# intel®

# Pentium® Pro Processor Specification Update

Release Date: March, 1997

Order Number 242689-017

The Pentium® Pro processor may contain design defects or errors known as errata. Characterized errata that may cause the Pentium Pro processor's behavior to deviate from published specifications are documented in this Specification Update.

# CONTENTS

## REVISION HISTORY

| Date of Revision | Version | Description |
|---|---|---|
| November, 1995 | -001 | This document is the first Specification Update for the Intel Pentium® Pro processor. |
| December, 1995 | -002 | Added Errata 29-38, and 3AP. Included additional S-specs. Added a case to Erratum 3. Clarified Errata 2, 6, 8, 10, 11, and 26. |
| January, 1996 | -003 | Added Errata 39-42. Updated the status for Erratum 10. Added documentation for 166-, 180-, and 200-MHz processors. |
| February, 1996 | -004 | Clarified Errata 11, 22, 30, and 34. Updated status for Errata 10, 15, 40, and 42. Added CPUID information for determining L2 cache size. Added S-specs and errata column for sA1 stepping. |
| March, 1996 | -005 | Added Errata 43-46. Added Specification Change 1 and Documentation Change 1. Clarified Erratum 2 and corrected Errata 7 and 39. Added L2 cache stepping information. |
| April, 1996 | -006 | Added Erratum 47. Updated General Information section with new S-specs and marking diagrams. |
| May, 1996 | -007 | Corrected Erratum 43. Added Erratum 48. |
| June, 1996 | -008 | Added Specification Change 2, Errata 49 and 4AP, Specification Clarifications 1-3, and Documentation Changes 2 and 3. Updated status of Errata 7, 33, and 43. Updated S-spec table with additional and corrected sA0 and sA1 S-spec information. |
| July, 1996 | -009 | Added Specification Change 3, Errata 50 and 5AP, Specification Clarifications 4 and 5, and Documentation Change 4. Updated Erratum 43. |
| August, 1996 | -010 | Updated Errata 50 and 2AP. Added Specification Changes 4-6, Errata 51 and 6AP, and Specification Clarifications 6 and 7. |
| September, 1996 | -011 | Updated Specification Change 1. Added Errata 52-54. Added Specification Clarifications 8 and 9. |
| October, 1996 | -012 | Updated Errata 49 and 54. Added Specification Clarifications 10-12. Added Documentation Changes 5-8. |
| November, 1996 | -013 | Updated S-spec table. Updated Errata 33, 49, 51, 54, 5AP and 6AP. Added Errata 55, 7AP and 8AP, Specification Clarifications 13 and 14, and Documentation Changes 9-11. Updated Specification Change 1. |
| December, 1996 | -014 | Updated S-spec table and corrected the L2 cache stepping for SY040. Updated marking diagrams for sB1 components. Updated Specification Change 1, Erratum 54. Added Specification Change 7 and Errata 56, 57, and 9AP. |
| January, 1997 | -015 | Added Errata 58 and 59. Added Specification Clarification 15. |
| February, 1997 | -016 | Updated Errata 58 and 59. Corrected Status bar for Erratum 56. Added Specification Change 8. Updated Specification Clarification 14. Added Documentation Change 12. |
| March, 1997 | -017 | Added Documentation Change 13. Updated S-Spec table. |

# PREFACE

This document is an update to the specifications contained in the *Pentium® Pro Family Developer's Manual*, *Volumes 1, 2, and 3* (Order Numbers 242690, 242691, and 242692, respectively). It is intended for hardware system manufacturers and software developers of applications, operating systems, or tools. It contains Specification Changes, S-Specs, Errata, Specification Clarifications, and Documentation Changes.

## Nomenclature

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

**S-Specs** are exceptions to the published specifications, and apply only to the units assembled under that s-spec.

**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.

**Errata** are design defects or errors. Errata may cause the Pentium® Pro 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.

## Identification Information

The Pentium Pro processor can be identified by the following values:

| Family[1] | 150-, 166-, 180-, and 200-MHz  Model1[2] |
|---|---|
| 0110 | 0001 |

**NOTES:**

1   The Family 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 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.

The Pentium Pro processor's level 2 (L2) cache size can be determined by the following values:

| 256-Kbyte Unified L2 Cache[1] | 512-Kbyte Unified L2 Cache[1] |
|---|---|
| 42h | 43h |

**NOTES:**

1   For this model of the Pentium Pro processor, the unified L2 cache size corresponds to the value in bits [3:0] of the EDX register after the CPUID instruction is executed with a 2 in the EAX register.  Other Intel microprocessor models or families may move this information to other bit positions or otherwise reformat the result returned by this instruction; generic code should parse the resulting token stream according to the definition of the CPUID instruction.

# Specification Update for 150-, 166-, 180-, and 200-MHz Pentium® Pro Processors

# GENERAL INFORMATION

## *Top Markings*

**B-Step Production Units – Top:**

KB8Ø521EXZZZ SYYYY 256K

**intel**®

FFFFFFFF-XXXX
INTEL Ⓜ Ⓒ '94 '95

**B-, C-, and sA-Step Production Units – Top:**

KB8Ø521EXZZZ SYYYY LLLK

**intel**®
**PENTIUM**®**PRO**

FFFFFFFF-XXXX
INTEL Ⓜ Ⓒ '94 '95

**sB1-Step Production Units – Top:**

KB8Ø521EZZZ SLYYY 256K
ICOMP® 2# =CCC

**intel**®
**PENTIUM**®**PRO**

FFFFFFFF-XXXX
INTEL Ⓜ Ⓒ '94 '96
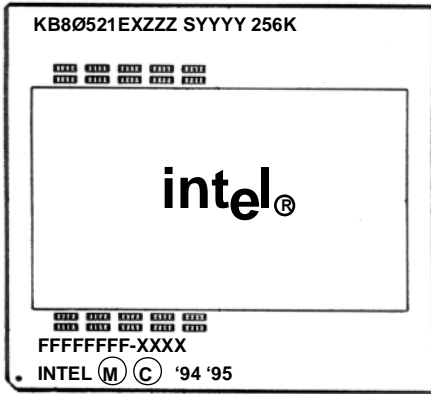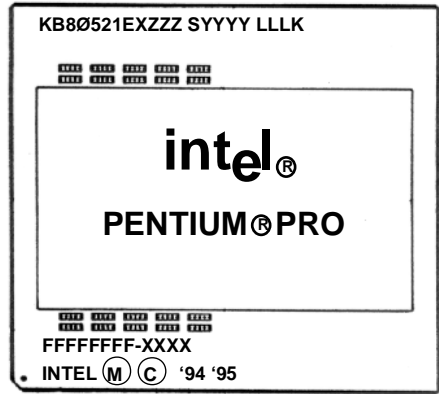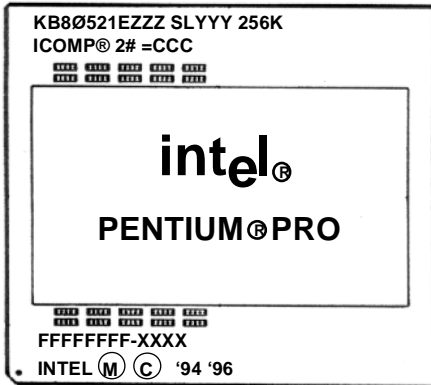
## *Bottom Markings*

**B-, C-, and sA-Step Production Units – Bottom:**

```
XXXXXXXXAA
YYYYYYYYAA
MALAY K


KB8Ø521EXZZZ
SYYYY LLLK
```

**sB1-Step Production Units – Bottom:**

```
XXXXXXXXXAA
MMMMM K


KB8Ø521EXZZZ
SLYYY 256K
```
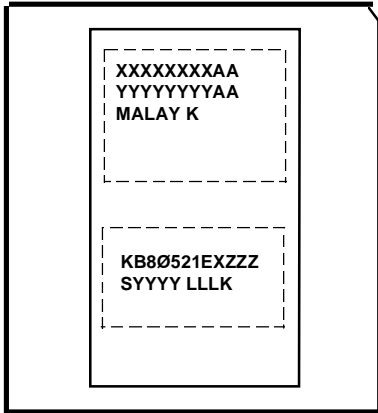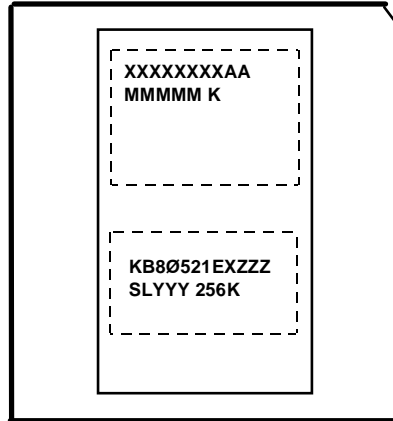
**NOTES:**

- ZZZ = Speed (MHz).
- QYYYY = Sample Specification Number.
- SYYYY = S-spec Number.
- LLLK = Level 2 Cache Size.
- FFFFFFFF = FPO # (Test Lot Traceability #).

- XXXX = Serialization Code.
- XXXXXXXXAA = Alternative Identification Number.
- Top side: inner line indicates boundary of heat spreader.

**Basic 150-, 166-, 180-, and 200-MHz Pentium® Pro Processor Identification Information**

| | CPUID | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Type | Family | Model | Stepping | Mfg. Stepping | L2[1] Size/ Cache Stepping | Speed (MHz) Core/Bus | S-Spec | $V_{CC}$ | $T_{CASE}$ | Notes |
| 0 | 6 | 1 | 1 | B0 | 256/α | 150/60 | SY002 | 3.1V ± 5% | 0°C - 85°C | 3 |
| 0 | 6 | 1 | 1 | B0 | 256/α | 150/60 | SY011 | 3.1V ± 5% | 0°C - 85°C | |
| 0 | 6 | 1 | 1 | B0 | 256/α | 150/60 | SY014 | 3.1V ± 5% | 0°C - 85°C | |
| 0 | 6 | 1 | 2 | C0 | 256/α | 150/60 | SY010 | 3.1V ± 5% | 0°C - 85°C | |
| 0 | 6 | 1 | 6 | sA0[2] | 256/α | 180/60 | SY012 | 3.3V ± 5% | 0°C - 85°C | |
| 0 | 6 | 1 | 6 | sA0[2] | 256/α | 200/66 | SY013 | 3.3V ± 5% | 0°C - 85°C | |
| 0 | 6 | 1 | 7 | sA1 | 256/α | 200/66 | SL245 | 3.5V ± 5% | 0°C - 85°C | 4 |
| 0 | 6 | 1 | 7 | sA1 | 256/α | 200/66 | SL247 | 3.5V ± 5% | 0°C - 85°C | 4 |
| 0 | 6 | 1 | 7 | sA1 | 256/β | 180/60 | SU103 | 3.3V ± 5% | 0°C - 85°C | |
| 0 | 6 | 1 | 7 | sA1 | 256/β | 200/66 | SU104 | 3.3V ± 5% | 0°C - 85°C | |
| 0 | 6 | 1 | 7 | sA1 | 256/β | 180/60 | SY031 | 3.3V ± 5% | 0°C - 85°C | |
| 0 | 6 | 1 | 7 | sA1 | 256/β | 200/66 | SY032 | 3.3V ± 5% | 0°C - 85°C | |
| 0 | 6 | 1 | 7 | sA1 | 512/α | 166/66 | SY034 | 3.3V ± 5% | 0°C - 85°C | |
| 0 | 6 | 1 | 7 | sA1 | 256/α | 180/60 | SY039 | 3.3V ± 5% | 0°C - 85°C | |
| 0 | 6 | 1 | 7 | sA1 | 256/β | 200/66 | SY040 | 3.3V ± 5% | 0°C - 85°C | |
| 0 | 6 | 1 | 7 | sA1 | 512/β | 166/66 | SY047 | 3.3V ± 5% | 0°C - 85°C | |
| 0 | 6 | 1 | 7 | sA1 | 512/β | 200/66 | SY048 | 3.3V ± 5% | 0°C - 85°C | |
| 0 | 6 | 1 | 9 | sB1 | 256/β | 180/60 | SL22S | 3.3V ± 5% | 0°C - 85°C | |
| 0 | 6 | 1 | 9 | sB1 | 256/β | 200/66 | SL22T | 3.3V ± 5% | 0°C - 85°C | |
| 0 | 6 | 1 | 9 | sB1 | 256/β | 180/60 | SL22U | 3.3V ± 5% | 0°C - 85°C | |
| 0 | 6 | 1 | 9 | sB1 | 256/β | 200/66 | SL22V | 3.3V ± 5% | 0°C - 85°C | |
| 0 | 6 | 1 | 9 | sB1 | 512/β | 166/66 | SL22X | 3.3V ± 5% | 0°C - 85°C | |
| 0 | 6 | 1 | 9 | sB1 | 512/β | 200/66 | SL22Z | 3.3V ± 5% | 0°C - 85°C | |
| 0 | 6 | 1 | 9 | sB1 | 256/β | 180/60 | SL23L | 3.3V ± 5% | 0°C - 85°C | |
| 0 | 6 | 1 | 9 | sB1 | 256/β | 200/66 | SL23M | 3.3V ± 5% | 0°C - 85°C | |
| 0 | 6 | 1 | 9 | sB1 | 256/β | 200/66 | SL254 | 3.5V ± 5% | 0°C - 85°C | 4 |
| 0 | 6 | 1 | 9 | sB1 | 256/β | 200/66 | SL255 | 3.5V ± 5% | 0°C - 85°C | 4 |

**NOTES:**

1. **L2 Cache Stepping** refers to the silicon revision of the 256-Kbyte or 512-Kbyte on-chip L2 cache. The α designation refers to the first production steppings, the β to the second production steppings, and so on.
2. The sA0 stepping is logically equivalent to the C0 stepping, but on a different manufacturing process.
3. The VID pins are not supported on these parts.

4.  These components have additional specification changes associated with them:

    a.  $V_{CC}P$ = Primary $V_{CC}$ = 3.5V ± 5%

    b.  $P_{MAX}$ = Max Thermal Design Power = 39.4W @ 200Mhz, 256K L2

    c.  $I_{CC}P$ = $V_{CC}P$ Current = 11.9A

    d.  The VID pins are not supported on these parts.

    e.  T9 = Minimum GTL+ Input Hold Time = 0.9ns

    f.  $V_{IH}$min = Minimum Non-GTL+ Input High Voltage = 2.2V  [Only PICD[1:0] signals affected. Clocks are unaffected]

## Summary Table of Changes

The following table indicates the Specification Changes, Errata, Specification Clarifications, or Documentation Changes which apply to the 150-, 166-, 180-, and 200-MHz Pentium Pro processors. 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**

| | |
|---|---|
| X: | Specification Change, Erratum, Specification Clarification, or Documentation Change applies to the given stepping. |
| Doc: | Intel intends to update the appropriate documentation in a future revision. |
| Fix: | Intel is investigating the possibility of fixing this erratum in a future stepping of the component(s). |
| Fixed: | This erratum has been previously fixed. |
| NoFix: | Intel is currently not investigating a fix for this erratum. |
| (No mark) or (blank box): | This item is fixed in or does not apply to the given stepping. |
| AP: | APIC related erratum. |
| Shaded: | This erratum is either new or modified from the previous version of the document. |

| NO. | B0 | C0 | sA0 | sA1 | sB1 | Plans | SPECIFICATION CHANGES |
|---|---|---|---|---|---|---|---|
| 1 | X | X | X | X | X | Doc | Mixing steppings in MP systems |
| 2 | X | X | X | X | X | Doc | Change in PICD[1:0] hold time and valid delay |
| 3 | X | X | X | X | X | Doc | Undershoot specification for some GTL+ signals |
| 4 | | | | | | Doc | OverDrive® VRM motherboard bulk capacitance specification |
| 5 | X | X | X | X | X | Doc | MTRR precedences and memory type overriding |
| 6 | X | X | X | X | X | Doc | $V_{CC}S$ pins removed from specification |
| 7 | | | | | X | Doc | Split lock across cache line boundary disable bit added |
| 8 | | | | | X | Doc | Instruction Streaming Buffer Disable Bit Added |

| NO. | B0 | C0 | sA0 | sA1 | sB1 | Plans | ERRATA |
|---|---|---|---|---|---|---|---|
| 1 | X | X | X | X | X | NoFix | Mixed cacheability of lock variables is problematic in MP systems |
| 2 | X | X | X | X | X | NoFix | FP Operand Pointer may be incorrectly calculated after FP access which wraps 64-Kbyte boundary in 16-bit code |
| 3 | X | X | X | X | X | NoFix | Differences exist in debug exception reporting |
| 4 | X | X | X | X | X | NoFix | FLUSH# servicing delayed while waiting for STARTUP_IPI in MP systems |
| 5 | X | | | | | Fixed | Fast Strings REP MOVS may not transfer all data |
| 6 | X | | | | | Fixed | Page table base change during task switch using Mode C paging may corrupt EIP |
| 7 | X | X | X | X | X | NoFix | Code fetch matching disabled debug register may cause debug exception |

