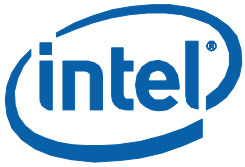

White Paper

Adrian Hoban
Software Engineer
Intel Corporation

# Designing Real-Time Solutions on Embedded Intel® Architecture Processors

May, 2010

323671

# Executive Summary

Intel develops a comprehensive portfolio of highly-advanced processors for the embedded market. The processors use the latest micro-architectural advancements and are implemented on leading-edge process technology.

Whether the embedded design focuses on networking applications, industrial control and automation, display centric functionality, or many other embedded designs, Intel has a range of high-performance, low-power consumption processors with advanced wired and wireless network connectivity, extended lifecycles, software tools and a comprehensive support model to enable the creation of intelligent, "real-time" solutions.

Embedded developers face many obstacles: how to increase performance, how to deliver faster network throughput, how to add innovative functionality, how to add "real-time" capability. These obstacles, coupled with the need to reduce footprint and power consumption over shorter development cycles represent a significant challenge. Whether the embedded design focuses on networking applications, industrial control and automation, display centric functionality, or many other embedded designs, Intel has a range of high-performance, low-power consumption processors with advanced wired and wireless network connectivity, extended lifecycles, software tools and a comprehensive support model to enable the creation of intelligent, "real-time" solutions.

One of the challenges facing some embedded designs is the ability to exhibit "real-time" determinism. This paper focuses on some of the key design considerations associated with "real-time" embedded solutions.

This paper defines a "real-time" system and distinguishes between "soft" and "hard" real-time systems. It introduces a number of the advanced features in Intel® architecture processors. For each feature, a sub-section describes real-time considerations and recommendations that a system designer may need to consider.

The paper concludes that Embedded Intel® architecture processors can be used effectively in embedded products that require real-time characteristics and notes the settings that a real-time developer should be aware of to add determinism to their product.

The Intel® Embedded Design Center provides qualified developers with web-based access to technical resources. Access Intel Confidential design materials, step-by step guidance, application reference solutions, training, Intel's tool loaner program, and connect with an e-help desk and the embedded community. Design Fast. Design Smart. Get started today: www.intel.com/embedded/edc. §

# Contents

# Introduction

Intel develops a comprehensive portfolio of highly-advanced processors for the embedded market. The processors use the latest micro-architectural advancements and are implemented on leading-edge process technology.

Embedded developers face many obstacles: how to increase performance, how to deliver faster network throughput, how to add innovative functionality, how to add "real-time" capability. These obstacles, coupled with the need to reduce footprint and power consumption over shorter development cycles represent a significant challenge. Whether the embedded design focuses on networking applications, industrial control and automation, display centric functionality, or many other embedded designs, Intel has a range of high-performance, low-power consumption processors with advanced wired and wireless network connectivity, extended lifecycles, software tools and a comprehensive support model to enable the creation of intelligent, "real-time" solutions.

## Real-Time

A "real-time" system is one which has a constraint on the time permitted for the system to respond to an event. The response must happen regardless of system loading. The performance requirements and the real-time requirements of the system are orthogonal. A real-time system must have deterministic, reliable predictability; the performance requirements (for example, in terms of packet throughput) are completely separate and may be relatively "low" even if real-time is required. An example of a real-time system is the Anti-lock Breaking System (ABS) on a vehicle. Failure of the system to respond to inputs from sensors indicating a wheel lock could have serious consequences.

A non-real-time system may have requirements for high performance or "quick" response characteristics, but an occasional failure to meet the requirements does not have a catastrophic impact on the system or user.

## Hard Real-Time vs. Soft Real-Time

Real-time systems can be categorized in two major groups, "hard" real-time systems and "soft" real-time systems.

