White Paper

Industrial

Intel® Time Coordinated Computing (Intel® TCC) for Industrial Applications

Improving Real-Time Performance of CODESYS Control Applications with Intel's Real-Time Technologies

Table of Contents

Executive Summary 1

Intel's Real-Time Processing Solutions..... 2

Utilizing Intel TCC with CODESYS Sample Application .. 2

- Overview..... 2
- Sample Application 2
- Setup..... 2
- System Overview 2
- Target System..... 3
- Hardware Components..... 4
- Software Configurations 4
- Test Configurations..... 5
- Configuration 1: Intel® TCC Mode Disabled 5
- Configuration 2: Intel® TCC Mode Enabled..... 5
- Configuration 3: Intel® TCC Mode Enabled, CAT Configured..... 6
- Experiment Execution 6
- Launch the Application 6
- Stress Workload..... 6
- Monitor CODESYS Tasks Cycle Time and Jitter 6
- Experiment Results 7
- Results Summary 7

Reference Documents 8

Appendix A: Cache Agent Labels.. 8

Appendix B: Stress Workload Parameters 8

Executive Summary

The need for high-performance, real-time processing is rapidly emerging for high-precision industrial use cases such as autonomous mobile robots, robotic arms, and other autonomous infrastructure. These applications demand processing in **hard real-time**, which is required for use cases where the capability will fail if it cannot be executed within the allotted time. For this paper, which is focused on Intel® real-time solutions that exclusively address hard real-time use cases, the term real-time will refer specifically to hard real-time.

Figure 1. Real-time systems taxonomy: Intel real-time solutions are focused on hard real-time use cases

Performance in real-time applications is highly dependent on system-level behavior: every element must perform reliably and predictably within a specific time window to avoid failure. Meanwhile, inefficiencies in every software layer add up to worst-case execution time. Furthermore, resource contention introduces interference that can result in jitter, which also impacts performance.

The paper is organized as follows:

- Section 1 summarizes what a real-time system is and defines the term "hard real-time."
- Section 2 introduces key terms to understand the scope of Intel's Real-Time Processing Solutions.
- Section 3 provides experiments with a CODESYS "Robotics Pick and Place" demo application. From system setup to testing, this section compares the real-time performance differences of CODESYS programs with different configurations.
- Section 4 provides the experiment's conclusion.

Intel's Real-Time Processing Solutions

Intel processors are multipurpose and can serve a wide range of use cases, including data analysis in the cloud, gaming PCs and traditional office laptops, and edge devices.

Intel® Time Coordinated Computing (Intel® TCC) is a set of features that augment the compute performance of Intel processors to address the stringent temporal requirements of real-time applications. Intel TCC reduces jitter and improves performance for latency-sensitive applications. It helps to maximize efficiency by aggregating time-critical and non-time-constrained applications onto a single board.

One of the key technologies under the Intel TCC umbrella is Cache Allocation Technology (CAT). CAT addresses shared resource concerns by providing software control of where data is allocated into the last-level cache (LLC), enabling isolation of critical applications.

Intel TCC features are built into the processor, and their full potential is unlocked when the entire solution stack is optimized from top to bottom. Intel offers a reference real-time software stack that abstracts these hardware features to accelerate hardware configuration ("tuning") and application development.

Utilizing Intel TCC with CODESYS Sample Application

Overview

The following experiment compares the differences in two key performance indicators, jitter and cycle time, when using a CODESYS program with different configurations.

Sample Application

The CODESYS use case, "Robotics Pick and Place," is chosen as our sample application. In this use case, the application directs a tripod robot to pick up rings (shown as circular parts in Figure 2) from a rotating turntable and place them on cones (shown as circles with bigger diameters in Figure 2) moving on a conveyor belt.

The tripod system consists of three linear drives on rails with traversing sliders (Figure 3). The guide rails are fixed parallel to the vertical axis. The pool plate is connected to all three traversing sliders with three pairs of connecting rods of equal length. The paired design of connecting rods holds the tool plate in the same orientation parallel to the XY plane. This kinematic design allows the tool plate to move within three dimensions.