| NO. | B0 | C0 | sA0 | sA1 | sB1 | Plans | ERRATA |
|-----|----|----|-----|-----|-----|-------|--------|
| 8 | X | X | X | X | X | NoFix | Mode C paging in SMM causes use of incorrect page tables |
| 9 | X | | | | | Fixed | Memory indirect near call may corrupt EIP |
| 10 | X | | | | | Fixed | L2 single-bit correctable error may cancel simultaneous valid data |
| 11 | X | | | | | Fixed | Page split access before write to CR3 may cause hang |
| 12 | X | | | | | Fixed | Active A20M# during SMM dump |
| 13 | X | | | | | Fixed | Split access across 4-Kbyte page boundary may cause hang |
| 14 | X | X | X | X | X | NoFix | Checker BIST failure in FRC mode not signaled |
| 15 | X | X | X | X | X | NoFix | BINIT# assertion causes FRCERR assertion in FRC mode |
| 16 | X | X | X | X | | Fixed | Extra page fault may occur on IRET during task switch |
| 17 | X | | | | | Fixed | Some caching models in SMM may cause shutdown |
| 18 | X | | | | | Fixed | Fast Strings feature re-enabled after INIT event |
| 19 | X | | | | | Fixed | THERMTRIP# feature not present |
| 20 | X | X | X | X | | Fixed | OUT instruction, Branch Trace Message may write incorrect data |
| 21 | | X | X | X | | Fixed | THERMTRIP# pin not asserted for catastrophic thermal condition |
| 22 | X | X | X | X | | Fixed | LBER value may not be updated correctly |
| 23 | X | X | X | X | X | NoFix | BTM for SMI will contain incorrect FROM EIP |
| 24 | X | X | X | X | X | NoFix | Task switch fault may allow read access of linear address 0h |
| 25 | X | | | | | Fixed | Low frequencies with 5:2 core to bus clock ratio may fail in FRC |
| 26 | X | X | X | X | | Fixed | RDPMC cannot be used in conjunction with SMM |
| 27 | X | X | X | X | | Fixed | PWRGOOD forced to 0 during boundary scan resets TAP |
| 28 | X | X | X | X | | Fixed | BIST failure not indicated when RUNBIST TAP command used |
| 29 | X | X | X | X | | Fixed | INVLPG may not invalidate targeted ITLB entry |
| 30 | X | X | X | X | | Fixed | SMI does not flush TLB entries with PGE enabled |
| 31 | X | X | X | X | X | NoFix | I/O restart in SMM may fail after simultaneous MCE |
| 32 | X | X | X | X | | Fixed | SMBASE reset on INIT# or INIT_IPI |
| 33 | X | X | X | X | X | NoFix | MCE due to L2 ECC error gives L1 MCACOD.LL |
| 34 | X | X | X | X | | Fixed | INVLPG does not invalidate entire 0 to 4Mbyte region |
| 35 | X | | | | | Fixed | BINIT# assertion during snoop hit may cause double Machine Check Exception |
| 36 | X | X | X | X | X | NoFix | Machine Check Exception handler may not always execute successfully |
| 37 | X | X | X | X | X | NoFix | Double ECC error on read may result in BINIT# |
| 38 | X | X | X | X | | Fixed | RSM cannot return to HALT or SHUTDOWN in 32-bit OS |
| 39 | X | X | X | | | Fixed | Accesses of Modified data may hang system |
| 40 | X | X | X | X | X | NoFix | Branch traps do not function if BTMs are also enabled |
| 41 | X | X | X | | | Fixed | Cache line may exist in two different ways in MP system |

| NO. | B0 | C0 | sA0 | sA1 | sB1 | Plans | ERRATA |
|-----|----|----|-----|-----|-----|-------|--------|
| 42 | X | X | X | X | X | Fixed | HALT, SHUTDOWN, and STPCLK special cycles not issued |
| 43 | X | X | X | X | | Fixed | L2 cache may incorrectly report BIST failure |
| 44 | X | X | X | X | X | NoFix | LOCK prefix does not generate trap with some instructions |
| 45 | X | X | X | X | | Fixed | FRCERR may be incorrectly asserted on explicit writeback cycles |
| 46 | X | X | X | X | X | NoFix | FP inexact-result exception flag may not be set |
| 47 | X | X | X | X | | Fixed | Snoop hit during RSM segment load may hang system |
| 48 | X | X | X | X | | Fixed | Mbyte-aligned SMM base address causes SHUTDOWN |
| 49 | X | X | X | X | X | NoFix | Machine Check Exception with coexisting large and small pages |
| 50 | X | X | X | X | | Fixed | WBINVD to some modified addresses hangs system |
| 51 | X | X | X | X | | Fixed | UC store may be reordered around WC transaction |
| 52 | X | X | X | X | | Fixed | IFU error with MCEs, self- or cross-modifying code |
| 53 | X | X | X | X | X | NoFix | Extended TRDY# may cause hang with DBSY# asserted |
| 54 | X | X | X | X | X | NoFix | Call gates can cause incorrect handling of stack size attribute |
| 55 | X | X | X | X | X | NoFix | LBER may be corrupted after some events |
| 56 | | | | | X | NoFix | SMI during REP string instruction clears CR4 |
| 57 | X | X | X | X | X | NoFix | BTMs may be corrupted during simultaneous L1 cache line replacement |
| 58 | X | X | X | X | X | NoFix | Spurious Machine Check Exception Via IFU Data Parity Error |
| 59 | X | X | X | X | X | NoFix | Loss of Inclusion In IFU Can Cause Machine Check Exception |
| 1AP | X | X | X | X | X | NoFix | APIC access to cacheable memory causes SHUTDOWN |
| 2AP | X | X | X | X | X | NoFix | Possible hang due to catastrophic errors during BSP determination |
| 3AP | X | X | X | X | | Fixed | INIT_IPI after STARTUP_IPI-STARTUP_IPI sequence may cause AP to execute at 0h |
| 4AP | | X | X | X | | Fixed | Small window for system hang after INIT# or INIT_IPI |
| 5AP | X | X | X | X | | Fixed | Virtual Wire mode through local APIC may cause Int 15 |
| 6AP | X | X | X | X | | Fixed | Divide By Zero error on LINT0 glitch |
| 7AP | X | X | X | X | X | NoFix | Write to mask LVT (programmed as EXTINT) will not deassert outstanding interrupt |
| 8AP | X | X | X | X | X | NoFix | APs do not respond to a STARTUP_IPI after an INIT# or INIT_IPI in Low Power Mode |
| 9AP | | | | | X | NoFix | Serial bus interrupts may be lost in APIC mixed mode |

| NO. | B0 | C0 | sA0 | sA1 | sB1 | Plans | SPECIFICATION CLARIFICATIONS |
|-----|----|----|-----|-----|-----|-------|------------------------------|
| 1 | X | X | X | X | X | Doc | Signal edge rate for 3.3V tolerant, APIC, and JTAG signals |
| 2 | X | X | X | X | X | Doc | OUTS instruction and subsequent instructions |
| 3 | X | X | X | X | X | Doc | Interrupt recognition determines priority |

| NO. | B0 | C0 | sA0 | sA1 | sB1 | Plans | SPECIFICATION CLARIFICATIONS |
|-----|----|----|-----|-----|-----|-------|------------------------------|
| 4 | X | X | X | X | X | Doc | References to 2-Mbyte pages should include 4-Mbyte pages |
| 5 | X | X | X | X | X | Doc | Modification of reserved areas in the SMRAM saved state map |
| 6 | X | X | X | X | X | Doc | FRCERR asserted for FRC checker BIST failure |
| 7 | X | X | X | X | X | Doc | Deassertion of BREQn# during RESET# |
| 8 | X | X | X | X | X | Doc | TLB flush necessary after PDPE change |
| 9 | X | X | X | X | X | Doc | LOCK# deasserted between locked sequences |
| 10 | X | X | X | X | X | Doc | EXF4# assertion in SMM |
| 11 | X | X | X | X | X | Doc | Preventing caching in the Pentium® Pro processor |
| 12 | X | X | X | X | X | Doc | MCi_CTL registers not cleared by INIT# |
| 13 | X | X | X | X | X | Doc | Stack use during interrupts, exceptions, and CALLs |
| 14 | X | X | X | X | X | Doc | Performance-Monitoring counter issues |
| 15 | X | X | X | X | X | Doc | Clock Jitter Measurement Clarification |

| NO. | B0 | C0 | sA0 | sA1 | sB1 | Plans | DOCUMENTATION CHANGES |
|-----|----|----|-----|-----|-----|-------|-----------------------|
| 1 | X | X | X | X | X | Doc | PE bit number in FPU status word, Volume 2, page 7-50 |
| 2 | X | X | X | X | X | Doc | GTL+ edge rate range incorrect |
| 3 | X | X | X | X | X | Doc | Power-on configuration register documented inconsistently |
| 4 | X | X | X | X | X | Doc | GTL+ minimum overshoot specification |
| 5 | X | X | X | X | X | Doc | Conditional move opcode incorrect |
| 6 | X | X | X | X | X | Doc | 32 entries for 4-Kbyte pages in ITLB |
| 7 | X | X | X | X | X | Doc | A5# value should be H for Arb Id 1 |
| 8 | X | X | X | X | X | Doc | Register names incorrect for CPUID return value |
| 9 | X | X | X | X | X | Doc | Corrections to *Volume 2: Programmer's Reference Manual* |
| 10 | X | X | X | X | X | Doc | Remapping WB, WT memory types |
| 11 | X | X | X | X | X | Doc | BTM "From" and "To" fields reversed |
| 12 | X | X | X | X | X | Doc | Numeric Underflow Exception Description Clarification |
| 13 | X | X | X | X | X | Doc | Page Directory Pointer Table Register Present Bit Identification |

# SPECIFICATION CHANGES

The Specification Changes listed in this section apply to the *Pentium® Pro Family Developer's Manual*, Volume 1 (Order Number 242690), or the *Pentium® Pro Processor at 150 MHz, 166 MHz, 180 MHz, and 200 MHz* datasheet (Order Number 242769). All Specification Changes will be incorporated into future versions of the appropriate document(s).

## 1. Mixing Steppings in MP Systems

Though Intel recommends using identical steppings of processor silicon in multiprocessor systems whenever possible (as this is the only configuration which receives full validation across all of Intel's testing), Intel supports mixing processor steppings, and does not actively prevent various steppings of the Pentium Pro processor from working together in MP systems. However, since Intel cannot validate every possible combination of devices, each new stepping of a device is fully validated only against the latest steppings of other processors and chipset components.

The following list and matrix explain the known issues with mixing steppings:

- While Intel has done nothing to prevent different frequency Pentium Pro processors within a system from working together, there may be uncharacterized errata which exist in such configurations. In mixed stepping systems, all processors must be run at an identical frequency (i.e. the lowest frequency acceptable to all components).

- The workarounds for various errata must take all processors into account.

- Errata for all processor steppings present in a system will affect that system, unless worked around.

- The following notes apply only to specific combinations:

    1. The 150-MHz and the 166-, 180-, and 200-MHz parts vary in their GTL+ $T_{CO\_MIN}$ and $T_{HOLD}$ timings by 250 ps. This may affect some system designs. Also, a customer may wish to use a higher frequency replacement for a 150-MHz processor in a 150-MHz system; this should also be considered.

    2. OverDrive® processor upgrades for the Pentium Pro processor will only be supported in one and two processor systems, and are only supported in same stepping configurations.

    3. Errata numbers 39 and 41 may be problematic for 3 or 4-way MP systems which use processor steppings previous to the sA1 stepping.

In the following table, "NI" implies that there are currently no known issues associated with mixing these steppings. An "X" implies that these steppings should not be used together in a system. A number indicates a known issue, and refers to the numbered list above. There are no currently known issues associated with mixing cache sizes.

| Pentium® Pro Processor Stepping | 150-MHz B0 | 150-MHz C0 | 166-, 180-, 200-MHz sA0 | 166-, 180-, 200-MHz sA1 | 166-, 180-, 200-MHz sB1 | OverDrive® Processor |
|---|---|---|---|---|---|---|
| 150-MHz B0 | 3 | 3 | 1, 3 | 1, 3 | 1, 3 | X |
| 150-MHz C0 | 3 | 3 | 1, 3 | 1, 3 | 1, 3 | X |
| 166-, 180-, 200-MHz sA0 | 1, 3 | 1, 3 | 3 | 3 | 3 | X |
| 166-, 180-, 200-MHz sA1 | 1, 3 | 1, 3 | 3 | NI | NI | X |
| 166-, 180-, 200-MHz sB1 | 1, 3 | 1, 3 | 3 | NI | NI | X |
| OverDrive Processor | X | X | X | X | X | 2 |

## 2. Change in PICD[1:0] Hold Time and Valid Delay

The minimum value of the hold time for the PICD[1:0] signals, T28, will be changed from 2 ns to 2.5 ns. The minimum value of the valid delay for these signals, T29, will be changed from 2.1 ns to 2.0 ns. These values change those documented in Table 11-15 of the *Pentium® Pro Family Developer's Manual, Volume 1: Specifications*.

## 3. Undershoot Specification for Some GTL+ Signals

The *Pentium® Pro Family Developer's Manual, Volume 1: Specifications*, Section 11.15., will include the following undershoot specification, to simplify signal integrity issues on signals that can be driven by more than one device simultaneously:

The Pentium Pro processor bus signals AERR#, BERR#, BINIT#, BNR#, HIT#, and HITM# (only) are capable of sinking an 85 mA current pulse at 2.4% average time duty cycle. This is equivalent to -1.7V applied to a 2Ω source in series with the device pin for 8 ns at 66 MHz with a utilization of 5%.



Note: This test covers the AC operating conditions only.
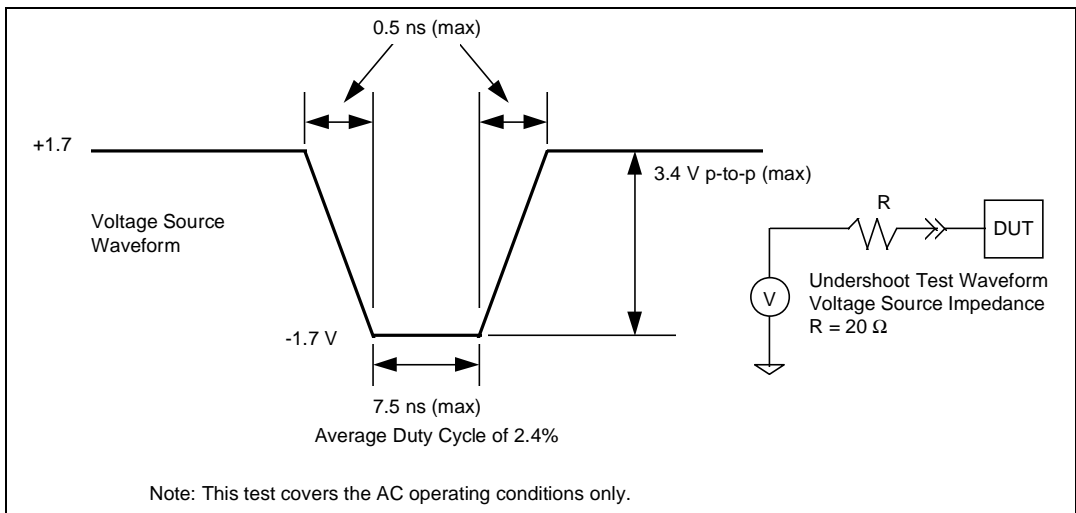
## 4. OverDrive® VRM Motherboard Bulk Capacitance Specification

There is a limit to the amount of bulk capacitance which may be moved to the motherboard and still allow the system to be upgraded with an OverDrive processor and OverDrive VRM. In the *Pentium® Pro Family Developer's Manual, Volume 1: Specifications*, Section 17.4.1.2., *OverDrive VRM D.C. Specifications*, the following entry will be added to Table 34.

| Symbol | Parameter | Min | Max | Unit | Notes |
|--------|-----------|-----|-----|------|-------|
| Cmb | Total bulk capacitance between Header 8 and Socket 8 | | 9000 | µF | |

## 5. MTRR Precedences and Memory Type Overriding

Part 2. of Section 11.11.4.1. of the *Pentium® Pro Family Developer's Manual, Volume 3: Operating System Writer's Guide* will change as follows:

a. If one variable memory range matches, the processor uses the memory type stored in the MTRRphysBase*n* register for that range.

b. If two or more variable memory ranges match and the memory types are identical, then that memory type is used.

c. If two or more variable memory ranges match and one of the memory types is UC, the UC memory type is used.

d. If two or more variable memory ranges match and the memory types are WT and WB, the WT memory type is used.

e. If two or more variable memory ranges match and the memory types are not combinations as specified in parts a. through d. above, the behavior of the processor is undefined.

## 6. V$_{CC}$S Pins Removed From Specification

The Pentium Pro family will not use the V$_{CC}$S pins. All references to these pins will be removed from the specification. These references currently appear in the *Pentium® Pro Family Developer's Manual: Specifications* Sections 11.3., 11.4.1., 11.8.1., 11.12., 11.13, and 17.4. The references to V$_{CC}$S in Section 15.2., *Pinout*, will be changed to Reserved. However, connection to 3.3V is still allowed for these signals.

## 7. Locks Across Cache Line Boundary Disable Bit Added

As of the sB1 stepping of the Pentium Pro processor, setting bit 31 of the Model Specific Register (MSR) at address 33h to '1' will prevent LOCK# from being asserted when locked transactions which are split across a cache line boundary are issued from the processor. This bit is disabled by default and remains Reserved for all previous steppings of the Pentium Pro processor. In the default state ('0'), unaligned data issued in a locked sequence by the processor will have atomicity with the LOCK# signal asserted. When the bit is set, any transactions issued which split a cache line boundary will not have the LOCK# signal asserted, and no atomicity can be guaranteed between the reads and writes in the sequence. Locked sequences which do not split a cache line boundary will still follow the normal LOCK# protocol with this bit set.

## 8. Instruction Streaming Buffer Disable Bit Added

As of the sB1 stepping of the processor, bit 30 of the Model Specific Register (MSR) at address 33h can be set to '1' to disable Instruction Streaming Buffers. [Note that on the sB1 stepping of the Pentium Pro processor, this bit can be used to work around errata 58 and 59.] By default, the bit is '0', and Instruction Streaming Buffers are enabled. The bit is Reserved for all previous steppings of the processor and should not be written. On steppings prior to the sB1, attempts to write to this bit will be silently ignored by the processor.

# ERRATA

## 1. Mixed Cacheability of Lock Variables Is Problematic in MP Systems

**PROBLEM:** This errata only affects multiprocessor systems where a lock variable address is marked cacheable in one processor and uncacheable in any others. The processors which have it marked uncacheable may stall indefinitely when accessing the lock variable. The stall is only encountered if:

- One processor has the lock variable cached, and is attempting to execute a cache lock.
- If the processor which has that address cached has it cached in the L2 only.
- Several other processors, meanwhile, issue back to back accesses to that same address on the bus.

**IMPLICATION:** MP systems where all processors either use cache locks or consistent locks to uncacheable space will not encounter this problem. If, however, a lock variable's cacheability varies in different processors, and several processors are all attempting to perform the lock simultaneously, an indefinite stall may be experienced by the processors which have it marked uncacheable in locking the variable (if the conditions above are satisfied). Intel has only encountered this problem in focus testing with artificially generated external events. Intel has not currently identified any commercial software which exhibits this problem.

**WORKAROUND:** Follow a homogenous model for the memory type range registers (MTRRs), ensuring that all processors have the same cacheability attributes for each region of memory; do not use locks whose memory type is cacheable on one processor, and uncacheable on others. Avoid page table aliasing, which may produce a non-homogenous memory model.

## 2. FP Operand Pointer May Be Incorrectly Calculated After FP Access Which Wraps 64-Kbyte Boundary in 16-Bit Code

**PROBLEM:** The FP Operand Pointer is the effective address of the operand associated with the last non-control floating-point instruction executed by the machine. If an 80-bit floating-point access (load or store) occurs in a 16-bit mode other than protected mode (in which case the access will produce a segment limit violation), accesses memory which wraps a 64-Kbyte boundary, and the floating-point environment is subsequently saved in 32-bit mode, the subtraction routine used to calculate the FP Operand Pointer will assume the floating-point access was in 32-bit mode, and the high word of the address will be FFFFh instead of 0000h.

