11th Gen Intel® Core™ Processors

Real-Time Tuning Guide

March 2022

Document Number: 640980-1.4

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting: http://www.intel.com/design/literature.html

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Your costs and results may vary.

Check with your system manufacturer or retailer or learn more at intel.com.

Contents

Figures

Tables

Revision History

Date	Revision	Description
March 2022	1.4	Updated content for Ethernet controllers, BIOS settings and KPI parameters
October 2021	1.3	Updated content for RC6 and Slim Bootloader
July 2021	1.25	Updated content for Graphics Power Management and description of OPIO recentering
June 2021	1.2	Updated for TCC Tools, Slim Bootloader, and ethernet controllers
March 2021	1.1	Updated content for RC6 Linux disablement and variant configurations
September 2020	1.0	Updated per launch of 11th Gen Intel® Core™ processors
August 2020	0.7	Initial release

§

1.0 Introduction

Intel processors are multi-purpose and can serve a wide range of use cases from data analysis in the cloud, to gaming PCs and traditional office laptops, to devices at the edge. Starting with 11th Gen Intel® Core™ processors (code name: Tiger Lake UP3), Intel is introducing a new set of features called Intel® Time Coordinated Computing (Intel® TCC) to augment the compute performance of its processors with ability to address the stringent temporal requirements of real-time applications. A key value is improved temporal performance for latency-sensitive applications when they are running alongside non-time-constrained applications on the same system.

Real-time applications (also called "workloads") must execute within a certain amount of time, consistently, across numerous iterations. While specific requirements vary by use case, real-time applications typically perform the following sequence of tasks:

1. Process new input such as a sensor measurement or camera video streams.
2. Perform a computation such as a new motion vector or object detection bounding box.
3. Control an actuator or render a result.

Furthermore, these applications have a deadline. They must complete the tasks within a certain time which may range in typical applications from microseconds to milliseconds.

For 11th Gen Intel® Core™ processors, Intel provides the following support for real-time applications:

- Selected processors with features to minimize worst-case execution time (WCET) within the system, referred to as Intel® TCC features.
- Ethernet controllers that support IEEE 802.1 Time-Sensitive Networking (TSN) standards.
- Best known configurations (BKC) with real-time optimizations. The BKC is the foundation of Intel key performance indicators (KPIs) of time performance.
- Software tools to accelerate hardware configuration/tuning and application development by enabling performance analysis and access to hardware features. Some of the tools are included in the BKC, for example, Intel® TCC Tools and others can be accessed separately.

1.1 About This Document

This document describes tuning for real-time applications. Tuning covers every part of the system, including hardware, BIOS, operating system, network, and the application itself. This document is for anyone working on these components, such as system engineers, firmware developers, operating system developers, and application developers.

Note: This document has an NDA version. The NDA version includes details on architecture for Data Direct Input/Output (DDIO), register values for ethernet virtual channels, and information on L3 class of service aliasing. Contact your Intel representative for access.

Consider this document the starting point for understanding real-time tuning. This document:

- Introduces real-time hardware features and related software.
- Introduces the KPIs that Intel uses to measure real-time performance.
- Describes the tuning strategy, including details about use of real-time features and analysis/profiling tools.

Note: This document is not intended to provide a comprehensive specification for real-time features or software implementations. Where feasible, reference documentation has been cited that gives further technical detail for the topics discussed.

§

2.0 Capabilities Overview

2.1 Hardware Features

The IoT technology segment increasingly requires time-related capabilities both within embedded devices and across networks to support smart or automated cyber-physical systems (CPS).

Supported 11th Gen Intel® Core™ processors provide Intel® Time Coordinated Computing (Intel® TCC) features to optimize real-time compute performance. Supported Ethernet controllers provide IEEE Time-Sensitive Networking (TSN) features to optimize network traffic.

In this document, the hardware features are described in relation to real-time tuning. These features include Cache Allocation Technology (CAT), graphics technology class of service (GT COS), data direct I/O / write cache (WRC), upstream virtual channels, and more. For information about hardware features beyond the scope of this document, see the *EDS Addendum* listed in Reference Documents.

2.2 Supported Hardware

2.2.1 11th Gen Intel® Core™ Processors

Intel recommends the following processors for real-time applications/solutions:

Table 1. 11th Gen Intel® Core™ Processors

	i7-1185GRE (11th Gen Intel® Core™ i7 Processor)	i5-1145GRE (11th Gen Intel® Core™ i5 Processor)	i3-1115GRE (11th Gen Intel® Core™ i3 Processor)
Use Condition	Industrial	Industrial	Industrial
Cores	4C/8T	4C/8T	2C/4T
TDP	12/15/28W	12/15/28W	12/15/28W
Base Frequency	1.2/1.8/2.8GHz	1.1/1.5/2.6GHz	1.7/2.2/3.0GHz
Turbo Max Frequency	N/A	N/A	N/A
Xe Graphics (Gen 12)	96EUs	80EUs	48EUs
GFX Max Frequency	1.35GHz	1.3GHz	1.25GHz
T_j	-40 to +100C	-40 to +100C	-40 to +100C
Integrated Heat Spreader (IHS) (i.e. Lid)	No	No	No
In-Band ECC	Yes	Yes	Yes

	i7-1185GRE (11th Gen Intel® Core™ i7 Processor)	i5-1145GRE (11th Gen Intel® Core™ i5 Processor)	i3-1115GRE (11th Gen Intel® Core™ i3 Processor)
Intel® Time Coordinated Computing	Yes	Yes	Yes
FuSa	No (Intel® Functional Safety Essential Design Package (Intel® FSEDP) only)	No (Intel® Functional Safety Essential Design Package (Intel® FSEDP) only)	No (Intel® Functional Safety Essential Design Package (Intel® FSEDP) only)

NOTES:

- Refer to EDS documentation for details on the GFX High Frequency Mode for Industrial SKUs.
- All TDPs assume power management is turned on.

2.2.2 Customer Reference Boards (CRBs)

The 11th Gen Intel® Core™ processors DDR4 CRB is available for evaluation and development.

Note: Contact an Intel representative to obtain the CRBs

2.2.3 Ethernet Controllers

Supported 11th Gen Intel® Core™ processors has one integrated TSN-capable 2.5 Gb Ethernet controller. There is also a second non-TSN 1 Gb Ethernet controller. Additional TSN Ethernet controllers may be added by using Intel® Ethernet Controller I225 Series connected via PCIe. A list of supported TSN standards is available in the *11th Gen Intel® Core™ Processors for IoT Platforms EDS Addendum* listed in Reference Documents.

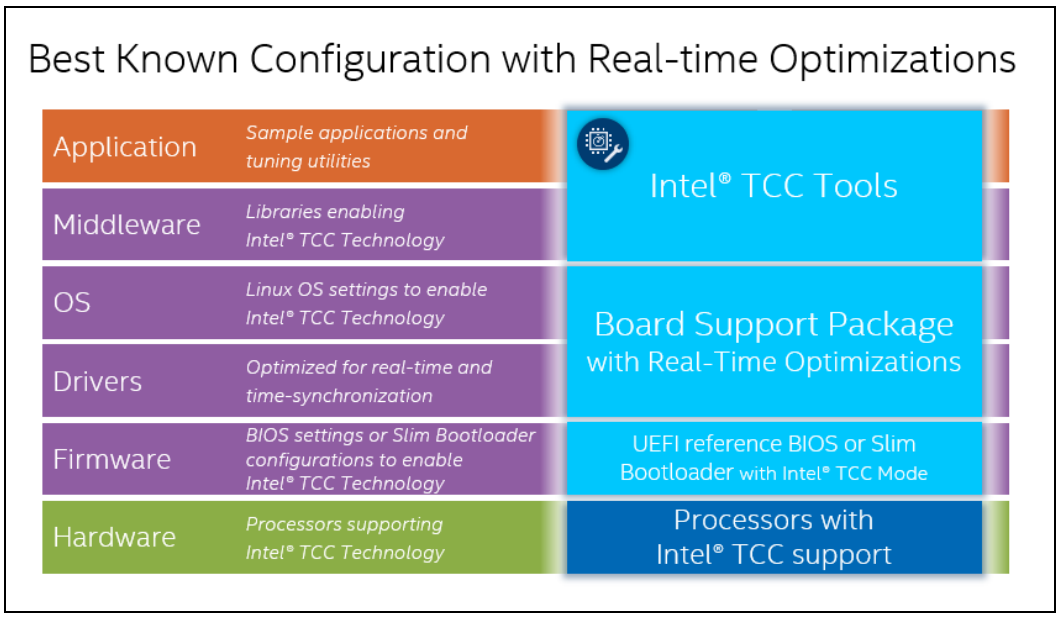
2.2.4 Where to Get Hardware

Contact your Intel representative to acquire hardware.

2.3 Intel® TCC Software

Intel provides software that enables developers to access Intel® TCC hardware features, including BIOS support, Linux* kernel drivers and patches, libraries, application programming interface (APIs), sample applications, and tools.

Figure 1. Intel® TCC Reference Software Stack



2.3.1 Application and Middleware Support

Intel® TCC Tools provide application and middleware support, including APIs, tools, and sample applications that enable developers to access certain Intel® TCC features (described later in this document). Intel® TCC Tools also enable developers to analyze the behavior of real-time applications.

For more information about Intel® TCC Tools beyond the scope of this document, see the Intel® TCC Tools product page.

2.3.2 Operating System Support

Intel provides Linux* OS support in the form of the Yocto Project*-based board support package (BSP). The BSP contains real-time configuration settings and the Linux PREEMPT_RT patch. It is pulled in from the upstream branch by the Yocto Project build to the BSP. The BSP is part of the BKC. For information about the BKC, see Best Known Configuration (BKC).

