Document Number: IMX51CE
Rev. 5, 02/2012

Chip Errata for the i.MX51

This document details the silicon errata known at the time of publication for the i.MX51 multimedia applications processors. The contents of this errata apply to the following devices: IMX51xA, IMX51xC, and IMX51xD.

Table 1 provides a revision history for this document.

Table 1. Document Revision History

Rev. Number	Date	Substantive Changes
0	11/2009	• Initial Release
1	03/2010	• ENGcm09125 - updated status - Case 2 and 3 are fixed in firmware • ENGcm10388 - fixed in firmware • ENGcm10724 - updated headline and description • Changed erratum number ENGcm10149 to ENGcm10150, no change in contents • Added ENGcm10676, ENGcm10967, ENGcm11043, ENGcm10974, ENGcm11060, ENGcm11004, ENGcm11036, ENGcm11041, ENGcm11065 • Added ENGcm11104, ENGcm10183, ENGcm11138, and ENGcm04750 • Updated ENGcm09114 headline • Changed ENGcm10198 number to ENGcm10407 • Changed ENGcm10259 number to ENGcm10260 • Changed ENGcm10267 number to ENGcm10272 • Removed ENGcm10353 - duplicate of ENGcm10356 • Removed ENGcm10342 - duplicate of ENGcm10344 • Added ENGcm11161 and updated ENGcm11065 workaround.

Table 1. Document Revision History (continued)

Rev. Number	Date	Substantive Changes
2	06/2010	• Updated description of ENGcm09107 • Added eSDHCv2/eSDHCv3 erratum ENGcm09399 • Added NFC/Boot ROM erratum ENGcm11189 • Added GPU 2D erratum ENGcm11199 • Added ARM Cortex-A8 erratum ENGcm11205 • Updated workaround for ENGcm07782 • Added CCM erratum ENGcm11208 • Added HSC bypass clarification ENGcm08316 • Added EMI/M4IF erratum ENGcm11226 • Added USB OTG ULPI erratum ENGcm11249 • Added clarification on WEIM 8-bit memory devices support ENGcm11244 • Added erratum ENGcm11408 for removing HS/FS USB TLL support • Updated description of ENGcm10716 • Added OneNAND boot erratum ENGcm11353
3	08/2010	• Changed ENGcm11202 number to ENGcm11226 • Added eSDCTL erratum ENGcm08971 • Added CCM erratum ENGcm08842 • Added ARM NEON erratum ENGcm11422 • Added eSDHC erratum ENGcm11403
	09/2010	Added Linux and WinCE BSP status information
4	07/2011	Added DPLL erratum ENGcm12051
5	02/2012	• Added eSDHC erratum ENGcm12364 • Added NFC erratum ENGcm12362 • Updated ROM erratum ENGcm10656 • Added eSDCTL erratum ENGcm12376 • Added EIM erratum ENGcm12378

Table 2 provides a cross-reference to match the revision code to the revision level marked on the device.

Table 2. Revision Level to Part Marking Cross-Reference

Revision	Package	Device Marking [1]	Mask Set
MCIMX51 3.0	19 x 19 mm	PCIMX511AJM6C	M77X
MCIMX51 3.0	19 x 19 mm	PCIMX512CJM6C	M77X
MCIMX51 3.0	19 x 19 mm	MCIMX512DJM8C	M77X
MCIMX51 3.0	19 x 19 mm	PCIMX513CJM6C	M77X
MCIMX51 3.0	19 x 19 mm	MCIMX513DJM8C	M77X
MCIMX51 3.0	19 x 19 mm	PCIMX514AJM6C	M77X
MCIMX51 3.0	19 x 19 mm	PCIMX515CJM6C	M77X
MCIMX51 3.0	19 x 19 mm	MCIMX515DJM8C	M77X
MCIMX51 3.0	19 x 19 mm	PCIMX516AJM6C	M77X
MCIMX51 3.0	13 x 13 mm	MCIMX511DVK8C	M77X
MCIMX51 3.0	13 x 13 mm	MCIMX512DVK8C	M77X
MCIMX51 3.0	13 x 13 mm	MCIMX513DVK8C	M77X
MCIMX51 3.0	13 x 13 mm	MCIMX515DVK8C	M77X

[1] Part numbers with a PC prefix indicate non-production engineering parts.

The following table summarizes errata on thei.MX51 catergorized by module.

Table 3. Summary of Silicon Errata

Errata	Name	Solution	Page
AIPS			
ENGcm07298	AIPS: Unaligned access causes abort on writes to the internal registers	No fix scheduled	11
ARM			
ENGcm09830	ARM: Load and Store operations on the shared device memory regions may not complete in program order	No fix scheduled	12
ENGcm07788	ARM: A RAW hazard on certain CP15 registers can result in a stale register read	No fix scheduled	14
ENGcm04786	ARM: ARPROT[0] is incorrectly set to indicate a USER transaction for memory accesses generated from user tablewalks	No fix scheduled	16
ENGcm04785	ARM: C15 Cache Selection Register (CSSELR) is not banked	No fix scheduled	18
ENGcm07784	ARM: Cache clean memory ops generated by the Preload Engine or Clean by MVA to PoC instructions may corrupt the memory	No fix scheduled	19
ENGcm07786	ARM: Under a specific set of conditions, a cache maintenance operation performed by MVA can result in memory corruption	No fix scheduled	21
ENGcm07782	ARM: Clean and Clean/Invalidate maintenance ops by MVA to PoC may not push data to external memory	No fix scheduled	23
ENGcm04758	ARM: Incorrect L2 cache eviction can occur when L2 is configured as an inner cache	No fix scheduled	25
ENGcm04761	ARM: Swap instruction, preload instruction, and instruction fetch request can interact and cause deadlock	No fix scheduled	26
ENGcm04759	ARM: NEON load data can be incorrectly forwarded to a subsequent request	No fix scheduled	28
ENGcm04760	ARM: Under a specific set of conditions, processor deadlock can occur when L2 cache is servicing write allocate memory	No fix scheduled	30
ENGcm10230	ARM: Clarification regarding the ALP bits in AMC register	No fix scheduled -Clarified in RM	32
ENGcm10700	ARM: If a Perf Counter OVFL occurs simultaneously with an update to a CP14 or CP15 register, the OVFL status can be lost	No fix scheduled	33
ENGcm10716	ARM: A Neon store to device memory can result in dropping a previous store	No fix scheduled	35
ENGcm10701	ARM: BTB invalidate by MVA operations do not work as intended when the IBE bit is enabled	No fix scheduled	37
ENGcm10703	ARM: Taking a watchpoint is incorrectly prioritized over a precise data abort if both occur simultaneously on the same address	No fix scheduled	39
ENGcm10724	ARM: VCVT.f32.u32 can return wrong result for the input 0xFFFF_FF01 in one specific configuration of the floating point unit	No fix scheduled	41

Table 3. Summary of Silicon Errata (continued)

Errata	Name	Solution	Page
ENGcm11205	ARM: Cache maintenance operations by MVA for a non-cacheable memory region can result in processor deadlock	No fix scheduled	43
ENGcm11422	ARM: A Neon load from device memory type may result in unpredictable behavior including system hang	No fix scheduled	45
CCM			
ENGcm11208	CCM: ARM clock source switch limitation	No fix scheduled	46
ENGcm08842	CCM: System hangs when EMI int1 clock is disabled	No fix scheduled	47
CSPI			
ENGcm08209	CSPI: Incorrectly clears the overrun status bit	No fix scheduled	48
DAP			
ENGcm04789	DAP: Clock synchronization bug prevents access to the IP bus for debug	No fix scheduled	49
ENGcm09395	DAP: Debug ROM address in DAP design is incorrect	No fix scheduled	50
DPLL			
ENGcm04750	DPLL: TOG_SEL bit not cleared after the TOG_DIS bit is set	No fix scheduled	51
ENGcm12051	DPLL: Meta-stability Issue	No fix scheduled	155
eCSPI			
ENGcm09397	eCSPI: Slave select remains asserted after transfer is complete when the SSB POL = 1	No fix scheduled	52
ENGcm10183	eCSPI Burst completion by SSB signal in Slave mode is not functional	No fix scheduled	53
EIM			
ENGcm12378	EIM: AUS mode is non functional for devices larger than 32MB	No fix scheduled	160
EMI			
ENGcm09421	EMI/SRC: Warm reset can not be issued in sleep mode	No fix scheduled	54
ENGcm09424	EMI: Exclusive protocol does not function as defined in the AXI protocol	No fix scheduled	55
ENGcm11244	EMI: WEIM 8-bit memory devices support clarification	No fix scheduled -Clarified in RM	57
EPIT			
ENGcm04773	EPIT: Possibility of additional pulse on src_clk when switching between clock sources	No fix scheduled	58
eSDCTL			
ENGcm06969	eSDCTL: Precharge after write may be delayed	No fix scheduled	59

Table 3. Summary of Silicon Errata (continued)

Errata	Name	Solution	Page
ENGcm08971	eSDCTL: ESDGPR register bits 19 to 31 are not readable	No fix scheduled	60
ENGcm12376	eSDCTL: ESDCTLv2 fails to wait the minimal 200uS between DDR clk & clk enable	No fix scheduled	159
eSDHC			
ENGcm06545	eSDHC: Buffer overrun prevents CPU polling reads when WML is set as 1	No fix scheduled	61
ENGcm09111	eSDHC: Cannot finish a write operation after a block gap stop	No fix scheduled	62
ENGcm06706	eSDHC: CMD12 does not always stop the clock	No fix scheduled	63
ENGcm09107	eSDHC: Does not support Infinite Block Transfer Mode	No fix scheduled	64
ENGcm09114	eSDHC: Interrupt is not forwarded to TZIC when HS-I2C is used with SDMA	No fix scheduled	65
ENGcm09403	eSDHC: Software can not clear DMA interrupt status bit after read operation	No fix scheduled	66
ENGcm09149	eSDHC: Wrong data read from buffer port while WML is 1	No fix scheduled	67
ENGcm10407	eSDHC: Glitch is generated on card clock with software reset or clock divider change	No fix scheduled	68
ENGcm11065	eSDHC: ADMA fails when data length in the last descriptor is less or equal to 4 bytes	No fix scheduled	69
ENGcm11104	eSDHCv2: ADMA transfer error when the block size is not a multiple of four	No fix scheduled	70
ENGcm11161	eSDHCv2: Problem when ADMA2 last descriptor is LINK or NOP	No fix scheduled	72
ENGcm09399	eSDHCv2: eSDHC misses SDIO interrupt when CINT is disabled	No fix scheduled	73
ENGcm12364	eSDHC AutoCMD12 and R1b polling problem	No fix scheduled	157
FEC			
ENGcm04798	FEC: Fast Ethernet Controller (FEC) accesses to NAND Flash Controller (NFC) does not work	No fix scheduled	75
GPT			
ENGcm07200	GPT: Possibility of additional pulse on src_clk when switching between clock sources	No fix scheduled	76
HS-I^{2}C			
ENGcm09194	HS-I^{2}C: Address Issue	No fix scheduled	78
ENGcm08218	HS-I^{2}C: After a read operation, the HS-I2C does not operate properly	No fix scheduled	79
ENGcm07892	HS-I^{2}C: Auto Restart not working	No fix scheduled	80
ENGcm09179	HS-I^{2}C: Clock Stretching Does not Work	No fix scheduled	81
ENGcm07894	HS-I^{2}C: HICR[HIIEN] bit does not mask the interrupts	No fix scheduled	82

Table 3. Summary of Silicon Errata (continued)

Errata	Name	Solution	Page
ENGcm09404	HS-I^{2}C: Read after write from an external device fails	No fix scheduled	83
ENGcm07886	HS-I^{2}C: TDC_ZERO and RDC_ZERO status bits are not cleared	No fix scheduled	84
ENGcm09396	HS-I^{2}C: The associated divider of the HIFSFDR does not operate as expected	No fix scheduled	85
ENGcm09113	HS-I^{2}C: High Speed mode of HS-I^{2}C does not work	No fix scheduled	86
IPU			
ENGcm09277	IPU: Combining error when setting the global alpha value of IC (Image Converter) to 0xFF	No fix scheduled	87
ENGcm09131	IPU/CCM: Configuration for DVFS_PER operation (pixel/EOL/EOF)	No fix scheduled	88
ENGcm09634	IPU: CSC1 + Combining + CSC2 Output is Incorrect	No fix scheduled	89
ENGcm10295	IPU: Error while combining in IC when two simultaneous tasks are involved	No fix scheduled	90
ENGcm08316	IPU: Clarification regarding the bypass mode registers setup for display and camera interfaces	No fix scheduled - spec clarification	91
M4IF			
ENGcm07168	M4IF: Step-by-step mechanism violates AXI protocol	No fix scheduled	92
ENGcm10678	M4IF power-saving mode should not be enabled before DDR is configured	No fix scheduled	93
ENGcm10709	M4IF: Power-saving restriction on FPST due to DDR	No fix scheduled	94
ENGcm11226	M4IF: Reading M4IF status registers of an inactive AXI master or slave stalls entire system	No fix scheduled, clarified in RM	95
NFC			
ENGcm09044	NFC: Auto_erase/auto_program does not latch status correctly	No fix scheduled	97
ENGcm09619	NFC: 8-Sym ECC mode does not work with 512 byte page x16 bus NAND Flash	No fix scheduled	98
ENGcm09575	NFC: Copy back function destination address restriction	No fix scheduled	99
ENGcm09135	NFC: Block write-protect does not support lock-tight	No fix scheduled	100
ENGcm06982	NFC: Block write-protect does not work in automatic operations	No fix scheduled	101
ENGcm09400	NFC: Copy-back does not work properly when addr_op = 01	No fix scheduled	102
ENGcm09186	NFC: Software reset does not work properly under certain conditions	No fix scheduled	103
ENGcm08208	NFC: Status read does not occur at the end of the program, with RBB_MODE = 1	No fix scheduled	104
ENGcm09394	NFC: Unlock registers are reset during warm reset	No fix scheduled	105
ENGcm09970	NFC: NFC does not work properly when RBB_MODE = 0 (read status)	No fix scheduled	106

Table 3. Summary of Silicon Errata (continued)

Errata	Name	Solution	Page
ENGcm09980	NFC: Misses read data when working in Symmetric mode with clock ratio 1:2, and using a 16-bit Flash bus width	No fix scheduled	107
ENGcm10033	NFC: Does not Issue DMA read request using 1/2 Kbyte Page in SDMA mode	No fix scheduled	108
ENGcm10036	NFC: Cannot reach entire address space when addr_op = 1 or 3	No fix scheduled	109
ENGcm10135	NFC: Incorrect ECC Error Detection when NFC_RST bit is set	No fix scheduled	110
ENGcm10150	NFC: Overrides read data in Asymmetric mode using a clock ratio of 1:2, and a 16-bit Flash bus width	No fix scheduled	111
ENGcm10158	NFC: Reads only from the last device using addr_op = 2 without read confirmation	No fix scheduled	112
ENGcm10176	NFC: Does not work when number of iterations are greater than number of devices	No fix scheduled	113
ENGcm10205	NFC Doesn't Protect the Last 10 Spare Bytes of the Last Section in 4KB+218	No fix scheduled	114
ENGcm10245	NFC in RBB_MODE = 1 and atomic operation monitors only rb_b0 instead of the rb_b# of the selected device	No fix scheduled	115
ENGcm10356	NFC: ECC mechanism may fail to report uncorrectable error situation	No fix scheduled	116
ENGcm10344	NFC fails to transfer data in read burst accesses when rready is deasserted	No fix scheduled	117
ENGcm10676	NFC fails to perform ECC encoding in interleave mode if FMP is larger than PS	No fix scheduled	118
ENGcm10967	NFC does not function properly for 4 Kbyte Page Size in interleave mode	No fix scheduled	120
ENGcm11043	NFC issues premature DMA read request in case of TOO configuration in interleave mode	No fix scheduled	121
ENGcm11060	NFC does not perform automatic status read operation (AUTO_STAT) according to ACTIVE_CS	No fix scheduled	122
ENGcm11004	NFC can miss the sampling of the ready/busy signal (R/B) when RBB_MODE = 1	No fix scheduled	123
ENGcm11036	SDMA multi-page read from the NFC, when the WEIM is operating, can result in data corruption or EMI hanging	No fix scheduled	124
ENGcm12362	NFC wrong indication of ECC uncorrectable error occurrence after reading the spare area	No fix scheduled	158
Power Supply			
ENGcm10640	Grounding nonfunctional UHVIO IO pads power rails can cause malfunction in other UHVIO IO cells	No fix scheduled	125
ENGcm11041	Dependency between VCC, VREG and NVCC_HSx supplies	No fix scheduled	126
ROM (Boot Code)			

