Freescale Semiconductor
Errata

Document Number: IMX35CE
Rev. 5, 05/2014

Chip Errata for the i.MX35

This document details all known silicon errata for the i.MX35. Table 1 provides a revision history for this document.

Table 1. Document Revision History

Rev. Number	Date	Substantive Change(s)
5	05/2014	• Added new PLL erratum, ERR007917. • Added note to USB erratum ENGcm09152.
4	09/2012	• Added a new IPU erratum, ENGcm11750. • Added a new SSI erratum, ENGcm11741. • Updated USB erratum, ENGcm09152, and added a new USB erratum, ENGcm11802. • Added a new WEIM erratum, ENGcm11889.
3	09/2010	Added the following two Errata to Table 3 on page 2: • ENGcm11601 • ENGcm11628
2	07/2010	Added the following two errata to Table 3 on page 2: • ENGcm11270 • ENGcm11409
1	01/12/2010	Updated Table 2 on page 2 to include all silicon revision 2.0 and 2.1part numbers.
0	05/11/2009	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

MCIMX35 Revision	Package	MCIMX35 Device Marking
2.0	17 mm x 17 mm	MCIMX351AVM4B
2.0	17 mm x 17 mm	MCIMX351AVM5B
2.0	17 mm x 17 mm	MCIMX353CVM5B
2.0	17 mm x 17 mm	MCIMX353DVM5B
2.0	17 mm x 17 mm	MCIMX355AVM4B
2.0	17 mm x 17 mm	MCIMX355AVM5B
2.0	17 mm x 17 mm	MCIMX356AVM4B
2.0	17 mm x 17 mm	MCIMX356AVM5B
2.0	17 mm x 17 mm	MCIMX357CVM5B
2.0	17 mm x 17 mm	MCIMX357DVM5B
2.1	17 mm x 17 mm	MCIMX351AJQ4C
2.1	17 mm x 17 mm	MCIMX351AJQ5C
2.1	17 mm x 17 mm	MCIMX353CJQ5C
2.1	17 mm x 17 mm	MCIMX353DJQ5C
2.1	17 mm x 17 mm	MCIMX355AJQ4C
2.1	17 mm x 17 mm	MCIMX355AJQ5C
2.1	17 mm x 17 mm	MCIMX356AJQ4C
2.1	17 mm x 17 mm	MCIMX356AJQ5C
2.1	17 mm x 17 mm	MCIMX357CJQ5C
2.1	17 mm x 17 mm	MCIMX357DJQ5C

Table 3 summarizes all known errata on the i.MX35 device, silicon revision 2.0 and 2.1. If silicon revision 2.0 is used, additional external filtering should be implemented as described in *Ensuring Data Integrity on the i.MX35 EMI* (EB717).

Table 3. Summary of Silicon Errata

Errata	Name	Solution	Page
ARM			
ENGcm09472	ARM: WFI and interrupt problems	No fix scheduled	5
Boot			
ENGcm08460	Boot: ATA boot failure	No fix scheduled	6
ENGcm08541	Boot: eMMC4.3 fast boot mode fails	No fix scheduled	7

Table 3. Summary of Silicon Errata (continued)

