Mask Set Errata for Mask 2N02G

This report applies to mask 2N02G for these products:

- VYBRID

Errata ID	Errata Title
7860	ADC1 specifications are not guaranteed at temperatures below 0C or ADC clock speeds greater than 20Mhz.
6235	ANADIG: Unstable clock if PFD_FRAC divisor is 19
6634	Accesses to secure RAM fail for certain CSU security settings
6162	CAAM: CAAM cannot handle interleaved READ data "beats" returned by two different slaves in the system replying to two different AXI-ID accesses
6378	Cache: Cache write buffer error enable (MCM_ISCR[CWBEE]) does not work
4588	DMAMUX: When using PIT with "always enabled" request, DMA request does not deassert correctly
7957	Digital Filtering is only possible in GPI mode
6358	ENET: Write to Transmit Descriptor Active Register (ENET_TDAR) is ignored
6165	FTM: FlexTimer external clock source cannot be faster than 20MHz.
6484	FTM: The process of clearing the FTMx_SC[TOF] bit does not work as expected under a certain condition when the FTM counter reaches FTM_MOD value.
5926	FlexCAN may transmit an error free frame in place of an expected corrupted frame when a non correctable error is introduced by error injection during Move-out internal process having MECR[NCEFAFRZ] cleared.
6032	FlexCAN: A frame with wrong ID or payload is transmitted into the CAN bus when the Message Buffer under transmission is either aborted or deactivated while the CAN bus is in Bus Idle state.
5295	FlexCAN: False correctable and/or non correctable error detected in FlexCAN Scratch Area read accesses.
5641	FlexCAN: Module does not transmit a message that is enabled to be transmitted at a specific moment during the arbitration process.
5889	FlexCAN: The FlexCAN does not block frame transmission when configured to enter freeze mode upon detection of a non-correctable error and a non-correctable error occurs
6054	Local Memory Controller (LMEM): Read and write cache line commands do not function correctly.
5746	PIT: When using the PIT to trigger DMA transfers using cycle steal mode, two data transfers per request are generated

Table continues on the next page...

Errata ID	Errata Title
5612	QuadSPI: Parallel Mode cannot be used on full memory map of dual die packages
3981	SDHC: ADMA fails when data length in the last descriptor is less or equal to 4 bytes
3982	SDHC: ADMA transfer error when the block size is not a multiple of four
4624	SDHC: AutoCMD12 and R1b polling problem
3977	SDHC: Does not support Infinite Block Transfer Mode
4627	SDHC: Erroneous CMD CRC error and CMD Index error may occur on sending new CMD during data transfer
3980	SDHC: Glitch is generated on card clock with software reset or clock divider change
3983	SDHC: Problem when ADMA2 last descriptor is LINK or NOP
3978	SDHC: Software can not clear DMA interrupt status bit after read operation
3984	SDHC: eSDHC misses SDIO interrupt when CINT is disabled
6385	SEC watchdog timer does not prevent all cases of illogical descriptors from hanging DECOs
6826	SNVS: lp_secure and lp_nonsecure are both set after reset of HP Power Domain
6119	Share SERIAL Update Issues for non-PDB Writes
6472	UART: ETU compensation needed for ISO-7816 wait time (WT) and block wait time (BWT)
4647	UART: Flow control timing issue can result in loss of characters if FIFO is not enabled
4945	UART: ISO-7816 T=1 mode receive data format with a single stop bit is not supported
7100	UART: LON feature is not supported
5704	UART: TC bit in UARTx_S1 register is set before the last character is sent out in ISO7816 T=0 mode
6857	USB: Adding dTD to Primed Endpoint May Not Recognized
4535	USB: USB suspend and resume flow clarifications
7955	Wakeup Unit internal pull-up resisotrs for low power mode exit are not working.
7128	XTAL: In some cases the 24MHz oscillator start-up is slow
6933	eDMA: Possible misbehavior of a preempted channel when using continuous link mode

e7860: ADC1 specifications are not guaranteed at temperatures below 0C or ADC clock speeds greater than 20Mhz.

Errata type: Errata

Description: ADC1 specifications are not guaranteed at temperatures below 0C or ADC clock speeds greater than 20Mhz.

Workaround: 1. Use ADC0 for critical measurements when possible.