Support for other operating systems is provided by their respective vendors.

This document describes OS tuning in more detail, as part of the Intel® TCC Platform Tuning Strategy.

2.3.3 Driver Support

Intel provides Linux* OS driver support with its Yocto Project*-based board support package (BSP). The BSP-provided drivers implement support for native Linux APIs, enabling software coordination of time between locally connected (i.e. PCIe) and integrated devices, using dedicated time synchronization hardware. Devices with support for hardware-based time coordination capabilities are:

- PTM-capable PCIe devices
- Time-Aware GPIO
- TSN-enabled Ethernet interfaces

The BSP-provided drivers use virtual channels to prioritize real-time data flows within the platform to the Ethernet interface where available.

2.3.4 BIOS Support

Many of the control registers involved in tuning are accessible only through the BIOS. Some companies may not want to enable real-time tuning as it fundamentally changes the performance profile of the platform. This necessitates configurability, which can be modified without recompiling and reflashing the BIOS.

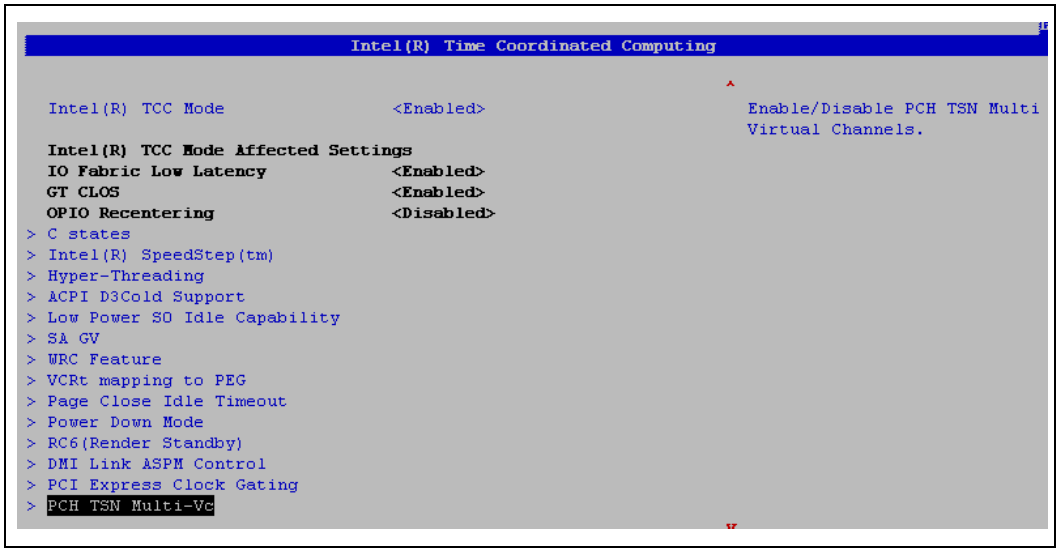
Intel provides a reference BIOS that enables companies to control tuning through an option called Intel® Time Coordinated Computing Mode (Intel® TCC Mode). This option is in the following BIOS menu:

Intel Advanced Menu > Intel® Time Coordinated Computing

The following figure is an example of the Intel® TCC Mode interface in the reference BIOS. The interface may differ depending on a given BIOS version.

Note: This guide describes the settings that are included in the reference BIOS. Your BIOS vendor may provide different settings. Contact your BIOS vendor for more information.

Figure 2. Example of Intel® TCC Mode Interface



Note: BIOS menu options may vary slightly from displayed figure.

Intel® TCC Mode provides access to a collection of settings. Some of these settings appear only in the Intel® Time Coordinated Computing submenu, while others are shortcuts to settings located elsewhere in the BIOS.

When Intel® TCC Mode is enabled, it configures the settings to predetermined values and makes those settings “read only” so they cannot be changed independently. This behavior provides a consistent configuration that is optimized for general real-time performance.

When Intel® TCC Mode is disabled, the settings can be set independently for granular control of specific capabilities. When one of these settings is selected from the Intel® Time Coordinated Computing submenu, the BIOS jumps to the BIOS page where each setting resides.

Intel® TCC Mode affected settings fall into the following categories:

- Disabling certain power management capabilities: Hardware and software power management features can negatively affect real-time performance for various I/O paths. Intel® TCC Mode disables power management settings in the BIOS.
- Enabling certain Intel® TCC hardware capabilities.

This document describes these settings in more detail, as part of the Intel® TCC Platform Tuning Strategy.

2.3.4.1 Additional Information

The reference BIOS is part of the BKC. For information about the BKC, see Best Known Configuration (BKC).

For register specifications, see the *BIOS Specification* (formerly BIOS Writer's Guide) listed in Reference Documents.

2.3.5 Slim Bootloader Support

Similar to the BIOS, Slim Bootloader (SBL) provides a configuration option to enable/disable Intel® TCC Mode, located in `BoardConfig.py`.

Check the build configuration file to confirm `self.ENABLE_TCC` is set to **1** to enable TCC subregion.

The other Intel® TCC related configuration data might also be enabled when `self.ENABLE_TCC` is set. Check the platform configuration BoardConfig.py for details. Then, rebuild SBL.

For additional information on TCC capsule generation, refer to the SBL wiki: https://slimbootloader.github.io/how-tos/enable-intel-tcc.html

2.4 Other Analysis and Profiling Tools

In addition to Intel® TCC Tools, Intel provides the following tools to help developers detect and analyze areas of the system that are negatively affecting real-time performance.

Table 2. Other Analysis and Profiling Tools

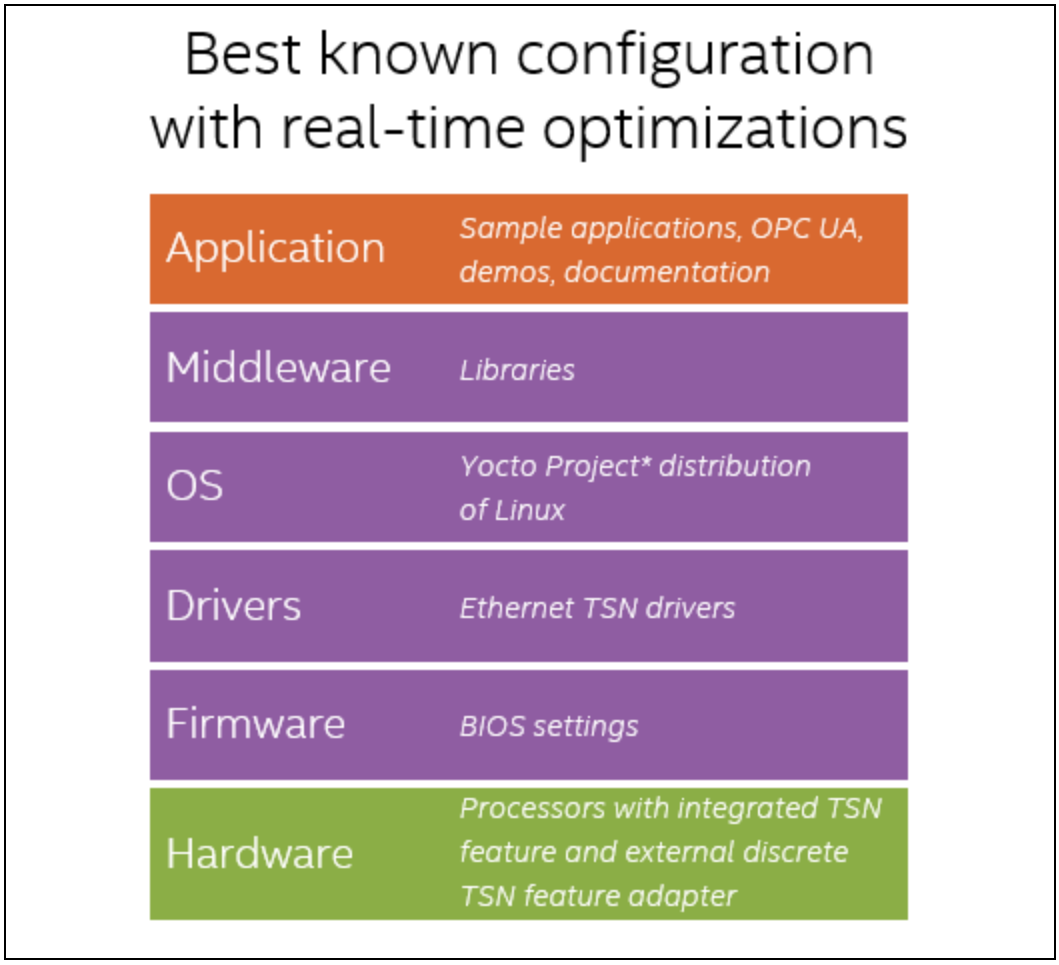
Tool	Description
Intel® VTune™ Profiler	Detect and analyze sources of system jitter. For information specific to real-time applications, see *Tutorial: Profile Applications with Intel® VTune™ Profiler* listed in Reference Documents.
Inter-event histogram triggers tool	A tool in the kernel that can assist with latency profiling and analysis. This tool is platform independent. See Section 2.2 of the kernel document *Event Histograms* in Reference Documents.

2.5 Time-Sensitive Networking Reference Software

Intel provides reference software that enables TSN features, including:

- Ethernet drivers (TSN reference software does not include ethernet drivers, it is just a sample app to demonstrate TSN capability).
- OS support
- Open-source libraries and utilities (open62541, iproute2, ethtool, linuxptp)
- Sample applications (socket-level and OPC UA Publish/Subscribe over TSN).

