Errata Sheet

Rel. 1.0, 2016-01

Device	**XMC4700/XMC4800**
Marking/Step	**ES-AA, AA**
Package	**PG-LQFP-100/144, PG-LFBGA-196**

Overview

Document ID is **03508AERRA**.

This “Errata Sheet” describes product deviations with respect to the user documentation listed below.

Table 1 Current User Documentation

Document	Version	Date
XMC4700/XMC4800 Reference Manual	V1.1	October 2015
XMC4700/XMC4800 Data Sheet	V1.0	January 2016

Make sure that you always use the latest documentation for this device listed in category “Documents” at **http://www.infineon.com/xmc4000**.

Notes

1. *The errata described in this sheet apply to all temperature and frequency versions and to all memory size and configuration variants of affected devices, unless explicitly noted otherwise.*
2. *Devices marked with EES or ES are engineering samples which may not be completely tested in all functional and electrical characteristics, therefore they must be used for evaluation only. Specific test conditions for EES and ES are documented in a separate “Status Sheet”, delivered with the device.*
3. *XMC4000 devices are equipped with an ARM® Cortex®-M4 core. Some of the errata have a workaround which may be supported by some compiler tools. In order to make use of the workaround the corresponding compiler switches may need to be set.*

Conventions used in this Document

Each erratum is identified by **Module_Marker**.**TypeNumber**:

- **Module**: Subsystem, peripheral, or function affected by the erratum.
- **Marker**: Used only by Infineon internal.
- **Type**: type of deviation
  - **(none)**: Functional Deviation
  - **P**: Parametric Deviation
  - **H**: Application Hint
  - **D**: Documentation Update
- **Number**: Ascending sequential number. As this sequence is used over several derivatives, including already solved deviations, gaps inside this enumeration can occur.

1 History List / Change Summary

Table 2 History List

Version	Date	Remark
1.0	2016-01	Initial AA step version.

Table 3 Errata fixed from previous step

Errata	Short Description	Change
- none -		

Table 4 Errata fixed compared to XMC4500 AC step

Errata	Short Description	Change
CCU4_AI.001	CCU4 period interrupt is not generated in capture mode	Fixed
CCU8_AI.001	CCU8 Floating Prescaler function does not work with Capture Trigger 1	Fixed
CCU8_AI.004	CCU8 output PWM glitch when using low side modulation via the Multi Channel Mode	Fixed
CCU_AI.001	CCU4 and CCU8 capture full flags do not work when module clock is faster than peripheral bus clock	Fixed
CCU_AI.002	CCU4 and CCU8 Prescaler synchronization clear does not work when Module Clock is faster than Peripheral Bus Clock	Fixed
CCU_AI.003	CCU4 and CCU8 capture full flag is not cleared if a capture event occurs during a bus read phase	Fixed
CCU_AI.004	CCU4 and CCU8 Extended Read Back loss of data	Fixed
CCU_AI.005	CCU4 and CCU8 External IP clock Usage	Fixed in CCU8

Table 4 Errata fixed compared to XMC4500 AC step (cont'd)

Errata	Short Description	Change
DAC_CM.001	DAC immediate register read following a write issue	Fixed
DAC_CM.002	No error response for write access to read only DAC ID register	Fixed
DAC_CM.P001	INL parameter limits violated by some devices	Fixed
DEBUG_CM.001	OCDS logic in peripherals affected by TRST	Fixed
ETH_CM.002	MAC provides incorrect status and corrupts frames when RxFIFO overflow occurs on penultimate word of Rx frames of specific lengths	Fixed
GPDMA_CM.001	Unexpected Block Complete Interrupt During Multi-Block Transfers	Fixed
GPDMA_CM.002	GPDMA doesn't Accept Transfer During/In 2nd Cycle of 2-Cycle ERROR Response	Fixed
RTC_CM.001	RTC event might get lost	Fixed
SCU_CM.002	Missed wake-up event during entering external hibernate mode	Fixed
SCU_CM.015	Functionality of parity memory test function limited	Fixed
SDMMC_CM.003	SDMMC input pins cannot be released for other usage	Fixed

Table 5 Functional Deviations

Functional Deviation	Short Description	Chg	Pg
ADC_AI.008	**Wait-for-Read condition for register GLOBRES not detected in continuous auto-scan sequence**	New	**10**
ADC_AI.016	**No Channel Interrupt in Fast Compare Mode with GLOBRES**	New	**11**

Table 5 Functional Deviations (cont'd)

Functional Deviation	Short Description	Chg	Pg
ADC_TC.064	Effect of conversions in 10-bit fast compare mode on post-calibration	New	11
CCU8_AI.003	CCU8 Parity Checker Interrupt Status is cleared automatically by hardware	New	12
CCU_AI.005	CCU4 External IP clock Usage	New	13
CCU_AI.006	Value update not usable in period dither mode	New	15
CPU_CM.001	Interrupted loads to SP can cause erroneous behavior	New	16
CPU_CM.004	VDIV or VSQRT instructions might not complete correctly when very short ISRs are used	New	17
DEBUG_CM.002	CoreSight logic only reset after power-on reset	New	19
DSD_AI.001	Possible Result Overflow with Certain Decimation Factors	New	19
DTS_CM.001	DTS offset calibration value limitations	New	20
ETH_AI.001	Incorrect IP Payload Checksum at incorrect location for IPv6 packets with Authentication extension header	New	20
ETH_AI.002	Incorrect IP Payload Checksum Error status when IPv6 packet with Authentication extension header is received	New	21
ETH_AI.003	Overflow Status bits of Missed Frame and Buffer Overflow counters get cleared without a Read operation	New	22
LEDTS_AI.001	Delay in the update of FNCTL.PADT bit field	New	23

Table 5 Functional Deviations (cont'd)

Functional Deviation	Short Description	Chg	Pg
PARITY_CM.001	Parity error signaling can be suppressed in write/read sequence	New	28
PARITY_CM.002	Clock limitations for ETH and SDMMC modules when using parity check of module SRAMs	New	29
PORTS_CM.001	P15_PDISC.[4,5] register bits cannot be written	New	29
PORTS_CM.005	Different PORT register reset values after module reset	New	30
PORTS_CM.006	PORT driver strength register Pn_PDR not writable for Ports 7,8,9	New	31
POSIF_AI.001	Input Index signal from Rotary Encoder is not decoded when the length is 1/4 of the tick period	New	32
SCU_CM.003	The state of HDCR.HIB bit of HCU gets updated only once in the register mirror after reset release	New	34
SCU_CM.006	Deep sleep entry with PLL power-down option generates SOSCWDGT and SVCOLCKT trap	New	34
SCU_CM.021	Registering of service requests in SRRAW register can fail	New	35
SDMMC_CM.001	Unexpected interrupts after execution of CMD13 during bus test	New	36
SDMMC_CM.002	Unexpected Tx complete interrupt during R1b response	New	36
SDMMC_CM.004	Busy response from card in write resume operation not detected	New	38
SDMMC_CM.005	Controller sends other command when Auto CMD12 enabled	New	38

Table 5 Functional Deviations (cont'd)

Functional Deviation	Short Description	Chg	Pg
SDMMC_CM.006	Stream write issue due to wrong FIFO handling causes data corruption in eMMC mode	New	38
SDMMC_CM.007	Consecutive write to the same register in SD clock domain	New	39
SDMMC_CM.008	Receive state machine hangs if driver programs stop at block gap request using CMD18 when receive buffers are full	New	40
SDMMC_CM.009	Latching current value in the response register	New	40
USB_CM.002	GAHBCFG.GlblIntrMsk not cleared with a software reset	New	40
USB_CM.003	Endpoint NAK not sent in Device Class applications with multiple endpoints enabled	New	41
USB_CM.004	USB core is not able to detect resume or new session request after PHY clock is stopped	New	42
USIC_AI.005	Only 7 data bits are generated in IIC mode when TBUF is loaded in SDA hold time	New	42
USIC_AI.006	Dual SPI format not supported	New	43
USIC_AI.007	Protocol-related argument and error bits in register RBUFSR contain incorrect values following a received data word	New	43
USIC_AI.008	SSC delay compensation feature cannot be used	New	45
USIC_AI.009	Baud rate generator interrupt cannot be used	New	46
USIC_AI.010	Minimum and maximum supported word and frame length in multi-IO SSC modes	New	46

Table 5 Functional Deviations (cont'd)

Functional Deviation	Short Description	Chg	Pg
USIC_AI.011	Write to TBUF01 has no effect	New	47
USIC_AI.013	SCTR register bit fields DSM and HPCDIR are not shadowed with start of data word transfer	New	47
USIC_AI.014	No serial transfer possible while running capture mode timer	New	47
USIC_AI.015	Wrong generation of FIFO standard transmit/receive buffer events when TBCTR.STBTEN/RBCTR.SRBTEN = 1	New	48
USIC_AI.016	Transmit parameters are updated during FIFO buffer bypass	New	48
USIC_AI.018	Clearing PSR.MSLS bit immediately deasserts the SELOx output signal	New	49
USIC_AI.020	Handling unused DOUT lines in multi-IO SSC mode	New	50

Table 6 Application Hints

Hint	Short Description	Chg	Pg
ADC_AI.H003	Injected conversion may be performed with sample time of aborted conversion	New	51
ADC_AI.H004	Completion of Startup Calibration	New	52
ADC_AI.H008	Injected conversion with broken wire detection	New	52
ADC_TC.H011	Bit DCMSB in register GLOBCFG	New	53
MultiCAN_AI.H005	TxD Pulse upon short disable request	New	54
MultiCAN_AI.H006	Time stamp influenced by resynchronization	New	54
MultiCAN_AI.H007	Alert Interrupt Behavior in case of Bus-Off	New	54

Table 6 Application Hints (cont'd)

Hint	Short Description	Chg	Pg
MultiCAN_AI.H008	Effect of CANDIS on SUSACK	New	55
MultiCAN_TC.H003	Message may be discarded before transmission in STT mode	New	55
MultiCAN_TC.H004	Double remote request	New	56
PORTS_CM.H001	RTC_XTAL pins are swapped	New	56
RESET_CM.H001	Power-on reset release	New	57

2 Functional Deviations

The errata in this section describe deviations from the documented functional behavior.

