Time-Sensitive Networking (TSN) for Industrial Automation Using Open Platform Communications Unified Architecture (OPC UA)

Introduction

In the swiftly advancing realm of industrial automation, the integration of sophisticated communication protocols and real-time data transmission is crucial. Using the existing Ethernet backbone for real-time data transfers, enhanced by Time-Sensitive Networking (TSN) features, is transforming industrial automation and enabling smarter, more efficient and highly responsive manufacturing processes.

This whitepaper introduces an innovative approach to industrial automation by combining the Open Platform Communications Unified Architecture (OPC UA) with TSN, leveraging low-power Microchip PolarFire® SoC FPGAs. OPC UA, renowned for its platform-independence, robust and secure data exchange capabilities enhanced with TSN, ensures deterministic data delivery with minimal latency and network synchronization. This significantly boosts data integrity and improves the overall responsiveness of industrial systems.

The proposed system architecture illustrates how TSN's capabilities such as time synchronization, traffic scheduling, preemption and resource reservation are effectively used to enhance the performance of OPC UA-based applications in industrial settings. It highlights the impact of enabling and disabling TSN features on system metrics such as latency, jitter and bandwidth utilization. The application aims to control an industrial motor using deterministic communication alongside video data transfers over Ethernet.

This whitepaper emphasizes the potential of TSN in augmenting OPC UA applications, paving the way for more robust and efficient industrial automation systems that capitalize on the current Ethernet infrastructure.

In the rapidly advancing field of industrial automation, the demand for sophisticated communication protocols and real-time data transmission is critical. Traditional industrial communication protocols often fail to meet the requirements for high bandwidth, low latency and deterministic data delivery needed by modern automation systems. To address these challenges, leveraging the existing Ethernet infrastructure, enhanced by Time-Sensitive Networking (TSN) features, is proving to be a transformative solution.

TSN is a set of IEEE® standards (1), (2), (3) , (4), (5), (6) and (7) that extends Ethernet capabilities to support real-time, deterministic communication. TSN offers features such as time synchronization (8), traffic scheduling (1), (3) and (4), preemption (5) and resource reservation, which are essential for ensuring reliable and timely data delivery in industrial environments. These capabilities make TSN an ideal solution for applications requiring high precision and low latency, such as robotics, factory automation and process control.

The integration of TSN with the Open Platform Communications Unified Architecture (OPC UA) further enhances the robustness and efficiency of industrial automation systems. OPC UA is a platform-independent, secure communication protocol widely adopted in industrial automation for its ability to facilitate seamless data exchange between heterogeneous systems. Combining OPC UA with TSN (15) and (16) enables deterministic data delivery with minimal latency and improved network synchronization, thereby enhancing data integrity and system responsiveness.

Microchip's PolarFire SoC FPGA plays a crucial role in this integration. These low-power, high-performance System-on-Chips (SoCs) are designed to support both TSN and OPC UA (15) and (16), providing a versatile platform for developing advanced industrial automation solutions. The PolarFire SoC enables the partitioning of user designs between the Microprocessor Subsystem (MSS) and the Field Programmable Gate Array (FPGA) fabric, allowing for efficient implementation of complex communication protocols and real-time data processing.

This whitepaper explores the innovative approach of combining TSN with OPC UA using a Microchip PolarFire SoC (16). It delves into the system architecture, highlighting how TSN features are effectively enhances the performance of OPC UA based applications in industrial settings. The application scenario focuses on controlling an industrial motor using deterministic communication alongside video data transfers over the Ethernet, showcasing the practical benefits of this integration.

The following is the list of the key features of TSN:

- Time Synchronization: Ensures all devices on the network share a common understanding of time, which is crucial for coordinating actions and data exchange (8) and (15).
- Traffic Scheduling: Enables the prioritization of critical data streams, ensuring their timely delivery even in congested networks (3) , (4) and (10).
- Low Latency: Minimizes the delay in data transmission, which is vital for real-time applications (11).
- Frame Preemption: Allows high-priority traffic to interrupt lower priority traffic, ensuring critical data are transmitted with minimal delay. This feature is defined in the IEEE 802.1Qbu and IEEE 802.3br standards (5) and (6).
- Reliability: Enhances network robustness through redundancy and fault-tolerant mechanisms (7).

