Intel® Ethernet Network Adapter E810-XXVDA4T

User Guide

Ethernet Products Group (EPG)

June 2022

Revision 1.2
646265-003

Revision History

Revision	Date	Comments
1.2	June 15, 2022	Updates include the following: • Updated Section 3.1, "Software Support/Packages". • Updated Section 5.9, "Example ptp4l Configuration File for BC".
1.1	March 25, 2022	Updates include the following: • Updated Section 3.1, "Software Support/Packages". • Updated Section 4.9, "Reading Status of the DPLL". • Added Section 4.11.1, "pin_cfg User Readable Format". • Added Section 4.11.2, "cgu_ref_pin/cgu_state Machine Readable Interface". • Updated Section 4.14, "GNSS Module Debug Interface". • Updated Section 5.2.2, "Software Configuration". • Updated Section 5.3.2, "Software Configuration". • Updated Appendix A, "Notes".
1.0	February 18, 2022	Initial public release.

Contents

1.0 Introduction

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

The Intel® Ethernet Network Adapter E810-XXVDA4T (E810-XXVDA4T), is based on the Intel® Ethernet Controller E810 (E810). It is Intel's second in the series of timing-enhanced adapters. The first was the Intel® Ethernet Network Adapter XXV710-DA2T (XXV710-DA2T), based on the Intel® Ethernet Controller XXV710 (XXV710) and optimized for IEEE 1588 PTP applications, launched in mid-2020.

For the latest Message of the Week (MoW), see:

https://cdrdv2.intel.com/v1/dl/getContent/634150

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 Linux Kernel Interface"
- Section 5.0, "Configuration Setup"
- Section 6.0, "Initial Test Setup"
- Appendix A, "Notes"
- Appendix B, "Glossary and Acronyms"

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

1.1 Reference Documents

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

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

Table 1. Reference Documents

Document Title	Document ID
Intel® Ethernet Controller E810 Datasheet	613875
Intel® Ethernet Controller E810 Specification Update	616943
Intel® Ethernet Controller E810 Feature Support Matrix	607252
Intel® Ethernet IEEE 1588 and SyncE Application Note	635237
Time Synchronization for 5G Baseband Unit (BBU) Application Note	630883
Snow Ridge BTS SoC Product Family Synchronous Ethernet Application Note	603587
Intel® Ethernet Network Adapter XXV710-DA2T User Guide	633250
Synce4l tool for Intel® E810-XXVDA4T Adapter	680147

Other documents that might be of interest include the following:

- IEEE 1588-2008 (v2.0)
- IEEE 1588-2019 (v2.1)
- IEEE 802.3AS
- ITU-T G.8271
- ITU-T G.8273
- ITU-T G.8273.2
- ITU-T G.8275.1
- ITU-T G.8275.2

Note: Linux PTP project information can be found at:

http://linuxptp.sourceforge.net/

2.0 E810-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).
- Carries forward the XXV710-DA2T PTP features, while adding support for additional timing signal sources.
- 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.
- Newly-developed utility (**ts2phc**) for the Linux PTP stack optimized for synchronizing the PHC to a 1PPS input, rather than PTP messages.
- When combined with an external GNSS 1PPS input, the features of the adapter and Linux PTP stack provide the complete solution for a low cost PTP grand leader, while maintaining the ability to provide the host system's LAN connectivity.
- Exposed PTP timing signals on a front panel using SMA connectors. This enables timing signals to easily be connected by the end user after the adapter is installed in a system. Timing signals on the SMAs can be configured as inputs or outputs, typically configured for one pulse per second (1PPS) operation, but they will support a 10 MHz 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.
- 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 ±1.5 µs.
- Add support for Synchronous Ethernet (SyncE) clock recovery (ITU G.8262).
- Add a high-quality DPLL to support multiple input clock functions and multiple output frequencies. The DPLL IC has separate integrated DPLL instances DPLL0 is used for generating high stability clock signal and DPLL1 used for driving the output signals.
- The timing information from the DPLL (that can arrive from the GNSS module, SMA connectors, E810 SDPs, or from the recovered SyncE signals) are routed to the E810 and can be used to synchronize the PHC (PTP hardware clocks). All E810 ports share only one physical PHC, and this is particularly useful in creating a Boundary Clock (BC) functionality like what is defined in ITU-T G.8273.2 (but without SyncE).

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

Table 2 provides additional information on the E810-XXVDA4T:

Table 2. E810-XXVDA4T Product Information

Product Description	1588 PTP/SyncE/GNSS 10/25GbE SFP28 PCIe adapter
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), 1x SMB (GNSS antenna)
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 (±1.5 µs under ±5 °C ΔT)
Power	23 W typical, 34 W max power consumption (with Class 3 optics at 25G maximum traffic)
Operating Temperature	0-61 °C with required airflow (passive heatsink); 65 °C target
EMI	FCC Class A
Manageability	NC-SI over MCTP (over PCIe/SMBus); EFI based iSCSI boot; EFI/legacy PXE boot support

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.