A hard real-time system is one in which the late completion of an operation is considered useless, could result in a failure of the entire system or cause harm to the user. The ABS example above has hard real-time constraints. However, not all hard real-time systems have such fine-grained response

characteristics. Consider the hypothetical case where an industrial controller is used to monitor the temperature in a large vat of chemicals being heated slowly. Once a certain temperature is reached, the industrial controller needs to shut off power to the heating element. There may be a window of several minutes in which the controller has to respond. Failure to respond to the temperature event could be catastrophic, but as several minutes is a virtual eternity for a modern high-speed processor, this should be a trivial hard real-time requirement to meet.

A soft real-time system also has well-defined response requirements, but unlike hard real-time systems, they can cope with lateness. Usually, the lateness has a temporary negative impact on the system, but is recoverable and does not cause harm to the system or user. For example, in a video streaming application, a few dropped frames may temporarily decrease the quality of service. Although undesired behavior, this causes no system or user harm and the quality of service is quickly recovered.

It is very important that system designers make the distinction between hard real-time and soft real-time for their application early in the design process. This distinction can have significant impacts on design choices to be made later in the development cycle. Applying hard real-time constraints to a system that does not really need them can result in greater development costs (for example, the extra effort needed in the validation stage) or in lower application performance.

## Document Scope

The concepts discussed in this document are mostly applicable to the entire range of Embedded Intel® Architecture Processors. The primarily focus of this paper is on the real-time considerations for the higher performance processors. It is recommended that the reader also uses the specific processor product documentation to determine how these concepts apply.

# Platform Settings

Embedded Intel® Architecture Processors are equipped with a number of advanced features that increase performance capability while reducing energy consumption. As already noted, the overriding requirement of a hard real-time system is meeting its time constraints. Absolute performance is usually important, but is a secondary consideration. This section introduces some of the advanced features available on Embedded Intel® Architecture Processors and discusses the considerations that a real-time system designer needs to keep in mind.

## CPU C-States

C-State is a term taken from the Advanced Configuration and Power Interface (ACPI)[1] specification Ref [1]. The C-State represents the processor power state of the core. The C-State is often more commonly known as the processor "idle" state of the core. C-State values range from C0 to Cn, where n is dependent on the specific processor. When the core is active and executing instructions it is in the C0 state. Higher C-States indicate how deep the CPU idle state is.

C1 is known as the halt state and is mandatory for compliance with the ACPI specification. It is entered on Intel® architecture processors using the HLT instruction (or C1E when using the MWAIT instruction). There is a low latency in moving from the C1 to the C0 state. Higher C-States consume less energy when resident in that state, but also require longer latency times to transition into the active C0 state and require more energy to make the transition. Table 1 shows the power specification for the Intel® Xeon® processor 5500 series for various C-State settings Ref [2]. Significant power savings are available in the lower power states.

**Table 1. Intel® Xeon® Processor 5500 Series C-State Power Specifications**

| Package C-State(A) | Xeon® 5500 Series with 130W TDP | Xeon® 5500 Series with 130W TDP | Xeon® 5500 Series with 130W TDP(B) | Xeon® 5500 Series with 130W TDP (C) | Xeon® 5500 Series with 38W TDP |
|---|---|---|---|---|---|
| C0 | 130W | 95W | 80W | 60W | 38W |
| C1E | 35W | 30W | 30/40W | 22W | 16W |
| C3 | 30W | 26W | 26/35W | 18W | 12W |
| C6 | 12W | 10W | 10W | 8W | 8W |

Legend:
(A) - Specifications are at $T_{case}$ = 50ºC with all cores in the specified C-State
(B) - Standard/Basic SKUs
(C) - Applies to Low Power SKU and Intel® Xeon® Processor L5518

**Real-Time Considerations:**

The Operating System (OS) power management policy manages transitions between available CPU power states. The transition between adjacent power states incurs a small latency cost. This latency cost becomes a little more significant during transitions from the lowest idle states to the active state. For non-real-time applications and for many soft real-time applications the power state transition latency time is effectively irrelevant. However, for a

[1] The ACPI specification V3.0a defines the following states: Global system power states (G-states, S0, S5), System sleeping states (S-states S1-S4), Device power states (D-states), Processor power states (C-states), Device and processor performance states (P-states). Refer to the specification for details.

hard real-time application, C-State transitions can be a source of indeterminism. At the CPU level, there is a fixed time required to transition between states. The source of indeterminism comes from the OS power management policy reacting to the continuously changing active software loads. If the OS requests C-State transitions at a high frequency (for example, 1000 times per second) the small transition latency can become significant.