2. Operate ADC1 with ADC clock at or below 20MHz for full temperature range

3. Operate ADC1 at or above 0C for full ADC clock speed range

e6235: ANADIG: Unstable clock if PFD_FRAC divisor is 19

Errata type: Errata

Description: There may be clock instability at the Phase Frequency Divider (PFD) output when the fractional divisor at ANADIG_PLLx_PFD[PFDn_FRAC] for that PFD output is 19. A PFD value of 19 is typically used to create a 500MHz system clock when PLL1 or PLL2 has a frequency of 528MHz.

Workaround: If a 500 MHz system clock is desired, modify the PLL configuration to generate the 500MHz clock directly and bypass the PFD. The following values will generate a 500MHz PLL clock:

MFI=20 (ANADIG_PLLx_CTRL[DIV_SELECT]=0)

MFD=100 (ANADIG_PLLx_DENUM)

MFN=83 (ANADIG_PLLx_NUM)

e6634: Accesses to secure RAM fail for certain CSU security settings

Errata type: Errata
Description: Accesses to secure RAM may hang when the CSU is programmed such that secure user accesses to secure RAM accesses are not allowed.

Workaround: The only legal values allowed for CSU_CSL4[23:16] are:

0b1111_1111

0b1011_1011

0b0011_1111

0b0011_1011

0b0000_0011

e6162: CAAM: CAAM cannot handle interleaved READ data "beats" returned by two different slaves in the system replying to two different AXI-ID accesses

Errata type: Errata
Description: The CAAM master port cannot handle data interleaving when multiple transactions are initiated that have different IDs. This data interleaving happens when two accesses are launched with different IDs almost instantaneously.

Workaround: Set the DMA pipeline depth to 1. This will prevent CAAM from ever generating more than one AXI transaction, and therefore prevent AXI read interleaving from occurring. The performance degradation will be minimal.

e6378: Cache: Cache write buffer error enable (MCM_ISCR[CWBEE]) does not work

Errata type: Errata
Description: The CM4 cache contains a one entry write buffer. The one entry write buffer is controlled by the write buffer enable bit in the cache control registers. If enabled, it will buffer all writethrough and non-cacheable writes. Also, if enabled, it will break the direct association between a write bus fault and the processor completing the write (i.e.write fault imprecise).

The processor will then not see this fault. It is possible to enable fault reporting and capture fault information on buffered writes that create bus errors. This logic is in the MCM (Miscellaneous Control Module). The CWBEE (Cache Write Buffer Error Enable) bit in the

MCM_ISCR (Interrupt Status and Control Register) enables an interrupt to be generated on buffered writes that fault. There are also registers in the MCM to capture fault information (address, attributes, write data) plus a sticky bit to indicate a write buffer fault or multiple write buffer bus faults have occurred. This interrupt will be imprecise and occur sometime after the write fault.

However, due to a logic error, the CWBEE function is not working. On write buffer accesses that fault, the write fault address, attributes and data are correctly captured in the MCM fault information registers but even if CWBEE is set, no interrupt is generated.

Workaround: 1. If a write buffer is enabled and a write access is being run to an address area that may fault, directly check the Cache Write Buffer Fault indicator in the MCM module.

2. Don't enable the cache write buffers, keeping all writethrough and non-cacheable write faults precise.

e4588: DMAMUX: When using PIT with "always enabled" request, DMA request does not deassert correctly

Errata type: Errata

Description: The PIT module is not assigned as a stand-alone DMA request source in the DMA request mux. Instead, the PIT is used as the trigger for the DMAMUX periodic trigger mode. If you want to use one of the PIT channels for periodic DMA requests, you would use the periodic trigger mode in conjunction with one of the "always enabled" DMA requests. However, the DMA request does not assert correctly in this case.

Instead of sending a single DMA request every time the PIT expires, the first time the PIT triggers a DMA transfer the "always enabled" source will not negate its request. This results in the DMA request remaining asserted continuously after the first trigger.

Workaround: Use of the PIT to trigger DMA channels where the major loop count is greater than one is not recommended. For periodic triggering of DMA requests with major loop counts greater than one, we recommended using another timer module instead of the PIT.

If using the PIT to trigger a DMA channel where the major loop count is set to one, then in order to get the desired periodic triggering, the DMA must do the following in the interrupt service routine for the DMA_DONE interrupt:

1. Set the DMA_TCDn_CSR[DREQ] bit and configure DMAMUX_CHCFGn[ENBL] = 0

2. Then again DMAMUX_CHCFGn[ENBL] = 1, DMASREQ=channel in your DMA DONE interrupt service routine so that "always enabled" source could negate its request then DMA request could be negated.

