Document Number: IMX50CE
Rev. 5, 05/2015

Chip Errata for the i.MX50

This document details the silicon errata known at the time of publication for the i.MX50 multimedia applications processors revision 1.1.1.

Table 1 provides a revision history for this document.

Table 1. Document Revision History

Rev. Number	Date	Substantive Changes
Rev. 5	05/2015	• Added WDOG erratum ERR004346.
Rev. 4	03/2014	• Update to DDR PHY erratum ERR007894.
Rev. 3	03/2014	• Added DDR PHY erratum ERR007894. • Added USB erratum ERR006308. •
Rev. 2	03/2012	• Added EIM erratum ENGcm12379. • Added eSDHC erratum ENGcm03648. • Updated eSDHC erratum ENGR143117.
Rev. 1	07/2011	Added DPLL erratum ENGcm12051.
Rev. 0	07/2011	Initial release.

freescale™ semiconductor

Table 2 summarizes errata on the i.MX50, categorized by module.

Table 2. Summary of Silicon Errata

Errata	Name	Solution	Page
AIPS			
ENGcm07186	AIPS unalign causes abort on writes on internal register	No fix scheduled	5
ARM Core			
ENGcm02157	ARM #468413: Incorrect L2 cache eviction can occur when L2 configured as inner cache	No fix scheduled	6
ENGcm02161	ARM #468414: NEON load data can be incorrectly forwarded to a subsequent request	No fix scheduled	7
ENGcm02163	ARM #468416: Processor deadlock can occur L2 cache servicing write allocate memory	No fix scheduled	9
ENGcm02164	ARM #468415: Instruction: swap, preload, fetch can interact and cause deadlock	No fix scheduled	11
ENGcm03161	ARM #485963: C15 Cache Selection Register (CSSELR) is not banked	No fix scheduled	13
ENGcm03162	ARM #488063: ARPROT[0] incorrectly indicates USER transaction for tablewalks	No fix scheduled	14
ENGcm07666	ARM #586320: Clean & Clean/Invalidate maintenance ops by MVA&PoC may not push data to external memory	No fix scheduled	15
ENGcm07667	ARM #586323: Cache clean memory ops by Preload or Clean MVA to PoC may corrupt the memory	No fix scheduled	17
ENGcm07668	ARM #586324: Cache maintenance operation done by MVA may corrupt memory	No fix scheduled	19
ENGcm07669	ARM #588115: A RAW hazard on certain CP15 registers can result in a stale register	No fix scheduled	21
ENGcm09828	ARM #709718: Load and store operations to shared device memory regions may not complete in program order	No fix scheduled	23
ENGcm10693	ARM #628216: Potential OVFL status loss when it occurs at CP15 and CP14 update	No fix scheduled	25
ENGcm10702	ARM #687067: BTB invalidate by MVA operations will not work when the IBE bit is set	No fix scheduled	27
ENGcm10704	ARM #693270: Taking a watchpoint is incorrectly prioritized over a precise data abort	No fix scheduled	29
ENGcm10713	ARM #507113: A Neon store to device memory can result in dropping a previous store	No fix scheduled	31
ENGcm10723	ARM #715847: VCVT.f32.u32 can return wrong result in a specific configuration of Floating Point Unit (FPU)	No fix scheduled	33
ENGcm11134	ARM #468416: Under a specific set of conditions, processor deadlock can occur when L2 cache is servicing write allocate memory	No fix scheduled	35
ENGcm11231	ARM #728018: MVA Cache operations for non-cacheable memory region cause deadlock	No fix scheduled	37
CSPI			
ENGcm08174	CSPI wrongly clears the overrun status bit	No fix scheduled	39
ENGcm11544	Issue in 32-bit Rx when CSPI is configured in slave mode with SSCTL bit set	No fix scheduled	40

Table 2. Summary of Silicon Errata (continued)

Errata	Name	Solution	Page
DCP			
ENGR120443	DCP fails to handle SHA1/SHA256 hashing for inputs that are split across multiple pages	No fix scheduled	41
DDR PHY			
ERR007894	Under certain conditions, the DDR DQS_Gate may close earlier at DDR2 mode, causing failure on DDR read.	No fix scheduled	42
DPLL			
ENGcm12051	DPLL: Meta-stability Issue	No fix scheduled	43
DRAM MC			
ENGR121624	APBHDMA channel may stall on waiting for PIO grant when freeze bit is set/clr frequently	No fix scheduled	45
ENGR125340	LPDDR2 ZQ calibrator state machine cannot meet 266 MHz design target		46
eCSPI			
ENGcm10185	eCSPI burst completion by SSB signal in slave mode not functional	No fix scheduled	47
ENGcm10188	eCSPI should send zeros when the burst is longer than the amount of data in FIFO	No fix scheduled	48
EIM			
ENGcm12379	EIM: AUS mode is non functional for devices larger than 32 MB.	No fix scheduled	49
EPDC			
ENGR138633	Histogram Fractional Macro Block Processing Issue	No fix scheduled	50
ENGR142735	TCE Underrun in EPDC	No fix scheduled	51
EPIT			
TLSbo90606	Glitch occurs on SCLK in EPIT when switching clock source	No fix scheduled	52
eSDHC			
ENGcm03648	eSDHC AutoCMD12 and R1b polling problem	No fix scheduled	53
ENGcm11088	eSDHC partial block read problem when block size is not a multiple of 4 (applicable to both eSDHCv2 and eSDHCv3)	No fix scheduled	54
ENGcm11113	eSDHC problem when ADMA2 last descriptor is LINK or NOP (applicable to both eSDHCv2 and eSDHCv3)	No fix scheduled	55
ENGR143117	eSDHCv3 on eSDHCv3 port has setup timing issue in SD SDR mode	No fix scheduled	56
GPT			
ENGcm04874	Glitch on SCLK in GPT while switching the clock source	No fix scheduled	57
OCRAM			

Table 2. Summary of Silicon Errata (continued)

Errata	Name	Solution	Page
ENGR139532	OCRAM Setup Timing Issue	No fix scheduled	58
ROM			
ENGR133711	ROM fails to boot from boot partition 1 or boot partition 2 in 4-bit and 8-bit DDR non-fast boot modes	No fix scheduled	59
USB			
ENGcm07150	Erroneous descriptor handling by USBOH module	No fix scheduled	60
ERR006308	USB: HOST controller lock-up issue	No fix scheduled	61
WDOG			
ERR004346	WDOG: SRS bit requires to be written twice	No fix scheduled	62

ENGcm07186 AIPS unalign causes abort on writes on internal register

Description:

Unaligned access to AIPS can be driven high by debug access port (DAP) and Fast Ethernet controller (FEC). If they access the AIPS internal registers during an unaligned access, an ABORT occurs.

Currently, if unalign is asserted, the AIPS will generate an abort on accesses to internal registers, as the v6 extensions are not supported.

Projected Impact:

Unaligned access to AIPS internal registers fails.

Workaround:

Make only aligned access to the AIPS internal registers.

Silicon Fix:

No fix scheduled.

ENGcm02157 ARM #468413: Incorrect L2 cache eviction can occur when L2 configured as inner cache

Description:

Under specific set of conditions, the stale data saved in the L2 cache can be erroneously returned to the processor on a subsequent load instruction.

The conditions are as follows:

- The L2 cache must be configured as an inner cache rather than as an outer cache
- The L2 cache must be configured to use write allocate memory type

The issue is reported by ARM, erratum ID 468413, Category 2[1].

Projected Impact:

If this erratum occurs, stale data can be read by a subsequent load instruction, resulting in an incorrect program behavior.

Workarounds:

There are two viable workarounds for this erratum. One workaround is, not to configure the L2 cache as an inner cache, but maintain the default setting as an outer cache. The second workaround is to use the remap registers to remap the inner cache attributes from write allocate to write back instead.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Workaround implemented in Linux BSP.

The software workaround is to disable write allocate in the Level 2 cache in the bootloader. So no condition is to trigger this issue. This workaround has performance penalty.

WinCE BSP Status:

Workaround implemented in WinCE BSP.

Write allocate is disabled in L2 cache control register.

1. Category 2 defined as: Behavior that contravenes the specified behavior and that can limit or severely impair the intended use of specified features, but does not render the product unusable in all or the majority of applications.

ENGcm02161 ARM #468414: NEON load data can be incorrectly forwarded to a subsequent request

Description:

Under very specific set of conditions, data from a Neon load request can be incorrectly forwarded to a subsequent, unrelated memory request.

The conditions are as follows:

- Neon loads and stores must be in use
- Neon L1 caching must be disabled
- Trustzone must be configured and in use
- The secure memory address space and the non-secure memory address space both use the same physical addresses, either as an alias or the same memory location or for separate memory locations

The issue is reported by ARM, erratum ID 468414, Category 2[1].

Projected Impact:

If this erratum is encountered, it is possible for a load request to receive the wrong data value which can likely result in incorrect operation of the program.

Workarounds:

There are many software solutions for this erratum and only one has to be applied. The recommended solution, if possible, is to map cacheable areas of memory so that both secure and non-secure do not share the same physical address space.

Another possible solution is to force NEON to cache in the L1 data cache. This can be programmed using the Auxiliary Control Register bit [5], L1NEON, as follows:

```
MRC p15, 0, r0, c1, c0,             1; read register
ORR r0, r0, #(1<<5)                 ; L1NEON caching enable
MCR p15, 0, r0, c1, c0, 1           ; write register.
```

Another possible solution is to disable L2 data forwarding from the victim buffers. This can be programmed using the L2 Auxiliary Control Register bit[27], Load data forwarding disable as follows:

```
MRC p15, 1, r0, c9, c0, 2           ; read register
ORR r0, r0, #(1<<27)                ; L2 load data forwarding disable
MCR p15, 1, r0, c9, c0, 2           ; write register
```

Both workarounds can be implemented with little or no perceived performance impact in the majority of applications.