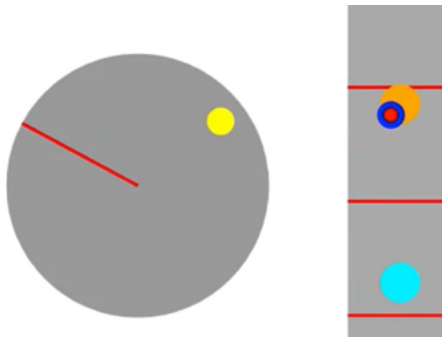

Figure 2. Robotics Pick and Place

Figure 3. Tripod kinematics with linear axes

Setup

System Overview

The experiment involves three different computers.

The first computer is a CODESYS development system installed with Windows and CODESYS IDE.

The second computer representing the target system is an industrial PC installed with Debian 10 with PREEMPT_RT patch and CODESYS Control for Linux. It is the controller to run the CODESYS workloads for the robot motion control. Three EtherCAT servo drives are connected to the target system to simulate the tripod system.

The third computer is an Intel TCC Tools host installed with Ubuntu 20.04. This host does the CAT configuration on the target system. Note: there are other ways to configure CAT.

The three computers were connected to the same network.

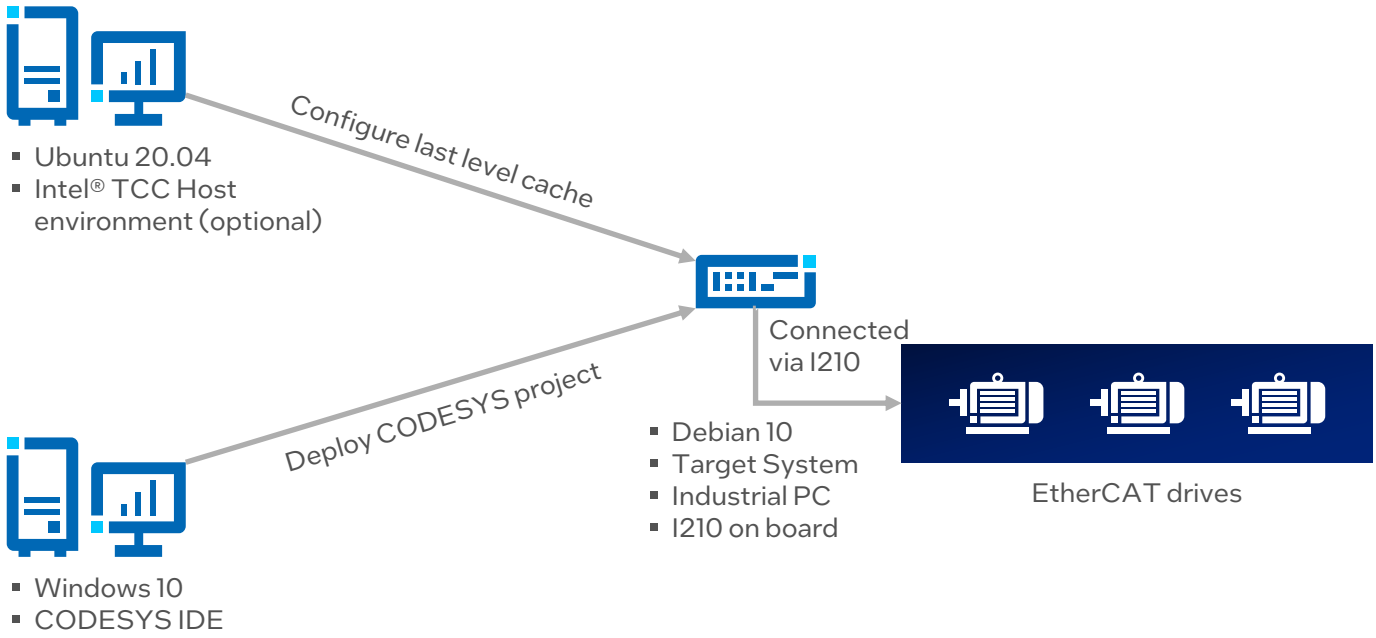

Figure 4. System Overview

Target System

The target system is an industrial PC equipped with Intel® Edge Controls for industrial software. Refer to the diagram below for the architecture of the system.