This will allow the desired periodic triggering to function as expected.

e7957: Digital Filtering is only possible in GPI mode

Errata type: Errata

Description: PORT chapter documentation states the digital input filter is functional in all pin muxing modes. However this is not correct and digital filters can only be applied when the IOMUX is configured in GPI mode.

Workaround: The PORT module digital input filter should only be considered for use when using GPI mode.

e6358: ENET: Write to Transmit Descriptor Active Register (ENET_TDAR) is ignored

Errata type: Errata

Description: If the ready bit in the transmit buffer descriptor (TxBD[R]) is previously detected as not set during a prior frame transmission, then the ENET_TDAR[TDAR] bit is cleared at a later time, even if additional TxBDs were added to the ring and the ENET_TDAR[TDAR] bit is set. This results in frames not being transmitted until there is a 0-to-1 transition on ENET_TDAR[TDAR].

Workaround: Code can use the transmit frame interrupt flag (ENET_EIR[TXF]) as a method to detect whether the ENET has completed transmission and the ENET_TDAR[TDAR] has been cleared. If ENET_TDAR[TDAR] is detected as cleared when packets are queued and waiting for transmit, then a write to the TDAR bit will restart TxBD processing.

e6165: FTM: FlexTimer external clock source cannot be faster than 20MHz.

Errata type: Errata

Description: The maximum frequency of a clock source for the FlexTimer is 20Mhz. This restriction affects the external clock source option.

Workaround: If using the external clock as the FTM clock source, use the clock divisors to get it below 20MHz.

e6484: FTM: The process of clearing the FTMx_SC[TOF] bit does not work as expected under a certain condition when the FTM counter reaches FTM_MOD value.

Errata type: Errata

Description: The process of clearing the TOF bit does not work as expected when FTMx_CONF[NUMTOF] != 0 and the current TOF count is less than FTMx_CONF[NUMTOF], if the FTM counter reaches the FTM_MOD value between the reading of the TOF bit and the writing of 0 to the TOF bit. If the above condition is met, the TOF bit remains set, and if the TOF interrupt is enabled (FTMx_SC[TOIE] = 1), the TOF interrupt also remains asserted.

Workaround: Two possible workarounds exist for this erratum and the decision on which one to use is based on the requirements of your particular application.

1) Repeat the clearing sequence mechanism until the TOF bit is cleared.

Below is a pseudo-code snippet that would need to be included in the TOF interrupt routine.

```
while (FTM_SC[TOF]!=0)

{

void FTM_SC() ; // Read SC register

FTM_SC[TOF]=0 ; // Write 0 to TOF bit

}
```

2) With FTMx_CONF[TOFNUM] = 0 and a variable in the software, count the number of times that the TOF bit is set. In the TOF interrupt routine, clear the TOF bit and increment the variable that counts the number of times that the TOF bit was set.

e5926: FlexCAN may transmit an error free frame in place of an expected corrupted frame when a non correctable error is introduced by error injection during Move-out internal process having MECR[NCEFAFRZ] cleared.

Errata type: Errata

Description: If a non-correctable error is introduced by error injection in the DATA field of a Message Buffer (MB) selected to be transmitted, and MECR[NCEFAFRZ] bit is cleared, an error will be detected during the Move-out internal process. The transmitted frame should have corrupted data due to the non-correctable error introduced in DATA field. FlexCAN detects and reports a non-correctable error, but it can transmit an error free frame which is unexpected.

Workaround: When error injection test is performed by software over Message Buffers (MB) address space and MECR[NCEFAFRZ] bit is negated, FlexCAN must be in Loop Back mode (to avoid transmitting any corrupted frame into the CAN bus), and only error report information must be analyzed. Avoid performing message content comparisons for transmitted frames.

e6032: FlexCAN: A frame with wrong ID or payload is transmitted into the CAN bus when the Message Buffer under transmission is either aborted or deactivated while the CAN bus is in Bus Idle state.

Errata type: Errata

Description: The FlexCAN module may transmit an incorrect message if one or more Message Buffers (MBs) are configured for transmission while FlexCAN is in Bus Idle state, and the MB selected for transmission is either aborted or deactivated at the exact moment it starts to be transmitted. This will cause FlexCAN to transmit a syntactically correct message, but with either incorrect ID or data field. The CRC information will be calculated over the incorrect data (in case data is affected) and all other fields of the frame will be correct.

The probability of the problem occurring is limited to one CAN bit during the transmission of one frame, however under a very specific combination of simultaneous events:

a) Bug event may take place in one specific CAN bit per frame.

b) The CAN bus must be in Bus Idle state.

c) The CPU must be triggered to configure one or more MBs for transmission while in Bus Idle state.

d) The CPU must be triggered to remove the MBs just configured, by abortion or deactivation, in a short period after starting the configuration in step (c).