**IMPLICATION:** A 32-bit operating system running 16-bit floating-point code may encounter this erratum, under the following conditions:

- The operating system is using a segment greater than 64 Kbytes in size.
- An application is running in a 16-bit mode other than protected mode.
- An 80-bit floating-point load which wraps the 64-Kbyte boundary is executed.
- The operating system uses a 32-bit handler on an unmasked exception which occurs during the load.
- The exception handler uses the value contained in the FP Operand Pointer.

Wrapping an 80-bit floating-point load around a segment boundary in this way is not a normal or useful programming practice. Intel has not currently identified any software which exhibits this behavior.

**WORKAROUND:** If the FP Operand Pointer is used in a 32-bit exception handler in an OS which may run 16-bit floating-point code, care must be taken to ensure that no 80-bit floating-point accesses are wrapped around a 64-Kbyte boundary.

## 3.    Differences Exist in Debug Exception Reporting

**PROBLEM:**  There exist some differences in the reporting of code and data breakpoint matches between that specified by previous Intel processors' specifications and the behavior of the Pentium Pro processor, as described below:

### CASE 1:

The first case is for a breakpoint set on a MOVSS or POPSS instruction, when the instruction following it causes a debug register protection fault (DR7.gd is already set, enabling the fault). The Pentium processor reports delayed data breakpoint matches from the MOVSS or POPSS instructions by setting the matching DR6.bi bits, along with the debug register protection fault (DR6.bd). If additional breakpoint faults are matched during the call of the debug fault handler, the Pentium processor sets the breakpoint match bits (DR6.bi) to reflect the breakpoints matched by both the MOVSS or POPSS breakpoint and the debug fault handler call. The Pentium Pro processor only sets DR6.bd in either situation, and does not set any of the DR6.bi bits.

### CASE 2:

In the second breakpoint reporting failure case, if a MOVSS or POPSS instruction with a data breakpoint is followed by a store to memory which crosses a 4-Kbyte page boundary, the breakpoint information for the MOVSS or POPSS will be lost. Previous processors retain this information across such a page split.

### CASE 3:

If they occur after a MOVSS or POPSS instruction, the INT*n*, INTO, and INT3 instructions zero the DR6.Bi bits (bits B0 through B3), clearing pending breakpoint information, unlike previous processors.

### CASE 4:

If a data breakpoint and an SMI (System Management Interrupt) occur simultaneously, the SMI will be serviced via a call to the SMM handler, and the pending breakpoint will be lost.

**IMPLICATION:**  When debugging or when developing debuggers for a Pentium Pro processor system, this behavior should be noted. Normal usage of the MOVSS or POPSS instructions (i.e. following them with a MOV ESP) will not exhibit the behavior of cases 1-3. Debugging in conjunction with SMM will be limited by case 4 (no workaround has been identified for this case).

**WORKAROUND:**  Following MOVSS and POPSS instructions with a MOV ESP instruction when using breakpoints will avoid the first three cases of this erratum.

## 4.    FLUSH# Servicing Delayed While Waiting for STARTUP_IPI in MP Systems

**PROBLEM:**  In a multiprocessor system, if an application processor is waiting for a startup interprocessor interrupt (STARTUP_IPI), then it will not service a FLUSH# pin assertion until it has received the STARTUP_IPI.

**IMPLICATION:**  After the MP initialization protocol, only one processor becomes the bootstrap processor (BSP). All others become slave application processors (APs). After losing the BSP arbitration, APs go into a wait loop waiting for a STARTUP_IPI.

The BSP can wake up an AP to perform some tasks with a STARTUP_IPI, and then put one or more APs back to sleep with an initialization interprocessor interrupt (INIT_IPI, which has the same affect as asserting INIT#), which returns them to a wait loop. The result is a possible loss of cache coherency if the offline processor is intended to service a FLUSH# assertion at this point. The FLUSH# will be serviced as soon as the processor is awakened by

a STARTUP_IPI, before any other instructions are executed. Intel has not encountered any operating systems that are affected by this problem.

**WORKAROUND:** Operating system developers should take care to execute a WBINVD instruction before an AP is taken offline using an INIT_IPI.

## 5. Fast Strings REP MOVS May Not Transfer All Data

**PROBLEM:** If the fast strings feature is enabled (bit 2 of MSR 1E0h is 0), a REP MOVS instruction may incorrectly use a pre-decremented counter, thereby skipping a cache line of the transfer. This will only occur if paging is enabled.

**IMPLICATION:** A REP MOVS instruction executed with this feature enabled may not transfer all requested data, leading to stale data being left at the destination.

**WORKAROUND:** For steppings affected by this erratum, the fast strings feature must be disabled in the BIOS, by setting bit 2 of the machine specific register (MSR) at address 1E0h to '1'.

## 6. Page Table Base Change During Task Switch Using Mode C Paging May Corrupt EIP

**PROBLEM:** While using mode C paging (a 36-bit address extension which uses 2-Mbyte pages), if data is written to CR3 during a task switch, the EIP will be corrupted, resulting in unpredictable failure. CR3, which contains the page table base, will be written during a task switch if the switch is moving to a new paging environment, or if the TLB entries must be cleared.

**IMPLICATION:** An operating system which uses a hardware tasking model (via the task switch segment, or TSS) along with the 36-bit extension mode C paging, may fail, most likely with an invalid opcode exception. Intel has not identified any operating systems which use this combination of features.

**WORKAROUND:** When developing an operating system which uses mode C paging, do not use hardware task switching, or else only perform hardware task switches which do not change the page table base register, CR3.

## 7. Code Fetch Matching Disabled Debug Register May Cause Debug Exception

**PROBLEM:** The bits L0-3 and G0-3 enable breakpoints local to a task and global to all tasks, respectively. If one of these bits are set, a breakpoint is enabled, corresponding to the addresses in the debug registers DR0-DR3. If at least one of these breakpoints are enabled, any of these registers are *disabled* (i.e. L$n$ and G$n$ are 0), and RW$n$ for the disabled register is 00 (indicating a breakpoint on instruction execution), normally no instruction fetch will cause an instruction-breakpoint fault based on a match with the address in the disabled register(s). However, if the address in a disabled register matches the address of a code fetch which also results in a page fault, an instruction-breakpoint fault will occur.

**IMPLICATION:** While debugging software, extraneous instruction-breakpoint faults may be encountered if breakpoint registers are not cleared when they are disabled. Debug software which does not implement a code breakpoint handler will fail, if this occurs. If a handler is present, the fault will be serviced. Mixing data and code may exacerbate this problem by allowing disabled data breakpoint registers to break on an instruction fetch.

**WORKAROUND:** The debug handler should clear breakpoint registers which become disabled.

## 8. Mode C Paging in SMM Causes Use of Incorrect Page Tables

**PROBLEM:** If mode C paging (a 36-bit address extension which uses 2-Mbyte pages) is in use, an SMI is serviced, and the SMM handler switches to mode C paging with a different set of page tables, then upon returning from the SMM handler via an RSM instruction, the processor will use the wrong page tables for address translation. Since paging is not, in general, supported in SMM, this is not a violation of the Pentium Pro processor specification, and has been removed as an erratum, but has been included here for continuity.

**IMPLICATION:** Using mode C paging both in normal operation and in System Management Mode (SMM) may cause the system to hang. Intel has not currently identified any software which is affected by this erratum. This is not a violation of the Pentium Pro processor specification; large size pages are not supported in SMM.

**WORKAROUND:** Do not use mode C paging for the SMM handler code.

## 9. Memory Indirect Near Call May Corrupt EIP

**PROBLEM:** Under some conditions, a near indirect call (opcode FF /2) may cause corruption of the EIP after partial completion of the instruction's execution. This can occur if an access due to the near indirect call:

- Accesses memory which crosses a page boundary.
- Results in a misaligned data breakpoint exception.
- Is the first access of the page containing the data used for the call.
- Results in a page fault which, upon walking the page tables (see Erratum #13), is found not to be a faulting condition.

**IMPLICATION:** The most likely failure mechanism for this erratum is an invalid opcode exception.

**WORKAROUND:** It is possible for BIOS code to contain a workaround for this erratum.

## 10. L2 Single-Bit Correctable Error May Cancel Simultaneous Valid Data

**PROBLEM:** If a single-bit correctable error is detected by the Pentium Pro processor on a transaction between the data cache unit (DCU) of the CPU and the L2 cache, and there is a second transaction pending which caches data from the Pentium Pro processor bus, the Pentium Pro processor will cancel both transactions instead of just the transaction on which the error occurred.

**IMPLICATION:** The canceled transaction will never complete under these conditions. Single-bit errors which encounter this boundary condition will cause the Pentium Pro processor to hang. However, since this erratum only affects error recovery, normal, hardware error-free execution will not be affected.

**WORKAROUND:** None identified.

## 11. Page Split Access Before Write to CR3 May Cause Hang

**PROBLEM:** If paging is enabled, and a memory access occurs which crosses a page boundary, a future write to CR3 may cause the processor to hang. The write to CR3 must occur at a specific time interval away from the page split access; in focus-testing, the only test which encountered this hang in mode A or B paging had 12 NOP instructions between the split access and the write to CR3. When using mode C paging (2-Mbyte pages), the probability of encountering this erratum increased somewhat.

**IMPLICATION:**  Using paging in conjunction with writes to CR3 (to invalidate the page tables or change to a new paging environment) may cause the system to hang. There is a small, timing dependent window of opportunity to encounter this erratum, which is wider if mode C paging is used.

**WORKAROUND:**  Some instructions can be inserted between the memory access and the write to CR3 to prevent this erratum, such as:

- CPUID,
- STD,
- CLD, or
- CLI/STI.

Though the following instructions between the memory access and the write to CR3 prevent the erratum, they may not be suitable for all situations as workarounds for this erratum due to their effect or dependency on system state:

- WRMSR,
- RDMSR,
- RDPMC,
- LDS, LFS, LGS, LSS, or LES,
- MOV CR0, reg,
- MOV CR2, reg,
- MOV CR4, reg,
- MOV DS, reg,
- MOV SS, reg,
- MOV ES, reg,
- MOV GS, reg,
- MOV FS, reg,
- A far transfer, interrupt, or software interrupt.

Any of these which occur between the memory access which crosses a 4-Kbyte boundary and the write to CR3 will prevent the conditions necessary for this erratum to occur. Due to its long latency, a non-string OUT instruction will also prevent the erratum.

## 12.  *Active A20M# During SMM Dump*

**PROBLEM:**  If the A20M# signal (mask A20) is active when the SMM handler is saving CPU state information to the SMM dump space prior to executing the handler code, the state information will be partially saved at an address with the A20 pin masked, and partially with A20 unmasked.

**IMPLICATION:**  If the SMM dump space uses addresses in an odd megabyte (A20 high), the stored state will be split among two different address spaces and the system will not recover state correctly after an RSM instruction.

**WORKAROUND:**  Either ensure that the SMM dump space does not coincide with any odd-megabyte addresses (under 1 Mbyte is acceptable), or use external logic to prevent the servicing of an SMI until A20M# is deasserted.

## 13. Split Access Across 4-Kbyte Page Boundary May Cause Hang

**PROBLEM:** When a load or store access is split across a page boundary, a page fault may be detected. If a page fault is signaled during such an access, the Pentium Pro processor will walk the page table to ensure that the page is not being made accessible by another agent (such as another processor in an MP system, or a DMA controller) before branching to the page fault handler. If the split access signals a page fault, and the page which caused the fault is then found to be non-faulting during the page table walk, the access will be executed incorrectly, resulting in a hang.

**IMPLICATION:** When multiple agents are capable of changing the page table, a load or store which is split across a page boundary may cause the machine to hang.

**WORKAROUND:** It is possible for BIOS code to contain a workaround for this erratum.

## 14. Checker BIST Failure in FRC Mode Not Signaled

**PROBLEM:** If a system is running in functional redundancy checking (FRC) mode, and the checker of the master-checker pair encounters a hard failure while running the built-in self test (BIST), the checker will tri-state all outputs without signaling an IERR#.

**IMPLICATION:** Assuming the master passes BIST successfully, it will continue execution unchecked, operating without functional redundancy. However, the necessary pull-up on the FRCERR pin will cause an FRCERR to be signaled. The operation of the master depends on the implementation of FRCERR.

**WORKAROUND:** For successful detection of BIST failure in the checker of an FRC pair, the FRCERR signal must be used, instead of IERR#.

## 15. BINIT# Assertion Causes FRCERR Assertion in FRC Mode

**PROBLEM:** If a pair of Pentium Pro processors are running in functional redundancy checking (FRC) mode, and a catastrophic error condition causes BINIT# to be asserted, the checker in the master-checker pair will enter shutdown. The next bus transaction from the master will then result in the assertion of FRCERR.

**IMPLICATION:** Bus initialization via an assertion of BINIT# occurs as the result of a catastrophic error condition which precludes the continuing reliable execution of the system. Under normal circumstances, the master-checker pair would remain synchronized in the execution of the BINIT# handler. However, due to this erratum, an FRCERR will be signaled. System behavior then depends on the implementation of FRCERR.

**WORKAROUND:** None identified.

## 16. Extra Page Fault May Occur on IRET During Task Switch

**PROBLEM:** During a hardware task switch (using the task switch segment, or TSS) which jumps to the new task with an IRET which causes a page fault, a second page fault exception may be generated upon completion of the page miss handler routine, prompting the Pentium Pro processor to execute the page miss handler a second time.

**IMPLICATION:** Intel has identified no adverse implications of this anomaly other than the slight performance loss from executing the page miss handler twice after a task switch which causes a page fault.

**WORKAROUND:** None identified.

## 17.    Some Caching Models in SMM May Cause Shutdown

**PROBLEM:**  A condition exists in the Pentium Pro processor such that upon returning from the SMM handler (via an RSM instruction), a read may be executed too soon for the data to be successfully loaded. This condition occurs whenever the load for the read is delayed because the cache line which the load accesses is not present in the cache. This may occur if:

- The SMM handler is too large to fit into the cache.
- A WBINVD instruction was executed upon servicing the SMI.
- The SMM handler is being run in memory mapped with the UC (uncacheable) memory type.

When the RSM instruction is executed under any of these conditions, the first memory access after the RSM will cause multiple stack faults, and the Pentium Pro processor will enter shutdown.

**IMPLICATION:**  Under many operating conditions, returning from SMM will result in the CPU entering shutdown.

**WORKAROUND:**  It is possible for BIOS code to contain a workaround for this erratum.

## 18.    Fast Strings Feature Re-enabled After INIT Event

**PROBLEM:**  If the Fast Strings feature has been disabled in the machine specific registers (MSRs) by clearing the appropriate bit (to '0'), after an INIT or an initialization interprocessor interrupt (INIT_IPI), the bit will be set (to '1'), and the feature will be re-enabled.

**IMPLICATION:**  After each INIT or INIT_IPI event, this feature will be re-enabled, exposing the system to the effects of Erratum #5.

**WORKAROUND:**  It is possible for BIOS code to contain a workaround for this erratum.

## 19.    THERMTRIP# Feature Not Present

**PROBLEM:**  The THERMTRIP# feature documented for the Pentium Pro processor is not present.

**IMPLICATION:**  The Pentium Pro processor will not shut down upon reaching a catastrophic thermal condition as defined in the *Pentium® Pro Processor Developer's Manual, Volume 1*.

**WORKAROUND:**  System manufacturers can implement a thermal management solution which does not rely upon THERMTRIP#.

## 20.    OUT Instruction, Branch Trace Message May Write Incorrect Data

**PROBLEM:**  A data cache load may be blocked by internal conditions, and speculatively awakened by the Pentium Pro processor. If this speculative wakeup occurs at the same time that an OUT instruction or a branch trace message (BTM) is dispatched, and the load access crosses a cache line boundary, the buffer which contains the data for the OUT or BTM will be corrupted.

**IMPLICATION:**  Some OUT instructions and BTMs may propagate corrupted data.

**WORKAROUND:**  It is possible for BIOS code to contain a workaround for this erratum.

## 21. THERMTRIP# Pin Not Asserted For Catastrophic Thermal Condition

**PROBLEM:** If a catastrophic thermal condition is reached as specified in the *Pentium® Pro Processor Developer's Manual, Volume 1,* the processor will correctly shut down until a hard reset is performed. However, the THERMTRIP# pin, which should be asserted after the processor shuts down, will remain unasserted.

**IMPLICATION:** If the processor shuts down due to a catastrophic thermal condition, there will be no electrical indication of the reason for the shutdown, since the THERMTRIP# pin will not be asserted.

**WORKAROUND:** An external temperature sensor should be implemented if a system design requires notification that a catastrophic thermal condition has caused the system to shut down.

## 22. LBER Value May Not Be Updated Correctly

**PROBLEM:** The last branch record (LBR) and the last branch before exception record (LBER) can be used to determine the source and destination information for previous branches or exceptions. The LBR contains the source and destination addresses for the last branch or exception, and the LBER contains similar information for the last branch taken before the last exception. This information is typically used to determine the location of a branch which leads to execution of code which causes an exception. However, after executing a trap under some conditions, the LBER may not be updated after a branch, or may be updated incorrectly.

**IMPLICATION:** The LBER and LBR registers are used only for debugging purposes. When this erratum occurs, the LBER will not contain reliable address information. The value of LBER should be used with caution when debugging branching code; if the values in the LBR and LBER are the same, the value of LBER is 0, or the value of the LBER is the same before and after the trap, then the LBER value is incorrect.

**WORKAROUND:** None identified.

## 23. BTM for SMI Will Contain Incorrect FROM EIP

**PROBLEM:** A system management interrupt (SMI) will produce a branch trace message (BTM), if BTMs are enabled. However, the FROM EIP field of the BTM (used to determine the address of the instruction which was being executed when the SMI was serviced) will not have been updated for the SMI, so the field will report the same FROM EIP as the previous BTM.

**IMPLICATION:** A BTM which is issued for an SMI will not contain the correct FROM EIP, limiting the usefulness of BTMs for debugging software in conjunction with System Management Mode (SMM).

**WORKAROUND:** None identified.

## 24. Task Switch Fault May Allow Read Access of Linear Address 0h

**PROBLEM:** For this erratum to occur, the following conditions must be satisfied:

- A Task State Segment (TSS) must be used to handle task switches.
- A task switch fault must occur while loading the new segment descriptors for a task switch.
- The resulting task switch fault handler call must not result in a task switch.
- The task switch fault handler must be executed successfully and return to the task switch using a far return or an IRET instruction.

