Application Note

Using TSN Ethernet Features to Improve Timing in Industrial Ethernet Controllers

Pekka Varis, Thomas Schneider, Nilabh Anand and Daolin Qiu

ABSTRACT

Control systems for factories, buildings, grid infrastructure and transportation currently use industrial Ethernet protocols such as PROFINET®, CC-Link®, EtherCAT®, and many others to manage and control production lines, robots, and automation. The short cycle time of real-time Ethernet makes it possible to run control algorithms with a central control unit. IEEE Ethernet standardization has introduced several standard features under the umbrella of Time Sensitive Networking (TSN) to enable achieving the real-time required by these applications.

In practice, the existing versions of industrial Ethernet protocols are sufficient and still used. Industrial Ethernet protocols such as EtherCAT often utilize standard IEEE 802.1Q Ethernet at the controller and modified solutions at the devices. To improve the preciseness of timing of placement of a frame on the wire, some non-IEEE standard solutions like time triggered send (TTS) are being used at the devices.

This application note features the use of the standard TSN feature of Enhancements to Scheduled Traffic (EST) at the controller in comparison to standard Ethernet protocols. EST, also known as time-aware-shaper or Qbv, is one of the fundamental TSN features available in modern embedded processors. A comparison of achievable timing accuracy between a standard TSN based solution and a vendor specific TTS are presented in this application note.

Table of Contents

Trademarks

1 Introduction

Industrial control requires real-time communication with deterministic latency. The technologies used have evolved from serial field buses to industrial Ethernet protocols defined in IEC standards such as EtherCAT, EtherNet/IP™ and PROFINET. These standards use parts of IEEE Ethernet to leverage some of the economies of scale that Ethernet offers, but they add small changes like cut-through switching that go beyond and partially restrict the use of typical IEEE bridges and endpoints. Contrary to typical consumer or enterprise systems, where average responsiveness or throughput are key performance indicators, the performance of industrial control applications are bounded by the worst-case latency of interacting with inputs and outputs across the network.

IEEE 802.1Q-2018 introduced many of the Time Sensitive Networking (TSN) features to standard IEEE Ethernet that used to require an industrial Ethernet network. With technologies like OPC UA FX [1] and the TSN profile IEC/IEEE 60802 [2], it is envisioned that a local area network engineered with TSN features would enable the use of standard IEEE Ethernet hardware to implement the industrial control network. Although introduced several years ago [4], this technology is still a work in progress. It looks like in earlier versions of this technology, the target is controller to controller communication, and thus co-existing with existing technologies. It is also worth noting that the existing industrial Ethernet technologies are still growing and are the backbone of modern factory automation.

Regardless of the transition, the typical industrial topology remains the same as shown in Figure 1-1. Generally, Ethernet, including TSN, specifies layers 1 and 2 of the local area network (LAN). This allows for stateless and unreliable transmission of variable-sized frames from one endpoint to another endpoint and the switching in between. This domain is shown shaded in light gray in Figure 1-1. The protocol on top, such as EtherCAT, is highly asymmetric; there is one controller managing a few or even hundreds of devices. Each of the protocols uses slightly different terminology for this asymmetric relationship, and there are varying levels of this asymmetry. Similar to other engineering specifications, the terminology is in transition as some have found it offensive [3]. The term “controller” for the managing entity and “device” for what is generally being controlled is used in this document. Specifically with EtherCAT, the terms “main” and “subordinate” will be used.

This application note features a comparison of standard TSN feature of EST and established industrial Ethernet technologies to show the optimization of the controller's performance.

