

# Intel<sup>®</sup> 81341 and 81342 I/O Processors

Specification Update

---

*November 2008*

**Notice:** The Intel<sup>®</sup> 81341 and 81342 I/O processors may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Order Number: 315042-008US



INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. Intel products are not intended for use in medical, life saving, life sustaining, critical control or safety systems, or in nuclear facility applications.

Intel may make changes to specifications and product descriptions at any time, without notice.

Intel Corporation may have patents or pending patent applications, trademarks, copyrights, or other intellectual property rights that relate to the presented subject matter. The furnishing of documents and other materials and information does not provide any license, express or implied, by estoppel or otherwise, to any such patents, trademarks, copyrights, or other intellectual property rights.

Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.

Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. See [http://www.intel.com/products/processor\\_number](http://www.intel.com/products/processor_number) for details.

The Intel® 81341 and 81342 I/O processors may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Hyper-Threading Technology requires a computer system with an Intel® Pentium® 4 processor supporting Hyper-Threading Technology and an HT Technology enabled chipset, BIOS and operating system. Performance will vary depending on the specific hardware and software you use. See <http://www.intel.com/info/hyperthreading/> for more information including details on which processors support HT Technology.

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

Copies of documents which have an order number and are referenced in this document, or other Intel literature may be obtained by calling 1-800-548-4725 or by visiting Intel's website at <http://www.intel.com>.

Intel, Intel logo, Intel StrataFlash, Intel XScale are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

The ARM\* and ARM Powered logo marks (the ARM marks) are trademarks of ARM, Ltd., and Intel uses these marks under license from ARM, Ltd.

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

Copyright © 2008, Intel Corporation. All Rights Reserved.



## Contents

---

<b>Revision History</b> .....	4
<b>Preface</b> .....	5
<b>Summary Table of Changes</b> .....	6
<b>Identification Information</b> .....	12
<b>Core Errata</b> .....	14
<b>Non-Core Errata</b> .....	23
<b>Specification Changes</b> .....	37
<b>Specification Clarifications</b> .....	38
<b>Documentation Changes</b> .....	44



## Revision History

---

Date	Version	Description
November 2008	008	<ul style="list-style-type: none"><li>Added Core Errata <a href="#">11</a>.</li><li>Revised Specification Clarification <a href="#">10</a>.</li></ul>
September 2008	007	<ul style="list-style-type: none"><li>Added Specification Clarification <a href="#">10</a>.</li></ul>
July 2008	006	<ul style="list-style-type: none"><li>Added Document Change <a href="#">8</a>.</li><li>Added Specification Clarification <a href="#">9</a>.</li><li>Added Non-Core Errata <a href="#">28 - 29</a>.</li><li>Added second paragraph to Non-Errata <a href="#">18</a> Implication.</li><li>Replaced text after "When using C-code, it would look like this:" in Non-Errata <a href="#">19</a>.</li></ul>
October 2007	005	<ul style="list-style-type: none"><li>Added Core Errata <a href="#">10</a>.</li><li>Added Non-Core Errata <a href="#">24 - 27</a>.</li><li>Added Specification Change <a href="#">2</a> and <a href="#">3</a>.</li><li>Added Specification Clarification <a href="#">7</a> and <a href="#">8</a>.</li><li>Revised Specification Clarification <a href="#">2</a>.</li><li>Added Document Change <a href="#">3</a> through <a href="#">6</a>.</li></ul>
March 2007	004	<ul style="list-style-type: none"><li>Revised Errata C1 column with appropriate "x" steppings.</li></ul>
March 2007	003	<ul style="list-style-type: none"><li>Revised Core Errata <a href="#">7</a> with no "x" in steppings.</li><li>Revised Non-Core Errata <a href="#">23</a> with no "x" in steppings.</li></ul>
February 2007	002	<ul style="list-style-type: none"><li>Added Core Erratum <a href="#">7 - 9</a>.</li><li>Added non-core Erratum <a href="#">21 - 23</a>.</li><li>Added Specification Change <a href="#">1</a>.</li><li>Added Specification Clarification <a href="#">6</a>.</li><li>Added Documentation Change <a href="#">1</a> and <a href="#">2</a>.</li><li>Updated <a href="#">Table 1</a>, "Die Details" on <a href="#">page 12</a>.</li><li>Updated non-core Erratum <a href="#">8</a>, <a href="#">15</a>, <a href="#">18 - 20</a>.</li><li>Updated Specification Clarification <a href="#">1 - 3</a> and <a href="#">5</a>.</li></ul>
September 2006	001	Launch Release.



## Preface

---

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

Information types defined in Nomenclature are consolidated into the specification update and are no longer published in other documents.

This document may also contain information that was not previously published.

## Affected Documents/Related Documents

Title	Order
<i>Intel® 81341 and 81342 I/O Processors Developer's Manual</i>	315037
<i>Intel® 81341 and 81342 I/O Processors Datasheet</i>	315039

## Nomenclature

**Errata** are design defects or errors. These may cause the Intel® 81341 and 81342 I/O Processors behavior to deviate from published specifications. Hardware and software designed to be used with any given stepping must assume that all errata documented for that stepping are present on all devices.

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

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

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

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



## Summary Table of Changes

---

The following table indicates the errata, specification changes, specification clarifications, or documentation changes which apply to the Intel® 81341 and 81342 product. Intel may fix some of the errata in a future stepping of the component, and account for the other outstanding issues through documentation or specification changes as noted. This table uses the following notations:

### Codes Used in Summary Table

#### Stepping

X: Errata exists in the stepping indicated. Specification Change or Clarification that applies to this stepping.

(No mark)  
or (Blank box): This erratum is fixed in listed stepping or specification change does not apply to listed stepping.

#### Page

(Page): Page location of item in this document.

#### Status

Doc: Document change or update will be implemented.

Fixed: This erratum has been previously fixed.

NoFix: There are no plans to fix this erratum.

Plan Fix: This erratum may be fixed in a future stepping of the product.

#### Row



Change bar to left of table row indicates this erratum is either new or modified from the previous version of the document.



## Core Errata

No.	Steppings					Page	Status	ERRATA
	A0	A1	B0	C0	C1			
1	X	X	X	X	X	14	No Fix	L1 Cache maintenance operations close to memory operations can overlock the L1 D-Cache
2	X	X	X	X	X	15	No Fix	Locking all 8 ways in L2 causes core to hang
3	X	X	X	X	X	15	No Fix	Debugger: hold_reset cannot be set before reset is asserted
4	X	X	X	X	X	16	No Fix	A rejected request is retried out of order at the core interface
5	X	X	X	X	X	17	No Fix	Aborted PLD sets lock bit
6	X	X	X	X	X	18	No Fix	In Special Debug State, data can return twice to a register for an aborting load
7	X	X	X	X		19	Fixed	When running the 1200 MHz Intel® 8134x I/O Processors in User Mode, back-to-back data access transactions resulting in data Translation Look-aside Buffer (dTLB) hits to different pages, can result in the second access getting the permissions from the first access
8	X	X	X	X	X	20	No Fix	The Intel XScale® Microarchitecture hangs in an infinite loop under a specific sequence of loads and stores to the Data Cache Unit
9	X	X	X	X	X	21	No Fix	The Intel XScale® Microarchitecture hangs under a specific sequence of loads and stores to the Data Cache Unit when using Strongly Ordered Memory
10	X	X	X	X	X	22	No Fix	Targeted core reset requires CP6 read on victim core
11	X	X	X	X	X	22	Doc	No ALIGN transactions generated when the SAS/SATA Tunneling Protocol (STP) connection is originated by the storage device (drive, tape drive, etc.)



## Non-Core Errata (Sheet 1 of 2)

No.	Steppings					Page	Status	ERRATA
	A0	A1	B0	C0	C1			
1	X	X	X	X	X	23	No Fix	When a CRC transfer is being executed and a parity error occurs on fetch, then the parity error does not get logged
2	X	X	X	X	X	23	No Fix	When using a Dual-interface Intel® 81341 and 81342 I/O Processors and the PCI-X interface is in Central Resource mode, the ATUX can claim configuration cycles
3	X	X	X	X	X	23	No Fix	The MCU cannot use CAS=3 with DDR2 and ODT enabled
4	X	X	X	X	X	23	No Fix	GNT# to FRAME# timing violation by one clock in ATUX
5	X	X	X	X	X	24	No Fix	ATUE ATUCR[4] functionality has been defeatured
6	X	X	X	X	X	24	No Fix	Spurious DMA0 End-Of-Transfer Interrupt
7	X	X	X	X	X	24	No Fix	Data parity errors occurring on an inbound write transactions are not be logged by the South Internal Bus (IB) to North IB bridge (XBG)
8	X	X	X	X	X	25	No Fix	PCI outputs might not float within 40ns after P_RST# asserts
9	X	X	X	X	X	25	No Fix	In Central Resource (CR) Mode, Intel® 81341 and 81342 I/O Processors is not in Interface Low Power (ILP) state during PCI bus reset
10	X	X	X	X	X	26	No Fix	When operating in Root Complex mode, a second write to the Slot Power fields in the Slot Capability register may not generate a set_slot_power_limit message
11	X	X	X	X	X	26	No Fix	With default drive strengths, the General Purpose output pads may fail to meet the minimum VOH1 specification (2.6 V) at maximum IOH (-10 mA)
12	X	X	X	X	X	26	No Fix	The MCU may log a write data parity error twice
13	X	X	X	X	X	26	No Fix	There is an internal timing violation through PMMRBAR
14	X	X	X	X	X	27	No Fix	The ATUISR[0] (Master Data Parity Error Interrupt) and ATUSR[8] (Master Data Parity Error) do not get set for CFGWR TYPE 1 transactions
15	X	X	X	X	X	27	No Fix	ATUE maximum allowable lane skew of 20 ns is not being met
16	X	X	X	X	X	28	No Fix	Following the de-assertion of WARM_RST#, the ATU-E can lock up, which can only be exited with a full reset (assertion of P_RST#)
17	X	X	X	X	X	28	No Fix	There is a one XCLK (Internal Bus clock) window in which an inter-core-interrupt (CP write to SINTGENR) may be lost (fails to set the designated bit in IPIPNDR)
18	X	X	X	X	X	29	No Fix	The PCIXCAP pin is now de-featured. This pin can either be left connected normally or tied to VCC
19	X	X	X	X	X	30	No Fix	When the processor is configured as a Root Complex and connected to a card that is strapped to a link width smaller than its connector (e.g., x8 card strapped to a x4 width), the link does not train correctly
20	X	X	X	X	X	32	No Fix	The PCI memory window is not retained following preemptive resets