Upon the coincidence of these conditions, any of the data segment registers which were not loaded by the task switch will permit a read access to a single byte at linear address 0h.

**IMPLICATION:** Under these circumstances, if the target task is of a privilege level of 1, 2, or 3, it may be allowed to read the byte at linear address 0h, which it may not normally be permitted to read. Intel has not currently identified any software which could cause this behavior.

**WORKAROUND:** When developing an operating system which uses the TSS to process task switches, a task switch fault should also generate a task switch using the TSS.

## 25.     Low Frequencies With 5:2 Core to Bus Clock Ratio May Fail in FRC

**PROBLEM:** If a functional redundancy checking (FRC) system is using a 5:2 core to bus clock ratio, and is running at frequencies of 120/48 MHz or lower, the phase-locked loops (PLL) of the master-checker pair may not function correctly, resulting in the assertion of FRCERR with the first request to the Pentium Pro processor bus. If the first request does not result in the assertion of FRCERR, the system will function normally. The minimum core frequency specification in the *Pentium*® *Pro Processor  Developer's Manual, Volume 1* is 100 MHz.

**IMPLICATION:** An FRC system should not be run at core frequencies lower than 120 MHz with a 5:2 core to bus clock ratio. If it is necessary to do so, the system may not always begin executing correctly, but power-cycling the system after such behavior may allow normal operation.

**WORKAROUND:** Run the system at its specified frequency, or power-cycle the system if it is necessary to run an FRC system under these conditions.

## 26.     RDPMC Cannot Be Used in Conjunction With SMM

**PROBLEM:** If the performance counter enable (PCE) bit (bit 8 of CR4) is enabled (set to '1'), and if an RSM instruction is executed, the processor will enter a shutdown state, believing that reserved space has been altered.

**IMPLICATION:** The PCE bit is set by an operating system to enable user code to issue an RDPMC (Read Performance Monitor Counter) instruction. In systems which support SMM, this bit should not be set, and this instruction will not be available. However, the performance monitoring capabilities of the Pentium Pro processor are still available through driver-based applications.

**WORKAROUND:** To prevent the shutdown from occurring, SMM handler code can clear this bit. However, this will disable the ability of users to issue the RDPMC instruction, and any application which attempts to issue this instruction after SMM handler code has been executed will cause a general protection fault, unless the operating system has re-enabled it by setting the PCE bit.

## 27.     PWRGOOD Forced to 0 During Boundary Scan Resets TAP

**PROBLEM:** According to IEEE 1149.1-1990 (JTAG) Specification, rule 5.3.1(b), no component input pin should initialize the test access port (TAP) controller, or any other test logic within the component. However, if the PWRGOOD pin on the Pentium Pro processor is forced to 0, it will reset the TAP Controller.

**IMPLICATION:** If boundary scan testing forces PWRGOOD to 0, boundary scan errors will result.

**WORKAROUND:** Do not force PWRGOOD to 0 during boundary scan testing. The BSDL file or the test vectors can be edited to ensure that this does not occur, by changing the declaration of the PWRGOOD pin from 'input' to 'internal' with a safe value of 1. Note that while this will allow boundary scan tests to be generated correctly, it will not allow the PWRGOOD pin to be tested.

## 28.    BIST Failure Not Indicated When RUNBIST TAP Command Used

**PROBLEM:**  The test access port (TAP) RUNBIST command is intended for use during boundary scan testing, and should run the built-in self test (BIST), storing the result in the BIST result register after completion. A zero result indicates BIST passed, and a nonzero result indicates BIST failure. Currently, if this command is executed from the TAP, a zero result will be stored in the BIST result register regardless of whether the BIST passes or fails.

**IMPLICATION:**  Running the RUNBIST command from the TAP will not indicate a failing condition. This command is not functional for the steppings of the Pentium Pro processor affected by this erratum.

**WORKAROUND:**  None identified. However, BIST may still be run separately from boundary scan, by allowing the INIT# signal to be sampled active on the RESET# signal's active to inactive transition. In this case, passage or failure is indicated in EAX.

## 29.    INVLPG May Not Invalidate Targeted ITLB Entry

**PROBLEM:**  When the INVLPG instruction is used, the entry in the translation look-aside buffer (TLB) which corresponds to the given address should be invalidated, under most circumstances preserving the validity of the rest of the TLB. However, if the variable memory type range registers (MTRRs) are used to map a region of memory as a particular memory type, this may not occur. Specifically, if the variable MTRR specifies a range which is not 4-Mbyte aligned (i.e. the MTRR's mask register contains a value with one or more bits in the range [21:12] nonzero), the INVLPG instruction will invalidate the proper entry in the data TLB (DTLB), but in so doing, may lose the information necessary to invalidate the entry in the instruction TLB (ITLB). It will, instead, invalidate entry 0 in the ITLB, leaving a stale translation in the ITLB. A subsequent code fetch from the addresses in this entry will retrieve and execute incorrect instructions.

**IMPLICATION:**  The INVLPG entry cannot be used on a system with an MTRR range which is not 4Mbyte aligned.

**WORKAROUND:**  BIOS and operating system code should not use a variable MTRR range which is not 4Mbyte aligned. All such ranges should have bits [21:12] of the MTRR's mask register set to '0' for processor steppings affected by this erratum.

## 30.    SMI Does Not Flush TLB Entries With PGE Enabled

**PROBLEM:**  When SMI# is asserted, all entries of the translation lookaside buffer (TLB) should be flushed. However, entries with the page global enable (PGE) bit set will not be flushed from the TLB.

**IMPLICATION:**  The SMM (System Management Mode) handler may access an address which will be incorrectly translated due to the stale entries in the TLB, resulting in unexpected system behavior.

**WORKAROUND:**  If a write to CR3 is executed to clear the TLBs upon entry into SMM, then BIOS code can contain a workaround for this erratum.

## 31.    I/O Restart in SMM May Fail After Simultaneous MCE

**PROBLEM:**  If an I/O instruction (IN, INS, REP INS, OUT, OUTS, or REP OUTS) is being executed, and if the data for this instruction becomes corrupted, the Pentium Pro processor will signal a machine check exception (MCE). If the instruction is directed at a device which is powered down, the processor may also receive an assertion of SMI#. Since MCEs have higher priority, the processor will call the MCE handler, and the SMI# assertion will remain pending. However, upon attempting to execute the first instruction of the MCE handler, the SMI# will be recognized and the processor will attempt to execute the SMM handler. If the SMM handler is completed

successfully, it will attempt to restart the I/O instruction, but will not have the correct machine state, due to the call to the MCE handler.

**IMPLICATION:** A simultaneous MCE and SMI# assertion may occur for one of the I/O instructions above. The SMM handler will attempt to restart such an I/O instruction, but will have corrupted state due to the MCE handler call, leading to failure of the restart and shutdown of the processor.

**WORKAROUND:** If a system implementation must support both SMM and MCEs, the first thing the SMM handler code (when an I/O restart is to be performed) should do is check for a pending MCE (after writing to CR3 as per the workaround for Erratum #30). If there is an MCE pending, the SMM handler should immediately exit via an RSM instruction and allow the MCE handler to execute and restart the I/O instruction. If there is not, the SMM handler may proceed with its normal operation.

## 32.    SMBASE Reset on INIT# or INIT_IPI

**PROBLEM:** In systems which support System Management Mode (SMM), the default value for the SMBASE is 30000h. This value may be changed, particularly in MP systems, where processors' SMM spaces do not overlap. However, when an assertion of the INIT# pin or an initialization interprocessor interrupt (INIT_IPI) is received by a Pentium Pro processor, the value of SMBASE is reset to 30000h.

**IMPLICATION:** An assertion of INIT# or an INIT_IPI will clear any changes made to the SMBASE value.

**WORKAROUND:** It is possible that BIOS code may contain a workaround for this erratum, in systems which contain C0 or subsequent steppings of Pentium Pro processor silicon. No workaround is available for the B0 stepping of the Pentium Pro processor.

## 33.    MCE Due to L2 Cache Error Gives L1 MCACOD.LL

**PROBLEM:** If an error occurs on an access to the Pentium Pro processor's L2 cache, any resulting Machine Check Exception (MCE) will be logged with '01' in the LL field of MCACOD. This value indicates an L1 cache error; the value should be '10'.

**IMPLICATION:** An MCE due to an L2 cache access will be improperly logged as due to an L1 error.

**WORKAROUND:** None identified.

## 34.    INVLPG Does Not Invalidate Entire 0 to 4 -Mbyte Region

**PROBLEM:** When using large paging modes (mode C or mode B paging), a TLB entry may be created for the large page beginning at address 0. This page may cover the 0 to 2 or 4Mbyte region. If the fixed memory type range registers (MTRRs) are used in conjunction with these paging modes, TLB entries for 4Kbyte pages may also be created to map the lower 1-Mbyte region of memory. If the INVLPG instruction is used to invalidate one of these 4-Kbyte pages, the large page which covers the same region will not be invalidated; conversely, if the large page is invalidated using INVLPG (with an address from 1 to 2 or 4 Mbytes), the 4Kbyte entries will not be invalidated.

**IMPLICATION:** Intel has not currently identified any software which is affected by this erratum. However, using INVLPG to invalidate regions from 0 to 1 Mbyte or from 1 to 4 Mbytes may result in the use of stale data if the other part of the 0 to 4-Mbyte range is expected to be invalidated when using 4Mbyte pages (or 0 to 2-Mbyte if the software uses 2-Mbyte pages). Future operating systems may be affected by this erratum, but only if the 0 to 4-Mbyte page is invalidated using INVLPG. Since this page would most likely contain operating system kernel code, it is unlikely that such an operating system would be designed to invalidate this page.

**WORKAROUND:** Operating systems should not set up a 4-Mbyte page at address 0, unless writes to the CR3 register are performed to clear the TLB entries for this region (if the PGE bit in CR4 is set, this bit must be cleared prior to writing to CR3, to clear global pages). This will invalidate the entire TLB. If a 4-Mbyte page is used, and INVLPG instructions are used to invalidate this region, INVLPG instructions must be used repeatedly to invalidate the large page as well as any of the 4-Kbyte ranges which should be invalidated.

## 35. BINIT# Assertion During Snoop Hit May Cause Double Machine Check Exception

**PROBLEM:** If the L1 cache has locked a modified cache line in response to an external snoop, then the assertion of BINIT# between the time ADS# is asserted for the snoop and the time that the implicit writeback data is transferred may cause a double machine check exception (MCE). The double MCE is encountered due to artificial internal parity errors generated by the BINIT# handler during its cleanup routines.

**IMPLICATION:** A BINIT# may cause the processor to shut down in response to the double MCE generated if the above conditions occur. IERR# will be asserted, and no further operations are possible until a hard reset is performed.

**WORKAROUND:** BINIT# is asserted upon detection of a catastrophic system condition. Leaving MCEs disabled will result in the BINIT# causing the CPU to enter shutdown as well. To avoid CPU shutdown, BINIT# observation can be disabled. However, a condition which would result in the generation of BINIT# may cause corrupt data to be propagated, and unexpected system behavior may occur. No workaround which would enable BINIT# to be recognized in this window without causing the CPU to shutdown has been identified.

## 36. Machine Check Exception Handler May Not Always Execute Successfully

**PROBLEM:** An asynchronous machine check exception (MCE), such as a BINIT# event, which occurs during an access that splits a 4-Kbyte page boundary may leave some internal registers in an indeterminate state. Thus, MCE handler code may not always run successfully if an asynchronous MCE has occurred previously.

**IMPLICATION:** An MCE may not always result in the successful execution of the MCE handler, hanging the CPU.

**WORKAROUND:** If MCEs are not enabled, execution of the MCE handler will not occur. However, asynchronous MCEs usually occur upon detection of a catastrophic system condition. Leaving MCEs disabled will result in the condition which caused the asynchronous MCE instead causing the CPU to enter shutdown. Therefore, leaving MCEs disabled may not improve overall system behavior. No workaround which would guarantee successful MCE handler execution under this condition has been identified.

## 37. Double ECC Error on Read May Result in BINIT#

**PROBLEM:** For this erratum to occur, the following conditions must be met:

- Machine check exceptions (MCEs) must be enabled.
- A dataless transaction (such as a write invalidate) must be occurring simultaneously with a transaction which returns data (a normal read).
- The read data must contain a double-bit uncorrectable ECC error.

If these conditions are met, the Pentium Pro processor will not be able to determine which transaction was erroneous, and instead of generating an MCE, it will generate a BINIT#.

**IMPLICATION:** The bus will be reinitialized in this case. However, since a double-bit uncorrectable ECC error occurred on the read, the MCE handler (which is normally reached on a double-bit uncorrectable ECC error for a read) would most likely cause the same BINIT# event.

**WORKAROUND:** Though the ability to drive BINIT# can be disabled in the Pentium Pro processor, which would prevent the effects of this erratum, overall system behavior would not improve, since the error which would normally cause a BINIT# would instead cause the machine to shut down. No other workaround has been identified.

## 38. RSM Cannot Return to HALT or SHUTDOWN in 32-Bit OS

**PROBLEM:** If a processor which was previously operating in 32-bit protected mode has shut down or executed a HLT instruction, and is then targeted by a System Management Interrupt (SMI), the SMM handler will be executed in a different mode. Upon returning to the HALT or SHUTDOWN state via an RSM, the mode state will not be correctly updated back to 32-bit protected mode, potentially crashing the operating system.

**IMPLICATION:** Systems which support SMM may hang if software which executes HLT instructions is being run, or if the processor is expected to be able to recover from a SHUTDOWN state via software recovery which uses SMM.

**WORKAROUND:** For the HALT case, SMM handler code may clear the Halt Auto Restart bit in the SMM dump space (offset 7F02h), then subtract 1 from the saved EIP (offset 7FF0h). This will cause the HLT instruction to be re-executed upon completion of the RSM, which will allow 32-bit protected mode to be re-entered and the processor to successfully enter the HALT state. No workaround has currently been identified for the SHUTDOWN case; however, it is not guaranteed that an SMI will be properly serviced when in this state.

## 39. Accesses of Modified Data in DCU May Hang System

**PROBLEM:** Under heavy bus traffic in a system with multiple snooping agents (including processors or DMA controllers) which are accessing multiple addresses that map to the same set of the data cache unit (DCU) and L2 cache, the following sequence of transactions may cause the system to hang:

1. A read-for-ownership cycle (a.k.a read and invalidate line) is issued to an address which is in Modified state in both the L2 cache and the DCU.

2. The DCU writes the data at that address back to the L2 cache, so that it is no longer in Modified state in the DCU, only in the L2 cache. It is now in Invalidated state in the DCU.

3. The DCU issues a read to the same address in the L2 cache, and the data is returned in Exclusive state (which is incorrect).

4. The L2 cache proceeds to evict its copy of the line in Modified state. Meanwhile, the DCU modifies its copy of the line in Exclusive state, which then becomes Modified.

5. An external snoop from another processor or DMA controller occurs to the address, receives a HITM# (hit Modified) response, and the DCU's data goes out onto the processor bus.

At this stage, the internal state of the processor may be incorrect, due to an incomplete L2 cache eviction. If the DCU attempts to access the line again, or if another external snoop occurs to the line, the processor will hang, never completing the operation.

**IMPLICATION:** Heavy traffic in a system with devices capable of causing external snoops may cause the system to hang. However, Intel has observed the conditions necessary to cause the traffic described above only on 3 or 4-way MP systems, under heavy traffic or stress-testing conditions where same-set data is being rapidly transferred between main memory, the L2 cache, and the DCU.

**WORKAROUND:** None identified.

## 40.    Branch Traps Do Not Function if BTMs Are Also Enabled

**PROBLEM:**  If branch traps or branch trace messages (BTMs) are enabled alone, both function as expected. However, if both are enabled, only the BTMs will function, and the branch traps will be ignored.

**IMPLICATION:**  These debugging features cannot be used together.

**WORKAROUND:**  If branch trap functionality is desired, BTMs must be disabled.

## 41.    Cache Line May Exist in Two Different Ways in MP System

**PROBLEM:**  In heavy traffic on an MP system, it is possible for a cache line to be read into two different "ways" of the L2 cache, or for an invalid line to become spuriously validated. The conditions necessary for this behavior to occur are as follows:

- A read-for-ownership (a.k.a. read and invalidate line), normal read, and second read-for-ownership must occur, in that order, with no other intervening transactions.
- These three transactions must be pipelined together.
- They must be accessing the same set of the L2 cache.
- A cache line which is in the process of being changed to Shared state must be snooped by an external agent (such as another processor or DMA controller).

**IMPLICATION:**  If these conditions occur, unexpected behavior will result. The most likely failure mechanism for this erratum is a processor hang, but it is possible that data corruption may result. Intel has observed this erratum only in a focus-testing environment with 3 or more processors in the system. Intel has not currently identified any software which causes this behavior.

**WORKAROUND:**  Intel has been unable to reproduce this erratum with an IOQ depth of 1, as configured by the BIOS. This is a recommended configuration for the affected steppings of the processor; production systems with this configuration are not known to be affected by this erratum.

## 42.    HALT, SHUTDOWN, and STPCLK Special Cycles Not Issued

**PROBLEM:**  The HALT, SHUTDOWN, and STPCLK special cycles will not be issued by the Pentium Pro processor.

**IMPLICATION:**  Systems which rely on these special cycles will not function correctly. No systems or software known to Intel rely on the processor issuing these cycles.

**WORKAROUND:**  No known workaround exists for systems which use the B0 stepping of the Pentium Pro processor or the A2 stepping of the 82450GX/KX PCIset. Other systems may contain a workaround for this erratum in BIOS code.

## 43.    L2 Cache May Incorrectly Report BIST Failure

**PROBLEM:**  If BIST (Built-In Self Test) is requested on reset by keeping INIT# asserted past the rising edge of RESET# (i.e. by performing a normal BIST request), it is possible that the L2 cache BIST will not be run. If this occurs, the BIST result reported in the EAX register upon completion of BIST will contain the signature for an L2 cache failure (bits 2, 16, and/or 17 may be non-zero).

**IMPLICATION:**  A BIST failure result from the L2 cache may not be valid. This invalid failure result becomes more probable as the RESET# pulse is lengthened. This erratum has been observed by Intel on devices with both

256-Kbyte and 512-Kbyte L2 caches.  This was fixed in the β steppings of the 256-Kbyte and 512-Kbyte L2 caches.

