



Intel® Ethernet Network Adapter E810-XXVDA4T

User Guide

NEX Cloud Networking Group (NCNG)

April 2025

Revision 1.7
646265

Revision History

Revision	Date	Comments
1.7	April 14, 2025	<p>Updates include the following:</p> <ul style="list-style-type: none"> • Updated Section 1.0, "Introduction". • Updated Section 1.1, "Reference Documents". • Updated Section 3.1, "Software Support/Packages". • Updated Section 3.2, "Building a linuxptp Project". • Updated Section 3.3, "Related linuxptp Information". • Updated Section 4.1, "Introduction". • Updated Section 4.2, "DPLL Priority". • Updated Section 4.3, "External Connectors". • Updated Section 4.4, "SMA1 and U.FL1 Configurations" (previous Channel 1 Configurations). • Updated Section 4.5, "SMA2 and U.FL2 Configurations" (previous Channel 2 Configurations). • Added Section 4.6, "Configuring SDP Pins". • Updated Section 4.7, "Recovered Clocks (G.8261 SyncE Support)". • Updated Section 4.8, "External Timestamp Signals". • Updated Section 4.9, "Periodic Outputs from DPLL (SMA and U.FL Pins)". • Updated Section 4.10, "Reading Status of the DPLL". • Updated Section 4.12.1, "pin_cfg User Readable Format". • Updated Section 4.13, "1PPS Signals from E810 Device to DPLL". • Updated Section 4.15, "GNSS Module Interface". • Updated Section 4.15, "GNSS Module Interface". • Updated Section 4.16.8, "Simulate Antenna Removal". • Updated Section 5.1, "Disable All SMA and U.FL 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 5.6.5, "Two E810-XXVDA4T Adapter Configuration without GNSS and with 1PPS". • Updated Section 5.6.6, "Two E810-XXVDA4T Adapters with GNSS Connection Setup". • Updated Section 5.10, "Example sync4l Configuration File for BC". • Updated Section 6.2.1, "TimeTransmitter Adapter". • Updated Section 6.2.2, "TimeReceiver Adapter". • Added Section 7.0, "The Linux Kernel DPLL API".

Revision	Date	Comments
1.6	July 4, 2024	<p>Updates include the following:</p> <ul style="list-style-type: none"> Updated Section 1.0, "Introduction". Updated Table 1, "Reference Documents". Updated Section 2.1, "E810-XXVDA4T Features". Updated Table 2, "E810-XXVDA4T Product Information". Updated Section 2.2, "Architecture". Updated Section 3.1, "Software Support/Packages". Updated Section 3.2, "Building a linuxptp Project". Updated Section 3.4, "Building a synce4l Tool". Updated Section 4.4, "SMA1 and U.FL1 Configurations". Updated Section 4.5, "Channel 2 Configurations". Updated Section 4.6, "Configuring SDP Pins". Updated Section 4.8, "External Timestamp Signals". Updated Section 4.9, "Periodic Outputs from DPLL (SMA and U.FL Pins)". Updated Section 4.10, "Reading Status of the DPLL". Updated Section 4.11, "DPLL Monitoring". Updated Section 4.12.1, "pin_cfg User Readable Format". Updated Section 4.13, "1PPS Signals from E810 Device to DPLL". Updated Section 4.15, "GNSS Module Interface". Updated Section 7, "Supported GNSS and Signals on ZED-F9T=00B". Updated Section 4.16.5, "Perform Survey-In for New Location Setup". Updated Section 4.16.7, "Saving Survey-In Position to FLASH". Updated Section 4.16.8, "Simulate Antenna Removal". Updated Section 4.16.9, "Check GNSS Overall Configuration Performance". Updated Figure 5, "External Connections: PTP GrandMaster with Optional GNSS Module" on page 42. Updated Figure 6, "External Connections: Single E810-XXVDA4T Adapter Configuration with GNSS" on page 42. Updated Section 5.2.2, "Software Configuration". Updated Figure 8, "External Connections: PTP GrandMaster with External GNSS Clock" on page 44. Updated Section 5.3.2, "Software Configuration". Updated Figure 9, "External Connections: Boundary Clock Configuration" on page 46. Updated Section 5.4.3, "Software Configuration". Updated Figure 10, "External Connections: Port Configured as TimeReceiver" on page 48. Updated Figure 11, "External Connections: SyncE Setup" on page 50. Updated Section 5.6.2, "Physical Clock Recovery". Updated Section 5.6.3, "ITU G.8264 ESMC Messaging Using synce4l". Updated Figure 12, "External Connections: Two E810-XXVDA4T Adapters without GNSS" on page 51. Updated Figure 16, "External Connections: Two E810-XXVDA4T Adapter Configuration (no GNSS)" on page 54. Updated Figure 20, "External Connections: Two E810-XXVDA4T Adapter Configuration (with GNSS)" on page 57. Updated Figure 25, "O-RAN Configuration 1 Connections" on page 60. Updated Section 5.7.2, "Software Configuration". Updated Section 5.9, "Example ptp4l Configuration File for BC". Updated Section 5.10, "Example synce4l Configuration File for BC". Updated Figure 26, "Test Setup" on page 67. Updated Section 6.2.1, "TimeTransmitter Adapter". Updated Section 6.2.2, "TimeReceiver Adapter". Updated Figure 27, "Test Results" on page 69. Updated Appendix A, "Debug Notes".

Revision	Date	Comments
1.5	October 18, 2023	<p>Updates include the following:</p> <ul style="list-style-type: none"> Updated Section 2.1, "E810-XXVDA4T Features". Updated operating temperature in Table 2.1, "E810-XXVDA4T Features". Updated Section 3.1, "Software Support/Packages". Updated Section 3.2, "Building a linuxptp Project". Updated Section 4.4, "SMA1 and U.FL1 Configurations". Updated Section 4.5, "Channel 2 Configurations". Updated Section 4.6, "Configuring SDP Pins". Updated Section 4.8, "External Timestamp Signals". Updated Section 4.10, "Reading Status of the DPLL". Updated Section 4.11, "DPLL Monitoring". Updated Section 4.12, "Advanced DPLL Configuration". Updated Section 4.13, "1PPS Signals from E810 Device to DPLL". Updated Section 4.14, "1PPS Signals from the DPLL to E810 Device". Updated Section 4.15, "GNSS Module Interface". Updated Section 4.16, "GNSS Advanced Features". Updated Section 4.16.1, "Prerequisites and Steps to Fully Enable GNSS Features". Added Section 4.16.2, "GNSS Receiver Configuration Layers in the E810-XXVDA4T". Added Section 4.16.4, "Enabling and Disabling Additional Constellations". Updated Section 4.16.5, "Perform Survey-In for New Location Setup". Updated Section 4.16.6, "Check Survey-In Status". Added Section 4.16.7, "Saving Survey-In Position to FLASH". Added Section 4.16.8, "Simulate Antenna Removal". Updated Section 4.16.9, "Check GNSS Overall Configuration Performance". Updated Section 5.2.2, "Software Configuration". Updated Section 5.3.2, "Software Configuration". Updated Section 5.6.6, "Two E810-XXVDA4T Adapters with GNSS Connection Setup". Updated Section 5.8, "Example ts2phc Configuration File". Updated Section 5.9, "Example ptp4l Configuration File for BC". Updated Section 6.2.1, "TimeTransmitter Adapter". Updated Appendix A, "Debug Notes".
1.4	March 7, 2023	<p>Updates include the following:</p> <ul style="list-style-type: none"> Updated Section 3.1, "Software Support/Packages". Updated Section 4.2, "DPLL Priority". Updated Section 4.6, "Configuring SDP Pins". Updated Section 4.8, "External Timestamp Signals". Updated Section 4.10, "Reading Status of the DPLL". Updated Section 4.11, "DPLL Monitoring". Updated Section 4.12.1, "pin_cfg User Readable Format". Updated Section 4.13, "1PPS Signals from E810 Device to DPLL". Updated Section 4.14, "1PPS Signals from the DPLL to E810 Device". Updated Section 4.15, "GNSS Module Interface". Updated Section 4.16, "GNSS Advanced Features". Updated Section 4.16.1, "Prerequisites and Steps to Fully Enable GNSS Features". Updated Section 4.16.3, "Perform Antenna Status Check New Location Setup". Updated Section 4.16.5, "Perform Survey-In for New Location Setup". Updated Section 4.16.6, "Check Survey-In Status". Updated Section 4.16.9, "Check GNSS Overall Configuration Performance". Updated Section 5.2.2, "Software Configuration". Updated Section 5.4.3, "Software Configuration". Updated Section 5.5.2, "Software Configuration". Updated Section 5.6.5, "Two E810-XXVDA4T Adapter Configuration without GNSS and with 1PPS". Updated Section 5.8, "Example ts2phc Configuration File". Updated Section 5.9, "Example ptp4l Configuration File for BC". Updated Section 6.2.2, "TimeReceiver Adapter". Updated Appendix A, "Debug Notes".

Revision	Date	Comments
1.3	December 15, 2022	<p>Updates include the following:</p> <ul style="list-style-type: none"> Updated GNSS interface naming to /dev/ttyGNSS_BBDD. Added Section 2.3, "Synchronization Signaling". Added Section 2.4, "Optional GNSS Module". Added Section 3.4, "Building a sync4I Tool". Updated Section 4.1, "Introduction". Added notes to Section 4.8, "External Timestamp Signals". Updated Section 4.10, "Reading Status of the DPLL". Updated Section 4.12.1, "pin_cfg User Readable Format". Updated Section 4.12.8, "dpll_<X>_ref_pin/dpll_<X>_state Machine Readable Interface (X = 0 /1)". Updated Section 4.15, "GNSS Module Interface". Added Section 4.16, "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 5.6, "SyncE Setup". Added Section 5.6.3, "ITU G.8264 ESMC Messaging Using sync4I". Added Section 5.6.4, "Two E810-XXVDA4T Adapter Configuration without GNSS". Added Section 5.6.5, "Two E810-XXVDA4T Adapter Configuration without GNSS and with 1PPS". Added Section 5.6.6, "Two E810-XXVDA4T Adapters with GNSS Connection Setup". Added Section 5.10, "Example sync4I Configuration File for BC". Updated Section 6.2.1, "TimeTransmitter Adapter". Updated Section 6.2.2, "TimeReceiver Adapter". Updated Appendix A, "Debug Notes".
1.2	June 15, 2022	<p>Updates include the following:</p> <ul style="list-style-type: none"> Updated Section 3.1, "Software Support/Packages". Updated Section 5.9, "Example ptp4I Configuration File for BC".
1.1	March 25, 2022	<p>Updates include the following:</p> <ul style="list-style-type: none"> Updated Section 3.1, "Software Support/Packages". Updated Section 4.10, "Reading Status of the DPLL". Added Section 4.12.1, "pin_cfg User Readable Format". Added Section 4.12.8, "dpll_<X>_ref_pin/dpll_<X>_state Machine Readable Interface (X = 0 /1)". Updated Section 4.15, "GNSS Module Interface". Updated Section 5.2.2, "Software Configuration". Updated Section 5.3.2, "Software Configuration". Updated Appendix A, "Debug Notes".
1.0	February 18, 2022	Initial public release.



NOTE: *This page intentionally left blank.*

Contents

1.0	Introduction	3
1.1	Reference Documents	3
2.0	E810-XXVDA4T Ethernet Network Adapter	5
2.1	E810-XXVDA4T Features	5
2.2	Architecture	7
2.3	Synchronization Signaling	9
2.4	Optional GNSS Module	9
3.0	Software, Firmware, and Drivers	10
3.1	Software Support/Packages	10
3.2	Building a linuxptp Project	11
3.3	Related linuxptp Information	12
3.4	Building a sync4I Tool	12
4.0	Configuring the E810-XXVDA4T Using the Linux API (PHC)	13
4.1	Introduction	13
4.2	DPLL Priority	14
4.3	External Connectors	15
4.4	SMA1 and U.FL1 Configurations	15
4.5	SMA2 and U.FL2 Configurations	17
4.6	Configuring SDP Pins	18
4.7	Recovered Clocks (G.8261 SyncE Support)	20
4.8	External Timestamp Signals	21
4.9	Periodic Outputs from DPLL (SMA and U.FL Pins)	21
4.10	Reading Status of the DPLL	22
4.11	DPLL Monitoring	23
4.12	Advanced DPLL Configuration	24
4.12.1	pin_cfg User Readable Format	24
4.12.2	Changing the DPLL priority list	26
4.12.3	Changing input/output pin configuration	26
4.12.4	Modifying frequency, phase-adjust and eSync using "The Linux Kernel DPLL API"	27
4.12.5	Changing priority, direction, and enabling/disabling pins on "The Linux Kernel DPLL API"	28
4.12.6	Setup Script for version 4.80 and later	28
4.12.7	Using DPLL api	29
4.12.8	dpll_<X>_ref_pin/dpll_<X>_state Machine Readable Interface (X = 0 /1)	29
4.13	1PPS Signals from E810 Device to DPLL	30
4.14	1PPS Signals from the DPLL to E810 Device	30
4.15	GNSS Module Interface	31
4.16	GNSS Advanced Features	32
4.16.1	Prerequisites and Steps to Fully Enable GNSS Features	32
4.16.2	GNSS Receiver Configuration Layers in the E810-XXVDA4T	35
4.16.3	Perform Antenna Status Check New Location Setup	36
4.16.4	Enabling and Disabling Additional Constellations	37
4.16.5	Perform Survey-In for New Location Setup	37
4.16.6	Check Survey-In Status	38
4.16.7	Saving Survey-In Position to FLASH	38
4.16.8	Simulate Antenna Removal	38
4.16.9	Check GNSS Overall Configuration Performance	39
5.0	Configuration Setup	40
5.1	Disable All SMA and U.FL Connections	41
5.2	PTP GrandMaster (GM) with Optional GNSS Module	41
5.2.1	External Connections	42

5.2.2	Software Configuration	43
5.3	PTP GrandMaster (GM) with External GNSS Clock	44
5.3.1	External Connections	44
5.3.2	Software Configuration	44
5.4	Boundary Clock Configuration	46
5.4.1	External Connections	46
5.4.2	Boundary Clock Notes	46
5.4.3	Software Configuration	46
5.5	Port Configured as TimeReceiver	48
5.5.1	External Connections	48
5.5.2	Software Configuration	48
5.6	SyncE Setup	50
5.6.1	External Connections	50
5.6.2	Physical Clock Recovery	50
5.6.3	ITU G.8264 ESMC Messaging Using sync4l	51
5.6.4	Two E810-XXVDA4T Adapter Configuration without GNSS	51
5.6.5	Two E810-XXVDA4T Adapter Configuration without GNSS and with 1PPS	54
5.6.6	Two E810-XXVDA4T Adapters with GNSS Connection Setup	57
5.7	O-RAN Configuration 1	60
5.7.1	External Connections	60
5.7.2	Software Configuration	60
5.8	Example ts2phc Configuration File	61
5.9	Example ptp4l Configuration File for BC	61
5.10	Example sync4l Configuration File for BC	63
6.0	Initial Test Setup	67
6.1	Test Diagram	67
6.2	Software Configuration	67
6.2.1	TimeTransmitter Adapter	67
6.2.2	TimeReceiver Adapter	68
6.2.3	Test Results	69
7.0	The Linux Kernel DPLL API	70
7.1	Overview	70
7.2	Commands	70
7.2.1	--dump device-get	70
7.2.2	--dump pin-get	71
7.2.3	--do device-get	77
7.2.4	--do device-id-get	77
7.2.5	--do pin-get	78
7.2.6	--do pin-id-get	80
7.2.7	--do pin-set	80
7.3	Monitoring DPLL Status	82
7.4	DPLL api Setup Script for version 4.80 and later	83
Appendix A	Debug Notes	85
Appendix B	Glossary and Acronyms	90

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 TimeTransmitter 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-XXVDA4T (E810-XXVDA4T), is based on the Intel® Ethernet Controller E810 (E810).

This document is structured as follows:

- [Section 2.0, "E810-XXVDA4T Ethernet Network Adapter"](#)
- [Section 3.0, "Software, Firmware, and Drivers"](#)
- [Section 4.0, "Configuring the E810-XXVDA4T Using the Linux API \(PHC\)"](#)
- [Section 5.0, "Configuration Setup"](#)
- [Section 6.0, "Initial Test Setup"](#)
- [Section 7.0, "The Linux Kernel DPLL API"](#)
- [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 Time-Transmitter (TT) and Time-Receiver (TR), previous version of the document used 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	630155
Intel® Ethernet Network Adapter E810-XXVDA4T Overview Video	730682
Intel® Ethernet Network Adapter E810-XXVDA4T Product Brief	641626
Intel® Ethernet Network Adapter E810-XXVDA4T - 68808 - MDDS	727036
Intel® Ethernet Network Adapter E810-XXVDA4T Product Brief	641626

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: <https://linuxptp.nwtime.org/about/features/>

2.0 E810-XXVDA4T Ethernet Network Adapter

2.1 E810-XXVDA4T Features

Following are the features of the Intel® Ethernet Network Adapter E810-XXVDA4T:

- 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-XXVDA4 (E810-XXVDA4).
- 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 GrandMaster, 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 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-XXVDA4T also includes two U.FL connectors for 1PPS and/or 10 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 $\pm 1.5 \mu\text{s}$.
- 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 I²C interface provides a way to access the serial port data on the GNSS, including configuration options.

Table 2 provides additional information on the E810-XXVDA4T:

Table 2. E810-XXVDA4T Product Information

Product Description	1588 PTP/SyncE/GNSS 10/25GbE SFP28 PCIe adapter ¹
Product Codes	E810-XXVDA4TG1 (no GNSS) E810-XXV-DA4TGG1 (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	4x SFP28, 2x SMA (1PPS/10 MHz) ² , 1x SMB (GNSS antenna) ³
Internal Connectors	1x U.FL input + 1x U.FL output (1PPS/10 MHz); 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 ($\pm 1.5 \mu\text{s}$ under $\pm 5^\circ\text{C } \Delta T$)
Power	23 W typical, 34 W max power consumption (with Class 3 optics at 25G maximum traffic)
Operating Temperature	0-65 °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. 10M/100M speeds are not supported.
2. Refer to [Section 2.3](#) for 1PPS and 10 MHz signaling requirements.
3. Refer to [Table 3](#) for GNSS antenna characteristics.