OPC Unified Architecture

OPC UA is a machine-to-machine communication protocol specifically designed for industrial automation. It offers a standardized and secure method for various machines and systems to communicate, irrespective of the manufacturer, platform or operating system.

The following is the list of key features of OPC UA:

- Client-Server Architecture: The OPC UA protocol operates on a client-server architecture, where the client initiates communication and the server supplies the requested data. This communication is organized around a set of predefined services that outline the interactions between clients and servers. These services encompass browsing, reading, writing, subscribing and monitoring. Additionally, OPC UA incorporates various security mechanisms, such as encryption, authentication and authorization, to ensure secure communication between clients and servers (16).
- Platform Independence: One of the primary benefits of OPC UA is its platform independence, enabling its use across various operating systems, programming languages and hardware platforms. This versatility makes it an ideal choice for industrial automation systems, where seamless communication between different machines and systems is essential. OPC UA protocol boasts several features that enhance its functionality and adaptability for industrial automation. Notably, it can manage complex data types, supporting a broad spectrum of data, including numerical values, strings, arrays and structures (16).
- Complex Data Types: OPC UA supports a broad range of data types, including numerical data, strings arrays and structures. Additionally, it supports user-defined data types, allowing industrial automation systems to create custom data structures tailored to their specific requirements (16).
- Time Synchronization: A key feature of OPC UA is its support for time synchronization. In industrial automation, precise time synchronization is essential to ensure that various machines and systems operate in sync with each other. OPC UA offers a mechanism for time synchronization between clients and servers, guaranteeing accurate and timely data exchange (15) and (16).
- Security Mechanisms: OPC UA consists of two layers, each incorporating built-in security features. The communication layer establishes a secure channel between the client and server through encryption, signatures and digital certificates. The application layer is responsible for user authentication and permission verification. While security mechanisms can impact application performance, OPC UA allows you to enable or disable these features to achieve a balance between performance and security requirements (17).
- Built-in Diagnostics and Error Handling: OPC UA offers integrated diagnostics and error handling features. These functionalities facilitate the detection and diagnosis of errors and faults in communication between clients and servers within industrial automation systems. Consequently, this helps to reduce downtime and ensures the efficient operation of the system.

Integrating OPC UA and TSN

The integration of OPC UA with TSN brings together the strengths of both technologies, creating a robust communication framework for industrial and space applications (15) and (16). The following is a list of the combined benefits to use TSN:

- Enhanced Interoperability: OPC UA's platform independence and TSN's standardized Ethernet protocols ensure seamless communication between different systems and devices.
- Improved Real-Time Performance: TSN's deterministic capabilities complement OPC UA's time synchronization features, ensuring timely and accurate data exchange.
- Increased Reliability and Security: The security mechanisms of OPC UA, combined with TSN's fault-tolerance and resiliency, provide a secure and reliable communication infrastructure.
- Scalability and Flexibility: The combined solution can be scaled to accommodate various industrial and space applications, from small-scale automation systems to large, complex networks.

Application

The following is the list of applications supported by TSN:

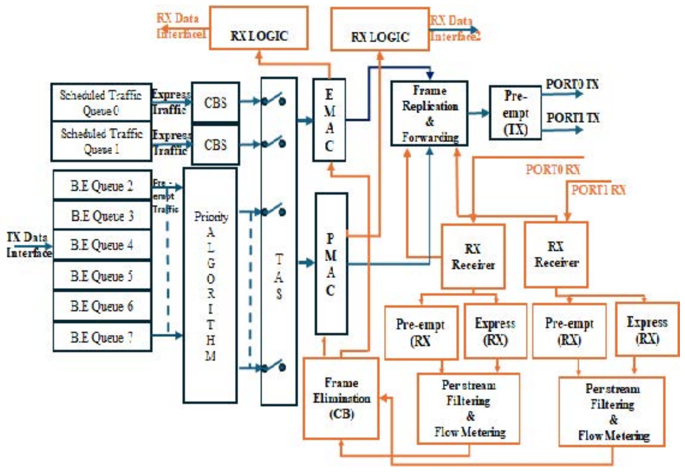
- Industrial Automation: The combination of OPC UA and TSN is ideal for industrial automation systems, where different machines and systems need to communicate seamlessly and operate in sync.
- Space Applications: TSN's fault-tolerance and synchronization capabilities make it suitable for mission-critical space applications, such as satellite communication and space exploration missions

Table of Contents

1. TSN Implementation

The following figure shows the TSN system architecture proposed in this whitepaper. The TSN functionality is implemented as Fabric Logic, and the software is implemented using the MSS of the Polarfire SoC.

