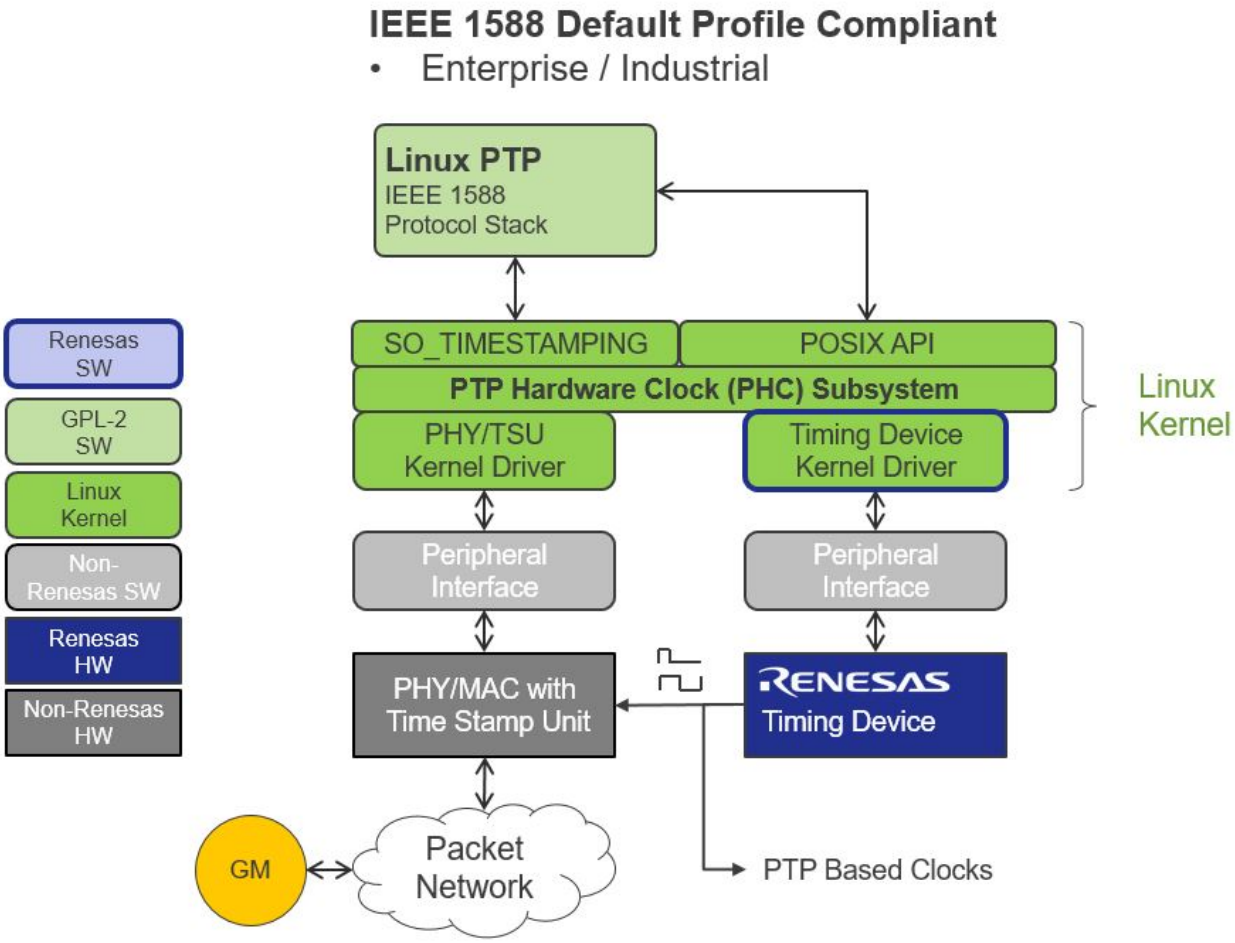
Linux PTP Using PHC Adjust Phase

This document provides instructions for getting Linux PTP to use the PTP hardware clock adjust phase mode.

Contents

1. Introduction

The Linux PTP project is an implementation of the Precision Time Protocol (PTP) according to IEEE standard 1588 for Linux licensed under the GNU General Public License.



Linux PTP supports both hardware and software time stamping via the Linux SO_TIMESTAMPING socket option.

Hardware time stamping is used because it provides better accuracy by time stamping packets at the exact moment they are sent and received. Hardware time stamping requires hardware PTP support from the PHY/MAC.

Some PTP hardware clocks have a adjust phase mode that has a built-in hardware filtering capability. The adjust phase mode uses a phase offset control word instead of a frequency offset control word.

Adjust phase support was introduced into the Linux PHC subsystem in Linux kernel v5.8.

Linux PTP v3.0 added ts2phc that is used to align the PHC and time stamper when the PHC and time stamper are not the same device.

2. System Requirements

- Linux PTP v3.0+
- Linux kernel v5.8+
- Network interface
  - Implements SIOCETHTOOL
  - Supports PTP hardware clock (PHC)
  - Pass PTP Ethernet packet time stamps using the SO_TIMESTAMPING API
- Linux PTP Hardware Clock driver
  - Renesas timing device that supports adjust phase
  - ClockMatrix PHC driver with adjust phase was introduced in Linux kernel v5.8
  - Driver code can be backported to Linux kernel v3.0+ that is patched to support adjust phase

2.1 Linux PTP Source

Instructions on cloning and compiling Linux PTP from source can be found below. Linux PTP added support for adjust phase in 2020-05-04 commit 7df88a.

Host sites:

- http://linuxptp.sourceforge.net
- https://sourceforge.net/p/linuxptp/code/ci/master/tree
- https://sourceforge.net/p/linuxptp/code/ci/v3.1/tree

Mirror site:

- https://github.com/richardcochran/linuxptp

2.2 Linux Kernel

Host sites:

- https://github.com/torvalds/linux
- https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git

Some PTP hardware clocks have an adjust phase mode that has a built-in hardware filtering capability. The adjust phase mode uses a phase offset control word instead of a frequency offset control word.

The Linux PHC subsystem added the capability to accept phase offset control word in Linux kernel v5.8.

Linux PTP v3.0 has also added support for adjust phase (i.e., write_phase_mode configuration parameter).

If you are using Linux kernel v3.x to v5.7, you will need to backport the adjust phase support into the kernel.

2.2.1. Adjust Phase Patch Files

The following patch files are the differences between Linux kernel v5.8-rc5 and v5.6.

Files affected:

- drivers/ptp/ptp_chardev.c
- drivers/ptp/ptp_clock.c
- include/linux/ptp_clock_kernel.h
- include/uapi/linux/ptp_clock.h
- tools/testing/selftests/ptp/testptp.c

2.2.1.1. Patch file for Linux v5.6

Copy and paste the following and save as a text file on the root directory of your linux kernel v5.6.

```
From 2ae6eb4fa4a7d7499322e68fc17fae14f35bfc1b Mon Sep 17 00:00:00 2001
From: Vincent Cheng <vincent.cheng.xh@renesas.com>
Date: Fri, 17 Jul 2020 16:50:20 -0400
Subject: [PATCH] adjust phase patch for linux v5.6

---
 drivers/ptp/ptp_chardev.c             |  1 +
 drivers/ptp/ptp_clock.c               |  9 +++++++++
 include/linux/ptp_clock_kernel.h      | 14 ++++++++++-----
 include/uapi/linux/ptp_clock.h        |  4 +++-
 tools/testing/selftests/ptp/testptp.c |  6 ++++--
 5 files changed, 26 insertions(+), 8 deletions(-)

diff --git a/drivers/ptp/ptp_chardev.c b/drivers/ptp/ptp_chardev.c
index 9d72ab5..4c9fc5c 100644
--- a/drivers/ptp/ptp_chardev.c
+++ b/drivers/ptp/ptp_chardev.c
@@ -133,12 +133,13 @@ long ptp_ioctl(struct posix_clock *pc, unsigned int cmd, unsigned long
arg)
      caps.n_alarm = ptp->info->n_alarm;
      caps.n_ext_ts = ptp->info->n_ext_ts;
      caps.n_per_out = ptp->info->n_per_out;
      caps.pps = ptp->info->pps;
      caps.n_pins = ptp->info->n_pins;
      caps.cross_timestamping = ptp->info->getcrosststamp != NULL;
+     caps.adjust_phase = ptp->info->adjphase != NULL;
      if (copy_to_user((void __user *)arg, &caps, sizeof(caps)))
         err = -EFAULT;
      break;

   case PTP_EXTTS_REQUEST:
   case PTP_EXTTS_REQUEST2:
diff --git a/drivers/ptp/ptp_clock.c b/drivers/ptp/ptp_clock.c
index ac1f2bf..4fc6e4b 100644
--- a/drivers/ptp/ptp_clock.c
+++ b/drivers/ptp/ptp_clock.c
@@ -143,12 +143,21 @@ static int ptp_clock_adjtime(struct posix_clock *pc, struct
__kernel_timex *tx)
         return -ERANGE;
      if (ops->adjfine)
         err = ops->adjfine(ops, tx->freq);
      else
         err = ops->adjfreq(ops, ppb);
      ptp->dialed_frequency = tx->freq;
+  } else if (tx->modes & ADJ_OFFSET) {
+     if (ops->adjphase) {
+        s32 offset = tx->offset;
+
+        if (!(tx->modes & ADJ_NANO))
+           offset *= NSEC_PER_USEC;
+
+        err = ops->adjphase(ops, offset);
+     }
   } else if (tx->modes == 0) {
      tx->freq = ptp->dialed_frequency;
```

```
      err = 0;
   }

   return err;
diff --git a/include/linux/ptp_clock_kernel.h b/include/linux/ptp_clock_kernel.h
index c64a1ef..aa805f6 100644
--- a/include/linux/ptp_clock_kernel.h
+++ b/include/linux/ptp_clock_kernel.h
@@ -33,13 +33,13 @@ struct system_device_crosststamp;
 struct ptp_system_timestamp {
   struct timespec64 pre_ts;
   struct timespec64 post_ts;
 };

 /**
- * struct ptp_clock_info - decribes a PTP hardware clock
+ * struct ptp_clock_info - describes a PTP hardware clock
   *
   * @owner:     The clock driver should set to THIS_MODULE.
   * @name:      A short "friendly name" to identify the clock and to
   *             help distinguish PHY based devices from MAC based ones.
   *             The string is not meant to be a unique id.
   * @max_adj:   The maximum possible frequency adjustment, in parts per billon.
@@ -62,12 +62,15 @@ struct ptp_system_timestamp {
   * @adjfreq:  Adjusts the frequency of the hardware clock.
   *            This method is deprecated.  New drivers should implement
   *            the @adjfine method instead.
   *            parameter delta: Desired frequency offset from nominal frequency
   *            in parts per billion
   *
+ * @adjphase:  Adjusts the phase offset of the hardware clock.
+ *             parameter delta: Desired change in nanoseconds.
+ *
   * @adjtime:  Shifts the time of the hardware clock.
   *            parameter delta: Desired change in nanoseconds.
   *
   * @gettime64:  Reads the current time from the hardware clock.
   *              This method is deprecated.  New drivers should implement
   *              the @gettimex64 method instead.
@@ -102,16 +105,16 @@ struct ptp_system_timestamp {
   *            zero if the function can be assigned to this pin, and
   *            nonzero otherwise.
   *            parameter pin: index of the pin in question.
   *            parameter func: the desired function to use.
   *            parameter chan: the function channel index to use.
   *
- * @do_work:  Request driver to perform auxiliary (periodic) operations
- *         Driver should return delay of the next auxiliary work scheduling
- *         time (>=0) or negative value in case further scheduling
- *         is not required.
+ * @do_aux_work:  Request driver to perform auxiliary (periodic) operations
+ *                Driver should return delay of the next auxiliary work
+ *                scheduling time (>=0) or negative value in case further
+ *                scheduling is not required.
   *
   * Drivers should embed their ptp_clock_info within a private
   * structure, obtaining a reference to it using container_of().
   *
```

```
  * The callbacks must all return zero on success, non-zero otherwise.
  */
@@ -125,12 +128,13 @@ struct ptp_clock_info {
   int n_per_out;
   int n_pins;
   int pps;
   struct ptp_pin_desc *pin_config;
   int (*adjfine)(struct ptp_clock_info *ptp, long scaled_ppm);
   int (*adjfreq)(struct ptp_clock_info *ptp, s32 delta);
+  int (*adjphase)(struct ptp_clock_info *ptp, s32 phase);
   int (*adjtime)(struct ptp_clock_info *ptp, s64 delta);
   int (*gettime64)(struct ptp_clock_info *ptp, struct timespec64 *ts);
   int (*gettimex64)(struct ptp_clock_info *ptp, struct timespec64 *ts,
           struct ptp_system_timestamp *sts);
   int (*getcrosststamp)(struct ptp_clock_info *ptp,
              struct system_device_crosststamp *cts);
diff --git a/include/uapi/linux/ptp_clock.h b/include/uapi/linux/ptp_clock.h
index 9dc9d00..ff070aa 100644
--- a/include/uapi/linux/ptp_clock.h
+++ b/include/uapi/linux/ptp_clock.h
@@ -86,13 +86,15 @@ struct ptp_clock_caps {
   int n_ext_ts;  /* Number of external time stamp channels. */
   int n_per_out; /* Number of programmable periodic signals. */
   int pps;       /* Whether the clock supports a PPS callback. */
   int n_pins;    /* Number of input/output pins. */
   /* Whether the clock supports precise system-device cross timestamps */
   int cross_timestamping;
-  int rsv[13];   /* Reserved for future use. */
+  /* Whether the clock supports adjust phase */
+  int adjust_phase;
+  int rsv[12];   /* Reserved for future use. */
 };

 struct ptp_extts_request {
   unsigned int index;  /* Which channel to configure. */
   unsigned int flags;  /* Bit field for PTP_xxx flags. */
   unsigned int rsv[2]; /* Reserved for future use. */
diff --git a/tools/testing/selftests/ptp/testptp.c b/tools/testing/selftests/ptp/testptp.c
index c0dd102..da7a9dd 100644
--- a/tools/testing/selftests/ptp/testptp.c
+++ b/tools/testing/selftests/ptp/testptp.c
@@ -266,20 +266,22 @@ int main(int argc, char *argv[])
               "  %d maximum frequency adjustment (ppb)\n"
               "  %d programmable alarms\n"
               "  %d external time stamp channels\n"
               "  %d programmable periodic signals\n"
               "  %d pulse per second\n"
               "  %d programmable pins\n"
-              "  %d cross timestamping\n",
+              "  %d cross timestamping\n"
+              "  %d adjust_phase\n",
              caps.max_adj,
              caps.n_alarm,
              caps.n_ext_ts,
              caps.n_per_out,
              caps.pps,
              caps.n_pins,
-             caps.cross_timestamping);
+             caps.cross_timestamping,
+             caps.adjust_phase);
```

```
+              caps.cross_timestamping,
+              caps.adjust_phase);
      }
   }

   if (0x7fffffff != adjfreq) {
      memset(&tx, 0, sizeof(tx));
      tx.modes = ADJ_FREQUENCY;
--
2.7.4
```