Errata	Name	Solution	Page
EMI			
TLSbo94318	EMI: No error response when a memory region outside of the EMI memory-mapped ranged is accessed	No fix scheduled	8
eSDHC			
ENGcm03648	eSDHC: eSDHC cannot send an interrupt to a card leaving the busy state at the end of an R1b response	No fix scheduled	9
ENGcm06704	eSDHC: Unable to issue CMD12 when the clock is stopped	No fix scheduled	10
ENGcm07207	eSDHC: CMD12 abort operation does not abort data transfer on AHB	No fix scheduled	11
ENGcm08334	eSDHC: Infinite block transfer mode not supported	No fix scheduled	12
ENGcm08539	eSDHC: DMA interrupt status bit cannot be cleared after a read operation	No fix scheduled	13
ENGcm08582	eSDHC: Cannot finish write operation after block gap stop	No fix scheduled	14
FlexCAN			
ENGcm09158	FlexCAN: Glitch filter is not implemented	No fix scheduled	15
GPU2D			
ENGcm11628	GPU2D: GPU2D locks up during burst cache flush	No fix scheduled	16
IPU			
ENGcm07151	IPU: Last pixel can be corrupt during a short starvation event	No fix scheduled	17
ENGcm09282	IPU: Cannot receive IPU ACK if CSI is still working when the OS goes into low power mode	No fix scheduled	18
ENGcm11750	IPU: Inputting signals to pads CSI_D8 through CSI_D11 may increase the observed jitter of the IPU display bit clock output	No fix scheduled	19
PLL			
ERR007917	PLL: PLL may not remain locked when configured with integer MF (MFN=0)	No fix scheduled	20
RTIC			
ENGcm05860	RTIC: Writing the SWRST (software reset) bit of the command register while the RTIC is performing a one-time hash operation can cause an error	No fix scheduled	21
TLSbo94838	RTIC: A hashing error can improperly cause a security violation	No fix scheduled	22
ENGcm05171	RTIC: Reading the RTIC command register (CMD) can cause a software reset	No fix scheduled	23
SSI			
ENGcm06222	SSI: Transmission does not take place in bit length early frame sync configuration	No fix scheduled	24
ENGcm06532	SSI: Incorrect data TX starts from FIFO1 when RX is enabled before TX in sync mode	No fix scheduled	25

Table 3. Summary of Silicon Errata (continued)

Errata	Name	Solution	Page
ENGcm11741	SSI: Excessive 4 bit shifts in AC97, Rx, 16-bit mode	No fix scheduled	26
USB			
ENGcm09459	USB: USB PHY auto-resume in host mode	No fix scheduled	27
ENGcm08321	USB: Host core wake-up failure when using internal serial PHY	No fix scheduled	28
ENGcm09152	USB: UTMI_USBPHY VBUS input impedance implementation error	No fix scheduled	29
ENGcm11601	USB: Core can lock up when a packet with less bytes than expected is received	No fix scheduled	31
ENGcm11802	USB: USB OTG generates extraneous pulse on DP on suspend	No fix scheduled	32
WEIM			
ENGcm11270	WEIM: AUSx bits do not work for address bit A[23]	No fix scheduled	33
ENGcm11409	WEIM: In FCE=1 mode, WEIM can not correctly sample data, if there is ECB asserted during burst access	No fix scheduled	34
ENGcm11889	WEIM: Unexpected deassertion when page mode emulation is used with non-32-bit wide port size	No fix scheduled	35

ENGcm09472 ARM: WFI and interrupt problems

Description:

There are two issues:

- The behavior of the FIQ signal to the ARM11 core can cause a problem when exiting WFI mode. The FIQ signal toggles after being initially asserted, which, as ARM has confirmed, is unexpected behavior to the ARM11 core. ARM has stated that this is not a fully validated case for their cores. This behavior occurs when core clocks continue to run and, along with particular caching and alignment schemes, can result in a corrupted cache line following a prefetch, as well as unexpected behavior in code. Also, the core can execute an instruction immediately following the WFI instruction before servicing the FIQ. This behavior of FIQ is caused by the design of the interrupt controller in the synchronization circuit.
- The same extra pulse on the FIQ signal can cause the core to execute instructions immediately following the WFI, before entering the ISR. If an ISR executes too quickly, the FIQ/IRQ may not clear by the time the core returns to main code, and may enter ISR two or more times for the same interrupt. This situation should only happen if the execution time of the code in the ISR that follows the initial write to the peripheral to clear the FIQ/IRQ, can execute in fewer than 25 hclk (AHB clock) cycles.

Projected Impact:

The first issue can result in a corrupted cache line following a prefetch, and thus unexpected behavior; the second issue can result in unexpected behavior of ISR execution.

Workarounds:

The WFI routine should change the clocking mode to a 1:1 (ARM:AHB) ratio. This must be ensured by following the programming with dummy reads. On wake-up, the clocks can then be changed back to the original ratio.