Table 3. Summary of Silicon Errata (continued)

Errata	Name	Solution	Page
ENGcm10656	Serial boot will fail if HAB_TYPE is PRODUCTION	No fix scheduled	127
ENGcm11189	ROM (Boot)/NFC: NAND Flash Boot fails when one of the unused NANDF_RBx signals are held at low	No fix scheduled	128
RTIC			
ENGcm05863	RTIC: Software reset during one time hash mode corrupts RTIC state machine	No fix scheduled	130
ENGcm10974	RTICv3 memory region unlock feature can cause the RTICv3 to hang	No fix scheduled	131
SAHARA			
ENGcm10334	SAHARA/CCM: Frequency ratio restriction for AHB and IP buses in SAHARA	No fix scheduled	132
SRTC			
ENGcm10272	SRTC: Possible status loss when peripheral power resumes if SRTC in fail state	No fix scheduled	133
SSI			
ENGcm06571	SSI: Data Tx starts from FIFO1 in case Rx is enabled before Tx in sync mode	No fix scheduled	135
ENGcm09220	SSI: If TX_EN bit toggled 5 clk cycles before FS, the data transmission not correct	No fix scheduled	136
ENGcm09222	SSI: Normal Async, Tx is disabled 2 clocks before FS, ddr_stxd is indefinitely high	No fix scheduled	137
ENGcm06569	SSI: Transmission does not take place in case of bit length early frame sync mode	No fix scheduled	138
ENGcm09212	SSI: With TFR bit set, Rx re-enabled, data not accepted according to masking	No fix scheduled	139
ENGcm09668	SSI: Receive Overrun Error Generated at Wrong Time for Watermark Level	No fix scheduled	140
ENGcm11138	SSI: In AC97, 16-bit mode, received data is shifted by four bit locations	No fix scheduled	141
USB			
ENGcm09134	USB: Core device fails to receive two sequential OUT transactions in short time	No fix scheduled	142
ENGcm09110	USB: Device mode ISO error problem	No fix scheduled	143
ENGcm07300	USB: Erroneous descriptor handling by USBOH module	No fix scheduled	144
ENGcm10636	USB: Issue when USB_CLK_ROOT Clock is enabled while PHCD bit is set	No fix scheduled	145
ENGcm11249	USB: USB-OTG port ULPI interface is not supported	No fix scheduled, clarified in RM	147

Table 3. Summary of Silicon Errata (continued)

Errata	Name	Solution	Page
ENGcm11408	USB: High Speed Transceiverless Logic interface (HS-TLL) and Full Speed Transceiverless Logic interface (FS-TLL) USB interfaces are not supported	No fix scheduled, feature removed from the RM	148
VPU			
ENGcm09125	VPU: VC-1 AP bug	Case 1 - no fix scheduled Cases 2 and 3 - fixed in last firmware release for silicon rev. 2.0 and rev 3.0	149
ENGcm10253	VPU H.263-P3 decoding Advanced Intra coding (Annex I) bug	No fix scheduled	151
ENGcm10260	VPU DivX V3.11 Variable-length-decoding (VLC) bug	No fix scheduled	152
ENGcm10388	VPU: MPEG-1 full-pel Motion Vector Update Failure	Fixed in firmware	153
ENGcm10390	VPU: JPEG decoder does not support different AC/DC Huffman tables for Cb and Cr	No fix scheduled	154

ENGcm07298 AIPS: Unaligned access causes abort on writes to the internal registers

Description:

Unaligned access to AIPS can be driven high by SAHARA, DAP, and FEC. If they access the AIPS internal registers during an unaligned access, an ABORT occurs.

Projected Impact:

Unaligned accesses to the AIPS internal registers fail.

Workarounds:

Make only aligned accesses to the AIPS internal registers.

Proposed Solution:

No fix scheduled

Linux BSP Status:

No software workaround required. Linux BSP does not use unaligned access to the AIPS registers.

WinCE BSP Status:

No software workaround required. WinCE BSP does not use unaligned access to the AIPS registers.

ENGcm09830 ARM: Load and Store operations on the 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 release 10.03.00.

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 release ER10_SP1.

Workaround implemented.

ENGcm07788 ARM: A RAW hazard on certain CP15 registers can result in a stale register read

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 occur.

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

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 release ER10_SP1.

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.

ENGcm04786 ARM: ARPROT[0] is incorrectly set to indicate a USER transaction for memory accesses generated from user 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 APROT[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.

ENGcm04785 ARM: 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.

ENGcm07784 ARM: Cache clean memory ops generated by the Preload Engine or Clean by MVA to PoC instructions 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 release 09.12.00.

All cache flush routines have DSB at end.

WinCE BSP Status:

Workaround implemented in WinCE BSP release ER10_SP1.

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.

ENGcm07786 ARM: Under a specific set of conditions, a cache maintenance operation performed by MVA can result in memory corruption

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.

ENGcm07782 ARM: Clean and Clean/Invalidate maintenance ops by MVA to 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 release 09.12.00.

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 release ER1.

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

ENGcm04758 ARM: Incorrect L2 cache eviction can occur when L2 is configured as an 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 release 09.12.00.

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 release ER1.

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.

ENGcm04761 ARM: Swap instruction, preload instruction, and instruction fetch request 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. Confirmed that swap instruction is not used in the WinCE OS.

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

ENGcm04759 ARM: 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.

ENGcm04760 ARM: 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 release 09.12.00.

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 release ER1.

Write allocate is disabled in L2 cache control register.

ENGcm10230 ARM: Clarification regarding the ALP bits in AMC register

Description:

In the register AMC of the Tiger/ARM Platform (0xBASE_0018) the bit ALPEN must be set to 1 and ALP[2:0] must be set to ‘000’. Other combinations are reserved and must be avoided.

Projected Impact:

Memory retention issues unless the guideline is followed.

Workarounds:

None

Proposed Solution:

No fix is scheduled. A clarification will be added to the reference manual.

Linux BSP Status:

Workaround implemented in Linux BSP release 09.12.00.

WinCE BSP Status:

Workaround implemented in WinCE BSP release SDK_1.6.

In the register AMC of the ARM Platform (0xBASE_0018), the bit ALPEN must be set to 1 and ALP[2:0] must be set to ‘000’. Other combinations are reserved and must be avoided.

ENGcm10700 ARM: If a Perf Counter OVFL occurs simultaneously with an update to a CP14 or CP15 register, the OVFL status can be lost

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.

ENGcm10716 ARM: 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, it is possible that an unrelated store instruction that is done to cacheable memory and hit the L1 cache has its data dropped and therefore 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 that relatively close in the instruction stream, the following must occur:

- 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 hit 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 release APR2010.

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.

ENGcm10701 ARM: BTB invalidate by MVA operations do not work as intended when the IBE bit is enabled

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.

ENGcm10703 ARM: Taking a watchpoint is incorrectly prioritized over a precise data abort if both occur simultaneously on the same address

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 is signalled 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.

ENGcm10724 ARM: VCVT.f32.u32 can return wrong result for the input 0xFFFF_FF01 in one specific configuration of the floating point unit

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 (FPSR[24] = 1'b1)
3. RP rounding mode is enabled (FPSR[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

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

mode which is another condition for this erratum. Our FP support comes from a DLL provided by ARM. The ARM DLL should avoid the specific rounding mode associated with the erratum. Need to ensure RP rounding mode is not enabled.

ENGcm11205 ARM: Cache maintenance operations by MVA for a non-cacheable memory region can result in processor 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 that the ARM v7 Architecture guarantees 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

ENGcm11422 ARM: A Neon load from device memory type may result in unpredictable behavior including system hang

Description:

When Neon performs a single byte load from Strongly Ordered or Device type memory with an access size of more than 8 bytes, the system AXI bus issues a burst access which is longer than 8 beats of a single byte.

However, the M4IF is capable of supporting only access with a burst length of less than or equal to 8, as indicated in the *i.MX51 Multimedia Applications Processor Reference Manual* (*MCIMX51).* The AXI bus access with burst length larger than 8 beat cannot complete. The results can be unpredictable and may lead to system hang. The MLEN bit indicates on an error, but it may not be possible to read it if the system hangs.

Conditions for this issue:

1. A single byte Neon load is issued with more than 8 bytes access, for example:
   ```
   Vld1.8 {d0,d1,d2,d3}, [Rs]!
   ```
2. The source address is Strongly Ordered or Device memory type

Projected Impact:

The Neon access to memory region marked as Strongly Ordered or Device are not usually a practical case in general. Note that there are also other reported limitations for Neon access to Device type memory such as ENGcm10716.

Workarounds:

Several software solutions can be proposed for this issue:

- Use 8-bit Neon load with access size of less or equal to 8 bytes. For example:
  ```
  Vld1.8 {d0-d1}, [Rs]!
  ```
  This solution results in some performance degradation.
- Use 16 or 32-bit Neon load instructions instead. For example:
  ```
  Vst1.32 {d0 – d3}, [Rs]!
  ```
  The limitation of this proposal is that the source data address must be 16 or 32-bit aligned.
- Define the memory region as Normal Non-Cacheable type instead of Strongly Ordered or Device memory type. In this case, need to avoid potential memory consistence issues and perform Data Synchronization Barrier (DSB) before other DMA engine access the region for read, as the Write Buffer is enabled.

Proposed Solution:

No fix scheduled

ENGcm11208 CCM: ARM clock source switch limitation

Description:

There are two muxes which select a clock source for pll1_sw_clk (which is also the source for ARM clock). One of them is a regular mux (step select mux), and the other is a synchronous mux. The clock sources are selected by a single register (CCSR). If the pll1_sw_clk_sel bit is cleared (CCSR[2]) and the selection of the regular mux (CCSR[8:7]) is changed at the same time, then the regular mux is likely to switch first and can cause a glitch on pll1_sw_clk and hence on ARM clock and possibly other clocks as well.

Due to above, the CCSR[8:7] bits may only be modified when step_clk is no longer selected. Therefore, CCSR[2] must be cleared in separate register access prior to changing CCSR[8:7].

Projected Impact:

None, if the proposed workaround is implemented.

Workarounds:

The CCSR[8:7] bits can be modified only when step_clk is no longer selected. The ARM clock source selection should be done in two accesses. The CCSR[2] must be cleared in separate register access prior to changing CCSR[8:7].

Proposed Solution:

No fix is scheduled. A clarification is added to the reference manual.

Linux BSP Status:

Partially implemented. Full implementation in next release.

WinCE BSP Status:

Implemented.

ENGcm08842 CCM: System hangs when EMI int1 clock is disabled

Description:

When the emi_int1 clock in the CCM_CCGR5 register is disabled, the transaction to the CCM passes through the INT1 channel of the EMI. However, the BRESP cannot be received by the ARM because the clocks are already turned off. This causes the system to hang.

Projected Impact:

None. User should refrain from disabling the EMI int1 clock.

Workarounds:

None

Proposed Solution:

No fix scheduled. A clarification is added to the reference manual.

ENGcm08209 CSPI: Incorrectly 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 RO 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 RO 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.

Workarounds:

When DMA is used for data transfers, the software can program the CSPI to only allow 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.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not required. DMA mode is not enabled in Linux BSP.

WinCE BSP Status:

Implemented.

ENGcm04789 DAP: Clock synchronization bug prevents access to the IP bus for debug

Description:

There is a synchronization mechanism at the DAP AHB access port (AHB-AP) between the clock domains connected at the SoC level (PCLKDBG and HCLK for dap_sys).The problem occurs in DAPAHBAp DapSync module when synchronizing AhbStateDbg HCLK to AhbStateSyncPCLK. This is a 3-bit bus and each bit is being sampled twice. This method does not ensure valid data on output bus.

Projected Impact:

Invalid data may appear on the output bus.

The DapStateSync bus is also implemented in the same way and suffers from the same issue ZEffectZ: preventing reliable access of the IPs by DAP without CPU intervention.

Workarounds:

Ensure the core is in the active mode and the debug IPs are accessed through the core.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

No software workaround.

WinCE BSP Status:

No software workaround.

ENGcm09395 DAP: Debug ROM address in DAP design is incorrect

Description:

Debug ROM address is defined wrong in DAP design (0x8000_0000 instead of 0x6000_0000).

Projected Impact:

When RVI (the debugger) connects to the i.MX51 (the target), the scan chain can not automatically be built.

Workarounds:

The scan chain must either be manually built or an external RVI script must be used.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

No software workaround.

WinCE BSP Status:

No software workaround.

ENGcm04750 DPLL: TOG_SEL bit not cleared after the TOG_DIS bit is set

Description:

Under certain conditions, the DPLL IP TOG_SEL bit is not cleared after the TOG_DIS bit is set. This issue is random in nature.

Projected Impact:

The proposed workaround resolves the issue.

Workarounds:

A software delay for a fixed amount of time based on TOG_COUNT after the TOG_DIS bit is set.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not required because not used

WinCE BSP Status:

Not required. WinCE BSP does not use the TOG_DIS bit.

ENGcm09397 eCSPI: Slave select remains asserted after transfer is complete when the SSB POL = 1

Description:

When the SSB_POL is set to ‘1’ the associated slave select does not automatically return to an inactive state after the data transaction is complete.

NOTE

This bug only applies to eCSPI modules and is not present in CSPI modules.

The BGA signals CSPI1_MOSI, CSPI1_MISO, CSPI1_SSO, CSPI1_SS1, and CSPI1_RDY are not labeled correctly. The actual source for these signals is one of the on-chip eCSPI modules.

Projected Impact:

This issue produces an undesired behavior in the channels configured to SSB_POL = 1. The ports operate properly when SSB_POL = 0.

Workarounds:

There are two workarounds, as follows:

- The software solution applies to channels configured as SSB_POL = 1. After the data transaction is complete, the SS signal must be reconfigured as a GPIO. The GPIO must be programmed to a low level.
- A hardware solution is to place an inverter on the SS signal and program SSB_POL = 0.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Workaround implemented in Linux BSP release 09.12.00.

WinCE BSP Status:

Workaround implemented in WinCE BSP release ER10_SP1.

ENGcm10183 eCSPI Burst completion by SSB signal in Slave mode is 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

Also, in 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 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 the SSB_CTRL[x] should be set to zero.

Workarounds:

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.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not implemented because eCSPI slave mode is not supported in the Linux BSP driver.

WinCE BSP Status:

Not required. WinCE BSP does not support slave mode.

ENGcm09421 EMI/SRC: Warm reset can not be issued in sleep mode

Description:

There is a problem in WARM_RESET during sleep mode. The WARM_RESET flow is as follows:

1. SRC recognize WARM_RESET case (for example, RESET_IN_B)
2. SRC issues dvfs_req to EMI
3. SRC wait to dvfs_ack
4. If after some time the acknowledge does not arrive a counter in SRC recognizes it and SRC issues a cold RESET. When in Sleep Mode, EMI does not get clocks and therefore results in the issue.