In summary, the probability of occurrence is very low, in the order of 1 per 10 million. Moreover, the procedure of configuring a MB followed by abortion or deactivation of the same MB in a short interval is unlikely to occur in normal applications.

In practice, there is no issue if the CPU guarantees that any MB configured for transmission will be aborted or deactivated just in the next frame. Said differently, there is no issue if any MB configured for transmission lasts active for a minimum of 1 CAN frame.

Workaround: The user can avoid the error by preventing to make Message Buffer (MB) configurations for transmission when the CAN bus is in the Bus Idle state.

To do so, there are bits in a FlexCAN debug register that can be used to determine when the CAN bus is in Idle state.

This debug register is located at:

FlexCAN Debug 1 Register (CAN_DBG1) - Base + 0x0058

The CAN Finite State Machine (CFSM) bits of CAN_DBG1 register monitor the FlexCAN's

internal state. The CFSM is the 6 least significant bits of the CAN_DBG1 register. The CAN Bit Number (CBN) is the 5 bits long field at bit positions 3 to 7 in the CAN_DBG1 register that indicates the current bit number in a given CFSM state value.

CAN_DBG1.CFSM = 0x0000_003F

CAN_DBG1.CBN = 0x1F00_0000

There are several internal states values that need to be looked for, listed below with their corresponding CFSM value.

RXINTERMISSION - 0x2F

TXINTERMISSION - 0x14

BUSIDLE - 0x02

The following procedure must be performed to configure a MB for transmission:

1) Disable all interrupts.

2) Read CAN_DBG1.CFSM and CAN_DBG1.CBN fields.

3) Check if CFSM value is either BUSIDLE, RXINTERMISSION or TXINTERMISSION. For the later two values, also check if CBN value is 3, to determine the paired conditions RXINTERMISSION bit 3 or TXINTERMISSION bit 3, and proceed as described below.

3.1) If CAN_DBG1 fields indicate BUSIDLE, wait N CPU clocks.

3.2) Else if CAN_DBG1 fields indicate either RXINTERMISSION bit 3 or TXINTERMISSION bit 3 wait until CFSM is different from either RXINTERMISSION or TXINTERMISSION.

3.3) Check again CAN_DBG1 fields, if they indicate BUSIDLE, wait for DELAY time.

4) Write 0x0 into Code field of CS word.

5) Enable all interrupts.

6) Write the ID word.

7) Write the DATA words.

8) Write 0xC into Code field of CS word.

Note: DELAY = {2*(MAXMB+1) + 18 } * peripheral_clock_period + 3 * PE_clock_period + 1 * CAN_bit_period

The "Number Of The Last Message Buffer" (MAXMB) are the 7 least significant bits in Module Configuration Register (CAN_MCR: base + 0x0).

e5295: FlexCAN: False correctable and/or non correctable error detected in FlexCAN Scratch Area read accesses.

Errata type: Errata

Description: False correctable and/or non correctable error can be detected in read accesses, performed by FlexCAN internal processes and/or host in FlexCAN Scratch Area in RAM, as follows:

a) in 0x0530 - 0x053F RAM address range for 64 Message Buffers

b) in 0x02B0 - 0x02BF RAM address range for 32 Message Buffers

Workaround: Assure Non-Correctable Errors In FlexCAN Access Put Device In Freeze Mode bit in Memory Error Control Register (FCMECR[NCEFAFRZ]) is negated to prevent the FlexCAN module entering in to freeze mode due to the false error detection. Also, ignore any error indication related to the following accesses:

a) in 0x0530 - 0x053F RAM address range, corresponding the IPS address range 0x0AB0 - 0x0ABF.

b) in 0x02B0 - 0x02BF RAM address range, corresponding the IPS address range 0x0AB0 - 0x0ABF.

e5641: FlexCAN: Module does not transmit a message that is enabled to be transmitted at a specific moment during the arbitration process.

Errata type: Errata

Description: FlexCAN does not transmit a message that is enabled to be transmitted in a specific moment during the arbitration process. The following conditions are necessary to have the issue.

- Only one MB is configured to be transmitted

- The write which enables the MB to be transmitted (write on Control status word) happens during a specific clock during the arbitration process.

After this arbitration process occurs, the bus goes to Idle state and no new message is received on bus.

