

Introduction [\(Ask a Question\)](#)

This solution is built around the CoreTSN IP, which implements the IEEE® 802.1Qbv Time-Aware Shaper (TAS) to ensure scheduled transmission of critical traffic with deterministic latency. In conjunction, the Core1588 IP supports Generalized Precision Time Protocol (gPTP) as per IEEE 802.1AS, enabling the system to synchronize its local clock to an external master with nanosecond-level accuracy.

The CoreTSN IP supports configurable traffic classes, gate control lists, and time-slot enforcement, providing the essential building blocks for deterministic Ethernet networks. It also supports advanced TSN features including Frame Preemption (IEEE 802.1Qbu/802.3br) and Per-Stream Filtering and Policing (PSFP – IEEE 802.1Qci).

This application note demonstrates how these capabilities can be evaluated and integrated using the PolarFire® SoC platform.

Table of Contents

Introduction.....	1
1. Demo Design.....	3
2. Demo Requirements.....	4
3. Demo Prerequisites.....	5
4. Demo Setup.....	6
4.1. Jumper Settings and Board Setup.....	6
4.2. Software Setup.....	7
4.3. Instructions to Run the Demo on Linux.....	7
5. Design Resource Utilization.....	11
6. Design Description.....	12
6.1. Design Control Flow.....	12
6.2. Hardware Implementation.....	12
6.3. Software Implementation.....	19
7. Programming the Device Using FlashPro Express.....	21
8. Appendix A: TSN Demo for Windows Support.....	22
9. Appendix B: Linux® Host Setup for Serial Communication.....	27
10. Appendix C: Verifying PTP Hardware Clock (PHC) Support.....	28
11. Appendix D: Design Creation Using TCL Scripts.....	29
12. Glossary.....	30
13. Revision History.....	31
Microchip FPGA Support.....	32
Microchip Information.....	33
Trademarks.....	33
Legal Notice.....	33
Microchip Devices Code Protection Feature.....	33

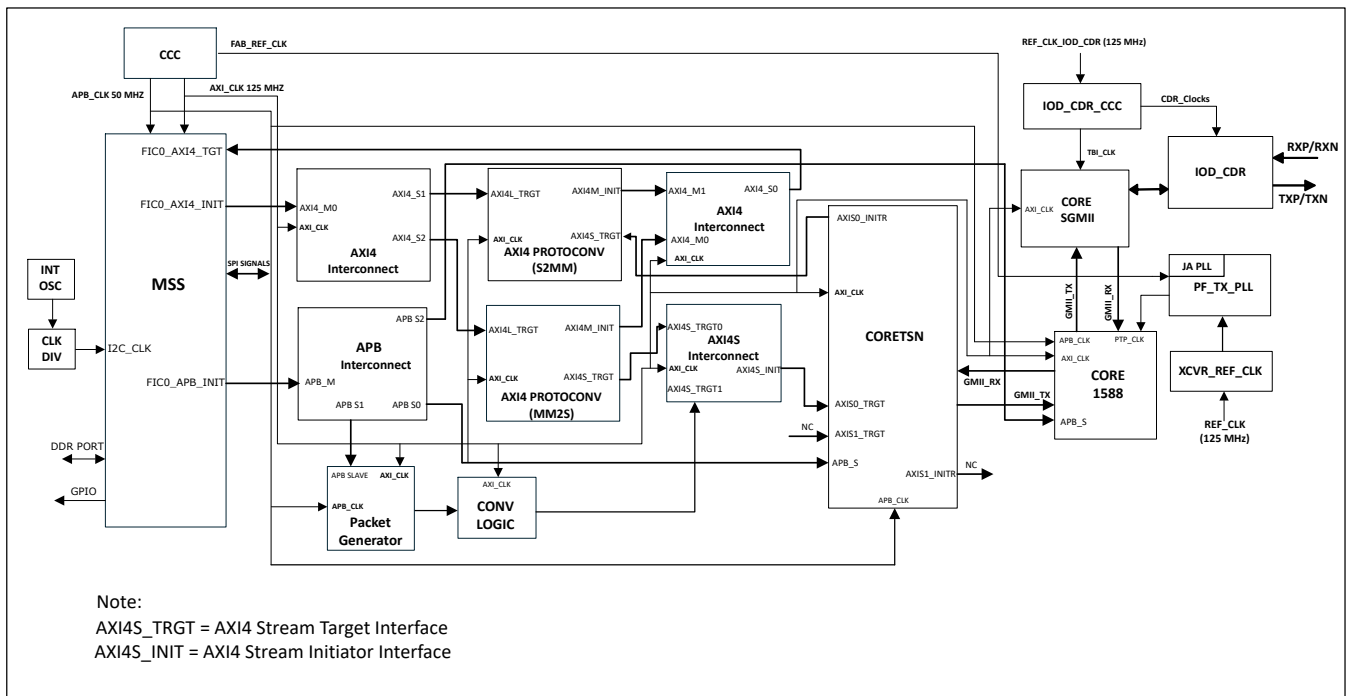
1. Demo Design (Ask a Question)

The demo design integrates the functionality of the TSN IP on a PolarFire® SoC Video kit. The TAS (Qbv) feature of TSN is demonstrated in this demo by controlling the stepper motor and the counter.

The design comprises of the MSS controlling the TSN IP and it schedules the stepper motor and the counter data as a high priority time scheduled traffic, to ensure timely delivery to the client. The design integrates the packet generator to generate the best-effort traffic. The time synchronization is achieved by using the Core1588 IP, which extracts and inserts the time stamp in the corresponding PTP packets received from the network. The line side serial interfaces uses the CoreSGMII IP to convert the Gigabit Media Independent Interface (GMII) to the Serial Gigabit Media Independent Interface (SGMII).

The following block diagram illustrates the block-level view of the current demo.

Figure 1-1. TSN IP Demo



2. Demo Requirements [\(Ask a Question\)](#)

The following table lists the hardware and software required for running the demo.

Table 2-1. Demo Requirements

Requirement	Description
Hardware and Accessories	
PolarFire® SoC Video Kit	MPFS250-VIDEO-KIT Kit Contents: <ul style="list-style-type: none"> • PolarFire SoC Video Kit Board with (MPFS250TS-1FCG1152I) • 4K30 Dual Camera Sensor with Sony IMX334s • HDMI cable • Micro B USB cables (2) • RJ45 Ethernet cable • 12V AC adapter, 12V power cord
FMC Daughter Card	ETH-DC-DUALGBE
Host PC	A host PC with USB, Ethernet port and Linux/Ubuntu OS.
Ethernet cable	To communicate with the PolarFire SoC Video kit or PolarFire SoC ICICLE kit, three Ethernet cables are required.
Network Switch	Generic Gigabit Ethernet Network Switch (Unmanaged), with associated power adapter
Stepper 7 Click	The Stepper 7 Click board™ is a versatile driver designed for bipolar step motors that features two integrated circuits from Microchip: the MTS62C19A Dual Full-Bridge Motor Driver and the MCP23S08 8-bit I/O Expander with a serial interface. It requires a 12V DC supply and a female connector (for more details, see Connector Product Page), which are not included in the package and needs to be procured separately.
Multi Stepper Motor	Stepper Motor QMot.eu, Part Number QSH 4218-35-10-027 Lot No. 1016062(1.001)A
Software	
Program Debug 2025.1	Executable for installing FlashPro Express 2025.1 that programs the FPGA
Putty or TeraTerm or Minicom	UART receiver-transmitter application on the host PC
USBImager	USBImager is required to program a Linux® image to a target memory using the Hart Software Services.
7-Zip	Freeware archiving tool for compressing and decompressing files. This tool is needed for extracting .wic file from the .wic.gz compressed file.
Design Files	
Linux wic Image	TSN_Demo_GUI_Video_Kit_WIC-Image.wic.gz
Job file	TSN_Demo_GUI_Video_Kit.job
Obfuscated and Licensed IPs	
CoreTSN	For obtaining License of the IP, see "Licensing" in the CoreTSN User Guide .
Core1588	For obtaining License of the IP, see "Licensing" in the Core1588 User Guide

3. Demo Prerequisites [\(Ask a Question\)](#)

Before you begin, download the programming job file and the Linux `.wic` image from:
www.microchip.com/en-us/application-notes/an5892.

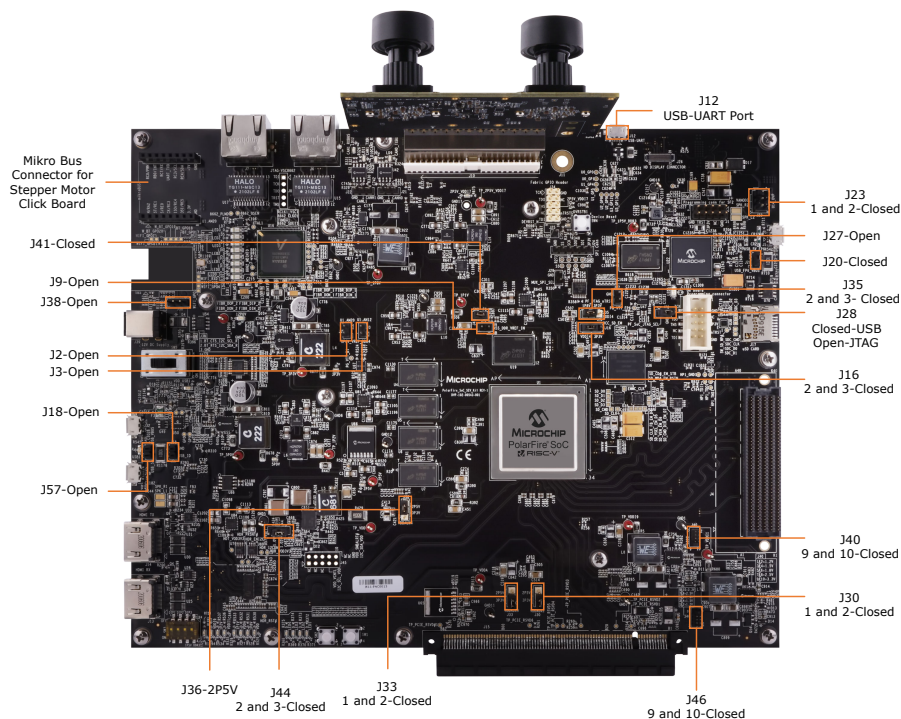
4. Demo Setup [\(Ask a Question\)](#)

This section provides detailed information how to setup and run the demo.

