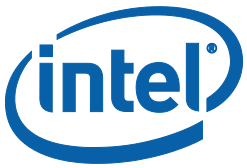


# Intel® Atom™ Processor D400 Series (Single Core)

## Specification Update

***July 2014***

***Revision 009***

Document Number: 322861-009US



By using this document, in addition to any agreements you have with Intel, you accept the terms set forth below.

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.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document 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.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

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

Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. Over time processor numbers will increment based on changes in clock, speed, cache, FSB, or other features, and increments are not intended to represent proportional or quantitative increases in any particular feature. Current roadmap processor number progression is not necessarily representative of future roadmaps. See www.intel.com/products/processor_number for details.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

# Contents

# Revision History

| Revision | Description | Revision Date |
|---|---|---|
| 001 | • Initial Release | December 2009 |
| 002 | • Errata update | February 2010 |
| 003 | • D425 SKU added | June 2010 |
| 004 | • AAR43 and AAR44 Errata added | September 2010 |
| 005 | • AAR45 and AAR46 Errata added | January 2012 |
| 006 | • AAR47 Erratum added | December 2012 |
| 007 | • Added Errata AAR48 and AAR49 | March 2013 |
| 008 | • Updated erratum AAR47 | May 2013 |
| 009 | • Added Errata AAR50<br>• Updated Note in Documentation changes | July 2014 |

§

# Preface

This document is an update to the specifications contained in the documents listed in the following Affected Documents/Related Documents table. It is a compilation of device and document errata and specification clarifications and changes, and is intended for hardware system manufacturers and for software developers of applications, operating system, and tools.

Information types defined in the Nomenclature section of this document are consolidated into this update document and are no longer published in other documents. This document may also contain information that has not been previously published.

## Affected Documents

| Document Title | Document Number/ Location |
|---|---|
| *Intel® Atom™ Processor D400 and D500 Series Datasheet – Volume 1 and 2* | 322844-001, 322845-001 |
| *Intel® Atom™ Processor D400 and D500 Series Thermal Mechanical Design Guidelines* | 322856-001 |

## Related Documents

| Document Title | Document Number/ Location |
|---|---|
| *Intel® 64 and IA-32 Architectures Software Developer's Manual Documentation Changes* | 252046 |
| *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 1: Basic Architecture* | 253665 |
| *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 2A: Instruction Set Reference, A-M* | 253666 |
| *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 2B: Instruction Set Reference, N-Z* | 253667 |
| *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A: System Programming Guide* | 253668 |
| *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B: System Programming Guide* | 253669 |
| *IA-32 Intel® Architectures Optimization Reference Manual* | 248966 |
| *Intel® Processor Identification and the CPUID Instruction Application Note (AP-485)* | 241618 |



## Nomenclature

**Errata** are design defects or errors. These may cause the processor behavior to deviate from published specifications. Hardware and software designed to be used with any given stepping must assume that all errata documented for that stepping are present on all devices.

**S-Spec Number** is a five-digit code used to identify products. Products are differentiated by their unique characteristics, for example, core speed, L2 cache size, package type, etc. as described in the processor identification information table. Read all notes associated with each S-Spec number.

**Specification Changes** are modifications to the current published specifications. These changes will be incorporated in any new release of the specification.

**Specification Clarifications** describe a specification in greater detail or further highlight a specification's impact to a complex design situation. These clarifications will be incorporated in any new release of the specification.

**Documentation Changes** include typos, errors, or omissions from the current published specifications. These will be incorporated in any new release of the specification.

***Note:*** Errata remain in the specification update throughout the product's lifecycle, or until a particular stepping is no longer commercially available. Under these circumstances, errata removed from the specification update are archived and available upon request. Specification changes, specification clarifications and documentation changes are removed from the specification update when the appropriate changes are made to the appropriate product specification or user documentation (datasheets, manuals, etc.).

§

# Identification Information

The Intel® Atom™ Processor D400 Series on 45-nm process stepping can be identified by the register contents in Table 1.

When EAX is initialized to a value of 1, the CPUID instruction returns the Extended Family, Extended Model, Type, Family, Model and Stepping value in the EAX register. Note that the EDX processor signature value after reset is equivalent to the processor signature output value in the EAX register.

**Table 1. Component Identification via Programming Interface**

| Reserved | Extended Family[1] | Extended Model[2] | Reserved | Processor Type[3] | Family Code[4] | Model Number[5] | Stepping ID[6] |
|---|---|---|---|---|---|---|---|
| 31:28 | 27:20 | 19:16 | 15:13 | 12 | 11:8 | 7:4 | 3:0 |
| | 0000000b | 0001b | | 0b | 0110b | 1100b | XXXXb |

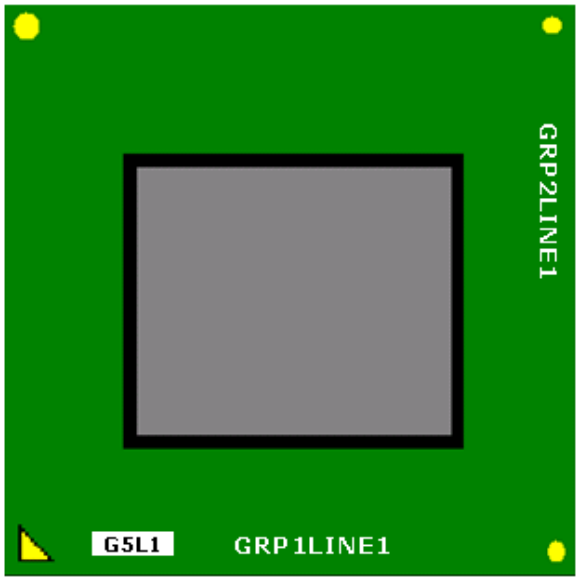
**NOTES:**

1. The Extended Family, bits [27:20] are used in conjunction with the Family Code, specified in bits [11:8], to indicate whether the processor belongs to the Intel386®, Intel486®, Pentium®, Pentium Pro, Pentium 4, or Intel Core processor family.
2. The Extended Model, bits [19:16] in conjunction with the Model Number, specified in bits [7:4], are used to identify the model of the processor within the processor's family.
3. The Processor Type, specified in bits [13:12] indicates whether the processor is an original OEM processor, an OverDrive processor, or a dual processor (capable of being used in a dual processor system).
4. The Family Code corresponds to bits [11:8] of the EDX register after RESET, bits [11:8] of the EAX register after the CPUID instruction is executed with a 1 in the EAX register, and the generation field of the Device ID register accessible through Boundary Scan.
5. The Model Number corresponds to bits [7:4] of the EDX register after RESET, bits [7:4] of the EAX register after the CPUID instruction is executed with a 1 in the EAX register, and the model field of the Device ID register accessible through Boundary Scan.
6. The Stepping ID in bits [3:0] indicates the revision number of that model. See Table 2 for the processor stepping ID number in the CPUID information.