Linux* users can see the available C-States and related state transition latencies on their machine via the proc file system with the following command:

```
cat /proc/acpi/processor/*/power
```

A real-time system designer should consider these latency measurements in the context of their real-time budget. For example, if the system needs to respond to an event in a certain time, the maximum exit latency from the deepest sleep/C-State should be accounted for in the design.

It is usually possible to disable this power saving feature by modifying the C-State settings in the BIOS/UEFI. By disabling the C-State changes, the processor is forced to stay in the C0 active state. Any potential sources of indeterminism arising from C-State transitions are removed from the system at the cost of higher power consumption.

By default, C-State transitions are typically enabled in BIOS/UEFI Ref [5]. If a real-time system developer is having difficulty meeting some real-time constraints, it is recommended that a test is run with C-State transitions disabled. If this resolves the issue, then the choice needs to be made between deploying with C-States disabled and accepting the higher power consumption of the device, or invest some time in restructuring the workloads to account for the worst case C-State transition times.

## Enhanced Intel® SpeedStep® Technology

Enhanced Intel® SpeedStep® Technology is an advanced method of altering the processor operating frequency and voltage between high and low levels based on the processor load Ref [3]. This technology enables Embedded Intel® Architecture Processors to provide very high performance computing capability while also enabling low energy consumption. A frequency transition results in a small period of system unavailability. The enhanced version of this technology permits the frequency and voltage levels to be altered separately. The amount of leakage current from transistors in the CPU is a function of the voltage level. A reduction in the voltage helps to save energy by leaking less current. By altering the voltage level independently to frequency changes, the periods of unavailability can be reduced, the energy consumption can also be minimized and a better balance can be achieved between performance and energy efficiency.

The voltage frequency pair is known as the Device and Processor Performance State (P-State). A P-State of P0 is the highest voltage/frequency pairing. A high P-State will have lower voltage and frequency levels. It takes the processor longer to complete a task in a high P-State, but less energy is consumed.

The operating system is responsible for managing when the P-State transitions occur. It typically does this by assessing the CPU load over a fixed period. The period is usually in the order of tens to hundreds of milliseconds. If the average load over that period is sufficiently low, then the OS requests that the processor switch into a higher P-State (that is, lower voltage/frequency pair) for the next period. From a quality of service perspective, and looking only at the averages, the same amount of work can be completed in the next period. If the demand rises, the OS requests a switch into a lower P-State to compensate.

**Real-Time Considerations:**

A P-State transition that includes a frequency change results in the processor core and shared cache being unavailable for a small period during the transition. On the Intel® Xeon® 5500 processor series, this unavailability period is less than 2µSec. A real-time application may be sensitive to this period of unavailability, especially if the processor is switching between P-States that include a frequency transition at a high rate. P-State transitions that do not include a frequency change do not have an unavailability period.

The VTune™ Performance Analyzer can be used to report on the number of Enhanced Intel® SpeedStep® Technology transitions that occur using the EIST_TRANS[2] event. The Tuning Assistant warns about a high number of these transitions.

Another concern for real-time systems is that the OS looks at average load over a period to decide if it is possible to enter a higher P-State.