This completely prevents the toggle on the interrupt line, and this code can now be located in a cacheable region.

Example:

```
mov r0, #0
  ldr r1, =<clock_control_BASE>
  ldr r2, [r1, #OFFSET]
  orr r3, r2, #1TO1MODE
  str r3, [r1, #OFFSET]
... // Delay while switch to 1:1 occurs
mcr p15, 0, r0, c7, c0, 4   //WFI
  str r2, [r1, #OFFSET]
  bx lr
```

Proposed Solution:

No fix scheduled

ENGcm08460 Boot: ATA boot failure

Description:

P-ATA Boot fails.

Workarounds:

No Workaround available. Use other available ports for boot purposes like NOR, NAND, SD/MMC, and so on.

Proposed Solution:

No fix scheduled

ENGcm08541 Boot: eMMC4.3 fast boot mode fails

Description:

Cannot fast boot from an eMMC4.3 card. Normal boot from an eMMC4.3 card presents no problem.

Projected Impact:

eMMC4.3 fast boot mode fails.

Workarounds:

Use normal boot from eMMC4.3 card, and set the eMMC4.3 card to fast mode after boot.

Proposed Solution:

No fix scheduled

TLSbo94318 EMI: No error response when a memory region outside of the EMI memory-mapped ranged is accessed

Description:

When any module attempts to access external memory via a DMA type of transfer, it is important that a memory region outside of the EMI range is not accessed because the EMI does not have a way to signal back that a non accessible region has been read or written.

Projected Impact:

No error response when a memory region outside of the EMI memory-mapped ranged is accessed.

Reading from the reserved memory-map region can get uncertain data, while writing to a reserved memory-map region is ignored.

Workarounds:

Avoid accessing unmapped memory space.

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 you do not poll the DAT0 line 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.

Table 4. Workarounds for AutoCMD12 and R1b Polling Problem

Workaround No	Clock Rate	Process
Workaround 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.
Workaround 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.
Workaround 3	Any clock rate	1. Wait for the TC interrupt. 2. Software sends CMD13 iteratively until the card returns to the TRAN state.

Proposed Solution:

No fix scheduled

ENGcm06704 eSDHC: Unable to issue CMD12 when the clock is stopped

Description:

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

However, if the user generates a CMD12 (with XFERTYP:CMDTYP = 3, abort) command to stop the MULTI_BLOCK_READ or MULTI_BLOCK_WRITE transfer, in the eSDHC it appears as if the CMD12 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 is unable to issue a CMD12 command when the clock is stopped.

Workarounds:

Read the PRSSTAT[7] (SDOFF) status bit to check whether the SDCLK is stopped or not. If the SDCLK is stopped, make several accesses (write or read, depending on the current transfer direction) to the buffer to leave the over/under-run state until SDCLK is automatically restarted. Send CMD12.

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 that, 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 that 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 that when the card detects an error during a non-ending 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 time-out occurs. After the read-data time-out 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

ENGcm08334 eSDHC: Infinite block transfer mode not supported

Description:

The eSDHC does not support infinite block transfer mode. The i.MX35 reference manual notes that, to execute, infinite block transfer XFERTYPE[1] (BCEN) bit must be set to 0; however, this feature does not work.

Projected Impact:

The eSDHC does not support infinite block transfer mode.

Workarounds:

Use another block transfer mode. In other transfer modes, the maximum block number is 65536.

Proposed Solution:

No fix scheduled

ENGcm08539 eSDHC: DMA interrupt status bit cannot be cleared after a read operation

Description:

If HCLK is automatically gated off after a DMA read operation, the DMA Interrupt status cannot be cleared by software. HCLK is gated off earlier, before the DMA interrupt-acknowledge signal is negated.

Projected Impact:

Software cannot clear the DMA interrupt status bit (bit DINT of the Interrupt Status Register). This affects both SD memory and SDIO.