**WORKAROUND:**  Ignore the result of the L2 BIST by masking bits 2, 17, and 16 of the EAX register before checking the BIST result, or do not run BIST.


## 44.     LOCK Prefix Does Not Generate Trap With Some Instructions

**PROBLEM:**  If the LOCK prefix is used with a CPUID or a near call indirect (FF/2 type of CALL) instruction, a #UD exception should be generated in protected or virtual 8086 mode, and an interrupt 6 should be generated in real address mode. On the affected steppings of the Pentium Pro processor, these exceptions may not occur; the processor may instead decode and execute the instruction as if the LOCK prefix was not present.

**IMPLICATION:**  Illegal use of the LOCK prefix may not always be detected. In cases where it is not detected, it will be ignored (the LOCK# signal will not be asserted for that instruction).

**WORKAROUND:**  Do not use the LOCK prefix with the CALL or CPUID instructions.


## 45.     FRCERR May Be Incorrectly Asserted on Explicit Writeback Cycles

**PROBLEM:**  In a system configured for FRC (Functional Redundancy Checking) mode, an explicit writeback operation (due to a capacity miss in the L2 cache, for example) may cause FRCERR to be asserted, even though no mismatch between the master and checker processors have occurred.

**IMPLICATION:**  For the affected steppings of the Pentium Pro processor, FRC mode is not reliable when used in conjunction with explicit writebacks.

**WORKAROUND:**  If the system is configured to run with all memory in write-through mode, explicit writebacks will not occur (because all writes go through to the processor bus), thus avoiding this erratum.


## 46.     FP Inexact-Result Exception Flag May Not Be Set

**PROBLEM:**  When the result of a floating-point operation is not exactly representable in the destination format (1/3 in binary form, for example), an inexact-result (precision) exception occurs. This exception is commonly masked or ignored by applications, as it happens frequently, and produces a rounded result acceptable to most applications. When this occurs, the PE bit (bit 5 of the FPU status word) is normally set by the processor. Under certain rare conditions, this bit may not be set when this rounding occurs. However, other actions taken by the processor (invoking the software exception handler if the exception is unmasked) are not affected. This erratum can only occur if the floating-point operation which causes the precision exception is immediately followed by one of the following instructions:

- FST m32real/m64real
- FSTP m32real/m64real

Note that even if this combination of instructions is encountered, there is still a very small probability of encountering this erratum, as it is also dependent on the internal pipelining and execution state of both instructions in the processor.

**IMPLICATION:**  The PE bit of the FPU status word may not always be set upon receiving an inexact-result exception. Thus, if these exceptions are unmasked, a floating-point error exception handler may not recognize that a precision exception occurred. Note that this is a "sticky" bit, i.e. once set by an inexact-result condition, it remains set until cleared by software.

**WORKAROUND:** This condition can be avoided by inserting a NOP between the two floating-point instructions.

## 47. Snoop Hit During RSM Segment Load May Hang System

**PROBLEM:** During the execution of an RSM instruction, the processor must reload the state which was saved into SMM (System Management Mode) dump space upon recognition of an SMI# event. If, while loading from an address in SMM memory which contains the saved contents of the segment registers (CS, DS, ES, FS, and GS), a snoop occurs from an external agent which matches the address (in an overlayed system memory region), then the resulting postponement and retry of the segment register load may result in the processor hanging due to an infinite loop in the microcode.

**IMPLICATION:** The processor will hang when exiting from SMM, if an address match occurs between an external snoop to memory and the address of a segment register being loaded from SMM dump space. Note that there is a very small window of opportunity for this erratum to occur.

**WORKAROUND:** Do not overlay system memory with the SMM dump space. If overlaying memory in this way is desired, ensure that the system does not snoop system memory overlaid with SMRAM (by making this region uncacheable, for example).

## 48. Mbyte-Aligned SMM Base Address Causes SHUTDOWN

**PROBLEM:** When servicing an SMI# event, the CS (code segment) selector is formed by shifting the bit value of the SMM (System Management Mode) base address field right by 4 bits and using the low order 16 bits for the selector. If the SMM base address is aligned on a megabyte boundary (i.e. with an SMM base address of 1, 2, 3, etc... Mbytes), these bits will all be 0, resulting in a segment selector value of 0000h, also known as a null selector. This results in the null bit for this segment being set in the segment register. If the CS selector containing the null value is subsequently used as a data selector while in SMM, a GP fault will result; since no interrupt table exists by default in SMM, the processor will then enter SHUTDOWN. Note that if the SMM base address is below 1 Mbyte, this erratum will not occur.

**IMPLICATION:** SMM base addresses which fall on a megabyte boundary (i.e. SMM_BASE mod 1048576 = 0) will result in system failure upon use of CS as a data selector in SMM. SMM base addresses below 1 Mbyte (most systems which support SMM are configured this way) will not be affected by this erratum.

**WORKAROUND:** If the SMM memory space must be placed above 1 Mbyte, do not use a Mbyte-aligned SMM base address when setting up SMM in the BIOS.

## 49. Machine Check Exception with Coexisting Large and Small Pages

**PROBLEM:** The Pentium Pro processor has separate translation lookaside buffers (TLBs) for caching 4Kbyte and 4-Mbyte page translations. When a single linear address is used to access data, and the processor contains an entry in both the 4-Kbyte and the 4-Mbyte TLBs for that address, an unrecoverable Machine Check Exception (MCE) is generated if MCEs are enabled in the system.

**IMPLICATION:** An unrecoverable MCE may be generated in an operating system environment which configures large pages after MCEs have been enabled (including cases where MCEs are enabled, an INIT# sequence is executed, and then large pages are configured).

**WORKAROUND:** This erratum may be avoided by ensuring that after the large page directory entries (PDEs) are created (e.g., by an operating system), all of the TLB entries are flushed before executing code which enables MCEs. Also, code which dynamically creates large pages should ensure that the large page PDEs are not mapped by small (4-Kbyte) PDEs. A BIOS which may cause an INIT# sequence must clear the MC1_CTL-MC4_CTL registers as part of the initialization sequence.

## 50.    WBINVD to Some Modified Addresses Hangs System

**PROBLEM:** If a system with cacheable memory in the address range 70000000h-7003FFFFh has a Modified address in that range, and a WBINVD is performed to write back and invalidate the caches, the processor may hang.

**IMPLICATION:** This region corresponds to a 256-Kbyte range beginning at 1792 Mbytes. Systems with cacheable memory in this region (i.e. with more than 1.792 Gbytes of DRAM) may hang while running software which performs WBINVD instructions. WBINVD is a privileged instruction only executable by operating systems and driver software; some operating systems are known to execute this instruction, though infrequently (usually during bootup).

**WORKAROUND:** Using the variable MTRRs, a write-through (WT) memory region can be set up for the address range 70000000h-7003FFFFh in systems which support over 1.792 Gbytes of DRAM (this should be expanded to a 4-Mbyte region instead of 256 Kbytes, however, due to Erratum 29). This will prevent addresses in this range from being in the Modified state, thus preventing this erratum. However, this also may have a performance impact on the operating system; BIOS should modify the Int 15 E820h function to mark this range as "reserved" in the memory descriptor list provided to the OS through this function. Operating systems which do not use this function call may experience some performance degradation when accessing memory in this region (since it is not WB type). Also note that the WT memory type can be used to override a portion of a WB memory region, similar to UC memory type as currently documented; this will be changed in the specification (see Specification Changes).

## 51.    UC Store May Be Reordered Around WC Transaction

**PROBLEM:** Store operations to uncacheable (UC) memory are intended to be system serializing events on the processor bus, with respect to transactions to write-combining (WC) memory. This means that all WC transactions before a subsequent UC store should complete before the store is allowed to execute, and any subsequent WC transactions should begin after the store completes. However, if the last cache line (or chunk) of data in a WC transaction is retried on the processor bus, a subsequent UC store may be reordered so that the store completes before the retried transaction. It is also possible for the transactions to be reordered in the processor core (as opposed to the external bus) and to be issued on the bus in reverse order. Note that UC loads operate as designed, and will never be reordered in this way.

**IMPLICATION:** If a device which uses WC memory is dependent on the completion of WC transactions before a UC store is executed, perhaps due to some designed-in effect of the UC store (such as activating a graphics accelerator sequence), this erratum may occur and the data from the last portion of the WC transaction would not be present, possibly causing the failure of the designed-in effect. Note that for a linear frame buffer in a video implementation, the effect of this erratum would be that some number of pixels would be updated on the screen on the order of a microsecond later than expected.

**WORKAROUND:** If system serialization by a UC store for WC memory is required, one of two workarounds may be used:

1.  Insert a UC load before the UC store instruction. The WC transaction will not be reordered behind the UC transactions in this case.

2.  Insert an *XCHG reg, stack* instruction before the UC store instruction. This will also guarantee the desired ordering.

## 52.    IFU Error With MCEs, Self- or Cross-Modifying Code

**PROBLEM:** The Pentium Pro processor signals an unrecoverable Machine Check Exception (MCE) if the instruction fetch unit (IFU) detects a particular condition under which the code present in the IFU may have been modified. This condition requires the processor to detect the internal signaling of a branch misprediction coincident

with the internal restarting of a memory load, after speculative execution of the mispredicted code sequence with multiple cache hits to the same set as the code previously being executed. If self- or cross-modifying code is being executed, the existence of this condition may cause corruption of the code.  With MCEs enabled, this corruption cannot occur; however, if self- or cross-modifying code is not being executed, the unrecoverable MCE and operating system processor shutdown are essentially spurious. If MCEs are disabled, and the system is executing self- or cross-modifying code which modifies instructions in the IFU, this erratum may allow the execution of incorrect code, resulting in unpredictable system failure.

**IMPLICATION:**  If the conditions above are met and MCEs are enabled in the operating system, a spurious MCE may occur, causing shutdown; if the same conditions are met with the addition of self- or cross-modifying code being executed under an operating system with MCEs disabled, incorrect code may be executed, resulting in unpredictable system failure. Intel has not identified any commercially available application or operating system software which is affected by this erratum.

**WORKAROUND:**  Enabling MCEs will prevent the execution of incorrect code, but will allow the possibility of a spurious MCE.  Intel has not identified a workaround to prevent the spurious MCEs, when they are enabled.

## 53.  *Extended TRDY# May Cause Hang With DBSY# Asserted*

**PROBLEM:**  The Pentium Pro processor asserts DBSY# during a transfer of data due to an implicit writeback for a write request which receives a HITM# (hit Modified) snoop response. If a responding agent asserts TRDY# for more than one clock beyond DBSY# deassertion (typically, TRDY# is deasserted the clock after DBSY# assertion is observed) to allow previously requested write data to transfer, the Pentium Pro processor may reassert DBSY# for the implicit writeback prematurely. Intel has only observed this erratum in a focused testing environment, when memory aliasing is used (i.e. in an MP system with processors mapping memory to different types, or when a memory type is changed dynamically by programming the Memory Type Range Registers (MTRRs) or changing a page-table entry's PCD bit on the fly).

**IMPLICATION:**  This protocol violation may result in a system hang in systems with agents which assert TRDY# past the typical window.

**WORKAROUND:**  Maintain a consistent MTRR memory model between processors in an MP system, and do not reprogram the MTRRs or change the PCD bits for main memory on the fly.

## 54.  *Call Gates Can Cause Incorrect Handling of Stack Size Attribute*

**PROBLEM:**  Transitioning processor privilege levels via a call gate descriptor may result in the use of an ESP value with the top 16-bits cleared to 0's for the parameter copy portion of the call gate, if the CALL opcode is restarted. If no parameters are copied to the new stack, this erratum will not occur.

The opcode may be restarted due to a processor-initiated update of a page's Access bit, or due to the occurrence of a particular processor-initiated memory ordering event during the opcode's execution. A processor-initiated update of the Access bit occurs when a call gate requires a Task State Segment (TSS) whose page is marked as not having been accessed, or when the page containing a more privileged stack segment is accessed by the call gate. The processor-initiated memory ordering event occurs whenever an address compare (on address signals A#[19:5]) of a previous store from a different processor or DMA agent on the external bus matches the address of a page containing the TSS (or the kernel stack segment) being loaded by the call gate.

Using the incorrect value derived from the ESP during a call gate results in an unexpected page fault (Trap 0Eh) during the parameter copy, unless the miscalculated linear address for the caller's stack resolves to a page that is also present, and is accessible by the caller.  If the page is present and accessible, it is possible that code could be executed using a stack containing incorrect data. At this point, system behavior is dependent upon what the called routine does given the incorrect stack data.

**IMPLICATION:** An application (or the operating system) may be terminated with a Page Fault Exception (Trap 0Eh) during a call gate, due to a miscalculated linear address for the caller's stack. In the event that the miscalculated linear address for the caller's stack resolves to a page that is also present and is accessible by the caller, system behavior becomes dependent upon the called routine's behavior when given the bad stack pointer.

**WORKAROUND:** For software using call gates, there are three possible complete workarounds for this erratum, any of which may be used alone or in combination:

1. Do not pass parameters using the parameter passing feature of a call gate,

2. Use a 32-bit stack in the routine or kernel being called, and either never clear the Access bit of the page containing the TSS or ensure that the Access bit is set when a call gate is performed, or

3. Align the TSS (or the page containing the TSS) on a 7 mod 32 linear address (i.e. 7, 39, 71, etc.), and either never clear the Access bits of the page containing the TSS or ensure that the Access bit is set when a call gate is performed.

In systems using sB1 silicon, BIOS code can contain a workaround for this erratum which precludes the need for any of the above three workarounds.

## 55. *LBER May Be Corrupted After Some Events*

**PROBLEM:** The last branch record (LBR) and the last branch before exception record (LBER) can be used to determine the source and destination information for previous branches or exceptions. The LBR contains the source and destination addresses for the last branch or exception, and the LBER contains similar information for the last branch taken before the last exception. This information is typically used to determine the location of a branch which leads to execution of code which causes an exception. However, after a catastrophic bus condition which results in an assertion of BINIT# and the reinitialization of the buses, the value in the LBER may be corrupted. Also, after either a CALL which results in a fault or a software interrupt, the LBER and LBR will be updated to the same value, when the LBER should not have been updated.

**IMPLICATION:** The LBER and LBR registers are used only for debugging purposes. When this erratum occurs, the LBER will not contain reliable address information. The value of LBER should be used with caution when debugging branching code; if the values in the LBR and LBER are the same, then the LBER value is incorrect. Also, the value in the LBER should not be relied upon after a BINIT# event.

**WORKAROUND:** None identified.

## 56. *SMI During REP String Instruction Clears CR4*

**PROBLEM:** If an assertion of SMI# occurs during a REP string instruction, the system will enter System Management Mode (SMM). When this happens, CR4 is cleared and all information in this register is lost. This register contains information on system settings, such as the Page Global Enable (PGE) and Performance Counter Enable (PCE) bits.

**IMPLICATION:** The settings that are stored in the CR4 register are lost when the system enters SMM during a REP string instruction. This may cause an operating system or application to operate incorrectly or halt unexpectedly.

**WORKAROUND:** It is possible for BIOS code to contain a workaround for this erratum.

## 57. BTMs May Be Corrupted During Simultaneous L1 Cache Line Replacement

**PROBLEM:** When Branch Trace Messages (BTMs) are enabled and such a message is generated, the BTM may be corrupted when issued to the bus by the L1 cache, if a new line of data is brought into the L1 data cache simultaneously. Though the new line being stored in the L1 cache is stored correctly, and no corruption occurs in the data, the information in the BTM may be incorrect due to the internal collision of the data line and the BTM.

**IMPLICATION:** Although BTM's may not be entirely reliable due to the erratum, the conditions for this erratum to occur have only been exhibited during focused simulation testing. Intel has not observed this erratum in a system level validation environment.

**WORKAROUND:** None identified at this time.

## 58. Spurious Machine Check Exception Via IFU Data Parity Error

**PROBLEM:** The Pentium Pro processor can signal an unrecoverable Machine Check Exception (MCE) in the event that the Instruction Fetch Unit (IFU) detects a mismatch when verifying instruction parity. The execution of code which modifies the current instruction sequence that may already be fetched into the processor can cause an instruction at a given address to appear differently depending on when it was fetched in time relative to its being modified. Thus, a speculatively prefetched instruction may have been modified such that it now differs from the copy of the same instruction resident in the instruction cache. This discrepancy (of one copy located in the speculative prefetch portion, and a different copy in the instruction cache) is sensed by the IFU. When the IFU detects that the instruction stream has been modified, it flushes the pipeline and attempts to restart the instruction stream. In the interim, the IFU recognizes the disparate instructions described above, and signals a data parity error. The data parity error is signaled as an MCE before the instruction stream has had a chance to restart. This MCE will cause an operating system that has enabled MCE to shutdown. No incorrect code is executed by the processor in this situation (even if MCE is disabled). Note that this erratum occurs under a specific set of address dependencies and timing events.

**IMPLICATION:** Executing such a sequence of modifying code without proper synchronization may not always result in predictable program behavior. The processor's signaling of an MCE due to a data parity error in the IFU may then result in an unexpected system halt if the above conditions are met and MCEs are enabled.

**WORKAROUND:** For the sB1 stepping of the processor, disabling Instruction Streaming Buffers by setting bit 30 of the Model Specific Register (MSR) at address 33h to"1", can workaround this erratum. The effect on system performance of setting this bit has been found to be smaller than run-to-run measurement variations of conventional benchmarks such as SpecInt95* or SYSmark32*, but the effect may vary depending on workload. For prior steppings, it is possible for BIOS code to contain a workaround for this erratum.

## 59. Loss of Inclusion In IFU Can Cause Machine Check Exception

**PROBLEM:** The Pentium Pro processor can signal an unrecoverable Machine Check Exception (MCE) as a consistency checking mechanism in the event that the Instruction Fetch Unit (IFU) detects differences in the consistency of code in instruction streaming buffers against code resident in the instruction cache i.e., a loss of inclusion. When application code makes an operating system call, the processor transitions execution privilege levels. If the code for the OS call is not already resident in the level 1 cache, then the processor may prefetch code while identifying a cache line(s) for eventual eviction to make space for the new code. Upon return from the OS call, the processor continues execution of application code at the user level. The processor, due to deep speculation and branch prediction, may attempt to execute instructions from the previously prefetched kernel code starting by attempting to replace the victim line with kernel code in a buffer internal to the IFU. The IFU detects that the current application is insufficiently privileged to execute the kernel code and so, suppresses the eviction of the previously selected victim line. Despite having detected this condition, the IFU does replace this victim line with the

kernel line. If the processor now attempts to restart execution of the current application code by refetching the original victim line it no longer finds it in the instruction cache. The IFU detects this loss of inclusion, and signals this by generating a MCE. If MCEs are enabled, this event can cause an operating system to shutdown. Note that this erratum occurs under a specific set of address dependencies and timing events.