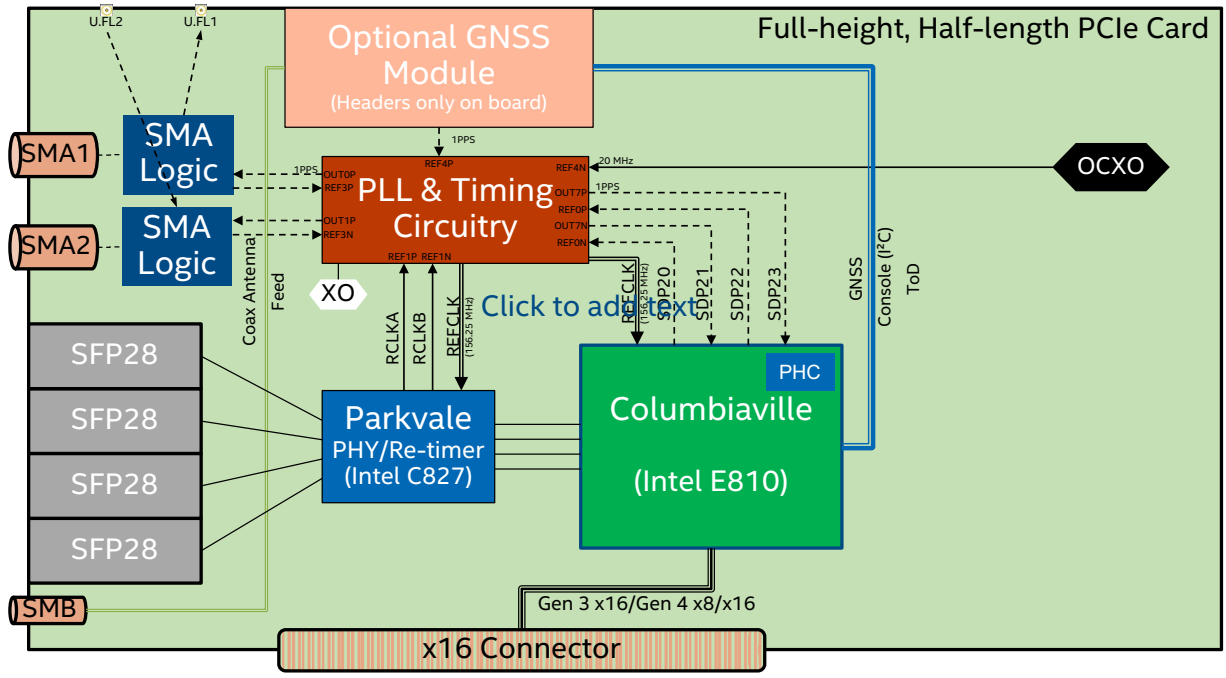


Figure 1. E810-XXVDA4T Block Diagram

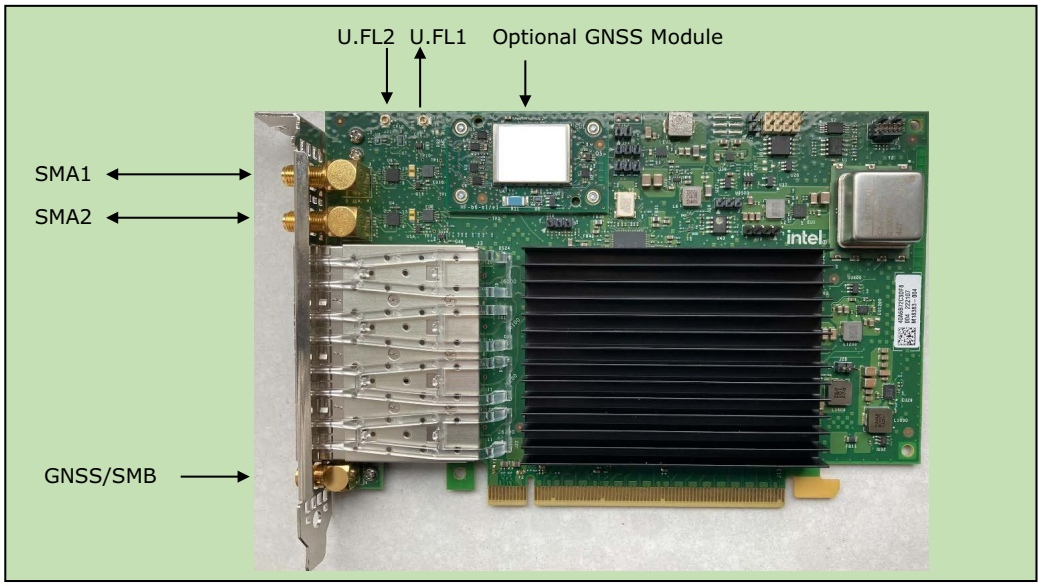


Figure 2. E810-XXVDA4T Connector Locations

Table 3. E810-XXVDA4T GNSS Antenna Characteristics

Characteristic	Min	Typical	Max
Characteristic impedance	-	50 Ω	-
DC operating voltage	< 2.9	3.3	> 3.6
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	-

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	-

3.0 Software, Firmware, and Drivers

3.1 Software Support/Packages

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

   E810 Technical Library:

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

   Operating System Scope for E810 Timing-Enhanced Adapters:

   The following table lists the operating system support for a given release starting with Release 26.8, which represents the first production release for E810-XXVDA4T.

Operating System	Software Release Version			Notes
	26.8	27.1	27.2.1	
Linux RHEL 7.9	SNV	X	X	
Linux RHEL 8.4	X	X	X	
Linux RHEL 8.5	X	X	X	
Linux SLES 12 SP5	X	X	X	
Linux SLES 15 SP3	X	X	X	
Linux Stable Kernel version 3.x	SNV	SNV	SNV	
Linux Stable Kernel version 4.x	SNV	SNV	SNV	
Linux Stable Kernel version 5.x	SNV	SNV	SNV	
Linux Ubuntu 18.04	SNV	SNV	SNV	
Linux Ubuntu 20.04	SNV	X	X	
UEFI 2.1	---	X	X	
UEFI 2.3	---	X	X	
UEFI 2.4	---	X	X	
UEFI 2.6	---	X	X	
UEFI 2.7	---	X	X	
UEFI 2.8	---	X	X	
VMware ESXi 7.0U3	---	X	X	1

Notes:

1. Support for Extended PTP but NOT SyncE.

Legend:

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

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

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

NVM Update Tool:

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

Always choose the latest.

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

2. Download and install the complete driver package:

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

   Always choose the latest.

   Or, download the Linux driver only:

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

   Note: For the newest features, always use the latest driver and NVM version. This document is based on driver version *ice-1.8.3* or newer.

