Intel® Celeron Processor E3000$^{\Delta}$ Series

Specification Update

— on 45 nm Process in the 775-land LGA Package

January 2013

Notice: The Intel® Celeron® processor may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are documented in this Specification Update.

Document Number: 322568 -006

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.

UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

Intel products are not intended for use in medical, life saving, or life sustaining applications.

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.

Enabling Execute Disable Bit functionality requires a PC with a processor with Execute Disable Bit capability and a supporting operating system. Check with your PC manufacturer on whether your system delivers Execute Disable Bit functionality.

Φ Intel® 64 requires a computer system with a processor, chipset, BIOS, operating system, device drivers, and applications enabled for Intel 64. Processor will not operate (including 32-bit operation) without an Intel 64-enabled BIOS. Performance will vary depending on your hardware and software configurations. See http://www.intel.com/technology/intel64/ for more information including details on which processors support Intel 64, or consult with your system vendor for more information.

Δ 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.

Intel, the Intel logo, Celeron, Pentium, Xeon, Intel SpeedStep, Intel Core, and Core Inside are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

*Other names and brands may be claimed as the property of others.

Contents

§

Revision History

Revision Number	Description	Date
001	Initial release of the *Intel® Celeron Dual-Core Processor E3000$^{\Delta}$ Series* Specification Update	August 31, 2009
002	Added Intel® Celeron E3400 Product details	January 17, 2010
003	• Added Errata AAQ64	March 16th, 2010
004	• Added Intel® Celeron E3500 Product details • Removed Item Numbering section	August 30th, 2010
005	• Added Erratum AAQ65 • Updated Microcode Update Table	December 8th 2010
006	• Added Document Change	January 16th, 2013

§

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
Intel® Celeron® processor E3000Δ Series Datasheet	322567

Related Documents

Document Title	Document Location
Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 1: Basic Architecture	http://www.intel.com/products/processor/manuals/index.htm
Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 2A: Instruction Set Reference Manual A–M	
Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 2B: Instruction Set Reference Manual, N–Z	
Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 3A: System Programming Guide	
Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 3B: System Programming Guide	

Nomenclature

S-Spec Number is a five-digit code used to identify products. Products are differentiated by their unique characteristics (e.g., core speed, L2 cache size, package type, etc.) as described in the processor identification information table. Care should be taken to read all notes associated with each S-Spec number

QDF Number is a several digit code that is used to distinguish between engineering samples. These processors are used for qualification and early design validation. The functionality of these parts can range from mechanical only to fully functional. The NDA specification update has a processor identification information table that lists these QDF numbers and the corresponding product sample details.

Errata are design defects or errors. Errata may cause the processor's 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.

Specification Changes are modifications to the current published specifications. These changes will be incorporated in the next release of the specifications.

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 the next release of the specifications.

Documentation Changes include typos, errors, or omissions from the current published specifications. These changes will be incorporated in the next release of the specifications.

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.).

§

Summary Tables of Changes

The following table indicates the Specification Changes, Errata, Specification Clarifications or Documentation Changes, which apply to the listed MCH 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.
(No mark) or (Blank Box):	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.

NO#	R0	Plan	ERRATA
AAQ1	X	No Fix	EFLAGS Discrepancy on Page Faults after a Translation Change
AAQ2	X	No Fix	INVLPG Operation for Large (2M/4M) Pages May be Incomplete under Certain Conditions
AAQ3	X	No Fix	Store to WT Memory Data May be Seen in Wrong Order by Two Subsequent Loads
AAQ4	X	No Fix	Non-Temporal Data Store May be Observed in Wrong Program Order
AAQ5	X	No Fix	Page Access Bit May be Set Prior to Signaling a Code Segment Limit Fault
AAQ6	X	No Fix	Updating Code Page Directory Attributes without TLB Invalidation May Result in Improper Handling of Code #PF
AAQ7	X	No Fix	Storage of PEBS Record Delayed Following Execution of MOV SS or STI
AAQ8	X	No Fix	Performance Monitoring Event FP_MMX_TRANS_TO_MMX May Not Count Some Transitions
AAQ9	X	No Fix	A REP STOS/MOVS to a MONITOR/MWAIT Address Range May Prevent Triggering of the Monitoring Hardware
AAQ10	X	No Fix	Performance Monitoring Event MISALIGN_MEM_REF May Over Count
AAQ11	X	No Fix	The Processor May Report a #TS Instead of a #GP Fault
AAQ12	X	No Fix	Code Segment limit violation may occur on 4 Gigabyte limit check
AAQ13	X	No Fix	A Write to an APIC Register Sometimes May Appear to Have Not Occurred
AAQ14	X	No Fix	Last Branch Records (LBR) Updates May be Incorrect after a Task Switch
AAQ15	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
AAQ16	X	No Fix	Upper 32 bits of ‘From’ Address Reported through BTMs or BTSs May be Incorrect
AAQ17	X	No Fix	Address Reported by Machine-Check Architecture (MCA) on Single-bit L2 ECC Errors May be Incorrect
AAQ18	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
AAQ19	X	No Fix	Store Ordering May be Incorrect between WC and WP Memory Types
AAQ20	X	No Fix	EFLAGS, CR0, CR4 and the EXF4 Signal May be Incorrect after Shutdown
AAQ21	X	No Fix	Premature Execution of a Load Operation Prior to Exception Handler Invocation
AAQ22	X	No Fix	Performance Monitoring Events for Retired Instructions (C0H) May Not Be Accurate
AAQ23	X	No Fix	Returning to Real Mode from SMM with EFLAGS.VM Set May Result in Unpredictable System Behavior
AAQ24	X	No Fix	CMPSB, LODSB, or SCASB in 64-bit Mode with Count Greater or Equal to 2^{48} May Terminate Early
AAQ25	X	No Fix	Writing the Local Vector Table (LVT) when an Interrupt is Pending May Cause an Unexpected Interrupt
AAQ26	X	No Fix	Pending x87 FPU Exceptions (#MF) Following STI May Be Serviced Before Higher Priority Interrupts