## Non-Core Errata (Sheet 2 of 2)

No.	Steppings					Page	Status	ERRATA
	A0	A1	B0	C0	C1			
21	X	X	X	X	X	32	No Fix	Hot-Swap Next Item Pointer (HS_NXTP) Register does not reset
22	X	X	X	X	X	33	No Fix	PBI read data gets corrupted when reading PBI MMRs while there is actual PBI read activity on the bus
23	X	X	X	X		33	Fixed	There is an internal design issue with the upper 32-bits of the PCI/PCI-X bus on the 81341/81342
24	X	X	X	X		33	No Fix	PCI Express lane reversal does not occur when in PCI Express root complex mode
25	X	X	X	X	X	34	No Fix	The SMBus reads are not supported
26	X	X	X	X	X	34	No Fix	The SMBus unit (SMU) implements the Packet Error Check (PEC) incorrectly
27	X	X	X	X	X	35	No Fix	An ADMA transfer fails when using the "Chain Resume" command
28	X	X	X	X	X	36	No Fix	The MSI-X Capability ID is incorrect for IOP341 and IOP342
29	X	X	X	X	X	36	No Fix	Inbound MSI gets lost when core is simultaneously writing to IMIPRx to clear a previous interrupt



## Specification Changes

No.	Steppings					Page	Status	ERRATA
	A0	A1	B0	C0	C1			
1	X	X	X	X	X	29	Doc	PCIXCAP pin is now defeatured in Intel® 81341 and 81342 I/O Processors
2	X	X	X	X	X	37	Doc	The Maximum Specification for the REFCLK Rise Time and Fall Time changed from 350ps to 700ps to match the PCI Express Specification
3	X	X	X	X	X	37	Doc	The VIH max level is changing from 1.8V to 2.0V for the p_mode2 and p_clkln pins

## Specification Clarifications

No.	Steppings					Page	Status	SPECIFICATION CLARIFICATIONS
	A0	A1	B0	C0	C1			
1	X	X	X	X	X	38	No Fix	ATUe and ATU-X Outbound Memory Windows overlap at power-on
2						38	Doc	DLLRCVER values are not listed in the Developer's Manual and are different for each stepping of the silicon
3	X	X	X	X	X	39	No Fix	Adequate delay required when programming some Memory-Mapped Registers
4	X	X	X	X	X	40	No Fix	During power-up, it is normal to see a current and voltage fluctuations on the 3.3 V, 1.8 V and 1.2 V power supplies
5	X	X	X	X	X	40	No Fix	When the PCI Express bus is in Root Complex mode, Master Aborts do not occur on configuration reads to device numbers greater than 0
6	X	X	X	X	X	41	Fixed	Specification changes for the C0 1200MHz 81341/81342 production units
7	X	X	X	X	X	41	No Fix	The order in which the DMCU registers are programmed before running the 4GB DIMM init sequence is not clearly listed
8	X	X	X	X	X	42	Doc	Clarification on the initialization procedure for the circular queues
9	X	X	X	X	X	42	Doc	CPLD bus keepers must be turned off during compilation
10	X	X	X	X	X	43	No Fix	PCI-Express link degradation when the 8134x IOC is connected to the Intel® 5400 MCH



## Documentation Changes

No.	Document Revision	Page	Status	DOCUMENTATION CHANGES
1	315036 315038 315053	45	Doc	PCIXCAP pin is now defeatured in Intel® 81341 and 81342 I/O Processors
2	315037	45	Doc	The ATU-E chapter of the Developer's Manual, Section 3.3.3, Note #7 needs to be modified
3	315036 315037	45	Doc	The Introduction chapter of the 81341/342 Developer's Manual in the Application DMA Controller section states that ""The Intel XScale processor supports L2 cache hardware coherency"". This is not correct
4	315036 315037	46	Doc	In the 81341/342 Developer's Manual, the Peripheral Registers chapter, the "Send Queue Not Full" interrupt generation is not documented correctly
5	315036 315037	46	Doc	In the 81341/342 Developer's Manual, the ATU chapter, Embedded Bridge Functionality information does not support 4GB inbound windows
6	315045	47	Doc	The "Note" in the section on supported Flash is wrong
7	315045	47	Doc	The TVB6 and TBA6 parameters are not listed in the 81348 Datasheet
8	315045	48	Doc	The VCCPLL Pin Requirements section of the Datasheet have changed



## Identification Information

**Table 1. Die Details**

Stepping	Part Number	QDF (Q)/ Specification Number (SL)	Processor Speed (MHz)	Notes
A0	HP8 1342M0820	Q436	800/400	Pb Beta Samples: Two Core
A1	HP8 1342M0820	Q347	800/400	Pb Beta Samples: Two Core
B0	WP8 1342M1220	Q538	1200/400	PbFree Samples: Two Core
B0	WP8 1342M0820	Q539	800/400	PbFree Samples: Two Core
B0	WP8 1341M1220	Q540	1200/400	PbFree Samples: Single Core
B0	WP8 1341M0820	Q541	800/400	PbFree Samples: Single Core
C0	WP8 1342M1220	Q619	1200/400	Pb-Free Samples: Two Core
C0	WP8 1342M0820	Q620	800/400	Pb-Free Samples: Two Core
C0	WP8 1341M1220	Q621	1200/400	PbFree Samples: Single Core
C0	WP8 1341M0820	Q622	800/400	PbFree Samples: Single Core
C0	WP8 1342M1220	SL9N6	1200/400	Pb-Free Production: Two Core
C0	WP8 1342M0820	SL9N7	800/400	Pb-Free Production: Two Core
C0	WP8 1341M1220	SL9N8	1200/400	PbFree Production: Single Core
C0	WP8 1341M0820	SL9N9	800/400	PbFree Production: Single Core
C1	WP8 1342M1220	Q828	1200/400	Pb-Free Samples: Two Core
C1	WP8 1342M0820	Q829	800/400	Pb-Free Samples: Two Core
C1	WP8 1341M1220	Q830	1200/400	PbFree Samples: Single Core
C1	WP8 1341M0820	Q831	800/400	PbFree Samples: Single Core
C1	WP8 1342M1220	SL9XZ	1200/400	Pb-free Production: Two Core
C1	WP8 1342M0820	SL9Y2	800/400	Pb-free Production: Two Core
C1	WP8 1341M1220	SL9Y3	1200/400	Pb-free Production: Single Core
C1	WP8 1341M0820	SL9Y4	800/400	Pb-free Production: Single Core



**Table 2. Device ID Registers**

Device and Stepping	Processor Device ID (CP15, Register 0 - opcode_2=0)	ATU Revision ID	JTAG Device ID
<b>Two-core: 1200 MHz cores and 800 MHz cores, Dual-interface (PCIe and PCI-X) ATUe/ATUX Device ID: 0x3382/0x338A</b>			
A0	0x69056010	0x0	0x01207013
A1	0x69056011	0x1	0x11207013
B0	0x69056014	0x4	0x41207013
C0	0x69056818	0x8	0x81207013
C1	0x69056819	0x9	0x91207013
<b>Single-core: 1200 MHz core and 800 MHz core, Dual-interface (PCIe and PCI-X) ATUe/ATUX Device ID: 0x3380/0x3388</b>			
A0	0x69056010	0x0	0x01206013
A1	0x69056011	0x1	0x11206013
B0	0x69056014	0x4	0x41206013
C0	0x69056818	0x8	0x81206013
C1	0x69056819	0x9	0x91206013



## Core Errata

---

### 1. L1 Cache maintenance operations close to memory operations can overlock the L1 D-Cache

**Problem:** Memory operations when the L1 Data Cache is setup for locking (bit 0 of the L1 Data Cache lock register is set), can set the lock bit for a different line as well as the desired line within the same set. This occurs when the memory operation stalls in the Data Cache pipeline, due to a L1 cache maintenance instruction preceding the memory operation.

**Required Conditions:**

1. Enable DCU locking.
2. Do a L1 cache maintenance operation (defined below).
3. Immediately following the cache maintenance operation do a memory operation, this operation stalls in the D3 stage of the pipe by the cache maintenance operation.

**Implication:** A location in the same set of the L1 Data Cache, as the stalled memory operation in (3.), locks and is not available for allocation. No data is modified or corrupted.

**Note:**

- This affects locking into the L1 Data Cache only. Locking the L1 Instruction cache or the L2 cache are not affected.
- This bug can only occur during the process of locking data into the L1 Data Cache. Once the Data Cache has been locked down, the bug has no impact.

**Workaround:**

1. Do not lock into the L1 data cache, use the L2 cache instead.
2. Use the locking routine specified in the Manzano Software Developer's Guide.

Full list of affected L1 cache maintenance commands

```
mcr p15, 0, Rd, c7, c6, 1 Invalidate L1 D-cache by MVA
mcr p15, 0, Rd, c7, c10, 1 Clean L1 D-cache by MVA
mcr p15, 0, Rd, c7, c10, 2 Clean L1 D-cache by set/way
mcr p15, 0, Rd, c7, c14, 1 Clean and Invalidate L1 D-cache by MVA
mcr p15, 0, Rd, c7, c14, 2 Clean and Invalidate L1 D-cache by set/way
```

**Status:** No Fix. See the [Table](#) , "Summary Table of Changes" on page 6.



## 2. Locking all 8 ways in L2 causes core to hang

**Problem:** The replacement behavior, on a miss in the L2 cache, on a line where all 8 ways have been locked, is defined as "architecturally unpredictable". However, the implementation treats "all ways locked" as "no ways locked", and as such, normal replacement schemes (excluding lock behavior) dictate which line gets replaced.

There is a corner case associated with this. Normally, when all ways are locked in the L2, the lock bits are ignored and find the first way that is "not used & not pend" for selecting a replacement. However, in certain situations, there is not a way left that is "not used & not pend". The result of this is, that the default replacement is used and a way that is marked PEND is erroneously replaced.

**Implication:** Users are not allowed to lock all 8 ways of a given set. The maximum number of ways that can be locked in a given set are 7 out of 8.

**Workaround:** Do not lock all 8 ways of a given set in L2 cache.

**Status:** No Fix. See the Table , "Summary Table of Changes" on page 6.

## 3. Debugger: hold\_reset cannot be set before reset is asserted

**Problem:** The hold\_reset bit (accessed through JTAG in DCSR data register) is what the debugger uses to hold the core in reset, while the debugger takes its time downloading the debug handler into the SRAM. Do not set before reset is asserted.

**Implication:** A debugger must only set the hold\_reset bit after the core has already entered reset. Setting this bit while the core is not already in reset results in unpredictable behavior of the core.

**Workaround:** Ensure the core has entered reset before setting hold\_reset.

**Status:** No Fix. See the Table , "Summary Table of Changes" on page 6.



#### 4. A rejected request is retried out of order at the core interface

**Problem:** A rejected request is retried out of order at the core interface.

**Required Conditions:**

1. The following code sequence (back-to-back, with no intervening instructions).
  - a. Any memory operation (load, store, cache management operation, etc.).
  - b. L2 cache control command (clean, invalidate, etc.).
  - c. L1 data cache clean command (by MVA or by set/way).
2. The memory buffers are full (heavy recent load/store traffic).
3. Two memory transactions complete within 1-2 cycles freeing up their buffers.

This can happen even when all pages are marked write-through. The clean command does not need to hit the cache or hit dirty data for the bug to occur.

**Implication:** When all the above conditions are hit, the L1 data cache clean command may proceed when there are no available buffers. Whether it hits or misses, it corrupts a buffer. This leads to unpredictable behavior. Possible symptoms:

- Dropped memory transaction (load or store).
- Part hangs.
- Incorrect data stored out.
- Incorrect data returned to a register.
- Incorrect data returned to a register.