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

```
# cd ice-1.x.x/src
# make -j install
# insmod ./auxiliary.ko  //Depends on the kernel version if you need it.
# insmod ./ice.ko
```

4. Update the NVM to the latest image:

```
# cd E810_NVMUpdatePackage_v3_10_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: 1.8.3
firmware-version: 3.20 0x8000d854 1.3146.0
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. Download and install **linuxptp**, version 3.1.1 or later from SourceForge:

   https://sourceforge.net/projects/linuxptp/

   or use:

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

3.2 Building a linuxptp Project

1. Download the latest **linuxptp** release from SourceForge (v3.1.1 or later):

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

   or use:

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

2. Extract the downloaded archive:

   ```
   # tar xzf linuxptp-3.1.1.tgz
   ```

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

   ```
   # cd linuxptp-3.1.1
   # make
   ```

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

   ```
   # make install
   ```

   This enables you to run the tools from any directory.

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

   http://linuxptp.sourceforge.net/

3.3 Related linuxptp Information

Following are links to related information:

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

4.0 Configuring the E810-XXVDA4T Using Linux Kernel Interface

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

4.1 Introduction

The Linux kernel provides the standard interface for controlling external synchronization pins. Older versions that do not have this feature are not supported. Only versions with the PTP pins interface implemented are supported. Users will need to back-port the pin interface.

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

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

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

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

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

For a listing of all E810-XXVDA4T adapters, run the following:

```
# grep 000e /sys/class/net/*/device/subsystem_device | awk -F"/" '{print $5}'
enp1s0f0
enp1s0f1
enp1s0f2
enp1s0f3
```

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

```
# export ETH=enp1s0f0
```

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 driver exposes the `ptp7` device:

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

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

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

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

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

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

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

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

```
# ethtool -T <interface name>

PTP Hardware Clock: 7
```

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

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

4.2 DPLL Priority

The E810-XXVDA4T automatically switches reference inputs according to the fixed DPLL priority list, as shown in Table 5.

where:

Pin index = DPLL device physical pin index

EEC - DPLL0 = Ethernet equipment clock source from DPLL0 for frequency adjustments., glitchless.

PPS - DPLL1 = 1 PPS generation from DPLL1 for phase adjustments. Glitches allowed. Slower locking.

Note: DPLL priority list can be changed. See Section 4.11, "Advanced DPLL Configuration".

Table 5. DPLL Priority List

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

4.3 External Connectors

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

The E810-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.

Example:

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

where:

function: 0 = Disabled
1 = Rx
2 = Tx

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

4.4 Channel 1 Configurations

1. SMA1 as 1PPS input:

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

2. SMA1 as 1PPS output:

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

3. U.FL1 as 1PPS output:

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

Note: Setting U.FL1 as a Tx automatically enables SMA1 as Rx.

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

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

5. Disable SMA1:

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

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

6. Disable U.FL1:

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

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

4.5 Channel 2 Configurations

1. SMA2 as 1PPS input:

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

2. SMA2 as 1PPS output:

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

3. U.FL2 as 1PPS input:

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

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

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

5. Disable SMA2:

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

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

6. Disable U.FL2:

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

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

4.6 Recovered Clocks (G.8261 SyncE Support)

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

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

There are two recovered clock outputs from 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.

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
```

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
```

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
```

4.7 External Timestamp Signals

The E810-XXVDA4T can use external 1PPS signals filtered out by the DPLL as its own time reference.

When the DPLL is synchronized to the GNSS module or an external 1PPS source, the **ts2phc** tool can be used to synchronize the time to the 1PPS signal.

Note: Extts event support is only enabled on Port 0.

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

# cat $TS2PHC_CONFIG
[global]
use_syslog 0
verbose 1
logging_level 7
ts2phc.pulsewidth 100000000
#For GNSS module
#ts2phc.nmea_serialport  /dev/ttyGNSS_BBDD_0 #BB bus number DD device number /dev/
ttyGNSS_1800_0
#leapfile /../<path to .list leap second file>
[<network interface>]
ts2phc.extts_polarity
rising
```

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

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

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

The E810-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 1 > /sys/class/net/$ETH/device/ptp/ptp*/pins/SMA1
   ```

2. U.FL1 as 1PPS output:

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

3. SMA2 as 1PPS output:

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

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

4.9 Reading Status of the DPLL

The E810-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. Most of the information is available using the **pin_cfg** or **cgu_ref_pin**/**cgu_state** option as detailed in Section 4.11, "Advanced DPLL Configuration".

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

CGU Input status:
                    |            |      priority     |
      input (idx)   |    state   | EEC (0) | PPS (1) |
  ------------------------------------------------------
     CVL-SDP22 (0)  |    invalid |       8 |       8 |
     CVL-SDP20 (1)  |    invalid |      15 |       3 |
  C827_0-RCLKA (2)  |    invalid |       4 |       4 |
  C827_0-RCLKB (3)  |    invalid |       5 |       5 |
          SMA1 (4)  |    invalid |       1 |       1 |
    SMA2/U.FL2 (5)  |    invalid |       2 |       2 |
     GNSS-1PPS (6)  |      valid |       0 |       0 |

EEC DPLL:
    Current reference:  GNSS-1PPS
    Status:             locked_ho_ack
```

```
PPS DPLL:
    Current reference:  GNSS-1PPS
    Status:             locked_ho_ack
    Phase offset:       -217
```

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.

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

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

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

4.10 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
```

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

```
# export ETH=ens801f0

ethtool --show-priv-flags $ETH

Private flags for ens801f0:

link-down-on-close           : off
fw-lldp-agent                : off
```

```
channel-inline-flow-director : off
channel-inline-fd-mark       : off
channel-pkt-inspect-optimize : on
channel-pkt-clean-bp-stop    : off
channel-pkt-clean-bp-stop-cfg: off
vf-true-promisc-support      : off
mdd-auto-reset-vf            : off
vf-vlan-prune-disable        : off
legacy-rx                    : off
dpll_monitor                 : on
extts_filter                 : off
```

Enabling DPLL monitoring:

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

Disabling DPLL monitoring:

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

4.11 Advanced DPLL Configuration

4.11.1 pin_cfg User Readable Format

The DPLL on the E810-XXVDA4T offer some advanced configuration options. These options are not needed on regular applications and can cause problems. Please use these commands with extra care. The DPLL will go back to the default values after AC cycle of the NIC.

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.

To check the DPLL pin configuration:

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

in
| pin| enabled|       freq| phase_delay| esync| DPLL0 prio| DPLL1 prio|
|   0|       1|          1|           0|     0|          8|          8|
|   1|       1|          1|           0|     0|         15|          3|
|   2|       1|    1953125|           0|     0|          4|          4|
|   3|       1|    1953125|           0|     0|          5|          5|
|   4|       1|          1|        7000|     0|          1|          1|
|   5|       1|          1|        7000|     0|          2|          2|
|   6|       1|          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.