NO#	R0	Plan	ERRATA
AAQ27	X	No Fix	VERW/VERR/LSL/LAR Instructions May Unexpectedly Update the Last Exception Record (LER) MSR
AAQ28	X	No Fix	INIT Does Not Clear Global Entries in the TLB
AAQ29	X	No Fix	Split Locked Stores May not Trigger the Monitoring Hardware
AAQ30	X	No Fix	Programming the Digital Thermal Sensor (DTS) Threshold May Cause Unexpected Thermal Interrupts
AAQ31	X	No Fix	Writing Shared Unaligned Data that Crosses a Cache Line without Proper Semaphores or Barriers May Expose a Memory Ordering Issue
AAQ32	X	No Fix	General Protection (#GP) Fault May Not Be Signaled on Data Segment Limit Violation above 4-G Limit
AAQ33	X	No Fix	An Asynchronous MCE During a Far Transfer May Corrupt ESP
AAQ34	X	Plan Fix	CPUID Reports Architectural Performance Monitoring Version 2 is Supported, When Only Version 1 Capabilities are Available
AAQ35	X	No Fix	B0-B3 Bits in DR6 For Non-Enabled Breakpoints May be Incorrectly Set
AAQ36	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
AAQ37	X	No Fix	Instruction Fetch May Cause a Livelock During Snoops of the L1 Data Cache
AAQ38	X	No Fix	Use of Memory Aliasing with Inconsistent Memory Type may Cause a System Hang or a Machine Check Exception
AAQ39	X	No Fix	A WB Store Following a REP STOS/MOVS or FXSAVE May Lead to Memory-Ordering Violations
AAQ40	X	No Fix	Using Memory Type Aliasing with cacheable and WC Memory Types May Lead to Memory Ordering Violations
AAQ41	X	No Fix	Benign Exception after a Double Fault May Not Cause a Triple Fault Shutdown
AAQ42	X	Plan Fix	Short Nested Loops That Span Multiple 16-Byte Boundaries May Cause a Machine Check Exception or a System Hang
AAQ43	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
AAQ44	X	No Fix	LER MSRs May be Incorrectly Updated
AAQ45	X	No Fix	IA32_MC1_STATUS MSR Bit[60] Does Not Reflect Machine Check Error Reporting Enable Correctly
AAQ46	X	No Fix	IRET under Certain Conditions May Cause an Unexpected Alignment Check Exception
AAQ47	X	Plan Fix	PSI# Signal Asserted During Reset
AAQ48	X	No Fix	Thermal Interrupts are Dropped During and While Exiting Intel® Deep Power-Down State
AAQ49	X	No Fix	Processor May Hold-off / Delay a PECI Transaction Longer than Specified by the PECI Protocol
AAQ50	X	No Fix	XRSTOR Instruction May Cause Extra Memory Reads

NO#	R0	Plan	ERRATA
AAQ51	X	Plan Fix	CPUID Instruction May Return Incorrect Brand String
AAQ52	X	No Fix	When Intel® Deep Power-Down State is Being Used, IA32_FIXED_CTR2 May Return Incorrect Cycle Counts
AAQ53	X	No Fix	Enabling PECI via the PECI_CTL MSR incorrectly writes CPUID_FEATURE_MASK1 MSR
AAQ54	X	No Fix	INIT Incorrectly Resets IA32_LSTAR MSR
AAQ55	X	No Fix	Corruption of CS Segment Register During RSM While Transitioning From Real Mode to Protected Mode
AAQ56	X	No Fix	LBR, BTS, BTM May Report a Wrong Address when an Exception/Interrupt Occurs in 64-bit Mode
AAQ57	X	No Fix	The XRSTOR Instruction May Fail to Cause a General-Protection Exception
AAQ58	X	No Fix	The XSAVE Instruction May Erroneously Set Reserved Bits in the XSTATE_BV Field
AAQ59	X	No Fix	Store Ordering Violation When Using XSAVE
AAQ60	X	No Fix	Memory Ordering Violation With Stores/Loads Crossing a Cacheline Boundary
AAQ61	X	Plan Fix	Unsynchronized Cross-Modifying Code Operations Can Cause Unexpected Instruction Execution Results
AAQ62	X	No Fix	A Page Fault May Not be Generated When the PS bit is set to "1" in a PML4E or PDPTE
AAQ63	X	No Fix	Not-Present Page Faults May Set the RSVD Flag in the Error Code
AAQ64	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
AAQ65	X	No Fix	A 64-bit Register IP-relative Instruction May Return Unexpected Results

Number	SPECIFICATION CHANGES
-	There are no Specification Changes in this Specification Update revision.

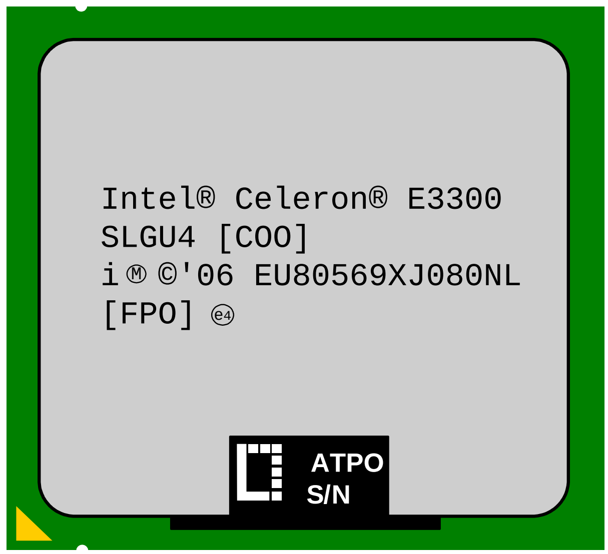
Number	SPECIFICATION CLARIFICATIONS
-	There are no specification clarifications in this Specification Update revision.

Number	DOCUMENTATION CHANGES
AAQ1	On-Demand Clock Modulation Feature Clarification

§

Identification Information

Figure 1. Processor Package Example



§

Component Identification Information

The Intel® Celeron Processor E3x00 series can be identified by the following values:

Family Code[1]	Model Number[2]
0110b	0111b

NOTES:

1. 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.
2. 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.