4.1. Jumper Settings and Board Setup [\(Ask a Question\)](#)

The following figure shows the board setup.

Figure 4-1. Board Setup showing Jumper Settings

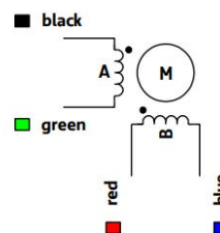


To setup the Video Kit board, perform the following steps:

1. Connect the RJ45 Ethernet cable to any one of the FMC Ethernet Port (J22) of the board and Ethernet network switch.
2. Ensure to make the required wire connections from Stepper Motor to the Click board, as shown in the following figure.

Figure 4-2. Lead Wire Configuration

Cable type	Coil	Function
Black	1B	Motor coil A pin 1
Green	1A	Motor coil A pin 2
Red	2B	Motor coil B pin 1
Blue	2A	Motor coil B pin 2



3. Make sure to connect the 12V DC power adapter to the Click board (for more information, [see this image](#)).

4.2. Software Setup [\(Ask a Question\)](#)

Ensure you have an Ubuntu machine, an Ethernet network switch, and administrative privileges, and verify that the system meets the minimum requirements to set up the TSN demo.

Extract the `.wic` image and to proceed with flashing the image, see [Updating Linux in MPFS Kit](#) (The instructions provided in the link are applicable to both the Icicle and Video Kit).

4.3. Instructions to Run the Demo on Linux [\(Ask a Question\)](#)

To configure the network communication between Ubuntu and Video Kit, perform the following steps (these must be repeated every power cycle):

1. On Video Kit:
 - a. Power on the TSN Video Kit, a login prompt appears. To configure the Linux host setup serial communication, see [Appendix B: Linux Host Setup for Serial Communication](#).
 - b. Enter the username as **root** with no password, and press **enter**.
 - c. To check the IP address, enter `ifconfig` command. The IP address will not display, because the network device has not been initialized yet.
 - d. Navigate to the `/opt/microchip/tsn/` folder and execute `./tsninit.sh` command. Upon successful execution of the script, the Demo Kit receives an IP address from the DHCP server (provided the server is operational and accessible). This script also configures a VLAN on the Demo Kit. The code snippet creates a VLAN interface `en01.13` on `en01`, assigns the IP address `192.168.13.2`, applies an egress QoS map (0:5), and sets the MTU to 1400 bytes. This setup enables isolated and prioritized network traffic within VLAN 13.

Figure 4-3. Configuring VLAN on the Demo Kit

```
root@mpfs-video-kit:/opt/microchip/tsn# ifconfig
ens0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether 00:04:a3:4a:b4:c9  txqueuelen 1000  (Ethernet)
    RX packets 0  bytes 0 (0.0 B)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 0  bytes 0 (0.0 B)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
    device interrupt 68  base 0x2000

eth0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether 00:04:a3:4a:b4:c0  txqueuelen 1000  (Ethernet)
    RX packets 0  bytes 0 (0.0 B)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 0  bytes 0 (0.0 B)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
    device interrupt 64


eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.13.2 netmask 255.255.255.0  broadcast 19.61.33.255
    inet6 fe80::204:a3ff:fe7a:c0d4  prefixlen 64  scopeid 0x20<link>
    ether 00:04:a3:7a:8c:d4  txqueuelen 1000  (Ethernet)
    RX packets 23  bytes 3064 (3.0 KiB)
    RX errors 0  dropped 7  overruns 0  frame 0
    TX packets 41  bytes 6027 (6.0 KiB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1.13: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1400
    inet 192.168.13.2 netmask 255.255.255.0  broadcast 0.0.0.0
    inet6 fe80::204:a3ff:fe7a:c0d4  prefixlen 64  scopeid 0x20<link>
    ether 00:04:a3:7a:8c:d4  txqueuelen 1000  (Ethernet)
    RX packets 0  bytes 0 (0.0 B)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 18  bytes 2311 (2.2 KiB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1  prefixlen 128  scopeid 0x10<host>
    loop  txqueuelen 1000  (Local Loopback)
    RX packets 88  bytes 7618 (7.4 KiB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 88  bytes 7618 (7.4 KiB)
```

2. On Ubuntu Machine: Log in to the Ubuntu machine and execute the following commands to configure VLAN 13, and assigns the IP address to `192.168.3.20`.


```
# TSN Demo configuration PCP 5 for VLAN 13
sudo ip link add link eno1 name eno1.13 type vlan id 13 egress-qos-map 0:5
sudo ip link set dev eno1.13 up
sudo ip addr add 192.168.13.20/24 dev eno1.13
sudo ifconfig eno1.13 mtu 1400
```

 **Important:** If the network device is not named `eno1`, update the commands accordingly.

3. Once the network is configured, test connectivity to the TSN demo kit to by running `ping 192.168.13.2` from the Ubuntu machine. Receiving replies confirms that the communication path is working.

4. Then, on the demo kit, run `ping 192.168.13.20` to test connectivity to the Ubuntu machine. If both ping tests succeed, the VLAN-based communication path is confirmed to be established.
5. For a PTP connection to function between an Ubuntu machine (acting as the initiator) and a PolarFire SoC device (acting as the target), `ptp4l` must be configured on both devices. Perform the following recommended instructions on Ubuntu machine:
 1. Verify that the Ubuntu system's Network Interface Card (NIC) supports Generalized Precision Time Protocol (gPTP). For more information, see [Appendix C: Verifying PTP Hardware Clock \(PHC\) Support](#).
 2. Install the LinuxPTP package with `sudo apt-get install linuxptp` command
 3. Download the `gPTP.cfg` and change the **neighborPropDelayThresh** value from 800 to 150000, to ensure it aligns with the NIC configuration.
 4. Run the `ptp4l` initiator with `sudo ptp4l -i <eth interface> -m -f <path to gPTP.cfg>` command

Note: `ptp4l` starts on the target (PolarFire SoC device) with the execution of the `tsninit.sh` script in Step 4.

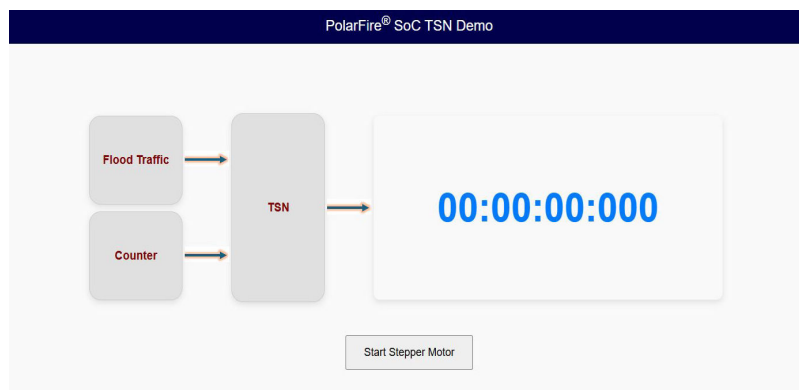
 **Important:** For Windows® support, see the [Appendix A: TSN Demo for Windows Support](#).

4.3.1. Demo at a Glance [\(Ask a Question\)](#)

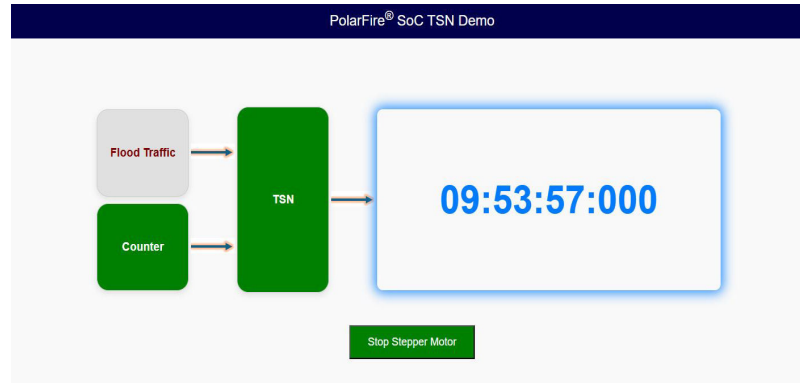
To setup and run the demo, perform the following steps:

1. Launch a web browser such as Chrome or Edge and navigate to <http://192.168.13.2/tsn/>. This displays the TSN demo kit's web page, as shown in the following figure.

Figure 4-4. PolarFire® SoC TSN Demo Page



2. Clicking the **Counter** icon initiates a countdown timer on the demo kit, decrementing by one unit every second. While active, the web browser updates the displayed counter value in real-time by retrieving the latest value from the demo kit every second, providing a live view on the web page.
3. When the **Counter** icon is deactivated, the web browser stops retrieving and displaying the counter value from the demo kit, but the countdown timer continues to operate in the background on the demo kit.
4. Clicking the **Start Stepper Motor** icon initiates continuous operation of the stepper motor, by changing the icon label to **Stop Stepper Motor**. The system then sends commands every second for the motor to complete a full revolution. This process continues until the **Stop Stepper Motor** icon is pressed, which stops the motor.