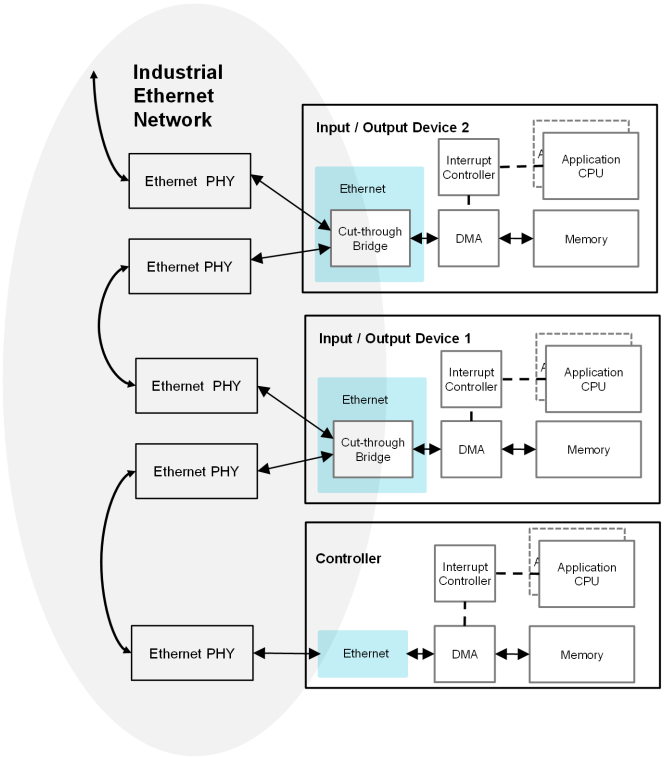

Figure 1-1. A Typical Industrial Ethernet Network

2 Industrial Ethernet Protocol Software Stack

2.1 Overview

An application level protocol like PROFINET or OPC UA is used above TSN capable Ethernet with or without transport and session level protocols like IP and UDP. Figure 2-1 shows an example of OPC UA and the networking layers. EtherCAT is always directly over Ethernet, shown as the blue arrow line for OPC UA Pub-Sub in Figure 2-1, with only the controller (Main) initiating the sending of a frame. The local area network (LAN) only has frames of the EtherType reserved for EtherCAT, sent by the Main or the last Subordinate in the line topology.

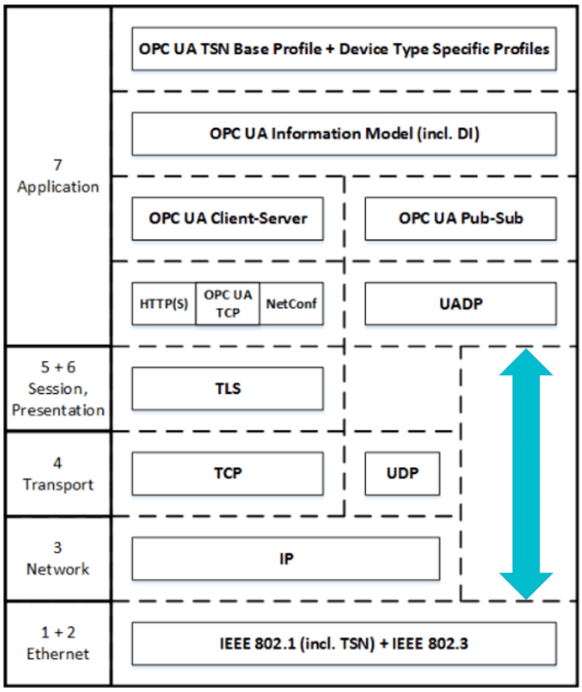

Figure 2-1. OPC UA Over TSN With Blue Line Indicating OPC UA Pub-Sub

Networking software stacks have successfully leveraged the layered model shown in Figure 2-2 for decades. While great for adding features and scalability, the drawback for real-time control is worst-case latency. This is where industrial protocols prioritize timeliness or managed latency over everything else, in practice bypassing the networking stack and sometimes even the Ethernet driver.

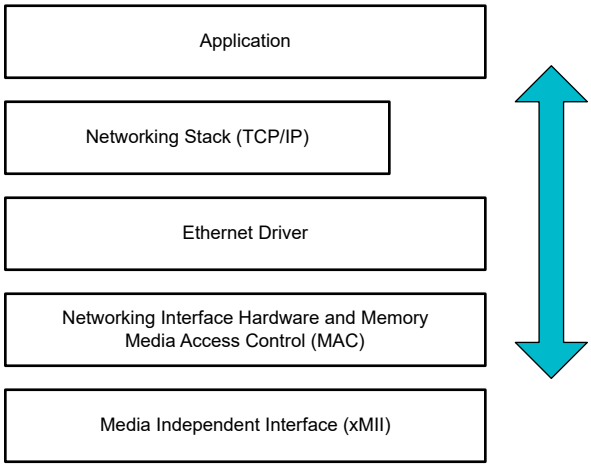

Figure 2-2. Networking Software Stack

Distributed real-time control applications typically use the concept of a cycle. The control application reads inputs, does computation on what the next desired state is, and then sends those outputs out. With industrial Ethernet the inputs and outputs are distributed over the network. This can be visualized in a timing diagram as shown in Figure 2-3. The gray part, labeled “App. Task”, is the time consumed by the application in each cycle. Since input and output communication are processes that consume application task processing time in the cycle period, the overhead of input and output communication needs to be budgeted. This budget is based on the worst-case for both compute and communication jitter.