To apply the patch file,

```
$ git am --signoff < 0001-adjust-phase-patch-for-linux-v5.6.patch

Applying: Adjust phase patch for linux v5.6
$ git log
commit d6e62c17b920a4425395fdacc7c174be955e7568
Author: Vincent Cheng <vincent.cheng.xh@renesas.com>
Date: Fri Jul 17 16:50:20 2020 -0400
adjust phase patch for linux v5.6
Signed-off-by: Vincent Cheng <vincent.cheng.xh@renesas.com>

commit 7111951b8d4973bda27ff663f2cf18b663d15b48
Author: Linus Torvalds <torvalds@linux-foundation.org>
Date: Sun Mar 29 15:25:41 2020 -0700
Linux 5.6
Or patch the files and manually commit the changes afterwards.
$ patch -p1 < 0001-adjust-phase-patch-for-linux-v5.6.patch
patching file drivers/ptp/ptp_chardev.c
patching file drivers/ptp/ptp_clock.c
patching file include/linux/ptp_clock_kernel.h
patching file include/uapi/linux/ptp_clock.h
patching file tools/testing/selftests/ptp/testptp.c

$ git status
On branch tmp
Changes not staged for commit:
(use "git add <file>..." to update what will be committed)
(use "git checkout -- <file>..." to discard changes in working directory)
modified: drivers/ptp/ptp_chardev.c
modified: drivers/ptp/ptp_clock.c
modified: include/linux/ptp_clock_kernel.h
modified: include/uapi/linux/ptp_clock.h
modified: tools/testing/selftests/ptp/testptp.c
no changes added to commit (use "git add" and/or "git commit -a")
```

2.2.2. Backporting to Linux v3.x+

For backporting adjust phase changes into other linux kernel versions, examine the following individual patch files and manually make the corresponding appropriate changes for your linux kernel version.

2.2.2.1. drivers/ptp/ptp_chardev.c

```
--- a/drivers/ptp/ptp_chardev.c
+++ b/drivers/ptp/ptp_chardev.c
@@ -133,12 +133,13 @@ long ptp_ioctl(struct posix_clock *pc, unsigned int cmd, unsigned long
arg)
      caps.n_alarm = ptp->info->n_alarm;
```

```
      caps.n_ext_ts = ptp->info->n_ext_ts;
      caps.n_per_out = ptp->info->n_per_out;
      caps.pps = ptp->info->pps;
      caps.n_pins = ptp->info->n_pins;
      caps.cross_timestamping = ptp->info->getcrosststamp != NULL;
+     caps.adjust_phase = ptp->info->adjphase != NULL;
      if (copy_to_user((void __user *)arg, &caps, sizeof(caps)))
         err = -EFAULT;
      break;

   case PTP_EXTTS_REQUEST:
   case PTP_EXTTS_REQUEST2:
```

2.2.2.2. drivers/ptp/ptp_clock.c

```
--- a/drivers/ptp/ptp_clock.c
+++ b/drivers/ptp/ptp_clock.c
@@ -143,12 +143,21 @@ static int ptp_clock_adjtime(struct posix_clock *pc, struct
__kernel_timex *tx)
         return -ERANGE;
      if (ops->adjfine)
         err = ops->adjfine(ops, tx->freq);
      else
         err = ops->adjfreq(ops, ppb);
      ptp->dialed_frequency = tx->freq;
+  } else if (tx->modes & ADJ_OFFSET) {
+     if (ops->adjphase) {
+        s32 offset = tx->offset;
+
+        if (!(tx->modes & ADJ_NANO))
+           offset *= NSEC_PER_USEC;
+
+        err = ops->adjphase(ops, offset);
+     }
   } else if (tx->modes == 0) {
      tx->freq = ptp->dialed_frequency;
      err = 0;
   }

   return err;
```

2.2.2.3. include/linux/ptp_clock_kernel.h

```
--- a/include/linux/ptp_clock_kernel.h
+++ b/include/linux/ptp_clock_kernel.h
@@ -125,12 +128,13 @@ struct ptp_clock_info {
   int n_per_out;
   int n_pins;
   int pps;
   struct ptp_pin_desc *pin_config;
   int (*adjfine)(struct ptp_clock_info *ptp, long scaled_ppm);
   int (*adjfreq)(struct ptp_clock_info *ptp, s32 delta);
+  int (*adjphase)(struct ptp_clock_info *ptp, s32 phase);
   int (*adjtime)(struct ptp_clock_info *ptp, s64 delta);
   int (*gettime64)(struct ptp_clock_info *ptp, struct timespec64 *ts);
   int (*gettimex64)(struct ptp_clock_info *ptp, struct timespec64 *ts,
           struct ptp_system_timestamp *sts);
   int (*getcrosststamp)(struct ptp_clock_info *ptp,
```

```
              struct system_device_crosststamp *cts);
```

2.2.2.4. include/uapi/linux/ptp_clock.h

```
--- a/include/uapi/linux/ptp_clock.h
+++ b/include/uapi/linux/ptp_clock.h
@@ -86,13 +86,15 @@ struct ptp_clock_caps {
   int n_ext_ts;  /* Number of external time stamp channels. */
   int n_per_out; /* Number of programmable periodic signals. */
   int pps;       /* Whether the clock supports a PPS callback. */
   int n_pins;    /* Number of input/output pins. */
   /* Whether the clock supports precise system-device cross timestamps */
   int cross_timestamping;
-  int rsv[13];   /* Reserved for future use. */
+  /* Whether the clock supports adjust phase */
+  int adjust_phase;
+  int rsv[12];   /* Reserved for future use. */
 };

 struct ptp_extts_request {
   unsigned int index;  /* Which channel to configure. */
   unsigned int flags;  /* Bit field for PTP_xxx flags. */
   unsigned int rsv[2]; /* Reserved for future use. */
```

2.2.2.5. tools/testing/selftests/ptp/testptp.c

```
--- a/tools/testing/selftests/ptp/testptp.c
+++ b/tools/testing/selftests/ptp/testptp.c
@@ -266,20 +266,22 @@ int main(int argc, char *argv[])
                "  %d maximum frequency adjustment (ppb)\n"
                "  %d programmable alarms\n"
                "  %d external time stamp channels\n"
                "  %d programmable periodic signals\n"
                "  %d pulse per second\n"
                "  %d programmable pins\n"
-               "  %d cross timestamping\n",
+               "  %d cross timestamping\n"
+               "  %d adjust_phase\n",
                caps.max_adj,
                caps.n_alarm,
                caps.n_ext_ts,
                caps.n_per_out,
                caps.pps,
                caps.n_pins,
-               caps.cross_timestamping);
+               caps.cross_timestamping,
+               caps.adjust_phase);
      }
   }

   if (0x7fffffff != adjfreq) {
      memset(&tx, 0, sizeof(tx));
      tx.modes = ADJ_FREQUENCY;
```

2.3 Network Interface Requirements

The network interface driver would need to:

- Support ioctl command SIOCETHTOOL
- Support PTP hardware clock (PHC)
  - Pass PTP Ethernet packet time stamps using the SO_TIMESTAMPING API
  - Support POSIX clock API clock_adjtime() with struct timex modes
    - ADJ_FREQUENCY
    - ADJ_OFFSET
    - ADJ_NANO
    - ADJ_SETOFFSET
- Support PTP request type
  - PTP_CLK_REQ_PPS
  - PTP_CLK_REQ_EXTTS (PTP_ENABLE_FEATURE)

2.3.1. ethtool

`ethtool` can show whether a MAC supports hardware or software time stamping and whether it supports PHC. The following example output indicates support for hardware time stamping listed under `Capabilities` and is registered as PHC device 0, `PTP Hardware Clock: 0`.

```
$ ethtool -T eth0
Time stamping parameters for eth0:
Capabilities:
      hardware-transmit     (SOF_TIMESTAMPING_TX_HARDWARE)
      hardware-receive      (SOF_TIMESTAMPING_RX_HARDWARE)
      hardware-raw-clock    (SOF_TIMESTAMPING_RAW_HARDWARE)
PTP Hardware Clock: 0
Hardware Transmit Timestamp Modes:
      off                   (HWTSTAMP_TX_OFF)
      on                    (HWTSTAMP_TX_ON)
Hardware Receive Filter Modes:
      none                  (HWTSTAMP_FILTER_NONE)
      all                   (HWTSTAMP_FILTER_ALL)
```

If the ethtool utility is not on your system, you can compile the utility from the source.

https://git.kernel.org/pub/scm/network/ethtool/ethtool.git

Build ethtool from source example:

```
# Build ethtool from source
$ git clone git://git.kernel.org/pub/scm/network/ethtool/ethtool.git

$ cd ethtool

$ ./autogen.sh
$ ./configure

# CROSS compile for target board, ie. sudo apt-get install gcc-aarch64-linux-gnu
$ make clean all CC=/usr/bin/aarch64-linux-gnu-gcc

# Copy ethtool binary to target board
```

2.3.2. SIOCETHTOOL

Used in `linuxptp/sk.c` to interrogate network interface timestamping capabilities.

```
linuxptp/sk.c:
int sk_get_ts_info(const char *name, struct sk_ts_info *sk_info)
{
#ifdef ETHTOOL_GET_TS_INFO
  struct ethtool_ts_info info;
  ...

  info.cmd = ETHTOOL_GET_TS_INFO;

  ...

  err = ioctl(fd, SIOCETHTOOL, &ifr);
  if (err < 0) {
      pr_err("ioctl SIOCETHTOOL failed: %m");
      close(fd);
      goto failed;
…
include/uapi/linux/ethtool.h:
   #define ETHTOOL_GET_TS_INFO 0x00000041 /* Get time stamping and PHC info */
```

2.3.3. SO_TIMESTAMPING

https://www.kernel.org/doc/Documentation/networking/timestamping.txt

```
1. Control Interfaces

 The interfaces for receiving network packages timestamps are:
 ...
 * SO_TIMESTAMPING
   Generates timestamps on reception, transmission or both. Supports
   multiple timestamp sources, including hardware. Supports generating
   timestamps for stream sockets.

 ...

 1.3 SO_TIMESTAMPING (also SO_TIMESTAMPING_OLD and SO_TIMESTAMPING_NEW):

 Supports multiple types of timestamp requests. As a result, this
 socket option takes a bitmap of flags, not a boolean. In

   err = setsockopt(fd, SOL_SOCKET, SO_TIMESTAMPING, &val, sizeof(val));

 val is an integer with any of the following bits set. Setting other
 bit returns EINVAL and does not change the current state.

 The socket option configures timestamp generation for individual
 sk_buffs (1.3.1), timestamp reporting to the socket's error
 queue (1.3.2) and options (1.3.3). Timestamp generation can also
 be enabled for individual sendmsg calls using cmsg (1.3.4).


 1.3.1 Timestamp Generation

 Some bits are requests to the stack to try to generate timestamps. Any
 combination of them is valid. Changes to these bits apply to newly
 created packets, not to packets already in the stack. As a result, it
```

```
is possible to selectively request timestamps for a subset of packets
(e.g., for sampling) by embedding an send() call within two setsockopt
calls, one to enable timestamp generation and one to disable it.
Timestamps may also be generated for reasons other than being
requested by a particular socket, such as when receive timestamping is
enabled system wide, as explained earlier.

SOF_TIMESTAMPING_RX_HARDWARE:
  Request rx timestamps generated by the network adapter.

SOF_TIMESTAMPING_RX_SOFTWARE:
  Request rx timestamps when data enters the kernel. These timestamps
  are generated just after a device driver hands a packet to the
  kernel receive stack.

SOF_TIMESTAMPING_TX_HARDWARE:
  Request tx timestamps generated by the network adapter. This flag
  can be enabled via both socket options and control messages.

SOF_TIMESTAMPING_TX_SOFTWARE:
  Request tx timestamps when data leaves the kernel. These timestamps
  are generated in the device driver as close as possible, but always
  prior to, passing the packet to the network interface. Hence, they
  require driver support and may not be available for all devices.
  This flag can be enabled via both socket options and control messages.

...

1.3.4. Enabling timestamps via control messages

In addition to socket options, timestamp generation can be requested
per write via cmsg, only for SOF_TIMESTAMPING_TX_* (see Section 1.3.1).
Using this feature, applications can sample timestamps per sendmsg()
without paying the overhead of enabling and disabling timestamps via
setsockopt:

  struct msghdr *msg;
  ...
  cmsg                    = CMSG_FIRSTHDR(msg);
  cmsg->cmsg_level        = SOL_SOCKET;
  cmsg->cmsg_type         = SO_TIMESTAMPING;
  cmsg->cmsg_len          = CMSG_LEN(sizeof(__u32));
  *((__u32 *) CMSG_DATA(cmsg)) = SOF_TIMESTAMPING_TX_SCHED |
                SOF_TIMESTAMPING_TX_SOFTWARE |
                SOF_TIMESTAMPING_TX_ACK;
  err = sendmsg(fd, msg, 0);

The SOF_TIMESTAMPING_TX_* flags set via cmsg will override
the SOF_TIMESTAMPING_TX_* flags set via setsockopt.

Moreover, applications must still enable timestamp reporting via
setsockopt to receive timestamps:

  __u32 val = SOF_TIMESTAMPING_SOFTWARE |
          SOF_TIMESTAMPING_OPT_ID /* or any other flag */;
  err = setsockopt(fd, SOL_SOCKET, SO_TIMESTAMPING, &val, sizeof(val));

...
```

```
3.1 Hardware Timestamping Implementation: Device Drivers

A driver which supports hardware time stamping must support the
SIOCSHWTSTAMP ioctl and update the supplied struct hwtstamp_config with
the actual values as described in the section on SIOCSHWTSTAMP.  It
should also support SIOCGHWTSTAMP.

Time stamps for received packets must be stored in the skb. To get a pointer
to the shared time stamp structure of the skb call skb_hwtstamps(). Then
set the time stamps in the structure:

struct skb_shared_hwtstamps {
    /* hardware time stamp transformed into duration
     * since arbitrary point in time
     */
    ktime_t hwtstamp;
};

Time stamps for outgoing packets are to be generated as follows:
- In hard_start_xmit(), check if (skb_shinfo(skb)->tx_flags & SKBTX_HW_TSTAMP)
  is set no-zero. If yes, then the driver is expected to do hardware time
  stamping.
- If this is possible for the skb and requested, then declare
  that the driver is doing the time stamping by setting the flag
  SKBTX_IN_PROGRESS in skb_shinfo(skb)->tx_flags , e.g. with

      skb_shinfo(skb)->tx_flags |= SKBTX_IN_PROGRESS;

  You might want to keep a pointer to the associated skb for the next step
  and not free the skb. A driver not supporting hardware time stamping doesn't
  do that. A driver must never touch sk_buff::tstamp! It is used to store
  software generated time stamps by the network subsystem.
- Driver should call skb_tx_timestamp() as close to passing sk_buff to hardware
  as possible. skb_tx_timestamp() provides a software time stamp if requested
  and hardware timestamping is not possible (SKBTX_IN_PROGRESS not set).
- As soon as the driver has sent the packet and/or obtained a
  hardware time stamp for it, it passes the time stamp back by
  calling skb_hwtstamp_tx() with the original skb, the raw
  hardware time stamp. skb_hwtstamp_tx() clones the original skb and
  adds the timestamps, therefore the original skb has to be freed now.
  If obtaining the hardware time stamp somehow fails, then the driver
  should not fall back to software time stamping. The rationale is that
  this would occur at a later time in the processing pipeline than other
  software time stamping and therefore could lead to unexpected deltas
  between time stamps.
```

2.3.4. PTP_CLK_REQ_EXTTS

Can check if device supports `PTP_CLK_REQ_EXTTS` by using the `testptp` utility. For details on `testptp`, see [testptp](#).