Figure 4-5. Starting and Stopping the Stepper Motor

5. When the **Flood Traffic** icon is pressed, it starts the flood traffic at a very high rate. This traffic is pre-configured to run for 15 seconds and stops automatically. It cannot be stopped manually.


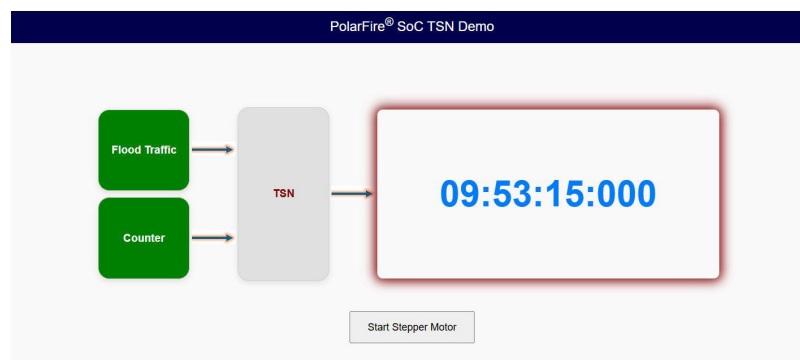
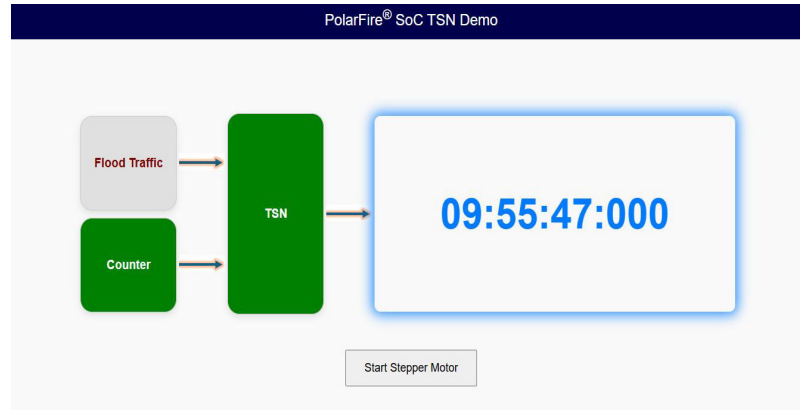
 **Important:** Ensure to run this setup on an isolated network to avoid interference with other systems. TSN is designed to handle and prioritize specific traffic patterns that might not align with standard network configurations.

Figure 4-6. Initiating Flood Traffic

6. To configure the demo kit with the default TSN profile, click the **TSN** icon. When TSN is enabled, a specific portion of the network bandwidth is reserved for motor control commands and counter data updates, ensuring deterministic and time-critical communication. This is managed using the Time-Aware Shaper (TAS), which divides the communication cycle into specific time slots and schedules critical messages to be sent exclusively during these windows. TAS temporarily restricts lower-priority traffic, ensuring that essential messages are delivered on time and without interference, maintaining real-time synchronization for motor controls and counter updates.

Figure 4-7. Configuring the Demo Kit with the Default TSN Profile

7. Deactivating the **TSN** icon removes the TSN profile from the demo kit, enabling dynamic switching between TSN-enabled and standard network modes for flexible configuration and testing.

The demo uses a Click board to control the stepper motor, interfacing with the FPGA over SPI. The SPI driver exposes control signals from the Click board, allowing the motor to be operated by sending data over SPI. Similarly, if other interfaces such as I2C are needed, ensure the appropriate drivers are available and can communicate with the FPGA to control the target device.

5. Design Resource Utilization [\(Ask a Question\)](#)

The following table lists the resource utilization of the TSN demo on PolarFire® SoC MPFS250TS. These values might vary slightly for different Libero® runs, settings and seed values.

Table 5-1. Resource Utilization

Module Name	4LUT (Fabric and Interface)	DFF (Fabric and Interface)	μSRAM (64x12)	LSRAM (20K)
CORERESET_PF_C0_0	1	17	0	0
CORERESET_PF_C1_0	1	17	0	0
CORE1588_C0_0	3450	2686	0	0
COREAXI4PROTOCONV_C0_0	1161	933	9	1
COREAXI4PROTOCONV_C1_0	1171	1032	8	1
CORESGMII_C0_0	1950	994	0	0
CORETSN_C0_0	20654	22074	0	121
CoreAXI4SInterconnect_C0_0	55	6	0	0
DMA_INITIATOR_0	2602	2245	27	0
FICO_INITIATOR_0	11085	6613	28	10
NAT_STR_CONV1_0	4	1	0	0
PF_IOD_CDR_C0_0	243	59	0	0
PF_IOD_CDR_CCC_C0_0	30	31	0	0
PF_XCVR_ERM_C0_0	0	0	0	0
STR_NAT_CONV_0	7	38	0	0
pckt_generator_checker_0	1300	559	0	0
CoreAPB3_C0_0	125	0	0	0
MSS	0	0	0	0

6. Design Description [\(Ask a Question\)](#)

The TSN Demo with GUI involves setting up an Ubuntu machine with VLAN configurations and deploying the Video Kit by installing the WIC image and job file. Once the network is configured, connectivity between the Ubuntu machine and the TSN Demo Video Kit is verified. The demonstration consists of the following phases:

1. Basic Functionality Check:
 - The demo webpage is accessible through a browser.
 - The counter initiates, displaying a decrementing count.
 - Halting the counter stops its status updates.
 - The stepper motor is controlled through the GUI.
2. Time Scheduling:
 - The time aware shaper of the TSN functionality is demonstrated by initiating the flood traffic, during this the counter and stepper motor control data generated by the MSS are placed on the line the unspecified intervals as the time aware shaper is not enabled by the MSS.
 - This results in the intermittent or paused updates of the counter status and the operation of the stepper motor is impacted. When the flood traffic is stopped the counter and motor resumes the normal operation.
 - The MSS enables the TSN time scheduling by sending the motor and counter data as high priority frames in a time scheduled manner by configuring the Gate control list and enabling the priority gates in the required intervals. The best-effort traffic is sent during the times when the priority traffic is not present.
 - This configuration results in the uninterrupted counter updates and the continuous operation of the stepper motor even when the flood traffic is active.

This demonstrates the efficiency of the TSN in the real time transmission without interruptions.

6.1. Design Control Flow [\(Ask a Question\)](#)

This section provides detailed information on control interfaces.

AXI4-Lite Interface: This interface configures the descriptors of the memory mapped IP's S2MM and MM2S. The same interface configures the registers and read the status of the corresponding IP's.

APB Interface: This interface configures the internal registers and read the status of CoreTSN, Core1588 and Packet generator modules. It connects to the CORETSN, CORE1588 and packet generator modules. It configures and reads the status information from these modules.

6.2. Hardware Implementation [\(Ask a Question\)](#)

The following are the Libero SmartDesign modules of the demo design:

- Clock and Reset Module
- APB Module
- Processor Subsystem
- TSN Peripheral Module

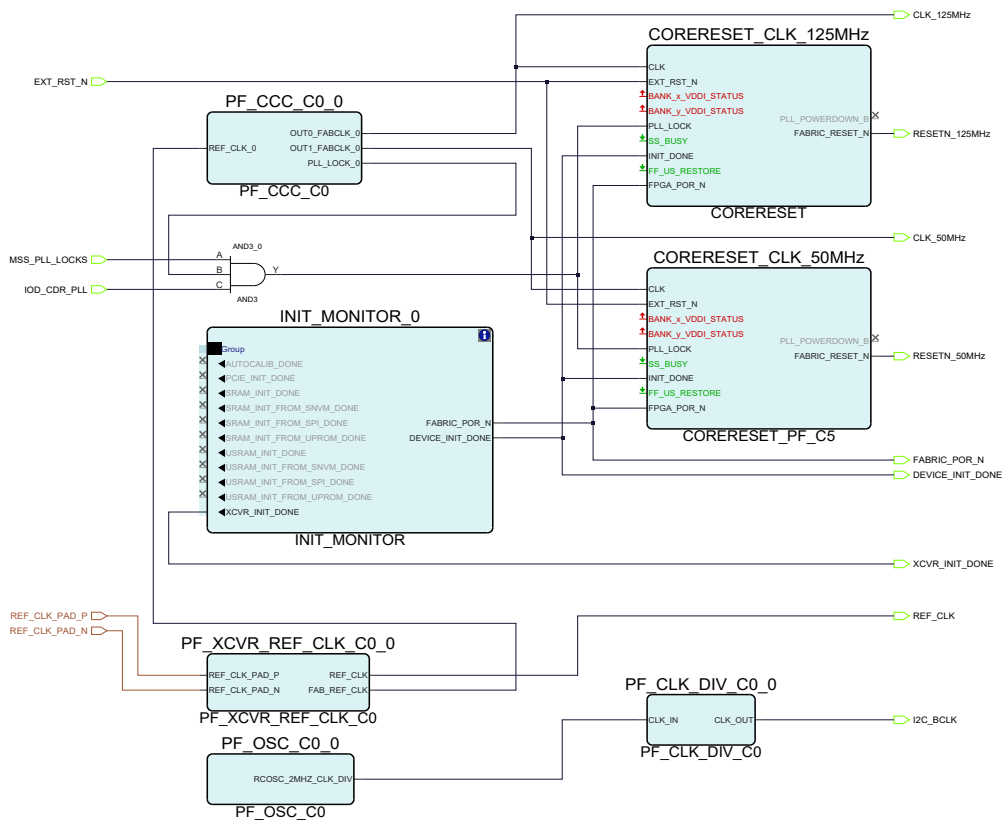
6.2.1. Clock and Reset Module [\(Ask a Question\)](#)

This SmartDesign module generates the necessary clocks and resets for the system. It produces three clocks: the AXI_CLK of 125 MHz and APB Clock of 50 MHz, both generated using a Clock Conditioning Circuit (CCC) with a reference clock of 125 MHz. Additionally, the I²C Clock of 1 MHz is derived from an internal 2 MHz oscillator, utilizing a clock divider module set to a division factor of 2. This module also includes the INIT monitor and the reset circuitry for the 125 MHz and 50 MHz clocks.