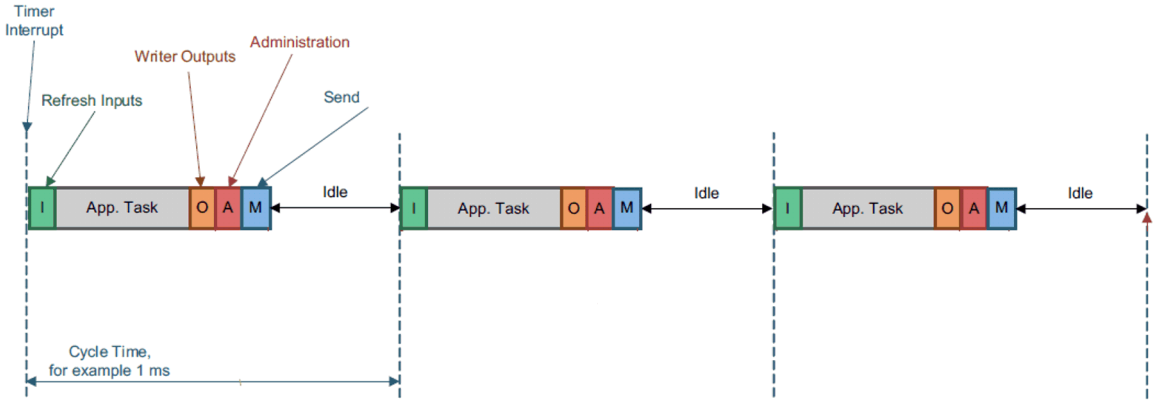

Figure 2-3. Time View of Cyclic Control [5]

In addition to the communication of inputs and outputs, an industrial communication protocol provides the distributed devices with synchronization, or a common view of time. The accuracy of synchronization is often directly relevant to the application, regardless of cycle time or compute needs. For instance, synchronization jitter at the microsecond level or even down to the sub-100 nanosecond level across a distributed system is often critical for a cyber-physical system. However, in the communication of inputs and outputs, the system can tolerate microsecond or even tens-of-microseconds level jitter. The cost of this jitter is reduced application computation time in each cycle.

2.2 EtherCAT

EtherCAT is an IEEE 802.3 Ethernet-based fieldbus system standardized in International Electrotechnical Commission (IEC 61158). The technology is supported by the EtherCAT Technology Group, an international community of users and vendors. The protocol is particularly popular in motion and motor control. The primary advantage of EtherCAT is that it supports automation applications that require short data-update times with low communication jitter. In the EtherCAT protocol, the EtherCAT Main (formerly called Master) sends a frame that passes through each Subordinate node (formerly called Slave). Each EtherCAT Subordinate device reads the data that is addressed to it as soon as the data is detected. Then, the subordinate device inserts the data into the frame, while the frame is on-the-fly bridged. The last Subordinate node in a segment (or branch) detects an open port and sends the message back to the main. The EtherCAT Main is the only node within a segment that actively sends a new EtherCAT frame. This capability permits the network to achieve over 90% of the available network bandwidth while preventing unpredictable delays, and thus guarantees real-time system response. EtherCAT is transported with EtherType identifier (0x88A4).

The only frames sent on the LAN are from the EtherCAT Main and the last Subordinate. The typical optimization at the Main is to have the stack directly access the Ethernet MAC controller, bypassing not only the networking stack, as with OPC UA Pub-Sub over raw Ethernet, but also bypassing the Ethernet driver to directly or natively own the entire Ethernet peripheral. Acontis [6] and IBV [7] are stack providers offering this optimization. An example of this is show in Figure 2-4.

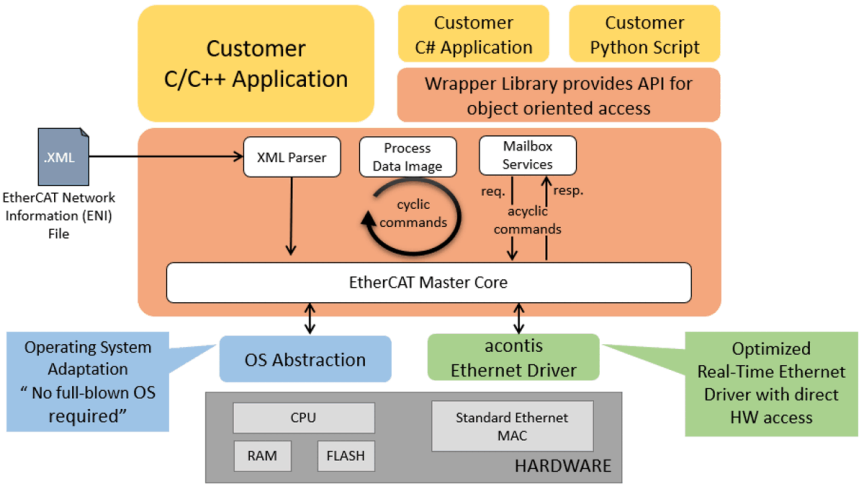

Figure 2-4. EtherCAT Main Software Architecture [6]