Cache and TLB descriptor parameters are provided in the EAX, EBX, ECX and EDX registers after the CPUID instruction is executed with a 2 in the EAX register. Refer to the *Intel Processor Identification and the CPUID Instruction Application Note* (AP-485) and the *Wolfdale Family Processor Family BIOS Writer's Guide (BWG)* for further information on the CPUID instruction.

Table 1. Intel® Celeron Processor E3x00 series

S-Spec	Core Stepping	L2 Cache Size (bytes)	Processor Signature	Processor Number	Speed Core / Bus	Package	Notes
SLGTY	R0	1MB	1067Ah	E3500	2.70 GHz/800 Mhz	775-landLGA	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
SLGTZ	R0	1MB	1067Ah	E3400	2.60 GHz/ 800 Mhz	775-land LGA	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
SLGU4	R0	1MB	1067Ah	E3300	2.50 GHz/ 800 Mhz	775-land LGA	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
SLGU5	R0	1MB	1067Ah	E3200	2.40 GHz/ 800 Mhz	775-land LGA	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

NOTES:

1. These processors support the 775_VR_CONFIG_06 specifications
2. These parts support Intel® 64
3. These parts support Execute Disable Bit Feature
4. These parts have PROCHOT# enabled
5. These parts have THERMTRIP# enabled
6. These parts support Thermal Monitor 2 (TM2) feature
7. These parts have PECI enabled
8. These parts have Enhanced Intel SpeedStep® Technology (EIST) enabled
9. These parts have Extended HALT State (C1E) enabled

10. These parts have Extended Stop Grant State (C2E) enabled
11. These parts have Deep Sleep State (C3E) enabled
12. These parts have Deeper Sleep State (C4E) enabled

Errata

AAQ1. EFLAGS Discrepancy on Page Faults after a Translation Change

Problem: This erratum is regarding the case where paging structures are modified to change a linear address from writable to non-writable without software performing an appropriate TLB invalidation. When a subsequent access to that address by a specific instruction (ADD, AND, BTC, BTR, BTS, CMPXCHG, DEC, INC, NEG, NOT, OR, ROL/ROR, SAL/SAR/SHL/SHR, SHLD, SHRD, SUB, XOR, and XADD) causes a page fault, the value saved for EFLAGS may incorrectly contain the arithmetic flag values that the EFLAGS register would have held had the instruction completed without fault. This can occur even if the fault causes a VM exit or if its delivery causes a nested fault.

Implication: None identified. Although the EFLAGS value saved may contain incorrect arithmetic flag values, Intel has not identified software that is affected by this erratum. This erratum will have no further effects once the original instruction is restarted because the instruction will produce the same results as if it had initially completed without a page fault.

Workaround: If the page fault handler inspects the arithmetic portion of the saved EFLAGS value, then system software should perform a synchronized paging structure modification and TLB invalidation.

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

AAQ2. INVLPG Operation for Large (2M/4M) Pages May be Incomplete under Certain Conditions

Problem: The INVLPG instruction may not completely invalidate Translation Look-aside Buffer (TLB) entries for large pages (2M/4M) when both of the following conditions exist:

- Address range of the page being invalidated spans several Memory Type Range Registers (MTRRs) with different memory types specified
- INVLPG operation is preceded by a Page Assist Event (Page Fault (#PF) or an access that results in either A or D bits being set in a Page Table Entry (PTE))

Implication: Stale translations may remain valid in TLB after a PTE update resulting in unpredictable system behavior. Intel has not observed this erratum with any commercially available software.

Workaround: Software should ensure that the memory type specified in the MTRRs is the same for the entire address range of the large page.

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

AAQ3. Store to WT Memory Data May be Seen in Wrong Order by Two Subsequent Loads

Problem: When data of Store to WT memory is used by two subsequent loads of one thread and another thread performs cacheable write to the same address the first load may get the data from external memory or L2 written by another core, while the second load will get the data straight from the WT Store.

Implication: Software that uses WB to WT memory aliasing may violate proper store ordering.

Workaround: Do not use WB to WT aliasing.

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

AAQ4. Non-Temporal Data Store May be Observed in Wrong Program Order

Problem: When non-temporal data is accessed by multiple read operations in one thread while another thread performs a cacheable write operation to the same address, the data stored may be observed in wrong program order (i.e. later load operations may read older data).

Implication: Software that uses non-temporal data without proper serialization before accessing the non-temporal data may observe data in wrong program order.

Workaround: Software that conforms to the Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A, section "Buffering of Write Combining Memory Locations" will operate correctly.

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

AAQ5. Page Access Bit May be Set Prior to Signaling a Code Segment Limit Fault

Problem: If code segment limit is set close to the end of a code page, then due to this erratum the memory page Access bit (A bit) may be set for the subsequent page prior to general protection fault on code segment limit.

Implication: When this erratum occurs, a non-accessed page which is present in memory and follows a page that contains the code segment limit may be tagged as accessed.

Workaround: Erratum can be avoided by placing a guard page (non-present or non-executable page) as the last page of the segment or after the page that includes the code segment limit.

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

AAQ6. Updating Code Page Directory Attributes without TLB Invalidation May Result in Improper Handling of Code #PF

Problem: Code #PF (Page Fault exception) is normally handled in lower priority order relative to both code #DB (Debug Exception) and code Segment Limit

Violation #GP (General Protection Fault). Due to this erratum, code #PF may be handled incorrectly, if all of the following conditions are met:

- A PDE (Page Directory Entry) is modified without invalidating the corresponding TLB (Translation Look-aside Buffer) entry
- Code execution transitions to a different code page such that both
    - The target linear address corresponds to the modified PDE
    - The PTE (Page Table Entry) for the target linear address has an A (Accessed) bit that is clear
- One of the following simultaneous exception conditions is present following the code transition
    - Code #DB and code #PF
    - Code Segment Limit Violation #GP and code #PF

Implication: Software may observe either incorrect processing of code #PF before code Segment Limit Violation #GP or processing of code #PF in lieu of code #DB.

Workaround: None identified.