For example, the device `/dev/ptp0` supports `PTP_CLK_REQ_EXTTS` because it says `1 external time stamp channels`.

```
# testptp -d /dev/ptp0 -c
capabilities:
  999999999 maximum frequency adjustment (ppb)
  0 programmable alarms
```

```
  1 external time stamp channels
  0 programmable periodic signals
  0 pulse per second
  0 programmable pins
  0 cross timestamping
```

2.4 Linux PTP Hardware Clock

Linux PTP works with devices with drivers that support the PTP hardware clock (PHC) infrastructure for Linux. Details on Linux PHC subsystem can be found at https://www.kernel.org/doc/Documentation/ptp/ptp.txt.

```
Clock drivers include include/linux/ptp_clock_kernel.h and register
themselves by presenting a 'struct ptp_clock_info' to the
registration method. Clock drivers must implement all of the
functions in the interface. If a clock does not offer a particular
ancillary feature, then the driver should just return -EOPNOTSUPP
from those functions.

Drivers must ensure that all of the methods in interface are
reentrant. Since most hardware implementations treat the time value
as a 64 bit integer accessed as two 32 bit registers, drivers
should use spin_lock_irqsave/spin_unlock_irqrestore to protect
against concurrent access. This locking cannot be accomplished in
class driver, since the lock may also be needed by the clock
driver's interrupt service routine.
```

For examples of drivers that support PHC, search for `ptp_clock_register` in the linux kernel source.

Sample of devices that support PHC API (Linux Kernel v5.7):

- drivers/net/ethernet/broadcom/bnx2x/bnx2x_main.c
- drivers/net/ethernet/broadcom/tg3.c
- drivers/net/ethernet/cadence/macb_ptp.c
- drivers/net/ethernet/cavium/common/cavium_ptp.c
- drivers/net/ethernet/freescale/fec_ptp.c
- drivers/net/ethernet/intel/e1000e/ptp.c
- drivers/net/ethernet/renesas/ravb_ptp.c
- drivers/net/phy/dp83640.c

Renesas Timing Devices with PHC driver support:

- drivers/ptp/ptp_clockmatrix.c
- drivers/ptp/ptp_idt82p33.c

2.4.1. ClockMatrix PHC Driver

The ClockMatrix™ family includes integrated devices that provide eight PLL channels. Each PLL channel can be independently configured as a frequency synthesizer, jitter attenuator, digitally controlled oscillator (DCO), or a digital phase lock loop (DPLL).

Typically these devices are used as timing references and clock sources for PTP applications.

2.4.1.1. Source Code

The ClockMatrix PHC driver with adjust phase support was introduced in Linux v5.8 kernel. The ClockMatrix PHC driver files are located in the Linux kernel ``drivers/ptp`` directory.

Latest driver code can be retrieved from the master branch of the linux kernel.

https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git/tree/drivers/ptp?h=master

Files of interest:

- Kconfig
- Makefile
- idt8a340_reg.h
- ptp_clockmatrix.h
- ptp_clockmatrix.c

Kconfig:

```
…
config PTP_1588_CLOCK_IDTCM
   tristate "IDT CLOCKMATRIX as PTP clock"
   depends on PTP_1588_CLOCK && I2C
   default n
   help
     This driver adds support for using IDT CLOCKMATRIX(TM) as a PTP
     clock. This clock is only useful if your time stamping MAC
     is connected to the IDT chip.

     To compile this driver as a module, choose M here: the module
     will be called ptp_clockmatrix.
…
```

Makefile:

```
…
obj-$(CONFIG_PTP_1588_CLOCK_IDTCM)  += ptp_clockmatrix.o
…
```

2.4.1.2. CM PHC Driver Release Notes

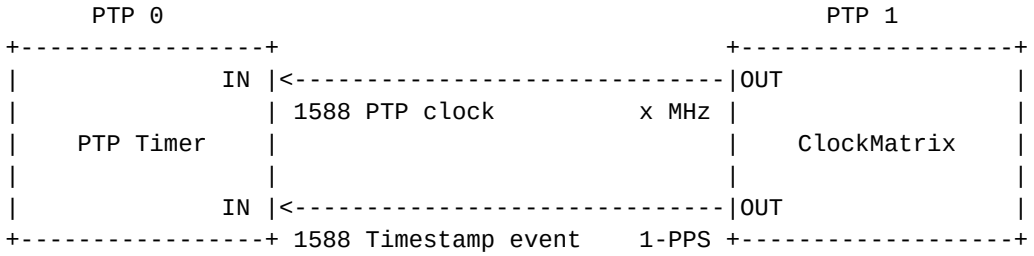
Kernel	Date	Description
5.12	2021-02-17	**Fixes**: Output to internal 1-PPS Alignment not complete when SYS_DPLL not locked. ▪ clean-up - parenthesis around a == b are unnecessary ▪ simplify code - remove unnecessary `err` variable ▪ coding style - tighten vertical spacing ▪ clean-up dev_*() messages ▪ add alignment of 1-PPS to idtcm_perout_enable ▪ add wait_for_sys_apll_dpll_lock.
5.11	2020-12-09	**Fixes**: 4.8.7 non-zero phase_adj is lost when driver starts ▪ deprecate firmware older than 4.8.7 ▪ fix non-zero phase_adj is lost after snap ▪ remove 5 second delay before entering write phase mode ▪ reset device and check BOOT_STATUS
5.9.12	2020-11-25 2020-08-19	**bug fix for idtcm_strverscmp** **Fixes:** I2C write bug for certain platforms ▪ use i2c_master_send for i2c write
5.9	2020-07-30	**Fixes**: Intermittent snap bug for 4.8.0 ▪ update to support 4.8.7 firmware

Kernel	Date	Description
5.8	2020-05-02 2020-04-25	Add adjphase() to support PHC write phase mode. Remove unnecessary comparison
5.6	2020-01-07 2020-01-02	Rework clockmatrix version information. Constify copied structure
5.5	2019-11-06 2019-11-03	Fix missing unlock on error in idtcm_probe() Add a ptp clock driver for IDT ClockMatrix

2.4.1.3. Device Connection

The ClockMatrix device is expected to output two signals to the PTP time stamping device.

- x MHz as 1588 PTP clock
- 1-PPS to align the rising clock edge of the `PTP Timer` with `ClockMatrix`




```
PTP time stamper driver supports: PTP_CLK_REQ_EXTTS
```

2.4.1.4. Kernel Device Tree

Insert ClockMatrix in the appropriate section of the device tree.

Sample entry on the Xilinx ZCU102 board:

```
…
i2c-mux@75 {
   i2c@1 {
      phc@5b { /* Clock Matrix */
         compatible = "idt,8a34000";
         reg = <0x5b>;
      };
   };
…
```

2.4.1.5. Kernel .config

```
#
# PTP clock support
#
CONFIG_PTP_1588_CLOCK=y

#
# Enable PHYLIB and NETWORK_PHY_TIMESTAMPING to see the additional clocks.
#
CONFIG_PTP_1588_CLOCK_IDTCM=m
```

2.4.1.6. Device Configuration

When the ClockMatrix driver is loaded, it looks in `/lib/firmware` a configuration file to load into the ClockMatrix device.

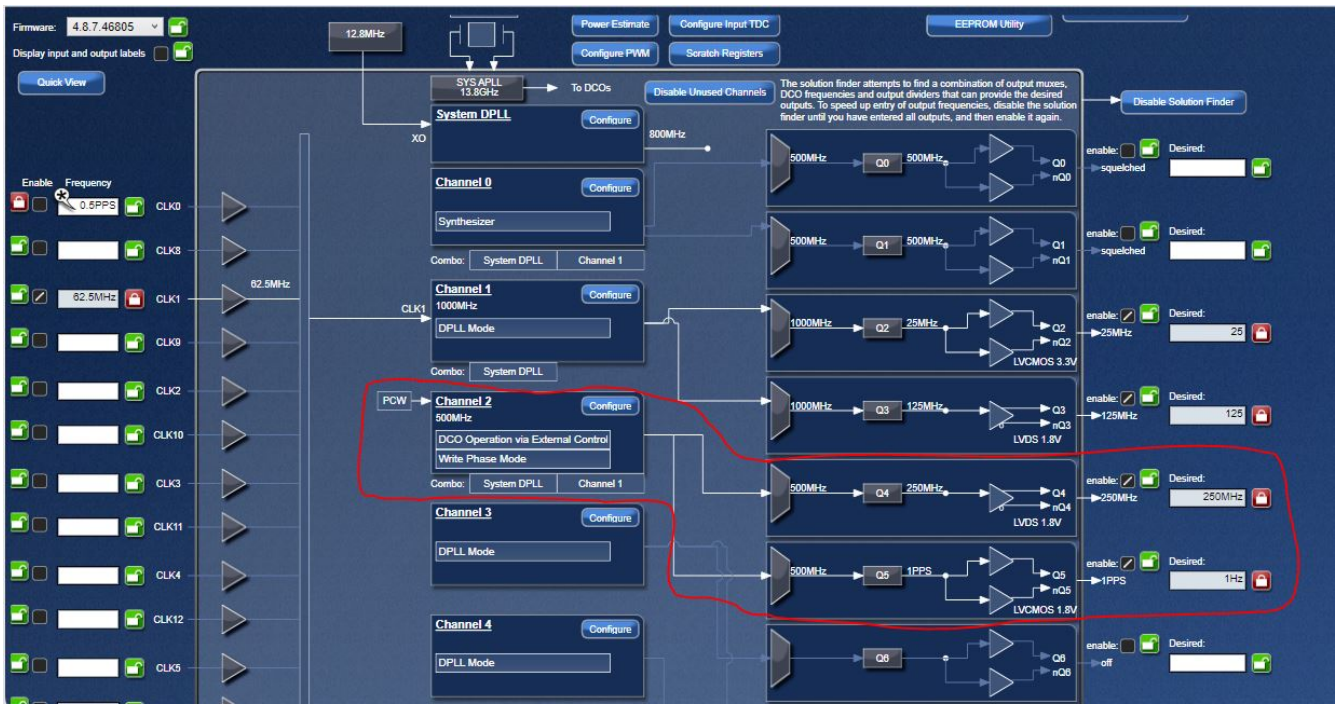
The filename is determined during compile time by `FW_FILENAME`:

```
static int idtcm_load_firmware(struct idtcm *idtcm,
                               struct device *dev)
{
    const struct firmware *fw;
    …
    err = request_firmware(&fw, FW_FILENAME, dev);
```

`FW_FILENAME` macro is defined in `drivers/ptp/ptp_clockmatrix.h`.

Renesas Timing Commander Software using Clockmatrix Timing Commander Personality v8.4. can be used to generate ClockMatrix device configuration `.bin` file.

```
#define FW_FILENAME  "idtcm.bin"
```



For device configuration support, email IDT-support-1588@lm.renesas.com.

3. Getting Started

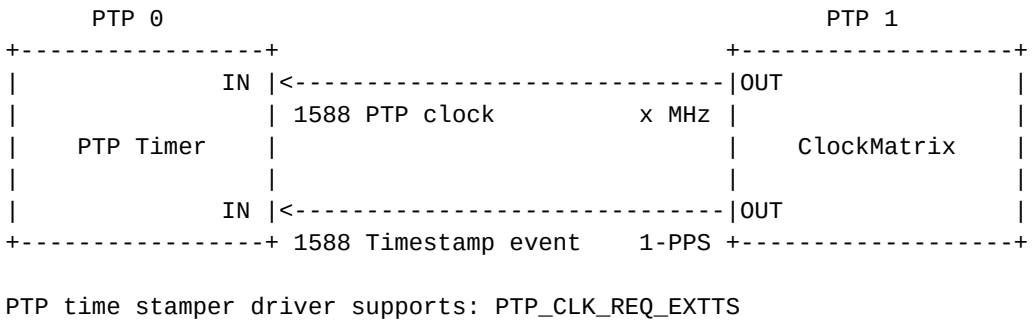
3.1 Sanity Testing

Use `testptp` and `phc_ctl` to check that the 1588 timestamp device and PTP clock device is connected and configured to behave as expected.

- Verify PHC capabilities
- Verify clock's time of day increments
- Verify clock set time
- Verify time stamper incoming 1-PPS from PTP clock
- Verify ts2phc aligns time stamper and 1-PPS from PTP clock
- Verify PTP clock time adjustment affects PTP timestamp
- Verify PTP clock frequency adjustment affects time stamper frequency

Some of the checks can be performed with `testptp` or `phc_ctl`. If the check passes using one of the programs, there is no need to repeat the same check using the other program.

The following example logs were captured using the Xilinx ZCU102 board with a ClockMatrix device as the PTP clock.




```
PTP time stamper driver supports: PTP_CLK_REQ_EXTTS
```

`/dev/ptp0`, Xilinx Timer, is the time stamper for 1588 packets.

`/dev/ptp1`, ClockMatrix, is the PTP clock source to the time stamper.

There is an expectation that the PTP timestamp with 0 nanoseconds aligns with the rising edge of outgoing 1-PPS clock signal.

3.1.1. Verify PHC Capabilities

For the Xilinx time stamper, the PHC driver must support at least one "external time stamp channels"..

