Document Number: IMX23CE
Rev. 4, 01/2013

Chip Errata for the IMX23

This document details all known silicon errata for the i.MX23. Table 1 provides a revision history for this document.

Table 1. Document Revision History

Rev. Number	Date	Substantive Change(s)
4	01/2013	New erratum: 5838.
3	09/2011	Added TA5, and updated the workaround for erratum 2836.
2	04/2010	New erratum: 2349
1	11/2009	New errata: 5835, 5837
0	09/2009	Initial release. New errata: 2229, 2725, 2727, 2745, 2747, 2748, 2749, 2752, 2765, 2783, 2786, 2798, 2811, 2814, 2815, 2816, 2829, 2836, 2847, 2854, 2858, 4559, 4733

Table 2 provides a cross-reference to match the revision code in the processor version register to the revision level marked on the part.

Table 2. Revision Level to Part Marking Cross-Reference

i.MX23 Revision	i.MX23 Revision Number	Device Marking	Processor Version Register (HW_DIGCTRL_CHIPID_REVISION)
TA4	1.3	MCIMX233DJM4B MCIMX233CJM4B MCIMX233DAG4B MCIMX233CAG4B	0x3
TA5	1.4	MCIMX233DJM4C MCIMX233CJM4C MCIMX233DAG4C MCIMX233CAG4C	0x4

Table 3 summarizes all known errata and lists the corresponding silicon revision level to which it applies. A 'Y' entry indicates the erratum applies to a particular revision level, while a '—' entry means it does not apply.

Table 3. Summary of Silicon Errata and Applicable Revision

Errata	Name	Silicon Rev. TA4[1]	Silicon Rev. TA5[2]
2229	Insufficient SD/MMC read delay leads to premature timeouts and can cause some cards to fail booting.	Y	—
2349	The DAC DMA halts unexpectedly	Y	Y
2725	Abort when the CPU accesses external memory while it is in self-refresh mode does not seem to work properly.	Y	Y
2727	I2C 9th Clock Pulse (ACK) not generated when RETAIN_CLOCK set.	Y	Y
2745	EMI interrupts are not available.	Y	Y
2747	Undetectable data corruption can occur during IN transfer in isochronous mode.	Y	Y
2748	AMBA-AHB Estimator clock crossing issue.	Y	Y
2749	USB: Port change interrupt not generated.	Y	Y
2752	PXP YBASE Calculation Incomplete	Y	Y
2765	Chip locks up when switching to synchronous mode if ref_cpu is not active.	Y	Y
2783	Audioout_ctrl_edge_sync bit is not used (audioin_ctrl_edge_sync is used for ADC and DAC)	Y	Y
2786	JTAG crash when HCLK is less than 24 MHz.	Y	Y
2798	Phase error between left/right ADC channels.	Y	Y
2811	Unreliability of VDD5V_GT_VDDIO functionality	Y	Y
2814	The DCDC converters inadvertently turn on when 5 V is removed while the DCDC_XFER bit is clear.	Y	Y
2815	The hardware functionality to disable battery brownouts when 5 V is present always does NOT follow the VBUSVALID_5VDETECT value.	Y	Y

Table 3. Summary of Silicon Errata and Applicable Revision (continued)

Errata	Name	Silicon Rev. TA4[1]	Silicon Rev. TA5[2]
2816	The PWDN_5VBRNOUT functionality trips during certain events even though 5 V voltage is valid.	Y	Y
2829	HW_I2C_CTRL1_OVERSIZE_XFER_TERM_IRQ set at end of every transaction	Y	Y
2836	APBH/APBX do not flush out the remaining FIFO data after a DMA termination	Y	Y
2847	BCH soft reset may cause bus master lock up.	Y	Y
2854	The DCDC fails to shut down when the battery is below the battery brownout level and PWD_BATTBRNOUT is set, due to the presence of a 5 V supply.	Y	Y
2858	USB controller may access a wrong address for the dTD (endpoint transfer descriptor) and then hang.	Y	Y
4559	ROM Fails to Boot from BA-NAND	Y	Y
4733	SD Boot does not support MMC with CSD_STRUCTURE version in EXT_CSD register	Y	—
5835	5 V disconnections can cause large droops on the VDDIO and VDDD supply rails under certain operating conditions	Y	Y
5837	Setting the ENABLE_DCDC bit in the HW_POWER_DCDC4P2 or HW_POWER_5VCTRL registers can result in false brownout detection.	Y	Y
5838	SAIF2 DMA completion IRQ in the APBX DMA module is not connected to the interrupt collector	Y	Y