2.2 Architecture

The block diagram for the E810-XXVDA4T is shown in Figure 1. The E810-XXVDA4T provides two coaxial input/output SMA connectors, two U.FL connectors, and an optional GNSS input connector, as shown in Figure 2.

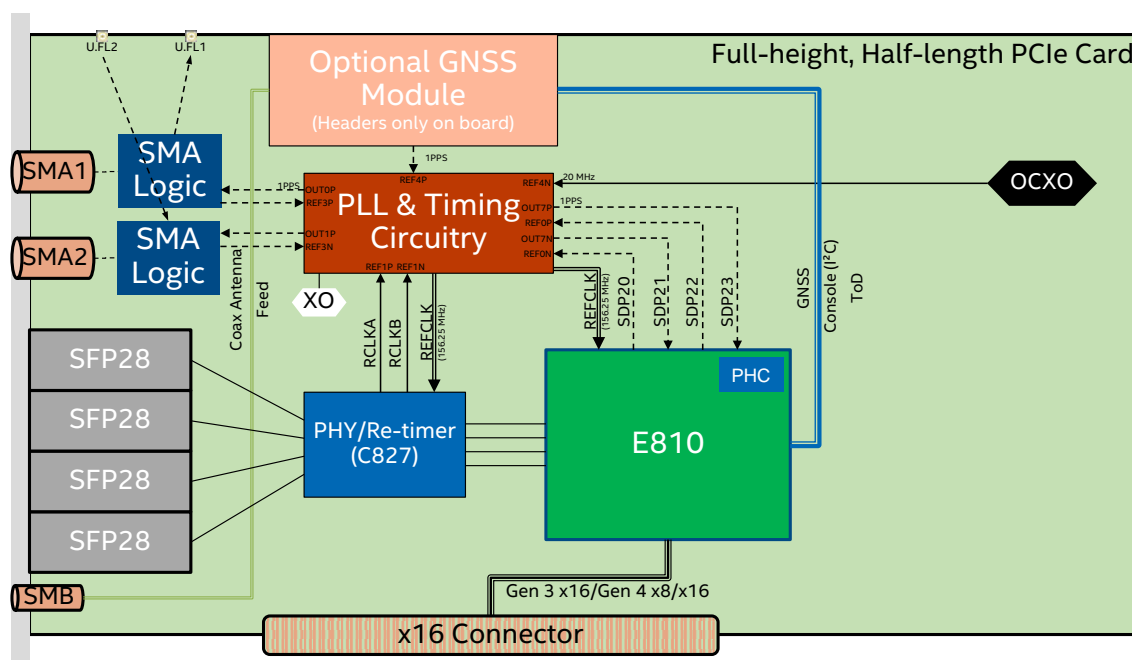


Figure 1. E810-XXVDA4T Block Diagram

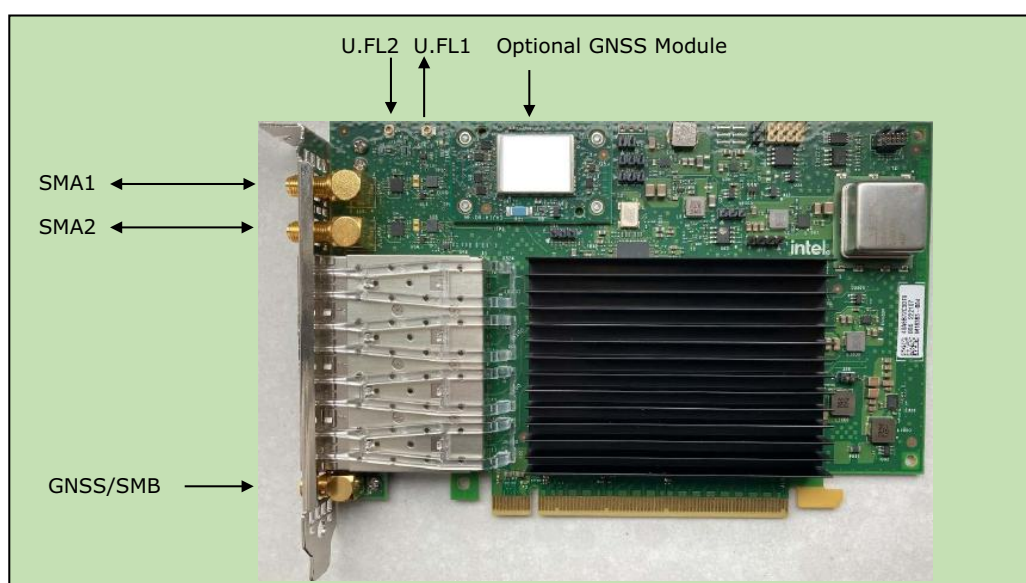


Figure 2. E810-XXVDA4T Connector Locations

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-XXVDA4T GNSS Antenna Characteristics

Characteristic	Min	Typical	Max
Characteristic impedance	-	50 Ω	-
DC operating voltage	2.9 V	3.3 V	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 WPC 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	-

Note: If the antenna is mounted on the open space, please make sure that lightning protection is used.

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-XXVDA4T 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-XXVDA4T 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-XXVDA4T) and bias tees (for signaling received by the E810-XXVDA4T Rx) must be used.

Table 5. E810-XXVDA4T 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-XXVDA4T 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-XXVDA4T antenna input is included on this module.

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:

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://github.com/intel/ethernet-linux-ice>

Note: For the newest features, always use the latest driver and NVM version. This document is based on driver version *ice-1.17.2* or newer.

Important: The latest ice-1.17.2 driver with the 4.80 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
# rmmod irdma
```

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_80_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: 1.17.2  
firmware-version: 4.80 0x80020682 1.3805.0  
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. Copy and install DDP into the following directory:

Note: This should only be done if DDP was not installed properly through “make install”.

Remove previous DDP installations:

```
# rm -rf /lib/firmware/intel/ice/ddp/*
```

Copy DDP into the following directory:

```
# cd ice-1.x.x/ddp  
# cp ice-1.3.39.1.pkg /lib/firmware/intel/ice/ddp/
```

Create a soft link for the DDP:

```
# ln -s /lib/firmware/intel/ice/ddp/ice-1.3.39.1.pkg /lib/firmware/intel/ice/ddp/  
ice.pkg
```

3.2 Building a linuxptp Project

1. Download the latest **linuxptp** release from nwttime (v4.21 or later):

<https://downloads.nwttime.org/linuxptp/>

2. or use:

```
# git clone https://github.com/nwttime/linuxptp linuxptp-4.4
```

3. Extract the downloaded archive, if required:

```
# tar xzf linuxptp-4.4.tgz
```

4. Navigate to the *linuxptp-4.4* directory and compile the source code:

```
# cd linuxptp-4.4  
# make
```

5. 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.

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

<https://linuxptp.nwtime.org>

Note: As the E810 family do not support the PTM technology, phc2sys will work only in SW mode.

3.3 Related linuxptp Information

Refer to this link for further information: <https://linuxptp.nwtime.org/documentation>

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
```

Note: For synce4l the libnl3-devel package need to be installed.

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>

Note: In this version of this document, refer to version 1.1.0 of the synce4l tool.

4.0 Configuring the E810-XXVDA4T Using the Linux API (PHC)

4.1 Introduction

The Linux kernel provides the standard interface for controlling external synchronization pins. New Linux DPLL API has been defined after kernel 6.7. The driver will support both the new DPLL API and also the legacy sysfs interface.

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-XXVDA4T adapters, run the following:

```
# grep 000e /sys/class/net/*/device/subsystem_device | awk -F"/" '{print $5}'
ens4f0np0
ens4f1np1
ens4f2np2
ens4f3np3
```

The example above shows one network device with four ports (f0-f3).

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

```
# export ETH=ens4f0np0
```

Or:

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

Note: This alternate command can only be used if one WPC 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 ens4f0np0 net interface exposes pins through the ptp2 interface:

```
#ls -R /sys/class/net/$ETH/device/ptp/ptp*/pins/
/sys/class/net/ens4f0np0/device/ptp/ptp2/pins:
SDP20 SDP21 SDP22 SDP23
```

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

```
# ethtool -T ens4f0np0 | grep
Clock

PTP Hardware Clock: 2
```

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-XXVDA4T automatically switches reference inputs according to the default DPLL priority list, as shown in [Table 6](#).

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 4.12, “Advanced DPLL Configuration”](#).

Note: From firmware version 4.20 or newer, the DPLL priority list and the DPLL configuration parameters have been changed 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 6. DPLL Priority List

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

Notes: We advise the user to use the DPLL priorities listed in “Recommended Priority” column, using the commands listed in [Section 4.12.6, “Setup Script for version 4.80 and later”](#). Below is the cgu output after applying the recommended DPLL settings:

```
# cat /sys/kernel/debug/ice/0000:81:00.0/cgu
```

CGU Input status:

input (idx)	state	priority		ESync fail
		EEC (0)	PPS (1)	
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)	invalid	0	0	N/A

Notes: In version 4.80 and later, to obtain the correct expected priorities, you need to run a setup script. See [Section 4.12.6, “Setup Script for version 4.80 and later”](#) for details.

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-XXVDA4T 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.

Notes:

- SMA and UFL pins can be configured through the DPLL API and through sysfs.
- If you need to use a different frequency on the SMA or U.FL than 1PPS, you will need to configure the DPLL accordingly.

These interfaces can be written to as shown below:

```
echo <function> > /sys/class/net/$ETH/device/ (SMA1,SMA2,U.FL1,U.FL2)
```

where:

```
function: 0 = Disabled
          1 = Rx
          2 = Tx
```

4.4 SMA1 and U.FL1 Configurations

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

```
# export ETH=ens4f0np0
Or
# export ETH=`grep 000e /sys/class/net/*/device/subsystem_device | awk -F"/" '{print $5}' | head -n 1`
```

2. SMA1 as 1PPS input:

```
# echo 1 > /sys/class/net/$ETH/device/SMA1
```

To double check:

```
# cat /sys/class/net/$ETH/device/SMA1
1
```

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

3. SMA1 as 1PPS output:

```
# echo 2 > /sys/class/net/$ETH/device/SMA1
```

To double check:

```
# cat /sys/class/net/$ETH/device/SMA1
2
```

Note: Setting SMA1 as TX automatically disables U.FL1.

4. U.FL1 as 1PPS output:

```
# echo 2 > /sys/class/net/$ETH/device/U.FL1
```

To double check:

```
# cat /sys/class/net/$ETH/device/U.FL1
2
```

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

5. Disable SMA1:

```
# echo 0 > /sys/class/net/$ETH/device/SMA1
```

To double check:

```
cat /sys/class/net/$ETH/device/SMA1
0
```

6. Disable U.FL1:

```
# echo 0 > /sys/class/net/$ETH/device/U.FL1
```

To double check:

```
# cat /sys/class/net/$ETH/device/U.FL1
0
```

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
```

Note: In VMware, for enabling 1PPS output over SMA1, period file should be configured. Otherwise, input is not valid.

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

Note: In VMware, for enabling 1PPS input over SMA, ts2phc needs to be enabled. Otherwise, input is not valid.

4.5 SMA2 and U.FL2 Configurations

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

```
# export ETH=ens4f0np0
Or
# export ETH=`grep 000e /sys/class/net/*/device/subsystem_device | awk -F"/" '{print $5}' | head -n 1`
```

2. SMA2 as 1PPS input:

```
# echo 1 > /sys/class/net/$ETH/device/SMA2
```

To double check:

```
# cat /sys/class/net/$ETH/device/SMA2
1
```

Note: Setting SMA2 as RX automatically disables U.FL2.

3. SMA2 as 1PPS output:

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

Note: Setting SMA2 as TX automatically enables U.FL2 as RX.

4. U.FL2 as 1PPS input:

```
# echo 1 > /sys/class/net/$ETH/device/U.FL2
```

Note: Setting U.FL2 as a RX automatically enables SMA2 as TX.

5. Disable SMA2:

```
# echo 0 > /sys/class/net/$ETH/device/SMA2
```

6. Disable U.FL2:

```
# echo 0 > /sys/class/net/$ETH/device/U.FL2
```

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
```

Note: In VMware, for enabling 1PPS output over SMA2, period file should be configured. Otherwise, input is not valid.

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

Note: In VMware, for enabling 1PPS input over SMA, ts2phc needs to be enabled. Otherwise, input is not valid.

4.6 Configuring SDP Pins

There are four configurable SDP pins (Software Defined Pins) found on the E810-XXVDA4T. They are labeled as (SDP20, SDP21, SDP22, SDP23). These can be found by listing the following directory:

```
# ls /sys/class/net/$ETH/device/ptp/ptp*/pins/
SDP20 SDP21 SDP22 SDP23
```

SDP pin configuration consists of three steps:

- A. Assigning function and channel number to an SDP
 - B. Using the chosen channel configure the SDP settings by writing to a "period" file
 - C. Setting the same by writing to a pin_cfg file, to ensure DPLL locking to signal
1. To run scripts in this section, set ETH to point to the first port of the adapter:

```
# export ETH=ens4f0np0
Or
# export ETH=`grep 000e /sys/class/net/*/device/subsystem_device | awk -F"/" '{print $5}' | head -n 1`
```

2. Assign a function and a channel number to the specific SDP as seen in the command below:

```
# echo <function> <channel> > /sys/class/net/$ETH/device/ptp/*/pins/SDPx
```

where:

function: 0 = Disabled
1 = Rx
2 = Tx

channel: 0,1,2. Important to use unique channel number per pin on each device.

Note: SDP20 and SDP22 are output pins only. SDP21 and SDP23 are input pins only. The SDP pins connected to the DPLL directly on the E810-XXVDA4T NIC.

- SDP20 is input pin 1 to the DPLL
- SDP22 is input pin 0 to the DPLL
- SDP21 is output pin 4 from the DPLL
- SDP23 is output pin 5 from the DPLL

3. SDP20 as 1PPS output:

For example, to assign output function with channel number 1 on SDP20, use the following command:

```
# export ETH=ens4f0np0
# echo 2 1 > /sys/class/net/$ETH/device/ptp/ptp*/pins/SDP20
# cat /sys/class/net/$ETH/device/ptp/ptp*/pins/SDP20
2 1
```

4. SDP21 as 1PPS input:

For example, to assign input function with channel number 0 on SDP21, use the following command:


```
# echo 1 0 > /sys/class/net/$ETH/device/ptp/ptp*/pins/SDP21
# cat /sys/class/net/$ETH/device/ptp/ptp*/pins/SDP21
1 0
```

5. SDP22 as 1PPS output:

For example, to assign output function with channel number 1 on SDP22, use the following command:

```
# echo 2 1 > /sys/class/net/$ETH/device/ptp/ptp*/pins/SDP22
# cat /sys/class/net/$ETH/device/ptp/ptp*/pins/SDP22
```

Note: Ensures that you are not using the same channel number for different SDP.

6. SDP23 as 1PPS input:

For example, to assign input function with channel number 0 on SDP23, use the following command:

```
# echo 1 0 > /sys/class/net/$ETH/device/ptp/ptp*/pins/SDP23
# cat /sys/class/net/$ETH/device/ptp/ptp*/pins/SDP23
1 0
```

7. In order to get 1PPS output from SDP20 and SDP22 pins, the period function shall be configured on the correct channel number (0,1,2), by using the following command:

```
echo <channel> <start-time (s)> <start-time (ns)> <frequency (s)> <frequency (ns)>
/sys/class/net/$ETH/ptp/ptp*/period
```

where:

channel: 0,1,2. The channel number you want to assign a periodic function to.

start-time (s): Start time in seconds, works as a delay in start time.

start-time (ns): Start time in nano-seconds, works as a delay in start time.

frequency (s): Set frequency over seconds.

frequency (ns): Set frequency over nanoseconds.

8. Example of complete configurations:

a. Using channel 1 to set SDP20 as an output with frequency 10 MHz:

```
# echo 2 1 > /sys/class/ptp/ptp*/pins/SDP20
# echo 1 0 0 0 100 > /sys/class/net/$ETH/device/ptp/ptp*/period
# echo "in pin 1 freq 10000000" > /sys/class/net/$ETH/device/pin_cfg
```

b. Using channel 2 to set SDP22 as an output with frequency 1 Hz:

```
# echo 2 2 > /sys/class/ptp/ptp*/pins/SDP22
# echo 2 0 0 1 0 > /sys/class/net/$ETH/device/ptp/ptp*/period
# echo "in pin 0 freq 1" > /sys/class/net/$ETH/device/pin_cfg
```

Note: Refer to [Section 4.12.1, “pin_cfg User Readable Format”](#) for further information about configuring the pin_cfg.

4.7 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 the C827 PHY. Only one pin can be assigned to one of the ports. 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.

Note: To configure the recover clocks we suggest to use the `sync4l` utility. `Sync4l` automatically enables, and disables recovered clocks.

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 (higher priority pin).
      1 = Enable C827_0-RCLKB (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
# 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 0000:03:00.0: Enabled recovered clock: pin C827_0-RCLKB
```

2. 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 0000:03:00.0: Disabled recovered clock: pin C827_0-RCLKB
```

3. 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 4.12.1](#).

Note: To complete same operation using DPLL API, see [Section 4.12.1](#).