ADC_AI.008 Wait-for-Read condition for register GLOBRES not detected in continuous auto-scan sequence

In the following scenario:

- A continuous auto-scan is performed over several ADC groups and channels by the Background Scan Source, using the global result register (GLOBRES) as result target (GxCHCTRy.RESTBS=1_B), and
- The Wait-for-Read mode for GLOBRES is enabled (GLOBRCR.WFR=1_B),

each conversion of the auto-scan sequence has to wait for its start until the result of the previous conversion has been read out of GLOBRES.

When the last channel of the auto-scan is converted and its result written to GLOBRES, the auto-scan re-starts with the highest channel number of the highest ADC group number. But the start of this channel does not wait until the result of the lowest channel of the previous sequence has been read from register GLOBRES, i.e. the result of the lowest channel may be lost.

Workaround

If either the last or the first channel in the auto-scan sequence does not write its result into GLOBRES, but instead into its group result register (selected via bit GxCHCTRy.RESTBS=0_B), then the Wait-for-Read feature for GLOBRES works correctly for all other channels of the auto-scan sequence.

For this purpose, the auto-scan sequence may be extended by a “dummy” conversion of group x/ channel y, where the Wait-for-Read mode must not be selected (GxRCRy.WFR=0_B) if the result of this “dummy” conversion is not read.

ADC_AI.016 No Channel Interrupt in Fast Compare Mode with GLOBRES

In fast compare mode, the compare value is taken from bitfield RESULT of the selected result register and the result of the comparison is stored in the respective bit FCR.

A channel event can be generated when the input becomes higher or lower than the compare value.

In case the global result register GLOBRES is selected, the comparison is executed correctly, the target bit is stored correctly, source events and result events are generated, but a channel event is not generated.

Workaround

If channel events are required, choose a local result register GxRESy for the operation of the fast compare channel.

ADC_TC.064 Effect of conversions in 10-bit fast compare mode on post-calibration

The calibrated converters Gx (x = 0..3) support post-calibration. Unless disabled by software (via bits GLOBCFG.DPCALx = 0), a calibration step is performed after each conversion, incrementally increasing/decreasing internal calibration values to compensate process, temperature, and voltage variations.

If a conversion in 10-bit fast-compare mode (bit field CMS/E = 101_B in corresponding Input Class register) is performed between two conversions in other (non-fast-compare) modes on a converter Gx, the information gained from the last post-calibration step is disturbed. This will lead to a slightly less accurate result of the next conversion in a non-fast-compare mode.

Depending on the ratio of conversions in fast-compare mode versus conversions in other modes, this effect will be more or less obvious.

In a worst case scenario (fast-compare with a constant result injected between each two normal conversions), all calibration values can drift to their maxima / minima, causing the converter Gx to deliver considerably inaccurate results.

Workaround

Do not mix conversions using 10-bit fast-compare mode and other conversions with enabled postcalibration on the calibrated converters Gx (x = 0..3). Instead, use a dedicated group for fast-compare operations.

CCU8_AI.003 CCU8 Parity Checker Interrupt Status is cleared automatically by hardware

Each CCU8 Module Timer has an associated interrupt status register. This Status register, CC8yINTS, keeps the information about which interrupt source triggered an interrupt. The status of this interrupt source can only be cleared by software. This is an advantage because the user can configure multiple interrupt sources to the same interrupt line and in each triggered interrupt routine, it reads back the status register to know which was the origin of the interrupt.

Each CCU8 module also contains a function called Parity Checker. This Parity Checker function, crosschecks the output of a XOR structure versus an input signal, as seen in Figure 1.

When using the parity checker function, the associated status bitfield, is cleared automatically by hardware in the next PWM cycle whenever an error is not present.

This means that if in the previous PWM cycle an error was detected and one interrupt was triggered, the software needs to read back the status register before the end of the immediately next PWM cycle.

This is indeed only necessary if multiple interrupt sources are ORed together in the same interrupt line. If this is not the case and the parity checker error source is the only one associated with an interrupt line, then there is no need to read back the status information. This is due to the fact, that only one action can be triggered in the software routine, the one linked with the parity checker error.

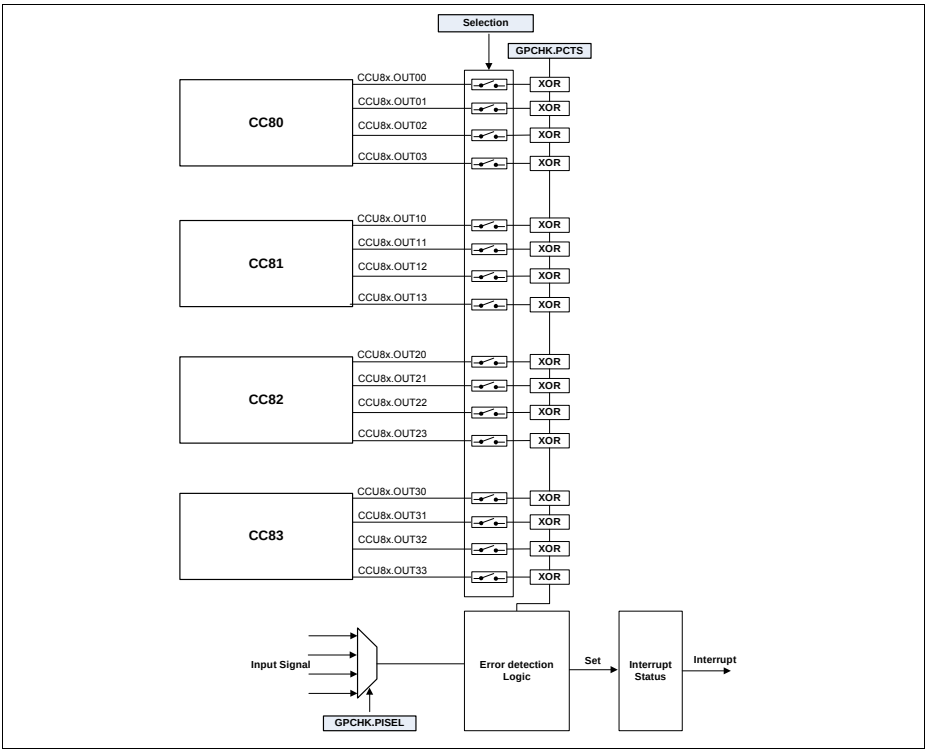

Figure 1 Parity Checker diagram

Workaround

Not ORing the Parity Checker error interrupt with any other interrupt source. With this approach, the software does not need to read back the status information to understand what was the origin of the interrupt - because there is only one source.

CCU_AI.005 CCU4 External IP clock Usage

Each CCU4 module offers the possibility of selecting an external signal to be used as the master clock for every timer inside the module Figure 1. External

signal in this context is understood as a signal connected to other module/IP or connected to the device ports.

The user has the possibility after selecting what is the clock for the module (external signal or the clock provided by the system), to also select if this clock needs to be divided. The division ratios start from 1 (no frequency division) up to 32768 (where the selected timer uses a frequency of the selected clock divided by 32768).

This division is selected by the PSIV field inside of the CC4yPSC register. Notice that each Timer Slice (CC4y) have a specific PSIV field, which means that each timer can operate in a different frequency.

Currently is only possible to use an external signal as Timer Clock when a division ratio of 2 or higher is selected. When no division is selected (divided by 1), the external signal cannot be used.

The user must program the PSIV field of each Timer Slice with a value different from 0000_B - minimum division value is /2.

This is only applicable if the Module Clock provided by the system (the normal default configuration and use case scenario) is not being used. In the case that the normal clock configured and programmed at system level is being used, there is not any type of constraints.

One should not also confuse the usage of an external signal as clock for the module with the usage of an external signal for counting. These two features are completely unrelated and there are not any dependencies between both.

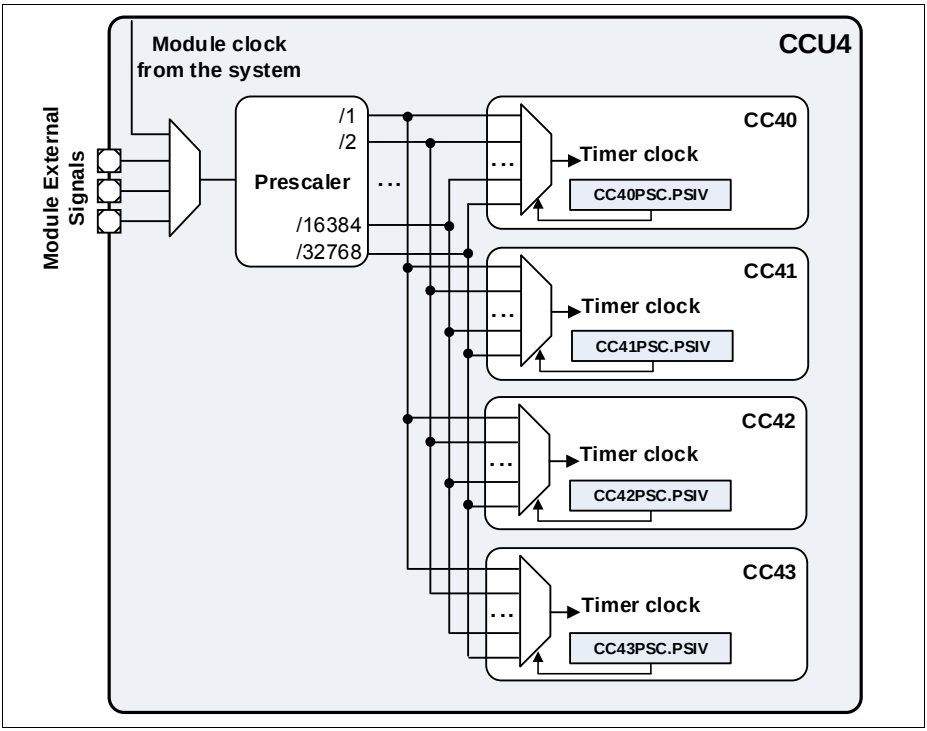

Figure 2 Clock Selection Diagram for CCU4

Workaround

None.

CCU_AI.006 Value update not usable in period dither mode

Each CCU4/CCU8 timer gives the possibility of enabling a dither function, that can be applied to the duty cycle and/or period. The duty cycle dither is done to increase the resolution of the PWM duty cycle over time. The period dither is done to increase the resolution of the PWM switching frequency over time.

Each of the dither configurations is set via the DITHE field:

- DITHE = 00_B - dither disabled

- DITHE = 01_B - dither applied to the duty-cycle (compare value)
- DITHE = 10_B - dither applied to the period (period value)
- DITHE = 11_B - dither applied to the duty-cycle and period (compare an period value)

Whenever the dither function is applied to the period (DITHE = 10_B or DITHE = 11_B) and an update of the period value is done via a shadow transfer, the timer can enter a stuck-at condition (stuck at 0).