Projected Impact:

Any reset arriving in sleep mode even if it is supposed to be a warm reset is treated as cold reset, for example resetting all EMI registers. DDR content is preserved even though a cold reset is issued because in Sleep mode the memory is already in self refresh.

Workarounds:

None.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

No software workaround.

WinCE BSP Status:

No software workaround.

ENGcm09424 EMI: Exclusive protocol does not function as defined in the AXI protocol

Description:

The exclusive accesses are used as a semaphore mechanism in the AXI protocol. The basic process of an exclusive access is as follows:

1. A master performs an exclusive read from an address location.
2. At some later time a master tries to complete the exclusive operation by performing an exclusive write to the same address location.
3. The exclusive write of a master is signaled as:
   — Successful, if no other master has written to that location between the read and write accesses.
   — Failed, if another master has written to that location between the read and write accesses. In this case the address location is not update.

On some occasions the EMI fails to detect a write access to the location tagged by the exclusive read and does not report on failure at exclusive write. This is happens when the tagged location is accessed indirectly in the middle of a burst.

For example, assume the following sequence:

- Master A tags address nn by performing exclusive read access
- Master B writes to address mm
- Master A tags performs exclusive write to address nn

If address nn = mm, the exclusive tag is cleared (exclusive write is not successful) - as expected.

If master B performs a burst with the first address of the burst is equal nn, the exclusive tag is also cleared as expected.

However, if master B performs a burst where the first address of the burst is not nn, but during the burst it does cross address nn, the exclusive tag is not cleared (exclusive write succeeds) - not as expected.

Projected Impact:

In i.MX51 based systems, only the Cortex-A8 is capable generating exclusive access.

The exclusive access of the core to non-shared memory locations are all handled internally within the Cortex-A8 (CA8) monitor and do not propagate to the EMI.

The exclusive access based semaphore mechanism between the core and other masters in the system is not utilized in practice.

In case this mechanism is still required, it can be successfully operated with non-burst access.

Workarounds:

Exclusive access to EMI works only if the tagged address is accessed directly. The access detection mechanism fails if the tagged location is written by consecutive burst access.

The issue can be avoided if the software constrains the regular write to the tagged location to use non-burst access or use and addresses corresponding to the first word in the burst.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Not required because it does not happen.

WinCE BSP Status:

Not required.

ENGcm11244 EMI: WEIM 8-bit memory devices support clarification

Description:

8-bit memory devices are supported by WEIM interface connecting only to one of the following three locations:

- EIM_DA[7:0] pads
- EIM_DA[15:8] pads
- EIM_D[31:24] pads

Connection to the EIM_D[23:16] pads is not supported.

This erratum clarifies the statement in the *MCIMX51 Multimedia Applications Processor Reference Manual* (MCIMX51RM) that only 16-bit and 32-bit memory devices are supported by the WEIM interface.

Projected Impact:

WEIM 8-bit memory devices are supported according to above description.

Workarounds:

None

Proposed Solution:

No fix scheduled. A clarification is added in the reference manual.

Linux BSP Status:

No software workaround

WinCE BSP Status:

No software workaround

ENGcm04773 EPIT: Possibility of additional pulse on src_clk when switching between clock sources

Description:

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

Projected Impact:

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

Workarounds:

Clock source should be changed only when the EPIT is disabled. A way to accomplish the same is as follows:

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

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not implemented because EPIT is not used now.

WinCE BSP Status:

Not required. BSP does not switch EPIT clk src.

ENGcm06969 eSDCTL: Precharge after write may be delayed

Description:

After application of write commands, tWR clocks are required between the last written data until the precharge command to same bank. In cases where there is also an access to bank x followed by a back-to-back write access to bank y, bank x does not accept a precharge command until the data write of the second access (to bank y) is completed. According to the JEDEC standard, the precharge command to bank x should only be held until it has finished its own data and tWR.

Projected Impact:

Due to the bug, writes to a bank are influenced by data writes to other banks. This results in minor performance degradation.

In a situation where two masters are accessing two different DDR banks and one of the masters performs a new access to a different row, a precharge sent to the first row must be performed before the master can access the new row. In that case the second master must finish its access before the precharge can be issued. This situation also can occur in the case of one master making multiple accesses one after another to two different rows. This produces a minor performance decrease, because the new access is waiting for a precharge command to be issued. Some precharge commands are delayed a few extra clocks. In cases where missed access to any of the banks occurred between the time of the last access and the access to the original bank there is no impact and this limitation does not occur.

Workarounds:

None required, as the performace degradation is approximately 0.1%.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

No software workaround required.

WinCE BSP Status:

No software workaround required.

ENGcm08971 eSDCTL: ESDGPR register bits 19 to 31 are not readable

Description:

The Enhanced SDRAM General Purpose Register (ESDGPR register at address 0xBASE+0x1034) bits 19 through 31 do not return correct value on read.

The write operation works according to the specification.

Projected Impact:

No functional impact.

Workarounds:

None.

Proposed Solution:

No fix scheduled. The reference manual is updated accordingly.

ENGcm06545 eSDHC: Buffer overrun prevents CPU polling reads when WML is set as 1

Description:

For slow CPU read polling with WML (Water Mark Level) as 1, excessive data can be read from the card due to the presence of a flop between the buffer and the system bus. This causes the read buffer to overrun and by this halting the data transfer at both the sides and hanging the operation.

Projected Impact:

CPU polling reads when WML is set to 1 can not be supported.

Workarounds:

Do not set WML to 1. The eSDHC does not allow WML as 1 for CPU polling reads. This scenario is unusual even for single access. A read when WML is set to 2 can still be split into two successive reads.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Not required because WML is not 1.

WinCE BSP Status:

Not required. This case does not happen in WinCE BSP.

ENGcm09111 eSDHC: Cannot finish a write operation after a block gap stop

Description:

After stopping at a block gap during a write operation, when the transfer is resumed, the data transfer can not complete and the transfer complete status bit is not set.

Projected Impact:

The eSDHC can not finish write operations after encountering a block gap stop.

Workarounds:

Do not use stop-at-block-gap during write operations.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Not required because the stop-at-block-gap feature is not implemented.

WinCE BSP Status:

Not required. Stop-at-block-gap is not supported in current driver.

ENGcm06706 eSDHC: CMD12 does not always stop the clock

Description:

When eSDHC attempts to stop the transmission by sending CMD12, the clock does not always stop even though the data transfer is aborted.

Projected Impact:

The result is communication with the SD/MMC card is lost

Workarounds:

After sending CMD12 to abort the data transfer, poll the card clock status to see if it is stopped. If it is stopped, write to the XFER_TYPE register to issue the CMD12 as normal, and make several accesses (write or read, depending on the current transfer direction) to the buffer until the clock is restored.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Does not occur in BSP.

WinCE BSP Status:

Workaround implemented in WinCE BSP release ER1.

ENGcm09107 eSDHC: Does not support Infinite Block Transfer Mode

Description:

The eSDHC does not support infinite data transfers, if the Block Count register is set to one, even when block count enable is not set.

Projected Impact:

The eSDHC does not support infinite Block Transfer Mode.

There is no limitation for single block transfers (with Multi/Single Block Select set to zero, Block Count Enable set to zero, and Block Count set to one).

Workarounds:

The following software workaround can be used instead of the infinite block mode:

1. Set BCEN bit to one and enable block count
2. Set the BLKCNT to the maximum value in Block Attributes Register (BLKATTR) (0xFFFF for 65535 blocks)

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Infinite Block Transfer Mode is not supported in BSP.

WinCE BSP Status:

Not required. BSP does not have such case in driver.

ENGcm09114 eSDHC: Interrupt is not forwarded to TZIC when HS-I^{2}C is used with SDMA

Description:

When HS-I^{2}C is used with SDMA (for example with a camera sensor) along with a read/write to/from an SD card, the eSDHC does not appear to forward the interrupt event to the TZIC (Cortex's interrupt controller).

Projected Impact:

The software ISR is never signaled, and as a result the read/write to the SD card causes it to hang.

NOTE

This erratum only applies to the HS-I^{2}C module and not to the two standard I^{2}C modules.

Workarounds:

Do not use SDMA with the HS-I^{2}C.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Not required because DMA mode is not enabled for HS-I^{2}C in current Linux BSP.

WinCE BSP Status:

Workaround implemented in WinCE BSP release ER7.

ENGcm09403 eSDHC: Software can not clear DMA interrupt status bit after read operation

Description:

After DMA read operation, if hclk is automatically gated off, the DINT status can not be cleared by software.

Projected Impact:

Can not clear bit

Workarounds:

Set HCKEN bit before starting DMA read operation, to disable hclk auto-gating feature; after the DINT and TC bit received when read operation is done, clear HCKEN bit to re-enable the hclk auto-gating feature.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Workaround implemented in Linux BSP release 09.12.00.

Set HCKEN bit before starting DMA read operation to disable hclk auto-gating feature. After the DINT and TC bit is received when read operation is done, clear HCKEN bit to re-enable the hclk auto-gating feature. See drivers/mmc/host/mx_sdhci.c (SDHCI_CLOCK_HLK_EN).

WinCE BSP Status:

Already implemented workaround of disabling auto clock gating feature while a command or data is in progress.

ENGcm09149 eSDHC: Wrong data read from buffer port while WML is 1

Description:

Corrupted data is read if the buffer RAM is read until it is empty and the buffer register only contains one word. At this time, the buffer ready is still active when the Watermark level is 1 (RD_WML register bits is 1 or [{(BLK_SIZE + 3) ÷ 4}% RD_WML = 1).

Projected Impact:

Corrupted data is read from a buffer when the buffer RAM is read.

Workarounds:

Do not use RD_WML and [{(BLK_SIZE + 3) ÷ 4}% RD_WML = 1) configuration for read operations.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Not required because WML is not 1.

WinCE BSP Status:

Not required. BSP does not have such case now.

ENGcm10407 eSDHC: Glitch is generated on card clock with software reset or clock divider change

Description:

A glitch may occur on the SDHC card clock when the software sets the RSTA bit (software reset) in the system control register. It can also be generated by setting the clock divider value. The glitch produced can cause the external card to switch to an unknown state. The occurrence is not deterministic.

Projected Impact:

Potential disruption of SD card operation.

Workarounds:

A simple workaround is to disable the SD card clock before the software reset, and enable it when the module resumes the normal operation. The Host and the SD card are in a master-slave relationship. The Host provides clock and control transfer across the interface. Therefore, any existing operation is discarded when the Host controller is reset.

The recommended flow is as follows:

1. Software disable bit[3] of the System Control register
2. Trigger software reset and/or set clock divider
3. Check bit[3] of the Present State Register for stable clock
4. Enable bit[3] of the System Control register.

Using the above method, the eSDHC cannot send command or transfer data when there is a glitch in the clock line, and the glitch does not cause any issue.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Not required because Linux driver avoids this issue.

WinCE BSP Status:

Partially implemented. Full implementation in next release.

ENGcm11065 eSDHC: ADMA fails when data length in the last descriptor is less or equal to 4 bytes

Description:

A possible data corruption or incorrect bus transactions on the internal AHB bus, causing possible system corruption or a stall, can occur under the combination of the following conditions:

1. ADMA2 or ADMA1 type descriptor
2. TRANS descriptor with END flag
3. Data length is less than or equal to 4 bytes (the length field of the corresponding descriptor is set to 1, 2, 3, or 4) and the ADMA transfers one 32-bit word on the bus
4. Block Count Enable mode

Projected Impact:

Data corruption or incorrect bus transactions on the internal AHB bus, causing possible system corruption or a stall.

Workarounds:

The software should avoid setting ADMA type last descriptor (TRANS descriptor with END flag) to data length less than or equal to 4 bytes. In ADMA1 mode, if needed, a last NOP descriptor can be appended to the descriptors list. In ADMA2 mode this workaround is not feasible due to ENGcm11161.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Not required because ADMA mode is not enabled.

WinCE BSP Status:

Not required. BSP does not have such case.

ENGcm11104 eSDHCv2: ADMA transfer error when the block size is not a multiple of four

Description:

Issue in eSDHC ADMA mode operation. The eSDHC read transfer is not completed when block size is not a multiple of 4 in transfer mode ADMA1 or ADMA2. The eSDHC DMA controller is stuck waiting for the TC bit in the interrupt status register.

The following examples trigger this issue:

1. Working with an SD card while setting ADMA1 mode in the eSDHC
2. Performing partial block read
3. Writing one block of length 0x200
4. Reading two blocks of length 0x22 each. Reading from the address where the write operation is performed. Start address is 0x512 aligned. Watermark is set as one word during read. This read is performed using only one ADMA1 descriptor in which the total size of the transfer is programmed as 0x44 (2 blocks of 0x22).

Projected Impact:

The issue exists only when the block size is not a multiple of 4 in transfer mode ADMA1 or ADMA2.

Workarounds:

When the ADMA1 or ADMA2 mode is used and the block size is not a multiple of 4, the software should set the block size to:

$$4 \cdot \left\lceil \frac{\text{block size}}{4} \right\rceil$$

In other words, the block size should be rounded to the next multiple of 4 bytes. In case of write, the software should add the corresponding number of bytes at each block end, before the write is initialized. In case of read, the software should remove the dummy bytes after the read is completed.

For example, if the original block length is 22 bytes, and there are several blocks to transfer, the software should set the block size to 24. The following data is written/stored in the external memory:

```
4 Bytes valid data
4 Bytes valid data
4 Bytes valid data
4 Bytes valid data
4 Bytes valid data
2 Bytes valid data + 2 Byte dummy data
4 Bytes valid data
4 Bytes valid data
4 Bytes valid data
4 Bytes valid data
4 Bytes valid data
2 Bytes valid data + 2 Byte dummy data
```

In this example, 48 (24 x 2) bytes are transferred instead of 44 bytes. The software should remove the dummy data.

Alternatively, the PIO mode can be used if the block size is non-4 byte aligned.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not required because ADMA mode is not enabled.

WinCE BSP Status:

Not required. BSP does not have such case.

ENGcm11161 eSDHCv2: Problem when ADMA2 last descriptor is LINK or NOP

Description:

ADMA2 mode in the eSDHC is used for transfers to/from the SD card. There are three types of ADMA2 descriptors: TRANS, LINK or NOP. The eSDHC has a problem when the last descriptor (which has the End bit '1') is a LINK descriptor or a NOP descriptor.

In this case, the eSDHC completes 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 operation is performed on 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:

Limitation for configuring the last descriptor as LINK or NOP.

Workarounds:

Software workaround is to always program TRANS descriptor as the last descriptor.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not required because ADMA mode is not enabled.

WinCE BSP Status:

Not required. BSP does not have such case.

ENGcm09399 eSDHCv2: eSDHC misses SDIO interrupt when CINT is disabled

Description:

An issue is identified when interfacing the SDIO card. There is a case where an SDIO interrupt from the card is not recognized by the hardware, resulting in a hang.

If the SDIO card lowers the DAT1 line (which indicates an interrupt) when the SDIO interrupt is disabled in the eSDHC registers (that is, CINTEN bits in IRQSTATEN and IRQSIGEN are set to zero), then, after the SDIO interrupt is enabled (by setting the CINTEN bits in IRQSTATEN and IRQSIGEN registers), the eSDHC does not sense that the DAT1 line is low. Therefore, it fails to set the CINT interrupt in IRQSTAT even if DAT1 is low.

Generally, CINTEN bit is disabled in interrupt service.

The SDIO interrupt service steps are as follows:

1. Clear CINTEN bit in IRQSTATEN and IRQSIGEN.
2. Reset the interrupt factors in the SDIO card and write 1 to clear the CINT interrupt in IRQSTAT.
3. Re-enable CINTEN bit in IRQSTATEN and IRQSIGEN.

If a new SDIO interrupt from the card occurs between step 2 and step 3, the eSDHC skips it.

Projected Impact:

The issue is relevant only for the SDIO card interrupt usage.