Proposed Solution:

No fix scheduled.

1. Category 2 defined as: Behavior that contravenes the specified behavior and that can limit or severely impair the intended use of specified features, but does not render the product unusable in all or the majority of applications.

Linux BSP Status:

Trustzone is not used and Neon L1 caching is enabled. So, this case does not occur.

WinCE BSP Status:

Trustzone is not used in WinCE BSP. Therefore, the erratum does not apply.

The issue occurs when two physical addresses in both Secure (S) and Non-Secure (NS) worlds are required. If the use of the NS world is never enabled, then the issue cannot occur. The hazard occurs between the NS and S worlds which does not get flushed properly, and data gets forwarded from the L2 data buffers.

ENGcm02163 ARM #468416: Processor deadlock can occur L2 cache servicing write allocate memory

Description:

If a load request is processed which misses the L2 cache, but cannot be immediately forwarded to the BIU, it encounters a special hazard which prevents the request from being required to access the L2 cache RAM again to save power. There can be multiple requestors with unique addresses, (that is, one address per cache line) with this special hazard. All write-allocate requests that access the L2 cache RAM, on port1, do not have address comparators to check for this special hazard condition. So, if a subsequent write-allocate request is issued to the L2 cache RAM on port1 and allocates a victim buffer, then all requests pending with this special hazard must be forced to perform a L2 cache RAM lookup again to maintain memory coherency. There is a 1-cycle window in which the write-allocate request must allocate to a victim buffer and a pending request to the BIU is not prohibited from going to the BIU, such that a deadlock can occur.

The conditions are as follows:

- The processor must have L2 cache present and enabled.
- The L2 cache must be configured to support the write allocate memory type.

The issue is reported by ARM, erratum ID 468416, Category 2.[1]

Projected Impact:

If this erratum is encountered and processor deadlock occurs, it can only be interrupted by asserting RESET on the processor.

Workarounds:

The workaround for this erratum is to disable write-allocate by programming the L2 Auxiliary Control Register bit[22], Write allocate disable:

```
MRC p15, 1, r0, c9, c0, 2; read register
ORR r0, r0, #(1&lt;&lt;22); Write allocate disable
MCR p15, 1, r0, c9, c0, 2; write register
```

Disabling write allocate in the L2 cache could have a performance impact for some applications.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Workaround implemented in Linux BSP.

The software workaround is to disable write allocate in the Level 2 cache. This workaround has performance penalty.

1. Category 2 defined as: Behavior that contravenes the specified behavior and that can limit or severely impair the intended use of specified features, but does not render the product unusable in all or the majority of applications.

WinCE BSP Status:

Workaround implemented in WinCE BSP.

Write allocate is disabled in L2 cache control register.

ENGcm02164 ARM #468415: Instruction: swap, preload, fetch can interact and cause deadlock

Description:

Three memory requests in the L2 cache can interact and result in a deadlock condition. The exact scenario involves a dependency chain of three requests, an instruction fetch request, a memory preload instruction (PLD), and a swap instruction (SWP). In this dependency loop, no request can progress as each one of them is dependent on the next request. That is, the PLD request cannot complete as the IF request is pending to use the BIU. The IF request cannot complete because of the pending SWP request, and the SWP request is not allowed to complete as it is waiting on the PLD to complete before obtaining the lock on the bus.

The conditions are as follows:

- PLD instructions must be used by the processor
- SWP instructions must be used by the processor

The issue is reported by ARM, erratum ID 468415, Category 3[1].

Projected Impact:

This erratum only impacts the users of swap instructions. Swap instructions have been deprecated from the ARMv7 version of the ARM Architecture as its functional use in terms of setting up semaphores is now replaced from the ARMv6 architecture forwards by the LDREX and STREX instructions. If this erratum is encountered and the processor deadlock occurs, it can only be interrupted by resetting the processor.

Workarounds:

One software workaround for this erratum is, not to use the swap instructions. If swap instructions are to be used in the code base, the other software workaround is to disable the PLD instructions and make them a NOP. The code required to implement this workaround is as follows:

```
MRC p15, 0, r0, c1, c0, 1; read register
ORR r0, r0, #(1<<9); PLDNOP - force PLD to be NOP
MCR p15, 0, r0, c1, c0, 1; write register
```

This workaround has some performance impact on the peak memory copy bandwidth.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Not implemented because the swap instructions are not used. They are deprecated for ARMv7.

1. Category 3 defined as: Behavior that is not the originally intended behavior but should not cause any problems in applications.

WinCE BSP Status:

Kernel Interlocked APIs do not make use of swap instructions. No usage of swap instruction found in the public/private code that Freescale has access to. It is confirmed that swap instruction is not used in the WinCE OS.

Freescale codecs and customer application code must avoid usage of swap instructions.

ENGcm03161 ARM #485963: C15 Cache Selection Register (CSSELR) is not banked

Description:

ARMv7 architecture specifies that the CSSELR should be banked between Secure and Non-secure states. Cortex-A8 does not currently bank this register.

The conditions are as follows:

- The system should have an active process in secure state and an active process in non-secure state at the same time.
- The system should perform cache maintenance operations in both secure and non-secure processes.

The issue is reported by ARM, erratum ID 485963, Category 2[1].

Projected Impact:

A cache cleaning sequence that reads the CSSELR may not work as expected. The published sequence for cleaning the entire cache (see *ARM Architecture Reference Manual*) includes setting the CSSELR followed by a read from the selected Cache Size ID register (CCSIDR). If the non-secure side executes this sequence, and is encountered by a secure interrupt between the setting of the CSSELR and the reading of the selected CCSIDR, then there is a possibility that the secure side may also use the CSSELR. On returning to the non-secure side, the CSSELR value may have changed, which makes the cache cleaning sequence to malfunction. Similarly, a non-secure interrupt can cause a secure cache cleaning sequence to malfunction.

Workarounds:

When transitioning security state, the secure monitor software should save the current CSSELR value (corresponding to the security state the processor is transitioning out of) and restore the previously saved CSSELR value (corresponding to the security state the processor is transitioning into).

Proposed Solution:

No fix scheduled.

Linux BSP Status:

No software workaround required. Trustzone is not used.

WinCE BSP Status:

No software workaround required. System does not perform cache maintenance in non-secure mode.

1. Category 2 defined as: Behavior that contravenes the specified behavior and that can limit or severely impair the intended use of specified features, but does not render the product unusable in all or the majority of applications.

ENGcm03162 ARM #488063: ARPROT[0] incorrectly indicates USER transaction for tablewalks

Description:

All memory transactions performed as part of a tablewalk should be considered Privileged, even in the User mode. However, Cortex-A8 incorrectly marks memory transactions generated from tablewalks performed in User mode as the user transactions on the AXI bus. This indication is given by the ARPROT[0] signal, which is set to zero during the transaction.

The conditions are as follows:

- Cortex-A8 must be in User mode
- A memory transaction (instruction or data) misses in the TLB and causes a tablewalk
- The address for the page table entry is not found in the L2 cache, resulting in an external memory request
- This erratum occurs when ARPROT[0] incorrectly indicates a user transaction for this memory request on the AXI bus.

The issue is reported by ARM, erratum ID 488063.

Projected Impact:

As the values broadcast on ARPROT[0] are completely transparent to the software, the implications for this erratum are only on a specific subset of the processor systems, specifically for a system that includes some form of system level memory protection unit, that uses the ARPROT bits to determine if a memory request can be allowed. For any system that does include such a unit, that unit may report false errors on page table accesses due to this erratum.

Workarounds:

As the processor directly does not make use of ARPROT[0], any workaround would be specific to the device that makes use of the values broadcast on ARPROT[0]. The most likely usage would be some form of system memory protection unit. If such protection unit exists, it may need to filter out any access to the page tables from the address space that is protected to operate properly. This implies that the external protection unit cannot provide additional protection for the page tables. For example, the page table cannot be inserted in a Secure RAM which cannot be accessed in User mode, as in this case, an additional protection is added beside the MMU. Alternatively, the CSU can be configured to transform User access to Privileged on addresses used by PAGE TABLE.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

System memory bus has no user mode protection, so this is not applicable.

WinCE BSP Status:

No software workaround is available.

ENGcm07666 ARM #586320: Clean & Clean/Invalidate maintenance ops by MVA&PoC may not push data to external memory

Description:

When a Clean to Point of Coherency or Clean and Invalidate to Point of Coherency by MVA operation is performed, it is possible that the line remains present in the L2 cache and any dirty data is not pushed out on to the AXI bus to main memory. This can occur whenever the requested address is present in the L1 cache but not the L2 cache.

The conditions are as follows:

- The memory region being cleaned is configured in write allocate mode
- The cache line being cleaned is initially present in the L1 cache and not in the L2 cache

The issue is reported by ARM, erratum ID 586320, Category 2[1].

Projected Impact:

If a Clean or Clean and Invalidate operation does not operate as intended, and leaves the data present in the L2 cache, the memory coherency in the system can no longer be guaranteed. Therefore, this erratum impacts any code sequence used to maintain the system coherence.