4.8 External Timestamp Signals

The E810-XXVDA4T can use external 1PPS signals filtered 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 E810-XXVDA4T time to the 1PPS signal. The configuration file needs to be edited before using the application. You must 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-4.3/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
ts2phc.pin_index 1
```

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: The pin index can be configured within the ts2phc configuration file, and will overwrite pre-configured values.

Note: You might want filter the 1PPS timestamps till the DPLL locked. See [Section 4.13, "1PPS Signals from E810 Device to DPLL"](#).

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

The E810-XXVDA4T 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 > /sys/class/net/$ETH/device/SMA1
```

2. U.FL1 as 1PPS output:

```
# echo 2 > /sys/class/net/$ETH/device/U.FL1
```

3. SMA2 as 1PPS output:

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

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

Note: In VMware, for enabling 1PPS output over SMA, period file should be configured. Otherwise, input is not valid.

```
SMA1: # echo 1 0 0 0 100 > /sys/class/net/$ETH/device/ptp/ptp*/period
```

```
SMA2: # echo 2 0 0 1 0 > /sys/class/net/$ETH/device/ptp/ptp*/period
```

4.10 Reading Status of the DPLL

The E810-XXVDA4T 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 **dppl_<X>_ref_pin/dppl_<X>_state** option as detailed in [Section 4.12, "Advanced DPLL Configuration"](#).

```
# export ETH=enpls0f0
# 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:

input (idx)	state	priority		ESync fail
		EEC (0)	PPS (1)	
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)	invalid	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 [ps]: -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-XXVDA4T 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 4.12, "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 1 ps unit).

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

Note: CGU information is unavailable when you are using PTP on VMware OS.

4.11 DPLL Monitoring

In the default configuration, the E810-XXVDA4T 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
```

Software 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-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-pruning              : off
legacy-rx                    : off
dpll_monitor                  : on
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
```

Note: With this command you will be able to disable the driver DPLL monitoring however the E810 FW itself does some DPLL configuration and monitoring as well that **cannot** be disabled.

4.12 Advanced DPLL Configuration

4.12.1 pin_cfg User Readable Format

The DPLL on the E810-XXVDA4T offers 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-XXVDA4T 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 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.

A new method of DPLL configuration is available through the Linux Kernel DPLL API, and the usage of this API will be detailed below. See [Section 7.0, “The Linux Kernel DPLL API”](#) for more information.

To check the DPLL pin configuration:

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

```
[root@pae-dbg-r750-02-263 przemek]# cat /sys/devices/pci0000:16/0000:16:04.0/0000:17:00.0/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	1	0	0	255	4		
2	1	invalid	1953125	0	0	8	8		
3	1	invalid	1953125	0	0	9	9		
4	1	invalid	1	7000	0	3	3		
5	1	invalid	1	7000	2	2	2		
6	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	1	1	0	
5	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.

Note: Starting from Firmware version 4.60, the default DPLL configuration has changed. See the end of the section for details.

To check the DPLL pin configuration using [“The Linux Kernel DPLL API”](#).

Set KERNEL point to the directory of the currently installed kernel:

```
# export KERNEL="/home/user/Documents/linux-6.11.7"
```

```
# python3 $KERNEL/tools/net/ynl/cli.py --spec $KERNEL/Documentation/netlink/specs/dpll.yaml --dump pin-get
```

```
[{'board-label': 'CVL-SDP22',
  'capabilities': {'state-can-change', 'priority-can-change'},
  'clock-id': 5799633565435100600,
  'frequency': 1,
  'frequency-supported': [{'frequency-max': 25000000, 'frequency-min': 1}],
  'id': 34,
  'module-name': 'ice',
  'parent-device': [{'direction': 'input',
    'parent-id': 4,
    'phase-offset': 0,
    'prio': 255,
    'state': 'selectable'},
    {'direction': 'input',
    'parent-id': 5,
    'phase-offset': 0,
    'prio': 5,
```

```
        'state': 'selectable' } }],
    'phase-adjust': 0,
    'phase-adjust-max': 16723,
    'phase-adjust-min': -16723,
    'type': 'int-oscillator' },
```

4.12.2 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 [0-6]
```

Note: A *prio value* 255 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
```

4.12.3 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 0-6 (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 SMA'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/$ETH/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 1 Hz 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
```

4.12.4 Modifying frequency, phase-adjust and eSync using "The Linux Kernel DPLL API"

```
# export KERNEL="/home/user/Documents/linux-6.11.7"
# $KERNEL/tools/net/ynl/cli.py --spec $KERNEL/Documentation/netlink/specs/
dpll.yaml --do pin-set --json '{"id":<id>, <config>}'
```

where:

id = ID of pin of interest (See [Section 7.0, "The Linux Kernel DPLL API"](#))

config = list of configuration parameters and values:

```
[ "frequency":<freq_value_in_Hz>,"
  "phase-adjust":<phase_adjust_value_in_ns>"
  "esync-frequency":<0:disable, 1:enable at 1Hz>" ]
```

Note:

- While "frequency-supported" gives upper and lower bounds of available frequencies, not all these frequencies are supported by a particular pin.
- Phase-adjust value is an offset and can be positive or negative.
- esync option only becomes available when pin is at 10 MHz base frequency, and only supports ePPS.

Example:

Set frequency of SMA1 to 10 MHz with a phase delay of 5ns:

```
# export KERNEL="/home/user/Documents/linux-6.11.7"
# $KERNEL/tools/net/ynl/cli.py --spec $KERNEL/Documentation/netlink/specs/
dpll.yaml --do pin-set --json '{"id":6, "frequency":10000000, "phase-
adjust":5000}'
```

Enable embedded 1PPS on SMA1:

```
# export KERNEL="/home/user/Documents/linux-6.11.7"
# $KERNEL/tools/net/ynl/cli.py --spec $KERNEL/Documentation/netlink/specs/
dpll.yaml --do pin-set --json '{"id":6, "esync-frequency":1}'
```

4.12.5 Changing priority, direction, and enabling/disabling pins on "The Linux Kernel DPLL API"

```
# export KERNEL="/home/user/Documents/linux-6.11.7"
# $KERNEL/tools/net/ynl/cli.py --spec $KERNEL/Documentation/netlink/specs/
dpll.yaml --do pin-set --json '{"id":<id>, "<parent>": {"parent-id":<p_id>,
<config>}}'
```

where:

id	=	ID of pin of interest (See "--dump device-get")
parent	=	parent-device or parent-pin (depending on the parent to the pin is a DPLL or another pin)
p_id	=	pins of the parent-ID
config	=	list of configuration parameters and values:
		["prio":<priority_value between 0 & 255>, "direction":"input"/"output", "state":"connected"/"disconnected"/"selectable"]

Note: Direction is not currently configurable for any available pins.

Example:

Changing priority of RCLKA on DPLL 0 to 1:

```
# export KERNEL="/home/user/Documents/linux-6.11.7"
# $KERNEL/tools/net/ynl/cli.py --spec $KERNEL/Documentation/netlink/specs/ dpll.yaml --
do pin-set --json '{"id": 3, "parent-device":{"parent-id":2, "prio":1}}'
```

Changing direction of SMA2 on DPLL 1 to "output":

```
# export KERNEL="/home/user/Documents/linux-6.11.7"
# $KERNEL/tools/net/ynl/cli.py --spec $KERNEL/Documentation/netlink/specs/ dpll.yaml --
do pin-set --json '{"id": 3, "parent-device":{"parent-id":2, "prio":1}}'
```

Disable RCLKB pin:

```
# export KERNEL="/home/user/Documents/linux-6.11.7"
# $KERNEL/tools/net/ynl/cli.py --spec $KERNEL/Documentation/netlink/specs/dpll.yaml --
do pin-set --json '{"id": 4, "parent-device":{"parent-id":2, "state":"disconnected}}'
```

```
# $KERNEL/tools/net/ynl/cli.py --spec $KERNEL/Documentation/netlink/specs/dpll.yaml --
do pin-set --json '{"id": 4, "parent-device":{"parent-id":3, "state":"disconnected}}'
```

4.12.6 Setup Script for version 4.80 and later

Note: Starting from Firmware version 4.60, the default DPLL configuration has changed. It is advised to revert these changes on firmware version 4.80 or later using sysfs (recommended option).

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

```
# export ETH=ens4f0np0
```

Or:

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

2. Run commands:

```
# echo "prio 255 dpll 0 pin 0" > /sys/class/net/$ETH/device/pin_cfg
# echo "prio 255 dpll 0 pin 1" > /sys/class/net/$ETH/device/pin_cfg
# echo "prio 8 dpll 0 pin 2" > /sys/class/net/$ETH/device/pin_cfg
# echo "prio 9 dpll 0 pin 3" > /sys/class/net/$ETH/device/pin_cfg
# echo "prio 3 dpll 0 pin 4" > /sys/class/net/$ETH/device/pin_cfg
# echo "prio 2 dpll 0 pin 5" > /sys/class/net/$ETH/device/pin_cfg
# echo "prio 0 dpll 0 pin 6" > /sys/class/net/$ETH/device/pin_cfg
# echo "prio 5 dpll 1 pin 0" > /sys/class/net/$ETH/device/pin_cfg
# echo "prio 4 dpll 1 pin 1" > /sys/class/net/$ETH/device/pin_cfg
# echo "prio 8 dpll 1 pin 2" > /sys/class/net/$ETH/device/pin_cfg
# echo "prio 9 dpll 1 pin 3" > /sys/class/net/$ETH/device/pin_cfg
# echo "prio 3 dpll 1 pin 4" > /sys/class/net/$ETH/device/pin_cfg
# echo "prio 2 dpll 1 pin 5" > /sys/class/net/$ETH/device/pin_cfg
# echo "prio 0 dpll 1 pin 6" > /sys/class/net/$ETH/device/pin_cfg

# echo "in pin 1 e_ref_sync 2" > /sys/class/net/$ETH/device/pin_cfg
# echo "in pin 5 e_ref_sync 2" > /sys/class/net/$ETH/device/pin_cfg
```

4.12.7 Using DPLL api

Note: Some of the DPLL priority configurations are available through DPLL api, refer to [Section 7.4, "DPLL api Setup Script for version 4.80 and later"](#) for more information.

4.12.8 dpll_<X>_ref_pin/dpll_<X>_state Machine Readable Interface (X = 0 /1)

```
# 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_HOLD OVER = 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 is used by **synce4l** as well.

Note: The user application can monitor the `dp1l_<X>_state` and `dp1l_<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 `dp1l_<X>_name` file.

4.13 1PPS Signals from E810 Device to DPLL

The E810-XXVDA4T 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: Remember to set the channels correctly for SDPs. More information about this can be found in [Section 4.6, "Configuring SDP Pins"](#).

Note: Intel recommends to use a 10 MHz or 1 kHz signal on SDP20 together with the 1 Hz signal from SDP22. For this you will need to configure both the DPLL input pin 1 to the same frequency. See [Section 4.12, "Advanced DPLL Configuration"](#).

1. To enable the frequency reference pin 10 MHz (CVL-SDP20), if assigned to channel 1:

```
# echo 1 0 0 0 100 > /sys/class/net/$ETH/device/ptp/ptp*/period
# echo "in pin 1 freq 10000000" > /sys/class/net/$ETH/device/pin_cfg
```

2. To enable the frequency reference pin 1 KHz (CVL-SDP20), if assigned to channel 1:

```
# echo 1 0 0 0 1000000 > /sys/class/net/$ETH/device/ptp/ptp*/period
# echo "in pin 1 freq 1000" > /sys/class/net/$ETH/device/pin_cfg
```

3. To enable the phase and frequency reference pin with 1 Hz signal (CVL-SDP22), if assigned to channel 2:

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

4. To disable the phase reference pin (CVL-SDP20), if assigned to channel 1:

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

5. To disable the phase and frequency reference pin (CVL-SDP22), if assigned to channel 2:

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

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.

Note: In VMware, for enabling 1PPS input over SMA, `ts2phc` needs to be enabled. Otherwise, input is not valid.

4.14 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.

4.15 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 run:

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

2. Disable UART2 in RAM, and Flash:

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

Or run:

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

The E810-XXVDA4T 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,0.000*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 while **gpsd** is running in the background, and modify the config file of **ts2phc** to use a local port, where **socat** and **gpspipe** provide 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](#)).

4.16 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-XXVDA4T 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>

4.16.1 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` file, ensure that you are using the E810-XXVDA4T 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
(options) # 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` (depends on the kernel) 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 7](#) 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 7. 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-XXVDA4T 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-XXVDA4T 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"
```

4.16.2 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-XXVDA4T.
- 1 - Battery Backed RAM (BBR) — Responsible for saving single configuration items on the E810-XXVDA4T.
- 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-XXVDA4T 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-XXVDA4T 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
```

4.16.3 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)

4.16.4 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
```

4.16.5 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 minimum 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
```


Or run:

```
# echo -ne "\xb5\x62\x06\x8a\x09\x00\x00\x01\x00\x00\xaa\x00\x11\x20\x00\x75\xf6" > /dev/gnssX
```

4.16.9 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 run:

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

5.0 Configuration Setup

The E810-XXVDA4T 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.

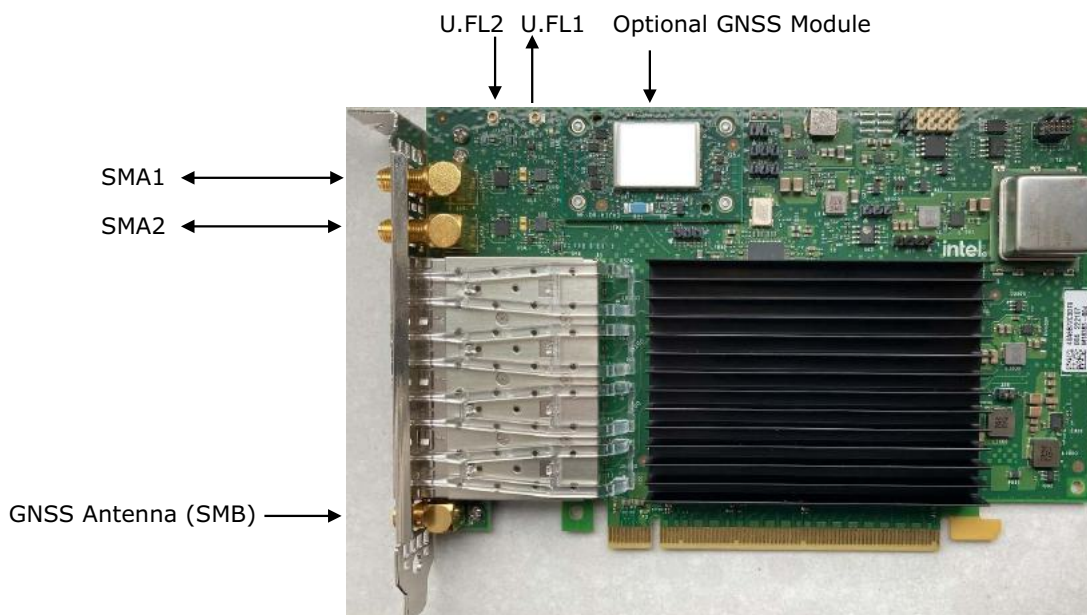


Figure 3. E810-XXVDA4T Connector Locations

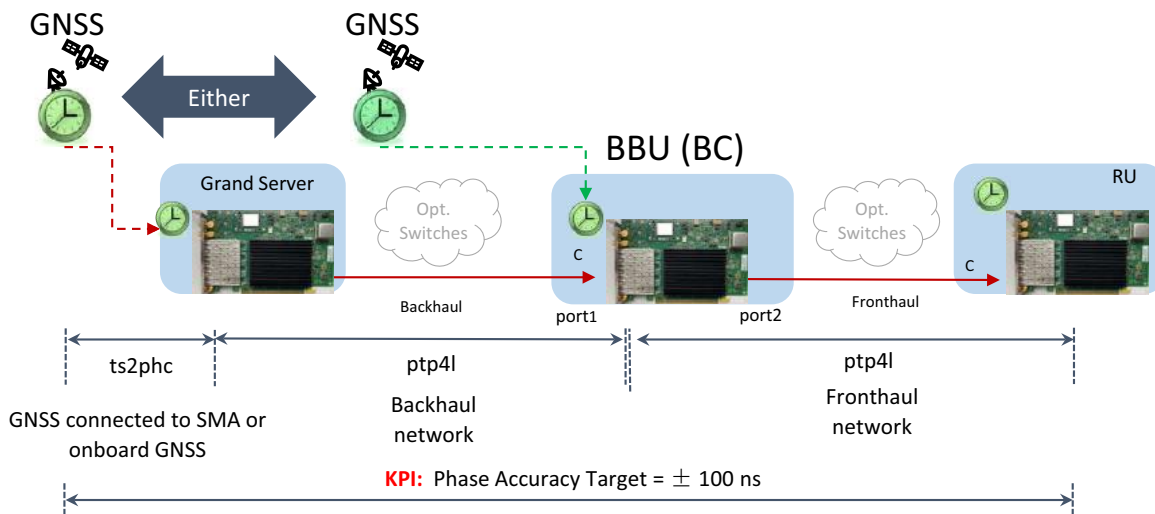


Figure 4. E810-XXVDA4T Connections

5.1 Disable All SMA and U.FL Connections

1. Assign environment variables (optional):

```
# export ETH=ens4f0np0
```

Or:

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

2. Disable U.FL2:

```
# echo 0 > /sys/class/net/$ETH/device/U.FL2
```

3. Disable U.FL1:

```
# echo 0 > /sys/class/net/$ETH/device/U.FL1
```

4. Disable SMA2:

```
# echo 0 > /sys/class/net/$ETH/device/SMA2
```

5. Disable SMA1:

```
# echo 0 > /sys/class/net/$ETH/device/SMA1
```

6. All disabled:

```
# echo 0 > /sys/class/net/$ETH/device/U.FL2
# echo 0 > /sys/class/net/$ETH/device/SMA2
# echo 0 > /sys/class/net/$ETH/device/U.FL1
# echo 0 > /sys/class/net/$ETH/device/SMA1
```