The AXI_CLK is used as the system side clock in the design and the APB logic of all the connected IP's in the design operate with the APB Clock.

The following figure shows the SmartDesign of the clock and reset module.

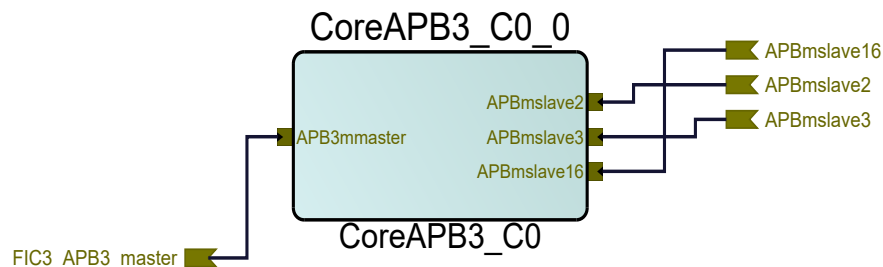
Figure 6-1. SmartDesign of the Clock and Reset Module



6.2.2. APB Module [\(Ask a Question\)](#)

The following figure shows the SmartDesign of the APB module.

Figure 6-2. SmartDesign of the APB Module

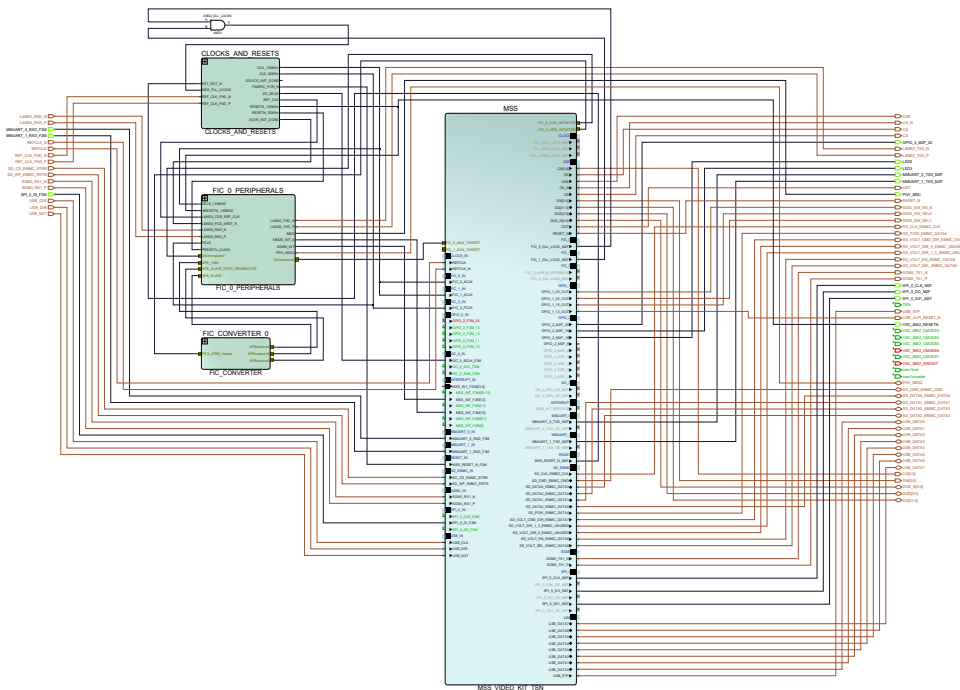


This SmartDesign includes an APB Interconnect module that links the APB Initiator interface from the processor subsystem module to various APB Targets in the demo design. The target devices in the design with APB control interface are connected as through this interconnect. The MSS accesses the internal registers and control status of the various targets through this interface. If the user wants to add any other module to the demo with APB control interface, it has to be connected through this interconnect module as an additional target device.

6.2.3. Processor Subsystem (Ask a Question)

The following figure shows the SmartDesign of processor subsystem module.

Figure 6-3. SmartDesign of Processor Subsystem Module

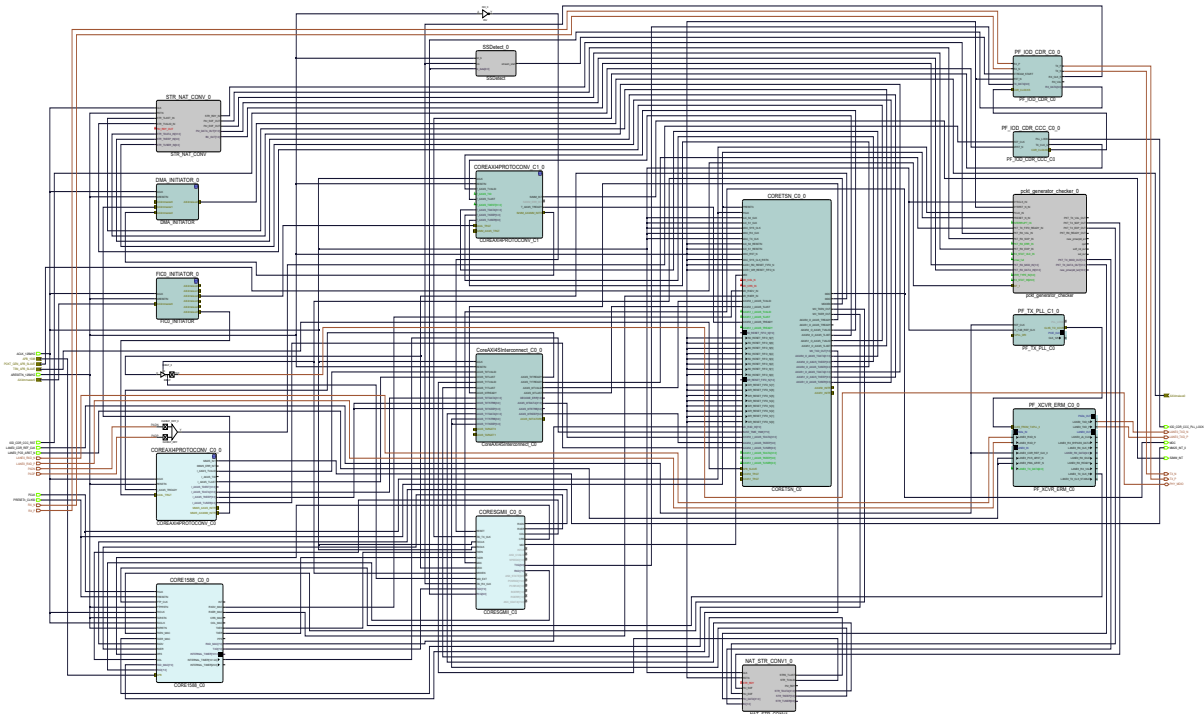


The PolarFire® SoC MSS Configurator provides a GUI that allows embedded software engineers to define the MSS start-up state efficiently. It generates an XML file initialization constructs and outputs a CXZ file for integration into the Libero design flow. The CXZ file contains metadata and port information required by the FPGA designer to complete the MSS and FPGA fabric connectivity. The MSS Configurator is available as a stand-alone application and as part of the Libero SoC design tool suite. The required interfaces of the MSS has to be configured using this configurator. The MSS connects to the fabric modules through the FIC interfaces (FIC_0_AXI4_intiator).

6.2.4. TSN Peripheral Module [\(Ask a Question\)](#)

The following figure shows the SmartDesign of TSN peripheral module.

Figure 6-4. SmartDesign of TSN Peripheral Module



This SmartDesign shows the interconnection of the CoreTSN IP to the system-side and the line-side. The system-side interface is connected to the AXI4 interface, while the line-side interface is connected to the onboard PHY module through a serial data interface operating at 1 Gbps.

6.2.4.1. COREAXI4PROTOCONV [\(Ask a Question\)](#)

This design includes two instances of the COREAXI4PROTOCONV IP: MM2S on the transmit side and S2MM on the receive side. The MSS transmits and receives data from the fabric modules as a memory mapped device. The TSN IP accepts and outputs data in AXI stream format which has to be read/written to the memory as per the descriptors configured by the MSS. This functionality is attained by the S2MM and M2SS Logic.

Transmit Operation

This module implements the AXI4-Memory Mapped initiator read channel interface. This module reads the data to be transmitted from memory based on descriptors configured by the MSS. The retrieved data is then transmitted to the CoreTSN IP through the AXI4-Stream Initiator Interface (AXI4S_INIT).

Receive Operation

This module implements the AXI4-Memory Mapped initiator write channel interface. Data received on the AXI4-Stream target interface is written to memory according to descriptors configured by the MSS.

6.2.4.2. CoreAXI4InterConnect [\(Ask a Question\)](#)

This IP provides AXI4 and AXI4-Lite interfaces and connects them with MSS through FIC interface. The AXI4 Crossbar connects one or more AXI4 memory-mapped initiators to one or more memory mapped targets. The demo design includes two instances of this IP:

1. One instance of this IP connects the AXI4 initiator interfaces of the S2MM and M2SS module to the MSS AXI4 target interface.