Workarounds:

The software workaround for this erratum is to disable the write allocate in the L2 cache, as shown in the following instruction sequence:

```
MRC p15, 1, <Rd>, c9, c0, 2; read L2 cache Aux Ctrl Reg
ORR <Rd>, <Rd>, #(1 << 22); set the Write Allocate disable bit
MCR p15, 1, <Rd>, c9, c0, 2; write the L2 cache Aux Ctrl Reg
```

Disabling the write allocate in the L2 cache can impact the performance of some applications. If this performance impact is deemed to be very high, there are two other software workarounds that can be used. The first is to disable write allocate around each sequence of clean by MVA to PoC or clean/invalidate by MVA to PoC instructions, as shown in the following instruction sequence:

```
MRC p15, 1, <Rd>, c9, c0, 2; read L2 cache Aux Ctrl Reg
ORR <Rd>, <Rd>, #(1 << 22); set the Write Allocate disable bit
MCR p15, 1, <Rd>, c9, c0, 2; write the L2 cache Aux Ctrl Reg

<perform sequence of MVA operations here>

MRC p15, 1, <Rd>, c9, c0, 2; read L2 cache Aux Ctrl Reg
BIC <Rd>, <Rd>, #(1 << 22); clear the Write Allocate disable bit
MCR p15, 1, <Rd>, c9, c0, 2; write the L2 cache Aux Ctrl Reg
```

The final workaround that can be implemented is to perform each maintenance operation twice with interrupts disabled. By performing the operation twice in back-to-back successions with no other memory operations executed in between, it can be assured that the line is evicted from both L1 and L2 cache and written out to main memory.

1. Category 2 defined as: Behavior that contravenes the specified behavior and that can limit or severely impair the intended use of specified features, but does not render the product unusable in all or the majority of applications.

Perform the following steps:

1. Disable the interrupts and the imprecise aborts
2. Execute the maintenance operation first pass
3. Execute the same maintenance operation, second pass
4. Enable the interrupts and the imprecise aborts

Repeat the above sequence for each cache maintenance operation. Interrupts can remain disabled for a longer sequence of maintenance operations, but this has a negative effect on interrupt latency. This workaround has a performance impact on the execution time of cache maintenance operations.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Workaround implemented in Linux BSP.

The software workaround is to disable write allocate in the Level 2 cache in the bootloader. This workaround has performance penalty.

WinCE BSP Status:

Workaround implemented in WinCE BSP.

Write allocate is disabled in L2 cache control register. This workaround impacts performance.

ENGcm07667 ARM #586323: Cache clean memory ops by Preload or Clean MVA to PoC may corrupt the memory

Description:

When a Clean to Point of Coherency (PoC) by MVA operation is performed, or the Preload Engine is programmed to clean a region of memory from the L2 cache, a cache line from that region can be corrupted with a stale copy of memory, and a memory store operation is lost.

The conditions are as follows:

- A Cache Clean by MVA to the PoC instruction is executed to clean cache line A, or the preload engine is configured to clean a memory region which includes cache line A. Either of the operations result in the placement of cache line A into a victim buffer for writeback to external memory. It also keeps the line still valid in the L2 cache.
- A memory store operation is performed to the same cache line A that is evicted by the cache clean operation. This operation results in a modification of cache line A in the L2 cache (but not to the copy of the line that may still remain in the victim buffer if memory response is slow).
- A cache eviction is done of cache line A due to an unrelated memory request to load cache line B. The modified copy of cache line A is placed in a victim buffer. At this point, the two victim buffers may contain two different versions of cache line A. As each victim buffer uses a different AXI ID and arbitrates independently for the AXI bus, there is no guarantee for the order in which the memory updates occur, and the store operation may be overwritten by the cache clean operation, leaving the external memory with stale contents.

The issue is reported by ARM, erratum ID 586323, Category 2[1].

Projected Impact:

If the operation sequence occurs as described above, one or more store operations are lost, resulting in incorrect program behavior. This can occur for any application which either uses the preload engine to clean a memory region, or uses Clean by MVA to PoC maintenance operations to clean a region of memory.

Workarounds:

There are two feasible workarounds that can be used for this erratum. The first workaround is to place a DMB or DSB barrier at the end of each cache clean routine or preload engine memory clean sequence. This barrier operation ensures that the cleaned line goes out and is seen by main memory before the store is executed and therefore guarantees that the clean is done correctly and memory contains the correct final value.

This workaround is consistent with the ARM recommended practice for ending the maintenance routine. The above workaround is convenient to implement and should work for all expected usage models. However, there is still the possibility that an interrupt can be taken during the clean routine, and the interrupt handler can perform a store operation to the line just cleaned, allowing for the scenario which can lead to the erratum.

1. Category 2 defined as: Behavior that contravenes the specified behavior and that can limit or severely impair the intended use of specified features, but does not render the product unusable in all or the majority of applications.

Another workaround that avoids even the case mentioned above, is to convert all Clean by MVA to PoC operations to Clean and Invalidate by MVA to PoC as described in the code sequence as follows:

- Replace all uses of: MCR p15, 0, <Rn>, c7, c10, 1;
- Clean Data cache line by MVA to PoC with this instruction: MCR p15, 0, <Rn>, c7, c14, 1;
- Clean and Invalidate cache line by MVA to PoC.

There is no Preload Engine equivalent for the second workaround option as it is not possible to configure the preload engine to perform a clean and invalidate operation. Therefore, if there are concerns that the DSB based workaround is insufficient, then it is advisable to not use the Preload Engine for cleaning memory regions. The preload engine can be configured such that it is not accessible at user/privilege and nonsecure/secure level of granularity.

For more information on Preload Engine configurability, see *Cortex-A8 Technical Reference Manual*.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Workaround implemented in Linux BSP.

All cache flush routines have DSB at end.

WinCE BSP Status:

Workaround implemented in WinCE BSP.

Interrupts are enabled during cache clean operation, so DMB/DSB workaround is not sufficient.WinCE BSP implements software workaround to replace clean cache line operation with Clean-and-Invalidate cache line operation. There may be a performance penalty due to the undesired invalidation of the cache line when invoking OEMCacheRangeFlush to clean individual cache lines.

ENGcm07668 ARM #586324: Cache maintenance operation done by MVA may corrupt memory

Description:

If a non-cacheable memory request is subsequently followed by any cache maintenance operation done by MVA, then the memory can be corrupted.

The conditions are as follows:

- The L1 data cache must be of size 32 Kbyte
- The L1 data cache hardware alias checks are enabled (the L1ALIAS bit in the Auxiliary Control Register is set to 0)
- The virtual memory management used by the operating system does not follow the page coloring guidelines and allows virtual to physical address alias cases to exist on bit 12 of the address
- A non-cacheable memory request to normal, device, or strongly ordered memory is subsequently followed by a cache maintenance operation done by MVA without any cacheable memory operations executed in between. The non-cacheable memory request can be fully executed, or can be a speculative instruction in the branch shadow that subsequently is flushed.

When the above conditions are met and the cache maintenance operation is performed to generate a hash alias scenario on its cache lookup, memory corruption or a false parity error can occur.

The issue is reported by ARM, erratum ID 586324, Category 2[1].

Projected Impact:

If the operation sequence occurs as described above, then memory can be corrupted or a false parity error can be generated. In addition, even if the workaround as described below is implemented, it is possible that a nonsecure maintenance operation could result in the invalidation of a secure memory location. Therefore, this could possibly be viewed as an avenue for a security attack. However, the contents of secure memory cannot be viewed as a direct result of this erratum and the lack of consistent repeatability makes it very difficult for the user to make use of this erratum as a security attack.

Workarounds:

If full PIPT caching support is not required by the operating system, or the processor includes a 16 Kbyte L1 data cache, then no workaround is required. If alias conditions can occur, then the workaround is to guarantee that a cache maintenance operation is not immediately preceded by a non-cacheable memory request. This is guaranteed by initiating every cache maintenance by MVA routine with a cacheable load or store request immediately preceding the main loop and ending with a DSB barrier operation at the end of the loop. The load or store that precedes the loop can be done to any cacheable memory location. In addition, both interrupts and aborts should be masked during the cache maintenance routine. Interrupt masking is required to prevent a non-cacheable memory request, either fully executed or in a branch shadow, from initiating the sequence that can

1. Category 2 defined as: Behavior that contravenes the specified behavior and that can limit or severely impair the intended use of specified features, but does not render the product unusable in all or the majority of applications.

result in this erratum. If there are concerns about the interrupt latency, the maintenance loop can be amended to enable and disable the interrupts directly around the maintenance operation. This impacts the time taken to complete the maintenance loop.

To workaround any concerns of a potential security attack due to this erratum, all secure memory should be marked as inner write through. This can be done either by using the caching attributes in the page tables for all secure page tables or by making use of the secure banked version of the remap registers. Apart from making all secure memory write through, a routine should be run out of reset to completely fill the cache with dummy data, to prevent invalid, uninitialized data in the cache from being written out to memory and potentially corrupting secure memory. Making all secure memory inner write through guarantees that even if the invalidation of a secure line in the L1 cache occurs due to this erratum, the correct data is not lost.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Does not apply to Linux BSP because page coloring guidelines are followed for VIPT cache types.

WinCE BSP Status:

It was determined that the conditions required by the erratum do not occur within WinCE, and therefore, no software workaround is required.