For the ClockMatrix device, the PHC driver must support at least one "programmable periodic signals" to generate the 1-PPS to the Xilinx time stamper.

testptp

For Xilinx timer stamper:

```
$ ./testptp -d /dev/ptp0 -c
capabilities:
  999999999 maximum frequency adjustment (ppb)
  0 programmable alarms
  1 external time stamp channels
  0 programmable periodic signals
  0 pulse per second
  0 programmable pins
  0 cross timestamping
```

For ClockMatrix device:

```
$ ./testptp -d /dev/ptp1 -c
capabilities:
  244000 maximum frequency adjustment (ppb)
  0 programmable alarms
  0 external time stamp channels
  12 programmable periodic signals
  0 pulse per second
  0 programmable pins
  0 cross timestamping
```

phc_ctl

For Xilinx timer stamper:

```
$ ./phc_ctl /dev/ptp0 -- caps
phc_ctl[60.986]:
capabilities:
  999999999 maximum frequency adjustment (ppb)
  0 programable alarms
  1 external time stamp channels
  0 programmable periodic signals
  0 configurable input/output pins
  doesn't have pulse per second support
  doesn't have cross timestamping support
```

For ClockMatrix device:

```
$ ./phc_ctl /dev/ptp0 -- caps
phc_ctl[94.634]:
capabilities:
  244000 maximum frequency adjustment (ppb)
  0 programable alarms
  0 external time stamp channels
  12 programmable periodic signals
  0 configurable input/output pins
  doesn't have pulse per second support
  doesn't have cross timestamping support
```

3.1.2. Verify Clocks Increment

If the PTP device is working (i.e., the input clock is present), repeated command to get the clock time should show incremental readings.

testptp

For Xilinx timer stamper.

```
$./testptp -d /dev/ptp0 -g
clock time: 1595449857.649425272 or Wed Jul 22 20:30:57 2020

$ ./testptp -d /dev/ptp0 -g
clock time: 1595449859.400688308 or Wed Jul 22 20:30:59 2020
```

For ClockMatrix device:

```
$ ./testptp -d /dev/ptp1 -g
clock time: 1595449899.113274703 or Wed Jul 22 20:31:39 2020

$ ./testptp -d /dev/ptp1 -g
clock time: 1595449900.072974835 or Wed Jul 22 20:31:40 2020
```

phc_ctl

For Xilinx timer stamper.

```
$ ./phc_ctl /dev/ptp0 -- get
phc_ctl[1515.896]: clock time is 1524.402616468 or Thu Jan  1 00:25:24 1970

$ ./phc_ctl /dev/ptp0 -- get
phc_ctl[1533.458]: clock time is 1541.966759556 or Thu Jan  1 00:25:41 1970
```

For ClockMatrix device:

```
$ ./phc_ctl /dev/ptp1 -- get
phc_ctl[1643.300]: clock time is 1637.701499548 or Thu Jan  1 00:27:17 1970

$ ./phc_ctl /dev/ptp1 -- get
phc_ctl[1647.248]: clock time is 1641.649999488 or Thu Jan  1 00:27:21 1970
```

3.1.3. Verify Clock Set Time

Check clock set time was able set to 1000 seconds and then 0 seconds.

For Xilinx timer stamper.

```
$ ./phc_ctl /dev/ptp0 -- set 1000
phc_ctl[12839.455]: set clock time to 1000.000000000 or Thu Jan  1 00:16:40 1970

$ ./phc_ctl /dev/ptp0 -- get
phc_ctl[12842.198]: clock time is 1002.743910000 or Thu Jan  1 00:16:42 1970

$ ./phc_ctl /dev/ptp0 -- set 0
phc_ctl[12852.670]: set clock time to 0.000000000 or Thu Jan  1 00:00:00 1970

$ ./phc_ctl /dev/ptp0 -- get
phc_ctl[12854.174]: clock time is 1.504634712 or Thu Jan  1 00:00:01 1970
```

For ClockMatrix device:

```
$ ./phc_ctl /dev/ptp1 -- set 1000
phc_ctl[13015.464]: set clock time to 1000.000000000 or Thu Jan  1 00:16:40 1970

$ ./phc_ctl /dev/ptp1 -- get
phc_ctl[13022.248]: clock time is 1006.783599918 or Thu Jan  1 00:16:46 1970

$ ./phc_ctl /dev/ptp1 -- set 0
phc_ctl[13025.308]: set clock time to 0.000000000 or Thu Jan  1 00:00:00 1970

$ ./phc_ctl /dev/ptp1 -- get
phc_ctl[13026.747]: clock time is 1.439099534 or Thu Jan  1 00:00:01 1970
```

3.1.4. Verify Time Stamper Incoming 1-PPS from PTP Clock

Since the Xilinx timer should be receiving a 1-PPS input, it should show periodic 1 second events.

```
$ ./testptp -e 5000 -d /dev/ptp0
external time stamp request okay
event index 0 at 1797.085533392
event index 0 at 1798.085533392
event index 0 at 1799.085533392
event index 0 at 1800.085533392
event index 0 at 1801.085533392
event index 0 at 1802.085533392
```

```
event index 0 at 1803.085533392
```

There may be an accumulation of 1-PPS events, and you may see the display scroll rapidly. However, the 1-PPS event queue should flush relatively quickly and you should see the display incrementing every second.

To quit testptp loop, press Ctrl+C.

If the 1-PPS clock going into the Xilinx timer is missing, there will be no `event index` output after `external time stamp request okay` is displayed.

```
$ ./testptp -e 5000 -d /dev/ptp0
external time stamp request okay
```

To quit `testptp` loop, press Ctrl+C.

3.1.5. Verify ts2phc Aligns Time Stamper and 1-PPS from PTP Clock

1. Set time stamper time to 1000.5 seconds.
2. Set PHC time to 80000 seconds.
3. Run `ts2phc` with debug log enabled to align time stamper to PHC clock.

When using `ts2phc` with a ClockMatrix device, the message: `PTP_PIN_SETFUNC failed: Invalid argument` may pop up. The ClockMatrix PHC driver does not support dynamic `PTP_PIN_SETFUNC` yet. This message can be ignored because the ClockMatrix device is expected to be configured to output the expected 1-PPS during start-up.

```
$ cat ts2phc.cfg

#
# ts2phc config file to get it to behave like syncd to align
# time stamper to PHC device's 1-PPS signal.
#
# Example:
# ./ts2phc -m -q -f ts2phc.cfg
#

[global]
clock_servo nullf
first_step_threshold 0.000000001
step_threshold 0.000000001

# time stamper, slave device
[/dev/ptp0]
ts2phc.channel 0

# PHC device (ex. CM), master device
# Set ts2phc.channel to 2 for Sabre
[/dev/ptp1]
ts2phc.master  1
ts2phc.channel 5

#
# Set time stamper time to 1000.5 seconds.
#
$ ./phc_ctl /dev/ptp0 -- set 1000.5
phc_ctl[15170.818]: set clock time to 1000.500000000 or Thu Jan  1 00:16:40 1970

#
# Set PHC time to 80000 seconds.
```

```
#
$ ./phc_ctl /dev/ptp1 -- set 80000
phc_ctl[15188.347]: set clock time to 80000.000000000 or Thu Jan  1 22:13:20 1970

#
# Run ``ts2phc`` with debug log enabled to align time stamper to PHC clock.
#
$ ./ts2phc -mqf ts2phc.cfg -l 7
ts2phc[15208.103]: config item (null).message_tag is '(null)'
ts2phc[15208.104]: config item (null).verbose is 1
ts2phc[15208.104]: config item (null).use_syslog is 0
ts2phc[15208.104]: config item (null).logging_level is 7
ts2phc[15208.104]: config item /dev/ptp0.ts2phc.master is 0
ts2phc[15208.104]: config item (null).clock_servo is 3
ts2phc[15208.104]: config item /dev/ptp0.ts2phc.pin_index is 0
ts2phc[15208.104]: config item /dev/ptp0.ts2phc.channel is 0
ts2phc[15208.104]: config item /dev/ptp0.ts2phc.extts_polarity is 2
ts2phc[15208.104]: config item /dev/ptp0.ts2phc.extts_correction is 0
ts2phc[15208.104]: config item /dev/ptp0.ts2phc.pulsewidth is 500000000
ts2phc[15208.104]: config item (null).free_running is 0
ts2phc[15208.104]: PHC slave /dev/ptp0 has ptp index 0
ts2phc[15208.104]: config item (null).step_threshold is 0.000000
ts2phc[15208.104]: config item (null).first_step_threshold is 0.000000
ts2phc[15208.104]: config item (null).max_frequency is 900000000
ts2phc[15208.104]: config item (null).servo_offset_threshold is 0
ts2phc[15208.104]: config item (null).servo_num_offset_values is 10
ts2phc[15208.104]: config item /dev/ptp1.ts2phc.master is 1
ts2phc[15208.105]: PHC master /dev/ptp1 has ptp index 0
ts2phc[15208.105]: config item /dev/ptp1.ts2phc.channel is 5
ts2phc[15208.105]: config item /dev/ptp1.ts2phc.pin_index is 0
PTP_PIN_SETFUNC failed: Invalid argument
ts2phc[15208.105]: Failed to set the pin. Continuing bravely on...
ts2phc[15209.346]: /dev/ptp0 extts index 0 at 1039.030816420 corr 0 src 80021.400568 diff -78981969183580
ts2phc[15209.346]: /dev/ptp0 master offset -78981969183580 s1 freq+0
ts2phc[15210.346]: /dev/ptp0 extts index 0 at 80022.000000000 corr 0 src 80022.400672 diff 0
ts2phc[15210.346]: /dev/ptp0 master offset 0 s2 freq+0
ts2phc[15211.346]: /dev/ptp0 extts index 0 at 80023.000000000 corr 0 src 80023.400674 diff 0
ts2phc[15211.346]: /dev/ptp0 master offset 0 s2 freq+0
ts2phc[15212.346]: /dev/ptp0 extts index 0 at 80024.000000000 corr 0 src 80024.400728 diff 0
ts2phc[15212.346]: /dev/ptp0 master offset 0 s2 freq+0
ts2phc[15213.346]: /dev/ptp0 extts index 0 at 80025.000000000 corr 0 src 80025.300776 diff 0
ts2phc[15213.346]: /dev/ptp0 master offset 0 s2 freq+0
```

`s1` means unlock state and `s2` is the locked state (i.e., master and are aligned). The states in ts2phc are independent of the states in ptp4l.

Looking closer at the following lines:

```
PHC slave /dev/ptp0 has ptp index 0
...
PHC master /dev/ptp1 has ptp index 0
...
/dev/ptp0 extts index 0 at 1039.030816420 corr 0 src 80021.400568 diff -78981969183580
/dev/ptp0 master offset -78981969183580 s1 freq+0
/dev/ptp0 extts index 0 at 80022.000000000 corr 0 src 80022.400672 diff 0
```

`1039.030816420` is the time from `/dev/ptp0`, the PHC slave.

`80021.400568` is the time from `/dev/ptp1`, the PHC master.

`diff -78981969183580` or `master offset -78981969183580` is in nanoseconds, ie. `-78981.969183580` s.

To align the PHC slave, `ts2phc` will take the difference between the PHC masters and adjust the PHC slave accordingly.

Note after the adjustment, the `master offset` becomes `0`.

```
/dev/ptp0 master offset -78981969183580 s1 freq+0
...
/dev/ptp0 master offset 0 s2 freq+0
...
/dev/ptp0 master offset 0 s2 freq+0
```

When you stop `ts2phc`, it will disable the PTP clock's 1-PPS output. Running `ts2phc` will enable it again.

3.1.6. Verify PTP Clock Time Adjustment Affects PTP Timestamp

Prerequisites:

- PTP master configured for 1 packet per second internal
- Run ``ts2phc`` in the background
- Run ``ptp4l`` as a slave in free run mode in the background using the time stamper PHC (i.e. no -p option)

`ptp4l` can be configured into "`free run`" mode by adding "`free_running 1`" to the ptp4l configuration file.

Or by adding "`--free_running=1`" at the command line when starting `ptp4l` .

The plan is to monitor the master offset from `ptp4l` as we manually change the ToD counter on the PTP clock.

The `master offset` value in `ptp4l` output represents the measured offset from the master in nanoseconds.

```
$ cat debug2.cfg
[global]
domainNumber 4

# Announce messages
announceReceiptTimeout  2
logAnnounceInterval  1

# Sync/Delay_Req/Delay_Resp messages
# ex. 0 = 1-PPS, -3 = 8 PPS, -4 = 16 PPS
logSyncInterval  0
logMinDelayReqInterval  0

slaveOnly  1
masterOnly 0

clockClass 255

hybrid_e2e 1
inhibit_multicast_service 1
unicast_listen   1
unicast_req_duration  300

network_transport UDPv4

[unicast_master_table]
table_id 1
logQueryInterval  2
UDPv4 10.64.10.1

[eth0]
unicast_master_table 1

#
```

```
#  Run ts2phc in background to align PHC clock to time stamper
#
$ ./ts2phc -mq -f ts2phc.cfg&
[1] 4611
PTP_PIN_SETFUNC failed: Invalid argument
ts2phc[31300.670]: Failed to set the pin. Continuing bravely on...

#
# Run ptp4l in free run mode against a 1-PPS master to monitor
# the effect of stepping the PTP clock.
#
$ ./ptp4l -mqf debug2.cfg --free_running=1&

...
ptp4l[31666.715]: master offset -1614041727993934660 s0 freq  -68 path delay  7723
...

#
# Master offset is -1614041727993934660 ns, so need to adj +
# phc_ctl adj value is in seconds.
#
$ ./phc_ctl /dev/ptp1 -- adj 1614041727.993934660
phc_ctl[31763.871]: adjusted clock by 1614041727.993935 seconds


ptp4l[31766.706]: master offset-5636 s0 freq -304195 path delay  7719
ptp4l[31768.706]: master offset-5785 s0 freq  -72 path delay  7724
ptp4l[31770.706]: master offset-5900 s0 freq  -56 path delay  7727
```

If the initial offset master offset was positive, use a negative adj value.

```
...
ptp4l[32019.682]: master offset 199999266496145525 s0 freq     -48 path delay  7722

#
# Master offset is 199999266496145525 ns, so need to adj -
# phc_ctl adj value is in seconds.

$ ./phc_ctl /dev/ptp1 -- adj -199999266.496145525
phc_ctl[32075.860]: adjusted clock by -199999266.496146 seconds

ptp4l[32077.677]: master offset      -3266 s0 freq +100010 path delay  7725
ptp4l[32079.677]: master offset      -3359 s0 freq     -48 path delay  7722
ptp4l[32081.676]: master offset      -3487 s0 freq     -64 path delay  7722
```

3.1.7. Verify PTP Clock Frequency Adjustment Affects Time Stamper Frequency

Prerequisites:

- PTP master configured for 1 packet per second internal
- Run `ts2phc` in the background
- Run `ptp4l` as a slave in free run mode in the background using the time stamper PHC (i.e. no -p option)

`ptp4l` can be configured into "`free run`" mode by adding "`free_running 1`" to the `ptp4l` configuration file.