For example:

1) MB13 is deactivated on RxIntermission (write 0x0 on CODE field from Control Status word) - First write on CODE

2) Reconfigure the ID and data fields

3) Enable the MB13 to be transmitted on BusIdle (write 0xC on Code field) - Second write on code

4) CAN bus keeps in Idle state

5) No write on Control status from any MB happens.

During the second write on code (step 3), the write must happen one clock before the current MB13 is to be scanned by arbitration process. In this case, it does not detect the new code (0xC) and no new arbitration is scheduled.

The problem can be detectable only if the message traffic ceases and the CAN bus enters into Idle state after the described sequence of events.

There is NO ISSUE if any of the conditions below holds:

a) Any MB (either Tx or Rx) is reconfigured (by writing its CS field) just after the Intermission field.

b) There is other configured MB to be transmitted

c) A new incoming message sent by any external node starts just after the Intermission field.

Workaround: To transmit a CAN frame, the CPU must prepare a Message Buffer for transmission by executing the following standard 5 step procedure:

1. Check if the respective interrupt bit is set and clear it.

2. If the MB is active (transmission pending), write the ABORT code (0b1001) to the CODE field of the Control and Status word to request an abortion of the transmission.Wait for the corresponding IFLAG to be asserted by polling the IFLAG register or by the interrupt request if enabled by the respective IMASK. Then read back the CODE field to check if the transmission was aborted or transmitted. If backwards compatibility is desired (MCR[AEN] bit negated), just write the INACTIVE code (0b1000) to the CODE field to inactivate the MB but then the pending frame may be transmitted without notification.

3. Write the ID word.

4. Write the data bytes.

5. Write the DLC, Control and CODE fields of the Control and Status word to activate the MB.

The workaround consists of executing two extra steps:

6. Reserve the first valid mailbox as an inactive mailbox (CODE=0b1000). If RX FIFO is disabled, this mailbox must be MB0. Otherwise, the first valid mailbox can be found by using table "RX FIFO filters" on FlexCAN3 chapter.

7. Write twice INACTIVE code (0b1000) into the first valid mailbox.

Note: The first mailbox cannot be used for reception or transmission process.

e5889: FlexCAN: The FlexCAN does not block frame transmission when configured to enter freeze mode upon detection of a non-correctable error and a non-correctable error occurs

Errata type: Errata

Description: When the NCEFAFRZ ("Non-Correctable Errors In FlexCAN Access Put Device In Freeze Mode") bit in the Memory Error Control Register (MECR) is set, the FlexCAN should be put into Freeze mode when a non-correctable error is detected in a memory read performed by the FlexCAN internal processes.

The problem occurs when a non-correctable error in a read memory access is detected at the exact moment that the FlexCAN is performing the move-out process, which occurs when the Message Buffer contents are copied to the Serial Message Buffer (SMB) for transmission. Under the described scenario, the expected behavior would be for the FlexCAN to block the transmission and enter Freeze mode. However, the FlexCAN transmits the frame before entering Freeze mode and the non-correctable error is present in the transmitted frame. Under any circumstance, the non-correctable error is reported in the Error Report Registers.

The problem only occurs in the FlexCAN versions with the ECC feature enabled.

The probability of such occurrence is lower than the normal occurrence of a non-correctable error event since this event must occur in a time window between the arbitration and the move-out internal processes. There is no issue outside of this time window.

Workaround: Do not set the MECR[NCEFAFRZ] bit.

e6054: Local Memory Controller (LMEM): Read and write cache line commands do not function correctly.

Errata type: Errata

Description: The read and write cache line commands specified in the LMEM Cache line control registers (LMEM_PCCLCR & LMEM_PSCLCR) do not function correctly. Results from the commands are potentially invalid. Normal cache functions are not affected. This issue only limits the ability to view and change cache data for debug purposes.

Workaround: No workaround available.

e5746: PIT: When using the PIT to trigger DMA transfers using cycle steal mode, two data transfers per request are generated

Errata type: Errata

Description: If the PIT is used to trigger DMA transfers using cycle steal mode, DMA_DCRn[CS] = 1, each transfer request will cause the source data to be written twice. The data will be written first to the desired destination address and then a second time to the destination address + 1. The destination address pointer increments by 2 for each transfer request.

Workaround: It is recommended that the PIT not be used for triggering DMA transfers and the low power timer (LPTMR) be used instead.