ENGcm07669 ARM #588115: A RAW hazard on certain CP15 registers can result in a stale register

Description:

Under certain conditions, a sequence of instructions, where an MCR instruction that writes a CP15 register is closely followed by an MRC that reads the same register, are executed such that the RAW hazard is not detected and the MRC reads the old value of the register. This scenario can only occur for accesses to one of the following four CP15 registers:

- CacheSizeSelection Register
- Thread and ProcessID user read/write
- Thread and ProcessID user read only
- Thread and ProcessID privilege only

These registers are both readable and writable and have been optimized to execute in a single cycle.

Furthermore, this scenario occurs only when a specific sequence of instructions is executed between the MCR and the MRC. The sequence must meet two criteria:

- It must take less than three cycles to execute
- It must have one of the instructions in the following list:
  - ARM PLD with [Rn, -Rm, <shift>] addressing mode
  - ARM or Thumb PLD with [Rn, Rm,<shift>] addressing mode (unless it is LSL #0 or LSL #2)
  - Thumb or ThumbEE load/store instruction with [Rn, Rm,<shift>] addressing mode (unless it is LSL #0 or LSL #2)
  - Thumb TBB instruction

The issue is reported by ARM, erratum ID 588115, Category 3[1].

Projected Impact:

If this erratum is encountered, the old stale value of the register is read rather than the newly written value, in which case the system software may appear to behave incorrectly. However, the usage model for such a software sequence is unclear, and hence the likelihood of encountering it in practice is very low, especially considering the requirement of the second unrelated instruction that must also fall between the MCR and the MRC.

Workarounds:

If a workaround for this erratum is desired, there are two options. The first simple option is to add a NOP immediately following the MCR register write in any case where encountering this erratum may be a concern. By adding a single NOP, the minimum required cycle window is guaranteed and the erratum does not occur.

The second option is to set bit 16 in the CP15 Auxiliary Control Register. This causes a pipeline flush on every write to the CP15 register and ensures that the RAW hazard condition does not

1. Category 3 defined as: Behavior that is not the originally intended behavior but should not cause any problems in applications.

occur. The second workaround has the advantage of requiring just one change to the CPU configuration that can be done statically. The disadvantage is that it has some impact on the performance of write updates to CP15 registers that would not otherwise require a pipeline flush. This second workaround can be implemented using the following code sequence to be executed in the Secure state:

```
MRC p15, 0, R1, c1, c0, 1           ; read Aux Ctl Register
ORR R1, R1 #(1 << 16)               ; set bit 16 to 1
MCR p15, 0, R1, c1, c0, 1           ; write Aux Ctl Register
```

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Not required because PLD instruction and Thumb mode are not used for affected registers.

WinCE BSP Status:

Fixed in WinCE BSP.

BSP implements software workaround to enable pipeline flush (set bit 16 of Aux Control Register) on writes to CP15 registers.

OSBench indicates a performance impact of up to 14% for a subset of the OSBench tests (interprocess PSL calls). An analysis of the OSBench performance on Cortex-A8 determined that interprocess PSL calls generate excessive cache maintenance.

ENGcm09828 ARM #709718: Load and store operations to shared device memory regions may not complete in program order

Description:

If a sequence of load and store operations are performed to different address locations in a memory region that is marked as shared device, then a load can incorrectly bypass a store.

The issue is reported by ARM, erratum ID 709718, Category 2[1].

Projected Impact:

If the load address and store address are mapped to access the memory region of the same device, and the device relies on memory operations to occur in program order, then this device may not operate as intended.

Workarounds:

The erratum occurs only for the shared device memory regions and not for the non-shared device memory regions. Therefore, this problem can be worked around by using the remap registers to remap all the shared device transactions to the non-shared device. The only difference between the shared device and the non-shared device is the attributes produced for the transaction on the AXI interface. Therefore, the user does not experience any impact in terms of performance from this workaround.

Another possible use of the TEX remap is to map the shared device regions to the strongly ordered transactions. This second remapping option is less desirable as it affects the performance, as strongly ordered transactions are not buffered.

The following code sequence is required to setup and enable the TEX remap. This should be done before enabling the MMU.

```
; Setup PRRR so device is always mapped to non-shared
    MRC p15, 0, r0, c10, c2, 0; Read Primary Region Remap Register
    BIC r0,#3<<16
    MCR p15, 0, r0, c10, c2, 0; Write Primary Region Remap Register

; Enable TEX remap
    MRC p15, 0, r0, c1, c0, 0; Read Control Register
    ORR r0,r0,#1<<28
    MCR p15, 0, r0, c1, c0, 0; Write Control Register
```

Another valid workaround is to place a data memory barrier (DMB) between all the memory accesses to the device regions, where ordering is required between a store and a subsequent load to a different physical address.

Proposed Solution:

No fix scheduled.

1. Category 2 defined as: Behavior that contravenes the specified behavior and that can limit or severely impair the intended use of specified features, but does not render the product unusable in all or the majority of applications.

Linux BSP Status:

Fixed in Linux BSP.

Shared Device memory type is not used. But PRRR setup codes were added before enabling MMU in the bootloader.

WinCE BSP Status:

Fixed in WinCE BSP.

Workaround implemented.

ENGcm10693 ARM #628216: Potential OVFL status loss when it occurs at CP15 and CP14 update

Description:

If the PMU is in use and an overflow event occurs simultaneously with a write to one of the subsets of CP15 and CP14 registers, the overflow event can be lost.

The conditions are as follows:

1. The performance counters must be in use
2. The performance counter must have an overflow (counter value goes beyond 0xFFFF_FFFF)
3. Simultaneous with the counter overflow, a MCR instruction must be executed that writes to one of the following CP14/CP15 registers:
   - Any PMU register other than PMU counter registers
   - ThumbEE Configuration Register
   - ThumbEE Handler Base Register
   - System Control Register
   - Auxiliary Control Register
   - Secure Configuration Register
   - Secure Debug Enable Register
   - Nonsecure Access Control Register
   - Context ID and Thread ID Registers
   - Coprocessor Access Register
   - Cache Size Select Register

The issue is reported by ARM, erratum ID 628216, Category 2[1].

Projected Impact:

If the erratum occurs, the overflow status flag is not set for that counter in the Overflow Flag Status Register, and an interrupt request is not generated, even when the Interrupt Enable Set Register is configured to generate an interrupt on counter overflow.

Workarounds:

The main workaround is to poll the performance counter. The maximum increment in a single cycle for a given event is 2. Therefore, polling can be infrequent as no counter can increment by more than 2^{32} in fewer than 2 billion cycles.

If the main usage model for performance counters is collecting values over a long period, then polling can be used to collect values (and reset the counter) rather than waiting for an overflow to occur. Polling can be done infrequently and overflow can be avoided.

If the main usage model for performance counters relies on presetting the counter to some value and waits for an overflow to occur, then polling can be used to detect when an overflow event is

1. Category 2 defined as: Behavior that contravenes the specified behavior and that can limit or severely impair the intended use of specified features, but does not render the product unusable in all or the majority of applications.

missed. An overflow can be determined to have been missed if the unsigned value in the counter is less than the value preset into the counter. Polling can be done infrequently because of the number of cycles it requires for this check to fail. If the erratum is triggered and an overflow event is missed, the counter sample can be thrown away or the true value can be reconstructed.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

No software workaround is implemented because performance counters are not used and are only for debug.

WinCE BSP Status:

WinCE BSP currently does not use the performance counters. No BSP change required.

ENGcm10702 ARM #687067: BTB invalidate by MVA operations will not work when the IBE bit is set

Description:

All BTB invalidate operations, including BTB Invalidate by MVA operations, by default are implemented as a NOP in the Cortex-A8 processor. These operations can be executed as NOPs as flushing BTB entries are not required by the Cortex-A8 processor for correct functionality, and there is no additional performance penalty for an incorrect branch prediction versus a non-prediction. However, it is possible for BTB operations to be enabled by setting the IBE bit in the CP15 Auxiliary Control Register. When enabled in this fashion, BTB invalidate by MVA operations may not work as intended. Instead of writing zeros to the valid bit of the BTB entry matching the MVA provided, the CP15 “Invalidate Branch Predictor by MVA” operation writes the value currently in the “Instruction L1 System Array Debug Register 0.” This register is not initialized at the reset time and can only be written in secure, privileged modes when CP15SDISABLE is not set.

The conditions are as follows:

1. The branch predictor is enabled (SCTLR.Z = 1)
2. The Auxiliary Control Register IBE bit is set to 1
3. An invalidate Branch predictor by MVA operation is executed
4. The Instruction L1 System Array Debug Register 0 contains a non-zero value which sets the valid bit and clears the page cross bit.

The issue is reported by ARM, erratum ID 687067, Category 3[1].

Projected Impact:

If the non-zero value contained in L1 System Array Debug Register 0 sets the valid bit of the BTB entry, then the entry is not invalidated as intended.

Workarounds:

A workaround for this erratum is, not to enable the IBE bit. ARM recommends that the IBE bit should not be enabled unless it is required for an erratum workaround.

If the IBE is to be enabled, then the L1 System Array Debug Register 0 should be initialized to a zero value. This register is for RAM array debug purposes and is not used as a part of normal functionality. It is only accessible in a privileged secure mode. Therefore, it can be statically initialized as a part of the boot code sequence. If the register is used for debug purposes, the value should be reset to zero when the debug sequence completes.

The code to initialize the L1 System Array Debug Register 0 is as follows:

```
MOV r1, #0
MCR p15, 0, r1, c15, c1, 0 ; write instruction data 0 register
MRC p15, 0, R1, c1, c0, 1 ; read Aux Ctl Register
ORR R1, R1 #(1 << 6) ; set IBE to 1
MCR p15, 0, R1, c1, c0, 1 ; write Aux Ctl Register
```

1. Category 3 defined as: Behavior that is not the originally intended behavior but should not cause any problems in applications.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

No software workaround is implemented because IBE is not set as 1.

