MCF54418 Chip Errata

Silicon Revision: All

This document identifies implementation differences between the MCF5441*x* processors and the description contained in the *MCF54418 ColdFire® Reference Manual and Datasheet*. Refer to http://www.freescale.com/coldfire for the latest updates.

The latest mask of the MCF5441x family is 0N51E.

Table 1. Summary of MCF5441*x* Errata

Errata	Module Affected	Date Errata Added	Revision Affected?	Revision Affected?
			1M38W	0N51E
SECF171	ADC/DAC pin	3/4/2010	Yes	No
SECF180	Interrupt Controller	8/12/2010	Yes	Yes
SECF181	Synchronous Serial Interface	10/20/2010	Yes	Yes
SECF182	Synchronous Serial Interface	10/20/2010	Yes	Yes
SECF183	Synchronous Serial Interface	10/20/2010	Yes	Yes
SECF184	USB HOST / OTG	10/20/2010	Yes	Yes
SECF193	FlexCAN	10/20/2010	Yes	Yes
SECF197	mcPWM	10/20/2010	Yes	Yes
SECF199	USB HOST / OTG	04/15/2011	Yes	Yes
SECF200	USB HOST / OTG	04/15/2011	Yes	Yes
SECF201	USB HOST / OTG	04/15/2011	Yes	Yes
SECF202	USB HOST / OTG	04/15/2011	Yes	Yes
SECF203	USB HOST / OTG	04/15/2011	Yes	Yes
SECF204	USB HOST / OTG	04/15/2011	Yes	Yes

Table continues on the next page...

Table 1. Summary of MCF5441x Errata (continued)

Errata	Module Affected	Date Errata Added	Revision Affected?	Revision Affected?
			1M38W	0N51E
SECF205	DDR2 Memory Controller	04/15/2011	Yes	No
SECF207	mcPWM	09/29/2011	Yes	Yes
SECF208	mcPWM	09/29/2011	Yes	Yes
SECF209	mcPWM	09/29/2011	Yes	Yes
SECF210	Secure Digital Host Controller	04/04/2012	Yes	Yes
SECF211	Secure Digital Host Controller	04/04/2012	Yes	Yes
SECF212	Secure Digital Host Controller	04/04/2012	Yes	Yes
SECF213	Secure Digital Host Controller	04/04/2012	Yes	Yes
SECF214	Secure Digital Host Controller	04/04/2012	Yes	Yes
SECF215	Secure Digital Host Controller	04/04/2012	Yes	Yes
SECF216	Secure Digital Host Controller	04/04/2012	Yes	Yes
SECF217	Secure Digital Host Controller	04/04/2012	Yes	Yes
SECF218	Secure Digital Host Controller	04/04/2012	Yes	Yes
SECF221	JTAG	09/08/2014	Yes	Yes

The table below provides a revision history for this document.

Table 2. Document Revision History

Rev. No.	Date	Substantive Changes
0	4/2010	Initial revision
1	8/2010	Added SECF180.
2	10/2010	Added SECF181, SECF182, SECF183, SECF184, SECF185, SECF186, SECF187, SECF188.
3	04/2011	Added SECF193, SECF197, SECF199, SECF200, SECF201, SECF202, SECF203, SECF204.
4	07/2011	Removed SECF188, SECF198.
5	09/2011	Added SECF207, SECF208, SECF209.
6	N/A	This revision was not published

Table continues on the next page...

Table 2. Document Revision History (continued)

Rev. No.	Date	Substantive Changes
7	08/2012	Added SECF210, SECF211, SECF212, SECF213, SECF214, SECF215, SECF216, SECF217, SECF218.
8	09/2013	Maskset corrected from “0E51E” to “0N51E”
9	09/2014	Added SECF221 and corrected revision history table beginning with revision 6.

SECF171: Latch-up susceptibility on ADC_IN7/DAC1_OUT pin

Errata type: Silicon
Affects: ADC/DAC
Description: ADC_IN7/DAC1_OUT pin has latch-up susceptibility at a low injection current level (40mA vs. 100mA spec). This is true for positive polarity injection at 125°C.

Workaround: Must prevent large injection currents with board level protection such as a series resistance.

Fix plan: Will be fixed.