Or by adding "`--free_running=1`" at the command line when starting `ptp4l`.

The plan is to monitor the increase/decrease in the master offset from `ptp4l` as we manually change the frequency.

The `freq` value in `ptp4l` output represents the frequency adjustment of the clock in parts per billion, ppb, to converge to zero using pi_servo. However, we are running in free run so no adjustments are made

```
#
#  Run ts2phc in background to align PHC clock to time stamper
#
$ ./ts2phc -mq -f ts2phc.cfg&

#
# Run ptp4l in free run mode against a 1-PPS master to monitor
# the effect of stepping the PTP clock.
#
$ ./ptp4l -mqf debug2.cfg --free_running=1&

...
ptp4l[43236.627]: master offset -1614053583657415603 s0 freq  -60 path delay  7730
...

#
# Adjust master offset close to 0 so it is easier to see freq changes.
#
$ ./phc_ctl /dev/ptp1 -- adj 1614053583.657414804
phc_ctl[43237.739]: adjusted clock by 1614053583.657415 seconds
...
ptp4l[43238.626]: master offset -1614053583657415723 s0 freq  -60 path delay  7730
ptp4l[43240.626]: master offset       -923 s0 freq +999999999 path delay  7726
...
ptp4l[43252.625]: master offset      -1587 s0 freq     -52 path delay      7726
ptp4l[43254.625]: master offset      -1725 s0 freq     -68 path delay      7728

#
# Master offset is -, adj +freq to make offset more positive
# Note the sudden increase in the master offset above 0
#
$ ./phc_ctl /dev/ptp1 -- freq 100000
phc_ctl[43256.419]: adjusted clock frequency offset to 100000.000000ppb
ptp4l[43256.625]: master offset      18739 s0 freq  +10232 path delay      7728
ptp4l[43258.625]: master offset     218724 s0 freq  +99978 path delay      7719
ptp4l[43260.624]: master offset     418657 s0 freq  +99954 path delay      7714
ptp4l[43262.624]: master offset     619941 s0 freq  +99962 path delay      6374
ptp4l[43264.624]: master offset     836138 s0 freq  +99972 path delay     -9859
ptp4l[43266.624]: master offset    1036107 s0 freq  +99936 path delay     -9936

#
# Master offset is +, adj -freq to make offset more negative
# Note the sudden decrease in the master offset to below 0
#
$ ./phc_ctl /dev/ptp1 -- freq -100000
phc_ctl[43267.670]: adjusted clock frequency offset to -100000.000000ppb
ptp4l[43268.624]: master offset    1045403 s0 freq   +4648 path delay -9936
ptp4l[43270.623]: master offset     845291 s0 freq -100066 path delay -9936
ptp4l[43272.623]: master offset     637303 s0 freq -100070 path delay -2068
ptp4l[43274.623]: master offset     428607 s0 freq -100082 path delay  6484
ptp4l[43276.623]: master offset     209219 s0 freq -100090 path delay  25712
ptp4l[43278.623]: master offset       2421 s0 freq -100058 path delay  32414
ptp4l[43280.622]: master offset    -197723 s0 freq -100082 path delay  32414

#
```

```
# Set freq to 0 to return to neutral position, will drift naturally.
# Note the return to a more stable master offset
#
$ ./phc_ctl /dev/ptp1 -- freq 0
phc_ctl[43281.814]: adjusted clock frequency offset to 0.000000ppb
ptp4l[43282.622]: master offset    -311498 s0 freq  -59672 path delay  26853
ptp4l[43284.622]: master offset    -308916 s0 freq     -56 path delay  24159
ptp4l[43286.622]: master offset    -305015 s0 freq     -32 path delay  20194
ptp4l[43288.622]: master offset    -302327 s0 freq     -80 path delay  17346
ptp4l[43290.622]: master offset    -297620 s0 freq     -52 path delay  12535
```

3.2 ts2phc

`ts2phc` one of the utilities under Linux PTP, is used to align the timestamping device with the Renesas timing device.

```
# ./ts2phc -h

usage: ts2phc [options]

 -c [dev|name]  phc slave clock (like /dev/ptp0 or eth0)
                (may be specified multiple times)
 -f [file]      read configuration from 'file'
 -h             prints this message and exits
 -l [num]       set the logging level to 'num'
 -m             print messages to stdout
 -q             do not print messages to the syslog
 -s [dev|name]  source of the PPS signal
                may take any of the following forms:
                    generic   - an external 1-PPS without ToD information
                    /dev/ptp0 - a local PTP Hardware Clock (PHC)
                    eth0      - a local PTP Hardware Clock (PHC)
                    nmea      - a gps device connected by serial port or network
 -v             prints the software version and exits
```

When using `ts2phc` with a ClockMatrix device, the message: `PTP_PIN_SETFUNC failed: Invalid argument` may pop up. The ClockMatrix PHC driver does not support dynamic `PTP_PIN_SETFUNC` yet. This message can be ignored because the ClockMatrix device is expected to be configured to output the expected 1-PPS during start-up.

3.3 ptp4l

`ptp4l` is an implementation of the Precision Time Protocol (PTP) according to IEEE standard 1588 for Linux. It implements Boundary Clock (BC), Ordinary Clock (OC), and Transparent Clock (TC).

```
usage: ptp4l [options]

Delay Mechanism

 -A        Auto, starting with E2E
 -E        E2E, delay request-response (default)
 -P        P2P, peer delay mechanism

 Network Transport

 -2        IEEE 802.3
 -4        UDP IPV4 (default)
 -6        UDP IPV6
```

```
 Time Stamping

 -H        HARDWARE (default)
 -S        SOFTWARE
 -L        LEGACY HW

 Other Options

 -f [file] read configuration from 'file'
 -i [dev]  interface device to use, for example 'eth0'
           (may be specified multiple times)
 -p [dev]  Clock device to use, default auto
           (ignored for SOFTWARE/LEGACY HW time stamping)
 -s        slave only mode (overrides configuration file)
 -l [num]  set the logging level to 'num'
 -m        print messages to stdout
 -q        do not print messages to the syslog
 -v        prints the software version and exits
 -h        prints this message and exits
```

Various `ptp4l` sample configuration files are in the <linuxptp>/configs directory. For more info about `ptp4l` configuration parameters see man page for `ptp4l`.

```
man -l ptp4l.8
```

To use the adjust phase feature, the following configuration parameters are important:

```
servo_num_offset_values
The number of offset values considered in order to transition from the
SERVO_LOCKED to the SERVO_LOCKED_STABLE state.
The transition occurs once the last 'servo_num_offset_values' offsets
are all below the 'servo_offset_threshold' value.

servo_offset_threshold
The offset threshold used in order to transition from the SERVO_LOCKED
to the SERVO_LOCKED_STABLE state.  The transition occurs once the last
'servo_num_offset_values' offsets are all below the threshold value.
The default value of offset_threshold is 0 (disabled).

tsproc_mode
Select the time stamp processing mode used to calculate offset and delay.
Possible values are filter, raw, filter_weight, raw_weight. Raw modes perform
well when the rate of sync messages (logSyncInterval) is similar to the rate of
delay messages (logMinDelayReqInterval or logMinPdelayReqInterval). Weighting
is useful with larger network jitters (e.g. software time stamping).
The default is filter.

write_phase_mode
This option enables using the "write phase" feature of a PTP Hardware
Clock.  If supported by the device, this mode uses the hardware's
built in phase offset control instead of frequency offset control.
The default value is 0 (disabled).
```

3.3.1. Example Write Phase Mode Configuration

```
first_step_threshold      0.000020000
step_threshold            0.000020000
tx_timestamp_timeout      100

write_phase_mode          1
servo_offset_threshold    100
servo_num_offset_values   64
tsproc_mode               raw
```

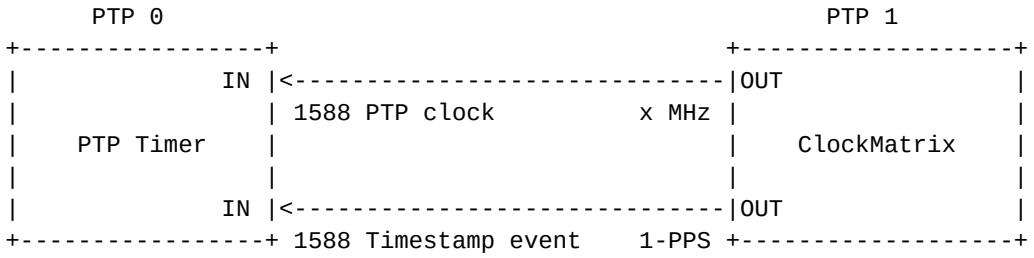
`tsproc_mode` in `raw` mode to let the hardware filter do the time stamp filtering.
Write phase adjustments only happen in SERVO_LOCK_STABLE state.
Configure the `servo_offset_threshold` and `servo_num_offset_values` so that it is close enough to 0 offset so that the hardware filtering, which has limited bandwidth, can converge to zero in a relatively short period of time.

3.4 Sample Session

The following sample session is on a Xilinx ZCU102 board with a ClockMatrix device.

CM DPLL2 is the 1588 PTP clock to the Xilinx Timer.

The Xilinx Timer is registered as ptp0 and ClockMatrix is registered as ptp1.




```
PTP time stamper driver supports: PTP_CLK_REQ_EXTTS
```

Start ts2phc

We want time adjustments and no frequency adjustments, so we configure `ts2phc` to use the `nullf` servo.

Sample configuration `ts2phc.cfg`:

```
$ cat ts2phc.cfg

#
# ts2phc config file to get it to behave like syncd to align
# time stamper to PHC device's 1-PPS signal.
#
# Example:
# ./ts2phc -m -q -f ts2phc.cfg
#

[global]
clock_servo nullf
first_step_threshold 0.000000001
step_threshold 0.000000001

# time stamper, slave device
[/dev/ptp0]
ts2phc.channel 0
```

```
# PHC device (ex. CM), master device
[/dev/ptp1]
ts2phc.master  1
ts2phc.channel 5
```

Continuing with the example on the Xilinx ZCU102 board:

- Xilinx time stamping device is `eth0` or `ptp0`
- ClockMatrix is `ptp1`

`ptp0` is the slave clock (i.e., it needs to align itself with the incoming 1-PPS from the source clock (ClockMatrix)).

When first starting `ts2phc`, it may be beneficial to see the log messages.

```
$ ./ts2phc -m -q -f ts2phc.cfg -l 7

ts2phc[84.381]: config item /dev/ptp0.ts2phc.master is 0
ts2phc[84.382]: config item (null).clock_servo is 3
ts2phc[84.382]: config item /dev/ptp0.ts2phc.pin_index is 0
ts2phc[84.382]: config item /dev/ptp0.ts2phc.channel is 0
ts2phc[84.382]: config item /dev/ptp0.ts2phc.extts_polarity is 2
ts2phc[84.382]: config item /dev/ptp0.ts2phc.extts_correction is 0
ts2phc[84.382]: config item /dev/ptp0.ts2phc.pulsewidth is 500000000
ts2phc[84.382]: config item (null).free_running is 0
ts2phc[84.382]: PHC slave /dev/ptp0 has ptp index 0
ts2phc[84.382]: config item (null).step_threshold is 0.000000
ts2phc[84.382]: config item (null).first_step_threshold is 0.000000
ts2phc[84.382]: config item (null).max_frequency is 900000000
ts2phc[84.382]: config item (null).servo_offset_threshold is 0
ts2phc[84.382]: config item (null).servo_num_offset_values is 10
ts2phc[84.382]: config item /dev/ptp1.ts2phc.master is 1
ts2phc[84.383]: PHC master /dev/ptp1 has ptp index -1
ts2phc[84.383]: config item /dev/ptp1.ts2phc.channel is 5
ts2phc[84.383]: config item /dev/ptp1.ts2phc.pin_index is 0
PTP_PIN_SETFUNC failed: Invalid argument
ts2phc[84.383]: Failed to set the pin. Continuing bravely on...
ts2phc[85.113]: /dev/ptp0 extts index 0 at 169492.956561892 corr 0 src 10.465738 diff 169482956561892
ts2phc[85.113]: /dev/ptp0 master offset 169482956561892 s1 freq+0
ts2phc[86.112]: /dev/ptp0 extts index 0 at 11.000000000 corr 0 src 11.265748 diff 0
ts2phc[86.112]: /dev/ptp0 master offset 0 s2 freq+0
ts2phc[87.112]: /dev/ptp0 extts index 0 at 12.000000000 corr 0 src 12.265798 diff 0
ts2phc[87.112]: /dev/ptp0 master offset 0 s2 freq+0
ts2phc[88.112]: /dev/ptp0 extts index 0 at 13.000000000 corr 0 src 13.265798 diff 0
ts2phc[88.112]: /dev/ptp0 master offset 0 s2 freq+0
```

`Failed to set the pin. Continuing bravely on...` is expected, this will be addressed in a future version of the ClockMatrix PHC driver.

`ptp0 extts index 0 at 169492.956561892` tells us the counter value on the rising edge of the incoming 1-PPS signal on `ptp0`.

Alignment is achieved when the sub-second portion of the counter is 0.

ts2phc shows that `ptp0` had a 956561892 ns offset at the rising edge of the incoming 1-PPS signal.

```
ts2phc[85.113]: /dev/ptp0 extts index 0 at 169492.956561892 corr 0 src 10.465738 diff 169482956561892
```

ts2phc adjusted Xilinx's timer to reduce the sub-second offset to 0.

```
ts2phc[85.113]: /dev/ptp0 master offset 169482956561892 s1 freq      +0
```

ts2phc subsequently reports that ptp0 is aligned with ptp1's PPS, ie. diff 0

The Xilinx timer time stamp jumped from xxxx.956561892 to xxxx.000000000.

```
ts2phc[86.112]: /dev/ptp0 extts index 0 at 11.000000000 corr 0 src 11.265748 diff 0
```

The subsequent 1-PPS events show 0 sub-second values ``12.000000000`` and ``13.000000000`` which indicates ``ptp0`` is aligned with the incoming 1-PPS signal from ClockMatrix.

```
ts2phc[87.112]: /dev/ptp0 extts index 0 at 12.000000000 corr 0 src 12.265798 diff 0
ts2phc[88.112]: /dev/ptp0 extts index 0 at 13.000000000 corr 0 src 13.265798 diff 0
```

Once the system is behaving as expected, ts2phc can be started as a background process without the verbose messaging.

```
$ ./ts2phc -m -q -f ts2phc.cfg &
PTP_PIN_SETFUNC failed: Invalid argument
ts2phc[31.038]: Failed to set the pin. Continuing bravely on...
```

Start ptp4l

The following examples use ptp4l compiled from the linuxptp master branch, Fri Feb 26, 2021, commit f774703cb1.