7. Example check of U.FL1 using the **cat command**:

```
#cat /sys/class/net/$ETH/device/U.FL1
0
```

5.2 PTP GrandMaster (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

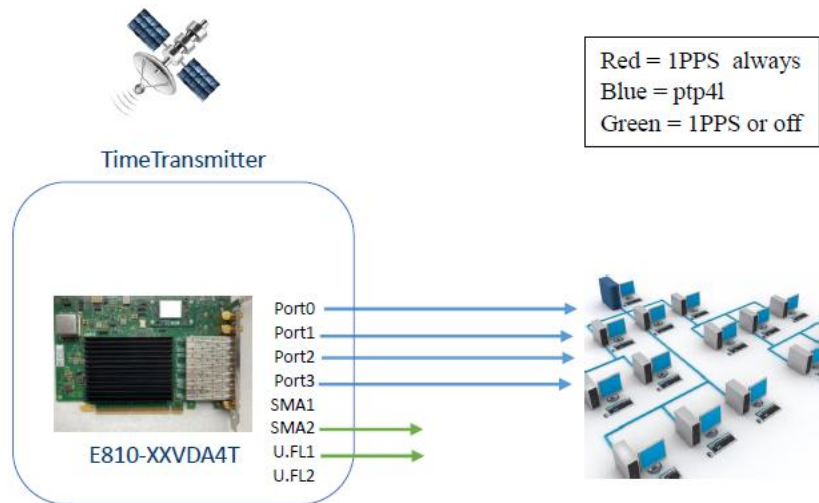


Figure 5. External Connections: PTP GrandMaster with Optional GNSS Module

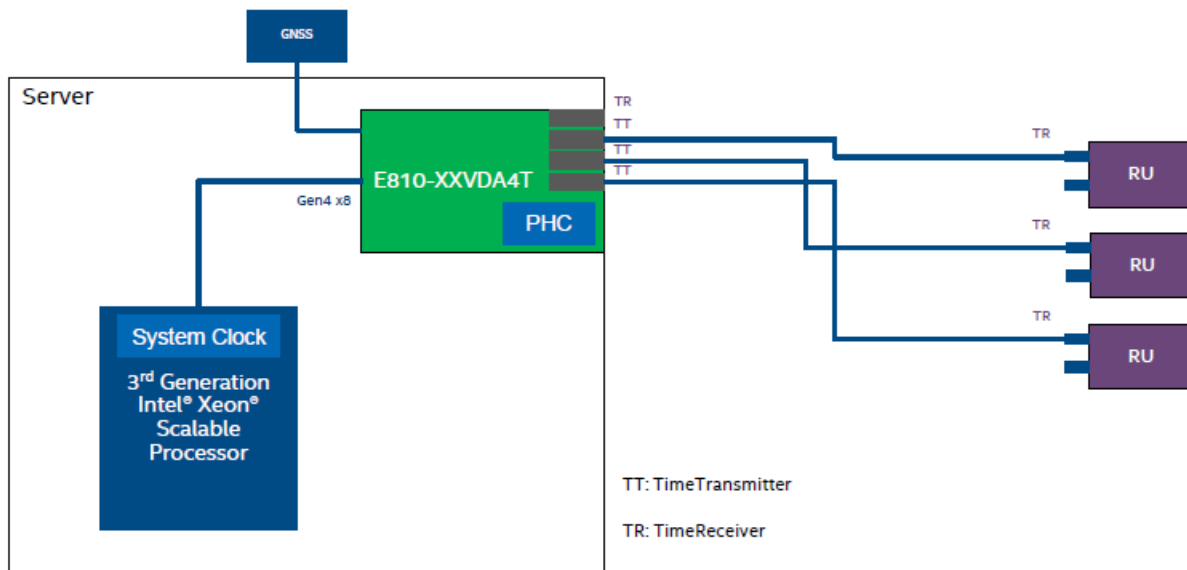


Figure 6. External Connections: Single E810-XXVDA4T Adapter Configuration with GNSS

5.2.2 Software Configuration

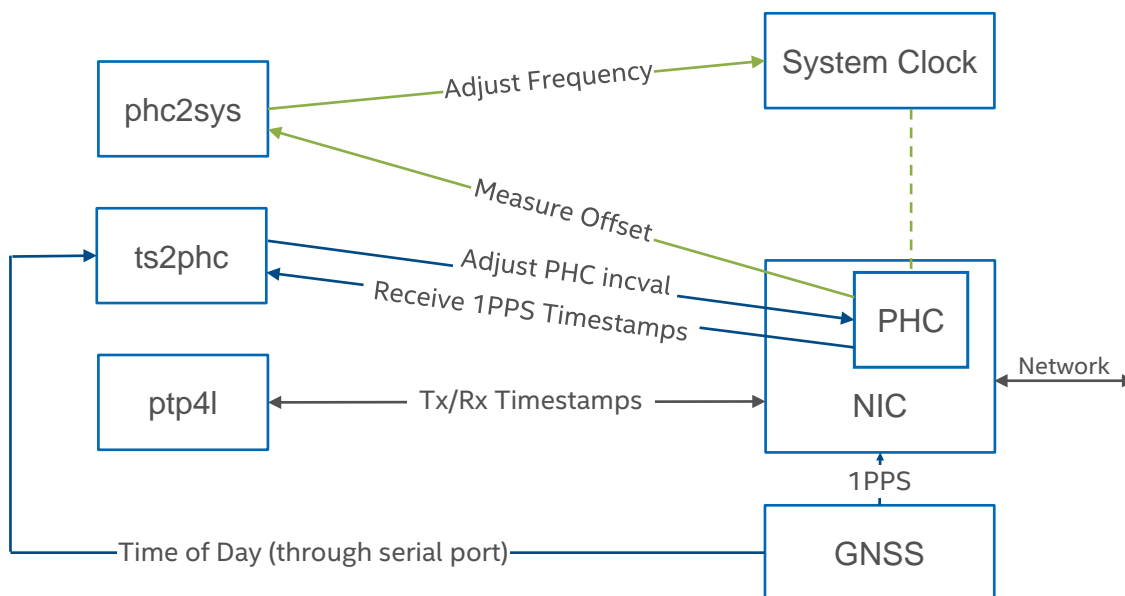


Figure 7. Linux Software Stack Overview: Single E810-XXVDA4T Adapter with GNSS Software Stack

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 000e /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**:

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

Note: You might want filter the 1PPS timestamps till the DPLL locked. See [Section 4.14](#).

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 (1,2 and 3 as well if required):

```
# ./ptp4l -f config.cfg -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".

5.3 PTP GrandMaster (GM) with External GNSS Clock

5.3.1 External Connections

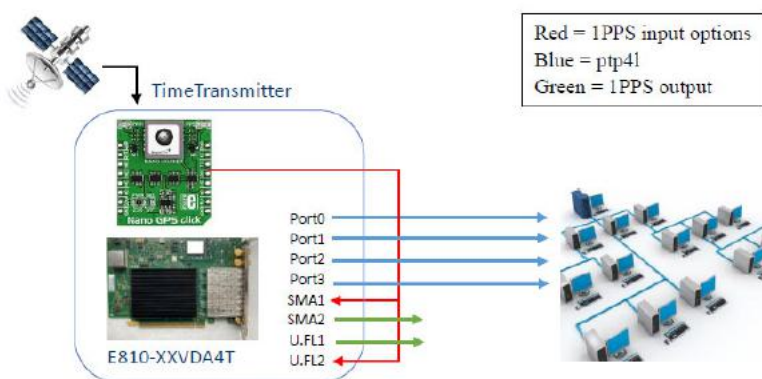


Figure 8. External Connections: PTP GrandMaster with External GNSS Clock

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 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)
# export ETH2=`grep 000e /sys/class/net/*/device/subsystem_device | awk -F"/" '{print $5}' | head -n 3 | tail -n +3` #(port2)
```

```
# export ETH3=`grep 000e /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 > /sys/class/net/$ETH/device/SMA1
# echo 2 > /sys/class/net/$ETH/device/SMA2
```

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

```
# echo 1 > /sys/class/net/$ETH/device/U.FL2
# echo 2 > /sys/class/net/$ETH/device/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
[1769.203160] ice 0000:17:00.0: SMA1 set to Rx. U.FL1 automatically enabled as Tx.
[1773.718752] ice 0000:17:00.0: SMA2 set to Tx. U.FL2 automatically enabled as Rx.
[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
```

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 may want to filter the 1PPS timestamps until the DPLL is locked. See [Section 4.14](#).

5. Run **ptp4l**:

Running on Port 0 (1,2 and 3 as well if required):

```
# ./ptp4l -f config.cfg -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"](#).

5.4 Boundary Clock Configuration

5.4.1 External Connections

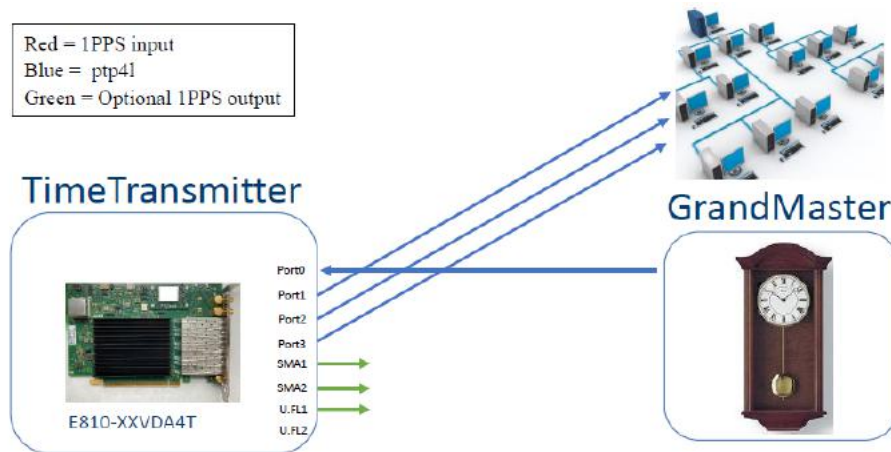


Figure 9. External Connections: Boundary Clock Configuration

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-XXVDA4T.

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

2. Set periodic output on SDP20 to 10 MHz (to synchronize the DPLL1 to the E810 PHC synced by **ptp4l**, assuming SDP20 is assigned to channel 1):

```
# 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**, assuming SDP22 is assigned to channel 2):

```
# 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 > /sys/class/net/$ETH/device/SMA1
```

Or:

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

```
# echo 2 > /sys/class/net/$ETH/device/U.FL1
```

- Enable SMA2 output: (for visibility):

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

Note: If U.FL1 is 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
[2042.312662] ice 0000:17:00.0: SMA1 set to Tx. U.FL1 automatically disabled.
[1773.718752] ice 0000:17:00.0: SMA2 set to Tx. U.FL2 automatically enabled as Rx.
[27526.852127] ice 0000:17:00.0: <DPLL1> state changed to: unlocked, pin CVL-SDP20
[30243.385109] ice 0000:17:00.0: <DPLL1> state changed to: locked, pin CVL-SDP20
```

Or:

```
#dmesg
[2173.090590] ice 0000:17:00.0: U.FL1 set to Tx. SMA1 automatically enabled as Rx.
[1773.718752] ice 0000:17:00.0: SMA2 set to Tx. U.FL2 automatically enabled as Rx.
[27611.824727] ice 0000:17:00.0: <DPLL1> state changed to: unlocked, pin CVL-SDP20
[30243.385109] ice 0000:17:00.0: <DPLL1> state changed to: locked, pin CVL-SDP20
```

5. Run **ptp4l**:

```
# ./ptp4l -i $ETH -i $ETH1 -i $ETH2 -i $ETH3 -m
```

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 TimeReceiver

The E810-XXVDA4T 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

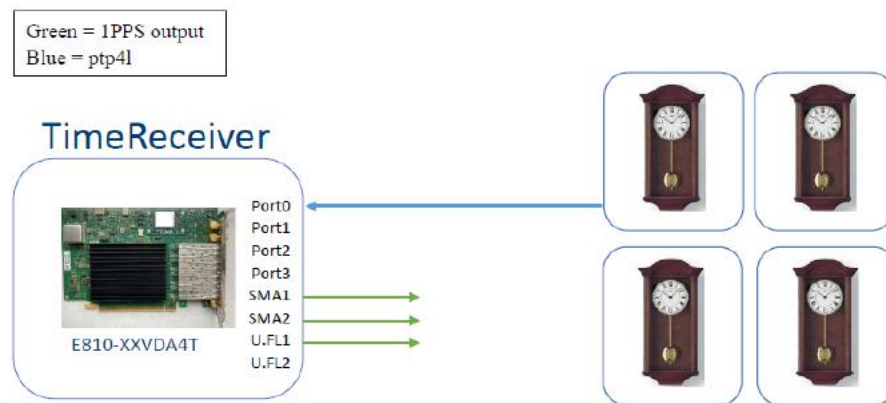


Figure 10. External Connections: Port Configured as TimeReceiver

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 000e /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**, assuming SDP20 is assigned to channel 1):

```
# 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**, assuming SDP22 is assigned to channel 2):

```
# 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 > /sys/class/net/$ETH/device/SMA1
```

Or:

- Enable U.FL1 output (for visibility):

```
# echo 2 > /sys/class/net/$ETH/device/U.FL1
```

- Enable SMA2 output (for visibility):

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

Note: If U.FL1 is 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
[ 836.312662] ice 0000:17:00.0: SMA1 set to Tx. U.FL1 automatically 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
[ 1769.203160] ice 0000:17:00.0: SMA1 set to Rx. U.FL1 automatically enabled as Tx.
[ 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
[ 835.718752] ice 0000:17:00.0: SMA2 set to Tx. U.FL2 automatically enabled as Rx.
[ 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 -f config.cfg -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 G.8262 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 sync4l ([Section 5.6.3](#)).

5.6.1 External Connections

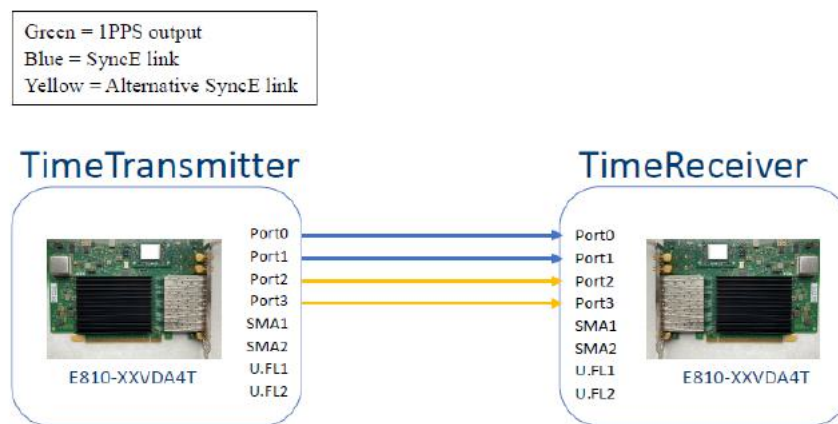


Figure 11. External Connections: SyncE Setup

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

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 TimeTransmitter or TimeReceiver is acceptable. If both are set, users might run into sync loop issues.

5.6.3 ITU G.8264 ESMC Messaging Using sync4l

The sync4l application is implementing the G.8264 ESMC protocol.

1. Run **sync4l**:

```
# ./sync4l -f configs/sync4l.cfg -l 7 -m
```

Note: Refer to [Section 5.10](#), "Example sync4l Configuration File for BC".

5.6.4 Two E810-XXVDA4T Adapter Configuration without GNSS

Figure 12 shows two E810-XXVDA4T adapters in a system with a cell site router.

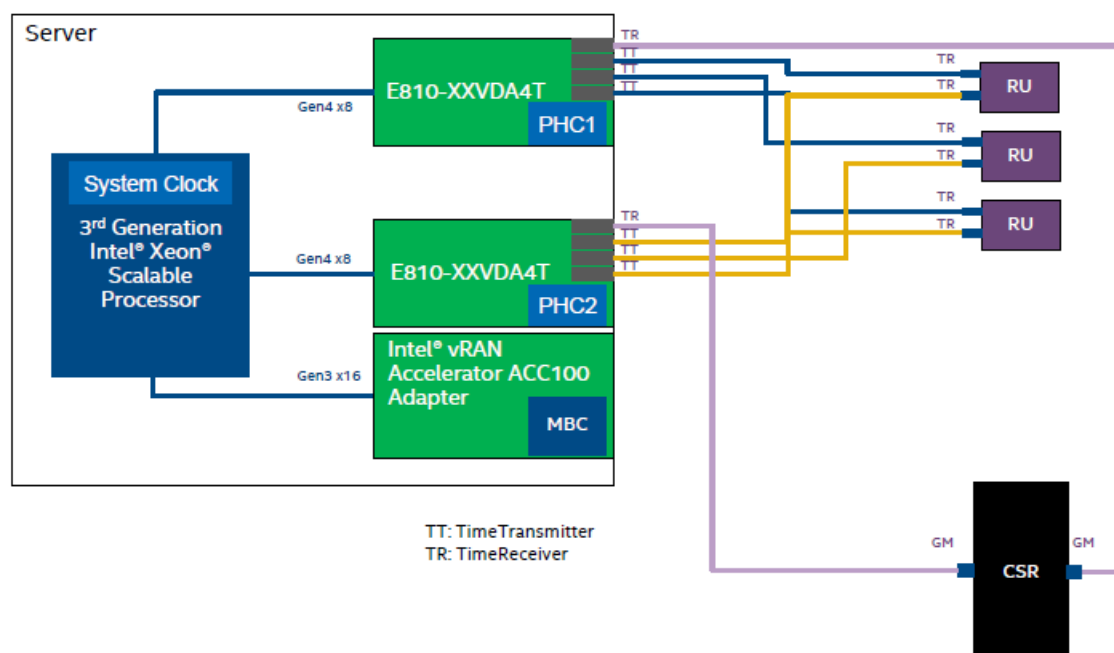


Figure 12. External Connections: Two E810-XXVDA4T Adapters without GNSS

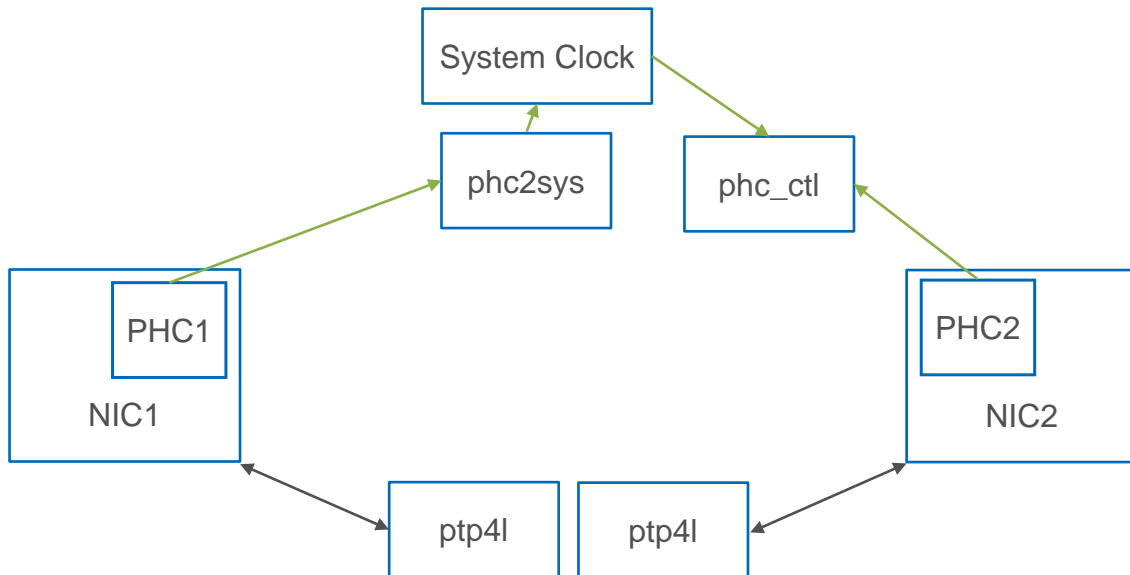


Figure 13. External Connections: Two E810-XXVDA4T Adapters without GNSS Linux Software Stack