SECF180: Spurious Interrupts Can Cause Incorrect Vector Fetch

Errata type: Silicon
Affects: INTC
Description: In rare cases the interrupt controller's spurious detection logic can cause a fetch to an incorrect vector number. This can occur when the core is starting the IACK for a spurious interrupt. During this small window of time, if a second interrupt at a different level arrives, the second interrupt causes the interrupt controller logic to clear the spurious request. Therefore, the interrupt controller sees no valid interrupt pending at the requested level and returns vector number 0 for INTC0, or vector number 64 for INTC1, or vector number 128 for INTC2.

The second interrupt can be at any level other than the level that caused the spurious interrupt (it can even be a lower priority than the spurious interrupt). If the second interrupt is at the same level as the spurious interrupt, then the correct vector number for the second interrupt is returned.

Workaround: In many systems spurious interrupts represent error conditions in and of themselves. So, it is always a good design practice to eliminate potential causes of spurious interrupts during product development. Proper interrupt management can help to prevent or reduce the possibility of spurious interrupts (and the potential occurrence of this errata). The correct procedure for masking an interrupt in the INTC or inside the module is:

1. Write the interrupt level mask in the core’s status register (SR[I]) to a value higher than the priority level of the interrupt you want to mask.
2. Mask the interrupt using the INTC’s IMR and/or an interrupt mask register inside the module.
3. Write the original value back to the core’s status register.

Even when steps are taken to remove spurious interrupts, it is still desirable to have a spurious interrupt handler to help manage unexpected events and glitches in a system. A workaround to allow for correct spurious interrupt handling is to:

1. After boot, copy the vector table to RAM
2. Modify the vector 0, and vector 64, and vector 128 entries so that they point to the spurious interrupt handler.

This way the system performs the same for any potential spurious interrupt vectors. Vectors 0, 64, 128, and 24 (the correct spurious interrupt vector) should point to the same handler.

Fix plan: No plans to fix.

SECF181: Rx re-enabled with Tx disabled and TFR_CLK DIS set

Errata type: Silicon
Affects: SSI
Description: If RX is re-enabled in I2S_master mode with TX disabled along with TFR_CLK DIS set, the first data after re-enabling is received in the second timeslot and hence channel swapping takes place.

In Network Sync mode if RX is re-enabled with TX disabled when TFR_CLK_DIS is set, data is not accepted into the FIFO according to the masking bits.

Workaround: Reset SSI before enabling RX or do not use TFR_CLK_DIS feature.

Fix plan: No plans to fix.

SECF182: If TX disabled before frame sync, data not transmitted in 1st timeslot of next frame

Errata type: Silicon
Affects: SSI
Description: I2S_slave mode: If TX is disabled just before frame sync, data is not transmitted for the first timeslot in the next frame but DDR signal is high for the first timeslot.

This issue occurs if TX is disabled within four-bit clock cycles of active edge of frame sync

Workaround: WAIT for TFS and then disable TX

Fix plan: No plans to fix

SECF183: TX enabling with respect to full speed in normal sync mode

Errata type: Silicon
Affects: SSI
Description: In Normal Sync Mode if TX is enabled 4 clock cycle before external early word length frame sync with F0 disabled, data is re-transmitted from STX0 since TDE does not occur.

In Normal Sync Mode if TX is re-enabled 2 clock cycle before external non-early word length frame sync , data is not transmitted for 1 clock cycle since ddr_stxd is low for 1 clock cycle.

This issue occurs if TX is enabled within 4 bit clock cycles of active edge of frame sync.

Workaround: To enable TX, use the following sequence:
1. Enable RX in SCR register.
2. Enable RIE and RFS_EN bit in SIER register.

3. Wait for occurrence of RFS interrupt.
4. Enable TX and disable RX.

Fix plan: No plans to fix.

SECF184: Marginal crossover failures across Voltage / Temp corners

Errata type: Silicon
Affects: USB HOST / OTG
Description: Description: Timing for the first bit transmitted in the controllers is too early and makes it difficult to achieve clean USB eye diagrams. While this has not been seen as an operational issue, it may result in a USB a specification violation. The behavior in question is related to the controller not pre-driving the transmit enable pin before the first J/K transition.

Workaround: May require compliance waiver if full USB certification is requirement.