Implication

Period value update via shadow transfer cannot be used if dither function is applied to the period (DITHE programmed to 10_B or 11_B).

Workaround

None.

CPU_CM.001 Interrupted loads to SP can cause erroneous behavior

If an interrupt occurs during the data-phase of a single word load to the stack-pointer (SP/R13), erroneous behavior can occur. In all cases, returning from the interrupt will result in the load instruction being executed an additional time. For all instructions performing an update to the base register, the base register will be erroneously updated on each execution, resulting in the stack-pointer being loaded from an incorrect memory location.The affected instructions that can result in the load transaction being repeated are:

1. LDR SP,[Rn],#imm
2. LDR SP,[Rn,#imm]!
3. LDR SP,[Rn,#imm]
4. LDR SP,[Rn]
5. LDR SP,[Rn,Rm]

The affected instructions that can result in the stack-pointer being loaded from an incorrect memory address are:

1. LDR SP,[Rn],#imm
2. LDR SP,[Rn,#imm]!

Conditions

1. An LDR is executed, with SP/R13 as the destination
2. The address for the LDR is successfully issued to the memory system
3. An interrupt is taken before the data has been returned and written to the stack-pointer.

Implications

Unless the load is being performed to Device or Strongly-Ordered memory, there should be no implications from the repetition of the load. In the unlikely event that the load is being performed to Device or Strongly-Ordered memory, the repeated read can result in the final stack-pointer value being different than had only a single load been performed.

Interruption of the two write-back forms of the instruction can result in both the base register value and final stack-pointer value being incorrect. This can result in apparent stack corruption and subsequent unintended modification of memory.

Workaround

Both issues may be worked around by replacing the direct load to the stack-pointer, with an intermediate load to a general-purpose register followed by a move to the stack-pointer.

If repeated reads are acceptable, then the base-update issue may be worked around by performing the stack pointer load without the base increment followed by a subsequent ADD or SUB instruction to perform the appropriate update to the base register.

CPU_CM.004 VDIV or VSQRT instructions might not complete correctly when very short ISRs are used

The VDIV and VSQRT instructions take 14 cycles to execute. When an interrupt is taken a VDIV or VSQRT instruction is not terminated, and completes its execution while the interrupt stacking occurs. If lazy context save of floating point state is enabled then the automatic stacking of the floating point context

does not occur until a floating point instruction is executed inside the interrupt service routine.

Lazy context save is enabled by default. When it is enabled, the minimum time for the first instruction in the interrupt service routine to start executing is 12 cycles. In certain timing conditions, and if there is only one or two instructions inside the interrupt service routine, then the VDIV or VSQRT instruction might not write its result to the register bank or to the FPSCR.

Conditions

1. The floating point unit is present and enabled
2. Lazy context saving is not disabled
3. A VDIV or VSQRT is executed
4. The destination register for the VDIV or VSQRT is one of s0 - s15
5. An interrupt occurs and is taken
6. The interrupt service routine being executed does not contain a floating point instruction
7. 14 cycles after the VDIV or VSQRT is executed, an interrupt return is executed

A minimum of 12 of these 14 cycles are utilized for the context state stacking, which leaves 2 cycles for instructions inside the interrupt service routine, or 2 wait states applied to the entire stacking sequence (which means that it is not a constant wait state for every access).In general this means that if the memory system inserts wait states for stack transactions then this erratum cannot be observed.

Implications

The VDIV or VQSRT instruction does not complete correctly and the register bank and FPSCR are not updated, meaning that these registers hold incorrect, out of date, data.

Workaround

A workaround is only required if the floating point unit is present and enabled. A workaround is not required if the memory system inserts one or more wait states to every stack transaction.

There are two workarounds:

1. Disable lazy context save of floating point state by clearing LSPEN to 0 (bit 30 of the FPCCR at address 0xE000EF34).
2. Ensure that every interrupt service routine contains more than 2 instructions in addition to the exception return instruction.

DEBUG_CM.002 CoreSight logic only reset after power-on reset

The CoreSight logic should also be reset with a debug reset (DBGRESET).

Opposed to this specification the debug reset does not have an effect on the CoreSight logic. Therefore CoreSight logic can only be reset by a power-on reset (PORESET).

Workaround

If the user quits the debug session and likes to leave the system clean, without a PORESET, the following steps have to be performed:

- Disable debug functions by disable of DHCSR.C_DEBUGEN bit in debug halting and status register.
- Disable HW breakpoints in FPB unit of each comparator by disable of FP_CTRL.ENABLE bit in flashpatch control register.
- Disable trace functions by disable of DEMCR.TRCENA bit in debug exception and monitor control register. This disables DWT, ITM, ETM and TPIU functions.

DSD_AI.001 Possible Result Overflow with Certain Decimation Factors

Certain combinations of CIC filter grade and oversampling rate (see below) can lead to an overflow within the CIC filter. These combinations must be avoided to ensure proper operation of the digital filter.

Critical combinations:

- CIC2 (CFMC/CFAC = 01_B) with oversampling rate of 182
- CIC3 (CFMC/CFAC = 10_B) with oversampling rate of 33, 41, 51, 65, 81, 102, 129, 162…182, 204
- CICF (CFMC/CFAC = 11_B) with oversampling rate of 129, 182

Note: Filter grade and oversampling rate are defined in register FCFGCx/FCFGAx. The shown oversampling rates are defined as CFMDF+1/CFADF+1.

Workaround

None.

DTS_CM.001 DTS offset calibration value limitations

When using the value $7F_H$ for offset calibration in DTSCON.OFFSET the Die Temperature Sensor may return invalid results in DTSSTAT.RESULT.

Implication

The value $7F_H$ (equivalent to -1) for DTSCON.OFFSET cannot be used.

Workaround

If the application needs a small negative offset then $7E_H$ (equivalent to -2) could be used.

ETH_AI.001 Incorrect IP Payload Checksum at incorrect location for IPv6 packets with Authentication extension header

When enabled, the Ethernet MAC computes and inserts the IP header checksum (IPv4) or TCP, UDP, or ICMP payload checksum in the transmitted IP datagram (IPv4 or IPv6) on per-packet basis. The Ethernet MAC processes the IPv6 header and the optional extension headers (if present) to identify the start of actual TCP, UDP, or ICMP payload for correct computation and insertion of payload checksum at appropriate location in the packet. The IPv6 header length is fixed (40 bytes) whereas the extension header length is specified in units of N bytes:

Extension Header Length Field Value x N bytes + 8 bytes
where N = 4 for authentication extension header and N = 8 for all other extension headers supported by the Ethernet MAC. If the actual payload bytes

are less than the bytes indicated in the Payload Length field of the IP header, the Ethernet MAC indicates the IP Payload Checksum error.

If the payload checksum is enabled for an IPv6 packet containing the authentication extension header, then instead of bypassing the payload checksum insertion, the Ethernet MAC incorrectly processes the packet and inserts a payload checksum at an incorrect location. As a result, the packet gets corrupted, and it is dropped at the destination. The software should not enable the payload checksum insertion for such packets because the Integrity Check Value (ICV) in the authentication extension header is calculated and inserted considering that the payload data is immutable (not modified) in transit. Therefore, even if the payload checksum is correctly calculated and inserted, it results into a failure of the ICV check at the final destination and the packet is eventually dropped.

Workaround

The software should not enable the IP payload checksum insertion by the Ethernet MAC for Tx IPv6 packets with authentication extension headers. The software can compute and insert the IP payload checksum for such packets.

ETH_AI.002 Incorrect IP Payload Checksum Error status when IPv6 packet with Authentication extension header is received

The Ethernet MAC processes a TCP, UDP, or ICMP payload in the received IP datagrams (IPv4 or IPv6) and checks whether the received checksum field matches the computed value. The result of this operation is given as an IP Payload Checksum Error in the receive status word. This status bit is also set if the length of the TCP, UDP, or ICMP payload does not match the expected payload length given in the IP header.

In IPv6 packets, there can be optional extension headers before actual TCP, UDP, or ICMP payload. To compute and compare the payload checksum for such packets, the Ethernet MAC sequentially parses the extension headers, determines the extension header length, and identifies the start of actual TCP, UDP, or ICMP payload. The header length of all extension headers supported by the Ethernet MAC is specified in units of 8 bytes (Extension Header Length Field Value x 8 bytes + 8 bytes) except in the case of authentication extension

header. For authentication extension header, the header length is specified in units of 4 bytes (Extension Header Length Field Value x 4 bytes + 8 bytes).

However, because of this defect, the Ethernet MAC incorrectly interprets the size of the authentication extension header in units of 8 bytes, because of which the following happens:

- Incorrect identification of the start of actual TCP, UDP, or ICMP payload
- Computing of incorrect payload checksum
- Comparison with incorrect payload checksum field in the received IPv6 frame that contains the authentication extension header
- Incorrect IP Payload Checksum Error status

As a result, the IP Payload checksum error status is generated for proper IPv6 packets with authentication extension header. If the Ethernet MAC core is programmed to drop such `error` packets, such packets are not forwarded to the host software stack.

Workaround

Disable dropping of TCP/IP Checksum Error Frames by setting Bit 26 (`DT`) in the Operation Mode Register (`OPERATION_MODE`). This enables the Ethernet MAC core to forward all packets with IP checksum error to the software driver. The software driver must process all such IPv6 packets that have payload checksum error status and check whether they contain the authentication extension header. If authentication extension header is present, the software driver should either check the payload checksum or inform the upper software stack to check the packet for payload checksum.

ETH_AI.003 Overflow Status bits of Missed Frame and Buffer Overflow counters get cleared without a Read operation

The DMA maintains two counters to track the number of frames missed because of the following:

- Rx Descriptor not being available
- Rx FIFO overflow during reception

The Missed Frame and Buffer Overflow Counter register indicates the current value of the missed frames and FIFO overflow frame counters. This register

also has the Overflow status bits (Bit 16 and Bit 28) which indicate whether the rollover occurred for respective counter. These bits are set when respective counter rolls over. These bits should remain high until this register is read.

However, erroneously, when the counter rollover occurs second time after the status bit is set, the respective status bit is reset to zero.

Effects

The application may incorrectly detect that the rollover did not occur since the last read operation.