EtherCAT is a broadly deployed protocol and detailed benchmarks are available on many platforms [8] [9] [10]. As a reference point of what can be achieved, the clock synchronization reached is usually claimed to be below 100 ns and in practice, ±20ns. Clock synchronization is usually measured by using an oscilloscope to view the required SYNC output on each Subordinate node and comparing the offset and jitter among each measurement. The SYNC output is logically similar to generating a 1 pulse per second (pps) type pin toggle from TSN time synchronization (IEEE 802.1AS). Figure 2-5 is an example of this measurement.

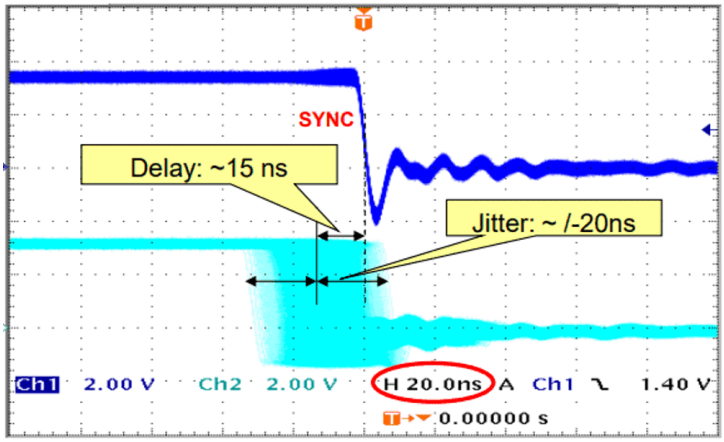

Figure 2-5. EtherCAT Time Synchronization Measurement Example [10]

For the second key timing metric, the ability to place an Ethernet frame precisely on the wire, some Ethernet MACs in embedded processors, like Texas Instruments Sitara™, and network interface cards (NICs) like Intel i210, added a non-IEEE feature called time triggered send (TTS). This feature allows placing an Ethernet frame precisely on the wire at a point of time, commonly applied to be exactly at the beginning of a communication cycle. With 100 Mbit/s (typical EtherCAT deployed today), this reaches +-40ns accuracy [5].

3 Evaluation Platform and Methods

3.1 Hardware

Two classes of embedded Arm® processors from the Texas Instruments Sitara™ family are used for this demonstration. AM6412 and AM625 processors with up to 1.4GHz Arm Cortex®-A53 and 800 MHz Cortex-R5 cores and an integrated embedded TSN switch are used as a lower end controller with the below 2W of power consumption. For the higher end controller, an AM682 processor with dual core 2GHz Arm Cortex-A72 was used.

3.2 Software Platform

The Cortex A53 and A72 cores were running Linux 5.10 with PREEMPT_RT patches, often called RT Linux. Configuration of TSN was done with the iproute2 software package, which includes standard interfaces for configuring networking including TSN features. The software package includes the *tc* command, which stands for "traffic classes queuing discipline" (*tc qdisc*). As the features used are based on standard software interfaces, the tests should be straightforward to replicate on any target with TSN Enhancements to Scheduled Traffic (EST) support using iproute2 package commands.

Optimized EtherCAT Main software stacks are available from IBV [7] and Acontis [6] for these targets.

3.3 Test Application

Linux has an extensive suite of networking test applications such as netperf and iperf3. In addition, plget [11] is a tool that utilizes the precision time protocol (PTP) standardized in IEEE 1588, a hardware capability found in all TSN-capable hardware, to capture timestamps of Ethernet frame arrival and transmit times at the MII interface.

3.4 Test Topology

In a test measuring time synchronization accuracy, an AM6442 starter kit was connected to a Keysight NovusONE as the grand leader.

For the test setup measuring baseline transmit timing, a small network of three Sitara AM2431 MCU devices as EtherCAT subordinates and a Sitara MPU device running CODESYS®, an EtherCAT Main software stack, as the controller. There were two Sitara MPU devices tested, the AM6412 as the lower end controller, and the AM682 as the higher end controller.