Adapter 1 Linux software stack:

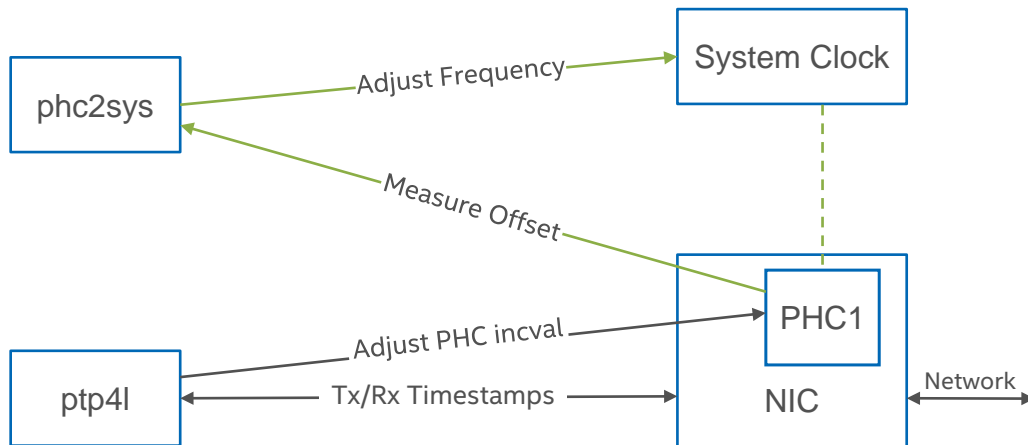


Figure 14. External Connections: Two E810-XXVDA4T Adapters without GNSS Adapter 1 Software Stack

Adapter 1 configuration:

1. Run one instance of **ptp4l** per E810-XXVDA4T:

```
# 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:

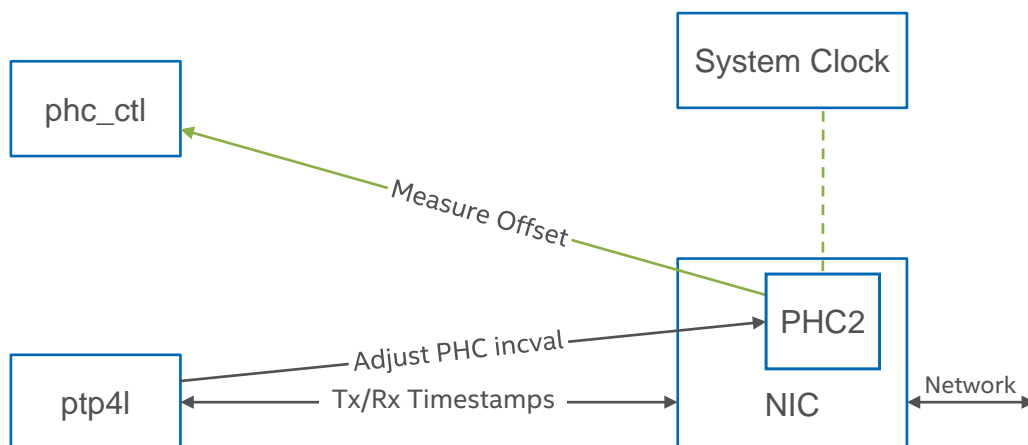


Figure 15. External Connections: Two E810-XXVDA4T Adapters without GNSS Adapter 2 Software Stack

Adapter 2 configuration:

1. Run one instance of **ptp4l** per E810-XXVDA4T:

```
# 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-XXVDA4T Adapter Configuration without GNSS and with 1PPS

Figure 16 shows two E810-XXVDA4T adapters in a system with a 1PPS connection between them.

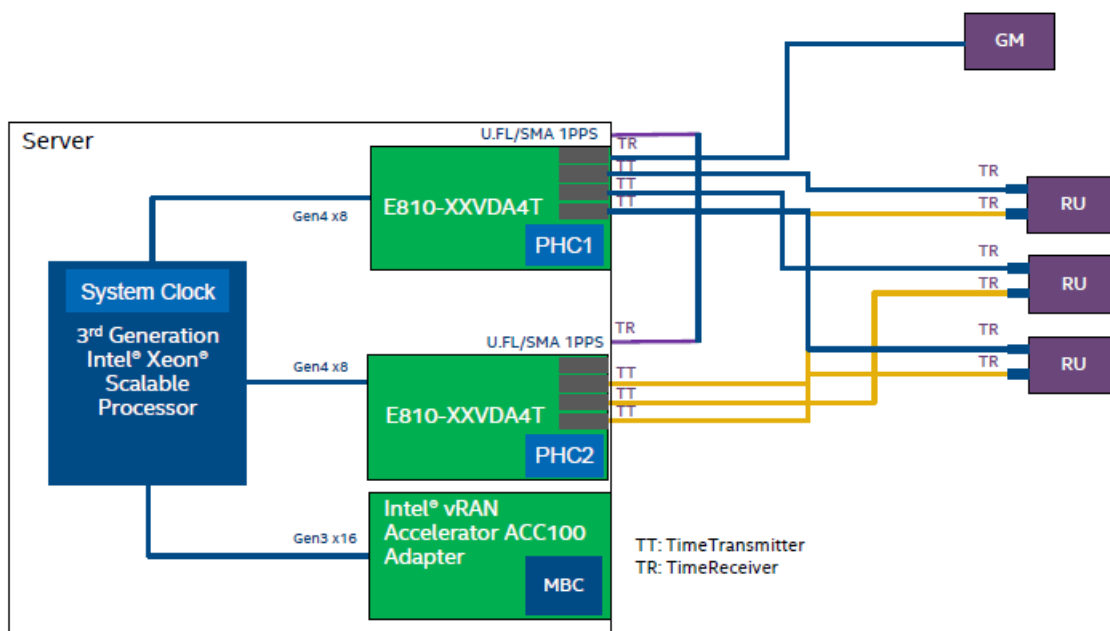


Figure 16. External Connections: Two E810-XXVDA4T Adapter Configuration (no GNSS)

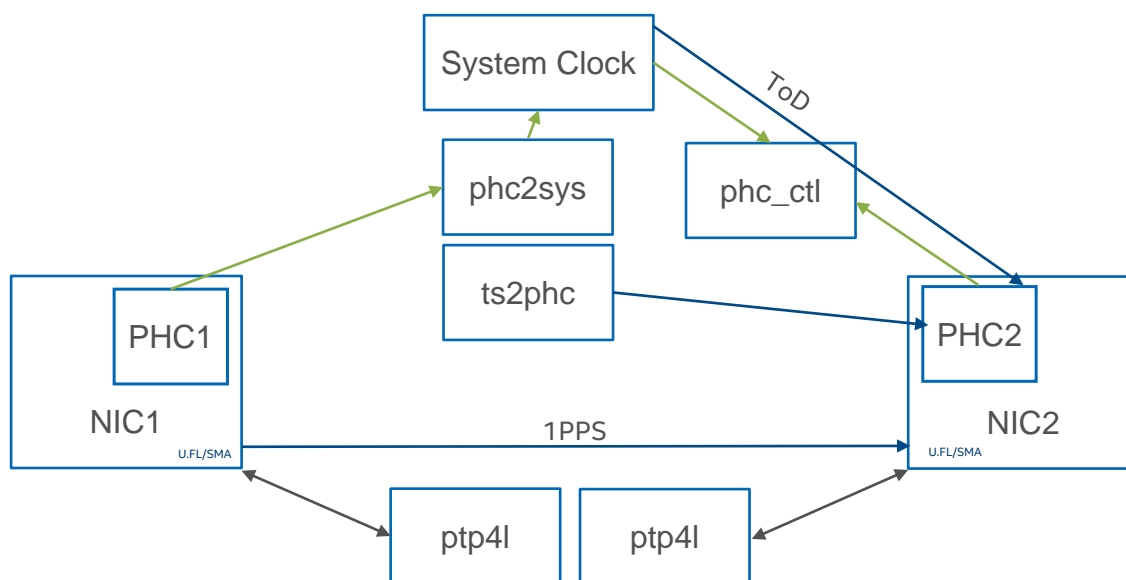


Figure 17. External Connections: Two E810-XXVDA4T Adapter Configuration Linux Software Stack (no GNSS)

Adapter 1 Linux software stack:

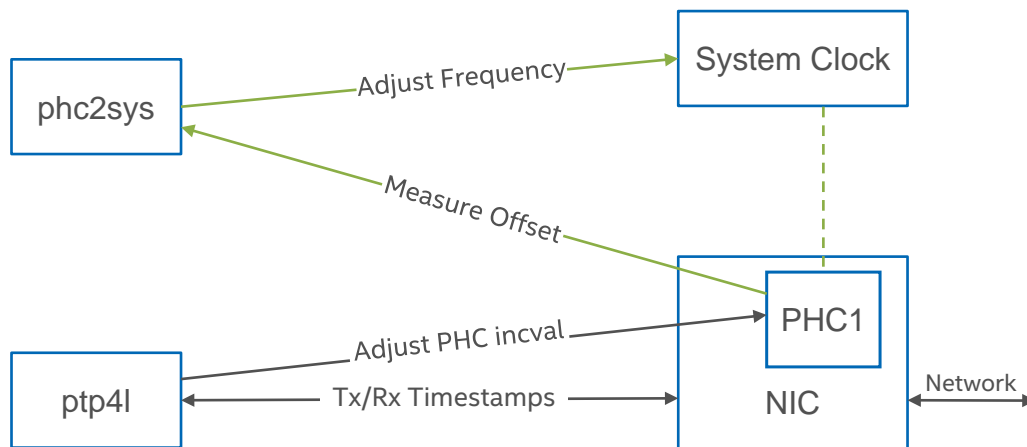


Figure 18. Linux Software Stack Overview: Adapter 1 (no GNSS)

Adapter 1 configuration:

1. Enable 1PPS output on U.FL1:

```
# echo 2 > /sys/class/net/$ETH/device/U.FL1
```

or SMA 1:

```
# echo 2 > /sys/class/net/$ETH/device/SMA1
```

2. Enable SDP20/SDP22

Set periodic output on SDP20 to 10 MHz (to synchronize the DPLL1 to the E810 PHC synced by **ptp4l**), assuming SDP20 is assigned to channel 1:

```
# 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**), assuming SDP22 is assigned to channel 2:

```
# 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:

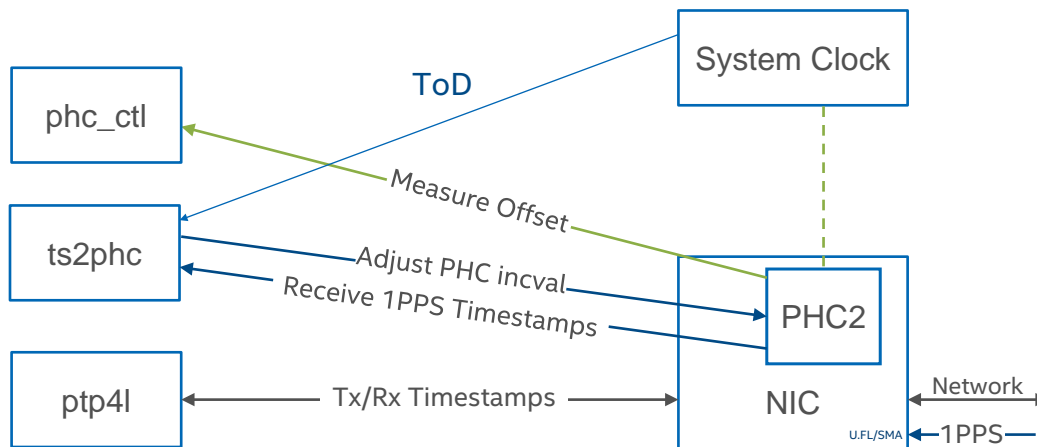


Figure 19. Linux Software Stack Overview: Adapter 2 (no GNSS)

Adapter 2 configuration:

1. Enable 1PPS input on U.FL2:

```
# echo 1 > /sys/class/net/$ETH/device/U.FL2
```

or, enable 1PPS input on SMA2:

```
# echo 1 > /sys/class/net/$ETH/device/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-XXVDA4T Adapters with GNSS Connection Setup

Figure 20 shows two E810-XXVDA4T adapters in a system with a GNSS connection.

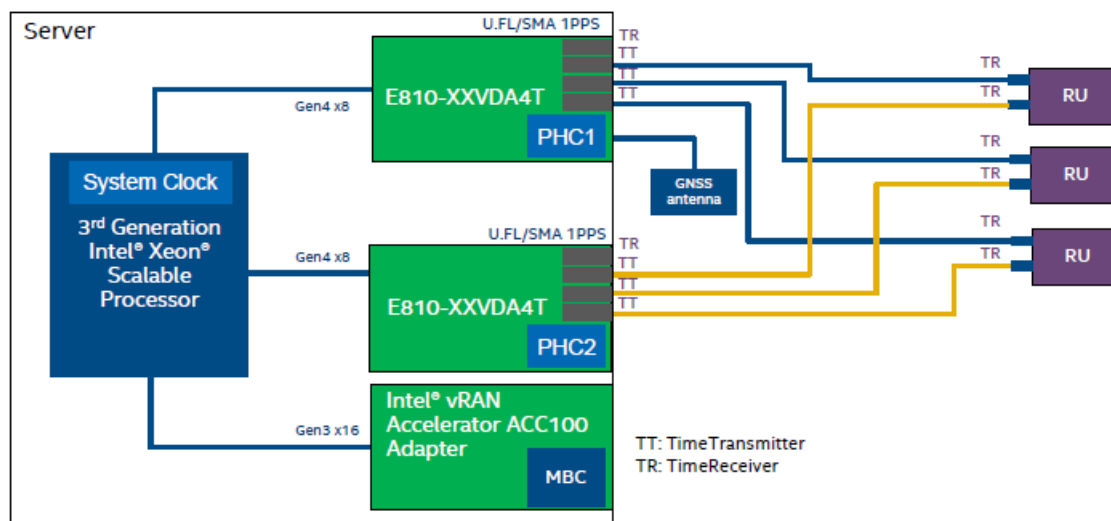


Figure 20. External Connections: Two E810-XXVDA4T Adapter Configuration (with GNSS)

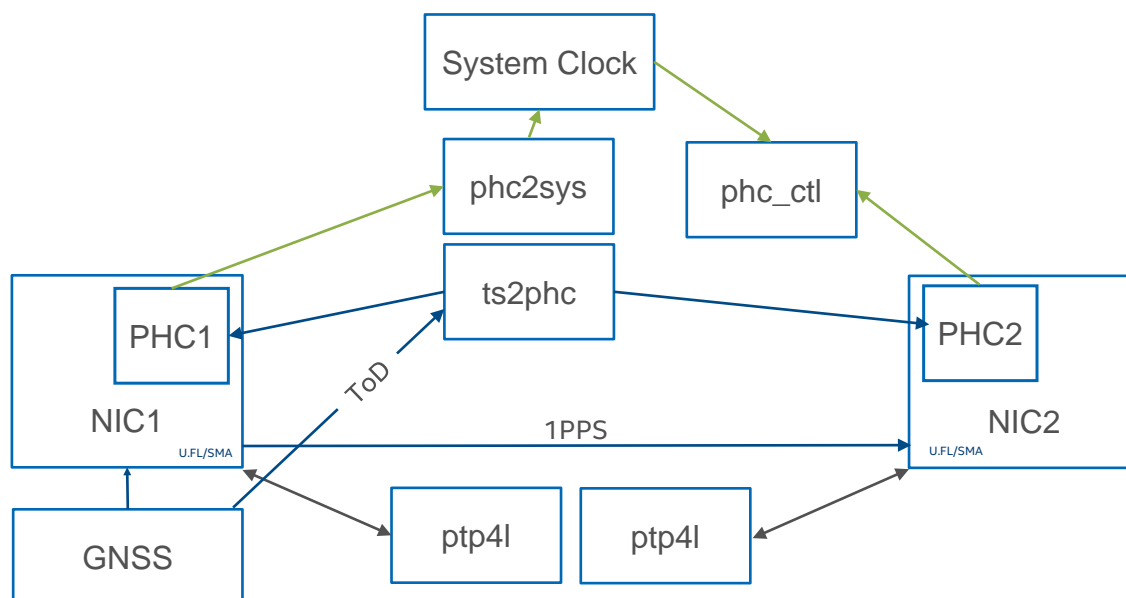


Figure 21. External Connections: Two E810-XXVDA4T Adapter Configuration Linux Software Stack (with GNSS)

Adapter 1 Linux software stack:

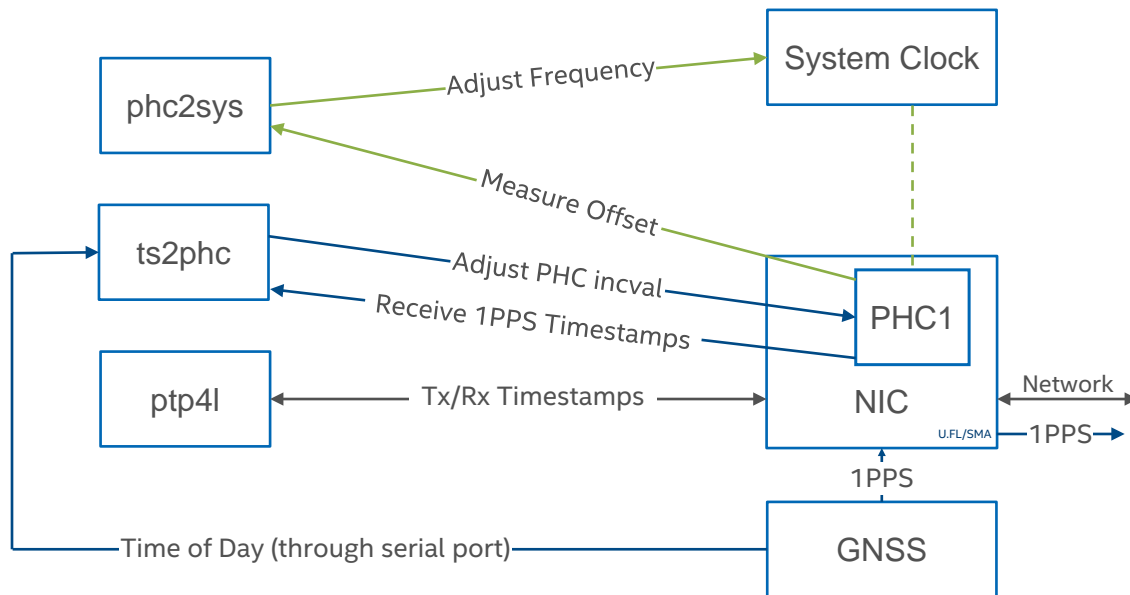


Figure 22. Linux Software Stack Overview: Adapter 1 (with GNSS)

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 > /sys/class/net/$ETH/device/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:

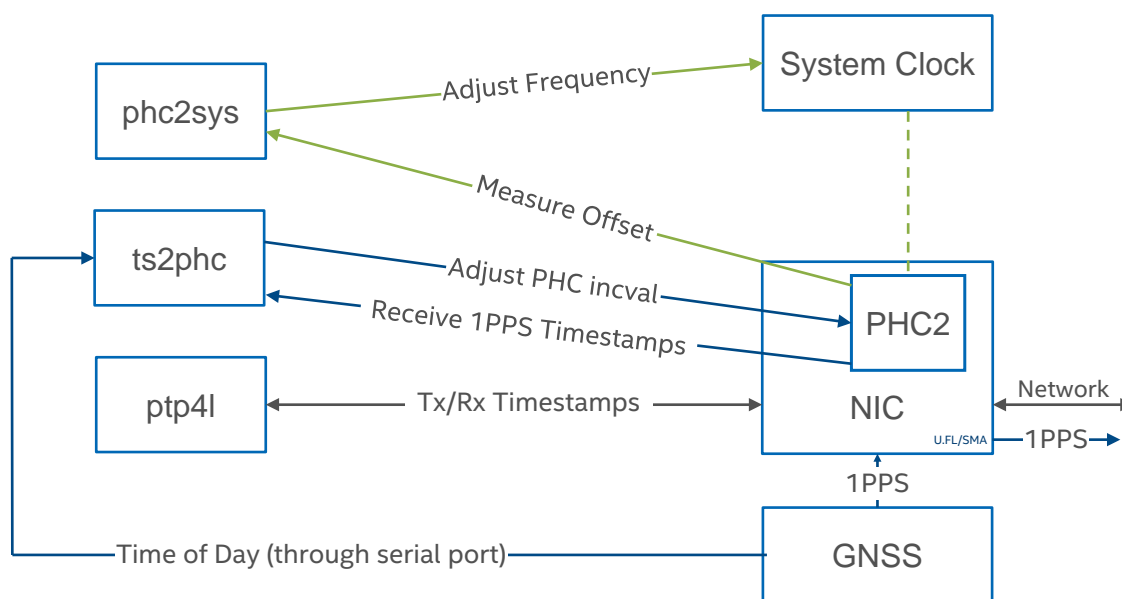


Figure 23. Linux Software Stack Overview: Adapter 2 (with GNSS)

Adapter 2 configuration:

1. Enable 1PPS in on U.FL2:

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

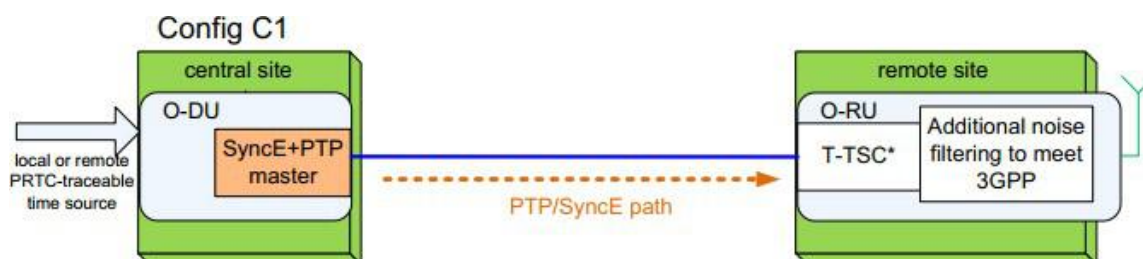


Figure 24. External Connections: O-RAN Configuration 1

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

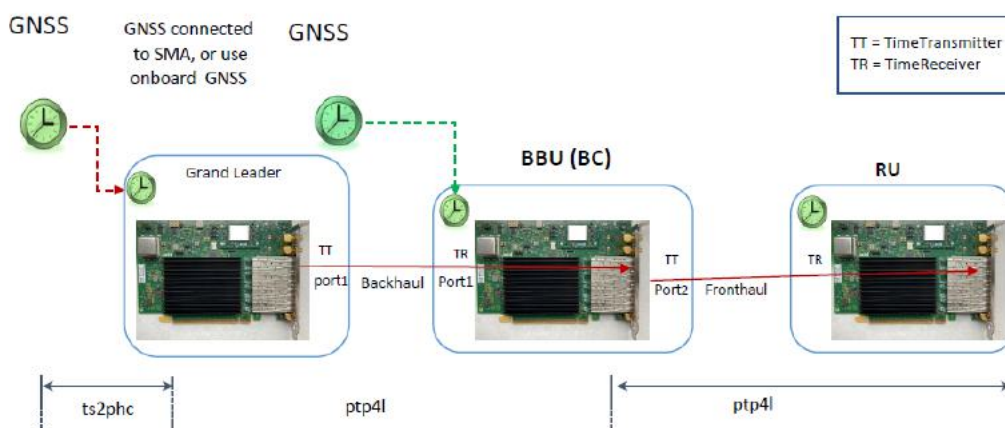


Figure 25. O-RAN Configuration 1 Connections

5.7.2 Software Configuration

- GrandMaster 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
[enpls0f0(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
clientOnly           0
priority1            129
priority2            255 #lower value = higher priority to become TimeTransmitter
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      0
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
sanity_freq_limit          200000000
#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

```

Note: For One-step TimeReceiver mode, please configure twoStepFlag "1" (enabled) and assume_two_step "0" disabled. The One-step operation in TimeTransmitter mode is not supported.

5.10 Example synce4l Configuration File for BC

```
# Global section is for debugging mostly
[global]
#
# Runtime options
#
logging_level      7
use_syslog         0
verbose           1
message_tag        [synce4l]
smc_socket_path    /tmp/synce4l_socket

#
# Device section
# Per-device configuration
#
# User defined name of a one logical device configured for SyncE in the system.
#
# Clock sources for synchronization can be either external clock sources on
# the device or inputs recovered from the PHY's.
# The QL of the external clock sources is configured in the external sources
# sections [{<clk src name>}], under the device.
# 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.
#
# If QL of external source clock is the best QL in the system, it shall be used
# to feed its frequency to all ports.
# If QL of clock recovered from port is the best QL in the system, its frequency
# recovered on that port shall be used to feed its frequency to all the other ports.
#
# All the external clock sources and ports configured after this section will
# be a part of this device (until next device section).
[<syncel>]

#
# 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:
#     - The TX version of TLV will be propagated from the port or external
#       clock source 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/ens785f0np0/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.
#
[ens785f0np0]

#
# 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/ens785f0np0/device/phy/synce
recover_clock_disable_cmd   echo 0 0 > /sys/class/net/ens785f0np0/device/phy/synce

#
# next configured interface for the device
#
[ens785f1np1]
tx_heartbeat_msec          1000
rx_heartbeat_msec          500
recover_clock_enable_cmd    echo 1 0 > /sys/class/net/ens785f1np1/device/phy/synce
recover_clock_disable_cmd   echo 0 0 > /sys/class/net/ens785f1np1/device/phy/synce
[ens785f3np3]
tx_heartbeat_msec          1000
rx_heartbeat_msec          500
recover_clock_enable_cmd    echo 1 0 > /sys/class/net/ens785f3np3/device/phy/synce
recover_clock_disable_cmd   echo 0 0 > /sys/class/net/ens785f3np3/device/phy/synce
#
# External source section(s)
#
# It starts per-external clock source configuration.
# External clock sources are either 1PPS from built-in GPS module or 1PPS
# from the on-board SMA connectors.
# Each port (of the device) that is used for SyncE, shall have its own section.
#
#{SMA1}}

# These values are sent to the peers on configured ports when this source clock
# selected as best clock source.
# 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.
#
# input_QL corresponds to the SSM code column.
#
# input_ext_QL corresponds to the Enhanced SSM code column
# (is used only if "extended_tlv = 1"). For example, for QL of 0x02,
# corresponding values of external QL will be 0x20, 0x21, and 0xFF.
#
#input_QL          0x2
#input_ext_QL      0xFF

#
# Shell commands for enabling/disabling this external clock source on a device
```

```
#
#external_enable_cmd      echo 2 > /sys/class/net/ens785f3np3/device/SMA1
#external_disable_cmd     echo 0 > /sys/class/net/ens785f3np3/device/SMA1

#
# An internal source priority, higher priority (lower value) is used to select
# a source as better quality in case two sources are considered equal quality
# [0-255], default:128
#
#internal_prio  0

#
# next configured external clock source for the device
#
#{SMA2}
#input_QL      0x2
#input_ext_QL   0xFF
#external_enable_cmd      echo 2 > /sys/class/net/ens785f3np3/device/SMA2
#external_disable_cmd     echo 0 > /sys/class/net/ens785f3np3/device/SMA2
#{GNSS}
#input_QL      0x2
#input_ext_QL   0xFF
#internal_prio  255
#external_enable_cmd echo ""
#external_disable_cmd echo ""
##
#####
#
# next SyncE device section
#
#<sync2>

#
# new port belonging to the "new" device
#
#<enp7s0f0>
```


6.0 Initial Test Setup

Two E810-XXVDA4T adapters on two systems with a Trimble GM200 used as a timeserver.

6.1 Test Diagram

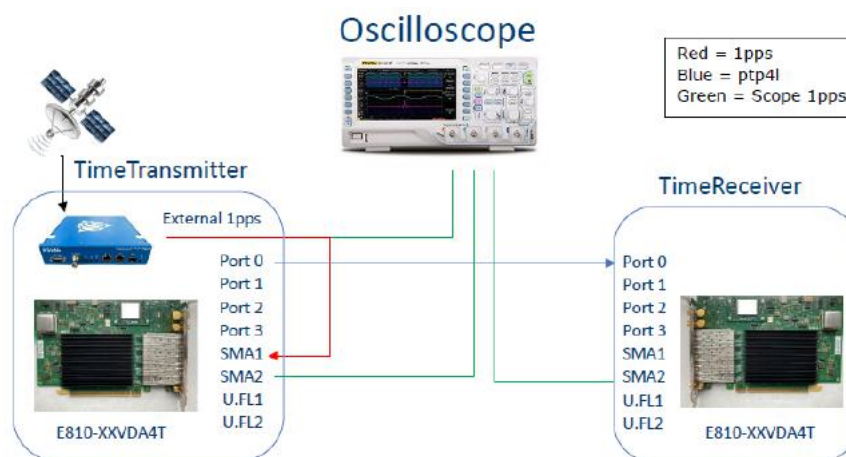


Figure 26. Test Setup

6.2 Software Configuration

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

6.2.1 TimeTransmitter 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 000e /sys/class/net/*/device/subsystem_device | awk -F"/" '{print $5}' | head -n 1`
```

2. SMA1 as input and SMA2 as output:

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

Note: As a side effect of having a very accurate DPLL, synchronization between the two E810-XXVDA4T 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**:

```
# ./ptp4l -i <network_interface_port0> -m -s
# ./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: As the default DPLL priority for 1PPS from the SMA is higher than the recover clock's priority.

6.2.2 TimeReceiver 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 000e /sys/class/net/*/device/subsystem_device | awk -F"/" '{print $5}' | head -n 1`
```

2. SMA2 as output:

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

3. Set periodic output on SDP20 and SDP22 (To synchronize the DPLL1 to the E810 PHC synced by **ptp4l**), assuming they are assigned to channels 1 and 2 respectively:

```
# 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-XXVDA4T 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: As the default DPLL priority for 1PPS from the SMA is higher than the recover clock's priority.

6.2.3 Test Results

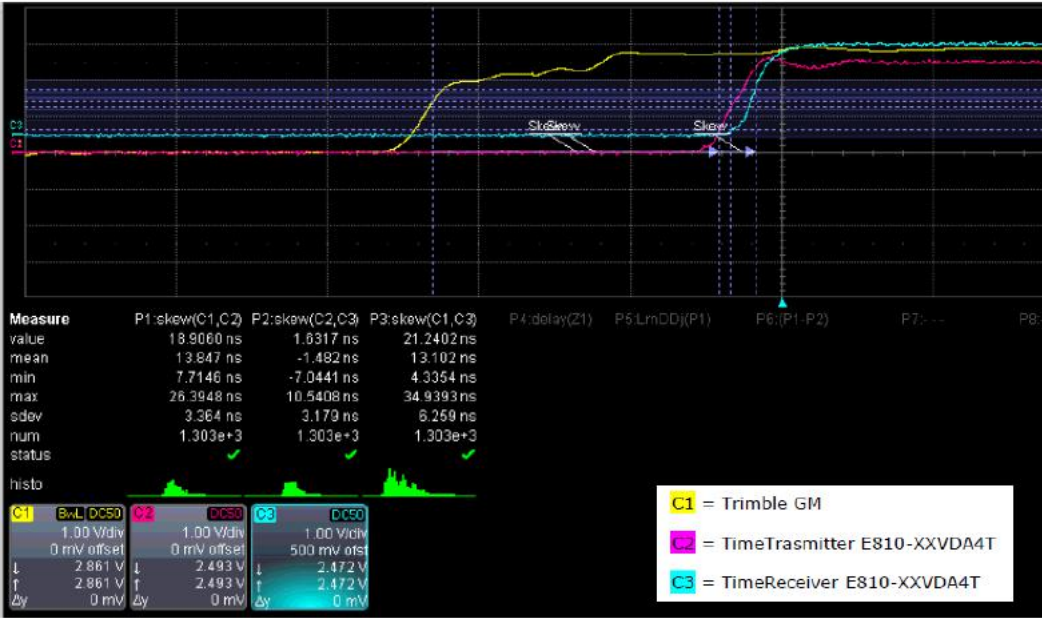


Figure 27. Test Results

7.0 The Linux Kernel DPLL API

7.1 Overview

The Linux Kernel DPLL API is an alternative method to configure DPLLs and DPLL. Users should ensure that their Linux Kernel version is 6.7 or later (but 6.11 or later is recommended), and Python 3.9 or later are installed. Additional dependencies required for usage are:

- Jschema version 4.*
- PyYAML version 6.*

A requirements.txt file is included in the main script's directory.

For reference, one Intel E810-XXVDA4T Ethernet Network Adapter is used in all example outputs unless otherwise specified.

This API uses a general purpose YNL utility to encode/decode netlink messages to and from the driver. It bases these messages on a YAML specification that is passed in from the Kernel files. To execute any commands, the YNL utility must be invoked with the YAML specification as a parameter before specifying any specific commands.

The YNL module can be found under:

```
<path_to_installed_kernel>/tools/net/ynl/cli.py
```

The YAML specification can be found under:

```
<path_to_installed_kernel>/Documentation/netlink/specs/dpll.yaml
```

Therefore, assuming that your current directory is the currently installed Kernel, to invoke the API, this is the command structure to be used:

```
./tools/net/ynl/cli.py --spec Documentation/netlink/specs/dpll.yaml <command>
```

7.2 Commands

7.2.1 --dump device-get

To retrieve configuration information for all DPLLs installed in the system, the command --dump device-get can be used:

```
# ./tools/net/ynl/cli.py --spec Documentation/netlink/specs/dpll.yaml --dump device-get
[{'clock-id': 5799633565435100600,
  'id': 4,
  'lock-status': 'unlocked',
  'mode': 'automatic',
  'mode-supported': ['automatic'],
  'module-name': 'ice',
  'type': 'eec'},
 {'clock-id': 5799633565435100600,
  'id': 5,
  'lock-status': 'unlocked',
  'mode': 'automatic',
  'mode-supported': ['automatic'],
  'module-name': 'ice',
  'type': 'pps'}]
```

This output describes two DPLLs of type EEC (Ethernet Equipment Clock) and PPS (Pulse Per Second). The following are the various fields explained:

Field	Description
clock-id	By default, the decimalised PCI serial number is used. With Intel Ethernet cards, this is typically the decimalised MAC address of the device, but this is not guaranteed.
id	Unique, non-persistent ID for all DPLLs and DPLL pins in the system are used for message addressing. These IDs can change after a driver reload or system reboot.
lock-status	If the DPLL has achieved phase lock with a signal.
mode [Manual/Automatic]	Determines whether the DPLL will lock onto a signal using a priority system or manual control.
mode-supported [Manual/Automatic]	The modes supported by the DPLL.
module- name	Name of the driver that communicates with the DPLLs.
type	Type of DPLL (EEC, PPS).