Workaround

The application should read the Missed Frame and Buffer Overflow Counter register periodically (or after the Overflow or Rollover status bits are set) such that the counter rollover does not occur twice between read operations.

LEDTS_AI.001 Delay in the update of FNCTL.PADT bit field

The touch-sense pad turn (PADT) value is updated, not at the end of the touch-sense time slice (ColA), but one time slice later (**Figure 3**).

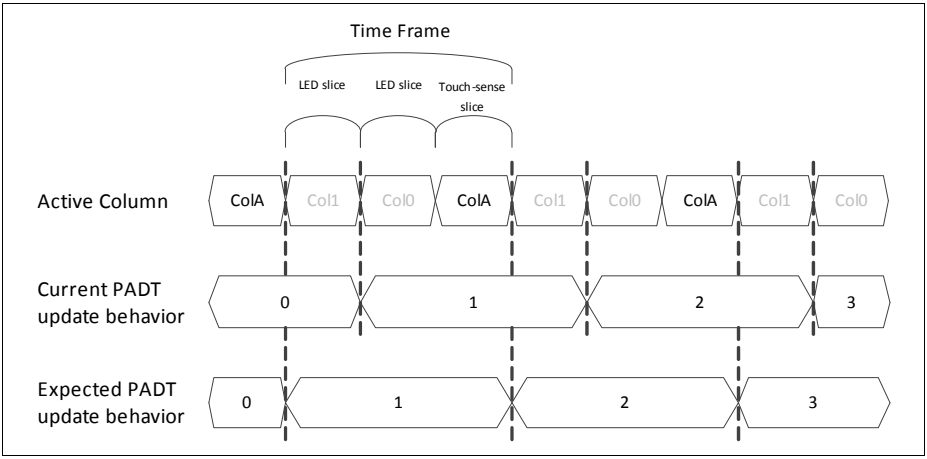

Figure 3 PADT update behavior

If the number of LED columns enabled is smaller than 2, the delay will affect the activation period of the current active pad. At the beginning of every new Col A, the value of the current PADT's compare register is updated to the internal compare register. However, the delay causes the value of the previous PADT's compare register is updated to the internal compare register instead. This means that the current active pad would be activated with the duration of the previous pad's oscillation window (**Figure 4**). In addition to this, when no LEDs are enabled, pad turn 0 will prevail for one time slice longer before it gets updated (**Figure 5**).

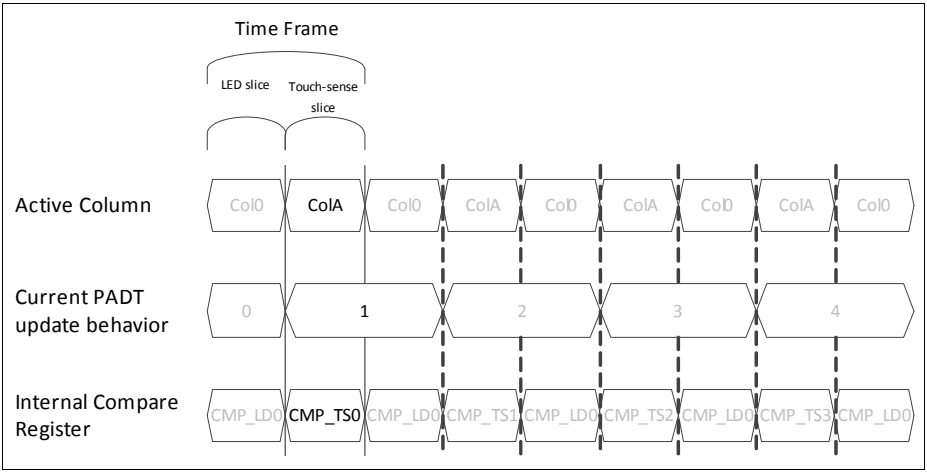

Figure 4 Effect of delay on the update of Internal Compare Register with 1 LED column enabled

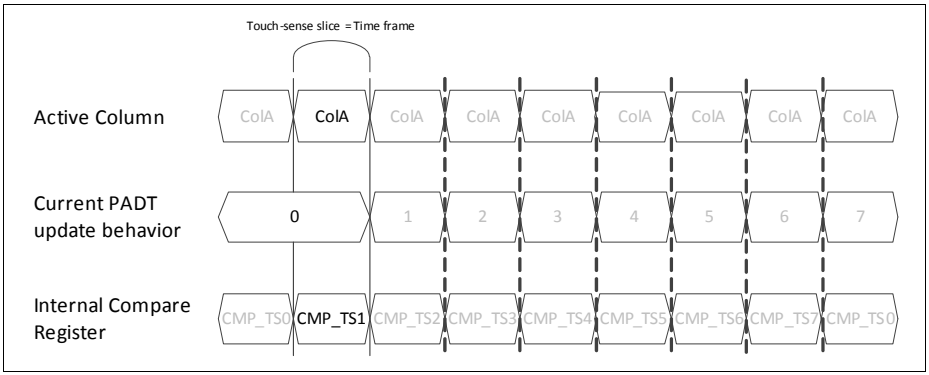

Figure 5 Pad turn 0 prevails for one time slice longer when no LEDs are enabled

If the number of LED columns enabled is 2 or more, the additional LED columns would provide some buffer time for the delay. So, at the start of a new touch-sense time slice, the update of PADT value would have taken place. Hence, the current active PADT compare register value is updated to the internal compare register (**Figure 6**).

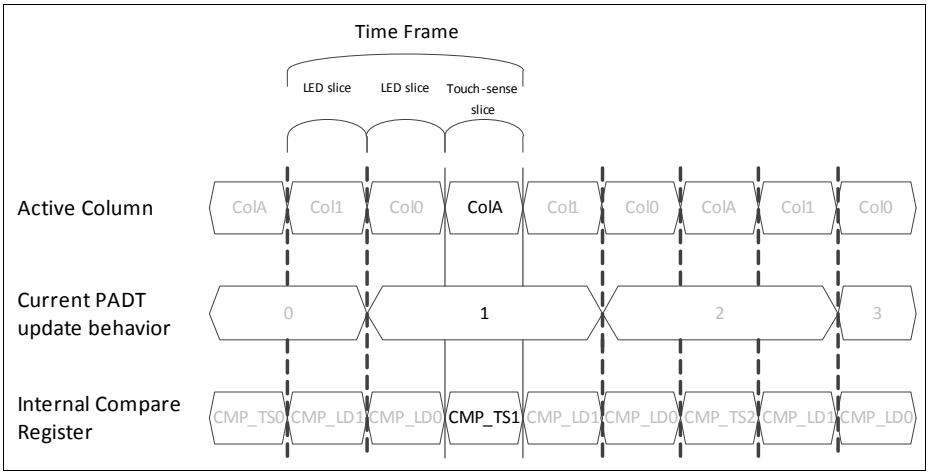

Figure 6 Internal Compare Register updated with correct compare register value with 2 LED columns enabled

Conditions

This delay in PADT update can be seen in cases where hardware pad turn control mode (FNCTL.PADTSW = 0) is selected and the touch-sense function is enabled (GLOBCTL.TS_EN = 1).

Workaround

This section is divided to two parts. The first part will provide a guide on reading the value of the bit field FNCTL.PADT via software. The second part will provide some workarounds for ensuring that the CMP_TS[x] values are aligned to the current active pad turn.

Workaround for reading PADT

Due to the delay in the PADT update, the user would get the current active pad turn when PADT is read in the time frame interrupt. However, this PADT value read differs when read in a time slice interrupt. This depends on the number of LED columns enabled and the active function or LED column in the previous time slice (**Table 7**). The bit field FNCTL.FNCOL provides a way of interpreting the active function or LED column in the previous time slice.

Table 7 PADT value as read in the time slice interrupt

No. of LED Columns Enabled	Previous active function / LED column	FNCTL.FNCOL	PADT value
0-1	Touch-sense or LED Col0	110_B or 111_B	Previous active pad turn
2-7	Touch-sense or first LED column after touch-sense	110_B or 111_B	Previous active pad turn
	Second LED column after touch-sense onwards	101_B to 000_B	Current or next active pad turn

Workaround for aligning CMP_TSx

One workaround is to use the software pad turn control. Then this issue can be avoided entirely because the pad turn update will have to be handled by software.

However, it is still possible to work around this issue when using the hardware pad turn control. In the previous section, it is known that when the number of LED columns enabled is smaller than 2, the current active pad is activated with the oscillation window of the previous active pad. This means that the current active pad is activated with the value programmed in the bit field CMP_TS[x-1] instead of CMP_TS[x]. There are two possible software workarounds for this issue:

1. At the end of the time frame interrupt service routine, software can prepare for the next active pad turn by programming the CMP_TS[x-1] bit field with the intended compare value for TSIN[x]. As an example, if the next active pad is TSIN[2], program CMP_TS[1] with the compare value intended for TSIN[2] (**Figure 7**).

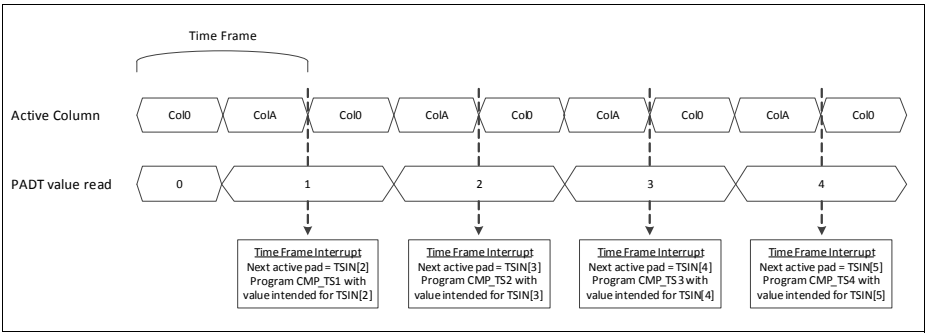

Figure 7 Software workaround demonstration

1. During the initialization phase, program the CMP_TS[x] bit fields with the left-shift factored in. Example: CMP_TS[0] for TSIN[1], CMP_TS[1] for TSIN[2], ... CMP[7] for TSIN[0].

PARITY_CM.001 Parity error signaling can be suppressed in write/read sequence

The device PSRAM and DSRAM offers parity protection. The parity information is stored byte wise. AHB memory access are executed word (32-bit) aligned.