Consider the following example where the OS looks at 300 ms period to assess P-State transitions. If a high priority real-time operation completes in 100 ms when the processor is running a maximum capacity P0 state and for the remaining 200 ms of the period other less intensive tasks run. Overall the OS interprets the average CPU load as being 50% and requests that a higher

---

[2] This event counts the number of Enhanced Intel® SpeedStep® Technology transitions that include a frequency change, either with or without voltage change. This event is incremented only while the counting core is in the C0 state. Since the CxE states include an Enhanced Intel® SpeedStep® Technology transition, the event is incremented accordingly.

Enhanced Intel® SpeedStep® Technology transitions are commonly initiated by the OS, but are also initiated by the hardware internally. For example, CxE states are C-states (C1,C2,C3) that not only place the CPU into a sleep state by turning off the clock and other components, but also lower the voltage, which reduces the leakage power consumption. The same is true for thermal throttling transition which uses Enhanced Intel® SpeedStep® Technology internally.

P-State is entered for the next 300 ms. The processor continues in this energy saving state for some time, then the real-time task needs to execute again. As the CPU speed is now lower, the real-time task requires more time to complete (for example, 110 ms). The difference in execution time for the real-time operation of 10 ms could be detrimental to the real-time budget constraints of the system.

By default, P-State transitions are typically enabled in BIOS/UEFI Ref [5]. If a real-time system developer is having difficulty in meeting some real-time constraints, it is recommended that a test is run with Enhanced Intel® SpeedStep® Technology disabled. If this resolves the issue, the choice needs to be made between deploying with Enhanced Intel® SpeedStep® Technology disabled and accepting the higher power consumption of the device, or investing some time in restructuring the workloads to account for the Enhanced Intel® SpeedStep® Technology unavailability periods, and the longer time required to execute operations when running at the highest P-States.

## Intel® Turbo Boost Technology

Intel® Turbo Boost Technology Ref. [4] is capable of increasing core operating frequencies above the base level operating frequency. The increase in frequency is in increments of 133.33 MHz. The number of increments that are possible depends on the exact processor version. This feature is enabled when the Operating System (OS) requests the P-0 performance state. The processor includes measurements of temperature, power and current in a closed loop control system to determine if a frequency boost is permitted. The number of active logical cores in the system also has a bearing on permitting a frequency boost. An active core is one that is in the C0 power state. If the system deviates outside of the factory defined operating parameters, the processor automatically drops back to the base operating frequency. Intel® Turbo Boost Technology requires that the Enhanced Intel® SpeedStep® Technology is enabled.

**Real-Time Considerations:**

A real-time application system designer only needs to consider the effect of Intel® Turbo Boost Technology if Enhanced Intel® SpeedStep® Technology is enabled. The key assumption here is that the real-time application can meet its requirement for determinism using Enhanced Intel® SpeedStep® Technology. An additional consideration is that by enabling Intel® Turbo Boost Technology, it is likely that there will be more frequency transitions. As each frequency transition incurs a small period of unavailability, the extra periods of unavailability may become relevant to the real-time application.

It is recommended that the real-time capability of the system is confirmed with Enhanced Intel® SpeedStep® Technology enabled before also enabling Intel® Turbo Boost Technology. If the real-time capability of the system is

verified first (and only) with Intel® Turbo Boost Technology enabled, then the validation could be skewed by the chance that the key real-time operations were running and tested on a core that was running at the boosted frequency. If the real-time test operation managed to complete in 100 ms at the maximum boosted frequency, and required 95 ms to complete at the maximum non-boosted frequency, then this difference could be significant for meeting the real-time requirement. In a deployment scenario many factors, such as cooling/environmental, can affect the processor thermal state and possibly prevent it entering the boosted frequency zone.

For real-time systems, it is recommended that the system should not rely on the performance advantage of Intel® Turbo Boost Technology to meet a timing constraint. The real-time constraints of the system should be validated with Enhanced Intel® SpeedStep® Technology enabled, and subsequently verified with Intel® Turbo Boost Technology also enabled. Assuming both test setups pass the rigorous testing required by real-time systems, then deployment with Intel® Turbo Boost Technology can be considered and the system can reap the benefits of having even higher processing capability in certain operating cases.