3.4.1.1. Unicast

The following is a sample ptp4l configuration file for a unicast master.

Please ensure the domainNumber matches the unicast master's domainNumber; otherwise, the unicast requests will be ignored by the unicast master.

```
$ cat standalone_G.8275.2.cfg
#
# Telecom G.8275.2 T-TSC example configuration containing those attributes
# which either differ from the defaults or are relevant to the profile.
#
[global]
domainNumber                    44

slaveOnly                       1
masterOnly                      0

# Announce messages
announceReceiptTimeout          2
logAnnounceInterval             1

# Sync/Delay_Req/Delay_Resp messages
# ex. 0 = 1-PPS, -3 = 8 PPS, -4 = 16 PPS
logSyncInterval                 -4
logMinDelayReqInterval          -4

#
# step_window is in units of sync packets
#
# 3 seconds:
#@ 16 PPS, set to 48
#@ 1  PPS, set to  3
step_window                     48

clockClass                      255
```

```
clockAccuracy                   0xFE
timeSource                      0xa0
maxStepsRemoved                 255

offsetScaledLogVariance         0xffff

G.8275.defaultDS.localPriority  128
G.8275.portDS.localPriority     128

priority1                       128
priority2                       255

dataset_comparison              G.8275.x
transportSpecific               0

clock_type                      OC
delay_mechanism                 E2E


first_step_threshold            0.000020000
step_threshold                  0.000020000
tx_timestamp_timeout            1000

write_phase_mode                0
servo_offset_threshold          100
servo_num_offset_values         64
tsproc_mode raw

network_transport               UDPv4

hybrid_e2e                      1
inhibit_multicast_service       1
unicast_listen                  1
unicast_req_duration            300

[unicast_master_table]
table_id                        1
logQueryInterval                2
UDPv4                           10.64.10.1

[eth0]
unicast_master_table            1
```

The sample configuration file is for G.8275.2, but we can use it as an example for a unicast master.

With `ts2phc` running in the background, `ptp4l` is used to synchronize with a PTP master.

Unicast slave using standalone.G.8275.2.cfg with 16 PPS for Sync/Delay Request

```
============================== ================================================================
Option                         Description
============================== ================================================================
-m                             Print messages to stdout
-q                             Do not print messages to the syslog
-p /dev/ptp1                   Use PHC clock device /dev/ptp1
-f standalone.G.8275.2.cfg     Read configuration from standalone.G.8275.2.cfg
--domainNumber 4               Override the domainNumber at the command line to match GM
============================== ================================================================

$ ./ptp4l -m -q -p /dev/ptp1 -f standalone_G.8275.2.cfg --domainNumber 4

option slaveOnly is deprecated, please use clientOnly instead
ptp4l[28.339]: selected /dev/ptp1 as PTP clock
ptp4l[28.341]: port 0 (/var/run/ptp4l): hybrid_e2e only works with E2E
ptp4l[28.341]: port 0 (/var/run/ptp4lro): hybrid_e2e only works with E2E
ptp4l[28.341]: port 1 (eth0): taking /dev/ptp1 from the command line, not the attached ptp0
ptp4l[28.362]: port 1 (eth0): INITIALIZING to LISTENING on INIT_COMPLETE
ptp4l[28.362]: port 0 (/var/run/ptp4l): INITIALIZING to LISTENING on INIT_COMPLETE
ptp4l[28.362]: port 0 (/var/run/ptp4lro): INITIALIZING to LISTENING on INIT_COMPLETE
ptp4l[32.782]: port 1 (eth0): new foreign master 00b0ae.fffe.02e810-1
ptp4l[32.988]: selected local clock 000a35.fffe.00bf01 as best master
ptp4l[36.782]: selected best master clock 00b0ae.fffe.02e810
ptp4l[36.782]: updating UTC offset to 37
ptp4l[36.782]: port 1 (eth0): LISTENING to UNCALIBRATED on RS_SLAVE
ptp4l[41.973]: clockcheck: clock jumped forward or running faster than expected!
ptp4l[43.911]: port 1 (eth0): UNCALIBRATED to SLAVE on MASTER_CLOCK_SELECTED
ptp4l[44.410]: rms 807315252816901888 max 1614630505633803776 freq    -34 +/- 174 delay  7705 +/-
ptp4l[45.410]: rms  120 max  172 freq   -203 +/-  17 delay  7709 +/-  10
ptp4l[46.410]: rms   14 max   28 freq   -107 +/-  19 delay  7706 +/-  13
ptp4l[47.410]: rms   20 max   40 freq    -53 +/-  13 delay  7709 +/-   9
ptp4l[48.410]: rms   15 max   34 freq    -43 +/-  18 delay  7702 +/-   7
ptp4l[49.410]: rms    7 max   16 freq    -56 +/-  12 delay  7705 +/-   7
ptp4l[50.410]: rms   12 max   20 freq    -62 +/-  20 delay  7703 +/-   9
ptp4l[51.410]: rms    7 max   18 freq    -56 +/-  12 delay  7710 +/-   6
ptp4l[52.410]: rms    8 max   22 freq    -57 +/-  14 delay  7708 +/-  11
ptp4l[53.410]: rms    7 max   14 freq    -61 +/-  11 delay  7710 +/-   9
ptp4l[54.410]: rms   12 max   26 freq    -56 +/-  20 delay  7708 +/-   9
ptp4l[55.409]: rms    7 max   12 freq    -52 +/-  11 delay  7708 +/-   7
ptp4l[56.409]: rms    9 max   14 freq    -60 +/-  14 delay  7705 +/-  10
ptp4l[57.409]: rms   11 max   24 freq    -62 +/-  19 delay  7705 +/-  11
ptp4l[58.409]: rms    9 max   24 freq    -56 +/-  15 delay  7708 +/-  10
ptp4l[59.409]: rms   13 max   28 freq    -61 +/-  21 delay  7706 +/-   9
ptp4l[60.409]: rms    6 max   12 freq    -51 +/-   8 delay  7708 +/-   6
```

Unicast slave using standalone.G.8275.2.cfg at 1-PPS Sync/Delay Request

```
============================== ================================================================
Option                         Description
============================== ================================================================
-m                             Print messages to stdout
-q                             Do not print messages to the syslog
-p /dev/ptp1                   Use PHC clock device /dev/ptp1
-f standalone.G.8275.2.cfg     Read configuration from standalone.G.8275.2.cfg
--domainNumber 4               Override the domainNumber at the command line to match GM
-step_window 3                 Delay 3 sync packets after adjtime() to allow timestamps to settle
--logSyncInterval 0            Sync packet rate at 1-PPS
--logMinDelayReqInterval 0     Delay request packet rate at 1-PPS
============================== ================================================================
```

```
$  ./ptp4l -mq -p /dev/ptp1 -f standalone_G.8275.2.cfg --domainNumber 4 --step_window 3 --
logSyncInterval 0 --logMinDelayReqInterval 0

option slaveOnly is deprecated, please use clientOnly instead
ptp4l[54.219]: selected /dev/ptp1 as PTP clock
ptp4l[54.221]: port 0 (/var/run/ptp4l): hybrid_e2e only works with E2E
ptp4l[54.221]: port 0 (/var/run/ptp4lro): hybrid_e2e only works with E2E
ptp4l[54.221]: port 1 (eth0): taking /dev/ptp1 from the command line, not the attached ptp0
ptp4l[54.242]: port 1 (eth0): INITIALIZING to LISTENING on INIT_COMPLETE
ptp4l[54.242]: port 0 (/var/run/ptp4l): INITIALIZING to LISTENING on INIT_COMPLETE
ptp4l[54.242]: port 0 (/var/run/ptp4lro): INITIALIZING to LISTENING on INIT_COMPLETE
ptp4l[54.714]: port 1 (eth0): new foreign master 00b0ae.fffe.02e810-1
ptp4l[58.714]: selected best master clock 00b0ae.fffe.02e810
ptp4l[58.714]: updating UTC offset to 37
ptp4l[58.714]: port 1 (eth0): LISTENING to UNCALIBRATED on RS_SLAVE
ptp4l[59.651]: master offset -1614631351385584072 s0 freq      +0 path delay         0
ptp4l[60.652]: master offset -1614631351385584108 s1 freq     -36 path delay         0
ptp4l[62.651]: clockcheck: clock jumped forward or running faster than expected!
ptp4l[64.651]: master offset       -252 s0 freq     -36 path delay         0
ptp4l[65.651]: master offset       -314 s2 freq     -98 path delay      7712
ptp4l[65.651]: port 1 (eth0): UNCALIBRATED to SLAVE on MASTER_CLOCK_SELECTED
ptp4l[66.651]: master offset       -298 s2 freq    -396 path delay      7712
ptp4l[67.651]: master offset       -116 s2 freq    -303 path delay      7702
ptp4l[68.651]: master offset        192 s2 freq     -30 path delay      7702
ptp4l[69.651]: master offset        258 s2 freq     +93 path delay      7732
ptp4l[70.651]: master offset        162 s2 freq     +75 path delay      7712
ptp4l[71.651]: master offset        -36 s2 freq     -75 path delay      7766
ptp4l[72.651]: master offset        -12 s2 freq     -61 path delay      7766
ptp4l[73.651]: master offset          8 s2 freq     -45 path delay      7710
ptp4l[74.650]: master offset         -8 s2 freq     -59 path delay      7710
ptp4l[75.650]: master offset        -18 s2 freq     -71 path delay      7708
ptp4l[76.650]: master offset         -4 s2 freq     -62 path delay      7706
ptp4l[77.650]: master offset         -4 s2 freq     -64 path delay      7718
ptp4l[78.650]: master offset          0 s2 freq     -61 path delay      7718
ptp4l[79.650]: master offset         -4 s2 freq     -65 path delay      7718
ptp4l[80.650]: master offset          4 s2 freq     -58 path delay      7718
ptp4l[81.650]: master offset         10 s2 freq     -51 path delay      7720
ptp4l[82.650]: master offset          8 s2 freq     -50 path delay      7710
```

Multicast

The sample configuration file is for G.8275.1, but we can use it as an example for a multicast master to show `write_phase_mode` enabled.

```
cat standalone_G.8275.1.cfg
#
# Telecom G.8275.1 T-TSC example configuration containing those attributes
# which either differ from the defaults or are relevant to the profile.
#
[global]
domainNumber    24

slaveOnly       1
masterOnly      0

# Announce messages
announceReceiptTimeout  3
logAnnounceInterval     -3
```

```
# Sync/Delay_Req/Delay_Resp messages â 16 packets-per-second, -4 = 16 PPS
logSyncInterval         -4
logMinDelayReqInterval  -4

#
# step_window is in units of sync packets
#
# 3 seconds:
#   @ 16 PPS, set to 48
#   @ 1  PPS, set to  3
step_window     48

clockClass      255
clockAccuracy   0xFE
timeSource      0xa0
maxStepsRemoved 255

offsetScaledLogVariance 0xffff

G.8275.defaultDS.localPriority  128
G.8275.portDS.localPriority     128

priority1       128
priority2       255

dataset_comparison      G.8275.x
transportSpecific       0

clock_type      OC
delay_mechanism E2E


first_step_threshold    0.000020000
step_threshold          0.000020000
tx_timestamp_timeout    1000

write_phase_mode        1
servo_offset_threshold  100
servo_num_offset_values 64
tsproc_mode             raw

network_transport       L2

# 01:1B:19:00:00:00     Forwardable multi-cast address
# 01:80:C2:00:00:0E     Non-forwardable multi-cast address
ptp_dst_mac             01:1B:19:00:00:00

[eth0]
```

With `ts2phc` running in the background, `ptp4l` is used to synchronize with a PTP multicast master.

Multicast slave using standalone.G.8275.1.cfg

```
============================== ================================================================
Option                         Description
============================== ================================================================
-m                             Print messages to stdout
-q                             Do not print messages to the syslog
```

```
-p /dev/ptp1                    Use PHC clock device /dev/ptp1
-f standalone.G.8275.1.cfg      Read configuration from standalone.G.8275.1.cfg
============================== ================================================================
```

`freq +0` indicates that ptp4l writes the phase offset directly into the PHC clock device, so no frequency adjustments are made.

```
$ ./ptp4l -m -q -p /dev/ptp1 -f standalone.G.8275.1.cfg

ptp4l[23.183]: selected /dev/ptp1 as PTP clock
ptp4l[23.185]: port 1 (eth0): taking /dev/ptp1 from the command line, not the attached ptp0
ptp4l[23.245]: port 1 (eth0): INITIALIZING to LISTENING on INIT_COMPLETE
ptp4l[23.245]: port 0 (/var/run/ptp4l): INITIALIZING to LISTENING on INIT_COMPLETE
ptp4l[23.245]: port 0 (/var/run/ptp4lro): INITIALIZING to LISTENING on INIT_COMPLETE
ptp4l[23.356]: port 1 (eth0): new foreign master 0080ea.fffe.4dc760-1
ptp4l[23.606]: selected best master clock 0080ea.fffe.4dc760
ptp4l[23.606]: updating UTC offset to 37
ptp4l[23.606]: port 1 (eth0): LISTENING to UNCALIBRATED on RS_SLAVE
ptp4l[25.801]: clockcheck: clock jumped forward or running faster than expected!
ptp4l[27.114]: port 1 (eth0): UNCALIBRATED to SLAVE on MASTER_CLOCK_SELECTED
ptp4l[27.614]: rms 807316614105398016 max 1614633228210796032 freq   -220 +/- 171 delay  7235 +/-   8
ptp4l[28.614]: rms   56 max   83 freq   -147 +/-  77 delay  7229 +/-   6
ptp4l[29.614]: rms   60 max   91 freq    -23 +/-  16 delay  7230 +/-  12
ptp4l[30.614]: rms   24 max   41 freq    -23 +/-  13 delay  7230 +/-   7
ptp4l[31.613]: rms   10 max   21 freq    -30 +/-  27 delay  7226 +/-   8
ptp4l[32.613]: rms   29 max   47 freq     +0 +/-   0 delay  7234 +/-   8
ptp4l[33.613]: rms   42 max   55 freq     +0 +/-   0 delay  7231 +/-  14
ptp4l[34.613]: rms   55 max   71 freq     +0 +/-   0 delay  7229 +/-   9
ptp4l[35.613]: rms   58 max   75 freq     +0 +/-   0 delay  7231 +/-   8
ptp4l[36.613]: rms   62 max   73 freq     +0 +/-   0 delay  7231 +/-   6
ptp4l[37.613]: rms   62 max   75 freq     +0 +/-   0 delay  7228 +/-  10
```

4. Appendix

4.1 Performance Metrics

The performance metric in the next section has the following definitions.