Workarounds:

There are two steps to the workaround:

1. Set the HCKEN (SYSCTL[1]) bit to 1 before starting a DMA read operation. This disables the HCLK auto-gating feature.
2. Wait until the DMA Interrupt is received and the Transfer Complete (IRQSTAT[1]) bit is set after a read operation is done, and then clear the HCKEN bit to re-enable the HCLK auto-gating feature.

Proposed Solution:

No fix scheduled

ENGcm08582 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

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

ENGcm11628 GPU2D: GPU2D locks up during burst cache flush

Description:

Under certain circumstances allowing the V3 block to continue processing commands while there is an ongoing burst cache flush will cause the z160 core to lock up. The only recovery from this condition is to reboot the processor.

Projected Impact:

This causes the GPU2D to hang until reboot.

Workarounds:

Use the V3 mark mechanism to halt command stream processing while waiting for the burst cache flush complete interrupt to be fired. Once the interrupt has been received, the mark count can be incremented and the stream processing can continue normally.

Proposed Solution:

No fix scheduled

ENGcm07151 IPU: Last pixel can be corrupt during a short starvation event

Description:

When the IPU drives a synchronous display like a dumb LCD, it must drive each pixel at a specific time. A situation where the IPU does not have data to be sent to the display is called starvation. This situation can happen as a result of a memory system overload. A short starvation is a situation where the data arrives few cycles later.

In the case of a short starvation, the IPU sends the last data on the bus to the display. After the IPU gets the correct data, the IPU sends it to the display at the correct time. This allows the IPU to overcome the starvation and keep the display synchronized. When the IPU is recovering from short starvation, only the last pixel is corrupted, and only if the following conditions occur:

1. Combining is being performed in a sync flow via the SDC
2. Short starvation occurs on the FG

The last pixel during the starvation period has a value that is different from the last data on the bus. This means that the IPU actually replaces one wrong value by another wrong value.

Projected Impact:

There is no obvious impact to image display quality on the screen. Because a short starvation may happen at a random time and on a random location of the screen, the effect is only in a single pixel, appears for a single frame, and is not noticeable.

Workarounds:

None. A short starvation can happen at a random time and on a random location of the screen, so it is unlikely that a user will notice. The effect is in a single pixel and appears for a single frame. Freescale verification is done against a bit-exact software model that reported a mismatch. In real world scenarios, the user does not notice this one-pixel random failure.

Proposed Solution:

No fix scheduled

ENGcm09282 IPU: Cannot receive IPU ACK if CSI is still working when the OS goes into low power mode

Description:

Data flow: input_data → camera interface (CSI) → pre-processor (PRP) → LCD

When the OS goes into suspend mode (low-power mode), IPU handshakes with the CCM to acknowledge a stop request if the IPU source clock HSP_CLK is not disabled (CCM CGR1 register CG9[19:18] does not equal 2'b00/2'b01/2'b10). When the CSI is busy dealing with data from a sensor, the IPU does not send an ACK back, and this causes an IPU handshake failure and low-power mode failure.

Only after detecting CSI_EOF, which indicates that the CSI has finished processing the current frame and is not busy at this moment), the IPU can successfully send an ACK to CCM and low-power mode can work.

Projected Impact:

This increases power consumption.

Workarounds:

When the OS wants to go into low-power mode, disable the IPU source clk (HSP_CLK) (CCM CGR1 register CG9[19:18] is equal to 2'b01 or 2'b10), the IPU does not do the handshake, and the OS successfully enters low-power mode.

Proposed Solution:

No fix scheduled

ENGcm11750 IPU: Inputting signals to pads CSI_D8 through CSI_D11 may increase the observed jitter of the IPU display bit clock output

Description:

Jitter observed on the IPU display clock output may be increased slightly when transitioning signals are present on pads CSI_D8 through CSI_D11. Due to the routing of these CSI pads through the package substrate and die surface to the IOMUX circuitry, coupling may occur within the i.MX35 device that in turn increases the jitter on the IPU display clock output. Because such coupling occurs before these traces reach the IOMUX circuitry, the IOMUX register settings for these pads have no effect on the increased jitter.

