Application Note

Performance Metrics of TI Embedded Processors as CODESYS EtherCAT Controller

Daolin Qiu, Pekka Varis, Jared McArthur

ABSTRACT

Nearly all real-time industrial control relies on an architecture of periodic computation and communication. Performance estimates for designing solutions are often based on microbenchmarks such as interrupt latency (for example, cyclictest) and some compute benchmarks such as the venerable Dhrystone. EtherCAT® is one of the main industrial Ethernet protocols for control systems in robotics, factory automation, and precise motor drive control. The primary performance metric is the shortest achievable cycle time, with a given number of devices and size of the exchanged data structures. Timing and synchronization stability is another key consideration, both within the controller and within the network. EtherCAT controller (formerly called EtherCAT master) solutions are available for high level operating systems, such as TwinCAT® for Windows® and CODESYS® for Linux, to highly optimized commercial solutions such as icECAT and the open source IgH stack. Some of the optimized designs are able to run on microcontroller cores, and involve different tradeoffs compared to each other and the full featured high-end solutions. This application note explores achievable cycle times on Texas Instruments embedded processors running real-time Linux using the CODESYS EtherCAT stack.

Table of Contents

Trademarks

EtherCAT® and TwinCAT® are registered trademarks of Beckhoff Automation GmbH.
Windows® is a registered trademark of Microsoft Corporation.
CODESYS® is a registered trademark of CODESYS Group.
ARM® is a registered trademark of ARM Limited.
All trademarks are the property of their respective owners.

1 Introduction

Industrial control requires real-time communication with deterministic latency. The used technologies have evolved from serial fieldbuses to industrial Ethernet protocols defined in IEC standards such as EtherCAT. These standards use parts of IEEE Ethernet to leverage some of the economies of scale that Ethernet offers, but the standards add small changes such as 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, in these applications the performance need is bounded by the worst-case latency of interacting with inputs and outputs across the network.

A typical industrial control network topology is shown in Figure 1-1. Ethernet specifies layers 1 and 2 of the local area network (LAN). For standard Ethernet, this allows for stateless unreliable transmission of variable-sized frames from one endpoint to another endpoint and the switching in between. This domain is shown by the area in the dotted rectangle to the left in Figure 1-1. The protocol on top, for example EtherCAT, is highly asymmetric; there is one controller managing a few or even hundreds of devices. Different protocols use slightly different terminology for this asymmetric relationship, and there are varying levels of this asymmetry. This document uses the terms “controller” for the controlling entity and “device” for the devices being controlled.

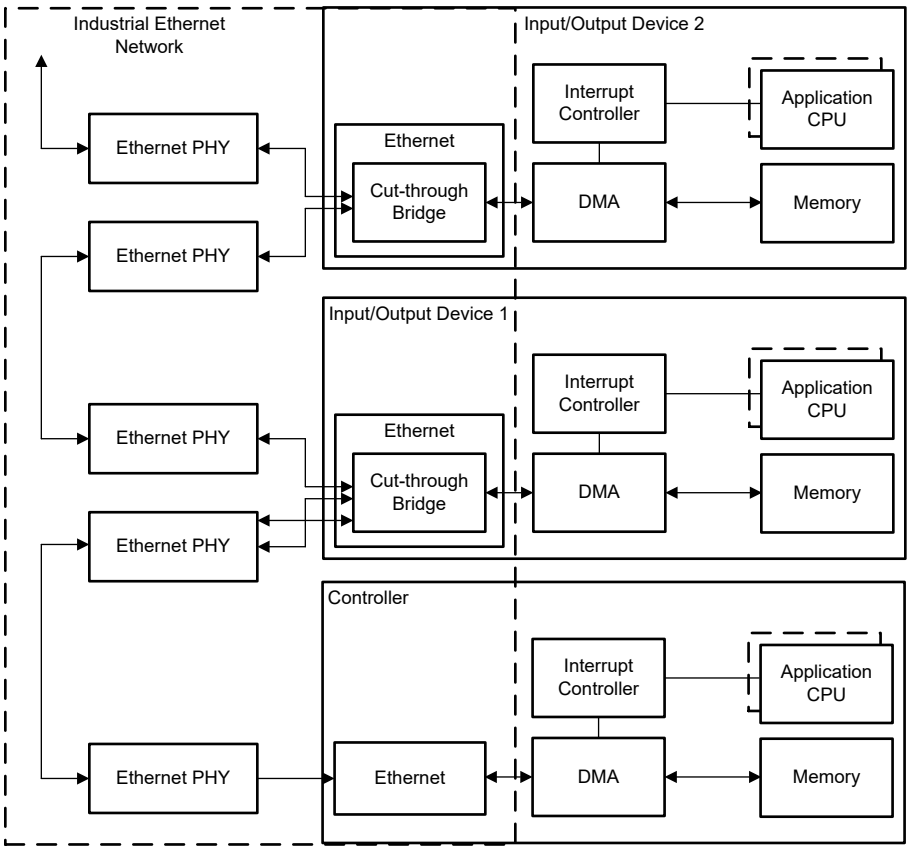


Figure 1-1. A Typical Industrial Ethernet Network

1.1 What is 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 especially popular in motion and motor control. The primary advantage of EtherCAT is found in automation applications with short data-update times and low communication jitter. In the EtherCAT protocol, the EtherCAT controller sends a frame that passes through each device node. Each EtherCAT device reads the data that is addressed to the device as soon as the data is detected. The device then inserts the data into the frame as the frame is bridged on the fly. The last device in a segment (or branch) detects an open port and sends the message back to the controller. The EtherCAT controller is the only node within a segment that actively sends an 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 controller, with the last subordinate on a branch returning the frame to the predecessor. The typical optimization within the controller 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 the Ethernet driver, to directly or natively own the entire Ethernet peripheral. An example of this is shown in Figure 1-2. Acontis and IBV are stack providers offering this optimization.

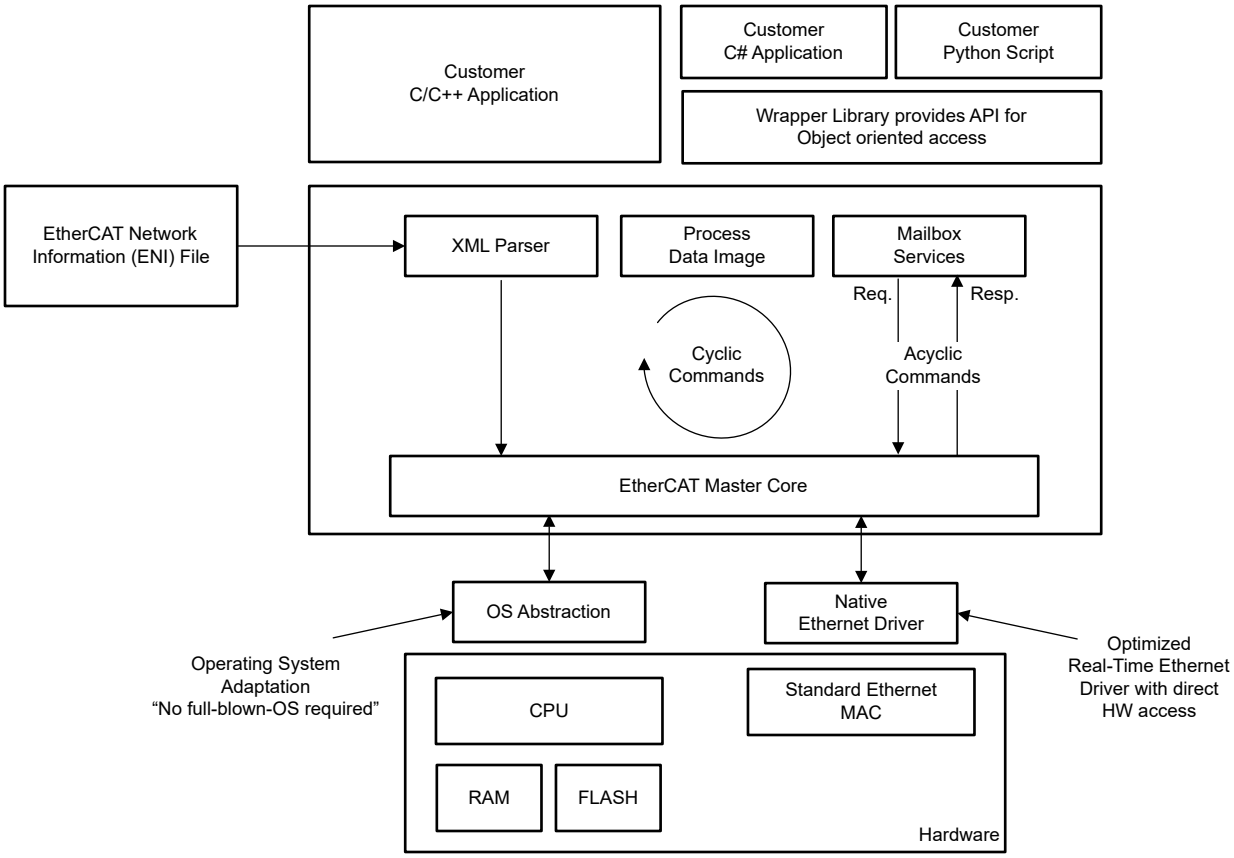


Figure 1-2. EtherCAT Controller Software Architecture

1.2 What is a PLC?

Programmable Logic Controllers (PLC) and motion controllers are industrial-grade computers meant for control of inputs, outputs, and motors, mostly for manufacturing processes, requiring high reliability, ease of programming, and process fault diagnosis. Typically, PLCs can use an industrial ethernet protocol stack such as an EtherCAT controller communication stack for communication of data from the controller to devices that control sensors and actuators distributed over the LAN. PLCs are often used in applications that are real-time and cyclic. When every cycle input arrives, calculations are then made for the new outputs. The calculation and communication variation in latency, commonly referred to as jitter, can adversely affect the achievable EtherCAT cycle time.

1.3 What is CODESYS?

CODESYS, short for *COntrols DEvelopment SYStem*, is an integrated development environment for programming PLC applications based on IEC 61131-3. CODESYS is the leading manufacturer-independent IEC 61131-3 automation software for engineering control systems. There are two key pieces of software when using CODESYS. The first piece is an Integrated Development Environment known as the CODESYS Development System which acts as a GUI for programmers and controls engineers to write control applications using the IEC 61131-3 programming languages and to create visualizations for those control applications. The second piece is a runtime software typically installed on a PLC, but CODESYS can be installed on any intelligent device that meets the minimum requirements of the runtime. One example of a runtime system is CODESYS Control for Linux ARM SL, intended for an ARM®-based board or device used for control applications with pre-installed Linux with or without hard real-time requirements.

2 Evaluation Platform and Methods

2.1 Hardware

Performance metrics are captured with SK-AM62B running as EtherCAT controller using the CODESYS EtherCAT stack. A network of ten Beckhoff digital output devices (EL2889) are connected to the EtherCAT controller using a Beckhoff EtherCAT coupler (EK1100). These metrics are further captured on TMDS64EVM, SK-AM69, and SK-TDA4VM with the same network.

All platforms in Table 2-1, are benchmarked as an EtherCAT controller except the AM62A platform. As shown in the table, AM62A offers some potential advantages in terms of higher DDR speed and larger DDR bus-width.

Table 2-1. Texas Instruments ARM®-Based Processors

Part Number	ARM CPU	ARM (max) MHz	Operating System	Cache	External Memory on EVM
AM62x	4 Cortex-A53 SoC	1400 (64-bit)	Linux	32KB L1 DCache 32KB L1 ICache 512KB L2 Cache	DDR4 1600 MT/s 16-bit width
AM64x	2 Cortex-A53 SoC with Cortex-R5F Co-CPU	800, 1000 (64-bit)	Linux, RTOS	32KB L1 DCache 32KB L1 ICache 256KB L2 Cache	DDR4 1600 MT/s 16-bit width
AM69	8 Cortex-A72 SoC with Cortex-R5F Co-CPU	2000 (64-bit)	Linux, RTOS	32KB L1 DCache 48KB L1 ICache 2MB L2 Shared Cache	LPDDR4 2133 MT/s 32-bit width
TDA4VM	2 Cortex-A72 SoC with Cortex-R5F Co-CPU	2000 (64-bit)	Linux, RTOS	32KB L1 DCache 48KB L1 ICache 1MB Shared L2 Cache	LPDDR4 2133 MT/s 32-bit width
AM62A	4 Cortex-A53 SoC	1400 (64-bit)	Linux	32KB L1 DCache 32KB L1 ICache 512KB Shared L2 Cache	LPDDR4 3733 MT/s 32-bit width

To best replicate a real factory automation environment, where EtherCAT is typically used for an indefinite period of time, a CODESYS license is acquired and applied to a dedicated CODESYS USB dongle. Without a CODESYS license, each run has a timeout at 30 minutes, at which point the run needs to be manually restarted. To run longer than 30 minutes, each target EtherCAT controller requires a USB port to read the license applied on the CODESYS USB dongle.