Figure 1-1. TSN Fabric Logic



1.1. Hardware Implementation

Figure 1-1 shows the blocks of the TSN. The architecture has one transmit and two receive interfaces on the system side and two ports at the line side.

The following is the list of key components and functions of the TSN architecture:

- Queuing: There are eight queues implemented, and each queue supports two stream identifications (ID). Depending upon the stream parameters the incoming packet is loaded into the corresponding queue. There are two queues which are used for scheduled traffic, and the rest of the queues are meant for the best effort traffic. This ensures that packets are processed in accordance with their priority levels, facilitating efficient and orderly network traffic management (13) and (14).
- Traffic scheduling: Time-Aware Shaper (TAS) and Credit-Based Shaper (CBS) (1) are two algorithms used for traffic scheduling, with time scheduling playing a crucial role in queue selection for packet transmission. The Gate Control List (GCL) specifies time slots and the status of queues that are open for those specific times. Packets loaded into priority queues are processed and transmitted according to the gate control states. The GCL is executed in a predetermined order and repeats continuously for each cycle time. Within a specific GCL, multiple queues can be enabled for packet transmission. Express traffic and preemption traffic are routed to the Express Medium Access Control (EMAC) and Preemption MAC (PMAC) respectively, with express traffic always being given the highest priority for transmission (10), (11), (12) and (14).
- Medium Access Control (MAC): EMAC/PMAC is a fully featured 1000 Mbps MAC with a standard Gigabit Media Independent Interface (GMII) interface. It is capable of operating at 1000 Mbps in full-duplex mode and complies with IEEE® 802.3x, which means it ignores both carrier and collisions. After each packet transmission or reception statistics are collected for analysis.
- Preemption: In the transmit path, preemption packets are divided into fragments whenever express traffic is available for transmission. After transmitting the minimum size of a fragment, the system continuously checks for the availability of express packets. If any express packets are found, the system ends the current fragment transmission and transmits the express traffic. In

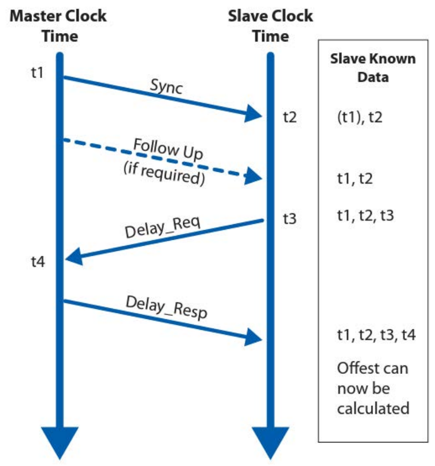
standard TSN communication, critical data is assigned to the EMAC channel, while best-effort traffic is directed to the PMAC channel (9).

The receive path includes decoder routines for both preemption and express packets. The preemption decoder reassembles fragments from multiple packets to form a full packet. If any errors are detected, the corresponding packet is discarded. All preemption packets are forwarded to the PMAC channel, while express packets are directly sent to the EMAC channel.

- Per stream filtering and flow metering: The filtering functionality is designed to manage and control the flow of data based on stream IDs. This mechanism employs a flow meter to monitor the amount of data received for each stream. The flow meters track the number of bytes received and compares it against a predefined threshold. If the data for a particular stream exceeds this threshold, the flow meter triggers a filtering action, resulting in the discarding of any additional packets for that stream. This ensures that the network can maintain its Quality of Service (QoS) by preventing any single stream from consuming excessive bandwidth, thereby protecting the performance and reliability of other streams within the network (2), (18) and (19).
- Frame replication and elimination: The current implementation duplicates the outgoing packet and forwards them through different network paths to ensure that at least one copy reaches the destination, even if some paths experience failures. The sequence numbers embedded in the packets allow the receiver to identify and discard duplicates, ensuring that only one copy of each packet is processed (7).