Due to weakness of the memories AHB bus interface the parity error signaling **by AHB bus error** is suppressed if the following usage scenario occurs:

- if a parity error is present in a byte (D1) of a 32-bit aligned word and
- if another byte or half-word (D2) is written to the same word and
- if data which contains both parts (D1) and (D2) is read immediately after the write.

Parity error signaling **by parity error NMI trap** is still available.

The functional problem exists due to buffering of AHB write requests and therefore occurs only in immediate write/read sequence scenarios on the same 32-bit aligned memory cell.

Following types of access sequences are affected.

Table 8 Affected write/read sequences

write	read
byte	half-word
byte	word
half-word	word

Example

If a parity error is present in a byte @200002303_H and if the byte @200002302_H (same 32-bit word) is written and immediately after this the word @200002300_H (or half-word @200002302_H) is read then no parity error by AHB bus error is signaled.

Workaround

If the described usage scenario can occur in the application then it is recommended to enable the **parity error trap** for the used PSRAM and/or DSRAM units. By this a **NMI** trap will be signaled to the CPU. Enabling is done by programming the SCU register PETE.

Note: The NMI trap occurs with a delay. Therefore program execution is likely to continue after the read access causing the parity error trap.

PARITY_CM.002 Clock limitations for ETH and SDMMC modules when using parity check of module SRAMs

The SRAM memories used by ETH and SDMMC (XMC4500 devices only) offer error detection by parity bit protection. If a parity error is detected then it is forwarded to SCU and if parity error detection is enabled by settings in SCU register PEEN then a trap request is triggered.

In affected devices the forwarding mechanism does not work with some clock settings.

Workaround

If parity detection shall be enabled then following clock setting limitations must be obeyed:

- For ETH: $f_{CPU} = f_{SYS}$ or CPU clock divider must be disabled (SCU register bit CPUCLKCR.CPUDIV = 0).
- For SDMMC: f_{CPU} ? f_{SDMMC}+25%. For example if SDMMC shall operate with f_{SDMMC} at 48 MHz then f_{CPU} must be set for 60 MHz or higher.

PORTS_CM.001 P15_PDISC.[4,5] register bits cannot be written

The bits 4 and 5 of the register P15_PDISC cannot be modified by software and always retain their reset value 0_B. As a result of this, the digital input path of the related shared analog and digital input pins cannot be disabled.

Implications

Software that sets one or both of these bits and later reads P15_PDISC will not see the expected read value, but always reads 0_B for P15_PDISC.[4,5].

Software that reads P15_IN will read undefined values for P15_IN[4,5]. The read values depend on the analog input level of the respective pin.

Workaround

None.

PORTS_CM.005 Different PORT register reset values after module reset

The PORTS registers can be reset independent of the reset of the system with SCU_PRSET1.PPORTSRS. After such a module reset, some PORTS registers have a reset value different to the reset value that is documented in the Reference Manual.

Table 9 PORTS registers reset values

Register	Sytem reset value	Module reset value
Pn_IOCR8	$0000\ 0000_H$	$2020\ 2020_H$ 1)
Pn_PDISC	$XXXX\ XXXX_H$ 2)	$0000\ 0000_H$
Pn_PDR0	$2222\ 2222_H$	$0000\ 0000_H$
Pn_PDR1	$2222\ 2222_H$	$0000\ 0000_H$

1) Only in XMC4500 devices.

2) Device and package dependent

Implications

The different value in Pn_IOCR8 configures the respective port pins Pn.[11:8] as inverted inputs instead of direct inputs. User software in Priviledged Mode can reconfigure them as needed by the application.

With the different value in Pn_PDISC of the digital ports the availability of digital pins in a device can no longer be verified via this register. Note that Pn_PDISC of pure digital ports is read-only; user software can't write to them.
The Pn_PDISC of the shared analog/digital port pins (P14 and P15) enables/disables the digital input path. After a system reset this path is disabled, after a module reset enabled. User software in Priviledged Mode can reconfigure them as needed by the application.

The different value in the Pn_PDR registers configures output port pins with a "Strong-Sharp" output driver mode, as opposed to "Strong-Soft" driver mode after a system reset. This may result in a higher current consumption and more

noise induced to the external system. User software in Priviledged Mode can reconfigure them as needed by the application.

Workaround

None.

PORTS_CM.006 PORT driver strength register Pn_PDR not writable for Ports 7,8,9

The PORTS allow the user to change the driver strength of each pin via register Pn_PDR. For Port 7, 8, 9 these Pn_PDR register can not be written by the user and remains in default reset value which is “Strong-Soft” driver mode.

Implications

A write access to the Pn_PDR register of Port 7,8,9 will not change the reset value of these register. In the SCU register TRAPRAW a Peripheral 1 Bridge TRAP gets requested.

The Embedded Trace Macrocell (ETM) enables program execution reconstruction. The ETM transmits the information as packets on the Trace Port output Unit (TPIU) with uses one clock and 4 data pins connected to a Debugger Box. Since the required trace clock frequency is fCPU/2 the pins on port 7 and 8 can not be used. To use the ETM Trace please select a other set of available pins for that interface.

Port 7 provides additional redundant connection for the EBU. These signals are limited to “Strong-Soft” driver mode which reduces the maximal performance of the external connected device. Affected are the address line signals A19-A22.

Workaround

None.

POSIF_AI.001 Input Index signal from Rotary Encoder is not decoded when the length is 1/4 of the tick period

Each POSIF module can be used as an input interface for a Rotary Encoder. It is possible to configure the POSIF module to decode 3 different signals: Phase A, Phase B (these two signals are 90° out of phase) and Index. The index signal is normally understood as the marker for the zero position of the motor Figure 1.

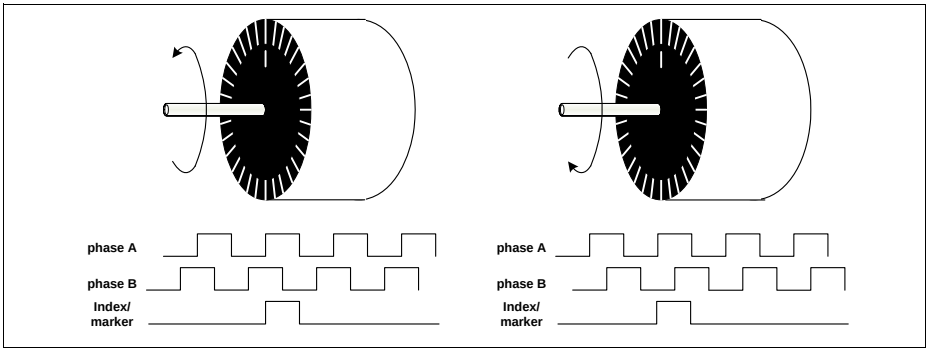

Figure 8 Rotary Encoder outputs - Phase A, Phase B and Index

There are several types of Rotary Encoder when it comes to length of the index signal:

- length equal or bigger than 1 tick period
- length equal or bigger than 1/2 tick period
- length equal or bigger than 1/4 tick period

When the index signal is smaller than 1/2 of the tick period, the POSIF module is not able to decode this signal properly, Figure 2 - notice that the reference edge of the index generation in this figure is the falling of Phase B, nevertheless this is an example and depending on the encoder type, this edge may be one of the other three.

Due to this fact it is not possible to use the POSIF to decode these type of signals (index with duration below 1/2 of the tick period).

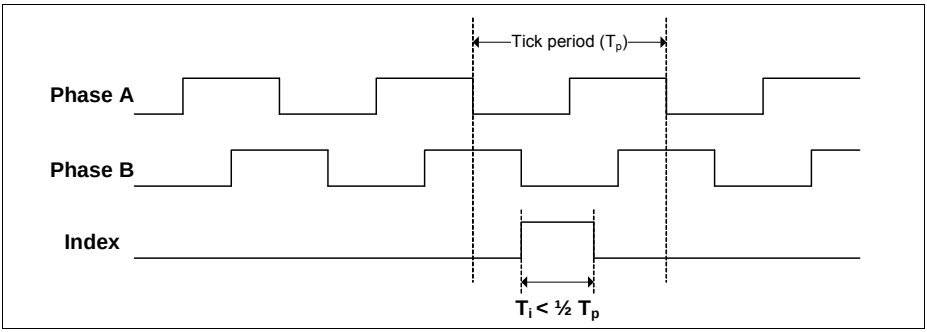

Figure 9 Different index signal types

Workaround

To make usage of the Index signal, when the length of this signal is less than 1/2 of the tick period, one should connect it directly to the specific counter/timer. This connection should be done at port level of the device (e.g. connecting the device port to the specific Timer/Counter(s)), Figure 3.

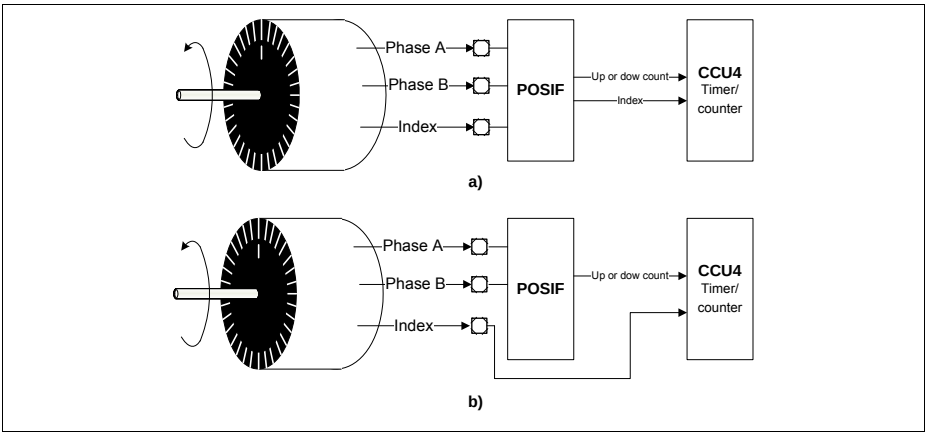

Figure 10 Index usage workaround - a) Non working solution; b) Working solution

SCU_CM.003 The state of HDCR.HIB bit of HCU gets updated only once in the register mirror after reset release

The state of HDCR.HIB bit of HCU gets updated only once in the register mirror in SCU after system reset. Any write access to this register gets propagated to hibernate domain but it will be not propagated back to the register mirror when altered by the hardware inside of the hibernate domain.

Implications

The state of HDCR.HIB cannot be effectively used for the purpose debugging of hibernate mode control software.