If it is required to use the PIT to trigger DMA transfers, the destination address must be in RAM and the buffer size must be twice the amount of data being transferred. Software must then skip every second entry in the destination buffer.

e5612: QuadSPI: Parallel Mode cannot be used on full memory map of dual die packages

Errata type: Errata

Description: When using dual die flashes on QuadSPI0 is is not possible to operate Flash A2 and B2 in parallel mode mode. This does not effect A1 and B1 which will work correctly in parallel mode.

QuadSPI1 does not support dual die packages and is therefore not affected by this errata.

Workaround: If using 2 dual die packages then parallel mode should only be used on A1/B1 combination. A2 and B2 should only be read in serial mode.

e3981: SDHC: ADMA fails when data length in the last descriptor is less or equal to 4 bytes

Errata type: Errata

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 ERR003983.

e3982: SDHC: ADMA transfer error when the block size is not a multiple of four

Errata type: Errata

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.

e4624: SDHC: AutoCMD12 and R1b polling problem

Errata type: Errata

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 ompletion. 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.

e3977: SDHC: Does not support Infinite Block Transfer Mode

Errata type: Errata

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)

e4627: SDHC: Erroneous CMD CRC error and CMD Index error may occur on sending new CMD during data transfer

Errata type: Errata

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.

e3980: SDHC: Glitch is generated on card clock with software reset or clock divider change

Errata type: Errata

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.

e3983: SDHC: Problem when ADMA2 last descriptor is LINK or NOP

Errata type: Errata

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.

e3978: SDHC: Software can not clear DMA interrupt status bit after read operation

Errata type: Errata

Description: After DMA read operation, if the SDHC System Clock is automatically gated off, the DINT status can not 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.

e3984: SDHC: eSDHC misses SDIO interrupt when CINT is disabled

Errata type: Errata

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.

e6385: SEC watchdog timer does not prevent all cases of illogical descriptors from hanging DECOs

Errata type: Errata

Description: The SEC block contains a watchdog timer designed to clear errors which have hung a DECO. Although mostly effective, there are cases in which poorly constructed descriptors can hang a DECO from which a watchdog can’t recover..

The SEC's programming model is based on the construction of Descriptors, which include commands and optionally, embedded keys and context. Users are expected to use the SEC's driver (SEC Descriptor Runtime Assembler) to build descriptors; however, because there is considerable innate flexibility in descriptor construction - even with the Descriptor Runtime Assembler - it is possible to create descriptors that appear to follow the rules of descriptor construction but still lead to errors, hangs, or corrupted data.

A few examples of descriptor constructions which can lead to hangs not cleared by the watchdog are:

- Ending a descriptor with a LOAD IMM command.

- Creating descriptors that command the PKHA to unload more data than the RAMs actually store

Workaround: To avoid the errata, it is highly recommended that users model their descriptors after the examples provided in Freescale's reference software. If there is a need to create a descriptor for which an example does not exist, do so based on SEC Programmer's Reference Manual.

e6826: SNVS: lp_secure and lp_nonsecure are both set after reset of HP Power Domain

Errata type: Errata

Description: The first time SNVS boots, the HAB will attempt to provision SNVS in the secure state. This will cause the lp_secure bit to be set in the LPSR register. If SNVS is rebooted then it is possible for the SNVS Secure State Machine to transition to the non-secure state. If software then attempts to load the ZMK through the HW interface the lp_nonsecure bit will also be set. The loading of the ZMK should have been blocked because SNVS_LP is provisioned to be secure and SNVS_HP is non-secure.

Workaround: The software that loads the ZMK needs to read both the HPSR and LPSR to verify that the state of the secure state machine matches the provisioned state of SNVS_LP. If they do not match then the software must not load the ZMK through the HW mechanism. If software ever detects that both lp_secure and lp_nonsecure are set then software should flag a fatal error to the system and set the master key select to only select the OTPMK and lock it. This will restrict CAAM from using the ZMK.

e6119: Share SERIAL Update Issues for non-PDB Writes

Errata type: Errata

Description: When using share type SERIAL, the following scenario could occur:

1. A job descriptor or shared descriptorwrites a value to memory location B, and the descriptor completes execution before the write response has been received. Note that the value written is NOT an update of the Shared Descriptor.

2. The next descriptor in the flow fetches and begins executing the same shared descriptor.

3. The second descriptor then reads the data from memory location B.

If, for some reason, the write done by the first descriptor is significantly delayed while the reads for the second descriptor are not delayed, it is possible that the reads will pass the write and the second descriptor will receive bad data. (i.e. the value in memory location B prior to the write by the first descriptor)

Note that this is not a problem for PDB updates because DECO already ensures that such updates complete before the Shared Descriptor can be read again from memory. Note also that this isn't a problem for share types WAIT or ALWAYS since the user is responsible in those cases for ensuring that there are no order dependencies. Only SERIAL sharing promises to take care of order dependencies.

