Intel® Atom™ Processor Z6xx Series

Specification Update

For the Intel® Atom™ Processor Z600$^{\Delta}$, Z605$^{\Delta}$, Z610$^{\Delta}$, Z612$^{\Delta}$, Z615$^{\Delta}$, Z620$^{\Delta}$, and Z625$^{\Delta}$ on 45 nm process technology

May 2011

Revision 001

Document Number: 325630-001US

Contents

Revision History

Document Number	Revision	Description	Date
325630	001	• Initial release	May 2011

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 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 Z6xx Series Datasheet For the Intel® Atom™ Processors Z600, Z605, Z610, Z612, Z615, Z620, and Z625 on 45-nm Process Technology	325567-001US

Related Documents

Document Title	Document Number/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, A-M	
Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 2B: Instruction Set Reference, 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

Errata are design defects or errors. These may cause the Intel® Atom™ processor Z6xx series on 45-nm process behavior to deviate from published specifications. Hardware and software designed to be used with any given stepping assumes 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, that is, core speed, L2 cache size, and package type as described in the processor identification information table. Read all notes associated with each S-Spec number.

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

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 (such as; datasheets, and manuals).

§

Summary Tables of Changes

The following table indicates the Specification Changes, Errata, Specification Clarifications or Documentation Changes, which apply to the listed 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 list 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.

Number	Stepping	PLAN	ERRATA
	C0		
BB1	X	No Fix	IO_SMI Indication in SMRAM State Save Area May be Set Incorrectly
BB2	X	No Fix	Writes to IA32_DEBUGCTL MSR May Fail when FREEZE_LBRS_ON_PMI is Set
BB3	X	No Fix	Address Reported by Machine-Check Architecture (MCA) on L2 Cache Errors May be Incorrect
BB4	X	No Fix	Pending x87 FPU Exceptions (#MF) Following STI May Be Serviced Before Higher Priority Interrupts
BB5	X	No Fix	Benign Exception after a Double Fault May Not Cause a Triple Fault Shutdown
BB6	X	No Fix	IA32_MC1_STATUS MSR Bit[60] Does Not Reflect Machine Check Error Reporting Enable Correctly
BB7	X	No Fix	Performance Monitoring Event for Outstanding Bus Requests Ignores AnyThread Bit
BB8	X	No Fix	IRET under Certain Conditions May Cause an Unexpected Alignment Check Exception
BB9	X	No Fix	Thermal Interrupts are Dropped During and While Exiting Intel Deep Power Down State
BB10	X	No Fix	Corruption of CS Segment Register During RSM While Transitioning From Real Mode to Protected Mode
BB11	X	No Fix	Performance Monitoring Counter with AnyThread Bit set May Not Count on a Non-Active Thread
BB12	X	No Fix	GP and Fixed Performance Monitoring Counters With AnyThread Bit Set May Not Accurately Count Only OS or Only USR Events
BB13	X	No Fix	PMI Request is Not Generated on a Counter Overflow if Its OVF Bit is Already Set in IA32_PERF_GLOBAL_STATUS
BB14	X	No Fix	Processor May Use an Incorrect Translation if the TLBs Contain Two Different Translations For a Linear Address
BB15	X	No Fix	A Write to an APIC Register Sometimes May Appear to Have Not Occurred
BB16	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
BB17	X	No Fix	The Processor May Report a #TS Instead of a #GP Fault
BB18	X	No Fix	Writing the Local Vector Table (LVT) when an Interrupt is Pending May Cause an Unexpected Interrupt
BB19	X	No Fix	MOV To/From Debug Registers Causes Debug Exception
BB20	X	No Fix	Using 2M/4M Pages When A20M# Is Asserted May Result in Incorrect Address Translations
BB21	X	No Fix	Values for LBR/BTS/BTM will be Incorrect after an Exit from SMM