# Component Marking Information

The Intel® Atom™ Processor D400 Series is identified by the following component markings.

**Figure 1. Intel® Atom™ Processor D400 Series Markings**




**SAMPLE MARKING INFORMATION:**
GRP1LINE1: INTEL{M}{C}08{e1}
GRP2LINE1: {FPO} SSPEC PROC#

**Table 2. Identification Table for Desktop Intel® Atom™ Processor D400 Series**

| QDF/ S-SPEC Number | Product Stepping | CPUID | Max Core Speed | Package | Cache Size (KB) | MCU |
|---|---|---|---|---|---|---|
| SLBMH | A0 | 0x000106CAh | 1.66 GHz | Micro-FCBGA8 | 512kB | M01106CA107 |
| SLBXD | A0 | 0x000106CAh | 1.80 GHz | Micro-FCBGA8 | 512kB | M01106CA107 |

§

# Summary Tables of Changes

The following table indicates the Specification Changes, Errata, Specification Clarifications or Documentation Changes, which apply to the listed processor steppings. Intel intends to fix some of the errata in a future stepping of the component, and to account for the other outstanding issues through documentation or Specification Changes as noted. This table uses the following notations:

## Codes Used in Summary Table

## Stepping

| | |
|---|---|
| X: | Erratum, Specification Change or Clarification that applies to this stepping. |
| Blank (No mark): | This erratum is fixed in listed stepping or specification change does not apply to listed stepping. |

## Status

| | |
|---|---|
| **Doc:** | Document change or update that will be implemented. |
| **Plan Fix:** | This erratum may be fixed in a future stepping of the product. |
| **Fixed:** | This erratum has been previously fixed. |
| **No Fix:** | There are no plans to fix this erratum. |

## Row

| | |
|---|---|
| Shaded: | This item is either new or modified from the previous version of the document. |

***Note:*** Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. See http://www.intel.com/products/processor_number for details.

| Number | Stepping | PLAN | ERRATA |
|---|---|---|---|
| | A0 | | |
| AAR1 | X | No Fix | An xTPR Update Transaction Cycle, if Enabled, May be Issued to the FSB after the Processor has Issued a Stop-Grant Special Cycle |
| AAR2 | X | No Fix | The Processor May Report a #TS Instead of a #GP Fault |
| AAR3 | X | No Fix | Writing the Local Vector Table (LVT) when an Interrupt is Pending May Cause an Unexpected Interrupt |
| AAR4 | X | No Fix | MOV To/From Debug Registers Causes Debug Exception |
| AAR5 | X | No Fix | A Write to an APIC Register Sometimes May Appear to Have Not Occurred |
| AAR6 | X | No Fix | Using 2M/4M Pages When A20M# Is Asserted May Result in Incorrect Address Translations |
| AAR7 | X | No Fix | Value for LBR/BTS/BTM will be Incorrect after an Exit from SMM |
| AAR8 | X | No Fix | Incorrect Address Computed For Last Byte of FXSAVE/FXRSTOR Image Leads to Partial Memory Update |
| AAR9 | X | No Fix | A Thermal Interrupt is Not Generated when the Current Temperature is Invalid |
| AAR10 | X | No Fix | Programming the Digital Thermal Sensor (DTS) Threshold May Cause Unexpected Thermal Interrupts |
| AAR11 | X | No Fix | Returning to Real Mode from SMM with EFLAGS.VM Set May Result in Unpredictable System Behavior |
| AAR12 | X | No Fix | Fault on ENTER Instruction May Result in Unexpected Value on Stack Frame |
| AAR13 | X | No Fix | With TF (Trap Flag) Asserted, FP Instruction That Triggers an Unmasked FP Exception May Take Single Step Trap before Retirement of Instruction |
| AAR14 | X | No Fix | An Enabled Debug Breakpoint or Single Step Trap May Be Taken after MOV SS/POP SS Instruction if it is Followed by an Instruction That Signals a Floating Point Exception |
| AAR15 | X | No Fix | Code Segment Limit/Canonical Faults on RSM May be Serviced before Higher Priority Interrupts/Exceptions and May Push the Wrong Address Onto the Stack |
| AAR16 | X | No Fix | BTS(Branch Trace Store) and PEBS(Precise Event Based Sampling) May Update Memory outside the BTS/PEBS Buffer |
| AAR17 | X | No Fix | Single Step Interrupts with Floating Point Exception Pending May Be Mishandled |
| AAR18 | X | No Fix | Unsynchronized Cross-Modifying Code Operations Can Cause Unexpected Instruction Execution Results |
| AAR19 | X | No Fix | A Page Fault May Not be Generated When the PS bit is set to “1” in a PML4E or PDPTE |
| AAR20 | X | No Fix | IO_SMI Indication in SMRAM State Save Area May be Set Incorrectly |

| Number | Stepping | PLAN | ERRATA |
|---|---|---|---|
| | A0 | | |
| AAR21 | X | No Fix | Writes to IA32_DEBUGCTL MSR May Fail when FREEZE_LBRS_ON_PMI is Set |
| AAR22 | X | No Fix | Address Reported by Machine-Check Architecture (MCA) on L2 Cache Errors May be Incorrect |
| AAR23 | X | No Fix | Performance Monitoring Event for Outstanding Bus Requests Ignores AnyThread Bit |
| AAR24 | X | No Fix | Corruption of CS Segment Register During RSM While Transitioning From Real Mode to Protected Mode |
| AAR25 | X | No Fix | GP and Fixed Performance Monitoring Counters With AnyThread Bit Set May Not Accurately Count Only OS or Only USR Events |
| AAR26 | X | No Fix | PMI Request is Not Generated on a Counter Overflow if Its OVF Bit is Already Set in IA32_PERF_GLOBAL_STATUS |
| AAR27 | X | No Fix | Processor May Use an Incorrect Translation if the TLBs Contain Two Different Translations For a Linear Address |
| AAR28 | X | No Fix | PEBS Record not Updated when in Probe Mode |
| AAR29 | X | No Fix | LBR/BTM/BTS Information Immediately After a Transition From Legacy/Compatibility Mode to 64-bit Mode May be Incorrect |
| AAR30 | X | No Fix | Pending x87 FPU Exceptions (#MF) Following STI May Be Serviced Before Higher Priority Interrupts |
| AAR31 | X | No Fix | Benign Exception after a Double Fault May Not Cause a Triple Fault Shutdown |
| AAR32 | X | No Fix | IA32_MC1_STATUS MSR Bit[60] Does Not Reflect Machine Check Error Reporting Enable Correctly |
| AAR33 | X | No Fix | LINT0 Assertion and Deassertion During an Inactive State May Cause Unexpected Operation When APIC is Disabled |
| AAR34 | X | No Fix | IRET under Certain Conditions May Cause an Unexpected Alignment Check Exception |
| AAR35 | X | No Fix | HSYNC and VSYNC Buffers Do Not Meet VESA Rise and Undershoot Specification |
| AAR36 | X | No Fix | Glitch on LVDS Display Interface Clocks and Data Lines May be Observed During Power Up Sequence |
| AAR37 | X | No Fix | CPUID Instruction Returns Incorrect Brand String |
| AAR38 | X | No Fix | IA32_MC2_STATUS [OVERFLOW] Bit is Not Set When Single-Bit Correctable ECC Error Occurs |
| AAR39 | X | No Fix | FP Data Operand Pointer May Be Incorrectly Calculated After an FP Access Which Wraps a 4-Gbyte Boundary in Code That Uses 32-Bit Address Size in 64-bit Mode |
| AAR40 | X | No Fix | Writes to Set IA32_MCG_STATUS.MCIP Will Fail |
| AAR41 | X | No Fix | Synchronous Reset of IA32_MPERF on IA32_APERF Overflow May Not Work |