Fix plan: No plans to fix

SECF193: FlexCAN: Global Masks misalignment

Errata type: Silicon
Affects: FlexCAN
Description: During CAN messages reception by FlexCAN, the RXGMASK (Rx Global Mask) is used as acceptance mask for most of the Rx Message Buffers (MB). When the FIFO Enable bit in the FlexCAN Module Configuration Register (CANx_MCR[FEN], bit 2) is set, the RXGMASK also applies to most of the elements of the ID filter table. However, there is a misalignment between the position of the ID field in the Rx MB and in RXIDA, RXIDB and RXIDC fields of the ID Tables. In fact RXIDA filter in the ID Tables is shifted one bit to the left from Rx MBs ID position as shown below:

Rx MB ID = bits 3-31 of ID word corresponding to message ID bits 0-28

RXIDA = bits 2-30 of ID Table corresponding to message ID bits 0-28

Note that the mask bits one-to-one correspondence occurs with the filters bits, not with the incoming message ID bits. This leads the RXGMASK to affect Rx MB and Rx FIFO filtering in different ways.

For example, if the user intends to mask out the bit 24 of the ID filter of Message Buffers then the RXGMASK will be configured as 0xffff_ffef. As result, bit 24 of the ID field of the incoming message will be ignored during filtering process for Message Buffers. This very same configuration of RXGMASK would lead bit 24 of RXIDA to be "don't care" and thus bit 25 of the ID field of the incoming message would be ignored during filtering process for Rx FIFO.

Similarly, both RXIDB and RXIDC filters have multiple misalignments with regards to position of ID field in Rx MBs, which can lead to erroneous masking during filtering process for either Rx FIFO or MBs. RX14MASK (Rx 14 Mask) and RX15MASK (Rx 15 Mask) have the same structure as the RXGMASK. This includes the misalignment problem between the position of the ID field in the Rx MBs and in RXIDA, RXIDB and RXIDC fields of the ID Tables.

Workaround: It is recommended that one of the following actions be taken to avoid problems:

1. Do not enable the RxFIFO. If CANx_MCR[FEN]=0 then the Rx FIFO is disabled and thus the masks RXGMASK, RX14MASK and RX15MASK do not affect it.

2. Enable Rx Individual Mask Registers. If the Backwards Compatibility Configuration bit in the FlexCAN Module Configuration Register (CANx_MCR[BCC], bit 15) is set then the Rx Individual Mask Registers (RXIMR0-63) are enabled and thus the masks RXGMASK, RX14MASK and RX15MASK are not used.
3. Do not use masks RXGMASK, RX14MASK and RX15MASK (i.e. let them in reset value which is 0xffff_ffff) when CANx_MCR[FEN]=1 and CANx_MCR[BCC]=0. In this case, filtering processes for both Rx MBs and Rx FIFO are not affected by those masks.
4. Do not configure any MB as Rx (i.e. let all MBs as either Tx or inactive) when CANx_MCR[FEN]=1 and CANx_MCR[BCC]=0. In this case, the masks RXGMASK, RX14MASK and RX15MASK can be used to affect ID tables without affecting filtering process for Rx MBs.

Fix plan: No plans to fix.

SECF197: FlexPWM: Configuring the count value to set PWMA/PWMB first low and then high in the same cycle the output signals are low.

Errata type: Silicon
Affects: Motor control PWM
Description: The VAL2 and VAL4 registers define the turn-on edge and the VAL3 and VAL5 registers define the turn off edge of the PWMA/PWMB signals respectively. VAL3 cannot be less than VAL2 and VAL5 cannot be less than VAL4. Doing so will cause the PWM signal to turn off at the correct time (VAL3 or VAL5), but it will not turn on at the time defined by VAL2 or VAL4.

This can be an issue during the generation of phase delayed pulses where the PWM signal goes high late in PWM cycle N and remains high across the cycle boundary before going low early in cycle N+1 and goes high again in PWM cycle N+1. These errata will allow that to happen. VAL3 register must be "greater than or equal to" VAL2 register and VAL5 must be "greater than or equal to" VAL4.

Workaround: None

Fix plan: No plans to fix.

SECF199: Wrong value read in TXPBURST/RXPBURST when AHBBRST=000/100

Errata type: Silicon
Affects: USB HOST / OTG
Description: When reading registers TXPBURST (BURSTSIZE[15:8]) and RXPBURST (BURSTSIZE[7:0]), if SBUSCFG [2:0] is set to 000 or 100, the read value is always 5'b 10000, regardless of the previously programmed value in either TXPBURST or RXPBURST.