7.2.2 --dump pin-get

To retrieve configuration information for all DPLL pins in the system, the command `--dump pin-get` can be used:

```
# ./tools/net/ynl/cli.py --spec ./Documentation/netlink/specs/dpll.yaml --dump pin-get
[{'board-label': 'CVL-SDP22',
  'capabilities': {'state-can-change', 'priority-can-change'},
  'clock-id': 5799633565435100600,
  'frequency': 1,
  'frequency-supported': [{'frequency-max': 25000000, 'frequency-min': 1}],
  'id': 34,
  'module-name': 'ice',
  'parent-device': [{'direction': 'input',
    'parent-id': 4,
    'phase-offset': 0,
    'prio': 255,
    'state': 'selectable'},
    {'direction': 'input',
    'parent-id': 5,
    'phase-offset': 0,
    'prio': 5,
    'state': 'selectable'}],
  'phase-adjust': 0,
  'phase-adjust-max': 16723,
  'phase-adjust-min': -16723,
  'type': 'int-oscillator'},
 {'board-label': 'CVL-SDP20',
  'capabilities': {'state-can-change', 'priority-can-change'},
  'clock-id': 5799633565435100600,
  'frequency': 1,
  'frequency-supported': [{'frequency-max': 25000000, 'frequency-min': 1}],
  'id': 35,
```

```

'module-name': 'ice',
'parent-device': [{ 'direction': 'input',
                    'parent-id': 4,
                    'phase-offset': 0,
                    'prio': 255,
                    'state': 'selectable' },
                  { 'direction': 'input',
                    'parent-id': 5,
                    'phase-offset': 0,
                    'prio': 4,
                    'state': 'selectable' } ],
'phase-adjust': 0,
'phase-adjust-max': 16723,
'phase-adjust-min': -16723,
'type': 'int-oscillator',
{'board-label': 'C827_0-RCLKA',
 'capabilities': {'state-can-change', 'priority-can-change'},
 'clock-id': 5799633565435100600,
 'frequency': 1953125,
 'frequency-supported': [{ 'frequency-max': 25000000, 'frequency-min': 1 } ],
 'id': 36,
 'module-name': 'ice',
 'parent-device': [{ 'direction': 'input',
                    'parent-id': 4,
                    'phase-offset': 0,
                    'prio': 8,
                    'state': 'selectable' },
                  { 'direction': 'input',
                    'parent-id': 5,
                    'phase-offset': 0,
                    'prio': 8,
                    'state': 'selectable' } ],
 'phase-adjust': 0,
 'phase-adjust-max': 16723,
 'phase-adjust-min': -16723,
 'type': 'mux' },
{'board-label': 'C827_0-RCLKB',
 'capabilities': {'state-can-change', 'priority-can-change'},
 'clock-id': 5799633565435100600,
 'frequency': 1953125,
 'frequency-supported': [{ 'frequency-max': 25000000, 'frequency-min': 1 } ],
 'id': 37,
 'module-name': 'ice',
 'parent-device': [{ 'direction': 'input',
                    'parent-id': 4,
                    'phase-offset': 0,
                    'prio': 9,
                    'state': 'selectable' },
                  { 'direction': 'input',

```

```
        'parent-id': 5,
        'phase-offset': 0,
        'prio': 9,
        'state': 'selectable'}],
    'phase-adjust': 0,
    'phase-adjust-max': 16723,
    'phase-adjust-min': -16723,
    'type': 'mux'},
{'board-label': 'SMA1',
 'capabilities': {'state-can-change', 'priority-can-change'},
 'clock-id': 5799633565435100600,
 'frequency': 1,
 'frequency-supported': [{'frequency-max': 25000000, 'frequency-min': 1}],
 'id': 38,
 'module-name': 'ice',
 'parent-device': [{ 'direction': 'input',
                     'parent-id': 4,
                     'phase-offset': 0,
                     'prio': 3,
                     'state': 'selectable'},
                   { 'direction': 'input',
                     'parent-id': 5,
                     'phase-offset': 0,
                     'prio': 3,
                     'state': 'selectable'}],
 'phase-adjust': 0,
 'phase-adjust-max': 16723,
 'phase-adjust-min': -16723,
 'type': 'ext'},
{'board-label': 'SMA2/U.FL2',
 'capabilities': {'state-can-change', 'priority-can-change'},
 'clock-id': 5799633565435100600,
 'frequency': 1,
 'frequency-supported': [{'frequency-max': 25000000, 'frequency-min': 1}],
 'id': 39,
 'module-name': 'ice',
 'parent-device': [{ 'direction': 'input',
                     'parent-id': 4,
                     'phase-offset': 0,
                     'prio': 2,
                     'state': 'selectable'},
                   { 'direction': 'input',
                     'parent-id': 5,
                     'phase-offset': 0,
                     'prio': 2,
                     'state': 'selectable'}],
 'phase-adjust': 0,
 'phase-adjust-max': 16723,
 'phase-adjust-min': -16723,
```

```

    'type': 'ext'},
  {'board-label': 'GNSS-1PPS',
    'capabilities': {'state-can-change', 'priority-can-change'},
    'clock-id': 5799633565435100600,
    'frequency': 1,
    'frequency-supported': [{'frequency-max': 25000000, 'frequency-min': 1}],
    'id': 40,
    'module-name': 'ice',
    'parent-device': [{'direction': 'input',
                        'parent-id': 4,
                        'phase-offset': 0,
                        'prio': 0,
                        'state': 'selectable'},
                      {'direction': 'input',
                        'parent-id': 5,
                        'phase-offset': 0,
                        'prio': 0,
                        'state': 'selectable'}],
    'phase-adjust': 0,
    'phase-adjust-max': 16723,
    'phase-adjust-min': -16723,
    'type': 'gnss'},
  {'board-label': 'REF-SMA1',
    'capabilities': {'state-can-change'},
    'clock-id': 5799633565435100600,
    'frequency': 1,
    'frequency-supported': [{'frequency-max': 25000000, 'frequency-min': 1}],
    'id': 41,
    'module-name': 'ice',
    'parent-device': [{'direction': 'output',
                        'parent-id': 4,
                        'state': 'disconnected'},
                      {'direction': 'output',
                        'parent-id': 5,
                        'state': 'connected'}],
    'phase-adjust': 0,
    'phase-adjust-max': 480307,
    'phase-adjust-min': -480307,
    'type': 'ext'},
  {'board-label': 'REF-SMA2/U.FL2',
    'capabilities': {'state-can-change'},
    'clock-id': 5799633565435100600,
    'frequency': 1,
    'frequency-supported': [{'frequency-max': 25000000, 'frequency-min': 1}],
    'id': 42,
    'module-name': 'ice',
    'parent-device': [{'direction': 'output',
                        'parent-id': 4,
                        'state': 'disconnected'}],

```



```
        {'direction': 'output',
         'parent-id': 5,
         'state': 'connected'}]],
    'phase-adjust': 0,
    'phase-adjust-max': 480307,
    'phase-adjust-min': -480307,
    'type': 'ext'},
{'board-label': 'PHY-CLK',
 'capabilities': set(),
 'clock-id': 5799633565435100600,
 'frequency': 156250000,
 'frequency-supported': [{'frequency-max': 25000000, 'frequency-min': 1}],
 'id': 43,
 'module-name': 'ice',
 'parent-device': [{'direction': 'output',
                     'parent-id': 4,
                     'state': 'connected'},
                   {'direction': 'output',
                     'parent-id': 5,
                     'state': 'disconnected'}],
 'phase-adjust': 0,
 'phase-adjust-max': 480307,
 'phase-adjust-min': -480307,
 'type': 'synce-eth-port'},
{'board-label': 'MAC-CLK',
 'capabilities': set(),
 'clock-id': 5799633565435100600,
 'frequency': 156250000,
 'frequency-supported': [{'frequency-max': 25000000, 'frequency-min': 1}],
 'id': 44,
 'module-name': 'ice',
 'parent-device': [{'direction': 'output',
                     'parent-id': 4,
                     'state': 'connected'},
                   {'direction': 'output',
                     'parent-id': 5,
                     'state': 'disconnected'}],
 'phase-adjust': 0,
 'phase-adjust-max': 480307,
 'phase-adjust-min': -480307,
 'type': 'synce-eth-port'},
{'board-label': 'CVL-SDP21',
 'capabilities': {'state-can-change'},
 'clock-id': 5799633565435100600,
 'frequency': 1,
 'frequency-supported': [{'frequency-max': 25000000, 'frequency-min': 1}],
 'id': 45,
 'module-name': 'ice',
 'parent-device': [{'direction': 'output',
```

```

        'parent-id': 4,
        'state': 'disconnected'},
    {'direction': 'output',
     'parent-id': 5,
     'state': 'connected'}]],
    'phase-adjust': 0,
    'phase-adjust-max': 480307,
    'phase-adjust-min': -480307,
    'type': 'ext'},
{'board-label': 'CVL-SDP23',
 'capabilities': {'state-can-change'},
 'clock-id': 5799633565435100600,
 'frequency': 1,
 'frequency-supported': [{'frequency-max': 25000000, 'frequency-min': 1}],
 'id': 46,
 'module-name': 'ice',
 'parent-device': [{'direction': 'output',
                    'parent-id': 4,
                    'state': 'disconnected'},
                  {'direction': 'output',
                    'parent-id': 5,
                    'state': 'connected'}]],
 'phase-adjust': 0,
 'phase-adjust-max': 480307,
 'phase-adjust-min': -480307,
 'type': 'ext'},
{'capabilities': {'state-can-change'},
 'clock-id': 5799633565435100600,
 'id': 47,
 'module-name': 'ice',
 'parent-pin': [{'parent-id': 36, 'state': 'disconnected'},
                {'parent-id': 37, 'state': 'disconnected'}]],
 'phase-adjust-max': 0,
 'phase-adjust-min': 0,
 'type': 'synce-eth-port'},
{'capabilities': {'state-can-change'},
 'clock-id': 5799633565435100600,
 'id': 48,
 'module-name': 'ice',
 'parent-pin': [{'parent-id': 36, 'state': 'disconnected'},
                {'parent-id': 37, 'state': 'disconnected'}]],
 'phase-adjust-max': 0,
 'phase-adjust-min': 0,
 'type': 'synce-eth-port'},
{'capabilities': {'state-can-change'},
 'clock-id': 5799633565435100600,
 'id': 49,
 'module-name': 'ice',
 'parent-pin': [{'parent-id': 36, 'state': 'disconnected'},

```

```

        {'parent-id': 37, 'state': 'disconnected'}]],
    'phase-adjust-max': 0,
    'phase-adjust-min': 0,
    'type': 'synce-eth-port'},
{'capabilities': {'state-can-change'},
 'clock-id': 5799633565435100600,
 'id': 50,
 'module-name': 'ice',
 'parent-pin': [{'parent-id': 36, 'state': 'disconnected'},
                 {'parent-id': 37, 'state': 'disconnected'}]],
 'phase-adjust-max': 0,
 'phase-adjust-min': 0,
 'type': 'synce-eth-port'}
```

This output describes all DPLL pins in the system, including SDPs, recovered clocks and their SyncE ports, SMAs, etc. For a full explanation of all the pin fields, see “[--do pin-get](#)”.

7.2.3 --do device-get

To retrieve configuration information on a specific DPLL, the command `--do device-get` can be used:

```
# ./tools/net/ynl/cli.py --spec Documentation/netlink/specs/dpll.yaml --do device-get -
-json '{"id":<id>'
```

Where:

<id> = ID of device to be retrieved

Example:

```
# ./tools/net/ynl/cli.py --spec Documentation/netlink/specs/dpll.yaml --do device-get -
-json '{"id":4}'
{'clock-id': 5799633565435100600,
 'id': 4,
 'lock-status': 'unlocked',
 'mode': 'automatic',
 'mode-supported': ['automatic'],
 'module-name': 'ice',
 'type': 'eec'}
```

7.2.4 --do device-id-get

A device’s ID can be retrieved if the clock ID and the type of DPLL are known.

```
# ./tools/net/ynl/cli.py --spec Documentation/netlink/specs/dpll.yaml --do device-id-
get --json '{"clock-id":<clock-id>, "type":<type>}'
```

Where:

<clock-id> = clock-id of DPLL of interest

<type> = type of DPLL of interest

Example:

```
# ./tools/net/ynl/cli.py --spec Documentation/netlink/specs/dpll.yaml --do device-id-
get --json '{"clock-id":5799633565435100600, "type":"eec"}'
{"id":4}
```

Note: If a third-party driver supports module names for DPLLs, that name can be used to identify a DPLL as well.

7.2.5 --do pin-get

To retrieve configuration information on a specific DPLL's pin, the command `--do pin-get` can be used:

```
# ./tools/net/ynl/cli.py --spec Documentation/netlink/specs/dpll.yaml --do pin-get --
  json '{"id":<id>'
```

Where:

<id> = ID of device to be retrieved

Example:

```
# ./tools/net/ynl/cli.py --spec Documentation/netlink/specs/dpll.yaml --do pin-get --
  json '{"id":38}'
```

```
{'board-label': 'SMA1',
 'capabilities': {'priority-can-change', 'state-can-change'},
 'clock-id': 5799633565435100600,
 'frequency': 1,
 'frequency-supported': [{'frequency-max': 25000000, 'frequency-min': 1}],
 'id': 38,
 'module-name': 'ice',
 'parent-device': [{'direction': 'input',
                    'parent-id': 4,
                    'phase-offset': 0,
                    'prio': 3,
                    'state': 'selectable'},
                  {'direction': 'input',
                    'parent-id': 5,
                    'phase-offset': 0,
                    'prio': 3,
                    'state': 'selectable'}],
 'phase-adjust': 0,
 'phase-adjust-max': 16723,
 'phase-adjust-min': -16723,
 'type': 'ext'}
```

This example output features all fields that may be present in other pins. The following are the various fields explained:

Field	Description
board-label	The name that the driver assigns to this pin in a human-readable format.
capabilities	The various capabilities of the pin include: <ul style="list-style-type: none"> State-can-change: This pin can connect/disconnect to its parent DPLLs to drive them Priority-can-change: The priority of this pin in the automatic selection algorithm can change Direction-can-change (not listed here): The direction of this pin can change from input to output, and output to input
clock-id	See "--dump device-get".
Esync-frequency (not shown here)	Frequency of the embedded sync signal within its base clock frequency.
Esync-frequency-supported (not shown here)	If a pin can support an embedded signal.
Esync-pulse (not shown here)	The ratio of the high to low state of the embedded sync signal pulse (in percentage).
frequency	The frequency of the pin, measured in Hz.
frequency-supported	The upper and lower bounds of the frequencies supported by the pin. Note that not all frequencies within this range are supported.
id	See "--dump device-get".
module-name	See "--dump device-get".
parent-device	Attributes of a child pin that directly relate to and are inseparable from its parent DPLL(s): <ul style="list-style-type: none"> Direction: The direction of the pin (input/output) Parent-ID: The ID of the parent DPLL to which this pin directly connects. Information in this section is related to this specific parent Phase-offset: The time difference between the reference input clock and the feedback input to the phase detector of the DPLL Prio: The priority of this pin in the automatic selection algorithm State: The current state of the pin – whether it is connected, disconnected or selectable by the DPLL
phase-adjust	Can adjust the DPLL phase based on known delays in the system. This method is used to adjust for the time it takes data to travel through a link, measured in picoseconds.
phase-adjust-max	The upper bound of phase-adjust values.
phase-adjust-min	The lower bound of phase-adjust values.
type	The type of the pin include: <ul style="list-style-type: none"> ext: external input GNSS: GNSS recovered clock int-oscillator: device's internal oscillator sync-eth-port: Ethernet port PHY's recovered clock mux: pin that aggregates another layer of selectable pins

7.2.6 --do pin-id-get

A pin's ID can be retrieved if the board label is known:

```
# ./tools/net/ynl/cli.py --spec Documentation/netlink/specs/dpll.yaml --do pin-id-get -
-json '{ "board-label": "<label>" }'
```

Where:

<label> = board label of pin of interest

Example:

```
# ./tools/net/ynl/cli.py --spec Documentation/netlink/specs/dpll.yaml --do pin-id-get -
-json '{ "board-label": "SMA1" }'
{ 'id': 55 }
```

Note: If there is more than one card of the same type in the system, you may not be able to search using only the board label as a parameter. Additional searching parameters include:

- clock-id
- type
- module-name (if included)
- panel-label (if included)
- package-type (if included)

7.2.7 --do pin-set

To modify a pin's attribute(s), the --do pin-set command can be used:

```
# ./tools/net/ynl/cli.py --spec Documentation/netlink/specs/dpll.yaml --do pin-set --
-json '{ "id": <id>, "<attr>": "<value>" }'
```

Where:

<id> = ID of pin of interest

<attr> = Pin attribute to be modified

<value> = Value to be assigned to the attribute

Example:

```
# ./tools/net/ynl/cli.py --spec Documentation/netlink/specs/dpll.yaml --do pin-set --
-json '{ "id": 28, "frequency": 10000000 }'
None
```

Note: None indicates that execution completed with no errors. However, this does not necessarily mean that the expected behavior occurred. Always check your device/pin's information after modification using "--do pin-get" to verify that the expected behavior occurred.

