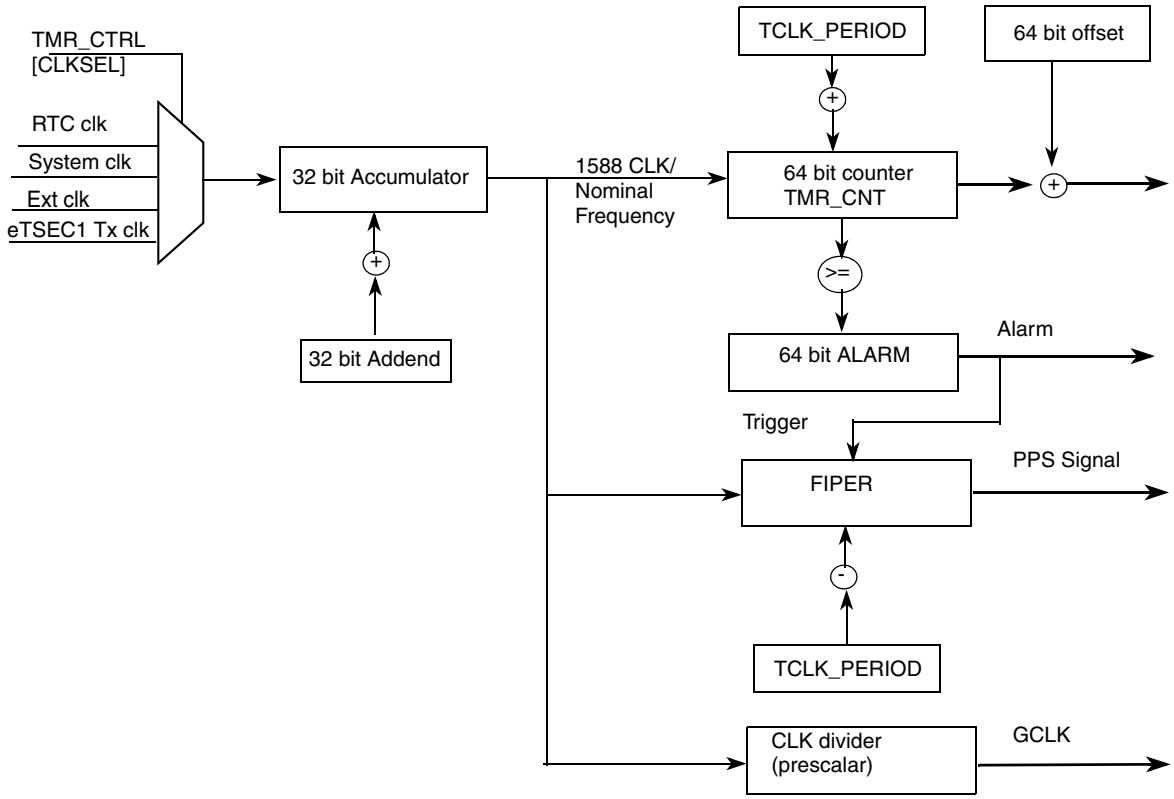

Figure 4. FIPER and ALARM

NOTE

If an application chooses to change the value of TMR_CNT for synchronization, the FIPER should be stopped. Value of ALARM1 should be re-calculated to trigger FIPER1 again.

2.4 Hardware Time-Stamp Mechanism

Precision time-stamping is achieved by time-stamping the packets as soon as they are received on the receiver side and as soon as they are transmitted from the transmitter side. As a prerequisite, TMR_CTRL[TE] should be set to 1 to enable the IEEE 1588 protocol timer.

2.4.1 Receive Time-Stamping

To enable time-stamping on the receive side, RCTRL[TS] in eTSEC (RCTRL[RTSE] in dTSEC) should be set to 1. As soon as the MAC receives a packet, the value of TMR_CNT is copied to TMR_RXTS_H/L. This time-stamp is also inserted into the packet data buffer as padding alignment bytes if TMR_CTRL[RTPE] is set to 1.

Time-stamp insertion into the data buffer requires receive pad alignment length (RCTRL[PAL]) to be set to a value greater than or equal to 8. Retrieving time-stamps from the packet data itself is easier and faster than retrieving time-stamps stored in registers. This solution easily scales to handle a large number of received PTP packets even when the CPU cannot process a PTP packet before the next PTP packet arrives.

2.4.2 Transmit Time-Stamping

To enable time-stamping of a particular packet on transmit side, TxFCB[PTP] in eTSEC should be set to 1. In dTSEC, setting TCTRL[TTSE] to 1 ensures that all the packets will be time-stamped during transmission. On the TX side, the MAC time-stamps the packet when it starts transmission of the packet. The packet ID and time-stamp are stored in the TMR_TXTS1-2_ID and TMR_TXTS1-2_H/L registers, respectively. Software can probe the ID registers to retrieve the correct time-stamp of the outgoing packet when requested.