Figure 3. TSN Software



The Yocto Project*-based BSP listed in Best Known Configuration (BKC) provides all TSN Reference Software.

2.6 Best Known Configuration (BKC)

The BKC consists of the following components:

- Reference BIOS
- Drivers
- OS: Yocto Project*-based Board Support Package (BSP) for 11th Gen Intel® Core™ processors on IoT Platforms
- Intel® TCC Tools

Intel real-time key performance indicators (KPIs) are based on the combination of a supported processor, the BKC, and the provided Intel workload.

Note: The BKC is tuned for the KPI workload and may not be the best configuration for other workloads and additional tuning may be required.

You can run the Intel workload with the processor and BKC to replicate the KPIs in your lab and start to assess the effects of BKC components on performance.

The BKC configuration relies on the recommended BIOS with Intel® TCC Mode enabled and kernel configuration depicted in Section 2.3.4 and 6.1, respectively. The kernel configuration is reflected in the board support package, but the "Intel® TCC Mode" option in the BIOS must be set to *enabled* to replicate KPI performance.

For more information about the BKC, see section 2.0 of *Yocto Project*-based Board Support Package for 11th Gen Intel® Core™ Processors Release Notes* in Reference Documents.

More information on KPIs is detailed in the following Real-Time Key Performance Indicators (KPIs).

§

3.0 Real-Time Key Performance Indicators (KPIs)

Key performance indicators (KPIs) are performance metrics. Intel provides information about KPIs, including measurements and configurations for replicating the tests on a given platform in the Real-Time Gold Deck (refer to Table 9).

The Gold Deck describes:

- Real-time capabilities enabled at product launch of the industrial processors.
- Real-time KPI requirements for use of Intel BKC / board support package (BSP) and associated documentation.

The information covers the following two benchmarks:

- **Cyclic Test**: a Linux* tool for measuring OS jitter
- **Real-Time Compute Performance (RTCP):** a KPI developed by Intel that measures the total elapsed time from when a synthetic sensor data packet is received, to when a synthetic actuator command packet is sent (including the elapsed time for the execution of a predefined compute workload).

Note: The report is intended for Intel NDA customers only. Contact your representative for more details.

§

4.0 Intel® TCC Platform Tuning Strategy

Intel® Time Coordinated Computing (Intel® TCC) tuning is the primary responsibility of the Intel® TCC Mode BIOS option, Intel® TCC Tools, and OS configuration. Intel systems are meant for multiple use cases and changes in BIOS configurations are meant to tune for specific use cases. BIOS is usually configured for a default non-RT use case. Thus, an real-time BKC exists for real-time tuning to achieve peak performance based on a reference use case and customers may need further tuning to reach their desired performance for their specific use cases. This tuning is divided into the following categories:

- System software tuning
- Platform power management tuning
- Intel® TCC features tuning
- Fabric tuning

Different types of tuning have different effects on latency. *Latency* refers to the duration of time between two events. It can mean any type of latency: I/O latency, buffer access latency, and the sum of these and others— *network packet in* to *network packet out* (cycle time). Usages with more relaxed requirements need less tuning to meet their latency requirements, while usages with stricter requirements necessitate substantial effort in tuning. When tuning the platform, address higher-impact tuning first, then work down to lower-impact tuning until the KPI requirements are satisfied.

Figure 4. Intel® TCC Platform Tuning Strategy

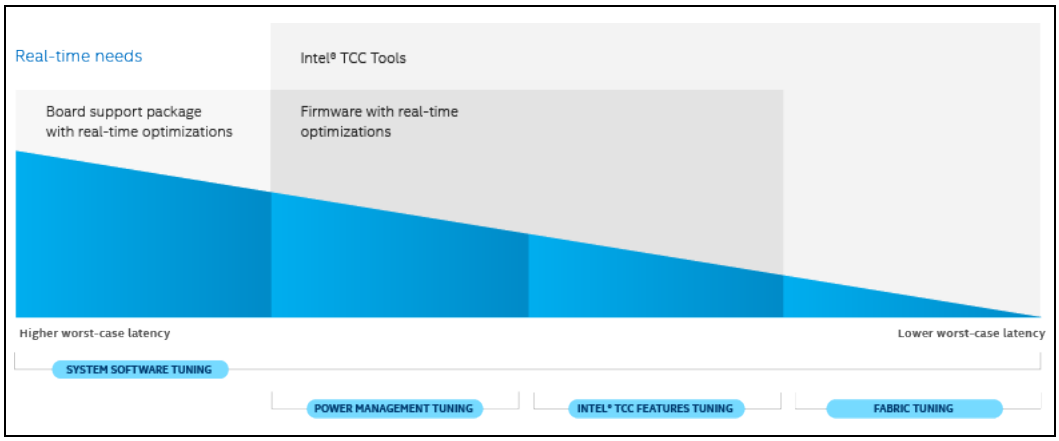


Figure 4 illustrates how different tuning categories help decrease the worst-case latency. Types of tuning on the left on the diagram have more absolute impact on performance, but still the same relative impact as types to the right.

For example, taking an out-of-the-box system and performing system software tuning may reduce latency on the order of milliseconds and decrease latency by a significant improvement. After performing system software tuning, if requirements still are not met, one can implement platform power management tuning. Power management tuning may reduce latency to the order of hundreds of microseconds, which is much less than system software tuning in absolute terms. Power Management tuning may also further reduce the latency jitter by hundreds of microseconds, complementing the software tuning that has been implemented.

Types of tuning that are further to the right on Figure 4 are more complex and incur higher penalties to concurrent workloads than those on the left. However, they may unlock the best possible real-time performance when enabled and used in combination with the other levels of tuning.

System Software Tuning: System software has the highest impact on real-time performance and can increase worst-case execution time (WCET) by many orders of magnitude. Tune system software such as OS and drivers before even attempting to begin optimizing hardware features.

The BKC recommends various system software tuning settings such as Linux* kernel build configuration settings (for example, "`CONFIG_PREEMPT_RT=y`") and boot parameters (for example, `isolcpus`).

Platform Power Management Tuning: After system software has been tuned, move on to tuning platform power management. Platform power management often negatively affects real-time performance by putting the platform and subcomponents into low-power states or by throttling performance (for example, via frequency scaling).

- The BKC recommends various platform power management tuning settings such as the Linux kernel boot parameters (for example, `intel_idle.max_cstate=0`).
- The data streams optimizer configures various platform power management tuning settings such as disabling Intel SpeedStep® when it determines it is necessary to improve the latency of data streams. For more information, see the Intel® TCC Tools product page.

Intel® TCC Features Tuning: After system software and power management have been tuned, the system is running at the optimal hardware performance using the basic platform capabilities. To further improve real-time performance, the platform provides a set of Intel® TCC features. Intel® TCC features tuning targets specific transaction flows, compute workloads, or corner cases.

Fabric Tuning: *Fabric* refers to the interconnect technology that carries on-chip communications between the different functional components of the processor. Fabric tuning includes microarchitecture-specific optimizations for the functional components in the fabric that include prioritization, arbitration, and flow control. These tunings come at a high cost in implementation time and performance tradeoffs, and they provide comparatively little improvement relative to the other types of tuning. However, the best possible real-time performance can be attained by tuning the fabric arbitration and prioritization characteristics (though only after the other three types of tuning have been completed).

The data streams optimizer uses various fabric tuning settings when generating tuning configurations to meet input requirements. For example, the data streams optimizer may configure arbiter priority weights within the fabric to prioritize one type of traffic over another.

§

5.0 System Software Tuning

System software has the highest impact on real-time performance and can improve worst-case execution time (WCET) by many orders of magnitude. This can take the form of various sources of jitter including, but not limited to, scheduler preemption, Read-Copy-Update (RCU) callbacks, and poorly optimized drivers.

Intel recommends tuning system software such as OS and drivers before attempting to optimize hardware features. Accordingly, a good method for debugging massive latency spikes (higher than 1 millisecond) is to first ensure the OS (and higher-level software) has been sufficiently tuned. Scheduler-caused latency spikes also tend to exhibit periodicity.

Tuning system software for real-time often comes at the cost of average performance and multitasking, reducing the maximum aggregate throughput of the system.

This section describes the configurations and settings that may be implemented to tune a given platform to improve its real-time performance. It includes OS configurations and kernel parameters. It also includes what libraries are required, the command-line optimizations during kernel boot time, and OS behavior changes to improve platform performance.

5.1 Linux* Operating Systems

5.1.1 Kernel Parameters

This section explains the kernel parameters that you can configure to tune the performance of the platform. The parameters are based on kernel version 5.4 with the PREEMPT_RT patch and apply to all version up to version 5.10.

In the BKC, the kernel is compiled with the following parameters. You can find a description of each parameter in the specified kernel configuration file.