Workaround: Software must maintain a copy of this value if needed.

Fix plan: No plans to fix.

SECF200: Short inter-packet delay between OUT transactions may terminate transfer

Errata type: Silicon
Affects: USB HOST / OTG

Description: This issue occurs when controller is configured as device and the host sends two consecutive OUT transactions (for example, two ISO OUT) with a short inter-packet delay between them (less than 200xns). In this case, the clear command from the protocol engine state machine to the protocol engine data path is not sent (and internal byte count in the data path module is not cleared). This causes a short packet to be reported to the DMA engine, which finishes the transfer and the current dTD is retired.

Workaround: It is up to the software to check the transferred number of bytes.

Fix plan: No plans to fix.

SECF201: Extra resume interrupt in HOST mode

Errata type: Silicon
Affects: USB HOST / OTG
Description: When working as host, and doing a resume (software-driven by writing to bit 6 of PORTSCx register), a port change interrupt will fire at the end of resume. This is an EHCI specification violation: section is 4.3.1, end of 3rd paragraph.

Workaround: The extra interrupt must be taken into account when doing resume.

Fix plan: No plans to fix.

SECF202: FSL Serial mode resets to parallel during state PORT_RESET

Errata type: Silicon
Affects: USB HOST / OTG
Description: During a port reset, in either host mode or device mode, the software selection of the serial PHY interface can be lost.

Workaround: At each port reset, software needs to re-write PORTCTR.

Fix plan: No plans to fix.

SECF203: Host RX FIFO overflow too close to End-Of-Frame generates false frame babble or USB reset

Errata type: Silicon
Affects: USB HOST / OTG
Description: During normal operation, if the RX FIFO becomes full and the protocol engine needs to send a command to the DMA state machine, it will wait in that state until the RX FIFO becomes not full. As the protocol state machine also handles the SOF generation, the SOFs will not be sent during this interval. If one SOF is missed, the host controller will issue a false babble detection. If more than 3.125ms are elapsed without SOFs, the peripheral will recognize the idle bus as a USB reset.

This issue only happens if the RX FIFO is full long enough for a SOF to be missed. For this to happen, the host controller must have lost access to the main bus and/or the RX FIFO is much too small, being this is a throughput issue. If using non-streaming mode, this issue does not apply.

Workaround: RX FIFO overflow events must be avoided.

Fix plan: No plans to fix.

SECF204: When using serial PHY interface the host may start transmitting next IN token before packet being received from device is finished

Errata type: Silicon
Affects: USB HOST / OTG
Description: When using ISO IN endpoints with MULT=3 and low bandwidth system bus access, the controller may enter into a wait loop situation without warning the software. Due to the low bandwidth the last packet from a mult3 sequence may not be fetched in time before the last token IN is received (for that µframe/endpoint). This will cause the controller to reply with a zero length packet (ZLP), breaking the prime sequence. The DMA state machine will not be warned of this situation and the controller will send a ZLP to all the following IN tokens for that endpoint. The transaction will not be completed because the DMA state machine is waiting for the TX complete command to come from the protocol engine, which will not happen (because it is unprimed).

Workaround: Software must set correct MULT, matching the number of packets to be transmitted in a given dTD.

Fix plan: No plans to fix.

SECF205: Bus masters may fetch corrupt data/instructions from DDR2 memory.

Errata type: Silicon
Affects: All Crossbar Switch (XBS) masters
Description: The DDR Memory Controller (DDRMC) may fetch corrupt data when both DDR ports are accessed by different bus masters concurrently. These two masters can generate read-to-write transactions with a single DDR cycle between transactions with both ports active. This short cycle may cause an errant data fetch.

Workaround: To Eliminate potential for this issue all of the following must be implemented.

1. Supply 1.3V to Ivdd.
2. Configure the DDRMC for 125 MHz operation, and bypass duty cycle correction:

   MISCCR2[DDR2CLK] = 0, MISCCR2[DCCBYP] = 1
3. Adjust DDRMC controller timing to match your DDR2 timing at 125 MHz.

Fix plan: Will be fixed.

SECF207: Incorrect PWM operation when IPOL is set.

Errata type: Silicon
Affects: Motor Control PWM
Description: When IPOL bit is set in complementary mode, the source of the PWMi waveform is supposed to be switched from the VAL2 and VAL3 registers to the VAL4 and VAL5 registers. This switch is not happening.

