Intel® Ethernet Network Adapter E810-CQDA2T

User Guide

Ethernet Products Group (EPG)

October 2023

Revision 1.1
722960-002

Revision History

Revision	Date	Comments
1.1	October 18, 2023	Updates include the following: • Updated GNSS interface naming to /dev/ttyGNSS_BBDD. • Updated Section 2.1, "E810-CQDA2T Features". • Added Section 2.3, "Synchronization Signaling". • Added Section 2.4, "Optional GNSS Module". • Updated Section 3.1, "Software Support/Packages". • Updated Section 3.2, "Building a linuxptp Project". • Added Section 3.4, "Building a synce4l Tool". • Updated Section 4.1, "Introduction". • Updated Section 4.2, "DPLL Priority". • Updated Section 4.4, "Channel 1 Configurations". • Updated Section 4.5, "Channel 2 Configurations". • Updated Section 4.6, "Recovered Clocks (G.8261 SyncE Support)". • Added notes to Section 4.7, "External Timestamp Signals". • Updated Section 4.9, "Reading Status of the DPLL". • Updated Section 4.10, "DPLL Monitoring". • Updated Section 4.11.1, "pin_cfg User Readable Format". • Updated Section 4.11.2, "dpll_<X>_ref_pin/dpll_<X>_state Machine Readable Interface". • Updated Section 4.12, "1PPS Signals from E810 Device to DPLL". • Updated Section 4.13, "1PPS Signals from the DPLL to E810 Device". • Updated Section 4.14, "GNSS Module Interface". • Added Section 4.15, "GNSS Advanced Features". • Updated Section 5.2.1, "External Connections". • Updated Section 5.2.2, "Software Configuration". • Updated Section 5.3.2, "Software Configuration". • Updated Section 5.4.3, "Software Configuration". • Updated Section 5.5.2, "Software Configuration". • Updated Section 5.6, "SyncE Setup". • Added Section 5.6.3, "ITU G.8264 ESMC Messaging Using synce4l". • Added Section 5.6.4, "Two E810-CQDA2T Adapter Configuration without GNSS". • Added Section 5.6.5, "Two E810-CQDA2T Adapter Configuration without GNSS and with 1PPS". • Added Section 5.6.6, "Two E810-CQDA2T Adapters with GNSS Connection Setup". • Updated Section 5.8, "Example ts2phc Configuration File". • Updated Section 5.9, "Example ptp4l Configuration File for BC". • Added Section 5.10, "Example synce4l Configuration File for BC". • Updated Section 6.2.1, "Leader Adapter". • Updated Section 6.2.2, "Follower Adapter". • Updated Appendix A, "Debug Notes".
1.0	July 19, 2022	Initial public release.

Contents

1.0 Introduction

Although IEEE 1588 Precision Time Protocol (PTP) support has been part of Intel's controller product line for generations. Intel only recently began developing PTP-optimized Ethernet Network Adapter products. The adoption of PTP in Ethernet connections is growing rapidly. Although the leader use case being considered is the build-out of 5G infrastructure, other applications that exist in datacenters, point of presence, financial, industrial, and energy sectors. These application areas can benefit from timing optimized, standard form factor, Ethernet adapters.

The Intel® Ethernet Network Adapter E810-CQDA2T (E810-CQDA2T), is based on the Intel® Ethernet Controller E810 (E810).

This document is structured as follows:

- Section 2.0, "E810-CQDA2T Ethernet Network Adapter"
- Section 3.0, "Software, Firmware, and Drivers"
- Section 4.0, "Configuring the E810-CQDA2T Using the Linux Kernel Interface"
- Section 5.0, "Configuration Setup"
- Section 6.0, "Initial Test Setup"
- Appendix A, "Debug Notes"
- Appendix B, "Glossary and Acronyms"

Note: In accordance with changes made in the IEEE 1588 Specification and where possible, the words Master and Slave have been replaced with Leader and Follower, respectively.

1.1 Reference Documents

Table 1 lists documents that can be found on the Intel Resource and Design Center (RDC) at:

https://www.intel.com/content/www/us/en/design/resource-design-center.html

Table 1. Reference Documents

Document Title	Document ID
Intel® Ethernet Controller E810 Datasheet	613875
Intel® Ethernet Controller E810 Specification Update	616943
Intel® Ethernet Controller E810 Feature Support Matrix	607252
Intel® Ethernet Network Adapter E810-CQDA2T Overview Video	737921
Intel® Ethernet Network Adapter E810-CQDA2T Product Brief	726375

Other documents that might be of interest include the following:

- IEEE 1588-2008 (v2.0)
- IEEE 1588-2019 (v2.1)
- IEEE 802.3AS
- ITU-T G.8271
- ITU-T G.8273
- ITU-T G.8273.2

- ITU-T G.8275.1
- ITU-T G.8275.2

Note: Linux PTP project information can be found at:

http://linuxptp.sourceforge.net/

2.0 E810-CQDA2T Ethernet Network Adapter

2.1 E810-CQDA2T Features

Following are the features of the Intel® Ethernet Network Adapter E810-CQDA2T:

- Based on Intel® Ethernet Controller E810 device using SFP28 form factor connections, as well as inheriting most of the features from the Intel® Ethernet Network Adapter E810-CQDA2 (E810-CQDA2).
- Carries forward the E810-XXVDA4T features.
- Software capability to configure the SMA signals via the standard Linux driver.
- Driver support for Linux PTP stack (**ptp4l**) to send synchronization messages and synchronize the host system clock to the adapter's PHC. This is present in all Intel network adapters.
- **ts2phc** utility for the Linux PTP stack optimized for synchronizing the PHC to a 1PPS input, rather than PTP messages.
- When combined with an external GNSS 1PPS input, the features of the adapter and Linux PTP stack provide the complete solution for a low cost PTP grand leader, while maintaining the ability to provide the host system's LAN connectivity.
- Exposed PTP timing signals on a front panel using SMA connectors. This enables timing signals to easily be connected by the end user after the adapter is installed in a system. Timing signals on the SMAs can be configured as inputs or outputs, typically configured for one pulse per second (1PPS) operation, but they will support a 10 MHz or 25 MHz signal (with or without the embedded 1PPS eSync). The rising edge of the 1PPS signal is used to mark the start of a new second, Top of Second (ToS). A 1PPS input signal is typically sourced from a GNSS module, or might be connected to the 1PPS output of another adapter. Circuitry is provided on board for isolation, buffering, and level-shifting to adapt the controller's CMOS signals for out-of-system use. Refer to Section 2.3 for 1PPS and 10 MHz signaling electrical requirements.
- In addition to the dual SMA connectors, the E810-CQDA2T also includes two U.FL connectors for 1PPS, 10 MHz, and/or 25 MHz; one is output only and the other is input only. This enables hardware-level traceability for connections to motherboards that do not have room for external SMA connectors.
- High accuracy reference clock. Intel adapters typically use on-board reference clocks with accuracy in the range of ±50 parts per million (ppm) as required to meet the Ethernet PHY requirements. Increased reference clock accuracy to ±2 parts per billion (ppb) can be achieved using an Oven-Controlled Crystal Oscillator (OCXO). This tighter accuracy enables the PTP Hardware Clock (PHC) inside the E810 to drift more slowly in the event the 1PPS source, or software adjustment via 1588 sync protocol, is lost. The ability to maintain time after the reference source is lost is known as holdover. This is estimated to extend holdover time to ~4 hours for ±1.5 µs.
- Add support for Synchronous Ethernet (SyncE) clock recovery (ITU G.8262).
- Add a high-quality DPLL to support multiple input clock functions and multiple output frequencies. The DPLL IC has separate integrated DPLL instances DPLL0 is used for generating high stability clock signal and DPLL1 used for driving the output signals.
- The timing information from the DPLL (that can arrive from the GNSS module, SMA connectors, E810 SDPs, or from the recovered SyncE signals) are routed to the E810 and can be used to synchronize the PHC (PTP hardware clocks). All E810 ports share only one physical PHC, and this is particularly useful in creating a Boundary Clock (BC) functionality like what is defined in ITU-T G.8273.2 (but without SyncE).

- Add mounting and connection provision for an optional GNSS module on board. When installed, the GNSS module provides the 1pps signal without the need for an external GNSS appliance. An I^2C interface provides a way to access the serial port data on the GNSS, including configuration options.

Table 2 provides additional information on the E810-CQDA2T:

Table 2. E810-CQDA2T Product Information

Product Description	1588 PTP/SyncE/GNSS 100/25/10 GbE QSFP28 PCIe adapter
Product Codes	E810CQDA2TG1 (no GNSS) E810CQDA2TGG1 (with GNSS)
Host Interface	PCIe 3.0x16 or PCIe 4.0x8/x16 (x16 connector)
Form Factor	Full-height, half-length PCIe card
Front Panel Connectors	2x QSFP28, 2x SMA (1PPS/10 MHz/25 MHz)[1], 1x SMB (GNSS antenna)[2]
Internal Connectors	1x U.FL input + 1x U.FL output (1PPS/10 MHz) / 25MHz; GNSS mezzanine
Optional GNSS SKU	Supports GPS, Galileo, GLONASS, BeiDou, QZSS
Synchronous Ethernet	ITU-T G.8261, G.8262, G.8262.1, and G.8264 (ESMC) support
Oscillator	1 ppb vs. temperature, 4 hours holdover (±1.5 µs under ±5 °C ΔT)
Power	36 W typical, 50 W max power consumption (with Class 7 optics at 100G maximum traffic)
Operating Temperature	0-55 °C with required airflow (passive heat sink)
EMI	FCC Class A
Manageability	NC-SI over MCTP (over PCIe/SMBus); EFI based iSCSI boot; EFI/legacy PXE boot support

1. Refer to Section 2.3 for 1PPS and 10 MHz signaling requirements.
2. Refer to Table 3 for GNSS antenna characteristics.

2.2 Architecture

The block diagram for the E810-CQDA2T is shown in Figure 1. The E810-CQDA2T provides two coaxial input/output SMA connectors, two U.FL connectors, and an optional GNSS input connector, as shown in Figure 2.

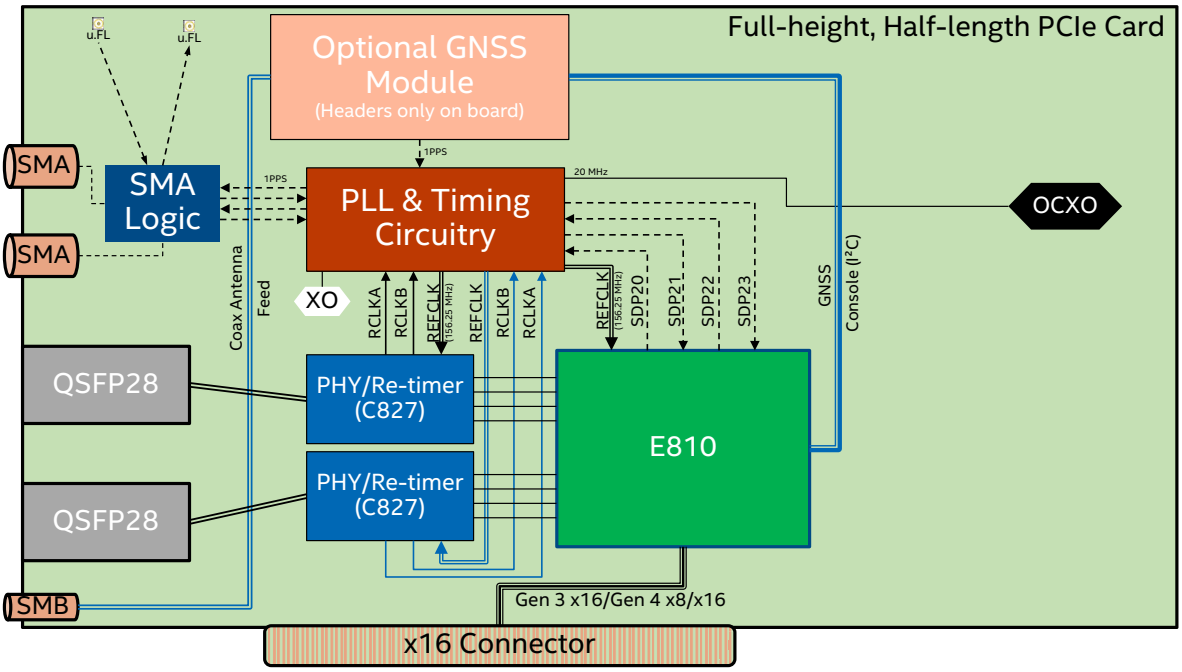


Figure 1. E810-CQDA2T Block Diagram

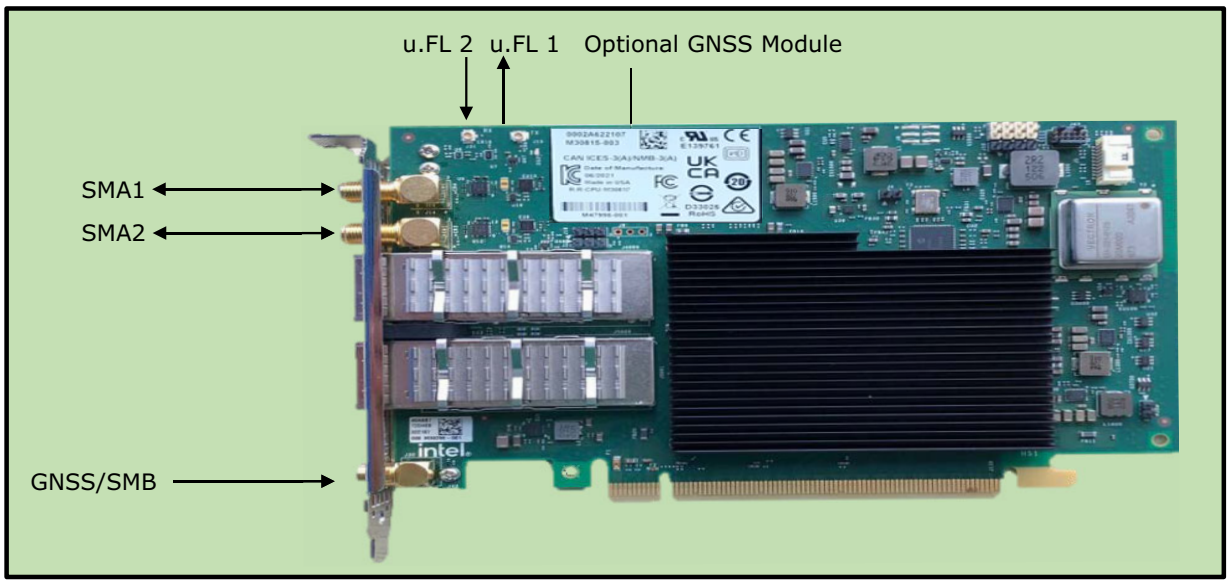


Figure 2. E810-CQDA2T Connector Locations

The optional GNSS module has a connection to an external antenna via an SMB female connector at the bottom of the faceplate. A cable with an SMB male connector and the characteristics listed in Table 3 are required to use the GNSS.

Table 3. E810-CQDA2T GNSS Antenna Characteristics

Characteristic	Min	Typical	Max
Characteristic impedance	-	50 Ω	-
DC operating voltage	< 2.9 V	3.3	> 3.6 V
Current consumption	-	-	30 mA
EMI immunity out-of-band	-	30 V/m	-
Out-of-band rejection	-	40 dB	-
Gain of active LNA at E810-CQDA2T SMB connector	17 dB	-	50 dB
Active antenna noise figure	-	< 4 dB	-
Axial ratio	-	-	2 dB
Phase center variation	-	< 10 mm over elevation/azimuth	-
Group delay variation in-band	-	10 ns max at each GNSS system bandwidth	-

The two SMA female connectors on the upper part of the faceplate are used for sending or receiving timing signals. Cables with SMA male connectors and the characteristics listed in Table 4 are required to use the SMA ports.

Table 4. E810-CQDA2T Timing Propagation Cable Characteristics

Characteristic	Min	Typical	Max
Frequency range	DC	–	>100 MHz
VSWR	-	1.25:1	1.33:1
Velocity of propagation	-	69%	-
RF shielding	-110 dB	-	-
Capacitance	-	95 pF/m	-
Impedance, cable	-	50 Ω	-
Impedance, connector (SMA or u.FL)	-	50 Ω	-
Tested cable assembly length	-	-	3 m
SMA connector torque	-	0.9 Nm	-

The U.FL cables have similar requirements as the SMA cables, except there is no torque requirement, and the tested cable length is <1 meter.

2.3 Synchronization Signaling

The E810-CQDA2T has two SMA connectors at the faceplate and two on-board U.FL connectors for 1588 high precision timing connectivity. Each SMA signal can be configured to be an input or an output.

The U.FL1 connector, associated with SMA1, can be output-only. The U.FL2 connector, associated with SMA2, can be input-only. Currently, 1PPS and 10 MHz signaling on both SMA and U.FL connectors is supported; additional signaling capabilities might be supported in future releases.

The 1PPS signaling is compliant with the ITU-T G.703 specification, but the 10 MHz signaling matches the 1PPS requirements rather than the ITU-T G.703 10 MHz requirements. For devices expecting ITU-T G.703 10 MHz requirements, DC blocks (for transmitted signaling from the E810-CQDA2T) and bias tees (for signaling received by the E810-CQDA2T Rx) must be used.

Table 5. E810-CQDA2T Supported 1PPS Input Signal (50 Ω Single-Ended)

Input	Min	Max	Units
VIH	1.65	5.5	V
VIL	-0.3	0.8	V
Rise Time		20	ns
Duty Cycle	1	51	%
Note: Assumes a 50 Ω termination. Parameters specified as measured at termination.			

2.4 Optional GNSS Module

The E810-XXVDA4TGG1 SKU includes a GNSS module installed as an add-in card, which provides the E810-CQDA2T with timing synchronization from GNSS constellations. This module outputs a 1PPS signal. Hardware support for presence detection of the antenna and current-limit protection for the E810-CQDA2T antenna input is included on this module.

2.5 Connectivity and Interoperability

E810-CQDA2T samples have been validated or verified with the following accessories. Contact your Intel representative if you require sample units (where available).

Table 6. E810-CQDA2T Accessories

Item	Validated	Samples Available from Intel	Notes
Intel® Silicon Photonics 100G PSM4 QSFP28 Optical Transceiver Receptacle, Version LH	Yes	Yes	Product code SPTSBP2PMCLH / MM 99AXT2 Requires separate LC fiber breakout cable (see Section 2.5.1).
Intel® Silicon Photonics 100G PSM4 QSFP28 Optical Transceiver, Version LH With 1.8m LC Breakout Fiber Pigtail	No	Yes	Product code SPTSBP3PCCLH002 / MM 99AXT3
Intel® Silicon Photonics 100G PSM4 QSFP28 Optical Transceiver, Version LH With 2.7m LC Breakout Fiber Pigtail	Yes	No	Product code SPTSBP3PCCLH003 / MM 99AXT8
Intel® Ethernet SFP+ LR Optics	See Notes column	No	Product code E10GSFPLR Intel has verified interoperability with 10 GbE Intel® Ethernet SFP+ LR Optics.
LC Fiber Breakout Cable	See Notes column	No	Intel has verified functionality with: • FS 8FMTPLCSMF • Hitachi 61538-12
Direct Attach Cable (DAC)	See Notes column	No	Intel has verified functionality with: • Amphenol SF-NDAQGF100G-003M • Cisco QSFP-4SFP25G-CU2M

Notes about PSM4 transceivers:

- Parallel Single Mode 4-Channel (PSM4) is a type of single-mode transceiver using parallel fiber designs consisting of four channels.
- The E810-CQDA2T adapter can connect two PSM4 transceivers and provide a total of eight 10 GbE signals via an MPO breakout cable. The Intel PSM4 transceiver has gone through extensive Intel qualification and is certified for use with the E810-CQDA2T.

2.5.1 E810-CQDA2T Support for PSM4 Modules Including Breakout Cables

To install the Intel 100G PSM4 Optical Transceiver:

1. Locate the QSFP28 cage slot on the host or switch equipment.
2. With the optical module oriented with the pull-tab on top, firmly push the optical transceiver into the QSFP28 cage slot on the host or switch equipment.
3. For QSFP28 modules with an optical receptacle-type connector, plug the optical connector of the fiber optic cable into the optical connector receptacle of the QSFP28 module. For pigtailed products, plug the fiber connector on the end of the pigtail into the mating connector of the patch panel or fiber jumper of the data center fiber plant.

To remove the Intel 100G PSM4 Optical Transceiver:

1. Disconnect the optical fiber from the module's optical receptacle connector (for receptacle modules) or the optical fiber pigtail from the mating connector (for pigtailed modules).
2. Using the pull tab, pull the module perpendicular to the equipment faceplate until the transceiver is free of the QSFP28 cage slot.

3.0 Software, Firmware, and Drivers

3.1 Software Support/Packages

1. Check the related Feature Support Matrix before installation to ensure that the correct NVM and driver versions are supported.

   E810 Technical Library:

   https://www.intel.com/content/www/us/en/products/details/ethernet/800-controllers/e810-controllers/docs.html?s=Newest

   Operating System Scope for E810 Timing-Enhanced Adapters:

   The following table lists the operating system support for a given release starting with Release 26.8, which represents the first production release for the E810-CQDA2T.