Table 3. Kernel Build Parameters

Kernel Build Parameter	Value	See Kernel Config File for Description
CONFIG_GENERIC_IRQ_MIGRATION	Y	kernel/irq/Kconfig
CONFIG_HIGH_RES_TIMERS	Y	kernel/time/Kconfig
CONFIG_CPU_ISOLATION	Y	init/Kconfig
CONFIG_RCU_NOCB_CPU	Y	kernel/rcu/Kconfig
CONFIG_SMP	Y	arch/x86/Kconfig
CONFIG_MIGRATION	Y	mm/Kconfig

Kernel Build Parameter	Value	See Kernel Config File for Description
CONFIG_PCIEPORTBUS	Y	drivers/pci/pcie/Kconfig
CONFIG_PCIE_PTM	Y	drivers/pci/pcie/Kconfig
CONFIG_GENERIC_IRQ_MIGRATION	Y	kernel/irq/Kconfig
CONFIG_EXPERT	Y	init/Kconfig
CONFIG_PREEMPT_RT	Y	kernel/Kconfig.preempt ***Note:*** This option is only used for preempt kernels and not required for tuning on standard kernels. Documentation can be found in the real-time kernel's source.
CONFIG_PREEMPT_RT_FULL	Y	kernel/Kconfig.preempt ***Note:*** This option is only used for preempt kernels and not required for tuning on standard kernels. Documentation can be found in the real-time kernel's source.
CONFIG_CPU_FREQ	N	drivers/cpufreq/Kconfig
CONFIG_SCHED_MC_PRIO	N	arch/x86/Kconfig
CONFIG_PREEMPT_RCU	Y	kernel/rcu/Kconfig
CONFIG_HUGETLBFS	Y	fs/Kconfig
CONFIG_EFI	Y	arch/x86/Kconfig

The kernel should be booted with the following parameters. For descriptions of these parameters, see Linux kernel command-line parameters in Reference Documents.

Table 4. Kernel Command-Line Parameters

Kernel Command-Line Parameter	Value
processor.max_cstate	0
processor_idle.max_cstate	0
intel_idle.max_cstate	0
clocksource	tsc
tsc	reliable
nmi_watchdog	0
nosoftlockup	No value required
intel_pstate	disable
idle	poll
noht	No value required. This can cause the timer tick to be enabled even with nohz_full set.

Kernel Command-Line Parameter	Value
nohz	No value required. ***Note:*** Although nohz is a common real-time parameter, Intel has found that nohz negatively affects the real-time performance of this processor. Intel recommends omitting nohz in the kernel boot parameters.
nohz_full	X-Y (where X is the first CPU and Y is the last CPU in a consecutive list of CPUs allocated to real-time applications)[1].
isolcpus	X-Y (where X is the first CPU and Y is the last CPU in a consecutive list of CPUs allocated to real-time applications).
rcu_nocbs	X-Y (where X is the first CPU and Y is the last CPU in a consecutive list of CPUs allocated to real-time applications).
irqaffinity	X-Y (where X is the first CPU and Y is the last CPU in a consecutive list of CPUs allocated to handling interrupts).
rcu_nocb_poll	No value required
hugepages	1024
cpufreq.off	1 ***Note***: Option may not be applicable to all kernel versions and may only benefit performance of some real-time workloads. On some kernel versions, it may be appropriate to handle after boot to ensure frequency is locked at max.
i915.enable_dc	0 ***Note:*** Option is only required when the platform includes integrated graphics.
i915.disable_power_well	0 ***Note:*** Option is only required when the platform includes integrated graphics.
mce	off
hpet	disable
numa_balancing	disable
efi	runtime
nowatchdog	No value required ***Note***: Option may only benefit performance of some real-time workloads. Recommended on a case-by-case basis.

[1] In kernel 4.18.x, when nohz_full is enabled, idle=poll can cause the tick to come back, but in 5.4 and beyond, this behavior has yet to be observed.

Kernel Command-Line Parameter	Value
iommu	pt ***Note***: Option may only benefit performance of some real-time workloads. Recommended on a case-by-case basis.
art	virtallow ***Note***: Option may only benefit performance of some real-time workloads. Recommended on a case-by-case basis.
rcupdate.rcu_cpu_stall_suppress	1 ***Note***: Option may only benefit performance of some real-time workloads. Recommended on a case-by-case basis.
i915.enable_rc6	0 ***Note***: Option may not be applicable to all kernel versions and may only benefit performance of some real-time workloads.

Variant configurations can be found in Appendix A.

5.1.2 Libraries

For information about libraries, see the Intel® TCC Tools product page.

5.1.3 Other Linux* OS Optimizations

This section provides additional optimizations that can be done in the OS environment to improve real-time performance. These optimizations are:

- changing the affinity of a certain process to align on a specific CPU
- modifying the behavior of the scheduler
- isolating certain cores for certain tasks

Modifying the scheduler to prioritize some tasks over others can help reduce downtime, which helps to achieve real-time performance. Linux* tools such as taskset and numactl enable the binding of a process or thread to core. Doing so can help localize memory, reducing memory accesses and leading to more desirable performance. Another tool that can be used to modify scheduling attributes is *CHRT*. This is a tool used to manipulate the real-time attributes of a process.

***Caution**: Enabling CHRT to manipulate the scheduler may cause conflicts with isolcpus settings.*

Numactl Example:

```
numactl  --cpunodebind=0 test.sh
```

This example binds all processes generated by test.sh to cores found only in CPU0.

By running this command, the processes generated by test.sh will have access to localized memory, resulting in better latency results than if processes were on separate CPU nodes.

CHRT Example:

```
chrt -f 99 <pid>
```

This example sets the scheduling policy of PID 99 to SCHED_FIFO. To find the PID, run `pidof -s` [process_name]

Taskset Example:

```
taskset -c 1,2,3 test.sh
```

This example assigns the processes generated to be bound to cores 1-3. By doing this, you are isolating test.sh from interference from processes running on core 0.

By running this command, the processes generated by test.sh will have access to localized memory, resulting in better latency results than if processes were on separate CPU nodes.

5.1.4 Linux* OS Resource Partitioning

The goal of this section is to provide insight into a methodology for partitioning resources on a platform using OS level tools.

The broad methodology relies on sectioning off cores for real-time workloads and leaving one core for best-effort work and system management. Typically, this is achieved with a combination of the command-line parameter *isolcpus* and the runtime parameter *taskset*. Isolcpus instructs Linux to not schedule anything on the instructed cores — they are considered isolated from the scheduler. Taskset then enables you to lock a program to a core. If the chosen core is one of the isolated cores, that program will be the only thing to run on that core. For example, in a four-core system, you could isolate cores 1 through 3 and leave core 0 for system management, then taskset a program to run on core 1.

Unfortunately, these tools do not restrict Linux from running certain kernel threads on the core, including interrupt handling. A couple methods to minimize the impact of these system management threads are moving interrupts and deprioritizing system threads.

To move interrupts, use the `/proc/irq` filesystem options in Linux. You can use the smp_affinity option for each registered interrupt to affine the interrupt to a specific core. For any interrupts that are not used by the real-time application, use the smp_affinity option to set them to the best-effort core. Make sure the interrupts that are important to the real-time application remain on the real-time core.

Further, using the PREEMPT_RT patch for Linux will expose system kernel threads as processes — using the function chrt enables you to set the priority and scheduling method of these processes. It is recommended to set both the system threads to a low priority and the critical workload priority much higher. Typically, a few processes are exposed such as ksoftirq (these can be seen by running ps –e). The core number assigned to a process appears at the end of the process name, for example, ksoftirq/2. Additionally, RCU processes are exposed as well. A few different types are available, but they also follow the same nomenclature of having a /<core> at the end of the process name. Using chrt, deprioritize these (priority 0), and set the scheduling method of these to SCHED_OTHER. SCHED_FIFO and a higher priority should be used for the critical workload.

Commands of Interest with Examples

These examples assume the best-effort core is 0, and this is a four-core system.

1. Command-Line Parameters:

```
isolcpus=nohz,domain,1-3 (offloads residual 1Hz scheduler tick)
rcu_nocbs=1-3 (offloads RCU callbacks)
nohz_full=1-3 (sets the scheduler clock to 1Hz if only one user-
space app is running)
i915.enable_dc=0
i915.disable_power_well=0
processor.max_cstate=0
intel_idle.max_cstate=0
intel_pstate=disable
```

2. OS Tools

```
taskset –c <core> <application>
chrt <scheduler method> –p <priority> <application PID>
       chrt -f –p 98 <application PID> = FIFO scheduling with
execution priority 98 (of 99)
       chrt -o –p 0 <application PID> = OTHER scheduling with
execution priority 0
/proc/irq/<irq number>/smp_affinity = core mask for affined cores
```

§

6.0 Platform Power Management Tuning

After System Software Tuning, real-time performance can be further improved by tuning platform power management. Platform power management affects real-time performance by putting the platform and subcomponents into low-power states or by throttling performance (for example, via frequency scaling). Entering and exiting these low-power states may incur significant latency penalties; similarly, frequency throttling or transitions can degrade performance as well.

High latency spikes caused by power management are characterized by their tendency to appear when the system is idle but disappear when the system is loaded with traffic. Related to power management, but rarely observed, is throttling caused by thermal events that often appear as a sudden drop in average (and worst) case performance after an extended runtime.

Platform power management tuning often involves disabling power management; thus, a tradeoff between real-time performance and TDP is required. BIOS largely handles power management tuning, but some power management is best or necessarily handled at the OS or driver level.

This section delves deeper into the power management (PM) aspect of the system that affects real-time performance. This includes the system PM, the graphics (GT) engine's PM, and the I/O fabric PM.

6.1 Intel® TCC Mode (BIOS): Power Management Settings

When enabled in BIOS, Intel® Time Coordinated Computing (Intel® TCC Mode) sets the following power management options.

Table 5. Intel® TCC Mode (BIOS): Power Management Settings

Setting Name	Option	Menu	Description
IO Fabric Low Latency	Enabled	Intel Advanced Menu\Intel® Time Coordinated Computing	When enabled, turns off some power management in the PCH I/O fabrics. This option provides the most aggressive I/O fabric performance setting. S3 state is not supported.
C states	Disabled	Intel Advanced Menu\Power & Performance\CPU-Power Management Control	When disabled, prohibits the core from entering low-power states.