| Number | Stepping | PLAN | ERRATA |
|---|---|---|---|
| | A0 | | |
| AAR42 | X | No Fix | During a C-State Exit due to a Pending External Interrupt the System May Hang |
| AAR43 | X | No Fix | High Temperature Circuit Marginality Issue May Cause the System to Hang or Auto Reboot |
| AAR44 | X | No Fix | FP Data Operand Pointer May Be Incorrectly Calculated After an FP Access Which Wraps a 64-Kbyte Boundary in 16-Bit Code |
| AAR45 | X | No Fix | Executing LTR in 64-bit Mode May Access Segment Descriptor Before Checking for Null Selector |
| AAR46 | X | No Fix | tREFI Exceeds DDR2 / DDR3 specifications |
| AAR47 | X | No Fix | Complex Conditions Associated With Instruction Page Remapping or Self/Cross-Modifying Code Execution May Lead to Unpredictable System Behavior |
| AAR48 | X | No Fix | REP MOVS/STOS Executing With Fast Strings Enabled and Crossing Page Boundaries with Inconsistent Memory Types May Use an Incorrect Data Size or Lead to Memory-Ordering Violations |
| AAR49 | X | No Fix | Paging Structure Entry May be Used Before Accessed And Dirty Flags Are Updated |
| AAR50 | X | No Fix | Incorrect Translation May be Used After MOV to CR3 |

| Number | Specification Changes |
|---|---|
| | There are no Specification Changes in this revision of the specification update |

| Number | Specification Clarifications |
|---|---|
| | There are no Specification Clarifications in this revision of the specification update |

| Number | Documentation Changes |
|---|---|
| 1 | Updated Note |

§

# Errata

## AAR1 An xTPR Update Transaction Cycle, if Enabled, May be Issued to the FSB after the Processor has Issued a Stop-Grant Special Cycle

**Problem:** According to the FSB (Front Side Bus) protocol specification, no FSB cycles should be issued by the processor once a Stop-Grant special cycle has been issued to the bus. If xTPR update transactions are enabled by clearing the IA32_MISC_ENABLES[bit-23] at the time of Stop-Clock assertion, an xTPR update transaction cycle may be issued to the FSB after the processor has issued a Stop Grant Acknowledge transaction.

**Implication:** When this erratum occurs in systems using C-states C2 (Stop-Grant State) and higher the result could be a system hang.

**Workaround:** BIOS must leave the xTPR update transactions disabled (default).

**Status:** For the steppings affected, see the Summary Tables of Changes.

## AAR2 Processor May Report a #TS Instead of a #GP Fault

**Problem:** During system reset, there is insufficient time for handshake between ICH and GMCH LVDS logic. As a result, timing from panel backlight enable going low to LVDS data going low (TX) and timing from LVDS data going low to panel VCC enable going low (T3) do not match the programmed values. Panel backlight enable (LBKLT_EN), panel Vcc enable (LVDD_EN) and LVDS data lines go low at the same time.

**Implication:** A jump to a busy TSS (Task-State Segment) may cause a #TS (invalid TSS exception) instead of a #GP fault (general protection exception).

**Workaround:** None.

**Status:** For the steppings affected, see the Summary Tables of Changes.



### AAR3 Writing the Local Vector Table (LVT) when an Interrupt is Pending May Cause an Unexpected Interrupt

**Problem:** If a local interrupt is pending when the LVT entry is written, an interrupt may be taken on the new interrupt vector even if the mask bit is set.

**Implication:** An interrupt may immediately be generated with the new vector when a LVT entry is written, even if the new LVT entry has the mask bit set. If there is no Interrupt Service Routine (ISR) set up for that vector the system will GP fault. If the ISR does not do an End of Interrupt (EOI) the bit for the vector is left set in the in-service register and mask all interrupts at the same or lower priority.

**Workaround:** Any vector programmed into an LVT entry must have an ISR associated with it, even if that vector was programmed as masked. This ISR routine must do an EOI to clear any unexpected interrupts that may occur. The ISR associated with the spurious vector does not generate an EOI; therefore the spurious vector should not be used when writing the LVT.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR4 MOV To/From Debug Registers Causes Debug Exception

**Problem:** When in V86 mode, if a MOV instruction is executed to/from a debug registers, a general-protection exception (#GP) should be generated. However, in the case when the general detect enable flag (GD) bit is set, the observed behavior is that a debug exception (#DB) is generated instead.

**Implication:** With debug-register protection enabled (i.e., the GD bit set), when attempting to execute a MOV on debug registers in V86 mode, a debug exception is generated instead of the expected general-protection fault.

**Workaround:** In general, operating systems do not set the GD bit when they are in V86 mode. The GD bit is generally set and used by debuggers. The debug exception handler should check that the exception did not occur in V86 mode before continuing. If the exception did occur in V86 mode, the exception may be directed to the general-protection exception handler.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR5 A Write to an APIC Register Sometimes May Appear to Have Not Occurred

**Problem:** With respect to the retirement of instructions, stores to the uncacheable memory based APIC register space are handled in a non-synchronized way. For example if an instruction that masks the interrupt flag, for example CLI, is executed soon after an uncacheable write to the Task Priority Register (TPR) that lowers the APIC priority, the interrupt masking operation may take effect before the actual priority has been lowered. This may cause interrupts whose priority is lower than the initial TPR, but higher than the final TPR, to not be serviced until the interrupt enabled flag is finally set, i.e. by STI instruction. Interrupts will remain pending and are not lost.