Number	Stepping	PLAN	ERRATA
	C0		
BB22	X	No Fix	Incorrect Address Computed For Last Byte of FXSAVE/FXRSTOR Image Leads to Partial Memory Update
BB23	X	No Fix	A Thermal Interrupt is Not Generated when the Current Temperature is Invalid
BB24	X	No Fix	Programming the Digital Thermal Sensor (DTS) Threshold May Cause Unexpected Thermal Interrupts
BB25	X	No Fix	Returning to Real Mode from SMM with EFLAGS.VM Set May Result in Unpredictable System Behavior
BB26	X	No Fix	Fault on ENTER Instruction May Result in Unexpected Values on Stack Frame
BB27	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
BB28	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
BB29	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
BB30	X	No Fix	BTS(Branch Trace Store) and PEBS(Precise Event Based Sampling) May Update Memory outside the BTS/PEBS Buffer
BB31	X	No Fix	Single Step Interrupts with Floating Point Exception Pending May Be Mishandled
BB32	X	No Fix	Unsynchronized Cross-Modifying Code Operations Can Cause Unexpected Instruction Execution Results
BB33	X	No Fix	LBR Stack May Not be Frozen on a PMI Request When FREEZE_LBRS_ON_PMI is Set
BB34	X	No Fix	Processor Throttling at 87.5% May Cause System Hang
BB35	X	No Fix	External STPCLK# Throttling May Cause System Hang During Thermal Event
BB36	X	No Fix	THERMTRIP# Will Not Assert Prior to RESET# De-assertion
BB37	X	No Fix	C6 Request May Cause a Machine Check if the Other Logical Processor is in C4 or C6
BB38	X	No Fix	EOI Transaction May Not be Sent if Software Enters Core C6 During an Interrupt Service Routine

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
	There are no Document Changes in this revision of the Specification Update

§

Identification Information

The Intel® Atom™ processor Z6xx series on 45-nm process stepping can be identified by the following register contents:

Table 1. Component Identification by Using the 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	0010b		0b	0110b	0110b	XXXXb

NOTES:

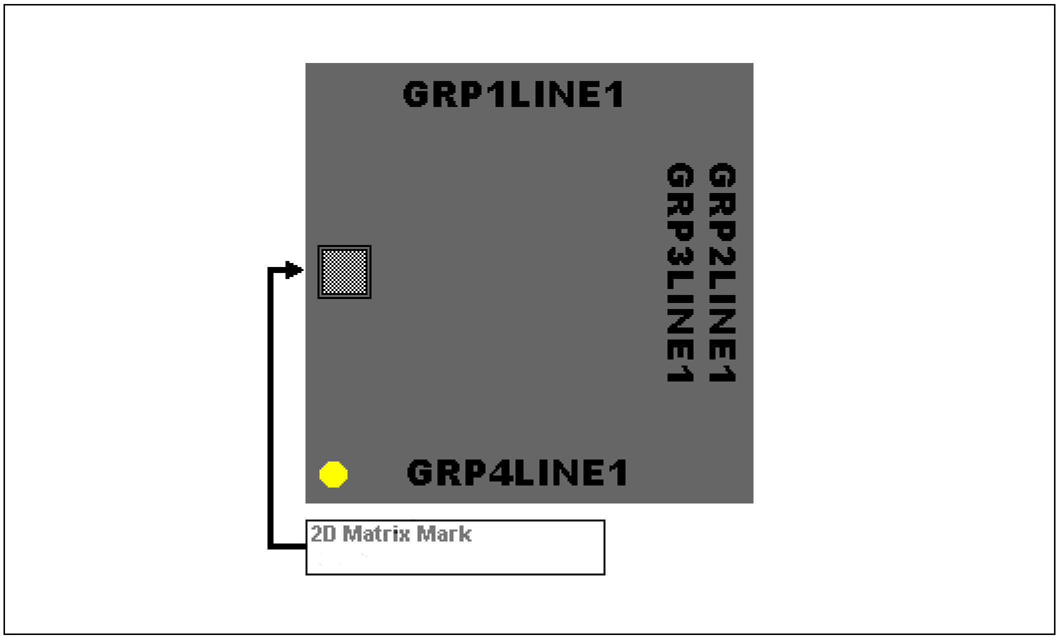
1. The Extended Family, bits [27:20] are used in conjunction with the Family Code, specified in bits [11:8], to indicate whether the processor belongs to the Intel386™, Intel486™, Pentium®, Pentium® Pro, Pentium® 4, or Intel Core™2 processor family.
2. The Extended Model, bits [19:16] in conjunction 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.