Workaround

For debugging of the hibernate mode control software observe the electrical states on the hibernate control pins in order to verify hibernate control circuit behavior.

SCU_CM.006 Deep sleep entry with PLL power-down option generates SOSCWDGT and SVCOLCKT trap

Entering the deep sleep mode with PLL power-down option (selected in DSLEEPCR register of SCU module) may result with system traps triggered by PLL watchdog (the SOSCWDGT trap) and/or loss-of-lock (the SVCOLCKT trap).

Implications

Occurrence of one of the enabled traps will result in an immediate wake-up from the deep sleep state, i.e. the deep sleep is effectively not entered.

Workaround

Disable SOSCWDGT and SVCOLCKT trap generation in TRAPDIS register of SCU before entering deep sleep mode with PLL power-down option selected.

SCU_CM.021 Registering of service requests in SRRAW register can fail

If a write to the service request clear register (SRCLR) occurs at the same time as one or multiple hardware request(s) then the hardware request(s) normally stored in SRRAW register is (are) lost.

The hardware request(s) and the cleared request(s) must not match to make the error occur.

Implications

If affected hardware requests (see SRRAW column in **Table 10**) are used by the application then these may get lost. The Workaround should be implemented.

Workaround

The interrupt routine assigned to an affected request must

- service the request(s) flagged in the SRSTAT register
- clear the corresponding bit(s) in SRRAW register via SRCLR register
- check the primary request source information of all affected and used service request sources and update the SRRAW via SRSET register accordly.

For checking of the primary request source please use information provided in **Table 10**. Example: if RTC bit RAWSTAT.RAI is set but SCU bit SRRAW.AI is not set then this request was lost. SRRAW should then be updated accordingly.

Table 10 Request source and related SRRAW register bit field

Request Source		SRRAW
Module	**Bit field**	**Bit field**
WDT	TIM counter value	PRWARN
RTC	RAWSTAT.RP*	PI
RTC	RAWSTAT.RAI	AI
DLR	OVRSTAT.LN*	DLROVR
SCU	HDSTAT.ULPWDG	ULP_WDG
SCU	MIRRSTS.HDSET	HDSET

Table 10 Request source and related SRRAW register bit field (cont'd)

Request Source		SRRAW
Module	**Bit field**	**Bit field**
SCU	MIRRSTS.OSCSICTRL	OSCSICTRL
SCU	MIRRSTS.RTC_CTR	RTC_CTR
SCU	MIRRSTS.RTC_ATIM0	RTC_ATIM0
SCU	MIRRSTS.RTC_ATIM1	RTC_ATIM1
SCU	MIRRSTS.RTC_TIM0	RTC_TIM0
SCU	MIRRSTS.RTC_TIM1	RTC_TIM1
SCU	MIRRSTS.RMX	RMX

SDMMC_CM.001 Unexpected interrupts after execution of CMD13 during bus test

This issue affects eMMC cards only.

The conditions for this behavior are as follows (all 2 conditions must be true):

- The host sends CMD19 (bus test pattern to a card), and driver issues CMD13 (SEND_STATUS command) to read the card status
- The transmit FSM is in Tx data state during bus testing procedure

The host controller may assert data timeout error SDMMC_INT_STATUS_ERR.DATA_TIMEOUT_ERR. As a consequence, unexpected interrupts may be generated.

Workaround

User should avoid sending CMD13 when bus testing is in progress.

SDMMC_CM.002 Unexpected Tx complete interrupt during R1b response

This issue affects both SD and eMMC cards.

R1b is a response type with an optional busy indication on the data line DAT[0]. SD and eMMC cards may send a busy response for the following commands:

Table 11 SD Commands with R1b response

CMD INDEX	Response Type	Abbreviation
CMD12	R1b	STOP_TRANSMISSION
CMD28	R1b	SET_WRITE_PROT
CMD29	R1b	CLR_WRITE_PROT
CMD38	R1b	ERASE

Table 12 eMMC Commands with R1b response

CMD INDEX	Response Type	Abbreviation
CMD5	R1b	SLEEP_AWAKE
CMD6	R1b	SWITCH
CMD12	R1b	STOP_TRANSMISSION
CMD28	R1b	SET_WRITE_PROT
CMD29	R1b	CLR_WRITE_PROT
CMD38	R1b	ERASE

When the card is in busy state for R1b, and driver sends the SEND_STATUS command (CMD13) to read the card status. Due to this CMD13, unexpected transfer complete interrupt SDMMC_INT_STATUS_NORM.TX_COMPLETE may be asserted by the host controller even before the busy signal gets released by the card.

Workaround

User should avoid sending CMD13 while the card is in busy state for R1b.

SDMMC_CM.004 Busy response from card in write resume operation not detected

The SDIO/SD device may support a suspend/resume mode for multi-function SDIO or a combo card. The host may temporarily halt a data transfer operation to one function or memory (suspend) in order to free the bus for a higher priority transfer to a different function or memory.The host controler can resume the write two cycles after the response end bit from Card. However, the controller fails to detect this and starts driving data to the card resulting in a collision.

Workaround

None.

SDMMC_CM.005 Controller sends other command when Auto CMD12 enabled

As per the SD physical layer specification, the NRC (timing between end bit of response to the start bit of next command) clock cycle should be 8 clocks. This applies for auto commands also. However, the controller sends the second command before 8 clocks which is a violation of the specification. The device may not sample the command correctly. This issue affects both SD and eMMC cards.

Workaround

Set SDMMC_TRANSFER_MODE.ACMD_EN = 00_B to disable auto command mode.

SDMMC_CM.006 Stream write issue due to wrong FIFO handling causes data corruption in eMMC mode

During stream write operation for eMMC cards the controller will stop the SD clock when the write FIFOs are empty. The related logic does not handle the two clocks turnaround cycles between internal finite state machine (FSM) and

the top IO. As a consequence the last two bits of the previous set of data are not driven out on the DATA line.

This issue affects eMMC cards only.

Workaround

None.

SDMMC_CM.007 Consecutive write to the same register in SD clock domain

If the software driver programs the same register as back to back write (two consecutive write) before previous one is synchronized, the register does not sync into the destination domain. Any registers that are synchronized to SD clock domain as control information will be affected if written multiple times to set or reset multiple bits in the same register. The following registers are synchronized to the SD clock (f_{SDMMC}).

BLOCK_SIZE

BLOCK_COUNT

ARGUMENT1

TRANSFER_MODE

COMMAND

HOST_CTRL

BLOCK_GAP_CTRL

Workaround

Ensure that the AHB clock frequency (f_{PERIPH}) is not greater than 4 times the SD clock (f_{SDMMC}) or program the software driver in way that two consecutive write to the listed register is not used.

SDMMC_CM.008 Receive state machine hangs if driver programs stop at block gap request using CMD18 when receive buffers are full

In case of CMD18 (read multiple blocks), data blocks are received in the read FIFO which will then be transferred to the data SRAM of the system memory. The host controller will stop the clock supplied to SD/MMC card (SDMMC.CLK_OUT) and waits until one block is transferred to the memory, to delay the next block from the card. During this situation if the host driver sets “stop at block gap request”, the host controller receiver state machine hangs and it never comes of out a particular state. Block gap event is also not generated by the host controller.

This issue affects both SD and eMMC cards.

Workaround

User shall use CMD17 (read single block) or issue a software reset and re-initialize the SDMMC module in case of CMD18 is used.

SDMMC_CM.009 Latching current value in the response register

When the peripheral clock frequency is less than half of SDMMC clock ($f_{PERIPH} < f_{SDMMC}/2$) then the controller fails to store the response contents correctly in to the response register. Hence during resume operation, the driver will not get the expected bits (DF Flag) set in the response register for the resume command and the transaction will fail.

Workaround

Ensure that f_{PERIPH} is greater than $f_{SDMMC}/2$.

USB_CM.002 GAHBCFG.GlblIntrMsk not cleared with a software reset

When the application issues a software reset to the core through the GRSTCTL.CSftRst bit, the GAHBCFG.GlblIntrMsk bit is not reset to 0.

Therefore, an interrupt will be generated in case any of the individual interrupt mask bit (in GINTMSK) is unmasked after the software reset by the application.

Workaround

The workaround is to clear GAHBCFG.GlblIntrMsk to 0 immediately after GRSTCTL.CSftRst is programmed for software reset.

USB_CM.003 Endpoint NAK not sent in Device Class applications with multiple endpoints enabled

In device descriptor DMA mode, the USB 2.0 OTG core does not send NAK handshake for all OUT endpoints once the transfer is complete.

This can be a problem for an application with high latency if it cannot re-enable the endpoint after the transfer is completed on the OUT endpoint.

If the host sends further OUT tokens when the endpoint is disabled, this packet blocks the RxFIFO till the application re-enables the endpoint to read out the packet. Blocking the RxFIFO results in all the other OUT endpoints not receiving any further data. Eventually, the RxFIFO becomes full.

Implications

The bug affects Communication Device Class (AMC) applications (e.g. Ethernet over USB) where multiple endpoints are enabled. When using the recommended Infineon USB device software stacks, the issue will be handled and no further workaround is needed.

Workaround

The application needs to set MTRF=1 for the OUT endpoints. This ensures that the OUT endpoints do not get disabled and hence the RxFIFO blocking limitation is not seen.

When MTRF=1, in order to ensure that there is no BNA (Buffer Not Available) scenario, the application needs to set a long descriptor chain for the OUT endpoints. When MTRF=1, the OUT EP is not disabled and the application and the core share the same descriptor chain simultaneously.

USB_CM.004 USB core is not able to detect resume or new session request after PHY clock is stopped

The control bit PCGCCTL.StopPclk is intended for the application to stop the PHY clock when USB is suspended, the session is not valid, or the device is disconnected.

However, in the current implementation, it also disables wrongly the logic to detect the USB resume and Session Request Protocol (for USB core with OTG capability) signalling.

Implications

If the PHY clock is stopped by setting the bit StopPclk to 1 following a USB suspend or session end, the USB core is not able to detect resume or new session request. Detection is possible again only after the clock gating is removed by clearing the bit StopPclk to 0.

Workaround

The PHY clock must not be stopped with the bit StopPclk for the cases where the application relies on the detection of resume or new session request to remove the clock gating.

USIC_AI.005 Only 7 data bits are generated in IIC mode when TBUF is loaded in SDA hold time

When the delay time counter is used to delay the data line SDA (HDEL > 0), and the empty transmit buffer TBUF was loaded between the end of the acknowledge bit and the expiration of programmed delay time HDEL, only 7 data bits are transmitted.