Changing the DPLL priority list:

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

where:

prio value = Desired priority of configured pin [0-14]

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

pin index = Index of pin being configured [0-9]

Example:

Set priority 1 for pin 3 on DPLL 0:

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

Changing input/output pin configuration:

```
# echo "<direction> pin <pin index> <config>" > /sys/class/net/<dev>/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>"
  "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.

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/<dev>/device/pin_cfg
```

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

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

Set 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 4" > /sys/class/net/<dev>/device/pin_cfg
```

4.11.2 cgu_ref_pin/cgu_state Machine Readable Interface

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

```
# cat /sys/class/net/<dev>/device/cgu_ref_pin
```

To check the state of the DPLL0 (EEC DPLL) you can check the `cgu_state`:

```
# cat /sys/class/net/<dev>/device/cgu_state

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

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

Note: The user application can monitor the `cgu_state` and `cgu_ref_pin` to detect the DPLL status changes. This changes will be visible in the dmesg as well.

4.12 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 phase 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.

1. To enable the phase reference pin (CVL-SDP20):

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

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

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

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

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

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

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

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

4.13 1PPS Signals from the DPLL to E810 Device

The DPLL automatically delivers 2x 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.

To ensure that the timestamps from the 1PPS signals are only used when the DPLL is locked to a signal, you can enable the 1 PPS filtering option. By default, filtering is disabled, and all the timestamps are passed along by the E810 *ice* driver.

The DPLL 1PPS filtering can be enabled (on) or disabled (off) by using the ethtool command in the Linux kernel.

```
# export ETH=ens801f0

#ethtool --show-priv-flags $ETH

Private flags for ens801f0:
link-down-on-close           : off
fw-lldp-agent                : off
channel-inline-flow-director : off
channel-inline-fd-mark       : off
channel-pkt-inspect-optimize : on
channel-pkt-clean-bp-stop    : off
channel-pkt-clean-bp-stop-cfg: off
vf-true-promisc-support      : off
mdd-auto-reset-vf            : off
vf-vlan-pruning              : off
legacy-rx                    : off
dpll_monitor                 : on
extts_filter                 : off
```

Enabling DPLL's 1 filtering:

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

Disabling DPLL's 1 filtering:

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

4.14 GNSS Module Debug Interface

The E810-XXVDA4T driver implements two TTY interfaces for receiving NMEA messages from the optional GNSS module (codename Horizon Arch). The TTY can be found in */dev/ttyGNSS_BBDD_X*.

where:

- BB is the bus number.
- DD is the device number.
- X can be the following:

  0 = The interface is read/write enabled.

  1 = The interface is read only.

```
# cat dev/ttyGNSS_1800_0
$GNRMC,182805.00,A,0.0,N,0.0,E,0.082,,050321,,,D,V*14
$GNVTG,,T,,M,0.082,N,0.152,K,D*34
$GNGGA,182805.00,0.0,N,0.0,E,2,12,0.57,11.1,M,32.7,M,,0000*7D
$GNGSA,A,3,18,20,26,23,16,29,07,15,27,10,,,1.07,0.57,0.90,1*09
$GNGSA,A,3,81,79,72,82,80,78,65,88,87,,,,1.07,0.57,0.90,2*0C
$GNGSA,A,3,02,30,04,36,11,,,,,,,,1.07,0.57,0.90,3*0E
[…]
```

Note: Make sure that you are using */dev/ttyGNSS_BBDD_0* in your *.cfg* file for **linuxptp** if using the **ts2phc** tool.

Note: The QS samples contained 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: With **gpsmon** and **gpsd** you can check some of the parameters provided by the GNSS module.

```
# gpsmon /dev/ttyGNSS_8400_0
```

```
localhost.localdomain:/dev/ttyNMEA0183>

┌──────────────────────────────────────────────────────────────────────────────┐
  Time: 2022-03-18T11:14:55.000Z Lat:  51 32.622040' N  Lon:   1 46.279180' W
└──────────────────────────── Cooked TPV ──────────────────────────────────────┘

┌──────────────────────────────────────────────────────────────────────────────┐
  GNRMC GNGGA GNTXT
└──────────────────────────── Sentences ──────────────────────────────────────┘

┌──────────────────┐┌──────────────────────────────┐┌──────────────────────────┐
  Ch PRN  Az El S/N || Time:      111455.00        || Time:      111455.00
   0                || Latitude:    5132.62204 N   || Latitude:  5132.62204
   1                || Longitude:  00146.27918 W   || Longitude: 00146.27918
   2                || Speed:     0.046            || Altitude:  142.9
   3                || Course:                     || Quality:   1   Sats: 08
   4                || Status:    A       FAA: A   || HDOP:      1.29
   5                || MagVar:                     || Geoid:     47.6
   6                ||└──────────── RMC ──────────┘└──────────── GGA ──────────┘
   7                ||┌──────────────────────────────┐┌──────────────────────────┐
   8                || Mode:    Sats:                || UTC:           RMS:
   9                || DOP: H=      V=      P=     || MAJ:           MIN:
  10                || TOFF: -209.879865740        || ORI:           LAT:
  11                || PPS:                        || LON:           ALT:
└──────── GSV ──────┘└──────────── GSA + PPS ──────┘└──────────── GST ──────────┘
```

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.