1 TA4 is equivalent to revision 1.3.

2 TA5 is equivalent to revision 1.4.

2229 Insufficient SD/MMC read delay leads to premature timeouts and can cause some cards to fail booting.

Description:

The SD/MMCC read delay being used by ROM is not sufficient, which leads to premature timeouts that cause some cards to fail booting. According to the SD specifications, a minimum of 100 ms or one hundred times the "typical delay" should be used. In a failing SD cards case, the "typical delay" calculated from the TAAC and NSAC fields is 1.5 ms. So the read response delay for this card should have been 100 ms, but ROM was using a ~4 ms delay.

After debugging in ROM, a particular failing card started working when the delay was modified to 13 ms. Four out of five 2 GB Sandisk blue SD cards tested could not boot.

This issue is only reproducible with SD 2.0, 2 GB devices, and is not reproducible with MMC, eMMC, and older SD 1 GB devices.

Workaround:

The only workaround is to patch ROM using other bootable devices like I^2C EEPROM, SPI EEPROM, and so on. The patch should first call Init of the ROM SSP driver, then update the pSspContext->timeout value to 1000000 (1 sec timeout).

This patch binary image is available on Freescale.com. It is called “IMX23 EVK SD ROM PATCH”, with a description of “EEPROM Patch Workaround for Errata #2229”.

2349 The DAC DMA halts unexpectedly

Description:

Some devices can experience an issue where audio output from the DAC halts unexpectedly. This occurs because of stopping and starting the DAC output several times.

A device having such an issue may show one of the following symptoms when a playback is attempted:

- no audio output (as the DAC DMA is halted)
- audio output is available at the beginning of playback for few seconds and then it is disabled (as the DAC DMA is halted)

Workaround:

Set the HW_AUDIOOUT_CTRL DMAWAIT_COUNT to 31 (recommended to be set during the peripheral initialization) whenever the AUDIOOUT uses the DMA.

There are no known side effects for this workaround.

2725 Abort when the CPU accesses external memory while it is in self-refresh mode does not seem to work properly.

Description:

If the EMI is in self-refresh mode, any external data accesses to the memory cause the system to hang. External access cannot be completed when the EMI is in this mode, so the accesses stall the system bus.

Workaround:

Do not allow external data access of the system bus when the EMI is in self refresh mode. Software must enforce this scenario.

2727 I2C 9th Clock Pulse (ACK) not generated when RETAIN_CLOCK set.

Description:

When RETAIN_CLOCK is set, the ninth clock pulse (ACK) is not generated. However, the SDA line is read at the proper timing interval. If RETAIN_CLOCK is cleared, the ninth clock pulse is generated.

Also, the HW_I2C_VERSION register incorrectly states the version is 1.2. It should be 1.3.

Workaround:

HW_I2C_CTRL1[ACK_MODE] has default value of 0. It should be set to 1 to enable the fix for this issue.

2745 EMI interrupts are not available.

Description:

The i.MX23 memory controller is capable of generating an interrupt under the following conditions:

- A single access outside the defined physical memory space is detected
- Multiple accesses outside the defined memory space are detected.
- DRAM initializations are complete
- DLL unlock condition is detected.

This interrupt is not used with the i.MX23 interrupt controller.