**Implication:** In this example the processor may allow interrupts to be accepted but may delay their service.

**Workaround:** This non-synchronization can be avoided by issuing an APIC register read after the APIC register write. This will force the store to the APIC register before any subsequent instructions are executed. No commercial operating system is known to be impacted by this erratum.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR6 Using 2M/4M Pages When A20M# Is Asserted May Result in Incorrect Address Translations

**Problem:** An external A20M# pin if enabled forces address bit-20 to be masked (forced to zero) to emulates real-address mode address wraparound at 1 megabyte. However, if all of the following conditions are met, address bit-20 may not be masked.

- paging is enabled
- a linear address has bit-20 set
- the address references a large page
- A20M# is enabled

**Implication:** When A20M# is enabled and an address references a large page the resulting translated physical address may be incorrect. This erratum has not been observed with any commercially available operating system.

**Workaround:** Operating systems should not allow A20M# to be enabled if the masking of address bit-20 could be applied to an address that references a large page. A20M# is normally only used with the first megabyte of memory.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR7 Value for LBR/BTS/BTM will be Incorrect after an Exit from SMM

**Problem:** After a return from SMM (System Management Mode), the CPU will incorrectly update the LBR (Last Branch Record) and the BTS (Branch Trace Store), hence rendering their data invalid. The corresponding data if sent out as a BTM on the system bus will also be incorrect.
**Note**: This issue would only occur when one of the 3 above mentioned debug support facilities are used.

**Implication:** The value of the LBR, BTS, and BTM immediately after an RSM operation should not be used.

**Workaround:** None.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR8 Incorrect Address Computed For Last Byte of FXSAVE/FXRSTOR Image Leads to Partial Memory Update

**Problem:** A partial memory state save of the 512-byte FXSAVE image or a partial memory state restore of the FXRSTOR image may occur if a memory address exceeds the 64KB limit while the processor is operating in 16-bit mode or if a memory address exceeds the 4GB limit while the processor is operating in 32-bit mode.

**Implication:** FXSAVE/FXRSTOR will incur a #GP fault due to the memory limit violation as expected but the memory state may be only partially saved or restored.

**Workaround:** Software should avoid memory accesses that wrap around the respective 16-bit and 32-bit mode memory limits.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR9 A Thermal Interrupt is Not Generated when the Current Temperature is Invalid

**Problem:** When the DTS (Digital Thermal Sensor) crosses one of its programmed thresholds it generates an interrupt and logs the event (IA32_THERM_STATUS MSR (019Ch) bits [9,7]). Due to this erratum, if the DTS reaches an invalid temperature (as indicated IA32_THERM_STATUS MSR bit[31]) it does not generate an interrupt even if one of the programmed thresholds is crossed and the corresponding log bits become set.

**Implication:** When the temperature reaches an invalid temperature the CPU does not generate a Thermal interrupt even if a programmed threshold is crossed.

**Workaround:** None.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR10 Programming the Digital Thermal Sensor (DTS) Threshold May Cause Unexpected Thermal Interrupts

**Problem:** Software can enable DTS thermal interrupts by programming the thermal threshold and setting the respective thermal interrupt enable bit. When programming DTS value, the previous DTS threshold may be crossed. This will generate an unexpected thermal interrupt.

**Implication:** Software may observe an unexpected thermal interrupt occur after reprogramming the thermal threshold.

**Workaround:** In the ACPI/OS implement a workaround by temporarily disabling the DTS threshold interrupt before updating the DTS threshold value.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR11 Returning to Real Mode from SMM with EFLAGS.VM Set May Result in Unpredictable System Behavior

**Problem:** Returning back from SMM mode into real mode while EFLAGS.VM is set in SMRAM may result in unpredictable system behavior.

**Implication:** If SMM software changes the value of the EFLAGS.VM in SMRAM, it may result in unpredictable system behavior. Intel has not observed this behavior in commercially available software.

**Workaround:** SMM software should not change the value of EFLAGS.VM in SMRAM.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR12 Fault on ENTER Instruction May Result in Unexpected Value on Stack Frame

**Problem:** The ENTER instruction is used to create a procedure stack frame. Due to this erratum, if execution of the ENTER instruction results in a fault, the dynamic storage area of the resultant stack frame may contain unexpected value (i.e. residual stack data as a result of processing the fault).

**Implication:** Data in the created stack frame may be altered following a fault on the ENTER instruction. Please refer to "Procedure Calls For Block-Structured Languages" in IA-32 Intel® Architecture Software Developer's Manual, Vol. 1, Basic Architecture, for information on the usage of the ENTER instructions. This erratum is not expected to occur in ring 3. Faults are usually processed in ring 0 and stack switch occurs when transferring to ring 0. Intel has not observed this erratum on any commercially available software.

**Workaround:** None.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR13 With TF (Trap Flag) Asserted, FP Instruction That Triggers an Unmasked FP Exception May Take Single Step Trap before Retirement of Instruction

**Problem:** If an FP instruction generates an unmasked exception with the EFLAGS.TF=1, it is possible for external events to occur, including a transition to a lower power state. When resuming from the lower power state, it may be possible to take the single step trap before the execution of the original FP instruction completes.

**Implication:** A Single Step trap is taken when not expected.

**Workaround:** None.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR14 An Enabled Debug Breakpoint or Single Step Trap May Be Taken after MOV SS/POP SS Instruction if it is Followed by an Instruction That Signals a Floating Point Exception

**Problem:** A MOV SS/POP SS instruction should inhibit all interrupts including debug breakpoints until after execution of the following instruction. This is intended to allow the sequential execution of MOV SS/POP SS and MOV [r/e]SP, [r/e]BP instructions without having an invalid stack during interrupt handling. However, an enabled debug breakpoint or single step trap may be taken after MOV SS/POP SS if this instruction is followed by an instruction that signals a floating point exception rather than a MOV [r/e]SP, [r/e]BP instruction. This results in a debug exception being signaled on an unexpected instruction boundary since the MOV SS/POP SS and the following instruction should be executed atomically.

**Implication:** This can result in incorrect signaling of a debug exception and possibly a mismatched Stack Segment and Stack Pointer. If MOV SS/POP SS is not followed by a MOV [r/e]SP, [r/e]BP, there may be a mismatched Stack Segment and Stack Pointer on any exception. Intel has not observed this erratum with any commercially available software, or system.