- **CPU%** - The percentage of the CPU that is being used by the process.
- **DMIPS -** Number is the CPU % multiplied by the board's measured DMIPS number.
- **RSS** - Resident Set Size is the memory occupied by a process that is held in main memory. It does not include memory that is swapped out. It does include memory from shared libraries as long as the pages from those libraries are actually in memory. It does include all stack and heap memory.

The REA measured DMIPS for the ZCU102 Evaluation Board is **2948.967211** DMIPS

For information about the DMIPS calculation, see DMIPS Calculation.

4.1.1. WritePhase Reference Tracker, Multicast master

	16 PPS		
	1 Master		
	CPU %	**DMIPS**	**RSS (MB)**
ptp4l	0.22%	6.52	0.604

4.1.2. DMIPS Calculation

4.1.2.1. Hardware Specifications

Board: Xilinx ZCU102 Eval Board, CPU @ 1.333 GHz

```
# cat /sys/devices/system/cpu/cpu0/cpufreq/cpuinfo_cur_freq
[25310.282758] cpu cpu0: dev_pm_opp_set_rate: failed to find current OPP for freq
1333333320 (-34)
1333333
```

https://www.xilinx.com/products/boards-and-kits/ek-u1-zcu102-g.html

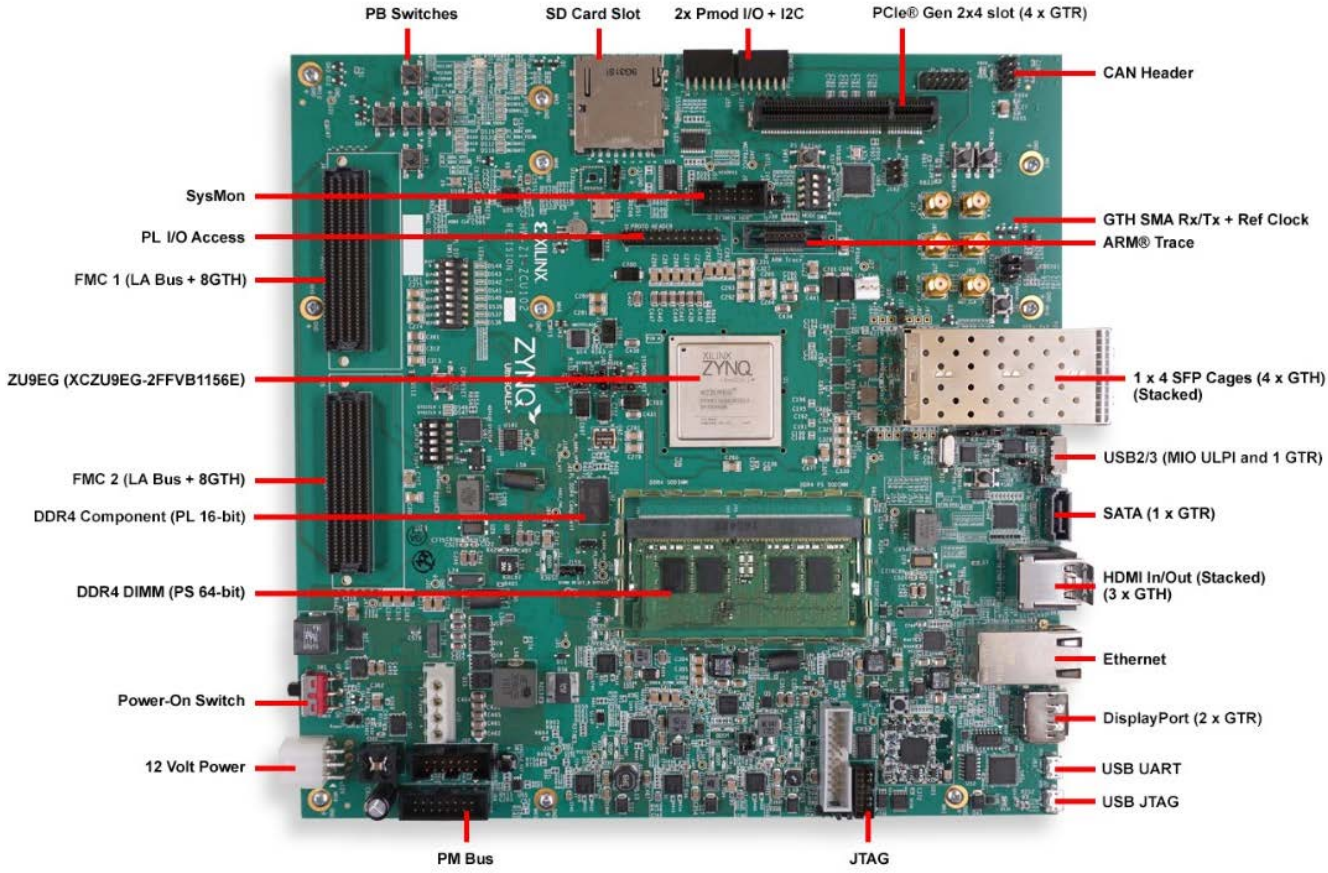


Figure 1. ZCU102 Evaluation Board Features

4.1.2.2. Software Environment

4.1.2.2.1. Compiler

```
aarch64-linux-gnu-gcc (Ubuntu/Linaro 5.4.0-6ubuntu1~16.04.9) 5.4.0 20160609
```

4.1.2.2.2. Compile Flags

```
-O3 -fno-inline -DTIME -DHZ=60 -DNDEBUG -w –DRENESAS
```

4.1.2.2.3. Linux Version

```
Linux pl_eth_1g_ptp_zcu102_cm 4.14.0-xilinx-v2018.3 #1 SMP Mon Mar 22 19:09:09 UTC 2021
aarch64 GNU/Linux
```

4.1.2.2.4. Dhrystone Version

https://fossies.org/linux/privat/old/dhrystone-2.1.tar.gz

Modified for ZCU102 environment – modified TIME macro time measurement mechanism to use clock_gettime() for nanosecond granularity versus second granularity.

4.1.2.2.5. ZCU102 Patch File

```
Type 'make' should work (sudo apt-get install gcc-aarch64-linux-gnu).

Copy gcc_dry2 to board and run for 150,000,000 which is about 20+ seconds.

$ ./gcc_dry2

Dhrystone Benchmark, Version 2.1 (Language: C)
```

```
Program compiled without 'register' attribute

Please give the number of runs through the benchmark: 150000000

Execution starts, 150000000 runs through Dhrystone

...

User_Time = 28.957748413
Microseconds for one run through Dhrystone:    0.2
Dhrystones per Second:                      5179960.5
---
 Makefile | 12 +++++++-----
 dhry_1.c | 34 ++++++++++++++++++++++++++++++++++
 2 files changed, 41 insertions(+), 5 deletions(-)

diff --git a/Makefile b/Makefile
index dfaa795..8b6ad76 100644
--- a/Makefile
+++ b/Makefile
@@ -17,13 +17,14 @@
 CC=        cl          # C compiler name goes here (MSC)
 CC=        cc          # C compiler name goes here (UNIX)
 GCC=       gcc
+GCC=       /usr/bin/aarch64-linux-gnu-gcc

 PROGS=     msc         # Programs to build (MSC)
 PROGS=     unix            # Programs to build (UNIX)

 #TIME_FUNC=    -DMSC_CLOCK     # Use Microsoft clock() for measurement
-#TIME_FUNC=    -DTIME          # Use time(2) for measurement
-TIME_FUNC= -DTIMES         # Use times(2) for measurement
+TIME_FUNC= -DTIME          # Use time(2) for measurement
+#TIME_FUNC=    -DTIMES         # Use times(2) for measurement
 #HZ=       50          # Frequency of times(2) clock ticks
 HZ=        60          # Frequency of times(2) clock ticks
 #HZ=       100         # Frequency of times(2) clock ticks
@@ -37,12 +38,12 @@ ENUMS=                  # Compiler does have enum type

 OPTIMIZE=  -Ox -G2         # Optimization Level (MSC, 80286)
 OPTIMIZE=  -O4         # Optimization Level (generic UNIX)
-GCCOPTIM=       -O
+GCCOPTIM=  -O3 -fno-inline

 LFLAGS=                    #Loader Flags

 CFLAGS=    $(OPTIMIZE) $(TIME_FUNC) -DHZ=$(HZ) $(ENUMS) $(STRUCTASSIGN) $(CFL)
-GCCFLAGS= $(GCCOPTIM) $(TIME_FUNC) -DHZ=$(HZ) $(ENUMS) $(STRUCTASSIGN) $(CFL)
+GCCFLAGS= $(GCCOPTIM) $(TIME_FUNC) -DHZ=$(HZ) $(ENUMS) $(STRUCTASSIGN) $(CFL) -DNDEBUG -w -
DRENESAS

 #
 #      You shouldn't need to touch the rest
@@ -50,7 +51,8 @@ GCCFLAGS= $(GCCOPTIM) $(TIME_FUNC) -DHZ=$(HZ) $(ENUMS) $(STRUCTASSIGN)
$(CFL)
 SRC=       dhry_1.c dhry_2.c
 HDR=       dhry.h
```

```
-UNIX_PROGS=    cc_dry2 cc_dry2reg gcc_dry2 gcc_dry2reg
+#UNIX_PROGS=   cc_dry2 cc_dry2reg gcc_dry2 gcc_dry2reg
+UNIX_PROGS=    gcc_dry2
 MSC_PROGS= sdry2.exe sdry2reg.exe mdry2.exe mdry2reg.exe \
        ldry2.exe ldry2reg.exe cdry2.exe cdry2reg.exe \
        hdry2.exe hdry2reg.exe
diff --git a/dhry_1.c b/dhry_1.c
index 164c6f9..fba717d 100644
--- a/dhry_1.c
+++ b/dhry_1.c
@@ -17,6 +17,12 @@

 #include "dhry.h"

+#ifdef RENESAS
+#include <stdint.h>
+#include <time.h>
+#define NSEC2SEC 1000000000LL
+#endif
+
 /* Global Variables: */

 Rec_Pointer     Ptr_Glob,
@@ -51,8 +57,12 @@ extern  int     times ();
                 /* Measurements should last at least about 2 seconds */
 #endif
 #ifdef TIME
+#ifdef RENESAS
+struct timespec now;
+#else
 extern long     time();
                 /* see library function "time"  */
+#endif
 #define Too_Small_Time 2
                 /* Measurements should last at least 2 seconds */
 #endif
@@ -61,9 +71,15 @@ extern clock_t   clock();
 #define Too_Small_Time (2*HZ)
 #endif

+#ifdef RENESAS
+int64_t         Begin_Time,
+                End_Time;
+float           User_Time;
+#else
 long            Begin_Time,
                 End_Time,
                 User_Time;
+#endif
 float           Microseconds,
                 Dhrystones_Per_Second;

@@ -137,8 +153,13 @@ main ()
   Begin_Time = (long) time_info.tms_utime;
 #endif
 #ifdef TIME
+#ifdef RENESAS
```

```
+  clock_gettime(CLOCK_MONOTONIC, &now);
+  Begin_Time = now.tv_sec * NSEC2SEC + now.tv_nsec;
+#else
   Begin_Time = time ( (long *) 0);
 #endif
+#endif
 #ifdef MSC_CLOCK
   Begin_Time = clock();
 #endif
@@ -198,8 +219,13 @@ main ()
   End_Time = (long) time_info.tms_utime;
 #endif
 #ifdef TIME
+#ifdef RENESAS
+  clock_gettime(CLOCK_MONOTONIC, &now);
+  End_Time = now.tv_sec * NSEC2SEC + now.tv_nsec;
+#else
   End_Time = time ( (long *) 0);
 #endif
+#endif
 #ifdef MSC_CLOCK
   End_Time = clock();
 #endif
@@ -257,9 +283,17 @@ main ()
   printf ("        should be:   DHRYSTONE PROGRAM, 2'ND STRING\n");
   printf ("\n");

+#ifdef RENESAS
+  User_Time = (float)(End_Time - Begin_Time) / NSEC2SEC;
+  printf("User_Time = %0.9f\n", User_Time);
+
+  if (User_Time < (float)Too_Small_Time)
+#else
   User_Time = End_Time - Begin_Time;
+  printf("User_Time = %ld\n", User_Time);

   if (User_Time < Too_Small_Time)
+#endif
   {
     printf ("Measured time too small to obtain meaningful results\n");
     printf ("Please increase number of runs\n");
--
2.7.4
```

4.1.2.3. Results

From power-up, start at 140,000,000 loop iterations and increment 10,000,000 each time.

Iterations	Dhrystones per Second
140000000	5181170.5
150000000	5181197.0
160000000	5181290.5
170000000	5181255.5
180000000	5181080.5

Iterations	Dhrystones per Second
190000000	5181232.0
200000000	5182418.0
210000000	5181273.5
220000000	5181138.0
230000000	5181133.5

Discarding the first run, The average of the next 9 runs is **5181335.389**.

DMIPS (Dhrystone MIPS) numbers are calculated using the formula:

DMIPS = Dhrystones per second / 1757

Using formula above:

DMIPS = 5181335.389 / 1757

DMIPS = **2948.967211**

A more commonly reported figure is DMIPS / MHz.

DMIPS / MHz = 2948.967211 / 1333.333

DMIPS / MHZ = **2.21**

4.2 Identifying PHC Device Number

4.2.1. ClockMatrix

On a successful CM PHC driver load, the registered ptp device number can be see in the system log.

View the kernel start-up log with 'dmesg' command.

```
$ dmesg | grep idt
…
[    3.419848] idtcm 15-005b: 4.8.0, Id: 0x4001  HW Rev: 5  OTP Config Select: 15
[    7.128855] idtcm 15-005b: PLL2 registered as ptp1
```

4.2.2. Network Interface

The Ethernet interface in this example is `eth0`. It uses the Xilinx Timer as its PTP time stamping device.

```
$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0A:35:00:BF:01
          inet addr:10.64.10.191  Bcast:10.64.11.255  Mask:255.255.254.0
          inet6 addr: fe80::20a:35ff:fe00:bf01%4886680/64 Scope:Link
```

`ethtool` can be used to find the corresponding PHC clock device number.

```
$ ethtool -T eth0
Time stamping parameters for eth0:
Capabilities:
        hardware-transmit     (SOF_TIMESTAMPING_TX_HARDWARE)
        hardware-receive      (SOF_TIMESTAMPING_RX_HARDWARE)
        hardware-raw-clock    (SOF_TIMESTAMPING_RAW_HARDWARE)
PTP Hardware Clock: 0
Hardware Transmit Timestamp Modes:
        off                   (HWTSTAMP_TX_OFF)
```

```
        on                    (HWTSTAMP_TX_ON)
Hardware Receive Filter Modes:
        none                  (HWTSTAMP_FILTER_NONE)
```

`PTP Hardware Clock: 0` indicates that eth0 is registered as `ptp0`.