**Workaround:** Separate L1 and L2 cache control commands by at least a single non-cache control instruction.

**Status:** No Fix. See the [Table](#) , “Summary Table of Changes” on page 6.



## 5. Aborted PLD sets lock bit

**Problem:** An aborted PLD can inadvertently set the lock bit.

**Required Conditions:**

- Enable L1 Data Cache locking.
- Do a PLD instruction that does not have permission.

**Implication:** This issue results in locations being locked in the cache incorrectly. When a PLD instruction that does not have permission to access a given memory region is used during data cache locking, then the DCU may incorrectly lock a "hole" into the data cache, effectively reducing the data cache size by one line.

- PLD instructions are defined to NOT abort - that is, when they do not have permission to access a line from memory, they are supposed to turn into a NOP and the data cache does not update. In this issue, the data cache is updated with a locked invalid line.

**Note:**

- This affects locking into the L1 Data Cache only. Locking the L1 Instruction cache or the L2 cache are not affected.
- This issue can only occur during the process of locking data into the L1 Data Cache. Once the Data Cache has been locked down, the bug has no impact.
- Data aborts taken while the lock bit is set are dangerous, because memory operations executed in the abort handler are locked.

**Workaround:** Use the L1 locking routine specified in the *Manzano Software Developer's Guide*. Contact your Intel Representative for more information on Manzano.

**Status:** **No Fix.** See the [Table](#) , "Summary Table of Changes" on [page 6](#).



## 6. In Special Debug State, data can return twice to a register for an aborting load

**Problem:** In SDS memory operations that normally abort and cause an event flush do not. Instead, they are supposed to cease operation, (in other words, clear the scoreboard bit, and set the "sticky" abort bit in the DCSR.) For certain aborting loads the DCU notices that the load wants to abort and clear the scoreboard bit, but also sends out a memory request for the aborting load; additionally, the DCU then returns the incorrect data to the register file.

**Required Conditions:**

1. There is an outstanding store in one of the DCU memory buffers that has not been globally observed.
2. A strongly ordered operation is executed (either a load or a store).
3. Before the strongly ordered operation in (2) has been globally observed, a load operation that aborts is executed. Because of (1) and (2) the DCU incorrectly returns twice to the register file.

**Implication:** This results in the DCU returning twice to a register for an aborting load in SDS (Special Debug State).

**Workaround:** Since this bug only occurs during SDS, a software workaround is appropriate:

### **Workaround #1:**

Before executing any load that "could" data abort, do a DCU DRAIN\_WRITEBUFFER (dwb) command (defined below)

### **Workaround #2:**

Because any load in SDS may abort and return incorrect data to a register the "sticky" abort bit should be checked in the DCSR before using the data from the load in question. If before any of the data from the loads is used a DCU\_DRAIN (defined below) command is executed, then this bug does not occur. Additionally, doing a DCU\_DRAIN command before reading the "sticky" abort bit in the DCSR has the added benefit that any imprecise data aborts on the loads have set the abort bit before the data is used.

- a. Do any number of loads/stores
- b. Before the data from the loads is used, do the following:
  - DCU\_DRAIN
  - Read the "sticky" abort bit and verify that no aborts have occurred
  - Use the data from the loads

A DCU\_DRAIN is defined as:

```
mrc    p15, 0, r3, c1, c0, 0 @# read CP15 control register
mcr    p15, 0, r3, c1, c0, 0 @# write CP15 control register
[the mcr stalls until the L1 DCache buffers have completely drained]
```

A DCU\_DRAIN\_WRITEBUFFER (dwb) is defined as:

```
mcr    p15, 0, r0, c7, c10, 4 @# drain write buffer (dwb)
```

**No Fix.** See the [Table](#) , "Summary Table of Changes" on page 6.



**7. When running the 1200 MHz Intel® 8134x I/O Processors in User Mode, back-to-back data access transactions resulting in data Translation Look-aside Buffer (dTLB) hits to different pages, can result in the second access getting the permissions from the first access**

**Problem:** When running the 1200 MHz Intel® 8134x I/O Processors in User Mode, back-to-back data access transactions resulting in data Translation Look-aside Buffer (dTLB) hits to different pages, can result in the second access getting the permissions from the first access.

**Implication:** This has resulted in various failing signatures involving valid and invalid abort exception conditions.

**Note:** This erratum only affects software that runs in USER mode, for example Linux\*. Operating systems or firmware that runs completely in SUPERVISOR mode, for example Windows CE 5.0\* or VxWorks\*, never execute code that is affected.

**Workaround:** Lock down all but one of the Data TLB entries to an unused address space so that back-to-back dTLB hits are not possible.

- or -

Configure the software to only run in Supervisor mode, which eliminates that permission checking on dTLB accesses.

**Status:** Fixed in C1. See the [Table](#) , "Summary Table of Changes" on page 6.



## 8. The Intel XScale® Microarchitecture hangs in an infinite loop under a specific sequence of loads and stores to the Data Cache Unit

**Problem:** With the L1 Data Cache cacheable -or- L1 Data Cache uncacheable and the L2 Cache cacheable, a specific sequence of loads and stores to the Data Cache Unit and a specific sequence of data being returned within an extremely narrow timing window, causes the Data Cache Unit to hang in an infinite loop.

Required Conditions for replicating the issue with and without evictions (All Loads/Stores are to L1 cacheable memory -or- L1 uncacheable and L2 cacheable memory):

1. A load misses the L1 data cache and must cause an eviction of a clean cacheline in the same set. However, this 1st load is not necessary (which means the L1D eviction is not a necessary condition).
2. A store is being retried by the Data Cache Unit due to a backed up Bus Interface Unit.
3. Four loads to the same cacheline that miss the data cache.
4. The following timing conditions must then happen:  
Must have some other unrelated instructions with no register dependencies occurring between #3 and #5.
5. A load, which is to the same cacheline as the store in #2, misses the data cache.
  - a. The first load in #3 gets its data back into the Data Cache Unit exactly one cycle after this load gets executed.
  - b. The store in #2 still retries from the Data Cache Unit and one of the attempts happens exactly one cycle after this load gets executed.
  - c. No dependencies on previous loads.
6. A load, which is to the same cacheline as the four loads in #3, misses the data cache.
  - a. No dependencies on previous loads.
  - b. Must be back-to-back with #5.
7. A load operation to any address.
  - a. Must be back-to-back with #6.

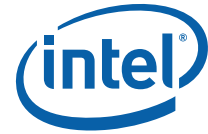
**Implication:** When the Data Cache Unit encounters this condition, it enters an infinite loop, resulting in no data being returned and thereby, hanging the Intel XScale® microarchitecture.

**Note:**

- This issue has only been observed on the 81348/81341 and 81342 when running hand-tuned focus tests written to specifically reproduce this issue.
- The only way to exit this condition is to reset the Intel XScale® microarchitecture.
- Independent of L1 data cache operating in write through or write back mode.
- Independent of L2 cache being present or not.
- Independent of Bus timings/ratios.

**Workaround:** No workaround. However, there is a recovery mechanism in the unlikely event that it does occur. When a lockup does occur, the only way to exit this condition is to reset the Intel XScale® microarchitecture. Examining the trace buffer after a reset gives information on the code sequencing (within 16 clock cycles) leading to the lockup. Once identified, make code corrections to eliminate the condition as listed in the "Required Conditions" section above, that created the lockup.

**Status:** No Fix. See the [Table](#) , "Summary Table of Changes" on page 6.



## 9. The Intel XScale® Microarchitecture hangs under a specific sequence of loads and stores to the Data Cache Unit when using Strongly Ordered Memory

**Problem:** With the L1 Data Cache cacheable -or- L1 Data Cache uncacheable and the L2 Cache cacheable, a specific sequence of loads and stores to the Data Cache Unit with a strongly ordered load and a specific sequence of data being returned within an extremely narrow timing window, causes the Data Cache Unit to hang, thereby hanging the Intel XScale® microarchitecture.

Required Conditions (All Loads/Stores are to L1 cacheable memory -or- L1 uncacheable and L2 cacheable memory):

1. A load misses the data cache and must cause an eviction of a dirty cacheline.
  - a. This eviction has to land between #2 and #3.
2. Four loads to the same cacheline that miss the data cache. No dependencies on previous loads.
3. A memory access that is Strongly Ordered within a few instruction cycles of #2. No dependencies on loads in #2.

**Implication:** When the Data Cache Unit encounters this condition, the load to strongly ordered memory is dropped, never returning the data and thereby, hanging the Intel XScale® microarchitecture.

Notes:

- This issue has only been observed on the 81348/81341 and 81342 when running hand-tuned focus tests written to specifically reproduce this issue.
- The only way to exit this condition is to reset the Intel XScale® microarchitecture.
- Independent of L1 data cache operating in write through or write back mode.
- Independent of L2 cache being present or not.
- Independent of Bus timings/ratios.

Workaround:

1. Convert all "Strongly Ordered Memory" operations into "Device Memory" operations inside the Kernel.  
Where memory is configured as Strongly Ordered (in the page table, the following bits are set TEX=000b and CB=00b) the change the memory to be configured as Device Memory (in the page table, the following bits are set (TEX=010b and CB=00b for non-shared device memory) -or- (TEX=001b and CB=01 for shared device memory)).  
Small Page Descriptors cannot specify the TEX attribute. In this case, the TEX attribute defaults to 0. When CB=00b is used with Small Page Descriptors, a Strongly Ordered Memory region is created. Therefore, Small Page Descriptors must not be used. Use Extended Small Page Descriptors instead.
2. 2.) This is more of a recovery mechanism (not a workaround) in the unlikely event that does occur. When a lockup does occur, the only way to exit this condition is to reset the Intel XScale® microarchitecture. Examining the trace buffer after a reset gives information on the code sequencing (within 16 clock cycles) leading to the lockup. Once identified, make code corrections to eliminate the condition as listed in the "Required Conditions" section above, that created the lockup.

**Status:** No Fix. See the Table , "Summary Table of Changes" on page 6.



## 10. Targeted core reset requires CP6 read on victim core

**Problem:** Targeted core reset requires CP6 read on victim core.

**Implication:** At times a core reset initiated from TPMI MMR TCSRx or from EIBS CP register TARRSTR fails to reset the core.

**Workaround:** It has been observed that a CP6 read idles the BIU and the busy signal deasserts.

- First interrupt the victim core to an ISR that does the CP6 read and then idles.
- Follow this with the reset.

**Status:** No Fix. See the Table , "Summary Table of Changes" on page 6.

## 11. No ALIGN transactions generated when the SAS/SATA Tunneling Protocol (STP) connection is originated by the storage device (drive, tape drive, etc.)

**Problem:** When the 8134x originates the STP connection, STP-specific ALIGNs, 2 per 256 d-words (2/256), transmit correctly. When the STP connection originates with the storage device, which happens when Native Command Queuing (NCQ) is used, the STP-specific ALIGN rate is not within spec (2/256).

**Implication:** The connection to the device fails.

**Workaround:** To solve the problem, the 2.0.3 release of the Transport Firmware has been modified to increase the general skew management rate to 1/120. The 8134x has the following ALIGN rates:

- "2/256 + 1/120 for STP connections initiated by the 81348.
- "1/120 for STP connections initiated by the storage drive (approximately equal to required 2/256 + 1/2048).