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.

Component Marking Information

The Intel® Atom™ processor Z6xx series can be identified by the following component markings.

Figure 1. Intel® Atom™ Processor Z6xx Series Markings



GRP1LINE1: PROC# SPEED

GRP2LINE1: {FPO}

GRP3LINE1: SSPEC

GRP4LINE1: INTEL {M}{C}YY{e1}

Table 2. Identification Table for Intel® Atom™ Processor Z6xx Series

QDF/ S-Spec	Product Stepping	HFM TDP	Processor #	Processor Signature	Core Speed		Package	MCU
					High Freq. Mode (HFM)	Burst Freq. Mode (BFM)		
LBZP	C0	1.3 W	Z600	00020661h	0.8 GHz	1.2 GHz	FCMBA3	M01106C2217
SLC3C	C0	2.2 W	Z605	00020661h	1.0 GHz	N/A	FCBGA8	M01106C2217
LBZQ	C0	1.3 W	Z610	00020661h	0.8 GHz	1.2 GHz	FCBGA8	M01106C2217
LBZN	C0	1.3 W	Z612	00020661h	0.9 GHz	1.5 GHz	FCBGA8	M01106C2217
LBZL	C0	2.2 W	Z615	00020661h	1.2 GHz	1.6 GHz	FCBGA8	M01106C2217
LBZE	C0	1.3 W	Z620	00020661h	0.9 GHz	1.5 GHz	FCBGA8	M01106C2217
LBZD	C0	2.2 W	Z625	00020661h	1.5 GHz	1.9 GHz	FCBGA8	M01106C2217

Microcode Updates

Each unique processor stepping/package combination has an associated microcode update that, when applied, constitutes a supported processor (that is, Specified processor = Processor Stepping + Microcode Update). The proper microcode update must be loaded on each processor in a system. The proper microcode update is defined as the latest production microcode update available from Intel for a given family, model and stepping of the processor. Any processor that does not have the correct microcode update loaded is considered to be operating out of specification. Contact your Intel Field Representative to receive the latest production microcode updates.

Table 3. Intel® Atom™ Processor Z6xx Series on 45-nm Process Microcode Update Guide

Microcode Update	Microcode Update Release Date	Processor Signature	Microcode Update Revision	Revision ID	Core Stepping	Errata ID
M0120661104	12/2009	20661h	104	104	C0	BB37

§

Errata

BB1 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.

BB2 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.

BB3 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 identified

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

BB4 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.

BB5 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.

BB6 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.

BB7 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.

BB8 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: InIA-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.

BB9 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.

BB10 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.

BB11 Performance Monitoring Counter with AnyThread Bit set May Not Count on a Non-Active Thread

Problem: A performance counter with the AnyThread bit (IA32_PERFEVTSEL0 MSR (186H)/ IA32_PERFEVTSEL1 MSR (187H) bit [21], IA32_FIXED_CTR_CTRL MSR (38DH) bit [2] for IA32_FIXED_CTR0, bit [6] for IA32_FIXED_CTR1, bit [10] for IA32_FIXED_CTR2) set should count that event on all logical processors on that core. Due to this erratum, a performance counter on a logical processor which has requested to be placed in the Intel® Deep Power Down State may not count events that occur on another logical processor.

Implication: The performance monitor count may be incorrect when the logical processor is asleep but still attempting to count another logical processor's events. This will only occur on processors supporting Hyper-Threading Technology (HT Technology).

Workaround: None identified.

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

BB12 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.

BB13 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.

BB14 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.

BB15 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.

BB16 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: The IA32 firmware must leave the xTPR update transactions disabled (default).

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

BB17 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: Operating 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.

BB18 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.

BB19 MOV To/From Debug Registers Causes Debug Exception

Problem: When in V86 mode, if a MOV instruction is executed to/from a debug register~~s~~, 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 will be 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.

BB20 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
- Linear address has bit-20 set
- 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.

BB21 Values 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 three 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 identified.

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

BB22 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 64-KB limit while the processor is operating in 16-bit mode or if a memory address exceeds the 4-GB 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.