2.2 Software

SK-AM62B and TMDS64EVM ran on a real-time Linux operating system, specifically booting using an SD card flashed with the PROCESSOR-SDK-LINUX-RT-AM62x and PROCESSOR-SDK-LINUX-RT-AM64x default wic image, respectively, from Software Development Kit (SDK) version 09.01.00.08.

The TDA4VM and AM69 ran on real-time builds from the respective SDK 09.01.00.06 versions.

CODESYS Control for Linux ARM64 version 4.9.0.0 were manually installed in the Linux root directory by downloading the software package from the CODESYS website, extracting the ipk package, copying the ipk into the Linux root directory of the target EtherCAT controller, and extracting the package with the `opkg` command.

In addition to the CODESYS Control for Linux ARM64 version ipk package, a CodeMeter application must also be installed to read the CODESYS license on a CODESYS USB dongle. This must be extracted as a deb package from the same CODESYS Control for Linux ARM64 downloaded package and manually installed on the target EtherCAT controller with the `opkg` command.

More specific instructions on how to setup the TI embedded processors as EtherCAT controller using the CODESYS stack can be found in Section 7.

2.3 Test Topology

As described in Section 2.1, each TI embedded processor acting as EtherCAT controller is connected to 10 Beckhoff digital output devices using a Beckhoff EtherCAT coupler. In addition to this, through another ethernet port, each tested EtherCAT controller is connected to an Ethernet switch which also connects a Windows machine running the CODESYS Development System for monitoring the performance metrics in real time. Optionally, a DHCP server can be connected to the Ethernet switch to dynamically assign IP addresses to the Windows machine and target EtherCAT controller. The benchmarks reported here are with statically assigned IP addresses. If the Windows machine is previously connected to the internet, after connecting to the EtherCAT network, internet access may no longer be available.

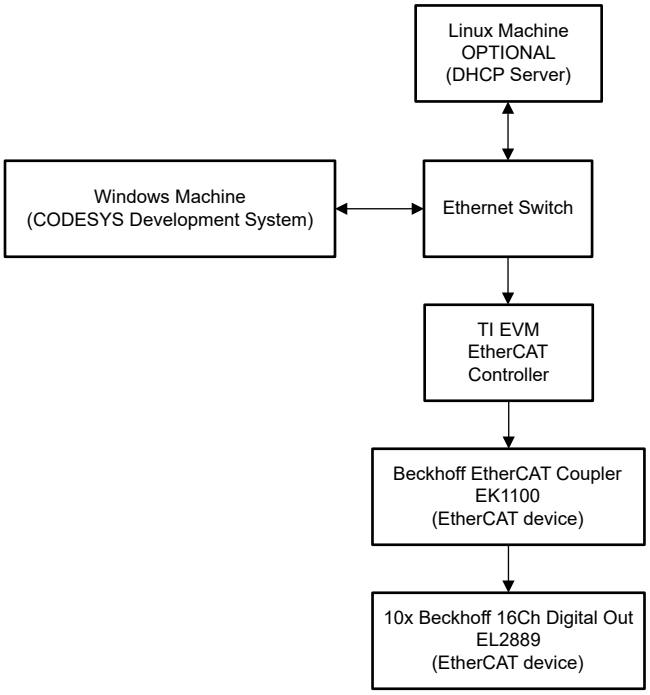


Figure 2-1. EtherCAT Network Test Topology Using CODESYS Control for Linux ARM SL

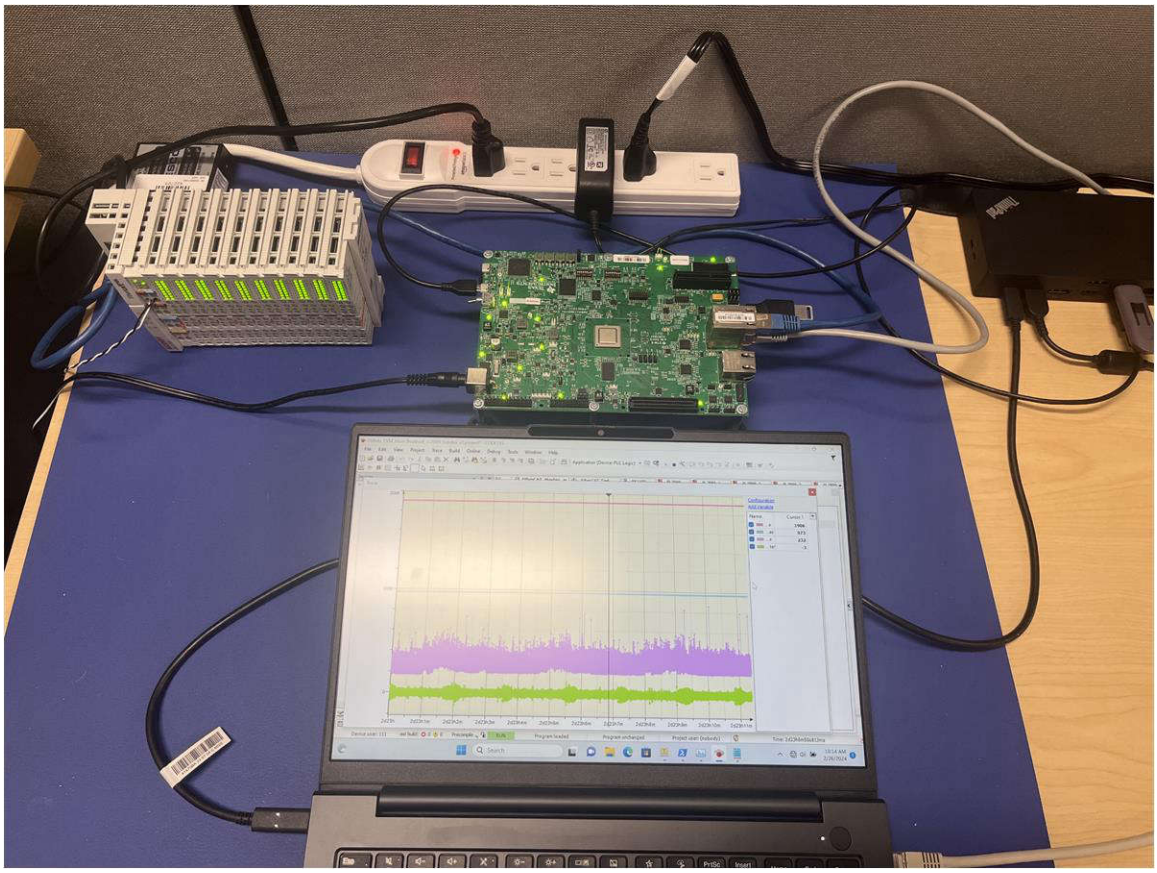


Figure 2-2. EtherCAT Network With AM64x as Controller and Beckhoff Devices as EtherCAT Devices

3 Performance Metrics

3.1 Cyclictest Performance Metrics

Before evaluating the performance of each hardware platform when running an EtherCAT application, gathering baseline performance statistics from running *cyclictest* is typically recommended. These statistics are a good indicator to the performance of an EtherCAT controller when running the CODESYS stack.

cyclictest is a utility tool in Linux (see also, the Linux Foundation) that accurately and repeatedly measures the difference between the intended wake-up time of a thread and the time at which the thread actually wakes up to provide statistics about the latencies of the system. Latencies caused by hardware, firmware, and the operating system in real-time systems can be measured by *cyclictest*.

The results of *cyclictest* are plotted in the same histogram format as the data Open Source Automation Development Lab (OSADL) publishes in *QA Farm on Real-time of Mainline Linux*. OSADL is an organization intended to facilitate individuals, groups, and companies to develop Open Source software by bringing in a broader community to help with development. A service provided by OSADL is the QA farm, which is a quality assurance and assessment test center for embedded systems.

The results of *cyclictest* were captured on the AM62x evaluation board (SK-AM62B), AM64x evaluation board (TMDS64EVM), TDA4VM starter kit (SK-TDA4VM), and AM69 starter kit (SK-AM69) to show the base latencies running on these platforms. SK-AM62B and TMDS64EVM ran on PROCESSOR-SDK-LINUX-RT-AM62x and PROCESSOR-SDK-LINUX-RT-AM64x default wic image, respectively, from Software Development Kit (SDK) version 09.01.00.08. The TDA4VM and AM69 ran on real-time builds from the respective SDK 09.01.00.06

versions. Each cyclictest ran for 6 hours under stress test using the stress-ng tool. See the following code snippet for this setup.

```
stress-ng -c <number of cpu cores> --cpu-method all &
cyclictest -m -Sp98 -D6h -h400 -i200 -q > <histogram name>.hist
```