  The vector recovery algorithm at the receiver end plays a crucial role in identifying and eliminating duplicate frames, ensuring that only one copy of each frame is processed. This algorithm uses vector-based tracking to efficiently manage and synchronize the incoming frames, reducing latency and improving the overall reliability of the network. By leveraging these techniques, TSN receivers can provide high levels of fault tolerance and maintain integrity of time-sensitive data streams (20).
- Packet forwarding: The forwarding mechanism is designed to ensure the timely and reliable delivery of data packets in a network. When a TSN receiver node receives a packet, it examines the destination address embedded within the packet. Based on this address, the receiver determines whether the packet is intended for itself or if it needs to be forwarded to another node. If the destination address is matched, the packet is processed locally else the packet is forwarded to the next node in the network, following a predetermined path or routing protocol. This mechanism ensures that data packets are efficiently routed through the network, minimizing latency and maximizing reliability, which is crucial for applications requiring precise timing and synchronization.
- Time synchronization: Precision Time Protocol (PTP) (8) is a network-based time synchronization protocol used to synchronize clocks throughout a computer network. The following figure shows the message exchange during the PTP operation.

Figure 1-2. Message Exchange PTP Operation



In a PTP system, one device is designated as the Master clock, which provides the reference time to other devices, known as Slaves. The synchronization process involves several key message exchanges. Initially, the Master sends a Sync message to the Slaves, which includes a timestamp of when the message was sent. The Slaves then send a Delay Request message back to the Master to measure the time it takes for messages to travel between the Master and Slave. Upon receiving the Delay Request, the Master responds with a Delay Response message that includes the precise time the Delay Request was received. By analyzing these timestamps, the Slaves can calculate the network delay and adjust their clocks accordingly to match the Master clock, achieving precise time synchronization across the network.

Path delay is calculated with the help of the following equation:

Delay = ((t2 - t1) + (t4 - t3)) / 2

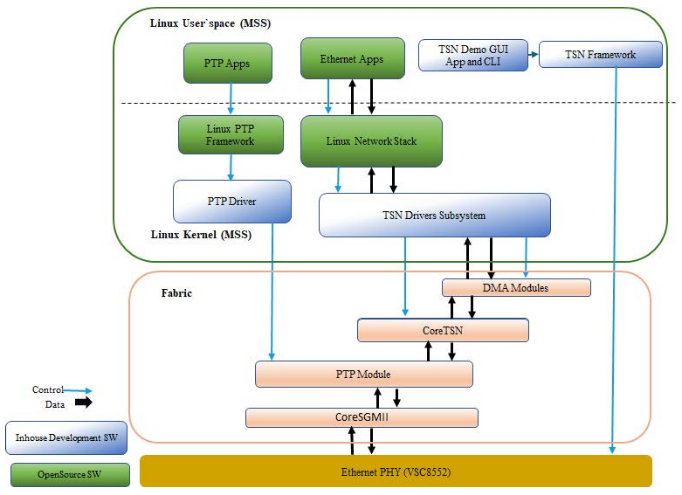
1.2. Software Implementation

The Linux® Software Stack consists of kernel modules, frameworks, and user-space applications operating within the MSS of the PolarFire device. It integrates necessary open-source components as needed. The following figure shows the software driver stack.

The TSN Drivers Subsystem is responsible for configuring CoreTSN, the MAC (a component of CoreTSN), DMA Modules, CoreSGMII, and Ethernet-PHY. It also abstracts their necessary functionalities for the Linux user-space. The MAC driver, which is part of the TSN Drivers Subsystem, operates as a network driver accessible by various user-space applications such as "ethtool," "ifconfig," and "iperf" through "ethx (eth0)." "ethX (eth0)" is the standard name for the first Ethernet network interface in Linux, used for identification purposes. The CoreTSN driver utilizes the DMA Module to transfer incoming network data from the Fabric to the MSS subsystem, and vice versa for the transmission path.

The CoreTSN Stack includes a user-space Command Line Interface (CLI) for configuring TSN specifications such as IEEE 802.1 Qbv, IEEE 802.1 Qbu, and IEEE 802.1 Qci. The CoreTSN Driver, part of the TSN Drivers Subsystem, processes requests from user-space applications to configure the TSN hardware.

Figure 1-3. Software Driver Stack



To enable TSN, it is essential to have PTP (IEEE 1588) functionality activated to achieve clock synchronization between an initiator and a PolarFire SoC device as the target, with precision in the sub-microsecond or nanosecond range. The PTP driver facilitates the required clock synchronization between the PolarFire SoC device and the initiator.