WinCE BSP Status:

WinCE BSP does not set the IBE bit. BTB invalidate operations will be NOPs. No BSP change required.

ENGcm10704 ARM #693270: Taking a watchpoint is incorrectly prioritized over a precise data abort

Description:

If a debug watchpoint and a precise data abort are both triggered from the same data access, the ARM Architecture specifies that the data abort should be prioritized. However, this does not occur on the Cortex-A8 and the watchpoint is taken instead.

The conditions for the erratum are as follows:

1. At least one debug watchpoint is programmed
2. A precise data abort occurs on the same address as the watchpoint

The issue is reported by ARM, erratum ID 693270, Category 3[1].

Projected Impact:

The implications of this erratum only affects the debug software. The data abort should take precedence over the watchpoint so that the OS has a chance to fix up paged-out memory before re-executing the instruction and presenting the debugger with the watchpointed address. Due to this erratum, this fix up does not occur and the debugger should be capable of handling a faulting address.

Workarounds:

The workaround for this erratum is to ensure that the debugger software handles the faulting address. When the debugger signals a watchpoint, and identifies that the page being accessed is subjected to an MMU fault, which it would like the OS to patch up before dealing with itself, it can perform the following actions:

- Disable the watchpoint
- Set vector catch on the local Data Abort exception (secure or non-secure, as appropriate)
- Set the PC at the watchpointed instruction and restart execution

The processor restarts, re-executes the instruction and generate the MMU fault. It then fetches the instruction from the Data Abort handler and re-enter Debug state because of the Vector Catch event. The debugger can then perform the following actions:

- Re-enable the watchpoint
- Disable the vector catch
- Set the PC at the Data Abort vector and restart execution

The processor restarts and re-executes the Data Abort vector instruction. The OS then patches up the MMU fault and attempts to re-execute the original instruction. Re-executing the instruction regenerates the Watchpoint debug event, but now the page is properly patched up.

Proposed Solution:

No fix scheduled.

1. Category 3 defined as: Behavior that is not the originally intended behavior but should not cause any problems in applications.

Linux BSP Status:

No software workaround is implemented because the watchpoints are for debug only.

WinCE BSP Status:

Watchpoints are only used by debug software. WinCE uses Microsoft kernel debugger that does not utilize hardware watchpoints. No BSP change required.

ENGcm10713 ARM #507113: A Neon store to device memory can result in dropping a previous store

Description:

If a Neon store is done to Device type memory and is followed in instruction sequence by a load instruction to Device type memory, then it is possible that an unrelated store instruction, which is done to cacheable memory and hit the L1 cache, has its data dropped and therefore, does not update memory.

There are three different memory types defined in the ARM architecture namely, Strongly Ordered, Device, or Normal. Device type memory is one of the three different memory types. This region is specified by the page table entries used by the MMU.

The conditions for this erratum are as follows:

- A Neon store is done to Device type memory.
- A load is executed to Device type memory (any load to Device type memory region, not just from Neon), consecutive to the Neon store.
- Several stores hit the L1 cache. (Any store that hits the L1 cache - Neon or integer core. The address does not matter.)

The issue is reported by ARM, erratum ID 507113, Category 3[1].

Projected Impact:

If the erratum occurs, one or more cacheable stores that hit the L1 cache do not update the cache, leaving stale contents in memory. This is likely to cause observable, incorrect behavior in the application.

The Neon access to memory region marked as Device is not a practical case in general.

Workarounds:

The only workaround for this erratum is to avoid accessing the Device type memory with Neon store instructions. (There should be no practical case for this, anyway). However, if needed, define the region as Strongly Ordered memory, instead.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Not required because Device memory type is not user space accessible.

WinCE BSP Status:

Workaround implemented in WinCE BSP.

1. Category 3 defined as: Behavior that is not the originally intended behavior but should not cause any problems in applications.

The memory which is Device Shared should be mapped as Normal Outer/Inner Non-Cacheable. This is the preferred memory type for RAM memory mapped as NCB. Customer software must avoid Device memory types.

ENGcm10723 ARM #715847: VCVT.f32.u32 can return wrong result in a specific configuration of Floating Point Unit (FPU)

Description:

If the integer to floating point conversion operation, VCVT.f32.u32, is executed with the FPSCR register configured for Default NaN and Flush-to-zero enabled, and the rounding mode used is RP (Round-to-Positive infinity), it returns the incorrect result for the source operation 0xFFFF_FF01. Specifically, it returns the result 0x0000_0000 instead of the correct result 0x4F80_0000. The erratum can occur only for this specific input value and this specific configuration of the FPSCR register.

The conditions are as follows:

1. Default NaN is enabled (FPSCR[25] = 1'b1)
2. Flush-to-zero is enabled (FPSCR[24] = 1'b1)
3. RP rounding mode is enabled (FPSCR[23:22] = 2'b01)
4. A VCVT.f32.u32 instruction is executed with the source operand 0xFFFF_FF01
5. The result of the instruction is incorrect 0x0000_0000 rather than 0x4F80_0000

The issue is reported by ARM, erratum ID 715847, Category 3[1].

Projected Impact:

The incorrect result from the conversion operation can result in further incorrect results calculated and unexpected program behavior.

Workarounds:

The erratum only occurs if the floating point unit is configured in run fast mode with RP rounding. The easiest workaround is to avoid using this particular mode combination. Round-to-Nearest (RN) is a common rounding mode used, but if RP functionality is desired, it should be done without using Default NaN and/or without Flush-to-zero enabled. Default NaN signalling, Flush-to-zero, and rounding mode are all configured using bits [25:22] of the FPSCR register. This register is typically configured by the system software and should not change within an application.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

No software workaround is implemented because the RN mode is used.

WinCE BSP Status:

The WinCE BSP configures the VFP for run-fast mode (default NAN enabled, flush-to-zero enabled), so it is subject to the erratum. WinCE BSP does not specifically configure RP rounding mode which is another condition for this erratum. Our FP support comes from a DLL provided by

1. Category 3 defined as: Behavior that is not the originally intended behavior but should not cause any problems in applications.

ARM. The ARM DLL should avoid the specific rounding mode associated with the erratum. Need to ensure RP rounding mode is not enabled.

ENGcm11134 ARM #468416: Under a specific set of conditions, processor deadlock can occur when L2 cache is servicing write allocate memory

Description:

If a load request is processed which misses the L2 cache, but cannot be immediately forwarded to the BIU, it encounters a special hazard which prevents the request from being required to access the L2 cache RAM again to save power. There can be multiple requestors with unique addresses, (that is, one address per cache line) with this special hazard. All write-allocate requests that access the L2 cache RAM, on port1, do not have address comparators to check for this special hazard condition. So, if a subsequent write-allocate request is issued to the L2 cache RAM on port1 and allocates a victim buffer, then all requests pending with this special hazard must be forced to perform a L2 cache RAM lookup again to maintain memory coherency. There is a 1-cycle window in which the write-allocate request must allocate to a victim buffer and a pending request to the BIU is not prohibited from going to the BIU, such that a deadlock can occur.

The conditions are as follows:

- The processor must have L2 cache present and enabled.
- The L2 cache must be configured to support the write allocate memory type.

The issue is reported by ARM, erratum ID 468416, Category 2[1].

Projected Impact:

If this erratum is encountered and processor deadlock occurs, it can only be interrupted by asserting RESET on the processor.

Workarounds:

The workaround for this erratum is to disable write-allocate by programming the L2 Auxiliary Control Register bit[22], Write allocate disable:

```
MRC p15, 1, r0, c9, c0, 2; read register
ORR r0, r0, #(1&lt;&lt;22); Write allocate disable
MCR p15, 1, r0, c9, c0, 2; write register
```

Disabling write allocate in the L2 cache could have a performance impact for some applications.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Workaround implemented in Linux BSP.

The software workaround is to disable write allocate in the Level 2 cache. This workaround has performance penalty.

1. Category 2 defined as: Behavior that contravenes the specified behavior and that can limit or severely impair the intended use of specified features, but does not render the product unusable in all or the majority of applications.

WinCE BSP Status:

Workaround implemented in WinCE BSP.

Write allocate is disabled in L2 cache control register.

ENGcm11231 ARM #728018: MVA Cache operations for non-cacheable memory region cause deadlock

Description:

If a Clean by MVA, Invalidate by MVA, or Clean and Invalidate by MVA cache maintenance operation is performed in a memory region that is marked non-cacheable, device, or strongly ordered, it is possible for the processor to deadlock or have stale data left in the processor. This erratum occurs when the address hits the cache in a way that is not predicted by the Hash Virtual Address Buffer (HVAB), which is a cache way predictor inside the processor. This erratum can occur only for the cache maintenance operations that are performed by MVA. It does not occur for the set/way based cache maintenance operations.

The conditions are as follows:

1. A memory region is marked cacheable in a page table entry, and a cache line from that region is placed in the data cache.
2. A second page table entry marks the same memory region as non-cacheable, device, or strongly ordered. This can occur by changing the memory attributes in the existing page table entry, or through an alternative page table entry that maps the same virtual to physical address but with non-cacheable, device, or strongly ordered attributes rather than cacheable.
3. A Clean by MVA, Invalidate by MVA, or Clean and Invalidate by MVA cache maintenance operation is done to this address
4. The maintenance operation receives a false hit indication from the HVAB array
5. The maintenance operation receives a true hit indication from the Tag lookup, which implies that the data is present in the array, but located in a different way that is not predicted by the HVAB.
6. An eviction of the dirty line has started but not finished, and the processor leaves stale data in the cache and can potentially enter a deadlock state.