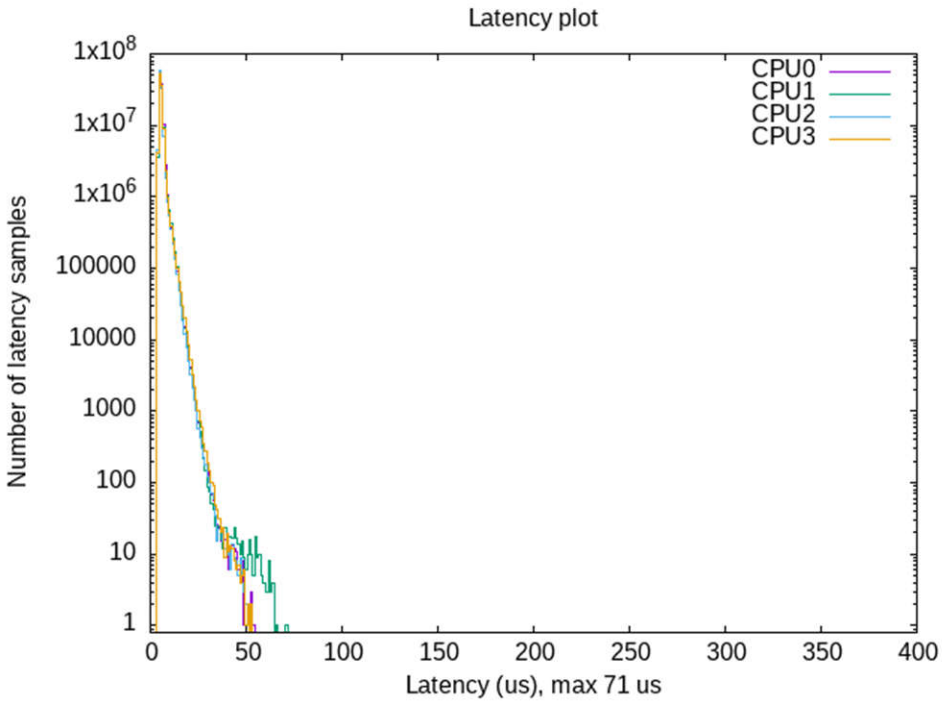


Figure 3-1. Latency Plot for AM62x Running for 6 Hours Under stress-ng

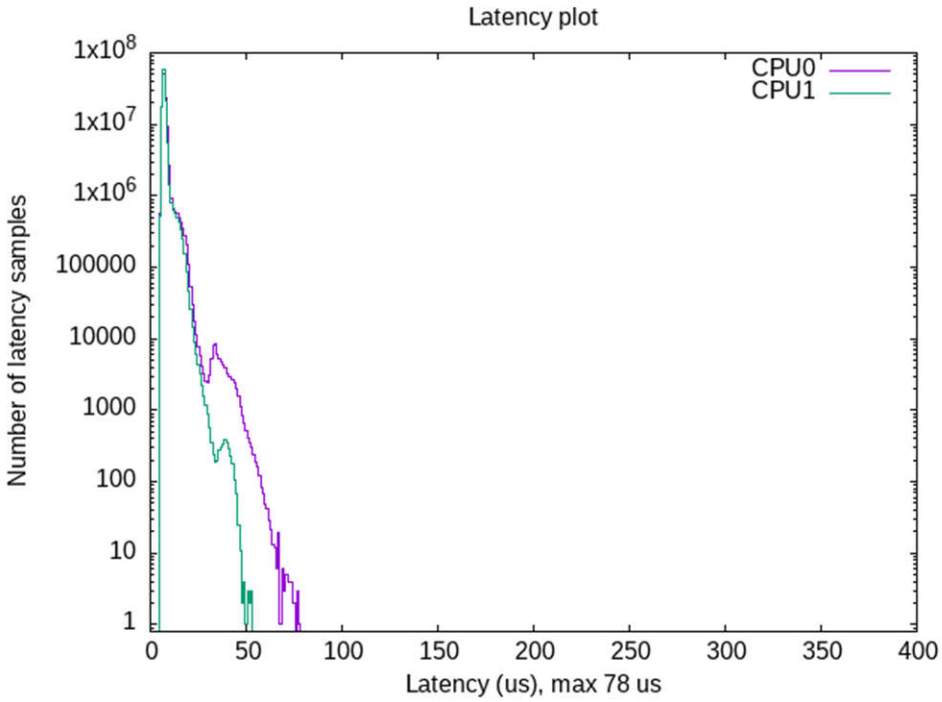


Figure 3-2. Latency Plot for AM64x Running for 6 Hours Under stress-ng

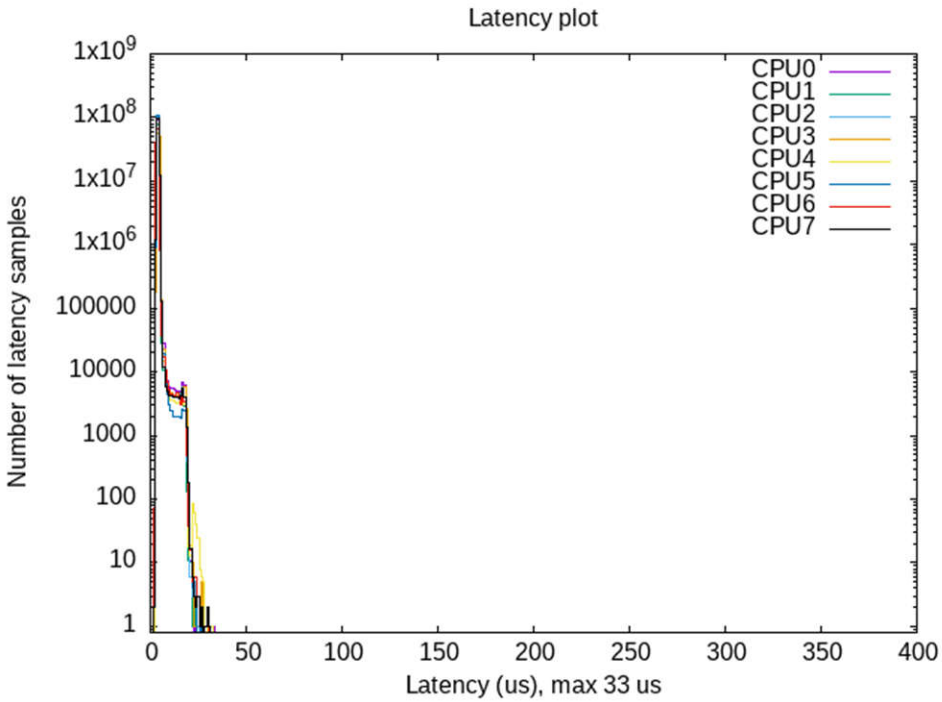


Figure 3-3. Latency Plot for AM69 Running for 6 Hours Under stress-ng

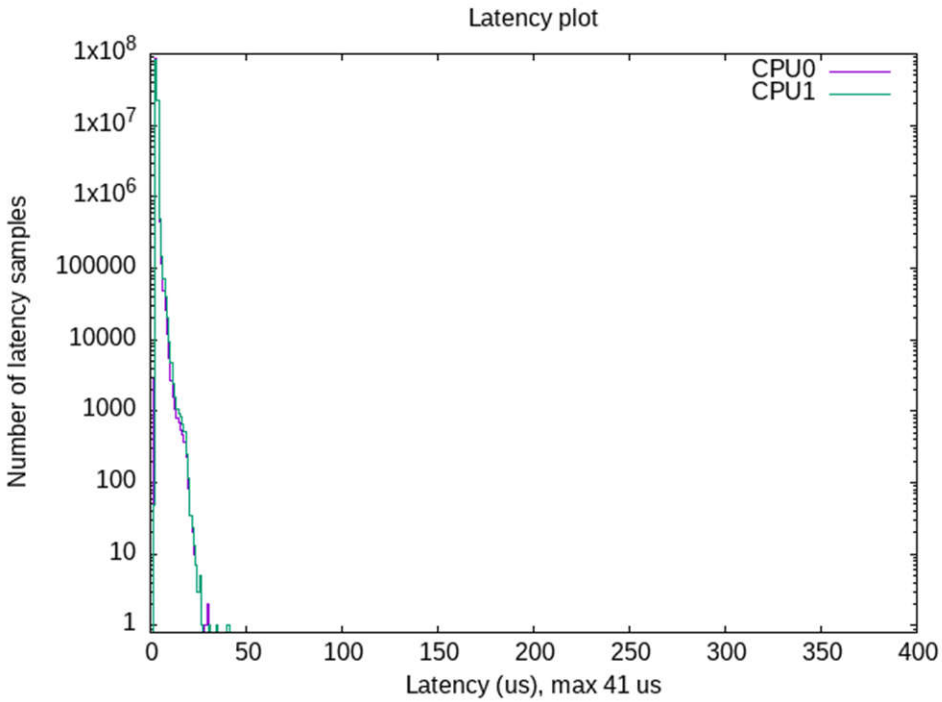


Figure 3-4. Latency Plot for the TDA4VM Running for 6 Hours Under stress-ng

3.2 EtherCAT Performance Metrics

Key performance indicators (KPI) of maximum measured cycle time and maximum jitter resulting from using the CODESYS EtherCAT stack are summarized in the following table, Table 3-1.

Table 3-1. CODESYS® EtherCAT Performance Summary (1ms cycle time)

HW	Run Time	ECAT Network	Max Cycle Time	Filtered Max Cycle Time[1]	Max Jitter	Filtered Max Jitter[1]
AM62x	63 hours	11x Beckhoff EtherCAT devices	700µs	500µs	116µs	80µs
AM64x	63 hours	11x Beckhoff EtherCAT devices	1906µs	865µs	973µs	112µs
AM69	63 hours	11x Beckhoff EtherCAT devices	384µs	250µs	53µs	45µs
TDA4VM	63 hours	11x Beckhoff EtherCAT devices	371µs	330µs	65µs	48µs

All TI hardware platforms serving as EtherCAT controllers from Table 3-1 were benchmarked with an EtherCAT device network of ten Beckhoff 16-channel digital output devices (EL2889) connected through a Beckhoff EtherCAT coupler (EK1100). All 160 channels toggled between high and low every 1 second. The cycle period was configured to 1000µs.

The results for AM62x and AM64x in Table 3-1 are from performance improvement tunings.

4 Optimizations

4.1 Implemented Optimizations

The AM62x and AM64x EtherCAT controllers are tuned for better performance by disabling several background processes and tuning CODESYS specific processes running in Linux on the ARM64 A53 cores. The main background processes that are disabled from the real-time “tisdk-default-image” Software Development Kit (SDK) version 09.01.00.08 image for AM62x are *ti-apps-launcher, pulseaudio, systemd-timesyncd, telnetd, weston, containerd.* These background processes are originally activated to showcase features irrelevant to an EtherCAT application. The purpose of disabling these processes is to reduce some of the CPU load on the four A53 cores and to configure the image as close to the real-time “tisdk-thinlinux-image” for AM62x. This is done because there is a slightly better cycle time performance from an average of about 350µs to about 250µs cycle time when comparing the default image to the thin image.

A major influence on cycle time is the CODESYS licensing application named Codemeter. Most EtherCAT applications that plan to use CODESYS require the use of running Codemeter to enable indefinite runtimes. Without the licensing application, CODESYS only runs for about 30 minutes before the system is terminated and must be restarted (see also, CODESYS Control Standard S). Codemeter is a necessary application to benchmark in an environment close to a real EtherCAT use-case; however, approximately every 5-6 minutes, the application spikes to near 100% CPU load which subsequently causes a spike in cycle time (from around 350µs to around 1000µs) and jitter. This spike can be observed through monitoring *htop* in Linux and the cycle time and jitter over time in the CODESYS Development System GUI.

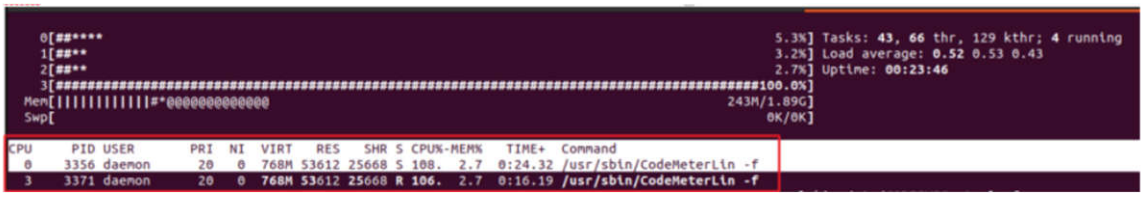


Figure 4-1. Codemeter CPU Spike in htop

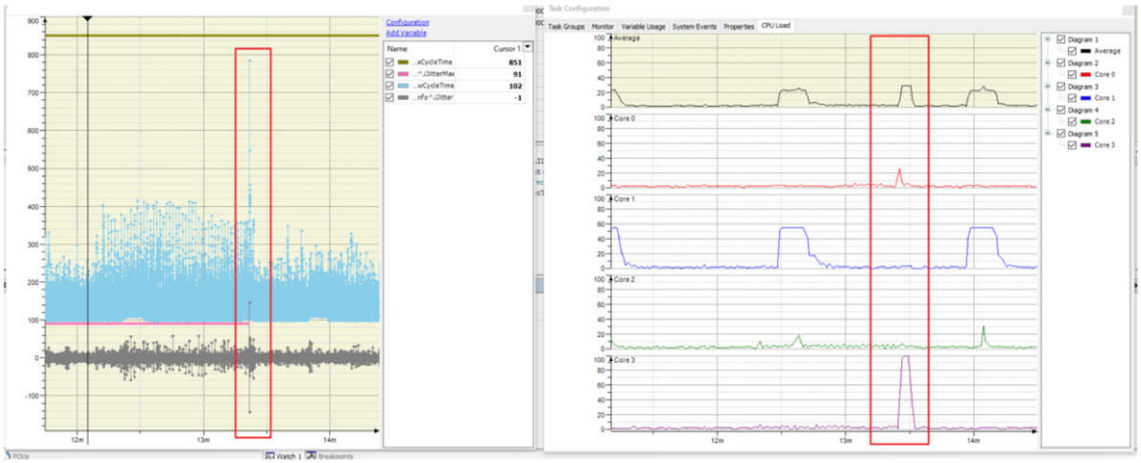


Figure 4-2. Spike in Maximum Cycle Time Due to Codemeter CPU Load Spike

Upon further investigation, the CPU affinity of Codemeter appears to change between CPU 0, 2, and 3 by default and never uses CPU 1. CPU 1 is the core that the CODESYS control application runs on by default. By changing the CPU affinity to CPU 1, the massive cycle time spike to around 1000µs reduces to around 500µs. The general spikes above 400µs in Figure 4-3 are due to running *htop* during those periods of time. The details on why CPU affinity changes reduce the spike is still unclear.