The hardware can also be configured to store transmission time-stamps in the padding alignment bytes (PAL) located between the transmission frame buffer and the frame data. In the eTSEC, it is required that a minimum of two TxBDs are used per packet. The first points to the start of the 8 byte TxFCB. The second points to the start of frame data. In memory, the TxFCB, and at least the first 16 bytes of the TxPAL must be located in contiguous memory locations. The first TxBD[TOE] is set. When TMR_CTRL[RTPE] and TxFCB[PTP] are set, the time-stamp is written to memory location TxBD[Data Buffer Pointer]+16.

NOTE

It is recommended that time-stamps written on the buffer be used instead of the registers to ensure accuracy even in high traffic environments.

3 The IEEE 1588 Protocol Driver

The lowest level software should take care of the hardware initialization, retrieval of time-stamps, and providing a bridge to upper level application. The lowest level software is often part of the Ethernet driver. This section explains the functions of the lowest level driver available in board support packages distributed by Freescale.

3.1 Initialization

The procedure to initialize the IEEE 1588 protocol hardware is as follows:

1. Select input clock by TMR_CTRL[CKSEL].
2. Initialize TMR_PRSC with the desired prescalar value.
3. TMR_ADD and TCLK_PERIOD should be initialized as described in Section 2.2, “IEEE 1588 Protocol Clock.”
4. If PPS is required, FIPER should be equal to $10^9 \div$ TCLK_PERIOD.
5. Setting TMR_CTRL[FS] to 1 ensures that FIPER1 is triggered by ALARM1.
6. ALARM should be few seconds ahead of TMR_CNT to make sure alarm is hit at the boundary of a second to trigger FIPER.

7. RCTRL[TS] in eTSEC or RCTRL[RTSE] in dTSEC should be set to 1 for time-stamp on receive side.
8. To enable time-stamping of a packet to be transmitted, TxFCB[PTP] in eTSEC or TCTRL[TTSE] in dTSEC should be set to 1.

As mentioned in Section 2.4, "Hardware Time-Stamp Mechanism," the time-stamps can be retrieved either through time-stamp registers or the frame control buffers. If the user wants to retrieve time-stamps through registers, software should initialize buffers to store time-stamps and packet identification numbers.

3.2 Recording Time-Stamps

During transmission, the driver should check if the packet to be transmitted is a PTP packet. If yes, the driver should add padding as described in Section 2.4.2, "Transmit Time-Stamping." After a packet is transmitted, the driver can store the time-stamp and packet identification number.

Upon receiving a PTP packet, the driver stores the time-stamp with a packet identification number in a circular buffer. A user-space application using IOCTL can later retrieve the time-stamp.

3.3 Application Programming Interface

The lower level driver interacts with the hardware on one end and higher level applications at the other. The higher level application, in this case, implements precision time protocol (PTP). This section discusses the IOCTLs implemented by a low level driver in Linux-based board support packages (BSP) distributed by Freescale.

- PTP_GET_RX_TIMESTAMP_SYNC

  The user-space application provides the message ID of SYNC message, for which time-stamp is to be retrieved, to the IOCTL. The driver returns the time-stamp. Retrieval of a particular time-stamp also ensures that any preceding time-stamps are automatically cleared. For example, suppose the buffer inside kernel currently has time-stamps for SYNC message IDs from 4–22 and the application requests the time-stamp for SYNC message ID 20. The kernel-space driver returns the time-stamp of ID 20 and also clears the time-stamps from values 4–20.
- PTP_GET_RX_TIMESTAMP_DEL_REQ

  The user-space application provides the message ID of Delay_Request message, for which time-stamp is to be retrieved. The IOCTL returns the time-stamp as a 64-bit value. The buffer is automatically cleared as mentioned above for SYNC messages.
- PTP_GET_TX_TIMESTAMP

  The user-space application calls this IOCTL without providing any message ID. The kernel-space driver returns the last recorded TX time-stamp to the application.
- PTP_ADJ_ADDEND

  This IOCTL is used by user-space application to program ADDEND registers.
- PTP_GET_ADDEND

  The user-space application may use this to read the current value of ADDEND.
- PTP_GET_CNT

The user-space application calls PTP_GET_CNT to retrieve the current value of TMR_CNT_H/L registers. This is provided back to the user-space application as a 64-bit value.

- PTP_SET_CNT

  The user-space application provides a 64-bit value to be loaded into the TMR_CNT_H/L registers. As discussed in Section 2.3, "Generating FIPER and ALARM," every time there is a change in TMR_CNT_H/L, FIPER should be stopped and value of ALARM should be recalculated to trigger FIPER.

4 References

The following list contains documents for further reference.

- MPC8313E PowerQUICC™ II Pro Integrated Host Processor Reference Manual
- P4080 QorIQ Integrated Multicore Communication Processor Reference Manual
- IEEE 1588 Precise Time Synchronization, available at http://www.ieee1588.com/Synchronisation.htm

5 Revision History

Table 2 provides a revision history for this application note.

Table 2. Document Revision History

Rev. Number	Date	Substantive Change(s)
0	09/2010	Initial release

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
1-800-521-6274 or
+1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku
Tokyo 153-0064
Japan
0120 191014 or
+81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor
Literature Distribution Center
1-800 441-2447 or
+1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor
@hibbertgroup.com

Document Number: AN3423
Rev. 0
09/2010