The issue is reported by ARM, erratum ID 728018, Category 2[1].

Projected Impact:

If stale data is left in the cache, the processor does not work as intended. If deadlock state occurs, it can only be exited by asserting the RESET pin on the processor.

Workarounds:

There are two possible workarounds for this erratum.

The first workaround is to avoid performing the cache maintenance operations to non-cacheable addresses previously marked cacheable and therefore may be resident in the cache. If the address is present in the cache, it implies that the memory region is marked cacheable at some earlier point of time and explicitly changed to non-cacheable before the maintenance operation is performed. If the region type is not changed to non-cacheable before executing the maintenance operation, this erratum can be avoided. The value of changing a memory region from cacheable to non-cacheable

1. Category 2 defined as: Behavior that contravenes the specified behavior and that can limit or severely impair the intended use of specified features, but does not render the product unusable in all or the majority of applications.

before performing the maintenance operations is that this is the only way by which the ARM v7 Architecture guarantees that the line is not immediately placed back into the cache due to the possibility of data speculation. However, in Cortex-A8, this degree of data speculation is never done. Therefore, changing the memory type to non-cacheable before executing the cache maintenance operation is not required to assure that the line is not immediately placed back into the cache. However, if there is a code compatibility with other v7 implementations (that may exhibit this level of data speculation) is a concern, then this first workaround is insufficient, and the second workaround should be used.

The second workaround is to execute the loop of cache maintenance operations twice. Execute the loop once with the memory region still marked cacheable. Then change the page table entry to make the memory region non-cacheable and execute the loop for a second time. The first loop cleans the data from the cache in the Cortex-A8. On the Cortex-A8, the second loop is redundant as it misses on all lines in the cache, but resolves the data speculation issue that can occur on a different v7 architecture implementation. The existing cache maintenance code in a dynamically paged environment can be dependent on the maintenance operation triggering a page fault to set the correct page table entry. The workaround code must independently ensure that the correct page table entry is present.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Not required because set/way is used in cache maintenance.

WinCE BSP Status:

Not implemented.

ENGcm08174 CSPI wrongly clears the overrun status bit

Description:

The CSPI automatically clears the overrun error status bit when the RxFIFO is read. This bit should not be cleared. This bit is designed for the interrupt access mode, and not for the DMA access mode.

The conditions are as follows:

- When the RxFIFO overflow/overrun bit is cleared by an RxFIFO read, it does not cause a problem if no DMA accesses to the CSPI occur.
- When DMA is utilized, the interrupt status of RxFIFO overflow/overrun can be lost because of uncontrolled RxFIFO access by DMA.

Projected Impact:

If the RxFIFO is read before reading the Overrun error status bit, it is possible to miss the overrun and thus miss the data.

Workaround:

When DMA is used for data transfer, the software can program the CSPI to allow only the interrupt generation during the overrun condition and not enable any other interrupt sources. In this way, whenever an interrupt comes from CSPI, the software can assume that it is the result of an overrun condition.

BSP Implementation:

BSP uses the PIO mode. No workaround required in the PIO mode.

Silicon Fix:

No fix scheduled.

ENGcm11544 Issue in 32-bit Rx when CSPI is configured in slave mode with SSCTL bit set

Description:

CSPI is configured in the slave mode to receive 32-bit data and SSCTL bit of CONREG is set.

The Rx FIFO is loaded twice for every 32-bit data Rx:

- First—when the 32 bit is received (32 pulses of CSPI clock)
- Second—when the SS signal goes low (SSPOL = 1)

Projected Impact:

When bit cnt is matched, it will store the data in rxfifo, while SSB de-active, it will store the data in shifter to rxfifo, so the last data will be stored twice.

Workaround:

The workaround is to discard one data for a length[11:5]+1 of word valid data received.

BSP Implementation:

No plan to implement. BSP does not support the eCSPI slave mode.

Silicon Fix:

No fix scheduled.

ENGR120443 DCP fails to handle SHA1/SHA256 hashing for inputs that are split across multiple pages

Description:

DCP has the requirement that the length of data in the non-final buffers in a chain be 64-byte aligned for SHA1/SHA256 hashing. When the data in a buffer is not 64-byte aligned, DCP pads it with 1 followed by 0s to make it 64-byte aligned.

In operating systems such as Linux, it is very common for allocated memory to be non-contiguous. When a user allocates memory for some data to be hashed, it may split across multiple physical pages that are not in 64-byte-aligned chunks. In such cases, the DCP will pad the buffer, resulting in the wrong hash being computed. This is because NIST only allows padding at the end of the data, not at the page boundary, if it splits across pages.

Projected Impact:

Crypto API in Linux BSP generates a test vector for SHA1 hashing that is 56 bytes of data split across 2 pages that have 28 bytes each. When the Linux DCP driver is loaded, it has to be registered with the Crypto API in order for users to take advantage of the hardware acceleration. But during registration, the above mentioned test vector fails, resulting in a failure to register the driver for hashing, rendering the DCP useless.

Workaround:

For each 64 B block that crosses page boundary, software will copy these bytes to a contiguous buffer and perform two hashing operations instead of one.

BSP Implementation:

Workaround is already implemented in Linux BSP.

Silicon Fix:

No fix scheduled.

ERR007894 Under certain conditions, the DDR DQS_Gate may close earlier at DDR2 mode, causing failure on DDR read.

Description:

In DDR read access, an internal signal, DQS_Gate, is generated by DDR PHY to filter out noise on DQS signal. If pull-down on DQS signal is not enabled, the DQS_Gate must be opened at the middle of DQS pre-amble, and closed at the DQS post-amble. However, when DDR mode is set to DDR2 mode, a glitch may be generated at the input port of the DQS_Gate close logic under certain condition. If the generated glitch is captured by read DQS, the DQS_Gate close logic will generate incorrect output, which will cause the DQS_Gate to close earlier than expected. Then the DDR PHY will latch in wrong data.

This issue may happen when all the following conditions are met:

1. DRAM_CLASS in Register DRAM_CTL00 is set to b’0100 (DDR2 mode).
2. DDR CLK frequency is lower than 266MHz.
3. ENABLE_HALF_CAS in Register DRAM_PHY02 is set to 1.

Projected Impact:

DDR PHY may latch in wrong data during read access.

Workarounds:

In DDR2 mode, when DDR CLK frequency is lower than 266MHz:

1. In DRAM_PHY02, set ENABLE_HALF_CAS = 0.
2. In IOMUXC_SPADC_PDRAM_SDQS0/1/2/3, set PKE = 1 and PUE = 1.
3. In IOMUXC_SPAD_GDDRMODE_CTL, set DDR_INPUT = 0.
4. In DRAM_PHY03/05/07/09/11, set DQS_TSEL_EN = 0.

BSP Implementation:

Linux BSP adopts 266MHz for DDR2.

Silicon Fix:

No fix scheduled.

ENGcm12051 DPLL: Meta-stability Issue

Description:

The PLL loses frequency lock, drifting either higher or lower than the locked frequency, for a period of ~2 µs, before re-locking itself with no user intervention. The root cause is a meta stable analog signal, which may cause the VCO to adjust the frequency of the output clock significantly out of range. The meta stable signal returns to normal operation after one VCO/4 clock cycle, and the PLL then acts as designed to return the PLL output to the programmed frequency (within ~2 µs).

Projected Impact:

Drifting faster than the lock frequency can result in DDR and internal logic failures. Drifting slower than the lock frequency can result in DDR failures. Depending on system activity and magnitude of frequency drift, corrupt memory transactions may occur, possibly causing system failure.

Workarounds:

The issue can be mitigated by using a Multiplication Factor Numerator (MFN) software workaround which puts the PLL into an operating mode to avoid unintended VCO adjustments in response to any potential meta stable event. The PLL should also be configured into Phase Lock Mode (PLM = 1). The MFN implementation is accomplished by locking the PLL at a higher than targeted frequency (864 MHz), and then changing the MFN to reach the target frequency (800 MHz). This is a specified dither mode PLL function. By running the PLL slower than the locked frequency, even if a meta stable event occurs, erroneous deviations to the VCO are avoided. The workaround applies to devices with a target CPU frequency of 800 MHz operation.

There are two parts to the workaround, and they need to be applied whenever the PLL shuts down and restarts. The implementation below assumes PLL1 is being used as the source for the CPU and DDR clock.

