

Chip Errata for the i.MX25

This document details all known silicon errata for the i.MX25. [Table 1](#) provides a revision history for this document.

Table 1. Document Revision History

Rev. Number	Date	Substantive Change(s)
Rev. 7.1	07/01/2014	<ul style="list-style-type: none"> Replaced PLL erratum with new, ERR007917.
Rev. 7	04/2014	<ul style="list-style-type: none"> Added new PLL erratum, ERR007917.
Rev. 6	09/2012	<ul style="list-style-type: none"> Added back the Drylce erratum, ENGcm11122. Added two new LCDDC errata, ENGcm12383 and TLSbo95476. Added a new USB erratum, BID2773, and updated USB erratum, ENGcm09152. Added a new WEIM erratum, ENGcm11891.
Rev. 5	03/2012	<ul style="list-style-type: none"> Added a new CCM erratum, ENGcm12381. Added a new USB erratum, ENGcm11802. Removed the Drylce erratum, ENGcm11122. Updated Table 2 with silicon version 1.2 information.
Rev. 4	05/2011	<ul style="list-style-type: none"> Added a new ESDRAMC erratum, ENGcm11876.
Rev. 3	07/2010	<ul style="list-style-type: none"> Added a new WEIM erratum, ENGcm11409.
Rev. 2	05/2010	<ul style="list-style-type: none"> Added a new WEIM erratum, ENGcm11270.
Rev. 1	02/2010	<ul style="list-style-type: none"> Added a new Drylce erratum, ENGcm11122.
Rev. 0	06/2009	Initial release.

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

MCIMX25 Revision	Package	MCIMX25 Device Marking	Mask Revision
1.1	17 x 17 mm	MCIMX251AJM4	M09P
1.1	17 x 17 mm	MCIMX251AVM4	M09P
1.1	17 x 17 mm	MCIMX253DJM4	M09P
1.1	17 x 17 mm	MCIMX253CJM4	M09P
1.1	17 x 17 mm	MCIMX253DVM4	M09P
1.1	17 x 17 mm	MCIMX253CVM4	M09P
1.1	17 x 17 mm	MCIMX255AJM4	M09P
1.1	17 x 17 mm	MCIMX255AVM4	M09P
1.1	17 x 17 mm	MCIMX257CJM4	M09P
1.1	17 x 17 mm	MCIMX257DJM4	M09P
1.1	17 x 17 mm	MCIMX257DVM4	M09P
1.1	17 x 17 mm	MCIMX257CVM4	M09P
1.1	17 x 17 mm	MCIMX258CJM4	M09P
1.1	17 x 17 mm	MCIMX258CVM4	M09P
1.2	17 x 17 mm	MCIMX251AJM4A	M09P
1.2	17 x 17 mm	MCIMX253DJM4A	M09P
1.2	17 x 17 mm	MCIMX253CJM4A	M09P
1.2	17 x 17 mm	MCIMX255AJM4A	M09P
1.2	17 x 17 mm	MCIMX257CJM4A	M09P
1.2	17 x 17 mm	MCIMX257DJM4A	M09P
1.2	17 x 17 mm	MCIMX257DJM4AR2	M09P
1.2	17 x 17 mm	MCIMX257CJN4A	M09P
1.2	17 x 17 mm	MCIMX258CJM4A	M09P

Table 3 summarizes all known errata on the i.MX25 device, silicon revision 1.2.

Table 3. Summary of Silicon Errata

Errata	Name	Solution	Page
ARM®			
ENGcm06383	ARM®, ETB: Unpredictable behavior in ETB after power-up	No fix scheduled	5
ENGcm06384	ARM, ETB: ETB9 function not guaranteed	No fix scheduled	6
ARM No. 14	ARM926: Setting the CP15 disable write back bit can cause data loss for dirty cache lines	No fix scheduled	7
ARM No. 15	ARM926: Shift-IR produces incorrect output on TDO when scan chain 15 is selected with INTEST	No fix scheduled	8

Table 3. Summary of Silicon Errata (continued)

Errata	Name	Solution	Page
CCM			
ENGcm07872	CCM: Drylce module ipg_clk enable signal is wrong	No fix scheduled	9
ENGcm12381	CCM: PLL locks at lower frequency when returning from a low power mode	No fix scheduled	10
Drylce			
ENGcm08458	Drylce: Clock-gating problem	No fix scheduled	11
ENGcm11122	Drylce: Unexpected Reset	Fixed in silicon revision 1.2	12
eSDHC			
ENGcm01112	eSDHC: For single block transfer, the block count must be set to 1 in eSDHCv2	No fix scheduled	13
ENGcm02523	eSDHC: eSDHC_WP signal is not available at a pad for the eSDHCv2 module	No fix scheduled	14
ENGcm02759	eSDHC: eSDHC_CLK value out of reset is not according to standard	No fix scheduled	15
ENGcm03419	eSDHC: eSDHC writes incorrect data after transfer is resumed	No fix scheduled	16
ENGcm03648	eSDHC: eSDHC cannot send an interrupt to a card leaving the busy state at the end of an R1b response	No fix scheduled	17
ENGcm06519	eSDHC: Buffer overrun for slow CPU polling read with WML set to 1	No fix scheduled	18
ENGcm06704	eSDHC: Unable to issue a CMD12 command because the SDCLK is stopped	No fix scheduled	19
ENGcm07207	eSDHC: CMD12 abort operation does not abort data transfer on AHB	No fix scheduled	20
ENGcm07452	eSDHC: eSDHC last transfer data fails when the WML register RD_WML field is set to DMA mode and is 1 byte or 127 bytes	No fix scheduled	21
ENGcm08158	eSDHC: Missing synchronizer for crc_reset signal	No fix scheduled	22
ENGcm08618	eSDHC: Cannot finish write operation after block gap stop	No fix scheduled	23
ESDRAMC			
ENGcm11876	ESDRAMC: Data corrupted during burst access across a page boundary of an 8-column SDRAM	No fix scheduled	24
FlexCAN			
ENGcm09158	FlexCAN: Glitch filter is not implemented	No fix scheduled	25
LCDC			
ENGcm07723	LCDC/IOMUX: Default out of display pins does not match LCDC 16 bpp mode in IOMUX	No fix scheduled	26
ENGcm12383	LCDC: Graphic window may fail to get disabled properly	No fix scheduled	27
TLSbo95476	LCDC: LSCLK is missing	No fix scheduled	28
RTIC			
ENGcm05171	RTIC: Reading the RTIC command register (CMD) can cause a software reset	No fix scheduled	30
ENGcm05860	RTIC: Writing the SWRST (software reset) bit of the command register while RTIC is performing a one-time hash operation can cause an error	No fix scheduled	31

Table 3. Summary of Silicon Errata (continued)