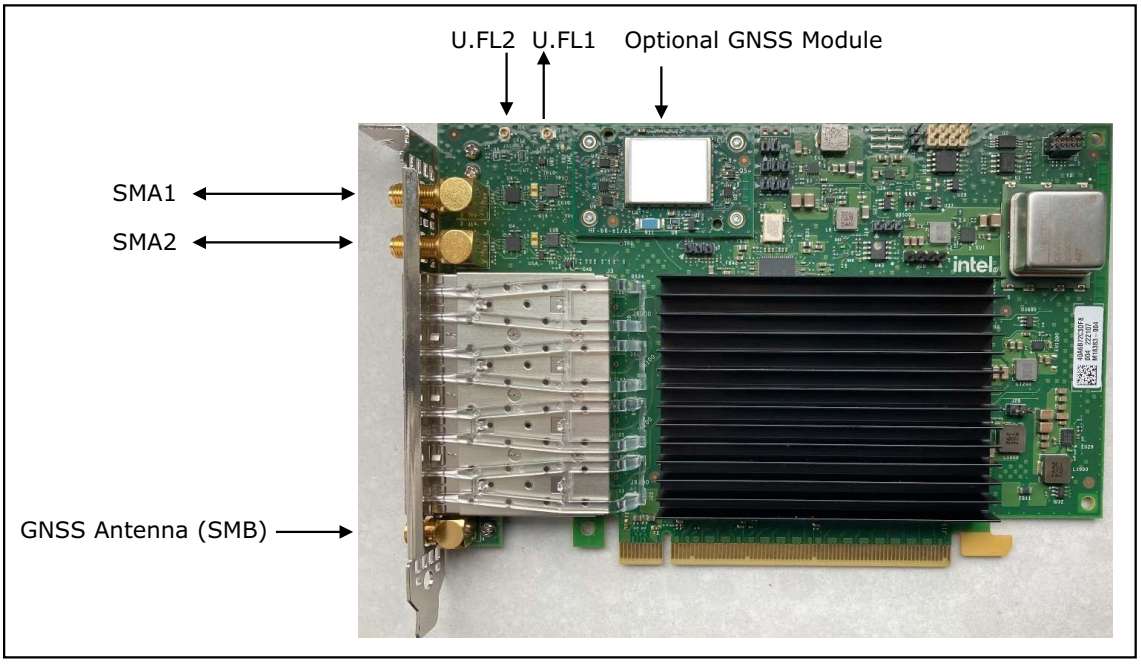


Figure 3. E810-XXVDA4T Connector Locations

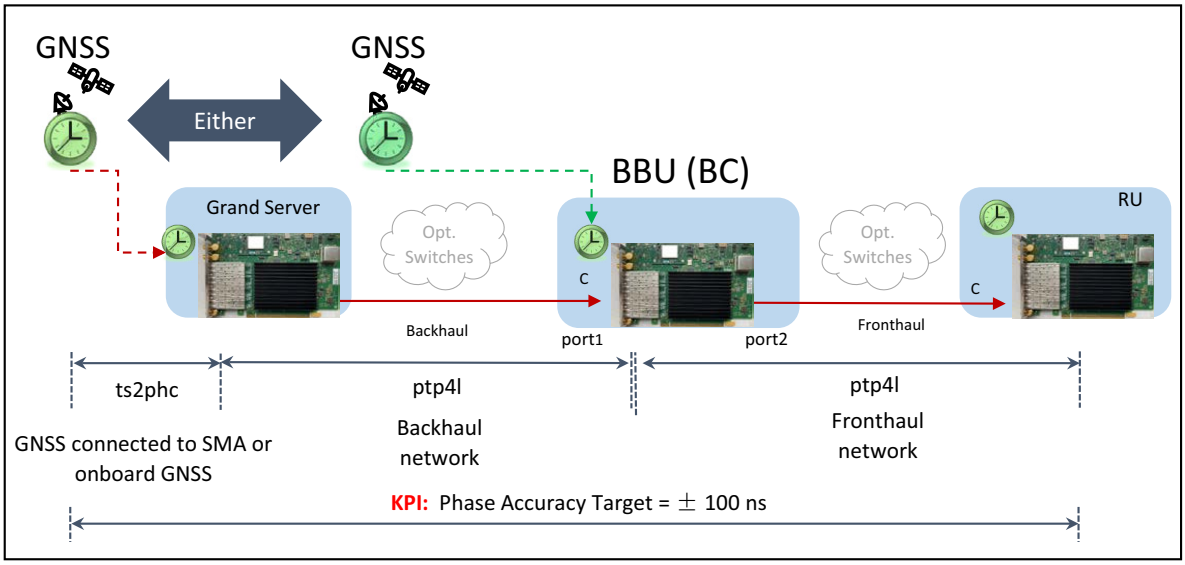


Figure 4. E810-XXVDA4T Connections

5.1 Disable All SMA and U.FL Connections

1. Set interface device:

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

2. Disable U.FL2:

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

3. Disable U.FL1:

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

4. Disable SMA2:

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

5. Disable SMA1:

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

6. All disabled:

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

7. Check with dmesg:

```
# dmesg

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

5.2 PTP Grand Leader (GM) with Optional GNSS Module

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

5.2.1 External Connections

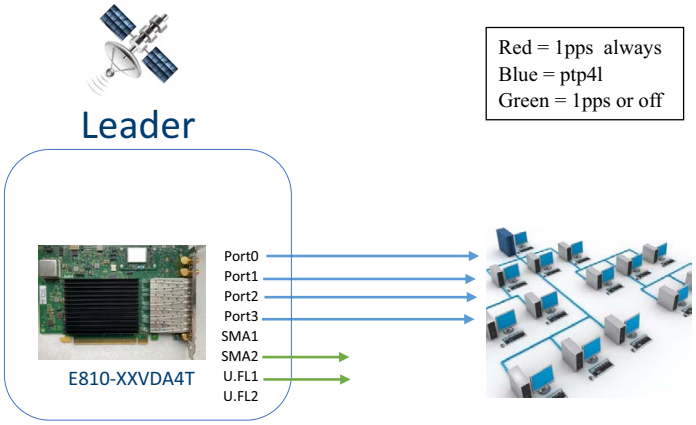


Figure 5. External Connections: PTP Grand Leader with Optional GNSS Module

5.2.2 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$5}' | head -n 1`
# export PCI_SLOT=`grep PCI_SLOT_NAME /sys/class/net/$ETH/device/uevent | cut -c
15-`
```