For the comparison with a non-EtherCAT setup with the TSN feature of EST applied, an EtherCAT network with a Main sending to a Subordinate was simulated by an AM625 starter kit in place of the Main and an AM6412 in place of the Subordinate without actually setting up a true EtherCAT network.

4 Results

The results are grouped into two categories: time synchronization accuracy and transmit timing.

4.1 Time Synchronization

As a baseline, the SYNC signal of an AM6442 starter kit connected to Keysight NovusONE as the grand leader with distributed clock was measured to match the reference shown in Figure 2-5.

With TSN, time synchronization is achieved using IEEE 802.1AS, often referred to as generalized precision time protocol (gPTP). Figure 4-1 shows the synchronization accuracy of the AM6442 starter kit. A jitter of -9 to +12 nanoseconds measured at 1 Gbit/s is better than what is measured on a network running at 100 Mbit/s.

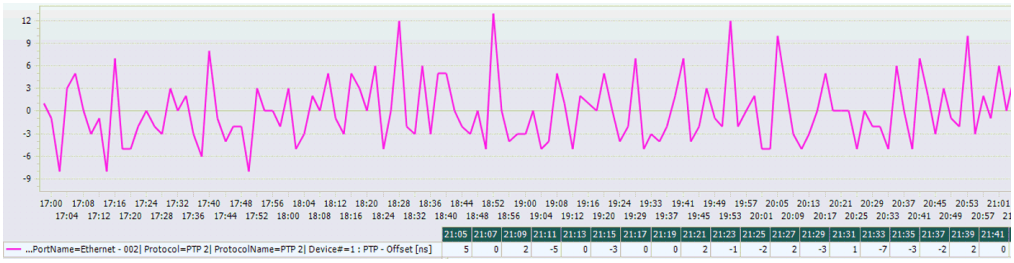

Figure 4-1. IEEE 802.1AS Accuracy AM6442 Starter Kit Measured on Keysight NovusONE as the Grand Leader Using 1 Gbit/s Ethernet

4.2 Transmit Timing

The timing of a Sitara AM6412 running on RT Linux with Codesys EtherCAT Main without any tuning or optimization is shown in Figure 4-2. The measurement is based on the TX_EN RGMII signal, which will be high from the beginning to the end of the frame. The signal stays high for the duration of transmitting the frame. For example, the duration of TX_EN RGMII signal at high level for a 200 byte frame at 100 Mbit/s EtherCAT is 16 µs. Jitter measurements needs to be accounted in the TX_EN measurement. The cycle time used is 1.5 ms, and the jitter of the start of the frames being placed on the wire is about ±120 µs. On the AM682 with 2 GHz Cortex A72 the same test in Figure 4-3 shows about half of that jitter.

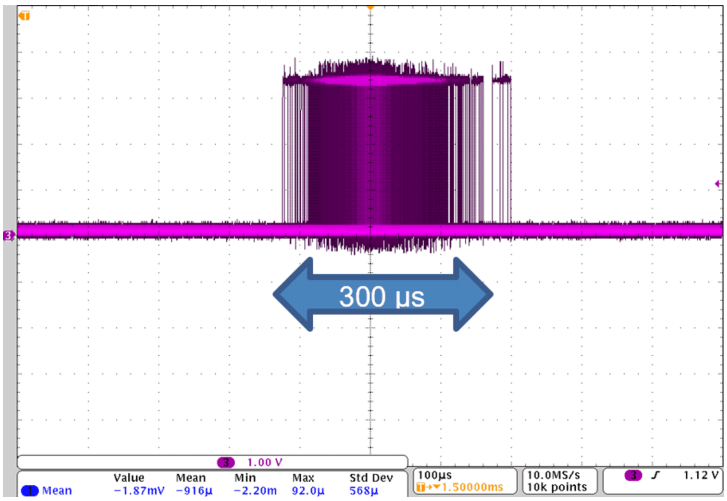

Figure 4-2. Baseline CODESYS on Sitara AM6412 TX_EN Timing of EtherCAT Frames - 1.5 ms Cycle Time, Oscilloscope in Persistence Mode