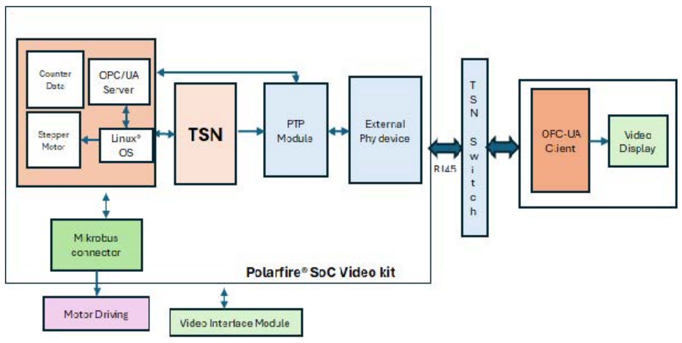
2. System Model

This section discusses the system model.

2.1. Use Case Architecture Model

The use case model is developed using the Polarfire SoC Video kit. The following figure shows the architecture of the model.

Figure 2-1. Use Case Model



The model has the following three major blocks:

- OPC UA client
- OPC UA server with TSN Implementation
- TSN switch

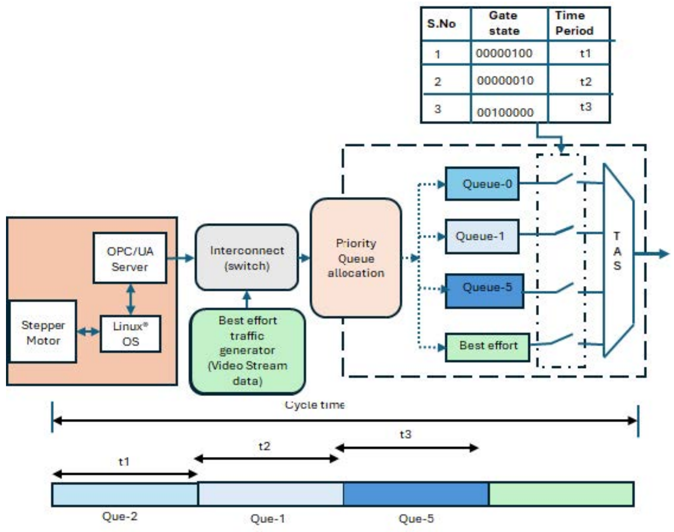
The OPC UA server implemented using the Polarfire SoC FPGA video kit has software implemented in MSS and TSN and PTP modules in the fabric portion. Both OPC UA server and clients are connected through a TSN switch. The entire communication operates at a speed of 1 Gbps on a copper interface.

On the server side, there is an interface with a stepper motor via the Mikrobus™ connector, which receives commands from the client to instruct the motor for the next rotation. The internal MSS on the server side accesses the TSN and PTP modules through data and control interfaces. Additionally, the server is equipped with a Video Interface Module to continuously stream video data.

The defined model includes an interconnect switch with two input ports: one from the server and the other from the Best-Effort Traffic Generator (video interface data) module. The switch processes traffic from both interfaces and directs it to the TSN module.

The server generates both the counter and motor control traffic, which are transmitted as scheduled traffic, while video traffic is sent as best-effort traffic. The Time-Aware Shaper (TAS) module within TSN is responsible for ensuring that the scheduled traffic reaches the client within a deterministic timeframe (13). The following figure shows the TAS and the traffic scheduling process.

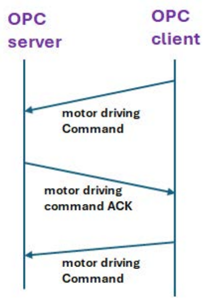
Figure 2-2. Time Aware Shaper for Traffic Scheduling



2.2. Motor Control Flow

The following figure shows the sequence of commands for motor control. Once all the peripherals have been properly initialized and set up, the OPC client sends motor driving commands to the OPC server. Upon receiving these commands, the server instructs the motor to rotate, ensuring seamless motor operation on the server side. For each driving command received, the server transmits an acknowledgment back to the client as scheduled traffic. This acknowledgment prompts the client to send the next driving command. To ensure seamless motor rotation, the acknowledgment packet must reach the client within the stipulated time for every request received.