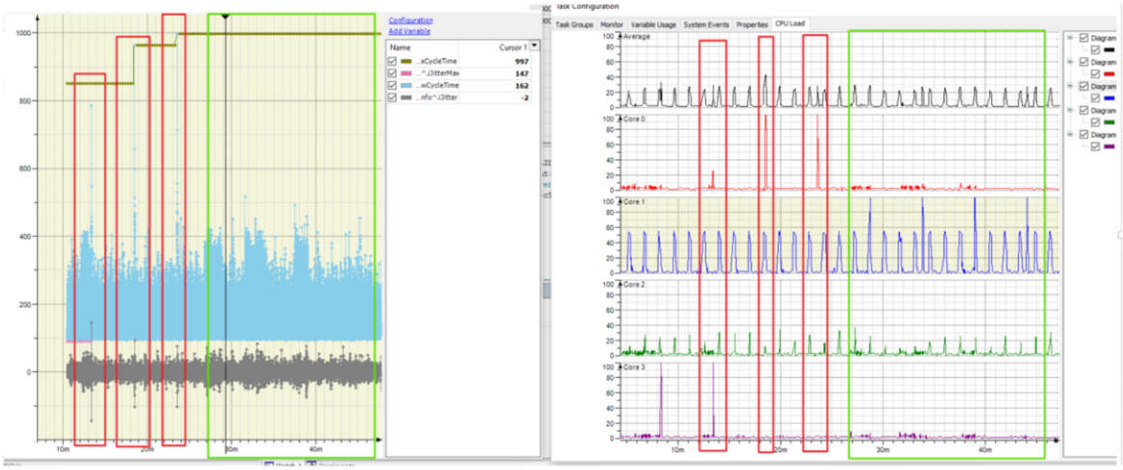


Figure 4-3. Codemeter Application on CPU 1 and Resulting KPI in Green Boxes and on CPU0 or 3 in Red Boxes

As an additional note, the maximum cycle time for all hardware platforms reported in Table 3-1 is an outlier related to the period of approximately 3 seconds during which the EtherCAT protocol starts up. This cycle time outlier includes the time when the devices cycles through several states from “Init” state to “Operational” state, based on the EtherCAT State Machine (ESM) and startup configuration is sent from the EtherCAT controller to the EtherCAT devices through acyclic mailbox protocols (see also, *EtherCAT | System Description*). Due to this, the focus on performance is when the device is in the steady state. A similar cycle time spike can also occur when logging out and back in through the CODESYS Development System interface. This use case is excluded in the analysis under the assumption that in a real factory automation environment, constant logout and login is not likely to occur.

The “Filtered Max Cycle Time” and “Filtered Max Jitter” columns in Table 3-1 refer to the approximate maximum cycle time after filtering out the startup outlier. These filtered values are based on the last 2 hours, 46 minutes, and 39 seconds of runtime, as the CODESYS Development System only has a buffer size large enough to store data points for this period of time.

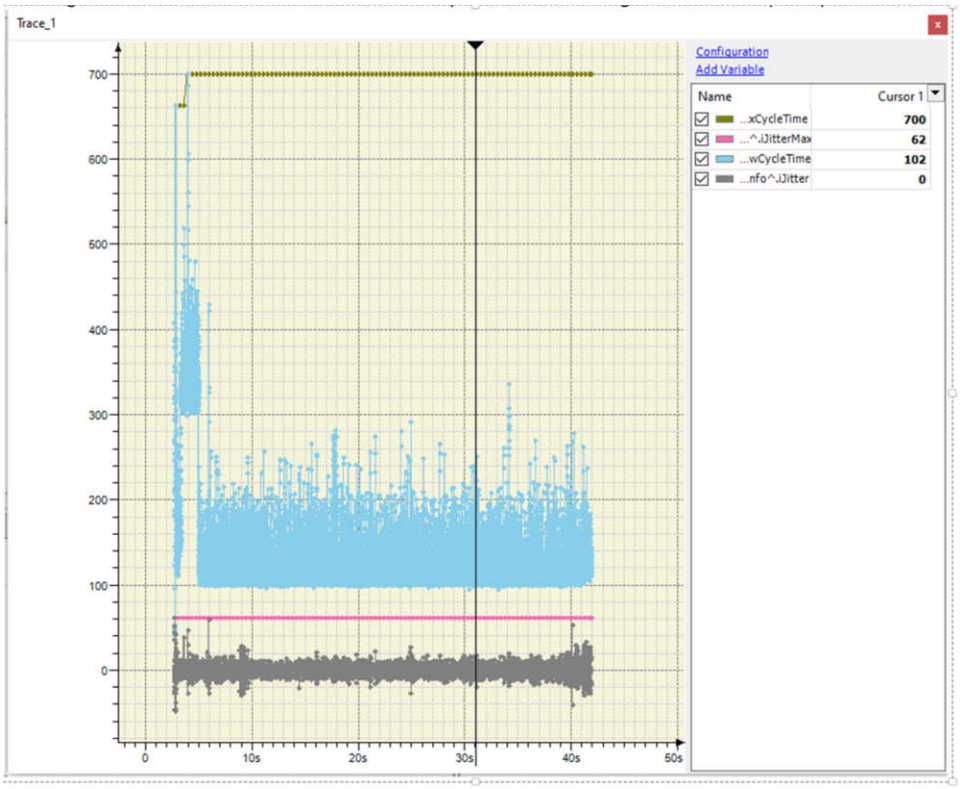
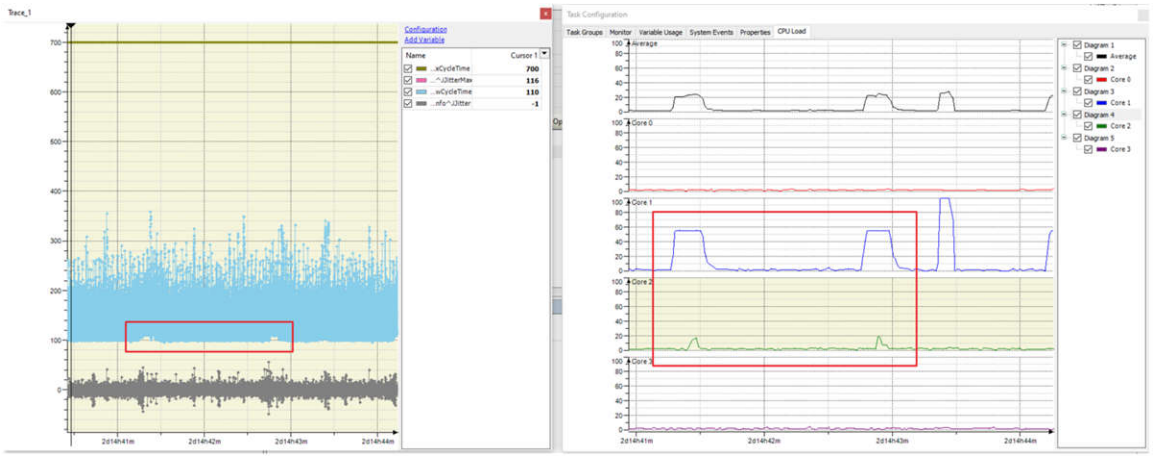


Figure 4-4. KPI During Startup on AM62x as EtherCAT Controller

Figure 4-5. KPI During Startup on AM69 as EtherCAT Controller

A snapshot of the final results of these optimizations for AM62x can be seen in Figure 4-6, where the maximum cycle time is the outlier of 700µs and the filtered maximum cycle time can be estimated to be around 500µs. There are still a number of periodic cycle time increases observed which have yet to be investigated.



Note

Includes periodic cycle time increases every 1 to 1.5 minutes as indicated by the red boxes.

Figure 4-6. KPI After 63 Hours of Runtime on AM62x as EtherCAT Controller

In general, the AM64x has lower performance than the AM62x, which is in part due to the reduction from 4 A53 cores to 2 A53 cores, resulting in less cores for various tasks required by EtherCAT and other background tasks to run. This difference in cores can impact the CPU load, resulting in the overall CPU load across all cores to be higher for AM64x than that of AM62x. Differences in cache size, DDR speed, and bus width can also play a factor in the performance differences between AM64x and AM62x. Additionally observed on the AM64x platform, during the 63-hour runtime, spikes in cyclic time ranging between 542µs to about 1032µs sporadically appear. The specific causes for this behavior on AM64x are still under investigation.

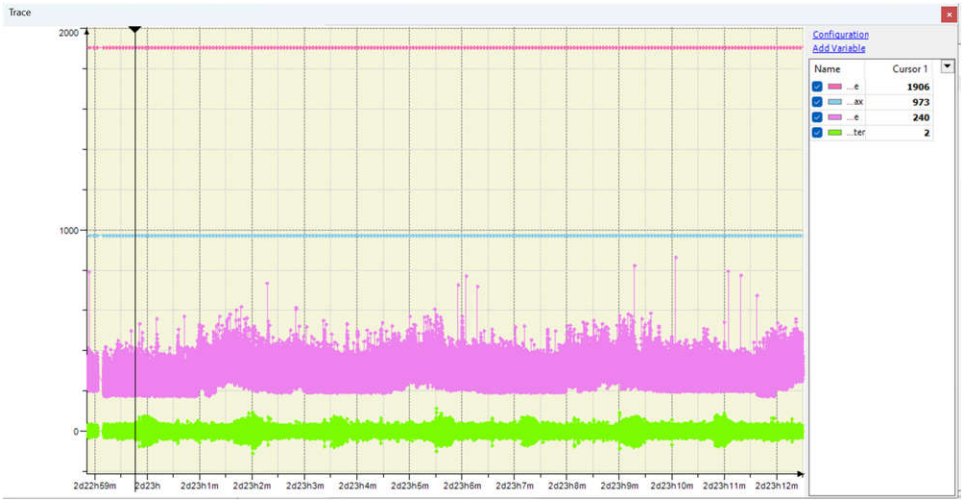


Figure 4-7. KPI on AM64x as EtherCAT Controller as Shown on CODESYS Development System

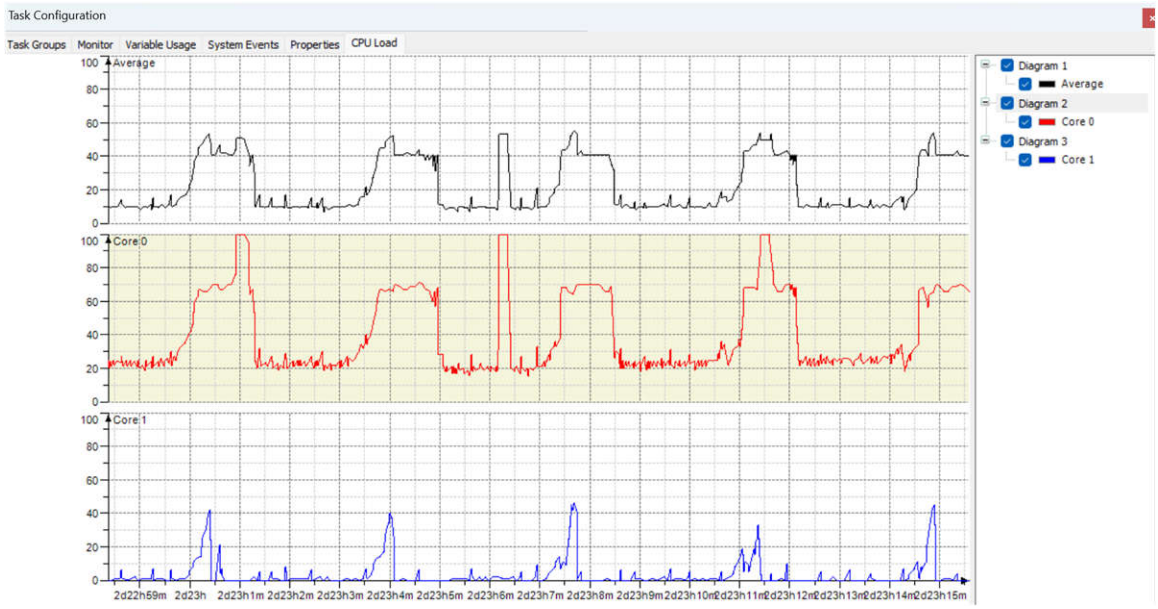


Figure 4-8. CPU Load of AM64x as EtherCAT Controller as Shown in CODESYS Development System

Note

These are the CPU load results from all CODESYS related tasks set to CPU0.