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

AAQ7. Storage of PEBS Record Delayed Following Execution of MOV SS or STI

Problem: When a performance monitoring counter is configured for PEBS (Precise Event Based Sampling), overflow of the counter results in storage of a PEBS record in the PEBS buffer. The information in the PEBS record represents the state of the next instruction to be executed following the counter overflow. Due to this erratum, if the counter overflow occurs after execution of either MOV SS or STI, storage of the PEBS record is delayed by one instruction.

Implication: When this erratum occurs, software may observe storage of the PEBS record being delayed by one instruction following execution of MOV SS or STI. The state information in the PEBS record will also reflect the one instruction delay.

Workaround: None identified.

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

AAQ8. Performance Monitoring Event FP_MMX_TRANS_TO_MMX May Not Count Some Transitions

Problem: Performance Monitor Event FP_MMX_TRANS_TO_MMX (Event CCH, Umask 01H) counts transitions from x87 Floating Point (FP) to MMX™ instructions. Due to this erratum, if only a small number of MMX instructions (including EMMS) are executed immediately after the last FP instruction, a FP to MMX transition may not be counted.

Implication: The count value for Performance Monitoring Event FP_MMX_TRANS_TO_MMX may be lower than expected. The degree of undercounting is dependent on the occurrences of the erratum condition while the counter is active. 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.

AAQ9. A REP STOS/MOVS to a MONITOR/MWAIT Address Range May Prevent Triggering of the Monitoring Hardware

Problem: The MONITOR instruction is used to arm the address monitoring hardware for the subsequent MWAIT instruction. The hardware is triggered on subsequent memory store operations to the monitored address range. Due to this erratum, REP STOS/MOVS fast string operations to the monitored address range may prevent the actual triggering store to be propagated to the monitoring hardware.

Implication: A logical processor executing an MWAIT instruction may not immediately continue program execution if a REP STOS/MOVS targets the monitored address range.

Workaround: Software can avoid this erratum by not using REP STOS/MOVS store operations within the monitored address range.

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

AAQ10. Performance Monitoring Event MISALIGN_MEM_REF May Over Count

Problem: Performance monitoring event MISALIGN_MEM_REF (05H) is used to count the number of memory accesses that cross an 8-byte boundary and are blocked until retirement. Due to this erratum, the performance monitoring event MISALIGN_MEM_REF also counts other memory accesses.

Implication: The performance monitoring event MISALIGN_MEM_REF may over count. The extent of the over counting depends on the number of memory accesses retiring while the counter is active.

Workaround: None identified.

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

AAQ11. The Processor May Report a #TS Instead of a #GP Fault

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

Implication: Operation systems that access a busy TSS may get invalid TSS fault instead of a #GP fault. 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.

AAQ12. Code Segment Limit Violation May Occur on 4 Gigabyte Limit Check

Problem: Code Segment limit violation may occur on 4 Gigabyte limit check when the code streamwraps around in a way that one instruction ends at the last byte of the segment and the next instruction begins at 0x0.

Implication: This is a rare condition that may result in a system hang. Intel has not observed this erratum with any commercially available software, or system.

Workaround: Avoid code that wraps around segment limit.

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

AAQ13. 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, e.g. 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.

AAQ14. Last Branch Records (LBR) Updates May be Incorrect after a Task Switch

Problem: A Task-State Segment (TSS) task switch may incorrectly set the LBR_FROM value to the LBR_TO value.

Implication: The LBR_FROM will have the incorrect address of the Branch Instruction.

Workaround: None identified.

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

AAQ15. 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 certain conditions as described in the Software Developers Manual section "Out-of-Order Stores For String Operations in Pentium 4, Intel Xeon, and P6 Family Processors" 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 write will now always be 8 bytes, as opposed to the original data size.
- WP the data size of each write will now always be 8 bytes, as opposed to 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.

AAQ16. Upper 32 bits of 'From' Address Reported through BTMs or BTSs May be Incorrect

Problem: When a far transfer switches the processor from 32-bit mode to IA-32e mode, the upper 32 bits of the 'From' (source) addresses reported through the BTMs (Branch Trace Messages) or BTSs (Branch Trace Stores) may be incorrect.

Implication: The upper 32 bits of the 'From' address debug information reported through BTMs or BTSs may be incorrect during this transition.

Workaround: None identified.

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

AAQ17. Address Reported by Machine-Check Architecture (MCA) on Single-bit L2 ECC Errors May be Incorrect

Problem: When correctable Single-bit ECC errors occur in the L2 cache, the address is logged in the MCA address register (MCi_ADDR). Under some scenarios, the address reported may be incorrect.

Implication: Software should not rely on the value reported in MCi_ADDR, for Single-bit L2 ECC errors.

Workaround: None identified.

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

AAQ18. Code Segment Limit/Canonical Faults on RSM May be Serviced before Higher Priority Interrupts/Exceptions

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 (e.g. NMI (Non-Maskable Interrupt), Debug break(#DB), Machine Check (#MC), etc.)

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 identified.

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

AAQ19. Store Ordering May be Incorrect between WC and WP Memory Types

Problem: According to Intel® 64 and IA-32 Intel Architecture Software Developer's Manual, Volume 3A "Methods of Caching Available", WP (Write Protected) stores should drain the WC (Write Combining) buffers in the same way as UC (Uncacheable) memory type stores do. Due to this erratum, WP stores may not drain the WC buffers.

Implication: Memory ordering may be violated between WC and WP stores.

Workaround: None identified.

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

AAQ20. EFLAGS, CR0, CR4 and the EXF4 Signal May be Incorrect after Shutdown

Problem: When the processor is going into shutdown due to an RSM inconsistency failure, EFLAGS, CR0 and CR4 may be incorrect. In addition the EXF4 signal may still be asserted. This may be observed if the processor is taken out of shutdown by NMI#.

Implication: A processor that has been taken out of shutdown may have an incorrect EFLAGS, CR0 and CR4. In addition the EXF4 signal may still be asserted.

Workaround: None identified.

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

AAQ21. Premature Execution of a Load Operation Prior to Exception Handler Invocation

Problem: If any of the below circumstances occur, it is possible that the load portion of the instruction will have executed before the exception handler is entered.