- The other interface connects the MSS AXI4 initiator interface to the S2MM and M2SS AXI4 Lite target interface. This interface serves as the control interface of this IP's.
If the user want to add any fabric logic with AXI4 Lite Control interface or any memory mapped device with AXI4 interface it has to be connected to the MSS through this interconnect IP Block.

6.2.4.3. PKT_GENERATOR_CHECKER [\(Ask a Question\)](#)

To create the traffic in the channel the demo design uses this module which can generate both Ethernet and VLAN tagged Ethernet packets. The traffic generated by this module is used as the best-effort traffic in the design. If required, you can replace this block with their own traffic generator Logic. The packets generated by this module are in native format.

6.2.4.4. NAT_STR_CONV [\(Ask a Question\)](#)

This module is responsible for converting the incoming native format packets to the AXI4-Stream interface. The packet generator generates data in native format whereas the TSN IP accepts the input in AXI4-Stream format. The conversion from Native to AXI4-Stream is done by this logic.

If you replace the traffic generator capable of generating traffic in AXI4 stream format, this module can be removed from the design.

6.2.4.5. CoreAXI4SInterconnect [\(Ask a Question\)](#)

This IP connects the MSS AXI4-Stream interface and best-effort traffic generator stream interface to the TSN IP AXI4 target interface. The CoreInterconnect functions as a AXI4-Stream Switch. There are two AXI4-Stream Initiators, Priority traffic is generated from the MSS, while best-effort traffic is generated from the packet generator. The combined traffic is then connected to the AXI4S0_TRGT interface of the CORETSN IP.

6.2.4.6. CoreTSN [\(Ask a Question\)](#)

The CoreTSN IP is responsible for transmitting both priority traffic and best-effort traffic to the line side at scheduled intervals. Priority traffic is assigned a higher priority than best-effort traffic and is written to higher priority queues. Data from these queues is sent to the line side at scheduled times when the corresponding queue gates are open, as configured by the MSS in the Gate Control List (GCL) of the IP. The IP uses an APB interface for control and an AXI4-Stream interface for data transfer.

The TSN also supports preemption feature wherein the best-effort traffic is fragmented and the channel is made available for the high priority traffic, fragmentation of best-effort packet takes place when the high priority packet is scheduled on the line and the best-effort traffic transmission is ongoing. The rest of the best-effort traffic packet is sent as the other fragment after the high priority packet is transmitted.

6.2.4.7. Core1588 [\(Ask a Question\)](#)

Core1588 provides hardware support for the IEEE 1588 Precision Time Protocol (PTP) capable system. The core timestamps both the receipt and transmission of PTP event message frames. The MSS accesses these PTP timestamps through the APB interface of this IP.

6.2.4.8. CoreSGMII [\(Ask a Question\)](#)

CoreSGMII provides a solution for Ten-Bit Interface (TBI) on GMII based designs. CoreSGMII supports auto-negotiation between two link partners by exchanging their capabilities. This IP handles the encoding and decoding between GMII and SGMII formats, converting standard GMII signals to the 10-bit SGMII symbol format and vice versa. Since the external onboard PHY is SGMII PHY, this IP is required. If the user has GMII PHY, then this IP is not required.

6.2.4.9. PF_TX_PLL [\(Ask a Question\)](#)

The jitter attenuator PLL aligns the PTP clock with the local clock. The software synchronizes the initiator and target clocks by adjusting the TX PLL, specifically by modifying the integer and fractional divider values. The synchronized clock is then used for timestamping the PTP packets. The output clock from this module is used as the PTP clock in the design.

6.2.4.10. PF_IOD_CDR [\(Ask a Question\)](#)

This is a 1 Gbps data interface that operates with a 10:1 serialization ratio, providing 10-bit data on both the transmit and receive interfaces. The block converts parallel data into a serial data stream, which is transmitted through TX_P and TX_N. On the receive side, the interface includes a clock recovery block that generates the recovered clock (RX_CLK_R) and outputs the parallel data on RXD[9:0].

PF_IOD_CDR_CCC: This PF_IOD_CDR_CCC generates high-speed clocks with four phases: 0, 90, 180 and 270 degrees. These phase-shifted clocks are utilized by the PF_IOD_CDR module for clock recovery.

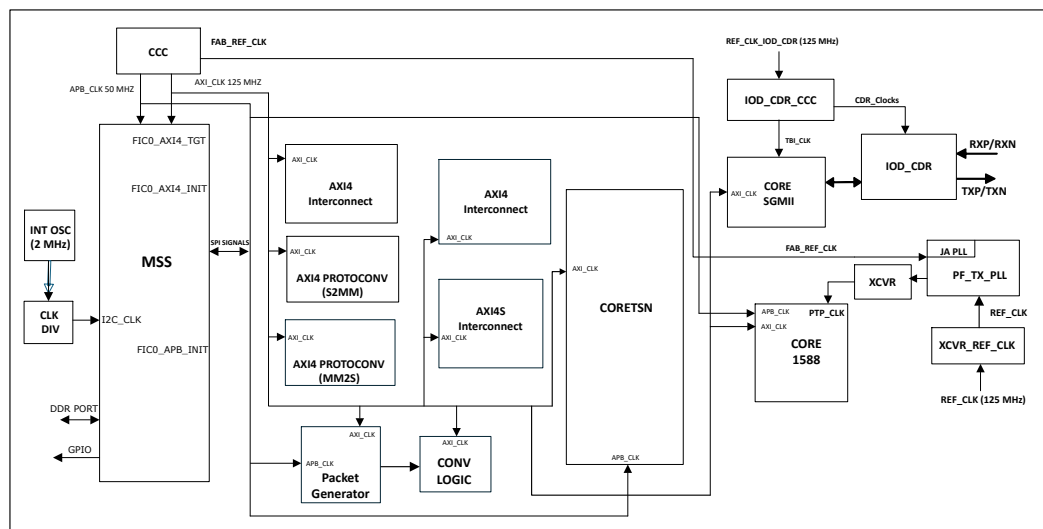
6.2.5. Clocking and Reset Structure [\(Ask a Question\)](#)

The following are the three clock domains in the demo design:

- Reference Onboard Clock 125 MHz: This clock is used for transceiver operation and as the reference clock for the CCC module. It generates the AXI interface clock of 125 MHz and the APB clock of 50 MHz. The transceiver module outputs a PTP_CLK signal at 125 MHz.
- Internal Oscillator: This oscillator generates a 2 MHz clock frequency. A clock divider divides this frequency by 2, resulting in a 1 MHz output. This 1 MHz output is used as the I²C clock.
- REF_CLK_IOD_CDR: The line side clocks in the design are derived from this reference clock using the IOD_CDR_CCC block. The TX clock generated by the IOD_CDR_CCC is used as the line side transmit clock and is connected CoreSGMII module. The IOD_CDR Block has clock recovery functionality and generates recovered clock and data. This recovered clock is used as the line side receive clock.

The following diagram shows the clocking structure of the TSN demo.

Figure 6-5. Clocking Structure



There are two resets used in the design. One is ARESETN_125 MHz, which resets the logic operating at a 125 MHz clock. The other is ARESETN_50 MHz, which resets the logic with the APB clock (50 MHz). The IOD_CDR_CCC PLL_Lock is used to reset the IOD_CDR block.

The following block diagram shows the reset structure of the TSN demo.

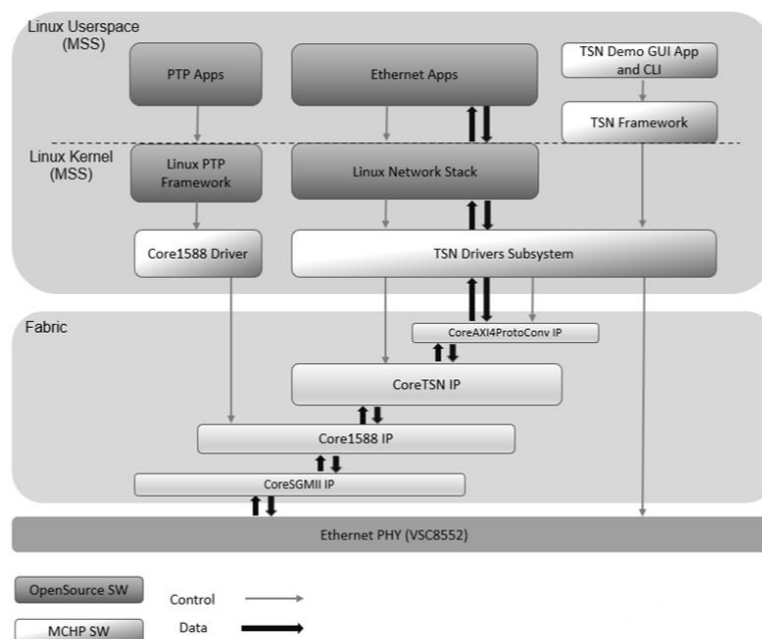
Figure 6-8. Memory Map of FIC_CONVERTER

Initiator/Bus/Bridge/Target	Offset Address	High Address	Range
FIC3 APB3 master			
CoreAPB3_CO_0:APB3mmaster			
FIC_CONVERTER:APBmslave2	0x0000_2000	0x0000_2FFF	4KB
FIC_CONVERTER:APBmslave3	0x0000_3000	0x0000_3FFF	4KB
FIC_CONVERTER:APBmslave16	0x0000_0000	0x0000_1FFF	8KB

6.3. Software Implementation [\(Ask a Question\)](#)

The following figure shows the software implementation.

Figure 6-9. Software Implementation



The TSN Demo Linux Software Stack comprises of kernel modules, frameworks, and user-space applications, operating within the MSS of PolarFire[®] SoC device. It integrates the necessary open-source components to meet specific functional and design requirements. The TSN Drivers Subsystem is responsible for configuring of CoreTSN and CoreAXI4ProtoConv IPs, Ethernet-PHY and the network device. It also abstracts these functionalities for access from Linux user space.