Workaround:

For software to detect and react to any of the four controller interrupt conditions, the software must read DRAM control register 16 and examine the values in the int_status field. Typically, however, this feature should have minimal value in a system with memory management, where physical memory mappings can be controlled via page tables. As such, Freescale does not recommend using these status bits or interrupts for the purpose of physical memory range control.

2747 Undetectable data corruption can occur during IN transfer in isochronous mode.

Description:

Undetectable data corruption can occur during IN transfer in isochronous mode.

Workaround:

There is a safe window for the software to prime the isochronous endpoint for an IN transfer that prevents the occurrence of this erratum.

For high-speed, this window is impossibly small. Do not use high-speed isochronous endpoints in high-speed.

However, for full-speed, the window is more reasonable. This time window corresponds to the amount of time in a frame allocated for the asynchronous schedule. The USB specification requires that no more than 90% of any frame be allocated for periodic (isochronous and interrupt) full-/low-speed transfers, so in the worst case the last 100 µs of the FS frame will be allocated for the asynchronous schedule. If the software primes the endpoint from 900 µs to 999 µs after receiving an SOF interrupt, this erratum never occurs.

2748 AMBA-AHB Estimator clock crossing issue.

Description:

USB internal AHB bus transfers can be lost when using anything other than AHB unspecified length incrementing burst types.

Workaround:

Only use type unspecified INCR bursts, which is guaranteed by this setting:

HW_USBCTRL_SBUSCFG[AHBBRST] = 0x0 (this is the default reset value)

This type of burst mode results in the highest external memory efficiency for the i.MX23.

2749 USB: Port change interrupt not generated.

Description:

In self-powered device mode, the USB controller may not generate a port change interrupt when VBUS goes away.

Workaround:

Use VBUSVALID interrupts in POWER module. Set POLARITY_VBUSVALID to 0x0 to check for 5 V disconnected. See Chapter 32, “Power Supply,” in the *i.MX23 Applications Processor Reference Manual* (document number IMX23RM), for details.

2752 PXP YBASE Calculation Incomplete

Description:

Chapter 17, “Pixel Pipeline,” in the *i.MX23 Applications Processor Reference Manual* (order # IMX23RM), states the following: “The [crop] XBASE and YBASE provide the coordinates of the first block to be displayed from the source image and the WIDTH and HEIGHT parameters specify an effective size of the resulting image in the output buffer.”

This statement is correct for non-interlaced, and interlaced_output. However, when doing interlacing input to one field, the crop_ybase value gets shifted by 2 to match the size of the new image. This results in an incorrect index into the original S0 image.

Workaround:

To get the correct location for cropping while doing interlaced input, the value of the crop_ybase register needs to be doubled to get the desired index location into the S0 image.

2765 Chip locks up when switching to synchronous mode if ref_cpu is not active.

Description:

In order to switch the EMI clock from asynchronous to synchronous mode, both the xtal_ref and the cpu_ref must be active, even if both BYPASS_CPU and BYPASS_EMI are set in the HW_CLKCTRL_CLKSEQ register.

Workaround:

If cpu_ref is inactive, then activate the cpu_ref before attempting to switch to synchronous mode.

2783 Audioout_ctrl_edge_sync bit is not used (audioin_ctrl_edge_sync is used for ADC and DAC)

Description:

The EDGE_SYNC bit in AUDIOOUT_CTRL register has no effect on the DAC.

Workaround:

The EDGE_SYNC bit in AUDIOIN_CTRL register should be used for both DAC and ADC.

2786 JTAG crash when HCLK is less than 24 MHz.

Description:

The chip can appear to “crash” when using JTAG to debug code and the clocks are changed such that HCLK is set to less than 24 MHz. The “crash” is not actually a chip problem in that the chip does continue to run. But it is no longer possible to halt or stop at a breakpoint via the debugger, so it seems that the chip has died.