Operating System	Software Release Version					
	26.8	27.1/ 27.2.1	27.5/ 27.6.1	27.7	28.0/28.1	28.2
Linux RHEL 7.9	SNV	X	SNV	SNV	SNV	SNV
Linux RHEL 8.4	X	X	X	X	SNV	SNV
Linux RHEL 8.5	X	X	X	X	X	X
Linux RHEL 8.6/9	---	---	X	X	X	X
Linux RHEL 8.7/9.1	---	---	---	---	X	X
Linux RHEL 8.8/9.2	---	---	---	---	---	X
Linux SLES 12 SP5	X	X	SNV	SNV	SNV	SNV
Linux SLES 15 SP3	X	X	X	X	X	X
Linux SLES 15 SP4	---	---	X	X	X	X
Linux SLES 15 SP5	---	---	---	---	---	X
Linux Stable Kernel version 3.x/ 4.x/5.x	SNV	SNV	SNV	SNV	SNV	SNV
Linux Ubuntu 18.04 LTS	SNV	SNV	SNV	SNV	SNV	SNV
Linux Ubuntu 20.04 LTS	SNV	X	X	X	X	X
Linux Ubuntu 22.04 LTS	---	---	SNV	SNV	SNV	X
UEFI 2.1/2.3/2.4/2.6/2.7/2.8	---	X	X	X	X	X
UEFI 2.9	---	---	X	X	X	X
VMware ESXi 7.0	---	X	X	X	X	X
VMware ESXi 8.0	---	---	---	---	X	X

Legend:

X	Supported with Intel® NVM and software driver.
---	Not supported with Intel® NVM and software driver.
SNV	Supported but Not Validated.
TBD	Available in a future release.

Refer to the *Intel® Ethernet Controller E810 Feature Support Matrix* for additional NVM and software driver details.:

https://cdrdv2.intel.com/v1/dl/getContent/607252

NVM Update Tool:

https://www.intel.com/content/www/us/en/download/19624/non-volatile-memory-nvm-update-utility-for-intel-ethernet-network-adapter-e810-series.html

Always choose the latest.

Note: To update the NVM, refer to the *Intel® Ethernet NVM Update Tool Quick Usage Guide for Linux*.

2. Download and install the complete driver package:

   https://www.intel.com/content/www/us/en/download/15084/intel-ethernet-adapter-complete-driver-pack.html

   Always choose the latest.

   Or, download the Linux driver only:

   https://sourceforge.net/projects/e1000/files/ice%20stable/

   Note: For the newest features, always use the latest driver and NVM version. This document is based on driver version *ice-1.12.7* or newer.

   Important: The latest ice-1.12.7 driver with the 4.31 NVM version has some limitations that will be addressed in later releases.

   For details, see the Release Notes:

   https://www.intel.com/content/www/us/en/download/19622/intel-ethernet-product-software-release-notes.html

3. Install the complete driver package with the following commands:

   ```
   # cd ice-1.x.x/src
   # make -j install
   ```

   Now reboot the system, or if not possible:

   ```
   # rmmod ice
   ```

   You might need to remove additional drivers that are using the ice driver:

   ```
   # insmod ./intel_auxiliary.ko   //Depends on the kernel version if you need it.
   # modprobe gnss                 //Depends on the kernel version if you need it.
   # insmod ./ice.ko
   ```

4. Update the NVM to the latest image:

   ```
   # cd E810_NVMUpdatePackage_v4_30_Linux/Linux_x64/
   #./nvmupdate64e -u -l -c nvmupdate.cfg
   ```

5. Power cycle the system and check correct driver and NVM version:

```
#ethtool -i ens801f0
driver: ice
version: 1.12.7
firmware-version: 4.31 0x8001af21 1.3429.0
expansion-rom-version:
bus-info: 0000:84:00.0
supports-statistics: yes
supports-test: yes
supports-eeprom-access: yes
supports-register-dump: yes
supports-priv-flags: yes
```

6. Configure the ports.

   a. The E810-CQDA2T supports 2x100 GbE or 8x10 GbE. To switch between the modes you must use the Ethernet Port configuration Tool (EPCT). It can be found under *\APPS\EPCT\Linux_x64* of the complete driver package.

```
# ./epct64e
Ethernet Port Configuration Tool
EPCT version: v1.38.03.06
Copyright 2019 - 2022 Intel Corporation.

Options:
    -? / -h / -help
      Displays command line help.
    -? (parameter) / -h (parameter) / -help (parameter)
      Usage help for specific command line parameter.
    -devices [parameter]
     Displays supported devices present in the system. If 'branding' parameter
     is specified branding view is presented.
    -get
      Displays port configuration for specified device.
    -nic=(device_index)
      Selects device with specified index.
    -set (option)
      Configures device with specified option.
    -location (Segment:Bus)
      Selects device with specified PCIe Address.

All actions succeeded.
```

   b. To update, you must first find the E810-CQDA2T adapter. In this example, it is NIC 2 (in 2x100 Gbps mode).

```
# ./epct64e -devices
Ethernet Port Configuration Tool
EPCT version: v1.38.03.06
Copyright 2019 - 2022 Intel Corporation.

NIC Seg:Bus:Fun   Ven-Dev   Connector Ports Speed    Quads  Lanes per PF
=== ============= ========= ========= ===== ======== ====== ============
 1) 000:024:00-03 8086-1593 SFP       4     25  Gbps Single 1
 2) 000:175:00-07 8086-1592 QSFP      2     100 Gbps Dual   4
 3) 000:178:00-03 8086-1593 SFP       4     25  Gbps Single 1
```

```
Warning: Any changes to the port option configuration will require a reboot
before the device will function correctly.

All actions succeeded.
```

c. Check the current and available configuration: (active 2 x 1 x 100)

```
#./epct64e -nic 2 -get
Ethernet Port Configuration Tool
EPCT version: v1.38.03.06
Copyright 2019 - 2022 Intel Corporation.

Available Port Options:
========================================================================
        Port                            Quad 0           Quad 1
Option  Option (Gbps)                   L0  L1  L2  L3   L4  L5  L6  L7
======= ============================    ================ ================
Active  2x1x100                      -> 100   -   -   -  100   -   -   -
        8x10                         ->  10  10  10  10   10  10  10  10

Warning: Any changes to the port option configuration will require a reboot
before the device will function correctly.

All actions succeeded.
```

d. Configure the NIC to the new port configuration: (8x10) **Need to restart the system**.

```
#./epct64e -nic 2 -set 8x10
Ethernet Port Configuration Tool
EPCT version: v1.38.03.06
Copyright 2019 - 2022 Intel Corporation.

New configuration was set: 8x10
Restart the system to apply the changes.

The port options have changed for this device. You must reboot for the device
to function correctly.

All actions succeeded.
```

3.2 Building a linuxptp Project

1. Download the latest **linuxptp** release from SourceForge (v4.0 or later):

   https://sourceforge.net/projects/linuxptp/files/latest/download

   or use:

   ```
   # git clone git://git.code.sf.net/p/linuxptp/code linuxptp-4.1
   ```

2. Extract the downloaded archive, if required:

   ```
   # tar xzf linuxptp-4.1.tgz
   ```

3. Navigate to the *linuxptp-4.1* directory and compile the source code:

   ```
   # cd linuxptp-4.1
   # make
   ```

4. To install the program and related man pages into */usr/local*, run `make install` with root privileges.

   ```
   # make install
   ```

   This enables you to run the tools from any directory.

5. For more details related to installing **linuxptp**, go to:

   http://linuxptp.sourceforge.net/

3.3 Related linuxptp Information

Following are links to related information:

- https://github.com/richardcochran/linuxptp/blob/master/phc2sys.8
- https://github.com/richardcochran/linuxptp/blob/master/ptp4l.8
- https://github.com/richardcochran/linuxptp/blob/master/ts2phc.8
- https://www.mankier.com/8/ts2phc
- https://www.mankier.com/8/ptp4l
- https://www.mankier.com/8/phc2sys

3.4 Building a synce4l Tool

1. Download the latest **synce4l** from: https://github.com/intel/synce4l

   or use:

   ```
   #git clone http://github.com/intel/synce4l synce4l
   ```

2. Navigate to the extracted directory and compile the source code:

   ```
   # cd synce4l
   # make
   ```

3. To install the program and related man pages into */usr/local*, run `make install` with root privileges.

   ```
   # make install
   ```

   This enables you to run the tools from any directory.

4. For more details related to installing **synce4l**, go to:

   https://github.com/intel/synce4l

4.0 Configuring the E810-CQDA2T Using the Linux Kernel Interface

Note: The way we communicate with the Linux kernel interface might change in a later revision of the driver.

4.1 Introduction

The Linux kernel provides the standard interface for controlling external synchronization pins. Older versions that do not have this feature are not supported. Only versions with the PTP pins interface implemented are supported.

The list of supported operating system can be found in the following document:

https://cdrdv2.intel.com/v1/dl/getContent/607252

To check if the kernel has the required PTP and pin interface, run the following command:

```
# cat /proc/kallsyms | grep ptp_pin
```

After installing the latest E810 network driver, users are able to find the pin interface in the corresponding PTP device under **sysfs**. Users can find it by navigating through multiple paths, depending on the Linux distribution being used.

For a listing of all E810-CQDA2T adapters, run the following:

```
# grep 000f /sys/class/net/*/device/subsystem_device | awk -F"/" '{print $5}'
enp1s0f0
enp1s0f1
```

The example above shows one network device with four ports (f0-f1).

To run scripts in this section, set ETH to point to the first port of the adapter:

```
# export ETH=enp1s0f0
```

Or:

```
# export ETH=`grep 000f /sys/class/net/*/device/subsystem_device | awk -F"/" '{print
$5}' | head -n 1`
```

Note: This alternate command can only be used if one E810-CQDA2T is running in your system. Otherwise, the script must be amended appropriately.

Some scripts also use the PCI_SLOT. Users can easily set it up by running the following:

```
# export PCI_SLOT=`grep PCI_SLOT_NAME /sys/class/net/$ETH/device/uevent | cut -c 15-`
```

In the following example, the driver exposes the `ptp7` device:

```
#ls -R /sys/class/ptp/*/pins/

/sys/class/ptp/ptp7/pins/:

GNSS SMA1 SMA2 U.FL1 U.FL2
```

In the following example, the `ens260f0` net interface exposes pins through the `ptp7` interface:

```
#ls -R /sys/class/net/*/device/ptp/*/pins

/sys/class/net/ens260f0/device/ptp/ptp7/pins:

GNSS SMA1 SMA2 U.FL1 U.FL2
```

Users can also run **ethtool -T** <*interface name*> to show the PTP clock number.

```
# ethtool -T <interface name>

PTP Hardware Clock: 7
```

The E810 only has one hardware timer shared between all ports. As a result, users find the PTP clock number only on Port 0.

If users need to use bonding or DPDK, do not use Port 0, as this prevents the use of Linux PHC API for the device. A better solution is to use any other port for this functionality or to use a virtual function for DPDK.

4.2 DPLL Priority

The E810-CQDA2T automatically switches reference inputs according to the default DPLL priority list, as shown in Table 7.

where:

Pin index	=	DPLL device physical pin index
EEC - DPLL0	=	Ethernet equipment clock source from DPLL0 for frequency adjustments., glitchless.
PPS - DPLL1	=	1PPS generation from DPLL1 for phase adjustments. Glitches allowed. Slower locking.

Note: The DPLL priority list can be changed. See Section 4.11, “Advanced DPLL Configuration”.

Note: From firmware version 4.20 or newer, the DPLL priority list and the DPLL configuration parameters have been change to meet the ITU-T specs. The firmware will update the needed parameters based on the incoming timing signals. SDP20 is expecting a 10 MHz signal.

Note: The input references are checked to ensure that they meet specified criteria before they are fed to the DPLL. Each reference has several monitoring circuits and the one of particular interest is called Precise Frequency Monitor (PFM). By default, input reference is measured in PFM for 10 seconds to avoid disqualifying a reference with jitter/wander, which is still acceptable by standards. This requirement originates from Telcordia GR-1244 standard. Though this requirement is not used in ITU specs, many telecom customers find this feature very useful. Only inputs that meet this 10 seconds of acceptable input are be seen as “valid”. All others are be seen as “invalid”.

Table 7. DPLL Priority List

Default Priority	Pin Index	EEC - DPLL0 (Frequency/Glitchless)	Frequency	PPS - DPLL1 (Phase/Glitch Allowed)	Frequency
0	6	1PPS from GNSS (GNSS-1PPS)	1PPS	1PPS from GNSS (GNSS-1PPS)	1PPS
2	5	1PPS from SMA2 (SMA2)	1PPS	1PPS from SMA2 (SMA2)	1PPS
3	4	1PPS from SMA1 (SMA1)	1PPS	1PPS from SMA1 (SMA1)	1PPS
4	1	Reserved	--	1PPS from E810 (CVL-SDP20)	10 MHz
5	0	Reserved	--	1PPS from E810 (CVL-SDP22)	1PPS
6	--	Reserved	--	Reserved	--
7	--	Reserved	--	Reserved	--
8	2	Recovered CLK1 (C827_0-RCLKA)	1.953125 MHz	Recovered CLK1 (C827_0-RCLKA)	1.953125 MHz
9	3	Recovered CLK2 (C827_0-RCLKB)	1.953125 MHz	Recovered CLK2 (C827_0-RCLKB)	1.953125 MHz
10	--	OCXO	20.000 MHz	OCXO	20.000 MHz

4.3 External Connectors

External connector configuration is available only on the port that owns the PTP timer. By default, it is Port 0.

The E810-CQDA2T has four connectors for external 1PPS signals: SMA1, SMA2, U.FL1, and U.FL2

- SMA connectors are bidirectional and U.FL are unidirectional.
- U.FL1 is 1PPS output and U.FL2 is 1PPS input.
- SMA1 and U.FL1 connectors share channel one.
- SMA2 and U.FL2 connectors share channel two.

Example:

```
# export ETH=enp1s0f0
echo <function> <channel> > /sys/class/net/$ETH/device/ptp/*/pins/SMA1
(SMA2,U.FL1,U.FL2)
```

where:

function: 0 = Disabled
1 = Rx
2 = Tx

channel: 1 = SMA1 or U.FL1
2 = SMA2 or U.FL2

Note: If different input or output frequencies are needed, please check details in Section 4.11, "Advanced DPLL Configuration"

4.4 Channel 1 Configurations

1. SMA1 as 1PPS input:

```
# echo 1 1 > /sys/class/net/$ETH/device/ptp/ptp*/pins/SMA1
# dmesg
[792194.583302] ice 0000:03:00.0: SMA1 RX
[792194.583304] ice 0000:03:00.0: SMA2 <current_state> U.FL2 <current_state>
```

2. SMA1 as 1PPS output:

```
# echo 2 1 > /sys/class/net/$ETH/device/ptp/ptp*/pins/SMA1
# dmesg
[792505.312096] ice 0000:03:00.0: SMA1 TX
[792505.312098] ice 0000:03:00.0: SMA2 <current_state> U.FL2 <current_state>
```

   Note: Setting SMA1 as TX automatically enables U.FL1 as RX.

3. U.FL1 as 1PPS output:

```
# echo 2 1 > /sys/class/net/$ETH/device/ptp/ptp*/pins/U.FL1
# dmesg
[792745.238452] ice 0000:03:00.0: SMA1 RX + uFL1 TX
[792745.238453] ice 0000:03:00.0: SMA2 <current_state> U.FL2 <current_state>
```

   Note: Setting U.FL1 as a TX automatically enables SMA1 as RX.

4. SMA1 as 1PPS input and U.FL1 as 1PPS output:

```
# echo 1 1 > /sys/class/net/$ETH/device/ptp/ptp*/pins/SMA1
echo 2 1 > /sys/class/net/$ETH/device/ptp/ptp*/pins/U.FL1
# dmesg
[27158.812512] ice 0000:03:00.0: SMA1 RX + uFL1 TX
[27158.812519] ice 0000:03:00.0: SMA2 <current_state> U.FL2 <current_state>
```

5. Disable SMA1:

```
# echo 0 1 > /sys/class/net/$ETH/device/ptp/ptp*/pins/SMA1
# dmesg
[793017.697870] ice 0000:03:00.0: SMA1 + U.FL1 disabled
SMA2 <current_state> U.FL2 <current_state>
```

   Note: Users must first disable U.FL1, as it shares the same channel number.

6. Disable U.FL1:

```
# echo 0 1 > /sys/class/net/$ETH/device/ptp/ptp*/pins/U.FL1
# dmesg
[793017.697870] ice 0000:03:00.0: SMA1 + U.FL1 disabled
[793017.812519] ice 0000:03:00.0: SMA2 <current_state> U.FL2 <current_state>
```

   Note: Users must first disable SMA1, as it shares the same channel number.

   Note: To compensate for path delay in the network, users can implement phase delay in Channel 1 using command (`input pin 4 is SMA1. 7000 is equal to 7 ns`):

```
# echo "in pin 4 phase_delay 7000" > /sys/class/net/$ETH/device/pin_cfg
```

4.5 Channel 2 Configurations

1. SMA2 as 1PPS input:

```
# echo 1 2 > /sys/class/net/$ETH/device/ptp/ptp*/pins/SMA2
# dmesg
[27158.812512] ice 0000:03:00.0: SMA1 <current_state> U.FL1 <current_state>
[27158.812519] ice 0000:03:00.0: SMA2 RX
```

2. SMA2 as 1PPS output:

```
# echo 2 2 > /sys/class/net/$ETH/device/ptp/ptp*/pins/SMA2
# dmesg
[27158.812512] ice 0000:03:00.0: SMA1 <current_state> U.FL1 <current_state>
[27158.812519] ice 0000:03:00.0: SMA2 TX
```

3. U.FL2 as 1PPS input:

```
# echo 1 2 > /sys/class/net/$ETH/device/ptp/ptp*/pins/U.FL2
# dmesg
[27158.812512] ice 0000:03:00.0: SMA1 <current_state> U.FL1 <current_state>
[27158.812519] ice 0000:03:00.0: SMA2 <current_state> U.FL2 RX
```

4. SMA2 as 1PPS out and U.FL2 as 1PPS in:

```
# echo 2 2 > /sys/class/net/$ETH/device/ptp/ptp*/pins/SMA2
# echo 1 2 > /sys/class/net/$ETH/device/ptp/ptp*/pins/U.FL2
# dmesg
[27158.812512] ice 0000:03:00.0: SMA1 <current_state> U.FL1 <current_state>
[27158.812519] ice 0000:03:00.0: SMA2 TX + U.FL2 RX
```

5. Disable SMA2:

```
# echo 0 2 > /sys/class/net/$ETH/device/ptp/ptp*/pins/SMA2
# dmesg
[27158.812512] ice 0000:03:00.0: SMA1 <current_state> U.FL1 <current_state>
[27158.812512] ice 0000:03:00.0: SMA2 + U.FL2 disabled
```

   Note: Users must first disable U.FL2, as it shares the same channel number.

6. Disable U.FL2:

```
# echo 0 2 > /sys/class/net/$ETH/device/ptp/ptp*/pins/U.FL2
# dmesg
[27158.812512] ice 0000:03:00.0: SMA1 <current_state> U.FL1 <current_state>
[27158.812512] ice 0000:03:00.0: SMA2 + U.FL2 disabled
```

   Note: Users must first disable SMA2 as it shares the same channel number.

   Note: To compensate for path delay in the network, users can implement phase delay in Channel 1 using command (`input pin 4 is SMA1. 7000 is equal to 7 ns`):

```
# echo "in pin 4 phase_delay 7000" > /sys/class/net/$ETH/device/pin_cfg
```

4.6 Recovered Clocks (G.8261 SyncE Support)

Recovered clocks can be configured using a special **sysfs** interface that is exposed by every port instance. Writing to a **sysfs** under a given port automatically enables a recovered clock from a given port that is valid for a current link speed. A link speed change requires repeating the steps to enable the recovered clock.

If a port recovered clock is enabled and no higher-priority clock is enabled at the same time, the DPLL starts tuning its frequency to the recovered clock reference frequency enabling G.8261 functionality.

There are two recovered clock outputs from each of the C827 PHYs. Only one pin can be assigned to one of the ports. Recovered clocks from Port 0-3 are assigned to C827 PHY0 and Port 4-7 to C827 PHY1. Re-enabling the same pin on a different port automatically disables it for the previously-assigned port.

Note: In the current version, if you switch from a 1PPS clock to recovered clock (SyncE) you might experience up to 200 ns deviation in the 1PPS signal.

1. To enable a recovered clock for a given Ethernet device run the following:

```
# echo <ena> <pin> > /sys/class/net/$ETH/device/phy/synce
```

   where:

   ena: 0 = Disable the given recovered clock pin.
   1 = Enable the given recovered clock pin.

   pin: 0 = Enable C827_0-RCLKA (Port 0-3) / C827_1-RCLKA (Port 4 -7) (higher priority pin).
   1 = Enable C827_0-RCLKB (Port 0-3) / C827_1-RCLKB (Port 4 -7) (lower priority pin).

   For example, to enable the higher-priority recovered clock from Port 0 and a lower-priority recovered clock from Port 1, run the following:

```
# export ETH0=enp1s0f0
# export ETH1=enp1s0f1
# export ETH3=enp1s0f3
# export ETH4=enp1s0f4

# echo 1 0 > /sys/class/net/$ETH0/device/phy/synce
# dmesg
[27575.495705] ice 0000:03:00.0: Enabled recovered clock: pin C827_0-RCLKA
# echo 1 1 > /sys/class/net/$ETH1/device/phy/synce
# dmesg
[27575.495705] ice 0000:03:00.1: Enabled recovered clock: pin C827_0-RCLKB
# echo 1 0 > /sys/class/net/$ETH3/device/phy/synce
# dmesg
[27575.495705] ice 0000:03:00.3: Enabled recovered clock: pin C827_1-RCLKA
# echo 1 1 > /sys/class/net/$ETH1/device/phy/synce
# dmesg
[27575.495705] ice 0000:03:00.4: Enabled recovered clock: pin C827_1-RCLKB
```

   Disable recovered clocks:

```
# echo 0 0 > /sys/class/net/$ETH0/device/phy/synce
# dmesg
[27730.341153] ice 0000:03:00.0: Disabled recovered clock: pin C827_0-RCLKA
# echo 0 1 > /sys/class/net/$ETH1/device/phy/synce
# dmesg
[27730.341153] ice 0000:03:00.1: Disabled recovered clock: pin C827_0-RCLKB
```

```
# echo 0 0 > /sys/class/net/$ETH3/device/phy/synce
# dmesg
[27730.341153] ice 0000:03:00.3: Disabled recovered clock: pin C827_1-RCLKA
# echo 0 1 > /sys/class/net/$ETH4/device/phy/synce
# dmesg
[27730.341153] ice 0000:03:00.4: Disabled recovered clock: pin C827_1-RCLKB
```

Check recovered clock status. You can add the current status of the recovered clock to the *dmesg*:

```
#echo dump rclk_status > /sys/kernel/debug/ice/0000:03:00.0/command
# dmesg
[311274.298749] ice 0000:03:00.0: State for port 0, C827_0-RCLKA: Disabled
[311274.300060] ice 0000:03:00.0: State for port 0, C827_0-RCLKB: Disabled
```

Note: In secure boot case, check the pin_cfg file. For details, see Section 4.11.1.