Workaround: There are two suggested workarounds.

Option 1: Enter JUMP WAIT for No Output Pending as the last command in the descriptor. This will force the DECO to wait for the write to be complete before the first descriptor finishes execution, thus eliminating the problem. The downside is that this will be an unbuffered write, which can be time-consuming, and the DECO will be idle for all that time.

Option 2: Finish the descriptor with a STORE of one word of the Shared Descriptor using type 0x42. This will trigger the PDB writeback waiting logic while freeing the DECO to start executing some other job. But the serial-shared descriptor will not be executed again until the final write response from its prior execution has been received. The disadvantage is an extra bus transaction. However, the advantage is that DECO does not sit idle while waiting. From a performance standpoint, this is probably the better solution.

e6472: UART: ETU compensation needed for ISO-7816 wait time (WT) and block wait time (BWT)

Errata type: Errata

Description: When using the default ISO-7816 values for wait time integer (UARTx_WP7816T0[WI]), guard time FD multiplier (UARTx_WF7816[GTFD]), and block wait time integer (UARTx_WP7816T1[BWI]), the calculated values for Wait Time (WT) and Block Wait Time (BWT) as defined in the Reference Manual will be 1 ETU less than the ISO-7816-3 requirement.

Workaround: To comply with ISO-7816 requirements, compensation for the extra 1 ETU is needed. This compensation can be achieved by using a timer, such as the low-power timer (LPTMR), to introduce a 1 ETU delay after the WT or BWT expires.

e4647: UART: Flow control timing issue can result in loss of characters if FIFO is not enabled

Errata type: Errata

Description: On UARTx modules with FIFO depths greater than 1, when the /RTS flow control signal is used in receiver request-to-send mode, the /RTS signal is negated if the number of characters in the Receive FIFO is equal to or greater than the receive watermark. The /RTS signal will not negate until after the last character (the one that makes the condition for /RTS negation true) is completely received and recognized. This creates a delay between the end of the STOP bit and the negation of the /RTS signal. In some cases this delay can be long enough that a transmitter will start transmission of another character before it has a chance to recognize the negation of the /RTS signal (the /CTS input to the transmitter).

Workaround: Always enable the RxFIFO if you are using flow control for UARTx modules with FIFO depths greater than 1. The receive watermark should be set to seven or less. This will ensure that there is space for at least one more character in the FIFO when /RTS negates. So in this case no data would be lost.

Note that only UARTx modules with FIFO depths greater than 1 are affected. The UARTs that do not have the RxFIFO feature are not affected. Check the Reference Manual for your device to determine the FIFO depths that are implemented on the UARTx modules for your device.

e4945: UART: ISO-7816 T=1 mode receive data format with a single stop bit is not supported

Errata type: Errata

Description: Transmission of ISO-7816 data frames with single stop bit is supported in T=1 mode. Currently in order to receive a frame, two or more stop bits are required. This means that 11 ETU reception based on T=1 protocol is not supported. T=0 protocol is unaffected.

Workaround: Do not send T=1, 11 ETU frames to the UART in ISO-7816 mode. Use 12 ETU transmissions for T=1 protocol instead.

e7100: UART: LON feature is not supported

Errata type: Information

Description: The LON feature of UART is not supported.

Workaround: Do not use the LON feature on any UART.

e5704: UART: TC bit in UARTx_S1 register is set before the last character is sent out in ISO7816 T=0 mode

Errata type: Errata

Description: When using the UART in ISO-7816 mode, the UARTx_S1[TC] flag sets after a NACK is received, but before guard time expires.

Workaround: If using the UART in ISO-7816 mode with T=0 and a guard time of 12 ETU, check the UARTn_S1[TC] bit after each byte is transmitted. If a NACK is detected, then the transmitter should be reset.