Workaround:

The workaround is to set the debugger JTAG target clock to 700 kHz or lower, depending on how low HCLK is set. The lower HCLK is set, the lower the JTAG target clock must be set. For example, a JTAG target clock setting of 100 kHz is recommended for HCLK at 3 MHz. If HCLK is 12 MHz, 700 kHz is recommended. Where HCLK is from 12 MHz up to 24 MHz, no higher than 700 kHz is recommended.

This is only required when using JTAG to debug portions of code where HCLK can be below 24 MHz (stopping at breakpoints, stepping, and so on). For downloading code via JTAG or debugging where HCLK is at or above 24 MHz, the JTAG target clock can be at 1 MHz.

2798 Phase error between left/right ADC channels.

Description:

There is a phase error of 1/2 sample between the left and right ADC (62.5 µs at 8 KSPS and 11.34 µs at 44.1 KSPS).

Workaround:

The phase error can be minimized (but not eliminated) by running at a higher sample rate.

2811 Unreliability of VDD5V_GT_VDDIO functionality

Description:

Due to unreliability of the VDD5V_GT_VDDIO functionality, the chip should default to using VBUSVALID detection and only boot up until the default VBUSVALID_TRSH level has been reached. Otherwise, software can start running before a reliable state has been reached. Currently, software has to handle this case by spinning in a loop, waiting for one of the following conditions to occur:

- VBUSVALID level is reached
- A good battery voltage plus PSWITCH is pressed
- RTC_ALARM is true
- AUTO_RESTART bit is true

Workaround:

There are inherent weaknesses with using the VDD5V_GT_VDDIO method as the hardware 5 V detection method. This method depends on a fixed offset between VDDIO and the VDD5V voltage. So the VDD5V_GT_VDDIO status changes as the VDDIO level changes. Secondly, as with any linear regulator, the VDDIO linear regulator supply strength is based on its supply voltage, which is VDD5V. As the VDD5V level decreases, so does the VDDIO supply strength. If the VDDIO rail has too much load and the VDD5V level decreases, the VDDIO output voltage can decrease as well. If the VDDIO voltage drops with VDD5V, then a case could exist in which VDD5V_GT_VDDIO never becomes false. One method for countering this particular weakness is to enable and properly handle VDDIO brownouts. However, this method produces other complications such as causing device shutdown on a 5 V disconnect in some cases where the device could have performed a 5 V to battery transfer and avoided a shutdown. Or this method could also necessitate adding functionality to the VDDIO interrupt handler to differentiate between brownouts caused by 5 V disconnects versus not.

To avoid these weaknesses with the VDD5V_GT_VDDIO functionality, the hardware can be programmed to use the VBUSVALID functionality for the 5 V status by setting the VBUSVALID_5VDETECT bit. However, great care must be taken to coordinate this change properly.

Keep in mind that the PWD_BATTBRNOUT functionality is broken and always uses VDD5V_GT_VDDIO as an internal enable/disable mechanism (see erratum 2815). A problem can occur when the 5 V detection is changed from VDD5V_GT_VDDIO to VBUSVALID if the VDD5V_GT_VDDIO status is true but the VBUSVALID_TRSH level being used is higher than the VDD5V level. If this occurs, then the VBUSVALID status being false will cause the system to believe that a 5 V disconnection has occurred. A hardware 5 V disconnection event causes the DCDC to automatically turn on, regardless of the battery voltage level, which could result in the DCDC operating from a voltage lower than the minimum operating voltage specified in Chapter 2, “Characteristics and Specifications,” in the *i.MX23 Applications Processor Reference Manual* (order # IMX23RM). This, in turn, could result in physical damage to the DCDC converter itself. To avoid this situation, always be sure that the VDD5V voltage level is higher than the

VBUSVALID_TRSH level programmed. Likewise, when raising the VBUSVALID_TRSH from one level to another, be sure that the VDD5V is higher than the target level.