**IMPLICATION:** The occurrence of all the conditions above can lead the IFU to signal a loss of inclusion by generating an MCE. If MCEs are enabled in the system, then the operating system may shutdown upon noticing the MCE resulting in system failure. If MCEs are disabled, then unpredictable application behavior is theoretically possible, although current validation has shown execution to continue normally.

**WORKAROUND:** For the sB1 stepping of the processor, disabling Instruction Streaming Buffers by setting bit 30 of the Model Specific Register (MSR) at address 33h to"1", can workaround this erratum. The effect on system performance of setting this bit has been found to be smaller than run-to-run measurement variations of conventional benchmarks such as SpecInt95* or SYSmark32*, but the effect may vary depending on workload. For prior steppings, it is possible for BIOS code to contain a workaround for this erratum.

## 1AP.   APIC Access to Cacheable Memory Causes SHUTDOWN

**PROBLEM:** APIC operations which access memory with any type other than uncacheable (UC) is illegal, and if machine check exceptions (MCEs) are disabled, the CPU will enter shutdown after such an access. If MCEs are enabled, an MCE will occur. However, in this circumstance, a second MCE will be signaled. This will also cause the Pentium Pro processor to enter shutdown.

**IMPLICATION:** Recovery from a PIC access to cacheable memory will not be successful. Correctly written software will not encounter this erratum.

**WORKAROUND:** Ensure that the memory space to which PIC accesses can be made is marked as type UC (uncacheable) in the memory type range registers (MTRRs) to avoid this erratum.

## 2AP.   Possible Hang Due to Catastrophic Errors During BSP Determination

**PROBLEM:** A catastrophic error during the bootstrap processor (BSP) determination process should cause the assertion of IERR#. If the catastrophic error is due to the APIC data bus being stuck at electrical zero, then the system hangs without asserting IERR#.

**IMPLICATION:** Systems may hang during boot due to a catastrophic error. Intel has not observed this erratum under normal system operation; it was found in a test environment using a grounded APIC data bus.

**WORKAROUND:** None identified.

## 3AP.   INIT_IPI After STARTUP_IPI-STARTUP_IPI Sequence May Cause AP to Execute at 0h

**PROBLEM:** The MP Specification states that to wake up an application processor (AP), the interprocessor interrupt sequence INIT_IPI, STARTUP_IPI, STARTUP_IPI should be sent to that processor. On the Pentium Pro processor, an INIT_IPI, STARTUP_IPI sequence will also work. However, if the INIT_IPI, STARTUP_IPI, STARTUP_IPI sequence is sent to an AP, an internal race condition may occur in the APIC logic which leaves the processor in an incorrect state. Operation will be correct in this state, but if another INIT_IPI is sent to the processor, the processor will not stop execution as expected, and will instead begin execution at linear address 0h. In order for the race condition to cause this incorrect state, the system's core to bus clock ratio must be 5:2 or greater.

**IMPLICATION:** If a system is using a core to bus clock ratio of 5:2 or greater, and the sequence INIT_IPI, STARTUP_IPI, STARTUP_IPI is generated on the APIC bus to wake up an AP, and then at some later time another INIT_IPI is sent to the processor, that processor may attempt to execute at linear address 0h, and will execute random opcodes. Some operating systems do generate this sequence when attempting to shut the system down, and in a multiprocessor system, may hang after taking the processors offline. The effect seen will be that the OS may not restart the system if 'shutdown and restart' or the equivalent is selected upon exiting the operating system. If an operating system gives the user the capability to take an AP offline using an INIT_IPI (Intel has not identified any operating systems which currently have this capability), this option should not be used.

**WORKAROUND:** BIOS code should execute a single STARTUP_IPI to wake up an application processor. Operating systems, however, will issue an INIT_IPI, STARTUP_IPI, STARTUP_IPI sequence, as recommended in the MP specification. It is possible that BIOS code may contain a workaround for this erratum in systems with C0 or subsequent steppings of Pentium Pro processor silicon. No workaround is available for the B0 stepping of the Pentium Pro processor.

## 4AP.    Small Window for System Hang After INIT# or INIT_IPI

**PROBLEM:**  There is a very small timing window during which assertion of the INIT# signal or reception of an INIT_IPI may cause a processor to hang, due to an internal race condition.  This window is dependent on the relationship of the APIC clock to the processor BCLK.

**IMPLICATION:**  The system may hang upon receiving an INIT_IPI or assertion of the INIT# signal.  This has only been observed by Intel in a diagnostic/test environment where the INIT# signal was asserted very frequently. If the system hangs due to this erratum, power cycling or performing a hard reset will restart the system successfully.

**WORKAROUND:**  Though Intel has never observed this failure in a system which uses a synchronous APIC clock, it is still theoretically possible to encounter this erratum in such a configuration.  Using an APIC clock frequency ratio of greater than 4.5 (i.e. BCLK / APIC clock frequency$\geq$ 4.5) will avoid this erratum.

## 5AP.    Virtual Wire Mode Through Local APIC May Cause Int 15

**PROBLEM:**  If the APIC subsystem is configured in mixed mode with Virtual Wire mode implemented through the local APIC, an interrupt vector of 0Fh (Intel reserved encoding) may be generated by the local APIC (Int 15). This vector may be generated upon receipt of a spurious interrupt (an interrupt which is removed before the system receives the INTA sequence) instead of the programmed 8259 spurious interrupt vector.

**IMPLICATION:**  The spurious interrupt vector programmed in the 8259 is normally handled by an operating system's spurious interrupt handler.  However, a vector of 0Fh is unknown to some operating systems, which would crash if this erratum occurred.

**WORKAROUND:** BIOS should disable the local APIC in uniprocessor systems and MP systems which do not use an I/O APIC, by setting bit 11 in the APICBASE MSR at address 1Bh to '0'. In MP systems with an I/O APIC, BIOS should configure the local APIC in Virtual Wire mode through the I/O APIC.  In the I/O APIC's redirection table, LVT0 should be programmed for EXTINT interrupts; all other LVT entries should be masked.  The MP Specification Table should be modified to reflect this configuration, as well. Symmetric I/O mode may also be used; this mode will not encounter this erratum.

## 6AP.    Divide By Zero Error on LINT0 Glitch

**PROBLEM:** If the APIC subsystem is configured in mixed mode with Virtual Wire mode implemented through the local APIC, an interrupt vector of 0 (Divide by Zero exception) may be generated by the local APIC (Int 0). This vector may be generated when the LINT0 signal to the local APIC is asserted, then deasserted and rapidly

reasserted again. If the reassertion coincides with the receipt of the corresponding Interrupt Acknowledge (INTA) cycle from the processor core by the local APIC, a vector of 00h is returned by the local APIC to the processor core.  The processor will then generate a Divide by Zero exception.

**IMPLICATION:** If this occurs, the resulting Int 0 may cause an operating system to crash with a Divide by Zero error message.

**WORKAROUND:** BIOS should disable the local APIC in uniprocessor systems and MP systems which do not use an I/O APIC, by setting bit 11 in the APICBASE MSR at address 1Bh to '0'. In MP systems with an I/O APIC, BIOS should configure the local APIC in Virtual Wire mode through the I/O APIC.  In the I/O APIC's redirection table, LVT0 should be programmed for EXTINT interrupts; all other LVT entries should be masked.  The MP Specification Table should be modified to reflect this configuration, as well. Symmetric I/O mode may also be used; this mode will not encounter this erratum.

## 7AP.    Write to Mask LVT (Programmed as EXTINT) Will Not Deassert Outstanding Interrupt

**PROBLEM:**  If the APIC subsystem is configured in Virtual Wire Mode implemented through the local APIC, (i.e. the LINT0 signal is connected to the 8259A and LVT1 is programmed as EXTINT), a write that sets the mask bit of the LVT will not deassert an outstanding interrupt if the LINT0 signal is already asserted. The interrupt will be erroneously posted to the processor core despite the attempt to mask the LVT.

**IMPLICATION:** Because of the masking attempt, interrupts may be generated from an unexpected source when the system expects no interrupts to be posted.

**WORKAROUND:**  Software can issue a write to the IMR of the 8259A to mask all interrupt sources, which will cause the deassertion of the LINT0 signal.  This should be followed by a read to the 8259A IRR to ensure that the LINT0 signal has been deasserted.  Software may then issue the write to mask LVT entry 1.

## 8AP.    APs  Do Not Respond to a STARTUP_IPI After an INIT# or INIT_IPI in Low Power Mode

**PROBLEM:**  When an INIT_IPI is received, all application processors (APs) wait for a STARTUP_IPI message. The AP starts executing at a location derived from this message, once it is received.  While the AP is waiting for a STARTUP_IPI, the core clock is stopped in order to save on power if the Low Power feature is enabled via bit 26 of the EBL_POWERON MSR. The core clock is not re-enabled when a STARTUP_IPI is received, preventing the AP from restarting.

**IMPLICATION:**  If Low Power Mode is enabled, the APs will not restart when a STARTUP_IPI message is received after an INIT# or INIT_IPI. Note that uniprocessor systems are not affected by this erratum.

**WORKAROUND:**  Do not enable Low Power Mode in MP systems.

## 9AP.    Serial Bus Interrupts May Be Lost in APIC Mixed Mode

**PROBLEM:**  If LINT0 or LINT1, set as EXTINT, are asserted, the processor internally issues an Interrupt Acknowledge (INTA) cycle. If the asserted interrupt is then deasserted, as a spurious interrupt, before the internal INTA cycle has completed, the INTA cycle is nonetheless sent to the external system bus. However, if a serial interrupt is received just as the processor has decided to send the INTA from spurious interrupt to the external bus, the spurious interrupt bus cycle is completed correctly but the serial bus interrupt cycle is never completed.

**IMPLICATION:** The APIC will block the service of any equal or lower priority serial interrupts until an internal INTA cycle is received for the initial serial bus interrupt possibly leading to a system hang. In a system where only the serial bus is used or only virtual wire through the local or I/O APIC is used this errata will not be seen.

**WORKAROUND:** Avoid using the APIC in mixed mode, which has the potential of losing serial interrupts due to this erratum. Use the APIC in either virtual wire through the local APIC, virtual wire through the I/O APIC or Symmetric APIC mode only.

# SPECIFICATION CLARIFICATIONS

The Specification Clarifications listed in this section apply to the *Pentium® Pro Family Developer's Manual*, Volumes 1, 2, and 3. All Specification Clarifications will be incorporated into a future version of the appropriate Pentium Pro processor documentation.

## 1. Signal Edge Rate for 3.3V Tolerant, APIC, and JTAG Signals

The following note will be added to the setup and hold times for the 3.3V tolerant, APIC, and boundary scan signals (T12, T13, T28, T29, T43, and T44):

"Specified over the rise time (Tr) and fall time (Tf) ranges of 0.3 ns to 2 ns for these signals, between 0.8V and 2.0V (as defined by Figure 11-7)."

The title "Generic Clock Waveform" of Figure 11-7 will be changed to "Generic Waveform" to allow this reference.

## 2. OUTS Instruction and Subsequent Instructions

The description for the OUTS instruction in the *Pentium® Pro Family Developer's Manual, Volume 2: Programmer's Reference Manual*, 6th paragraph, contains incorrect information regarding sampling of the EWBE# pin, which is not a Pentium Pro processor signal. This paragraph should document that upon execution of an OUTS, OUTSB, OUTSW, or OUTSD instruction, the Pentium Pro processor will not execute the next instruction until the data phase of the transaction is complete.

## 3. Interrupt Recognition Determines Priority

The interrupt priority documented in Table 52 of the *Pentium® Pro Family Developer's Manual, Volume 3: Operating System Writer's Manual*, reflects the order in which interrupts will be serviced upon simultaneous recognition by the processor (for example, when multiple interrupts are pending at an instruction boundary). This table does not necessarily reflect the order in which interrupts will be recognized by the processor if received simultaneously at the processor pins.

## 4. References to 2-Mbyte Pages Should Include 4-Mbyte Pages

Generically, "large pages" refers to either 2-Mbyte or 4-Mbyte pages. In Section 3.8. of the *Pentium® Pro Family Developer's Manual, Volume 3: Operating System Writer's Manual*, 2-Mbyte pages are often referenced alone, when the behavior of 4-Mbyte pages is identical; these references should include all large pages.

## 5. Modification of Reserved Areas in the SMRAM Saved State Map

If data is incorrectly written to reserved areas of the saved state map, the processor will enter the shutdown state. This can also occur if invalid state information is saved in the SMRAM (such as if illegal combinations of bits are written to CR0 or CR4 before an SMI is serviced). CR4 is not distinctly part of the saved state map, as implied in Section 9.3.1.1. of the *Pentium® Pro Family Developer's Manual, Volume 3: Operating System Writer's Manual*

## 6.     FRCERR Asserted for FRC Checker BIST Failure

In Section 3.4.8., *Error Signals*, in the *Pentium® Pro Family Developer's Manual, Volume 1:  Specifications*, the last paragraph states:  "When BIST completes, the Pentium Pro processor deasserts FRCERR if BIST succeeds and continues to assert FRCERR if BIST fails." This is only true for the FRC checker in an FRC system; in a non-FRC system, FRCERR is always deasserted, regardless of the BIST result.  There is no external indication of BIST failure in a non-FRC system or for an FRC master; to determine the results of BIST, the result must be read from EAX after BIST (a non-zero result indicates failure).

## 7.     Deassertion of BREQn# During RESET#

The description of BREQ*n*# deassertion protocol in Section 4.1.5.1, *Reset Conditions*, in the *Pentium® Pro Family Developer's Manual, Volume 1: Specifications* does not match that described elsewhere in the document (including Table 11-14, *Reset Conditions A.C. Specifications*.  This section will be modified as follows:

### 4.1.5.1     Reset Conditions

Table 11-14, *Reset Conditions A.C. Specifications*, describes the timing requirements for configuration signals on observation of active RESET#. When a reinitialization condition is generated by the activation of BINIT#, BREQ*n*# must remain deasserted until 4 clocks after BINIT# is driven inactive. The first BREQ*n*# sample point is 4 clocks after BINIT# is sampled inactive.

## 8.     TLB Flush Necessary After PDPE Change

As described in Section 3.7., *Translation Lookaside Buffers (TLBs),* in the *Pentium® Pro Family Developer's Manual, Volume 3: Operating System Writer's Manual,* the operating system is required to invalidate the corresponding entry in the TLB after any change to a page-directory or page-table entry. However, if the physical address extension (PAE) feature is enabled to use 36-bit addressing, a new table is added to the paging hierarchy, called the page directory pointer table (as per Section 3.8., *Physical Address Extension*). If an entry is changed in this table (to point to another page directory), the TLBs must then be flushed by writing to CR3.

## 9.     LOCK# Deasserted Between Locked Sequences

The following clarification will be added to the description of the bus lock action in Sections 3.4.2., *Arbitration Phase Signals*, and A.1.37., *LOCK# (I/O)*, of the *Pentium® Pro Family Developer's Manual, Volume 1: Specifications*:

The LOCK# signal is always deasserted between two sequences of locked transactions on the Pentium Pro processor bus.

## 10.     EXF4# Assertion in SMM

Section 3.4.3, *Request Signals*, in the *Pentium® Pro Family Developer's Manual, Volume 1:  Specifications*, lists the EXF[4:0]# signals and their functions in Table 3-11.  The text for this table will contain the following clarification of the behavior of EXF4#:

EXF4# (SMM Memory) is asserted by the Pentium Pro processor during the second clock of the Request Phase in each transaction when the processor is in System Management Mode and indicates that the processor is accessing a separate "shadow" memory, the SMRAM.

## 11.    Preventing Caching in the Pentium® Pro Processor

Section 11.5.2, *Preventing Caching*, in the *Pentium® Pro Family Developer's Manual, Volume 3:  Operating System Writer's Guide*, documents the procedure to prevent the L1 and L2 caches from performing all caching operations.  However, this procedure differs from that given in Section 11.11.8, *Multiple-Processor Considerations*. The correct procedure that should be used is as follows:

1.  Enter the no-fill cache mode.  (Set the CD flag in control register CR0 to 1 and the NW flag to 0.)

2.  Flush all caches using the WBINVD instruction.

3.  Disable the MTRRs and set the default memory type to uncached, or set all MTRRs for the uncached memory type (see the discussion of the TYPE field and the E flag in Section 11.11.2.1., "MTRRdefType Register").

The caches must be flushed when the CD flag is cleared to insure system memory coherency.  If the caches are not flushed in step 2, cache hits on reads will still occur and data will be read from valid cache lines.

## 12.    MC*i*_CTL Registers Not Cleared by INIT#

During an assertion of the INIT# signal, all Pentium Pro processor bus agents are reset without affecting their internal caches or floating-point registers, as stated in Section 3.4.1. of the *Pentium® Pro Family Developer's Manual, Volume 1: Specifications*.  This section should also state that the Machine Check Architecture registers (MC*i*_CTL) are also unaffected by the INIT# sequence.

## 13.    Stack Use During Interrupts, Exceptions, and CALLs

Sections 4.3.6. and 4.4.1. of the *Pentium® Pro Family Developer's Manual, Volume 2: Programmer's Reference Manual* contain inconsistencies which will be clarified to reflect the actual behavior of the architecture, as follows:

### 4.3.6.    CALL and RET Operation Between Privilege Levels

When making a call to a more privileged protection level, the processor does the following (see Figure 4-3):

1.  Performs an access rights check (privilege check).

2.  Temporarily saves (internally) the current contents of the SS, ESP, CS, and EIP registers.

3.  Loads the segment selector and stack pointer for the new stack (that is, the stack for the privilege level being called) from the TSS into the SS and ESP registers and switches to the new stack.

4.  Pushes the temporarily saved SS and ESP values for the calling procedure's stack onto the new stack.

5.  Copies the parameters from the calling procedure's stack to the new stack. (A value in the call gate descriptor determines how many parameters to copy to the new stack.)

6.  Pushes the temporarily saved CS and EIP values for the calling procedure to the new stack.

7.  Loads the segment selector for the new code segment and the new instruction pointer from the call gate into the CS and EIP registers, respectively.