1) If an instruction that performs a memory load causes a code segment limit violation.
2) If a waiting X87 floating-point (FP) instruction or MMX™ technology (MMX) instruction that performs a memory load has a floating-point exception pending.
3) If an MMX or SSE/SSE2/SSE3/SSSE3 extensions (SSE) instruction that performs a memory load and has either CR0.EM=1 (Emulation bit set), or a floating-point Top-of-Stack (FP TOS) not equal to 0, or a DNA exception pending.

Implication: In normal code execution where the target of the load operation is to write back memory there is no impact from the load being prematurely executed, or from the restart and subsequent re-execution of that instruction by the exception handler. If the target of the load is to uncached memory that has a system side-effect, restarting the instruction may cause unexpected system behavior due to the repetition of the side-effect. Particularly, while CR0.TS [bit 3] is set, a MOVD/MOVQ with MMX/XMM register operands may issue a memory load before getting the DNA exception.

Workaround: Code which performs loads from memory that has side-effects can effectively workaround this behavior by using simple integer-based load instructions when accessing side-effect memory and by ensuring that all code is written such that a code segment limit violation cannot occur as a part of reading from side-effect memory.

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

AAQ22. Performance Monitoring Events for Retired Instructions (C0H) May Not Be Accurate

Problem: The INST_RETIRED performance monitor may miscount retired instructions as follows:

- Repeat string and repeat I/O operations are not counted when a hardware interrupt is received during or after the last iteration of the repeat flow.
- VMLAUNCH and VMRESUME instructions are not counted.
- HLT and MWAIT instructions are not counted. The following instructions, if executed during HLT or MWAIT events, are also not counted:
  a) RSM from a C-state SMI during an MWAIT instruction.
  b) RSM from an SMI during a HLT instruction.

Implication: There may be a smaller than expected value in the INST_RETIRED performance monitoring counter. The extent to which this value is smaller than expected is determined by the frequency of the above cases.

Workaround: None identified.

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

AAQ23. 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 values 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.

AAQ24. CMPSB, LODSB, or SCASB in 64-bit Mode with Count Greater or Equal to 2^{48} May Terminate Early

Problem: In 64-bit Mode CMPSB, LODSB, or SCASB executed with a repeat prefix and count greater than or equal to 2^{48} may terminate early. Early termination may result in one of the following.

- The last iteration not being executed
- Signaling of a canonical limit fault (#GP) on the last iteration

Implication: While in 64-bit mode, with count greater or equal to 2^{48}, repeat string operations CMPSB, LODSB or SCASB may terminate without completing the last iteration. Intel has not observed this erratum with any commercially available software.

Workaround: Do not use repeated string operations with RCX greater than or equal to 2^{48}.

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

AAQ25. 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 will be 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.

AAQ26. 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 1 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.

AAQ27. VERW/VERR/LSL/LAR Instructions May Unexpectedly Update the Last Exception Record (LER) MSR

Problem: The LER MSR may be unexpectedly updated, if the resultant value of the Zero Flag (ZF) is zero after executing the following instructions

1. VERR (ZF=0 indicates unsuccessful segment read verification)
2. VERW (ZF=0 indicates unsuccessful segment write verification)
3. LAR (ZF=0 indicates unsuccessful access rights load)
4. LSL (ZF=0 indicates unsuccessful segment limit load)

Implication: The value of the LER MSR may be inaccurate if VERW/VERR/LSL/LAR instructions are executed after the occurrence of an exception.

Workaround: Software exception handlers that rely on the LER MSR value should read the LER MSR before executing VERW/VERR/LSL/LAR instructions.

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

AAQ28. INIT Does Not Clear Global Entries in the TLB

Problem: INIT may not flush a TLB entry when:

- The processor is in protected mode with paging enabled and the page global enable flag is set (PGE bit of CR4 register)
    - G bit for the page table entry is set

- TLB entry is present in TLB when INIT occurs

Implication: Software may encounter unexpected page fault or incorrect address translation due to a TLB entry erroneously left in TLB after INIT.

Workaround: Write to CR3, CR4 (setting bits PSE, PGE or PAE) or CR0 (setting bits PG or PE) registers before writing to memory early in BIOS code to clear all the global entries from TLB.

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

AAQ29. Split Locked Stores May not Trigger the Monitoring Hardware

Problem: Logical processors normally resume program execution following the MWAIT, when another logical processor performs a write access to a WB cacheable address within the address range used to perform the MONITOR operation. Due to this erratum, a logical processor may not resume execution until the next targeted interrupt event or O/S timer tick following a locked store that spans across cache lines within the monitored address range.

Implication: The logical processor that executed the MWAIT instruction may not resume execution until the next targeted interrupt event or O/S timer tick in the case where the monitored address is written by a locked store which is split across cache lines.

Workaround: Do not use locked stores that span cache lines in the monitored address range.

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

AAQ30. 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.

AAQ31. Writing Shared Unaligned Data that Crosses a Cache Line without Proper Semaphores or Barriers May Expose a Memory Ordering Issue

Problem: Software which is written so that multiple agents can modify the same shared unaligned memory location at the same time may experience a memory ordering issue if multiple loads access this shared data shortly thereafter.

Exposure to this problem requires the use of a data write which spans a cache line boundary.

Implication: This erratum may cause loads to be observed out of order. Intel has not observed this erratum with any commercially available software or system.

Workaround: Software should ensure at least one of the following is true when modifying shared data by multiple agents:

- The shared data is aligned
- Proper semaphores or barriers are used in order to prevent concurrent data accesses.

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

AAQ32. General Protection (#GP) Fault May Not Be Signaled on Data Segment Limit Violation above 4-G Limit

Problem: In 32-bit mode, memory accesses to flat data segments (base = 00000000h) that occur above the 4G limit (0ffffffffh) may not signal a #GP fault.

Implication: When such memory accesses occur in 32-bit mode, the system may not issue a #GP fault.

Workaround: Software should ensure that memory accesses in 32-bit mode do not occur above the 4G limit (0ffffffffh).

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

AAQ33. An Asynchronous MCE During a Far Transfer May Corrupt ESP

Problem: If an asynchronous machine check occurs during an interrupt, call through gate, FAR RET or IRET and in the presence of certain internal conditions, ESP may be corrupted.