2814 The DCDC converters inadvertently turn on when 5 V is removed while the DCDC_XFER bit is clear.

Description:

If the DCDC_XFER bit is clear, the DCDC convert should not automatically turn on when 5 V is removed. Instead, a power down should occur if 5 V is removed, and DCDC_XFER and ENABLE_DCDC are zero.

Workaround:

Because the DCDC will automatically be enabled by hardware once 5 V is removed, proper protection must be configured so the device shuts down if the battery voltage is below the minimum voltage limit, as specified in Chapter 2, “Characteristics and Specifications,” in the *i.MX23 Applications Processor Reference Manual* (order # IMX23RM). To do this, leave the PWDN_5VBRNOUT bit set until sufficient battery voltage is present. Then, the software can properly enable the power supply to prepare for a DCDC transfer on the loss of 5 V. After that preparation is complete, software can clear the PWDN_5VBRNOUT bit.

2815 The hardware functionality to disable battery brownouts when 5 V is present always does NOT follow the VBUSVALID_5VDETECT value.

Description:

The hardware functionality to disable battery brownouts when 5 V is present does NOT follow the VBUSVALID_5VDETECT value. However, it should in order to coordinate with the rest of the hardware (and software).

Workaround:

The automatic hardware functionality to disable hardware battery brownout shutdowns once 5 V is connected must use VDD5V_GT_VDDIO instead of following the VBUSVALID_5VDETECT configuration. The developer should take this into consideration when developing the power protection scheme. At some point in the application, software handling of battery brownouts is enabled, and hardware battery brownouts are enabled, which makes the effects of this problem go away.

2816 The PWDN_5VBRNOUT functionality trips during certain events even though 5 V voltage is valid.

Description:

The PWDN_5VBRNOUT bit will incorrectly power down the chip when specific power bits are modified. The 5 V brownout comparator is injected with a small amount of noise when these bits are set, causing the output to glitch momentarily.

Workaround:

1. Save the current value of HW_POWER_5VCTRL__PWDN_5VBRNOUT.
2. Clear HW_POWER_5VCTRL__PWDN_5VBRNOUT.
3. Modify HW_POWER_5VCTRL__ENABLE_DCDC.
4. Restore value of HW_POWER_5VCTRL__PWDN_5VBRNOUT.

2829 HW_I2C_CTRL1_OVERSIZE_XFER_TERM_IRQ set at end of every transaction

Description:

The HW_I2C_CTRL1_OVERSIZE_XFER_TERM_IRQ is set at the end of every transaction. This IRQ is not used in master mode and therefore should be ignored.

NOTE

If HW_I2C_CTRL1_OVERSIZE_XFER_TERM_EN is set, the HW_I2C_STAT_ANY_ENABLED_IRQ is also set at the end of any MASTER transaction since HW_I2C_CTRL1_OVERSIZE_XFER_TERM_IRQ will be high.

Workaround:

Ignore the HW_I2C_CTRL1_OVERSIZE_XFER_TERM_IRQ bit and do not set HW_I2C_CTRL1_OVERSIZE_XFER_TERM_EN.

2836 APBH/APBX do not flush out the remaining FIFO data after a DMA termination

Description:

Both APBX and APBH bridge DMAs are affected by this erratum. The issue occurs when a peripheral terminates a DMA transaction abruptly without sending all the data the corresponding DMA channel was set up to receive. There is a possibility of leftover data remaining in the DMA channel's FIFO; that data is not written into memory and is lost.

For example, if a timeout occurs in UART RX, it issues a dma_terminate to the corresponding APBX DMA channel. If the DMA channel does not receive all the data that it was set up to receive, and there was data lingering in the DMA channel's FIFO, the remaining data is not written to memory.