If the Ethernet interface is not supported as a PTP hardware clock, it will say `PTP Hardware Clock: none`.

For support in getting the Ethernet interface to register as a PTP Hardware Clock, contact the PHY/MAC vendor for support.

4.3 View ptp Device Name

View registered ptp devices:

```
$ ls /sys/class/ptp
ptp0  ptp1
```

View name of a ptp device:

```
$ cat /sys/class/ptp/ptp1/clock_name
IDT CM PLL2

$ cat /sys/class/ptp/ptp0/clock_name
Xilinx Timer
```

4.4 SO_SELECT_ERR_QUEUE: Protocol not available

`SO_SELECT_ERR_QUEUE` was introduced into the kernel in 3.10.

When using Linux kernels prior to 3.10, customers may encounter this message in `ptp4l`.

```
ptp4l[33.123]: eth1: SO_SELECT_ERR_QUEUE: Protocol not available
```

This message can be ignored; it is a warning message and does not prevent `ptp4l` from working

4.5 testptp

`testptp` is a Linux PHC test tool that is part of the kernel.

It is located in `tools/testing/selftests/ptp/testptp.c`.

```
usage: testptp [options]
 -a val     request a one-shot alarm after 'val' seconds
 -A val     request a periodic alarm every 'val' seconds
 -c         query the ptp clock's capabilities
 -d name    device to open
 -e val     read 'val' external time stamp events
 -f val     adjust the ptp clock frequency by 'val' ppb
 -g         get the ptp clock time
 -h         prints this message
 -i val     index for event/trigger
 -k val     measure the time offset between system and phc clock
            for 'val' times (Maximum 25)
 -l         list the current pin configuration
 -L pin,val configure pin index 'pin' with function 'val'
            the channel index is taken from the '-i' option
            'val' specifies the auxiliary function:
            0 - none
            1 - external time stamp
            2 - periodic output
```

```
 -p val     enable output with a period of 'val' nanoseconds
 -P val     enable or disable (val=1|0) the system clock PPS
 -s         set the ptp clock time from the system time
 -S         set the system time from the ptp clock time
 -t val     shift the ptp clock time by 'val' seconds
 -T val     set the ptp clock time to 'val' seconds
```

The following example is cross compiling the ``testptp`` tool in a linux v5.8-rc5 repository on an Ubuntu 16.04 build machine.

4.5.1. Sample Build for ZCU102 board

```
# Compile for ZCU102
$ sudo apt-get install gcc-aarch64-linux-gnu
$ export CROSS_COMPILE=/usr/bin/aarch64-linux-gnu-

# Linux kernel root (v5.8-rc5)
$ cd tools/testing/selftests/ptp
$ make
```

4.5.1.1. Compilation Error

When cross-compiling, the build machine's Linux kernel's ``ptp_clock.h`` may be outdated.

For example, the Ubuntu 16.04 build machine is using Linux v4.15.0.

```
$ uname -srm
Linux 4.15.0-107-generic x86_64
```

And produces the following compilation error:

```
# Linux kernel root (v5.8-rc5)
$ make
Makefile:10: warning: overriding recipe for target 'clean'
../lib.mk:126: warning: ignoring old recipe for target 'clean'
/usr/bin/aarch64-linux-gnu-gcc -I../../../../usr/include/    testptp.c  -lrt -o testptp
testptp.c: In function ‘do_flag_test’:
testptp.c:61:3: error: ‘PTP_EXTTS_REQUEST2’ undeclared (first use in this function)
   PTP_EXTTS_REQUEST2,
   ^
testptp.c:61:3: note: each undeclared identifier is reported only once for each function it
appears in
testptp.c:68:25: error: ‘PTP_EXTTS_VALID_FLAGS’ undeclared (first use in this function)
   PTP_ENABLE_FEATURE | (PTP_EXTTS_VALID_FLAGS + 1),
                         ^
testptp.c: In function ‘main’:
testptp.c:281:15: error: ‘struct ptp_clock_caps’ has no member named ‘adjust_phase’
           caps.adjust_phase);
               ^
<builtin>: recipe for target 'testptp' failed
make: *** [testptp] Error 1
```

4.5.1.2. Workaround

In `testptp.c`, replace `#include <linux/ptp_clock.h>` with a relative include to the local repo's `ptp_clock.h`.

```
testptp.c:
      #include <sys/time.h>
      #include <sys/timex.h>
      #include <sys/types.h>
```

```
      #include <time.h>
      #include <unistd.h>

-     #include <linux/ptp_clock.h>
+     #include "../../../../include/uapi/linux/ptp_clock.h"
```

With above workaround, `testptp` compiles.

```
$ make
Makefile:10: warning: overriding recipe for target 'clean'
../lib.mk:126: warning: ignoring old recipe for target 'clean'
/usr/bin/aarch64-linux-gnu-gcc -I../../../../usr/include/    testptp.c  -lrt -o testptp
```

4.6 phc_ctl

`phc_ctl` is part of the `linuxptp` collection of programs.

For details on `linuxptp`, see linuxptp

```
$ ./phc_ctl -h

usage: phc_ctl [options] <device> -- [command]

 device         ethernet or ptp clock device
 options
 -l [num]       set the logging level to 'num'
 -q             do not print messages to the syslog
 -Q             do not print messages to stdout
 -v             prints the software version and exits
 -h             prints this message and exits

 commands
 specify commands with arguments. Can specify multiple
 commands to be executed in order. Seconds are read as
 double precision floating point values.
  set  [seconds]  set PHC time (defaults to time on CLOCK_REALTIME)
  get             get PHC time
  adj  <seconds>  adjust PHC time by offset
  freq [ppb]      adjust PHC frequency (default returns current offset)
  cmp             compare PHC offset to CLOCK_REALTIME
  caps            display device capabilities (default if no command given)
  wait <seconds>  pause between commands.
```

4.7 1588 Profile Configuration Validation Tool for ptp4l

When creating a ptp4l configuration file for the G.8265.1, G.8275.1, or G.8275.2 profiles the `validate` tool can be used to check if the 1588 parameters conform to the ranges specified by the profile selected by option '`-p`'.

Valid -p options:

- G.8265.1
- G.8275.1
- G.8275.2

If a value does not conform to the profile, the value is highlighted and `validate` exits.

```
$ ./validate -h

usage: validate [options]
```

```
 -f [file]         read configuration from 'file'
 -p [profile name] Profile to be validated
                   Options:
                   G.8265.1/g.8265.1
                   G.8275.1/g.8275.1
                   G.8275.2/g.8275.2
 -l [num]          set the logging level to 'num'
 -m                print messages to stdout
 -h                prints this message and exits
```

Apply Patch File

`3.1-Introduce-validate-profile-program.patch` adds validate program to linuxptp v3.1 source.

For the patch file, email IDT-support-1588@lm.renesas.com.

```
$ git clone git://git.code.sf.net/p/linuxptp/code linuxptp
Cloning into 'linuxptp'...
...
Checking connectivity... done.

$ cp v3.1-Introduce-validate-profile-program.patch linuxptp

$ cd linuxptp

$ git checkout v3.1
Note: checking out 'v3.1'.
...
HEAD is now at 38ad326... Version 3.1

$ patch -p1 < v3.1-Introduce-validate-profile-program.patch
patching file .gitignore
patching file configs/G.8265.1_sample_master.cfg
patching file configs/G.8265.1_sample_slave.cfg
patching file configs/G.8275.1_sample.cfg
patching file configs/G.8275.1_sample_BC.cfg
patching file configs/G.8275.1_sample_GM.cfg
patching file configs/G.8275.2_sample.cfg
patching file configs/G.8275.2_sample_BC.cfg
patching file configs/G.8275.2_sample_GM.cfg
patching file configs/externServo_G.8275.1.cfg
patching file configs/externServo_G.8275.2.cfg
patching file configs/standalone_G.8275.1.cfg
patching file configs/standalone_G.8275.2.cfg
patching file makefile
patching file validate.c
patching file validate.h
```

Sample Compile Log

```
$ make validate
DEPEND validate.c
DEPEND ts2phc_slave.c
DEPEND ts2phc_nmea_master.c
...
DEPEND e2e_tc.c
DEPEND designated_fsm.c
DEPEND config.c
DEPEND clockcheck.c
```

```
DEPEND clockadj.c
DEPEND clock.c
DEPEND bmc.c
gcc -Wall -DVER=3.1  -D_GNU_SOURCE -DHAVE_CLOCK_ADJTIME -DHAVE_POSIX_SPAWN -
DHAVE_ONESTEP_SYNC     -c -o validate.o validate.c
gcc -Wall -DVER=3.1  -D_GNU_SOURCE -DHAVE_CLOCK_ADJTIME -DHAVE_POSIX_SPAWN -
DHAVE_ONESTEP_SYNC     -c -o hash.o hash.c
gcc -Wall -DVER=3.1  -D_GNU_SOURCE -DHAVE_CLOCK_ADJTIME -DHAVE_POSIX_SPAWN -
DHAVE_ONESTEP_SYNC     -c -o interface.o interface.c
gcc -Wall -DVER=3.1  -D_GNU_SOURCE -DHAVE_CLOCK_ADJTIME -DHAVE_POSIX_SPAWN -
DHAVE_ONESTEP_SYNC     -c -o phc.o phc.c
gcc -Wall -DVER=3.1  -D_GNU_SOURCE -DHAVE_CLOCK_ADJTIME -DHAVE_POSIX_SPAWN -
DHAVE_ONESTEP_SYNC     -c -o print.o print.c
gcc -Wall -DVER=3.1  -D_GNU_SOURCE -DHAVE_CLOCK_ADJTIME -DHAVE_POSIX_SPAWN -
DHAVE_ONESTEP_SYNC     -c -o sk.o sk.c
gcc -Wall -DVER=3.1  -D_GNU_SOURCE -DHAVE_CLOCK_ADJTIME -DHAVE_POSIX_SPAWN -
DHAVE_ONESTEP_SYNC     -c -o util.o util.c
gcc   validate.o hash.o interface.o phc.o print.o sk.o util.o  -lm -lrt -pthread  -o
validate
```

Sample Usage

The following is an example of a G.8275.1 T-TSC profile configuration with some invalid values.

```
$ cat g.8275.1.slave.cfg

[global]
announceReceiptTimeout     3
clockAccuracy              0xfe
clockClass                 255
clock_type                 OC
delay_mechanism            E2E
domainNumber               4
dataset_comparison         G.8275.x
G.8275.defaultDS.localPriority   128
G.8275.portDS.localPriority      128
logAnnounceInterval       -3
logMinDelayReqInterval     0
logSyncInterval            0
masterOnly                 0
maxStepsRemoved            255
network_transport          UDPv4
offsetScaledLogVariance    0xffff
priority1                  128
priority2                  128
ptp_dst_mac                01:1B:19:00:00:00
slaveOnly                  1
timeSource                 0xa0

$ ./validate -p G.8275.1 -f g.8275.1.slave.cfg
4 is an out of range value for option domainNumber at line 7
failed to parse configuration file g.8275.1.slave.cfg

Default values for G.8275.1 SLAVEONLY profile are:

 announceReceiptTimeout         3                The only allowed value is [3]
 clockAccuracy                  0xfe             The only allowed value is [0xfe]
 clockClass                     255              The only allowed value is [255]
```

```
 clock_type                     OC               The only allowed value is [OC]
 delay_mechanism                E2E              The only allowed value is [E2E]
 domainNumber                   24               The range is [24, 43]
 G.8275.defaultDS.localPriority 128              The range is [1, UINT8_MAX]
 G.8275.portDS.localPriority    128              The range is [1, UINT8_MAX]
 logAnnounceInterval            -3               The only allowed value is [-3]
 logMinDelayReqInterval         -4               The only allowed value is [-4]
 logSyncInterval                -4               The only allowed value is [-4]
 masterOnly                     0                The only allowed value is [0]
 maxStepsRemoved                255              The range is [1, UINT8_MAX]
 network_transport              L2               The only allowed value is [L2]
 offsetScaledLogVariance        0xffff           The only allowed value is [0xffff]

 priority1                      128              The only allowed value is [128]
 priority2                      255              The only allowed value is [255]
 ptp_dst_mac                    01:1B:19:00:00:00 The only allowed value is
[01:1B:19:00:00:00, 01:80:C2:00:00:0E]
 slaveOnly                      1                The only allowed value is [1]
 timeSource                     0xa0             The range is [0, UINT8_MAX]

// Changed domainNumber from 4 to 24

$ ./validate -p G.8275.1 -f g.8275.1.slave.cfg
0 is an out of range value for option logMinDelayReqInterval at line 12
failed to parse configuration file g.8275.1.slave.cfg
...

// Changed logMinDelayReqInterval and logSyncInterval from 0 to -4

$ ./validate -p G.8275.1 -f g.8275.1.slave.cfg
128 is an out of range value for option priority2 at line 19
failed to parse configuration file g.8275.1.slave.cfg
...

// Changed priority2 to 255

$ ./validate -p G.8275.1 -f g.8275.1.slave.cfg
Port item network_transport out of range in the global section!
...

// Changed network_transport to L2

$ $ ./validate -p G.8275.1 -f g.8275.1.slave.cfg
g.8275.1.slave.cfg Checked successfully for G.8275.1 3 profile
```

5. Revision History

Revision	Date	Description
1.01	August 18, 2021	▪ Clarify 3.1.6/3.1.7 is using time stamper PHC.
1.0	Jun 9, 2021	▪ Add Sanity Testing section to Getting Started ▪ Add note about SO_SELECT_ERR_QUEUE message to Appendix ▪ Update Getting Started Sample Session to use standalone_*.cfg files that include step_window parameter ▪ Rename to “LinuxPtpUsingPhcAdjustPhaseQuickStart” from “LinuxPTPUsingPHCAdjustPhaseReferenceManual” ▪ Add Performance Metrics section
0.2	Dec 17, 2020	▪ Add Revision History section ▪ Add section numbers to bookmarks on PDF ▪ Remove IDT from "The IDT ClockMatrix (TM) family ..." ▪ Update with ts2phc.cfg change ▪ Revise cover page, add logo, page numbering
0.1	Aug 7, 2020	▪ First release of this document

IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers skilled in the art designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only for development of an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising out of your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use o any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.0 Mar 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan

www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:

www.renesas.com/contact/

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.