4.2 Future Considerations

There are several additional steps to gain better visibility into why the CODESYS EtherCAT stack is affecting the CPU load and KPI as described in the previous section. These steps have not yet been extensively investigated but are described here as potential future steps to take. Many of these steps are based on suggestions given on the CODESYS Optimization for Linux Systems resource.

The optimizations described in the previous section are predominately discovered through experimentation. What is currently known is that when the CODESYS application is started on each hardware platform, a list of several threads related to the application appear. Figure 4-9 shows an example of this observation using an htop capture. This htop is taken during the time the Codemeter application caused a spike in CPU load. CODESYS documentation refers to these threads as “IEC tasks”.

The thread with priority -56 is specifically the EtherCAT task scheduled with a FIFO priority scheme. This can be verified by checking on the CODESYS Development System and the associated priority found on the CODESYS Mapping of Task Priorities on a Linux System resource. Understanding the function of the other CODESYS threads (“IEC tasks”) is more difficult.

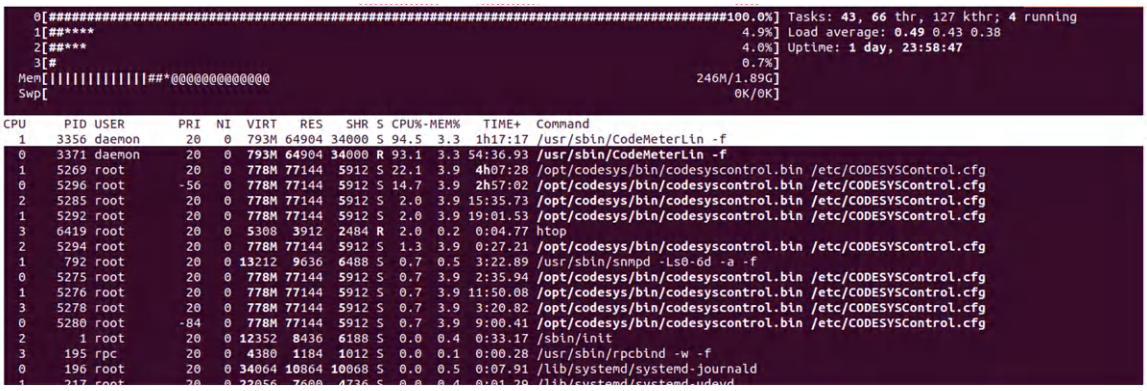


Figure 4-9. CODESYS Related Threads on AM62x

To gain more visibility, the following steps can be taken:

- Enable kernel tracing (ftrace) to detect if an interrupt or another service can be interfering with the performance of the EtherCAT task
- Implement CPU load tracing object in CODESYS Development System for better granularity of CPU load over time
- Run cyclictest while CODESYS EtherCAT application is running in the background to see if the metrics significantly differ from cyclictest running with stress-ng. The motivation for doing this task is to see if the metrics match up with EtherCAT cycle time and to see if the maximum latencies increased in a that is way indicative of large amounts of ethernet traffic.

Other experiments to try include the following. Keep in mind that for each experiment, first testing the results using *cyclictest* can be beneficial compared to the more extensive process of capturing data from CODESYS.

4.2.1 Set Maximum CPU Frequency

By default, the CPU frequency on all 4 A53 cores of AM62x runs at 1.25GHz on the default wic image from PROCESSOR-SDK-LINUX-RT-AM62x; however, AM62x is capable of running at a maximum of 1.4GHz. Change the CPU frequency to the maximum rating with “k3conf dump clock | grep A53” to check the clock numbers associated with each core and “k3conf set clock <clock number> 3 1400000000” with the clock number associated with each core to set the CPU frequency to 1.4GHz. Performing this procedure provides the best platform to process threads as quickly as possible and improve the KPI metrics.

4.2.2 Isolate Cores

Since IEC tasks competes with other threads scheduled by the Linux kernel (see CODESYS Task Configuration) which can impact the load on the cores the IEC tasks are operating on, isolating these other threads from the cores the IEC tasks are running from is best. Isolating a core from kernel scheduling can be accomplished using the command line parameter *isolcpus.* More specifically, the U-boot command `optargs="isolcpus=<core number>"` can be used.

4.2.3 Set CPU Affinity

When the IEC tasks are distributed over several CPU cores, the processing of IEC tasks by priority is no longer guaranteed (see Multicore). To prevent this situation, set the cpu affinity of all CODESYS threads to a single core. Setting the cpu affinity can be accomplished using the `taskset -p <cpu core number (1 indexed)> <Process ID>` command in the Linux shell.

This procedure is tested during a benchmarking on the AM64x by setting cpu affinity of all visible IEC tasks in htop to one core. However, every 4-5 minutes, there is still an increase of approximately 40% CPU load (see Figure 4-8) on the other core. During benchmark testing, the core containing the IEC tasks increases from ≅20% to ≅65% CPU load. Based on the increased CPU load, there can be additional threads in relation to the IEC tasks left running on the other core. The improvement in cycle time is difficult to track. That is, there are no obvious improvements to the cycle time observed.

4.2.4 Isolate Cores and Set CPU Affinity

This procedure combines tests from Section 4.2.3 and Section 4.2.2. For more simplicity, run this test on the AM64x, which features 2 A53 cores. One core must be isolated from kernel scheduling using the kernel command line parameter *isolcpus*. Additionally, set the cpu affinity of all CODESYS threads to the isolated core. The other threads scheduled by the kernel should only run on the other core.

4.2.5 Ksoftirqs to FIFO

By default, ksoftirqs associated with each core are scheduled as “TS”, “time-sliced”, “time-sharing”, or “SCHED_OTHER” scheduling policy (these are four different names for the same scheduling policy). This is a standard round-robin time-sharing policy that is intended for all processes that do not require special real-time mechanisms. Since ksoftirqs are used by ethernet-related interrupts that EtherCAT packets use, verifying that ksoftirqs are scheduled with as high a priority as possible in a real time scheme is important. Therefore, change from “TS” to “FIFO”, where possible. In other words, use “SCHED_FIFO” with priority higher than the -56 the EtherCAT task is set to. This procedure can be performed with the `chrt -f -p <priority> <process id>` command.

4.2.6 Increase the Real-Time Scheduling Time

The Linux kernel places a global limit on how much time threads with a real-time scheduling policy can use. The remaining time is used to schedule SCHED_OTHER processes. Increasing the allocated real-time scheduling time can help to prevent real-time threads from missing deadlines including the EtherCAT task thread and the ksoftirq threads. By default on RT-Linux, the real-time allocated runtime can be read from */proc/sys/kernel/sched_rt_runtime _us* to be 950000 or 0.95 seconds. The entire measurement period can be read from */proc/sys/kernel/sched_rt_period_us* to be 1000000 or 1 second. Based on these runtimes, by default, 0.05 seconds is reserved for SCHED_OTHER threads. Completely setting real-time runtime to be 1 second is not recommended; however, increasing the real-time runtime to 0.98 or 0.99 seconds can be helpful.

4.2.7 Disable irqbalance

While benchmarking on AM64x, irqbalance is observed on the list of top 10 CPU load heavy threads. Additionally, this feature automatically distributes load, generated by different interrupt sources across CPUs to achieve better system performance. This, however, can distribute load onto the same cores the EtherCAT task and other IEC tasks are running which in turn can negatively affect observed cycle time and jitter of the EtherCAT task.

4.2.8 Use Separate Network Interface Card (NIC)

During benchmarking on AM62x, one Common Platform Switch (CPSW) ethernet interface is connected the EtherCAT network and the other CPSW ethernet interface is connected to the PC running CODESYS Development System to view EtherCAT statistics. Due to how CPSW is designed with 2 external ports connected to a single internal port, all frames passing through the 2 external ports pass through the single internal port. This design prevents the possibility of distinctly isolating any ethernet related interrupts between the two external ports. As a result, isolating which CPU core the packets pass through is not possible. The only potential method to control which CPU core the packets get processed is through application level threads. If one application only uses one external port and the other application uses another, set the cpu affinity of each application to the desired CPU core. However, this setup still does not mean that the ethernet interrupts use the desired CPU core. Another reason for why the two external ports cannot be distinct, from an interrupt and CPU core perspective, is because both CPSW ports are using the same CPSW ethernet driver. Isolating the two external ports by using CPSW for one and Industrial Control Communication Subsystem - Gigabit (ICSSG) ethernet for the other port is possible as this strategy can use two separate ethernet drivers.

4.2.9 Disable Unnecessary Drivers

The Tips for Tuning The Real-Time Kernel guide states that the drivers used for unnecessary peripherals like remote cores, I2C controllers, MMC, and USB devices can take time away from the CPU running the target application. Disabling the devices in the device tree prevents the kernel from loading the corresponding drivers, thereby limiting the number of interrupts the CPU cores must service.

5 Summary

The resulting benchmarks across AM62x, AM64x, AM69, and TDA4VM indicates that configuring a 1ms cycle time is achievable for the case of filtering out results from CODESYS startup or log-in. After some optimizations, higher end processors such as AM69 and TDA4VM can achieve down to 500µs cycle times whereas lower end processors such as AM62x and AM64x can achieve 1ms cycle times. Additionally, AM62x performed better than AM64x in terms of measuring a much lower maximum cycle time of 500µs compared to about 900µs. AM62x features four A53 cores running at a higher clock speed of 1.4GHz compared to AM64x at only two A53 cores running at 1GHz. The higher number of cores to redistribute “IEC tasks” that the CODESYS EtherCAT controller application uses and higher clock speed could have contributed the improvement in cycle time by about 400µs.

The expectation is to align the worst-case interrupt latency observed from cyclictest results with the cycle time observed from running CODESYS EtherCAT stack. The actual results show that CODESYS adds a significant amount of additional jitter. Further tunings such as changing clock frequency can improve cycle time performance; however, a better understanding of how CODESYS related threads impact scheduling and CPU load, helps in determining how to further optimize cycle time performance.

Note that the analysis provided in this application note are from the CODESYS EtherCAT controller stack running on Linux using a generic ethernet driver. Running an EtherCAT controller stack with a native driver can

show even better performance. This is already shown by an RTOS-based EtherCAT controller stack with an optimized native driver developed by IBV (stack name is icECAT), reaching 100µs cycle time when running on a high-end MCU-based platform. With a more focused commercial stack such as IBV icECAT, better performance can also be achieved on a system running Linux, even with the standard ethernet driver (see also *icECAT EtherCAT Master Stack Benchmark*).

6 References

1. Acontis, *EtherCAT Master Stack*, web page
2. IBV icECAT, *EtherCAT® Master Stack for Embedded Systems*, web page
3. RealPars, *CODESYS Basics | What is CODESYS and Why is it Important?*, web page
4. The Linux Foundation, *Cyclictest*, web page
5. Open Source Automation Development Lab (OSADL) eG, *OSADL QA Farm on Real-time of Mainline Linux*, web page
6. Open Source Automation Development Lab (OSADL) eG, *Home*, web page
7. CODESYS, *CODESYS Control Standard S*, web page
8. Beckhoff, *EtherCAT | System Description | EtherCAT State Machine*, web page
9. CODESYS, *Optimization for Linux Systems*, web page
10. CODESYS, *Mapping of Task Priorities on a Linux System*, web page
11. CODESYS, *CODESYS Task Configuration,* , web page
12. CODESYS, *Multicore*, web page
13. Texas Instruments, *How to Tune Real Time Linux- Disable Unnecessary Drivers,* , web page
14. IBV - Echtzeit, icECAT EtherCAT Master Stack Benchmark, document

7 Appendix A: How to Setup TI Embedded Processors as EtherCAT Controller Using the CODESYS Stack

7.1 Hardware Requirements

Table 7-1 describes the hardware required to setup a similar EtherCAT network described in this application note. Similar hardware can be required for some of the other hardware platforms described in this application note.