Workaround:

A possible workaround is to disable the dma_terminate from the peripheral, switch the DMA source to allow current descriptor to finish, and then, manually fetch the data left in the peripheral buffer and copy it into DMA receive buffer. This workaround involves the following steps (assuming that a DMA termination occurred due to a timeout in UART RX:

1. Set RXTIMEOUT in HW_UARTAPP_CTRL0 to a value larger than 0. This will enable the timeout detection in the UART RX block.
2. Clear RXTO_ENABLE bit in HW_UARTAPP_CTRL0 to disable the dma_terminate event from UART timeout. Note that the UART RX will still happen, and it will not send terminate to DMA.
3. Enable RX timeout interrupt by setting RTIEN bit in HW_UARTAPP_INTR register.
4. Initialize and start RX DMA descriptor, there is no limitation on the DMA side. For example, buffer length can be 4K, any buffer address can be taken, and chain can also be used.
5. When there is a timeout on UART RX, there will be an interrupt from UART block.
6. Read HW_APBX_CHn_DEBUG2 to get the APB_BYTES, and read HW_APBX_CHn_CMD to get XFER_COUNT, the bytes received from UART by DMA in this decriptor is XFER_COUNT-APB_BYTES.
7. Read RXFE bit in HW_UARTAPP_STAT to see if there is still any data inside UART FIFO. If RXFE is not 0, read HW_UARTAPP_DATA to read the data in the UART RX FIFO.
8. Read RXBYTE_INVALID bits in HW_UARTAPP_STAT to check which byte(s) is valid in the 32-bit word read from HW_UARTAPP_DATA, then save the valid byte(s) to a temporary buffer. It is possible that 1/2/3 bytes are valid.
9. Set RX_SOURCE in HW_UARTAPP_CTRL0 to 1. After this bit is set, the DMA will read the STAT register instead of DATA register, repeatedly until XFER_COUNT for this DMA descriptor is reached.
10. When the DMA complete IRQ comes, all the data inside DMA FIFO has been flushed to DMA receive buffer in the memory.
11. Copy the byte(s) from the temporary buffer to DMA receive buffer. The length can be calculated from RXBYTE_INVALID, and the offset can be taken from XFER_COUNT-APB_BYTES.

A software implementation of this workaround under Linux is available in the Linux source code patch package, titled “LINUX_1005_PATCHS”, which is available on www.freescale.com. The specific patch with this workaround implementation is called “AUART:SW workaround implemented”.

2847 BCH soft reset may cause bus master lock up.

Description:

The BCH bus master may lock up if the soft reset bit (HW_BCH_CTRL_SFTRST) is set. HW_BCH_CTRL_CLKGATE may be set and cleared as desired to clock gate the block, as long as HW_BCH_CTRL_SFTRST is never set.

Workaround:

Do not assert HW_BCH_CTRL_SFTRST. To reset the register set, write desired values to the BCH registers.

2854 The DCDC fails to shut down when the battery is below the battery brownout level and PWD_BATTBRNOUT is set, due to the presence of a 5 V supply.

Description:

If the DCDC is trying to source off a depleted battery while PWD_BATTBRNOUT is set, the DCDC should shut down.

Workaround:

The software workaround is to use the software battery brownout.

2858 USB controller may access a wrong address for the dTD (endpoint transfer descriptor) and then hang.

Description:

Currently, software checks the Active bit in dTD to see whether it is finished. If the Active bit is 0, then software frees the allocated memory for the dTD.

Hardware sequence after all data of a dTD is transferred is as follows:

1. Update dTD. (This includes three DWs AHB write access. The Active bit is cleared in the first DW write.)
2. Update qHead (this includes three DWs AHB write access)
3. Read the dTD again to check if software added a new dTD (this is a SINGLE AHB read). At the same time, send out an interrupt if needed.

After step 1, if software finds the Active bit is cleared, then the dTD memory space is freed and this space may be allocated for other thread use. In step 3, hardware may get a wrong dTD.

This issue does not occur if some delay is added before freeing the dTD memory space.

This issue only occurs in USB INCR8 mode, because steps 1 and 2 have 6 SINGLE AHB transfers in INC8 mode, but only 2 burst AHB transfers in INCR mode.