Errata	Name	Solution	Page
SJC			
ENGcm06977	SJC: Undefined test data output (TDO) state when SJC bypass is invoked and ARM is in low-power mode	No fix scheduled	32
SSI			
ENGcm03133	SSI: Enabling SSI within 2-3 clock cycles of a Frame Sync signal results in a glitch on the data line	No fix scheduled	33
ENGcm06222	SSI: Transmission does not take place in bit-length early frame sync configuration	No fix scheduled	34
ENGcm06474	SSI: Incorrect data TX starts from FIFO1 when RX is enabled before TX in sync mode	No fix scheduled	35
ENGcm09835	SSI: The EMI pad does not have input hysteresis protection, so avoid using the EMI pad to mux the SSI bit clock	No fix scheduled	36
USB			
BID1232	USB: EHCI NYET response to a complete split non compliance	No fix scheduled	37
BID1340	USB: CERR counter not reset during an ACK or NAK response to a PING	No fix scheduled	38
BID2053	USB: RX endpoint flush when receiving a token OUT	No fix scheduled	39
BID2108	USB:1 byte CRC error/2 bytes long packet error	No fix scheduled	40
BID2477	USB: Port-change interrupt not generated	No fix scheduled	41
BID2581	USB: Host RX FIFO overflow generates a false frame babble.	No fix scheduled	42
BID2655	USB: ISO mult3 fulfillment error	No fix scheduled	43
BID2656	USB: ISO mult3 third packet not fetched before the active microframe	No fix scheduled	44
BID2773	USB: CRC not inverted when FIFO underruns	No fix scheduled	45
ENGcm08333	USB: Host core wake up fails when using internal serial PHY.	No fix scheduled	46
ENGcm09152	USB: UTMI_USBPHY VBUS input impedance implementation error	No fix scheduled	47
ENGcm09460	USB: USB PHY autoresume in host mode	No fix scheduled	49
ENGcm11802	USB: USB OTG generates extraneous pulse on DP on suspend	No fix scheduled	50
WEIM			
ENGcm11270	WEIM: AUSx bits do not work for address bit A[23]	No fix scheduled	51
ENGcm11409	WEIM: In FCE=1 mode, WEIM cannot correctly sample the data, if there is ECB asserted during burst access	No fix scheduled	52
ENGcm11891	WEIM: Unexpected deassertion when page mode emulation is used with non-32-bit wide port size	No fix scheduled	53

ENGcm06383 ARM®, ETB: Unpredictable behavior in ETB after power-up**Description:**

Non-resettable flip-flops (in ETM) drive ETB internal states and cause unpredictable behavior after power-up.

Projected Impact:

This issue causes unpredictable ETB behavior after power-up.

Workarounds:

There is a software workaround:

1. Power up ETM
2. Enable ETB as normal
3. Disable ETB (clear ETB control register, bit 0: TraceCaptEn)
4. Reset ETB write pointer
5. Re-enable ETB (set ETB control register, bit 0: TraceCaptEn)

This workaround is confirmed by ARM.

Proposed Solution:

No fix scheduled

ENGcm06384 ARM, ETB: ETB9 function not guaranteed

Description:

Due to a timing problem, ETB9 function is not guaranteed to be correct when ARM clock frequency is above 133 MHz.

Projected Impact:

ETB9 functions are not guaranteed when the ARM clock frequency is larger than 133 MHz.

Workarounds:

While using ETB functions, keep the ARM clock at or below 133 MHz.

Proposed Solution:

No fix scheduled

ARM No. 14 ARM926: Setting the CP15 disable write back bit can cause data loss for dirty cache lines**Description:**

Setting the disable write back (DWB) bit of the cache debug control register can cause stale data to be written to main memory on a cache line eviction. The DWB bit is normally only set during debug sessions; however, write-back memory locations can occasionally be corrupted.

Projected Impact:

If a data cache is being used, setting the disable write back bit of the cache debug control register can corrupt cached memory locations.

Workarounds:

This can be avoided by setting the disable DCache linefill (DDL) bit (bit[0]) of the cache debug control register whenever the DWB bit is set.

```
MRC p15, 7, Rd, c15, c0, 0 ; Read Cache debug control register
ORR Rd, Rd, #0x5 ; Set DWB and DDL
MCR p15, 7, Rd, c15, c0
```

Proposed Solution:

No fix scheduled

ARM No. 15 ARM926: Shift-IR produces incorrect output on TDO when scan chain 15 is selected with INTEST

Description:

Incorrect data is output on TDO during Shift-IR when scan chain 15 is selected. If scan chain 15 is selected with INTEST and the TAP state machine is in the Shift-IR state, the least significant bit of the scan chain 15 data register is output on TDO rather than the expected instruction register data.

Projected Impact:

If scan chain 15 has just been accessed and a new instruction is shifting in, the shifted-out data observed on JTAG TDO is not the last instruction.

Workarounds:

To avoid this situation, another scan chain should be selected after accessing scan chain 15.

Proposed Solution:

No fix scheduled

ENGcm07872 CCM: DryIce module ipg_clk enable signal is wrong**Description:**

The ECT module clock-enable signal does not gate ECT's ipg_clk, but instead gates DryIce's ipg_clk. The ECT module clock-enable signal is low (disabled) when there is no debug activity on ECT.

Projected Impact:

DryIce has an active ipg_clk and works normally when the ECT module clock is enabled, while ECT always has an active ipg_clk whether ECT is in debug status or not.

Workarounds:

A software workaround is to always keep CCM_CCTR bit 15 (CG CTL) at 0, so that all clock gating is controlled by the individual IPG clock-gating registers (CCM CGCR1 and CGCR2) instead of the individual modules.

Proposed Solution:

No fix scheduled

ENGcm12381 CCM: PLL locks at lower frequency when returning from a low power mode**Description:**

If the OSC24M_DOWN option is enabled when returning back to run mode from a low power mode, then, the PLL will lock at a lower frequency than the one that was configured before going to low power mode. The PLL needs to be reset to run at the configured frequency.

Projected Impact:

To ensure that the PLL resumes at the desired frequency following exit from low power mode, caution must be exercised if the OSC24M_DOWN option is used.

Workarounds:

The steps to follow after returning from stop mode are:

1. Run code in IRAM after stop mode.
2. Wait about 500 μ s after resume.
3. Precharge SDRAM (or DDR) and put SDRAM (or DDR) to manual self-refresh mode. No access to SDRAM (or DDR) is allowed before (5) setting SDRAM (or DDR) to normal mode.
4. Reset PLL.
5. Set SDRAM (or DDR) to normal mode.
6. Go to normal processing as original.

Proposed Solution:

No fix scheduled

ENGcm08458 DryIce: Clock-gating problem**Description:**

RNGB and a gasket that connects DryIce and RNGB use the same ipg_clk. This ipg_clk can be gated in the CCM module, but the gate-enable signal comes from RNGB. When the RNGB FIFO is full, the RNGB ipg_clk is gated off, and this causes the gasket to stop working, which essentially disconnects DryIce and RNGB, even if DryIce requests a random key change from RNGB.

Projected Impact:

The subroutine that loads a random key to DryIce needs to be modified to bypass this problem as the shown in the workaround below; otherwise, no random key will be generated in DryIce.