The recommended code sequence is:

```
UART0_C2 &= ~UART_C2_TE_MASK; //make sure the transmitter is disabled at first
UART0_C3 |= UART_C3_TXDIR_MASK; //set the TX pin as output
UART0_C2 |= UART_C2_TE_MASK; //enable TX
UART0_C2 |= UART_C2_RE_MASK; //enable RX to detect NACK
for(i=0;i<length;i++)
{
while(!(UART0_S1&UART_S1_TDRE_MASK)){}
UART0_D = data[i];
while(!(UART0_S1&UART_S1_TC_MASK)){}//check for NACK
if(UART0_IS7816 & UART_IS7816_TXT_MASK)//check if TXT flag set
{
/* Disable transmit to clear the internal NACK detection counter */
UART0_C2 &= ~UART_C2_TE_MASK;
UART0_IS7816 = UART_IS7816_TXT_MASK;// write one to clear TXT
UART0_C2 |= UART_C2_TE_MASK; // re-enable transmit
}
}
UART0_C2 &= ~UART_C2_TE_MASK; //disable after transmit
```

e6857: USB: Adding dTD to Primed Endpoint May Not Recognized

Errata type: Errata

Description: There is an issue with the add dTD tripwire semaphore (ATDTW bit in USBCMD register) that can cause the controller to ignore a dTD that is added to a primed endpoint. When this happens, the software can read the tripwire bit and the status bit at '1' even though the endpoint is unprimed.

Workaround: The software must implement a periodic poll cycle, and check for each dTD pending on execution (Active = 1), if the enpoint is primed. It can do this by reading the corresponding bits in the ENDPTPRIME and ENDPTSTAT registers. If these bits are read at 0, the software needs to re-prime the endpoint by writing 1 to the corresponding bit in the ENDPTPRIME register. This can be done for every microframe, every frame or with a larger interval, depending on the urgency of transfer execution for the application.

e4535: USB: USB suspend and resume flow clarifications

Errata type: Errata

Description: In device mode, The PHY can be put into Low Power Suspend when the device is not running or the host has signaled suspend. The PHY Low power suspend bit (PORTSC1.PHCD) will be cleared automatically when the host initials resume. Before forcing a resume from the device, the device controller driver must clear this bit. In host mode, the PHY can be put into Low Power Suspend when the downstream device has been placed into suspend mode (PORTSC1.SUSP) or when no downstream device is connected. Low power suspend is completely under the control of software.

To place the PHY into Low power mode, software needs to set PORTSC1.PHCD bit, set all bits in USBPHY_PWD register and set the USBPHY_CTRL.CLKGATE bit.

When a remote wakeup occurs after the Suspend (SUSP) bit is set while the PHY Low power suspend bit (PHCD) is cleared, a USB interrupt (USBSTS.PCI) will be generated. In this case, the PHCD bit will NOT be set because of the interrupt. However, if a remote wakeup occurs after the PHCD bit is set while the USB PHY Power-Down Register (USBPHY_PWD) and the UTMI clock gate (USBPHY_CTRL.CLKGATE) bit is cleared, a remote wakeup interrupt will be generated. In this case, all the bits in the HW_USBPHY_PWD register and the USBPHY_CTRL.CLKGATE bit will be set, even after the remote wakeup interrupt is generated, which is incorrect.

Workaround: To place the USB PHY into low power suspend mode, the following sequence should be performed in an atomic operation (interrupts should be disabled during these three steps):

1. Set the PORTSC1.PHCD bit

2. Set all bits in the USBPHY_PWD register

3. Set the USBPHY_CTRL.CLKGATE bit

e7955: Wakeup Unit internal pull-up resisotrs for low power mode exit are not working.

Errata type: Errata

Description: WKPU_WIPUER register should enable internal pull up resistors on wakeup pins during low power mode. However this does not work as described or required.

Workaround: Use external pull up resistors on the wakeup pins if they are required to exit from low power modes.

e7128: XTAL: In some cases the 24MHz oscillator start-up is slow

Errata type: Errata

Description: In some cases the 24 MHz oscillator start-up is slow.

Workaround: A 2.2 MO resistor from XTAL to ground will make the startup times reasonable

e6933: eDMA: Possible misbehavior of a preempted channel when using continuous link mode

Errata type: Errata

Description: When using continuous link mode (DMA_CR[CLM] = 1) with a high priority channel linking to itself, if the high priority channel preempts a lower priority channel on the cycle before its last read/write sequence, the counters for the preempted channel (the lower priority channel) are corrupted. When the preempted channel is restored, it runs past its "done" point instead of performing a single read/write sequence and retiring.

The preempting channel (the higher priority channel) will execute as expected.

Workaround: Disable continuous link mode (DMA_CR[CLM]=0) if a high priority channel is using minor loop channel linking to itself and preemption is enabled. The second activation of the preempting channel will experience the normal startup latency (one read/write sequence + startup) instead of the shortened latency (startup only) provided by continuous link mode.

Document Number: VYBRID_2N02G
Rev. 09 JUN 2014