## Cache

A cache is a temporary storage location that is used to reduce the access time to frequently accessed instructions or data. Intel® architecture processors support multiple levels of cache. Level 1 (L1) cache is the smallest in size and offers the lowest data accesses latency from the CPU. Level 2 (L2) cache typically offers quite a bit more temporary storage than L1 cache, but the access latencies increase. The Intel® Xeon® 5500 series processor also has a very large Level 3 (L3) cache (up to 8 MB). The L3 cache has larger access latencies than L1 or L2 caches, but it is still much faster than a memory access.

In general, the processor cache helps to increase the overall performance of the system and its operation is transparent to the software. Typically, the software developer does not need to exercise explicit control on the cache. Software performance can be optimized if the software developer is familiar with the size, structure and behavior of the cache.

The operation of the cache varies depending on the memory type defined for the memory being operated on. Up to six different memory types (cache types) are possible with Intel® architecture and are configurable down to the granularity of a memory page basis. The following is a brief description of the memory types:

- Strong Uncacheable (UC): Reads/Writes to memory locations are not cached and appear on the system bus in program order.

- Uncacheable (UC-): The same as UC, but also allows for some subsequent override via software control using the Memory Type Range Registers (MTRR).
- Write Combining (WC): Memory is not cached in the typical sense (that is, writes do not end up in any level of cache), but writes can be combined in a dedicated write combining buffer.
- Write Through (WT): Caching is enabled and all writes are written through to system memory.
- Write Back (WB): Caching is enabled and writes may be delayed in the cache until the processor needs to flush the cache line (for example, as a result of the Least Recently Used cache eviction algorithm). Typically, this type offers best performance.
- Write Protected (WP): Reads are cacheable, but writes are not.

Embedded Intel® Architecture Processors are capable of speculatively predicting that data is probably going to be needed by the pipeline in the near future and can read data into cache before the processor actually requires it. This is known as prefetching and helps to reduce the pipeline stalls that are attributable to waiting on memory accesses.

**Real-Time Considerations:**

In general, a software developer does not need to be concerned about the operation of the cache affecting a real-time attribute of the system. However, if there is a requirement that a write to memory happens in a defined period of time (for example, a write to a memory mapped register), then the software developer must consider the behavior of the cache, the write combining buffer and the store buffer since writes can get delayed at each of these locations.

Accesses to memory regions configured with the UC or UC- memory type are executed in program order. Accesses to memory defined with the other cache types are executed in processor order.

Assuming that paging is enabled, if the memory type for the page that contains the memory being accessed permits caching, then a write could remain in the cache for some non-deterministic period. It is possible to force the eviction of a cache line using the CFLUSH instruction. This forces the write operation to continue to the next step in the memory hierarchy.

The store buffer is used in an intermediate step when the processor executes a write to memory, a write to cache, or a write from cache to memory. The store buffer helps to decouple the processor execution pipeline from the low-level bus accesses by permitting the pipeline to continue executing while the write operation completes. It can also help to optimize the processors utilization of bus bandwidth by delaying writes to combine them into larger block writes. The possible delay in the store buffer is non-deterministic. Use

of a memory ordering instruction, such as SFENCE or MFENCE, can force the processor to drain the store buffer and hence force completion of the write.

Writes can be combined in the write combining buffer and can be delayed there. When a write combining buffer is filled, the processor can chose to write that data to memory (via the store buffer). When this write happens, is implementation-specific. Partial writes of data from the write combining buffer are also possible. Again, the use of a memory ordering instruction, such as SFENCE or MFENCE, can force the data in a write combining buffer to be written through to memory.