**Status:** Doc. Software fix available in Maintenance Release 3 (December 2008). See the Table , "Summary Table of Changes" on page 6.



## Non-Core Errata

---

### 1. When a CRC transfer is being executed and a parity error occurs on fetch, then the parity error does not get logged

**Problem:** When a CRC transfer is being executed with ADCR[10], CRC Transfer Disable bit, set and a parity error occurs on fetch, then the parity error does not get logged.

**Implication:** Parity error does not get logged, but the expectation is that the CRC fails, so the error is still detected.

**Workaround:** No workaround.

**Status:** No Fix. See the Table , "Summary Table of Changes" on page 6.

### 2. When using a Dual-interface Intel® 81341 and 81342 I/O Processors and the PCI-X interface is in Central Resource mode, the ATUX can claim configuration cycles

**Problem:** When using a Intel® 81341 and 81342 I/O Processors and the PCI-X interface is in Central Resource mode, the ATUX can claim configuration cycles from an Endpoint device.

**Implication:** This is not a likely scenario because Endpoint devices do not issue configuration cycles to the central resource.

**Workaround:** Do not connect IDSEL in Central Resource mode configuration.

**Status:** No Fix. See the Table , "Summary Table of Changes" on page 6.

### 3. The MCU cannot use CAS=3 with DDR2 and ODT enabled

**Problem:** The MCU cannot use CAS=3 with DDR2 and ODT enabled.

**Implication:** No implication. This is a documentation clarification and provided for information purposes.

**Workaround:** When using ODT, CAS latency must be set to four or greater.

**Status:** No Fix in B0. See the Table , "Summary Table of Changes" on page 6x.

### 4. GNT# to FRAME# timing violation by one clock in ATUX

**Problem:** The GNT# to FRAME timing is in violation of the *PCI-X Addendum to the PCI Local Bus Specification*, Revision 2.0,. This violation can occur only when the ATU-X masters back-to-back requests. When there is another PCI device that wants the bus, GNT# rotates away from Intel® 81341 and 81342 I/O Processors. So, in a system with balanced bus traffic this violation is less likely to occur.

**Implication:** Although a violation of *PCI-X Addendum to the PCI Local Bus Specification*, Revision 2.0, it is for a performance issue which other PCI devices tolerate. Intel® 81341 and 81342 I/O Processors taking an extra clock to assert FRAME# is not a functional problem.

**Workaround:** No workaround.

**Status:** No Fix. See the Table , "Summary Table of Changes" on page 6.



## 5. **ATUE ATUCR[4] functionality has been defeated**

**Problem:** The "Inbound Minimum Completion Size" feature of the ATUCR (bit 4) has been defeated in the ATUE.

**Implication:** The "Inbound Minimum Completion Size" feature of the ATUCR (bit 4) cannot be enabled. ATUCR[4] needs to remain at the default value of '0'.

**Workaround:** No workaround.

**Status:** **No Fix.** See the [Table](#) , "Summary Table of Changes" on page 6.

## 6. **Spurious DMA0 End-Of-Transfer Interrupt**

**Problem:** When the interrupt controller goes from having no interrupts asserted to 1 or more asserted, there is a 1 clock cycle window in which the IINTVEC (IRQ Interrupt Vector register: CP6, Page 2, Register 3) or FINTVEC (FIQ Interrupt Vector register: CP6, Page 2, Register 4) may report the value of the INTBASE register (Interrupt Base register: CP6, Page 2, Register 0), which is the vector address for interrupt 0, DMA0 End-of-Transfer.

This condition can occur even if the DMA0 EOT interrupt is masked, INTCTLO.0 = 0.

**Implication:** No negative impact expected. If the interrupt service routine that reads the IINTVEC/FINTVEC qualifies the return value against IINTSRC0.0/FINTSRC0.0, it either sees there is nothing to do or it validly calls the DMA0 End-of-Transfer handler.

**Workaround:** If IINTVEC/FINTVEC equals INTBASE, then re-read the IINTVEC/FINTVEC register.

**Status:** **No Fix.** See the [Table](#) , "Summary Table of Changes" on page 6.

## 7. **Data parity errors occurring on an inbound write transactions are not be logged by the South Internal Bus (IB) to North IB bridge (XBG)**

**Problem:** During inbound transactions (transactions flowing from the South IB to the North IB) when the inbound request to XBG is corrupted with data parity error it is not being logged. The bug is exercised by writes, which would cross a 16B boundary and result in two XSI data phases with a parity error in the 1st phase (in other words, 32B writes.) Aligned DWORD (4B) writes, QWORD (8B) writes and OWORD (16B) writes are all unaffected, but anything larger could trigger the erratum. It affects only parity errors in the 1st data phase of a two data phase inbound write (in other words, bytes 0 through 15 of a 32B write.) It does not affect completions.

**Implication:** When a data parity error occurs during inbound write requests to the XBG, it is supposed to be logged. Because the data parity error is not logged, an XBG error interrupt does not occur.

**Workaround:** No workaround.

**Status:** **No Fix.** See the [Table](#) , "Summary Table of Changes" on page 6.



## 8. PCI outputs might not float within 40ns after P\_RST# asserts

**Problem:** Per the PCI Specification, when P\_RST# asserts, PCI outputs must be floated within 40 ns. PCI outputs might not float within 40 ns after P\_RST# asserts.

The issue is, that the divider codes to the PCI-X PLL can change after P\_RST# assertion, but before the PLL has been disabled (the divider codes are changing on-the-fly). This can mess with pp\_clk, which Intel® 81341 and 81342 I/O Processors is relying on to provide clocks for the pad OE flops, so that the PCI-X pads float within the required time after P\_RST# asserts.

**Implication:** The danger is not only the violation of the 40 ns float time in the specification, but the possibility of not turning off the OE signals. This would cause contention with the Central Resource and result in it not being able to configure the device properly (contention could occur on the initialization pattern signals, for example).

If the MCU is enabled there is even less of a chance of this happening since the powerfail sequence must complete before internal reset, and during the powerfail sequence the PCI output buffers have plenty of time to become disabled. Even with the MCU disabled, there is very little likelihood, and maybe none, of triggering an issue. This has never been seen in any post silicon validation or testing and therefore, unlikely to ever occur.

**Workaround:** When this problem surfaces, force the ATU-X off of the PCI bus before asserting P\_RST# (turn off bus master enable, turn off memory enable). This guarantees the PCI-X output enables are OFF when P\_RST# asserts.

**Status:** No Fix. See the Table , “Summary Table of Changes” on page 6.

## 9. In Central Resource (CR) Mode, Intel® 81341 and 81342 I/O Processors is not in Interface Low Power (ILP) state during PCI bus reset

**Problem:** When in CR Mode, the Intel® 81341 and 81342 I/O Processors PCI-X interface is not in the ILP state.

In Endpoint Mode, the Intel® 81341 and 81342 I/O Processors is in the low power state during P\_RST#, as expected.

In CR mode, the Intel® 81341 and 81342 I/O Processors leaves the low power state (receive enables on, ODT on) when EXTERNAL reset deasserts, but long before the P\_RST# deasserts.

**Implication:** None.

**Workaround:** No workaround.

**Status:** No Fix. See the Table , “Summary Table of Changes” on page 6.



**10. When operating in Root Complex mode, a second write to the Slot Power fields in the Slot Capability register may not generate a set\_slot\_power\_limit message**

**Problem:** The PCIe specification states that a set\_slot\_power\_limit message is generated by an upstream device on any configuration write to the Slot Capabilities register. The ATUE drops a second set\_slot\_power\_limit message when a second write to the Slot Capabilities register occurs before the initial set\_slot\_power\_limit message has been issued on PCIe link. This is not a problem when the Slot Power Limit values are the same in both writes.

**Implication:** A second write to the Slot Power fields in the Slot Capability register may not generate a set\_slot\_power\_limit message.

**Workaround:** Do not change the Slot Power Limit fields on subsequent writes to the Slot Capability register or verify the first message has been delivered, by reading the Captured Slot Power Limit fields in the Device Capabilities Registers of the downstream device, before changing the Slot Power Limit.

**Status:** No Fix. See the Table , "Summary Table of Changes" on page 6.

**11. With default drive strengths, the General Purpose output pads may fail to meet the minimum VOH1 specification (2.6 V) at maximum IOH (-10 mA)**

**Problem:** With default drive strengths, the General Purpose output pads may fail to meet the minimum VOH1 specification (2.6 V) at maximum IOH (-10 mA).

**Implication:** In high current source application, (in other words, sourcing current to an LED), the General Purpose output pads (all pads except for PCI, DDR and analog PHYs) may not meet the VOH1 2.6 V specification. They are however, capable of driving to 2.4 V at -10 mA. Also, there is no problem with the VOL1 specification when sinking IOL (+10 mA), so driving LEDs in an open-drain manner is OK.

**Workaround:** No workaround.

**Status:** No Fix. See the Table , "Summary Table of Changes" on page 6.

**12. The MCU may log a write data parity error twice**

**Problem:** Under some conditions the MCU may log a write data parity error twice, (for example, in 32-bit mode or during partial writes w/ ECC enabled.)

**Implication:** The "Parity Error N" bit (DMCISR[9]) can erroneously set some write data parity errors.

**Workaround:** No workaround.

**Status:** No Fix. See the Table , "Summary Table of Changes" on page 6.

**13. There is an internal timing violation through PMMRBAR**

**Problem:** There is an internal timing violation through PMMRBAR.

**Implication:** The need to use the default values of the PMMRBAR.

**Workaround:**

1. Do not write to this register, just use the default value.
2. Ensure that both the North and South Internal Bus are completely inactive for 20 bus clocks following any update to this register.

**Status:** No Fix. See the Table , "Summary Table of Changes" on page 6.



#### 14. **The ATUISR[0] (Master Data Parity Error Interrupt) and ATUSR[8] (Master Data Parity Error) do not get set for CFGWR TYPE 1 transactions**

**Problem:** The Master Data Parity status bit in ATUSR[8] and related interrupt ATUISR[0] are currently being set for CFG TYPE 0 transactions only and not for CFG TYPE 1 transactions.

From the *Intel® 81341 and 81342 I/O Processors Developer's Manual* definition for "Master Data Parity Error":

**Master Data Parity Error:** This bit is set by the ATU when its Parity Error Enable bit is set and either of the following two conditions occur:

- ATU receives a Poisoned Completion for an Outbound Read Request
- ATU transmits a Poisoned TLP for an Outbound Write Request.

**Implication:** This problem can only be observed when the Intel® 81341 and 81342 I/O Processors is a Root Complex and connected to a switch. When a parity error occurs in the internal data path, for an outbound CFGWR Type 1, the Intel® 81341 and 81342 I/O Processors PCIe port transmits the TYPE 1 CFG with the poison bit set, but it does not log this in ATUSR[8]. Since the target of the write drops the packet, log the reception of the poison TLP and, return an error completion the error condition is not lost. The user only sees the error logged in the switch.