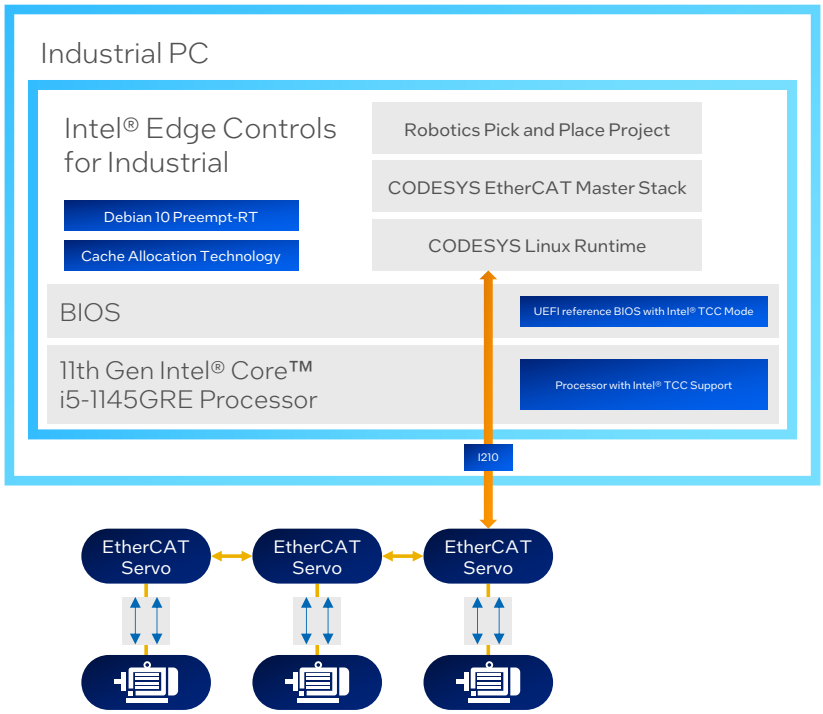

Figure 5. Target System Architecture Overview

Hardware Components

The hardware configurations of the target system are described in table 1 below.

Hardware	Details
Processor	11th Gen Intel® Core™ i5-1145GRE @ 2.60GHz. Products formerly known as Tiger Lake
Number of Cores	4
TDP	28 W
RAM	8 GB
Network Interface Controller (NIC)	UT-CLR-PCIE 4x Intel® I210 Ethernet Controller

Table 1. Hardware Components of the Target System

Refer to below pictures for the industrial PC and servo drives used in the experiment.

Figure 6. Industrial PC

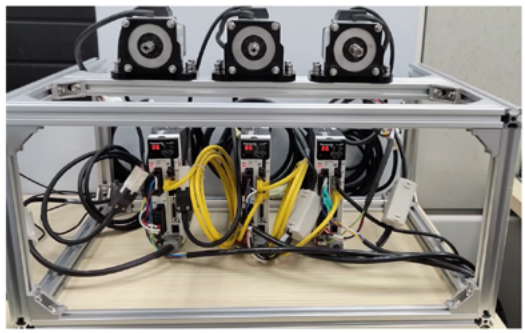

Figure 7. Servo Drives for Tripod Robot

Software Components

The software components of the target are described in table 2 below.

Software	Details
Operation System	Debian 10 buster[1] PREEMPT_RT
Linux Kernel	v5.4.115-rt57
Intel® Edge Controls for Industrial	Release 2.5
Intel TCC Tools	2021.2
stress-ng[2]	0.09.50
CODESYS Development System V3	3.5.17.10 64 bit
CODESYS Control for Linux SL[3]	4.1.0.0

[1] The kernel and PREEMPT_RT source code are from Intel® Edge Controls for Industrial (Intel® ECI) project.
[2] The stress workloads core affinity is set to core 2.
[3] The CODESYS main task core affinity is set to core 1.

Table 2. Software Configurations on the Target System

To further optimize the system for each configuration:

1. Stop services that are not relevant for the benchmark to reduce the interference with the real-time task.
2. Disable timer migration on PREEMPT_RT Linux.