BB23 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 identified.

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

BB24 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.

BB25 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.

BB26 Fault on ENTER Instruction May Result in Unexpected Values 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 values (that is, 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 the *Intel® 64 and IA-32 Architectures 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 identified.

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

BB27 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 will be taken when not expected.

Workaround: None identified.

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

BB28 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 *Intel® 64 and IA-32 Architectures Software Developer's Manual*, the use of MOV SS/POP SS in conjunction with MOV [r/e]SP and [r/e]BP will avoid the failure since the MOV [r/e]SP and [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.

BB29 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 identified.

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

BB30 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 4-G 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 *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3.*

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

BB31 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 will be 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 identified.

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

BB32 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® 64 and IA-32 Architectures 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® 64 and IA-32 Architectures 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.

BB33 LBR Stack May Not be Frozen on a PMI Request When FREEZE_LBRS_ON_PMI is Set

Problem: When the FREEZE_LBRS_ON_PMI flag (IA32_DEBUGCTL MSR (1D9H) bit [11]) is set on an Atom processor, a PMI (performance monitor interrupt) request should cause the LBR and TR flags (IA32_DEBUGCTL MSR (1D9H) bit [1:0]) to be cleared and the LBR (last branch record) stack to stop being updated by branches/interrupts/exceptions. Due to this erratum, the processor may clear the LBR and TR flags but not stop the LBR stack from being updated when FREEZE_LBRS_ON_PMI is set and a PMI request occurs.

Implication: Following a PMI request, the LBRs may continue to be updated by branches/interrupts/exceptions even when FREEZE_LBRS_ON_PMI is set. The LBRs may contain values recorded after the PMI request.

Workaround: None identified.

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

BB34 Processor Throttling at 87.5% May Cause System Hang

Problem: At certain processor core ratios, a request for 87.5% throttling (on-demand clock modulation duty cycle) may cause the processor to hang.

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

Workaround: A Power Management Unit firmware code change has been identified and may be implemented as a workaround for this erratum. Throttling at 87.5% will not be supported and any requests for 87.5% will be re-directed to 75%.

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

BB35 External STPCLK# Throttling May Cause System Hang During Thermal Event

Problem: During a TM1 thermal event when external STPCLK# throttling is enabled through software, the time between STPCLK# assertion and STOP GRANT may not be enough for the processor to de-assert STPCLK# and execute instructions.

Implication: When this erratum occurs, the system will hang.

Workaround: Software should not enable software-controlled STPCLK# throttling.

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

BB36 THERMTRIP# Will Not Assert Prior to RESET# De-assertion

Problem: Potentially catastrophic temperature should be detected and signaled using the THERMTRIP# mechanism after PWRGD assertion. Due to this erratum, THERMTRIP# functionality is not supported during the period from PWRGD assertion to RESET# de-assertion. After RESET# de-assertion, THERMTRIP# functions correctly.

Implication: Due to this erratum, THERMTRIP# will not function until after RESET# de-assertion.

Workaround: None identified.

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

BB37 C6 Request May Cause a Machine Check if the Other Logical Processor is in C4 or C6

Problem: A machine check may be generated if a logical processor requests the C6 C-state and the other logical processor is in either the C4 or C6 C-states.

Implication: This erratum may result in unexpected machine-check exceptions.

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

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

BB38 EOI Transaction May Not be Sent if Software Enters Core C6 During an Interrupt Service Routine

Problem: If core C6 is entered after the start of an interrupt service routine but before a write to the APIC EOI (End of Interrupt) register, and the core is woken up by an event other than a fixed interrupt source the core may drop the EOI transaction the next time APIC EOI register is written and further interrupts from the same or lower priority level will be blocked.

Implication: EOI transactions may be lost and interrupts may be blocked when core C6 is used during interrupt service routines.

Workaround: Software should check the ISR register and if any interrupts are in service only enter C1.

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

§

Specification Changes

There are no Document Changes in this revision of the specification Update.

§

Specification Clarifications

There are no Document Changes in this revision of the specification Update.

§

Documentation Changes

There are no Document Changes in this revision of the specification Update.

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 http://www.intel.com/products/processor_number for details.

§