With setting HDEL=0 the delay time will be t_{HDEL} = 4 x $1/f_{SYS}$ + delay (approximately 60ns @ 80MHz).

Workaround

- Do not use the delay time counter, i.e use only HDEL=0 (default), or
- write TBUF before the end of the last transmission (end of the acknowledge bit) is reached.

USIC_AI.006 Dual SPI format not supported

Dual SPI format is not supported in SSC mode. Therefore, user should always configure either the standard SPI or Quad SPI format in this mode.

Workaround

None.

USIC_AI.007 Protocol-related argument and error bits in register RBUFSR contain incorrect values following a received data word

The protocol-related argument and error bits (PAR and PERR respectively) in register RBUFSR contain incorrect values following a received data word. This leads to the following errors:

Table 13

Protocol	Error due to incorrect PAR and PERR values
ASC	• Received parity bit (RBUFSR.PAR) and result of the parity check (RBUFSR.PERR) are incorrect. • When a data word is received, an alternate receive event (PSR.AIF) may be indicated instead of a receive event (PSR.RIF) even though parity mode is disabled.
SSC	• Received parity bit (RBUFSR.PAR) and result of parity check (PSR.PAERR) are incorrect. • The first data word of a frame may be indicated by a receive event (PSR.RIF) instead of an alternate receive event (PSR.AIF). Similarly, a data word that is not the first word of a frame may be indicated by PSR.AIF instead of PSR.RIF.
IIC	• Received acknowledge bit in RBUFSR.PAR is incorrect. • The first data word of a frame may be indicated by a receive event (PSR.RIF) instead of an alternate receive event (PSR.AIF). Similarly, a data word that is not the first word of a frame may be indicated by PSR.AIF instead of PSR.RIF.
IIS	• Sampling of condition WA = 1 may be indicated by a receive event (PSR.RIF) instead of an alternate receive event (PSR.AIF). Similarly, sampling of condition WA = 0 may be indicated by PSR.AIF instead of PSR.RIF.

Workaround

The workarounds are summarized below:

Table 14

Protocol	Workaround
ASC	• Parity mode cannot be used. • To check if a data word is received, both PSR.RIF and PSR.AIF flags need to be monitored. If interrupts are used, interrupt service handlers need to be set up for both interrupt sources.
SSC	• Parity mode cannot be used. • To check if a data word is received, both PSR.RIF and PSR.AIF flags need to be monitored. If interrupts are used, interrupt service handlers need to be set up for both interrupt sources. • To check if a data word is the first data word of a frame, the bit RBUFSR.SOF can be used.
IIC	• To check for the acknowledge bit, bit 8 of the receive buffer RBUF can be used. • To check if a data word is received, both PSR.RIF and PSR.AIF flags need to be monitored. If interrupts are used, interrupt service handlers need to be set up for both interrupt sources. • To check if a data word is the first data word of a frame, bit 9 of RBUF can be used.
IIS	• To check if a data word is received, both PSR.RIF and PSR.AIF flags need to be monitored. If interrupts are used, interrupt service handlers need to be set up for both interrupt sources. • To check the sampled value of WA, the bit PSR.WA can be used.

USIC_AI.008 SSC delay compensation feature cannot be used

SSC master mode and complete closed loop delay compensation cannot be used. The bit DX1CR.DCEN should always be written with zero to disable the delay compensation.

Workaround

None.

USIC_AI.009 Baud rate generator interrupt cannot be used

The baud rate generator interrupt cannot be used. The bit CCR.BRGIEN must always be written with zero to disable baud rate generator interrupt generation.

Workaround

None.

USIC_AI.010 Minimum and maximum supported word and frame length in multi-IO SSC modes

The minimum and maximum supported word and frame length in multi-IO SSC modes are shown in the table below:

Table 15

Multi-IO SSC Modes	Word Length (bits)		Frame Length (bits)	
	Minimum	Maximum	Minimum	Maximum
Dual-SSC	4	16	4	64
Quad-SSC	8	16	8	64

Workaround

If a frame length greater than 64 data bits is required, the generation of the master slave select signal by SSC should be disabled by PCR.MSLSEN.

To generate the master slave select signal:

- Configure the same pin (containing the SELOx function) to general purpose output function instead by writing 10000_B to the pin's input/output control register (Pn_IOCRx.PCy); and
- Use software to control the output level to emulate the master slave select signal

This way, multiple frames of 64 data bits can be made to appear as a single much larger frame.

USIC_AI.011 Write to TBUF01 has no effect

Writing data to Transmit Buffer Input Location 01 (register TBUF01 at offset address 084_H) does not load the data to the transmit buffer (TBUF).

Workaround

Use registers TBUF00 or TBUFx (x = 02 to 31) to load data to the transmit buffer.

If the Transmit Control Information (TCI) value of 00001_B needs to be generated together with the load to TBUF, use a FIFO setup and Transmit FIFO Buffer Input location 01 (register IN01 at offset address 184_H) instead.

USIC_AI.013 SCTR register bit fields DSM and HPCDIR are not shadowed with start of data word transfer

The bit fields DSM and HPCDIR in register SCTR are not shadowed with the start of a data word transfer.

Workaround

If the transfer parameters controlled by these bit fields need to be changed for the next data word, they should be updated only after the current data word transfer is completed, as indicated by the transmit shift interrupt PSR.TSIF.

USIC_AI.014 No serial transfer possible while running capture mode timer

When the capture mode timer of the baud rate generator is enabled (BRG.TMEN = 1) to perform timing measurements, no serial transmission or reception can take place.

Workaround

None.

USIC_AI.015 Wrong generation of FIFO standard transmit/receive buffer events when TBCTR.STBTEN/RBCTR.SRBTEN = 1

Transmit FIFO buffer modes selected by TBCTR.STBTEN = 1 generates a standard transmit buffer event whenever TBUF is loaded with the FIFO data or there is a write to INxx register, except when TRBSR.TBFLVL = TBCTR.LIMIT. This is independent of TBCTR.LOF setting.

Similarly, receive FIFO buffer modes selected by RBCTR.SRBTEN = 1 generates a standard receive buffer event whenever data is read out from FIFO or received into the FIFO, except when TRBSR.RBFLVL = RBCTR.LIMIT. This is independent of RBCTR.LOF setting.

Both cases result in the wrong generation of the standard transmit and receive buffer events and interrupts, if interrupts are enabled.

Workaround

Use only the modes with TBCTR.STBTEN and RBCTR.SRBTEN = 0.

USIC_AI.016 Transmit parameters are updated during FIFO buffer bypass

Transmit Control Information (TCI) can be transferred from the bypass structure to the USIC channel when a bypass data is loaded into TBUF. Depending on the setting of TCSR register bit fields, different transmit parameters are updated by TCI:

- When SELMD = 1, PCR.CTR[20:16] is updated by BYPCR.SELO (applicable only in SSC mode)
- When WLEMD = 1, SCTR.WLE and TCSR.EOF are updated by BYPCR.BWLE
- When FLEMD = 1, SCTR.FLE[4:0] is updated by BYPCR.BWLE
- When HPCMD = 1, SCTR.HPCDIR and SCTR.DSM are updated by BHPC
- When all of the xxMD bits are 0, no transmit parameters will be updated

However in the current device, independent of the xxMD bits setting, the following are always updated by the TCI generated by the bypass structure, when TBUF is loaded with a bypass data:

- WLE, HPCDIR and DSM bits in SCTR register

- EOF and SOF bits in TCSR register
- PCR.CTR[20:16] (applicable only in SSC mode)

Workaround

The application must take into consideration the above behaviour when using FIFO buffer bypass.

USIC_AI.018 Clearing PSR.MSLS bit immediately deasserts the SELOx output signal

In SSC master mode, the transmission of a data frame can be stopped explicitly by clearing bit PSR.MSLS, which is achieved by writing a 1 to the related bit position in register PSCR.

This write action immediately clears bit PSR.MSLS and will deassert the slave select output signal SELOx after finishing a currently running word transfer and respecting the slave select trailing delay (T_{td}) and next-frame delay (T_{nf}).

However in the current implementation, the running word transfer will also be immediately stopped and the SELOx deasserted following the slave select delays.

If the write to register PSCR occurs during the duration of the slave select leading delay (T_{ld}) before the start of a new word transmission, no data will be transmitted and the SELOx gets deasserted following T_{td} and T_{nf}.

Workaround

There are two possible workarounds:

- Use alternative end-of-frame control mechanisms, for example, end-of-frame indication with TSCR.EOF bit.
- Check that any running word transfer is completed (PSR.TSIF flag = 1) before clearing bit PSR.MSLS.

USIC_AI.020 Handling unused DOUT lines in multi-IO SSC mode

In multi-IO SSC mode, when the number of DOUT lines enabled through the bit field CCR.HPCEN is greater than the number of DOUT lines used as defined in the bit field SCTR.DSM, the unused DOUT lines output incorrect values instead of the passive data level defined by SCTR.PDL.

Implications

Unintended edges on the unused DOUT lines.

Workaround

To avoid unintended edges on the unused DOUT lines, it is recommended to use the exact number of DOUT lines as enabled by the hardware controlled interface during a multi-IO data transfer.

3 Application Hints

The errata in this section describe application hints which must be regarded to ensure correct operation under specific application conditions.

ADC_AI.H003 Injected conversion may be performed with sample time of aborted conversion

For specific timing conditions and configuration parameters, a higher prioritized conversion c_i (including a synchronized request from another ADC kernel) in cancel-inject-repeat mode may erroneously be performed with the sample time parameters of the lower prioritized cancelled conversion c_c. This can lead to wrong sample results (depending on the source impedance), and may also shift the starting point of following conversions.

The conditions for this behavior are as follows (all 3 conditions must be met):

1. **Sample Time setting**: injected conversion c_i and cancelled conversion c_c use different sample time settings, i.e. bit fields `STC*` in the corresponding Input Class Registers for c_c and for c_i (`GxICLASS0/1, GLOBICLASS0/1`) are programmed to different values.
2. **Timing condition**: conversion c_i starts during the first f_{ADCI} clock cycle of the sample phase of c_c.
3. **Configuration parameters**: the ratio between the analog clock f_{ADCI} and the arbiter speed is as follows:

   $$N_A > N_D*(N_{AR}+3),$$

   with

   a) N_A = ratio f_{ADC}/f_{ADCI} (N_A = 2 .. 32, as defined in bit field `DIVA`),

   b) N_D = ratio f_{ADC}/f_{ADCD} = number of f_{ADC} clock cycles per arbitration slot (N_D = 1 .. 4, as defined in bit field `DIVD`),

   c) N_{AR} = number of arbitration slots per arbitration round (N_{AR} = 4, 8, 16, or 20, as defined in bit field `GxARBCFG.ARBRND`).