Projected Impact:

Usage of the pads CSI_D8 through CSI_D11 may cause the loss of PLL clock internal to an LVDS serializer device connected to the i.MX35 display peripheral, due to the increased jitter in the display bit clock output.

Workarounds:

In applications using an LVDS serializer device, these are the possible workarounds for this issue:

- Workaround 1: Avoid the use of pads CSI_D8 through CSI_D11 when the LVDS serializer is active.
- Workaround 2: Select an LVDS serializer device with an internal PLL tolerant of jitter on its bit clock input.
- Workaround 3: Use external circuitry to regenerate the IPU bit clock with a maximum jitter that meets the requirements of the LVDS serializer being used.

Proposed Solution:

No fix scheduled

ERR007917 PLL: PLL may not remain locked when configured with integer MF (MFN=0)

Description:

The PLL may not remain locked when configured with an integer value of MF, that is, when MFN is set to zero. Both the MPLL and PPLL are affected by this errata.

Projected Impact:

Impact of this errata may range from the PLL output stopping completely with loss of lock, or down stream clocks may exhibit excessive jitter as PLL loses and regains lock.

Workarounds:

To avoid loss, MFN of both PLLs should never be programmed to zero.

Proposed Solution:

No fix scheduled

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

Description:

If the RTIC is hashing data in the one-time hash mode, a write to the Software Reset bit, CMD[1] (SWRST), can cause the RTIC to generate an error and cause unexpected behavior.

Projected Impact:

This erratum can cause unexpected RTIC behavior.

Workarounds:

Do not write 1 to SWRST bit in the command register while RTIC is hashing data. Allow RTIC to finish the hashing operation, and then write 1 to the SWRST bit. If writing to the SWRST bit while the RTIC is hashing data causes the RTIC to generate an error, it 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

TLSbo94838 RTIC: A hashing error can improperly cause a security violation

Description:

There are two sets of control bits that cause RTIC to hash data. These are hashing data once, and hashing data continuously (run-time mode). During run-time mode, a hashing error generates a security violation (this is the definition of a run-time error).

If the run-time enable bits are set, but RTIC is given a hash-once command, an error can also incorrectly generate a security violation. In this case, RTIC should only set a status error, but not generate a security violation.

When RTIC generates a hashing error it is also supposed to generate a security violation if it is in run-time mode. This defect causes RTIC to generate a security violation if the run-time enable bit is set (rather than if it is in run-time mode).

Projected Impact:

This can cause RTIC to generate a security violation if the run-time enable bit is set (rather than if it is in run-time mode).

Workarounds:

Delay setting the run-time enable bits until RTIC is put into run-time mode. Do not set the run-time bits if RTIC will still be used for a hash-once operation.

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. Since this cannot be controlled, it is never safe to read the command register.

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, this module hangs if the following bits are programmed in sequential order:

1. SCR[0] (SSI_EN) 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 the SCR[1] (TXEN) and SCR[0] (SSI_EN) bits to 1 in the same frame. In other words, these two bits should be set to 1 during the same register write.

Proposed Solution:

No fix scheduled

ENGcm06532 SSI: Incorrect data TX starts from FIFO1 when RX is enabled before TX in sync mode

Description:

The SSI module is configured as follows:

- SSI module is operating in network synchronous mode (SCR[4], SYN & SCR[3], NET bits are set).
- 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 if, after the passage of one or more frames, the TX_EN bit (SCR[1]) is set; 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

ENGcm11741 SSI: Excessive 4 bit shifts in AC97, Rx, 16-bit mode

Description:

In AC97, 16-bit mode, the Rx data is received in bits [19:4] of Rx FIFO, 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

ENGcm09459 USB: USB PHY auto-resume 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, the USB device 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 the 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 2.5µs, 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

ENGcm08321 USB: Host core wake-up failure 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 it 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:

- Workaround 1: The USB 2.0 specification does not enforce the turning off of 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 USB FS HOST clock is off.
- Workaround 2: When USB wake-up events can be detected (that is, the 60 MHz clock for the USB host controller is turned 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 kΩ–100 kΩ 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 kΩ 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 kΩ and the lower resistor is 18.2 kΩ; the Schottky diode needs to tolerate > 5.25 V, so a higher rating (> 8 V) value is preferred. The current limit is approximately 8 mA in forward-bias mode and low on voltage (<= 0.7 V) for a 3.3 V supply and 8 mA current.