Workaround: Instead of setting IPOL bt, the application can swap the VAL2/3 values with the VAL4/5 values.

Fix plan: No plans to fix.

SECF208: Incorrect PWM operation when mixing DMA and non-DMA controlled channels

Errata type: Silicon
Affects: Motor Control PWM
Description: When some submodules use DMA to load their VALx registers and other submodules use non-DMA means that means direct writes from the CPU, the LDOK bits for the non-DMA submodules can be incorrectly cleared at the completion of the DMA controlled load cycle. This leads to the non-DMA channels not being properly updated. Submodules that use DMA to read the input capture registers do not cause a problem for non- DMA submodules.

Workaround: Set the DMA enable bit to 1 also for non-DMA submodules, according to this the DMA will not incorrectly clear the LDOK bit for non-DMA submodules but they will be set to 1 at the end of each DMA cycle. When the CPU has to update the VALx registers of non-DMA submodules, first clear LDOK bit for non-DMA submodules.

Fix plan: No plans to fix.

SECF209: PWM signals are improperly synced when using Master Sync

Errata type: Silicon
Affects: Motor Control PWM
Description: If Master Sync signal , originated as the Local Sync from sub-module 0, is selected as the counter initialization signal for submodules 1-2-3 (slaves), with a prescaler PRSC less than 0x2 on PWM clock, the slave sub-module PWM outputs are delayed approx 2 IP clock against submodule0.

For PRSC > 0x1 the delay on slave submodules disappears.

Workaround: If Master Sync signal is requested, use a prescaler value PRSC >=0x2 to synchronize PWM outputs of sub-module 0 to slave submodules PWM outputs, or don't use the sub-module 0 channel A and B.

Fix plan: No plans to fix.

SECF210: ADMA fails when data length in the last descriptor is less or equal to 4 bytes

Errata type: Silicon
Affects: Secure Digital Host Controller
Description: A possible data corruption or incorrect bus transactions on the internal AHB bus, causing possible system corruption or a stall, can occur under the combination of the following conditions:

1. ADMA2 or ADMA1 type descriptor

2. TRANS descriptor with END flag

3. Data length is less than or equal to 4 bytes (the length field of the corresponding descriptor is set to 1, 2, 3, or 4) and the ADMA transfers one 32-bit word on the bus

4. Block Count Enable mode

Workaround: The software should avoid setting ADMA type last descriptor (TRANS descriptor with END flag) to data length less than or equal to 4 bytes. In ADMA1 mode, if needed, a last NOP descriptor can be appended to the descriptors list. In ADMA2 mode this workaround is not feasible due to SECF216.

Fix plan: No plans to fix.

SECF211: ADMA transfer error when the block size is not a multiple of four

Errata type: Silicon
Affects: Secure Digital Host Controller
Description: Issue in eSDHC ADMA mode operation. The eSDHC read transfer is not completed when block size is not a multiple of 4 in transfer mode ADMA1 or ADMA2. The eSDHC DMA controller is stuck waiting for the IRQSTAT[TC] bit in the interrupt status register.

The following examples trigger this issue:

1. Working with an SD card while setting ADMA1 mode in the eSDHC

2. Performing partial block read

3. Writing one block of length 0x200

4. Reading two blocks of length 0x22 each. Reading from the address where the write operation is performed. Start address is 0x512 aligned. Watermark is set as one word during read. This read is performed using only one ADMA1 descriptor in which the total size of the transfer is programmed as 0x44 (2 blocks of 0x22).

Workaround: When the ADMA1 or ADMA2 mode is used and the block size is not a multiple of 4, the block size should be rounded to the next multiple of 4 bytes via software. In case of write, the software should add the corresponding number of bytes at each block end, before the write is initialized. In case of read, the software should remove the dummy bytes after the read is completed.

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

4 Bytes valid data

4 Bytes valid data

4 Bytes valid data

4 Bytes valid data

4 Bytes valid data

2 Bytes valid data + 2 Byte dummy data

4 Bytes valid data

4 Bytes valid data

4 Bytes valid data

4 Bytes valid data

4 Bytes valid data

2 Bytes valid data + 2 Byte dummy data

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

Fix plan: No plans to fix.