Setting Name	Option	Menu	Description
Intel SpeedStep®	Disabled	Intel Advanced Menu\Power & Performance\CPU-Power Management Control	When disabled, turns off Intel SpeedStep® technology. The technology enables the management of processor power consumption via performance state (P-State) transitions.
ACPI D3Cold Support	Disabled	Intel Advanced Menu\ACPI D3Cold Settings	When disabled, prohibits some device power management states.
Low Power S0 Idle Capability	Disabled	Intel Advanced Menu\ACPI Settings	When disabled, prohibits S0ix states. S0ix states shut off parts of the SoC when they are not in use.
SA GV	Disabled	Intel Advanced Menu\Memory Configuration	When disabled, turns off SA GV. SA GV dynamically scales the work point (V/F), by applying DVFS (Dynamic Voltage Frequency Scaling) based on memory bandwidth utilization and/or the latency requirement of the various workloads for better energy efficiency at the System Agent.
Page Close Idle Timeout	Disabled	Intel Advanced Menu\Memory Configuration	When disabled, turns off memory power management.
Power Down Mode	Disabled	Intel Advanced Menu\Memory Configuration	When disabled, turns off memory power management.
RC6 (Render Standby)	Disabled	Intel Advanced Menu\Power & Performance\GT - Power Management Control	When disabled, prohibits the GPU from entering low-power state (RC6).
DMI Link ASPM Control	Disabled	Intel Advanced Menu\PCH-IO Configuration\PCI Express Configuration	When disabled, turns off Active State Power Management (ASPM) control for the DMI link.
Legacy I/O Low Latency	Enabled	Intel Advanced Menu\PCH-IO Configuration	When enabled, turns off PCH clock gating.

Setting Name	Option	Menu	Description
PCH PCI Express Configuration	For each root port: • ASPM: Disabled • L1 Substates: Disabled	ASPM: Intel Advanced Menu\PCH-IO Configuration\PCI Express Configuration\PCI Express Root Port # L1 Substates: Intel Advanced Menu\PCH-IO Configuration\PCI Express Configuration\PCI Express Root Port #	ASPM: When disabled, turns off Active State Power Management (ASPM). ASPM is an autonomous hardware-based, active state mechanism that enables power savings even when the connected components are in the D0 state. After a period of idle link time, an ASPM Physical-Layer protocol places the idle link into a lower power state. L1 Substates: When disabled, turns off PCIe L1 substates. The fundamental idea behind L1 substates is to use something other than the high-speed logic inside the PCIe* transceivers to wake the devices. The goal is to achieve near zero power consumption with an active state.
PCI Express Clock Gating	Disabled	Intel Advanced Menu\System Agent (SA) Configuration\PCI Express Configuration	When disabled, turns off PCIe clock gating. This feature gates platform reference clocks in certain PCI Express Link Power Management States

Setting Name	Option	Menu	Description
CPU PCI Express Configuration	For each root port: • ASPM: Disabled • L1 Substates: Disabled	ASPM: Intel Advanced Menu\System Agent (SA) Configuration\PCI Express Configuration\PCI Express Port # L1 Substates: Intel Advanced Menu\System Agent (SA) Configuration\PCI Express Configuration\PCI Express Port #	ASPM: When disabled, turns off Active State Power Management (ASPM). ASPM is an autonomous hardware-based, active state mechanism that enables power savings even when the connected components are in the D0 state. After a period of idle link time, an ASPM Physical-Layer protocol places the idle link into a lower power state. L1 Substates: When disabled, turns off PCIe L1 substates. The fundamental idea behind L1 substates is to use something other than the high-speed logic inside the PCIe transceivers to wake the devices. The goal is to achieve near zero power consumption with an active state
Intel® Speed Shift Technology	Disabled	Intel Advanced Menu\Power & Performance\CPU - Power Management Control	When disabled, turns off Intel® Speed Shift Technology. The technology enables the management of processor power consumption via hardware performance state (P-State) transitions.
Intel® Turbo Boost Max Technology 3.0	Disabled	Intel Advanced Menu\Power & Performance\CPU - Power Management Control	When disabled, turns off Intel® Turbo Boost Max. Intel Turbo Boost Max Technology 3.0 enabled CPUs have a core ordering determined at manufacturing time, which allows certain cores to reach higher turbo frequencies (when running single-threaded workloads) than others.

6.2 System PM

6.2.1 Intel® Turbo Boost Max Technology 3.0 (P0 States)

Intel® Turbo Boost Max Technology 3.0 uses the principle of leveraging thermal headroom to dynamically increase processor performance for single-threaded and multi-threaded/multi-tasking environment. P0 states are also known as turbo states or P0 frequency is usually referred to as turbo frequency.

Intel® Turbo Boost Max Technology 3.0 is set to disabled.

6.2.2 Enhanced Intel SpeedStep® Technology and SpeedStep® (P-States)

P-states are the various execution power states of the processor. These are the frequency-voltage pairs that dictate the speed at which the processor will run.

Conventional Intel SpeedStep® Technology switches both voltage and frequency in tandem between high and low levels in response to processor load. A later version of SpeedStep® called Enhanced Intel SpeedStep® Technology was released. It is an advanced means of enabling high performance while meeting the power-conservation needs of mobile systems. Enhanced Intel SpeedStep® Technology builds upon that architecture using design strategies such as Separation between Voltage and Frequency Changes, and Clock Partitioning and Recovery.

These two items are generally known as SpeedStep® and is currently set to disabled, as P-states transitions introduce latencies.

6.2.3 C-States

C-states are the various idle states the individual cores can be in and are meant to save power while the processor or cores are not executing any instructions. There are many C-states ranging from C0 where the CPU is fully turned on to C6 where the CPU internal voltages are reduced to any value, including 0V (also referred to as deep power down).

C-states are disabled for real-time configurations as transitions between C-states introduce latencies.

6.2.4 S-States

S-states are sleep states of a system ranging from S0, the most demanding S-state, to Sx, where x > 0, which could range from sleep to hibernate.

S-states are disabled for real-time configurations as transitions between S-states introduce latencies.

6.3 Graphics Power Management

The Intel integrated graphics engine manages power and frequency, which impacts WCET and latency for real-time workloads.

6.3.1 Intel® Graphics Render Standby Technology

Intel® Graphics Render Standby Technology (Intel® GRST), RC6, or RC6+ adjusts the integrated graphics engine's voltage very low, or very close to zero when the system is asleep. In some cases, RC6 has caused latency spikes in real-time workloads. Therefore, RC6 should be disabled to improve real-time performance.

6.3.1.1 RC6 Linux Disablement

Because of the way the Linux i915 graphics driver handles RC6, this feature must be disabled in both the BIOS and in the i915 Linux graphics driver. The Linux command-line interface command used to disable RC6 are shown below:

```
echo 0 > /sys/class/drm/card0/gt_rc6_enable
```

The user can check if RC6 is enabled by sampling the RC6 residency counter, delaying and then sampling the counter again. When comparing the output values of the two samples, different values indicate RC6 is still enabled, identical values indicate RC6 is disabled.

In the example below, the user samples the residency counter, disables RC6, then samples the residency counter twice with a delay in between.

```
root@intel-corei7-64:~# cat
/sys/class/drm/card0/power/rc6_residency_ms
58456
root@intel-corei7-64:~# echo 0 >
/sys/class/drm/card0/gt_rc6_enable
root@intel-corei7-64:~# cat
/sys/class/drm/card0/power/rc6_residency_ms
61422
[Delay]
root@intel-corei7-64:~# cat
/sys/class/drm/card0/power/rc6_residency_ms
61422
```

Different values for the first two samples show the system entered RC6. After disabling RC6, identical values for the second two samples show the system did not enter RC6.

Note: This is a quick check and only confirms that the system did not enter RC6 between the two samples

Using the BKC referenced in Section 2.6 of this guide will ensure that the capability to disable RC6 is present in the Linux graphics driver. The method of disabling RC6 described in this section will likely not work on other versions of the graphic driver.

6.3.2 Configure GT for Use

The Intel CRB BIOS provides a BIOS setting named "Configure GT for Use" which initializes the GT and configures it for handoff to the GFX driver. For stability purposes, this option must always be enabled.

6.3.3 Display C-States

Display Engine provides internal power states that place the component into low-power modes of operation (DC0–DC9) that are initiated during Package C-State flows. For example, the system cannot enter package C10 unless Display C-States are enabled. It is recommended to disable Display C-States to improve real-time I/O performance.

6.4 I/O PM

There are several stages of PM in the I/O system. The fabric where multiple I/O units are switched can have its own PM. The individual I/O units can have its own PM as well.

6.4.1 I/O Fabric PM

The I/O system can have one or more stages of I/O switch fabric. Each of these will likely have its PM unit to manage power consumption more efficiently.

Note: **Turning off I/O Fabric PM may prevent sleep, hibernation, and resume flows. If crashes or hangs are encountered during or after system states (for example, S3, S4, S0ix), ensure this BIOS setting is Disabled.**

6.4.2 Individual I/O System PM

Currently, the I/O system that is of importance to real-time performance is the Peripheral Component Interconnect express* (PCIe*) interface. Intel® TCC/TSN-capable devices are currently plugged into the PCIe expansion slots. The power management mechanism for PCIe is called ASPM, L1, etc. Currently, to get the best real-time performance out of the system, these are generally disabled in BIOS.

§

7.0 Intel® TCC Features Tuning

After system software and power management have been tuned, the system is running at the optimal hardware performance using the basic platform capabilities. To further improve real-time performance, the platform provides a set of Intel® Time Coordinated Computing (Intel® TCC) features.