Workarounds:

The workaround interrupt service steps are as follows:

1. Clear CINTEN bit in IRQSTATEN and IRQSIGEN.
2. Reset the interrupt factors in the SDIO card and write 1 to clear CINT interrupt in IRQSTAT.
3. Clear and then set D3CD bit in the PROCTL register. Clearing D3CD bit sets the reverse signal of DAT1 to low, even if DAT1 is low. After D3CD bit is re-enabled, the eSDHC can catch the posedge of the reversed DAT1 signal, if the DAT1 line is still low.
4. Re-enable CINTEN bit in IRQSTATEN and IRQSIGEN.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Workaround implemented in Linux BSP release 09.12.00.

WinCE BSP Status:

Not implemented yet

Description:

ENGcm11403 eSDHCv2: Erroneous CMD CRC error and CMD Index error may occur on sending new CMD during data transfer

When sending new, non data CMD during data transfer between the eSDHC and EMMC card, the module may return an erroneous CMD CRC error and CMD Index error. This occurs when the CMD response has arrived at the moment the FIFO clock is stopped. The following bits after the start bit of the response are wrongly interpreted as index, generating the CRC and Index errors.

The data transfer itself is not impacted.

The rate of occurrence of the issue is very small, as there is a need for the following combination of conditions to occur at the same cycle:

- The FIFO clock is stopped due to FIFO full or FIFO empty
- The CMD response start bit is received

Projected Impact:

Sending new, non data CMD during data transfer between eSDHC and EMMC card may not succeed with indication on CRC and Index errors.

Workarounds:

The recommendation is to not set FIFO watermark level to a too small value in order to reduce frequency of clock pauses.

The problem is identified by receiving the CMD CRC error and CMD Index error. Once this issue occurs, one can send the same CMD again until operation is successful.

Proposed Solution:

No fix scheduled.

ENGcm04798 FEC: Fast Ethernet Controller (FEC) accesses to NAND Flash Controller (NFC) does not work

Description:

The FEC only creates wrap burst accesses upon data access (non-buffer descriptor). The NFC under EMIv2 cannot handle such accesses from the MAX (M2) input port.

Projected Impact:

The FEC cannot use the NFC to access memory.

Workarounds:

The FEC should use the DDR for its buffer.

Proposed Solution:

No fix scheduled

Linux BSP Status:

The case does not occur in Linux BSP.

WinCE BSP Status:

Implemented.

ENGcm07200 GPT: Possibility of additional pulse on src_clk when switching between clock sources

Description:

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

Projected Impact:

The bug can produce an incorrect counter increment in the GPT when switching between the clock sources.

Workarounds:

Changing the clock source should only be done when the GPT is disabled. A way to accomplished this is as follows:

1. Disable GPT—Write 1'b0 to EN bit of GPTCR
2. Disable interrupts—Write 6'b000000 in Bits [5:0] of GPTIR
3. Configure Output Mode to unconnected/ disconnected—Write zeros in OM3, OM2, OM1 in GPTCR
4. Disable Input Capture Modes—Write zeros in IM1,IM2 in GPTCR
5. Change clock source CLKSRC in GPTCR
6. Clear Status register—Write 003F in GPTSR
7. Set ENMOD in GPTCR
8. 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).

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not required because GPT parent is not changed in current code.

WinCE BSP Status:

Workaround implemented in WinCE BSP release ER7.

ENGcm11199 GPU2D: Transformed images have artifacts, OpenVG core gradient issue

Description:

Due to an issue with ALU precision, several features of the GPU2D hardware incur accumulation of errors, which can result in visual artifacts. This can affect gradient rendering for certain step values and image transformation for larger render target sizes.

Projected Impact:

The issue can cause artifacts in gradient rendering for certain step values and image transformation for larger render target sizes.

Workarounds:

This issue will be fixed in the software driver libraries resolving the visual artifacts. There is no customer-side workaround for earlier driver versions.

Proposed Solution:

No hardware fix scheduled. This issue will be addressed in the software drivers in the next driver release.

WinCE BSP Status:

Workaround implemented in WinCE BSP release ER1004 (for i.MX51).

ENGcm09194 HS-I^{2}C: Address Issue

Description:

When HS-I^{2}C is configured for 7-bit address master mode operation, and the slave does not acknowledge the address, the HS-I^{2}C starts generating high frequency clocks on the SCL line. This bug is only applicable to certain addresses, as follows:

00_00001—This combination is reserved for CBUS operation.
01_00001—Issue address
10_00001—Issue address
11_00001— Issue address

There is no issue with other addresses.

Projected Impact:

The impact of this bug is minor (loss of 3 addresses out of 127) and it is unlikely that amount of devices that are attached to the HS-I^{2}C bus in an actual application reaches that number.

NOTE

This erratum only applies to the HS-I^{2}C module and not to the two standard I^{2}C modules.

Workarounds:

The workaround is to ensure that the above addresses are not used while configuring the hardware on the board.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

HS-I^{2}C is not supported.

WinCE BSP Status:

Workaround implemented in WinCE BSP.

ENGcm08218 HS-I^{2}C: After a read operation, the HS-I^{2}C does not operate properly

Description:

When writing the WA(0x15) in the FIFO (HITDR), the bit TDE remains asserted, the write in the FIFO is not detected.

Projected Impact:

During data transfers the device address is sent twice (instead of the address + WA). but, if the FIFO is written a second time just after the first FIFO write, the TDE is cleared and the read works as expected.

NOTE

This erratum only applies to the HS-I^{2}C module and not to the two standard I^{2}C modules.

Workarounds:

There are two software workarounds, as follows:

- Replace the simple write in FIFO register with the following code:

```
while(MEM_GET_BITS16(HSI2C_HISR, BIT1)!=0) {MEM_WRITE16(HSI2C_HITDR, *data)
```

- Generate the start signal only after filling both HIMADR and HITDR with data. This workaround is under review.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

HS-I^{2}C is not supported.

WinCE BSP Status:

Workaround implemented in WinCE BSP.

ENGcm07892 HS-I^{2}C: Auto Restart not working

Description:

In HS-I^{2}C, the purpose of the HICR[AUTO_RSTA] bit is to generate a repeat start when NOACK is received in Master mode Tx operation with FIFO enabled or FIFO and DMA enabled mode.

When the HICR[AUTO_RSTA] (AUTO repeat start generation) bit is set and NOACK is received in address phase. The HS-I2C is not generating the repeat start.

Projected Impact:

The auto-restart feature does not work.

NOTE

This erratum only applies to the HS-I^{2}C module and not to the two standard I^{2}C modules.

Workarounds:

Use the HICR[RSTA] bit for repeat start. There is a NOACK status (HISR[RXAK]) bit. The corresponding interrupt should always be enabled and AUTO_RSTA should be disabled.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

HS-I^{2}C is not supported.

WinCE BSP Status:

Workaround implemented in WinCE BSP.

ENGcm09179 HS-I^{2}C: Clock Stretching Does not Work

Description:

If the slave device uses clock stretching to delay the HS-I^{2}C controller after every byte is sent, the i.MX51 waits for the clock to go high between bytes but does not wait between the final byte and the STOP signal. Because of this, no valid STOP signal is seen on the bus.

In other words, the use of HIRDCR automatically generates a stop condition on the data bus (HIRDCR[RDC_RSTA] = 0) after the final byte of data is received regardless of whether the slave is in a condition to accept it. If the slave device holds the clock low (clock stretch) after the final byte of data is transferred, the slave never receives the immediate STOP condition from the i.MX51.

Projected Impact:

TBD

NOTE

This erratum only applies to the HS-I^{2}C module and not to the two standard I^{2}C modules.

Workarounds:

The workaround is to use the GPIO alternate input to the I2C1_CLK pad to detect when the external device has released the clock and only after sending the STOP signal.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

HS-I^{2}C is not supported.

WinCE BSP Status:

Workaround implemented in WinCE BSP.

ENGcm07894 HS-I^{2}C: HICR[HIIEN] bit does not mask the interrupts

Description:

The HICR[HIIEN] bit that should operate as a interrupt enable bit, does not mask the interrupts.

Projected Impact:

The HICR[HIIEN] bit does not work.

Workarounds:

Use the individual interrupt mask bits of the status register in the Interrupt Mask Register (HIIMR) or disable them in TZIC.

NOTE

This erratum only applies to the HS-I^{2}C module and not to the two standard I^{2}C modules.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

HS-I^{2}C is not supported.

WinCE BSP Status:

Workaround implemented in WinCE BSP.

ENGcm09404 HS-I^{2}C: Read after write from an external device fails

Description:

Read after write from an external device fails randomly

Projected Impact:

The first data is not written before START condition therefore the Read after write from an external device fails randomly

NOTE

This erratum only applies to the HS-I^{2}C module and not to the two standard I^{2}C modules.

Workarounds:

See "7-Bit/10-Bit Address Master Transmit" section of the "High Speed Inter IC (HS_I2C)" chapter in the *i.MX51 Applications Processor Reference Manual (MCIMX51RM)*. Item 6 in the section reads: "Write one Byte of DATA to be transmitted to the Tx Data Register (HITDR)." Before the Start condition, data must be written to HITDR and the second data has to be written to HITDR after the ADDRESS phase of Byte Transfer Done (BTD). Going forward, Data must be written to the HITDR in advance before the BTD of previous data has been issued.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

HS-I^{2}C is not supported.

WinCE BSP Status:

Workaround implemented in WinCE BSP.

ENGcm07886 HS-I^{2}C: TDC_ZERO and RDC_ZERO status bits are not cleared

Description:

In the HS-I^{2}C module the HISR[RDC_ZERO] status bit is not cleared when HIRDCR[RDC_RSTA] is not set.

Projected Impact:

The HS-I^{2}C Interrupt may occur unexpectedly.

NOTE

This erratum only applies to the HS-I^{2}C module and not to the two standard I^{2}C modules.

Workarounds:

There are two software workarounds, as follows:

1. Use the mask bit to clear the HS-I^{2}C interrupt and enable it again after generating the start (MSTA bit setting) condition when required.
2. Avoid this issue by eliminating the use of the RDC and TDC counters.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

HS-I^{2}C is not supported.

WinCE BSP Status:

Workaround implemented in WinCE BSP.

ENGcm09396 HS-I^{2}C: The associated divider of the HIFSFDR does not operate as expected

Description:

When 9 is written to the HIFSFDR, the ipg_hsi2c_clk should be divided by 72, but the output is incorrect.

Projected Impact:

Specifically when the ipg_hsi2c_clk = 26/3 MHz the expected output is 120 kHz (approximately) where as the actual output is approximately 103.3 kHz.

NOTE

This erratum only applies to the HS-I^{2}C module and not to the two standard I^{2}C modules.

Workarounds:

The I^{2}C protocol works with range of frequencies and an accurate frequency is not a requirement.

In I^{2}C protocol FS mode, bit by bit clock stretching by slave device is possible.

If a very accurate frequency is required, software may need to execute the following steps:

1. Configure HIFSFDR according to the SCL clock frequency required.
2. Check if it works.
3. If it does not work, configure for a lower division frequency until it works.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

HS-I^{2}C is not supported.

WinCE BSP Status:

Workaround implemented in WinCE BSP.

ENGcm09113 HS-I^{2}C: High Speed mode of HS-I^{2}C does not work

Description:

The HS-I^{2}C module does not operate in high speed mode.

Projected Impact:

The high speed mode of the HS-I^{2}C module can not be used.

Workarounds:

None.

NOTE

This erratum only applies to the HS-I^{2}C module and not to the two standard I^{2}C modules.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

HS-I^{2}C is not supported.

ENGcm09277 IPU: Combining error when setting the global alpha value of IC (Image Converter) to 0xFF

Description:

If the alpha value is set to 0xFF for IC (Image Converter) combining, combining result is incorrect. This is relevant to the IC module only.

Projected Impact:

Very minor difference in result because we do not use full masking visible only in very special case.

Workarounds:

The workaround is to use any value between 0x0 to 0xFE and to avoid the usage of 0xFF.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Not implemented because i.MX51 release 3 does not show such an issue.

WinCE BSP Status:

Global alpha is supported for overlay surfaces. The IC combining unit is only utilized for the multiple overlay use case (for the use case of just 1 overlay, the DP combining unit is used).

First, this issue is only encountered in practice through the display driver when multiple overlays are active. When only one overlay is being combined with the UI frame buffer, this combining operation is performed in the IPU's Display Processor (DP). In the DP, an alpha value of 0xFF is valid and does not cause a combining error. When using “n” multiple overlays, the first “n-1” overlays are combined using the IC, where the problem exists.

The workaround for this issue is for the display driver to check for a global alpha of 0xFF and program it into the IC as 0xFE. The result is equivalent in observation - the graphics component is fully opaque in the combined image when using the value 0xFE, just as it is with the value 0xFF. The resultant combined pixel may be very slightly different, but this change is negligible and non-discernible.

ENGcm09131 IPU/CCM: Configuration for DVFS_PER operation (pixel/EOL/EOF)

Description:

When the IPU frequency change is configured to End-of-Line or End-of-Frame operation, the frequency change ready ack is stuck due to and early EMI ACK to the DVFS frequency change request (EMI stops transfers until the frequency changes). When the IPU frequency change is configured to “pixel clock”, the ACK from the IPU to the CCM is granted rapidly (before the EMI can stop the transfer) and system functions correctly at the new DVFS frequency. In order to support End-of-Line or End-of-Frame configuration, the CCM must send the DVFS frequency change request to the EMI after all the ACKs are received from other masters (IPU, HSC).

Projected Impact:

There is no support for End-of-Line and End-of-Frame IPU frequency changes when using DVFS.

Workarounds:

Configure IPU for DVFS changes only on pixel boundary.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Implemented, but not sure if pixel mode works or is valid for all LCDs.

WinCE BSP Status:

There is no DVFS transitions which affect the IPU, so this erratum has no effect on the system.

ENGcm09634 IPU: CSC1 + Combining + CSC2 Output is Incorrect

Description:

This issue applies to the Image Processing Unit's Pre-processing and Post-processing. When the flow of CSC1 + Combining + CSC2 is activated in the Image Convertor and the input to CSC1 is in YUV (4:2:0 or 4:2:2) format, then every second pixel in the output is black resulting in 50% black columns on the screen.

The issue is observed for CSC1 outputs of YUV or RGB.

The issue disappears if CSC2 is off. In addition, the issue disappears if the input to CSC1 is RGB (8:8:8 or 5:6:5).

The combined picture is not impacted by black pixels

Projected Impact:

Two CSC (color space conversions) can not be performed in the IC (Image Converter) if the input to the first CSC is in YUV format.

Workarounds:

The CSC function in the IC requires 3 cycles per pixels. The CSC function in the DP (Display Processor) requires one cycle per pixel.

Due to the very high load on the IC module when CSC2 is used, it was never planned to use CSC2. If a second CSC is required, the DP should be used instead.

In a use case where CSC has to be done twice, the user should perform the first CSC in the IC and the second one in the DP.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

This case does exist. The BSP performs CSC for the graphic layer. The workaround is to use software CSC instead of enabling CSC2.

ENGcm10295 IPU: Error while combining in IC when two simultaneous tasks are involved

Description:

The problem can happen under the following conditions:

1. Two simultaneous tasks are running in the IC.
2. Both tasks perform combining.
3. Local alpha is used.

The IC reads a set of 4 pixels and 4 alpha values. The IC has a 2 bit pointer that points to the current alpha value to be used (out of 4).

Due to an error during task switching there's a situation where the alpha pointer is not updated correctly. As a result the pointer of task1 is used for task2. Hence, a wrong alpha value is used.

The IPU overcomes the problem on the following task switching.

Projected Impact:

The actual visual impact is very small as the problem may happen sporadically and the IPU overcomes it on the next task switching. The problem affects the correctness of the data only so wrong data can be displayed for a short period of time.