SECF212: AutoCMD12 and R1b polling problem

Errata type: Silicon
Affects: SDHC
Description: Occurs when a pending command which issues busy is completed. For a command with R1b response, the proper software sequence is to poll the DLA for R1b commands to determine busy state completion. The DLA polling is not working properly for the ESDHC module and thus the DLA bit in PRSSTAT register cannot be polled to wait for busy state completion. This is relevant for all eSDHC ports (eSDHC1-4 ports).

Workaround: Poll bit 24 in PRSSTAT register (DLSL[0] bit) to check that wait busy state is over.

Fix plan: No plans to fix.

SECF213: : SDHC: Does not support Infinite Block Transfer Mode

Errata type: Silicon
Affects: Secure Digital Host Controller
Description: The eSDHC does not support infinite data transfers, if the Block Count register is set to one, even when block count enable is not set.

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

1. Set BCEN bit to one and enable block count

2. Set the BLKCNT to the maximum value in Block Attributes Register (BLKATTR) (0xFFFFfor 65535 blocks)

Fix plan: No plans to fix.

SECF214: Erroneous CMD CRC error and CMD Index error may occur on sending new CMD during data transfer

Errata type: Silicon
Affects: Secure Digital Host Controller
Description: When sending new, non data CMD during data transfer between the eSDHC and EMMC card, the module may return an erroneous CMD CRC error and CMD Index error. This occurs when the CMD response has arrived at the moment the FIFO clock is stopped. The following bits after the start bit of the response are wrongly interpreted as index, generating the CRC and Index errors. The data transfer itself is not impacted. The rate of occurrence of the issue is very small, as there is a need for the following combination of conditions to occur at the same cycle:

- The FIFO clock is stopped due to FIFO full or FIFO empty

- The CMD response start bit is received

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

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

Fix plan: No plans to fix.

SECF215: Glitch is generated on card clock with software reset or clock divider change

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

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

The recommended flow is as follows:

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

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

Fix plan: No plans to fix.

SECF216: Problem when ADMA2 last descriptor is LINK or NOP

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

In this case, the eSDHC completes the transfers associated with this descriptor set, whereas it does not even start the transfers associated with the new data command. For example, if a WRITE transfer operation is performed on the card using ADMA2, and the last descriptor of the WRITE descriptor set is a LINK descriptor, then the WRITE is successfully finished. Now, if a READ transfer is programmed from the SD card using ADMA2, then this transfer does not go through.

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

Fix plan: No plans to fix.

SECF217: Software can not clear DMA interrupt status bit after read operation

Errata type: Silicon
Affects: Secure Digital Host Controller
Description: After DMA read operation, if the SDHC System Clock is automatically gated off, the DINT status cannot be cleared by software.

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

Fix plan: No plans to fix.

SECF218: Misses SDIO interrupt when CINT is disabled

Errata type: Silicon
Affects: Secure Digital Host Controller
Description: An issue is identified when interfacing the SDIO card. There is a case where an SDIO interrupt from the card is not recognized by the hardware, resulting in a hang.

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

The SDIO interrupt service steps are as follows:

1. Clear CINTEN bit in IRQSTATEN and IRQSIGEN.

2. Reset the interrupt factors in the SDIO card and write 1 to clear the CINT interrupt in IRQSTAT.

3. Re-enable CINTEN bit in IRQSTATEN and IRQSIGEN.

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

Workaround: The workaround interrupt service steps are as follows:

1. Clear CINTEN bit in IRQSTATEN and IRQSIGEN.

2. Reset the interrupt factors in the SDIO card and write 1 to clear CINT interrupt in IRQSTAT.

3. Clear and then set D3CD bit in the PROCTL register. Clearing D3CD bit sets the reverse signal of DAT1 to low, even if DAT1 is low. After D3CD bit is re-enabled, the eSDHC can catch the posedge of the reversed DAT1 signal, if the DAT1 line is still low.

4. Re-enable CINTEN bit in IRQSTATEN and IRQSIGEN.

Fix plan: No plans to fix.

SECF221: TDO slow slew rate

Errata type: Silicon
Affects: JTAG
Description: When the EXTEST instruction is executed, a different, slower slew rate is used for the TDO pin.

Workaround: Limit JTAG TCLK operation to 3 MHz.

Fix plan: Currently, there are no plans to fix this.

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: MCF54418DE
Rev. 9, 9/2014