**Workaround:** As recommended in the IA32 Intel® Architecture Software Developer's Manual, the use of MOV SS/POP SS in conjunction with MOV [r/e]SP, [r/e]BP will avoid the failure since the MOV [r/e]SP, [r/e]BP will not generate a floating point exception. Developers of debug tools should be aware of the potential incorrect debug event signaling created by this erratum.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR15 Code Segment Limit/Canonical Faults on RSM May be Serviced before Higher Priority Interrupts/Exceptions and May Push the Wrong Address Onto the Stack

**Problem:** Normally, when the processor encounters a Segment Limit or Canonical Fault due to code execution, a #GP (General Protection Exception) fault is generated after all higher priority Interrupts and exceptions are serviced. Due to this erratum, if RSM (Resume from System Management Mode) returns to execution flow that results in a Code Segment Limit or Canonical Fault, the #GP fault may be serviced before a higher priority Interrupt or Exception (for example NMI (Non-Maskable Interrupt), Debug break(#DB), Machine Check (#MC), etc.). If the RSM attempts to return to a non-canonical address, the address pushed onto the stack for this #GP fault may not match the non-canonical address that caused the fault.

**Implication:** Operating systems may observe a #GP fault being serviced before higher priority Interrupts and Exceptions. Intel has not observed this erratum on any commercially available software.

**Workaround:** None.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR16 BTS(Branch Trace Store) and PEBS(Precise Event Based Sampling) May Update Memory outside the BTS/PEBS Buffer

**Problem:** If the BTS/PEBS buffer is defined such that:

- The difference between BTS/PEBS buffer base and BTS/PEBS absolute maximum is not an integer multiple of the corresponding record sizes
- BTS/PEBS absolute maximum is less than a record size from the end of the virtual address space
- The record that would cross BTS/PEBS absolute maximum will also continue past the end of the virtual address space

A BTS/PEBS record can be written that will wrap at the 4G boundary (IA32) or $2^{64}$ boundary (EM64T mode), and write memory outside of the BTS/PEBS buffer.

**Implication:** Software that uses BTS/PEBS near the 4G boundary (IA32) or $2^{64}$ boundary (EM64T mode), and defines the buffer such that it does not hold an integer multiple of records can update memory outside the BTS/PEBS buffer.

**Workaround:** Define BTS/PEBS buffer such that BTS/PEBS absolute maximum minus BTS/PEBS buffer base is integer multiple of the corresponding record sizes as recommended in the IA-32 Intel® Architecture Software Developer's Manual, Volume 3.

**Status:** For the steppings affected, see the Summary Tables of Changes.



### AAR17 Single Step Interrupts with Floating Point Exception Pending May Be Mishandled

**Problem:** In certain circumstances, when a floating point exception (#MF) is pending during single-step execution, processing of the single-step debug exception (#DB) may be mishandled.

**Implication:** When this erratum occurs, #DB is incorrectly handled as follows:

- #DB is signaled before the pending higher priority #MF (Interrupt 16)
- #DB is generated twice on the same instruction

**Workaround:** None.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR18 Unsynchronized Cross-Modifying Code Operations Can Cause Unexpected Instruction Execution Results

**Problem:** The act of one processor, or system bus master, writing data into a currently executing code segment of a second processor with the intent of having the second processor execute that data as code is called cross-modifying code (XMC). XMC that does not force the second processor to execute a synchronizing instruction, prior to execution of the new code, is called unsynchronized XMC. Software using unsynchronized XMC to modify the instruction byte stream of a processor can see unexpected or unpredictable execution behavior from the processor that is executing the modified code.

**Implication:** In this case, the phrase "unexpected or unpredictable execution behavior" encompasses the generation of most of the exceptions listed in the Intel Architecture Software Developer's Manual Volume 3A: System Programming Guide, including a General Protection Fault (#GP) or other unexpected behaviors.

**Workaround:** In order to avoid this erratum, programmers should use the XMC synchronization algorithm as detailed in the *Intel Architecture Software Developer's Manual Volume 3A: System Programming Guide*, Section: Handling Self- and Cross-Modifying Code.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR19 A Page Fault May Not be Generated When the PS bit is set to "1" in a PML4E or PDPTE

**Problem:** On processors supporting Intel® 64 architecture, the PS bit (Page Size, bit 7) is reserved in PML4Es and PDPTEs. If the translation of the linear address of a memory access encounters a PML4E or a PDPTE with PS set to 1, a page fault should occur. Due to this erratum, PS of such an entry is ignored and no page fault will occur due to its being set.

**Implication:** Software may not operate properly if it relies on the processor to deliver page faults when reserved bits are set in paging-structure entries.

**Workaround:** Software should not set bit 7 in any PML4E or PDPTE that has Present Bit (Bit 0) set to "1".

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR20 IO_SMI Indication in SMRAM State Save Area May be Set Incorrectly

**Problem:** The IO_SMI bit in SMRAM's location 7FA4H is set to "1" by the CPU to indicate a System Management Interrupt (SMI) occurred as the result of executing an instruction that reads from an I/O port. Due to this erratum, the IO_SMI bit may be incorrectly set by:

- A SMI that is pending while a lower priority event is executing
- A REP I/O read
- A I/O read that redirects to MWAIT

**Implication:** SMM handlers may get false IO_SMI indication.

**Workaround:** The SMM handler has to evaluate the saved context to determine if the SMI was triggered by an instruction that read from an I/O port. The SMM handler must not restart an I/O instruction if the platform has not been configured to generate a synchronous SMI for the recorded I/O port address.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR21 Writes to IA32_DEBUGCTL MSR May Fail when FREEZE_LBRS_ON_PMI is Set

**Problem:** When the FREEZE_LBRS_ON_PMI, IA32_DEBUGCTL MSR (1D9H) bit [11], is set, future writes to IA32_DEBUGCTL MSR may not occur in certain rare corner cases. Writes to this register by software or during certain processor operations are affected.

**Implication:** Under certain circumstances, the IA32_DEBUGCTL MSR value may not be updated properly and will retain the old value. Intel has not observed this erratum with any commercially available software.

**Workaround:** Do not set the FREEZE_LBRS_ON_PMI bit of IA32_DEBUGCTL MSR.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR22 Address Reported by Machine-Check Architecture (MCA) on L2 Cache Errors May be Incorrect

**Problem:** When an L2 Cache error occurs (Error code 0x010A or 0x110A reported in IA32_MCi_STATUS MSR bits [15:0]), the address is logged in the MCA address register (IA32_MCi_ADDR MSR). Under some scenarios, the address reported may be incorrect.

**Implication:** Software should not rely on the value reported in IA32_MCi_ADDR MSR for L2 Cache errors.

**Workaround:** None.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR23 Performance Monitoring Event for Outstanding Bus Requests Ignores AnyThread Bit

**Problem:** The Performance Monitoring Event of Outstanding Bus Requests will ignore the AnyThread bit (IA32_PERFEVTSEL0 MSR (186H)/ IA32_PERFEVTSEL1 MSR (187H) bit [21]) and will instead always count all transactions across all logical processors, even when AnyThread is clear.