Intel® TCC feature tuning targets specific transaction flows, compute workloads, or corner cases. This tuning often requires a tradeoff of performance between the WCET of critical real-time paths, and best-effort workloads or average execution time (throughput).

Intel® TCC Mode configures relevant Intel® TCC features to predetermined values that have shown to provide the best general boost to real-time performance, though the best performance for a specific workload may require independent configuration of Intel® TCC features.

7.1 Intel® TCC Mode (BIOS): Intel® TCC Feature Settings

7.1.1 Overview

The BIOS provides an option that configures many individual settings in a single location. These settings include:

- Disabling features that are known to negatively impact latency/jitter
- Enabling features known to increase temporal isolation
- Configuring the platform to bias more toward lower latency vs. throughput or power efficiency

7.1.2 Architecture

Some notable features affected by Intel® TCC Mode include the following (details of these features are covered below):

- Cache allocation, including GT COS
- PCIe*/fabric virtual channels

7.1.3 Usage

Customers who desire the increased temporal isolation and low latency afforded by this setting and are willing to accept the trade-off in terms of reduced best-effort performance and increased power consumption, would enable this capability in the BIOS. As a result of the limitations of Intel® TCC Mode, Intel recommends using the data streams optimizer.

Customers that may not accept this trade-off can consider using Intel® TCC Tools which allow for more granular use case-specific optimizations by configuring advanced cache management, the I/O fabric, power management and best-effort throughput.

For more information about the tool, see the Intel® TCC Tools product page.

7.1.4 Intel® TCC Mode (BIOS): Intel® TCC Feature Settings

When enabled, Intel® TCC Mode sets the following Intel® TCC options:

Table 6. Intel® TCC Mode (BIOS): Intel® TCC Feature Settings

Setting Name	Option	Menu	Description
GT COS	Enabled	Intel Advanced Menu\Intel® Time Coordinated Computing	When enabled, limits the number of cache ways the GPU can allocate into in the L3.
WRC Feature	Enabled	Intel Advanced Menu\System Agent (SA) Configuration	When enabled, turns on Intel® Data Direct I/O Technology (Intel® DDIO).

Setting Name	Option	Menu	Description
CPU PCI Express Configuration	For each root port: • PTM: Enabled • VC: Enabled • Multi-VC: Enabled	PTM: Intel Advanced Menu\System Agent (SA) Configuration\PCI Express Configuration\PCI Express Port # VC: Intel Advanced Menu\System Agent(SA) Configuration\PCI Express Configuration\PCI Express Port # Multi-VC: Intel Advanced Menu\System Agent (SA) Configuration\PCI Express Configuration\PCI Express Port #	**PTM**: When enabled, provides a hardware mechanism for PCIe devices to correlate clock domains between an endpoint and the root complex. **VC**: When enabled, turns on PCIe virtual channel capability. PCIe virtual channels provide a mechanism for differentiating PCIe traffic throughout the fabric. **Multi-VC**: When enabled, provides support for PCIe virtual channel capability.
PCH PCI Express Configuration	For each root port: • PTM: Enabled	PTM: Intel Advanced Menu\PCH-IO Configuration\PCI Express Configuration\PCI Express Root Port #	**PTM**: When enabled, provides a hardware mechanism for PCIe devices to correlate clock domains between an end point and the root complex.

7.2 Cache Allocation Technology (CAT)

7.2.1 Overview

Cache Allocation Technology (CAT) allows shared caches to be partitioned at the way level between classes of service. Judicious use of CAT can greatly reduce the jitter in real-time applications due to disturbances in the cache state due to sharing. For optimal real-time performance, the real-time process should have exclusive access to the cache ways it will be using.

The processor provides support for CAT on the Level 2 cache (L2) and last level cache (LLC, also known as L3). Programming interfaces for L2 CAT are architectural and can be discovered by referring to the *Intel® 64 and IA-32 Architectures Software Developer's Manual,* Chapter 17.19 "Intel® Resource Director Technology (Intel® RDT) Allocation Features." L3 CAT is model specific and non-architectural. As such, programming interfaces for L3 CAT may not align with the Intel Software Developer's Manual and therefore are provided here for completeness.

7.2.2 Architecture

Details on the CAT architecture can be found in *Intel® 64 and IA-32 Architectures Software Developer's Manual*, Chapter 17.19 "Intel® Resource Director Technology (Intel® RDT) Allocation Features."

7.2.3 Usage

Generally, the hypervisor partitions resources between itself and guest operating systems. In native environments, the operating system partitions the resources between processes or CPUs/cores. Hypervisors and operating systems can use CAT to partition shared caches such as the L3. The hypervisor can optionally allow the guest operating systems to further partition its portion of the cache between processes, although this would require hypervisor support to trap and emulate the correct CAT functionality while enforcing the partitioning that the hypervisor originally defined. It is not expected that user applications would directly use CAT. As a result, Intel provides enhanced cache management capabilities via Intel® TCC Tools. For more information, see the Intel® TCC Tools product page.

7.2.3.1 Detecting Cache Allocation Technology Support

To detect the presence of L2 CAT, see Chapter 17.19.4.2 "Cache Allocation Technology: Resource Type and Capability Enumeration" in Volume 3B of the *Intel® 64 and IA-32 Architectures Software Developer's Manual*.

To detect the presence of L3 CAT, attempt to read MSR 0xC90. If the system generates a fault, L3 CAT is not supported.

7.2.3.2 Detecting Code Data Prioritization (CDP) Support

To determine support for Code Data Prioritization, see Chapter 17.19.4.2 "Cache Allocation Technology: Resource Type and Capability Enumeration" in Volume 3B of the *Intel® 64 and IA-32 Architectures Software Developer's Manual*.

7.2.3.3 Detecting the Number of Classes of Service

To detect the number of L2 Classes of Service, see Chapter 17.19.4.2 “Cache Allocation Technology: Resource Type and Capability Enumeration” in Volume 3B of the *Intel® 64 and IA-32 Architectures Software Developer’s Manual*.

The processor has four L3 Classes of Service.

7.2.3.4 Detecting the Capacity Bit Mask Length

To detect the Capacity Bit Mask Length (CBM) for L2 CAT, see Chapter 17.19.4.2 “Cache Allocation Technology: Resource Type and Capability Enumeration” in Volume 3B of the *Intel® 64 and IA-32 Architectures Software Developer’s Manual*.

The L3 CAT CBMs are defined as follows:

Processor	CBM
i3-1115GRE (11th Gen Intel® Core™ i3 Processor)	12
i5-1145GRE (11th Gen Intel® Core™ i5 Processor)	8
i7-1185GRE (11th Gen Intel® Core™ i7 Processor)	12

7.2.3.5 Intel® Architecture Class of Service Cache Mask Register Details

To detect the L2 CAT cache mask details, see Chapter 17.19.4.3 “Cache Allocation Technology: Cache Mask Configuration” in Volume 3B of the *Intel® 64 and IA-32 Architectures Software Developer’s Manual*.

Since the number of L3 classes of service is four, the L3 CAT cache mask details are as follows:

MSR range: 0xC90 – 0xC93 for COS0 through COS3, respectively.

7.2.3.6 IA Class of Service Register Details

For register details on how to select a class of service, see Chapter 17.19.4.3 “Cache Allocation Technology: Cache Mask Configuration” in Volume 3B of the *Intel® 64 and IA-32 Architectures Software Developer’s Manual*.

There is only one class of service register, referred to as the IA32_PQR_ASSOC, that sets the class of service for both L2 CAT and L3 CAT. This is a per-logical thread.

```
IA32_PQR_ASSOC = MSR 0xC8F
```

Note: RMID is not supported.

7.3 GT COS

7.3.1 Overview

Since the integrated GPU shares the L3 with the CPU cores, the GPU can also be considered a *noisy neighbor* for the shared resource of the L3. GT COS refers to the GPU’s interaction with CAT, specifically which cache ways of the L3 it can allocate into.

The traditional Intel use case of a desktop or laptop benefits from a high-performing GPU, for example, when the user is playing a game and wants maximum GPU performance. As a result, the default configuration is to favor maximum GPU performance, by allowing the GPU to use as much of the L3 as it needs. This, however, creates a significant source of jitter to real-time applications. Using the GT COS capability, Intel® TCC Mode severely limits the number of cache ways the GPU can allocate into in the L3. This sacrifices GPU performance in favor of lower jitter for the real-time task running on the CPU.

7.3.2 Architecture

This microarchitectural capability is managed exclusively by firmware, and thus has no direct or indirect software interaction outside of setting Intel® TCC Mode in the BIOS. Intel® TCC Mode limits the number of cache ways available to the GPU.

7.3.3 Usage

Once enabled in the BIOS via Intel® TCC Mode, no additional effort is needed. However, for optimal results, software should avoid the cache ways allocated to the GPU.

If Intel® TCC Mode is *enabled*, the following cache ways are allocated to the GPU:

- COS0=0x1
- COS1=0x1
- COS2=0x1
- COS3=0x1

All 4 COS masks are intentionally set to the same value to avoid situations where the graphics driver attempts to change the active COS, particularly when the GPU is assigned to a virtual machine. The driver can select which COS is active but is unable to change the mask definitions.

If Intel® TCC Mode is *disabled*, the following GPU capacity bit masks are set:

- COS0=0xFFF
- COS1=0xFC0
- COS2=0xC00
- COS3=0x800

COS0 is the default selected GT COS.