Workarounds:

1. Use global alpha for combining.
2. Perform one of the combining tasks in the DP.
3. Run the flows through the IC in consecutive order.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Not implemented. The case may occur using PP and PRP at the same time while using IC local alpha blending function for both tasks. But no plan to fix it because this case is required by the users and the issue impact is small.

WinCE BSP Status:

WinCE BSP does not have such case.

ENGcm08316 IPU: Clarification regarding the bypass mode registers setup for display and camera interfaces

Description:

The registers setup described in Table 4 is required for proper display and sensor interfaces operation. This configuration bypasses the unsupported HSC module (removed from specification) and configures the system for legacy mode operation.

Table 4 shows bypass mode registers setup.

Table 4. Bypass Mode Registers Setup

Register	Address	Access	Reset Value	Required Bypass Value
MCG Control Designation Register (MCD)	0x83FD_C000	R/W	0x0000_0000	0x0000_0F00
MCG CCM Control Register (MCCMC)	0x83FD_C0D8	R/W	0x0000_0000	0x0000_000C
MXT Configuration Register (MXT_CONF)	0x83FD_C800	R/W	0x0000_0000	0xF003_008B

Projected Impact:

None.

Workarounds:

Setup the registers accordingly.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

BSP follows the clarification.

WinCE BSP Status:

Partially implemented. Need to fully align with this workaround on next BSP release.

ENGcm07168 M4IF: Step-by-step mechanism violates AXI protocol

Description:

If the step-by-step mechanism is enabled while the M4IF is operational (there are transactions in the internal buffers), there can be a situation where one or more of the arbitrations' AXI protocol is violated in the “write address channel” or “read address channel.”

Projected Impact:

The M4IF violates the AXI protocol and crashes if step-by-step is set ON or OFF during the run-time.

Workarounds:

Before entering the step-by-step mode, configure the EMI to the software LPMD and then read the LPACK register. The LPACK register indicates that the M4IF is idle and the step-by-step can be enabled. The difference between the regular procedure of LPMD and this procedure is that the EMI clocks remain ON.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not required

WinCE BSP Status:

Not required

ENGcm10678 M4IF power-saving mode should not be enabled before DDR is configured

Description:

Enabling the power-saving mode in M4IF before configuring the DDR module causes deadlock during DDR configuration.

Projected Impact:

None

Workarounds:

Initialization must be performed first, followed by enabling the M4IF power saving.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Workaround implemented in Linux BSP release 09.12.00.

WinCE BSP Status:

Workaround implemented in WinCE BSP.

ENGcm10709 M4IF: Power-saving restriction on FPST due to DDR

Description:

The FPST (Fast arbitration Power Saving Timer) control bit field in the M4IF MCR1 register defines the timer value of the power saving mode of the fast arbitration. The value in this field represents clock cycles of the fast memory arbitration clock multiplied by 8.

The minimum value for FPST in MCR1 reg is defined as follows:

The minimum time interval between LPMD/DVFS requests requires that two refreshes be sent to memory during that interval. So the minimum time is tXSR + tRFC, which is the time from exit self-refresh to first refresh command and the time from first refresh to second refresh.

Projected Impact:

None.

Workarounds:

M4IF and DDR control programing must meet this restriction:

tXSR (DDR parameter) + tRFC (DDR parameter) < 8 × FPST (M4IF parameter)

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Workaround implemented in Linux BSP release 09.12.00.

WinCE BSP Status:

Workaround implemented in WinCE BSP.

ENGcm11226 M4IF: Reading M4IF status registers of an inactive AXI master or slave stalls entire system

Description:

Reading the M4IF status registers of an inactive AXI master or slave ports stalls the entire IC system. This occurs when a specific master or slave clock is not provided to the M4IF, but the ARM/JTAG tries to read that master's status bits.

When the master or slave clock is not active, the read request fails to propagate the status bits that go through synchronization (IPG_CLK to *_clk), acknowledge never comes back, and the entire chip's IP bus is stuck.

Some M4IF status registers bundle the status of several masters or slaves. If one of the masters or slaves is inactive, accessing such a register stalls the entire system, even if the user is interested only in the status of other active ports.

Table 5 lists the impacted registers and the clock signals that should be active for read access to succeed.

Table 5. List of Impacted Registers

Registers	Required Master/Slave to be Active for Read Action to Succeed
MDCR	fast, slow, int1, int2
WMIS0	fast
WMIS1	fast
MLEN	m0, m1, m2, m3, m4, m5, m6, m7, fast, slow, int1, int2
FDPS	fast
SSRL0	fast
SSRL1	fast
SSRH0	fast
SSRH1	fast
MDSR0	The arbitration domain selected by MDCR/RARB
MDSR1	The arbitration domain selected by MDCR/RARB
MDSR2	The arbitration domain selected by MDCR/RARB
MDSR3	The arbitration domain selected by MDCR/RARB
MDSR4	The arbitration domain selected by MDCR/RARB
MDSR5	The arbitration domain selected by MDCR/RARB
MDSR6	The arbitration domain selected by MDCR/RARB
MDSR7	The arbitration domain selected by MDCR/RARB
MDSR8	The arbitration domain selected by MDCR/RARB
SBS0	The arbitration domain selected by MDCR/RARB
SBS1	The arbitration domain selected by MDCR/RARB

Table 5. List of Impacted Registers (continued)

Registers	Required Master/Slave to be Active for Read Action to Succeed
PSM0	m0,m1
PSM1	m2,m3
PSM2	m4,m5
PSM3	m6,m7
MDCR	fast, slow, int1, int2
MCR0	fast, slow, int1, int2

Projected Impact:

No impact on regular functionality. The status registers are usually accessed for debugging purposes.

Workarounds:

Enable the relevant M4IF masters or slaves clocks for the status read.

Proposed Solution:

No fix scheduled

ENGcm09044 NFC: Auto_erase/auto_program does not latch status correctly

Description:

When performing auto_erase/auto_program/copy_back operation, NFC is supposed to detect the deassertion of RB_B signal, and then send a status-read command in order to capture the status of the last operation, and thus reducing software overhead. In ~50% of the cases, the NFC does not send this status-read command.

Projected Impact:

Status is not correct.

Workarounds:

The software must launch a status-read command at the end of auto_erase, auto_prog or copy_back operations.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Workaround implemented in Linux BSP release 09.12.00.

WinCE BSP Status:

Workaround implemented in WinCE BSP release ER9.

ENGcm09619 NFC: 8-Sym ECC mode does not work with 512 byte page x16 bus NAND Flash

Description:

When ECC_MODE = 1 the 8-sym error detection and correction does not work with 512 byte main area ×16 NAND Flash.

Projected Impact:

The 8-sym ECC mode does not work with ×16 bus width NAND. This mode works with ×8 bus width NAND. This restricts 8-sym ECC operation to 8-bit NAND devices.

Workarounds:

None

Proposed Solution:

No fix scheduled. The reference manual is updated accordingly.

Linux BSP Status:

No software workaround required.

WinCE BSP Status:

No plan to support as there is no NAND Flash on EVK.

ENGcm09575 NFC: Copy back function destination address restriction

Description:

The copy back feature of the NFC module does not work as expected. When trying to copy a page from source address to destination address, the destination address is always the successive page of the source address.

Projected Impact:

The copy back cannot be done to an address that is not successive to the source address.

Workarounds:

There are two options:

- Perform the copy back through Atomic operations
- Instead of copy back, perform a read operation followed by a write operation

Both options affect the overall performance.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Not required because BSP does not use this feature.

WinCE BSP Status:

No plan to support as there is no NAND Flash on EVK.

ENGcm09135 NFC: Block write-protect does not support lock-tight

Description:

NFC offers a block-write-protect feature in which only a range of pre-defined blocks can be modified. This range of blocks can be set to UNLOCK (blocks that can be modified), LOCK (blocks that cannot be modified) or LOCK_TIGHT (blocks that cannot be modified and the range cannot be changed).

Though switching to LOCK_TIGHT mode, the range of blocks can still be modified.

Projected Impact:

Lock cannot be trusted to prevent data from being overwritten. As the software does not use this feature (both WinCE and LINUX), there is no impact.

Workarounds:

As there is no difference between LOCK and LOCK_TIGHT modes, the software should not use LOCK_TIGHT mode.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not required because BSP does not use this feature.

WinCE BSP Status:

No plan to support as there is no NAND Flash on EVK.

ENGcm06982 NFC: Block write-protect does not work in automatic operations

Description:

NFC offers a block-write-protect mechanism. Only a configurable range of NAND blocks can be modified. Any erase/program operations on the blocks outside this range should be blocked by NFC. This mechanism does not work in automatic program and in automatic erase.

Projected Impact:

As there is no workaround for this bug, the write protect mechanism cannot be used.

Workarounds:

It should be handled in the software, as it is done now for the Windows and Linux drivers.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not required because BSP does not use this feature.

WinCE BSP Status:

No plan to support as there is no NAND Flash on EVK.

ENGcm09400 NFC: Copy-back does not work properly when addr_op = 01

Description:

When working with addr_op = 01, and trying to perform a copy-back operation, the NFC ignores the destination address configured, and copies the page to “source address” +1. Additionally, the following automatic operation is carried out from address_register1 instead of address_register0.

Projected Impact:

Automatic operation is carried out from address_register1 instead of address_register0.

Workaround:

If the system requires to work in addr_op = 01, switch to addr_op = 11, before the copy-back operation, and switch back after the copy-back operation is complete.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not required because BSP does not use this feature.

WinCE BSP Status:

No plan to support as there is no NAND Flash on EVK.

ENGcm09186 NFC: Software reset does not work properly under certain conditions

Description:

The software reset (setting the NFC_RST bit in NFC and then sending reset command 0xFF to NAND Flash) does not work correctly under the following conditions:

- Reset between the read operations
- With atomic program operation, the RESET command is not being issued
- Auto program operation—If reset occurs after writing data to the NFC and before the write confirm command, the CACK bit is not set after setting CREQ

Projected Impact:

The software reset does not work consistently, but the failure conditions are not common in real world applications.

Workarounds:

Do not to apply software reset for the above conditions.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not required because BSP does not use this feature.

WinCE BSP Status:

Not required. BSP does not have such case.

ENGcm08208 NFC: Status read does not occur at the end of the program, with RBB_MODE = 1

Description:

After automatic Program operation the NFC is expected to perform a status read and store the status in STATUS_SUM register. With RBB_MODE = 1 (that is, NFC waiting using BSY_B signal), the status read does not occur at the end of the Program/CopyBack0 and CopyBack1 operations.

Projected Impact:

The status is not correct.

Workarounds:

Workaround for this erratum:

- In the Automatic mode, use RBB_MODE = 0
- If RBB_MODE = 1 is to be used, then after the Automatic program, Automatic CopyBack0 and Automatic CopyBack1 operations, Status read should be done explicitly.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Workaround implemented in Linux BSP release 09.12.00.

WinCE BSP Status:

Workaround implemented in WinCE BSP release ER9.

ENGcm09394 NFC: Unlock registers are reset during warm reset

Description:

When a warm reset is issued, all the NFC registers should not be reset. This is done to enable a quick return to operation after the warm reset occurs. But when a warm reset is asserted to the NFC, the unlock registers are also reset. Therefore, the commands are not executed after warm reset.

Projected Impact:

Commands are not executed after warm reset. Workaround has minor performance impact.

Workarounds:

After a warm reset, the user should unlock the unlock registers of the NFC. This allows the NFC to start working as expected. Unlocking the registers cause a minor delay until the NFC is ready to work.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Workaround implemented in Linux BSP release 09.12.00.

WinCE BSP Status:

Not required. Eboot sets the registers properly.

ENGcm09970 NFC: NFC does not work properly when RBB_MODE = 0 (read status)

Description:

The NFC should issue a read status command and move to the next operation only when the status is ready (status = 0xE0) but the NFC uses the ready/busy signal instead. For example, in automatic program operation, the NFC should perform a read status loop until the status is ready and only then assert the interrupt. But the NFC asserts the interrupt when the ready/busy signal returns to 1 (ready) while the read status is still busy.

Projected Impact:

When RBB_MODE = 0, the user can not tell if the device is busy or not. Automatic operations fail. Also, the system can hang.

Workarounds:

Connect the ready/busy signal from the Flash to the i.MX51 and program RBB_MODE = 1. However, using this workaround restricts the number of CS supported to 4 instead of 8.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Not required because using rbb_mode = 1.

WinCE BSP Status:

Software workaround implemented.

ENGcm09980 NFC: Misses read data when working in Symmetric mode with clock ratio 1:2, and using a 16-bit Flash bus width

Description:

When working in symmetric mode with a clock ratio 1:2, and using a 16-bit Flash bus width, the NFC reads data and organizes it using a shift in the internal RAM in such a way that the last 16 bits of the data block being transferred are not written to the memory.

Projected Impact:

Wrong data may be read.

Workarounds:

Avoid combining the following parameters/conditions:

- 16-bit flash bus width
- Symmetric mode
- 1:2 clock-ratio

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not required because BSP does not have such case.

WinCE BSP Status:

Not required because the 16-bit NAND is not supported with current WinCE BSP.

ENGcm10033 NFC: Does not Issue DMA read request using 1/2 Kbyte Page in SDMA mode

Description:

The NFC does not issue DMA read request (dma_rd_req) when the page size is 1/2 Kbyte (PS = 00) and in SDMA mode (NO_SDMA = 0). It works normally when using other page sizes.

Projected Impact:

Data can not be read by the SDMA

Workarounds:

Do not use the combination of parameters/conditions described above.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Not required because BSP does not use this feature.

WinCE BSP Status:

Not required because BSP does not use this feature.

ENGcm10036 NFC: Cannot reach entire address space when addr_op = 1 or 3

Description:

When the NFC is configured to addr_op = 1 or 3

- If num_of_devices = 0–1, then the LSB of the page/blocks address section of the addr_group bits is not used for address generation.
- If num_of_devices = 2–3, then the 2 LSB's of the page/blocks address section of the addr_group bits is not used for address generation.
- If num_of_devices = 4–7, then the 3 LSB's of the page/blocks address section of the addr_group bits is not used for address generation.

Projected Impact:

Working in these addr_op modes has some limitations on the size of the devices (depending on the number of devices). Such large devices do not exist at this time. This can be an issue when using larger devices that may become available in future.

NOTE

addr_op = 0 works as designed with any combination.

Workarounds:

Do not use the combination of parameters/modes described above.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not required because such devices are not available yet.

WinCE BSP Status:

Not required because such devices are not available yet.

ENGcm10135 NFC: Incorrect ECC Error Detection when NFC_RST bit is set

Description:

When the NFC_CONFIGURATION1[NFC_RST] bit is asserted, the state machines of the NFC is supposed to be initialized, but the state machines of the ECC encoder/decoder do not return to IDLE mode. Therefore the operation that is executed right after NFC_RST operation incorrectly indicates an uncorrectable ECC error.

Projected Impact:

The program/read operations execute correctly, but the NFC indicates uncorrectable ECC errors in the ECC_STATUS_RESULT register.

Workarounds:

Avoid using the NFC_RST bit of the NFC_CONFIGURATION1 register.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Workaround implemented in Linux BSP release 09.12.00.

WinCE BSP Status:

Not required. BSP does not use this feature.

ENGcm10150 NFC: Overrides read data in Asymmetric mode using a clock ratio of 1:2, and a 16-bit Flash bus width

Description:

When working in asymmetric mode, using a clock ratio of (flash_clock:axi_clock) 1:2 and a Flash bus width of 16-bits, the NFC reads data from the NAND Flash it sometimes repeats the same address of the internal RAM and therefore writes to the same addresses, overwriting data instead of incrementing the write pointer.

Projected Impact:

Corrupted/invalid read data when operating with the combination of conditions described above.