Figure 2-3. Motor Control Commands



2.3. Time Synchronization

In addition to sending motor control commands, the OPC client functions as a master and transmits Precision Time Protocol (PTP) synchronous clock messages to the OPC server. The slave adjusts its clock based on the synchronous and follow-up messages received from the Master (14). Following the PTP protocol, the slave calculates the path delay, which aids in configuring the Gate Control List (GCL) time slot registers. The following table lists the GCL configuration used for testing.

Table 2-1. GCL Configuration

Packet type	Priority (PCP)
PTP packet	PCP 4
Motor control/Counter Packet	PCP 5
Video traffic packet	Best Effort

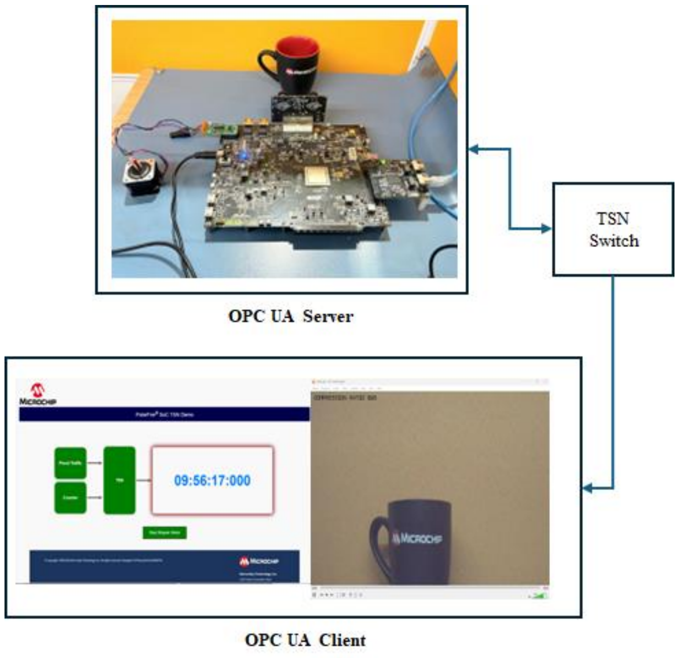
3. Hardware Testing

This section discusses hardware testing.

3.1. Test Setup

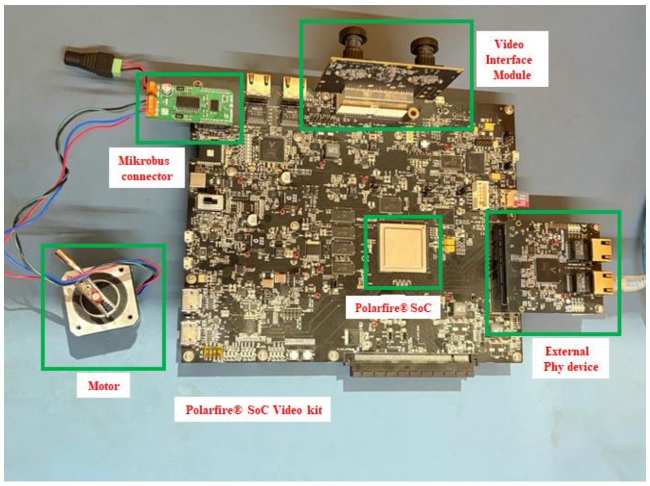
The following figure shows the test setup used for the hardware testing.

Figure 3-1. Test Setup



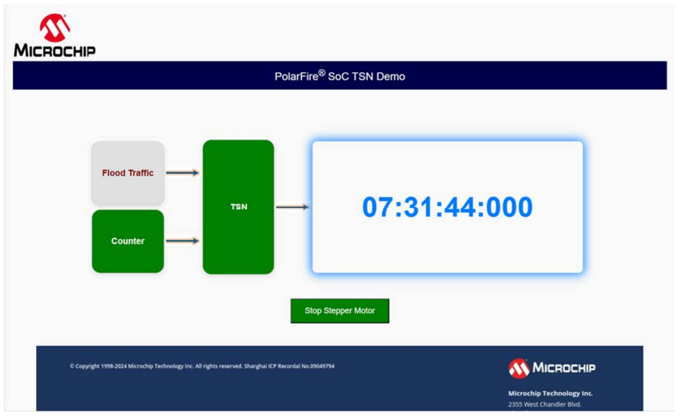
The OPC UA server is implemented using the Polarfire SoC Video kit. The following figure shows the details of the interfaces.