Within this subsystem, the Network Device Driver enables user-space applications such as `ethtool`, `ifconfig` and `iperf` through `ethx` (`eth1`) in this demonstration. It utilizes CoreAXI4ProtoConv to facilitate data transfer between the Fabric and MSS subsystems.

The CoreTSN Stack provides a user-space CLI for configuring the TSN features in accordance with IEEE 802.1 Qbv and Qbu standards. The CoreTSN Driver processes configuration requests from

user-space applications to set up the TSN hardware. TSN operation relies on PTP (IEEE 1588) to achieve precise clock synchronization at sub-microsecond or nanosecond accuracy between the initiator and the PolarFire SoC device. The Core1588 Driver serves as the PTP driver, accessible at `/dev/ptpx`, and can be utilized by the LinuxPTP (ptp4l) application. A custom PI Controller within the PTP Apps ensures the required level of clock synchronization.

In the TSN Demo Solution, IP's are configured with default values using dedicated applications. Configuration parameters are transmitted to the PolarFire SoC Fabric layer through Linux kernel frameworks and the drivers subsystem. Once TSN is enabled through the GUI, data packets are exchanged between the software and Fabric layers and the external network.

7. Programming the Device Using FlashPro Express [\(Ask a Question\)](#)

Connect a micro-USB to the Video Kit (J5) from the host PC and launch the FlashPro Express tool from the installation directory. Ensure that the jumper J28 is closed on the Video Kit. For detailed information about how to program the device using FlashPro Express, see [FlashPro Express User Guide](#).

8. Appendix A: TSN Demo for Windows Support [\(Ask a Question\)](#)

By default, Windows® does not offer built-in VLAN tagging configuration through its standard network settings. Many Ethernet drivers do not display VLAN options in their User Interface (UI). To enable VLAN tagging (802.1Q) on Windows, specific driver configurations or command-line steps are required.

Requirements for configuring the TSN demo for windows are:

- Windows machine (Laptop/Desktop) with USB Port
- Windows PowerShell with Admin privileges
- USB Ethernet Adaptor (Realtek)

To configure VLAN ID 13 and to set Priority Code Point (PCP) to 5 on the Ethernet adapter of a Windows machine, perform the following steps. It enables VLAN tagging, assigns a static IP (192.168.13.13), and sets the MTU to 1400 to accommodate VLAN headers. A QoS policy is also applied to mark traffic with PCP 5 for priority handling on VLAN-aware networks.

1. Open PowerShell as an administrator and run `ipconfig` command.

```
C:\Windows\System32\> ipconfig
```

Please note down all the interfaces listed. Ethernet and Ethernet 4 are listed, as shown in the following figure.

Figure 8-1. IP Config Command

```
PS C:\>
PS C:\> ipconfig

Windows IP Configuration

Ethernet adapter Ethernet:

    Connection-specific DNS Suffix  . : microchip.com
    Link-local IPv6 Address . . . . . : Fe80::d3fc:33fc:7e4a:83d4%12
    IPv4 Address. . . . . : 
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 

Ethernet adapter Ethernet 4:

    Connection-specific DNS Suffix  . : microchip.com
    Link-local IPv6 Address . . . . . : Fe80::4bea:6f20:d781:78af%11
    IPv4 Address. . . . . : 
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 

Wireless LAN adapter Wi-Fi:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . : microchip.com

Wireless LAN adapter Local Area Connection* 1:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . : 

Wireless LAN adapter Local Area Connection* 2:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . : 

Ethernet adapter Bluetooth Network Connection:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . : 
PS C:\>
```

2. Now connect the USB Ethernet Adapter into the laptop/desktop without having ethernet cable connected to the adapter.
3. Check the interfaces again by running `ipconfig` command. The new Ethernet port gets listed as "Ethernet 5", could be a different name in your case. Since the network is not connected yet, Media disconnected will be listed as shown in the following figure.

Figure 8-2. Verifying Ethernet Interface Name and IP Configuration

```

PS C:\> ipconfig

Windows IP Configuration

Ethernet adapter Ethernet 5:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . . . . . :

Ethernet adapter Ethernet:

Connection-specific DNS Suffix . . . . . : microchip.com
Link-local IPv6 Address . . . . . : fe80::d3fc:33fc:7e4a:83d4%12
IPv4 Address. . . . . : 
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . :

Ethernet adapter Ethernet 4:

Connection-specific DNS Suffix . . . . . : microchip.com
Link-local IPv6 Address . . . . . : fe80::4bea:6f20:d781:78af%11
IPv4 Address. . . . . : 
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . :

Wireless LAN adapter Wi-Fi:

```

4. Insert the network cable into the USB.
5. If you enable the DHCP, an IP Address will be assigned to this adapter automatically, as shown in the figure.

Figure 8-3. Automatic IP Assignment through DHCP on Adapter

```

Administrator: Windows PowerShell
PS C:\Windows\System32\WindowsPowerShell\v1.0> ipconfig

Windows IP Configuration

Ethernet adapter Ethernet:

Connection-specific DNS Suffix . . . . . : microchip.com
Link-local IPv6 Address . . . . . : fe80::d3fc:33fc:7e4a:83d4%12
IPv4 Address. . . . . : 
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . :

Ethernet adapter Ethernet 4:

Connection-specific DNS Suffix . . . . . : microchip.com
Link-local IPv6 Address . . . . . : fe80::e974:62b1:88a7:1338%10
IPv4 Address. . . . . : 
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . :

Ethernet adapter Ethernet 5:

Connection-specific DNS Suffix . . . . . : microchip.com
Link-local IPv6 Address . . . . . : fe80::e974:62b1:88a7:1338%10
IPv4 Address. . . . . : 192.168.13.13
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . :

Wireless LAN adapter Local Area Connection* 1:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . . . . . :

Wireless LAN adapter Local Area Connection* 2:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . . . . . :

Wireless LAN adapter Wi-Fi:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . . . . . : microchip.com

Ethernet adapter Bluetooth Network Connection:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . . . . . :

PS C:\Windows\System32\WindowsPowerShell\v1.0>

```

6. To create VLAN/PCP, run the listed commands in PowerShell which configures the Ethernet 5, for PCP 5 and VLAN ID 13.

Table 8-1. PowerShell Commands

Task	Command	Remarks
Check Advance Properties	Get-NetAdapterAdvancedProperty -Name "Ethernet 5"	—
Enable VLAN/PCP Tagging	Set-NetAdapterAdvancedProperty -Name "Ethernet 5" -RegistryKeyword "*PriorityVLANTag" -RegistryValue 3	Enables 802.1Q VLAN tagging and prioritization by setting the registry flag for VLAN/PCP tagging to 3.
Set VLAN ID 13	Set-NetAdapterAdvancedProperty -Name "Ethernet 5" -RegistryKeyword "RegVlanID" -RegistryValue 13	Assigns VLAN ID 13 to the Ethernet interface, marking traffic from this interface with the specified VLAN tag.
Assign PCP	New-NetQoSPolicy -Name "VLAN_Priority5" -AppPathNameMatchCondition "*" -PriorityValue8021Action 5	Creates a QoS policy to tag all outbound packets (AppPathNameMatchCondition "*") with a PCP value of 5, indicating a medium-high traffic priority.
Assign IP Address to VLAN	New-NetIPAddress -InterfaceAlias "Ethernet 5" -IPAddress "192.168.13.13" -PrefixLength 24	Sets a static IP address (192.168.13.13/24) on the interface after enabling VLAN tagging.

Table 8-1. PowerShell Commands (continued)

Task	Command	Remarks
Set MTU to 1400	netsh interface ipv4 set subinterface interface="Ethernet 5" mtu=1400 store=persistent	Reduces the MTU to 1400 bytes to accommodate for VLAN headers (which add 4 bytes), preventing fragmentation in VLAN-tagged packets.

- To check advance properties, run the following command:

```
Get-NetAdapterAdvancedProperty -Name "Ethernet 5"
```

Priority & VLAN status is **Disabled**, with registered value as "0", as shown in the following figure.

Figure 8-4. Checking Priority & VLAN Status on Ethernet 5 Adapter (Disabled)

```
PS C:\Windows\System32> Get-NetAdapterAdvancedProperty -Name "Ethernet 5"
Name      DisplayName      DisplayValue      RegistryKeyword  RegistryValue
-----      -
Ethernet 5  IPv4 Checksum offload  Rx & Tx Enabled  *IPChecksum... [3]
Ethernet 5  Jumbo Frame        Disabled          *JumboPacket    [1514]
Ethernet 5  Large Send offload v2 (IPv4) Enabled          *LsoV2IP4       [1]
Ethernet 5  Large Send offload v2 (IPv6) Enabled          *LsoV2IPv6      [1]
Ethernet 5  ARP Offload        Enabled          *PMARPOffload  [1]
Ethernet 5  NS Offload         Enabled          *PMNSOffload    [1]
Ethernet 5  Priority & VLAN    Priority & VLAN Disabled *PriorityVLAN... [0]
Ethernet 5  Recv Segment Coalescing (IPv4) Enabled        *RscIPv4        [1]
Ethernet 5  Recv Segment Coalescing (IPv6) Enabled        *RscIPv6        [1]
Ethernet 5  Speed & Duplex     Auto Negotiation *SpeedDuplex    [0]
Ethernet 5  TCP Checksum offload (IPv4) Rx & Tx Enabled  *TCPChecksum... [3]
Ethernet 5  TCP Checksum offload (IPv6) Rx & Tx Enabled  *TCPChecksum... [3]
Ethernet 5  UDP Checksum offload (IPv4) Rx & Tx Enabled  *UDPChecksum... [3]
Ethernet 5  UDP Checksum offload (IPv6) Rx & Tx Enabled  *UDPChecksum... [3]
Ethernet 5  Wake on Magic Packet Enabled        *WakeOnMagic... [1]
Ethernet 5  Wake on pattern match Enabled        *WakeOnPattern... [1]
Ethernet 5  Network Address    6             NetworkAddress  [---]
Ethernet 5  Receive UBSs      3             PendingTrans... [3]
Ethernet 5  Transmit UBSs     16           PendingTrans... [3]
Ethernet 5  Receive Buffers   0             RegVlanID       [0]
Ethernet 5  Transmit Buffers  18           Transm1 Buff... [18]
Ethernet 5  Wake on Link change Enabled        WakeOnLnkCh... [1]
Ethernet 5  WoL & Shutdown Link Speed 10 Mbps First WoLShutdown... [0]
```

- To enable VLAN/PCP tagging, run the following command:

```
Set-NetAdapterAdvancedProperty -Name "Ethernet 5" -RegistryKeyword "*PriorityVLANTag" -RegistryValue 3
```

Priority & VLAN status is changed to **Enabled**, with registered value as "3", as shown in the following figure.