Bit fields `DIVA` and `DIVD` mentioned above are located in register `GLOBCFG`.

As can be seen from the formula above, a problem typically only occurs when the arbiter is running at maximum speed, and a divider $N_A > 7$ is selected to obtain f_{ADCI}.

Recommendation 1

Select the same sample time for injected conversions c_i and potentially cancelled conversions c_c, i.e. program all bit fields `STC*` in the corresponding Input Class Registers for c_c and for c_i (`GxICLASS0/1, GLOBICLASS0/1`) to the same value.

Recommendation 2

Select the parameters in register `GLOBCFG` and `GxARBCFG` according to the following relation:

$N_A \leq N_D*(N_{AR}+3)$.

ADC_AI.H004 Completion of Startup Calibration

Before using the VADC the startup calibration must be completed.

The calibration is started by setting GLOBCFG.SUCAL. The active phase of the calibration is indicated by GxARBCFG.CAL = 1. Completion of the calibration is indicated by GxARBCFG.CAL = 0.

When checking for bit CAL = 1 immediately after setting bit SUCAL, bit CAL might not yet be set by hardware. As a consequence the active calibration phase may not be detected by software.The software may use the following sequence for startup calibration:

1. GLOBCFG.SUCAL = 1
2. Wait for GxARBCFG.CAL = 1
3. Check for GxARBCFG.CAL = 0 before starting a conversion

Make sure that steps 1 and 2 of this sequence are not interrupted to avoid a deadlock situation with waiting for GxARBCFG.CAL = 1.

ADC_AI.H008 Injected conversion with broken wire detection

If a higher prioritized injected conversion c_i (in cancel-inject-repeat mode) using the broken wire detection feature (GxCHCTRy.BWDEN = 1_B) interrupts a lower prioritized conversion c_c before start of the conversion phase of c_c, the following

effects will occur for the injected conversion c_i (independent of the recommendations in ADC_AI.H003):

1. The effective sample time is either doubled, or it is equal to the sample time of the lower prioritized cancelled conversion c_c. This will shift the starting point of following conversions, and may lead to wrong sample results if the sample time for c_c is considerably shorter than the programmed sample time for c_i (depending on the source impedance).
2. The preparation phase for c_i may be skipped, i.e. during the effective sample phase (as described above), the selected channel is sampled without precharging the capacitor network to the level selected for the broken wire detection. Depending on the synchronization between c_i and c_c, this may increase the time until a broken connection is detected.

The interrupted conversion c_c will be correctly restarted after completion of the injected conversion c_i.

Recommendation

Perform injected conversions without enabling the broken wire detection feature, and follow the recommendations given in ADC_AI.H003.

Alternatively, configure the trigger source that includes channels using the broken wire detection feature such that it will not cancel other conversions. This can be achieved by setting the priority of the request source s to the lowest priority (GxARBPR.PRIOs = 00_B), or by setting the conversion start mode to “wait-for-start mode” (GxARBPR.CSMs = 0_B).

ADC_TC.H011 Bit DCMSB in register GLOBCFG

The default setting for bit DCMSB (Double Clock for the MSB Conversion) in register GLOBCFG is 0_B, i.e. one clock cycle for the MSB conversion step is selected.

DCMSB = 1_B is reserved in future documentation and must not be used.

MultiCAN_AI.H005 TxD Pulse upon short disable request

If a CAN disable request is set and then canceled in a very short time (one bit time or less) then a dominant transmit pulse may be generated by MultiCAN module, even if the CAN bus is in the idle state.

Example for setup of the CAN disable request:

`CAN_CLC.DISR` = 1 and then `CAN_CLC.DISR` = 0

Workaround

Set all INIT bits to 1 before requesting module disable.

MultiCAN_AI.H006 Time stamp influenced by resynchronization

The time stamp measurement feature is not based on an absolute time measurement, but on actual CAN bit times which are subject to the CAN resynchronization during CAN bus operation.The time stamp value merely indicates the number of elapsed actual bit times. Those actual bit times can be shorter or longer than nominal bit time length due to the CAN resynchronization events.

Workaround

None.

MultiCAN_AI.H007 Alert Interrupt Behavior in case of Bus-Off

The MultiCAN module shows the following behavior in case of a bus-off status:

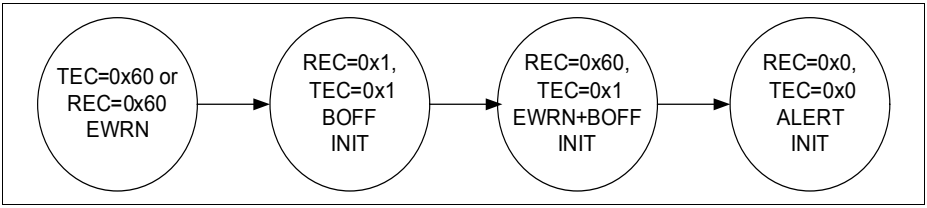

Figure 11 Alert Interrupt Behavior in case of Bus-Off

When the threshold for error warning (EWRN) is reached (default value of Error Warning Level EWRN = 0x60), then the EWRN interrupt is issued. The bus-off (BOFF) status is reached if TEC > 255 according to CAN specification, changing the MultiCAN module with REC and TEC to the same value 0x1, setting the INIT bit to 1_B, and issuing the BOFF interrupt. The bus-off recovery phase starts automatically. Every time an idle time is seen, REC is incremented. If REC = 0x60, a combined status EWRN+BOFF is reached. The corresponding interrupt can also be seen as a pre-warning interrupt, that the bus-off recovery phase will be finished soon. When the bus-off recovery phase has finished (128 times idle time have been seen on the bus), EWRN and BOFF are cleared, the ALERT interrupt bit is set and the INIT bit is still set.

MultiCAN_AI.H008 Effect of CANDIS on SUSACK

When a CAN node is disabled by setting bit NCR.CANDIS = 1_B, the node waits for the bus idle state and then sets bit NSR.SUSACK = 1_B.

However, SUSACK has no effect on applications, as its original intention is to have an indication that the suspend mode of the node is reached during debugging.

MultiCAN_TC.H003 Message may be discarded before transmission in STT mode

If `MOFCRn.STT=1` (Single Transmit Trial enabled), bit TXRQ is cleared (TXRQ=0) as soon as the message object has been selected for transmission and, in case of error, no retransmission takes places.

Therefore, if the error occurs between the selection for transmission and the real start of frame transmission, the message is actually never sent.

Workaround

In case the transmission shall be guaranteed, it is not suitable to use the STT mode. In this case, `MOFCRn.STT` shall be 0.

MultiCAN_TC.H004 Double remote request

Assume the following scenario: A first remote frame (dedicated to a message object) has been received. It performs a transmit setup (TXRQ is set) with clearing NEWDAT. MultiCAN starts to send the receiver message object (data frame), but loses arbitration against a second remote request received by the same message object as the first one (NEWDAT will be set).

When the appropriate message object (data frame) triggered by the first remote frame wins the arbitration, it will be sent out and NEWDAT is not reset. This leads to an additional data frame, that will be sent by this message object (clearing NEWDAT).

There will, however, not be more data frames than there are corresponding remote requests.

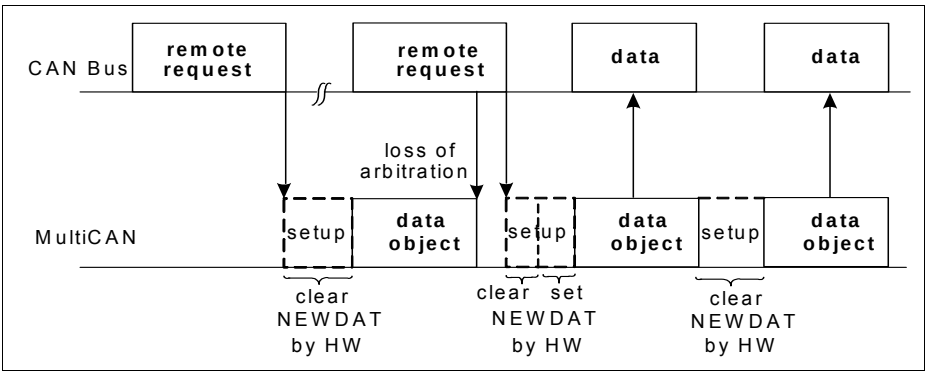

Figure 12 Loss of Arbitration

PORTS_CM.H001 RTC_XTAL pins are swapped

The RTC_XTAL1 and RTC_XTAL2 are swapped in the XMC4300, XMC4700 and XMC4800 device series with respect to the order in the remaining XMC4000 family device series.

Implications

The swap needs to be considered when migrating to or from the XMC4300, XMC4700 or XMC4800 device series from or to other XMC4000 family device series.

RESET_CM.H001 Power-on reset release

The on-chip EVR implements a power validation circuitry which supervises V_{DDP} and V_{DDC}. This circuit releases or asserts the system reset to ensure safe operation. This reset is visible on bidirectional $\overline{\text{PORST}}$ pin. If the $\overline{\text{PORST}}$ release requirement cannot be met due to external circuitry then spikes or toggling on the $\overline{\text{PORST}}$ pin may occur.

Implications

Spikes or repeated PORST assertions may have an effect on the rest of the system if the reset signal is shared with other electronic components on the PCB.

A repeated $\overline{\text{PORST}}$ may also result in loss of information about hibernation status after an interrupted wake-up has been performed.

Recommendation

It is required to ensure a fast rising edge of the $\overline{\text{PORST}}$ signal as specified in section “Power-Up and Supply Monitoring” of the Data Sheet. The recommended approach is to apply a pull-up resistor on the $\overline{\text{PORST}}$ pin.

Typically a 10 - 90 kΩ resistor is sufficient in application cases where the device is in control of the reset generation performed by its internal power validation circuit and no additional load is applied to the $\overline{\text{PORST}}$ pin. The required pull-up resistor value may vary depending on the electrical parameters of the system influencing the signal edges of the $\overline{\text{PORST}}$ signal; for example resistance and capacitance of the PCB and other components connected to the $\overline{\text{PORST}}$ pin.