**Implication:** The performance monitor count may be incorrect when counting only the current logical processor's outstanding bus requests on a processor supporting Hyper-Threading Technology.

**Workaround:** None identified.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR24 Corruption of CS Segment Register During RSM While Transitioning From Real Mode to Protected Mode

**Problem:** During the transition from real mode to protected mode, if an SMI (System Management Interrupt) occurs between the MOV to CR0 that sets PE (Protection Enable, bit 0) and the first far JMP, the subsequent RSM (Resume from System Management Mode) may cause the lower two bits of CS segment register to be corrupted.

**Implication:** The corruption of the bottom two bits of the CS segment register will have no impact unless software explicitly examines the CS segment register between enabling protected mode and the first far JMP. *Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 3A: System Programming Guide, Part 1*, in the section titled "Switching to Protected Mode" recommends the far JMP immediately follows the write to CR0 to enable protected mode. Intel has not observed this erratum with any commercially available software.

**Workaround:** None identified.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR25 GP and Fixed Performance Monitoring Counters With AnyThread Bit Set May Not Accurately Count Only OS or Only USR Events

**Problem:** A fixed or GP (general purpose) performance counter with the AnyThread bit (IA32_FIXED_CTR_CTRL_MSR (38DH) bit[2] for IA32_FIXED_CTR0, bit[6] for IA32_FIXED_CTR1, bit [10] for IA32_FIXED_CTR2; IA32_PERFEVTSEL0 MSR (186H)/ IA32_PERFEVTSEL1 MSR (187H) bit [21]) set may not count correctly when counting only OS (ring 0) events or only USR (ring>0) events. The counters will count correctly if they are counting both OS and USR events or if the AnyThread bit is clear.

**Implication:** A performance monitor counter may be incorrect when it is counting for all logical processors on that core and not counting at all privilege levels. This erratum will only occur on processors supporting multiple logical processors per core.

**Workaround:** None identified.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR26 PMI Request is Not Generated on a Counter Overflow if its OVF Bit is Already Set in IA32_PERF_GLOBAL_STATUS

**Problem:** If a performance counter overflows and software does not clear the corresponding OVF (overflow) bit in IA32_PERF_GLOBAL_STATUS MSR (38Eh) then future overflows of that counter will not trigger PMI (Performance Monitoring Interrupt) requests.

**Implication:** If software does not clear the OVF bit corresponding to a performance counter then future counter overflows may not cause PMI requests.

**Workaround:** Software should clear the IA32_PERF_GLOBAL_STATUS.OVF bit in the PMI handler.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR27 Processor May Use an Incorrect Translation if the TLBs Contain Two Different Translations For a Linear Address

**Problem:** The TLBs may contain both ordinary and large-page translations for a 4-KByte range of linear addresses. This may occur if software modifies a PDE (page-directory entry) that is marked present to set the PS bit (this changes the page size used for the address range). If the two translations differ with respect to page frame, permissions, or memory type, the processor may use a page frame, permissions, or memory type that corresponds to neither translation.

**Implication:** Due to this erratum, software may not function properly if it sets the PS flag in a PDE and also changes the page frame, permissions, or memory type for the linear addresses mapped through that PDE.

**Workaround:** Software can avoid this problem by ensuring that the TLBs never contain both ordinary and large-page translations for a linear address that differ with respect to page frame, permissions, or memory type.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR28 PEBS Record not Updated When in Probe Mode

**Problem:** When a performance monitoring counter is configured for PEBS (Precise Event Based Sampling), overflows of the counter can result in storage of a PEBS record in the PEBS buffer. Due to this erratum, if the overflow occurs during probe mode, it may be ignored and a new PEBS record may not be added to the PEBS buffer.

**Implication:** Due to this erratum, the PEBS buffer may not be updated by overflows that occur during probe mode.

**Workaround:** None.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR29 LBR/BTM/BTS Information Immediately After a Transition From Legacy/Compatibility Mode to 64-bit Mode May be Incorrect

**Problem:** If a transition from legacy/compatibility mode to 64-bit mode occurs and another branch event occurs before the first instruction executes (for example an external interrupt or trap) then any FROM address recorded by LBR (Last Branch Record), BTM (Branch Trace Message) or BTS (Branch Trace Store) on that second event may incorrectly report the upper 32-bits as zero.

**Implication:** Due to this erratum, bits 63:32 of the 'FROM' value for LBR/BTM/BTS may be improperly zeroed after a transition to 64 bite mode when the RIP (Instruction Pointer Register) is greater than 4 Gigabyte.

**Workaround:** None identified. This erratum may be detected by a 'FROM' address having its upper 32-bits zero but its lower 32-bits matching the previous 'TO' address recorded.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR30 Pending x87 FPU Exceptions (#MF) Following STI May Be Serviced Before Higher Priority Interrupts

**Problem:** Interrupts that are pending prior to the execution of the STI (Set Interrupt Flag) instruction are normally serviced immediately after the instruction following the STI. An exception to this is if the following instruction triggers a #MF. In this situation, the interrupt should be serviced before the #MF. Because of this erratum, if following STI, an instruction that triggers a #MF is executed while STPCLK#, Enhanced Intel SpeedStep Technology transitions or Thermal Monitor events occur, the pending #MF may be serviced before higher priority interrupts.

**Implication:** Software may observe #MF being serviced before higher priority interrupts.

**Workaround:** None identified.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR31 Benign Exception after a Double Fault May Not Cause a Triple Fault Shutdown

**Problem:** According to the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A, "Exception and Interrupt Reference",* if another exception occurs while attempting to call the double-fault handler, the processor enters shutdown mode. Due to this erratum, any benign faults while attempting to call double-fault handler will not cause a shutdown. However Contributory Exceptions and Page Faults will continue to cause a triple fault shutdown.

**Implication:** If a benign exception occurs while attempting to call the double-fault handler, the processor may hang or may handle the benign exception. Intel has not observed this erratum with any commercially available software.

**Workaround:** None identified.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR32 IA32_MC1_STATUS MSR Bit[60] Does Not Reflect Machine Check Error Reporting Enable Correctly

**Problem:** IA32_MC1_STATUS MSR (405H) bit[60] (EN- Error Enabled) is supposed to indicate whether the enable bit in the IA32_MC1_CTL MSR (404H) was set at the time of the last update to the IA32_MC1_STATUS MSR. Due to this erratum, IA32_MC1_STATUS MSR bit[60] instead reports the current value of the IA32_MC1_CTL MSR enable bit.

**Implication:** IA32_MC1_STATUS MSR bit [60] may not reflect the correct state of the enable bit in the IA32_MC1_CTL MSR at the time of the last update.