This issue only occurs when the dTD list is used, because if only one dTD is used, the software only checks the Active bit after an interrupt is received (step 3). But when the dTD list is used, the software may check the entire list after the interrupt for the first dTD is received, and hardware just finished the transfer of the second dTD at this time.

Figure 1 illustrates this issue:

Figure 1. dTD Issue Sequence of Events

Workaround:

Postpone freeing the current dTD; free it when its next dTD can be freed, so the last completed dTD (followed by an ACTIVE dTD) is always freed when the next IOC irq comes.

4559 ROM Fails to Boot from BA-NAND

Description:

In ROM code, the definition of the data structure returned by ONFi BA-NAND's parameters page command is not enclosed within #pragma pack(1) compiler directive. As such, data read from the BA-NAND does not correspond to actual fields in the data structure, causing improper values in the m_u16SectorSize field. Due to an incorrect sector size calculation in ROM, boot from BA-NAND fails. The ROM executes the following steps after reading the parameter page's data:

1. It reads the first 2 KB of data present on sector 0 by issuing start lba address 0. Each lba is 512 bytes. The first sector contains the MBR.

2. Micron's BA-NAND sector size is 4 KB. To skip the next 2 K of sector 0, ROM sends an abort cmd to the BA-NAND device.

3. Next, ROM will prepare to read the config block present at sector 1 or lba 8.

4. ROM will calculate the next sector address in lba units. However, because the sector size is incorrect from the parameters page, an incorrect lba address for sector 1 will result. Reading from this incorrect address results in unidentified data and the boot will fail.

Workaround:

To work around this issue, a simple patch should be present on the first 2 K of firmware data that should be loaded first and called. The patch should do the following to fix the problem.

```
MyNandItf.BANandHalAbort(0); // to abort reading next 2K of first sector
pNandContext->zSerializerState.m_u32PageDataSize = 4096; // fix the sector size
pNandContext->zSerializerState.m_u32SectorDataRead = 0; // initialize to 0
```

The actual firmware should be programmed starting at the next 4 K from the beginning of the firmware area.

4733 SD Boot does not support MMC with CSD_STRUCTURE version in EXT_CSD register

Description:

The ROM is not able to boot an eMMC with the CSD_STRUCTURE version in the EXT_CSD register. See the code in sd_ddi.c. It should check for CSD_STRUCTURE==3 and then look in the EXT_CSD register for the actual CSD_STRUCTURE version. See the MMC specification, v. 4.2 or greater.

Workaround:

The only workaround is to patch ROM by providing the fix in other boot mode devices like I^2C EEPROM, or SPI EEPROM, and so on.

5835 5 V disconnections can cause large droops on the VDDIO and VDDD supply rails under certain operating conditions

Description:

When 5 V is disconnected, the DCDC does not switch its input from the VDD4P2 source to the battery quickly enough to prevent the DCDC output from drooping under heavy loads. One condition that exhibits the problem is a disconnect from 5 V with a 3.5 V battery voltage and a ~250 mA load on VDDIO.

Workaround:

To prevent a significant dip, the VDD4P2 must switch off as soon as possible when 5 V is lost to alert the DCDC to use battery power. To do this, configure the VDD5V_DROOP interrupt as an FIQ and set the droop interrupt threshold to a voltage level above the VBUSVALID threshold. Inside the VDD5V_DROOP handler, clear the ENABLE_DCDC and ENABLE_4P2 bits in the HW_POWER_DCDC4P2 register.

5837 Setting the ENABLE_DCDC bit in the HW_POWER_DCDC4P2 or HW_POWER_5VCTRL registers can result in false brownout detection

Description:

When the ENABLE_DCDC bit in HW_POWER_DCDC4P2 is set, a glitch is propagated through the brownout comparators. If the glitch is sufficiently large, it can cause a false brownout detection. The VDDD, VDDA, VDDIO, and VBUSVALID comparators are all susceptible to the glitch.