To modify an attribute on a pin that directly relates to its parent device (an attribute contained under the 'parent-device' array):

```
# ./tools/net/ynl/cli.py --spec Documentation/netlink/specs/dpll.yaml --do pin-set --
-json '{ "id": <id>, "parent-device": { "parent-id": <pid>, "<attr>": "<value>" } }'
```

Where:

<id> = ID of pin of interest
<pid> = The parent's ID of pin of interest
<attr> = Pin attribute to be modified
<value> = Value to be assigned to the attribute

Example:

```
# ./tools/net/ynl/cli.py --spec Documentation/netlink/specs/dpll.yaml --do pin-set --  
json '{"id":19, "parent-device":{"parent-id":2, "state":"connected"}}'  
None
```

Pin of type 'mux' encapsulate another layer of pins below them and serve as a 'parent pin' to these pins. To modify an attribute relating to a child pin on a parent pin:

```
# ./tools/net/ynl/cli.py --spec Documentation/netlink/specs/dpll.yaml --do pin-set --  
json '{"id":<id>, "parent-pin":{"parent-id":<pid>, "<attr>":"<value>"}}'
```

Where:

<id> = ID of pin of interest
<pid> = The parent's ID of pin of interest
<attr> = Pin attribute to be modified
<value> = Value to be assigned to the attribute

Example:

```
# ./tools/net/ynl/cli.py --spec Documentation/netlink/specs/dpll.yaml --do pin-set --  
json '{"id":23, "parent-pin":{"parent-id":3, "prio":"7"}}'  
None
```

7.3 Monitoring DPLL Status

Sample output is listed below and does not display all DPLL pins.

```
{'board-label': 'CVL-SDP20',
  'capabilities': {'state-can-change', 'priority-can-change'},
  'clock-id': 5799633565435100600,
  'frequency': 1,
  'frequency-supported': [{'frequency-max': 25000000, 'frequency-min': 1}],
  'id': 35,
  'module-name': 'ice',
  'parent-device': [{'direction': 'input',
    'parent-id': 4,
    'phase-offset': 0,
    'prio': 255,
    'state': 'selectable'},
    {'direction': 'input',
    'parent-id': 5,
    'phase-offset': 0,
    'prio': 4,
    'state': 'selectable'}],
  'phase-adjust': 0,
  'phase-adjust-max': 16723,
  'phase-adjust-min': -16723,
  'type': 'int-oscillator'},
{'board-label': 'C827_0-RCLKA',
  'capabilities': {'state-can-change', 'priority-can-change'},
  'clock-id': 5799633565435100600,
  'frequency': 1953125,
  'frequency-supported': [{'frequency-max': 25000000, 'frequency-min': 1}],
  'id': 36,
  'module-name': 'ice',
  'parent-device': [{'direction': 'input',
    'parent-id': 4,
    'phase-offset': 0,
    'prio': 8,
    'state': 'selectable'},
    {'direction': 'input',
    'parent-id': 5,
    'phase-offset': 0,
    'prio': 8,
    'state': 'selectable'}],
  'phase-adjust': 0,
  'phase-adjust-max': 16723,
  'phase-adjust-min': -16723,
  'type': 'mux'}
```


7.4 DPLL api Setup Script for version 4.80 and later

1. Assign environment variable for your kernel version:

```
# export KERNEL="/home/user/Documents/linux-6.11.7" (point to your kernel folder)
```

2. Check available devices in dpll subsystem, write to a file, and assign environment variables for later use.

```
python3 $KERNEL/tools/net/ynl/pyynl/cli.py --spec $KERNEL/Documentation/netlink/specs/dpll.yaml --dump device-get > device-get-output.txt
```

```
export DPLL0=$(cat device-get-output.txt | grep -v clock-id | grep id | awk '{print $NF}' | tr -d ',' | head -n 1)
```

```
export DPLL1=$(cat device-get-output.txt | grep -v clock-id | grep id | awk '{print $NF}' | tr -d ',' | tail -n 1)
```

3. Check assigned ID of pins in your dpll subsystem, and write it to a file for use in next steps.

```
# python3 $KERNEL/tools/net/ynl/pyynl/cli.py --spec $KERNEL/Documentation/netlink/specs/dpll.yaml --dump pin-get > pin-get-output.txt
```

4. Use pin-get-output.txt to assign environment variables as pin IDs of specific pins.

```
export SDP22_id=$(awk '/board-label.*CVL-SDP22/,/module-name/' pin-get-output.txt | grep "id" | grep -v clock | awk '{print $NF}' | tr -d ',')
```

```
export SDP20_id=$(awk '/board-label.*CVL-SDP20/,/module-name/' pin-get-output.txt | grep "id" | grep -v clock | awk '{print $NF}' | tr -d ',')
```

```
export SMA1_id=$(awk '/board-label.*SMA1/,/module-name/' pin-get-output.txt | grep "id" | grep -v clock | awk '{print $NF}' | tr -d ',')
```

```
export SMA2_id=$(awk '/board-label.*SMA2/,/module-name/' pin-get-output.txt | grep "id" | grep -v clock | awk '{print $NF}' | tr -d ',')
```

```
export RCLKA_id=$(awk '/board-label.*C827_0-RCLKA/,/module-name/' pin-get-output.txt | grep "id" | grep -v clock | awk '{print $NF}' | tr -d ',')
```

```
export RCLKB_id=$(awk '/board-label.*C827_0-RCLKB/,/module-name/' pin-get-output.txt | grep "id" | grep -v clock | awk '{print $NF}' | tr -d ',')
```

```
export GNSS_id=$(awk '/board-label.*GNSS-1PPS/,/module-name/' pin-get-output.txt | grep "id" | grep -v clock | awk '{print $NF}' | tr -d ',')
```

5. Check if the IDs were set.

```
# env | grep -E "SDP22|SDP20|SMA2|SMA1|RCLKA|RCLKB|DPLL0|DPLL1|GNSS"
```

```
SDP20_id=22 DPLL0=2 DPLL1=3 RCLKA_id=23 SDP22_id=21 GNSS_id=27 SMA2_id=35 SMA1_id=34 RCLKB_id=24
```

6. Use multi-flag set-pin command.

```
"$KERNEL/tools/net/ynl/pyynl/cli.py" --spec "$KERNEL/Documentation/netlink/specs/dpll.yaml" \
```

```
--multi pin-set "{\"id\":$SDP22_id, \"parent-device\":{\"parent-id\":$DPLL0,\"prio\":255}}\" \
```

```
--multi pin-set "{\"id\":$SDP20_id, \"parent-device\":{\"parent-id\":$DPLL0,\"prio\":255}}\" \
```

```
--multi pin-set "{\"id\":$RCLKA_id, \"parent-device\":{\"parent-id\":$DPLL0,\"prio\":8}}\" \
--multi pin-set "{\"id\":$RCLKB_id, \"parent-device\":{\"parent-id\":$DPLL0,\"prio\":9}}\" \
--multi pin-set "{\"id\":$GNSS_id, \"parent-device\":{\"parent-id\":$DPLL0,\"prio\":0}}\" \
--multi pin-set "{\"id\":$SDP22_id, \"parent-device\":{\"parent-id\":$DPLL1,\"prio\":5}}\" \
--multi pin-set "{\"id\":$SDP20_id, \"parent-device\":{\"parent-id\":$DPLL1,\"prio\":4}}\" \
--multi pin-set "{\"id\":$RCLKA_id, \"parent-device\":{\"parent-id\":$DPLL1,\"prio\":8}}\" \
--multi pin-set "{\"id\":$RCLKB_id, \"parent-device\":{\"parent-id\":$DPLL1,\"prio\":9}}\" \
--multi pin-set "{\"id\":$GNSS_id, \"parent-device\":{\"parent-id\":$DPLL1,\"prio\":0}}\" \
[None, None, None, None, None, None, None, None, None]
```

7. Check the result (SMA1 and e_ref_sync have to be configured using echo commands).

```
# cat /sys/kernel/debug/ice/0000:17:00.0/cgu
```

CGU Input status:

input (idx)	state	priority	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	1	1	N/A	
SMA2/U.FL2 (5)	invalid	2	2	N/A	
GNSS-1PPS (6)	invalid	0	0	N/A	

Note: When encountering an error such as "json.decoder.JSONDecodeError: Expecting value: line 1 column 40 (char 39)", double-check your environment variables (see [Step 5](#)).

Note: SMA and e_ref_sync cannot be configured using the DPLL api in this release. SMA configuration works in the upstream driver. Echo commands are recommended for that purpose.

```
# echo "in pin 1 e_ref_sync 2" > /sys/class/net/$ETH/device/pin_cfg
# echo "in pin 5 e_ref_sync 2" > /sys/class/net/$ETH/device/pin_cfg
# echo "prio 3 dpll 1 pin 4" > /sys/class/net/$ETH/device/pin_cfg
# echo "prio 3 dpll 0 pin 4" > /sys/class/net/$ETH/device/pin_cfg
```

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 **phc2sys** 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: Two instances of ptp4l as TimeReceiver or TimeTransmitter at the same time.
- If **ts2phc** shows SKIPS, it is likely that the extts polarity is set wrong in the *ts2phc* config file. It is likely that the extts polarity is set wrong in the *ts2phc* config file, or the controller is not receiving the 1PPS signal.
- 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:

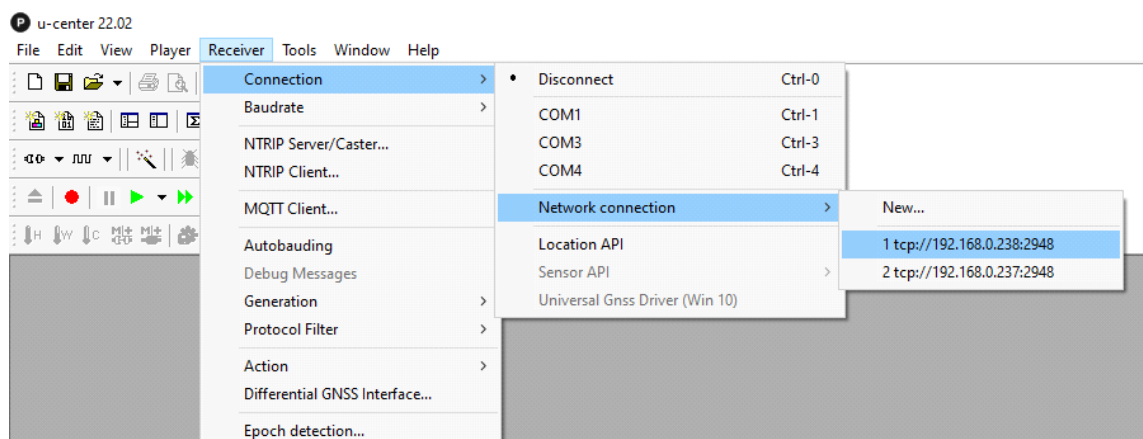
```
/etc/default/grub GRUB_CMDLINE_LINUX_DEFAULT="quiet splash nohz=off" "sudo update-grub"
```

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:



Note: U-blox U-center will only run on Windows OS.

- 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, send the **echo** command in this format:

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

Example - run:

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

- 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 run:

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

2. Disable UART2 in RAM, and Flash:

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

Or run:

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

- If you have AppArmor in your OS, it might block your access to the GNSS interfaces. To avoid that, in your `/etc/apparmor.d/usr.sbin.gpsd` file, add `/dev/gpsd[0-9] rw` and `/dev/gnss[0-9] rw`.
- 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 run:

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

- If your GNSS receiver is not working correctly anymore due to `ubxtool -p reset` (factory reset command was issued), try to re-enable the default settings of the module by:

```
# ubxtool -z CFG-HW-ANT_CFG_VOLTCTRL,1 -z CFG-HW-ANT_CFG_SHORTDET,1 -z CFG-HW-
ANT_CFG_OPENDET,1 -z CFG-HW-ANT_CFG_RECOVER,1 -z CFG-HW-ANT_CFG_PWRDOWN,1 -z CFG-
UART1-ENABLED,0 -z CFG-UART2-ENABLED,0 -z CFG-SIGNAL-SBAS_ENA,0 -z CFG-SIGNAL-
SBAS_L1CA_ENA,0 -z CFG-SIGNAL-GAL_ENA,0 -z CFG-SIGNAL-GAL_E1_ENA,0 -z CFG-SIGNAL-
GAL_E5B_ENA,0 -z CFG-SIGNAL-BDS_ENA,0 -z CFG-SIGNAL-BDS_B1_ENA,0 -z CFG-SIGNAL-
BDS_B2_ENA,0 -z CFG-SIGNAL-GLO_ENA,0 -z CFG-SIGNAL-GLO_L1_ENA,0 -z CFG-SIGNAL-
GLO_L2_ENA,0 -z CFG-MSGOUT-NMEA_ID_GSV_I2C,0 -z CFG-MSGOUT-NMEA_ID_GLL_I2C,0 -z
CFG-MSGOUT-NMEA_ID_VTG_I2C,0 -z CFG-MSGOUT-NMEA_ID_ZDA_I2C,0 -z CFG-MSGOUT-
NMEA_ID_GSA_I2C,0 -P 29.20 -w 3
```

or run:

```
#echo -ne
"\xb5\x62\x06\x8a\x27\x00\x00\x07\x00\x00\x2e\x00\xa3\x10\x01\x2f\x00\xa3\x10\x01
\x31\x00\xa3\x10\x01\x35\x00\xa3\x10\x01\x33\x00\xa3\x10\x01\x05\x00\x52\x10\x00
\x05\x00\x53\x10\x00\x07\x8b" > /dev/gnssX
```

and then:

```
#echo -ne
"\xb5\x62\x06\x8a\x54\x00\x00\x07\x00\x00\x20\x00\x31\x10\x00\x05\x00\x31\x10\x00
\x21\x00\x31\x10\x00\x07\x00\x31\x10\x00\x0a\x00\x31\x10\x00\x22\x00\x31\x10\x00
\x0d\x00\x31\x10\x00\x0e\x00\x31\x10\x00\x25\x00\x31\x10\x00\x18\x00\x31\x10\x00
\x1a\x00\x31\x10\x00\x0c\x00\x91\x20\x00\x09\x00\x91\x20\x00\xb0\x00\x91\x20\x00
\x08\x00\x91\x20\x00\x0b\x00\x91\x20\x00\x0e\x00\x74" > /dev/gnssX
```

Note: It is suggested to check with Intel representative in case of problems with resetting the GNSS module.

- Procedure to build adjtimex on all Linux system:

The Github program in <https://github.com/rogers0/adjtimex/tags> is majorly for Debian but user can download its source code for cross OS compiler for a benefit to Time Sync application.

Below is the procedure to build adjtimex on most of Linux OS:

```
# wget https://github.com/rogers0/adjtimex/archive/refs/tags/debian/1.29-11.tar.gz
# tar -xvf 1.29-11.tar.gz
# cd adjtimex-debian-1.29-11/
# ./configure
# make -j && make -j install
# adjtimex -p
    mode: 0
    offset: -314905
    frequency: -227775
    maxerror: 187000
    esterror: 0
    status: 8193
time_constant: 6
precision: 1
tolerance: 32768000
tick: 10000
raw time: 1699327999s 827878401us = 1699327999.827878401
```

For more operation for adjtimex see detail in “man adjtimex”

- LinuxPTP as a time sensitive API to the OS, it is important and critical that OS need response to all PTP traffic within expected response time (i.e. 100ns for PRTC or 30ns for ePRTC). User could experience unexpected time jump, PTP no response, unknown PTP status or system reboot itself during heavy system workload and PTP workload simultaneously. User shall take care OS RAS (Reliability, Accessibility, Stability) from the beginning along with any heavy workload and time sensitive task (i.e. PTP or Synce).

In Linux Kernel, there are multiple settings that OS will trigger memory data flush to sync all run time data into file system or persistent memory.

Below example shows how to manage dirty data and configure the Kernel behavior:

```
# sysctl -a --pattern dirty\|hung
kernel.hung_task_all_cpu_backtrace = 0
kernel.hung_task_check_count = 4194304
kernel.hung_task_check_interval_secs = 0
kernel.hung_task_panic = 1
kernel.hung_task_timeout_secs = 120
kernel.hung_task_warnings = 10
vm.dirty_background_bytes = 0
vm.dirty_background_ratio = 10
vm.dirty_bytes = 0
```

```
vm.dirty_expire_centisecs = 3000  
vm.dirty_ratio = 40  
vm.dirty_writeback_centisecs = 500  
vm.dirtytime_expire_seconds = 43200
```

Above shows default setting, user can tune the `dirty_ratio` to lower value to get shorter “freeze” time of kernel; also a longer `hung_task_timeout_secs` can prevent kernel to auto-reboot.

Refer to <https://docs.kernel.org/admin-guide/sysctl/vm.html> for more details.

If you get:

```
ptp4l[6855.691]: port 1 (ens3f0): have SYNC 25158, expecting FOLLOW_UP but got SYNC  
25159, dropping  
ptp4l[6855.713]: port 1 (ens3f0): delay timeout  
ptp4l[6855.751]: port 1 (ens3f0): have SYNC 25159, expecting FOLLOW_UP but got SYNC  
25160, dropping  
ptp4l[6855.799]: port 1 (ens3f0): delay timeout  
ptp4l[6855.811]: port 1 (ens3f0): have SYNC 25160, expecting FOLLOW_UP but got SYNC  
25161, dropping  
ptp4l[6855.841]: port 1 (ens3f0): delay timeout  
ptp4l[6855.871]: port 1 (ens3f0): have SYNC 25161, expecting FOLLOW_UP but got SYNC  
25162, dropping
```

means you are using a one step TimeTransmitter partner, and the `assume_two_step` parameter in the config file should be set to 0.

If you see **ts2phc[609113.074]: UTC-TAI offset not set in system! Trying to revert to leapfile,** then try to use an older version of the ts2phc utility.

Appendix B Glossary and Acronyms

Table 8. 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
PDCCP	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 8. Definition of Terms [continued]

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



LEGAL

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

This document (and any related software) is Intel copyrighted material, and your use is governed by the express license under which it is provided to you. Unless the license provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this document (and related materials) without Intel's prior written permission. This document (and related materials) is provided as is, with no express or implied warranties, other than those that are expressly stated in the license.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors which may cause deviations from published specifications.

Copies of documents that are referenced in this document can be obtained by visiting the [Intel Resource and Documentation Center](#).

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

Other names and brands may be claimed as the property of others.

© 2022-2025 Intel Corporation.