**Workaround:** None identified.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR33 LINT0 Assertion and De-assertion During an Inactive State May Cause Unexpected Operation When APIC is Disabled

**Problem:** An interrupt delivered via LINT0 pins when the APIC is hardware disabled (IA32_APIC_BASE MSR (1BH) bit [11] is cleared) will usually keep the pin asserted until after the interrupt is acknowledged. However, if LINT0 is asserted and then de-asserted before the interrupt is acknowledged and both of the following are true:

- The APIC is hardware disabled (IA32_APIC_BASE MSR bit [11] is clear) and
- The processor is in an inactive state that was requested by MWAIT, I/O redirection, VM-entry or RSM,

then the processor may operate incorrectly

**Implication:** Due to this erratum, the processor may run unexpected code and/or generate an unexpected exception. Intel has not observed this erratum with any commercially available software.

**Workaround:** If LINT0 is used, it is recommended to either leave the APIC enabled (IA32_APIC_BASE MSR bit [11] set to 1) or do not use MWAIT, I/O redirection, VM-entry or RSM to enter an inactive state.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR34 IRET under Certain Conditions May Cause an Unexpected Alignment Check Exception

**Problem:** In IA-32e mode, it is possible to get an Alignment Check Exception (#AC) on the IRET instruction even though alignment checks were disabled at the start of the IRET. This can only occur if the IRET instruction is returning from CPL3 code to CPL3 code. IRETs from CPL0/1/2 are not affected. This erratum can occur if the EFLAGS value on the stack has the AC flag set, and the interrupt handler's stack is misaligned. In IA-32e mode, RSP is aligned to a 16-byte boundary before pushing the stack frame.

**Implication:** In IA-32e mode, under the conditions given above, an IRET can get a #AC even if alignment checks are disabled at the start of the IRET. This erratum can only be observed with a software generated stack frame.

**Workaround:** Software should not generate misaligned stack frames for use with IRET.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR35 HSYNC/VSYNC Buffer Does Not Meet VESA Rise & Undershoot Specification

**Problem:** Both HSYNC (horizontal Sync) and VSYNC (vertical sync) signals are violating VESA (Video Electronics Standards Association) specification due to non-monotonic slow rise time on both signals.

**Implication:** HSYNC and VSYNC signals may not meet VESA specification.

**Workaround:** Insert a buffer in the HSYNC/VSYNC signal path before the video connector. Refer to Platform Design Guide and Customer Reference Board (CRB) schematic for reference.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR36 Glitch on LVDS Display Interface Clocks and Data Lines May be Observed during Power-Up Sequences

**Problem:** During power up sequence (transition to S0 state from G3, S3, S4 or S5 states) when LVDS (Low Voltage Differential Signal) power supply (1.8V source) ramps up, a glitch on LVDS clocks (LVD_A_CLKP, LVD_A_CLKN) and data lines (LVD_A_DAPAP[2:0], LVD_A_DATAN[2:0]) may be observed.

**Implication:** Due to this erratum, a glitch may be seen during power up sequence. The glitch is not seen once the LVDS power supply is stable.

**Workaround:** None identified.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR37 CPUID Instruction Returns Incorrect Brand String

**Problem:** When the CPUID instructions is executed with EAX = 80000002H, 80000003H and 80000004H, the returned brand string may be incorrect. The model number in the brand string may be prefixed with a “K” instead of the expected “D”.

**Implication:** When this erratum occurs, the processor will report an incorrect model number in the brand string.

**Workaround:** It is possible for the BIOS to contain a workaround for this erratum.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR38 IA32_MC2_STATUS [OVERFLOW] Bit is Not Set When Single-Bit Correctable ECC Error Occurs

**Problem:** The OVERFLOW bit should be set if the VAL bit (IA32_MC2_STATUS (409H) bit [63]) is set when a new error occurs. Due to this erratum, the OVERFLOW bit (IA32_MC2_STATUS (409H) bit [62]) is only set when a prior uncorrected error (as indicated by the UC bit (IA32_MC2_STATUS (409H) bit [61])) is present at the time the second error occurs.

**Implication:** Any L2 correctable error will not set the IA32_MC2_STATUS.OVERFLOW bit when overwriting a prior L2 correctable error.

**Workaround:** The frequency of occurrence of this problem is reduced greatly if an operating system regularly polls and clears the machine check banks as this reduces the likelihood of an overflow condition.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR39 FP Data Operand Pointer May Be Incorrectly Calculated After an FP Access Which Wraps a 4-Gbyte Boundary in Code That Uses 32-Bit Address Size in 64-bit Mode

**Problem:** The FP (Floating Point) Data Operand Pointer is the effective address of the operand associated with the last non-control FP instruction executed by the processor. If an 80-bit FP access (load or store) uses a 32-bit address size in 64-bit mode and the memory access wraps a 4-Gbyte boundary and the FP environment is subsequently saved, the value contained in the FP Data Operand Pointer may be incorrect.

**Implication:** Due to this erratum, the FP Data Operand Pointer may be incorrect. Wrapping an 80-bit FP load around a 4-Gbyte boundary in this way is not a normal programming practice. Intel has not observed this erratum with any commercially available software.

**Workaround:** If the FP Data Operand Pointer is used in a 64-bit operating system which may run code accessing 32-bit addresses, care must be taken to ensure that no 80-bit FP accesses are wrapped around a 4-Gbyte boundary.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR40 Writes to Set IA32_MCG_STATUS.MCIP Will Fail

**Problem:** An MSR write that attempts to set the IA32_MCG_STATUS MSR (17AH) MCIP (machine check in progress) bit [2] will fail (e.g. #GP fault on WRMSR) instead of setting the bit. An MSR write that specifies 0 for the MCIP bit will function correctly.

**Implication:** Due to this erratum, software writes to set this bit will not succeed and may cause an unexpected General Protection fault.

**Workaround:** None identified.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR41 Synchronous Reset of IA32_MPERF on IA32_APERF Overflow May Not Work

**Problem:** When either the IA32_MPERF or IA32_APERF MSR (E7H, E8H) increments to its maximum value of 0xFFFF_FFFF_FFFF_FFFF, both MSRs are supposed to synchronously reset to 0x0 on the next clock. Due to this erratum, IA32_MPERF may not be reset when IA32_APERF overflows. Instead, IA32_MPERF may continue to increment without being reset.

**Implication:** Due to this erratum, software cannot rely on synchronous reset of the IA32_MPERF register. The typical usage of IA32_MPERF/IA32_APERF is to initialize them with a value of 0; in this case the overflow of the counter wouldn't happen for over 10 years.