2. Make sure GNSS has an active connection: */dev/ttyGNSS_BBDD_0* where *BB* stands for the bus number and DD is the device number.

```
# cat /dev/ttyGNSS_1800_0
$GNRMC,182805.00,A,0.0,N,0.0,E,0.082,,050321,,,D,V*14
$GNVTG,,T,,M,0.082,N,0.152,K,D*34
$GNGGA,182805.00,0.0,N,0.0,E,2,12,0.57,11.1,M,32.7,M,,0000*7D
$GNGSA,A,3,18,20,26,23,16,29,07,15,27,10,,,1.07,0.57,0.90,1*09
$GNGSA,A,3,81,79,72,82,80,78,65,88,87,,,,1.07,0.57,0.90,2*0C
$GNGSA,A,3,02,30,04,36,11,,,,,,,,1.07,0.57,0.90,3*0E
```

3. Run **ts2phc**:

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

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

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

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

4. Run **phc2sys**:

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

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

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

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

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

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

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

5. Run **ptp4l**:

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

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

5.3 PTP Grand Leader (GM) with External GNSS Clock

5.3.1 External Connections

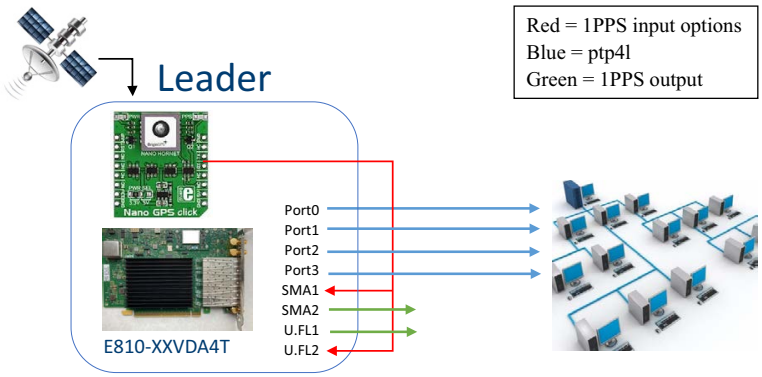


Figure 6. External Connections: PTP Grand Leader with External GNSS Clock

5.3.2 Software Configuration

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

1. Set interface device:

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

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

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

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

3. Verify that the outputs were set correctly:

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

   Or:

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

4. Run **ts2phc**:

   Running on Port 0:

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

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

   Note: See Section 5.8, “Example ts2phc Configuration File”.

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

5. Run **ptp4l**:

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

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

5.4 Boundary Clock Configuration

5.4.1 External Connections

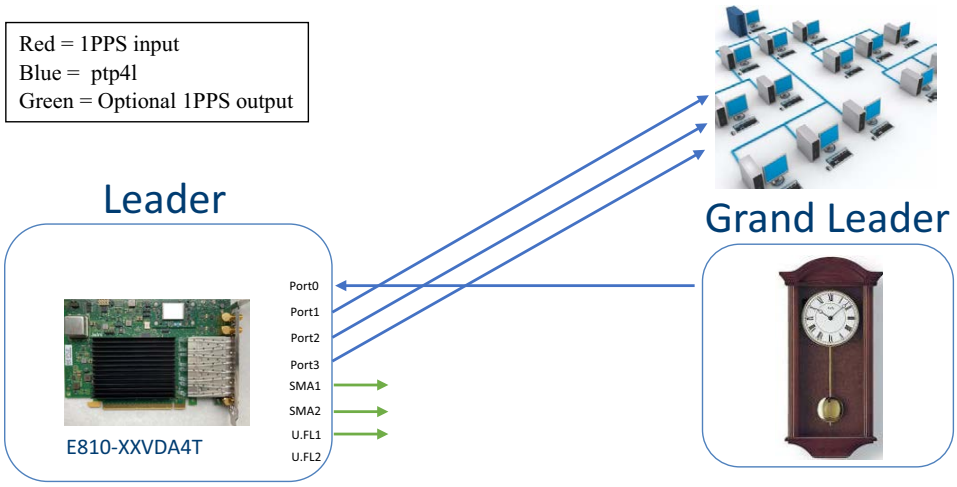


Figure 7. External Connections: Boundary Clock Configuration

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 synchronize the DPLL1 to the E810 PHC synced by **ptp4l**):

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

   Or, if users want to set SDP22 (to synchronize the DPLL0 to E810 PHC synced by **ptp4l**):

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

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

3. Enable SMA1 output (for visibility):

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

Or:

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

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

- Enable SMA2 output: (for visibility):

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

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

4. Verify if the outputs were set correctly:

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

Or:

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

5. Run **ptp4l**:

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

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

5.5 Port Configured as Follower

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

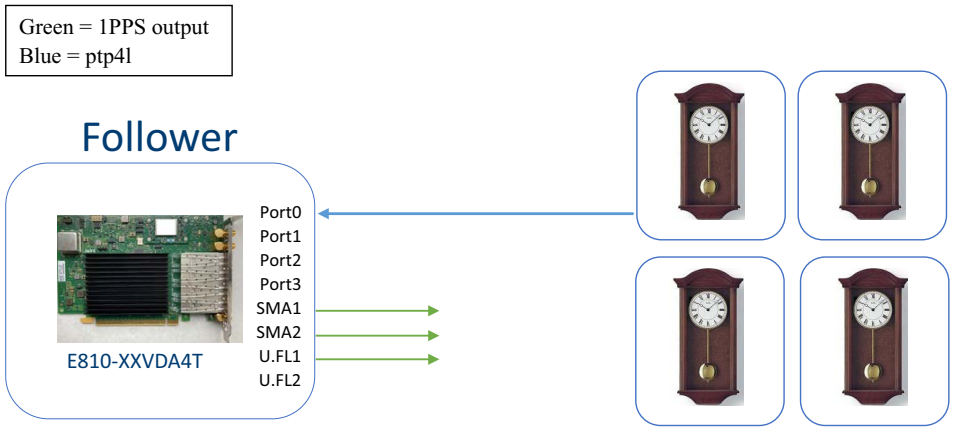


Figure 8. External Connections: Port Configured as Follower

5.5.2 Software Configuration

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

1. Set interface device:

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

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

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

   Or, if users want to set SDP22 (to synchronize the DPLL0 and DPLL1 to the E810 PHC synced by **ptp4l**):

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

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

3. Enable SMA1 output: (for visibility):

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

   Or:

   - Enable U.FL1 output (for visibility):

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

   - Enable SMA2 output (for visibility):

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

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

4. Verify if the outputs were set correctly:

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

Or:

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

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

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

5. Run **ptp4l**:

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

5.6 SyncE Setup

This configuration shows how to specifically setup SyncE. Note that users can use this SyncE configuration with tools such as **ptp4l** for clock recovery if the time synchronization is disabled.