Workarounds:

If DryIce requests a new random key by writing 1'b1 to DRYICE_DKCR[0] and clearing CRM_CCTL[15]=0, a module-enable signal is not used to generate clock-gating control signals and the gasket that connects DryIce and RNGB has a clock to drive its logic. After DRYICE_DSR[5] becomes 1'b1, CRM_CCTL[15]=1 can be recovered.

Proposed Solution:

No fix scheduled

ENGcm11122 DryIce: Unexpected Reset**Description:**

The DryIce module contains an internal power switch, which is used to switch between the backup battery input (BAT_VDD) and the system power (QVDD). When the system is powered off (shut down), the backup battery power is used by the DryIce module to maintain the RTC and the security features. The advantage of the battery backup power is, when the system is powered on again, the SRTC features (time counter, monotonic counter, general purpose register, secure keys, and so on) are still valid.

There is an error in the switching mechanism that may cause a reset in the DryIce module during the switch from the backup battery to system power. In turn, this invalidates the SRTC features when the system is powered on again.

Projected Impact:

Although this behavior is exhibited randomly, the SRTC features are not guaranteed to be maintained with backup battery (BAT_VDD) when the system is powered off. RTC and security related information can be lost during the power switch.

Workarounds:

None

Proposed Solution:

Fixed in silicon revision 1.2

ENGcm01112 eSDHC: For single block transfer, the block count must be set to 1 in eSDHCv2**Description:**

The eSDHC block guide mentions that in the case of a single block transfer, the BLKCNT field of the block attributes register can be ignored (“don’t care”). However, not setting it to a non-zero value does not issue a write or read command. For this reason, the block count must be set to 1 for a single block transfer.

Projected Impact:

Software must set BLKCNT when performing a single-block transfer.

Workarounds:

Set the block counter to 1 for a single block transfer.

Proposed Solution:

No fix scheduled

ENGcm02523 eSDHC: eSDHC_WP signal is not available at a pad for the eSDHCv2 module**Description:**

The eSDHC_WP (write protect) is an input signal from the SD card into the eSDHC module. However, the eSDHC module does not have a dedicated pad available for eSDHC_WP, and the input port to the module is fixed at 1'b0, which means that the hardware solution for write-protect is disabled. For this reason, a GPIO is required to receive the protect-switching signal from the card device if a write-protect is going to be implemented.

Projected Impact:

There is no write-protect signal for the eSDHC module.

Workarounds:

Use a GPIO signal to receive the write-protect signal from the card device and a software solution to implement write-protect.

Proposed Solution:

No fix scheduled

ENGcm02759 eSDHC: eSDHC_CLK value out of reset is not according to standard**Description:**

Based on the SD standard, the eSDHC_CLK should be low when it is disabled. Instead, this clock is high when disabled.

Projected Impact:

Although this is not compliant with the SD standard, there is no functional impact.

Workarounds:

None.

Proposed Solution:

No fix scheduled

ENGcm03419 eSDHC: eSDHC writes incorrect data after transfer is resumed**Description:**

If a write operation transfer is paused, but all the data in the internal FIFO has been sent, the eSDHC writes dirty data to the card when the transfer is resumed.

The chance that this errata would occur in a real application is rare. It only happens when all of the following conditions are met:

- Write operation
- Transfer paused
- The system-side write is slow, so there is no data remaining in the FIFO when the transfer stops.
- The transfer is resumed without sending a suspend command to suspend the transfer.

Projected Impact:

Pausing a transfer during a write operation can cause an error.

Workarounds:

Avoid using the SABGREQ feature for write operation when the system-side write is slow.

Proposed Solution:

No fix scheduled

ENGcm03648 eSDHC: eSDHC cannot send an interrupt to a card leaving the busy state at the end of an R1b response

Description:

MMC/SD protocol specifies that after sending an AutoCMD12 command, the MMC/SD card returns an R1b response with a busy signal transmitted on the DAT0 line. If, at the end of the AutoCMD12 command, the MMC/SD card is not in the TRAN state (that is, the card has not left the busy state), a subsequent data command may cause a DCE error.

To determine the MMC/SD card-busy state after the AutoCMD12 command and R1b response, poll the DAT0 line.

Projected Impact:

A DCE error can be introduced by data commands following an AutoCMD12 command if the DAT0 line is not polled to check whether the MMC/SD card has already left the busy state.

Workarounds:

Software can use both the transfer complete (TC) interrupt and DAT0 to check if AutoCMD12 with busy is completed. There are three workarounds, as described in [Table 4](#).

NOTE

Workaround 3 is the preferred workaround.

Table 4. Workaround for AutoCMD12 and R1b Polling Problem

Workaround No.	Clock Rate	Process
1	Fast clock rate, high-speed card	1. Wait for the TC interrupt. 2. Software polls DAT[0] via the PRSSTAT[24] bit until it is HIGH.
2	Any clock rate	1. Wait for the TC interrupt. 2. Software sends CMD13. 3. Software polls DAT[0] via the PRSSTAT[24] bit until it is HIGH.
3	Any clock rate	Preferred Workaround: 1. Wait for the TC interrupt. 2. Software sends CMD13 iteratively until the card returns to the TRAN state.

Proposed Solution:

No fix scheduled

ENGcm06519 eSDHC: Buffer overrun for slow CPU polling read with WML set to 1**Description:**

When the watermark level (WML) is set to 1, if the system read speed is too slow (that is, CPU accesses the eSDHC internal buffer at a very slow rate), the card_addr signal goes around the whole buffer and overrides the buffer space. Because a buffer overrun occurs and the buffer-read ready signal is updated by a read pulse, the buffer-read ready signal is not asserted again. A buffer overrun occurs and the card clock is stopped. This prevents the use of a CPU polling read with WML set as 1.

Projected Impact:

Add a software restriction. WML set to 1 is not a requirement of the host controller standard specification, so software can set WML from 2 to 126.

Workarounds:

Do not set the watermark level (WML) to 1.

Proposed Solution:

No fix scheduled

ENGcm06704 eSDHC: Unable to issue a CMD12 command because the SDCLK is stopped**Description:**

The eSDHC stops the SDCLK when buffer overrun or underrun occurs. If the eSDHC does not read from its internal buffer as quickly as data is written to the buffer from the card, and the buffer fills up, then the eSDHC stops the card clock (SDCLK) to avoid buffer overrun. An attempt to send a CMD12 command fails when the SDCLK is stopped.

However, in this situation, if the user generates a CMD12 command (with XFERTYP:CMDTYP=3, abort) to stop the MULTI_BLOCK_READ or MULTI_BLOCK_WRITE transfer, in the eSDHC it looks as if the CMD12 command does not make it to the SD/MMC interface.

The SD/MMC clock never restarts after a buffer overrun or underrun.

Projected Impact:

The eSDHC failed to issue a CMD12 command when the eSDHC had a buffer overrun or underrun.

Workarounds:

Read the PRSSTAT[7] (SDOFF) status bit to check whether SDCLK is stopped or not. If the SDCLK is stopped, make several accesses (write or read, depending on the transfer direction) to the buffer to leave the overrun or underrun state so that SDCLK is automatically restarted. Send a CMD12 command.