Workarounds:

Avoid operating in this combination of conditions

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Not required because BSP does not have such case.

WinCE BSP Status:

Not required because we do not support 16-bit NAND with current WinCE BSP.

ENGcm10158 NFC: Reads only from the last device using addr_op = 2 without read confirmation

Description:

The bug occurs when the NFC is configured as follows:

Interleaved mode

Addr_op = 2

More than one device

Number of iterations > 1

During automatic read operations the NFC only reads the data from the last device and not from all the devices. The reason for that is that in addr_op = 2 the NFC issues the address phase to each of the devices separately because the address is taken from different register.

Projected Impact:

Incorrect read operations when using the combination of conditions described above.

Workarounds:

Avoid using this combination of conditions.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Not required. BSP does not use this feature.

WinCE BSP Status:

Not required. BSP does not use this feature.

ENGcm10176 NFC: Does not work when number of iterations are greater than number of devices

Description:

When the number of iterations are greater than the number of devices then the NFC is supposed to issue accesses to each device in a loop until the proper number of iterations is reached. The problem is that when the NFC completes the first loop it does not check whether the device is busy and issues accesses to the devices when they are in busy state.

Projected Impact:

Incorrect read operations under the combination of conditions mentioned.

Workarounds:

Configure the number of iterations to less than or equal to the number of devices used in interleaved mode, so that the NFC can issue one page per connected device.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Not required. BSP does not use this feature.

WinCE BSP Status:

Not required. BSP does not use this feature.

ENGcm10205 NFC Doesn't Protect the Last 10 Spare Bytes of the Last Section in 4KB+218

Description:

The NFC-ECC engine does not protect the last 10 spare bytes of the last section in page size of 4 Kbytes + 218 bytes.

In the case where the page size is 4 Kbytes + 218 bytes, the first 7 sections are of 512 bytes and have spare of 26 bytes, and the last section has spare of 36 bytes.

In the case where 4/8 bits ECC protection, the arrangement of the last section of the memory is as following: 18 bytes/12 bytes then 8 bytes/14 bytes of ECC and 10 bytes of user-specific. Those 10 bytes of the last section are not ECC protected.

Projected Impact:

The last 10 spare bytes of the last section in 4 Kbytes + 218 bytes are not ECC protected.

Workarounds:

Do not use these bytes. The Freescale WinCE and Linux BSPs (board support packages) do not use these bytes.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Not required. BSP does not use this feature.

WinCE BSP Status:

Not required. BSP does not use this feature.

ENGcm10245 NFC in RBB_MODE = 1 and atomic operation monitors only rb_b0 instead of the rb_b# of the selected device

Description:

If RBB_MODE = 1 and atomic operation, the NFC monitors only the rb_b signal of device0 (rb_b0) instead of the rb_b# signal that corresponds to the selected device. Thus, when the NFC executes atomic operation with a device other than device0, it monitors the incorrect ready/busy signal.

Projected Impact:

Can not work in atomic operation with a device other than device0.

Workarounds:

Connect on board to rb_b0 port a wired-OR of all the ready/busy signals of all the devices.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

No software workaround is required.

WinCE BSP Status:

Not required. BSP does not use this feature.

ENGcm10356 NFC: ECC mechanism may fail to report uncorrectable error situation

Description:

The NFC error correction mechanism is based on BCH error correction codes. The BCH error correction code allows correction up to a particular number of errors (T = 4 or 8 depending on the configuration), and detection in case of higher number of error bits. In a situation where the number of errors (NOBER) is larger than the number of errors the NFC can correct (T), the NFC should report on uncorrectable error.

Due to a theoretical limitation, the BCH code can fail in detecting the error when the number of actual error bits is much larger than the number of correctable bits. The failure depends on the location of the errors (related to Hamming distance), and its probability is very small. The detection failure probability can be calculated using the formula as follows:

$$P_e = \frac{1}{T! \cdot 2^T}$$

where T is number of correctable errors.

- For 4-bit ECC, the calculated probability for a decoder error is 1 : 384
- For 8-bit ECC, the calculated probability for a decoder error is 1 : 10,321,920

In case of failure, the NFC reports only on T or less errors (NOBER=T) and tries to correct them. As a result the data is damaged and the NFC does not report on it.

Projected Impact:

There is a small probability for failure in detection of the errors by the ECC mechanism due to theoretical limitation of the ECC code.

Workarounds:

To reduce the failure significantly, treat the case where NFC reports NOBER = T as uncorrectable error. This implies that if the number of errors is equal to T, the software must invalidate the block.

It is a common practice for the software to mark the block as bad in advance when the number of reported errors approaches T.

Proposed Solution:

No fix scheduled

Linux BSP Status:

No complete software workaround.

WinCE BSP Status:

Software workaround implemented. It can alleviate the issue, but cannot completely solve it. Such use case (4-bit ECC) is rare under current market environment.

ENGcm10344 NFC fails to transfer data in read burst accesses when rready is deasserted

Description:

When the host issues a read burst access to the NFC and during the data phase the host deasserts the rready signal (indication from the host that he is ready to accept the read data), then the NFC fails to drive the next datums after the deassertion and also the assertion of the rlast (indication from the slave for end of read data transfer) does not occur on time. Thus, the current read access is incorrect and causes a severe AXI protocol violation.

Projected Impact:

The NFC is not able to handle more than 4 data word read transactions at a given time.

A restriction is imposed on read access to NAND Flash that require special attention in software and may cause some performance degradation for NAND Flash usage.

Workarounds:

Due to this coherency issue, the system should prevent the NFC buffers from being cached. This can be prevented by limiting the outstanding read accesses to the NFC buffers to four data words for all combined accesses at any time.

In case of access from single AHB bus masters, which can only submit one outstanding access at a time (such as the SDMA), the burst size must not exceed 4.

In case of AXI bus masters, such as the ARM core, the software must control the data read accesses to be less then 4 data words at a time. Note that there is also a need to make sure that both Neon and the Core do not read together more then 4 data words at a time from the NFC buffers.

In case of multiple masters accessing the NFC, the limitation remains the same. For example, if both ARM and SDMA need to access the NFC buffers, the SDMA could limit its burst reads to 2 and ARM must not request more the 2 data words at a time.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Not required. BSP does not use this feature.

WinCE BSP Status:

Not required. BSP does not use this feature.

ENGcm10676 NFC fails to perform ECC encoding in interleave mode if FMP is larger than PS

Description:

This issue is related to SDMA access to NFC in automatic interleave mode.

The faulty NFC operation occurs in the following conditions:

— When NUM_OF_ITERATION is greater than zero, and NO_SDMA = 0 for automatic interleave mode access from SDMA.

— FMP (Fifo Mode Protect) is set to a value smaller than the Page Size (PS). The FMP determines the interval between the write and read pointers to memory buffer.

— FMP threshold is reached before the host drives more data to the RAM

In this case, the NFC fetches data from the RAM and drives it to the flash line. This causes the difference between the write and read pointers to violate the FMP threshold and as a result, the NFC stops driving data to the flash line. In such a situation, the last data that is driven to the flash line does not pass through the ECC (Error Correction) encoding engine. As a result, the ECC engine fails to encode the whole 0.5 Kbyte data and reports wrongly on uncorrectable errors.

Projected Impact:

The above described case with the FMP (FIFO Mode Protection) feature activated and with NUM_OF_ITERATIONS greater than zero (that is, for SDMA) can result in wrong function of NFC.

The issue impacts only SDMA access to NFC in interleave mode with page size equal to 4 Kbyte. In this case, the automatic interleave mode can not be supported and has to be managed manually.

Other cases can be resolved using the settings described in the workaround section. The implication of proposed setting is some additional latency in write operation only. The latency is derived from the time it takes to fill the internal RAM with the page data.

Workarounds:

The 0.5 Kbyte and 2 Kbyte use cases can be supported under the specified restrictions:

- In case of the write operation with page size of 0.5 Kbyte (PS = 00), it is allowed to configure the NFC to automatic interleaved mode (that is, with NO_SDMA=0 and setting NUM_OF_ITERATIONS to greater than zero). In this case, the FMP field (that defines a safety buffer inside the NFC internal RAM between the AXI host address and the NAND address) must be set to 0.5 Kbyte (set FMP = 4).
  In read operation, the FMP can be set to any value that is less than or equal to 0.5 Kbyte, (that is, setting FMP to either 0, 1, 2, 3 or 4).

- In case of write operation with page size of 2 Kbyte (PS = 01), it is allowed to configure the NFC to automatic interleaved mode (that is, with NO_SDMA=0 and setting NUM_OF_ITERATIONS to greater than zero). Here, the FMP field must be set to 2 Kbyte (that is, setting FMP = 6).
  In the read operation, any value of FMP which is less than or equal to 2KB can be configured (that is, setting FMP to either 0, 1, 2, 3, 4, 5, or 6).

In case the page size is 4 Kbyte (PS = 10), it is forbidden to configure the NFC to automatic interleaved mode for both write and read operations. NO_SDMA should be set to 1 and the field NUM_OF_ITERATIONS should be set to 0. Instead, the interleave operation should be implemented manually as also proposed in erratum ENGcm10967.

- Set the number of pages to read/write to 1 (set NUM_OF_ITERATIONS = 0).
- Set the number of devices connected (that is, setting the field NUM_OF_DEVICES).
- Use addressing option 1 (set ADR_OP = 1, part of the address is used for chip select) and configure the target address and the target chip select at address group0 (that is, setting the fields NAND_ADD0 and NAND_ADD8).
- An interrupt is issued after each single page read/write transfer. Upon receiving the interrupt, reconfigure RBA (Ram Buffer Address) and the corresponding bits of ADDR_ADD0 to repeat the operation for the next device.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Not required. BSP does not use this feature.

WinCE BSP Status:

Not required. BSP does not use this feature.

ENGcm10967 NFC does not function properly for 4 Kbyte Page Size in interleave mode

Description:

This issue is related to SDMA access to NFC in automatic interleave mode (NO_SDMA = 0).

NFC buffer mechanism does not operate properly under the following set of conditions, potentially feeding wrong data from the internal NFC RAM to FLASH data line.

The fault occurs under the following set of conditions:

- NFC is set to operate in interleave mode (NO_SDMA = 0) and NUM_OF_ITERATION > 0 for SDMA access.
- Page Size is set to 4 Kbyte (PS = 10)
- FIFO Mode Protect is set to 4 Kbyte (FMP = 7)

Projected Impact:

The above described case can result in wrong functioning of NFC and should be avoided. The issue is relevant only for SDMA access to NFC in interleave mode with page size equal to 4 Kbyte.

Workarounds:

For Page Size equal to 4 Kbyte (PS = 10) the NUM_OF_ITERATION must be set to 0 and NO_SDMA must be set to 1, disabling the interleave mode.

An interleave mode can be implemented manually by configuring the appropriate NUM_OF_DEVICES while setting ADD_OP = 1 (Addressing Option 1, part of the address is used for chip select). In this case, an interrupt is issued after each single page read/write transfer. Upon receiving the interrupt, reconfigure RBA (Ram Buffer Address) and the corresponding bits of ADDR_ADD0 to repeat the operation for the next device.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Not required. BSP does not use this feature.

WinCE BSP Status:

Not required. BSP does not use this feature.

ENGcm11043 NFC issues premature DMA read request in case of TOO configuration in interleave mode

Description:

NFC issues premature DMA read request for each page in case of the TOO configuration in interleaved mode.

The TOO (Two On One) configuration is a case where two 8-bit NAND devices are connected to the same CS line to form a 16-bit data width (one device on the lower byte of data bus and second device on the upper byte).

An interleaved mode is configured by setting the number of devices and NUM_OF_ITERATION to a value greater than zero, and NO_SDMA = 0 for SDMA access.

When the NFC is configured for the above two modes, in addition to the DMA request at the end of the buffer, it wrongly issues additional DMA read request for each page in the middle of the buffer.

Due to this, the SDMA starts fetching data from the RAM buffer before the complete data is available.

Projected Impact:

Erroneous behavior under the above mentioned conditions.

Workarounds:

Avoid using interleaved mode in TOO configuration.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Not required. BSP does not use this feature.

WinCE BSP Status:

Not required. BSP does not use this feature.

ENGcm11060 NFC does not perform automatic status read operation (AUTO_STAT) according to ACTIVE_CS

Description:

When the NFC is configured to perform an automatic status read operation (that is, setting AUTO_STAT), it should execute that operation on the device ID that is configured in the ACTIVE_CS. Due to the reported issue the NFC executes the operation on another device rather than the one specified in ACTIVE_CS.

Projected Impact:

Incorrect status return for automatic status read operation.

Workarounds:

Perform atomic status read operation (that is, send command 70 followed by single toggle of RE).

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Workaround implemented in Linux BSP release 09.12.00.

WinCE BSP Status:

Software workaround implemented.

ENGcm11004 NFC can miss the sampling of the ready/busy signal ($\overline{R/B}$) when RBB_MODE = 1

Description:

The NFC may not properly sample the ready/busy ($\overline{R/B}$) signal under the following conditions:

1. RBB_MODE = 1 (Ready-Busy mode 1 - NFC monitors ready-busy status by checking NANDF_RBx ($\overline{R/B}$) signals)
2. ADD_OP = 01 (Addressing Option 01 - NFC uses only address_group0 with single NAND device
3. enfc_clk period is less than $t_{wb}/3$ (t_{wb} is the period from $\overline{WE}$ write enable signal HIGH to $\overline{R/B}$ ready/busy signal assertion)

If RBB_MODE = 1 and ADD_OP = 01, the NFC may miss the sampling of the ready/busy signal ($\overline{R/B}$) if the enfc_clk is too fast. According to the NAND flash protocol, the NAND device should enter the busy mode and assert the $\overline{R/B}$ signal (driving $\overline{R/B}$ signal to zero) after the maximum time period of t_{wb} after the deassertion of the $\overline{WE}$ signal (driving write enable to one).

The NFC samples the $\overline{R/B}$ signal after a fixed time of 3 enfc_clk cycles after the deassertion of $\overline{WE}$. Typically, the t_{wb} period is 100 ns. When the enfc_clk is set to less than 33 ns period, the NFC may miss sampling the $\overline{R/B}$.

Projected Impact:

Violation of NAND interface protocol can result in data corruption. The implementation of the proposed workaround may have insignificant performance impact.

Workarounds:

The recommended workaround is to set RBB_MODE = 0. This also frees up the NANDF_RBx pads for other usages. The NFC monitors the ready-busy status by performing a status-read command.

Another option is to work with ADD_OP other than 01, allowing to work automatically with a single device other than device0.

See, NAND_FLASH_CONFIG Register Field Descriptions table in the *i.MX51 Multimedia Applications Processor Reference Manual* (*MCIMX51)* for detailed modes description.

If the above two options are not feasible, the enfc_clk frequency can be reduced such that clock period is larger than $t_{wb}/3$. This results in small performance degradation as the supported ONFI1.0 NAND can run at up to 40 MHz.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Workaround implemented in Linux BSP release 09.12.00.

ENGcm11036 SDMA multi-page read from the NFC, when the WEIM is operating, can result in data corruption or EMI hanging

Description:

Data corruption or EMI hanging can occur on SDMA Multi-Page read from the NFC in case of simultaneous read access by other master from WEIM. Use of SDMA is relevant for a case of multiple pages read from the NFC (that is, setting NUM_OF_ITERATION to a value greater than zero and setting NO_SDMA to zero in NFC for automatic interleave mode).

Projected Impact:

SDMA read access from NFC may be corrupted in case other masters perform read from WEIM at same time.

There are no issues in following cases:

- ARM accesses the NFC when WEIM is accessed by any other master in the system
- SDMA accesses the NFC when the WEIM is idle
- NFC is idle and WEIM is accessed by any master in the system

Workarounds:

Avoid the above described situation. Note that, in current WinCE and Linux BSP releases from Freescale, the SDMA Multi-Page mode read from NFC is not activated.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not required because the SDMA Multi-Page mode read from NFC is not activated.