Table 7-1. Hardware Requirements for TI Processors as EtherCAT Controller

Item	Quantity	Description	Link
SK-AM62B-P1	0-1	AM62x device to serve as EtherCAT controller	https://www.ti.com/tool/SK-AM62B-P1
TMDS64EVM/SK-AM64B	0-1	AM64x device to serve as EtherCAT controller	https://www.ti.com/tool/TMDS64EVM https://www.ti.com/tool/SK-AM64B
SK-AM69	0-1	AM69 starter kit to serve as EtherCAT master	https://www.ti.com/tool/SK-AM69
TDA4VM	0-1	TDA4VM to serve as EtherCAT master	https://www.ti.com/tool/J721EXCPXEVM https://www.ti.com/tool/J721EXSOMXEVM
Beckhoff EK1100	1	EtherCAT Coupler	https://www.beckhoff.com/en-us/products/i-o/ethercat-terminals/ek1xxx-bk1xx0-ethercat-coupler/ek1100.html
Beckhoff EL2889	10	EtherCAT Terminal, 16-channel digital output	https://www.beckhoff.com/en-us/products/i-o/ethercat-terminals/el2xxx-digital-output/el2889.html
Linux PC	0-1	Optional Linux PC to serve as DHCP server	N/A
Windows PC	1	64-bit Windows PC to run the CODESYS Development System	N/A
Ethernet Switch	0-1	Optional Ethernet Switch to connect CODESYS Development System to EtherCAT network	N/A
Micro USB-B to USB-A Adapter	0-1	Required when using CODESYS License USB dongle on TMDS64EVM as EtherCAT Controller	N/A
CODESYS Key	1	USB dongle for secure storage of CODESYS licenses based on CodeMeter technology	https://us.store.codesys.com/codesys-key.html
Ethernet Cables	2+	At least 2 CAT5 or CAT6 ethernet cables for the EtherCAT network. More required if an ethernet switch and additional EtherCAT devices are added	N/A
12V (up to 8A) power supplies with 5.5mmx2.5mmx9.6mm barrel jack	0-1	12V power supply for TMDS64EVM or TDA4VM	N/A
5V-15V (up to 3A) power supplies	0-1	5V power supply for SK-AM64B, SK-AM62B-P1, SK-AM69	N/A

7.2 Software Requirements

Table 7-2 describes software used by to setup the same EtherCAT network as described in this application note. Similar software is required for some of the other hardware platforms described in this application note. Note that

the specific software version described in the table is what has been tested and verified to work. Newer versions could also work but have not been specifically tested.

Table 7-2. Software Requirements for TI Processors as EtherCAT Controller

Item	Version	Description	Link
AM64x/AM62x TI Processor SDK RT-Linux Image	09.01.00.008	Default RT-Linux wic image to etch onto the SD card	https://www.ti.com/tool/download/PROCESSOR-SDK-LINUX-RT-AM64X/09.01.00.08 https://www.ti.com/tool/download/PROCESSOR-SDK-LINUX-RT-AM62X/09.01.00.08
AM69x/TDA4VM TI Processor SDK RT-Linux Image	09.01.00.07	Real-time build of SDK 09.01.00.07 image etched onto SD card	See footnotes (1) (2) (3)
CODESYS Development System V3	3.5.19.10 64-bit	Programming tool for industrial control and automation technology GUI	https://us.store.codesys.com/codesys.html
CODESYS Control for Linux ARM SL	4.9.0.0 64-bit	SoftPLC for Linux/ARM based industrial controllers	https://us.store.codesys.com/codesys-control-for-linux-arm-sl-1.html

(1) There is no link for AM69 or TDA4VM RT Linux image releases on ti.com. Generating an RT Linux image requires the user to follow the steps outlined in the "Building the SDK with Yocto" section of the Processor SDK Linux documentation (specifically keeping the "RT Support" subsection in mind).
(2) See also *TDA4VM SDK 09.01* documentation
(3) See also *AM69 SDK 09.01* documentation

7.3 Hardware Setup

The EtherCAT network must be setup in a Tree, Line, or Star topology. Loop topology can also be used for redundancy purposes. See Figure 2-1 for the same network described in this application note.

7.4 Software Setup

The following provides instructions on how to setup a CODESYS EtherCAT project with TI embedded processors as the EtherCAT controller. Note that CODESYS is a commercial software. For any issues related to CODESYS specific setup, please directly contact CODESYS technical support.

7.4.1 Windows PC Setup

1. Create an account with CODESYS
   a. A "Corporate Customer" account require4s approval before you can download and install any software
   b. An "Individual Customer" account allows downloading and installing all the required software
2. Download the CODESYS Development System V3

Note

Recommended: Download the **.zip** version as the **.exe** version can cause your Windows PC to crash when trying to run

3. Run the CODESYS Development System V3 installer as an administrator
   a. Administrator rights allows installer to install dependencies user does not already have installed on their system
   b. Keep in mind that virus prevention software might interfere with CODESYS installation. If CODESYS installation fails, try disabling any virus prevention software
   c. These dependencies potentially include in Figure 7-1

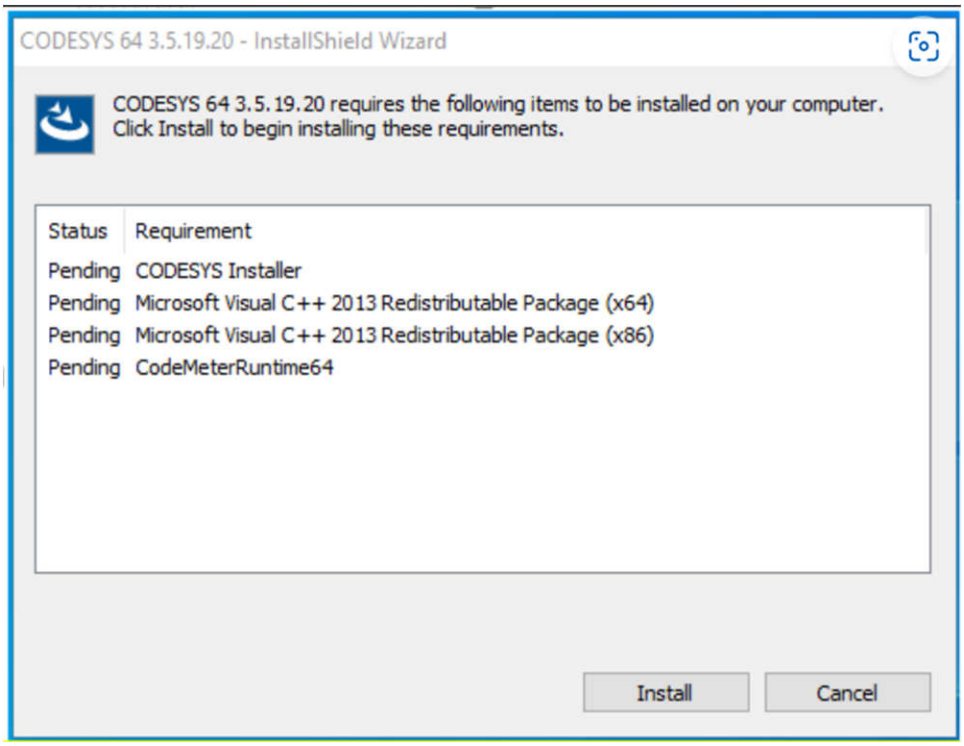


Figure 7-1. CODESYS Development System Installation Dependencies

d. Trying to run as administrator when not granted full administrator rights causes the Windows PC to freeze up and crash

4. Download the CODESYS Control for Linux ARM SL
5. Install CODESYS Control for Linux ARM SL into the previously installed CODESYS Development System using the following steps:
   a. Double-click the downloaded **CODESYS Control for Linux ARM64 SL <version>.package** and select *CODESYS 64 <version>.*
   b. Click *Continue* to install the package

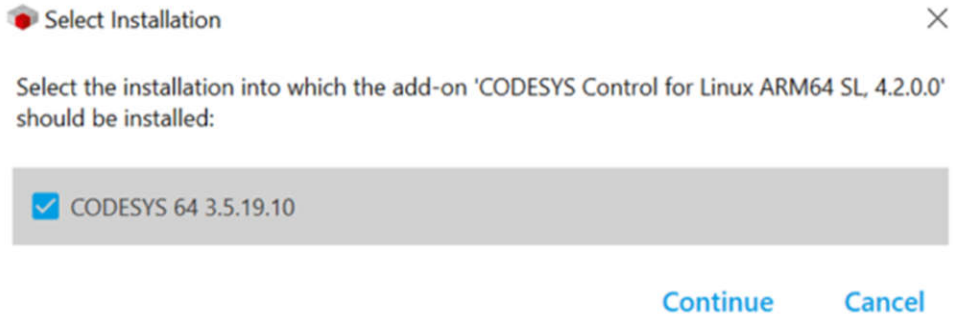


Figure 7-2. CODESYS Installation Version

6. Copy required files from the CODESYS Control for Linux ARM SL package to the device to be set up as an EtherCAT controller with the following steps:
   a. Extract the contents of **CODESYS Control for Linux ARM64 SL <version>.package** using an utility like 7-Zip
   b. Navigate to the **CODESYS Control for Linux ARM64 SL** directory within the **CODESYS <version>** program files
   c. Place these two files anywhere in the root directory of the EtherCAT controller:
      i. **codemeter-lite_<version>_arm64.deb**: found within the **'CODESYS Control for Linux ARM64 SL [<version>]'/Dependency** directory
      ii. **codesyscontrol_linuxarm64_<version>_arm64.ipk**: found within the **'CODESYS Control for Linux ARM64 SL [<version>]'/Delivery/[linuxarm64]** directory

d. A file transfer protocol (scp, sftp, etc) can be used to copy the previous files to the EtherCAT controller or these files can be directly placed on the SD card containing the OS the EtherCAT controller is running on

7.4.2 EtherCAT Controller Setup

1. Flash an SD card with the default RT Linux SDK image for the EtherCAT controller using a tool such as BalenaEtcher
   a. For AM62x/AM64x devices
      i. If the device is a *GP* device, the *GP* version of *tiboot3.bin* must be renamed to *tiboot3.bin*, replacing the original *tiboot3.bin*
      ii. If the device is a *HS-FS* device, the *tiboot3.bin* can be left as the default version
   b. For AM69x devices
      i. The default *tiboot3.bin* must be replaced with the *HS-FS* version of *tiboot3.bin*
      ii. For example:
         1. `$ cd <path-to-boot-partition>`
         2. `$ mv tiboot3-j784s4-hsfs-evm.bin tiboot3.bin`
2. After the required files from the CODESYS Control for Linux ARM SL package has been copied into the root directory of the device to be set up as an EtherCAT controller and the device has been booting up with the SD card, install the packages required to setup the device as EtherCAT controller with the copied over files. Run the following commands on the target device
   a. `$ opkg -v2 install <path-to-file>/codemeter-lite_<version>_arm64.deb`
      i. For AM62x devices, if there is a "*does not have a compatible architecture*" error, run the following command instead
         1. `opkg -v2 install --nodeps --offline-root / --add-arch arm64:13 <path-to-file>/codemeter-lite_<version>_arm64.deb`
   b. `$ opkg -v2 install <path-to-file>/codesyscontrol_linuxarm64_<version>_arm64.ipk`
3. Start the CODESYS application on the EtherCAT controller with one of the following commands
   a. `$ /opt/codesys/bin/codesyscontrol.bin /etc/CODESYSControl.cfg`
   b. `$ systemctl start codesyscontrol.service`
   c. `$ /etc/init.d/codesyscontrol start`
4. Verify that the CODESYS application is running before attempting to connect the EtherCAT controller to the CODESYS Development System on your Windows PC
5. Verify that the CodeMeter application (on your EtherCAT controller device) is running with an activated CODESYS license on the EtherCAT controller. See Section 8 on how to activate a CODESYS license. Once a CODESYS license has been activated, use the following command to start the licensing application:
   a. `$ systemctl start codemeter.service`

7.4.3 CODESYS Development System Project

1. Set up an EtherCAT network similar to what is shown in Section 7.3
2. Start the CODESYS control service on the EtherCAT controller with one of the following commands

```
$ /opt/codesys/bin/codesyscontrol.bin /etc/CODESYSControl.cfg
$ systemctl start codesyscontrol.service
$ /etc/init.d/codesyscontrol start
```

3. Open CODESYS Development System V3 on your Windows PC
4. Create a new standard project with the following steps
   a. Click on **"File > New Project"**
   b. Select **"Standard project"**
   c. Select a name for the project and where to store the project
   d. Select **"CODESYS Control for Linux ARM64 SL (CODESYS GmbH)"** for **"Device"** entry
   e. Select **"Structured Text(ST)"** for **"PLC_PRG in"** entry
   f. Click **"OK"**