**Workaround:** No workaround.

**Status:** **No Fix.** See the [Table](#) , "Summary Table of Changes" on page 6.

#### 15. **ATUE maximum allowable lane skew of 20 ns is not being met**

**Problem:** The maximum allowable lane skew of 20 ns is not being met. The PCIe electrical specification, Version 1.1, Section 5.3.5: "Lane-to-Lane Skew", defines the allowable Interconnect Lane-to-Lane Skew as 1.6 ns. The most lane-to-lane skew seen externally is 1.6 ns + 1.3 ns (Ltx=500+2UIps) = 2.9 ns.

The maximum lane-to-lane skew the device can tolerate is ~15.76 ns.

**Implication:** Although this is a specification violation, the impact is negligible as the expected maximum lane skew is on order of ~2.9 ns which leaves ~12.9 ns of margin.

**Workaround:** No workaround

**Status:** **No Fix.** See the [Table](#) , "Summary Table of Changes" on page 6.



**16. Following the de-assertion of WARM\_RST#, the ATU-E can lock up, which can only be exited with a full reset (assertion of P\_RST#)**

**Problem:** Following the de-assertion of WARM\_RST#, the ATU-E can lock up, which can only be exited with a full reset (assertion of P\_RST#).

**Implication:** The PCI-E link is completely unusable since no traffic passes through the ATU-E.

**Workaround:** When using WARM\_RST#, the maximum reset duration should not exceed 24 ms time.

Other workarounds:

1. When in Root Complex mode, after the WARM\_RST# is de-asserted, the software disables the link by writing a 1 to PE\_LCTCL [4] followed by a 0 to PE\_LCTCL [4]. This forces a retrain of the link.
2. When in Endpoint Mode, set LK\_DN\_RST\_BYPASS# to "0". Following WARM\_RST# deassertion, issue a Hot Reset to the 81341 or 81342. Setting the LK\_DN\_RST\_BYPASS# strap to '0' prevents a full-chip reset as a result of the hot reset message.

**Status:** No Fix. See the Table , "Summary Table of Changes" on page 6.

**17. There is a one XCLK (Internal Bus clock) window in which an inter-core-interrupt (CP write to SINTGENR) may be lost (fails to set the designated bit in IPIPNDR)**

**Problem:** There is a one XCLK window in which an inter-core-interrupt (CP write to SINTGENR) may be lost (fails to set the designated bit in IPIPNDR). This can occur when the receiving core is simultaneously clearing a prior interrupt in IPIPNDR (i.e., CoreX write to SINTGENR (targeting CoreY) concurrently with CoreY write to IPIPNDR.)

**Implication:** An inter-core interrupt may be lost.

**Workaround:**

1. Use the IMU instead  
OR
2. When sending interrupts, write the SINTGENR twice in rapid succession. Disable Interrupts so that the writes do not get separated. When clearing interrupts, ensure that writes to IPIPNDR do NOT occur in rapid succession (e.g., after an IPIPNDR write, perform an IPIPNDR read). Ensuring that the interrupt is sent twice in rapid succession (faster than the interrupt service routines can clear interrupts) ensures that one of the sent interrupts is NOT lost.

**Status:** No Fix. See the Table , "Summary Table of Changes" on page 6.



### 18. The PCIXCAP pin is now de-featured. This pin can either be left connected normally or tied to VCC

**Problem:** The PCIXCAP pin is now de-featured. This pin should be left connected normally or tied to VCC.

**Implication:** The value read from the PCSR might not accurately reflect the true state of the PCIXCAP pin.

The PCIXSR and PCIXCMD registers needs to be read to ascertain the PCI-X device status. Then a write to the PCSR is needed to set the PCI-X bus to the appropriate bus status.

When using P\_CLKIN as the main clock input here is a limited to 100 MHz or 133 MHz. Please see the "PCI Bus Frequency Initialization" table in the Developer's Manual. This is due to the de-features and tying the PCIXCAP pin high. PCI speed 'bits' in the PCSR register are write only when a 100 MHz differential clock is connected.

**Workaround:** Once the Intel XScale® processor is out of reset, program the PCSR for PCI-33 mode and then release the PCI bus from reset and do a bus scan.

The supported bus mode(s) can be ascertained by reading the following registers for each function on the bus segment:

PCI-X 133 (PCIXSR[17]). When set to '1', the device is 133 MHz capable.

PCI-X 66 (PCIXSR[17]). When set to '0', the device is not 133 MHz capable.

PCI-X 266 (PCIXCMD[13:12]). Indicates if Mode 2.

PCI-X 533 (not supported by SL).

PCI (no PCI-X Capability ID).

By doing a bus scan of the above registers, the processor can know what is on the bus, set PCSR[19:16] accordingly, and reset the PCI bus again while driving out the new initialization pattern. This relies on the fact that the ATU-X PCSR register is r/w, and can be set to operate the PCI bus in any supported mode. S/W can override the PCIXCAP pin.

Once the bus scan is complete, one of the following workarounds is required.

1. When the PCIXCAP pin is tied normally, the bus scan has no unexpected adverse side effects and is all that's required.
2. When the PCIXCAP pin is tied to VCC, then:
  - a. Latency timer register (ATULT[6]) has the wrong default value, but it is a r/w register so S/W can update if needed.
  - b. Fast back-to-back cap register (ATUSR[7]) has the wrong default value (incorrectly hardwired to '0' in PCI mode). It is RO so it cannot be changed by S/W, but it does not control any functionality. Additionally, setting this bit to a '1' is optional according to the PCI Spec (PCI 2.3, Section 3.4.2, Fast Back-to-Back Transactions).
  - c. The Outbound Configuration Address Register (OCCAR) does not have bits [15:11] cleared during outbound PCI configuration transactions. No PCI devices are expected to use these bits for configuration address decode, so the affect should be benign.

**Note:** An alternate workaround would be to implement PCIXCAP circuitry, which can be read by the Intel XScale® processor via GPIOs. Update the PCSR accordingly and reset the PCI bus.

**Status:** No Fix. See the [Table](#) , "Summary Table of Changes" on page 6.



**19. When the processor is configured as a Root Complex and connected to a card that is strapped to a link width smaller than its connector (e.g., x8 card strapped to a x4 width), the link does not train correctly**

**Problem:** When the Intel® 81341 and 81342 I/O Processors is configured as a Root Complex and connected to a card that is strapped to a link width smaller than its connector (e.g., x8 card strapped to a x4 width), the link does not train correctly.

**Implication:** In this case, the Intel® 81341 and 81342 I/O Processors sends TS1s with COM, LINKNUM, LANENUM on all eight lanes to the card. The card detects this as a specification violation, and goes from transmitting TS1s back to Electrical Idle. The Intel® 81341 and 81342 I/O Processors is not able to train with the card in this case unless the upper lanes are either physically taped off or they are turned off in the detection circuit in the PCIE hard macro by writing 0x0 to register address 0x3D to the unused lanes.

The PCI Express specification requires that the Root Complex sends TS1 ordered sets with COM, LANENUM, PAD on all lanes and then send COM, LINKNUM, LANENUM only on the lanes that received the same COM, LANENUM, PAD which it transmitted. The Root Complex must also send COM, PAD, PAD on the remaining lanes.

**Workaround:** The code for the workaround is listed below.

The macro definitions for these two macros is to just write to the addresses of the registers. The address is based on these formulas:

EIBADDR register address = MMR base address + Internal Bus MMR Address offset + 0x148 (EIBADDR register offset)  
EIBDATA register address = MMR base address + Internal Bus MMR Address offset + 0x14C (EIBDATA register offset)

When using C code, it would look like this:

1. Disable link in PE\_LCTL register.
2. Write to PCIE macro per Specification Update.
3. Enable link in PE\_LCTL register.
4. Poll for link training complete in PE\_LSTS register.

```

=====
PUBLIC STATUS bcmPcieLinkWidthSet(int linkWidth)
{
    int linkWidthCurrent;
    int uiLoop;
    UINT32 uiAddress;
    UINT32 uiData;
    UINT32 uiTimer = 0;

    linkWidthCurrent = bcmPcieLinkWidthGet();

    /* If the link width is already correct, skip setting it */
    if ((linkWidthCurrent == linkWidth) || (linkWidth == 0))
    {
        printf ("link width is already set to %d\n", linkWidth);
        return OK;
    }

    /* This method only lets us go from more lanes to less lanes. To reset to
    * a larger number of lanes, we need a hard reset */
    if (linkWidth > linkWidthCurrent)
    {
extern int VKI_PRINTEXC(char *fmt, ...);
        VKI_PRINTEXC("changing PCI-E link width from %d to %d\n", linkWidthCurrent,
linkWidth);
    }
}

```



```

        sysReboot();
    }

    #if 1 /* combined code */

    // Here...add code to disable link in PE_LCTL[4] = 1

        uiAddress = 0x8000013D + (linkWidth*0x100);
        uiData = 0x0;
        for (uiLoop = 0; uiLoop < (8-linkWidth); uiLoop++)
        {
            printf("EIBDATA %p, EIBADDR %p, uiAddress x%08x, uiData x%08x, uiLoop %d,
linkWidthCurrent %d, linkWidth %d\n",
                &(ATUE_REGS->EIBDATA), &(ATUE_REGS->EIBADDR), uiAddress, uiData,
                uiLoop, linkWidthCurrent, linkWidth);
            // put the data to write to the PCIE macro in the EIBDATA register:
            (volatile UINT32)ATUE_REGS->EIBDATA = uiData;
            // write to lane number "uiLoop" for the location 0x3D:
            (volatile UINT32)ATUE_REGS->EIBADDR = uiAddress;
            uiAddress += 0x100;
        }

    // Here...add code to enable link in PE_LCTL[4] = 0

    #define ATUE_RC_LINKTRAINING_TO 1000000

    // Here...Keep this section of code...instead of adding delay, just poll until
    PE_LSTS == 0

        /* wait for link training to occur: */
        for(uiTimer = 0;
            ((ATUE_REGS->PE_LSTS & ATUE_PE_LSTS_LNK_TRAINING) == 1) && (uiTimer <
            ATUE_RC_LINKTRAINING_TO);
            uiTimer++)
        {
            if (uiTimer == ATUE_RC_LINKTRAINING_TO)
            {
                printf("Link Training timeout! PE_LSTS = 0x%x\n", ATUE_REGS->PE_LSTS);
                return ERROR;
            }
        }
        printf("The PCIE link is trained. PE_LSTS = 0x%x uiTimer=%d\n",
            ATUE_REGS->PE_LSTS, uiTimer);
    #endif

        /* check to verify check was successful */
        if (bcmPcieLinkWidthGet() == linkWidth)
        {
            printf ("linkwidth set to %d\n", linkWidth);
            return OK;
        }

        printf ("Could not set PCI-E link: actual %d, expected %d\n",
            bcmPcieLinkWidthGet(), linkWidth);
        return ERROR;

    }

    PUBLIC int bcmPcieLinkWidthGet()
    {
        return ((UINT16)ATUE_REGS->PE_LSTS & ATUE_PE_LSTS_NEG_LNK_WIDTH_MASK) >>
            ATUE_PE_LSTS_NEG_LNK_WIDTH_SHIFT;
    }

    <EOR>

```

Status: [No Fix. See the Table , "Summary Table of Changes" on page 6.](#)



## 20. The PCI memory window is not retained following preemptive resets

**Problem:** The PCI memory window is not retained following preemptive resets.

PCI configuration space should be preserved during an Internal Bus (IB) Reset.