Figure 3-2. OPC UA Server Interfaces



The following figure shows the Graphical User Interface (GUI) used for the testing.

Figure 3-3. Graphical User Interface



The following table lists the key UI elements available in the preceding GUI:

Table 3-1. User Interface Elements in the GUI

UI Element	Description
Flood Traffic	Generates best-effort data, such as video traffic
Counter	Transmits counter data at 1-second intervals.
Start Stepper Motor	Allows the client to send motor control commands, while the server responds with acknowledgment packets to ensure continuous motor operation. The client initiates this command every second upon receiving an acknowledgment packet.
TSN	Enables traffic scheduling based on the Gate Control List (GCL) configuration outlined in the Table 2-1

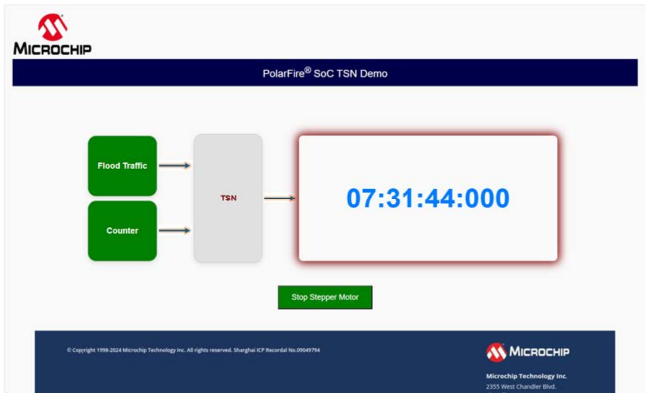
3.2. Configurations

The test is conducted in three configurations:

1. Data flow with best-effort traffic disabled: In this scenario, video data transmission is halted and only motor control and counter packets are transmitted from the OPC UA server. The graphical user interface for this configuration is shown in Figure 3-3.

   Observation: The motor spins continuously without halting as all motor control requests and acknowledgements are sent with determined latency. The counter values are received properly by the OPC UA client and displayed on the screen without any interruption.
2. Data flow with best-effort traffic enabled and traffic scheduling disabled: Video data is transmitted as best-effort packets. Motor control and counter data packets are also sent, but time scheduling is disabled. The following figure shows the GUI for this scenario.

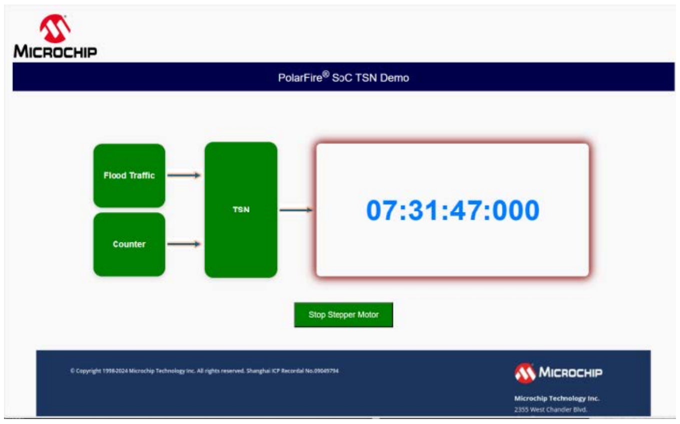
   Figure 3-4. Best-Effort Traffic Enabled, Traffic Scheduling Disabled

Observation: Due to the disabling of time scheduling, the motor control packets and counter data packets do not reach the OPC UA client within the specified latency. This results in the motor operating intermittently and the counter data updating at irregular intervals

3. Data flow with best-effort traffic enabled and traffic scheduling enabled: The video data is activated and transmitted as best-effort packets. The motor control and counter data packets are also sent, with time scheduling enabled (11) and (13). The following figure shows the GUI for this scenario.

Figure 3-5. Best-Effort Traffic and Traffic Scheduling Enabled



Observation: Enabling time scheduling for the motor control packets and counter data packets ensures that they reach the OPC UA client with the required latency. This results in the motor spinning continuously and the counter data being updated at regular intervals.