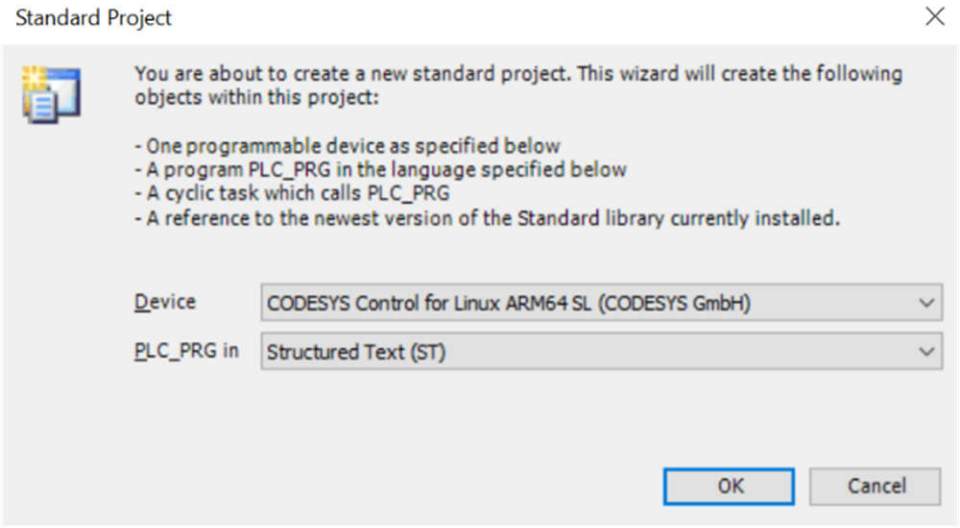


Figure 7-3. CODESYS Standard Project

5. Add the EtherCAT device ESI (EtherCAT Slave Information) files to the project with the following steps
   a. For AM243x/AM64x as EtherCAT device
   b. Verify that AM243x MCU+ SDK or AM64x MCU+ SDK have been installed
   c. Click on **"Tools > Device Repository > Install"**
   d. Navigate to and select (Similar step for AM64x EtherCAT devices)

```
<path-to-AM243-MCU+-SDK>/source/industrial_comms/ethercat_slave/stack/esi/'TI AM243X.R5F Simple.xml'
```

6. Click **"Open"**
7. Click **"Close"**
8. For other EtherCAT devices (for example, Beckhoff devices), obtain and download the ESI files and install to Device Repository in a similar manner
9. Scan for EtherCAT controller device ("Origin" Device) with the following steps
   a. In the **"Devices"** window, double click **"Device (CODESYS Control for Linux ARM64 SL)"**
   b. Go to the **"Communication Settings"** and click on **"Scan Network"** tab
   c. The name of the EtherCAT controller device running the CODESYS control service shows up under *Gateway-1*
      i. If the EtherCAT controller does not appear, uncheck **"Hide non-matching devices, filter by Target ID" then click "Scan Network"**
   d. Select the device and click **"Ok"**
   e. If prompted about **"user management"**:
      i. Click **"Yes"**
      ii. Create a username and password
         1. The username and password can be changed by right clicking "*Device (CODESYS Control for Linux ARM64 SL)"* and clicking "*Reset Origin Device [Device]"*
      iii. Click **"OK"**
      iv. Log in
10. Add an EtherCAT_Master device with the following steps
    a. In the **"Devices"** window, right click **"Device (CODESYS Control for Linux ARM64 SL)"**
    b. Click **"Add Device"**
    c. Expand **"Fieldbuses > EtherCAT > Master > EtherCAT Master"**
    d. Click **"Add Device"**
11. Setup your EtherCAT controller as the EtherCAT_Master by matching the MAC address of the Ethernet port that is connecting with the rest of the EtherCAT devices.
    a. In the **"Devices"** window, double click "**(CODESYS Control for Linux ARM64 SL)"**
    b. Click the **"General > EtherCAT NIC Settings > Source address (MAC) > Select"** button
    c. Select the MAC address and ethernet port that connects to the rest of the system

12. Setup the rest of the EtherCAT devices in network through scanning for devices
    a. Click **"Toolbar > Login (gear icon)"** or **"Alt + F8"**
        i. Click **"Yes"**, if a window appears that states: "No online change possible due to severe changes: Do you want to perform a download?"
    b. Within the **"Devices"** window, right click on **"EtherCAT_Master (EtherCAT Master)"**
    c. Click **"Scan for Devices..."**
    d. Click **"Copy All Devices to Project"**
    e. If the EtherCAT devices do not appear under the list of scanned devices
        i. Try to power cycle the EtherCAT devices
        ii. Try to reflash the boards
13. Obtain process data from the EtherCAT devices with the following steps
    a. Click **"Toolbar > Start"** or **"F5"**
    b. In the **"Devices"** window, double click one of the EtherCAT devices
    c. Select **"General > Address > Additional > Expert settings"**
    d. Click **"Expert Process Data > Load PDO from the Device"**
    e. Repeat these steps to select **"Expert Process Data"** for each EtherCAT device
    f. Press **"Ctrl + S"**
    g. Click **"Toolbar > Stop"** or **"Shift + F8"**
    h. Click **"Toolbar > Logout"** or **"Ctrl + F8"**
14. Mapping the variables within the **"EtherCAT I/O Mapping"** tab for each device can be accomplished by writing a custom PLC_PRG program
    a. In the **"Devices"** window, double click **“PLC_PRG”**
    b. Contact CODESYS for more details on how to write a PLC_PRG program

7.4.4 Execution

1. Start the CODESYS control service on the EtherCAT controller with one of the following commands

```
$ /opt/codesys/bin/codesyscontrol.bin /etc/CODESYSControl.cfg
$ systemctl start codesyscontrol.service
$ /etc/init.d/codesyscontrol start
```

2. Open the CODESYS project
3. Within the **"Devices"** window, double click **"Device (CODESYS Control for Linux ARM64 SL)"**
4. Click the **"Communication Settings > Scan Network"** tab
5. Click on the controller device (AM64x or AM62x device)
6. Click **"OK"**
7. Log in if necessary
8. Within the **"Devices"** window, double click **"EtherCAT_Master (EtherCAT Master)"**
9. Click the **"General > EtherCAT NIC Settings > Source address (MAC) > Select..."** button
10. Select the MAC address and ethernet port that connects to the rest of the system (connects to the EtherCAT network)
11. Within the **"Devices"** window, double click **"MainTask (IEC-Tasks) > PLC_PRG"**
12. Click **"Toolbar > Login (gear icon)"** or **"Alt + F8"**
13. Click **"Toolbar > Start"** or **"F5"**

7.5 How to View Performance Measurements

To view basic cycle time and jitter statistics in table format

1. Within the **"Devices"** window, double click **"Task Configuration"**
2. Click on the **"Monitor"** tab

To view frame specific statistics

1. Within the **"Devices"** window, double click **"EtherCAT_Master (EtherCAT Master)"**
2. Click on the **"Status"** tab

To graphically view CPU Load measurements in real-time

1. Within the **"Devices"** window, double click **"Task Configuration"**

2. Click on the **"CPU Load"** tab. Note that this tab only appears when the CODESYS project is logged in and started

To graphically view cycle time and jitter statistics in real-time

1. Add “CmpIecTask” library manager
2. Put the following in the "Var" section in your PLC_PRG

```
PROGRAM PLC_PRG
   VAR
    tTask : Task_Info2;                                              (* Task Info *)
    aIecTasks : ARRAY[1..20] OF Task_Info2;                          (* All Task Info *)
    hCurrentTask : RTS_IEC_HANDLE := SysTypes.RTS_INVALID_HANDLE;
    Result : RTS_IEC_RESULT;                                         (* Result Code *)

    pTaskInfo : POINTER TO Task_Info2;
    hIecTask  : RTS_IEC_HANDLE;
    pResult   : POINTER TO RTS_IEC_RESULT;

     ...

   END_VAR
```

3. Place the following text in the body of the PLC_PRG

```
...

(* Retrieve information about the current task *)
IF hCurrentTask = SysTypes.RTS_INVALID_HANDLE THEN
        hCurrentTask  := IecTaskGetCurrent(pResult:=ADR(Result));
        pTaskInfo     := IecTaskGetInfo3(hIecTask:=hCurrentTask, pResult:=ADR(Result));
END_IF

...
```

4. Add a Trace object to your CODESYS project
   a. Right click on **“Application”** on the left hand panel
   b. Select **“Add Object” > “Trace”**

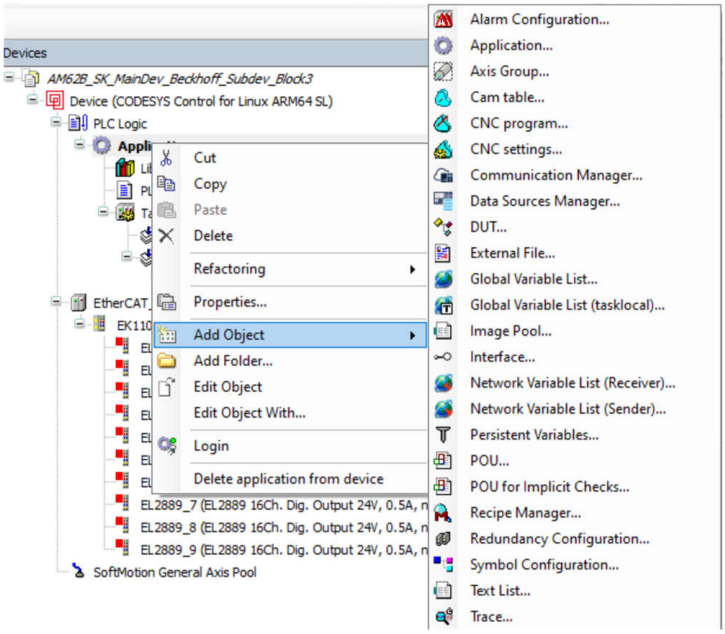


Figure 7-4. CODESYS Add Trace Object

   c. Under **"Task for Trace Recording" select the Task containing PLC_PRG**
   d. Select **"pTaskInfo->cycle time"** and **"pTaskInfo->jitter"** to trace in the Trace object
      i. Double click on the **“Trace”** object under **“Application”** in the left hand panel
      ii. Right click anywhere on the Trace object graph and select **“Add Variable”**
      iii. Next to the **“Variable”** entry, click on the 3 dots

iv. Select the variable corresponding to **"pTaskInfo->cycle time"** and **"pTaskInfo->jitter"**

To start the graphical interface in Trace object

a. Right click anywhere on the Trace object graph and select **“Download Trace”**

To save the trace object content as csv or txt file

a. Right click anywhere on the Trace object graph and select “**Save Trace”**

7.5.1 Appendix A Resources

TI Developer Resources

- Texas Instruments, *[FAQ] How to set up an EtherCAT System with AM6xx MainDevice and AM64/AM243 SubDevice using CODESYS Development System*, blog
- Texas Instruments, *CODESYS Industrial Communications Controller*, online documentation

Creating Trace Object in CODESYS Resources

- CODESYS, *Example: Task Manager*, online documentation
- CODESYS, *CODESYS Trace*, online documentation
- CODESYS, *Trace Example*, online documentation
- CODESYS, *Trace Example*, store page
- CODESYS, *Convert Array Of DINT To STRING*, blog
- CODESYS, *Data Type: ARRAY OF*, online documentation
- Tohid Alizadeh, *CODESYS: PLC Cycle time using Cycle_time function block from OSCAT library*, video
- Tohid Alizadeh, *CODESYS: Saving and Loading data using Trace*, video
- Zhou Gong, *CODESYS Tutorial: Getting the Actual Cycle Time of a Current Task*, blog

8 Appendix B: How to Enable Unlimited Runtime on CODESYS Stack

8.1 CODESYS Licensing Background

- Without purchasing a CODESYS License for the CODESYS Runtime system on the EtherCAT controller, the EtherCAT network only runs in Demo mode
- Running in Demo mode means that fieldbus devices such as EtherCAT devices terminate at 30 minutes
- As of December 2023, CODESYS has migrated to a new way of licensing across different hardware platforms. The new licenses are Application Based Licenses
- A quick method to check if the application is running in Demo mode is if there is an orange cycle symbol next to "EtherCAT_Master (EtherCAT_Master)" device in the device tree