8.  Begins execution of the called procedure at the new privilege level.

When executing a return from the privileged procedure, the processor performs these actions:

1.  Performs a privilege check.

2.  Restores the CS and EIP registers to their values prior to the call.

3. (If the RET instruction has an optional n argument.) Increments the stack pointer by the number of bytes specified with the n operand to release parameters from the stack. If the call gate descriptor specifies that one or more parameters be copied from one stack to the other, a RET n instruction must be used to release the parameters from both stacks. Here, the n operand specifies the number of bytes occupied on each stack by the parameters. On a return, the processor increments ESP by n for each stack to step over (effectively remove) these parameters from the stacks.

4. Restores the SS and ESP registers to their values prior to the call, which causes a switch back to the stack of the calling procedure.

5. (If the RET instruction has an optional n argument.) Increments the stack pointer by the number of bytes specified with the n operand to release parameters from the stack (see explanation in step 3).

6. Resumes execution of the calling procedure.

See Chapter 4, *Protection*, in the *Pentium® Pro Family Developer's Manual, Volume 3* for detailed information on calls to privileged levels and the call gate descriptor.
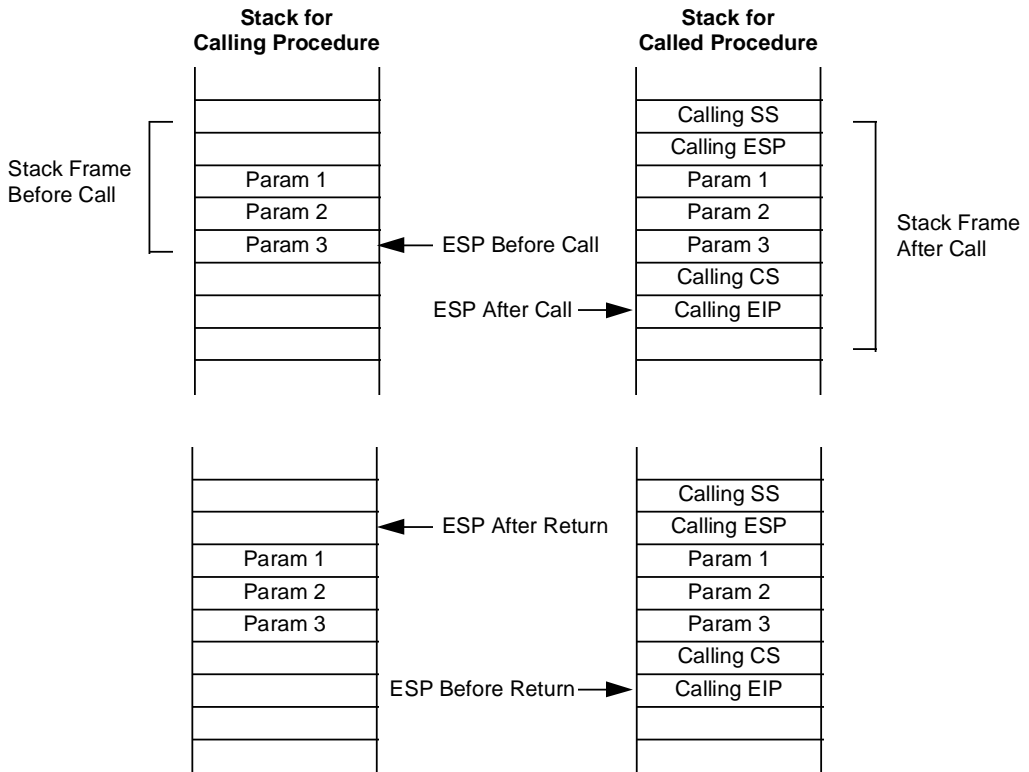


Note: On a return, parameters are
released on both stacks if the
correct value is given for the *n*
operand in the RET *n* instruction.

**Figure 4-3. Stack Switch on a Call to a Different Privilege Level**

## 4.4.1. Call and Return Operation for Interrupt or Exception Handling Procedures

A call to an interrupt or exception handler procedure is similar to a procedure call to another protection level (as described in Section 4.3.6., *CALL and RET Operation Between Privilege Levels*). Here, the interrupt vector references one of two kinds of gates: an interrupt gate or a trap gate. Interrupt and trap gates are similar to call gates in that they provide the following information:

• Access rights information.

• The segment selector for the code segment that contains the handler procedure.

• An offset into the code segment to the first instruction of the handler procedure.

The difference between an interrupt gate and a trap gate are as follows. If an interrupt or exception handler is called through an interrupt gate, the processor clears the interrupt enable (IF) flag in the EFLAGS register to prevent subsequent interrupts from interfering with the execution of the handler. When a handler is called through a trap gate, the state of the IF flag is not changed.

If the code segment for the handler procedure has the same privilege level as the currently executing program or task, the handler procedure uses the current stack; if the handler executes at a more privileged level, the processor switches to the stack for the handler's privilege level.

If no stack switch occurs, the processor does the following when calling an interrupt or exception handler (see Figure 44):

1. Pushes the current contents of the EFLAGS, CS, and EIP registers (in that order) on the stack.

2. Pushes an error code (if appropriate) on the stack.

3. Loads the segment selector for the new code segment and the new instruction pointer (from the interrupt gate or trap gate) into the CS and EIP registers, respectively.

4. If the call is through an interrupt gate, clears the IF flag in the EFLAGS register.

5. Begins execution of the handler procedure at the new privilege level.

If a stack switch does occur, the processor does the following:

1. Temporarily saves (internally) the current contents of the SS, ESP, EFLAGS, CS, and EIP registers.

2. Loads the segment selector and stack pointer for the new stack (that is, the stack for the privilege level being called) from the TSS into the SS and ESP registers and switches to the new stack.

3. Pushes the temporarily saved SS, ESP, EFLAGS, CS, and EIP values for the interrupted procedure's stack onto the new stack.

4. Pushes an error code on the new stack (if appropriate).

5. Loads the segment selector for the new code segment and the new instruction pointer (from the interrupt gate or trap gate) into the CS and EIP registers, respectively.

6. If the call is through an interrupt gate, clears the IF flag in the EFLAGS register.

7. Begins execution of the handler procedure at the new privilege level.

A return from an interrupt or exception handler is initiated with the IRET instruction. The IRET instruction is similar to the far RET instruction, except that it also restores the contents of the EFLAGS register for the interrupted procedure:

When executing a return from an interrupt or exception handler from the same privilege level as the interrupted procedure, the processor performs these actions:

1. Restores the CS and EIP registers to their values prior to the interrupt or exception.

2. Restores the EFLAGS register.

3. Increments the stack pointer appropriately

4. Resumes execution of the interrupted procedure.

When executing a return from an interrupt or exception handler from a different privilege level than the interrupted procedure, the processor performs these actions:

1. Performs a privilege check.

2. Restores the CS and EIP registers to their values prior to the interrupt or exception.

3. Restores the EFLAGS register.

4. Restores the SS and ESP registers to their values prior to the interrupt or exception, resulting in a stack switch back to the stack of the interrupted procedure.

5. Resumes execution of the interrupted procedure.

**Stack Usage with No
Privilege-Level Change**

Interrupted Procedure's
and Handler's Stack



| | |
|---|---|
| | ◄— ESP Before Transfer to Handler |
| EFLAGS | |
| CS | |
| EIP | |
| Error Code | ◄— ESP After Transfer to Handler |

**Stack Usage with
Privilege-Level Change**

Interrupted Procedure's
Stack

Handler's Stack



| | |
|---|---|
| ◄— ESP Before Transfer to Handler | |
| | SS |
| | ESP |
| | EFLAGS |
| | CS |
| | EIP |
| ESP After —► Transfer to Handler | Error Code |
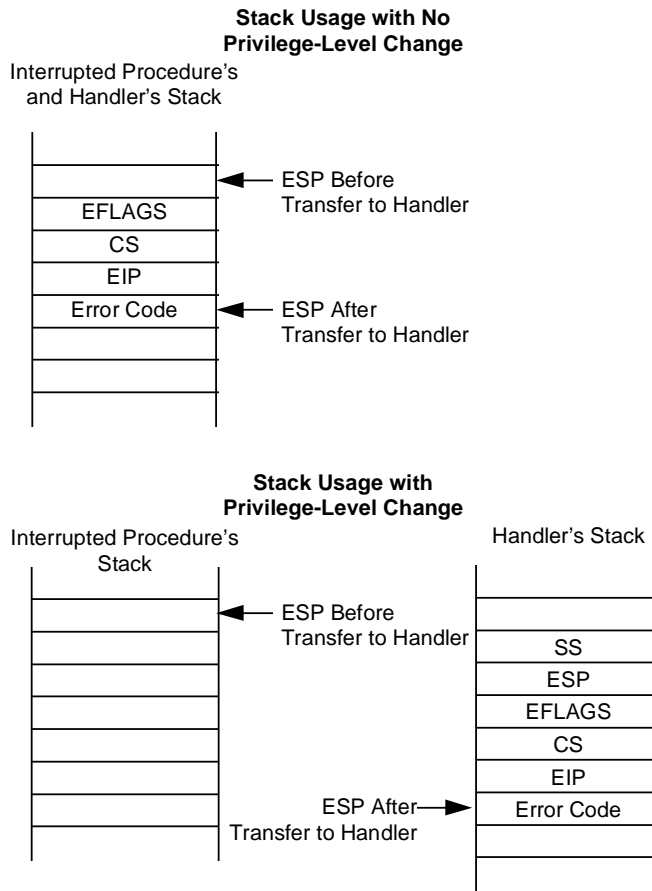
**Figure 4-4. Stack Usage on Transfers to Interrupt and Exception Handling Routines**

## 14.    *Performance-Monitoring Counter Issues*

This Specification Clarification documents the currently characterized differences between the behavior of the Pentium® Pro processor's performance-monitoring counters and that documented in Appendix B, *Performance Monitoring Counters*, of the *Pentium® Pro Family Developer's Manual, Volume 3: Operating System Writer's Guide*.

The following table replaces Table B-1, Appendix B, *Pentium® Pro Family Developer's Manual, Volume 3*. The only change to this new table are enhanced descriptions of the events counted.

| Unit | Event Number | Mnemonic Event Name | Unit Mask | Description | Comments |
|------|--------------|---------------------|-----------|-------------|----------|
| Data Cache Unit (DCU) | 43H | DATA_MEM_REFS | 00H | All loads from any memory type. All stores to any memory type. Each part of a split is counted separately. The internal logic counts not only external memory loads and stores, but also internal retries.<br><br>Note: 80 bit floating point accesses are double counted, since they are decomposed into a 16 bit exponent load and a 64 bit mantissa load. Memory accesses are only counted when they are actually performed. E.g. a load that gets squashed because a previous cache miss is outstanding to the same address, and which finally gets performed, is only counted once.<br>Does not include I/O accesses, or other non-memory accesses. | |
| | 45H | DCU_LINES_IN | 00H | Total lines allocated in the DCU. | |
| | 46H | DCU_M_LINES_IN | 00H | Number of M state lines allocated in the DCU. | |
| | 47H | DCU_M_LINES_OUT | 00H | Number of M state lines evicted from the DCU. This includes evictions via snoop HITM, intervention or replacement. | |

| Unit | Event Number | Mnemonic Event Name | Unit Mask | Description | Comments |
|------|--------------|---------------------|-----------|-------------|----------|
| | 48H | DCU_MISS_OUTSTAND-ING | 00H | Weighted number of cycles while a DCU miss is outstanding. Incremented by the number of outstanding cache misses at any | An access that also misses the L2 is short-changed by 2 cycles. (i.e. if count is N cycles, should be N+2 cycles.) |

| | | | | particular time. Cacheable read requests only are considered. Uncacheable requests are excluded. Read-for-ownerships are counted as well as line fills, invalidates, stores. | Subsequent loads to the same cache line will not result in any additional counts.Count value not precise, but still useful. |
|---|---|---|---|---|---|
| Instr-uction Fetch Unit (IFU) | 80H | IFU_IFETCH | 00H | Number of instruction fetches, both cacheable and non-cacheable. Including UC fetches. | |
| | 81H | IFU_IFETCH_MISS | 00H | Number of instruction fetch misses. All instruction fetches that do not hit the IFU, i.e. that produce memory requests. Includes UC accesses. | |
| | 85H | ITLB_MISS | 00H | Number of ITLB misses. | |
| | 86H | IFU_MEM_STALL | 00H | Number of cycles instruction fetch is stalled, for any reason. Includes IFU cache misses, ITLB misses, ITLB faults and other minor stalls. | |
| | 87H | ILD_STALL | 00H | Number of cycles that the instruction length decoder is stalled. | |

| Unit | Event Number | Mnemonic Event Name | Unit Mask | Description | Comments |
|------|-------------|---------------------|-----------|-------------|----------|
| L2 Cache [1] | 28H | L2_IFETCH | MESI 0FH | Number of L2 instruction fetches. This event indicates that a normal instruction fetch was received by the L2. The count includes only L2 cacheable instruction fetches; it does not include UC instruction fetches. It does not include ITLB miss accesses. | |
| | 29H | L2_LD | MESI 0FH | Number of L2 data loads. This event indicates that a normal, unlocked, load memory access was received by the L2. It includes only L2 cacheable memory accesses; it does not include I/O accesses, other non-memory accesses, or memory accesses such as UC/WT memory accesses. It does include L2 cacheable TLB miss memory accesses. | |
| | 2AH | L2_ST | MESI 0FH | Number of L2 data stores. This event indicates that a normal, unlocked, store memory access was received by the L2. Specifically, it indicates that the DCU sent a read-for-ownership request to the L2. It also includes Invalid to Modified requests sent by the DCU to the L2. It includes only L2 cacheable store memory accesses; it does not include I/O accesses, other non-memory accesses, or memory accesses like UC/WT stores. It includes TLB miss memory accesses. | |

| Unit | Event Number | Mnemonic Event Name | Unit Mask | Description | Comments |
|------|--------------|---------------------|-----------|-------------|----------|
| | 24H | L2_LINES_ IN | 00H | Number of lines allocated in the L2. | |
| | 26H | L2_LINES_ OUT | 00H | Number of lines removed from the L2 for any reason. | |
| | 25H | L2_M_LIN ES_INM | 00H | Number of modified lines allocated in the L2. | |
| | 27H | L2_M_LIN ES_OUTM | 00H | Number of modified lines removed from the L2 for any reason. | |
| | 2EH | L2_RQSTS | MESI 0FH | Total number of L2 requests. | |
| | 21H | L2_ADS | 00H | Number of L2 address strobes. | |
| | 22H | L2_DBUS_ BUSY | 00H | Number of cycles during which the L2 cache data bus was busy. | |
| | 23H | L2_DBUS_ BUSY_RD | 00H | Number of cycles during which the data bus was busy transferring read data from L2 to the processor. | |
| Exter-nal Bus Logic (EBL)[2] | 62H | BUS_DRD Y_CLOCK S | 00H (Self) 20H (Any) | Number of clocks during which DRDY# is asserted. Essentially, utilization of the external system data bus | Unit Mask = 00H counts bus clocks when the processor is driving DRDY#.Unit Mask = 20H counts in processor clocks when any agent is driving DRDY#. |
| | 63H | BUS_LOC K_CLOCK S | 00H (Self) 20H (Any) | Number of clocks during which LOCK# is asserted on the external system bus. | Always counts in processor clocks |

| Unit | Event Number | Mnemonic Event Name | Unit Mask | Description | Comments |
|---|---|---|---|---|---|
| | 60H | BUS_REQ _OUTSTA ND-ING | 00H (Self) | Number of bus requests outstanding. This counter is incremented by the number of cacheable read bus requests outstanding in any given cycle | Counts only DCU full-line cacheable reads, not RFOs, writes, instruction fetches, or anything else. Counts "waiting for bus to complete" (last data chunk received). |
| | 65H | BUS_TRA N_BRD | 00H (Self) 20H (Any) | Number of burst read transactions. | |
| | 66H | BUS_TRA N_RFO | 00H (Self) 20H (Any) | Number of completed read for ownership transactions. | |
| | 67H | BUS_ TRANS_W B | 00H (Self) 20H (Any) | Number of completed write back transactions. | |
| | 68H | BUS_TRA N_IFETCH | 00H (Self) 20H (Any) | Number of completed instruction fetch transactions. | |
| | 69H | BUS_TRA N_INVAL | 00H (Self) 20H (Any) | Number of completed invalidate transactions. | |
| | 6AH | BUS_TRA N_PWR | 00H (Self) 20H (Any) | Number of completed partial write transactions. | |
| | 6BH | BUS_ TRANS_P | 00H (Self) 20H (Any) | Number of completed partial transactions. | |
| | 6CH | BUS_TRA NS_IO | 00H (Self) 20H (Any) | Number of completed I/O transactions. | |
| | 6DH | BUS_TRA N_DEF | 00H (Self) 20H (Any) | Number of completed deferred transactions. | |