Proposed Solution:

No fix scheduled

ENGcm07207 eSDHC: CMD12 abort operation does not abort data transfer on AHB**Description:**

Write Multiple Block: If a CMD12 command is sent during a Write Multiple Block transfer, the AHB bus keeps writing to the internal buffers. This is undesirable behavior. During this situation, the AHB bus does not stop until all the blocks are written to the internal buffer, and an AutoCMD12 command is sent.

A typical scenario is as follows: After sending a non-ending block, the card replies with a CRC error. The software detects the CRC error and manually sends a CMD12 command to the card to stop the transmission. Internally, the AHB bus keeps writing to the internal buffer even though the software stopped the transfer.

Read Multiple Block: If a CMD12 command is sent during a Read Multiple Block transfer, the AHB bus keeps reading from the internal buffers. This is undesirable behavior. During this situation the AHB bus does not stop until all the blocks are read from the internal buffer and an AutoCMD12 command is sent.

A typical scenario is as follows: The software decides to stop the Read Multiple Block transfer. The software sends a CMD12 command to signal to the card that the transfer should stop, but internally, the AHB keeps reading data from the internal buffer.

Another possible scenario is, when the card detects an error during a nonending block (for example, out of range, address misalignment, internal error), the card stops data transmission and remains in the data state. Since the following block never arrives, the AutoCMD12 command is not sent, and a read-data timeout occurs. After the read-data timeout is detected by the software, it must send a CMD12 command (per MMC/eMMC protocol standard), but the AHB bus keeps reading from the internal buffer.

Projected Impact:

The CMD12 abort operation does not abort data transfers on AHB.

Workarounds:

To abort data transfers on the AHB, software can reset the eSDHC by writing 1 to SYSCTL[24] (RSTA).

Proposed Solution:

No fix scheduled

ENGcm07452 eSDHC: eSDHC last transfer data fails when the WML register RD_WML field is set to DMA mode and is 1 byte or 127 bytes**Description:**

In eSDHC, when using DMA access mode, the last transfer of data is wrong when the read watermark level (WML) is 1 byte or 127 bytes.

Projected Impact:

Add a software restriction. WML set to 1 is not a requirement of the host controller standard specification, so software can set WML from 2 to 126.

Workarounds:

When using DMA access mode, configure the RD_WML field of the WML register with any value from 2 bytes to 126 bytes. Avoid setting the RD_WML field to 1 byte or 127 bytes (typically, it is not common to set WML as 1 byte or 127 bytes in a real application).

Proposed Solution:

No fix scheduled

ENGcm08158 eSDHC: Missing synchronizer for crc_reset signal**Description:**

dat_trans_dir is 1'b1 in read mode, and 1'b0 is in write mode. When moving from a read operation to a write operation, software configures the DTDSEL bit, causing a dat_trans_dir change. This signal drives the crc_reset signal. If the negation of the change happens to sit in the setup/hold timing window, it can lead to uncertainty in the crc_val[15:0] for the next write transfer. This causes the write-CRC generator to work incorrectly.

Projected Impact:

A CRC check finds an error when writing data.

Workarounds:

There are two possible workarounds:

- Apply bit[25] of the system control register RSTC (software reset for CMD line) after the read operation is done (transfer complete).

NOTE

This workaround is recommended because it is easier to implement.

- After the read operation is complete, clear bits [2:0] (PEREN, HCKEN, IPGEN) of the system control register, and wait for bit 7 (stops the card clock) or bit 6 (stops the internal baud rate clock) of the present state register to be set. Then send any command (either CMD13 to poll status or CMD24 or CMD25 to launch write operation).

Proposed Solution:

No fix scheduled

ENGcm08618 eSDHC: Cannot finish write operation after block gap stop**Description:**

The eSDHC supports a mechanism called Stop At Block Gap to stop data transfer during a block gap, the time between the transmissions of two blocks. Software can, at any time, write 1 to PROCTL[16] (SABGREQ) to request the bus to stop the transfer at the next block gap. To resume the transfer, software must wait until transfer complete is set to 1, and it can continue the transfer by writing 1 to PROCTL[17] (CREQ).

If Stop At Block Gap is enabled (SABGREQ set to 1) in MULTIBLOCK WRITE, and transfer is resumed afterwards, the data transfer cannot finish and the transfer complete status bit is not set, because the eSDHC internal FIFO fetches the wrong data when transfer resumes from Stop At Block Gap.

Projected Impact:

Stop At Block Gap does not work for Multiblock Write, but it works for Multiblock Read.

Workarounds:

No workaround. Software should be written so that it does not attempt to stop write transfers during block gaps.

Proposed Solution:

No fix scheduled

ENGcm11876 ESDRAMC: Data corrupted during burst access across a page boundary of an 8-column SDRAM**Description:**

When the Enhanced SDRAM Controller (ESDRAMC) is connected to an SDRAM addressed with 8 columns, the read and write accesses can be corrupted. This corruption is due to the fact that the ESDRAMC cannot perform a burst access across a row/page boundary. As a result, the necessary split of the burst access between row N and row N+1 is corrupted.

Projected Impact:

SDRAM devices addressed with 8 columns, organized in 1M x 16 bit x 4 banks, cannot be used.

Workarounds:

An SDRAM addressed with at least 9 columns must be used.

Proposed Solution:

No fix scheduled

ENGcm09158 FlexCAN: Glitch filter is not implemented**Description:**

When the system is in low-power mode, if there is a lot of noise on the CANRX bus, the FlexCAN wake-up logic is triggered, because there is no glitch filter logic circuit in the design. The system will then be awakened by mistake.

Projected Impact:

Increased power consumption.

Workarounds:

Select the appropriate CAN transceiver that supports the wake-up mechanism and has a glitch-filter circuit. Then, the system can be awakened by a transceiver through a GPIO interrupt instead of the FlexCAN wake-up logic. An example of this kind of transceiver is TJA1041, which is an NXP product.

Proposed Solution:

No fix scheduled

ENGcm07723 LCDC/IOMUX: Default out of display pins does not match LCDC 16 bpp mode in IOMUX**Description:**

The default display pin-out in i.MX25 is mismatched to LCDC 16-bpp mode. When running in 16-bpp mode, LD[1:11] and LD[13:17] are used for the 16 bits, instead of LD[0:15].

Users need to use LD[16] and LD[17] to implement 16 bpp mode, but LD[16] and LD[17] is not a default mux option.

Projected Impact:

Special considerations are needed to mux LCDC data signals out of chip pads when using LCDC in 16-bpp mode.

Workarounds:

Use LD[16] and LD[17] to implement 16 bpp mode. There are several ways to mux out LD[16] and LD[17] signals from the chip pads. The suggested option is to use GPIO_E and GPIO_F pads to output LD[16] and LD[17] signals.

Proposed Solution:

No fix scheduled

ENGcm12383 LCDC: Graphic window may fail to get disabled properly

Description:

When the LCDC graphic window (GW) is used in an application, it may fail to get disabled properly. This may result in an unexpected memory traffic from the LCDC as GW's FIFO is still fetching data.