Test Configurations

Three test configurations were used in the experiment.

Table 3 compares the configurations:

	Configuration 1	Configuration 2	Configuration 3
Intel TCC Mode	✗	✓	✓
Cache Allocation Technology for CPU	✗	✗[4]	✓[5]

[4] When Intel TCC Mode is enabled in the BIOS, it will enable the GT CLOS and WRC Feature simultaneously and lock them from changing. The GT CLOS limits the number of cache ways the GPU can allocate into in the L3 cache. The WRC Feature turns on Intel® Data Direct I/O Technology (Intel® DDIO) which enables the I/O devices to leverage Last Level Cache (LLC) as the intermediate buffer. No per-CPU last level cache partition applied.

[5] Based on Configuration 2, the per-CPU last level cache partition is further added.

Table 3. Test Configurations

Configuration 1: Intel TCC Mode Disabled

Intel TCC mode is disabled in the BIOS. No CAT configuration is applied to this configuration.

BIOS	Details
Version	TL37I103
Intel TCC Mode	Disabled
GPU Frequency	300 Mhz
System Agent Frequency	SA GV Fixed High

Table 4. Configuration 1 BIOS Configuration

The L3 cache of the 11th Gen Intel Core i5-1145GRE processor is divided into eight equal cache ways. When there is no CAT configuration, all cache ways are shared by all CPU cores and GPU. The cache partition of this configuration is as below. When Intel TCC mode is disabled in the BIOS, it will enable Hyperthreading by default, so the system will display 8 cores.

Cache Way Capacity Mask	7	6	5	4	3	2	1	0
Caching Agent	C[0-7], G							

Table 5. L3 Cache Partition

Caching agent label details can be found in Appendix A.

Configuration 2: Intel TCC Mode Enabled

Intel TCC mode is enabled in the BIOS.

BIOS	Details
Version	TL37I103
Intel TCC Mode	Disabled
GPU Frequency	300 Mhz
System Agent Frequency	SA GV Fixed High

Table 6. Configuration 2 BIOS Configuration

The cache partition of this configuration is described in table 7 below.

No explicit cache partition is in this configuration. The default cache partition is as below when Intel TCC mode is enabled in the BIOS. When Intel TCC mode enabled in the BIOS, it will disable Hyperthreading by default so the system will display 4 cores.

Cache Way Capacity Mask	7	6	5	4	3	2	1	0
Caching Agent	C[0-3]	C[0-3]	C[0-3]	C[0-3]	C[0-3]	C[0-3]	C[0-3], G	C[0-3], I

Table 7. Configuration 2 L3 Cache Partition

Configuration 3: Intel TCC Mode Enabled, CAT Configured

Intel TCC mode is enabled in the BIOS. Cache is configured for the system.

BIOS	Details
Version	TL37I103
Intel TCC Mode	Enabled
GPU Frequency	300 Mhz
System Agent Frequency	SA GV Fixed High

Table 8. Configuration 3 BIOS Configuration

The cache partition of this configuration is described in Table 9 below.

Cache ways 7,6 are shared among workloads running on the CPU 0, Cache ways 5,4 are shared among workloads running on the CPU 1, Cache ways 3,2 are shared among workloads running on the CPU 2, Cache ways 1 are shared among workloads running on the CPU 3 and for GPU. Cache ways 0 are shared among workloads running on the CPU 3 and for I/O.

Cache Way Capacity Mask	7	6	5	4	3	2	1	0
Caching Agent	C0	C0	C1	C1	C2	C2	C3, G	C3, I

Table 9. Configuration 3 L3 Cache Partition

Experiment Execution

Launch the Application

Download and launch the sample application to the target system.

Stress Workload

The stress workloads below were added to the target system to simulate a noisy neighbor.

```
$ taskset -c 2 stress-ng --cpu 8 --cpu-load 100 --cache 4 --cache-flush
--memthrash 4 --stream 10 --vm 4 --vm-bytes 128M --fork 4
```

Stress workload parameters details can be found in Appendix B.

Monitor CODESYS Tasks Cycle Time and Jitter

In the CODESYS task configuration, the "Monitor" tab shows the real time indicators of the running tasks.