Figure 8-5. Enabling Priority & VLAN on Ethernet 5 Adapter

```
PS C:\Windows\System32> Set-NetAdapterAdvancedProperty -Name "Ethernet 5" -RegistryKeyword "*PriorityVLANTag" -RegistryValue 3
PS C:\Windows\System32> Get-NetAdapterAdvancedProperty -Name "Ethernet 5"
Name      DisplayName      DisplayValue      RegistryKeyword  RegistryValue
-----      -
Ethernet 5  IPv4 Checksum offload  Rx & Tx Enabled  *IPChecksum... [3]
Ethernet 5  Jumbo Frame        Disabled          *JumboPacket    [1514]
Ethernet 5  Large Send offload v2 (IPv4) Enabled          *LsoV2IP4       [1]
Ethernet 5  Large Send offload v2 (IPv6) Enabled          *LsoV2IPv6      [1]
Ethernet 5  ARP Offload        Enabled          *PMARPOffload  [1]
Ethernet 5  NS Offload         Enabled          *PMNSOffload    [1]
Ethernet 5  Priority & VLAN    Priority & VLAN Enabled *PriorityVLAN... [3]
Ethernet 5  Recv Segment Coalescing (IPv4) Enabled        *RscIPv4        [1]
Ethernet 5  Recv Segment Coalescing (IPv6) Enabled        *RscIPv6        [1]
Ethernet 5  Speed & Duplex     Auto negotiation *SpeedDuplex    [0]
Ethernet 5  TCP Checksum offload (IPv4) Rx & Tx Enabled  *TCPChecksum... [3]
Ethernet 5  TCP Checksum offload (IPv6) Rx & Tx Enabled  *TCPChecksum... [3]
Ethernet 5  UDP Checksum offload (IPv4) Rx & Tx Enabled  *UDPChecksum... [3]
Ethernet 5  UDP Checksum offload (IPv6) Rx & Tx Enabled  *UDPChecksum... [3]
Ethernet 5  Wake on Magic Packet Enabled        *WakeOnMagic... [1]
Ethernet 5  Wake on pattern match Enabled        *WakeOnPattern... [1]
Ethernet 5  Network Address    6             NetworkAddress  [6]
Ethernet 5  Receive UBSs      3             PendingTrans... [3]
Ethernet 5  Transmit UBSs     16           PendingTrans... [16]
Ethernet 5  Receive Buffers   0             RegVlanID       [0]
Ethernet 5  Transmit Buffers  18           Transm1 Buff... [18]
Ethernet 5  Wake on Link change Enabled        WakeOnLnkCh... [1]
Ethernet 5  WoL & Shutdown Link Speed 10 Mbps First WoLShutdown... [0]
```

- To set VLAN ID 13, run the following command:

```
Set-NetAdapterAdvancedProperty -Name "Ethernet 5" -RegistryKeyword "RegVlanID" -RegistryValue 13
```

VLAN ID is changed to 13, with RegistryValue as "13", as shown in the following figure.

Figure 8-6. Change VLAN ID to 13 on Ethernet 5

```

PS C:\Windows\System32> Set-NetAdapterAdvancedProperty -Name "Ethernet 5" -RegistryKeyword "RegVlanID" -RegistryValue 13
PS C:\Windows\System32> Get-NetAdapterAdvancedProperty -Name "Ethernet 5"

```

Name	DisplayName	DisplayValue	RegistryKeyword	RegistryValue
Ethernet 5	IPv4 Checksum Offload	Rx & Tx Enabled	*IPChecksum...	{3}
Ethernet 5	Jumbo Frame	Disabled	*JumboFrame	{1514}
Ethernet 5	Large Send offload v2 (IPv4)	Enabled	*LSOv2IPv4	{1}
Ethernet 5	Large Send offload v2 (IPv6)	Enabled	*LSOv2IPv6	{1}
Ethernet 5	ARP Offload	Enabled	*PMARPOffload	{1}
Ethernet 5	NS Offload	Enabled	*PMNSOffload	{1}
Ethernet 5	Priority & VLAN	Priority & VLAN Enabled	*PriorityVLA...	{13}
Ethernet 5	Recv Segment Coalescing (IPv4)	Enabled	*RscIPv4	{1}
Ethernet 5	Recv Segment Coalescing (IPv6)	Enabled	*RscIPv6	{1}
Ethernet 5	Speed & Duplex	Auto Negotiation	*SpeedDuplex	{0}
Ethernet 5	TCP Checksum Offload (IPv4)	Rx & Tx Enabled	*TCPChecksum...	{3}
Ethernet 5	TCP Checksum Offload (IPv6)	Rx & Tx Enabled	*TCPChecksum...	{3}
Ethernet 5	UDP Checksum Offload (IPv4)	Rx & Tx Enabled	*UDPChecksum...	{3}
Ethernet 5	UDP Checksum Offload (IPv6)	Rx & Tx Enabled	*UDPChecksum...	{3}
Ethernet 5	Wake on Magic Packet	Enabled	*WakeOnMagic...	{1}
Ethernet 5	Wake on pattern match	Enabled	*WakeOnPattern	{1}
Ethernet 5	Network Address	Enabled	*NetworkAddress	{-}
Ethernet 5	Receive URBS	6	*ReceiveURBS	{6}
Ethernet 5	Transmit URBS	6	*TransmitURBS	{6}
Ethernet 5	Receive Buffers	16	*ReceiveBuffe...	{16}
Ethernet 5	VLAN ID	13	*RegVlanID	{13}
Ethernet 5	Transmit Buffers	18	*TransmitBuff...	{18}
Ethernet 5	Wake on Link Change	Enabled	*WakeOnLinkCh...	{1}
Ethernet 5	Wake & Shutdown Link Speed	10 Mbps First	*WoShutDown...	{0}

- To assign PCP, run the following command as shown in the following figure.

```

New-NetQoSPolicy -Name "VLAN_Priority5" -AppPathNameMatchCondition "*"
-PriorityValue8021Action 5

```

Figure 8-7. Creating QoS Policy with VLAN Priority 5

```

PS C:\Windows\System32>
PS C:\Windows\System32>
PS C:\Windows\System32> New-NetQoSPolicy -Name "VLAN_Priority5" -AppPathNameMatchCondition "*" -PriorityValue8021Action 5

```

Name	: VLAN_Priority5
Owner	: Group Policy (Machine)
NetworkProfile	: All
Precedence	: 127
AppPathName	: *
ObjObject	: *
PriorityValue	: 5

Note: Priority Value has been set to "5".

- To assign IP address to VLAN, run the following command as shown in the following figure.

```

New-NetIPAddress -InterfaceAlias "Ethernet 5" -IPAddress "192.168.13.13" -PrefixLength 24

```

Figure 8-8. Assigning Static IP to Ethernet 5 Interface

```

PS C:\Windows\System32>
PS C:\Windows\System32> New-NetIPAddress -InterfaceAlias "Ethernet 5" -IPAddress "192.168.13.13" -PrefixLength 24

```

IPAddress	: 192.168.13.13
InterfaceIndex	: 10
InterfaceAlias	: Ethernet 5
AddressFamily	: IPv4
Type	: Unicast
PrefixLength	: 24
PrefixOrigin	: Manual
SuffixOrigin	: Manual
AddressState	: Tentative
ValidLifetime	: ...
PreferredLifetime	: ...
SkipAsSource	: False
PolicyStore	: ActiveStore
IPAddress	: 192.168.13.13
InterfaceIndex	: 10
InterfaceAlias	: Ethernet 5
AddressFamily	: IPv4
Type	: Unicast
PrefixLength	: 24
PrefixOrigin	: Manual
SuffixOrigin	: Manual
AddressState	: Invalid
ValidLifetime	: ...
PreferredLifetime	: ...
SkipAsSource	: False
PolicyStore	: PersistentStore

Note: Address has been set to "Ethernet 5".

- To set MTU to 1400, run the following command as shown in the following figure.

```

netsh interface ipv4 set subinterface interface="Ethernet 5" mtu=1400 store=persistent

```

Figure 8-9. Setting MTU on Ethernet 5 Interface

```

PS C:\Windows\System32>
PS C:\Windows\System32> netsh interface ipv4 set subinterface interface="Ethernet 5" mtu=1400 store=persistent
PS C:\Windows\System32> netsh interface ipv4 show subinterfaces

```

MTU	MediasenseState	Bytes In	Bytes Out	Interface
4294967295	1	0	137634	Loopback Pseudo-Interface 1
1500	1	340160573	89298120	Wi-Fi
1500	1	5180336147	570628120	Ethernet Connection 1
1500	0	0	0	Local Area Connection 3
1500	0	0	0	Bluetooth Network Connection
1400	1	238747444	125247648	Ethernet 5
1500	0	0	2414	Wi-Fi Shown...