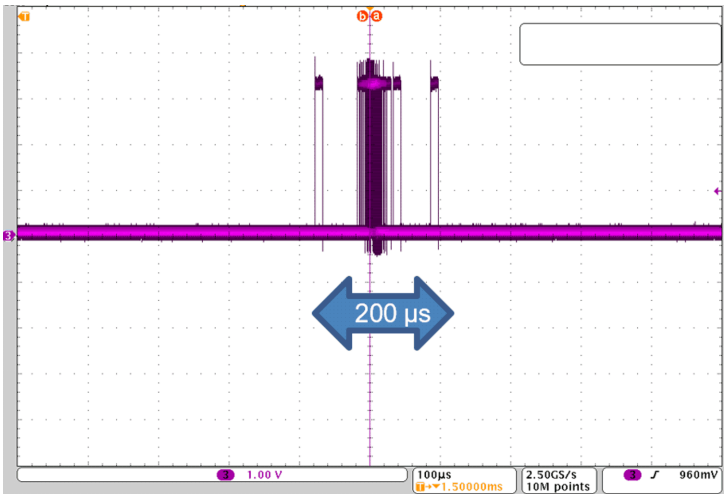

Figure 4-3. Baseline Codesys on Sitara AM682 TX_EN Timing of EtherCAT Frames - 1.5 ms Cycle Time, Oscilloscope in Persistence Mode

Figure 4-4 shows a wireshark capture of the traffic on the Ethernet wire for the same use case on the AM6412.

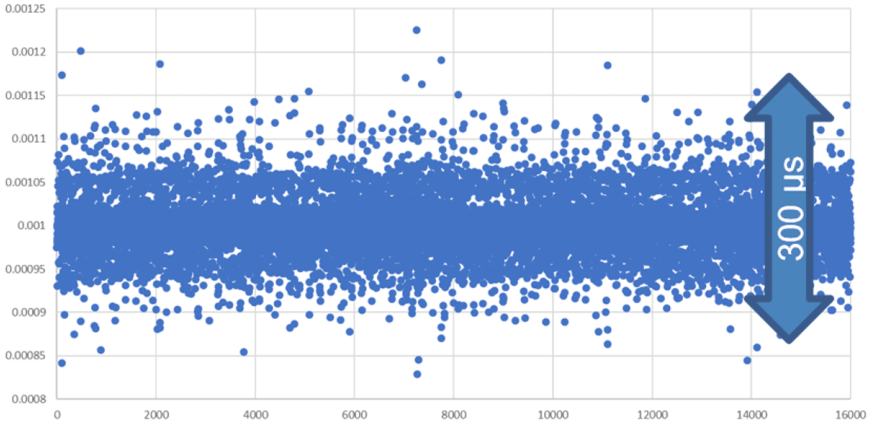

Figure 4-4. Baseline Codesys on Sitara AM6412 Wireshark Capture of EtherCAT Frames - 1 ms Cycle Time

These jitter results are consistent with a typical RT Linux system interrupt latency [12]. Steps as shown in [13] [14] can be used to get this down to the low tens of microseconds. This is still at least two or even three orders of magnitude higher than what is achievable with TTS approaches. Instead of using TTS with EtherCAT, a schedule was created using the TSN feature of EST, where all the transmit gates are closed, but at every 100 µs, the gate is open to send one frame out as shown in the command line snippet in Figure 4-5.

```
tc qdisc replace dev eth0 parent root
\ handle 100 taprio num_tc 8
\ map 0 1 2 3 4 5 6 7 0 0 0 0 0 0 0 0
\ queues 1@0 1@1 1@2 1@3 1@4 1@5 1@6 1@7
\ base-time 100000000
\ sched-entry S ff 2848
\ sched-entry S 00 97152
\ flags 2
```

Figure 4-5. Command for EST Schedule to Achieve Time Triggered Send

The command in Figure 4-5 can initially look complex; however, details are explained in [15]. The two lines with "sched-entry" are important for achieving TTS. These lines specify that for 2848 nanoseconds, all 8 queues (ff, bitmap of 8 queues) are open, while for 97152 nanoseconds, all gates are closed (00). This schedule enables sending one frame at a time, assuming the application has completed its processing and has the frame ready to be sent. To simulate an EtherCAT Main sending to a Subordinate, an AM625 starter kit was used as an endpoint (in place of the Main) and a Sitara AM6412 configured as a bridge (in place of the Subordinate). Plget packet generator was used to generate traffic that represents the EtherCAT Main. Plget hardware timestamping of the receive Ethernet port was used by the Subordinate. The inter packet gap of a run of 10k packets with the EST schedule from Figure 4-5 was within -2 ns to +10 ns, as shown in Figure 4-6.