Note: enp1s0f0, enp1s0f1, etc, refer to the physical function. In the case of the E810-CQDA2T it has what is called inverted PF-to-Port mapping. This means that functions 0-3 correspond to quad 1 and functions 4-7 correspond to quad 0. The PHY lanes, that is the physical Tx and Rx differential pairs, on the E810-CQDA2T are connected to the quad 0 and quad 1 cages. See the following table:

Inverted Port Mapping	
Physical Function	**PHY (Lane)**
PF0	L7
PF1	L6
PF2	L5
PF3	L4
PF4	L3
PF5	L2
PF6	L1
PF7	L0

L4-L7 correspond to lanes 4 through 7, which correspond to quad 1. Therefore PF0-PF3 are driving quad 1. L0-L3 correspond to lanes 0 through 3, which correspond to quad 0. Therefore PF4-PF7 are driving quad 0. For example if the desired recovered clock is from L4, it corresponds to PF3.

4.7 External Timestamp Signals

The E810-CQDA2T can use external 1PPS signals filtered out by the DPLL as its own time reference.

When the DPLL is synchronized to the GNSS module or an external 1PPS source, the **ts2phc** tool can be used to synchronize the time to the 1PPS signal. The configuration file needs to be edited before using the application. You must to remember to change `network_interface`, accordingly to his device's name.

Note: External Timestamp (extts) event support is only enabled on Port 0.

```
# export ETH=enp1s0f0
# export TS2PHC_CONFIG=/home/<USER>/linuxptp-3.1/configs/ts2phc-generic.cfg
# ts2phc -f $TS2PHC_CONFIG -s generic -m -c $ETH
```

```
# cat $TS2PHC_CONFIG
[global]
use_syslog 0
verbose 1
logging_level 7
ts2phc.pulsewidth 100000000
#For GNSS module
#ts2phc.nmea_serialport  /dev/gnssX #/dev/gnssX, where X - GNSS device number
#leapfile /../<path_to_.list_leap_second_file>
[<network interface>]
ts2phc.extts_polarity
rising
```

Note: The **phc2sys** tool should not be run at the same time as **ts2phc** using the generic source of ToD (`-s generic`). In the default configuration, **ts2phc** uses hardware-generated timestamps along with the system timer to create correction values. Running the tools in parallel can create a feedback that breaks time synchronization. The **leapfile** option is available but not necessary for the program to run. Also, the default *.leap* file is not compatible with **ts2phc**.

Note: You might want filter the 1PPS timestamps till the DPLL locked. See Section 4.12, "1PPS Signals from E810 Device to DPLL".

4.8 Periodic Outputs from DPLL (SMA and U.FL Pins)

The E810-CQDA2T supports two periodic output channels (SMA1 or U.FL1 and SMA2). Channels can be enabled independently and output 1PPS generated by the embedded DPLL. 1PPS outputs are synchronized to the reference input driving the DPLL1. Users can read the current reference signal driving the 1PPS subsystem by running the following command:

```
# dmesg | grep <DPLL1> state changed" | grep locked | tail -1
[ 342.850270] ice 0000:01:00.0: DPLL1 state changed to: locked, pin GNSS-1PPS
```

The following configurations of 1PPS outputs are supported:

1. SMA1 as 1PPS output:

   ```
   # echo 2 1 > /sys/class/net/$ETH/device/ptp/ptp*/pins/SMA1
   ```

2. U.FL1 as 1PPS output:

   ```
   # echo 2 1 > /sys/class/net/$ETH/device/ptp/ptp*/pins/U.FL1
   ```

3. SMA2 as 1PPS output:

```
# echo 2 2 > /sys/class/net/$ETH/device/ptp/ptp*/pins/SMA2
```

Note: Configurations (1 and 3) or (2 and 3) can be enabled at the same time, but not (1, 2, and 3).

4.9 Reading Status of the DPLL

The E810-CQDA2T driver exposes a simple **debugfs** interface that enables monitoring of the on-board DPLL device state.

Note: The main purpose of this interface is for ***debug only***. The **debugfs** interface is not available when using Secure Boot option. All of the information is available using the **pin_cfg** or **dpll_<*X*>_ref_pin/dpll_<*X*>_state** option as detailed in Section 4.11, "Advanced DPLL Configuration".

Note: Because of a known limitation, the **Phase offset** field might show incorrect values. This will be fixed in the future releases.

```
# export ETH=enp1s0f0
# export PCI_SLOT=`grep PCI_SLOT_NAME /sys/class/net/$ETH/device/uevent | cut -c 15-`
#
# cat /sys/kernel/debug/ice/0000:82:00.0/cgu
Found ZL80032 CGU
DPLL Config ver: 1.3.0.1
DPLL FW ver: 6201

CGU Input status:
                    |            |      priority     |            |
      input (idx)   |    state   | EEC (0) | PPS (1) | eSync fail |
  ------------------------------------------------------------------
     CVL-SDP22 (0)  |    invalid |     255 |       5 |    N/A     |
     CVL-SDP20 (1)  |    invalid |     255 |       4 |    N/A     |
  C827_0-RCLKA (2)  |    invalid |       8 |       8 |    N/A     |
  C827_0-RCLKB (3)  |    invalid |       9 |       9 |    N/A     |
          SMA1 (4)  |    invalid |       3 |       3 |    N/A     |
    SMA2/U.FL2 (5)  |    invalid |       2 |       2 |    N/A     |
     GNSS-1PPS (6)  |      valid |       0 |       0 |    N/A     |

EEC DPLL:
    Current reference:  GNSS-1PPS
    Status:             locked_ho_acq

PPS DPLL:
    Current reference:  GNSS-1PPS
    Status:             locked_ho_acq
    Phase offset [ns]:  -717
```

The first section of the log shows the status of CGU inputs (references), including its index number. Active references currently selected are listed in Section 4.2. EEC Ethernet equipment clock (DPLL0) skips the 1PPS signal received on the CVL-SDP20 pin.

The second section lists all internal DPLL units. ECC (DPLL0) driving the internal clocks and PPS (DPLL1) driving all 1PPS signals.

The E810-CQDA2T supports embedded sync (eSync), including embedded pulse per second (ePPS),

The third section talks about the eSync, if it is enabled, as detailed in Section 4.11, "Advanced DPLL Configuration".

Lock times for each of the DPLLs are different and are subject to change and can take a long time to lock depending on their initial synchronization.

Status options: **invalid**, **freerun**, **locked**, **locked_ho_acq** (locked and holdover acquired), **holdover**, and **uninitialized**.

The Phase offset value helps to find out how well the PPS DPLL is synchronized (in 100 ps units)

Note: The DPLL firmware version should be 6201 for all production units. If you have a different version, please contact your Intel representative.

4.10 DPLL Monitoring

In the default configuration, the E810-CQDA2T driver enables monitoring of the DPLL events and reports state changes in the default system log (*dmesg*) with the WARN level independently for each of the DPLL units.

DPLLs start in a holdover mode and enter an unlocked and locked state when a valid reference input is enabled. If the current input becomes invalid, DPLLs change state to holdover. When the reference reappears (or a different valid input is present), the DPLL state changes to the unlocked state and locks to a new signal.

```
# dmesg | grep "state changed"
[23.740121] ice 0000:01:00.0: <DPLL0> state changed to: holdover, pin CVL-SDP22
[23.740833] ice 0000:01:00.0: <DPLL1> state changed to: holdover, pin CVL-SDP22
#GNSS signal appeared
[57.857575] ice 0000:01:00.0: <DPLL0> state changed to: unlocked, pin GNSS-1PPS
[57.858088] ice 0000:01:00.0: <DPLL1> state changed to: unlocked, pin GNSS-1PPS
[119.810415] ice 0000:01:00.0: <DPLL0> state changed to: locked, pin GNSS-1PPS
[178.818056] ice 0000:01:00.0: <DPLL1> state changed to: locked, pin GNSS-1PPS
#GNSS signal lost:
[18.833552] ice 0000:01:00.0: <DPLL0> state changed to: holdover, pin GNSS-1PPS
[18.834065] ice 0000:01:00.0: <DPLL1> state changed to: holdover, pin GNSS-1PPS
#GNSS signal re-appeared:
[19.825608] ice 0000:01:00.0: <DPLL0> state changed to: unlocked, pin GNSS-1PPS
[19.826122] ice 0000:01:00.0: <DPLL1> state changed to: unlocked, pin GNSS-1PPS
[21.841638] ice 0000:01:00.0: <DPLL0> state changed to: locked, pin GNSS-1PPS
[21.850270] ice 0000:01:00.0: <DPLL1> state changed to: locked, pin GNSS-1PPS
```

DPLL monitoring can be enabled (on) or disabled (off) by using the **ethtool** command in the Linux kernel.

```
# export ETH=ens801f0

ethtool --show-priv-flags $ETH

Private flags for ens801f0:

link-down-on-close           : off
fw-lldp-agent                : off
channel-inline-flow-director : off
channel-inline-fd-mark       : off
channel-pkt-inspect-optimize : on
channel-pkt-clean-bp-stop    : off
channel-pkt-clean-bp-stop-cfg: off
vf-true-promisc-support      : off
mdd-auto-reset-vf            : off
vf-vlan-prune-disable        : off
```

```
legacy-rx                    : off
cgu_fast_lock: off
dpll_monitor                 : on
extts_filter                 : off
itu_g8262_filter_used:off
allow-no-fec-modules-in-auto : off
```

Enabling DPLL monitoring:

```
ethtool --set-priv-flags $ETH dpll_monitor on
```

Disabling DPLL monitoring:

```
ethtool --set-priv-flags $ETH dpll_monitor  off
```

4.11 Advanced DPLL Configuration

4.11.1 pin_cfg User Readable Format

The DPLL on the E810-CQDA2T offer some advanced configuration options. These options are not needed on regular applications and can cause problems. Please use these commands with extra care. The DPLL will go back to the default values after a power cycle of the adapter.

The E810-CQDA2T supports embedded sync (eSync), including embedded pulse per second (ePPS), but not embedded pulse per two seconds (ePP2S).

Timing signals on the SMAs can be configured as inputs or outputs, typically configured for one pulse per second (1PPS) operation, but they will support a 10 MHz or 25 MHz signal (with or without the embedded 1PPS eSync).

Note: Do not change any other output pin than 0 and 1.

The Ref-sync pair (e_ref_sync) feature uses two DPLL inputs, one is used as a reference clock (typically higher than 1 KHz) and a sync signal (1PPS). This feature allows a wider loop bandwidth resulting in much faster DPLL lock time and was also empirically found to be required to pass ITU-T certification. This Ref-sync_pair feature can be enabled on SMA or SDP (input pin 1 for SDP (pairing pin 0 & 1) and pin 5 for SMA (pairing pin 4 &5)

Note: In the *pin_cfg* the eSync/Ref-sync column 0: both eSync and e_ref_sync are disabled, 1: eSync is enabled, 2: Ref-sync_pair is enabled.

To check the DPLL pin configuration:

```
# cat /sys/class/net/ens4f0/device/pin_cfg

in
| pin| enabled|   state|       freq| phase_delay| eSync/Ref-sync| DPLL0 prio| DPLL1 prio|
|   0|       1| invalid|          1|           0|              0|        255|          5|
|   1|       1| invalid|   10000000|           0|              2|        255|          4|
|   2|       1| invalid|    1953125|           0|              0|          8|          8|
|   3|       1| invalid|    1953125|           0|              0|          9|          9|
|   4|       1| invalid|    1953125|           0|              0|         10|         10|
|   5|       1| invalid|    1953125|           0|              0|         11|         11|
|   6|       1| invalid|          1|        7000|              0|          3|          3|
|   7|       1| invalid|          1|        7000|              2|          2|          2|
|   8|       1| invalid|          1|           0|              0|          0|          0|
```

```
out
| pin| enabled| dpll|       freq| esync|
|   0|       1|    1|          1|     0|
|   1|       1|    1|          1|     0|
|   2|       1|    0|  156250000|     0|
|   3|       1|    0|  156250000|     0|
|   4|       1|    0|  156250000|     0|
|   5|       1|    1|          1|     0|
|   6|       1|    1|          1|     0|
```

In the “in” table. the pin numbers are referred from the DPLL Priority See Section 4.2, “DPLL Priority”.

In the “out” table pin 0 is SMA1 pin 1 is SMA2, all the other values do not modify.

Changing the DPLL priority list:

```
# echo "prio <prio value> dpll <dpll index> pin <pin index>" > \ /sys/class/net/$ETH/
device/pin_cfg
```

where:

prio value = Desired priority of configured pin [0-9]

dpll index = Index of DPLL being configured [0:EEC (DPLL0), 1:PPS (DPLL1)]

pin index = Index of pin being configured [0-8]

Note: A prio value 15 disables the input for synchronization.

Example:

Set priority 1 for pin 3 on DPLL 0:

```
# export ETH=enp1s0f0
# echo "prio 1 dpll 0 pin 3" > /sys/class/net/$ETH/device/pin_cfg
```

Changing input/output pin configuration:

```
# echo "<direction> pin <pin index> <config>" > /sys/class/net/$ETH/device/pin_cfg
```

where:

direction = pin direction being configured [“in”: input pin, “out”: output pin]

pin index = index of pin being configured [for in 0-8 (see DPLL priority section); for out 0: SMA1 1: SMA2]

config = list of configuration parameters and values:

```
[ "freq <freq_value_in_Hz>",
  "phase_delay <phase_delay_value_in_ns>"  // NOT used for out,
  "eSync <0:disabled, 1:enabled>"
  "e_ref_sync <0:disabled, 1:eSync enabled 2:Ref_Sync enabled>"
  "enable <0:disabled, 1:enabled>" ]
```

Note: The **eSync** setting has meaning only with the 10 MHz frequency, you need to have eSync to have the same setting in both ends of the SMA. eSync can be enabled only for the MA's (input pin 4 / 5 and output pin 0 /1)

Example:

```
# export ETH=enp1s0f0
```

Set freq to 10 MHz on input pin 4: DPLL will lock only if 10 MHz signals arrive on SMA1 and it has been enabled for input.

```
# echo "in pin 4 freq 10000000" > /sys/class/net/$ETH/device/pin_cfg
```

Set freq to 10 MHz on output pin 1: SMA2 will drive 10 MHz signal with embedded 1PPS if it has been enabled for output.

```
# echo "out pin 1 freq 10000000 eSync 1" > /sys/class/net/$ETH/device/pin_cfg
```

Disable input pin 2: DPLL will ignore anything on pin 2.

```
# echo "in pin 2 enable 0" > /sys/class/net/<dev>/device/pin_cfg
```

Set Ref-Sync_pair on pin 0 & 1 (SDPs)

```
# echo "in pin 1 e_ref_sync 2" > /sys/class/net/$ETH/device/pin_cfg
```

Set freq to 1 Hz (1PPS) on input pin 4: DPLL will lock only if 10 MHz signals arrive on SMA1 and it has been enabled for input with phase delay of 4 ns.

```
# echo "in pin 4 freq 1 phase_delay 4000" > /sys/class/net/$ETH/device/pin_cfg
```

4.11.2 dpll_<X>_ref_pin/dpll_<X>_state Machine Readable Interface

```
# export ETH=enp1s0f0
```

To find out which pin the DPLL0 (EEC DPLL) is locked on, check the `dpll_0_ref_pin`:

```
# cat /sys/class/net/$ETH/device/dpll_0_ref_pin
```

To check the state of the DPLL0 (EEC DPLL), check the `dpll_0_state`:

```
# cat /sys/class/net/$ETH/device/dpll_0_state
```

```
DPLL_UNKNOWN = -1,
DPLL_INVALID =  0,
DPLL_FREERUN =  1,
DPLL_LOCKED =  2,
DPLL_LOCKED_HO_ACQ =  3,
DPLL_HOLDOVER =  4
```

To find out which pin the DPLL1 (PPS DPLL) is locked on, check the `dpll_1_ref_pin`:

```
# cat /sys/class/net/$ETH/device/dpll_1_ref_pin
```

To check the state of the DPLL1 (PPS DPLL), check the `dpll_1_state`:

```
# cat /sys/class/net/$ETH/device/dpll_1_state
```

The `dpll_0_state` interface used by **synce4l** as well.

Note: The user application can monitor the `dpll_<X>_state` and `dpll_<X>_ref_pin` to detect the DPLL status changes. These changes will be visible in the ***dmesg*** as well.

Note: The application can also check the DPLL name in the *dpll_<X>_name* file.

4.12 1PPS Signals from E810 Device to DPLL

The E810-CQDA2T implements two 1PPS signals coming out of the MAC (E810 device) to the DPLL. They serve as the frequency reference (CVL-SDP20) and as both phase and frequency reference (CVL-SDP22) signals.

To enable a periodic output, write five integers into the file: channel index, start time seconds, start time nanoseconds, period seconds, and period nanoseconds. To disable a periodic output, set all the seconds and nanoseconds values to zero.

Note: From firmware version 4.20 and onward, Intel recommends to use a 10 MHz signal on SDP20.

1. To enable the frequency reference pin 10 MHz (CVL-SDP20):

```
# echo 1 0 0 0 100 > /sys/class/net/$ETH/device/ptp/ptp*/period
```

2. To enable the phase and frequency reference pin (CVL-SDP22):

```
# echo 2 0 0 1 0 > /sys/class/net/$ETH/device/ptp/ptp*/period
```

3. To disable the phase reference pin (CVL-SDP20):

```
# echo 1 0 0 0 0 > /sys/class/net/$ETH/device/ptp/ptp*/period
```

4. To disable the phase and frequency reference pin (CVL-SDP22):

```
# echo 2 0 0 0 0 > /sys/class/net/$ETH/device/ptp/ptp*/period
```

Note: Because of a known driver limitation, before enabling the periodic outputs to the DPLL, make sure that your PHC is synchronized (wait for stable **ptp4l** connection with dual digit offset). If you need to do a jump on the PHC clock (like restarting **ptp4l**), you must to disable 1PPS and 10 MHz signal to DPLL. If the clock jumped, the periodic outputs to the DPLL might not ever show a "valid" state in the CGU Linux interface. To restore correct operation of DPLL, reload the driver:

```
# rmmod ice; modprobe ice
```

Note: Enabling the phase and frequency reference signal (CVL-SDP22) on the adapter without external reference (**ptp4l**) is not recommended and might cause a frequency drift.

4.13 1PPS Signals from the DPLL to E810 Device

The DPLL automatically delivers two 1PPS signals to the E810 device on pin 21 and 23. These signals can be used to synchronize the E810 to the DPLL phase with the ts2phc program. The E810 will capture the timestamp when the 1PPS signal arrives.

4.14 GNSS Module Interface

Intel recommends disabling UART1 and UART2 interfaces on the GNSS receiver. To do that, use:

1. Disable UART1 in RAM, and Flash:

```
# ubxtool -v 1 -w 1 -P 29.20 -z CFG-UART1-ENABLED,0,5
```

Or open two terminals and run:

```
# cat /dev/gnssX //Terminal 1

# echo -ne "\xb5\x62\x06\x8a\x09\x00\x00\x05\x00\x00\x05\x00\x52\x10\x00\x05\x80"
> /dev/gnssX //Terminal 2
```

2. Disable UART2 in RAM, and Flash:

```
# ubxtool -v 3 -w 1 -P 29.20 -z CFG-UART2-ENABLED,0,5
```

Or open two terminals and run:

```
# cat /dev/gnssX //Terminal 1

# echo -ne "\xb5\x62\x06\x8a\x09\x00\x00\x05\x00\x00\x05\x00\x53\x10\x00\x06\x83"
> /dev/gnssX //Terminal 2
```

The E810-CQDA2T driver implements GNSS interfaces for receiving NMEA messages from the optional GNSS module. The interface can be found in */dev/gnssX*, where *X* is the GNSS interface.

If you have only one adapter installed, the GNSS interface number will always be */dev/gnss0*.

To associate the GNSS interface number with the network interface name if you have multiple adapters in a system, use the following command:

```
# ls /sys/class/net/*/device/gnss/*

/sys/class/net/ens1f0/device/gnss/gnss1:
dev  device  power  subsystem  type  uevent
/sys/class/net/ens2f0/device/gnss/gnss0:
dev  device  power  subsystem  type  uevent

# cat /dev/gnss0
$GNRMC,182805.00,A,0.0,N,0.0,E,0.082,,050321,,,D,V*14
$GNVTG,,T,,M,0.082,N,0.152,K,D*34
$GNGGA,182805.00,0.0,N,0.0,E,2,12,0.57,11.1,M,32.7,M,,0000*7D
$GNGSA,A,3,18,20,26,23,16,29,07,15,27,10,,,1.07,0.57,0.90,1*09
$GNGSA,A,3,81,79,72,82,80,78,65,88,87,,,,1.07,0.57,0.90,2*0C
$GNGSA,A,3,02,30,04,36,11,,,,,,,,1.07,0.57,0.90,3*0E
[…]
```

Note: If two applications, like **gpsd** and **ts2phc** need to be run at the same time, you might encounter issues regarding one application taking over the GNSS interface. A workaround would be to run **gpspipe** on local port, modify the config file of **ts2phc**, and use a local port to provide access to NMEA messages through TCP:

```
# socat EXEC:'gpspipe -r' TCP-LISTEN:2948,reuseaddr,fork

# cat configs/ts2phc-nmea.cfg

...
ts2phc.nmea_remote_host 127.0.0.1
ts2phc.nmea_remote_port 2948
…
```

ts2phc is now capable of using the local port 2948 to retrieve NMEA messages.