7.4 OPIO Recentering

During operation, the PCH "recenters", or correlates the time between the CPU and PCH. This recentering can delay traffic moving between the core and the PCH, so an optimal value for real-time workloads must be configured. The Intel customer reference board BIOS provides an "OPIO Recentering" BIOS setting that configures this optimal setting when enabled. It is recommended to disable this BIOS setting to improve real-time performance.

7.5 Data Direct Input/Output (DDIO) / Write Cache (WRC)

7.5.1 Overview

I/O streams, such as PCIe* MMIO read/write, can be considered "temporally local," that is, the data is used by the CPU right after it arrives. Especially in cyclic real-time use cases, placing the data directly in the cache provides lower latency for the CPU when processing the data, and reduces unnecessary memory traffic, improving general performance and power efficiency.

Data Direct Input/Output (DDIO) addresses this issue by allowing I/O devices to have their writes allocate directly into specified ways of the L3 cache. Data Direct IO (DDIO) was originally introduced in Intel® Xeon® processors. A variant of this capability is available on selected 11th Gen Intel® Core™ processors as the WRC or write cache.

7.5.2 Usage

Once enabled in BIOS via the Intel® TCC Mode, no additional effort is needed. However, for optimal results, software should consume the data as quickly as possible to ensure it is consumed from cache rather than memory, as subsequent I/O writes will victimize older data if the CPU has not consumed it.

If Intel® TCC Mode is *enabled*, the following cache ways are allocated to the WRC:
COS3 = 0x001

Note: VCrt is fixed at COS3, so only this capacity bit mask can be selected when WRC is enabled for VCrt.

If Intel® TCC Mode is *disabled*, the following WRC capacity bit mask is set to:
COS3=0x800

Additionally, a different capacity bit mask can be selected using the cache configurator. For more information about the cache configurator, see the Intel® TCC Tools product page.

7.6 Upstream Virtual Channels

7.6.1 Overview

In real-time applications, it is necessary to allow critical transactions, both reads and writes, to pass low-priority transactions that may be congesting the data path. The PCIe specification from PCI-SIG introduces Virtual Channels and the concept of Traffic Classes (TCs), which allow PCIe transactions to be prioritized by applying service policies.

There is still a need to optimize the entire data path. The concept of fabric Virtual Channels addresses this by extending the PCIe Virtual Channel concept to the internal data buses.

The fabric provides multiple Virtual Channels (VCs) that differentiate between traffic types by distributing and ordering traffic between VCs, prioritizing VCs through arbitration, and tailoring VC attributes to each VC's specific purpose.

Note: No ordering relationship is enforced between transactions on different VCs. Thus, transactions within one VC may pass those of another within the fabric.

VCs are given a "VCx" designation where “x" is some identifier (for example, VC0, VC1, VC1a, VC1b, VCrt, VCbr). A general-purpose VC (often designated VC0) optimized for high bandwidth and given standard priority is the default VC for generic traffic. Common purposes for VCs include different types of Display Engine (DE) traffic and Image Processing Unit (IPU) traffic, as well as real-time traffic.

VCs are implemented on a per-hop basis, so each IP and each sub-component within an IP must explicitly enable each VC. It is possible that some IPs do not support one or more VCs, thus creating a bottleneck where VC transactions from multiple VCs share queues which eliminates differentiation and serializes the data path. In order to take full advantage of this feature, end-to-end (E2E) VCs support VCs along each hop between source and destination of a given data path.

To support real-time applications, the fabric implements a low-bandwidth, high-priority VC that is generically and sometimes architecturally referred to as VCrt. In this context, "Virtual Channels" henceforth refers to support for VCrt.

7.6.2 Architecture

7.6.2.1 PCIe Virtual Channels

PCIe Virtual Channels are limited in scope to the PCIe controllers, switches, and devices that make up the local PCIe hierarchy of a platform. While in practice, the same TC and VC concepts are usually shared end to end, the fabric may implement VCs differently than PCIe while still adhering to the PCI-SIG specification.

A PCIe root port that supports PCIe VCs must correctly configure the VC Capability structure to denote which VCs are supported, and the TC/VC mapping. The VC Capability structure is defined in the PCIe Base Specification.

7.6.2.2 Ethernet Virtual Channels

Ethernet Virtual Channels extend the PCIe VC concept to Ethernet endpoints by providing mechanisms to specify which transactions should use VCrt.

Note: For more information on Ethernet Virtual Channels (such as registry values), contact your Intel representative.

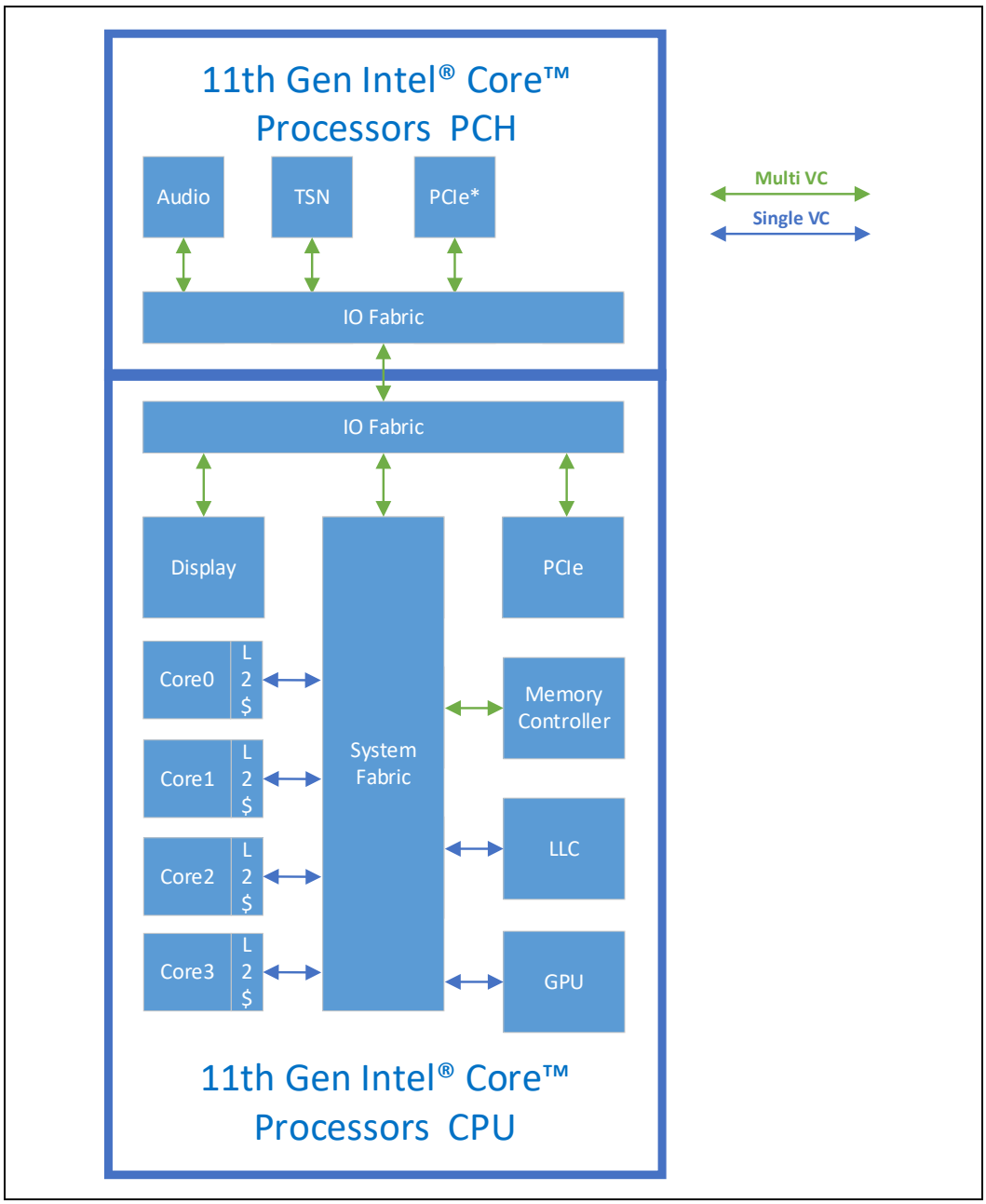
7.6.2.3 Fabric Virtual Channels

Fabric Virtual Channels include VC implementation in the SoC fabric from the coherent domain through the I/O fabric. Each stage in the hardware pipeline from the source of a transaction (for example, PCIe) to the destination (for example, memory) may implement VCs to the next hop upstream or downstream. Any stage that doesn't implement VCs may cause a bottleneck.

The main IPs that implement fabric VCs include:

- IOSF (I/O Scalable Fabric)
- CMF (Coherent Memory Fabric)

Figure 5. Fabric Virtual Channels Block Diagram



7.6.3 Usage

With Intel® TCC Mode enabled, TC0 uses the high-bandwidth, best-effort virtual channel, and TC1–TC7 use the low-bandwidth, real-time virtual channel.

For register specifications, see the BIOS Writer's Guide listed in Reference Documents.

7.7 Time Synchronization

Time synchronization features are available by default; no platform tuning is needed to enable these features.

Intel® TCC Tools provide sample applications that demonstrate use of these features. For more information, see the Intel® TCC Tools product page.

7.8 Intel® TCC Related Features

7.8.1 Intel® TCC Mode (BIOS): Intel® TCC Related Settings

Intel® TCC Mode sets additional options that enable real-time processing. When enabled, Intel® TCC Mode sets the following options:

Table 7. Intel® TCC Mode (BIOS): Intel® TCC Related Settings

Setting Name	Option	Menu	Description
Hyper-Threading	Disabled	Intel Advanced Menu\CPU Configuration	When disabled, turns off Intel's proprietary simultaneous multithreading implementation.