Projected Impact:

When the GW is disabled, the associated FIFO, responsible for pixel transfer from the frame buffer memory to the LCDC external bus, may remain in a state in which it continues to fetch data from the frame buffer indefinitely. This generates a continuous unexpected flow of data that impacts the memory bus bandwidth, even if the GW has been disabled visually. This load is higher than when the GW is enabled as the trigger mechanisms of the FIFO are faulty, making the data demand permanent and no more under control.

Workarounds:

If the GW is not needed in an application, the driver must never enable the GW. Otherwise, there is a risk to leave it in the described state when disabled.

Recommended workaround:

If the GW is used in an application, it is recommended to never disable the GW once it has been enabled. Instead of disabling the GW, the frame size should be reduced to the minimum (16x16 pixels) to minimize the data traffic. In addition, the GW can be made visually disabled by setting the full transparent mode.

When the GW is not in use, the permanent traffic in 16bpp @60fps is: $16 \times 16 \times 2\text{bytes} \times 60\text{fps} = 30720 \text{ bytes/s}$. In this situation, the impact on the memory bus is very limited.

Alternative workaround:

If the following steps are performed when disabling the GW, the problem disappears:

1. Clear the LCDC_EN bit in CCM register, CGCR1.
2. Clear the LCDC register, LGWDCR.
3. Set the LCDC_EN bit in CCM register CGCR1.
4. Disable the GW.

When the GW is enabled, the correct window size must be reset in LGWDCR.

This workaround is logically functional; however, if the LCDC clock is disabled, even for few nano/micro seconds between step 1 and step 3, it may create a visual glitch artifact on the LCD panel.

Although this is not an acceptable solution in most use cases, it is left to the user to decide what is the best solution for the product.

Proposed Solution:

No fix scheduled

TLsbo95476 LCDC: LSCLK is missing**Description:**

When Bits Per Pixel (PBIX) in LPCR register is set to 4 bpp, 8 bpp, or 18 bpp, LSCLK of LCDC is missed in following three cases:

- One clock cycle before assertion of OE_ACD
- When VSYNC is asserted
- When VSYNC is negated

Projected Impact:

If the user connects LSCLK directly with LCD panel and the LCD panel allows missing clock, then, there is no issue. However, if the user adds a serializer between LSCLK and LCD panel, then, the serializer may not work correctly because of missing clock and the LCD panel may have noise.

Workarounds:

Connect LSCLK directly with LCD panel, if the LCD panel allows it.

Proposed Solution:

No fix scheduled

ERR007917 PLL: The PLL lock detection circuit may indicate loss of lock unexpectedly when MFN=0.**Description:**

The PLL lock detection circuit may indicate loss of lock unexpectedly when MFN=0. Both the MPLL and UPLL are affected by this erratum.

Projected Impact:

PLL may indicate the loss of lock (LOCK bit de-asserts) unexpectedly when MFN=0. The details of the impact on MPLL and UPLL are as follows:

MPLL:

- The LOCK status bit in MPCTL is used within the CCM circuit for ARM core and ARM bus clock-gating. When the LOCK bit incorrectly de-asserts, the CCM clock-gating logic can create narrow clock pulses which will corrupt the ARM core and ARM bus clock. Thus the system cannot operate properly when MFN=0 in the MPLL.
- The MPCTL register and BSP default settings for the MPLL's MFN are non-zero.

UPLL:

- The LOCK status bit in UPCTL may incorrectly de-assert when MFN=0.
- The LOCK status bit in UPCTL is NOT used to drive any clock-gating function and therefore there are no downstream logic issues within the device.

Workarounds:

For MPLL: DO NOT program MFN to zero.

For UPLL: MFN must first be set to a non-zero value when SW polling is required for the LOCK status bit.

Proposed Solution:

No fix scheduled.

ENGcm05171 RTIC: Reading the RTIC command register (CMD) can cause a software reset

Description:

A software reset should only occur when writing to the SWRST bit of the command register CMD[1]. Currently, reading CMD[1] can also cause a software reset.

Projected Impact:

This can cause an unexpected RTIC software reset on reading the CMD register.

Workarounds:

Read all status information from the STATUS register instead of the CMD register. Whether reading the CMD register causes a software reset depends on the value on an internal data bus. Because this cannot be controlled, it is never safe to read the command register.

Proposed Solution:

No fix scheduled

ENGcm05860 RTIC: Writing the SWRST (software reset) bit of the command register while RTIC is performing a one-time hash operation can cause an error

Description:

If RTIC is hashing data in the one-time hash mode, a write to the software reset bit, CMD[1] (SWRST), can cause RTIC to generate an error, which can cause RTIC to misbehave.

Projected Impact:

This erratum can cause unexpected RTIC behavior.

Workarounds:

Do not write 1 to the SWRST bit in the command register while RTIC is hashing data. Allow RTIC to finish the hashing operation, and then write 1 to SWRST bit.

If writing to the SWRST bit while RTIC is hashing data causes RTIC to generate an error, this can be cleared by writing 1 to the SWRST bit a second time.

In general, production software should rarely, if ever, need to write to the SWRST bit of the CMD register.

Proposed Solution:

No fix scheduled

ENGcm06977 SJC: Undefined test data output (TDO) state when SJC bypass is invoked and ARM is in low-power mode**Description:**

When putting the ARM platform into low-power mode (LPM), if the bypass circuitry in the SJC module is invoked (setting ARM_bypass=1, and DAP_bypass=1) prior to the ARM entering into LPM, the output on TDO goes to an undefined state.

If arm_bypass and dap_bypass are set (by writing SJC's DCR[8]) before entering into low-power mode, ARM cannot enter into low-power mode. This is because ARM's dbgen signal is asserted when SJC tries to set arm_bypass and dap_bypass, and the dbgen signal forces the ARM clock to be always on. It can be explained as follows: when in daisy mode, ARM tap and DAP tap are on chain by default; any operation on this chain is passed into these 2 taps. When SJC writes to set arm_bypass, ARM detects this action through its tap, assumes that debug is enabled, and forces its clock on to support this debug action.

Projected Impact:

ARM cannot enter low-power mode after any SJC operation when it is in daisy mode (in which ARM tap and DAP tap are on chain by default).

Workarounds:

Do not invoke SJC bypass circuitry prior to putting the ARM platform into low-power mode.

Proposed Solution:

No fix scheduled

ENGcm03133 SSI: Enabling SSI within 2-3 clock cycles of a Frame Sync signal results in a glitch on the data line**Description:**

If SSI mode is configured in network mode (SCR[3]), and SSI is configured to provide the frame sync (frame sync is generated by SSI (STCR[6], TFDIR bit is set), then if TX_EN bit (SCR[1]) is set 2-3 clock cycles before frame sync occurs on the SSI_STFS line, a glitch can be seen on the data line SSI_STXD for the second time slot. After the second time slot, proper data transmission takes place.

Projected Impact:

This can cause noise in the first sample of the signal, but subsequent samples will be correct.

Workarounds:

None.