## Non-Uniform Memory Architecture

The key defining characteristic of a Non-Uniform Memory Architecture (NUMA) is that from the processors perspective, there is a variable time required to access different memory locations. The Intel® Xeon® 5500 processor series has NUMA capability. In a dual-processor configuration, this means one processor requires less time to access memory local to it than to access memory local to the other processor (remote memory). A remote memory access by a processor results in a memory transaction that must first cross the Intel® Quick Path Interconnect (Intel® QPI) bus before getting to the memory controller for the remote processor. Although the comparison of local and remote memory access latency times shows local memory accesses to be impressively fast, it should be noted that even the remote memory access latency times compare favorably with previous generation products.

The software developer should leverage the NUMA features of the OS they are using to ensure local memory accesses are the predominant type in the system. Not all OSes have NUMA capability. For example, at the time of writing, 32-bit Linux* variants are not “NUMA aware”. In general, the best system performance (in terms of memory bandwidth and latency) is achieved for non-NUMA aware OSes when NUMA is disabled in the BIOS. By disabling NUMA in the BIOS/UEFI, the system switches into an interleaved memory mode which distributes memory transactions between the different memory controllers in the system and also helps to balanced memory transactions across the Intel® QPI bus.

**Real-Time Considerations:**

A properly configured system with NUMA capability provides the processor with the benefit of optimal memory bandwidth combined with low memory access latencies. These lower memory latencies may help a system designer to achieve a very fine-grained real-time target.

A system with NUMA capability that has not been configured in an optimal way can also enable real-time semantics in the system. However, as a result of the lower bandwidth capability and higher memory latency than can be achieved in a properly configured system, the granularity of the real-time requirement may need to be slightly higher for it to be achieved.

## System Management Interrupt

On Embedded Intel® Architecture Processors, System Management Mode (SMM) is a special purpose operating mode intended for use by the firmware that helps manage system-wide functions such as power management and system hardware control. This mode allows the firmware to run in a distinct and isolated environment that is transparent to the OS and software applications running on the OS.

A System Management Interrupt (SMI) is required to invoke SMM. On transition to SMM, the processor saves the current state to the System Management RAM (SMRAM), then jumps to the separate operating environment that is contained in the SMRAM. After the processor has switched into SMM, the SMI handler is invoked to perform operations such as powering down idle disk drives or monitors. Once the SMM task has completed, the processor reloads the previous state and switches back to the previous mode (for example, real or protected).

**Real-Time Considerations:**

SMIs can be problematic for real-time application developers. Not only are SMIs the highest priority interrupts in the system (Non-Maskable Interrupts (NMIs) included), the OS does not control them and is largely unaware of their activity. The SMI handlers can take hundreds of microseconds to complete. This time can pose an issue for some real-time applications.

The real-time behavior of multi-core devices can also be affected by SMIs. An SMI is handled by one processor core at a time. Consider the case where core A acquires a spinlock that core B is contenting on. While holding the spinlock, core A handles an SMI. Core B must now wait until the SMI handler has completed its task on core A, core A has resumed the application that acquired the spinlock and the application releases the lock.

Despite the obvious difficulties for real-time applications, it is extremely important that SMIs are NOT disabled globally to avoid serious harm to the processor. SMIs are responsible for carrying out some critical tasks such as those related to managing the temperature of the processor. Instead, a real-time system designer should look at some techniques to reduce the type, number and frequency of SMIs. Here are some suggestions:

- Disable USB mouse and keyboard support in the BIOS.
  - The BIOS can use the SMI to access the USB host controller to monitor status for legacy USB support. By using a PS/2 mouse and keyboard and disabling USB mouse and USB keyboard functionality in the BIOS, the number of SMIs can be significantly reduced.
- Use an Advanced Configuration and Power Interface (ACPI) enabled OS.
  - The older Advanced Power Management (APM) technology allowed the BIOS to control the power management for the processor. ACPI gives control to the OS instead and results in fewer SMIs being generated.
- Disable Total Cost of Ownership (TCO) timer generation of SMIs.
  - TCO refers to a logic block that is included in Intel chipsets. The Watchdog Timer (WDT) is one of the pieces of functionality provided in the TCO block. The frequency of this interrupt is typically very low, for example, the interrupts may be several seconds apart. It may take hundreds of microseconds to service the interrupt, so a real-time system designer may forego the Watchdog Timer functionality in favor of greater determinism.
  - The SMI interrupt for the Watchdog Timer can be disabled by writing to the TCO_EN bit of the SMI_EN register. Refer to the relevant datasheet for the particular chipset model for more details. The Watchdog functionality is not available when the SMI for the Watchdog Timer is disabled.