NOTE

The Schottky diode is present to support the VBUS pulsing method of the Session Request Protocol (USB OTG specification rev. 1.1). VBUS pulsing is no longer defined in the current (rev.2) OTG specification, hence the diode is no longer needed.

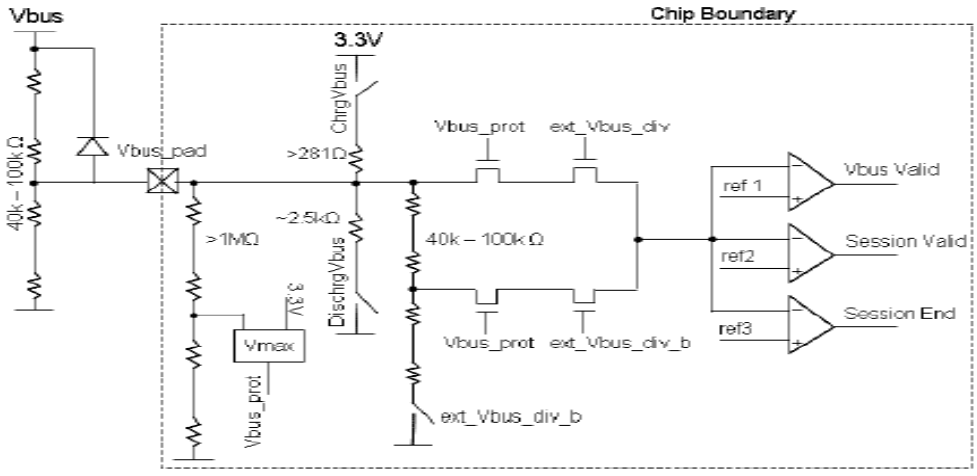

Figure 1. External Divider Schematic

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

ENGcm11601 USB: Core can lock up when a packet with less bytes than expected is received

Description:

There is an issue in the USB's DMA controller & AHB bus interface that can cause the DMA state machine to lock up when an AHB bus master with a higher priority requests the bus at the time when the last word is transferred from the USB FIFO to memory.

The USB controllers only lock-up when the packet data is less than the requested size. The packet data is fully transferred to main memory and the packet is acknowledged, but the transfer descriptor remains active and the controller does not respond to further requests.

The lock up can only occur when all of the following conditions are true:

- SBUSCFG register is set to 1, 2 or 3 (fixed burst length transfers with SINGE for non-burst).
- The actual USB transfer has less bytes than requested (total byte count in qTD is not 0 at the end of the transfer).
- The last word of the packet is written from the FIFO to memory using a bus cycle of type SINGLE.
- An AHB bus master with a higher priority requests the bus in the last AHB bus cycle.

Projected Impact:

AHB bus cycles of type SINGLE cannot be used for USB data transfer. No impact to other modules.

Workarounds:

Because the lock up can only occur when the AHB Bus cycle is of type SINGLE, the issue can be avoided by using Unspecified burst length (INCR) accesses instead of type SINGLE. Register SBUSCFG (offset 0x0090) should be set to 0, 5, 6 or 7.

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 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 µs 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 can not correctly sample 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 in synchronous mode only (SYNC=1).This signal can be used in two different modes depending on the EW bit in the Chip Select Control Register. 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 initiate a new one. 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 one parameter in the register CSCRxA that is used to enable/disable feedback clock.

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

The issue is found that, if FCE is configured to 1, and there is ECB assertion during access, WEIM will not sample the correct data.

Projected Impact:

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

Workarounds:

Use FCE=0 mode instead of FCE=1 mode, if external device will assert ECB_B signal during burst access in FCE=1 mode.

Proposed Solution:

No fix scheduled

ENGcm11889 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.

Document Number: IMX35CE
Rev. 5
05/2014