Proposed Solution:

No fix scheduled

ENGcm06222 SSI: Transmission does not take place in bit-length early frame sync configuration**Description:**

When SSI is programmed to generate the clock and is configured for bit-length early frame sync, SSI hangs if the following bits are programmed as follows in sequential order:

1. SCR[0] (SSI_EN) bit is set. The module starts generating the clock and frame sync as soon as this bit is set.
2. SCR[1] (TXEN) is set after a frame cross-over.

Projected Impact:

This can cause the SSI module to work incorrectly.

Workarounds:

Software must write SCR[1] (TXEN) and SCR[0] (SSI_EN) bits to 1 in the same register write.

Proposed Solution:

No fix scheduled

ENGcm06474 SSI: Incorrect data TX starts from FIFO1 when RX is enabled before TX in sync mode**Description:**

The SSI module is configured as follows:

- The module is operating in network synchronous mode (SCR[4], SYN & SCR[3], NET bits are set).
- The module is configured for two-channel mode of operation (SCR[8], TCH_EN bit is set).

With this configuration, if the RX_EN bit (SCR[2]) is set and, after the passage of one or more frames, the TX_EN bit (SCR[1]) is set, then wrong data is transmitted—the first data is transmitted from the second channel (FIFO1) instead of the first channel (FIFO0).

Projected Impact:

This causes an SSI data-transfer error.

Workarounds:

When configured in network synchronous mode, the RX_EN (SCR[2]) bit and TX_EN (SCR[1]) should both be enabled during the same register write.

Another way to avoid this issue is to not use the two-channel mode of operation. That is, the TCH_EN(SCR[8]) bit should not be set in the SCR register.

Proposed Solution:

No fix scheduled

ENGcm09835 SSI: The EMI pad does not have input hysteresis protection, so avoid using the EMI pad to mux the SSI bit clock**Description:**

If the SSI module signals are muxed out of the EMI pads, random left/right channel swap is heard on the SSI port. This is caused by the fact that there is no hysteresis protection implemented on the input bit clock signal on the EMI pads. Because the EMI pads in the power bank are noisy, the SSI bit clock is very sensitive. Random left/right channel swapping occurs when, without the protection of hysteresis on these pads, power-bank noise causes a glitch in the rising/falling edges of the SSI bit clock and causes a FIFO data-pointer mismatch with the frame indication.

Workarounds:

Avoid using the EMI pad to mux out the SSI bit clock, because there is no input hysteresis protection on this pad.

Use other muxing options for SSI.

Proposed Solution:

No fix scheduled

BID1232 USB: EHCI NYET response to a complete split non compliance**Description:**

The NYET response to CSPLIT bulk is not EHCI-compliant as far as the NackCnt is concerned; NackCnt field is not decrementing when receiving NYET.

The NYET response to the CSPLIT interrupt is not EHCI-compliant as far as the Cerr is concerned; the Cerr field adjusts to the maximum value 3 on receipt of an NYET on every CSPLIT transaction. The behavior of the reclamation bit is not totally compliant with the EHCI specification; the reclamation bit is setting to zero every time the host fetches the queue head with the H bit set, giving rise to empty list detection, and the async list is not yet empty. This situation is leading the host to the async sleep state repeatedly, because the host is ignoring NackCnt and RL.

Workarounds:

The system software can cancel the transfer if it does not reach the end after some time and retry later.

Proposed Solution:

No fix scheduled

BID1340 USB: CERR counter not reset during an ACK or NAK response to a PING

Description:

When an ACK or NAK is sent from the device in response to a PING, the CERR counter value does not reset to the initial value.

Workarounds:

A software workaround for this issue is possible.

If a value of 0 is used in the CERR field of the dTD when building the data structures, the controller retries the transaction continuously and is not retired due to consecutive errors. This change increases the chance for the transaction to complete. Using CERR = 0 is compliant with the EHCI specification.

Proposed Solution:

No fix scheduled

BID2053 USB: RX endpoint flush when receiving a token OUT**Description:**

When using the OTG controller in device mode, if an RX flush command is issued at the same time as receiving a token OUT to the same endpoint, the packet is lost. That is, it reports ACK, but the data is not copied to memory. Additionally, if the controller is in the stream disable mode (SDIS bit set to 1), the endpoint is also blocked and NAKs any token sent to that endpoint forever.

Workarounds:

If an endpoint is active in device mode, do not use the RX flush feature to clear/stop it.

Proposed Solution:

No fix scheduled

BID2108 USB:1 byte CRC error/2 bytes long packet error

Description:

When using the OTG controller in device mode, when doing IN transfers that have a maximum size of 1 or 2 bytes, CRC and long packet errors can be generated.

Workarounds:

See the application note, *Workaround for i.MX25 USB OTG Erratum BID2108* (order #AN3683 on the web site listed on the back cover of this document), for a description of the workaround.

Proposed Solution:

No fix scheduled

BID2477 USB: Port-change interrupt not generated**Description:**

The self-powered device controller does not generate an interrupt when the session is no longer valid (VBUS not present).

Workarounds:

For the USB On-The-Go (USB OTG) controller, an interrupt is generated in this corner case. After the generation of this interrupt, the software needs to check the status of the OTGSC register to understand the cause of the interrupt (VBUS-related flags change). In the worst case, this interrupt appears around 2 ms after the end of the specific session.

Proposed Solution:

No fix scheduled

BID2581 USB: Host RX FIFO overflow generates a false frame babble.

Description:

Host RX FIFO overflow too close to EndOfFrame generates false frame babble. If an RX FIFO overflow happens close to the next SOF, and the master system interface port is not available, a false frame babble is reported to software, and the port is halted by hardware. This could degrade performance and have system implications because the host has to reset the bus and re-enumerate the connected device.

Workarounds:

None.

Proposed Solution:

No fix scheduled

BID2655 USB: ISO mult3 fulfillment error**Description:**

When using high-bandwidth ISO IN endpoints (MULT=3) combined with low-bandwidth system bus access, the controller may enter into a wait loop situation without warning the software.

Due to the low bandwidth on the system bus, the last packet from a mult3 sequence may not be fetched in time before the last token IN is received (for that uframe/endpoint). This causes the controller to reply with a zero length packet. The controller does not complete the data fetch and the DMA engine is left waiting forever for the data fetch to complete. The controller replies with zero-length packets to all subsequent IN tokens for that endpoint.

Workarounds:

There is no workaround for this, other than making sure the data can be fetched in due time.

Proposed Solution:

No fix scheduled

BID2656 USB: ISO mult3 third packet not fetched before the active microframe

Description:

ISO mult3 third packet is not fetched before the active microframe. When using device ISO IN Mult=3, the third packet is not prefetched until the second packet is being sent. This can result in an underrun.

Workarounds:

None.

Proposed Solution:

No fix scheduled

BID2773 USB: CRC not inverted when FIFO underruns**Description:**

In systems with high latency, the HOST can underrun on OUT transactions. In this situation, it is expected that the transmitted CRC of the truncated data packet is the inverse (complement) of the calculated CRC, signaling a underrun situation and guaranteeing a CRC error at the receiver.