# Conclusion

It is possible to design and implement a system with hard real-time attributes using Embedded Intel® Architecture Processors. Depending on granularity of the real-time specification, the behavior of a number of the advanced features in the processors may need to be considered during the development.

It is imperative that a system designer understands how to properly classify the system software in terms of its real-time requirements. For a system that needs hard real-time support, under specifying the system software with no real-time requirements or soft-real time requirements can result in a system that does not respond to events as expected, and in worse case scenarios could lead to catastrophic system failure or user harm. Conversely, over specifying a system that does not need hard real-time behavior, with hard real-time requirements can have less dramatic consequences but the

potential reduction in system performance could leave a product struggling in the market place beside more accurately specified competitive products.

Embedded Intel® Architecture Processors are equipped with numerous advanced features that help optimize different attributes of the system. For example, Intel® Turbo Boost Technology can help deliver increased performance, and Intel CPU C-State changes can help deliver very low power consumption levels for the processor. Many of the advanced features are flexible enough to allow precise configuration and control by the platform software and in some instances if required, they can even be disabled completely. It is highly recommended that real-time system designers consider the capability of the advanced Intel® architecture features and understand how these features could impact the overall system attributes.

Specifically, the real-time developer should test that the system meets the real-time requirements with C-States enabled. If there is not enough determinism, then C-States can be disabled in the BIOS/UEFI or the system can be modified to account for the C-State behavior.

Similarly, Enhanced Intel® SpeedStep® Technology should be assessed independently to ensure it does not negatively impact the real-time system behavior. Intel® Turbo Boost Technology can be utilized if the real-time determinism is sufficient when using Enhanced Intel® SpeedStep® Technology. However, the system should not rely on the performance boost to meet a real-time constraint.

With caching behavior, the real-time system designer should understand the different memory types and how they impact the caching behavior. It may be necessary to employ the use of cache and memory ordering instructions such as, CLFLUSH, MFENCE and SFENCE, to add extra determinism to some operations.

The impact of the System Management Interrupts (SMIs) needs to be considered in real-time designs to ensure sufficient determinism is available in the platform. The use of an ACPI enabled operating system helps to cut down the number of SMIs generated, as does disabling USB mouse and keyboard support in the BIOS/UEFI.

Finally, it is highly recommended that the system designer study the datasheet for the selected processor thoroughly so that they understand all of the features available.

# References

[1] Advanced Configuration and Power Interface (ACPI) specification: http://www.acpi.info/spec.htm

[2] Intel® Xeon 5500 Datasheet: http://www.intel.com/p/en_US/products/server/processor/xeon5000/technical-documents

[3] Enhanced Intel® SpeedStep® Technology: http://www.intel.com/support/processors/sb/CS-028855.htm

[4] Intel® Turbo Boost Technology: http://www.intel.com/technology/turboboost

[5] Unified Extensible Firmware Interface: http://www.uefi.org/home/

§

## Author

**Adrian Hoban** is a Software Engineer with the Embedded and Communications Group at Intel Corporation.

## Terminology

| | |
|---|---|
| ACPI | Advanced Configuration and Power Interface |
| BIOS | Basic Input Output System |
| NUMA | Non-Uniform Memory Architecture |
| SMI | System Management Interrupt |
| SMM | System Management Mode |
| TCO | Total Cost of Ownership |
| UEFI | Unified Extensible Firmware Interface |
| USB | Universal Serial Bus |
| WDT | Watchdog Timer |

## Keywords

real-time, determinism, embedded