The IALR (limit register) and the IATVRO/IAUTVRO (translate value registers) are implemented in control register space (not in configuration space) so these get reset on an IB reset.

**Implication:** Because the limit register is AND'ed into the BAR, the resetting of the limit register causes the BAR to also get reset (indirectly).

This causes the BAR to go back to its default of 16 MB. This can cause overlapping BARs (if the BAR had been previously reduced to some size less than 16 MB) and potentially a system crash.

**Workaround:** Always use the default 16 MB limit register value.

**Status:** No Fix. See the [Table](#) , "Summary Table of Changes" on page 6.

## 21. Hot-Swap Next Item Pointer (HS\_NXTP) Register does not reset

**Problem:** The Hot-Swap Next Item Pointer register normally resets to 0 (to indicate that it is the final capability in the capabilities list). Instead, it resets to 0xE8.

The Developer's Manual is inconsistent. The "Default" column clearly states 0xE8, but the "Description" column clearly states 0x00. The RTL implements the "Default" column.

**Implication:** This only impacts applications that use the PCI-X bus in endpoint mode. When the HS\_NXTP resets to 0xE8 as the default, it points to the Hot-swap capability, thereby creating a possible infinite loop in enumeration S/W.

The other thing to point out that when cores are held in reset and Configuration Retry is disabled, then there is still exposure to a possible lock-up in the enumeration BIOS. The two scenarios impacted by this are FRU and Flashless boots.

**Workaround:** FW is already given the option to program the register to 0x90 (to link in the VPD capability). Therefore, we recommend that F/W always programs the register; either to 0x00 (no VPD) or to 0x90.

**Status:** No Fix. See the [Table](#) , "Summary Table of Changes" on page 6.



## 22. PBI read data gets corrupted when reading PBI MMRs while there is actual PBI read activity on the bus

**Problem:** PBI read data gets corrupted when reading PBI MMRs while there is PBI read activity on the bus and while there is long latency bus activity on the South XSI bus (XSI bus activity other than PBI reads and MMR reads). The XSI activity is the component of this issue that holds data in the PBI completion buffer making it susceptible to being overwritten by a subsequent PBI read while reading a PBI MMR. This occurrence is limited to reads of the following MMRs: PBCR, PBISR, PBBAR0, PBLR0, PBBAR1, PBLR1.

**Implication:** PBI read data can get corrupted.

**Workaround:** The occurrence of this issue is very unlikely, but the following precautions need to be observed:

- Read the MMRs listed above only during initialization and prior to any large transactions occurring on the South XSI bus.
- During normal operating mode, avoid reads of these MMRs. When a PBI MMR read is required, ensure that there are no PBI bus reads occurring.
- The only affected PBI register that a user normally considers reading after initialization is the PBI Status Register (PBISR). However, since there is only one interrupt source for the PBI bus, when the Interrupt Pending Register1 indicates a PBI Interrupt, it is redundant to read the PBISR to determine the cause of the interrupt.

**Status:** No Fix. See the Table , “Summary Table of Changes” on page 6.

## 23. There is an internal design issue with the upper 32-bits of the PCI/PCI-X bus on the 81341/81342

**Problem:** There is an internal design issue with the upper 32-bits of the PCI/PCI-X bus on the 81341/81342. The issue is isolated to only the upper 32-bits of the Address/Data lines.

**Implication:** The upper 32-bits of the PCI/PCI-X bus do not work reliably.

**Workaround:** Only use the lower 32-bits of the PCI/PCI-X bus.

**Status:** Fixed on C1. See the Table , “Summary Table of Changes” on page 6

## 24. PCI Express lane reversal does not occur when in PCI Express root complex mode

**Problem:** When an endpoint PCI Express device that does not implement optional PCI Express lane reversal is connected to 81341/81342 (when in root complex mode), with the PCI Express lanes reversed, 81341/81342 does not initiate lane reversal during the PCI Express link training process.

**Implication:** PCI Express link training fails to complete.

**Workaround:** When an endpoint device that does not have PCI Express lane reversal implemented needs to be connected to 81341/81342, lane reversal can not be used for the PCI Express interface. In this case board designers need to ensure the lanes of the root complex and downstream devices are wired correctly, such that lane numbers of the root complex device are matched with the same lane number on the downstream device. For example, a x4 PCI Express device must connect Lane 0 to 0, Lane 1 to 1, Lane 2 to 2 and Lane 3 to 3.

**Status:** No Fix. See the Table , “Summary Table of Changes” on page 6.



## 25. The SMBus reads are not supported

**Problem:** Writes corrupt SMBus read data which renders SMBus reads non-functional.

**Implication:** SMBus reads are not supported. No problem has been found with SMBus writes.

**Workaround:**

1. Use I<sup>2</sup>C Master to send outbound messages/responses (requires slave capability on the customer end of SMBus).

HW Changes to implement this workaround:

- I2C0 SCL -> SMBus SCL
- I2C0 SDA -> SMBus SDA
- I2C0 GND -> SMBus GND

On most boards there are three pins per each I<sup>2</sup>C: clock, data, and ground. The external I<sup>2</sup>C master connects to a ground, but this is just a common ground, so it is just two wires (clock and data) on the board and with no need to worry about the GND pin.

2. Use GPIOs to emulate an I<sup>2</sup>C Master (requires slave capability on the customer end of SMBus).

**Status:** No Fix. See the [Table](#) , “Summary Table of Changes” on page 6.

## 26. The SMBus unit (SMU) implements the Packet Error Check (PEC) incorrectly

**Problem:** The SMU slave is calculating the PEC incorrectly. The slave address is incorrectly being added to the start of the byte stream. For example, in a 4 byte block write sequence the PEC needs to be the CRC-8 of the 7 bytes from Address to D3 as follows:

PEC = CRC-8 (Address, CommandCode, ByteCount, D0, D1, D2, D3)

The SMBus unit is calculating the PEC as follows:

PEC = CRC-8 (SA, Address, CommandCode, ByteCount, D0, D1, D2, D3) where SA = the SlaveAddress programmed into the slave.

**Implication:** The PEC calculation will be incorrect.

**Workaround:** No workaround. PEC is an optional feature for SMBus and has been de-featured.

**Status:** No Fix. See the [Table](#) , “Summary Table of Changes” on page 6.



## 27. An ADMA transfer fails when using the "Chain Resume" command

**Problem:** An ADMA transfer fails when using the "Chain Resume" command.

The ADMA detects an End Of Chain (EOC) condition indicated by a NULL value in the ADMA Next Descriptor Address Register (ANDAR). Software then appends a new chain of descriptors by updating the ANDAR with the descriptor address of the new chain and setting the Chain Resume bit in the ADMA Channel Control Register - ACCR[1].

The fault situation occurs when setting of the Chain Resume bit is missed. When the missed Chain Resume command occurs, the "ADMA Active Flag" - ACSR[13] is clear, the ADMA Descriptor Address Register (ADAR) contains the value of the last completed transfer and the ANDAR is NULL, but none of the newly appended descriptors have been transferred.

**Implication:** There exists a window of opportunity where a "Chain Resume" issued to the ADMA is missed resulting in an appended descriptor chain not being executed, in other words, a missed ADMA transfer.

**Workaround:** When the last descriptor of every appended chain of descriptors has interrupts enabled, the fault condition always generates an EOC interrupt. When polling the ADMA with ADMA interrupts masked, the EOC status flag indicates the completion of a chain.

Software can detect the failed transfer by observing the following:

- When the EOC occurs indicating the ADMA is idle, i.e. the ADMA Active Flag is clear.
- The ADAR points to the previous expected "end of chain" descriptor, in other words, the last descriptor in the previous appended chain provided interrupts are enabled in that descriptor and/or the ADAR does not point to the currently expected "end of chain" descriptor.

When these conditions occur, the software can recover by reissuing "Chain Resume" and waiting for the next End of Transfer or EOC indicating the transfer has completed.

- Always clear EOC/EOT at the start of EOC/EOT processing.
- Everything prior to ADAR is always Processed/Returned as done.
- When the ADMA is inactive, then ADAR is Processed/Returned as done.
- Never read the ADMA Active Flag prior to reading ADAR (it gives a false Inactive, implying ADAR is done).
- To avoid vulnerability to lost Resume, when it appears to stop on anything other than EOC, issue a Resume.

**Pseudo Code:**

```
//Normal Flow (Does not include error handling)
Read ADAR;
Read ACSR [Active];
Process/Return all descriptors prior to ADAR;
If (not Active)
{
Process/Return ADAR;
If (ADAR /= EOC)
Resume
}
```

**Status:** **No Fix.** See the [Table](#) , "Summary Table of Changes" on page 6.



## 28. The MSI-X Capability ID is incorrect for IOP341 and IOP342

**Problem:** The MSI-X Capability ID is incorrect for 81341/81342; MSI-X Capability ID is 0x11 on the PCI Specification while 81341/81342 is 0x0D.

**Implication:** 81341/81342 do not support MSI-X.

**Workaround:** A change in firmware is needed to **\*hide\*** MSI-X capabilities from the OS.

**Status:** **No Fix.** See the [Table](#) , "Summary Table of Changes" on page 6.

## 29. Inbound MSI gets lost when core is simultaneously writing to IMIPRx to clear a previous interrupt

**Problem:** Inbound MSI gets lost when core is simultaneously writing to IMIPRx to clear a previous interrupt.

1. Intel XScale® microarchitecture writes to IMIPRx to clear the interrupt bit for the first MSI.
2. In the same clock cycle, the MU is trying to set a bit in the same IMIPRx for the second MSI.

When these two conditions occur in the same clock cycle to the same IMIPRx, the "set" operation for the second interrupt fails to set the bit.

**Implication:** Inbound MSI gets lost when core is simultaneously writing to IMIPRx to clear a previous interrupt.

**Workaround:** Use no more than one bit in each IMIPRx giving a total of only four possible inbound MSIs, in other words, one bit in each of IMIPR0, IMIPR1, IMIPR2, IMIPR3.

**Status:** **No Fix.** See the [Table](#) , "Summary Table of Changes" on page 6.



## Specification Changes

---

### 1. **PCIXCAP pin is now defeatured in Intel® 81341 and 81342 I/O Processors**

In the Intel® 81341 and 81342 I/O Processors, the PCIXCAP pin is now defeatured. Refer to Erratum 18 for more information.

Affected Docs: *Intel® 81341 and 81342 I/O Processors Developer's Manual*

*Intel® 81341 and 81342 I/O Processors Datasheet*

*Intel(R) 81341 and 81342 I/O Processors Design Guide*

### 2. **The Maximum Specification for the REFCLK Rise Time and Fall Time changed from 350ps to 700ps to match the PCI Express Specification**

Problem: The Maximum Specification for REFCLK Rise Time and Fall Time in "Table 21, PCI Express Clock Timing", in the 8134x Datasheet, has changed from 350ps to 700ps to match the PCI Express Specification.