- Part 1: Workaround applied during system initialization (boot code)
  - Workaround should be applied before DDR is initialized when the boot code is running from IRAM.
    - Change ARM clock to be sourced from 24 MHz OSC
    - Disable auto-restart of PLL1 by clearing AREN bit in DP_CONFIG
    - Configure PLL1 multiplication factors for 864 MHz using the following factors: MFI = 8; MFN = 180; MFD = 179; PDF = 0
    - Manually restart PLL1 (RST = 1) with phase and frequency lock (PLM = 1) using DP_CTL
    - Wait for PLL1 to lock by polling lock ready flag (LRF) in DP_CTL
    - PLL1 will now be locked at 864 MHz
    - Update MFN to transition PLL1 to 800 MHz by applying the following factor: MFN = 60

  - Request PLL to load new MFN using DP_CONFIG (LDREQ bit)
  - Wait for acknowledgement of new MFN factor from PLL by polling DP_CONFIG (LDREQ bit)
  - PLL1 will now be locked at 800 MHz
  - Delay 4 µs to avoid PLL instability window
  - Move ARM clock to be sourced from PLL1
- Part 2: Workaround applied to suspend/resume code or whenever PLL1 is disabled/enabled (kernel)
  - Put DDR into self-refresh
  - Change ARM clock to be sourced from 24 MHz OSC
  - Update MFN to transition PLL1 to 864 MHz by applying the following factor: MFN = 180
  - Request PLL to load new MFN using DP_CONFIG (LDREQ bit)
  - No need to wait for new PLL rate, PLL will be disabled during suspend
  - Enter suspend
  - Interrupt wakes system
  - System will resume with PLL1 locked at 864 MHz
  - Update MFN to transition PLL1 to 800 MHz by applying the following factor: MFN = 60
  - Request PLL to load new MFN using DP_CONFIG (LDREQ bit)
  - Wait for acknowledgement of new MFN factor from PLL by polling DP_CONFIG (LDREQ bit)
  - PLL1 will now be locked at 800 MHz
  - Delay 4 µs to avoid PLL instability window
  - Move ARM clock to be sourced from PLL1
  - Move DDR clock to be sourced from PLL1
  - Continue resuming system

Proposed Solution:

No fix scheduled.

BSP Status:

Workaround is implemented through standalone software patch to u-boot and kernel, available on www.freescale.com/imx50tools. Software workaround implemented in BSP releases dated 7/15/2011 or later; refer to the accompanying BSP release notes for details.

ENGR121624 APBHDMA channel may stall on waiting for PIO grant when freeze bit is set/clr frequently

Description:

Frequent setting and clearing of the freeze bit to stop communications to the DRAM MC during the refresh period may cause the APBHDMA channel to stall.

Projected Impact:

None. On previous i.MX chips, it was necessary to set the freeze bit to stop the APBHDMA channel when the DRAM enters refresh. On the i.MX508, the DRAM controller can accept commands during the refresh period. So, although the APBHDMA errata item exists, there is no impact on the i.MX50.

Workaround:

None. The i.MX50 DRAM MC does not require use of the freeze bit; so, this issue should not occur.

Silicon Fix:

No fix scheduled.

ENGR125340 LPDDR2 ZQ calibrator state machine cannot meet 266 MHz design target

Description:

The design target requires the digital state machine that controls the ZQ Calibrator to work at 266 MHz. But calibrator cannot operate at greater than 2.5 MHz due to a large comparator delay.

Projected Impact:

This issue affects the impedance matching between the i.MX50 and the DRAM device. The board design must take care of impedance matching issue.

Workaround:

Calibrate ZQ before initializing DRAM_MC with very low ddr_clk frequency. It is easy to implement this workaround in software, but it cannot compensate for temperature and voltage changes.

To compensate the temperature and voltage changes, calibrate the ZQ during ddr self-refresh period with very low ddr_clk frequency. This might have impact to the software, for both system efficiency and software complexity.

Silicon Fix:

No fix scheduled.

ENGcm10185 eCSPI burst completion by SSB signal in slave mode not functional

Description:

According to the eCSPI specifications, when eCSPI is set to operate in the Slave mode (CHANNEL_MODE[x] = 0), the SSB_CTRL[x] bit controls the behavior of burst completion.

In the Slave mode, the SSB_CTRL bit controls the behavior of SPI burst completion as follows:

- 0—SPI burst completed when (BURST_LENGTH + 1) bits are received
- 1—SPI burst completed when SSB input negated

In the BURST_LENGTH definition, it is stated “In the Slave mode, this field takes effect in SPI transfer only when SSCTL is cleared.” However, the mode SSB_CTRL[x] = 1 is not functional in the Slave mode. Currently, BURST_LENGTH always defines the burst length. According to the SPI protocol, negation of SSB always causes completion of the burst. However, due to the above issue, the data is not sampled correctly in RxFIFO when {BURST_LENGTH+1}mod32 is not equal to {actual burst length}mod32. Therefore, setting the BURST_LENGTH parameter to a value greater than the actual burst does not resolve the issue.

Projected Impact:

Slave mode with unspecified burst length cannot be supported due to this issue. The burst length should always be specified with the BURST_LENGTH parameter and SSB_CTRL[x] should be set to zero.

Workaround:

There is no workaround except for not using the SSB_CTRL[x] = 1 option in the Slave mode. The accurate burst length should always be specified using the BURST_LENGTH parameter.

BSP Implementation:

No plan to implement. BSP does not support eCSPI slave mode.

Silicon Fix:

No fix scheduled.

ENGcm10188 eCSPI should send zeros when the burst is longer than the amount of data in FIFO

Description:

The current IP behavior in Slave mode: when the burst continues beyond the BURST_LENGTH, the eCSPI continues sending the data from TxFIFO and corrupts the contents that has been prepared for the next burst. If there is no more data in the FIFO, then it continues repeatedly sending the last word. This is in case the SSB_CTRL[x] = 0. The SSB_CTRL[x] = 1 mode is not functional.

Projected Impact:

Corrupted SPI data.

Workaround:

None. The functionality does not differ to what is described in the specification and this is not specified facet of the standard.

BSP Implementation:

No plan to implement. BSP does not support eCSPI slave mode.

Silicon Fix:

No fix scheduled.

ENGcm12379 EIM: AUS mode is non functional for devices larger than 32 MB.

Description:

When AUS bit is set, the address lines of the EIM are un-shifted. Due to an error, the address bits 27:24 are shifted when AUS = 1.

Projected Impact:

This mode is related to a unique memory configuration, which is used very rarely. Most systems can work in the default mode (AUS = 0). Board designers should connect the EIM address bus without a shift (A0 -> A0, A1 -> A1, etc.), while working in AUS = 0 mode.

Workaround:

Use the AUS = 0 mode (default) while connecting the address signals without a shift (A0 -> A0, A1 -> A1, etc.).

Proposed Solution:

No fix scheduled.

Linux BSP Status:

NA

WinCE BSP Status:

NA

ENGR138633 Histogram Fractional Macro Block Processing Issue

Description:

The histogram hardware always processes complete macro blocks. For frame buffers that are not an integral multiple of the selected block size, either 8x8 or 16x16, the histogram will check all pixels in the frame buffer that is rounded up to the macro block boundary. This could result in pixels that are not in the desired frame buffer to be checked by the histogram hardware. In other words, this could result in pixels that are outside the frame buffer to affect the histogram result.

The histogram hardware checking module currently exists between the CSC2 and ROT processing engines. The histogram should be relocated to the output of the ROT engine, which has all the controls necessary to detect output frames that are not a multiple of the block size.

Projected Impact:

As an example, let's assume a frame buffer of 12x12 pixels. Regardless of block size selected, the desired frame buffer is not a multiple of the block size, either 8x8 or 16x16. The PXP always processes complete blocks. There are four pixels on the right side and bottom side that will be processed by the PXP. These four vertical columns and horizontal rows of pixels will effect the histogram results even though they are not written to the output buffer. If these redundant pixels contain values that affect the histogram in a way that is different from the way the pixels inside the 12x12 region will, then the histogram will not correctly indicate the state of the 12x12 process output frame buffer.

Workaround:

There are two workarounds:

1. Always guarantee that the processed frame buffer is a multiple of the block size.

2. The right vertical column and/or the bottom horizontal row can be copied to a temporary buffer for processing. The redundant pixels that are outside the actual frame buffer can be overwritten to a state that will not effect the histogram state. Then, the PXP can be programmed to process these regions to return the correct state for pixels that will be inside the desired frame buffer.

Silicon Fix:

No fix scheduled.

ENGR142735 TCE Underrun in EPDC

Description:

This issue is caused by a corner condition: when the LUT[n] finished waveform loading but still working on WB processing, the LUT[n+1:15] waveform loading will be hold. It will be triggered when the working buffer processing time for LUT[n] spans over 2 blanking periods.

Projected Impact:

One frame scan of data will be corrupted on the frame in which the TCE underrun occurred. All subsequent frames will be correct.

Software Workarounds:

The issue may be resolved by implementing one or all of the below workarounds:

1. When update is submitted, ensure the LUT acquires a higher LUT number than current active LUTs.
   - Order— LUT[0], LUT[1], LUT[15]
   - If LUT[15] is in use, TCE Underrun could still occur if new update is sent.
2. Ignore TCE Underrun IRQ
   - If TCE Underrun occurs, EPD hardware will continue to complete the update.
   - On frame where TCE Underrun occurs, the frame's data will be corrupted. All pixels will receive the last value in the FIFO. The rest of frames for an update are accurate.
3. Time update request with Frame End IRQ
   - If the update is submitted to the EPD, just after the VSCAN hold off, it will allow for ~2 full frames or over 23.4 mS to complete WB preprocessing for an 85-Hz panel.
4. Increase DRAM bandwidth to ensure the working buffer processing finishes within 1 frame scan.
   - For example, if the TCE Underrun occurs when the DRAM clock = 160 MHz, increase the DRAM clock to 200 MHz.

BSP Implementation:

- Workarounds 1, 2, and 3 available in the Linux BSP release 11.04.01.
- WINCE EPDC driver always ignores TCE Underrun IRQ (workaround 2 is available).

Silicon Fix:

No fix scheduled.

TLSbo90606 Glitch occurs on SCLK in EPIT when switching clock source

Description:

There is a possibility of an extra pulse occurring on SCLK in the EPIT when switching between the clock sources.