**intel®**

| Unit | Event Number | Mnemonic Event Name | Unit Mask | Description | Comments |
|------|--------------|---------------------|-----------|-------------|----------|
| | 6EH | BUS_TRAN_BURST | 00H (Self) 20H (Any) | Number of completed burst transactions. | |
| | 70H | BUS_TRAN_ANY | 00H (Self) 20H (Any) | Number of all completed bus transactions. Address bus utilization can be calculated knowing the minimum address bus occupancy. Includes special cycles etc. | |
| | 6FH | BUS_TRAN_MEM | 00H (Self) 20H (Any) | Number of completed memory transactions. | |
| | 64H | BUS_DATA_RCV | 00H (Self) | Number of bus clock cycles during which this processor is receiving data. | |
| | 61H | BUS_BNR_DRV | 00H (Self) | Number of bus clock cycles during which this processor is driving the BNR# pin. | |

| Unit | Event Number | Mnemonic Event Name | Unit Mask | Description | Comments |
|------|--------------|---------------------|-----------|-------------|----------|
| | 7AH | BUS_HIT_ DRV | 00H (Self) | Number of bus clock cycles during which this processor is driving the HIT# pin. | Includes cycles due to snoop stalls. The event counts correctly, but the BPM*i* pins function as follows based on the setting of the PC bits (bit 19 in the PerfEvtSel0 and PerfEvtSel1 registers). If the core clock to bus clock ratio is 2:1 or 3:1, and a PC bit is set, the BPM*i* pins will be asserted for a single clock when the counters overflow. If the PC bit is clear, the BPM*i* pins will toggle (remaining high for 2 clocks) when the counter is incremented. If the clock ratio is not 2:1 or 3:1, the BPM*i* pins will not function for these performance-monitoring counter events. |

| Unit | Event Number | Mnemonic Event Name | Unit Mask | Description | Comments |
|------|--------------|---------------------|-----------|-------------|----------|
| | 7BH | BUS_HITM _DRV | 00H (Self) | Number of bus clock cycles during which this processor is driving the HITM# pin. | Includes cycles due to snoop stalls. The event counts correctly, but the BPM*i* pins function as follows based on the setting of the PC bits (bit 19 in the PerfEvtSel0 and PerfEvtSel1 registers). If the core clock to bus clock ratio is 2:1 or 3:1, and a PC bit is set, the BPM*i* pins will be asserted for a single clock when the counters overflow. If the PC bit is clear, the BPM*i* pins will toggle (remaining high for 2 clocks) when the counter is incremented. If the clock ratio is not 2:1 or 3:1, the BPM*i* pins will not function for these performance-monitoring counter events. |
| | 7EH | BUS_ SNOOP_ STALL | 00H (Self) | Number of clock cycles during which the bus is snoop stalled. | |

| Unit | Event Number | Mnemonic Event Name | Unit Mask | Description | Comments |
|------|--------------|---------------------|-----------|-------------|----------|
| Float-ing Point Unit | C1H | FLOPS | 00H | Number of computational floating-point operations retired. Excludes floating point computational operations that cause traps or assists. Includes floating point computational operations executed by the assist handler. Includes internal sub-operations of complex floating point instructions like transcendentals. Excludes floating point loads and stores. | Counter 0 only |
| | 10H | FP_COMP_OPS_EXE | 00H | Number of computational floating-point operations executed. The number of FADD, FSUB, FCOM, FMULs, integer MULs and IMULs, FDIVs, FPREMs, FSQRTS, integer DIVs and IDIVs. Note not the number of cycles bus, the number of operations. This event does not distinguish an FADD used in the middle of a transcendental flow from a separate FADD instruction. | Counter 0 only |
| | 11H | FP_ASSIST | 00H | Number of floating-point exception cases handled by microcode. | Counter 1 only. This event includes counts due to speculative execution. |
| | 12H | MUL | 00H | Number of multiplies. Note: includes integer and well FP multiplies and is speculative. | Counter 1 only |
| | 13H | DIV | 00H | Number of divides. Note: includes integer and FP multiplies and is speculative. | Counter 1 only |

| Unit | Event Number | Mnemonic Event Name | Unit Mask | Description | Comments |
|---|---|---|---|---|---|
| | 14H | CYCLES_ DIV_BUSY | 00H | Number of cycles that the divider is busy, and cannot accept new divides. Note: includes integer and FP divides, FPREM, FPSQRT, etc. and is speculative. | Counter 0 only |
| Mem- ory Order- ing | 03H | LD_BLOC KS | 00H | Number of store buffer blocks. Includes counts caused by preceding stores whose addresses are unknown, preceding stores whose addresses are known to conflict, but whose data is unknown and preceding stores that conflicts with the load, but which incompletely overlap the load. | |
| | 04H | SB_DRAIN S | 00H | Number of store buffer drain cycles. Incremented during every cycle the store buffer is draining. Draining is caused by serializing operations like CPUID, synchronizing operations like XCHG, Interrupt acknowledgment as well as other conditions such as cache flushing. | |

| Unit | Event Number | Mnemonic Event Name | Unit Mask | Description | Comments |
|------|--------------|---------------------|-----------|-------------|----------|
| | 05H | MISALIGN_MEM_REF | 00H | Number of misaligned data memory references. Incremented by 1 every cycle during which either the PPro load or store pipeline dispatches a misaligned uop. Counting is performed if its the first half or second half, or if it is blocked, squashed or misses.<br>Note in this context misaligned means crossing a 64 bit boundary. | It should be noted that MISALIGN_MEM_REF is only an approximation, to the true number of misaligned memory references. The value returned is roughly proportional to the number of misaligned memory accesses, i.e. the size of the problem |
| Instr-uction Decod-ing and Retire ment | C0H | INST_ RETIRED | OOH | Number of instructions retired. | |
| | C2H | UOPS_ RETIRED | 00H | Number of UOPs retired. | |
| | D0H | INST_ DECODER | 00H | Number of instructions decoded. | |
| Inter-rupts | C8H | HW_INT_R X | 00H | Number of hardware interrupts received. | |
| | C6H | CYCLES_ INT_ MASKED | 00H | Number of processor cycles for which interrupts are disabled. | |
| | C7H | CYCLES_ INT_ PENDING_ AND_ MASKED | 00H | Number of processor cycles for which interrupts are disabled and interrupts are pending. | |
| Bran-ches | C4H | BR_INST_ RETIRED | 00H | Number of branch instructions retired. | |

intel®

| Unit | Event Number | Mnemonic Event Name | Unit Mask | Description | Comments |
|---|---|---|---|---|---|
| | C5H | BR_MISS_ PRED_ RETIRED | 00H | Number of mispredicted branches retired. | |
| | C9H | BR_TAKE N_RETIRE D | 00H | Number of taken branches retired. | |
| | CAH | BR_MISS_ PRED_TA KEN_RET | 00H | Number of taken mispredictions branches retired. | |
| | E0H | BR_INST_ DECODED | 00H | Number of branch instructions decoded. | |
| | E2H | BTB_ MISSES | 00H | Number of branches that for which the BTB did not produce a prediction | |
| | E4H | BR_BOGU S | 00H | Number of bogus branches. | |
| | E6H | BACLEAR S | 00H | Number of time BACLEAR is asserted. This is the number of times that a static branch prediction was made, where the branch decoder decided to make a branch prediction because the BTB did not. | |

| Unit | Event Number | Mnemonic Event Name | Unit Mask | Description | Comments |
|------|--------------|---------------------|-----------|-------------|----------|
| Stalls | A2H | RESOURCE_STALLS | 00H | Incremented by one during every cycle that there is a resource related stall. Includes register renaming buffer entries, memory buffer entries. Does not include stalls due to bus queue full, too many cache misses, etc. In addition to resource related stalls, this event counts some other events. Includes stalls arising during branch misprediction recovery e.g. if retirement of the mispredicted branch is delayed and stalls arising while store buffer is draining from synchronizing operations. | |
| | D2H | PARTIAL_ RAT_ STALLS | 00H | Number of cycles or events for partial stalls. Note Includes flag partial stalls. | |
| Seg-ment Reg-ister Loads | 06H | SEGMENT_ REG_ LOADS | 00H | Number of segment register loads | |
| Clocks | 79H | CPU_CLK_ UNHALTE D | 00H | Number of cycles during which the processor is not halted. | |

NOTES:
1.  Several L2 cache events, where noted, can be further qualified using the Unit Mask (UMSK) field in the PerfEvtSel0 and PerfEvtSel1 registers. The lower 4 bits of the Unit Mask field are used in conjunction with L2 events to indicate the cache state or cache states involved. The Pentium Pro processor identifies cache states using the "MESI" protocol and consequently each bit in the Unit Mask field represents one of the four states: UMSK[3] = M (8H) state, UMSK[2] = E (4H) state, UMSK[1] = S (2H) state, and UMSK[0] = I (1H) state. UMSK[3:0] = MESI (FH) should be used to collect data for all states; UMSK = 0H, for the applicable events, will result in nothing being counted.
2.  All of the external bus logic (EBL) events, except where noted, can be further qualified using the Unit Mask (UMSK) field in the PerfEvtSel0 and PerfEvtSel1 registers. Bit 5 of the UMSK field is used in conjunction with the EBL events to indicate whether the processor should count transactions that are self generated (UMSK[5] = 0) or transactions that result from any processor on the bus (UMSK[5] = 1).

## 15.    *Clock Jitter Measurement Clarification*

The following text will be added to footnote 3 of Table 11-9, Bus Clock A.C. Specifications, in the *Pentium® Pro Family Developer's Manual, Volume 1*.

"To ensure a 1:1 relationship between the amplitude of the input jitter and the internal and external clocks, the jitter frequency spectrum should not have any power spectrum peaking between 100kHz and 10MHz.  A spectrum analyzer can display the frequency spectrum of the clock driver in your system."

# DOCUMENTATION CHANGES

The Documentation Changes listed in this section apply to the *Pentium® Pro Family Developer's Manual,* Volumes 1, 2, and 3. All Documentation Changes will be incorporated into a future version of the appropriate Pentium Pro processor documentation.

## 1. PE Bit Number in FPU Status Word, Volume 2, Page 7-50

Section 7.8.6, Inexact-Result (Precision) Exception (#P)

Paragraph 2 of this section should read "The inexact-result exception flag (PE) is bit **5** of the FPU status word, and the mask bit (PM) is bit **5** of the FPU control word."

## 2. GTL+ Edge Rate Range Incorrect

Figures 12-4 and 12-14 in the *Pentium® Pro Family Developer's Manual, Volume 1: Specifications* document falling edge rate ranges in the waveforms for characterizing receiver ringback tolerance and setup time. Both figures document ranges of 0.3 V/ns to 3.0 V/ns; these values should be 0.3 V/ns to **1.5 V/ns**. Also, Figure 12-13 documents the rising edge rate range as 0.3 V/ns to 1.5 V/ns; this should be 0.3 V/ns to **0.8 V/ns**.

## 3. Power-on Configuration Register Documented Inconsistently

In the *Pentium® Pro Family Developer's Manual,* the Power-on Configuration Register (or EBL_CR_POWERON) values documented in Section 9.3 of *Volume 1: Specifications* and Appendix C of *Volume 3: Operating System Writer's Guide* are inconsistent. The correct values are presented below.

| Register Address | | Register Name | Bit Description |
|---|---|---|---|
| **Hex** | **Dec** | | |
| 2AH | 42 | EBL_CR_POWERON | |
| | | 0 | Reserved* |
| | | 1 | Data Error Checking Enable<br>1 = Enabled<br>0 = Disabled<br>Read/Write |
| | | 2 | Response Error Checking Enable<br>FRCERR Observation Enable<br>1 = Enabled<br>0 = Disabled<br>Read/Write |
| | | 3 | AERR# Driver Enable<br>1 = Enabled<br>0 = Disabled<br>Read/Write |

| Register Address | | | |
|---|---|---|---|
| **Hex** | **Dec** | **Register Name** | **Bit Description** |
| | | 4 | BERR# Driver Enable for Initiator Bus Requests<br>1 = Enabled<br>0 = Disabled<br>Read/Write |
| | | 5 | Reserved |
| | | 6 | BERR# Driver Enable for Initiator Internal Errors<br>1 = Enabled<br>0 = Disable<br>Read/Write |
| | | 7 | BINIT# Driver Enable<br>1 = Enabled<br>0 = Disabled<br>Read/Write |
| | | 8 | Output Tristate Enabled<br>1 = Enabled<br>0 = Disabled<br>Read |
| | | 9 | Execute BIST<br>1 = Enabled<br>0 = Disabled<br>Read |
| | | 10 | AERR# Observation Enabled<br>1 = Enabled<br>0 = Disabled<br>Read |
| | | 11 | Reserved |
| | | 12 | BINIT# Observation Enabled<br>1 = Enabled<br>0 = Disabled<br>Read |
| | | 13 | In Order Queue Depth<br>1 = 1<br>0 = 8<br>Read |

| Register Address | | Register Name | Bit Description |
|------|-----|------|------|
| Hex | Dec | | |
| | | 14 | 1 Mbyte Power-on Reset Vector |
| | | | 1 = 1 Mbyte |
| | | | 0 = 4 Gbytes |
| | | | Read |
| | | 15 | FRC Mode Enable |
| | | | 1 = Enabled |
| | | | 0 = Disabled |
| | | | Read |
| | | 17:16 | APIC Cluster ID |
| | | | Read |
| | | 19:18 | Reserved |
| | | 21:20 | Symmetric Arbitration ID |
| | | | Read |
| | | 24:22 | Clock Frequency Ratio |
| | | | Read |
| | | 25 | Reserved |
| | | 26 | Low Power Enable |
| | | | Read/Write |
| | | 63:27 | Reserved |

*Bit 0 of this register has been redefined several times, and is no longer used in the Pentium Pro processor.

## 4. GTL+ Minimum Overshoot Specification

The minimum overshoot specified in Table 11-12 in the *Pentium*® *Pro Family Developer's Manual, Volume 1: Specifications* is incorrectly stated as 0.55 mV. This value (T#α in Figure 11-10) should be **100** mV, as specified in the *Pentium*® *Pro Processor at 150 MHz, 166 MHz, 180 MHz, and 200 MHz* datasheet.

## 5. Conditional Move Opcode Incorrect

In Section 11.3 of the *Pentium*® *Pro Family Developer's Manual, Volume 2: Programmer's Reference Manual*, the description of the opcode for *CMOVcc—Conditional Move* should be 0F 4*x* /r, rather than 0F 4*x* cw/cd.

## 6. 32 Entries for 4-Kbyte Pages in ITLB

In Table 11-1, *Characteristics of the Caches, TLBs, and Write Buffer in the First Version of the Pentium*® *Pro Processor*, the instruction TLB is listed as having 64 entries for 4-Kbyte pages. It should list the instruction TLB as having 32 entries for 4-Kbyte pages, 4-way set associative.

## 7. A5# Value Should be H for Arb Id 1

In Table 9-3, *Arbitration ID Configuration*, in the *Pentium® Pro Family Developer's Manual, Volume 1: Specifications*, lists an A5# value of L for Arb Id 1. This should be an H.

## 8. Register Names Incorrect for CPUID Return Value

In Table 11-7, *Information Returned by CPUID Instruction*, in the *Pentium® Pro Family Developer's Manual, Volume 2: Programmer's Reference Manual*, the initial EAX value of 0 is shown to return "inel" into the ECX register and "ntel" into the EDX register. These names are reversed; the "inel" is returned into EDX, and the "ntel" is returned into ECX, as described in the text below the table.

## 9. Corrections to Volume 2: Programmer's Reference Manual

A number of typos and other documentation errors will be corrected in the next revision of the *Pentium® Pro Family Developer's Manual, Volume 2: Programmer's Reference Manual*. A list of significant changes is given below. Note that other changes may be made, and not all significant changes may be listed here.

- Page 2-6: There are two Dispatch/Execute Units depicted in Figure 2-1. The lower of the two should be listed as an **Instruction Pool (Reorder Buffer)**.

- Page 6-16: In Figure 6-1, *Operation of the PUSH Instruction*, the position of the doubleword value after pushing is one doubleword higher than depicted (at **n - 4**, not at n - 8).

- Page 6-17: The third and fifth sentences of the last paragraph reference the address-size attribute of the stack, and should reference the **operand**-size attribute.

- Page 6-29: The last sentence of Section 6.9.1.3, *Return-From-Interrupt Instruction*, is incorrect and will be deleted.

- Page 11-69: Under *Flags Affected*, the ZF flag is set if the values in the destination operand and register AL, AX, or EAX are **equal**.

- Page 11-82: The example for the DAA instruction is incorrect, and should read:

```
 ADD AL, BL  Before: AL=79H  BL=35H  EFLAGS(0SZAPC)=XXXXXX
             After:  AL=AEH  BL=35H  EFLAGS(0SZAPC)=110000
 DAA         Before: AL=2EH  BL=35H  EFLAGS(0SZAPC)=110000
             After:  AL=04H  BL=35H  EFLAGS(0SZAPC)=X00101
```

- Page 11-101: The first entry in the table under the ST(0) header should be −∞.

- Page 11-104: The last sentence of the description for the FCHS instruction should read "The following table shows the results obtained when **changing the sign** of various classes of numbers."

- Page 11-122: The first entry of the table under the +0 header should be −∞. Under the +∞ header, the entries for a SRC of +I or +F should be **+0**, not +∞.

- Page 11-143: The description of the FMULP instruction is incorrect; the result is actually stored in **ST(1)**.

- Page 11-183: The description of the FSUBRP ST(i), ST(0) instruction is incorrect; it subtracts **ST(*i*)** from **ST(0)**, stores the result in ST(*i*), and pops the register stack.

- Page 11-184: The row and column headers for the table should be interchanged.

- Page 11-211: The operation of the INC instruction is DEST ← DEST **+** 1.

- Page 11-233: The TASK-RETURN parameters are (* PE=1, VM=**0**, NT=1 *).

intel®

- Page 11-241:  Near and far jumps are sometimes referred to as intrasegment and intersegment **jumps**, not calls.

- Page 11-243:  The last first-level IF statement on this page should start IF far **jump**...

- Page 11-250/276:  On these two pages, the tables give Type 6 the name "16-bit trap gate" and Type 7 the name "16-bit interrupt gate".  These names are reversed.  This also applies to Types E and F.

- Page 11-292:  For the MOVZX instruction, the second paragraph of the description should be deleted.  The first paragraph should state that the instruction **zero** extends the value.

- Page 11-308:  The second paragraph of the description should begin "The current **operand**-size attribute..."

- Page 11-328:  For the RDMSR instruction, a #GP fault will not be generated if the current privilege level is not 0 in the Real Address Mode.  The first condition under Real Address Mode Exceptions should be deleted.

## 10.  Remapping WB, WT Memory Types

On page 11-21 of the *Pentium® Pro Family Developer's Manual, Volume 3: Operating System Writer's Guide*, Section 11.11.6, *Remapping Memory Types*, rule 2 states that the write-through type cannot be mapped into the write-back type.  This is incorrect; the statement should state that the write-back type cannot be mapped into the write-through type.

## 11.  BTM "From" and "To" Fields Reversed

In Section 5.2.3.5 (Branch Trace Message) of the *Pentium® Pro Family Developer's Manual, Volume 1: Specifications*, the information provided on the data lines is documented in reverse order.  The document should specify that D[63:32]# contain either the address of the first byte of the branch instruction or the address of the instruction immediately following the branch, and D[31:0]# contain the linear address of the target.

## 12.  Numeric Underflow Exception Description Clarification

Chapter 7 of the *Pentium® Pro Family Developer's Manual, Volume 2*, Section 7.8.5 on page 7-50,51 describes the Numeric Underflow Exception. There is a typographical error in the first paragraph of this description.

The second line of the first paragraph on page 7-51 reads: "Here, if overflow occurs again, after the result has been biased, a properly signaled 0 is stored in the destination operand". This sentence should read "Here, if *underflow* occurs again, after the result has been biased, a properly signaled 0 is stored in the destination operand."

## 13.  Page Directory Pointer Table Register Present Bit Identification

In the *Pentium® Pro Family Developer's Manual, Volume 3, Operating System Writer's Manual*, Chapter 3, '*Protected-Mode Memory Management*' contains Figures 3-20 and 3-21 on pages 3-31 and 3-32 respectively, which are visual representations of the bit definitions for the Page Directory Pointer Table Register entries. Bit 0 of this register should be labeled "P" signifying a "Present Bit", and needs to be set to '1' anytime Extended Physical Addressing mode is used.