Note: Make sure that you are using */dev/gnssX* in your *.cfg* file for **linuxptp** if using the **ts2phc** tool.

Note: The QS samples contain prototype firmware, which expired on December 26, 2021. Pre-production samples in use at that time stop (no longer sync/lock to GNSS) and output a text message stating that the firmware has expired. An update is required to resolve. See more details in the Dear Customer Letter.

Note: Intel recommends disabling UART1 and UART2 interfaces (see Appendix A).

4.15 GNSS Advanced Features

To achieve the best GM, additional configuration might be required, like survey-in, cable delay, antenna setting, and so on. This requires additional packages under Linux.

In addition to using the **cat** command, you can use 3rd party APIs to improve the position accuracy and timing accuracy, as well as optimize the overall performance of the GNSS module. One of the most popular GNSS Linux APIs is **gpsd**, which is Open Source and is support by community at:

https://gpsd.gitlab.io/gpsd/index.html

The **gpsd** API follows chip vendor original specifications and fully integrates into a rich GUI tool, **pygpsclient**. Furthermore, **gpsd** provides the command line tool **ubxtool**, which support all original configuration commands.

The optional GNSS module inside the E810-CQDA2T uses u-Blox 9-series chip ZED-F9T-00B. Before you start yours first investigation, we suggest you study most popular questions at:

https://gpsd.io/faq.html#willitwork

4.15.1 Prerequisites and Steps to Fully Enable GNSS Features

Prerequisites:

- Python3.9+ and pip3 20.2+
- **scons** 4.2.0 or later

Note: Ensure that these components are installed correctly before any other step. During the installation process, ensure that there are no hidden error messages. After the installation, check if scons is present.

```
(# scons -v)
```

Note: Before installing **gpsd** on your system, ensure that all parts of any previous installation have been removed. Do not mix **gpsd** parts from different sources. The **gpsd** clients and the server must be of the same version.

Steps to prepare system and install gpsd:

Note: For additional information, refer to https://gpsd.io/installation.html.

1. If you cannot find the */dev/gnss0* (depends on the kernel) file, ensure that you are using the E810-CQDA2T with the optional GNSS installed and the latest driver and NVM (see Section 3.1).
2. If you receive any error message from the above two commands, follow the trouble shooting guide to solve OS and kernel dependencies:

   https://gpsd.io/installation.html#_check_that_your_system_configuration_will_allow_gpsd_to_work
3. A minimum build of **gpsd** can run near bare metal; C runtime support is the only requirement. The test clients and various additional features have additional prerequisites found at:

   https://gpsd.io/installation.html#_check_your_installation_prerequisites

   Note: It is important to install system dependencies using your system package manager first, before installing SCons or Python dependencies.

   Note: On some systems, GPSD might not work properly as a service out of the box. If that is the case, check that your *systemd* configuration files are set correctly.

4. You might need to add additional repositories to get dependencies, based on the distribution you are using. For example, for Red Hat 8 use commands:

   ```
   # subscription-manager repos --enable codeready-builder-for-rhel-8-$(arch)-rpms

   # sudo dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
   ```

5. Suggested dependencies to install. Some dependencies might not be needed if all features are not needed.

   For Red Hat 8:

   ```
   # dnf -y install cairo-devel pkg-config python3-devel libffi-devel gobject-introspection-devel cairo-gobject-devel gtk3-devel libtinfo.so.5 dbus libusb qt5-qtnetworkauth llvm-toolset clang-analyzer libsmi* python3-pycodestyle python3-flake8 python3-pyflakes cppcheck dia valgrind asciidoctor dbus gnuplot libusb ncurses pps-tools* qt-creator pygobject3* python3-pyserial python3-serial gtk3* python3-matplotlib python-matplotlib* libtinfo5 libcairo2 libcairo2-dev libevent-devel python39-devel qt5-qtnetworkauth llvm-toolset clang-analyzer libsmi* python3-pycodestyle python3-flake8 python3-pyflakes cppcheck dia valgrind asciidoctor dbus gnuplot libusb ncurses pps-tools* qt-creator pygobject3* gtk3* python3-matplotlib* libtinfo5 python-pyserial texlive-full python3-gps --skip-broken
   ```

6. Install Python 3.9. For now, Python older than 3.9, and newer than 3.9 (like 3.10.6) are not recommended due to incompatible libraries. Version 3.9.14 is best. Install Python with `make altinstall`. You can also try the Python virtual environment, if you are confident in what you are doing.

7. Suggested Python dependencies:

   # python3.9 -m pip install pycairo pygobject matplotlib serial rouge coverage python-can pyserial gps wheel Pillow pygnssutils requests urllib3 idna charset-normalizer certify pyrtcm pynmeagps pyubx2

8. Setup your PYTHONPATH variable to */usr/local/lib/python3.9/site-packages* (most common), or wherever you have your python libraries, and install **scons**:

   ```
   # python setup.py install
   ```

9. Install gpsd:

   ```
   # git clone https://gitlab.com/gpsd/gpsd.git
   # cd gpsd
   # scons -c && scons
   # scons udev-install
   (optionsl) # python3.9 -m pip install pygpsclient
   ```

10. Check **gpsd** version with `# gpsd -V`, and check **ubxtool** version with `# ubxtool -V`. If you receive an error, at this point the usual reason is incorrectly set PATH and PYTHONPATH variable - make sure they are pointing to the correct directories.

11. (Optional) Configure the **gpsd** service daemon:

    a. edit */etc/sysconfig/gpsd* and add `OPTIONS="-G -n -p /dev/gnssX"` to include the GNSS interface.

12. (Optional) Restart the **gpsd** service daemon and check its status (`/dev/gnss0` as an example).

```
#systemctl restart gpsd
#systemctl status gpsd
? gpsd.service - GPS (Global Positioning System) Daemon
   Loaded: loaded (/usr/lib/systemd/system/gpsd.service; enabled; vendor preset:
enabled)
   Active: active (running) since Mon 2022-10-31 10:09:59 CST; 1s ago
  Process: 1856694 ExecStart=/usr/local/sbin/gpsd $GPSD_OPTIONS $OPTIONS $DEVICES
(code=exited, status=0/SUCCESS)
 Main PID: 1856695 (gpsd)
    Tasks: 1 (limit: 3355442)
   Memory: 904.0K
   CGroup: /system.slice/gpsd.service
           ??1856695 /usr/local/sbin/gpsd -G -n -p /dev/gnss0
```

Setup compliance verification:

Before starting the **gpsd** APIs, the Antenna setup and GNSS firmware must be done.

- Antenna compliance:
  - All the GNSS applications' performance is expected to get good quality of antenna signal so the software APIs can run on top of stable NMEA sentences.
  - Table 8 shows the Antenna frequency band for ZED-F9T-00B. You should follow the original vendor specification including supported band and circular polarization (the ZED-F9T-00B uses right-hand circular polarization or RHCP).

Table 8. Supported GNSS and Signals on ZED-F9T=00B

GPS / QZSS	GLONASS	Galileo	BeiDou	NavIC
L1C/A (1575.420 MHz)	L1OF (1602 MHz + k*562.5 kHz, k=-7,...,6)	E1-B/C (1575.420 MHz)	B1I (1561.098 MHz)	-
			B1C (1575.420 MHz)[1]	
L2C (1227.600 MHz)	L2OF (1246 MHz + k*437.5 kHz, k=-7,...,6)	E5b (1207.140 MHz)	B2I (1207.140 MHz)	-

1. B1I and B1C not to be enabled concurrently.

- Leverage GPSD APIs to improve the E810-CQDA2T with the optional GNSS accuracy:
  - **gpsctl**
    - **gpsctl** can switch a dual-mode GPS between NMEA and vendor-binary modes. The **gpsctl** API needs the *gosd.socket* service daemon running. Following is a basic device identification example. You can use `gpsctl -help` to get all command details.
    - With proper GPSD installation, the **gpsctl** can export terminal interface, as follows:

      ```
      # gpsctl /dev/gnss0
      ```

      where `/dev/gnss0` is identified as a u-Blox SW EXT CORE 1.00 (3fda8e), HW 00190000 at 9600 baud.

— **ubxtool**

ubxtool is an original vendor-supported tool that can make use of u-Blox vendor commands on support protocol version 29.20 to ZED-F9T-00B. If the 5G deploy environment does not allow you to use the Desktop environment, you can consider **ubxtool** as efficient CLI for local and remote access.

- The default setting of the E810-CQDA2T configuration minimizes NMEA sentence message for better timing usage. Only "GNRMC" and "GNGGA" are needed for 5G timing usage. For better understanding of NMEA sentence focus and its purpose, refer to:

  https://w3.cs.jmu.edu/bernstdh/web/common/help/nmea-sentences.php

- The ZED-F9T-00B module requires protocol 29.20, so it needs **ubxtool** v3.24+. Use the following command to double check the **ubxtool** you install:

```
# ubxtool -V

ubxtool: Version 3.25.1~dev
```

- Get u-Blox protocol version for your GNSS device

```
# ubxtool -t -w 5 -v 1 -p MON-VER
ubxtool: poll MON-VER

sent:
1667189283.3474
UBX-MON-VER:
  Poll request

1667189285.2572
UBX-MON-VER:
  swVersion EXT CORE 1.00 (3fda8e)
  hwVersion 00190000
  extension ROM BASE 0x118B2060
  extension FWVER=TIM 2.20
  extension PROTVER=29.20
  extension MOD=ZED-F9T
  extension GPS;GLO;GAL;BDS
  extension SBAS;QZSS
  extension NAVIC
WARNING:  protVer is 10.00, should be 29.20.  Hint: use option "-P 29.20"
```

4.15.2 GNSS Receiver Configuration Layers in the E810-XXVDA4T

Once you confirm that **gpsd** is running, either in the foreground or as a service, **ubxtool** can be run. The **ubxtool** command is capable of using five different "layers" to read from and save configuration. These layers are:

- 0 - RAM — Responsible for reading configuration on the E810-CQDA2T.
- 1 - Battery Backed RAM (BBR) — Responsible for saving single configuration items on the E810-CQDA2T.
- 2 - Flash Layer
- 5 - RAM, and FLASH — Responsible for saving configuration to two memory types with one command. Use this layer if you would like to write your configuration settings to RAM and FLASH at the same time.

- 7 - Default configuration

The E810-CQDA2T uses Layer 0 (RAM) to read RAM configuration from the device, and Layer 1 (BBR) to save configuration to the RAM of the device. If layers are used incorrectly, the **ubxtool** command will fail with a UBX-ACK-NAK response. The E810-CQDA2T also uses Layer 2 (FLASH), and Layer 7 (default configuration).

If you do not state the layer, the **-z** flag (change configuration item) writes to FLASH and RAM, while the **-g** flag reads from all the possible layers.

For more information about the use of **ubxtool**, and to get a list of possible configuration items, use the following command:

```
# ubxtool -h -v 3
```

4.15.3 Perform Antenna Status Check New Location Setup

The current active antenna status can be determined by polling the *UBX-MON-RF* message. If an antenna is connected, the initial state after power-up is "Active Antenna OK".

The antenna supervisor can be configured through the *CFG-HW-ANT_** configuration items. The current configuration of the active antenna supervisor can also be checked by polling the related *CFG-HW_ANT_** configuration items.

Antenna status (as reported in the *UBX-MON-RF* and *UBX-INF-NOTICE* messages) is not reported unless the antenna voltage control has been enabled. To enable it:

```
# ubxtool -v 1 -P 29.20 -z CFG-HW-ANT_CFG_VOLTCTRL,1
sent:
UBX-CFG-VALSET:
 version 0 layer 0x7 transaction 0x0 reserved 0
  layers (ram bbr flash) transaction (Transactionless)
    item CFG-HW-ANT_CFG_VOLTCTRL/0x10a3002e val 1

UBX-ACK-ACK:
  ACK to Class x06 (CFG) ID x8a (VALSET)
```

To check your antenna status:

```
# ubxtool -v 1 -P 29.20 -p MON-RF
ubxtool: poll MON-RF

sent:
UBX-MON-RF:
  Poll request

UBX-MON-RF:
 version 0 nBlocks 2 reserved1 0 0
   blockId 0 flags x0 antStatus 2 antPower 1 postStatus 0 reserved2 0 0 0 0
    noisePerMS 71 agcCnt 5616 jamInd 18 ofsI 17 magI 147 ofsQ 2 magQ 139
    reserved3 0 0 0
   blockId 1 flags x0 antStatus 2 antPower 1 postStatus 0 reserved2 0 0 0 0
    noisePerMS 45 agcCnt 5265 jamInd 26 ofsI 12 magI 164 ofsQ 5 magQ 160
    reserved3 0 0 0
```

Status of the antenna supervisor state machine (0x00=INIT,0x01=DONTKNOW, 0x02=OK,0x03=SHORT,0x04=OPEN)

Current power status of antenna (0x00=OFF,0x01=ON,0x02=DONTKNOW)

4.15.4 Enabling and Disabling Additional Constellations

Before enabling survey-in functionality to achieve precise time information, it is beneficial to enable additional constellations. With more constellations enabled, the GNSS receiver can access more satellites, allowing better precision with good sky view. It is up to the user which constellations should be enabled. It is recommended to enable at least the GALILEO constellation for better timing capabilities.

To enable constellations in RAM, use the following commands (this will take a few minutes for the receiver to see constellations):

ubxtool -P 29.20 -e BEIDOU,1 -w 2 - Chinese constellation

ubxtool -P 29.20 -e GLONASS,1 -w 2 - Russian constellation

ubxtool -P 29.20 -e GPS,1 -w 2 - American constellation

ubxtool -P 29.20 -e SBAS,1 -w 2 - Regional Satellite-based Augmentation Systems

ubxtool -P 29.20 -e GALILEO,1 -w 2 - European constellation

To disable GLONASS constellation in RAM, use the following command:

ubxtool -P 29.20 -d GLONASS,1 -w 2

To check enabled constellations in all layers, use the following command:

ubxtool -v 1 -P 29.20 -g CFG-SIGNAL -v 1 -w 2

4.15.5 Perform Survey-In for New Location Setup

To achieve precise time information, the GNSS module needs to do a process “survey-in”. This process can take up to 24 hours to achieve best precision. Depending on the antenna placement and view of the sky, you might not be able to achieve certain survey-in precision.

It is suggested to start with 50 m precision and 600-second survey in time.

Note: The 50 m precision and the 600 second survey in time is just an example (or a starting point). Most likely you will want to use better a precision value and a longer survey-in time for better accuracy. This depends on your location, sky view, signal quality, and your antenna.

For survey-in, use the `ubxtool -t -w 5 -P 29.20 -v 1 -e SURVEYIN,<time_in_sec> , <precision_in_mm>` command:

```
# ubxtool -t -w 5 -P 29.20 -v 1 -e SURVEYIN,600,50000
ubxtool: enable SURVEYIN,600,50000

sent:
1667189015.3247
UBX-CFG-TMODE3:
  version 0 reserved1 0 flags x1
  ecefXOrLat 0 ecefYOrLon 0 ecefZOrAlt 0
  ecefXOrLatHP 0 ecefYOrLonHP 0 ecefZOrAltHP 0
  reserved2 0 fixedPosAcc 0 svinMinDur 600 svinAccLimit 50000
  reserved3 0 0
```

4.15.6 Check Survey-In Status

You can create a while-loop to track survey-in status before going to the next step. The following **ubxtool** command can be a major portion to get survey-in status/result (the last line, `valid 1 active 0`, indicates that survey-in is successful:

```
#ubxtool -t -w 5 -P 29.20 -v 1 -p TIM-SVIN
1669003112.3786
UBX-TIM-SVIN:
  dur 1378 meanX -303002288 meanY 492404391 meanZ 268440058 meanV 2024749696
  obs 1379 valid 1 active 0
```

After a successful procedure, this command will display XYZ coordinates, required later to save the current position to FLASH layer. Make sure to write down meanX, meanY, and meanZ coordinates in a text file.

4.15.7 Saving Survey-In Position to FLASH

Once the survey-in process is finished, and precise position is known to the receiver, ensure that the position is saved in the FLASH layer. This way, the found position of the receiver inside memory is kept between power-cycles of the adapter. First, check the position found by the survey-in procedure, then update the XYZ coordinates accordingly.

1. Update X coordinate in FLASH and RAM, using value from the first command:

   ```
   # ubxtool -v 1 -P 29.20 -z CFG-TMODE-ECEF_X,<value>,5 -w 1
   ```

2. Update Y coordinate in FLASH and RAM, using value from the first command:

   ```
   # ubxtool -v 1 -P 29.20 -z CFG-TMODE-ECEF_Y,<value>,5 -w 1
   ```

3. Update Z coordinate in FLASH and RAM, using value from the first command:

   ```
   # ubxtool -v 1 -P 29.20 -z CFG-TMODE-ECEF_Z,<value>,5 -w 1
   ```

4. Set TMODE-MODE (0 - disabled, 1 - survey-in, 2 - fixed) to fixed in BBR and FLASH layers:

   ```
   # ubxtool -v 1 -P 29.20 -z CFG-TMODE-MODE,2,5 -v 1 -w 1
   ```

5. Check, if TMODE-MODE, and coordinates were saved correctly to the FLASH:

   ```
   # ubxtool -v 1 -P 29.20 -g CFG-TMODE-MODE,0 -v 1 -w 2
   # ubxtool -v 1 -P 29.20 -g CFG-TMODE-ECEF_X,2 -v 1 -w 2
   # ubxtool -v 1 -P 29.20 -g CFG-TMODE-ECEF_Y,2 -v 1 -w 2
   # ubxtool -v 1 -P 29.20 -g CFG-TMODE-ECEF_Z,2 -v 1 -w 2
   ```

4.15.8 Simulate Antenna Removal

Sometimes you may want to check the functionality and simulate what happens if the antenna is removed, with these commands you can check that without physically unplugging the antenna cable.

```
# ubxtool -P 29.20 -w 1 -v 3 -z CFG-NAVSPG-INFIL_NCNOTHRS,50,1
```

To restore it, use:

```
# ubxtool -P 29.20 -w 1 -v 3 -z CFG-NAVSPG-INFIL_NCNOTHRS,0,1
```

4.15.9 Check GNSS Overall Configuration Performance

You might want to check your antenna and GNSS receiver overall performance with the Time-to-First-Fix (TTFF) value.

After a power-down (warm starts, hot starts), the GNSS device triggers a next TTFF value based on antenna setting (or skyview, etc.). See the command line and TTFF value `ttff 329907` in the following example. The unit of ttff is nanosecond (smaller number is better).

```
# ubxtool -t -w 3 -p NAV-STATUS -P 29.20
1668999366.1330
UBX-NAV-STATUS:
  iTOW 96984000 gpsFix 3 flags 0xdd fixStat 0x0 flags2 0x8
  ttff 329907, msss 325236062
```

For more details how to configure the u-Blox ZED-F9T-00B module, refer to:

https://content.u-blox.com/sites/default/files/ZED-F9T_IntegrationManual_UBX-21040375.pdf?hash=undefined

Note: In some cases, when GNSS loses the antenna reference, the GNSS might output for couple of seconds the 1PPS signal and NMEA messages. To more rapidly disqualify the 1PPS and NMEA messages, increase the filtering of GNSS receiver with **ubxtool** command:

```
# ubxtool -P 29.20 -v 1 -w 1 -z CFG-NAVSPG-OUTFIL_TACC,10,5
```

Or open two terminals and run:

```
# cat /dev/gnssX //Terminal 1

#echo -ne
"\xb5\x62\x06\x8a\x0a\x00\x00\x05\x00\x00\xb4\x00\x11\x30\x0a\x00\x9e\x1b
" > /dev/gnssX //Terminal2
```

5.0 Configuration Setup

The E810-CQDA2T provides two coaxial input-output SMA connectors, two unidirectional U.FL connectors and an optional GNSS input connector, as shown in the following two illustrations.