Projected Impact:

It can result in an incorrect counter increment in the EPIT.

Workaround:

Clock source should be changed only when the EPIT is disabled. Following is the method to accomplish this:

- Disable EPIT—EPITCR[0] = 0 (EN = 0)
- Disable EPIT output—EPITCR[23:22] = 00 (OM = 00)
- Disable EPIT capture interrupt—EPITCR[2] = 0 (OCIEN = 0)
- Change clock source—EPITCR[25:24] (CLKSRC), determines which clock source is selected for running the counter
- Clear status register—EPITSR[0] (OCIF), this is a write one to clear register
- Configure EPIT to start count once enabled from load value—EPITCR[1] = 1 (ENMOD = 1)
- Re-enable EPIT EPITCR[0] = 1 (EN = 1), that is, enable EPIT
- Reconfigure output and interrupt

BSP Implementation:

Use case not implemented in Linux BSP.

Silicon Fix:

No fix scheduled.

ENGcm03648 eSDHC AutoCMD12 and R1b polling problem

Description:

According to the Standard Host Controller Spec, for a command with R1b response, a TC interrupt should happen when command with busy state is complete. eSDHC does not implement this feature. So, before sending the next command, we need to do polling on DLA for R1b commands. However, this signaling is not working properly.

Projected Impact:

Cannot poll on DLA in PRSSTAT register to wait for busy state completion.

Software Workarounds:

Poll DAT[0], not DLA, to detect when busy state is over.

Silicon Fix:

No fix scheduled.

ENGcm11088 eSDHC partial block read problem when block size is not a multiple of 4 (applicable to both eSDHCv2 and eSDHCv3)

Description:

The following is a test case description:

- Working with an SD card. Using ADMA1.
- Performing partial block read.
- Write 1 block of length 0x200. (Data written in bytes 0x1, 0x2, 0x3 and so on)
- Reading 2 blocks of length 0x22 each. Reading from the address where the write was done. Start address is 0x512 aligned. Watermark is set as 1 word during read. This read is done using only 1 ADMA1 descriptor in which the total size of the transfer is programmed as 0x44 (2 blocks * 0x22)

In this case, the read transfer is not getting completed. The eSDHC is stuck waiting in the loop for TC bit in the interrupt status register to get set.

Projected Impact:

Minor error

Workaround:

eSDHC should use only block sizes that are multiples of 4, or host driver can insert several dummy bytes to ensure each block has multiples of 4 bytes.

BSP Implementation:

Linux and WinCE use a eSDHC block size that is a multiple of 4.

Silicon Fix:

No fix scheduled.

ENGcm11113 eSDHC problem when ADMA2 last descriptor is LINK or NOP (applicable to both eSDHCv2 and eSDHCv3)

Description:

ADMA2 in eSDHC is used for transfers to/from the SD card. ADMA2 descriptors are created with TRANS, LINK, and NOP descriptors. The issue occurs when the last descriptor (which has the End bit 1) is a LINK descriptor or a NOP descriptor. In this case, eSDHC is completing the transfers associated with this descriptor set, whereas it does not even start the transfers associated with the new data command. For example, if a WRITE transfer is being done to the card using ADMA2, and the last descriptor of the WRITE descriptor set is a LINK descriptor, then the WRITE is successfully finished. Now, if a READ transfer is programmed from the SD card using ADMA2, then this transfer does not go through.

Projected Impact:

Minor error.

Workaround:

Always program TRANS descriptor as the last descriptor.

BSP Implementation:

Linux BSP uses simple DMA rather than ADMA2, so this errata is avoided.

Silicon Fix:

No fix scheduled.

ENGR143117 eSDHCv3 on eSDHCv3 port has setup timing issue in SD SDR mode

Description:

STA report on eSDHCv3 in SDR mode is passed, which is based on EMMC4.4 timing requirement. EMMC4.4 requires 3 ns setup time but SD3.0 requires 6 ns setup time. eSDHCv3 cannot match SD setup timing requirement.

Projected Impact:

May have write timing issue if SD cards require more than 5 ns setup timing.

Software Workarounds:

Once we find write timing issue while connecting SD card, we need to slow down SD clock to 40 MHz in SDR mode. Actually, all the SD cards we have tested don't have such strict timing requirement. All the SD cards in hand can be passed on Linux and WinCE stress test.

Silicon Fix:

No fix scheduled.

ENGcm04874 Glitch on SCLK in GPT while switching the clock source

Description:

There is a possibility of an extra pulse occurring on SCLK in the GPT when switching between the clock sources.

Projected Impact:

It can result in an incorrect counter increment in the GPT.

Workaround:

Changing the clock source should only be done when the GPT is disabled. The following is the method to accomplish this:

- Disable GPT—Write 1'b0 to EN bit of GPTCR
- Disable interrupts—Write 6'b000000 in bits [5:0] of GPTIR
- Configure output mode to unconnected/disconnected—Write zeros in OM3, OM2, and OM1 in GPTCR
- Disable input capture modes—Write zeros in IM1 and IM2 in GPTCR
- Change clock source CLKSRC in GPTCR
- Clear status register—Write 003F in GPTSR
- Set ENMOD in GPTCR
- ENABLE GPT—Write 1'b1 to EN bit of GPTCR. The GPTSR should not be read immediately after changing the clock source (a wait of at least one SCLK is required)

BSP Implementation:

Use case not implemented in Linux BSP.

Silicon Fix:

No fix scheduled.

ENGR139532 OCRAM Setup Timing Issue

Description:

Incorrect parasitic modeling of the WEM pins on the OCRAM has resulted in a setup timing issue on the OCRAM write paths.

Projected Impact:

Review of the logic associated with this timing path has determined that this path can only be activated in one of the following two scenarios:

1) When 8 masters (for example, ARM, DCP, PXP, DCP, AHBMAX, USBOH1, EPDC, SDMA, or FEC) attempt to simultaneously access the OCRAM. This would be highly unlikely to occur as there are only 8 masters on the i.MX50 and some of them would not be used with OCRAM (for example, EPDC).

2) During a GPU2D write to OCRAM. As long as the GPU2D is not used with OCRAM, the issue should not occur.

Workaround:

Because this issue only occurs in the two scenarios described above, the software should be written to prevent those scenarios from occuring. Analysis of FSL drivers and software has determined that neither of the above use cases occur in our deliverables.

Silicon Fix:

No fix scheduled.

ENGR133711 ROM fails to boot from boot partition 1 or boot partition 2 in 4-bit and 8-bit DDR non-fast boot modes

Description:

The ROM attempts to read the extended csd structure from card after setting the card into DDR mode. It should read ext_csd before setting card into DDR mode. This causes the i.MX50 TO 1.1 ROM to fail to boot from boot partition for 4-bit or 8-bit DDR mode.

Projected Impact:

The impact should be minimal as the i.MX50 supports several other eMMC boot modes without this issue.

Workarounds:

There are several workarounds:

1. Boot in fast boot mode. Fast boot is validated to work for all modes (SDR, DDR, 8-bit and 4-bit bus widths).

2. Boot from user partition for non-fast-boot DDR mode. The problem only occurs when booting from a boot partition— no issues occur when booting from a user partition

3. Use single data rate (SDR) mode instead of dual-data rate (DDR) if fast boot is not an option.

Silicon Fix:

No fix scheduled.

ENGcm07150 Erroneous descriptor handling by USBOH module

Description:

This issue can occur if the driver does not set up the buffers correctly. For example, the error could occur if the buffers were declared when the driver submits the request, but during the USB transfer the memory is reassigned. The USBOH core uses the erroneous hrdata (when hresp is high) and results in a false USB transfer at the external interface.

Projected Impact:

False transfer of USB.

Workaround:

The following are the two levels of workaround:

- First level workaround—Configure driver correctly to ensure that the USB data buffer is allocated in software. The issue should only occur if the driver did not set up the buffers correctly. So, the first level workaround is simply to write the driver so that this situation does not occur.
- Second level workaround—If for some reason the issue does occur, the second level workaround is to implement USB bus error handling. In this case, the USB module is reset when an error is detected.

BSP Implementation:

Implemented in WinCE ER2 and Linux ER3.

Silicon Fix:

No fix scheduled.

ERR006308 USB: HOST controller lock-up issue

Description:

The USB host controller can lock-up when a FIFO under run occurs on a non-32-bit aligned data buffer. This applies to both the Host controller and OTG controller in host mode.

Projected Impact:

USB HOST Controller may lock up.

Workaround:

1. Set Stream Disable bit (SDIS) in the USBMODE register. This will force the controller to load an entire packet in the FIFO before starting to transmit on the USB bus. Hence, the FIFO will never under run. This will somewhat reduce the max bandwidth of the USB since there will be idle time as the the controller waits for the entire packet to be loaded.
2. Instead of setting SDIS, the FIFO threshold can be increased such that more data will be in the FIFO before a packet transmit is started. This increases the tolerance to bus latency and avoid FIFO under run. The Threshold can be increased by using higher values for the TXTHRESHOLD filed in the TXFILLTUNING register. The default value is 2 bursts (64 bytes if burst size=8).

Silicon Fix:

No fix scheduled.

ERR004346 WDOG: WDOG SRS bit requires to be written twice

Description:

In order to initiate a software reset through WDOG, the SRS bit should be written twice.

Projected Impact:

WDOG software reset request might be ignored.

Workarounds:

The WDOG SRS software reset bit should be written twice within one period of the 32 kHz clock.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Software workaround implemented in BSP version ER3.

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

Document Number: IMX50CE
Rev. 5
05/2015