WinCE BSP Status:

Not required. BSP does not use this feature.

ENGcm10640 Grounding nonfunctional UHVIO IO pads power rails can cause malfunction in other UHVIO IO cells

Description:

If the power supply of an unused UHVIO pad power group is pulled down, it can cause malfunction in the other functional UHVIO IO cells. The UHVIO IO cells (that support signal levels above 3 V) are used in interfaces such as SD card, NAND Flash. For example, grounding SD2 power supply rail through a small resistor (say 470 Ω) causes boot issues from the SD1 interface. This is caused due to a particular HVIO IO cell design limitation.

Projected Impact:

Pay attention to this limitation during board design. An unintentional grounding can occur because of nonfunctional or disabled voltage supply device connected to the UHVIO based interface VCC line.

Workarounds:

Take measures to avoid such cases. Leaving the unused interface supply open, does not cause any issue. The issue occurs only when a supply is grounded through a small resistor. The best design practice is to apply power to all the supply rails.

Proposed Solution:

No fix scheduled

ENGcm11041 Dependency between VCC, VREG and NVCC_HSx supplies

Description:

Pulling down the power supply of any of NVCC_HSx supplies (from the group of NVCC_HS6, NVCC_HS10, NVCC_HS4_1, NVCC_HS4_2 supplies) or VREG supply results in high leakage current from VCC through the supply that is pulled down. Leakage of order of a few hundreds of mA is observed in case NVCC_HSx or VREG supplies are shortened to ground.

If the supplies in NVCC_HSx supplies and VREG are left unconnected or in high impedance state, the leakage would not be observed.

This is caused due to a particular MIPI shielding IO cell design limitation. The leakage current in this case is caused by the ESD protection diode inside the MIPI interface shielding pads.

Projected Impact:

Need to consider this limitation during the board design.

This is mostly relevant to suspend mode, in which the supply regulator of those pins may be turned off to save power.

Typically, when a regulator is off, there would be high impedance on the associated supply pin(s), and leakage depends on the resistive path to ground that exists on the board. If there is a high resistive path, almost no leakage would be observed.

This is also relevant to power-up sequence, because these supplies would be powered-up after VCC supply and depends on the resistive path, there could be some leakage till these supplies are stable.

Workarounds:

Take measures to avoid such cases. Note that leaving the unused interface supply open, does not cause any issue. Having this situation for short periods (such as during power up sequence) should not cause any damage.

Alternatively, if high impedance with a low resistive path to ground is projected on the supply, then leakage is negligible.

Proposed Solution:

No fix scheduled.

ENGcm10656 Serial boot will fail if HAB_TYPE is PRODUCTION

Description:

If the HAB_TYPE e-fuse is blown as *Production*, then accesses to peripherals from DCD and SDP commands is disabled and DDR controller cannot be configured, so DDR cannot be used.

The ROM does not check if the address is valid if HAB_TYPE is not *Production* for write header and write file SDP commands. So there is no issue in this case.

However, if HAB_TYPE is Production, the ROM allows SDP commands only if the target address lies within the DDR, IRAM, GMEM, NAND flash buffer or WEIM memory regions. SDP Write commands to peripherals such as the memory and clock controllers are denied.

Projected Impact:

Can not execute serial boot if HAB_TYPE is set to *Production*.

Workarounds:

Use internal RAM boot image. In this case DDR configuration is not required.

Proposed Solution:

No fix scheduled.

ENGcm11189 ROM (Boot)/NFC: NAND Flash Boot fails when one of the unused NANDF_RBx signals are held at low

Description:

The Boot procedure fails when the boot code tries to initialize and access the NAND Flash, in case one of the unused R/Bx signals is grounded. In a system with a single NANDF Flash device, NANDF_RB1, NANDF_RB2, NANDF_RB3 are not in use, as the Boot code uses only CS0. These signals can be alternatively used for other functions (such as GPIO input). Due to reported limitation, the boot in NAND Flash mode fails, if one of the unused NANDF_RBx signals is held at zero during the boot.

Projected Impact:

This limitation impacts the possible usage of NANDF_RB1, NANDF_RB2 and NANDF_RB3 in other alternate IOMUX modes as input, with zero level at boot.

Workarounds:

The only workaround is to avoid setting the unused R/Bx signals to zero during boot, in case of NAND Flash boot mode.

Proposed Solution:

No fix scheduled.

ENGcm11353 ROM (Boot): OneNAND boot fails for 4 Kbyte devices

Description:

OneNAND boot driver calculates page and sector numbers incorrectly for 4 Kbyte devices.

Projected Impact:

4 Kbyte OneNAND devices boot does not work for image sizes above 1 Kbyte.

2 Kbyte OneNAND devices boot works properly.

Workarounds:

For 4 Kbyte devices, it is possible to use only the top half of pages. In this case, ROM reads the boot image properly, but the bottom half of pages are wasted.

Proposed Solution:

No fix scheduled for the i.MX51.

ENGcm05863 RTIC: Software reset during one time hash mode corrupts RTIC state machine

Description:

The sw_reset bit in the RTIC spec says that it should not be modified in run-time check mode. A software reset was done during one time hash mode and immediately after that RTIC was enabled. The RTIC state machine is corrupted in such a scenario. The assertion of software reset when hash_once_command is set in rtic_control, leads to the assertion of hash_once and hence of zero_byte_length in rtic_dma_request. This in turn sets the mem_part_1_done bit and even when RTIC is enabled this remains set, leading to corruption of the state machine.

Projected Impact:

RTIC hangs.

Workarounds:

Be aware of limitation. Do not reset RTIC during one time hash mode.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Not required. BSP does not use this feature.

WinCE BSP Status:

Not required. BSP does not use this feature.

ENGcm10974 RTICv3 memory region unlock feature can cause the RTICv3 to hang

Description:

The RTICv3 feature that allows the TrustZone software to disable the run-time check of the selected memory regions (region unlock), can cause the RTICv3 to hang and stop the run time integrity check of other regions.

Projected Impact:

The RTICv3 feature that allows the TrustZone software to disable run-time check of the selected memory regions cannot be utilized.

Workarounds:

None. Avoid using the memory region unlock feature in the TrustZone code.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not required. BSP does not use this feature.

WinCE BSP Status:

Not required. BSP does not use this feature.

ENGcm10334 SAHARA/CCM: Frequency ratio restriction for AHB and IP buses in SAHARA

Description:

SAHARA does not work properly if AHB:IP buses clock ratio is 1:1. It works fine with AHB:IP buses clock ratio 2:1.

Projected Impact:

Adds limitation on system configuration for the clock ratio of 1:1 between AHB and IP buses.

Workarounds:

Avoid using a ratio of 1:1 between AHB and IP buses clock frequencies. Other software workarounds are considered, but not yet confirmed.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not required. Be aware of the limitation. Avoid using a ratio of 1:1 between the AHB and IP buses clock frequencies.

WinCE BSP Status:

SAHARA is not used in the current BSP release.

ENGcm10272 SRTC: Possible status loss when peripheral power resumes if SRTC in fail state

Description:

There is the possibility of a loss of SRTC failure status upon peripheral supply voltage (the HP section) power up.

The SRTC incorporates a System State Retention Register (SSRR) that stores system parameters during system shutdown modes. This register and all SRTC counters are powered by dedicated supply rail NVCC_SRTC_POW that remains active, allowing all other supply rails to shut down. The SSRR also stores the system security state. If a security violation occurs, the SSRR marks the event as a security violation indication.

The SRTC is split to two supplies: LP and HP. The LP section is powered by NVCC_SRTC_POW and always remains active, whereas the HP section is powered by VCC rail and can be shut down. As long as the SRTC LP section remains active with its separate power supply and clock inputs, it continues to detect and record security violations when the HP section and the rest of the chip are powered down. When power is restored to the HP section, any security violation detected during power down becomes visible to the rest of the system through a security monitor alarm and the SRTC register interface. In particular, software can examine the LP status register to observe the LP Failure state and status bits indicating which security violation occurred.

In rare cases, an erroneous reset of the LP section has been observed during HP power-up when a security violation was recorded during HP section power down. This causes the LP to move to Initialize state rather than Failure state, and erases all record of the security violation.

Please note that this scenario can occur only if SRTC LP has gone into failure state when the peripheral supply power (HP) is off. The unintended SRTC reset does not occur while the SRTC LP remains in the Valid state.

Projected Impact:

This issue is relevant for systems supporting DRM (digital rights management) and other applications requiring a secure clock or monotonic counter. If the SRTC failure status is lost as above, then a system which has been tampered while the main power was off may appear indistinguishable from a system which has a SRTC that is not violated. If software is then permitted to provision the LP time and monotonic counters insecurely, this could mislead the application relying on the SRTC.

Note that this issue does not permit the SRTC LP clock and monotonic counters to be modified directly without leaving the Valid state. In order to complete the attack, there is a need to execute malicious or insecure software that tries to reconfigure the SRTC after it is reset. The system can be protected against such software by code signing and authentication methods.

Workarounds:

None. However, this issue can be mitigated by ensuring that the LP time and monotonic counters are securely reprovisioned. One approach is to authenticate all provisioning software using HAB (high assurance boot) and to lock the LP time and monotonic counters against further changes before allowing unauthenticated software to execute.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Not required. The driver should ensure it is moving to valid state.

ENGcm06571 SSI: Data Tx starts from FIFO1 in case Rx is enabled before Tx in sync mode

Description:

In Network and Synchronous mode of operation, if Rx is enabled and after one or more frames passes Tx is enabled, the first data is transmitted from FIFO1 instead of FIFO0.

Projected Impact:

Wrong data is transmitted.

Workarounds:

Enable Tx and Rx in same frame when in Network and Synchronous mode of operation

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Workaround implemented in Linux BSP release 09.12.00.

WinCE BSP Status:

Not required. Network Synchronous mode is not used.

ENGcm09220 SSI: If TX_EN bit toggled 5 clk cycles before FS, the data transmission not correct

Description:

If Tx is enabled or disabled within a 5 clock cycle window before Frame Sync. Data transmission may not be proper in the following conditions:

- Normal Synchronous mode: With Rx disabled if Tx is enabled 3 clock cycle before non-early FS, glitch of 0.5 clock cycles can be seen on the STXD line.
- Normal Synchronous mode: If Tx is enabled 4 clock cycle before external early word length frame sync with fifo0 disabled two channel disabled, ddr_stxd is high for 1.5 clock cycles.
- Normal Synchronous mode: If Tx is reenabled 2 clock cycles before early FS, ddr_stxd line goes high for 0.5 clk cycles.
- I2s_slave: Tx is disabled in the first time slot and is again reenabled in the 2nd time slot of a different frame wrong data getting transmitted in the next frame. The tx state machine is not running for the frame in which wrong data is transmitted.
- I2s_master: Tx is disabled 2 clk cycle before FS data is repeated in second time slot 6.Network Synchronous mode: Reenabling Tx, 1 clock cycles before External FS, and frame count as 2, results in data in first time slot getting missed.
- Network Synchronous mode: External Early word length FS. If Tx is reenabled 2 clk cycles before FS signal then in the next frame data is transmitted for 0.5 clk cycles.

Projected Impact:

Data transmitted may be lost.

Workarounds:

The following workaround can be implemented to avoid these issues. Enable/disable Tx after occurrence of RFS interrupt:

1. Enable RE(SCR[2])
2. Enable RIE and RFS_EN in SIER
3. Wait for occurrence of RFS
4. Enable/disable TE(SCR[1])

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Not implemented. No failure was observed in Linux BSP.

WinCE BSP Status:

Workaround implemented in WinCE BSP release ER10_SP1.

ENGcm09222 SSI: Normal Async, Tx is disabled 2 clocks before FS, ddr_stxd is indefinitely high

Description:

In Normal Asynchronous mode: If Tx is disabled 2 clk cycle before non early bit length external FS, ddr_stxd signal is indefinitely set high even when no data is transmitted.

Projected Impact:

Wrong data transmission may be recognized.

Workarounds:

Disable Tx after occurrence of TFS interrupt, using the following steps to disable TX_EN bit is:

1. Enable TIE and TFS_EN in SIER.
2. Wait for occurrence of TFS.
3. Disable TE(SCR[1]).

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Not required. Asynchronous mode is not used.

WinCE BSP Status:

Workaround implemented in WinCE BSP release ER10_SP1.

ENGcm06569 SSI: Transmission does not take place in case of bit length early frame sync mode

Description:

When SSI is in normal synchronous mode and frame sync is configured as bit length early frame sync, data is not transmitted.

Projected Impact:

There is no toggling on the TXD data line.

Workarounds:

1. Enable SSI_EN and Tx_EN in the same frame.
2. Do not use early FS in Internal FS mode.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Not implemented. No failure was observed in Linux BSP.

WinCE BSP Status:

Not required. BSP uses word length frame sync.

ENGcm09212 SSI: With TFR bit set, Rx re-enabled, data not accepted according to masking

Description:

I^2S_master mode: If Rx is disabled and again enabled with Tx disabled along with TFR_CLK DIS set, the first data after reenabling is received in the second time slot and hence channel swapping takes place. Network Synchronous mode: If Rx is disabled and again enabled with Tx disabled when TFR_CLK_DIS is set. Data is not accepted into the fifo according to the masking bits.

Projected Impact:

Channel swapping or data lost (according to mode).

Workarounds:

Two different workarounds can be used for this issue:

- Reset SSI every time Rx is enabled.
- Do not use TFR_CLK_DIS feature.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Not required now. BSP does not use I^2S master mode.

WinCE BSP Status:

No workaround is required because WinCE BSP only uses slave mode.

ENGcm09668 SSI: Receive Overrun Error Generated at Wrong Time for Watermark Level

Description:

The receive overrun error is generated when the received FIFO index is 1 more than the watermark level, but the data is received properly in the FIFO. There is no loss of data, the only problem is that the receive overrun interrupt is generated at the wrong time.

Projected Impact:

The receive overrun error is not valid when the watermark mechanism is used.

Workarounds:

If no overruns can be guaranteed on a system level (based on the serial data rate and latency of servicing the SSI), two options are suggested:

a) Give up on the overflow indication. Set the watermark at a sufficiently low level such that the probability for true overflow condition is very low. This option allows smoother performance but lacks any indication that a true overflow has occurred.

b) Set the watermark level to the FIFO size so that the overflow interrupt corresponds with the actual overflow condition. This option provides overflow indication, however there is larger probability for overflow to occur because the watermark indicator is the FIFO size.

Option (a) can be utilized in cases where the application actually has no means to recover from overflow condition in any case (for example in case of audio playback). This option gives smoother performance.

Option (b) can be utilized in cases where there are good means to recover from an overflow state (for example in case the SSI channel is used for control or status indication and there is an option for data retransmit). In this case, the probability of actual overflow is higher, but there is a reliable indication that the overflow has actually occurred.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Not required.

WinCE BSP Status:

Workaround implemented in WinCE BSP.

ENGcm11138 SSI: In AC97, 16-bit mode, received data is shifted by four bit locations

Description:

In AC97, 16-bit mode, the Rx data is received in bits [19:4] of RxFIFO, instead of [15:0] bits.

Projected Impact:

The SDMA script should be updated accordingly to perform the shift to the right location on the fly during data transfer. If the data register is accessed directly by software, it should account for the shifted data and perform shifting to the right location.

Workarounds:

The data should be shifted to the right location by the SDMA script or by the software in case of direct access to the register.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not required because the AC97 mode is not supported.

WinCE BSP Status:

Not required because the AC97 mode is not supported.