7.8.2 Real-Time Configuration Management

Real-time configuration management is the framework that enables two new technologies, data streams optimizer and cache allocation. These technologies assist real-time developers looking to optimize the performance of their latency-sensitive applications.

For more information, refer to the Developer Guide on the Intel® TCC Tools product page.

§

8.0 Fabric Tuning

Fabric tuning includes microarchitecture-specific optimizations for functional components in the fabric that include prioritization, arbitration, and flow control. These tunings come at a high cost in implementation time and performance tradeoffs, and they provide comparatively little improvement relative to the other types of tuning. As such, manual fabric tuning is not recommended.

However, since the best possible real-time performance can be attained by tuning the fabric arbitration and prioritization characteristics, the data streams optimizer provides a mechanism for software-assisted fabric tuning. For more information about the tool, see the Intel® TCC Tools product page.

§

9.0 Terminology

Table 8. Terminology

Term	Description
API	Application Programming Interface
ASPM	Active State Power Management
BKC	Best Known Configuration
BSP	Board Support Package
CAT	Cache Allocation Technology
CBM	Capacity Bit Mask
CDP	Code Data Prioritization
CMF	Coherent Memory Fabric
CPS	cyber-physical systems
CRBs	Customer Reference Boards
Deadline	The time when some computation or data must complete or arrive. For some applications, computations or data that arrive late are no longer useful.
DDIO	Data Direct Input/Output
DE	Display Engine
E2E	end-to-end
EDS	External Design Specification
GT COS	Graphics Technology Class of Service
IHS	Integrated Heat Spreader
Intel® GRST	Intel® Graphics Render Standby Technology
Intel® RDT	Intel® Resource Director Technology
Intel® TCC	Intel® Time Coordinated Computing A modern approach to architecting distributed, synchronized, scalable computing systems that address real-time application requirements for cyber-physical systems (CPS). Intel® TCC moves beyond traditional real-time systems based on simple microcontrollers.
IOSF	Input/Output Scalable Fabric

Term	Description
IoT	Internet of Things
Jitter	The difference between the maximum and minimum of some quantity, such as latency measured in units of time. Jitter matters a lot at sensors and actuators, but other mechanisms (such as TSN mechanisms) typically hide software-execution jitter (so long as WCET bounds are satisfied).
KPI	Key Performance Indicator
L2	Level 2 cache
LL3/L3	Last level cache
Latency	The duration of time between two events; for example, the time a signal is detected, and a response is received, or the time between an application sending a UDP message until it arrives on the Ethernet wire, or the time required to execute a code segment.
Noisy neighbor	An application or device, the functioning of which, affects the device or application with temporal requirements (e.g., because of shared resources). Temporal Isolation seeks to reduce the deleterious effect of a noisy neighbor.
OPC UA	A platform-agnostic standard for communication between devices using an “Unified Architecture”, created by the OPC Foundation focusing on the needs of industrial automation.
PCIe*	Peripheral Component Interconnect express*
PM	Power Management
PREEMPT_RT	The PREEMPT_RT patch (also the -rt patch or RT patch) makes Linux into a Real-Time Operating System (RTOS) to a large degree.
RCU	Read-Copy-Update
RDC	Resource & Documentation Center
Real-time application	An application that has a requirement to complete execution within some WCET with a specified level of reliability. For example, “Has to finish running every millisecond without missing a deadline in 7 days.” Typically, a real-time application contains a sequence of three tasks: sense > compute > actuate and increasingly uses a network to interconnect these. The extent to which a missed deadline impacts the overall system is sometimes described using “soft”, “firm”, and “hard” real-time, but Intel avoids these terms, preferring to quantify the reliability with number of 9s.
RTCP	Real-Time Compute Performance
RTOS	Real-Time Operating System

Term	Description
SA	System Agent
TC	Traffic Classes
Temporal Isolation	The degree to which a system can meet the time-related requirements of a real-time workload when the workload is running alongside other workloads on the system. In a typical system, concurrent workloads create contention for shared resources that can cause spikes in latency and increased jitter for the real-time workload. Intel® TCC capabilities help mitigate concurrent workload interference.
Time-Sensitive Networking (TSN)	A task-group of IEEE 802.1 that creates / amends the Ethernet and other standards, enabling dramatically better worst-case time performance (time-synchronization & latency). Also used to describe the standards / amendments from the TSN task group. https://1.ieee802.org/tsn/
Time synchronization	The degree to which two or more systems or devices agree on what time it is, to within some maximum error. Enables sensors, compute systems, actuators, and network elements to operate on a global schedule. Enables time-coordinated computing devices to time-division multiplex real-time and non-real-time tasks, measure latencies, and detect violation of deadlines. Enables an RTOS to schedule a task at a specific time.
VCs	Virtual Channels
Workload	An application that performs some useful computational work, including (perhaps) receiving input, performing computation, and generating an output.
WCET	Worst-case execution time. The maximum measured latency of the compute portion of an application, across multiple iterations. WCET relates to reliability, described by “the number of 9s”. For instance, a reliability of two 9s refers to missing the deadline 1 out of 100 times.
WRC	Write Cache

§

10.0 Reference Documents

In the following table, documents with a document number are available on 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.

Table 9. Reference Documents

Document	Document No./Location
Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for IoT Applications, Datasheet, Volume 1	636112
11th Gen Intel® Core™ processors BIOS Specification (BIOS Writer's Guide) Addendum	616309
Yocto Project*-based Board Support Package for 11th Gen Intel® Core™ processors Release Notes	615079
Real-Time Gold Deck	627170
Event Histograms	https://git.kernel.org/pub/scm/linux/kernel/git/rt/linux-stable-rt.git/tree/Documentation/trace/histogram.rst?h=v5.10-rt
Intel® TCC Tools product page, with links to: • Get Started Guide • Developer Guide • Tutorial: Profile Applications with Intel® VTune™ Profiler • Guide to Operating System and Hypervisor Components for Cache Allocation • Intel® TCC Security for UEFI BIOS White Paper • Training • Release Notes	https://www.intel.com/content/www/us/en/developer/tools/time-coordinated-computing-tools/overview.html
Intel® 64 and IA-32 Architectures Software Developer's Manual	https://software.intel.com/en-us/articles/intel-sdm
PCI-SIG, PCI Express Base Specification Revision 2.1 (subscription based)	http://pcisig.com/specifications/pciexpress/base

Document	Document No./Location
Linux kernel command-line parameters	• https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/admin-guide/kernel-parameters.txt?h=v5.10 • https://www.linuxtopia.org/online_books/linux_kernel/kernel_configuration/re46.html
PREEMPT_RT patches	https://wiki.linuxfoundation.org/realtime/start
Get Started with Linux TSN Reference Software	https://github.com/intel/iotg_tsn_ref_sw This opensource project is a set of C-applications and scripts used to showcase different Ethernet-TSN features in Linux on specific Intel IOTG platform/software. These serve as a practical example for those interested in developing TSN-capable software. This project introduces three applications, each with their own set of examples.

§

Appendix A Variant Kernel Configurations

This appendix details the kernel boot parameters for tuning the performance of the platform. The parameters were based on kernel version 5.4 with the PREEMPT_RT patch and tested on BIOS TGLIFUI1.R00.3313.A03.2008071806.

Table 10. Kernel Command-Line Parameters for RTCP/XDP and Cyclic Test

Kernel Command-Line Parameter	RTCP Value	Cyclic Test
processor.max_cstate=0	✓	✓
intel.max_cstate=0	✓	✓
processor_idle.max_cstate=0	✓	✓
intel_idle.max_cstate=0	✓	✓
clocksource=tsc	✓	✓
tsc=reliable	✓	✓
nmi_watchdog=0	✓	✓
intel_pstate=disable	✓	✓
nosoftlockup	✓ ***Note:*** No value required	✓
idle=poll	✓	✓
noht	✓	✓
isolcpus=2-3	✓	✓
rcu_nocbs=2-3	✓	✓
irqaffinity=0,1	✓	✓
i915.enable_dc=0	✓ ***Note:*** Option is only required when the platform includes integrated graphics.	✓ ***Note:*** Option is only required when the platform includes integrated graphics.
i915.disable_power_well=0	✓ ***Note:*** Option is only required when the platform includes integrated graphics.	✓ ***Note:*** Option is only required when the platform includes integrated graphics.
hugepages=1024	✓	✓
mce=off	✓	✓
hpet=disable	✓	✓

Kernel Command-Line Parameter	RTCP Value	Cyclic Test
numa_balancing=disable	✓	✓
efi=runtime	✓	✓
BOOT_IMAGE=(hd0,gpt2)/boot/bzImage-linux-intel-ese-lts-rt-5.4-kernel	✓	✓
root=PARTLABEL=primary	✓	✓
apparmor=1	✓	✓
security=apparmor	✓	✓
mem_sleep_default=deep	✓	✓
mender.efi=PARTLABEL=efi	✓	✓
mender.primary=PARTLABEL=primary	✓	✓
mender.secondary=PARTLABEL=secondary	✓	✓
mender.data=PARTLABEL=data	✓	✓
mender.swap=PARTLABEL=swap resume=PARTLABEL=swap	✓	✓
rootwait	✓ ***Note:*** No value required	✓
console=ttyS0,115200	✓	✓
console=tty0	✓	✓
iommu=pt	✓	✓
nohz_full=2-3	✓	---
init=/sbin/preinit-env	✓	✓
console=ttyS4,115200n8	✓	✓
console=ttyS5,115200n8	✓	✓
i915.force_probe=*	✓	✓
i915.enable_guc=2	✓	✓

§