Implication: If the MCE (Machine Check Exception) handler is called without a stack switch, then a triple fault will occur due to the corrupted stack pointer, resulting in a processor shutdown. If the MCE is called with a stack switch, e.g. when the CPL (Current Privilege Level) was changed or when going through an interrupt task gate, then the corrupted ESP will be saved on the new stack or in the TSS (Task State Segment), and will not be used.

Workaround: Use an interrupt task gate for the machine check handler.

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

AAQ34. CPUID Reports Architectural Performance Monitoring Version 2 is Supported, When Only Version 1 Capabilities are Available

Problem: CPUID leaf 0Ah reports the architectural performance monitoring version that is available in EAX[7:0]. Due to this erratum CPUID reports the supported version as 2 instead of 1.

Implication: Software will observe an incorrect version number in CPUID.0Ah.EAX [7:0] in comparison to which features are actually supported.

Workaround: Software should use the recommended enumeration mechanism described in the Architectural Performance Monitoring section of the Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3: System Programming Guide.

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

AW35. B0-B3 Bits in DR6 For Non-Enabled Breakpoints May be Incorrectly Set

Problem: Some of the B0-B3 bits (breakpoint conditions detect flags, bits [3:0]) in DR6 may be incorrectly set for non-enabled breakpoints when the following sequence happens:

1. MOV or POP instruction to SS (Stack Segment) selector;
2. Next instruction is FP (Floating Point) that gets FP assist
3. Another instruction after the FP instruction completes successfully
4. A breakpoint occurs due to either a data breakpoint on the preceding instruction or a code breakpoint on the next instruction.

Due to this erratum a non-enabled breakpoint triggered on step 1 or step 2 may be reported in B0-B3 after the breakpoint occurs in step 4.

Implication: Due to this erratum, B0-B3 bits in DR6 may be incorrectly set for non-enabled breakpoints.

Workaround: Software should not execute a floating point instruction directly after a MOV SS or POP SS instruction.

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

AAQ36. 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.

AAQ37. Instruction Fetch May Cause a Livelock During Snoops of the L1 Data Cache

Problem: A livelock may be observed in rare conditions when instruction fetch causes multiple level one data cache snoops.

Implication: Due to this erratum, a livelock may occur. Intel has not observed this erratum with any commercially available software.

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

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

AAQ38. Use of Memory Aliasing with Inconsistent Memory Type may Cause a System Hang or a Machine Check Exception

Problem: Software that implements memory aliasing by having more than one linear addresses mapped to the same physical page with different cache types may cause the system to hang or to report a machine check exception (MCE). This would occur if one of the addresses is non-cacheable and used in a code segment and the other is a cacheable address. If the cacheable address finds its way into the instruction cache, and the non-cacheable address is fetched in the IFU, the processor may invalidate the non-cacheable address from the fetch unit. Any micro-architectural event that causes instruction restart will be expecting this instruction to still be in the fetch unit and lack of it will cause a system hang or an MCE.

Implication: This erratum has not been observed with commercially available software.

Workaround: Although it is possible to have a single physical page mapped by two different linear addresses with different memory types, Intel has strongly discouraged his practice as it may lead to undefined results. Software that needs to implement memory aliasing should manage the memory type consistency.

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

AAQ39. A WB Store Following a REP STOS/MOVS or FXSAVE May Lead to Memory-Ordering Violations

Problem: Under certain conditions, as described in the Software Developers Manual section "Out-of-Order Stores For String Operations in Pentium 4, Intel Xeon, and P6 Family Processors", the processor may perform REP MOVS or REP STOS as write combining stores (referred to as “fast strings”) for optimal performance. FXSAVE may also be internally implemented using write combining stores. Due to this erratum, stores of a WB (write back) memory type to a cache line previously written by a preceding fast string/FXSAVE instruction may be observed before string/FXSAVE stores.

Implication: A write-back store may be observed before a previous string or FXSAVE related store. Intel has not observed this erratum with any commercially available software.

Workaround: Software desiring strict ordering of string/FXSAVE operations relative to subsequent write-back stores should add an MFENCE or SFENCE instruction between the string/FXSAVE operation and following store-order sensitive code such as that used for synchronization.

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

AAQ40. Using Memory Type Aliasing with Cacheable and WC Memory Types May Lead to Memory Ordering Violations

Problem: Memory type aliasing occurs when a single physical page is mapped to two or more different linear addresses, each with different memory types. Memory type aliasing with a cacheable memory type and WC (write combining) may cause the processor to perform incorrect operations leading to memory ordering violations for WC operations.

Implication: Software that uses aliasing between cacheable and WC memory types may observe memory ordering errors within WC memory operations. Intel has not observed this erratum with any commercially available software.

Workaround: None identified. Intel does not support the use of cacheable and WC memory type aliasing, and WC operations are defined as weakly ordered.

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

AAQ41. 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. However due to this erratum, only Contributory Exceptions and Page Faults will cause a triple fault shutdown, whereas a benign exception may not.

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.

AAQ42. Short Nested Loops That Span Multiple 16-Byte Boundaries May Cause a Machine Check Exception or a System Hang

Problem: Under a rare set of timing conditions and address alignment of instructions in a short nested loop sequence, software that contains multiple conditional jump instructions and spans multiple 16-byte boundaries, may cause a machine check exception or a system hang.

Implication: Due to this erratum, a machine check exception or a system hang may occur.

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.

AAQ43. 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 AAQare of the potential incorrect debug event signaling created by this erratum.

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

AAQ44. LER MSRs May be Incorrectly Updated

Problem: The LER (Last Exception Record) MSRs, MSR_LER_FROM_LIP (1DDH) and MSR_LER_TO_LIP (1DEH) may contain incorrect values after any of the following:

- Either STPCLK#, NMI (NonMaskable Interrupt) or external interrupts
- CMP or TEST instructions with an uncacheable memory operand followed by a conditional jump
- STI/POP SS/MOV SS instructions followed by CMP or TEST instructions and then by a conditional jump

Implication: When the conditions for this erratum occur, the value of the LER MSRs may be incorrectly updated.

Workaround: None identified.

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

AAQ45. 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.

AAQ46. 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.