5.6.1 External Connections

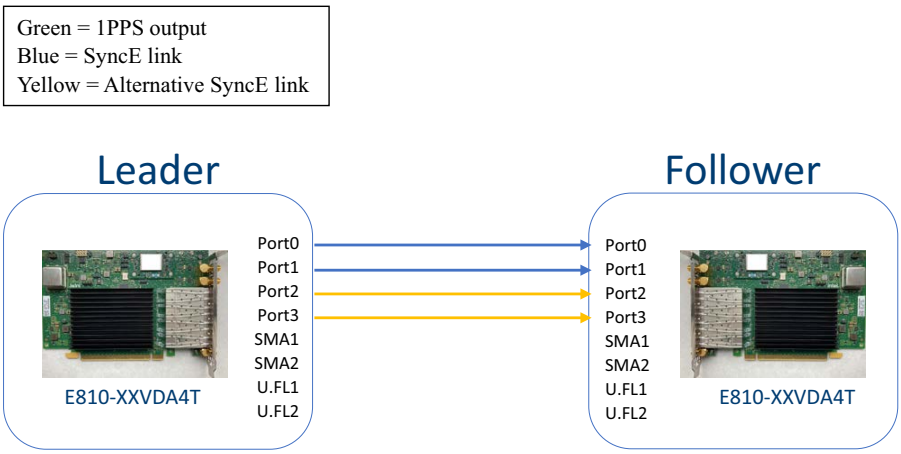


Figure 9. External Connections: SyncE Setup

5.6.2 Software Configuration

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

5.7 O-RAN Configuration 1

5.7.1 External Connections

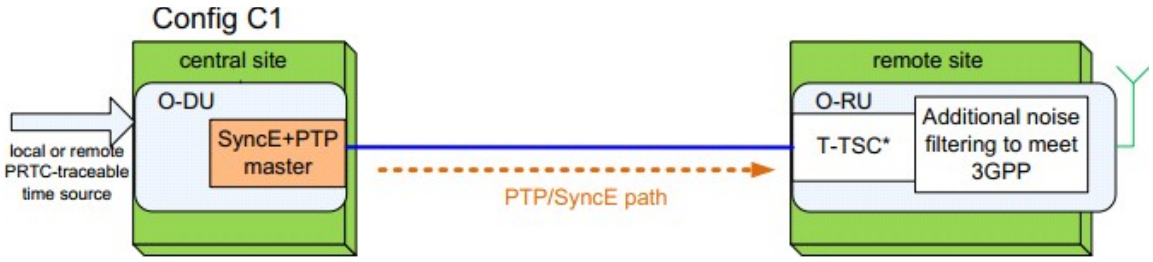


Figure 10. External Connections: O-RAN Configuration 1

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

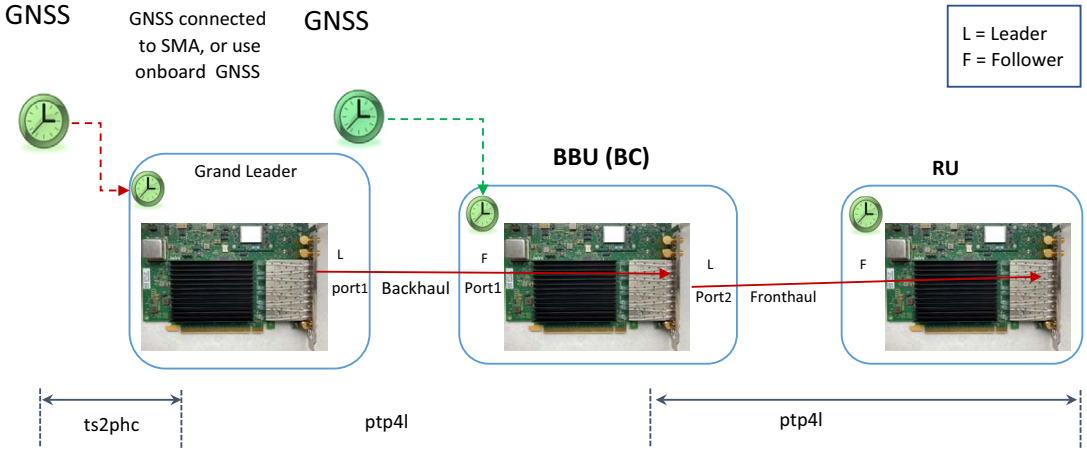


Figure 11. O-RAN Configuration 1 Connections

5.7.2 Software Configuration

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

5.8 Example ts2phc Configuration File

```
[global]
use_syslog          0
verbose             1
logging_level       7
ts2phc.pulsewidth   100000000
# For GNSS module
ts2phc.nmea_serialport   /dev/ttyGNSS_BBDD_0 #BB bus number DD device number /dev/
ttyGNSS_1800_0
leapfile    /home/<user>/linuxptp-3.1/leapseconds.list
[enp1s0f0(dev/ptp4)]
ts2phc.extts_polarity   rising
```

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

5.9 Example ptp4l Configuration File for BC

```
LEADER
[global]
dataset_comparison          G.8275.x
G.8275.defaultDS.localPriority    128
maxStepsRemoved                   255
logAnnounceInterval         -3
logSyncInterval             -4
logMinDelayReqInterval      -4
serverOnly                  0
G.8275.portDS.localPriority       128
network_transport           UDPv4
clock_type                  BC
tx_timestamp_timeout        100
clock_servo                 linreg

[enp1s0f0]
serverOnly          1
[enp1s0f1]
serverOnly          1
[enp1s0f2]
serverOnly          1

FOLLOWER
[global]
dataset_comparison          G.8275.x
G.8275.defaultDS.localPriority    128
maxStepsRemoved                   255
logAnnounceInterval         -3
logSyncInterval             -4
logMinDelayReqInterval      -4
```

```
serverOnly                  0
G.8275.portDS.localPriority       128
network_transport           UDPv4
clock_type                  OC
tx_timestamp_timeout        100
clock_servo                 linreg

[unicast_master_table]
table_id                    1
logQueryInterval            2
UDPv4                       192.168.2.1

[enp1s0f3]
serverOnly                  0
unicast_master_table        1
```