Note: Check the MTU for Ethernet 5 has been updated to 1400.

- Run the ipconfig again and verify if the address 192.168.13.13 has been assigned to the "Ethernet 5" adaptor, as shown in the following figure.

Figure 8-10. Verifying IP Assignment on Ethernet 5 Adapter

```

Administrator: Windows PowerShell
PS C:\Windows\System32> ipconfig
Windows IP Configuration

Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . : microchip.com
   Link-local IPv6 Address . . . . . : fe80::d11e:33fc:7e4a:83d4%12
   IPv4 Address. . . . . : 192.168.13.2
   Subnet Mask . . . . . : 255.255.255.0
   Default Gateway . . . . . : fe80::5959

Ethernet adapter Ethernet 4:

   Connection-specific DNS Suffix  . : microchip.com
   Link-local IPv6 Address . . . . . : fe80::d4ea:6f20:d781:78af%11
   IPv4 Address. . . . . : 192.168.13.1
   Subnet Mask . . . . . : 255.255.255.0
   Default Gateway . . . . . : fe80::5959

Ethernet adapter Ethernet 5:

   Connection-specific DNS Suffix  . : microchip.com
   Link-local IPv6 Address . . . . . : fe80::e974:62b1:88a2:1380%10
   IPv4 Address. . . . . : 192.168.13.13
   Subnet Mask . . . . . : 255.255.255.0
   Default Gateway . . . . . : fe80::5959

Wireless LAN adapter Local Area Connection* 1:

   Media State . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 2:

   Media State . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Wi-Fi:

   Media State . . . . . : Media disconnected
   Connection-specific DNS Suffix  . : microchip.com

Ethernet adapter Bluetooth Network Connection:

   Media State . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
PS C:\Windows\System32>

```

- To verify the connection to the PolarFire[®] SoC Video Kit, execute the command `ping 192.168.13.2`. Ensure that you receive a reply from the video kit, as shown in the following figure.

Figure 8-11. Ping Command to Verify Connection with PolarFire SoC Video Kit

```

PS C:\Windows\System32>
PS C:\Windows\System32>
PS C:\Windows\System32>
PS C:\Windows\System32> ping 192.168.13.2

Pinging 192.168.13.2 with 32 bytes of data:
Reply from 192.168.13.2: bytes=32 time<1ms TTL=64
Reply from 192.168.13.2: bytes=32 time<1ms TTL=64
Reply from 192.168.13.2: bytes=32 time=1ms TTL=64
Reply from 192.168.13.2: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.13.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 1ms, Average = 0ms
PS C:\Windows\System32>

```

This confirms that communication between the Video Kit and the Windows machine is established and operational and to proceed further, see the [Demo at a Glance](#) section.

The set up is tested on the listed source of OS:

- OS Name: Microsoft Windows 11 Enterprise
- OS Version: 10.0.22631 N/A Build 22631

9. Appendix B: Linux® Host Setup for Serial Communication [\(Ask a Question\)](#)

To connect your Ubuntu PC to the Video Kit using a USB-to-UART cable and Minicom, perform the following steps:

1. Identify the Serial Port:
 - a. Connect the Video Kit (J12) to the Ubuntu PC through the USB-to-UART cable.
 - b. Open a Terminal.
 - c. Before plugging in, check for any connected serial devices by running the `ls /dev/ttyUSB*` command (It is normal, if no devices are listed).
 - d. Plug in the cable and wait a few seconds. Then, run the command again:
`ls /dev/ttyUSB*` You must see a new device, such as `/dev/ttyUSB0`.
2. Install Minicom (if it is not already installed), using the following command:

```
sudo apt update
sudo apt install minicom
```

3. To connect and configure with Minicom, you can use one of the two following approaches:
 - **Single Terminal, Sequential Method:**
 - i. Start Minicom setup with the command `sudo minicom -s`.
 - ii. In the **setup** menu, configure the serial device by selecting following options:
 1. A: Set Serial Device (start with, such as `/dev/ttyUSB0`)
 2. E: Set Baud Rate/Parity/Data Bits (use 115200 8N1)
 3. F: Hardware Flow Control as "No"
 4. G: Software Flow Control as "No"
 - iii. Save as default (**Save setup as dfl**), then exit setup.
 - iv. Launch Minicom using `sudo minicom` command.
 - v. Look for a **root login prompt**, an output from the Video Kit. If the prompt is not visible, exit Minicom (**Ctrl+A**, then **Z**, then **X**), repeat steps from a-d using the next device (`/dev/ttyUSB1`, `/dev/ttyUSB2`, `/dev/ttyUSB3`) until the correct port is found.
 - **Multiple Terminals Method:** Open four separate terminals and run each command in a different terminal.
 - `sudo minicom -b 115200 -o -D /dev/ttyUSB0`
 - `sudo minicom -b 115200 -o -D /dev/ttyUSB1`
 - `sudo minicom -b 115200 -o -D /dev/ttyUSB2`
 - `sudo minicom -b 115200 -o -D /dev/ttyUSB3`
- Observe each terminal; use the one where you see the Video Kit's output or login prompt.
4. To exit Minicom, press **Ctrl+A**, then **Z** and then **X**.

Once communication between the Video Kit and Linux is established, see the [Instructions to Run the Demo on Linux](#) section.

10. Appendix C: Verifying PTP Hardware Clock (PHC) Support [\(Ask a Question\)](#)

To confirm if the NIC supports gPTP, check for a PHC using `ethtool`. Execute the `ethtool -T` command followed by the network interface name (for example, `eno1`), and then pipe the output to `grep PTP`: `ethtool -T [your_interface_name] | grep PTP`

The following example shows how to verify PHC support:

Run `ethtool -T eno1 | grep PTP` and look for a line in the output similar to: `PTP Hardware Clock: x` (where `x` is a numerical value like 0, 1 or 2). The presence of this line indicates that NIC has a dedicated PHC, which is essential for gPTP's hardware timestamping and precise synchronization. If the **PTP Hardware Clock** is present, it confirms that NIC card supports gPTP. For information about PTP connection, see the [Step 8](#) of "Instructions to Run the Demo on Linux" section.

11. Appendix D: Design Creation Using TCL Scripts [\(Ask a Question\)](#)

TCL scripts are provided in the [GitHub Repository](#). Download the repository `polarfire-soc-video-kit-reference-design` and unzip it into a local directory.

To run the TCL, perform the following steps:

1. Launch the Libero SoC software.
2. Click **Project** > **Execute Script.....**
3. Click **Browse** and in the downloaded `polarfire-soc-video-kit-reference-design` directory, select `MPFS_VIDEO_KIT_REFERENCE_DESIGN.tcl` and pass the argument `TSN SYNTHESIZE PLACEROUTE VERIFY_TIMING HSS_UPDATE SPIFLASH_DATA EXPORT_FPE`.
4. Click **Run**.

After successful execution of TCL script, Libero project is created within `polarfire-soc-video-kit-reference-design` directory. For more information about TCL scripts, see [GitHub Repository](#). For more details on TCL commands, see [Libero SoC TCL Command Reference Guide](#). Contact [Technical Support](#) for any queries encountered when running the TCL script.

12. Glossary [\(Ask a Question\)](#)

Following are the list of terms and definitions used in the document.

Table 12-1. Terms and Definitions

Term	Definition
APB	Advanced Peripheral Bus
AXI4	Advanced eXtensible Interface 4
FPGA	Field-Programmable Gate Array
GUI	Graphical User Interface
MM2S	Memory Mapped to Stream
MSS	Microprocessor Subsystem
PTP	Precision Time Protocol
RTL	Register-Transfer Level
S2MM	Stream to Memory Mapped
SGMII	Serial Gigabit Media Independent Interface
SoC	System on Chip
TCL	Tool Command Language
TSN	Time-Sensitive Networking
VLAN	Virtual Local Area Network

13. **Revision History** [\(Ask a Question\)](#)

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

Revision	Date	Description
A	11/2025	Initial Revision

Microchip FPGA Support

Microchip FPGA products group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, and worldwide sales offices. Customers are suggested to visit Microchip online resources prior to contacting support as it is very likely that their queries have been already answered.

Contact Technical Support Center through the website at www.microchip.com/support. Mention the FPGA Device Part number, select appropriate case category, and upload design files while creating a technical support case.

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

- From North America, call **800.262.1060**
- From the rest of the world, call **650.318.4460**
- Fax, from anywhere in the world, **650.318.8044**

Microchip Information

Trademarks

The “Microchip” name and logo, the “M” logo, and other names, logos, and brands are registered and unregistered trademarks of Microchip Technology Incorporated or its affiliates and/or subsidiaries in the United States and/or other countries (“Microchip Trademarks”). Information regarding Microchip Trademarks can be found at <https://www.microchip.com/en-us/about/legal-information/microchip-trademarks>.

ISBN: 979-8-3371-2298-4

Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. Contact your local Microchip sales office for additional support or, obtain additional support at www.microchip.com/en-us/support/design-help/client-support-services.

THIS INFORMATION IS PROVIDED BY MICROCHIP “AS IS”. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL LOSS, DAMAGE, COST, OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP’S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION.

Use of Microchip devices in life support and/or safety applications is entirely at the buyer’s risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip products:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip products are strictly prohibited and may violate the Digital Millennium Copyright Act.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is “unbreakable”. Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.