The controller will not send this inverted CRC. Instead, it sends only one byte of the inverted CRC followed by the last byte of payload. It is unlikely but possible that this sent CRC evaluates as a correct CRC at the receiver, causing the device to accept the truncated packet from the host.

Projected Impact:

This issue affects only CRC transmission of a truncated packet as a result of a FIFO underrun. FIFO underrun occurs when the USB controller fails to backfill its FIFO before it runs out of data. This condition occurs when other bus masters prevent the USB controller from accessing the bus in due time.

Workarounds:

The risk for FIFO underrun can be reduced by increasing the TXFIFOTHRES parameter in the TXFILLTUNING register. Increasing this value will instruct the USB controller to load more data in the FIFO, before starting the transmission. This allows for more tolerance to bus access delays.

Proposed Solution:

No fix scheduled

ENGcm08333 USB: Host core wake up fails when using internal serial PHY.**Description:**

The USB full speed host (USB FS HOST) core can support internal and external serial PHY (two RX signals that come from two different serial PHYs). The RX signal input uses a multiplexer to select the RX signal to be used. The wake-up logic RX signal always comes from the external PHY. After a connection event, the USB FS HOST core cannot be awakened when it uses internal serial PHY and is in suspend mode.

Projected Impact:

If the host core uses USB internal PHY and the 60-MHz clock for the USB host controller is turned off, the USB host cannot detect wake-up events.

Workarounds:

There are two workarounds:

- The USB 2.0 specification does not enforce turning off the internal USB HOST IP clock in suspend mode. During USB suspend mode, as long as the USB FS HOST clock is on, USB FS HOST interrupts are still available and intercept any event coming from the internal USB FS PHY. The USB FS HOST is able to send the interrupt to the CPU, even though the interrupt was initially generated by the internal USB FS PHY. The interrupt service routine can recover the USB from suspend mode. To increase power savings, the USB FS HOST clock can be decreased.
If the USB FS HOST clock is reduced to 5 MHz, it introduces an additional 11–16 mA power consumption in suspend mode, compared with the system low-power mode when the USB FS HOST clock is off.
- The user can use other wake-up resources, such as the keyboard, GPIO, touch screen, USB event in OTG, and so on to wake up the SOC from the system-stop mode. When the software detects user interaction, it can quit USB suspend mode by clearing the PORTSC.PHCD bit and restarting USB enumeration.

Proposed Solution:

No fix scheduled

ENGcm09152 USB: UTMI_USBPHY VBUS input impedance implementation error

Description:

The OTG specification states that the input impedance of VBUS should be $40\text{ k}\Omega$ – $100\text{ k}\Omega$ over the 0 V – 5.25 V VBUS input range. USB UTMI PHY implementation violates the requirement from 3 V – 5.25 V .

Projected Impact:

The VBUS input impedance drops to below $1\text{ k}\Omega$ when VBUS is between 3.5 V and 4.2 V , which violates the USB OTG specification. Within that range, the USB PHY consumes 5 mA current on a VBUS line. This can impact normal VBUS charge function, especially in a bus-power system. The low VBUS input impedance may also present a risk for long-term reliability.

Workarounds:

Case 1: The system needs OTG functionality with HNP/SRP. There are two steps:

1. Add an external divider (including two resistors) with a Schottky diode to the USBPHY1_VBUS pin. The external divider decreases VBUS from 0 – 5.25 V to 0 – 1.37 V (the external VBUS divider ratio is 0.2626). [Figure 1](#) shows the schematic of the external divider; the upper resistor is $51.1\text{ k}\Omega$ and the lower resistor is $18.2\text{ k}\Omega$; the Schottky diode needs to tolerate $> 5.25\text{ V}$, so a higher rating ($> 8\text{ V}$) value is preferred. The current limit is approximately 8 mA in forward-bias mode and low on voltage ($\leq 0.7\text{ V}$) for a 3.3 V supply and 8 mA current.

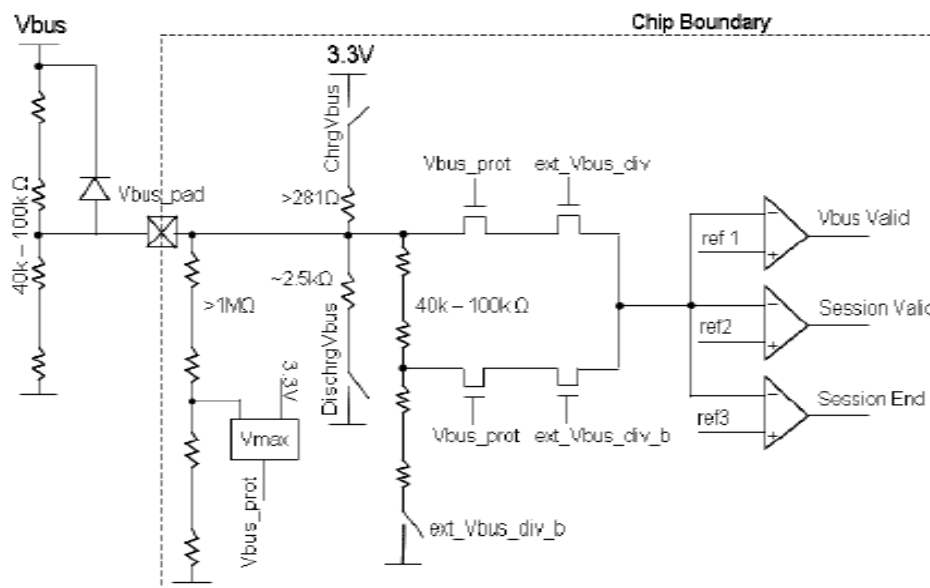


Figure 1. External Divider Schematic

2. Set EVDO[23] of USB_PHY_CTRL_FUNC to 1. This enables the external VBUS divider.

NOTE

The system cannot boot from USB in this case because the boot ROM code does not set the EVDO bit and hence both the internal and external VBUS dividers are enabled. Therefore, the device controller will not see a valid VBUS level and will not signal a connection to the host.

Case 2: The system is a standard host or standard device and does not support OTG operation. There are two options:

- Enable the internal divider. With this option, the voltages measured by the comparators reflect the levels on VBUS. The VBUS valid level will be detected around 4.2 V in this case. The leakage current will exist during VBUS transition from 3.3 V to 4.2 V.
- Disable internal divider. The VBUS levels are only defined in the USB OTG specification and only used in the OTG Session Request and Host negotiation protocols. Standard devices do not need to observe these VBUS levels. Therefore, VBUS detection levels do not have to be observed by these devices. It is sufficient to detect when VBUS is present. With the internal divider disabled, a valid VBUS level will be flagged at 1.1 V. At that point, the overvoltage protection devices of the comparators will limit the voltage. As a result, the leakage cannot occur because the internal VBUS level of the PHY never exceeds 3.3 V.

NOTE

The external divider is not needed in this case. The impedance mismatch that occurs when VBUS transitions from 3.3 V to 4.4 V does not affect USB operation and does not pose a reliability issue, unless the VBUS level remains in the 3.3 V–4.2 V range for an extended period of time.

