## Testing timestamp using timestamping app

### 1. Build timestamping app

1. Build timestamping app
   - cd $(kernel_tree)/Documentation/networking/timestamping
   - open timestamping.c in the editor
   - look for 224.0.1.130 (line 414) and change it to 224.0.1.129
   - save the changes
   - arm-linux-gnueabihf-gcc -I../../../usr/include -o timestamping timestamping.c

   (assumption is that you have installed the ARM toolchain and have built the kernel and generated the headers i.e. make headers_install)

### 2. Copy timestamping app to your EVM filesystem

### 3. From your EVM serial port:

1. To your location of timestamping app in the filesystem
   - chmod +x timestamping

### 4. To test TX Hardware timestamp

1. ./timestamping eth0 SOF_TIMESTAMPING_TX_HARDWARE SOF_TIMESTAMPING_SYS_HARDWARE

   TX timestamp are provided on the error socket. Hence look for the line in the log containing "received error data". Here is the sample log:

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1352911185.650991</td>
<td>select 4349009us</td>
</tr>
<tr>
<td>1352911185.651167</td>
<td>select returned: 1, success ready for reading</td>
</tr>
<tr>
<td>1352911190.004529</td>
<td>sent 124 bytes</td>
</tr>
<tr>
<td>1352911190.004786</td>
<td>received error data, 166 bytes from 69.0.0.152, 80 bytes control messages</td>
</tr>
<tr>
<td>1352911195.005163</td>
<td>sent 124 bytes</td>
</tr>
<tr>
<td>1352911195.005362</td>
<td>received error data, 166 bytes from 69.0.0.152, 80 bytes control messages</td>
</tr>
</tbody>
</table>

These values should match
Understanding the log message:

1. 1352911185.651315: time returned by gettimeofday()
2. SW 0.000000000 ➔ Software generated timestamp for the TX packet. Since we didn’t set SOF_TIMESTAMPING_TXSOFTWARE and SOF_TIMESTAMPINGSOFTWARE flag in the command line, kernel doesn’t report any value in this field.
3. HW transformed 1352911185.7286786 ➔ This is PA hardware timestamp for the TX packet transformed to linux time format. You can plug this value in http://www.epochconverter.com/ and convert it to human date. It should roughly correspond to the value returned by the date command on your Appleton board. Also this value should roughly match the value returned by gettimeofday() (1352911185.651315)
4. HW raw 68210.643672961 ➔ This is the PA raw hardware timestamp for the TX packet.

Tests to be performed for TX timestamp:
1. Check if SIOCSHWTSTAMP line in the log contains matching value for “requested” and “got” fields
2. Check if “received error data” is printed at regular intervals (every 5 secs)
3. Check if HW transformed and HW raw timestamps have meaningful values (see above for explanation)
4. Check if HW transformed and HW raw timestamps are incrementing by 5 secs in each message. Run this test for more than 30 secs to see if there is no rollover in HW raw timestamp.

5. To test RX timestamp

First you need to start a PTP server. If you don’t have one, then download the source from http://ptpd2.svn.sourceforge.net/ (click on “Download GNU tarball”) onto a linux desktop machine and build it. Run the daemon using following command:

```
sudo /ptpd2 -G -b <ifname> -c -D -u <appleton-ip-addr>
```

On the EVM run the following command

```
./timestamping eth0 SOF_TIMESTAMPING_RX_HARDWARE SOF_TIMESTAMPING_SYS_HARDWARE SOF_TIMESTAMPING_RAW_HARDWARE
```

RX timestamp are provided on the regular socket along with the received message. Hence look for the line in the log containing “received regular data”. Here is the sample log:

```
SIOSCHWTSTAMP: tx_type 0 requested, got 0; rx_filter 4 requested, got 1
SO_TIMESTAMP 0
SO_TIMESTAMPNS 0
SO_TIMESTAMPING 100
1352914706.137518: sent 124 bytes
1352914706.137542: select 3862458us
1352914706.822993: select returned: 1, success
ready for reading
1352914706.823139: received regular data, 44 bytes from 10.218.113.170, 60 bytes control messages
   cmsg len 36: SOL_SOCKET SO_TIMESTAMPING SW 0.000000000 HW transformed 1352914706.902624013 HW raw 68210.643672961
   cmsg len 24: IPPROTO_IP IP_PKTINFO interface index 2
recvmsg error: Resource temporarily unavailable
1352914707.822888: select returned: 1, success
ready for reading
1352914707.822943: received regular data, 44 bytes from 10.218.113.170, 60 bytes control messages
   cmsg len 36: SOL_SOCKET SO_TIMESTAMPING SW 0.000000000 HW transformed 1352914707.902622866 HW raw 68210.643672961
   cmsg len 24: IPPROTO_IP_IP_PKTINFO interface index 2
recvmsg error: Resource temporarily unavailable
```

Since kernel driver turns on timestamping on all rx packets, value returned will be 1
Understanding the log message:

1. **1352914706.823139**: time returned by gettimeofday()
   
2. **SW 0.000000000**: Software generated timestamp for the RX packet. Since we didn’t set SOF_TIMESTAMPING_RX_SOFTWARE and SOF_TIMESTAMPING_SOFTWARE flag in the command line, kernel doesn’t report any value in this field.
   
3. **HW transformed 1352914706.902624013** → This is PA hardware timestamp for the RX packet transformed to linux time format. You can plug this value in [http://www.epochconverter.com/](http://www.epochconverter.com/) and convert it to human date. It should roughly correspond to the value returned by the date command on your Appleton board. Also this value should roughly match the value returned by gettimeofday() (1352914706.823139)
   
4. **HW raw 71731.817618188**: This is the PA raw hardware timestamp for the RX packet.

**Tests to be performed for RX timestamp:**

1. Check if SIOCSHWTSTAMP line in the log contains matching value for “requested” and “got” fields for tx_type. For rx_filter, since kernel driver turns on timestamping on all rx packets, value returned will be 1.
2. Check if “received regular data” is printed at regular intervals (rate depends on PTP server sending SYNC packets – there should be one line per SYNC packet)
3. Check if HW transformed and HW raw timestamps have meaningful values (see above for explanation)
4. Check if HW transformed and HW raw timestamps are incrementing at the same rate at which server is transmitting SYNC packets (1 sec in this example). Run this test for more than 30 secs to see if there is no rollover in HW raw timestamp.