- The key performance indicators are cycle time and jitter.
- The Max. Cycle Time is the maximum measured cycle execution time over all cycles.
- The Average Cycle Time is the average measured cycle execution time over all cycles.
- The Max. Jitter is the maximum measured periodic jitter.
- The Min. Jitter is the minimum measured periodic jitter.

Periodic jitter (J_{per}) is the deviation of the task cycle period (T_{per}) from the configured task cycle period (T0). The configured (ideal) cycle period T_0 is 4ms in this experiment.

$$J_{per} = T_{per} - T_0$$

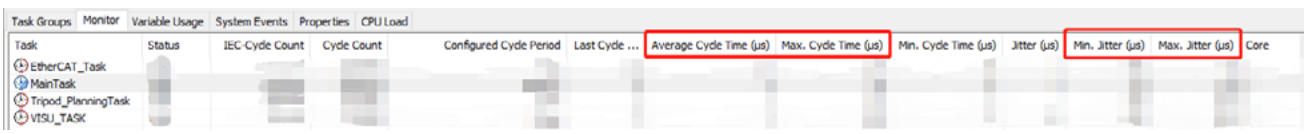

Figure 8. CODESYS Task Configuration — Monitor Panel

Experiment Results

To obtain meaningful results, a test duration of 48 hours was chosen for each test run.

	Maximum Jitter		Minimum Jitter		Maximum Cycle Time		Average Cycle Time	
Test Configurations	(µs)	vs. Config 1	(µs)	vs. Config 1	(µs)	vs. Config 1	(µs)	vs. Config 1
Configuration 1	163	100%	-185	100%	1279	100%	128	100%
Configuration 2	60	37%	-66	36%	507	40%	110	86%
Configuration 3	22	13%	-23	12%	335	26%	57	45%

Table 10. Cycle Time and Jitter for 48 hours test

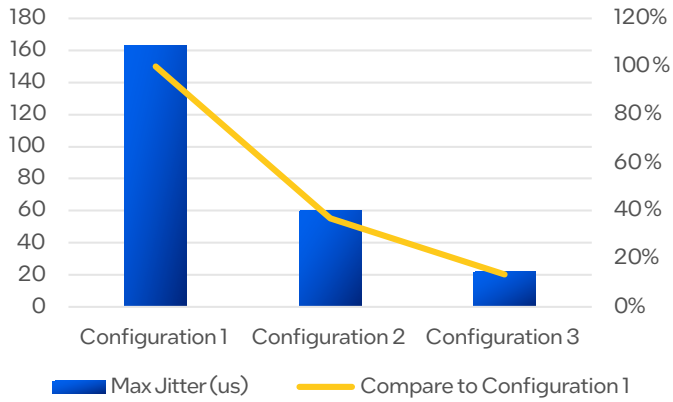

Figure 9. Maximum Jitter Comparison

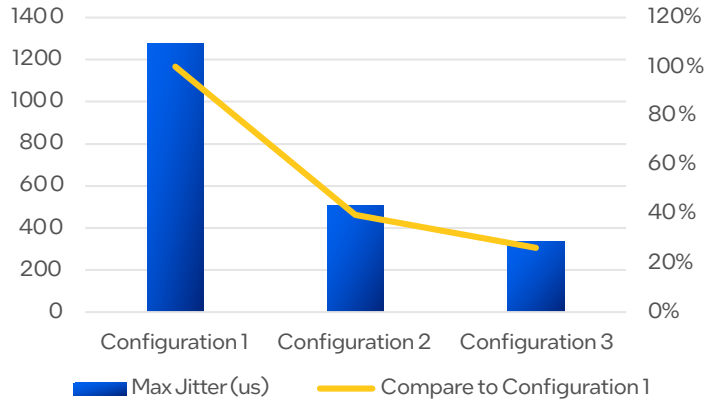

Figure 10. Maximum Cycle Time Comparison

Results Summary

Configuration 1 is not enabled with Intel TCC Mode and CAT. Configuration 2, based on Configuration 1, is enabled with Intel TCC Mode in the BIOS. When Intel TCC Mode is enabled in the BIOS, it will enable the GT CLOS and WRC Feature simultaneously and lock them from changing. Configuration 3 is based on Configuration 2, with per core last level cache partition configured.