**Workaround:** None identified.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAV42 During a C-state Exit due to a Pending External Interrupt the System May Hang

**Problem:** Under a precise set of conditions, a processor waking from a C-state due to a pending external interrupt may not complete the exiting process and the system may hang.

**Implication:** Due to this erratum, the system may hang.

**Workaround:** It is possible for the BIOS to contain a workaround for this erratum.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR43 High Temperature Circuit Marginality Issue May Cause the System to Hang or Auto Reboot

**Problem:** A subset of processors may experience circuit marginality issues when operating at high temperature. Due to this erratum a system hang may occur or the processor may proceed to reboot.

**Implication:** Due to this erratum, the system may hang or auto reboot.

**Workaround:** A BIOS workaround has been identified. Please refer to memory reference code version 1.12 or later.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR44 FP Data Operand Pointer May Be Incorrectly Calculated After an FP Access Which Wraps a 64-Kbyte Boundary in 16-Bit Code

**Problem:** The FP (Floating Point) Data Operand Pointer is the effective address of the operand associated with the last non-control FP instruction executed by the processor. If an 80-bit FP access (load or store) occurs in a 16-bit mode other than protected mode (in which case the access will produce a segment limit violation), the memory access wraps a 64-Kbyte boundary, and the FP environment is subsequently saved, the value contained in the FP Data Operand Pointer may be incorrect.

**Implication:** Due to this erratum, the FP Data Operand Pointer may be incorrect. Wrapping an 80-bit FP load around a segment boundary in this way is not a normal programming practice. Intel has not observed this erratum with any commercially available software.

**Workaround:** If the FP Data Operand Pointer is used in an operating system which may run 16-bit FP code, care must be taken to ensure that no 80-bit FP accesses are wrapped around a 64-Kbyte boundary.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR45 Executing LTR in 64-bit Mode May Access Segment Descriptor Before Checking for Null Selector

**Problem:** When executing the LTR instruction with a null segment selector, #GP(0) should be delivered without accessing the memory in the GDT (Global Descriptor Table). Due to this erratum, such an execution of the LTR instruction in 64-bit mode may access that memory. Side effects of this memory access (e.g. a page fault or EPT violation) that occur may prevent the #GP(0) from being delivered.

**Implication:** Executing the LTR instruction with a null segment selector may incorrectly access the GDT. Intel has not observed this erratum with any commercially available system.

**Workaround:** None identified.

**Status:** For the steppings affected, see the Summary Tables of Changes

### AAR46 tREFI Exceeds DDR2 / DDR3 Specifications

**Problem:** tREFI (Average DRAM Refresh Interval) is 7.825us which exceeds the 7.8us stated in the DDR2/DDR3 specification. Due to this erratum, the processor will take more time to refresh rows. (example: 64.1ms instead of 64.0ms to issue 8192 refreshes)

**Implication:** tREFI specification is exceeded. Intel has not observed any other issues with DRAM refresh due to this erratum.

**Workaround:** None identified.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR47 Complex Conditions Associated With Instruction Page Remapping or Self/Cross-Modifying Code Execution May Lead to Unpredictable System Behavior

**Problem:** Under a complex set of internal conditions, instruction page remapping, or self/cross modifying code events may lead to unpredictable system behavior.

**Implication:** Due to this Erratum, unpredictable system behavior may be observed.

**Workaround:** None identified.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR48 REP MOVS/STOS Executing With Fast Strings Enabled and Crossing Page Boundaries with Inconsistent Memory Types May Use an Incorrect Data Size or Lead to Memory-Ordering Violations

**Problem:** Under the conditions described in the Software Developers Manual section “Fast String Operation,” the processor performs REP MOVS or REP STOS as fast strings. Due to this erratum, fast string REP MOVS/REP STOS instructions that cross page boundaries from WB/WC memory types to UC/WP/WT memory types, may start using an incorrect data size or may observe memory ordering violations.

**Implication:** Upon crossing the page boundary, the following may occur, dependent on the new page memory type:

- UC: The data size of each read and write may be different than the original data size.
- WP: The data size of each read and write may be different than the original data size and there may be a memory ordering violation.
- WT: There may be a memory ordering violation.

**Workaround:** Software should avoid crossing page boundaries from WB or WC memory type to UC, WP or WT memory type within a single REP MOVS or REP STOS instruction that will execute with fast strings enabled.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR49 Paging Structure Entry May be Used Before Accessed And Dirty Flags Are Updated

**Problem:** If software modifies a paging structure entry while the processor is using the entry for linear address translation, the processor may erroneously use the old value of the entry to form a translation in a TLB (or an entry in a paging structure cache) and then update the entry's new value to set the accessed flag or dirty flag. This will occur only if both the old and new values of the entry result in valid translations.

**Implication:** Incorrect behavior may occur with algorithms that atomically check that the accessed flag or the dirty flag of a paging structure entry is clear and modify other parts of that paging structure entry in a manner that results in a different valid translation.

**Workaround:** Affected algorithms must ensure that appropriate TLB invalidation is done before assuming that future accesses do not use translations based on the old value of the paging structure entry.

**Status:** For the steppings affected, see the Summary Tables of Changes.

### AAR50. Incorrect Translation May be Used After MOV to CR3

**Problem:** If MOV to CR3 modifies CR3[35:32], the processor may subsequently use incorrect translations for some linear addresses. A MOV to CR3 that modifies CR3[35:32] can occur only in 64-bit mode on a system with memory at addresses above 4 GBs. Additionally, this erratum cannot occur if global pages have never been enabled (CR4.PGE has never been set) on either logical processor on the core. If MOV to CR3 modifies CR3[35:32], the processor may subsequently use incorrect translations for some linear addresses. A MOV to CR3 that modifies CR3[35:32] can occur only in 64-bit mode on a system with memory at addresses above 4 GBs. Additionally, this erratum cannot occur if global pages have never been enabled (CR4.PGE has never been set) on either logical processor on the core.

**Implication:** When this erratum occurs, the processor may use incorrect translations. This may result in unexpected faults or other unpredictable system behavior.

**Workaround:** It is possible for the BIOS to contain a workaround for this erratum.

**Status:** For the steppings affected, see the Summary Tables of Changes.

§

# Specification Changes

There are no specification changes in this revision of the specification update.

§

# Specification Clarifications

There are no specification clarifications in this revision of the specification update.

§

# Documentation Changes

There are no new Documentation Changes in this Specification Update revision.

***Note:*** Documentation changes for Intel® 64 and IA-32 Architecture Software Developer's Manual volumes 1, 2A, 2B, 3A, and 3B will be posted in a separate document, Intel® 64 and IA-32 Architecture Software Developer's Manual Documentation Changes. Follow the link below to become familiar with this file.

http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html



§