AAQ47. PSI# Signal Asserted During Reset

Problem: Power Status Indicator (PSI) is a feature that, when available, may be used to enable voltage regulator power savings while idle and in the Deeper Sleep State (C4 state). Under proper operation the processor will assert the PSI# signal to indicate that the voltage regulator can enter a higher efficiency mode of operation. The processor will incorrectly assert the PSI# signal while the RESET# signal is asserted. This PSI# assertion will extend beyond the deassertion of the RESET# signal for a short duration (maximum of one millisecond).

Implication: When this erratum occurs on a platform designed to support PSI, the voltage regulator will transition to mode of operation that may not be capable of supplying the necessary voltage and current required by the processor.

Workaround: Do not use PSI# signal without blocking the assertion during the error period as specified from RESET# assertion to a maximum of 1ms from the deasserted edge.

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

AAQ48. Thermal Interrupts are Dropped During and While Exiting Intel® Deep Power-Down State

Problem: Thermal interrupts are ignored while the processor is in Intel Deep Power-Down State as well as during a small window of time while exiting from Intel Deep Power-Down State. During this window, if the PROCHOT signal is driven or the internal value of the sensor reaches the programmed thermal trip point, then the associated thermal interrupt may be lost.

Implication: In the event of a thermal event while a processor is waking up from Intel Deep Power-Down State, the processor will initiate an appropriate throttle response. However, the associated thermal interrupt generated may be lost.

Workaround: None identified.

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

AAQ49. Processor May Hold-off / Delay a PECI Transaction Longer than Specified by the PECI Protocol

Problem: PECI (Platform Environment Control Interface) transactions may be held off longer than the PECI protocol hold-off limit while the processor is exiting C-states. This may occur if STPCLK# has been asserted by the system, the beginning of a PECI message coincides with a C-state transition, and the processor is executing a long instruction flow. Note that the processor can still complete the PECI transaction if the host chooses to process the remainder of the message.

Implication: Due to this erratum, the processor may violate the PECI hold-off protocol.

Workaround: PECI hosts can choose to either complete or not complete PECI transactions when the processor goes beyond the hold-off limit. The processor generates the PECI hold-off indication by keeping the PECI bus high when the PECI host sends the first bit of the address timing negotiation phase. If the PECI host does not choose to complete the transaction, it should consider the transaction a failure and retry 1ms after the processor deactivates the hold-off indication.

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

AAQ50. XRSTORE Instruction May Cause Extra Memory Reads

Problem: An XRSTOR instruction will cause non-speculative accesses to XSAVE memory area locations containing the FCW/FSW and FOP/FTW Floating Point (FP) registers even though the 64-bit restore mask specified in the EDX:EAX register pair does not indicate to restore the x87 FPU state.

Implication: Page faults, data breakpoint triggers, etc. may occur due to the unexpected non-speculative accesses to these memory locations.

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.

AAQ51. CPUID Instruction May Return Incorrect Brand String

Problem: When a CPUID instruction is executed with EAX = 8000_0002H, 8000_0003H, or 8000_0004H, the returned EAX, EBX, ECX, and/or EDX values may be incorrect.

Implication: When this erratum occurs, the processor may report an incorrect 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.

AAQ52. When Intel® Deep Power-Down State is Being Used, IA32_FIXED_CTR2 May Return Incorrect Cycle Counts

Problem: When the processor is operating at an N/2 core to front side bus ratio, after exiting an Intel Deep Power-Down State, the internal increment value for IA32_FIXED_CTR2 (Fixed Function Performance Counter 2, 30BH) will not take into account the half ratio setting.

Implication: Due to this erratum, IA32_FIXED_CTR2 MSR will not return reliable counts after returning from an Intel Deep Power-Down State.

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.

AAQ53. Enabling PECI via the PECI_CTL MSR Does Not Enable PECI and May Corrupt the CPUID Feature Flags

Problem: Writing PECI_CTL MSR (Platform Environment Control Interface Control Register) will not update the PECI_CTL MSR (5A0H), instead it will write to the VMM Feature Flag Mask MSR (CPUID_FEATURE_MASK1, 478H).

Implication: Due to this erratum, PECI (Platform Environment Control Interface) will not be enabled as expected by the software. In addition, due to this erratum, processor features reported in ECX following execution of leaf 1 of CPUID (EAX=1) may be masked. Software utilizing CPUID leaf 1 to verify processor capabilities may not work as intended.

Workaround: It is possible for the BIOS to contain a workaround for this erratum. Do not initialize PECI before processor update is loaded. Also, load processor update as soon as possible after RESET as documented in the *RS – Wolfdale Processor Family Bios Writers Guide, Section 14.8.3 Bootstrap Processor Initialization Requirements*.

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

AAQ54. INIT Incorrectly Resets IA32_LSTAR MSR

Problem: In response to an INIT reset initiated either via the INIT# pin or an IPI (Inter Processor Interrupt), the processor should leave MSR values unchanged. Due to this erratum IA32_LSTAR MSR (C0000082H), which is used by the iA32e SYSCALL instruction, is being cleared by an INIT reset.

Implication: If software programs a value in IA32_LSTAR to be used by the SYSCALL instruction and the processor subsequently receives an INIT reset, the SYSCALL instructions will not behave as intended. Intel has not observed this erratum in any commercially available software.

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.

AAQ55. 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.

AAQ56. LBR, BTS, BTM May Report a Wrong Address when an Exception/Interrupt Occurs in 64-bit Mode

Problem: An exception/interrupt event should be transparent to the LBR (Last Branch Record), BTS (Branch Trace Store) and BTM (Branch Trace Message) mechanisms. However, during a specific boundary condition where the exception/interrupt occurs right after the execution of an instruction at the lower canonical boundary (0x00007FFFFFFFFFFF) in 64-bit mode, the LBR return registers will save a wrong return address with bits 63 to 48 incorrectly sign extended to all 1's. Subsequent BTS and BTM operations which report the LBR will also be incorrect.

Implication: LBR, BTS and BTM may report incorrect information in the event of an exception/interrupt.

Workaround: None identified.