ENGcm09134 USB: Core device fails to receive two sequential OUT transactions in short time

Description:

When receiving this sequence OUT - DATA0 - OUT - DATA1, if the inter-packet delay between DATA0 and the 2nd OUT is under 200ns, the device sees the DATA1 packet as a short-packet, even if it is correctly formed. This terminates the transfer from the device point of view, generating an IOC interrupt. DATA0 is correctly received, though.

Projected Impact:

It should only be applicable to ISO OUT as a Device.

Workarounds:

When MX51's USB is operating in device mode, the host should not send two ISO_OUT sequences within less that 200 ns, otherwise i.MX51's USB does not operate correctly.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Not required.

WinCE BSP Status:

Not implemented.

ENGcm09110 USB: Device mode ISO error problem

Description:

The occurrence of an error in device mode sends ISO data in the next μFrame that should have been discarded.

Projected Impact:

Critical for timing-dependent device ISO IN Mult = 3.

Workarounds:

The SW has to set the MULT < 3 to avoid this bug.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Not required. No ISO feature is supported.

WinCE BSP Status:

Not implemented.

ENGcm07300 USB: Erroneous descriptor handling by USBOH module

Description:

The USBOH core uses the erroneous hrdata (when hresp is high) and results in the false USB transfer at the external Interface.

Projected Impact:

False transfer of USB.

Workarounds:

This should not happen in real system because when configured correctly, the USB should not get an error response. If it happens, then Garbage In Garbage Out.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not required

WinCE BSP Status:

Not implemented

ENGcm10636 USB: Issue when USB_CLK_ROOT Clock is enabled while PHCD bit is set

Description:

The following software flow produces this issue.

1. Configure PTS of Host (or OTG) controller to select ULPI (or UTMI) mode (default is serial mode) and turn USB_CLK_ROOT (comes from on-chip PLL) off.
2. Set PORTSC.PHCD bit to put USB controller & PHY into suspend mode -> ULPI (or UTMI) PHY clock stops correctly.
3. Turn USB_CLK_ROOT on and wait a short time.
4. Turn USB_CLK_ROOT off.
5. Clear PHCD bit → PHY clock starts running correctly
6. Delay to allow PHY clock to become active
7. Set PHCD bit again → Controller suspends the PHY and immediately wakes it up, which is incorrect behavior.

This happens because after Step 4, the serial USB engine stays in suspend mode even after the PHCD bit is cleared. This is because the USB_CLK_ROOT is not valid any more.

The root cause is that USB_CLK_ROOT is used for the USB controller's serial engine. For one USB controller, the USB_CLK_ROOT for serial engine can be always turned off in ULPI or UTMI mode. The case mentioned above (turn on, turn off this clock while PHCD is high) is not a real use case.

However on MX51, there are several USB controllers which share the same USB_CLK_ROOT clock. In this case, if one USB works in ULPI or UTMI mode while PHCD is set, another USB may work in serial mode which may turn on, turn off the USB_CLK_ROOT. This is the flow which has issue mentioned above.

Projected Impact:

Improper USB interface operation as described above.

Workarounds:

The situation can be avoided if the USB_CLK_ROOT clock is turned on briefly when any port wakes up from low power suspend. This ensures that the “suspended” state is cleared in the serial PHY interface.

One can optionally check, during wake-up processing, if the USB clock did run during low-power suspend and only turn on USB_CLK_ROOT when needed.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Not implemented. Plan to fix in next release.

WinCE BSP Status:

No workaround is required because WinCE BSP does not manage USB_CLK_ROOT this way.

ENGcm11249 USB: USB-OTG port ULPI interface is not supported

Description:

The USB-OTG port ULPI mode is provided for debugging purposes and is not supported for regular operation. There is no commitment for proper timing functionality on this interface.

This erratum is related to USB OTG port only, for which internal USB-PHY is provided.

Projected Impact:

The USB-OTG port does not support ULPI interface.

The erratum does not relate to the functionality of ULPI mode on other USB-Host ports or to the USB-OTG internal PHY interface.

Workarounds:

Utilize the USB-OTG PHY interface instead or use other USB-Host ports for ULPI interface.

Proposed Solution:

No fix scheduled. Specification documentation updated accordingly.

ENGcm11408 USB: High Speed Transceiverless Logic interface (HS-TLL) and Full Speed Transceiverless Logic interface (FS-TLL) USB interfaces are not supported

Description:

HS-TLL and FS-TLL are not supported. These features will be removed from the specification.

Projected Impact:

Removed support for the option of on board USB connection without transceiver.

Workarounds:

None. Need to add transceiver for on board connection.

Proposed Solution:

No fix scheduled. Feature is removed from the reference manual.

ENGcm09125 VPU: VC-1 AP bug

Description:

The VC-1 intensity bug occurred during motion compensation interpolation process. This bug is that the intensity register is set for wrong intensity scale/shift values in special cases.This bug induces the image distortion because VPU has one intensity information about motion vector in the intensity process. In VC-1 standard, this codec should have all intensity information by line. This bug affects the VC-1 stream which have both field interlace mode and frame mode. There is no impact on the normal decoding stream of VPU except the VC-1 Intensity field mode stream.

The VC-1 AP bug includes three sub bugs:

- Field picture boundary distortion:
  - Appear condition: Current picture is a field, reference picture is a frame, reference vector is set for the outer of P/B Frame, intensity is enabled, and scale/shift intensity value for top and bottom are different to each other.
  - Description: If the reference vector is set for the outer of reference picture, the intensity scale/shift value should be changed in VC-1 Standard. VPU has the intensity information which is the “intensity bottom” without change. But, In VC-1 standard, the intensity information changed from “-1:intensity bottom” to “0:intensity Top”.
- Field picture reference to the same frame field picture:
  - Appear condition: Current picture is a field, reference picture is in the same frame, scale/shift intensity value for top and bottom are different to each other.
  - Description: When current field picture is referencing a field picture in same frame, current version of MC is always referencing parameters for forward intensity compensation which cause distortion on the any field position.
- Frame picture reference to previous field picture:
  - Appear condition: Current picture is a frame, reference picture is a field, intensity is enabled, and scale/shift intensity value for top and bottom are different to each other.
  - Description: In VC-1 standard, the intensity value must be changed by y-position on the two field reference picture. But VPU VC-1 intensity value always indicate the “top Field Intensity value” which cause distortion on the line of the picture.

Projected Impact:

There are three different cases for the projected impact:

- Case 1

  This bug only affects VC-1 stream which have both field interlace mode and frame mode. That kind of bitstream rarely appears in real application case.
- Case 2

  There is no impact on the normal decoding stream of VPU except the VC-1 Intensity field mode stream.
- Case 3

  Visual quality impact.

Workarounds:

Workaround for case 1:

This case needs RTL fix, no firmware workaround. And it has minor visual impact along the boundary of the picture.

Workaround for case 2:

This case can be fixed by firmware. Firmware sets the parameters for forward intensity compensation with values for forward and current intensity compensation.

Workaround for case 3:

Can not be fixed by firmware, but visual quality can be improved by firmware. RTL update can fully fix the problem.

Proposed Solution:

Case 1 — No fix scheduled.

Cases 2 and 3 — Fixed in last firmware release. For case3, the issue is not completely fixed; however, the visual quality is improved.

Linux BSP Status:

Implemented for case 2 and case 3 in VPU firmware in Linux release 09.12.00.

WinCE BSP Status:

Implemented for case 2 and case 3 in VPU firmware.

ENGcm10253 VPU H.263-P3 decoding Advanced Intra coding (Annex I) bug

Description:

This logic bug is in the DC prediction for H.263-P3 Advanced Intra Coding (Annex I) decoding. A clip function logic design was mistakenly implemented thus causes a wrong clipping result when H.263 Annex I is enabled and the DC prediction value falls in the range of [0, -2048]. This condition most likely happens in a picture having very black and white neighborhood 8x8 blocks.

Projected Impact:

This bug can cause a very noticeable visual impact. However, because H.263-P3 content is not common, so the general impact is small.

There is no impact on the encoder, the effect is on H.263-P3 decoder only. Annex I is not implemented/supported by the H.263-P3 encoder.

Workarounds:

None. No firmware workaround fix feasible.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

No workaround.

WinCE BSP Status:

No workaround.

ENGcm10260 VPU DivX V3.11 Variable-length-decoding (VLC) bug

Description:

This logic bug is for the VLC implementation for DivX Version 3.11. Almost escape and run tables had wrong values, and escape run manipulation had been implemented incorrectly.

Regarding to the level tables, there's no problem in normal stream except the table intra_mb_last_10. Some parts of syntax could be regarded as an error with the table intra_mb_last_10, in a normal, error free, stream.

Regarding to the run tables, some run values are coded wrong. So, the counts of run length is different as maximum one in these cases.

The wrong run-escape manipulation can make difference either as minus one to the counts of run length against luminance values.

In general, there are two kinds of bug. One is wrong error detection due to one omitted column of a table array. The other is difference of run length due to wrong run-escape manipulation and wrong table values. The differences of the counts could be maximum one.

Projected Impact:

This bug should cause a noticeable visual quality degradation (theoretically) for some of the video sequences. But visual quality impact is not significant, and the clip we used to find this bug does not have any visual quality degradation.

This bug occurs only in DivX V3.11, and DivX V4, V5, V6 do not have this bug. Since DivX V5 and V6 content is more popular than that of V3.11, so the overall impact should not be significant.

Workarounds:

None. There is no way for firmware workaround fix because the tables and escape run manipulation are done in hardware logic.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

No workaround.

WinCE BSP Status:

No workaround.

ENGcm10388 VPU: MPEG-1 full-pel Motion Vector Update Failure

Description:

Some MPEG1 bit streams are decoded unsuccessfully.

This hardware bug occurs in the process of updating the integer motion vector when the motion vector is negative. The VPU always assumes “0” in MSB even for a negative motion vector, thus mistakenly producing a positive motion vector.

Projected Impact:

Some MPEG1 bit streams can be decoded unsuccessfully.

A wrong motion vector is produced for a negative integer motion vector. Because negative integer motion vectors in MPEG1 are very common, visual quality degradation is significant due to the wrong motion vectors for motion compensation.

The usage of MPEG1 is rare in new products, so the overall impact is small.

Workarounds:

There is a firmware workaround.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Fixed in firmware workaround.

WinCE BSP Status:

Fixed in firmware workaround.

ENGcm10390 VPU: JPEG decoder does not support different AC/DC Huffman tables for Cb and Cr

Description:

There is a limitation when selecting different Huffman tables for AC/DC coefficients for the two chrominance components, Cb and Cr. The JPEG decoder design cannot handle two different Huffman tables for Cb and Cr. Cb and Cr are assumed to use the same Huffman table, which is true in most cases.

Projected Impact:

The bug occurs when the two chroma components, Cb and Cr, use different Huffman tables for AC/DC coefficients. Therefore, if Cb and Cr use the same Huffman table, this problem does not occur. Normally, JPEG streams use the same Huffman tables for Cb and Cr. It should be rare for JPEG streams to use different Huffman tables for the two chroma components.

There is a large visual quality degradation. However, in reality, the probability of using two different Huffman tables for Cb and Cr is low. As a result, the overall impact for JPEG decoding should not be significant.

Workarounds:

None. There is no firmware workaround fix for this erratum.

Proposed Solution:

No fix scheduled

WinCE BSP Status:

Not implemented

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).

This erratum applies to i.MX512D, i.MX513D and i.MX515D devices.

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.

Work Arounds:

The issue can be mitigated by using a Multiplication Factor Numerator (MFN) software work around 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 work around, and they need to be applied whenever the PLL is shutdown and re-started. The implementation below assumes PLL1 is being used as the source for the CPU and DDR clock.

- Part 1: Work around applied during system initialization (boot code)
  - 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
  - Change DDR clock to be sourced from PLL2
  - Change ARM clock to be sourced from PLL2
  - 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 to 800 MHz by applying the following factor: MFN = 60
  - Request PLL to load new MFN using DP_CONFIG (LDREQ bit)
  - Wait for acknowledge 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
- Part 2: Work around applied to suspend/resume code (kernel)
  - Change DDR clock to be sourced from PLL2
  - Change ARM clock to be sourced from PLL2
  - Update MFN to transition to 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. DDR and ARM are sourced from PLL2
  - Update MFN to transition to 800 MHz by applying the following factor: MFN = 60
  - Request PLL to load new MFN using DP_CONFIG (LDREQ bit)
  - Wait for the 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 via standalone software patch to u-boot and kernel, available on www.freescale.com/imx51tools. Software workaround implemented in BSP releases dated 7/15/2011 or later; refer to the accompanying BSP release notes for details.

ENGcm12364 eSDHC AutoCMD12 and R1b polling problem

Description:

Occurs when a pending command which issues busy is completed. For a command with R1b response, the proper software sequence is to poll the DLA for R1b commands to determine busy state completion. The DLA polling is not working properly for the ESDHC module. This is relevant for all eSDHC ports (eSDHC1-4 ports).

Projected Impact:

DLA bit in PRSSTAT register cannot be polled to wait for busy state completion.

Software Work Around:

Updated block guide to reflect that DLA is not applicable to detect busy state, instead, should poll bit 24 in PRSSTAT register (DLSL[0] bit) to check that wait busy state is over.

Silicon Fix:

No plan to fix.

ENGcm12362 NFC wrong indication of ECC uncorrectable error occurrence after reading the spare area

Description:

In the event that an uncorrectable ECC error occurs while reading Main/Main+Spare from the Nand device, then all spare read operations would fail until the next successful Main/Main+Spare read operation.

Projected Impact:

The NFC wrongly indicates ECC error.

Software Work Around:

Read main or main+spare after first uncorrectable error.

Silicon Fix:

No hardware fix scheduled. This issue will be addressed in the software drivers in the next driver release.

ENGcm12376 eSDCTL: ESDCTLv2 fails to wait the minimal 200uS between DDR clk & clk enable

Description:

The DDR2 JEDEC standard requires the DDR clock (SDCLK) to start toggling at least 200 μS before clock enable (SDCKE) signal rise. In the ESDCTLv2 implementation, it was planned to count 7 CKIL clock periods, calculating to 30.05 μS*7 or 210.35 μS, which is greater than 200 μS and within the JEDEC requirement. However, in practice, SDCKE wait period can be as short as 6 CKIL periods, thus violating the above JEDEC requirement.

Impact:

So far, no issue has been seen.

Software Workaround:

No software workaround.

Silicon Fix:

No hardware fix is scheduled. This is a silicon design deviation from the JEDEC standard.

ENGcm12378 EIM: AUS mode is non functional for devices larger than 32MB

Description:

When the AUS bit is set, the address lines of the EIM are un-shifted. By default, AUS bit is cleared and address lines are shifted according to port size (8, 16 or 32 bits). Due to an error, the address bits 27:24 are shifted when AUS=1. For example, CPU address: 0xBD00_0000 ([A27:20]=1101 0000 becomes: 0xB600_0000 ([A27:20]=0110 0000) on the EIM bus. Since A[27:25] is shifted to [A26:24] and A[23:0] is not shifted. As a result A[24] is missed.

Impact:

If the memory used does not exceed 32 MB, there is no impact. This mode is related to a unique memory configuration that is not often used. Most systems can work in the default mode (AUS=0). Board designers should connect the EIM address bus without a shift (for example, A0→A0, and A1→A1), while working in AUS=0 mode.

Software Workaround:

- Use the AUS = 0 mode (default) while connecting the address signals without a shift (for example, A0→A0 and A1→A1).
- For AUS=1, for devices larger than 32 MB, need to build a memory map that takes this shifting into consideration and does not include A[24] line.

Silicon Fix:

No hardware fix is scheduled.

Linux BSP Status:

NA.

WinCE BSP Status:

NA.

How to Reach Us:

Home Page:
www.freescale.com

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

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
1-800-521-6274 or
+1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku
Tokyo 153-0064
Japan
0120 191014 or
+81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

Document Number: IMX51CE
Rev. 5
02/2012