6.0 Initial Test Setup

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

6.1 Test Diagram

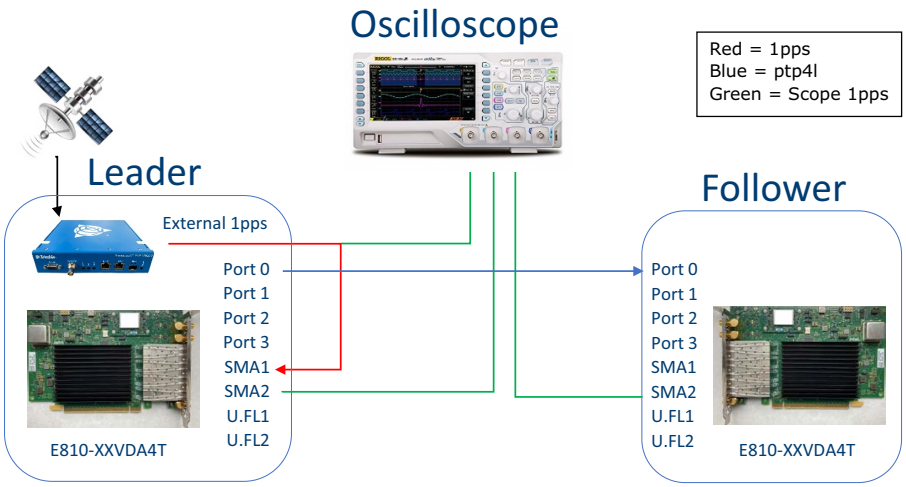


Figure 12. Test Setup

6.2 Software Configuration

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

6.2.1 Leader Adapter

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

1. Set interface device:

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

2. SMA1 as input and SMA2 as output:

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

   Note: As a side effect of having a very accurate DPLL, synchronization between the two E810-XXVDA4T adapters can take up to several hours to change to a locked state.

3. Run **ts2phc**:

   Running on Port 0:

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

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

4. Run **ptp4l**:

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

6.2.2 Follower Adapter

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

1. Set interface device:

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

2. SMA2 as output:

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

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

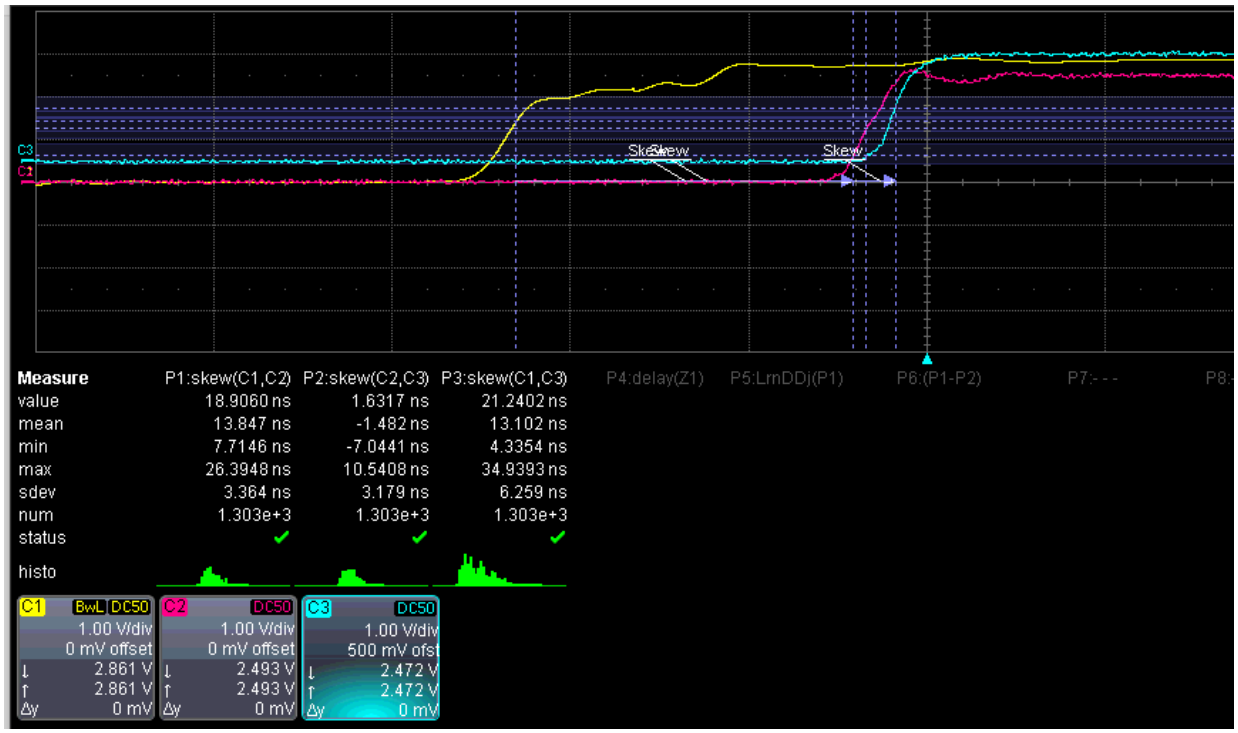
```
# echo 1 0 0 1 0 > /sys/class/net/$ETH/device/ptp/ptp*/period
```

   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
```

6.2.3 Test Results



C1 = Trimble GM	C2 = Leader E810-XXVDA4T	C3 = Follower E810-XXVDA4T

Figure 13. Test Results

Appendix A Notes

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

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

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

- If users run into a “clock frequency higher than an expected” **ptp4l** error, they might be running **ptp4l** as follower and leader on the wrong system.
- If they have **ts2phc** giving SKIPS, they probably have their exit polarity set wrong in the *ts2phc* config file.
- 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.

- 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.

Appendix B Glossary and Acronyms

Table 6. Definition of Terms

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

Table 6. Definition of Terms [continued]

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

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

LEGAL