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

AAQ57. The XRSTOR Instruction May Fail to Cause a General-Protection Exception

Problem: The XFEATURE_ENABLED_MASK register (XCR0) bits [63:9] are reserved and must be 0; consequently, the XRSTOR instruction should cause a general-protection exception if any of the corresponding bits in the XSTATE_BV field in the header of the XSAVE/XRSTOR area is set to 1. Due to this erratum, a logical processor may fail to cause such an exception if one or more of these reserved bits are set to 1.

Implication: Software may not operate correctly if it relies on the XRSTOR instruction to cause a general-protection exception when any of the bits [63:9] in the XSTATE_BV field in the header of the XSAVE/XRSTOR area is set to 1.

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.

AAQ58. The XSAVE Instruction May Erroneously Modify Reserved Bits in the XSTATE_BV Field

Problem: Bits 63:2 of the HEADER.XSTATE_BV are reserved and must be 0. Due to this erratum, the XSAVE instruction may erroneously modify one or more of these bits.

Implication: If one of bits 63:2 of the XSTATE_BV field in the header of the XSAVE/XRSTOR area had been 1 and was then cleared by the XSAVE instruction, a subsequent execution of XRSTOR may not generate the #GP (general-protection exception) that would have occurred in the absence of this erratum. Alternatively, if one of those bits had been 0 and was then set by the XSAVE instruction, a subsequent execution of XRSTOR may generate a #GP that would not have occurred in the absence of this erratum.

Workaround: It is possible for the BIOS to contain a partial workaround for this erratum that prevents XSAVE from setting HEADER.XSTATE_BV reserved bits. To ensure compatibility with future processors, software should not set any XSTATE_BV reserved bits when configuring the header of the XSAVE/XRSTOR save area.

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

AAQ59. Store Ordering Violation When Using XSAVE

Problem: The store operations done as part of the XSAVE instruction may cause a store ordering violation with older store operations. The store operations done to save the processor context in the XSAVE instruction flow , when XSAVE is used to store only the SSE context, may appear to execute before the completion of older store operations.

Implication: Execution of the stores in XSAVE, when XSAVE is used to store SSE context only, may not follow program order and may execute before older stores. 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.

AAQ60. Memory Ordering Violation With Stores/Loads Crossing a Cacheline Boundary

Problem: When two logical processors are accessing the same data that is crossing a cacheline boundary without serialization, with a specific set of processor internal conditions, it is possible to have an ordering violation between memory store and load operations.

Implication: Due to this erratum, proper load/store ordering may not be followed when multiple logical processors are accessing the same data that crosses a cacheline boundary without serialization.

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.

AAQ61. 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 3: System Programming Guide*, including a General Protection Fault (GPF) or other unexpected behaviors. In the event that unpredictable execution causes a GPF the application executing the unsynchronized XMC operation would be terminated by the operating system.

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 3: System Programming Guide*, Section: Handling Self- and Cross-Modifying Code.

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

AAQ62. 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.

AAQ63. Global Instruction TLB Entries May Not be Invalidated on a VM Exit or VM Entry

Problem: If a VMM is using global page entries (CR4.PGE is enabled and any present page-directories or page-table entries are marked global), then on a VM entry, the instruction TLB (Translation Lookaside Buffer) entries caching global page translations of the VMM may not be invalidated. In addition, if a guest is using global page entries, then on a VM exit, the instruction TLB entries caching global page translations of the guest may not be invalidated.

Implication: Stale global instruction linear to physical page translations may be used by a VMM after a VM exit or a guest after a VM entry.

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.

AAQ64. 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.

AAQ65. A 64-bit Register IP-relative Instruction May Return Unexpected Results

Problem: Under an unlikely and complex sequence of conditions in 64-bit mode, a register IP-relative instruction result may be incorrect.

Implication: A register IP-relative instruction result may be incorrect and could cause software to read from or write to an incorrect memory location. This may result in an unexpected page fault or 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

The Specification Changes listed in this section apply to the following documents:

- *Intel® Celeron Processor E3x00 Series Datasheet*
- *Intel® 64 and IA-32 Architectures Software Developer's Manual volumes 1,2A, 2B, 3A, and 3B*

All Specification Changes will be incorporated into a future version of the appropriate processor documentation.

§

Specification Clarifications

The Specification Clarifications listed in this section apply to the following documents:

- *Intel® Celeron Processor E3x00 Series Datasheet*
- *Intel® 64 and IA-32 Architectures Software Developer's Manual volumes 1,2A, 2B, 3A, and 3B*

All Specification Clarifications will be incorporated into a future version of the appropriate processor documentation.

§

Documentation Changes

The Documentation Changes listed in this section apply to the following documents:

- *Intel® Celeron Processor E3x00 Series Datasheet*

All Documentation Changes will be incorporated into a future version of the appropriate processor documentation.

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

http://www.intel.com/design/processor/specupdt/252046.htm

AAQ1. On-Demand Clock Modulation Feature Clarification

Software Controlled Clock Modulation section of the Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B: System Programming Guide will be modified to differentiate On-demand clock modulation feature on different processors. The clarification will state:

For Hyper-Threading Technology enabled processors, the IA32_CLOCK_MODULATION register is duplicated for each logical processor. In order for the On-demand clock modulation feature to work properly, the feature must be enabled on all the logical processors within a physical processor. If the programmed duty cycle is not identical for all the logical processors, the processor clock will modulate to the highest duty cycle programmed for processors if the CPUID DisplayFamily_DisplayModel signatures is listed in Table 14-2. For all other processors, if the programmed duty cycle is not identical for all logical processors in the same core, the processor will modulate at the lowest programmed duty cycle.

For multiple processor cores in a physical package, each core can modulate to a programmed duty cycle independently.

For the P6 family processors, on-demand clock modulation was implemented through the chipset, which controlled clock modulation through the processor's STPCLK# pin.

Table 14-2. CPUID Signatures for Legacy Processors That Resolve to Higher Performance Setting of Conflicting Duty Cycle Requests

DisplayFamily_ DisplayModel	DisplayFamily_ DisplayModel	DisplayFamily_ DisplayModel	DisplayFamily_ DisplayModel
0F_xx	06_1C	06_1A	06_1E
06_1F	06_25	06_26	06_27
06_2C	06_2E	06_2F	06_35
06_36			

§