Affected Docs: *Intel® 81341 and 81342 I/O Processors Datasheet*

### 3. **The VIH max level is changing from 1.8V to 2.0V for the p\_mode2 and p\_clkin pins**

Issue: The VIH max level is changing from 1.8V to 2.0V for the p\_mode2 and p\_clkin pins.

Affected Docs: *Intel® 81341 and 81342 I/O Processors Datasheet*



## Specification Clarifications

### 1. ATUe and ATU-X Outbound Memory Windows overlap at power-on

**Issue:** The Outbound Memory Windows (OUMBARs) for the ATUe and ATU-X are enabled and overlap by default when the 81341 and 81342 are powered on. This causes internal bus conflict when writing to the Outbound Window since each ATU tries to claim that address. The simplest fix is to only enable OUMBAR0 for ATU-X and OUMBAR1 for ATUe. The enable bit must be cleared in the other OUMBAR (Bit 31) to disable them.

**Affected Docs:** Intel® 81341 and 81342 I/O Processors Developer’s Manual

**Status:** No Fix. See the Table , “Summary Table of Changes” on page 6.

### 2. DLLRCVER values are not listed in the Developer’s Manual and are different for each stepping of the silicon

**Issue:** The DLLR values are listed in the Intel® 81341 and 81342 I/O Processors Developer’s Manual, but do not differentiate between DIMM and Embedded memory. Below is a summary of the recommended values for DIMM and Embedded memory designs that use ECC. Use the Auto-cal routine whenever possible (see section: 8.8.35.1.1). However, please note that the Auto-cal is not designed for non-ECC DIMM or non-ECC embedded topologies.

**DIMM Designs:** Set DLLR4[28:24] = 01101

DLLRCVER[10:8, 4:0] (Places trailing edge at center of post-amble)				
	400 MHz - 60 Ohms	400 MHz - 50 Ohms	533 MHz - 60 Ohms	533 MHz - 50 Ohms
2”	001,00101	001,00110	001,00010	001,00011
4”	001,01101	001,01101	001,01101	001,01101
6”	001,10101	001,10110	001,11000	001,11001
8”	001,11110	001,11110	010,00011	010,00100

**Embedded Designs:** Set DLLR4[28:24] = 01101

DLLRCVER[10:8, 4:0] (Places trailing edge at center of post-amble)				
	400 MHz - 60 Ohms	400 MHz - 50 Ohms	533 MHz - 60 Ohms	533 MHz - 50 Ohms
1”	001,00001	001,00010	000,11100	000,11101
1.5”	001,00011	001,00100	000,11111	001,00000
2”	001,00101	001,00110	001,00010	001,00011
4”	001,01101	001,01101	001,01101	001,01101
6”	001,10101	001,10110	001,11000	001,11001
8”	001,11110	001,11110	010,00011	010,00100

**Affected Docs:** Intel® 81341 and 81342 I/O Processors Developer’s Manual

**Status:** Doc. See the Table , “Summary Table of Changes” on page 6.



### 3. Adequate delay required when programming some Memory-Mapped Registers

**Issue:** Most peripheral units must be initialized/programmed by firmware using their respective memory-mapped registers (MMRs) before they can be used. Some examples of these registers are:

- Base Address Registers
- Limit Registers
- Size Registers

In addition, some of the peripheral units provide registers that are used as Address/Data Ports and for testing purposes. Firmware must ensure that any change made to these types of memory-mapped registers have successfully taken effect before further action is taken on that peripheral unit.

The registers which are written will not be updated immediately, so a subsequent transaction that relies on the register being updated may fail. When writing a value to an MMR that can change the operating mode or configuration of the system, the programmer must ensure that the register has been properly updated before subsequent transactions that are dependent on the written data begin execution. To do this, perform a read of the MMR after writing it and also create a data dependency on the read completing before subsequent instructions can execute.

One example where a register update must be in place before continuing, is the DLLRCVER register auto-calibration routine. In this example, the DLLRCVER register must be updated before a memory location can be read and then the memory operation must be complete before the DLLRCVER register can be read. Therefore, the DLLRCVER register is written, read-back, and then a register dependency is placed on the register for the read data, which means the read data must be returned before the read from DDR takes place. Then memory is read and a dependency is placed on the memory read to ensure data has returned before the DLLRCVER register is sampled.

```
#define MCU_DLLRCVER ((volatile unsigned int*)0xFFD82030)

/* Write DLLRCVER Register */
*MCU_DLLRCVER = data;

/* Read register back to push the value to the register */
val = *MCU_DLLRCVER;

/* Put a register dependency on the data being read back to make the core stall until
read data returns, which ensures the write has occurred*/
asm volatile("mov %0, %0" : : "r" (val));

/* Read a DDR SDRAM Memory Location, this memory read causes the auto calibration
circuit to sample the DQS signal */
val = *ddr_mem_addr;

/* Put a register dependency on the data being read back to ensure that the read from
memory has taken place */
asm volatile("mov %0, %0" : : "r" (val));

/* Read DLLRCVER register bit 24 to get DQS sample */
val = *MCU_DLLRCVER & (1<<24);

/* The comparison creates a dependency to ensure the data has returned from the
register */
return (val == 0 ? val : 1);
```

**Affected Docs:** Intel® 81341 and 81342 I/O Processors Developer's Manual.

**Status:** No Fix in B0. See the Table , "Summary Table of Changes" on page 6.



**4. During power-up, it is normal to see a current and voltage fluctuations on the 3.3 V, 1.8 V and 1.2 V power supplies**

Issue: The following may be observed during initial device power-up.

1. During initial power-up, it is normal to see a current drop on supplies that were powered up earlier as subsequent supplies power up. This is due to internal leakage paths being disabled.
2. When the 3.3 V supplies are powered up first it is normal to see the 1.2 V rail come part way up. This is due to internal paths between the two supplies that get isolated after the 1.2 V rail powers up.
3. During device power-on, it is normal to see a current drop on the 3.3 V supplies after the 1.2 V supplies powers up. The 3.3 V I/Os are disabled until all power rails are up and running. Once the other power rails are up and running the 3.3 V I/Os are enabled and this is the reason for the current drop.
4. Vcc1p8e voltage goes up to ~750 mV when the Vcc1p2e power rails go up. It stays at ~750 mV until it is powered up and then it goes to 1.8 V.

Affected Docs: *Intel® 81341 and 81342 I/O Processors Developer's Manual.*

Status: No Fix. See the [Table](#) , "Summary Table of Changes" on page 6.

**5. When the PCI Express bus is in Root Complex mode, Master Aborts do not occur on configuration reads to device numbers greater than 0**

Issue: Basically, a customer is expecting the ATUe to report a Master Abort when performing a configuration read to a device number other than 0.

A Master Abort does not occur when the PCI Express bus is in Root Complex mode and a configuration read occurs to device numbers greater than zero.

Only enumerate device 0. A Master Abort will not occur for any device# > 0.

The PCI Express specification states that Root Ports should Master Abort any configuration read to a device# other than 0: "Configuration Requests targeting the Bus Number associated with a Link specifying Device Number 0 are delivered to the device attached to the Link; Configuration Requests specifying all other Device Numbers (1-31) must be terminated by the Switch Downstream Port or the Root Port with an Unsupported Request Completion Status (equivalent to Master Abort in PCI)."

Legacy PCI enumeration software increments device# as it scans a bus with configuration reads. This causes issues when scanning the PCI Express bus because per the PCI Express specification section 7.3.1: "Devices must respond to all Type 0 Configuration Read Requests, regardless of the Device Number specified in the Request."

Affected Docs: *Intel(R) 81341 and 81342 I/O Processors Developer's Manual.*

Status: No Fix. See the [Table](#) , "Summary Table of Changes" on page 6.



**6. Specification changes for the C0 1200MHz 81341/81342 production units**

**Issue:** The C0 1200 MHz 81341 and 81342 production devices have the following specification changes for the 1.2 V voltage supply and the case temperature (Tc).

Impacted parts:

WP8 1342 M1220, SL9N6	MM#: 883345
WP8 1341 M1220, SL9N8	MM#: 883348

**1.2V Voltage Supply:** The 1.2 V voltage supply specification has changed:  
*From:* 1.2 V +/- 3%  
*To:* 1.25 V +/-2%

**Case Temperature (Tc):** The Tc specification has changed:  
*From:* 100°C  
*To:* 80°C

Affected Docs: *Intel(R) 81341 and 81342 I/O Processors Developer’s Manual*

**Status:** Fixed in C1. These two specification changes only impact the C0 1200 MHz 81341 and 81342 production units. These specifications revert back to the original 1.2 V voltage and case temperature specifications of 1.2 V +/-3% and 100°C, respectively. See the Table , “Summary Table of Changes” on page 6.

**7. The order in which the DMCU registers are programmed before running the 4GB DIMM init sequence is not clearly listed**

**Issue:** The order in which the DMCU registers is programmed before running the 4GB DIMM init sequence is not clearly listed.

When the Bank Size register is programmed for a 4GB DIMM that has 2 banks, each with a size of 2GB before the base address registers are programmed, a condition is entered into, where the DMCU claims all memory accesses across the entire 4GB range in the lower 32 bit address space - Address 0x0 to 0xFFFFFFFF.

**Implication:** This prevents the core and probably any other unit from accessing MMRs or the PBI windows since they are above 0xF0000000. This allows a problem for those that were to swap out a 1 or 2GB DIMM for a 4GB DIMM.

**Workaround:** The solution is to make sure that the Base Address registers are programmed first. As long as the upper base address register (SDUBR) is programmed to a value other than 0x0 before setting the Bank Size register, then things are okay since the DMCU does not attempt to claim memory addresses it should not.

**Status:** No Fix. See the Table , “Summary Table of Changes” on page 6.



## 8. Clarification on the initialization procedure for the circular queues

Issue: Do not let Host write to outbound post queue.

The circular queues must be initialized starting with the HEAD first and then the tail, or the queue is considered full and a write is continuously retried, causing a hang condition.

Figure 1. Required Initialization Order for Circular Queues

```
IN_FREE_HEAD_REG = IN_FREE_BOTTOM
IN_FREE_TAIL_REG = IN_FREE_BOTTOM
IN_POST_HEAD_REG = IN_POST_BOTTOM
IN_POST_TAIL_REG = IN_POST_BOTTOM
OUT_FREE_HEAD_REG = OUT_FREE_BOTTOM
OUT_FREE_TAIL_REG = OUT_FREE_BOTTOM
OUT_POST_HEAD_REG = OUT_POST_BOTTOM
OUT_POST_TAIL_REG = OUT_POST_BOTTOM
```

The MU is now a device on the Internal Bus (IB) and it can claim any transaction on the South IB, which hits its MUBAR/MUUBAR. This includes for example, using the ADMA for initial DDR memory scrub. Note that core access to memory does not conflict with the MU since the core comes in from the North IB and is isolated via the internal bridge. In situations where South IB access conflicts with the MU region, use the core to access that region of DDR.

Because the MU is now movable on the IB, the inbound translation value register, for the appropriate inbound BAR, is programmed to match the MU base, to assure inbound traffic translates as expected.