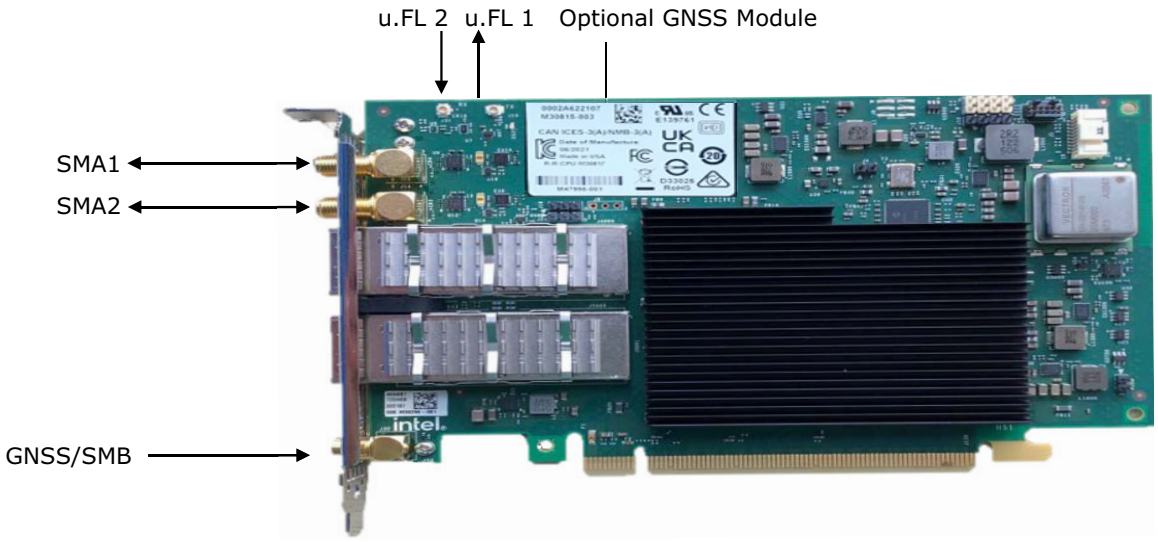


Figure 3. E810-CQDA2T Connector Locations

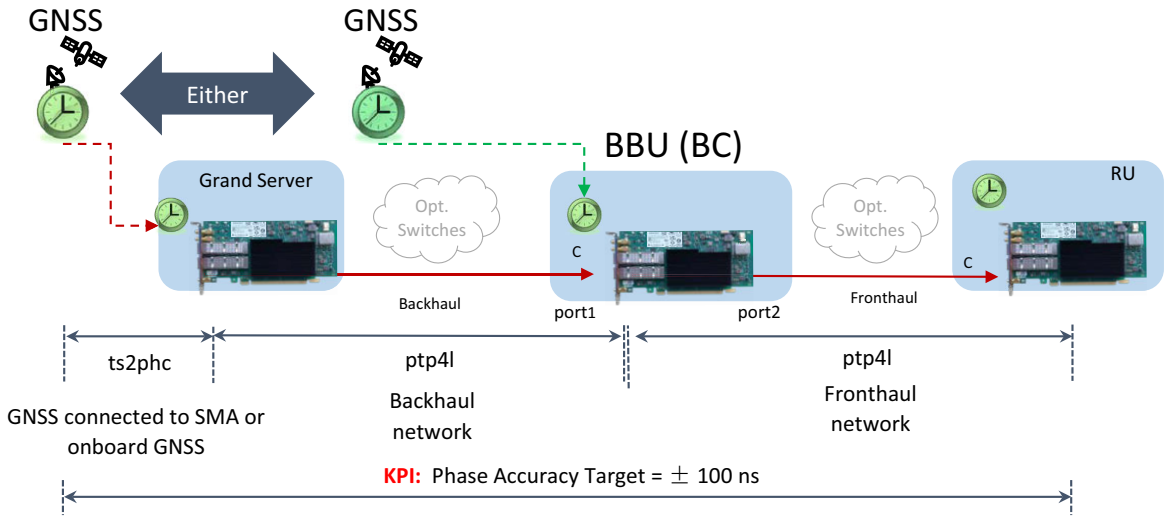


Figure 4. E810-CQDA2T Connections

5.1 Disable All SMA and U.FL Connections

1. Set interface device:

```
# export ETH=`grep 000f /sys/class/net/*/device/subsystem_device | awk -F"/"
'{print $5}' | head -n 1`
```

2. Disable U.FL2:

```
# echo 0 2 > /sys/class/net/$ETH/device/ptp/ptp*/pins/U.FL2
```

3. Disable U.FL1:

```
# echo 0 1 > /sys/class/net/$ETH/device/ptp/ptp*/pins/U.FL1
```

4. Disable SMA2:

```
# echo 0 2 > /sys/class/net/$ETH/device/ptp/ptp*/pins/SMA2
```

5. Disable SMA1:

```
# echo 0 1 > /sys/class/net/$ETH/device/ptp/ptp*/pins/SMA1
```

6. All disabled:

```
# echo 0 2 > /sys/class/net/$ETH/device/ptp/ptp*/pins/U.FL2
# echo 0 1 > /sys/class/net/$ETH/device/ptp/ptp*/pins/U.FL1
# echo 0 2 > /sys/class/net/$ETH/device/ptp/ptp*/pins/SMA2
# echo 0 1 > /sys/class/net/$ETH/device/ptp/ptp*/pins/SMA1
```

7. Check with dmesg:

```
# dmesg

[793017.697870] ice 0000:84:00.0: SMA1 + U.FL1 disabled
[793017.697871] ice 0000:84:00.0: SMA2 + U.FL2 disabled
```

5.2 PTP Grand Leader (GM) with Optional GNSS Module

This configuration is the highest-priority configuration and overrides any other if the GNSS is installed and operational.

5.2.1 External Connections

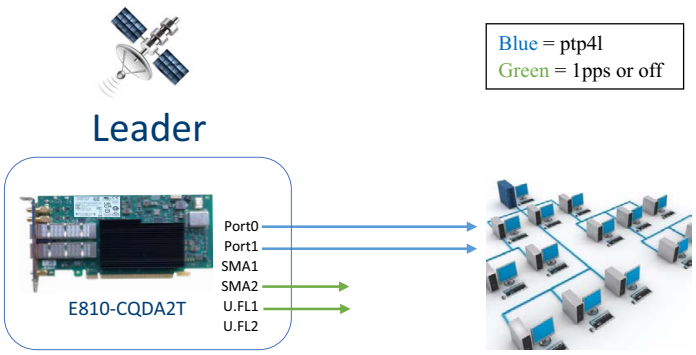


Figure 5. External Connections: PTP Grand Leader with Optional GNSS Module

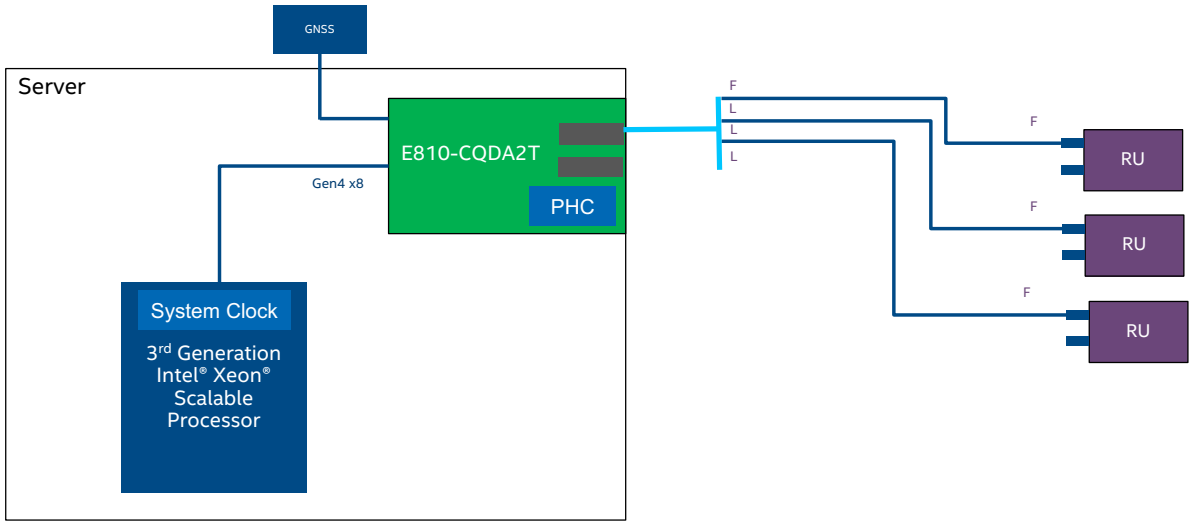


Figure 6. External Connections: Single E810-CQDA2T Adapter Configuration with GNSS (and Breakout Cable)

5.2.2 Software Configuration

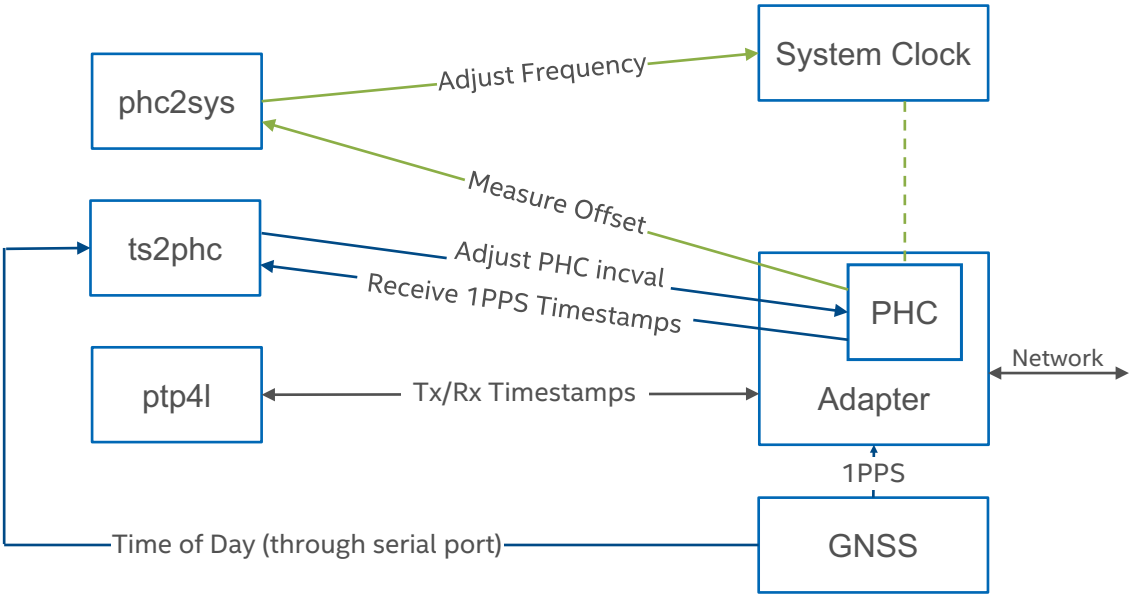


Figure 7. Linux Software Stack Overview: Single E810-CQDA2T Adapter with GNSS Software Stack

Before proceeding, it is recommended to set all SDP pins and U.FL to off (see Section 4.0).

1. Set interface device:

```
# export ETH=`grep 000f /sys/class/net/*/device/subsystem_device | awk -F"/"
'{print$5}' | head -n 1`
# export PCI_SLOT=`grep PCI_SLOT_NAME /sys/class/net/$ETH/device/uevent | cut -c
15-`
```

2. Make sure GNSS has an active connection: */dev/gnssX* where *X* stands for the GNSS device number.

```
# cat /dev/gnssX
$GNRMC,182805.00,A,0.0,N,0.0,E,0.082,,050321,,,D,V*14
$GNVTG,,T,,M,0.082,N,0.152,K,D*34
$GNGGA,182805.00,0.0,N,0.0,E,2,12,0.57,11.1,M,32.7,M,,0000*7D
$GNGSA,A,3,18,20,26,23,16,29,07,15,27,10,,,1.07,0.57,0.90,1*09
$GNGSA,A,3,81,79,72,82,80,78,65,88,87,,,,1.07,0.57,0.90,2*0C
$GNGSA,A,3,02,30,04,36,11,,,,,,,,1.07,0.57,0.90,3*0E
```

3. Run **ts2phc**:

```
# ethtool --set-priv-flags $ETH extts_filter on

# ./ts2phc -f configs/ts2phc-generic.cfg -s nmea -m
```

Note: You might want filter the 1PPS timestamps till the DPLL locked. See Section 4.13

Note: See Section 5.8, "Example ts2phc Configuration File".

4. Run **phc2sys**:

   ```
   # ./phc2sys -s $ETH -O -37 -m
   ```

   The `-O -37` can be used to accommodate for leap seconds

   Note: To update **linuxptp** version, use **git clone**:

   ```
   git clone git://git.code.sf.net/p/linuxptp/code
   ```

   To get an appropriate leapfile for RHEL-based Linux distributions:

   https://github.com/eggert/tz/blob/main/leap-seconds.list

   DO NOT use */usr/share/zoneinfo/leapseconds* defined in the *ts2phc.8* file.

5. Run **ptp4l**:

   Running on Port 0 (1 as well if required):

   ```
   # ./ptp4l -i $ETH -m
   ```

6. Run **synce4l** (optional):

   ```
   #./synce4l -f configs/synce4l.cfg  -m
   ```

   Note: Refer to Section 5.10, "Example synce4l Configuration File for BC".

   Note: For GM mode, ensure that `input_mode = external` is used in the config file.

5.3 PTP Grand Leader (GM) with External GNSS Clock

5.3.1 External Connections

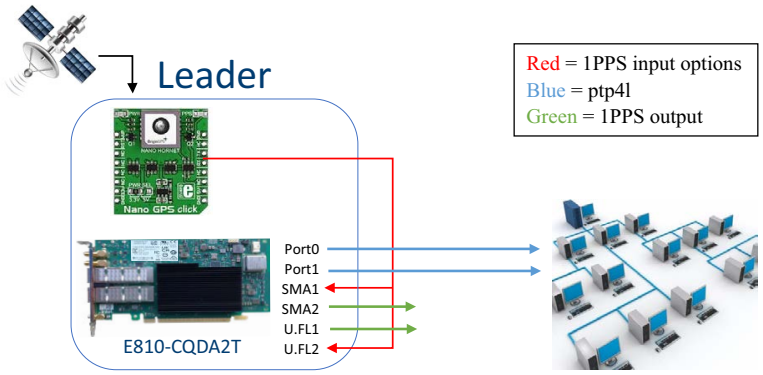


Figure 8. External Connections: PTP Grand Leader with External GNSS Clock

5.3.2 Software Configuration

Before proceeding, configure the GNSS to output a 1PPS signal and connect it to SMA1 (or to U.FL2). It is recommended to set all SDP pins and U.FL to off (see Section 4.0).

1. Set interface device:

```
# export ETH=`grep 000f /sys/class/net/*/device/subsystem_device | awk -F"/"
'{print $5}' | head -n 1` #(port0)
# export ETH1=`grep 000f /sys/class/net/*/device/subsystem_device | awk -F"/"
'{print $5}' | head -n 2 | tail -n +2` #(port1)
# export ETH2=`grep 000f /sys/class/net/*/device/subsystem_device | awk -F"/"
'{print $5}' | head -n 3 | tail -n +3` #(port2)
# export ETH3=`grep 000f /sys/class/net/*/device/subsystem_device | awk -F"/"
'{print $5}' | head -n 4 | tail -n +4` #(port3)
```

2. SMA1 as input and SMA2 as output:

```
# echo 1 1 > /sys/class/net/$ETH/device/ptp/ptp*/pins/SMA1
# echo 2 2 > /sys/class/net/$ETH/device/ptp/ptp*/pins/SMA2
```

   Or, U.FL2 as input and U.FL1 as output:

```
# echo 1 2 > /sys/class/net/$ETH/device/ptp/ptp*/pins/U.FL2
# echo 2 1 > /sys/class/net/$ETH/device/ptp/ptp*/pins/U.FL1
```

   Note: If you have SMA1 connected to an external 1PPS source, you cannot use the U.FL configuration.

3. Verify that the outputs were set correctly:

```
#dmesg
[793494.341435] ice 0000:84:00.0: SMA1 RX
[793494.341437] ice 0000:84:00.0: SMA2 TX
[793506.667670] ice 0000:84:00.0: <DPLL0> state changed to: unlocked, pin SMA1
[793506.668178] ice 0000:84:00.0: <DPLL1> state changed to: unlocked, pin SMA1
[793518.699755] ice 0000:84:00.0: <DPLL0> state changed to: locked, pin SMA1
[793518.700264] ice 0000:84:00.0: <DPLL1> state changed to: locked, pin SMA1
```

Or:

```
#dmesg
[101056.462309] ice 0000:18:00.0: SMA1 RX + U.FL1 TX
[101056.462317] ice 0000:18:00.0: UFL2 RX
[101090.668497] driver cannot use function 1 on pin 0
```

4. Run **ts2phc**:

   Running on Port 0:

   ```
   # ./ts2phc -f configs/ts2phc-generic.cfg -s generic -m
   ```

   Note: See Section 5.8, "Example ts2phc Configuration File".

   Note: You might want filter the 1PPS timestamps till the DPLL locked. See Section 4.13.

5. Run **ptp4l**:

   Running on Port 0 (1 as well if required):

   ```
   # ./ptp4l -i $ETH -m
   ```

6. Run **synce4l** (optional):

   ```
   #./synce4l -f configs/synce4l.cfg  -m
   ```

   Note: Refer to Section 5.10, "Example synce4l Configuration File for BC".

   Note: For GM mode, ensure that `input_mode = external` is used in the config file.

5.4 Boundary Clock Configuration

5.4.1 External Connections

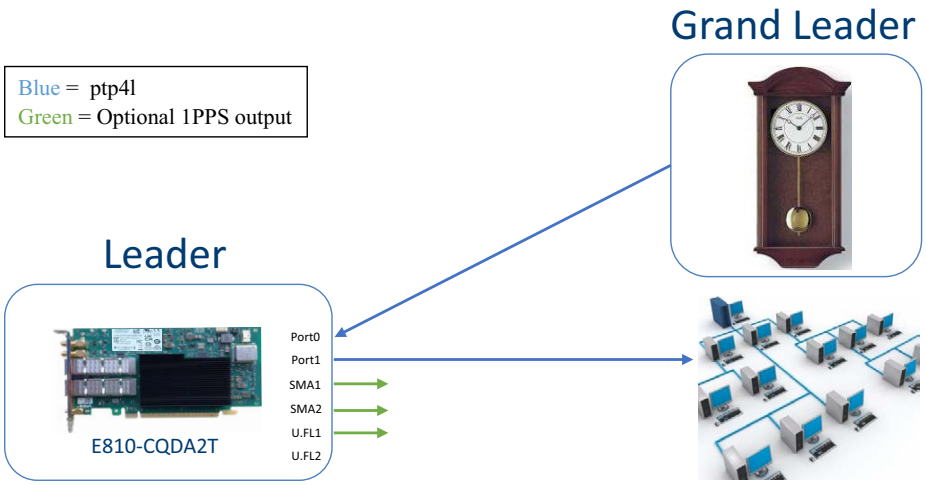


Figure 9. External Connections: Boundary Clock Configuration

5.4.2 Boundary Clock Notes

- For visibility, users might want to enable 1PPS on the SMA1.
- If users want to synchronize the DPLL to the E810 PHC, they must enable the appropriate SDP pin, as there are two DPLLs: DPLL0 drives the external clock source of the E810 controller, while DPLL1 drives the SMA outputs.
- If users want to synchronize DPLL1 to **ptp4l**, then they must use SDP20.
- SMA1 Tx and U.FL1 Tx is not a supported configuration on the E810-CQDA2T.

5.4.3 Software Configuration

Before proceeding, it is recommended to set all SDP pins and U.FL to off (see Section 4.0).

1. Set interface device (only top command is essential):

```
# export ETH=`grep 000e /sys/class/net/*/device/subsystem_device | awk -F"/"
'{print $5}' | head -n 1’ (port0)
# export ETH1=`grep 000e /sys/class/net/*/device/subsystem_device | awk -F"/"
'{print $5}' | head -n 2 | tail -n +2’ (port1)
```

2. Set periodic output on SDP20 to 10 MHz (to synchronize the DPLL1 to the E810 PHC synced by **ptp4l**):

```
# echo 1 0 0 0 100 > /sys/class/net/$ETH/device/ptp/ptp*/period
```

   and set SDP22 (to synchronize the DPLL1 to the E810 PHC synced by **ptp4l**):

```
# echo 2 0 0 1 0 > /sys/class/net/$ETH/device/ptp/ptp*/period
```

   Note: DPLL only syncs to SDP20/SDP22 if it is the higher priority. Setting SDP20 is the preferred method for synchronizing 1PPS outputs.

3. Enable SMA1 output (for visibility):

   ```
   # echo 2 1 > /sys/class/net/$ETH/device/ptp/ptp*/pins/SMA1
   ```

   Or:

   - Enable U.FL1 output: (for visibility):

     ```
     # echo 2 1 > /sys/class/net/$ETH/device/ptp/ptp*/pins/U.FL1
     ```

   - Enable SMA2 output: (for visibility):

     ```
     # echo 2 2 > /sys/class/net/$ETH/device/ptp/ptp*/pins/SMA2
     ```

   Note: If U.FL1 if set to Tx, then SMA1 is also set to Rx and cannot be changed. Make sure no 1PPS input is connected to SMA1 if using U.FL1 as Tx.

4. Verify if the outputs were set correctly:

   ```
   #dmesg
   [27526.767803] ice 0000:84:00.0: SMA1 TX
   [27526.767804] ice 0000:84:00.0: SMA2 TX
   [27526.852127] ice 0000:18:00.0: <DPLL1> state changed to: unlocked, pin CVL-SDP20
   [30243.385109] ice 0000:18:00.0: <DPLL1> state changed to: locked, pin CVL-SDP20
   ```

   Or:

   ```
   #dmesg
   [27611.114869] ice 0000:84:00.0: SMA1 RX + U.FL1 TX
   [27611.114870] ice 0000:84:00.0: SMA2 TX
   [27611.824727] ice 0000:18:00.0: <DPLL1> state changed to: unlocked, pin CVL-SDP20
   [30243.385109] ice 0000:18:00.0: <DPLL1> state changed to: locked, pin CVL-SDP20
   ```

5. Run **ptp4l**:

   ```
   # ./ptp4l -i $ETH -i $ETH1  -m - -f ./ptp4l_bc.conf
   ```

   Note: For BC, it is recommended to use one **ptp4l** instance.

6. Run **synce4l** (optional):

   ```
   #./synce4l -f configs/synce4l.cfg  -m
   ```

   Note: Refer to Section 5.10, "Example synce4l Configuration File for BC".

5.5 Port Configured as Follower

The E810-CQDA2T has one PHC that is shared across all the ports. As a result, only one PTP follower can be run as shown.

5.5.1 External Connections

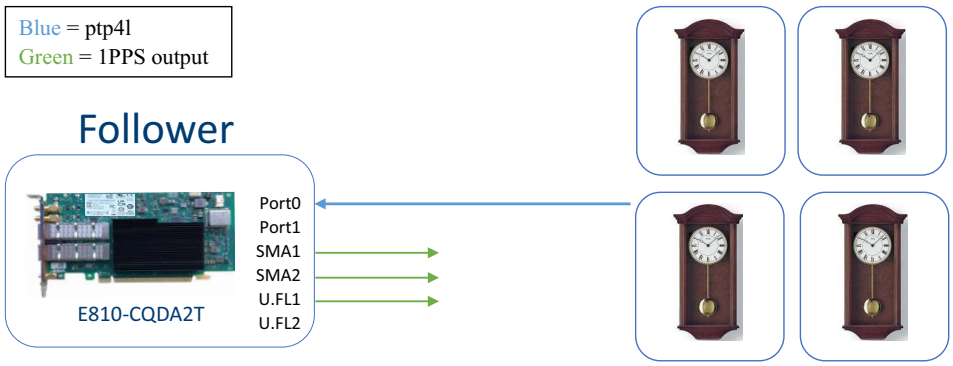


Figure 10. External Connections: Port Configured as Follower

5.5.2 Software Configuration

Before proceeding, it is recommended to set all SDP pins and U.FL to off (see Section 4.0).