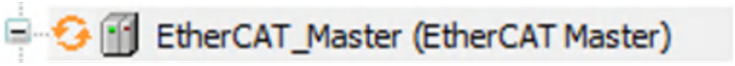


Figure 8-1. CODESYS Demo Mode Symbol

- If the application is running with a license a green cycle symbol is shown

Figure 8-2. CODESYS Licensed Mode Symbol

- Note that the CODESYS Development System that runs on the Windows PC for GUI purposes does not require a CODESYS license

8.2 Obtaining a CODESYS License

- Go to Explanation Application-Based Licenses | CODESYS Store International and select a license package
- For test and demo purposes, recommended licenses are
  - CODESYS Control Standard S (for single EtherCAT controller)
  - CODESYS Control Standard L (if two Ethernet Fieldbus networks are required)
- A list of other available licenses is found here Application Based Licenses
- Purchase and download the license package to obtain the license code

8.3 Activating CODESYS License

8.3.1 Background

- There are two methods of activating the license. See this link for more details on these two methods: Licensing of Products (helpme-codesys.com)
  - Online activation - requires the CODESYS Development System to have internet access, the EtherCAT controller (target system) does not require internet access
  - Offline activation - requires using CodeMeter Control Center, which is installed with the CODESYS Development System installation
- Online activation through the License Manager diaglog on the CODESYS Development System can not be possible if the CODESYS Development System connected to the EtherCAT network with a direct Ethernet connection or through a switch without DHCP router services.
- The license must also be activated on a device (EtherCAT controller) or USB dongle. CODESYS calls what the license is activated on as a "container". A device is called "soft container" and the USB dongle is called "Dongle" or "Key".
- The key differences between activation on a Soft Container and activation on a Dongle are listed below. The same list can be found at this site CODESYS Key

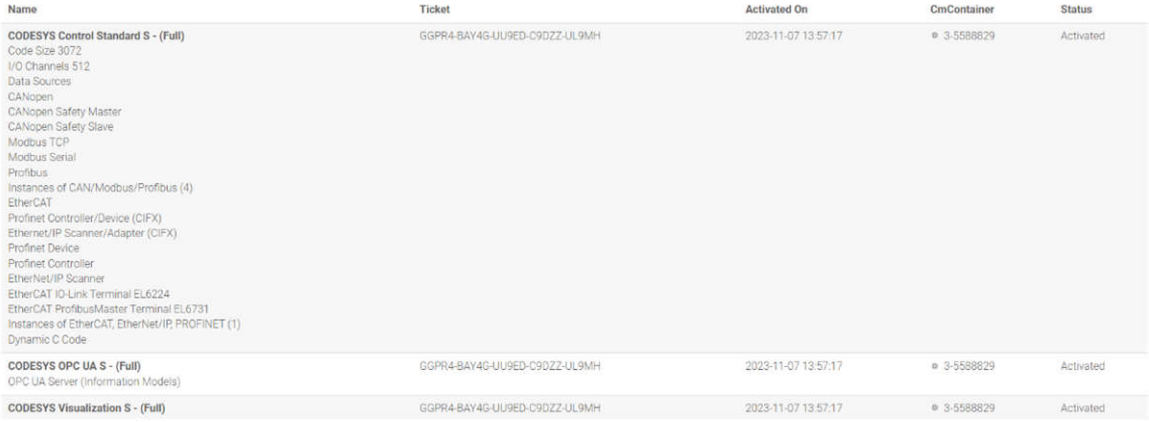
Table 8-1. Advantages and Disadvantages for Different Activation Methods

ACTIVATION METHOD	ADVANTAGES	DISADVANTAGES
Software license container (SoftContainer)	• No additional hardware required • No additional costs • No physical delivery necessary • Unlimited number of storable licenses • License usable immediately • No loss of license, in case of theft of a dongle	• License holder independent of the target devices • Easy transfer of the license to another device • License remains valid even in the case of defect of the device • Number of storable licenses almost unlimited (approximately 4000) • Use of the CODESYS Key also for security tasks (including project encryption)
CODESYS Key (dongle)	• No transfer of the license to another device • Loss of the license in case of a defect of the device	• Need for additional hardware and matching slot (USB) • Additional costs for key, shipping, and customs fees (if applicable) • Waiting time before using the license until delivery of the CODESYS key • Loss of the license in case of theft of the dongle

8.3.2 Recommended Steps

The following steps detail a known working method uses a combination of Online and Offline activation

- Order a CODESYS Key (USB dongle) here CODESYS Key
- Obtain License Code
- Plug in the CODESYS Key into the PC that is used for accessing CodeMeter Control Center and entering the CodeMeter server CodeMeter License Central WebDepot v23.03.554.500.ws4 (codesys.com) from
- Open CodeMeter Control Center and find the name of the associated with the CODESYS Key. Use this name in the next steps.
- Go to CodeMeter License Central WebDepot v23.03.554.500.ws4 (codesys.com) and enter the license code
- Click on "Activate Licenses"
- Click on the "CmDongle" option
- Find the name associated with the CODESYS Key from the "Select CmContainer" drop down menu; this is the same name as in the name found in the CodeMeter Control Center
- Click on "ACTIVATE SELECTED LICENSES NOW"
- The license must now be activated on the CODESYS Key. Verifying the license activation requires entering the activated license code into the CodeMeter License Central WebDepot to see if the status column is "Activated" and the CmContainer column states the name of the CODESYS Key. See the following as an example



Name	Ticket	Activated On	CmContainer	Status
CODESYS Control Standard S - (Full) Code Size 3072 I/O Channels 512 Data Sources CANopen CANopen Safety Master CANopen Safety Slave Modbus TCP Modbus Serial Profibus Instances of CAN/Modbus/Profibus (4) EtherCAT Profinet Controller/Device (CIFX) Ethernet/IP Scanner/Adapter (CIFX) Profinet Device Profinet Controller EtherNet/IP Scanner EtherCAT IO-Link Terminal EL6224 EtherCAT ProfibusMaster Terminal EL6731 Instances of EtherCAT, EtherNet/IP, PROFINET (1) Dynamic C Code	GGPR4-BAY4G-UU9ED-C9DZZ-UL9MH	2023-11-07 13:57:17	3-5588829	Activated
CODESYS OPC UA S - (Full) OPC UA Server (Information Models)	GGPR4-BAY4G-UU9ED-C9DZZ-UL9MH	2023-11-07 13:57:17	3-5588829	Activated
CODESYS Visualization S - (Full)	GGPR4-BAY4G-UU9ED-C9DZZ-UL9MH	2023-11-07 13:57:17	3-5588829	Activated

Figure 8-3. Activated CODESYS License

8.4 Verifying CODESYS License Applied

To test that the license is applied to the CODESYS Runtime system running on the EtherCAT controller perform the following steps

1. Plug the CODESYS Key (USB Dongle) into the EtherCAT controller
2. On the EtherCAT controller Linux terminal run the following command to start an instance of codemeter service
   - `$ systemctl start codemeter.service`
3. Verify that an instance of codesyscontrol is running on your EtherCAT controller to connect the device to CODESYS Development System. Use one of the following commands to start an instance of codesyscontrol
   - `$ /opt/codesys/bin/codesyscontrol.bin /etc/CODESYSControl.cfg`
   - `$ systemctl start codesyscontrol.service`
   - `$ /etc/init.d/codesyscontrol start`
4. Connect the CODESYS Development System running on the Windows PC to the EtherCAT network
5. Open **"Tools→ License Manager"**
6. Select **"Device"**
7. Select **"Dongle"**
8. Select the name of the device that is acting as the EtherCAT controller and click **"Ok"**
9. A list of the license details shows up under the **"Licenses"** box on the left. This action can take a couple seconds to a couple minutes

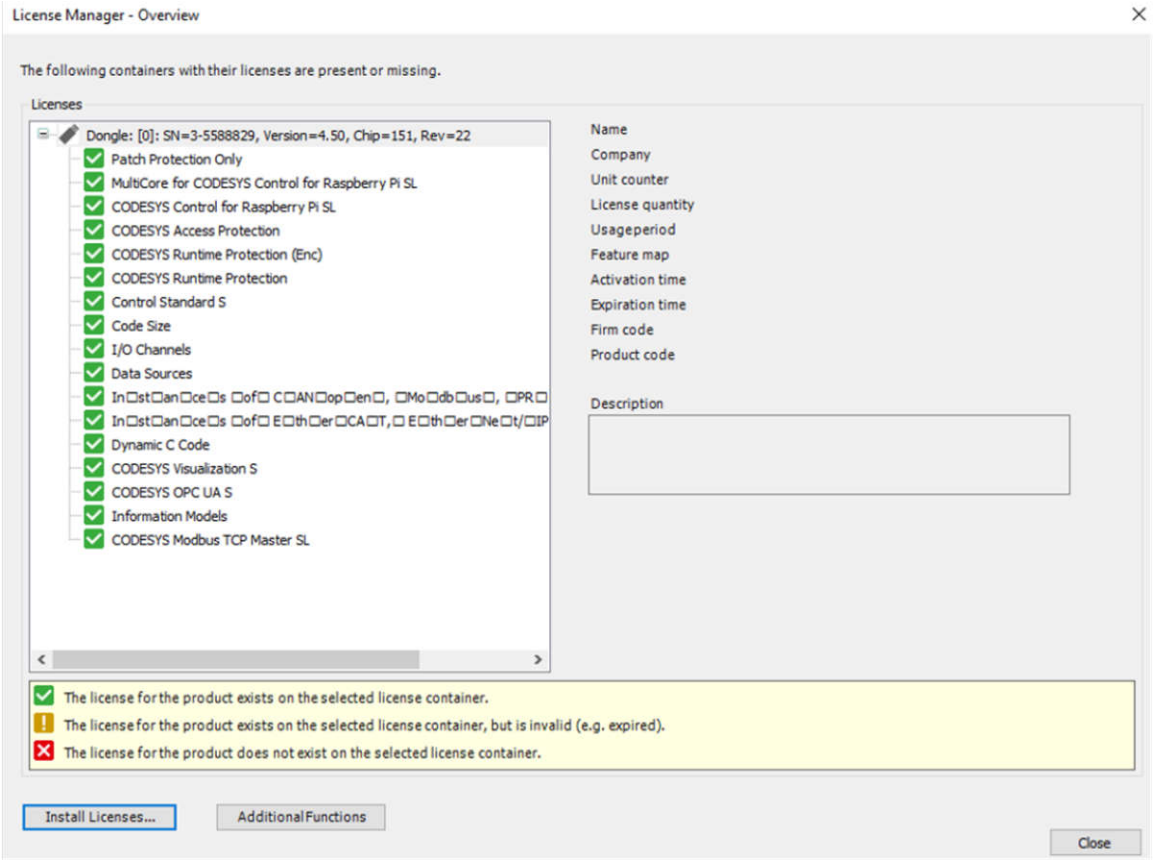


Figure 8-4. CODESYS Activated License Checkboxes

a. Another quick method to check if the license has been "applied" to the EtherCAT controller is to see if the green cycle symbol shows up next to the **"EtherCAT_Master (EtherCAT_Master)"** device in the left-hand panel

8.4.1 Known Issues With Verifying CODESYS License Applied

If the verifying the CODESYS License fails, the following are some tips for fixing or troubleshooting the problem

- For some EtherCAT controller devices running on Linux, verify that the HID and HIDRAW are enabled in the kernel configuration. This can involve modifying the existing kernel configuration and rebuilding the kernel.
  - See the following link for more details about this kernel configuration: Wibu, CODESYS Key, Dongle: License is not found on the USB dongle
- Check that EtherCAT controller device registers that the USB dongle has been connected by entering "lsusb" in the command line. A message similar to the following appears

```
root@am62xx-evm:~# lsusb
Bus 001 Device 002: ID 064f:2af9 WIBU-Systems AG CmStick (HID, article no. 1001-xx-xxx)
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

- Run the following command in the EtherCAT controller device Linux terminal to capture a CodeMeter log. This log shows details on if codemeter is able to read the USB dongle/CmContainers

```
cmu --cmdust --file CmDust.log
```

- Run the following command to check if there is a container matching the serial number of your USB dongle (3-XXXXXXX)

```
cmu --list --list-content
```

Some other potentially helpful information can be found in this CODESYS thread. Note that being signed into a CODESYS online account is required to view the thread.

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