Status: Doc. See the Table , “Summary Table of Changes” on page 6.

## 9. CPLD bus keepers must be turned off during compilation

Issue: CPLD bus keepers must be turned off during compilation.

The bus keepers in the CPLD take the Flash bus width selection during power on to an indeterminate level (~1-1.2V) when the supply voltage is 3.6 V. By turning off the bus keepers for the CPLD this does not occur.

Status: Doc. See the Table , “Summary Table of Changes” on page 6.



## 10. PCI-Express link degradation when the 8134x IOC is connected to the Intel® 5400 MCH

**Problem:** When using a x8 (or x16) PCIe link to connect the 8134x IOC with the 5400 MCH Chipset (5400), link degradation has been observed.

**Implication:** Causes the interface to operate as a x1, x2, x4 or x8 link.

**Workaround:** The 5400 base platform has the ability to reinitiate the PCI-e link training. Utilizing this mechanism results in the Intel 8134x-based product retraining to a x8 rate.

When running in an 5400 base platform, the root component of this work around is as follows:

1. Analyze the link status rate by inspecting the PCI-e Link Status Register Use the *Intel® 5400 Chipset Memory Controller Hub (MCH) Datasheet* found at [www.intel.com/Assets/PDF/datasheet/318610.pdf](http://www.intel.com/Assets/PDF/datasheet/318610.pdf). Review the information on the Link Status Register on how to determine the negotiated link speed.
2. Issue the link retrain command (set and clear the Link Retrain bit).  
In the datasheet, search for the Bridge Control Register. Bit 6 is the Secondary Bus Reset. It needs to be set and then cleared to initiate the link retraining. Please note that this resets the ATU-e, as well as, the rest of the processor. The configuration information must be re-supplied to the 8134x IOP/IOC. The usual board reset does not reset the ATU-e.  
After the link retrain sequence completes, monitor the Link Training session status register to obtain the status of the retraining session (LNKTRG: Link Training in the datasheet). When it is 0 retraining is complete.
  - 0 – Indicates that the LTSSM (Link Training and Status State Machine) is neither in a "Configuration" nor a "Recovery" state.
  - 1 – Indicates that the Link training is in progress.
 This corresponds to PEXLNKSTS register (PCIe Link Status Register) which is a standard PCI-e register.

There are two ways to implement this work around (a possible complete solution is a combination of both). The first method uses the Option ROM (OROM) on the host bus adapter (HBA). Refer to the following steps:

1. The Option ROM loads into system memory and starts to execute.
2. The OROM confirms it is running in a 5400 environment. Refer to the 5400 datasheet for how this is done.
3. The OROM code determines in which slot the HBA is located. This is done by scanning the bus chain and comparing the content against what the OROM knows about its slot configuration
4. When the HBA is in a x8 slot but has not trained to a x8, save the PCI-e configuration information to re-supply the configuration information back to 8134x IOP/IOC. Issue the link retrain command sequence (set and clear the Link Retrain bit on the 5400 as described above.)
5. Look at the Link Training session status register (LNKTGR: Link Training)
  - 0 – Indicates that the LTSSM is neither in "Configuration" nor "Recovery" states.
  - 1 – Indicates that the Link training is in progress.
6. When the HBA still does not come up as a x8, repeat (say three times) then provide the configuration information to the 81348 and go on.



The second option is to implement this work around in the Host based driver for the 8134x IOP/IOC based product.

Use the driver-based workaround when the operational platform is required to support Sleep Mode. This is required since the OROM does not run when the system is coming out of Sleep Mode.

Status: **No Fix.** See the [Table , “Summary Table of Changes” on page 6](#). *Intel® External Design Specification* to be provided to customers working with this problem upon request.



## Documentation Changes

---

### 1. **PCIXCAP pin is now defeatured in Intel® 81341 and 81342 I/O Processors**

In the Intel® 81341 and 81342 I/O Processors, the PCIXCAP pin is now defeatured. Refer to Erratum 18 for more information.

Affected Docs: *Intel® 81341 and 81342 I/O Processors Developer's Manual*

*Intel® 81341 and 81342 I/O Processors Datasheet*

*Intel(R) 81341 and 81342 I/O Processors Design Guide*

Status: Doc. See the [Table](#) , "Summary Table of Changes" on page 6.

### 2. **The ATU-E chapter of the Developer's Manual, Section 3.3.3, Note #7 needs to be modified**

Problem: There is odd or unpredictable behavior in the use the outbound windows when using the core to do writes with the memory setup as bufferable. The issue is that too much host memory gets written (data corruption) with at least core writes of 4 bytes or less.

Workaround: Set the memory attribute of the outbound window to non-bufferable.

- When the target of the writes is DDR, bufferable or non-bufferable (note that this write does not cross the XSI bridge).
- When the DMA engine is used to do the small writes to the outbound window (local memory to IB - not only is the core not generating the write here, but the XSI bridge is taken out of the equation as well).
- When the core is used to target memory on the South IB bus (so we cross XSI) using bufferable or non-bufferable.

Affected Docs: *Intel® 81341 and 81342 I/O Processors Developer's Manual*.

Status: Doc. See the [Table](#) , "Summary Table of Changes" on page 6.

### 3. **The Introduction chapter of the 81341/342 Developer's Manual in the Application DMA Controller section states that ""The Intel XScale processor supports L2 cache hardware coherency". This is not correct**

Problem: The 81341/342 do not support cache coherency. The ADMA control bit for L2 cache coherency is not available.

Affected Docs: *Intel® 81341 and 81342 I/O Processors Developer's Manual*

Status: Doc. See the [Table](#) , "Summary Table of Changes" on page 6.



**4. In the 81341/342 Developer’s Manual, the Peripheral Registers chapter, the “Send Queue Not Full” interrupt generation is not documented correctly**

**Problem:** The Developer’s Manual states that the "Send Queue Not Full" interrupt is generated as follows:

$(\text{Put-Get}) \neq \text{Size}$ . This only works when the size is set to the max value 0xFFFF.

Ex1: Size = 0xFFFF

Put = 0x0004

Get = 0x0005

Put-Get = -1 == 0xFFFF == Size (WORKS)

Ex2: Size = 0x00FF

Put = 0x0004

Get = 0x0005

Put-Get = -1 == 0xFFFF != Size (Does NOT Work)

To handle the rollover case, the real equation should have be:

$((\text{Put-Get}) \& \text{Size}) \neq \text{Size}$

**Workaround:** Either:

1. Always use 64K entries, Size = 0xFFFF, or
2. When less than 64K entries desired, then:
  - a. When setting the Put pointer to Actual-Size, set the Size to Actual-Size. This works because in this case the Get will not have rolled over.
  - b. When setting the Put pointer to anything else, set the Size to -1. This works because the Get pointer has rolled over.

**Affected Docs:** Intel® 81341 and 81342 I/O Processors Developer’s Manual

**Status:** Doc. See the Table , “Summary Table of Changes” on page 6.

**5. In the 81341/342 Developer’s Manual, the ATU chapter, Embedded Bridge Functionality information does not support 4GB inbound windows**

**Affected Docs:** Intel® 81341 and 81342 I/O Processors Developer’s Manual

**Status:** Doc. See the Table , “Summary Table of Changes” on page 6.



## 6. The “Note” in the section on supported Flash is wrong

**Problem:** The “Note” in the section on supported Flash in the “Design Notes” section of the 81341/81342 Design Review Checklist is wrong.

The “Note” in the section on supported Flash in the “Design Notes” section of the 81341/81342 Design Review Checklist should read:

“Note: Flash device that support **CFI Cmd set 1 (CFI1).**”

**Affected Docs:** Intel® 81341 and 81342 I/O Processors Design Review Checklist

**Status:** Doc. See the [Table](#) , “Summary Table of Changes” on page 6.

## 7. The TVB6 and TBA6 parameters are not listed in the 81348 Datasheet

**Problem:** The TVB6 and TBA6 parameters are not listed in the 8134x Datasheet. These two parameters need to be listed in Table 23 under the TVB5/TVA5 parameters.

TVB6	DQ, CB read input valid time before DQS rising or falling edges.		0.700	ns	
TVA6	DQ, CB read input valid time after DQS rising or falling edges.		0.700	ns	

**Note:** See Figure 16, DDR2 SDRAM Read Timing on Page 85.

**Affected Docs:** Intel® 81341 and 81342 I/O Processors Datasheet

**Status:** Doc. See the [Table](#) , “Summary Table of Changes” on page 6.

**8. The VCCPLL Pin Requirements section of the Datasheet have changed**

**Problem:** The VCCPLL Pin Requirements section of the Datasheet have changed. See below for the updates in Section 4.1 and Figure 8 in the Datasheet.

**Implication:**

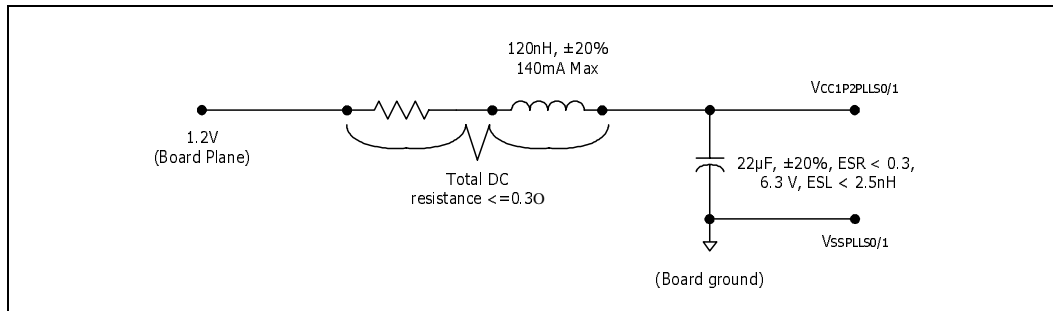
**New wording for Section 4.1 - VCCPLL Pin Requirements (last two paragraphs):**

This paragraph pertains to the VCC1P2PLLS0, VCC1P2PLLS1 filters. The recommended filter for the PLL supplies is shown in Figure 8. The purpose of this filter is to achieve at least 10 dB rejection of frequencies between 1 and 20 MHz. The board supply distribution system must ensure that the minimum voltage into the PLL ball is equal to or greater than 1.164 V. The current draw for the IC is 95 mA typical less than 140 mA maximum. The filter components are selected to achieve a corner frequency of 100 KHz. The series resistance keeps the Q of this resonant circuit safely below unity for all component variations. The total DC resistance of the resistor and the inductor must be equal to or less than 0.3 Ohms.

The bypass capacitor must be placed as close to the supply pins as possible. The series impedances to both the supply pin and the PCB analog ground plane must be an order of magnitude lower than the ESR and ESL specified for the capacitor. The S0/S1 PLLs have dedicated internal supplies, so the VSSPLLS0/S1 pins must be soldered directly to the analog ground plane of the PCB.

**New Figure 8:**

**Figure 8. VCC1P2PLLS0, VCC1P2PLLS1 Low-Pass Filter**



**Affected Docs:** Intel® 81341 and 81342 I/O Processors Datasheet

**Status:** Doc. See the Table , "Summary Table of Changes" on page 6.