Proposed Solution:

No fix scheduled

ENGcm09460 USB: USB PHY autoresume in host mode**Description:**

If the USB is put into low-power mode when no device or host is connected, the USB PHY incorrectly issues a RESUME signal when the USB OTG port is connected to a host. When no device or host is connected, the on-chip UTMI PHY always works in host mode. When the PHY detects a wake-up event in low-power mode, it automatically triggers a RESUME signal. This automatic RESUME function should be removed from the PHY.

This USB PHY issue only applies to device mode. The USB PHY works correctly in host mode when the USB is put into low-power mode (VBUS is not valid or USBCMD.RS bit is not set).

Projected Impact:

This can cause a bus-enumeration failure in USB, which means that the USB device is not recognized by the host. There are two scenarios that determine whether bus-enumeration failure happens:

- When a USB interrupt is used as the system wake-up source, enumeration failure does not occur if the USBCMD.RS bit is set to 1 immediately after the system resumes. Otherwise, it is treated as a low-speed (LS) device by the host.
- When a USB interrupt is not used as the system wake-up source, enumeration failure always occurs, because the system cannot resume even if there is a USB connection interrupt. In other words, the USB driver cannot be scheduled to respond to host enumeration.

Workarounds:

Case 1: The system needs to support wake-up from USB. There are two options.

- Software sets the USBCMD register RS bit immediately after the host detects USB wake-up interrupt.
- Software should set the USBCMD register RS bit before putting USB into low-power suspend mode (setting PORTSC.PHCD bit) when no device/host is connected.

With these two workarounds, a small data minus (DM) pulse can still be automatically issued by the PHY, but if the DM pulse is less than 2.5 μ s, the host does not treat it as a low-speed device connection event.

Case 2: The system does not need to support a wake-up from the USB. In this case, software can disable the USBEN bit (USB_PHY_CTRL_FUNC[24]) to avoid a false resume signal.

Proposed Solution:

No fix scheduled

ENGcm11802 USB: USB OTG generates extraneous pulse on DP on suspend**Description:**

When the OTG controller is operating in high-speed and software sets the PORTSC.SUSP bit to suspend the USB port, a 16.7 ns pulse is generated on the DP output of the OTG port.

Per USB 2.0 specification, the USB controller puts the PHY in Full Speed mode when the port is suspended. The output enabled of the PHY is switched off for one 60 MHz clock cycle after the transmitter switched to FS, causing it to drive a FS_J state on the bus for 1 clock cycle.

Projected Impact:

The pulse on DP has no impact on the functionality nor does it create any reliability issue.

- The pulse will only occur on the bus when the host controller suspends the port. At that time, the connected device will be in High Speed (HS) receiving mode. Therefore, there will be no bus contention.
- The connected device will not see the pulse as data. As there is no sync pattern, the device's PHY will not try to receive this as data.
- Normally, the connected device will not detect the pulse as bus activity, but even if it does, the device will suspend for a maximum of 125 μ s later. This is not a problem in the USB system.

Workarounds:

No workaround needed.

Proposed Solution:

No fix scheduled

ENGcm11270 WEIM: AUSx bits do not work for address bit A[23]**Description:**

The AUS bits in the WEIM Configuration Register (WCR) do not work for the address bus bit A[23]. The WEIM address bus most significant bits (ADDR[25:16], Address Bus MSB) are used for address bits [25:16]. If the corresponding AUSx bit (each WEIM chip select has a corresponding AUS bit) is set to 1 in the WCR register, then these MSB signals reflect the AHB address bits [25:16]. If the AUSx bit is set to 0, then these signals should represent AHB address bits [27:18] for word width memory, [26:17] for half-word width memory, and [25:16] for byte-width memory. The error is that when the AUSx bit is set to 1, the A[23] bit does not match the correct value of the corresponding AHB address bit.

Projected Impact:

This errata affects all Chip Select region. (that is CS0-CS5).

Cannot use the WEIM AUS feature to use un-shifted address mode if address bit A[23] is needed to address the external memory device.

NOTE

AUS feature is not only for ADDR[25:16] (Address Bus MSB), but also for ADDR[15:0] (Multiplexed Address Bus LSB).

Workarounds:

Set AUSx to 0, if address bit A[23] is needed to address the external device.

Proposed Solution:

No fix scheduled

ENGcm11409 WEIM: In FCE=1 mode, WEIM cannot correctly sample the data, if there is ECB asserted during burst access**Description:**

End current burst (WAIT). This active-low input signal ECB is asserted by external burst capable devices. It is serviced only in synchronous mode (SYNC=1). This signal can be used in the following modes depending on the EW bit in the Chip Select Control Register.

1. In the ECB mode (EW=0), ECB indicates the end of the current (continuous) burst sequence. Following assertion, the WEIM terminates the current burst sequence and initiates a new one.
2. In the WAIT mode (EW=1), the memory device asserts this signal to insert WAIT states during refresh collisions or during a row boundary crossing. Following assertion, the WEIM does not terminate the current burst sequence and continues it once WAIT is negated.

FCE is a parameter in the register CSCRxA that is used to enable or disable feedback clock.

- If FCE=0, WEIM samples the data by internal AHB bus clock.
- If FCE=1, WEIM samples the data by BCLK_FB signal that is from PAD.

If FCE is configured to 1 and there is ECB assertion during access, WEIM does not sample the correct data.

Projected Impact:

Cannot use FCE=1 mode when there is ECB assertion during access.

Workarounds:

If external device asserts ECB_B signal during burst access in FCE=1 mode, use FCE=0 mode instead of FCE=1 mode.

Proposed Solution:

No fix scheduled

ENGcm11891 WEIM: Unexpected deassertion when page mode emulation is used with non-32-bit wide port size**Description:**

There is an issue when the WEIM is used in page emulation mode with a port size that differs from 32 bit, that is, either 16 or 8 bits. Unexpected deassertion occurs on OE, LBA, and EB at word (external memory) boundaries within a 32 bit word (internal bus). For a 16 bit flash, deassertion occurs between the high 16 bit word and the low 16 bit word. The only way to avoid deassertion on these signals is to set OEA, LBA, and EBRA to zero.

Projected Impact:

This erratum affects all applications where the WEIM is used with page mode emulation and the port size is not 32-bits wide.

Workarounds:

If page mode emulation is used in 16 or 8 bit wide port size, then OEA, EBRA, and LBA must be set to 0.

Proposed Solution:

No fix scheduled

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Information in this document is provided solely to enable system and software implementers to use Freescale products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: freescale.com/SalesTermsandConditions.

Freescale and the Freescale logo are trademarks of Freescale Semiconductor, Inc. Reg. U.S. Pat. & Tm. Off. All other product or service names are the property of their respective owners. ARM is the registered trademark of ARM Limited. ARM9 is a trademark of ARM Limited.

© 2014 Freescale Semiconductor, Inc. All rights reserved.

Document Number: IMX25CE
Rev. 7.1
07/2014