Workaround:

The sequence below is needed to work around this issue prior to setting the ENABLE_DCDC bit in HW_POWER_DCDC4P2:

1. Disable the power rail brownout interrupts (clear HW_POWER_CTRL VDDA, VDDD, VDDIO ENIRQ bits).
2. Set the HW_POWER_5VCTRL PWRUP_VBUS_CMPS bit (may already be set).
3. Set the HW_POWER_5VCTRL VBUSVALID_5VDETECT bit to 0.
4. Set the HW_POWER_5VCTRL VBUSVALID_TRSH to 0x0 (2.9 V).
5. Disable VBUSDROOP status and interrupts (clear VDD5V_DROOP_IRQ).
6. Set the ENABLE_DCDC bit in HW_POWER_DCDC4P2.
7. Wait 100 µs.
8. Check VBUSVALID_IRQ bit. If it is set, then set and clear the PWD_CHARGE_4P2 bit to repower on the 4P2 regulator because it is automatically shut off on a VBUSVALID false condition. It may be helpful to ramp up the CHARGE_4P2_ILIMIT value at this point to gradually draw power from 5 V rail. If HW_POWER_5VCTRL ENABLE_DCDC is already set, the DCDC will we sourcing charge from 4P2 as soon as PWD_CHARGE_4P2 is toggled.
9. Clear VBUSDROOP, VBUSVALID, and the output rails IRQ bits as needed.
10. Restore the output rail ENIRQ bits, the VBUSVALID_THRSH level, VBUSVALID_DETECT value, VBUSVALID ENIRQ, and VBUSDROOP ENIRQ to their original values.

The sequence below is needed to work-around this issue prior to setting the ENABLE_DCDC bit in HW_POWER_5VCTRL. Note that the below workaround assumes the usual requirements for setting the HW_POWER_5VCTRL ENABLE_DCDC bit are met (that is, the hardware and/or software battery brownout protection mechanism is enabled to properly protect the system against the DCDC sourcing from the battery if the voltage if the battery voltage is too low.).

1. Disable the power rail brownout interrupts (clear HW_POWER_CTRL VDDA, VDDD, VDDIO ENIRQ bits).
2. Set the HW_POWER_5VCTRL PWRUP_VBUS_CMPS bit (may already be set).
3. Set the HW_POWER_5VCTRL VBUSVALID_5VDETECT bit to 1.
4. Set the HW_POWER_5VCTRL VBUSVALID_TRSH to 0x0 (2.9 V).

5.Disable VBUSDROOP status and interrupts (clear VDD5V_DROOP_IRQ).

6.Set the ENABLE_DCDC bit in HW_POWER_5VCTRL.

7.Wait 100 µs.

8.Check VBUSVALID_IRQ bit. If it is set, and 5 V is present and the 4P2 rail was enabled, then repeat the sequence for enabling the 4P2 regulator and DCDC from 4P2. This bit indicates that the DCDC has tried to source from the battery, even if 4P2 sourcing is enabled. This is because a VBUSVALID false condition automatically disables the 4P2 regulators and DCDC source, so the DCDC then falls back to battery sourcing.

9.Clear VBUSDROOP, VBUSVALID, and the output rails IRQ bits as needed.

10.Restore the output rail ENIRQ bits, the VBUSVALID_THRSH level, VBUSVALID_5VDETECT value, VBUSVALID ENIRQ, and VBUSDROOP ENIRQ to their original values.

5838 SAIF2 DMA completion IRQ in the APBX DMA module is not connected to the interrupt collector

Description:

The SAIF2 DMA completion IRQ in the APBX DMA module (channel 10) is not connected to IRQ #9 of the interrupt collector.

Workaround:

SAIF2 can only be used in FIFO mode or by polling the channel 10 DMA completion IRQ bit (CH10_CMDCMPLT_IRQ in Register HW_APBX_CTRL1)."

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: IMX23CE
Rev. 4
01/2013