1. Set interface device:

   ```
   # export ETH=`grep 000f /sys/class/net/*/device/subsystem_device | awk -F"/"
   '{print $5}' | head -n 1`
   ```

2. Set the periodic output on SDP20 to 10 MHz (to synchronize the DPLL1 to the E810 PHC synced by **ptp4l**):

   ```
   # echo 1 0 0 0 100 > /sys/class/net/$ETH/device/ptp/ptp*/period
   ```

   and set SDP22 (to synchronize the DPLL1 to the E810 PHC synced by **ptp4l**):

   ```
   # echo 2 0 0 1 0 > /sys/class/net/$ETH/device/ptp/ptp*/period
   ```

   Note: DPLL only syncs to SDP20/SDP22 if it is the higher priority. Setting SDP20 is the preferred method for synchronizing 1PPS outputs.

3. Enable SMA1 output: (for visibility):

   ```
   # echo 2 1 > /sys/class/net/$ETH/device/ptp/ptp*/pins/SMA1
   ```

   Or:

   - Enable U.FL1 output (for visibility):

     ```
     # echo 2 1 > /sys/class/net/$ETH/device/ptp/ptp*/pins/U.FL1
     ```

   - Enable SMA2 output (for visibility):

     ```
     # echo 2 2 > /sys/class/net/$ETH/device/ptp/ptp*/pins/SMA2
     ```

   Note: If U.FL1 if set to Tx then SMA1 is also set to Rx and cannot be changed. Make sure no 1PPS input is connected to SMA1 if using U.FL1 as Tx.

4. Verify if the outputs were set correctly:

```
#dmesg
[ 827.397307] ice 0000:18:00.0: SMA1 TX
[ 827.397315] ice 0000:18:00.0: SMA2 + U.FL2 disabled
[ 837.852127] ice 0000:18:00.0: <DPLL1> state changed to: unlocked, pin CVL-SDP20
[1243.385109] ice 0000:18:00.0: <DPLL1> state changed to: locked, pin CVL-SDP20
```

   Or:

```
#dmesg
[ 7827.397307] ice 0000:18:00.0: SMA1 RX + U.FL1 TX
[ 7827.397315] ice 0000:18:00.0: SMA2 + U.FL2 disabled
[ 7837.852127] ice 0000:18:00.0: <DPLL1> state changed to: unlocked, pin CVL-SDP22
[10243.385109] ice 0000:18:00.0: <DPLL1> state changed to: locked, pin CVL-SDP22
```

```
#dmesg
[ 827.397307] ice 0000:18:00.0: SMA1 + U.FL1 disabled
[ 827.397315] ice 0000:18:00.0: SMA2 TX
[ 837.852127] ice 0000:18:00.0: <DPLL1> state changed to: unlocked, pin CVL-SDP20
[1243.385109] ice 0000:18:00.0: <DPLL1> state changed to: locked, pin CVL-SDP20
```

   Note: The first two lines of each *dmesg* changes depending on what outputs are enabled for visibility.

5. Run **ptp4l**:

```
./ptp4l -i $ETH -m -s
```

5.6 SyncE Setup

This configuration shows how to specifically setup SyncE. Note that users can use this SyncE configuration with tools such as **ptp4l** together for clock recovery. SyncE ITU G.811 and G8262 can assist with better frequency synchronization.

There are two main configuration options for SyncE:

- Only physical clock recovery (Section 5.6.2).
- With ITU G.8264 ESMC messaging using synce4l (Section 5.6.3).

5.6.1 External Connections

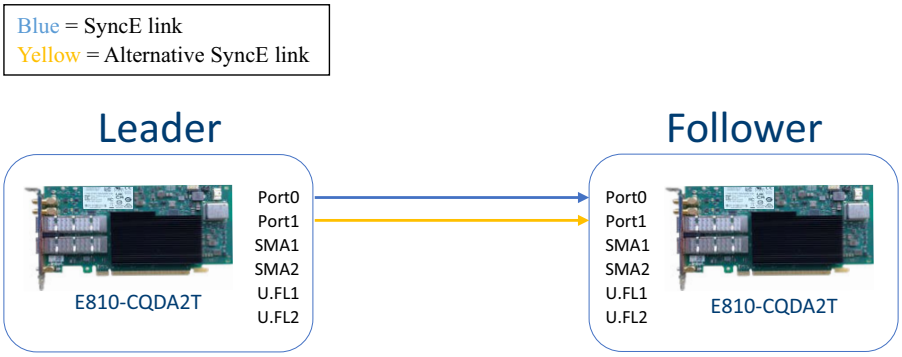


Figure 11. External Connections: SyncE Setup

5.6.2 Physical Clock Recovery

Before proceeding, it is recommended to set all SMA and U.FL connectors to off (see Section 4.0).

1. Set relevant interface device (only top command is essential):

```
# export ETH=`grep 000f /sys/class/net/*/device/subsystem_device | awk -F"/"
'{print $5}' | head -n 1'                (port0)
# export ETH1=`grep 000f /sys/class/net/*/device/subsystem_device | awk -F"/"
'{print $5}' | head -n 2 | tail -n +2' (port1)
```

2. To enable recovered clock from port0 on the highest priority clock run:

```
# echo 1 0 > /sys/class/net/$ETH/device/phy/synce
```

3. To enable recovered clock from port1 on the lowest priority clock run:

```
# echo 1 1 > /sys/class/net/$ETH1/device/phy/synce
```

Note: First integer represents enable (1) or disable (0), and the second represents the highest clock priority (0) and lowest clock priority (1).

4. Verify if the outputs were set correctly

```
#dmesg
[19707.757036] ice 0000:18:00.1: Enabled recovered clock: pin C827_0-RCLKB
[19718.804404] ice 0000:18:00.0: <DPLL0> state changed to: unlocked, pin C827_0-
RCLKB
[19718.804920] ice 0000:18:00.0: <DPLL1> state changed to: unlocked, pin C827_0-
RCLKB
[19719.511845] ice 0000:18:00.0: Enabled recovered clock: pin C827_0-RCLKA
```

```
[19789.633771] ice 0000:18:00.0: <DPLL0> state changed to: locked, pin C827_0-
RCLKA
```

Note: Only enable SyncE on one adapter, either leader or follower is acceptable. If both are set, users might run into sync loop issues.

5.6.3 ITU G.8264 ESMC Messaging Using synce4l

The synce4l application is implementing the G.8264 ESMC protocol.

1. Run **synce4l**:

```
# ./synce4l -f configs/synce4l.cfg -l 7 -m
```

Note: Refer to Section 5.10, "Example synce4l Configuration File for BC".

Note: For GM mode, ensure that `input_mode = external` is used in the config file.

5.6.4 Two E810-CQDA2T Adapter Configuration without GNSS

Figure 12 shows two E810-CQDA2T adapters in a system with a cell site router.

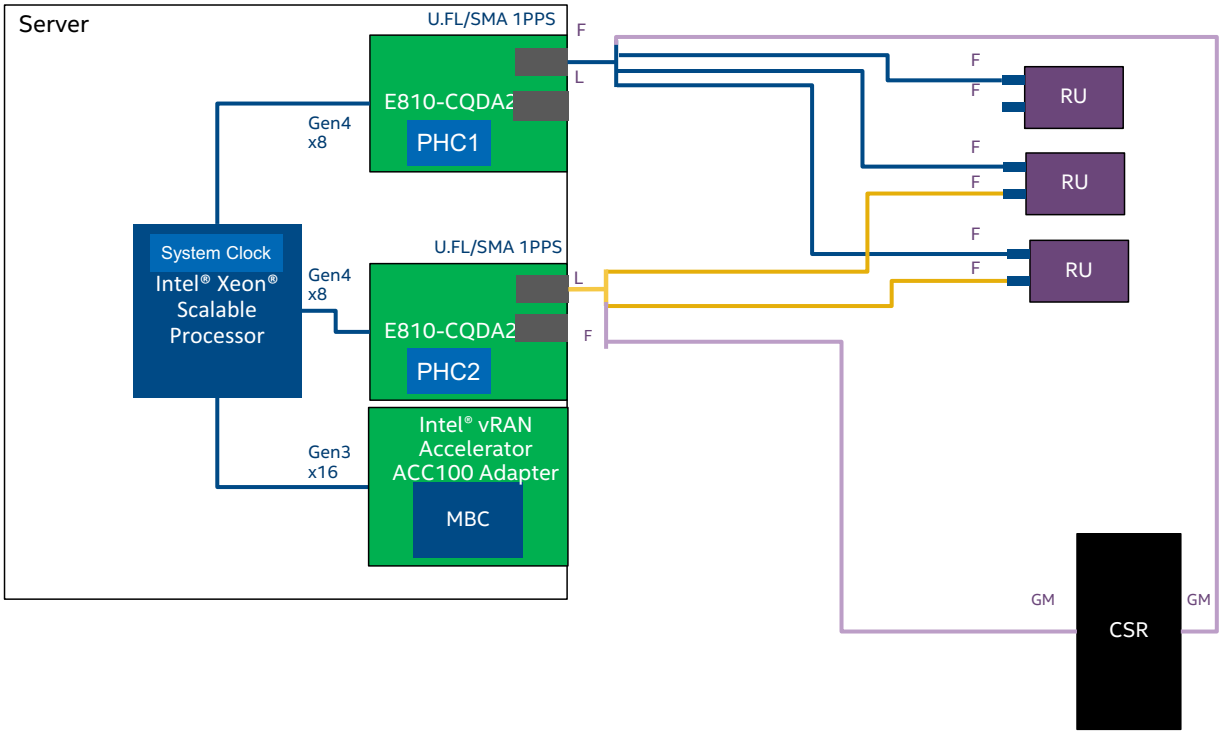


Figure 12. External Connections: Two E810-CQDA2T Adapters without GNSS

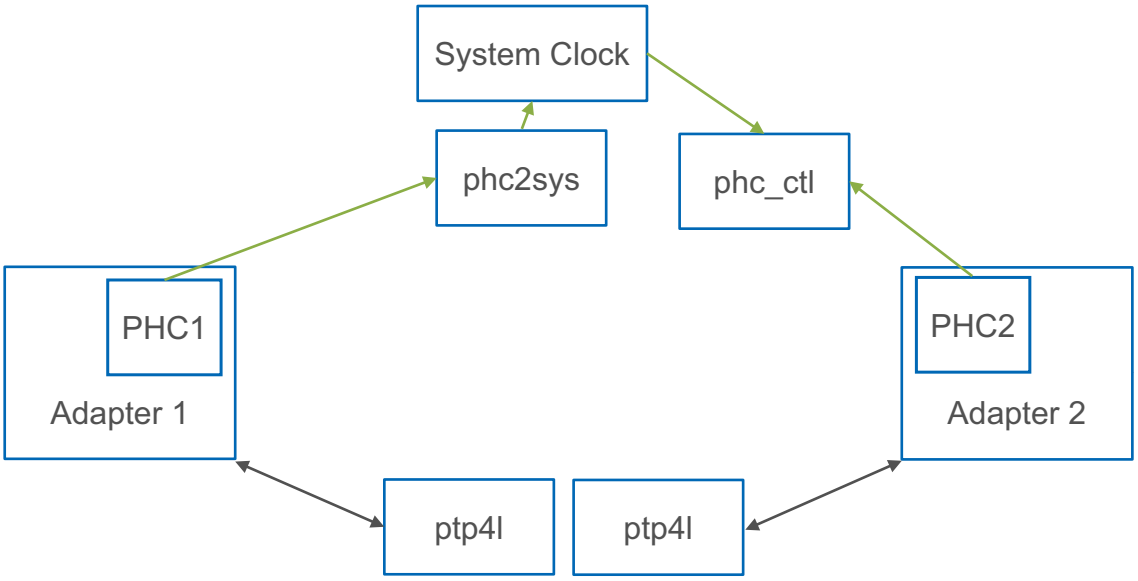


Figure 13. External Connections: Two E810-CQDA2T Adapters without GNSS Linux Software Stack

Adapter 1 Linux software stack:

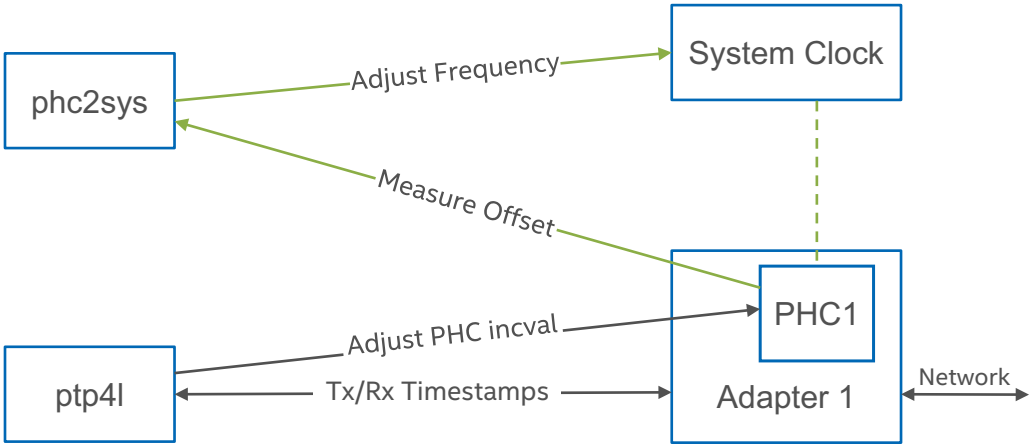


Figure 14. External Connections: Two E810-CQDA2T Adapters without GNSS Adapter 1 Software Stack

Adapter 1 configuration:

1. Run one instance of **ptp4l** per E810-CQDA2T:

   ```
   # ptp4l -m -f config.cfg -i ens1f0 -i ens1f1 -i ens1f2 -i ens1f3
   ```

2. Run **phc2sys** to synchronize system time to the PHC time:

   ```
   # phc2sys -s ens1f0 -c CLOCK_REALTIME -w -m
   ```

Adapter 2 Linux software stack:

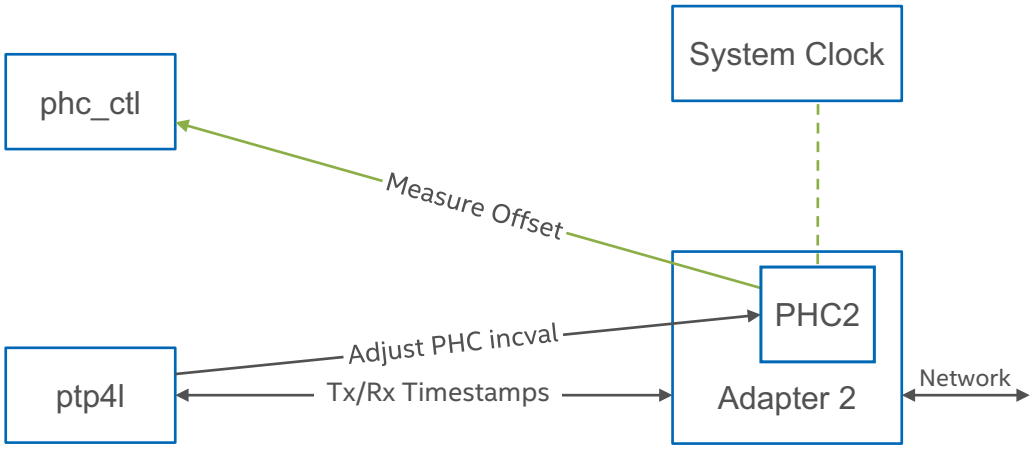


Figure 15. External Connections: Two E810-CQDA2T Adapters without GNSS Adapter 2 Software Stack

Adapter 2 configuration:

1. Run one instance of **ptp4l** per E810-CQDA2T:

   ```
   # ptp4l -m -f config.cfg -i ens2f0 -i ens2f1 -i ens2f2 -i ens2f3
   ```

2. Monitor PHC time on adapter two vs. system time:

   ```
   # phc_ctl ens2f0 cmp
   ```

5.6.5 Two E810-CQDA2T Adapter Configuration without GNSS and with 1PPS

Figure 16 shows two E810-CQDA2T adapters in a system with a 1PPS connection between them.

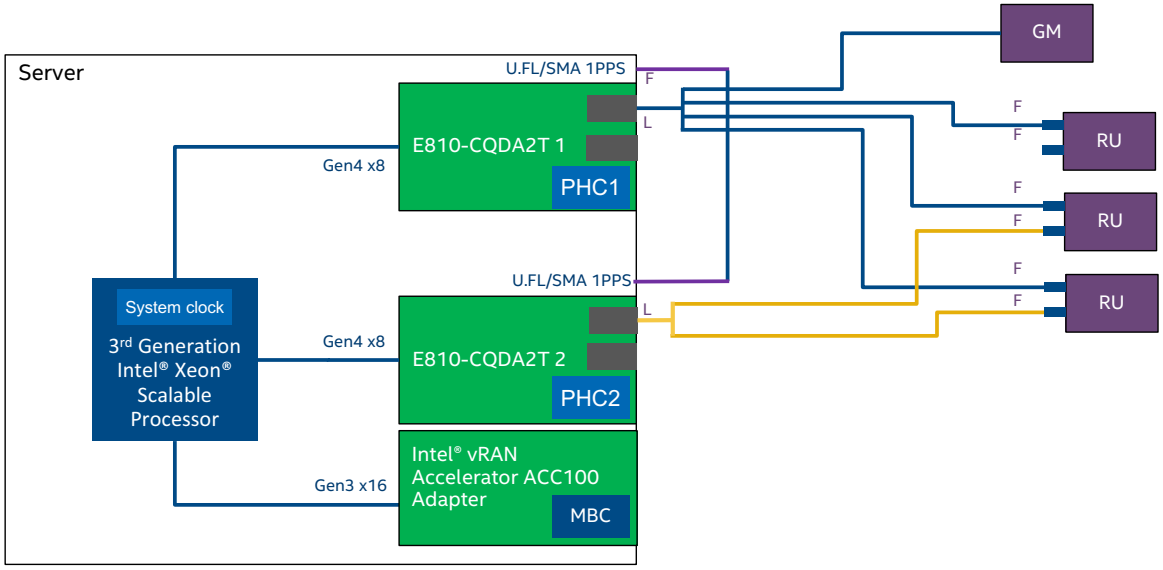


Figure 16. External Connections: Two E810-CQDA2T Adapter Configuration (no GNSS)

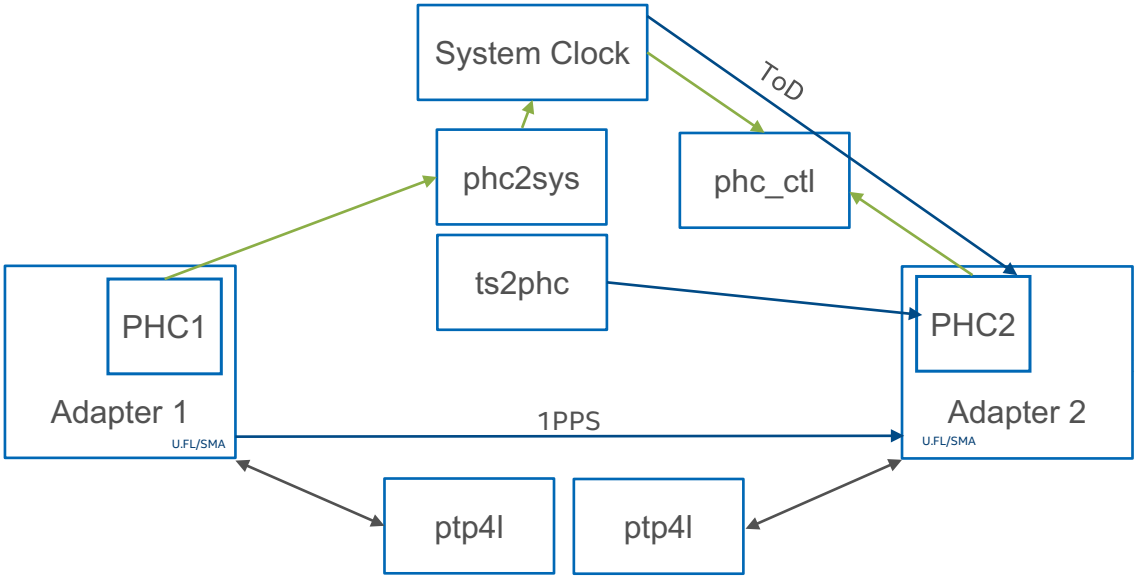


Figure 17. External Connections: Two E810-CQDA2T Adapter Configuration Linux Software Stack (no GNSS)

Adapter 1 Linux software stack:

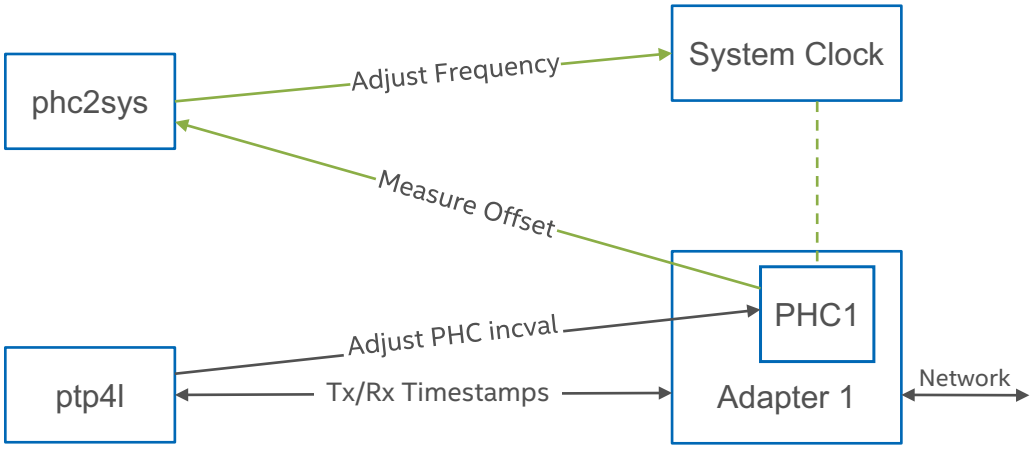


Figure 18. Linux Software Stack Overview: Adapter 1 (no GNSS)

Adapter 1 configuration:

1. Enable 1PPS output on U.FL1:

   ```
   # echo 2 1 > /sys/class/net/ens1f0/device/ptp/ptp*/pins/U.FL1
   ```

   or SMA 1:

   ```
   # echo 2 1 > /sys/class/net/$ETH/device/ptp/ptp*/pins/SMA1
   ```

2. Enable SDP20/SDP22

   Set periodic output on SDP20 to 10 MHz (to synchronize the DPLL1 to the E810 PHC synced by **ptp4l**):

   ```
   # echo 1 0 0 0 100 > /sys/class/net/$ETH/device/ptp/ptp*/period
   ```

   and set SDP22 (to synchronize the DPLL1 to E810 PHC synced by **ptp4l**):

   ```
   # echo 2 0 0 1 0 > /sys/class/net/$ETH/device/ptp/ptp*/period
   ```

3. Run **ptp4l**:

   ```
   # ptp4l -m -f config.cfg -i ens1f0 -i ens1f1 -i ens1f2 -i ens1f3
   ```

4. Run **phc2sys** to synchronize system time to the PHC time:

   ```
   # phc2sys -c ens1f0 -s CLOCK_REALTIME -w -m
   ```

Adapter 2 Linux software stack:

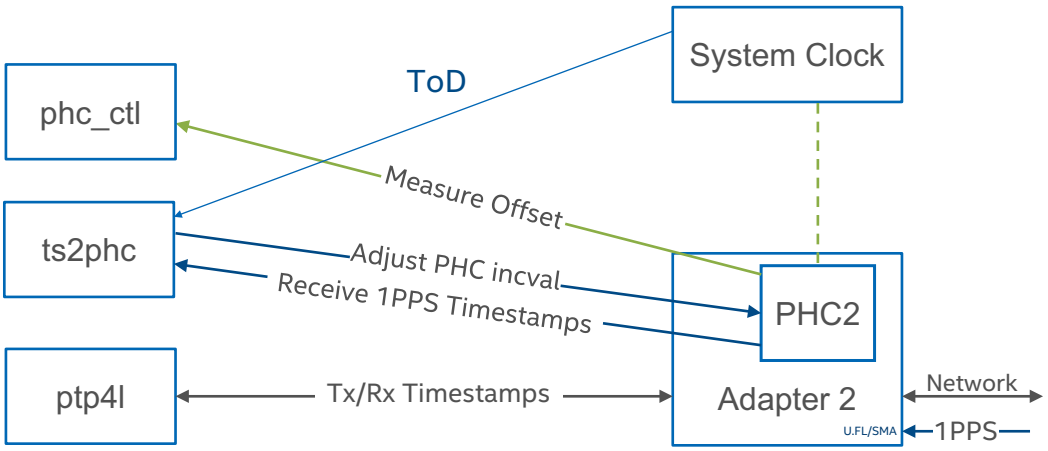


Figure 19. Linux Software Stack Overview: Adapter 2 (no GNSS)

Adapter 2 configuration:

1. Enable 1PPS input on U.FL2:

   ```
   # echo 1 2 > /sys/class/net/ens2f0/device/ptp/ptp*/pins/U.FL2
   ```

   or, enable 1PPS input on SMA2:

   ```
   # echo 1 2 > /sys/class/net/$ETH/device/ptp/ptp*/pins/SMA2
   ```

2. Run **ts2phc** to get time from 1PPS over U.FL2 and ToD from system:

   ```
   # ts2phc -f config.cfg -s generic -c ens2f0
   ```

3. Run **ptp4l**:

   ```
   # ptp4l -m -f config.cfg -i ens2f0 -i ens2f1  -i ens2f2 -i ens2f3
   ```

4. Monitor PHC time on adapter 2 comparing to system time:

   ```
   # phc_ctl ens2f0 cmp
   ```

Notes:

- Only enable SDP20/SDP22 1PPS when PTP or SyncE is actively synced.
- Monitor/switch **phc2sys**, if one adapter loses sync in multi-adapter system.
- Use a different UDS socket when running multiple **ptp4l** instances.

5.6.6 Two E810-CQDA2T Adapters with GNSS Connection Setup

Figure 20 shows two E810-CQDA2T adapters in a system with a GNNS connection.

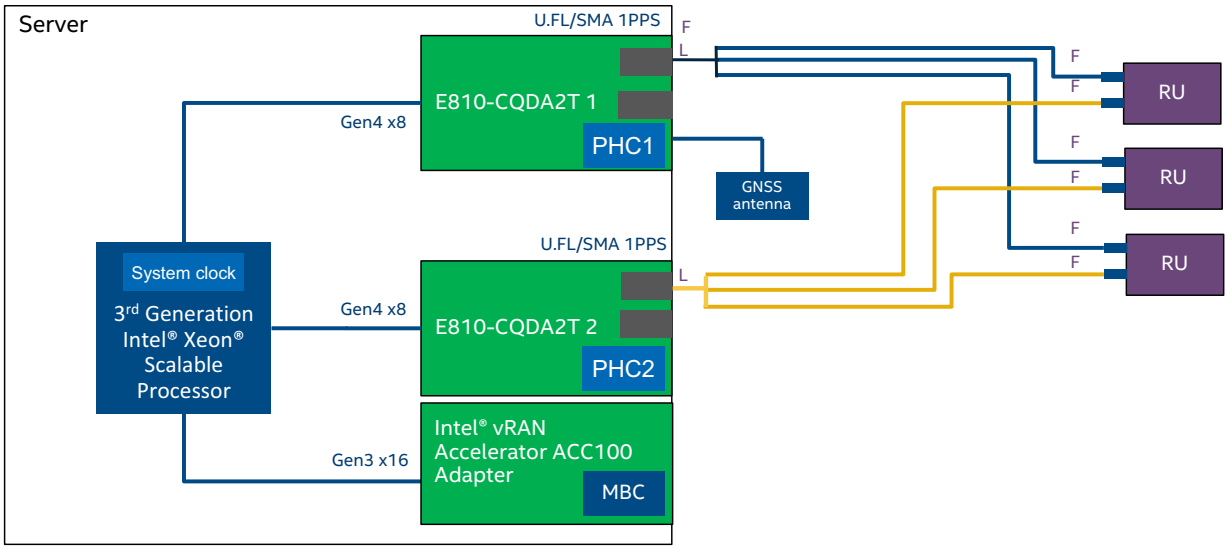


Figure 20. External Connections: Two E810-CQDA2T Adapter Configuration (with GNSS)

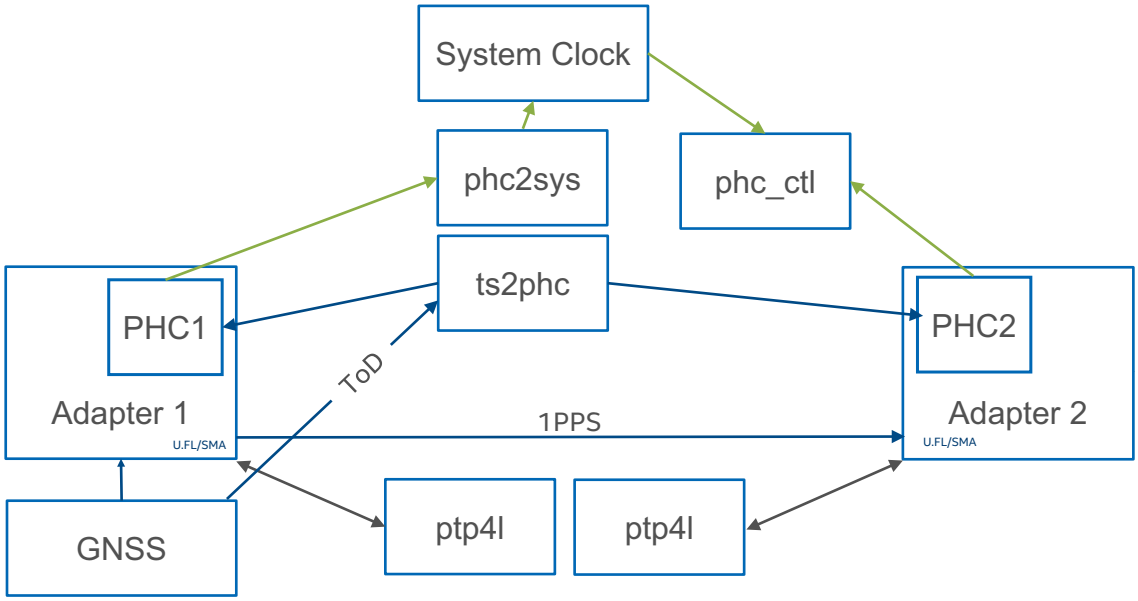


Figure 21. External Connections: Two E810-XXVDA4T Adapter Configuration Linux Software Stack (with GNSS)

Adapter 1 Linux software stack:

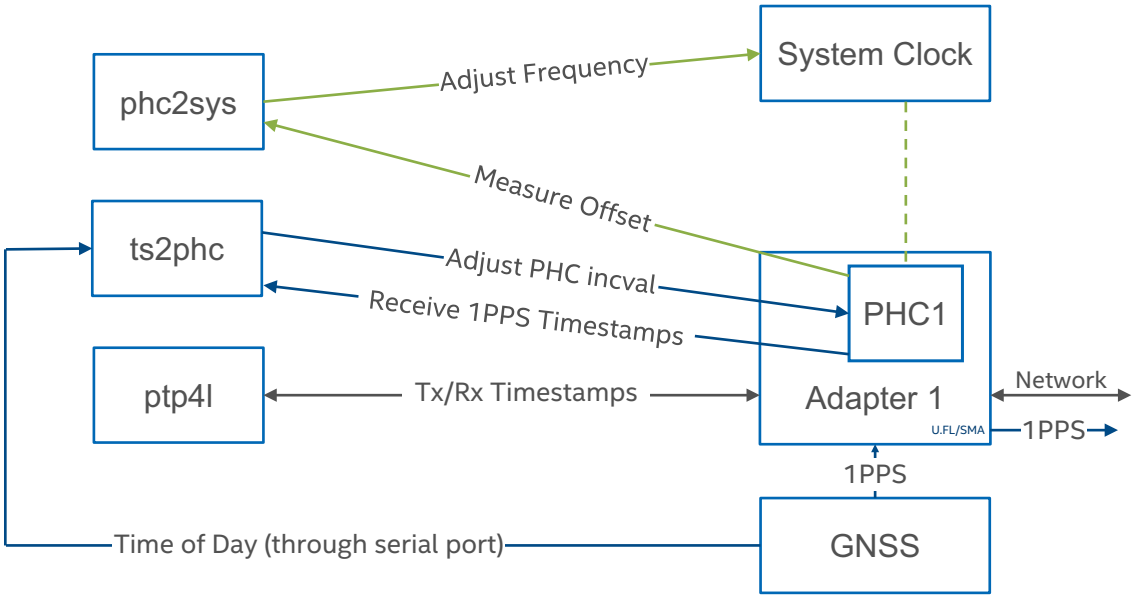


Figure 22. Linux Software Stack Overview: Adapter 1 (with GNSS)

Adapter 1 configuration:

1. Run **ts2phc** to get time from the GNSS:

   ```
   # ts2phc -f config.cfg -s nmea -c ens1f0
   ```

   You can also use NMEA to get time to both adapters:

   ```
   # ts2phc -f config.cfg -s nmea -c ens1f0 -c ens2f0
   ```

2. Enable 1PPS out on U.FL1:

   ```
   # echo 2 1 > /sys/class/net/ens1f0/device/ptp/ptp*/pins/U.FL1
   ```

3. Run **ptp4l**:

   ```
   # ptp4l -m -f config.cfg -i ens1f0 -i ens1f1  -i ens1f2 -i ens1f3
   ```

4. Run **phc2sys** to synchronize system time to the PHC time:

   ```
   # phc2sys -s ens1f0 -c CLOCK_REALTIME -w -m
   ```

Adapter 2 Linux software stack:

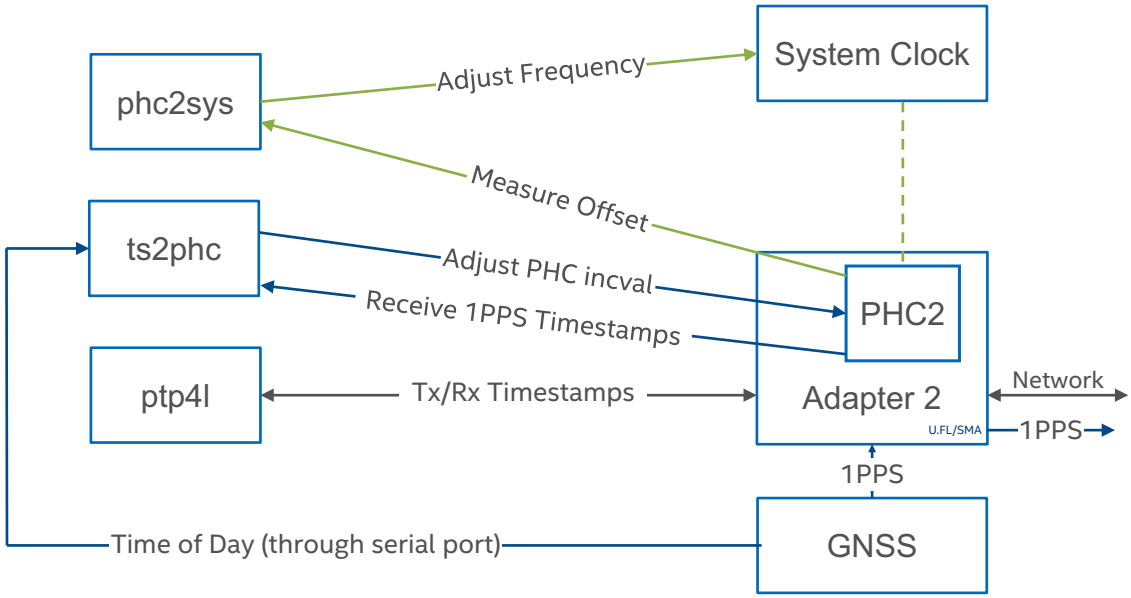


Figure 23. Linux Software Stack Overview: Adapter 2 (with GNSS)

Adapter 2 configuration:

1. Enable 1PPS in on U.FL2:

   ```
   # echo 1 2 > /sys/class/net/ens2f0/device/ptp/ptp*/pins/U.FL2
   ```

2. Run **ts2phc** to get time from 1PPS over U.FL2 and ToD from system:

   ```
   # ts2phc -f config.cfg -s generic -c ens2f0
   ```

3. Run **ptp4l**:

   ```
   # ptp4l -m -f config.cfg -i ens2f0 -i ens2f1  -i ens2f2 -i ens2f3
   ```

4. Monitor PHC time on adapter 2 comparing to system time:

   ```
   # phc_ctl ens2f0 cmp
   ```

5.7 O-RAN Configuration 1

5.7.1 External Connections

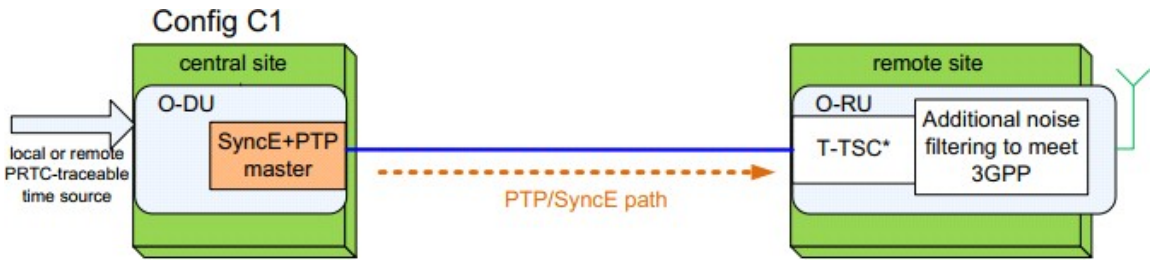


Figure 24. External Connections: O-RAN Configuration 1

Note: O-RAN Fronthaul Working Group Control, User and Synchronization Plane Specification, 2020.

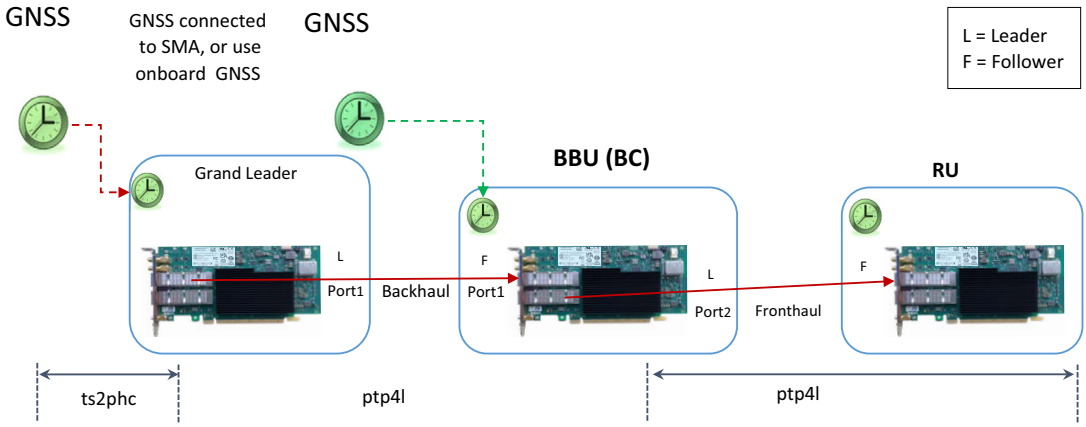


Figure 25. O-RAN Configuration 1 Connections

5.7.2 Software Configuration

- Grand leader adapter (see Section 5.2 and Section 5.3)
- BBU (BC) Adapter (see Section 5.4)
- RU Adapter (see Section 5.5)

5.8 Example ts2phc Configuration File

```
[global]
use_syslog          0
verbose             1
logging_level       7
ts2phc.pulsewidth   100000000
# For GNSS module
ts2phc.nmea_serialport /dev/gnss0 #/dev/gnssX, where X is GNSS device number.
leapfile    /home/<USER>/linuxptp-4/leapseconds.list
[enp1s0f0(dev/ptp4)]
ts2phc.extts_polarity  rising
```

Note: The **leapfile** option is available but not necessary for the program to run.