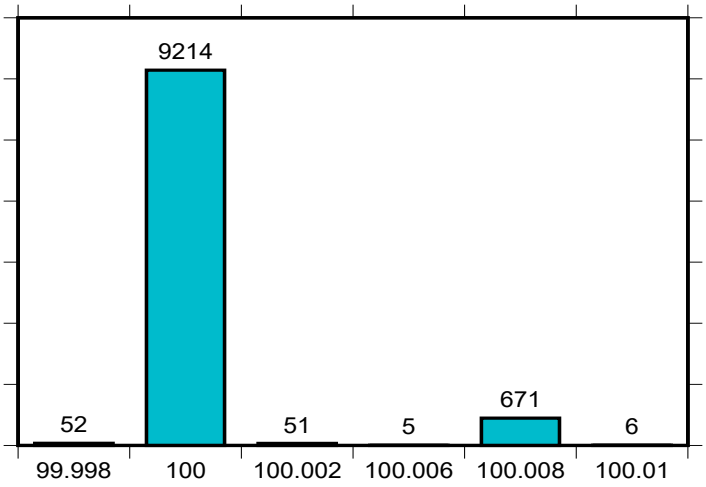

Figure 4-6. Inter Packet Gap in Microseconds of 10k Frames at 1 Gbit/s with a 100 µs EST Shaper

Figure 4-7 shows a TX_EN RGMII measurement of the Codesys EtherCAT Main setup with the EST schedule. This figure is comparable to the baseline TX_EN RGMII measurement in Figure 4-2 without the EST schedule. The time duration due to the length on the wire of sending one frame at 100Mbits/s is ~110 µs.

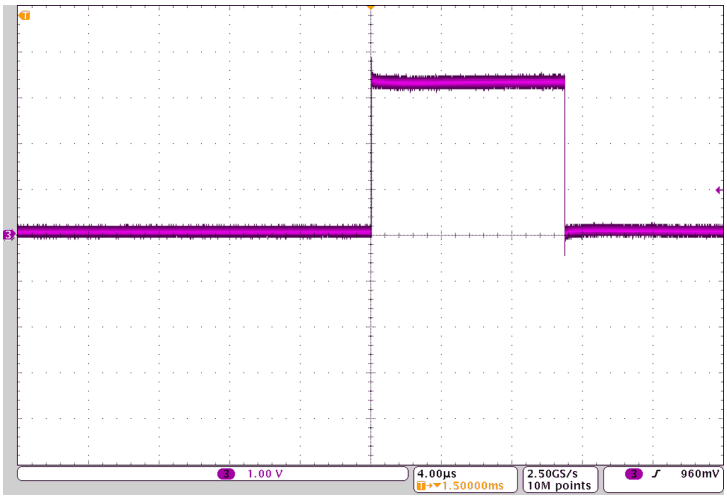

Figure 4-7. TX_EN Timing of EST Shaped Codesys EtherCAT Main - 1.5 ms Cycle Time, Oscilloscope in Persistence Mode

In terms of jitter, the start of frame jitter at 10 ns is not visible at this resolution in the TX_EN signal. Figure 4-8 and Figure 4-9 shows a magnified view into the beginning and ending of sending the frame. This confirms jitter below 10 ns.

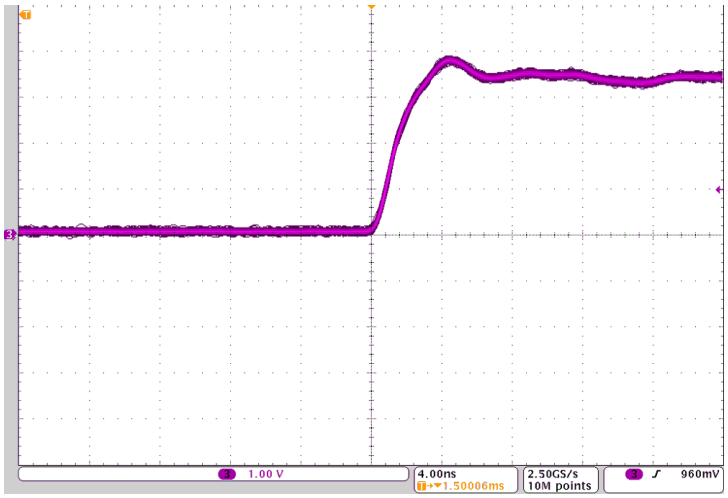

Figure 4-8. Magnified View of Beginning of TX_EN Signal for EST Shaped Codesys EtherCAT Main - 1.5 ms Cycle Time, Oscilloscope in Persistence Mode