Each test runs for 48 hours with a stress workload as a noisy neighbor.

Comparing the results, the max jitter in Configuration 2 is 37% of Configuration 1. The max cycle time in Configuration 2 is 40% of Configuration 1. It shows that enabling Intel TCC mode in BIOS can effectively improve the system's determinism without any higher-level software change.

When CAT is enabled in Configuration 3, the max jitter is 22 µs, which is 13% of Configuration 1, and the max cycle time is 26% of Configuration 1. CAT provided software-programmable control over the amount of cache space the real-time task could consume. This allowed the OS to protect the real-time tasks in a noisy environment.

Conclusion

In this experiment, we demonstrated that 11th Gen Intel Core i5-1145GRE processors@2.60Hz running the described configurations in this paper are capable of hard real-time applications. In using real-time capable processors developers no longer have to sacrifice high performance for time determinism. The results in Table 10 show that Configuration 3 yields better maximum/minimum jitter while improving cycle time performance, which are essential real-time determinism challenges for developers. Developers can leverage a powerful and ruggedized PC-based hardware system for software-defined industrial controller together with other critical applications like security, gateway, AI/ML, and orchestration in the same edge system as part of digital transformation or OT/IT convergence.

The experiment also demonstrated that the same Intel processor could produce three levels of latency and jitter improvement without changing the application software. This is important because the latency and jitter improvements at the hardware processor level benefit all of the software stack running, such as RTOS and soft PLC, allowing developers to preserve the investment in software, hence reducing development cost and speeding time to market.

The ongoing industry transformation drives the demand for converged solutions capable of satisfying real-time requirements while generally staying power efficient and leaving sufficient performance for other concurrent tasks. Intel TCC delivers performance improvements for latency-sensitive applications when running alongside non-time-sensitive applications on the same system.

Reference Documents

The following table shows documents, with document numbers, available on Intel Resource and Design Center. Log into the Resource and Documentation Center to search for and download the document numbers. Contact your Intel field representative for access.

Note: Third-party links are provided as a reference only. Intel does not control or audit third-party benchmark data or the websites referenced in this document. Visit the referenced website and confirm whether referenced data are accurate.

Documents	Document No./Location
Improving Real-Time Performance of CODESYS Control Applications with Intel's Real-Time Technologies	757527
Tiger Lake for IoT Platforms BIOS Writer's guide Addendum - 1.1 section 3.2	616309
Intel® TCC Tools 2022.1 Developer Guide	Link
11th Gen Intel® Core™ Processors Real-Time Tuning Guide	621732
Intel® 64 and IA-32 Architectures Software Developer's Manual Combined Volumes: 1, 2A, 2B, 2C, 2D, 3A, 3B, 3C, 3D, and 4 - RDC #671200	671200
Intel® Edge Controls for Industrial	Link
CODESYS Online Help	Link
CODESYS STORE	Link

Table 11. Reference Documents

Appendix A: Cache Agent Labels

This appendix details the Intel TCC Cache Allocation Technology configurations labels.

Intel® TCC Cache Allocation Technology Configurations Labels	Detail
G	Cache ways assigned to GPU use
I	Cache ways assigned to I/O use
C	Cache ways assigned to CPU use

Table 12. Intel® TCC Cache Allocation Technology Configurations Labels

Appendix B: Stress Workload Parameters

This appendix details the stress workload parameters.

Arguments	Description
--cpu N	start N workers spinning on sqrt(rand()).
--cpu-load P	load CPU by P %%, 0=sleep, 100=full load (see -c).
--cache N	start N CPU cache thrashing workers.
--cache-flush	flush cache after every memory write (x86 only).
--memthrash N	start N workers thrashing a 16MB memory buffer.
--stream N	start N workers exercising memory bandwidth.
--vm N	start N workers spinning on anonymous mmap.
--vm-bytes N	allocate N bytes per vm worker (default 256MB).
--fork N	start N workers spinning on fork() and exit().

Table 14. Stress-ng parameters