5.9 Example ptp4l Configuration File for BC

```
[global]
#
# Default Data Set
#
twoStepFlag               1
slaveOnly                 0
priority1                 129
priority2                 255
domainNumber              24
utc_offset                37
clockClass                255
clockAccuracy             0xFE
offsetScaledLogVariance   0xFFFF
free_running              0
freq_est_interval         1
dataset_comparison G.8275.x
G.8275.portDS.localPriority 128

#
# Port Data Set
#
logAnnounceInterval       -3
logSyncInterval           -4
logMinDelayReqInterval    -4
logMinPdelayReqInterval   0
announceReceiptTimeout    3
syncReceiptTimeout        3
delayAsymmetry            0
fault_reset_interval      4
neighborPropDelayThresh   800
min_neighbor_prop_delay   -20000000
#
# Run time options
#
assume_two_step           1
logging_level             6
path_trace_enabled        1
follow_up_info            0
```

```
hybrid_e2e                 0
net_sync_monitor           0
tx_timestamp_timeout       10
use_syslog                 1
verbose                    0
summary_interval           -4
kernel_leap                1
check_fup_sync             0
#
# Servo options
#
pi_proportional_const      0.60
pi_integral_const          0.001
pi_proportional_scale      0.0
pi_proportional_exponent -0.3
pi_proportional_norm_max   0.7
pi_integral_scale          0.0
pi_integral_exponent       0.4
pi_integral_norm_max       0.3
step_threshold             0.00002
first_step_threshold       0.00002
#first_step_threshold      0.0
max_frequency              900000000
clock_servo                pi
#clock_servo               nullf
sanity_freq_limit          0200000000
#freq_est_interval         0
ntpshm_segment             0
#
# Transport options
#
transportSpecific          0x0
#ptp_dst_mac               01:80:C2:00:00:0E
p2p_dst_mac                01:80:C2:00:00:0E
uds_address                /var/run/ptp4l
ptp_dst_mac                01:1B:19:00:00:00
#p2p_dst_mac               01:1B:19:00:00:00

#
# Default interface options
#
network_transport          L2
delay_mechanism            e2e
time_stamping              hardware
tsproc_mode                raw
delay_filter               moving_median
delay_filter_length        10
#tsproc_mode               filter
#delay_filter              moving_average
#delay_filter_length       200
egressLatency              0
ingressLatency             0
boundary_clock_jbod        0
#[ens801f0]
#serverOnly                1
```

```
#[ens801f1]
#serverOnly               1
```

5.10 Example synce4l Configuration File for BC

Note: For GM mode, ensure that `input_mode = external` is used in the config file.

```
# Global section is for debuging mostly
[global]
#
# Runtime options
#
logging_level    7
use_syslog       0
Verbose          1
message_tag      [synce4l]

#
# Device section
# Per-device configuration
#
# User defined name of a one logical device configured for SyncE in the system.
# All the ports configured after this section will be a part of this device
# (until next device section).
[<synce1>]

#
# Mode for synce4l operation.
# There are two currently supported modes:
# - line
# - external
#
# Input mode "line" mean the inputs recovered from the PHY's.
# The ports configured (in the port-sections [<dev_name>], under the device
# section) will be monitored for the QL (Quality Level).
# QL is sent by the peer connected to the port and represents the Holdover
# performance of the peer.
# The best QL is selected and frequency recovered on that port shall be used
# to feed its frequency to all the other ports.
#
# The "external" input mode are either 1PPS from built-in GPS module or 1PPS
# from the on-board SMA connectors. Device must be pre-confiured to use this
# setting. # Before running the application, one of the external inputs shall be
# selected. This is done through the interface supplied by the adpater vendor.
#
# In this mode synce4l application is only responsible for sending
# the QL to the peers on all configured ports.
# The QL value that is sent equals configured "external_input_QL"
# (and "external_input_ext_QL" in case of "extended_tlv=1").
#
input_mode  line           //use this for SyncE follower
# input_mode    external   //use this for SyncE leader

#
# These values are sent to the peers on configured ports ONLY when 'external'
# mode is enabled.
```

```
# Valid values are defined in Table 11-7 and Table 11-8 of recommendation
# ITU-T G.8264.
# They shall be configured appropriately so they are understood by the peer.
#
# external_input_QL corresponds to the SSM code column.
#
# external_input_ext_QL corresponds to the Enhanced SSM code column
# (is used only if "extended_tlv = 1")
#
external_input_QL        2
external_input_ext_QL    255


#
# If extended TLV shall be supported on the device.
# 0 if no extended tlv shall be supported
# 1 if extended tlv shall be supported
# default: 0
#
# In case of 0:
#       - the port will always TX the non-extended TLV, for RX only
#         non-extended TLV will be processed for reference signal selection
# In case of 1:
#       - If port is configured with external_input=1, the TX will always use
#         extended TLV (no RX is required in this case)
#       - If port is configured with line or external input mode, the TX version
#         of TLV will be propagated from the port that was chosen as
#         candidate for frequency synchronization
#
extended_tlv          1

#
# Which network option shall be supported
#
# 1 or 2 as defined in T-REC-G.8264
# default: 1
#
# This is rather per-network option, all device in SyncE network
# shall have this configured for the same value
#
network_option        1


#
# Seconds indicating minimum time to recover from the QL-failed state on the
# port.
# Range: 10-720
# Default: 300
#
# If valid QL was not received from one of the source ports within 5 seconds
# the port is no longer a valid source (marked as QL-failed)
#
# Valid QL must be received for more then "recover_time" seconds on that port
# to use its PHY recovered signal again as a valid source
#
```

```
recover_time         10

#
# Shell command to be executed in order to obtain current EEC status of a
# device.
#
eec_get_state_cmd          cat /sys/class/net/enp1s0f0/device/dpll_0_state

#
# EEC state values, must equal to values produced by stdout of
# "eec_get_state_cmd" command
#
eec_holdover_value   4
eec_locked_ho_value  3
eec_locked_value     2
eec_freerun_value    1
eec_invalid_value    0

#
# Port section(s)
#
# It starts per-port configuration.
# Each port (of the device) that is used for SyncE, shall have its own section.
#
[enp1s0f0]

#
# msec resolution of TX the QL from this port to the peer
# [100-3000], default:1000 (1000 = 1 second is expected by the standard)
#
# As the standard expects 1 sec, it is not recommended to use different
# than a 1000.
#
tx_heartbeat_msec          1000

#
# recovered PHY signal can be lost at anytime, this is msec resolution of
# reading the socket, acting on signal lost shall be done just after
# [10-500], default:50
#
rx_heartbeat_msec          500

#
# Shell commands for enabling/disabling this port as main recovered clock on a
# device.
#
recover_clock_enable_cmd   echo 1 0 > /sys/class/net/enp1s0f0/device/phy/synce
recover_clock_disable_cmd  echo 0 0 > /sys/class/net/enp1s0f0/device/phy/synce

#
# next configured interface for the device
#
[enp1s0f1]
recover_clock_enable_cmd   echo 1 0 > /sys/class/net/enp1s0f1/device/phy/synce
recover_clock_disable_cmd  echo 0 0 > /sys/class/net/enp1s0f1/device/phy/synce
```

```
############################################################
#
# next SyncE device section
#
#[<synce2>]
#input_mode  line
# input_mode    external

#
# new port belonging to the "new" device
#
#[enp7s0f0]
```

6.0 Initial Test Setup

Two E810-CQDA2T adapters on two systems with a Trimble GM200 used as a timeserver.

6.1 Test Diagram

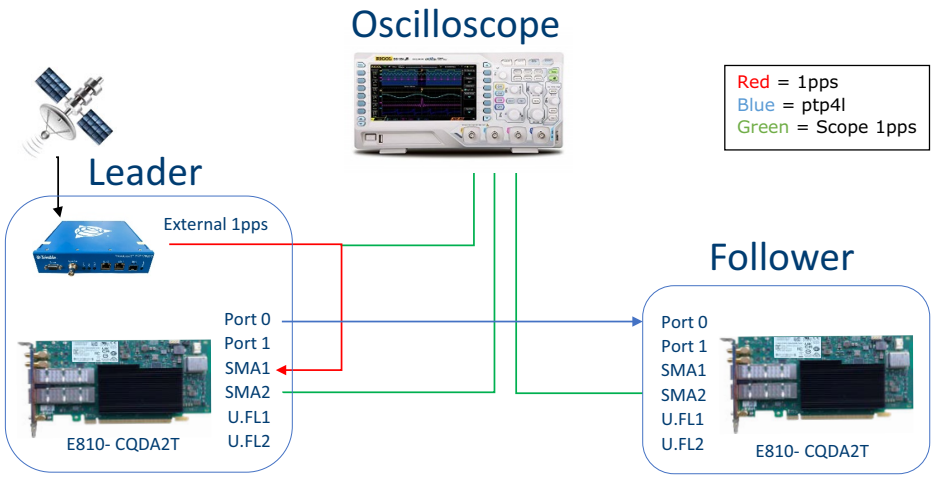


Figure 26. Test Setup

6.2 Software Configuration

Note: Before proceeding, configure a 1PPS signal on the grand leader output.

6.2.1 Leader Adapter

Before proceeding, configure the GNSS to the output 1PPS signal and connect it to the SMA1 connector (or to U.FL2). Set all SMA and U.FL connectors to off (see Section 4.0).

1. Set interface device:

```
# export ETH=`grep 000f /sys/class/net/*/device/subsystem_device | awk -F"/"
'{print $5}' | head -n 1`
```

2. SMA1 as input and SMA2 as output:

```
# echo 1 1 > /sys/class/net/$ETH/device/ptp/ptp*/pins/SMA1
# echo 2 2 > /sys/class/net/$ETH/device/ptp/ptp*/pins/SMA2
```

   Note: As a side effect of having a very accurate DPLL, synchronization between the two E810-CQDA2T adapters can take up to several hours to change to a locked state.

3. Run **ts2phc**:

   Running on Port 0:

```
# ./ts2phc -f configs/ts2phc-generic.cfg -s generic -m
```

4. Run **ptp4l**:

```
# ifconfig $ETH 192.168.2.1
# ./ptp4l -i $ETH -m
```

5. Run **synce4l** (optional):

```
#./synce4l -f configs/synce4l.cfg -m
```

Note: Refer to Section 5.10, "Example synce4l Configuration File for BC".

Note: For GM mode, ensure that `input_mode = external` is used in the config file.

Note: As the default DPLL priority for 1PPS from the SMA is higher than the recover clock's priority.

6.2.2 Follower Adapter

Before proceeding, configure the GNSS to the output 1PPS signal and connect it to the SMA1 connector (or to U.FL2). Set all SMA and U.FL connectors to off (see Section 4.0).

1. Set interface device:

```
# export ETH=`grep 000f /sys/class/net/*/device/subsystem_device | awk -F"/"
'{print $5}' | head -n 1
```

2. SMA2 as output:

```
# echo 2 2 > /sys/class/net/$ETH/device/ptp/ptp*/pins/SMA2
```

3. Set periodic output on SDP20 and SDP22 (To synchronize the DPLL1 to the E810 PHC synced by **ptp4l**):

```
# echo 1 0 0 0 100 > /sys/class/net/$ETH/device/ptp/ptp*/period
# echo 2 0 0 1 0 > /sys/class/net/$ETH/device/ptp/ptp*/period
```

Note: As a side effect of having a very accurate DPLL, synchronization between the two E810-CQDA2T adapters can take up to several hours to change to a locked state.

4. Run **ptp4l**:

```
# ifconfig <network_interface_port0> 192.168.2.2
# ./ptp4l -i <network_interface_port0> -m -s
```

5. Run **synce4l** (optional):

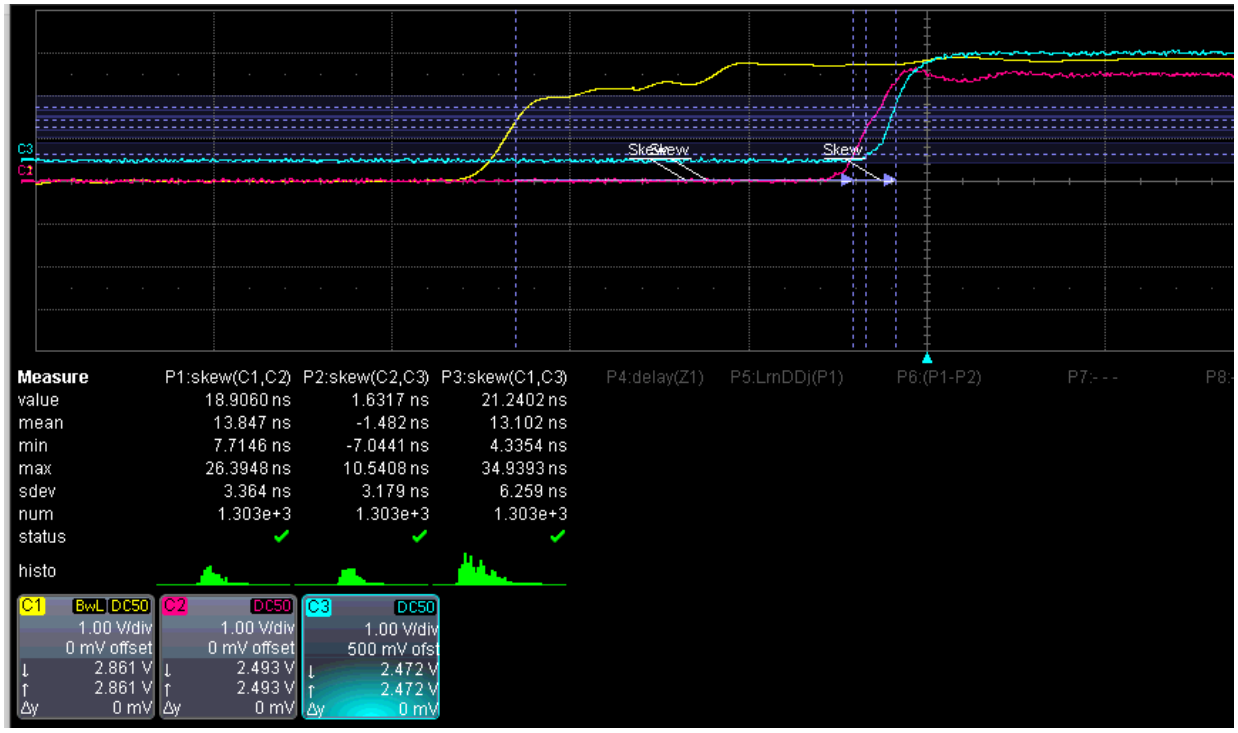
```
#./synce4l -f configs/synce4l.cfg -m
```

Note: Refer to Section 5.10, "Example synce4l Configuration File for BC".

Note: For GM mode, ensure that `input_mode = external` is used in the config file.

Note: As the default DPLL priority for 1PPS from the SMA is higher than the recover clock's priority.

6.2.3 Test Results



C1 = Trimble GM	C2 = Leader E810-CQDA2T	C3 = Follower E810-CQDA2T

Figure 27. Test Results

NOTE: ***This page intentionally left blank.***

Appendix A Debug Notes

- If you cannot find the SMA or ptpX files, you should check that:
  - — You have the latest NVM.
  - — You have the latest driver.
  - — You ran `make install`.
  - — Your kernel support **sysfs** interface (https://www.kernel.org/doc/html/latest/driver-api/pps.html). Otherwise, you should update your kernel to a newer one.
- It can take a couple of hours to synchronize the phase while the frequency is already locked. The DPLL status shows unlocked until both the frequency and phase are synchronized.
- When doing measurement, users need to take into consideration together the cable length that is connected the system, scope, and GM.
- Users also need to check equipment precision as some GMs can have >15 ns accuracy.
- If using newer Linux versions, use **systemd**, which has NTP and as a result sysclock based on NTP when using **phy2sys** commands to turn it off/on.

```
# timedatectl set-ntp true(false)
$ timedatectl status
               Local time: Thu 2015-07-09 18:21:33 CEST
           Universal time: Thu 2015-07-09 16:21:33 UTC
                 RTC time: Thu 2015-07-09 16:21:33
                Time zone: Europe/Amsterdam (CEST, +0200)
System clock synchronized: yes
              NTP service: (in)active
          RTC in local TZ: no
```

  https://wiki.archlinux.org/index.php/systemd-timesyncd

- If users run into a “clock frequency higher than an expected” **ptp4l** error, they might be running **ptp4l** as follower and leader on the wrong system.
- If **ts2phc** shows SKIPS, it is likely that the extts polarity is set wrong in the *ts2phc* config file.
- If **ts2phc** reports:

```
# ts2phc[4980.802]: enp138s0f0 ignoring invalid master time stamp
```

  The problem is that the system time and the GNSS time is too far away and the system time needs to be updated. Set initial system time via one-time update via NTP (using NIST server as an example):

```
ntpd -g -q -x 132.163.97.5
```

- If *tx_timestamp* does not arrive within the specified time, **linuxptp** completely restarts synchronization. Try increasing *Tx_timestamping_timeout* to a larger number, like 10 or 100, but this depends entirely on hardware.

  https://manpages.debian.org/unstable/linuxptp/ptp4l.8.en.html

- Linux kernels going to sleep, edit:

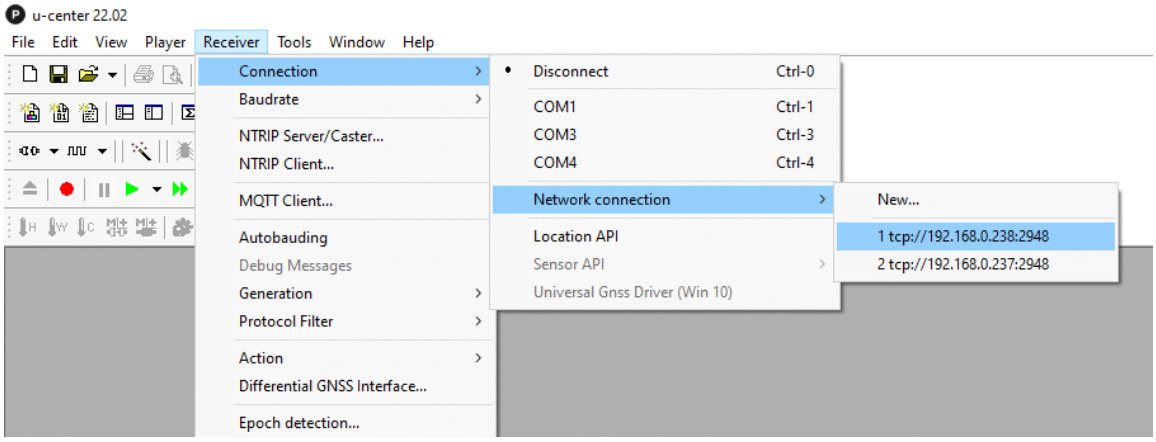
```
/etc/default/grub GRUB_CMDLINE_LINUX_DEFAULT="quiet splash nohz=off" "sudo update-grub"
```

  then restart.

- To check the GNSS receiver performance you might want to use **u-blox U-center** tool. To enable access, you can use the **Netcat** tool that listens on port 2948 for active connection from the U-center:

  ```
  # nc -nvlp 2948 > /dev/gnssX < /dev/gnssX
  ```

  Connecting through TCP in the U-center software:



- If you receive the following using **ts2phc** with GNSS module:

  ```
  nmea: unable to find utc time in leap second table

  ts2phc[1539626.441]: ens785f0 extts index 0 at 1623669837.999999264 corr 0 src
  1623669801.870596298 diff 36999999264

  ts2phc[1539626.441]: ens785f0 ignoring invalid master time stamp
  ```

  then you might be using **linuxptp**, Version 3.1, which is incompatible or your pts number in the config file is wrong. Also make sure you have an appropriate leap second file defined in the *.cfg* file (see Section 5.8).

- If you are experiencing problems where **ethtool -T** does not show Hardware Tx or Rx, you might need to reinsert the Intel *ice* driver with `make install` to include the DPK.

- Some improperly formatted messages, or tools can also “break” the functionality of GNSS module, making it impossible to send configuration changes to GNSS module. Some of these issues can be resolved by restarting the *ice* driver with the following command:

  ```
  rmmod ice; modprobe ice
  ```

- If for some reason you do not want to install **gpsd** on the system, Linux **echo** commands can be used. You must know the hex string sent by **ubxtool** beforehand, keep opened GNSS interface, and send the **echo** command in this format:

  ```
  # cat /dev/gnssX
  # echo -ne "<hex_string>" > /dev/gnssX
  ```

  Example - Open two terminals and run:

  ```
  # cat /dev/gnssX //Terminal 1

  # echo -ne "\xb5\x62\x06\x8a\x09\x00\x00\x02\x00\x00\x05\x00\x52\x10\x00\x02\x68"
  > /dev/gnssX //Terminal 2
  ```

- Intel recommends disabling UART1 and UART2 interfaces on the GNSS receiver. To do that, use:

  1. Disable UART1 in RAM, and Flash:

     ```
     # ubxtool -v 1 -w 1 -P 29.20 -z CFG-UART1-ENABLED,0,5
     ```

     Or open two terminals and run:

     ```
     # cat /dev/gnssX //Terminal 1

     # echo -ne
     "\xb5\x62\x06\x8a\x09\x00\x00\x05\x00\x00\x05\x00\x52\x10\x00\x05\x80" > /dev/
     gnssX //Terminal 2
     ```

  2. Disable UART2 in RAM, and Flash:

     ```
     # ubxtool -v 3 -w 1 -P 29.20 -z CFG-UART2-ENABLED,0,5
     ```

     Or open two terminals and run:

     ```
     # cat /dev/gnssX //Terminal 1

     # echo -ne
     "\xb5\x62\x06\x8a\x09\x00\x00\x05\x00\x00\x05\x00\x53\x10\x00\x06\x83" > /dev/
     gnssX //Terminal 2
     ```

- In some cases, when GNSS loses the antenna reference, the GNSS might output for couple of seconds the 1PPS signal and NMEA messages. To more rapidly disqualify the 1PPS and NMEA messages, increase the filtering of GNSS receiver with **ubxtool** command:

  ```
  # ubxtool -P 29.20 -v 1 -w 1 -z CFG-NAVSPG-OUTFIL_TACC,10,5
  ```

  Or open two terminals and run:

  ```
  # cat /dev/gnssX //Terminal 1

  #echo -ne
  "\xb5\x62\x06\x8a\x0a\x00\x00\x05\x00\x00\xb4\x00\x11\x30\x0a\x00\x9e\x1b" >
  /dev/gnssX //Terminal2
  ```

- Because of a known driver limitation, before enabling the periodic outputs to the DPLL, make sure that your PHC is synchronized (wait for stable **ptp4l** connection with dual digit offset). If you need to do a jump on the PHC clock (like restarting **ptp4l**), you must to disable 1PPS and 10 MHz signal to DPLL. If the clock jumped, the periodic outputs to the DPLL might not ever show a “valid” state in the CGU Linux interface. To restore correct operation of DPLL, reload the driver:

  ```
  # rmmod ice; modprobe ice
  ```

NOTE: ***This page intentionally left blank.***

Appendix B Glossary and Acronyms

Table 9. Definition of Terms

Term	Definition
AAU	Active Antenna Unit
BBU	Baseband Unit
BC	Boundary Clock
BMC	Best Main Clock
CN	Core Network
CPRI	Common Public Radio Interface
CU	Centralized Unit
DU	Distributed Unit
EEC	Ethernet Equipment Clock
eMBB	Enhanced Mobile Broadband
EPC	Evolved Packet Core
Extts	External Timestamp
GMC	Grand Main Clock
GNSS	Global Navigation Satellite System (include GPS, Galileo, Glonass, Beidou, QZSS, and NavIC)
MIMO	Multiple-Input Multiple-Output
mMTC	Massive Machine Type Communication
ms	Milliseconds
NGCN	Next Generation Core Network
ns	Nanoseconds
NTP	Network Time Protocol
OC	Ordinary Clock
OCXO	Oven-Controlled Crystal Oscillator
OTII	Open Telecom IT Infrastructure
PDCP	Packet Data Convergence Protocol
PHC	PTP Hardware Clock
ppb	Parts Per Billion
ppm	Parts Per Million
PRC	Primary Reference Clock
PTP	Precision Time Protocol
ptp4l	PTP for Linux
RAN	Radio Access Network
RU	Radio Unit
S	Seconds
SSU	Synchronization Supply Unit
SyncE	Synchronous Ethernet
TC	Transparent Clock

Table 9. Definition of Terms [continued]

Term	Definition
ToD	Time of Day
ToS	Top of Second
uRLLC	Ultra-Reliable and Low Latency Communications
µs	Microseconds

NOTE: ***This page intentionally left blank.***

LEGAL