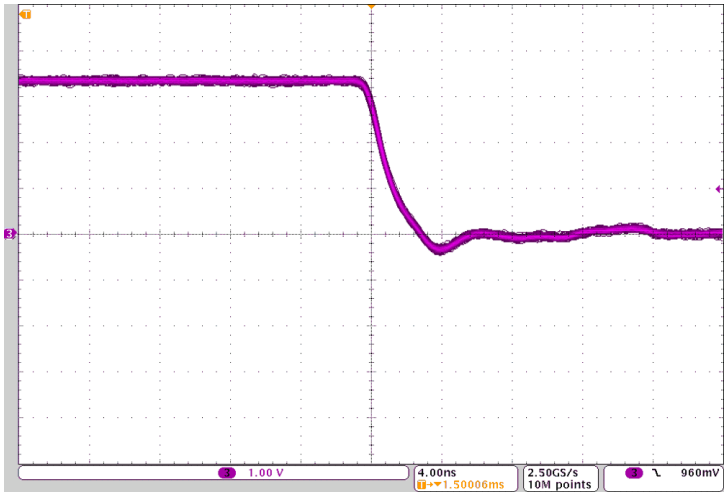

Figure 4-9. Magnified View of Ending of TX_EN Signal for EST Shaped Codesys EtherCAT Main - 1.5 ms Cycle Time, Oscilloscope in Persistence Mode

5 Summary

When properly engineered, Gigabit Ethernet local area networks with IEEE 802.1Q-2018 TSN features can achieve time synchronization comparable to EtherCAT. In both systems, the greatest sources of error will be the hardware design of the clocking and physical level transceivers. By utilizing the standard TSN feature of EST, it is possible to precisely place frames on the wire without relying on vendor proprietary implementations of TTS. These findings are significant for industries that require precise and reliable communication, such as motion and motor control. In addition, the ability to precisely place frames on the wire demonstrate that TSN capable hardware can be used to improve current EtherCAT Main implementations as well as paving the way for TSN solutions to be a viable alternative to EtherCAT in certain use cases.

6 References

1. The OPC Foundation releases the OPC UA Field eXchange (UAFX) Specifications, https://opcfoundation.org/news/press-releases/the-opc-foundation-releases-the-opc-ua-field-exchange-uafx-specifications/
2. IEC/IEEE 60802 TSN Profile for Industrial Automation, https://1.ieee802.org/tsn/iec-ieee-60802/
3. EtherCAT FAQs, 1.5 EtherCAT uses the Master/Slave Medium Access Control (MAC) method. How about inclusive language?, https://www.ethercat.org/en/faq.html#:~:text=Since%20ETG%20does,list%20of%20abbreviations
4. D. Bruckner, R. Blair, M-P. Stanica, A. Ademaj, W. Skeffington, D. Kutscher, S. Schriegel, R. Wilmes, K. Wachswender, L. Leursx, M. Seewaldxi, R. Hummenxii E-C. Liuxiii S. Ravikumarxiv. OPC UA TSN A new Solution for Industrial Communication. https://cdn.weka-fachmedien.de/whitepaper/files/OPC_UA_TSN_-_A_new_Solution_for_Industrial_Communication.pdf
5. EtherCAT® Master Reference Design for Sitara™ AM57x Gigabit Ethernet and PRU-ICSS with Time-Triggered Send, https://www.ti.com/tool/TIDEP0079
6. Acontis EtherCAT Master Stack, https://www.acontis.com/en/ethercat-master.html
7. icECAT, EtherCAT® Master Stack for Embedded Systems,https://www.ibv-augsburg.de/en/products/icnet/ethercat-master/
8. icECAT. EtherCAT Master Stack Benchmark, https://www.ibv-augsburg.de/downloads/icECAT_EtherCAT_Master_Stack_Benchmark.pdf
9. EtherCAT Master Stack Technical Presentation, http://software.acontis.com/Documents/AT1020_V1.1_EC-Master-Technical.pdf
10. EtherCAT Feature: Distributed Clocks, https://training.ti.com/sites/default/files/docs/f2838x_ethercat_distributed_clocks.pdf
11. Ivan Khoronzhuk, “plget”, https://github.com/ikhorn/plgett
12. OSADL QA Farm on Real-time of Mainline Linux, Latency plots of all RT systems under test, https://www.osadl.org/Combined-latency-plot-of-all-RT-systems.qa-latencyplot-allrt.0.html?latencies=&showno=&slider=0
13. CODESYS Control Performance Optimization of Real-Time Performance, https://content.helpme-codesys.com/en/CODESYS%20Control/_rtsl_performance_optimization.html
14. Song, Yoong Siang , Kweh, Hock Leong, Choong, Chwee-Lin, Ong, Boon Leong; Systemic Analysis and Guide to Achieve Concurrent Real-Time Linux Applications over TSN Network, Embedded World Conference 2022
15. TSN Documentation Project for Linux, https://tsn.readthedocs.io/

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265