4. Conclusion

In conclusion, one of the most promising advancements in industrial communication is the combination of TSN with the OPC UA. The practical implementation of the system using the PolarFire SoC kit, which includes a motor and a video interface unit, successfully demonstrated the capabilities of an OPC server communicating with an OPC client in a scheduled TSN format. The motor control application, a pivotal element in industrial automation, was effectively managed through the enabling and disabling TSN features. This not only displayed the system's robustness but also highlighted the significant impact of TSN on critical system metrics such as latency, jitter and bandwidth utilization. Our findings underscore the transformative potential of TSN in enhancing OPC UA applications, paving the way for more robust, efficient and responsive industrial automation systems that leverage existing Ethernet infrastructure.

Future work focuses on further optimizing the integration of TSN with OPC UA and exploring additional industrial applications to fully realize the benefits of this innovative approach. This continued research and development is crucial in advancing the field of industrial automation, ensuring that systems are well-equipped to meet the demands of modern industrial environments.

5. References

The following references are used in this document:

1. IEEE Standard for Local and metropolitan area networks— “Virtual Bridged Local Area Networks Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams” IEEE Std 802.1Qav™-2009
2. IEEE Standard for Local and metropolitan area networks —“Bridges and Bridged Networks— Amendment 28: Per-Stream Filtering and Policing” IEEE Std 802.1Qci™-2017
3. IEEE Standard for Local and metropolitan area networks— “Bridges and Bridged Networks Amendment 25: Enhancements for Scheduled Traffic” IEEE Std 802.1Qbv™-2015
4. IEEE Standard for Local and metropolitan area networks—"Bridges and Bridged Networks— Amendment 29: Cyclic Queuing and Forwarding” IEEE Std 802.1Qch™-2017
5. IEEE Standard for Local and metropolitan area networks—"Bridges and Bridged Networks— Amendment 26: Frame Preemption” IEEE Std 802.1Qbu™-2016
6. IEEE Standard for Ethernet Amendment 5: “Specification and Management Parameters for Interspersing Express Traffic” IEEE Std 802.3br™-2016
7. IEEE Standard for Local and metropolitan area networks — “Frame Replication and Elimination for Reliability” IEEE Std 802.1CB™-2017
8. IEEE Standard for Local and Metropolitan Area Networks— “Timing and Synchronization for Time-Sensitive Applications” IEEE Std 802.1AS™-2020.
9. Lejla Murselovi_c, “PERFORMANCE ANALYSIS OF THE PREEMPTION MECHANISM IN TSN”, June 9, 2020.
10. Miklós Máté1, Csaba Simon1, Markosz Maliosz1, “Asynchronous Time-Aware Shaper for Time-Sensitive Networking”, 12 September 2022.
11. Thomas St¨uber, Lukas Osswald, Steffen Lindner, Michael Menth, A Survey of Scheduling Algorithms for the Time-Aware Shaper in Time-Sensitive Networking (TSN), 27 Mar 2023.
12. Moonbeom Kim, Junhong Min, Doyeon Hyeon, and Jeongyeup Paek, “TAS Scheduling for Real-Time Forwarding of Emergency Event Traffic in TSN”, 2020.
13. Thomas Stuber, Lukas Osswald, Steffen Lindner and Michael Menth, (Senior Member, IEEE) “A Survey of Scheduling Algorithms for the Time-Aware Shaper in Time-SensitiveNetworking (TSN)”, 22 June 2023.
14. Morteza Hashemi Farzaneh and Alois Knoll, “Time-Sensitive Networking (TSN): An Experimental Setup”, 2017.
15. Simon Brooks, Echan Uludag, “Time sensitive Networking From Theory to Implementation in industrial Automation”.
16. AN4977, “OPC-UA Industrial Edge demo”, 2024
17. Huang Renjie, Liu Feng, Pan Dongbo, “Research on OPC UA Security”, 2010
18. Marc Boyer, “Usage of TSN Per-Stream Filtering and Policing”, 2023
19. Fabian Ihle , Steffen Lindner , and Michael Menth , Senior Member, IEEE , “P4-PSFP: P4-Based Per-Stream Filtering and Policing for Time-Sensitive Networking”, IEEE Transactions on network and service management, Vol. 21, No. 5, October 2024
20. Lisa Maile, Dominik Voitlein, Kai-Steffen Hielscher, Reinhard German, “Ensuring Reliable and Predictable Behavior of IEEE 802.1CB Frame Replication and Elimination”, 2022

6. Revision History

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	09/2025	Initial Revision

Microchip Information

Trademarks

ISBN: 979-8-3371-2046-1

Legal Notice

Microchip Devices Code Protection Feature